Atajan Arashev

Personal email:

[arashevatajan@gmail.com](mailto:arashevatajan@gmail.com)

University email:

[atajanarashev@my.unt.edu](mailto:atajanarashev@my.unt.edu)

Contents

[1. Project Overview 1](#_Toc161087344)

[2. Data Collection 2](#_Toc161087345)

[3. Data Preparation 2](#_Toc161087346)

[4. Feature Engineering 3](#_Toc161087347)

[5. Model Building 3](#_Toc161087348)

[6. Training 4](#_Toc161087349)

[7. Evaluation 4](#_Toc161087350)

[Advanced Model Development and Time Series Cross-Validation 5](#_Toc161087351)

[8. Prediction 7](#_Toc161087352)

[9. Conclusion 9](#_Toc161087353)

"Forecasting Financial Futures: Enhancing Stock Market Predictions with LSTM Neural Networks"

# 1. Project Overview

This project delves into the predictive power of machine learning by employing a Long Short-Term Memory (LSTM) neural network to forecast the closing stock prices of Apple Inc. By leveraging historical stock price data, this deep learning approach aims to provide a forward-looking perspective on market trends, focusing specifically on the 60-day price movements leading up to predictions. The endeavor showcases the application of LSTM networks in deciphering complex patterns within time-series data, demonstrating their potential to offer valuable insights for financial forecasting. Through this project, we explore the integration of advanced machine learning techniques in the financial domain, illustrating the LSTM model's capability to navigate the intricacies of stock market fluctuations and predict future price trajectories.

A graph showing a line graph

Description automatically generated with medium confidence

# 2. Data Collection

The stock price data for Apple Inc. (AAPL) is downloaded from Yahoo Finance using the “**yfinance”** library. The dataset spans from January 1, 2012, to December 17, 2019. This data includes various stock market indicators, but for this project, we primarily focus on the closing price.

# Import necessary libraries

import pandas\_datareader as pdr

import yfinance as yf

# Download the stock data

df = yf.download('AAPL', start='2012-01-01', end='2019-12-17')

# 3. Data Preparation

* A new DataFrame is created containing only the 'Close' column.
* The data is then converted into a numpy array for processing.
* The dataset is split into training and testing sets, with 80% of the data used for training.
* The data is scaled using MinMaxScaler to fit within a specified range (between 0 and 1) to improve the model's convergence.

# Create a new dataframe with only the 'Close' column

data = df.filter(['Close'])

# Convert the dataframe to a numpy array

dataset = data.values

# Get the number of rows to train the model on (80% of the dataset)

train\_data\_len = math.ceil(len(dataset) \* .8)

# Scale the data

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler(feature\_range=(0, 1))

scaled\_data = scaler.fit\_transform(dataset)

# 4. Feature Engineering

* The training data is organized into sequential datasets for the LSTM model. For each data point, the corresponding input feature is the stock prices for the previous 60 days, and the target output is the stock price on the next day.
* This process is repeated for both training and testing datasets to prepare them for the model.
* # Create the scaled training data set
* train\_data = scaled\_data[0:train\_data\_len, :]
* # Split the data into x\_train and y\_train datasets
* x\_train, y\_train = [], []
* for i in range(60, len(train\_data)):
* x\_train.append(train\_data[i-60:i, 0])
* y\_train.append(train\_data[i, 0])
* # Convert the x\_train and y\_train to numpy arrays
* x\_train, y\_train = np.array(x\_train), np.array(y\_train)
* # Reshape the data to be 3-dimensional (samples, time steps, features)
* x\_train = np.reshape(x\_train, (x\_train.shape[0], x\_train.shape[1], 1))

# 5. Model Building

* An LSTM model is constructed using the Keras library. The model consists of two LSTM layers and two Dropout layers to prevent overfitting. This is followed by two Dense layers, with the final layer outputting the predicted stock price.
* The model is compiled with the Adam optimizer and the mean squared error loss function.
* from keras.models import Sequential
* from keras.layers import Dense, LSTM, Dropout
* # Build the LSTM model
* model = Sequential()
* model.add(LSTM(50, return\_sequences=True, input\_shape=(x\_train.shape[1], 1)))
* model.add(Dropout(0.2))
* model.add(LSTM(50, return\_sequences=False))
* model.add(Dropout(0.2))
* model.add(Dense(25))
* model.add (Dense(1))
* # Compile the model

model.compile(optimizer='adam', loss='mean\_squared\_error')

# 6. Training

The LSTM model is trained on the training dataset for 100 epochs with a batch size of 32. This step involves the model learning to predict the stock price based on the input features (past 60 days of stock prices).

# Train the model

model.fit(x\_train, y\_train, batch\_size=32, epochs=100)

# 7. Evaluation

* The model's performance is evaluated using the root mean squared error (RMSE) metric, which measures the difference between the predicted and actual stock prices in the testing Root Mean Squared Error (RMSE): This metric provides a measure of the model's prediction errors by calculating the square root of the average squared differences between the predicted and actual values. A lower RMSE indicates better model performance.
* Mean Absolute Error (MAE): MAE measures the average magnitude of the errors in a set of predictions, without considering their direction. It's the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal weight.
* Mean Absolute Percentage Error (MAPE): MAPE expresses accuracy as a percentage of the error and is calculated as the average of the absolute percentage errors of forecasts. It provides a simple interpretation of prediction accuracy by indicating how far the model's predictions are off from their actual values on average.
* from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error
* # Calculate RMSE
* rmse = np.sqrt(mean\_squared\_error(y\_test, predictions))
* # Calculate MAE
* mae = mean\_absolute\_error(y\_test, predictions)
* # Calculate MAPE
* mape = np.mean(np.abs((y\_test - predictions) / y\_test)) \* 100
* # Display the metrics
* print(f'Root Mean Squared Error (RMSE): {rmse}')
* print(f'Mean Absolute Error (MAE): {mae}')
* print(f'Mean Absolute Percentage Error (MAPE): {mape}%')

Root Mean Squared Error: 1.2819476740201412

Mean Absolute Error: 1.0732230949401855

Mean Absolute Percentage Error: 2.1213006176101947 %

# Advanced Model Development and Time Series Cross-Validation

In this advanced section of the project, we leverage sophisticated data preprocessing, model building techniques, and rigorous validation methods to enhance the predictive capability of our LSTM model for forecasting Apple Inc.'s closing stock prices based on the past 60 days of data.

**Output of the code:**

9/9 [==============================] - 5s 47ms/step - loss: 0.0045

9/9 [==============================] - 1s 15ms/step

Fold 1 - Root Mean Squared Error (RMSE): 1.6127

Fold 1 - Mean Absolute Error (MAE): 1.4508

Fold 1 - Mean Absolute Percentage Error (MAPE): 7.4813%

20/20 [==============================] - 5s 71ms/step - loss: 0.0019

9/9 [==============================] - 4s 23ms/step

Fold 2 - Root Mean Squared Error (RMSE): 2.4579

Fold 2 - Mean Absolute Error (MAE): 2.2462

Fold 2 - Mean Absolute Percentage Error (MAPE): 7.3738%

30/30 [==============================] - 8s 54ms/step - loss: 0.0049

9/9 [==============================] - 1s 23ms/step

Fold 3 - Root Mean Squared Error (RMSE): 1.6505

Fold 3 - Mean Absolute Error (MAE): 1.4342

Fold 3 - Mean Absolute Percentage Error (MAPE): 4.8883%

40/40 [==============================] - 6s 47ms/step - loss: 0.0039

9/9 [==============================] - 1s 14ms/step

Fold 4 - Root Mean Squared Error (RMSE): 2.3207

Fold 4 - Mean Absolute Error (MAE): 1.9763

Fold 4 - Mean Absolute Percentage Error (MAPE): 4.4586%

51/51 [==============================] - 7s 48ms/step - loss: 0.0110

9/9 [==============================] - 1s 15ms/step

Fold 5 - Root Mean Squared Error (RMSE): 2.5270

Fold 5 - Mean Absolute Error (MAE): 1.7547

Fold 5 - Mean Absolute Percentage Error (MAPE): 3.7553%

|  |
| --- |
| Fold 1 |
| **Training:** Completed in 5 seconds for 9 steps, averaging 47ms/step, with a loss of 0.0045. |
| **Prediction Time:** Took 1 second for 9 steps, averaging 15ms/step. |
| **Metrics:** |
| RMSE: 1.6127, indicating an average error of 1.6127 units. |
| MAE: 1.4508, showing the mean error magnitude is 1.4508 units. |
| MAPE: 7.4813%, meaning predictions were off by 7.4813% on average. |
|  |
| Fold 2 |
| **Training:** Required 5 seconds over 20 steps, with each step taking about 71ms, and a loss of 0.0019. |
| **Prediction Time:** 4 seconds for 9 steps, at 23ms/step. |
| **Metrics:** |
| RMSE: 2.4579, suggesting predictions deviated by an average of 2.4579 units. |
| MAE: 2.2462, indicating an average error magnitude of 2.2462 units. |
| MAPE: 7.3738%, implying predictions were 7.3738% away from actual values on average. |
| Fold 3 |
| **Training:** 8 seconds were needed for 30 steps, averaging 54ms/step, with a loss of 0.0049. |
| **Prediction Time:** 1 second for 9 steps, averaging 23ms/step. |
| **Metrics:** |
| RMSE: 1.6505, showing an average error of 1.6505 units. |
| MAE: 1.4342, reflecting the mean error magnitude of 1.4342 units. |
| MAPE: 4.8883%, indicating predictions were off by 4.8883% on average. |
| Fold 4 |
| **Training:** Took 6 seconds over 40 steps, averaging 47ms/step, with a loss of 0.0039. |
| **Prediction Time:** 1 second for 9 steps, at 14ms/step. |
| **Metrics:** |
| RMSE: 2.3207, suggesting an average error of 2.3207 units. |
| MAE: 1.9763, showing an average error magnitude of 1.9763 units. |
| MAPE: 4.4586%, meaning predictions were 4.4586% away from actual values on average. |
| Fold 5 |
| **Training:** Required 7 seconds for 51 steps, averaging 48ms/step, with a loss of 0.0110. |
| **Prediction Time:** 1 second for 9 steps, averaging 15ms/step. |
| **Metrics:** |
| RMSE: 2.5270, indicating predictions were off by an average of 2.5270 units. |
| MAE: 1.7547, reflecting a mean error magnitude of 1.7547 units. |
| MAPE: 3.7553%, implying predictions were 3.7553% away from actual values on average. |

# 8. Prediction

* In this phase, the developed LSTM model is employed to predict future closing stock prices. The prediction task commences from the last recorded trading day of the dataset and extends for an anticipated two-month period. The prediction strategy involves updating the model iteratively with new data points as predictions are made, which simulates a rolling forecast mechanism often used in time series analysis.

A graph showing a line graph

Description automatically generated with medium confidence

1. **Data Preparation and Model Training:** The dataset is initially filtered to include only the 'Close' column, which is then converted into a numpy array for numerical processing. The data is scaled using MinMaxScaler to normalize the values within a [0, 1] range, improving the neural network's performance.
2. **Model Architecture:** A Sequential LSTM model is constructed with 50 neurons in the first LSTM layer, followed by a Dropout layer to mitigate overfitting. A second LSTM layer with another Dropout layer precedes two Dense layers, finalizing the model design.
3. **Training Process:** The model is compiled with the Adam optimizer and means squared error as the loss function. It is then trained on the pre-processed training data for 100 epochs with a batch size of 32.
4. **Forecasting**: Post-training, the model performs predictions on the test set, which comprises the most recent 60 days of data that the model has not seen during training. The predictions are then inverse transformed to scale back to the original price range.
5. **Evaluation**: The model’s predictive accuracy is quantitatively evaluated by calculating the root mean squared error (RMSE) between the predicted and actual stock prices.

# "Future Price Forecasting"

For the prediction phase, the LSTM model anticipates the closing stock prices for a period of approximately two months, starting from the final trading day of 2023. The prediction is based on the most up-to-date data, specifically the last 60 days leading up to the “**start\_date**,” which is set to December 31, 2023. The model updates its inputs sequentially with the newly predicted values to generate a series of future prices.

The provided predictions offer an estimate of the future closing prices based on the LSTM model's learned patterns from historical data.

| **Real Date** | **Prediction Day** | **Predicted Price** |
| --- | --- | --- |
| 2024-01-02 | Day 1 | $138.76 |
| 2024-01-03 | Day 2 | $138.22 |
| 2024-01-04 | Day 3 | $137.38 |
| 2024-01-05 | Day 4 | $136.36 |
| 2024-01-08 | Day 5 | $135.22 |
| 2024-01-09 | Day 6 | $134.00 |
| 2024-01-10 | Day 7 | $132.76 |
| 2024-01-11 | Day 8 | $131.50 |
| 2024-01-12 | Day 9 | $130.25 |
| 2024-01-15 | Day 10 | $129.03 |
| 2024-01-16 | Day 11 | $127.85 |
| 2024-01-17 | Day 12 | $126.71 |
| 2024-01-18 | Day 13 | $125.62 |
| 2024-01-19 | Day 14 | $124.58 |
| 2024-01-22 | Day 15 | $123.59 |
| 2024-01-23 | Day 16 | $122.65 |
| 2024-01-24 | Day 17 | $121.75 |
| 2024-01-25 | Day 18 | $120.90 |
| 2024-01-26 | Day 19 | $120.09 |
| 2024-01-29 | Day 20 | $119.32 |
| 2024-01-30 | Day 21 | $118.59 |
| 2024-01-31 | Day 22 | $117.89 |
| 2024-02-01 | Day 23 | $117.23 |
| 2024-02-02 | Day 24 | $116.60 |
| 2024-02-05 | Day 25 | $116.00 |
| 2024-02-06 | Day 26 | $115.42 |
| 2024-02-07 | Day 27 | $114.88 |
| 2024-02-08 | Day 28 | $114.35 |
| 2024-02-09 | Day 29 | $113.85 |
| 2024-02-12 | Day 30 | $113.38 |
| 2024-02-13 | Day 31 | $112.92 |
| 2024-02-14 | Day 32 | $112.48 |
| 2024-02-15 | Day 33 | $112.06 |
| 2024-02-16 | Day 34 | $111.66 |
| 2024-02-19 | Day 35 | $111.27 |
| 2024-02-20 | Day 36 | $110.90 |
| 2024-02-21 | Day 37 | $110.55 |
| 2024-02-22 | Day 38 | $110.21 |
| 2024-02-23 | Day 39 | $109.88 |
| 2024-02-26 | Day 40 | $109.57 |

# 9. Conclusion

The project's exploration into the utilization of a Long Short-Term Memory (LSTM) model has underscored its aptitude as a sophisticated analytical tool for financial time-series forecasting. Demonstrating a commendable capacity to glean insights and predict stock price movements, the LSTM model stands out as a compelling embodiment of machine learning’s transformative impact on financial analysis.

Looking ahead, to further enhance the model's robustness and predictive accuracy, integrating an ensemble method such as stacking would be prudent. Stacking, which strategically combines various machine learning algorithms, including the LSTM, can create a more powerful predictive model that leverages the unique strengths of each constituent method. By doing so, we can mitigate the biases inherent to any single-model approach and provide a more reliable forecast.

However, transitioning from a proof-of-concept to a business-ready solution entails additional considerations. The model must be interpretable and its decisions transparent to ensure compliance with financial regulations and to build trust among stakeholders. This clarity in the model's workings is pivotal in a domain where explanatory capacity is as valuable as predictive power.

Furthermore, in an ever-evolving financial landscape, the model cannot remain static. A regime of continuous monitoring, retraining with new data, and model tuning is essential to keep pace with market changes. This adaptive approach ensures the model's outputs remain accurate, relevant, and actionable.

By embracing these future directions and addressing the necessary business considerations, the LSTM model can transition from an analytical tool to an integral component of financial decision-making frameworks. Its integration into automated trading systems or as a decision-support tool for financial analysts and portfolio managers exemplifies its potential to revolutionize financial strategy formulation. As we continue to harness the capabilities of machine learning, the horizons of its application in finance will undoubtedly expand, paving the way for more informed and strategic market participation.