УДК 004.896

Интеллектуальный анализ динамики цен на криптовалюты на криптовалютных биржах с использованием методов машинного обучения

Малик Александр Игоревич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
Статья исследует динамику цен на криптовалюты с использованием методов машинного обучения для прогнозирования цен. Собраны данные, разработаны и обучены модели (Ridge, DecisionTreeRegressor, RandomForestRegressor, GradientBoostingRegressor, CatBoostRegressor, XGBRegressor, LGBMRegressor). Эффективность оценена по метрикам MRE, MSE и R^2, выбрана лучшая модель. Проведен сравнительный анализ с существующими методами, даны рекомендации.
Ключевые слова: криптовалюты, машинное обучение, прогнозирование цен, временные ряды, Ridge регрессия, DecisionTreeRegressor, RandomForestRegressor, GradientBoostingRegressor, CatBoostRegressor, XGBRegressor, LGBMRegressor, анализ данных, метрики оценки моделей.

Intelligent analysis of cryptocurrency price dynamics on cryptocurrency exchanges using machine learning methods

Malik Aleksandr Igorevich
Sholom-Aleichem Priamursky State University
Student

Abstract
The article examines the dynamics of cryptocurrency prices using machine learning methods for price prediction. Data were collected, and models (Ridge, DecisionTreeRegressor, RandomForestRegressor, GradientBoostingRegressor, CatBoostRegressor, XGBRegressor, LGBMRegressor) were developed and trained. Effectiveness was evaluated using MRE, MSE, and R^2 metrics, and the best model was selected. A comparative analysis with existing methods was conducted, and recommendations were provided.
Keywords: cryptocurrencies, machine learning, price prediction, time series, Ridge regression, DecisionTreeRegressor, RandomForestRegressor, GradientBoostingRegressor, CatBoostRegressor, XGBRegressor, LGBMRegressor, data analysis, model evaluation metrics.

1 Введение

1.1 Актуальность

В современном мире криптовалюты стали объектом все более широкого интереса как среди инвесторов, так и среди исследователей финансовых рынков. Динамика и колебания цен на криптовалюты представляют собой сложную и непредсказуемую систему, которая требует разработки эффективных методов анализа для принятия обоснованных инвестиционных решений.

В условиях быстрого развития рынка криптовалют актуальность исследований по анализу и прогнозированию цен на криптовалюты становится все более высокой. Методы машинного обучения позволяют эффективно обрабатывать и анализировать большие объемы данных, что делает их особенно полезными для решения задач прогнозирования на финансовых рынках.

1.2 Обзор исследований

На сегодняшний день накоплен значительный объем работ по анализу цен на криптовалюты с применением методов машинного обучения, охватывающих различные аспекты обработки данных, предсказания стоимости и факторов, влияющих на изменение курсов.

Ф. Фанг, К. Вентре и М. Басиоз рассмотрели возрастающий интерес финансовых учреждений к криптовалютам как к цифровым активам, анализируя 146 статей, в которых исследуются торговля, волатильность, управление портфелем и риски. Авторы выделяют важные тренды и возможности, связанные с платформами, стратегиями и данными в данной области [1]. М. Орту, Н. Урас, К. Конверсано, С. Бартолуччи и Д. Дестефанис проанализировали прогнозирование цен на криптовалюты, включая Ethereum и Bitcoin, за период с 2017 по 2020 год, применяя четыре глубокие нейронные сети и три класса признаков: технические, торговые и социальные. Они отметили значительное повышение точности прогнозов при учете торговых и социальных индикаторов [2]. И. Хёнсонг из Кангвонского национального университета исследовал методы глубокого обучения для прогнозирования цен Bitcoin, применяя DNN, LSTM, CNN и глубокие остаточные сети. Результаты показали, что LSTM немного превосходит другие модели по точности прогнозирования цен, а DNN лучше справляется с предсказанием изменений цен. Классификационные модели продемонстрировали большую эффективность в алгоритмической торговле [3]. М. А. Ямин и М. Чаудри изучили прогнозирование рынка криптовалют с помощью машинного обучения [4]. Э. Акылдырым и его коллеги исследовали предсказуемость цен на двенадцать криптовалют на дневных и минутных интервалах, применяя алгоритмы, такие как метод опорных векторов, логистическая регрессия, искусственные нейронные сети и случайные леса. Все алгоритмы продемонстрировали среднюю точность прогнозов выше 50%, при этом метод опорных векторов показал наилучшие результаты [5]. Н. Манглы из Института технологии Atria проанализировал

цену Bitcoin, учитывая различные влияющие факторы, с целью максимально точного предсказания ежедневных изменений. Рыночная капитализация криптовалют превышает $230 млрд, а Bitcoin, как наиболее ценная криптовалюта, привлекает внимание как цифровое хранилище стоимости [6]. С. Макнелли, Д. Роше и С. Д. Кэтон из Университетского колледжа Дублина предсказывали направление цены Bitcoin в долларах, используя данные Bitcoin Price Index и глубокие нейронные сети LSTM. Модель LSTM показала 52% точности и среднеквадратичную ошибку 8%, превосходя модель ARIMA. Обучение на GPU было более эффективным на 67.7% [7]. Т. Чэн и К. Гуэстрин представили систему XGBoost - масштабируемый инструмент для усиления деревьев, использующий новые алгоритмы для работы с разреженными данными и методы приближенного построения деревьев. XGBoost позволяет обрабатывать миллиарды примеров, затрачивая меньше ресурсов по сравнению с существующими системами [8]. И. Ливиерис, Э. Пинтелас и С. Ставрояннис исследовали прогнозирование цен крупных криптовалют с применением ансамблевых методов глубокого обучения, включая усреднение, бэггинг и стекинг. Современные модели, такие как LSTM и сверточные слои, показали, что ансамблевые подходы обеспечивают надежное прогнозирование как цен на следующий час, так и направления изменения цен [9]. А. Хассан, Н. Паила, В. Баммиди и А. В. Диввала проанализировали временные ряды для прогнозирования цен Bitcoin, включая предварительную обработку данных, оценку стационарности, создание моделей ARIMA и SARIMAX, и их оценку через диагностику и валидацию на отложенных данных, что помогает в предсказании будущих цен и понимании рыночной динамики [10].

1.3 Цель исследования

Цель исследования заключается в анализе динамики цен на криптовалюты на криптовалютных биржах с использованием методов машинного обучения. В рамках исследования предполагается подготовка и предварительная обработка исторических данных о ценах на криптовалюты, разработка и обучение моделей машинного обучения для прогнозирования цен, оценка их эффективности с помощью метрик точности, выбор наиболее успешной модели, а также проведение сравнительного анализа с существующими методами. На основе полученных данных будут сделаны выводы и даны рекомендации по применимости машинного обучения для анализа ценовых колебаний криптовалют.

2 Материалы и методы

Предварительный анализ данных: Визуализация основных статистических показателей и временного ряда цен биткойна позволяет получить представление о распределении данных и их динамике.

Корреляционный анализ: Вычисление и визуализация корреляционной матрицы позволяет оценить взаимосвязь между различными признаками в данных.

Обучение и оценка моделей: Использование различных моделей машинного обучения, таких как регрессоры Ridge, DecisionTreeRegressor, RandomForestRegressor, GradientBoostingRegressor, CatBoostRegressor, XGBRegressor и LGBMRegressor, для прогнозирования цен закрытия биткойна на основе других признаков. Для каждой модели оценивается средняя относительная ошибка (MRE), среднеквадратичная ошибка (MSE) и коэффициент детерминации (R^2).

Визуализация результатов: Представление результатов прогнозирования различных моделей на временном ряду цен биткойна позволяет сравнить их эффективность и оценить, как каждая модель соотносится с реальными данными.

3 Результаты и обсуждения

3.1 Получение данных

Задача будет выполнена в облачной среде разработки Google Colab. Однако из-за ограниченного доступа Colab к API Binance, для получения датасета мы воспользуемся одной из локальных IDE, а именно Visual Studio Code.

Процесс получения данных начнется с отправки HTTP-запроса к API Binance по адресу **(https://api.binance.com/api/v3/klines)**. Для отправки и обработки этих запросов будет использована встроенная библиотека requests. Данные, полученные в результате запросов, будут сохраняться и обрабатываться с помощью внешней библиотеки pandas (Рис. 1).

```
import requests
import pandas as pd
```

Рисунок 1. Подключение библиотек

Так как работа связана с криптовалютой Bitcoin и требует обращения к API Binance, нужно определить ключевые константы: дата начала торговли валютой на бирже и символ валютной пары BTCUSDT. Эти данные необходимы для настройки функции парсинга данных в будущем (Рис. 2).

```
# Определение символа и начальной даты
symbol = 'BTCUSDT'
start_date = '1502937600000'  # Unix-время для '2017-08-17'
```

Рисунок 2. Определение констант

Далее создаем функцию **fetch_binance_data**, которая принимает символ валютной пары, дату начала и лимит (по умолчанию 1000) и возвращает исторические данные OHLCV (открытие, максимум, минимум, закрытие, объем) с биржи Binance. Функция отправляет запрос к API Binance с указанными параметрами и возвращает данные в формате JSON. Если запрос завершился успешно (код статуса 200), функция возвращает данные, в противном случае выводит сообщение об ошибке и возвращает None (Рис. 3).

```python
# Функция для получения исторических данных OHLCV с биржи Binance
def fetch_binance_data(symbol, start_date, limit=1000):
    base_url = 'https://api.binance.com/api/v3/klines'
    params = {
        'symbol': symbol,
        'interval': '1d',  # Дневной интервал
        'startTime': start_date,
        'limit': limit  # Максимальное количество точек данных в одном запросе
    }
    response = requests.get(base_url, params=params)
    if response.status_code == 200:
        data = response.json()
        return data
    else:
        print("Не удалось получить данные с биржи Binance. Код статуса:", response.status_code)
        return None
```

Рисунок 3. Функция парсинга данных

Следующим шагом получаем первую порцию данных в переменную **ohlc_data**, создаем переменную для хранения всех полученных данных **all_data** и переменную с числом оставшихся записей **remaining_limit** (Рис. 4).

```python
# Получение первой порции данных
ohlc_data = fetch_binance_data(symbol, start_date)

# Первая порция данных
all_data = ohlc_data

# Число записей, которые еще нужно получить
remaining_limit = 3000 - len(ohlc_data)
```

Рисунок 4. Переменные

Получение оставшихся данных, если требуется. Цикл продолжается, пока остается необходимое количество данных для получения (remaining_limit > 0). Для получения оставшихся данных отправляются дополнительные запросы к API Binance, начиная с времени последней полученной записи. Данные добавляются к списку **all_data**, а количество оставшихся записей уменьшается (Рис. 5).

```python
# Получение остальных данных, если требуется
while remaining_limit > 0:
    last_timestamp = all_data[-1][0]  # Время последней записи
    next_data = fetch_binance_data(symbol, last_timestamp + 1, min(remaining_limit, 1000))
    if next_data:
        all_data.extend(next_data)
        remaining_limit -= len(next_data)
    else:
        break
```

Рисунок 5. Цикл получения оставшихся данных

После получения данных – необходимо преобразовать их в объект класса ранее импортированной библиотеки **pandas**, где каждому столбцу в DataFrame соответствует один из параметров данных OHLCV (Рис. 6).

```
# Конвертация данных в DataFrame
columns = ['timestamp', 'open', 'high',
           'low', 'close', 'volume',
           'close_time', 'quote_asset_volume', 'number_of_trades',
           'taker_buy_base_asset_volume', 'taker_buy_quote_asset_volume', 'ignore']
df = pd.DataFrame(all_data, columns=columns)
```

Рисунок 6. Конвертация данных в датафрейм

Преобразуем временные метки в формат DateTime для удобства работы с данными (Рис. 7).

```
# Конвертация временной метки в формат datetime
df['timestamp'] = pd.to_datetime(df['timestamp'], unit='ms')
```

Рисунок 7. Преобразование времени формата Unix в формат DateTime

Датасет готов. Осталось только сохранить его в корневом каталоге для продолжения работы с ним уже в среде **Google Colab**. Для этого, используя метод класса **to_csv** библиотеки **pandas** сохраним датасет в виде CSV-файла (Рис. 8).

```
# Создание csv файла с датасетом
df.to_csv('BTC_Binance.csv', index=False)
```

Рисунок 8. Преобразование датафрейма в CSV-файл

3.2 Анализ данных

Приступая к анализу данных, первым делом - загрузим датасет из CSV-файла в датафрейм и откроем его. Для этого необходимо повторно загрузить библиотеку, так как, ранее загруженная библиотека была загружена на IDE локального устройства, а не текущей IDE, а затем выполнить считывание, используя метод **read_csv** (Рис. 9).

```
import pandas as pd

# Загрузка данных
data = pd.read_csv("BTC_Binance.csv")

# Осмотр первых нескольких строк данных
print(data.head())
```

Рисунок 9. Вывод датасета из файла формата CSV

После того, как была выполнена функция **print()**, можно заметить первые 5 строк датасета, а также заголовки столбцов. По которым можно

сделать некоторые выводы и конкретнее понять, что из себя представляет каждый столбец датасета (Рис. 10).

	timestamp	open	high	low	close	volume	close_time	\
0	2017-08-17	4261.48	4485.39	4200.74	4285.08	795.150377	1503014399999	
1	2017-08-18	4285.08	4371.52	3938.77	4108.37	1199.888264	1503100799999	
2	2017-08-19	4108.37	4184.69	3850.00	4139.98	381.309763	1503187199999	
3	2017-08-20	4120.98	4211.08	4032.62	4086.29	467.083022	1503273599999	
4	2017-08-21	4069.13	4119.62	3911.79	4016.00	691.743060	1503359999999	

	quote_asset_volume	number_of_trades	taker_buy_base_asset_volume	\
0	3.454770e+06	3427	616.248541	
1	5.086958e+06	5233	972.868710	
2	1.549484e+06	2153	274.336042	
3	1.930364e+06	2321	376.795947	
4	2.797232e+06	3972	557.356107	

	taker_buy_quote_asset_volume	ignore
0	2.678216e+06	0
1	4.129123e+06	0
2	1.118002e+06	0
3	1.557401e+06	0
4	2.255663e+06	0

Рисунок 10. Первые 5 строк датасета

Исходя из полученной информации, можно сделать вывод о том, что датасет представлен в виде временных рядов данных о курсе биткоина на бирже Binance с 17 августа 2017 года и содержит в себе порядка 11 столбцов:

timestamp: Временная метка, обозначающая момент измерения данных.

open: Цена открытия для данного временного периода.

high: Наивысшая цена за данный временной период.

low: Самая низкая цена за данный временной период.

close: Цена закрытия для данного временного периода.

close_time: Время закрытия данного временного периода.

quote_asset_volume: Объем торгов в котируемой валюте за данный временной период.

number_of_trades: Общее количество сделок за данный временной период.

taker_buy_base_asset_volume: Объем покупок (как тейкер) базового актива за данный временной период.

taker_buy_quote_asset_volume: Объем покупок (как тейкер) котируемого актива за данный временной период.

ignore: Игнорируемый столбец, возможно, не несущий смысловой нагрузки для анализа.

Этот датасет может использоваться для анализа изменения цен биткойна со временем, оценки, объемов торгов и других параметров, связанных с криптовалютной биржей.

Следом выведем основные статистические показатели для каждого столбца данных при помощи метода **describe()** (Рис. 11).

```
# Основные статистические показатели
print(data.describe())
```

Рисунок 11. Функция вывода основных статистических данных

Результат вывода статистических данных представлен следующим образом на рисунке 12.

	open	high	low	close	volume	\
count	2439.000000	2439.000000	2439.000000	2439.000000	2439.000000	
mean	22475.090004	23052.324309	21842.224600	22499.526523	72063.563995	
std	17327.409867	17771.287801	16831.919518	17343.631529	83015.908050	
min	3188.010000	3276.500000	2817.000000	3189.020000	228.108068	
25%	8121.605000	8296.500000	7850.030000	8126.915000	29929.658650	
50%	16850.360000	17143.130000	16579.850000	16885.200000	46114.359022	
75%	34709.635000	35819.000000	33582.380000	34829.185000	76342.507571	
max	73072.400000	73777.000000	71333.310000	73072.410000	760705.362783	

	close_time	quote_asset_volume	number_of_trades	\
count	2.439000e+03	2.439000e+03	2.439000e+03	
mean	1.608336e+12	1.673738e+09	1.462081e+06	
std	6.084492e+10	2.044956e+09	1.960466e+06	
min	1.503014e+12	9.778657e+05	2.153000e+03	
25%	1.555675e+12	2.929213e+08	3.108160e+05	
50%	1.608336e+12	8.290118e+08	8.243730e+05	
75%	1.660997e+12	2.417086e+09	1.641540e+06	
max	1.713658e+12	1.746531e+10	1.522359e+07	

	taker_buy_base_asset_volume	taker_buy_quote_asset_volume	ignore
count	2439.000000	2.439000e+03	2439.0
mean	35845.225324	8.297412e+08	0.0
std	41301.373948	1.016576e+09	0.0
min	56.190141	2.413638e+05	0.0
25%	15109.552510	1.511546e+08	0.0
50%	22842.831276	4.051931e+08	0.0
75%	37772.274226	1.200405e+09	0.0
max	374775.574085	8.783916e+09	0.0

Рисунок 12. Статистические данные

Из представленных статистических показателей можно сделать несколько выводов:

- Средние значения: Среднее значение открытия, максимальной и минимальной цены, а также закрытия находятся примерно в одной области, что говорит о том, что средний диапазон изменения цены биткойна за период составляет примерно 22 500 - 23 000 USD.
- Разброс: Стандартное отклонение показывает, что цены могут значительно изменяться вокруг среднего значения. Это подтверждается и разницей между квартилями: от первого до третьего квартиля значительный разброс.
- Минимальные и максимальные значения: Минимальные и максимальные значения цен позволяют оценить абсолютные минимумы и максимумы цены биткойна за рассматриваемый период.

- Торговый объем: Средний объем торгов составляет около 72 000 BTC, при этом есть значительные колебания в объеме торгов, что может указывать на различные режимы торгов и активность рынка.
- Количество сделок: Среднее количество сделок за период составляет около 1.5 миллиона, что также может указывать на высокую активность рынка биткойна на бирже Binance.
- Игнорируемый столбец: Столбец "ignore" имеет одно и то же значение для всех записей (0), что может указывать на то, что этот столбец несущественен для анализа данных и может быть исключен из рассмотрения.

Теперь посмотрим на график изменения цены в зависимости от времени, для этого воспользуемся библиотекой для визуализации данных **matplotlib**.

Вначале преобразуем столбец timestamp в удобный формат времени и установим его в качестве индекса (Рис. 13).

```
import matplotlib.pyplot as plt

# Преобразование столбца 'timestamp' в тип datetime
data['timestamp'] = pd.to_datetime(data['timestamp'])

# Установка столбца 'timestamp' в качестве индекса
data.set_index('timestamp', inplace=True)
```

Рисунок 13. Подключение библиотеки. Преобразование времени и установка индекса для графика

Далее строим график изменения цены закрытия биткойна со временем, используя столбец **close** в качестве значений по оси Y и **временные метки** в качестве значений по оси X (Рис. 14-15).

```
# Визуализация временного ряда цен биткойна
plt.figure(figsize=(12, 6))
plt.plot(data['close'], label='Цена закрытия')
plt.title('Изменение цены биткойна со временем')
plt.xlabel('Дата')
plt.ylabel('Цена в USD')
plt.legend()
plt.show()
```

Рисунок 14. Определение вида и вывод графика

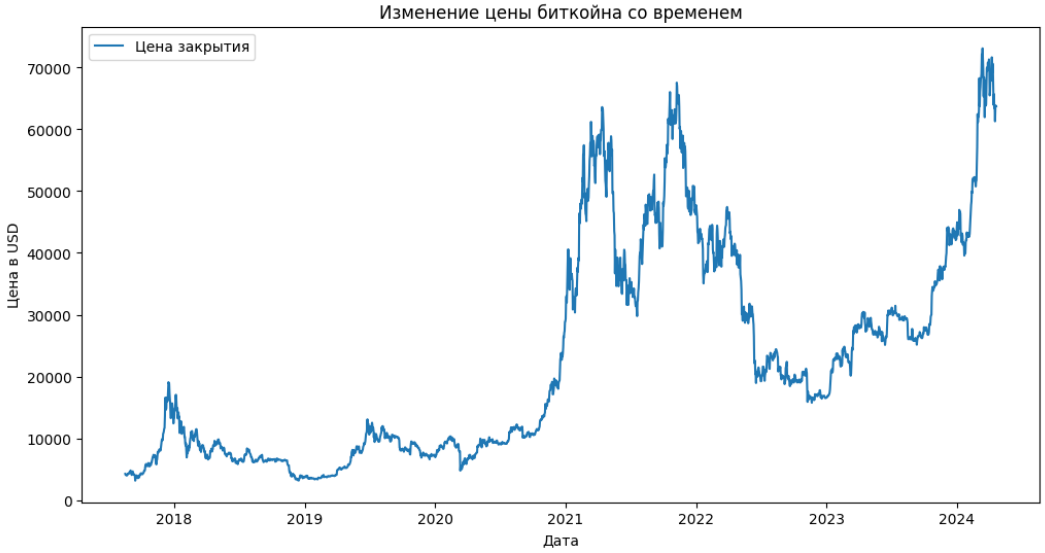

Рисунок 15. График стоимости биткоина

На графике можно видеть резкие скачки цен в начале и конце 2021 года, а также резкий подъем до рекордных цифр в начале 2024 года, что соответствует периодам значимых моментов на рынке криптовалют, таким как всплеск интереса со стороны институционных инвесторов, увеличение принятия криптовалюты крупными компаниями и рост участия розничных инвесторов. Эти факторы способствовали значительной волатильности цен, что приводило к резкому росту цен, за которым следовали коррекции.

В целом, эти движения цен отражают динамичную природу рынка криптовалют, характеризующуюся периодами быстрого роста, за которыми следует консолидация или коррекция.

Продолжим исследование с вычисления корреляционной матрицы для всех числовых столбцов данных и визуализируем её в виде тепловой карты с помощью библиотеки **Seaborn** (Рис. 16-17).

```
import seaborn as sns
# Вычисление корреляционной матрицы
correlation_matrix =  data.corr()

# Визуализация корреляционной матрицы
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", annot_kws={"size": 10})
plt.title('Корреляционная матрица')
plt.show()
```

Рисунок 16. Вычисление матрицы и создание ее визуализации

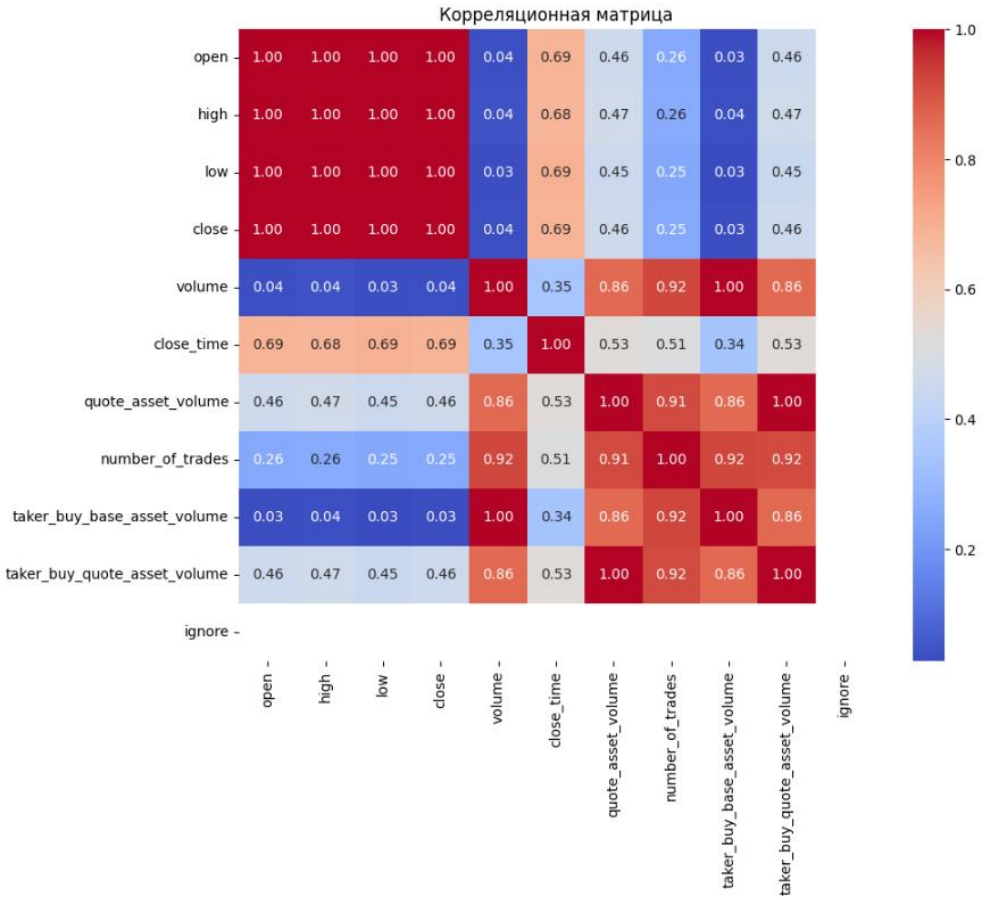

Рисунок 17. Корреляционная матрица

Из представленной корреляционной матрицы видно, что:

- Цены открытия (**open**), закрытия (**close**), наивысшей (**high**) и самой низкой (**low**) за день тесно связаны между собой, что логично, так как они являются разными характеристиками одного и того же финансового инструмента.
- Объем торгов (**volume**) имеет высокую корреляцию с объемом покупок (**taker_buy_base_asset_volume**) и объемом продаж (**quote_asset_volume**), что также ожидаемо, так как они отражают торговые активности.
- Время закрытия (**close_time**) имеет некоторую корреляцию с ценами и объемом торгов, что может указывать на временные зависимости в данных.
- Некоторые пары признаков имеют более высокую корреляцию, например, **quote_asset_volume** и **volume**, что означает, что они сильно взаимосвязаны.

3.3 Обучение моделей

Исходя из всего ранее перечисленного, приступим к обучению моделей. Для начала разделим данные на признаки X и целевую переменную y. Отсечем лишние. Затем делим на обучающий и тестовый наборы в соотношении 8:2 с помощью функции **train_test_split** (Рис. 18).

```
from sklearn.model_selection import train_test_split

# Разделение данных на признаки (X) и целевую переменную (y)
X = data.drop(columns=['close', 'open', 'high', 'low', 'close_time', 'taker_buy_quote_asset_volume', 'taker_buy_base_asset_volume'])
y = data['close']  # Целевая переменная - столбец 'close'

# Разделение данных на обучающую и тестовую выборки
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Рисунок 18. Определение признаков и целевой переменной с последующим разделением на выборки

Затем определяемся с моделями, которые будут использоваться для предсказания цены закрытия биткойна. В исследовании были выбраны 7 моделей: Ridge, Decision Tree Regression, Gradient Boosting Regression, Cat Boost Regression, XGB Regression, LGBM Regression. Далее необходимо подключить модели и зависимости. Все модели, кроме Cat Boost уже встроены в среду и не нуждаются в инсталляции, в связи с чем их можно просто импортировать, а для установки Cat Boost нужно дописать в начале: **!pip install catboost** (Рис. 19).

```
# установка модулей моделей и метрик
!pip install catboost
from sklearn.tree import DecisionTreeRegressor
from xgboost import XGBRegressor
from lightgbm import LGBMRegressor
from catboost import CatBoostRegressor
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.linear_model import Ridge
import numpy as np
```

Рисунок 19. Установка и подключение библиотек с необходимыми моделями

Создадим массив моделей и словарь для хранения метрик каждой модели. (Рис. 20).

```
# Создание списка моделей
models = [
    Ridge(),
    DecisionTreeRegressor(),
    RandomForestRegressor(),
    GradientBoostingRegressor(),
    CatBoostRegressor(verbose=0),
    XGBRegressor(),
    LGBMRegressor()
]

# Словарь для хранения результатов каждой модели
results = {'Model': [], 'MRE': [], 'MSE': [], 'R^2': []}
```

Рисунок 20. Переменные, для хранения моделей и значений метрик моделей

Далее пройдемся по созданному массиву моделей циклом, чтобы обучить каждую и сохранить ее результаты метрик в собственную ячейку словаря **results**. (Рис. 21). В процессе предсказания моделей используются следующие метрики:

1. **Средняя относительная ошибка (MRE)** - среднее значение относительной ошибки предсказаний модели. Она вычисляется как среднее абсолютное значение отношения абсолютной разницы между фактическим и предсказанным значением к фактическому значению.
2. **Среднеквадратичная ошибка (MSE)** - среднее значение квадратов разностей между фактическими и предсказанными значениями. Она измеряет среднеквадратичное отклонение предсказанных значений от фактических значений.
3. **Коэффициент детерминации (R^2)** - мера соответствия модели данным. Он оценивает объясненную дисперсию в зависимой переменной от всех предикторов. Значение R^2 близкое к 1 указывает на то, что модель хорошо объясняет изменения в данных.

Эти метрики позволяют оценить качество работы моделей и их способность делать точные прогнозы на основе имеющихся данных.

```
# Обучение и оценка каждой модели
for model in models:
    model.fit(X_train, y_train)  # Обучение модели
    y_pred = model.predict(X_test)  # Предсказание на тестовой выборке

    # Оценка модели
    mre = np.mean(np.abs((y_test - y_pred) / y_test)) * 100  # Средняя относительная ошибка
    mse = mean_squared_error(y_test, y_pred)  # Среднеквадратичная ошибка
    r2 = r2_score(y_test, y_pred)  # Коэффициент детерминации

    # Сохранение результатов в словарь
    results['Model'].append(model.__class__.__name__)
    results['MRE'].append(mre)
    results['MSE'].append(mse)
    results['R^2'].append(r2)
```

Рисунок 21. Цикл для обучения всех моделей массива models

3.4 Вывод результатов и оптимизация

Следующим этапом будет вывод полученных измерений. Для этого преобразуем словарь **results** в датафрейм и выведем его (Рис. 22-23).

```
# Создание DataFrame для удобного отображения результатов
results_df = pd.DataFrame(results)
print(results_df)
```

Рисунок 22. Преобразование словаря в датафрейм и его вывод

	Model	MRE	MSE	R^2
0	Ridge	64.300850	8.077804e+07	0.719072
1	DecisionTreeRegressor	6.700575	5.117526e+06	0.982202
2	RandomForestRegressor	4.502258	2.811023e+06	0.990224
3	GradientBoostingRegressor	9.297889	4.513837e+06	0.984302
4	CatBoostRegressor	4.328914	1.481362e+06	0.994848
5	XGBRegressor	4.840796	2.455467e+06	0.991460
6	LGBMRegressor	4.800929	2.237730e+06	0.992218

Рисунок 23. Вывод результатов обучения и работы моделей

1. Ridge: Данная модель имеет среднюю относительную ошибку (MRE) в размере 64.3%, что довольно высоко. Среднеквадратичная ошибка (MSE) также высока, составляя около 8.08 * 10^7. Коэффициент детерминации (R^2) равен 0.719, что означает, что только примерно 71.9% дисперсии зависимой переменной объясняется моделью.
2. DecisionTreeRegressor: эта модель имеет более низкую среднюю относительную ошибку (MRE) в размере 6.7% и более низкую среднеквадратичную ошибку (MSE) около 5.12 * 10^6. Коэффициент детерминации (R^2) равен 0.982, что указывает на более высокую объясненную дисперсию.
3. RandomForestRegressor: эта модель также показывает хорошие результаты с низкой средней относительной ошибкой (MRE) и среднеквадратичной ошибкой (MSE). Коэффициент детерминации (R^2) близок к 1, что свидетельствует о том, что модель хорошо соответствует данным.
4. GradientBoostingRegressor, CatBoostRegressor, XGBRegressor, LGBMRegressor: Эти модели также показывают хорошие результаты с низкими значениями MRE, MSE и высокими значениями R^2, что говорит о их способности делать точные прогнозы на основе предоставленных данных.

Общий вывод: модели RandomForestRegressor, CatBoostRegressor, XGBRegressor и LGBMRegressor имеют лучшие показатели по сравнению с остальными моделями, так как они имеют более низкие значения ошибок и более высокий коэффициент детерминации.

Визуализируем предсказания моделей в виде диаграммы рассеяния. Для этого, используем ранее подключенные библиотеки matplotlib для построения графика и seaborn для определения палитры цветов моделей на графике, после чего определим параметры и вид диаграммы (Рис. 24-25).

```
import seaborn as sns
palette = sns.color_palette('tab10', len(models))

# Предсказания всех моделей
plt.figure(figsize=(12, 8))
plt.scatter(y_test.index, y_test, label='Фактические значения', color='blue')

for i, model in enumerate(models):
    y_pred = model.predict(X_test)
    plt.scatter(y_test.index, y_pred, label=model.__class__.__name__, color=palette[i])

plt.title('Предсказания разных моделей')
plt.xlabel('Дата')
plt.ylabel('Цена в USD')
plt.legend()
plt.show()
```

Рисунок 24. Код визуализации предсказаний моделей

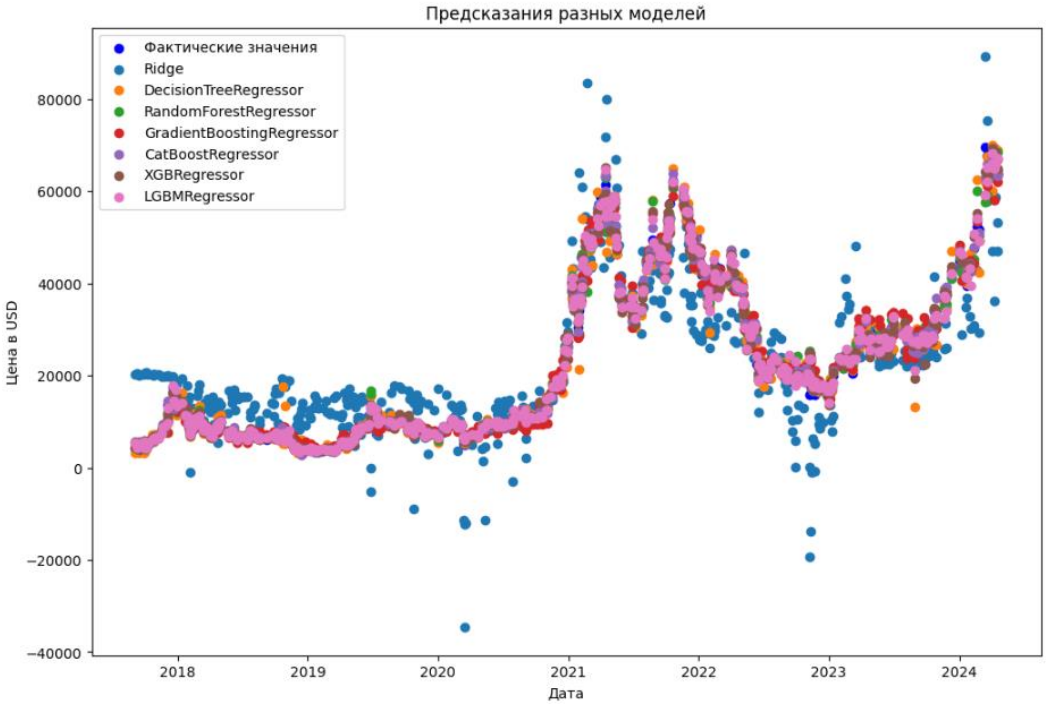

Рисунок 25. Диаграмма рассеивания предсказания моделей

Из графика видно, что все модели показывают примерно одинаковые результаты, но модель Ridge не относится к ним, имея видимую визуально большую дисперсию. Попробуем исправить данную проблему. Для этого создадим новые признаки на основе существующих данных, которые могли бы улучшить производительность моделей. Например, можно добавить скользящие средние, отклонения от тренда. Вслед за этим – удалим пустые значения. Сделать это можно сразу же, после кода со считыванием CSV-файла (Рис. 26).

```
# Загрузка данных
data = pd.read_csv("BTC_Binance.csv")

window_sizes = [7, 14, 30]  # Размеры окон для скользящих средних

for window in window_sizes:
    data[f'moving_average_{window}'] = data['close'].rolling(window=window).mean()

# Добавление стандартного отклонения
data['std_dev'] = data['close'].rolling(window=30).std()

data.dropna(inplace=True)
```

Рисунок 26. Добавление скользящих средних и стандартного отклонения цен закрытия

Далее смотрим на результаты, уже после внесенных изменений и делаем выводы (Рис. 27-28).

	Model	MRE	MSE	R^2
0	Ridge	3.413760	1.390561e+06	0.995223
1	DecisionTreeRegressor	3.677426	2.221947e+06	0.992366
2	RandomForestRegressor	3.012759	1.138678e+06	0.996088
3	GradientBoostingRegressor	3.701864	1.357308e+06	0.995337
4	CatBoostRegressor	3.539470	1.083040e+06	0.996279
5	XGBRegressor	3.047500	1.046672e+06	0.996404
6	LGBMRegressor	3.208853	1.061385e+06	0.996353

Рисунок 27. Вывод результатов обучения и работы моделей, после добавления скользящих средних и стандартного отклонения цен закрытия

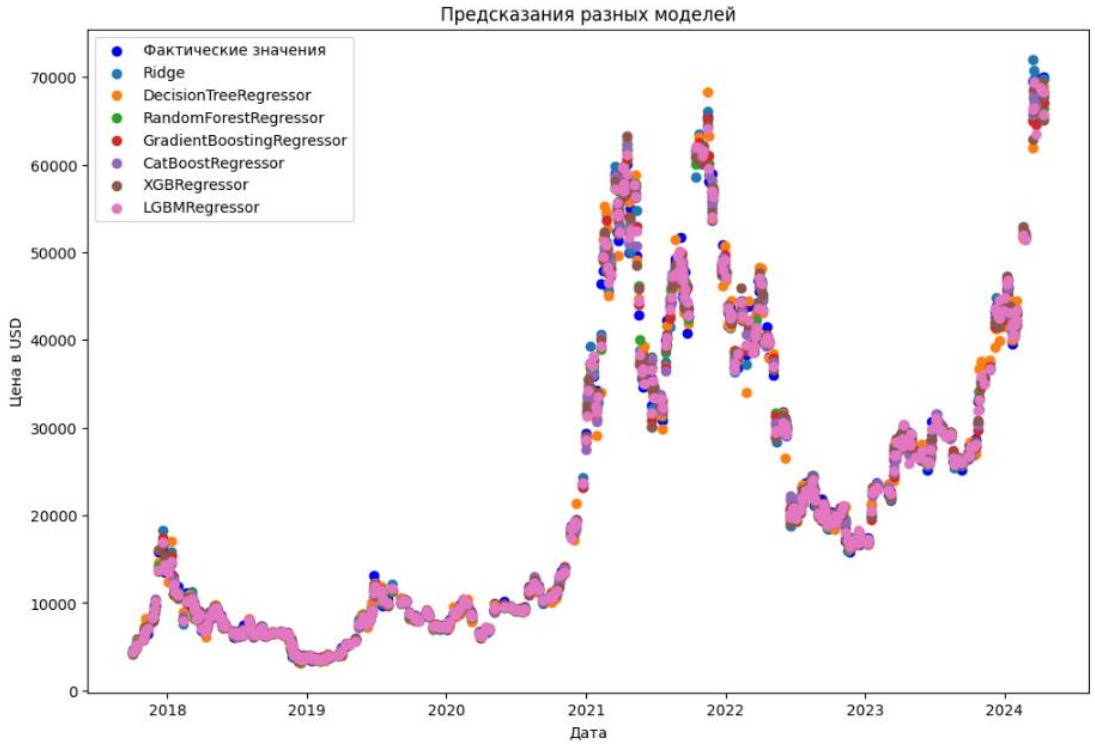

Рисунок 28. Диаграмма рассеивания предсказания моделей, после добавления скользящих средних и стандартного отклонения цен закрытия

Исходя из представленных данных после оптимизации, можно сделать вывод о том, что прогресс отличный и эти действия повлияли в лучшую сторону не только на одну модель, но и на показатели оценки других моделей, что свидетельствует о хорошей работе.

3.5 Заключение

Результат исследования представлен в таблице 1.

Таблица 1. Сравнение результатов предсказания моделей и метрик

Модель	MAE	MSE	R^2	Почему лучше	Почему хуже
Ridge	3.41	1.39e+06	0.995	Модель обладает достаточно высокой точностью и показывает сопоставимые результаты с другими	Возможно, модель не учитывает некоторые сложные нелинейные зависимости в данных, что могло бы улучшить ее результаты.
DecisionTreeRegressor	3.62	2.02e+06	0.993	Модель показывает хорошие результаты, и ее простота и интерпретируемость могут быть полезными	Модель более склонна к переобучению и может недостаточно хорошо обобщать данные
RandomForestRegressor	3.02	1.12e+06	0.996	Модель обладает высокой точностью и устойчивостью к переобучению	Возможно, модель накладывает слишком сильные ограничения на данные, что может привести к недостаточной гибкости модели
GradientBoostingRegressor	3.69	1.34e+06	0.995	Модель хорошо обобщает данные и обладает высокой точностью	Возможно, модель требует более тщательной настройки гиперпараметров для достижения лучших результатов
CatBoostRegressor	3.54	1.08e+06	0.996	Модель обладает высокой точностью и устойчивостью к переобучению	Модель может быть более медленной в обучении и предсказаниях из-за особенностей своего алгоритма
XGBRegressor	3.05	1.05e+06	0.996	Модель показывает высокую точность и обладает гибкостью для настройки гиперпараметров	Модель может требовать больше вычислительных ресурсов и времени для обучения и предсказания

Модель	MAE	MSE	R^2	Почему лучше	Почему хуже
LGBMRegressor	3.21	1.06e+06	0.996	Модель обладает высокой скоростью обучения и предсказания, что делает ее эффективной в использовании	Модель может быть более чувствительной к выбросам и шумам в данных, что может снизить ее точность

В данной таблице представлены результаты оценки производительности различных моделей машинного обучения на основе нескольких метрик: средней абсолютной ошибки (MAE), среднеквадратичной ошибки (MSE) и коэффициента детерминации (R^2). Каждая модель была обучена и протестирована на данных о ценах биткойна с биржи Binance.

4 Выводы

- Модели CatBoostRegressor и RandomForestRegressor показали наилучшие результаты по всем метрикам среди всех рассмотренных моделей. Они обладают высокой точностью предсказаний (низкими значениями MAE и MSE) и хорошо обобщают данные (высокие значения R^2).
- Модели XGBRegressor и LGBMRegressor также показали отличные результаты и конкурентоспособны с CatBoostRegressor и RandomForestRegressor. Они обладают высокой точностью предсказаний и обобщают данные схожим образом.
- Ridge и DecisionTreeRegressor показали менее точные результаты по сравнению с другими моделями. Вероятно, они не учитывают некоторые сложные нелинейные зависимости в данных, что снижает их производительность.
- Модель GradientBoostingRegressor также показала хорошие результаты, но она более склонна к переобучению, что может негативно сказаться на ее обобщающей способности.
- При сравнении между CatBoostRegressor и RandomForestRegressor, стоит отметить, что CatBoostRegressor обучается медленнее из-за особенностей своего алгоритма, но обладает аналогичной точностью.

Таким образом, для данной задачи прогнозирования цен биткойна наиболее эффективными моделями оказались CatBoostRegressor и RandomForestRegressor, за ними следуют XGBRegressor и LGBMRegressor.

Библиографический список

1. Ortu M. et al. On technical trading and social media indicators in cryptocurrencies price classification through deep learning // Expert systems with applications. 2022. Т. 198. № 1. С. 1-45.

2. Fang F., Ventre C., Basios M. Cryptocurrency trading: a comprehensive survey // Financial innovation. 2022. Т. 8. № 1. С. 59.
3. Ji S., Kim J., Im H. A comparative study of bitcoin price prediction using deep learning // Mathematics. 2019. Т. 7. № 10. С. 20.
4. Yamin M., Chaudhry M. Cryptocurrency market trend and direction prediction using machine learning: a comprehensive survey // Authorea, 2023. С. 26. URL: https://www.authorea.com/doi/full/10.22541/au.167285886.66422340 (дата обращения 20.04.2024).
5. Akyildirim E., Goncu A., Sensoy A. Prediction of cryptocurrency returns using machine learning //Annals of Operations Research. 2021. Т. 297. С. 3-36.
6. Mangla N. et al. Bitcoin price prediction using machine learning // International Joutnal of Information And Computing Science. 2019. Т. 6. №. 5. С. 318-320.
7. McNally et al. Predicting the Price of Bitcoin Using Machine Learning // 2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP). 2018. Т. 26.
8. Chen T., Guestrin C. XGBoost: A Scalable Tree Boosting System. // The 22ndACM SIGKDD International Conference. 2016. С. 785–794.
9. Livieris I. et al. Ensemble Deep Learning Models for Forecasting Cryptocurrency Time-Series // Algorithms. 2020. Т. 13. № 5. С. 1-21.
10. Hassan A., Paila N., Bammidi V., Divvala A. V., et al. Bitcoin price prediction by using Arima // International journal of scientific research in engineering and management. 2024. С. 1-5.