Федеральное государственное автономное

образовательное учреждение высшего образования

«Пермский государственный национальный исследовательский университет» (ПГНИУ)

Региональный институт непрерывного образования (РИНО ПГНИУ)

Цифровая кафедра

Выпускная аттестационная (квалификационная) работа

по курсу профессиональной переподготовки «Анализ данных»

**ПОСТРОЕНИЕ МОДЕЛИ LSTM ДЛЯ ПРОГНОЗИРОВАНИЯ ЦЕН ЗАКРЫТИЯ ЧАСОВЫХ ИНТЕРВАЛОВ ETH/USDT**

Разработчики проекта:

Созинов Г.Н.,

Агапов М.Я.,

Косарский И.А.

Пермь, 2024

**Оглавление**

[ПАСПОРТ ПРОЕКТА 3](#_Toc185572298)

[СОДЕРЖАНИЕ ПРОЕКТА 4](#_Toc185572299)

[Анализ проблемы исследования 4](#_Toc185572300)

[Исходные данные 7](#_Toc185572301)

[Реализация проекта 11](#_Toc185572302)

[Этап 1. Сбор данных и их подготовка к анализу 11](#_Toc185572303)

[Этап 2. Предварительный анализ данных 15](#_Toc185572304)

[Этап 3. Корреляционный анализ данных 22](#_Toc185572305)

[Этап 4. Исследование данных как временных рядов 26](#_Toc185572306)

[Этап 5. Моделирование 33](#_Toc185572307)

[Заключение 44](#_Toc185572308)

[Список использованных источников и литературы 45](#_Toc185572309)

[Приложения 47](#_Toc185572310)

ПАСПОРТ ПРОЕКТА

**Название проекта:**Построение модели LSTM для прогнозирования цен закрытия часовых интервалов ETH/USDT

**Сведения об авторах:**Созинов Г.Н., Агапов М.Я., Косарский И.А.

**Цель:**собрать данные, создать и обучить предиктивную нейростевую модель на основе архитектуры LSTM, которая позволит прогнозировать стоимость закрытия пары ETH/USDT на часовых интервалах.

**Задачи:**

1. Выполнить анализ проблемы, обосновать ее актуальность.
2. Обосновать начальный выбор признаков, осуществить загрузку данных и подготовку их к анализу количественными методами, включая устранение пропущенных значений.
3. Выполнить предварительный анализ данных, включая анализ корреляции, автокорреляции и частичной автокорреляции, а также стационарности.
4. Из множества доступных факторов выбрать признаки, наиболее подходящие для моделирования.
5. Разработать и обучить модель LSTM, включая выбор оптимальных гиперпараметров и оценку ее качества.
6. Выполнить интерпретацию полученных результатов и сделать выводы о достижении цели.

**Краткое описание проекта:**

Требуется проанализировать исторические данные и определить возможность прогнозирования стоимости закрытия пары ETH/USDT с использованием LSTM-модели. Дать интерпретацию полученным результатам. Сделать выводы.

**Конкретные ожидаемые результаты:**

Обученная LSTM-модель и рекомендации по её применению.

СОДЕРЖАНИЕ ПРОЕКТА

Анализ проблемы исследования

Криптовалюта – это разновидность цифровой валюты, учёт внутренних расчётных единиц которой обеспечивает децентрализованная платёжная система, функционирующая в полностью автоматическом режиме. В отличие от традиционных фиатных денег, криптовалюты не выпускаются и не контролируются центральными банками или государственными органами. Их эмиссия и учёт транзакций осуществляются через распределённую сеть компьютеров, называемую блокчейном.

Криптовалюты представляют собой уникальный класс финансовых инструментов, отличающийся от традиционных активов, таких как акции или облигации. Они не имеют базовой стоимости, связанной с физическими активами или финансовой отчётностью. Их ценность определяется рыночным спросом и предложением, а также доверием участников рынка.

Использование криптовалюты актуально благодаря её уникальным свойствам, которые отвечают на вызовы современной экономики. Децентрализованный характер криптовалют обеспечивает независимость от банков и государственных органов, что даёт пользователям свободу от традиционных финансовых ограничений. Это особенно важно в регионах с низким уровнем доступа к банковским услугам.

Быстрые и экономичные транзакции делают криптовалюты удобным инструментом для международных переводов и глобальной торговли, снижая издержки на обслуживание операций. Многие криптовалюты имеют ограниченное предложение, что защищает их от инфляции и позволяет использовать их как средство сбережения. Кроме того, технологии блокчейн открывают новые возможности для цифровой экономики, включая децентрализованные приложения, смарт-контракты и децентрализованные финансы.

Рынок криптовалют продолжает активно развиваться. Согласно данным TradingView, на декабрь 2024 года совокупная рыночная капитализация криптовалют превысила 3,5 триллиона долларов США, а ежедневные объёмы торгов достигали 50–100 миллиардов долларов. Ethereum занимает второе место по рыночной капитализации (около 400 миллиардов долларов) после Bitcoin.

На стоимость криптовалют влияет множество факторов. Баланс спроса и предложения является ключевым фактором: ограниченное предложение криптовалют, таких как Bitcoin и Ethereum, способствует росту их стоимости при увеличении спроса. Регуляторные меры также оказывают значительное влияние: признание криптовалют официальным средством обмена или введение запретов меняют рыночную динамику. Внешние экономические и политические условия, технологические обновления и даже общественное мнение играют важную роль. Например, заявления известных предпринимателей или события, связанные с проектами на блокчейне, могут вызывать резкие изменения цен.

Цены криптовалют, включая Ethereum, характеризуются высокой волатильностью и сложными временными зависимостями, что делает их прогнозирование сложной, но крайне важной задачей. Это актуально для:

* инвесторов и трейдеров, которые используют прогнозы для определения оптимальных точек входа и выхода, минимизации убытков и увеличения прибыли;
* компаний-разработчиков, применяющих Ethereum как технологическую платформу, которым важно планировать затраты и доступность своих услуг;
* майнеров и валидаторов, анализирующих стоимость для оценки доходности своей деятельности;
* институциональных инвесторов и фондов, которым важно прогнозировать ценовые тренды для оценки рисков и создания стратегий управления капиталом;
* обычных пользователей, владеющих Ethereum как средством сбережения или расчётов.

Динамика цен на Ethereum, обусловленная технологическими изменениями, экономическими факторами и глобальными событиями, делает традиционные подходы к анализу данных малоэффективными. Современные методы, такие как нейронные сети, позволяют анализировать сложные временные зависимости и выявлять скрытые закономерности.

Модели на основе архитектуры LSTM доказали свою эффективность в задачах прогнозирования временных рядов благодаря способности учитывать долгосрочные и нелинейные зависимости, а также устойчивости к шумам. Применение LSTM для прогнозирования цен закрытия ETH/USDT предоставляет мощный инструмент для аналитиков и трейдеров, помогая повысить точность прогнозов и минимизировать финансовые риски.

*Цель:* выполнить анализ данных и разработать модель на основе LSTM для прогнозирования цены закрытия криптовалюты ETH/USDT на дневных интервалах.

*Задачи:*

1. Выполнить анализ проблемы, обосновать её актуальность.
2. Обосновать начальный выбор признаков, загрузить и подготовить данные к анализу, включая устранение пропущенных значений.
3. Провести анализ временных зависимостей данных, используя методы автокорреляции и частичной автокорреляции, для выбора оптимальных лагов.
4. Из множества доступных факторов выбрать признаки, наиболее подходящие для моделирования, чтобы снизить переобучение.
5. Разработать и обучить модель LSTM, включая подбор оптимальных гиперпараметров и оценку её качества.
6. Провести интерпретацию результатов и сделать выводы о достижении цели.

Исходные данные

В ходе анализа проблемы было установлено, что стоимость Ethereum зависит от множества факторов. Для работы выбраны показатели из различных категорий, которые находятся в открытом доступе и обладают длительной историей. Эти данные включают:

1. Биржевые данные OHLCV (цены открытия, максимума, минимума, закрытия и объём торгов) по криптовалютам ETH, BTC, BNB и XRP – основным представителям криптовалютного рынка с высокой капитализацией и ликвидностью.
2. Фондовые индексы – мировые биржевые показатели, отражающие состояние глобальной экономики и рыночные настроения:
   * США: Dow Jones Industrial Average, S&P 500, Nasdaq Composite, Russell 2000;
   * Европа: FTSE 100, DAX, CAC 40, Euro Stoxx 50;
   * Азия: Nikkei 225, Hang Seng Index, ASX 200.
3. Товарные индексы, важные для анализа инфляции и макроэкономической стабильности:
   * Энергетика: нефть WTI, нефть Brent, природный газ;
   * Металлы: золото, серебро, медь;
   * Сельское хозяйство: кукуруза, пшеница, соя.
4. Валютные индексы и пары, отражающие движение капитала – Dollar Index, EUR/USD, GBP/USD, JPY/USD.
5. Доходности облигаций, представляющие состояние долгового рынка, конкурирующего с криптовалютами:
   * 10-летние, 30-летние, 5-летние казначейские облигации США;
   * 13-недельные казначейские векселя.
6. Индекс волатильности (VIX), известный как "индекс страха", который измеряет уровень неопределённости на рынках.

В процессе сбора данных была выявлена проблема, связанная с различием в частоте обновления данных по криптовалютам и остальным индикаторам. Биржевые данные о криптовалютах, такие как ETH, BTC, BNB и XRP, доступны круглосуточной, обновляясь каждый час без выходных. В то же время, данные по мировым фондовым индексам, товарным индексам, валютным парам и облигационным доходностям доступны лишь на дневной основе и ограничены рабочими днями бирж.

Эта разница в частоте обновления данных потребовала пересмотра дальнейшей тактики исследования. Первый подход – сосредоточится на краткосрочных прогнозах (1 час) и использовать биржевые данные ETH, BTC и рассчитанные на их основе индикаторы технического анализа – индекс относительной силы (RSI), индикатор схождения/расхождения скользящих средних (MACD), сигнальная линия MACD, гистограмма MACD, стохастический осциллятор slow\_K, осциллятор slow\_D, средний истинный диапазон (ATR), а также VIX и ставка ФРС США, публикации которых согласованы с частотой обновления биржевых данных. Второй подход – уменьшить частотность анализа с 1 часа до 1 дня. Однако, данный подход значительно ограничивает количество наблюдений, так как биржевые данные по ETH доступны лишь с 2017 года. Таким образом, в работе было принято решение остановиться на краткосрочном прогнозе.

Итоговый файл содержит данные с 2021-07-06 21:00:00 по 2024-10-28 10:00:00 (всего 29006 наблюдений) по 26 показателям:

1. **btc\_open** — цена открытия BTC за час.
2. **btc\_high** — максимальная цена BTC за час.
3. **btc\_low** — минимальная цена BTC за час.
4. **btc\_close** — цена закрытия BTC за час.
5. **btc\_volume** — общий объём торгов BTC за час.
6. **btc\_rsi** — индекс относительной силы (RSI) для BTC, рассчитанный на часовом интервале.
7. **btc\_macd** — значение индикатора MACD для BTC, рассчитанное на часовом интервале.
8. **btc\_macd\_signal** — сигнальная линия MACD для BTC, рассчитанная на часовом интервале.
9. **btc\_macd\_hist** — гистограмма MACD для BTC, показывающая расхождение между MACD и сигнальной линией на часовом интервале.
10. **btc\_slowK** — значение %K стохастического осциллятора для BTC на часовом интервале.
11. **btc\_slowD** — значение %D стохастического осциллятора для BTC на часовом интервале (сглаженное значение %K).
12. **btc\_atr** — средний истинный диапазон (ATR) для BTC, оценка волатильности на часовом интервале.
13. **eth\_open** — цена открытия ETH (Ethereum) за час.
14. **eth\_high** — максимальная цена ETH за час.
15. **eth\_low** — минимальная цена ETH за час.
16. **eth\_close** — цена закрытия ETH за час.
17. **eth\_volume** — общий объём торгов ETH за час.
18. **fear\_greed** — индекс страха и жадности, актуальный на данный час, отражающий рыночные настроения.
19. **eth\_rsi** — индекс относительной силы (RSI) для ETH, рассчитанный на часовом интервале.
20. **eth\_macd** — значение индикатора MACD для ETH, рассчитанное на часовом интервале.
21. **eth\_macd\_signal** — сигнальная линия MACD для ETH, рассчитанная на часовом интервале.
22. **eth\_macd\_hist** — гистограмма MACD для ETH, показывающая расхождение между MACD и сигнальной линией на часовом интервале.
23. **eth\_slowK** — значение %K стохастического осциллятора для ETH на часовом интервале.
24. **eth\_slowD** — значение %D стохастического осциллятора для ETH на часовом интервале (сглаженное значение %K).
25. **eth\_atr** — средний истинный диапазон (ATR) для ETH, оценка волатильности на часовом интервале.
26. **fed\_rate** — ставка ФРС США, актуальная на данный час.

Реализация проекта

Этап 1. Сбор данных и их подготовка к анализу

На данном этапе производится выгрузка данных о криптовалютах BTC/USDT и ETH/USDT с биржи Bybit с последующим расчётом технических индикаторов, а также выгрузка ставки ФРС США с сайта Federal Reserve Economic Data:

import time

import requests

import ccxt

import pandas as pd

# Функция для загрузки данных с Bybit

def fetch\_ohlcv(exchange, symbol, timeframe, start\_date, end\_date):

since = exchange.parse8601(start\_date)

now = exchange.parse8601(end\_date)

all\_ohlcv = []

while since < now:

ohlcv = exchange.fetch\_ohlcv(symbol, timeframe, since, limit=200)

if not ohlcv:

break

all\_ohlcv.extend(ohlcv)

since = ohlcv[-1][0] + 1

time.sleep(1) # Учитываем ограничение запросов

df = pd.DataFrame(all\_ohlcv, columns=['timestamp', 'open', 'high', 'low', 'close', 'volume'])

df['timestamp'] = pd.to\_datetime(df['timestamp'], unit='ms')

return df

# API-ключи для Bybit

api\_key = "api\_key" # Укажите ключ для Bybit

api\_secret = "api\_secret" # Укажите секрет для Bybit

# Инициализация соединения с Bybit

exchange = ccxt.bybit({

'apiKey': api\_key,

'secret': api\_secret,

'enableRateLimit': True

})

# Загрузка данных ETH и BTC

df\_ETH = fetch\_ohlcv(exchange, 'ETH/USDT', '1h', '2021-07-01T00:00:00Z', '2024-10-28T00:00:00Z')

df\_BTC = fetch\_ohlcv(exchange, 'BTC/USDT', '1h', '2021-07-01T00:00:00Z', '2024-10-28T00:00:00Z')

# Загрузка данных ставки ФРС

url\_frs = f"https://api.stlouisfed.org/fred/series/observations?series\_id=FEDFUNDS&api\_key={api\_key}&file\_type=json"

response\_frs = requests.get(url\_frs)

if response\_frs.status\_code == 200:

data\_frs = response\_frs.json()

df\_FRS = pd.DataFrame(data\_frs['observations'])

df\_FRS['date'] = pd.to\_datetime(df\_FRS['date'])

df\_FRS = df\_FRS[['date', 'value']]

df\_FRS = df\_FRS.rename(columns={"value": "fed\_rate"})

df\_FRS['year\_month'] = df\_FRS['date'].dt.to\_period('M').astype(str)

# Загрузка индекса страха и жадности (VIX)

url\_vix = "https://api.alternative.me/fng/?limit=0&format=json"

response\_vix = requests.get(url\_vix)

if response\_vix.status\_code == 200:

fear\_greed\_data = response\_vix.json()

df\_VIX = pd.DataFrame(fear\_greed\_data["data"])

df\_VIX['date'] = pd.to\_datetime(df\_VIX['timestamp'], unit='s')

df\_VIX = df\_VIX.drop(columns=['value\_classification', 'timestamp', 'time\_until\_update'])

df\_VIX = df\_VIX.rename(columns={"value": "fear\_greed\_index"})

Далее добавляем расчет основных индикаторов технического анализа и объединяем данные:

import talib

# Функция для добавления технических индикаторов в DataFrame

def add\_technical\_indicators(df, symbol\_prefix):

# RSI (Relative Strength Index)

df[f'{symbol\_prefix}\_rsi'] = talib.RSI(df[f'{symbol\_prefix}\_close'], timeperiod=14)

# MACD (Moving Average Convergence Divergence)

df[f'{symbol\_prefix}\_macd'], df[f'{symbol\_prefix}\_macd\_signal'], df[f'{symbol\_prefix}\_macd\_hist'] = talib.MACD(

df[f'{symbol\_prefix}\_close'], fastperiod=12, slowperiod=26, signalperiod=9

)

# Stochastic Oscillator (STOCH)

df[f'{symbol\_prefix}\_slowK'], df[f'{symbol\_prefix}\_slowD'] = talib.STOCH(

df[f'{symbol\_prefix}\_high'], df[f'{symbol\_prefix}\_low'], df[f'{symbol\_prefix}\_close'],

fastk\_period=14, slowk\_period=3, slowk\_matype=0, slowd\_period=3, slowd\_matype=0

)

# ATR (Average True Range)

df[f'{symbol\_prefix}\_atr'] = talib.ATR(

df[f'{symbol\_prefix}\_high'], df[f'{symbol\_prefix}\_low'], df[f'{symbol\_prefix}\_close'], timeperiod=14

)

return df

# Переименование столбцов и добавление технических индикаторов для BTC

df\_BTC = add\_technical\_indicators(df\_BTC.rename(columns={

'open': 'btc\_open', 'high': 'btc\_high', 'low': 'btc\_low', 'close': 'btc\_close', 'volume': 'btc\_volume'

}), 'btc')

# Переименование столбцов и добавление технических индикаторов для ETH

df\_ETH = add\_technical\_indicators(df\_ETH.rename(columns={

'open': 'eth\_open', 'high': 'eth\_high', 'low': 'eth\_low', 'close': 'eth\_close', 'volume': 'eth\_volume'

}), 'eth')

# Слияние данных ETH и BTC по дате

df\_market = pd.merge(df\_ETH, df\_BTC, on='timestamp', how='inner')

# Добавление данных ставки ФРС

df\_FRS = df\_FRS.rename(columns={'date': 'timestamp'}) # Переименование 'date' в 'timestamp'

df\_market = pd.merge\_asof( #данные ФРС имеют меньшую частоту обновления

df\_market.sort\_values('timestamp'),

df\_FRS.sort\_values('timestamp'),

on='timestamp',

direction='backward' # добавляет ближайшее предыдущее значение ставки

)

# Добавление данных индекса страха и жадности (VIX)

df\_VIX = df\_VIX.rename(columns={'date': 'timestamp'}) # Переименование 'date' в 'timestamp'

df\_market = pd.merge\_asof(

df\_market.sort\_values('timestamp'),

df\_VIX.sort\_values('timestamp'),

on='timestamp',

direction='backward'

)

# Сохранение итогового DataFrame в CSV

output\_file = "market\_project\_data\_usd.csv"

df\_market.to\_csv(output\_file, index=False)

Загрузим данные:

df = pd.read\_csv('market\_project\_data\_usd.csv')

df.head()

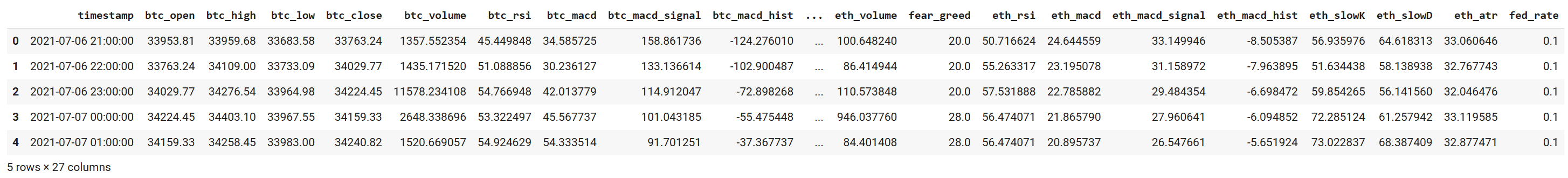


Рисунок 1. Исходный датафрейм

Убедимся, что все количественные столбцы имеют числовой тип.

df.info()

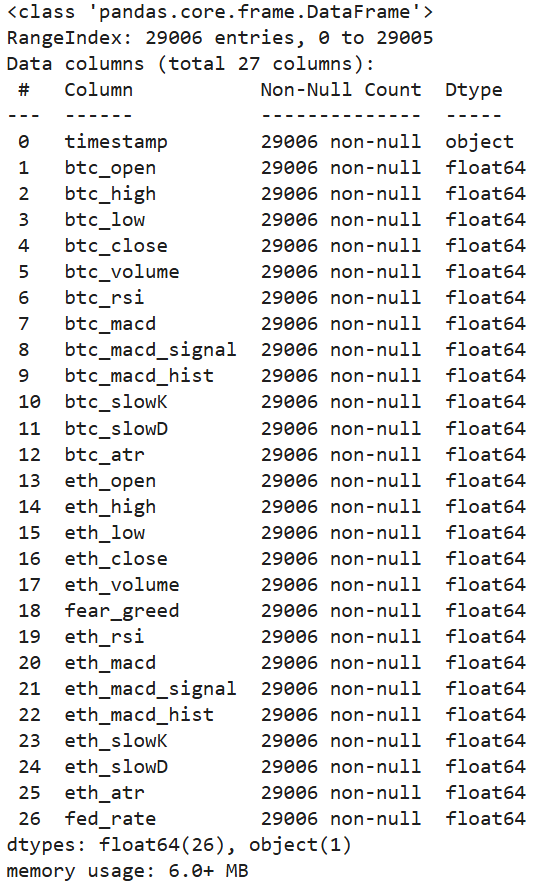


Рисунок 2. Типы данных колонок

Видно, что числовым типом обладают все столбцы кроме timestamp, который показывает дату и время наблюдения. Данные в силу загрузки идут по порядку и не имеют пропуском, поэтому этот столбец можно удалить:

df.drop(columns=['timestamp'], inplace=True)

Итак, результат первого этапа – это готовый к анализу набор данных в виде датафрейма.

Этап 2. Предварительный анализ данных

Вычислим описательные статистики по колонкам (среднее, моду, медиану, стандартное отклонение, квартили).

***Среднее арифметическое*** *равно сумме значений всех вариант выборки, деленной на объем выборки:*

.

Здесь *п* − объем выборки, а *xi* − варианты выборки.

***Модой*** называется значение признака, встречающееся в выборке наиболее часто. Условимся использовать для обозначения моды символы *Mo*.

В случае несгруппированных данных для нахождения медианы необходимо ранжировать выборку, т. е. расположить данные в порядке их возрастания или убывания. Медианой будет являться значение признака, находящееся в середине ранжированного ряда. Медиана находится по формуле

Выборочная дисперсия находится по формуле *.*

Используется также другая формула для вычисления дисперсии: , где *.*

Дисперсия имеет размерность квадрата размерности случайной величины, что затрудняет ее интерпретацию и делает не очень наглядной. Для более наглядного описания рассеяния удобнее пользоваться характеристикой, размерность которой совпадает с размерностью исследуемого признака. С этой целью вводится понятие ***стандартного отклонения*** (или ***среднего квадратичного отклонения***).

***Стандартным отклонением*** называется положительный квадратный корень из дисперсии:

.

Стандартное отклонение имеет те же единицы измерения, что и результаты измерения исследуемого признака, и, таким образом, оно характеризует степень отклонения признака от среднего арифметического. Иными словами, оно показывает, как расположена основная часть вариант относительно среднего арифметического.

df.describe()

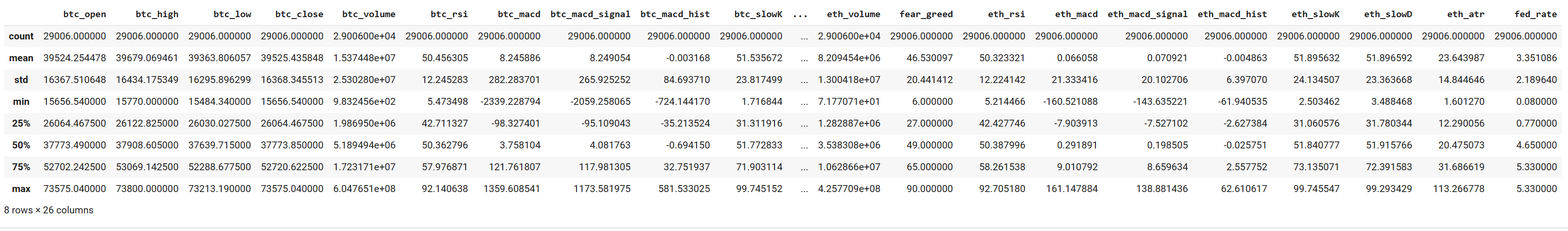


Рисунок 3. Описательные статистики по колонкам

***Аномальными наблюдениями*** (*выбросами*, англ. *Outliers, Extreme values*) называют такие значения уровня временного ряда, которые значительно отличаются от остальных. При выявлении подобных «выбросов» возникают серьезные вопросы: являются ли отклоняющиеся данные действительно ошибками (например, регистрации) или это реальные значения и как получить адекватные оценки для параметров изучаемой совокупности.

Проверим данные на наличие выбросов, для этого можно использовать диаграмму «ящик с усами» (boxplot). График ***«ящик с усами»,*** или ***«ящичковая диаграмма»***, или ***диаграмма размаха*** − график, используемый описательной статистике и компактно изображающий одномерное [распределение вероятностей](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D1%81%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D0%B2%D0%B5%D1%80%D0%BE%D1%8F%D1%82%D0%BD%D0%BE%D1%81%D1%82%D0%B5%D0%B9). Такой вид диаграммы в удобной форме показывает медиану, нижний и верхний квартили, минимальное и максимальное значения выборки и [выбросы](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D0%B1%D1%80%D0%BE%D1%81_(%D1%81%D1%82%D0%B0%D1%82%D0%B8%D1%81%D1%82%D0%B8%D0%BA%D0%B0)).

Несколько таких ящиков можно нарисовать рядом друг с другом, чтобы визуально сравнивать одно распределение с другим, их можно рисовать горизонтально либо вертикально. Расстояния между различными частями ящика позволяют определить степень распространения (дисперсии) и асимметрии в данных и выявить выбросы.

Границами ящика служат первый и третий [квартили](https://ru.wikipedia.org/wiki/%D0%9A%D0%B2%D0%B0%D1%80%D1%82%D0%B8%D0%BB%D1%8C) (25-й и 75-й [процентили](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D0%BD%D1%82%D0%B8%D0%BB%D1%8C) соответственно), линия в середине ящика — медиана (50-й процентиль). Концы усов — края статистически значимой выборки (без [выбросов](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D0%B1%D1%80%D0%BE%D1%81_(%D1%81%D1%82%D0%B0%D1%82%D0%B8%D1%81%D1%82%D0%B8%D0%BA%D0%B0))) могут определяться несколькими способами. В общем виде эта формула имеет вид

*.*

*X*н — нижняя граница уса, *X*в — верхняя граница уса, *Q*1 — первый квартиль ,*Q*3 — третий квартиль, *k* — коэффициент, наиболее часто употребляемое значение которого равно 1,5. Данные, выходящие за границы усов ([выбросы](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D0%B1%D1%80%D0%BE%D1%81_(%D1%81%D1%82%D0%B0%D1%82%D0%B8%D1%81%D1%82%D0%B8%D0%BA%D0%B0))), отображаются на графике в виде точек, маленьких кружков или звёздочек. Иногда на графике отмечают среднее арифметическое и его [доверительный интервал](https://ru.wikipedia.org/wiki/%D0%94%D0%BE%D0%B2%D0%B5%D1%80%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D0%B2%D0%B0%D0%BB_%D0%B4%D0%BB%D1%8F_%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B6%D0%B8%D0%B4%D0%B0%D0%BD%D0%B8%D1%8F_%D0%BD%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B9_%D0%B2%D1%8B%D0%B1%D0%BE%D1%80%D0%BA%D0%B8) («зарубка» на ящике). На рис. изображен график «ящик с усами».

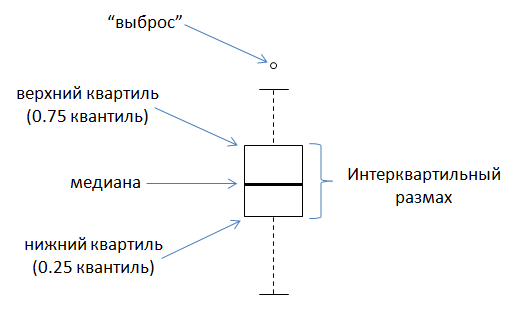


Рисунок 4. «Ящик с усами»

Посмотрим на диаграммы boxplot всех числовых колонок датафрейма:

df.boxplot();

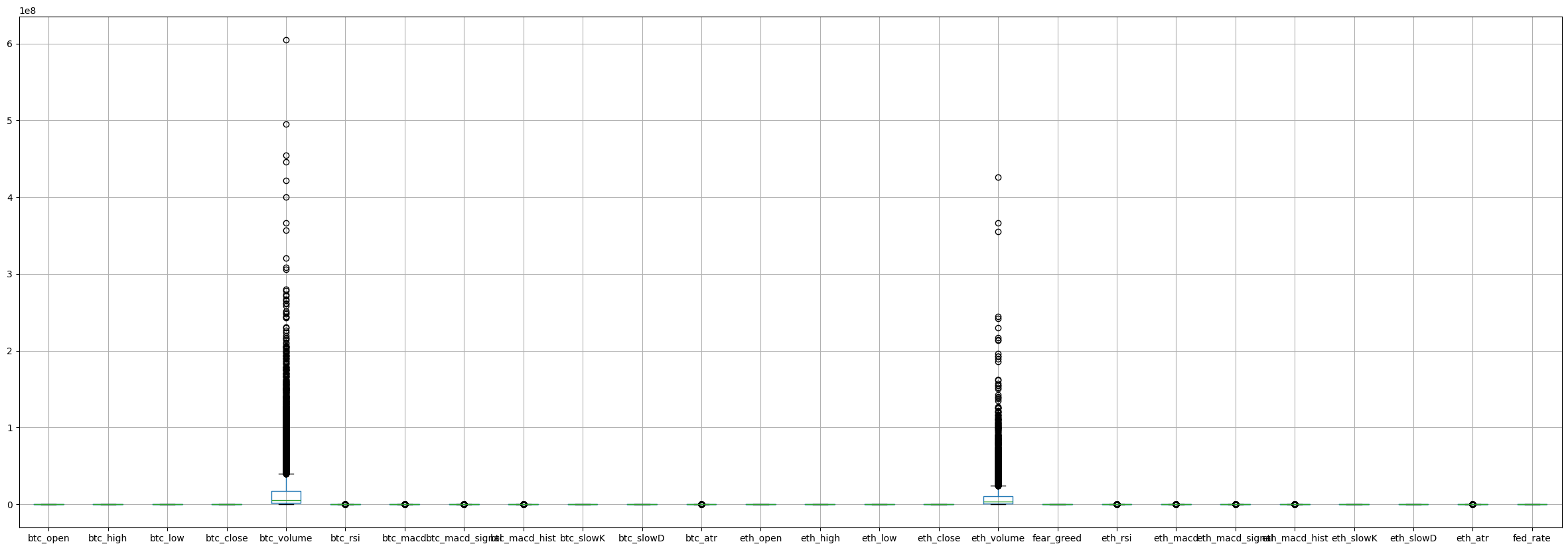


Рисунок 5. Диаграммы boxplot всех числовых колонок

Заметно, что есть явные выбросы в двух колонках – объемы торгов, однако, удалять эти данные нельзя, так как это разрушит структуру временного ряда.

***Гистограмма****,* представляющая собой совокупность примыкающих друг к другу прямоугольников, основание каждого из которых равно ширине интервала группировки, а площадь – частоты этого интервала.

Гистограмма строится в декартовой (прямоугольной) системе координат следующим образом. По оси абсцисс откладываются отрезки, отображающие интервалы группировки, а затем на каждом из них строится прямоугольник, площадь которого равна частости данного интервала. В случае если все интервалы группировки имеют одинаковую ширину, высоты прямоугольников пропорциональны соответствующим частотам.

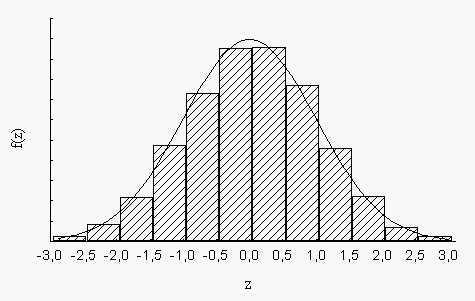


Рисунок 6. Гистограмма

Построим гистограммы для каждого показателя:

df.hist(figsize=(20,15), bins=30)

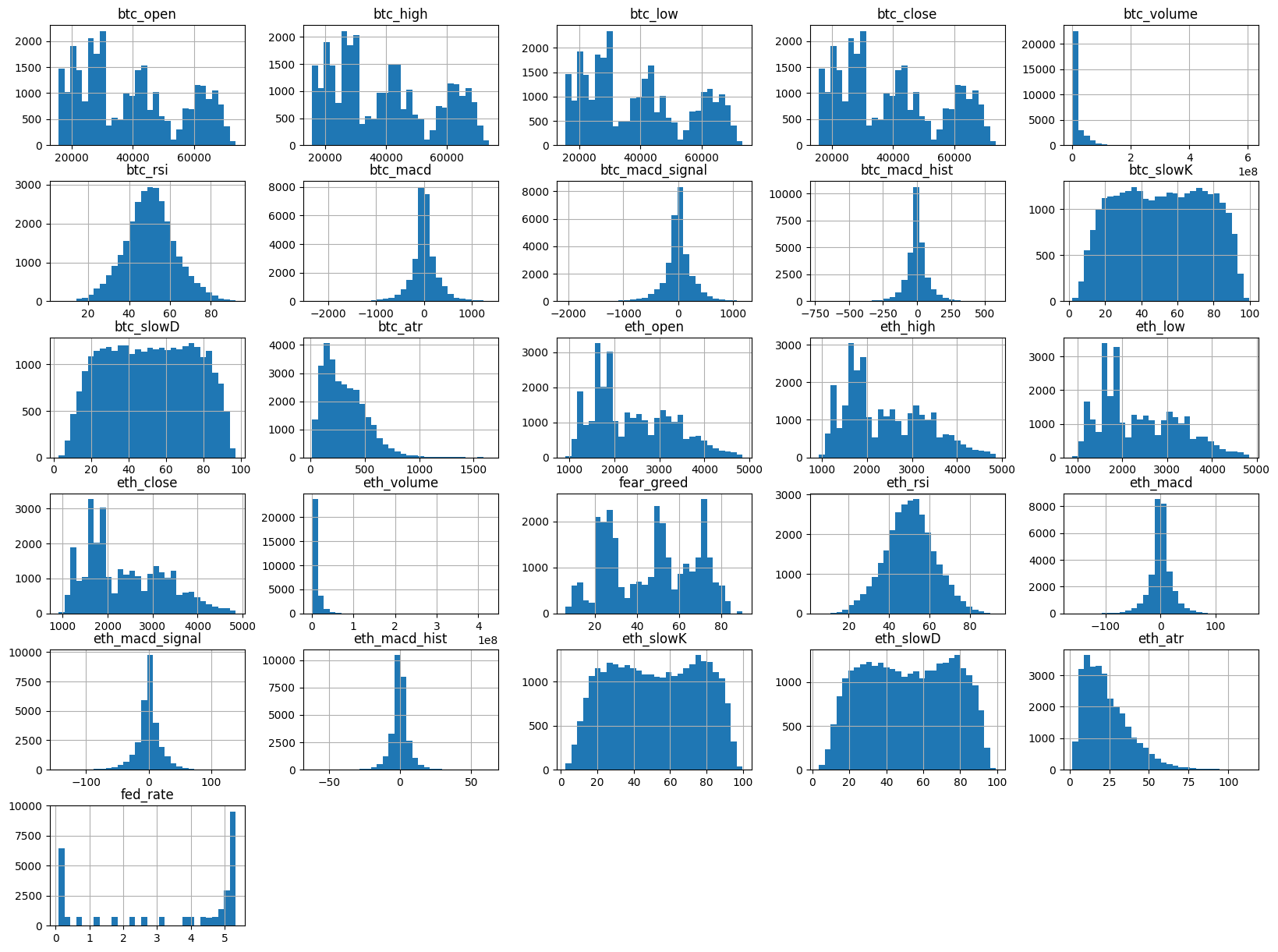


Рисунок 7. Boxplot и гистограммы для всех колонок

Визуально видно, что тут есть нормально распределенные колонки, поэтому применим статистические тесты (критерии согласия).

***Критерии согласия з***аключаются в проверке предположения о том, что результаты наблюдений могут быть описаны с помощью определенного закона распределения (в нашем случае нормального распределения).При малом числе измерений часто используется критерий Шапиро−Уилка.

На формальном языке проверяется гипотеза:

H0: наши данные согласуются с нормальным распределением.

Если p-value меньше заданного уровня значимости (обычно 0,05 или 0,01), то основная гипотезу отвергается.

Используем тест Шапиро-Уилка:

import scipy.stats as stats

for col in df.columns:

print(col, stats.shapiro(df[col]))

Результаты теста для колонок:

btc\_open ShapiroResult(statistic=0.9256832029091924, pvalue=4.248830614924243e-78)

btc\_high ShapiroResult(statistic=0.925553114786955, pvalue=3.8834765210616425e-78)

btc\_low ShapiroResult(statistic=0.9258694383081834, pvalue=4.833632852236131e-78)

btc\_close ShapiroResult(statistic=0.9256756295007288, pvalue=4.226632594211285e-78)

btc\_volume ShapiroResult(statistic=0.5918260282901862, pvalue=7.327997884489537e-121)

btc\_rsi ShapiroResult(statistic=0.9981004112328105, pvalue=3.815145042350994e-18)

btc\_macd ShapiroResult(statistic=0.9196536521918746, pvalue=7.530371215635807e-80)

btc\_macd\_signal ShapiroResult(statistic=0.9250303443930622, pvalue=2.7094748949884187e-78)

btc\_macd\_hist ShapiroResult(statistic=0.9185778311247071, pvalue=3.768814933753607e-80)

btc\_slowK ShapiroResult(statistic=0.9644187544455759, pvalue=1.5330738672591339e-62)

btc\_slowD ShapiroResult(statistic=0.9656701153929059, pvalue=7.883256699537093e-62)

btc\_atr ShapiroResult(statistic=0.9234299261281214, pvalue=9.120055528267194e-79)

eth\_open ShapiroResult(statistic=0.9452928957586978, pvalue=2.060996116373939e-71)

eth\_high ShapiroResult(statistic=0.9447266565470434, pvalue=1.2425671042767264e-71)

eth\_low ShapiroResult(statistic=0.9458935994072667, pvalue=3.5426336650883596e-71)

eth\_close ShapiroResult(statistic=0.9452947136017924, pvalue=2.0643616121040723e-71)

eth\_volume ShapiroResult(statistic=0.5630684108333351, pvalue=9.094353954909846e-123)

fear\_greed ShapiroResult(statistic=0.9471795150933786, pvalue=1.1493372753932333e-70)

eth\_rsi ShapiroResult(statistic=0.9993318608030375, pvalue=4.833218834174849e-09)

eth\_macd ShapiroResult(statistic=0.9152482987353354, pvalue=4.64423457367241e-81)

eth\_macd\_signal ShapiroResult(statistic=0.918503668324736, pvalue=3.594241999765873e-80)

eth\_macd\_hist ShapiroResult(statistic=0.9130446635144582, pvalue=1.2072271419161102e-81)

eth\_slowK ShapiroResult(statistic=0.9604432031228044, pvalue=1.1408298309579614e-64)

eth\_slowD ShapiroResult(statistic=0.9614509672908996, pvalue=3.794469998973637e-64)

eth\_atr ShapiroResult(statistic=0.9127753688743657, pvalue=1.0260246436566393e-81)

fed\_rate ShapiroResult(statistic=0.7602575834338676, pvalue=1.745701333716113e-106)

Как видим, p-value везде меньше 0,01, поэтому на уровне значимости 1% гипотеза о нормальности (H0) отвергается для всех колонок.

Этап 3. Корреляционный анализ данных

***Корреляционный анализ*** – это совокупность методов оценивания степени тесноты статистической связи между анализируемыми переменными.

Выполним корреляционный анализ данных с помощью матрицы корреляции. В случае нормальности всех данных следует использовать коэффициент корреляции Пирсона, в противном случае – ранговые коэффициенты корреляции Спирмена.

**Парный коэффициент** корреляции характеризует взаимосвязь двух переменных на фоне действия остальных показателей и является самым распространенным показателем тесноты связи при статистическом анализе данных.

Парный коэффициент корреляции между количественными случайными переменными и носит название *выборочного коэффициента корреляции* Пирсона (*sample correlation coefficient*) (или просто коэффициента корреляции) и находится по формуле

где и  — *выборочные дисперсии (sample variances*) переменных  и , а — *выборочная ковариация* или выборочный ковариационный момент, и соответствующие *средние (means)* определяются по формулам



Коэффициент корреляции обладает следующими свойствами:

1. Принимает значения от –1 до +1.
2. Если , то связь между переменными  и  считается сильной. Если , то связь слабая.
3. Если , то корреляционное поле наблюдений представляет собой совокупность точек, которые можно расположить на одной прямой. Знак «+» свидетельствует о прямой линейной зависимости между переменными и , а знак «—» − об обратной линейной зависимости.
4. При  линейная корреляционная связь отсутствует.

Метод ранговой корреляции Спирмена позволяет определить тесноту (силу) и направление корреляционной связи между двумя признаками (как количественными, так и качественными). Коэффициент ранговой корреляции имеет границы изменения от –1 до +1. Полное совпадение рангов означает максимально тесную прямую связь, полная противоположность рангов – максимально тесную обратную связь. Формула расчета ***коэффициента корреляции рангов Ч. Спирмена:***

где  – ранг  в выборке .

Матрицу корреляции отобразим с помощью диаграммы «тепловая карта» (heatmap).

import seaborn as sns

plt.rcParams["figure.figsize"] = 8, 6

Данные всех колонок не имеют нормального распределения, поэтому используем ранговый коэффициент Спирмена.

sns.heatmap(df.corr(method="spearman", numeric\_only=True), annot=True)

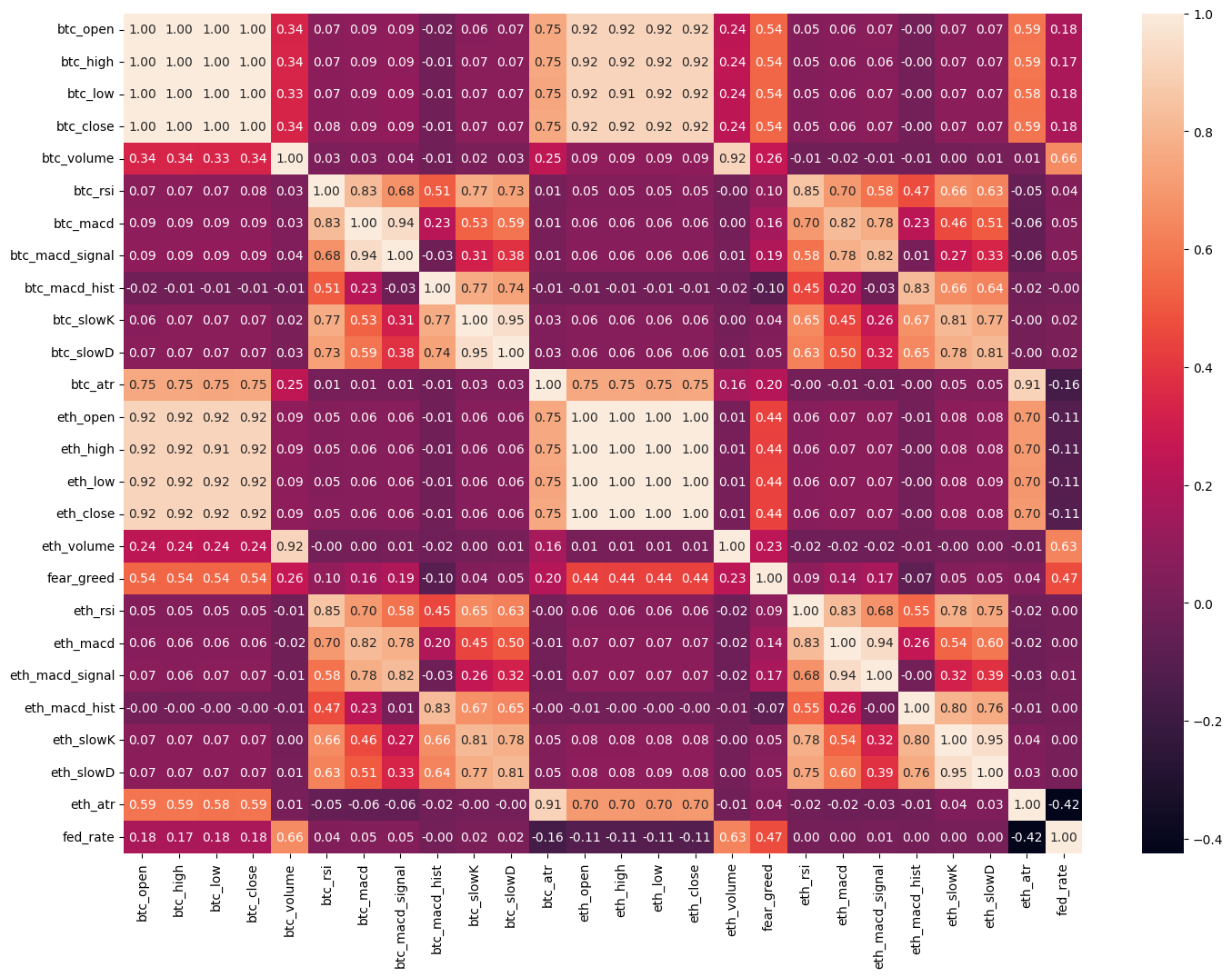


Рисунок 7. Тепловая карта матрицы корреляции

Целевой переменной является цена закрытия ETH (eth\_close).

y\_col = "eth\_close"

Выедем коэффициенты корреляции Спирмена целевой переменной с каждым фактором:

def print\_correlations(data, target):

    # Рассчитываем корреляции

    correlations = data.corr(method="spearman")[target].sort\_values(ascending=False)

    print(f"Коэффициенты корреляции с {target}:")

    print("-" \* 50)

    for feature, corr in correlations.items():

        print(f"{feature:20}: {corr:.4f}")

print\_correlations(df, y\_col)

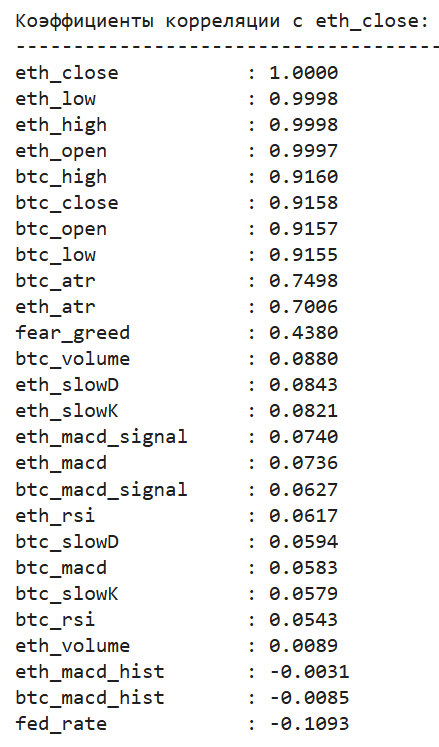


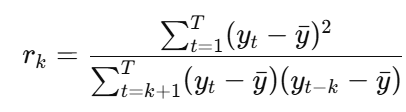
Рисунок 8. Корреляции целевой переменной с признаками

Этап 4. Исследование данных как временных рядов

Временной ряд – это последовательность значений, описывающих протекающий во времени процесс, измеренных в последовательные моменты времени, обычно через равные промежутки.

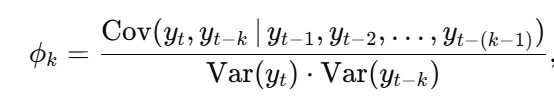
Для временных рядов полезно изучить автокорреляцию, так как она позволяет выявить зависимость текущих значений ряда от его предыдущих значений. Анализ автокорреляции помогает определить сезонность, тренды и циклические закономерности.

***Автокорреляционная функция (ACF)*** – статистическая взаимосвязь между последовательностями величин одного ряда, взятыми со сдвигом.



где  – лаг автокорреляции, а среднее вычисляется по всему ряду.

***Частичная автокорреляционная функция (PACF)*** измеряет степень линейной зависимости между значениями временного ряда на текущем шаге и значениями на k-ом лаге ​, исключая влияние промежуточных лагов (, *,* …, .

****

Построим графики ACF и PACF для целевой переменной:

import statsmodels.tsa.api as smt

fig = plt.figure(figsize=(20, 9))

layout = (2, 2)

ts\_ax = plt.subplot2grid(layout, (0, 0), colspan=2)

acf\_ax = plt.subplot2grid(layout, (1, 0))

pacf\_ax = plt.subplot2grid(layout, (1, 1))

ts = df[y\_col]

ts.plot(ax=ts\_ax)

ts\_ax.set\_title('Time Series Analysis Plots')

smt.graphics.plot\_acf(ts, lags=24, ax=acf\_ax, alpha=0.5)

smt.graphics.plot\_pacf(ts, lags=24, ax=pacf\_ax, alpha=0.5)

None

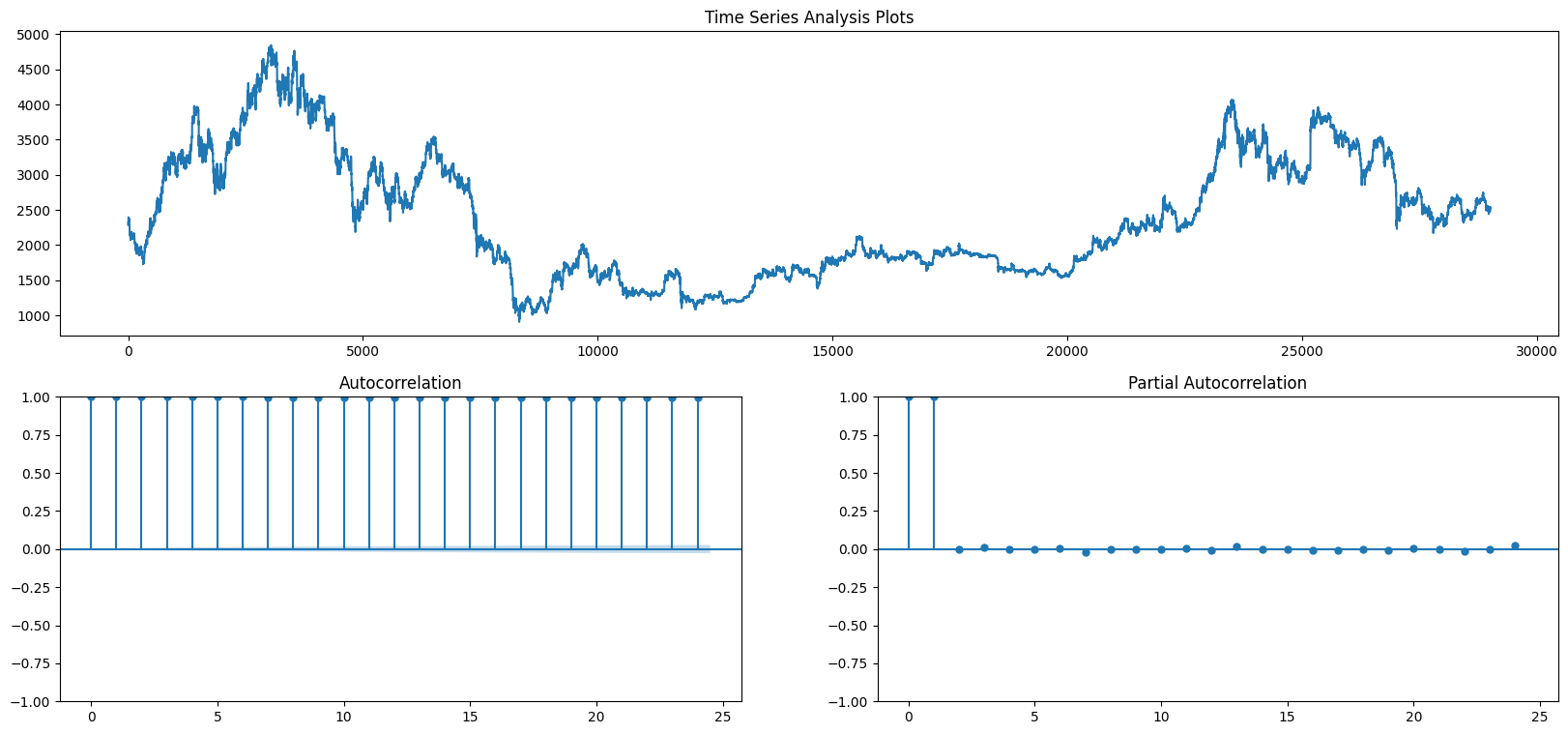


Рисунок 9. ACF и PACF целевой переменной

По графику видно, что все значения автокорреляционной функции (ACF) близки к 1, что указывает на сильную взаимосвязь между текущими и предыдущими значениями временного ряда на всех лагах. Это типичный признак нестационарности данных. В частичной автокорреляционной функции (PACF) высокие значения наблюдаются только на первых двух лагах, после чего они резко падают. Это может говорить о том, что временной ряд подвержен влиянию ограниченного количества предыдущих наблюдений, что характерно для рядов с трендом или интегрированных рядов.

Временной ряд называется ***стационарным в узком смысле***, если для любых моментов времени ,, … , и сдвига τ, вектор значений

(, ) совпадает по распределению с вектором (, ). То есть, совместное распределение значений ряда не изменяется при сдвиге моментов времени на одну и ту же величину τ.

Временной ряд ​ называется ***стационарным в широком смысле***, если выполняются следующие условия:

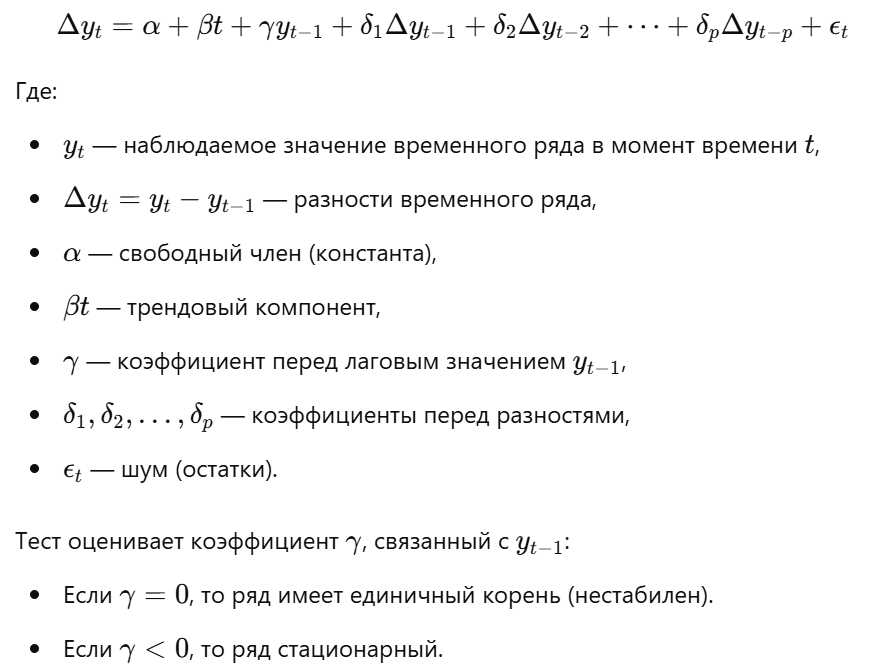
* E[] < +∞ для любого момента времени t, то есть дисперсия конечна.
* Математическое ожидание ряда E[] не зависит от времени t, то есть среднее значение ряда постоянно:

E[] = μ, ∀t

* Автоковариация значений временного ряда зависит только от временного лага между ними, а не от абсолютных моментов времени: Cov(, ​) = Cov(, ​) ​, ∀t,s,τ

То есть корреляция между значениями и ​ определяется только расстоянием ∣t − s∣.

Для проверки ряда на стационарность используют тест Дики-Фулера (ADF). Нулевая гипотеза (H₀): ряд имеет единичный корень, то есть он нестационарный. Альтернативная гипотеза (H₁): ряд стационарный, то есть не имеет единичного корня. Тест основан на следующем уравнении:



Проверим временные ряды на стационарность с помощью ADF:

from statsmodels.tsa.stattools import adfuller

def ADF(data, name, alpha=0.05):

    ts = data[name]

    print(f'Тест Дики-Фуллера ряда {name} :')

    dftest = adfuller(ts, autolag='AIC')

    dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-value','Lags Used','Number of Observations Used'])

    if dfoutput["p-value"] < alpha:

        print(f"Значение p меньше {alpha \* 100}%. Ряд стационарный.")

    else:

        print(f"Значение p больше {alpha\*100}%. Ряд не стационарный.")

for col in df.columns:

    ADF(df, col)

Результаты теста для колонок:

Тест Дики-Фуллера ряда btc\_open :

Значение p больше 5.0%. Ряд не стационарный.

Тест Дики-Фуллера ряда btc\_high :

Значение p больше 5.0%. Ряд не стационарный.

Тест Дики-Фуллера ряда btc\_low :

Значение p больше 5.0%. Ряд не стационарный.

Тест Дики-Фуллера ряда btc\_close :

Значение p больше 5.0%. Ряд не стационарный.

Тест Дики-Фуллера ряда btc\_volume :

Значение p меньше 5.0%. Ряд стационарный.

Тест Дики-Фуллера ряда btc\_rsi :

Значение p меньше 5.0%. Ряд стационарный.

Тест Дики-Фуллера ряда btc\_macd :

Значение p меньше 5.0%. Ряд стационарный.

Тест Дики-Фуллера ряда btc\_macd\_signal :

Значение p меньше 5.0%. Ряд стационарный.

Тест Дики-Фуллера ряда btc\_macd\_hist :

Значение p меньше 5.0%. Ряд стационарный.

Тест Дики-Фуллера ряда btc\_slowK :

Значение p меньше 5.0%. Ряд стационарный.

Тест Дики-Фуллера ряда btc\_slowD :

Значение p меньше 5.0%. Ряд стационарный.

Тест Дики-Фуллера ряда btc\_atr :

Значение p меньше 5.0%. Ряд стационарный.

Тест Дики-Фуллера ряда eth\_open :

Значение p больше 5.0%. Ряд не стационарный.

Тест Дики-Фуллера ряда eth\_high :

Значение p больше 5.0%. Ряд не стационарный.

Тест Дики-Фуллера ряда eth\_low :

Значение p больше 5.0%. Ряд не стационарный.

Тест Дики-Фуллера ряда eth\_close :

Значение p больше 5.0%. Ряд не стационарный.

Тест Дики-Фуллера ряда eth\_volume :

Значение p меньше 5.0%. Ряд стационарный.

Тест Дики-Фуллера ряда fear\_greed :

Значение p меньше 5.0%. Ряд стационарный.

Тест Дики-Фуллера ряда eth\_rsi :

Значение p меньше 5.0%. Ряд стационарный.

Тест Дики-Фуллера ряда eth\_macd :

Значение p меньше 5.0%. Ряд стационарный.

Тест Дики-Фуллера ряда eth\_macd\_signal :

Значение p меньше 5.0%. Ряд стационарный.

Тест Дики-Фуллера ряда eth\_macd\_hist :

Значение p меньше 5.0%. Ряд стационарный.

Тест Дики-Фуллера ряда eth\_slowK :

Значение p меньше 5.0%. Ряд стационарный.

Тест Дики-Фуллера ряда eth\_slowD :

Значение p меньше 5.0%. Ряд стационарный.

Тест Дики-Фуллера ряда eth\_atr :

Значение p меньше 5.0%. Ряд стационарный.

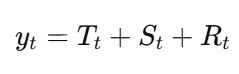
Тест Дики-Фуллера ряда fed\_rate :

Значение p больше 5.0%. Ряд не стационарный.

Помимо стационарности, важными характеристиками временного ряда являются тренд и сезонность. ***Тренд*** — это долгосрочная направленная динамика временного ряда, отражающая общую тенденцию изменения значений с течением времени. ***Сезонность*** — это повторяющиеся закономерности или циклы в данных, которые возникают с регулярными интервалами времени. Определение тренда и сезонности помогает выбрать подходящую модель прогнозирования, например, добавить сезонные компоненты в особенности цикла в LSTM.

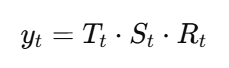
Модуль Модуль seasonal\_decompose из библиотеки statsmodels реализует декомпозицию временного ряда на три основные компоненты: тренд, сезонность, остаток. Декомпозиция временного ряда может быть выполнена двумя способами:

Аддитивная модель:

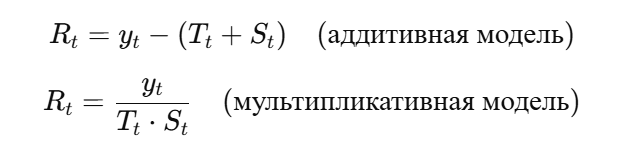


где yt​ — значение временного ряда в момент времени t, Tt​ — трендовая компонента, St — сезонная компонента, Rt​ — остаток.

Мультипликативная модель:



|  |  |
| --- | --- |
|  |  |



Где N — длина окна скользящего среднего, обычно равна длине периода сезонности; K — количество повторений полного цикла сезонности в данных.

Проведем декомпозицию ряда целевой переменной:

from statsmodels.tsa.seasonal import seasonal\_decompose

from pylab import rcParams

rcParams['figure.figsize'] = 11, 9

decompose = seasonal\_decompose(df['eth\_close'], period=24\*7)#неделя

decompose.plot()

plt.show()

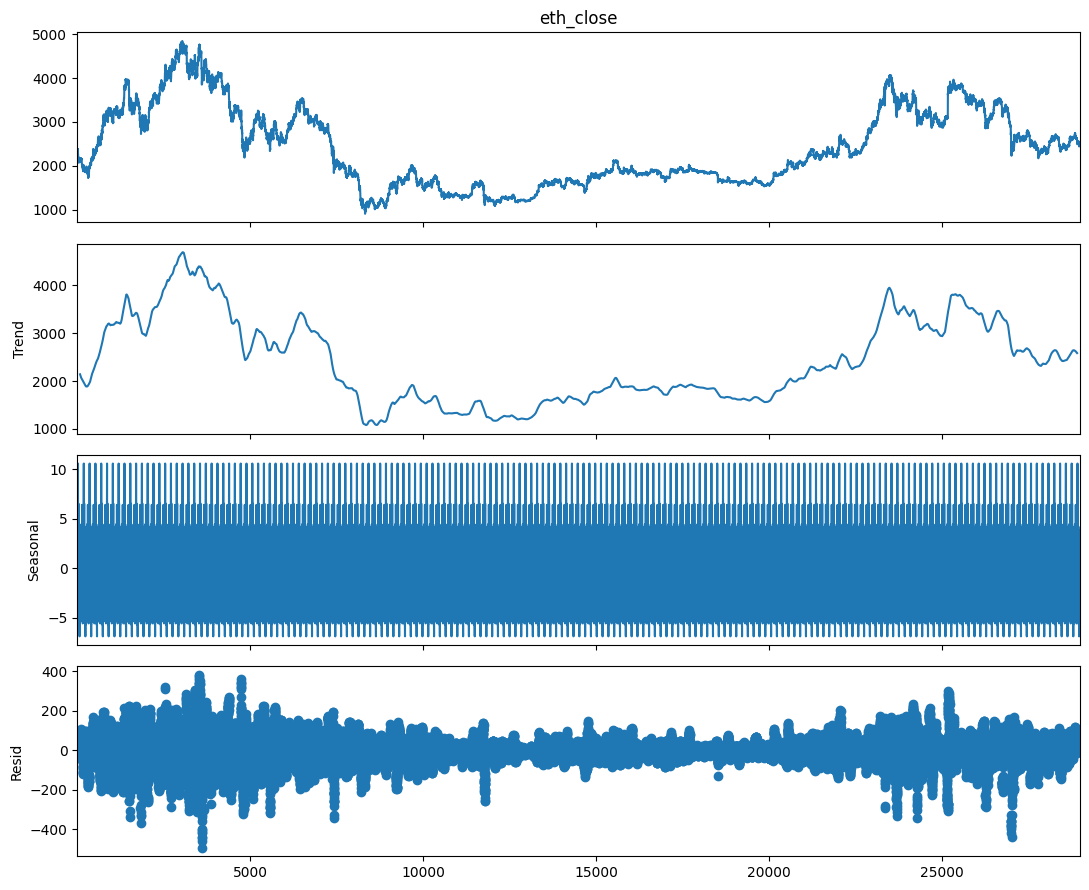


Рисунок 10. Декомпозиция целевой переменной

Результаты такого анализа можно было бы применить при построение статистических моделей (например, ARIMA, SARIMA, GARCH, VAR), однако, LSTM обучается непосредственно на оригинальных данных, включая тренд и сезонность, без необходимости их удаления, она автоматически обнаруживает временные зависимости любой длины, используя внутренние веса и рекуррентные слои, без ручной настройки лагов. Кроме этого, вместо дифференцирования для достижения стационарности, для LSTM достаточно стандартизировать данные.

Этап 5. Моделирование

Разобьём данные на обучающую и тестовую выборки. 75% данных будут находится в обучающей выборке, а 25% - тестовой:

train\_size = int(0.75 \* len(market\_data))

test\_size = len(market\_data) - train\_size

train\_data = market\_data[:train\_size]

test\_data = market\_data[train\_size:]

Масштабируем данные, используя MinMaxScaler:

from sklearn.preprocessing import MinMaxScaler

scaler\_X = MinMaxScaler()

scaler\_y = MinMaxScaler()

train\_data[X\_cols] = scaler\_X.fit\_transform(train\_data[X\_cols])

train\_data[y\_col] = scaler\_y.fit\_transform(train\_data[[y\_col]])

test\_data[X\_cols] = scaler\_X.transform(test\_data[X\_cols])

test\_data[y\_col] = scaler\_y.transform(test\_data[[y\_col]])

Функция TimeSeriesDataset – это класс, унаследованный от torch.utils.data.Dataset, который используется для подготовки данных временных рядов в формате, подходящем для обучения LSTM.

import torch

from torch.utils.data import Dataset

class TimeSeriesDataset(Dataset):

    def \_\_init\_\_(self, data, target\_column, sequence\_length=24):

        self.data = data

        self.target\_column = target\_column

        self.sequence\_length = sequence\_length

    # Возвращает общее количество доступных последовательностей в датасете

    def \_\_len\_\_(self):

        return len(self.data) - self.sequence\_length

    # Возвращает одну выборку из дата-сета

    def \_\_getitem\_\_(self, idx):

        # Извлекаем последовательность признаков длиной sequence\_length, исключая целевой столбец

        X = self.data.iloc[idx:idx + self.sequence\_length].drop(columns=[self.target\_column]).values

        # Извлекаем значение целевой переменной, соответствующее следующему значению после последовательности

        y = self.data.iloc[idx + self.sequence\_length][self.target\_column]

        # Преобразуем данные в тензоры PyTorch.

        return torch.tensor(X, dtype=torch.float32), torch.tensor(y, dtype=torch.float32)

        '''

        # X: Тензор PyTorch с признаками размером [sequence\_length,n\_features]

        # y: Тензор PyTorch с целевым значением (строка с индексом idx + sequence\_length)

        '''

Нейронная сеть LSTM (Long Short Term Memory) является подклассом рекуррентных нейронных сетей (RNN) (Hochreiter, Schmidhuber, 1997). Хотя RNN-сети имеют внутреннюю память для обработки последовательных данных, при обработке возникают проблемы с исчезновением градиента и его «взрывом». Сети с долговременной кратковременной памятью LSTM были специально разработаны для устранения этого ограничения. LSTM использует три элемента управления, включая входной элемент, элемент забывания и выходной элемент, для модуляции потока информации между ячейками и предотвращения исчезновения градиента и его резкого увеличения.

На рис. 1.2.1.1 представим рекуррентный модуль LSTM:

Изображение выглядит как текст, диаграмма, снимок экрана, графическая вставка

Автоматически созданное описание

Рисунок 11. Рекуррентный модуль LSTM

Формулы, описывающие вычисления на каждом временном шаге внутри LSTM-сети для одного примера последовательных данных:

,

,

,

,

,

,

где

– вентили LSTM для забывания, входа и выхода соответственно;

– новая кандидатская ячейка памяти;

​ – обновленная ячейка памяти;

– скрытое состояние;

- функции активации сигмоида и гиперболический тангенс, соответственно;

– байес (смещение) модели.

Таким образом, нейронная сеть LSTM является мощным инструментом для работы с последовательными данными, особенно в условиях, когда требуется учесть как краткосрочные, так и долгосрочные зависимости. Преимущество LSTM перед классическими RNN заключается в решении проблемы исчезновения или взрывного роста градиента, что позволяет этой архитектуре эффективно обучаться на длинных временных рядах. Благодаря введению механизмов управления потоком информации через вентильные структуры (входной, забывающий и выходной вентиль), LSTM-сети могут эффективно сохранять и обновлять информацию на протяжении длинных временных интервалов.

LSTM-модели обладают множеством гиперпараметров, которые напрямую влияют на её способность обучаться и прогнозировать временные ряды. Чтобы упростить процесс настройки модели, мы зафиксировали некоторые из них, основываясь на их стандартных значениях для задач прогнозирования временных рядов. Однако часть гиперпараметров осталась переменной, чтобы настроить модель наилучшим образом:

* 1. Гиперпараметры, определяющие архитектуру модели:
* input\_size: определяет размер входного вектора в каждый момент времени (количество признаков) – этот параметр не настраивается, так как зависит от данных;
* num\_layers: количество слоёв LSTM, отвечает за сложность удавливаемых зависимостей – обычно используется от 1 до 3 скрытых слоев, в работе это число зафиксировано и равно 2, так как данные имеют достаточно сложной структурой, но их количество не достаточно велико;
* hidden\_size: размер скрытого слоя LSTM, который определяет, сколько информации модель может хранить и обрабатывать (большие значения позволяют модели запоминать сложные зависимости, но увеличивают риск переобучения) – в работе подбираем от 16 до 128 с шагом в 8;
* fc\_size: число нейронов в промежуточном полносвязном слое, определяет степень преобразования выходных данных LSTM перед финальным прогнозом - в работе подбираем от 16 до 64 с шагом в 8;
* output\_size: размерность выходного вектора - для регрессии это обычно 1, так как предсказывается одно числовое значение (цена закрытия).

2. Гиперпараметры, определяющие процесс обучения:

* learning\_rate: скорость обучения, определяет, насколько сильно обновляются веса модели на каждом шаге – в работе используются значения 0.0001 и 0.001;
* optimizer: алгоритм оптимизации – в данном исследовании применяется только Adam, так как Adam хорошо подходит для LSTM и адаптируется к динамике данных;
* loss\_function: функция потерь, измеряющая разницу между прогнозом и истинным значением - для регрессии обычно используется среднеквадратическая ошибка ();
* sequence\_length: длина последовательности временного ряда, которая подаётся на вход модели, определяет, сколько предыдущих временных шагов используется для предсказания – в работе используются значения 24, 48, 72 (день, 2 дня, 3 дня);
* batch\_size: количество последовательностей, обрабатываемых одновременно, обычно используются числа кратные 8 для ускорения обучения на видеокартах – в данной работе значение 32;
* dropout\_rate: вероятность исключения нейронов во время обучения, регулирует уровень регуляризации для предотвращения переобучения – значения подбирается из диапазона от 0.1 до 0.5 с шагом 0.1;
* num\_epochs: количество эпох обучения, определяет, сколько раз весь тренировочный набор данных будет пропущен через модель во время обучения – в данной работе варьируется от 50 до 100.

Для оценки качества моделей рассматриваем коэффициент детерминации. измеряет долю дисперсии зависимой переменной, объясняемой моделью. Значение варьируется от 0 до 1, где 1 означает идеальное соответствие модели данным:

,

Выполним построение модели LSTM с использованием библиотеки PyTorch, настройку гиперпараметров с помощью Optuna, обучение модели и оценку её производительности на основе коэффициента детерминации:

import torch

import torch.nn as nn

import torch.optim as optim

from torch.utils.data import DataLoader, TensorDataset

from torchmetrics import R2Score

import optuna

# Определение архитектуры модели LSTM

class LSTMModel(nn.Module):

    def \_\_init\_\_(self, input\_size, hidden\_size1, hidden\_size2, fc\_size, output\_size, dropout\_rate):

        super(LSTMModel, self).\_\_init\_\_()

        # Первый LSTM слой

        self.lstm1 = nn.LSTM(input\_size, hidden\_size1, batch\_first=True)

        self.dropout1 = nn.Dropout(dropout\_rate)  # Dropout для регуляризации

        # Второй LSTM слой

        self.lstm2 = nn.LSTM(hidden\_size1, hidden\_size2, batch\_first=True)

        self.dropout2 = nn.Dropout(dropout\_rate)  # Dropout для регуляризации

        # Полносвязные слои (выходная часть модели)

        self.fc1 = nn.Linear(hidden\_size2, fc\_size)  # Скрытый слой

        self.fc2 = nn.Linear(fc\_size, output\_size)  # Выходной слой

        # Активационная функция (сигмоида)

        self.sigmoid = nn.Sigmoid()

    def forward(self, x):

        # Прямой проход через первый LSTM слой

        out, \_ = self.lstm1(x)

        out = self.dropout1(out)  # Регуляризация

        # Прямой проход через второй LSTM слой

        out, \_ = self.lstm2(out)

        out = self.dropout2(out)  # Регуляризация

        # Использование последнего временного шага

        out = self.fc1(out[:, -1, :])  # Преобразование через скрытый слой

        out = self.fc2(out)  # Преобразование через выходной слой

        # Возвращение выхода через сигмоидальную функцию

        return self.sigmoid(out)

# Целевая функция для оптимизации гиперпараметров

def objective(trial):

    # Поиск оптимальных гиперпараметров

    hidden\_size1 = trial.suggest\_int("hidden\_size1", 16, 128, step=8)  # Размер скрытых состояний 1-го LSTM

    hidden\_size2 = trial.suggest\_int("hidden\_size2", 16, 128, step=8)  # Размер скрытых состояний 2-го LSTM

    fc\_size = trial.suggest\_int("fc\_size", 16, 64, step=8)  # Размер полносвязного слоя

    dropout\_rate = trial.suggest\_float("dropout\_rate", 0.1, 0.5, step=0.1)  # Вероятность Dropout

    learning\_rate = trial.suggest\_loguniform("learning\_rate", 1e-4, 1e-3)  # Скорость обучения

    # Создание модели с текущими гиперпараметрами

    model = LSTMModel(

        input\_size=25,  # Количество признаков во входных данных

        hidden\_size1=hidden\_size1,

        hidden\_size2=hidden\_size2,

        fc\_size=fc\_size,

        output\_size=1,  # Прогнозируемое значение (целевая переменная)

        dropout\_rate=dropout\_rate

    ).to(device)

    # Оптимизатор и функция потерь

    optimizer = torch.optim.Adam(model.parameters(), lr=learning\_rate)  # Adam оптимизатор

    loss\_fn = nn.MSELoss()  # Среднеквадратичная ошибка

    num\_epochs = 30  # Количество эпох для обучения

    train\_r2 = R2Score().to(device)  # Метрика для оценки R^2

    model.train()

    # Обучение модели

    for epoch in range(num\_epochs):

        for X, y in train\_loader:

            X, y = X.to(device), y.to(device).unsqueeze(-1)  # Перемещение данных на устройство (GPU/CPU)

            optimizer.zero\_grad()  # Сброс градиентов

            y\_pred = model(X)  # Предсказания модели

            loss = loss\_fn(y\_pred, y)  # Вычисление функции потерь

            loss.backward()  # Обратное распространение ошибки

            optimizer.step()  # Обновление весов модели

    # Оценка модели на валидационной выборке

    model.eval()

    val\_r2 = R2Score().to(device)

    with torch.no\_grad():  # Отключение вычисления градиентов

        for X, y in val\_loader:

            X, y = X.to(device), y.to(device).unsqueeze(-1)

            y\_pred = model(X)

            val\_r2.update(y\_pred, y)

    # Возврат значения метрики R^2

    return val\_r2.compute().item()

# Устройство для вычислений (GPU, если доступен)

device = torch.device("cuda" if torch.cuda.is\_available() else "cpu")

# Даталоадеры для тренировочных и валидационных данных

train\_loader = DataLoader(train\_dataset, batch\_size=32, shuffle=True)  # Обучающая выборка

val\_loader = DataLoader(val\_dataset, batch\_size=32, shuffle=False)  # Валидационная выборка

# Настройка гиперпараметров с использованием Optuna

study = optuna.create\_study(direction="maximize")  # Максимизация метрики R^2

study.optimize(objective, n\_trials=50)  # Проведение 50 итераций поиска гиперпараметров

# Вывод лучших параметров

print(study.best\_params)

Далее в таблице представлены модели и параметры, которые обеспечивают наиболее высокий коэффициент детерминации:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **(test)** | **Размер окна** | **Размер первого слоя LSTM** | **Размер второго слоя LSTM** | **Размер полносвязного линейного слоя** | **Dropout** | **Кол-во эпох** | **Скорость обучения** |
| 0.87 | ***24*** | 48 | 24 | 16 | 0.4 | ***100*** | 0.001 |
| 0.72 | ***24*** | 48 | 24 | 16 | 0.4 | ***50*** | 0.001 |
| 0.61 | ***48*** | 48 | 24 | 16 | 0.4 | ***100*** | 0.001 |
| 0.50 | ***48*** | 48 | 24 | 16 | 0.4 | ***50*** | 0.001 |
| 0.35 | ***72*** | 48 | 24 | 16 | 0.4 | ***100*** | 0.001 |

Визуализируем результаты лучшей модели:

start\_date = "2023-12-30 07:00:00"

freq = "1H"

num\_points = len(predictions\_real)

data\_timestamps = pd.date\_range(start=start\_date, periods=num\_points, freq=freq)

plt.figure(figsize=(12, 7))

plt.plot(data\_timestamps, actuals\_real, label="Истинные значения (цены закрытия)", color="blue", alpha=0.7)

plt.plot(data\_timestamps, predictions\_real, label="Предсказания", color="red", alpha=0.7)

plt.title("Истинные цены закрытия и предсказания ETH/USDT", fontweight='bold', fontsize=14)

plt.xlabel("Дата", fontweight='bold', fontsize=12)

plt.ylabel("Цена закрытия (USDT)", fontweight='bold', fontsize=12)

plt.xticks(rotation=45)

plt.grid(True)

plt.legend(loc="upper right")

plt.tight\_layout()

plt.show()



Рисунок 12. Результаты предсказания лучшей модели

На графиках видно, что модель успешно уловила основные закономерности в данных. Тем не менее, в октябре-ноябре 2024 года наблюдается увеличение расхождения между прогнозными и реальными значениями, несмотря на правильный общий паттерн поведения предсказанных значений.



Рисунок 13. Рост ошибки прогноза

На Рисунке 13 видно, что ошибка возросла в указанный период. Это связано с тем, что нормализация данных была проведена на основе минимального и максимального значения обучающей выборки. В октябре-ноябре 2024 года цена BTC превысила максимальное значение, использованное при нормализации, что вывело значения целевой переменной за пределы обученного диапазона модели. Как следствие, модель стала давать менее точные прогнозы в этом периоде.



Рисунок 14. Цена BTC превысила максимальное значение, использованное для нормализации

Для минимизации подобных эффектов в будущем будет обращено особое внимание на данную проблему, которую можно решить следующими способами:

1. Использовать расширенный диапазон нормализации, увеличив минимум и максимум, чтобы учесть потенциальные экстремумы.
2. Периодически обновлять параметры нормализации на основе новых данных, чтобы модель могла адаптироваться к изменяющимся рыночным условиям.

Заключение

Проект «Построение модели LSTM для прогнозирования цен закрытия часовых интервалов ETH/USDT» показал частичное достижение поставленных целей. Были собраны и проанализированы исторические данные о ценах закрытия криптовалюты ETH/USDT, проведён предварительный анализ, который подтвердил наличие сильных временных зависимостей между объясняющими и целевой переменной, что позволило разработать и обучить модель на основе архитектуры LSTM. Показано, что эта модель может эффективно улавливать паттерны для предсказания цены закрытия ETH на часовых интервалах. Разработанная модель, адаптированная для работы с временными рядами, была настроена с учётом ключевых гиперпараметров, таких как размерность скрытых слоёв, скорость обучения, количество эпох и уровень Dropout. Благодаря этому модель достигла коэффициента детерминации до 0.87 на тестовых данных, что подтверждает её высокую точность. Однако в октябре–ноябре 2024 года наблюдалось увеличение ошибки прогнозов из-за выхода значений за пределы диапазона нормализации, что указывает на необходимость доработок. Для повышения точности модели в будущем планируется пересматривать параметры нормализации с учётом новых данных, а также рассмотреть использование альтернативных архитектур, таких как Transformer или GRU. Периодическое переобучение модели также может улучшить её адаптацию к изменяющимся рыночным условиям.

Практическая значимость проекта заключается в том, что разработанная модель может быть интегрирована в аналитические системы трейдеров и инвесторов, способствуя более обоснованному принятию решений, минимизации рисков и увеличению прибыльности операций. В целом, результаты проекта подтверждают эффективность применения моделей LSTM для прогнозирования цен криптовалют и открывают перспективы для дальнейших исследований в этой области.

Список использованных источников и литературы

1. Абдукаева А. А. и др. ПРОГНОСТИЧЕСКИЕ МОДЕЛИ РАЗВИТИЯ РЫНКА КРИПТОВАЛЮТЫ (ARMA, LSTM): СРАВНИТЕЛЬНЫЙ АНАЛИЗ //Казанский экономический вестник. – 2019. – №. 6. – С. 88-96.
2. Аггарвал Ч. Нейронные сети и глубокое обучение: учебный курс //Пер. с англ.–СПб.: ООО «Диалектика. – 2020.
3. Алжеев А. В., Кочкаров Р. А. Сравнительный анализ прогнозных моделей ARIMA и LSTM на примере акций российских компаний //Финансы: теория и практика. – 2020. – Т. 24. – №. 1. – С. 14-23.
4. Амоа К.А. Разработка программных пакетов на языке Python: учебное пособие / К.А. Амоа, Н.А. Рындин, Ю.С. Скворцов. – Воронеж: Воронежский государственный технический университет, ЭБС АСВ, 2020. – 61 c. // Электронно-библиотечная система IPR BOOKS: [сайт].
5. Антонопулос А. Осваиваем биткоин. Программирование блокчейна. – Litres, 2022.
6. Афанасьев В. Н., Юзбашев М. М. Анализ временных рядов и прогнозирование //М.: Финансы и статистика. – 2001. – Т. 228. – №. 2.
7. Вандер П. Python для сложных задач: наука о данных и машинное обучение. — СПб.: Питер, 2018. — 576 с.
8. Васильев, А.Н. Программирование на Python в примерах и задачах / А.Н. Васильев. — Москва: Эксмо, 2021. — 616 с.
9. Видмант О. С. Прогнозирование финансовых временных рядов с использованием рекуррентных нейронных сетей LSTM //Общество: политика, экономика, право. – 2018. – №. 5. – С. 63-66.
10. Вильямс Л. Долгосрочные секреты краткосрочной торговли. – Litres, 2022.
11. Волкова В.М., Программные системы статистического анализа. Обнаружение закономерностей в данных с использованием системы R и языка Python [Электронный ресурс]: учебное пособие / Волкова В.М. - Новосибирск: Изд-во НГТУ, 2017. - 74 с. - ISBN 978-5-7782-3183-2 - Режим доступа: http://www.studentlibrary.ru/book/ISBN9785778231832.html
12. Кильдишев Г. С., Френкель А. А. Анализ временных рядов и прогнозирование. – Общество с ограниченной ответственностью" Ленанд", 2021. – С. 104-104.
13. Кондратьева Т. Н. Прогнозирование тенденции финансовых временных рядов с помощью нейронной сети LSTM //Вестник евразийской науки. – 2017. – Т. 9. – №. 4 (41). – С. 61.
14. Николенко С., Кадурин А., Архангельская Е. Глубокое обучение. – Издательский дом Питер, 2017.
15. Ростовцев В. С. Искусственные нейронные сети: учебник для вузов //Санкт-Петербург: Лань. – 2021.
16. Тарик Р. Создаём нейронную сеть. Пер. с англ //СПб.: ООО “Альфа-книга. – 2017.
17. Хайкин С. Нейронные сети: полный курс, 2-е издание. – Издательский дом Вильямс, 2008.
18. Швагер Д. Технический анализ: Полный курс. – Альпина Паблишер, 2017.
19. Hamayel M. J., Owda A. Y. A novel cryptocurrency price prediction model using GRU, LSTM and bi-LSTM machine learning algorithms //Ai. – 2021. – Т. 2. – №. 4. – С. 477-496.
20. Hochreiter S. Long Short-term Memory //Neural Computation MIT-Press. – 1997.
21. Seabe P. L., Moutsinga C. R. B., Pindza E. Forecasting cryptocurrency prices using LSTM, GRU, and bi-directional LSTM: a deep learning approach //Fractal and Fractional. – 2023. – Т. 7. – №. 2. – С. 203.
22. Sunny M. A. I., Maswood M. M. S., Alharbi A. G. Deep learning-based stock price prediction using LSTM and bi-directional LSTM model //2020 2nd novel intelligent and leading emerging sciences conference (NILES). – IEEE, 2020. – С. 87-92.

Приложения

Приложение 1

Программный код загрузки данных

### Библиотеки

**import** time

**import** requests

**import** ccxt

**import** pandas **as** pd

**import** talib

# Функция для загрузки данных с Bybit

**def** fetch\_ohlcv**(**exchange**,** symbol**,** timeframe**,** start\_date**,** end\_date**):**

since **=** exchange**.**parse8601**(**start\_date**)**

now **=** exchange**.**parse8601**(**end\_date**)**

all\_ohlcv **=** **[]**

**while** since **<** now**:**

ohlcv **=** exchange**.**fetch\_ohlcv**(**symbol**,** timeframe**,** since**,** limit**=**200**)**

**if** **not** ohlcv**:**

**break**

all\_ohlcv**.**extend**(**ohlcv**)**

since **=** ohlcv**[-**1**][**0**]** **+** 1

time**.**sleep**(**1**)** # Учитываем ограничение запросов

df **=** pd**.**DataFrame**(**all\_ohlcv**,** columns**=[**'timestamp'**,** 'open'**,** 'high'**,** 'low'**,** 'close'**,** 'volume'**])**

df**[**'timestamp'**]** **=** pd**.**to\_datetime**(**df**[**'timestamp'**],** unit**=**'ms'**)**

**return** df

# API-ключи для Bybit

api\_key **=** "api\_key" # Укажите ключ для Bybit

api\_secret **=** "api\_secret" # Укажите секрет для Bybit

# Инициализация соединения с Bybit

exchange **=** ccxt**.**bybit**({**

'apiKey'**:** api\_key**,**

'secret'**:** api\_secret**,**

'enableRateLimit'**:** **True**

**})**

# Загрузка данных ETH и BTC

df\_ETH **=** fetch\_ohlcv**(**exchange**,** 'ETH/USDT'**,** '1h'**,** '2021-07-01T00:00:00Z'**,** '2024-10-28T00:00:00Z'**)**

df\_BTC **=** fetch\_ohlcv**(**exchange**,** 'BTC/USDT'**,** '1h'**,** '2021-07-01T00:00:00Z'**,** '2024-10-28T00:00:00Z'**)**

# Загрузка данных ставки ФРС

url\_frs **=** f"https://api.stlouisfed.org/fred/series/observations?series\_id=FEDFUNDS&api\_key={api\_key}&file\_type=json"

response\_frs **=** requests**.**get**(**url\_frs**)**

**if** response\_frs**.**status\_code **==** 200**:**

data\_frs **=** response\_frs**.**json**()**

df\_FRS **=** pd**.**DataFrame**(**data\_frs**[**'observations'**])**

df\_FRS**[**'date'**]** **=** pd**.**to\_datetime**(**df\_FRS**[**'date'**])**

df\_FRS **=** df\_FRS**[[**'date'**,** 'value'**]]**

df\_FRS **=** df\_FRS**.**rename**(**columns**={**"value"**:** "fed\_rate"**})**

df\_FRS**[**'year\_month'**]** **=** df\_FRS**[**'date'**].**dt**.**to\_period**(**'M'**).**astype**(str)**

# Загрузка индекса страха и жадности (VIX)

url\_vix **=** "https://api.alternative.me/fng/?limit=0&format=json"

response\_vix **=** requests**.**get**(**url\_vix**)**

**if** response\_vix**.**status\_code **==** 200**:**

fear\_greed\_data **=** response\_vix**.**json**()**

df\_VIX **=** pd**.**DataFrame**(**fear\_greed\_data**[**"data"**])**

df\_VIX**[**'date'**]** **=** pd**.**to\_datetime**(**df\_VIX**[**'timestamp'**],** unit**=**'s'**)**

df\_VIX **=** df\_VIX**.**drop**(**columns**=[**'value\_classification'**,** 'timestamp'**,** 'time\_until\_update'**])**

df\_VIX **=** df\_VIX**.**rename**(**columns**={**"value"**:** "fear\_greed\_index"**})**

# Функция для добавления технических индикаторов в DataFrame

**def** add\_technical\_indicators**(**df**,** symbol\_prefix**):**

# RSI (Relative Strength Index)

df**[**f'{symbol\_prefix}\_rsi'**]** **=** talib**.**RSI**(**df**[**f'{symbol\_prefix}\_close'**],** timeperiod**=**14**)**

# MACD (Moving Average Convergence Divergence)

df**[**f'{symbol\_prefix}\_macd'**],** df**[**f'{symbol\_prefix}\_macd\_signal'**],** df**[**f'{symbol\_prefix}\_macd\_hist'**]** **=** talib**.**MACD**(**

df**[**f'{symbol\_prefix}\_close'**],** fastperiod**=**12**,** slowperiod**=**26**,** signalperiod**=**9

**)**

# Stochastic Oscillator (STOCH)

df**[**f'{symbol\_prefix}\_slowK'**],** df**[**f'{symbol\_prefix}\_slowD'**]** **=** talib**.**STOCH**(**

df**[**f'{symbol\_prefix}\_high'**],** df**[**f'{symbol\_prefix}\_low'**],** df**[**f'{symbol\_prefix}\_close'**],**

fastk\_period**=**14**,** slowk\_period**=**3**,** slowk\_matype**=**0**,** slowd\_period**=**3**,** slowd\_matype**=**0

**)**

# ATR (Average True Range)

df**[**f'{symbol\_prefix}\_atr'**]** **=** talib**.**ATR**(**

df**[**f'{symbol\_prefix}\_high'**],** df**[**f'{symbol\_prefix}\_low'**],** df**[**f'{symbol\_prefix}\_close'**],** timeperiod**=**14

**)**

**return** df

# Переименование столбцов и добавление технических индикаторов для BTC

df\_BTC **=** add\_technical\_indicators**(**df\_BTC**.**rename**(**columns**={**

'open'**:** 'btc\_open'**,** 'high'**:** 'btc\_high'**,** 'low'**:** 'btc\_low'**,** 'close'**:** 'btc\_close'**,** 'volume'**:** 'btc\_volume'

**}),** 'btc'**)**

# Переименование столбцов и добавление технических индикаторов для ETH

df\_ETH **=** add\_technical\_indicators**(**df\_ETH**.**rename**(**columns**={**

'open'**:** 'eth\_open'**,** 'high'**:** 'eth\_high'**,** 'low'**:** 'eth\_low'**,** 'close'**:** 'eth\_close'**,** 'volume'**:** 'eth\_volume'

**}),** 'eth'**)**

# Слияние данных ETH и BTC по дате

df\_market **=** pd**.**merge**(**df\_ETH**,** df\_BTC**,** on**=**'timestamp'**,** how**=**'inner'**)**

# Добавление данных ставки ФРС

df\_FRS **=** df\_FRS**.**rename**(**columns**={**'date'**:** 'timestamp'**})** # Переименование 'date' в 'timestamp'

df\_market **=** pd**.**merge\_asof**(** #данные ФРС имеют меньшую частоту обновления

df\_market**.**sort\_values**(**'timestamp'**),**

df\_FRS**.**sort\_values**(**'timestamp'**),**

on**=**'timestamp'**,**

direction**=**'backward' # добавляет ближайшее предыдущее значение ставки

**)**

# Добавление данных индекса страха и жадности (VIX)

df\_VIX **=** df\_VIX**.**rename**(**columns**={**'date'**:** 'timestamp'**})** # Переименование 'date' в 'timestamp'

df\_market **=** pd**.**merge\_asof**(**

df\_market**.**sort\_values**(**'timestamp'**),**

df\_VIX**.**sort\_values**(**'timestamp'**),**

on**=**'timestamp'**,**

direction**=**'backward'

**)**

# Сохранение итогового DataFrame в CSV

output\_file **=** "market\_project\_data\_usd.csv"

df\_market**.**to\_csv**(**output\_file**,** index**=False)**

Приложение 2

Программный код анализа данных и моделирования

### Библиотеки

**import** pandas **as** pd

**import** numpy **as** np

**from** sklearn**.**preprocessing **import** MinMaxScaler

**import** torch

**from** torch**.**utils**.**data **import** Dataset**,** DataLoader

**import** torch**.**nn **as** nn

**import** torch**.**optim **as** optim

**from** torchmetrics **import** MeanSquaredError**,** R2Score

**import** time

**from** matplotlib **import** pyplot **as** plt

**import** seaborn **as** sns

**import** scipy**.**stats **as** stats

**import** statsmodels**.**tsa**.**api **as** smt

**from** statsmodels**.**tsa**.**stattools **import** adfuller

**from** statsmodels**.**tsa**.**seasonal **import** seasonal\_decompose

**from** pylab **import** rcParams

**import** optuna

### Готовим данные

market\_data **=** pd**.**read\_csv**(**'C:/Projects/vsCode/ml\_unik\_project/data/market\_project\_data\_usd.csv'**,** sep**=**','**)**

market\_data**.**info**()**

# Можем удалить timestamp, т.к. данные уже отсортированы

market\_data **=** market\_data**.**drop**([**'timestamp'**],** axis**=**1**)**

market\_data**.**info**()**

# Вывод графика

plt**.**plot**(**market\_data**[**'eth\_close'**])**

plt**.**show**()**

### Предварительный анализ данных

# Загрузка данных

df **=** market\_data**.**copy**()**

# Описательные статистики

df**.**describe**()**

# Box-plot и гистограммы

df**.**boxplot**();**

df**.**hist**(**figsize**=(**20**,**15**),** bins**=**30**)**

# Проверка нормальности распределения (тест Шапиро-Уилка)

**for** col **in** df**.**columns**:**

**print(**col**,** stats**.**shapiro**(**df**[**col**]))**

# Матрица корреляций (Спирмена)

plt**.**rcParams**[**"figure.figsize"**]** **=** 8**,** 6

sns**.**heatmap**(**df**.**corr**(**method**=**"spearman"**,** numeric\_only**=True),** annot**=True)**

# Выбор целевой переменной и вывод корреляций для нее

y\_col **=** 'eth\_close'

**def** print\_correlations**(**data**,** target**):**

    correlations **=** data**.**corr**(**method**=**"spearman"**)[**target**].**sort\_values**(**ascending**=False)**

**print(**f"Коэффициенты корреляции с {target}:"**)**

**print(**"-" **\*** 50**)**

**for** feature**,** corr **in** correlations**.**items**():**

**print(**f"{feature:20}: {corr:.4f}"**)**

print\_correlations**(**df**,** y\_col**)**

# Анализ автокорреляций (графики ACF и PACF)

fig **=** plt**.**figure**(**figsize**=(**20**,** 9**))**

layout **=** **(**2**,** 2**)**

ts\_ax **=** plt**.**subplot2grid**(**layout**,** **(**0**,** 0**),** colspan**=**2**)**

acf\_ax **=** plt**.**subplot2grid**(**layout**,** **(**1**,** 0**))**

pacf\_ax **=** plt**.**subplot2grid**(**layout**,** **(**1**,** 1**))**

ts **=** df**[**y\_col**]**

ts**.**plot**(**ax**=**ts\_ax**)**

ts\_ax**.**set\_title**(**'Time Series Analysis Plots'**)**

smt**.**graphics**.**plot\_acf**(**ts**,** lags**=**24**,** ax**=**acf\_ax**,** alpha**=**0.5**)**

smt**.**graphics**.**plot\_pacf**(**ts**,** lags**=**24**,** ax**=**pacf\_ax**,** alpha**=**0.5**)**

**None**

# Анализ стационарности (тест Дики-Фулера или ADF)

**def** ADF**(**data**,** name**,** alpha**=**0.05**):**

    ts **=** data**[**name**]**

**print(**f'Тест Дики-Фуллера ряда {name} :'**)**

    dftest **=** adfuller**(**ts**,** autolag**=**'AIC'**)**

    dfoutput **=** pd**.**Series**(**dftest**[**0**:**4**],** index**=[**'Test Statistic'**,**'p-value'**,**'Lags Used'**,**'Number of Observations Used'**])**

**if** dfoutput**[**"p-value"**]** **<** alpha**:**

**print(**f"Значение p меньше {alpha **\*** 100}%. Ряд стационарный."**)**

**else:**

**print(**f"Значение p больше {alpha**\***100}%. Ряд не стационарный."**)**

**for** col **in** df**.**columns**:**

    ADF**(**df**,** col**)**

# Визуализация трендовой и сезонной состовляющей

rcParams**[**'figure.figsize'**]** **=** 11**,** 9

decompose **=** seasonal\_decompose**(**df**[**'eth\_close'**],** period**=**24**\***7**)**#неделя

decompose**.**plot**()**

plt**.**show**()**

## Подготовка данных к моделированию

# Разделение на train-test

train\_size **=** **int(**0.75 **\*** **len(**market\_data**))**

test\_size **=** **len(**market\_data**)** **-** train\_size

train\_data **=** market\_data**[:**train\_size**]**

test\_data **=** market\_data**[**train\_size**:]**

# Масштабирование

scaler\_X **=** MinMaxScaler**()**

scaler\_y **=** MinMaxScaler**()**

train\_data**[**X\_cols**]** **=** scaler\_X**.**fit\_transform**(**train\_data**[**X\_cols**])**

train\_data**[**y\_col**]** **=** scaler\_y**.**fit\_transform**(**train\_data**[[**y\_col**]])**

test\_data**[**X\_cols**]** **=** scaler\_X**.**transform**(**test\_data**[**X\_cols**])**

test\_data**[**y\_col**]** **=** scaler\_y**.**transform**(**test\_data**[[**y\_col**]])**

# Подготовка данных временных рядов в формате, подходящем для обучения LSTM

**class** **TimeSeriesDataset(**Dataset**):**

**def** \_\_init\_\_**(**self**,** data**,** target\_column**,** sequence\_length**=**24**):**

        self**.**data **=** data

        self**.**target\_column **=** target\_column

        self**.**sequence\_length **=** sequence\_length

    # Возвращает общее количество доступных последовательностей в датасете

**def** \_\_len\_\_**(**self**):**

**return** **len(**self**.**data**)** **-** self**.**sequence\_length

    # Возвращает одну выборку из дата-сета

**def** \_\_getitem\_\_**(**self**,** idx**):**

        # Извлекаем последовательность признаков длиной sequence\_length, исключая целевой столбец

        X **=** self**.**data**.**iloc**[**idx**:**idx **+** self**.**sequence\_length**].**drop**(**columns**=[**self**.**target\_column**]).**values

        # Извлекаем значение целевой переменной, соответствующее следующему значению после последовательности

        y **=** self**.**data**.**iloc**[**idx **+** self**.**sequence\_length**][**self**.**target\_column**]**

        # Преобразуем данные в тензоры PyTorch.

**return** torch**.**tensor**(**X**,** dtype**=**torch**.**float32**),** torch**.**tensor**(**y**,** dtype**=**torch**.**float32**)**

        '''

        # X: Тензор PyTorch с признаками размером [sequence\_length,n\_features]

        # y: Тензор PyTorch с целевым значением (строка с индексом idx + sequence\_length)

        '''

## Моделирование

# Определение архитектуры модели LSTM

**class** **LSTMModel(**nn**.**Module**):**

**def** \_\_init\_\_**(**self**,** input\_size**,** hidden\_size1**,** hidden\_size2**,** fc\_size**,** output\_size**,** dropout\_rate**):**

**super(**LSTMModel**,** self**).**\_\_init\_\_**()**

        # Первый LSTM слой

        self**.**lstm1 **=** nn**.**LSTM**(**input\_size**,** hidden\_size1**,** batch\_first**=True)**

        self**.**dropout1 **=** nn**.**Dropout**(**dropout\_rate**)**  # Dropout для регуляризации

        # Второй LSTM слой

        self**.**lstm2 **=** nn**.**LSTM**(**hidden\_size1**,** hidden\_size2**,** batch\_first**=True)**

        self**.**dropout2 **=** nn**.**Dropout**(**dropout\_rate**)**  # Dropout для регуляризации

        # Полносвязные слои (выходная часть модели)

        self**.**fc1 **=** nn**.**Linear**(**hidden\_size2**,** fc\_size**)**  # Скрытый слой

        self**.**fc2 **=** nn**.**Linear**(**fc\_size**,** output\_size**)**  # Выходной слой

        # Активационная функция (сигмоида)

        self**.**sigmoid **=** nn**.**Sigmoid**()**

**def** forward**(**self**,** x**):**

        # Прямой проход через первый LSTM слой

        out**,** \_ **=** self**.**lstm1**(**x**)**

        out **=** self**.**dropout1**(**out**)**  # Регуляризация

        # Прямой проход через второй LSTM слой

        out**,** \_ **=** self**.**lstm2**(**out**)**

        out **=** self**.**dropout2**(**out**)**  # Регуляризация

        # Использование последнего временного шага

        out **=** self**.**fc1**(**out**[:,** **-**1**,** **:])**  # Преобразование через скрытый слой

        out **=** self**.**fc2**(**out**)**  # Преобразование через выходной слой

        # Возвращение выхода через сигмоидальную функцию

**return** self**.**sigmoid**(**out**)**

# Целевая функция для оптимизации гиперпараметров

**def** objective**(**trial**):**

    # Поиск оптимальных гиперпараметров

    hidden\_size1 **=** trial**.**suggest\_int**(**"hidden\_size1"**,** 16**,** 128**,** step**=**8**)**  # Размер скрытых состояний 1-го LSTM

    hidden\_size2 **=** trial**.**suggest\_int**(**"hidden\_size2"**,** 16**,** 128**,** step**=**8**)**  # Размер скрытых состояний 2-го LSTM

    fc\_size **=** trial**.**suggest\_int**(**"fc\_size"**,** 16**,** 64**,** step**=**8**)**  # Размер полносвязного слоя

    dropout\_rate **=** trial**.**suggest\_float**(**"dropout\_rate"**,** 0.1**,** 0.5**,** step**=**0.1**)**  # Вероятность Dropout

    learning\_rate **=** trial**.**suggest\_loguniform**(**"learning\_rate"**,** 1e-4**,** 1e-3**)**  # Скорость обучения

    # Создание модели с текущими гиперпараметрами

    model **=** LSTMModel**(**

        input\_size**=**25**,**  # Количество признаков во входных данных

        hidden\_size1**=**hidden\_size1**,**

        hidden\_size2**=**hidden\_size2**,**

        fc\_size**=**fc\_size**,**

        output\_size**=**1**,**  # Прогнозируемое значение (целевая переменная)

        dropout\_rate**=**dropout\_rate

**).**to**(**device**)**

    # Оптимизатор и функция потерь

    optimizer **=** torch**.**optim**.**Adam**(**model**.**parameters**(),** lr**=**learning\_rate**)**  # Adam оптимизатор

    loss\_fn **=** nn**.**MSELoss**()**  # Среднеквадратичная ошибка

    num\_epochs **=** 30  # Количество эпох для обучения

    train\_r2 **=** R2Score**().**to**(**device**)**  # Метрика для оценки R^2

    model**.**train**()**

    # Обучение модели

**for** epoch **in** **range(**num\_epochs**):**

**for** X**,** y **in** train\_loader**:**

            X**,** y **=** X**.**to**(**device**),** y**.**to**(**device**).**unsqueeze**(-**1**)**  # Перемещение данных на устройство (GPU/CPU)

            optimizer**.**zero\_grad**()**  # Сброс градиентов

            y\_pred **=** model**(**X**)**  # Предсказания модели

            loss **=** loss\_fn**(**y\_pred**,** y**)**  # Вычисление функции потерь

            loss**.**backward**()**  # Обратное распространение ошибки

            optimizer**.**step**()**  # Обновление весов модели

    # Оценка модели на тестовой выборке

    model**.eval()**

    val\_r2 **=** R2Score**().**to**(**device**)**

**with** torch**.**no\_grad**():**  # Отключение вычисления градиентов

**for** X**,** y **in** val\_loader**:**

            X**,** y **=** X**.**to**(**device**),** y**.**to**(**device**).**unsqueeze**(-**1**)**

            y\_pred **=** model**(**X**)**

            val\_r2**.**update**(**y\_pred**,** y**)**

    # Возврат значения метрики R^2

**return** val\_r2**.**compute**().**item**()**

# Устройство для вычислений (GPU, если доступен)

device **=** torch**.**device**(**"cuda" **if** torch**.**cuda**.**is\_available**()** **else** "cpu"**)**

# Даталоадеры для тренировочных и валидационных данных

train\_loader **=** DataLoader**(**train\_dataset**,** batch\_size**=**32**,** shuffle**=True)**  # Обучающая выборка

val\_loader **=** DataLoader**(**val\_dataset**,** batch\_size**=**32**,** shuffle**=False)**  # Валидационная выборка

# Настройка гиперпараметров с использованием Optuna

study **=** optuna**.**create\_study**(**direction**=**"maximize"**)**  # Максимизация метрики R^2

study**.**optimize**(**objective**,** n\_trials**=**50**)**  # Проведение 50 итераций поиска гиперпараметров

# Вывод лучших параметров

**print(**study**.**best\_params**)**

# Функция для обучения модели с заданными параметрами

**def** train\_model**(**model\_nn**,** train\_loader**,** test\_loader**,** optimizer**,** loss\_fn**,** num\_epochs**,** device**):**

    # Инициализация метрик R² для тренировочной и тестовой выборки

    train\_r2 **=** R2Score**().**to**(**device**)**

    test\_r2 **=** R2Score**().**to**(**device**)**

    # Основной цикл по эпохам

**for** epoch **in** **range(**num\_epochs**):**

        start **=** time**.**time**()**

        train\_loss **=** 0.0  # Суммарная ошибка на обучающей выборке

        train\_iters **=** 0  # Счётчик итераций на обучающей выборке

        model\_nn**.**train**()**  # Переключение модели в режим обучения

        # Цикл обучения на батчах

**for** X**,** y **in** train\_loader**:**

            X**,** y **=** X**.**to**(**device**),** y**.**to**(**device**)**  # Перемещение данных на устройство (GPU/CPU)

            y **=** y**.**unsqueeze**(-**1**)**  # Изменение формы целевой переменной (добавление измерения)

            y\_pred **=** model\_nn**(**X**)**  # Прогноз модели

            optimizer**.**zero\_grad**()**  # Сброс градиентов

            loss **=** loss\_fn**(**y\_pred**,** y**)**  # Вычисление функции потерь

            loss**.**backward**()**  # Обратное распространение ошибки

            optimizer**.**step**()**  # Обновление параметров модели

            train\_loss **+=** loss**.**item**()**  # Суммирование значения функции потерь

            train\_iters **+=** 1  # Увеличение счётчика итераций

            train\_r2**.**update**(**y\_pred**,** y**)**  # Обновление метрики R² на обучающей выборке

        # Расчёт метрики R² для тренировочной выборки

        train\_r2\_out **=** train\_r2**.**compute**()**

        train\_r2**.**reset**()**  # Сброс метрики для следующей эпохи

        # Оценка модели на тестовой выборке

        test\_loss **=** 0.0  # Суммарная ошибка на тестовой выборке

        test\_iters **=** 0  # Счётчик итераций на тестовой выборке

        model\_nn**.eval()**  # Переключение модели в режим оценки

**with** torch**.**no\_grad**():**  # Отключение вычисления градиентов

**for** X**,** y **in** test\_loader**:**

                X**,** y **=** X**.**to**(**device**),** y**.**to**(**device**)**  # Перемещение данных на устройство (GPU/CPU)

                y **=** y**.**unsqueeze**(-**1**)**  # Изменение формы целевой переменной

                y\_pred **=** model\_nn**(**X**)**  # Прогноз модели

                loss **=** loss\_fn**(**y\_pred**,** y**)**  # Вычисление функции потерь

                test\_loss **+=** loss**.**item**()**  # Суммирование значения функции потерь

                test\_iters **+=** 1  # Увеличение счётчика итераций

                test\_r2**.**update**(**y\_pred**,** y**)**  # Обновление метрики R² на тестовой выборке

        # Расчёт метрики R² для тестовой выборки

        test\_r2\_out **=** test\_r2**.**compute**()**

        test\_r2**.**reset**()**  # Сброс метрики для следующей эпохи

        # Вывод результатов для текущей эпохи

**if** epoch **<** 9**:**  # Для отображения нуля перед однозначным номером эпохи

**print(**f"Эпоха: 0{epoch**+**1} | Время: {time**.**time**()-**start:.2f} сек | "

                  f"Train Loss: {train\_loss**/**train\_iters:.4f} | Train R²: {train\_r2\_out:.2f} | "

                  f"Test Loss: {test\_loss**/**test\_iters:.4f} | Test R²: {test\_r2\_out:.2f}"**)**

**else:**

**print(**f"Эпоха: {epoch**+**1} | Время: {time**.**time**()-**start:.2f} сек | "

                  f"Train Loss: {train\_loss**/**train\_iters:.4f} | Train R²: {train\_r2\_out:.2f} | "

                  f"Test Loss: {test\_loss**/**test\_iters:.4f} | Test R²: {test\_r2\_out:.2f}"**)**

# Вызов функции обучения

train\_model**(**model**,** train\_loader**,** test\_loader**,** optimizer**,** loss\_fn**,** num\_epochs**,** device**)**

# Сохраняем обученную модель

model\_param **=** **{**

    "model\_state\_dict"**:** model**.**state\_dict**(),**

    "input\_size"**:** 25**,**

    "hidden\_size1"**:** 48**,**

    "hidden\_size2"**:** 24**,**

    "fc\_size"**:** 16**,**

    "output\_size"**:** 1**,**

    "dropout\_rate"**:** 0.4

**}**

torch**.**save**(**model\_param**,** "models/lstm\_model\_param\_88.pth"**)**

## Прогноз и визуализация

# Предсказания и реальные значения

predictions **=** **[]**

actuals **=** **[]**

# Отключение вычисления градиентов для ускорения

**with** torch**.**no\_grad**():**

**for** X**,** y **in** test\_loader**:**

        X**,** y **=** X**.**to**(**device**),** y**.**to**(**device**)**

        # Предсказания модели

        y\_pred **=** model**(**X**)**

        predictions**.**extend**(**y\_pred**.**squeeze**().**cpu**().**numpy**())**  # Преобразование в список numpy

        # Истинные значения

        actuals**.**extend**(**y**.**cpu**().**numpy**())**

# Обратное преобразование нормализации

predictions\_real **=** scaler\_y**.**inverse\_transform**(**np**.**array**(**predictions**).**reshape**(-**1**,** 1**))**

actuals\_real **=** scaler\_y**.**inverse\_transform**(**np**.**array**(**actuals**).**reshape**(-**1**,** 1**))**

# Создание временной шкалы для визуализации

start\_date **=** "2023-12-30 07:00:00"  # Начальная дата

freq **=** "1H"  # Частота данных

num\_points **=** **len(**predictions\_real**)**  # Количество точек

data\_timestamps **=** pd**.**date\_range**(**start**=**start\_date**,** periods**=**num\_points**,** freq**=**freq**)**

# Визуализация предсказаний и реальных значений

plt**.**figure**(**figsize**=(**12**,** 7**))**

plt**.**plot**(**data\_timestamps**,** actuals\_real**,** label**=**"Истинные значения (цены закрытия)"**,** color**=**"blue"**,** alpha**=**0.7**)**

plt**.**plot**(**data\_timestamps**,** predictions\_real**,** label**=**"Предсказания"**,** color**=**"red"**,** alpha**=**0.7**)**

plt**.**title**(**"Истинные цены закрытия и предсказания ETH/USDT"**,** fontweight**=**'bold'**,** fontsize**=**14**)**

plt**.**xlabel**(**"Дата"**,** fontweight**=**'bold'**,** fontsize**=**12**)**

plt**.**ylabel**(**"Цена закрытия (USDT)"**,** fontweight**=**'bold'**,** fontsize**=**12**)**

plt**.**xticks**(**rotation**=**45**)**  # Поворот меток оси X

plt**.**grid**(True)**

# Добавление легенды

plt**.**legend**(**loc**=**"upper right"**)**

# Улучшение компоновки

plt**.**tight\_layout**()**

# Отображение графика

plt**.**show**()**

Приложение 3

Расчет основных технических индикаторов

* 1. Индекс относительной силы (RSI)

RSI измеряет силу тренда, сравнивая недавний рост и падение цен:

* 1. MACD (Moving Average Convergence Divergence)

MACD — это разница между двумя экспоненциальными скользящими средними (EMA):

* 1. Сигнальная линия MACD

Сигнальная линия — это EMA MACD за заданный период (обычно 9):

* 1. Гистограмма MACD

Гистограмма MACD показывает разницу между MACD и сигнальной линией:

* 1. Стохастический осциллятор %K (slowK)

%K рассчитывается следующим образом:

* 1. Стохастический осциллятор %D (slowD)

%D — это сглаженная версия %K:

* 1. Средний истинный диапазон (ATR)

ATR измеряет волатильность и основан на истинном диапазоне (TR):