

Прогнозирование цен на акции с использованием LSTM в Google Colab

Анишкова Анастасия Сергеевна

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

Цель данного исследования - создание интерактивного инструмента для анализа и прогнозирования цен на акции компании Apple с использованием модели LSTM и визуализации данных в Google Colab. Модель была реализована на облачной платформе Google Colab. Полученный результат предоставляет инструмент для анализа и прогнозирования цен на акции, который может быть полезен как для индивидуальных инвесторов, так и для финансовых аналитиков.

Ключевые слова: Google Colab, LSTM, прогнозирование цен и акций.

Stock Price Forecasting using LSTM in Google Colab

Anishkova Anastasia Sergeevna

Sholom Aleichem Priamurskiy State University

Student

Abstract

The purpose of this study is to create an interactive tool for analyzing and forecasting Apple stock prices using the LSTM model and data visualization in Google Colab. The model was implemented on the Google Colab cloud platform. The result provides a tool for analyzing and forecasting stock prices, which can be useful for both individual investors and financial analysts.

Key words: Google Colab, LSTM, price and stock forecasting.

1 Введение

1.1 Актуальность

Прогнозирование цен на акции с использованием LSTM в Google Colab является актуальным направлением благодаря своей практической пользе, доступности технологий и высокой точности прогнозов. Этот подход открывает новые возможности для инвесторов, аналитиков и исследователей, стремящихся применять современные методы машинного обучения для анализа финансовых рынков. LSTM позволяет эффективно анализировать временные ряды, такие как исторические цены акций, учитывать долгосрочные зависимости в данных, прогнозировать будущие цены, обучаться на последовательностях данных для повышения точности предсказаний и визуализировать результаты для оценки качества модели.

1.2 Обзор исследований

А. А. Архипова применила нейронные сети в задаче прогнозирования финансовых временных рядов [1], эффективность использования показателя Херста для прогнозирования курсов акций продемонстрировали Д. С. Сизых, Н. А. Буданова, Н. В. Сизых [2], К. С. Нешина спрогнозировала стоимости акций на основе нейросетевых моделей [3], прогнозирование технологических тенденций на основе анализа разнородных данных описали Н. Т. Вьет, А. Г. Кравец [4], О. С. Видмант использовал прогнозирование финансовых временных рядов с использованием рекуррентных нейронных сетей LSTM [5].

1.3 Цель исследования

Цель данного исследования - разработка интерактивного инструмента на основе модели LSTM для анализа и прогнозирования цен на акции компании Apple с визуализацией результатов в Google Colab.

2 Материалы и методы

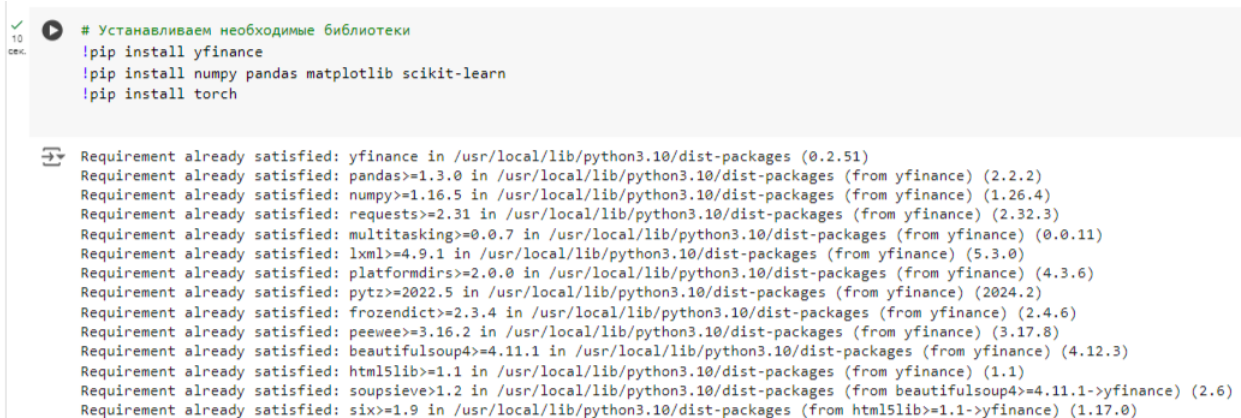
Google Colab предоставляет удобную облачную среду для разработки и выполнения кода, устраняя необходимость в установке дополнительного программного обеспечения, что делает процесс создания и обучения моделей машинного обучения более доступным и масштабируемым. LSTM, в свою очередь, обеспечивает эффективную обработку временных рядов, что позволяет анализировать исторические данные и прогнозировать будущие значения, такие как цены на акции. Это сочетание позволяет исследователям и аналитикам быстро тестировать и внедрять сложные модели, визуализировать результаты и принимать обоснованные решения на основе актуальных данных.

3 Результаты

Ссылка на ноутбук <https://colab.research.google.com/drive/1js-47Ehp-8-EAvAhXTd1BaZ-qmOKUO7J?usp=sharing>.

Для анализа данных будут использоваться исторические данные, загруженные с сайта Yahoo Finance [1]. Эти данные содержат информацию о ценах акций, включая цену открытия (Open), максимальную цену за день (High), минимальную цену за день (Low), цену закрытия (Close), объем торгов (Volume) и скорректированную цену закрытия (Adj Close).

Необходимо установить библиотеки (рис.1) yfinance-для загрузки данных о ценах акций, numpy, pandas, matplotlib и scikit-learn-для работы с данными и визуализации, а также torch (PyTorch)-для создания и обучения нейронной сети.

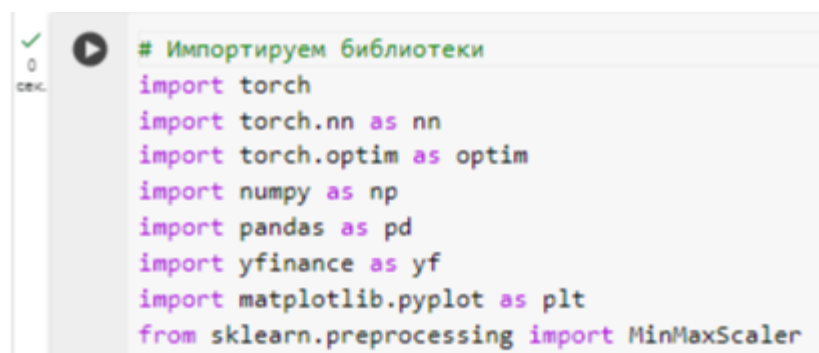


```
# Устанавливаем необходимые библиотеки
!pip install yfinance
!pip install numpy pandas matplotlib scikit-learn
!pip install torch

Requirement already satisfied: yfinance in /usr/local/lib/python3.10/dist-packages (0.2.51)
Requirement already satisfied: pandas>=1.3.0 in /usr/local/lib/python3.10/dist-packages (from yfinance) (2.2.2)
Requirement already satisfied: numpy>=1.16.5 in /usr/local/lib/python3.10/dist-packages (from yfinance) (1.26.4)
Requirement already satisfied: requests>=2.31 in /usr/local/lib/python3.10/dist-packages (from yfinance) (2.32.3)
Requirement already satisfied: multitasking>=0.0.7 in /usr/local/lib/python3.10/dist-packages (from yfinance) (0.0.11)
Requirement already satisfied: lxml>=4.9.1 in /usr/local/lib/python3.10/dist-packages (from yfinance) (5.3.0)
Requirement already satisfied: platformdirs>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from yfinance) (4.3.6)
Requirement already satisfied: pytz>=2022.5 in /usr/local/lib/python3.10/dist-packages (from yfinance) (2024.2)
Requirement already satisfied: frozendict>=2.3.4 in /usr/local/lib/python3.10/dist-packages (from yfinance) (2.4.6)
Requirement already satisfied: peewee>=3.16.2 in /usr/local/lib/python3.10/dist-packages (from yfinance) (3.17.8)
Requirement already satisfied: beautifulsoup4>=4.11.1 in /usr/local/lib/python3.10/dist-packages (from yfinance) (4.12.3)
Requirement already satisfied: html5lib>=1.1 in /usr/local/lib/python3.10/dist-packages (from yfinance) (1.1)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.10/dist-packages (from beautifulsoup4>=4.11.1->yfinance) (2.6)
Requirement already satisfied: six>=1.9 in /usr/local/lib/python3.10/dist-packages (from html5lib>=1.1->yfinance) (1.17.0)
```

Рисунок 1. Установка библиотек

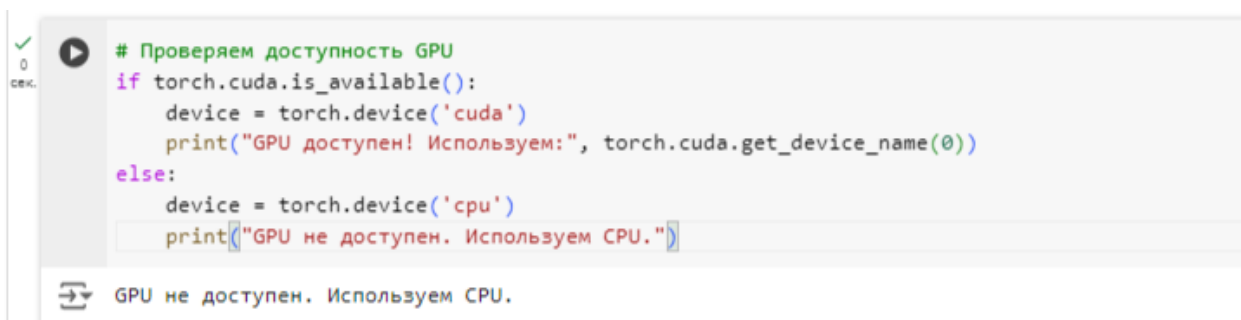
А также импортируем все необходимые библиотеки для работы с данными, построения модели и визуализации (рис.2).



```
# Импортируем библиотеки
import torch
import torch.nn as nn
import torch.optim as optim
import numpy as np
import pandas as pd
import yfinance as yf
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
```

Рисунок 2. Импортирование библиотек

Проверим доступен ли GPU (графический процессор) для выполнения вычислений с использованием библиотеки PyTorch (рис.3), и в зависимости от результата определяет, на каком устройстве (GPU или CPU) будут выполняться вычисления.



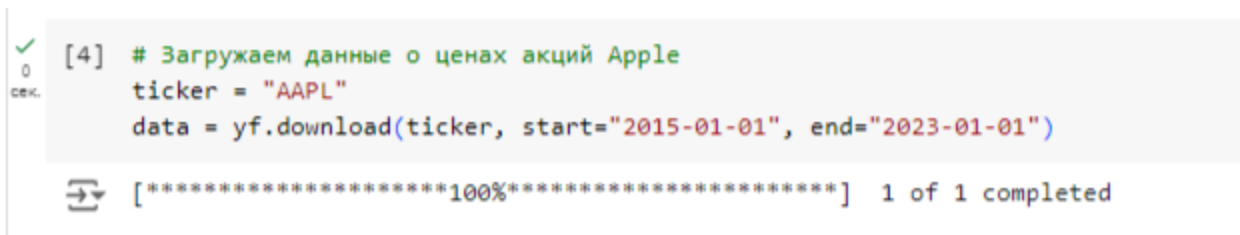
```
# Проверяем доступность GPU
if torch.cuda.is_available():
    device = torch.device('cuda')
    print("GPU доступен! Используем:", torch.cuda.get_device_name(0))
else:
    device = torch.device('cpu')
    print("GPU не доступен. Используем CPU.")

GPU не доступен. Используем CPU.
```

Рисунок 3. Доступ GPU

Загрузим исторические данные о ценах акций компании Apple (AAPL) с помощью библиотеки yfinance (рис.4). Эта библиотека использует API сервиса Yahoo Finance — популярной платформы, предоставляющей финансовую информацию, включая исторические данные о ценах акций, индексов, валют и других финансовых инструментов. Yahoo Finance собирает данные с бирж

(например, NASDAQ, NYSE) и предоставляет их в удобном формате для анализа. С помощью `yfinance` можно легко загрузить данные за указанный период, чтобы использовать их для анализа или обучения моделей.

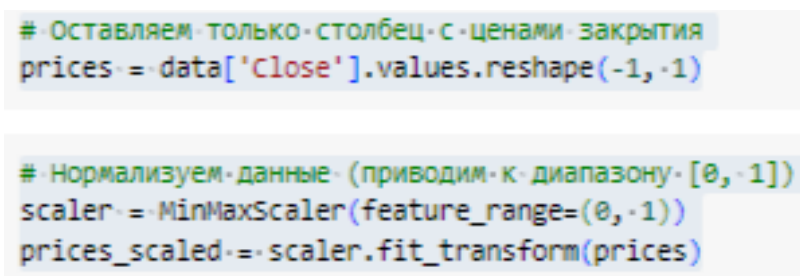


```
[4] # Загружаем данные о ценах акций Apple
     ticker = "AAPL"
     data = yf.download(ticker, start="2015-01-01", end="2023-01-01")

[*****100%*****] 1 of 1 completed
```

Рисунок 4. Загрузка данных

Извлечем данные о ценах закрытия акций из таблицы, содержащей историческую информацию, а затем преобразуем эти данные в формат, где каждая цена будет представлена отдельной строкой. Это потребуется для дальнейшей обработки. После этого создадим инструмент для нормализации данных, который приведет все значения к диапазону от 0 до 1. Применяв нормализацию, улучшим стабильность и скорость обучения моделей (рис.5).



```
# Оставляем только столбец с ценами закрытия
prices = data['Close'].values.reshape(-1, 1)

# Нормализуем данные (приводим к диапазону [0, 1])
scaler = MinMaxScaler(feature_range=(0, 1))
prices_scaled = scaler.fit_transform(prices)
```

Рисунок 5. Преобразование данных

Визуализируем нормализованные цены закрытия акций в виде графика, чтобы можно было наглядно оценить их динамику (рис.6).

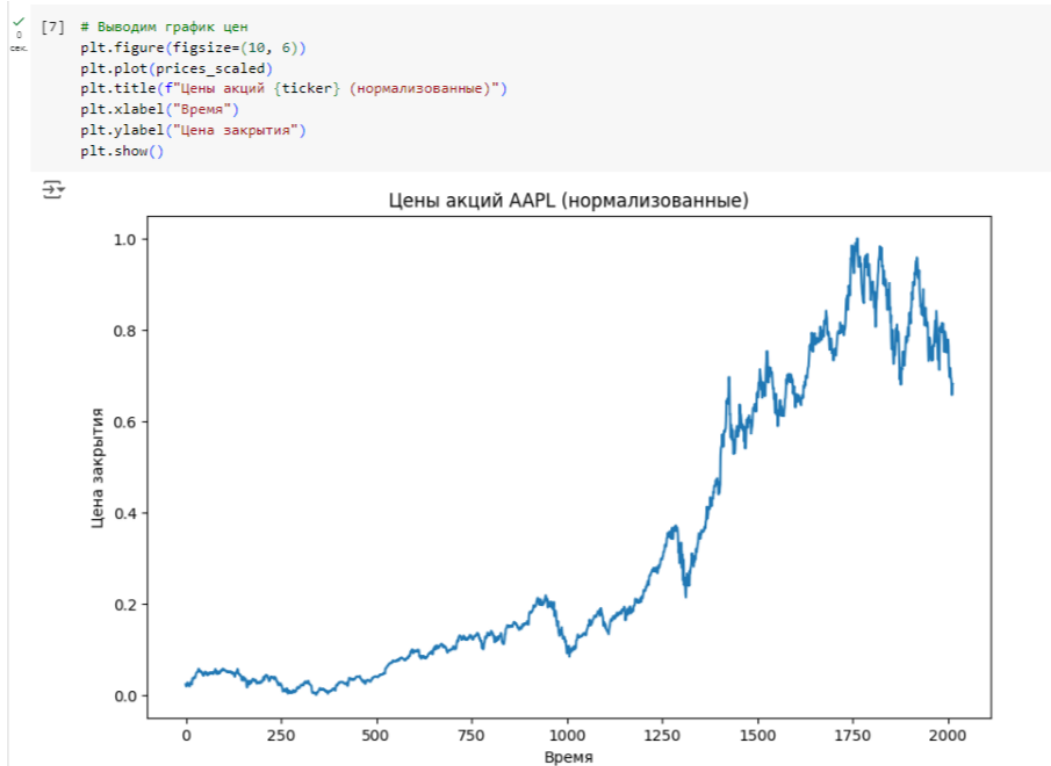


Рисунок 6. Визуализация

Создадим функцию `create_sequences`, которая преобразует временной ряд данных в последовательности (рис.7).

```

[8] # Функция для создания последовательностей
def create_sequences(data, seq_length):
    xs, ys = [], []
    for i in range(len(data) - seq_length):
        x = data[i:i + seq_length]
        y = data[i + seq_length]
        xs.append(x)
        ys.append(y)
    return np.array(xs), np.array(ys)

```

Рисунок 7. Функция для создания последовательностей

Подготовим данные для обучения модели: создадим последовательности, разделяя их на обучающую и тестовую выборки, а затем преобразует данные в формат, подходящий для PyTorch, и перемещает их на выбранное устройство CPU (рис.8).

```

✓ [9] # Длина последовательности (например, 60 дней)
0 seq. seq_length = 60

✓ [10] # Создаем последовательности
0 seq. X, y = create_sequences(prices_scaled, seq_length)

✓ [11] # Разделяем данные на обучающую и тестовую выборки
0 seq. train_size = int(len(X) * 0.8)
X_train, X_test = X[:train_size], X[train_size:]
y_train, y_test = y[:train_size], y[train_size:]

✓ [12] # Преобразуем данные в тензоры PyTorch
0 seq. X_train = torch.tensor(X_train, dtype=torch.float32).to(device)
y_train = torch.tensor(y_train, dtype=torch.float32).to(device)
X_test = torch.tensor(X_test, dtype=torch.float32).to(device)
y_test = torch.tensor(y_test, dtype=torch.float32).to(device)

```

Рисунок 8. Подготовка данных

Создает модель LSTM (рис.9), которая анализирует последовательности цен акций и предсказывает следующую цену, используя LSTM для обработки данных и полносвязный слой для получения итогового результата.

```

✓ [13] # Создаем модель LSTM
0 seq. class LSTMModel(nn.Module):
def __init__(self, input_size, hidden_size, output_size, num_layers=1):
super(LSTMModel, self).__init__()
self.hidden_size = hidden_size
self.num_layers = num_layers
self.lstm = nn.LSTM(input_size, hidden_size, num_layers, batch_first=True)
self.fc = nn.Linear(hidden_size, output_size)

def forward(self, x):
# Инициализация скрытых состояний
h0 = torch.zeros(self.num_layers, x.size(0), self.hidden_size).to(device)
c0 = torch.zeros(self.num_layers, x.size(0), self.hidden_size).to(device)

# Применяем LSTM
out, _ = self.lstm(x, (h0, c0))

# Применяем полносвязный слой к последнему выходу LSTM
out = self.fc(out[:, -1, :])
return out

```

Рисунок 9. Создание модели LSTM

Настроим модель LSTM, зададим функцию потерь, оптимизатор и параметры обучения, чтобы подготовить модель к процессу обучения (рис.10).

```

✓ 0 [14] # Параметры модели
сек. input_size = 1 # Размер входных данных (цена закрытия)
      hidden_size = 50 # Размер скрытого состояния
      output_size = 1 # Размер выходных данных (предсказанная цена)
      num_layers = 1 # Количество слоев LSTM

✓ 0 [15] # Создаем модель
сек. model = LSTMModel(input_size, hidden_size, output_size, num_layers).to(device)

✓ 5 [16] # Определяем функцию потерь и оптимизатор
сек. criterion = nn.MSELoss()
      optimizer = optim.Adam(model.parameters(), lr=0.001)

✓ 0 [17] # Параметры обучения
сек. num_epochs = 20
      batch_size = 32

```

Рисунок 10. Настройка модели LSTM

Обучим модель LSTM, корректируя её параметры на основе ошибок предсказаний, и продемонстрируем, как уменьшается ошибка после каждого этапа обучения (рис.11).

```

✓ 24 # Обучение модели
сек. for epoch in range(num_epochs):
      model.train()
      for i in range(0, len(X_train), batch_size):
          # Подготовка батча
          X_batch = X_train[i:i + batch_size]
          y_batch = y_train[i:i + batch_size]

          # Прямой проход
          outputs = model(X_batch)
          loss = criterion(outputs, y_batch)

          # Обратный проход и оптимизация
          optimizer.zero_grad()
          loss.backward()
          optimizer.step()

      print(f'Epoch [{epoch+1}/{num_epochs}], Loss: {loss.item():.4f}')

```

```

Epoch [1/20], Loss: 0.0283
Epoch [2/20], Loss: 0.1703
Epoch [3/20], Loss: 0.1084
Epoch [4/20], Loss: 0.0216
Epoch [5/20], Loss: 0.0011
Epoch [6/20], Loss: 0.0033
Epoch [7/20], Loss: 0.0039
Epoch [8/20], Loss: 0.0034
Epoch [9/20], Loss: 0.0002
Epoch [10/20], Loss: 0.0017
Epoch [11/20], Loss: 0.0003
Epoch [12/20], Loss: 0.0006
Epoch [13/20], Loss: 0.0002
Epoch [14/20], Loss: 0.0003
Epoch [15/20], Loss: 0.0002
Epoch [16/20], Loss: 0.0002
Epoch [17/20], Loss: 0.0002
Epoch [18/20], Loss: 0.0002
Epoch [19/20], Loss: 0.0002
Epoch [20/20], Loss: 0.0003

```

Рисунок 11. Обучение модели

Протестируем модель на тестовых данных, сделаем предсказания и вернем их к исходному масштабу, чтобы сравнить с реальными значениями (рис.12).

```
[19] # Тестирование модели
      model.eval()
      with torch.no_grad():
          # Предсказание на тестовых данных
          y_pred = model(X_test).cpu().numpy()

          # Обратное преобразование данных к исходному масштабу
          y_pred = scaler.inverse_transform(y_pred)
          y_test_actual = scaler.inverse_transform(y_test.cpu().numpy())
```

Рисунок 12. Тестирование модели

Создадим график, на котором сравниваются фактические и предсказанные цены акций, чтобы визуально оценить качество работы модели (рис.13).

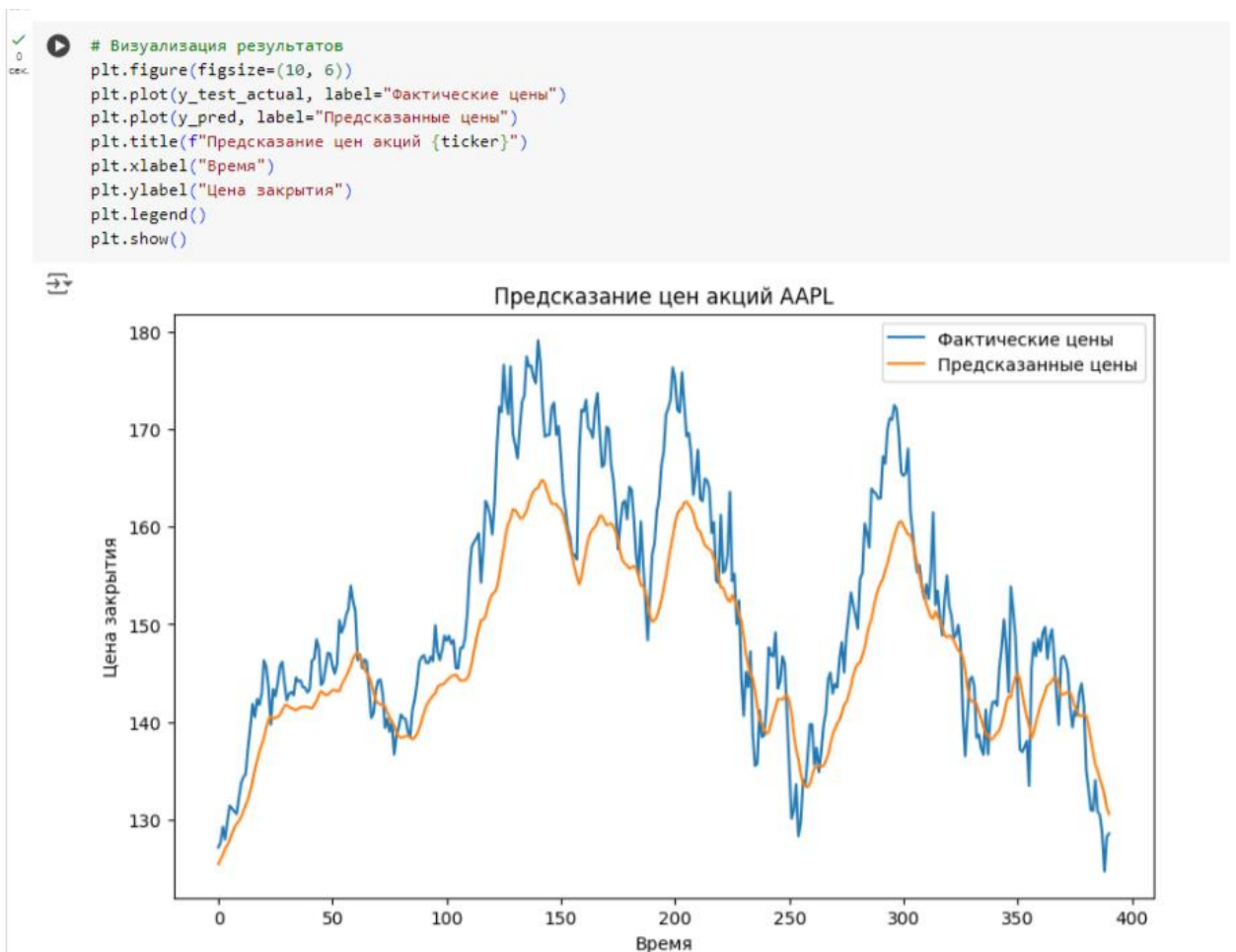


Рисунок 13. Визуализация результатов

Модель работает хорошо, так как предсказанные цены (оранжевая линия) близко следуют за реальными ценами (синяя линия). Это показывает, что модель правильно улавливает закономерности в данных и делает точные прогнозы.

Библиографический список

1. <https://finance.yahoo.com/>;
2. Архипова А. А. Применение нейронных сетей в задаче прогнозирования финансовых временных рядов // Экономика и бизнес: теория и практика. 2023. №6. С. 18-22.
3. Сизых Д. С., Буданова Н. А., Сизых Н. В. Эффективность использования показателя Херста для прогнозирования курсов акций // Управление развитием крупномасштабных систем. 2022. №4. С. 295-302.
4. Нешина К. С. Прогнозирование стоимости акций на основе нейросетевых моделей: наук: 38.04. 01-Экономика. Томск, 2022.
5. Вьет Н. Т., Кравец А. Г. Прогнозирование технологических тенденций на основе анализа разнородных данных // Программные продукты и системы. 2022. №3. С. 396-412.
6. Видмант О. С. Прогнозирование финансовых временных рядов с использованием рекуррентных нейронных сетей LSTM // Общество: политика, экономика, право. 2018. №5. С. 63-66.