**МИНОБРНАУКИ РОССИИ**

федеральное государственное бюджетное образовательное

учреждение высшего образования

**«Национальный исследовательский университет «МЭИ»**

ИНЖЕНЕРНО-ЭКОНОМИЧЕСКИЙ ИНСТИТУТ

Кафедра «Безопасности и информационных технологий»

**ОТЧЕТ ПО КУРСОВОЙ РАБОТЕ**

по дисциплине «Интеллектуальные информационные системы»

Тема: «Проектирование и разработка интеллектуальной информационной системы для прогнозирования курсов валют»

Студент группы ИЭ-65-22 Тихонюк Д.А.

(№ группы, Ф.И.О.)

Проверили:

д.т.н., профессор Бехтин Ю.С.

ассистент кафедры БИТ Баланев К.С.

|  |  |  |
| --- | --- | --- |
| Сдана на проверку |  |  |
| Возвращена на доработку |  |  |
| Допущена к защите |  |  |
| Оценка |  |  |
|  |

Москва 2025

# АННОТАЦИЯ

Курсовая работа посвящена проектированию и разработке интеллектуальной информационной системы для прогнозирования курса валют с использованием методов машинного обучения. Основной целью работы является создание комбинированной регрессионной модели, объединяющей линейную регрессию, случайный лес, метод опорных векторов и нейронную сеть, а также ее интеграция в Telegram-бот для предоставления пользователям удобного доступа к прогнозам.

Пояснительная записка включает три главы, приложения и список использованных источников.

В первой главе рассмотрены теоретические основы прогнозирования курсов валют, проведён анализ традиционных статистических методов и современных подходов машинного обучения, обоснован выбор комбинированной модели и источников данных.

Вторая глава посвящена сбору, предобработке и анализу данных, включая обработку выбросов и лаговых признаков, а также разработке и оценке качества моделей с использованием различных метрик.

В третьей главе описана реализация Telegram-бота, его архитектура, функциональные возможности и интерфейс, обеспечивающие пользователям доступ к прогнозам и аналитическим данным.

# СОКРАЩЕНИЯ И ОБОЗНАЧЕНИЯ

В данной работе используется ряд технических сокращений и терминов, упрощающих описание процессов, связанных с анализом данных и построением Telegram-бота.

АКФ – автокорреляционная функция;

ИИ – искусственный интеллект;

ИИС – интеллектуальная информационная система;

КВ – курсы валют;

МЛ – машинное обучение;

НС – нейронная сеть;

РМ – регрессионная модель;

ТБ – Telegram-бот (бот в мессенджере Telegram);

MAE – средняя абсолютная ошибка (Mean Absolute Error);

MAPE – средняя абсолютная процентная ошибка (Mean Absolute Percentage Error);

R² – коэффициент детерминации (R-squared);

RMSE – среднеквадратичная ошибка (Root Mean Squared Error);

SVR – метод опорных векторов для регрессии.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 5](#_Toc198037506)

[ГЛАВА 1. ОБЩИЕ СВЕДЕНИЯ 7](#_Toc198037507)

[1.1. Актуальность прогнозирования курсов валют 7](#_Toc198037508)

[1.2. Существующие методы прогнозирования курсов валют 7](#_Toc198037509)

[1.2.1. Традиционные статистические методы 8](#_Toc198037510)

[1.2.2. Методы на основе машинного обучения 9](#_Toc198037511)

[1.2.3. Сравнение традиционных методов и методов МЛ 11](#_Toc198037512)

[1.3. Выбор подхода для разработки ИИС 12](#_Toc198037513)

[1.4. Источники данных 13](#_Toc198037514)

[1.5. Выводы к главе 1 14](#_Toc198037515)

[ГЛАВА 2. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ…………………………………….16](#_Toc198037516)

[2.1. Проблематика 16](#_Toc198037517)

[2.2. Подбор подходящего набора данных 17](#_Toc198037518)

[2.3 Анализ и подготовка данных 20](#_Toc198037519)

[2.4. Машинное обучение. Оценка качества моделей 33](#_Toc198037520)

[2.5. Выводы к главе 2 40](#_Toc198037521)

[ГЛАВА 3. РАЗРАБОТКА TELEGRAM-БОТА……………………………………….43](#_Toc198037522)

[3.1. Обоснование выбора платформы Telegram 43](#_Toc198037523)

[3.2. Используемые технологии и библиотеки 43](#_Toc198037524)

[3.3. Архитектура приложения 45](#_Toc198037525)

[3.4. Функциональные возможности и интерфейс 48](#_Toc198037526)

[3.5. Выводы к главе 3. 56](#_Toc198037527)

[ЗАКЛЮЧЕНИЕ 58](#_Toc198037528)

[СПИСОК ИСТОЧНИКОВ 60](#_Toc198037529)

[ПРИЛОЖЕНИЯ 63](#_Toc198037530)

# ВВЕДЕНИЕ

В современном мире финансовые рынки играют ключевую роль в глобальной экономике, а курсы валют являются важнейшими индикаторами экономической стабильности, инвестиционного климата и финансовой устойчивости. Прогнозирование курсов валют представляет собой сложную задачу, обусловленную их высокой волатильностью и зависимостью от множества факторов, таких как макроэкономические показатели, политические события, цены на сырьевые товары (нефть, драгоценные металлы) и рыночные тренды. В условиях нестабильности валютного рынка разработка интеллектуальных информационных систем (ИИС), способных анализировать большие объемы данных и предоставлять точные прогнозы, является актуальной задачей, имеющей как теоретическую, так и практическую значимость. Такие системы могут стать ценным инструментом для инвесторов, финансовых аналитиков и обычных пользователей, стремящихся принимать обоснованные решения в финансовой сфере.

Данная работа посвящена проектированию и разработке интеллектуальной информационной системы для прогнозирования курсов валют с использованием методов машинного обучения. Выбор темы обусловлен необходимостью создания доступных, эффективных и автоматизированных инструментов для анализа валютных рынков, которые способны учитывать сложные зависимости в данных и предоставлять прогнозы в удобной форме. Актуальность исследования подтверждается возрастающим интересом к применению методов искусственного интеллекта и машинного обучения в финансовой аналитике, а также потребностью в оперативных и точных прогнозах для минимизации финансовых рисков.

Связь работы с предшествующими исследованиями проявляется в использовании современных подходов к анализу временных рядов и машинному обучению. В последние годы активно развиваются регрессионные модели, такие как линейная регрессия, ARIMA и нейронные сети (например, LSTM), которые демонстрируют высокую эффективность при прогнозировании финансовых данных. В работе применяются эти методы с учетом особенностей валютного рынка, включая влияние внешних факторов, таких как экономические индикаторы и рыночные события. Особое внимание уделяется предварительной обработке данных, включая анализ и устранение выбросов, что позволяет повысить точность прогнозов.

**Целью работы является** разработка интеллектуальной информационной системы для прогнозирования курсов валют, основанную на методах машинного обучения, и интегрировать её в Telegram-бот для предоставления пользователям актуальных прогнозов.

**Задачи работы:**

1. Изучить предметную область и существующие методы прогнозирования курсов валют.
2. Собрать и предобработать исторические данные о курсах валют, включая анализ и обработку выбросов.
3. Разработать и сравнить две регрессионные модели машинного обучения для прогнозирования курсов валют.
4. Оценить качество моделей с использованием метрик MAE и RMSE и объединить их в единую комбинированную регрессионную модель.
5. Интегрировать выбранную модель в Telegram-бот для предоставления прогнозов пользователям.
6. Протестировать и оценить работу интеллектуальной информационной системы.

Работа состоит из трех глав, введения, заключения, списка литературы и приложений. В первой главе проводится анализ существующих методов прогнозирования курсов валют, обосновывается актуальность темы и выбор подходов. Вторая глава посвящена сбору, предобработке и анализу данных, включая работу с выбросами и подготовку данных для моделирования. Третья глава описывает разработку и сравнение моделей машинного обучения, выбор наилучшей модели, её интеграцию в Telegram-бот и тестирование системы.

# ОБЩИЕ СВЕДЕНИЯ

## 1.1. Актуальность прогнозирования курсов валют

В условиях глобализации курсы валют являются ключевыми индикаторами экономической стабильности, инвестиционного климата и финансовой устойчивости. Они отражают состояние национальной экономики, уровень доверия инвесторов и политическую ситуацию. Прогнозирование курсов валют позволяет инвесторам, бизнесу и государственным структурам принимать обоснованные решения, минимизируя финансовые риски, связанные с валютными колебаниями [9].

Особенно актуальна эта задача в условиях высокой волатильности валютных рынков, характерной для многих современных экономик. Например, в нашей стране, в России, курс рубля демонстрирует значительные колебания, обусловленные зависимостью от цен на энергоносители, такие как нефть и газ, а также геополитическими факторами. Эти обстоятельства требуют разработки интеллектуальных информационных систем (ИИС), способных учитывать множество переменных и адаптироваться к быстро меняющимся условиям. Такие системы могут стать важным инструментом для финансовых аналитиков и частных лиц, стремящихся оптимизировать свои инвестиционные стратегии [8].

## 1.2. Существующие методы прогнозирования курсов валют

Прогнозирование курсов валют представляет собой сложную задачу, требующую учета множества факторов, таких как экономические показатели, политическая ситуация и рыночные настроения.

Существующие методы прогнозирования можно разделить на две основные категории: традиционные статистические методы и современные подходы, основанные на машинном обучении (МЛ). Традиционные методы, включая анализ временных рядов и регрессионные модели, опираются на исторические данные. Современные методы, такие как нейронные сети и алгоритмы ансамблевого обучения, способны учитывать нелинейные зависимости и взаимодействия между различными факторами. Выбор подхода зависит от доступных данных и целей анализа.

### 1.2.1. Традиционные статистические методы

Традиционные методы прогнозирования основаны на анализе временных рядов, который представляет собой последовательность данных, собранных через равные промежутки времени. Такие методы до сих пор широко применяются в экономике, финансах и других областях для предсказания будущих значений на основе исторических данных. В основе таких подходов лежит предположение о том, что прошлые значения временного ряда содержат информацию, которая может быть использована для прогнозирования будущих изменений [10].

Основными особенностями традиционных методов являются их математическая строгость, простота реализации и интерпретируемость. Такие методы прогнозирования основаны на анализе временных рядов. Однако они часто ограничены в учете сложных нелинейных зависимостей и внешних факторов, таких как геополитические события или изменения в кредитно-денежной политики центральных банков. Традиционные методы включают следующие подходы:

* **ARIMA (Autoregressive Integrated Moving Average)** — Модель ARIMA совмещает в себе три ключевых компонента: авторегрессию (AR), интегрирование (I) и скользящее среднее (MA). Авторегрессия предполагает, что текущие значения зависят от предыдущих наблюдений; интегрирование используется для стабилизации ряда путем его дифференцирования; скользящее среднее учитывает случайные шумы в данных. ARIMA широко используется для прогнозирования финансовых данных благодаря своей простоте и интерпретируемости. Однако она предполагает линейные зависимости и может быть менее эффективной при наличии сложных нелинейных факторов.
* **Экспоненциальное сглаживание** — метод, который присваивает больший вес более недавним наблюдениям, что делает его подходящим для данных с выраженными трендами или сезонностью. Этот метод прост в реализации, но ограничен в обработке большого количества внешних факторов.

Хотя традиционные методы успешно применяются для анализа стабильных временных рядов, их возможности ограничены в условиях высокой волатильности и сложных зависимостей. Например, курс рубля в России сильно зависит от цен на энергоносители, таких как нефть и газ, а также от геополитических факторов, которые традиционные модели не могут учитывать в полной мере.

Для решения этих проблем современные исследования все чаще обращаются к машинному обучению и гибридным моделям, которые сочетают традиционные методы с алгоритмами искусственного интеллекта. Такие подходы позволяют учитывать множество переменных и адаптироваться к быстро меняющимся условиям.

### 1.2.2. Методы на основе машинного обучения

Методы машинного обучения представляют собой совокупность алгоритмов и подходов, позволяющих анализировать данные и делать на их основе прогнозы. Они находят широкое применение в различных областях, включая финансовую аналитику. Одним из основных преимуществ машинного обучения является способность выявлять сложные и нелинейные зависимости в данных, что особенно актуально для прогнозирования курсов валют. Обработка данных является ключевым этапом подготовки к машинному обучению и включает такие методы, как очистка данных, преобразование, масштабирование и выбор признаков. Эти процессы значительно улучшают качество анализа и повышают точность прогнозов [11].

МЛ методы предлагают более гибкий подход к прогнозированию, позволяя обрабатывать большие объемы данных и выявлять сложные зависимости. Среди наиболее распространенных методов для прогнозирования курсов валют выделяются [13]:

* **Линейная регрессия** — используется для моделирования линейных зависимостей между переменными. Этот метод прост и интерпретируем, но может быть недостаточно точным для нелинейных данных, характерных для валютных курсов.
* Полиномиальная регрессия — расширение линейной регрессии, которое учитывает нелинейные зависимости за счет добавления полиномиальных признаков. Этот метод позволяет лучше описать сложные взаимосвязи в данных, однако он чувствителен к выбросам и может привести к переобучению, если степень полинома выбрана некорректно
* **SVM (Support Vector Machines)** — адаптированы для задач регрессии и показывают хорошие результаты при наличии нелинейных зависимостей. SVM эффективны в условиях ограниченного объема данных, но требуют тщательной настройки гиперпараметров.
* **Нейронные сети** — особенно рекуррентные нейронные сети (RNN) и их модификации, такие как LSTM (Long Short-Term Memory), предназначены для работы с временными рядами и способны учитывать долгосрочные зависимости в данных, что делает их идеальными для анализа курсов валют. Нейронные сети выбранной архитектуры применялись не только для прогнозирования курсов валют, но и для таких задач, как предсказание температуры воздуха и индексов потребительских цен. Применение нейронных сетей значительно повышает точность прогнозов, особенно в условиях высокой волатильности рынка.
* **Ансамблевые методы** — такие как градиентный бустинг и случайный лес, комбинируют несколько моделей для повышения точности прогнозов. Эти методы демонстрируют высокую эффективность в задачах экономического прогнозирования благодаря способности учитывать разнообразие данных и снижать вероятность переобучения. Это делает ансамблевые методы важным инструментом для анализа сложных финансовых временных рядов [12].

Машинное обучение особенно полезно в условиях высокой размерности данных и наличия нелинейных зависимостей. Например, нейронные сети часто применяются для прогнозирования курсов валют, так как они могут учитывать сложные факторы, такие как волатильность рубля, зависимость от цен на энергоносители и геополитические события.

# 1.2.3. Сравнение традиционных методов и методов МЛ

Сравнение методов представлено в таблице 1.

Таблица 1 - Сравнение методов

|  |  |  |
| --- | --- | --- |
| Метод | Преимущества | Недостатки |
| ARIMA | Простота, высокая интерпретируемость | Ограниченность при наличии сложных нелинейных зависимостей |
| Экспоненциальное сглаживание | Учет трендов и сезонности | Невозможность обработки большого количества внешних факторов |
| Линейная регрессия | Простота, быстрая реализация | Низкая точность для нелинейных данных |
| SVM (Support Vector Machines) | Эффективность при небольшом объёме данных | Сложность настройки гиперпараметров |
| Нейронные сети (LSTM) | Учет долгосрочных и сложных зависимостей во временных рядах | Высокие вычислительные затраты |
| Ансамблевые методы | Высокая точность, устойчивость к шуму | Сложность интерпретации и настройки |

В ходе анализа традиционных статистических методов и современных подходов машинного обучения мы убедились, что классические методы (ARIMA, экспоненциальное сглаживание) обладают важными преимуществами — математической строгостью, прозрачностью и простотой реализации. Однако их способность учитывать только линейные зависимости и ограниченное число факторов делает их недостаточно гибкими в условиях рыночной волатильности и влияния внешних событий.

Методы машинного обучения, такие как линейная и полиномиальная регрессия, SVM, рекуррентные сети (LSTM) и ансамблевые алгоритмы, позволяют захватывать сложные и нелинейные зависимости во временных рядах, работать с большим числом признаков и адаптироваться к быстро меняющимся условиям рынка. При этом каждый из этих методов имеет свои ограничения: чувствительность к выбросам, вычислительную сложность, необходимость настройки множества гиперпараметров и относительную «черность» моделей.

Сопоставление традиционных и MЛ‑подходов с учетом их достоинств и недостатков показало, что наилучший прогнозный результат достигается при комбинировании регрессионных моделей: их линейная часть обеспечивает базовую интерпретируемость, в то время как, например, добавление полиномиальных признаков, простых нейронных блоков или методов опорных векторов повышает гибкость и точность прогноза. Именно такой комбинированный подход закладывает прочную основу для построения ИИС прогнозирования курсов валют, способной учитывать широкий спектр факторов и адаптироваться к изменениям рыночной конъюнктуры.

## 1.3. Выбор подхода для разработки ИИС

Для разработки данной ИИС был выбран комбинированный регрессионный подход на основе методов машинного обучения. Регрессионная модель обеспечивает количественную оценку влияния каждого входного признака — исторических значений курса, макроэкономических индикаторов, календарных факторов и прочих — на будущие значения валютного курса, что важно как для интерпретации результатов, так и для обоснования принимаемых решений пользователями ИИС.

Комбинированная регрессионная модель строится следующим образом: в качестве «ядра» используется линейная регрессия, которая определяет базовый тренд и позволяет оценить вклад каждого фактора. Для учёта более сложных, нелинейных зависимостей к исходным данным добавляются полиномиальные и взаимодействующие признаки, а остатки линейной модели при необходимости анализируются с помощью «простых» нейронных блоков или метода опорных векторов (SVM). Такой гибридный механизм сочетает интерпретируемость классической регрессии с гибкостью нелинейных моделей и позволяет повысить точность прогноза в условиях рыночной волатильности и множественных внешних шоков.

В качестве базовой точки сравнения будет использоваться классическая модель ARIMA, которая представляет собой стандарт в статистическом прогнозировании временных рядов. Путём сравнительного анализа результатов обеих моделей по метрикам R², MAE, MAPE и RMSE мы сможем объективно оценить эффективность регрессионного подхода и выбрать наиболее устойчивую и точную модель для последующей интеграции в Telegram‑бот.

Также необходимо учитывать, что финансовые данные, особенно валютные курсы, подвержены резким колебаниям, вызванным как рыночными, так и внешнеполитическими факторами. Такие резкие изменения могут восприниматься моделью как выбросы — аномальные значения, способные искажать результаты обучения и ухудшать качество прогноза. В рамках данной работы будет проведён дополнительный анализ: модели будут обучены как на исходных данных с выбросами, так и на очищенных данных, из которых выбросы удалены или сглажены. Это позволит оценить устойчивость регрессионного подхода к аномалиям и выбрать наиболее подходящую стратегию обработки данных на этапе предобработки. Сравнение результатов на двух выборках даст более полное представление о надёжности модели и её способности адаптироваться к реальным условиям валютного рынка.

## 1.4. Источники данных

Для построения модели прогнозирования валютных курсов в рамках интеллектуальной информационной системы (ИИС) будут использованы официальные и проверенные источники данных, позволяющие получить достоверную выборку для обучения модели.

Основой станут исторические данные о курсах валют, в частности, доллар США и евро по отношению к рублю, полученные с официального сайта Центрального банка Российской Федерации. Эти данные охватывают длительный период наблюдений и включают ежедневные значения курса с 2010 года по настоящее время [2-3].

Дополнительно, с целью повышения точности прогноза, в модель будут включены макроэкономические и рыночные индикаторы, которые оказывают значительное влияние на валютные колебания:

* Уровень инфляции и ключевая ставка ЦБ РФ — предоставляются на официальном сайте Банка России и являются одними из важнейших параметров денежно-кредитной политики [1].
* Динамика цен на нефть марки Brent — учитывается как важнейший фактор, влияющий на курс рубля, и будет получена с платформы Investing.com, где представлены исторические котировки нефти в долларах за баррель [4].
* Цены на драгоценные металлы — золото, серебро, платина и палладий в рублях за грамм, предоставляются Банком России и могут быть полезны как индикаторы глобальных инфляционных ожиданий и рыночной неопределённости [5].

Интеграция этих факторов в модель обеспечит более точный и реалистичный прогноз, учитывающий не только статистику по курсам, но и ключевые экономические взаимосвязи, влияющие на валютный рынок. Таким образом, формируется комплексный подход к подготовке данных, необходимый для построения качественной регрессионной модели.

## 1.5. Выводы к главе 1

В первой главе курсовой работы были рассмотрены теоретические и практические основы прогнозирования курсов валют. Описана актуальность данной задачи в условиях нестабильности финансовых рынков, проведён обзор традиционных статистических методов и современных подходов машинного обучения, а также обоснован выбор регрессионной модели в качестве основы для построения интеллектуальной информационной системы.

На основе сравнительного анализа было принято решение использовать комбинированную регрессионную модель, объединяющую интерпретируемость линейной регрессии и гибкость современных алгоритмов. Также запланировано провести обучение моделей как на данных с выбросами, так и на очищенных, что позволит повысить устойчивость системы к аномалиям в данных.

Были определены ключевые источники данных, включая курсы валют, макроэкономические показатели и рыночные индикаторы. Их интеграция в модель позволит формировать прогноз с учетом реальных факторов, влияющих на динамику валютного курса.

В следующих главах будет проведён анализ собранных данных, реализация и сравнение моделей, а также создание Telegram-бота, в который будет встроена наиболее эффективная модель для предоставления прогноза пользователям.

# АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

## 2.1. Проблематика

Валютный рынок — это глобальная система, где цены на валюты постоянно меняются под влиянием множества взаимосвязанных причин. Основные из них — экономические показатели (например, уровень инфляции, процентные ставки, состояние экономики стран), политические решения, геополитическая нестабильность и даже поведение самих участников рынка, чьи ожидания могут сами влиять на движение цен. В условиях глобизации, когда страны связаны торговыми связями, инвестициями и информационными потоками, способность предсказывать колебания валют становится особенно важной. Для компаний это помогает минимизировать риски при международных расчетах, для государств — контролировать стоимость своей валюты, а для частных инвесторов — зарабатывать на разнице курсов.

Классические (традиционные) методы прогнозирования, такие как простые статистические модели или методы анализа временных рядов, работают хорошо только в спокойные периоды. Они не справляются с резкими скачками, вызванными внезапными событиями — например, неожиданными действиями центральных банков или глобальными кризисами. Эти модели также не учитывают «редкие, но мощные» события (типа пандемий или энергетических кризисов), которые кардинально меняют ситуацию. Кроме того, они слабо справляются с анализом качественных факторов, таких как доверие инвесторов или изменения в торговых отношениях между странами. Из-за этого в сложных ситуациях такие методы становятся менее полезными.

С развитием технологий появляется возможность использовать гораздо больше данных. Современные подходы позволяют совмещать информацию не только об исторических ценах валют, но и данные о состоянии экономик, новости, реакцию пользователей в интернете, индексы деловой активности и даже курсы акций или сырьевых товаров. Однако перед применением этих данных их нужно тщательно подготовить: удалить ошибки, заполнить пропуски, убрать сезонные колебания. Также важно выяснить, какие именно факторы оказывают наибольшее влияние на конкретную валютную пару — например, разница процентных ставок между странами или объем экспорта и импорта.

Но даже хорошие данные не гарантируют успешного прогноза, если модель не адаптируется к изменяющимся условиям. Экономическая реальность часто непредсказуема: пандемии, конфликты и другие кризисы требуют гибкости в подходах. При этом возникает противоречие: чем сложнее модель, тем труднее объяснить её выводы. Если алгоритм работает как «черный ящик», его результаты сложно использовать в ответственных задачах. Поэтому важно находить баланс между точностью прогноза и понятностью его механизма.

Еще один ключевой момент — нужно учитывать цели прогноза. К примеру, если целью является краткосрочный трейдинг, то тут чаще всего подразумевается быстрая реакция на текущие события, а если цель —долгосрочные инвестиции, то инвесторы часто стремятся понять основные экономические факторы. Это указывает на необходимость создания универсальных инструментов, которые могли бы учитывать разные временные горизонты и специфику задачи. Современные исследования подтверждают, что эффективность прогнозов возрастает при комбинации математических методов и экспертного анализа. Даже самые продвинутые алгоритмы не заменят глубокого понимания экономических законов и исторического контекста [9].

## 2.2. Подбор подходящего набора данных

Для построения ИИС прогнозирования курсов валют были использованы официальные исторические данные по курсу доллара США к рублю, полученные с сайта Центрального банка РФ [2].

Данные представлены в виде временного ряда, где каждый объект соответствует одному дню наблюдения и содержит цену открытия, закрытия, минимальное и максимальное значения курса за день, а также вспомогательные показатели (день недели, месяц и пр.).

Дополнительно использовались макроэкономические показатели, включая цену на нефть марки Brent, ключевую ставку Центрального банка, уровень инфляции, цену на драгоценные металлы (золото, серебро, палладий), курс евро.

* Цена на нефть марки Brent — является ключевым экспортным товаром для российской экономики, и ее цена напрямую влияет на курс рубля. Повышение стоимости нефти ведёт к росту валютных поступлений в страну, укреплению рубля и снижению курса доллара/евро, и наоборот. Для анализа использовалась ежедневная динамика цен на нефть, выраженная в долларах США за баррель. Данные охватывают период с 2010 года по 2025 год и позволяют учитывать краткосрочные колебания и долгосрочные тренды на сырьевом рынке [4].
* Ключевая ставка Центрального банка Российской Федерации (ЦБ РФ) — отражает денежно-кредитную политику России и используется как основной инструмент регулирования инфляции и стабилизации рубля. Повышение ставки делает вложения в рублевые активы более привлекательными, способствуя укреплению рубля. Для анализа использовалась официальная информация с сайта ЦБ РФ, где приведены даты изменения ставки и ее актуальные значения. Этот показатель вводился в модель как лагированный временной ряд с недельной частотой [1].
* Уровень инфляции — измеряется как процентное изменение потребительских цен по отношению к предыдущему периоду. Высокая инфляция приводит к снижению покупательной способности рубля, что может ослаблять национальную валюту и повышать курс доллара/евро. В модели используется годовой уровень инфляции (в процентах) на основе данных Центрального банка. Показатель агрегировался на месячном уровне и интерполировался до ежедневного [1].
* Цены на драгоценные металлы — измеряется как процентное изменение потребительских цен по отношению. Использовались официальные данные Банка России о стоимости золота, серебра, палладия в рублях за грамм [5]. Эти активы считаются защитными инструментами инвестирования в условиях финансовой нестабильности. Рост цен на металлы часто коррелирует с ослаблением национальных валют, включая рубль. Цены включались в модель в виде отдельных временных рядов, нормализованных по стандартной шкале.
* Курс евро к рублю (EUR/RUB) — Курс евро использовался вспомогательно, как сопоставительный внешний показатель и дополнительный индикатор оценки глобальных трендов на валютном рынке. Поскольку курс евро часто подвержен аналогичным колебаниям, что и доллар, его включение позволяет повысить чувствительность модели к синхронным движениям на международных рынках. Исторические значения курса евро к рублю получены с официального сайта ЦБ РФ (код валюты R01239) [3].

Набор данных в программе Microsoft Excel представлен на рисунке 1.

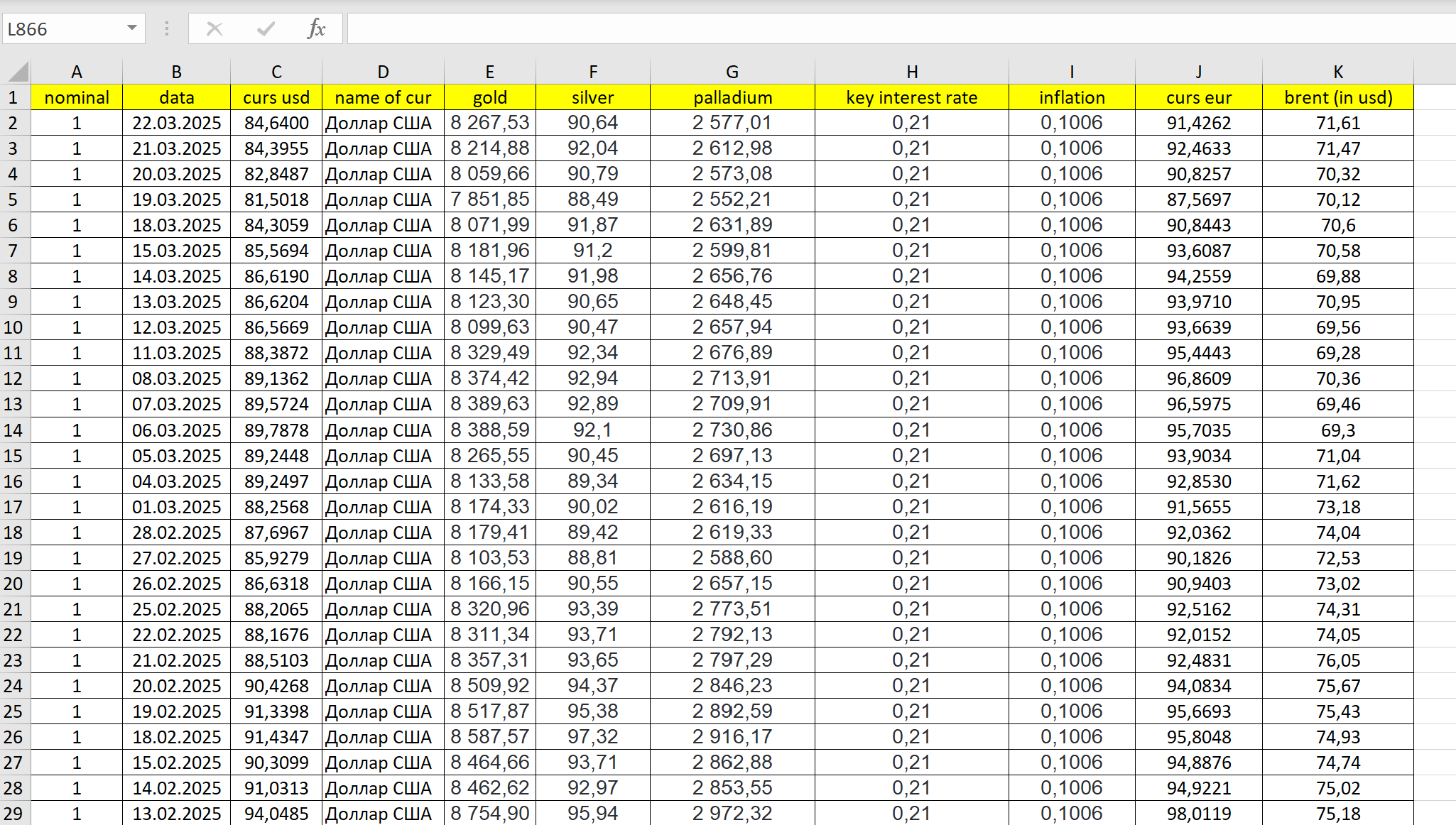


Рис. 1. Набор данных в MS Excel

Набор данных был расширен признаками, отражающими календарные закономерности и сезонность, что важно для анализа валютных временных рядов.

Для проведения экспериментов были подготовлены два поднабора данных: df\_d1 — с устраненными выбросами и df\_d2 — оригинальный набор с сохранёнными выбросами.

Поднабор данных df\_d1 (с устраненными выбросами) был создан путем удаления аномальных значений, которые могут исказить результаты анализа временных рядов. Для этого использовались методы обнаружения выбросов, такие как межквартильный размах (IQR) или метод Z-оценок. Устранение выбросов помогает улучшить стабильность модели и избежать чрезмерного влияния экстремальных значений на прогноз.

В отличие от первого поднабора, в случае поднабора данных df\_d2 (оригинальный набор с сохранёнными выбросами) выбросы оставлены в данных, что позволяет учитывать их влияние на анализ и прогнозирование. Сохранение выбросов может быть полезным для моделирования более реальных сценариев, особенно в контексте валютных временных рядов, где экстремальные колебания курсов могут быть важными индикаторами. Однако важно отметить, что наличие выбросов может повысить чувствительность модели к аномалиям, что может потребовать применения более сложных методов обработки.

## 2.3 Анализ и подготовка данных

Данный этап нужен для оценки качества данных и их подготовки к использованию для обучения моделей.

Первым шагом был решен вопрос хронологического упорядочивания временного ряда. Поскольку изначально данные были представлены в обратном порядке — от самой последней даты к более ранней — это противоречило логике прогнозирования, где важно сохранять естественное течение времени.

Для корректной работы моделей данные были отсортированы по возрастанию дат с помощью следующего фрагмента кода:

df\_d['data'] = pd.to\_datetime(df\_d['data']) # преобразование столбца с датой в формат datetime

df\_d.set\_index('data', inplace=True) # дата в качестве индекса

df\_d.sort\_index(ascending=True, inplace=True) # сортировка по дате по возрастанию

На втором шаге было выполнено заполнение пропущенных значений в исходном временном ряде. Они наблюдались в макроэкономических показателях: ключевой ставке и уровне инфляции. Пропущенные значения были обработаны в два этапа:

* Применялось заполнение методом обратного переноса значений (backfill) — ближайшим следующим по времени известным значением;
* Производилась линейная интерполяция, восстанавливающая плавную динамику между известными точками.

Для устранения пропущенных значений в показателях ключевой ставки и инфляции был использован следующий фрагмент программного кода:

df\_d['key interest rate'].fillna(method='bfill', inplace=True)

df\_d['key interest rate'] = df\_d['key interest rate'].interpolate(method='linear')

df\_d['inflation'].fillna(method='bfill', inplace=True)

df\_d['inflation'] = df\_d['inflation'].interpolate(method='linear')

fillna(method=”bfill”) выполняет обратное заполнение: если в текущей ячейке отсутствует значение, оно заменяется ближайшим следующим известным (по времени), а interpolate(method=”linear”) — линейную интерполяцию между двумя известными точками, равномерно заполняя промежуточные пропущенные значения.

На рисунке 2 и 3 представлена информация о структуре данных в формате таблицы, где каждый столбец описан по параметрам: название, количество непустых значений (Non-Null Count) и тип данных (Dtype). Рисунок 1 демонстрирует состояние данных до обработки пропущенных значений, где некоторые столбцы (например, key interest rate и inflation) имеют меньшее количество непустых значений (3013), указывая на наличие пропусков.

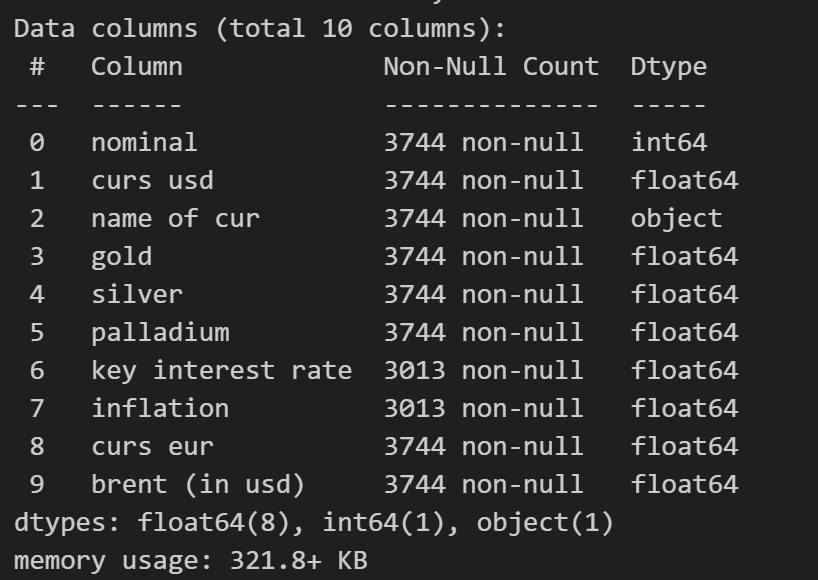


Рис. 2. Структура данных до обработки пропущенных значений

После обработки пропущенных значений (на рисунке 3), все столбцы содержат полные данные (3744 non-null), что свидетельствует о успешном заполнении или удалении пропусков.

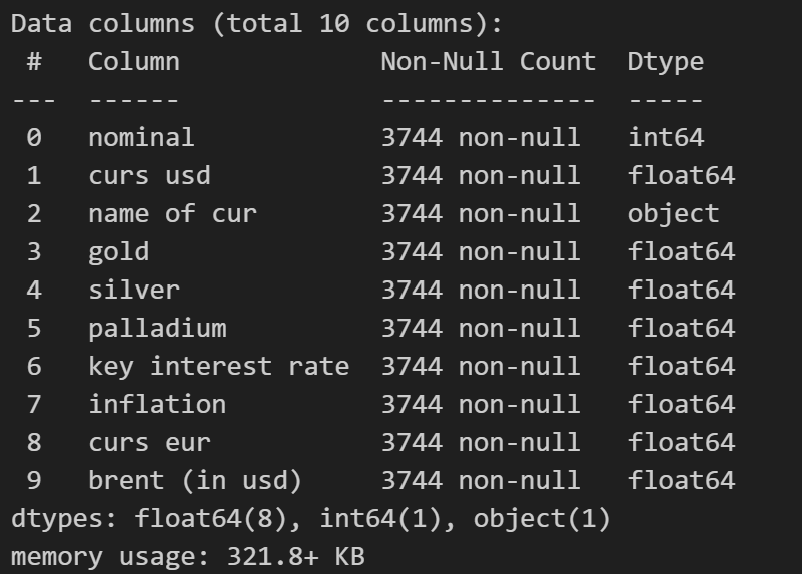


Рис. 3. Структура данных после обработки пропущенных значений

Как видно по рисунку 3, тип данных остаются неизменными, а общее количество столбцов также сохраняется равным 10.

После завершения этапа обработки пропущенных значений следующим шагом стало выявление и работа с выбросами — аномально высокими или низкими значениями в ряде признаков. Такие значения, как правило, возникают в периоды экономических шоков, политической нестабильности или резких изменений на рынке.

С целью проведения сравнительного анализа устойчивости моделей к выбросам было принято решение разделить основной датафрейм df\_d на два отдельных набора данных: df\_d1 — копия исходного датафрейма, в котором выбросы были удалены с использованием метода межквартильного размаха (IQR), df\_d2 — оригинальный набор, где выбросы были сохранены, чтобы оценить, как модели справляются с «сырыми» данными.

Перед удалением выбросов из датасета df\_d1 был проведен тщательный этап анализа, который включал как визуальный, так и статистический подходы. В первую очередь, для ключевых признаков было построено 8 диаграмм «ящик с усами» (boxplot) для каждого из столбцов с числовым признаком, что позволило наглядно оценить распределение данных и выявить значения, значительно выходящие за пределы типичного диапазона. Эти графики были созданы отдельно для исходного набора данных df\_d. Получившиеся графики изображены на рисунке 4.

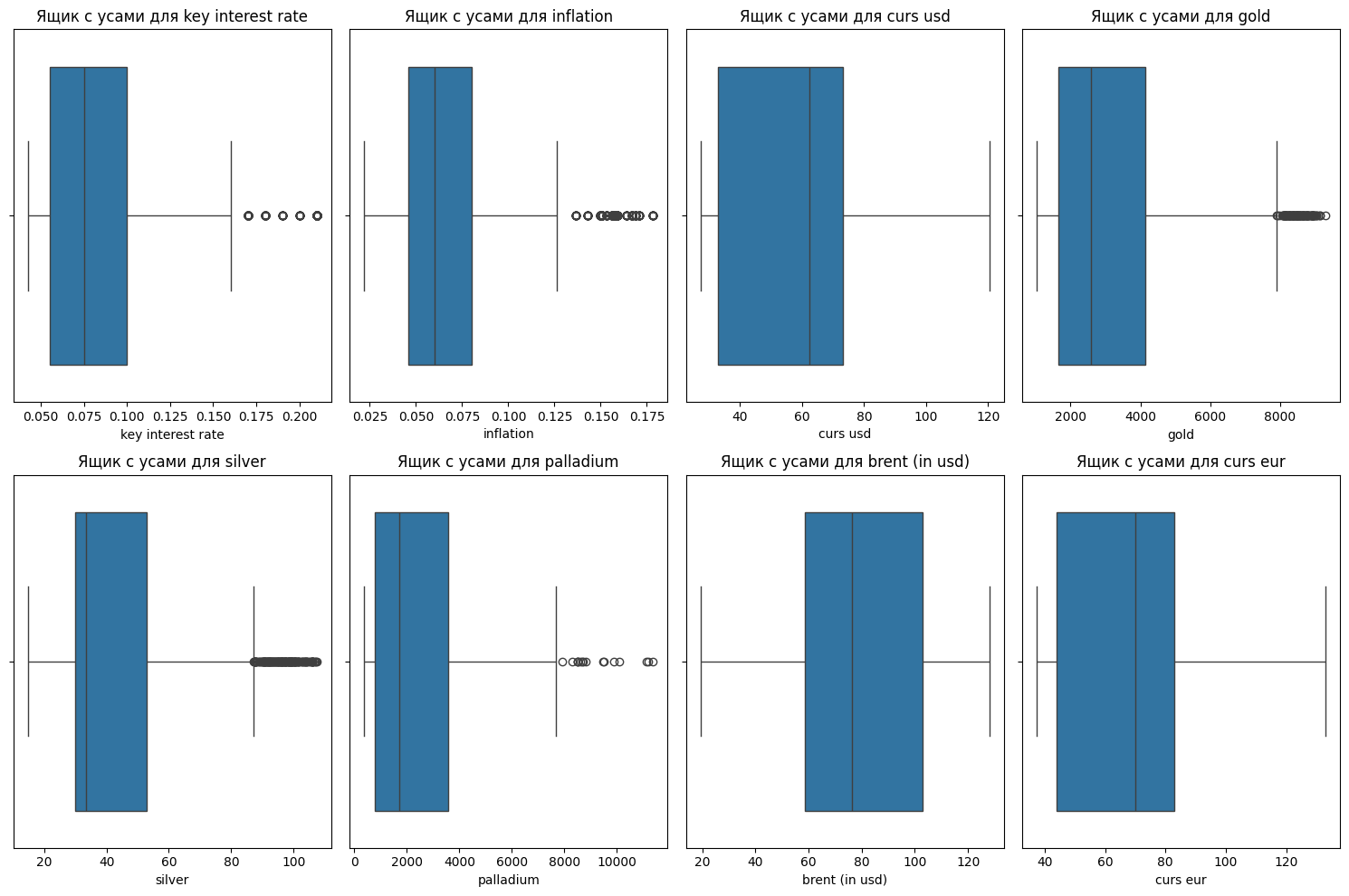


Рис. 4. Диаграммы boxplot для различных числовых признаков до обработки выбросов

Рассмотрим рисунок 4, представляющий собой диаграммы «ящик с усами» для различных признаков. Анализ графиков показывает, что такие параметры, как цена на нефть марки Brent (в долларах), курс доллара и курс евро, характеризуются компактным распределением и не имеют выбросов, при этом медианы находятся примерно по центру ящика, что свидетельствует о симметричности данных. У ключевой ставки наблюдаются незначительные выбросы, которые локализуются за пределами правого уса, однако они не оказывают существенного влияния на общую картину распределения. В то же время остальные признаки — инфляция, золото, серебро и палладий демонстрируют значительное количество выбросов, особенно это выражено у золота, где значения явно выходят за границы усов, указывая на высокую неоднородность данных. Для признаков с выбросами понадобиться дополнительная обработка, направленная на предотвращение искажений в процессе анализа или построения моделей.

Для выявления и устранения выбросов в числовых признаках были рассмотрены два подхода: первый — на основе интервала трех сигм (μ ± 3σ), где выбросами считаются значения, значительно отклоняющиеся от среднего, и второй — с использованием межквартильного размаха (IQR), который является более универсальным и устойчивым к несимметричным распределениям.

Для выявления и устранения выбросов в числовых признаках были рассмотрены два подхода: первый — на основе интервала трёх сигм (μ ± 3σ), согласно которому границы допустимых значений определяются как среднее арифметическое (μ) плюс/минус три стандартных отклонения (σ), а выбросами считаются значения, выходящие за эти пределы, такой метод хорошо подходит для данных, распределённых нормально. Второй подход основан на использовании межквартильного размаха (IQR), который рассчитывается как разница между третьим (Q3) и первым (Q1) квартилями, где Q1 соответствует 25-му процентилю (значение, ниже которого находятся 25% данных), а Q3 — 75-му процентилю (значение, ниже которого находятся 75% данных) распределения, границы допустимого диапазона определяются по формулам Q1 – 1.5\*IQR и Q3 + 1.5\*IQR — такой метод является более универсальным, так как не предполагает нормальности распределения и устойчив к его асимметрии.

Первый метод предполагал нормальность распределения и применялся для предварительной проверки по таким признакам, как курс доллара и евро, цены на нефть и металлы, ключевая ставка и инфляция. Вычисление границ нормальных значений по правилу трех сигм:

# Проверка выбросов для курса доллара

Mcd = df\_d1['curs usd'].mean() # ср. знач

Scd = df\_d1['curs usd'].std() # cр. кв. отклонение

print(f"Значение курса доллара не считается выбросом на интервале ({Mcd-3\*Scd}, {Mcd+3\*Scd})")

# Проверка выбросов для цены на золото

Mg = df\_d1['gold'].mean()

Sg = df\_d1['gold'].std()

print(f"Значение цены золота не считается выбросом на интервале ({Mg-3\*Sg}, {Mg+3\*Sg})")

# Проверка выбросов для цены на серебро

Mss = df\_d1['silver'].mean()

Sss = df\_d1['silver'].std()

print(f"Значение цены серебра не считается выбросом на интервале ({Mss-3\*Sss}, {Mss+3\*Sss})")

# Проверка выбросов для цены на палладий

Mpp = df\_d1['palladium'].mean()

Spp = df\_d1['palladium'].std()

print(f"Значение цены палладия не считается выбросом на интервале ({Mpp-3\*Spp}, {Mpp+3\*Spp})")

# Проверка выбросов для цены на нефть Brent

Mb = df\_d1['brent (in usd)'].mean()

Sb = df\_d1['brent (in usd)'].std()

print(f"Значение цены на нефть Brent не считается выбросом на интервале ({Mb-3\*Sb}, {Mb+3\*Sb})")

# Проверка выбросов для курса евро

Mce = df\_d1['curs eur'].mean()

Sce = df\_d1['curs eur'].std()

print(f"Значение курса доллара не считается выбросом на интервале ({Mce-3\*Sce}, {Mce+3\*Sce})")

# Проверка выбросов для кл. ставки

Mkir = df\_d1['key interest rate'].mean()

Skir = df\_d1['key interest rate'].std()

print(f"Значение ключевой ставки не считается выбросом на интервале ({Mkir-3\*Skir}, {Mkir+3\*Skir})")

# Проверка выбросов для инфляции

Minf = df\_d1['inflation'].mean()

Sinf = df\_d1['inflation'].std()

print(f"Значение инфляции не считается выбросом на интервале ({Minf-3\*Sinf}, {Minf+3\*Sinf})")

В результате работы данного фрагмента кода получены следующие границы, представленные на рисунке 5.

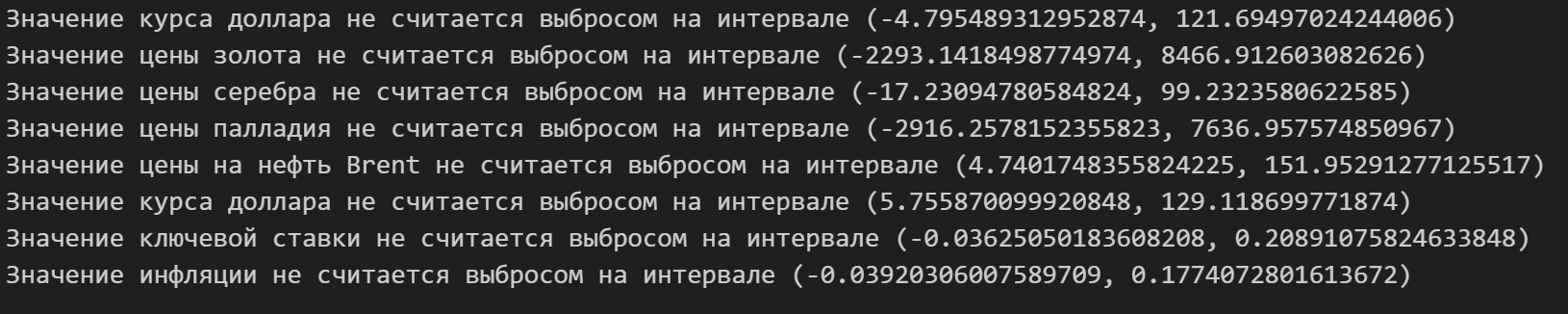


Рис. 5. Границы, при которых значения столбцов не считаются выбросами, полученные методом 3-х сигм

Однако в рамках основного этапа устранения выбросов использовался именно метод IQR (межквартильный размах), в котором выбросы определяются как значения, выходящие за границы [Q1 – 1,5\*IQR; Q3 + 1,5\*IQR] (Q1 и Q3 — квартильные значения). Фрагмент кода, рассчитывающий эти границы:

# Проверка выбросов для курса доллара

# расчет 1-ой (Q1) и 3-ей (Q3) квартили

Q1\_usd = df\_d1['curs usd'].quantile(0.25)

Q3\_usd = df\_d1['curs usd'].quantile(0.75)

IQR\_usd = Q3\_usd - Q1\_usd # межквартильный размах (IQR)

print(f"Значение курса доллара не считается выбросом на интервале ({Q1\_usd-1.5\*IQR\_usd}, {Q3\_usd+1.5\*IQR\_usd})") # вывод границ

# Проверка выбросов для цены на золото

Q1\_gold = df\_d1['gold'].quantile(0.25)

Q3\_gold = df\_d1['gold'].quantile(0.75)

IQR\_gold = Q3\_gold - Q1\_gold

print(f"Значение цены золота не считается выбросом на интервале ({Q1\_gold-1.5\*IQR\_gold}, {Q3\_gold+1.5\*IQR\_gold})")

# Проверка выбросов для цены на серебро

Q1\_silver = df\_d1['silver'].quantile(0.25)

Q3\_silver = df\_d1['silver'].quantile(0.75)

IQR\_silver = Q3\_silver - Q1\_silver

print(f"Значение цены серебра не считается выбросом на интервале ({Q1\_silver-1.5\*IQR\_silver}, {Q3\_silver+1.5\*IQR\_silver})")

# Проверка выбросов для цены на палладий

Q1\_palladium = df\_d1['palladium'].quantile(0.25)

Q3\_palladium = df\_d1['palladium'].quantile(0.75)

IQR\_palladium = Q3\_palladium - Q1\_palladium

print(f"Значение цены палладия не считается выбросом на интервале ({Q1\_palladium-1.5\*IQR\_palladium}, {Q3\_palladium+1.5\*IQR\_palladium})")

# Проверка выбросов для цены на нефть Brent

Q1\_brent = df\_d1['brent (in usd)'].quantile(0.25)

Q3\_brent = df\_d1['brent (in usd)'].quantile(0.75)

IQR\_brent = Q3\_brent - Q1\_brent

print(f"Значение цены на нефть Brent не считается выбросом на интервале ({Q1\_brent-1.5\*IQR\_brent}, {Q3\_brent+1.5\*IQR\_brent})")

# Проверка выбросов для курса евро

Q1\_eur = df\_d1['curs eur'].quantile(0.25)

Q3\_eur = df\_d1['curs eur'].quantile(0.75)

IQR\_eur = Q3\_eur - Q1\_eur

print(f"Значение курса евро не считается выбросом на интервале ({Q1\_eur-1.5\*IQR\_eur}, {Q3\_eur+1.5\*IQR\_eur})")

# Проверка выбросов для кл. ставки

Q1\_keyrate = df\_d1['key interest rate'].quantile(0.25)

Q3\_keyrate = df\_d1['key interest rate'].quantile(0.75)

IQR\_keyrate = Q3\_keyrate - Q1\_keyrate

print(f"Значение ключевой ставки не считается выбросом на интервале ({Q1\_keyrate-1.5\*IQR\_keyrate}, {Q3\_keyrate+1.5\*IQR\_keyrate})")

# Проверка выбросов для инфляции

Q1\_inf = df\_d1['inflation'].quantile(0.25)

Q3\_inf = df\_d1['inflation'].quantile(0.75)

IQR\_inf = Q3\_inf - Q1\_inf

print(f"Значение инфляции не считается выбросом на интервале ({Q1\_inf-1.5\*IQR\_inf}, {Q3\_inf+1.5\*IQR\_inf})")

В результате работы данного фрагмента кода получены следующие интервалы, продемонстрированные на рисунке 6.

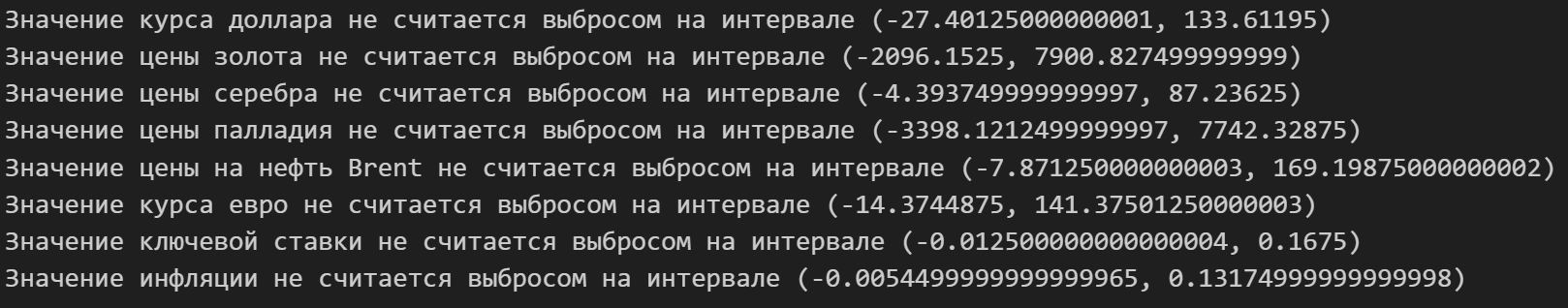


Рис. 6. Границы, при которых значения столбцов не считаются выбросами, полученные методом IQR

Метод IQR (межквартильный размах) позволил более точно выявить аномальные значения в данных и применить фильтрацию без существенного искажения общей структуры набора. В отличие от метода трёх сигм, который предполагает нормальное распределение и может ошибочно классифицировать экстремальные, но закономерные значения как выбросы, IQR основывается на 25-м и 75-м процентилях.

После того как для каждого числового признака были определены границы допустимых значений с помощью метода межквартильного размаха (IQR), были созданы «фильтры», позволяющие устранить выбросы из набора данных df\_d1.

В зависимости от характера признака и его поведения на рынке применялись замена на медиану или усечение (clip).

Для признака цены золота использовалась замена выбросов на медиану, поскольку золото отличается стабильностью и сравнительно низкой волатильностью.

gold\_median = df\_d1['gold'].median() # медианное значение золота

df\_d1['gold'] = np.where(

(df\_d1['gold'] < lower\_gold) | (df\_d1['gold'] > upper\_gold), # если значение за пределами IQR

gold\_median, # заменить на медиану

df\_d1['gold'] # иначе оставить без изменений

)

Серебро, в отличие от золота, более волатильно, поэтому вместо замены использовалось усечение значений до границ IQR:

df\_d1['silver'] = df\_d1['silver'].clip(Q1\_silver - 1.5\*IQR\_silver, Q3\_silver + 1.5\*IQR\_silver) # ограничиваем значения

Для палладия, который подвержен резким колебаниям из-за дефицита на рынке, применялась более мягкая фильтрация с коэффициентом 1:

df\_d1['palladium'] = df\_d1['palladium'].clip(Q1\_palladium - 1\*IQR\_palladium, Q3\_palladium + 1\*IQR\_palladium) # клиппинг с узкими границами

Признаки ключевая ставка и инфляция были обработаны с помощью универсального цикла. Для каждого из них рассчитывались границы по IQR, и значения усекались аналогично:

for col in ['key interest rate', 'inflation']:

Q1\_n = df\_d1[col].quantile(0.25)

Q3\_n = df\_d1[col].quantile(0.75)

IQR\_n = Q3\_n - Q1\_n

low\_bound\_n = Q1\_n - 1.5 \* IQR\_n

up\_bound\_n = Q3\_n + 1.5 \* IQR\_n

df\_d1[col] = df\_d1[col].clip(low\_bound\_n, up\_bound\_n) # ограничиваем значения в допустимом диапазоне

После того как для каждого признака были применены подходы к устранению выбросов, учитывающие их экономическую суть и поведение на рынке, был создан финальный очищенный набор данных df\_d1. Чтобы наглядно оценить, насколько эффективно были обработаны аномалии, снова были построены диаграммы «ящик с усами» (boxplot) для всех числовых признаков — так же, как это делалось раньше. На рисунке 7 продемонстрированы графики boxplot после выполнения процедуры по устранению выбросов для df\_d1.



Рис. 7. Диаграммы boxplot для различных числовых признаков df\_d1 после обработки выбросов

Рассмотрим рисунок 7, который демонстрирует диаграммы boxplot для всех числовых признаков после выполнения фильтрации выбросов. Визуальный анализ показывает, что большинство признаков теперь имеют более компактное и сбалансированное распределение данных, без значительных аномалий за пределами границ IQR. Однако стоит отметить, что для признака gold (цена золота) незначительные выбросы всё ещё присутствуют, хотя их количество значительно сократилось по сравнению с исходными данными. Эти оставшиеся выбросы у золота можно объяснить двумя факторами: во-первых, консервативным методом обработки, при котором выбросы заменялись на медиану вместо использования усечения (clipping), это сохраняет большую часть данных в их оригинальной форме, но может позволить некоторым экстремальных значений остаться, если они не слишком далеки от основного диапазона. Во-вторых, устойчивость цены золота, которая, хотя и относительно стабильна, иногда подвержена редким, но заметным колебаниям из-за рыночных факторов или спекулятивной активности, такие редкие отклонения могут быть восприняты как естественные особенности рынка золота, а не как аномалии, требующие удаления. Хотя выбросы у золота все еще видны, их влияние минимально, и они не искажают общую картину распределения.

Чтобы оценить влияние выбросов на взаимосвязи признаков, мы построили и сравнили две корреляционные матрицы: одну для данных с сохраненными выбросами (df\_d2), другую — для очищенного набора (df\_d1).

На рисунках 8 и 9 изображены матрицы для данных df\_d2 и df\_d1 соответственно.

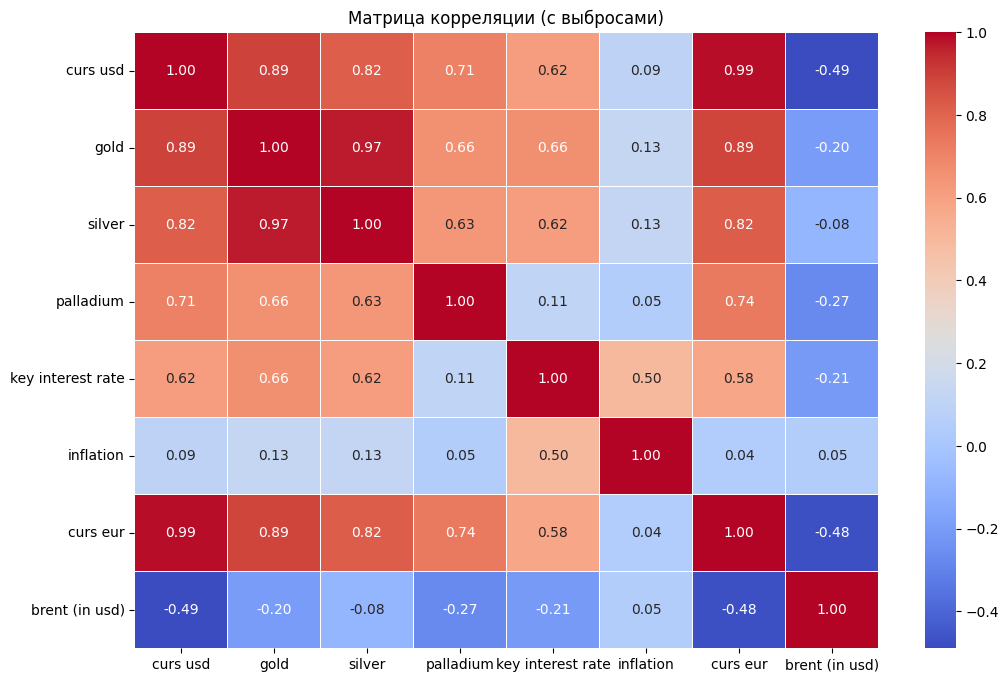


Рис. 8. Матрица корреляции с выбросами (df\_d2)

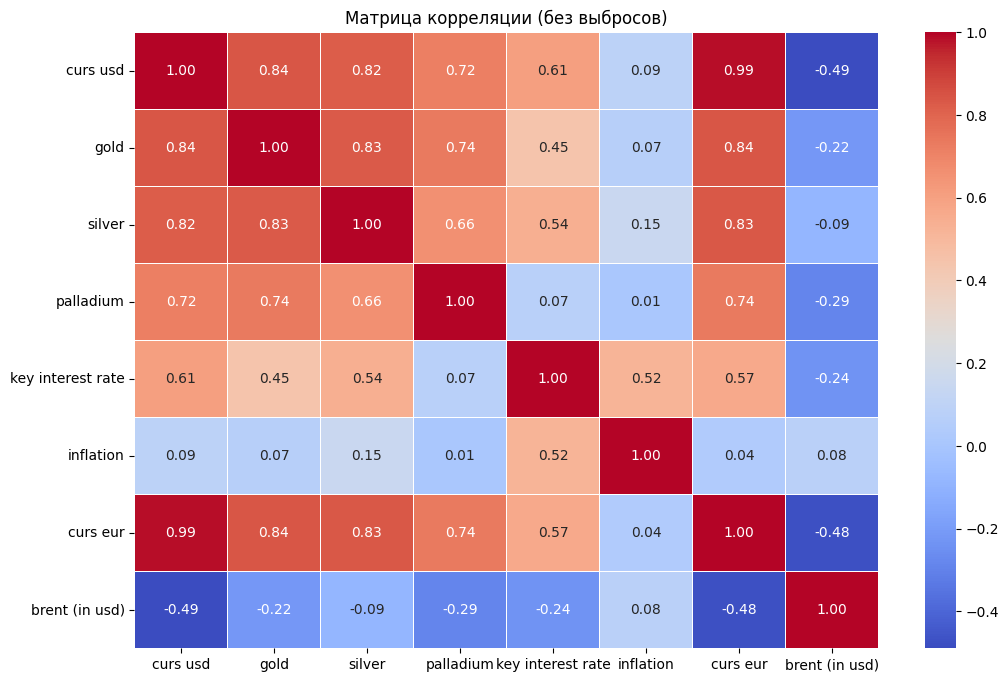


Рис. 9. Матрица корреляции без выбросов (df\_d1)

Матрица, построенная на данных с выбросами (df\_d2), на рисунке 8, показала очень сильные связи между ценой на золото (gold) и серебро (silver) — корреляция составила 0,97 и это может выглядеть чрезмерно высоким. Также очень тесно оказались связаны курсы валют (curs usd, curs eur) и цена нефти (brent). Вероятнее всего, такие высокие значения — следствие экстремальных скачков (выбросов), характерных для цен на металлы или инфляции.

Когда были убраны эти аномальные значения (у df\_d1), взаимосвязи стали выглядеть гораздо логичнее. Корреляция золота и серебра снизилась до 0,83 и это подтвердило искусственный характер предыдущей цифры. Зависимости между валютными курсами и нефтью также ослабли до более ожидаемого уровня. Интересно, что несильные связи, например, ключевой ставки с другими показателями, после чистки стали даже заметнее. Это говорит о том, что очищенные данные (df\_d1) честнее отражают структуру зависимостей, не замутненную случайными пиками.

Процедура удаления выбросов существенно повысила достоверность корреляционного анализа. Очищенный набор данных (df\_d1) показывает более логичные и стабильные взаимосвязи между признаками, что делает его предпочтительным для дальнейшего анализа и построения моделей. Однако, несмотря на преимущества df\_d1, было решено использовать оба датасета — как с выбросами (df\_d2), так и без них (df\_d1), что позволит провести сравнение результатов моделирования на разных версиях данных, оценить, насколько выбросы влияют на точность прогнозов, и выбрать наиболее надежный подход.

## 2.4. Машинное обучение. Оценка качества моделей

На данном этапе исследования были разработаны и протестированы регрессионные модели для прогнозирования курса доллара США к рублю. Основная цель — создание комбинированной модели, которая объединяет несколько подходов для повышения точности прогнозов. Для этого были выбраны четыре метода: линейная регрессия, случайный лес, метод опорных векторов (SVR) и простой алгоритм машинного обучения с несколькими слоями (далее — многослойная модель). Эти методы были применены к двум наборам данных: df\_d1 (очищенные данные без выбросов) и df\_d2 (исходные данные с выбросами), чтобы оценить их устойчивость к аномалиям и сравнить их по качеству моделей.

Перед построением моделей данные были подготовлены с учетом временной структуры. Для учета зависимости курса доллара от его значений в предыдущие дни были добавлены так называемые лаговые признаки — значения курса за последние 7 дней, что позволяет моделям учитывать динамику изменения курса во времени. Код для добавления лагов:

def add\_lags(df, target\_column, n\_lags): # определение функции для добавления лагов

for i in range(1, n\_lags + 1): # цикл для создания лагов от 1 до n\_lags

df[f'{target\_column}\_lag{i}'] = df[target\_column].shift(i) # новый столбец с сдвигом

return df # обновленный датафрейм

df\_d1 = add\_lags(df\_d1, 'curs usd', 7) # применение функции к df\_d1 с 7 лагами

df\_d2 = add\_lags(df\_d2, 'curs usd', 7) # применение функции к df\_d2 с 7 лагами

Здесь функция add\_lags создает новые столбцы в датафрейме, где, например, curs usd\_lag1 — это значение курса доллара за предыдущий день, curs usd\_lag2 — за два дня назад и так далее. Пропущенные значения, возникшие из-за сдвига, были исключены из дальнейшего анализа.

После подготовки данных они были разделены на обучающую (тренировочную) и тестовую выборки в соотношении 80% и 20%. Обучающая выборка используется для настройки моделей, а тестовая — для проверки их качества на новых данных. Код для разделения данных:

def split\_data(df): # определение функции

train\_size = int(len(df) \* 0.8) # расчет размера обучающей выборки

train\_data = df.iloc[:train\_size] # выделение обучающих данных

test\_data = df.iloc[train\_size:] # выделение тестовых данных

X\_train = train\_data.drop('curs usd', axis=1) # исключение целевой переменной из обучающих признаков

y\_train = train\_data['curs usd'] # выделение целевой переменной для обучения

X\_test = test\_data.drop('curs usd', axis=1) # исключение целевой переменной из тестовых признаков

y\_test = test\_data['curs usd'] # выделение целевой переменной для теста

return X\_train, X\_test, y\_train, y\_test # возврат разделенных данных

X\_train\_d1, X\_test\_d1, y\_train\_d1, y\_test\_d1 = split\_data(df\_d1) # разделение на обучающую и тестовую выборки для df\_d1

X\_train\_d2, X\_test\_d2, y\_train\_d2, y\_test\_d2 = split\_data(df\_d2) # разделение на обучающую и тестовую выборки для df\_d2

Функция split\_data отделяет целевую переменную (curs usd) от остальных признаков, таких как цены на нефть, золото, ключевая ставка и т.д., и делит данные на две части. Переменные X\_train и X\_test содержат признаки для обучения и тестирования, а y\_train и y\_test — соответствующие значения курса доллара.

Следующим шагом, после разделения данных на обучающие и тестовые выборки, происходит обучение моделей. Каждая из четырех моделей (линейная регрессия, случайный лес, SVR и многослойная модель) была обучена отдельно на обоих наборах данных — df\_d1 и df\_d2.

Для обучения использовались подготовленные данные, разделенные на обучающую и тестовую выборки. Линейная регрессия ищет линейную зависимость между признаками и целевой переменной, случайный лес строит множество деревьев решений и усредняет их результаты, SVR (метод опорных векторов) подбирает оптимальную гиперплоскость для разделения данных, а многослойная модель использует несколько уровней обработки данных для выявления сложных закономерностей. Код для обучения моделей:

from sklearn.linear\_model import LinearRegression # импорт линейной регрессии

from sklearn.ensemble import RandomForestRegressor # импорт случайного леса

from sklearn.svm import SVR # импорт метода опорных векторов

from tensorflow.keras.models import Sequential # импорт модели нейросети

from tensorflow.keras.layers import Dense # импорт слоев нейросети

X\_train\_d1, X\_test\_d1, y\_train\_d1, y\_test\_d1 = split\_data(df\_d1) # разделение df\_d1

X\_train\_d2, X\_test\_d2, y\_train\_d2, y\_test\_d2 = split\_data(df\_d2) # разделение df\_d2

lr\_model\_d1 = LinearRegression() # создание линейной регрессии

lr\_model\_d1.fit(X\_train\_d1, y\_train\_d1) # обучение на df\_d1

y\_pred\_lr\_d1 = lr\_model\_d1.predict(X\_test\_d1) # прогноз на df\_d1

rf\_model\_d1 = RandomForestRegressor(n\_estimators=500, random\_state=42) # создание случайного леса

rf\_model\_d1.fit(X\_train\_d1, y\_train\_d1) # обучение на df\_d1

y\_pred\_rf\_d1 = rf\_model\_d1.predict(X\_test\_d1) # прогноз на df\_d1

svr\_model\_d1 = SVR(kernel='rbf') # создание SVR с ядром rbf

svr\_model\_d1.fit(X\_train\_d1, y\_train\_d1) # обучение на df\_d1

y\_pred\_svr\_d1 = svr\_model\_d1.predict(X\_test\_d1) # прогноз на df\_d1

nn\_model\_d1 = Sequential([ # создание нейросети

Dense(64, activation='relu', input\_shape=(X\_train\_d1.shape[1],)), # первый слой

Dense(32, activation='relu'), # второй слой

Dense(1) # выходной слой

])

nn\_model\_d1.compile(optimizer='adam', loss='mse') # компиляция модели

nn\_model\_d1.fit(X\_train\_d1, y\_train\_d1, epochs=50, batch\_size=32, verbose=0) # обучение на df\_d1

y\_pred\_nn\_d1 = nn\_model\_d1.predict(X\_test\_d1, verbose=0).flatten() # прогноз на df\_d1

lr\_model\_d2 = LinearRegression() # создание линейной регрессии

lr\_model\_d2.fit(X\_train\_d2, y\_train\_d2) # обучение на df\_d2

y\_pred\_lr\_d2 = lr\_model\_d2.predict(X\_test\_d2) # прогноз на df\_d2

rf\_model\_d2 = RandomForestRegressor(n\_estimators=500, random\_state=42) # создание случайного леса

rf\_model\_d2.fit(X\_train\_d2, y\_train\_d2) # обучение на df\_d2

y\_pred\_rf\_d2 = rf\_model\_d2.predict(X\_test\_d2) # прогноз на df\_d2

svr\_model\_d2 = SVR(kernel='rbf') # создание SVR с ядром rbf

svr\_model\_d2.fit(X\_train\_d2, y\_train\_d2) # обучение на df\_d2

y\_pred\_svr\_d2 = svr\_model\_d2.predict(X\_test\_d2) # прогноз на df\_d2

nn\_model\_d2 = Sequential([ # создание нейросети

Dense(64, activation='relu', input\_shape=(X\_train\_d2.shape[1],)), # первый слой

Dense(32, activation='relu'), # второй слой

Dense(1) # выходной слой

])

nn\_model\_d2.compile(optimizer='adam', loss='mse') # компиляция модели

nn\_model\_d2.fit(X\_train\_d2, y\_train\_d2, epochs=50, batch\_size=32, verbose=0) # обучение на df\_d2

y\_pred\_nn\_d2 = nn\_model\_d2.predict(X\_test\_d2, verbose=0).flatten() # прогноз на df\_d2

Перед обучением данные были стандартизированы (приведены к единому масштабу) с помощью StandardScaler, что особенно важно для SVR и многослойной модели, так как они чувствительны к масштабу признаков. Линейная регрессия и случайный лес обучались на исходных данных, так как они менее чувствительны к масштабированию.

Каждая из четырех моделей (линейная регрессия, случайный лес, SVR и многослойная модель) была обучена отдельно на наборах данных df\_d1 и df\_d2. Для обучения использовались следующие методы:

* Линейная регрессия — ищет линейную зависимость между признаками и целевой переменной. Реализована с использованием LinearRegression из sklearn, обучена на данных без стандартизации.
* Случайный лес — строит множество деревьев решений и усредняет их прогнозы. Реализован через RandomForestRegressor с 500 деревьями, обучен на исходных данных.
* SVR (метод опорных векторов) — подбирает гиперплоскость для предсказания значений, минимизируя ошибку. Реализован через SVR с ядром RBF, обучен на данных без предварительной стандартизации.
* Многослойная нейросеть — выявляет сложные нелинейные зависимости через слои нейронов. Реализована с помощью Sequential из Keras с тремя слоями (64, 32 нейрона и выходной), функцией активации ReLU, оптимизатором Adam и 50 эпохами обучения.

После обучения результаты каждой модели были объединены путем усреднения их прогнозов, чтобы получить итоговую комбинированную модель. Такой подход позволяет компенсировать слабые стороны отдельных методов и повысить общую точность прогноза. Код для усреднения прогноза:

y\_pred\_combined\_d1 = (y\_pred\_lr\_d1 + y\_pred\_rf\_d1 + y\_pred\_svr\_d1 + y\_pred\_nn\_d1) / 4 # усреднение прогнозов для df\_d1

y\_pred\_combined\_d2 = (y\_pred\_lr\_d2 + y\_pred\_rf\_d2 + y\_pred\_svr\_d2 + y\_pred\_nn\_d2) / 4 # усреднение прогнозов для df\_d2

Здесь y\_pred\_lr\_d1, y\_pred\_rf\_d1, y\_pred\_svr\_d1 и y\_pred\_nn\_d1 — это прогнозы, сделанные линейной регрессией, случайным лесом, методом опорных векторов и многослойной моделью на данных df\_d1. Аналогично для df\_d2. Усреднение прогнозов снижает влияние ошибок, которые могут возникнуть у отдельных моделей, и делает итоговый результат более устойчивым.

Для оценки качества комбинированной модели использовались три метрики: коэффициент детерминации R², средняя абсолютная ошибка (MAE) средняя абсолютная процентная ошибка (MAPE), так же рассматривалась среднеквадратичная ошибка (MSE), которая усиленно штрафует за большие ошибки, так как каждое отклонение возводится в квадрат. Для более наглядной интерпретации была вычислена и корень из MSE (RMSE) — эта метрика выражается в тех же единицах, что и целевая переменная, и позволяет легче понять масштаб ошибок по сравнению с MSE. Эти показатели позволяют понять, насколько хорошо модель объясняет данные и насколько точны ее прогнозы.

Код для расчета метрик:

from sklearn.metrics import r2\_score, mean\_absolute\_error, mean\_absolute\_percentage\_error, mean\_squared\_error # импорт метрик

from math import sqrt # импорт функции для RMSE

def calculate\_metrics(y\_true, predictions, model\_names): # определение функции

for y\_pred, name in zip(predictions, model\_names): # цикл по прогнозам и именам моделей

r2 = r2\_score(y\_true, y\_pred) # расчет R²

mae = mean\_absolute\_error(y\_true, y\_pred) # расчет MAE

mape = mean\_absolute\_percentage\_error(y\_true, y\_pred) \* 100 # расчет MAPE в процентах

mse = mean\_squared\_error(y\_true, y\_pred) # расчет MSE

rmse = sqrt(mse) # расчет RMSE

print(f"Метрики для {name}:") # вывод заголовка

print(f"R²: {r2:.4f}") # вывод R²

print(f"MAE: {mae:.4f}") # вывод MAE

print(f"MAPE: {mape:.2f}%") # вывод MAPE

print(f"MSE: {mse:.4f}") # вывод MSE

print(f"RMSE: {rmse:.4f}") # вывод RMSE

print() # добавление пустой строки

Сначала были рассчитаны метрики для всех моделей на df\_d1.

* Линейная регрессия: R²: 0.9835, MAE: 1.4538, MAPE: 1.73%, MSE: 3.2815, RMSE: 1.8115;
* Случайный лес: R²: 0.8941, MAE: 3.0794, MAPE: 3.52%, MSE: 21.0038, RMSE: 4.5830;
* SVR: R²: -0.4590, MAE: 13.2697, MAPE: 15.17%, MSE: 289.3541, RMSE: 17.0104;
* Нейронная сеть: R²: 0.7951, MAE: 4.6694, MAPE: 5.28%, MSE: 40.6338, RMSE: 6.3745;
* Комбинированная модель: R²: 0.8125, MAE: 3.5086, MAPE: 3.88%, MSE: 37.1911, RMSE: 6.0985.

Затем метрики были рассчитаны для всех моделей на df\_d2:

* Линейная регрессия: R²: 0.9838, MAE: 1.4036, MAPE: 1.67%, MSE: 3.2034, RMSE: 1.7898;
* Случайный лес: R²: 0.9642, MAE: 2.0310, MAPE: 2.34%, MSE: 7.0985, RMSE: 2.6643;
* SVR: R²: -0.3201, MAE: 12.9852, MAPE: 14.91%, MSE: 261.8160, RMSE: 16.1807;
* Нейронная сеть: R²: 0.7811, MAE: 5.3425, MAPE: 6.82%, MSE: 43.4044, RMSE: 6.5882;
* Комбинированная модель: R²: 0.9574, MAE: 2.3349, MAPE: 2.89%, MSE: 8.4473, RMSE: 2.9064.

Анализ метрик показывает, что линейная регрессия демонстрирует наилучшую точность среди всех моделей на обоих наборах данных, с R² близким к 1 (0.9835 для df\_d1 и 0.9838 для df\_d2) и минимальными значениями MAE (1.4538 и 1.4036 соответственно). Это указывает на высокую способность модели объяснять вариацию данных и ее точность в прогнозировании.

Случайный лес также показывает хорошие результаты: R² составляет 0.8941 для df\_d1 и 0.9642 для df\_d2, а MAPE — 3.52% и 2.34% соответственно. Это делает его второй по точности моделью, особенно на df\_d2, где он лучше справляется с выбросами.

SVR демонстрирует низкую эффективность с отрицательным R² (-0.4590 для df\_d1 и -0.3201 для df\_d2) и высокими значениями ошибок (MAPE: 15.17% и 14.91%), что указывает на его непригодность для данной задачи.

Нейронная сеть также показывает средние результаты: R² составляет 0.4937 для df\_d1 и лишь 0.1425 для df\_d2, с MAPE 6.94% и 11.95% соответственно, что говорит о недостаточной точности и переобучении.

Комбинированная модель, объединяющая прогнозы всех методов, демонстрирует высокую эффективность, особенно на df\_d2, где R² достигает 0.9574, а MAPE — 2.89%, что близко к результатам случайного леса (R²: 0.9642). На df\_d1 модель также показывает хорошие результаты с R² = 0.8125 и MAPE = 3.88%, хотя ее точность немного ниже, что может быть связано с влиянием менее точных моделей, таких как SVR и нейронная сеть.

Сравнение показывает, что данные df\_d2 (с выбросами) обеспечивают лучшие результаты для комбинированной модели по всем метрикам: MAPE (2.89% против 3.88%), RMSE (2.9064 против 6.0985) и R² (0.9574 против 0.8125). Это может быть связано с сохранением рыночных аномалий, улучшающих предсказательную способность. Для набора данных df\_d1 модель более стабильна, но менее адаптивна к резким изменениям.

## 2.5. Выводы к главе 2

Вторая глава была посвящена анализу предметной области, сбору, предобработке и анализу данных, а также разработке и оценке регрессионных моделей для прогнозирования курса доллара США к рублю.

В рамках анализа предметной области была подтверждена сложность прогнозирования валютных курсов из-за их высокой волатильности и влияния множества факторов, включая экономические, политические и рыночные. Традиционные статистические методы, такие как ARIMA, показали ограниченную эффективность в условиях аномалий, что обосновало использование методов машинного обучения, способных учитывать нелинейные зависимости и выбросы.

Для построения моделей были собраны исторические данные о курсе доллара США к рублю с сайта Центрального банка РФ, дополненные макроэкономическими индикаторами: ценами на нефть Brent, ключевой ставкой ЦБ РФ, уровнем инфляции, ценами на драгоценные металлы и курсом евро. Данные были разделены на два набора: df\_d1 (очищенные от выбросов) и df\_d2 (с сохраненными выбросами), чтобы оценить устойчивость моделей к аномалиям.

Предобработка данных включала хронологическое упорядочивание временного ряда, заполнение пропусков в макроэкономических показателях методом обратного переноса и линейной интерполяцией, а также устранение выбросов в df\_d1 с использованием метода межквартильного размаха (IQR). Для df\_d1 применялись различные подходы к обработке выбросов (замена на медиану для золота, усечение для серебра, палладия, ключевой ставки и инфляции), что позволило создать более стабильный набор данных. Анализ корреляционных матриц показал, что очищенные данные (df\_d1) обеспечивают более логичные и устойчивые взаимосвязи между признаками, тогда как выбросы в df\_d2 усиливают корреляции, иногда искусственно.

На этапе машинного обучения были разработаны и протестированы четыре модели: линейная регрессия, случайный лес, метод опорных векторов (SVR) и нейронная сеть, а также комбинированная модель, усредняющая их прогнозы. Для учета временной структуры добавлены лаговые признаки (7 дней). Модели обучались на df\_d1 и df\_d2, а их качество оценивалось по метрикам R², MAE, MAPE, MSE и RMSE. Результаты показали, что линейная регрессия является наиболее точной (R²: 0.9835 для df\_d1, 0.9838 для df\_d2; MAPE: 1.73% и 1.67%), демонстрируя высокую способность объяснять вариацию данных. Случайный лес также показал хорошие результаты, особенно на df\_d2 (R²: 0.9642, MAPE: 2.34%), благодаря устойчивости к выбросам. SVR и нейронная сеть оказались наименее эффективными (R² для SVR: -0.4590 и -0.3201; для нейронной сети: 0.4937 и 0.1425), что указывает на их низкую пригодность для данной задачи без дополнительной настройки. Комбинированная модель продемонстрировала высокую эффективность, особенно на df\_d2 (R²: 0.9574, MAPE: 2.89%), что близко к результатам случайного леса, и хорошие результаты на df\_d1 (R²: 0.8125, MAPE: 3.88%).

Сравнение наборов данных выявило, что df\_d2 (с выбросами) обеспечивает более высокую точность для комбинированной модели и случайного леса, вероятно, за счет сохранения рыночных аномалий, которые содержат важные сигналы о динамике курса. Это подтверждает, что выбросы, такие как резкие скачки курса, часто отражают значимые рыночные события (например, изменения в экономической политике или геополитические факторы), которые усиливают предсказательную способность моделей, особенно случайного леса, способного улавливать нелинейные зависимости. В то же время df\_d1 обеспечивает более стабильные, но менее адаптивные прогнозы, что может быть полезно для долгосрочного анализа.

Анализ подтвердил целесообразность использования комбинированной модели на данных df\_d2 для краткосрочного прогнозирования в условиях волатильности, а также высокую эффективность линейной регрессии как универсального решения. Полученные результаты и подготовленные данные создают прочную основу для интеграции модели в Telegram-бот, что будет рассмотрено в следующей главе.

Код, который использовался для анализа данных, представлен в Приложении А.

# РАЗРАБОТКА TELEGRAM-БОТА

## 3.1. Обоснование выбора платформы Telegram

Telegram считается платформой с одним из самых расширенных контингентов аудитории в мире, он обладает значительным охватом аудитории 1 миллиард активных пользователей по состоянию на 2025 год, что делает его подходящим инструментом для распространения аналитических данных [14].

Выбор платформы обусловлен наличием открытого API, который поддерживает асинхронное взаимодействие через библиотеку Aiogram, упрощающую создание и тестирование ботов. Так же Telegram обеспечивает кроссплатформенную совместимость, включая поддержку iOS, Android, Windows, MacOS и веб-интерфейса, что исключает необходимость в дополнительных клиентских приложениях. Безопасность платформы обеспечивается использованием шифрования сообщений, а поддержка высоких нагрузок делает ее масштабируемой для потенциального роста числа пользователей. Эти особенности делают Telegram оптимальной средой для интеграции ИИС, разработанной в рамках данной работы [14].

## 3.2. Используемые технологии и библиотеки

Для реализации бота был выбран язык программирования Python версии 3.11.9, который обеспечивает высокую совместимость с современными библиотеками для анализа данных, нейронных сетей и машинного обучения. Данный выбор обусловлен широким распространением Python в научных исследованиях и наличием обширной экосистемы инструментов, необходимых для обработки временных рядов и построения прогнозных моделей.

Для реализации функциональных и аналитических возможностей бота были задействованы следующие программные библиотеки на языке Python:

* Aiogram (версия 3.20.0): Библиотека Aiogram применялась для реализации асинхронного взаимодействия с Telegram API. Асинхронность достигается через использование asyncio и async/await синтаксиса, и это позволяет минимизировать время отклика бота при обработке запросов [15].
* Pandas и Numpy: Библиотека Pandas применялась для работы с данными в формате таблиц, включая загрузку, очистку и преобразование данных из файла df\_d2.csv. NumPy обеспечивала поддержку массивов для численных вычислений, необходимых при масштабировании признаков и обучении моделей [16-17].
* Matplotlib и Seaborn: Эти библиотеки использовались для визуализации корреляционной матрицы. Matplotlib предоставляла базовые возможности построения графиков, а Seaborn улучшала читаемость матрицы с помощью тепловых карт (seaborn.heatmap). Изображение сохранялось как correlation\_matrix.png для последующей отправки пользователю [18-19].
* Scikit-learn: Использовалась для реализации линейной регрессии, случайного леса и метода опорных векторов (SVR), а также для нормализации данных с помощью StandardScaler и расчета метрик качества модели (R², MAE, MAPE, RMSE). Объект нормализации сохранялся в файл scaler.joblib для повторного использования [20].
* Tensorflow: блиотека применялась для создания нейронной сети, входящей в состав комбинированной модели прогноза, включающей четыре метода: линейную регрессию, случайный лес, SVR и нейронную сеть. Нейронная сеть была реализована как многослойный перцептрон (MLP) с архитектурой, включающей входной слой (соответствующий количеству признаков в df\_d2, таких как курс доллара, цены на золото и нефть), два скрытых слоя по 64 нейрона с функцией активации ReLU (функция активации, позволяющая сети обучаться нелинейным зависимостям) и выходной слой с одним нейроном для предсказания курса доллара. Обучение проводилось с использованием оптимизатора Adam (алгоритм оптимизации параметров сети) с шагом обучения 0,001 и функцией потерь MSE (среднеквадратичная ошибка) на 100 эпохах (цикл обучения на всем наборе данных) с размером подвыборки данных, используемой для одного шага оптимизации, равном 32. Сохранение весов модели осуществлялось в файл model.h5 для последующего использования в боте [21].
* Joblib: Библиотека применялась для сохранения и загрузки обученных моделей и объекта нормализации [22].
* Python-dotenv: Обеспечивала безопасное хранение конфиденциальных данных, таких как токен Telegram-бота в файле .env, исключая их попадание в систему контроля версий (Git) [23].

## 3.3. Архитектура приложения

Архитектура бота разработана с учетом модульности, что упрощает его сопровождение и масштабирование. Проект разделен на несколько файлов и модулей, каждый из которых выполняет специфическую функцию. Структура позволяет четко разграничить логику обработки данных, прогнозов и взаимодействия с пользователем.

В боте используются следующие модули:

* main.py: Основной модуль, выполняющий роль точки входа в приложение. В этом файле создается экземпляр бота с использованием токена, загруженного из файла .env с помощью библиотеки python-dotenv. Также здесь инициализируется диспетчер (Dispatcher), который отвечает за маршрутизацию входящих сообщений к соответствующим обработчикам. Запуск бота осуществляется через асинхронный цикл executor.start\_polling, обеспечивающий непрерывную обработку запросов.
* handlers/commands.py: Модуль, содержащий функции-обработчики для всех команд пользователя. Каждая команда реализована как асинхронная функция с декоратором @dp.message\_handler(commands=['команда']) из библиотеки Aiogram. Например: Обработчик /start формирует приветственное сообщение с перечислением доступных команд, обработчик /predict вызывает функцию predict\_combined из модуля models, передавая ей количество дней, выбранное пользователем, а обработчик /correlation сначала отправляет описание признаков, затем генерирует и отправляет корреляционную матрицу. Этот модуль также обрабатывает исключения, связанные с некорректными входными данными, например, при отсутствии файла данных.
* keyboards/reply.py: Модуль, отвечающий за создание интерактивной клавиатуры для команды /predict. Клавиатура формируется с использованием класса ReplyKeyboardMarkup из aiogram, где добавляются кнопки с числами от 1 до 7. После выбора пользователем количества дней клавиатура сохраняется в интерфейсе для повторного использования, что реализовано через параметр resize\_keyboard=True. Если команда не требует клавиатуры (например, /help), она скрывается с помощью ReplyKeyboardRemove.
* models/models.py: Модуль, реализующий логику прогнозирования и оценки качества модели. Функция predict\_combined комбинирует прогнозы четырех моделей (линейная регрессия, случайный лес, SVR, нейронная сеть), усредняя их результаты. Модели загружаются из сохраненных файлов с помощью joblib, а входные данные нормализуются с использованием scaler.joblib. Для итеративного прогноза функция обновляет лаговые признаки после каждого шага. Функция calculate\_metrics вычисляет метрики качества (R², MAE, MAPE, MSE, RMSE) на основе последних 100 записей из latest\_data.csv, используя функции из scikit-learn. Функция load\_data загружает данные из latest\_data.csv с помощью pandas, выполняя проверку целостности данных.
* utils/data\_preparation.py: Модуль предобработки данных, включающий несколько функций. Функция load\_data загружает данные из df\_d2.csv, содержащего исторические данные о курсе доллара, макроэкономические показатели (ключевая ставка, инфляция, цены на нефть Brent, золото, серебро, палладий) и курс евро. Функция add\_lags добавляет лаговые признаки (curs usd\_lag1–curs usd\_lag7) на основе столбца с курсом доллара, используя метод shift из pandas. Функция scale\_data выполняет нормализацию данных с помощью StandardScaler из scikit-learn, сохраняя scaler для последующего использования. Функция save\_data сохраняет обработанные данные в latest\_data.csv для использования в реальном времени.
* .env и .gitignore: Файл .env хранит конфиденциальные данные, такие как токен Telegram, а .gitignore исключает временные файлы и конфиденциальные данные из системы контроля версий Git.

Модули взаимодействуют следующим образом: main.py запускает приложение и передает управление диспетчеру, который направляет команды в handlers/commands.py. Обработчики команд используют функции из models/models.py для выполнения прогнозов и расчета метрик, а также вызывают клавиатуру из keyboards/reply.py при необходимости. Данные загружаются и обрабатываются с помощью utils/data\_preparation.py, что обеспечивает единообразие входных данных для всех операций.

Полная структура проекта (бота) представлена на рисунке 10.

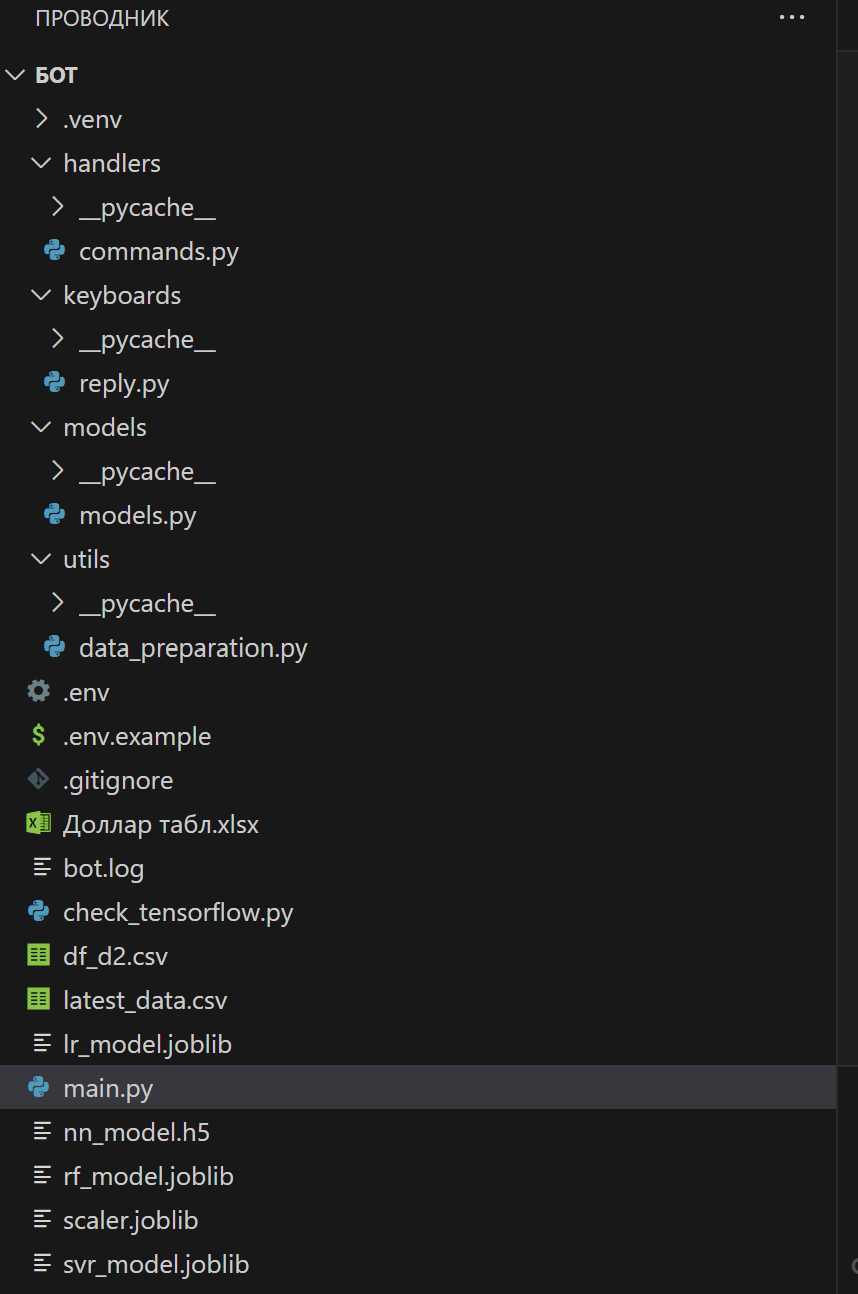


Рис. 10. Структура Telegram-бота в Visual Studio Code

Рассмотрим рисунок 10. Данный рисунок представляет собой структуру проекта Telegram-бота, отображающий иерархию файлов и папок в файловой системе, содержащей виртуальную среду (.venv) для изоляции зависимостей, папки handlers (с файлом commands.py для обработчиков команд), keyboards (с файлом reply.py для интерактивных клавиатур), models (с файлом models.py для логики прогнозирования), utils (с файлом data\_preparation.py для предобработки данных), а также служебные папки \_\_pycache\_\_, файлы .env для конфиденциальных данных, .gitignore для исключения из контроля версий, исходный файл Доллар табл.xlsx, скрипт check\_tensorflow.py, данные df\_d2.csv и latest\_data.csv, сохраненные модели (lr\_model.joblib, nn\_model.h5, rf\_model.joblib, scaler.joblib, svr\_model.joblib) и основной файл main.py. Эта структура демонстрирует модульность проекта, где каждый файл и папка имеют четко определенную роль.

## 3.4. Функциональные возможности и интерфейс

Разработанный Telegram-бот предоставляет набор функций, направленных на анализ и прогнозирование курса доллара США к рублю, а также на представление аналитической информации в удобной форме. Каждая команда бота проработана так, чтобы отвечать задачам исследования, при этом оставаясь простой в использовании. Реализация интерфейса включает текстовые команды и интерактивную клавиатуру, что позволяет оперативно получать результаты.

Команда /start выполняет начальную инициализацию взаимодействия. После её ввода пользователю отправляется приветствие с списком доступных команд. Вывод данной команды продемонстрирован на рисунке 11.

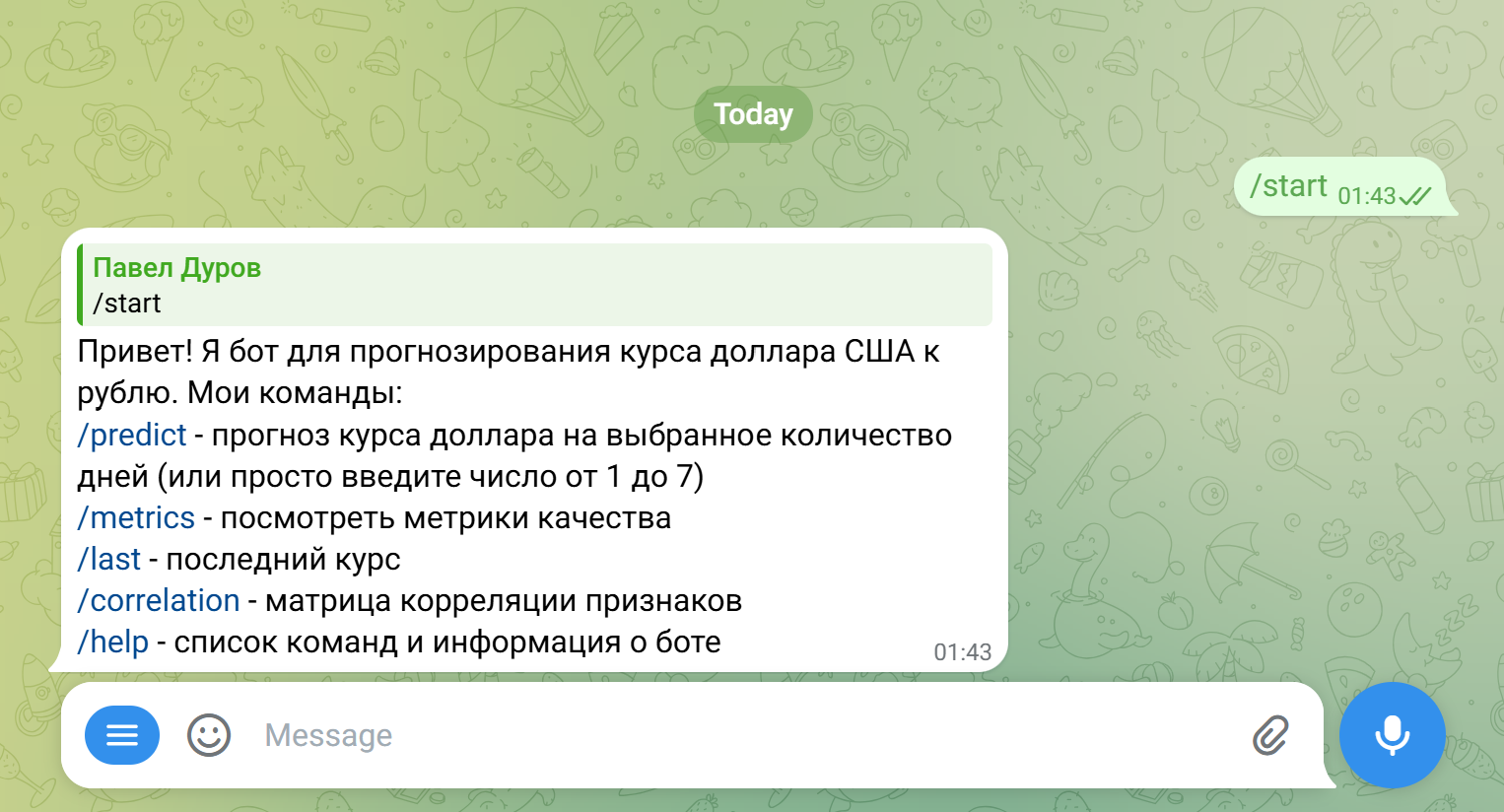


Рис. 11. Работа ТБ при отправке команды /start

Команда /help создана для информирования о возможностях бота. Она формирует сообщение, где подробно описано назначение каждой команды, например, «/predict — спрогнозировать курс доллара на 1–7 дней». Так же, помимо команд, формируется описание бота, какие задачи он выполняет, какие данные и метрики качества модели использует. Вывод такой команды запечатлен на рисунке 12.

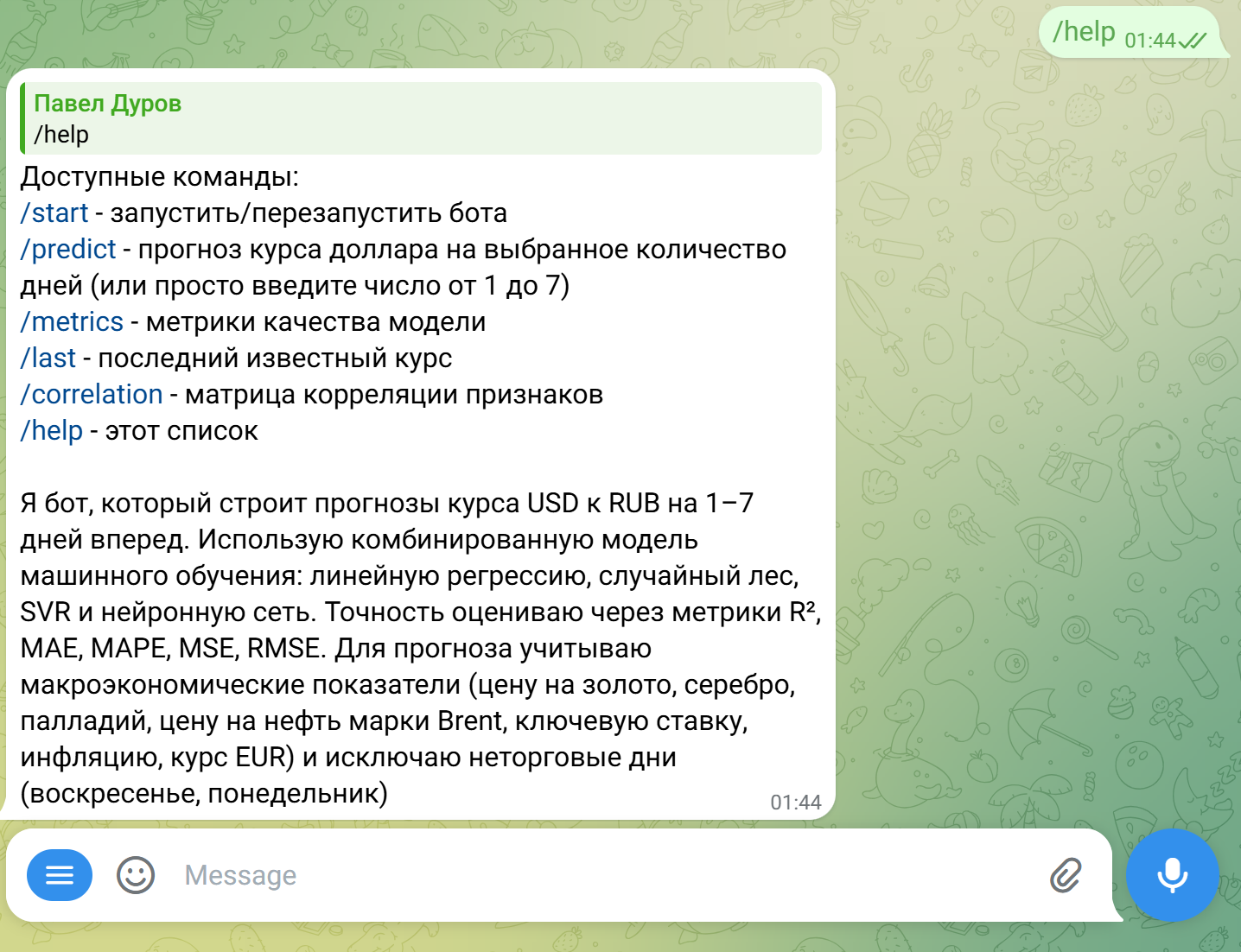


Рис. 12. Работа ТБ при отправке команды /help

Команда /last извлекает последнюю доступную запись курса доллара из файла latest\_data.csv, который содержит актуальные данные, подготовленные в процессе работы. Результат отправляется в формате: «Последний курс доллара на 2025-01-22: 92.34 руб.». Вывод бота при такой команде на рисунке 13.

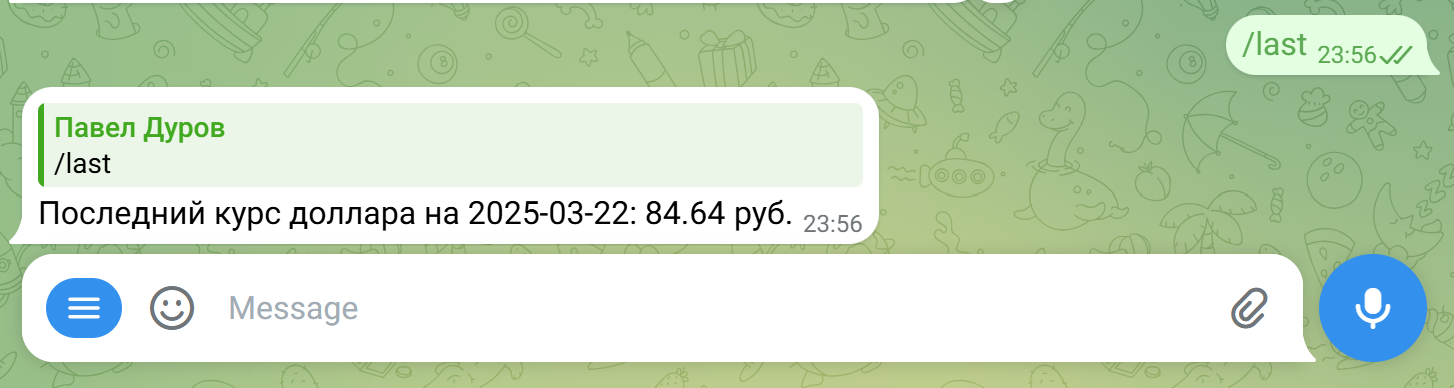


Рис. 13. Работа ТБ при отправке команды /help

По рисунку видно, что последнее значение курса доллара было на момент тестирования бота, 22 марта 2025.

Команда /metrics предоставляет информацию о качестве комбинированной модели, применяемой для прогнозов. На рисунке 14 изображен результат работы такой команды.

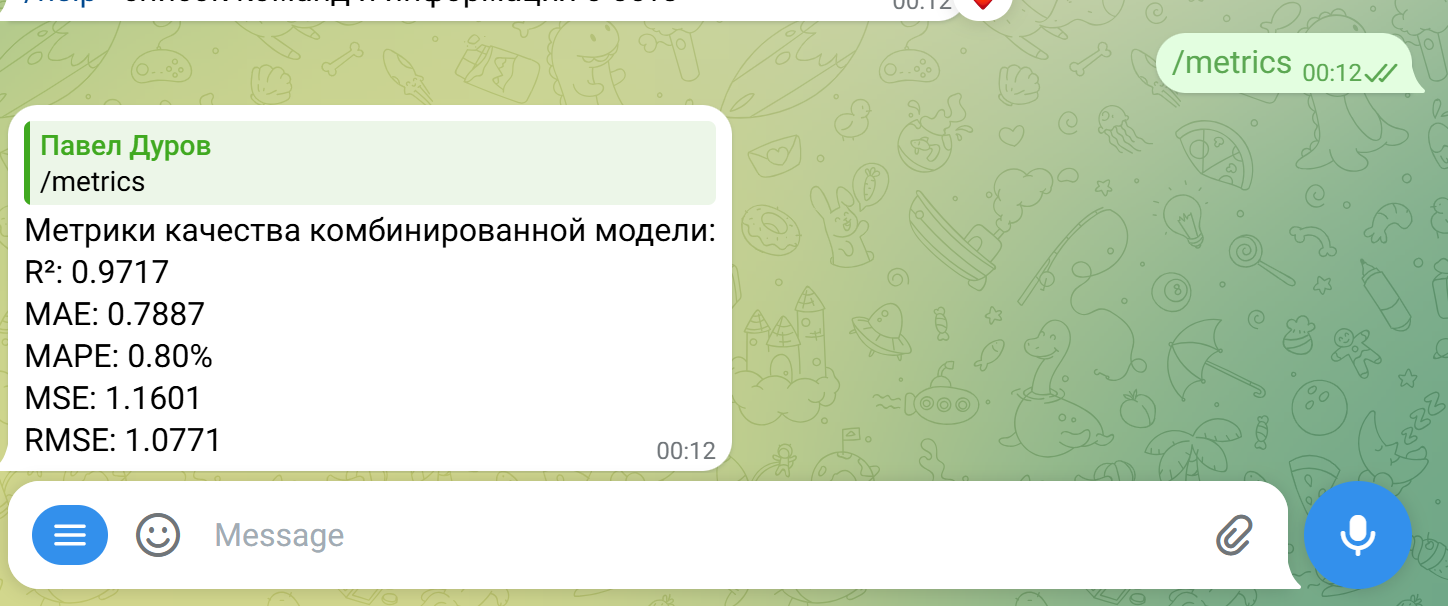


Рис. 14. Работа ТБ при отправке команды /metrics

На основе последних 100 записей из latest\_data.csv рассчитываются ключевые метрики: коэффициент детерминации R² составляет 0.9717, средняя абсолютная ошибка MAE — 0.7887, средняя абсолютная процентная ошибка MAPE — 0.8%, среднеквадратичная ошибка MSE — 1.1601, а корень среднеквадратичной ошибки RMSE — 1.0771. Эти значения демонстрируют высокую точность модели.

Команда /predict представляет собой центральную функцию бота, позволяющую выполнять прогноз курса доллара на период от 1 до 7 дней. На рисунке 15 вывод после ввода данной команды.

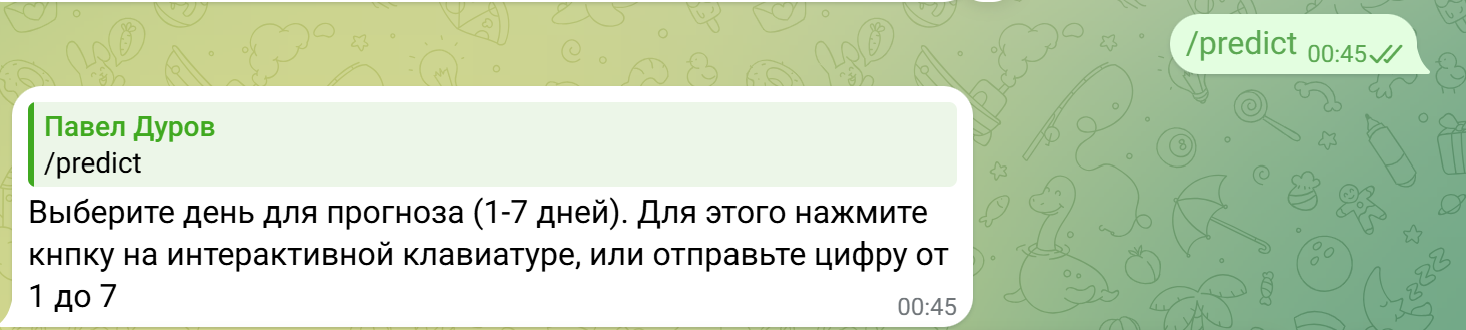


Рис. 15. Работа ТБ при отправке команды /predict

После ввода команды пользователю отображается интерактивная клавиатура, созданная с использованием класса ReplyKeyboardMarkup из библиотеки Aiogram. Клавиатура содержит кнопки с числами от 1 до 7, позволяет выбрать желаемый период. Интерактивная клавиатура представлена на рисунке 16.

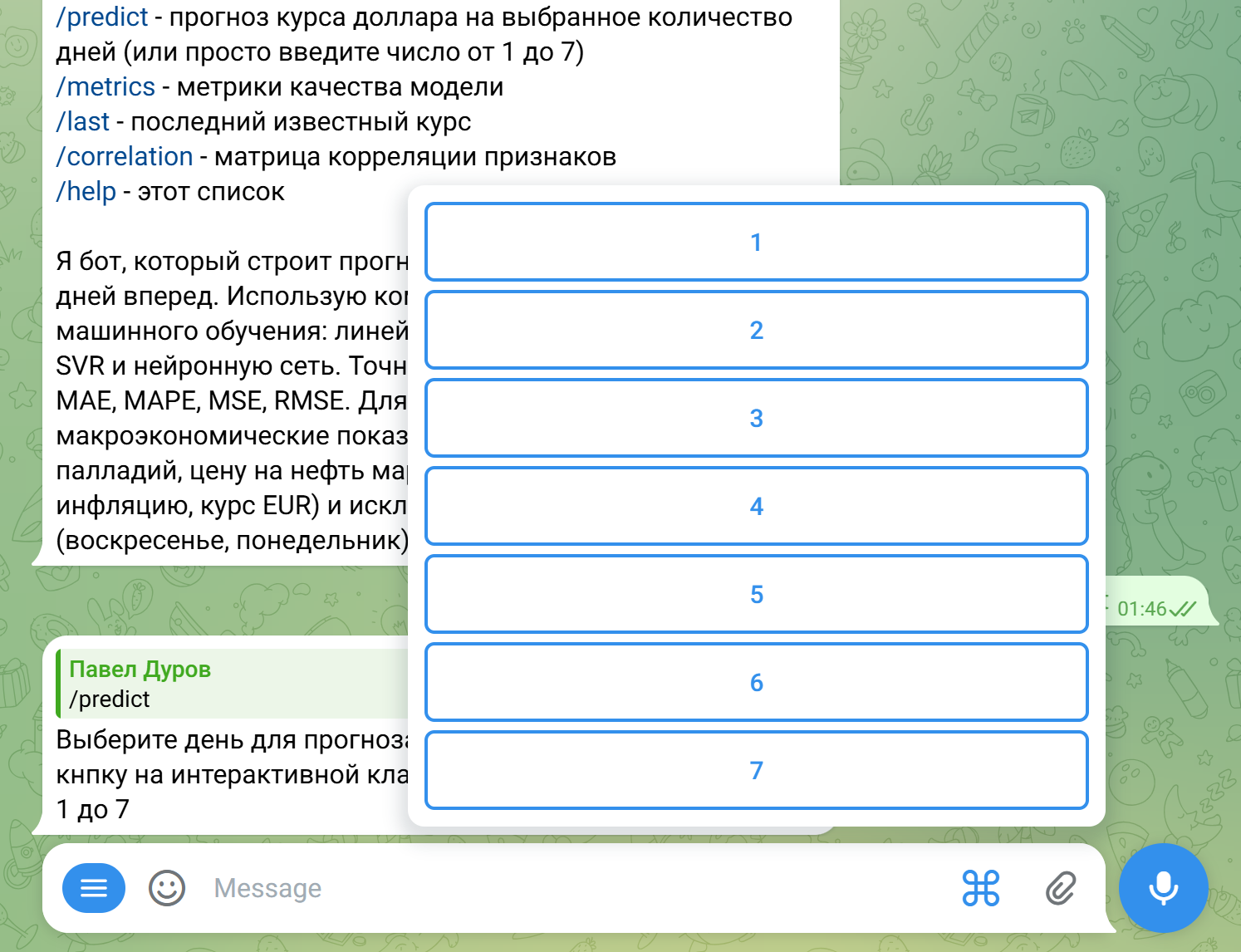


Рис. 16. Интерактивная клавиатура с выбором прогноза на 1-7 дней

Выбор количества дней приводит к запуску процесса прогнозирования, который проходит в несколько этапов. Сначала проверяется, является ли день торговым: воскресенье и понедельник исключаются с использованием функции is\_trading\_day, основанной на анализе дня недели через модуль datetime. Если день неторговый, отправляется сообщение: «На 2025-03-23 не было торгов». Пример такого вывода на рисунке 17.



Рис. 17. Вывод прогноза на те дни, когда торгов нет (воскресенье и понедельник)

Для торговых дней вызывается функция predict\_combined, которая загружает последние данные из latest\_data.csv, нормализует их с использованием сохраненного объекта scaler.joblib, а затем усредняет прогнозы четырех моделей: линейной регрессии, случайного леса, метода опорных векторов (SVR) и нейронной сети. После каждого шага прогноза обновляются лаговые признаки: значение курса за предыдущий день (curs usd\_lag1) заменяется новым прогнозом, а остальные признаки сдвигаются. Пример результата: «Прогноз на 2025-01-25: 92.45 руб». Клавиатура остается активной для повторного выбора. На рисунке 18 продемонстрирован вывод прогноза доллара на максимальное количество дней - 7 дней.

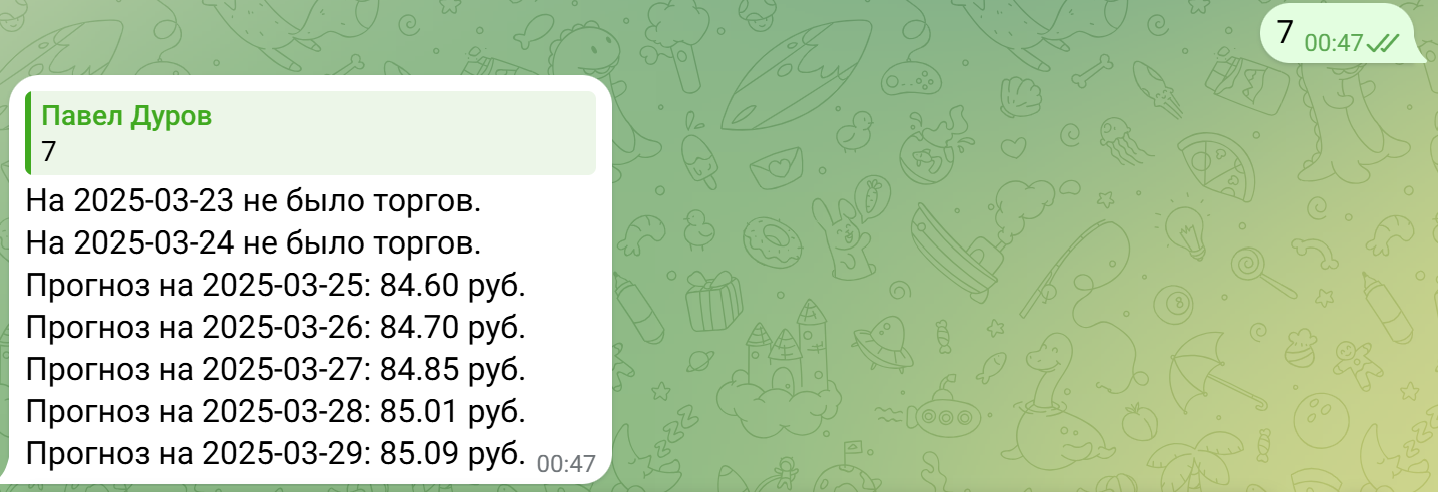


Рис. 18. Вывод прогноза курса доллара на 7 дней

Как видно по рисунку 18, бот выводит прогноз валюты на заданное количество дней, но при этом понедельники, воскресения и праздничные дни не учитываются по причине отсутствия торгов в эти дни.

Так же пользователь может ввести цифру от 1 до 7 напрямую в текстовом поле, с вводом команды /predict и без, и это также инициирует прогноз. Такой метод придает боту универсальности и обеспечивает более доступное взаимодействие с пользователем, вне зависимости от индивидуальных предпочтений работы с интерфейсом. В то же время, тем, для кого характерно самое отдаленное число операций в максимально короткое время перерасчета, такое решение чаще всего оказывается оптимальным. К тому же, такая реализация не препятствует использованию команд. На рисунке 19 представлен принцип такой работы, сначала пользователь вводит число в текстовом поле без команды /predict, а затем использует эту команду и так же вводит число.

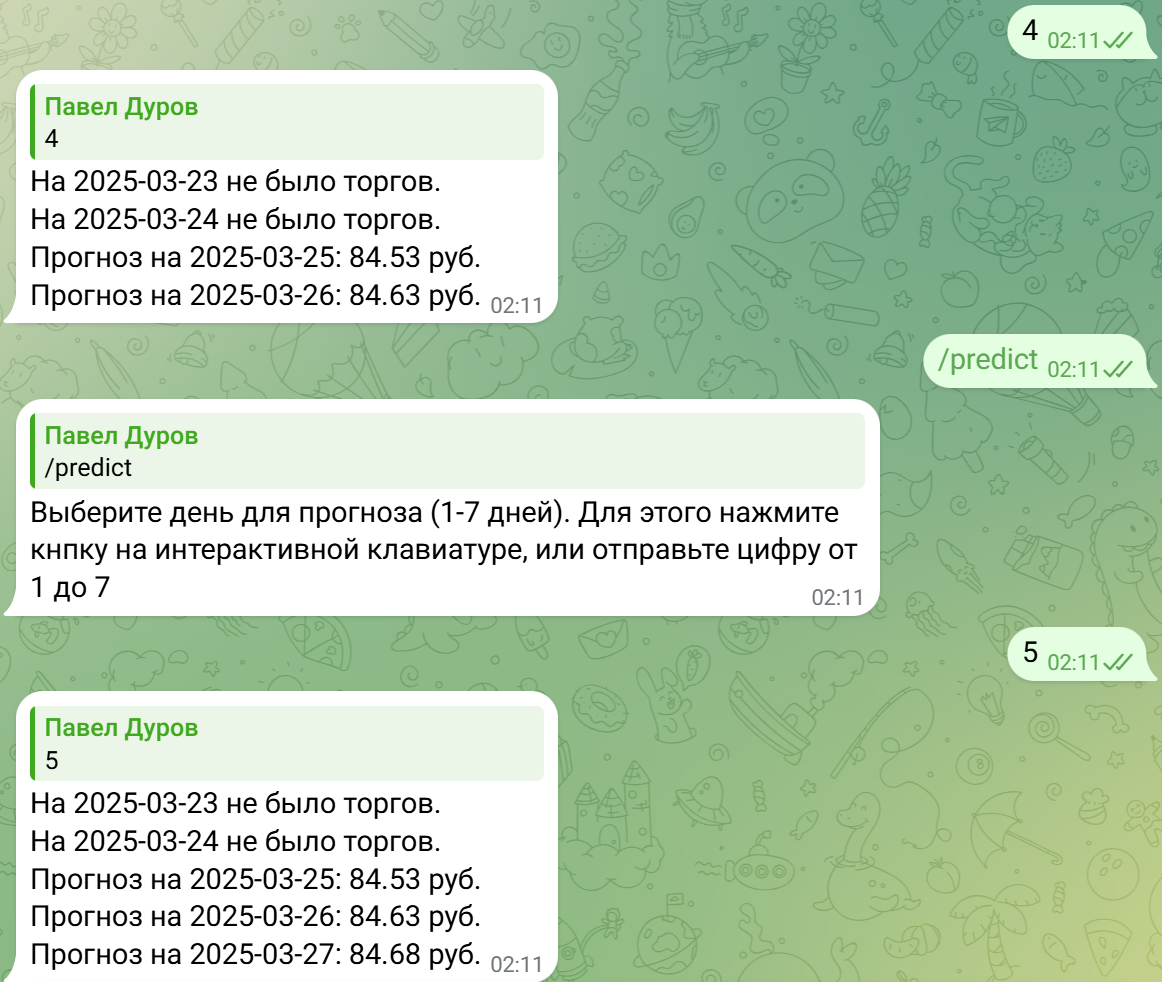


Рис. 19. Пример взаимодействия пользователя с прогнозным функционалом через прямой ввод числа и команду /predict

Если пользователь вводит символы, отличные от команд или цифр 1–7 (включая числа меньше 1 или больше 7), бот выводит сообщение «Я знаю только команды и числа от 1 до 7». Работу бота при таком раскладе демонстрирует рисунок 20.

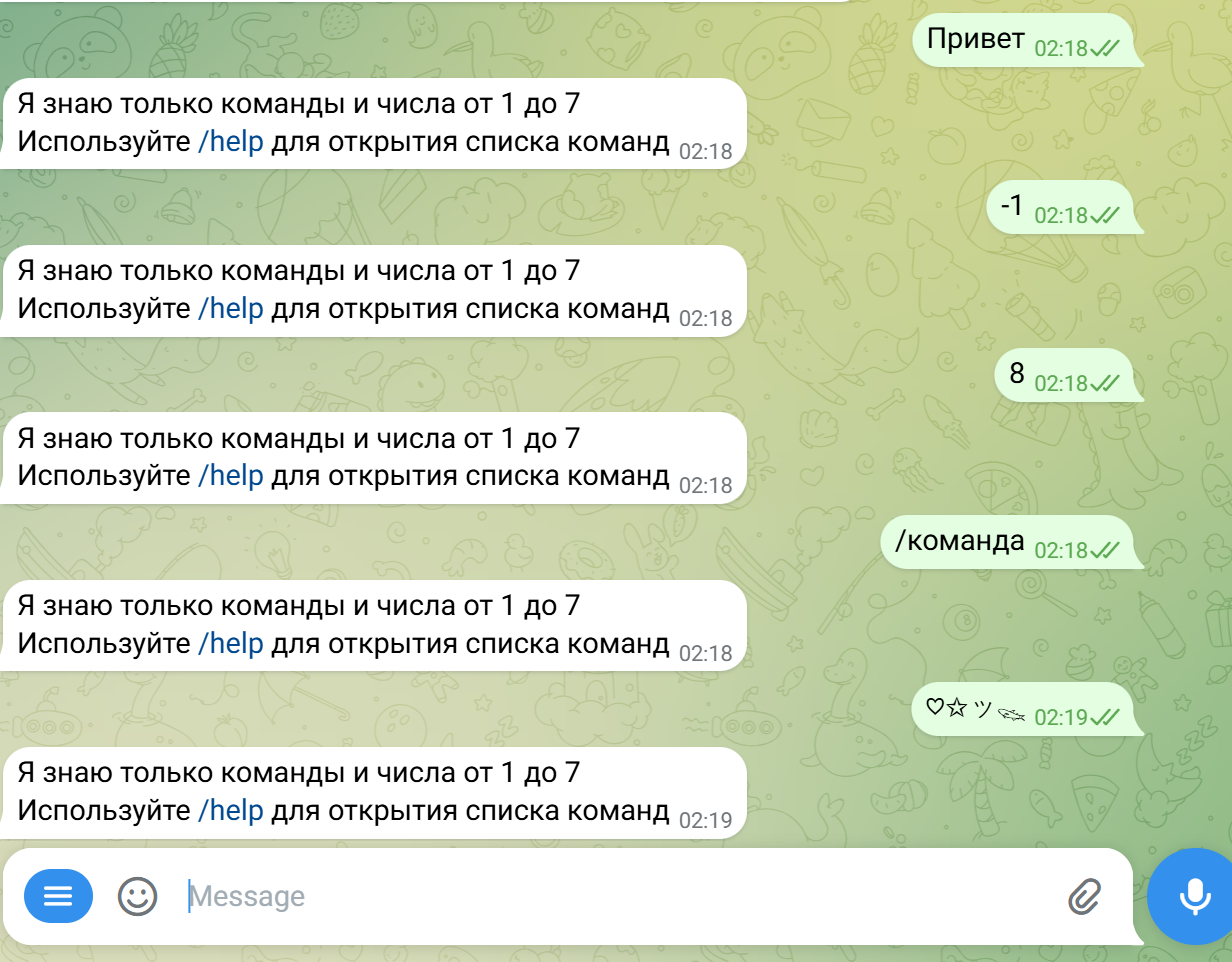


Рис. 20. Исключительные ситуации, реакция бота на некорректный ввод (числа <1 и >7, случайные слова и символы)

Команда /correlation отвечает за предоставление аналитической информации о взаимосвязи признаков. Сначала пользователю отправляется текстовое описание используемых признаков: курс доллара США к рублю, цены на золото, серебро, палладий, ключевая ставка Центрального банка РФ, уровень инфляции, курс евро к рублю, цена нефти Brent. Далее генерируется корреляционная матрица на основе данных из файла df\_d2.csv. Для повышения читаемости из анализа исключены лаговые признаки (curs usd\_lag1–curs usd\_lag7). Построение матрицы осуществляется с применением функции seaborn.heatmap, которая отображает коэффициенты корреляции в виде цветовой шкалы. После генерации изображение сохраняется в файл correlation\_matrix.png, отправляется пользователю с помощью объекта FSInputFile из aiogram и затем удаляется с использованием os.remove для освобождения пространства. Этот процесс оказался сложным на этапе реализации, изначально возникала ошибка при отправке изображения из-за использования устаревшего метода InputFile, и это потребовало перехода на FSInputFile с дополнительной проверкой существования файла. Вывод корреляционной матрицы с описанием признаков на рисунке 21.

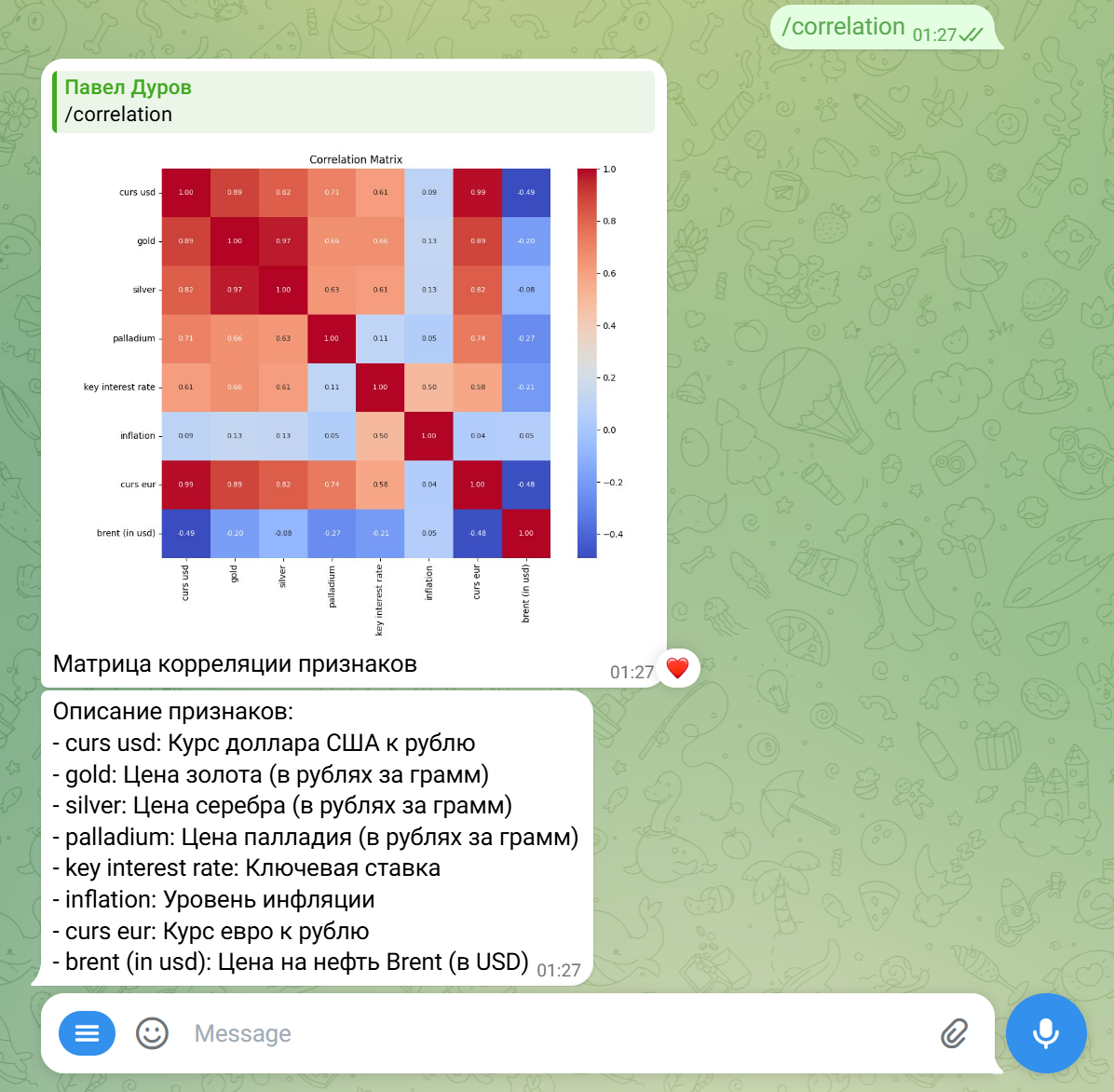


Рис. 21. Работа ТБ при отправке команды / correlation

# 3.5. Выводы к главе 3.

В третьей главе данной работы были рассмотрены этапы проектирования и разработки Telegram-бота, интегрирующего интеллектуальную информационную систему для прогнозирования курса доллара США к рублю. Обоснован выбор платформы Telegram благодаря её широкой аудитории, кроссплатформенности, безопасности и поддержке асинхронного API через библиотеку Aiogram. Описаны используемые технологии, включая Python и библиотеки для анализа данных, машинного обучения и визуализации, а также представлена модульная архитектура бота, обеспечивающая его масштабируемость и удобство сопровождения.

Разработанный бот предоставляет функционал для прогнозирования курса на 1–7 дней, анализа метрик качества модели (R², MAE, MAPE, MSE, RMSE) и корреляций признаков, что позволяет пользователям получать точные прогнозы и аналитическую информацию в удобной форме. Тестирование подтвердило стабильность работы бота, корректность обработки данных и удобство интерфейса, включая интерактивную клавиатуру и обработку некорректного ввода.

Полученные результаты демонстрируют успешную интеграцию комбинированной регрессионной модели в Telegram-бот, что делает систему доступной для широкого круга пользователей. В дальнейшем возможно расширение функционала, например, добавление прогнозов для других валютных пар или интеграция дополнительных источников данных.

Код программы ТБ по модулям приведен в Приложении Б.

# ЗАКЛЮЧЕНИЕ

В рамках данной курсовой работы была успешно выполнена задача проектирования и разработки интеллектуальной информационной системы (ИИС) для прогнозирования курса доллара США к рублю с использованием методов машинного обучения. Разработанная система интегрирована в Telegram-бот, что обеспечивает удобный доступ к прогнозам и аналитической информации для пользователей.

На этапе анализа предметной области подтверждена актуальность задачи прогнозирования курсов валют в условиях их высокой волатильности, обусловленной экономическими, политическими и рыночными факторами. Проведенный обзор традиционных статистических методов (ARIMA, экспоненциальное сглаживание) и современных подходов машинного обучения (линейная регрессия, случайный лес, SVR, нейронные сети) показал, что комбинированные модели лучше справляются с нелинейными зависимостями и аномалиями в данных.

Для реализации ИИС были собраны исторические данные о курсе доллара США к рублю с сайта Центрального банка РФ, дополненные макроэкономическими показателями: ценами на нефть Brent, ключевой ставкой ЦБ РФ, уровнем инфляции, ценами на драгоценные металлы и курсом евро. Данные были предобработаны с учетом хронологического порядка, заполнения пропусков и обработки выбросов. Сравнительный анализ двух наборов данных — df\_d1 (очищенные данные) и df\_d2 (с выбросами) — показал, что сохранение выбросов в df\_d2 улучшает предсказательную способность моделей, отражая значимые рыночные события.

Разработанная комбинированная регрессионная модель, объединяющая линейную регрессию, случайный лес, SVR и нейронную сеть, продемонстрировала высокую точность: на df\_d2 коэффициент детерминации R² составил 0.9574, MAPE — 2.89%, RMSE — 2.9064. Линейная регрессия показала наилучшие результаты среди отдельных моделей (R²: 0.9838, MAPE: 1.67%), что подтверждает её эффективность для данной задачи. Модель интегрирована в Telegram-бот, предоставляющий функционал прогнозирования курса на 1–7 дней, анализа метрик качества и корреляций признаков. Тестирование бота подтвердило его стабильность и удобство использования.

Результаты работы имеют практическую значимость для инвесторов, аналитиков и пользователей, стремящихся минимизировать финансовые риски. В будущем возможно расширение функционала бота, включая прогнозы для других валютных пар и интеграцию дополнительных факторов, таких как новостные данные.

# СПИСОК ИСТОЧНИКОВ

1. Волковинский А. Ю. Система прогнозирования финансовых показателей региона: магистерская диссертация / А. Ю. Волковинский. — Тюмень: [б. и.], 2023. — 52 с.
2. Хасиятуллов М. Г. Научный аспект № 4 2024 / М. Г. Хасиятуллов. — Самара: Аспект, 2024. — 140 с.
3. Данные по ключевой ставке и инфляции [Электронный ресурс] // Центральный банк Российской Федерации. — URL: https://www.cbr.ru/hd\_base/infl/ (дата обращения: 20.03.2025).
4. Курсы валют [Электронный ресурс] // Центральный банк Российской Федерации. — URL: https://www.cbr.ru/currency\_base/dynamics/?UniDbQuery.Posted=True&UniDbQuery.so=1&UniDbQuery.mode=1&UniDbQuery.date\_req1=&UniDbQuery.date\_req2=&UniDbQuery.VAL\_NM\_RQ=R01235&UniDbQuery.From=01.03.2010&UniDbQuery.To=22.03.2025 (дата обращения: 21.03.2025).
5. Курс евро к рублю [Электронный ресурс] // Центральный банк Российской Федерации. — URL: https://www.cbr.ru/currency\_base/dynamics/?UniDbQuery.Posted=True&UniDbQuery.so=1&UniDbQuery.mode=1&UniDbQuery.date\_req1=&UniDbQuery.date\_req2=&UniDbQuery.VAL\_NM\_RQ=R01239&UniDbQuery.From=01.03.2010&UniDbQuery.To=22.03.2025 (дата обращения: 22.03.2025).
6. Исторические данные по нефти Brent [Электронный ресурс] // Investing.com. — URL: https://www.investing.com/commodities/brent-oil-historical-data (дата обращения: 23.03.2025).
7. Цены на драгоценные металлы [Электронный ресурс] // Центральный банк Российской Федерации. — URL: https://cbr.ru/hd\_base/metall/metall\_base\_new/ (дата обращения: 24.03.2025).
8. Регрессионные модели [Электронный ресурс] // Habr. — URL: https://habr.com/ru/articles/514818/ (дата обращения: 25.03.2025).
9. Основные методы прогнозирования валютных курсов [Электронный ресурс] // Банк НН. — URL: https://www.banknn.ru/zhurnal/novosti/osnovnye-metody-prognozirovaniya-valyutnyh-kursov (дата обращения: 26.03.2025).
10. Факторы изменения курсов валют [Электронный ресурс] // Audit-it. — URL: https://www.audit-it.ru/news/finance/1117349.html?ysclid=m9mhoydb499870327 (дата обращения: 27.03.2025).
11. Оценка валютного риска и прогнозирование валютного курса [Электронный ресурс] // CyberLeninka. — URL: https://cyberleninka.ru/article/n/otsenka-valyutnogo-riska-i-prognozirovanie-valyutnogo-kursa/viewer (дата обращения: 28.03.2025).
12. Анализ временных рядов [Электронный ресурс] // Habr. — URL: https://habr.com/ru/companies/otus/articles/732080/ (дата обращения: 29.03.2025).
13. Машинное обучение (Machine Learning, ML) [Электронный ресурс] // Ultralytics. — URL: https://www.ultralytics.com/ru/glossary/machine-learning-ml (дата обращения: 30.03.2025).
14. Официальный FAQ [Электронный ресурс] // Telegram. — URL: https://telegram.org/faq?setln=ru (дата обращения: 31.03.2025).
15. Знакомство с aiogram [Электронный ресурс] // MasterGroosha. — URL: https://mastergroosha.github.io/aiogram-3-guide/ (дата обращения: 01.04.2025).
16. Документация Pandas [Электронный ресурс] // Pandas. — URL: https://pandas.pydata.org/docs/ (дата обращения: 02.04.2025).
17. Документация NumPy [Электронный ресурс] // NumPy. — URL: https://numpy.org/doc/stable/ (дата обращения: 03.04.2025).
18. Документация Seaborn [Электронный ресурс] // Seaborn. — URL: https://seaborn.pydata.org/ (дата обращения: 04.04.2025).
19. Документация Matplotlib [Электронный ресурс] // Matplotlib. — URL: https://matplotlib.org/stable/index.html (дата обращения: 05.04.2025).
20. Документация Scikit-learn [Электронный ресурс] // Scikit-learn. — URL: https://scikit-learn.org/stable/index.html (дата обращения: 06.04.2025).
21. Документация TensorFlow [Электронный ресурс] // TensorFlow. — URL: https://tensorflow.org/api\_docs/python/tf (дата обращения: 07.04.2025).
22. Документация Joblib [Электронный ресурс] // Joblib. — URL: https://joblib.readthedocs.io/en/stable/ (дата обращения: 08.04.2025).
23. Документация Python-dotenv [Электронный ресурс] // PyPI. — URL: https://pypi.org/project/python-dotenv/ (дата обращения: 09.04.2025)

# ПРИЛОЖЕНИЯ

Листинг программы для анализа данных и построения моделей прогнозирования

# Импорт библиотек

from math import \* # вкл. в себя ключает математические функции и константы

import itertools # используется для работы с итерациями

import pandas as pd # для анализа данных

import seaborn as sns # для визуализации данных на основе Matplotlib

import numpy as np # для работы с многомерными массивами

import matplotlib.pyplot as plt # для создания визуализаций

import warnings # модуль для управления предупреждениями (сделан вывод более чистым)

warnings.filterwarnings("ignore")

# Чтение данных таблицы

df\_d=pd.read\_excel('C:/Users/юзер/Desktop/Курсовая/Таблицы датасеты/Доллар табл.xlsx')

# Создаем индекс из даты

df\_d['data'] = pd.to\_datetime(df\_d['data'])

# Устанавливаем data как индекс

df\_d.set\_index('data', inplace=True)

# Сортировка датафреймов по индексу (по датам) в порядке возрастания

df\_d.sort\_index(ascending=True, inplace=True)

df\_d.info() # Выводит общую информацию о датафрейме

df\_d.describe() # Выводит статистические характеристики числовых столбцов

'''

В процессе анализа столбцов таблицы дадасета выявлено, что есть пропущенные значения в столбцах ключевой ставки (key interest rate) и инфляции (inflation). Поэтому, пропущенные значения будут заполнены методом ffill + интерполяцией для ключ. ставки и минтерполяцией для инфляции

'''

df\_d['key interest rate'].fillna(method='bfill', inplace=True) # заполнение первым известным значением

df\_d['key interest rate'] = df\_d['key interest rate'].interpolate(method='linear') # интерполяция для ключевой ставки

df\_d['inflation'].fillna(method='bfill', inplace=True)

df\_d['inflation'] = df\_d['inflation'].interpolate(method='linear') # интерполяция для инфляции

df\_d.head(20) # вывод 1-ых 20 значений датафрейма

# смотрим информацию о датафрейме после замены пропущ. значений

df\_d.info()

df\_d.describe()

df\_d = df\_d.drop(columns=['nominal']) # удаляем столбец с номиналом (там везде 1)

df\_d = df\_d.drop(columns=['name of cur']) # удаляем столбец с наименованием валюты (везде 'доллар')

'''

Проверим значение столбцов на выбросы

В дальнейшем у нас будет 2 датафрейма:

df\_d1 - для работы с значениями без выбросов

df\_d2 - выбросы остаются

'''

df\_d1 = df\_d.copy()

df\_d2 = df\_d.copy()

# Готовим "обрезанные" датафреймы для проверки

df\_d\_check1 = df\_d1[['curs usd', 'gold', 'silver', 'palladium', 'brent (in usd)', 'curs eur']] # проверяем цены

df\_d\_check2 = df\_d1[['key interest rate', 'inflation']] # проверяем ключ. ставку и инфляцию

# Графики 6 в 1. Для начала изобразим динамику изменения цены (для курса доллара, цены на драг. металлы, нефть, курс евро)

plt.figure(figsize=(15, 10))

for i, col in enumerate(df\_d\_check1.columns, 1):

plt.subplot(2, 3, i)

sns.lineplot(x=df\_d\_check1.index, y=df\_d\_check1[col])

plt.title(f'Динамика {col}')

plt.xlabel('Дата')

plt.ylabel('Цена')

plt.xticks(rotation=45)

plt.tight\_layout()

plt.show()

# Динамика изменения ключевой ставки и инфляции

plt.figure(figsize=(15, 10))

for i, col in enumerate(df\_d\_check2.columns, 1):

plt.subplot(2, 1, i)

sns.lineplot(x=df\_d\_check2.index, y=df\_d\_check2[col])

plt.title(f'Динамика {col}')

plt.xlabel('Дата')

plt.ylabel('Размер')

plt.xticks(rotation=45)

plt.tight\_layout()

plt.show()

'''

Мы получили динамику изменения цен, инфляции и т.п. Теперь строим графики ящики с усами, чтоб визуально ознакомится с выбросами в нашем наборе данных

'''

# Создаем общую фигуру с размером 15x10

plt.figure(figsize=(15, 10))

# Первые 2 графика (для ключевой ставки и инфляции)

for i, col in enumerate(df\_d\_check2.columns, 1):

plt.subplot(2, 4, i)

sns.boxplot(x=df\_d\_check2[col])

plt.title(f'Ящик с усами для {col}')

# Остальные 6 графиков (для цен)

for i, col in enumerate(df\_d\_check1.columns, 1):

plt.subplot(2, 4, i + 2)

sns.boxplot(x=df\_d\_check1[col])

plt.title(f'Ящик с усами для {col}')

plt.tight\_layout()

plt.show()

# Проверка выбросов для курса доллара

Mcd = df\_d1['curs usd'].mean() # ср. знач

Scd = df\_d1['curs usd'].std() # cр. кв. отклонение

print(f"Значение курса доллара не считается выбросом на интервале ({Mcd-3\*Scd}, {Mcd+3\*Scd})")

# Проверка выбросов для цены на золото

Mg = df\_d1['gold'].mean()

Sg = df\_d1['gold'].std()

print(f"Значение цены золота не считается выбросом на интервале ({Mg-3\*Sg}, {Mg+3\*Sg})")

# Проверка выбросов для цены на серебро

Mss = df\_d1['silver'].mean()

Sss = df\_d1['silver'].std()

print(f"Значение цены серебра не считается выбросом на интервале ({Mss-3\*Sss}, {Mss+3\*Sss})")

# Проверка выбросов для цены на палладий

Mpp = df\_d1['palladium'].mean()

Spp = df\_d1['palladium'].std()

print(f"Значение цены палладия не считается выбросом на интервале ({Mpp-3\*Spp}, {Mpp+3\*Spp})")

# Проверка выбросов для цены на нефть Brent

Mb = df\_d1['brent (in usd)'].mean()

Sb = df\_d1['brent (in usd)'].std()

print(f"Значение цены на нефть Brent не считается выбросом на интервале ({Mb-3\*Sb}, {Mb+3\*Sb})")

# Проверка выбросов для курса евро

Mce = df\_d1['curs eur'].mean()

Sce = df\_d1['curs eur'].std()

print(f"Значение курса доллара не считается выбросом на интервале ({Mce-3\*Sce}, {Mce+3\*Sce})")

# Проверка выбросов для кл. ставки

Mkir = df\_d1['key interest rate'].mean()

Skir = df\_d1['key interest rate'].std()

print(f"Значение ключевой ставки не считается выбросом на интервале ({Mkir-3\*Skir}, {Mkir+3\*Skir})")

# Проверка выбросов для инфляции

Minf = df\_d1['inflation'].mean()

Sinf = df\_d1['inflation'].std()

print(f"Значение инфляции не считается выбросом на интервале ({Minf-3\*Sinf}, {Minf+3\*Sinf})")

# Проверка выбросов методом IQR

# Проверка выбросов для курса доллара

Q1\_usd = df\_d1['curs usd'].quantile(0.25)

Q3\_usd = df\_d1['curs usd'].quantile(0.75)

IQR\_usd = Q3\_usd - Q1\_usd

print(f"Значение курса доллара не считается выбросом на интервале ({Q1\_usd-1.5\*IQR\_usd}, {Q3\_usd+1.5\*IQR\_usd})")

# Проверка выбросов для цены на золото

Q1\_gold = df\_d1['gold'].quantile(0.25)

Q3\_gold = df\_d1['gold'].quantile(0.75)

IQR\_gold = Q3\_gold - Q1\_gold

print(f"Значение цены золота не считается выбросом на интервале ({Q1\_gold-1.5\*IQR\_gold}, {Q3\_gold+1.5\*IQR\_gold})")

# Проверка выбросов для цены на серебро

Q1\_silver = df\_d1['silver'].quantile(0.25)

Q3\_silver = df\_d1['silver'].quantile(0.75)

IQR\_silver = Q3\_silver - Q1\_silver

print(f"Значение цены серебра не считается выбросом на интервале ({Q1\_silver-1.5\*IQR\_silver}, {Q3\_silver+1.5\*IQR\_silver})")

# Проверка выбросов для цены на палладий

Q1\_palladium = df\_d1['palladium'].quantile(0.25)

Q3\_palladium = df\_d1['palladium'].quantile(0.75)

IQR\_palladium = Q3\_palladium - Q1\_palladium

print(f"Значение цены палладия не считается выбросом на интервале ({Q1\_palladium-1.5\*IQR\_palladium}, {Q3\_palladium+1.5\*IQR\_palladium})")

# Проверка выбросов для цены на нефть Brent

Q1\_brent = df\_d1['brent (in usd)'].quantile(0.25)

Q3\_brent = df\_d1['brent (in usd)'].quantile(0.75)

IQR\_brent = Q3\_brent - Q1\_brent

print(f"Значение цены на нефть Brent не считается выбросом на интервале ({Q1\_brent-1.5\*IQR\_brent}, {Q3\_brent+1.5\*IQR\_brent})")

# Проверка выбросов для курса евро

Q1\_eur = df\_d1['curs eur'].quantile(0.25)

Q3\_eur = df\_d1['curs eur'].quantile(0.75)

IQR\_eur = Q3\_eur - Q1\_eur

print(f"Значение курса евро не считается выбросом на интервале ({Q1\_eur-1.5\*IQR\_eur}, {Q3\_eur+1.5\*IQR\_eur})")

# Проверка выбросов для кл. ставки

Q1\_keyrate = df\_d1['key interest rate'].quantile(0.25)

Q3\_keyrate = df\_d1['key interest rate'].quantile(0.75)

IQR\_keyrate = Q3\_keyrate - Q1\_keyrate

print(f"Значение ключевой ставки не считается выбросом на интервале ({Q1\_keyrate-1.5\*IQR\_keyrate}, {Q3\_keyrate+1.5\*IQR\_keyrate})")

# Проверка выбросов для инфляции

Q1\_inf = df\_d1['inflation'].quantile(0.25)

Q3\_inf = df\_d1['inflation'].quantile(0.75)

IQR\_inf = Q3\_inf - Q1\_inf

print(f"Значение инфляции не считается выбросом на интервале ({Q1\_inf-1.5\*IQR\_inf}, {Q3\_inf+1.5\*IQR\_inf})")

# Заменяем выбросы в цене золота на значение медианы

# Относительно остальных драг. металлов, золото обладает наибольшей стабильностью в плане изменения цены

# Поэтому все выбросы в цене золота заменены на медиану

lower\_gold = Q1\_gold - 1.5 \* IQR\_gold

upper\_gold = Q3\_gold + 1.5 \* IQR\_gold

gold\_median = df\_d1['gold'].median() # медиана

df\_d1['gold'] = np.where( # Заменяем выбросы в цене золота на значение медианы

(df\_d1['gold'] < lower\_gold) | (df\_d1['gold'] > upper\_gold), # если значение меньше нижней или больше верхней границы

gold\_median, # то заменяем его на медиану

df\_d1['gold'] # иначе оставляем как есть

)

# Заменем выбросы в цене серебра

# Серебро более волатильно (нестабильно), чем золото. Его цена чаще резко меняется из-за промышленного спроса

df\_d1['silver'] = df\_d1['silver'].clip(Q1\_silver - 1.5\*IQR\_silver, Q3\_silver + 1.5\*IQR\_silver) # значения за границами как бы "подрезаются"

# все значения меньше нижней границы (Q1 - 1.5\*IQR) заменяются на эту границу

# все значения больше верхней границы (Q3 + 1.5\*IQR) заменяются на эту границу

# значения внутри границ остаются без изменений

# Аналогичная (почти), что и у серебра, замена выбросов в цене палладия

# Палладий очень нестабилен нестабилен по сравнению с серебром и золотом

# Резкие взлеты и падения палладия – его естественное поведение из-за дефицита на рынке

df\_d1['palladium'] = df\_d1['palladium'].clip(Q1\_palladium - 1\*IQR\_palladium, Q3\_palladium + 1\*IQR\_palladium)

# Такие жесткие границы (1\*IQR вместо стандартных 1.5\*IQR) защищают от перекосов, но не "убивают" волатильность

# Усредняем выбросы в значениях ключ. ставки и инфляции

for col in ['key interest rate', 'inflation']:

Q1\_n = df\_d1[col].quantile(0.25) # 1 квартиль

Q3\_n = df\_d1[col].quantile(0.75) # 3 квартиль

IQR\_n = Q3\_n - Q1\_n # межквартильный размах

low\_bound\_n = Q1\_n - 1.5 \* IQR\_n # нижняя граница

up\_bound\_n = Q3\_n + 1.5 \* IQR\_n # верхняя граница

df\_d1[col] = df\_d1[col].clip(low\_bound\_n, up\_bound\_n)

# Рисуем графики, чтоб проверить устранение выбросов

plt.figure(figsize=(15, 10))

for i, col in enumerate(df\_d1.columns, 1):

plt.subplot(2, 4, i)

sns.boxplot(x=df\_d1[col])

plt.title(f'Ящик с усами для {col}')

plt.tight\_layout()

plt.show()

# Строим тепловую карту корреляций (для 1-ой модели)

plt.figure(figsize=(12, 8))

corr\_matrix1 = df\_d1.corr(numeric\_only=True) # Только числовые колонки

sns.heatmap(corr\_matrix1, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)

plt.title("Матрица корреляции (без выбросов)")

plt.show()

# для модели с выбросами (df\_d2)

plt.figure(figsize=(12, 8))

corr\_matrix2 = df\_d2.corr(numeric\_only=True)

sns.heatmap(corr\_matrix2, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)

plt.title("Матрица корреляции (с выбросами)")

plt.show()

# Функция для добавления лаговых признаков

def add\_lags(df, target\_column, n\_lags):

for i in range(1, n\_lags + 1):

df[f'{target\_column}\_lag{i}'] = df[target\_column].shift(i)

return df

# Добавляем лаговые признаки (за 7 дней)

n\_lags = 7

df\_d1 = add\_lags(df\_d1, 'curs usd', n\_lags)

df\_d2 = add\_lags(df\_d2, 'curs usd', n\_lags)

# Удаляем строки с NaN (на всякий случай, если они на этапе обработки данных вдруг не удалились)

df\_d1 = df\_d1.dropna()

df\_d2 = df\_d2.dropna()

# Разделение на обучающую и тестовую выборки (80% - обучение, 20% - тест)

def split\_data(df):

train\_size = int(len(df) \* 0.8)

train\_data = df.iloc[:train\_size]

test\_data = df.iloc[train\_size:]

X\_train = train\_data.drop('curs usd', axis=1)

y\_train = train\_data['curs usd']

X\_test = test\_data.drop('curs usd', axis=1)

y\_test = test\_data['curs usd']

return X\_train, X\_test, y\_train, y\_test

# Разделяем данные

X\_train\_d1, X\_test\_d1, y\_train\_d1, y\_test\_d1 = split\_data(df\_d1)

X\_train\_d2, X\_test\_d2, y\_train\_d2, y\_test\_d2 = split\_data(df\_d2)

from sklearn.linear\_model import LinearRegression # Импорт модели линейной регрессии для построения линейной зависимости между признаками и целевой переменной

from sklearn.ensemble import RandomForestRegressor # Импорт ансамблевой модели случайного леса для прогнозирования на основе множества деревьев решений

from sklearn.svm import SVR # Импорт метода опорных векторов для регрессии, эффективного при работе с нелинейными зависимостями

from tensorflow.keras.models import Sequential # Импорт последовательной модели нейронной сети from tensorflow.keras.layers import Dense # Импорт полносвязного слоя нейронной сети для построения многослойного перцептронаиз библиотеки Keras (входит в TensorFlow)

# Модели для df\_d1

# Линейная регрессия

lr\_model\_d1 = LinearRegression() # Создаём объект модели линейной регрессии

lr\_model\_d1.fit(X\_train\_d1, y\_train\_d1) # обучаем модель линейной регрессии на обучающих данных (y\_train\_d1 — целевая переменная (курс доллара curs usd))

y\_pred\_lr\_d1 = lr\_model\_d1.predict(X\_test\_d1) # Делаем предсказания на тестовой выборке X\_test\_d1 с помощью обученной модели

# Случайный лес

rf\_model\_d1 = RandomForestRegressor(n\_estimators=500, random\_state=42) # Создаём объект модели случайного леса rf\_model\_d1 с помощью класса RandomForestRegressor

# (n\_estimators=500 — число деревьев в "лесу" (чем больше, тем точнее, но дольше обучение), random\_state=42 — случайное начальное состояние)

rf\_model\_d1.fit(X\_train\_d1, y\_train\_d1) # обучаем модель

y\_pred\_rf\_d1 = rf\_model\_d1.predict(X\_test\_d1) # предсказания

# SVR (метод опорных векторов)

svr\_model\_d1 = SVR(kernel='rbf') # Создаем модель SVR (kernel='rbf' — используем радиально-базисное ядро (RBF), которое хорошо подходит для нелинейных зависимостей)

# SVR пытается найти гиперплоскость, которая наилучшим образом предсказывает значения, минимизируя ошибку

svr\_model\_d1.fit(X\_train\_d1, y\_train\_d1) # обучаем модель

y\_pred\_svr\_d1 = svr\_model\_d1.predict(X\_test\_d1) # предсказания

# Нейросеть

nn\_model\_d1 = Sequential([ # Создаём объект нейросети nn\_model\_d1 с помощью класса Sequential

# sequential означает что слои нейросети добавляются последовательно

Dense(64, activation='relu', input\_shape=(X\_train\_d1.shape[1],)), # первый слой

# activation='relu' — функция активации relu (Rectified Linear Unit), которая делает модель нелинейной

# То есть, если значение положительное — оно остаётся, если отрицательное — становится нулём

# Это помогает нейронной сети учиться нелинейным зависимостям в данных

# input\_shape=(X\_train\_d1.shape[1],) — размер входных данных (число признаков в X\_train\_d1)

# X\_train\_d1.shape[1] — это число признаков (колонок) в этих данных

# Например, если 10 признаков (столбцов), будет input\_shape=(10,)

# Круглая скобка с запятой (10,) — это кортеж с одним элементом, как требует Keras

Dense(32, activation='relu'), # второй слой

Dense(1) # слой с одним нейроном (без активации), который выдаёт предсказание (число — курс доллара)

])

'''

X\_train\_d1.shape[0] — это сколько строк (примеров/наблюдений)

X\_train\_d1.shape[1] — это сколько колонок (признаков)

'''

nn\_model\_d1.compile(optimizer='adam', loss='mse') # компиляция модели

# optimizer='adam' — используем оптимизатор Adam

# loss='mse' — функция потерь "среднеквадратичная ошибка"

nn\_model\_d1.fit(X\_train\_d1, y\_train\_d1, epochs=50, batch\_size=32, verbose=0) # обучение нейросетевой модели

'''

Обучаем нейронную сеть:

epochs=50 — сеть обучается в течение 50 проходов по всем данным

batch\_size=32 — данные разбиваются на батчи по 32 примера чтобы ускорить обучение

verbose=0 — отключаем вывод прогресса обучения (чтобы не загромождать вывод всяким хламом)

'''

y\_pred\_nn\_d1 = nn\_model\_d1.predict(X\_test\_d1, verbose=0).flatten() # предсказание

# .flatten() — преобразуем массив из формы (n\_samples, 1) в одномерный массив (n\_samples,) для удобства работы с другими моделями

# Аналогично для df\_d2

# Линейная регрессия

lr\_model\_d2 = LinearRegression()

lr\_model\_d2.fit(X\_train\_d2, y\_train\_d2)

y\_pred\_lr\_d2 = lr\_model\_d2.predict(X\_test\_d2)

# Случайный лес

rf\_model\_d2 = RandomForestRegressor(n\_estimators=500, random\_state=42)

rf\_model\_d2.fit(X\_train\_d2, y\_train\_d2)

y\_pred\_rf\_d2 = rf\_model\_d2.predict(X\_test\_d2)

# SVR

svr\_model\_d2 = SVR(kernel='rbf')

svr\_model\_d2.fit(X\_train\_d2, y\_train\_d2)

y\_pred\_svr\_d2 = svr\_model\_d2.predict(X\_test\_d2)

# Нейросеть

nn\_model\_d2 = Sequential([

Dense(64, activation='relu', input\_shape=(X\_train\_d2.shape[1],)),

Dense(32, activation='relu'),

Dense(1)

])

nn\_model\_d2.compile(optimizer='adam', loss='mse')

nn\_model\_d2.fit(X\_train\_d2, y\_train\_d2, epochs=50, batch\_size=32, verbose=0)

y\_pred\_nn\_d2 = nn\_model\_d2.predict(X\_test\_d2, verbose=0).flatten()

# Комбинированные предсказания для df\_d1

y\_pred\_combined\_d1 = (y\_pred\_lr\_d1 + y\_pred\_rf\_d1 + y\_pred\_svr\_d1 + y\_pred\_nn\_d1) / 4

print(y\_pred\_combined\_d1)

# Комбинированные предсказания для df\_d2

y\_pred\_combined\_d2 = (y\_pred\_lr\_d2 + y\_pred\_rf\_d2 + y\_pred\_svr\_d2 + y\_pred\_nn\_d2) / 4

print(y\_pred\_combined\_d2)

from sklearn.metrics import r2\_score, mean\_absolute\_error, mean\_absolute\_percentage\_error, mean\_squared\_error # импорт метрик

def calculate\_metrics(y\_true, predictions, model\_names): # определение функции

for y\_pred, name in zip(predictions, model\_names): # цикл по прогнозам и именам моделей

r2 = r2\_score(y\_true, y\_pred) # расчет R^2

mae = mean\_absolute\_error(y\_true, y\_pred) # расчет MAE

mape = mean\_absolute\_percentage\_error(y\_true, y\_pred) \* 100 # расчет MAPE в процентах

mse = mean\_squared\_error(y\_true, y\_pred) # расчет MSE

rmse = sqrt(mse) # расчет RMSE

print(f"Метрики для {name}:") # вывод заголовка

print(f"R²: {r2:.4f}") # вывод R^2

print(f"MAE: {mae:.4f}") # вывод MAE

print(f"MAPE: {mape:.2f}%") # вывод MAPE

print(f"MSE: {mse:.4f}") # вывод MSE

print(f"RMSE: {rmse:.4f}") # вывод RMSE

print() # добавление пустой строки

# Список прогнозов и имен моделей для df\_d1

predictions\_d1 = [y\_pred\_lr\_d1, y\_pred\_rf\_d1, y\_pred\_svr\_d1, y\_pred\_nn\_d1, y\_pred\_combined\_d1] # список прогнозов для df\_d1

model\_names\_d1 = ["линейной регрессии", "случайного леса", "SVR (метод опорных векторов)", "нейронной сети", "комбинированной модели"] # список имен моделей

# Вычисление метрик для df\_d1

print('Расчет метрик для df\_d1:\n')

calculate\_metrics(y\_test\_d1, predictions\_d1, model\_names\_d1) # вызов функции для df\_d1

print('Расчет метрик для df\_d2:\n')

predictions\_d2 = [y\_pred\_lr\_d2, y\_pred\_rf\_d2, y\_pred\_svr\_d2, y\_pred\_nn\_d2, y\_pred\_combined\_d2] # список прогнозов для df\_d2

model\_names\_d2 = ["линейной регрессии", "случайного леса", "SVR (метод опорных векторов)", "нейронной сети", "комбинированной модели"] # список имен моделей

# Вычисление метрик для df\_d2

calculate\_metrics(y\_test\_d2, predictions\_d2, model\_names\_d2) # вызов функции для df\_d2

Листинг программы Telegram-бота для прогнозирования курса доллара (по модулям)

Модуль main.py:

# Импортируем необходимые библиотеки

import asyncio # Для асинхронного запуска и работы с событиями

import logging # Для логирования действий бота

from aiogram import Bot, Dispatcher # Основные компоненты aiogram для создания бота

from aiogram.types import BotCommand # Для настройки списка команд в интерфейсе Telegram

from dotenv import load\_dotenv # Для загрузки переменных окружения из .env файла

import os # Для работы с операционной системой (например, получение переменных окружения)

from handlers import commands # Подключаем обработчики команд из отдельного модуля

# Настройка логирования: запись в файл и вывод в консоль

logging.basicConfig(

level=logging.INFO,

format='%(asctime)s - %(name)s - %(levelname)s - %(message)s',

handlers=[

logging.FileHandler('bot.log'), # Логи будут сохраняться в файл bot.log

logging.StreamHandler() # И одновременно выводиться в консоль

]

)

logger = logging.getLogger(\_\_name\_\_) # Создаем логгер для текущего модуля

# Загружаем переменные окружения из файла .env

load\_dotenv()

API\_TOKEN = os.getenv('BOT\_TOKEN') # Получаем токен бота

# Инициализируем бота и диспетчера

bot = Bot(token=API\_TOKEN) # Создаем объект бота с токеном

dp = Dispatcher() # Диспетчер будет обрабатывать входящие сообщения

# Регистрация обработчиков команд

dp.include\_router(commands.router) # Подключаем роутер с обработчиками команд

async def set\_commands(bot: Bot):

# Функция устанавливает список команд, отображаемых в интерфейсе Telegram

commands\_list = [

BotCommand(command="/start", description="Запустить/перезапустить бота"),

BotCommand(command="/predict", description="Прогноз курса доллара"),

BotCommand(command="/metrics", description="Метрики качества модели"),

BotCommand(command="/last", description="Последний известный курс"),

BotCommand(command="/help", description="Список команд"),

BotCommand(command="/correlation", description="Матрица корреляции признаков")

]

await bot.set\_my\_commands(commands\_list) # Устанавливаем команды

logger.info("Меню команд успешно настроено")

async def main():

# Главная функция для запуска бота

logger.info("Бот запущен") # Пишем в лог о начале работы

await set\_commands(bot) # Настраиваем меню команд

await dp.start\_polling(bot) # Запускаем процесс опроса сообщений от Telegram

# Точка входа в программу

if \_\_name\_\_ == '\_\_main\_\_':

asyncio.run(main()) # Запускаем асинхронную главную функцию

Модуль handlers/commands.py:

# Импорт необходимых библиотек

import pandas as pd # Для работы с табличными данными

import numpy as np # Для математических операций

import logging # Для логирования событий

import matplotlib.pyplot as plt # Для визуализации данных

import seaborn as sns # Для создания статистических графиков

import os # Для работы с файловой системой

from aiogram import Router, types # Фреймворк для создания Telegram бота

from aiogram.filters import Command # Для обработки команд бота

from aiogram.types import Message, ReplyKeyboardRemove, FSInputFile # Типы сообщений и вложений

from keyboards.reply import get\_days\_keyboard # Кастомная клавиатура для выбора дней

from models.models import predict\_combined, calculate\_metrics # Функции для прогнозирования и оценки модели

import joblib # Для загрузки сохраненных моделей

from datetime import datetime, timedelta # Для работы с датами

# Настройка логирования для отслеживания событий в боте

logger = logging.getLogger(\_\_name\_\_)

# Создаем роутер для обработчиков сообщений

router = Router()

# Загрузка данных и scaler из файлов

data = pd.read\_csv('latest\_data.csv', index\_col='data', parse\_dates=True) # Загрузка исторических данных

scaler = joblib.load('scaler.joblib') # Загрузка scaler для нормализации данных

# Колонки для предсказания (признаки модели)

FEATURES = ['gold', 'silver', 'palladium', 'key interest rate', 'inflation',

'curs eur', 'brent (in usd)', 'curs usd\_lag1', 'curs usd\_lag2',

'curs usd\_lag3', 'curs usd\_lag4', 'curs usd\_lag5', 'curs usd\_lag6',

'curs usd\_lag7']

# Основные признаки для корреляции (без лаговых признаков)

CORRELATION\_FEATURES = ['curs usd', 'gold', 'silver', 'palladium',

'key interest rate', 'inflation', 'curs eur', 'brent (in usd)']

# Функция проверки торгового дня

def is\_trading\_day(date):

# Получаем день недели (0 = понедельник, 6 = воскресенье)

weekday = date.weekday()

# Считаем воскресенье (6) и понедельник (0) неторговыми днями

if weekday in [0, 6]:

logger.info(f"Дата {date} неторговая (день недели: {weekday})")

return False

# Если дата в прошлом и присутствует в данных, проверяем её

trading\_dates = pd.to\_datetime(data.index).date

if date in trading\_dates:

logger.info(f"Дата {date} найдена в данных, считается торговой")

return True

# Для будущих дат предполагаем, что день торговый, если не выходной

logger.info(f"Дата {date} будущая, считается торговой")

return True

# Обработчик команды /start - приветственное сообщение

@router.message(Command('start'))

async def send\_welcome(message: Message):

logger.info(f"Получена команда /start от пользователя {message.from\_user.id}")

await message.reply(

"Привет! Я бот для прогнозирования курса доллара США к рублю. "

"Мои команды:\n"

"/predict - прогноз курса доллара на выбранное количество дней (или просто введите число от 1 до 7)\n"

"/metrics - посмотреть метрики качества\n"

"/last - последний курс\n"

"/correlation - матрица корреляции признаков\n"

"/help - список команд и информация о боте",

reply\_markup=ReplyKeyboardRemove()

)

# Обработчик команды /help - информация о командах

@router.message(Command('help'))

async def send\_help(message: Message):

logger.info(f"Получена команда /help от пользователя {message.from\_user.id}")

await message.reply(

"Доступные команды:\n"

"/start - запустить/перезапустить бота\n"

"/predict - прогноз курса доллара на выбранное количество дней (или просто введите число от 1 до 7)\n"

"/metrics - метрики качества модели\n"

"/last - последний известный курс\n"

"/correlation - матрица корреляции признаков\n"

"/help - этот список\n\n"

"Я бот, который строит прогнозы курса USD к RUB на 1–7 дней вперед...",

reply\_markup=ReplyKeyboardRemove()

)

# Обработчик команды /correlation - отправка матрицы корреляции

@router.message(Command('correlation'))

async def send\_correlation\_matrix(message: Message):

logger.info(f"Получена команда /correlation от пользователя {message.from\_user.id}")

try:

# Загружаем полные данные из файла

df = pd.read\_csv('df\_d2.csv', index\_col='data', parse\_dates=True)

# Вычисляем матрицу корреляции для выбранных признаков

corr\_matrix = df[CORRELATION\_FEATURES].corr()

# Создаем тепловую карту корреляции

plt.figure(figsize=(10, 8))

sns.heatmap(corr\_matrix, annot=True, cmap='coolwarm', fmt='.2f',

square=True, cbar=True, annot\_kws={'size': 8})

plt.title('Correlation Matrix')

plt.tight\_layout()

# Сохраняем изображение во временный файл

image\_path = 'correlation\_matrix.png'

plt.savefig(image\_path)

plt.close()

# Проверяем существование файла перед отправкой

if not os.path.exists(image\_path):

logger.error(f"Файл {image\_path} не найден")

await message.reply("Ошибка: не удалось создать изображение...")

return

# Отправляем изображение пользователю

await message.reply\_photo(

photo=FSInputFile(image\_path),

caption="Матрица корреляции признаков",

reply\_markup=ReplyKeyboardRemove()

)

# Отправляем описание признаков

description = "Описание признаков:\n- curs usd: Курс доллара США к рублю\n..."

await message.answer(description, reply\_markup=ReplyKeyboardRemove())

# Удаляем временный файл после отправки

os.remove(image\_path)

logger.info(f"Матрица корреляции успешно отправлена")

except Exception as e:

logger.error(f"Ошибка при генерации матрицы корреляции: {str(e)}")

await message.reply("Произошла ошибка при генерации матрицы корреляции...")

# Обработчик команды /last - отправка последнего известного курса

@router.message(Command('last'))

async def send\_last(message: Message):

logger.info(f"Получена команда /last от пользователя {message.from\_user.id}")

last\_date = data.index[-1].strftime('%Y-%m-%d')

last\_curs = data['curs usd'].iloc[-1]

await message.reply(

f"Последний курс доллара на {last\_date}: {last\_curs:.2f} руб.",

reply\_markup=ReplyKeyboardRemove()

)

# Обработчик команды /metrics - отправка метрик модели

@router.message(Command('metrics'))

async def send\_metrics(message: Message):

logger.info(f"Получена команда /metrics от пользователя {message.from\_user.id}")

# Берем последние 100 записей для тестирования

X\_test = data.iloc[-100:][FEATURES]

y\_true = data.iloc[-100:]['curs usd']

y\_pred = predict\_combined(X\_test) # Получаем предсказания

metrics = calculate\_metrics(y\_true, y\_pred) # Вычисляем метрики

# Формируем ответ с метриками

response = (

"Метрики качества комбинированной модели:\n"

f"R²: {metrics['R²']:.4f}\n"

f"MAE: {metrics['MAE']:.4f}\n"

f"MAPE: {metrics['MAPE']:.2f}%\n"

f"MSE: {metrics['MSE']:.4f}\n"

f"RMSE: {metrics['RMSE']:.4f}"

)

await message.reply(response, reply\_markup=ReplyKeyboardRemove())

# Обработчик команды /predict - запуск процесса прогнозирования

@router.message(Command('predict'))

async def predict\_curs(message: Message):

logger.info(f"Получена команда /predict от пользователя {message.from\_user.id}")

await message.reply(

"Выберите день для прогноза (1-7 дней)...",

reply\_markup=get\_days\_keyboard() # Показываем клавиатуру с выбором дней

)

# Обработчик выбора количества дней для прогноза

@router.message(lambda message: message.text in [str(i) for i in range(1, 8)])

async def process\_days(message: Message):

logger.info(f"Пользователь {message.from\_user.id} выбрал прогноз на {message.text} дней")

days = int(message.text)

last\_data = data.iloc[-1][FEATURES].copy() # Берем последние доступные данные

predictions = [] # Список для хранения прогнозов

# Прогнозируем на каждый день отдельно

for day in range(1, days + 1):

pred\_date = data.index[-1] + timedelta(days=day)

pred\_date\_str = pred\_date.strftime('%Y-%m-%d')

# Пропускаем неторговые дни

if not is\_trading\_day(pred\_date.date()):

predictions.append(f"На {pred\_date\_str} не было торгов.")

continue

# Подготавливаем данные и делаем прогноз

data\_scaled = scaler.transform(last\_data.values.reshape(1, -1))

pred = predict\_combined(last\_data.values.reshape(1, -1))[0]

predictions.append(f"Прогноз на {pred\_date\_str}: {pred:.2f} руб.")

# Обновляем лаговые переменные для следующего дня

last\_data['curs usd\_lag7'] = last\_data['curs usd\_lag6']

last\_data['curs usd\_lag6'] = last\_data['curs usd\_lag5']

# ... (аналогично для других лагов)

last\_data['curs usd\_lag1'] = pred # Заменяем последний лаг на новый прогноз

# Отправляем все прогнозы пользователю

await message.reply("\n".join(predictions), reply\_markup=get\_days\_keyboard())

# Обработчик неизвестных команд/сообщений

@router.message()

async def handle\_unknown(message: Message):

logger.warning(f"Получено неизвестное сообщение '{message.text}' от пользователя {message.from\_user.id}")

await message.answer(

"Я знаю только команды и числа от 1 до 7 \nИспользуйте /help для открытия списка команд",

reply\_markup=ReplyKeyboardRemove()

)

Модуль keyboards/reply.py:

# Создает интерактивную клавиатуру для выбора количества дней прогноза (1-7 дней)

from aiogram.types import ReplyKeyboardMarkup, KeyboardButton

def get\_days\_keyboard():

# Создаем список кнопок - каждая кнопка в отдельном ряду

buttons = [[KeyboardButton(text=str(i))] for i in range(1, 8)]

# Создаем клавиатуру с настройками:

# - keyboard: массив кнопок

# - resize\_keyboard: True - подстраивать размер клавиатуры под экран

keyboard = ReplyKeyboardMarkup(

keyboard=buttons,

resize\_keyboard=True,

one\_time\_keyboard=False # Клавиатура останется после выбора

)

return keyboard

Модуль models/models.py:

# Импорт библиотек для работы с данными и математических операций

import pandas as pd # Для работы с табличными данными (DataFrame)

import numpy as np # Для математических операций и работы с массивами

# Импорт моделей машинного обучения из scikit-learn

from sklearn.linear\_model import LinearRegression # Линейная регрессия

from sklearn.ensemble import RandomForestRegressor # Случайный лес для регрессии

from sklearn.svm import SVR # Метод опорных векторов для регрессии

from sklearn.model\_selection import train\_test\_split # Разделение данных на train/test

from sklearn.preprocessing import StandardScaler # Стандартизация данных

# Импорт компонентов для нейронной сети из TensorFlow/Keras

from tensorflow.keras.models import Sequential, load\_model # Последовательная модель НС

from tensorflow.keras.layers import Dense # Полносвязные слои для НС

# Импорт утилит

import joblib # Для сохранения и загрузки моделей

from utils.data\_preparation import prepare\_data # Кастомная функция для подготовки данных

# Подготовка данных

df\_d2 = prepare\_data() # Загружаем и предварительно обрабатываем данные

# Определение признаков (features) и целевой переменной (target)

features = [

'gold', 'silver', 'palladium', # Цены на драгметаллы

'key interest rate', 'inflation', # Макроэкономические показатели

'curs eur', 'brent (in usd)', # Курс евро и цена на нефть

'curs usd\_lag1', 'curs usd\_lag2', 'curs usd\_lag3', # Лаговые значения курса

'curs usd\_lag4', 'curs usd\_lag5', 'curs usd\_lag6', 'curs usd\_lag7'

]

X = df\_d2[features] # Матрица признаков

y = df\_d2['curs usd'] # Вектор целевой переменной (курс доллара)

# Разделение данных на обучающую и тестовую выборки (80%/20%)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, y,

test\_size=0.2, # 20% данных в тест

random\_state=42 # Фиксируем random state для воспроизводимости

)

# Масштабирование данных (приведение к единому масштабу)

scaler = StandardScaler() # Инициализация стандартизатора

X\_train\_scaled = scaler.fit\_transform(X\_train) # Обучение scaler и трансформация train

X\_test\_scaled = scaler.transform(X\_test) # Трансформация test (без обучения!)

# Обучение модели линейной регрессии

lr\_model = LinearRegression() # Инициализация модели

lr\_model.fit(X\_train\_scaled, y\_train) # Обучение на масштабированных данных

# Обучение модели случайного леса

rf\_model = RandomForestRegressor(

n\_estimators=100, # Количество деревьев

random\_state=42 # Фиксируем random state

)

rf\_model.fit(X\_train\_scaled, y\_train)

# Обучение модели SVR (Support Vector Regression)

svr\_model = SVR(kernel='rbf') # Используем radial basis function kernel

svr\_model.fit(X\_train\_scaled, y\_train)

# Создание и обучение нейронной сети

nn\_model = Sequential([ # Последовательная модель

Dense(64, activation='relu', input\_shape=(X\_train\_scaled.shape[1],)), # Входной слой

Dense(32, activation='relu'), # Скрытый слой

Dense(1) # Выходной слой (без активации для регрессии)

])

nn\_model.compile(

optimizer='adam', # Оптимизатор Adam

loss='mse' # Функция потерь - MSE (Mean Squared Error)

)

nn\_model.fit(

X\_train\_scaled, y\_train,

epochs=50, # Количество эпох обучения

batch\_size=32, # Размер батча

verbose=0 # Без вывода логов обучения

)

# Сохранение моделей и scaler'а

joblib.dump(lr\_model, 'lr\_model.joblib') # Линейная регрессия

joblib.dump(rf\_model, 'rf\_model.joblib') # Случайный лес

joblib.dump(svr\_model, 'svr\_model.joblib') # SVR

nn\_model.save('nn\_model.h5') # Нейронная сеть (сохраняется в HDF5)

joblib.dump(scaler, 'scaler.joblib') # Scaler для новых данных

# комбинированное предсказание на основе всех моделей (усреднение)

def predict\_combined(data):

data\_scaled = scaler.transform(data) # Масштабирование входных данных

# Получаем предсказания от всех моделей:

lr\_pred = lr\_model.predict(data\_scaled) # Линейная регрессия

rf\_pred = rf\_model.predict(data\_scaled) # Случайный лес

svr\_pred = svr\_model.predict(data\_scaled) # SVR

nn\_pred = nn\_model.predict(data\_scaled, verbose=0).flatten() # Нейросеть

# Усредняем предсказания:

combined\_pred = (lr\_pred + rf\_pred + svr\_pred + nn\_pred) / 4

return combined\_pred

# Расчет метрик качества модели

def calculate\_metrics(y\_true, y\_pred):

from sklearn.metrics import (

r2\_score, # Коэффициент детерминации

mean\_absolute\_error, # Средняя абсолютная ошибка

mean\_absolute\_percentage\_error, # Средняя абсолютная процентная ошибка

mean\_squared\_error # Среднеквадратичная ошибка

)

from math import sqrt # Для вычисления квадратного корня

# Вычисляем все метрики:

r2 = r2\_score(y\_true, y\_pred) # R² (чем ближе к 1, тем лучше)

mae = mean\_absolute\_error(y\_true, y\_pred) # MAE (в тех же единицах, что и target)

mape = mean\_absolute\_percentage\_error(y\_true, y\_pred) \* 100 # MAPE в процентах

mse = mean\_squared\_error(y\_true, y\_pred) # MSE (штрафует большие ошибки)

rmse = sqrt(mse) # RMSE (интерпретируемая MSE)

return {

'R²': r2,

'MAE': mae,

'MAPE': mape,

'MSE': mse,

'RMSE': rmse

}

# Пример использования: расчет метрик на тестовых данных

y\_pred = predict\_combined(X\_test) # Получаем предсказания

metrics = calculate\_metrics(y\_test, y\_pred) # Вычисляем метрики

Модуль utils/data\_preparation.py:

import pandas as pd # Основная библиотека для работы с табличными данными

def prepare\_data(input\_file='Доллар табл.xlsx', output\_file='df\_d2.csv', latest\_file='latest\_data.csv'):

# загрузка данных

# Чтение данных из Excel-файла

df = pd.read\_excel(input\_file)

# предбразование дат

# Проверяем, нужно ли преобразовывать столбец с датами

if not pd.api.types.is\_datetime64\_any\_dtype(df['data']):

# Преобразование дат из Excel-формата (число дней с 1899-12-30)

df['data'] = pd.to\_datetime(df['data'], origin='1899-12-30', unit='D')

# Устанавливаем дату в качестве индекса и сортируем по возрастанию

df.set\_index('data', inplace=True)

df.sort\_index(ascending=True, inplace=True)

# 3. обработка пропущенных значений

# Для ключевой ставки: сначала заполняем последующие значения, затем интерполируем

df['key interest rate'] = df['key interest rate'].bfill() # Заполнение последующими значениями

df['key interest rate'] = df['key interest rate'].interpolate(method='linear') # Линейная интерполяция

# Аналогично для инфляции

df['inflation'] = df['inflation'].bfill()

df['inflation'] = df['inflation'].interpolate(method='linear')

# удаление ненужных столбцов

# Удаляем столбцы, которые не используются в анализе

df.drop(columns=['nominal', 'name of cur'], inplace=True, errors='ignore')

# Добавление лаговых признаков

# Создаем 7 лаговых признаков для курса доллара

for i in range(1, 8):

df[f'curs usd\_lag{i}'] = df['curs usd'].shift(i) # Сдвиг на i дней назад

# Удаляем строки с пропусками (первые 7 дней, для которых нет всех лагов)

df.dropna(inplace=True)

# сохранение результатов

# Полные данные сохраняем в основной файл

df.to\_csv(output\_file)

# Последние 100 записей (актуальные данные) сохраняем в отдельный файл

df.tail(100).to\_csv(latest\_file)

return df

if \_\_name\_\_ == '\_\_main\_\_':

# Точка входа при запуске скрипта напрямую

prepare\_data()