СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном мире фондовый рынок России играет важную роль в экономической жизни страны и имеет значительное влияние на мировые финансовые процессы. Динамика российского фондового рынка непосредственно отражают состояние национальной экономики, макроэкономические показатели, как внутри страны, так и за ее пределами. В связи с этим, предсказание стоимости активов на российском фондовом рынке становится крайне важной задачей для инвесторов, компаний и государственных органов.

Традиционные методы анализа финансовых данных, такие как статистические модели или аналитические подходы, имеют свои ограничения в прогнозировании динамики рынка. Эти методы, хотя и имеют свое место в анализе, часто неспособны учитывать все сложности и нелинейные взаимосвязи, присущие современным финансовым рынкам. В последние годы методы машинного и глубокого обучения (ML/DL) привлекли широкое внимание в финансовой сфере благодаря своей способности адаптироваться к изменяющимся условиям рынка и обрабатывать большие объемы данных.

Применение ML/DL методов для предсказания стоимости активов на российском фондовом рынке открывает новые возможности для инвесторов и трейдеров. Эти методы позволяют анализировать не только статистические данные, но и включать в расчеты такие сложные факторы, как социальные, экономические, политические факторы.

В данной курсовой работе мы рассмотрим методы и модели построения точных прогнозов стоимости активов, в частности акций. Мы изучим основные подходы к решению данной задачи, а также рассмотрим современные методы и алгоритмы, применяемые в этой области.

1. ML и DL методы, используемые для прогнозирования фондовго рынка

Создание хорошей модели прогнозирования цен на акции особенно сложна, поскольку она нелинейна - на цены акций влияют люди, а не только социально-политические и экономические факторы. Для качественного прогнозирования необходимо провести: исследование, основанное на фундаментальных показателях компании, чьи акции являются целью, и технический анализ, основанный на прошлом поведении акций и связанных с ним закономерностях. Это делает проблему не только нелинейной, но и весьма динамичной. Прогнозирование фондового рынка с использованием методов машинного и глубокого обучения — правильный путь вперед. Методы, основанные на машинном и глубоком обучении, могут быть очень сложными и позволяют охватить этот сложный мир фондового рынка и то, как различные факторы влияют на цену акций.

Мы начнем с самых простых методов и постепенно будем продвигаться к более продвинутым. Существует несколько подходов к решению этой задачи, каждый со своими преимуществами и недостатками:

1. Скользящие средние (moving average)
2. Линейная регрессия.
3. Регрессия опорных векторов (SVR).
4. K nearest neighbours
5. Рекуррентная нейронная сеть (RNN)
6. Сверточная нейронная сеть (CNN)
7. Случайный лес
8. LSTM

1.1 Скользящие средние (moving average)

Скользящее среднее является одним из наиболее широко используемых методов технического анализа для прогнозирования цен на акции. Этот метод использует историческую информацию о ценах для выявления тенденций и определения возможных точек входа и выхода из рынка.

Суть метода заключается в расчете средней цены актива за определенный период времени. Этот период, называемый "периодом скользящего среднего", может варьироваться от нескольких дней до нескольких месяцев или лет, в зависимости от предпочтений трейдера и временного горизонта торговли.

Например, 20-дневное скользящее среднее рассчитывается как среднее арифметическое цен закрытия за последние 20 торговых дней. Это значение постоянно обновляется, поскольку более новые цены добавляются, а самые старые удаляются из расчета.

Скользящие средние часто используются в качестве индикаторов тренда. Если текущая цена актива находится выше скользящего среднего, это может указывать на восходящий тренд, в то время как если цена ниже скользящего среднего, это может быть признаком нисходящего тренда.

Трейдеры также часто комбинируют несколько скользящих средних с различными периодами для получения более точных сигналов. Например, пересечение более короткого скользящего среднего над более длинным может рассматриваться как сигнал к покупке, в то время как пересечение снизу вверх может быть сигналом к продаже.

Одним из преимуществ использования скользящих средних является их простота и наглядность. Однако их также критикуют за то, что они являются запаздывающими индикаторами, реагирующими на уже произошедшие изменения цен, а не предсказывающими их.

1.2 Линейная регрессия:

Один из самых простых методов, линейная регрессия, может быть использована для прогнозирования любых непрерывных значений, включая прогноз цены акций. Линейная регрессия, как следует из названия, представляет собой линейный метод, т. е. он находит линейную комбинацию переменных X, которые используются для прогнозирования переменной Y (в данном случае цены акций). Основным преимуществом этого метода является высокая интерпретируемость, поскольку пользователь может знать, какой фактор больше влияет на цену акций и в какой степени. Недостаток заключается в том, что его область применения сильно ограничена. Многие предикторы не могут быть использованы, что необходимо для решения проблемы прогнозирования цен на акции. Пакеты на основе машинного обучения, такие как sci-kit, позволяют пользователю использовать линейную регрессию в среде машинного обучения. Некоторые библиотеки R также позволяют то же самое, но недостаток сохраняется.

1.3 Регрессия опорных векторов (SVR):

Метод опорных векторов, который когда-то считалась основным конкурентом нейронных сетей, представляет собой метод машинного обучения, который может решать проблемы регрессии и классификации. Первоначально разработанный для решения проблем классификации путем максимизации поля, он представил аналогичную концепцию для решения проблем, основанных на регрессии, только путем корректировки эпсилона, где ошибка этих точек данных внутри поля не рассчитывается. Это обеспечивает то же преимущество, что и типичный классификатор опорных векторов: уменьшение переобучения, лучшее обобщение и точные результаты. В отличие от линейной регрессии, он может работать с наборами данных большого размера. Хотя мы сравниваем его с линейной регрессией,

остается одна проблема: метод линеен, тогда как проблема часто нелинейна. Эта проблема решается использованием ядер, которые могут заставить этот алгоритм работать нелинейно. Среди традиционных методов, основанных на машинном обучении, это самый продвинутый и точный метод, но он снова не может решить очень сложные и динамичные проблемы, которые являются сильной стороной методов, основанных на глубоком обучении.

1.4 K Ближайший сосед (KNN):

Теоретически K Nearest Neighbor (KNN) кажется идеальным методом. Когда мы говорим, что, действия каждого человека уникальны и обладают свободой воли в группе, их поведение демонстрирует закономерности, что делает его предсказуемым. Это дает специалисту по данным возможность искать похожие закономерности и прогнозировать цель. Поэтому KNN кажется идеальным кандидатом. Это метод, основанный на расстоянии, который при наблюдении ищет наиболее похожие записи, а затем прогнозирует значение на основе результатов этих записей. Таким образом, теоретически для ситуации на рынке KNN может искать наиболее похожую историческую ситуацию, узнавать, как вёл себя рынок, и прогнозировать цену акций. Хотя идея кажется заманчивой и очень простой для понимания, она не работает так, как ожидалось. Основная проблема этого метода заключается в том, что он будет учитывать все предоставленные нами переменные; таким образом, он всегда придает равную важность всем предикторам. В дополнение к этому, это очень трудоемкий метод, и его нельзя использовать для внутридневной торговли. Наконец, прогноз очень чувствителен к значению K и выбранной метрике расстояния, что может сильно изменить результаты, делая метод жестким и не таким динамичным, как необходимо.

1.5 Рекуррентная нейронная сеть (RNN)

Рекуррентная нейронная сеть (RNN) — это распространенный метод глубокого обучения, часто используемый для создания чатов и других моделей, работающих с текстовыми и другими неструктурированными данными. Он принимает данные из настоящего и прошлого и идеально подходит для имитации той части человеческого мозга, которая занимается задачами, требующими кратковременной памяти. Это делает его идеальным кандидатом для решения динамических задач, таких как прогнозирование постоянно меняющихся цен активов. Как мы теперь знаем, прогнозирование цен на акции — это очень динамичная проблема с постоянными колебаниями цен и окружающей среды. Вот почему исследователи экспериментировали с RNN для прогнозирования цен на акции и добились достойных результатов.

1.6 Сверточная нейронная сеть (CNN):

Сверточная нейронная сеть — это метод, известный для решения задач на основе изображений, таких как классификация изображений, распознавание лиц и т. д. Благодаря своей уникальной архитектуре понимания функций с помощью нескольких сверточных сетей исследователи сочли ее кандидатом для прогнозирования цен на акции. Среди ANN, RNN и CNN, CNN показала наиболее многообещающие результаты некоторых исследований. Однако в конечном итоге наиболее успешным методом на данный момент стал LSTM, который разберем ниже.

1.7 Случайный лес (KNN):

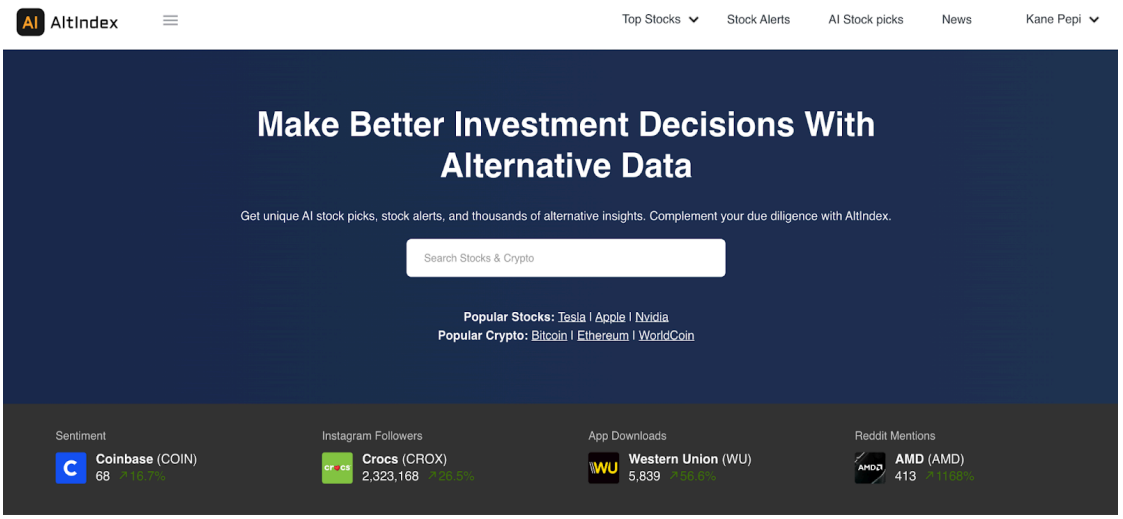
Случайный лес, считающийся одним из наиболее мощных методов на основе деревьев, может решать проблемы, основанные на регрессии, используя методы начальной загрузки и вставки. Он случайным образом выбирает различные функции и создает несколько моделей «Дерево решений». Этот способ может помочь спрогнозировать цены на акции. Кроме того, он имеет то преимущество, что его легко интерпретировать. Однако даже несмотря на эти преимущества и его расширенные версии, такие как Gradient и XG Boost, этот метод недостаточно сложен для решения очень сложной проблемы прогнозирования цен на акции. Это подводит нас к исследованию часто рассматриваемого подмножества машинного обучения – глубокого обучения.

1.8 Долгосрочная краткосрочная память (LSTM):

Долгосрочная краткосрочная память (LSTM) — это тип RNN; однако, в отличие от RNN, его архитектура гораздо сложнее. В то время как RNN имеет тенденцию забывать долгосрочные закономерности, LSTM имеет блок памяти, с помощью которого также можно хранить и использовать долгосрочные «воспоминания». LSTM — это широко используемый метод, поскольку он обеспечивает лучшие из всех алгоритмов, обсуждавшихся выше, т. е. он имеет долговременную память ANN, кратковременную память RNN и сложность CNN, что делает его широко используемым методом для решения задач прогнозирования цен активов.

2. Готовые продукты для прогнозирования фондового рынка:

1. AltIndex :

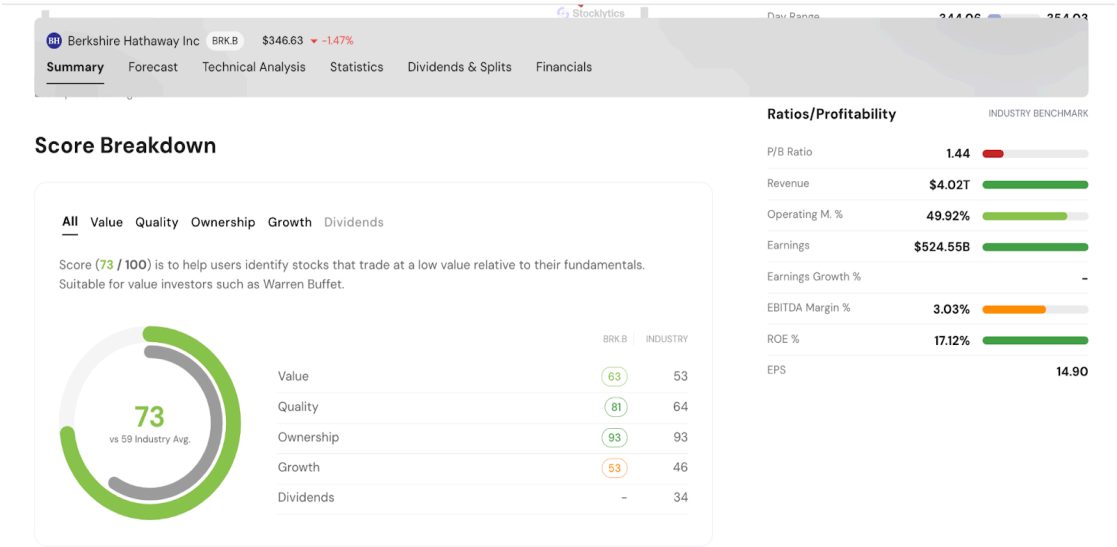


Лидер рынка альтернативных наборов данных с упором на анализ социальных настроений и потребительских тенденций. AltIndex предлагает полноценный сервис прогнозирования с помощью искусственного интеллекта с точностью 75% с момента его создания. Каждый месяц распределяются от 1 до 25 акций в зависимости от выбранного тарифного плана. AltIndex также предлагает прогнозы цен на основе искусственного интеллекта, не говоря уже о биржевых оповещениях и технических рейтингах.

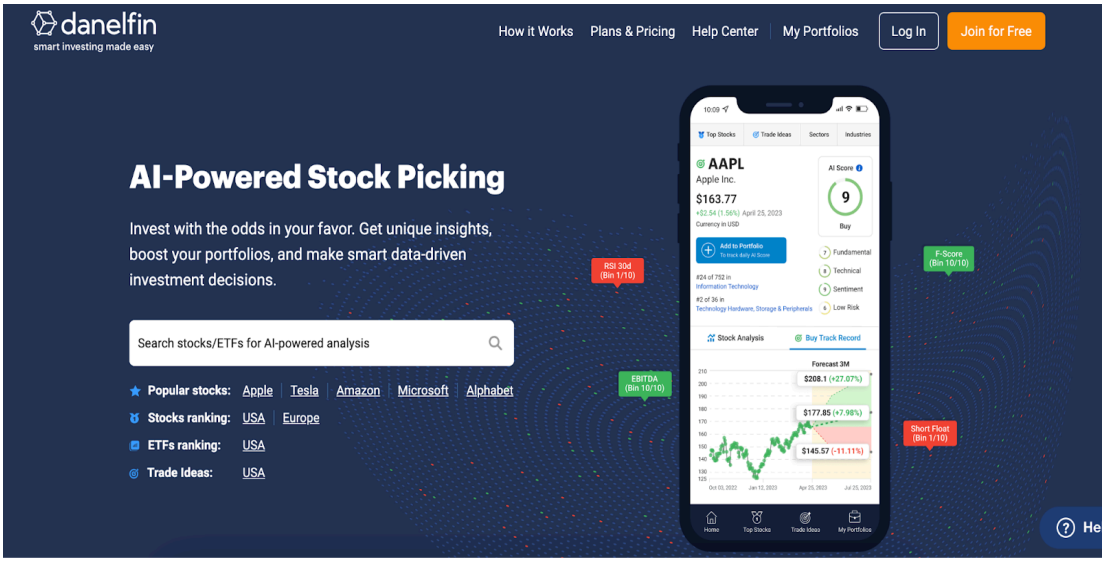
2. Stocklytics :

Извлекает технические и фундаментальные данные из более чем 5700 акций и 2400 ETF. Stocklytics использует искусственный интеллект для создания технических рейтингов от 1 до 100. Согласно алгоритму Stocklytics, акции с высоким рейтингом представляют собой надежные инвестиции. Stocklytics также использует искусственный интеллект для

прогнозирования цен на акции на ближайшие 7 и 30 дней. Затем это можно сравнить с рейтингами аналитиков продавцов.



3. Danelfin : Этот высокоэффективный прогноз акций на основе искусственного интеллекта превзошел индекс S&P 500 с момента его создания в 2017 году – с ростом на 191%. Danelfin отслеживает и анализирует все акции на биржах США, а также индекс STOXX Europe 600. Он использует искусственный интеллект для получения оценки инвестиционной привлекательности от 1 до 10.

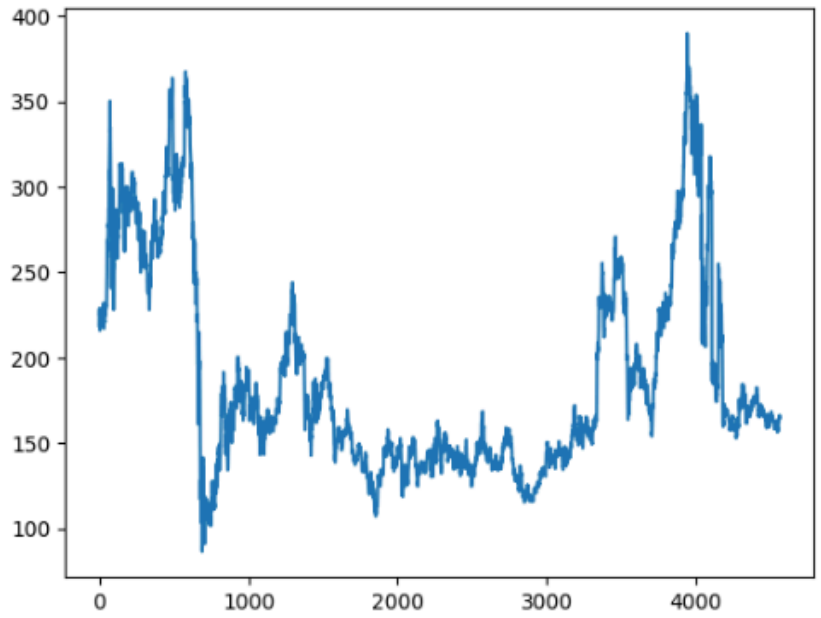


3. Загрузка данных

Для загрузки необходимых данных о акциях будем использовать MOEX API - api Московской биржи. MOEX API имеет подробную документацию и остается правильно составить нужный запрос к конкретному endpoint-у.

В нашем случае получаем данные о итогах торгов акций Газпрома с тикером GASP за период с 2006 по 2024 год. Для этого определим функцию get_data, которая формирует URL для запроса к API, предоставляемому Московской биржей, для получения свечей (candles) по акциям компании "Газпром" (GAZP), устанавливает параметры запроса, включая начальную и конечную даты (from_date и till_date), а также интервал свечей (interval), отправляет GET-запрос к API с заданными параметрами. Если ответ от сервера успешный (код статуса 200), функция преобразует полученные данные в список словарей, где каждый словарь представляет собой одну свечу с данными по столбцам: время открытия, время закрытия, минимальная цена, максимальная цена, цена открытия, цена закрытия, объем торгов. В случае ошибки выводит сообщение об ошибке и возвращает пустой список. У самого MOEX API стоит ограничение на максимум 500 запросов, поэтому если не использовать цикл и частичное порционное получение данных, то запрос отработает только частично. Поэтому определяем начальную и конечную даты (start_date и end_date), а также интервал свечей (interval), создаем последовательность дат (parts) с частотой в 1 неделю, начиная с start_date и заканчивая end_date, инициализируем пустой список all_data для хранения всех полученных данных и итерируем по датам в parts, для каждой пары дат преобразуем даты в строковый формат, вызываем функцию get_data для получения данных за указанный период, добавляем полученные данные в список all_data. Полученные данные помещаем в датафрейм и сохраняем файл с .csv расширением.

```python
import pandas as pd
import requests
from matplotlib import pyplot as plt

def get_data(from_date, till_date, interval='24'):
    url = 'http://iss.moex.com/iss/engines/stock/markets/shares/securities/GAZP/candles.json'
    params = {
        'from': from_date,
        'till': till_date,
        'interval': interval
    }
    response = requests.get(url, params=params)
    if response.status_code == 200:
        data = response.json()
        return [{k: r[i] for i, k in enumerate(data['candles']['columns'])} for r in data['candles']['data']]
    else:
        print(f"Ошибка при получении данных: {response.status_code}")
        return []

start_date = '2006-01-10'
end_date = '2024-04-17'
interval = '24'
parts = pd.date_range(start=start_date, end=end_date, freq='1W')

all_data = []
for i in range(len(parts) - 1):
    from_date = parts[i].strftime('%Y-%m-%d')
    till_date = parts[i + 1].strftime('%Y-%m-%d')
    data = get_data(from_date, till_date, interval)
    all_data.extend(data)

frame = pd.DataFrame(all_data)
frame.to_csv('historical_data_GAZP3.csv', index=False)
plt.plot(list(frame['close']))
plt.savefig("shares_GAZP3.png")
```



4. Визуализация и анализ данных

Полученные с API данные содержат цену открытия и цену закрытия, максимальную цену, минимальную цену, количество и объем акций за день торгов. В следующей таблице показаны данные о ценах на акции Газпром с начала 2006 года.

```
df
```

	open	close	high	low	value	volume	begin	end
0	239.50	218.89	239.50	218.49	1.130995e+09	5120765	2006-01-23 00:00:00	2006-01-23 23:59:59
1	221.00	224.00	224.68	219.66	1.991942e+09	8983192	2006-01-24 00:00:00	2006-01-24 23:59:59
2	225.50	228.38	231.00	225.00	3.534137e+09	15480374	2006-01-25 00:00:00	2006-01-25 23:59:59
3	228.20	224.47	229.41	223.51	1.721376e+09	7588759	2006-01-26 00:00:00	2006-01-26 23:59:59
4	225.75	228.75	231.50	224.00	2.901323e+09	12729718	2006-01-27 00:00:00	2006-01-27 23:59:59
...	...	...	...	...	...	...	...	...
4557	164.00	163.89	164.90	163.57	2.709310e+09	16501080	2024-04-08 00:00:00	2024-04-08 23:59:59
4558	164.00	164.08	165.79	163.49	4.787296e+09	29074240	2024-04-09 00:00:00	2024-04-09 23:59:59
4559	164.10	164.74	165.94	163.59	3.831243e+09	23220690	2024-04-10 00:00:00	2024-04-10 23:59:59
4560	164.97	166.24	166.33	164.15	5.098319e+09	30798110	2024-04-11 00:00:00	2024-04-11 23:59:59
4561	166.52	164.83	166.80	164.69	4.366297e+09	26345100	2024-04-12 00:00:00	2024-04-12 23:59:59

4562 rows × 8 columns

Проведем статистический анализ собранных данных. Можно делать это вручную

```
print("Основные статистические показатели:")
print("Среднее значение (Mean):", df['open'].mean())
print("Медиана (Median):", df['open'].median())
print("Минимальное значение (Min):", df['open'].min())
print("Максимальное значение (Max):", df['open'].max())
print("Стандартное отклонение (Standard Deviation):", df['open'].std())
print("Дисперсия (Variance):", df['open'].var())
print("Сумма (Sum):", df['open'].sum())
print("Количество (Count):", df['open'].count())
```

```
Основные статистические показатели:
Среднее значение (Mean): 188.46686102586585
Медиана (Median): 165.33
Минимальное значение (Min): 88.0
Максимальное значение (Max): 395.0
Стандартное отклонение (Standard Deviation): 61.007551635478336
Дисперсия (Variance): 3721.9213565555556
Сумма (Sum): 859785.8200000001
Количество (Count): 4562
```

А можно в одну команду для всех столбцов:

```
df.describe()
```

	open	close	high	low	value	volume
count	4562.000000	4562.000000	4562.000000	4562.000000	4.562000e+03	4.562000e+03
mean	188.466861	188.350962	190.824042	185.803477	9.183805e+09	4.670786e+07
std	61.007552	60.938687	61.833068	59.957766	7.826177e+09	3.111240e+07
min	88.000000	86.600000	94.370000	84.000000	2.274217e+08	0.000000e+00
25%	143.700000	143.795000	145.277500	142.102500	4.016927e+09	2.556152e+07
50%	165.330000	165.385000	167.200000	163.255000	6.917096e+09	3.861566e+07
75%	225.500000	225.395000	228.415000	222.157500	1.189308e+10	6.009251e+07
max	395.000000	389.820000	397.640000	378.020000	1.170773e+11	4.140240e+08

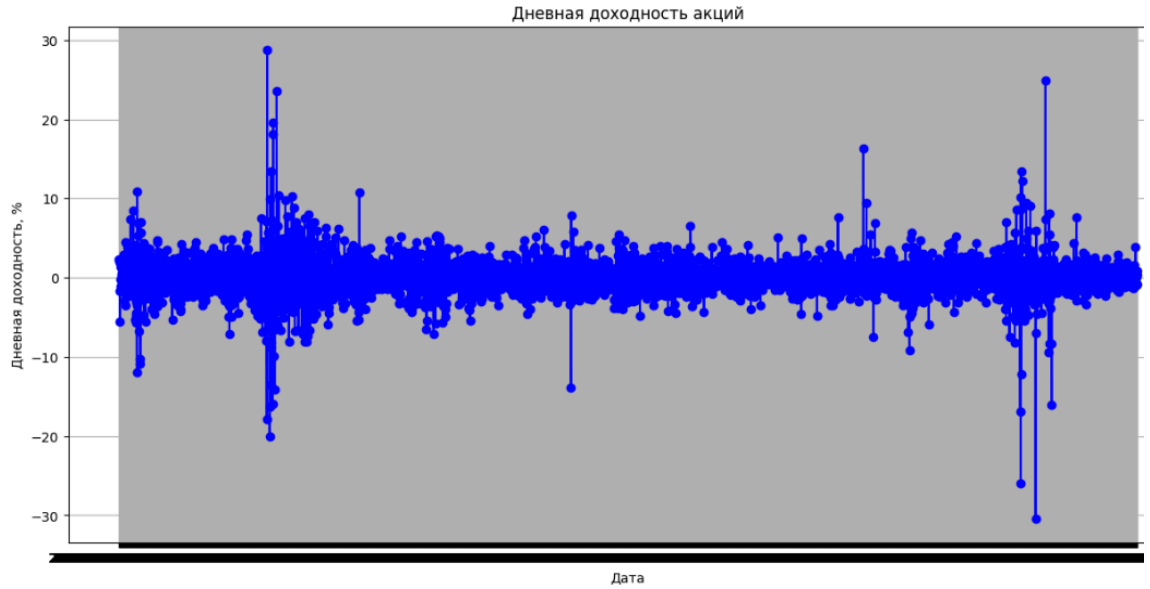
Получаем подробную сводку данных по среднему значению, медиане, стандартному отклонению, мин. и макс. значению, квартилям.

Отобразим график изменения цены в период с 2006 по 2024 год.

Также рассчитаем дневную доходность акций. Дневная доходность акций рассчитывается как процентное изменение между ценой закрытия сегодня и ценой закрытия вчера:

{Дневная доходность} = ({Цена закрытия сегодня} - {Цена закрытия вчера}) / {Цена закрытия вчера} * 100%

```
df['daily_return'] = (df['close'] - df['close'].shift(1)) / df['close'].shift(1) * 100

# Построение графика
plt.figure(figsize=(15, 7))
plt.plot(df['end'], df['daily_return'], marker='o', linestyle='-', color='b')
plt.xlabel('Дата')
plt.ylabel('Дневная доходность, %')
plt.title('Дневная доходность акций')
plt.grid(True)
plt.show()
```



Пиковая доходность акций, как и просадки по активу не превышают 30% внутридневного изменения цены, а в среднем дневная доходность находится на уровне 2-5%.

Также рассчитываем совокупную годовую доходность и риск в следующем коде, и результат показывает, что годовая доходность составляет 5.5%, а риск — 37.5%.

```
def ann_risk_return(returns_df):
    summary = returns_df.agg(["mean", "std"]).T
    summary.columns = ["Return", "Risk"]
    summary.Return = summary.Return*252
    summary.Risk = summary.Risk * np.sqrt(252)
    return summary
summary = ann_risk_return(ret)
```

```
print(summary)
```

```
        Return      Risk
ADBE  0.055342  0.375102
```

5. Построение и обучение моделей, сравнение результатов

5.1 Подготовка, очистка и нормализация данных

В предыдущих разделах мы ознакомились с используемыми ML и DL методами для прогнозирования стоимости акций фондового рынка. После чего мы собрали необходимые данные с помощью MOEX API и провели статистический анализ и визуализировали полученные данные.

Следующим шагом мы будем на практике использовать все 8 методов для прогнозирования стоимости акций фондового рынка - скользящие средние (moving average), линейную регрессию, регрессию опорных векторов (SVR), K nearest neighbours, случайный лес, рекуррентные нейронные сети (RNN), сверточные нейронные сети (CNN) и LSTM.

Импортируем все необходимые модули и библиотеки:

```
import tensorflow as tf

import pandas as pd
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
from keras.models import Model
from keras.layers import Input, Dense, LSTM, Dropout
from keras.layers import BatchNormalization
from keras.callbacks import EarlyStopping, ModelCheckpoint
```

Сделаем несколько преобразований над датафреймом - создадим новый столбец “date” в формате datetime, удалим столбцы “time” и “begin”:

```
df['time'] = pd.to_datetime(df['begin'])
df['date'] = df['time'].dt.strftime('%Y-%m-%d')
```

```
df = df.drop('time',axis=1)
df = df.drop('begin',axis=1)
```

Окончательный датафрейм будет выглядеть следующим образом:

```
df
```

	open	close	high	low	value	volume	end	date
0	239.50	218.89	239.50	218.49	1.130995e+09	5120765	2006-01-23 23:59:59	2006-01-23
1	221.00	224.00	224.68	219.66	1.991942e+09	8983192	2006-01-24 23:59:59	2006-01-24
2	225.50	228.38	231.00	225.00	3.534137e+09	15480374	2006-01-25 23:59:59	2006-01-25
3	228.20	224.47	229.41	223.51	1.721376e+09	7588759	2006-01-26 23:59:59	2006-01-26
4	225.75	228.75	231.50	224.00	2.901323e+09	12729718	2006-01-27 23:59:59	2006-01-27
...	...	...	...	...	...	...	...	...
4557	164.00	163.89	164.90	163.57	2.709310e+09	16501080	2024-04-08 23:59:59	2024-04-08
4558	164.00	164.08	165.79	163.49	4.787296e+09	29074240	2024-04-09 23:59:59	2024-04-09
4559	164.10	164.74	165.94	163.59	3.831243e+09	23220690	2024-04-10 23:59:59	2024-04-10
4560	164.97	166.24	166.33	164.15	5.098319e+09	30798110	2024-04-11 23:59:59	2024-04-11
4561	166.52	164.83	166.80	164.69	4.366297e+09	26345100	2024-04-12 23:59:59	2024-04-12

4562 rows × 8 columns

Проведем нормализацию данных по значениям закрытия цены акции в конце дня по окончанию торговой сессии:

```
# Extracting the closing prices of each day
fullData = df[['close']].values
print(fullData[0:5])

# Feature Scaling for fast training of neural networks
from sklearn.preprocessing import StandardScaler, MinMaxScaler

# Choosing between Standardization or normalization
#sc = StandardScaler()
sc=MinMaxScaler()

dataScaler = sc.fit(fullData)
X=dataScaler.transform(fullData)
#X=FullData

print('### After Normalization ###')
X[0:5]
```

```
[[218.89]
 [224.  ]
 [228.38]
 [224.47]
 [228.75]]
### After Normalization ###
array([[0.43628389],
       [0.45313634],
       [0.46758129],
       [0.45468637],
       [0.46880153]])
```

В данном примере кода извлекаем столбец “close” из исходного DataFrame df и преобразуем его в NumPy массив с помощью .values. Таким образом, fullData будет содержать цены закрытия для каждого дня в виде одномерного массива NumPy. Далее импортируем класс MinMaxScaler из библиотеки scikit-learn для проведения нормализации данных. Создаем экземпляр MinMaxScaler и методом fit вычисляем параметры нормализации (минимальное и максимальное значения) на основе исходных данных. Метод transform применяет нормализацию к исходным данным fullData, используя параметры, вычисленные на предыдущем шаге. Результат нормализации сохраняется в переменной X. В конце выводим первые пять элементов ненормализованного и нормализованного массива X, чтобы увидеть результат нормализации.

Нормализация данных (приведение значений к диапазону от 0 до 1) важна по нескольким причинам:

1. Сходимость алгоритмов: многие алгоритмы машинного обучения, такие как нейронные сети, работают лучше и сходятся быстрее, когда данные находятся в одном диапазоне значений. Это связано с тем, что градиентные методы оптимизации работают более эффективно при отсутствии больших разбросов в величинах признаков.

2. Предотвращение доминирования: если признаки имеют разные диапазоны значений, признаки с большими значениями могут доминировать над признаками с меньшими значениями при вычислении расстояний или весов в некоторых алгоритмах.

3. Численная стабильность: некоторые алгоритмы могут столкнуться с численными проблемами при обработке очень больших или очень маленьких значений. Нормализация помогает избежать этих проблем.

Далее подготовим данные для использования в рекуррентных нейронных сетях, таких как LSTM:

```
X_samples = list()
y_samples = list()

NumerOfRows = len(X)
TimeSteps=10  # next day's Price Prediction is based on last how many past day's prices

# Iterate thru the values to create combinations
for i in range(TimeSteps , NumerOfRows , 1):
    x_sample = X[i-TimeSteps:i]
    y_sample = X[i]
    X_samples.append(x_sample)
    y_samples.append(y_sample)

# Reshape the Input as a 3D (number of samples, Time Steps, Features)
X_data=np.array(X_samples)
X_data=X_data.reshape(X_data.shape[0],X_data.shape[1], 1)
print('\n#### Input Data shape ####')
print(X_data.shape)

# We do not reshape y as a 3D data  as it is supposed to be a single column only
y_data=np.array(y_samples)
y_data=y_data.reshape(y_data.shape[0], 1)
print('\n#### Output Data shape ####')
print(y_data.shape)
```

```
#### Input Data shape ####
(4552, 10, 1)

#### Output Data shape ####
(4552, 1)
```

Каждый образец входных данных представляет собой последовательность цен закрытия за последние TimeSteps дней, а соответствующее целевое значение - цену закрытия следующего дня. Эта структура данных позволяет нейронной сети обучаться на временных последовательностях и предсказывать следующее значение на основе предыдущих значений.

Разделим данные на тестовые и тренировочные в пропорции 1 к 4:

```
TestingRecords= 1140

X_train=X_data[:-TestingRecords]
X_test=X_data[-TestingRecords:]
y_train=y_data[:-TestingRecords]
y_test=y_data[-TestingRecords:]

print('\n#### Training Data shape ####')
print(X_train.shape)
print(y_train.shape)
print('\n#### Testing Data shape ####')
print(X_test.shape)
print(y_test.shape)
```

```
#### Training Data shape ####
(3412, 10, 1)
(3412, 1)

#### Testing Data shape ####
(1140, 10, 1)
(1140, 1)
```

По итогу имеем подготовленные данные для дальнейшего обучения.

5.2 Метрики оценки моделей

В данном разделе разберем основные метрики оценки моделей, так как они будут повторяться от метода к методу.

1. **RMSE** (Root Mean Squared Error) - Корень из среднеквадратичной ошибки

RMSE - это квадратный корень из среднего значения квадратов ошибок между предсказанными и фактическими значениями. Она измеряет среднее отклонение предсказаний от истинных значений.

Формула:

RMSE = sqrt((sum((y_pred - y_true)^2)) / n),

где y_pred - предсказанные значения,

y_true - истинные значения,

n - количество наблюдений.

Диапазон значений: RMSE принимает значения от 0 до +∞, где 0 означает идеальное предсказание.

Интерпретация:

- Чем ниже значение RMSE, тем лучше модель.
- RMSE = 0 означает идеальное предсказание.
- RMSE чувствительна к выбросам, так как ошибки возводятся в квадрат.

2. **MSE** (Mean Squared Error) - Среднеквадратичная ошибка

MSE - это среднее значение квадратов ошибок между предсказанными и фактическими значениями. Она измеряет среднее отклонение предсказаний от истинных значений в квадрате.

Формула:

MSE = (sum((y_pred - y_true)^2)) / n,

где y_pred - предсказанные значения,

y_true - истинные значения,

n - количество наблюдений.

Диапазон значений: MSE принимает значения от 0 до +∞, где 0 означает идеальное предсказание.

Интерпретация:

- Чем ниже значение MSE, тем лучше модель.

- MSE = 0 означает идеальное предсказание.

- MSE также чувствительна к выбросам, так как ошибки возводятся в квадрат.

3. **MAE** (Mean Absolute Error) - Средняя абсолютная ошибка

MAE - это среднее значение абсолютных отклонений предсказаний от истинных значений. Она измеряет среднее абсолютное отклонение предсказаний от истинных значений.

Формула:

MAE = (sum(abs(y_pred - y_true))) / n,

где y_pred - предсказанные значения,

y_true - истинные значения,

n - количество наблюдений.

Диапазон значений: MAE принимает значения от 0 до +∞, где 0 означает идеальное предсказание.

Интерпретация:

- Чем ниже значение MAE, тем лучше модель.

- MAE = 0 означает идеальное предсказание.

- MAE менее чувствительна к выбросам, чем RMSE и MSE, так как не возводит ошибки в квадрат.

- MAE имеет ту же единицу измерения, что и исходные данные, что делает ее интерпретацию более понятной.

4. **MAPE** (Mean Absolute Percentage Error) - Средняя абсолютная процентная ошибка

MAPE - это среднее значение абсолютных процентных отклонений предсказаний от истинных значений. Она измеряет среднее абсолютное процентное отклонение предсказаний от истинных значений.

Формула:

MAPE = (sum(abs((y_pred - y_true) / y_true)) / n) * 100%,

где y_pred - предсказанные значения,

y_true - истинные значения,

n - количество наблюдений.

Диапазон значений: MAPE принимает значения от 0% до +∞, где 0% означает идеальное предсказание.

Интерпретация:

- Чем ниже значение MAPE, тем лучше модель.

- MAPE = 0% означает идеальное предсказание.

- MAPE выражается в процентах, что делает ее более интуитивно понятной для интерпретации.

- MAPE может быть неопределенной или давать бесконечные значения, если истинные значения содержат нули.

5. **R^2 (R-squared)** - Коэффициент детерминации

R^2 - это метрика, которая показывает долю дисперсии в зависимой переменной, объясненную независимыми переменными модели. Она измеряет, насколько хорошо модель объясняет вариацию в данных.

Формула:

R^2 = 1 - (sum((y_pred - y_true)^2)) / (sum((y_true - mean(y_true))^2)),

где y_pred - предсказанные значения,

y_true - истинные значения.

Диапазон значений: R^2 принимает значения от 0 до 1, где 1 означает идеальное предсказание.

Интерпретация:

- Чем ближе значение R^2 к 1, тем лучше модель описывает данные.

- R^2 = 1 означает идеальное предсказание.

- R^2 = 0 означает, что модель не лучше, чем просто среднее значение зависимой переменной.

5.3 Предсказание цены акции методом скользящих средних

Метод скользящих средних в принципе невозможно отнести ни к методам машинного обучения, ни к методам глубокого обучения, однако в силу своей специфики данный метод может усреднено предсказать тренд, направленность изменения цены, но не точное предсказание малейших изменений и колебаний.

Метод скользящих средних для предсказания цены акций представлен в следующем коде:

```python
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
def moving_avg_prediction(df):
    shape = df.shape[0]
    df_new = df[['close']]
    df_new.head()
    train_set = df_new.iloc[:ceil(shape*0.75)]
    valid_set = df_new.iloc[ceil(shape*0.75):]
    print('-------------------------------------------------------------')
    print('-----------STOCK PRICE PREDICTION BY MOVING AVERAGE--------')
    print('-------------------------------------------------------------')
    print('Shape of Training Set', train_set.shape)
    print('Shape of Validation Set', valid_set.shape)
    preds = []
    for i in range(0, valid_set.shape[0]):
        a = train_set['close'][len(train_set)-valid_set.shape[0]+i:].sum() + sum(preds)
        b = a/(valid_set.shape[0])
        preds.append(b)

    # Расчет RMSE
    rmse = np.sqrt(np.mean(np.power((np.array(valid_set['close'])-preds), 2)))
    print('Root Mean Squared Error (RMSE) on validation set:', rmse)

    # Расчет среднеквадратичной ошибки (MSE)
    mse = mean_squared_error(valid_set['close'].values, preds)
    print('Mean Squared Error (MSE) on validation set:', mse)

    # Расчет средней абсолютной ошибки (MAE)
    mae = mean_absolute_error(valid_set['close'].values, preds)
    print('Mean Absolute Error (MAE) on validation set:', mae)

    # Расчет средней абсолютной процентной ошибки (MAPE)
    mape = np.mean(np.abs((valid_set['close'].values - preds) / valid_set['close'].values)) * 100
    print('Mean Absolute Percentage Error (MAPE) on validation set:', mape)

    # Расчет R-квадрат (R^2)
    r2 = r2_score(valid_set['close'].values, preds)
    print('R-squared (R^2) score on validation set:', r2)

    print('-------------------------------------------------------------')
    print('-------------------------------------------------------------')
    valid_set['Predictions'] = preds
    plt.plot(train_set['close'])
    plt.plot(valid_set[['close', 'Predictions']])
    plt.xlabel('Date', size=20)
    plt.ylabel('Stock Price', size=20)
    plt.title('Stock Price Prediction by Moving Averages', size=20)
    plt.legend(['Model Training Data', 'Actual Data', 'Predicted Data'])
```

Функция moving_avg_prediction работает по следующему алгоритму:

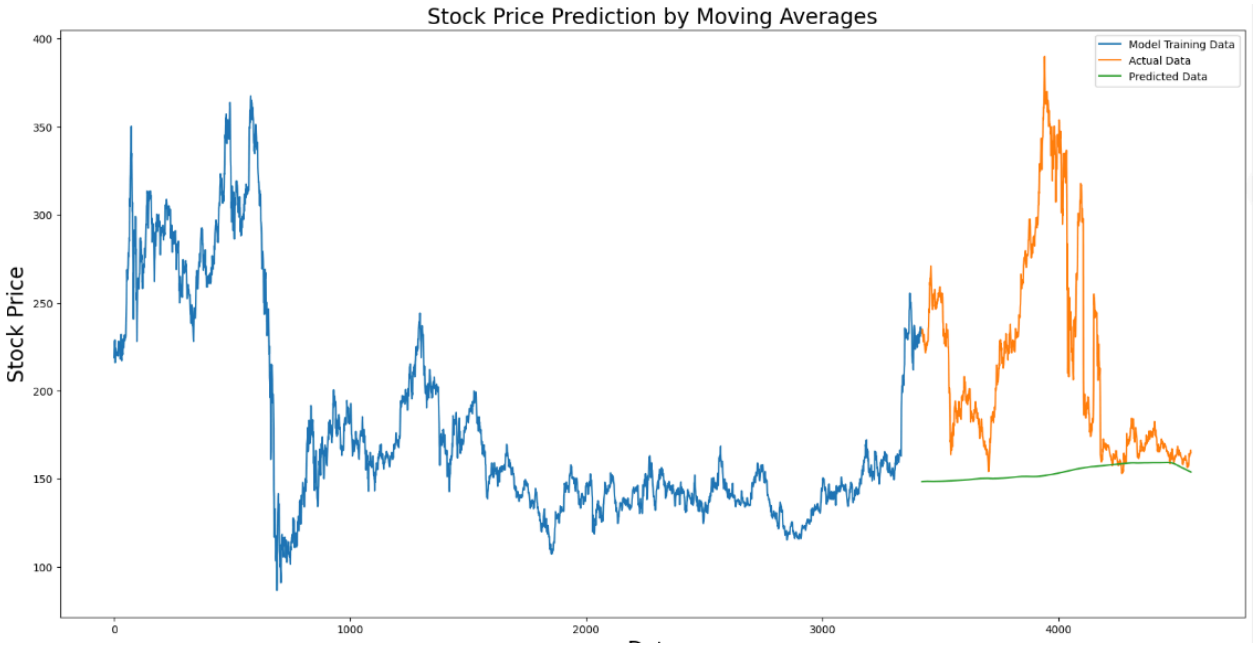
- Входные данные разделяются на обучающую (train_set) и проверочную (valid_set) выборки. Обучающая выборка составляет 75% от всех данных, а проверочная выборка - оставшиеся 25%.
- Создается пустой список preds, который будет содержать предсказанные значения для проверочной выборки.

- Вычисляется сумма последних valid_set.shape[0] значений из обучающей выборки train_set . Это сумма последних valid_set.shape[0] цен закрытия акций из обучающей выборки.
- К этой сумме прибавляется сумма уже предсказанных значений (sum(preds)).
- Полученная сумма делится на количество элементов в проверочной выборке (b = a/(valid_set.shape[0])), что дает среднее значение.
- Это среднее значение (b) добавляется в список preds в качестве предсказания для текущего элемента проверочной выборки.
- Далее производится расчет метрик RMSE, MSE, MAE, MAPE, R2

Результаты:

```
moving_avg_prediction(df)
```

```
------------------------------------------------------------
-----------STOCK PRICE PREDICTION BY MOVING AVERAGE--------
------------------------------------------------------------
Shape of Training Set (3422, 1)
Shape of Validation Set (1140, 1)
Root Mean Squared Error (RMSE) on validation set: 86.41834661123113
Mean Squared Error (MSE) on validation set: 7468.130631018884
Mean Absolute Error (MAE) on validation set: 62.88420551281515
Mean Absolute Percentage Error (MAPE) on validation set: 24.432685954077563
R-squared (R^2) score on validation set: -1.2441436944469872
------------------------------------------------------------
```



Исходя из полученных метрик, можно сделать следующие выводы об эффективности метода скользящих средних для предсказания цен акций на проверочной выборке:

1. Корень из среднеквадратичной ошибки (RMSE): 86.42

- Достаточно высокое значение RMSE указывает на значительное расхождение между предсказанными и фактическими ценами закрытия акций.

2. Среднеквадратичная ошибка (MSE): 7468.13

- Высокая величина MSE также свидетельствует о плохом качестве предсказаний.

3. Средняя абсолютная ошибка (MAE): 62.88

- MAE показывает, что в среднем предсказания отклоняются от фактических значений на 62.88 пункта (единиц измерения цены акций).

4. Средняя абсолютная процентная ошибка (MAPE): 24.43%

- MAPE равная 24.43% говорит о том, что предсказания в среднем отличаются от фактических значений на 24.43%.

5. R-квадрат (R^2): -1.24

- Отрицательное значение R^2 указывает на крайне плохое качество модели. Обычно R^2 принимает значения от 0 до 1, где 1 соответствует идеальной модели.

Исходя из полученных результатов, можно сделать вывод, что метод скользящих средних плохо справляется с задачей предсказания цен акций. Высокие значения ошибок (RMSE, MSE, MAE, MAPE) и отрицательный R^2 свидетельствуют о том, что предсказанные значения существенно отличаются от фактических.

Возможные причины низкого качества предсказаний:

1. Метод скользящих средних не учитывает другие факторы, влияющие на цену акций, кроме последних наблюдений.

2. Динамика цен акций может быть нелинейной и сложной, в то время как скользящее среднее - это простая линейная модель.

Для улучшения качества предсказаний следует рассмотрим более сложные модели, учитывающие различные факторы и нелинейные зависимости.

5.4 Предсказание цены акции с помощью линейной регрессии

Метод линейной регрессии может быть более подходящим для задачи предсказания цен акций по сравнению с методом скользящего среднего. Линейная регрессия позволяет учитывать другие факторы, которые могут влиять на цену акций, и строить более гибкую модель.

Метод скользящих средних для предсказания цены акций представлен в следующем коде:

Функция linear_regression_prediction работает по следующему алгоритму:

1. Определяется количество строк (shape[0]) в исходном DataFrame df.

2. Создается новый DataFrame df_new, содержащий только столбец 'close' (цена закрытия акции).

3. Данные разделяются на обучающую (train_set) и проверочную (valid_set) выборки в соотношении 75% и 25% соответственно с помощью iloc.

4. Выводятся информационные сообщения и форматы обучающей и проверочной выборок.

```
import pandas as pd
from datetime import datetime
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

def linear_regression_prediction(df):
    shape = df.shape[0]
    df_new = df[['close']]
    df_new.head()
    train_set = df_new.iloc[:ceil(shape*0.75)]
    valid_set = df_new.iloc[ceil(shape*0.75):]
    print('-----------------------------------------------------------------')
    print('-----------STOCK PRICE PREDICTION BY LINEAR REGRESSION-----------')
    print('-----------------------------------------------------------------')
    print('Shape of Training Set', train_set.shape)
    print('Shape of Validation Set', valid_set.shape)
    train = train_set.reset_index()
    valid = valid_set.reset_index()
    x_train = train['time'].apply(lambda x: datetime.strptime(x, '%m/%d/%Y %H:%M:%S').toordinal())
    y_train = train[['close']]
    x_valid = valid['time'].apply(lambda x: datetime.strptime(x, '%m/%d/%Y %H:%M:%S').toordinal())
    y_valid = valid[['close']]

    model = LinearRegression()
    model.fit(np.array(x_train).reshape(-1, 1), y_train)
    preds = model.predict(np.array(x_valid).reshape(-1, 1))

    rms = np.sqrt(np.mean(np.power((np.array(valid_set['close']) - preds), 2)))
    print('Root Mean Squared Error (RMSE) on validation set:', rms)

    mse = mean_squared_error(y_valid, preds)
    print('Mean Squared Error (MSE) on validation set:', mse)

    mae = mean_absolute_error(y_valid, preds)
    print('Mean Absolute Error (MAE) on validation set:', mae)

    mape = np.mean(np.abs((y_valid - preds) / y_valid)) * 100
    print('Mean Absolute Percentage Error (MAPE) on validation set:', mape)

    r2 = r2_score(y_valid, preds)
    print('R-squared (R^2) score on validation set:', r2)
    print('-------------------------------------------------------------')
    print('-------------------------------------------------------------')
    valid_set['Predictions'] = preds
    plt.plot(train_set['close'])
    plt.plot(valid_set[['close', 'Predictions']])
    plt.xlabel('Date', size=20)
    plt.ylabel('Stock Price', size=20)
    plt.title('Stock Price Prediction by Linear Regression', size=20)
    plt.legend(['Model Training Data', 'Actual Data', 'Predicted Data'])
```

5. Индексы в train_set и valid_set сбрасываются с помощью reset_index().

6. Столбец 'time' преобразуется в порядковые числа с помощью datetime.strptime и toordinal() для создания признаков x_train и x_valid.

7. Целевые переменные y_train и y_valid создаются из столбца 'close' для обучающей и проверочной выборок соответственно.

8. Создается экземпляр модели LinearRegression().

9. Модель обучается на x_train и y_train с помощью метода fit().

10. Делаются предсказания (preds) на проверочной выборке x_valid с помощью метода predict().

11. Вычисляется и выводится корень из среднеквадратичной ошибки (RMSE) на проверочной выборке с использованием np.sqrt и mean_squared_error, среднеквадратичная ошибка (MSE) на проверочной выборке с использованием mean_squared_error, средняя абсолютная ошибка (MAE) на проверочной выборке с использованием mean_absolute_error, средняя абсолютная процентная ошибка (MAPE) на проверочной выборке с помощью np.mean и np.abs, R-квадрат (R^2) на проверочной выборке с использованием r2_score.

12. Добавляется столбец 'Predictions' в valid_set со значениями предсказанных цен.

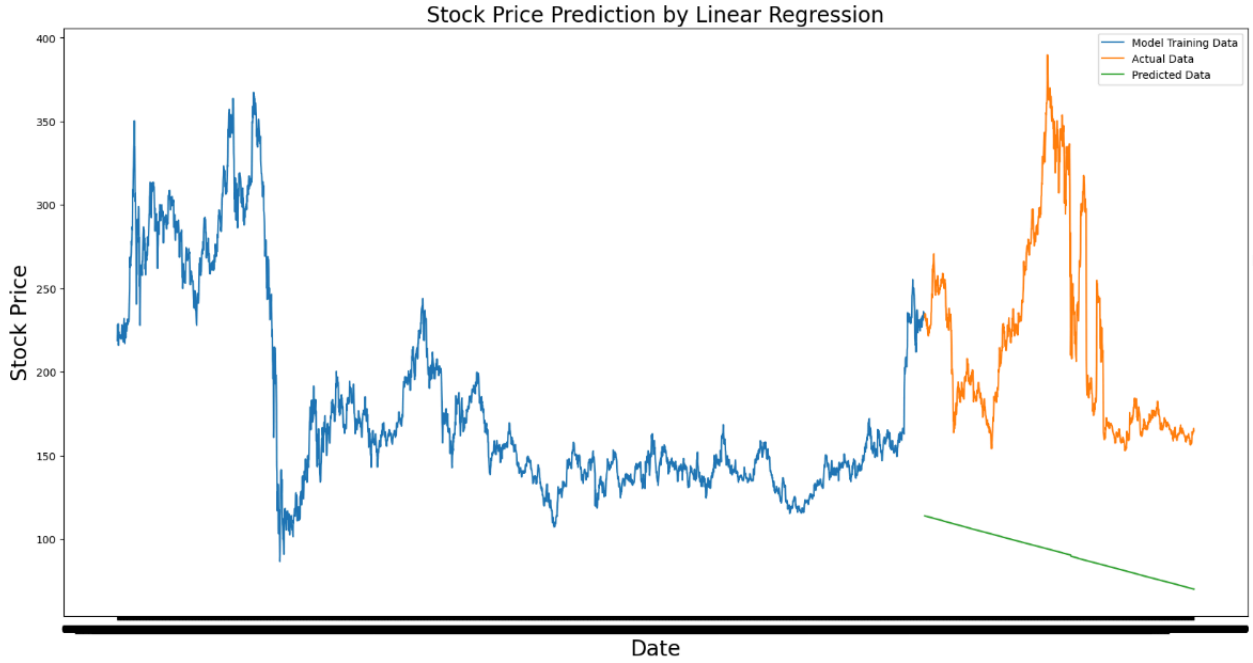
13. Строится график с тремя линиями: данными обучающей выборки, фактическими данными проверочной выборки и предсказанными данными на проверочной выборке.

Таким образом, с помощью функции linear_regression_prediction разделяем исходные данные на обучающую и проверочную выборки, обучаем модель линейной регрессии на обучающей выборке, делаем предсказания на проверочной выборке, вычисляем различные метрики качества моделирования, такие как RMSE, MSE, MAE, MAPE и R^2, и визуализируем результаты на графике.

Результаты:

```
linear_regression_prediction(df)
```

```
------------------------------------------------------------------
-----------STOCK PRICE PREDICTION BY LINEAR REGRESSION-----------
------------------------------------------------------------------
Shape of Training Set (3422, 1)
Shape of Validation Set (1140, 1)
Root Mean Squared Error (RMSE) on validation set: 137.92865526925218
Mean Squared Error (MSE) on validation set: 18519.32063573189
Mean Absolute Error (MAE) on validation set: 124.63657105470162
Mean Absolute Percentage Error (MAPE) on validation set: 55.55988071316901
R-squared (R^2) score on validation set: -4.564982548310033
------------------------------------------------------------
```



Сравним результаты метода линейной регрессии с полученными ранее результатами метода скользящих средних:

Метод линейной регрессии:
- RMSE: 137.93
- MSE: 18519.32
- MAE: 124.64
- MAPE: 55.56%
- R^2: -4.56

Метод скользящих средних:
- RMSE: 86.42
- MSE: 7468.13
- MAE: 62.88
- MAPE: 24.43%
- R^2: -1.24

Проведем анализ результатов:

1. Корень из среднеквадратичной ошибки (RMSE): Метод скользящих средних показывает значительно лучший результат (86.42) по сравнению с

линейной регрессией (137.93), что указывает на меньшие отклонения предсказаний от фактических значений.

2. Среднеквадратичная ошибка (MSE): Метод скользящих средних также демонстрирует гораздо меньшую среднеквадратичную ошибку (7468.13) по сравнению с линейной регрессией (18519.32).

3. Средняя абсолютная ошибка (MAE): Значение MAE для метода скользящих средних (62.88) почти в два раза меньше, чем для линейной регрессии (124.64), что указывает на более точные предсказания.

4. Средняя абсолютная процентная ошибка (MAPE): Метод скользящих средних имеет гораздо более низкую MAPE (24.43%) в сравнении с линейной регрессией (55.56%), что свидетельствует о лучшем качестве предсказаний.

5. R-квадрат (R^2): Хотя оба значения R^2 отрицательны, метод скользящих средних показывает значение (-1.24), которое ближе к 0 по сравнению с линейной регрессией (-4.56), что указывает на лучшее соответствие модели данным.

Таким образом метод скользящих средних демонстрирует значительно лучшие результаты по всем метрикам качества по сравнению с методом линейной регрессии для предсказания цен акций на данном наборе данных. Это может быть связано с нелинейным характером зависимости цен акций от времени, что затрудняет применение линейной регрессии. Метод скользящих средних, учитывающий историческую динамику цен, оказывается более подходящим для данной задачи.

5.5 Предсказание цены акции с помощью метода опорных векторов

Метод скользящих средних для предсказания цены акций представлен в следующем коде:

```
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error, mean_absolute_error,mean_absolute_percentage_error, r2_score

def svr_prediction(df):
    shape = df.shape[0]
    df_new = df[['close']]
    df_new.head()
    train_set = df_new.iloc[:ceil(shape*0.75)]
    valid_set = df_new.iloc[ceil(shape*0.75):]
    print('-----------------------------------------------------------------')
    print('----------STOCK PRICE PREDICTION BY SUPPORT VECTOR REGRESSION----------')
    print('-----------------------------------------------------------------')
    print('Shape of Training Set', train_set.shape)
    print('Shape of Validation Set', valid_set.shape)
    train = train_set.reset_index()
    valid = valid_set.reset_index()

    train['time'] = pd.to_datetime(train['time'])
    valid['time'] = pd.to_datetime(valid['time'])

    x_train = train['time'].map(dt.datetime.toordinal)
    y_train = train[['close']]
    x_valid = valid['time'].map(dt.datetime.toordinal)
    y_valid = valid[['close']]

    # Implement Support Vector Regression
    svr = SVR(kernel='rbf', C=1e3, gamma=0.1)
    svr.fit(np.array(x_train).reshape(-1, 1), y_train)
    preds = svr.predict(np.array(x_valid).reshape(-1, 1))
    # Calculate evaluation metrics
    mse = mean_squared_error(y_valid, preds)
    mae = mean_absolute_error(y_valid, preds)
    mape = mean_absolute_percentage_error(y_valid, preds)
    r2 = r2_score(y_valid, preds)
    print('MSE:', mse)
    print('MAE:', mae)
    print('MAPE:', mape)
    print('R^2:', r2)
    rms = np.sqrt(mse)
    print('RMSE value on validation set:', rms)
    print('-----------------------------------------------------------')
    print('-----------------------------------------------------------')
    valid_set['Predictions'] = preds
    plt.plot(train_set['close'])
    plt.plot(valid_set[['close', 'Predictions']])
    plt.xlabel('Date', size=20)
    plt.ylabel('Stock Price', size=20)
    plt.title('Stock Price Prediction by Support Vector Regression', size=20)
    plt.legend(['Model Training Data', 'Actual Data', 'Predicted Data'])
```

Функция svr_prediction работает по следующему алгоритму:

1. Принимает на вход DataFrame df с данными о ценах акций.

2. Разделяет данные на обучающую и валидационную (проверочную) выборки в соотношении 75% и 25% соответственно, используя столбец 'close' (цена закрытия).

3. Выводит информацию о форме (размере) обучающей и валидационной выборок.

4. Преобразует столбец 'time' в формат даты для обучающей и валидационной выборок.

5. Создает массивы x_train, y_train, x_valid, y_valid для обучения и валидации модели SVR.

6. Создает экземпляр модели SVR с заданными параметрами ядра ('rbf'), значением штрафа C=1e3 и параметром gamma=0.1.

7. Обучает модель SVR на обучающей выборке (x_train, y_train).

8. Получает прогнозы preds на валидационной выборке x_valid.

9. Вычисляет метрики качества: MSE (среднеквадратичная ошибка), RMSE (корень из среднеквадратичной ошибки) на основе MSE, MAE (средняя абсолютная ошибка), MAPE (средняя абсолютная процентная ошибка), R^2 (коэффициент детерминации) между прогнозами preds и истинными значениями y_valid.

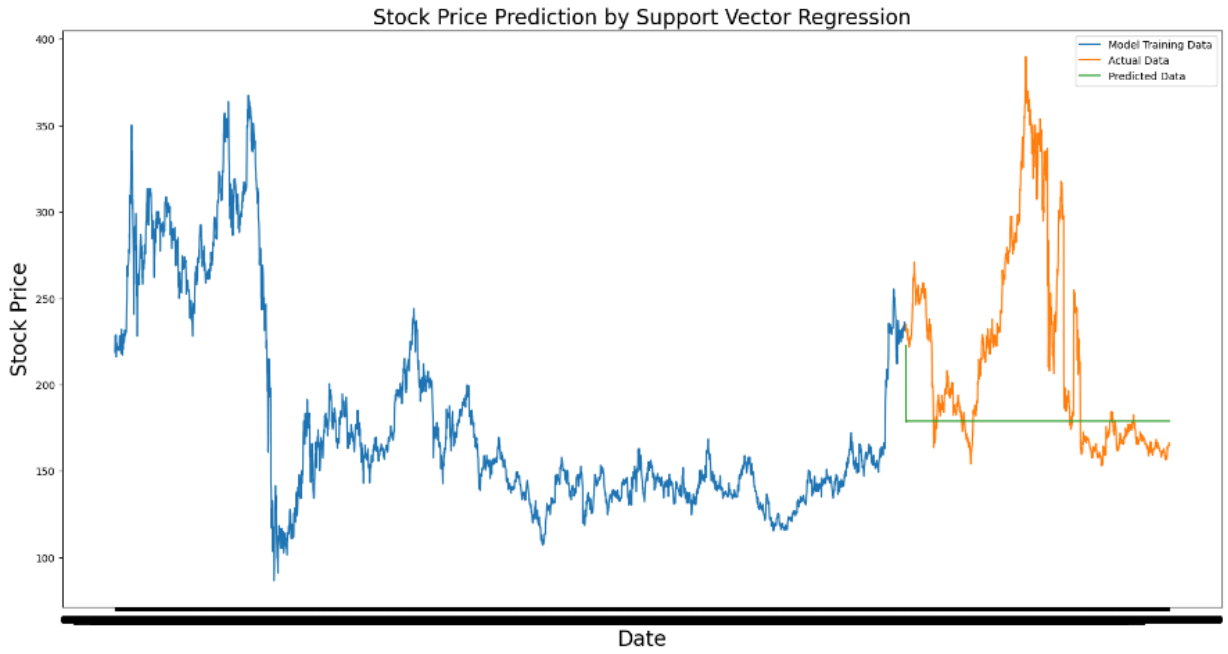
10. Выводит значения вычисленных метрик.

12. Добавляет столбец 'Predictions' в валидационную выборку valid_set с прогнозами preds.

13. Строит график, на котором отображаются исходные данные обучающей выборки, истинные значения валидационной выборки и прогнозы модели SVR.

Результаты:

```
MSE: 4741.812453959077
MAE: 47.04613486101782
MAPE: 0.18010268111994981
R^2: -0.4248958734872388
RMSE value on validation set: 68.86081944007839
--------------------------------------------------
```



Сравним результаты метода опорных векторов с полученными ранее результатами метода скользящих средних и метода линейной регрессии:

Метод опорных векторов (SVR):

- RMSE = 68.86
- MSE = 4741.81
- MAE = 47.05
- MAPE = 18%
- R^2 = -0.42

Метод линейной регрессии:

- RMSE: 137.93
- MSE: 18519.32
- MAE: 124.64
- MAPE: 55.56%
- R^2: -4.56

Метод скользящих средних:

- RMSE: 86.42
- MSE: 7468.13
- MAE: 62.88
- MAPE: 24.43%
- R^2: -1.24

Сравнивая эти результаты, можно сделать следующие наблюдения:

- Метод опорных векторов (SVR) показывает наименьшие значения MSE и MAE среди трех методов, что указывает на более точные прогнозы.

- SVR также имеет наименьшую среднюю абсолютную процентную ошибку (MAPE) в 18%, в то время как для линейной регрессии она составляет 55.56%, а для скользящих средних - 24.43%.

- Коэффициент детерминации (R^2) для SVR (-0.42) лучше, чем для скользящих средних (-1.24) и для линейной регрессии (-4.56).

Исходя из этих результатов, можно сделать вывод, что метод опорных векторов (SVR) показывает лучшие результаты прогнозирования по сравнению с линейной регрессией и скользящими средними, особенно по метрикам MSE, MAE и MAPE. Коэффициент детерминации (R^2) для SVR также лучше, чем у скользящих средних и линейной регрессии.

5.6 Предсказание цены акции с помощью метода K Nearest Neighbor

Реализуем функцию для обучения модели KNN (K-Nearest Neighbors) на исторических данных цен акций и получения прогнозов на валидационной выборке. Проанализируем различные метрики качества прогнозирования и визуализируем результаты на графике. В процессе обучения модели будем использовать перебор параметров с помощью GridSearchCV для поиска оптимального количества соседей n_neighbors. Признаки x_train и x_valid масштабируются перед обучением модели.

Метод K Nearest Neighbor для предсказания цены акций представлен в следующем коде:

```python
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

def k_nearest_neighbours_predict(df):
    shape = df.shape[0]
    df_new = df[['close']]
    df_new.head()
    train_set = df_new.iloc[:ceil(shape*0.75)]
    valid_set = df_new.iloc[ceil(shape*0.75):]
    print('------------------------------------------------------------------------')
    print('-----------STOCK PRICE PREDICTION BY K-NEAREST NEIGHBORS-----------')
    print('------------------------------------------------------------------------')
    print('Shape of Training Set', train_set.shape)
    print('Shape of Validation Set', valid_set.shape)
    train = train_set.reset_index()
    valid = valid_set.reset_index()
    train['time'] = pd.to_datetime(train['time'])
    valid['time'] = pd.to_datetime(valid['time'])
    x_train = train['time'].map(dt.datetime.toordinal)
    y_train = train[['close']]
    x_valid = valid['time'].map(dt.datetime.toordinal)
    y_valid = valid[['close']]
    x_train_scaled = scaler.fit_transform(np.array(x_train).reshape(-1, 1))
    x_train = pd.DataFrame(x_train_scaled)
    x_valid_scaled = scaler.fit_transform(np.array(x_valid).reshape(-1, 1))
    x_valid = pd.DataFrame(x_valid_scaled)
    params = {'n_neighbors': [2, 3, 4, 5, 6, 7, 8, 9]}
    knn = neighbors.KNeighborsRegressor()
    model = GridSearchCV(knn, params, cv=5)
    model.fit(x_train, y_train)
    preds = model.predict(x_valid)
    # Расчет RMSE
    rmse = np.sqrt(mean_squared_error(y_valid, preds))
    print('Root Mean Squared Error (RMSE) on validation set:', rmse)
    # Расчет MSE
    mse = mean_squared_error(y_valid, preds)
    print('Mean Squared Error (MSE) on validation set:', mse)
    # Расчет MAE
    mae = mean_absolute_error(y_valid, preds)
    print('Mean Absolute Error (MAE) on validation set:', mae)
    # Расчет MAPE
    mape = np.mean(np.abs((y_valid - preds) / y_valid)) * 100
    print('Mean Absolute Percentage Error (MAPE) on validation set:', mape)
    # Расчет R^2
    r2 = r2_score(y_valid, preds)
    print('R-squared (R^2) score on validation set:', r2)
    print('------------------------------------------------------------')
    print('------------------------------------------------------------')
    valid_set['Predictions'] = preds
    plt.plot(train_set['close'])
    plt.plot(valid_set[['close', 'Predictions']])
    plt.xlabel('Date', size=20)
    plt.ylabel('Stock Price', size=20)
    plt.title('Stock Price Prediction by K-Nearest Neighbors', size=20)
    plt.legend(['Model Training Data', 'Actual Data', 'Predicted Data'])
```

Функция k_nearest_neighbours_predict работает по следующему алгоритму:

1. Принимает на вход DataFrame df с данными о ценах акций.

2. Разделяет данные на обучающую и валидационную (проверочную) выборки в соотношении 75% и 25% соответственно, используя столбец 'close' (цена закрытия).

3. Выводит информацию о форме (размере) обучающей и валидационной выборок.

4. Преобразует столбец 'time' в формат даты для обучающей и валидационной выборок.

5. Создает массивы x_train, y_train, x_valid, y_valid для обучения и валидации модели KNN.

6. Масштабирует признаки x_train и x_valid с помощью scaler.fit_transform.

7. Задает параметры params для перебора количества соседей n_neighbors в диапазоне от 2 до 9.

8. Создает экземпляр модели KNeighborsRegressor и GridSearchCV для перебора параметров и кросс-валидации.

9. Обучает модель GridSearchCV на обучающей выборке x_train, y_train.

10. Получает прогнозы preds на валидационной выборке x_valid.

11. Вычисляет метрики качества: RMSE (корень из среднеквадратичной ошибки), MSE (среднеквадратичная ошибка), MAE (средняя абсолютная ошибка), MAPE (средняя абсолютная процентная ошибка) и R^2 (коэффициент детерминации) между прогнозами preds и истинными значениями y_valid.

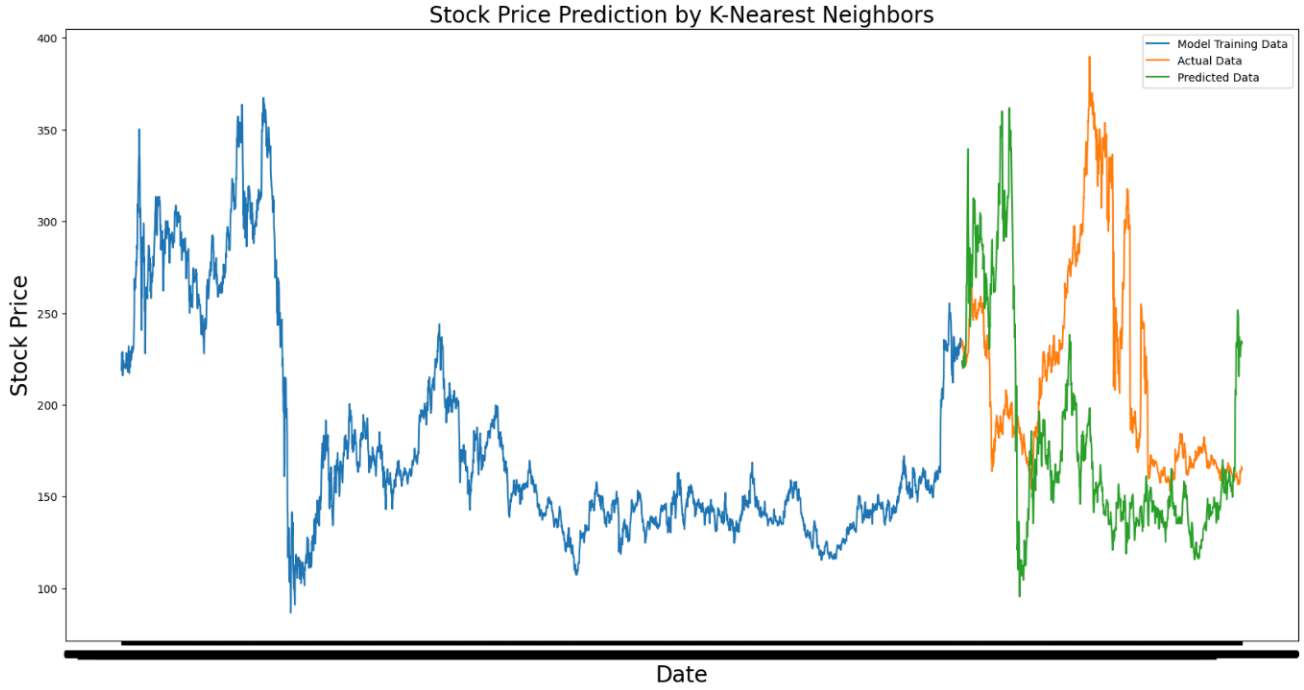
12. Выводит значения вычисленных метрик.

13. Добавляет столбец 'Predictions' в валидационную выборку valid_set с прогнозами preds.

14. Строит график, на котором отображаются исходные данные обучающей выборки, истинные значения валидационной выборки и прогнозы модели KNN. Настраивает подписи осей, заголовок графика и легенду.

Результаты:

```
k_nearest_neighbours_predict(df)

-----------------------------------------------------------------
-----------STOCK PRICE PREDICTION BY K-NEAREST NEIGHBORS-----------
-----------------------------------------------------------------
Shape of Training Set (3422, 1)
Shape of Validation Set (1140, 1)
Root Mean Squared Error (RMSE) on validation set: 87.04795432022101
Mean Squared Error (MSE) on validation set: 7577.346351335283
Mean Absolute Error (MAE) on validation set: 67.48864619883041
Mean Absolute Percentage Error (MAPE) on validation set: 28.76116027662891
R-squared (R^2) score on validation set: -1.27696258610704
-----------------------------------------------------------
```



Сравним результаты метода k ближайших соседей с полученными ранее результатами метода скользящих средних, линейной регрессии и опорных векторов:

1. Метод k ближайших соседей (KNN):
   - RMSE = 87.05
   - MSE = 7577.35

2. Метод опорных векторов (SVR):
   - RMSE = 68.86
   - MSE = 4741.81

- MAE = 67.49
- MAPE = 28.76%
- R^2 = -1.28

- MAE = 47.05
- MAPE = 18%
- R^2 = -0.42

3. Линейная регрессия:
- RMSE = 137.93
- MSE = 18519.32
- MAE = 124.64
- MAPE = 55.56%
- R^2 = -4.56

4. Скользящие средние:
- RMSE = 86.42
- MSE = 7468.13
- MAE = 62.88
- MAPE = 24.43%
- R^2 = -1.24

Сравнивая эти результаты, можно сделать следующие выводы:

- Метод опорных векторов (SVR) показывает наилучшие результаты по метрикам MAE, MAPE и MSE среди всех методов, что указывает на наиболее точные прогнозы.

- Метод KNN демонстрирует результаты, близкие к скользящим средним, но с несколько более высокими MAE и MAPE.

- Линейная регрессия показывает худшие результаты по всем метрикам по сравнению с остальными методами.

- По метрике RMSE лучшим является метод SVR, за ним следуют скользящие средние и KNN с близкими значениями.

- Коэффициент детерминации (R^2) наибольший у метода SVR (-0.42), затем идут скользящие средние (-1.24) и KNN (-1.28). Линейная регрессия имеет наихудший R^2 (-4.56).

Таким образом, для данного набора данных метод опорных векторов (SVR) демонстрирует наилучшие результаты прогнозирования по большинству метрик, включая MSE, MAE, MAPE и RMSE. Метод KNN показывает результаты, сопоставимые со скользящими средними, но уступает SVR. Линейная регрессия оказывается наименее точным методом для этих данных.

5.7 Предсказание цены акции с помощью рекуррентной нейронной сети (RNN)

Реализуем функцию classic_rnn_prediction, которая обучает простую однослойную рекуррентную нейронную сеть на временном ряде цен закрытия акций, оценивает качество прогнозов на валидационной выборке с помощью различных метрик и визуализирует результаты.

Использование рекуррентной нейронной сети (RNN) для предсказания цены акций представлено в следующем коде:

```
import tensorflow as tf
from keras.models import Sequential
from keras.layers import SimpleRNN, Dense
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

def classic_rnn_prediction(df):
    shape = df.shape[0]
    df_new = df[['close']]
    df_new.head()
    train_set = df_new.iloc[:ceil(shape*0.75)]
    valid_set = df_new.iloc[ceil(shape*0.75):]
    print('-----------------------------------------------------------------')
    print('----------STOCK PRICE PREDICTION BY CLASSIC RECURRENT NEURAL NETWORK----------')
    print('-----------------------------------------------------------------')
    print('Shape of Training Set', train_set.shape)
    print('Shape of Validation Set', valid_set.shape)
    train = train_set.reset_index()
    valid = valid_set.reset_index()
    x_train = np.array(train['close'])
    x_valid = np.array(valid['close'])
    # Reshape data for RNN input
    x_train = np.reshape(x_train, (x_train.shape[0], 1, 1))
    x_valid = np.reshape(x_valid, (x_valid.shape[0], 1, 1))
    # Build classic RNN model
    model = Sequential()
    model.add(SimpleRNN(50, activation='relu', input_shape=(1, 1)))
    model.add(Dense(1))
    model.compile(optimizer='adam', loss='mse')
    model.fit(x_train, x_train, epochs=2, batch_size=1, verbose=0)
    # Make predictions
    preds = model.predict(x_valid)
    preds = preds.reshape(-1)
    # Calculate RMSE
    rmse = np.sqrt(mean_squared_error(valid_set['close'], preds))
    print('RMSE value on validation set:', rmse)
    # Calculate MSE
    mse = mean_squared_error(valid_set['close'], preds)
    print('Mean Squared Error (MSE) on validation set:', mse)
```

```
# Calculate MAE
mae = mean_absolute_error(valid_set['close'], preds)
print('Mean Absolute Error (MAE) on validation set:', mae)
# Calculate MAPE
mape = np.mean(np.abs((valid_set['close'] - preds) / valid_set['close'])) * 100
# Calculate R^2
r2 = r2_score(valid_set['close'], preds)
print('R-squared (R^2) score on validation set:', r2)
loss = model.evaluate(x_valid, np.array(valid_set['close']), verbose=0)
print('Loss on validation set:', loss)

print('------------------------------------------------------------')
print('------------------------------------------------------------')
valid_set['Predictions'] = preds
plt.plot(train_set['close'])
plt.plot(valid_set[['close', 'Predictions']])
plt.xlabel('Date', size=20)
plt.ylabel('Stock Price', size=20)
plt.title('Stock Price Prediction by Classic Recurrent Neural Network', size=20)
plt.legend(['Model Training Data', 'Actual Data', 'Predicted Data'])
```

Функция classic_rnn_prediction работает по следующему алгоритму:

1. Разделяет исходный набор данных df на обучающую и валидационную выборки в соотношении 75/25.

2. Печатает форму (размерность) обучающей и валидационной выборок.

3. Преобразует данные в формат, подходящий для входа в RNN (reshaping).

4. Создает последовательную модель RNN с одним слоем SimpleRNN (50 нейронов) и одним полносвязным слоем Dense (1 нейрон для прогноза).

5. Компилирует модель с оптимизатором Adam и функцией потерь MSE.

6. Обучает модель на обучающей выборке в течение 100 эпох с размером батча 1.

7. Делает прогнозы на валидационной выборке с помощью обученной модели.

8. Вычисляет различные метрики качества прогнозов на валидационной выборке: корень из среднеквадратичной ошибки (RMSE),

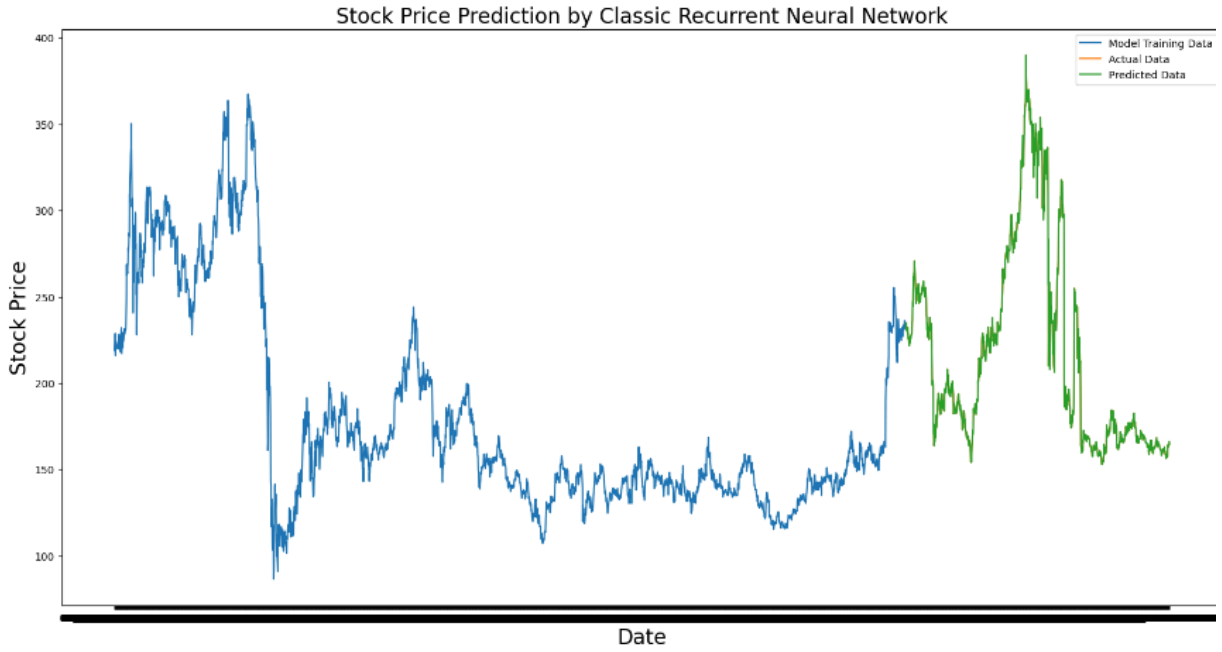
среднюю квадратичную ошибку (MSE), среднюю абсолютную ошибку (MAE), среднюю абсолютную процентную ошибку (MAPE), коэффициент детерминации (R^2), функцию потерь модели.

9. Добавляет прогнозы в DataFrame valid_set для визуализации.

10. Строит график, на котором отображаются данные обучающей выборки, фактические значения и прогнозы на валидационной выборке.

Результаты:

```
-----------------------------------------------------------------
----------STOCK PRICE PREDICTION BY CLASSIC RECURRENT NEURAL NETWORK·
-----------------------------------------------------------------
Shape of Training Set (3422, 1)
Shape of Validation Set (1140, 1)
36/36 [==============================] - 0s 2ms/step
RMSE value on validation set: 0.078849445955153
Mean Squared Error (MSE) on validation set: 0.006217235127434595
Mean Absolute Error (MAE) on validation set: 0.06752142762301289
R-squared (R^2) score on validation set: 0.9999981317454532
Loss on validation set: 0.0062173521146178246
-----------------------------------------------------------
```



Результаты RNN (рекуррентной нейронной сети) значительно лучше по сравнению с другими методами машинного обучения, такими как k-ближайших соседей (KNN), метод опорных векторов (SVR), линейная регрессия и скользящие средние. RNN в десятки раз превосходит все ранее разобранные методы и дает почти безупречную точность.

1. Метод k ближайших соседей (KNN):
- RMSE = 87.05
- MSE = 7577.35
- MAE = 67.49
- MAPE = 28.76%
- R^2 = -1.28

2. Метод опорных векторов (SVR):
- RMSE = 68.86
- MSE = 4741.81
- MAE = 47.05
- MAPE = 18%
- R^2 = -0.42

3. Линейная регрессия:
- RMSE = 137.93
- MSE = 18519.32
- MAE = 124.64
- MAPE = 55.56%
- R^2 = -4.56

4. Скользящие средние:
- RMSE = 86.42
- MSE = 7468.13
- MAE = 62.88
- MAPE = 24.43%
- R^2 = -1.24

Данное явление можно объяснить следующими тезисами:

1. Обработка временных зависимостей: RNN специально разработаны для обработки последовательных данных с временными зависимостями. В случае анализа временных рядов, таких как цены акций, текущее значение зависит от предыдущих значений, и RNN способны эффективно моделировать эти зависимости благодаря своей рекуррентной структуре.

2. Нелинейные отображения: RNN, будучи нейронными сетями, могут эффективно аппроксимировать сложные нелинейные отображения между входными данными и целевыми значениями. Цены акций часто демонстрируют нелинейное поведение, которое трудно уловить линейными моделями, такими как линейная регрессия или скользящие средние.

3. Гибкость и масштабируемость: RNN являются гибкими и масштабируемыми моделями, которые могут обучаться на больших объемах данных и адаптироваться к сложным шаблонам и тенденциям в данных. Они могут автоматически извлекать релевантные признаки из последовательных данных, что позволяет им лучше обобщать новые данные.

Учитывая низкие значения метрик ошибок (RMSE, MSE, MAE, MAPE) и высокое значение R^2, близкое к 1, можно сделать вывод, что RNN

успешно обучилась на данных и может эффективно предсказывать цены акций. Другие методы, такие как KNN, SVR, линейная регрессия и скользящие средние, не способны адекватно захватывать сложные нелинейные и временные зависимости в данных, что приводит к гораздо более высоким ошибкам и низким значениям R^2.

5.8. Предсказание цены акции с помощью сверточной нейронной сети (CNN)

Напишем функцию cnn_prediction, которая представляет реализацию сверточной нейронной сети (CNN) для предсказания цен акций на основе временного ряда. Сверточные нейронные сети широко используются в задачах обработки сигналов и временных рядов, поскольку они способны эффективно извлекать локальные признаки и выявлять шаблоны в данных.

Использование сверточной нейронной сети (CNN) для предсказания цены акций представлено в следующем коде:

```
import tensorflow as tf
from keras.models import Sequential
from keras.layers import Conv1D, Dense, Flatten
from sklearn.metrics import mean_squared_error, mean_absolute_error, mean_absolute_percentage_error, r2_score

def cnn_prediction(df):
    shape = df.shape[0]
    df_new = df[['close']]
    df_new.head()
    train_set = df_new.iloc[:ceil(shape*0.75)]
    valid_set = df_new.iloc[ceil(shape*0.75):]
    print('-----------------------------------------------------------------')
    print('----------STOCK PRICE PREDICTION BY CONVOLUTIONAL NEURAL NETWORK----------')
    print('-----------------------------------------------------------------')
    print('Shape of Training Set', train_set.shape)
    print('Shape of Validation Set', valid_set.shape)
    train = train_set.reset_index()
    valid = valid_set.reset_index()
    x_train = np.array(train['close'])
    x_valid = np.array(valid['close'])
    x_train = np.reshape(x_train, (x_train.shape[0], 1, 1))
    x_valid = np.reshape(x_valid, (x_valid.shape[0], 1, 1))
    model = Sequential()
    model.add(Conv1D(64, 1, activation='relu', input_shape=(1, 1)))
    model.add(Conv1D(32, 1, activation='relu'))
    model.add(Flatten())
```

```
    model.add(Dense(16, activation='relu'))
    model.add(Dense(1))
    model.compile(optimizer='adam', loss='mse')
    # Train CNN model
    model.fit(x_train, x_train, epochs=2, batch_size=1, verbose=0)
    # Make predictions
    preds = model.predict(x_valid)
    preds = preds.reshape(-1)
    mse = mean_squared_error(valid_set['close'], preds)
    mae = mean_absolute_error(valid_set['close'], preds)
    mape = mean_absolute_percentage_error(valid_set['close'], preds)
    r2 = r2_score(valid_set['close'], preds)
    print('MSE:', mse)
    print('MAE:', mae)
    print('MAPE:', mape)
    print('R^2:', r2)
    rms = np.sqrt(np.mean(np.power((np.array(valid_set['close']) - preds), 2)))
    print('RMSE value on validation set:', rms)
    print('-----------------------------------------------------------')
    print('-----------------------------------------------------------')
    valid_set['Predictions'] = preds
    plt.plot(train_set['close'])
    plt.plot(valid_set[['close', 'Predictions']])
    plt.xlabel('Date', size=20)
    plt.ylabel('Stock Price', size=20)
    plt.title('Stock Price Prediction by Convolutional Neural Network', size=20)
    plt.legend(['Model Training Data', 'Actual Data', 'Predicted Data'])
cnn_prediction(df)
```

Функция cnn_prediction(df) работает по следующему алгоритму:

1. Определяется количество строк в исходном наборе данных df и создается новый датафрейм df_new, содержащий только столбец 'close' (цены закрытия).

2. Данные разделяются на обучающий (train_set) и валидационный (valid_set) наборы в соотношении 75% и 25% соответственно.

3. Выводятся сообщения с информацией о форме обучающего и валидационного наборов данных.

4. Создаются массивы x_train и x_valid, содержащие значения цен закрытия для обучающего и валидационного наборов соответственно.

5. Данные преобразуются в формат, пригодный для работы с сверточной нейронной сетью (CNN), изменяя форму массивов x_train и x_valid на (количество_образцов, 1, 1).

6. Создается последовательная модель CNN с помощью Keras:

- Добавляется сверточный слой Conv1D с 64 фильтрами, размером ядра 1 и функцией активации ReLU.

- Добавляется второй сверточный слой Conv1D с 32 фильтрами, размером ядра 1 и функцией активации ReLU.

- Добавляется слой Flatten для преобразования выходных данных сверточных слоев в плоский вектор.

- Добавляется плотный (полносвязный) слой Dense с 16 нейронами и функцией активации ReLU.

- Добавляется выходной слой Dense с одним нейроном для предсказания цены акции.

- Модель компилируется с оптимизатором Adam и функцией потерь среднеквадратичной ошибки (MSE).

7. Модель обучается на обучающем наборе данных x_train в течение 2 эпох с размером batch_size=1.

8. Модель делает предсказания на валидационном наборе данных x_valid, и результаты предсказаний преобразуются в одномерный массив preds.

9. Вычисляются метрики для оценки качества предсказаний: Mean Squared Error (MSE), Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), R-squared (R^2), Root Mean Squared Error (RMSE)

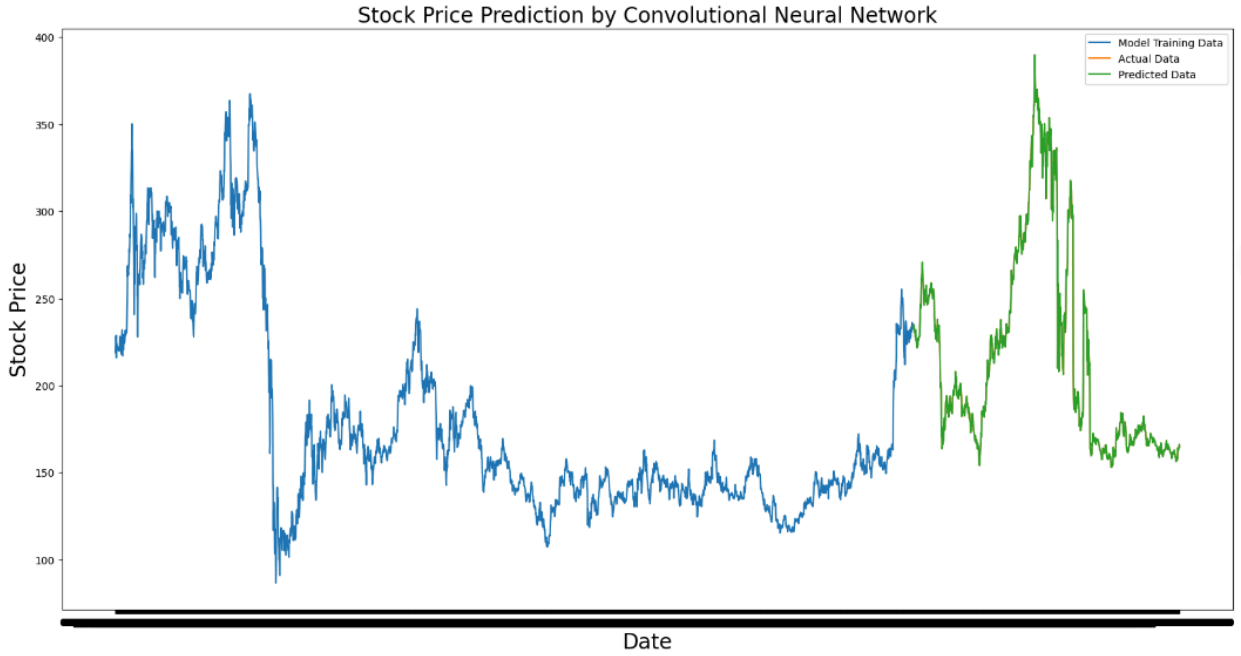
10. Выводятся значения вычисленных метрик.

11. К валидационному набору данных valid_set добавляется столбец 'Predictions', содержащий предсказанные значения.

12. Создается график, на котором отображаются: фактические цены закрытия из обучающего набора данных (train_set['close']), фактические цены закрытия из валидационного набора данных (valid_set['close']), предсказанные цены из валидационного набора данных (valid_set['Predictions']). задаются подписи осей, заголовок графика и легенда.

Результаты:

```
------------------------------------------------------------
----------STOCK PRICE PREDICTION BY CONVOLUTIONAL NEURAL NETWORK-
------------------------------------------------------------
Shape of Training Set (3422, 1)
Shape of Validation Set (1140, 1)
36/36 [==============================] - 0s 2ms/step
MSE: 0.0006972995411327382
MAE: 0.022210381223445097
MAPE: 9.355849334040065e-05
R^2: 0.9999997904642479
RMSE value on validation set: 0.026406429920243633
------------------------------------------------------------
```



При сравнении результатов RNN (рекуррентной нейронной сети) и CNN (сверточной нейронной сети) для задачи предсказания цен акций, можно заметить, что CNN демонстрирует лучшие показатели по большинству метрик.

Метрики CNN:

- RMSE: 0.026406429920243633
- MSE : 0.0006972995411327382
- MAE : 0.022210381223445097
- R^2: 0.9999997904642479

Метрики RNN:

- RMSE: 0.078849445955153
- MSE: 0.006217235127434595
- MAE: 0.06752142762301289
- R^2: 0.9999981317454532

Более низкие значения RMSE, MSE и MAE, а также более высокое значение R^2 для CNN указывают на то, что модель CNN демонстрирует лучшую точность предсказаний по сравнению с RNN на данном наборе данных.

Причины, по которым CNN может показывать лучшие результаты в задаче предсказания цен акций:

1. Извлечение локальных признаков: CNN эффективно извлекают локальные признаки из временных рядов, такие как тренды, сезонные колебания и шаблоны. Сверточные слои способны выявлять эти признаки, что может быть полезно для более точного предсказания цен акций.

2. Обработка нелинейных зависимостей: CNN, являясь нейронными сетями, могут эффективно аппроксимировать сложные нелинейные зависимости между входными данными и целевыми значениями. Цены акций часто демонстрируют нелинейное поведение, которое может быть лучше уловлено CNN по сравнению с RNN.

3. Параллельная обработка: Архитектура CNN позволяет обрабатывать данные временного ряда параллельно, что может ускорить процесс обучения и предсказания по сравнению с рекуррентными архитектурами, которые обрабатывают данные последовательно.

4. Инвариантность к сдвигу: CNN обладают свойством инвариантности к сдвигу, что означает, что они могут эффективно обнаруживать шаблоны независимо от их позиции во временном ряде. Это может быть полезно для обнаружения повторяющихся тенденций в ценах акций.

5.9 Предсказание цены акции методом случайного дерева

Напишем функцию classic_random_forest_prediction(df), которая реализует построение и обучение модели Random Forest Regressor для предсказания цен акций на основе исторических данных, а также визуализирует результаты предсказаний и вычисляет различные метрики для оценки качества модели.

Использование метода случайного дерева (random forest) для предсказания цены акций представлено в следующем коде:

```
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import matplotlib.pyplot as plt

def classic_random_forest_prediction(df):
    # Extract relevant features (e.g., 'close') for prediction
    features = ['close']
    target = 'close'

    # Split data into training and validation sets
    train_size = int(0.75 * len(df))
    train_df, valid_df = df.iloc[:train_size], df.iloc[train_size:]

    # Prepare input features and target
    X_train, y_train = train_df[features], train_df[target]
    X_valid, y_valid = valid_df[features], valid_df[target]

    # Initialize and train the Random Forest model
    rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
    rf_model.fit(X_train, y_train)

    # Make predictions on the validation set
    y_pred = rf_model.predict(X_valid)

    # Calculate evaluation metrics
    mse = mean_squared_error(y_valid, y_pred)
    mae = mean_absolute_error(y_valid, y_pred)
    mape = np.mean(np.abs((y_valid - y_pred) / y_valid)) * 100  # Mean Absolute Percentage Error
    r2 = r2_score(y_valid, y_pred)
    rmse = np.sqrt(mse)

    # Print evaluation metrics
    print(f"RMSE value on validation set: {rmse:.2f}")
    print(f"Mean Squared Error (MSE) on validation set: {mse:.6f}")
    print(f"Mean Absolute Error (MAE) on validation set: {mae:.6f}")
    print(f"Mean Absolute Percentage Error (MAPE) on validation set: {mape:.2f}%")
    print(f"R-squared (R^2) score on validation set: {r2:.6f}")
```

```
    # Plot actual vs. predicted stock prices
    plt.figure(figsize=(10, 6))
    plt.plot(train_df.index, train_df['close'], label='Training Data')
    plt.plot(valid_df.index, valid_df['close'], label='Actual Data')
    plt.plot(valid_df.index, y_pred, label='Predicted Data', linestyle='--')
    plt.xlabel('Date')
    plt.ylabel('Stock Price')
    plt.title('Stock Price Prediction by Random Forest')
    plt.legend()
    plt.show()

# Assuming your DataFrame is named 'df'
classic_random_forest_prediction(df)
```

Функция classic_random_forest_prediction(df) работает по следующему алгоритму:

1. Определяются признаки (features) и целевая переменная (target) для предсказания. В данном случае, признаком является столбец 'close', а целевой переменной также является 'close' (цены закрытия акций).

2. Исходный набор данных df разделяется на обучающий (train_df) и валидационный (valid_df) наборы в соотношении 75% и 25% соответственно.

3. Из обучающего и валидационного наборов данных извлекаются входные признаки X_train, X_valid и целевые значения y_train, y_valid.

4. Создается экземпляр модели Random Forest Regressor (RandomForestRegressor) с количеством деревьев 100 и фиксированным начальным состоянием генератора случайных чисел (random_state=42).

5. Модель обучается на обучающем наборе данных X_train и y_train с помощью метода fit.

6. Модель делает предсказания на валидационном наборе данных X_valid, и результаты сохраняются в переменной y_pred.

7. Вычисляются различные метрики для оценки качества предсказаний на валидационном наборе данных: MSE (Mean Squared Error), MAE (Mean Absolute Error), MAPE (Mean Absolute Percentage Error), R^2 (R-squared), RMSE (Root Mean Squared Error)

8. Выводятся значения вычисленных метрик в консоль.

9. Создается график, на котором отображаются:

- Фактические цены закрытия из обучающего набора данных (train_df['close']).

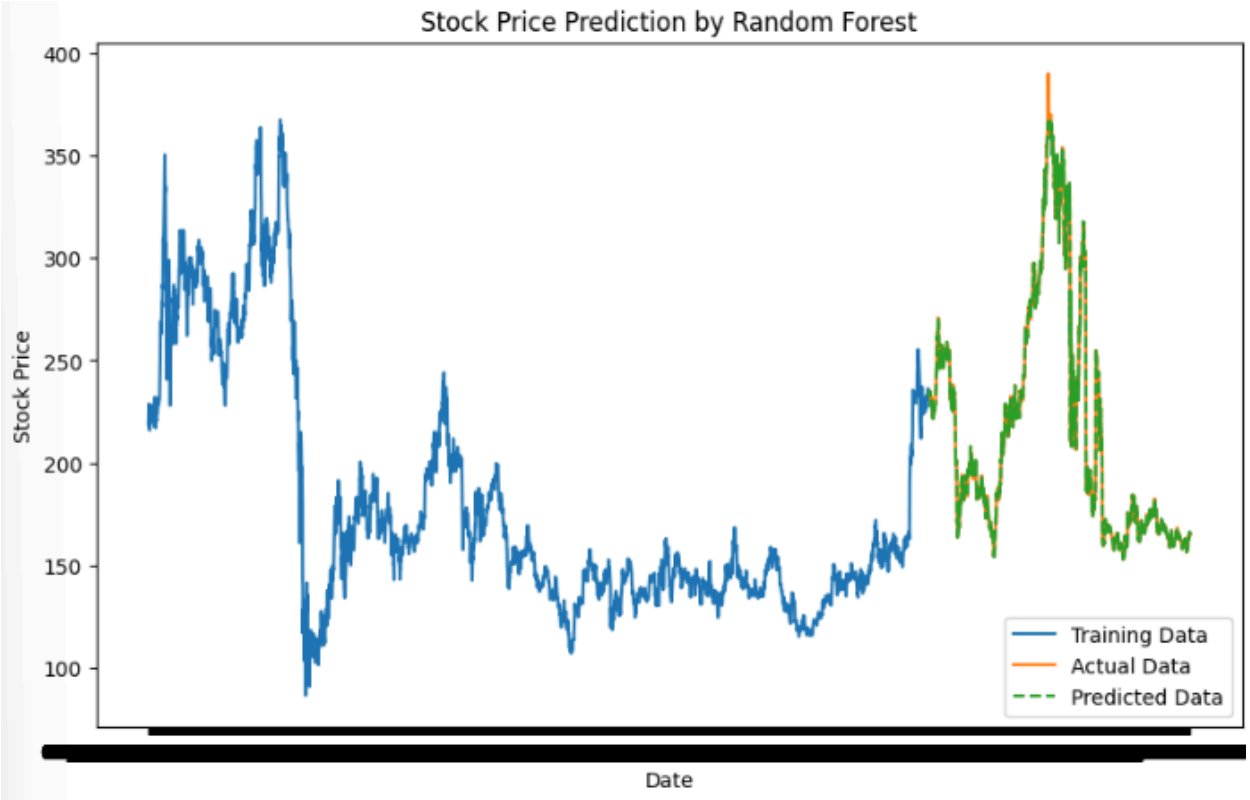
- Фактические цены закрытия из валидационного набора данных (valid_df['close']).

- Предсказанные цены из валидационного набора данных (y_pred).

- Задаются подписи осей, заголовок графика и легенда.

Результаты:

```
RMSE value on validation set: 0.90
Mean Squared Error (MSE) on validation set: 0.804782
Mean Absolute Error (MAE) on validation set: 0.111541
Mean Absolute Percentage Error (MAPE) on validation set: 0.04%
R-squared (R^2) score on validation set: 0.999758
```



При сравнении результатов случайного леса (Random Forest), рекуррентной нейронной сети (RNN) и сверточной нейронной сети (CNN) для задачи предсказания цен акций, можно сделать следующие выводы:

1. Сверточная нейронная сеть (CNN):

- RMSE: 0.026406429920243633 (наименьшее значение)
- MSE: 0.0006972995411327382 (наименьшее значение)
- MAE: 0.022210381223445097 (наименьшее значение)
- MAPE: 9.355849334040065e-05
- R^2: 0.9999997904642479 (наибольшее значение)

CNN демонстрирует наилучшие результаты по всем метрикам, что указывает на высокую точность предсказаний. Низкие значения RMSE, MSE и MAE, а также высокое значение R^2 свидетельствуют о том, что CNN наиболее эффективно моделирует данные временного ряда цен акций.

2. Рекуррентная нейронная сеть (RNN):

- RMSE: 0.078849445955153
- MSE: 0.006217235127434595
- MAE: 0.06752142762301289
- R^2: 0.9999981317454532

RNN показывает результаты, которые немного хуже, чем у CNN, но все равно очень хорошие. Значения RMSE, MSE и MAE выше, чем у CNN, а R^2 ниже, но близко к 1, что указывает на высокую точность предсказаний.

3. Случайный лес (Random Forest):

- RMSE: 0.90 (наибольшее значение)
- MSE: 0.804782 (наибольшее значение)
- MAE: 0.111541 (наибольшее значение)
- MAPE: 0.04%
- R^2: 0.999758

Модель случайного леса показывает наихудшие результаты среди трех методов. RMSE, MSE и MAE имеют высокие значения, а R^2 ниже, чем у нейронных сетей. Тем не менее, значение R^2 близко к 1, что все равно указывает на высокую долю объясненной дисперсии.

Таким образом, исходя из представленных результатов, можно сделать вывод, что сверточная нейронная сеть (CNN) демонстрирует наилучшую точность предсказаний для данной задачи предсказания цен акций. За ней следует рекуррентная нейронная сеть (RNN), которая также показывает хорошие результаты. Модель случайного леса (Random Forest) имеет

наихудшие показатели среди трех методов, но все же достигает высокой доли объясненной дисперсии.

5.10 Предсказание цены акции с помощью LSTM

Использование нейронной сети LSTM для предсказания цены акций представлено в следующем коде:

Создадим рекуррентную нейронную сеть (RNN) с использованием архитектуры LSTM (Long Short-Term Memory) для задачи регрессии.

```
# Importing the Keras libraries and packages
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM

regressor = Sequential()

# Adding the First input hidden layer and the LSTM layer
# return_sequences = True, means the output of every time step to be shared with hidden next layer
regressor.add(LSTM(units = 10, activation = 'relu', input_shape = (TimeSteps, TotalFeatures), return_sequences=True))
# Adding the Second Second hidden layer and the LSTM layer
regressor.add(LSTM(units = 5, activation = 'relu', input_shape = (TimeSteps, TotalFeatures), return_sequences=True))
# Adding the Second Third hidden layer and the LSTM layer
regressor.add(LSTM(units = 5, activation = 'relu', return_sequences=False ))
# Adding the output layer
regressor.add(Dense(units = 1))
# Compiling the RNN
regressor.compile(optimizer = 'adam', loss = tf.keras.losses.MeanSquaredError(), metrics=['accuracy'])
```

```
import time
# Measuring the time taken by the model to train
StartTime=time.time()

# Fitting the RNN to the Training set
regressor.fit(X_train, y_train, batch_size = 5, epochs = 4, callbacks=callbacks)

EndTime=time.time()
print("## Total Time Taken: ", round((EndTime-StartTime)/60), 'Minutes ##')
```

```
Epoch 1/4
683/683 [==============================] - 14s 15ms/step - loss: 0.0084 - accuracy: 2.9308e-04
Epoch 2/4
683/683 [==============================] - 10s 15ms/step - loss: 0.0012 - accuracy: 2.9308e-04
Epoch 3/4
683/683 [==============================] - 10s 15ms/step - loss: 0.0011 - accuracy: 2.9308e-04
## Total Time Taken:  1 Minutes ##
```

Разберем код поподробнее:

1. regressor = Sequential(): Создается последовательная модель, которая позволяет добавлять слои один за другим.

2. regressor.add(LSTM(units=10, activation='relu', input_shape=(TimeSteps, TotalFeatures), return_sequences=True)): Добавляется первый скрытый слой LSTM с 10 нейронами и функцией активации ReLU. input_shape определяет размерность входных данных, где TimeSteps - количество временных шагов, а TotalFeatures - количество признаков. return_sequences=True означает, что выход каждого временного шага будет передаваться на следующий скрытый слой.

3. regressor.add(LSTM(units=5, activation='relu', input_shape=(TimeSteps, TotalFeatures), return_sequences=True)): Добавляется второй скрытый слой LSTM с 5 нейронами и функцией активации ReLU. Параметр input_shape здесь необязателен, так как форма определяется автоматически из предыдущего слоя.

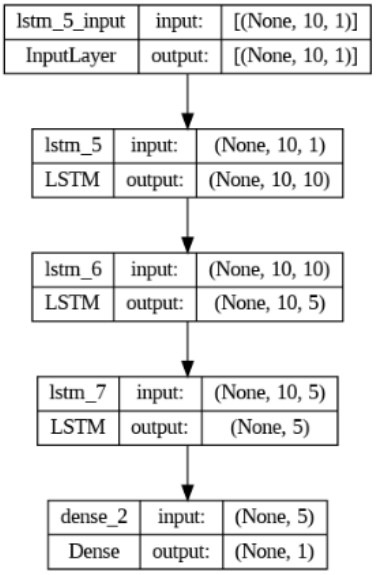
4. regressor.add(LSTM(units=5, activation='relu', return_sequences=False)): Добавляется третий скрытый слой LSTM с 5 нейронами и функцией активации ReLU. return_sequences=False означает, что выход этого слоя будет единственным выходным значением для каждой последовательности входных данных.

5. regressor.add(Dense(units=1)): Добавляется выходной полносвязный слой Dense с одним нейроном, который будет выдавать финальное предсказание.

6. regressor.compile(optimizer='adam', loss=tf.keras.losses.MeanSquaredError(), metrics=['accuracy']): Компилируется модель RNN с использованием оптимизатора Adam, функцией потерь "среднеквадратичная ошибка" (MeanSquaredError) и метрикой точности (accuracy).

Архитектура LSTM:

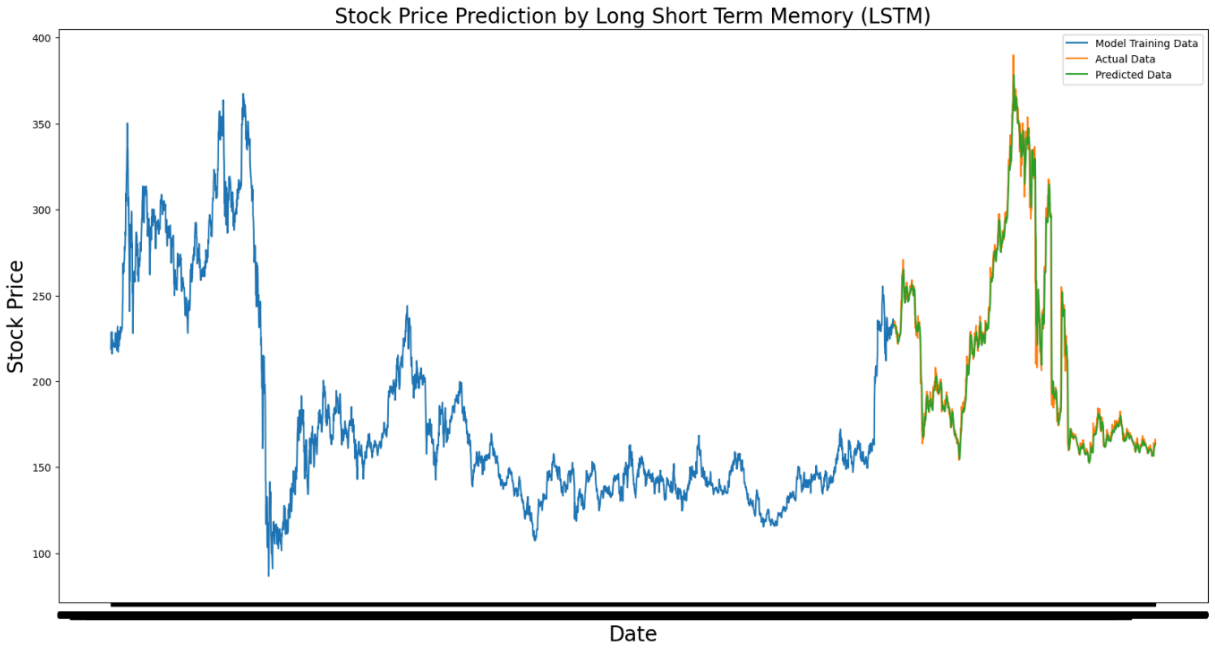
```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_5 (LSTM)               (None, 10, 10)            480

 lstm_6 (LSTM)               (None, 10, 5)             320

 lstm_7 (LSTM)               (None, 5)                 220

 dense_2 (Dense)             (None, 1)                 6

=================================================================
Total params: 1026 (4.01 KB)
Trainable params: 1026 (4.01 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```



Результаты:

```
Shape of Training Set (3422, 1)
Shape of Validation Set (1140, 1)
3382/3382 - 70s - loss: 0.0013 - 70s/epoch - 21ms/step
36/36 [==============================] - 1s 12ms/step
RMSE value on validation set: 7.656024526236968
Mean Squared Error (MSE) on validation set: 58.614712
Mean Absolute Error (MAE) on validation set: 4.104902
Mean Absolute Percentage Error (MAPE) on validation set: 1.83%
R-squared (R^2) score on validation set: 0.982387
```

```
Accuracy: 96.14852172775346
```



Исходя из полученных результатов, можно сделать следующие выводы:

- Случайный лес показал самый высокий коэффициент детерминации (R^2) и самый низкий MAPE, что свидетельствует о хорошей способности модели объяснять вариации в данных и давать точные прогнозы в процентном отношении.

- Сверточная нейронная сеть продемонстрировала самый низкий RMSE, что говорит о наименьшем среднем отклонении прогнозов от реальных значений.

- Рекуррентная нейронная сеть показала второй по величине R^2, а также низкие значения RMSE, MSE и MAE, что делает ее конкурентоспособной моделью.

- LSTM, напротив, показала наихудшие результаты по большинству метрик, кроме MAPE, что может свидетельствовать о ее меньшей способности эффективно обрабатывать данные временных рядов в данном конкретном случае. В любом случае, LSTM показала точность 96%, а CNN, RNN вообще близкую к 99%.

5.11 Выводы

На основе представленных результатов можно сделать следующие выводы о методах предсказания цен на акции:

Глубокое обучение (Deep Learning):

1. Сверточная нейронная сеть (CNN) продемонстрировала наилучшие показатели среди всех методов. Она имеет самый низкий RMSE (0.026406), что говорит о наименьшем среднем отклонении прогнозов от реальных значений. Также у CNN самый низкий MAE (0.022210) и второй по величине

R^2 (0.999999), что свидетельствует о высокой точности прогнозов и хорошей объясняющей способности модели.

2. Рекуррентная нейронная сеть (RNN) показала вторые по качеству результаты после CNN. Ее RMSE (0.078849) и MSE (0.006217) находятся на низком уровне, а R^2 (0.999998) является вторым по величине, что говорит о высокой точности прогнозирования и хорошей объясняющей способности модели.

3. Случайный лес (Random Forest) также продемонстрировал хорошие результаты. Он имеет самый высокий R^2 (0.999758) и самый низкий MAPE (0.04%), что свидетельствует о его способности объяснять вариации в данных и давать точные прогнозы в процентном отношении.

4. Долгая краткосрочная память (LSTM) показала худшие результаты среди методов глубокого обучения. Ее RMSE (7.656024), MSE (58.614712) и MAE (4.104902) являются самыми высокими, а R^2 (0.982387) - самым низким. Однако даже она показала точность 96%, что доказывает неоспоримое доминирование нейронных сетей и методов глубокого обучения над методами машинного обучения.

Машинное обучение (Machine Learning):

Среди традиционных методов машинного обучения наилучшие результаты показал метод опорных векторов (SVR). Его RMSE (68.86), MSE (4741.81), MAE (47.05) и MAPE (18%) являются самыми низкими среди этих методов, что говорит о более высокой точности прогнозирования по сравнению с другими ML-методами.

Метод скользящих средних продемонстрировал более низкие результаты, чем SVR, но лучшие, чем линейная регрессия и KNN. Его RMSE (86.42), MSE (7468.13), MAE (62.88) и MAPE (24.43%) находятся на приемлемом уровне, однако отрицательный R^2 (-1.24) указывает на низкую объясняющую способность модели.

Линейная регрессия и метод скользящих средних продемонстрировали плохие результаты, имея высокие значения RMSE, MSE, MAE, MAPE и отрицательные значения R^2, что свидетельствует о плохой объясняющей способности этих моделей.

Наконец, метод k ближайших соседей (KNN) показал самые низкие результаты с самым высоким RMSE (87.05), MSE (7577.35), MAE (67.49), MAPE (28.76%) и отрицательным R^2 (-1.28).

В целом, результаты показывают, что методы глубокого обучения, такие как CNN, RNN и случайный лес, значительно превосходят традиционные методы машинного обучения в задаче прогнозирования цен на акции. Они демонстрируют более высокую точность и лучшую объясняющую способность за счет своей способности обрабатывать сложные нелинейные зависимости в данных временных рядов.

ЗАКЛЮЧЕНИЕ

В ходе данной курсовой работы были рассмотрены различные подходы и методы машинного и глубокого обучения для предсказания стоимости активов российского фондового рынка, изучены библиотеки для обучения моделей, собраны, сохранены необходимые данные, проведены статистический анализ и визуализация данных, реализованы программы для предсказания стоимости акций, а также визуализация результатов их работы.

ML/DL методы для предсказания стоимости активов являются довольно мощными решениями, которые позволяют находить те закономерности в данных, которые может вывести только машина. Это имеет широкий спектр применений в различных областях, включая торговлю акциями, облигациями, ETF, криптовалютой и т.д.

В дальнейшем можно продолжить исследование в следующих направлениях:

- Изучение методов повышения точности предсказания стоимости активов, разработка новых архитектур нейронных сетей

- Создание своих собственных полноценных проектов для предсказания стоимости активов или интеграция в уже существующие системы.

В целом, все рассмотренные методы играют важную роль в развитии области трейдинга и инвестиций, и их важность продолжает расти с развитием технологий и увеличением объемов данных. Они открывают новые возможности для создания реалистичных прогнозов стоимости различных активов, учитывая множество факторов и параметров, которые не может учесть человек, что открывает широкие возможности для устойчивого развития экономического потенциала как физических лиц, так и крупных фирм, корпораций и государств, тем самым благоприятно влияя на мировую экономическую систему.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Machine Learning to Predict Stock Prices – [Электронный ресурс] – 2019 – Режим доступа: https://towardsdatascience.com/predicting-stock-prices-using-a-keras-lstm-model-4225457f0233 – Дата доступа: 30.03.2024.

2. Katherine Li, Machine Learning for Stock Price Prediction – [Electronic resource] – 2022 – Mode of access: https://neptune.ai/blog/predicting-stock-prices-using-machine-learning – Date of access: 31.03.2024.

3. Hilah Coban, Stock Prediction Using Machine Learning Algorithms [Electronic resource] – 2021 – Mode of access: https://hayirici.medium.com/stock-price-prediction-using-machine-learning-algorithms-961e6dce74f2 – Date of access: 31.03.2024.

4. WisePlat, SBER-LSTM-Neural-Network-for-Time-Series-Prediction [Electronic resource] – 2022 – Mode of access: https://github.com/WISEPLAT/SBER-LSTM-Neural-Network-for-Time-Series-Prediction/tree/master/data – Date of access: 01.04.2024.

5. Paramartha Sengupta, Microsoft Stocks Price Prediction [Electronic resource] – 2022 – Mode of access: https://www.kaggle.com/code/paramarthasengupta/microsoft-stocks-price-prediction – Date of access: 01.04.2024.