Capstone Three Project

Tweet Sentiment's Impact on Stock Price

Erin Amoueyan

June 2023

1 Introduction

In today's digital age, social media platforms have become an integral part of our lives, transforming the way we communicate, share information, and express opinions. Among these platforms, Twitter stands out as a microblogging platform that generates an enormous amount of real-time textual data. This wealth of information has not only revolutionized the way individuals interact but has also attracted the attention of researchers and businesses seeking insights into various domains, including finance and stock markets.

Understanding the impact of social media sentiment on stock returns has gained significant attention in recent years. The rise of sentiment analysis techniques, powered by Natural Language Processing (NLP) and machine learning, has allowed researchers to explore the relationship between investor sentiment expressed on Twitter and subsequent stock market movements. Sentiment analysis refers to the process of extracting subjective information from text, enabling the classification of sentiment as positive, negative, or neutral.

This report aims to delve into the captivating intersection of tweet sentiment and stock returns, utilizing NLP techniques and an LSTM (Long Short-Term Memory) model. The LSTM model is a type of recurrent neural network (RNN) that excels at capturing long-term dependencies in sequential data, making it particularly well-suited for analyzing the temporal dynamics inherent in financial markets.

The primary objective of this study is to investigate whether there exists a relationship between the sentiment expressed in tweets and subsequent stock returns. By applying sentiment analysis on a large corpus of tweets related to specific stocks or financial events, we aim to identify potential patterns and insights that may help investors make informed decisions. If a significant correlation between tweet sentiment and stock returns is found, it could potentially be leveraged as an additional tool in investment strategies or risk management processes.

Furthermore, this report seeks to provide an overview of the NLP techniques and LSTM model employed in the analysis. NLP techniques, such as tokenization, word embeddings, and sentiment classification, will be utilized to preprocess and analyze the textual data. The LSTM model will be trained on the sentiment-labeled tweet data, allowing us to capture the sequential dependencies and patterns within the tweets, ultimately establishing a predictive link between tweet sentiment and subsequent stock returns.

It is important to note that this study does not claim to predict stock returns solely based on tweet sentiment, as financial markets are influenced by numerous factors. Rather, it aims to explore the relationship between tweet sentiment and stock returns to enhance our understanding of the potential impact of social media sentiment on market dynamics.

2 Data Wrangling

Data wrangling played a pivotal role in preparing the dataset for the analysis of the impact of tweet sentiment on stock returns. The initial dataset comprised tweet data for 101 different stocks obtained from Kaggle. However, due to the imbalanced distribution, with the top 30 stocks accounting for 93% of the data, the focus was narrowed down to these 30 stocks.

To ensure data integrity, various tasks were performed during the data wrangling process. Duplicate entries were removed, ensuring that each tweet was unique. Missing data was addressed through strategies like imputation or removal, preventing biased or incomplete results. Data types were appropriately converted for consistency, and unnecessary features were eliminated to simplify the analysis. Additionally, an auxiliary dataset was utilized to find the company's tickers for each stock, enhancing interpretability.

To augment the tweet data, stock price information was obtained from Yahoo Finance using their API. This additional dataset provided the adjusted close prices of the stocks for the selected time period from 2017-01-31 to 2018-10-31. By calculating the stock returns based on the adjusted close prices, the relationship between tweet sentiment and subsequent stock returns could be explored. The incorporation of Yahoo Finance API data added a crucial dimension to the analysis, enabling a more comprehensive examination of the impact of tweet sentiment on stock performance.

Histogram analysis was performed to gain insights into the distributions of various features within the dataset. This visual representation helped identify any potential skewness or patterns, aiding in understanding the characteristics of the data.

Figure 1 illustrates the stock prices of the first five companies in the dataset during the selected period.

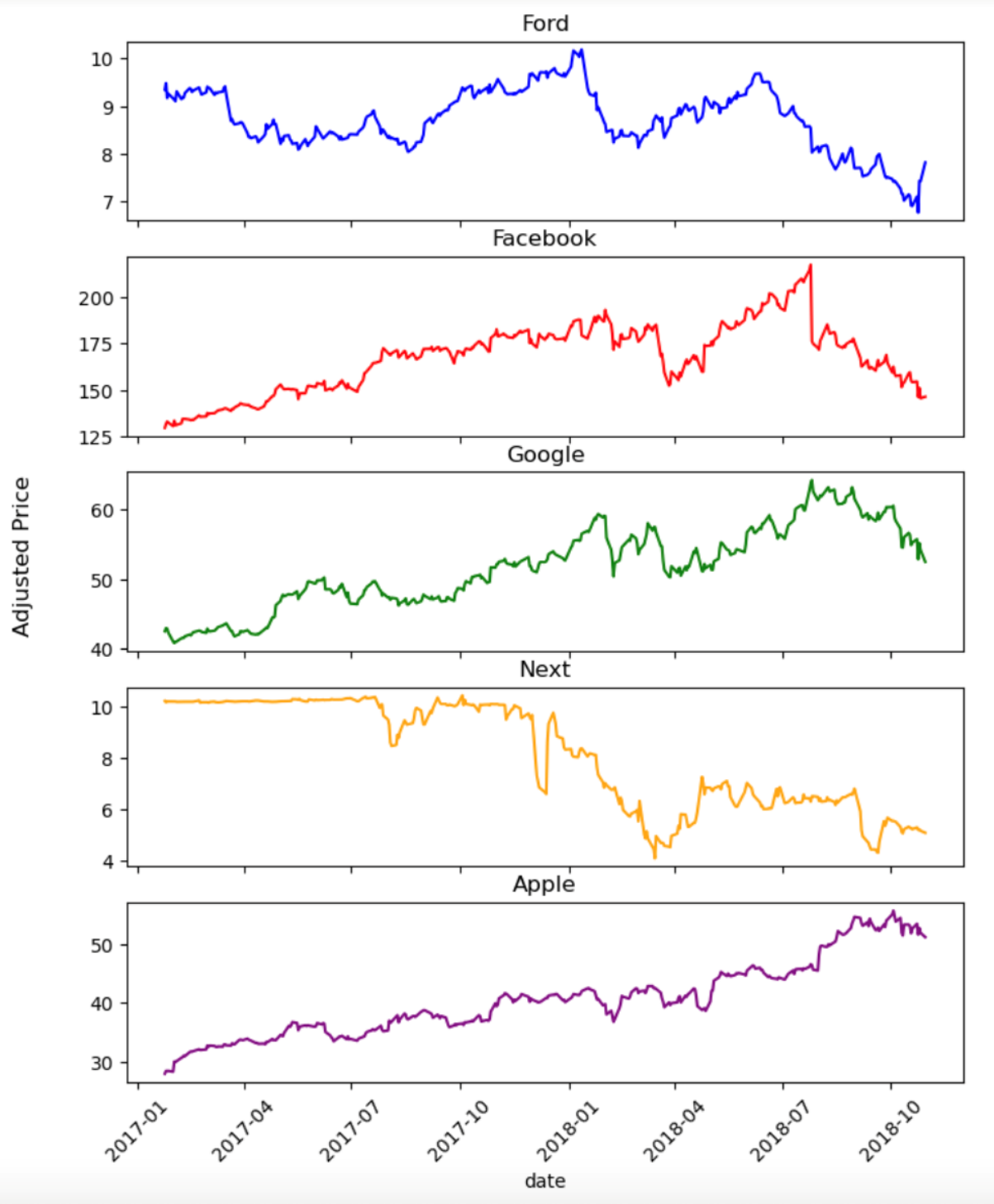

Figure 1. Stock Prices of Five Companies from Yahoo Finance

The stock returns for the first five companies in the dataset were computed based on the adjusted close price obtained from Yahoo Finance. Figure 2 displays these stock returns,

providing insights into the performance and changes in value for these companies during the selected period.

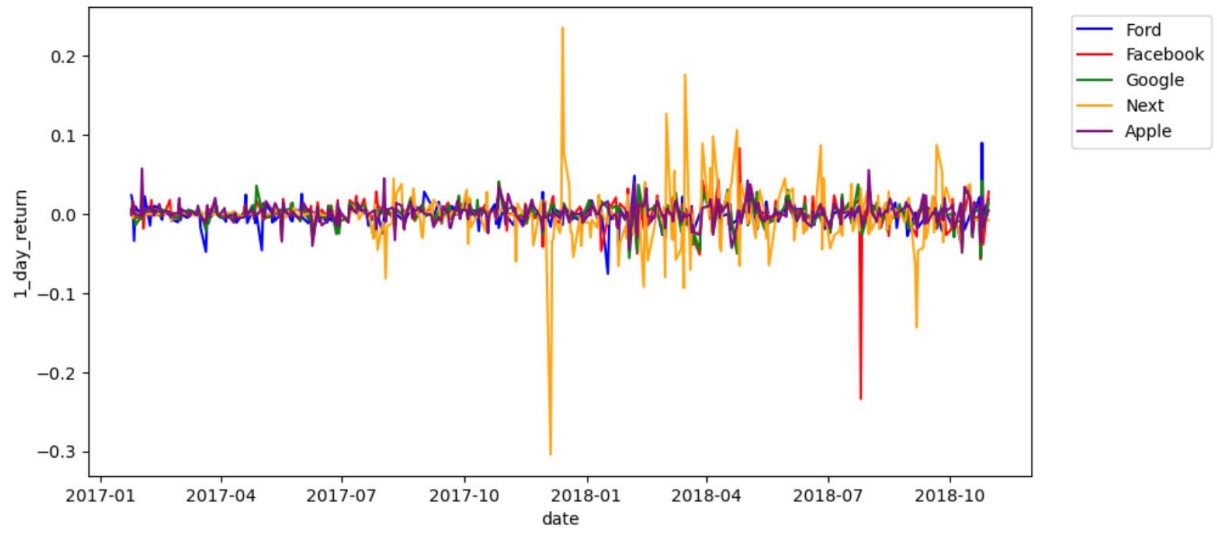

Figure 2. Stock Returns of Five Companies Calculated from Yahoo Finance's Adjusted Close Price

Upon merging the information from Yahoo Finance with the tweet data, we present Figure 3, which illustrates the graph of one-day returns.

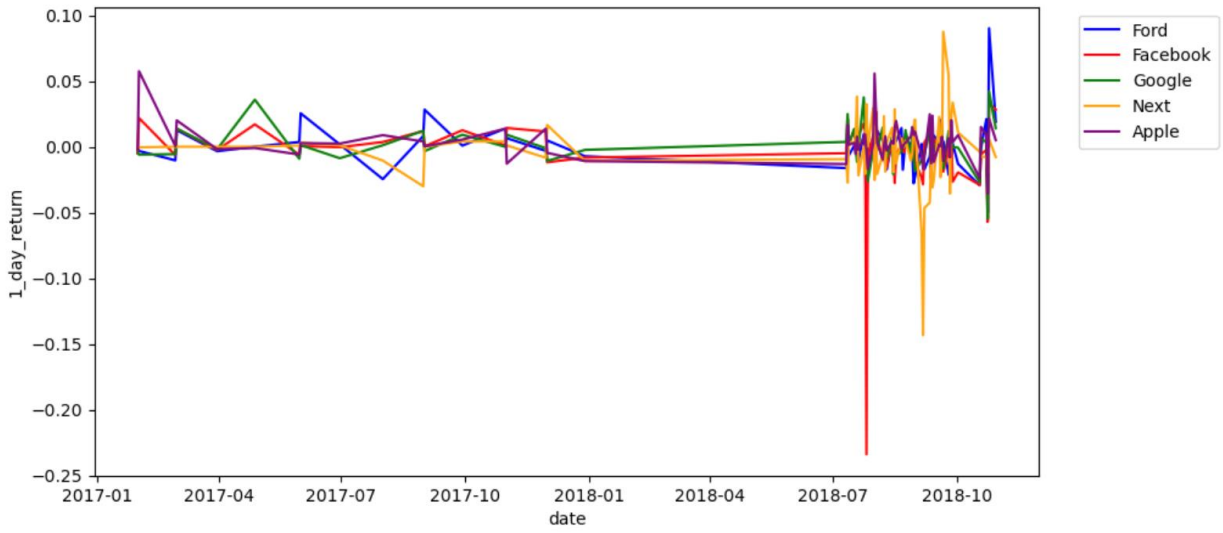

Figure 3. One-Day Return Graph after Merging Yahoo Finance Information with Tweet Data

3 Preprocessing and Modeling

3.1 Stock Sentiment Analysis

An LSTM model was developed for sentiment analysis using the cleaned tweet data.
Prior to training the LSTM model, we performed preprocessing steps specific to NLP tasks. This included tokenization, removing stop words, handling emoticons, and applying techniques such as stemming or lemmatization to reduce the dimensionality of the text data, sequency, and padding. These preprocessing steps were crucial in preparing the tweet data for input into the LSTM model.
“Tokenization” was used to split texts into individual tokens (words or characters). This step builds the vocabulary based on the unique tokens in the text.
“Sequence” converts the text data into sequences of integers. Each word in the text is replaced with its corresponding integer index based on the vocabulary built by the tokenizer.
Then the maximum length of the sequences was calculated. This step is important because neural networks typically require input sequences of the same length. The maximum length of the sequences determines the length to which all sequences will be padded or truncated. Padding is done by adding zeros at the beginning or end of sequences to make them of equal length.
Next, we had to come up with a method to assign sentiment labels (e.g., positive, negative, neutral) to each tweet based on their content. For this purpose, SentimentIntensityAnalyzer, a built-in tool in the Natural Language Toolkit (NLTK) library was used. Sentiment scores greater than and equal to 0.5 were defined as “positive”, less than and equal to -0.5 were defined as “negative”, and all other scores between -0.5 and 0.5 were defined as “neutral.”
Next, we have done some basic statistics about the dataset to check if the dataset is balanced or not (equal number of all labels). The analysis showed that 46%, 22%, and 32% of the data were labeled as positive, negative, and neutral respectively.

3.1.1 Encoding Labels and Making Train-Test Splits

LabelEncoder() from sklearn.preprocessing was used to convert the labels (‘positive’, ‘negative’, ‘neutral’) into 1’s, -1’s, and 0’s respectively.
Finally, we split the dataset into train and test parts using train_test_split from sklearn.model_selection. We use 80% of the dataset for training and 20% for testing.

3.1.2 Building the LSTM model

A Keras sequential model was constructed consisting of the following layers:

1. Embedding Layer: This layer converts each word in the sentence into a fixed-length dense vector of dimension 100. The vocabulary size is used as the input dimension, and the output dimension is set to 100. As a result, each word in the input is represented by a 100-dimensional vector.
2. Bidirectional LSTM Layer: This layer consists of 64 LSTM units that process the input sequence in both forward and backward directions. This bidirectional nature allows the model to capture contextual information from both past and future words in the sentence.
3. Dense Layer: This fully connected layer comprises 32 units with a softmax activation function. It performs nonlinear transformations on the input data, enabling the model to learn complex patterns and representations.
4. Output Dense Layer: This dense layer consists of 3 units and uses a softmax activation function which is commonly used in multi-class classification and converts the raw output values into a probability distribution over multiple classes. Each value in the output of the softmax function is interpreted as the probability of membership for each class.

The model is compiled with categorical cross-entropy loss and “adam” optimizer. Since we have a multi-class classification problem and the labels are provided as integers, sparse categorical cross-entropy loss was used. The Adam optimizer uses stochastic gradient descent to train deep learning models, and it compares each of the predicted probabilities to the actual class label. Accuracy was used as the primary performance metric. The model summary can be seen below :

```
model = Sequential()
model.add(Embedding(input_dim=len(tokenizer.word_index) + 1 , output_dim=100,
input_length=max_sequence_length))
model.add(LSTM(units=128, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(units=64, activation='softmax'))
model.add(Dense(units=3, activation='softmax'))
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])
print(model.summary())
```

```
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 embedding_2 (Embedding)      (None, 26, 100)           13261300

 lstm_2 (LSTM)                (None, 128)               117248

 dense_4 (Dense)              (None, 64)                8256

 dense_5 (Dense)              (None, 3)                 195

=================================================================
Total params: 13,386,999
Trainable params: 13,386,999
Non-trainable params: 0
_________________________________________________________________
None
```

3.1.3 Model Training and Evaluation

The model is trained for 15 epochs, with batch size of 256. The model is evaluated by calculating its accuracy.

Model performance was evaluated by predicting the labels on the test set. The accuracy of prediction on the test set comes out to be 94.05%.

Figure 4 shows how the training and validation loss change over the training epochs. The decreasing loss indicates that the model is learning and making progress in minimizing the error between the predicted and actual sentiment labels. Monitoring the loss graph helps in assessing the convergence of the model and ensuring that it is improving its sentiment prediction capabilities.

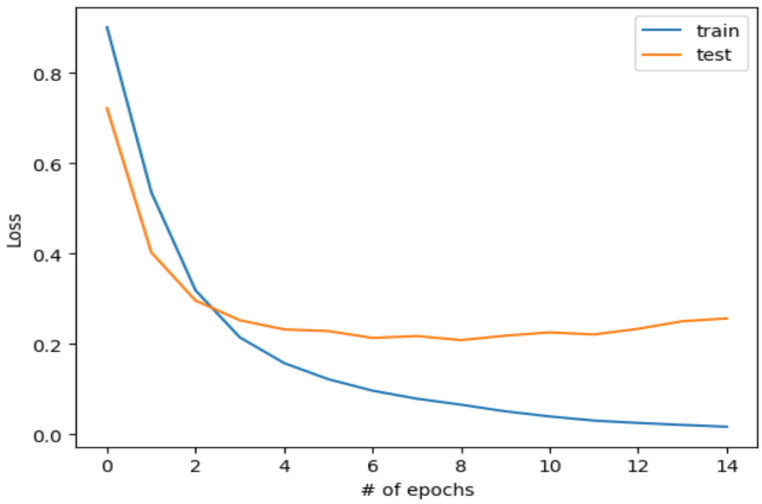

Figure 4. Training and validation loss change over the training epoch

Figure 5 illustrates the performance of the LSTM model in terms of correctly predicting the sentiment of stock-related tweets. As the training progresses, the accuracy increases, indicating that the model becomes more proficient in differentiating positive, negative, and neutral sentiments expressed in the tweets. A higher accuracy value implies a higher level of confidence in the model's sentiment predictions.

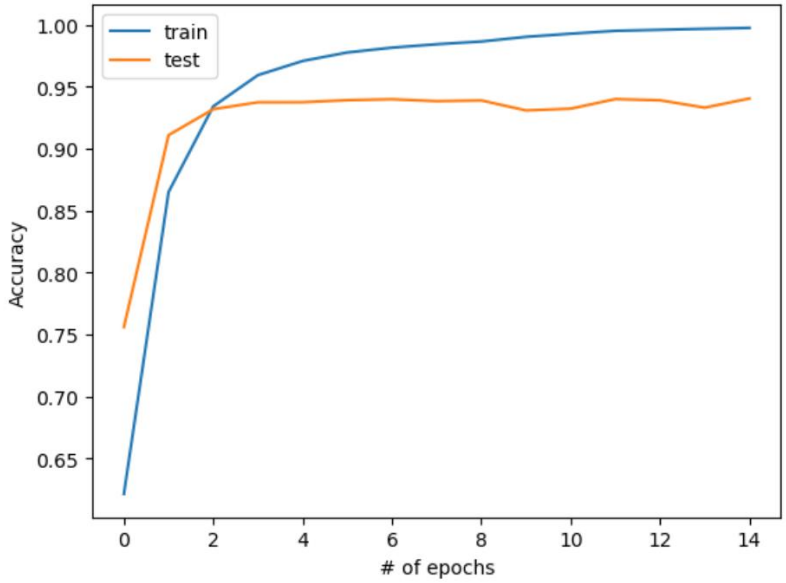

Figure 5. Performance of the LSTM model in terms of predicting the sentiment of tweets

The output of the model is an array that includes the probability of each class (negative, neutral, and positive) for each sentiment. The higher probability indicates the sentiment label. A new column “predicted_label” was added to the dataframe which includes the sentiment label based on the output of the LSTM model. Final dataset was saved as “sentiment_labels.csv.” The first few rows of the data are shown in Figure 6.

	date	adj_close	volume	ticker	1_day_return	predicted_label
0	2017-01-31	28.308922	196804000	AAPL	-0.002307	0
1	2017-01-31	28.308922	196804000	AAPL	-0.002307	1
2	2017-01-31	28.308922	196804000	AAPL	-0.002307	1
3	2017-01-31	28.308922	196804000	AAPL	-0.002307	2
4	2017-01-31	28.308922	196804000	AAPL	-0.002307	0

Figure 6. Updated dataframe with stock tickers and return values

3.2 Time-Series Forecasting: Predicting Stock Prices

3.2.1 Data preparation

To prepare the dataset for the LSTM, we need to follow a few steps:

- Framing the dataset as a supervised learning problem: In this step, we structure the dataset in a way that it can be used for supervised learning. We'll formulate it as predicting the stock price at the current hour (t), using the stock price and sentiment label from the previous time step.
- Normalizing the input variables: It's essential to normalize the input variables to ensure they are on a similar scale. This step helps in improving the performance and convergence of the LSTM model.

First, the “sentiment_labels.csv” dataset is loaded, and the data is grouped by the stock tickers and the average of the sentiment labels and adjusted close price. Then all features are normalized, and the dataset is transformed into a supervised learning problem.

We have used 10 days of data as input. We have 10 * 2 + 2 columns in our framed dataset now. We will take 10*2 or 20 columns as input for the observations of all features across the previous 10 days. We will take just the "adj_close" variable as output at the following hour.

Finally, we reshaped the inputs (X) into the 3D format required by LSTMs, which is [samples, timesteps, features]. With the inputs prepared, we can now define and fit our LSTM model.

3.2.2 Building the LSTM model

The LSTM model was defined as below:

- Input Feature: The input shape was 10 time steps with 2 features.
- Output Feature: The output feature was set as "adj_price." This adjustment allows the LSTM model to predict the stock price for the next day based on the input data.
- Model Architecture: The LSTM model was defined with 400 neurons in the first hidden layer, 128 neurons in the second hidden layer, 64 neurons in the third hidden layer, and 1 neuron in the output layer for predicting stock price.
- Loss Function: The Mean Squared Error (MSE) was utilized as the chosen loss function. By employing MSE, the model aims to minimize the average squared difference between the predicted stock price and the actual observed values.
- Optimizer: Adam optimizer was utilized to efficiently update the weights and biases of the model during the training process.

- Training Parameters: The model was trained for 200 epochs, representing the number of times the model iterated through the entire training dataset during training. A batch size of 64 was used, indicating that the model updates its weights and biases after processing every 64 data samples.
- Activation function= Linear activation function was used to predict continuous values of the output.

The model summary is shown below:

```
model = Sequential()
model.add(LSTM(400, input_shape=(train_X.shape[1], train_X.shape[2])))
model.add(Dropout(0.3))
model.add(Dense(128))
model.add(Dropout(0.3))
model.add(Dense(64))
model.add(Dropout(0.3))
model.add(Dense(1, activation ='linear'))
model.compile(loss='mse', optimizer='adam', metrics= ['mse', 'mae']
history = model.fit(train_X, train_y, epochs=100, batch_size=16,
validation_data=(test_X, test_y), verbose=0, shuffle=False)
```

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 400)	644800
dropout (Dropout)	(None, 400)	0
dense (Dense)	(None, 128)	51328
dropout_1 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 64)	8256
dropout_2 (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 1)	65

Total params: 704,449
Trainable params: 704,449
Non-trainable params: 0

3.2.3 Model Training and Evaluation

Figure 7 illustrates how the training and validation loss change over the training epochs. The decreasing loss indicates that the model is learning and making progress in minimizing the error between the predicted and actual sentiment labels.

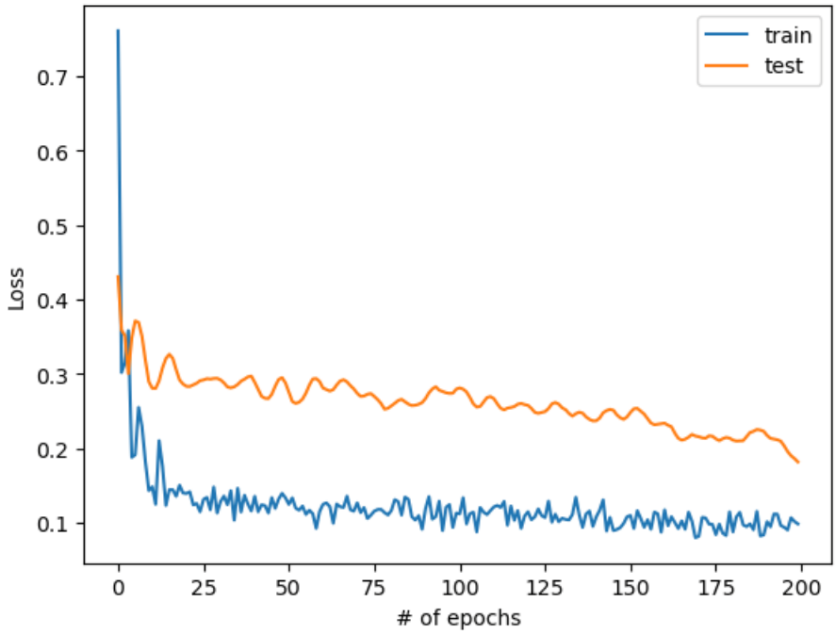

Figure 7. Training and validation loss change over the training epoch

To evaluate the model, we combined the forecast with the test dataset and proceeded to invert the scaling applied to both the forecast and the test dataset. This step brings them back to their original scale, allowing us to compare the predicted stock prices with the actual stock prices. Once we have the forecasts and actual values in their original scale, we can calculate an error score for the model. In this case, we use the Root Mean Squared Error (RMSE) as our error metric. The RMSE provides an error measurement in the same units as the variable itself, enabling us to assess the model's performance accurately.

The stock price prediction graphs for the top 5 companies are shown in Figures 8-12.

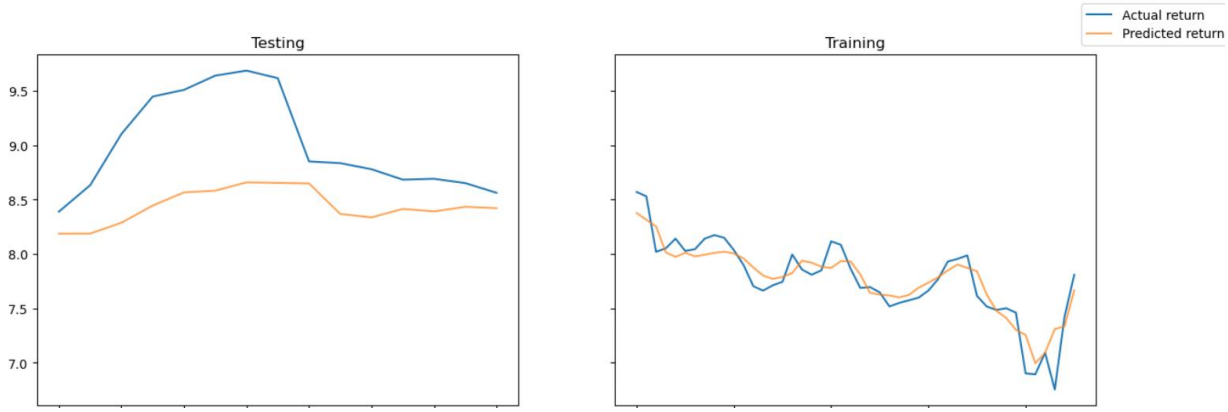

Figure 8. Stock Price Prediction for Ford Motor- Train RMSE = 0.153, Test RMSE = 0.663

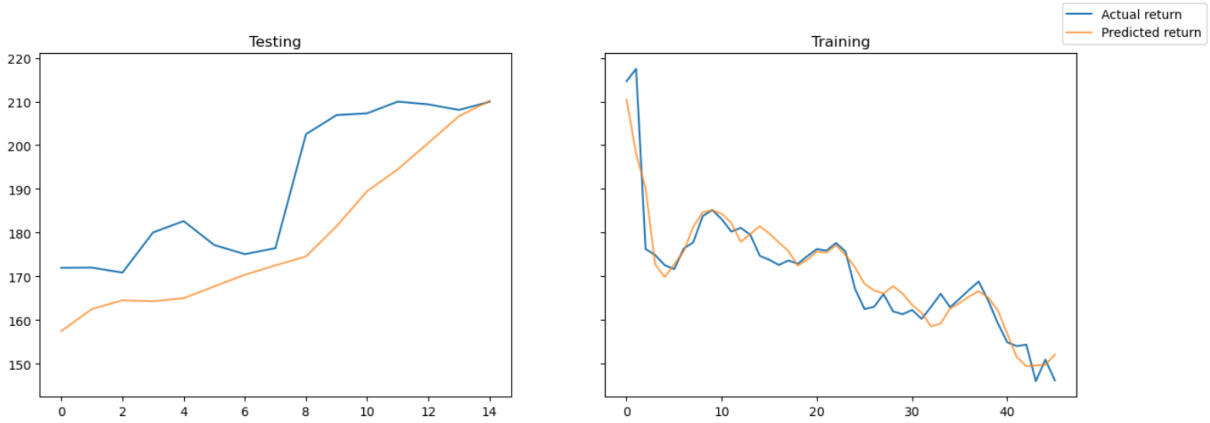

Figure 9. Stock Price Prediction for Facebook - Train RMSE = 4.758, Test RMSE = 14.356

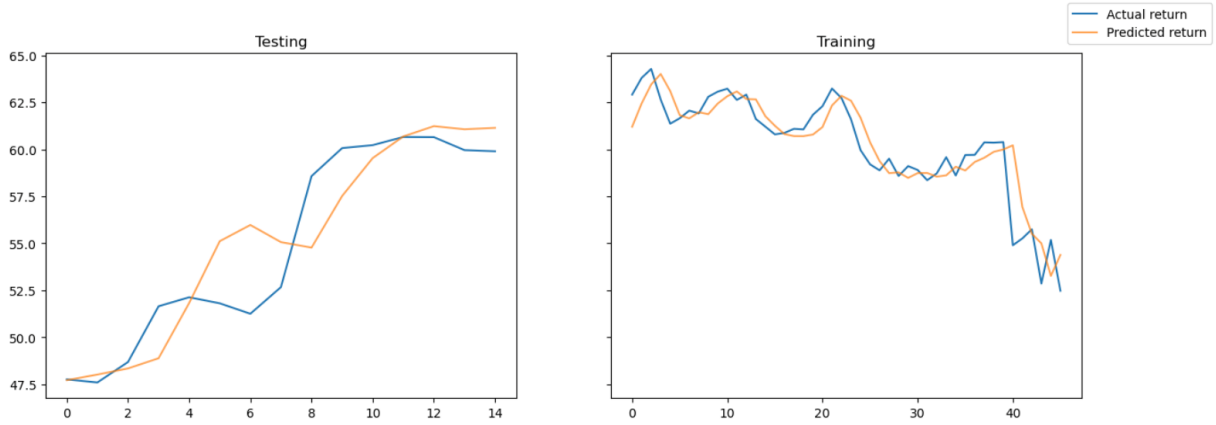

Figure 10. Stock Price Prediction for Google - Train RMSE = 1.234, Test RMSE = 2.183

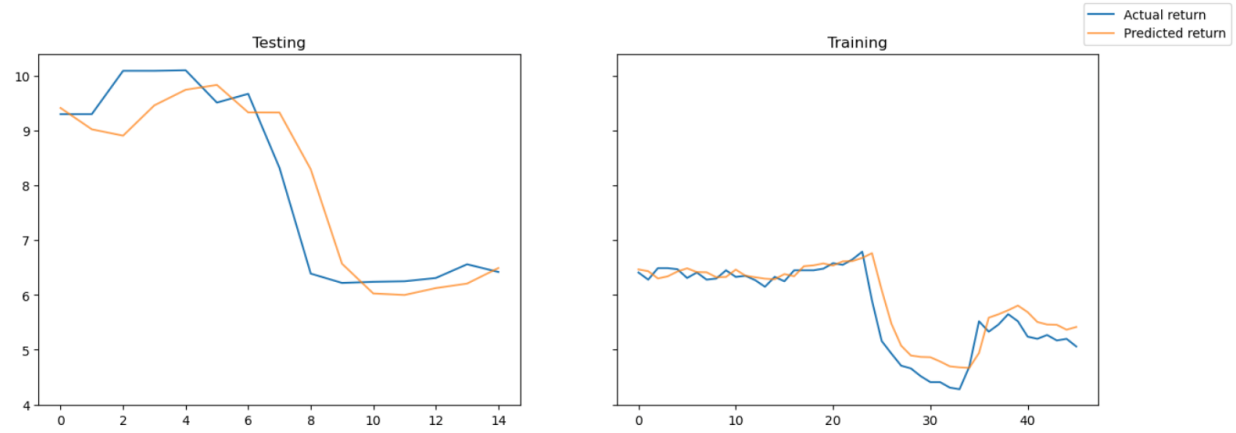

Figure 11. Stock Price Prediction for Next - Train RMSE = 0.303, Test RMSE = 0.696

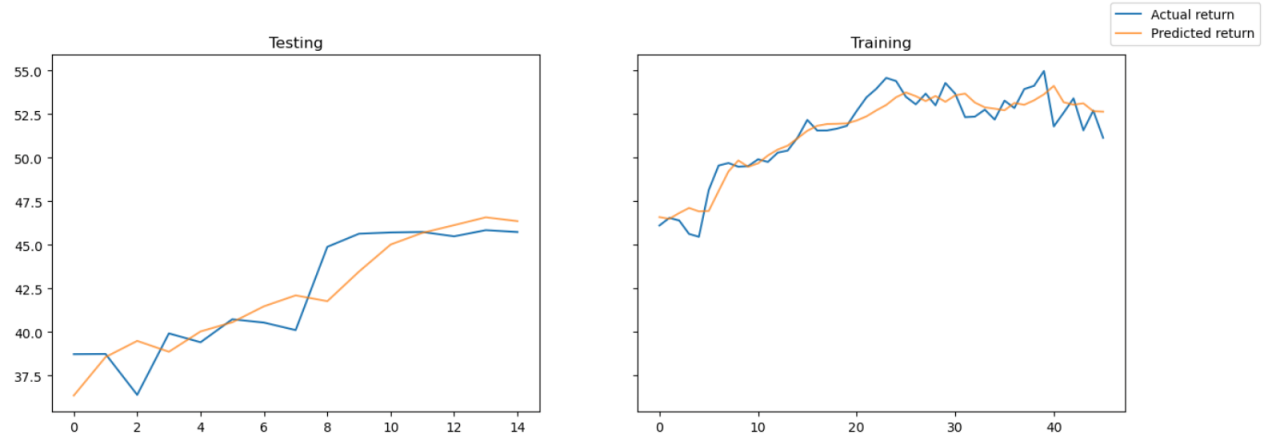

Figure 12. Stock Price Prediction for Apple - Train RMSE = 0.872, Test RMSE = 1.588

While the exact price points from our predicted price weren't always close to the actual price, our model did still indicate overall trends such as going up or down.

Next, we have changed our output feature from stock price to 1-day stock return to see how the accuracy of the prediction will change compared to the stock price prediction.

3.3 Time-Series Forecasting: Predicting Stock Returns

An LSTM model was developed with some notable changes compared to the previous section. The key modifications are as follows:

- Output Feature: The output feature was altered to "1-day return."

- Model Architecture: The LSTM model was defined with 300 neurons in the first hidden layer and 1 neuron in the output layer.
- Loss Function: The MSE was utilized as the chosen loss function. By employing MSE, the model aims to minimize the average squared difference between the predicted 1-day return and the actual observed values.
- Training Parameters: The model was trained for 150 epochs, and a batch size of 10, indicating that the model updates its weights and biases after processing every 10 data samples.

The model summary is shown below:

```
model = Sequential()
model.add(LSTM(300, input_shape=(train_X.shape[1], train_X.shape[2])))
model.add(Dropout(0.3))
model.add(Dense(1, activation= 'linear'))
model.compile(loss='mse', optimizer='adam', metrics= ['mse','mae'] )
history = model.fit(train_X, train_y, epochs=150, batch_size=10,
validation_data=(test_X, test_y), verbose=1, shuffle=False)
```

```
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 lstm_9 (LSTM)                (None, 300)               363600

 dropout_11 (Dropout)         (None, 300)               0

 dense_11 (Dense)             (None, 1)                 301

=================================================================
Total params: 363,901
Trainable params: 363,901
Non-trainable params: 0
_________________________________________________________________
```

3.3.1 Model Training and Evaluation

Figure 13 shows how the training and validation loss change over the training epochs. The decreasing loss indicates that the model is learning and making progress in minimizing the error between the predicted and actual sentiment labels.

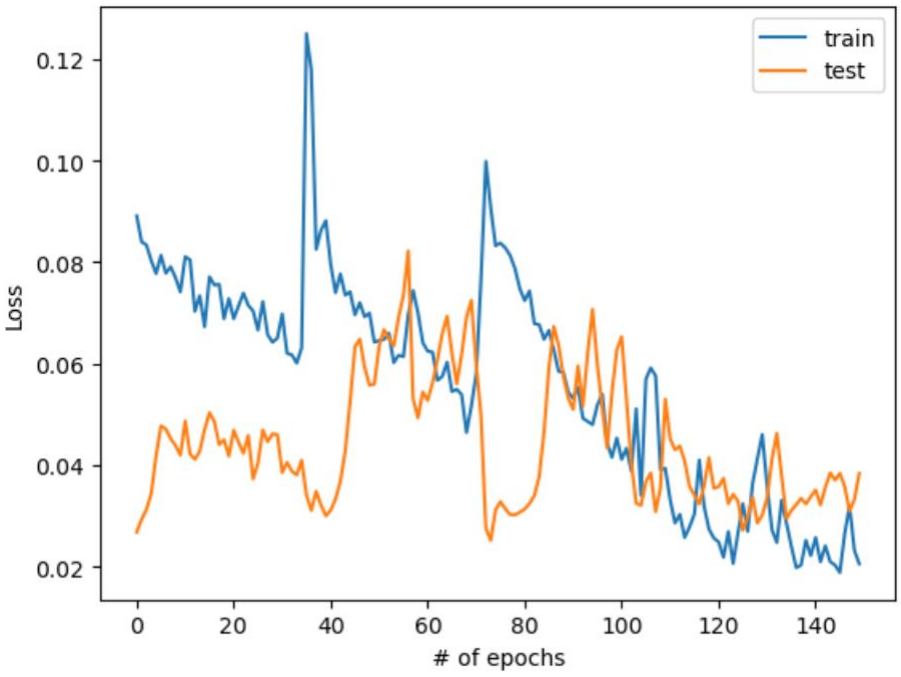

Figure 13. Training and validation loss change over the training epoch

The stock price prediction graphs for the top 5 companies are shown in Figures 14- 18.

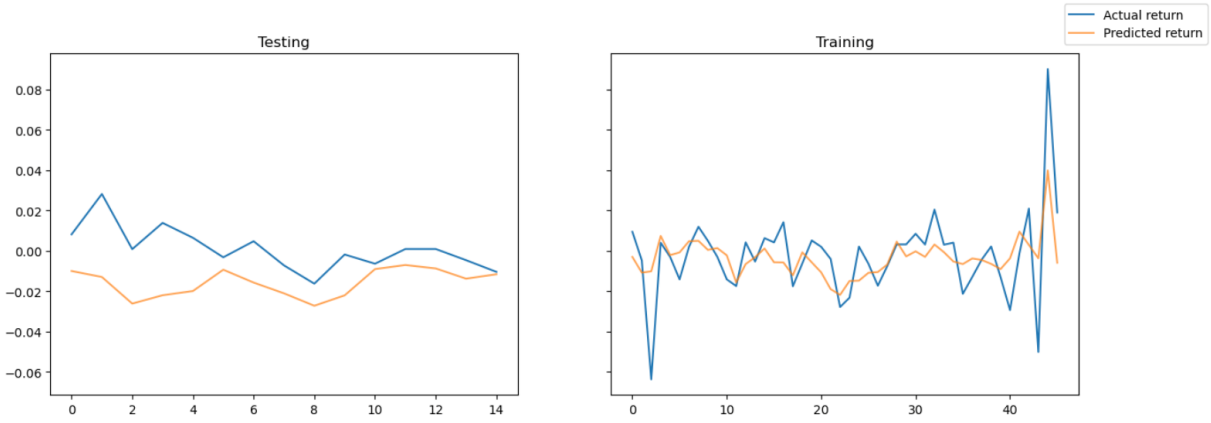

Figure 14. Stock Price Prediction for Ford - Train RMSE = 0.016, Test RMSE = 0.020

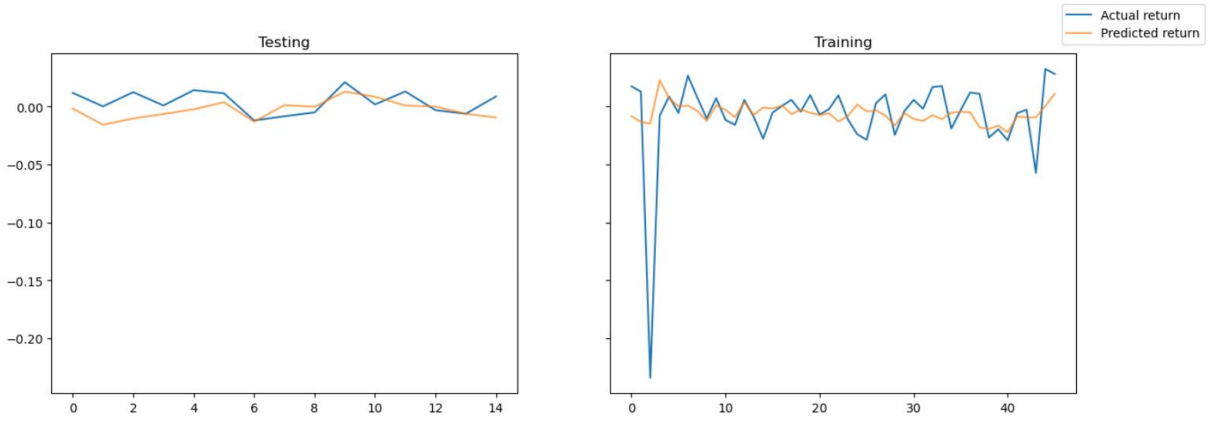

Figure 15. Stock Price Prediction for Facebook - Train RMSE = 0.037, Test RMSE = 0.012

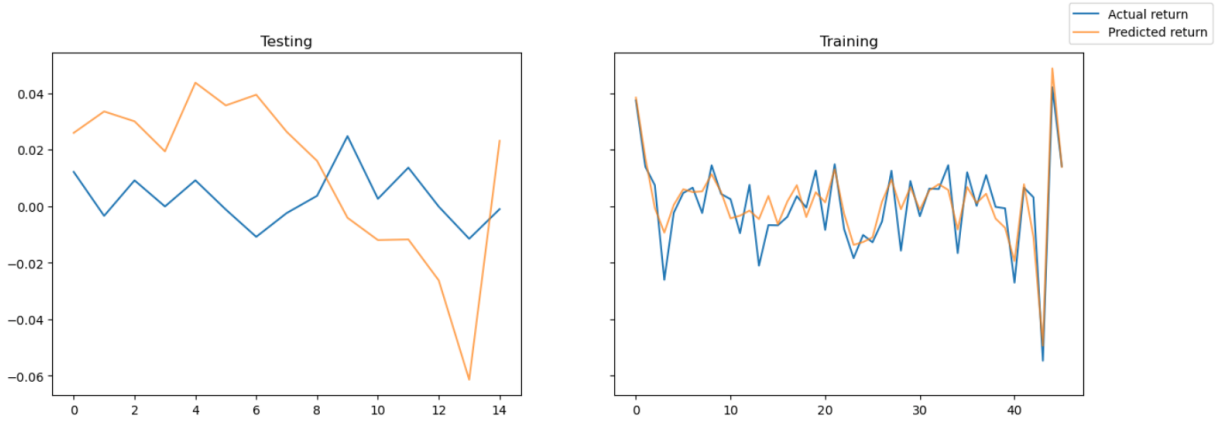

Figure 16. Stock Price Prediction for Google - Train RMSE = 0.007, Test RMSE = 0.030

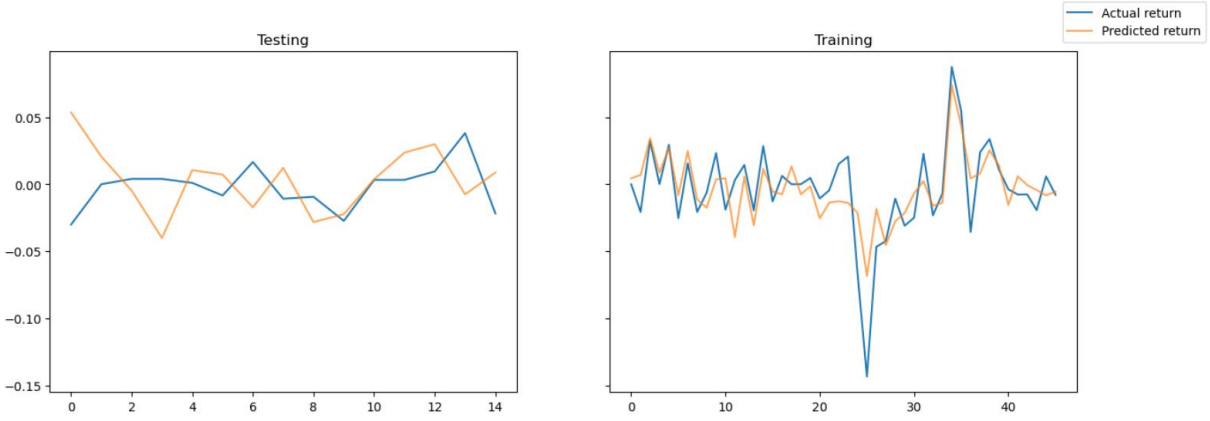

Figure 17. Stock Price Prediction for Next - Train RMSE = 0.021, Test RMSE = 0.032

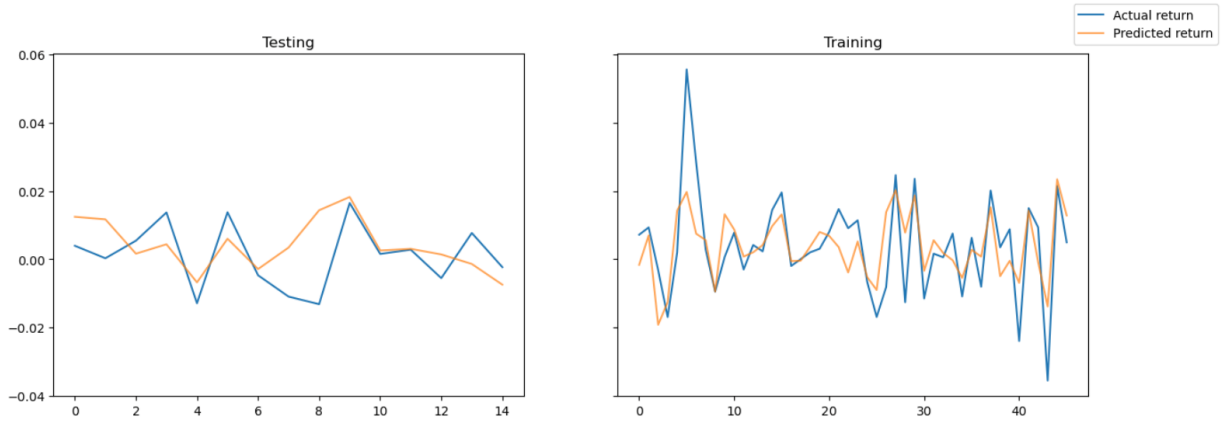

Figure 18. Stock Price Prediction for Apple - Train RMSE = 0.011, Test RMSE = 0.010

4 Conclusion

In this evaluation, we examined the performance of LSTM models in predicting the price and return of various stocks. Our analysis reveals that the accuracy of the predictions varies across different stocks, with some demonstrating higher accuracy in price prediction, while others exhibiting better accuracy in return prediction. However, in general, the LSTM models were successful in capturing the overall trend of both price and return for the majority of the evaluated stocks. This underscores the power and efficacy of LSTM models in predicting time series data.

By effectively leveraging the LSTM architecture, we were able to capture significant patterns and trends in the stock market data. The models showcased their capability to comprehend the underlying dynamics of the time series and provide reasonably accurate predictions. Such predictive capabilities are invaluable in making informed investment decisions and identifying potential market trends.

It is worth noting that there is room for further enhancement in the performance of these LSTM models. In addition to gathering more data, refining the models through parameter adjustments, hyperparameter optimization, and exploring alternative architectures remains crucial. Incorporating additional features or considering different data preprocessing techniques may also contribute to enhanced performance and more robust predictions.

Further evaluation and analysis will be conducted to fine-tune the models, explore additional avenues for improvement, and leverage the benefits of an expanded dataset, enabling us to harness the full potential of LSTM models in predicting stock market dynamics.