Stock Price Prediction using LSTM Neural Network

Nirwan Vimukthijith

July 27, 2025

Abstract

This project uses a Long Short-Term Memory (LSTM) neural network to predict the closing stock prices of Apple Inc. using historical stock data. The model is trained on five years of data using TensorFlow and Keras. The process includes data preprocessing, visualization, model training, and evaluation of predictions.

1 Introduction

Predicting stock market trends is a well-known challenge due to the dynamic nature of financial data. This project focuses on applying an LSTM neural network—a form of recurrent neural network (RNN) suitable for time series data—to forecast the closing price of Apple (AAPL) stock.

2 Partitioned and detailed code

1. Importing Required Libraries

Listing 1: Import Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error, r2_score
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
```

2. Loading and Preparing the Dataset

Listing 2: Load and Prepare Data

```
data = pd.read_csv('AAPL.csv')
```

```
data = data[['Date', 'Close']]
data['Date'] = pd.to_datetime(data['Date'])
data.set_index('Date', inplace=True)
data = data.sort_index()
```

3. Normalizing the Data

Listing 3: Normalize Data

```
scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(data[['Close']])
```

4. Creating Sequences for LSTM Input

Listing 4: Generate Time Series Sequences

```
X, y = [], []
sequence_length = 60
for i in range(sequence_length, len(scaled_data)):
    X.append(scaled_data[i-sequence_length:i, 0])
    y.append(scaled_data[i, 0])

X = np.array(X)
y = np.array(y)
X = np.reshape(X, (X.shape[0], X.shape[1], 1))
```

5. Splitting Data into Training and Testing Sets

Listing 5: Train-Test Split

```
split = int(0.8 * len(X))
X_train, X_test = X[:split], X[split:]
y_train, y_test = y[:split], y[split:]
```

6. Defining and Training the LSTM Model

Listing 6: Build and Train LSTM Model

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

model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(X_train, y_train, epochs=10, batch_size=32)
```

7. Making Predictions and Evaluating Performance

Listing 7: Predict and Evaluate

```
predicted = model.predict(X_test)
predicted_prices = scaler.inverse_transform(predicted.reshape(-1, 1))
actual_prices = scaler.inverse_transform(y_test.reshape(-1, 1))

mse = mean_squared_error(actual_prices, predicted_prices)
r2 = r2_score(actual_prices, predicted_prices)
print(f"MSE: {mse}, R2 Score: {r2}")
```

8. Visualizing Results

Listing 8: Plot Actual vs Predicted Prices

```
plt.figure(figsize=(12,6))
plt.plot(actual_prices, label='Actual')
plt.plot(predicted_prices, label='Predicted')
plt.title('Apple Stock Price Prediction')
plt.xlabel('Days')
plt.ylabel('Price')
plt.legend()
plt.show()
```

3 Required Libraries

The following Python libraries were used:

- **pandas**: For data loading and manipulation.
- **numpy**: For numerical operations and array handling.
- **matplotlib & seaborn**: For data visualization.
- **tensorflow.keras**: To define, train, and evaluate the LSTM model.
- **sklearn.preprocessing.MinMaxScaler**: To scale features between 0 and 1.
- **datetime & warnings**: For date filtering and warning suppression.

4 Library Descriptions

This section explains the purpose and usage of each Python library used in the project.

4.1 1. `pandas`

- **What it is**: A powerful library for data analysis and manipulation.
- **Why used**: To read the dataset, filter companies, and handle date formats.
- **How used**:
  - `pd.read_csv()` loads the dataset.
  - `data['Name'] == 'AAPL'` filters Apple stock data.
  - `pd.to_datetime()` converts date strings to datetime objects.

4.2 2. `numpy`

- **What it is**: A core scientific library for numerical computing in Python.
- **Why used**: For handling numerical data and performing array operations.
- **How used**:
  - `np.array()` creates numerical arrays.
  - `np.mean()`, `np.sqrt()` calculate evaluation metrics like MSE and RMSE.
  - Reshaping arrays to match LSTM input format.

4.3 3. `matplotlib.pyplot`

- **What it is**: A plotting library for creating static and interactive graphs.
- **Why used**: To visualize stock data and model predictions.
- **How used**:
  - Line plots for `open`, `close`, and `volume` over time.
  - Plotting actual vs predicted prices after model evaluation.

4.4 4. `seaborn`

- **What it is**: A statistical data visualization library built on top of matplotlib.
- **Why used**: Typically used for attractive statistical plots.
- **How used**: *Not used directly in this code*, but helpful for correlation or heatmaps.

4.5 5. `tensorflow` and `keras`

- **What it is**: TensorFlow is a deep learning framework; Keras is its high-level API.
- **Why used**: To build and train the LSTM model for time-series forecasting.
- **How used**:
  - `keras.models.Sequential` defines the model structure.
  - `keras.layers.LSTM`, `Dense`, `Dropout` create model layers.
  - `model.fit()` trains the model.
  - `model.predict()` makes future predictions.

4.6 6. `sklearn.preprocessing.MinMaxScaler`

- **What it is**: A feature scaler from the Scikit-Learn library.
- **Why used**: LSTM performs better when input features are normalized.
- **How used**:
  - `fit_transform()` normalizes the training data.
  - `inverse_transform()` converts predicted data back to actual scale.

4.7 7. `os`

- **What it is**: Python's built-in module to interact with the operating system.
- **Why used**: *Not used directly* in the code.
- **Note**: Could be used for file management (e.g., saving plots, models).

4.8 8. `datetime`

- **What it is**: A module to handle dates and times.
- **Why used**: To filter data by specific date ranges.
- **How used**:
  - Used to select a range for analysis: e.g., `datetime(2013, 1, 1)`.

4.9 9. `warnings`

- **What it is**: Built-in Python module to manage warning messages.
- **Why used**: To suppress unwanted warnings during execution.
- **How used**:
  - `warnings.filterwarnings("ignore")` disables all warnings.

5 Data Preprocessing

```
data = pd.read_csv('all_stocks_5yr.csv', on_bad_lines='skip')
data['date'] = pd.to_datetime(data['date'])
companies = ['AAPL', 'AMD', 'FB', 'GOOGL', 'AMZN', 'NVDA', 'EBAY', 'CSCO',
    'IBM']
```

The CSV file is loaded and parsed. Only selected companies are filtered for visualization, and dates are converted for time-based analysis.

6 Data Visualization

Two main plots are created:

- Open vs Close prices over time for each company.
- Trading volume over time.

```
plt.plot(c['date'], c['close'], label="Close")
plt.plot(c['date'], c['open'], label="Open")
```

Apple's closing prices are further visualized separately to better understand historical trends.

7 Data Preparation

The model uses a 60-day lookback window to predict the next day's stock price. The data is normalized using MinMaxScaler.

```
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(close_data.values)

for i in range(60, len(train_data)):
    x_train.append(train_data[i-60:i, 0])
    y_train.append(train_data[i, 0])
```

8 Model Architecture

A sequential LSTM model is built using TensorFlow/Keras with the following layers:

- Two LSTM layers with 64 units each.
- One Dense layer with 32 units.
- Dropout layer to prevent overfitting.
- Output layer with 1 unit (closing price).

```
model = keras.models.Sequential([
    keras.layers.LSTM(64, return_sequences=True, input_shape=(60, 1)),
    keras.layers.LSTM(64),
    keras.layers.Dense(32),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(1)
])
model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(x_train, y_train, epochs=10)
```

9 Testing and Prediction

The model is tested on the last 5% of the dataset. Predictions are made and inverse-transformed to get actual price values.

```
predictions = model.predict(x_test)
predictions = scaler.inverse_transform(predictions)
```

10 Evaluation and Results

Model performance is measured using MSE and RMSE. A plot is created to compare actual vs predicted stock prices.

```
mse = np.mean((predictions - y_test) ** 2)
rmse = np.sqrt(mse)
```

```
plt.plot(train['date'], train['close'], label='Train')
plt.plot(test['date'], test['close'], label='Test')
plt.plot(test['date'], test['Predictions'], label='Predicted')
```

11 Conclusion

The LSTM model successfully captured trends in Apple stock data and was able to make reasonably accurate forecasts. While LSTM networks are powerful, the stock market is influenced by many external factors, so further improvements may include integrating news sentiment, technical indicators, or attention-based models.

References

- TensorFlow Documentation: `https://www.tensorflow.org/`
- Yahoo Finance (for real-time data): `https://finance.yahoo.com/`
- Dataset Source: *all_stocks_5yr.csv*