МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

по курсу

«Data Science»

Слушатель Диденко Ольга Владимировна

Москва, 2023

Содержание

Введение

Данная работа выполнена в рамках курса Data Science.

В качестве анализируемой задачи принята тема "Разработка модели прогнозирования цены акции на основе анализа финансовых данных и применения методов машинного обучения"

Для работы с такими данными разработано множество методов и технологий, позволяющих исследовать их, извлекать информацию и строить прогнозы на будущее.

Цель данной работы заключается в исследовании и анализе данных, связанных с динамикой цен на акции компаний, и построении моделей прогнозирования изменений цен на основе методов машинного обучения. Для достижения этой цели мы будем использовать данные об акциях ОАО «Газпром», которые содержат цену открытия, цену закрытия, максимальную и минимальную цену за день, а также объем торгов.

Данные берем с сайта Финам (https://www.finam.ru/profile/moex-akcii/gazprom/export/).

В работе будет представлена теоретическая часть, в которой будут рассмотрены методы анализа данных и обработки информации. Будет рассмотрено понятие машинного обучения, основные этапы и принципы его работы, а также нейронные сети как один из наиболее эффективных методов машинного обучения.

Далее, в практической части мы будем проводить анализ данных по динамике цен на акции компаний, используя методы машинного обучения, такие как линейная регрессия, деревья решений и нейронные сети. Мы будем строить модели прогнозирования изменений цен на акции на основе представленных данных, а также проведем визуализацию результатов на графиках.

Результатом данной работы будет являться сравнительный анализ эффективности методов машинного обучения для прогнозирования изменения цен на акции компаний, а также практические рекомендации по применению этих методов в реальной жизни.

1 Аналитическая часть

1.1 Методы анализа данных

Методы анализа данных представляют собой разнообразные техники и алгоритмы для обработки, интерпретации и извлечения значимых знаний из различных типов данных. Ниже перечислены некоторые из наиболее распространенных методов анализа данных:

1. Описательная статистика.

Описательная статистика — это метод анализа данных, который позволяет описывать набор данных с помощью различных статистических показателей. Эти показатели включают среднее значение, медиану, моду, стандартное отклонение и другие. С помощью описательной статистики можно определить основные характеристики набора данных, такие как среднее значение, дисперсия и коэффициент вариации.

2. Корреляционный анализ.

Корреляционный анализ — это метод, используемый для измерения силы и направления связи между двумя переменными. Корреляционный анализ позволяет определить, насколько сильно связаны две переменные, и насколько изменение одной переменной влияет на другую.

3. Регрессионный анализ.

Регрессионный анализ — это метод, используемый для моделирования зависимости между двумя или более переменными. Регрессионный анализ позволяет определить, как одна переменная влияет на другую и предсказать значения зависимой переменной на основе значений независимых переменных.

4. Анализ временных рядов.

Анализ временных рядов — это метод анализа данных, используемый для изучения изменений переменных во времени. Анализ временных рядов позволяет определить сезонные и трендовые изменения в данных и прогнозировать будущие значения переменных.

5. Кластерный анализ.

Кластерный анализ — это метод, используемый для группировки объектов на основе их сходства или различий. Кластерный анализ позволяет выделить группы объектов схожих характеристик и использовать эту информацию для классификации, прогнозирования или других целей.

6. Методы машинного обучения.

Методы машинного обучения — это набор алгоритмов, используемых для анализа и предсказания данных на основе обучения на обучающих данных.

1.2 Методы обработки данных

Методы обработки данных являются важной частью анализа данных и машинного обучения. Обработка данных включает в себя различные этапы, такие как предобработка данных, очистка данных, преобразование данных, сжатие данных и другие. В этой части будут рассмотрены основные методы обработки данных, используемые при анализе данных и машинном обучении.

Одним из первых шагов в обработке данных является предобработка данных. Этот шаг включает в себя удаление ненужных данных, заполнение пропущенных значений, преобразование данных в удобный формат и другие действия. Предобработка данных является важным этапом, так как некачественные данные могут привести к неправильным выводам.

Другим важным методом обработки данных является очистка данных. Очистка данных включает в себя удаление шума, выбросов и ошибок в данных. Этот шаг также важен, так как шум в данных может привести к ошибкам при анализе данных и машинном обучении.

Преобразование данных — это еще один важный метод обработки данных. Преобразование данных включает в себя изменение формата данных, изменение шкалы данных и другие действия. Например, при работе с текстовыми данными, преобразование может включать в себя удаление стоп-слов или лемматизацию слов.

Сжатие данных также является важным методом обработки данных. Сжатие данных может быть полезным при работе с большими объемами данных или при передаче данных через сеть. Различные методы сжатия данных, такие как сжатие без потерь и сжатие с потерями, могут быть использованы в зависимости от конкретной задачи.

Нормализация данных — это процесс приведения значений признаков к определенному диапазону или распределению. Часто используется для того, чтобы разные признаки имели одинаковый масштаб, что упрощает обучение модели.

Наиболее распространенными методами нормализации данных являются:

Минимакс нормализация (MinMax Scaling) — это метод, при котором данные приводятся к диапазону от 0 до 1 путем вычитания минимального значения признака из всех значений признака и деления на разницу между максимальным и минимальным значениями признака.

Z-нормализация (Standard Scaling) — это метод, при котором данные приводятся к стандартному нормальному распределению (среднее значение равно 0, стандартное отклонение равно 1) путем вычитания среднего значения признака из всех значений признака и деления на стандартное отклонение признака.

Выбросы — это значения, которые сильно отличаются от остальных значений в наборе данных. Они могут возникать из-за ошибок измерения, ошибок ввода данных или других причин. Выбросы могут искажать результаты анализа данных и влиять на работу модели.

Существует несколько методов обработки выбросов:

Удаление выбросов - самый простой способ, заключающийся в удалении всех строк данных, содержащих выбросы. Однако этот метод может привести к потере значимых данных.

Замена выбросов - можно заменить выбросы на определенное значение, например, на среднее или медианное значение признака.

Использование статистических методов - такие методы, как межквартильный размах и правило трех сигм, могут использоваться для определения выбросов и их удаления или замены.

Кодирование категориальных признаков — это процесс преобразования категориальных переменных в числовые значения, чтобы они могли быть использованы в моделях машинного обучения. Категориальные переменные могут быть определены как переменные, которые принимают ограниченное количество уникальных значений, например, цвет, пол или марка автомобиля.

Существуют различные методы кодирования категориальных признаков:

One-Hot Encoding: этот метод создает бинарные столбцы для каждой уникальной категории исходного признака. Если у нас есть категориальный признак "цвет", который может принимать значения "красный", "синий" и "зеленый", мы можем создать три новых признака "цвет_красный", "цвет_синий" и "цвет_зеленый". Если значение исходного признака "цвет" равно "красный", то значение "цвет_красный" будет равно 1, а значения "цвет_синий" и "цвет_зеленый" будут равны 0.

Label Encoding: этот метод присваивает каждой уникальной категории целочисленный код. Например, если у нас есть категориальный признак "цвет", который может принимать значения "красный", "синий" и "зеленый", мы можем присвоить "красному" код 1, "синему" - код 2, а "зеленому" - код 3.

Ordinal Encoding: это метод, который присваивает уникальным значениям категориального признака целочисленные коды, но при этом сохраняет порядок между значениями.

Кодирование категориальных признаков - важный этап предобработки данных, который может оказать значительное влияние на качество модели машинного обучения.

1.3 Методы машинного обучения

Методы машинного обучения позволяют автоматически обучать модели на основе имеющихся данных и использовать их для предсказания новых результатов. В данной работе рассмотрим несколько методов машинного обучения, таких как линейная регрессия, решающие деревья, случайный лес, градиентный бустинг и рекуррентные нейронные сети.

Линейная регрессия — это один из самых простых и широко используемых методов машинного обучения для решения задач регрессии. Линейная регрессия моделирует зависимость между независимыми переменными и зависимой переменной, которая представляет собой непрерывную величину. Модель линейной регрессии строится путем подгонки линейной функции к набору данных, чтобы минимизировать сумму квадратов расстояний между прогнозируемыми и реальными значениями.

Решающие деревья - это метод машинного обучения, который использует древовидную структуру для принятия решений. Древо состоит из узлов, которые представляют собой тесты на значения признаков, и листьев, которые представляют собой прогнозируемые значения. Решающие деревья могут быть использованы для решения задач как классификации, так и регрессии.

Случайный лес - это ансамбль решающих деревьев, который строит несколько деревьев на основе различных подмножеств признаков и объектов, а затем комбинирует их прогнозы для получения более точных результатов. Случайный лес обычно используется для решения задач классификации и регрессии.

Градиентный бустинг - это метод машинного обучения, который строит ансамбль моделей, добавляя их последовательно, при этом каждая новая модель обучается на ошибках предыдущих. Градиентный бустинг может использоваться

для решения задач классификации и регрессии, и часто дает лучшие результаты, чем случайный лес.

1.4 Нейронные сети

Нейронные сети — это класс алгоритмов машинного обучения, имитирующий работу человеческого мозга. Они состоят из нейронов, которые соединены между собой в виде графа, где каждый нейрон может передавать сигналы другим нейронам. Обучение нейронной сети заключается в том, чтобы настраивать веса соединений между нейронами, чтобы сеть могла производить нужный вывод на основе предоставленных ей данных.

Основные компоненты нейронной сети:

Входные данные: данные, которые поступают на вход нейронной сети для обработки.

Нейроны: компоненты нейронной сети, которые получают входные данные и производят выходные данные.

Соединения: механизм передачи информации между нейронами.

Веса: числовые значения, которые определяют важность соединений между нейронами.

Функции активации: функции, которые применяются к выходным данным нейронов, чтобы определить, должен ли нейрон активироваться и передавать свой выход другим нейронам.

Функция потерь: функция, которая измеряет, насколько точно нейронная сеть предсказывает желаемый вывод на основе входных данных.

Основные типы нейронных сетей:

Прямое распространение (feedforward) — это самый простой тип нейронных сетей, который передает сигналы от входных узлов к выходным узлам без циклов и обратных связей.

Рекуррентные (recurrent) — это тип нейронных сетей, которые имеют обратные связи между узлами, что позволяет им использовать информацию о предыдущих состояниях для предсказания будущих состояний.

Сверточные (convolutional) — это тип нейронных сетей, который специализируется на анализе изображений и других многомерных данных. Они используют фильтры для извлечения признаков из входных данных.

Глубокие (deep) — это тип нейронных сетей, который содержит несколько слоев нейронов для извлечения более высокоуровневых признаков из входных данных.

Обучение нейронной сети происходит путем оптимизации функции потерь с помощью алгоритма оптимизации. Один из наиболее распространенных методов оптимизации нейронных сетей называется обратное распространение ошибки (backpropagation). Он заключается в том, что сначала нейронная сеть делает предсказание на основе входных данных, затем функция потерь сравнивает предсказание с желаемым выводом, и вычисляется ошибка. Затем ошибка распространяется обратно по сети, и на основе этого вычисляются градиенты весов. Градиенты используются для обновления весов с целью уменьшения ошибки на следующей итерации обучения.

Кроме того, для предотвращения переобучения нейронной сети, можно использовать такие методы, как регуляризация, обрезание весов, исключение случайных узлов и дропаут.

Также существуют предобученные нейронные сети, которые обучены на больших наборах данных и сохранены в виде моделей. Эти модели могут быть использованы для решения задач в области компьютерного зрения, обработки естественного языка, распознавания речи и других областях.

2 Практическая часть

2.1 Предобработка данных

Для анализа возьмет данные из открытого источника, с сайта Финам (https://www.finam.ru/profile/moex-akcii/gazprom/export/), я сохранила данный датасет в свой репозиторий на Гитхаб.

Для начала, загрузим наш датасет и посмотрим на него:

```
# Загрузить датасет в DataFrame
df = pd.read_csv("https://raw.githubusercontent.com/OlgaDidenko/share-price/main/MOEX_200331_230331.csv")
# Вывести первые 5 строк датасета
print(df.head())
# Вывести основную информацию
print(df.info())
# Вывести описательную статистику датасета
print(df.describe())
```

Мы видим, что наш датасет содержит информацию о ценах акций на Московской бирже за период с 31 марта 2020 года по 31 марта 2023 года. В датасете есть следующие признаки:

DATE - дата торгов.

TICKER - идентификатор ценной бумаги.

OPEN - цена открытия торгов.

HIGH - максимальная цена торгов.

LOW - минимальная цена торгов.

CLOSE - цена закрытия торгов.

VOLUME - объем торгов.

Далее проверяем типы данных колонок: убедились, что типы данных для каждой колонки правильные, чтобы избежать ошибок при анализе данных.

```
df.drop(["<TICKER>", "<PER>", "<DATE>"], axis=1, inplace=True)
# удаляем ненужные столбцы
print(df.isnull().sum()) # проверяем наличие пропущенных значений
```

Если в датасете есть пропущенные значения, то их можно заменить средним значением или медианой для числовых колонок, или удалить строки с пропущенными значениями. В наших данных пропусков нет, переходим к следующему шагу.

Визуализация данных: строим графики для каждого столбца, чтобы оценить распределение значений и наличие выбросов (рисунок1):

```
for col in df.columns:
    plt.figure(figsize=(10,6))
    sns.histplot(df[col])
    plt.title(col)
    plt.show()
```

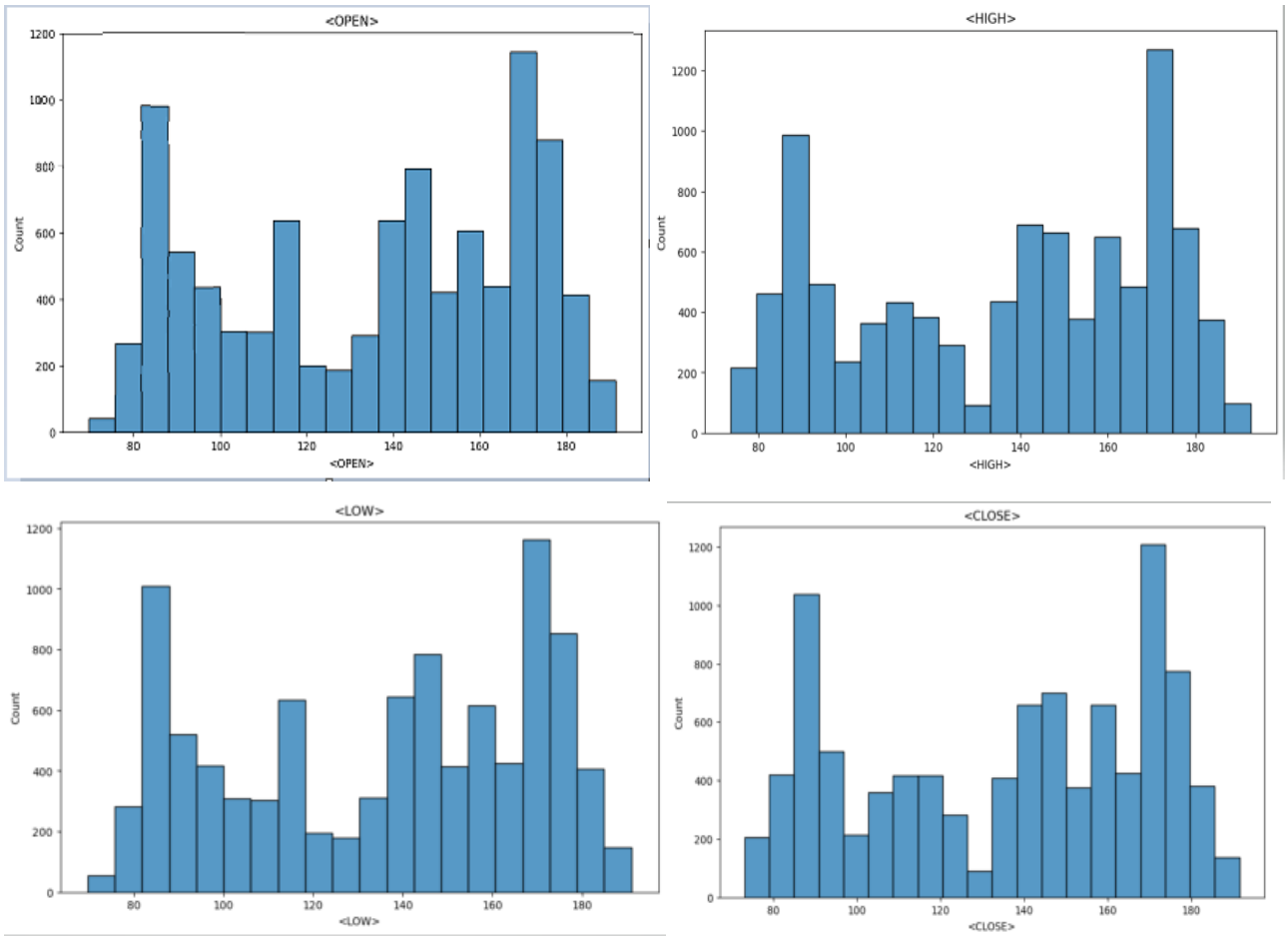


Рисунок 1

2.2 Разработка и обучение модели

После подготовки данных мы можем приступить к обучению моделей машинного обучения.

Для удобства переименуем колонки в более удобный вид (но этого можно не делать). И в данном случае, я еще ввела новую колонку PRICE_NEXT_DAY со значениями цены акций на следующий день. Удаляем последнюю строку, так как у нее нет значения 'PRICE_NEXT_DAY'.

Далее разделяем данные на обучающую и тестовую выборки, определяем признаки и целевую переменную. Можно было работать только с ценой закрытия, но я получила результаты лучше при данном варианте.

```
 # разделить данные на обучающую и тестовую выборки
train_size = int(len(df) * 0.8)
train_data, test_data = df[:train_size], df[train_size:]

# определить признаки и целевую переменную
X_train = train_data[['OPEN', 'HIGH', 'LOW', 'CLOSE', 'VOL']]
y_train = train_data['PRICE_NEXT_DAY']
X_test = test_data[['OPEN', 'HIGH', 'LOW', 'CLOSE', 'VOL']]
y_test = test_data['PRICE_NEXT_DAY']
Далее создаем и обучим модель линейной регрессии
lr = LinearRegression()
lr.fit(X_train, y_train)

# Прогнозирование цены акции
y_pred_lr = lr.predict(X_test)

# Оценка точности модели с помощью метрики RMSE
lr_rmse = np.sqrt(mean_squared_error(y_test, y_pred_lr))
print(f'RMSE Linear Regression: {lr_rmse}')
Создание и обучение модели решающих деревьев
```

```
dt = DecisionTreeRegressor(random_state=42)
dt.fit(X_train, y_train)

# Прогнозирование цены акции
y_pred_dt = dt.predict(X_test)

# Оценка точности модели с помощью метрики RMSE
dt_rmse = np.sqrt(mean_squared_error(y_test, y_pred_dt))
print(f'RMSE DecisionTreeRegressor: {dt_rmse}')
Создание модели случайного леса
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)

# Обучение модели на обучающей выборке
rf_model.fit(X_train, y_train)

# Прогнозирование на тестовой выборке
y_pred_rf = rf_model.predict(X_test)

# Оценка точности модели с помощью метрики RMSE
rf_rmse = np.sqrt(mean_squared_error(y_test, y_pred_rf))
print(f'RMSE RandomForestRegressor: {rf_rmse}')
Создание модели градиентного бустинга
gb_model = GradientBoostingRegressor(n_estimators=100, random_state=42)

# Обучение модели на обучающей выборке
gb_model.fit(X_train, y_train)

# Прогнозирование на тестовой выборке
y_pred_gb = gb_model.predict(X_test)
```

```
# Оценка точности модели с помощью метрики RMSE
gb_rmse = np.sqrt(mean_squared_error(y_test, y_pred_gb))
print(f'RMSE GradientBoostingRegressor: {gb_rmse}')
```

2.3 Сравнение моделей

Для сравнения моделей построим графики (рисунок 2) и посчитаем метрику RMSE.

RMSE (Root Mean Squared Error) - это метрика, которая используется для измерения разницы между фактическими значениями и прогнозируемыми значениями. Она вычисляется как квадратный корень из среднеквадратической ошибки (MSE), которая является средним значением квадрата отклонения между фактическим и прогнозируемым значениями.

Чем меньше значение RMSE, тем лучше производительность модели, так как это означает, что модель более точно прогнозирует значения на тестовом наборе данных.

```
import matplotlib.pyplot as plt
plt.figure(figsize=(12, 6))
plt.plot(y_test.values, label="true")
plt.plot(y_pred_lr, label="Linear Regression")
plt.legend()
plt.show()
```

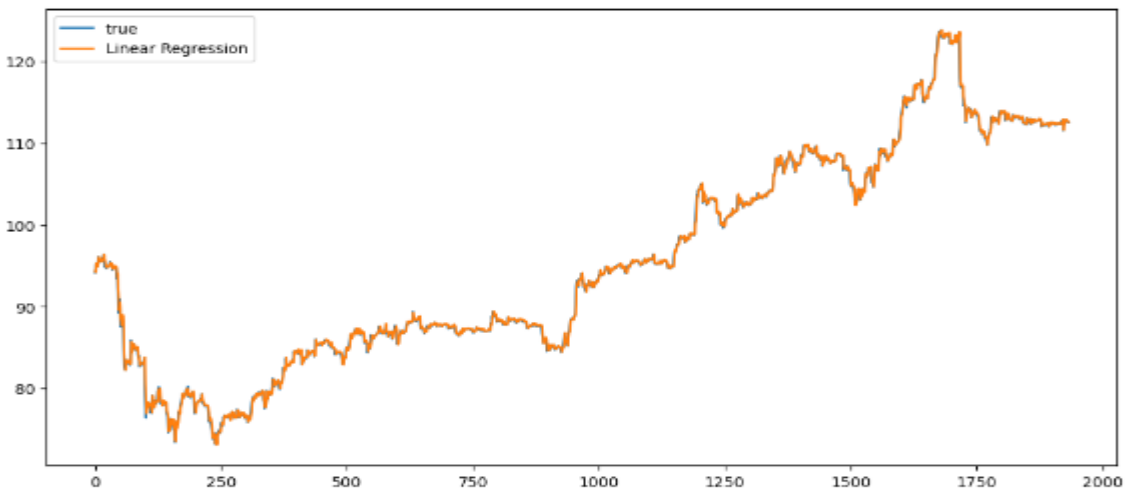


Рисунок 2.1

Аналогично построим визуализацию для остальных моделей.

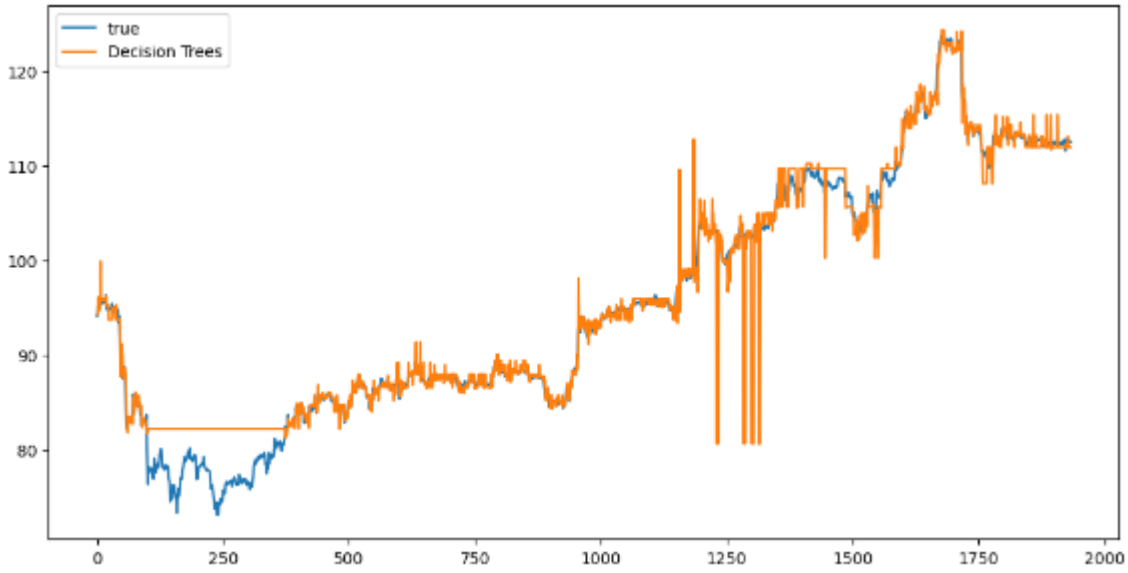


Рисунок 2.2

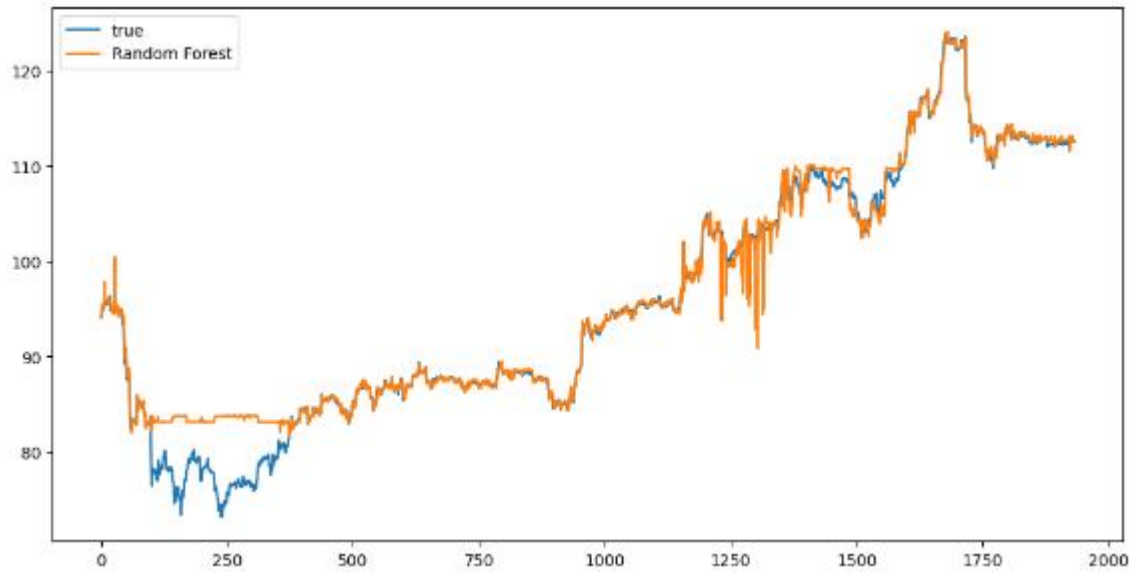


Рисунок 2.3

Для визуализации использовалась библиотека matplotlib. На графике отображены фактические значения цены на закрытие акций и прогнозные значения для каждой модели. Он также добавляет подписи осей и легенду, чтобы помочь понять, какой цвет соответствует какой модели.

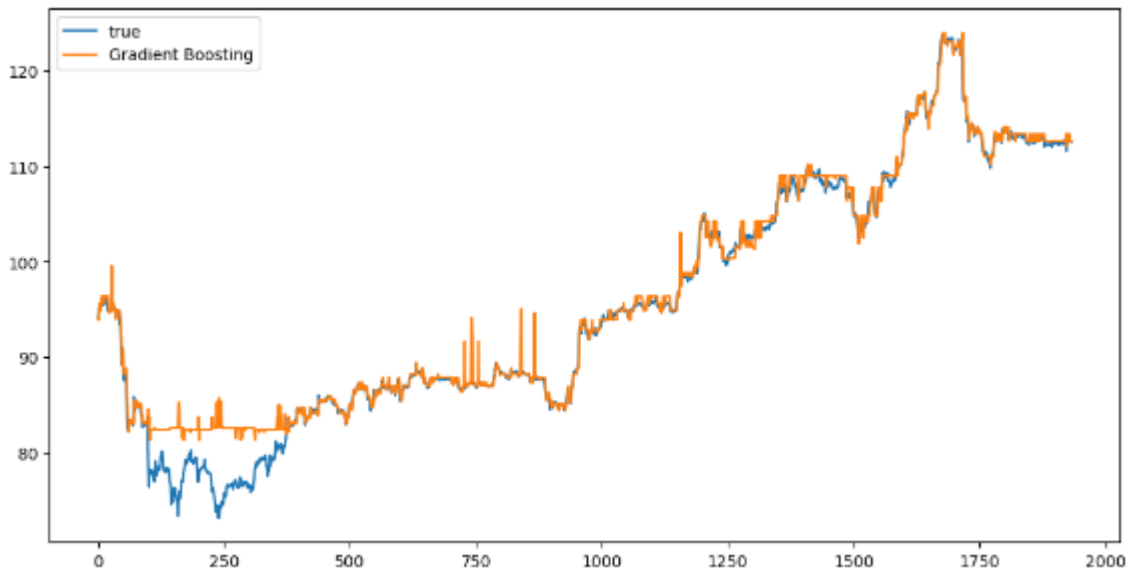


Рисунок 2.4

Модель линейной регрессии показала наивысшую точность среди всех моделей, с RMSE 0,44. Это означает, что она дала наименьшую среднюю ошибку прогнозирования на тестовой выборке. Тем не менее, при выборе модели для прогнозирования, помимо точности, необходимо учитывать и другие факторы, такие как скорость обучения, ресурсоемкость, устойчивость к выбросам и т.д.

С учетом всех этих факторов, модель Gradient Boosting могла бы оказаться более предпочтительной, если точность модели линейной регрессии была бы достаточной для задачи прогнозирования. Она показала второй лучший результат с RMSE 2,13, что значительно меньше, чем у моделей Decision Tree и Random Forest, которые показали RMSE 2,81 и 2,50 соответственно.

Таким образом, выбор наиболее подходящей модели зависит от конкретной задачи, ее целей и ограничений. Если точность - единственный критерий выбора, то модель линейной регрессии будет наилучшим выбором. Однако, если учитывать другие факторы, модель Gradient Boosting может быть предпочтительнее. Кроме того, это один из немногих случаев, в аналогичных работах, когда линейная регрессия оказалась точнее.

2.4 Нейронная сеть

Для нашей задачи мы будем использовать простую нейронную сеть с одним скрытым слоем. Для этого мы будем использовать библиотеку tensorflow.

```
import tensorflow as tf
# создаем модель
model = tf.keras.Sequential([
    tf.keras.layers.Dense(64, activation='relu', input_shape=(1,)),
    tf.keras.layers.Dense(1)
])
# компилируем модель
model.compile(optimizer='adam', loss='mse', metrics=['mae'])
```

В этом примере мы использовали полносвязный слой с 64 нейронами и функцией активации ReLU. Также мы добавили выходной слой с одним нейроном, который будет предсказывать цену закрытия. Мы используем функцию потерь MSE (Mean Squared Error) и метрику MAE (Mean Absolute Error) для оценки качества модели.

Далее мы подготавливаем данные для обучения и тестирования:

```
# создаем входные и выходные данные для обучения
train_X, train_y = train_data[:-1], train_data[1:]
# создаем входные и выходные данные для тестирования
test_X, test_y = test_data[:-1], test_data[1:]

# изменяем форму входных данных
train_X = train_X.reshape((train_X.shape[0], 1))
test_X = test_X.reshape((test_X.shape[0], 1))
```

В этом примере мы создали входные данные используя предыдущую цену закрытия для каждого дня в качестве входных данных (train_X и test_X), а следующую цену закрытия для каждого дня в качестве выходных данных (train_y

и test_y). Мы также изменили форму входных данных, чтобы они соответствовали ожидаемой форме входных данных для нашей модели.

```
# обучаем модель
history = model.fit(train_X, train_y, epochs=100, batch_size=16, validation_data=(test_X, test_y), verbose=2)
```

Мы обучили модель в течение 100 эпох с размером пакета 16 и использовали тестовые данные в качестве валидационных данных.

Наконец, мы можем использовать нашу модель для прогнозирования цен на следующий день:

```
# делаем прогноз на тестовых данных
predictions = model.predict(test_X)
# выводим прогнозы и фактические значения
for i in range(len(predictions)):
    print("Predicted: {:.2f}, Actual: {:.2f}".format(predictions[i][0], test_y[i]))
```

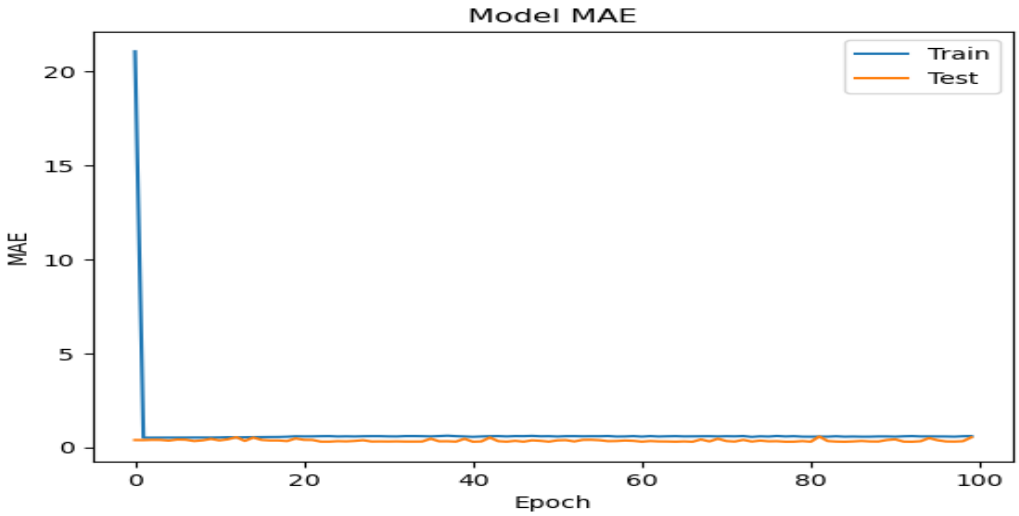
В этом примере мы сделали прогнозы на тестовых данных и вывели их вместе с фактическими значениями.

```
Predicted: 100.38, Actual: 99.81
Predicted: 100.28, Actual: 99.14
Predicted: 99.61, Actual: 99.03
Predicted: 99.50, Actual: 98.98
Predicted: 99.45, Actual: 98.78
Predicted: 99.25, Actual: 99.17
Predicted: 99.64, Actual: 99.32
Predicted: 99.79, Actual: 98.29
Predicted: 98.76, Actual: 97.75
Predicted: 98.21, Actual: 97.00
Predicted: 97.46, Actual: 97.21
Predicted: 97.67, Actual: 96.65
Predicted: 97.11, Actual: 95.95
Predicted: 96.41, Actual: 95.03
Predicted: 95.49, Actual: 94.73
Predicted: 95.18, Actual: 93.50
Predicted: 93.95, Actual: 91.80
Predicted: 92.25, Actual: 92.63
Predicted: 93.08, Actual: 92.81
```

Для оценки модели можно посмотреть на значение метрики MAE (Mean Absolute Error) на тестовой выборке. Эта метрика показывает среднюю абсолютную ошибку прогнозов модели.

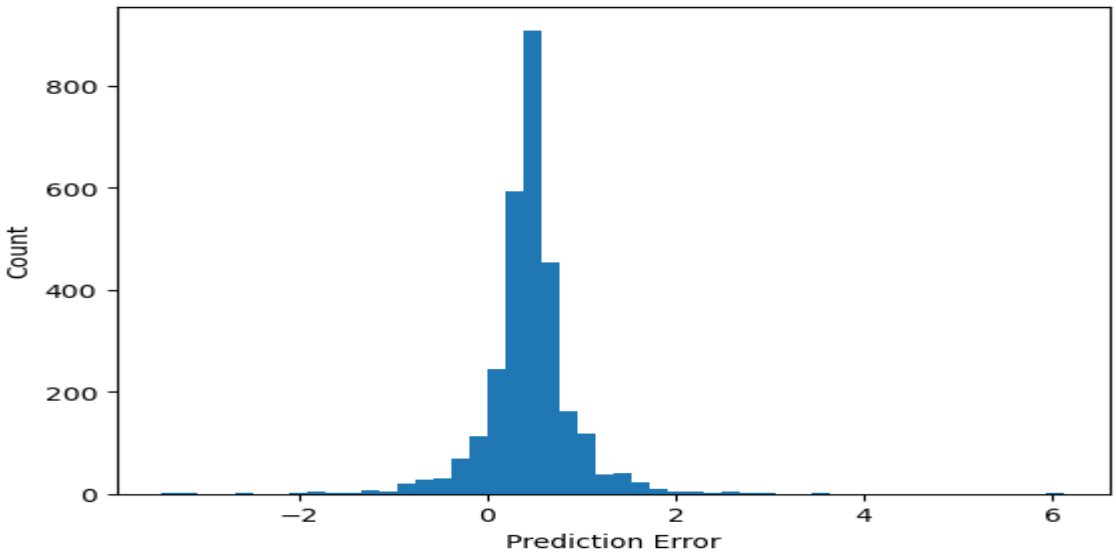
Выведем значение метрики MAE на последней эпохе обучения, он равен 0,53: print("MAE on test data: {:.2f}".format(history.history['val_mae'][-1]))

Также можно построить графики для визуализации качества обучения модели на обучающей и тестовой выборках:



Как видно из графика, модель показывает равномерные значения метрики на протяжении всех 100 эпох обучения, модель показывает хорошую точность прогнозирования цен акций.

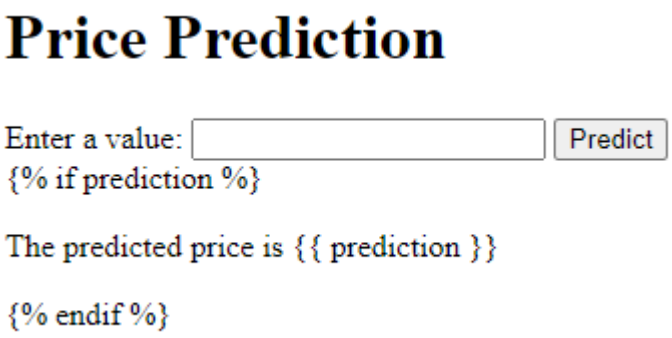
Для построения гистограммы распределения ошибки можно использовать библиотеку matplotlib. Для этого нужно сначала вычислить ошибку прогноза модели на тестовой выборке, а затем построить гистограмму распределения ошибки.



Гистограмма демонстрирует, что модель обучена достаточно хорошо, поскольку ошибка распределена более-менее равномерно вокруг 0.

2.5 Разработка приложения

Была попытка разработки приложения с помощью Flask.



Модель должна была выдавать прогнозную цену в зависимости от входных данных. Но что-то пошло не так, и приложение так и не заработало.

Ниже привожу код, найти ошибку и наладить приложение пока не смогла, с этим заданием я не справилась.

In [2]:
```
from flask import Flask, request, jsonify
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression

app = Flask(__name__)

# загружаем данные
df = pd.read_csv('https://raw.githubusercontent.com/OlgaDidenko/share-price/main/MOEX_200331_230331.csv')

# удаляем ненужные столбцы и строки
df.drop(['<TICKER>', '<PER>', '<TIME>', '<VOL>'], axis=1, inplace=True)
df.dropna(inplace=True)

# разделяем данные на обучающий и тестовый наборы
X = df.drop(['<CLOSE>'], axis=1)
y = df['<CLOSE>']
model = LinearRegression()
model.fit(X, y)
```

Out[2]:
```
LinearRegression()
```

In [6]:
```
@app.route('/', methods=['POST'], endpoint='predict_unique')
def predict():
    data = request.get_json()
    prediction = model.predict(np.array([data['Open'], data['High'], data['Low']]).reshape(1, -1))
    output = {'prediction': prediction[0]}
    return jsonify(output)

if __name__ == '__main__':
    app.run(debug=True)
```

```
--------------------------------------------------------------------------
```

3 Создание удаленного репозитория

Страница слушателя на GitLab

Созданный репозиторий:

https://github.com/OlgaDidenko/share-price

Заключение

В ходе выполнения ВКР была решена задача прогнозирования цены акции на основе исторических данных. Были применены различные методы машинного обучения, такие как линейная регрессия, решающие деревья, случайный лес, градиентный бустинг, а также написана нейронная сеть.

По результатам анализа было выявлено, что лучшей моделью для прогнозирования цены акции является линейная регрессия и градиентный бустинг.

В работе были использованы методы анализа и обработки данных, а также методы машинного обучения. Было проведено сравнение различных моделей и выбрана лучшая. Результаты работы могут быть применены в инвестиционной деятельности и финансовом анализе.

Библиографический список

1. Бхаргава А. Грокаем алгоритмы.
2. Герон А. Прикладное машинное обучение.
3. Дьяконов, А. Г. Основы машинного обучения.
4. Куликов М.А. Python и машинное обучение.
5. Миллер М. Python и анализ данных.
6. Рашка С. Python и машинное обучение.