Title: Stock Price Prediction Using LSTM

Artificial Intelligence 5th semester assignment project

B L Sunayana (225890048)

Aditya Pandey (225890264)

Anwesha Goel (225890318)

Harsheeta (225890340)

Ishita Singh (225890344)

Aarushi Garg (225890388)

Nidhi Tiwari (225890392)

Abstract

This project presents a Stock Price Prediction web application using Long Short-Term Memory (LSTM) networks, leveraging Streamlit for user interaction. The app allows users to select a company, define a date range for historical data, and predict future stock prices using a deep learning model. Stock data is sourced from Yahoo Finance[3], and the model provides forecasts based on past trends, visualized through interactive charts.

1. Introduction

Stock price forecasting is crucial in financial markets, helping investors make informed decisions. However, stock prices are volatile, requiring sophisticated techniques like machine learning for reliable predictions [6]. In this project, an LSTM neural network is used to predict future stock prices based on historical data. LSTM, a type of recurrent neural network, is especially effective in time series forecasting due to its ability to retain long-term dependencies [1].

This project simplifies stock price prediction for users without finance or data science expertise. Built with Streamlit, it lets users select major tech companies (Apple, Google, Microsoft, Amazon, Tesla, Meta) and forecast future stock prices by choosing a date range.

Historical data from yfinance is preprocessed using MinMaxScaler to improve model performance [3]. The LSTM model captures short- and long-term trends, and visualizations using Plotly display both historical and predicted stock trends for better investment insights [5].

2. Motivation

With the rapid growth of financial markets and the increasing number of retail investors, having access to reliable stock price prediction tools is more important than ever. Traditional methods often fail to capture complex patterns in stock data [6]. Deep learning models like LSTM can capture these patterns and provide more accurate forecasts [2]. This project aims to make these advanced predictions accessible through an intuitive, user-friendly interface.

3. Objective

The main objective of this project is to develop an easy-to-use web-based application for stock price prediction using LSTM networks. The app allows users to select a stock, define a time range for analysis, and predict future prices based on the historical data provided by Yahoo Finance.

4. Related Work

Previous research and projects have applied various machine learning techniques, including linear regression, decision trees, and random forests, to predict stock prices [2]. However, time series models such as ARIMA and LSTM have shown superior performance due to their ability to model sequential data and capture temporal dependencies. LSTM has become a popular choice for stock market prediction, outperforming many traditional models [4].

The application of machine learning to stock price prediction has gained significant attention in recent years, with various models and techniques explored to improve accuracy.

4.1 Traditional Stock Prediction Methods: Historically, stock prediction relied on fundamental and technical analysis. Fundamental analysis evaluates financial statements and market conditions, while technical analysis uses historical price patterns. Both methods struggle with the complexities of financial markets, driving interest in machine learning for better accuracy [7].

4.2 Machine Learning Models: Linear regression, decision trees, and SVM have been applied to stock prediction but often fail with non-linear, dynamic price data. This led to the use of more complex models like neural networks [1].

4.3 RNNs and LSTMs: Recurrent Neural Networks (RNNs) are suitable for time series data like stock prices but suffer from the vanishing gradient problem [2]. Long Short-Term Memory (LSTM) networks overcome this by retaining long-term dependencies, making them more effective in capturing both short-term and long-term trends in stock price data. LSTMs have shown superior performance in predicting market movements compared to traditional methods.

5. Methodology

The methodology for this project involves several key steps, from data collection to model training and deployment, aimed at accurately predicting stock prices. The process is structured as follows:

5.1 Data Collection:

Historical stock price data is sourced using the yfinance API, which provides comprehensive and up-to-date financial information for major technology companies [3]. The data includes daily stock prices, consisting of open, high, low, close, and volume values, which are essential for building accurate predictive models. Users can select a company and define a custom date range for analysis.

5.2 Data Preprocessing

Before training the model, the data undergoes preprocessing to ensure consistency and improve model performance. This includes handling missing values, scaling the data using MinMaxScaler to normalize the stock prices between 0 and 1, and splitting the data into training and test sets [4]. This step ensures that the model can generalize well and is not biased by the raw scale of the stock prices.

5.3 LSTM Model Development

The core predictive model used is a Long Short-Term Memory (LSTM) neural network. LSTMs are particularly suited for time series data due to their ability to remember previous time steps and learn dependencies in sequential data. A sequential Keras model is created, consisting of LSTM layers followed by dense layers, which output the predicted stock prices [4]. The model is trained on the historical stock data using a backpropagation algorithm with optimization techniques like Adam to minimize the prediction error [7].

5.4 Model Training and Evaluation

The model is trained using a portion of the dataset, with the remaining data reserved for evaluation. The performance of the model is assessed using metrics such as Mean Squared Error (MSE) to evaluate the accuracy of the predictions. Cross-validation techniques may also be applied to further validate the model's reliability.

5.5 Prediction and Visualization

Once trained, the model is used to predict future stock prices based on the user-defined date range and company selection. The results are visualized using Plotly, which provides interactive graphs of both historical and predicted prices, allowing users to easily interpret the stock trends [5].

5.6 Deployment

The application is deployed using Streamlit, providing a simple and intuitive interface where users can input their company choice and date range, receive predictions, and view the results in real-time. This makes the tool accessible to a broad range of users, regardless of their technical background.

This methodology ensures that the application is both accurate and user-friendly, delivering reliable stock price forecasts in an intuitive format.

6. Code Explanation

6.1 Streamlit UI Setup

The UI is designed using Streamlit, allowing users to interact with the app through a sidebar where they can select the stock, date range, and forecast period.

```python
import yfinance as yf
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM
import plotly.graph_objects as go
import streamlit as st
from datetime import date

# Streamlit UI Setup
st.set_page_config(page_title="Stock Price Prediction App", layout="centered", initial_sidebar_state="auto")

# Sidebar Configuration
st.sidebar.header("Stock Price Prediction")
st.sidebar.markdown("Select the company and date range to predict future stock prices.")

# Dropdown for stock selection in the sidebar
stock_options = {
    'Apple (AAPL)': 'AAPL',
    'Google (GOOG)': 'GOOG',
    'Microsoft (MSFT)': 'MSFT',
    'Amazon (AMZN)': 'AMZN',
    'Tesla (TSLA)': 'TSLA',
    'Facebook (META)': 'META'
}
```

```
28
29 stock_choice = st.sidebar.selectbox("Select a Company:", list(stock_options.keys()))
30 stock_ticker = stock_options[stock_choice]
31
32 # User input for date range in the sidebar
33 start_date = st.sidebar.date_input("From Date", value=date(2015, 1, 1), max_value=date.today())
34 end_date = st.sidebar.date_input("To Date", min_value=start_date, max_value=date.today())
35
36 # Option for predicting future prices
37 forecast_days = st.sidebar.slider("Number of days to forecast into the future", min_value=1, max_value=30, value=7)
38
39 # Main Page Title
40 st.title(f"Stock Price Prediction for {stock_choice}")
```

6.2 Data Fetching

The stock data is fetched from the yfinance API based on the user's input[3]. The data is cached using @st.cache_data to avoid re-downloading.

```
41
42 # Cache the data fetching to avoid re-downloading
43 @st.cache_data
44 def load_stock_data(ticker, start, end):
45     return yf.download(ticker, start=start, end=end)
46
47 # Button to trigger stock data download and prediction
48 if st.sidebar.button('Predict Stock Price'):
49
50     # Download stock data using yfinance
51     with st.spinner('Fetching stock data...'):
52         data = load_stock_data(stock_ticker, start_date, end_date)
53
54     # Handling missing values by filling forward
55     data.fillna(method='ffill', inplace=True)
```

6.3 Displaying Stock Data

The current stock price and its change from the previous day are displayed using st.metric(), and the historical stock prices are plotted using Plotly [5].

```
56
57     # Display the current stock price
58     current_price = data['Close'].iloc[-1]
59     previous_price = data['Close'].iloc[-2]
60     if current_price > previous_price:
61         st.metric(label="Current Stock Price", value=f"${current_price:.2f}", delta=f"{(current_price - previous_price):.2f}", delta_color="normal")
62     else:
63         st.metric(label="Current Stock Price", value=f"${current_price:.2f}", delta=f"{(current_price - previous_price):.2f}", delta_color="inverse")
64
65     # Visualize the historical stock price
66     st.subheader("Historical Stock Prices")
67     fig = go.Figure()
68     fig.add_trace(go.Scatter(x=data.index, y=data['Close'], mode='lines', name='Closing Price', line=dict(color='royalblue')))
69     st.plotly_chart(fig, use_container_width=True)
```

6.4 Preprocessing

The stock data is preprocessed by scaling it between 0 and 1 using MinMaxScaler[4]. The data is split into training (80%) and test sets. A function create_dataset() creates sequences of 60 days (time steps) of data to predict the next day's stock price.

```
    # Preprocessing the data
    df = data['Close'].values.reshape(-1, 1)

    # Feature scaling
    scaler = MinMaxScaler(feature_range=(0, 1))
    scaled_data = scaler.fit_transform(df)

    # Define train and test size
    train_size = int(len(scaled_data) * 0.8)
    train_data = scaled_data[0:train_size, :]
    test_data = scaled_data[train_size - 60:, :]

    # Function to create dataset
    def create_dataset(dataset, time_step=60):
        X, y = [], []
        for i in range(len(dataset) - time_step - 1):
            X.append(dataset[i:(i + time_step), 0])
            y.append(dataset[i + time_step, 0])
        return np.array(X), np.array(y)

    X_train, y_train = create_dataset(train_data)
    X_test, y_test = create_dataset(test_data)
```

6.5 LSTM Model

The LSTM model is built using the Sequential API in Keras. It has two LSTM layers followed by two dense layers.

```
    model = Sequential()
    model.add(LSTM(50, return_sequences=True, input_shape=(X_train.shape[1], 1)))
    model.add(LSTM(50, return_sequences=False))
    model.add(Dense(25))
    model.add(Dense(1))

    # Compile the model
    model.compile(optimizer='adam', loss='mean_squared_error')

    # Train the model
    with st.spinner('Training the LSTM model...'):
        model.fit(X_train, y_train, batch_size=32, epochs=20)
    st.success("Model trained successfully!")
```

6.6 Prediction

The model is used to predict the stock prices for the test data. The predicted values are then inverse transformed back to the original scale using the scaler.

```
    # Predict on test data
    predictions = model.predict(X_test)
    predictions = scaler.inverse_transform(predictions)

    # Adjust the length of the test data to match the predictions
    test = df[train_size:train_size + len(predictions), :]  # Adjust the length of test data
    test = pd.DataFrame(test, columns=['Actual Price'])
    test['Predictions'] = predictions

```

6.7 Future Forecasting

The model is also used to forecast future stock prices based on the last 60 days of stock data.

```

    # Forecast future prices
    last_60_days = scaled_data[-60:]
    X_forecast = []
    for i in range(forecast_days):
        pred_input = last_60_days.reshape(1, last_60_days.shape[0], 1)
        forecast_pred = model.predict(pred_input)
        last_60_days = np.append(last_60_days[1:], forecast_pred, axis=0)
        X_forecast.append(scaler.inverse_transform(forecast_pred)[0, 0])

    # Display forecast results
    future_dates = pd.date_range(end_date, periods=forecast_days + 1)[1:]
    future_dates = future_dates.date
    forecast_df = pd.DataFrame(X_forecast, index=future_dates, columns=['Forecast Price'])
    st.subheader("Predicted Future Stock Prices")
    st.dataframe(forecast_df)

```

6.8 Visualization

The predicted vs actual stock prices and future predictions are visualized using Plotly.

```

    # Plot the predicted vs actual prices and future predictions
    st.subheader("Predicted vs Actual Stock Prices")
    fig2 = go.Figure()
    fig2.add_trace(go.Scatter(x=test.index, y=test['Actual Price'], mode='lines', name='Actual Price', line=dict(color='yellow')))
    fig2.add_trace(go.Scatter(x=test.index, y=test['Predictions'], mode='lines', name='Predicted Price', line=dict(color='blue')))
    fig2.add_trace(go.Scatter(x=future_dates, y=forecast_df['Forecast Price'], mode='lines', name='Future Predictions', line=dict(color='green')))
    st.plotly_chart(fig2, use_container_width=True)

```

7. Output

Figure1:UI of the Stock Prediction Application

Figure 2: Graph displaying historical stock data

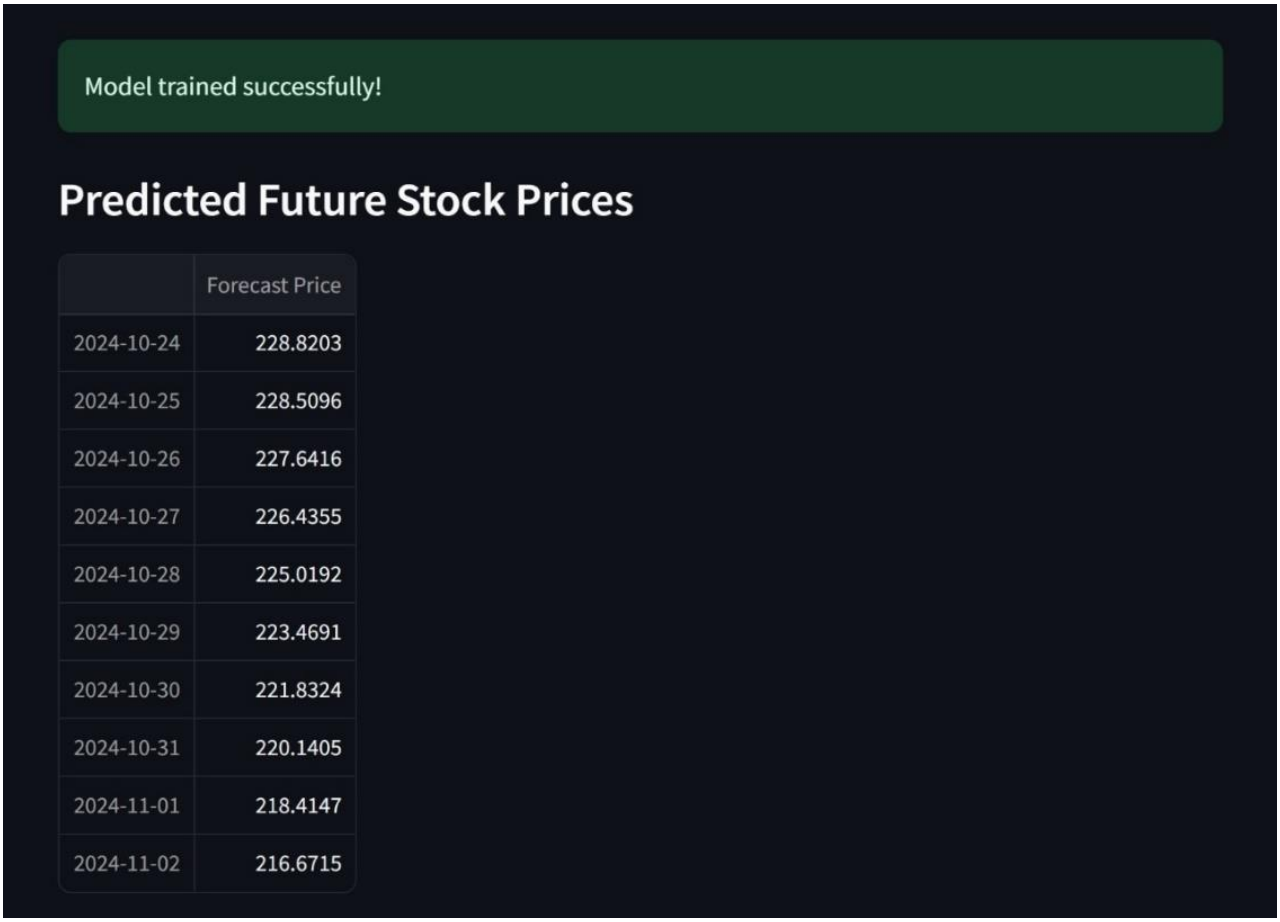

Figure 3: Stock prices predicted for the range of days specified by the user

Figure 4: Line Chart comparing stock prices predicted by the model, and actual stock prices recorded on the same day

8. Conclusion

This project successfully demonstrates the application of Long Short-Term Memory (LSTM) neural networks for predicting stock prices, providing a valuable tool for investors and analysts. By integrating real-time stock data with an intuitive web interface, the application makes advanced machine learning accessible to non-experts, allowing users to make informed financial decisions based on data-driven predictions . The use of LSTM models ensures that both short-term and long-term trends in stock price movements are captured, improving forecast accuracy. Overall, this project highlights the potential of AI in financial markets and sets the stage for future enhancements.

9. Future Scope

Enhancing the model with additional data features and refining the architecture will improve prediction accuracy and robustness. Real-time data integration and more sophisticated visualizations can elevate the user experience, transforming this tool into a comprehensive stock analysis platform for investors, analysts, and financial enthusiasts.

References

1. **Hochreiter, S., & Schmidhuber, J. (1997).** Long Short-Term Memory. *Neural Computation*, 9(8), 1735-1780. https://direct.mit.edu/neco/article-abstract/9/8/1735/6109/Long-Short-Term-Memory?redirectedFrom=fulltext
   Cited in: Introduction[1], Related Work[4.2], Methodology

2. **Brownlee, J. (2017).** *Deep Learning for Time Series Forecasting with Python*. Machine Learning Mastery. *Cited in: Motivation[3], Related Work[4,4.2]*

3. **Yahoo Finance. (2024).** Financial data for stocks. Available at Yahoo Finance API. Link to API Documentation
   Cited in: Introduction [1], Methodology[5.1]

4. **Keras Documentation. (2024).** *Keras: The Python Deep Learning API*. Link to Documentation
   Cited in: Related Works[4], Methodology[5.2,5.3]

5. **Plotly. (2024).** *Plotly Python Graphing Library*. Link to Documentation
   Cited in: Introduction[1], Methodology[5.5], Code Explanation [6.3]

6. **K. C. A and A. James**, "A Survey on Stock Market Prediction Techniques," 2023 International Conference on Power, Instrumentation, Control and Computing (PICC), Thrissur, India, 2023, pp. 1-6, doi: 10.1109/PICC57976.2023.10142717.
   Cited in: Introduction[1], Motivation[2]

7. **Nti, Isaac & Adekoya, Adebayo & Weyori, Benjamin. (2020)**. A systematic review of fundamental and technical analysis of stock market predictions. Artificial Intelligence Review. 53. 10.1007/s10462-019-09754-z.
   Cited in: Related Work[4.1], Methodology [5.3]