Интерактивная цифровая книга для профессионалов

Анатолий Постолит

Анализ и прогнозирование данных в веб приложениях на Python и Django



Postolit Press

Постолит А.В.

Анализ и прогнозирование данных в веб приложениях на Python и Django (интерактивная цифровая книга). – М.: Postolit Press, 2025. -615 с.: 294 ил.

Интерактивная цифровая книга предназначена для программистов и аналитиков, имеющих опыт создания приложений на Python и Django. В книге раскрываются возможности некоторых библиотек и API, которые обеспечивают полный цикл обработки открытых данных: загрузка, очистка, анализ, выбор моделей для описания данных, машинное обучение моделей, прогнозирование, представление результатов обработки данных в графическом виде.

Все программные модули данной книги реализованы на Python и Django. Основой программного обеспечения являются веб технологии, так как они обеспечивают работу пользователей с приложением в любое время, в любом месте и с любых устройств. Такой подход позволяет объединить усилия аналитиков, работающих в географически распределенных офисах компаний любого размера.

В книге читатель найдет множество примеров программного кода, который можно не только изучить и скачать, но и выполнить в рамках самой интерактивной цифровой книги, так как она содержит весь необходимый инструментарий для выполнения программных модулей. Все примеры программного кода взяты из модулей, находящихся на публичных серверах, поэтому они гарантировано не содержат ошибок, их можно скачать и использовать в своих проектах.

Книга предназначена для программистов и аналитиков, имеющих опыт работы с **Python и Django**. Она может быть использована как практическое пособие для студентов университетов и слушателей компьютерных курсов при изучении основ анализа временных рядов (Data Analyst, Data Science), и практических приемов программирования **web** приложений.

Оглавление

ПРЕДИСЛОВИЕ 9
Структура книги 10
О примерах программного кода… 12
ГЛАВА 1. БИБЛИОТЕКА PANDAS ДЛЯ АНАЛИЗА И ОБРАБОТКИ ДАННЫХ 14
1.1. Основные объекты библиотеки Pandas 14
1.2. Создание объектов Series в приложении на Python 16
1.3. Создание объектов Series в приложении на Django 18
1.4. Создание объектов DateFrame в приложении на Python 20
1.5. Создание объектов DataFrame в приложении на Django 22
1.6. Работа с данными Pandas формата CSV 23
1.7. Пример работы с данными формата CSV в приложениях на Django 26
1.8. Работа с данными Pandas формата json 28
1.8.1. Пример работы с данными формата JSON в приложениях на Django 29
1.8.2. Работа в Pandas со строковыми объектами данных формата JSON 31
1.9. Индексация и доступ к элементам набора данных в Pandas 33
1.9.1 Выборка данных из DataFrame 34
1.9.2 Фильтрация данных в DataFrame 38
1.9.3 Манипуляции с данными DataFrame 40
1.9.4 Методы статистической агрегации DataFrame 44
1.9.5 Группировка и агрегация в DataFrame 46
1.9.6 Сводная таблица PivotTables 48
1.10. Краткие итоги 50
ГЛАВА 2. ОБЩИЕ ПОНЯТИЯ О ВРЕМЕННЫХ РЯДАХ ..51
2.1. Основные компоненты временного ряда 51
2.2. Статистические параметров оценки временных рядов 54
2.3. Стационарность временных рядов 55
2.4. Методы приведения временных рядов к стационарности 57
2.5. Машинное обучение и модели прогнозирования временных рядов 58
2.6. Краткие итоги 60
ГЛАВА 3. ИСПОЛЬЗОВАНИЕ API ДЛЯ ДОСТУПА К ОТКРЫТЫМ ДАННЫМ 62
3.1. Основные принципы использования API в Python 62
3.2. Доступ к открытым данным из библиотеки Pandas 64
3.2.1. Получение данных о курсах валют из API Центрального банка 64
3.2.2. Получение данных о ценах на драгметаллы из API Центрального банка 67
3.3. Доступ к открытым данным из API Всемирного банка с библиотекой wbdata 70
3.3.1. Модуль Python запроса к данным через API Всемирного банка 70

3.3.2. Запроса к данным через API Всемирного банка в приложениях Django 72
3.3.3. Запроса к данным через API Всемирного банка в приложениях Django с формированием сводной таблицы 75
3.4. Доступ к открытым данным с библиотекой pandas_datareader 78
3.4.1. Модуль Python запроса к данным с библиотекой pandas_datareader 78
3.4.2. Запроса к данным с библиотекой pandas_datareader в приложениях Django 79
3.5. Доступ к данным финансовой платформы Stooq с библиотекой pandas_datareader 81
3.5.1. Запроса к данным финансовой платформы Stooq из модуля Python 81
3.5.2. Запроса к данным финансовой платформы Stooq из приложения на Django 83
3.6. Доступ к данным финансовой платформы Yahoo Finance 85
3.6.1. Запроса к данным финансовой платформы Yahoo Finance из модуля Python 86
3.6.2. Запроса к данным финансовой платформы Yahoo Finance из приложения Django 87
3.7. Краткие итоги 89
ГЛАВА 4. БИБЛИОТЕКИ PYTHON ДЛЯ ВИЗУАЛИЗАЦИИ ДАННЫХ 90
4.1. Визуализация данных с библиотекой Plotly 91
4.2. Визуализация данных с интерфейсом Plotly Express 91
4.2.1. Графики линий с интерфейсом Plotly Express 92
4.2.2. График линий в Plotly Express с группировкой столбцов набора данных 94
4.2.3. Столбчатая диаграмма в Plotly Express 97
4.3. Фасетные сетки для отображения группы графиков в Plotly Express 99
4.3.1. Фасетные сетки для отображения группы графиков методом facet_col 99
4.3.2. Фасетные сетки для отображения группы графиков методом facet_row 102
4.4. Визуализация данных с интерфейсом Plotly Graph Objects 104
4.4.1. График линии с интерфейсом Plotly Graph Objects 105
4.4.2. График нескольких линии с интерфейсом Plotly Graph Objects 107
4.4.3. Сетка графиков с интерфейсом Plotly Graph Objects 109
4.5. Визуализация данных с библиотекой Pandas 113
4.5.1. Визуализация данных в Pandas с библиотекой Matplotlib 114
4.5.2. Визуализация данных в Pandas с библиотекой Plotly 116
4.6. Краткие итоги 118
ГЛАВА 5. ОЧИСТКА ДАННЫХ И ПОДГОТОВКА К ГРАФИЧЕСКОЙ ВИЗУАЛИЗАЦИИ 120
5.1. Модуль Python для загрузки и анализа данных из CSV файла 121
5.2. Модули Django для визуализации данных из CSV файла 123
5.3. Модуль Python для загрузки и анализа данных из файла формата JSON 124
5.4. Модули Django для визуализации данных из JSON файла 126
5.5. Библиотека Pandas для работы со строковыми объектами JSON 128
5.6. Загрузки очистки данных через API Центрального банка России 130
5.6.1. Модуль Python для загрузки очистки данных через API Центрального банка (курсы валют) 130

5.6.2. Модули Django для визуализации данных из API Центрального банка (курсы валют) 132
5.6.3. Модуль Python для загрузки и очистки данных из API Центрального банка (цены на драгметаллы) 134
5.6.5. Модули Django для визуализации данных из API Центрального банка (цены на драгметаллы) 136
5.7. Модуль Python для загрузки и очистки данных через API Московской биржи 137
5.7.1. Модуль Python для загрузки и очистки данных из API Московской биржи (цены на драгметаллы) 138
5.7.2. Модули Django для визуализации данных из API Московской биржи (цены на акции Газпрома) 139
5.8. Модуль Python для загрузки и очистки данных через API Всемирного банка 142
5.8.1. Модуль Python для загрузки и очистки данных из API Всемирного банка с библиотекой wbdata 142
5.8.2. Модули Django для визуализации данных из API Всемирного банка (ВВП стран) 144
5.8.3. Модуль Python для загрузки и очистки данных из API Всемирного банка с библиотекой pandas_datareader 145
5.8.4. Модули Django для визуализации данных из API Всемирного банка (данные об экспорте) 148
5.9. Модуль Python для загрузки и очистки данных из API Stooq 149
5.9.1. Модуль Python для загрузки и очистки данных из API Stooq (котировки акций) с библиотекой pandas_datarider 150
5.9.2. Модули Django для визуализации данных из API Stooq (котировка акций) 151
5.10. Модуль Python для загрузки и очистки данных из API Yahoo Finance 153
5.10.1. Модуль Python для загрузки и очистки данных из API Yahoo Finance (котировка акций) 153
5.10.2. Модули Django для визуализации данных из API Yahoo Finance (котировка акций) 155
5.10.3. Модуль Python для загрузки и очистки данных из API Yahoo Finance (стоимость нефти) 156
5.10.4. Модули Django для визуализации данных из API Yahoo Finance (стоимость нефти) 158
5.11. Краткие итоги 160
ГЛАВА 6. ПОЛУЧЕНИЕ И ОБРАБОТКА ДАННЫХ МЕЖДУНАРОДНОГО ВАЛЮТНОГО ФОНДА 161
6.1. Библиотека imfp 162
6.2. Получение сведений о базах данных МВФ с функцией imf_databases 163
6.3. Получение сведений о параметрах запросов к базам данных МВФ с функцией imf_parameters 165
6.4. Получение сведений об описании параметров запросов к базам данных МВФ с функцией imf_parameter_defs 167
6.5. Загрузка набора данных из баз данных МВФ с использованием функции imf_dataset 170
6.5.1. Загрузка данных из БД МВФ с передачей значений параметрам 170

6.5.2.Загрузка данных из БД МВФ с передачей значений параметров в виде словаря с набором данных..... 173
6.6.Очистка набора данных перед графической визуализацией...175
6.7.Модули Django для визуализации данных о среднегодовой стоимости угля 176
6.8.Модули Django для визуализации данных о среднемесячной стоимости угля 178
6.9.Модули Django для визуализации квартальных данных о стоимости угля 180
6.10.Модули Django для визуализации данных мировых ценах на золото 182
6.11.База данных международной финансовой статистики IFS ...185
6.11.1.База данных международной финансовой статистики IFS (загрузка и очистка данных) 185
6.11.2.Модули Django для визуализации данных об экспорте товаров из БД МВФ (IFS)188
6.12.Общий алгоритм доступа к данным через API IMF с библиотекой requests 190
6.12.1.Модуль Python загрузки и очистки данных из БД IFS с библиотекой requests..... 190
6.12.2.Модули Django для визуализации данных об индексе цен импорта товаров из БД МВФ (IFS) 192
6.12.3.Модуль Python загрузки и очистки данных из БД торговой статистики с библиотекой requests..... 195
6.12.4.Модули Django для визуализации данных об импорте товаров из БД МВФ (DOT) 196
6.13.Получение ключевых показателей API МВФ с библиотекой requests 198
6.13.1.Получение данных о финансово экономических индикаторах..... 199
6.13.2.Получение данных о географических регионах и аналитических группах 200
6.13.3.Получение данных для заданного индикатора финансово экономической деятельности 202
6.14. Получение данных из API МВФ 203
6.14.1. Получение данных из API МВФ с использованием меток параметров..... 203
6.14.2.Получение данных из API IFS с использованием идентификаторов параметров 205
6.14.3.Модули Django для визуализации данных о ВВП стран, полученных из API МВФ на основе меток параметров 207
6.14.4.Модули Django для визуализации данных о ВВП стран, полученных из API МВФ на основе идентификаторов параметров 209
6.15. Краткие итоги 212
ГЛАВА 7. АНАЛИЗ И ОЧИСТКА ДАННЫХ ВРЕМЕННЫХ РЯДОВ.....213
7.1.Методы анализа временных рядов 213
7.2. Декомпозиция временного ряда временного ряда 214
7.3.Определение скользящего среднего временного ряда.....218
7.4.Стационарность временного ряда.....221
7.4.1.Оценка стационарности временного ряда (пассажиропотоки) 221
7.4.2.Оценка стационарности временного ряда (объем продажи нефти)..... 223
7.5.Приведение временных рядов к стационарности.....226

7.5.1.Приведение временных рядов к стационарности путем дифференцирования 227
7.5.2.Приведение временных рядов к стационарности путем двойного дифференцирования 230
7.5.3.Приведение временных рядов к стационарности путем логарифмирования с дифференцированием 232
7.6.Изменение частоты дискретизации временных рядов.....234
7.6.1.Увеличение частоты дискретизации данных с использованием метода asfreq..... 235
7.6.2.Восстановление непрерывности дат временного ряда методом resample..... 239
7.6.3. Уменьшение частоты дискретизации данных с использованием метода resample242
7.7. Краткие итоги 244
ГЛАВА 8. СТАТИСТИЧЕСКИЕ МОДЕЛИ ДЛЯ АНАЛИЗА ВРЕМЕННЫХ РЯДОВ И ПРОГНОЗИРОВАНИЯ 245
8.1. Авторегрессионные модели (AR) для анализа и прогнозирования временных рядов 246
8.1.1. Авторегрессионные модели (AR) для анализа временных рядов 246
8.1.2. Авторегрессионные модели (AR) для прогнозирования временных рядов 251
8.1.3.Преимущества и недостатки авторегрессионных моделей (AR) 253
8.2. Модели скользящего среднего (MA) для анализа и прогнозирования временных рядов 254
8.2.1. Модели скользящего среднего (MA) для анализа временных рядов 255
8.2.2. Модели скользящего среднего (MA) для прогнозирования временных рядов 258
8.2.3. Преимущества и недостатки модели скользящего среднего (MA) 262
8.3. Модели авторегресионного скользящего среднего (ARMA) для анализа и прогнозирования временных рядов.....263
8.3.1. Модели ARMA для анализа временных рядов..... 265
8.3.2. Поиск оптимальных параметров модели ARMA..... 276
8.3.3. Прогнозирование временного ряда с моделью ARMA 281
8.4. Трехкомпонентные модели ARIMA для анализа и прогнозирования временных рядов 285
8.4.1. Анализ временного ряда и оптимизация модели ARIMA методом auto_arima..... 286
8.4.2. Прогнозирование временного ряда с моделью ARIMA 290
8.5. Модель SARIMA для анализа и прогнозирования временных рядов.....296
8.5.1. Предварительный анализ временных рядов на наличие сезонности 297
8.5.2.Анализ временного ряда и оптимизация модели SARIMA методом auto_arima..... 299
8.5.3. Прогнозирование временного ряда с моделью SARIMA 304
8.6. Модель SARIMAX для анализа и прогнозирования временных рядов.....309
8.6.1.Анализ временного ряда и оптимизация модели SARIMAX методом auto_arima ... 311
8.6.2. Прогнозирование временного ряда с моделью SARIMAX..... 318
8.7. Краткие итоги 322
ГЛАВА 9. БИБЛИОТЕКА PROPHET ДЛЯ ПРОГНОЗИРОВАНИЯ ДАННЫХ.....324
9.1. Принципиальные основы библиотеки Prophet 325
9.1.1. Прогнозирование цен на нефть по данным API Yahoo Finance 326
9.1.2. Сохранение обученной модели в Prophet..... 332

9.1.3. Прогнозирование котировок акций по данным API Yahoo Finance 335
9.1.4. Прогнозирование курса валют по данным API Центрального банка 344
9.1.5. Прогнозирование цен на драгоценные металлы по данным API Центрального банка 355
9.1.6. Прогнозирование экономических показателей стран по данным API Всемирного банка 365
9.1.7. Прогнозирование продаж торгового центра 374
9.2. Краткие итоги 379
ПОСЛЕСЛОВИЕ 380
СПИСОК ИСТОЧНИКОВ И ЛИТЕРАТУРЫ 381
ПРИЛОЖЕНИЕ 1. СПИСОК КОДОВ СТРАН API ВСЕМИРНОГО БАНКА 384
ПРИЛОЖЕНИЕ 2. НЕКОТОРЫЕ ИНДИКАТОРЫ ФИНАНСОВЫХ ПОКАЗАТЕЛЕЙ API ВСЕМИРНОГО БАНКА 385
ПРИЛОЖЕНИЕ 3. СПИСОК ИДЕНТИФИКАТОРОВ (ТИКЕРОВ) КОМПАНИЙ API МОСКОВСКОЙ БИРЖИ ... 386
ПРИЛОЖЕНИЕ 4. КОД ПРОГРАММНОГО МОДУЛЯ CLEAR_DATA 387
ПРИЛОЖЕНИЕ 5. КОД ПРОГРАММНОГО МОДУЛЯ CLEAR_DATA_METALL 388
ПРИЛОЖЕНИЕ 6. КОД ПРОГРАММНОГО МОДУЛЯ FN_WB_DATA 389
ПРИЛОЖЕНИЕ 7. СПИСОК ИСПОЛЬЗУЕМЫХ БИБЛИОТЕК С УКАЗАНИЕМ ВЕРСИЙ 390

Предисловие

В настоящее время аналитик данных – это одна самых востребованных специализаций, так как стремительными темпами развивается цифровизация бизнес процессов. Если вы задумываетесь о карьере аналитика, или хотите стать программистом, который разрабатывает приложения для аналитиков данных, то эта книга именно для вас.

Чем же занимается аналитик данных (Data Analyst)? Это специалист, который собирает, структурирует, обрабатывает и анализирует большие массивы информации, для того, чтобы помочь компаниям или государственным структурам принимать обоснованные и стратегически правильные решения. Этот специалист превращает хаотичные наборы данных в понятные выводы и рекомендации, которые обеспечивают успешное развитие компании, отрасли, или даже целой страны. Аналитики данных востребованы в самых разных отраслях: транспорт, торговля, электронная коммерция, банковский сектор, медицина, государственные структуры.

Работа аналитика данных включает несколько последовательных видов деятельности:

- сбор данных из разных источников: баз данных, сайтов, открытых источников через API, или проводимых специальных исследований;
- очистка данных: удаляет ошибки, пропуски, дубликаты, всплески, чтобы данные были пригодны для анализа;
- анализ данных: использует различные инструменты для поиска закономерностей в хаотическом нагромождении цифр;
- визуализация: создает графики, и отчеты, чтобы превратить большие массивы данных в достоверные понятные выводы;
- коммуникация: объясняет обоснованность своих выводов, предлагает решения и отвечает на вопросы руководителей бизнеса или государственных структур.

Однако аналитик данных без специального инструментария мало чего может сделать. А разработкой такого инструментария занимаются программисты, которые работают на стыке программирования, статистики, математики, аналитики и машинного обучения, или другими словами в области - Data Science (Дата-сайенс - наука о данных).

Всем, кто только выбирает специализацию в сфере IT, или кто хочет сменить направление деятельности стоит обратить на Дата-сайенс самое пристальное внимание. Специалисты в этой нише входят в число сотрудников с самыми высокими зарплатами. В ближайшее десятилетие, даже с учетом нестабильной экономической ситуации, спрос на профессионалов Data Science во всем мире будет только расти.

Сложность профессий, связанных с Data Science, в том, что эта сфера стремительно прогрессирует и все время развивается. Специалистам приходится постоянно актуализировать свои знания и осваивать смежные области. При этом глубокое знание языков программирования и умение писать программный код остаются базой для каждого сотрудника этого направления.

В сфере Data Science особенно востребованы специалисты, которые имеют практические навыки реализации веб приложений. Ведь крупные банки, торговые, транспортные и консалтинговые компании имеют сотни филиалов в различных городах с тысячами сотрудников. Устанавливать, обновлять и сопровождать работу программного обеспечения

на рабочих местах сотрудников распределенных офисах либо очень дорого, либо просто невозможно. Применение же веб технологий позволяет сконцентрировать программно-вычислительные мощности в одном месте, при этом все сотрудники будут иметь доступ к единым данным и к единому программному обеспечению, при этом расходы на сопровождение информационного комплекса будут минимальны.

На российском рынке труда количество вакансий в этой сфере в несколько раз превышает число опубликованных резюме. В этом сегменте востребованы специалисты любых возрастов. Для людей с техническим складом ума и опытом работы в IT- сфере это отличная возможность расширить свои компетенции или освоить новую профессию с нуля.

Эта книга предназначена не для начинающих программистов. Она нацелена на специалистов с опытом, которые хорошо знают Python, кто уже занимался разработкой веб-приложений с использованием Django. Она может быть интересна и практикующим аналитикам, которые хотят глубже окунуться в мир программирования и создавать необходимый инструментарий без посредников.

Сразу следует отметить, что веб-программирование требует от разработчиков бо̲льших знаний, умений и усилий, чем программирование традиционных приложений. Здесь, кроме основного языка, который реализует логику приложения, необходимо знание структуры HTML-документов, основ работы с базами данных, принципов взаимодействия удаленных пользователей с веб-приложением.

Книга, как уже отмечено, предназначена для программистов, которые разрабатывают приложения для анализа данных, и для аналитиков, которые стремятся расширить свои компетенции в области программирования. Она может быть полезна и математикам, которые разрабатывают статистические методы и модели для анализа данных. Ведь после теоретических изысканий нужно проверить, как теория, основанная на сложных формулах, проявит себя после практической реализации.

В книге описаны все этапы, связанные с разработкой веб приложений для обработки данных: сбор, очистка, анализ и визуализация. На максимально упрощенном уровне рассмотрены теоретические аспекты, связанные с моделями прогнозирования временных рядов. Каждый раздел заканчивается примером с детальным разбором программного кода. Все примеры могут быть выполнены в рамках самой цифровой интерактивной книги, так как в нее встроен весь необходимый инструментарий. Листинги программ могут быть скопированы и использованы при разработке собственных приложений.

Структура книги

Первая глава книги посвящена изучению функций и методов библиотеки Pandas. Это основной инструмент, который используется при разработке любых приложений, связанных с обработкой и анализом данных. В ней рассказываются сведения о наиболее важных составляющих данной библиотеки. Эти материалы станут основой для разработки абсолютно всех программных модулей, обеспечивающих загрузку данных из открытых источников, их очистку анализ и визуализацию. Модули библиотеки Pandas используются на всех этапах работы с данными, поэтому уметь работать с этой библиотекой должен каждый программист и аналитик.

Во *второй главе* книги описываются характеристики и особенности временных рядов: тренд, сезонность, цикличность, случайность. Кроме того, рассказано о статистических

параметрах оценки временных радов, о их стационарности и моделях прогнозирования. В этой главе приводятся только вводные базовые понятия о временных рядах без разбора сложных математических методов, лежащих в основе прогнозирования.

В *третьей главе* представлены технологии доступа к открытым источникам данных через API, ведь для того, чтобы приступить к анализу данных, их сначала нужно получить. В качестве примеров рассмотрены возможности загрузки данных через API Центрального банка России, Всемирного банка, финансовых платформ Stooq и Yahoo Finance. На примерах показано, как можно загружать удаленные данные с использованием библиотек requests, wbdata, pandas_datareader, yfinance.

Четвертая глава посвящена визуализации данных. Визуализация данных является важнейшим инструментом в Data Science, поскольку позволяет наглядно показать структуру и динамику изменения больших массивов данных. Здесь рассмотрены основы визуализации данных с библиотеками Pandas, Plotly и Matplotlib, приведены примеры создания простейших графиков. Материалы данной главы не претендуют на полноту описания всех возможностей этих библиотек, для этого нужно обращаться к специальной литературе или к оригинальной документации. Но полученных сведений вполне достаточно для понимания примеров визуализации, которые будут приведены в последующих главах. Самое важное, что необходимо усвоить из материалов данной главы – это технология отображения графиков в шаблонах Django.

П*ятая глава* содержит примеры предварительной «очистки» данных, полученных из различных источников. Под «очисткой» здесь понимается предварительный анализ структуры и типов данных, их подготовка к графической визуализации. Данные, полученные из внешних источников, имеют разную структуру, числа и даты могут быть представлены в текстовом виде, что не позволить корректно показать данные на графиках. Исходя из этого в этой главе на примерах показано, каким образом «сырые» данные можно трансформировать в типовой набор DataFrame или Series, на основе которых можно строит графики.

В *шестой главе* приведены примеры доступа к данным международного валютного фонда (МВФ). Дело в том, что МВФ имеет сотни баз данных, открытых для общего доступа. Это сведения об индикаторах экономического развития стран, о товарных и природных запасах, торговли между странами, мировыми ценами на сырье и товары, курсы валют, статистика труда, численность населения и т.п. Эти данные очень важны для государственных и коммерческих структур для анализа стратегий развития той или иной отрасли, эффективности операций импорта – экспорта, демографической ситуации и других важных экономических показателей. Каждая база данных имеет свой набор параметров запросов к данным, которые можно получить через API МВФ. Исходя из этого описание методов загрузки и «очистки» данных, полученных через API МВФ, выделено в отдельную главу.

В седьмой главе приводятся некоторые полезные советы, связанные с анализом и вторичной «очисткой» данных временных рядов. Здесь под анализом данных понимается выявление характеристик и особенностей временных рядов, которые нужно учитывать при решении задач прогнозирования. Кроме того в данной главе описаны методы структурного анализа временных рядов, которые позволяют выявить их полноту и необходимость вторичной «очистки», которая включает добавление пропущенных значений, удаление не характерных выбросов, повышение или понижение частоты дискретизации данных. Очищенные таким образом данные обеспечивают получение более точных прогнозов.

В восьмой главе представлены материалы, связанные анализом и прогнозированием временных рядов с использованием различных статистических моделей и машинного

обучения. Рассмотрены примеры прогнозирования на моделях AR, MA, ARMA, ARIMA, SARIMA и SARIMAX. Все примеры главы завершены графическим представлением результатов прогнозирования, сделан разбор функциональной сущности наиболее значимых фрагментов программного кода и графиков, построенных с использованием библиотек Pandss, Plotly и Matplotlib.

Девятая глава посвящена использованию библиотеки Prophet для прогнозирования временных рядов, с выводом результатов прогноза в интерактивные диаграммы. Библиотека Prophet хороша тем, что позволяет строить достоверные прогнозы даже в условиях ограниченного объема данных, сезонных колебаний и нестабильных трендов. Ее применение может быть полезным для начинающих программистов и аналитиков, даже не имеющих глубоких знаний структуры и особенностей временных рядов. Библиотека Prophet дает возможность минимизировать размер программных модулей и не требует от программистов и аналитиков глубоких знаний структуры используемых моделей.

О примерах программного кода…

На протяжении всей книги раскрываемые вопросы сопровождаются по возможности упрощенными, но полностью законченными примерами. Ход решения той или иной задачи показан на большом количестве иллюстративного материала. Желательно изучение тех или иных разделов осуществлять, сидя непосредственно за компьютером, — тогда вы сможете последовательно повторять выполнение тех шагов, которые описаны в примерах, и тут же видеть результаты своих действий, что в значительной степени облегчит восприятие материала книги.

Некоторые фрагменты программного кода этой книги могут показаться сложными, но они взяты из реально работающих программ, развернутых на публичных серверах. В них гарантированно нет ошибок и использованы разработанные сообществом действующих программистов приемы, о которых ничего не сказано в официальной документации. Абсолютно все примеры, сопровождающие интерактивную книгу, можно запустить на выполнение в рамках самой книги, так как она содержит весь необходимый инструментарий. Когда вы увидите результаты работы программного кода, появится ясность — как работает та или иная статистическая модель или фрагмент программного кода. Мало того зарегистрированные пользователи могут скачать листинги программ и использовать их в своих проектах.

> ***Интерактивная цифровая платформа***
> Работать с интерактивной цифровой платформой можно, зайдя в нее по адресу **rumnik.ru**. Демо-версию книги можно найти в опции **Каталог** главного меню. Полная версия интерактивной цифровой книги будет доступна в разделе **Мои книги** личного кабинета зарегистрированного пользователя после ее приобретения в магазине сайта.
>
> Использование полной версии цифровой интерактивной книги, если это покажется удобным, избавит вас от необходимости приобретения ее «бумажной» версии.

Наилучший способ обучения — это практика. Все листинги программ в данной книге реализованы на языке Python, а шаблоны веб-страниц с использованием HTML и тегов Bootstrap. Любой пример этой книги, выставленной на интерактивную образовательную платформу **rumnik.ru**, можно запустить на выполнение и посмотреть, как выглядит конечный результат.

Примечание.
К интерактивной цифровой книге подключен сервис, который позволяет переводить и содержимое страниц на более чем 100 языков мира, с возможностью озвучивания на выбранном языке. Это может быть полезно при обучении студентов – иностранцев, так как они могут читать и слушать содержимое страниц на русском, и на родном языке.

Итак, если вас заинтересовали вопросы создания эффективных веб-приложений, связанных с анализом и прогнозированием данных то самое время перейти к изучению материалов этой книги на цифровой интерактивной платформе.

Примечание.
В демо-версии книги листинги программ удалены.

Глава 1. Библиотека Pandas для анализа и обработки данных

Название библиотеки pandas никак не связано с бамбуковым медведем - пандой, это просто сокращение английского термина panel data (панельные данные). Данная библиотека имеет набор методов и функций, которые позволяют использовать готовые и протестированные алгоритмы обработки данных, размещенных не хаотично, а в таблицах. Такие данные еще называют «панельными».

Библиотека находит свое применение и для решения следующих задач:

- обработка и анализ финансовых данных (цены акций, объемы торгов, финансовые индикаторы и т.д.);
- прогнозирование тенденций на финансовых рынках;
- обработка экономической информации;
- создание и тестирование статистических моделей анализа данных при машинном обучении.
- визуализация данных.

Из материалов этой главы вы узнаете:

- об основных объектах библиотеки Pandas;
- как можно создать наборы данных типа Series и DataFrame;
- как можно обрабатывать данные формата CSV;
- как можно обрабатывать данные формата json;
- как можно получить доступ к элементам наборов данных;
- как обеспечить фильтрацию данных;
- какие манипуляции можно выполнять с набором данных, включая группировку и агрегирование;
- как можно преобразовывать наборы данных в сводные таблицы.

Конечно, в рамках одной главы невозможно раскрыть все возможности библиотеки Pandas, однако, этого материала будет вполне достаточно для того, чтобы использовать функции и методы данной библиотеки для решения практических задач.

1.1. Основные объекты библиотеки Pandas

Библиотека Pandas предоставляет удобные структуры данных для их обработки и анализа. Данные в Pandas похожи на данные в Excel в том смысле, что оба инструмента позволяют удобно работать с данными в табличной форме. С помощью Pandas можно выполнять множество операций над данными, аналогичные тем, которые можно выполнить в Excel, такие как фильтрация, сортировка, объединение таблиц, вычисление статистик и т.д. Однако Pandas предоставляет более мощные и гибкие возможности для работы с данными в Python, чем Excel.

Данные в Pandas могут быть представлены в объектах двух типов: DataFrame — соответствует таблице из нескольких столбцов, Series — соответствует таблице из одного одному столбца (рис 1.1).

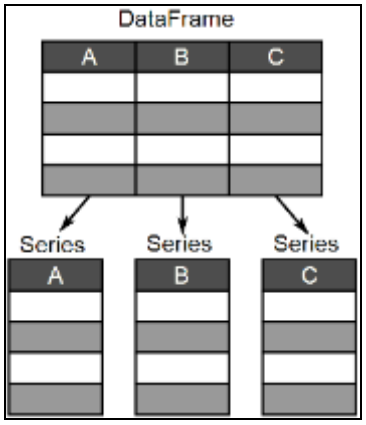


Рис. 1.1. Объекты библиотеки Pandas

Кроме столбцов с данными объекты Pandas содержат еще одни специфичный столбец, который называется индексным (index).

Рассмотрим сущность этих трех ключевых объектов.

1. **DataFrame** — это основной объект, используемый для представления табличных данных в Pandas, который представляет двумерные табличные данные в формате столбцы и строки. DataFrame содержит столбцы, каждый из которых представляет собой серию (Series), и индекс, который может быть числовым или нечисловым.
2. **Series** — это одномерный массив данных в Pandas, который может быть использован как столбец в DataFrame. Series также имеет индекс.
3. **Index** — представляет собой структуру данных, которая используется для индексации осей (строк и столбцов) в объектах DataFrame и Series. Index в Pandas может быть одномерным или двумерным и обеспечивает быстрый доступ к данным по меткам.

Библиотека Pandas построена на основе библиотеки NumPy и расширяет ее функциональность, предоставляя удобные структуры данных для работы с табличными данными. Поэтому в Pandas достаточно часто используются методы NumPy. В Pandas можно использовать файлы «.npy» для хранения данных.

В Pandas есть зарезервированные имена осей, которые используются для обращения к строкам и столбцам в DataFrame.

1. index — это зарезервированное имя для обращения к строкам (нулевая ось) в DataFrame. Можно использовать метод df.index для доступа к индексу DataFrame.
2. columns — это зарезервированное имя для обращения к столбцам (первая ось) в DataFrame. Можно использовать метод df.columns для доступа к столбцам DataFrame.
3. axis— это параметр, который может принимать значения 0 или 1 и используется для указания оси операций, таких как суммирование, группировка, и т.д. Можно использовать значение axis=0 для операций по строкам, и axis=1 для операций по столбцам.

Эти зарезервированные имена помогают упростить обращение к строкам и столбцам в наборе данных DataFrame.

В наборе данных Pandas можно хранить данные различных типов:

- object— общий тип данных, который может хранить любой Python объект;
- int64— целочисленный тип данных для хранения целых чисел;
- float64— тип данных для хранения чисел с плавающей точкой;
- bool— логический тип данных для хранения значений True или False;
- datetime64— тип данных для хранения даты и времени;
- timedelta— тип данных для хранения разницы между двумя датами или временем;
- category— тип данных для хранения категориальных данных, которые могут принимать ограниченное число уникальных значений.

Тип данных определяется отдельно для каждого столбца. Ячейки с отсутствующими данными помечаются как NaN (тип данных в этом столбце автоматически станет float).

Если в столбце есть хоть одна ячейка со строкой, то тип данных в столбце станет object. Тип данных можно в таком столбце изменить на category. Столбец с датами в виде строк и типом данных object можно изменить на тип данных datetime64. Отсутствующее значение в таком столбце будет обозначено как NaT.

В Pandas для представления отсутствующих данных в различных контекстах используются два специальных значения NaN и NaT.

NaN (Not a Number) — это значение, которое обычно используется для представления отсутствующих числовых данных или данных с плавающей точкой. Обычно NaN возникает в результате математических операций, где значение не может быть определено, например, деление на ноль или невозможность провести вычисление. Обозначение NaN может быть использовано в Pandas для представления отсутствующих числовых значений.

NaT (Not a Time) — это значение, которое используется для представления отсутствующих временных данных или значений даты и времени. Обычно NaT возникает при работе с временными данными в Pandas, когда значение даты или времени не определено или неизвестно. Обозначение NaT может быть использовано в Pandas для представления отсутствующих временных значений.

Существует еще значение NA— это обобщенное специальное значение, которое используется в Pandas для представления отсутствующих данных независимо от типа данных (числовые, временные, объекты и т.д.). Обозначение NA является универсальным значением в Pandas, которое может быть использовано для представления отсутствующих значений в любом контексте (NA более общее обозначение, чем NaN или NaT).

Для получения информации о типах данных в объектах Pandas используется метод df.dtypes. Установить библиотеку Pandas можно с использованием команды pip:

pip install pandas

1.2. Создание объектов Series в приложении на Python

Конструктор класса Series в Pandas принимает несколько аргументов для создания объекта Series и имеет следующий синтаксис:

```
s = pd.Series(data=dtype, index=index, dtype=dtype, name=name, copy=False, fastpath=False)
```

где:

- data: это данные, которые могут быть массивом, списком, словарем, кортежем или другими структурами данных;
- index (необязательный параметр): это индексы, которые могут быть заданы в виде массива, списка, диапазона и т.д. (если индекс не задан, то по умолчанию будет использоваться целочисленный индекс);
- dtype (необязательный параметр): это тип данных, который определяет тип значений в Series (например, int, float, str, datetime и т.д.);
- name (необязательный параметр): это имя Series, которое может быть использовано для обращения к Series в DataFrame.
- copy (необязательный параметр): определяет, должен ли быть скопирован переданный в качестве данных объект, или он будет использоваться напрямую;
- fastpath: параметр указывающий, являются ли данные однородным массивом (по умолчанию False).

Параметр copy по умолчанию имеет значение False, что означает, что данные не будут скопированы, а будут использованы напрямую. Если параметр copy установлен в True, то данные будут скопированы.

Если при создании Series мы передаем index, то количество индексов должно соответствовать количеству элементов во множестве данных data.

Рассмотрим пример на Python, в котором создаются два объекта Series из списка и из словаря (листинг 1.1)

Листинг 1.1. Создание объектов Series библиотеки Pandas (модуль my_series.py)

Примечание.
В демо-версии книги листинги программ удалены.

В этом программном модуле сначала заданы два списка: data – данные, index – индексный или ключевой столбец. Затем из этих списков создан набор данных Series (объект s_list). При формировании данного объекта использовались не именованные параметры (они находятся на месте параметров data и index). Далее в окно терминала выведен весь набор данных и его отдельные элементы.

После этого создан словарь с данными dic и index – индексный или ключевой столбец. Затем из этих объектов создан набор данных Series (объект s_dic). При формировании данного объекта использовались именованные параметры (data=dic, index=index). Далее в окно терминала выведен весь набор данных и его отдельные элементы.

Если выполнить данный программный код, то будет получен следующий результат.

```
Создание Series из списка
A   100
B   300
C   800
D   500
```

```
E    400
dtype: int64
B= 300
C= 800
Срез C-E
C    800
D    500
E    400
dtype: int64
-------------------------------
Создание Series из словаря
A    10.1
B    20.2
C    30.3
D    40.4
E    50.5
dtype: float64
A    10.1
D    40.4
E    50.5
dtype: float64
20.2
```

Примечание.
Представленный в данном разделе программный модуль не включен в список исполняемых примеров интерактивной цифровой книги, так как он может работать только в рамках приложения Python локального компьютера, или быть вызванным из другого модуля веб приложения.

Как видно из листинга программы и из результатов ее работы, доступ к элементам набора данные Series можно получить, как по имени ключевого столбца (index), так и по номеру строки, или номерам строк.

1.3. Создание объектов Series в приложении на Django

А теперь покажем, как можно встроить программный модуль, о котором говорилось в предыдущем разделе, в web приложение и вывести созданные объекты Series в шаблон Django. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 1.2).

Листинг 1.2. Ссылка активации маршрута made_series

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем made_series (листинг 1.3).

Листинг 1.3. Маршрут URL с именем made_series

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление made_series (листинг 1.4).

Листинг 1.4. Представление (view) made_series

Примечание.
В демо-версии книги листинги программ удалены.

В первой строке этого программного кода выполняется импорт модуля my_series.py (см. предыдущий раздел). После этого выполнено обращение к функции m_series модуля my_series, где формируются объекты Series (они сохраняются в объектных переменных s1 s2). Содержимое s1 и s2 можно было бы сразу вывести в шаблон Django, но они там будут представлены в виде текстовых строк, а нам желательно показать их в виде таблицы. Для этого они сначала преобразуются в набор данных DateFrame, а потом в таблицу HTML. После этого объекты s1 и s2 загружается в контекстную переменную context и передается в шаблон с именем made_series.html. Этот шаблон имеет следующий код (листинг 1.5).

Листинг 1.5. Шаблон made_series.html

Примечание.
В демо-версии книги листинги программ удалены.

Это достаточно простой шаблон, в котором имеется два контейнера: первый отвечает за вывод объекта s1, второй – за вывод объекта s2. Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 1» и активировать ссылку «Создание Series».

Если выполнить набор вышеприведенных модулей, то получим следующий результат (рис.1.2).

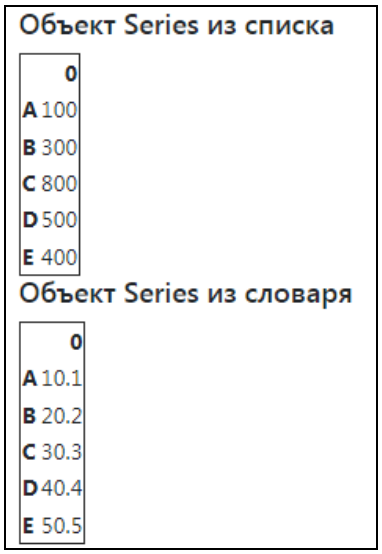


Рис. 1.2. Объекты Series библиотеки Pandas в шаблоне Django

Используя данный подход, можно создавать объекты Series с любыми данными и любым количеством строк.

1.4. Создание объектов DateFrame в приложении на Python

Для создания набора данных DataFrame в Pandas необходимо передать любой итерируемый объект (словарь, список, массив), содержащий данные для столбцов DataFrame. Конструктор класса DataFrame в библиотеке Pandas имеет следующий синтаксис:

```
df = pd.DataFrame(data=None, index=None, columns=None, dtype=None, copy=False),
```

где:

- data: данные, которые передаются в DataFrame;
- index: индексы, которые передаются в качестве индекса строк (по умолчанию RangeIndex);
- columns: метки столбцов, которые передаются в качестве индекса колонок (по умолчанию RangeIndex);
- dtype: тип данных, которые мы хотим передать;
- copy: копировать ли данные.

Параметр copy для словарей по умолчанию имеет значение True, а если в качестве данных передается DataFrame или массив NumPy, то по умолчанию False.

Создадим объект df, у которого в качестве индекса будет использован интервал дат (листинг 1.6)

Листинг 1.6. Создание объектов DataFram библиотеки Pandas (модуль my_data_frame.py)

Примечание.

В демо-версии книги листинги программ удалены.

В этом модуле создан объект period (интервал дат) с 11-го января по 20-го января 2025 года (американский формат дат). Затем сформирован словарь с данными из трех колонок. На основе этих данных создан набор данных DataFrame (объект df), которому в качестве источника данных задан словарь dic, а в качестве индексного столбца передан интервал дат (period). Затем индексной колонке задано имя «Date». Итоговый df и некоторые его параметры выведены в окно терминала.

Если выполнить данный программный модуль, то будет получен следующий результат.

Примечание.

Представленный в данном разделе программный модуль не включен в список исполняемых примеров интерактивной цифровой книги, так как он может работать только в рамках приложения Python локального компьютера, или быть вызванным из другого модуля веб приложения.

```
DatetimeIndex(['2025-01-11', '2025-01-12', '2025-01-13', '2025-01-14',
               '2025-01-15', '2025-01-16', '2025-01-17', '2025-01-18',
               '2025-01-19', '2025-01-20'],
              dtype='datetime64[ns]', freq='D')
--------------------------
Созданный df
              col1  col2  col3
Date
2025-01-11   1.0  11.0  21.0
2025-01-12   2.0  12.0  22.0
2025-01-13   3.0  13.0  23.0
2025-01-14   4.0  14.0  24.0
2025-01-15   5.0  15.0  25.0
2025-01-16   6.0  16.0  26.0
2025-01-17   7.0  17.0  27.0
2025-01-18   8.0  18.0  28.0
2025-01-19   9.0  19.0  29.0
2025-01-20  10.0  20.0  30.0
------------------------------
col1    float64
col2    float64
col3    float64
dtype: object
Поименованный индекс
DatetimeIndex(['2025-01-11', '2025-01-12', '2025-01-13', '2025-01-14',
               '2025-01-15', '2025-01-16', '2025-01-17', '2025-01-18',
               '2025-01-19', '2025-01-20'],
              dtype='datetime64[ns]', name='Date', freq='D')
Date
------------------------
```

Здесь сначала выведен объект period с датами (индексная колонка), затем созданный объект df в виде таблицы из 4-х колонок (одна индексная и три с данными), и параметры поименованного индекса. Из данной распечатки видно, что период представлен данными типа datetime (американский формат), в таком же виде эта информация присутствует в созданном объекте df, а сами данные имеют формат float64.

1.5. Создание объектов DataFrame в приложении на Django

А теперь покажем, как можно встроить программный модуль, о котором говорилось в предыдущем разделе, в web приложение и вывести созданный объект DataFrame в шаблон Django. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 1.7).

Листинг 1.7. Ссылка активации маршрута made_data_frame

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем made_data_frame (листинг 1.8).

Листинг 1.8. Маршрут URL с именем made_data_frame

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление made_data_frame (листинг 1.9).

Листинг 1.9. Представление (view) made_data_frame

Примечание.
В демо-версии книги листинги программ удалены.

В первой строке этого программного кода выполняется импорт модуля my_data_frame.py (см. предыдущий раздел). После этого выполнено обращение к функции m_df модуля my_data_frame, где формируется объект DataFrame (он сохраняется в объектной переменной df). Содержимое объекта df преобразуется в таблицу HTML, загружается в контекстную переменную context и передается в шаблон с именем made_data_frame.html. Этот шаблон имеет следующий код (листинг 1.10).

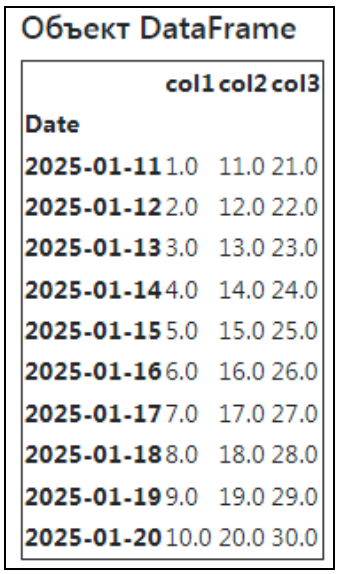
Листинг 1.10. Шаблон made_data_frame.html

Примечание.
В демо-версии книги листинги программ удалены.

Это достаточно простой шаблон, в котором имеется контейнер для вывода объекта df. Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 1» и активировать ссылку «Создание DataFrame».

Если выполнить набор вышеприведенных модулей, то получим следующий результат (рис.1.3).


Объект DataFrame

Date	col1	col2	col3
2025-01-11	1.0	11.0	21.0
2025-01-12	2.0	12.0	22.0
2025-01-13	3.0	13.0	23.0
2025-01-14	4.0	14.0	24.0
2025-01-15	5.0	15.0	25.0
2025-01-16	6.0	16.0	26.0
2025-01-17	7.0	17.0	27.0
2025-01-18	8.0	18.0	28.0
2025-01-19	9.0	19.0	29.0
2025-01-20	10.0	20.0	30.0

Рис. 1.3. Объект DataFrame библиотеки Pandas в шаблоне Django

Используя данный подход, можно создавать объекты DataFrame с любыми данными, с любым количеством строк и столбцов.

1.6. Работа с данными Pandas формата CSV

Формат CSV (Comma Separated Values — значения, разделенные запятыми) — один из самых популярных форматов хранения табличных данных. Он используется повсеместно, от экспорта Excel-таблиц до обмена данными между веб-сервисами и базами данных. Python предлагает мощные и простые инструменты для работы с файлами формата CSV, как на базовом уровне с модулем csv, так и с помощью библиотеки Pandas — де-факто стандарта для анализа данных. В этом разделе мы рассмотрим возможности библиотеки Pandas для чтения и формирования файлов формата CSV.

Метод to_csv() в библиотеке Pandas используется для записи данных из DataFrame в файл формата CSV. Этот метод предоставляет возможность настраивать различные аспекты

процесса сохранения данных в CSV файл. Вот некоторые основные аргументы метода to_csv():

- path_or_buf (обязательный параметр): путь к файлу, в который будут записаны данные (может быть строкой с именем файла или объектом файлового дескриптора);
- sep: разделитель, который будет использован при записи данных в файл CSV (по умолчанию это запятая «,»);
- columns: список столбцов, которые будут записаны в файл (по умолчанию записываются все столбцы);
- index: будет ли записываться индекс DataFrame (по умолчанию это True);
- index_label: позволяет указать название для столбца, который содержит индексы DataFrame при сохранении данных в файл CSV (этот параметр полезен, когда вы хотите добавить название к столбцу с индексами);
- header: будут ли записываться заголовок с названиями столбцов (по умолчанию это True);
- mode: режим записи файла, например «w» для перезаписи файла, или «a» для добавления данных к существующему файлу (по умолчанию «w»);
- encoding: кодировка, которая будет использована при записи данных в файл;
- date_format: формат даты для записи колонок дат в CSV файл.

Ниже приводятся некоторые варианты программного кода для записи набора данных DataFrame в файл с именем «output.csv».

```
# Сохраняет данные DataFrame в файл 'output.csv'
df.to_csv("output.csv")
# Сохраняет данные DataFrame в файл 'output.csv' с разделителем ';'
df.to_csv("output.csv", sep=";")
# Сохраняет только столбцы 'A' и 'B' из DataFrame в файл 'output.csv'
df.to_csv("output.csv", columns=["A", "B"])
# Не сохраняет индекс DataFrame в файл 'output.csv'
df.to_csv("output.csv", index=False)
# Запись DataFrame в файл CSV с указанием названия для столбца индексов
df.to_csv("output.csv", index_label="index_label")
# Не сохраняет заголовок (названия столбцов) в файл 'output.csv'
df.to_csv("output.csv", header=False)
# Добавляет данные из DataFrame в конец файла 'output.csv' (режим добавления)
df.to_csv("output.csv", mode="a")
# Сохраняет данные в файл 'output.csv' с использованием кодировки 'utf-8'
df.to_csv("output.csv", encoding="utf-8")
# Форматирует даты в соответствии с указанным форматом при сохранении в файл
'output.csv'
df.to_csv("output.csv", date_format="%Y-%m-%d")
```

Метод read_csv() в библиотеке Pandas используется для чтения данных из CSV файлов и загрузки их в объект DataFrame. Вот некоторые основные аргументы, которые можно передать в метод read_csv():

- filepath_or_buffer: этот аргумент указывает путь к файлу CSV или объект, который представляет CSV данные.
- sep: определяет разделитель столбцов в CSV файле. По умолчанию это запятая, но можно указать другой разделитель, например «;» для точки с запятой.
- header: указывает, на какой строке находится заголовок, то есть названия столбцов (по умолчанию это первая строка, можно также передать header=None, если заголовка нет);
- index_col: позволяет указать, какой столбец использовать в качестве индекса DataFrame;
- usecols: позволяет выбрать определенные столбцы для загрузки;
- dtype: позволяет указать тип данных для столбцов, что может быть полезно для дальнейшей обработки данных;
- nrows: позволяет указать количество строк, которые нужно прочитать из CSV файла (это может быть полезно для быстрой проверки данных или обработки только части данных);
- infer_datetime_format: позволяет автоматически определить формат даты и времени при чтении данных из CSV файла (если этот параметр установлен в True, Pandas будет пытаться самостоятельно определить формат даты и времени в столбцах, что может быть полезно при работе с разными форматами дат);
- parse_dates: используется для указания столбцов, которые нужно интерпретировать как даты при чтении данных из CSV файла (это помогает Pandas автоматически преобразовать значения в указанных столбцах в объекты datetime, что делает работу с такими данными более удобной).
- keep_date_col: позволяет сохранить исходный столбец с датами после преобразования его в тип данных datetime (если этот параметр установлен в True, то Pandas сохранит исходный столбец с датами в DataFrame после преобразования его в datetime, что может быть полезно для сохранения исходной информации);
- skiprows: позволяет пропустить определенное количество строк в начале файла;
- squeeze: если данные содержат 1 столбец, то вернет Series (False по умолчанию);
- encoding: позволяет указать кодировку символов при чтении данных из CSV файла.

Ниже приводятся некоторые варианты программного кода для чтения данных из файла с именем file.csv.

```
# Чтение данных из CSV файла по указанному пути
df = pd.read_csv("path/to/file.csv")
# Чтение данных из CSV файла с разделителем точка с запятой
df = pd.read_csv("file.csv", sep=";")
# Чтение данных из CSV файла без заголовка
df = pd.read_csv("file.csv", header=None)
# Использование столбца 'ID' как индекса DataFrame
df = pd.read_csv("file.csv", index_col="ID")
# Выбор только столбцов 'A', 'B', 'C' для загрузки
df = pd.read_csv("file.csv", usecols=["A", "B", "C"])
# Указание типа данных для столбца 'Value' как float
df = pd.read_csv("file.csv", dtype={"Value": float})
# Пропуск первых двух строк при чтении данных из CSV файла
```

```
df = pd.read_csv("file.csv", skiprows=2)
# Чтение только первых 5 строк из CSV файла
df = pd.read_csv("file.csv", nrows=5)
# Чтение CSV файла с интерпретацией столбца 'date' как дату
df = pd.read_csv('file.csv', parse_dates=['date'])
# Преобразование DataFrame в Series с помощью метода squeeze
series = df["A"].squeeze()
# Чтение CSV файла с попыткой автоматического определения формата даты
df = pd.read_csv("file.csv", infer_datetime_format=True)
# Чтение CSV файла с указанием кодировки символов
df = pd.read_csv("file.csv", encoding="utf-8")
```

1.7. Пример работы с данными формата CSV в приложениях на Django

В качестве примера реализуем программный модуль, в котором создадим набор данных DataFrame, сохраним его на диске в формате CSV, а потом выполним чтение данных из этого файла. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 1.11).

Листинг 1.11. Ссылка активации маршрута save_df_csv

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем save_df_csv (листинг 1.12).

Листинг 1.12. Маршрут URL с именем save_df_csv

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление save_df_csv (листинг 1.13).

Листинг 1.13. Представление (view) save_df_csv

Примечание.
В демо-версии книги листинги программ удалены.

В первой строке этого программного кода выполняется импорт библиотеки Pandas. После этого задан путь к CSV файлу с именем person.csv (объект path_csv), сформирован словарь с тестовыми данными (объект data), и создан исходный набор данных (объект df).

Содержимое этого набора данных сохранено на диске в файле формата CSV с параметрами по умолчанию:

```
df.to_csv(path_csv)
```

На следующем шаге в объект df_r загружены данные из CSV файла, из этого набора данных выбраны три колонки по их именам и сформирован новый набор данных (объект df_n).

Затем содержимое всех трех наборов данных преобразуется в таблицы HTML, загружаются в контекстную переменную context и передаются в шаблон с именем save_df_csv.html. Этот шаблон имеет следующий код (листинг 1.14).

Листинг 1.14. Шаблон save_df_csv.html

Примечание.
В демо-версии книги листинги программ удалены.

Этот шаблон имеет контейнер для вывода трех объектов - df, df_r и df_n. Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 1» и активировать ссылку «Запись DataFrame в файл CSV».

Если выполнить набор вышеприведенных модулей, то получим следующий результат (рис.1.4).

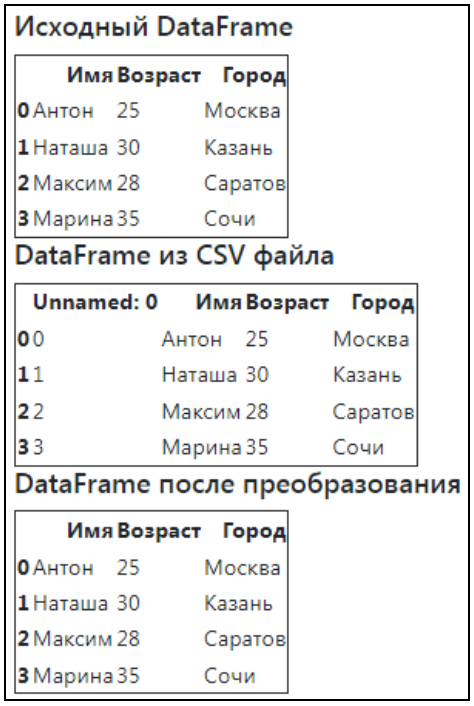


Рис. 14. Объект DataFrame до загрузки в CSV файл, и после чтения из CSV файл

Как видно из данного рисунка, исходный набор данных состоит из индексной колонки и трех колонок с данными. После того, как этот набор данных был записан на диск, а потом считан из CSV файла, у него появилась четвертая колонки с именем Unnamed:0. Эта колонки, по сути, дублирует индексную колонку. Для того, чтобы на выходе получить данные, аналогичные исходному набору, лишняя колонка была удалена. Таким образом, нужно помнить, что при работе с CSV файлами нужно проверять как структуру загруженного из CSV файла набора данных, таки тип данных, содержащихся в его колонках.

Если созданный csv файл открыть любым текстовым редактором, то будет получен следующий результат:

```
,Имя,Возраст,Город
0,Антон,25,Москва
1,Наташа,30,Казань
2,Максим,28,Саратов
3,Марина,35,Сочи
```

Как видно из полученных результатов, это обычный текстовый файл, в котором значения отделены друг от друга запятыми.

1.8. Работа с данными Pandas формата json

Формат JSON (JavaScript Object Notation) — это стандартный текстовый формат для хранения и передачи структурированных данных. Он основан на синтаксисе объектов в языке JavaScript, но не привязан к этому языку. Работу с данным форматом поддерживают многие современные языки программирования: Python, Java и другие.

До создания формата JSON запросы к серверу требовали постоянного обновления страницы в браузере. Это замедляло работу сайтов. Теперь же запросы, работающие с форматом JSON, выполняются в фоновом режиме и не требуют обновления страницы. Благодаря этому скорость работы с веб-ресурсами значительно выросла, а JSON стал стандартом передачи данных в сети интернет. В библиотеке Pandas есть встроенные функции, которые позволяют обрабатывать данные формата JSON.

Чтобы сохранить набор данных DataFrame в файле формата JSON, можно использовать функцию to_json(). Она также поставляется с рядом полезных аргументов для настройки файла JSON. Ниже приведен упрощенный синтаксис этой функции:

```
df.to_json("имя_файла.json")
```

Давайте рассмотрим ключевые параметры, которые можно задать для данной функции:

- path_or_buf - этот параметр указывает, куда следует записать данные в формате JSON (если он не указан, результат возвращается в виде строки);
- orient: этот параметр определяет формат вывода JSON;
- date_format: указывает, как форматировать объекты даты и времени (можно выбрать между iso форматом ISO 8601 и epoch форматом метки времени Unix);
- double_precision: указывает количество знаков после запятой для чисел с плавающей запятой, что делает вывод более читаемым, или позволяет сохранить точные числовые значения;

- force_ascii: если параметр имеет значение True, то все символы, отличные от ASCII, экранируются (это полезно для файлов JSON, предназначенных для систем, не поддерживающих Unicode);
- date_unit: указывает единицу измерения времени для объектов datetime, по умолчанию используется 'ms' (миллисекунды). Также можно использовать другие единицы измерения, такие как 's' (секунды), 'us' (микросекунды) или 'ns' (наносекунды), что позволяет контролировать точность измерения времени в выводе;
- indent: указывает количество пробелов для отступа на каждом уровне JSON, что упрощает чтение при установке положительного целого числа (например, indent=4).

Библиотека Pandas поддерживает несколько вариантов ориентации для формата JSON, что позволяет по-разному структурировать данные. Выбор правильной ориентации зависит от сценария использования.

- records - список словарей;
- columns - словарь с метками столбцов;
- index: - словарь с индексами строк;
- split: - словарь с индексом, столбцами и данными;
- table: - схема таблицы JSON.

Для чтения данных из файла json используется функция pandas:

pd.read_json("имя_файла.json")

Ниже приведены ключевые параметры этой функции:

- path_or_buf - . Указывает путь к файлу JSON или URL, по которому доступны данные (также можно передать строку с данными JSON).
- orient - задает ориентацию файла JSON (возможные значения: records, split, index, columns, values).
- typ - указывает тип ожидаемого результата (frame по умолчанию или series).
- dtype - позволяет указать типы данных столбцов.
- encoding -указывает кодировку, которая будет использоваться при чтении файла JSON.
- lines - указывает, читать ли файл как объект JSON по строке (по умолчанию — False).
- chunksize - позволяет читать файл частями (только при lines=True).
- nrows - указывает количество строк, которые нужно прочитать из файла, разделённого строками (только при lines=True).

1.8.1. Пример работы с данными формата JSON в приложениях на Django

В качестве примера реализуем программный модуль, в котором создадим набор данных DataFrame, сохраним его на диске в формате JSON, а потом выполним чтение данных из этого файла. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 1.15).

Листинг 1.15. Ссылка активации маршрута save_df_json

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем save_df_json (листинг 1.16).

Листинг 1.16. Маршрут URL с именем save_df_json

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление save_df_json (листинг 1.17).

Листинг 1.17. Представление (view) save_df_json

Примечание.
В демо-версии книги листинги программ удалены.

В первой строке этого программного кода выполняется импорт библиотеки Pandas. После этого задан путь к JSON файлу с именем «person.json» (объект path_json), сформирован словарь с тестовыми данными (объект data), и создан исходный набор данных (объект df). Содержимое этого набора данных сохранено на диске в файле формата JSON с параметром force_ascii=False:

```
df.to_json(path_json, force_ascii=False)
```

На следующем шаге в объект df_r загружены данные из JSON файла.

Затем содержимое двух наборов данных преобразуется в таблицы HTML, загружаются в контекстную переменную context и передаются в шаблон с именем save_df_json.html. Этот шаблон имеет следующий код (листинг 1.18).

Листинг 1.18. Шаблон save_df_json.html

Примечание.
В демо-версии книги листинги программ удалены.

Этот шаблон имеет контейнер для вывода двух объектов - df и df_r. Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 1» и активировать ссылку «Запись DataFrame в

Если выполнить набор вышеприведенных модулей, то получим следующий результат (рис.1.5).

Исходный DataFrame

	Имя	Возраст	Зарплата	Дата
0	Михаил	30	50000.0	2025-01-01
1	Елена	25	60000.0	2025-02-15
2	Василий	40	70000.0	2025-03-30

DataFrame из JSON файла

	Имя	Возраст	Зарплата	Дата
0	Михаил	30	50000	2025-01-01
1	Елена	25	60000	2025-02-15
2	Василий	40	70000	2025-03-30

Рис. 1.5. Объект DataFrame после загрузки из JSON файла

Как видно из данного рисунка, исходный набор данных состоит из индексной колонки и четырех колонок с данными. После того, как этот набор данных был записан на диск, а потом считан из JSON файла, у него сохранилась структура и количество колонок не изменилось. Однако тип данных изменился – в колонке «Зарплата» потерялась дробная часть. Таким образом, нужно помнить, что при работе с JSON файлами нужно проверять тип данных, содержащихся в его колонках.

Если созданный JSON файл открыть любым текстовым редактором, то будет получен следующий результат:

{"Имя":{"0":"Михаил","1":"Елена","2":"Василий"},"Возраст":{"0":30,"1":25,"2":40},"Зарплата":{"0":50000.0,"1":60000.0,"2":70000.0},"Дата":{"0":"2025-01-01","1":"2025-02-15","2":"2025-03-30"}}

Как видно из полученных результатов, это обычный текстовый файл, в котором значения представлены в виде вложенного словаря.

1.8.2. Работа в Pandas со строковыми объектами данных формата JSON

Набор данных DataFrame можно сохранить в формате JSON не только в виде файла, но и в виде строкового объекта. Это удобно в тех случаях, когда нужно передать данные из одного программного модуля в другой, или выполнить экспорт - импорт данных между приложениями.

Для записи набора данных в строковый объект используется следующий метод библиотеки Pandas:

```
data_json = df.to_json(orient='index')
```

Для загрузки данных из строкового объекта JSON в набор данных DataFrame используется следующий метод библиотеки Pandas:

```
df = pd.read_json(data_json, orient='index')
```

Напишем программный код модуля, в котором создадим набор данных DataFrame, запишем эти данные в строковый объект формата JSON, и загрузим данные из строки JSON в объект DataFrame. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 1.19).

Листинг 1.19. Ссылка активации маршрута save_str_json

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем save_str_json (листинг 1.20).

Листинг 1.20. Маршрут URL с именем save_str_json

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление save_str_json (листинг 1.21).

Листинг 1.21. Представление (view) save_str_json

Примечание.
В демо-версии книги листинги программ удалены.

В первой строке этого программного кода выполняется импорт библиотеки Pandas. После этого сформирован словарь с тестовыми данными (объект data), и создан исходный набор данных (объект df). Содержимое этого набора данных сохранено в строковом объекте формата JSON (объект data_json):

```
data_json = df.to_json(orient='index')
```

На следующем шаге в объект df_r загружены данные из строкового объекта формата JSON:

```
df_r = pd.read_json(data_json, orient='index')
```

Затем содержимое двух наборов данных преобразуются в таблицы HTML, загружаются в контекстную переменную context и передаются в шаблон с именем save_str_json.html. Этот шаблон имеет следующий код (листинг 1.22).

Листинг 1.22. Шаблон save_str_json.html

Примечание.
В демо-версии книги листинги программ удалены.

Этот шаблон имеет контейнер для вывода двух объектов - df и df_r. Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 1» и активировать ссылку «Запись DataFrame в строку JSON».

Если выполнить набор вышеприведенных модулей, то получим следующий результат (рис.1.6).

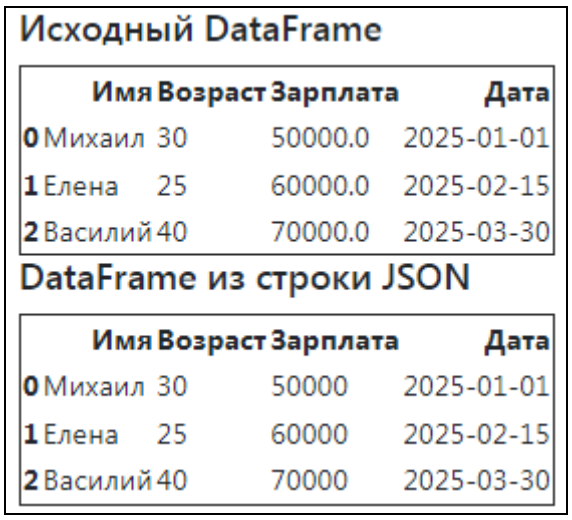


Рис. 1.6. Исходный объект DataFrame, и он же после загрузки из строкового объекта формата JSON

Как видно из данного рисунка, исходный набор данных состоит из индексной колонки и четырех колонок с данными. После того, как этот набор данных был записан в строковый объект формата JSON, а потом считан из строкового объекта, у него сохранилась структура и количество колонок не изменилось. Однако тип данных изменился – в колонке «Зарплата» потерялась дробная часть. Таким образом, нужно помнить, что при работе со строковыми объектами формата JSON нужно проверять тип данных, содержащихся в его колонках.

1.9. Индексация и доступ к элементам набора данных в Pandas

Следует отметить, что DataFrame библиотеки Pandas это универсальный набор данных, без которого не обходится ни один аналитик в Data Science (науке о данных). Если вы только начинаете свой путь в машинном обучении или в анализе временных рядов, то умение работать с этим инструментом — ваш первый и самый важный шаг.

Практически все знакомы с обычными таблицами, например, в Excel или в базах данных. У любой таблицы есть строки и столбцы, у столбцов есть названия (заголовки), а у строк — номера (или какие-то иные метки). Набор данных DataFrame в библиотеке Pandas — это, по сути, такая же двумерная табличная структура данных, но живущая внутри кода на языке Python. В этой таблице столбцы (Columns) могут содержать данные разных типов (числа, строки, даты, булевы значения), при этом у каждого столбца есть имя (или метка). Каждая строка (Rows) — это запись с данными в ячейках. У строк тоже есть метки, которые

называются индексом (Index). По умолчанию это просто целые числа от 0, но могут быть и осмысленные метки (например, даты или идентификаторы пользователей).

Конечно, данные можно хранить и в стандартных структурах Python – списки, кортежи, словари. Но как только данных становится много, а задачи обработки сложнее, то DataFrame показывает свои сильные стороны. Вот только некоторые преимущества наборов данных DataFrame:

1. **Удобная фильтрация и выборка**. Легко выбрать нужные строки или столбцы по условиям, меткам или позициям.
2. **Мощные матричные операции.** Различные групповые операции (агрегирование, группировка, подсчет среднего и суммы, слияние таблиц, обработка временных рядов) — все это делается парой строк программного кода.
3. **Обработка пропусков**. В реальных данных всегда есть пропуски. Pandas дает удобные инструменты, чтобы их искать и заполнять или удалять.
4. **Интеграция**. DataFrame фактически является стандартом обмена данными между другими библиотеками Data Science (Scikit-learn, Matplotlib, Seaborn, Plotly и т.п.).
5. **Оптимизация**. Многие операции в Pandas написаны на C или Cython, что делает их значительно быстрее, чем аналогичные операции на чистом Python с использованием циклов.

1.9.1 Выборка данных из DataFrame

Иногда объекты DataFrame настолько большие, что просматривать все строки одновременно становится неудобно. Чтобы показать первые несколько строк набора данных, используется метод df.head(), а для показа последних строк набора данных - метод — df.tail(). По умолчанию будет показано первые или последние 5 строк. С помощью необязательного параметра «n», который указывается в скобках, можно регулировать количество строк, которое вы хотите отобразить, например:

```
df.head(n=2)
df.tail(n=10
```

Есть два основных способа получить значение ячеек, строк или столбцов:

- df.loc[] - Location или местоположение (использует МЕТКИ индекса и столбцов);
- df.iloc[] - Integer Location или целочисленное местоположение (использует целочисленные индексы, начиная с 0).

Метод df.loc[] обеспечивает следующие варианты выборки данных:

- df.loc[метка_строки, метка_столбца] - выбрать конкретную ячейку;
- df.loc[метка_строки] или df.loc[метка_строки, :] - выбрать всю строку по метке;
- df.loc[:, метка_столбца] - выбрать весь столбец по имени;
- df.loc[[метка1, метка2], [столбец1, столбец2]] - выбрать несколько строк и столбцов по их меткам;
- df.loc[метка_начала:метка_конца] - срез строк по меткам (включая конец!).

Метод df.iloc[] обеспечивает следующие варианты выборки данных:

- df.iloc[индекс_строки, индекс_столбца] - выбрать ячейку по номеру строки и столбца;

- df.iloc[индекс_строки] или df.iloc[индекс_строки, :] - выбрать всю строку по ее номеру;
- df.iloc[:, индекс_столбца] - выбрать весь столбец по его номеру;
- df.iloc[[0, 2], [0, 1]] - выбрать строки 0 и 2, столбцы 0 и 1;
- df.iloc[0:2] - срез строк по позициям (НЕ включая конец, как в Python!).

> ***Примечание.***
> Важно не путать методы df.loc[] и df.iloc[]! Метод df.loc работает с именами (метками), а метод df.iloc – с номерами позиций.

Доступ к столбцам. Самый простой способ выбрать один столбец — использовать квадратные скобки с именем столбца: df[[‘имя_столбца’]]. Это вернет объект Series. Можно также использовать атрибутный доступ df.имя_столбца, но это работает, только если имя столбца — валидный идентификатор Python (без пробелов, спецсимволов, не совпадает с методами DataFrame). Безопаснее использовать df[‘имя_столбца’]. Выборка столбца по его имени показана на рис. 1.7.



Рис. 1.7. Выборка столбца из DataFrame по его имени

Доступ ко всей строке можно получить по ее номеру (рис. 1.8).

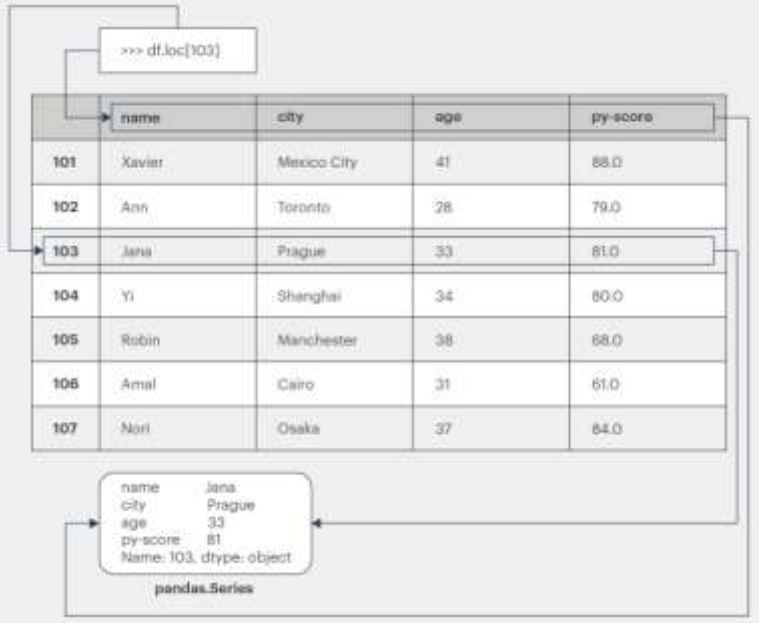


Рис. 1.8. Выборка строки из DataFrame по его номеру

Есть еще два метода доступа к элементам DataFrame:

df.at[] - обеспечивает доступ к одному элементу по метке индекса и столбца;
df.iat[] - предоставляет доступ к одному элементу по числовому индексу и позиции.
Предположим, что мы имеем следующий набор данных (df):

	Фрукты	Овощи
0	150	420
1	250	520
2	350	625

Метод df.at[0, 'Фрукты'] выведет значение- 150, а метод df.iat[0, 1] выдаст значение - 420

Достаточно часто требуется получить метки (имена) строк и столбцов. Метки строк DataFrame можно получить с помощью метода df.index, а метки столбцов с помощью метода df.columns. Например:

```
print(df.index) # Вывод меток строк
print(df.columns) # Вывод меток столбцов
```

Напишем программный код модуля, в котором создадим набор данных DataFrame, и выполним выборку данных из него разными способами. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 1.23).

Листинг 1.23. Ссылка активации маршрута get_df

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем get_df (листинг 1.24).

Листинг 1.24. Маршрут URL с именем get_df

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление get_df (листинг 1.25).

Листинг 1.25. Представление (view) get_df

Примечание.
В демо-версии книги листинги программ удалены.

В первой строке этого программного кода выполняется импорт библиотеки Pandas. После этого сформирован словарь с тестовыми данными (объект data), и создан исходный набор данных (объект df).

На следующем шаге выполнена выборка данных по именам двух колонок (объект df_col), выбор всей строки (объект df_str), выбор ячейки (объект df_str_col).

Затем содержимое набора данных df_col преобразуется в таблицу HTML, все полученные данные загружаются в контекстную переменную context и передаются в шаблон с именем get_df.html. Этот шаблон имеет следующий код (листинг 1.26).

Листинг 1.26. Шаблон get_df.html

Примечание.
В демо-версии книги листинги программ удалены.

Этот шаблон имеет контейнер для вывода трех объектов - df_col, df_str и df_str_col. Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 1» и активировать ссылку «Выборка данных из DataFrame».

Если выполнить набор вышеприведенных модулей, то получим следующий результат (рис.1.9).

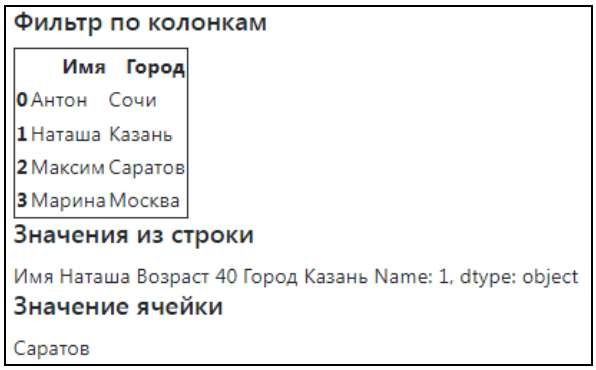


Рис. 1.9. Выборка данных DataFrame

1.9.2 Фильтрация данных в DataFrame

Для фильтрации данных в DataFrame используется следующий принцип: сначала создается условие, которое для каждой строки возвращает значение True или False. Фактически создается «маска», а затем эта «маска» используется для фильтрации DataFrame.

Фильтрация на основе простого условия. При фильтрации на основе простого условия используется простое логическое выражение. Рассмотрим следующий код:

```
mask = df['Возраст'] < 30
df_filter = df[mask]
```

или

```
df_filter = df[df['Возраст'] < 30]
```

В данном примере в первой строке для исходного набора данных df создается маска (объект mask) или условие – выбрать все записи (строки), для которых значения в колонке «Возраст» меньше 30 лет. Во второй строке создается новый объект df_filtr, в который перенесутся все строки, для которых в колонке «Возраст» значения меньше 30 лет. С использованием фильтра можно изменить и сам исходный набор данных, например:

```
mask = df[('Возраст'] < 30)
df = df[mask])
```

Фильтрация на основе комбинированного условия. При фильтрации на основе комбинированного условия используется несколько логических выражений, объединенных логическими операторами: «&» - логическое **И**, «|» - логическое **ИЛИ**, «~» - логическое **НЕ**. Рассмотрим следующий код:

```
df_filter = df[(df['Возраст'] > 30) & (df['Город'] == 'Москва')]
```

Обратите внимание, что каждое логическое условие нужно заключить в круглые скобки.

Напишем программный код модуля, в котором создадим набор данных DataFrame, и выполним его фильтрацию разными способами. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 1.27).

Листинг 1.27. Ссылка активации маршрута filter_df

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем filter_df (листинг 1.28).

Листинг 1.28. Маршрут URL с именем filter_df

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление filter_df (листинг 1.29).

Листинг 1.29. Представление (view) filter_df

Примечание.
В демо-версии книги листинги программ удалены.

В первой строке этого программного кода выполняется импорт библиотеки Pandas. После сформирован словарь с тестовыми данными (объект data), и создан исходный набор данных (объект df).

На следующем шаге выполнена фильтрация полученного набора данных по маске (объект df_filter), с использованием простого фильтра (объект df_filter1), с использованием комбинированного фильтра (объект df_filter2).

Затем содержимое трех наборов данных преобразуется в таблицы HTML, загружаются в контекстную переменную context и передаются в шаблон с именем filter_df.html. Этот шаблон имеет следующий код (листинг 1.30).

Листинг 1.30. Шаблон filter_df.html

Примечание.
В демо-версии книги листинги программ удалены.

Этот шаблон имеет контейнер для вывода трех объектов - df_filter, df_filter1 и df_filter2. Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 1» и активировать ссылку «Фильтрация DataFrame».

Если выполнить набор вышеприведенных модулей, то получим следующий результат (рис.1.10).

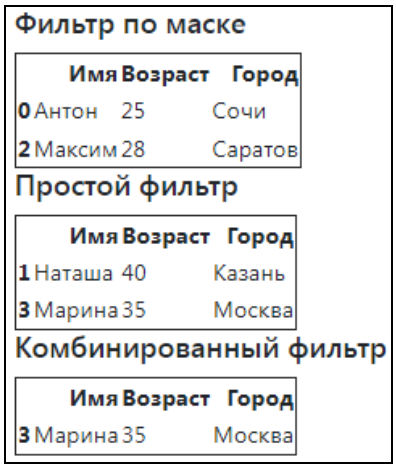


Рис. 1.10. Фильтрация набора данных DataFrame

1.9.3 Манипуляции с данными DataFrame

Рассмотрим некоторые базовые методы Pandas, которые обеспечивают обработку наборов данных DataFrame:

- df.drop() - метод позволяет удалять столбцы и строки;
- df.sort_values() - метод выполняет сортировку набора данных;
- df.apply() - метод применяет функцию к каждому элементу набора данных (строке, столбцу).

Метод df.drop. Достаточно часто возникают ситуации, когда для ускорения процессов обработки данных из большого набора данных нужно удалить часть строк или столбцов. Для этой цели используется метод df.drop(), который имеет следующие параметры:

```
df.drop(labels axis, index, columns, level, inplace, errors)
```

Здесь параметры имеют следующий физический смысл:

- labels - метки строк или столбцов, которые нужно удалить (если меток несколько, их указывают в списке);
- axis – определяет, удалять строки (axis=0, по умолчанию) или столбцы (axis=1);
- index - список имен строк, которые нужно удалить (может использоваться вместо параметра labels);
- columns - список имён столбцов, которые нужно удалить;
- level - для многоиндексных наборов данных указывает уровень, с которого нужно удалить метки;
- inplace – если параметр имеет значение True, то изменения вносятся в исходный DataFrame, если False — возвращается новая копия;
- errors – если параметр имеет значение «ignore», то ошибки игнорируются, если какие-либо указанные метки не существуют (по умолчанию — «raise», возникает ошибка, если метки не найдены).

Рассмотрим использование данного метода на простом примере (листинг 1.31).

Листинг 1.31. Модуль metod_drop.py

Примечание.
В демо-версии книги листинги программ удалены.

В этом программном модуле выполнен импорт библиотеки Pandas и создан набор данных из 4-х столбцов и 3-х строк. Затем из исходного набора данных удалены два столбца (объект df_drop_col), и удалены две строки (объект df_drop_row). Все наборы данных выведены на печать.

Если выполнить данный программный код, то будет получен следующий результат.

```
Выводим весь df
   А  Б  В  Г
0  10  45  74  94
1  20  55  85  35
2  30  65  96  66
------------------------------
df с удаленными столбцами 'A' и 'B'
   Б  Г
0  45  94
1  55  35
2  65  66
------------------------------
df с удаленными строками 0 и 1
   А  Б  В  Г
2  30  65  96  66
```

Примечание.
Представленный в данном разделе программный модуль не включен в список исполняемых примеров интерактивной цифровой книги, так как он может работать только в рамках приложения Python локального компьютера, или быть вызванным из другого модуля веб приложения.

Как видно из данной распечатки, в первом случае из набора данных были удалены колонки с именами «В» и «Г», во втором случае были удалены строки с индексами 0 и 1.

Метод df.sort_values. Достаточно часто возникают ситуации, когда набор данных необходимо пересортировать в том или ином порядке. Для этой цели используется метод df.sort_values(), который имеет следующие параметры:

```
df.sort_values(by, axis, ascending, inplace, kind, na_position)
```

Здесь параметры имеют следующий физический смысл:

- by - указывает имя или список имен столбцов, по которым будут сортироваться значения (если передать список, можно задать иерархию сортировки: сначала по одному столбцу, затем по другому);
- axis – определяет ось сортировки (axis=0 сортировки по столбцам, а axis=1 - по строкам);

- ascending – порядок сортировки (True – по возрастанию, False – по убыванию);
- inplace – выполнение операций на месте (значение True означает, что изменения будут отражены в исходном наборе данных;
- kind - определяет алгоритм сортировки (quicksort - по умолчанию, mergesort, heapsort, stable);
- na_position - указывает, где будут расположены NaN-значения (first — в начале, last — в конце).

Рассмотрим использование данного метода на простом примере (листинг 1.32).

Листинг 1.32. Модуль metod_sort.py

Примечание.
В демо-версии книги листинги программ удалены.

В этом программном модуле выполнен импорт библиотеки Pandas и создан набор данных из 3-х столбцов и 3-х строк. Затем исходный набор отсортирован по столбцу «А» (объект df_sorted). Все наборы данных выведены на печать.

Если выполнить данный программный код, то будет получен следующий результат.

```
Выводим весь df
   А  Б  В
0  3  6  9
1  2  5  8
2  1  4  7
------------------------------
Сортируем данные по столбцу 'А'
   А  Б  В
2  1  4  7
1  2  5  8
0  3  6  9
```

Примечание.
Представленный в данном разделе программный модуль не включен в список исполняемых примеров интерактивной цифровой книги, так как он может работать только в рамках приложения Python локального компьютера, или быть вызванным из другого модуля веб приложения.

Как видно из данной распечатки, после сортировки все строки набора данных выведены в другом порядке (по возрастанию значений столбца «А»).

Метод df.apply. Метод apply() в Pandas позволяет применять функцию к каждому элементу, строке или столбцу DataFrame. Данный метод имеет следующие параметры:

```
df.apply(func, axis, raw, result_type)
```

Здесь параметры имеют следующий физический смысл:

- func - функция, которая применяется к данным (может быть встроенной или пользовательской);
- axis – указывает направление обхода данных, 0 (по умолчанию) – обход по столбцам, 1 – обход по строкам;

- raw – определяет, передавать ли в функцию объекты `Series` или `ndarray:` False (по умолчанию) — каждый ряд или столбец передаётся как `Series,` True — вместо `Series` функция получает `ndarray`;
- result_type – контролирует формат вывода;

Параметр result_type может принимать следующие значения:

- «expand» — значения, похожие на список, превращаются в столбцы;
- «reduce» — метод возвращает Series вместо расширения списка;
- «broadcast» — результаты распространяются на исходную форму DataFrame, при этом сохраняются исходный индекс и столбцы.

По умолчанию (result_type=None) тип возврата определяется по типу возвращаемого значения функции.
Рассмотрим использование данного метода на простом примере (листинг 1.33).

Листинг 1.33. Модуль metod_ apply.py

Примечание.
В демо-версии книги листинги программ удалены.

В этом программном модуле выполнен импорт библиотеки Pandas и создан набор данных из 3-х столбцов и 3-х строк (объект df). Затем в этом наборе данных делаем следующие изменения:

- столбец «Сливы» преобразуем к цифровому виду;
- все ячейки набора данных умножаем на 2;
- столбец «Яблоки» делим на 2;
- вторую строку умножаем на 10.

Все промежуточные результаты выведены в окно терминала.

Если выполнить данный программный код, то будет получен следующий результат.

```
Выводим весь df
   Яблоки  Груши Сливы
0      10     40  70.7
1      20     50  80.8
2      30     60  90.9
------------------------------
Столбец Сливы делаем цифровым
   Яблоки  Груши  Сливы
0      10     40   70.7
1      20     50   80.8
2      30     60   90.9
------------------------------
Все ячейки умножаем на 2
   Яблоки  Груши  Сливы
0      20     80  141.4
1      40    100  161.6
2      60    120  181.8
```

```
------------------------------
Столбец Яблоки делим на  2
   Яблоки  Груши  Сливы
0    10.0     80  141.4
1    20.0    100  161.6
2    30.0    120  181.8
------------------------------
Вторую строку умножаем на 10
   Яблоки  Груши   Сливы
0    10.0     80   141.4
1   200.0   1000  1616.0
2    30.0    120   181.8
```

Примечание.
Представленный в данном разделе программный модуль не включен в список исполняемых примеров интерактивной цифровой книги, так как он может работать только в рамках приложения Python локального компьютера, или быть вызванным из другого модуля веб приложения.

Как видно из данных результатов, указанные функции успешно обработали заданные элементы набора данных.

1.9.4 Методы статистической агрегации DataFrame

В библиотеке Pandas для статистической агрегации данных используются встроенные методы sum(), mean(), min() и max(). Эти функции позволяют вычислять сумму, среднее значение, минимальное и максимальное значения в DataFrame.

Функция sum(). Вычисляет сумму значений по указанной оси. По умолчанию ось — столбцы (axis=0). Можно указать ось строк (axis=1).

Функция mean(). Возвращает среднее значение по указанной оси. Если ось не указана, по умолчанию вычисляется среднее значение для каждого столбца.

Функция min(). Находит минимальное значение по указанной оси. По умолчанию ось — столбцы (axis=0). Можно включить обработку пустых значений (NaN), установив параметр skipna в False.

Функция max(). Возвращает максимальное значение по указанной оси. По умолчанию метод работает по столбцам, но можно указать ось строк (axis=1).

Рассмотрим использование этих методов на простом примере (листинг 1.34).

Листинг 1.34. Модуль metod_stat.py

Примечание.
В демо-версии книги листинги программ удалены.

В этом программном модуле выполнен импорт библиотеки Pandas и создан набор данных из 3-х столбцов и 3-х строк (объект df). Затем в этом наборе данных делаем следующие изменения:

- подсчитана сумма значений по столбцам;

- подсчитано среднее значения для каждого столбца;
- определено минимальное значение для каждого столбца;
- определено максимальное значение для каждого столбца.

Все промежуточные результаты выведены в окно терминала.

Если выполнить данный программный код, то будет получен следующий результат.

```
Выводим весь df
   Яблоки  Груши  Сливы
0      10     40     70
1      20     50     80
2      30     60     90
------------------------------
Сумма значений по столбцам
Яблоки     60
Груши     150
Сливы     240
dtype: int64
------------------------------
Среднее значение по столбцам
Яблоки    20.0
Груши     50.0
Сливы     80.0
dtype: float64
------------------------------
Минимальное значение по столбцам
Яблоки    10
Груши     40
Сливы     70
dtype: int64
------------------------------
Максимальное значение по столбцам
Яблоки    30
Груши     60
Сливы     90
dtype: int64
------------------------------
```

Примечание.

Представленный в данном разделе программный модуль не включен в список исполняемых примеров интерактивной цифровой книги, так как он может работать только в рамках приложения Python локального компьютера, или быть вызванным из другого модуля веб приложения.

Как видно из данных результатов, указанные встроенные методы успешно обработали заданные элементы набора данных, при этом сам набор данных не изменился.

1.9.5 Группировка и агрегация в DataFrame

Группировка в Pandas. Группировка в Pandas означает объединение данных в группы по значениям некоторых столбцов. После группировки вы можете выполнять такие действия, как вычисление суммы, среднего значения, количества или даже выбор первой строки из каждой группы.

К сгруппированным данным могут быть применены следующие функции агрегации:

- sum() — сумма значений в группе;
- mean() — среднее значение в группе;
- count() — количество элементов в группе;
- min() — минимальное значение в группе;
- max() — максимальное значение в группе.

Давайте посмотри, как работает группировка в Pandas, на примере небольшого набора данных о стоимости кондитерских изделий (листинг 1.35).

Листинг 1.35. Модуль metod_ group

Примечание.
В демо-версии книги листинги программ удалены.

В этом программном модуле выполнен импорт библиотеки Pandas и создан набор данных из 3-х столбцов и 4-х строк (объект df). Затем в этом наборе данных выполнена группировка по столбцу «Продукты» с расчетом средней цены каждого продукта (объект grouped_data):

```
grouped_data = df.groupby('Продукт').agg({'Цена': 'mean'})
```

Результаты группировки и расчета средней цены выведены в окно терминала.

Если выполнить данный программный код, то будет получен следующий результат.

```
Выводим весь df
   Продукт      Название        Цена
0  Пирожные  Шоколадное    80
1  Пирожные   Ванильное     90
2   Конфеты     Белочка         300
3   Конфеты     Грильяж        400
4   Конфеты    Ласточка        200
------------------------------
Средняя цена продукта
              Цена
Продукт
Конфеты     300.0
Пирожные   85.0
------------------------------
```

Примечание.
Представленный в данном разделе программный модуль не включен в список исполняемых примеров интерактивной цифровой книги, так как он может работать только в рамках

приложения Python локального компьютера, или быть вызванным из другого модуля веб приложения.

Как видно из полученных результатов, в исходном наборе данных создано две группы товаров (Конфеты, Пирожные), для каждой группы подсчитана средняя цена товара.

Агрегация в Pandas. Агрегация означает применение математической функции для обобщения данных. Ее можно использовать для получения сводной информации по столбцам в наборе данных, например для получения суммы, минимума, максимума и т. д. Функция, используемая для агрегации — agg(). Для этой функции в качестве параметра указывается другая функция, которую нужно выполнить. Вот некоторые функции, используемые для агрегации данных:

- sum() - вычисление суммы значений в столбце;
- min() - вычисление минимального значения в столбце;
- max() - вычисление максимального значения в столбце;
- mean() - вычисление среднего значения столбца;
- describe() - генерирует описательную статистику;
- first() - вычисление первого значения в группе;
- last() - вычисление последнего значения в группе;
- count() - подсчитывает количество значений в столбце;
- std() - стандартное отклонение столбца;
- var() - вычислить дисперсию столбца;
- sem() - стандартная ошибка среднего значения в столбце.

Давайте посмотри, как работает агрегация в Pandas, на примере небольшого набора данных о некоторых столичных городах (листинг 1.36).

Листинг 1.36. Модуль metod_ group

Примечание.
В демо-версии книги листинги программ удалены.

В этом программном модуле выполнен импорт библиотеки Pandas и создан набор данных из 3-х столбцов и 6-х строк (объект df). Затем в этом наборе данных выполнена агрегация по столбцам «Площадь» и «Население» с использованием функций агрегации: min, max и sum:

```
aggregated_data = df.agg({'Площадь': ['max', 'min', 'sum']})
aggregated_data = df.agg({'Население': ['max', 'min', 'sum']})
```

Результаты агрегации и расчета выведены в окно терминала.
Если выполнить данный программный код, то будет получен следующий результат.

```
Выводим весь df
  Столица   Население  Площадь
0  Лондон    8908081    1572.00
1   Париж    2140526     105.40
2   Токио    13929286   2190.93
3  Берлин    3748148     891.68
```

```
4    Рим     2870493     1285.31
5  Москва   13149803    2561.50
------------------------------
    Площадь
max  2561.50
min   105.40
sum  8606.82
--------------------------
    Население
max   13929286
min    2140526
sum   44746337
--------------------------
```

Примечание.
Представленный в данном разделе программный модуль не включен в список исполняемых примеров интерактивной цифровой книги, так как он может работать только в рамках приложения Python локального компьютера, или быть вызванным из другого модуля веб приложения.

Как видно из полученных результатов, для колонки данных определены значения: максимальное, минимальное и суммарное.

1.9.6 Сводная таблица PivotTables

Сводная таблица - это таблица статистики, которая помогает обрабатывать данные большей таблицы путем "поворота" этих данных. Сводные таблицы популярны в Excel, где они известны как PivotTables. Библиотека Pandas предоставляет возможность создавать сводные таблицы в Python с использованием функции .pivot_table(). Функция имеет следующие параметры:

df.pivot_table(data, values, index, columns, aggfunc, fill_value, margins, dropna, margins_name, observed, sort)

Эти параметры имеют следующий физический смысл:

- data – набор данных DataFrame;
- values - столбец для агрегирования (если он пуст, будут агрегированы все числовые значения);
- index - столбец или столбцы, по которым группируются данные (один столбец может быть строкой, а несколько столбцов должны представлять собой список строк);
- columns - столбец или столбцы, по которым группируются данные (один столбец может быть строкой, а несколько столбцов должны представлять собой список строк);
- aggfunc - функция или список функций для агрегирования данных (по умолчанию это ‘mean’ – среднее значение);
- fill_value - значение для замены отсутствующих значений;
- margins - добавление строки и столбца для итогов (по умолчанию – False);
- dropna - не включает столбцы, в которых все записи имеют значение NaN (по умолчанию – True);

- margins_name - название итоговой строки/столбца (по умолчанию – ‘All’);
- observed - только для категориальных данных (если True), будут отображаться только наблюдаемые значения для категориальных групп (по умолчанию – False);
- sort - сортировать ли полученные значения (по умолчанию – True).

Давайте посмотри, как можно использовать сводные таблицы в Pandas, на примере небольшого набора данных о продуктах (листинг 1.37).

Листинг 1.37. Модуль metod_pivit_table.py

Примечание.
В демо-версии книги листинги программ удалены.

В этом программном модуле выполнен импорт библиотеки Pandas и создан набор данных из 4-х столбцов и 4-х строк (объект df). Затем в этом наборе данных выполнены различные расчеты с использованием сводной таблицы. Все промежуточные результаты выведены в окно терминала.

Если выполнить данный программный код, то будет получен следующий результат.

```
Исходный df
  Продукты Категории  Количество  Сумма
0  Морковь     Овощи           8            270
1  Капуста     Овощи           5            239
2  Яблоки      Фрукты          3            617
3   Груши      Фрукты          4            384
------------------------------
Сумма по каждому продукту
              Сумма
Продукты
Груши         384
Капуста       239
Морковь       270
Яблоки        617
------------------------------
Сумма по каждой категории продукта
              Сумма
Категории
Овощи          509
Фрукты        1001
------------------------------
Сумма по каждой категории и каждому продукта
                          Сумма
Продукты Категории
Груши    Фрукты          384
Капуста  Овощи           239
Морковь  Овощи          270
Яблоки   Фрукты          617
------------------------------
Параметры по каждой категории продукта
```

```
             median   max      min
             Сумма    Сумма    Сумма
Категории
Овощи        254.5    270      239
Фрукты       500.5    617      384
------------------------------
```

Примечание.

Представленный в данном разделе программный модуль не включен в список исполняемых примеров интерактивной цифровой книги, так как он может работать только в рамках приложения Python локального компьютера, или быть вызванным из другого модуля веб приложения.

Как видно из полученных результатов, с использованием сводной таблицы можно выполнять обработку исходного набора данных в разных разрезах, при этом сам набор данных не изменяется.

1.10. Краткие итоги

Итак, мы познакомились с некоторыми базовыми модулями библиотеки Pandas, которые используются для обработки данных. Знание и умение использовать возможности библиотеки Pandas очень важны как для разработчиков программного обеспечения, так и для аналитика данных. Все знания, полученные при изучении данной главы, пригодятся вам при рассмотрении материалов следующих глав. Библиотека Pandas это фундамент, на котором базируются методы анализа и обработки данных других библиотек, связанных с наукой о данных.

Но прежде чем приступить к рассмотрению методов решения задач анализа и прогнозирования данных, необходимо разобраться с очень важным типом данных – с временными рядами. Это базовый тип данных, который используется при анализе финансовой и экономической деятельности, в торговле, промышленности, транспорте, исследовании климата и т.п. Базовые понятия о природе и структуре временных рядов будут рассмотрены в следующей главе, из которой вы узнаете:

- об основных компонентах и характеристиках временных рядов;
- о базовых статистических моделях, которые позволяют моделировать поведение временных рядов.

Глава 2. Общие понятия о временных рядах

В последние годы наука о данных переживает настоящий бум. Все большее количество компаний используют различные инструменты для сбора, хранения и анализа данных, чтобы повысить эффективность своих бизнес-процессов и принятие оптимальных решений.

Одним из наиболее важных инструментов в науке о данных является анализ временных рядов. Временной ряд - это последовательность наблюдений за определенным параметром (или группы параметров) в разные моменты времени. Таким образом, временной ряд содержит информацию о том, как изменялся тот или иной параметр с течением времени.

Примерами временных рядов могут служить данные о продажах товаров, сведения о пассажиропотоках на различных видах транспорта, данные о температуре или загрязнении воздуха на определенной территории в различные временные промежутки. Цель анализа временных рядов - определить закономерности в изменениях параметра во времени и на основе исторических данных делать прогнозы на будущее.

Применение анализа временных рядов может быть полезно для многих областей, включая финансы, экономику, производство, транспорт, социальные аспекты, изменение климата и т.д. Конечно, важно понимать, что в процессе анализа временных рядов требуется учитывать множество случайных факторов, поэтому следует уметь применять правильные методы и инструменты для получения корректных результатов.

Таким образом, временные ряды являются одним из важных компонентов современной аналитики данных, и имеет широкое применение в различных областях. В данной главе будут рассмотрены основные особенности и характеристики временных рядов.

В частности из материалов этой главы вы узнаете:

- об основных компонентах временных рядов;
- какие статистические меры используются для оценки параметров временных рядов;
- что такое стационарность временного ряда;
- какие методы используются для приведения временных рядов к стационарности;
- какие статистические модели используются для прогнозирования временных рядов и роль машинного обучения при решении задач прогнозирования.

2.1. Основные компоненты временного ряда

Временной ряд — это последовательность наблюдений за некоторым показателем, который зафиксирована во времени с регулярными интервалами. Примеры временных рядов включают ежедневные данные о продажах товаров, о котировках акций или курсах валют, показатели температуры воздуха, пассажиропотоки, среднегодовые данные о численности населения, индикаторы экономического развития стран и т.д. Временные ряды могут анализироваться для обнаружения тенденций, циклов, сезонных колебаний и других закономерностей.

Данные, которые связаны со временем, могут иметь разную природу, а соответственно потребуются и разные подходы к их анализу. Можно выделить следующие типы временных данных:

- перекрестные данные (cross-sectional) — это наблюдения за несколькими объектами в один (фиксированный) момент времени (например, цены на квартиры на текущий день);
- временные ряды (time series) — это данные, последовательно собранные в регулярные промежутки времени (например, курс валют, котировки акций, цены на нефть);
- панельные данные (panel data) — данные, получаемые серией измерений или наблюдений за несколько периодов времени для одних и тех же компаний или стран (например, ежегодные данные о ВВП в разных странах).

Задачи, связанные с временными рядами можно разбить на две категории:

- анализ исторических данных временного ряда (поиск аномальных выбросов, наличие сезонных колебаний, устойчивых тенденций, корреляций с различными событиями и внешними факторами);
- прогнозирование изменения исследуемого параметра на основе анализа исторических данных.

Анализ временных рядов требует понимания их основных составляющих. Каждый временной ряд можно разложить на несколько ключевых компонентов, которые помогают лучше понять структуру данных и сделать более точные прогнозы. Временной ряд может иметь следующие компоненты (рис. 2.1):

- тренд (Trend);
- сезонность (Seasonal);
- цикличность (Cyclical);
- случайность (Irregular).

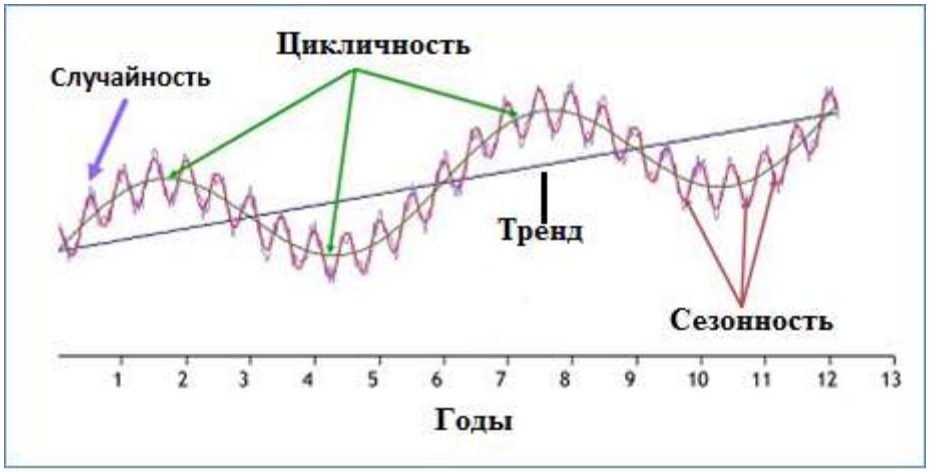


Рис. 2.1. Компоненты временного ряда

Тренд. Тренд отражает долгосрочную тенденцию изменения значений временного ряда. Он показывает общее направление движения данных: возрастающее, убывающее или стабильное. Тренд может быть линейным или нелинейным, и его выделение важно для анализа общей динамики.

Тренды могут быть следующих видов:

- линейный тренд: когда значения увеличиваются или уменьшаются с постоянной скоростью.
- экспоненциальный тренд: когда скорость роста или падения изменяется со временем.

Сезонность. Сезонность — это повторяющиеся и предсказуемые колебания значений временного ряда, связанные с определенными периодами времени (день, неделя, месяц). Сезонные колебания часто обусловлены внешними факторами, такими как погодные условия, праздничные периоды или изменение потребительских предпочтений.

Примеры сезонности:

- рост продаж прохладительных напитков в летние месяцы;
- увеличение количества пассажиров в летние периоды.

Цикличность. Цикличность характеризуется более долгосрочными колебаниями значений, которые не имеют строго определенного периода, в отличие от сезонности. Циклы могут быть вызваны экономическими условиями, деловыми циклами или другими факторами, влияющими на показатели на протяжении нескольких лет. Анализ циклов помогает понять влияние глобальных макроэкономических факторов на временной ряд.

Примеры цикличности:

- экономические кризисы, которые происходят каждые 7-10 лет.
- периоды подъема и спада на фондовых рынках.

Случайность. Случайная компонента — это остаток временного ряда после устранения тренда, сезонности и цикличности. Она представляет собой шум или непредсказуемые колебания, вызванные случайными событиями. Случайность невозможно предсказать или моделировать, но она важна для понимания общей структуры данных и улучшения точности прогнозов.

Примеры случайности:

- неожиданные изменения курса валют из-за внезапных политических событий.
- колебания цен на акции в результате случайных новостей или слухов.

С использованием специальных математических методов временной ряд можно разложить на приведенные выше составляющие и показать их в графическом виде (рис. 2.2).

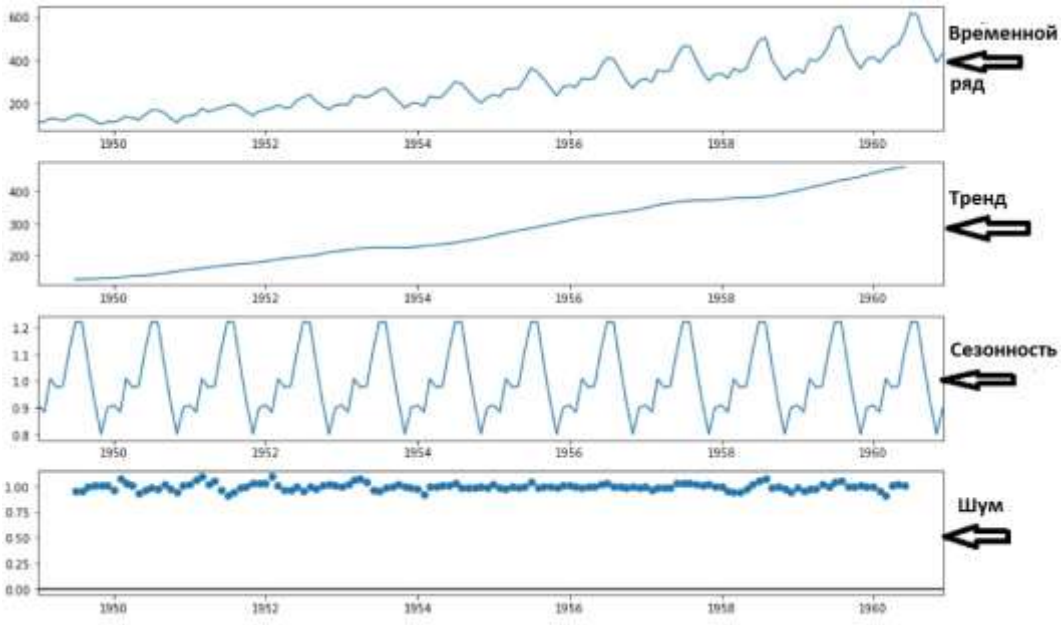


Рис. 2.2. Разложение временного ряда на компоненты

В Python выделить и визуализировать компоненты временного ряда можно с использованием функции seasonal_decompose() библиотеки statsmodels. Примеры использования данной функции будут рассмотрены позднее, в главе, посвященной анализу временных рядов.

2.2. Статистические параметров оценки временных рядов

Временные ряды могут характеризоваться рядом статистических параметров, рассмотрим наиболее важные из них.

Математическое ожидание. В литературе обозначают - E[X] или M[X] . Это среднее значение случайной величины.

Дисперсия случайной величины. Это мера разброса значений случайной величины относительно ее математического ожидания (среднего значения). Обозначается D[X] в русской литературе и Var(X) в зарубежной. Квадратный корень из дисперсии (в литературе обозначается – σ), называется стандартным отклонением.

Ковариация. Это мера совместной изменчивости двух случайных величин (обозначается - cov). Нормализованная версия ковариации (коэффициент корреляции) — это сила линейной зависимости двух величин.

Автокорреляция. Автокорреляция — это статистическая мера, которая описывает, насколько текущие значения временного ряда связаны с его предыдущими значениями. Проще говоря, автокорреляция показывает, есть ли закономерность в данных, то есть

повторяются ли значения ряда на каком-то временном интервале (лаге). Автокорреляция может быть положительной или отрицательной.

Положительная автокорреляция означает, что высокие значения временного ряда следуют за высокими значениями, а низкие — за низкими значениями. Например, если температура в один день высокая, то с большой вероятностью она будет высокой и на следующий день.

Отрицательная автокорреляция означает, что высокие значения временного ряда имеют тенденцию следовать за низкими значениями и наоборот.

Автокорреляция может варьироваться от «-1» до «1». Значение «1» говорит о полной положительной зависимости, «-1» — о полной отрицательной зависимости, а значение 0 указывает на отсутствие автокорреляции. Автокорреляция помогает определить, есть ли в данных цикличность, что важно для прогнозирования.

В статистическом моделировании временных рядов автокорреляция остатков (ошибок модели) используется для проверки качества модели. Если остатки модели имеют значимую автокорреляцию, это может указывать на то, что модель недостаточно хорошо описывает данные.

Здесь были показаны два достаточно схожих параметра: ковариация и автокорреляция. Однако, не смотря на схожесть, эти два параметра имеют ряд существенных отличий.

6. **Ковариация** измеряет степень линейной зависимости между двумя случайными переменными. Она показывает, как изменения одной переменной связаны с изменениями другой. А **автокорреляция** измеряет зависимость значений одного и того же временного ряда, но с разными временными отрезками (лагами). Это мера того, насколько текущее значение ряда связано с его прошлыми значениями.
7. **Ковариация** применяется для оценки зависимости между двумя разными переменными, тогда как **автокорреляция** используется для оценки зависимости между значениями одного и того же временного ряда на разных лагах.
8. **Ковариация** изучает взаимосвязь между двумя переменными, а **автокорреляция** — взаимосвязь между значениями одной переменной, но в разные моменты времени.
9. **Ковариация** дает представление о линейной зависимости между двумя переменными в общем случае, **автокорреляция** — о повторяемости (цикличности) во временном ряду.

2.3. Стационарность временных рядов

Временные ряды имеют очень важную характеристику – **стационарность**. Стационарность временного ряда важна, потому что многие методы анализа временных рядов предполагают именно это свойство. Если данные нестационарные, анализ может быть неточным или вообще не привести к достоверным результатам. Временной ряд считается стационарным, если он имеет следующие свойства:

- постоянное среднее значение (нет тенденции к повышению или снижению среднего значения);
- постоянная дисперсия (разброс точек данных относительно средней величины не меняется на протяжении всех периодов времени);
- постоянная структура автокорреляции (функция автокорреляции [ACF] или функция частичной автокорреляции [PACF] не должны существенно меняться со временем).

Примечание.

Функция автокорреляции (Autocorrelation Function - ACF) - представляет собой графическое представление корреляции временного ряда с самим собой при различных временных лагах. Частичная автокорреляционная функция (Partial Autocorrelation Function - PACF) - представляет собой графическое представление корреляции временного ряда с самим собой при различных лагах после удаления эффектов предыдущих лагов.

Понятие стационарного временного ряда означает, что его среднее значение не изменяется во времени, т.е. временной ряд не имеет тренда. Кроме того, ковариация между разными элементами временного ряда (как между случайными величинами) зависит только от того, насколько сильно они отдалены друг от друга во времени. Визуально отличие стационарного временного ряда от не стационарного представлено на рис. 2.3.

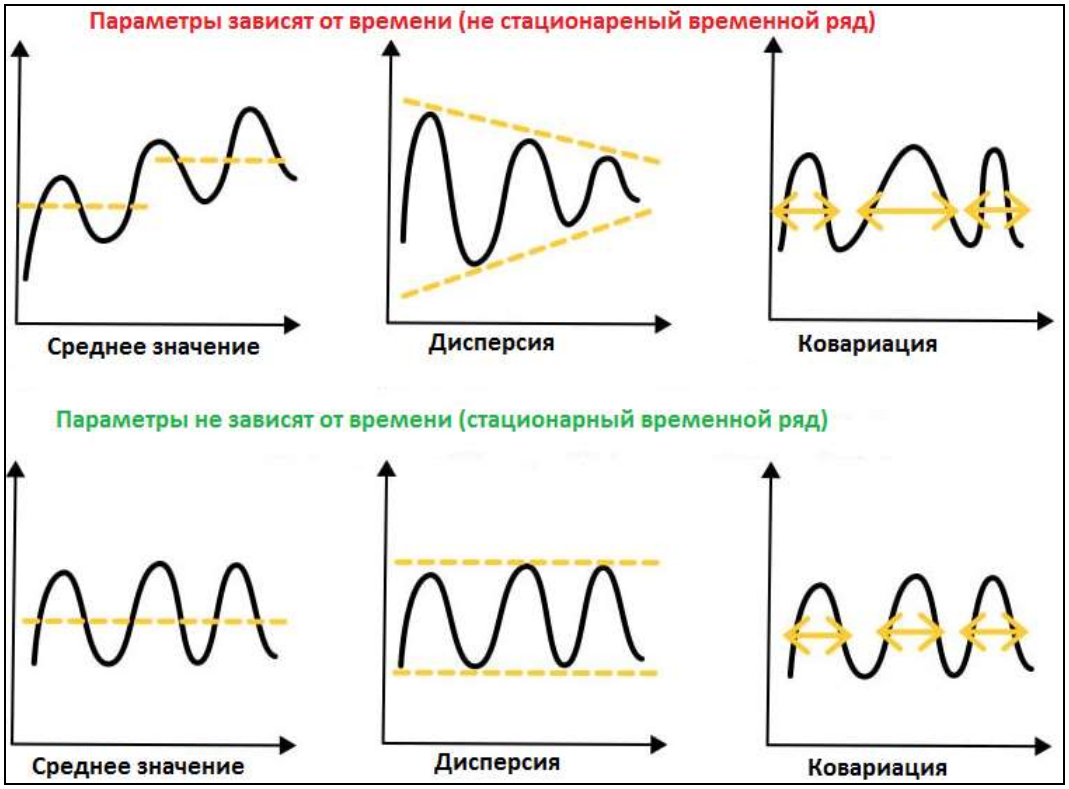


Рис. 2.3. Отличие стационарного временного ряда от не стационарного

Для проверки стационарности временного ряда используются различные методы, среди которых наиболее популярным является тест Дики-Фуллера

Тест Дики-Фуллера — это статистический тест, используемый для оценки стационарности временного ряда. При помощи этого теста проверяют значение коэффициента «a» в авторегрессионном уравнении первого порядка:

$$Yt = a * Yt_{-1} + Et$$

Где Yt — временной ряд, а Et — ошибка.
Если a=1, то процесс имеет единичный корень, в этом случае ряд Yt не стационарен.

В Python для проведения теста на стационарность используется функция adfuller() из библиотеки Statsmodels. На выходе данный тест выдает статистику Дики-Фуллера и контрольное «p» значение. Если «p» значение меньше уровня значимости (обычно 0,05), то можно отклонить нулевую гипотезу о не стационарности временного ряда и считать его стационарным.

> ***Примечание.***
> Практическое применение метода Дики-Фуллера будет рассмотрено в следующих главах на конкретных примерах.

Нестационарным называют временной ряд, свойства которого изменяются со временем. Это может проявляться в виде трендов, изменения дисперсии, сезонных колебаний или изменения автокорреляции. Нестационарные ряды чаще всего встречаются в реальной жизни. Временной ряд не будет стационарным если:

- присутствует тренд (например, постоянный рост температуры с течением времени, постоянное снижение спроса на товар);
- присутствует сезонность (повторяющиеся изменения с фиксированной периодичностью, например, сезонные колебания продаж мороженного);
- изменяющаяся дисперсия (колебания, амплитуда которых изменяется с течением времени).

Для целей прогнозирования на основе исторических данных временных рядов часто требуется преобразование нестационарного временного ряда в стационарный ряд. Для этого существует несколько методов, рассмотрим их.

2.4. Методы приведения временных рядов к стационарности

Преобразование нестационарного временного ряда в стационарный ряд является важным этапом анализа временных рядов по нескольким причинам. Дело в том, что многие стандартные методы анализа и моделирования временных рядов, такие как авто регрессионные модели (AR), модели скользящего среднего (MA), и их комбинации (ARMA, ARIMA), предполагают, что данные стационарны. Стационарность позволяет применять эти модели корректно и надежно. В стационарных рядах средние значения, дисперсия и автокорреляция постоянны во времени, что упрощает моделирование и интерпретацию.

Стационарные ряды позволяют решать задачи прогнозирования, так как их характеристики стабильны и не изменяются со временем. В нестационарных рядах, где могут наблюдаться тренды, сезонные эффекты или изменение дисперсии, точность прогнозов может быть значительно снижена.

Преобразование ряда к стационарности помогает лучше выявить истинные зависимости между переменными. Например, в нестационарных рядах могут возникать ложные корреляции между переменными, которые исчезают при переходе к стационарности.

В нестационарных рядах могут присутствовать тренды, которые создают видимость зависимости, которая на самом деле отсутствует. Приведение временного ряда стационарности позволяет устранить тренды и сконцентрироваться на реальных колебаниях и взаимосвязях.

В нестационарных рядах дисперсия может изменяться с течением времени, что затрудняет анализ. Преобразование в стационарный ряд стабилизирует дисперсию и делает анализ более надежным. Таким образом, преобразование нестационарного ряда в стационарный — это ключевой шаг, который позволяет применять стандартные методы анализа временных рядов, улучшает прогнозы и позволяет выявлять реальные закономерности и зависимости в данных. Существует несколько методов преобразования нестационарного временного ряда в стационарный ряд.

Дифференцирование. При этом методе выполняется процесс вычисления разностей между последовательными значениями временного ряда. Оно помогает устранить тренды и сделать ряд стационарным.

Логарифмирование. Если временной ряд имеет экспоненциальный рост или изменяющуюся дисперсию, можно применить логарифмирование. Это стабилизирует дисперсию и может помочь удалить экспоненциальные тренды. Логарифмирование также может быть полезным перед дифференцированием.

Декомпозиция. Данный метод делает разложение временного ряда на компоненты: тренд, сезонность и остатки (шум). После этого компоненты тренд и сезонность могут быть удалены для получения стационарного ряда.

Удаление тренда. Если временной ряд содержит только тренд (без сезонности), то можно просто вычесть трендовую линию из данных. Это может быть сделано путем регрессии на временной компоненте и вычитания прогнозируемых значений.

Сезонное дифференцирование. Если в данных присутствует сезонность, то ее можно удалить с помощью сезонного дифференцирования, где вычитается значение ряда, наблюдавшееся в предыдущем сезоне.

Применение скользящего среднего. Этот метод может использоваться для сглаживания временного ряда и удаления краткосрочных колебаний. Однако это может быть полезно только для устранения сезонных колебаний (этот метод не устраняет тренд).

Преобразование Бокса-Кокса. Это метод нелинейного преобразования данных, который может использоваться для стабилизации дисперсии и улучшения нормальности временного ряда.

Применение комбинации методов. Часто несколько методов могут применяться последовательно или в комбинации, например, логарифмирование, а затем дифференцирование, или дифференцирование, а затем удаление тренда.

2.5. Машинное обучение и модели прогнозирования временных рядов

Прогнозирование временных рядов — это статистический метод, используемый для прогнозирования будущих значений временного ряда на основе прошлых наблюдений. Проще говоря, это как заглянуть в будущее по точкам данных, отложенным во времени. Анализируя закономерности и тенденции в исторических данных, прогнозирование временных рядов помогает делать обоснованные прогнозы о том, что может произойти дальше, что способствует принятию решений и планированию на будущее.

Рассмотрим некоторые статистические модели, которые используются для прогнозирования временных рядов.

Модель AR (авторегрессионная модель). Это тип модели временных рядов, которая прогнозирует будущие значения на основе линейных комбинаций прошлых значений того же временного ряда. В модели AR текущее значение временного ряда моделируется как линейная функция его предыдущих значений плюс случайная погрешность. Порядок модели (p) показывает, сколько предыдущих значений ряда используется для прогноза текущего значения. Например, AR(2) означает, что для прогноза применяются два предыдущих значения

Модель MA (скользящего среднего). Это математический метод сглаживания временных рядов путем усреднения значений за определенный период. У данной модели следующий принцип работы: берется фиксированное число последовательных точек данных (окно), вычисляется их среднее значение, затем окно «скользит» вперед — исключается самая старая точка, добавляется новая, и среднее пересчитывается.

Для целей прогнозирования используются следующие виды моделей MA:

- **простая скользящая средняя** (Simple Moving Average, SMA). Рассчитывается как среднее арифметическое параметра за выбранный период;
- **экспоненциальная скользящая средняя** (Exponential Moving Average, EMA). Придает больший вес последним значениям параметра, что позволяет быстрее реагировать на изменения;
- **взвешенная скользящая средняя** (Weighted Moving Average, WMA). Присваивает данным линейно убывающие веса — чем старее данные, тем меньше их влияние на результат;
- **сглаженное скользящее среднее** (Smoothed Moving Average, SMMA). Учитывает более длинный период данных, минимизируя мелкие колебания параметра.

Модель ARMA (автогрегрессионная модель со скользящим средним. Эта модель объединяет свойства авторегрессионной модели (AR) и модели скользящего среднего (MA). Она используется для моделирования стационарных временных рядов, которые зависят как от своих предыдущих значений, так и от предыдущих ошибок.

Модель ARIMA (авторегрессионное интегрированное скользящее среднее). Модель ARIMA — это широко используемый статистический метод прогнозирования временных рядов. Он моделирует следующее значение временного ряда на основе линейной комбинации его собственных прошлых значений и прошлых ошибок прогнозирования. Параметры модели включают порядок авторегрессии (p), дифференцирование (d) и скользящее среднее (q).

Модель ARIMAX. Это расширение модели ARIMA за счет включения экзогенных переменных, которые могут повысить точность прогноза.

Модель SARIMA (сезонное авторегрессионное интегрированное скользящее среднее). Модель SARIMA это расширение модели ARIMA, за счет включения в модель сезонности. Она включает дополнительные сезонные параметры для отслеживания периодических колебаний в данных.

Модель SARIMAX. Это расширение модели SARIMA, за счет включения экзогенных переменных для прогнозирования сезонных временных рядов.

Более детальное представление об этих моделях вы получите в главе, связанной с прогнозированием временных рядов. А пока отметим, что все упомянутые выше

статистические модели перед использованием для прогнозирования временных рядов должны пройти «курс» машинного обучения на исторических данных, а для этого нужно выполнить кросс-валидацию временного ряда.

Кросс-валидация — это процедура для оценки качества работы модели, которая широко применяется в машинном обучении. Она помогает сравнить между собой различные модели и выбрать наилучшую модель для конкретной задачи. Эта процедура необходима, так как данные временных рядов отображают автокорреляцию, то есть значения точек данных связаны с их предыдущими значениями. Например, на основе данных о температуре в регионе с 1920 до 2021 года можно дать прогноз, какой будет погода в следующем году или через пять лет.

Перед обучением статистической модели временной ряд необходимо разделить на обучающую выборку и тестовую выборку. Главная сложность кросс-валидации на временных рядах заключается в том, что данные не должны пересекаться по времени: тренировочные данные должны идти строго до тестовых.

При обучении модели могут использоваться следующие методы кросс-валидации.

10. **Расширяющаяся (разрастающаяся) кросс-валидация (Expanding Window).** Модель обучается на увеличивающемся наборе данных, начиная с минимального объема и с добавлением новых данных на каждом шаге. Например, сначала модель обучается на первых двух годах данных, затем на трех годах и так далее. Каждый новый набор включает предыдущие данные плюс новый временной отрезок. Это имитирует реальный процесс, где мы прогнозируем будущее, опираясь на весь доступный прошлый опыт.
11. **Скользящее окно (Rolling Window).** Модель обучается на фиксированном объеме данных, который «скользит» по временной шкале, каждый раз оставляя самые старые данные и добавляя новые. Например, если окно длиной в два года, то на первом шаге модель обучается на данных за 1-й и 2-й годы и тестируется на данных за 3-й год. На втором шаге обучающая выборка сдвигается на один год: обучающая выборка: [данные за 2-й и 3-й годы], тестовая выборка: [данные за 4-й год]. Этот метод полезен, если предполагается, что данные могут изменяться со временем, и старые данные могут быть менее релевантными.
12. **Блочная кросс-валидация (Blocked Cross-Validation).** В этом подходе данные разбиваются на несколько блоков, и каждый блок поочередно используется в качестве тестовой выборки, в то время как остальные блоки служат обучающей выборкой. Важно, чтобы блоки шли последовательно во времени. Например, если данные разбиты на пять блоков, первый блок может быть тестовым, а остальные четыре — обучающими. Затем второй блок становится тестовым, а остальные — обучающими и так далее.

2.6. Краткие итоги

Итак, мы познакомились с некоторыми базовыми параметрами и характеристиками временных рядов, получили общее представление о статистических моделях, которые позволяют решать задачи анализа и прогнозирования временных рядов. Однако, для того, чтобы «потрогать» руками временной ряд и поработать с ним, его нужно получить. Есть несколько вариантов доступа к ременным рядам. Большинство компаний, которые занимаются коммерцией, имеют собственные базы данных, из которых можно получить любую информацию, развернутую во времени. Есть государственные, банковские и коммерческие структуры, которые собирают, структурируют и хранят большие массивы

данных, и предоставляют доступ к этой информации любым пользователям. Как правило, такие данные находятся в открытом доступе и могут быть получены через сеть интернет в виде набора данных DataFrame библиотеки Pandas. Ответ на вопрос, где и как можно получить данные из открытых источников будет подробно освещен в следующе главе, из который вы узнаете

- что такое API, и какие библиотеки Python используются для доступа к открытым источникам данных;
- как можно загрузить данные через API Центрального банка России;
- как можно загрузить данные через API Всемирного банка;
- как загружать данные из различных открытых источников с библиотекой pandas_datareader;
- как получить доступ к финансовым и экономическим данным через платформу Yahoo Finance.

Глава 3. Использование API для доступа к открытым данным

Аббревиатура API является сокращением английского термина Application Programming Interface (программный интерфейс приложения). По сути, API действует как коммуникационный уровень или интерфейс, который позволяет различным системам взаимодействовать друг с другом без необходимости точно понимать, что делает каждая из систем.

API-интерфейсы имеют разные формы. Это может быть API операционной системы, для включения камеры и микрофона при подключении, например, к Zoom. Или это могут быть веб-API, используемые для действий, ориентированных на социальные сети, таких как «Одноклассники» или «В контакте». Интерфейсы API могут обеспечивать подключение к удаленным базам данных для доступа к хранимой там информации.

Независимо от типа, все API-интерфейсы работают приблизительно одинаково. Обычно программа-клиент запрашивает информацию или данные, а API возвращает ответ в соответствии с тем, что было запрошено. В этой главе мы подробно остановимся на веб-API, которые обеспечивают доступ к публичным базам данных.

В частности из материалов этой главы вы узнаете:

- Об основных принципах использования API с Python;
- как можно загружать данные через API Центрального банка России;
- как можно загружать данные через API Всемирного банка с библиотекой wbdata;
- как можно получить доступ к открытым источникам данных с библиотекой pandas_datareader;
- как можно получить доступ к открытым данным финансовой платформы Stooq;
- как можно получить доступ к открытым данным финансовой платформы Yahoo Finance.

3.1. Основные принципы использования API в Python

При использовании API с Python нам понадобится всего одна библиотека - requests. С ее помощью можно выполнять практически все действия, необходимые для использования любого общедоступного API. Библиотека requests не является частью Python, она сторонняя. В Python есть встроенный модуль urllib3, который обладает сходным функционалом, но requests чаще используют большинство разработчиков из-за удобства и простоты работы с HTTP-запросами. Установите библиотеку можно с помощью pip:

```
pip install requests
```

После установки библиотеки requests сделаем первый тестовый вызов API - вызовем общедоступный API для генерации случайных пользовательских данных. Единственное, что нужно знать для работы с API — это его URL адрес. В этом примере будет использован следующий URL адрес: «https://randomuser.me/api/» (листинг 3.1).

Листинг 3.1. Модуль my_api.py

Примечание.
В демо-версии книги листинги программ удалены.

В этом программном модуле выполнен импорт библиотеки requests. Затем с использованием метода requests.get выполнен запрос (обращение) к API, а полученные данные сохранены в объекте data.

Если выполнить данный программный код, то будет получен следующий результат.

```
<Response [200]>
{"results":[{"gender":"male","name":{"title":"Mr","first":"Alexander","last":"Rasmussen"},"location":{"street":{"number":4391,"name":"Nordre Ringvej"},"city":"Bælum","state":"Syddanmark","country":"Denmark","postcode":95434,"coordinates":{"latitude":"-39.5657","longitude":"-6.3778"},"timezone":{"offset":"0:00","description":"Western Europe Time, London, Lisbon, Casablanca"}},"email":"alexander.rasmussen@example.com","login":{"uuid":"0d7792ff-62d6-4c9b-b555-95b921993ff7","username":"greengoose638","password":"furball","salt":"AiZxWKSn","md5":"07c443731bdf94b504aecbbc39d91ec5","sha1":"80ff9555fb0ef00e5b2243ab05ab439870bb1161","sha256":"08201327fb1f659240ebe8862796de788fdc4e71acc053fe122a667713c71c2f"},"dob":{"date":"1964-05-09T17:14:34.350Z","age":61},"registered":{"date":"2016-10-19T00:40:47.077Z","age":8},"phone":"10552994","cell":"42484415","id":{"name":"CPR","value":"090564-1707"},"picture":{"large":"https://randomuser.me/api/portraits/men/56.jpg","medium":"https://randomuser.me/api/portraits/med/men/56.jpg","thumbnail":"https://randomuser.me/api/portraits/thumb/men/56.jpg"},"nat":"DK"}],"info":{"seed":"87bbe3550d306708","results":1,"page":1,"version":"1.4"}}
```

Примечание.
Представленный в данном разделе программный модуль не включен в список исполняемых примеров интерактивной цифровой книги, так как он может работать только в рамках приложения Python локального компьютера, или быть вызванным из другого модуля веб приложения.

Инструкция print(data) вывела строку - <Response [200]>. В терминах API такой результат означает, что все прошло нормально. Чтобы увидеть фактические данные, нужно к имени переменной data добавить атрибут «.text» - print(data.text). Как видно из полученных результатов, в ответе был возвращен словарь с набором данных. В данном случае нас не очень интересует содержание ответа, важен сам факт успешного получения данных через API. Теперь мы сможем использовать базовые принципы работы с API для получения реальных данных из различных источников.

3.2. Доступ к открытым данным из библиотеки Pandas

Библиотека Pandas позволяет напрямую загружать данные через API открытых источников в формате XML с использованием метода pd.read_xml(url), и преобразовывать их в объект DataFrame. Этот метод принимает URL-адрес источника данных в качестве параметра и возвращает структурированный набор данных DataFrame. При этом обязательным условием является то, что API открытого источника данных должен обеспечивать интерфейс обмена данных в формате XML.

3.2.1. Получение данных о курсах валют из API Центрального банка

Рассмотрим пример получения данных из API Центрального банка России о динамике курсов валют с использованием библиотеки Pandas. Для этого сначала разберемся со структурой URL адреса API Центрального банка, через который можно получить данные о курсах валют:

```
https://www.cbr.ru/scripts/XML_dynamic.asp?date_req1=10/01/2022&date_req2=20/01/2022&VAL_NM_RQ=R01235
```

Это URL адрес можно разбить на следующие фрагменты:

- https://www.cbr.ru/scripts/XML_dynamic.asp? – заголовок URL адреса;
- date_req1=10/01/2022 – дата начала периода;
- date_req2=20/01/2022 – дата конца периода;
- VAL_NM_RQ=R01235 – код пары валют (руб./$).

Для лучшей читаемости программного кода желательно URL адрес разбить на фрагменты, присваивать данные каждому фрагменту по отдельности, а потом объединить все фрагменты URL адреса в один объект.

Рассмотрим пример, в котором получим данные об изменении курса валют (руб./$) за заданный период. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 3.2).

Листинг 3.2. Ссылка активации маршрута cuts_r_cb

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем cuts_r_cb (листинг 3.3).

Листинг 3.3. Маршрут URL с именем cuts_r_cb

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление cuts_r_cb (листинг 3.4).

Листинг 3.4. Представление (view) cuts_r_cb

Примечание.
В демо-версии книги листинги программ удалены.

В первой строке этого программного кода выполняется импорт библиотеки Pandas. После этого сформированы фрагменты URL адреса API Центрального банка:

- url_cb = 'https://www.cbr.ru/scripts/XML_dynamic.asp?' – заголовок URL адреса;
- date_req1 = 'date_req1=' – переменная для обозначения даты начала периода;
- d1 = '01/01/2020' – дата начала периода;
- date_req2 = '&date_req2=' - переменная для обозначения даты конца периода;
- d2 = '12/04/2025' - дата конца периода;
- VAL_NM_RQ = '&VAL_NM_RQ=' переменная для обозначения кода пары валют;
- kod_val = 'R01235' – код пары валют (руб./$).

На этом этапе, по сути, мы разложили URL адрес API Центрального банка на постоянные и переменные фрагменты. Переменными фрагментами являются: d1 – дата начала периода, d2 - дата конца периода, kod_val – код пары валют. Остальные параметры являются постоянными и их можно не менять.
После этого был сформирован полный URL адрес для запроса к API Центрального банка:

```
url = url_cb + date_req1 + d1 + date_req2 + d2 + VAL_NM_RQ + kod_val
```

Затем выполнен запрос к API Центрального банка с загрузкой результатов в объект data.

```
data = pd.read_xml(url)
```

Объект data будет содержать набор данных DataFrame, состоящий из нескольких тысяч записей. Для того чтобы показать только первые и последние записи этого набора данных сформировано два объекта (df_1 и df_2). Затем наборы данных df_1 и df_2 преобразованы в таблицу HTML, вместе с датами начала и конца периода загружены в контекстную переменную context и переданы в шаблон с именем cuts_r_cb.html. Этот шаблон имеет следующий код (листинг 3.5).

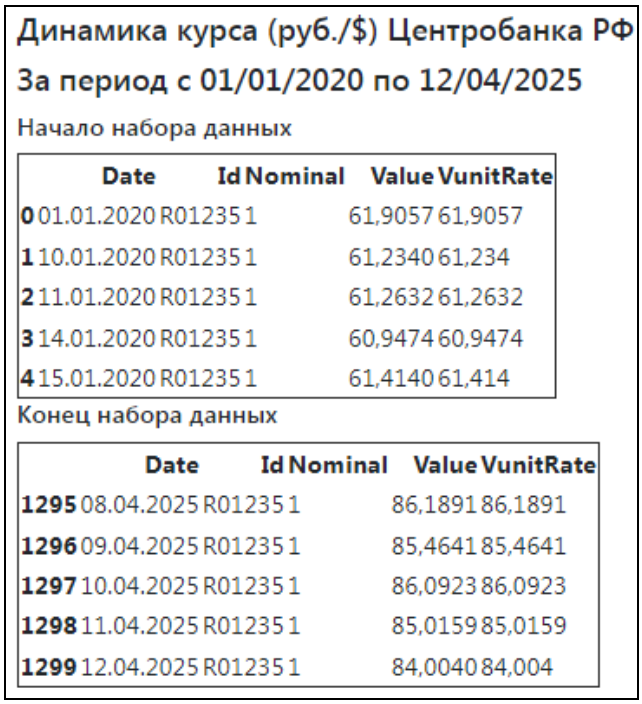
Листинг 3.5. Шаблон cuts_r_cb.html

Примечание.
В демо-версии книги листинги программ удалены.

Это достаточно простой шаблон с контейнером, в котором выведены объекты ds_1 и df_2. Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 3» и активировать ссылку «Курс (руб./$) ЦБ РФ».

Если выполнить набор вышеприведенных модулей, то получим следующий результат (рис. 3.1).



Динамика курса (руб./$) Центробанка РФ

За период с 01/01/2020 по 12/04/2025

Начало набора данных

	Date	Id	Nominal	Value	VunitRate
0	01.01.2020	R01235	1	61,9057	61,9057
1	10.01.2020	R01235	1	61,2340	61,234
2	11.01.2020	R01235	1	61,2632	61,2632
3	14.01.2020	R01235	1	60,9474	60,9474
4	15.01.2020	R01235	1	61,4140	61,414

Конец набора данных

	Date	Id	Nominal	Value	VunitRate
1295	08.04.2025	R01235	1	86,1891	86,1891
1296	09.04.2025	R01235	1	85,4641	85,4641
1297	10.04.2025	R01235	1	86,0923	86,0923
1298	11.04.2025	R01235	1	85,0159	85,0159
1299	12.04.2025	R01235	1	84,0040	84,004

Рис. 3.1. Данные о динамике курса валют, полученные из API Центрального банка РФ

Как видно из данного рисунка, полученный набор данных содержит 4 колонки и 1299 строк. Конечно такой большой массив данных, представленных в виде таблицы, достаточно трудно анализировать, и его нужно преобразовать к графическому виду, однако, как трансформировать таблицу в график будет подробно рассмотрено в следующей главе.

Для того, чтобы получить данные по обменному курсу других валют, необходимо знать соответствующие коды. Коды, с помощью которых можно получить динамику курса других валют, приставлены в табл.3.1.

Таблица 3.1.

Коды обменного курса некоторых валют API Центрального банка России

№ пп	Код	Валюта
1	R01035	фунт стерлингов
2	R01235	доллар США
3	R01239	евро
4	R01305	ирландский фунт
5	R01315	испанская песета

6	R01325	итальянская лира
7	R01375	китайский юань
8	R01820	японская иена

С полным перечнем кодов валют можно познакомиться на сайте Центрального банка.

3.2.2. Получение данных о ценах на драгметаллы из API Центрального банка

Рассмотрим пример получения данных из API Центрального банка России о динамике цен на драгоценные металлы с использованием библиотеки Pandas. Для этого сначала разберемся со структурой URL адреса API Центрального банка, через который можно получить данные о ценах на драгоценные металлы:

http://www.cbr.ru/scripts/xml_metall.asp?date_req1=01/07/2024&date_req2=13/07/2025

Это URL адрес можно разбить на следующие фрагменты:

- http://www.cbr.ru/scripts/xml_metall.asp? – заголовок URL адреса;
- date_req1=01/07/2024– дата начала периода;
- date_req2=13/07/2025– дата конца периода.

Для лучшей читаемости программного кода желательно URL адрес разбить на фрагменты, присваивать данные каждому фрагменту по отдельности, а потом объединить все фрагменты URL адреса в один объект.

Рассмотрим пример, в котором получим данные об изменении стоимости драгметаллов за заданный период. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 3.6).

Листинг 3.6. Ссылка активации маршрута curs_dm_cb

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем curs_dm_cb (листинг 3.7).

Листинг 3.7. Маршрут URL с именем curs_dm_cb

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление curs_dm_cb (листинг 3.8).

Листинг 3.8. Представление (view) curs_dm_cb

Примечание.
В демо-версии книги листинги программ удалены.

В первых строках этого программного кода выполняется импорт необходимых библиотек. Затем были сформированы даты начала и конца периода (объекты d1 и d2). Здесь объекту d1 задано значение даты на 5 лет ранее текущей даты, а объекту d2 задано значение даты текущей даты. Таким образом, был выбран период последние пять лет, начиная от текущей даты.

После этого сформированы фрагменты URL адреса API Центрального банка:

- url_cb = 'http://www.cbr.ru/scripts/xml_metall.asp?' – заголовок URL адреса;
- date_req1 = 'date_req1=' – переменная для обозначения даты начала периода;
- date_req2 = '&date_req2=' - переменная для обозначения даты конца периода.

На этом этапе, по сути, мы получили постоянные фрагменты URL адрес API Центрального банка.
После этого был сформирован полный URL адрес для запроса к API Центрального банка:

```
url = url_cb + date_req1 + d1_str + date_req2 + d2_str
```

Затем выполнен запрос к API Центрального банка с загрузкой результатов в объект df.

```
df = pd.read_xml(url)
```

Объект df будет содержать набор данных DataFrame, состоящий из нескольких тысяч записей. Для того чтобы показать только первые и последние записи этого набора данных сформировано два объекта (df_1 и df_2). Затем наборы данных df_1 и df_2 преобразованы в таблицу HTML, вместе с датами начала и конца периода загружены в контекстную переменную context и переданы в шаблон с именем cuts_dm_cb.html. Этот шаблон имеет следующий код (листинг 3.9).

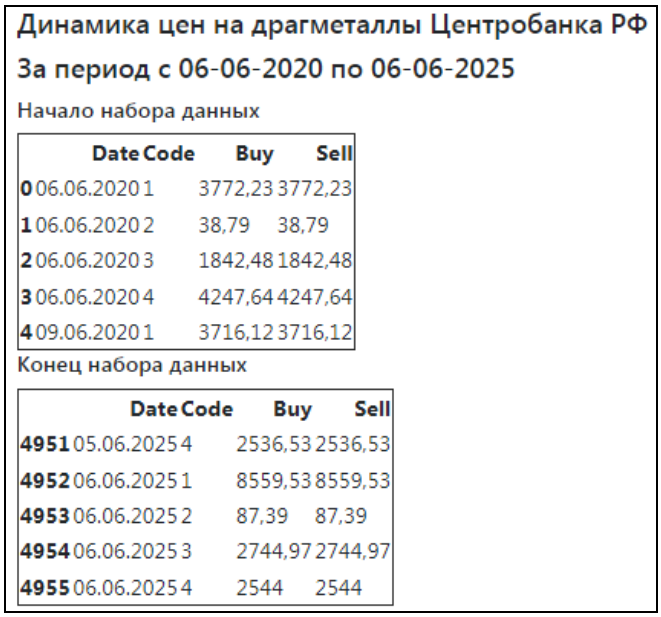
Листинг 3.9. Шаблон cuts_dm_cb.html

Примечание.
В демо-версии книги листинги программ удалены.

Это достаточно простой шаблон с контейнером, в котором выведены объекты ds_1 и df_2. Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 3» и активировать ссылку «Цены на драгметаллы ЦБ РФ».

Если выполнить набор вышеприведенных модулей, то получим следующий результат (рис. 3.2).



Динамика цен на драгметаллы Центробанка РФ

За период с 06-06-2020 по 06-06-2025

Начало набора данных

	Date	Code	Buy	Sell
0	06.06.2020	1	3772,23	3772,23
1	06.06.2020	2	38,79	38,79
2	06.06.2020	3	1842,48	1842,48
3	06.06.2020	4	4247,64	4247,64
4	09.06.2020	1	3716,12	3716,12

Конец набора данных

	Date	Code	Buy	Sell
4951	05.06.2025	4	2536,53	2536,53
4952	06.06.2025	1	8559,53	8559,53
4953	06.06.2025	2	87,39	87,39
4954	06.06.2025	3	2744,97	2744,97
4955	06.06.2025	4	2544	2544

Рис. 3.2. Данные о динамике цен на драгметаллы, полученные из API Центрального банка РФ

Как видно из данного рисунка, полученный набор данных содержит 4 колонки и 4955 строк. Конечно такой большой массив данных, представленных в виде таблицы, достаточно трудно анализировать, и его нужно преобразовать к графическому виду. В этом наборе данных 4 колонки: Date – дата, Code – код металла, Buy – цена покупки, Sell – цена продажи. Коды, которые относятся к тому или иному драгоценному металлу, приставлены в табл.3.2.

Таблица 3.2.

Коды драгоценных металлов в API Центрального банка России

№ пп	**Код**	**Драгоценный металл**
1	1	Золото
2	2	Серебро
3	3	Платина
4	4	Палладий

3.3. Доступ к открытым данным из API Всемирного банка с библиотекой wbdata

Анализ экономических данных имеет решающее значение для ряда приложений: от разработки государственной политики до финансового прогнозирования и академических исследований. Открытое для общего пользования API Всемирного банка предоставляет богатый набор данных к тысячам экономических показателей из более чем 200 стран. Эти данные можно свободно скачивать, используя язык программирования Python. Для этого Всемирный банк разработал специальную библиотеку wbdata.

Библиотека wbdata — это простой интерфейс Python для поиска и запроса информации из различных баз данных Всемирного банка, как в виде словаря, содержащего полные метаданные, так и в виде набора данных DataFrame. Библиотека wbdata охватывает большую часть API Всемирного банка, а также добавляет некоторые удобные функции для поиска и извлечения информации.

Библиотека wbdata обеспечивает следующие возможности:

- получение более 16 000 экономических показателей;
- фильтрация данных по стране, дате и конкретным показателям;
- простая интеграция с библиотекой pandas;
- возвращает данные в формате JSON, идеально подходящем для программной обработки на языке программирования Python.

Для установки библиотеки wbdata можно использовать следующую команду:

```
pip install wbdata
```

Библиотек wbdata абстрагирует вызовы API, упрощая запросы к наборам данных Всемирного банка с помощью всего нескольких строк программного кода. Она использует словари Python для индикаторов и кодов стран (формат ISO 3166-1 alpha-2) и поддерживает запросы на основе даты.

3.3.1. Модуль Python запроса к данным через API Всемирного банка

Рассмотрим применение библиотеки wbdata для получения экономических показателей различных стран. Для начала создадим программный модуль Python, который формирует запрос к API Всемирного банка. Этот модуль позволит отделить логику формирования запроса к данным Всемирного банка от логики работы приложения Django. Программный код формирования запросов к API Всемирного банка представлен в листинге 3.10.

Листинг 3.10. Программный модуль fn_wb_data.py

Примечание.
В демо-версии книги листинги программ удалены.

В данном модуле выполнен импорт необходимых библиотек и создана функция с именем get_data_wb(). Эта функция принимает следующие параметры:

- indicator – индикатор или финансовый показатель в виде словаря;
- countries – перечень стран в виде списка;
- s_date – дата начала периода в виде текстовой строки в европейском формате;
- e_date - дата конца периода в виде текстовой строки в европейском формате.

В теле функции даты начала и конца периода из текстовой строки преобразуются в тип данных datetime американского формата. Затем с использованием API делается запрос к базе данных Всемирного банка, а результаты сохраняются в объекте dfr (формат DataFrame библиотеки Pandas). Объект dfr возвращается в точку вызова.

В этом же программном модуле выполнено тестовое обращение к данной функции, в которую переданы следующие параметры:

- dic_indic = {'NY.GDP.MKTP.CD': 'GDP (current US$)'}- индикатор экономического показателя (ВВП);
- list_country = ['CN', 'US', 'RUS', 'GB'] - список стран (в виде кодов стран);
- d1_str = '01-01-2019' – дата начала периода;
- d2_str = '17-04-2025' – дата конца периода.

Здесь коды стран представлены в виде списка со строковыми значениями, а индикатор (или финансовый показатель) задается в виде словаря. Даты начала и конца периода заданы в виде строк в европейском формате. В теле функции эти даты будут преобразованы в тип данных datetime американского формата.

Если выполнить данный программный код, то будет получен следующий результат.

		GDP (current US$)
country	date	
China	2024-01-01	NaN
	2023-01-01	1.779478e+13
	2022-01-01	1.788178e+13
	2021-01-01	1.782046e+13
	2020-01-01	1.468774e+13
	2019-01-01	1.427997e+13
United Kingdom	2024-01-01	NaN
	2023-01-01	3.380855e+12
	2022-01-01	3.114042e+12
	2021-01-01	3.143323e+12
	2020-01-01	2.696778e+12
	2019-01-01	2.851407e+12
Russian Federation	2024-01-01	NaN
	2023-01-01	2.021421e+12
	2022-01-01	2.266029e+12
	2021-01-01	1.843392e+12
	2020-01-01	1.493076e+12
	2019-01-01	1.693115e+12
United States	2024-01-01	NaN
	2023-01-01	2.772071e+13

```
2022-01-01    2.600689e+13
2021-01-01    2.368117e+13
2020-01-01    2.135410e+13
2019-01-01    2.153998e+13
```

Примечание.
Представленный в данном разделе программный модуль может работать только на локальном компьютере в приложении на Python.

В этом наборе данных 4 колонки: country (страна), date (дата), GDP (ВВП), при этом видно, что данные за 2024 год отсутствуют (значение NaN).

Для того чтобы сформировать запрос к API Всемирного банка нужно знать коды стран и коды индикаторов финансовых показателей. Коды стран можно получить с использованием следующей команды:

```
print(wb.country_codes)
```

Список кодов стран приведен в приложении 1, а перечень некоторых индикаторов финансовых показателей приведен в приложении 2.

3.3.2. Запроса к данным через API Всемирного банка в приложениях Django

Теперь рассмотрим пример, в котором получим данные об экономических показателях некоторых стран в приложении на Django. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 3.11).

Листинг 3.11. Ссылка активации маршрута wb_vvp_tabl

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем wb_vvp_tabl (листинг 3.12).

Листинг 3.12. Маршрут URL с именем wb_vvp_tabl

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление wb_vvp_tabl (листинг 3.13).

Листинг 3.13. Представление (view) wb_vvp_tabl

Примечание.
В демо-версии книги листинги программ удалены.

В этом представлении выполнен импорт модуля загрузки данных (fn_wb_data), который был описан в предыдущем разделе. Задан словарь индикатора финансовых показателей (dic_indic). Сформирован список стран (list_country). Заданы даты начала и конца периода (d1_str, d2_str). Затем выполнено обращение к функции загрузки данных через API Всемирного банка, полученные данные переданы объекту df и преобразованы к формату для отображения на HTML странице (объект df_html). После этого сформирована контекстная переменная context, которая передана в шаблон wb_vvp_tabl.html (листинг 3.14).

Листинг 3.14. Шаблон wb_vvp_tabl.html

Примечание.
В демо-версии книги листинги программ удалены.

Это достаточно простой шаблон с контейнером, в котором выведен объект df_html. Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 3» и активировать ссылку «ВВП стран (таблица)».

Если выполнить набор вышеприведенных модулей, то получим следующий результат (рис. 3.3).

Валовый внутренний продукт
по данным Всемирного банка, ($)
За период с 01-01-2019 по 17-04-2025

		GDP (current US$)
country	**date**	
China	**2024-01-01**	NaN
	2023-01-01	1.779478e+13
	2022-01-01	1.788178e+13
	2021-01-01	1.782046e+13
	2020-01-01	1.468774e+13
	2019-01-01	1.427997e+13
United Kingdom	**2024-01-01**	NaN
	2023-01-01	3.380855e+12
	2022-01-01	3.114042e+12
	2021-01-01	3.143323e+12
	2020-01-01	2.696778e+12
	2019-01-01	2.851407e+12
Russian Federation	**2024-01-01**	NaN
	2023-01-01	2.021421e+12
	2022-01-01	2.266029e+12
	2021-01-01	1.843392e+12
	2020-01-01	1.493076e+12
	2019-01-01	1.693115e+12
United States	**2024-01-01**	NaN
	2023-01-01	2.772071e+13
	2022-01-01	2.600689e+13
	2021-01-01	2.368117e+13
	2020-01-01	2.135410e+13
	2019-01-01	2.153998e+13

Рис. 3.3. Данные о ВВП, полученные из API Всемирного банка

В этом наборе данных 4 колонки: country (страна), date (дата), GDP (ВВП), при этом данные о ВВП представлены в экспоненциальной форме. Данные, представленные в таком виде, не очень информативны, и лучше бы показать их в графическом виде, но структура представленной таблицы не позволяет этого сделать. Полученные данные нужно преобразовать таким образом, чтобы показатель каждой страны располагался в отдельной колонке. Для этого в библиотеке pandas имеется метод pivot (поворот), позволяющий сформировать сводную таблицу. Воспользуемся этим методом.

3.3.3. Запроса к данным через API Всемирного банка в приложениях Django с формированием сводной таблицы

В библиотеке pandas есть метод pivot, который позволяет изменить форму представления данных, преобразуя строки в столбцы. С его помощью можно перестроить данные из «длинного, вертикального» формата в «широкий, горизонтальный» формат. По сути, определенные значения строк преобразуются в заголовки столбцов, что позволяет получить более понятное визуальное восприятие данных. Но главное достоинство этого метода заключается в том, что преобразованные данные пригодны для построения графиков. Сделаем такое преобразование с данными ВВП ряда стран. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 3.15).

Листинг 3.15. Ссылка активации маршрута wb_vvp_pivot

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем wb_vvp_pivot (листинг 3.16).

Листинг 3.16. Маршрут URL с именем wb_vvp_pivot

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление wb_vvp_pivot (листинг 3.17).

Листинг 3.17. Представление (view) wb_vvp_pivot

Примечание.
В демо-версии книги листинги программ удалены.

В этом представлении выполнен импорт модуля загрузки данных (fn_wb_data), код которого описан в начала данного раздела. Задан словарь индикатора финансовых показателей (dic_indic). Сформирован список стран (list_country). Заданы даты начала и конца периода (d1_str, d2_str). Затем выполнено обращение к функции загрузки данных через API Всемирного банка, а полученные данные переданы объекту df.

Далее выполняется формирование сводной таблицы. Из объекта df удаляется текущий индекс - df = df.reset_index(), после чего с использованием метода pivot формируется сводная таблица:

```
df_pivot = df.pivot(index='date', columns='country', values='GDP (current US$)')
```

Здесь заданы следующие параметры:

- index='date' – даты становится индексами, это будет первая колонка, по которой сортируются все данные;
- columns='country' – названия стран станут заголовками колонок;
- values='GDP (current US$)' – данные о ВВП будут являться значениями ячеек, то есть выводиться на пересечении строки и столбца.

После этого данные сводной таблицы (объект df_pivot) переводятся в триллионы и преобразуются к формату данных, пригодных для отображения на HTML странице (объект df_html). После этого сформирована контекстная переменная context, которая передана в шаблон wb_vvp_pivot.html (листинг 3.18).

Листинг 3.18. Шаблон wb_vvp_pivot.html

Примечание.
В демо-версии книги листинги программ удалены.

Это достаточно простой шаблон с контейнером, в котором выведен объект df_html. Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 3» и активировать ссылку «ВВП стран (табл. pivot)».

Если выполнить набор вышеприведенных модулей, то получим следующий результат (рис. 3.4.).

Валовый внутренний продукт (сводная таблица)
по данным Всемирного банка, ($)
За период с 01-01-2000 по 17-04-2025

country	China	Russian Federation	United Kingdom	United States
date				
2000-01-01	1.211332	0.259710	1.665535	10.250952
2001-01-01	1.339401	0.306602	1.649827	10.581929
2002-01-01	1.470558	0.345470	1.785730	10.929108
2003-01-01	1.660281	0.430347	2.054423	11.456450
2004-01-01	1.955347	0.591017	2.421525	12.217196
2005-01-01	2.285961	0.764016	2.543180	13.039197
2006-01-01	2.752119	0.989932	2.708442	13.815583
2007-01-01	3.550328	1.299703	3.090510	14.474228
2008-01-01	4.594337	1.660848	2.929412	14.769862
2009-01-01	5.101691	1.222646	2.412840	14.478067
2010-01-01	6.087192	1.524917	2.485483	15.048971
2011-01-01	7.551546	2.045923	2.663806	15.599732
2012-01-01	8.532185	2.208294	2.707090	16.253970
2013-01-01	9.570471	2.292470	2.784854	16.880683
2014-01-01	10.475625	2.059242	3.064708	17.608138
2015-01-01	11.061573	1.363482	2.927911	18.295019
2016-01-01	11.233314	1.276786	2.689107	18.804913
2017-01-01	12.310491	1.574199	2.680148	19.612102
2018-01-01	13.894908	1.657329	2.871340	20.656516
2019-01-01	14.279969	1.693115	2.851407	21.539982
2020-01-01	14.687744	1.493076	2.696778	21.354105
2021-01-01	17.820460	1.843392	3.143323	23.681171
2022-01-01	17.881783	2.266029	3.114042	26.006893
2023-01-01	17.794783	2.021421	3.380855	27.720709
2024-01-01	NaN	NaN	NaN	NaN

Рис. 3.4. Данные о ВВП, полученные из API Всемирного банка в виде сводной таблицы

В этой таблице 5 колонок. Первая колонка это даты, отсортированные по возрастанию. В следующих 4-х колонках находятся данные по ВВП для каждой страны. Сформированные таким образом данные сводной таблицы могут быть представлены в графическом виде, что более удобно для их визуального анализа.

3.4. Доступ к открытым данным с библиотекой pandas_datareader

Среди разработчиков программного обеспечения и аналитиков данных большой популярностью пользуется библиотека pandas_datareader, которая позволяет читать данные из различных открытых источников в формате DataFrame. Этот инструментарий расширяет функциональность библиотеки pandas, обеспечивая доступа к различным открытым финансовым и экономическим источникам данных: Yahoo Finance, Google Finance, World Bank, Stooq т.п. С перечнем доступных источников данных можно познакомиться по следующей ссылке:
https://pandas-datareader.readthedocs.io/en/latest/remote_data.html

При использовании данной библиотеки разработчики получают следующие преимущества:

- поддержка множества источников данных (библиотека работает с данными, доступными через API различных открытых баз данных);
- кэширование запросов для оптимизации производительности;
- полная интеграция с pandas (данные, полученные с помощью pandas_datareader, можно анализировать и манипулировать, используя методы и функции библиотеки pandas)

Установить библиотеку pandas_datareader можно с использованием следующей команды:

```
pip install pandas-datareader
```

3.4.1. Модуль Python запроса к данным с библиотекой pandas_datareader

В библиотеке pandas_datareader есть модуль wb, который обеспечивает доступ к API Всемирного банка и загрузку экономических показателей различных стран. Выполнить импорт этого модуля можно с использованием следующей инструкции:

```
from pandas_datareader import wb
```

Реализуем пользовательскую функцию, которая будет загружать данные Всемирного банка с использованием модуля wb библиотеки pandas_datareader (модуль pand_dr_wb.py, листинг 3.19.).

Листинг 3.19. Программный модуль pand_dr_wb.py

> ***Примечание.***
> В демо-версии книги листинги программ удалены.

В этом модуле выполнен импорт модуля wb библиотеки pandas_datareader и создана функция def get_dr_wb(), которая принимает следующие параметры:

- indic - индикатор или финансовый показатель в виде словаря;
- countries – перечень стран в виде списка;
- start - дата начала периода (год в виде целого числа);

- end - дата конца периода (год в виде целого числа).

После этого из словаря (indic) извлекается ключ (ind_key) и значение ключа (ind_val). Эти параметры понадобятся для извлечения и отображения данных. После этого с использованием метода wb.download информация извлекается через API Всемирного банка и сохраняется в объекте dat. Так как полученные данные планируется выводить в графическом виде, то они в теле функции трансформируются в сводную таблицу (объект df_pivot). Сводная таблица (pivot) и значение ключа финансового показателя (ind_val) возвращаются в точку вызова.

В этом же программном модуле выполнено тестовое обращение к данной функции, в которую переданы следующие параметры:

- dic_indic = {'NE.EXP.GNFS.ZS': 'Exports'}- индикатор экономического показателя (экспорт);
- list_country = ['CN', 'US', 'RUS', 'GB'] - список стран (в виде кодов стран);
- start_d = 2015 – год начала периода;
- end_d = 2023 – год конца периода.

Здесь коды стран представлены в виде списка со строковыми значениями, а индикатор (или финансовый показатель) задается в виде словаря. Даты (годы) начала и конца периода заданы в виде целых чисел.

Если данный модуль запустить на выполнение, то будет получен следующий результат.

country	China	Russian Federation	United Kingdom	United States
year				
2015	21.354080	28.703389	27.845951	12.411149
2016	19.584380	25.854483	28.902340	11.888159
2017	19.692277	26.090881	30.890351	12.177481
2018	19.112104	30.793257	31.580111	12.287111
2019	18.409992	28.433431	31.634780	11.789197
2020	18.586139	25.522186	29.669796	10.073656
2021	19.943974	29.771214	29.180275	10.790978
2022	20.791483	28.025719	33.567194	11.602228
2023	19.743072	23.083003	31.662677	11.011511
Exports				

> ***Примечание.***
> Представленный в данном разделе программный модуль может работать только на локальном компьютере в приложении на Python.

В этом наборе данных 5 колонок: year (год), и 4 колонки с данными об экспорте товаров разных стран. Так как в теле функции из набора данных сформирована сводная таблица, то возвращаемый набор данных полностью готов для его отображения в графическом виде.

3.4.2. Запроса к данным с библиотекой pandas_datareader в приложениях Django

Теперь рассмотрим пример, в котором получим данные о показателях экспорта разных стран в приложении на Django. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 3.20).

Листинг 3.20. Ссылка активации маршрута wb_export_dr_pivot

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем wb_export_dr_pivot (листинг 3.21).

Листинг 3.21. Маршрут URL с именем wb_export_dr_pivot

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление wb_export_dr_pivot (листинг 3.22).

Листинг 3.22. Представление (view) wb_export_dr_pivot

Примечание.
В демо-версии книги листинги программ удалены.

В этом представлении выполнен импорт модуля загрузки данных с использованием библиотеки data_reader (pand_dr_wb), код которого приведен в предыдущем разделе. Задан словарь индикатора финансовых показателей (dic_indic). Сформирован список стран (list_country). Заданы даты начала и конца периода в виде целых чисел (d1_int, d2_int). Затем выполнено обращение к функции загрузки данных из БД Всемирного банка, а полученные данные переданы объектам df_pivot и val. Данные, полученные из модуля pand_dr_wb, уже имею формат сводной таблицы.

Затем объект df_pivot приводится к формату, который позволяет отобразить его на HTML странице (объект df_html). После этого сформирована контекстная переменная context, которая передана в шаблон wb_export_dr_pivot.html (листинг 3.23).

Листинг 3.23. Шаблон wb_export_dr_pivot.html

Примечание.
В демо-версии книги листинги программ удалены.

Это достаточно простой шаблон с контейнером, в котором выведен объект df_html. Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 3» и активировать ссылку «Экспорт стран (табл. pivot)».

Если выполнить набор вышеприведенных модулей, то получим следующий результат (рис. 3.5).

Динамика экспорта (по данным Всемирного банка)

country	China	Russian Federation	United Kingdom	United States
year				
2015	21.354080	28.703389	27.845951	12.411149
2016	19.584380	25.854483	28.902340	11.888159
2017	19.692277	26.090881	30.890351	12.177481
2018	19.112104	30.793257	31.580111	12.287111
2019	18.409992	28.433431	31.634780	11.789197
2020	18.586139	25.522186	29.669796	10.073656
2021	19.943974	29.771214	29.180275	10.790978
2022	20.791483	28.025719	33.567194	11.602228
2023	19.743072	23.083003	31.662677	11.011511

Рис. 3.5. Данные об экспорте стран, полученные из API Всемирного банка в виде сводной таблицы

В этой таблице 5 колонок. Первая колонка это даты (годы), отсортированные по возрастанию. В следующих 4-х колонках находятся данные экспорта для каждой страны. Сформированные таким образом данные сводной таблицы могут быть представлены в графическом виде, что более удобно для их визуального анализа.

3.5. Доступ к данным финансовой платформы Stooq с библиотекой pandas_datareader

С использованием модуля pandas_datareader.data можно получать данные из различных источников. Для этого в строке запроса к данным достаточно указать имя источника. Однако следует помнить, что ряд источников данные требуют указания ключа API, который может быть платным. Рассмотрим популярную финансовую платформу Stooq. Это источник финансовых данных и информации для инвесторов и трейдеров, с которого можно загрузить ежедневные данные о фондовом рынке и других финансовых рынках со всего мира. В отличие от других поставщиков данных, Stooq предоставляет исторические данные о ценах акций и объемах продаж бесплатно. Подключение к этому источнику данных не требует наличия ключа API. Финансовая платформа Stooq обеспечивает следующие возможности:

- ❑ бесплатный доступ к данным, которые охватывают более 21 000 глобальных ценных бумаг.
- ❑ доступ к данным на уровне минут, что важно для алгоритмических торговых стратегий, требующих точного и детального анализа данных.
- ❑ возможность загружать данные в формате CSV, что упрощает интеграцию с аналитическими инструментами.

3.5.1. Запроса к данным финансовой платформы Stooq из модуля Python

Напишем функцию на Python, которая обеспечит загрузку данных о стоимости акций из источника данных Stooq (листинг 3.24).

Листинг 3.24. Программный модуль pand_dr_data.py

Примечание.
В демо-версии книги листинги программ удалены.

В этом модуле выполнен импорт необходимых библиотек и создана функция def get_dr_data(), которая принимает следующие параметры:

- firm – перечень фирм в виде списка;
- s – источник данных в виде текстовой строки;
- start - дата начала периода (символьная строка в европейском формате);
- end - дата конца периода (символьная строка в европейском формате).

После этого даты начала и конца периода преобразуются из символьной строки в объект типа datetime (американский формат). Затем с использованием метода web.DataReader информация извлекается из финансовой платформы Stooq и сохраняется в объекте data, который возвращаются в точку вызова.

```
data = web.DataReader(firm, s, start=d1, end=d2).dropna()
```

Примечание.
В этом запросе к данным используется метод dropna(). Метод Pandas dropna() позволяет удалять строки (столбцы) с пустыми значениями. Использовать данный метод не обязательно, но он может помочь в тех случаях, когда источник неожиданно перестает возвращать данные. Такая ситуация может возникнуть либо при обновлении API, либо при изменении структуры возвращаемой информации.

В этом же программном модуле выполнено тестовое обращение к данной функции, в которую переданы следующие параметры:

- d1_str = '01-01-2024' – дата начала периода (строка, европейский формат);
- d2_str = '17-04-2025' - дата конца периода (строка, европейский формат);
- list_firm = ['MSFT'] – список фирм или тикеров (строка, значение тикера);
- sours = 'stooq' – источник данных (финансовая платформа Stooq).

Здесь тикеры, определяющие отношение актива к фирме или компании, представлены в виде списка со строковыми значениями.

Примечание.
Тикер в контексте финансовой платформы Stooq — это буквенное или числовое обозначение актива на бирже. Тикер заменяет полное название ценной бумаги и обычно состоит из нескольких символов. Его присваивают разным активам: акциям, фондам, фьючерсам. На платформе Stooq тикеры используются для поиска исторических данных о ценах и объемах торгов по различным активам. Например, тикер AAPL обозначает акции компании Apple, TSLA – акции компании Tesla.

Если данный модуль запустить на выполнение, то будет получен следующий результат.

```
Attributes   Close      High       Low      Open       Volume
```

```
Symbols       MSFT     MSFT     MSFT     MSFT        MSFT
Date
2025-04-17  367.780  374.3215  366.8900  373.745  2.112019e+07
2025-04-16  371.610  381.6100  368.0000  380.670  2.196783e+07
2025-04-15  385.730  391.8900  384.1599  388.505  1.719989e+07
2025-04-14  387.810  394.6500  384.2100  393.220  1.925117e+07
2025-04-11  388.450  390.0500  378.8900  380.635  2.383922e+07
...             ...      ...       ...       ...         ...
2024-01-08  371.909  372.4150  366.2720  366.558  2.330696e+07
2024-01-05  365.020  369.2980  363.7800  366.232  2.116164e+07
2024-01-04  365.209  370.3300  364.4450  367.915  2.105780e+07
2024-01-03  367.850  370.4850  365.7750  366.272  2.325608e+07
2024-01-02  368.116  373.1090  364.0490  371.085  2.544751e+07
[325 rows x 5 columns]
```

Примечание.
Представленный в данном разделе программный модуль может работать только на локальном компьютере в приложении на Python.

Эта таблица содержит колонку с датами и 5 информационных колонок:

- Close – стоимость акций на момент закрытия торгов;
- High – максимальная стоимость акций;
- Low – минимальная стоимость акций;
- Open - стоимость акций на момент открытия торгов;
- Volume – объем продаж.

3.5.2. Запроса к данным финансовой платформы Stooq из приложения на Django

Теперь рассмотрим пример, в котором получим данные о котировках акций из финансовой платформы Stooq в приложении на Django. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 3.25).

Листинг 3.25. Ссылка активации маршрута stock_price_tabl

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем stock_price_tabl (листинг 3.26).

Листинг 3.26. Маршрут URL с именем stock_price_tabl

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление stock_price_tabl (листинг 3.27).

Листинг 3.27. Представление (view) stock_price_tabl

Примечание.
В демо-версии книги листинги программ удалены.

В этом программном коде выполнен импорт модуля pand_dr_data, код которого описан в предыдущем разделе. Затем сформированы параметры запроса к БД:

- начальное значение интервала дат (объект d1_str);
- конечное значение интервала дат (объекты d2_str);
- список фирм (тикеров) – объект list_firm (фирма Microsoft);
- источник данных – объект sours (финансовая платформа stooq).

После этого выполнено обращение к функции get_dr_data модуля pand_dr_data, в которую переданы сформированные аргументы. Полученная из источника данных информация загружена в объект df.

Так как полученный набор данных содержит несколько сотен строк, то для отображения в шаблоне выбирается только первые и последние строки этого набора (объекты df_1 и df_2).

Затем объекты df_1 и df_2 приводится к формату, который позволяет отобразить их на HTML странице (объекты df1_html и df2_html). После этого сформирована контекстная переменная context, которая передана в шаблон stock_price_tabl.html (листинг 3.28).

Листинг 3.28. Шаблон stock_price_tabl.html

Примечание.
В демо-версии книги листинги программ удалены.

Это достаточно простой шаблон с контейнером, в котором выведены объекты df1_html и df2_html. Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 3» и активировать ссылку «Стоимость акций от Stooq (таблица)».

Если выполнить набор вышеприведенных модулей, то получим следующий результат (рис. 3.6).

Динамика изменения стоимости акция
по данным Stooq, ($)
За период с 01-01-2024 по 07-06-2025

Начало набора данных

Attributes	Close	High	Low	Open	Volume
Symbols	MSFT	MSFT	MSFT	MSFT	MSFT
Date					
2025-06-06	470.38	473.335	468.7800	470.085	15285624.0
2025-06-05	467.68	469.650	464.0300	464.955	20154460.0
2025-06-04	463.87	465.690	463.0201	464.000	14162688.0
2025-06-03	462.97	464.140	460.8622	461.470	15743760.0
2025-06-02	461.97	462.110	456.8900	457.140	16626495.0

Конец набора данных

Attributes	Close	High	Low	Open	Volume
Symbols	MSFT	MSFT	MSFT	MSFT	MSFT
Date					
2024-01-08	371.909	372.415	366.272	366.558	2.330696e+07
2024-01-05	365.020	369.298	363.780	366.232	2.116164e+07
2024-01-04	365.209	370.330	364.445	367.915	2.105780e+07
2024-01-03	367.850	370.485	365.775	366.272	2.325608e+07
2024-01-02	368.116	373.109	364.049	371.085	2.544751e+07

Рис. 3.6. Динамика стоимости акций от финансовой платформы Stooq

3.6. Доступ к данным финансовой платформы Yahoo Finance

Теперь рассмотрим еще один источник финансово-экономической информации – это платформа Yahoo Finance. На данной медиа платформе кроме актуальных финансовых новостей можно получить данные о котировках акций, курсах валют и о различной финансовой отчетности.

Платформа Yahoo Finance является одной из самых популярных площадок для загрузки данных. Одно из самых больших преимуществ Yahoo Finance заключается в том, что это бесплатная платформа. Вам не нужно платить никаких ежемесячных или ежегодных сборов за ее использование. Это делает ее отличным выбором для тех, кто только начинает заниматься онлайн-торговлей акциями и анализом экономической информации.

В качестве недостатка можно отметить, что иногда загрузка данных может выполняться медленно из-за наплыва большого количества пользователей. Сотрудники компании могут внести изменения в API, что приведет в невозможности загрузки данных. Из-за этого, например, могут перестать работать ряд модулей библиотеки pandas. Однако для работы с платформой Yahoo Finance есть специализированная библиотека, которую можно использовать как альтернативный вариант для загрузки необходимых данных. Установить данную библиотеку можно с использованием следующей инструкции:

pip install yfinance

3.6.1. Запроса к данным финансовой платформы Yahoo Finance из модуля Python

Напишем функцию на Python, которая обеспечит загрузку данных о стоимости акций из источника данных Yahoo Finance (листинг 3.29).

Листинг 3.29. Программный модуль yfin_data.py

Примечание.
В демо-версии книги листинги программ удалены.

В этом модуле выполнен импорт необходимых библиотек и создана функция с именем def get_data, которая принимает следующие параметры:

- firma – список кодов фирм или тикеров (тип данных – список);
- start – дата начала периода в европейском формате (тип данных – строка);
- end - дата конца периода в европейском формате (тип данных – строка).

Далее принятые строковые даты преобразуются в данные типа datetime (американский формат). С использованием метода yf.download делается запрос к API финансовой платформы Yahoo Finance. Полученные данные сохраняются в объекте data, который возвращается в точку вызова.

В этом же программном модуле выполнено тестовое обращение к данной функции, в которую переданы следующие параметры:

- d1_str = '01-01-2025' – дата начала периода (строка, европейский формат);
- d2_str = '17-04-2025' - дата конца периода (строка, европейский формат);
- list_firm = ['AAPL', 'GOOGL'] – список фирм или тикеров (строка, значения тикеров).

Здесь тикеры, определяющие отношение актива к фирме или компании, представлены в виде списка со строковыми значениями. В качестве примера при отладке работы данного модуля был выполнен запрос истории стоимости акций компаний Apple и Google. Если выполнить данный программный модуль, то будут получены следующие результаты.

```
Price          Close                 High  ...       Open    Volume
Ticker          AAPL      GOOGL       AAPL  ...      GOOGL      AAPL     GOOGL
Date                                        ...
2025-01-02  243.263199  189.212082  248.500565  ...  190.430680   55740700  20370800
2025-01-03  242.774368  191.569366  243.592387  ...  191.149851   40244100  18596200
2025-01-06  244.410416  196.643524  246.734810  ...  193.756849   45045600  29563600
2025-01-07  241.627136  195.265121  244.959095  ...  196.883253   40856000  26487200
2025-01-08  242.115952  193.726883  243.123531  ...  192.348481   37628900  24864800
...                ...         ...         ...  ...         ...        ...       ...
2025-04-10  190.170624  152.820007  194.524915  ...  156.539993  121880000  48022000
2025-04-11  197.890488  157.139999  199.278667  ...  152.899994   87435900  33636200
2025-04-14  202.254776  159.070007  212.661128  ...  160.000000  101352900  30333000
2025-04-15  201.875275  156.309998  203.243476  ...  159.130005   51343900  27551500
2025-04-16  194.015579  153.330002  200.437151  ...  153.100006   59732400  28187400
[72 rows x 10 columns]
```

Примечание.
Представленный в данном разделе программный модуль может работать только на локальном компьютере в приложении на Python.

Здесь первой колонкой является дата, а далее следует 10 колонок с данными по объемам продаж и стоимости акций компаний с кодом AAPL и GOOGL:

- Close – стоимость акций на момент закрытия торгов;
- High – максимальная стоимость акций;
- Low – минимальная стоимость акций;
- Open - стоимость акций на момент открытия торгов;
- Volume – объем продаж.

3.6.2. Запроса к данным финансовой платформы Yahoo Finance из приложения Django

Теперь рассмотрим пример, в котором получим данные о котировках акций из финансовой платформы Yahoo Finance в приложении на Django. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 3.30).

Листинг 3.30. Ссылка активации маршрута yahoo_fin_tabl

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем yahoo_fin_tabl (листинг 3.31).

Листинг 3.31. Маршрут URL с именем yahoo_fin_tabl

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление yahoo_fin_tabl (листинг 3.32).

Листинг 3.32. Представление (view) yahoo_fin_tabl

Примечание.
В демо-версии книги листинги программ удалены.

В этом представлении выполнен импорт модуля загрузки данных (yfin_data), код которого описан в предыдущем разделе. Заданы даты начала и конца периода (d1_str, d2_str). Задан список с кодами (тикерами) фирм (list_firm). Затем выполнено обращение к функции

загрузки данных из финансовой платформы Yahoo Finance, а полученные данные переданы объекту df.

После этого объекты df_1 и df_2 приводится к формату, который позволяет отобразить их на HTML странице (объекты df1_html и df2_html). После этого сформирована контекстная переменная context, которая передана в шаблон yahoo_fin_tabl.html (листинг 3.33).

Листинг 3.33. Шаблон yahoo_fin_tabl.html

Примечание.
В демо-версии книги листинги программ удалены.

Это достаточно простой шаблон с контейнером, в котором выведены объекты df1_html и df2_html. Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 3» и активировать ссылку «Стоимость акций от YahooFin (таблица)».

Если выполнить набор вышеприведенных модулей, то получим следующий результат (рис. 3.7).

Динамика изменения стоимости акция

по данным Yahoo Finance, ($)

За период с 01-01-2024 по 17-04-2025

Начало набора данных

Price	Close		High		Low		Open		Volume	
Ticker	AAPL	GOOGL	AAPL	GOOGL	AAPL	GOOGL	AAPL	GOOGL	AAPL	GOOGL
Date										
2024-01-02	184.290436	137.512558	187.070083	138.786466	182.553158	135.830597	185.789453	137.890755	82488700	23711200
2024-01-03	182.910522	138.259003	184.528677	138.965631	182.096477	136.427761	182.880742	136.596951	58414500	24212100
2024-01-04	180.587540	135.741028	181.758954	138.497852	179.565029	135.701225	180.825785	137.761367	71983600	27137700
2024-01-05	179.862854	135.084183	181.431370	136.507387	178.860202	134.506941	180.666978	136.099334	62303300	22506000
2024-01-08	184.210999	138.179382	184.250716	138.348572	180.180517	135.611656	180.766224	135.641512	59144500	21404000

Конец набора данных

Price	Close		High		Low		Open		Volume	
Ticker	AAPL	GOOGL	AAPL	GOOGL	AAPL	GOOGL	AAPL	GOOGL	AAPL	GOOGL
Date										
2025-04-10	190.170624	152.820007	194.524915	157.720001	182.760343	149.929993	188.822401	156.539993	121880000	48022000
2025-04-11	197.890488	157.139999	199.278667	157.669998	185.816325	152.820007	185.856281	152.899994	87435900	33636200
2025-04-14	202.254776	159.070007	212.661128	161.720001	200.896557	157.559998	211.163092	160.000000	101352900	30333000
2025-04-15	201.875275	156.309998	203.243476	159.649994	199.538343	155.210007	201.595643	159.130005	51343900	27551500
2025-04-16	194.015579	153.330002	200.437151	155.889999	192.118058	151.509995	198.100219	153.100006	59732400	28187400

Рис. 3.7. Динамика стоимости акций от финансовой платформы Yahoo Finance

Это достаточно длинная таблица и ее анализ в таком виде сильно затруднен. Однако эти данные после некоторой предварительной обработки могут быть представлены в более понятном графическом виде.

3.7. Краткие итоги

Итак, мы познакомились с некоторыми возможностями доступа к открытым источникам данных через API. Конечно, это далеко не полный перечень API, который позволяет выполнить удаленную загрузку данных. Однако, изучив базовые принципы работы с API на примерах этой главы, вы без труда сможете использовать удаленную загрузку данных из других источников. В последующих главах будет еще много примеров использования API, так что будет возможность закрепить полученные знания.

Итак, мы научились получать данные, преобразовывать их в наборы DataFrame и выводить в шаблоны Django в табличном виде. Однако массив данных, состоящий и десятка тысяч строк, мало о чем может сказать до тех пор, пока он не будет представлен в графическом виде. Язык программирования Python, благодаря специализированным библиотекам, обеспечивает широкие возможность визуализации данных любой сложности. Изучению этих библиотек и посвящена следующая глава.

Глава 4. Библиотеки Python для визуализации данных

Визуализация данных является важнейшим инструментом в Data Science, поскольку помогает более наглядно показать структуру и динамику изменения больших массивов данных. Прежде чем приступить к знакомству с графическими библиотеками, разберемся с основными видами диаграмм, которые используются при анализе данных.

Статические диаграммы. Это статичные графики, которые представляют собой обычные изображения.

Интерактивные диаграммы. Интерактивные диаграммы это «живые» графики, которые откликаются на действия пользователя. Такими графиками можно управлять с использованием мыши, встроенными кнопками или слайдерами. Интерактивность обеспечивают встроенные модули на JavaScript, поэтому такие графики отображаются только в браузере.

Статичные диаграммы могут быть созданы с использованием таких популярных библиотек, как Matplotlib и Seaborn. Эти библиотеки обычно используют в научных исследованиях, так как они позволяют строить графики с минимально коротким программным кодом.

Интерактивные диаграммы, которые реагируют на действия пользователя, можно реализовать с библиотекой Plotly. Эту библиотеку часто используют для графического представления финансовой и экономической информации в веб приложениях. Для создания аналитических веб панелей используют фреймворк Dash. Этот инструмент позволяет создавать адаптивные веб-панели, которые имеют привлекательный внешний вид и работают очень быстро, при этом нет необходимости разбираться в сложных структурах HTML, CSS и JavaScript. Все эти библиотеки имеют тесную интеграцию с библиотекой Pandas, что является хорошим поводом использования этого инструментария для анализа данных и прогнозирования.

Популярный инструмент анализа и обработки данных Pandas предоставляет встроенные функции для построения графиков на основе своих наборов данных (DataFrame и Series). Исторически Pandas была интегрирована с библиотекой matplotlib для построения статичных графиков. А начиная с версии 4.8 можно использовать в фоновом режиме (как бэкэнд) библиотеку Plotly Express для построения интерактивных графиков. Это означает, что теперь можно создавать интерактивные графики непосредственно из набора данных, даже не импортируя Plotly. Однако, если нужна тонкая настройка сложных графиков, то библиотеку Plotly можно использовать с Pandas в режиме прямого импорта. Поскольку мы будем реализовывать веб приложения анализа данных, то нам потребуется интеграция указанного выше инструментария с фреймворком Django.

Из материалов этой главы вы узнаете:

- как установить библиотеку Plotly;
- как можно строить различные графики с интерфейсом Plotly Express;
- как можно визуализировать данные с библиотекой Pandas;
- как можно визуализировать данные с библиотекой Matplotlib.

В материале этой главы мы не будем рассматривать все виды диаграмм, которые можно построить с использованием приведенных выше библиотек. Будут рассмотрены только базовые диаграммы, которые используются для визуализации данных временных рядов. Главный упор здесь сделан на том, как можно графики преобразовать к формату, в котором они могут быть отображены в шаблонах Django.

4.1. Визуализация данных с библиотекой Plotly

Plotly — это библиотека Python, которая помогает создавать интерактивные и визуально привлекательные диаграммы и графики. Она позволяет отображать данные таким образом, чтобы их было легко изучать и понимать, например, с помощью масштабирования, наведения курсора на точки данных для получения более подробной информации. Plotly использует JavaScript для обеспечения интерактивности, но вам не нужно беспокоиться об этом при использовании данной библиотеки. Вы просто пишете код на Python для создания диаграмм, а Plotly позаботится об их интерактивности.

Библиотека Plotly не входит в стандартную комплектацию Python. Для установки данной библиотеки используйте следующую команду.

```
pip install plotly
```

Библиотека Plotly имеет модульную архитектуру, которая обеспечивает гибкость при создании визуализаций. В основе архитектуры Plotly лежат два основных интерфейса: **Plotly Express** и **Plotly Graph Objects**. Каждый из них служит определенной цели и удовлетворяет различные потребности пользователей, предлагая разные уровни сложности и настройки.

4.2. Визуализация данных с интерфейсом Plotly Express

Plotly Express — это высокоуровневый, удобный интерфейс для создания графиков. Он предназначен для упрощения процесса создания распространенных типов визуализаций с минимальным количеством кода. Plotly Express создан на основе Plotly Graph Objects и предоставляет более удобный и интуитивно понятный интерфейс для пользователей, с которым можно быстро создавать интерактивные графики, не вникая в тонкости низкоуровневых функций Plotly.

Интерфейс Plotly Express имеет следующие особенности.

13. **Простота и удобство использования**. Plotly Express позволяет пользователям создавать графики с помощью лаконичного и понятного синтаксиса. Эта простота достигается за счет абстрагирования от многих сложных функций Plotly, что делает его доступным даже для новичков в области визуализации данных.
14. **Быстрая визуализация**. Plotly Express идеально подходит для исследовательского анализа данных и быстрой визуализации. Пользователи могут создавать широкий спектр распространенных типов графиков, таких как точечные, гистограммы и линейные графики, используя всего несколько строк кода.
15. **Автоматическая обработка**. Plotly Express автоматически управляет многими аспектами графика, такими как метки осей, заголовки и легенды. Такая автоматизация

помогает пользователям сосредоточиться на анализе данных, а не на технических деталях конфигурации графика.

4.2.1. Графики линий с интерфейсом Plotly Express

Реализуем простейший пример формирования графика с использованием интерфейса Plotly Express. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 4.1).

Листинг 4.1. Ссылка активации маршрута fig_plot_exp

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем fig_plot_exp (листинг 4.2).

Листинг 4.2. Маршрут URL с именем fig_plot_exp

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление fig_plot_exp (листинг 4.3).

Листинг 4.3. Представление (view) fig_plot_exp

Примечание.
В демо-версии книги листинги программ удалены.

В этом модуле выполнен импорт необходимых библиотек, создана простейший набор данных, имитирующих прямую линию (объект df), и сформирован график с использованием упрощенного интерфейса Plotly Express (объект fig). Для построения графика был использован метод px.line, которому переданы следующие параметры:

- df – исходный набор данных DataFrame;
- x='data_x' – данные набора данных df, значения которой буду располагаться на оси «x»;
- y='data_y - данные набора данных df, значения которой буду располагаться на оси «y»;
- title – заголовок графика.

После этого график приведен к формату, в котором оно может быть показано на HTML странице (объект plot_div) и создана строка с заголовком страницы (title_str). Полученный график (plot_div) с заголовком страницы через контекстную переменную context передается в шаблон fig_plotly.html (листинг 4.4).

Листинг 4.4. Шаблон fig_ploly.html

Примечание.
В демо-версии книги листинги программ удалены.

Это достаточно простой шаблон с контейнером, в котором выведен объекты plot_div (график). Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 4» и активировать ссылку «График Plotly Express».

Если выполнить набор вышеприведенных модулей, то получим следующий результат (рис. 4.1).

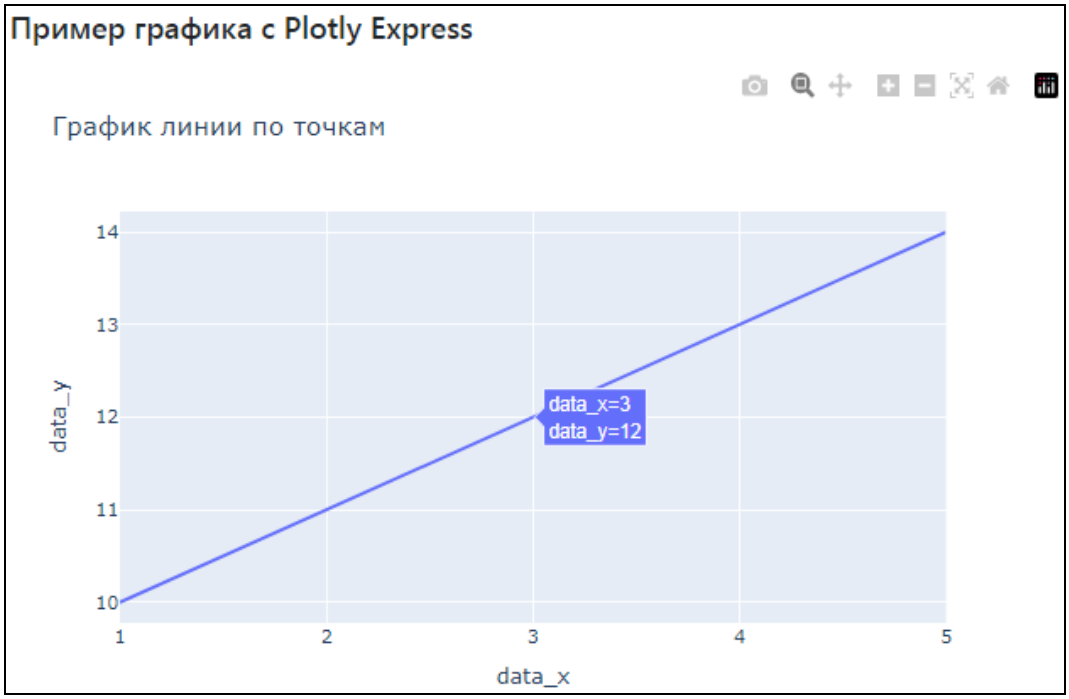


Рис. 4.1. Пример вывода графика, созданного в Plotly Express, в шаблон Django

Полученный график является интерактивным. При наведении курсора мыши на линию появляется окно с координатами точек, в правом верхнем углу имеется кнопочное меню для изменения параметров графика. Так как график выводится в адаптивный контейнер Bootstrap, то он будет менять свои размеры при изменении размеров окна браузера. Здесь не были заданы подписи к осям «x» и «y», поэтому по умолчанию они взяты из имен колонок набора данных df.

4.2.2. График линий в Plotly Express с группировкой столбцов набора данных

Достаточно часто возникают ситуации, когда в столбце набора данных находится информация, имеющая разные признаки. При традиционном подходе на графике весь столбец будет показан одной линией без выделения классификационного признака. Однако в Plotly Express есть возможность данные одного столбца с разными характеристиками отобразить линиями разных цветов и разных типов. Рассмотрим такую возможность на примере набора данных о параметрах цветков Ирис (он входит в стандартный пакет библиотеки Plotly_Express). Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 4.5).

Листинг 4.5. Ссылка активации маршрута fig_plot_exp_group

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем fig_plot_exp_group (листинг 4.6).

Листинг 4.6. Маршрут URL с именем fig_plot_exp_group

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление fig_plot_exp_group (листинг 4.7).

Листинг 4.7. Представление (view) fig_plot_exp_group

Примечание.
В демо-версии книги листинги программ удалены.

В этом примере выполнен импорт библиотеки Plotly Express, из данной библиотеки в объект df загружен набор данных с параметрами разных сортов цветка Ирис, и содержание набора данных df выведено в окно терминала:

```
   sepal_length  sepal_width    ...    species    species_id
0           5.1           3.5    ...     setosa           1
1           4.9           3.0    ...     setosa           1
2           4.7           3.2    ...     setosa           1
3           4.6           3.1    ...     setosa           1
4           5.0           3.6    ...     setosa           1
..          ...           ...    ...        ...         ...
145         6.7           3.0    ...  virginica           3
146         6.3           2.5    ...  virginica           3
147         6.5           3.0    ...  virginica           3
148         6.2           3.4    ...  virginica           3
149         5.9           3.0    ...  virginica           3
[150 rows x 6 columns]
```

В этом наборе данных имеются следующие колонки:

- sepal_length - длина чашелистика;
- sepal_width - ширина чашелистика;
- species – название сорта цветка Ирис (setosa, versicolor и virginica);
- species_id - код сорта цветка Ирис (1, 2, 3).

В следующей инструкции сформирован график (объект fig_1) значений ширины чашелистика (sepal_width) для разных сортов цветка Ирис (по50 значений для каждого сорта).

```
fig_1 = px.line(df, y="sepal_width")
```

Здесь в качестве значений для оси «x» использована индексная колонка набора данных df, а в качестве значений для оси «y» использована колонка набора данных «sepal_width».

В следующей инструкции сформирован тот же график (объект fig_2), но с группировкой данных по колонке species (сорт цветка).

```
fig_2 = px.line(df, y="sepal_width", line_group='species')
```

В следующей инструкции сформирован и выведен тот же график (объект fig_3) с группировкой данных по колонке species (сорт цветка), и с выделением сорта цветка линиями разного цвета и разного типа.

```
fig_3 = px.line(df, y="sepal_width", line_dash='species', color='species')
```

Здесь тип линии задан в параметре line_dash, а цвет линии в параметре color.

После этого все три графика приведены к формату, в котором они могут быть показаны на HTML странице (объекты plot_div1, plot_div2, plot_div3), и созданы строки с заголовками графика (title_str1, title_str2, title_str3). Полученные графики и их заголовки через контекстную переменную context передается в шаблон fig_plot_exp_group.html (листинг 4.8).

Листинг 4.8. Шаблон fig_plot_exp_group.html

Примечание.
В демо-версии книги листинги программ удалены.

Это шаблон с контейнером, в котором имеются три блока для вывода заголовков графиков и самих графиков. Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 4» и активировать ссылку «График Plotly Express (группировка)».

Если выполнить набор вышеприведенных модулей, то получим три графика. Первый график, на котором представлены данные о ширине чашелистика без группировки по сортам, представлен на рис. 4.2.

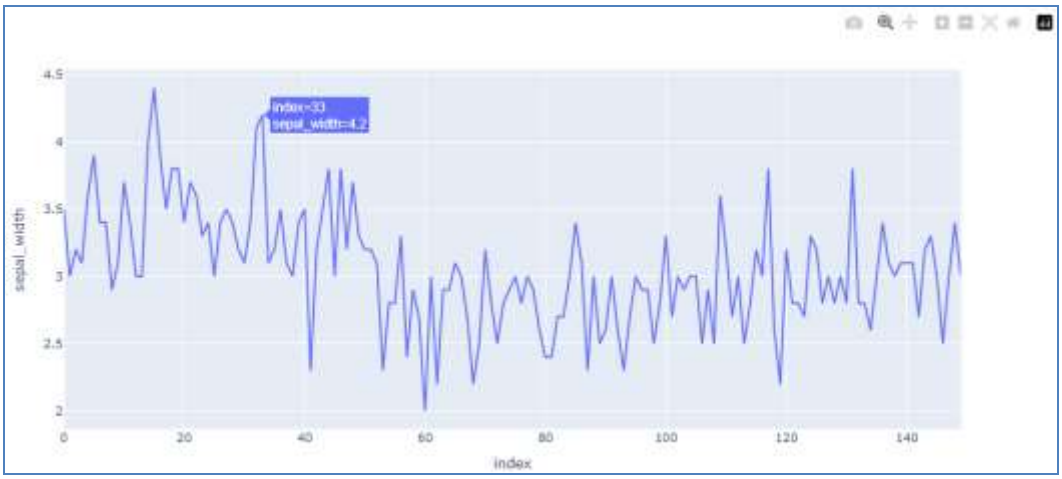


Рис.4.2. Данные ширины чашелистика без группировки сортов

Как видно из данного рисунка, на графике визуально трудно определить, какая часть линии относится к тому или иному сорту цветка.

Второй график, на котором представлены данные о ширине чашелистика с группировкой по сортам и линией одного цвета, представлен на рис. 4.3.

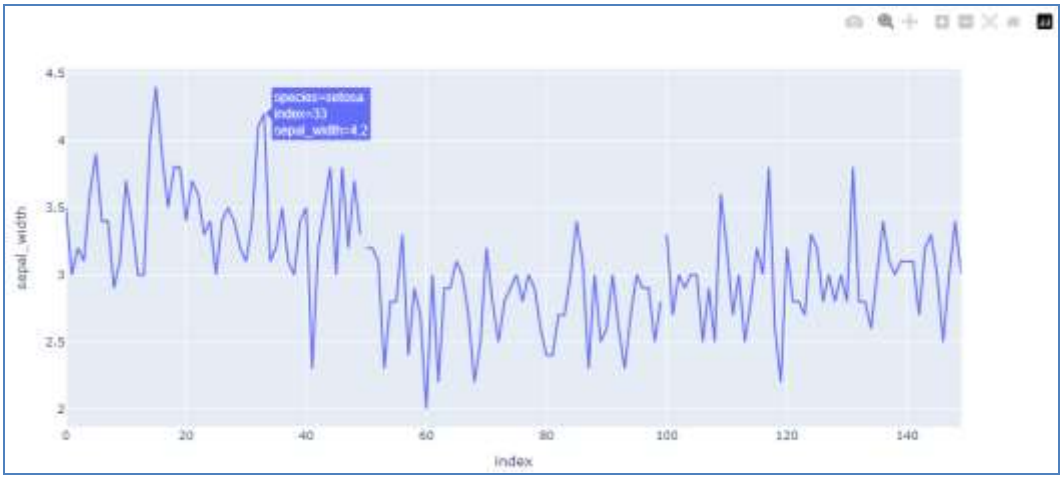


Рис.4.3. Данные ширины чашелистика с группировкой по сортам (линия одного цвета)

Как видно из данного рисунка, линии, относящие к разным сортам цветка, имеют разрывы. Это не очень наглядно, но, тем не менее, обеспечивает возможность определить, какая часть линии относится к тому или иному сорту цветка.

Третий график, на котором представлены данные о ширине чашелистика с группировкой по сортам и линиями разного цвета и типа, представлен на рис. 4.4.

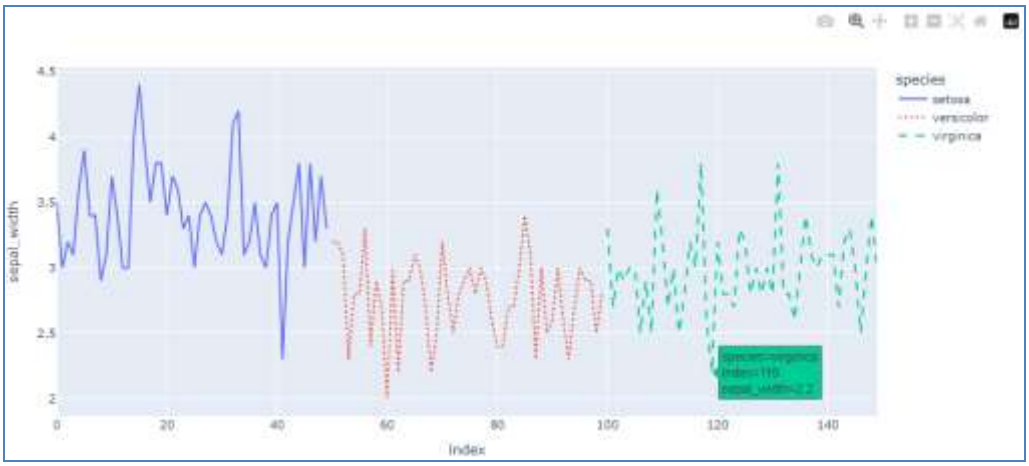


Рис.4.4. Данные ширины чашелистика с группировкой по сортам (линии разного цвета и типа)

Как видно из данного рисунка, линии, относящие к разным сортам цветка, имеют разный тип и разный цвет. Это обеспечивает возможность достаточно наглядно определить, какая часть графика относится к тому или иному сорту цветка. Кроме того, в правой части графика появилась легенда, которая поясняет, какая линия описывает параметр соответствующего сорта цветка.

4.2.3. Столбчатая диаграмма в Plotly Express

Реализуем простейший пример формирования столбчатой диаграммы с использованием интерфейса Plotly Express. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 4.9).

Листинг 4.9. Ссылка активации маршрута fig_plot_bar

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем fig_plot_bar (листинг 4.10).

Листинг 4.10. Маршрут URL с именем fig_plot_bar

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление fig_plot_bar (листинг 4.11).

Листинг 4.11. Представление (view) fig_plot_bar

Примечание.
В демо-версии книги листинги программ удалены.

В этом примере выполнен импорт библиотеки Plotly Express и создано два списка с условными данными:

- x – годы;
- y – население.

В следующей инструкции с использованием метода bar создана столбчатая диаграмма:

```
fig = px.bar(x=x, y=y,
             title='Динамика изменения населения',
             labels={'y': 'Население, тыс.чел.', 'x': 'Годы'})
```

После этого график приведен к формату, в котором оно может быть показано на HTML странице (объект plot_div) и создана строка с заголовком страницы (title_str). Полученный график (plot_div) с заголовком страницы через контекстную переменную context передается в шаблон fig_plotly.html (листинг 4.12).

Листинг 4.12. Шаблон fig_ploly.html

Примечание.
В демо-версии книги листинги программ удалены.

Это достаточно простой шаблон с контейнером, в котором выведен объекты plot_div (график). Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 4» и активировать ссылку «Столбчатая диаграмма в Plotly Express».

Если выполнить набор вышеприведенных модулей, то получим следующий результат (рис. 4.5).



Рис. 4.5. Пример вывода столбчатой диаграммы, созданной в Plotly Express, в шаблон Django

4.3.Фасетные сетки для отображения группы графиков в Plotly Express

Сетка фасетов — это набор графиков, имеющих одни и те же оси («x», «y»), которые отображают один и тот же тип графика (или точки, или лини, или столбчатые диаграммы и т.д.), но для разных подмножеств данных некого набора (df). Это особенно полезно, когда необходимо визуально показать связь (или различия) между несколькими объектами в наборе данных.

В Plotly Express можно создать сетку фасетов, используя параметр facet_col (поместить графики в разных колонках) или facet_row (поместить графики в разных строках). В этом разделе в качестве исходного набора данных будут использоваться данные о параметрах разных сортов цветка Iris Данный набор данных входит в стандартный пакет библиотеки Plotly_Express.

4.3.1.Фасетные сетки для отображения группы графиков методом facet_col

Давайте создадим сетку фасетов на примере набора данных о параметрах разных сортов цветка Ирис с использованием метода facet_col. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 4.13).

Листинг 4.13. Ссылка активации маршрута fig_px_facet_col

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем fig_px_facet_col (листинг 4.14).

Листинг 4.14. Маршрут URL с именем fig_px_facet_col

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление fig_px_facet_col (листинг 4.15).

Листинг 4.15. Представление (view) fig_px_facet_col

Примечание.
В демо-версии книги листинги программ удалены.

В этом примере выполнен импорт библиотеки Plotly Express, из данной библиотеки в объект df загружен набор данных с параметрами разных сортов цветка Ирис.

В следующем фрагменте программного кода сформирован график рассеивания размеров чашелистика (длины - sepal_length и ширины - sepal_width) для разных сортов цветка Ирис с использованием следующей инструкции:

```
fig = px.scatter(df,
                 x='sepal_width',
                 y='sepal_length',
                 color='species',
                 facet_col='species',
                 title='Соотношение ширины и длины чашелистиков для видов цветка Ирис',
                 )
```

В этой инструкции в методе px.scatter заданы следующие параметры:

- df – исходный набор данных;
- x='sepal_width' – значение колонки (ширина чашелистика);
- y='sepal_length' – значение колонки (длина чашелистика);
- color='species' - значение цвета графика будет определено колонкой species (сорт цветка);
- facet_col='species' – создать сетку графиков на основе значений колонки species (сорт цветка);
- title – заголовок для всей сетки графиков.

С использованием этой инструкции будет создано три графика (по числу сортов цветка) с их размещением в одну линию (три колонки).

В следующих инструкциях изменены подписи к осям «x» сетки графиков:

```
fig.update_xaxes(title_text="Ширина", row=1, col=1)
fig.update_xaxes(title_text="Ширина", row=1, col=2)
fig.update_xaxes(title_text="Ширина", row=1, col=3)
```

В этом примере графики будут расположены в одну линию (row=1) и в трех колонках (col=1, col=2, col=3), в этих же координатах будут присутствовать подписи к оси x.

На следующем шаге выполнено обновление макета сетки графиков. В макете сетки графиков задана легенда и подпись к оси y:

```
fig.update_layout(legend_title="Виды Ириса", yaxis_title='Длина')
```

После этого график приведен к формату, в котором он может быть показан на HTML странице (объект plot_div), и создана строка с заголовком графика (title_str). Полученный блок графиков с заголовками через контекстную переменную context передается в шаблон fig_px_facet.html (листинг 4.16).

Листинг 4.16. Шаблон fig_px_facet.html

Примечание.
В демо-версии книги листинги программ удалены.

Это шаблон с контейнером, в котором имеется блок для вывода заголовка графика и самого графика. Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 4» и активировать ссылку «График Plotly Express (facet_col)».

Если выполнить набор вышеприведенных модулей, то получим блок из трех графиков (рис. 4.6).



Рис.4.6. Данные рассеивания размеров чашелистика цветка Ирис в фасетной сетке графиков (facet_col)

4.3.2.Фасетные сетки для отображения группы графиков методом facet_row

Давайте создадим сетку фасетов на примере набора данных о параметрах разных сортов цветка Ирис с использованием метода facet_row. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 4.17).

Листинг 4.17. Ссылка активации маршрута fig_px_facet_row

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем fig_px_facet_row (листинг 4.18).

Листинг 4.18. Маршрут URL с именем fig_px_facet_row

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление fig_px_facet_row (листинг 4.19).

Листинг 4.19. Представление (view) fig_px_facet_row

Примечание.
В демо-версии книги листинги программ удалены.

В этом примере выполнен импорт библиотеки Plotly Express, из данной библиотеки в объект df загружен набор данных с параметрами разных сортов цветка Ирис.

В следующем фрагменте программного кода сформирован график рассеивания размеров чашелистика (длины - sepal_length и ширины - sepal_width) для разных сортов цветка Ирис с использованием следующей инструкции:

```
# Графики с использованием сетки facet_row
fig = px.scatter(df,
                 x='sepal_width',
                 y='sepal_length',
                 color='species',
                 facet_row='species',
                 height=600,
                 title='Соотношение ширины и длины чашелистиков для видов цветка Ирис'
                 )
```

В этой инструкции в методе px.scatter заданы следующие параметры:

- df – исходный набор данныхd

- x='sepal_width' – значение колонки (ширина чашелистика);
- y='sepal_length' – значение колонки (длина чашелистика);
- color='species' - значение цвета графика будет определено колонкой species (сорт цветка);
- height=600 –размер графика по вертикали;
- facet_col='species' – создать сетку графиков на основе значений колонки species (сорт цветка);
- title – заголовок для всей сетки графиков.

С использованием этой инструкции будет создано три графика (по числу сортов цветка) с их размещением в один столбец (три строки).

В следующих инструкциях изменены подписи к осям «x» сетки графиков:

```
fig.update_yaxes(title_text="Ширина", row=1, col=1)
fig.update_yaxes(title_text="Ширина", row=2, col=1)
fig.update_yaxes(title_text="Ширина", row=3, col=1)
```

В этом примере графики будут расположены в трех строках (row=1, row=2, row=3) и одну колонку (col=1), и в этих же координатах будут присутствовать подписи к оси y.

На следующем шаге выполнено обновление макета сетки графиков. В макете сетки графиков задана легенда и подпись к оси x:

```
fig.update_layout(legend_title="Виды Ириса", xaxis_title='Длина')
```

После этого график приведен к формату, в котором он может быть показан на HTML странице (объект plot_div), и создана строка с заголовком графика (title_str). Полученный блок графиков с заголовками через контекстную переменную context передается в шаблон fig_px_facet.html (листинг 4.20).

Листинг 4.20. Шаблон fig_px_facet.html

> ***Примечание.***
> В демо-версии книги листинги программ удалены.

Это шаблон с контейнером, в котором имеется блок для вывода заголовка графика и самого графика. Теперь можно проверить работу данных модулей.

> ***Примечание.***
> В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 4» и активировать ссылку «График Plotly Express (facet_row)».

Если выполнить набор вышеприведенных модулей, то получим три график, представленный на рис. 4.7.

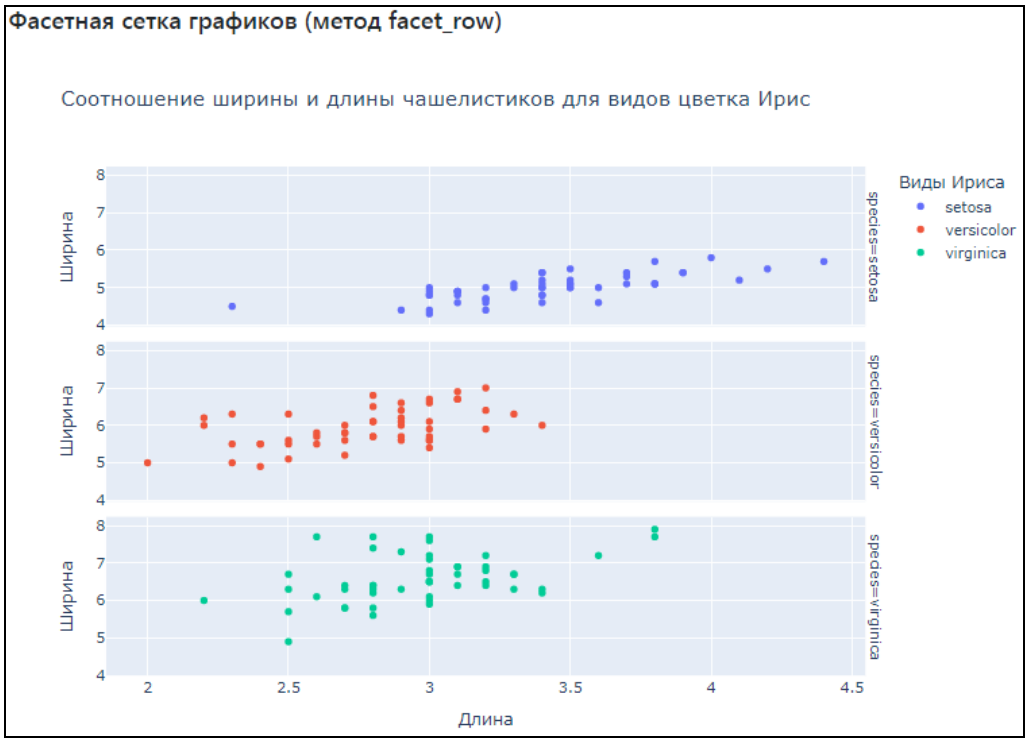


Рис.4.7. Данные рассеивания размеров чашелистика цветка Ирис в фасетной сетке графиков (facet_row)

4.4. Визуализация данных с интерфейсом Plotly Graph Objects

Plotly Graph Objects (часто сокращаемый до Plotly GO) — это низкоуровневый интерфейс, который обеспечивает более детальный контроль над созданием и настройкой графиков. В отличие от Plotly Express, который абстрагирует многие детали, Plotly Graph Objects позволяет пользователям создавать графики, вручную задавая каждый компонент. Этот подход подходит для пользователей, которым нужен детальный контроль над визуализацией или которые работают с более сложными требованиями к построению графиков.

Интерфейс Plotly Graph Objects имеет следующие особенности.

16. **Подробная настройка**. Plotly Graph Objects предлагает широкие возможности настройки, позволяющие пользователям управлять практически всеми аспектами графика. Это включает в себя настройку свойств отдельных линий, параметров макета и добавление аннотаций.
17. **Сложные графики**. Пользователи могут создавать замысловатые и сложные визуализации, комбинируя различные типы графиков и слоев. Эта возможность необходима для создания сложных графиков, требующих нескольких рядов данных или подробной настройки.

18. **Ручная настройка**. Для объектов Plotly Graph пользователи должны вручную определять структуру графика, включая линии, макет и настройки конфигурации. Такой подход обеспечивает гибкость, но требует более глубокого понимания синтаксиса и параметров Plotly.

4.4.1. График линии с интерфейсом Plotly Graph Objects

Реализуем простейший пример формирования графика с использованием интерфейса Plotly Graph Objects. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 4.21).

Листинг 4.21. Ссылка активации маршрута fig_plot_go

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем fig_plot_go (листинг 4.22).

Листинг 4.22. Маршрут URL с именем fig_plot_go

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление fig_plot_go (листинг 4.23).

Листинг 4.23. Представление (view) fig_plot_go

Примечание.
В демо-версии книги листинги программ удалены.

В этом модуле выполнен импорт необходимых библиотек, создана простейший набор данных, имитирующих прямую линию (объект df), и сформирован график с использованием упрощенного интерфейса Plotly Graph Objects. Сначала был создан объект fig, это своеобразная заготовка графика, или пустой макет. Затем с использованием метода fig.add_trace в макет графика добавлена линия. Данному методу переданы следующие параметры:

- x=df['data_x'] – имя колонки исходного набора данных, значения которой буду располагаться на оси «x»;
- y=df['data_y'] имя колонки исходного набора данных, значения которой буду располагаться на оси «y».

После этого было выполнено обновление макета графика – добавлен заголовок графика и подписи к осям «x» и «y».

После этого график приведен к формату, в котором он может быть показано на HTML странице (объект plot_div) и создана строка с заголовком страницы (title_str). Полученный

график (plot_div) с заголовком страницы через контекстную переменную context передается в шаблон fig_plotly.html (листинг 4.24).

Листинг 4.24. Шаблон fig_plotly.html

Примечание.
В демо-версии книги листинги программ удалены.

Это достаточно простой шаблон с контейнером, в котором выведен объекты plot_div (график). Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 4» и активировать ссылку «График линиии в Plotly GO».

Если выполнить набор вышеприведенных модулей, то получим следующий результат (рис. 4.8).

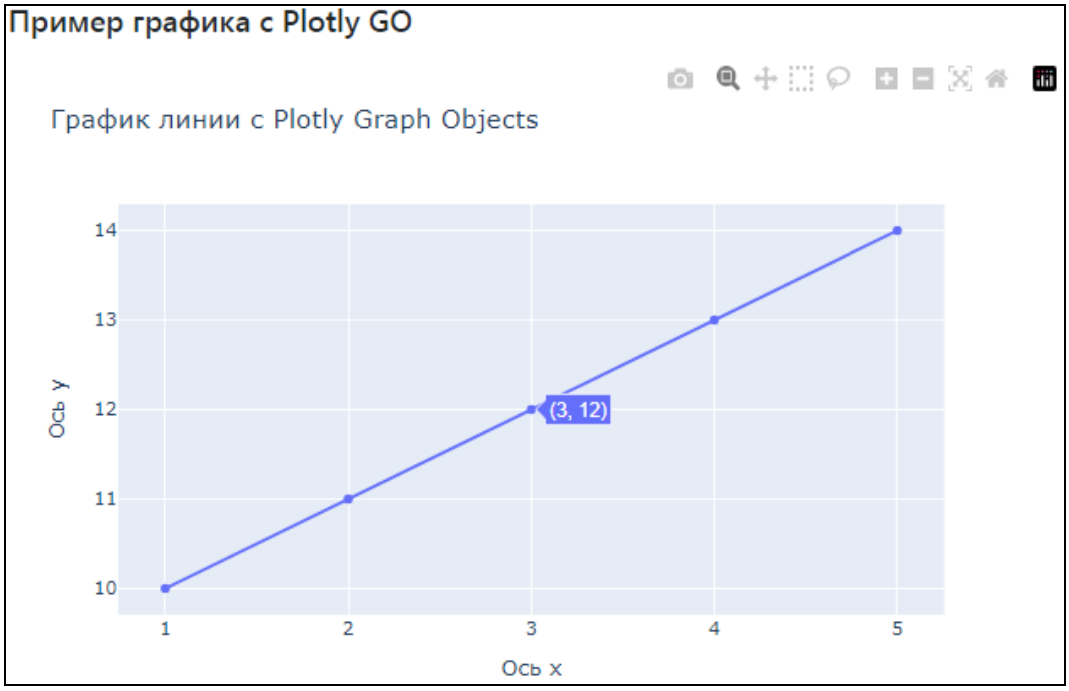


Рис. 4.8. Пример вывода графика, созданного в Plotly GO, в шаблон Django

Полученный график является интерактивным. При наведении курсора мыши на линию появляется окно с координатами точек, в правом верхнем углу имеется кнопочное меню для изменения параметров графика. Так как график выводится в адаптивный контейнер Bootstrap, то он будем менять свои размеры при изменении размеров окна браузера.

4.4.2. График нескольких линии с интерфейсом Plotly Graph Objects

Реализуем простейший пример формирования графика с использованием интерфейса Plotly Graph Objects, на котором будет выведено две линии. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 4.25).

Листинг 4.25. Ссылка активации маршрута fig_plot_go2

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем fig_plot_go2 (листинг 4.26).

Листинг 4.26. Маршрут URL с именем fig_plot_go2

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление fig_plot_go2 (листинг 4.27).

Листинг 4.27. Представление (view) fig_plot_go2

Примечание.
В демо-версии книги листинги программ удалены.

В этом модуле выполнен импорт необходимых библиотек, создано два простейших набора данных, имитирующих прямую линию (объекты df1, df2), и сформирован график с использованием упрощенного интерфейса Plotly Graph Objects. Сначала был создан объект fig, это своеобразная заготовка графика, или пустой макет. Затем с использованием метода fig.add_trace в макет графика добавлено две линии:

```
fig.add_trace(go.Scatter(x=df1['data_x'], y=df1['data_y'], name='Линия 1'))
fig.add_trace(go.Scatter(x=df2['data_x'], y=df2['data_y'], name='Линия 2'))
```

Данному методу переданы следующие параметры:

- x=df1['data_x'] – имя колонки исходного набора данных df1, значения которой буду располагаться на оси «x»;
- y=df1['data_y'] имя колонки исходного набора данных df1, значения которой буду располагаться на оси «y».
- x=df2['data_x'] – имя колонки исходного набора данных df2, значения которой буду располагаться на оси «x»;

- y=df2['data_y'] имя колонки исходного набора данных df2, значения которой буду располагаться на оси «y».

После этого было выполнено обновление макета графика – добавлен заголовок графика и подписи к осям «x» и «y».

После этого график приведен к формату, в котором он может быть показано на HTML странице (объект plot_div) и создана строка с заголовком страницы (title_str). Полученный график (plot_div) с заголовком страницы через контекстную переменную context передается в шаблон fig_plotly.html (листинг 4.28).

Листинг 4.28. Шаблон fig_plotly.html

Примечание.
В демо-версии книги листинги программ удалены.

Это достаточно простой шаблон с контейнером, в котором выведен объекты plot_div (график). Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 4» и активировать ссылку «График двух линий с Plotly GO».

Если выполнить набор вышеприведенных модулей, то получим следующий результат (рис. 4.9).

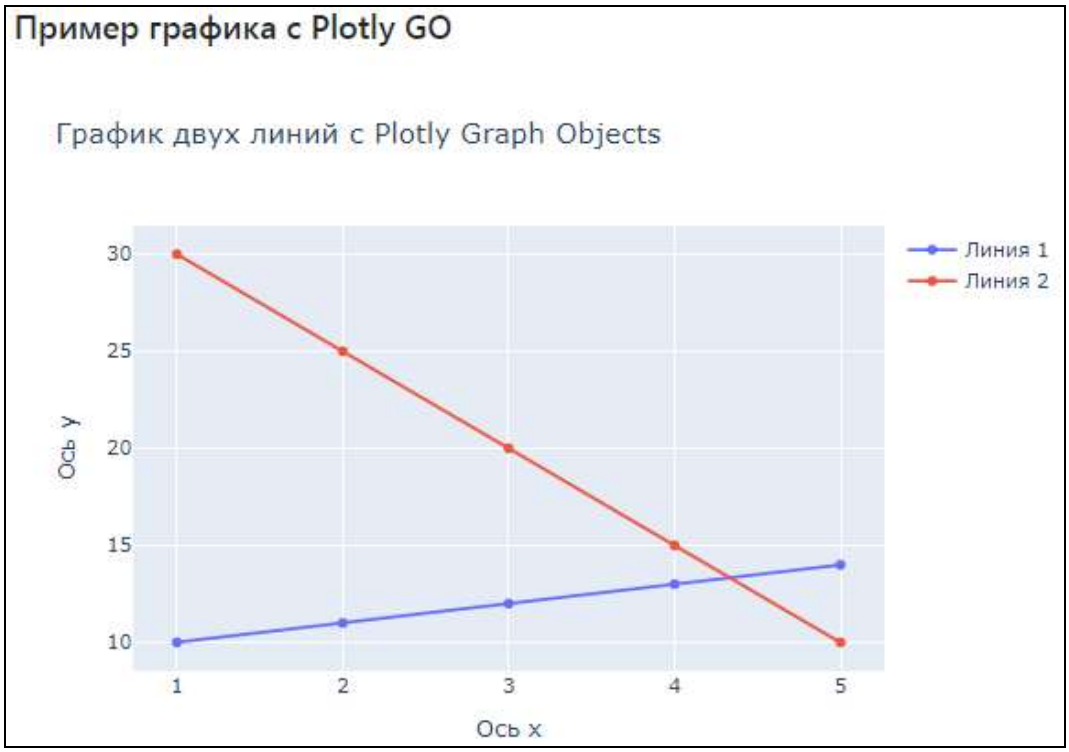


Рис. 4.9. Пример вывода графика двух линий, созданного в Plotly GO, в шаблон Django

4.4.3. Сетка графиков с интерфейсом Plotly Graph Objects

В библиотеке plotly имеется возможность разместить несколько графиков в одном окне с использованием функция make_subplots. Данная функция создает для группы графиков единый макет в виде таблицы из заданного количества строк и столбцов. Создаваемые графики можно вставлять в ячейки данной таблицы. Данная функция имеет несколько параметров:

- rows (по умолчанию 1) — количество строк в сетке графиков (должен быть больше нуля).
- cols (по умолчанию 1) — количество столбцов в сетке графиков (должен быть больше нуля);
- shared_xaxes (по умолчанию False) — назначение общих осей X для 2D-графиков;

  Возможные варианты:

  - True или «columns» — общие оси у графиков в одном столбце;
  - «rows» — общие оси у графиков в одной строке;
  - «all» — общие оси у всех графиков в сетке.
- shared_yaxes (по умолчанию False) — назначение общих осей Y для 2D-графиков;

  Возможные варианты:

- «columns» — общие оси у графиков в одном столбце,
- «rows» — общие оси у графиков в одной строке,
- «all» — общие оси у всех графиков в сетке.

- start_cell («bottom-left» или «top-left», по умолчанию «top-left») — выбор начальной ячейки в сетке графиков, используемой для установки доменов графиков;
- print_grid (по умолчанию True) — выводит строковое представление сетки графиков;
- horizontal_spacing (по умолчанию 0,2 / cols) — пространство между столбцами графиков в нормализованных координатах графика;
- vertical_spacing (по умолчанию 0,3 / rows) — пространство между строками графиков в нормализованных координатах графика;
- subplot_titles (список строк, по умолчанию None) — название каждого графика в виде списка в порядке следования строк.

Каждый график ассоциирован с собственными данными, или трейсом данных (trace). Новый трейс можно добавить с помощью метода add_trace, который принимает графический объект и добавляет его к графику.

Примечание.
Trace в Plotly — это слой (след) данных, который отображается на графике. У каждого следа свой набор данных и визуальных свойств, таких как тип графика, цвет и размер маркеров, стиль линий. Параметры Traces используются для создания отдельных компонентов графика, таких как линии, столбчатые диаграммы, круговые диаграммы, маркеры или другие визуальные элементы.

Теперь рассмотрим использование библиотеки plotly для графического представления нескольких наборов данных в одном макете с использованием функция make_subplots. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 4.29).

Листинг 4.29. Ссылка активации маршрута fig_subploats

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем fig_subploats (листинг 4.30).

Листинг 4.30. Маршрут URL с именем fig_subploats

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление fig_subploats (листинг 4.31).

Листинг 4.31. Представление (view) fig_subploats

Примечание.
В демо-версии книги листинги программ удалены.

В этом модуле выполнен импорт необходимых библиотек, создано 4 простых тестовых набора данных:

- df1 – данные для формирования столбчатой диаграммы;
- df2 – данные для формирования круговой диаграммы;
- df3 – данные для формирования линии;
- df3 – данные для формирования линии.

Затем с использованием функции make_subplots создано сетка (таблица – объект fig), состоящая из двух строк и двух столбцов. В каждую ячейку это таблицы будет вставлен свой график:

```
fig = make_subplots(rows=2, cols=2,
                specs=[[{'type': 'xy'}, {'type': 'domain'}],
                    [{'type': 'xy'}, {'type': 'xy'}]],
                subplot_titles=("Динамика роста населения",
                            "Структура продаж",
                            "Динамика доходов",
                            "Динамика убытков"))
```

Так как в ячейках таблицы будут вставлены графики разных типов, то в параметре specs задан тип графика для каждой ячейки. Здесь параметр «type: xy» говорит о том, что в ячейке будет график, который отображает зависимость двух величин. Параметр «type: 'domain» говорит о том, что в ячейке будет график, который отображает круговую диаграмму.

Примечание.
Параметр specs имеет значение по умолчанию «type: xy». Поэтому в тех случаях, когда в сетке графиков присутствуют только графики, отображающие зависимости двух величин, параметр specs можно не указывать

Параметр subplot_titles задает заголовки для графиков, которые будут размещены в сетке.

В следующих инструкциях в ячейки сетки графиков вставляются графические элементы, в данном случае это столбчатая диаграмма, круговая диаграмма и две линии:

```
fig.add_trace(go.Bar(x=df1['data_x'], y=df1['data_y'], name='Население'), row=1, col=1)
fig.add_trace(go.Pie(values=df2['values'], labels=df2['labels']), row=1, col=2)
fig.add_trace(go.Scatter(x=df3['data_x'], y=df3['data_y'], name='Доход'), row=2, col=1)
fig.add_trace(go.Scatter(x=df4['data_x'], y=df4['data_y'], name='Убытки'), row=2, col=2)
```

После этого для каждого графика формируются названия осей «x» и «y»:

```
# Обновить оси x
   fig.update_xaxes(title_text="Дата", row=1, col=1)
   fig.update_xaxes(title_text="Дата", row=2, col=1)
   fig.update_xaxes(title_text="Дата", row=2, col=2)

   # Обновить оси Y
   fig.update_yaxes(title_text="Население, млн.ч.", row=1, col=1)
   fig.update_yaxes(title_text="Доходы,руб.", row=2, col=1)
```

```
fig.update_yaxes(title_text="Убытки,руб.", row=2, col=2)
```

Обратите внимание, что для круговой диаграммы названия осей не задаются.

Далее выполняется обновление макета, в данном случае задаются размеры сетки графиков:

```
fig.update_layout(autosize=False,
                  width=800,
                  height=700,
                  hoverlabel=dict(font_size=15))
```

После этого график (весь макет) приведен к формату, в котором оно может быть показано на HTML странице (объект plot_div) и создана строка с заголовком страницы (title_str). Полученный график (plot_div) с заголовком страницы через контекстную переменную context передается в шаблон fig_plotly.html (листинг 4.32).

Листинг 4.32. Шаблон fig_plotly.html

Примечание.
В демо-версии книги листинги программ удалены.

Это достаточно простой шаблон с контейнером, в котором выведен объекты plot_div (график). Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 4» и активировать ссылку «Сетка графиков с Plotly GO».

Если выполнить набор вышеприведенных модулей, то получим следующий результат (рис. 4.10).

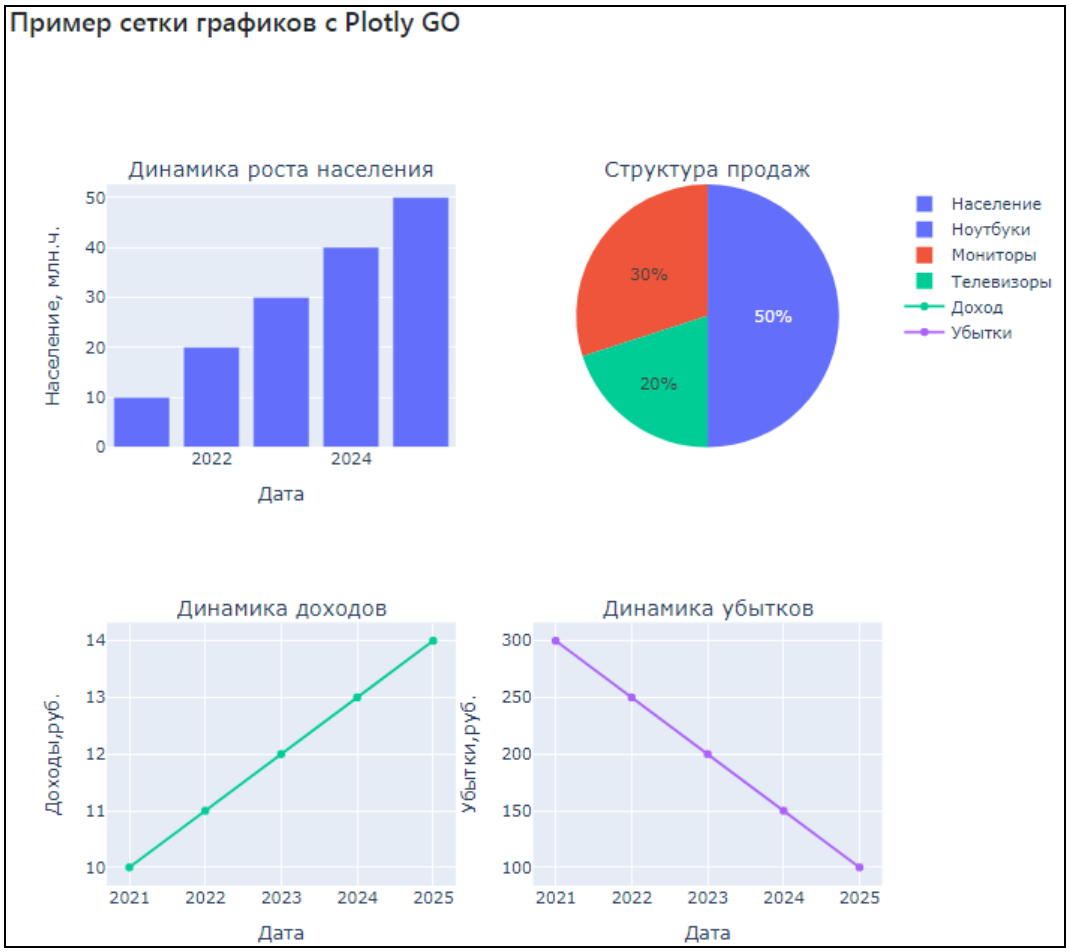


Рис. 4.10. Пример вывода сетки графиков, созданной в Plotly GO, в шаблон Django

Как видно из данного рисунка, в общем макете находится четыре графика разных типов. Каждый график имеет собственный заголовок и подписи к осям, при этом все графики имеют одну общую легенду.

4.5. Визуализация данных с библиотекой Pandas

Популярный инструмент анализа и обработки данных Pandas предоставляет функции построения графиков для своих объектов DataFrame и Series. В ранних версиях Pandas можно было строить графики в режиме бэкэнд с библиотекой matplotlib. А начиная с версии 4.8 для построения графиков можно использовать и библиотеку plotly (как бэкэнд на базе Plotly Express). При этом можно создавать интерактивные графики непосредственно из фрейма данных без прямого импорта библиотеки Plotly.

4.5.1. Визуализация данных в Pandas с библиотекой Matplotlib

Интеграция библиотек Pandas и Matplotlib позволяет создавать визуализации данных в традиционных (настольных) приложениях на Python. Нам же понадобится вывод графиков в шаблон Django. Поэтому сначала покажем на простом примере, как можно график, созданный средствами Matplotlib вывести в шаблон Django, а для этого необходимо предварительно установить библиотеку Matplotlib:

pip install matplotlib

Рассмотрим пример, в котором сгенерируем данные для графика прямой линии, сформируем график и выведем его в шаблон Django. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 4.33).

Листинг 4.33. Ссылка активации маршрута fig_matplot

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем fig_matplot (листинг 4.34).

Листинг 4.34. Маршрут URL с именем fig_matplot

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление fig_matplot (листинг 4.35).

Листинг 4.35. Представление (view) fig_matplot

Примечание.
В демо-версии книги листинги программ удалены.

В первых строках этого программного кода выполняется импорт необходимых библиотек. Затем подключается бэкенд «agg».

Примечание.
Здесь agg (Anti-Grain Geometry) – это высококачественный бэкенд для Matplotlib, предназначенный для рендеринга графиков в растровые форматы (например, PNG, JPG, TIFF). Он отличается высокой скоростью работы и качеством визуализации, особенно при работе с векторной графикой. Agg является одним из самых популярных бэкендов Matplotlib и часто используется по умолчанию, но в некоторых случаях его нужно активировать принудительно.

После этого с использованием библиотеки Matplotlib формируется график прямой линии.

```
plt.plot(range(10))
```

В таком виде он не может быть выведен в шаблон Django. Теоретически график можно сохранить на диске в формате png и через его URL адрес вывести в шаблон. Но аналогичная операция выполнится быстрее, если мы сохраним его в памяти. Поэтому затем изображение сохраняется в буфере, где преобразуется к изображению формата png (объект image_png). После этого читаем изображение из буфера и преобразуем к виду, в котором оно может быть показано на HTML странице (объект graphic). Полученное изображение (graphic) через контекстную переменную context передается в шаблон fig_matplot.html (листинг 4.36).

Листинг 4.36. Шаблон fig_matplot.html

Примечание.
В демо-версии книги листинги программ удалены.

Это достаточно простой шаблон с контейнером, в котором выведен объекты data (изображение). Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 4» и активировать ссылку «График matplotlib».

Если выполнить набор вышеприведенных модулей, то получим следующий результат (рис. 4.11).

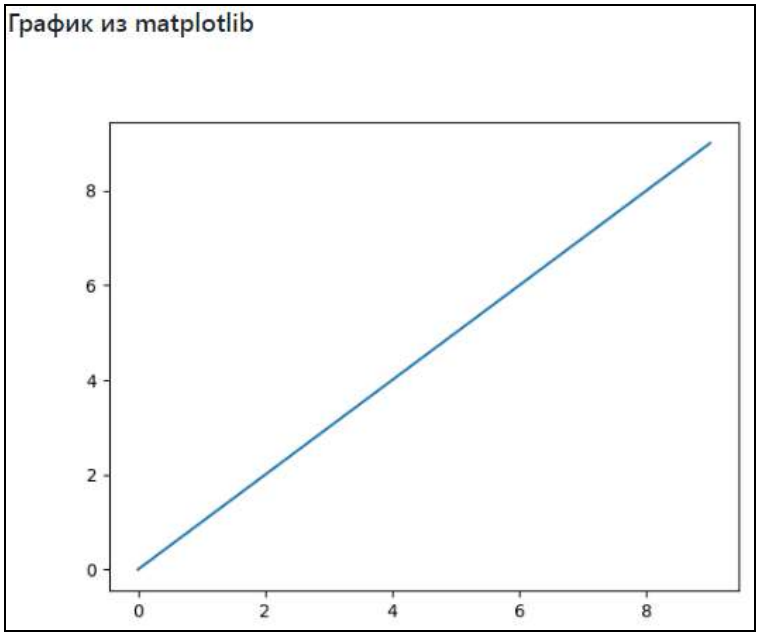


Рис. 4.11. Пример вывода графика matplotlib в шаблон Django

Следует отметить, что это не лучшее решение, так как выведенный график лишен интерактивности. Но достаточно часто возникают ситуации, когда интерактивность не нужна, а библиотека matplotlib позволяет сформировать график с минимальным количеством строк кода.

4.5.2. Визуализация данных в Pandas с библиотекой Plotly

Теперь рассмотрим возможность построения графиков с библиотеками Pandas и Plotly. Чтобы активировать Plotly в режиме бэкэнд, понадобится установить библиотеку Plotly, а затем просто вызвать ее с использованием инструкции pd.options.plotting.backend. Данная инструкция позволяет использовать большинство функций точно так же, как если бы вы выполнили прямой импорт Plotly Express. Установить библиотеку Plotly можно с использованием следующей команды:

```
pip install plotly
```

Реализуем пример формирования группы графиков с использованием библиотеки Pandas. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 4.37).

Листинг 4.37. Ссылка активации маршрута fig_pandas

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем fig_pandas (листинг 4.38).

Листинг 4.38. Маршрут URL с именем fig_pandas

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление fig_pandas (листинг 4.39).

Листинг 4.39. Представление (view) fig_pandas

Примечание.
В демо-версии книги листинги программ удалены.

В этом модуле выполнен импорт библиотеки Pandas. Обратите внимание, что здесь нет прямого импорта библиотеки Plotly. Так как она уже встроена в библиотеку Pandas, то просто выполняется ее подключение в фоновом режиме (как backend):

```
pd.options.plotting.backend = "plotly"
```

Затем создан простейший набор данных (объект df), и на основе этих данных сформировано четыре график с использованием различных вариантов метода df.plot (объекты fig1, fig2, fig3, fig4).

Первый график (объект fig1) - это график линии с параметрами по умолчанию.

Второй график (объект fig2) представляет собой тот же график линии, но для макета задана тема (темный фон - plotly_dark) и сформированы подписи к осям.

Третий график (объект fig3) представляет собой столбчатую диаграмму, для которой задана тема (ggplot2) и подписи к осям.

Четвертый график объект (fig4) представляет собой столбчатую диаграмму с группировкой данных, с темой по умолчанию и подписями к осям.

После этого все графики приведены к формату, в котором они могут быть показаны на HTML странице (объекты plot_div1, plot_div2, plot_div3, plot_div4). Полученные графики через контекстную переменную context передается в шаблон fig_pandas.html (листинг 4.40).

Листинг 4.40. Шаблон fig_pandas.html

Примечание.
В демо-версии книги листинги программ удалены.

Это шаблон с контейнером, в котором имеется четыре блока для вывода объектов plot_div1, plot_div2, plot_div3, plot_div4. Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 4» и активировать ссылку «График в Pandas».

Если выполнить набор вышеприведенных модулей, то получим следующий результат (рис. 4.12).

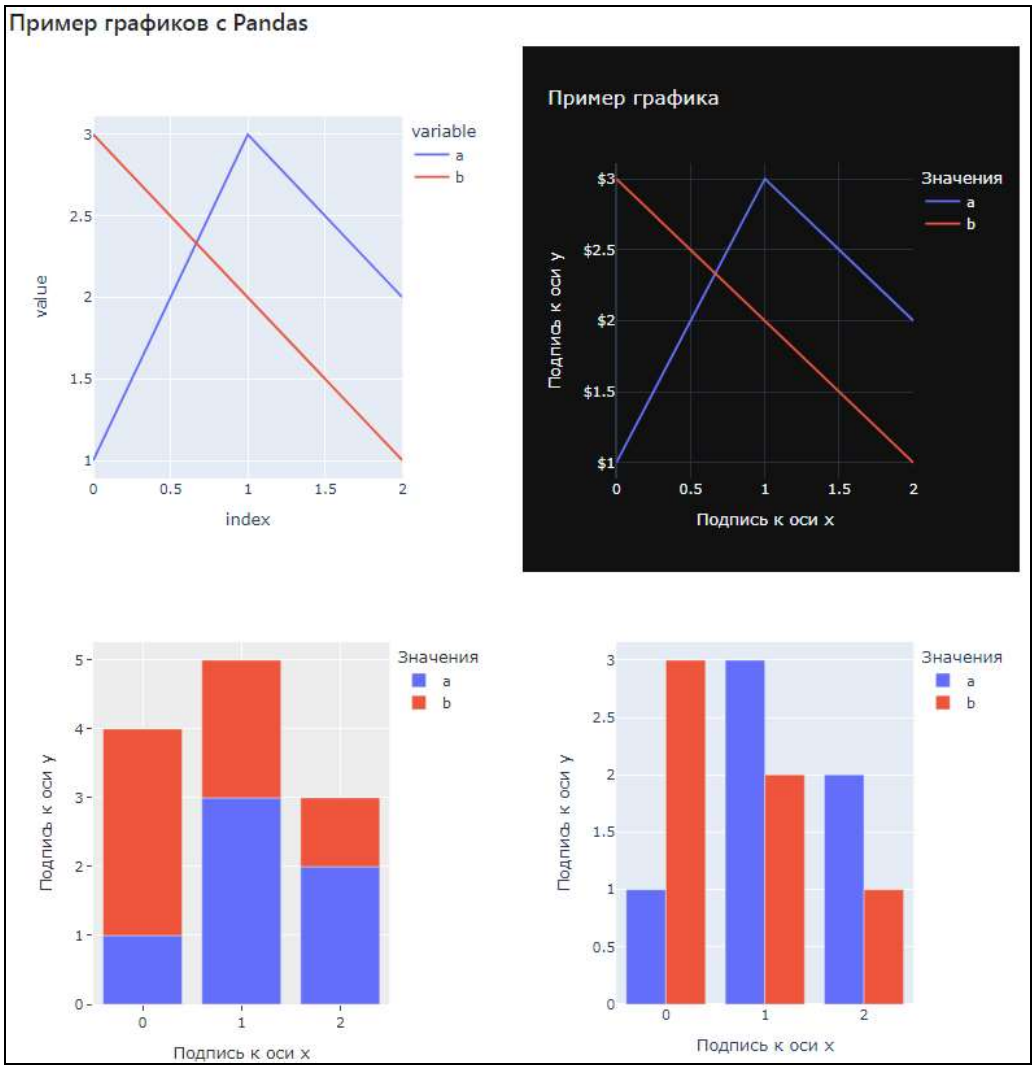


Рис. 4.12. Пример вывода графика в шаблон Django с библиотекой Pandas и Plotly

4.6. Краткие итоги

Итак, мы познакомились с некоторыми базовыми возможностями визуализации данных с использованием библиотек Pabdas, Matplotlib и Plotly. В это главе не был сделан обзор всей палитры графиков, которые можно строить с указанными выше библиотеками, для этого нужно писать отдельную книгу. Здесь показан принципиальный подход вывода графических объектов в шаблоны Django: сформировать график средствами соответствующей библиотеки, трансформировать объект в формат для вывода на html страницу, в шаблоне Django создать тег для вывода графического объекта.

В материалах этой главы для упрощения примеров были созданы тестовые данные, которые имеют стандартную структуру DataFrame библиотеки Pandas. Однако реальные данные, полученные из открытых источником, не всегда будут соответсвтовать структуре данных библиотеки Pandas. В этих данных могут быть пропуски, даты и числа представлены строковым типом, структура данных может отличаться от структуры DataFrame. Перед графической визуализацией такие данные нужно «очистить», то есть привести к виду, удобному для графической визуализации. Очистки и подготовки данных к визуализации и посвящена следующая глава.

Глава 5. Очистка данных и подготовка к графической визуализации

Анализ временных рядов, равно как и данных других типов, практически невозможно представить себе без их визуализации в графическом виде. Набор данных в виде цифр из десятка столбцов и нескольких тысяч строк практически ни о чем не говорит, однако те же данные в графическом виде могут рассказать о многом. Однако наборы данных DataFrame, полученные через API, загруженные из CSV файлов или из файлов формата JSON не всегда готовы к тому, чтобы их можно было представить в графическом виде. Например, даты могут быть представлены в строковом формате, хотя внешне выглядят как даты. Аналогично и цифровые данные, особенно загруженные из CSV или XML файлов, могут представлять собой строки с запятой в качестве разделителя дробной части. Набор данных DataFrame может иметь заголовок из нескольких строк, колонку с неиндексированными датами, многоуровневые заголовки столбцов. В связи с этим, перед тем как сформировать тот или иной график нужно выполнить подготовку данных к визуализации. Это могут быть следующие действия:

- проверка типов данных (строки, даты, числа);
- анализ таблицы набора данных (структура заголовка, имена столбцов, наличие индекса, наличие колонок или строк с ключевой информацией для фильтрации или группировки данных и т.п.);
- выборка из набора данных только определенных строк или столбцов;
- необходимость смены индекса или выполнения сортировки;
- необходимость фильтрации или группировки данных;
- проверка полноты данных (наличие пропусков и их заполнение).

Набор этих процедур принято называть «очистка» данных. Перед тем, как формировать график из набора данных, нужно вывести его на печать (или в окно терминала) и выполнить анализ его состава и структуры. При создании веб приложений желательно загрузку и очистку данных реализовывать в виде отдельных программных модулей, чтобы не усложнять логику работы основной программы. Отлаженный модуль загрузки и очистки данных можно просто импортировать в веб приложение и использовать многократно. Рассмотрим процедуры подготовки и очистки данных на некоторых примерах временных рядов, загруженных из локальных файлов и полученных через API.

Из материалов этой главы вы узнаете:

- как подготовить и отобразить данные, загруженные из CSV файла;
- как подготовить и отобразить данные, загруженные из файла формата json;
- как подготовить и отобразить данные, загруженные из API Центрального банка России;
- как подготовить и отобразить данные, загруженные из API Московской биржи;
- как подготовить и отобразить данные, загруженные из API Всемирного банка;
- как подготовить и отобразить данные, загруженные из API Stooq;
- как подготовить и отобразить данные, загруженные из API Yahoo Finance;

5.1. Модуль Python для загрузки и анализа данных из CSV файла

Создадим программный модуль Python, который обеспечит загрузку данных из CSV файла. Мы будем использовать файл с фрагментом реальных данных о динамике экспорта нефти, на которых научимся выполнять предварительный анализ данных и приведения их к виду, приемлемому для построения графиков. Исходный набор данных будет содержать множество колонок с данными строкового типа. Нам потребуется выбрать только две колонки, данные этих колонок привести к нужному типу и сформировать набора данных типа Series. Такой набор данных удобен для графической визуализации временного ряда, так как содержит индексную колонку с датами (ось «x») и колонку с данными (ось «y»). Программный код этого модуля приведен в листинге 5.1.

Листинг 5.1. Модуль get_data_csv.py для загрузки данных из CSV файла

Примечание.
В демо-версии книги листинги программ удалены.

В этом программном модуле выполнен импорт необходимых библиотек и создана функция с именем get_data(name_csv), которая принимает один параметр – имя CSV файла (name_csv).

В теле данной функции выполняется чтение данных из CSV файла, вывод в окно терминала структуры набора данных, типа данных каждой колонки и результатов их преобразования. Чтобы порядок работы этого программного кода был более понятен, покажем структуру полученных данных. Итак, после запуска этого программного модуля на печать будут выведены следующие данные.

```
Исходный набор данных
        Period     Year    Month   Oil Type      Volume (m3/d)  Volume (bbl/d)
0     06/01/1985  1985     June       Light      28019.093333      1.763201e+05
1     06/01/1985  1985     June       Medium     14596.950000      9.185652e+04
2     06/01/1985  1985     June       Heavy      22124.423333      1.392258e+05
3     06/01/1985  1985     June       Total      64740.466666      4.074025e+05
4     07/01/1985  1985     July       Light      25181.764516      1.584652e+05
...        ...        ...     ...          ...             ...              ...
1890  01/01/2025  2025   January    Total       685020.098134     4.310734e+06
1891  02/01/2025  2025  February    Light       148703.472320     9.357697e+05
1892  02/01/2025  2025  February    Medium      16473.867500      1.036677e+05
1893  02/01/2025  2025  February    Heavy       521833.029439     3.283821e+06
1894  02/01/2025  2025  February    Total       687010.369260     4.323258e+06

[1895 rows x 6 columns]
-----------------------
Period           object
Year             int64
Month            object
Oil Type         object
Volume (m3/d)    float64
```

```
Volume (bbl/d)    float64
dtype: object
-----------------------
Данные для формирования графика
Period
1985-06-01    4.074025e+05
1985-07-01    3.680593e+05
1985-08-01    3.925685e+05
1985-09-01    5.325918e+05
1985-10-01    4.984505e+05
                 ...
2024-10-01    4.150392e+06
2024-11-01    4.233533e+06
2024-12-01    4.310521e+06
2025-01-01    4.310734e+06
2025-02-01    4.323258e+06
Name: Volume (bbl/d), Length: 477, dtype: float64
--------------------------
```

Из полученных результатов видно, что исходный CSV файл содержит 1895 строк, одну индексную колонку и 6 колонок с данными. Для построения графика нам из этого набора данных нужно выбрать всего две колонки: период – Period и объем экспорта - Volume (bbl/d). Кроме того, нам нужны не все строки. На основе значений колонки с типом нефти (Oil Type) нужно оставить только те строки, в которых содержатся данные об общем объеме экспорта (Total). Поэтому на полученный набор данных накладывается следующий фильтр:

```
series = series.loc[series['Oil Type'] == 'Total'].filter(['Period', 'Volume (bbl/d)'])
```

Колонка с датами (Period) имеет строковые значения. Для формирования временного ряда необходимо изменить тип данных этой колонки на datetime (американский формат) и сделать эту колону индексной:

```
series['Period'] = series['Period'].transform(lambda x: datetime.strptime(x, '%m/%d/%Y'))
series.set_index(keys='Period', drop=True, inplace=True)
```

Наконец мы создаем объект Series:

```
series = series.squeeze(axis=1)
```

Данный объект будет иметь всего две колонки: индексную, состоящую из дат (Period) и колонку с данными об общем объеме экспортируемой нефти - Volume (bbl/d) с типом данных - dtype: float64. Полученный объект series будет возвращен в точку вызова.

Теперь в этом программном модуле можно закомментировать строки вывода данных в окно терминала и использовать его в приложениях Django. Что и сделано в следующем разделе.

> ***Примечание.***
> Представленный в данном разделе программный модуль не включен в список исполняемых примеров интерактивной цифровой книги, так как он может работать только в рамках приложения Python локального компьютера, или быть вызванным из другого модуля веб приложения.

5.2. Модули Django для визуализации данных из CSV файла

Реализуем пример формирования графика на основе данных, загруженных из CSV файла с использованием интерфейса Plotly Graph Objects. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 5.2).

Листинг 5.2. Ссылка активации маршрута fig_csv

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем fig_csv (листинг 5.3).

Листинг 5.3. Маршрут URL с именем fig_csv

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление fig_csv (листинг 5.4).

Листинг 5.4. Представление (view) fig_csv

Примечание.
В демо-версии книги листинги программ удалены.

В этом модуле выполнен импорт необходимых библиотеки и модулей. Затем был сформирован путь к файлу с именем oil_exports.csv (объект path_csv).

После этого выполнено обращение к функции get_data (path_csv) модуля get_data_csv, код которого приведен в предыдущем разделе. В эту функцию передан полный путь к CSV файлу (объект path_csv). Загруженные из CSV файла данные сохранены в объекте df.

На следующем шаге создан пустой макет графика (объект fig), к этому макету добавлена линия, сформированная на основе данных объекта df:

fig.add_trace(go.Scatter(x=df.index, y=df.values))

Здесь данными для оси x является индексная колонка объекта df, а для оси y - колонка с данными объекта df. Затем к осям графика добавлены поясняющие надписи.

После этого график приведен к формату, в котором он может быть показано на HTML странице (объект plot_div) и создана строка с заголовком страницы (title_str). Полученный график (plot_div) с заголовком страницы через контекстную переменную context передается в шаблон fig_plotly.html (листинг 5.5).

Листинг 5.5. Шаблон fig_plotly.html

Примечание.
В демо-версии книги листинги программ удалены.

Это шаблон с контейнером, в котором имеется блок для вывода объекта plot_div. Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 5» и активировать ссылку «График из файла CSV».

Если выполнить набор вышеприведенных модулей, то получим следующий результат (рис.5.1).



Рис. 5.1. График в шаблоне Django с данными, загруженными из CSV файла

5.3. Модуль Python для загрузки и анализа данных из файла формата JSON

Создадим программный модуль Python, который обеспечит загрузку данных из файла формата JSON. Мы будем использовать файл с фрагментом реальных данных о динамике цен на нефть, на которых научимся выполнять предварительный анализ данных и

приведения их к виду, приемлемому для построения графиков. Исходный набор данных будет содержать множество колонок. Нам нужно будет выбрать только две колонки, если потребуется, то данные этих колонок привести к нужному типу. Такой набор данных будет удобен для графической визуализации временного ряда, так как в нем останется только индексная колонка с датами (ось «x») и колонка с данными (ось «y»). Программный код этого модуля приведен в листинге 5.6.

Листинг 5.6. Модуль get_data_json.py для загрузки данных из файла формата JSON

Примечание.
В демо-версии книги листинги программ удалены.

В этом программном модуле выполнен импорт необходимых библиотек и создана функция с именем def get_data(name_json), которая принимает один параметр – имя JSON файла (name_json).

В теле данной функции выполняется чтение данных из JSON файла, вывод в окно терминала структуры набора данных, типа данных каждой колонки и результатов их преобразования. Чтобы порядок работы этого программного кода был более понятен, покажем структуру полученных данных. Итак, после запуска этого программного модуля в окно терминала будут выведены следующие данные.

```
Данные из json файла
                Open        High         Low       Close   Volume
2020-04-29  13.350000  16.780001  12.670000  15.060000  554134
2020-04-30  15.640000  19.440001  15.450000  18.840000  545344
2020-05-01  19.040001  20.480000  18.070000  19.780001  403201
2020-05-04  19.110001  21.420000  18.049999  20.389999  311488
2020-05-05  21.240000  25.490000  21.129999  24.559999  387240
-----------------------------
Типы данных из json
Open      float64
High      float64
Low       float64
Close     float64
Volume      int64
dtype: object
-----------------------------
Индекс
DatetimeIndex(['2020-04-29', '2020-04-30', '2020-05-01', '2020-05-04',
               '2020-05-05', '2020-05-06', '2020-05-07', '2020-05-08',
               '2020-05-11', '2020-05-12',
               ...
               '2025-04-15', '2025-04-16', '2025-04-17', '2025-04-21',
               '2025-04-22', '2025-04-23', '2025-04-24', '2025-04-25',
               '2025-04-28', '2025-04-29'],
              dtype='datetime64[ns]', length=1258, freq=None)
-----------------------------
2020-04-29    15.060000
2020-04-30    18.840000
```

```
2020-05-01    19.780001
2020-05-04    20.389999
2020-05-05    24.559999
Name: Close, dtype: float64
```

Из полученных результатов видно, что исходный JSON файл содержит одну индексную колонку и 5 колонок с данными. Для построения графика нам из этого набора данных нужно выбрать всего две колонки: период – даты (индексная колонка) и цена нефти на момент закрытия торгов (Close). Солонка Close имеет тип float64, индексная колонка с датами имеет тип datetime64[ns]. То есть выполнять приведение данных к нужному типу нет необходимости, однако желательно убрать лишние колонки, это делается в следующей инструкции:

```
df = df['Close']
```

Полученный объект df будет возвращен в точку вызова.

Обратите внимание, что в отличие от файла формата CSV, который возвращает данные в строковом формате, файл формата JSON возвращает данные в формате, пригодном для графической визуализации.

Теперь в этом программном модуле можно закомментировать строки вывода данных в окно терминала и использовать его в приложениях Django. Что и сделано в следующем разделе.

> ***Примечание.***
> Представленный в данном разделе программный модуль не включен в список исполняемых примеров интерактивной цифровой книги, так как он может работать только в рамках приложения Python локального компьютера, или быть вызванным из другого модуля веб приложения.

5.4. Модули Django для визуализации данных из JSON файла

Реализуем пример формирования графика на основе данных, загруженных из файла формата JSON с использованием интерфейса Plotly Graph Objects. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 5.7).

Листинг 5.7. Ссылка активации маршрута fig_json

> ***Примечание.***
> В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем fig_json (листинг 5.8).

Листинг 5.8. Маршрут URL с именем fig_json

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление fig_json (листинг 5.9).

Листинг 5.9. Представление (view) fig_json

Примечание.
В демо-версии книги листинги программ удалены.

В этом модуле выполнен импорт необходимых библиотеки и модулей. Затем был сформирован путь к файлу с именем price_oil.json (объект path_json).

После этого выполнено обращение к функции get_data (path_json) модуля get_data_json, код которого приведен в предыдущем разделе. В эту функцию передан полный путь к JSON файлу (объект path_json). Загруженные из JSON файла данные будут сохранены в объекте df.

На следующем шаге создан пустой макет графика (объект fig), к этому макету добавлена линия, сформированная на основе данных объекта df:

fig.add_trace(go.Scatter(x=df.index, y=df.values))

Здесь данными для оси x является индексная колонка объекта df, а для оси y - колонка с данными объекта df. Затем к осям графика добавлены поясняющие надписи.

После этого график приведен к формату, в котором он может быть показано на HTML странице (объект plot_div) и создана строка с заголовком страницы (title_str). Полученный график (plot_div) с заголовком страницы через контекстную переменную context передается в шаблон fig_plotly.html (листинг 5.10).

Листинг 5.10. Шаблон fig_plotly.html

Примечание.
В демо-версии книги листинги программ удалены.

Это шаблон с контейнером, в котором имеется блок для вывода объекта plot_div. Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 5» и активировать ссылку «График из файла JSON».

Если выполнить набор вышеприведенных модулей, то получим следующий результат (рис.5.2).

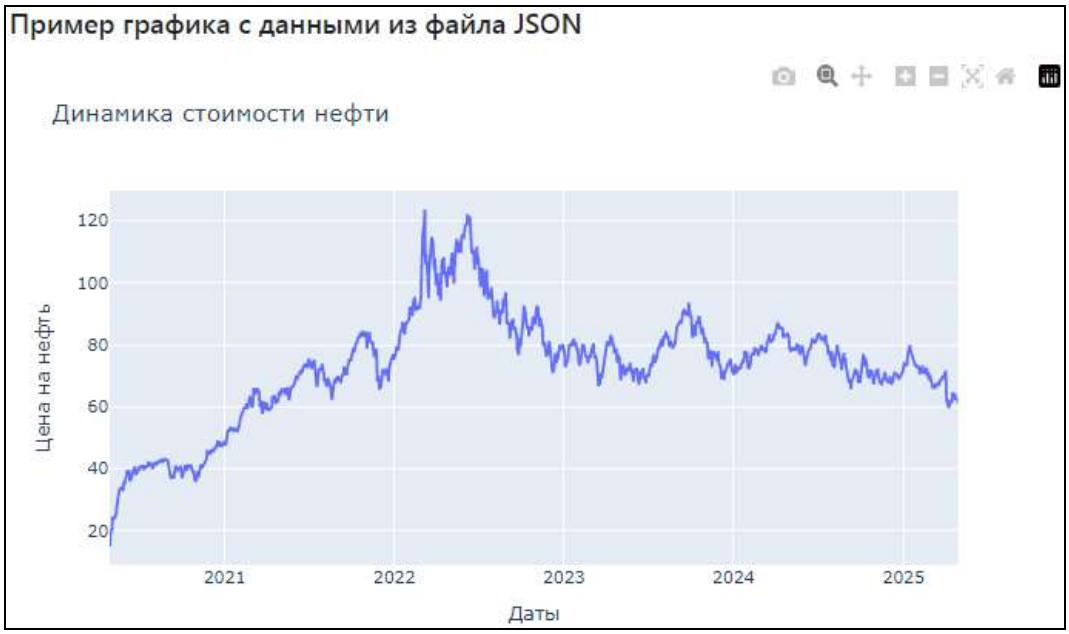


Рис. 5.2. График в шаблоне Django с данными, загруженными из JSON файла

5.5. Библиотека Pandas для работы со строковыми объектами JSON

Набор данных Data Frame можно сохранить в формате json не только в виде файла, но и в виде строкового объекта. Это удобно в тех случаях, когда нужно передать данные из одного модуля в другой, или выполнить экспорт - импорт данных между приложениями. Для записи набора данных в строковый объект используется следующий метод библиотеки pandas:

data_json = df.to_json(orient='index')

Для загрузки данных из строкового объекта json в набор данных Data Frame используется следующий метод библиотеки pandas:

df = pd.read_json(data_json, orient='index')

Напишем модули приложения Dnango, в которых загрузим данные о стоимости нефти на бирже от Yahoo Finance, запишем эти данные в строковый объект формата json, загрузим данные из строки json в объект Data Frame и выполним визуализацию данных с библиотекой Plotly. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 5.11).

Листинг 5.11. Ссылка активации маршрута fig_json_str

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем fig_json_str (листинг 5.12).

Листинг 5.12. Маршрут URL с именем fig_json_str

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление fig_json_str (листинг 5.13).

Листинг 5.13. Представление (view) fig_json_str

Примечание.
В демо-версии книги листинги программ удалены.

В этом модуле выполнен импорт необходимых библиотек и загружены текущие данные с платформы Yahoo Finance о динамике стоимости нефти (объект df). Это данные за последние пять лет, начиная с текущей даты.

После этого полученный набор данных загружен в строковую переменную data_json в формате JSON:

```
data_json = df.to_json(orient='index')
```

В таком виде данные могут быть переданы по сети интернет или другому модулю приложения. Здесь этот набор данных передан другому объекту df в этом же приложении.

```
df = pd.read_json(data_json, orient='index')
```

Мы знаем, что в формате JSON данные не «теряют» свой тип, то есть после получения данных из источника числовые данные останутся числовыми, а даты - датами. Поэтому, все, что нам нужно сделать для формирования графика, это выбрать из набора данных одну колонку:

```
df = df['Close']
```

В этой инструкции был сформирован набор данных, состоящий из индексной колонки с датами и колонки с данными о стоимости нефти на момент закрытия торгов (Close).

На следующем шаге создан пустой макет графика (объект fig), к этому макету добавлена линия, сформированная на основе данных объекта df:

```
fig.add_trace(go.Scatter(x=df.index, y=df.values))
```

Здесь данными для оси x является индексная колонка объекта df, а для оси y - колонка с данными объекта df. После этого к осям графика добавлены поясняющие надписи.

После этого график приведен к формату, в котором он может быть показано на HTML странице (объект plot_div) и создана строка с заголовком страницы (title_str). Полученный график (plot_div) с заголовком страницы через контекстную переменную context передается в шаблон fig_plotly.html (листинг 5.14).

Листинг 5.14. Шаблон fig_plotly.html

Примечание.
В демо-версии книги листинги программ удалены.

Это шаблон с контейнером, в котором имеются блоки для вывода заголовка страницы и объекта plot_div. Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 5» и активировать ссылку «График из строки JSON».

Если выполнить набор вышеприведенных модулей, то получим следующий результат (рис.5.3).

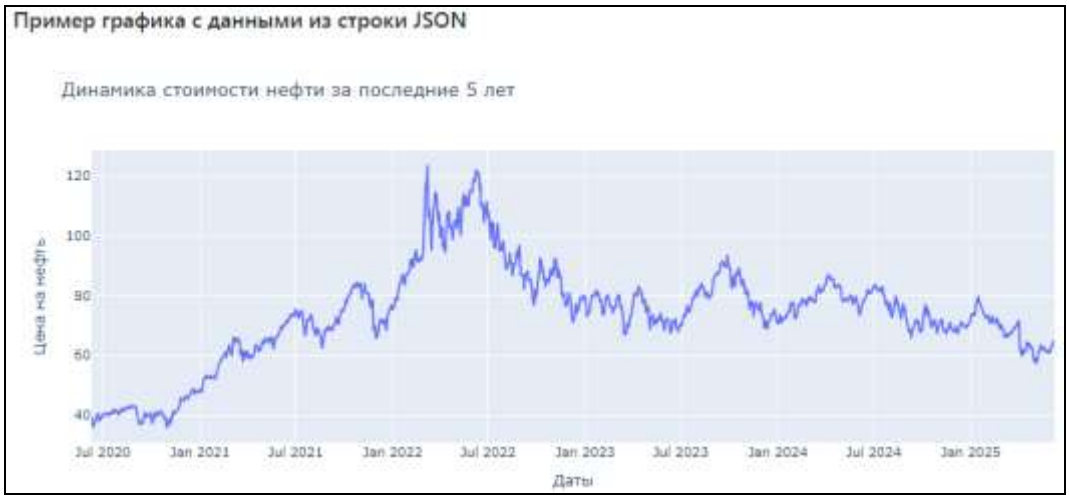


Рис. 5.3. График в шаблоне Django с данными, загруженными из строки JSON

5.6. Загрузки очистки данных через API Центрального банка России

Через API центрального банка России можно загружать данные в формате XML. Рассмотрим несколько примеров загрузки данных, их чистки и визуализации с использованием стандартных методов библиотек Pandas и Plotly.

5.6.1. Модуль Python для загрузки очистки данных через API Центрального банка (курсы валют)

Создадим программный модуль Python, который обеспечит загрузку данных о курсе рубля из API Центрального банка России, их очистку и подготовку к выводу в графическом виде. Мы будем использовать данные, получаемые в режиме реального времени в формате XML.

Исходный набор данных будет содержать несколько колонок с данными в строковом формате. Нам нужно будет данные этих колонок привести к нужному типу (числовому и datenime). Такой набор данных будет удобен для графической визуализации временного ряда. Программный код этого модуля приведен в листинге 5.15.

Листинг 5.15. Модуль curs_cbr.py для загрузки данных из API Центрального банка

Примечание.
В демо-версии книги листинги программ удалены.

В этом программном модуле выполнен импорт библиотеки Pandas и создана функция с именем def get_data(), которая принимает следующие параметры:

- d_start – дата начала периода;
- d_end– дата конца периода;
- val – код пары валют.

В теле данной функции выполняется формирование URL адреса запроса к API Центрального банка (объект url) и загрузка данных в объект dat. Чтобы порядок работы этого программного кода был более понятен, покажем структуру полученных данных. Итак, после запуска этого программного модуля в окно терминала будут выведены следующие данные.

```
      Date      Id        Nominal   Value    VunitRate
0  01.01.2020  R01235       1      61,9057   61,9057
1  10.01.2020  R01235       1      61,2340    61,234
2  11.01.2020  R01235       1      61,2632   61,2632
3  14.01.2020  R01235       1      60,9474   60,9474
4  15.01.2020  R01235       1      61,4140    61,414
-------------------------------------
      Date      Id        Nominal   Value    VunitRate
0 2020-01-01  R01235       1      61.9057   61,9057
1 2020-01-10  R01235       1      61.2340   61,234
2 2020-01-11  R01235       1      61.2632   61,2632
3 2020-01-14  R01235       1      60.9474   60,9474
4 2020-01-15  R01235       1      61.4140    61,414
```

Здесь первый набор содержит данные, полученные через API, второй набор содержит очищенные данные. Из полученных результатов видно, что все даты имеют строковый формат, а разделителем дробной и целой части чисел является запятая. Поэтому в столбцах с числовыми данными меняем запятую на десятичную точку, приводим данные к числовому формату (с плавающей точкой):

```
dat['Value'] = dat['Value'].str.replace(',', '.')
dat['Value'] = dat['Value'].astype(float)
```

В столбце с датами меняем точку на дефис, приводим данные к формату datetime:

```
dat['Date'] = dat['Date'].str.replace('.', '-')
dat['Date'] = pd.to_datetime(dat['Date'], format='%d-%m-%Y',
errors='coerce')
```

Итоговый набор данных dat возвращаем в точку вызова.

Теперь в этом программном модуле можно закомментировать строки вывода данных в окно терминала и использовать его в приложениях Django. Что и сделано в следующем разделе.

Примечание.
Представленный в данном разделе программный модуль не включен в список исполняемых примеров интерактивной цифровой книги, так как он может работать только в рамках приложения Python локального компьютера, или быть вызванным из другого модуля веб приложения.

5.6.2. Модули Django для визуализации данных из API Центрального банка (курсы валют)

Реализуем пример формирования графика динамики курса валют на основе данных, полученных из API Центрального банка с использованием интерфейса Plotly Graph Objects. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 5.16).

Листинг 5.16. Ссылка активации маршрута fig_curs_val

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем fig_curs_val (листинг 5.17).

Листинг 5.17. Маршрут URL с именем fig_curs_val

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление fig_curs_val (листинг 5.18).

Листинг 5.18. Представление (view) fig_curs_val

Примечание.
В демо-версии книги листинги программ удалены.

В этом модуле выполнен импорт необходимых библиотек и модуля curs_cbr, код которого приведен в предыдущем разделе. Затем сформированы даты начала и конца запрашиваемого периода (d1, d2). После этого трижды выполнено обращение к API Центрального банка для получения данных о курсах валют: доллар (kod_val = 'R01235'), евро (kod_val = 'R01239'), фунт (kod_val = 'R01035'):

```
df_d = curs_cbr.get_data(d1, d2, kod_val)
df_e = curs_cbr.get_data(d1, d2, kod_val)
df_f = curs_cbr.get_data(d1, d2, kod_val)
```

Полученные данные сохранены в объектах: df_d, df_e, df_f.

На следующем шаге создан пустой макет графика (объект fig), к этому макету добавлены линии, сформированные на основе загруженных данных:

```
fig.add_trace(go.Scatter(x=df_d['Date'], y=df_d['Value'], name='Курс $'))
fig.add_trace(go.Scatter(x=df_e['Date'], y=df_e['Value'], name='Курс евро'))
fig.add_trace(go.Scatter(x=df_f['Date'], y=df_f['Value'], name='Курс фунт'))
```

Здесь данными для оси x является колонка Date, а для оси y - колонка с данными Value объектов df_d, df_e, df_f. После этого к осям графика добавлены поясняющие надписи.

После этого график приведен к формату, в котором он может быть показано на HTML странице (объект plot_div) и создана строка с заголовком страницы (title_str). Полученный график (plot_div) с заголовком страницы через контекстную переменную context передается в шаблон fig_plotly.html (листинг 5.19).

Листинг 5.19. Шаблон fig_plotly.html

Примечание.
В демо-версии книги листинги программ удалены.

Это шаблон с контейнером, в котором имеются блоки для вывода заголовка страницы и объекта plot_div. Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 5» и активировать ссылку «Курсы валют (Центробанк)».

Если выполнить набор вышеприведенных модулей, то получим следующий результат (рис. 5.4).



Рис. 5.4. График в шаблоне Django с данными и курсах валю, загруженными из API Центробанка

5.6.3. Модуль Python для загрузки и очистки данных из API Центрального банка (цены на драгметаллы)

Создадим программный модуль Python, который обеспечит загрузку данных о ценах на драгметаллы из API Центрального банка России, их очистку и подготовку к выводу в графическом виде. Мы будем использовать данные, получаемые в режиме реального времени в формате XML. Исходный набор данных будет содержать несколько колонок с данными в строковом формате. Нам нужно будет данные этих колонок привести к нужному типу (числовому и datenime), а также выполнить поворот таблицы. Такой набор данных будет удобен для графической визуализации временного ряда. Программный код этого модуля приведен в листинге 5.20.

Листинг 5.20. Модуль drag_met_cbr.py для загрузки данных из API Центрального банка

Примечание.
В демо-версии книги листинги программ удалены.

В этом программном модуле выполнен импорт необходимых библиотек и создана функция с именем def get_data(), которая принимает следующие параметры:

- d_start – дата начала периода;
- d_end– дата конца периода.

В теле данной функции выполняется формирование URL адреса запроса к API Центрального банка (объект url) и загрузка данных в объект df. Чтобы порядок работы

этого программного кода был более понятен, покажем структуру полученных данных. Итак, после запуска этого программного модуля в окно терминала будут выведены следующие данные.

```
      Date        Code     Buy       Sell
0  09.06.2020    1        3716,12  3716,12
1  09.06.2020    2        38,72     38,72
2  09.06.2020    3        1827,31  1827,31
3  09.06.2020    4        4282,77  4282,77
4  10.06.2020    1        3770,05  3770,05
             Золото  Серебро  Платина  Палладий
Date
2020-06-09  3716.12   38.72  1827.31   4282.77
2020-06-10  3770.05   38.93  1817.13   4345.22
2020-06-11  3789.36   39.08  1850.94   4246.80
2020-06-12  3848.83   39.72  1833.41   4282.41
2020-06-16  3871.06   38.67  1821.92   4304.70
```

Здесь первый набор содержит данные, полученные через API, второй набор содержит очищенные данные. Из полученных результатов видно, что все даты имеют строковый формат, а разделителем дробной и целой части чисел является запятая. Поэтому в столбце с числовыми данными (Sell – цена продажи) меняем запятую на десятичную точку, приводим данные к числовому формату (с плавающей точкой):

```
df['Sell'] = df['Sell'].str.replace(',', '.')
df['Sell'] = df['Sell'].astype(float)
```

В столбце с датами меняем точку на дефис, приводим данные к формату datetime:

```
df['Date'] = df['Date'].str.replace('.', '-')
df['Date'] = pd.to_datetime(df['Date'], format='%d-%m-%Y', errors='coerce')
```

В загруженном из API наборе данных присутствует колонка с кодами драгоценных металлов: 1 – золото, 2 – серебро, 3- платина, 4 – палладий. Такая структура набора данных не удобна для построения графиков. Нам необходимо, чтобы каждая колонка относилась к одному из металлов, и в ней были данные о стоимости металла на момент закрытия торгов. Поэтому в следующем фрагменте программного кода формируется сводная таблица (объект df_pivot):

```
df_pivot = df.pivot(index='Date', columns='Code', values="Sell")
df_pivot.columns = ['Золото', 'Серебро', 'Платина', 'Палладий']
```

Здесь в первой строке формируется сводная таблица, во второй строке колонкам сводной таблицы даются имена. Итоговая сводная таблица df_pivot возвращаем в точку вызова.

Теперь в этом программном модуле можно закомментировать строки вывода данных в окно терминала и использовать его в приложениях Django. Что и сделано в следующем разделе.

> ***Примечание.***
> Представленный в данном разделе программный модуль не включен в список исполняемых примеров интерактивной цифровой книги, так как он может работать только в рамках приложения Python локального компьютера, или быть вызванным из другого модуля веб приложения.

5.6.5. Модули Django для визуализации данных из API Центрального банка (цены на драгметаллы)

Реализуем пример формирования графика динамики цен на драгметаллы на основе данных, полученных из API Центрального банка с использованием интерфейса Plotly Graph Objects. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 5.21).

Листинг 5.21. Ссылка активации маршрута fig_drag_met

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем fig_drag_met (листинг 5.22).

Листинг 5.22. Маршрут URL с именем fig_drag_met

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление fig_drag_met (листинг 5.23).

Листинг 5.23. Представление (view) fig_drag_met

Примечание.
В демо-версии книги листинги программ удалены.

В этом модуле выполнен импорт необходимых библиотек и модуля drag_met_cbr, код которого приведен в предыдущем разделе. Затем сформированы даты начала и конца запрашиваемого периода (d1, d2). После этого выполнено обращение к API Центрального банка для получения данных о ценах на драгметаллы (объект df).

На следующем шаге создан пустой макет графика (объект fig), к этому макету добавлены линии, сформированные на основе загруженных данных:

```
fig.add_trace(go.Scatter(x=df.index, y=df['Золото'], name='Золото'))
fig.add_trace(go.Scatter(x=df.index, y=df['Серебро'], name='Серебро'))
fig.add_trace(go.Scatter(x=df.index, y=df['Платина'], name='Платина'))
fig.add_trace(go.Scatter(x=df.index, y=df['Палладий'], name='Палладий'))
```

Здесь данными для оси x является индексная колонка объекта df, а для оси y - колонки с данными для каждого вида металлы.

После этого график приведен к формату, в котором он может быть показано на HTML странице (объект plot_div) и создана строка с заголовком страницы (title_str). Полученный график (plot_div) с заголовком страницы через контекстную переменную context передается в шаблон fig_plotly.html (листинг 5.24).

Листинг 5.24. Шаблон fig_plotly.html

Примечание.
В демо-версии книги листинги программ удалены.

Это шаблон с контейнером, в котором имеются блоки для вывода заголовка страницы и объекта plot_div. Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 5» и активировать ссылку «Цены на драгметаллы (Центробанк)».

Если выполнить набор вышеприведенных модулей, то получим следующий результат (рис. 5.5).



Рис. 5.5. График в шаблоне Django с данными о ценах на драгметаллы, загруженными из API Центробанка

5.7. Модуль Python для загрузки и очистки данных через API Московской биржи

Информационно-статистический сервер Московской Биржи (ИСС / ISS) функционирует в рамках программного комплекса Интернет представительства Московской Биржи и служит для предоставления клиентским приложениям данных о торгах.

В рамках интерфейса Московской Биржи доступны следующие типы информации: статические данные о рынках (режимы торгов и их группы, финансовые инструменты и их описание), данные для построения графиков, сделки, котировки, исторические данные, различные метаданные.

Рассмотрим процедуру загрузки и очистки данных через API Московской бирже на примере ежедневных котировок акций Газпрома.

5.7.1. Модуль Python для загрузки и очистки данных из API Московской биржи (цены на драгметаллы)

Создадим программный модуль Python, который обеспечит загрузку данных о котировках акций Газпрома, их очистку и подготовку к выводу в графическом виде. Мы будем использовать данные, получаемые в режиме реального времени в формате JSON с использованием библиотеки request. Исходный набор данных будет содержать несколько колонок с данными в формате, пригодном для графической интерпретации. Однако нам нужно будет данные формата JSON трансформировать в набор данных DadaFrame. Такой набор данных будет удобен для графической визуализации временного ряда. Программный код этого модуля приведен в листинге 5.25.

Листинг 5.25. Модуль drag_met_cbr.py для загрузки данных из API Центрального банка

Примечание.
В демо-версии книги листинги программ удалены.

В этом программном модуле выполнен импорт необходимых библиотек и создан класс с именем Data_moex, который имеет следующие параметры (свойства):

- firma – идентификатор (тикер) фирмы;
- start – дата начала периода;
- end– дата конца периода;

В теле данной функции выполняется формирование URL адреса запроса к API Московской биржи (объект url) и загрузка данных формата JSON в объект j. Это по сути дела строка, которая содержит вложенный словарь с данными формата Json. Следующие инструкции делают разбор этой строки с формированием набора данных data.

```
data = [{k: r[i] for i, k in enumerate(j['candles']['columns'])} for r in j['candles']['data']]
data = pd.DataFrame(data)
```

Чтобы порядок работы этого программного кода был более понятен, покажем структуру полученных данных. Итак, после запуска этого программного модуля в окно терминала будут выведены следующие данные.

```
    open   close   high  ...    volume              begin                end
0  159.73  161.44  161.58  ...  8117860  2024-01-03 00:00:00  2024-01-03 23:59:59
1  161.51  161.25  161.64  ...  4888930  2024-01-04 00:00:00  2024-01-04 23:59:59
2  161.20  161.94  162.30  ...  8784940  2024-01-05 00:00:00  2024-01-05 23:59:59
3  162.44  163.05  163.20  ...  11173290  2024-01-08 00:00:00  2024-01-08 23:59:59
4  163.01  162.41  163.06  ...  9608200  2024-01-09 00:00:00  2024-01-09 23:59:59

[5 rows x 8 columns]
                          open    close  ...    volume              begin
end                         ...
2024-01-03 23:59:59  159.73  161.44  ...   8117860  2024-01-03 00:00:00
2024-01-04 23:59:59  161.51  161.25  ...   4888930  2024-01-04 00:00:00
2024-01-05 23:59:59  161.20  161.94  ...   8784940  2024-01-05 00:00:00
```

```
2024-01-08 23:59:59  162.44  163.05  ...  11173290  2024-01-08 00:00:00
2024-01-09 23:59:59  163.01  162.41  ...   9608200  2024-01-09 00:00:00
```

Здесь первый набор содержит данные, полученные через API, второй набор содержит очищенные данные. Из полученных результатов видно, что все даты имеют формат datatime, а числовые данные представлены числами с дробной частью. Поэтому приводить данные к нужному формату нет необходимости. Набор данных содержит следующие колонки:

- open – цена акций на момент открытия торгов;
- close - цена акций на момент закрытия торгов;
- high – максимальная цена акций в течение дня торгов;
- volume – количество проданных акций;
- begin – дата, время начала торгов;
- end - дата, время окончания торгов.

В полученном наборе данных индексной колонкой являются целые числа. Так как нам необходимо сформировать временной ряд, то одну из колонок с датами нужно сделать индексной. Здесь в индексную колонку преобразована дата - время закрытия торгов:

```
data = data.set_index('end')
```

Итоговый набор данных data возвращаем в точку вызова.

Теперь в этом программном модуле можно закомментировать строки вывода данных в окно терминала и использовать его в приложениях Django. Что и сделано в следующем разделе.

> ***Примечание.***
> Представленный в данном разделе программный модуль не включен в список исполняемых примеров интерактивной цифровой книги, так как он может работать только в рамках приложения Python локального компьютера, или быть вызванным из другого модуля веб приложения.

5.7.2. Модули Django для визуализации данных из API Московской биржи (цены на акции Газпрома)

Реализуем пример формирования графика динамики цен на акции Газпрома на основе данных, полученных из API Московской биржи с использованием интерфейса Plotly Graph Objects. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 5.26).

Листинг 5.26. Ссылка активации маршрута fig_moex

> ***Примечание.***
> В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем fig_moex (листинг 5.27).

Листинг 5.27. Маршрут URL с именем fig_moex

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление fig_moex (листинг 5.28).

Листинг 5.28. Представление (view) fig_moex

Примечание.
В демо-версии книги листинги программ удалены.

В этом модуле выполнен импорт необходимых библиотек и модуля data_moex, код которого приведен в предыдущем разделе. Затекм создан идентификатор (тикер) фирмы (объект firma), сформированы даты начала и конца запрашиваемого периода (d1, d2). После этого выполнено обращение к API Московской биржи для получения данных о ценах на акции Газпрома (объект df).
На следующем шаге создан пустой макет графика (объект fig), к этому макету добавлена линия, сформированная на основе загруженных данных:

```
fig.add_trace(go.Scatter(x=df.index, y=df['close'], name='Газпром'))
```

Здесь данными для оси x является индексная колонка объекта df, а для оси y - колонки с ценой акции на момент закрытия торгов.

После этого график приведен к формату, в котором он может быть показано на HTML странице (объект plot_div) и создана строка с заголовком страницы (title_str). Полученный график (plot_div) с заголовком страницы через контекстную переменную context передается в шаблон fig_plotly.html (листинг 5.29).

Листинг 5.29. Шаблон fig_plotly.html

Примечание.
В демо-версии книги листинги программ удалены.

Это шаблон с контейнером, в котором имеются блоки для вывода заголовка страницы и объекта plot_div. Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 5» и активировать ссылку «Цены акций Газпрома (Мос. биржа)».

Если выполнить набор вышеприведенных модулей, то получим следующий результат (рис. 5.6).



Рис. 5.6. График в шаблоне Django с данными о ценах на акции Газпрома, загруженными из API Московской биржи

Обратите внимание, что к данному графику добавлен слайдер, который позволяет изменять интервал дат. С использованием этого слайдера можно изменить временной интервал и посмотреть более детальную информацию о ценах (рис. 5.7).



Рис. 5.7. Изменение интервала дат для более детального отображения информации о ценах

Для запроса к данным Московской биржи нужно знать идентификатор (тикер) компании. Перечень некоторых тикеров компаний Московской биржи приведен приложении 3.

5.8. Модуль Python для загрузки и очистки данных через API Всемирного банка

В этом примере будет использована библиотека wbdata. Это простой и удобный интерфейс Python для загрузки информации из различных баз данных Всемирного банка, как в виде словаря, содержащего полные метаданные, так и в виде DataFrame. В настоящее время wbdata охватывает большую часть API Всемирного банка, а также добавляет некоторые удобные функции для поиска и извлечения информации. Кроме того, для загрузки данных будет использован библиотека pandas_datareader

5.8.1. Модуль Python для загрузки и очистки данных из API Всемирного банка с библиотекой wbdata

Создадим программный модуль Python, который обеспечит загрузку данных о внутреннем валовом продукте (ВВП) различных стран, их очистку и подготовку к выводу в графическом виде. Мы будем использовать данные, получаемые в режиме реального времени. Исходный набор данных будет содержать несколько колонок с данными в формате Pandas DataFrame, однако для их графической интерпретации потребуется формирование сводной таблицы. Программный код этого модуля приведен в листинге 5.30.

Листинг 5.30. Модуль wb_pivot.py для загрузки данных из API Всемирного банка

Примечание.
В демо-версии книги листинги программ удалены.

В этом программном модуле выполнен импорт необходимых библиотек и создана функция с именем def get_data(), которая принимает следующие параметры:

- indicator – индикатор (экономический показатель);
- countries – список стран;
- s_date – дата начала периода;
- e_date - – дата конца периода.

В теле данной функции выполняется извлечение данных из API Всемирного банка и загрузка данных в объект dfr. Чтобы порядок работы этого программного кода был более понятен, покажем структуру полученных данных. Итак, после запуска этого программного модуля в окно терминала будут выведены следующие данные.

```
                    GDP (current US$)
country      date
China        2024-01-01           NaN
             2023-01-01  1.779478e+13
             2022-01-01  1.788178e+13
             2021-01-01  1.782046e+13
             2020-01-01  1.468774e+13
```

```
                2019-01-01       1.427997e+13
United Kingdom  2024-01-01                NaN
                2023-01-01       3.380855e+12
                2022-01-01       3.114042e+12
                2021-01-01       3.143323e+12
                2020-01-01       2.696778e+12
                2019-01-01       2.851407e+12
Russian Federation 2024-01-01            NaN
                2023-01-01       2.021421e+12
                2022-01-01       2.266029e+12
                2021-01-01       1.843392e+12
                2020-01-01       1.493076e+12
                2019-01-01       1.693115e+12
United States   2024-01-01                NaN
                2023-01-01       2.772071e+13
                2022-01-01       2.600689e+13
                2021-01-01       2.368117e+13
                2020-01-01       2.135410e+13
                2019-01-01       2.153998e+13
country             China  Russian Federation  United Kingdom  United States
date
2019-01-01  1.427997e+13       1.693115e+12   2.851407e+12   2.153998e+13
2020-01-01  1.468774e+13       1.493076e+12   2.696778e+12   2.135410e+13
2021-01-01  1.782046e+13       1.843392e+12   3.143323e+12   2.368117e+13
2022-01-01  1.788178e+13       2.266029e+12   3.114042e+12   2.600689e+13
2023-01-01  1.779478e+13       2.021421e+12   3.380855e+12   2.772071e+13
```

Здесь первый набор содержит данные, полученные через API, второй набор содержит очищенные данные. Из полученных результатов видно, что все загруженные данные имеют приемлемый формат, поэтому преобразование их типов делать не нужно. Данные за 2024 год отсутствуют, поэтому эти строки можно удалить.

В загруженном из API наборе данных присутствует колонка с наименованиями стран. Такая структура набора данных не удобна для построения графиков, Нам необходимо для каждой страны создать отдельную колонку, в которой будут данные о ВВП, относящиеся к этой стране. Поэтому в следующем фрагменте программного кода формируется сводная таблица (объект df_pivot):

```
df_pivot = dfr.pivot(index='date', columns='country', values='GDP (current US$)')
```

Здесь в качестве индекса задается колонка с датами (date), формируются колонки с названиями стран в качестве имен (country), а значениями строк будут данные о ВВП - GDP (current US$). Итоговая сводная таблица df_pivot возвращаем в точку вызова.
Теперь в этом программном модуле можно закомментировать строки вывода данных в окно терминала и использовать его в приложениях Django. Что и сделано в следующем разделе.

> ***Примечание.***
> Представленный в данном разделе программный модуль не включен в список исполняемых примеров интерактивной цифровой книги, так как он может работать только в рамках приложения Python локального компьютера, или быть вызванным из другого модуля веб приложения.

5.8.2. Модули Django для визуализации данных из API Всемирного банка (ВВП стран)

Реализуем пример формирования графика динамики ВВП стран на основе данных, полученных из API Всемирного банка с использованием интерфейса Plotly библиотеки Pandas. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 5.31).

Листинг 5.31. Ссылка активации маршрута fig_vvp_wb

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем fig_vvp_wb (листинг 5.32).

Листинг 5.32. Маршрут URL с именем fig_vvp_wb

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление fig_vvp_wb (листинг 5.33).

Листинг 5.33. Представление (view) fig_vvp_wb

Примечание.
В демо-версии книги листинги программ удалены.

В этом модуле выполнен импорт необходимых библиотек и подключен графический интерфейс plotly библиотека Pandas. Далее создан индикатор (экономический показатель) – ВВП в виде словаря (объект dic_indic), задан список стран (объект list_country), и сформированы даты начала и конца запрашиваемого периода (d1, d2). После этого выполнено обращение к API Всемирного банка для получения данных о показателе ВВП для заданных стран (объект df).

На следующем шаге создан графика (объект fig), к которому добавлено описание макета:

```
fig = df.plot()
fig.update_layout(
    xaxis=dict(title="Годы"),
    yaxis=dict(title="ВВП, трлн. $"),
    title='Динамика ВВП за период: с ' + d1_str + ' по ' + d2_str,
    hoverlabel=dict(font_size=20),  # Размера шрифта для данных
)
```

После этого график приведен к формату, в котором он может быть показано на HTML странице (объект plot_div) и создана строка с заголовком страницы (title_str). Полученный график (plot_div) с заголовком страницы через контекстную переменную context передается в шаблон fig_plotly.html (листинг 5.34).

Листинг 5.34. Шаблон fig_plotly.html

Примечание.
В демо-версии книги листинги программ удалены.

Это шаблон с контейнером, в котором имеются блоки для вывода заголовка страницы и объекта plot_div. Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 5» и активировать ссылку «ВВП стран (Всемирный банк)».

Если выполнить набор вышеприведенных модулей, то получим следующий результат (рис. 5.8).

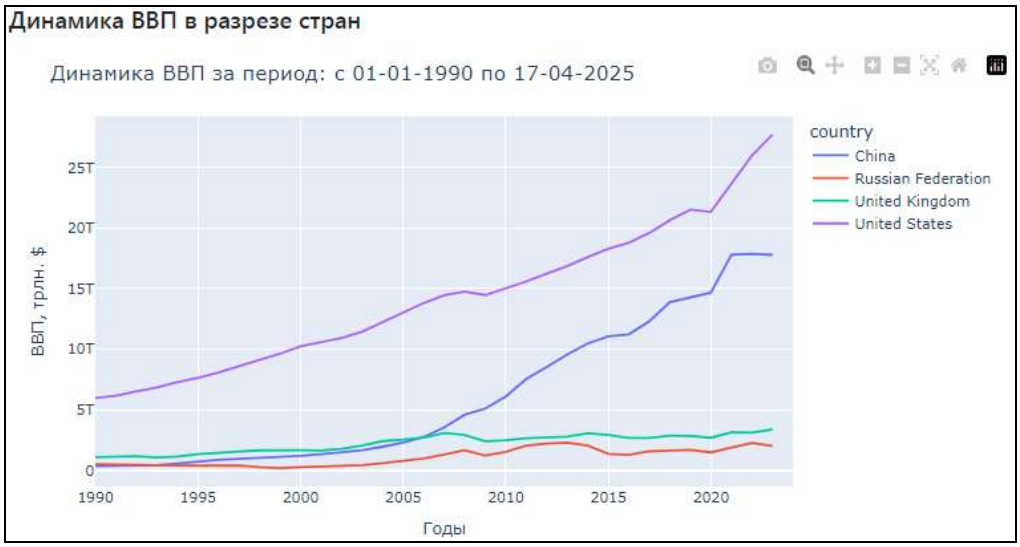


Рис. 5.8. График в шаблоне Django с данными о ВВП стран, загруженными из API Всемирного банка

5.8.3. Модуль Python для загрузки и очистки данных из API Всемирного банка с библиотекой pandas_datareader

В этом примере будет использована библиотека pandas_datareader. Из данной библиотеки будет использован специальный модуль «wb», который обеспечивает загрузку данных через API Всемирного банка в виде DataFrame

Создадим программный модуль Python, который обеспечит загрузку данных об объемах экспорта некоторых стран, их очистку и подготовку к выводу в графическом виде. Мы будем использовать данные, получаемые в режиме реального времени. Исходный набор данных будет содержать несколько колонок с данными в формате Pandas DataFrame, однако для их графической интерпретации необходимо формирование сводной таблицы. Программный код этого модуля приведен в листинге 5.35.

Листинг 5.35. Модуль pand_dr_wb.py для загрузки данных из API Всемирного банка

Примечание.
В демо-версии книги листинги программ удалены.

В этом программном модуле выполнен импорт модуля wb из библиотеки pandas_datareader, и создана функция с именем def get_dr_wb(), которая принимает следующие параметры:

- indicator – индикатор (экономический показатель);
- countries – список стран;
- stars – дата начала периода;
- end – дата конца периода.

В теле данной функции выполняется извлечение данных из индикатора, который является словарем: ключ индикатора (объект ind_key), и значение индикатора (объект val). После этого, на основе полученных параметров, выполняется загрузка данных из API Всемирного банка в объект df. Чтобы порядок работы этого программного кода был более понятен, покажем структуру полученных данных. Итак, после запуска этого программного модуля в окно терминала будут выведены следующие данные.

```
                NE.EXP.GNFS.ZS
country      year
China        2023      19.743072
             2022      20.791483
             2021      19.943974
             2020      18.586139
             2019      18.409992
             2018      19.112104
             2017      19.692277
             2016      19.584380
             2015      21.354080
United Kingdom 2023    31.662677
             2022      33.567194
             2021      29.180275
             2020      29.669796
             2019      31.634780
             2018      31.580111
             2017      30.890351
             2016      28.902340
             2015      27.845951
Russian Federation 2023 23.083003
             2022      28.025719
             2021      29.771214
```

```
                2020      25.522186
                2019      28.433431
                2018      30.793257
                2017      26.090881
                2016      25.854483
                2015      28.703389
United States   2023      11.011511
                2022      11.602228
                2021      10.790978
                2020      10.073656
                2019      11.789197
                2018      12.287111
                2017      12.177481
                2016      11.888159
                2015      12.411149
country         China  Russian Federation  United Kingdom  United States
year
2015    21.354080       28.703389       27.845951       12.411149
2016    19.584380       25.854483       28.902340       11.888159
2017    19.692277       26.090881       30.890351       12.177481
2018    19.112104       30.793257       31.580111       12.287111
2019    18.409992       28.433431       31.634780       11.789197
Exports
```

Здесь первый набор содержит данные, полученные через API, второй набор содержит очищенные данные. Из полученных результатов видно, что все загруженные данные имеют приемлемый формат, поэтому преобразование их типов делать не нужно.

В загруженном из API наборе данных присутствует колонка с наименованиями стран. Такая структура набора данных не удобна для построения графиков. Необходимо для каждой страны создать отдельную колонку, в которой будут данные об экспорте, относящиеся к этой стране. Поэтому в следующем фрагменте программного кода сбрасывается индекс и формируется сводная таблица (объект df_pivot):

```
dfr = dat.reset_index()
df_pivot = dfr.pivot(index='year', columns='country', values=ind_key)
```

Здесь в качестве индекса задается колонка с датами (year), формируются колонки с названиями стран в качестве имен (country), а значениями строк будут данные об экспорте (ind_key). Итоговая сводная таблица df_pivot возвращаем в точку вызова.

Теперь в этом программном модуле можно закомментировать строки вывода данных в окно терминала и использовать его в приложениях Django. Что и сделано в следующем разделе.

Примечание.

Представленный в данном разделе программный модуль не включен в список исполняемых примеров интерактивной цифровой книги, так как он может работать только в рамках приложения Python локального компьютера, или быть вызванным из другого модуля веб приложения.

5.8.4. Модули Django для визуализации данных из API Всемирного банка (данные об экспорте)

Реализуем пример формирования графика динамики экспорта стран на основе данных, полученных из API Всемирного банка с использованием интерфейса Plotly библиотеки Pandas. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 5.36).

Листинг 5.36. Ссылка активации маршрута fig_exp_wb

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем fig_exp_wb (листинг 5.37).

Листинг 5.37. Маршрут URL с именем fig_exp_wb

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление fig_exp_wb (листинг 5.38).

Листинг 5.38. Представление (view) fig_exp_wb

Примечание.
В демо-версии книги листинги программ удалены.

В этом модуле выполнен импорт необходимых библиотек и модуля pand_dr_wb, код которого описан в предыдущем разделе. Затем подключен интерфейс plotly библиотеки Pandas. Далее создан индикатор (экономический показатель) – экспорт в виде словаря (объект dic_indic), задан список стран (объект list_country), и сформированы даты начала и конца запрашиваемого периода (d1, d2). После этого выполнено обращение к API Всемирного банка для получения данных об экспорте для заданных стран (объект df).

На следующем шаге создан графика (объект fig), к которому добавлено описание макета:

```
fig = df.plot()
fig.update_layout(
    xaxis=dict(title="Годы"),
    yaxis=dict(title="Экспорт, трлн. $"),
    title='Динамика экспорта за период: с ' + str(d1) + ' по ' + str(d2),
    hoverlabel=dict(font_size=20),  # Размера шрифта для данных
)
```

После этого график приведен к формату, в котором он может быть показано на HTML странице (объект plot_div) и создана строка с заголовком страницы (title_str). Полученный график (plot_div) с заголовком страницы через контекстную переменную context передается в шаблон fig_plotly.html (листинг 5.39).

Листинг 5.39. Шаблон fig_plotly.html

Примечание.
В демо-версии книги листинги программ удалены.

Это шаблон с контейнером, в котором имеются блоки для вывода заголовка страницы и объекта plot_div. Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 5» и активировать ссылку «Экспорт стран (Всемирный банк)».

Если выполнить набор вышеприведенных модулей, то получим следующий результат (рис. 5.9).

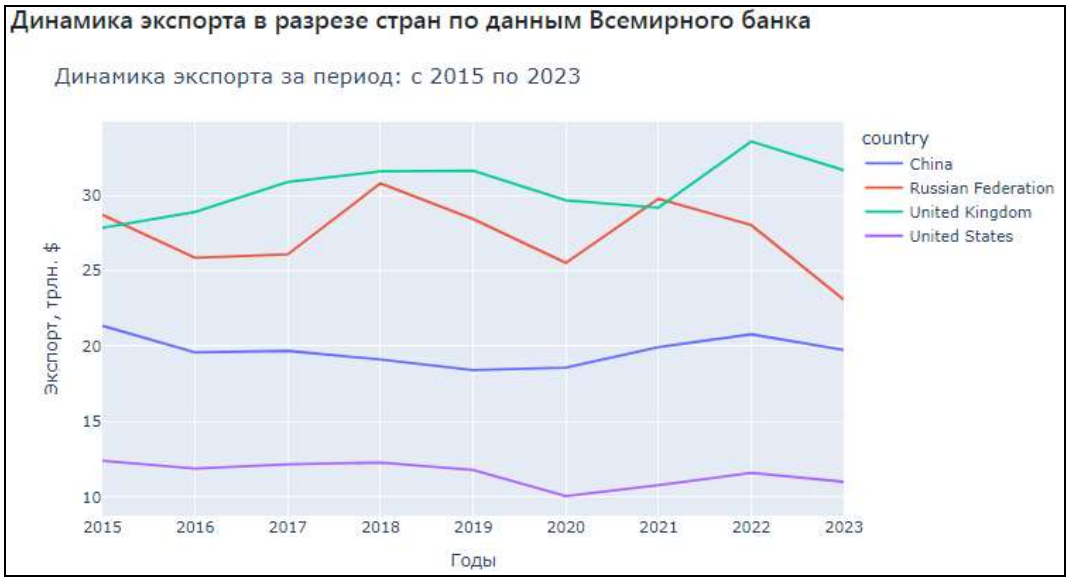


Рис. 5.9. График в шаблоне Django с данными об экспорте стран, загруженными из API Всемирного банка

5.9. Модуль Python для загрузки и очистки данных из API Stooq

В этом примере будет использована библиотека pandas_datareader, а данные о котировке акций будут получены из API финансовой платформы Stooq.

5.9.1. Модуль Python для загрузки и очистки данных из API Stooq (котировки акций) с библиотекой pandas_datarider

Создадим программный модуль Python, который обеспечит загрузку данных о стоимости акций некоторых фирм, их очистку и подготовку к выводу в графическом виде. Мы будем использовать данные, получаемые в режиме реального времени. Исходный набор данных будет содержать несколько колонок с данными в допустимом формате, однако для их графической интерпретации необходимо выполнить некоторые преобразования. Программный код этого модуля приведен в листинге 5.40.

Листинг 5.40. Модуль pand_dr_data.py для загрузки данных из API платформы Stooq

Примечание.
В демо-версии книги листинги программ удалены.

В этом программном модуле выполнен импорт необходимых библиотек, и создана функция с именем def get_dr_data(), которая принимает следующие параметры:

- firm – список фирм (тикеров);
- s – источник данных;
- start – дата начала периода;
- end - stars – дата конца периода;

В теле данной функции выполняется загрузка данных из API источника данных в объект data. Чтобы порядок работы этого программного кода был более понятен, покажем структуру полученных данных. Итак, после запуска этого программного модуля в окно терминала будут выведены следующие данные.

```
Attributes   Close           High  ...     Open     Volume
Symbols       MSFT    AAPL    MSFT  ...     AAPL       MSFT        AAPL
Date                          ...
2025-04-17  367.78  196.98  374.3215  ...  197.200  21120192.0   52164675.0
2025-04-16  371.61  194.27  381.6100  ...  198.360  21967826.0   59732423.0
2025-04-15  385.73  202.14  391.8900  ...  201.855  17199890.0   51343872.0
2025-04-14  387.81  202.52  394.6500  ...  211.440  19251172.0  101352911.0
2025-04-11  388.45  198.15  390.0500  ...  186.100  23839220.0   87435915.0

[5 rows x 10 columns]
Symbols       MSFT    AAPL
Date
2025-04-17  367.78  196.98
2025-04-16  371.61  194.27
2025-04-15  385.73  202.14
2025-04-14  387.81  202.52
2025-04-11  388.45  198.15
```

Здесь первый набор содержит данные, полученные через API, второй набор содержит очищенные данные. Из полученных результатов видно, что все загруженные данные имеют приемлемый формат, поэтому преобразование их типов делать не нужно.

Набор данных, загруженный из API, имеет сложную структуру заголовка, состоящую из двух строк.

19. Строка с наименованием параметров: стоимость акций на момент закрытия торгов (Close), максимальная стоимость акций (High), стоимость акций на момент открытия торгов (Open), объем продаж (Volume).
20. Строка с наименованиями (тикерами) фирм.

Такая структура набора данных не удобна для построения графиков. Нам необходимо для каждой фирмы создать отдельную колонку, в которой будут данные о стоимости акций, например, на момент закрытия торгов. Поэтому в следующем фрагменте программного кода формируется набор данных, в котором для каждой фирмы останется только одна колонка со стоимостью акций на момент закрытия торгов:

```
df = data['Close']
```

Итоговый набор данных df возвращаем в точку вызова.

Теперь в этом программном модуле можно закомментировать строки вывода данных в окно терминала и использовать его в приложениях Django. Что и сделано в следующем разделе.

> ***Примечание.***
> Представленный в данном разделе программный модуль не включен в список исполняемых примеров интерактивной цифровой книги, так как он может работать только в рамках приложения Python локального компьютера, или быть вызванным из другого модуля веб приложения.

5.9.2. Модули Django для визуализации данных из API Stooq (котировка акций)

Реализуем пример формирования графика динамики стоимости акций нескольких фирм на основе данных, полученных из API Stooq с использованием интерфейса Plotly библиотеки Pandas. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 5.41).

Листинг 5.41. Ссылка активации маршрута fig_stooq

> ***Примечание.***
> В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем fig_stooq (листинг 5.42).

Листинг 5.42. Маршрут URL с именем fig_stooq

> ***Примечание.***
> В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление fig_stooq (листинг 5.43).

Листинг 5.43. Представление (view) fig_stooq

Примечание.
В демо-версии книги листинги программ удалены.

В этом модуле выполнен импорт необходимых библиотек и модуля pand_dr_data, код которого описан в предыдущем разделе. Затем подключен интерфейс plotly библиотеки Pandas. Далее создан список фирм (объект list_firm), задан источник данных (объект sours), и сформированы даты начала и конца запрашиваемого периода (d1_str, d2_str). После этого выполнено обращение к API Stooq для получения данных о котировки акций (объект df).

На следующем шаге создан графика (объект fig), к которому добавлено описание макета:

```
fig.update_layout(
      xaxis=dict(title="Даты"),
      yaxis=dict(title="Стоимость акций, $"),
      title='Изменение стоимости акций за период: с ' + d1_str + ' по ' + d2_str,
      hoverlabel=dict(font_size=20),  # Размера шрифта для данных
   )
```

После этого график приведен к формату, в котором он может быть показано на HTML странице (объект plot_div) и создана строка с заголовком страницы (title_str). Полученный график (plot_div) с заголовком страницы через контекстную переменную context передается в шаблон fig_plotly.html (листинг 5.44).

Листинг 5.44. Шаблон fig_plotly.html

Примечание.
В демо-версии книги листинги программ удалены.

Это шаблон с контейнером, в котором имеются блоки для вывода заголовка страницы и объекта plot_div. Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 5» и активировать ссылку «Стоимость акций от Stoq».

Если выполнить набор вышеприведенных модулей, то получим следующий результат (рис. 5.10).



Рис. 5.10. График в шаблоне Django с данными о стоимости акций, загруженными из API Stooq

5.10. Модуль Python для загрузки и очистки данных из API Yahoo Finance

В этом разделе будет использована библиотека yfinance, а данные о котировке акций будут получены из API финансовой платформы Yahoo Finance.

5.10.1. Модуль Python для загрузки и очистки данных из API Yahoo Finance (котировка акций)

Создадим программный модуль Python, который обеспечит загрузку данных о стоимости акций некоторых фирм, их очистку и подготовку к выводу в графическом виде. Мы будем использовать данные, получаемые в режиме реального времени. Исходный набор данных будет содержать несколько колонок с данными в допустимом формате, однако для их графической интерпретации необходимо выполнить некоторые преобразования. Программный код этого модуля приведен в листинге 5.45.

Листинг 5.45. Модуль yfin_data.py для загрузки данных из API платформы Yahoo Finance

Примечание.
В демо-версии книги листинги программ удалены.

В этом программном модуле выполнен импорт необходимых библиотек, и создана функция с именем def get_data(), которая принимает следующие параметры:

- firma – список фирм (тикеров);

- start – дата начала периода;
- end - stars – дата конца периода;

В теле данной функции выполняется загрузка данных из API источника данных в объект data. Чтобы порядок работы этого программного кода был более понятен, покажем структуру полученных данных. Итак, после запуска этого программного модуля в окно терминала будут выведены следующие данные.

```
Price           Close                 High  ...       Open    Volume
Ticker           AAPL      GOOGL       AAPL  ...      GOOGL      AAPL     GOOGL
Date                                         ...
2025-01-02  243.263199  188.983292  248.500565  ...  190.200416  55740700  20370800
2025-01-03  242.774368  191.337723  243.592387  ...  190.918715  40244100  18596200
2025-01-06  244.410416  196.405762  246.734810  ...  193.522577  45045600  29563600
2025-01-07  241.627136  195.029022  244.959095  ...  196.645197  40856000  26487200
2025-01-08  242.115936  193.492645  243.123515  ...  192.115910  37628900  24864800

[5 rows x 10 columns]
Ticker           AAPL      GOOGL
Date
2025-01-02  243.263199  188.983292
2025-01-03  242.774368  191.337723
2025-01-06  244.410416  196.405762
2025-01-07  241.627136  195.029022
2025-01-08  242.115936  193.492645
```

Здесь первый набор содержит данные, полученные через API, второй набор содержит очищенные данные. Из полученных результатов видно, что все загруженные данные имеют приемлемый формат, поэтому преобразование их типов делать не нужно.

Набор данных, загруженный из API, имеет сложную структуру заголовка, состоящую из двух строк.

21. Строка с наименованием параметров: стоимость акций на момент закрытия торгов (Close), максимальная стоимость акций (High), стоимость акций на момент открытия торгов (Open), объем продаж (Volume).
22. Строка с наименованиями (тикерами) фирм.

Такая структура набора данных не удобна для построения графиков. Нам необходимо для каждой фирмы создать отдельную колонку, в которой будут данные о стоимости акций, например, на момент закрытия торгов. Поэтому в следующем фрагменте программного кода формируется набор данных, в котором для каждой фирмы останется только одна колонка со стоимостью акций на момент закрытия торгов:

```
data = data['Close']
```

Итоговый набор данных data возвращаем в точку вызова.

Теперь в этом программном модуле можно закомментировать строки вывода данных в окно терминала и использовать его в приложениях Django. Что и сделано в следующем разделе.

> ***Примечание.***
> Представленный в данном разделе программный модуль не включен в список исполняемых примеров интерактивной цифровой книги, так как он может работать только в рамках

приложения Python локального компьютера, или быть вызванным из другого модуля веб приложения.

5.10.2. Модули Django для визуализации данных из API Yahoo Finance (котировка акций)

Реализуем пример формирования графика динамики стоимости акций нескольких фирм на основе данных, полученных из API Yahoo Finance с использованием интерфейса Plotly библиотеки Pandas. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 5.46).

Листинг 5.46. Ссылка активации маршрута fig_yahoo_fin

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем fig_yahoo_fin (листинг 5.47).

Листинг 5.47. Маршрут URL с именем fig_yahoo_fin

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление fig_yahoo_fin (листинг 5.48).

Листинг 5.48. Представление (view) fig_yahoo_fin

Примечание.
В демо-версии книги листинги программ удалены.

В этом модуле выполнен импорт необходимых библиотек и модуля yfin_data, код которого описан в предыдущем разделе. Затем подключен интерфейс plotly библиотека Pandas. Далее создан список фирм (объект list_firm) и сформированы даты начала и конца запрашиваемого периода (d1_str, d2_str). После этого выполнено обращение к API Yahoo Finance для получения данных о котировки акций (объект df).

На следующем шаге создан графика (объект fig), к которому добавлено описание макета:

```
fig.update_layout(
    xaxis=dict(title="Даты"),
    yaxis=dict(title="Стоимость акций, $"),
    title='Изменение стоимости акций за период: с ' + d1_str + ' по ' + d2_str,
    hoverlabel=dict(font_size=20),  # Размера шрифта для данных
)
```

После этого график приведен к формату, в котором он может быть показано на HTML странице (объект plot_div) и создана строка с заголовком страницы (title_str). Полученный

график (plot_div) с заголовком страницы через контекстную переменную context передается в шаблон fig_plotly.html (листинг 5.49).

Листинг 5.49. Шаблон fig_plotly.html

Примечание.
В демо-версии книги листинги программ удалены.

Это шаблон с контейнером, в котором имеются блоки для вывода заголовка страницы и объекта plot_div. Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 5» и активировать ссылку «Стоимость акций от YahooFin».

Если выполнить набор вышеприведенных модулей, то получим следующий результат (рис. 5.11).



Рис. 5.11. График в шаблоне Django с данными о стоимости акций, загруженными из API Yahoo Finance

5.10.3. Модуль Python для загрузки и очистки данных из API Yahoo Finance (стоимость нефти)

В этом примере будет использована библиотека yfinance, а данные о стоимости нефти будут получены из API финансовой платформы Yahoo Finance.

Создадим программный модуль Python, который обеспечит загрузку данных о стоимости нефти, их очистку и подготовку к выводу в графическом виде. Мы будем использовать данные, получаемые в режиме реального времени. Исходный набор данных будет

содержать несколько колонок с данными в допустимом формате, однако для их графической интерпретации необходимо выполнить некоторые преобразования. Программный код этого модуля приведен в листинге 5.50.

Листинг 5.50. Модуль yfin_data_oil.py для загрузки данных из API платформы Yahoo Finance

Примечание.
В демо-версии книги листинги программ удалены.

В этом программном модуле выполнен импорт необходимых библиотек, и создана функция с именем def get_data(), которая принимает следующие параметры:

- tic – тикер (обозначение торгового актива, в данном случае это нефть);
- p – период.

Здесь период задается в виде текстовой строки (в этом примере строка «5y» означает 5 лет назад от текущей даты). В теле данной функции выполняется загрузка данных из API источника данных в объект df. Чтобы порядок работы этого программного кода был более понятен, покажем структуру полученных данных. Итак, после запуска этого программного модуля в окно терминала будут выведены следующие данные.

```
                           Open       High  ...  Dividends  Stock Splits
Date                                        ...
2020-06-10 00:00:00-04:00  38.500000  39.910000  ...        0.0           0.0
2020-06-11 00:00:00-04:00  39.090000  39.090000  ...        0.0           0.0
2020-06-12 00:00:00-04:00  36.259998  36.990002  ...        0.0           0.0
2020-06-15 00:00:00-04:00  36.029999  37.259998  ...        0.0           0.0
2020-06-16 00:00:00-04:00  37.049999  39.060001  ...        0.0           0.0

[5 rows x 7 columns]
Index(['Open', 'High', 'Low', 'Close', 'Volume', 'Dividends', 'Stock Splits'], dtype='object')
--------------------------
2020-06-10    39.599998
2020-06-11    36.340000
2020-06-12    36.259998
2020-06-15    37.119999
2020-06-16    38.380001
Name: Close, dtype: float64
```

Здесь первый набор содержит данные, полученные через API, затем выведена строка с заголовками столбцов, второй набор содержит очищенные данные. Из данной распечатки видно, что все полученные данные имеют приемлемый формат, правда индексная колонка содержит дату и время, поскольку нам параметр времени не интересен, то его нужно будет удалить.

В наборе данных несколько столбцов: стоимость нефти на момент закрытия торгов (Close), максимальная стоимость нефти (High), стоимость нефти на момент открытия торгов (Open), объем продаж (Volume). Нам для построения графика понадобится только один столбец – Close.

В следующем фрагменте программного кода из индексного столбца убирается время:

```
df.index = df.index.date
```

Затем из набора данных выбираем только один столбец (Close):

```
df = df['Close']
```

Итоговый набор данных df возвращаем в точку вызова.

Теперь в этом программном модуле можно закомментировать строки вывода данных в окно терминала и использовать его в приложениях Django. Что и сделано в следующем разделе.

> ***Примечание.***
> Представленный в данном разделе программный модуль не включен в список исполняемых примеров интерактивной цифровой книги, так как он может работать только в рамках приложения Python локального компьютера, или быть вызванным из другого модуля веб приложения.

5.10.4. Модули Django для визуализации данных из API Yahoo Finance (стоимость нефти)

Реализуем пример формирования графика динамики стоимости нефти на основе данных, полученных из API Yahoo Finance с использованием интерфейса Plotly библиотеки Pandas. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 5.51).

Листинг 5.51. Ссылка активации маршрута fig_yahoo_fin_oil

> ***Примечание.***
> В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем fig_yahoo_fin_oil (листинг 5.52).

Листинг 5.52. Маршрут URL с именем fig_yahoo_fin_oil

> ***Примечание.***
> В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление fig_yahoo_fin_oil (листинг 5.53).

Листинг 5.53. Представление (view) fig_yahoo_fin_oil

> ***Примечание.***
> В демо-версии книги листинги программ удалены.

В этом модуле выполнен импорт необходимых библиотек и модуля yfin_data_oil, код которого описан в предыдущем разделе. Затем подключен интерфейс plotly библиотеки Pandas. Далее создан тикер (объект ticker - обозначение торгового актива, в данном случае это нефть) и сформирован период (объект period). Здесь период задается в виде текстовой

строки (в этом примере строка «5y» означает 5 лет назад от текущей даты). После этого выполнено обращение к API Yahoo Finance для получения данных о стоимости нефти (объект df).

На следующем шаге создан графика (объект fig) с описанием макета:

```
fig = df.plot(title="Динамики стоимости нефти за последние 5 лет",
        labels=dict(index="Дата",
                value="Стоимость нефти",
                variable="Сырая нефть"))
```

После этого график приведен к формату, в котором он может быть показано на HTML странице (объект plot_div) и создана строка с заголовком страницы (title_str). Полученный график (plot_div) с заголовком страницы через контекстную переменную context передается в шаблон fig_plotly.html (листинг 5.54).

Листинг 5.54. Шаблон fig_plotly.html

Примечание.
В демо-версии книги листинги программ удалены.

Это шаблон с контейнером, в котором имеются блоки для вывода заголовка страницы и объекта plot_div. Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 5» и активировать ссылку «Стоимость нефти от YahooFin».

Если выполнить набор вышеприведенных модулей, то получим следующий результат (рис. 5.12).



Рис. 5.12. График в шаблоне Django с данными о стоимости акций, загруженными из API Yahoo Finance

5.11. Краткие итоги

Итак, мы познакомились с некоторыми приемами загрузки данных из различных источников и подготовки их к визуализации. Так как API разных источников данных поставляет информацию в разных видах, то ее, перед визуализацией, необходимо проанализировать и привести к удобному формату. Конечно, здесь рассмотрен далеко не полный перечень источников данных, однако последовательность их подготовки к графическому представлению будет идентична.

Особый интерес представляют данные о финансово экономической деятельности разных стран, поставляемых через API международного валютного фонда (МВФ). Так как через API МВФ можно получить доступ к сотням баз данных, то материалы по загрузке и очистки данных через API МВФ выделены в отдельную главу.

Глава 6. Получение и обработка данных международного валютного фонда

Международный валютный фонд МВФ (International Monetary Fund - IMF) - одна из крупнейших международных финансовых организаций. Его штаб-квартира находится в Вашингтоне. Фонд создан по соглашению стран для консультаций, поддержки и финансирования государств и играет важную роль в мировой экономике. Он был создан в 1944 году для обеспечения стабильности международной валютной системы — системы обменных курсов и международных платежей, которая позволяет странам совершать сделки друг с другом.

Однако МВФ это не только финансовая организация, фонд имеет множество баз данных, содержащих финансово экономическими показателями разных стран, Вот далеко не полный перечень баз данных МВФ.

1. **World Economic Outlook (WEO)**. Обзор мировой экономики. База данных создается дважды в год и включает публикации в апреле и сентябре/октябре.
2. **Fiscal Monitor (FM)**. База данных содержит данные по мониторингу и анализу изменений в государственных финансах.
3. **Consumer Price Index (CPI)**. База данных включает национальные индексы потребительских цен по экономике.
4. **International Financial Statistics (IFS)**. База данных содержит глобальные данные об обменных курсах, международной ликвидности, деньгах и банковском деле, процентных ставках, ценах, производстве, международных транзакциях и национальных счетах.
5. **Coordinated Direct Investment Survey (CDIS)**. В этой базе содержаться ежегодные данные о запасах и потоках прямых иностранных инвестиций по всему миру.
6. **Financial Data Query tool (IMF)**. База данных обеспечивает досиуп к финансовым данным по странам с 1984 года, включая квоты членов МВФ, позиции резервных траншей, запасы специальных прав заимствования, непогашенный кредит, ожидаемые платежи и ежемесячные исторические транзакции с МВФ.

Доступ к базам данных МВФ можно получить через API МВФ с использованием языка программирования Python. Этот интерфейс позволяет запрашивать данные экономических показателей, информацию о странах, регионах и аналитических группах. Кроме того для Python существует множество специализированных библиотек, которые упрощают использование API МВФ для конечных разработчиков. В данной главе будут рассмотрены возможности загрузки финансово экономической информации из баз данных МВФ и ее графической визуализации с использованием только некоторых наиболее популярных библиотек. А также будут рассмотрены примеры загрузки данных напрямую через API МВФ с использованием библиотеки requests.

Из материалов этой главы вы узнаете:

- о возможностях библиотеки imfp для загрузки информации о базах данных МВФ;
- как можно загружать данные из БД МВФ с использованием функции imf_dataset;
- как можно обработать данные перед графической визуализацией;
- как можно получить и визуализировать данные о ценах на природные ископаемые;

- как можно получить и визуализировать данные о ценах на драгоценные металлы;
- как можно получать данные из БД международной финансовой статистике;
- как можно получить и обработать данные о движении товаров (экспорт);
- как можно использовать библиотеку requests для получения и обработки данных о движении товаров (импорт);
- как можно использовать библиотеку requests для получения и обработки данных о статичтики международной торговли);
- как можно загрузить ключевые показатели API МВФ с библиотекой requests;
- как загружать данные через API МВФ с использованием параметров запросов.

6.1.Библиотека imfp

Библиотека imfp позволяет загружать данные из API МВФ в формате DataFrame библиотеки pandas. Поэтому можно использовать все возможности pandas для анализа и обработки полученных данных. Для визуализации данных можно использовать библиотеки matplotlib, seaborn или plotly. Библиотека impf имеет одно ограничение, она работает с Python версии 3.10 и выше. Установить последнюю версию данной библиотеки можно с использованием следующей команды:

```
pip install --upgrade imfp
```

Для обработки данных понадобится библиотека pandas, а для формирования графиков библиотека plotly. Кроме того, для сохранения больших наборов данных в формате Excel потребуется библиотека openpyxl. Установить эти библиотеки можно с использованием следующих команд:

```
pip install pandas
pip install plotly
pip install openpyxl
```

При выполнении запросов к базам данных МВФ желательно данной библиотеке передать имя приложения, из которого делается запрос. Для этого можно использовать метод imfp.imf_app_name(). Данный метод позволяет пользователям устанавливать пользовательское имя приложения, которое будет использоваться при выполнении вызовов к API IMF. Дело в том, что API IMF имеет ограничение скорости загрузки данных для приложения (50 запросов в секунду), при этом приложение идентифицируется переменной "user_agent" в заголовке запроса. Такое ограничение может вызвать задержку загрузки данных, если большое количество пользователей будут делать одновременные запросы к API, используя имя приложения по умолчанию. Установив пользовательское имя приложения, пользователи могут избежать ограничений скорости и блокировки API. Метод imfp.imf_app_name() устанавливает имя приложения, изменяя IMF_APP_NAME переменной в среде. Если эта переменная не существует, то imfp.imf_app_name() создаст ее. Для того чтобы задать пользовательское имя приложения, просто вызовите функцию imfp.imf_app_name(), указав желаемое имя приложения в качестве аргумента, например:

```
imfp.imf_app_name("my_custom_app_name")
```

Функция выдаст ошибку, если указанное имя отсутствует (NULL, NA), не является строкой или длиннее 255 символов. Если указанное имя — «imfr» (по умолчанию) или пустая

строка, то функция выдаст предупреждение, рекомендующее использовать уникальное имя приложения, чтобы избежать превышения ограничений по скорости.

Библиотека imfp представляет четыре основные функции:

- imfp.imf_databases - получение списка баз данных;
- imfp.imf_parameters — получение списка параметров запроса;
- imfp.imf_parameter_defs — получение описания параметров запросов
- imfp.imf_dataset — загрузка наборов данных.

Рассмотрим использование этих функция на простых примерах

6.2.Получение сведений о базах данных МВФ с функцией imf_databases

Для загрузки наборов данных нужно использовать функцию imfp.imf_dataset, но для этого ей нужно передать набор дополнительных параметров. Например, при вызове функции imfp.imf_dataset нужно будет обязательно указать идентификатор базы данных (database_id), из которой нужно загрузить данные. Это связано с тем, что МВФ обслуживает множество различных баз данных через API, и в запросе нужно указать, из какой базы данных будет запрошена информация. Чтобы получить список доступных баз данных, нужно использовать функцию imfp.imf_databases. Пример использования данной функции приведен в модуле **load_db_id.py** (листинге 6.1).

Листинг 6.1. Модуль load_db_id.py для получения доступных баз данных МВФ

> ***Примечание.***
> В демо-версии книги листинги программ удалены.

В этом модуле импортирована библиотека imfp, а затем выполнены следующие действия:

- в объект df загружен набор данных с информацией обо всех доступных БД МВФ;
- в объект db загружены строки с пятой по девятую набора данных df;
- в объект db_com загружены строки набора данных df, в описании которых присутствует слово Commodity (товар).

В последних строках набор данных df сохранен в файлах csv и xlsx. Все наборы данных выведены в окно терминала.

Если выполнить данный программный код, то будут получены следующие результаты:

```
---- Полный список БД МВФ ------------------
      database_id        description
0     BOP_2017M06   Balance of Payments (BOP), 2017 M06
1     BOP_2020M3    Balance of Payments (BOP), 2020 M03
2     BOP_2017M11   Balance of Payments (BOP), 2017 M11
3     DOT_2020Q1    Direction of Trade Statistics (DOTS), 2020 Q1
4     GFSMAB2016    Government Finance Statistics Yearbook (GFSY 2...
..       ...                          ...
318   PCTOT         Commodity Terms of Trade
```

```
319     RAFIT3P          RA-FIT Round3 Completion and Participation Rates
320     FSI              Financial Soundness Indicators (FSIs)
321     FSIRE            Financial Soundness Indicators: Reporting enti...
322     PCPS             Primary Commodity Price System (PCPS)
[323 rows x 2 columns]

---- Строки с 5-й по 9-ю списка БД МВФ ---------
      database_id                        description
5     BOP_2019M12      Balance of Payments (BOP), 2019 M12
6     GFSYFALCS2014  Government Finance Statistics Yearbook (GFSY 2...
7     GFSE2016         Government Finance Statistics Yearbook (GFSY 2...
8     FM201510         Fiscal Monitor (FM) October 2015
9     GFSIBS2016       Government Finance Statistics Yearbook (GFSY 2...

---- База данных с товарами (Commodity) ----------
   database_id          description
318    PCTOT          Commodity Terms of Trade
322    PCPS           Primary Commodity Price System (PCPS)
```

Примечание.

Представленный в данном разделе программный модуль не включен в список исполняемых примеров интерактивной цифровой книги, так как он может работать только в рамках приложения Python локального компьютера, или быть вызванным из другого модуля веб приложения.

Здесь мы видим содержимое трех наборов данных. Для первого набора данных df в окно терминала выведено содержимое первых и последних строк (всего в данном наборе 323 строки). Это идентификаторы всех БД, из которых можно получить данные через API МВФ. В первой колонке (database_id) находятся идентификаторы баз данных, во второй колонке (description) — описание базы данных.

Для второго набора данных db в окно терминала выведено содержимое строк с пятой по девятую с использованием следующей инструкции:

```
db = df.loc[5:9]
```

То есть можно получить идентификаторы баз данных по интервалу номеров строк.

Для третьего набора данных db_com в окно терминала выведены идентификаторы только тех баз данных, в описании которых встречается слово Commodity (товар). Для этого используется следующая инструкция:

```
db_com = df[df['description'].str.contains("Commodity")]
```

Здесь, по сути, на исходный набор данных наложен фильтр, согласно которому из полного набора данных выбираются только те строки, у которых в столбце с именем description встречается слово Commodity.

Таким образом, использованием методов библиотеки Pandas можно получить доступ к идентификаторам любой из 323-х баз данных Международного валютного фонда. Кроме того, мы сохранили полученный набор данных в файле Excel, то есть теперь можно открыть файл data_bases_imf.xlsx и найти запись с идентификатором нужной базы данных.

В табл. 6.1 приведены первые десять записей из списка баз данных МВФ с переводом на русский язык.

Таблица 6.1

Идентификаторы некоторых баз данных API МВФ

№	database_id	Описание
0	BOP_2017M06	Платежный баланс (ПБ), 2017 M06
1	BOP_2020M3	Платежный баланс (ПБ), 2020 M03
2	BOP_2017M11	Платежный баланс (ПБ), 2017 M11
3	DOT_2020Q1	Направление статистики торговли (DOTS), 2020 Q1
4	GFSMAB2016	Ежегодник статистики государственных финансов (GFSY 2...)
5	BOP_2019M12	Платежный баланс (ПБ), 2019 M12
6	GFSYFALCS2014	Ежегодник статистики государственных финансов (GFSY 2...)
7	GFSE2016	Ежегодник статистики государственных финансов (GFSY 2...)
8	FM201510	Финансовый монитор (FM) октябрь 2015 г.
9	GFSIBS2016	Ежегодник статистики государственных финансов (GFSY 2...)

6.3.Получение сведений о параметрах запросов к базам данных МВФ с функцией imf_parameters

Запросы к базам данных, доступным через API МВФ, осложняются тем, что каждая база данных использует разный набор параметров при выполнении запроса. Кроме того, необходимо иметь список допустимых входных кодов для каждого параметра. Функция imfp.imf_parameters решает эту проблему. Используйте функцию, чтобы получить полный список параметров и допустимых входных кодов для заданной базы данных. Пример использования данной функции приведен в модуле **load_db_param.py** (листинге 6.2).

Листинг 6.2. Модуль load_db_param.py для получения доступных параметров баз данных МВФ

Примечание.
В демо-версии книги листинги программ удалены.

В этом модуле импортирована библиотека imfp, а затем выполнены следующие действия:

- в объект params загружен набор данных с информацией обо всех доступных параметрах конкретной базы данных (PCPS);
- в объект keys загружены имена ключей из набора данных params;

В заключение колонка commodity (товары) набора params сохранена в файле params_imf.xlsx. Все наборы данных выведены в окно терминала.

Если выполнить данный программный код, то будут получены следующие результаты:

```
---------- Параметры ----------
{'freq':
      input_code description
0         A          Annual
1         M         Monthly
2         Q        Quarterly,
```

```
'ref_area':
        input_code                  description
0       W00             All Countries, excluding the IO,
```

```
'commodity':
            input_code                              description
0           PRAWM             Primary Commodity Prices, Agr. Raw Material In...
1           PAGRI             Primary Commodity Prices, Agriculture
2           PALLFNF           Primary Commodity Prices, All index
3           PEXGMETA          Primary Commodity Prices, All Metals EX GOLD I...
4           PALLMETA          Primary Commodity Prices, All Metals Index
..               ...                                     ...
106         PWOOLC           Primary Commodity Prices, Wool, Coarse
107         PWOOLF            Primary Commodity Prices, Wool, Fine
108         POILWTI           Primary Commodity Prices, WTI Crude
109         PZINC             Primary Commodity Prices, Zinc
110         All_Indicators    All Indicators
[111 rows x 2 columns],
```

```
'unit_measure':
      input_code          description
0     IX                  Index
1     PC_CP_A_PT          Percent Change, Corresponding Period, Previous...
2     PC_PP_PT            Percent Change, Previous Period, Percent
3     USD                 US Dollars
4     All_Units           All Units}
-----------------------------------
```

```
------------ Имена ключей ----------
dict_keys(['freq', 'ref_area', 'commodity', 'unit_measure'])
```

> ***Примечание.***
> Представленный в данном разделе программный модуль не включен в список исполняемых примеров интерактивной цифровой книги, так как он может работать только в рамках приложения Python локального компьютера, или быть вызванным из другого модуля веб приложения.

В этом программном коде функция imf_parameters возвращает словарь наборов данных DataFrame и имена ключей к этому словарю. Каждое имя ключа словаря соответствует параметру, используемому при выполнении запросов к базе данных. В данном случае были получены следующие имена ключей:

- freq - частота (относительно дат);
- ref_area — географическая область (или страна);
- commodity — индикатор (или товар);
- unit_measure — единица измерения.

Первая группа параметров (freq) позволяет задать частоту запрашиваемых данных при обращении к API. Здесь используются следующие коды: «A» — год, «M» - месяц, «Q» — квартал.

Вторая группа параметров (ref_area) позволяет задать географический регион (или страну) при обращении к API. Здесь используются всего один код: «W00» — все страны.

Третья группа параметров (commodity) позволяет задать индикатор (или товар) при обращении к API. Здесь для данной конкретной базы выведено 111 кодов индикаторов. Эти коды вместе с их описанием сохранены в файле с именем params_imf.xlsx.

Четвертая группа параметров (unit_measure) позволяет задать единицу измерения при обращении к API. Здесь для определения единицы измерения используется 5 кодов.

6.4.Получение сведений об описании параметров запросов к базам данных МВФ с функцией imf_parameter_defs

В тех случая, когда необходимо получить краткое текстовое описание каждого параметра можно использовать функцию imfp.imf_parameter_defs. Пример использования данной функции приведен в модуле **load_db_param_defs.py** (листинге 6.3).

Листинг 6.3. Модуль load_db_param_defs.py для получения описания доступных параметров баз данных МВФ

Примечание.
В демо-версии книги листинги программ удалены.

В этом модуле импортирована библиотека imfp, а затем выполнены следующие действия:
- в объект params загружен набор данных с информацией обо всех доступных параметрах конкретной базы данных (PCPS);
- в объект par загружено описание ключевых параметров базы данных (PCPS);
- в объект frg загружено описание параметра, характеризующего частоту загружаемых данных (с точки зрения периода);
- в объект ref загружено описание параметра, характеризующего географическую область;
- в объект unit загружено описание параметра, характеризующего единицы измерения;
- в объект comm загружено описание параметра, характеризующего товары.

Примечание.
Представленный в данном разделе программный модуль не включен в список исполняемых примеров интерактивной цифровой книги, так как он может работать только в рамках

приложения Python локального компьютера, или быть вызванным из другого модуля веб приложения.

Если выполнить данный программный код, то в окно терминала будут выведены следующие результаты:

```
-------- Описание параметров базы PCPS ----------------
     parameter        description
0  freq               Frequency
1  ref_area           Geographical Areas
2  commodity          Indicator
3  unit_measure       Unit
--------------------------
```

```
--- Параметры частоты ----------------
   input_code   description
0          A         Annual
1          M        Monthly
2          Q       Quarterly
```

```
--- Параметры географической области----------------
  input_code                  description
0        W00  All Countries, excluding the IO
```

```
--- Параметры единиц измерения ----------------
     input_code          description
0   IX                  Index
1   PC_CP_A_PT    Percent Change, Corresponding Period, Previous...
2   PC_PP_PT         Percent Change, Previous Period, Percent
3   USD                US Dollars
4   All_Units           All Units
```

```
--- Параметры товаров----------------
          input_code       description
0         PRAWM          Primary Commodity Prices, Agr. Raw Material In...
1         PAGRI           Primary Commodity Prices, Agriculture
2         PALLFNF         Primary Commodity Prices, All index
3         PEXGMETA    Primary Commodity Prices, All Metals EX GOLD I...
4         PALLMETA     Primary Commodity Prices, All Metals Index
..             ...                                    ...
106       PWOOLC        Primary Commodity Prices, Wool, Coarse
107       PWOOLF         Primary Commodity Prices, Wool, Fine
108       POILWTI        Primary Commodity Prices, WTI Crude
109       PZINC          Primary Commodity Prices, Zinc
110       All_Indicators  All Indicators
```

Как видно из полученных результатов, на выходе было получено пять наборов данных.

Первый набор данных получен с использованием функции imf_parameter_defs и содержит ключи для доступа к описанию параметров базы данных PCPS со сведениями о товарах. Значения ключей доступа к описанию параметров представлены в табл.6.2.

Таблица 6.2

Значения ключей доступа к описанию параметров базы данных PCPS

index	Параметр	Описание параметра
0	freq	Частота
1	ref_area	Географические области
2	commodity	Индикатор
3	unit_measure	Единица измерения

Второй набор данных содержит описание параметра базы данных PCPS, который определяет частоту загружаемых данных (с точки зрения периода). Чтобы получить доступ к описанию этого параметра использовалось значение ключа «freg» в следующей инструкции:

```
frg = params['freq']
```

Описания параметра, характеризующего период, представлено в табл.6.3.

Таблица 6.3

Описания параметра, характеризующего период при запросе к базе данных PCPS

index	Параметр	Описание параметра
0	A	Ежегодные данные
1	M	Ежемесячные данные
2	B	Ежеквартальные данные

Третий набор данных содержит описание параметра базы данных PCPS, который определяет географическую область. Чтобы получить доступ к описанию этого параметра использовалось значение ключа «ref_area» в следующей инструкции:

```
ref = params['ref_area']
```

Этот параметр имеет всего одно значение «W00» - это все страны, входящие в МВФ.

Четвертый набор данных содержит описание параметра базы данных PCPS, который определяет единицы измерения. Чтобы получить доступ к описанию этого параметра использовалось значение ключа «unit_measure» в следующей инструкции:

```
ref = params['unit_measure']
```

Описания параметра, характеризующего единицы измерения, приведено в табл.6.4.

Таблица 6.4

Описания параметра, характеризующего единицу измерения при запросе к базе данных PCPS

index	Параметр	Описание
0	IX	Индекс
1	PC_CP_A_PT	Процентное изменение за соответствующий период
2	PC_PP_PT	Процентное изменение за предыдущий период, в процентах
3	USD	Американский доллар
4	All_Units	Все

6.5.Загрузка набора данных из баз данных МВФ с использованием функции imf_dataset

После того, как был получен идентификатор нужной базы данных (database_id), можно запросить данные из этой БД с использованием функции imf_dataset, например:

```
df = imfp.imf_dataset(database_id)
```

Такой запрос может вернуть данные, если размер БД не очень велик, однако для большинства баз данных, такой запрос завершиться неудачей. И даже если получиться загрузить данные, то процесс загрузки займет много времени. Для того чтобы гарантировано получить нужный набор данных необходимо использовать фильтры.

Обратите внимание, что Python при выводе в терминал набора данных Pandas, «обрезает» большой DataFrame, поскольку в Python нет встроенного обозревателя переменных. Однако вы можете запретить Python обрезать наборы данных, используя метод options библиотеки Pandas. Например, чтобы увеличить максимально допустимую ширину столбца до 100 символов, можно использовать следующую инструкцию

```
pandas.options.display.max_colwidth = 100.
```

Итак, для того, чтобы запросить из БД МВФ только нужный набор данных необходимо использовать фильтры, а вернее функцию imf_dataset с параметрами. Существует два способа предоставления параметров функции imf_dataset:

- присвоение значения каждому параметру;
- присвоение значения параметрам из словаря с набором данных.

6.5.1.Загрузка данных из БД МВФ с передачей значений параметрам

Чтобы выполнить загрузку данных из БД МВФ с передачей значений каждому параметру, необходимо получить нужные коды и передать их в функцию imfp.imf_dataset, используя имя параметра и его значение. Ниже приведен пример запроса среднегодовых изменений цен на уголь за период с 2015 по 2025 годы из базы данных Primary Commodity Price System (PCPS) – листинг 6.4.

Листинг 6.4. Модуль data_coal.py для запроса динамики среднегодовых цен на уголь

Примечание.
В демо-версии книги листинги программ удалены.

В этом модуле выполнен импорт библиотеки imfp и получены параметры базы данных на сырьевые товары — PCPS (объект params). Теперь из объекта params можно получить набор нужных нам кодов.

Так как мы собираемся получить данные о среднегодовых ценах на уголь, то временным лагом будет год (Annual). Код этого параметра (selected_freg) получен с использованием следующей инструкции:

```
selected_freq = list(
   params['freq']['input_code'][params['freq']['description'].str.contains("Annual")])
```

Код временного лага выдан в окно терминала:

```
Частота - selected_freq= ['A']
```

Далее получен список кодов для товара уголь (Coal). Список этих кодов сохранен в объекте selected_commodity. Для этого использовалась следующая инструкция:

```
selected_commodity = list(
   params['commodity']['input_code'][params['commodity']['description'].str.contains("Coal")]
)
```

Коды товара «Уголь» имеют следующие значения:

```
Товар - selected_commodity= ['PCOAL', 'PCOALAU', 'PCOALSA']
```

Это коды на следующие виды угля: PCOAL - уголь, PCOALAU — уголь из Австралии, PCOALSA - уголь из южной Африки.

На следующем шаге получен код 'unit_measure' (единицы измерения). Этот код сохранен в объекте selected_unit_measure. Для этого использовалась следующая инструкция:

```
selected_unit_measure = list(
params['unit_measure']['input_code'][params['unit_measure']['description'].str.contains("Index")]
)
```

Код единицы измерения (индекс) выдан в окно терминала:

```
Единица измерения - selected_unit_measure= ['IX']
```

Затем, с использованием полученных параметров, сделано обращение к БД МВФ:

```
df = imfp.imf_dataset(database_id = "PCPS",
                      freq = selected_freq,
                      commodity = selected_commodity,
                      unit_measure = selected_unit_measure,
                      start_year = 2015,
                      end_year = 2025)
```

Примечание.

Представленный в данном разделе программный модуль не включен в список исполняемых примеров интерактивной цифровой книги, так как он может работать только в рамках приложения Python локального компьютера, или быть вызванным из другого модуля веб приложения.

Полученный набор данных сохранен в объекте df. Этот набор данных и имена его колонок в результате работы программы будут выведены в окно терминала:

```
------- Полученный набор данных ----------
  freq ref_area commodity  ... time_format time_period     obs_value
0    A      W00     PCOAL  ...         P1Y        2015   89.0450008241451
1    A      W00     PCOAL  ...         P1Y        2016   100
2    A      W00     PCOAL  ...         P1Y        2017   132.73894436317
3    A      W00     PCOAL  ...         P1Y        2018   156.791334530415
4    A      W00     PCOAL  ...         P1Y        2019   113.954852320044
5    A      W00     PCOAL  ...         P1Y        2020   94.8459750220672
6    A      W00     PCOAL  ...         P1Y        2021   199.93913111868
7    A      W00     PCOAL  ...         P1Y        2022   484.504479365903
8    A      W00     PCOAL  ...         P1Y        2023   227.808322316902
9    A      W00     PCOAL  ...         P1Y        2024   184.201023491731
[10 rows x 8 columns]
```

```
------- Имена колонок ----------
Index(['freq', 'ref_area', 'commodity', 'unit_measure', 'unit_mult',
       'time_format', 'time_period', 'obs_value'],
      dtype='object')
```

Обратите внимание, что все столбцы в возвращаемом наборе данных имеют строковый формат, и для построения графика их нужно преобразовывать в числа, а даты в формат datetime.

В этом наборе данных присутствует колонка time_format со значением P1Y, которое говорит, что даты представлены в формате года. Здесь имеется еще одна колонка - unit_mult (множитель единицы измерения). Она указывает на количество нулей, которые нужно добавить к столбцу obs_value. Если, например, данные в колонке obs_value являются миллионами, то множитель единиц измерения будет равен 6.

Далее из полученного набора данных выбираются только 4 колонки (объект df_col), на который затем накладывается фильтр на код товара PCOAL (объект f_df). То есть, если в загруженном наборе данных будут присутствовать строки с данными на уголь с кодами PCOALAU и PCOALSA, то они будут отсеяны. После фильтрация набора данных df_f будет иметь следующую структуру:

```
------- Фильтр на строки ----------
  time_period unit_measure commodity     obs_value
0        2015           IX     PCOAL   89.0450008241451
1        2016           IX     PCOAL   100
2        2017           IX     PCOAL   132.73894436317
3        2018           IX     PCOAL   156.791334530415
4        2019           IX     PCOAL   113.954852320044
5        2020           IX     PCOAL   94.8459750220672
```

6	2021	IX	PCOAL	199.93913111868
7	2022	IX	PCOAL	484.504479365903
8	2023	IX	PCOAL	227.808322316902
9	2024	IX	PCOAL	184.201023491731

Полученный набор данных df_f может служить основой для формирования графика изменения цен на уголь.

6.5.2.Загрузка данных из БД МВФ с передачей значений параметров в виде словаря с набором данных

Для того чтобы сформировать список параметров запроса, необходимо выбрать нужные фрагменты набора данных из объекта params с параметрами запроса, объединить их, а затем передать в функцию mfp.imf_dataset в качестве общего параметра. В следующем программном коде реализован запрос на среднегодовые данные о ценах на уголь с использованием списка параметров, представленных в словаре (листинг 6.5):

Листинг 6.5. Модуль data_coal_list.py для запроса динамики среднегодовых цен на уголь (словарь параметров)

Примечание.
В демо-версии книги листинги программ удалены.

В этом программном модуле выполнен импорт библиотеки imfp, а затем объект params получает коды всех необходимых параметров запроса к данным МВФ:

- имя базы данных (PCPS);
- freq —временной лаг (годы);
- commodity — товар (уголь);
- unit_measure — единица измерения.

Примечание.
Представленный в данном разделе программный модуль не включен в список исполняемых примеров интерактивной цифровой книги, так как он может работать только в рамках приложения Python локального компьютера, или быть вызванным из другого модуля веб приложения.

Эти параметры собраны в словарь (объект params), который при выполнении данной прогаммы будет выведен в окно терминала:

```
{'freq':
        input_code description
0       A               Annual,
'ref_area':
          input_code   description
0         W00            All Countries, excluding the IO,
 'commodity':
              input_code      description
14            PCOAL           Primary Commodity Prices, Coal index
```

```
15              PCOALAU     Primary Commodity Prices, Coal, Australia
16              PCOALSA     Primary Commodity Prices, Coal, South Africa ,
 'unit_measure':
                  input_code description
0                 IX              Index}
```

Как видно из полученных результатов, объект params представляет собой словарь, где ключом словаря является имя параметра, а значением словаря — набор данных со значениями параметров.

После этого выполнен запрос к базе данных МВФ с передачей идентификатора базы данных и значений параметров из объекта params в виде словаря:

```
Запрос данных через API
df = imfp.imf_dataset(database_id = "PCPS",
                  parameters = params,
                  start_year = 2015,
                  end_year = 2025)
```

Первые строки полученного набора данных df выведены в окно терминала:

```
    freq  ref_area commodity  ...  time_format time_period  obs_value
0   A     W00      PCOAL      ...  P1Y         2015         89.0450008241451
1   A     W00      PCOAL      ...  P1Y         2016         100
2   A     W00      PCOAL      ...  P1Y         2017         132.73894436317
3   A     W00      PCOAL      ...  P1Y         2018         156.791334530415
4   A     W00      PCOAL      ...  P1Y         2019         113.954852320044
```

Далее из полученного набора данных выбираются только 3 колонки (объект df_col), на который затем накладывается фильтр на код товара PCOAL (объект f_df). То есть, если в загруженном наборе данных будут присутствовать строки с данными на уголь с кодами PCOALAU и PCOALSA, то они будут отсеяны. После фильтрация набора данных df_f будет иметь следующую структуру:

```
------- Фильтр на строки ----------
    time_period  commodity   obs_value
0   2015         PCOAL       89.0450008241451
1   2016         PCOAL       100
2   2017         PCOAL       132.73894436317
3   2018         PCOAL       156.791334530415
4   2019         PCOAL       113.954852320044
5   2020         PCOAL       94.8459750220672
6   2021         PCOAL        199.93913111868
7   2022         PCOAL       484.504479365903
8   2023         PCOAL       227.808322316902
9   2024         PCOAL       184.201023491731
```

Полученный набор данных df_f может служить основой для формирования графика изменения цен на уголь.

6.6.Очистка набора данных перед графической визуализацией

Обратите внимание, что все столбцы в возвращаемом наборе данных из БД МВФ являются символьными, а для формирования временного ряда и построения графиков необходимо даты и числовые данные преобразовать допустимый тип. Напишем функцию, которая обеспечивает загрузку и очистку данных об изменении стоимости угля, полученных из БД МВФ (листинг 6.6).

Листинг 6.6. Модуль data_coal_fn.py функция загрузки и очистки данных о стоимости угля

Примечание.
В демо-версии книги листинги программ удалены.

В этом модуле выполнен импорт необходимых библиотек и создана функция get_data, которая принимает следующие параметры:

- db — идентификатор базы данных МВФ;
- fr — параметр, задающий временной лаг (годы);
- com — параметр, задающий товар (уголь);
- uni — единица измерения товара;
- start — дата начала периода (целое число);
- end - дата конца периода (целое число).

Затем с использованием этих параметров делается запрос к БД МВФ, полученные данные сохраняются в объекте data, а типы полученных данных выводятся в окно терминала.

Примечание.
Представленный в данном разделе программный модуль не включен в список исполняемых примеров интерактивной цифровой книги, так как он может работать только в рамках приложения Python локального компьютера, или быть вызванным из другого модуля веб приложения.

В результате работы этого программного модуля выводятся следующие сведения о типах данных загруженного DataFrame:

```
freq            object
ref_area        object
commodity       object
unit_measure    object
unit_mult       object
time_format     object
time_period     object
obs_value       object
dtype: object
```

Как видно из полученных результатов, все загруженные данные имею тип object, фактически это строковые объекты. Для построения графиков необходимо сначала выбрать только нужные колонки и строки.. Это сделано с использованием следующих инструкций:

```
data = data[['time_period', 'commodity', 'obs_value']]
data = data[data['commodity'] == 'PCOAL']
```

Затем значения колонки obs_value приведены к числовому типу (числа с плавающей точкой), а значения колонки time_period трансформированы в даты. Очищенный набор данных возвращается в точку вызова. Теперь загруженный набор данных имеет следующую структуру:

```
   time_period  commodity  obs_value
0  2015-01-01    PCOAL      89.045001
1  2016-01-01    PCOAL     100.000000
2  2017-01-01    PCOAL     132.738944
3  2018-01-01    PCOAL     156.791335
4  2019-01-01    PCOAL     113.954852
```

А данные этого набора имеют следующие типы:

```
time_period   datetime64[ns]
commodity     object
obs_value     float64
dtype:        object
```

В таком виде данные пригодны для формирования графиков. Теперь можно закомментировать строки вывода результатов в окно терминала и использовать данный модуль в приложениях Django. В следующем разделе показано, как можно на основе полученных данных построить график изменения цен на уголь с использованием интерфейса plotly_express библиотеки plotly.

6.7.Модули Django для визуализации данных о среднегодовой стоимости угля

Рассмотрим пример, в котором получим данные об изменении среднегодовой стоимости угля за заданный период на основе данных, загруженных из БД МВФ. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 6.7).

Листинг 6.7. Ссылка активации маршрута fig_coal_year

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем fig_coal_year (листинг 6.8).

Листинг 6.8. Маршрут URL с именем fig_coal_year

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление fig_coal_year (листинг 6.9).

Листинг 6.9. Представление (view) fig_coal_year

Примечание.
В демо-версии книги листинги программ удалены.

В этом модуле выполнен импорт необходимых библиотек и модуля data_coal_fn, код которого описан в предыдущем разделе. Далее сформированы параметры запроса к БД МВФ:

- data_base = 'PCPS' — идентификатор базы данных МВФ;
- f_reg = 'A' - код временного лага (годы);
- commod = 'PCOAL' — код товара (уголь);
- u_m = 'IX' — код единицы измерения (индекс);
- d1 = 2015 — начала периода (год);
- d2 = 2025 - конец периода (год).

После этого выполнено обращение к БД МВФ для получения данных о стоимости угля (объект df).

На следующем шаге создан график (объект fig) с описанием макета графика:

```
fig = px.line(df, x='time_period', y='obs_value')
    # Формирование параметров графика
    fig.update_layout(
        xaxis=dict(title="Годы"),
        yaxis=dict(title="Мировые цены на уголь, $"),
        title='Динамика изменения цен на уголь за период: с ' + str(d1) + ' по ' + str(d2),
        hoverlabel=dict(font_size=20),  # Размера шрифта для данных
    )
```

После этого график приведен к формату, в котором он может быть показан на HTML странице (объект plot_div) и создана строка с заголовком страницы (title_str). Полученный график (plot_div) с заголовком страницы через контекстную переменную context передается в шаблон fig_plotly.html (листинг 6.10).

Листинг 6.10. Шаблон fig_plotly.html

Примечание.
В демо-версии книги листинги программ удалены.

Это шаблон с контейнером, в котором имеются блоки для вывода заголовка страницы и объекта plot_div. Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 6» и активировать ссылку «Среднегодовые цены на уголь (БД МВФ)».

Если выполнить набор вышеприведенных модулей, то получим следующий результат (рис.6.1).



Рис. 6.1. График в шаблоне Django с данными о среднегодовой стоимости угля, загруженными из БД МВФ

Как видно из данного графика, наибольший всплеск цен на уголь был 2022 году.

6.8.Модули Django для визуализации данных о среднемесячной стоимости угля

Если нужно получить информацию за более короткий период, например в разрезе месяцев, то нужно установить код временного лага на месяц «M» и поменять формат представления интервала дат.

Рассмотрим пример, в котором получим данные об изменении среднемесячной стоимости угля за заданный период на основе данных, загруженных из БД МВФ. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 6.11).

Листинг 6.11. Ссылка активации маршрута fig_coal_m

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем fig_coal_m (листинг 6.12).

Листинг 6.12. Маршрут URL с именем fig_coal_m

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление fig_coal_m (листинг 6.13).

Листинг 6.13. Представление (view) fig_coal_m

Примечание.
В демо-версии книги листинги программ удалены.

В этом модуле выполнен импорт необходимых библиотек и модуля data_coal_fn, код которого описан в предыдущем разделе. Далее сформированы параметры запроса к БД МВФ:

- data_base = 'PCPS' — идентификатор базы данных МВФ;
- f_reg = 'M' - код временного лага (месяцы);
- commod = 'PCOAL' — код товара (уголь);
- u_m = 'IX' — код единицы измерения (индекс);
- d1 = 2015 — начала периода (год);
- d2 = 2025 - конец периода (год).

После этого выполнено обращение к БД МВФ для получения данных о стоимости угля (объект df).

На следующем шаге создан графика (объект fig) с описанием макета графика:

```
fig = px.line(df, x='time_period', y='obs_value')
   # Формирование параметров графика
   fig.update_layout(
      xaxis=dict(title="Годы"),
      yaxis=dict(title="Мировые цены на уголь, $"),
      title='Динамика изменения цен на уголь за период: с ' + str(d1) + ' по ' + str(d2),
      hoverlabel=dict(font_size=20),  # Размера шрифта для данных
   )
```

После этого график приведен к формату, в котором он может быть показан на HTML странице (объект plot_div) и создана строка с заголовком страницы (title_str). Полученный график (plot_div) с заголовком страницы через контекстную переменную context передается в шаблон fig_plotly.html (листинг 6.14).

Листинг 6.14. Шаблон fig_plotly.html

Примечание.
В демо-версии книги листинги программ удалены.

Это шаблон с контейнером, в котором имеются блоки для вывода заголовка страницы и объекта plot_div. Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 6» и активировать ссылку «Среднемесячные цены на уголь (БД МВФ)».

Если выполнить набор вышеприведенных модулей, то получим следующий результат (рис.6.2).

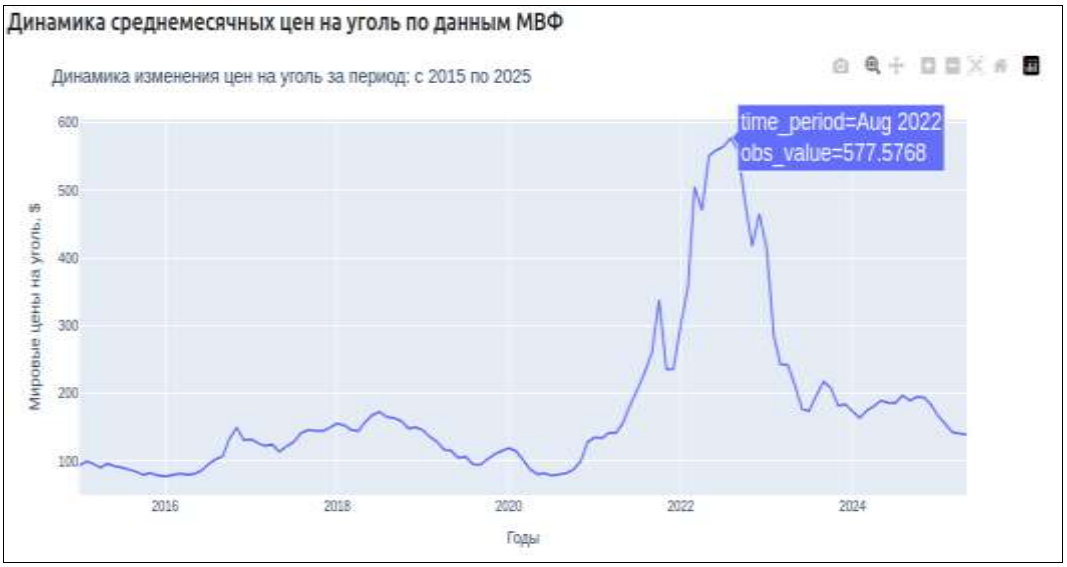


Рис. 6.2. График в шаблоне Django с данными о среднемесячной стоимости угля, загруженными из БД МВФ

Среднемесячные данные позволяют получить более детальную информацию о динамике изменения цены на уголь. Как видно из данного графика, наибольший всплеск цен на уголь был в августе 2022 года.

6.9.Модули Django для визуализации квартальных данных о стоимости угля

Если нужно укрупнить месячные данные, например, получить квартальные показатели, то нужно установить код временного лага на квартал - «Q».

Рассмотрим пример, в котором получим данные об изменении стоимости угля за заданный период на основе квартальных данных, загруженных из БД МВФ. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 6.15).

Листинг 6.15. Ссылка активации маршрута fig_coal_q

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем fig_coal_q (листинг 6.16).

Листинг 6.16. Маршрут URL с именем fig_coal_q

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление fig_coal_q (листинг 6.17).

Листинг 6.17. Представление (view) fig_coal_q

Примечание.
В демо-версии книги листинги программ удалены.

В этом модуле выполнен импорт необходимых библиотек и модуля data_coal_fn, код которого описан в предыдущем разделе. Далее сформированы параметры запроса к БД МВФ:

- data_base = 'PCPS' — идентификатор базы данных МВФ;
- f_reg = 'Q' - код временного лага (квартал);
- commod = 'PCOAL' — код товара (уголь);
- u_m = 'IX' — код единицы измерения (индекс);
- d1 = 2015 — начала периода (год);
- d2 = 2025 - конец периода (год).

После этого выполнено обращение к БД МВФ для получения данных о стоимости угля (объект df).

На следующем шаге создан графика (объект fig) с описанием макета графика:

```
fig = px.line(df, x='time_period', y='obs_value')
   # Формирование параметров графика
   fig.update_layout(
      xaxis=dict(title="Годы"),
      yaxis=dict(title="Мировые цены на уголь, $"),
      title='Динамика изменения цен на уголь за период: с ' + str(d1) + ' по ' + str(d2),
      hoverlabel=dict(font_size=20),  # Размера шрифта для данных
   )
```

После этого график приведен к формату, в котором он может быть показан на HTML странице (объект plot_div) и создана строка с заголовком страницы (title_str). Полученный

график (plot_div) с заголовком страницы через контекстную переменную context передается в шаблон fig_plotly.html (листинг 6.18).

Листинг 6.18. Шаблон fig_plotly.html

Примечание.
В демо-версии книги листинги программ удалены.

Это шаблон с контейнером, в котором имеются блоки для вывода заголовка страницы и объекта plot_div. Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 6» и активировать ссылку «Квартальные данные о ценах угля (БД МВФ)».

Если выполнить набор вышеприведенных модулей, то получим следующий результат (рис.6.3).

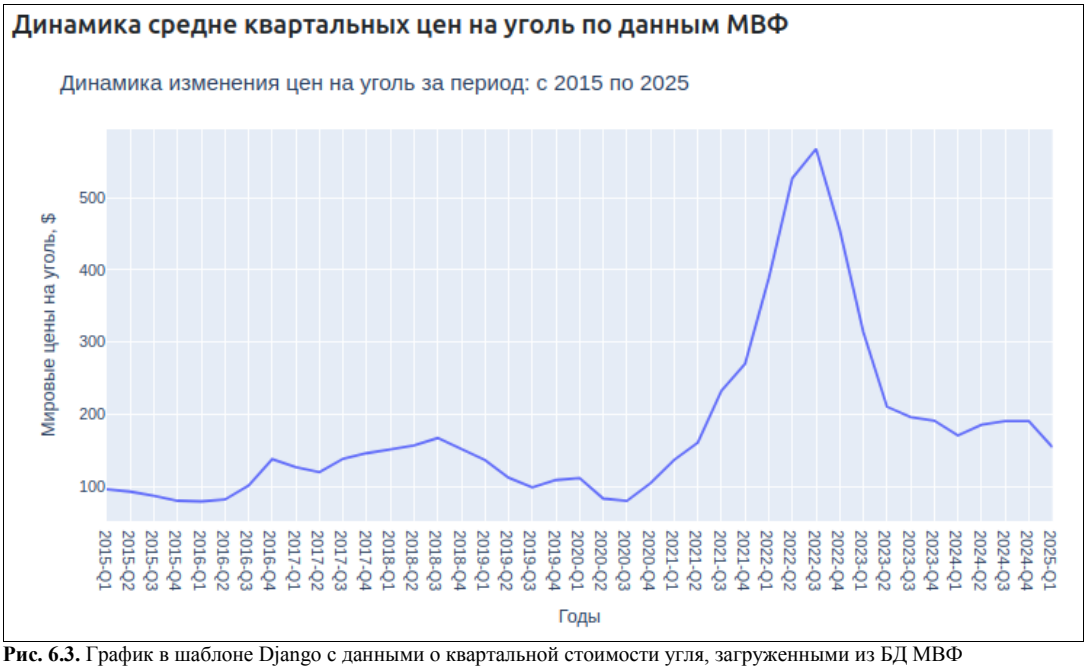


Рис. 6.3. График в шаблоне Django с данными о квартальной стоимости угля, загруженными из БД МВФ

6.10.Модули Django для визуализации данных мировых ценах на золото

Если известны значения кодов и индикаторов, то в программном коде можно сформировать запрос к БД МВФ, указав значения этих кодов в параметрах запроса.

В качестве примера получим данные динамике мировых цен на золото за заданный период на основе месячных данных, загруженных из БД МВФ. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 6.19).

Листинг 6.19. Ссылка активации маршрута fig_gold_m

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем fig_gold_m (листинг 6.20).

Листинг 6.20. Маршрут URL с именем fig_gold_m

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление fig_gold_m (листинг 6.21).

Листинг 6.21. Представление (view) fig_gold_m

Примечание.
В демо-версии книги листинги программ удалены.

В этом модуле выполнен импорт необходимых библиотек и модуля data_coal_fn, код которого описан в предыдущем разделе. Далее сформированы параметры запроса к БД МВФ:

- data_base = 'PCPS' — идентификатор базы данных МВФ;
- f_reg = 'M' - код временного лага (месяц);
- commod = 'PGOLD' — код товара (золото);
- u_m = 'USD' — код единицы измерения (доллары);
- d1 = 2010 — начала периода (год);
- d2 = 2025 - конец периода (год).

После этого выполнено обращение к БД МВФ для получения данных о стоимости золота (объект df).

На следующем шаге создан графика (объект fig) с описанием макета графика:

```
fig = px.line(df, x='time_period', y='obs_value')
   # Формирование параметров графика
   fig.update_layout(
      xaxis=dict(title="Годы"),
      yaxis=dict(title="Мировые цены на уголь, $"),
      title='Мировые цены на зологто за период: с ' + str(d1) + ' по ' + str(d2),
```

```
    hoverlabel=dict(font_size=20),  # Размера шрифта для данных
)
```

После этого график приведен к формату, в котором он может быть показан на HTML странице (объект plot_div) и создана строка с заголовком страницы (title_str). Полученный график (plot_div) с заголовком страницы через контекстную переменную context передается в шаблон fig_plotly.html (листинг 6.22).

Листинг 6.22. Шаблон fig_plotly.html

Примечание.
В демо-версии книги листинги программ удалены.

Это шаблон с контейнером, в котором имеются блоки для вывода заголовка страницы и объекта plot_div. Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 6» и активировать ссылку «Мировые цены на золото (БД МВФ)».

Если выполнить набор вышеприведенных модулей, то получим следующий результат (рис.6.4).



Рис. 6.4. График в шаблоне Django с данными о динамике мировых цен на золото, загруженными из БД МВФ

6.11.База данных международной финансовой статистики IFS

База данных международной финансовой статистики (IFS - International Financial Statistics) Международного валютного фонда содержит финансовые показатели большинства стран-членов МВФ, а также некоторых других стран и регионов. Вот только некоторые показатели, которые содержаться в этой БД:

- платежный баланс;
- цены на товары;
- обменные курсы;
- промышленное производство;
- процентные ставки;
- положение международных инвестиций;
- международная ликвидность;
- международные транзакции;
- статистика труда;
- деньги и банковская деятельность;
- национальные счета;
- население и т.д.

6.11.1.База данных международной финансовой статистики IFS (загрузка и очистка данных)

Загрузку данных из базы данных IFS можно выполнять с использованием библиотеки imfp, для чего в запросе необходимо указать нужные параметры. Напишем функцию, которая обеспечивает загрузку и очистку данных на примере изменения индикатора NXG_R_XDC (Exports of Goods, Real, Domestic Currency - экспорт товаров, реальный, в национальной валюте). Код этой функции приведен в листинге 6.23.

Листинг 6.23. Модуль data_ifs функция загрузки и очистки данных об экспорте товаров

Примечание.
В демо-версии книги листинги программ удалены.

В этом модуле выполнен импорт необходимых библиотек и создана функция get_data, которая принимает следующие параметры:

- db - идентификатор базы данных МВФ;
- freq - параметр, задающий временной лаг;
- ref_area — список параметров, определяющий регион (страны);
- indicator — индиатор (экономический показатель);

- start - дата начала периода (целое число);
- end -дата конца периода (целое число).

Затем с использованием эти параметров делается запрос к БД МВФ, полученные данные сохраняются в объекте data и выводятся в окно терминала.

> ***Примечание.***
> Представленный в данном разделе программный модуль не включен в список исполняемых примеров интерактивной цифровой книги, так как он может работать только в рамках приложения Python локального компьютера, или быть вызванным из другого модуля веб приложения.

В результате работы программы будут получены следующие сведения о загруженном наборе данных DataFrame:

	freq	ref_area	indicator	...	time_format	time_period	obs_value
0	A	US	NXG_R_XDC	...	P1Y	2015	1475691.625
1	A	US	NXG_R_XDC	...	P1Y	2016	1485211.125
2	A	US	NXG_R_XDC	...	P1Y	2017	1546657.3125
3	A	US	NXG_R_XDC	...	P1Y	2018	1612140.125
4	A	US	NXG_R_XDC	...	P1Y	2019	1614882.625
5	A	US	NXG_R_XDC	...	P1Y	2020	1452570.75
6	A	US	NXG_R_XDC	...	P1Y	2021	1564358.8125
7	A	US	NXG_R_XDC	...	P1Y	2022	1656846.8125
8	A	US	NXG_R_XDC	...	P1Y	2023	1694280.3125
9	A	US	NXG_R_XDC	...	P1Y	2024	1731554.8125
0	A	GB	NXG_R_XDC	...	P1Y	2015	397847
1	A	GB	NXG_R_XDC	...	P1Y	2016	401232
2	A	GB	NXG_R_XDC	...	P1Y	2017	429483
3	A	GB	NXG_R_XDC	...	P1Y	2018	431517
4	A	GB	NXG_R_XDC	...	P1Y	2019	440569
5	A	GB	NXG_R_XDC	...	P1Y	2020	385324
6	A	GB	NXG_R_XDC	...	P1Y	2021	389082
7	A	GB	NXG_R_XDC	...	P1Y	2022	434123
8	A	GB	NXG_R_XDC	...	P1Y	2023	393696
9	A	GB	NXG_R_XDC	...	P1Y	2024	362716
0	A	RU	NXG_R_XDC	...	P1Y	2015	31720890.4
1	A	RU	NXG_R_XDC	...	P1Y	2016	32984943.8
2	A	RU	NXG_R_XDC	...	P1Y	2017	34234702.4
3	A	RU	NXG_R_XDC	...	P1Y	2018	35707337.1
4	A	RU	NXG_R_XDC	...	P1Y	2019	36268881.5
5	A	RU	NXG_R_XDC	...	P1Y	2020	35806320.5
6	A	RU	NXG_R_XDC	...	P1Y	2021	40421469.3

Как видно из полученных результатов, для построения графиков из загруженных данных нам необходимы следующие колонки: 'time_period', 'ref_area', 'obs_value'. Выбор этих колонок выполняется с использованием следующей инструкции:

```
data = data[['time_period', 'ref_area', 'obs_value']]
```

Затем значения колонки obs_value приведены к числовому типу (числа с плавающей точкой), а значения колонки time_period трансформированы в даты:

```
data = data.astype({"obs_value": float})
if freq == 'A':
   data['time_period'] = pd.to_datetime(data['time_period'], format='%Y')
elif freq == 'M':
   data['time_period'] = pd.to_datetime(data['time_period'], format='%Y-%m')
```

Информация в таком виде не удобна для формирования графиков, так как данные об экспорте товаров для разных стран находятся в одной колонке. Потому далее из полученного набора данных формируется сводная таблица с использованием следующих инструкций:

```
data = data.reset_index()
data = data.pivot(index='time_period', columns='ref_area', values='obs_value')
```

Здесь в первой строке удаляется текущий индекс, затем колонка time_period делается индексной, формируются новые колонки для стран из значений ref_area, в качестве данных будут использоваться значения колонки obs_value. После этого набор данных приобретает следующую структуру:

```
ref_area            GB        RU            US
time_period
2015-01-01   397847.0  31720890.4  1.475692e+06
2016-01-01   401232.0  32984943.8  1.485211e+06
2017-01-01   429483.0  34234702.4  1.546657e+06
2018-01-01   431517.0  35707337.1  1.612140e+06
2019-01-01   440569.0  36268881.5  1.614883e+06
```

В таком виде данные возвращаются в точку вызова. Обработанные данные пригодны для формирования графиков.

Для отладки данного модуля были использованы следующие параметры:

- data_base = 'IFS' - идентификатор базы данных;
- f_req = 'A' — временной лаг (год);
- ref = ['GB', 'US', 'RU'] — список стран (Великобритания, Америка, Россия);
- indic = 'NXG_R_XDC' — индикатор (экспорт товаров);
- d1 = 2015 — начало периода;
- d2 = 2025 — конец периода.

Обратите внимание на следующие инструкции, которые использовались при отладке работы данного модуля:

```
params = imfp.imf_parameters("IFS")
print(params)
# Сохранение полного списка параметров
params['ref_area'].to_excel('params_ref_area_ifs.xlsx')
params['indicator'].to_excel('params_indicator_ifs.xlsx')
```

Здесь в окно терминала были выведены параметры, которые можно использовать при запросах к БД IFS. Так как список параметров ref_area и indicator достаточно длинный (более сотни строк), то они сохранены в XLSX файле. Эти файлы можно открыть с

помощью Excel и выбрать значения нужных параметров для формирования запросов к БД МВФ.

Теперь можно закомментировать строки вывода результатов в окно терминала и использовать данный программный модуль в приложениях Django.

В следующем разделе показано, как можно на основе полученных данных построить график изменения экспорта товаров для разных стран с использованием интерфейса plotly.graph_objs библиотеки plotly.

6.11.2.Модули Django для визуализации данных об экспорте товаров из БД МВФ (IFS)

Если известны значения кодов и индикаторов базы данных IFS, то в программном коде можно сформировать запрос, указав значения этих кодов в параметрах запроса.

В качестве примера получим данные о динамике экспорта товаров за заданный период на основе годовых данных, загруженных из базы данных IFS МВФ. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 6.24).

Листинг 6.24. Ссылка активации маршрута fig_ifs

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем fig_ifs (листинг 6.25).

Листинг 6.25. Маршрут URL с именем fig_ifs

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление fig_ifs (листинг 6.26).

Листинг 6.26. Представление (view) fig_ifs

Примечание.
В демо-версии книги листинги программ удалены.

В этом модуле выполнен импорт необходимых библиотек и модуля data_ifs, код которого описан в предыдущем разделе. Далее сформированы параметры запроса к БД IFS МВФ:

- data_base = 'IFS' — идентификатор базы данных МВФ;
- f_reg = 'A' - код временного лага (год);
- ref = ['GB', 'US'] — список идентификаторов регионов (стран);
- indic = 'NXG_R_XDC' — индикатор (экспорт товаров);
- d1 = 2010 — начала периода (год);

- d2 = 2025 - конец периода (год).

После этого выполнено обращение к БД IFS МВФ для получения данных об экспорте товаров (объект df).

На следующем шаге создан графика (объект fig) с описанием макета графика:

```
fig = go.Figure()
fig.add_trace(go.Scatter(x=df.index, y=df['US'], name='США'))
  fig.add_trace(go.Scatter(x=df.index, y=df['GB'], name='Великобритания'))
    # Формирование параметров графика
  fig.update_layout(
     xaxis_title="Дата",
     yaxis_title="Экспорт товаров",
     title='ДинLiberation Serifамика изменения экспорта товаров за период: с ' + str(d1) + ' по
' + str(d2))
```

После этого график приведен к формату, в котором он может быть показан на HTML странице (объект plot_div) и создана строка с заголовком страницы (title_str). Полученный график (plot_div) с заголовком страницы через контекстную переменную context передается в шаблон fig_plotly.html (листинг 6.27).

Листинг 6.27. Шаблон fig_plotly.html

> ***Примечание.***
> В демо-версии книги листинги программ удалены.

Это шаблон с контейнером, в котором имеются блоки для вывода заголовка страницы и объекта plot_div. Теперь можно проверить работу данных модулей.

> ***Примечание.***
> В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 6» и активировать ссылку «Динамика экспорта товаров из БД IFS».

Если выполнить набор вышеприведенных модулей, то получим следующий результат (рис.6.5).

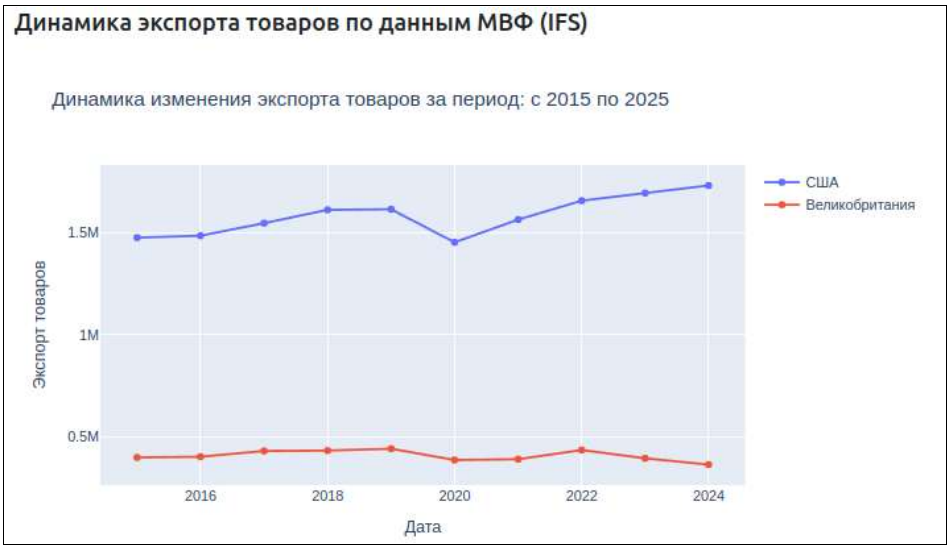


Рис. 6.5. График в шаблоне Django с данными о динамике экспорта товаров, загруженными из БД IMF МВФ

6.12.Общий алгоритм доступа к данным через API IMF с библиотекой requests

Данные из базы International Financial Statistics (IFS) можно загрузить в Python с помощью библиотеки requests. Для этого нужно выполнить запрос к API базы данных и получить ответ в формате JSON. Для этого нужно выполнить следующие действия:

- ❑ импортировать необходимые библиотеки;
- ❑ определить URL - адрес API базы IFS;
- ❑ указать параметры запроса (индекс, временной лаг, страну или регион, период);
- ❑ отправить запрос к API с помощью метода requests.get();
- ❑ преобразовать полученный ответ в формат, подходящий для работы с pandas;
- ❑ привести полученные данные к виду, удобному для отображения в графическом виде.

С помощью библиотеки requests и API МВФ можно получить доступ к макроэкономическим данным в формате JSON, охватывающим более 180 стран.

6.12.1.Модуль Python загрузки и очистки данных из БД IFS с библиотекой requests

Напишем функцию, которая обеспечивает загрузку и очистку данных из БД IFS на примере изменения индикатора PMP_IX (индекс цен на импорт товаров) с библиотекой requiests (листинг 6.28).

Листинг 6.28. Модуль request_ifs.py функция загрузки и очистки данных об импорте товаров

Примечание.
В демо-версии книги листинги программ удалены.

В этом программном модуле выполнен импорт необходимых библиотек и создана функция с именем get_data(). Эта функция принимает следующие параметры»:

- params - параметры запроса к БД МВФ;
- start — дата начала периода;
- end — дата конца периода.

В теле функции сформировано несколько объектов:

- api_url — заголовок URL адреса;
- star_p — фрагмент URL адреса для определения даты начала периода;
- end_p - фрагмент URL адреса для определения даты конца периода.

В теле функции из полученных параметров и фрагментов URL адреса формируется полный URL адрес запроса к удаленной БД:

```
url = api_url + params + star_p + start + end_p + end
```

После этого с использованием метода requests.get выполняется загрузка данных из API IFS в объект data. Данные загружаются в формате JSON:

```
data = requests.get(url).json()
```

На следующем шаге полученные данные преобразованы в набор данных pandas:

```
data = pd.DataFrame(data['CompactData']['DataSet']['Series']['Obs'])
```

Примечание.
Представленный в данном разделе программный модуль не включен в список исполняемых примеров интерактивной цифровой книги, так как он может работать только в рамках приложения Python локального компьютера, или быть вызванным из другого модуля веб приложения.

Для оценки структуры данных они выведены в окно терминала:

```
  @TIME_PERIOD       @OBS_VALUE
0      2010-01  99.2967466316135
1      2010-02   99.217877094972
2      2010-03  99.6122247781794
3      2010-04   100.71639829116
4      2010-05  99.9277029247453
```

```
@TIME_PERIOD    object
@OBS_VALUE      object
dtype: object
```

Как видно из полученных результатов, в наборе данных две колонки (@TIME_PERIOD и @OBS_VALUE), тип данных — object. В таком виде данные не пригодны для формирования графиков, поэтому далее выполнены некоторые преобразования:

```
data.columns = ['date', 'value']
data = data.astype({"value": float})
data['date'] = data['date'].astype('datetime64[ns]')
```

В этих инструкциях колонкам присвоены новые имена (date, value), значения колонки value приведены к числовому формату, а значения колонки date к формату datetime. Обработанный набор данных возвращается в точку вызова.

Для отладки этого программного модуля использовались следующие данные:

- par = 'M.US.PMP_IX.' - параметры запроса к БД IFS;
- d1 = '2010' — дата начала периода;
- d2 = '2025' — дата конца периода.

В параметрах запроса использовались следующие ключи, которые разделены точками:

- M — период (помесячные данные);
- US — код страны (США);
- PMP_IX - экономический индикатор (индекс цен на импорт).

Возвращенные функцией get_data данные выведены в окно терминала и имеют следующую структуру:

```
      date       value
0 2010-01-01  99.296747
1 2010-02-01  99.217877
2 2010-03-01  99.612225
3 2010-04-01 100.716398
4 2010-05-01  99.927703
```

```
date    datetime64[ns]
value          float64
```

В таком виде данные пригодны для формирования графиков. Теперь можно закомментировать строки вывода в окно терминала и использовать данную функцию в приложениях Django. В следующем разделе показано, как можно на основе полученных данных построить график изменения индекса цен на импорт товаров с использованием интерфейса plotly_express библиотеки plotly.

6.12.2.Модули Django для визуализации данных об индексе цен импорта товаров из БД МВФ (IFS)

Если известны значения кодов и индикаторов базы данных IFS, то в программном коде можно сформировать запрос, указав значения этих кодов в параметрах запроса.

В данном примере извлекаются ежемесячные данные (частота - M), индекса цен на импорт товаров (индикатор - PMP_IX) для США (код станы - US) из БД Международной финансовой статистики (IFS). Этот показатель отражает изменения стоимости товаров и услуг, ввозимых в страну из-за рубежа, и измеряет, насколько дороже или дешевле становятся импортные продукты с течением времени. Показатель рассчитывается относительно базового периода, который принимается за точку отсчета. Например, если в базовом году индекс равен 100, а через год он вырос до 105, это означает, что цены на импорт выросли на 5%.

Для реализации данного примера в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 6.29).

Листинг 6.29. Ссылка активации маршрута fig_import_us

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем fig_import_us (листинг 6.30).

Листинг 6.30. Маршрут URL с именем fig_import_us

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление fig_import_us (листинг 6.31).

Листинг 6.31. Представление (view) fig_import_us

Примечание.
В демо-версии книги листинги программ удалены.

В этом модуле выполнен импорт необходимых библиотек и модуля request_ifs, код которого описан в предыдущем разделе. Далее сформированы параметры запроса к БД IFS МВФ:

- par = 'M.US.PMP_IX.' - параметры запроса к БД IFS;
- d1 = '2010' — дата начала периода;
- d2 = '2025' — дата конца периода.

В параметрах запроса использовались следующие ключи, которые разделены точками:

- M — период (месячные данные);
- US — код страны (США);
- PMP_IX - экономический индикатор (индекс цен на импорт).

После этого выполнено обращение к БД IFS МВФ для получения данных об экспорте товаров (объект df).

На следующем шаге создан графика (объект fig) с описанием макета графика:

```
# Формируем график
   fig = px.line(df, x='date', y='value',
                 title='Индекс цен на импорт товаров (США)')
   fig.update_layout(xaxis_title='Дата (годы)',
                     yaxis_title='Индекс цен на импорт')
```

После этого график приведен к формату, в котором он может быть показан на HTML странице (объект plot_div) и создана строка с заголовком страницы (title_str). Полученный график (plot_div) с заголовком страницы через контекстную переменную context передается в шаблон fig_plotly.html (листинг 6.32).

Листинг 6.32. Шаблон fig_plotly.html

Примечание.
В демо-версии книги листинги программ удалены.

Это шаблон с контейнером, в котором имеются блоки для вывода заголовка страницы и объекта plot_div. Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 6» и активировать ссылку « Импорт товаров, США (IFS requests)».

Если выполнить набор вышеприведенных модулей, то получим следующий результат (рис.6.6).



Рис. 6.6. График в шаблоне Django с данными о динамике индекса цен на импорт товаров, загруженными из БД IFS МВФ

6.12.3.Модуль Python загрузки и очистки данных из БД торговой статистики с библиотекой requests

Предположим, что нас интересует объем товаров, импортируемых Великобританией из стран Европейского союза, и общий мировой импорт. Для этого требуется информация как о торговле между Великобританией и странами ЕС, так и об общей мировой торговле Великобритании. Запрос к API позволяет объединять данные по одному географическому измерению, добавляя знак «+» межу нужными кодами (код для стран ЕС - B0, код для мировой торговли - W00).

Напишем функцию, которая обеспечивает загрузку и очистку данных из БД торговой статистики на примере изменения индикатора TMG_CIF_USD (товары, стоимость импорта, стоимость, страхование, фрахт (CIF), доллары США) с библиотекой requests (листинг 6.33).

Листинг 6.33. Модуль request_dot.py функция загрузки и очистки данных об импорте товаров

Примечание.
В демо-версии книги листинги программ удалены.

В этом программном модуле выполнен импорт необходимых библиотек и создана функция с именем get_data(). Эта функция принимает следующие параметры»

- params - параметры запроса к БД МВФ;
- start — дата начала периода;
- end — дата конца периода.

В параметрах запроса использовались следующие ключи, которые разделены точками:

- M — период (помесячные данные);
- GB — код страны (Великобритания);
- MG_CIF_USD - экономический индикатор (импорт товаров);
- B0 — объем импорта из стран Европейского союза;
- W00 — объем мирового импорта.

В теле функции сформировано несколько объектов:

- api_url — заголовок URL адреса;
- star_p — фрагмент URL адреса для определения даты начала периода;
- end_p - фрагмент URL адреса для определения даты конца периода.

В теле функции из полученных параметров и фрагментов URL адреса формируется полный URL адрес запроса к удаленной БД:

```
url = api_url + params + star_p + start + end_p + end
```

После этого с использованием метода requests.get выполняется загрузка данных из API DOTS в объект data. Данные загружаются в формате JSON:

```
data = requests.get(url).json()
```

На следующем шаге полученные данные преобразованы в набор данных pandas:

```
data = pd.DataFrame({s['@COUNTERPART_AREA']: {pd.to_datetime(i['@TIME_PERIOD']):
                round(float(i['@OBS_VALUE']), 1) for i in s['Obs']}
                for s in data['CompactData']['DataSet']['Series']})
```

Примечание.
Представленный в данном разделе программный модуль не включен в список исполняемых примеров интерактивной цифровой книги, так как он может работать только в рамках приложения Python локального компьютера, или быть вызванным из другого модуля веб приложения.

Для оценки структуры данных они выведены в окно терминала:

```
                W00      B0
2010-01-01  44287.5  21494.7
2010-02-01  42874.2  21428.0data['date'] = data['date'].astype('datetime64[ns]')
2010-03-01  49062.8  25025.0
2010-04-01  46573.1  23343.5
2010-05-01  46238.0  21150.5
W00    float64
B0     float64
dtype: object
```

Как видно из полученных результатов, в наборе данных две колонки (W00 и B0), которые имею числовой тип данных. В таком виде данные пригодны для формирования графиков, поэтому далее они возвращаются в точку вызова.

Теперь можно закомментировать строки вывода результатов в окно терминала и использовать данную функцию в приложениях Django. В следующем разделе показано, как можно на основе полученных данных построить график объема импорта с использованием интерфейса plotly.graph_objs библиотеки plotly.

6.12.4.Модули Django для визуализации данных об импорте товаров из БД МВФ (DOT)

Если известны значения кодов и индикаторов базы данных, то в программном коде можно сформировать запрос, указав значения этих кодов в параметрах запроса. В данном примере извлекаются ежемесячные данные (частота - M), импорта товаров (индикатор - TMG_CIF_USD) для Великобритании (код станы - UC) из БД торговой статистики (DOT).

Для реализации данного примера в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 6.34).

Листинг 6.34. Ссылка активации маршрута fig_import_uc

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем fig_import_uc (листинг 6.35).

Листинг 6.35. Маршрут URL с именем fig_import_uc

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление fig_import_uc (листинг 6.36).

Листинг 6.36. Представление (view) fig_import_uc

Примечание.
В демо-версии книги листинги программ удалены.

В этом модуле выполнен импорт необходимых библиотек и модуля request_dot, код которого описан в предыдущем разделе. Далее сформированы параметры запроса к БД DOT МВФ:

- par = 'M.GB.TMG_CIF_USD.B0+W00.' - параметры запроса к БД DOT;
- d1 = '2010' — дата начала периода;
- d2 = '2025' — дата конца периода.

В параметрах запроса использовались следующие ключи, которые разделены точками:

- M — период (месячные данные);
- GB — код страны (Великобритания);
- MG_CIF_USD - экономический индикатор (импорт товаров);
- B0 — объем импорта из стран Европейского союза;
- W00 — объем мирового импорта.

После этого выполнено обращение к БД IFS МВФ для получения данных об экспорте товаров (объект df).

На следующем шаге создан графика (объект fig) с описанием макета графика:

```
# Формируем график
    fig = go.Figure()
    fig.add_trace(go.Scatter(x=df.index, y=df['W00'], name='Из всех стран'))
    fig.add_trace(go.Scatter(x=df.index, y=df['B0'], name='Из стан ЕС'))
    # Формирование параметров графика
    fig.update_layout(
        xaxis_title="Дата",
        yaxis_title="Импорт товаров",
        title='Динамика импорта Великобритании за период: с ' + str(d1) + ' по ' + str(d2))
```

После этого график приведен к формату, в котором он может быть показан на HTML странице (объект plot_div) и создана строка с заголовком страницы (title_str). Полученный график (plot_div) с заголовком страницы через контекстную переменную context передается в шаблон fig_plotly.html (листинг 6.37).

Листинг 6.37. Шаблон fig_plotly.html

Примечание.
В демо-версии книги листинги программ удалены.

Это шаблон с контейнером, в котором имеются блоки для вывода заголовка страницы и объекта plot_div. Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 6» и активировать ссылку «Импорт товаров, Великобритания (DOT, requests)».

Если выполнить набор вышеприведенных модулей, то получим следующий результат (рис.6.7).

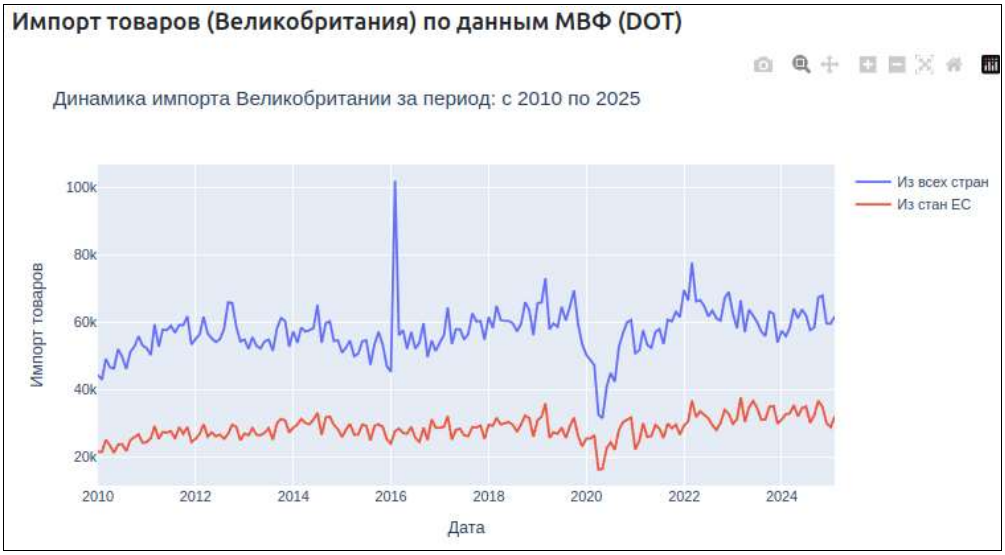


Рис. 6.7. График в шаблоне Django с данными о динамике индекса цен на импорт товаров, загруженными из БД IFS МВФ

6.13.Получение ключевых показателей API МВФ с библиотекой requests

Для загрузки тех или иных данных из API МВФ необходимо знать значения ключевых показателей, таких как коды финансово-экономических индикаторов, коды географических регионов и стран, коды периодов группировки данных и т. п. Эти коды можно получить, использую запросы получения ключевых данных с использованием библиотек requests, json и pandas.

Рассмотрим получение ключевых данных из API IMF на конкретных примерах

6.13.1.Получение данных о финансово экономических индикаторах

Через API МВФ можно получить доступ к различным экономическим показателям, указав ключевое значение соответствующего индикатора. Получить список доступных индикаторов можно по следующему по URL адресу:

https://www.imf.org/external/datamapper/api/v1/indicators.

В следующем примере создана функция, которая извлекает данные об индикаторах и создает из них набор данных Pandas (листинг 6.38).

Листинг 6.38. Модуль get_indicators.py функция извлечения данных об индикаторах

Примечание.
В демо-версии книги листинги программ удалены.

В этом программном модуле создана функция get_imf_indicators, в теле которой выполняются следующие действия.

1. Сформирован URL адрес запроса к API МВФ (объект url).
2. Выполнен запрос к API МВФ, полученные данные в формате JSON сохранены в объекте response.
3. Извлечен текст из данных формата JSON (объект ind_txt).
4. Выполнен разбор полученных данных (объект indicators).
5. Создан набор данных DataFrame (объект ind_df).
6. Объект ind_df возвращен в точку вызова функции.

Примечание.
Представленный в данном разделе программный модуль не включен в список исполняемых примеров интерактивной цифровой книги, так как он может работать только в рамках приложения Python локального компьютера, или быть вызванным из другого модуля веб приложения.

При отладке данного программного модуля в окно терминала выводится содержимое объекта df (сведения об индикаторах), и заголовки объекта df.

```
       id              ...    dataset
0      NGDP_RPCH       ...    WEO
1      NGDPD           ...    WEO
2      NGDPDPC         ...    WEO
3      PPPGDP          ...    WEO
4      PPPPC           ...    WEO
..      ...            ...     ...
128    AI_PI           ...     AIPI
129    DI              ...     AIPI
130    IEI             ...     AIPI
131    HCLMP           ...     AIPI
132    RE              ...     AIPI
```

```
[133 rows x 6 columns]
Index(['id', 'label', 'description', 'source', 'unit', 'dataset'], dtype='object')
```

Как видно из полученных результатов, из БД МВФ были загружены сведения о 133-х индикаторах финансово — экономической деятельности. Кроме того, весь набор данных сохранен в файле indicators_api_imf.xlsx. Этот файл можно открыть в Excel и найти в нем сведения о нужном индикаторе.

6.13.2.Получение данных о географических регионах и аналитических группах

Данные, получаемые через API МВФ, классифицируются по странам, географическим регионам и аналитическим группам. Получить список доступных ключей этих показателей можно по следующему URL адресу:

https://www.imf.org/external/datamapper/api/v1/<groups | regions | countries>.

В следующем примере создана функция, которая извлекает данные о ключевых показателях географических групп и создает из них набор данных Pandas (листинг 6.39).

Листинг 6.39. Модуль get_groups.py функция извлечения данных о ключевых показателях географических групп

Примечание.
В демо-версии книги листинги программ удалены.

В этом программном модуле создана функция get_imf_groups, в теле которой выполняются следующие действия.

1. Сформирован URL адрес запроса к API МВФ (объект url).
2. Выполнен запрос к API МВФ, полученные данные в формате JSON сохранены в объекте response.
3. Извлечен текст из данных формата JSON (объект resp_txt).
4. Выполнен разбор полученных данных (объект results).
5. Создан набор данных DataFrame (объект df).
6. Объект df возвращен в точку вызова функции.

При отладке данного программного модуля было выполнено три обращения к функции get_imf_groups с выводом результатов в окно терминала.

Примечание.
Представленный в данном разделе программный модуль не включен в список исполняемых примеров интерактивной цифровой книги, так как он может работать только в рамках приложения Python локального компьютера, или быть вызванным из другого модуля веб приложения.

При первом обращении к функции были запрошены идентификаторы географических групп (group):

```
------- Группы -----------
          id                      label
0       ADVEC          Advanced economies
```

```
1       AEEUEJ              Adv econ excl US, Euro, Japan
2       AFR                 Africa (Analytical)
3       AS5                 ASEAN-5
4       CEE                 Central and Eastern Europe
..        ...                         ...
124     FR_FC_ADV       Advanced Economies
125     FR_FC_EME       Emerging Market and Developing Economies
126     FR_FC_EU        European Union
127     FR_FC_NonEU     Non European Union
128     EME             Emerging market economies
[129 rows x 2 columns]
Index(['id', 'label'], dtype='object')
```

Как видно из полученных результатов, данный запрос выдал идентификаторы для 129 географических групп.

При втором обращении к функции были запрошены идентификаторы регионов (regions):

```
-------- Регионы ---------
Эти коды можно сохранить в файлах Excel и использовать для формирования запросов к
API IMF          id                      label
0    AFQ                 Africa (Region)
1    AFRREO              African Regional Economic Outlook
2    APQ                 Asia and Pacific
3    AZQ                 Australia and New Zealand
..        …                         ...
22   SPR_GD_EUR      Europe
23   SPR_GD_AFR      Africa
24   SPR_GD_CPI      Caribbean and Pacific Islands
25   SPR_GD_WHD      Western Hemisphere
26   SPR_GD_Asia     Asia

Index(['id', 'label'], dtype='object')
```

Как видно из полученных результатов, данный запрос выдал идентификаторы для 26 регионов.

При третьем обращении к функции были запрошены идентификаторы стран (countries):

```
------- Страны ------------
        id          label
0     ABW     Aruba
1     AFG     Afghanistan
2     AGO     Angola
3     AIA     Anguilla
4     ALB     Albania
..      ...             ...
236  AND      Andorra
237  IOT      British Indian Ocean Territories
238  BES      Bonaire, Sint Eustatius and Saba
239  KOS      Kosovo
240  UVK      Kosovo
```

```
[241 rows x 2 columns]
Index(['id', 'label'], dtype='object')
```

Как видно из полученных результатов, данный запрос выдал идентификаторы для 241 страны.

Эти коды можно сохранить в файлах Excel и использовать для формирования запросов к API МВФ.

6.13.3.Получение данных для заданного индикатора финансово экономической деятельности

Через API МВФ можно извлечь данные для заданного индикатора. Получить полный набор доступных данных для конкретного индикатора можно по следующему URL адресу:

https://www.imf.org/external/datamapper/api/v1/<indicator_id>

Для того чтобы извлечь нужный набор данных к этому URL адресу необходимо добавить id индикатора.

В следующем примере создана функция, которая извлекает данные для переданного ей индикатора и создает из них набор данных Pandas (листинг 6.40).

Листинг 6.40. Модуль get_indicator_data.py функция извлечения данных об индикаторе

Примечание.
В демо-версии книги листинги программ удалены.

В этом программном модуле создана функция get_imf_indicator_data, которая получает идентификатор индикатора (int_id). Затем в теле данной функции выполняются следующие действия.

1. Сформирован URL адрес запроса к API МВФ (объект url).
2. Выполнен запрос к API МВФ, полученные данные в формате JSON сохранены в объекте response.
3. Извлечен текст из данных формата JSON (объект resp_txt).
4. Выполнен разбор полученных данных (объект response_values).
5. Создан набор данных DataFrame (объект indicator_df).
6. Объект indicator_df возвращен в точку вызова функции.

При отладке данного программного модуля было выполнено обращение к функции get_imf_indicator_data с передачей ей идентификатора NGDPD (ВВП в текущих ценах) и выводом результатов в окно терминала.

Примечание.
Представленный в данном разделе программный модуль не включен в список исполняемых примеров интерактивной цифровой книги, так как он может работать только в рамках приложения Python локального компьютера, или быть вызванным из другого модуля веб приложения.

При выполнении этого программного модуля был получен следующий результат:

```
      ABW     ADVEC    AFG      AFQ  ...    YEM     ZAF    ZMB    ZWE
1980   NaN   8478.130    NaN  474.247  ...    NaN  89.412  4.246    NaN
1981   NaN   8613.564    NaN  483.634  ...    NaN  93.150  4.385    NaN
1982   NaN   8552.751    NaN  489.398  ...    NaN  85.897  4.232    NaN
1983   NaN   8900.581    NaN  472.309  ...    NaN  96.205  3.653    NaN
1984   NaN   9308.254    NaN  413.733  ...    NaN  84.838  3.003    NaN
1985   NaN   9800.652    NaN  407.292  ...    NaN  64.481  2.848    NaN
1986   NaN  11968.639    NaN  446.536  ...    NaN  73.399  1.962    NaN
. . .

[51 rows x 228 columns]
Index(['ABW', 'ADVEC', 'AFG', 'AFQ', 'AGO', 'ALB', 'AND', 'APQ', 'ARE', 'ARG',
       ...
       'WBG', 'WE', 'WEOWORLD', 'WEQ', 'WHQ', 'WSM', 'YEM', 'ZAF', 'ZMB',
       'ZWE'],
      dtype='object', length=228)
```

В данном случае был получен достаточно большой массив данных (51 строка, 228 колонок). Результирующий набор данных будет динамическим, и содержать следующие элементы:

- индексная колонка будет представлять собой год в виде строки (в строке находятся данные для заданного индикатора за указанный год);
- имена столбцов будут установлены в соответствии с идентификаторами групп;
- тип значения будет зависеть от заданного набора данных (значения для некоторых ячеек могут отсутствовать).

Ограничение запроса определенными группами и годами не является обязательным, однако при этом возвращаемый набор данных будет достаточно большим, что является не рациональным с точки зрения времени загрузки данных и передачи по каналам связи лишней информации. Для оптимизации запросов необходимо дополнительно указывать группу запрашиваемых данных и период. Однако нужно помнить, что если указать группы или период (годы), не существующие в запрошенном индикаторе, то результирующий набор данных будет пустым. Имейте в виду, что индикаторы часто включают в себя только часть доступных стран, географических регионов и аналитических групп.

6.14. Получение данных из API МВФ

Итак, мы научились извлекать из API МВФ ключевые параметры, с помощью которых можно фильтровать данные. Теперь реализуем программные модули, которые позволят загрузить только нужный набор данных и представить их в графическом виде.

6.14.1. Получение данных из API МВФ с использованием меток параметров

Через API МВФ можно извлечь необходимые данные с использованием меток (label) параметров. Получить набор отфильтрованных данных для параметров на основе их меток можно по следующему URL адресу:

https://www.imf.org/external/datamapper/api/v1/<indicator_label><group_label><period>

Здесь:

- <indicator_label> - метка индикатора;
- <group_label> - метки географических групп;
- <period> - запрашиваемый период.

Для того чтобы извлечь нужный набор данных к заголовку URL адреса необходимо добавить метку индикатора, метку группы и период.

В следующем примере создана функция, которая получает набор меток и период, а затем через API МВФ загружает нужную информацию и создает из нее набор данных Pandas (листинг 6.41).

Листинг 6.41. Модуль get_data_label.py функция извлечения данных через API МВФ с использованием меток параметров

Примечание.
В демо-версии книги листинги программ удалены.

В этом программном модуле выполнен импорт необходимых библиотек и локальных модулей, которые описаны в предыдущих разделах. Затем создана функция get_data, которая получает следующие параметры:

- lab_indic - метка индикатора;
- list_countries — список меток географических групп (список стран);
- period — период (строковый список с годами начала и конца периода).

Затем в теле данной функции выполняются следующие действия.

1. Получены данные по всем индикаторам (объект indicators).
2. Получены данные по всем географическим группам (странам, объект - countries).
3. Найдена запись для заданного индикатора (объект indicator).
4. Найдены коды стран по заданному списку стран (объект selected_countries).
5. Получен набор данных по заданным параметрам (объект data).

Обратите внимание, что для получения ключевых параметров по их меткам и загрузки данных использовались три локальные функции, описание которых приведено в предыдущих разделах. В полученном наборе данных для индексной колонки изменен тип данных на datetime:

```
data.index = pd.to_datetime(data.index, format='%Y')
```

Объект data возвращен в точку вызова функции.

При отладке данного программного модуля было выполнено обращение к функции get_data с передачей ей метки индикатора (GDP, current prices), списка меток географических групп, в данном случае стран ('United States', 'France', 'Russian Federation'), список дат запрашиваемого периода (с 2015 года по 2025год). Полученные результаты выведены в окно терминала.

Примечание.

Представленный в данном разделе программный модуль не включен в список исполняемых примеров интерактивной цифровой книги, так как он может работать только в рамках приложения Python локального компьютера, или быть вызванным из другого модуля веб приложения.

При выполнении этого программного модуля был получен следующий результат:

	FRA	RUS	USA
2015-01-01	2442.731	1356.703	18295.000
2016-01-01	2469.726	1280.648	18804.900
2017-01-01	2587.955	1575.140	19612.100
2018-01-01	2782.838	1653.005	20656.525
2019-01-01	2723.094	1695.724	21539.975
2020-01-01	2645.806	1488.118	21354.125
2021-01-01	2968.405	1828.927	23681.175
2022-01-01	2797.049	2295.527	26006.900
2023-01-01	3056.880	2059.762	27720.725
2024-01-01	3162.023	2161.205	29184.900
2025-01-01	3211.292	2076.396	30507.217

В этом наборе данных 11 строк и три колонки с данными о ВВП стран. В этом виде данные готовы для построения графиков.

Здесь в запросе использовалась только одна географическая группа (страны). Однако допускается запрашивать данные, используя разные группы в одном запросе, например, данные из США (страна), а также группа стран Центральной Америки (географический регион).

6.14.2.Получение данных из API IFS с использованием идентификаторов параметров

Через API МВФ можно извлечь данные для известных индикаторов параметров. Получить отфильтрованный набор доступных данных по известным идентификаторам параметров можно по следующему URL адресу:

https://www.imf.org/external/datamapper/api/v1/<indicator_id><group_id><period>

Здесь:

- <indicator_id> - идентификатор индикатора;
- <group_id> - идентификатор географических групп;
- <period> - запрашиваемый период.

Для того, чтобы извлечь нужный набор данных к заголовку URL адреса необходимо добавить id индикатора, id группы и период в виде списка дат.

В следующем примере создана функция, которая получает набор идентификаторов и период, а затем через API МВФ загружает нужную информацию и создает из нее набор данных Pandas (листинг 6.42).

Листинг 6.42. Модуль get_data_params.py функция извлечения данных через API МВФ с использованием идентификаторов параметров

Примечание.
В демо-версии книги листинги программ удалены.

В этом программном модуле создана функция get_imf_indicator_data, которая получает следующие параметры:

- indicator_id - идентификатор индикатора;
- group_ids=None — идентификатор географических групп (список);
- years=None — период (строковый список лет).

После этого делается проверка, получены ли данные по идентификатор групп и период. Если эти данные не были переданы, то данным параметрам присваивается пустой список.

```
if years is None:
    years = []
  if group_ids is None:
    group_ids = []
```

Затем в теле данной функции выполняются следующие действия.

1. Из фрагментов URL адресов формируется полный URL адрес запроса к API МВФ (объект url).
2. Выполнен запрос к API МВФ, полученные данные в формате JSON сохранены в объекте response.
3. Извлечен текст из полученных данных формата JSON (объект resp_txt).
4. Выполнен разбор полученных данных (объект response_values).
5. Создан набор данных DataFrame (объект indicator_df).
6. Объект indicator_df возвращен в точку вызова функции.

При отладке данного программного модуля было выполнено обращение к функции get_imf_indicator_data с передачей ей идентификатора индикатора (NGDPD - ВВП в текущих ценах), список идентификаторов географических групп (RU — Россия), список дат запрашиваемого периода (с 2015 года по 2022 год). Полученные результаты выведены в окно терминала.

Примечание.
Представленный в данном разделе программный модуль не включен в список исполняемых примеров интерактивной цифровой книги, так как он может работать только в рамках приложения Python локального компьютера, или быть вызванным из другого модуля веб приложения.

При выполнении этого программного модуля был получен следующий результат:

```
       RUS
2015  1356.703
2016  1280.648
2017  1575.140
2018  1653.005
2019  1695.724
```

```
2020  1488.118
2021  1828.927
2022  2295.527
Index(['RUS'], dtype='object')
```

В этом наборе данных 8 строк и одна колонка. В этом виде данные готовы для построения графиков. Здесь в запросе использовалась только одна географическая группа (страны). Однако допускается запрашивать данные, используя разные группы в одном запросе, например, данные из США (страна), а также группа стран Центральной Америки (географический регион).

Теперь рассмотрим, как можно построить графики на основе данных, полученных из API МВФ с использованием меток параметров и идентификаторов параметров.

6.14.3.Модули Django для визуализации данных о ВВП стран, полученных из API МВФ на основе меток параметров

Если известны значения меток параметров, то в программном коде можно сформировать запрос, указав значения этих меток в параметрах запроса. В данном примере извлекаются данные о ВВП стран через API МВФ для следующих условий:

- 'GDP, current prices' - метка индикатора (ВВП в текущих ценах);
- 'United States', 'France', 'Russian Federation' - метки стран (для стран США, Франция и Россия);
- период с 2015 по 2025 годы.

Для реализации данного примера в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 6.43).

Листинг 6.43. Ссылка активации маршрута fig_vvp_label

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем fig_vvp_label (листинг 6.44).

Листинг 6.44. Маршрут URL с именем fig_vvp_label

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление fig_vvp_label (листинг 6.45).

Листинг 6.45. Представление (view) fig_vvp_label

Примечание.
В демо-версии книги листинги программ удалены.

В этом модуле выполнен импорт необходимых библиотек и модулей. Обратите внимание, что здесь используется три локальных модуля, разработанных ранее и описанных в предыдущих разделах.
Далее выполнены следующие действия:

1. Получены данные по всем индикаторам (объект indicators).
2. Получены данные по всем географическим группам (странам, объект — countries).
3. Задана метка запрашиваемого индикатора (объект label_indic - GDP, current prices).
4. Найдена запись для заданного индикатора по его метке (объект indicator).
5. Задан список меток стран (объект list_count - ['United States', 'France', 'Russian Federation'])
6. Найдены коды (идентификаторы) стран по заданному списку меток (объект selected_countries).
7. Сформирован запрашиваемый период (объект - list_date).
8. Получен набор данных по заданным параметрам (объект df).
9. Для индексной колонки объект df задан тип данных datetime.

В учебных целях полученный набор данных выведен в окно терминала:

```
                FRA       RUS       USA
2015-01-01  2442.731  1356.703  18295.000
2016-01-01  2469.726  1280.648  18804.900
2017-01-01  2587.955  1575.140  19612.100
2018-01-01  2782.838  1653.005  20656.525
2019-01-01  2723.094  1695.724  21539.975
2020-01-01  2645.806  1488.118  21354.125
2021-01-01  2968.405  1828.927  23681.175
2022-01-01  2797.049  2295.527  26006.900
2023-01-01  3056.880  2059.762  27720.725
2024-01-01  3162.023  2161.205  29184.900
2025-01-01  3211.292  2076.396  30507.217
```

При развертывании приложения строку вывода результатов в окно терминала можно закомментировать.

На следующем шаге создан графика (объект fig) с описанием макета графика:

```
fig = go.Figure()
   fig.add_trace(go.Scatter(x=df.index, y=df['USA'], name='США'))
   fig.add_trace(go.Scatter(x=df.index, y=df['FRA'], name='Франция'))
   fig.add_trace(go.Scatter(x=df.index, y=df['RUS'], name='Россия'))
   fig.update_layout(title='Сравнение ВВП стран',
                     xaxis_title='Дата',
                     yaxis_title='ВВП, текущие цены')
```

После этого график приведен к формату, в котором он может быть показан на HTML странице (объект plot_div) и создана строка с заголовком страницы (title_str). Полученный график (plot_div) с заголовком страницы через контекстную переменную context передается в шаблон fig_plotly.html (листинг 6.46).

Листинг 6.46. Шаблон fig_plotly.html

Примечание.
В демо-версии книги листинги программ удалены.

Это шаблон с контейнером, в котором имеются блоки для вывода заголовка страницы и объекта plot_div. Теперь можно проверить работу данных модулей.

> ***Примечание.***
> В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 6» и активировать ссылку «ВВП для группы стран (по меткам параметров)».

Если выполнить набор вышеприведенных модулей, то получим следующий результат (рис.6.8).



Рис. 6.8. График в шаблоне Django с данными о динамике ВВП стран, загруженными из API МВФ на основе меток параметров

6.14.4.Модули Django для визуализации данных о ВВП стран, полученных из API МВФ на основе идентификаторов параметров

Если известны значения идентификаторов параметров, то в программном коде можно сформировать запрос, указав значения этих идентификаторов в параметрах запроса. При работе с идентификаторами параметров программный код получается значительно проще и лучше читается.
В данном примере извлекаются данные о ВВП стран через API МВФ для следующих условий:

- NGDPD - id индикатора;
- ['RUS', 'DEU', 'FRA'] - id стран (для стран Россия, Германия, Франция);
- период с 2015 по 2025 годы.

Для реализации данного примера в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 6.47).

Листинг 6.47. Ссылка активации маршрута fig_vvp_id

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем fig_vvp_id (листинг 6.48).

Листинг 6.48. Маршрут URL с именем fig_vvp_id

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление fig_vvp_id (листинг 6.49).

Листинг 6.49. Представление (view) fig_vvp_id

Примечание.
В демо-версии книги листинги программ удалены.

В этом модуле выполнен импорт необходимых библиотек и модулей. Обратите внимание, что здесь используется локальный модуль get_data_params, разработанный ранее и описанный в предыдущих разделах.
Далее выполнено формирование параметров запроса:

- ind - идентификатор индикатора;
- list_group — список идентификаторов географических групп (страны);
- list_years — период (список, годы запрашиваемого периода).

В учебных целях полученный набор данных выведен в окно терминала:

```
      DEU       FRA       RUS
2015  3423.915  2442.731  1356.703
2016  3536.808  2469.726  1280.648
2017  3761.764  2587.955  1575.140
2018  4053.846  2782.838  1653.005
2019  3957.645  2723.094  1695.724
2020  3936.989  2645.806  1488.118
2021  4351.188  2968.405  1828.927
2022  4166.872  2797.049  2295.527
2023  4527.009  3056.880  2059.762
2024  4658.526  3162.023  2161.205
2025  4744.804  3211.292  2076.396
```

При развертывании приложения строку вывода результатов в окно терминала можно закомментировать.

На следующем шаге создан графика (объект fig) с описанием макета графика:

```
fig = go.Figure()
  fig.add_trace(go.Scatter(x=df.index, y=df['USA'], name='США'))
  fig.add_trace(go.Scatter(x=df.index, y=df['FRA'], name='Франция'))
  fig.add_trace(go.Scatter(x=df.index, y=df['RUS'], name='Россия'))
  fig.update_layout(title='Сравнение ВВП стран',
                    xaxis_title='Дата',
                    yaxis_title='ВВП, текущие цены')
```

После этого график приведен к формату, в котором он может быть показан на HTML странице (объект plot_div) и создана строка с заголовком страницы (title_str). Полученный график (plot_div) с заголовком страницы через контекстную переменную context передается в шаблон fig_plotly.html (листинг 6.50).

Листинг 6.50. Шаблон fig_plotly.html

Примечание.
В демо-версии книги листинги программ удалены.

Это шаблон с контейнером, в котором имеются блоки для вывода заголовка страницы и объекта plot_div. Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 6» и активировать ссылку «ВВП для группы стран (по ID параметров)».

Если выполнить набор вышеприведенных модулей, то получим следующий результат (рис.6.9).



Рис. 6.9. График в шаблоне Django с данными о динамике ВВП стран, загруженными из API МВФ на основе идентификаторов параметров

Примечание.
Для запроса ежедневных данных с помощью API IMF необходимо указать параметр freq как «D» в сочетании с другими параметрами запроса, например, кодом индикатора и областью данных. Например, запрос может выглядеть так: CompactData/IFS/D.GB.PMP_IX. Обратите внимание, что не все данные, получаемые через API МВФ, могут быть получены в разрезе дней.

6.15. Краткие итоги

Итак, мы познакомились с некоторыми базовыми методами загрузки и «очистки» данных из различных БД МВФ и других источников данных, а также с визуализацией полученной информации. Загружаемая финансово экономическая информация, как правило, представляет собой временные ряды, которые служат основой для решения задач прогнозирования. Прогнозирование основано на применении статистических моделей, которые с той или иной точностью описывают временной ряд. Для того, чтобы подобрать статистическую модель под конкретный временной ряд, нужно выполнить его анализ на полноту данных, наличие тренда и сезонности, наличие выбросов или пропусков данных.

Исходя из этого, в следующей главе будут рассмотрены методы и инструментарий, которые используемого для анализа данных.

Глава 7. Анализ и очистка данных временных рядов

Анализ временных рядов — это метод статистического анализа данных, собранных в хронологическом порядке. Временные ряды используются для выявления тенденций, сезонных колебаний, которые могут помочь в прогнозировании будущих значений. Важно понимать, что временные ряды могут включать в себя как регулярные, так и нерегулярные данные, и каждый тип данных требует своего подхода к анализу.

Анализ временных рядов использовался задолго до появления вычислительной техники, Люди всегда пытались предсказать погоду на основе многолетних наблюдений, или востребованность того или иного товара в зависимости от сезона. С развитием вычислительной техники и статистических методов, анализ временных рядов стал более точным и доступным. Сегодня, благодаря современным инструментам и библиотекам, таким как Python, анализ временных рядов стал неотъемлемой частью многих областей науки и бизнеса.

Из материалов этой главы вы узнаете:

- об основных методах анализа временных рядов;
- какие компоненты содержит временной ряд;
- как определить скользящее среднее временного ряда;
- как можно оценить стационарность временного ряда;
- какими методами можно привести временной ряд к стационарности;
- как можно изменить частоту дискретизации временного ряда;
- как можно восстановить непрерывность временного ряда.

7.1.Методы анализа временных рядов

Основными методами анализа временных рядов являются:

- декомпозиция временного ряда;
- определение скользящего среднего;
- оценка стационарности;
- определение автокорреляции;
- экспоненциальное сглаживание;
- использование статистических моделей.

Анализ временных рядов позволяет выявить ряд характеристик и особенностей временного ряда, которые нужно будет учитывать при решении задач прогнозирования. Кроме того, структурный анализ временного ряда позволяет выявить его полноту и необходимость «очистки», которая включает добавление пропущенных значений, удаление не характерных выбросов, повышение или понижение частоты дискретизации данных. Очищенные данные обеспечивают получение более точных прогнозов.

Рассмотрим наиболее популярные инструменты, которые используются для анализа временных рядов.

Python. Python является одним из наиболее популярных языков программирования для анализа временных рядов благодаря богатому набору библиотек и инструментов. Python предоставляет широкий спектр инструментов для работы с временными рядами, от простых функций до сложных моделей.

Pandas. Библиотека Pandas является наиболее популярным инструментом для работы с временными рядами и табличными данными. Pandas предоставляет удобные функции для манипуляции и анализа временных рядов, табличного и графического представления данных.

Statsmodels. Библиотека для статистического моделирования Statsmodels, включает различные модели AR, MA, ARMA, ARIMA, SARIMA и экспоненциальное сглаживание. Данная библиотека предоставляет мощные инструменты для анализа временных рядов, статистического моделирования и прогнозирования.

Prophet. Библиотека **Prophet**, разработанная компанией Facebook, предназначена для прогнозирования временных рядов с учетом сезонности и праздников. Prophet позволяет легко создавать и настраивать модели для решения задач прогнозирования на базе временных рядов.

Итак, приступим созданию программных модулей для анализа временных рядов, и начнем с выявления основных составляющих временного ряда.

7.2. Декомпозиция временного ряда временного ряда

Анализ временных рядов требует понимания их основных составляющих. Каждый временной ряд можно разложить на несколько ключевых компонентов, которые помогают лучше понять структуру данных и сделать более точные прогнозы. Временной ряд может иметь следующие компоненты:

- тренд (Trend);
- сезонность (Seasonal);
- цикличность (Cyclical);
- случайность (Irregular).

Тренд. Тренд отражает долгосрочную тенденцию изменения значений временного ряда. Он показывает общее направление движения данных: возрастающее, убывающее или стабильное. Тренд может быть линейным или нелинейным, и его выделение важно для анализа общей динамики.

Тренды могут быть следующих видов:

- Линейный тренд: когда значения увеличиваются или уменьшаются с постоянной скоростью.
- Экспоненциальный тренд: когда скорость роста или падения ускоряется со временем.

Сезонность. Сезонность — это повторяющиеся и предсказуемые колебания значений временного ряда, связанные с определенными периодами времени (день, неделя, месяц). Сезонные колебания часто обусловлены внешними факторами, такими как погодные условия, праздничные периоды или изменение потребительских предпочтений.

Примеры сезонности:

- Рост продаж мороженого или прохладительных напитков в летние месяцы;
- Увеличение количества пассажиров в летние периоды.

Цикличность. Цикличность характеризуется более долгосрочными колебаниями значений, которые не имеют строго определенного периода, в отличие от сезонности. Циклы могут быть вызваны экономическими условиями, деловыми циклами или другими факторами, влияющими на показатели на протяжении нескольких лет. Анализ циклов помогает понять влияние глобальных макроэкономических факторов на временной ряд.

Примеры цикличности:

- Экономические кризисы, которые происходят каждые 7-10 лет;
- Периоды подъема и спада на фондовых рынках.

Случайность. Случайная компонента — это остаток временного ряда после устранения тренда, сезонности и цикличности. Она представляет собой шум или непредсказуемые колебания, вызванные случайными событиями. Случайность невозможно предсказать или моделировать, но она важна для понимания общей структуры данных и улучшения точности прогнозов.

Примеры случайности:

- Неожиданные изменения курса валют из-за внезапных политических событий;
- Колебания цен на акции в результате случайных новостей или слухов.

В Python в библиотеке statsmodels есть функция seasonal_decompose(), с помощью которой можно выделить и визуализировать компоненты временного ряда. Для установки данной библиотеки можно использовать следующую команду:

```
pip install statsmodels
```

Рассмотрим пример, в котором выявим составляющие временного ряда с данными о динамике изменения пассажиропотока за заданный период на основе данных, загруженных из файла CSV. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 7.1).

Листинг 7.1. Ссылка активации маршрута pass_decompose

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем pass_decompose (листинг 7.2).

Листинг 7.2. Маршрут URL с именем pass_decompose

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление pass_decompose (листинг 7.3).

Листинг 7.3. Представление (view) pass_decompose

Примечание.
В демо-версии книги листинги программ удалены.

В этом модуле выполнен импорт необходимых библиотек и сформирован путь к CSV файлу с именем «passengers.csv» (объект path_csv). Это тестовый временной ряд, в основе которого лежат реальные данные о пассажиропотоках. Он имеет явно выраженный тренд и сезонность и достаточно часто используется в учебных целях.

Затем данные из CSV файла загружены в объект df_pass и первые пять записей выведены в окно терминала:

```
   Month  Passengers
0  2015-01         145
1  2015-02         150
2  2015-03         178
3  2015-04         163
4  2015-05         172
```

Как видно из полученных результатов, колонка Month имеет формат «год-месяц», что не совсем удобно для временного ряда. Поэтому далее эта колонка трансформирована в индексную колонку с форматом datetime:

```
df_pass.set_index('Month', inplace=True)
df_pass.index = pd.to_datetime(df_pass.index)
```

Первые записи обновленного набора данных выведены в окно терминала:

```
            Passengers
Month
2015-01-01         145
2015-02-01         150
2015-03-01         178
2015-04-01         163
2015-05-01         172
```

Теперь мы имеет временной ряд в виде традиционного набора данных DataFrame, который можем анализировать с использованием библиотеки statsmodels.

Далее полученный набор данных обработан функциями декомпозиции библиотеки statsmodels:

```
decompose = seasonal_decompose(df_pass, model='additive', period=12)
observed = decompose.observed.dropna()
trend = decompose.trend.dropna()
seasonal = decompose.seasonal.dropna()
resid = decompose.resid.dropna()
```

В созданных объектах получены следующие компоненты временного ряда:

- observed – наблюдения (исходный временной ряд);
- trend – тренд;
- seasonal – сезонность;
- resid – случайность (или остаток после удаления тренда и сезонности).

Затем с использованием функции make_subplots создано сетка графиков (объект fig), состоящая из двух строк и двух столбцов. В каждую ячейку этой сетки вставлен свой график:

```
fig.add_trace(go.Scatter(x=observed.index, y=observed, name="Наблюдения"), row=1, col=1)
fig.add_trace(go.Scatter(x=trend.index, y=trend, name="Тренд"), row=1, col=2)
fig.add_trace(go.Scatter(x=seasonal.index, y=seasonal, name="Сезонность"), row=2, col=1)
fig.add_trace(go.Scatter(x=resid.index, y=resid, name="Нерегулярность"), row=2, col=2)
```

На следующем шаге к макету графика добавлен заголовок и изменена его высота.

После этого график приведен к формату, в котором он может быть показан на HTML странице (объект plot_div) и создана строка с заголовком страницы (title_str). Полученный график (plot_div) с заголовком страницы через контекстную переменную context передается в шаблон fig_plotly.html (листинг 7.4).

Листинг 7.4. Шаблон fig_plotly.html

Примечание.
В демо-версии книги листинги программ удалены.

Это достаточно простой шаблон с контейнером, в котором выведен заголовок страницы и объект plot_div (график). Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 7» и активировать ссылку «Декомпозиция временного ряда».

Если выполнить набор вышеприведенных модулей, то получим следующий результат (рис.7.1).

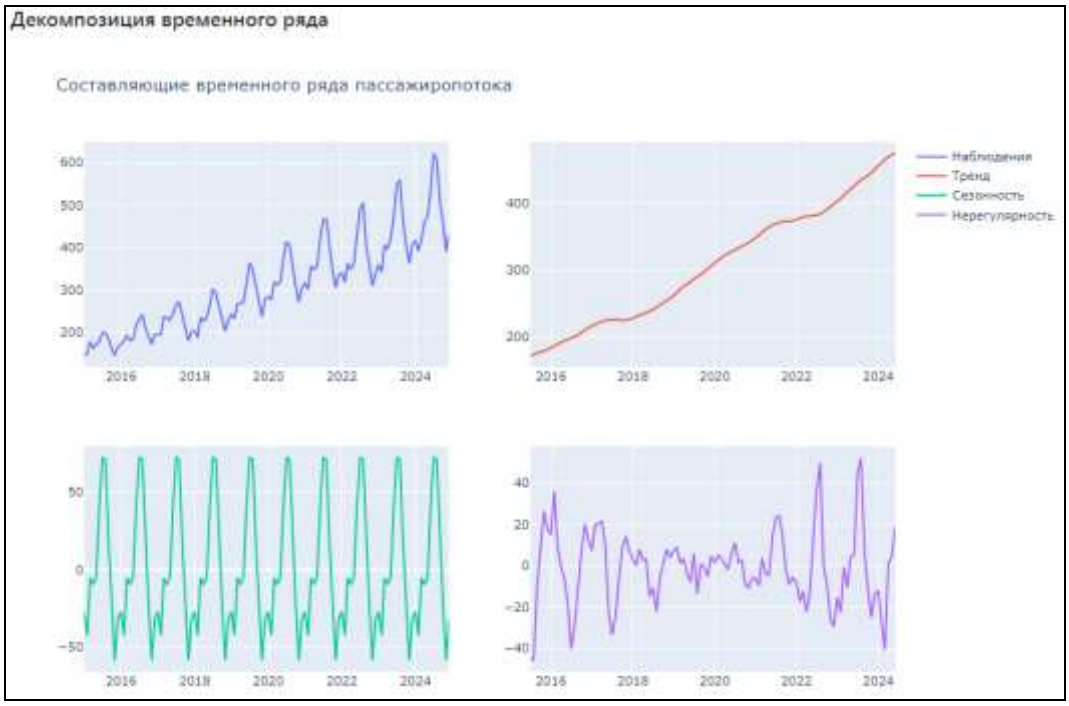


Рис. 7.1. Сетка графиков с декомпозицией временного ряда о пассажиропотоках

Как видно из полученных результатов, это временной ряд имеет возрастающий тренд и явно выраженную сезонность.

7.3.Определение скользящего среднего временного ряда

Скользящая средняя (Moving Average, или модель MA) - это математический метод сглаживания временных рядов путем усреднения значений за определенный период. По сути, это фильтр, отсекающий краткосрочные колебания и выявляющий долгосрочные тенденции в данных. У данной модели следующий принцип работы: берется фиксированное число последовательных точек данных (окно), вычисляется их среднее значение, затем окно «скользит» вперед — исключается самая старая точка, добавляется новая, и среднее пересчитывается. Скользящее среднее, также известное как скользящее среднее арифметическое, помогает уменьшить шум и выделить значимые закономерности путем усреднения точек данных за определенный период.

В библиотеке Pandas имеется возможность расчета следующих вариантов скользящего среднего:

- простое скользящее среднее (SMA) – метод rolling;
- экспоненциальное скользящее среднее (EMA) – метод ewm;
- кумулятивное скользящее среднее (CMA) – метод expanding.

Рассмотрим на примере расчет скользящего среднего для пассажиропотока. Так как этот временной ряд имеет сезонные месячные колебания (а в году 12 месяцев), то для него будет использоваться скользящее окно 12. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 7.5).

Листинг 7.5. Ссылка активации маршрута pass_mean

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем pass_mean (листинг 7.6).

Листинг 7.6. Маршрут URL с именем pass_mean

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление pass_mean (листинг 7.7).

Листинг 7.7. Представление (view) pass_mean

Примечание.
В демо-версии книги листинги программ удалены.

В этом модуле выполнен импорт необходимых библиотек и сформирован путь к CSV файлу с именем «passengers.csv» (объект path_csv). Это тестовый временной ряд, в основе которого лежат реальные данные о пассажиропотоках. Он имеет явно выраженный тренд и сезонность и достаточно часто используется в учебных целях.

Далее колонка Month трансформирована в индексную колонку с форматом datetime:

```
df.set_index('Month', inplace=True)
df.index = pd.to_datetime(df.index)
```

Теперь мы имеет временной ряд в виде традиционного набора данных DataFrame, который можем анализировать с использованием библиотеки Pandas.

Далее полученный набор данных обработан функциями библиотеки Pandas для расчета трех вариантов скользящего среднего:

```
df['SMA'] = df['Passengers'].rolling(window=12).mean()
df['EMA'] = df['Passengers'].ewm(span=12).mean()
df['CMA'] = df['Passengers'].expanding().mean()
```

В этих инструкциях к набору данных df добавлены следующие колонки со значениями скользящих средних:

- df['SMA'] - простое скользящее среднее (SMA);
- df['EMA'] - экспоненциальное скользящее среднее (EMA);
- df['CMA'] - кумулятивное скользящее среднее (CMA).

Затем создан макет графиков (объект fig), и к этому макету добавлены четыре линии:

```
fig.add_trace(go.Scatter(x=df.index, y=df['Passengers'], name="Пассажиропоток"))
fig.add_trace(go.Scatter(x=df.index, y=df['SMA'], name="Простое-SMA"))
fig.add_trace(go.Scatter(x=df.index, y=df['EMA'], name="Экспоненциальное-EMA"))
fig.add_trace(go.Scatter(x=df.index, y=df['CMA'], name="Кумулятивное-CMA"))
```

На следующем шаге к макету графика добавлен заголовок и подписи к осям.

После этого график приведен к формату, в котором он может быть показан на HTML странице (объект plot_div) и создана строка с заголовком страницы (title_str). Полученный график (plot_div) с заголовком страницы через контекстную переменную context передается в шаблон fig_plotly.html (листинг 7.8).

Листинг 7.8. Шаблон fig_plotly.html

Примечание.
В демо-версии книги листинги программ удалены.

Это достаточно простой шаблон с контейнером, в котором выведен заголовок страниц и объект plot_div (график). Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 7» и активировать ссылку «Скользящее среднее временного ряда».

Если выполнить набор вышеприведенных модулей, то получим следующий результат (рис.7.2).



Рис. 7.2. Скользящие средние временного ряда с данными о пассажиропотоках

Как видно из данного рисунка, каждый вид скользящего среднего по своему характеризует усредненную динамику изменения пассажиропотока. Однако все три скользящие средние показывают устойчивую тенденцию роста пассажиропотока.

7.4.Стационарность временного ряда

Стационарность (stationarity) временного ряда означает, что такие компоненты как тренд и сезонность отсутствуют. Говоря более точно, среднее значение и дисперсия не меняются со смещением времени. Понимание того, что данные являются стационарными, важно для последующего моделирования и прогнозирования. Стационарность временного ряда можно оценить визуально, но для более точной оценки стационарности можно применить тест Дики-Фуллера (Dickey-Fuller test). В данном тесте для оценки стационарности используется пороговое значение, равное 0,05 (или 5%). Если значение данного теста не превышает 0,05, то временной ряд является стационарным, в противном случае он не является стационарным. Для оценки стационарности временного ряда в pandas используется функция adfuller.

7.4.1.Оценка стационарности временного ряда (пассажиропотоки)

Напишем программный код, позволяющий оценить стационарность временного ряда с данными о пассажиропотоках тестом Дики-Фуллера (Dickey-Fuller test). Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 7.9).

Листинг 7.9. Ссылка активации маршрута stat_pass

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем stat_pass (листинг 7.10).

Листинг 7.10. Маршрут URL с именем stat_pass

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление stat_pass (листинг 7.11).

Листинг 7.11. Представление (view) stat_pass

Примечание.
В демо-версии книги листинги программ удалены.

В этом модуле выполнен импорт необходимых библиотек и сформирован путь к CSV файлу с именем «passengers.csv» (объект path_csv). Это тестовый временной ряд, в основе которого лежат реальные данные о пассажиропотоках. Он имеет явно выраженный тренд и сезонность и достаточно часто используется в учебных целях.

Далее колонка Month трансформирована в индексную колонку с форматом datetime:

```
df.set_index('Month', inplace=True)
df.index = pd.to_datetime(df.index)
```

Теперь мы имеет временной ряд в виде традиционного набора данных DataFrame, который можем привести к графическому виду (объект fig):

```
fig = go.Figure()
fig.add_trace(go.Scatter(x=df.index, y=df['Passengers']))
fig.update_layout(title='Динамика изменения пассажиропотока',
                  xaxis_title="Дата",
                  yaxis_title="Пассажиропоток, тыс.чел.",
                  autosize=False,
                  width=800,
                  height=600,
                  hoverlabel=dict(font_size=15))
```

Далее полученный набор данных обработан модулем adfuller библиотеки statsmodels и получено значение теста Дики-Фуллера:

```
adf_test = adfuller(df['Passengers'])
p_test = adf_test[1]
```

Затем значение теста Дики-Фуллера сопоставлено с критической величиной (0.05) и в текстовую переменную stat_txt занесено заключение о стационарности временного ряда.

После этого график приведен к формату, в котором он может быть показан на HTML странице (объект plot_div) и создана строка с заголовком страницы (title_str). Полученный график (plot_div) с заголовком страницы и параметрами теста через контекстную переменную context передается в шаблон fig_stat.html (листинг 7.12).

Листинг 7.12. Шаблон fig_stat.html

> ***Примечание.***
> В демо-версии книги листинги программ удалены.

Это достаточно простой шаблон с контейнером, в котором выведен объекты plot_div (график) и текстовые объекты. Теперь можно проверить работу данных модулей.

> ***Примечание.***
> В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 7» и активировать ссылку «Стационарность данных пассажиропотока».

Если выполнить набор вышеприведенных модулей, то получим следующий результат (рис. 7.3).



Рис. 7.3. Оценка стационарности временного ряда с данными о пассажиропотоках

Из данных результатов видно, что и визуально, и по данным теста Дики-Фуллера временной ряд пассажиропотока не является стационарным. Данный временной ряд имеет явно выраженный тренд и сезонность. Перед решением задач прогнозирования такой временной ряд необходимо приводить к стационарности. Как это делается, будет показано в следующих разделах.

7.4.2.Оценка стационарности временного ряда (объем продажи нефти)

Напишем программный код, позволяющий оценить стационарность временного ряда с данными об объемах продажи нефти тестом Дики-Фуллера (Dickey-Fuller test). В этом примере будут использоваться реальные данные, получаемые режиме on-line через API Yahoo Finanace. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 7.13).

Листинг 7.13. Ссылка активации маршрута stat_oil

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем stat_oil (листинг 7.14).

Листинг 7.14. Маршрут URL с именем stat_oil

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление stat_oil (листинг 7.15).

Листинг 7.15. Представление (view) stat_oil

Примечание.
В демо-версии книги листинги программ удалены.

В этом модуле выполнен импорт необходимых библиотек, создан тикер для получения данных о продаже нефти на мировых рынках (объект tick)

```
tick = 'CL=F
ticker = yf.Ticker(tick)
```

После этого загружены данные о торгах на нефть за последние 5 лет, начиная с текущей даты (объект df):

```
try:
    df = ticker.history(period='5y')
except Exception:
    title_str = 'Ошибка загрузки данных из Yahoo Finance, повторите загрузку позже!'
    context = {'title_str': title_str}
    return render(request, "api_finance/fig_stat.html", context)
```

Примечание.
Инструкция загрузки данных df = ticker.history(period='5y') заключена в блок try…except. Это сделано для того, чтобы избежать ошибки программы в тех случаях, когда произойдет сбой загрузки данных. Это может случиться в случаях большой нагрузки на сервер Yahoo Finance, или если изменился доступ к данным через API.

Затем из объектов df выбрана одна колонка (Volume) с данными об объемах проданной нефти:

```
df = df[['Volume']]
```

Теперь мы имеет временной ряд в виде традиционного набора данных DataFrame, который можем привести к графическому виду (объект fig):

```
fig = go.Figure()
fig.add_trace(go.Scatter(x=df.index, y=df['Volume'], name="Объем продаж"))
fig.update_layout(title='Динамика объема продаж нефти',
```

```
        xaxis_title="Дата",
        yaxis_title="Объем продаж",
        autosize=False,
        width=800,
        height=600,
        hoverlabel=dict(font_size=15))
```

Далее полученный набор данных обработан модулем adfuller библиотеки statsmodels и получено значение теста Дики-Фуллера:

```
adf_test = adfuller(df['Volume'])
p_test = adf_test[1]
```

Затем значение теста Дики-Фуллера сопоставлено с критической величиной (0.05) и в текстовую переменную stat_txt занесено заключение о стационарности временного ряда.

После этого график приведен к формату, в котором он может быть показан на HTML странице (объект plot_div) и создана строка с заголовком страницы (title_str). Полученный график (plot_div) с заголовком страницы и параметрами теста через контекстную переменную context передается в шаблон fig_stat.html (листинг 7.16).

Листинг 7.16. Шаблон fig_stat.html

Примечание.
В демо-версии книги листинги программ удалены.

Это достаточно простой шаблон с контейнером, в котором выведен объекты plot_div (график) и текстовые объекты. Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 7» и активировать ссылку «Стационарность данных о продаже нефти».

Если выполнить набор вышеприведенных модулей, то получим следующий результат (рис. 7.4).



Рис. 7.4. Оценка стационарности временного ряда об объемах продажи нефти

Как видно из данного рисунка, и визуально, и по данным теста Дики-Фуллера временной ряд объемов продаж нефти является стационарным.

> ***Примечание.***
> Так как эти данные загружаются в режиме реального времени, то при следующих загрузках данных график может иметь другой вид, другое значение теста Дики-Фуллера, и даже временной ряд может потерять стационарность.

7.5.Приведение временных рядов к стационарности

И тренд, и сезонность обычно присутствуют в большинстве данных временных рядов реального мира. Когда мы хотим сделать прогноз с временными рядами, нам нужен стационарный временной ряд.

Стационарный временной ряд — это набор данных временного ряда, в котором отсутствует информация о тренде или сезонности. Стационарный временной ряд — это ряд с

постоянным средним значением, постоянной дисперсией и постоянной автокорреляцией. Чтобы сделать временной ряд стационарным, нужно удалить тренды и сезонность из временного ряда, чтобы можно было использовать его с моделями прогнозирования. Теперь, когда мы знакомы с концепцией стационарности, можем перейти к рассмотрению методов приведения временных рядов к стационарности.

7.5.1.Приведение временных рядов к стационарности путем дифференцирования

Дифференцирование — это процесс вычисления разностей между последовательными значениями временного ряда.

$Y't = Y\_t - Y\_t{-}1$

Дифференцирование обеспечивает устранение тренда и позволяет сделать ряд стационарным. При дифференцировании выполняется преобразование исходного ряда в новый временной ряд, у которого стабилизируется среднее значение ряда. В библиотеке pandas для дифференцирования первого порядка (вычитание каждого значения в ряду из предыдущего) используется метод diff().

В предыдущих разделах был использован временной ряд о пассажиропотоках, и было выяснено, что он не является стационарным. Реализуем программный модуль, в котором попытаемся привести его к стационарности путем дифференцирования. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 7.17).

Листинг 7.17. Ссылка активации маршрута stat_diff_1

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем stat_diff_1 (листинг 7.18).

Листинг 7.18. Маршрут URL с именем stat_diff_1

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление stat_diff_1 (листинг 7.19).

Листинг 7.19. Представление (view) stat_diff_1

Примечание.
В демо-версии книги листинги программ удалены.

В этом модуле выполнен импорт необходимых библиотек и сформирован путь к CSV файлу с именем «passengers.csv» (объект path_csv). Это тестовый временной ряд, в основе

которого лежат реальные данные о пассажиропотоках. Он имеет явно выраженный тренд и сезонность и достаточно часто используется в учебных целях.

Далее колонка Month трансформирована в индексную колонку с форматом datetime:

```
df.set_index('Month', inplace=True)
df.index = pd.to_datetime(df.index)
```

В следующей инструкции выполнено дифференцирование временного ряда:

```
df['Pass_diff'] = df['Passengers'].diff()
```

Обновленный набор данных выведен в окно терминала:

```
            Passengers  Pass_diff
Month
2015-01-01         145        NaN
2015-02-01         150        5.0
2015-03-01         178       28.0
2015-04-01         163      -15.0
2015-05-01         172        9.0
...                ...        ...
2024-08-01         606      -16.0
2024-09-01         508      -98.0
2024-10-01         461      -47.0
2024-11-01         390      -71.0
2024-12-01         432       42.0
```

Как видно из полученных результатов, к набору данных была добавлено новая колонка с именем «Pass_diff», которая в первой строке не имеет значения (NaN).

Теперь мы имеем возможность дифференцированный временной ряд привести к графическому виду (объект fig):

```
fig = go.Figure()
fig.add_trace(go.Scatter(x=df.index, y=df['Pass_diff'], name="Пассажиропоток"))
fig.update_layout(title='Приведение временного ряда к стационарности -
дифференцирование',
                  xaxis_title="Дата",
                  yaxis_title="Пассажиропоток, тыс.чел.",
                  autosize=False,
                  width=800,
                  height=600,
                  hoverlabel=dict(font_size=20))
```

После дифференцирования первая строка новой колонки Pass_diff будет иметь значение NaN, что приведет к ошибке при проверке временного ряда на стационарность. Поэтому, для исключения данной ошибки, перед выполнением теста на стационарность значение NaN заменено на 0.

```
df['Pass_diff'] = df['Pass_diff'].fillna(0)
```

Далее полученный набор данных обработан модулем adfuller библиотеки statsmodels и получено значение теста Дики-Фуллера:

```
adf_test = adfuller(df['Pass_diff'])
p_test = adf_test[1]
```

Затем значение теста Дики-Фуллера сопоставлено с критической величиной (0.05) и в текстовую переменную stat_txt занесено заключение о стационарности временного ряда.

После этого график приведен к формату, в котором он может быть показан на HTML странице (объект plot_div) и создана строка с заголовком страницы (title_str). Полученный график (plot_div) с заголовком страницы и параметрами теста через контекстную переменную context передается в шаблон fig_stat.html (листинг 7.20).

Листинг 7.20. Шаблон fig_stat.html

Примечание.
В демо-версии книги листинги программ удалены.

Это достаточно простой шаблон с контейнером, в котором выведен объекты plot_div (график) и текстовые объекты. Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 7» и активировать ссылку «Дифференцирование временного ряда».

Если выполнить набор вышеприведенных модулей, то получим следующий результат (рис. 7.5).



Рис. 7.5. Дифференцирование временного ряда

Из данного графика видно, что из временного ряда было удалено влияние тренда, а значение теста Дики-Фуллера уменьшилось до 0.12, но еще превышает критический порог в 0.05, поэтому временной ряд все еще не является стационарным. На полученном графике видно, что амплитуда разброса пассажиропотока со временем увеличивается. Если дифференцирование не приводит к стационарности временного ряда, то к нему можно применить двойное дифференцирование.

7.5.2.Приведение временных рядов к стационарности путем двойного дифференцирования

Если после дифференцирования в данных остается сезонность или тренд, то можно провести еще одно дифференцирование, но уже с новым рядом. При необходимости такое преобразование можно повторять несколько раз. Такие шаги последовательно приводят временной ряд к стационарному виду, что обеспечивает его использование для прогнозирования.

Реализуем программный модуль, в котором попытаемся привести временной ряд с данными о пассажиропотоках к стационарности путем двойного дифференцирования. Для этого в

одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 7.21).

Листинг 7.21. Ссылка активации маршрута stat_diff_2

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем stat_diff_2 (листинг 7.22).

Листинг 7.22. Маршрут URL с именем stat_diff_2

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление stat_diff_2 (листинг 7.23).

Листинг 7.23. Представление (view) stat_diff_2

Примечание.
В демо-версии книги листинги программ удалены.

Этот программный код практически не отличается от аналогичного модуля предыдущего раздела, поэтому мы не будем приводить его подробное описание. Единственное отличие заключается в инструкции дифференцирования временного ряда:

```
df['Pass_diff'] = df['Passengers'].diff().diff()
```

Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 7» и активировать ссылку «Двойное дифференцирование временного ряда».

Если выполнить набор вышеприведенных модулей, то получим следующий результат (рис. 7.6).



Рис. 7.6. Двойное дифференцирование временного ряда

Из данного рисунка, что из временного ряда были удалены тренд и сезонность, а значение теста Дики-Фуллера значительно меньше критического порога 0,05 (практически равно нулю), поэтому временной ряд является стационарным. На полученном графике и визуально видно, что и тренд, и сезонность практически отсутствуют.

7.5.3.Приведение временных рядов к стационарности путем логарифмирования с дифференцированием

Если временной ряд имеет экспоненциальный рост или изменяющуюся дисперсию, то к его данным можно применить логарифмирование. Это стабилизирует дисперсию и может помочь удалить экспоненциальные тренды. Логарифмирование данных временного ряда может снизить его вариативность и привести стационарности. Для логарифмирования временного ряда можно использовать библиотеку Numpy. После логарифмирования к временному ряду можно применить дифференцирование

Реализуем программный модуль, в котором попытаемся привести временной ряд с данными о пассажиропотоках к стационарности путем логарифмирования. Для этого в одном из

шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 7.24).

Листинг 7.24. Ссылка активации маршрута stat_log

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем stat_log (листинг 7.25).

Листинг 7.25. Маршрут URL с именем stat_log

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление stat_log (листинг 7.26).

Листинг 7.26. Представление (view) stat_log

Примечание.
В демо-версии книги листинги программ удалены.

В этом программном модуле выполнен импорт необходимых библиотек, включая numpy, а в остальном он практически не отличается от аналогичного модуля предыдущих двух разделов, поэтому мы не будем приводить его подробное описание. Единственное отличие заключается в инструкции логарифмировании временного ряда:

```
df['Pass_log'] = np.log(df['Passengers'])
df['Pass_log_diff'] = df['Pass_log'] - df['Pass_log'].shift(1)
```

На первом шаге выполнено логарифмирование исходных данных (добавлена колонка «Pass_log»), а затем применено дифференцирование к логарифмически преобразованному временному ряду, сдвинув его значение на 1 период и вычтя его из исходного логарифмически преобразованного временного ряда (добавлена колонка «Pass_log_diff»).

Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 7» и активировать ссылку «Логарифмирование временного ряда».

Если выполнить набор вышеприведенных модулей, то получим следующий результат (рис. 7.7).



Рис. 7.7. Логарифмирование временного ряда

Из данного графика видно, что из временного ряда было удалено влияние тренда и сезонности, а значение теста Дики-Фуллера значительно меньше критического порога 0,05, поэтому временной ряд является стационарным. Из графика даже визуально видно, что влияние тренда и сезонности значительно уменьшилось.

Для приведения временных рядов к стационарности можно использовать другие методы и алгоритмы, с которыми вы можете познакомиться в специальной литературе. Но мы пока ограничимся приведенными выше методами.

7.6.Изменение частоты дискретизации временных рядов

При анализе временных рядов достаточно часто возникает необходимость выполнения различных операций, связанных с преобразованием исходных наборов данных. Одной из таких операция является изменение частоты дискретизации временных рядов, которую еще называют повторной выборкой.

Повторная выборка временных рядов — это процесс изменения частоты, на которой регистрируются точки данных (наблюдения). Повторная выборка обычно выполняется для того, чтобы проанализировать, как будут вести себя данные временных рядов при разных частотах дискретизации (например, с изменением частоты данных с месяца на неделю).

Повторная выборка временных рядов обычно выполняется в следующих случаях:

- принудительное применение заданной частоты к данным (например, данные, собранные с разной разницей во времени между различными измерениями).
- задание новой частоты, отличной от уже существующей частоты в имеющихся данных.

Повторная выборка обычно выполняется для повышения или понижения частоты дискретизации.

23. **Повышение частоты дискретизации**. В этом случае временной ряд преобразуется из более низкой частоты в более высокую частоту (например, годовые данные трансформируются в месячные, месячные - в дневные). Когда данные временного ряда преобразуются из более низкой частоты в более высокую частоту, число наблюдений увеличивается, поэтому необходимо использовать методы для заполнения вновь созданной частоты.
24. **Понижение частоты дискретизации**. В этом случае временной ряд преобразуется из высокой частоты дискретизации в более низкую частоту (например, из дневных в недельные, из недельных в месячные, из месячных в годовые). При преобразовании временных рядов из высокой частоты в более низкую частоту количество выборок уменьшится, что приводит к потере некоторых значений. Обычно в этом случае к данным применяются методы агрегирования (например, усреднение или суммирование).

Схематично изменение частоты дискретизации представлено на рис. 7.8.

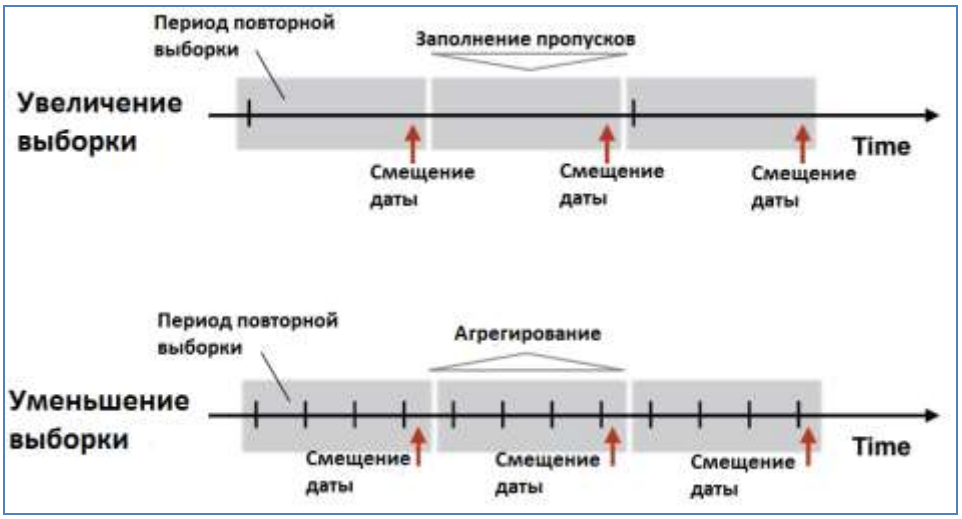


Рис. 7.8. Изменение частоты дискретизации временного ряда

Далее будут рассмотрены способы повторной выборки данных временных рядов с помощью библиотеки pandas.

7.6.1.Увеличение частоты дискретизации данных с использованием метода asfreq

Метод asfreq () позволяет увеличить частоту дискретизации и принимает следующие параметры для заполнения вновь создаваемых данных:

- freq - задает новую частоту для объекта временного ряда;

- method – указывает метод заполнения создаваемых данных;
- fill_value - позволяет заменить значения NaN (задается по умолчанию) на значения, указанные в этом параметре.

Параметр freg может принимать строки, указывающие желаемую частоту. Этот параметр может принимать следующий значения частоты:

- «M» — ежемесячная;
- «W» — еженедельная;
- «D» — ежедневная;
- «H» — часовая;
- «60s» — секундная частота (здесь 60s - это 60 секунд).

Параметр method предоставляет список следующих методов для заполнения вновь созданных индексов при повышении частоты дискретизации данных временных рядов:

- ffill или pad – заполняет создаваемую запись значением предыдущей записи;
- bfill или backfill - заполняет создаваемую запись значением следующей записи.

По умолчанию параметра method имеет значение None, и при повышении частоты дискретизации во вновь создаваемые индексы помещаются значения NaN .

Параметр fill_value позволяет заполнять NaN значением, указанным в этом параметре. Он не заполняет существующие NaN в исходных данных, а заполняет только те значения NaN, которые генерируются методом asfreq() при повторной выборке данных.

Реализуем программный модуль, в котором создадим тестовый временной ряд с пропусками, а потом увеличим его частоту с заполнением новых данных заданными значениями. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 7.27).

Листинг 7.27. Ссылка активации маршрута add_time_row

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем add_time_row (листинг 7.28).

Листинг 7.28. Маршрут URL с именем add_time_row

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление add_time_row (листинг 7.29).

Листинг 7.29. Представление (view) add_time_row

Примечание.
В демо-версии книги листинги программ удалены.

В этом программном модуле выполнен импорт библиотеки pandas и создан тестовый временной ряд (объект s6), который содержит 6 строк с датами через день. То есть в исходном временном ряде есть пропуски дат.

Далее, с использованием метода asfreg, к этому набору данных были добавлены строки с пропущенными датами (создан объект s11), при этом никаких значений ячейкам новых строк не присваивалось:

```
s11 = s6.asfreq(freq="1D")
```

В следующем фрагменте программного кода с использованием метода asfreg, к исходному набору данных в индексную колонку были добавлены строки с пропущенными датами и одновременно выполнено заполнение колонки с данными с использованием метода ffill (создан объект s11_fill):

```
s11_fill = s6.asfreq(method="ffill", freq="1D")
```

Затем с использованием метода asfreg, к исходному набору данных в индексную колонку были добавлены строки с пропущенными датами и одновременно выполнено заполнение колонки с данными с использованием метода pad (создан объект s11_pad):

```
s11_pad = s6.asfreq(method="pad", freq="1D")
```

В следующем фрагменте программного кода с использованием метода asfreg, к исходному набору данных в индексную колонку были добавлены строки с пропущенными датами и одновременно выполнено заполнение колонки с данными с использованием метода bffill (создан объект s11_bfill):

```
s11_bfill = s6.asfreq(method="bfill", freq="1D")
```

Затем с использованием метода asfreg, к исходному набору данных в индексную колонку были добавлены строки с пропущенными датами и одновременно выполнено заполнение колонки с данными с использованием метода fill_value (создан объект s11_val):

```
s11_val = s6.asfreq(fill_value=10.0, freq="1D")
```

На следующем шаге все созданные объекты, которые имеют тип Series, преобразованы в объекты DataFrame:

```
s6 = s6.to_frame()
s11 = s11.to_frame()
s11_fill = s11_fill.to_frame()
s11_pad = s11_pad.to_frame()
s11_bfill = s11_bfill.to_frame()
s11_val = s11_val.to_frame()
```

Затем, для того, чтобы объекта dataFrame можно было вывести в шаблон Django, они адаптированы к формату HTML:

```
s6 = s6.to_html()
s11 = s11.to_html()
s11_fill = s11_fill.to_html()
s11_pad = s11_pad.to_html()
s11_bfill = s11_bfill.to_html()
s11_val = s11_val.to_html()
```

Полученные наборы данных с заголовком страницы через контекстную переменную context передается в шаблон df_time_row.html (листинг 7.30).

Листинг 7.30. Шаблон df_time_row.html

Примечание.
В демо-версии книги листинги программ удалены.

Это шаблон с контейнером, в котором имеется несколько тегов для вывода содержимого наборов данных. Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 7» и активировать ссылку «Увеличение частоты временного ряда».

Если выполнить набор вышеприведенных модулей, то получим следующий результат (рис. 7.9).

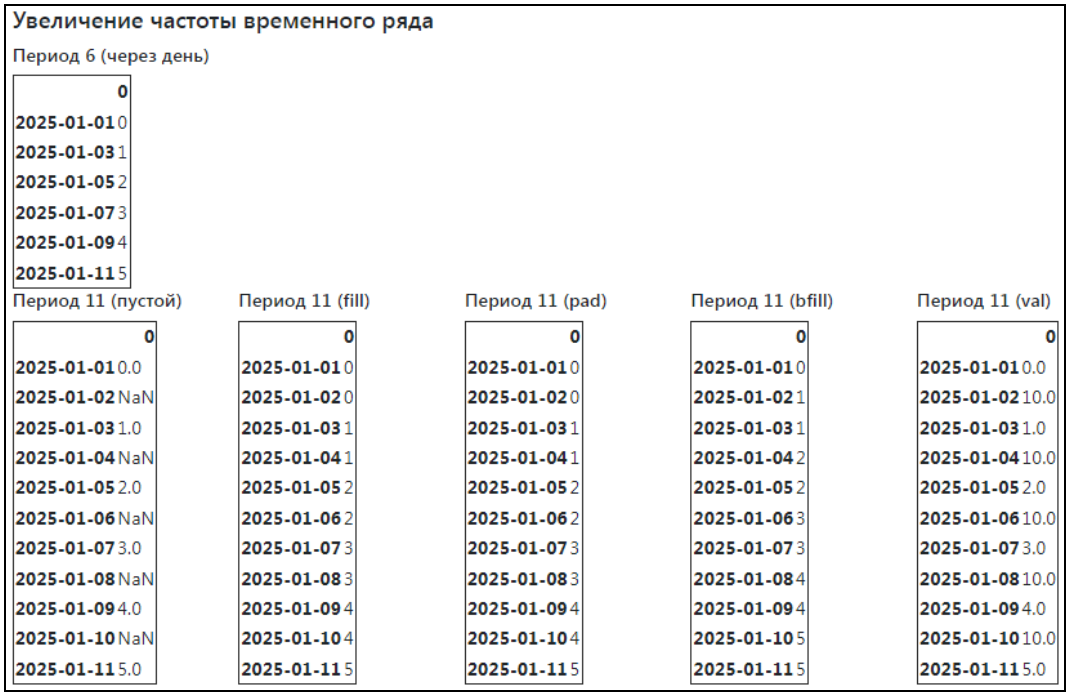


Рис. 7.9. Результаты работы методов увеличения частоты дискретизации временного ряда

Как видно из данного рисунка, в исходном наборе данных (набор – «Период 6 через день») присутствует 6 строк с датами через один день.

Далее, с использованием метода asfreg, к этому набору данных были добавлены строки с пропущенными датами: набор - «Период 11 (пустой)». В этом наборе в колонке с датами заполнены все пропуски, и мы имеем временной ряд без «пустот». Однако так как новым данным никаких значений не присваивалось, то новые ячейки имеют значения NaN.

Следующие два набора данных абсолютно одинаковы: «Период 11 (fill)» и «Период 11 (pad)». Методы fill и pad заполнять пропуски NaN данными из предыдущих строк.

У следующего набора данных «Период 11 (bfill)» методом bfill пропуски NaN заполнены данными из последующих строк.

У последнего набора данных «Период 11 (vall)» методом fill_value пропуски NaN заполнены заданными данными (в этом примере числом 10).

7.6.2.Восстановление непрерывности дат временного ряда методом resample

Для включения пропущенных дат в DataFrame в Pandas можно использовать метод resample. Он позволяет воссоздать полный спектр дат в существующем наборе данных. Инструкция восстановления непрерывности индексного столбца с датами имеет следующий вид:

fill_df = df["<имя колонки>"].resample("1D").mean().fillna(method="ffill")

Здесь используются следующие обозначения:

- df – исходный набор данных;
- <имя колонки> - колонка, которую нужно перенести в создаваемый набор данных;
- «1D» указывает на то, что в периоде должны присутствовать ежедневные данные, и не должно быть пропущенных дат.

Методы mean и ffili указывают на то, что для заполнения значения в строке дополняемой даты используется средне значение предыдущей даты. Этот метод можно использовать, например, для выравнивания интервалов дат в данных о торгах на бирже. Как правило, в выходные и праздничные дни торги на бирже не проводятся, соответственно эти даты во временном ряде отсутствуют, что приводит к его неоднородности. Рассмотрим применение этого метода на примере восстановления непрерывности дат в данных биржевых торгах. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 7.31).

Листинг 7.31. Ссылка активации маршрута resamle_date

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем resamle_date (листинг 7.32).

Листинг 7.32. Маршрут URL с именем resamle_date

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление resamle_date (листинг 7.33).

Листинг 7.33. Представление (view) resamle_date

Примечание.
В демо-версии книги листинги программ удалены.

В этом программном модуле выполнен импорт необходимых библиотек и сформирован интервал дат (объекты d1, D2). Здесь за конец периода взята текущая дата, а за начало периода дата годом ранее от текущей даты (задан период один год от текущей даты).

Затем сформирован список компаний (объект firm), задан источник данных (объект sours) и выполнена загрузка данных о стоимости акций фирм через API Stooq (объект data). В учебных целях загруженный набор данных выведен в окно терминала.

В следующей инструкции выполнено восстановление пропущенных дат:

```
df = data["Close"].resample("1D").mean().fillna(method="ffill")
```

Здесь фактически сформировать новый набор данных (объект df), в который из объекта data перенесены данные колонки Close (стоимость акций на момент закрытия торгов). При этом была восстановлена вся цепочка дат методом ffill (в строку восстановленной даты перенесены данные из строки с предыдущей датой).

В следующих инструкциях сформирован график и подписи к его осям:

```
fig = go.Figure()
fig.add_trace(go.Scatter(x=df.index, y=df['AAPL'], name="Apple"))
fig.add_trace(go.Scatter(x=df.index, y=df['GOOGL'], name="Google"))
fig.add_trace(go.Scatter(x=df.index, y=df['MSFT'], name="Microsoft"))
fig.update_layout(xaxis_title="Дата",
                  yaxis_title="Стоимость акций, $",
                  title='Котировки акций от Stooq')
```

После этого график приведен к формату, в котором он может быть показано на HTML странице (объект plot_div) и создана строка с заголовком страницы (title_str). Полученный график (plot_div) с заголовком страницы через контекстную переменную context передается в шаблон fig_plotly.html (листинг 7.34).

Листинг 7.34. Шаблон fig_plotly.html

Примечание.
В демо-версии книги листинги программ удалены.

Это достаточно простой шаблон с контейнером, в котором выведен заголовок страницы и объекты plot_div (график). Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 7» и активировать ссылку «Восстановление непрерывности дат».

Если выполнить набор вышеприведенных модулей, то получим следующий результат (рис. 7.10).



Рис. 7.10. Котировки акций на основе данных API Sooq

Как видно из данного графика, на основе данных, загруженных из API Stooq, сформирован график котировки акций компаний Apple, Google и Microsoft. Но нам в данном примере важно выполнить сравнение данных, полученных через API с данными после восстановления цепочки дат. Для этого обратимся к данным, которые выведены в окно терминала.

Итак, из API Stooq были получены следующие данные:

```
Symbols       AAPL    GOOGL    MSFT
Date
2025-06-17  195.640  175.950  478.040
2025-06-16  198.420  176.770  479.140
2025-06-13  196.450  174.670  474.960
2025-06-12  199.200  175.700  478.870
2025-06-11  198.780  177.350  472.620
...            ...      ...      ...
2024-06-25  208.369  183.576  449.240
2024-06-24  207.443  178.777  445.972
2024-06-21  206.795  179.187  448.074
2024-06-20  208.978  175.864  444.009
2024-06-18  213.572  174.658  444.647
[250 rows x 3 columns]
```

Из данной распечатки видно, что в наборе данных всего 250 строк. В цепочки дат отсутствуют данные за 14-е и 15-е июня 2025 года. Если посмотреть на календарь, то это субботние и воскресные дни. В эти дни торги на бирже не проводились.

После того, как цепочка дат была восстановлена, сформировался следующий набор данных:

```
Symbols       AAPL    GOOGL    MSFT
Date
2024-06-18  213.572  174.658  444.647
```

```
2024-06-19  213.572  174.658  444.647
2024-06-20  208.978  175.864  444.009
2024-06-21  206.795  179.187  448.074
2024-06-22  206.795  179.187  448.074
...            ...      ...      ...
2025-06-13  196.450  174.670  474.960
2025-06-14  196.450  174.670  474.960
2025-06-15  196.450  174.670  474.960
2025-06-16  198.420  176.770  479.140
2025-06-17  195.640  175.950  478.040
[365 rows x 3 columns]
```

В этом наборе данных уже 365 строк, то есть присутствуют все дни календарного года. Здесь присутствуют строки с данными за 14 и 15 июня (это суббота и воскресенье), а сами данные перенесены от предыдущего дня (от 13 июня).

7.6.3. Уменьшение частоты дискретизации данных с использованием метода resample

Метод resample() позволяет уменьшить частоту выборки. Общий вид инструкции, реализующей данный метод, и имеет вид:

df_m = fill_df.resample(rule="<правило>").<функция_агрегирования>()

Здесь параметр rule может принимать следующие значения для снижения частоты;

- «M» — ежемесячная;
- «W» — еженедельная;
- «D» — ежедневная;

Для заполнения вновь создаваемой выборки используются различные функций агрегации данных с учетом выбранной частоты:

- sum() — вычисляет сумму значений;
- mean() — вычисляет среднее значение;
- count() — вычисляет количество значений;
- max() – максимальное значение;
- min() – минимальное значение;
- std() — вычисляет стандартное отклонение;
- var() — вычисляет дисперсию;
- sem() — стандартная ошибка среднего значения.

Рассмотрим применение этого метода на примере снижение частоты выборки дат в данных биржевых торгов (динамика котировки акций компаний Apple, Google и Microsoft). Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 7.35).

Листинг 7.35. Ссылка активации маршрута up-time_row

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем up-time_row (листинг 7.36).

Листинг 7.36. Маршрут URL с именем up-time_row

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление up-time_row (листинг 7.37).

Листинг 7.37. Представление (view) up-time_row

Примечание.
В демо-версии книги листинги программ удалены.

Программный код этого модуля практически полностью повторяет код представления resamle_date предыдущего раздела. Поэтому мы здесь не будем делать подробный анализ кода этого модуля, а остановимся только на отличиях

Здесь, как и в модуле предыдущего раздела, сначала были загружены данные о котировках акций и выполнено восстановление цепочки дат:

```
df = data["Close"].resample("1D").mean().fillna(method="ffill")
```

Таким образом, были получены котировки акций за все 365 дней календарного кода. А вот в следующих инструкциях была выполнена процедура увеличения (укрупнения) частоты выборки. Сначала получили среднемесячную котировку акций (объект df_m):

```
df_m = df.resample(rule="M").mean()
```

Потом получили недельные котировки акций (строки закомментированы):

```
# Уменьшаем частоту (недельные данные)
# df_w = df.resample(rule="W").count()
# df_w = df.resample(rule="W").max()
# df_w = df.resample(rule="W").min()
```

Примечание.
Сняв комментарии с этих строк, вы можете получить различные варианты недельных котировок с формированием соответствующих графиков.

В этом примере далее сформирован график среднемесячных котировок акций. Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 7» и активировать ссылку «Уменьшение частоты временного ряда».

Если выполнить набор вышеприведенных модулей, то получим следующий результат (рис. 7.11).



Рис. 7.11. Котировки акций на основе данных API Sooq при уменьшении частоты временного ряда

7.7. Краткие итоги

Итак, мы познакомились с некоторыми базовыми методами анализа и очистки временных рядов. Теперь, зная структуру и характеристики временного ряда, можно подобрать подходящую для него статистическую модель, определить параметры модели и решать задачи прогнозирования. Решению задач прогнозирования на основе временных рядов посвящена следующая глава.

Глава 8. Статистические модели для анализа временных рядов и прогнозирования

Большинство процессов, происходящих в окружающем нас мире, например, климатические, экономические, финансовые, изменяются во времени. Для математического представления такого рода данных используется понятие временного ряда, которое предполагает, что данные о чем-либо, выстроены в цепочку, связанную со временем. Такие наборы данных полезно анализировать, чтобы находить статистические закономерности и характеристики, которые помогут понять природу этих данных, выявить закономерности, а главное спрогнозировать развитие ситуации.

О возникновении временного ряда, как правило, говорят в результате многократных измерений показателей тех или иных характеристик системы. Это могут быть показатели природных, социально-экономических явлений и процессов или данные о состоянии технологической системы. Например, известна динамика изменения стоимость квадратного метра жилого помещения, при анализе которого стремятся выявить основное направление изменение стоимости, т.е. тренда. Или анализ объема спроса на продукцию компании с целью расчета ожидаемых поставок в торговые центры.

Для анализа временных рядов и прогнозирования используют различные статистические модели: авторегрессионные модели (AR), модели скользящего среднего (MA), а также модели ARMA, ARIMA и SARIMA. Прогнозирование на основе этих статистических моделей требует прохождения следующих этапов:

- анализ временного ряда и получение его характеристики;
- подбор подходящего типа статистической модели для временного ряда;
- определение оптимальных параметров для выбранной модели;
- обучение модели на основе исторических данных;
- проверка обученной модели на адекватность описания характера исторических данных;
- прогнозирования значений временного ряда на будущие периоды.

Исходя из этого, из материалов этой главы вы узнаете:

- об анализе и прогнозировании временных рядов на основе авторегрессионные модели (AR);
- об анализе и прогнозировании временных рядов на основе модели скользящего среднего (MA);
- об анализе и прогнозировании временных рядов на основе модели ARMA;
- об анализе и прогнозировании временных рядов на основе трехкомпонентной модели ARIMA;
- об анализе и прогнозировании временных рядов, имеющих сезонность, на основе модели SARIMA;
- об анализе и прогнозировании временных рядов, имеющих сезонность, на основе модели SARIMAX.

8.1. Авторегрессионные модели (AR) для анализа и прогнозирования временных рядов

Одна из главных целей анализа и изучения временных рядов – их прогнозирование. Это часто используется в экономике, финансах и бизнесе для предсказывания будущих тенденций. Один из популярных способов прогнозирования временных рядов – построение авторегрессионных моделей. Авторегрессионная модель основана на модели линейной регрессии. Рассмотрим понятия регрессии и авторегрессии.

Регрессия (лат. regressio – обратное движение) в статистике – это статистическая зависимость математического ожидания случайной величины от значений другой случайной величины или нескольких случайных величин.

Отличительной особенностью регрессионной зависимости от функциональной является ее многозначность. Функциональная зависимость y=f(x), сопоставляет значение аргумента x с единственным значением y, в регрессионной модели каждому значению аргумента может соответствовать несколько значений параметра y. При исследовании закономерностей природного характера, финансах, экономике и технике встречается достаточно много задач, целью которых является поиск отношения между двумя и более переменными. Регрессионный анализ хорошо подходит и часто используется для решения подобных задач, включая прогнозирование.

Авторегрессионная модель использует предыдущие значения временного ряда для получения следующих значений. Для вычисления коэффициентов модели используются различные методы, но самый надежный и простой – метод наименьших квадратов. Главным преимуществом такого подхода является то, что все алгоритмы из класса авторегрессионных моделей имеют четкое математико-статистическое обоснование, это делает авторегрессионные модели одними из наиболее научно обоснованных из всего разнообразия используемых методов прогнозирования временных рядов.

Но авторегрессионные модели имеют и недостатки. Одним из них является требование к временным рядам: для подбора точной модели требуется довольно большая выборка, что на практике не всегда возможно. Также большим недостатком является низкая адаптивность модели, при получении новых данных того же ряда модель нужно снова обучать. Тем не менее, модели авторегрессии остаются актуальными, так как показывают хорошие результаты при грамотном применении.

8.1.1. Авторегрессионные модели (AR) для анализа временных рядов

Авторегрессионные модели (AR-модели) отражают взаимосвязь между наблюдением и несколькими отложенными наблюдениями, то есть предыдущими временными шагами. Идея заключается в том, что текущее значение данных временного ряда можно выразить как линейную комбинацию его прошлых значений с некоторым случайным шумом.

Математически авторегрессионная модель порядка p, обозначаемая как AR(p), может быть представлена следующим образом:

$$Xt = c + \phi 1Xt-1 + \phi 2Xt-2 + \ldots + \phi pXt-p + \varepsilon t$$

Где:

- Xt является значением параметра в момент времени t;
- c - это константа;
- $\phi 1, \phi 2, \ldots, \phi p$ являются параметрами модели;
- $Xt-1, Xt-2, \ldots, Xt-p$ являются запаздывающими значениями;
- εt представляет собой белый шум или случайную ошибку во времени t.

Количество временных шагов, на которые сдвигается ряд, еще называют задержка или лагом. Например, если лаг равен единице, то сравнивают каждое значение с предыдущим, если лаг равен двум, то сравнивают значения с предыдущими двумя временными интервалами и так далее.
Для анализа моделей автокорреляции используется график ACF, который отображает значения автокорреляции для различных запаздываний. При этом:

- на оси X отображаются значения задержки;
- на оси Y отображаются коэффициенты автокорреляции.

Пики, которые находятся за пределами доверительных границ (эти границы на графиках обычно выделены цветом), указывают на статистически значимые корреляции. Такие закономерности помогают выявить основную временную структуру данных и определить подходящий порядок запаздывания в моделях AR. Для того чтобы выбрать подходящий порядок запаздывания график ACF, как правило, анализируется в паре с графиком частной автокорреляционной функции (PACF).

Анализ пригодности модели AR для прогнозирования временного ряда выполняется в следующей последовательности:

1. Загрузка исходных данных, которые приводятся к стандартам наборов данных DataFrame (библиотеки Pandas);
2. Проверка временного ряда на стационарность и приведение к стационарному виду (если это необходимо);
3. Формирование графиков автокорреляции (ACF) и частичной автокорреляции (PACF) для дифференцированных рядов (чтобы определить подходящий порядок запаздывания «p» для модели AR);
4. Разделение данных на две выборки (обучающую - 80% и тестовую- 20%);
5. Создание модели AR и ее обучение на обучающей выборке;
6. Формирование прогноза на основе обученной модели для того же временного интервала, в котором находится тестовая выборка;
7. Сопоставление данных тестовой выборки с прогнозом и оценка точности работы обученной модели.

Реализуем простой пример анализа временного ряда с данными о пассажиропотоках с использованием модели AR и библиотеки matplotlib. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 8.1).

Листинг 8.1. Ссылка активации маршрута analiz_ar

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем analiz_ar (листинг 8.2).

Листинг 8.2. Маршрут URL с именем analiz_ar

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление analiz_ar (листинг 8.3).

Листинг 8.3. Представление (view) analiz_ar

Примечание.
В демо-версии книги листинги программ удалены.

В этом модуле выполнен импорт необходимых библиотек, задан путь к CSV файлу с данными о пассажиропотоках (объект path_csv), и набор данных из CSV файла загружен в объект df:

```
df = pd.read_csv(path_csv)
```

В файлах CSV данные хранятся в текстовом виде, поэтому далее для формирования полноценного временного ряда колонка с датами (Month) сделана индексной с типом данных datetime.

После этого выполнена проверка временного ряда на стационарность тестом Дики-Фуллера:

```
result = adfuller(df["Passengers"])
p_value = result[1]
```

Если значение теста Дики-Фуллера превышает предельное значение, то временной ряд приводится к стационарности путем дифференцирования.

После этого для анализа автокорреляции в едином макете графиков библиотеки matplotlib формируются графики автокорреляции (ACF) и частичной автокорреляции (PACF). Макет с этими графиками приводится к формату, пригодному для отображения в шаблонах Django (объект graphic).

Затем исходный набор данных разбит на две выборки: обучающая (объект train), и тестовая (объект test).

На следующем шаге сформирована статистическая AR модель (объект model) с использованием метода AutoReg библиотеки statsmodels с временным лагом lags=12, и проведено обучение модели на исторических данных:

```
p = 12
model = AutoReg(train, lags=p, old_names=False)
model_fit = model.fit()
```

Затем с использованием данной модели выполнен прогноз изменения пассажиропотока на период, в котором находится тестовая выборка (объект forecast_orig):

```
pred_test = model_fit.predict(start=test.index[0], end=test.index[-1], dynamic=False)
last_train_value = df["Passengers"].iloc[train_end]
forecast_orig = pred_test.cumsum() + last_train_value
forecast_orig.index = test.index
```

После этого с использованием библиотеки matplotlib сформирован график, на который выведено три линии:

- исходный набор данных (текущие наблюдения);
- прогноз на период, в котором находится тестовая выборка;
- вертикальная линия, отделяющая данные обучающей выборки и тестовой выборки.

Для формирования этих линий использовались следующие инструкции:

```
plt.plot(df["Passengers"], label="Наблюдения", linewidth=1)
plt.plot(forecast_orig, label="AR прогноз", linestyle="--")
plt.axvline(df.index[train_end], alpha=0.5, linestyle=":")
```

Макет с этим графиком приводится к формату, пригодному для отображения в шаблонах Django (объект graphic_1) и создана строка с заголовком страницы (title_str). Полученный графики (graphic, graphic_1) с заголовком страницы через контекстную переменную context передается в шаблон fig_mpl_analiz.html (листинг 8.4).

Листинг 8.4. Шаблон fig_mpl.html

Примечание.
В демо-версии книги листинги программ удалены.

Это достаточно простой шаблон с контейнером, в котором выведены строка заголовка страниц и два тега для вывода графиков. Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 8» и активировать ссылку «Анализ модели AR (пассажиропотоки)».

Если выполнить набор вышеприведенных модулей, то в шаблон Django будет выведено два графика. Графики автокорреляции (ACF) и частичной автокорреляции (PACF), которые представлены на рис. 8.1.

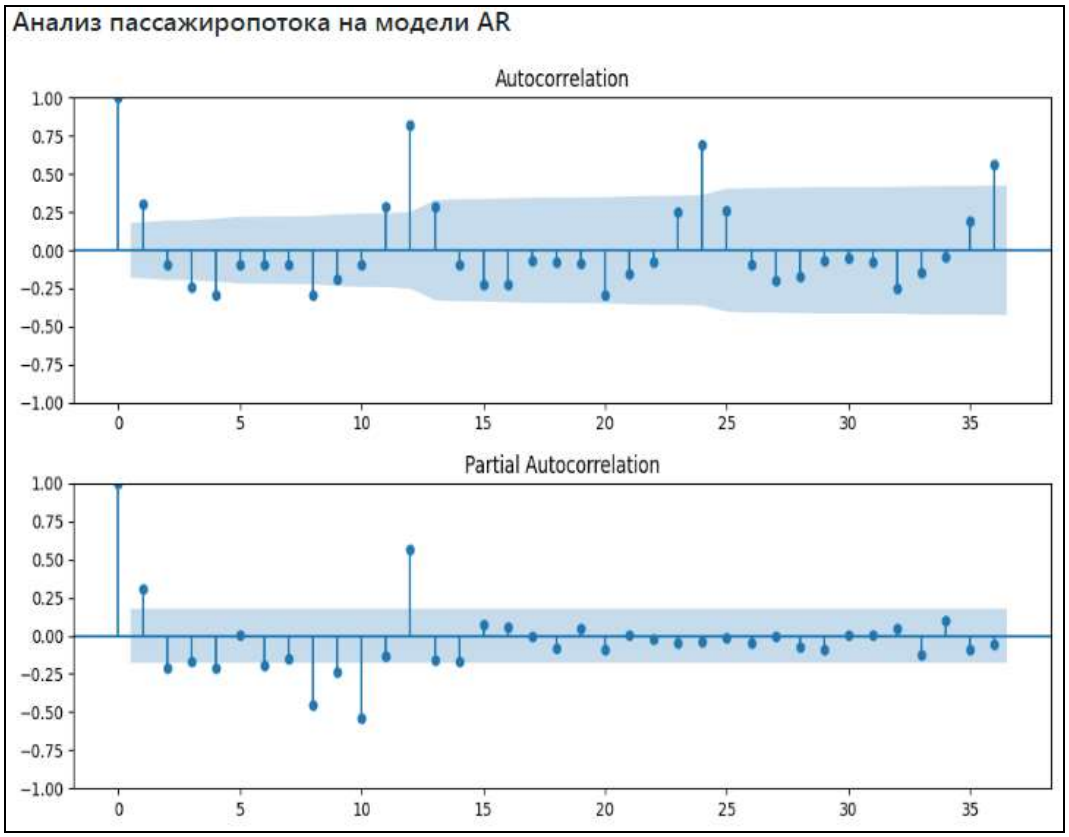


Рис. 8.1. Графики автокорреляции (ACF) и частичной автокорреляции (PACF) пассажиропотока с моделью AR

Из данных графиков видно, что временной ряд близок к стационарному ряду, а после 12-го лага функция автокорреляции PACF практически не выходит за граничные пределы

Графики с прогнозом на тестовый период представлены на рис. 8.2.

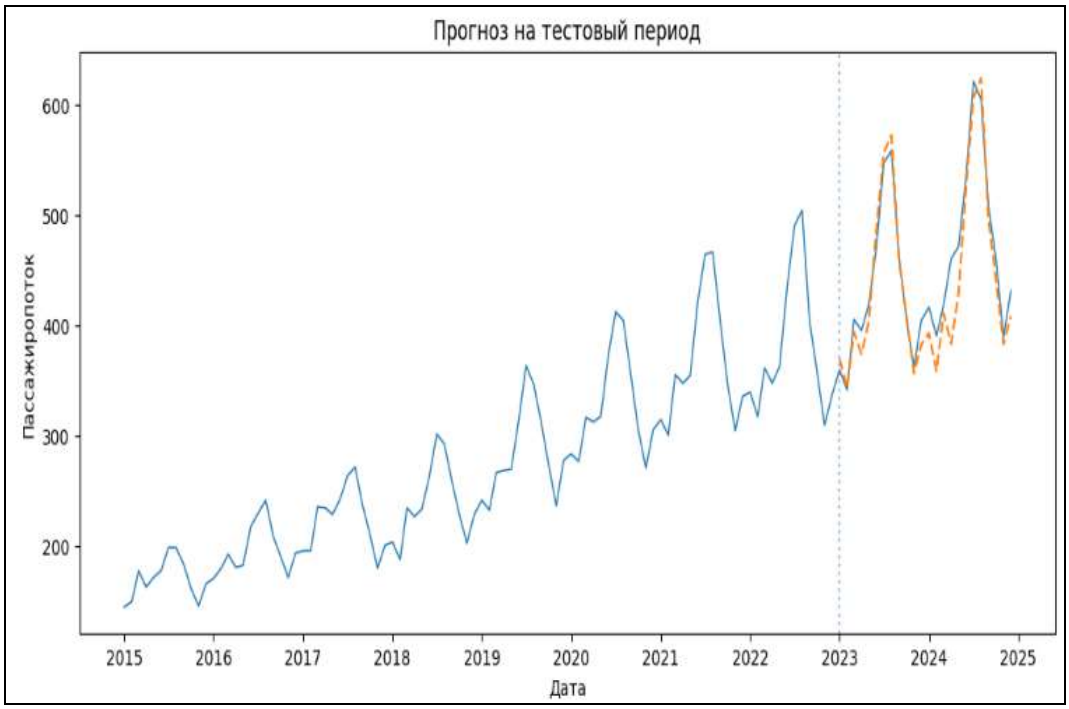


Рис. 8.2. Прогноз пассажиропотока с моделью AR на тестовый период

Из данного графика видно, что модель AR на периоде с тестовыми данными дает вполне адекватный прогноз – фактические наблюдения и прогнозные данные достаточно хорошо коррелируются друг с другом. Это говорит о том, что полученную AR модель вполне можно использовать для прогнозирования временного ряда на будущие периоды. Пример формирования такого прогноза будет рассмотрен в следующем разделе.

8.1.2. Авторегрессионные модели (AR) для прогнозирования временных рядов

Реализуем простой пример прогнозирования временного ряда на примере данных о пассажиропотоках с использованием библиотеки matplotlib. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 8.5).

Листинг 8.5. Ссылка активации маршрута prg_ar

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем prg_ar (листинг 8.6).

Листинг 8.6. Маршрут URL с именем prg_ar

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление prg_ar (листинг 8.7).

Листинг 8.7. Представление (view) prg_ar

Примечание.
В демо-версии книги листинги программ удалены.

В этом модуле выполнен импорт необходимых библиотек, задан путь к CSV файлу с данными о пассажиропотоках (объект path_csv), и набор данных из CSV файла загружен в объект df:

```
df = pd.read_csv(path_csv)
```

В файлах CSV данные хранятся в текстовом виде, поэтому далее для формирования полноценного временного ряда колонка с датами (Month) сделана индексной с типом данных datetime.

На следующем шаге сформирована статистическая модель (объект model) с использованием метода AutoReg библиотеки statsmodels с временным лагом lags=20, и проведено обучение модели на исторических данных:

```
model = AutoReg(df, lags=20)
model_fit = model.fit()
```

Затем с использованием данной модели выполнен прогноз изменения пассажиропотока на следующие два года (объект forecast):

```
forecast = model_fit.forecast(steps=24)
```

После этого с использованием интерфейса matplotlib сформирован график, который отражает исторические данные и прогноз. Далее этого график приведен к формату, в котором он может быть показан на HTML странице (объект graphic) и создана строка с заголовком страницы (title_str). Полученный график (graphic) с заголовком страницы через контекстную переменную context передается в шаблон fig_mpl.html (листинг 8.8).

Листинг 8.8. Шаблон fig_mpl.html

Примечание.
В демо-версии книги листинги программ удалены.

Это достаточно простой шаблон с контейнером, в котором выведена строка заголовка страниц и график. Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 8» и активировать ссылку «Прогноз на модели AR».

Если выполнить набор вышеприведенных модулей, то получим следующий результат (рис. 8.3).

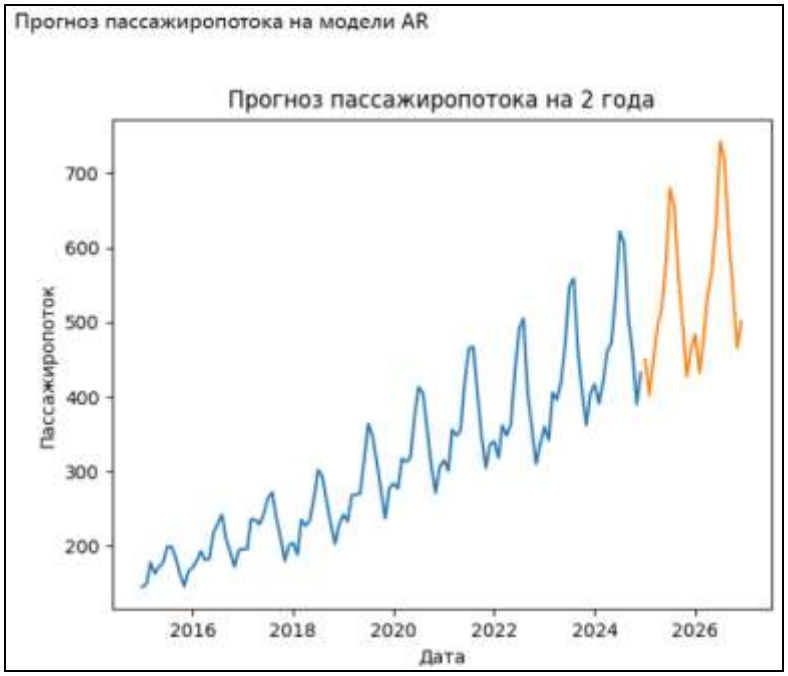


Рис. 8.3. Исторические данные и прогноз пассажиропотока с моделью AR

8.1.3.Преимущества и недостатки авторегрессионных моделей (AR)

Для авторегрессионных моделей AR можно отметить следующие преимущества.

25. **Простота модели**. Модели AR просты в понимании сущности их работы, удобные в использовании и реализации.
26. **Хорошее отражение связи прошлых и будущих периодов**. Модели AR хорошо описывают влияние прошлых данных, на данные будущих периодов.
27. **Эффективность для стационарных данных**. Модели AR хорошо работает, когда данные имеют устойчивые закономерности во времени.
28. **Быстрота работы**. Модели AR работают достаточно быстро и идеально подходят для небольших наборов данных.

29. **Обнаружение краткосрочных закономерностей**. Модели AR эффективно выявляет последние тенденции и изменения временного ряда.

При этом можно выделить следующие недостатки моделей MA.

30. **Требуется стационарность**: Для большинства реальных данных необходима предварительная обработка, например дифференцирование, чтобы соответствовать этому условию, а это может привести к частичной потере данных
31. **Ограничение недавним прошлым**. Модели AR не могут хорошо отображать долгосрочные зависимости.
32. **Сильная зависимость от выбора временного лага**. Слишком малое или слишком большое значение задержки (временного лага) может снизить производительность.
33. **Чувствительность к шуму**. Большое количество случайных колебаний и выбросов данных временного ряда могут привести к неточным прогнозам.
34. **Малый срок прогноза**. Эффективность моделей AR снижается при выборе длительных периодах прогнозирования.
35. **Требовательность к качеству данных**. Случайные выбросы или пропущенные значения могут сильно повлиять на точность прогноза.

Авторегрессионные модели рекомендуется использовать для краткосрочных прогнозов на временных рядах, которые характеризуются устойчивыми закономерностями и не имеют систематических случайных выбросов.

8.2. Модели скользящего среднего (MA) для анализа и прогнозирования временных рядов

Модель скользящего среднего (Moving Average - MA) — это тип модели анализа временных рядов, обычно используемой в эконометрике для прогнозирования тенденций и понимания закономерностей в данных временных рядов. В моделях скользящего среднего текущее значение временного ряда зависит от линейной комбинации прошлых значений ошибок белого шума временного ряда. В анализе временных рядов скользящее среднее обозначается буквой "q", которая представляет собой порядок модели скользящего среднего, или, проще говоря, текущее значение временного ряда будет зависеть от прошлых значений ошибок q.

Математически модель скользящего среднего порядка q может быть представлена в виде:

$$\mathbf{Xt} = c + \epsilon t + \theta 1.\epsilon t-1 + \theta 2.\epsilon t-2 + ... + \theta q.\epsilon t-q$$

Здесь:

- Xt - это значение временного ряда в момент времени t
- c — константа или среднее значение временного ряда
- $\epsilon t, \epsilon t-1, \epsilon t-2, ..., \epsilon t-q$ - это термины, обозначающие белый шум, связанный с временным рядом в моменты времени t, t-1, t-2, ..., t-q.
- $\theta 1, \theta 2, ..., \theta q$ - являются константами скользящего среднего.

Например, если мы рассмотрим модель MA(1), то в этой модели текущее значение временного ряда будет зависеть только от одной ошибки в прошлом, и временной ряд будет выглядеть следующим образом:

$$Xt=c+\epsilon t+\theta 1.\epsilon t-1$$

Один из наиболее важных аспектов моделей скользящих средних заключается в том, что чем выше значение порядка модели скользящего среднего (q), тем дольше сохраняется память модели и тем сильнее она зависит от прошлых значений.

Теперь давайте вернемся к некоторым свойствам временных рядов, которые помогут лучше понять модель скользящего среднего:

Стационарность. Стационарность - это свойство временных рядов, которое означает, что статистические свойства данных не меняются с течением времени. То есть среднее значение данных остается неизменным (или данные колеблются вокруг определенной величины), стандартное отклонение данных временных рядов остается постоянным, и в данных временных рядов не должно быть сезонности. Стационарность набора данных можно проверить визуально или с помощью теста Дики-Фуллера. Стационарность считается одним из наиболее важных аспектов, которым должны обладать данные временных рядов.

Дифференцирование. Дифференцирование — один из наиболее важных этапов, который следует учитывать при анализе временных рядов. Если данные не являются стационарными, то необходимо рассмотреть возможность их дифференцирования, поскольку для прогнозирования изменения данных временного ряда он должен быть стационарными. При обычном дифференцировании текущие данные временного ряда вычитаются из предыдущего значения.

$$\Delta yt = yt - yt - 1$$

Этот метод устраняет тренды в данных.

Белый шум. Белый шум - это погрешность, среднее значение которой равно нулю, а стандартное отклонение постоянно, при этом точки данных не коррелируют друг с другом. Белый шум служит ориентиром в процессе прогнозирования с помощью моделирования временных рядов. Если ошибка прогноза не является белым шумом, в модель можно внести дополнительные изменения, но если она достигает состояния, при котором ошибки прогноза являются белым шумом, то модель не нуждается в дальнейших улучшениях. Значения в рядах белого шума случайны и непредсказуемы, поэтому, если какие-либо данные временного ряда представляют собой белый шум, то не существует метода их моделирования или прогнозирования.

График ACF. График автокорреляционной функции (ACF) — это график корреляции между временным рядом и его запаздывающей версией. Он показывает, насколько временной ряд похож на свои различные запаздывающие значения. Здесь запаздывание — это фиксированное смещение во времени, на графике ACF по оси X отложен запаздывающий временной ряд, а по оси Y — корреляция, которая варьируется от -1 до 1.

8.2.1. Модели скользящего среднего (MA) для анализа временных рядов

Реализуем пример, в котором будем использовать данные Yahoo Finance о котировках акций компании AMD (Advanced Micro Devices) и смоделируем динамику их изменения с помощью модели скользящего среднего (MA). Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 8.9).

Листинг 8.9. Ссылка активации маршрута analiz_ma

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем analiz_ma (листинг 8.10).

Листинг 8.10. Маршрут URL с именем analiz_ma

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление analiz_ma (листинг 8.11).

Листинг 8.11. Представление (view) analiz_ma

Примечание.
В демо-версии книги листинги программ удалены.

В этом модуле выполнен импорт необходимых библиотек, задан тикер компании AMD (объект tick), и набор данных из API Yahoo Finance загружен в объект data:

```
data = tick.history(start="2020-01-01")
```

На биржах торги в воскресные дни не проходят, поэтому загруженный набор данных имеет пропуски дат. Для устранения этих пропусков используется метод resample библиотеки Pandas:

```
df = data["Close"].resample("1D").mean().fillna(method="ffill")
```

Затем с использованием методов plot_acf и plot_pacf библиотеки statsmodels строятся графики автокорреляции (ACF) и частичной автокорреляции (PACF):

```
plot_acf(df, lags=130, ax=ax[0])
plot_pacf(df, lags=130, ax=ax[1])
```

Для вывода графиков используется библиотека matplotlib. В следующих инструкциях макет с графиками ACF и PACF приводятся к формату, в котором он может быть отображен в шаблоне Django (объект graphic).

После этого выполняется расчет скользящего среднего с параметром q=10:

```
df['MA'] = df['Close'].rolling(q).mean()
```

Затем на основе набора данных df строится график котировок акций на момент закрытия торгов (столбец Close), и скользящего среднего (столбец MA). Полученный график приводится к формату, в котором он может быть отображен в шаблоне Django (объект plot_div).

Полученные графики с заголовком страницы (объект title_str) через контекстную переменную context передается в шаблон fig_mpl_analiz_ma.html (листинг 8.12).

Листинг 8.12. Шаблон fig_mpl_analiz_ma.html

Примечание.
В демо-версии книги листинги программ удалены.

Это достаточно простой шаблон с контейнером, в котором выведена строка заголовка страницы и два графика. Теперь можно проверить работу данных модулей.

> ***Примечание.***
> В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 8» и активировать ссылку «Анализ модели MA

Если выполнить набор вышеприведенных модулей, то в шаблон Django будет выведено два графика. Графики автокорреляции (ACF) и частичной автокорреляции (PACF) представлены на рис. 8.4.

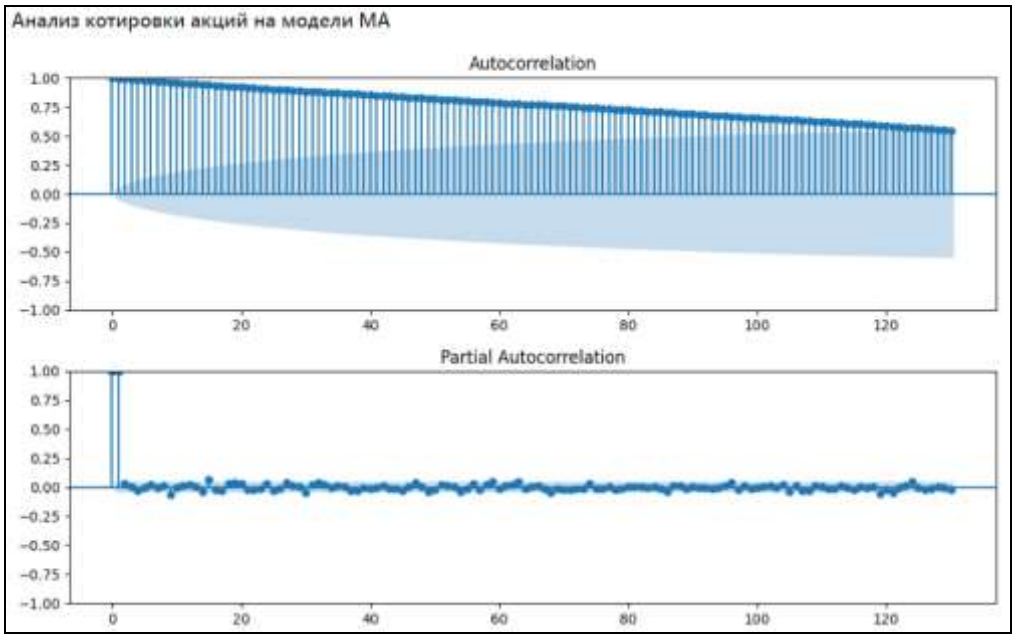


Рис.8.4. Графики автокорреляции (ACF) и частичной автокорреляции (PACF) котировки акций с моделью MA

Здесь на графике ACF видно, что первый лаг имеет корреляцию, равную 1, поскольку это тот же член ошибки, что и в текущем временном ряду, но начиная со второго лага значения графика ACF уменьшаются, но снижение корреляции между лагами достаточно медленное.

Синяя область обозначает границу, в которой значения перестают быть статистически значимыми. Из приведённого выше графика мы видим, что последним значимым запаздыванием является 130-е. Следовательно, порядок нашей модели MA будет равен 130. Значение q равно 130 означает, что текущее значение будет зависеть от последних 130 значений ошибок в данных.

Графики стоимости акций на момент закрытия торгов (Close) и скользящего среднего (MA) представлены на рис. 8.5.

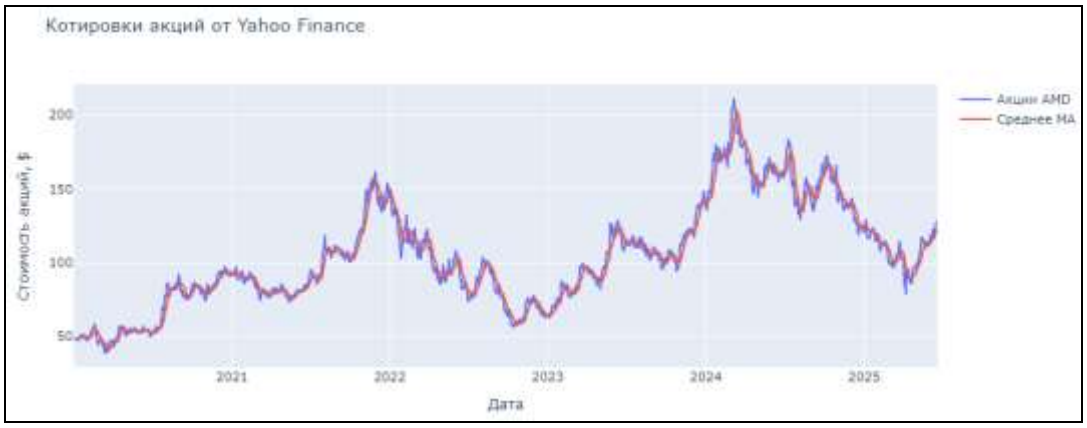


Рис. 8.5. Графики стоимости акций (Close) и скользящего среднего (MA) с моделью MA

Из данного графика видно, что модель MA на исторических данных достаточно адекватно описывает динамику изменения стоимости акций и может быть использована для решения задач прогнозирования.

8.2.2. Модели скользящего среднего (MA) для прогнозирования временных рядов

Реализуем пример, в котором будем использовать данные Yahoo Finance о котировках акций компании AMD (Advanced Micro Devices) и выполним прогноз изменения стоимости акций с помощью модели скользящего среднего (MA). Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 8.13).

Листинг 8.13. Ссылка активации маршрута prg_ma

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем prg_ma _ma (листинг 8.14).

Листинг 8.14. Маршрут URL с именем prg_ma

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление prg_ma (листинг 8.15).

Листинг 8.15. Представление (view) prg_ma

Примечание.
В демо-версии книги листинги программ удалены.

В этом модуле выполнен импорт необходимых библиотек, задан тикер компании AMD (объект tick), и набор данных из API Yahoo Finance загружен в объект data:

```
df = tick.history(start="2023-01-01", end="2025-06-30")
```

Затем создана модель AR (объект model) с временным лагом q=60 и выполнено ее обучение. Результаты обучения модели сохранены в объекте result:

```
model = ARIMA(endog=df['Close'], order=(0, 0, 60))
results = model.fit()
```

На следующем шаге была выполнена оценка модели на корректность описания исторических данных. Для этого был задан период, который частично совпадает с периодом исторических данных, и с использованием обученной модели выполнен прогноз изменения цены акций в этом периоде:

```
start_date = '2023-12-15'
end_date = '2025-06-20'
df['prediction'] = results.predict(start=start_date, end=end_date)
```

Фактически здесь к существующему набору данных df добавлена новая колонка с именем prediction. На этих данным можно проверить, насколько обученная модель адекватно описывает динамику изменения цен на акции.

Затем с использованием обученной модели сделан прогноз изменения цен на акции на 30 дней, следующие за последней датой полученной выборки данных. Для этого сначала был сформированы индексные данные будущего периода (объект predict_index):

```
forecast_steps = 30
predict_index = pd.date_range(start=df['Close'].index[-1], periods=forecast_steps + 1,
freq='D')[1:]
```

Затем создан набор данных типа DataFrame (объект df_ind), в котором в колонку Date были загружены даты будущего периода:

```
df_ind = pd.DataFrame({'Date': predict_index})
```

После этого с использованием обученной модели MA создан прогноз динамики цен на акции на будущий период в 30 дней (объект forecast):

```
forecast = results.forecast(steps=forecast_steps)
```

Фактически здесь создается объект типа Series.

Теперь необходимо объединить набор данных df_ind с датами будущего периода с данными прогноза на этот же период (объект forecast).

```
df_predict = forecast.to_frame().reset_index()
extracted_col = df_predict["predicted_mean"]
df_ind = pd.concat([df_ind, extracted_col.rename("prog_noz")], axis=1)
```

Здесь объект forecast преобразуется из Series в DataFrame (объект df_predict). Затем из полученного набора данных df_predict извлекается колонка с данными прогноза (объект extracted_col). Колонка с данными прогноза добавляется к набору данных с будущим периодом (к объекту df_ind). Таким образом, объект df_ind получил две колонки: Date – с датами будущего периода, prog_noz – с прогнозируемыми значениями цен акций на каждый день будущего периода.

Затем на основе наборов данных df и df_ind строятся графики трех линий:

- Динамика цен на акции на основе исходных исторических данных (синяя линия);
- Прогноз цен на акции на основе модели MA по фрагменту исторических данных (красная линия);
- Прогноз цен на акции на будущий период (зеленая линия)/

```
fig = go.Figure()
fig.add_trace(go.Scatter(x=df.index, y=df['Close'], name="Акции AMD"))
fig.add_trace(go.Scatter(x=df.index, y=df['prediction'], name="Модель MA"))
fig.add_trace(go.Scatter(x=df_ind['Date'], y=df_ind['prog_noz'], name="Прогноз"))
```

Полученный график приводится к формату, в котором он может быть отображен в шаблоне Django (объект plot_div).

Кроме того, набор данных с прогнозом на будущий период (объект df_ind, первые 10 строк) приведен к формату, в котором он может быть отображен в шаблоне Django (объект prog_noz_html)

Полученный график, заголовок страницы (объект title_str), и данные прогноза (объект prog_noz_html) через контекстную переменную context передается в шаблон fig_prog_noz.html (листинг 8.16).

Листинг 8.16. Шаблон fig_prog_noz.html

> ***Примечание.***
> В демо-версии книги листинги программ удалены.

Это достаточно простой шаблон с контейнером, в котором имеется три блока: для вывода строка заголовка страниц, для вывода графика, для вывода таблицы с данными прогноза на будущий период. Теперь можно проверить работу данных модулей.

> ***Примечание.***
> В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 8» и активировать ссылку «Прогноз на модели MA».

Если выполнить набор вышеприведенных модулей, то в шаблон Django будет выведен график с историческими данными и данными прогноза на модели MA (рис. 8.6).



Рис. 8.6. График исторических данных стоимости акций и прогнозы с моделью MA

На данном графике **синей линией** показаны исторические данные о стоимости акций компании AMD. Эти данные получены в режиме on-line через API Yahoo Finance. На этих данных было проведено обучение модели MA.

Красной линией показан прогноз изменения цены на акции по фрагменту исторических данных. Из данных графиков видно, что синяя и красная линии практически совпадают, то есть модель скользящего среднего (MA) вполне адекватно описывает реальные данные и может быть использована для составления прогнозов на будущие периоды.

Зеленой линией показан прогноз изменения цен на акции на будущие периоды. Из данного графика видно, что в ближайшие 30 дней ожидается рост цен на акции компании AMD.

Под графиком имеется слайдер, с помощью которого можно изменить интервал дат и посмотреть более детальную информацию о динамике цен на акции. Изменим интервал дат и сместим слайдер ближе к будущему периоду (рис. 8.7).



Рис. 8.7. Укрупненный график исторических данных стоимости акций и прогнозы с моделью MA

Как видно из данного рисунка, красная линия прогноза с минимальными отклонениями повторяет характер изменения синей линии с реальными данными. Это говорит о том, что прогнозу на будущий период, который отражен зеленой линией, вполне можно доверять!

В шаблоне fig_prog_noz.html также выведены первые 10 строк набора данных с прогнозом цен на будущий период (рис. 8.8).

Прогноз цен на следующие 30 дней

	prog_noz	date
0	127.526246	2025-06-21
1	129.498274	2025-06-22
2	130.955537	2025-06-23
3	133.558897	2025-06-24
4	135.369683	2025-06-25
5	136.188389	2025-06-26
6	136.880547	2025-06-27
7	139.748728	2025-06-28
8	141.298376	2025-06-29
9	137.095115	2025-06-30

Рис.8.8. Прогноз цен на акции с моделью MA в табличном виде

В таком виде данные могут быть полезны в тех случаях, когда необходимо более детально познакомится уровнем прогнозируемых цен на конкретную дату.

8.2.3. Преимущества и недостатки модели скользящего среднего (MA)

Использование модели скользящего среднего имеет ряд преимуществ и недостатков, на которые следует обратить внимание, чтобы добиться лучших результатов в моделировании и прогнозировании данных временных рядов.

Следует отметить следующие преимущества моделей MA

36. **Снижение шума**. Модели скользящего среднего эффективно сглаживают шум, присутствующий в данных временных рядов, что помогает уменьшить краткосрочные колебания в данных временных рядов.
37. **Стационарность**. Модели скользящего среднего сами по себе предполагают, что данные временного ряда стационарны, что является основным требованием при анализе временных рядов.
38. **Простота понимания сущности модели**. Понять, как работают модели скользящего среднего, не так уж сложно по сравнению с некоторыми другими моделями прогнозирования временных рядов. Порядок модели скользящего среднего — это количество предыдущих значений ошибки, от которых зависит текущее значение временного ряда.

39. **Возможность комбинирования**. Модели скользящего среднего можно легко комбинировать с авторегрессионными и интегрированными моделями, применительно к данным временных рядов.

При этом можно выделить следующие недостатки моделей MA.

40. **Возможная потеря данных при приведении к стационарности**. В моделях скользящего среднего предполагается, что данные временного ряда стационарны, но в действительности большинство реальных данных не являются стационарными. Поэтому при приведении временного ряда к стационарности может произойти потеря информации.
41. **Ограниченный диапазон прогнозирования**. Модели скользящего среднего подходят для краткосрочного прогнозирования, так как по мере увеличения периода прогнозирования (продолжительности будущего периода) возможности модели снижаются.
42. **Чувствительность к выбросам**. Если временной ряд имеет множество случайных выбросов, то это негативно скажется на точности прогноза.
43. **Сильное влияние порядка модели на точность прогноза**. Если порядок модели выбран неправильно, модель может оказаться неспособной эффективно прогнозировать данные временного ряда.

Из этого можно сделать вывод, что анализ данных временных рядов с помощью модели скользящего среднего возможен, но подходит только для коротких периодов времени, а прогнозы становятся менее точными по мере увеличения временного интервала.

8.3. Модели авторегресионного скользящего среднего (ARMA) для анализа и прогнозирования временных рядов

Модели авторегрессионного скользящего среднего (ARMA) используют фундаментальную концепцию анализа временных рядов, сочетающую в себе два ключевых компонента: авторегрессию (AR) и скользящее среднее (MA). Часть модели AR указывает, что текущее значение ряда основано на его предыдущих значениях, а часть MA предполагает, что на текущее значение также влияют прошлые ошибки прогноза. Этот двойной подход позволяет моделировать и прогнозировать данные, зависящие от времени, что делает ARMA популярным выбором среди статистиков и специалистов по обработке данных.

Для того чтобы модель ARMA была валидной и давала качественный прогноз, данные временного ряда должны быть стационарными. Это означает, что статистические свойства ряда, такие как среднее значение и дисперсия, должны оставаться постоянными с течением времени. Стационарность можно оценить с помощью различных тестов, таких как расширенный тест Дики-Фуллера. Если данные не стационарны, то для достижения стационарности перед подбором параметров модели ARMA можно применить такие преобразования, как дифференцирование или логарифмическое масштабирование.

Прогнозирование на основе модели ARMA широко используются в различных областях, включая финансы, экономику и экологию. Они особенно эффективны для прогнозирования будущих значений на основе исторических данных. Например, при анализе фондового

рынка модели ARMA могут помочь предсказать будущие цены на акции, анализируя прошлые движения цен. Аналогичным образом, в экономике модели ARMA могут использоваться для прогнозирования экономических показателей, таких как ВВП или уровень инфляции, помогая экономистам, финансистам и политикам принимать стратегически правильные решения.

Несмотря на свою полезность, модели ARMA имеют ограничения. Они предполагают, что лежащие в основе данные линейны и стационарны, что не всегда может иметь место в реальных сценариях. Кроме того, модели ARMA могут с трудом улавливать сложные закономерности в данных, такие как сезонность или внезапные сдвиги. В таких случаях более подходящими могут оказаться более продвинутые модели, такие как ARIMA, SARIMA и SARIMAX, но о них речь пойдет немного позже.

Модель ARMA состоит из двух компонентов (рис. 8.9).



Рис. 8.9. Структура модели ARIMA

Эта модель включает две составляющие: авторегрессию и скользящее среднее.

Авторегрессия (autoregressivel, AR) — это регрессия (связь) ряда с собственными значениями в прошлом (влияние прошлых данных на текущие). Другими словами, признаки в модели обычной регрессии мы заменяем значениями той же переменной, но за предыдущие периоды.

Когда мы прогнозируем значение в период t с помощью данных за предыдущий период с моделью AR(1), то уравнение будет выглядеть следующим образом:

$Yt = c + f * Yt-1,$

где:

- c — это константа;
- f — вес модели,
- Yt−1— значение в период t−1.

Количество используемых предыдущих периодов определяется параметром p модели (period). Обычно этот параметр записывается как AR(p).

Скользящее среднее (moving average model, MA) помогает учесть случайные колебания или отклонения (ошибки) истинного значения от прогнозного значения. Можно сказать, что модель скользящего среднего — это авторегрессия на ошибку. Обратите внимание, что ***скользящее среднее*** временного ряда, и ***модель скользящего среднего*** — это разные понятия.

Если использовать ошибку только предыдущего наблюдения, то уравнение будет выглядеть следующим образом.

Yt = m + f * E−1,

где:

- m — это среднее значение временного ряда,
- f — вес модели,
- E−1— ошибка в период t−1.

Такую модель принято называть ***моделью скользящего среднего*** с параметром q = 1 или MA(1). Разумеется, параметр q может принимать и другие значения. Обычно этот параметр записывается как MA(q).

Модель ARMA с параметрами (или как еще говорят порядками, orders) «p» и «q», или ARMA(p, q) позволяет описать любой стационарный временной ряд.

Модель ARMA предполагает, что в данных отсутствует тренд и сезонность (данные стационарны). Если данные не стационарны, то нужно приводить их к стационарности или использовать более сложные версии этих моделей:

- ARIMA, здесь добавляется компонент Integrated (I), который отвечает за удаление тренда (сам процесс называется дифференцированием);
- SARIMA, эта модель учитывает сезонность (Seasonality, S);
- SARIMAX включает еще и внешние или экзогенные факторы (eXogenous factors, отсюда и буква X в названии), которые напрямую не учитываются моделью, но влияют на нее.

Для модели ARMA необходимо определить диапазон возможных значений для параметров «p» и «q». Но прежде чем приступить к кодированию, необходимо пояснить, то такое модель ARMA, какой физический смысл имеют параметры «p» и «q».

Параметр «p» в модели ARMA обозначает порядок той части модели, которая связана с авторегрессией. Эта часть использует взаимосвязь между наблюдением и рядом запаздывающих (предыдущих) наблюдений для прогнозирования будущих значений. Например, если нужно спрогнозировать температуру на завтра, используя данные за последние несколько дней, то в части AR предполагается, что текущая температура и температуры предыдущих дней связаны.

Параметр «q» в модели ARMA обозначает порядок той части модели, которая связана со скользящим средним значением. Эта часть использует зависимость между наблюдением и остаточной ошибкой из модели скользящего среднего, применяемой к запаздывающим наблюдениям.

Таким образом, физический смысл параметра «p» — количество запаздывающих наблюдений, а параметра «q» — количество запаздывающих ошибок прогноза.

В этом разделе будет выполнен анализ временного ряда с динамикой цен на акции компании Microsoft, затем создадим модель ARMA и подберем параметры для этой модели, и, наконец, выполним прогноз изменения цен на акции на будущий период.

8.3.1. Модели ARMA для анализа временных рядов

Для анализа временного ряда загрузим реальные данные о котировках акций компании Microsoft через API Stooq. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 8.17).

Листинг 8.17. Ссылка активации маршрута analiz_arma

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем analiz_arma (листинг 8.18).

Листинг 8.18. Маршрут URL с именем analiz_arma

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление analiz_arma (листинг 8.19).

Листинг 8.19. Представление (view) analiz_arma

Примечание.
В демо-версии книги листинги программ удалены.

Обратите внимание, что из данного представления вызывается функция create_corr_plot. Эта функция обеспечивает формирование графиков автокорреляции ACF и PACF. Программный код формирования этих графиков достаточно сложный, поэтому он выделен в отдельную функцию для того, чтобы не загромождать листинг данного представления.

В этом представлении выполнен импорт необходимых библиотек, сформированы даты начала и конца периода в один год (объекты d1 и d2), задан тикер фирмы Microsoft (объект firm) и источник данных (объект sours), временной ряд с динамикой стоимости акций загружен в объект data:

```
data = web.DataReader(firm, sours, start=d1, end=d2)
```

Полученный набор данных выведен в окно терминала.

Затем из этих данных выбрана одна колонка Close (стоимость акций на момент закрытия торгов) и на ее основе сформирован соответствующий график (объект fig_close).

Так как торги в выходные дни не производятся, то на следующем шаге (перед формированием гистограммы) было проведено восстановление всех пропущенных дат с формированием нового набора данных (объект df_hist):

```
df_hist = data["Close"].resample("1D").mean().fillna(method="ffill")
```

Этот набор данных выведен в окно терминала и на его основе сформирован график с гистограммой (объект fig_hist).

После этого был выполнен расчет скользящего среднего с периодом запаздывания 30 дней (объект ssr):

```
n_dn = 30
ssr = df_close.rolling(n_dn).mean()
```

На основе выполненного расчета сформирован график скользящего среднего (объект fig_ssr).

На следующем этапе из полученного набора данных data сформирован набор данных типа Series (объект ser), на основе которого сформированы график автокорреляции ACF и PACF (объекты fig_acf и fig_pacf):

```
ser = data["Close"].squeeze()
fig_acf = create_corr_plot(ser, False, n_dn)
fig_pacf = create_corr_plot(ser, True, n_dn)
```

Для формирования этих графиков использовалась функция create_corr_plot, код которой будет приведен ниже.

После этого все графики приведены к формату, в котором они могут быть отображены в шаблоне Django. Затем графики вместе с заголовком страницы через контекстный словарь context переданы в шаблон fig_analiz_arma.html (листинг 8.20).

Листинг 8.20. Шаблон fig_analiz_arma.html

Примечание.
В демо-версии книги листинги программ удалены.

Это достаточно простой шаблон с контейнером, в котором имеется шесть блоков: один для вывода строка заголовка страниц, и пять для вывода графиков.

Теперь рассмотрим код функции create_corr_plot, которая формирует графики ACF и PACF (листинг 8.21).

Листинг 8.21. Функция create_corr_plot

Примечание.
В демо-версии книги листинги программ удалены.

Для работы этой функции выполнен импорт необходимых библиотек, а заголовок функции обеспечивает прием следующих параметров:

- series – набор данных типа Series;
- plot_pacf – признак типа формируемого графика (False – график ACF, True - график, PACF);
- nlags – количество дней временного лага.

Далее следует довольно сложный программный код, который мы не будем разбирать, так как в данном случае нашей целью является не изучение алгоритмов визуализации данных, а анализ временных рядов. На выходе функция возвращает объект fig в виде готового графика.

Теперь можно проверить работу данных модулей.

Примечание.

В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 8» и активировать ссылку «Анализ модели ARMA (котировка акций)».

Если выполнить набор вышеприведенных модулей, то в шаблон Django будут выведены наборы данных и пять графиков. Рассмотрим результаты работы программы по ходу ее выполнения, и начнем с табличных данных

Сначала в окно терминала будет выведен исходный набор данных:

```
------ Исходный набор данных-------
Attributes     Close       High        Low       Open          Volume
Symbols         MSFT       MSFT       MSFT       MSFT          MSFT
Date
2025-06-20  477.400  483.4600  476.870  482.230  3.757621e+07
2025-06-18  480.240  481.0000  474.460  478.000  1.752645e+07
2025-06-17  478.040  478.7399  474.080  475.395  1.541413e+07
2025-06-16  479.140  480.6943  475.000  475.210  1.562610e+07
2025-06-13  474.960  479.1800  472.760  476.410  1.681446e+07
...             ...       ...       ...      ...           ...
2024-06-28  445.255  453.6530  444.717  451.352  2.847026e+07
2024-06-27  451.132  454.4390  450.056  450.459  1.486270e+07
2024-06-26  450.444  451.8790  446.490  447.297  1.656988e+07
2024-06-25  449.240  449.7080  445.055  446.550  1.681130e+07
2024-06-24  445.972  451.0320  444.717  448.094  1.597431e+07
[249 rows x 5 columns]
```

В этом наборе данных 249 строк и 6 колонок:

- Date – даты (индексная колонка);
- Close – стоимость акций на момент закрытия торгов;
- High – максимальная стоимость акций в течение торгового дня;
- Low– минимальная стоимость акций в течение торгового дня;
- Open– стоимость акций на момент открытия торгов;
- Volume – объем продаж в течение торгового дня.

Нас будет интересовать только одна колонка - стоимость акций на момент закрытия торгов, поэтому только она включается в набор данных df_close:

```
------ Стоимость акций на момент закрытия торгов -------
Symbols       MSFT
Date
2025-06-20  477.400
2025-06-18  480.240
2025-06-17  478.040
2025-06-16  479.140
2025-06-13  474.960
...             ...
2024-06-28  445.255
```

```
2024-06-27  451.132
2024-06-26  450.444
2024-06-25  449.240
2024-06-24  445.972
[249 rows x 1 columns]
```

Этот набор данных содержит те же 249 строк, но только одну колонку с данными. Обратите внимание, что в индексной колонке есть пропущенные даты (например, нет данных за 14 и 15 июня), это были выходные дни, когда торги не проводились. Поэтому в следующем наборе данных df_hist пропущенные даты восстановлены:

```
------ Набор данных для гистограммы-------
Symbols        MSFT
Date
2024-06-24  445.972
2024-06-25  449.240
2024-06-26  450.444
2024-06-27  451.132
2024-06-28  445.255
...            ...
2025-06-16  479.140
2025-06-17  478.040
2025-06-18  480.240
2025-06-19  480.240
2025-06-20  477.400
[362 rows x 1 columns]
```

Итак, мы выполнили предварительный анализ структуры данных, но такие большие массивы данных, представленные в табличном виде, мало о чем говорят, поэтому обратимся к полученным графикам.

Динамика изменения стоимости акций во времени представлена на рис. 8.10.



Рис. 8.10. Динамика изменения стоимости акций во времени

Из данного рисунка видно, что с течением времени цены на акции имели значительные колебания, однако с мая 2025 года наблюдается их устойчивый рост.

Теперь проверим, есть ли значительные выбросы стоимости акций, или они меняются вокруг некоторой средней величины. Для этого достаточно взглянуть на гистограмму распределения стоимости акций (рис. 8.11).

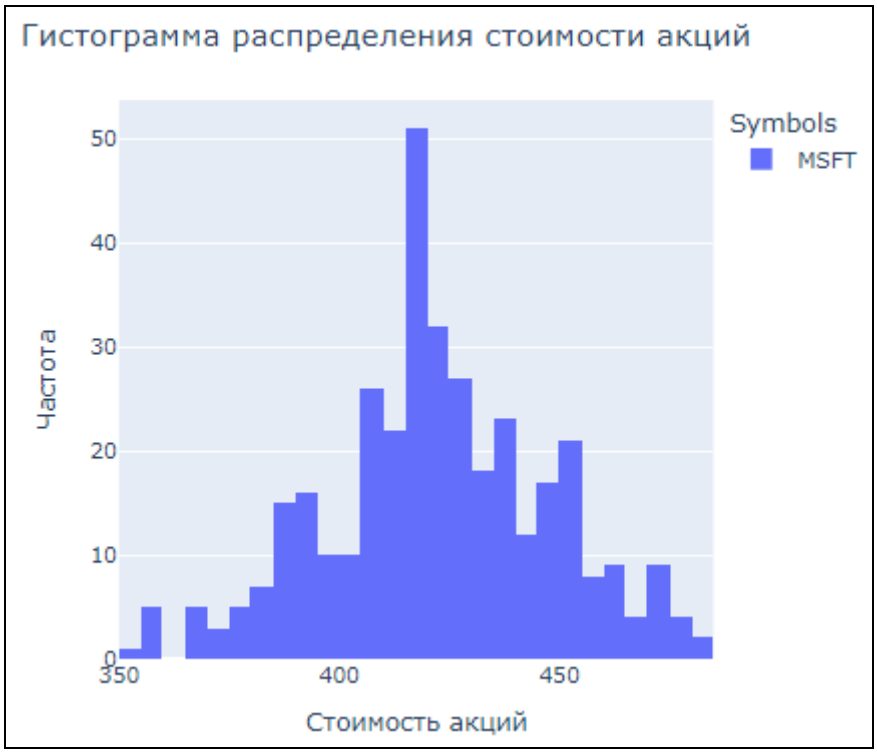


Рис. 8.11. Гистограмма распределения стоимости акций

Из данного рисунка видно, что стоимость акций достаточно равномерно колеблется относительно некой средней величины, рассеивание достаточно близко к нормальному закону распределения и явных выбросов не наблюдается. Это хороший признак того, временной ряд близок к стационарности.

Теперь посмотрим на общую тенденцию изменения цены на момент закрытия торгов. Для этого был построен график скользящего среднего, где ежедневные колебания цены сглаживаются среднемесячными значениями (рис. 8.12).

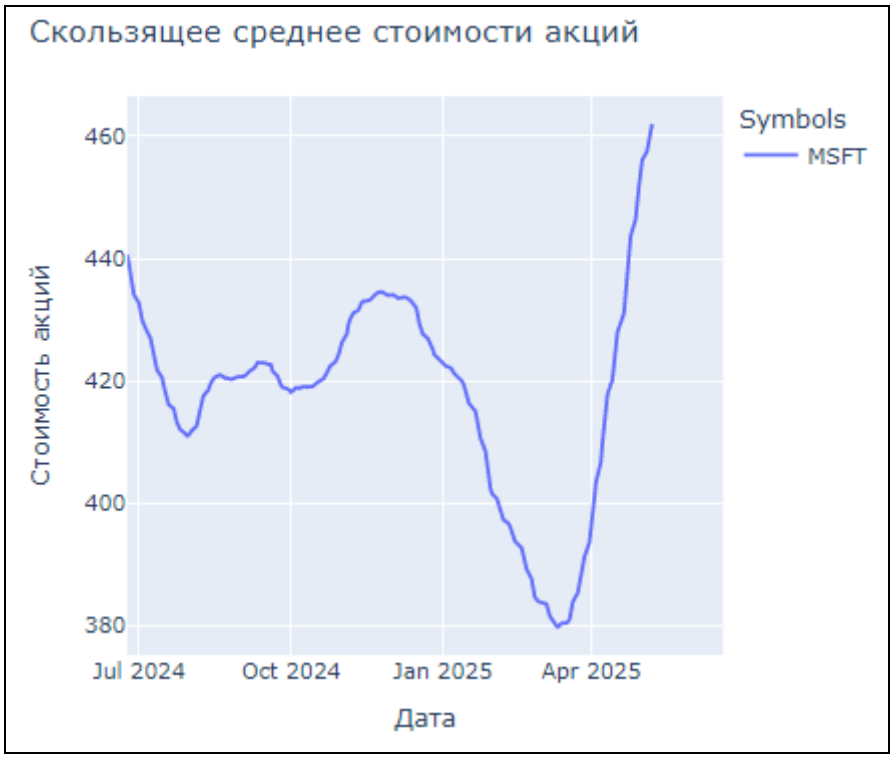


Рис. 8.12. Скользящее среднее стоимости акций

На данном графике видна более общая тенденция изменения цена. Из данного рисунка видно, что с января по апрель наблюдалась тенденция снижения стоимости акций, но начиная с апреля месяца, наметился устойчивый рост цены.

Создаваемая нами модель ARMA будет использовать предыдущие значения для прогнозирования значений текущего и будущих периодов. Например, мы сможем использовать то, что произошло день назад, чтобы предсказать, что происходит сейчас. Ведь то, что произошло день назад, косвенно информирует нас о том, что происходит сегодня. Но мы не должны ограничивать себя задержкой в один день. От выбранного количества дней задержки будет сильно зависеть качество прогноза. Поэтому первой задачей, которую нужно решить на основе анализа исходного массива данных - сколько дней задержек должно быть в нашей ARMA модели. Чтобы найти ответ на этот вопрос, проанализировать графики ACF и PACF.
При анализе временных рядов используется два вида корреляции:

- автокорреляция или автокорреляционная функция (autocorrelation function - ACF);
- частичная автокорреляционная функция (partial autocorrelation function – PACF).

Автокорреляционная функция (ACF) показывает, как сильно каждое последующее значение коррелирует с предыдущим, пред-предыдущим, пред-пред-предыдущим и так далее. По графику ACF можно сделать вывод о наличии тренда или сезонности.

Частичная автокорреляция (PACF) фиксирует корреляцию между двумя переменными, но после учета влияния других переменных. Другими словами, PACF убирает влияние каждого предыдущего шага на последующий шаг.

Таким образом, ACF помогает определить, как влияет значение предыдущего шага на последующий шаг, а PACF позволяет предсказать будущие значения, учитывая только определенную взаимосвязь между переменными, без влияния других факторов (например, без сезонности и тренда).

Задержка, или период запаздывания (лаг - lag), соответствует определенному моменту времени, после которого мы наблюдаем первое значение во временном ряду. Запаздывание может измеряться в часах, днях, неделях, месяцах годах. Это зависит от периода временного ряда. Например, данные о котировках акций могут быть предоставлены с периодичность 1 день, а данные о ВВП – с периодичностью 1 год.

Коэффициент корреляции может варьироваться от -1 (идеальная отрицательная зависимость) до +1 (идеальная положительная зависимость). Коэффициент, равный 0, означает отсутствие связи между переменными.

Цветная область на графике ACF (обычно показана голубым цветом) — это диапазон ошибок, и все, что находится в пределах этой области, не является статистически значимым. То есть значения корреляции за пределами этой области действительно выражают связь между переменными, а значения корреляции внутри этой области, скорее всего, являются статистической случайностью. По умолчанию доверительный интервал установлен на 95%. Обратите внимание, что для нулевого запаздывания значение ACF всегда равно единице, что логично, поскольку любое значение всегда идеально коррелирует с самим с собой.

Подводя итог, можно сказать, что автокорреляция — это связь между текущим значением параметра с его прошлыми значениями, а автокоррелограмма (ACF) — это график зависимости коэффициента корреляции от величины задержки (лага), который является визуальным представлением автокорреляции. А частичная автокорреляции (PACF) отражает «прямую» корреляцию между временным рядом и его запаздывающей версией.

Итак, начнем разбор нашего временного ряда с анализа графика ACF, который представлен на рис. 8.13.

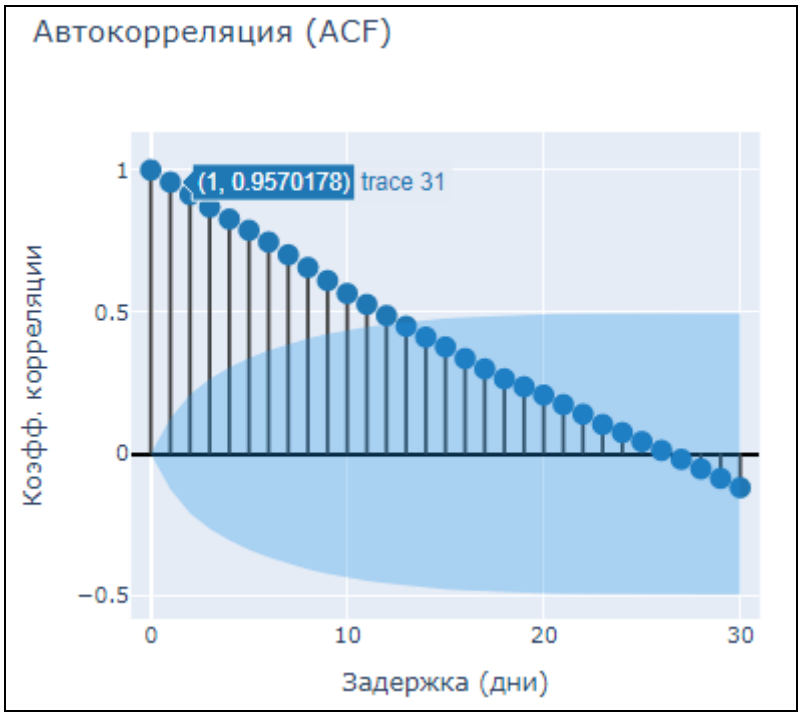


Рис. 8.13. График автокорреляции ACF

Из этого графика можно подчерпнуть много полезной информации. Сначала посмотрим на оси. Итак, на оси «x» представлены дни задержки. По мере того, как мы движемся слева направо, задержка становится все больше и больше, и мы идем дальше в прошлое, где, как мы думаем, предсказательная сила становится все меньше и меньше. На оси «y» показан коэффициент корреляции, значения которого изменяются в пределах от -1 до 1. Если имеет место сильная корреляция, он равен единице или (-1), если корреляции нет, он равен нулю.

Еще один элемент, на который нужно обратить внимание, это синее (цветное) облако. Все, что находится внутри этой области, имеет настолько малый коэффициент корреляции, что это статистически незначимо, это скорее случайность, чем устойчивая закономерность.

При задержке в ноль дней коэффициент корреляции равен единицы (прогноз точно соответствует реальным данным). При задержке в один день, то коэффициент корреляции немного уменьшится, но будет очень близок к единице (в нашем случае при задержке в один день он равен 0,957). Из вышеприведенного графика видно, что с увеличением количества дней задержки коэффициент корреляции уменьшается, а значит и предсказательная сила модели (достоверность прогноза) снижается по мере увеличения количества дней задержки. На графике нет повторяющихся пиков, значит у временного ряда нет сезонности. После 12-го лага коэффициент корреляции входит в область потери статистической значимости, поэтому в модели ARMA нет смысла использовать лаг более 12 дней.

Из этих рассуждений можно предположить, что график ACF можно использовать для того, чтобы определить, какой временной лаг (количество дней задержки) необходимо задать для ARMA модели. Но на самом деле это не так. График ACF дает лишь представление о том, как меняется корреляция при изменении временного лага. Но чтобы определить, сколько

дней должно быть во временном лаге в ARMA модели, необходимо использовать график PACF. Ведь график ACF — это только половина истории изменения временного ряда.

Мы знаем, что существует сильная корреляция между ценой акций текущего дня и ценой акций дня вчерашнего, графика ACF наглядно демонстрирует эту корреляцию. Но нет никаких оснований ограничивать себя текущим днем и возвратом на один день назад. Ведь мы можем вернуться назад по временной шкале на любое количество дней. Можно вернуться, например, на три дня назад, ведь есть сильная корреляция между тем, что произошло три дня назад и четыре дня назад.

Корреляция между двумя соседними днями при любом количестве шагов назад на самом деле формирует единообразное эхо с повторяющимися значениями. Это демонстрируют последние строки данного программного модуля. Мы передвинулись на один, а потом на два шага назад и получили практически идентичную корреляцию между текущим и предыдущим днем. Это легко проверить на графике ACF, который является интерактивным и есть возможность подсветить значение коэффициента корреляции при любом количестве шагов назад. Так, при шаге назад на один день коэффициент корреляции К1=0,957, а при шаге назад на два дня К2=0,913 (разница 0,044). Коэффициент корреляции при шаге назад на 6 дней К6=0,746, а при шаге назад на 7 дней К7=0,701 (разница 0,045). То есть, при любом количестве шагов назад разница между коэффициентами корреляции текущего дня и предыдущего практически не меняется.

Можно удалить возникшее повторяющееся эхо и посмотреть, какие лаги имеют предсказательную солидную силу, а какие просто удерживают это эхо. На графике частичной автокорреляционной функции (PACF) как раз и показана корреляция после удаления этого эха.

Обратимся теперь графику PACF (рис. 8.14).

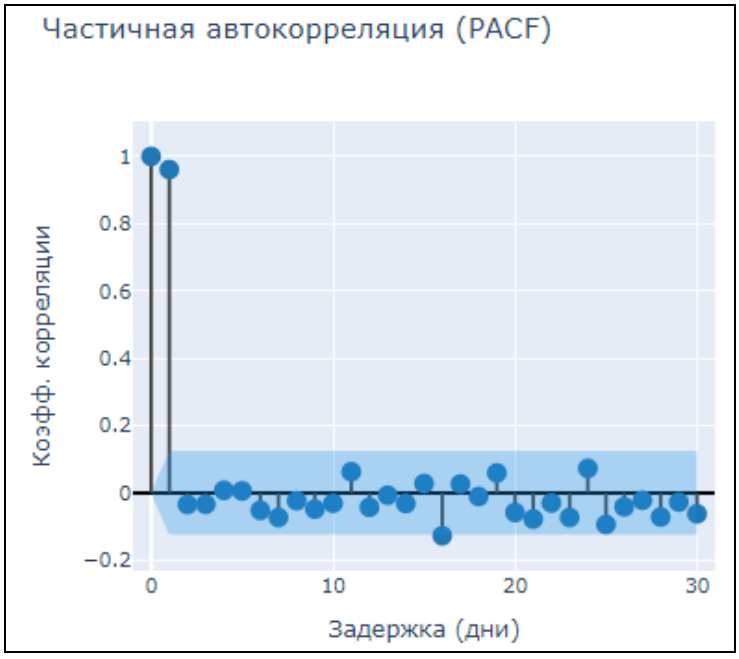


Рис. 8.14. График частичной автокорреляции PACF

График PACF похож на график ACF и имеет те же оси, на оси «x» показаны дни запаздывания, а по оси «y» - коэффициент корреляции. Но на этом сходство заканчивается, поскольку значения коэффициентов корреляции сильно отличаются (значение корреляции здесь намного меньше).

Каждое значение в этом графике содержит номер лага и соответствующее ему значение PACF. Положительные или отрицательные значения указывают на положительную или отрицательную корреляцию соответственно, а значения, близкие к нулю, указывают на слабую корреляцию при данном лаге. Например, при лаге в 16 дней коэффициент корреляции сильно отрицательный, а при лале в 11 дней положительный.

Исходя из выполненного анализа графиков ACF и PACF, для данного временного ряда в модели ARMA можно использовать временной лаг 11.

8.3.2. Поиск оптимальных параметров модели ARMA

Временные ряды хоть формально и похожи друг на друга, но каждый ряд будет иметь свои особенности и характеристики. Поэтому для решения задач прогнозирования для каждого временного ряда необходимо подбирать индивидуальные параметры. Конечно, можно визуально оценить значения параметров модели ARMA, если на основе временного ряда построить графики. На основе анализа графика PACF, можно определить, каким будет наилучший шаг отставания (наилучшее значение для параметра «p» - количество запаздывающих наблюдений). А на основе анализа график ACF можно определить наилучшее значение для параметра «q - количество запаздывающих ошибок прогноза». Но это довольно субъективный метод, который не даст нужной точности.

Иногда бывает очень сложно найти оптимальные значения «p» и «q» для модели ARMA, анализируя графики ACF и PACF, как было описано в предыдущем разделе. Поэтому существуют иные подходы к настройке этой модели. Сегодня большинство статистических инструментов имеют встроенную функцию, которую часто называют «auto_arima». Например, в Python метод «auto_arima» сам генерирует оптимальные параметры «p» и «q» , которые подходят для набора данных и обеспечивают более точное прогнозирование. Логика, лежащая в основе этого, такая же, как и логика, лежащая в основе настройки параметров любой другой модели машинного обучения. Нам нужно попробовать несколько комбинаций параметров «p» и «q» и сравнить результаты с помощью проверочного набора данных.

Поскольку пространство поиска оптимальных значений параметров модели невелико, обычно значения «p» и «q» не превышают 10, то можно применить популярный метод оптимизации параметров модели, называемый перебором по сетке. **Перебор по сетке** — это простой поиск по заданному вручную подмножеству значений параметров модели в процессе ее обучения. По сути, это означает, что этот метод будет перебирать все комбинации «p» и «q» из заданного нами подмножества с последующим выбором оптимальных значений. Для того чтобы найти наилучшую комбинацию «p» и «q», нужна целевая функция, которая будет оценивать эффективность модели на проверочном наборе данных. Для этой цели можно использовать критерии AIC и BIC. Чем ниже значение этих критериев, тем лучше модель.

> ***Примечание.***
> AIC (Akaike Information Criterion) и BIC (Bayesian Information Criterion) — это информационные критерии, которые используются для выбора модели в анализе временных рядов.

AIC — критерий, который оценивает, насколько хорошо модель подходит под данные, при этом учитывает количество параметров. Цель — найти баланс между соответствием модели данным и ее сложностью, чтобы избежать переобучения

BIC — критерий, который также оценивает соответствие модели данным, но более строго штрафует за сложность. Цель — выбрать модель, которая наилучшим образом объясняет данные, при этом остается простой.

Но для оценки оптимальности параметров модели можно использовать и другой критерий - среднюю абсолютную ошибку обучения модели (MAE), именно он и будет использован в данном примере. Мы напишем программный код, который выберет наилучшую комбинацию параметров «p» и «q» из заданного диапазона. Для возможных значений параметра «p» зададим диапазон от 0 до 30 дней (один месяц) с шагом 5. Для параметра «q» зададим меньший диапазон от 0 до 3 дней с шагом 1.

Почему для параметра «p» задается больший диапазон, чем для параметра «q». Когда мы задействуем первую часть модели ARMA (AR), то заглядываем на несколько шагов в прошлое, чтобы предсказать настоящее или будущее. Поскольку мы исследуем временной ряд со стоимостью акций, то нам нужно заглянуть не на 1-2 дня назад, а хотя бы на месяц. Поэтому максимально значение для временного лага задано в 30 дней. А вот параметр «q» должен зафиксировать быстрые и краткосрочные шоковые выбросы во временном лаге. Поэтому для данного параметра мы задаем более короткий интервал (1, 2, 3 дня).

Теперь нужно определить, какая комбинация параметров, даст наилучшую производительность и точность модели ARMA, а для этого нужно перебрать все возможные комбинации параметров. Такая процедура называется подбор параметров модели по сетке. При подборе параметров по сетке нужно выполнить несколько итераций обучения модели для каждой комбинации параметров «p» и «q». А затем необходимо вычислить среднюю абсолютную ошибку обучения (MAE) для каждой конкретной модели и сохранить этот результат. После этого нужно рассмотреть все значения MAE и найти то, которое имеет наилучшую производительность.

Примечание.

Следующий программный модуль может работать с ошибками при использовании последних версий библиотеки numpy. Если вы столкнетесь с такой ситуацией, установите numpy версии 1.24.4.: pip install numpy==1.24.4.

Реализуем данный алгоритм в программном коде. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 8.22).

Листинг 8.22. Ссылка активации маршрута params_arma

Примечание.

В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем params_arma (листинг 8.23).

Листинг 8.23. Маршрут URL с именем params_arma

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление params_arma (листинг 8.24).

Листинг 8.24. Представление (view) params_arma

Примечание.
В демо-версии книги листинги программ удалены.

В этом представлении выполнен импорт необходимых библиотек, сформированы даты начала и конца периода в один год (объекты d1 и d2), задан тикер фирмы Microsoft (объект firm) и источник данных (объект sours), временной ряд с динамикой стоимости акций загружен в объект data:

```
data = web.DataReader(firm, sours, start=d1, end=d2)
```

Полученный набор данных выведен в окно терминала.

Затем из этих данных выбрана одна колонка Close (стоимость акций на момент закрытия торгов) и на ее основе сформирован соответствующий график (объект fig_close).

Так как торги в выходные дни не производятся, то на следующем шаге (перед формированием гистограммы) было проведено восстановление всех пропущенных дат с формированием нового набора данных (объект df):

```
df = data["Close"].resample("1D").mean().fillna(method="ffill")
```

Этот набор данных разбит на две части: обучающая выборка (объект train) и тестовая выборка (объект test), и выполнен расчет средней цены акций (объект m_price) и базовой ошибки (объект base_mae). Затем создана сетка параметров для модели ARMA:

```
p_params = range(0, 35, 5)
q_params = range(0, 3, 1)
```

Здесь параметр «p» будет меняться от нуля до 30 с шагом 5, а параметр «q» будет принимать значения от нуля до трех.

В следующих инструкциях созданы вложенные циклы. Во внешнем цикле изменяется параметр «p», во внутреннем цикле изменяется параметр «q». В теле вложенного цикла выполняется обучение модели ARMA с текущими параметрами и делается расчет ошибки обучения MAE. Все рассчитанные значения ошибок сохраняются в объекте mae_grid, а по завершению циклов создать набор данных из ошибок обучения (объект mae_df).

В процессе работы над данным модулем был выполнен анализ полученных ошибок MAE, и определены оптимальные параметры для модели: p=5, q=0. Поэтому далее была создана модель ARMA с этими параметрами и проведено ее обучение:

```
order = (5, 0, 0)
odel = ARIMA(train, order=order)
result = model.fit()
```

После этого было сформировано четыре графика диагностики полученной модели, которые размещаются в едином макете (объект graphic).

После этого набор данных с ошибками mae_df приведены к формату, в котором он моет быть отображен в шаблоне Django и задан заголовок страницы. Затем графики и все рассчитанные параметры через контекстный словарь context переданы в шаблон fig_params_arma.html (листинг 8.25).

Листинг 8.25. Шаблон fig_params_arma.html

Примечание.
В демо-версии книги листинги программ удалены.

Это достаточно простой шаблон с контейнером, в котором имеются блоки для вывода строки заголовка страниц и рассчитанных параметров, и блок для вывода графиков. Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 8» и активировать ссылку «Поиск параметров модели ARMA».

Если выполнить набор вышеприведенных модулей, то в шаблон Django будет выведен набор данных с ошибками и четыре графика. Рассмотрим результаты работы программы по ходу ее выполнения, и начнем с табличных данных, которые представлены на рис. 8.15.

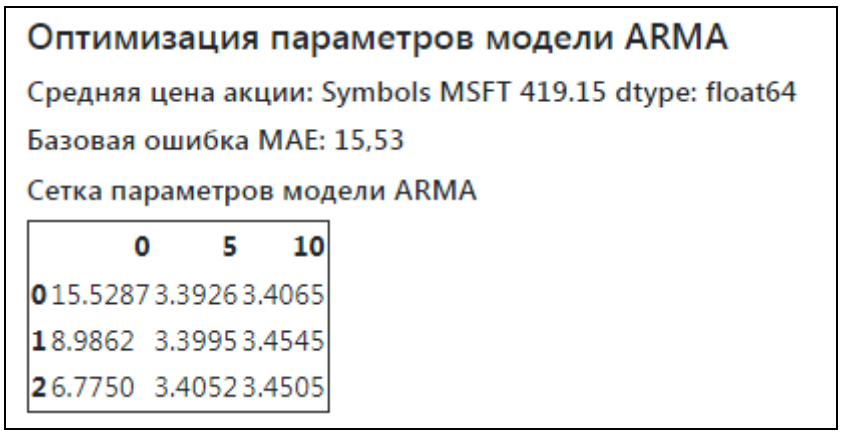


Рис. 8.15. Результаты расчетов при оптимизации параметров модели ARMA

Из данного рисунка видно, что средняя цена акций в заданном периоде составила 419.15$, а базовая ошибка MAE – 15.53. Далее следует таблица с расчетом ошибки MAE для разных сочетаний параметров «p» и «q». Из данной таблицы видно, что минимальное значение ошибки наблюдается при p=5 и q=0. Это и есть оптимальные параметры, которые нужно использовать с исходным временным рядом при решении задач прогнозирования.

Графики диагностики модели с оптимальными параметрами приведены на рис. 8.16.

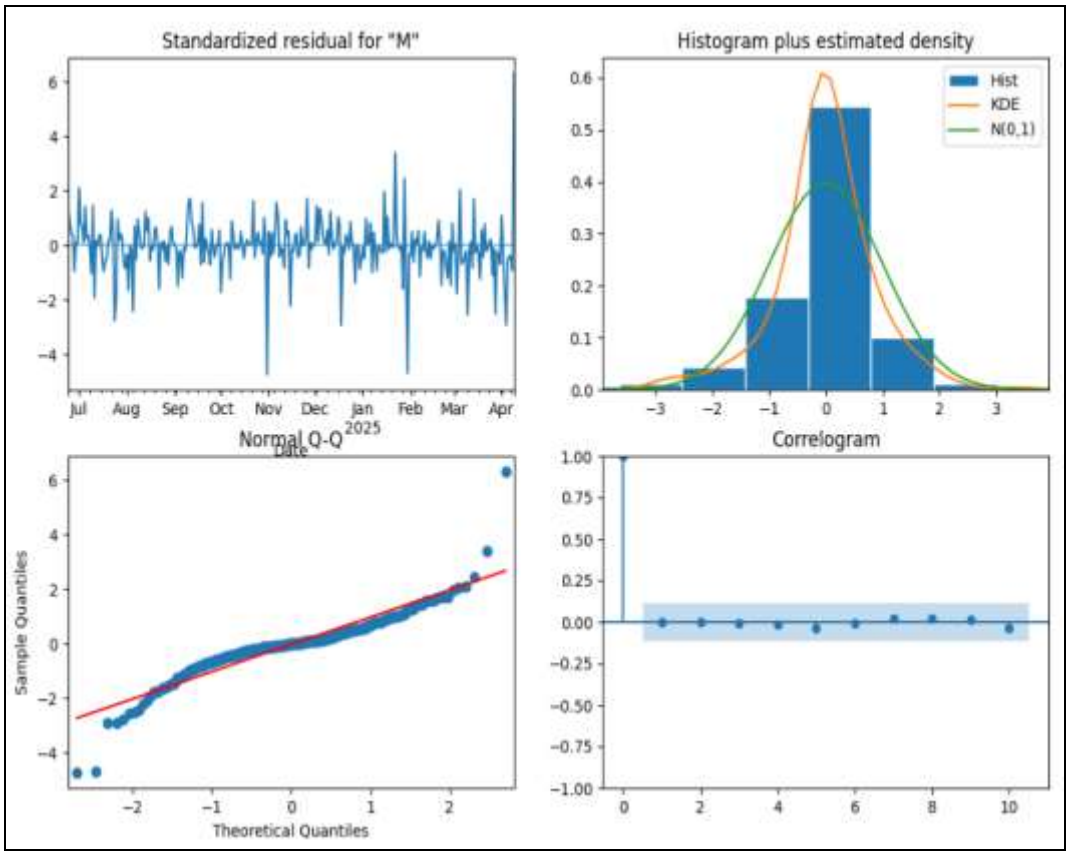


Рис. 8.16. Графики диагностики модели ARMA

На данном рисунке представлено четыре графика. Первый график — это остатки для нашей модели. Эти остатки наглядно демонстрируют стационарность временного ряда – все отклонения центрированы вокруг нуля и рассеивание является однородным.

На втором графике представлена гистограмма остатков. Здесь желательно, чтобы столбцы гистограммы соответствовали зеленой линии нормального распределения. Из данного рисунка видно, что остатки соответствуют нормальному закону распределения, нет явных выбросов и сильной асимметрии.

Третий график – характеризует распределение нормального квантиля (Q-Q plot). Это график, который сравнивает распределение остатков с идеальным нормальным распределением. На нормальном графике QQ все точки должны идеально совпадать с красной линией. Любые значительные отклонения будут означать, что распределение перекошено. Из данного рисунка видно, что точки расположены достаточно близко к красной линии, что опять же подтверждает стационарность временного ряда.

Четвертый график представляет собой коррелограмму, или график ACF остатков. На этом графике мы видим практически идеальную картину. Здесь нулевой лаг соответствует коэффициенту корреляции, равной единице. А далее, по мере увеличения дней запаздывания он резко падает и не выходит за границы области, выделенной голубым цветом.

Итак, мы проанализировали состояния временного ряда, определили оптимальные параметры модели ARMA и убедились, что полученная модель вполне пригодна для решения задач прогнозирования. В следующем разделе мы проверим, как данная модель работает при прогнозе по историческим данным, и при прогнозе на будущие периоды.

8.3.3. Прогнозирование временного ряда с моделью ARMA

Реализуем программный код, который обеспечивает прогнозирование временного ряда стоимости акций компании Microsoft (здесь будут использованы реальные данные из API Stooq). Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 8.26).

Листинг 8.26. Ссылка активации маршрута prg_arma

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем prg_arma (листинг 8.27).

Листинг 8.27. Маршрут URL с именем prg_arma

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление prg_arma (листинг 8.28).

Листинг 8.28. Представление (view) prg_arma

Примечание.
В демо-версии книги листинги программ удалены.

В этом представлении выполнен импорт необходимых библиотек, сформированы даты начала и конца периода в один год (объекты d1 и d2), задан тикер фирмы Microsoft (объект firm) и источник данных (объект sours), временной ряд с динамикой стоимости акций загружен в объект data:

```
data = web.DataReader(firm, sours, start=d1, end=d2)
```

Так как торги в выходные дни не производятся, то на следующем шаге было проведено восстановление всех пропущенных дат с формированием нового набора данных (объект df):

```
df = data["Close"].resample("1D").mean().fillna(method="ffill")
```

Затем заданы параметры для модели ARMA:

```
order = (5, 0, 0)
```

Здесь параметр «p» равен 5, а параметр «q» равен нулю.

Затем создана модель ARMA и выполнено ее обучение на исторических данных:

```
model = ARIMA(endog=df['MSFT'], order=order)
results = model.fit()
```

Обученная модель сохранена в объекте results.

На следующем шаге была выполнена оценка модели на корректность описания исторических данных. Для этого был задан период, который совпадает с периодом исторических данных, и с использованием обученной модели выполнен прогноз изменения цены акций в этом периоде:

```
start_date = '2024-06-20'
end_date = '2025-06-20'
df['prediction'] = results.predict(start=start_date, end=end_date)
```

Фактически здесь к существующему набору данных df добавлена новая колонка с именем prediction. На этих данным можно проверить, насколько обученная модель адекватно описывает динамику изменения цен на акции.

Затем с использованием обученной модели сделан прогноз изменения цен на акции на 15 дней, следующие за последней датой полученной выборки данных. Для этого сначала был сформированы индексные данные будущего периода (объект predict_index):

```
forecast_steps = 15
predict_index = pd.date_range(start=df['Close'].index[-1], periods=forecast_steps + 1,
freq='D')[1:]
```

Затем создан набор данных типа DataFrame (объект df_ind), в котором в колонку Date были загружены даты будущего периода:

```
df_ind = pd.DataFrame({'Date': predict_index})
```

После этого с использованием обученной модели MA создан прогноз динамики цен на акции на будущий период в 15 дней (объект forecast):

```
forecast = results.forecast(steps=forecast_steps)
```

Фактически здесь создается объект типа Series.

Теперь необходимо объединить набор данных df_ind (даты будущего периода) с данными прогноза на этот же период (объект forecast).

```
df_predict = forecast.to_frame().reset_index()
extracted_col = df_predict["predicted_mean"]
df_ind = pd.concat([df_ind, extracted_col.rename("prog_noz")], axis=1)
```

Здесь объект forecast преобразуется из Series в DataFrame (объект df_predict). Затем из полученного набора данных df_predict извлекается колонка с данными прогноза (объект extracted_col). Колонка с данными прогноза добавляется к набору данных с будущим периодом (к объекту df_ind). Таким образом, объект df_ind получил две колонки: Date – с датами будущего периода, prog_noz – с прогнозируемыми значениями цен акций на каждый день будущего периода.

Затем на основе наборов данных df и df_ind строятся графики трех линий:

- динамика цен на акции на основе исходных исторических данных (синяя линия);
- прогноз цен на акции на основе модели ARMA по фрагменту исторических данных (красная линия);
- прогноз цен на акции на будущий период (зеленая линия).

```
fig = go.Figure()
fig.add_trace(go.Scatter(x=df.index, y=df['MSFT'], name="Акции AMD"))
fig.add_trace(go.Scatter(x=df.index, y=df['prediction'], name="Модель ARMA"))
fig.add_trace(go.Scatter(x=df_ind['Date'], y=df_ind['prog_noz'], name="Прогноз"))
```

Полученный график приводится к формату, в котором он может быть отображен в шаблоне Django (объект plot_div).

Кроме того, набор данных с прогнозом на будущий период (объект df_ind, первые 10 строк) приведен к формату, в котором он может быть отображен в шаблоне Django (объект prog_noz_html)

Полученный график, заголовок страницы (объект title_str), и данные прогноза (объект prog_noz_html) через контекстную переменную context передаются в шаблон fig_prog_noz.html (листинг 8.29).

Листинг 8.29. Шаблон fig_prog_noz.html

Примечание.
В демо-версии книги листинги программ удалены.

Это достаточно простой шаблон с контейнером, в котором имеется три блока: для вывода строки заголовка страниц, для вывода графика, для вывода таблицы с данными прогноза на будущий период. Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 8» и активировать ссылку «Прогноз на модели ARMA».

Если выполнить набор вышеприведенных модулей, то в шаблон Django будет выведен график с историческими данными и данными прогноза на модели ARMA (рис. 8.17).

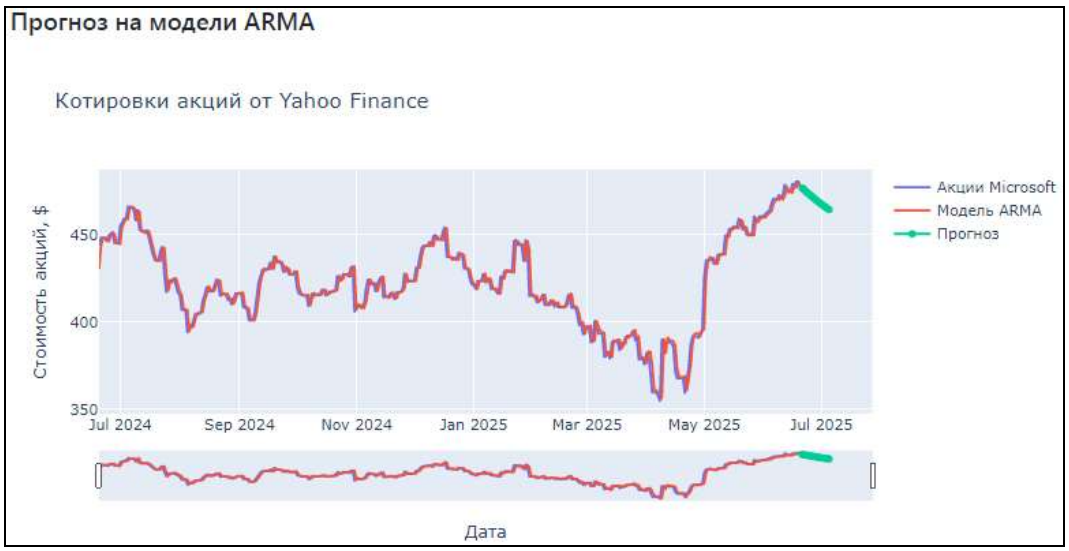


Рис. 8.17. График исторических данных стоимости акций и прогнозы с моделью ARMA

На данном графике **синей линией** показаны исторические данные о стоимости акций компании Micrisoft. Эти данные получены в режиме on-line через API Yahoo Finance. На этих данных было проведено обучение модели ARMA.

Красной линией показан прогноз изменения цены на акции по историческим данным. Из данных графиков видно, что синяя и красная линии практически совпадают, то есть модель ARMA вполне адекватно описывает реальные данные и может быть использована для составления прогнозов на будущие периоды.

Зеленой линией показан прогноз изменения цен на акции на будущие периоды. Из данного графика видно, что в ближайшие 15 дней ожидается падение цен на акции компании Microsoft.

Под графиком имеется слайдер, с помощью которого можно изменить интервал дат и посмотреть более детальную информацию о динамике цен на акции. Изменим интервал дат и сместим слайдер ближе к будущему периоду (рис. 8.18).

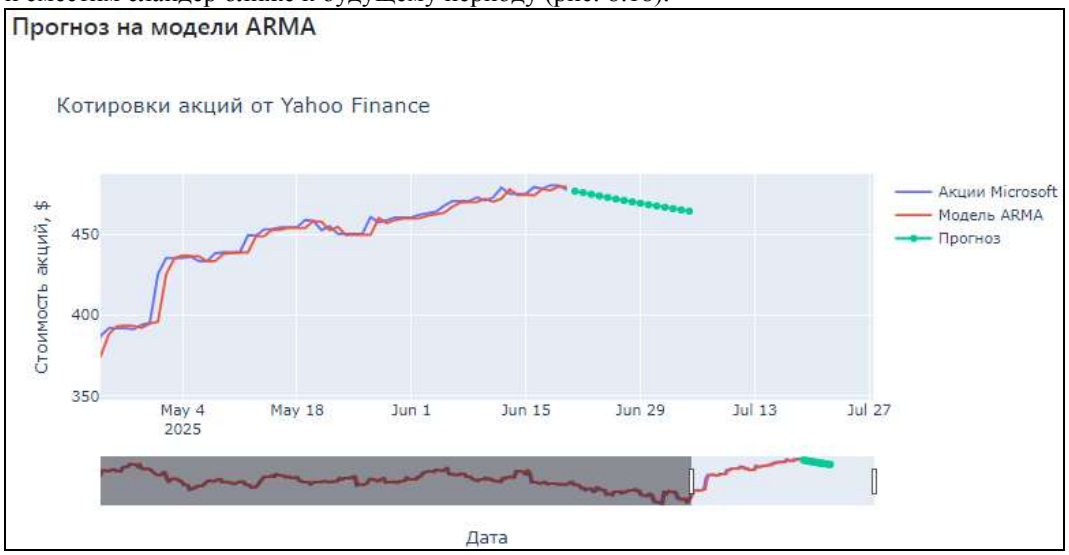


Рис. 8.18. Укрупненный график исторических данных стоимости акций и прогнозы с моделью ARMA

Как видно из данного рисунка, красная линия прогноза с минимальными отклонениями повторяет характер изменения синей линии с реальными данными. Это говорит о том, что прогнозу на будущий период, который отражен зеленой линией, вполне можно доверять. А согласно этому прогнозу в предстоящие две недели должно произойти падение цен на акции.

В шаблоне fig_prog_noz.html также выведены первые 10 строк набора данных с прогнозом цен на будущий период (рис. 8.19).

Прогноз цен на будущий период

	Date	prog_noz
0	2025-06-21	476.587316
1	2025-06-22	475.730570
2	2025-06-23	474.647003
3	2025-06-24	473.643631
4	2025-06-25	472.708654
5	2025-06-26	471.776036
6	2025-06-27	470.861137
7	2025-06-28	469.970751
8	2025-06-29	469.099807
9	2025-06-30	468.247244

Рис. 8.19. Прогноз цен на акции с моделью ARMA в табличном виде

В таком виде данные могут быть полезны в тех случаях, когда необходимо более детально познакомится уровнем прогнозируемых цен на конкретную дату.

8.4. Трехкомпонентные модели ARIMA для анализа и прогнозирования временных рядов

В мире данных, где точность предсказаний может привести аналитика к вершине успеха, статистическая модель ARIMA выступает как мощный инструмент прогноза изменения финансово-экономических показателей. Представьте себе финансового аналитика, который точно предсказал колебания курсов акций, или ритейлера, оптимизировавшего товарные запасы благодаря точному прогнозу спроса. За каждым таким успехом часто стоит авторегрессионная интегрированная модель скользящего среднего (ARIMA) — математически элегантный, но мощный подход к раскрытию тайн временных рядов.

Модель ARIMA — это статистическая модель, используемая для прогнозирования временных рядов, которая объединяет три компонента: авторегрессию (AR), интеграцию (I) и скользящее среднее (MA).

Авторегрессия - Autoregression (AR). Этот компонент моделирует зависимость между наблюдением и несколькими запаздывающими наблюдениями. Он основан на идее, что прошлые значения временного ряда можно использовать для прогнозирования будущих значений. Порядок авторегрессии, обозначаемый буквой «p», определяет количество запаздывающих наблюдений (шагов назад от даты текущего наблюдения).

Интеграция - Integration (I). Этот компонент обрабатывает нестационарные данные временных рядов, удаляя тренды и сезонность. Порядок интеграции, обозначаемый буквой «d», — это количество раз, которое необходимо продифференцировать исходные данные временного ряда, чтобы сделать их стационарными, то есть устранить тренд и сезонность.

Скользящее среднее - Moving Average (MA). Этот компонент моделирует зависимость между остаточными ошибками временного ряда после применения компонентов «AR» и

«I». Порядок скользящего среднего, обозначаемый буквой «q», определяет количество запаздывающих остаточных ошибок, используемых в качестве предикторов.

Модель ARIMA — это расширение моделей типа ARMA для нестационарных временных рядов, которые можно привести к стационарности путем дифференцирования ряда. Общая форма данной модели — ARIMA (p, d, q), где параметры p, d и q — это порядки авторегрессии, интегрирования и скользящего среднего соответственно. Чтобы использовать модель ARIMA для прогнозирования, необходимо сначала определить значения p, d и q, которые лучше всего соответствуют данным. Это можно сделать с помощью процесса, известного как подбор параметров модели ARIMA (с разными комбинациями p, d и q) и выбор модели с наименьшей ошибкой прогноза.

8.4.1. Анализ временного ряда и оптимизация модели ARIMA методом auto_arima

Рассмотрим пример программного кода, в котором будет выполнен анализ временного ряда с динамикой экспорта нефти. В этом примере мы определим, какая комбинация параметров p, d и q, даст наилучшую производительность и точность модели ARIMA, а для этого нужно перебрать все возможные комбинации параметров. Такая процедура называется подбор параметров модели по сетке. Этот метод мы применяли в предыдущем разделе для модели ARMA с использованием вложенных циклов. Но для модели ARIMA имеется более простой способ подбора оптимальных параметров модели с использованием метода auto_arima библиотеки pmdarima. Эту библиотеку можно установить с помощью следующей команды:

```
pip install pmdarima
```

При использовании этого метода оптимизация параметров модели программируется всего одно строкой программного кода.

Реализуем процедуру анализа и оптимизации параметров модели ARIMA в программном коде. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 8.30).

Листинг 8.30. Ссылка активации маршрута analiz_arima

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем analiz_arima (листинг 8.31).

Листинг 8.31. Маршрут URL с именем analiz_arima

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление analiz_arima (листинг 8.32).

Листинг 8.32. Представление (view) analiz_arima

Примечание.
В демо-версии книги листинги программ удалены.

В этом программном модуле выполнен импорт необходимых библиотек и задан путь к CSV файлу, в котором содержится временной ряд с данными об экспорте нефти (объект path_csv). Затем временной ряд загружен в объект series:

```
series = get_data_csv.get_data(path_csv)
```

На основе этих данных сформирован график динамики изменения экспорта нефти (объект fig). Для анализа структуры данных они выведены в окно терминала.

Затем с использованием метода auto_arima библиотеки pmdarima запущен процесс выбора оптимальных параметров модели ARIMA:

```
order = pm.auto_arima(series, max_order=5, seasonal=True, m=12)
```

Этому методу переданы следующие аргументы:

- series – исходный временной ряд;
- max_order=5 – максимальный общий порядок модели (по умолчанию 5);
- seasonal=True – проверка наличия сезонности во временном ряде и учет сезонности (по умолчанию True);
- m=12 – число наблюдений за цикл (по умолчанию 1).

Здесь max_order это максимальное значение суммы параметров (p+q+P+Q), если выбор модели не является пошаговым. Если сумма параметров p и q больше значения max_order, то модель не будет соответствовать этим параметрам, но перейдет к следующей комбинации. Значение по умолчанию — 5. Если max_order=None, это означает, что ограничений на максимальный порядок нет.

Параметр m относится к числу периодов в каждом сезоне (период для сезонной дифференциации). Например, m=4 для квартальных данных, m=12 для ежемесячных данных или m=1 для годовых (несезонных) данных. Значение по умолчанию 1. Обратите внимание, что если m== 1 (т. е. нет сезонности), то seasonal будет установлено значение False.

После того, как будут подобраны оптимальные параметры, они сохранятся в объекте order. В следующих инструкциях параметры модели p, d и q извлекаются из объекта order:

```
get_param = order.get_params()
param = get_param.get('order')
p, d, q = param[0], param[1], param[2]
```

Далее на основе полученных оптимальных параметров создается модель ARIMA (объект model) и выполняется ее обучение на исторических данных:

```
model = ARIMA(series, order=(p, d, q))
result = model.fit()
```

Обученная модель сохраняется в объекте result. После этого с использованием метода plot_diagnostics формируются графики анализа модели ARIMA, прошедшей обучение на исторических данных исходного временного ряда:

```
result.plot_diagnostics(figsize=(12, 8))
```

Этот график формируется с использованием библиотеки matplotlib, которая подключена к приложению в режиме backend.

Затем график анализа (диагностики) модели приводится к формату, в котором он может быть отображен в шаблоне Django (объект graphic), к этому же формату приводится график динамики экспорта нефти (объект plot_div).

Полученные графики, заголовок страницы (объект title_str), и параметры модели (объекты p, d, и q) через контекстную переменную context передается в шаблон analiz_arima.html (листинг 8.33).

Листинг 8.33. Шаблон analiz_arima.html

Примечание.
В демо-версии книги листинги программ удалены.

Это достаточно простой шаблон с контейнером, в котором имеется три блока: для вывода строки заголовка страницы и параметров модели, графика исходного временного ряда, графиков с диагностикой модели ARIMA. Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 8» и активировать ссылку «Анализ параметров модели ARIMA».

Если выполнить набор вышеприведенных модулей, то в шаблон Django будет выведен график с историческими данными временного ряда (рис. 8.20).

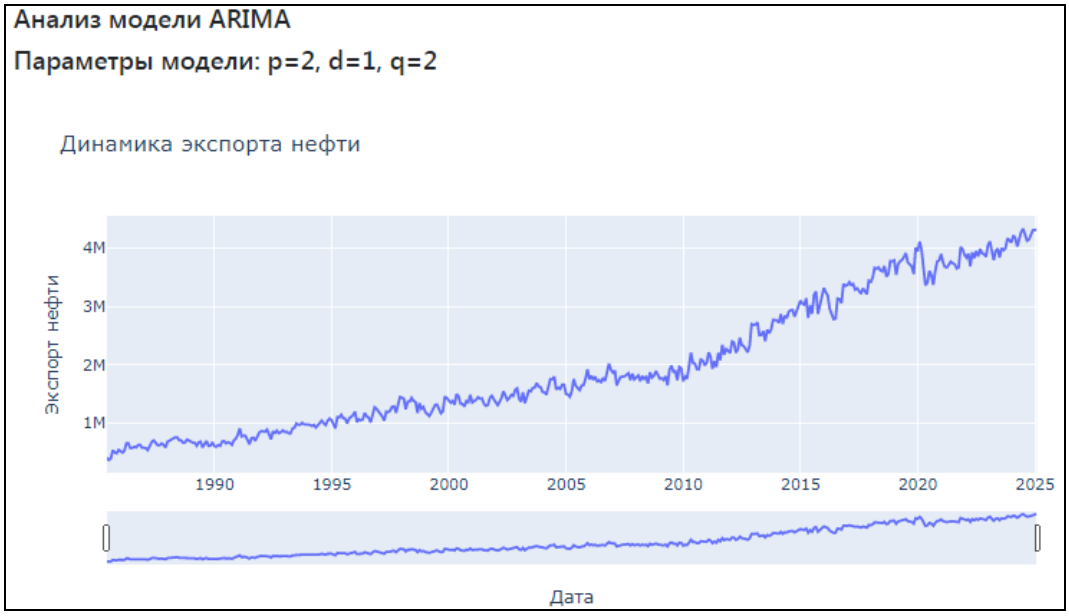


Рис. 8.20. График временного ряда с динамикой изменения объема экспорта нефти

Из данного рисунка видно, что объемы экспорта нефти имеют некоторые колебания, сезонность явно не прослеживается, но имеется устойчивая тенденция к росту.
График анализа модели ARIMA представлен на рис. 8.21.

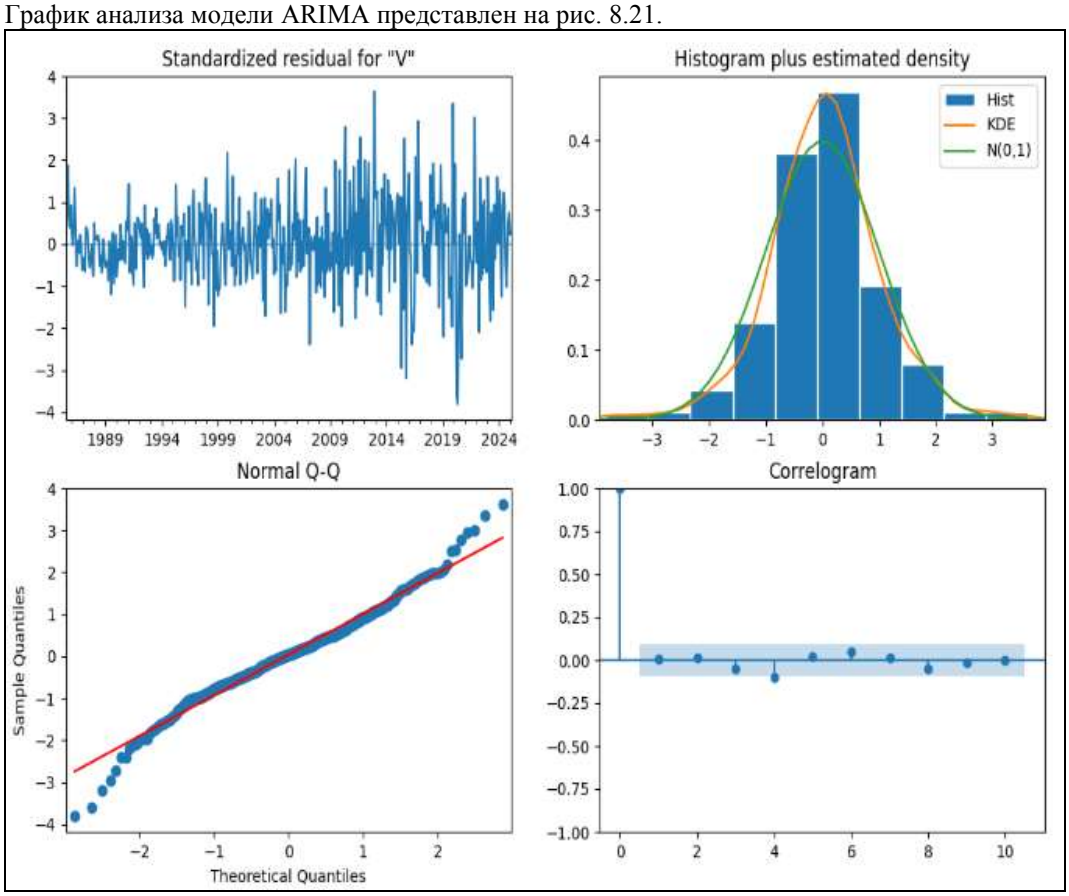


Рис. 8.21. Графики диагностики модели ARIMA

На данном рисунке представлено четыре графика. Первый график — это остатки временного ряда. Остатки наглядно демонстрируют стационарность временного ряда, все отклонения центрированы вокруг нуля и рассеивание является однородным, хотя и наблюдаются некоторое изменение дисперсии.

На втором графике представлена гистограмма остатков. Здесь желательно, чтобы столбцы гистограммы соответствовали зеленой линии нормального распределения. Из данного рисунка видно, что остатки соответствуют нормальному закону распределения, нет явных выбросов и сильной асимметрии.

Третий график – характеризует распределение нормального квантиля (Q-Q plot). Это график, который сравнивает распределение остатков с идеальным нормальным распределением. На нормальном графике QQ все точки должны идеально совпадать с красной линией. Любые значительные отклонения будут означать, что распределение перекошено. Из данного рисунка видно, что точки расположены достаточно близко к красной линии, что опять же подтверждает стационарность временного ряда.

Четвертый график представляет собой коррелограмму, или график ACF остатков. На этом графике мы видим практически идеальную картину. Здесь нулевой лаг соответствует коэффициенту корреляции, равной единице. А далее, по мере увеличения дней запаздывания он резко падает и не выходит за границы области, выделенной голубым цветом.

Итак, мы проанализировали состояния временного ряда, определили оптимальные параметры модели ARIMA и убедились, что полученная модель вполне пригодна для решения задач прогнозирования. В следующем разделе мы проверим, как данная модель работает при прогнозе по историческим данным, и при прогнозе на будущие периоды.

8.4.2. Прогнозирование временного ряда с моделью ARIMA

Реализуем программный код, который обеспечивает прогнозирование временного ряда объема экспорта нефти. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 8.34).

Листинг 8.34. Ссылка активации маршрута prg_arima

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем prg_arima (листинг 8.35).

Листинг 8.35. Маршрут URL с именем prg_arima

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление prg_arima (листинг 8.36).

Листинг 8.36. Представление (view) prg_arima

Примечание.
В демо-версии книги листинги программ удалены.

В этом программном модуле выполнен импорт необходимых библиотек и задан путь к CSV файлу, в котором содержится временной ряд с данными об экспорте нефти (объект path_csv). Затем временной ряд загружен в объект series:

```
series = get_data_csv.get_data(path_csv)
```

Из объекта типа Series сформирован объект DataFrame:

```
df = series.to_frame(name='volume')
```

Затем заданы параметры для модели ARIMA:

```
p, d, q = 2, 1, 2
```

Здесь мы не выполняли поиск оптимальных параметров модели, а задали их вручную. Выбор оптимальных параметров модели для данного временного ряда быт выполнен в

программном модуле предыдущего раздела, эти значения и используются в этом программном модуле.

Затем создана модель ARMA и выполнено ее обучение на исторических данных:

```
model = ARIMA(series, order=(p, d, q))
results = model.fit()
```

Обученная модель сохранена в объекте results.

На следующем шаге была выполнена оценка модели на корректность описания исторических данных. Для этого был задан период, который совпадает с частью периода исторических данных (последние 15 лет), и с использованием обученной модели выполнен прогноз изменения объемов экспорта нефти в этом периоде:

```
start_date = '2010-01-01'
end_date = '2025-08-01'
df['prediction'] = results.predict(start=start_date, end=end_date)
```

Фактически здесь к существующему набору данных df добавлена новая колонка с именем prediction. На этих данным можно проверить, насколько обученная модель адекватно описывает динамику изменения исходного временного ряда.

Затем с использованием обученной модели сделан прогноз изменения объема экспорта нефти в течение 12 месяцев, следующих за последней датой полученной выборки данных. Для этого сначала был сформированы индексные данные будущего периода (объект predict_index):

```
forecast_steps = 12
predict_index = pd.date_range(start=df['volume'].index[-1], periods=forecast_steps + 1,
freq='M')[1:]
```

Затем создан набор данных типа DataFrame (объект df_ind), в котором в колонку Date были загружены даты будущего периода:

```
df_ind = pd.DataFrame({'Date': predict_index})
```

После этого с использованием обученной модели ARIMA получен прогноз динамики объемов экспорта нефти на будущий период в 12 месяцев (объект forecast):

```
forecast = results.forecast(steps=forecast_steps)
```

Фактически здесь создается объект типа Series.

Теперь необходимо объединить набор данных df_ind (даты будущего периода) с данными прогноза на этот же период (объект forecast).

```
df_predict = forecast.to_frame().reset_index()
extracted_col = df_predict["predicted_mean"]
df_ind = pd.concat([df_ind, extracted_col.rename("prog_noz")], axis=1)
```

Здесь объект forecast преобразуется из Series в DataFrame (объект df_predict). Затем из полученного набора данных df_predict извлекается колонка с данными прогноза (объект extracted_col). Колонка с данными прогноза добавляется к набору данных с будущим периодом (к объекту df_ind). Таким образом, объект df_ind получил две колонки: Date – с датами будущего периода, prog_noz – с прогнозируемыми значениями на каждый день будущего периода.

Затем на основе наборов данных df и df_ind строятся графики трех линий:

- динамика экспорта нефти на основе исходных исторических данных (синяя линия);
- прогноз экспорта нефти на основе модели ARIMA по фрагменту исторических данных (красная линия);

- прогноз экспорта нефти на будущий период (зеленая линия).

```
fig = go.Figure()
fig.add_trace(go.Scatter(x=df.index, y=df['volume'], name="Объем экспорта"))
fig.add_trace(go.Scatter(x=df.index, y=df['prediction'], name="Модель ARIMA"))
fig.add_trace(go.Scatter(x=df_ind['Date'], y=df_ind['prog_noz'], name="Прогноз"))
```

Это интерактивный график, созданный с библиотекой plotly, но на данном графике не показан доверительный интервал, или те пределы, в которых может находиться ошибка прогноза. В библиотеке plotly формирование графика с доверительным интервалом потребует полтора десятка строк дополнительного программного кода. Однако такой график достаточно просто сформировать с использованием библиотеки matplotlib, чем мы и воспользуемся (хотя это потребует дополнительного кода для отображения графика в шаблоне Django). В данном примере это график будет находиться в объекте graphic. В дополнение к этим двум графикам здесь выполнен расчет ошибки прогноза (объект mape):

```
forecasted = pred.predicted_mean[:'2025-08-01']
actual = series['2010':]
mape = np.mean(np.abs((actual - forecasted) / actual)) * 100
mape = round(mape, 2)
```

График, построенный в plotly, и набор данных с прогнозом на будущий период (объект df_ind) приведены к формату, в котором они могут быть отображены в шаблоне Django (объекты plot_div и prog_noz_html).

Полученные графики, заголовок страницы (объект title_str), данные прогноза (объект prog_noz_html), параметры модели и рассчитанная ошибка прогноза через контекстную переменную context передается в шаблон prg_arima.html (листинг 8.37).

Листинг 8.37. Шаблон prg_arima.html

> ***Примечание.***
> В демо-версии книги листинги программ удалены.

Это достаточно простой шаблон с контейнером, в котором имеется четыре блока:

- для вывода строки заголовка страниц и параметров модели;
- для вывода графика исторических данных и прогноза, созданных в plotly;
- для вывода графика исторических данных и прогноза, созданных в matplotlib;
- для вывода таблицы с данными прогноза на будущий период.

Теперь можно проверить работу данных модулей.

> ***Примечание.***
> В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 8» и активировать ссылку «Прогноз на модели ARIMA».

Если выполнить набор вышеприведенных модулей, то в шаблон Django будет выведен график с историческими данными и данными прогноза на модели ARIMA, построенный с библиотекой plotly (рис. 8.22).



Рис. 8.22. График исторических данных и прогноз объемов экспорта нефти на модели ARIMA (библиотека plotly)

На данном графике **синей линией** показаны исторические данные об объемах экспорта нефти. На этих данных было проведено обучение модели ARIMA.

Красной линией показан прогноз изменения объемов экспорта нефти по фрагменту исторических данных, начиная с 2010 года. Из данных графиков видно, что синяя и красная линии практически совпадают, то есть модель ARIMA вполне адекватно описывает реальные данные и может быть использована для составления прогнозов на будущие периоды.

Зеленой линией показан прогноз изменения объемов экспорта нефти на будущие периоды. Из данного графика видно, что в ближайшие 12 месяцев ожидается рост объемов экспорта нефти.

Под графиком имеется слайдер, с помощью которого можно изменить интервал дат и посмотреть более детальную информацию о динамике объемов экспорта. Изменим интервал дат и сместим слайдер ближе к будущему периоду (рис. 8.23).

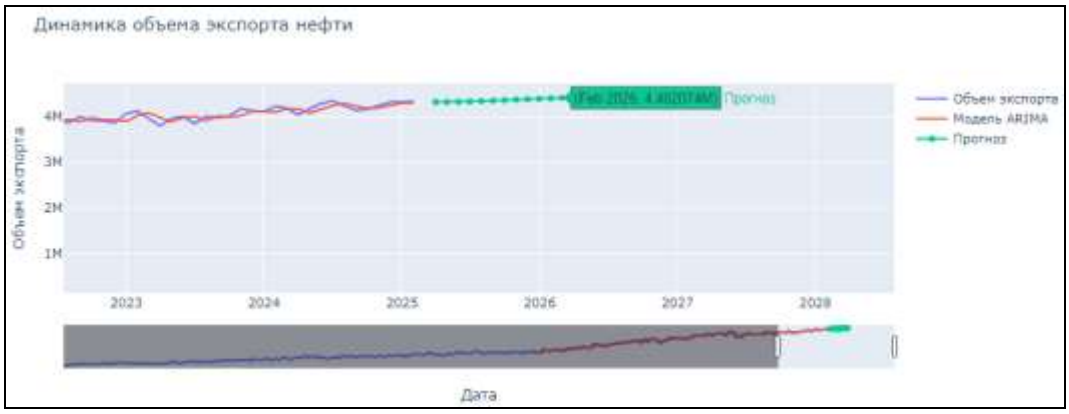


Рис. 8.23. Укрупненный график исторических данных объемов экспорта нефти и прогнозы с моделью ARIMA

Как видно из данного рисунка, красная линия прогноза с минимальными отклонениями повторяет характер изменения синей линии с реальными данными. Это говорит о том, что прогнозу на будущий период, который отражен зеленой линией, вполне можно доверять. А согласно этому прогнозу в предстоящие месяцы должен произойти рост объемов экспорта нефти. Если навести маркер мыши на зеленую точку, то будет подсвечено числовое значение прогноза.

В шаблоне prg_arima.html также выведена таблица с числовыми данными прогноза изменения объемов экспорта нефти на будущий период (рис. 8.24).

Прогноз цен на будущий период

	Date	prog_noz
0	2025-06-21	476.587316
1	2025-06-22	475.730570
2	2025-06-23	474.647003
3	2025-06-24	473.643631
4	2025-06-25	472.708654
5	2025-06-26	471.776036
6	2025-06-27	470.861137
7	2025-06-28	469.970751
8	2025-06-29	469.099807
9	2025-06-30	468.247244

Рис. 8.24. Прогноз изменения объемов экспорта нефти с моделью ARIMA в табличном виде

В таком виде данные могут быть полезны в тех случаях, когда необходимо более детально познакомится уровнем прогнозируемых цен на конкретный месяц.

График с историческими данными и данными прогноза на модели ARIMA, построенный с библиотекой maplotlib представлен на рис. 8.25.

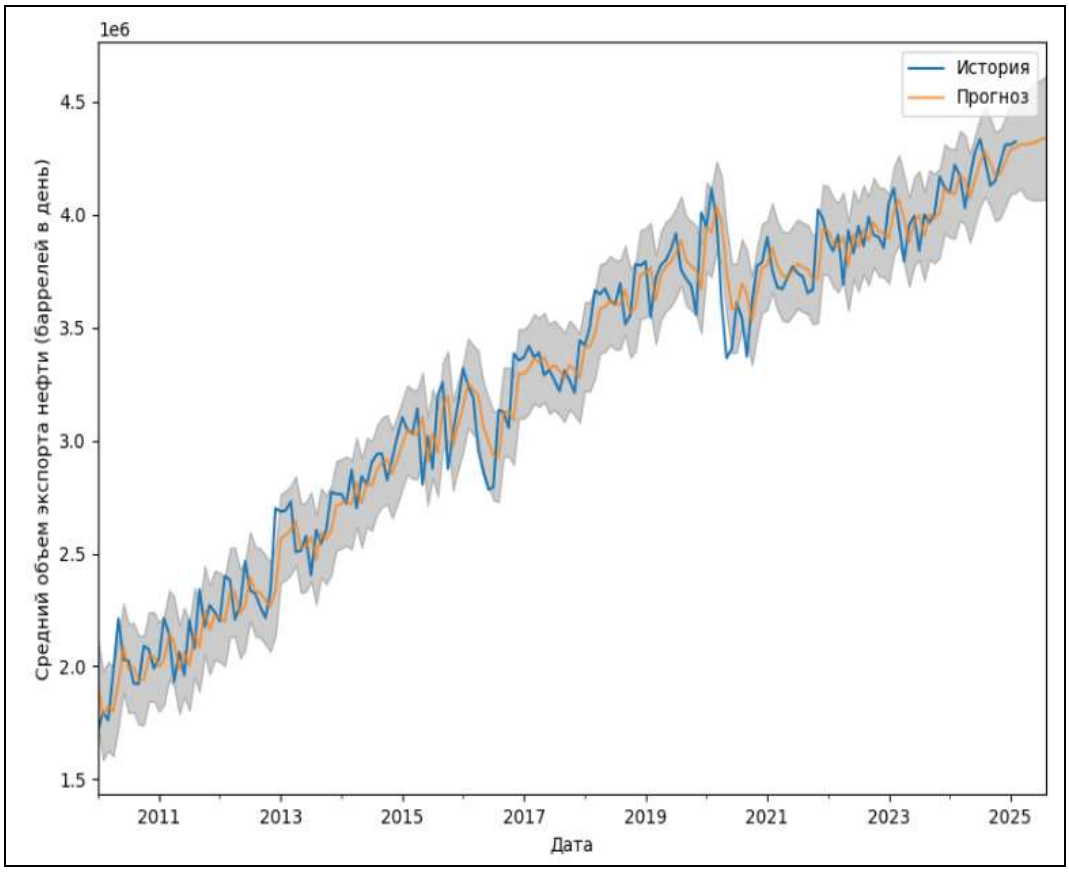


Рис. 8.25. Исторические данные и прогноз объемов экспорта нефти на модели ARIMA (библиотека matplotlib)

На данном графике **синей линией** показан фрагмент исторических данных об объемах экспорта нефти (начиная с 2010 года). На этих данных было проведено обучение модели ARIMA.

Красной линией показан прогноз изменения объемов экспорта нефти по фрагменту исторических данных, начиная с 2010 года. Из данных графиков видно, что синяя и красная линии практически совпадают, то есть модель ARIMA вполне адекватно описывает реальные данные и может быть использована для составления прогнозов на будущие периоды. Красная линия выходит за пределы последней даты исторических данных, показывая прогноз изменения объемов экспорта нефти на будущие периоды.

Серым фоном показана область с возможными ошибками прогноза.

8.5. Модель SARIMA для анализа и прогнозирования временных рядов

Данные временных рядов окружают нас повсюду: от цен на акции и погодных условий до прогнозирования спроса и сезонных тенденций продажи товаров. Мы уже рассмотрели модели AR, MA, ARMA и ARIMA. Теперь рассмотрим следующий тип модели - сезонное авторегрессионное интегрированное скользящее среднее, или SARIMA. В этом разделе рассмотрим использованием моделей SARIMA для прогнозирования загрязнения воздуха.

Модель SARIMA, расшифровывается как «сезонное, авторегрессионное, интегрированное скользящее среднее», является универсальной и широко используемой моделью прогнозирования временных рядов. Фактически модель SARIMA это расширение **несезонной** модели ARIMA, которая предназначена для работы с данными, имеющими сезонные закономерности. Модель SARIMA учитывает как краткосрочные, так и долгосрочные зависимости в данных, что делает его надежным инструментом для прогнозирования. Она объединяет концепции авторегрессионных (AR), интегрированных (I) и скользящих средних (MA) моделей с сезонными компонентами.

Чтобы разобраться с моделью SARIMA, рассмотрим ее четыре компоненты.

44. **Сезонный компонент (S)**. Буква «S» в SARIMA обозначает сезонность, которая относится к повторяющимся закономерностям в данных. Это может быть ежедневный, ежемесячный, ежегодный или любой другой регулярный интервал. Выявление и моделирование сезонного компонента — ключевое преимущество SARIMA.
45. **Авторегрессионный (AR)** компонент. Символами «AR» в SARIMA обозначается авторегрессионный компонент, который моделирует взаимосвязь между текущей точкой данных и ее прошлыми значениями. Он фиксирует автокорреляцию данных, то есть то, насколько данные коррелируют сами с собой с течением времени.
46. **Интегрированный (I)** компонент. Буква «I» в SARIMA указывает на дифференцирование, которое преобразует нестационарные данные в стационарные. Стационарность имеет решающее значение для моделирования временных рядов. Интегрированный компонент показывает, сколько разностей необходимо для достижения стационарности.
47. **Компонент скользящего среднего (MA)**. Символами «MA» в SARIMA обозначается компонент скользящего среднего, который моделирует зависимость между текущей точкой данных и прошлыми ошибками прогнозирования. Он помогает улавливать краткосрочный шум в данных.

Прежде чем мы перейдем к модели SARIMA, важно понять, что такое сезонное дифференцирование. Сезонное дифференцирование — это процесс вычитания данных временного ряда с задержкой, равной сезонности. Это помогает удалить сезонную составляющую и делает данные стационарными, что упрощает моделирование. Сезонное дифференцирование в SARIMA часто обозначается буквой «D». В литературе можно встретить следующие обозначения параметров модели SARIMA:

SARIMA(p, d, q)(P, D, Q, s)
Здесь:

- p - авторегрессионный компонент AR(p);
- d – интегрированный компонент I(d);
- q - компонент скользящего среднего MA(q):

- P – сезонная авторегрессионная составляющая AR(P);
- D – сезонная интегрированная составляющая I(D);
- Q – сезонная составляющая скользящего среднего MA(Q);
- s – сезонный период.

Модели SARIMA оцениваются путем подгонки параметров модели к историческим данным, а затем используется для прогнозирования будущих значений в следующих областях:

- экономика - прогнозирование экономических показателей, таких как инфляция и ВВП;
- розничная торговля - прогнозирование продаж и спроса на сезонные товары;
- энергетика - прогнозирование потребления и спроса на энергию;
- здравоохранение - прогнозирование вспышек заболеваний и госпитализации пациентов;
- финансы - прогнозирование цен на акции и рыночных тенденций.

8.5.1. Предварительный анализ временных рядов на наличие сезонности

Прежде чем использовать модель SARIMA нужно убедиться, что временной ряд имеет сезонность. Для определения наличия сезонности используются метод декомпозиции, или разложения временного ряда на составляющие компоненты. Выполним предварительный анализ временного ряда с данными о загрязнении воздуха выхлопными газами автомобилей. Для этого воспользуемся тестовым набором данных datasets.co2, который входит в состав библиотеки statmodels. Этот временной ряд часто используется в учебных целях, так как он содержит явно выраженный тренд и сезонность, но давайте лишний раз убедимся в этом. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 8.38).

Листинг 8.38. Ссылка активации маршрута seek_season

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем seek_season (листинг 8.39).

Листинг 8.39. Маршрут URL с именем seek_season

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление seek_season (листинг 8.40).

Листинг 8.40. Представление (view) seek_season

Примечание.
В демо-версии книги листинги программ удалены.

В этом программном модуле выполнен импорт необходимых библиотек и из библиотеки statmodels загружены ежедневные данные о динамике загрязнения воздуха (объект df). В этом наборе более двух тысяч записей, в которых имеются пропуски данных за некоторые даты. Нам необходимо укрупнить эти данные, усреднив показатели до среднемесячных. Тем самым мы сможем выявить наличие сезонности во временном ряде в разрезе месяцев. Это делается в следующих инструкциях:

```
df = df['co2'].resample('MS').mean()
s = df.fillna(df.bfill())
df = s.to_frame(name='co2')
```

В итоге был сформирован нужный нам набор данных (объект df).

К этому набору данных была применена функция декомпозиции, в результате чего было получено четыре набора данных:

- observed – исходный временной ряд;
- trend – тренд;
- seasonal – сезонность;
- resid – остаток (не регулярная случайная составляющая).

На основе этих наборов данных сформировано 4 графика, которые вложены в единый макет (объект fig). Этот макет графика приведен к формату, в котором он может быть отображен в шаблоне Django (объект plot_div).

Полученные графики и заголовок страницы (объект title_str) через контекстную переменную context передается в шаблон seek_season.html (листинг 8.41).

Листинг 8.41. Шаблон seek_season.html

Примечание.
В демо-версии книги листинги программ удалены.

Это достаточно простой шаблон с контейнером, в котором имеется блок для вывода строки заголовка страниц и блок для вывода графика.

Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 8» и активировать ссылку «Оценка сезонности для модели SARIMA».

Если выполнить набор вышеприведенных модулей, то в шаблон Django будет выведен график с компонентами временного ряда, построенный с библиотекой plotly (рис. 8.26).

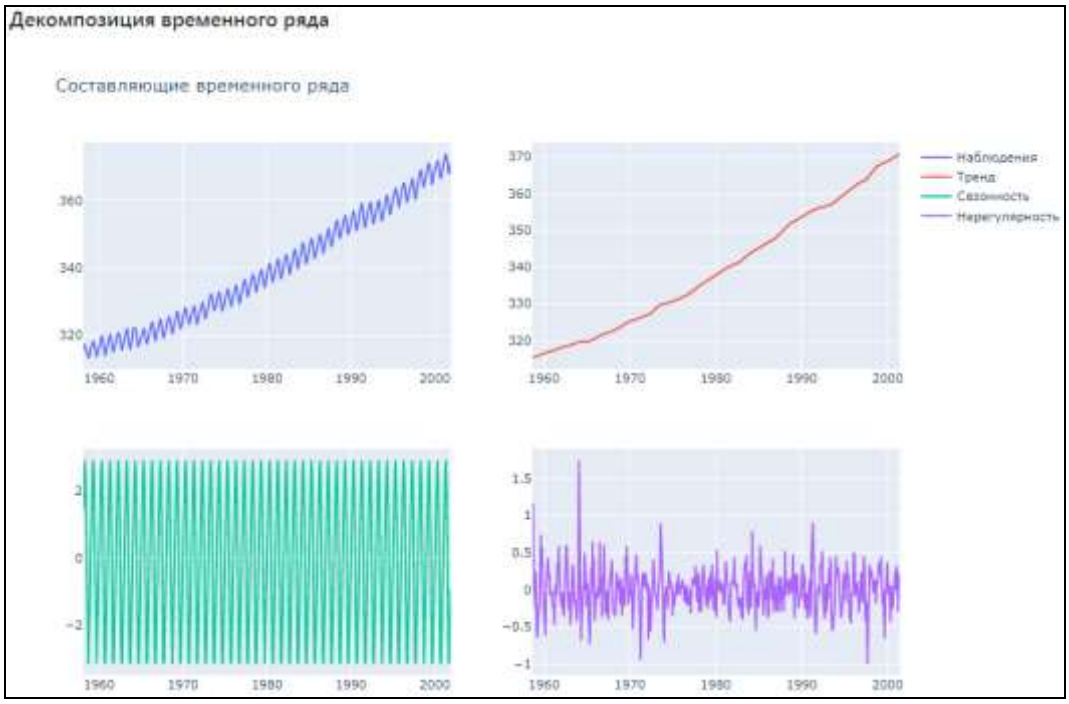


Рис. 8.26. Декомпозиция временного ряда с данными о загрязнении воздуха

На первом графике показан исходный временной ряд. Из данного графика видно наличие периодических колебаний уровня загрязнения воздуха и присутствие тренда. Наличие тренда подтверждается вторым графиком, на лицо устойчивый рост уровня загрязнения. Третий график демонстрирует ярко выраженную сезонность. На четвертом графике показан остаток или шум, который присутствует во временном ряду после удаления тренда и сезонности.
Из этого анализа видно, что к данному временному ряду можно применить модель SARIMA для решения задач прогнозирования.

8.5.2.Анализ временного ряда и оптимизация модели SARIMA методом auto_arima

Рассмотрим пример программного кода, в котором будет выполнен анализ временного ряда с динамикой загрязнения воздуха. В этом примере мы определим, какая комбинация параметров (p, d, q)(P, D, Q, s) даст наилучшую производительность и точность модели SARIMA. Для этого воспользуемся возможностью подбора оптимальных параметров модели SARIMA методом auto_arima библиотеки pmdarima.

Реализуем процедуру анализа и оптимизации параметров модели SARIMA в программном коде. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 8.42).

Листинг 8.42. Ссылка активации маршрута analiz_sarima

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем analiz_sarima (листинг 8.43).

Листинг 8.43. Маршрут URL с именем analiz_sarima

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление analiz_sarima (листинг 8.44).

Листинг 8.44. Представление (view) analiz_sarima

Примечание.
В демо-версии книги листинги программ удалены.

В этом программном модуле выполнен импорт необходимых библиотек и из библиотеки statmodels загружены ежедневные данные о динамике загрязнения воздуха (объект df). В этом наборе более двух тысяч записей, в которых имеются пропуски данных за некоторые даты. Нам необходимо укрупнить эти данные, усреднив показатели до среднемесячных и заполнить данными пропущенные даты.

```
df = df['co2'].resample('MS').mean()
# Заполняем пропуски значениями перед пропущенными значениями
series = df.fillna(df.bfill())
```

Итоговый набор данных был сохранен в объекте series.

На основе этих данных сформирован график динамики изменения загрязнения воздуха (объект fig).

Затем с использованием метода auto_arima библиотеки pmdarima запущен процесс выбора оптимальных параметров модели SARIMA:

```
order = pm.auto_arima(series, max_order=None, seasonal=True, m=12)
```

Этому методу переданы следующие аргументы:

- series – исходный временной ряд;
- max_order=5 – максимальный общий порядок модели (по умолчанию 5);
- seasonal=True – проверка наличия сезонности во временном ряде и учет сезонности (по умолчанию True);
- m=12 – число наблюдений за цикл (по умолчанию 1).

Здесь max_order то максимальное значение суммы параметров (p+q+P+Q), если выбор модели не является пошаговым. Если сумма параметров p и q больше значения max_order, то модель не будет соответствовать этим параметрам, но перейдет к следующей комбинации. Значение по умолчанию — 5. Если max_order=None, это означает, что ограничений на максимальный порядок нет.

Параметр m относится к числу периодов в каждом сезоне (период для сезонной дифференциации). Например, m=4 для квартальных данных, m=12 для ежемесячных данных или m=1 для годовых (несезонных) данных. Значение по умолчанию 1. Обратите внимание, что если m== 1 (т. е. нет сезонности), то seasonal будет установлено значение False.

После того, как будут подобраны оптимальные параметры, они сохранятся в объекте order. В следующих инструкциях параметры модели извлекаются из объекта order:

```
order = pm.auto_arima(series, max_order=None, seasonal=True, m=12)
get_param = order.get_params()
param = get_param.get('order')
p, d, q = param[0], param[1], param[2]
s_param = get_param.get('seasonal_order')
P, D, Q, S = s_param[0], s_param[1], s_param[2], s_param[3]
```

Далее на основе полученных оптимальных параметров создается модель SARIMA (объект model) и выполняется ее обучение на исторических данных:

```
model = sm.tsa.statespace.SARIMAX(series,
                        order=(p, d, q),
                        seasonal_order=(P, D, Q, S),
                        enforce_stationarity=False,
                        enforce_invertibility=False)
result = model.fit()
```

Обученная модель сохраняется в объекте result. После этого с использованием метода plot_diagnostics формируются графики анализа модели SARIMA, прошедшей обучение на исторических данных исходного временного ряда:

```
result.plot_diagnostics(figsize=(12, 8))
```

Этот график формируется с использованием библиотеки matplotlib, которая подключена к приложению в режиме backend.

Затем график анализа (диагностики) модели приводится к формату, в котором он может быть отображен в шаблоне Django (объект graphic), к этому же формату приводится график динамики загрязнения воздуха (объект plot_div).

Полученные графики, заголовок страницы (объект title_str), и параметры модели через контекстную переменную context передается в шаблон analiz_sarima.html (листинг 8.45).

Листинг 8.45. Шаблон analiz_sarima.html

Примечание.
В демо-версии книги листинги программ удалены.

Это достаточно простой шаблон с контейнером, в котором имеется три блока: для вывода строки заголовка страницы и параметров модели, для вывода графика исходного временного ряда, для вывода графиков с диагностикой модели SARIMA. Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 8» и активировать ссылку «Анализ параметров модели SARIMA».

Если выполнить набор вышеприведенных модулей, то в шаблон Django будет выведен график с историческими данными временного ряда (рис. 8.27).

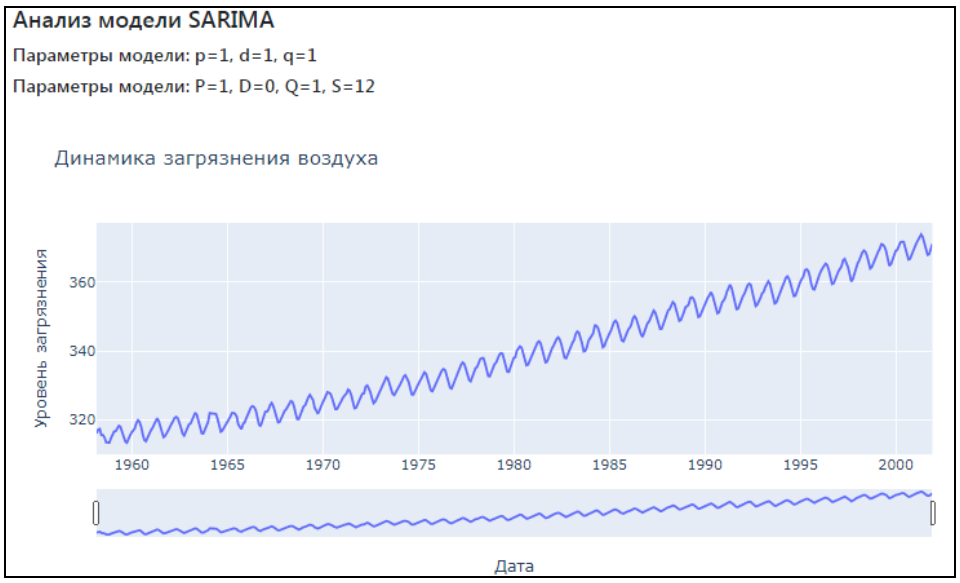


Рис. 8.27. График временного ряда с динамикой изменения загрязнения воздуха

Из данного рисунка видно, что показатели загрязнения воздуха имеют устойчивые колебания, явно выраженную сезонность, имеется устойчивая тенденция к росту.

График анализа модели SARIMA представлен на рис. 8.28.

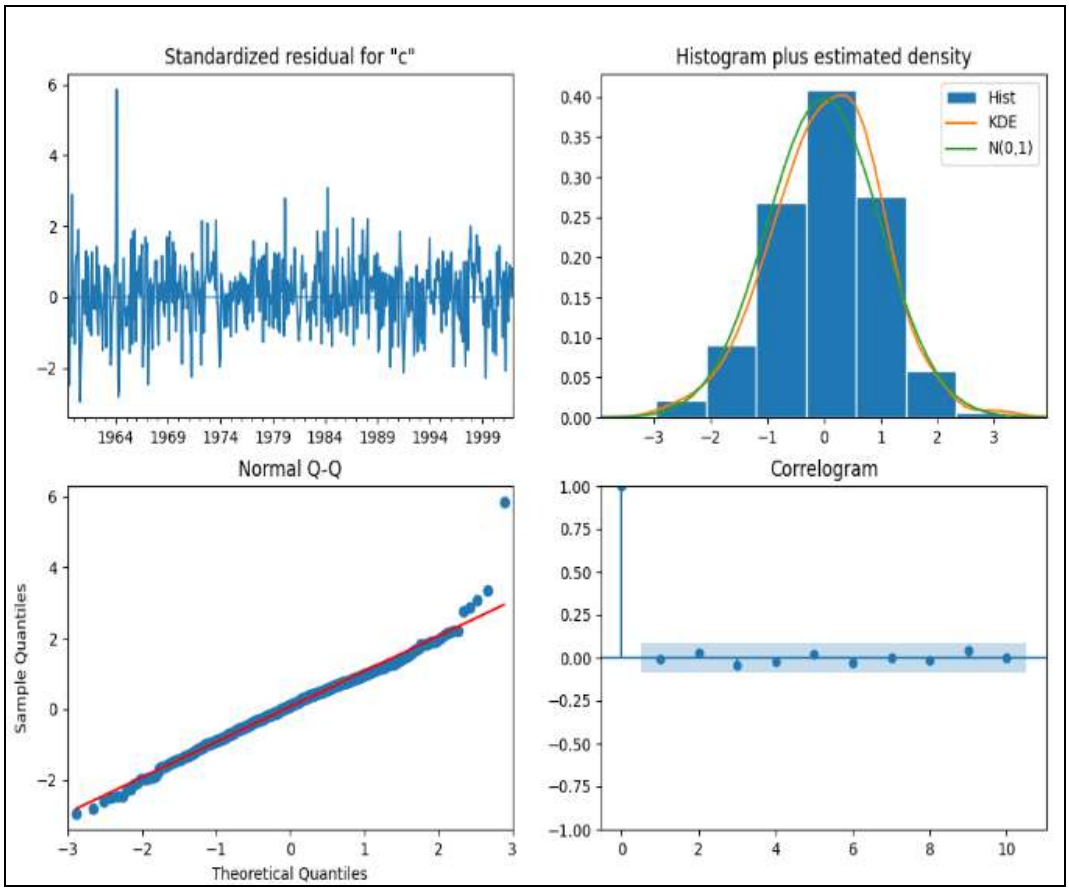


Рис. 8.28. Графики диагностики модели SARIMA

На данном рисунке представлено четыре графика. Первый график — это остатки для нашей модели. Эти остатки наглядно демонстрируют стационарность временного ряда, все отклонения центрированы вокруг нуля и рассеивание является однородным.

На втором графике представлена гистограмма остатков. Здесь желательно, чтобы столбцы гистограммы соответствовали зеленой линии нормального распределения. Из данного рисунка видно, что остатки соответствуют нормальному закону распределения, нет явных выбросов и сильной асимметрии.

Третий график – характеризует распределение нормального квантиля (Q-Q plot). Это график, который сравнивает распределение остатков с идеальным нормальным распределением. На нормальном графике QQ все точки должны идеально совпадать с красной линией. Любые значительные отклонения будут означать, что распределение перекошено. Из данного рисунка видно, что точки практически совпадают с красной линией, что опять же подтверждает стационарность временного ряда.

Четвертый график представляет собой коррелограмму, или график ACF остатков. На этом графике мы видим практически идеальную картину. Здесь нулевой лаг соответствует коэффициенту корреляции, равной единице. А далее, по мере увеличения дней

запаздывания он резко падает и не выходит за границы области, выделенной голубым цветом.

Итак, мы проанализировали состояния временного ряда, определили оптимальные параметры модели SARIMA и убедились, что полученная модель вполне пригодна для решения задач прогнозирования. В следующем разделе мы проверим, как данная модель работает при прогнозе по историческим данным, и при прогнозе на будущие периоды.

8.5.3. Прогнозирование временного ряда с моделью SARIMA

С использованием модели SARIMA можно делать два вида прогноза – статический и динамический:

- **статический прогноз** предполагает использование всего известного набора данных для прогнозирования каждого следующего значения;
- **динамический прогноз** начинает применять самые свежие прогнозные значения для предсказания будущих значений, после определенной пользователем даты старта.

Динамический прогноз обычно предоставляет более реалистичное представление о будущем поведении временного ряда, поскольку он учитывает оценки в процессе прогнозирования.

Реализуем программный код, который обеспечивает прогнозирование временного ряда с данными о загрязнении воздуха. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 8.46).

Листинг 8.46. Ссылка активации маршрута prg_sarima

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем prg_sarima (листинг 8.47).

Листинг 8.47. Маршрут URL с именем prg_sarima

По этому маршруту будет активировано представление prg_sarima (листинг 8.48).

Листинг 8.48. Представление (view) prg_sarima

Примечание.
В демо-версии книги листинги программ удалены.

В этом программном модуле выполнен импорт необходимых библиотек и из библиотеки statmodels загружены ежедневные данные о динамике загрязнения воздуха (объект df). В этом наборе более двух тысяч записей, в которых имеются пропуски данных за некоторые даты. Нам необходимо укрупнить эти данные, усреднив показатели до среднемесячных и заполнить данными пропущенные даты.

```
df = df['co2'].resample('MS').mean()
# Заполняем пропуски значениями перед пропущенными значениями
series = df.fillna(df.bfill())
```

Итоговый набор данных был сохранен в объекте series.

На основе этих данных сформирован график динамики изменения загрязнения воздуха (объект fig).

В этом модуле не был сделан подбор оптимальных параметров модели, так как они уже были найдены в программном модуле предыдущего раздела. Здесь мы просто воспользовались известными значениями параметров, присвоив их значения следующим переменным:

```
p, d, q = 1, 1, 1
P, D, Q, S = 1, 0, 1, 12
```

Затем создана модель SARMA и выполнено ее обучение на исторических данных:

```
model = sm.tsa.statespace.SARIMAX(series,
                                  order=(p, d, q),
                                  seasonal_order=(P, D, Q, S),
                                  enforce_stationarity=False,
                                  enforce_invertibility=False)
results = model.fit()
```

Обученная модель сохранена в объекте results.

На следующем шаге была выполнена оценка модели на корректность описания исторических данных. Для этого был задан период, который совпадает с частью периода исторических данных (с 1990 по 2001 годы), и с использованием обученной модели выполнен статический прогноз изменения степени загрязнения воздуха в этом периоде (параметр dynamic=False):

```
pred = results.get_prediction(start=pd.to_datetime('1990-01-01'),
                              dynamic=False)
pred_ci = pred.conf_int()
```

После этого был сформирован график (объект fig_prg1), на основе которого можно оценить насколько адекватно модель описывает исторические данные с использованием статического прогноза.

На следующем шаге была выполнена оценка модели на корректность описания исторических данных. Для этого был задан период, который совпадает с частью периода исторических данных (с 1998 по 2001 годы), и с использованием обученной модели выполнен динамический прогноз изменения степени загрязнения воздуха в этом периоде (параметр dynamic=True):

```
pred_dynamic = results.get_prediction(start=pd.to_datetime('1998-01-01'),
                                      dynamic=True,
                                      full_results=True)
pred_dynamic_ci = pred_dynamic.conf_int()
```

После этого был сформирован график (объект fig_prg2), на основе которого можно оценить насколько адекватно модель описывает исторические данные с использованием динамического прогноза.

Затем с использованием обученной модели сделан прогноз изменения степени загрязнения воздуха в течение 10 лет, следующих за последней датой полученной выборки данных:

```
pred_uc = results.get_forecast(steps=10 * 12)
pred_ci = pred_uc.conf_int()
```

После этого был сформирован график (объект fig_prg3), на основе которого можно оценить изменение степени загрязнения воздуха на будущие периоды.

График, построенный в plotly, приведен к формату, в котором он может быть отображен в шаблоне Django (объект plot_div). Полученные графики, заголовок страницы и рассчитанная ошибка прогноза через контекстную переменную context передается в шаблон prg_sarima.html (листинг 8.49).

Листинг 8.49. Шаблон prg_sarima.html

Примечание.
В демо-версии книги листинги программ удалены.

Это достаточно простой шаблон с контейнером, в котором имеется пять блоков для вывода графиков и рассчитанных ошибках прогноза. Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 8» и активировать ссылку «Прогноз на модели SARIMA».

Если выполнить набор вышеприведенных модулей, то в шаблон Django будет выведен график с историческими данными и данными прогнозов на модели SARMA. График, динамики изменения степени загрязнения воздуха, построенный на основе исходных исторических данных, представлен на рис. 8.29.

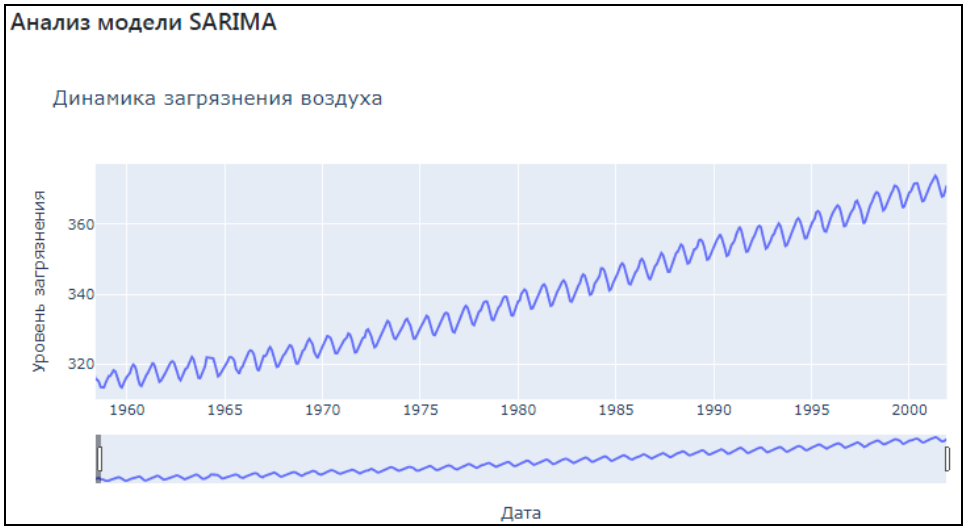


Рис. 8.29. График исторических данных изменения степени загрязнения воздуха на модели SARIMA

На этом графике показана вся выборка данных с 1958 года по 2001 год. Из данного рисунка виден явно выраженный тренд (рост степени загрязнения воздуха) и цикличность (сезонность)

На следующем графике показано оценка модели SARIMA, которая описывает этот временной ряд по историческим данным за период, начиная с 1990 года с использованием статического прогноза (рис. 8.30).

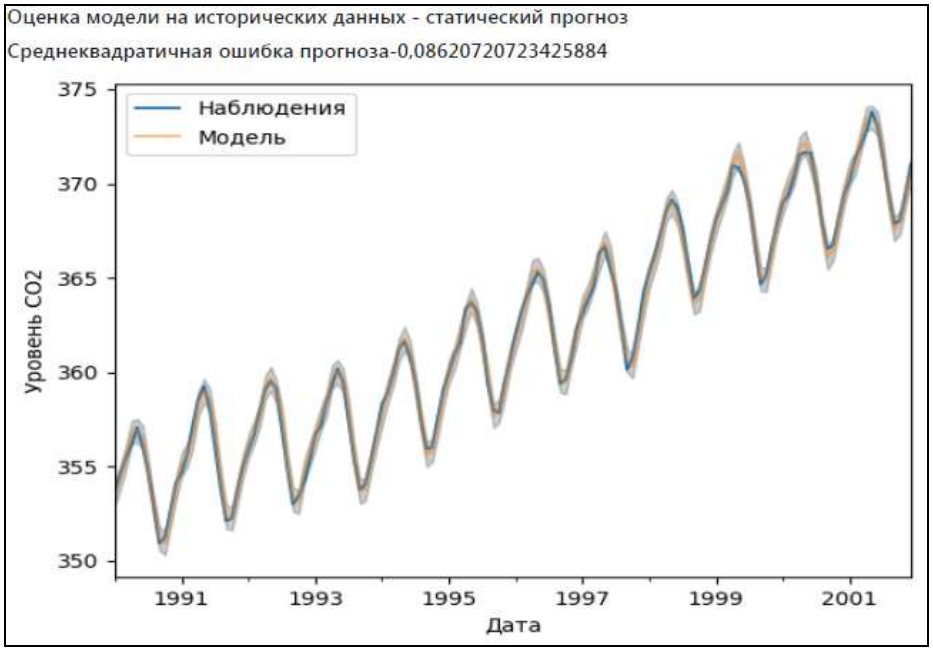


Рис. 8.30. График исторических данных и прогноз загрязнения воздуха на модели SARIMA (статический прогноз)

На данном графике **синей линией** показан фрагмент исторических данных (начиная с 1990 года), а к**расной линией** прогноз изменения загрязнения воздуха по этому фрагменту исторических данных. Здесь для наглядности был урезан период и использован только фрагмент исторических данных (чем меньше период, тем нагляднее будет видна сезонность). Из данного графика видно, что синяя и красная линии практически совпадают, то есть модель SARIMA вполне адекватно описывает реальные данные и может быть использована для составления прогнозов на будущие периоды.

На следующем графике показано оценка модели SARIMA, которая описывает этот временной ряд по историческим данным за период, начиная с 1998 года с использованием динамического прогноза (рис. 8.31).

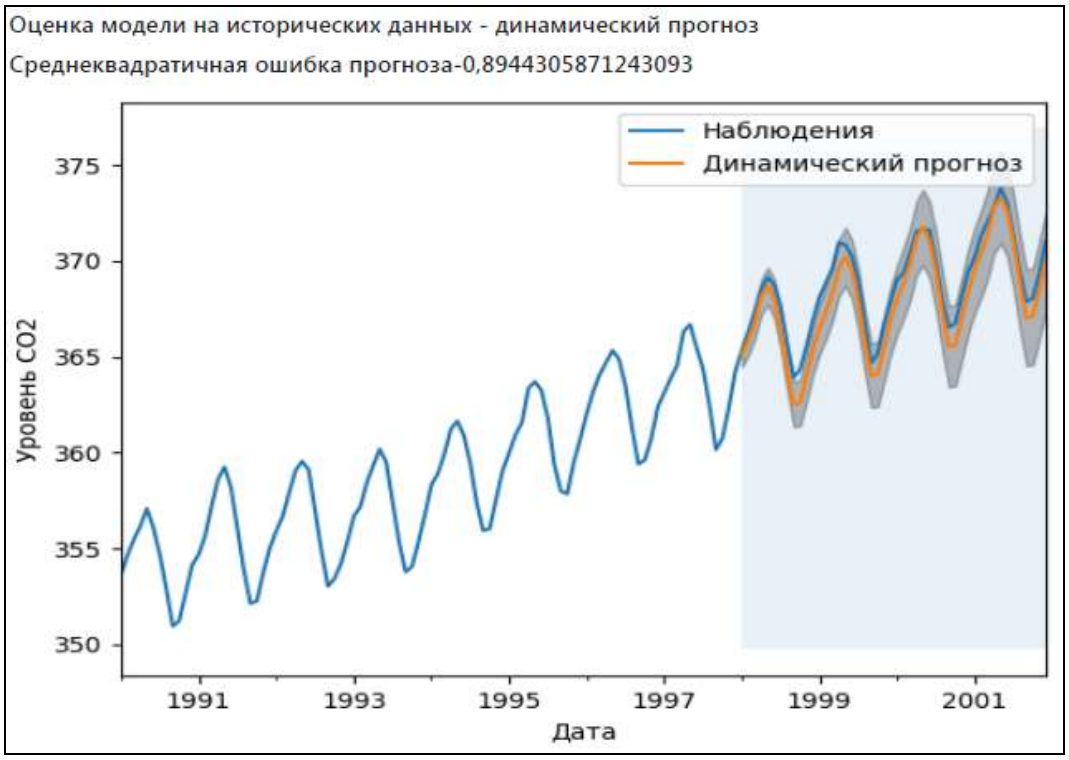


Рис. 8.31. График исторических данных и прогноз загрязнения воздуха на модели SARIMA (динамический прогноз)

Здесь кроме красной и синей линии, которые отражают исторические данные и прогноз, а еще выделена серая область, которая определяет возможную ошибку прогноза.

На следующем графике показан прогноз изменения степени загрязнения воздуха на будущие периоды (на 10 лет после 2001 года), сделанной на модели SARIMA (рис. 8.32).

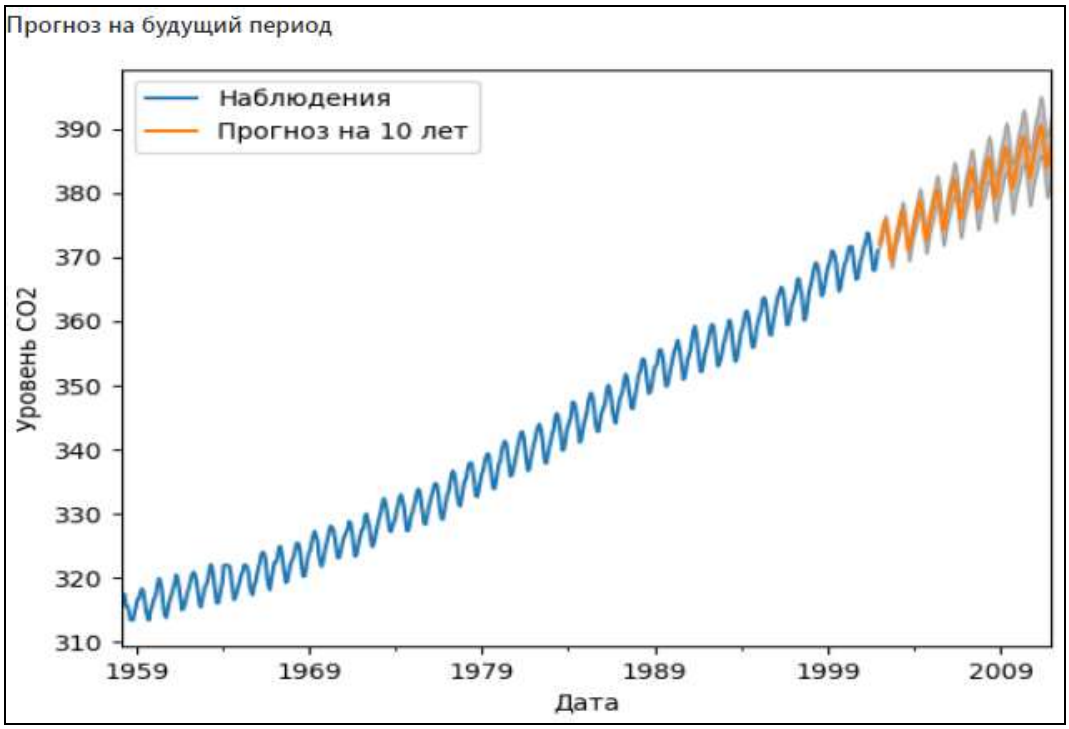


Рис. 8.32. График исторических данных и прогноз загрязнения воздуха на модели SARIMA (будущий период)

На данном графике показаны все исторические данные (за период с 1958 года по 2001 год), и прогноз на период 2001-2011 годы. Здесь была выбрана глубина прогноза 10 лет для того, чтобы показать, как будет зависеть ошибка прогноза о его глубины. На данном графике серым цветом выделена область, отражающая рассеивание возможных ошибок прогноза. На данном рисунке наглядно видно, что по мере увеличения глубины прогноза вероятность получения ошибочного результата увеличивается.

8.6. Модель SARIMAX для анализа и прогнозирования временных рядов

Модель сезонного авторегрессионного интегрированного скользящего среднего с экзогенными регрессорами (SARIMAX) — это мощный метод прогнозирования временных рядов, который расширяет традиционную модель SARIMA, позволяя учитывать и сезонность, и внешние факторы. Это универсальная модель, которая может включать в себя как авторегрессионные (AR), так и скользящие средние (MA) компоненты, интегрирование (дифференцирование) для получения стационарных данных и включать внешние переменные или регрессоры. Модель SARIMAX особенно полезна при работе с данными, зависящими от времени, которые демонстрируют повторяющиеся закономерности в течение определенных временных интервалов.

Модель SARIMAX имеет следующие компоненты.

48. **Сезонный компонент (S).** Этот компонент улавливает периодические закономерности в данных, такие как недельные, месячные или годовые циклы.
49. **Авторегрессионный компонент (AR)**. Этот компонент представляет собой взаимосвязь между текущим значением и предыдущими значениями во временном ряду.
50. **Интегрированный компонент (I)**. Данный компонент включает в себя дифференцирование, предназначенное для преобразования временного ряда в стационарный ряд путем удаления тренда и сезонности.
51. **Компонент скользящего среднего (MA)**. Этот компонент учитывает зависимость текущего значения от прошлых ошибок и используется для вычисления тренда.
52. **Экзогенные регрессоры (X)**. Этот компонент позволяет включать внешние переменные, которые могут влиять на временные ряды.

Таким образом, модель SARIMAX объединяет авторегрессионные и сезонные авторегрессионные компоненты, дифференцирование для обеспечения стационарности и включает экзогенные переменные для учета дополнительных факторов, влияющих на зависимую переменную с течением времени.

Экзогенные переменные – это факторы, влияющие на прогнозируемый ряд, но не являющиеся его частью. Это мощный инструмент, позволяющий увеличить точность прогнозов и получить более глубокое понимание взаимосвязей в данных.

Экзогенные переменные могут быть самыми разными: экономические индикаторы (инфляция, ВВП, курс валюты), погодные условия (температура, осадки), маркетинговые акции по привлечению покупателей и многое другое. Ключевое условие использования экзогенных переменных – наличие данных по этим переменным за тот же период, что и для прогнозируемого ряда. Важно отметить, что экзогенные переменные должны быть стационарными или быть предварительно преобразованы к стационарному виду, подобно тому, как это делается для целевого временного ряда.

Предположим, вы прогнозируете продажи мороженого. В качестве экзогенных переменных можно включить среднесуточную температуру и данные о маркетинговых акциях. Модель SARIMAX будет учитывать сезонные колебания продаж, а также влияние температуры и маркетинговых усилий. Это значительно повысит точность прогноза по сравнению с моделью, не учитывающей эти факторы.

Рассмотрим параметры данной модели и их влияние на результаты прогноза.

Порядок дифференцирования (d и D).

Чрезмерное дифференцирование. Слишком большое количество дифференцирований может привести к тому, что ряд станет «излишне» стационарным, что потенциально приведет к потере ценной информации. Если исходный временной ряд уже демонстрирует стабильный тренд, слишком большое количество дифференцирований (слишком высокие значения d или D) может устранить тренд и сделать данные менее интерпретируемыми.

Недостаточное дифференцирование. Недостаточное дифференцирование может привести к не стационарности ряда, что приведет к неточностям в оценке модели. Если временной ряд имеет явный тренд или сезонность, неправильное дифференцирование данных (слишком низкие значения d или D) может привести к тому, что модель не сможет уловить эти закономерности.

Сезонные и несезонные условия AR (p и P).

Слишком много авторегрессионных членов (p или P). Включение слишком большого количества авторегрессионных членов может привести к переобучению модели, когда шум будет восприниматься как реальная закономерность. Если ежемесячный временной ряд моделируется с большим количеством авторегрессионных членов, модель может улавливать краткосрочные колебания, которые не отражают основную структуру.

Слишком мало авторегрессионных членов (p или P). Недостаточное количество авторегрессионных членов может привести к тому, что модель не сможет выявить важные зависимости между прошлыми и настоящими наблюдениями. Если временной ряд имеет сильную автокорреляцию в прошлом, недостаточное количество авторегрессионных членов может привести к низкой эффективности прогнозирования.

Сезонные и несезонные условия для MA (q и Q).

Слишком большой период скользящего среднего (q или Q). Включение слишком большого периода скользящего среднего может привести к тому, что модель будет слишком чувствительна к краткосрочным колебаниям, что приведет к переобучению. В наборе данных с высокой сезонностью слишком большое количество периодов скользящего среднего может пытаться подогнать шум под реальную закономерность, что приведет к менее надежным прогнозам.

Слишком малый период скользящего среднего (q или Q). Малый период скользящего среднего может привести к неспособности уловить краткосрочные колебания, особенно при наличии шума. Если во временном ряду есть краткосрочные колебания, которые не учитываются, модель может с трудом адаптироваться к внезапным изменениям в данных.

Сезонный период (m).

Неправильное значение сезонного периода. Выбор неправильного значения сезонного периода может привести к тому, что модель будет пытаться выявить закономерности, которых нет в данных. Если сезонный период ошибочно установлен на 6 месяцев, а фактическая закономерность повторяется каждые 12 месяцев, модель может неверно интерпретировать сезонность.

Игнорирование сезонного компонента. Если временной ряд демонстрирует явную сезонность, игнорирование сезонного компонента может привести к тому, что модель не сможет уловить важные закономерности. Данные о ежемесячных продажах часто демонстрируют сезонность, и игнорирование сезонного компонента может привести к неоптимальным прогнозам.

8.6.1.Анализ временного ряда и оптимизация модели SARIMAX методом auto_arima

Рассмотрим пример программного кода, в котором будет выполнен анализ временного ряда с динамикой изменения пассажиропотока. В этом примере мы определим, какая комбинация параметров (p, d, q)(P, D, Q, s) даст наилучшую производительность и точность модели SARIMAX. Для этого воспользуемся возможностью подбора оптимальных параметров модели SARIMA методом auto_arima библиотеки pmdarima. На характер изменения пассажиропотока влияет сезон года (месяц). Поэтому в качестве дополнительной экзогенной переменной можно использовать либо среднемесячную температуру воздуха, либо номер месяца. В данном примере будет использован номер месяца.

Реализуем процедуру анализа и оптимизации параметров модели SARIMAX в программном коде. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 8.50).

Листинг 8.50. Ссылка активации маршрута analiz_sarimax

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем analiz_sarimax (листинг 8.51).

Листинг 8.51. Маршрут URL с именем analiz_sarimax

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление analiz_sarimax (листинг 8.52).

Листинг 8.52. Представление (view) analiz_sarimax

Примечание.
В демо-версии книги листинги программ удалены.

В этом программном модуле выполнен импорт необходимых библиотек, задан путь к CSV файлу «passengers.csv» с данными о пассажиропотоках (объект path_csv). Затем данные из этого файла загружены в объект df и для полученного набора данных сформирована индексная колонка:

```
df = pd.read_csv(path_csv)
df.set_index('Month', inplace=True)
df.index = pd.to_datetime(df.index)
```

После этого к полученному набору данных добавлена колонка «month_index» с экзогенной переменной:

```
df['month_index'] = df.index.month
```

В учебных целях этот набор данных выведен в окно терминала, и давайте посмотрим на его структуру:

```
            Passengers  month_index
Month
2015-01-01         145            1
2015-02-01         150            2
2015-03-01         178            3
2015-04-01         163            4
2015-05-01         172            5
...                ...          ...
2024-08-01         606            8
2024-09-01         508            9
```

```
2024-10-01      461         10
2024-11-01      390         11
2024-12-01      432         12
[120 rows x 2 columns]
```

В итоге был получен набор данных из 120 строк и трех колонок:

- индексная колонка – Month (год и первое число каждого месяца);
- данные о среднемесячном пассажиропотоке – Passengers;
- колонка с экзогенными данными - month_index (номер месяца).

Затем сформирован график с экзогенными данными (объект fig) и этот график трансформирован в формат html (объект plot_exogen).

Поскольку мы уже неоднократно сталкивались с данными о пассажиропотоках в предыдущих разделах, то здесь не будем делать разложение временного ряда и анализировать его составляющие, а перейдем сразу к поиску оптимальных параметров модели SARIMAX, для чего воспользуемся методом auto_arima библиотеки pmdarima:

```
param = pm.auto_arima(df[['Passengers']], exogenous=df[['month_index']],
                      start_p=1, start_q=1,
                      test='adf',
                      max_p=3, max_q=3, m=12,
                      start_P=0, seasonal=True,
                      d=1, D=1,
                      trace=False,
                      error_action='ignore',
                      suppress_warnings=True,
                      stepwise=True)
```

Обратите внимание, что при использовании данного метода для модели SARIMAX методу auto_arima нужно передать дополнительный параметр с экзогенными данными – exogenous. В нашем примере экзогенными данными является колонка с номером месяца набора данных df.

После того, как для модели SARIMAX найдены оптимальные параметры, они присвоены соответствующим переменным:

```
get_param = param.get_params()
param = get_param.get('order')
p, d, q = param[0], param[1], param[2]
s_param = get_param.get('seasonal_order')
P, D, Q, S = s_param[0], s_param[1], s_param[2], s_param[3]
```

Теперь, зная эти параметры, можно создать модель и провести ее обучение либо на полном наборе исторических данных, либо на части этого набора. В данном примере мы будем использовать традиционный подход машинного обучения:

- сделаем разбивку исходного набора данных на обучающую и тестовую выборку;
- обучим модель SARIMAX на обучающей выборке;
- сделаем прогноз изменения пассажиропотока на период, соответствующий тестовой выборке;

- вычислим ошибку прогноза:
- сравним соответствие прогноза тестовым данным в графическом виде.

Полученный временной ряд разбиваем на две части: на обучающую выборку - первые 9 лет (объект train), и на тестовую выборку – последний 1 год (объект test):

```
train = df[:'2023-12']
test = df['2024-01':]
```

Далее создаем модель SARIMAX (объект model) и запускаем процесс ее обучения (results - объект с обученной моделью):

```
model = SARIMAX(train['Passengers'], exog=train['month_index'],
                order=(p, d, q),
                seasonal_order=(P, D, Q, S))
results = model.fit()
```

Обратите внимание, что при формировании модели ей, кроме параметров, были переданы два набора данных:

- данные о пассажиропотоках из обучающей выборки - train['Passengers'];
- экзогенные данные из обучающей выборки - exog=train['month_index'].

Теперь можно проверить качество обучения модели и корректность полученных параметров. Для этого необходимо сделать прогноз изменения временного ряда на тот период, для которого имеются реальные данные в тестовой выборке. Формируем даты начала и конца периода, который соответствует периоду тестовой выборке, и делаем прогноз (объект predict):

```
start_date = '2024-01-01'
end_date = '2024-12-01'
predict = results.predict(start_date, end_date, exog=test['month_index'])
df['predict'] = predict
```

Здесь результаты прогноза добавлены в качестве новой колонки «predict» к исходному набору данных. Обратите внимание, что при формировании прогноза с моделью SARIMAX требуется обязательное подключение экзогенных данных, в нашем случае это колонка «month_index» из тестовой выборки данных.

Далее выполнен расчет ошибки прогноза. Для этого сформирован новый набор данных (объект df1), состоящий из реальных данных и данных прогноза (в интервале дат тестового периода). Для наглядности этот набор данных выведен в окно терминала:

```
            Passengers  month_index     predict
Month
2024-01-01         417            1  424.454518
2024-02-01         391            2  407.240171
2024-03-01         419            3  471.066076
2024-04-01         461            4  461.104654
2024-05-01         472            5  485.096105
2024-06-01         535            6  537.098000
2024-07-01         622            7  613.097580
2024-08-01         606            8  624.097673
2024-09-01         508            9  528.097652
2024-10-01         461           10  472.097657
```

```
2024-11-01      390         11  427.097656
2024-12-01      432         12  470.097656
```

Здесь мы имеем 12 строк с реальными данными (колонка Passengers), и с данными прогноза, полученного с использованием модели SARIMAX (колонка predict). На основе этих данных далее делается расчет ошибки прогноза (объект mape):

```
actual = df1['Passengers']
predicted = df1['predict']
mape = np.mean(np.abs((actual - predicted) / actual)) * 100
mape = round(mape, 2)
```

После этого сформирован график (объект fig) на котором показаны реальные данные и данные прогноза на модели SARIMAX.

График, построенный в plotly, приведен к формату, в котором он может быть отображен в шаблоне Django (объект plot_div). Полученные графики, заголовок страницы, параметры модели и рассчитанная ошибка прогноза через контекстную переменную context передается в шаблон analiz_sarimax.html (листинг 8.53).

Листинг 8.53. Шаблон analiz_sarimax.html

Примечание.
В демо-версии книги листинги программ удалены.

Это достаточно простой шаблон с контейнером, в котором имеется три блока: для вывода рассчитанных параметров и полученных графиков.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 8» и активировать ссылку «Анализ параметров модели SARIMAX».

Если выполнить набор вышеприведенных модулей, то в шаблон Django будет выведен график с экзогенными данными (рис. 8.33).

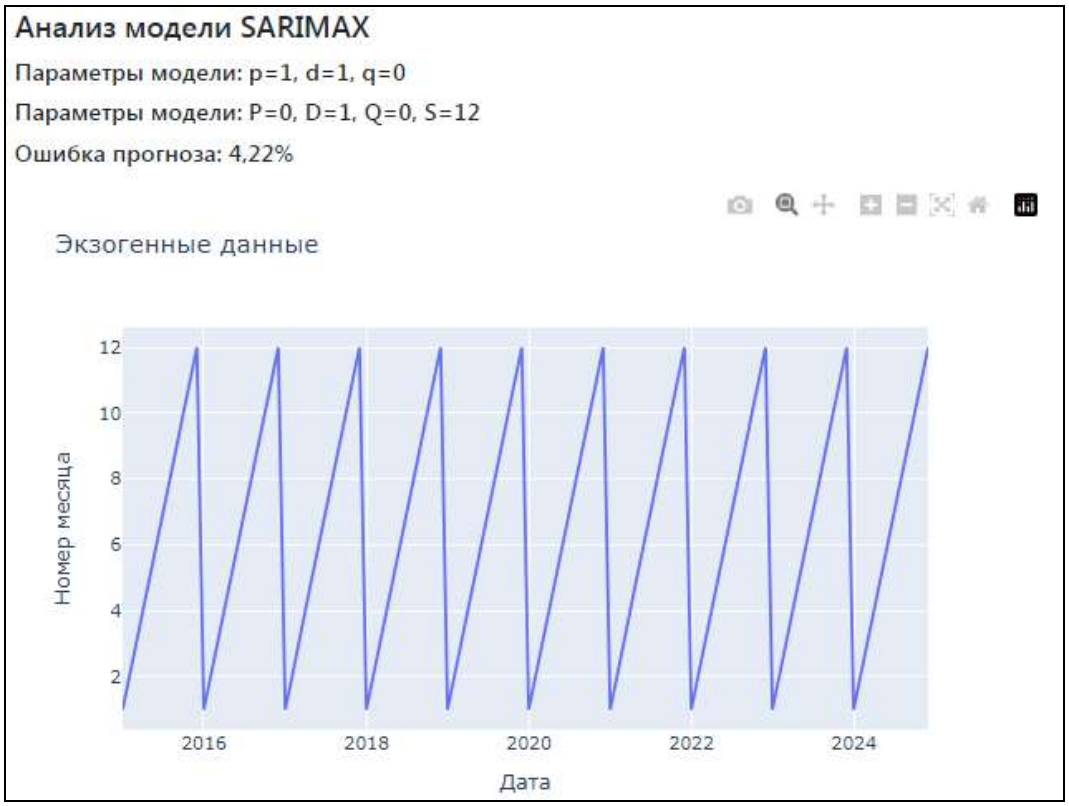


Рис. 8.33. Параметры модели SARIMAX и график с экзогенными данными временного ряда

Как видно из данного рисунка, экзогенные данные имеют стабильную цикличность с абсолютно одинаковыми верхней и нижней границами. Такая устойчивая цикличность обеспечит более тонкую настройку модели SARIMAX и более гибкую подгонку модели под реальные данные. Об этом свидетельствует и значение ошибки прогноза – всего 4,22%, то есть вероятность правильного прогноза составит более 95%.

График с историческими данными и данными прогноза на модели SARMAX представлены на рис. 8.34.

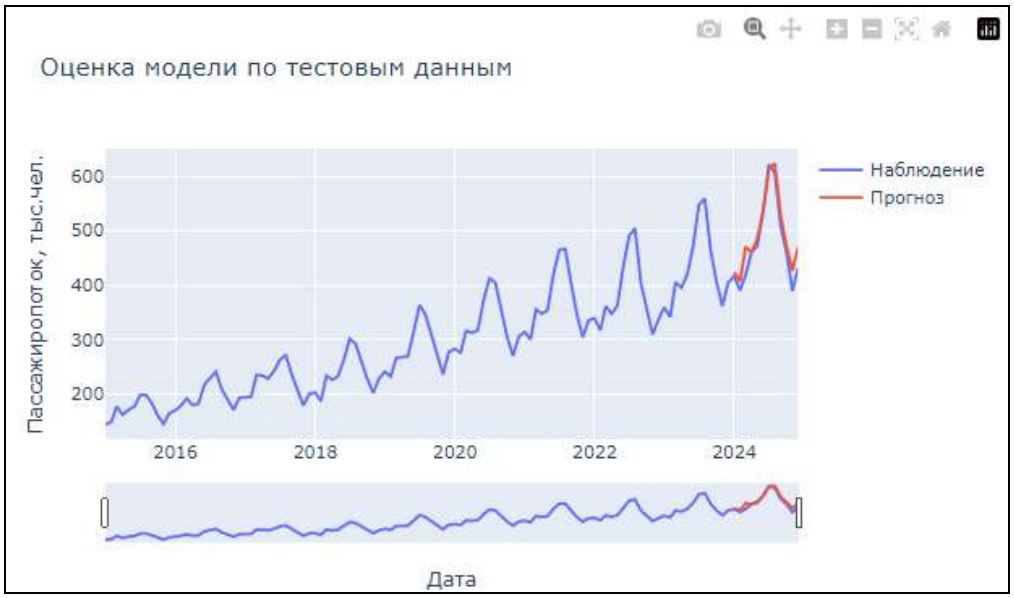


Рис. 8.34. Исторические данные и прогноз пассажиропотока по тестовым данным на модели SARIMAX

Как видно из данного рисунка, прогнозируемые данные вполне адекватно описывают реальную статистику. На основании этого можно сделать вывод, что модель SARIMAX с найденными параметрами может использоваться для прогнозирования данных на будущие периоды.

В нижней части графика имеется слайдер, с помощью которого можно увеличить интервал дат и более детально показать тот или иной фрагмент графика (рис. 8.35).

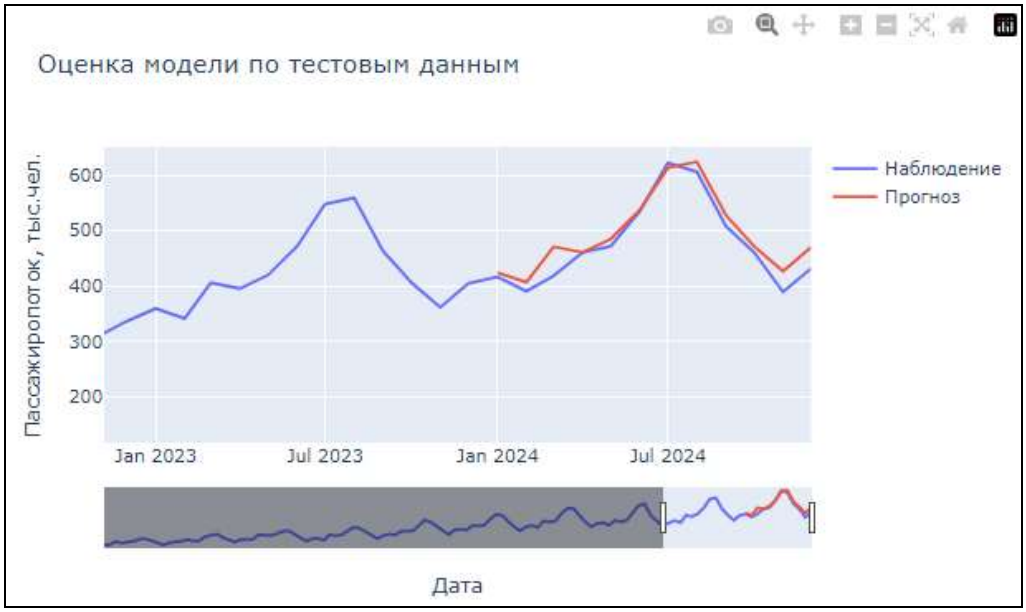


Рис. 8.35. Использование слайдера для масштабирования интервала дат

8.6.2. Прогнозирование временного ряда с моделью SARIMAX

С использованием модели SARIMAX можно делать два вида прогноза – статический и динамический:

- **статический прогноз** предполагает использование всего известного набора данных для прогнозирования каждого следующего значения;
- **динамический прогноз** начинает применять самые свежие прогнозные значения для предсказания будущих значений, после определенной пользователем даты старта.

Динамический прогноз обычно предоставляет более реалистичное представление о будущем поведении временного ряда, поскольку он учитывает оценки в процессе прогнозирования.

Реализуем программный код, который обеспечивает прогнозирование временного ряда с данными о пассажиропотоках с использованием статического прогноза. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 8.54).

Листинг 8.54. Ссылка активации маршрута prg_sarimax

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем prg_sarimxa (листинг 8.55).

Листинг 8.55 Маршрут URL с именем prg_sarimax

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление prg_sarimax (листинг 8.56).

Листинг 8.56. Представление (view) prg_sarimax

Примечание.
В демо-версии книги листинги программ удалены.

В этом программном модуле выполнен импорт необходимых библиотек, задан путь к CSV файлу «passengers.csv» с данными о пассажиропотоках (объект path_csv). Затем данные из этого файла загружены в объект df и для полученного набора данных сформирована индексная колонка:

```
df = pd.read_csv(path_csv)
df.set_index('Month', inplace=True)
df.index = pd.to_datetime(df.index)
```

После этого к полученному набору данных добавлена колонка «month_index» с экзогенной переменной:

```
df['month_index'] = df.index.month
```

Поскольку в программном коде предыдущего раздела были найдены оптимальные параметры для модели SARIMAX, мы можем их присвоить соответствующим переменным:

```
p, d, q = 1, 1, 0
P, D, Q, S = 0, 1, 0, 12
```

Теперь, зная эти параметры, можно создать модель и провести ее обучение на полном наборе исторических данных:

```
model = SARIMAX(df['Passengers'], exogenous=df['month_index'],
                order=(p, d, q),
                seasonal_order=(P, D, Q, S))
results = model.fit()
```

Обратите внимание, что при обучении модели SARIMAX ей обязательно нужно передать набор экзогенных параметров - exogenous=df['month_index'].

Далее определена глубина прогноза (2 года) созданы индексы для дат будущего периода:

```
forecast_steps = 24
predict_index = pd.date_range(start=df['Passengers'].index[-1],
                              periods=forecast_steps + 1, freq='M')[1:]
```

На основе полученных индексов сформированы наборы данных будущего периода (объекты df_ind и df_1).

```
df_ind = pd.DataFrame({'Month': predict_index})
df_1 = pd.DataFrame({'Month': predict_index})
```

К набору данных df_ind добавлена колонка с экзогенными данными:

```
df_ind['month_index'] = df_ind.index.month
```

После этого набор данных df_ind полностью готов для того, чтобы сделать прогноз изменения пассажиропотока на следующие два года:

```
forecast = results.forecast(steps=forecast_steps,
                            exog=df_ind['month_index'],
                            dynamic=False)
```

Обратите внимание, что при формировании прогноза с использованием модели SARIMAX методу forecast кроме количества шагов будущего периода в 12 месяцев (объект forecast_steps), нужно передать и экзогенные данные (df_ind['month_index']). Результаты прогноза в виде набора данных типа Series сохранены в объекте forecast.

> ***Примечание.***
> Здесь для получения прогноза использовался метод forecast(). Данный метод используется для получения вне выборочных прогнозов на основе последнего известного значения. Этот метод возвращает только предсказанные значения.

На следующем шаге результаты прогноза из набора данных типа Series трансформируются в набор данных DataFrame (объект df_predict):

```
df_predict = forecast.to_frame().reset_index()
df_predict.set_index(keys='index', inplace=True)
df_predict.index.name = 'Month'
```

Здесь в наборе данных df_predict создана индексная колонка с датами будущего периода, которой задано имя «Month». Теперь можно сформировать график с отображением исторических данных и данных прогноза на будущий период (объект fig).

На следующем шаге выполнен прогноз с использованием метода get_prediction (объект pred) для получения интервала прогнозирования:

```
pred = results.get_prediction(start=pd.to_datetime('2025-01-31'),
                              end=pd.to_datetime('2026-12-31'),
                              exog=df_ind['month_index'],
                              dynamic=False)
pred_ci = pred.conf_int()
```

> ***Примечание.***
> Метод get_prediction() обычно используется для внутри выборочных предсказаний. Он возвращает как предсказанные значения, так и граничные значения интервала предсказания. Интервал прогнозирования охватывает диапазон возможных значений, которые могут быть получены при применении модели к новым данным.

Обратите внимание, что при формировании прогноза с моделью SARIMAX методу get_prediction кроме дат начала и конца периода, нужно передать и экзогенные данные (df_ind['month_index']). Результаты прогноза сохранены в объекте forecast, а диапазон возможных значений сохранен в объекте pred_ci.

Далее с использованием библиотеки matplotlib построен график, на котором показаны исторические данные, экзогенные данные, прогноз и диапазон возможных значений прогноза. Этот график приведен к формату, в котором он может быть отображен в шаблоне Django (объект fig_prg).

График, построенный в plotly, приведен к формату, в котором он может быть отображен в шаблоне Django (объект plot_div). Полученные графики и заголовок страницы через контекстную переменную context передается в шаблон prg_sarimax.html (листинг 8.57).

Листинг 8.57. Шаблон prg_sarimax.html

> ***Примечание.***
> В демо-версии книги листинги программ удалены.

Это достаточно простой шаблон с контейнером, в котором имеется три блока: для вывода рассчитанных параметров и полученных графиков.

> ***Примечание.***
> В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 8» и активировать ссылку «Прогноз на модели SARIMAX».

Если выполнить набор вышеприведенных модулей, то в шаблон Django будет выведен график с историческими данными и прогнозом на будущий период (рис. 8.36).

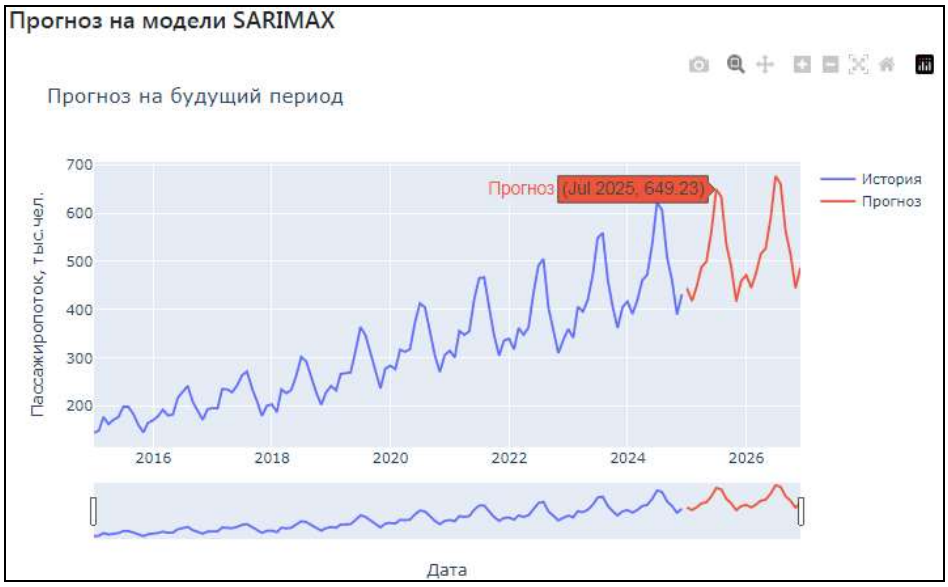


Рис. 8.36. Исторические данные и прогноз пассажиропотока на будущий период на модели SARIMAX

Как видно из данного рисунка, прогнозируемые данные вполне адекватно описывают характер динамики реального временного ряда на периоде в следующие 2 года. На основании этого можно сделать вывод, что модель SARIMAX с найденными параметрами хорошо подходит для прогнозирования данных этого временного ряда. Этот график является интерактивным, если курсор мыши навести на линию графика, то будет показано прогнозируемое значение

В нижней части графика имеется слайдер, с помощью которого можно увеличить интервал дат и более детально показать тот или иной фрагмент графика (рис. 8.37).



Рис. 8.7. Использование слайдера для масштабирования интервала дат

На следующем графике показан прогноз с интервалом прогнозирования, который описывает возможный разброс прогноза от реальных данных будущего периода (рис. 8.38).

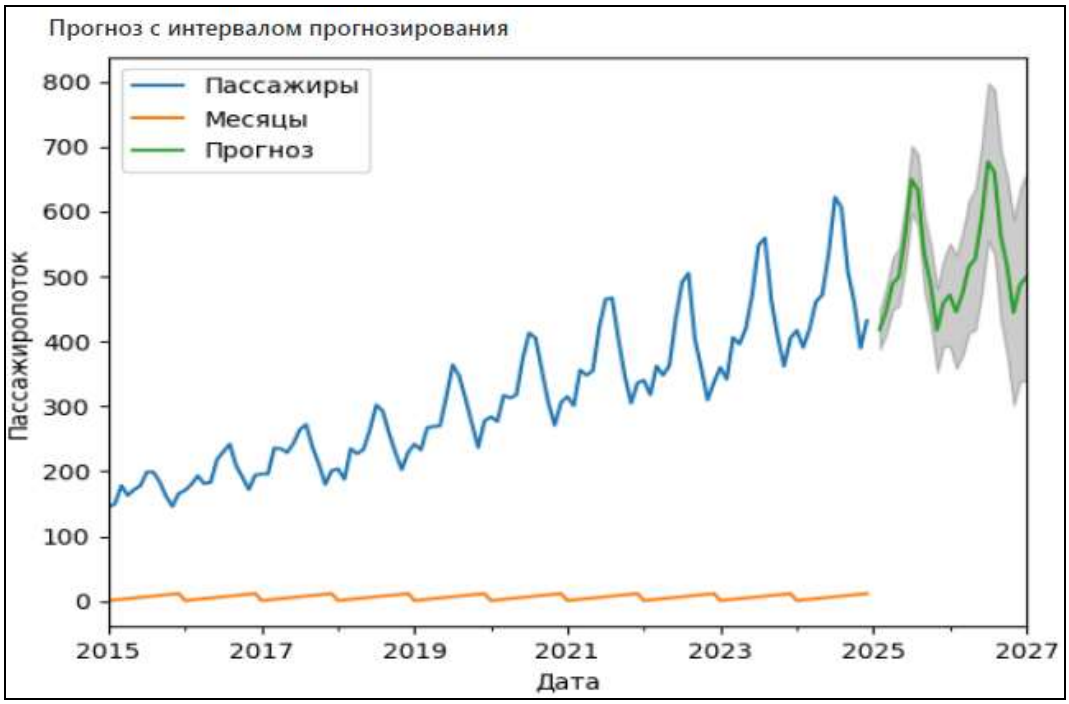


Рис. 8.38. Исторические данные и прогноз пассажиропотока на будущий период на модели SARIMAX с интервалом прогнозирования

На данном графике показаны следующие элементы:

- исторические данные исходного временного ряда (синяя линия);
- экзогенные данные, которые использовались в модели SARIMAX (желтая линия);
- прогноз изменения пассажиропотока на ближайшие два года (зеленая линия);
- интервал прогнозирования (серая область).

Из данного рисунка видно, что с увеличением глубины прогноза, диапазон возможных ошибок прогноза расширяется.

8.7. Краткие итоги

Итак, мы познакомились с некоторыми статистическими моделями, которые позволяют выполнить анализ временных рядов, а главное – спрогнозировать изменение данных временных рядов на будущие периоды. Использование этих моделей требует достаточно высокой квалификации аналитика или программиста, который разрабатывает программное обеспечение для аналитиков данных. Неправильно подобранная модель под характерные особенности конкретного временного ряда, или не оптимальные параметры модели могут привести к ошибочному прогнозу.

Однако усилиями компании Facebook была разработана библиотека Prophet ("Пророк"), которая в значительной степени автоматизирует процессы анализа временных рядов и решения задач прогнозирования. Это позволяет даже начинающим аналитикам обрабатывать временные ряды и получать достоверные результаты с минимальной вероятностью получения ошибочных выводов. Использованию библиотека Prophet для решения задач прогнозирования посвящена следующая глава.

Глава 9. Библиотека Prophet для прогнозирования данных

Прогнозирование временных рядов — это достаточно популярная аналитическая задача. Прогнозы используются, например, для понимания, сколько будет стоить тот или иной сырьевой ресурс через год, каков будет спрос на товары в гипермаркете, или как будут меняться котировки акций различных компаний.

Существует большое количество различных подходов для прогнозирования временных рядов, такие статистические модели как AR, MA, ARIMA, SARIMA, SAROMAX, нейронные сети и т.д. В этой главе мы познакомимся с библиотекой для прогнозирования временных рядов Prophet от компании Facebook. Библиотека Prophet (в переводе с английского, "Пророк"), выпущена в феврале 2017 года и является достаточно популярным инструментом среди разработчиков и аналитиков данных. Эта библиотека предназначена для прогнозирования различных финансово экономических тенденций и строит достаточно хорошие прогнозы. Кроме того, библиотека дает возможность, изменяя человеко-понятные параметры, улучшать прогноз и не требует от аналитиков глубоких знаний структуры используемых моделей.

Библиотека Prophet для прогнозирования временных рядов хороша тем, что позволяет строить точные и интерпретируемые прогнозы даже в условиях ограниченного объема данных, сезонных колебаний и нестабильных трендов. Вот только некоторые достоинства данной библиотеки:

53. **Простота использования**. У данной библиотеки интуитивно понятный программный интерфейс, что делает ее доступной для широкого круга пользователей, даже не имеющих глубоких знаний структуры и особенностей временных рядов.
54. **Устойчивость к отсутствию данных.** Эта библиотека автоматически обрабатывает отсутствующие данные и аномалии, что упрощает работу с реальными временными данными.
55. **Учет сезонности и праздников**. Алгоритмы, заложенные в основу библиотеки, позволяют учитывать сезонные и праздничные отклонения, что важно для прогнозирования временных рядов с сезонными колебаниями.
56. **Гибкость в настройке**. Пользователь может настраивать модель на основе своих потребностей, включая изменение сезонности, добавление праздников и определение структуры тренда.
57. **Простота формирования графиков**. Данная библиотека интегрирована с известной библиотекой Plotly, так что сформировать нужные графики можно буквально одной - двумя строчками программного кода.

Из материалов этой главы вы узнаете принципы работы библиотеки Prophet, как можно сохранить и многократно использовать обученную модель. Кроме того, вы получите практические навыки загрузки данных из открытых источников в режиме on-line и прогнозирования следующих финансово экономических показателей:

- мировые цены на нефть (по данным API Yahoo Finance);
- котировки акций известных компаний (по данным API Yahoo Finance);

- курсы валют (по данным API Центрального банка России);
- цены на драгоценные металлы (по данным API Центрального банка России);
- экономические показатели стран (по данным API Всемирного банка);
- продажи товаров торгового центра.

9.1. Принципиальные основы библиотеки Prophet

Давайте рассмотрим, на чем же основан принцип работы библиотеки Prophet. По сути, это аддитивная регрессионная модель, состоящая из следующих компонент:

$y(t) = g(t) + s(t) + h(t) + Et$

Рассмотрим подробнее эти составляющие.

Тренд g(t). Это кусочно-линейная или логистическая функция. С линейной функцией все понятно – это функция линейной зависимости одного значения от другого. Логистическая же функция позволяет моделировать рост параметра с насыщением, когда при увеличении показателя снижается темп его роста. Типичный пример — это темп рост активных пользователей сайта после его появления с последующим снижением и насыщением. Кроме всего прочего, библиотека умеет по историческим данным выбирать оптимальные точки изменения тренда, но их также можно задать и вручную.

Сезонность s(t). Сезонные компоненты отвечают за моделирование периодических изменений, связанных с недельной или годовой сезонностью. Недельная сезонность моделируется с помощью фиктивных переменных. Для недельной сезонности добавляются 6 дополнительных признаков (monday, tuesday, wednesday, thursday, friday, saturday), которые принимают значения 0 и 1 в зависимости от даты. Признак sunday, соответствующий седьмому дню недели, не добавляют, потому что он будут линейно зависеть от других дней недели, и это будет влиять на модель. Годовая же сезонность моделируется рядами Фурье.

Аномалии h(t). Компонента h(t) отвечает за заданные пользователем аномальные дни, в том числе и нерегулярные, такие как, например, Black Fridays (черные пятницы).

Ошибка Et. Этот элемент содержит информацию, которая не учтена моделью.

Алгоритмы, используемые в Prophet, обеспечивают достаточно достоверные прогнозы и имеют более низкие показатели ошибок по сравнению с другими методами прогнозирования временных рядов. Эта библиотека достаточно хорошо интегрирована с библиотекой Plotly, что упрощает формирование интерактивных графиков, особенно в веб приложениях.

Для установки библиотеки Prophet можно использовать следующую инструкцию:

```
pip install prophet
```

9.1.1. Прогнозирование цен на нефть по данным API Yahoo Finance

Рассмотрим возможности данной библиотеки на примере оценки прогноза стоимости нефти на основе данных, полученных от Yahoo Finance. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 9.1).

Листинг 9.1. Ссылка активации маршрута prog_noz_oil

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем prog_noz_oil (листинг 9.2).

Листинг 9.2. Маршрут URL с именем prog_noz_oil

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление prog_noz_oil (листинг 9.3).

Листинг 9.3. Представление (view) prog_noz_oil

Примечание.
В демо-версии книги листинги программ удалены.

В этом модуле выполнен импорт необходимых библиотек, задан тикер котировок цен на нефть (объект tick) и из API Yahoo Finance загружены данные о ценах на нефть за указанный период (объект df):

```
tick = 'BZ=F'
ticker = yf.Ticker(tick)
df = ticker.history(start="2022-01-01", end="2025-05-21", interval="1d")
```

Из полученного набора данных выбрана только одна колонка (Close) с ценой на момент закрытия торгов:

```
df = df[['Close']]
```

В учебных целях (чтобы ознакомиться со структурой полученных данных) это набор данных выведен в окно терминала:

```
                           Close
Date
2022-01-03 00:00:00-05:00  78.980003
2022-01-04 00:00:00-05:00  80.000000
2022-01-05 00:00:00-05:00  80.800003
2022-01-06 00:00:00-05:00  81.989998
2022-01-07 00:00:00-05:00  81.750000
```

```
...                        ...
2025-05-14 00:00:00-04:00  66.089996
2025-05-15 00:00:00-04:00  64.529999
2025-05-16 00:00:00-04:00  65.410004
2025-05-19 00:00:00-04:00  65.540001
2025-05-20 00:00:00-04:00  65.379997
[850 rows x 1 columns]
```

Как видно из полученных результатов, было загружено 850 строк ежедневных данных, значения для выходных дней пропущены (торги в выходные дни не проводятся), колонка с датой имеет временную составляющую.

На основе исходного набора данных сформирован график (объект fig), который приведен к формату для отображения в шаблоне Django (объект hist_html).

Если бы мы для прогноза использовали статистические модели типа AR, MA, ARMA, ARIMA или SARIMA, то нам нужно было сначала выполнить очистку и анализ исходного временного ряда, определиться с типом модели и найти для нее оптимальные параметры. Однако при работе с библиотекой Prophet ничего этого делать не нужно, так как всю эту работу выполнят алгоритмы, «зашитые» в модули данной библиотеки. Единственное что нужно сделать, это структурировать данные под требования данной библиотеки. А эти требования очень просты.

В наборе данных должны присутствовать только две колонки: колонка с датами временного ряда, колонка со значениями временного ряда. Колонка с датами обязательно должна иметь имя «ds», колонка с данными обязательно должна иметь имя «y». Такое изменение исходного набора данных выполнено в следующих инструкциях:

```
df = df.reset_index()
df.columns = ['ds', 'y']
df['ds'] = pd.to_datetime(df['ds']).dt.date
```

Здесь сначала в наборе данных df был сброшен индекс, затем колонкам присвоены новые имена (ds, y), и в колонки ds из даты убрана составляющая времени (оставлена только дата).

В учебных целях (чтобы ознакомиться со структурой полученных данных) это набор данных выведен в окно терминала:

```
           ds         y
0    2022-01-03  78.980003
1    2022-01-04  80.000000
2    2022-01-05  80.800003
3    2022-01-06  81.989998
4    2022-01-07  81.750000
..          ...        ...
845  2025-05-14  66.089996
846  2025-05-15  64.529999
847  2025-05-16  65.410004
848  2025-05-19  65.540001
849  2025-05-20  65.379997
[850 rows x 2 columns]
```

Как видно из полученных результатов, был загружен набор данных, состоящий из 850 строк и двух колонок (ds, y). В таком виде данные полностью готовы к тому, что бы передать их в модель для прогнозирования.

На следующем шаге была создана и обучена модель:

```
model = Prophet(yearly_seasonality=True)
model.fit(df)
```
Обученная модель готова для решения задач прогнозирования, это можно сделать буквально в две строки программного кода:

```
predict = 30
future = model.make_future_dataframe(periods=predict, freq='D')
forecast = model.predict(future)
```
Результаты прогноза сохраняются в объекте forecast.

Еще проще формируется график результатов прогноза (объект fig):

```
fig = plot_plotly(model, forecast)
```
То есть для формирования графика в модуль plot_plotly библиотеки Prophet нужно передать модель (объект model) и результаты прогноза (объект forecast). Ну а сформировать собственные подписи к осям и заголовок графика можно традиционным методом update_layout библиотеки Plotly:

```
fig.update_layout(title='Динамика изменения цены на нефть',
                  xaxis_title='Дата',
                  yaxis_title='Цена, $')
```
Полученный график приведен к формату, в котором он может быть показан в шаблоне Django (объект prog_html).

Так же просто можно получить графики компонент исходного временного ряда:

```
fig = plot_components_plotly(model, forecast)
comp_html = fig.to_html(full_html=False)
```
Обратите внимание, насколько программный код проще и понятней при работе с библиотекой Prophet.

После этого создана строка с заголовком страницы (title_str), полученные графики с заголовком страницы через контекстную переменную context передаются в шаблон prog_noz_oil.html (листинг 9.4).

Листинг 9.4. Шаблон prog_noz_oil.html

Примечание.
В демо-версии книги листинги программ удалены.

Это достаточно простой шаблон с контейнером, в котором имеется блок для вывода заголовка страницы, и три блока для вывода графиков. Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 9» и активировать ссылку «Прогноз цен на нефть».

Если выполнить набор вышеприведенных модулей, то получим график с исходным набором данных (рис. 9.1).



Рис. 9.1. Временной ряд с динамикой изменения цен на нефть

Как видно из данного графика, до 2022 года цена на нефть росла, а после 2022 года с некоторыми колебаниями видна устойчивая тенденция снижения цены. Полученный график является интерактивным. При наведении курсора мыши на линию появляется окно со значением цены на выбранную дату, в правом верхнем углу имеется кнопочное меню для изменения параметров графика. Так как график выводится в адаптивный контейнер Bootstrap, то он будем менять свои размеры при изменении размеров окна браузера.

График с прогнозом цены на нефть показан на рис. 9.2.

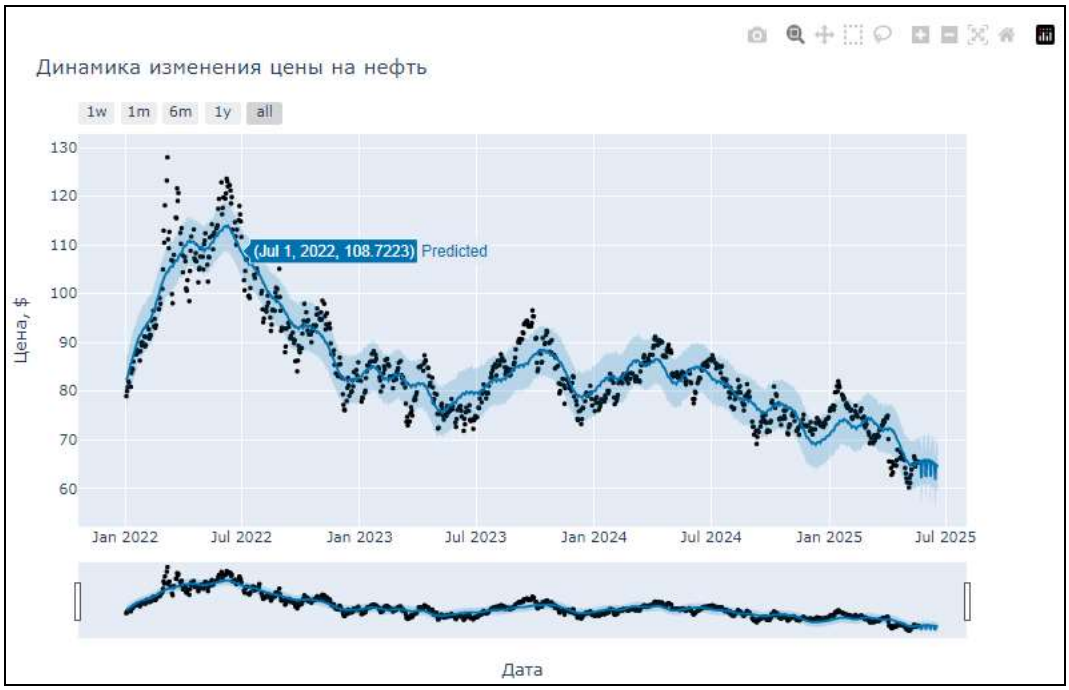


Рис. 9.2. Временной ряд с динамикой изменения цен на нефть и данными прогноза

На данном графике показаны следующие значения:

- черными точками показаны исторические данные;
- синей линией показан прогноз по историческим данным, и на будущий период;
- голубым фоном выделена область доверительного интервала, в пределах которого возможны ошибки прогноза.

Полученный график является интерактивным. При наведении курсора мыши на черную точку появляется окно с реальной ценой в данный день. При наведении курсора мыши на синюю линию появляется окно со значением прогнозируемой цены на выбранную дату. В верхней части графика имеется пять кнопок, которые позволяют менять разрешение шкалы дат (увеличивать или уменьшать интервал дат). В нижней части графика имеется слайдер, который позволяет, как менять разрешение шкалы дат, так и перемещать окно с увеличенным диапазоном дат по оси «x».

На рис. 9.3 показан этот же график, в котором расширен диапазон дат, и окно с расширенным диапазоном находится в районе прогноза будущего периода.

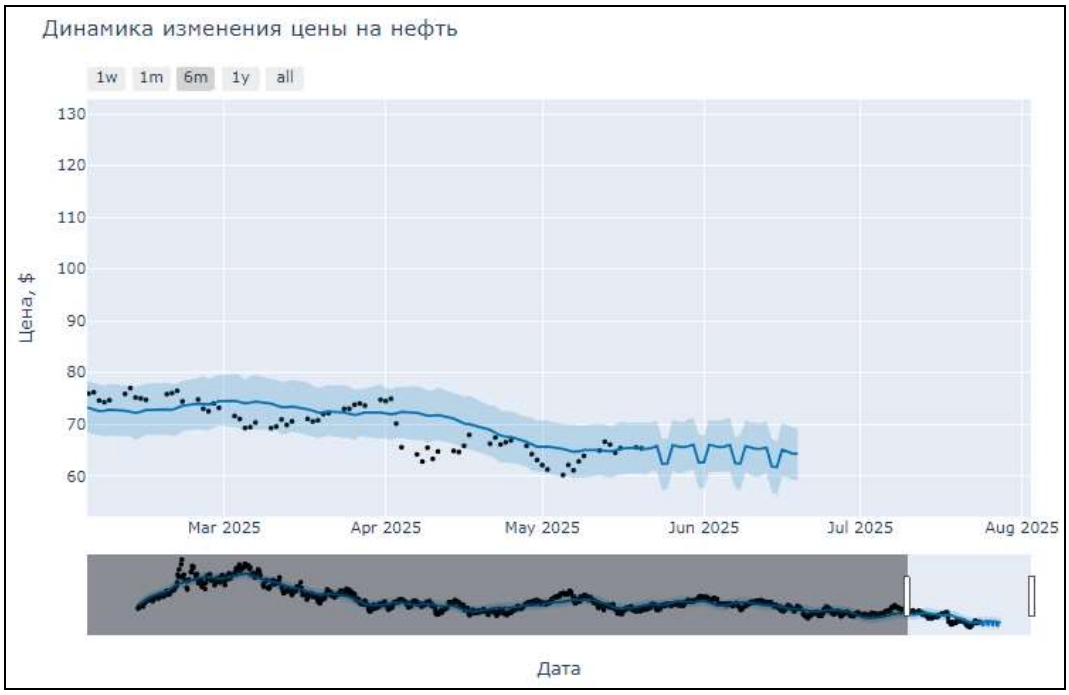


Рис. 9.3. Прогноз изменения цены на нефть в будущем периоде

На рис. 9.4 представлены компоненты исходного временного ряда

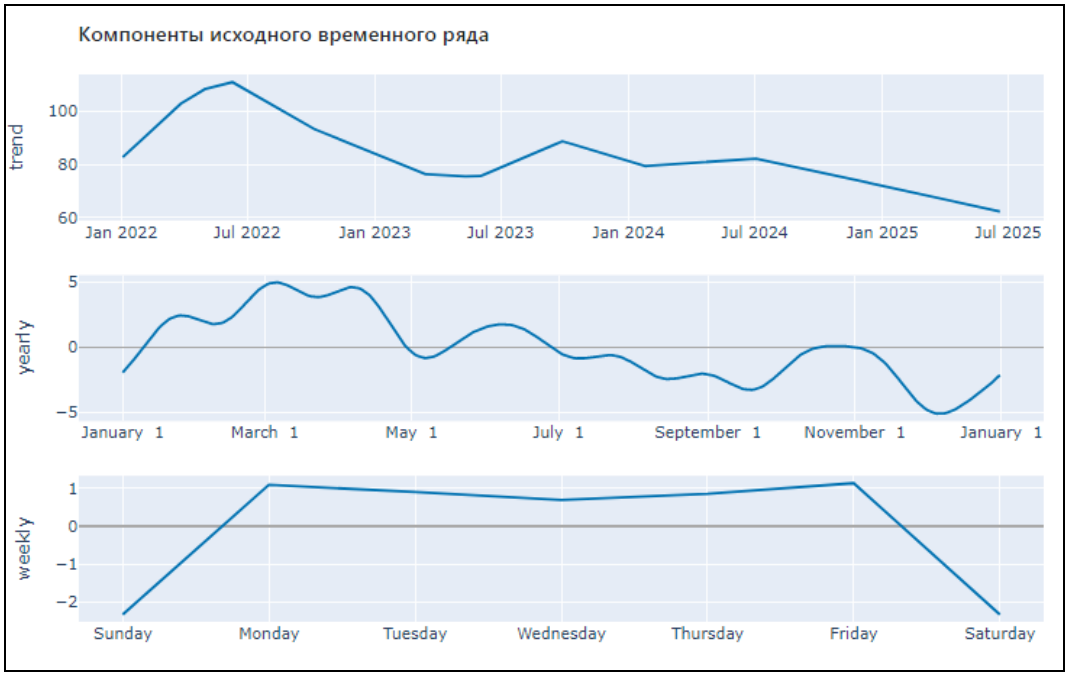


Рис. 9.4. Компоненты исходного временного ряда

Как видно из данного рисунка, тренд временного ряда показывает постепенно снижение цены на нефть. Годовая сезонность демонстрирует увеличение цены на нефть в весенние месяцы, и падение цен в осенние месяцы. Недельная сезонность говорит о том, что в течение недели колебание цены незначительные Небольшое уменьшение к среде, а в воскресные дни она резко падает (это связано с тем, что по выходным дням торги на бирже не проводятся и во временном ряде сведения о цене в эти дни отсутствуют).

Обратите внимание, что мы не делали анализа временного ряд, не подбирали модель и не определяли ее параметры, не разбивали временной ряд на компоненты, однако получили достаточно адекватный прогноз. Всю эту работу за нас выполнили модули библиотеки Prophet.

9.1.2. Сохранение обученной модели в Prophet

Библиотека позволяет сохранить обученную модель в файле, а затем использовать ее многократно без потерь времени на обучение. Рассмотрим эту возможность данной библиотеки на примере прогноза стоимости акций компании Apple на основе данных, полученных от Yahoo Finance. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 9.5).

Листинг 9.5. Ссылка активации маршрута seve_model

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем seve_model (листинг 9.6).

Листинг 9.6. Маршрут URL с именем seve_model

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление seve_model (листинг 9.7).

Листинг 9.7. Представление (view) seve_model

Примечание.
В демо-версии книги листинги программ удалены.

Код данного программного модуля практически идентичен коду аналогичного модуля предыдущего раздела, поэтому здесь остановимся только на добавленных инструкциях.

Здесь выполнен импорт дополнительных модулей библиотеки Prophet: для записи модели в файл - model_to_json, для чтения модели из файла - model_from_json.

Для реализации этого примера были использованы данные о ценах на акции компании Apple:

```
firma = ['AAPL']
```

```
df = yf.download(firma, start="2022-01-01", end="2025-06-01")
```
В полученный набор данных добавлены пропущенные данные выходных дней, когда торги на бирже не проводятся:

```
df = df["Close"].resample("1D").mean().fillna(method="ffill")
```
Параметры обученной модели сохранены во внешнем файле с именем «prophet_mod.json»:

```
family = 'mod'
with open(f'prophet_{family}.json', 'w') as f:
    f.write(model_to_json(model))
```
Обученная модель загружена в объект model из файла «prophet_mod.json»:

```
with open(f'prophet_{family}.json', 'r') as f:
    model = model_from_json(f.read())
```
После выполнения этого программного кода в базовом каталоге приложения появится новый файл с именем «prophet_mod.json».

> ***Примечание.***
> В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 9» и активировать ссылку «Сохранение моделей в Prophet».

Если выполнить набор вышеприведенных модулей, то получим график с исходным набором данных (рис. 9.5).



Рис. 9.5. Временной ряд с динамикой изменения цен на акции компании Apple

Как видно из данного графика, до января 2025 года цена на акции компании Apple росла с некоторыми колебаниями, а после января 2025 года видна устойчивая тенденция снижения цены.

График с прогнозом цены на акции компании Apple показан на рис. 9.6.

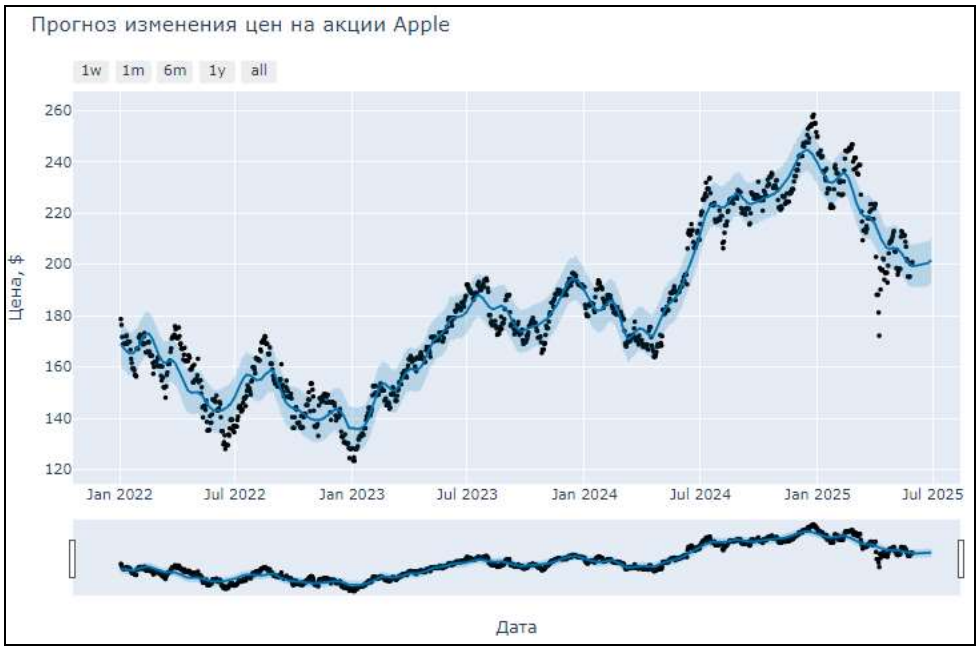


Рис. 9.6. Временной ряд с динамикой изменения цен на акции Apple и данными прогноза

Из данного графика видно, что в ближайшие 30 дней должно прекратиться падение цен на акции. Более детальную картину прогноза можно посмотреть, если воспользоваться слайдером, который позволяет менять разрешение шкалы дат (рис. 9.7).

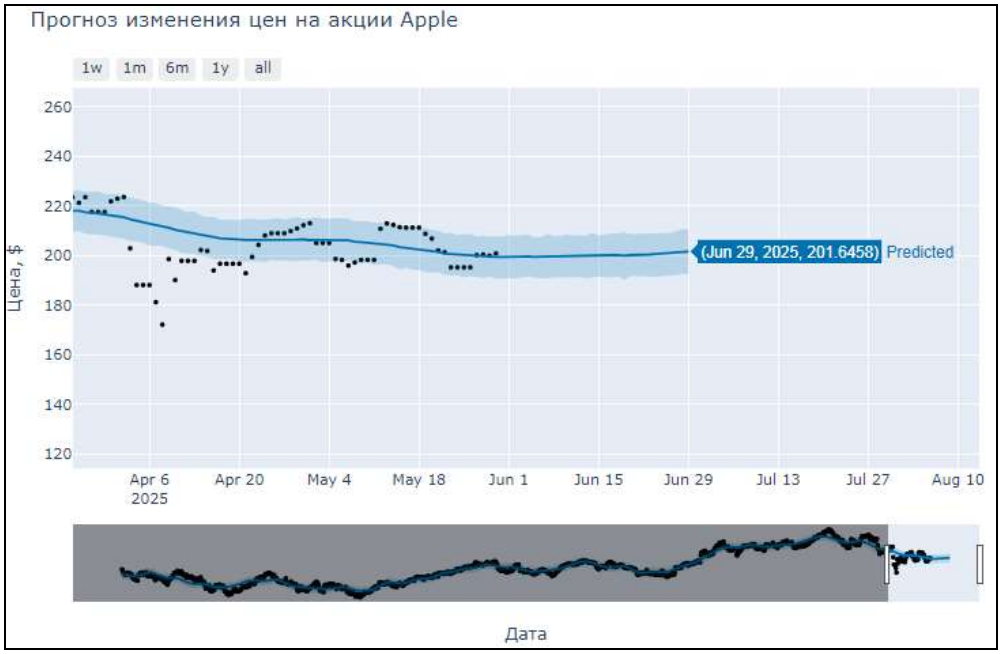


Рис. 9.7. Прогноз изменения цены на нефть в будущем периоде

Из данного графика видно, что действительно в ближайший период цена должна стабилизироваться на уровне 201$.

На рис. 9.8 представлены компоненты исходного временного ряда

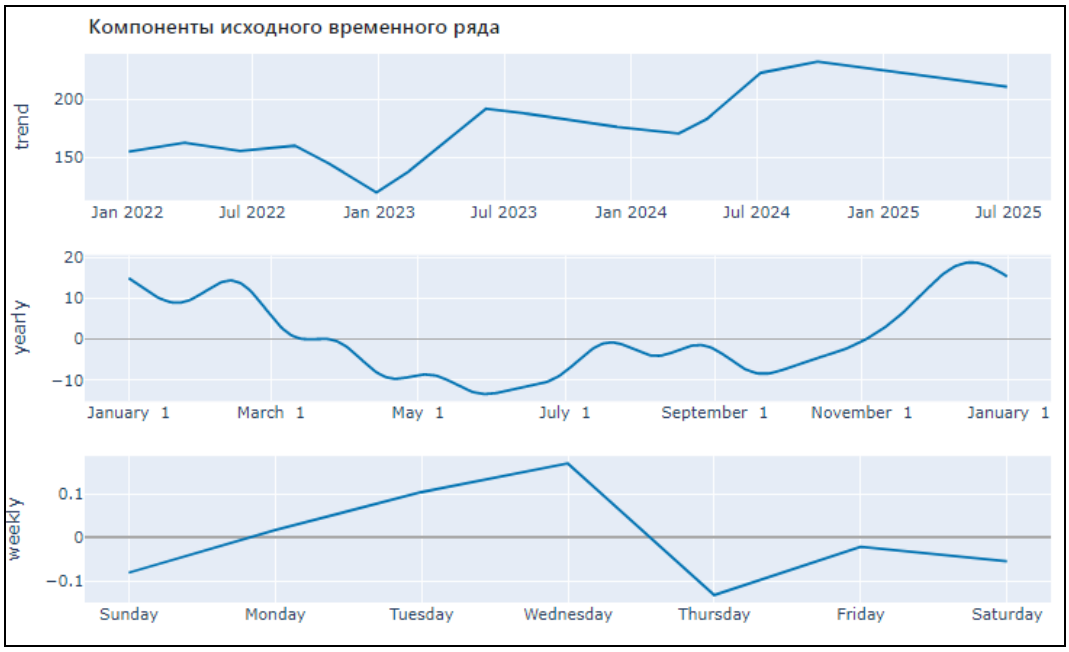


Рис. 9.8. Компоненты исходного временного ряда

Как видно из данного рисунка, тренд временного ряда показывает постепенно увеличение цен на акции, с некоторым снижением в последние месяцы. Годовая сезонность демонстрирует увеличение цены на акции в зимние месяцы, и падение цен в летние месяцы. Недельная сезонность говорит о том, что в течение недели цены на акции растут с понедельника до среды, а потом происходит их падение.

9.1.3. Прогнозирование котировок акций по данным API Yahoo Finance

Теперь реализуем программные модули, которые обеспечат загрузку данных о котировках акций из API Yahoo Finance, выбор параметров запроса, анализ этих данных и прогнозирование с библиотекой Prophet, а также вывод итогов прогнозирования в графическом виде. С использованием параметров запроса можно будет сделать выбор фирмы из списка, изменить интервал дат запрашиваемого периода и задать глубину прогнозирования.

Так как пользователю будет предложен выбор параметров, то в приложении необходимо создать форму. Для данного проекта в файле main/api_finance/forms.py написан код класса Stock_Form – форма для выбора фирмы, задания периода выборки данных и глубины прогнозирования (листинг 9.8).

Листинг 9.8. Программный код класса Stock_Form (форма для выбора фирмы, задания периода выборки данных и глубины прогнозирования)

Примечание.
В демо-версии книги листинги программ удалены.

В этой форме выполнен импорт необходимых библиотек. Затем объекту years_ago задана дата на пять лет ранее текущей даты, то есть по умолчанию из API будут запрошены данные за последние пять лет. В объект firms, представляющий вложенный список, загружены тикеры и названия фирм. Этот список используется для выбора фирмы, для которой будет выполнен запрос о котировках акций.

После этого создано четыре поля для выбора данных:

- date_1 – дата начала периода;
- date_2 - дата конца периода;
- kod_firms – коды фирм;
- deep_prog – глубина прогноза.

Для начала периода по умолчанию задана дата, на 5 лет ранее текущей даты, а для конца периода задана текущая дата. По умолчанию будет сформирован период, представляющий последние пять лет от текущей даты. Для выбора дат задан виджет, который представляет собой календарь. Для выбора фирм задан виджет, представляющий собой выпадающий список с множеством выбора. То есть пользователь сможет выбрать любое количество фирм в любом сочетании.

Примечание.
При выборе нескольких фирм будут загружены данные по всем фирмам, но прогноз выполнен только по одной из них.

Для выбора глубины прогноза задано поле для ввода целых чисел.

Для решения поставленной задачи в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 9.9).

Листинг 9.9. Ссылка активации маршрута yf_param_fig

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем yf_param_fig (листинг 9.10).

Листинг 9.10. Маршрут URL с именем yf_param_fig

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление yf_param_fig (листинг 9.11).

Листинг 9.11. Представление (view) yf_param_fig

Примечание.
В демо-версии книги листинги программ удалены.

В этом представлении выполнен импорт необходимых библиотек и при поступлении GET запроса сформированы следующие начальные параметры:

- d1 = date.today() - relativedelta(years=5) – дата начала периода;
- d2 = date.today() - дата конца периода;
- list_firm - список фирм (пустое значение);
- deep_pog = 30 – глубина прогноза (в данном случае это 30 дней);
- title_str – заголовок страницы;
- form = Stock_Form() – форма для выбора параметров запроса к данным.

Здесь же даты начала и конца периода приводятся к строковому типу европейского формата.

```
d1_str = d1.strftime("%d-%m-%Y")
d2_str = d2.strftime("%d-%m-%Y")
```

В таком более привычном виде они будут отображаться в шаблоне.

Затем все полученные параметры загружаются в контекстный словарь context, который передается в шаблон с именем yf_param_fig.html. Этот шаблон имеет следующий код (листинг 9.12).

Листинг 9.12. Шаблон yf_param_fig.html

Примечание.
В демо-версии книги листинги программ удалены.

Этот шаблон имеет более сложную структуру, чем все ранее рассматриваемые шаблоны. Он имеет два раздела и несколько блоков. Первый раздел содержит основную страницу, второй раздел представлен в виде боковой всплывающей панели.

Первый раздел имеет шесть блоков:

- блок для вывода заголовка страницы;
- блок с кнопкой для открытия боковой панели;
- блок для отображения параметров запроса;
- блок для вывода графика исторических данных;
- блок для вывода графика с прогнозом;
- блок для вывода структуры исторических данных.

На боковой всплывающей панели находится форма для выбора параметров запроса к API Yahoo Finance.

При первоначальной загрузке этот шаблон будет иметь следующий вид (рис. 9.9).

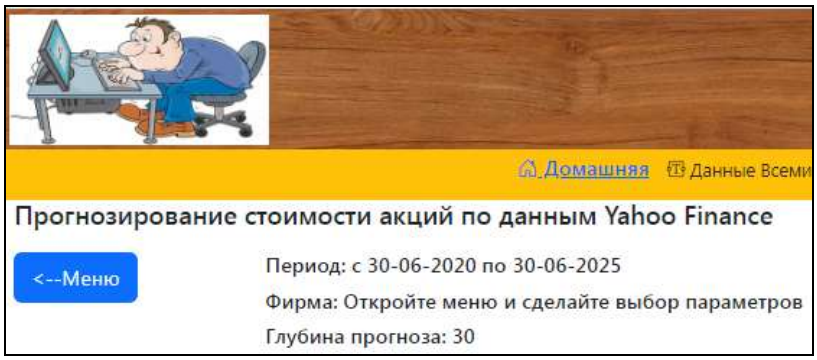


Рис. 9.9. Вид шаблона yf_param_fig.html при первоначальной загрузке.

В этом разделе страницы показан блок с кнопкой открытия боковой панели и параметрами запроса к данным, заданным по умолчанию:

- дата начала периода;
- дата конца периода,
- поясняющий текст, предлагающий нажать на кнопку «Меню» (для открытия боковой панели);
- глубина прогноза, заданная по умолчанию.

При нажатии на кнопку «Меню» с левой стороны страницы выдвигается боковая панель с формой для изменения параметров запроса (рис. 9.10).

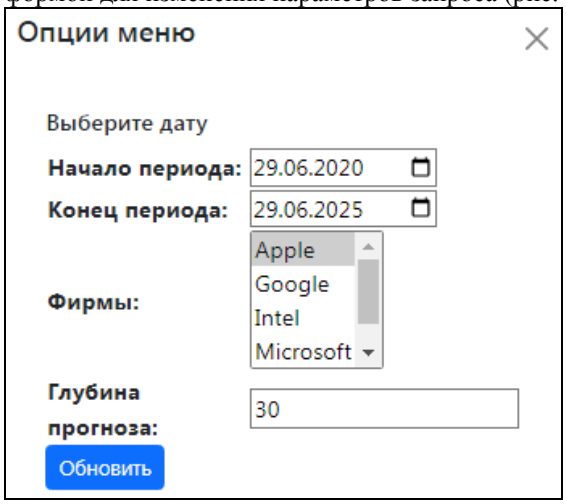


Рис. 9.10. Боковая панель с формой для выбора параметров запроса к данным

В этой форме пользователь может изменить дату начала и конца периода, из предложенного списка выбрать одну или несколько фирм, изменить значение глубины прогноза. Одна фирма выбирается простым нажатием левой кнопкой мыши на названии фирмы. Если требуется выбрать несколько фирм, то делается нажатие левой кнопкой мыши на название фирмы с удерживанием клавиши Ctrl.

После нажатия кнопки «Обновить» боковая панель закроется, а в блоках отображения графических данных появятся соответствующие графики.

Вернемся к программному коду представления и разберем программный код, который обрабатывает нажатие кнопки «Обновить».

После нажатия на кнопку «Обновить» в представлении yf_param_fig начинает работать программный код обработки POST запроса, следующий за следующей инструкцией:

```
if request.method == "POST":
```

Здесь, после проверки валидности формы, из нее загружаются следующие параметры:

- d1 – дата начала периода;
- d2 – дата окончания периода;
- list_firm – список выбранных фирм;
- deep_prog – глубина прогноза.

После получения данных из формы делается проверка значения глубины прогноза. Если пользователь ошибся и указал нулевое или отрицательное значение глубины прогноза, то ему присваивается минимально допустимое значение – 1.

Затем делается запрос к API платформы Yahoo Finance, полученные данные сохраняются в объекте df, из них выбирается только одна колонка – стоимость акций на момент закрытия торгов – Clise (изменяется объект df). Затем выполняются стандартные процедуры формирования графика (объект fig), который приводится к формату для отображения в шаблоне Django (объект plot_div).

На следующем шаге структура полученного набора данных приводится к виду, в котором его может принять библиотека Prophet:

```
df.reset_index(level=0, inplace=True)
df.columns.values[[0, 1]] = ['ds', 'y']
```

В этих инструкциях сначала удаляется индексная колонка, а оставшимся колонкам с данными присваиваются имена «ds» и «y». Здесь:

- ds – колонка с датами:
- y – колонка с данными (цена на акции).

Затем создается модель (объект model) и выполняется ее обучение:

```
model = Prophet(yearly_seasonality=True)
model.fit(df)
```

Теперь на основе обученной модели можно сделать прогноз на выбранную пользователем глубину:

```
future = model.make_future_dataframe(periods=deep_prog, freq='D')
forecast = model.predict(future)
```

После этого формируется график с результатами прогноза, который приводится к формату для отображения в шаблоне Django (объект prog_html).

Аналогичным образом формируется график анализа компонент временного ряда (тренда и сезонности), который то же приводится к формату для отображения в шаблоне Django (объект comp_html).

Затем все полученные объекты через контекстную переменную context передаются в шаблон yf_param_fig.html.

Теперь можно проверить работу данных модулей.

> ***Примечание.***
> В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 9» и активировать ссылку «Прогноз котировок акций от Yahoo Finance».

Если выполнить набор вышеприведенных модулей с параметрами по умолчанию, то получим график с исходным набором данных котировок акций компании Apple (рис. 9.11).



Рис. 9.11. Временной ряд с котировкой акций компании Apple

Как видно из данного графика, до 2025 года цена на акции росла, а после января 2025 года с некоторыми колебаниями видна устойчивая тенденция снижения цены.

График прогноза изменения цены на акции этой компании показан на рис. 9.12.



Рис. 9.12. Прогноз изменения цены на акции компании Apple

Как видно из данного графика, линия прогноза достаточно хорошо описывает исторические данные, а в ближайшие 30 дней ожидается повышение стоимости акций. Так как это график является интерактивным, можно с помощью слайдера укрупнить интервал дат и посмотреть на более детальную информацию прогноза (рис. 9.13).

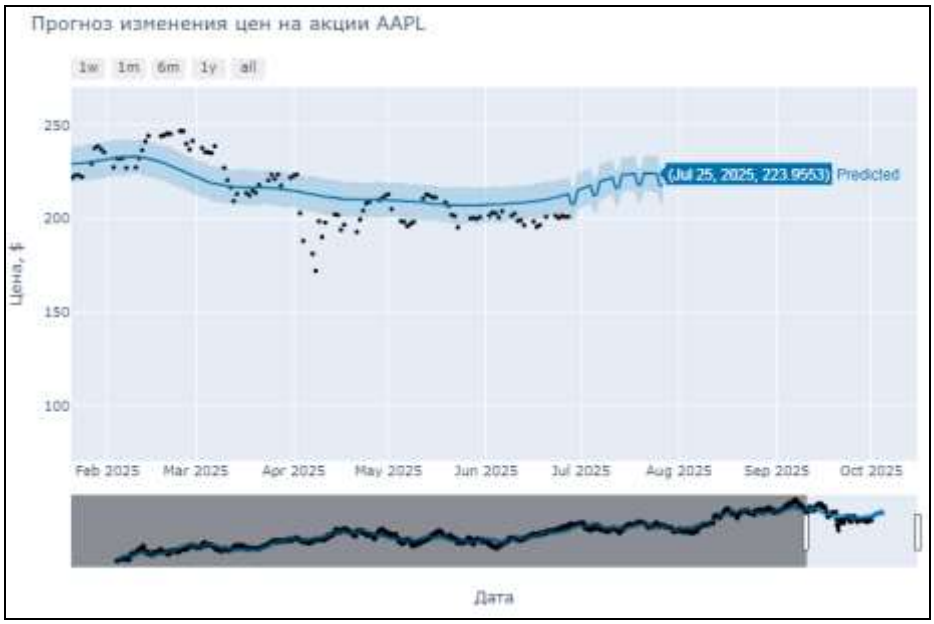


Рис. 9.13. Укрупнение интервала дат в прогнозируемом периоде

Действительно, в течение следующего месяца с некоторыми колебаниями ожидается повышение стоимости акций.

График разложения исходного временного ряда на компоненты показан на рис. 9.14.

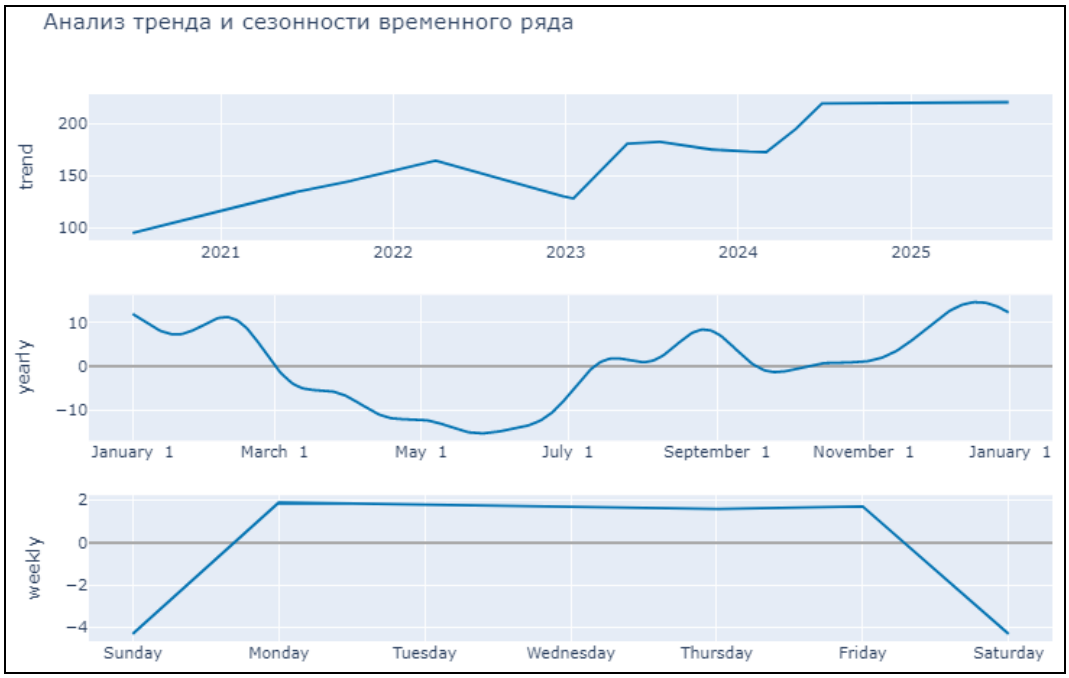


Рис. 9.14. Компоненты исходного временного ряда

Из данного графика видно, что тренд, не смотря на некоторые колебания в разные годы, имеет устойчивую тенденцию к росту.

Годовая сезонность показывает, что в зимние месяцы стоимость акций выше, чем в весенний и летний сезоны.

В течение недели особых колебаний цен нет, за исключением выходных дней (в выходные дни торги на бирже не проводятся).

На графике с прогнозом голубым фоном выделена область доверительного интервала (область возможных ошибок прогноза). Чем больше значение глубины прогноза (дальше заглядываем в будущее), тем шире должен быть интервал возможных ошибок. Поверим это, вернемся в форму выбора параметров и зададим глубину прогноза 365 дней (заглянем на 1 год вперед). Результаты такого прогноза представлены на рис. 9.15.

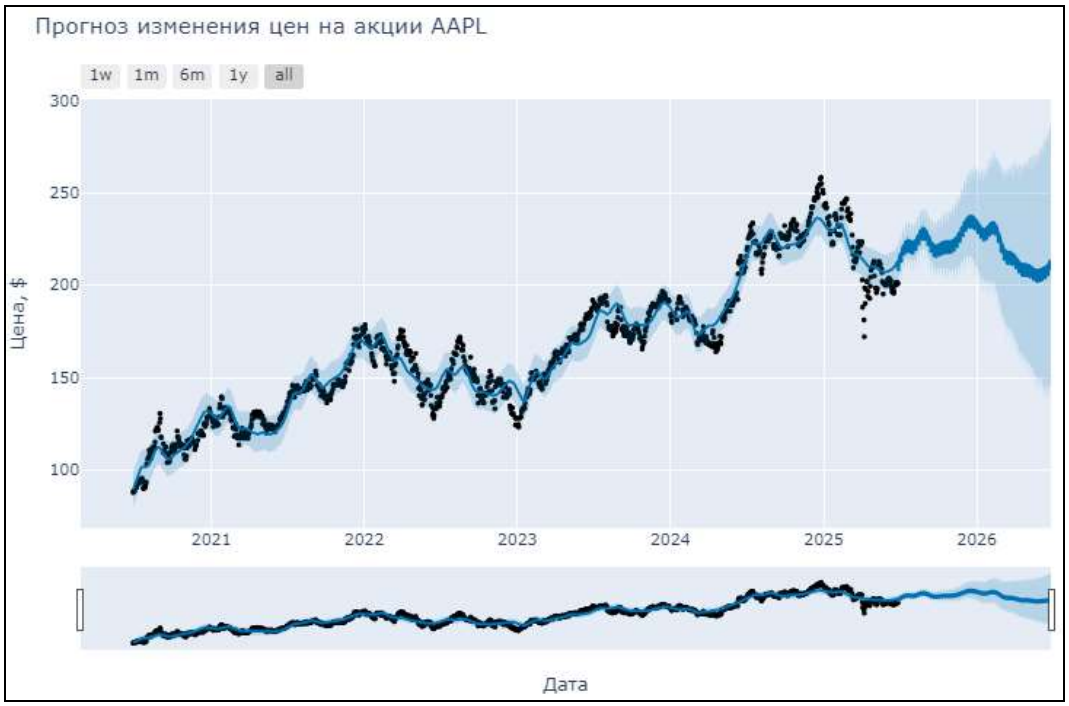


Рис. 9.15. Результаты прогноза с глубиной прогнозирования 1 год.

Как видно из данного рисунка, чем больше глубина прогноза, тем больше вероятность получения ошибочных данных.

В форме выбора параметров запроса к данным пользователь может выбрать сразу несколько фирм, для этого нужно кликнуть левой кнопкой мыши на название фирмы с удержанием клавиши Ctrl (рис. 9.16).

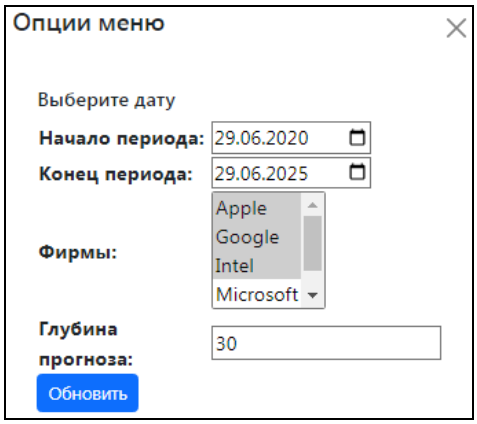


Рис. 9.16. Выбор нескольких фирм в форме изменения параметров запроса к данным

При этом будут загружены данные по котировкам акций всех фирм, и можно будет выполнить сравнение стоимости акций (рис. 9.17).



Рис. 9.17. Сравнение динамики стоимости акций разных фирм

Однако при этом прогноз изменения цены будет показан только по одной фирме, так как в наборе данных только одна колонка получит заголовок «y» (колонка с индексом 1).

9.1.4. Прогнозирование курса валют по данным API Центрального банка

Теперь реализуем программные модули, которые обеспечат загрузку данных о курсах валют из API Центрального банка России, выбор параметров запроса, анализ этих данных и прогнозирование с библиотекой Prophet, а также вывод итогов прогнозирования в графическом виде. С использованием параметров запроса можно будет сделать выбор валюты из списка, изменить интервал дат запрашиваемого периода и задать глубину прогнозирования.

Так как пользователю будет предложен выбор параметров, то в приложении необходимо создать форму. Для данного проекта в файле main/api_finance/forms.py написан код класса Curs_Form – форма для выбора фирмы, задания периода выборки данных и глубины прогнозирования (листинг 9.13).

Листинг 9.13. Программный код класса Curs_Form (форма для выбора валюты, задания периода выборки данных и глубины прогнозирования)

Примечание.
В демо-версии книги листинги программ удалены.

В этой форме объекту years_ago задана дата на пять лет ранее текущей даты, то есть по умолчанию из API будут запрошены данные за последние пять лет. В объект k_val, представляющий вложенный список, загружены коды валют и их названия. Этот список используется для выбора валюты, для которой будет выполнен запрос об изменении ее обменного курса.

После этого создано четыре поля для выбора данных:

- date_1 – дата начала периода;
- date_2 - дата конца периода;
- kod_val – коды валюты (из API центрального банка);
- deep_prog – глубина прогноза.

Для начала периода по умолчанию задана дата, на 5 лет ранее текущей даты, а для конца периода задана текущая дата. По умолчанию будет сформирован период, представляющий последние пять лет от текущей даты. Для выбора дат задан виджет, который представляет собой календарь. Для выбора валюты задан виджет, представляющий собой выпадающий список с множеством выбора. То есть пользователь сможет выбрать любое количество валют в любом сочетании.

Примечание.
При выборе нескольких валют будут загружены данные по всем валютам, но прогноз выполнен только по одной из них.

Для выбора глубины прогноза задано поле для ввода целых чисел.

Для решения поставленной задачи в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 9.14).

Листинг 9.14. Ссылка активации маршрута cb_param_fig

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем cb_param_fig (листинг 9.15).

Листинг 9.15. Маршрут URL с именем cb_param_fig

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление cb_param_fig (листинг 9.16).

Листинг 9.16. Представление (view) cb_param_fig

Примечание.
В демо-версии книги листинги программ удалены.

В этом представлении выполнен импорт необходимых библиотек и модуля загрузки и очистки данных API Центрального банка - clear_data.

Примечание.
Программный код модуля clear_data приведен в приложении 4.

В данном представлении при поступлении GET запроса сформированы следующие параметры:

- d1 = date.today() - relativedelta(years=5) – дата начала периода;
- d2 = date.today() - дата конца периода;
- deep_pog = 30 – глубина прогноза (в данном случае это 30 дней);
- title_str – заголовок страницы;
- name_val – текстовая строка;
- form = Stock_Form() – форма для выбора параметров запроса к данным.

Здесь же даты начала и конца периода приводятся к строковому типу европейского формата.

```
d1_str = d1.strftime("%d-%m-%Y")
d2_str = d2.strftime("%d-%m-%Y")
```

В таком более привычном виде они будут отображаться в шаблоне.

Затем все полученные параметры загружаются в контекстный словарь context, который передается в шаблон с именем cb_param_fig.html. Этот шаблон имеет следующий код (листинг 9.17).

Листинг 9.17. Шаблон cb_param_fig.html

Примечание.
В демо-версии книги листинги программ удалены.

Этот шаблон имеет два раздела и несколько блоков. Первый раздел содержит основную страницу, второй раздел представлен в виде боковой всплывающей панели.

Первый раздел имеет шесть блоков:

- блок для вывода заголовка страницы;
- блок с кнопкой для открытия боковой панели;
- блок для отображения параметров запроса;
- блок для вывода графика исторических данных;
- блок для вывода графика с прогнозом;
- блок для вывода структуры исторических данных.

На боковой всплывающей панели находится форма для выбора параметров запроса к API Центрального банка.

При первоначальной загрузке этот шаблон будет иметь следующий вид (рис. 9.18).

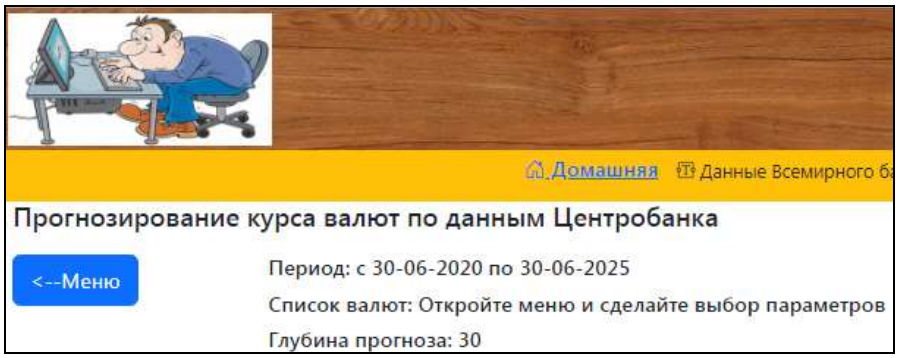


Рис. 9.18. Вид шаблона cb_param_fig.html при первоначальной загрузке.

В этом разделе страницы показан блок с кнопкой открытия боковой панели и параметрами запроса к данным, заданным по умолчанию:

- дата начала периода;
- дата конца периода,
- поясняющий текст, предлагающий нажать на кнопку «Меню» (для открытия боковой панели);
- глубина прогноза, заданная по умолчанию.

При нажатии на кнопку «Меню» с левой стороны страницы выдвигается боковая панель с формой для изменения параметров запроса (рис. 9.19).

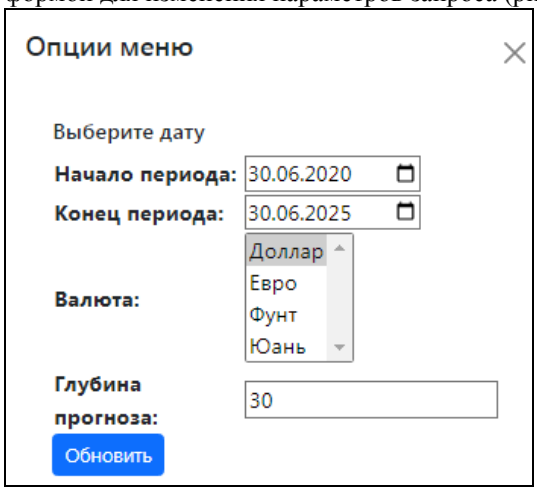


Рис. 9.19. Выбор валюты в форме изменения параметров запроса к данным

В этой форме пользователь может изменить дату начала и конца периода, из предложенного списка выбрать одну или несколько валют, изменить значение глубины прогноза. Одна валюта выбирается простым нажатием левой кнопкой мыши на названии валюты. Если требуется выбрать несколько валют, то делается нажатие левой кнопкой мыши на название валюты с удерживанием клавиши Ctrl.

После нажатия кнопки «Обновить» боковая панель закроется, а в блоках отображения графических данных появятся соответствующие графики.

Вернемся к программному коду представления и разберем программный код, который обрабатывает нажатие кнопки «Обновить».

После нажатия на кнопку «Обновить» в представлении cb_param_fig начинает работать программный код обработки POST запроса, следующий за следующей инструкцией:

```
if request.method == "POST":
```

Здесь, после проверки валидности формы, из нее загружаются следующие параметры:

- d1 – дата начала периода;
- d2 – дата окончания периода;
- list_val – список выбранных валют;
- deep_prog – глубина прогноза.

После получения данных из формы делается проверка значения глубины прогноза. Если пользователь ошибся и указал нулевое или отрицательное значение глубины прогноза, то ему присваивается минимально допустимое значение – 1.

Затем создается объект fig для формирования графика исторических данных и организован цикл с использованием инструкции for. В теле цикла делается перебор валют, выбранных пользователем в форме, и для каждой валюты делается запрос данных к API Центрального банка. Полученные данные сохраняются в объекте df, и для них стоится график исторических данных (fig.add_trace):

```
try:
    df = clear_data.clear_data(d1_str, d2_str, k_val)
except Exception:
    title_str = 'Ошибка загрузки данных API ЦБ, повторите загрузку позже!'
    context = {'title_str': title_str}
    return render(request, "api_finance/cb_param_fig.html", context)

fig.add_trace(go.Scatter(x=df['Date'], y=df['Value'], name=name))
```

> ***Примечание.***
> Для загрузки и очистки данных через API Центрального банка используется ранее разработанный программный модуль clear_data. Код модуля clear_data приведен в приложении 4.

Процедура запроса данных к API Центрального банка заключена в тело инструкций try – except. Это сделано для того, чтобы избежать прекращения работы программы по ошибке в случае невозможности загрузки данных (проблемы с интернетом, изменились условия доступа к данным через API, профилактика сервера и т.п.).

По завершению цикла выполняются стандартные процедуры формирования макета графика с историческими данными (объект fig), и его приведение к формату для отображения в шаблоне Django (объект plot_div).

На следующем шаге структура полученного набора данных df приводится к виду, в котором его может принять библиотека Prophet:

```
df = df[['Date', 'Value']]
df.rename(columns={'Date': 'ds', 'Value': 'y'}, inplace=True)
```

В этих инструкциях из набора данных df выбираются только две колонки (Date, Value), и им присваиваются имена «ds» и «y». Здесь:

- ds – колонка с датами:
- y – колонка с данными (курс валюты).

Затем создается модель (объект model) и выполняется ее обучение:

```
model = Prophet(yearly_seasonality=True)
model.fit(df)
```

Теперь на основе обученной модели можно сделать прогноз на выбранную пользователем глубину:

```
future = model.make_future_dataframe(periods=deep_prog, freq='D')
forecast = model.predict(future)
```

После этого формируется график с результатами прогноза, который приводится к формату для отображения в шаблоне Django (объект prog_html).

Аналогичным образом формируется график анализа компонент временного ряда (тренда и сезонности), который то же приводится к формату для отображения в шаблоне Django (объект comp_html).

Затем все полученные объекты через контекстную переменную context передаются в шаблон cb_param_fig.html.

Теперь можно проверить работу данных модулей.

> ***Примечание.***
> В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 9» и активировать ссылку «Прогноз курса валют по данным API ЦБ».

Если выполнить набор вышеприведенных модулей с датой начала периода август 2022 года, то получим график с исходным набором данных изменения курса доллара (рис. 9.20).



Рис. 9.20. Временной ряд с котировкой акций компании Apple

Как видно из данного графика, до 29 ноября 2024 года курс доллара колебался, но имел тенденцию к постоянному росту. После этой даты с некоторыми колебаниями прослеживается устойчивая тенденция снижения курса.

График прогноза изменения курса доллара на будущий период в 30 дней показан на рис. 9.21.

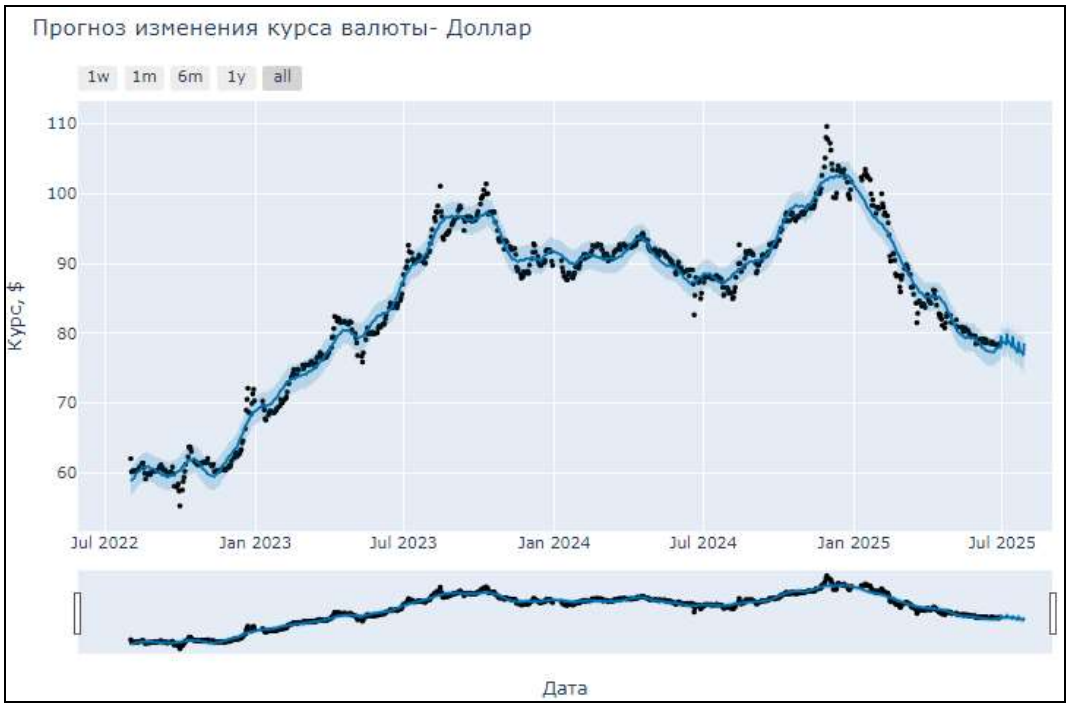


Рис. 9.21. Прогноз изменения курса доллара на будущий период в 30 дней

Как видно из данного графика, линия прогноза достаточно хорошо описывает исторические данные, а в ближайшие 30 дней ожидается снижение курса доллара. Так как этот график является интерактивным, можно с помощью слайдера укрупнить интервал дат и посмотреть на более детальную информацию прогноза (рис. 9.22).

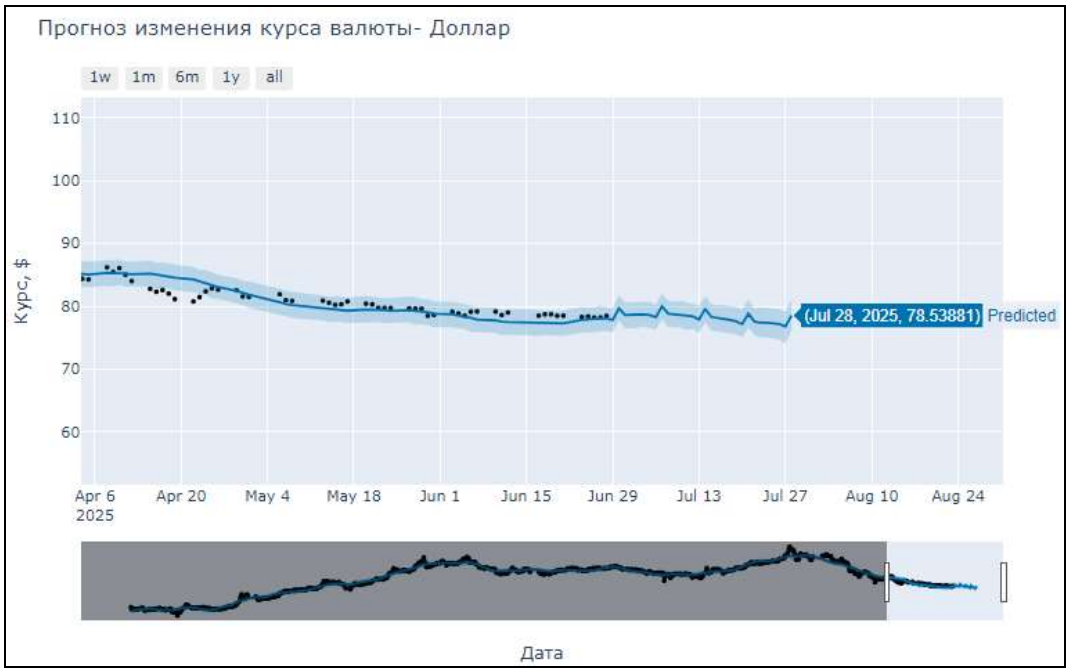


Рис. 9.22. Укрупнение интервала дат в прогнозируемом периоде

Действительно, в течение следующего месяца с некоторыми колебаниями ожидается снижение курса доллара.

График разложения исходного временного ряда на компоненты показан на рис. 9.23.



Рис. 9.23. Компоненты исходного временного ряда

Из данного графика видно, что тренд, не смотря на некоторые колебания в разные годы, сначала имел тенденцию к росту, а с ноября 2024 наметилась устойчивое снижение тренда.

Годовая сезонность показывает, что в осенний и зимний периоды курс доллара выше, чем в весенний и летний сезоны.

В течение недели особых колебаний курса нет, за исключением понедельника (в этот день он, как правило, выше, чем в другие дни недели).

В форме выбора параметров запроса к данным пользователь может выбрать сразу несколько валют, для этого нужно кликнуть левой кнопкой мыши на название валюты с удержанием клавиши Ctrl (рис. 9.24).

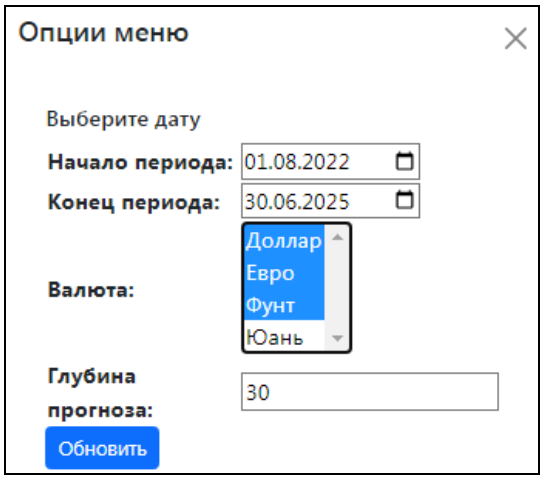


Рис. 9.24. Выбор нескольких валют в форме изменения параметров запроса к данным

При этом будут загружены данные по курсам нескольких валют, и можно будет выполнить их сравнение (рис. 9.25).



Рис. 9.25. Сравнение динамики курса разных валют

Из данного графика видно, что колебание курса разных валют имеет одинаковый характер. Самый высокий курс имеет английский фунт. В начале выбранного периода курсы евро и доллара практически одинаковы, а к концу периода доллар относительно евро значительно подешевел.

Однако при этом прогноз изменения курса будет показан только по одной валюте, которая в форме выбора параметров будет последней в списке валют.

9.1.5. Прогнозирование цен на драгоценные металлы по данным API Центрального банка

Теперь реализуем программные модули, которые обеспечат загрузку данных о ценах на драгоценные металлы из API Центрального банка России, выбор параметров запроса, анализ этих данных и прогнозирование с библиотекой Prophet, а также вывод итогов прогнозирования в графическом виде. С использованием параметров запроса можно будет сделать выбор драгоценного металла из списка, изменить интервал дат запрашиваемого периода и задать глубину прогнозирования.

Так как пользователю будет предложен выбор параметров, то в приложении необходимо создать форму. Для данного проекта в файле main/api_finance/forms.py написан код класса Drag_met_Form – форма для выбора фирмы, задания периода выборки данных и глубины прогнозирования (листинг 9.18).

Листинг 9.18. Программный код класса Drag_met_Form (форма для выбора драгметалла, задания периода выборки данных и глубины прогнозирования)

Примечание.
В демо-версии книги листинги программ удалены.

В этой форме объекту years_ago задана дата на пять лет ранее текущей даты, то есть по умолчанию из API будут запрошены данные за последние пять лет. В объект k_metl, представляющий вложенный список, загружены коды драгметаллов и их названия. Этот список используется для выбора драгметалла, для которого будет выполнен запрос об изменении его цены.

После этого создано четыре поля для выбора данных:

- date_1 – дата начала периода;
- date_2 - дата конца периода;
- kod_met – коды драгметаллов (из API центрального банка);
- deep_prog – глубина прогноза.

Для начала периода по умолчанию задана дата, на 5 лет ранее текущей даты, а для конца периода задана текущая дата. По умолчанию будет сформирован период, представляющий последние пять лет от текущей даты. Для выбора дат задан виджет, который представляет собой календарь. Для выбора драгметалла задан виджет, представляющий собой выпадающий список с множеством выбора. То есть пользователь сможет выбрать любое количество драгметаллов в любом сочетании.

Примечание.
При выборе нескольких драгметаллов будут загружены данные по всем драгметаллам, но прогноз будет выполнен только по одному из них (последнему в выбранном списке).

Для выбора глубины прогноза задано поле для ввода целых чисел.

Для решения поставленной задачи в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 9.19).

Листинг 9.19. Ссылка активации маршрута cb_drag_met_fig

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем cb_drag_met_fig (листинг 9.20).

Листинг 9.20. Маршрут URL с именем cb_drag_met_fig

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление cb_drag_met_fig (листинг 9.21).

Листинг 9.21. Представление (view) cb_drag_met_fig

Примечание.
В демо-версии книги листинги программ удалены.

В этом представлении выполнен импорт необходимых библиотек и модуля загрузки и очистки данных API Центрального банка - clear_data_metall.

Примечание.
Программный код модуля clear_data_metall приведен в приложении 5.

В данном представлении при поступлении GET запроса сформированы следующие параметры:

- d1 = date.today() - relativedelta(years=5) – дата начала периода;
- d2 = date.today() - дата конца периода;
- deep_pog = 30 – глубина прогноза (в данном случае это 30 дней);
- title_str – заголовок страницы;
- name_met – текстовая строка;
- form = Stock_Form() – форма для выбора параметров запроса к данным.

Здесь же даты начала и конца периода приводятся к строковому типу европейского формата.

```
d1_str = d1.strftime("%d-%m-%Y")
d2_str = d2.strftime("%d-%m-%Y")
```

В таком более привычном виде они будут отображаться в шаблоне.

Затем все полученные параметры загружаются в контекстный словарь context, который передается в шаблон с именем cb_drag_met_fig.html. Этот шаблон имеет следующий код (листинг 9.22).

Листинг 9.22. Шаблон cb_drag_met_fig.html

Примечание.
В демо-версии книги листинги программ удалены.

Этот шаблон имеет два раздела и несколько блоков. Первый раздел содержит основную страницу, второй раздел представлен в виде боковой всплывающей панели.

Первый раздел имеет шесть блоков:

- блок для вывода заголовка страницы;
- блок с кнопкой для открытия боковой панели;
- блок для отображения параметров запроса;
- блок для вывода графика исторических данных;
- блок для вывода графика с прогнозом;
- блок для вывода структуры исторических данных.

На боковой всплывающей панели находится форма для выбора параметров запроса к API Центрального банка.

При первоначальной загрузке этот шаблон будет иметь следующий вид (рис. 9.26).

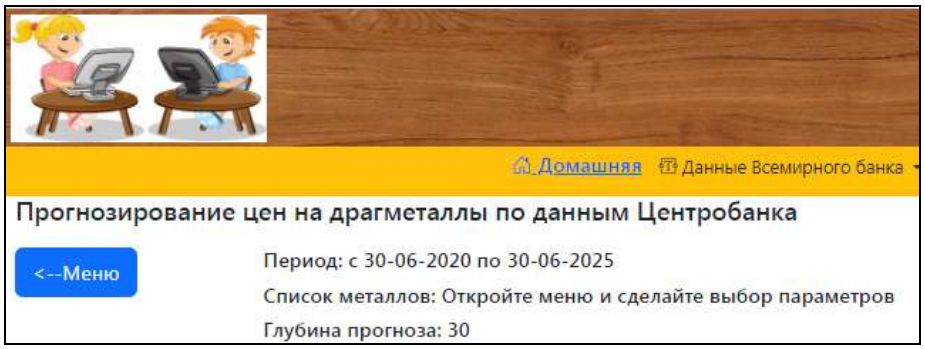


Рис. 9.26. Вид шаблона cb_drag_met_fig.html при первоначальной загрузке.

В этом разделе страницы показан блок с кнопкой открытия боковой панели и параметрами запроса к данным, заданным по умолчанию:

- дата начала периода;
- дата конца периода,
- поясняющий текст, предлагающий нажать на кнопку «Меню» (для открытия боковой панели);
- глубина прогноза, заданная по умолчанию.

При нажатии на кнопку «Меню» с левой стороны страницы выдвигается боковая панель с формой для изменения параметров запроса (рис. 9.27).

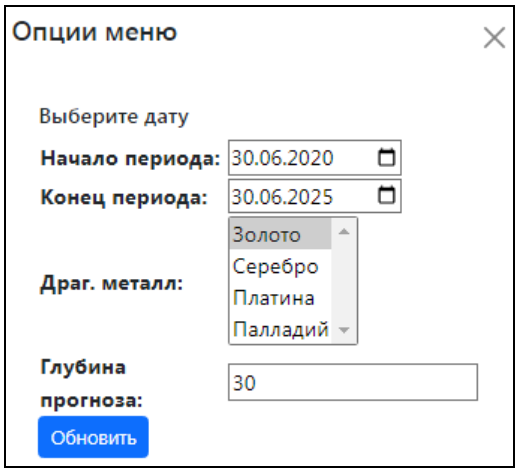


Рис. 9.27. Форма изменения параметров запроса к данным Центрального банка

В этой форме пользователь может изменить дату начала и конца периода, из предложенного списка выбрать один или несколько драгметаллов, изменить значение глубины прогноза. Один тип драгметалла выбирается простым нажатием левой кнопкой мыши на его названии. Если требуется выбрать несколько типов драгметаллов, то делается нажатие левой кнопкой мыши на название драгметалла с удерживанием клавиши Ctrl.

После нажатия кнопки «Обновить» боковая панель закроется, а в блоках отображения графических данных появятся соответствующие графики.

Вернемся к программному коду представления и разберем программный код, который обрабатывает нажатие кнопки «Обновить».

После нажатия на кнопку «Обновить» в представлении cb_drag_met_fig начинает работать программный код обработки POST запроса, следующий за следующей инструкцией:

```
    if request.method == "POST":
```

Здесь, после проверки валидности формы, из нее загружаются следующие параметры:

- d1 – дата начала периода;
- d2 – дата окончания периода;
- list_met – список выбранных типов драгметаллов;
- deep_prog – глубина прогноза.

После получения данных из формы делается проверка значения глубины прогноза. Если пользователь ошибся и указал нулевое или отрицательное значение глубины прогноза, то ему присваивается минимально допустимое значение – 1.

На следующем шаге делается запрос к API Центрального банка и в объект dat загружаются исторические данные динамике цен на драгметаллы:

```
try:
    dat = clear_data_metall.clear_data(d1_str, d2_str)
except Exception:
    title_str = 'Ошибка загрузки данных API ЦБ, повторите загрузку позже!'
    context = {'title_str': title_str}
    return render(request, "api_finance/cb_param_fig.html", context)
```

Примечание.
Для загрузки и очистки данных через API Центрального банка используется ранее разработанный программный модуль clear_data_metall. Код модуля clear_data_metall приведен в приложении 5.

Процедура запроса данных к API Центрального банка заключена в тело инструкций try – except. Это сделано для того, чтобы избежать прекращения работы программы по ошибке в случае невозможности загрузки данных (проблемы с интернетом, изменились условия доступа к данным через API, профилактика сервера и т.п.).

В объект dat будут загружены данные по четырем типам драгметаллов, не зависимо от того, какой из них был задан в форме выбора параметров. Поэтому далее создается объект fig для формирования графика исторических данных и организован цикл с использованием инструкции for. В теле цикла при формировании графика будут использованы данные только о тех драгметаллов, которые пользователь выбрал в форме. По завершению цикла выполняются стандартные процедуры формирования макета графика с историческими данными (объект fig), и его приведение к формату для отображения в шаблоне Django (объект plot_div).

На следующем шаге структура последнего полученного набора данных df приводится к виду, в котором его может принять библиотека Prophet:

```
df = df[['Date', 'Sell']]
df.rename(columns={'Date': 'ds', 'Sell': 'y'}, inplace=True)
```

В этих инструкциях из набора данных df выбираются только две колонки (Date, Sell), и им присваиваются имена «ds» и «y». Здесь:

- ds – колонка с датами:
- y – колонка с данными (цена драгметалла).

Затем создается модель (объект model) и выполняется ее обучение:

```
model = Prophet(yearly_seasonality=True)
model.fit(df)
```

Теперь на основе обученной модели можно сделать прогноз на выбранную пользователем глубину:

```
future = model.make_future_dataframe(periods=deep_prog, freq='D')
forecast = model.predict(future)
```

После этого формируется график с результатами прогноза, который приводится к формату для отображения в шаблоне Django (объект prog_html).

Аналогичным образом формируется график анализа компонент временного ряда (тренда и сезонности), который то же приводится к формату для отображения в шаблоне Django (объект comp_html).

Затем все полученные объекты через контекстную переменную context передаются в шаблон cb_drag_met_fig.html.

Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 9» и активировать ссылку «Прогноз цен на драгметаллы по данным API ЦБ».

Если выполнить набор вышеприведенных модулей с датой начала периода июль 2022 года, то получим график с исходным набором данных изменения цен на золото (рис. 9.28).



Рис. 9.28. Временной ряд с ценами на золото по данным API Центробанка

Как видно из данного графика, цена на золото с некоторыми колебаниями имеет тенденцию к росту. После этой даты с некоторыми колебаниями прослеживается устойчивая тенденция снижения курса.

График прогноза изменения цены золота на будущий период в 30 дней показан на рис. 9.29.

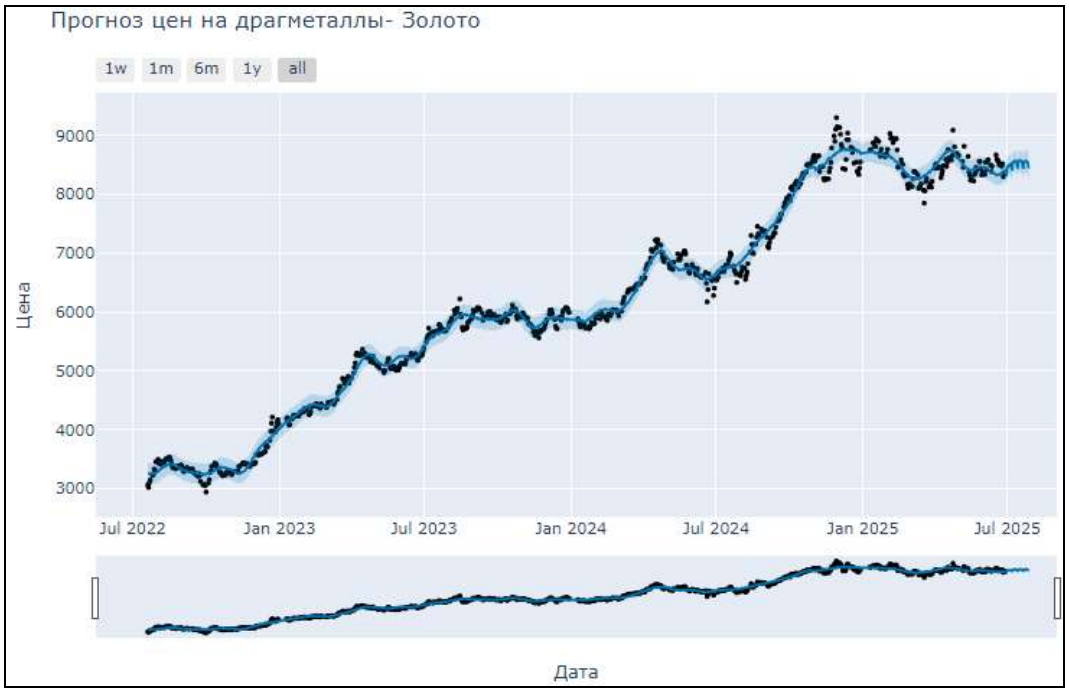


Рис. 9.29. Прогноз изменения курса доллара на будущий период в 30 дней

Как видно из данного графика, линия прогноза достаточно хорошо описывает исторические данные, а в ближайшие 30 дней ожидается небольшой рост цены на золото. Так как этот график является интерактивным, можно с помощью слайдера укрупнить интервал дат и посмотреть на более детальную информацию прогноза (рис. 9.30).

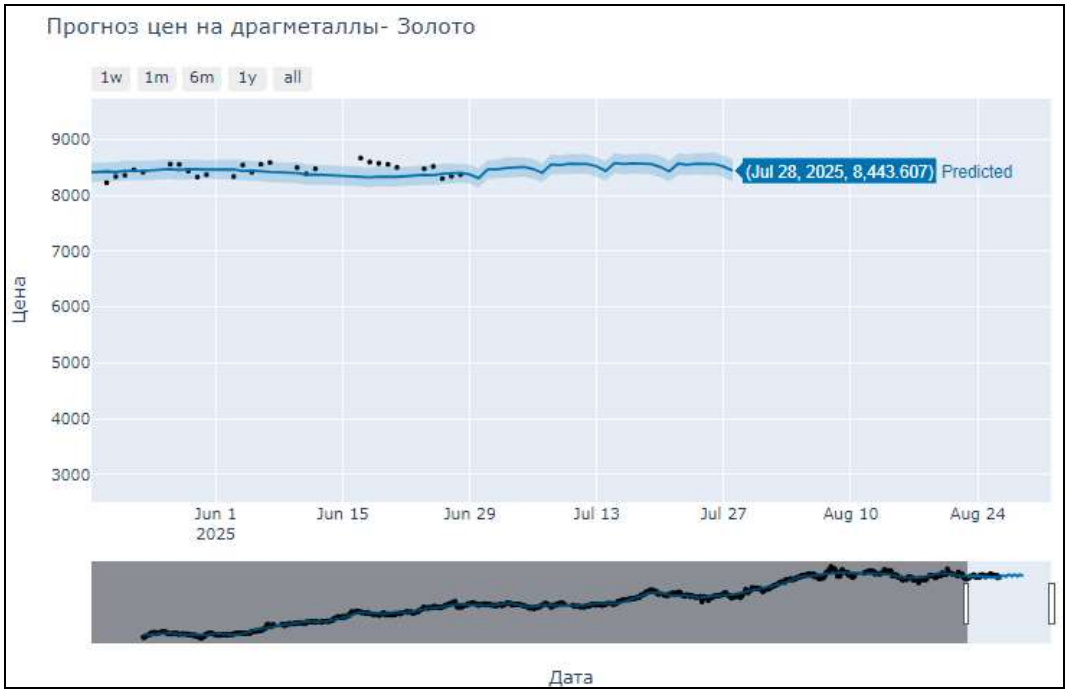


Рис. 9.30. Укрупнение интервала дат в прогнозируемом периоде

Действительно, в течение следующего месяца с некоторыми колебаниями ожидается небольшой рост цены на золото.

График разложения исходного временного ряда на компоненты показан на рис. 9.31.

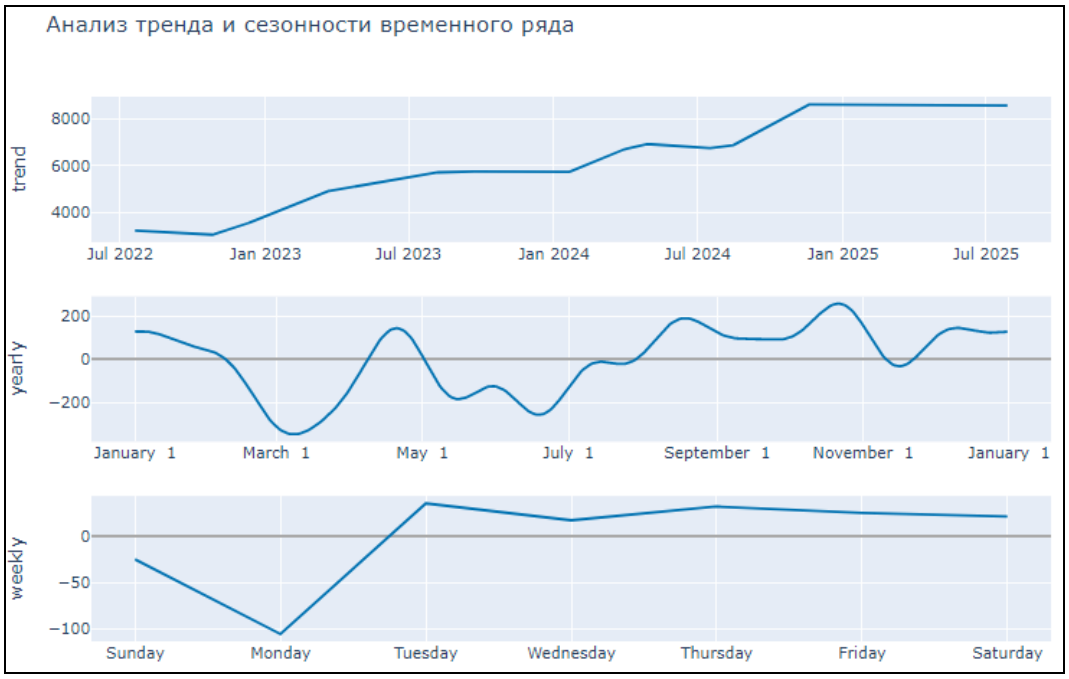


Рис. 9.31. Компоненты исходного временного ряда

Из данного графика видно, что тренд, не смотря на некоторые колебания в разные годы, имеет тенденцию к постоянному росту.

Годовая сезонность показывает, что весной и летом (за исключением мая) цены на золото двигаются в сторону снижения, а в осенний и зимний сезон – двигаются в сторону повышения.

В течение недели особых колебаний цен не золото нет, за исключением понедельника (в этот день он, как правило, ниже, чем в другие дни недели).

В форме выбора параметров запроса к данным пользователь может выбрать сразу несколько видов драгметаллов, для этого нужно кликнуть левой кнопкой мыши на название драгметалла с удержанием клавиши Ctrl (рис. 9.32).

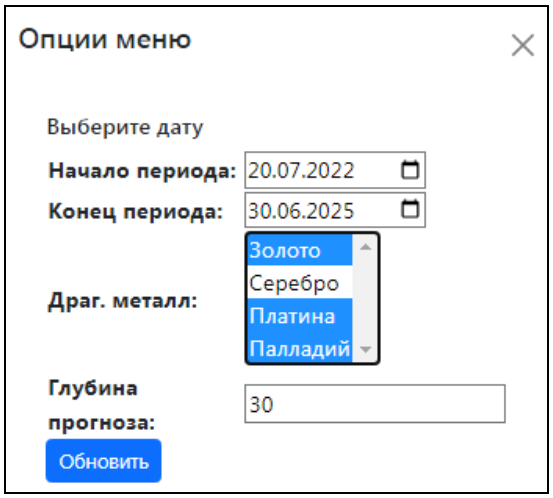


Рис. 9.32. Выбор нескольких видов драгметаллов в форме изменения параметров запроса к данным

При этом будут загружены данные по динамике цен по нескольким видам драгметаллов, и можно будет выполнить их сравнение (рис. 9.33).



Рис. 9.33. Сравнение динамики цен на драгметаллы

Из данного графика видно, что динамика изменения цен для разных драгметаллов имеет разный характер. Цена на золото самая высокая и в заданном периоде постоянно растет. Цена на палладий имеет тенденцию к падению, а цена на платину имеет небольшой рост.

Однако при этом прогноз изменения цены будет показан только по одному виду драгметалла, который в форме выбора параметров будет последним в списке.

9.1.6. Прогнозирование экономических показателей стран по данным API Всемирного банка

Теперь реализуем программные модули, которые обеспечат загрузку данных об экономических показателях стран из API Всемирного банка, выбор параметров запроса, анализ этих данных и прогнозирование с библиотекой Prophet, а также вывод итогов прогнозирования в графическом виде. С использованием параметров запроса можно будет сделать выбор страны и экономического индикатора из списка, изменить интервал дат запрашиваемого периода и задать глубину прогноза.

Так как пользователю будет предложен выбор параметров, то в приложении необходимо создать форму. Для данного проекта в файле main/api_finance/forms.py написан код класса Wb_Form – форма для выбора страны, экономического индикатора, задания периода выборки данных и глубины прогнозирования (листинг 9.23).

Листинг 9.23. Программный код класса Wb_Form (форма для выбора драгметалла, задания периода выборки данных и глубины прогнозирования)

Примечание.
В демо-версии книги листинги программ удалены.

В этой форме объекту years_ago задана дата на 20 лет ранее текущей даты, то есть по умолчанию из API будут запрошены данные за последние двадцать лет (в API Всемирного банка экономические показатели стран имеют ежегодные значения).

В объект countries, представляющий вложенный список, загружены коды стран и их названия. Этот список используется для выбора страны, для которой будет выполнен запрос об изменении экономических показателей.

В объект indicators, представляющий вложенный список, загружены коды экономических показателей и их расшифровка. Этот список используется для выбора экономического индикатора, для которого будет выполнен запрос об его изменении.

После этого создано пять полей для выбора данных:

- date_1 – дата начала периода;
- date_2 - дата конца периода;
- kod_country – коды страны (из API Всемирного банка);
- kod_indic – коды экономического индикатора (из API Всемирного банка);
- deep_prog – глубина прогноза.

Для начала периода по умолчанию задана дата, на 20 лет ранее текущей даты, а для конца периода задана текущая дата. По умолчанию будет сформирован период, представляющий последние двадцать лет от текущей даты. Для выбора дат задан виджет, который представляет собой календарь. Для выбора страны и экономического индикатора задан виджет, представляющий собой выпадающий список с множеством выбора. То есть пользователь сможет выбрать любое количество параметров в любом сочетании.

Примечание.
При выборе нескольких стран будут загружены данные по всем странам и данные только по одному экономическому индикатору, но прогноз будет выполнен только для одной из стран (последней в выбранном списке).

Для выбора глубины прогноза задано поле для ввода целых чисел.

Для решения поставленной задачи в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 9.24).

Листинг 9.24. Ссылка активации маршрута wb_param_fig

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем wb_param_fig (листинг 9.25).

Листинг 9.25. Маршрут URL с именем wb_param_fig

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление wb_param_fig (листинг 9.26).

Листинг 9.26. Представление (view) wb_param_fig

Примечание.
В демо-версии книги листинги программ удалены.

В этом представлении выполнен импорт необходимых библиотек и модуля загрузки и очистки данных API Всемирного банка - fn_wb_data.

Примечание.
Программный код модуля fn_wb_data приведен в приложении 6.

Далее сформирован словарь с экономическими индикаторами (объект dic_indic), и вложенный список с кодами и наименования стран (объект countries). Эти объекты нам понадобятся при выборе названий стран и расшифровки экономических индикаторов для их отображения в шаблонах Django.

В данном представлении при поступлении GET запроса сформированы следующие параметры:

- d1 = date.today() - relativedelta(years=20) – дата начала периода;
- d2 = date.today() - дата конца периода;
- name_country – текстовая строка для получения названия страны;
- name_indic – текстовая строка для получения индикатора;
- title_str – заголовок страницы;

- deep_pog = 2 – глубина прогноза (в данном случае это 2 года);
- form = Wb_Form() – форма для выбора параметров запроса к данным.

Здесь же даты начала и конца периода приводятся к строковому типу европейского формата.

```
d1_str = d1.strftime("%d-%m-%Y")
d2_str = d2.strftime("%d-%m-%Y")
```

В таком более привычном виде они будут отображаться в шаблоне.

Затем все полученные параметры загружаются в контекстный словарь context, который передается в шаблон с именем wb_param_fig.html. Этот шаблон имеет следующий код (листинг 9.27).

Листинг 9.27. Шаблон wb_param_fig.html

Примечание.
В демо-версии книги листинги программ удалены.

Этот шаблон имеет два раздела и несколько блоков. Первый раздел содержит основную страницу, второй раздел представлен в виде боковой всплывающей панели.

Первый раздел имеет шесть блоков:

- блок для вывода заголовка страницы;
- блок с кнопкой для открытия боковой панели;
- блок для отображения параметров запроса;
- блок для вывода графика исторических данных;
- блок для вывода графика с прогнозом;
- блок для вывода структуры исторических данных.

На боковой всплывающей панели находится форма для выбора параметров запроса к API Всемирного банка.

При первоначальной загрузке этот шаблон будет иметь следующий вид (рис. 9.34).

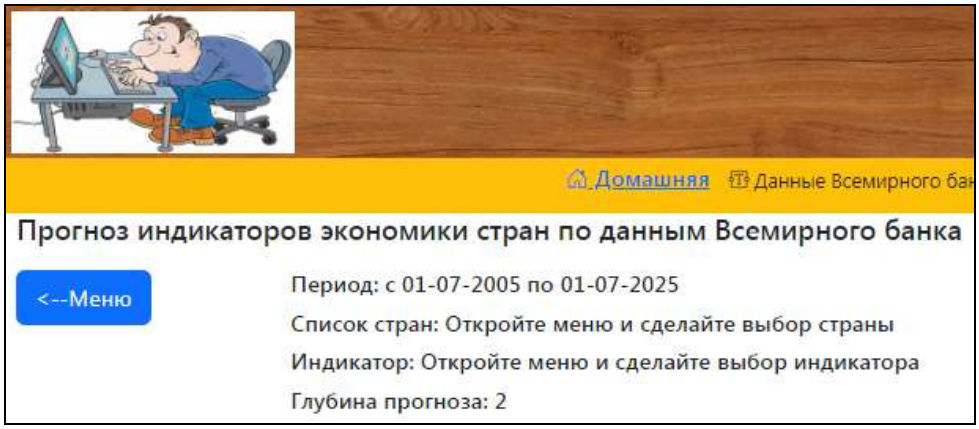


Рис. 9.34. Вид шаблона wb_param_fig.html при первоначальной загрузке.

В этом разделе страницы показан блок с кнопкой открытия боковой панели и параметрами запроса к данным, заданным по умолчанию:

- дата начала периода;
- дата конца периода,
- поясняющий текст, предлагающий нажать на кнопку «Меню» (для открытия боковой панели);
- глубина прогноза, заданная по умолчанию.

При нажатии на кнопку «Меню» с левой стороны страницы выдвигается боковая панель с формой для изменения параметров запроса (рис. 9.35).

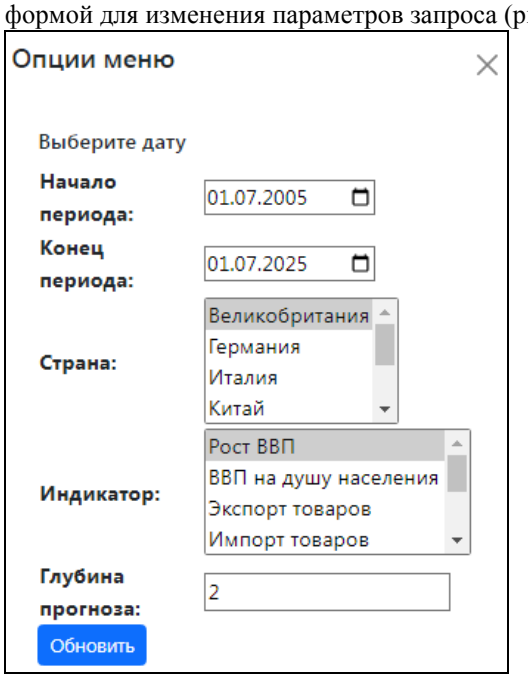


Рис. 9.35. Форма выбора параметров запроса к данным Всемирного банка

В этой форме пользователь может изменить дату начала и конца периода, из предложенного списка выбрать одну или несколько стран, выбрать один индикатор экономического показателя, изменить значение глубины прогноза. Одна страна выбирается простым нажатием левой кнопкой мыши на его названии. Если требуется выбрать несколько стран, то делается нажатие левой кнопкой мыши на название страны с удерживанием клавиши Ctrl.

После нажатия кнопки «Обновить» боковая панель закроется, а в блоках отображения графических данных появятся соответствующие графики.

Вернемся к программному коду представления и разберем программный код, который обрабатывает нажатие кнопки «Обновить».

После нажатия на кнопку «Обновить» в представлении wb_param_fig начинает работать программный код обработки POST запроса, следующий за следующей инструкцией:

```
if request.method == "POST":
```

Здесь, после проверки валидности формы, из нее загружаются следующие параметры:

- d1 – дата начала периода;
- d2 – дата окончания периода;
- list_countryt – список выбранных стран;
- list_indic – список экономических индикаторов;
- deep_prog – глубина прогноза.

После получения данных из формы делается проверка значения глубины прогноза. Если пользователь ошибся и указал нулевое или отрицательное значение глубины прогноза, то ему присваивается минимально допустимое значение – 1. Если пользователь указал глубину прогноза более 20 лет, то ему присваивается значение 20 (при большой глубине прогноза теряется достоверность данных).

Далее следуют два блока, в которых формируются названия выбранных стран (объект name_country), и расшифровка выбранного экономического индикатора (объект indicator).

На следующем шаге делается запрос к API Всемирного банка и в объект dat загружаются исторические данные динамике цен на драгметаллы:

```
try:
   df = fn_wb_data.get_data_wb(indicator, list_country, d1_str, d2_str)
   # Удалить строки с пустыми значениями NaN
   df = df.dropna()
except Exception:
   title_str = 'Ошибка загрузки данных API ВБ, повторите загрузку позже!'
   context = {'title_str': title_str}
   return render(request, "api_finance/wb_param_fig.html", context)
```

> ***Примечание.***
> Для загрузки и очистки данных через API Всемирного банка используется ранее разработанный программный модуль fn_wb_data. Код модуля fn_wb_data приведен в приложении 6.

Процедура запроса данных к API Всемирного банка заключена в тело инструкций try – except. Это сделано для того, чтобы избежать прекращения работы программы по ошибке в случае невозможности загрузки данных (проблемы с интернетом, изменились условия доступа к данным через API, профилактика сервера и т.п.). Кроме того, здесь из полученного набора данных удаляются строки с пустыми данными. Дело в том, что Всемирный банк публикует данные об экономических показателях стран с задержкой в 1-2 года, поэтому в запросе могут отсутствовать данные за текущий и даже за предыдущий год.

В объект df могут быть загружены данные для одной, или для нескольких стран, в зависимости от этого набор данных df будут иметь разную структуру. Поэтому далее из набора данных df формируется сводная таблица с единой структурой (объект df_pivot):

```
if len(list_country) >= 2:
   df = df.reset_index()
   df_pivot = df.pivot(index='date', columns='country', values=ind_val)
else:
   df_pivot = df
```

После этого создается объект fig - график исторических данных, который трансформирован в формат для отображения в шаблоне Django (объект plot_div).

На следующем шаге структура полученного набора данных (df или df_pivot, в зависимости от количества выбранных стран) приводится к виду, в котором его может принять библиотека Prophet:

```
if len(list_country) >= 2:  # Если выбрано более 2-х стран
   # Убрать индекс
   df_pivot.reset_index(level=0, inplace=True)
   # Переименовать колонки по их индексу
   df_pivot.columns.values[[0, 1]] = ['ds', 'y']
   # Выбрать две колонки
   df = df_pivot.take([0, 1], axis=1)
else:  # Если выбрана одна страна
   # Убрать индекс
   df.reset_index(level=0, inplace=True)
   # Переименовать колонки по их индексу
   df.columns.values[[0, 1]] = ['date', 'value']
   # Переименовать колонки по их именам
   df.rename(columns={'date': 'ds', 'value': 'y'}, inplace=True)
```

В этих инструкциях из наборов данных (df или df_pivit) выбираются только две колонки (с индексами 0 и 1), и им присваиваются имена «ds» и «y». Здесь:

- ds – колонка с датами:
- y – колонка с данными (экономический индикатором).

Затем создается модель (объект model) и выполняется ее обучение:

```
model = Prophet(yearly_seasonality=False)
model.fit(df)
```

Поскольку исходные данные содержат годовые показатели, то у них нет месячной и недельной сезонности, поэтому параметр yearly_seasonality имеет значение – False.

Теперь на основе обученной модели можно сделать прогноз на выбранную пользователем глубину:

```
future = model.make_future_dataframe(periods=deep_prog, freq='Y')
forecast = model.predict(future)
```

Поскольку исходные данные содержат годовые показатели, то параметр freq имеет значение – «Y».

После этого формируется график с результатами прогноза, который приводится к формату для отображения в шаблоне Django (объект prog_html).

Аналогичным образом формируется график анализа компонент временного ряда (тренда и сезонности), который тоже приводится к формату для отображения в шаблоне Django (объект comp_html).

Затем все полученные объекты через контекстную переменную context передаются в шаблон wb_param_fig.html.

Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 9» и активировать ссылку «Прогноз показателей экономики стран из API ВБ».

Если выполнить набор вышеприведенных модулей с выбором параметров: страна – Великобритания, индикатор – Импорт товаров, то получим следующий график с историческими данными (рис. 9.36).

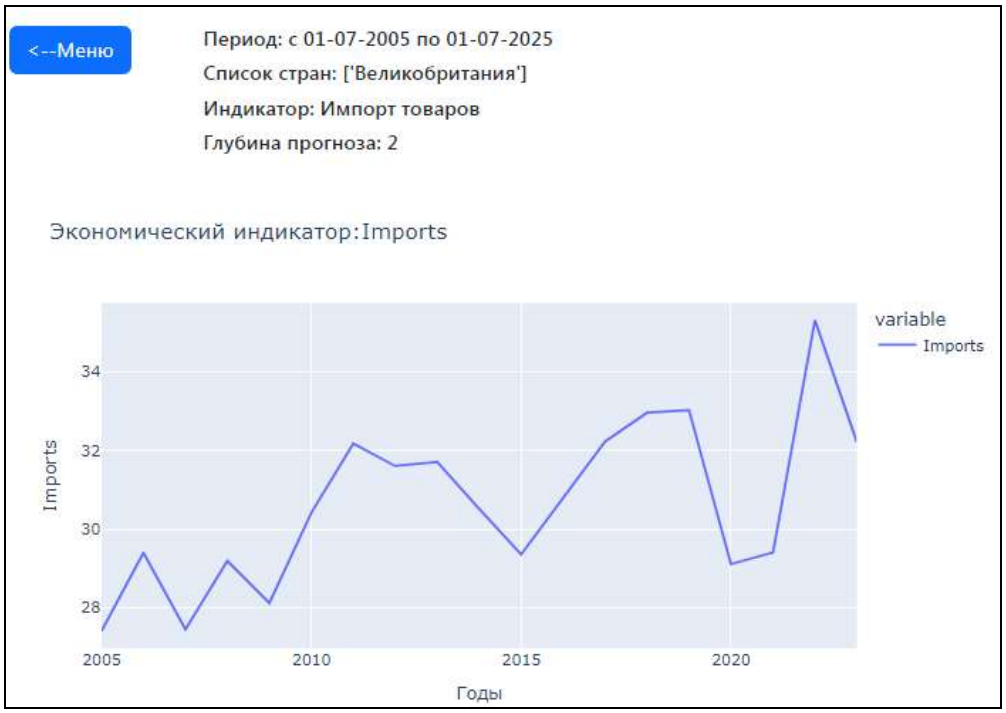


Рис. 9.36. Временной ряд с динамикой экономического индикатора по данным API Всемирного банка

Как видно из данного графика, для Великобритании импорт товаров с некоторыми колебаниями имеет тенденцию к росту.

График прогноза изменения импорта товаров на будущий период в 2 года показан на рис. 9.37.

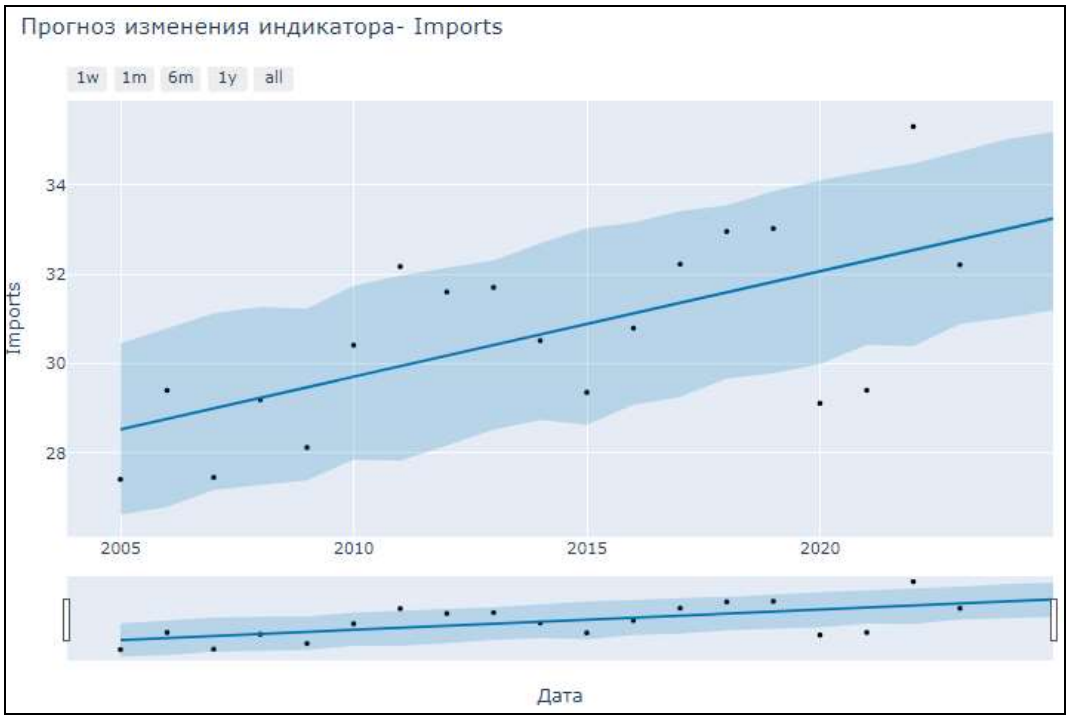


Рис.9.37. Прогноз изменения импорта товаров на будущий период в 2 года

Как видно из данного графика, линия прогноза не очень хорошо описывает исторические данные (доверительный интервал достаточно большой). Но в ближайшие два года ожидается небольшой прирост импорта товаров.

График разложения исходного временного ряда на компоненты показан на рис. 9.38.



Рис. 9.38. Компоненты исходного временного ряда

Из данного графика видно, что тренд имеет тенденцию к постоянному росту. Поскольку у данного временного ряда нет месячной и недельной сезонности, то эти графики не выводятся.

В форме выбора параметров запроса к данным пользователь может выбрать сразу несколько стран, для этого нужно кликнуть левой кнопкой мыши на название страны с удержанием клавиши Ctrl (рис 9.39).

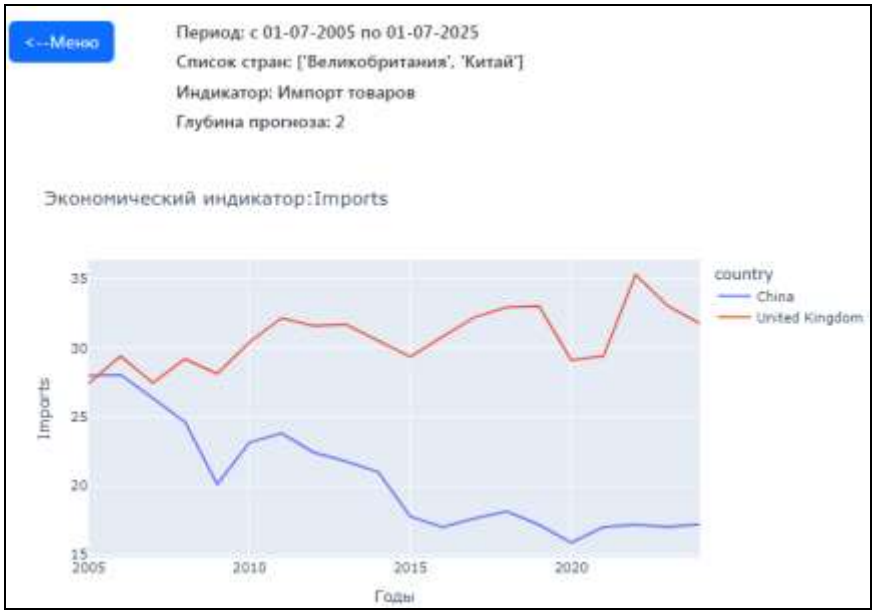


Рис. 9.39. Сравнение экономических индикаторов для разных стран

Из данного графика видно, что динамика импорта товаров для разных стран имеет разный характер. Импорт товаров для Великобритании постоянно растет, а для Китая падает. Однако при этом прогноз изменения экономического индикатора будет показан только по одной стране, который в форме выбора параметров будет последней в списке. В данном случае прогноз будет показан по Китаю (рис. 9.40).

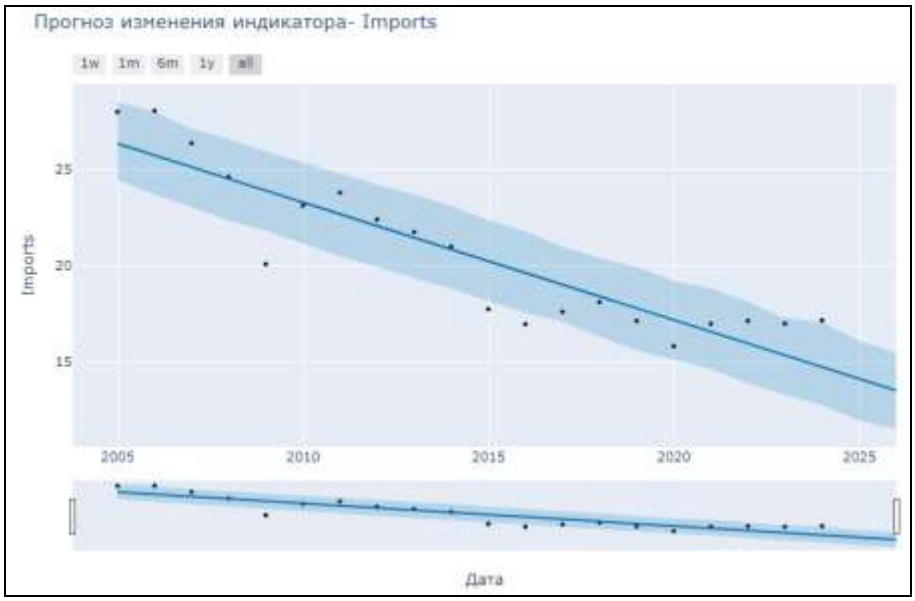


Рис. 9.40. Прогноз импорта товаров для Китая

Для данного временного ряда можно получить боле уверенный прогноз, если использовать другие статистические модели.

9.1.7. Прогнозирование продаж торгового центра

Рассмотрим возможности данной библиотеки на примере оценки прогноза продаж товаров на основе данных, загруженных из CSV файла. В данном примере будет использован файл с тестовыми данными, однако их основой являются реальные данные торгового центра за четыре года. Для этого в одном из шаблонов или элементе меню необходимо создать ссылку, которая активирует нужный URL маршрут (листинг 9.28).

Листинг 9.28. Ссылка активации маршрута prog_sales_fig

Примечание.
В демо-версии книги листинги программ удалены.

По данной ссылке в приложении api_finance будет активирован URL маршрут с именем prog_sales_fig (листинг 9.29).

Листинг 9.29. Маршрут URL с именем prog_sales_fig

Примечание.
В демо-версии книги листинги программ удалены.

По этому маршруту будет активировано представление prog_sales_fig (листинг 9.30).

Листинг 9.30. Представление (view) prog_sales_fig

Примечание.
В демо-версии книги листинги программ удалены.

В этом модуле выполнен импорт необходимых библиотек, задан путь к CSV файлу (объект path_csv) и из него загружены данные о доходах от проданных товаров за указанный период (объект df).

В учебных целях (чтобы ознакомиться со структурой полученных данных) это набор данных выведен в окно терминала:

```
         Date  Profit
0    2011-01-01    3225
1    2011-01-02    2925
2    2011-01-03   12174
3    2011-01-04    6240
4    2011-01-05    1644
...         ...     ...
1269  2014-06-24   38056
1270  2014-06-25   40933
1271  2014-06-26   35498
1272  2014-06-27   34822
```

```
1273  2014-06-28   46063
[1274 rows x 2 columns]
```

Как видно из полученных результатов, было загружено 1274 строки ежедневных данных. Здесь присутствуют данные за выходные дни, так как для торгового центра выходные дни являются рабочими.

На основе исходного набора данных сформирован график (объект fig), который приведен к формату для отображения в шаблоне Django (объект hist_html).

Затем набор данных приведен к виду, в котором его может принять и обработать библиотека Prophet. Колонке с датами присвоено имя «ds», а колонке с данными - имя «y». Такое изменение исходного набора данных выполнено в следующей инструкции:

```
df.rename(columns={'Date': 'ds', 'Profit': 'y'}, inplace=True)
```

В учебных целях (чтобы ознакомиться со структурой полученных данных) это набор данных выведен в окно терминала:

```
           ds      y
0    2011-01-01   3225
1    2011-01-02   2925
2    2011-01-03  12174
3    2011-01-04   6240
4    2011-01-05   1644
...         ...    ...
1269  2014-06-24  38056
1270  2014-06-25  40933
1271  2014-06-26  35498
1272  2014-06-27  34822
1273  2014-06-28  46063
[1274 rows x 2 columns]
```

Как видно из полученных результатов, был сформирован набор данных, состоящий из 1274-х строк и двух колонок (ds, y). В таком виде данные полностью готовы к тому, что бы передать их в модель для прогнозирования.

На следующем шаге была создана и обучена модель:

```
model = Prophet(yearly_seasonality=True)
model.fit(df)
```

Обученная модель готова для решения задач прогнозирования, это можно сделать буквально в две строки программного кода:

```
predict = 365
future = model.make_future_dataframe(periods=predict, freq='D')
forecast = model.predict(future)
```

Результаты прогноза сохраняются в объекте forecast. Здесь прогноз сделан на год вперед (365 дней). Это сделано для того, чтобы проверить будут ли в прогнозе отражены сезонные колебания.

После этого формируется график результатов прогноза и графики компонент исходного временного ряда. Полученные графики приведены к формату, в котором они могут быть показаны в шаблоне Django (объекты prog_html и comp_html).

После этого создана строка с заголовком страницы (title_str), полученные графики с заголовком страницы через контекстную переменную context передаются в шаблон prog_noz_oil.html (листинг 9.31).

Листинг 9.31. Шаблон prog_noz_oil.html

Примечание.
В демо-версии книги листинги программ удалены.

Это достаточно простой шаблон с контейнером, в котором имеется блок для вывода заголовка страницы, и три блока для вывода графиков. Теперь можно проверить работу данных модулей.

Примечание.
В интерактивной цифровой книге для запуска данного примера нужно на вкладке «Показ примеров» нажать на кнопку «Примеры главы 9» и активировать ссылку «Прогноз продаж товаров».

Если выполнить набор вышеприведенных модулей, то получим график с исходным набором данных (рис. 9.41).



Рис. 9.41. Временной ряд с динамикой продажи товаров

Как видно из данного графика, до 2013 года уровень доходов от продажи товаров с некоторыми колебаниями сохранялся на одном уровне, а после 2013 года наметилась устойчивая тенденция роста доходов. Это связано с тем, что торговый центр открыл несколько новых филиалов своих магазинов.

График с прогнозом доходов торгового центра на ближайший год показан на рис. 9.42.

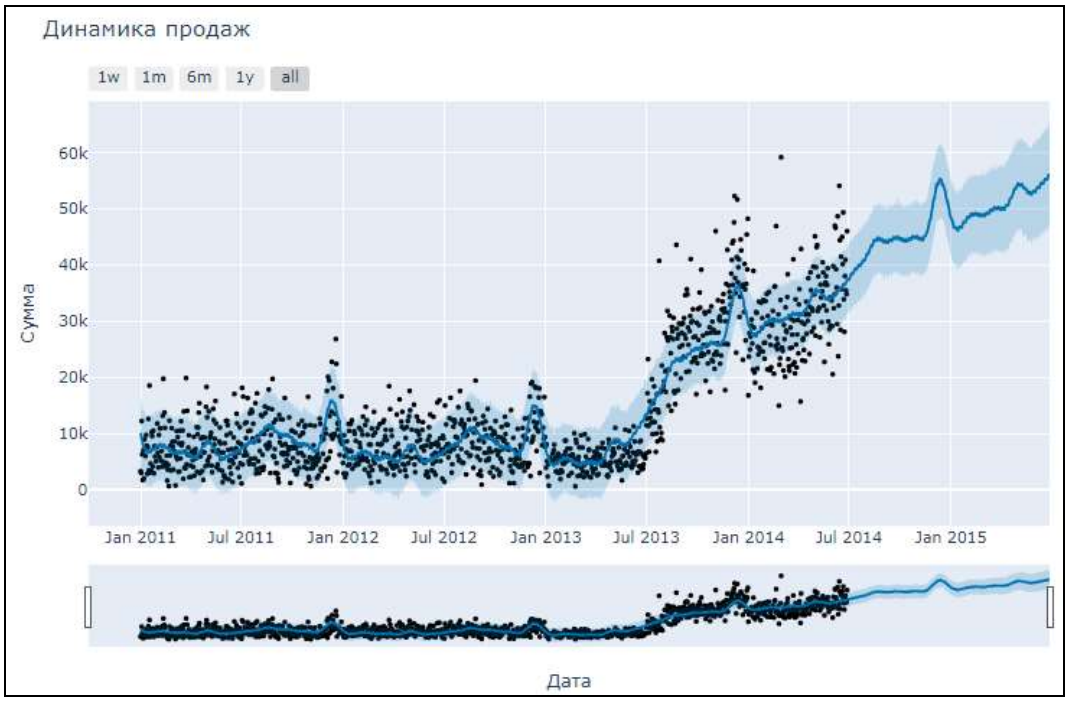


Рис. 9.42. Временной ряд с динамикой изменения доходов торгового центра и данными прогноза

На данном графике показаны следующие значения:

- черными точками показаны исторические данные;
- синей линией показан прогноз по историческим данным, и на будущий период;
- голубым фоном выделена область доверительного интервала, в пределах которого возможны ошибки прогноза.

Из данного графика видно, что линия прогноза адекватно описывает характер изменения доходов, как в историческом, так и в прогнозируемом периоде.

Полученный график является интерактивным. При наведении курсора мыши на черную точку появляется окно с реальными доходами в данный день. При наведении курсора мыши на синюю линию появляется окно со значением прогнозируемого дохода на выбранную дату. В верхней части графика имеется пять кнопок, которые позволяют менять разрешение шкалы дат (увеличивать или уменьшать интервал дат). В нижней части графика имеется слайдер, который позволяет, как менять разрешение шкалы дат, так и перемещать окно с увеличенным диапазоном дат по оси «x».

На рис. 9.43 показан этот же график, в котором расширен диапазон дат, и окно с расширенным диапазоном находится в районе прогноза будущего периода.

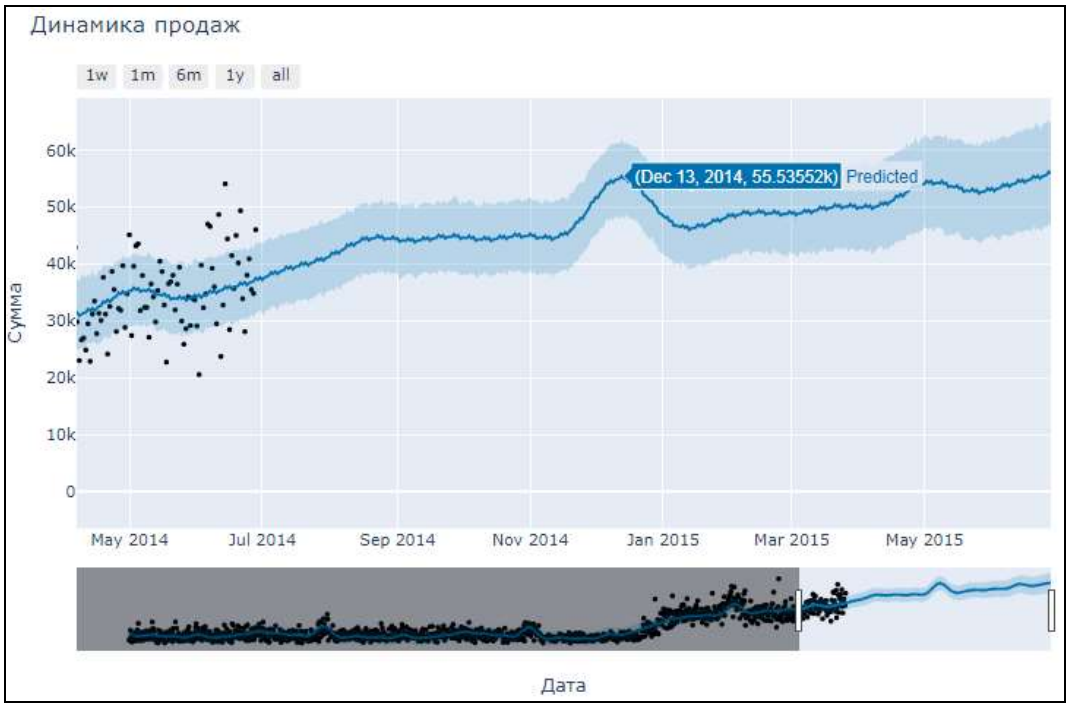


Рис. 9.43. Прогноз изменения доходов в будущем периоде

Из данного графика видно, что в декабре и мае следующего года будет наблюдаться увеличение продаж, значит, на эти периоды необходимо обеспечить увеличение товарной массы.

На рис. 9.44 представлены компоненты исходного временного ряда.

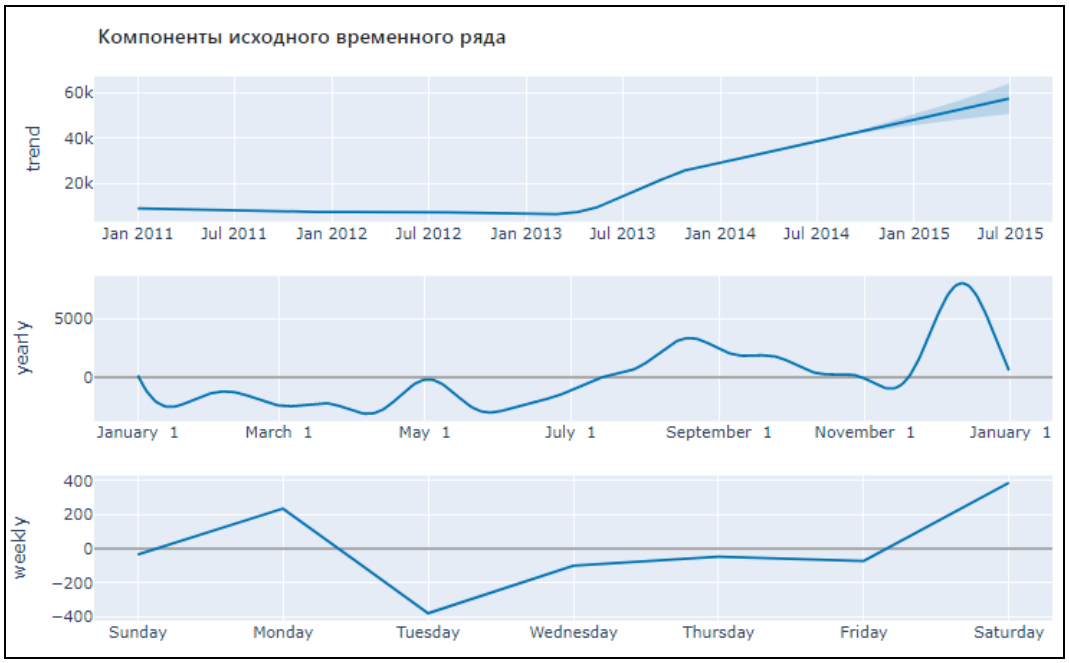


Рис. 9.44. Компоненты исходного временного ряда

Как видно из данного рисунка, тренд временного ряда после 2013 года показывает увеличение доходов торгового центра, при этом с увеличением глубины прогноза рассеивание достоверного результата прогноза увеличивается.

Годовая сезонность демонстрирует три всплеска торговой активности покупателей: в апреле - мае, в августе, и самый большой всплеск в декабре. Первый всплеск связан с активностью покупателей перед майскими праздниками. Второй всплеск связан с началом учебного года, когда большое количество граждан возвращаются из отпусков и активно закупают товары, связанные подготовкой детей к школе. Ну и самый большой всплеск – середина декабря, перед наступлением новогодних праздников. А вот в первых числах января наступает спад торговой активности.

Недельная сезонность говорит о том, что в течение недели колебание продаж незначительные. Наблюдается уменьшение продаж во вторник, а наибольшая активность покупателей в субботний день.

9.2. Краткие итоги

Итак, мы познакомились с замечательной библиотекой Prophet, которая не только упрощает анализ временных рядов и прогнозирование, но и строит соответствующие графики. Правда, ей подвластны не все временные ряды. Например, мы получили слишком большое рассеивание достоверности прогноза на годовых данных экономических индикаторах стран. Для таких временных рядов необходимо выполнять более глубокий анализ, и, скорее всего, делать индивидуальный подбор той или иной статистической модели. Но в целом библиотека Prophet значительно упрощает решение задач прогнозирования.

Послесловие

В этой цифровой интерактивной книге были рассмотрены только некоторые базовые аспекты анализа и прогнозирования данных на примере временных рядов и приемы визуализации данных в веб приложениях на Django. По возможности были максимально упрощены примеры и шаблоны **html** страниц. Основная цель книги – познакомить читателей с некоторыми тонкостями обработки данных в приложениях на **Django**, принципами взаимодействия между представлениями (views) и шаблонами данного фреймворка с элементами библиотеки Pandas и API загрузки данных из открытых источников.

Если после знакомства с приведенными материалами, вы почувствовали, что **Python** и **Django** прекрасно взаимодействую с **Pandas** и различными **API**, и что применение этого набора инструментов упрощает программистам разработку **web** приложений, а аналитикам решение практических задач, то основная цель данного издания достигнута.

В данном издании охвачены далеко не все тонкости разработки **web** приложений с использованием Pandas и API. В частности описаны только базовые методы обработки наборов данных DataFrame, поэтому рекомендуется более изучить специальную литературу по библиотеке Pandas или обратиться к оригинальной документации.

Достаточно актуальными являются вопросы удаленной загрузки данных из открытых источников через API. Каждый доступный открытый источник данных имеет собственный API, однако все они имеют схожие принципы работы. Если вы разобрались с примерами взаимодействия с API, которые приведены в книге, то не должно возникнуть проблем при работе с другими источниками данных.

Если вы аналитик или программист, который работает с данными заказчика, хранимыми в корпоративной СУБД, то Django позволит достаточно просто извлечь информацию из любых баз данных, а Pandas трансформировать ее в формат, удобный для анализа и прогнозирования.

Список всех библиотек с указанием версий, которые использованы при написании программнеого обеспечения данной книги, приведен в приложении 7.

А в завершении остается пожелать удачного применения в своей практической деятельности тех навыков и знаний, которые Вы приобрели при изучении данного материала.

д.т.н., профессор Анатолий Постолит

Список источников и литературы

1. A Guide to Time Series Forecasting with ARIMA in Python 3.
   https://www.digitalocean.com/community/tutorials/a-guide-to-time-series-forecasting-with-arima-in-python-3
2. A Python-based analysis on crude oil price data extracted from Yahoo Finance.
   https://medium.com/@dkoulialias/a-python-based-analysis-on-crude-oil-price-data-extracted-from-yahoo-finance-d6287d011a3c
3. Converting matplotlib png to base64 for viewing in html template.
   https://www.iditect.com/faq/python/converting-matplotlib-png-to-base64-for-viewing-in-html-template.html
4. display django-pandas dataframe in a django template.
   https://stackoverflow.com/questions/39003732/display-django-pandas-dataframe-in-a-django-template
5. Financial charts and visuals with Plotly in Python.
   https://www.insightbig.com/post/financial-charts-and-visuals-with-plotly-in-python
6. Forecasting Stock Prices Using ARIMA Model. https://blog.quantinsti.com/forecasting-stock-returns-using-arima-model/
7. FXCMPY Python Wrapper.
   https://fxcm-api.readthedocs.io/en/latest/fxcmpy.html
8. How to create plotly charts from pandas dataframe with one line of code.
   https://coderzcolumn.com/tutorials/data-science/cufflinks-how-to-create-plotly-charts-from-pandas-dataframe-with-one-line-of-code
9. Quick Intro: auto_arima from pmdarima package.
   https://stevenkyle2013.medium.com/quick-intro-auto-arima-from-pmdarima-package-e7aab5e8dfb8

   Matplotlib into Django Template.
10. https://mdhvkothari.medium.com/matplotlib-into-django-template-5def2e159997
11. Multiple Time Series Forecasting with Prophet in Python.
    https://forecastegy.com/posts/multiple-time-series-forecasting-with-prophet-in-python/
12. Pandas. https://ru.linux-console.net/?p=25268
13. Pandas — конспект знаний для алготрейдера.
    https://xn--80agadetfnxfwx.xn--p1ai/python/pandas-konspekt-znaniy-dlya-algotreydera/?ysclid=ma0vwffo8z385792118

    Pandas Plotting Backend in Python.

14. https://plotly.com/python/pandas-backend/
15. Plot_pacf, plot_acf, autocorrelation_plot and lag_plot.
https://community.plotly.com/t/plot-pacf-plot-acf-autocorrelation-plot-and-lag-plot/24108
16. Plotly for Data Visualization in Python.
https://www.geeksforgeeks.org/using-plotly-for-interactive-data-visualization-in-python/
17. Prophet.
https://filippomb.github.io/python-time-series-handbook/notebooks/09/prophet.html
18. Website Traffic Forecasting.
https://medium.com/@hazallgultekin/website-traffic-forecasting-a5ae06b2c78b
19. Write a Pandas DataFrame to a JSON File.
https://datascienceparichay.com/article/write-pandas-dataframe-to-json-file/
20. Time Series Analysis and Forecasting using Statsmodels.
https://glingden.github.io/datascience_html/Forecasting_with_AR-MA-ARMA-ARIMA-SARIMA-VAR-VARMAX.html
21. Анализ акций Tesla с помощью Python.
https://mlgu.ru/432/?ysclid=mamqa9vwi1230989987&utm_source=yandex&utm_medium=organic
22. Анализ временных рядов с помощью Python.
https://www.pvsm.ru/python/253543?ysclid=majei0pbws310999987
23. Визуализация данных в Python с помощью Plotly.
https://blog.skillfactory.ru/vizualizatsiya-dannyh-v-python-s-pomoschyu-plotly/?ysclid=mamhyvj93k295307259
24. Временные ряды.
https://www.dmitrymakarov.ru/intro/time-series/?ysclid=ma2aaf356s436422243
25. Временные ряды. Прогнозирование и модели. Библиотеки Python.
https://ivan-shamaev.ru/time-series-analysis-forecasting-and-models-python-libraries/?ysclid=m8lvgfj49l456779869
26. Главное руководство по редиректам Django. https://www.codeflow.site/ru/article/django-redirects.
27. Основы анализа данных с Pandas.
https://proglib.io/p/samouchitel-po-pandas-dlya-nachinayushchih-chast-25-osnovy-analiza-dannyh-s-pandas-2023-07-24?ysclid=m9pskr23ra960191781
28. Получение данных, используя XML.
https://cbr.ru/development/SXML/
29. Постолит А. Python, Django и Bootstrap для начинающих. Санкт-Петербург. «БХВ-Петербург», 2023. с. 623.
30. Постолит А. Мастер веб разработки на Python, Django и Bootstrap. Санкт-Петербург. «БХВ-Петербург», 2025. с. 655.

31. Предсказываем будущее с помощью библиотеки Facebook Prophet.
https://habr.com/ru/companies/ods/articles/323730/
32. Прогнозирование временных рядов в python.
https://pygame.ru/blog/prognozirovanie-vremennih-ryadov-python.php#google_vignette
33. Регрессия и прогнозирование ARIMA в statsmodels.
https://newtechaudit.ru/arima-v-statsmodels/
34. Шпаргалка по визуализации данных в Python с помощью Plotly.
https://habr.com/ru/articles/502958/

Приложение 1. Список кодов стран API Всемирного банка

['AD', 'AE', 'AF', 'AG', 'AI', 'AL', 'AM', 'AO', 'AQ', 'AR', 'AS', 'AT', 'AU', 'AW', 'AX', 'AZ', 'BA', 'BB', 'BD', 'BE', 'BF', 'BG', 'BH', 'BI', 'BJ', 'BL', 'BM', 'BN', 'BO', 'BQ', 'BR', 'BS', 'BT', 'BV', 'BW', 'BY', 'BZ', 'CA', 'CC', 'CD', 'CF', 'CG', 'CH', 'CI', 'CK', 'CL', 'CM', 'CN', 'CO', 'CR', 'CU', 'CV', 'CW', 'CX', 'CY', 'CZ', 'DE', 'DJ', 'DK', 'DM', 'DO', 'DZ', 'EC', 'EE', 'EG', 'EH', 'ER', 'ES', 'ET', 'FI', 'FJ', 'FK', 'FM', 'FO', 'FR', 'GA', 'GB', 'GD', 'GE', 'GF', 'GG', 'GH', 'GI', 'GL', 'GM', 'GN', 'GP', 'GQ', 'GR', 'GS', 'GT', 'GU', 'GW', 'GY', 'HK', 'HM', 'HN', 'HR', 'HT', 'HU', 'ID', 'IE', 'IL', 'IM', 'IN', 'IO', 'IQ', 'IR', 'IS', 'IT', 'JE', 'JM', 'JO', 'JP', 'KE', 'KG', 'KH', 'KI', 'KM', 'KN', 'KP', 'KR', 'KW', 'KY', 'KZ', 'LA', 'LB', 'LC', 'LI', 'LK', 'LR', 'LS', 'LT', 'LU', 'LV', 'LY', 'MA', 'MC', 'MD', 'ME', 'MF', 'MG', 'MH', 'MK', 'ML', 'MM', 'MN', 'MO', 'MP', 'MQ', 'MR', 'MS', 'MT', 'MU', 'MV', 'MW', 'MX', 'MY', 'MZ', 'NA', 'NC', 'NE', 'NF', 'NG', 'NI', 'NL', 'NO', 'NP', 'NR', 'NU', 'NZ', 'OM', 'PA', 'PE', 'PF', 'PG', 'PH', 'PK', 'PL', 'PM', 'PN', 'PR', 'PS', 'PT', 'PW', 'PY', 'QA', 'RE', 'RO', 'RS', 'RU', 'RW', 'SA', 'SB', 'SC', 'SD', 'SE', 'SG', 'SH', 'SI', 'SJ', 'SK', 'SL', 'SM', 'SN', 'SO', 'SR', 'SS', 'ST', 'SV', 'SX', 'SY', 'SZ', 'TC', 'TD', 'TF', 'TG', 'TH', 'TJ', 'TK', 'TL', 'TM', 'TN', 'TO', 'TR', 'TT', 'TV', 'TW', 'TZ', 'UA', 'UG', 'UM', 'US', 'UY', 'UZ', 'VA', 'VC', 'VE', 'VG', 'VI', 'VN', 'VU', 'WF', 'WS', 'YE', 'YT', 'ZA', 'ZM', 'ZW', 'ABW', 'AFG', 'AGO', 'AIA', 'ALA', 'ALB', 'AND', 'ARE', 'ARG', 'ARM', 'ASM', 'ATA', 'ATF', 'ATG', 'AUS', 'AUT', 'AZE', 'BDI', 'BEL', 'BEN', 'BES', 'BFA', 'BGD', 'BGR', 'BHR', 'BHS', 'BIH', 'BLM', 'BLR', 'BLZ', 'BMU', 'BOL', 'BRA', 'BRB', 'BRN', 'BTN', 'BVT', 'BWA', 'CAF', 'CAN', 'CCK', 'CHE', 'CHL', 'CHN', 'CIV', 'CMR', 'COD', 'COG', 'COK', 'COL', 'COM', 'CPV', 'CRI', 'CUB', 'CUW', 'CXR', 'CYM', 'CYP', 'CZE', 'DEU', 'DJI', 'DMA', 'DNK', 'DOM', 'DZA', 'ECU', 'EGY', 'ERI', 'ESH', 'ESP', 'EST', 'ETH', 'FIN', 'FJI', 'FLK', 'FRA', 'FRO', 'FSM', 'GAB', 'GBR', 'GEO', 'GGY', 'GHA', 'GIB', 'GIN', 'GLP', 'GMB', 'GNB', 'GNQ', 'GRC', 'GRD', 'GRL', 'GTM', 'GUF', 'GUM', 'GUY', 'HKG', 'HMD', 'HND', 'HRV', 'HTI', 'HUN', 'IDN', 'IMN', 'IND', 'IOT', 'IRL', 'IRN', 'IRQ', 'ISL', 'ISR', 'ITA', 'JAM', 'JEY', 'JOR', 'JPN', 'KAZ', 'KEN', 'KGZ', 'KHM', 'KIR', 'KNA', 'KOR', 'KWT', 'LAO', 'LBN', 'LBR', 'LBY', 'LCA', 'LIE', 'LKA', 'LSO', 'LTU', 'LUX', 'LVA', 'MAC', 'MAF', 'MAR', 'MCO', 'MDA', 'MDG', 'MDV', 'MEX', 'MHL', 'MKD', 'MLI', 'MLT', 'MMR', 'MNE', 'MNG', 'MNP', 'MOZ', 'MRT', 'MSR', 'MTQ', 'MUS', 'MWI', 'MYS', 'MYT', 'NAM', 'NCL', 'NER', 'NFK', 'NGA', 'NIC', 'NIU', 'NLD', 'NOR', 'NPL', 'NRU', 'NZL', 'OMN', 'PAK', 'PAN', 'PCN', 'PER', 'PHL', 'PLW', 'PNG', 'POL', 'PRI', 'PRK', 'PRT', 'PRY', 'PSE', 'PYF', 'QAT', 'REU', 'ROU', 'RUS', 'RWA', 'SAU', 'SDN', 'SEN', 'SGP', 'SGS', 'SHN', 'SJM', 'SLB', 'SLE', 'SLV', 'SMR', 'SOM', 'SPM', 'SRB', 'SSD', 'STP', 'SUR', 'SVK', 'SVN', 'SWE', 'SWZ', 'SXM', 'SYC', 'SYR', 'TCA', 'TCD', 'TGO', 'THA', 'TJK', 'TKL', 'TKM', 'TLS', 'TON', 'TTO', 'TUN', 'TUR', 'TUV', 'TWN', 'TZA', 'UGA', 'UKR', 'UMI', 'URY', 'USA', 'UZB', 'VAT', 'VCT', 'VEN', 'VGB', 'VIR', 'VNM', 'VUT', 'WLF', 'WSM', 'YEM', 'ZAF', 'ZMB', 'ZWE', 'all', 'ALL', 'All']

Приложение 2. Некоторые индикаторы финансовых показателей API Всемирного банка

№	Словарь	Расшифровка финансового показателя
	'NY.GDP.MKTP.KD.ZG': 'GDP growth'	Рост ВВП
	'FP.CPI.TOTL.ZG': 'Inflation'	Инфляция
	'FR.INR.RINR': 'Real interest rate'	Реальная процентная ставка
	'NE.EXP.GNFS.ZS': 'Exports'	Экспорт товаров и услуг (% от ВВП)
	'NE.IMP.GNFS.ZS': 'Imports'	Импорт товаров и услуг (% от ВВП)
	'BN.CAB.XOKA.GD.ZS': 'Current account balance'	Сальдо счета текущих операций (% от ВВП)
	'GC.DOD.TOTL.GD.ZS': 'Government debt'	Государственный долг (% от ВВП)
	'SL.UEM.TOTL.ZS': 'Unemployment rate'	Уровень безработицы (% от общей численности рабочей силы)
	'NY.GNP.PCAP.CD': 'GNI per capita'	ВНД на душу населения (в текущих долларах США)
	'NY.GDP.PCAP.KD.ZG': 'GDP per capita growth'	Рост ВВП на душу населения (без изменений в 2010 году в долларах США)
	'NE.RSB.GNFS.ZS': 'Reserves in months of imports'	Запасы в месяцах импорта
	'NY.GDP.DEFL.KD.ZG': 'GDP deflator'	Дефлятор ВВП (постоянный показатель в долларах США за 2010 год)
	'NY.GDP.PCAP.KD': 'GDP per capita (constant 2015 US$)'	ВВП на душу населения (постоянный показатель за 2015 год в долларах США)
	'NY.GDP.PCAP.PP.CD': 'GDP per capita, PPP (current international $)'	ВВП на душу населения по ППС (в текущих международных долларах)
	'NY.GDP.PCAP.PP.KD': 'GDP per capita, PPP (constant 2017 international $)'	ВВП на душу населения по ППС (постоянный международный показатель за 2017 год в долларах США)
	'NY.GDP.PCAP.CN': 'GDP per capita (current LCU)'	ВВП на душу населения (текущий LCU)
	'NY.GDP.PCAP.KN': 'GDP per capita (constant LCU)'	ВВП на душу населения (постоянный LCU)
	'NY.GDP.PCAP.CD': 'GDP per capita (current US$)'	ВВП на душу населения (в текущих долларах США)
	'NY.GDP.PCAP.KD': 'GDP per capita (constant 2010 US$)'	ВВП на душу населения (в неизменном виде в долларах США за 2010 год)
	'NY.GDP.PCAP.KD.ZG': 'GDP per capita growth (annual %)'	Рост ВВП на душу населения (% в годовом исчислении)
	'NY.GDP.PCAP.KN.ZG': 'GDP per capita growth (constant LCU)'	# Рост ВВП на душу населения (постоянный LCU)

Приложение 3. Список идентификаторов (тикеров) компаний API Московской биржи

tickers = ['ABIO', 'ABRD', 'ACKO', 'AFKS', 'AFLT', 'AGRO', 'AKRN', 'ALRS', 'AMEZ', 'APTK', 'AQUA', 'ARSA', 'ASSB', 'ASTR', 'AVAN', 'BANE', 'BANEP', 'BELU', 'BISVP', 'BLNG', 'BRZL', 'BSPB', 'BSPBP', 'CARM', 'CBOM', 'CHGZ', 'CHKZ', 'CHMF', 'CHMK', 'CIAN', 'CNTL', 'CNTLP', 'DIOD', 'DSKY', 'DVEC', 'DZRD', 'DZRDP', 'EELT', 'ELFV', 'ENPG', 'ETLN', 'EUTR', 'FEES', 'FESH', 'FIVE', 'FIXP', 'FLOT', 'GAZA', 'GAZAP', 'GAZC', 'GAZP', 'GAZS', 'GAZT', 'GCHE', 'GECO', 'GEMA', 'GEMC', 'GLTR', 'GMKN', 'GTRK', 'HHRU', 'HIMCP', 'HMSG', 'HNFG', 'HYDR', 'IGST', 'IGSTP', 'INGR', 'IRAO', 'IRKT', 'JNOS', 'JNOSP', 'KAZT', 'KAZTP', 'KBSB', 'KCHE', 'KCHEP', 'KGKC', 'KGKCP', 'KLSB', 'KMAZ', 'KMEZ', 'KOGK', 'KRKN', 'KRKNP', 'KRKOP', 'KROT', 'KROTP', 'KRSB', 'KRSBP', 'KTSB', 'KTSBP', 'KUBE', 'KUZB', 'KZOS', 'KZOSP', 'LENT', 'LIFE', 'LKOH', 'LNZL', 'LNZLP', 'LSNG', 'LSNGP', 'LSRG', 'LVHK', 'MAGE', 'MAGEP', 'MAGN', 'MDMG', 'MFGS', 'MFGSP', 'MGNT', 'MGNZ', 'MGTS', 'MGTSP', 'MISB', 'MISBP', 'MOEX', 'MRKC', 'MRKK', 'MRKP', 'MRKS', 'MRKU', 'MRKV', 'MRKY', 'MRKZ', 'MRSB', 'MSNG', 'MSRS', 'MSTT', 'MTLR', 'MTLRP', 'MTSS', 'MVID', 'NAUK', 'NFAZ', 'NKHP', 'NKNC', 'NKNCP', 'NKSH', 'NLMK', 'NMTP', 'NNSB', 'NNSBP', 'NSVZ', 'NVTK', 'OGKB', 'OKEY', 'OMZZP', 'OZON', 'PAZA', 'PHOR', 'PIKK', 'PLZL', 'PMSB', 'PMSBP', 'POLY', 'POSI', 'PRFN', 'PRMB', 'QIWI', 'RASP', 'RBCM', 'RDRB', 'RENI', 'RGSS', 'RKKE', 'RNFT', 'ROLO', 'ROSB', 'ROSN', 'ROST', 'RTGZ', 'RTKM', 'RTKMP', 'RTSB', 'RTSBP', 'RUAL', 'RUSI', 'RZSB', 'SAGO', 'SAGOP', 'SARE', 'SAREP', 'SBER', 'SBERP', 'SELG', 'SFIN', 'SGZH', 'SIBN', 'SLEN', 'SMLT', 'SNGS', 'SNGSP', 'SOFL', 'SPBE', 'STSB', 'STSBP', 'SVAV', 'SVCB', 'SVET', 'TASB', 'TASBP', 'TATN', 'TATNP', 'TCSG', 'TGKA', 'TGKB', 'TGKBP', 'TGKN', 'TNSE', 'TORS', 'TORSP', 'TRMK', 'TRNFP', 'TTLK', 'TUZA', 'UGLD', 'UKUZ', 'UNAC', 'UNKL', 'UPRO', 'URKZ', 'USBN', 'UTAR', 'VEON-RX', 'VGSB', 'VGSBP', 'VJGZ', 'VJGZP', 'VKCO', 'VLHZ', 'VRSB', 'VRSBP', 'VSMO', 'VSYD', 'VSYDP', 'VTBR', 'WTCM', 'WTCMP', 'WUSH', 'YAKG', 'YKEN', 'YKENP', 'YNDX', 'YRSB', 'YRSBP', 'ZILL', 'ZVEZ']

Приложение 4. Код программного модуля clear_data

```
import pandas as pd


def clear_data(d1, d2, kod_val):
    # Формируем URL адрес запроса в ЦБ
    url_cb = 'https://www.cbr.ru/scripts/XML_dynamic.asp?'
    date_req1 = 'date_req1='
    date_req2 = '&date_req2='
    VAL_NM_RQ = '&VAL_NM_RQ='
    url = url_cb + date_req1 + d1 + date_req2 + d2 + VAL_NM_RQ + kod_val
    # Загрузка данных из БД ЦБ
    dat = pd.read_xml(url)

    # Заменяем запятые на точки в столбце с валютой
    dat['Value'] = dat['Value'].str.replace(',', '.')
    # Преобразуем строки в числа в столбце с валютой
    dat['Value'] = dat['Value'].astype(float)

    # В дате заменяем точки на "-"
    dat['Date'] = dat['Date'].str.replace('.', '-')
    # Заменяем строковый столбец с датой на тип datetime
    dat['Date'] = pd.to_datetime(dat['Date'], format='%d-%m-%Y', errors='coerce')
    return dat


if __name__ == "__main__":
    # Начало периода
    d1_str = '01/01/2020'
    # Конец периода
    d2_str = '21/04/2025'
    # Код валюты
    k_val = 'R01235'
    df = clear_data(d1_str, d2_str, k_val)
    print(df)
```

Приложение 5. Код программного модуля clear_data_metall

```
import pandas as pd
# Для работы с датами
from datetime import date
from dateutil.relativedelta import relativedelta


def clear_data(d_start, d_stop):
    # Формируем URL адрес запроса в ЦБ
    url_cb = 'http://www.cbr.ru/scripts/xml_metall.asp?'
    date_req1 = 'date_req1='
    date_req2 = '&date_req2='
    url = url_cb + date_req1 + d_start + date_req2 + d_stop
    # Загрузка данных из БД ЦБ
    dat = pd.read_xml(url)

    # Заменяем запятые на точки в столбце с ценой продажи
    dat['Sell'] = dat['Sell'].str.replace(',', '.')
    # Преобразуем строки в числа в столбце с ценой продажи
    dat['Sell'] = dat['Sell'].astype(float)

    # Заменяем запятые на точки в столбце с ценой продажи
    dat['Buy'] = dat['Buy'].str.replace(',', '.')
    # Преобразуем строки в числа в столбце с ценой покупки
    dat['Buy'] = dat['Buy'].astype(float)

    # В дате заменяем точки на "-"
    dat['Date'] = dat['Date'].str.replace('.', '-')
    # Заменяем строковый столбец с датой на тип datetime
    dat['Date'] = pd.to_datetime(dat['Date'], format='%d-%m-%Y', errors='coerce')
    return dat


if __name__ == "__main__":
    d1 = date.today() - relativedelta(years=5)
    d2 = date.today()
    d1_str = d1.strftime("%d-%m-%Y")
    d2_str = d2.strftime("%d-%m-%Y")
    df = clear_data(d1_str, d2_str)
    print(df)
```

Приложение 6. Код программного модуля fn_wb_data

```
########################################
# Получить данные Всемирного банка #
########################################
import wbdata
from datetime import datetime


def get_data_wb(indicator, countries, s_date, e_date):
    # Интервал дат - американский формат из строк
    start_d = datetime.strptime(s_date, '%d-%m-%Y')
    end_d = datetime.strptime(e_date, '%d-%m-%Y')
    # Извлечение данных из API Всемирного банка
    dfr = wbdata.get_dataframe(indicator,
                               country=countries,
                               date=(start_d, end_d),
                               parse_dates=True)
    return dfr


if __name__ == "__main__":
    # Задан показатель (ВВП в текущих долларах США)
    dic_indic = {'NY.GDP.MKTP.CD': 'GDP (current US$)'}
    # Страны (коды ISO для стран)
    list_country = ['CN', 'US', 'RUS', 'GB']
    # Интервал дат - европейский формат - строки
    d1_str = '01-01-2019'
    d2_str = '17-04-2025'
    df = get_data_wb(dic_indic, list_country,  d1_str, d2_str)
    # Напечатать данные
    # print(df.head())
    print(df)
```

Приложение 7. Список используемых библиотек с указанием версий

annotated-types==0.7.0
appdirs==1.4.4
asgiref==3.8.1
backoff==2.2.1
beautifulsoup4==4.13.4
blinker==1.9.0
cachetools==5.5.2
certifi==2025.4.26
cffi==1.17.1
channels==4.2.2
charset-normalizer==3.4.2
click==8.1.8
click-option-group==0.5.7
cmdstanpy==1.2.5
contourpy==1.3.2
curl_cffi==0.11.1
cycler==0.12.1
Cython==3.1.1
dash==2.18.2
dash-bootstrap-components==1.7.1
dash-core-components==2.0.0
dash-html-components==2.0.0
dash-table==5.0.0
dateparser==1.2.1
decorator==5.2.1
Deprecated==1.2.18
Django==5.2.1
django-cleanup==9.0.0
django-plotly-dash==2.4.6
dotty-dict==1.3.1
dpd_components==0.1.0
et_xmlfile==2.0.0
Flask==3.0.3
fonttools==4.58.1
frozendict==2.4.6
gitdb==4.0.12
GitPython==3.1.44
holidays==0.75
idna==3.10
imfp==1.2.0
importlib_metadata==8.7.0
importlib_resources==6.5.2
itsdangerous==2.2.0

Jinja2==3.1.6
joblib==1.5.1
kiwisolver==1.4.8
lxml==5.4.0
markdown-it-py==3.0.0
MarkupSafe==3.0.2
matplotlib==3.10.3
mdurl==0.1.2
multitasking==0.0.11
narwhals==1.41.0
nest-asyncio==1.6.0
numpy==1.24.4
openpyxl==3.1.5
packaging==25.0
pandas==2.2.3
pandas-datareader==0.10.0
patsy==1.0.1
peewee==3.18.1
pillow==11.2.1
platformdirs==4.3.8
plotly==6.1.2
pmdarima==2.0.4
prophet==1.1.7
protobuf==6.31.1
pycparser==2.22
pydantic==2.11.5
pydantic_core==2.33.2
Pygments==2.19.1
pyparsing==3.2.3
python-dateutil==2.9.0.post0
python-gitlab==6.0.0
python-semantic-release==10.1.0
pytz==2025.2
regex==2024.11.6
requests==2.32.3
requests-toolbelt==1.0.0
retrying==1.3.4
rich==14.0.0
scikit-learn==1.6.1
scipy==1.15.3
shellingham==1.5.4
shelved-cache==0.3.1
six==1.17.0
smmap==5.0.2
soupsieve==2.7
sqlparse==0.5.3
stanio==0.5.1
statsmodels==0.14.4
tabulate==0.9.0

threadpoolctl==3.6.0
tomlkit==0.13.3
tqdm==4.67.1
type_enforced==1.10.2
typing-inspection==0.4.1
typing_extensions==4.14.0
tzdata==2025.2
tzlocal==5.3.1
urllib3==2.4.0
wbdata==1.0.0
websockets==15.0.1
Werkzeug==3.0.6
wrapt==1.17.2
yfinance==0.2.61
zipp==3.22.0