STOCK PRICE PREDICTION USING RECURRENT NEURAL NETWORKS

A Paper
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Israt Jahan

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Program:
Software Engineering

June 2018

Fargo, North Dakota

North Dakota State University

Graduate School

Title

STOCK PRICE PREDICTION USING RECURRENT NEURAL NETWORKS

By

Israt Jahan

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Kendall E. Nygard

Chair

Oksana Myronovych

Benjamin D. Braaten

Approved:

07/06/2018

Date

Kendall E. Nygard

Department Chair

ABSTRACT

The stock market is generally very unpredictable in nature. There are many factors that might be responsible to determine the price of a particular stock such as the market trend, supply and demand ratio, global economy, public sentiments, sensitive financial information, earning declaration, historical price and many more. These factors explain the challenge of accurate prediction. But, with the help of new technologies like data mining and machine learning, we can analyze big data and develop an accurate prediction model that avoids some human errors. In this work, the closing prices of specific stocks are predicted from sample data using a supervised machine learning algorithm. In particular, a Recurrent Neural Network (RNN) algorithm is used on time-series data of the stocks. The predicted closing prices are cross checked with the true closing price. Finally, it is suggested that this model can be used to make predictions of other volatile financial instruments.

ACKNOWLEDGMENTS

First of all, I would like to thank Almighty for His endless blessings on me and my family.

Secondly, I would like to express my sincere thanks to Dr. Kendall Nygard for providing immense guidance, valuable suggestion, strong support, inspiration, and supervision throughout my studies and research at North Dakota State University.

I am really grateful to Dr. Oksana Myronovych and Dr. Benjamin Braaten for giving me inspiration and guidance to achieve my goal and also for their valuable time to be members of my supervisory committee. I appreciate their time, kindness, and valuable support.

Finally, I would like to express a special thanks to my husband Dr. Sayeed Sajal and my family for their support and inspiration.

DEDICATION

To my family, especially my beloved husband Dr. Sayeed Sajal.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

LIST OF SYMBOLS

H_0 ..... Null Hypothesis

H_a ..... Alternative Hypothesis

CHAPTER 1. INTRODUCTION

Any kind of prediction is a difficult task, especially where the future is very volatile. The stock market is highly volatile and unpredictable by nature. Therefore, investors are always taking risks in hopes of making a profit. People want to invest in the stock market and expect profit from their investments. There are many factors that influence stock prices [1–3], such as supply and demand, market trends, the global economy, corporate results, historical price, public sentiments, sensitive financial information, popularity (such as good or bad news related to a company), all of which may result in an increase or decrease in the number of buyers etc. Even though one may analyze a lot of factors, it is still difficult to achieve a better performance in the stock market and to predict the future price.

Predicting the price of a specific stock one day ahead is, by itself, a very complicated task. In this study, next day stock prices are predicted for each of the individual days of one whole year. For each day, comparisons are made with the actual prices to validate the model. In this research, the two questions below are answered.

1. How can we predict day-ahead stock prices using only historical price data?

2. How can we validate the results for the developed model?

In this study, a Recurrent Neural Network with Long Short-Term Memory (LSTM) is used as the machine learning technique to analyze and predict future stock prices based on historical prices.

CHAPTER 2. LITERATURE REVIEW AND BACKGROUND

2.1. Literature Review

With the advancement of new technology and statistical tools, many scholars have explored ways to predict stock prices. In 1997, prior knowledge and a neural network were used to predict stock price [4]. Later, a genetic algorithm approach and a support vector machine was introduced to predict stock prices [5, 6]. Lee introduced stock price prediction using reinforcement learning [7]. In 2008, Chang used a TSK-type fuzzy rule-based system for stock price prediction [8]. In 2009, Tsai used a hybrid machine learning algorithm to predict stock prices [9]. Over time, the scholars predicted the stock prices using different kinds of machine learning algorithms such as deep learning [10, 11], extreme machine learning [12] and applied econometric approach using machine learning [13]. In 2015, AM Rather proposed a hybrid model composed of two linear models and one non-linear model. The non-linear model was a recurrent neural network. They found though this approach that it was the non-linear model, the recurrent neural network, that gave a satisfactory prediction of stock prices [14].

In 2018, popular machine learning algorithms such as pattern graphs [15], convolutional neural networks [16], artificial neural networks [17], recurrent neural networks [18] were used to predict stock prices.

2.2. Machine Learning

Machine Learning is a class of techniques that can be used to analyze data or information in order to generalize and observe the patterns of that data or information. To predict the future value or behavior from those observations or patterns, it will then iteratively learn from data, unlike typical computer programs. The purpose of machine learning is to program computers to use sample data as a

past experience or model, and use the patterns of this data to predict the future based on that data. Machine Learning does not only deal with database problems, it is also an application of artificial intelligence (AI) too. It helps us to find solutions to various problems in image recognition, voice or speech recognition, face recognition, biometrics authentication, medical diagnoses, agriculture, economics, computer networks, robotics, etc. [19,20]. Machine learning opens up a new avenue of computer science that often uses statistical techniques to give computers the ability to "learn" with data, without being explicitly programmed.

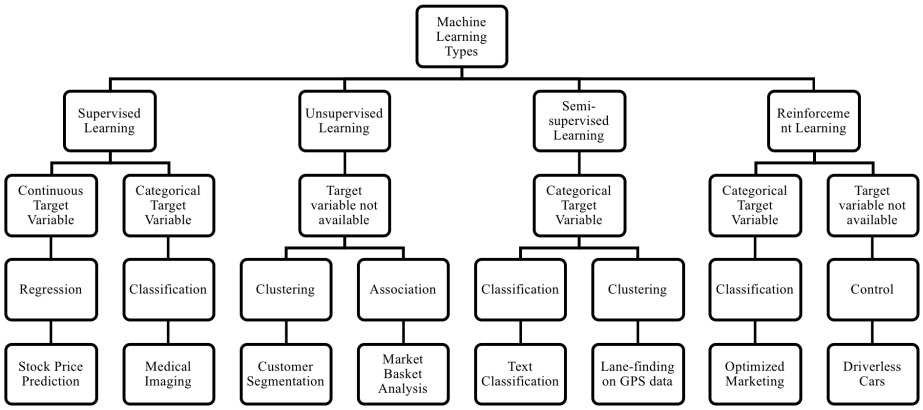

Figure 1. Types of Machine Learning

2.2.1. Types and Classification of Machine Learning

Machine learning algorithms can be categorized according to the types of data [21–23], whether they give feedback or a learning signal. Four types of machine learning are described below.

1. Supervised machine learning - Supervised Learning uses labeled data to predict future label data or events with some given features. If the label data is continuous then it is called a regression problem and if label data is categorical it is a classification problem. Sometimes it is called binary classification.

2. Unsupervised machine learning - Unsupervised learning uses information that is neither classified nor labeled. This is called non-structure data. We can group

similar types of things together from the data. We can call these either clustering or association problems.

3. Semi-supervised machine learning - When the data are a combination of supervised and unsupervised learning, mostly a large amount of unlabeled data and a smaller amount of label data, then the problem falls under the Semi-supervised learning category. Under this method, the system is able to considerably improve learning accuracy.

4. Reinforcement machine learning - Reinforcement learning interacts with environments and works through trial and error to find which actions provide the greatest rewards.

2.2.2. Advantages and Disadvantages

Machine learning has some advantages and disadvantages. Those are described below.

- Advantages of Machine Learning
  - Machine learning applications are widely used in sectors such as financial, banking, healthcare, rental, education, economics, agriculture etc.
  - Facebook and Google are using machine learning to push relevant advertisements based on users past search behavior.
  - In dynamic or uncertain environments, machine learning can handle multi-dimensional and multi-variety of data.
  - Machine learning allows time cycle reduction and efficient utilization.
  - Machine learning includes processes that can lead to automation of tasks by increased uses of algorithms in different applications.
  - In machine learning, there are tools available to provide continuous quality improvements in large and complex process environments.

- Disadvantages of Machine Learning
  - Acquisition is the major challenge of Machine Learning, for this data needs to be processed based on different algorithms before providing it as input to its respective algorithms.
  - Interpretation of results is also another major challenge to determine the effectiveness of machine learning.
  - Different machine learning techniques are tried based on which action or algorithm needs to be used and when it is to be used.
  - Machine learning needs lots of data.
  - A fixed pattern of data or historical data is another limitation of machine learning. As it works with statistical truths, it will be difficult to prove or predict anything without historical data.
  - Error diagnosis and correction is another limitation of machine learning.

2.3. Theory

2.3.1. Theory of Machine Learning

In the old days, only companies had data. But nowadays, with the arrival of things like the personal computer, high bandwidth networks, sensors, and social media, we generate lots of data every day and in every workplace. Whether we buy or sell a product, rent a movie, withdraw money from a bank, use a credit card, post on social media, write blogs etc. we create data. For a variety of purposes, we analyze those data in order to meet our goals. The technique used to analyze and manipulate the data is Machine Learning. The objective of machine learning is to program computers to use sample data as a past experience or model (called historical data) to obtain patterns of data for predicting the future based on the historical data [24].

Suppose a supermarket chain sells thousands of goods to millions of customers. To maximize sales, the supermarket chain tries to predict which customer will buy which product. Most customers find those products that meet their needs. Sometimes customer behavior changes due to geographical location and time. For example, people buy ice-cream in summer and spices for gardening in winter. When they buy beer they also buy or there is the possibility to buy chips or nuts. So, there are patterns in data. To predict the combinations of products that customers buy together and to arrange the store aisles accordingly, a computer needs an algorithm, a step by step set of instructions, to get only the desired output from the given input. There are various algorithms to solve the same problem, and from those we need to find the one that makes the best prediction with the least number of steps.

Because the main task in machine learning is to make an inference from a sample, statistics theory is used in building the mathematical model. Machine learning is the computer program for optimizing performance using historical data. Once the model is learned then its representation and algorithm for the solution for inference needs to be efficient. The main purpose that we want of machine learning is to understand our needs so that then our interest will be predicted.

Machine learning theory involves tasks such as:

- Analyzing general issues mathematically.
- Developing the mathematical model to analyze the difficulty of a different kind of learning problem.
- Ensuring that algorithms meet specific conditions, assessing the amount of computational time they need, and ensuring that they have the data necessary for success.
- Developing machine learning algorithms that meet most of the desired criteria.

2.3.2. Time-Series Analysis

A time series is a series of data that is collected over a period of time. Time series data are sequential data which follow some patterns. In order of time, data are points in an index or listed or graphed. Time series data are also called historical data or past data. Time series data are used for predicting a future value based on an historical value. This is called time series analysis [25]. The daily closing price of stocks, heights of ocean tides, and counts of sunspots are some examples of time series data. Time series data are studied for several purposes, such as forecasting the future based on knowledge of the past, understanding of the phenomenon. Underlying measures, or simply succinctly describing the salient features of the series. Forecasting or predicting future prices of an observed time series plays an important role in nearly all fields of science, engineering, finance, business intelligence, economics, meteorology, telecommunications etc. [26,27]. To predict an outcome based on time series data, we can use regression analysis, which is one of the types of supervised learning. Recurrent Neural Networks (RNN) are widely used for regression analysis on time-series data.

2.3.3. Neural Networks

Computer programs that (see http://www.writersdigest.com/online-editor/which-vs-that) work similar to the functioning of a human nervous system are called artificial neural networks. There are various kinds of artificial neural networks and they are applied based on what mathematical operations and set of parameters are required to determine the output. Some types of the neural networks are as follows [28]:

1. Feedforward Neural Network Artificial Neuron
2. Radial basis function Neural Network
3. Kohonen Self Organizing Neural Network

4. Recurrent Neural Network(RNN) Long Short-Term Memory

5. Convolutional Neural Network

6. Modular Neural Network

2.3.4. Choice of the Model

In Feed-forward neural network, it allow signals to travel one way only: from input to output and there are no feedback (loops), the output of any layer does not affect that same layer.

In Recurrent neural network, it allow signals to traveling in both directions by introducing loops in the network. There is a feedback (loops) and feedback networks are powerful, dynamic and can get extremely complicated. Their 'state' is changing continuously until they reach an equilibrium point.

In stock price prediction we need a dynamic feedback network where we can follow the unpredictable trend of the stock price dynamically. That's why we chose recurrent neural network for this work.

2.3.5. Recurrent Neural Network

A recurrent neural network (RNN) is a class of advanced artificial neural network (ANN) that involves directed cycles in memory. One goal of recurrent neural networks is the ability to build on earlier types of networks with fixed-size input vectors and output vectors [29]. In a recurrent neural network (RNN); connections between nodes form a directed graph along a sequence which allows exhibiting dynamic temporal behavior for a time sequence.

Suppose one wants to predict the next word in a sentence or to predict the next day stock price etc. by using Machine Learning. The simplest form has an input layer which receives the input, a hidden layer where the activation is applied and an output layer where one finally receives the output. In more complex forms, where

multiple hidden layers are present, the input layer receives the input, the first hidden layer applies its activations, these activations are sent to the next hidden layer, and each successive layers activations are sent through the layers to finally produce the output. Each hidden layer has its own weights and bias. For this, each layer behaves independently and, unless they have the same weights and bias, these hidden layers cannot be combined with one another. So, a recurrent neuron stores the state of a previous input and combines it with the current input, thereby preserving some relationship of the current input with the previous input.

Long Short-Term Memory networks usually called LSTMs are a special kind of RNN capable of learning long-term dependencies. LSTMs have a chain-like structure, but the repeating module has a slightly different structure. There are multiple layers which interact in a very special way. A common LSTM architecture is composed of a memory cell, an input gate, an output gate and a forget gate. Three of the gates can be thought of as a conventional artificial neuron, as in a multi-layer or feedforward neural network that computes using an activation function of a weighted sum. There are connections between these gates and the cell. An LSTM (memory) cell stores a value (or state), for either long or short time periods [30, 31].

In the figure 2, i, f and O are the input, forget and output gates, respectively. C and $\tilde{C}$ are the memory cell and new memory cell content. Mathematically the gradient is a multi-variable generalization of a derivative that is a vector-valued function, as opposed to a scalar-valued derivative. In deep learning, there are two factors that affect the magnitude of gradients - the weights and the activation functions, especially their derivatives, which the gradient passes through. If either of these factors is smaller than 1, then the gradients may vanish in time; if larger than 1, then exploding might happen. In the recurrency of the LSTM the activation function is an identity function with a derivative of 1.0. So, the backpropagated gradient neither vanishes

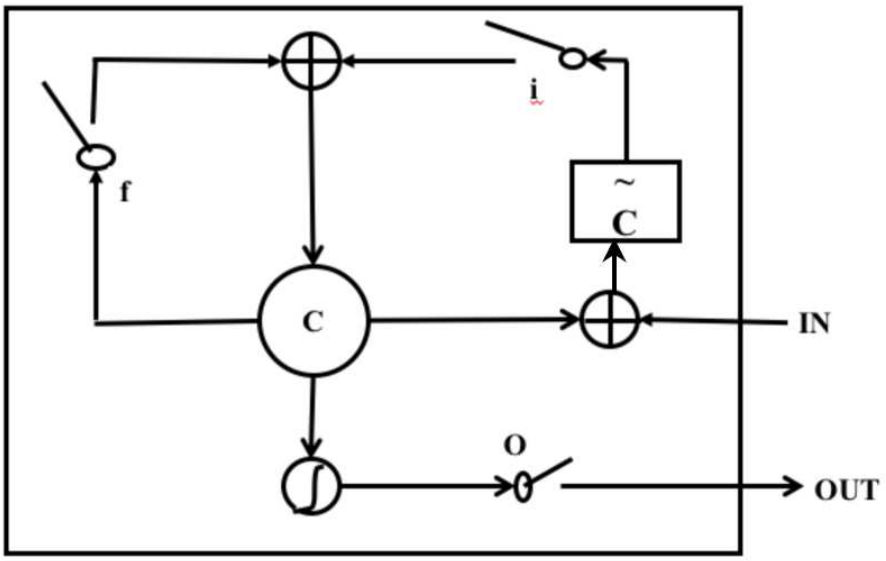

Figure 2. Long Short-Term Memory (LSTM)

nor explodes when passing through, but remains constant. The effective weight of the recurrency is equal to the forget gate activation. So, if the forget gate is on (activation close to 1.0), then the gradient does not vanish. Since the forget gate activation is never greater than 1.0, the gradient cant explode either. Thats why LSTMs are so good for learning long range dependencies and are well-suited to classify process and predict time series given time lags of unknown size and duration between important events. This is how LSTMs were developed to deal with the exploding and vanishing gradient problem when training traditional RNNs.

CHAPTER 3. DEVELOPMENT OF RNN-LSTM MODEL

The first step was to design an RNN-LSTM model for predicting the price of stocks. After successful implementation of the model, the model was validated on 5 popular stocks. The popular stocks used in our experiments are listed below.

- Amazon Inc. (AMZN)
- FB Inc. (FB)
- Google Inc. (GOOGL)
- Microsoft Inc. (MSFT)
- Netflix Inc. (NFLX)

3.1. Data Collection

Five years of historical stock price data was collected from Yahoo Finance [32]. The period of the data was from January, 2013 to December, 2017. The Python pandas library was used to get the data from csv file [33]. Then the collected data was used to train the model. The Python code used for collecting the data from the Pandas library is shown in Figure 3.

```
stock = pd.read_csv('company_name.csv',index_col='Date',parse_dates=True)
stock.info()
stock
```

Figure 3. Code for data collection

3.2. Model Design

To design a prediction model, a particular machine learning algorithm needs to be chosen. For the purpose of this study, a Recurrent Neural Network was chosen, because stock prices are time-series data, and the recurrent characteristics of such a

network work better for predicting time-series data than any other machine learning algorithm. The design of the model may be described as in the flowchart shown in Figure 4 below. In this flow chart, the train-test data first needs to be identified. Then the data set is scaled using MinMaxScaler in order to train the data. A batch function is defined before setting up the RNN. There are important parameters that need to be tuned to develop the optimum result. After the RNN setup, placeholders are defined to hold the desired values, an LSTM cell is defined to deal with short term and long term memory, and dynamic RNN and loss are defined to calculate the Mean Squared Error (MSE). After evaluating the MSE the model decides whether it should run the session to get the final result for predicted price or re-run again for tuning the parameters of the RNN setup. Finally, after necessary iterations, the model is able to predict the stock price very precisely and accurately.

3.2.1. Define Train-Test Data

To train the machine, the data set must be divided into two sets; one for training and one for testing. The separation into two sets is very important, since the training process forms the basis of the ability of the procedure to generalize, which is measured by performance on the testing set. Typically, most of the data is used for training, and a smaller portion of the data is used for testing. Models will be developed using the training dataset and will make predictions on the test dataset.

As said above, five years of historical stock price data was collected for prediction. In particular, a total of 1259 days of data was collected. The first 1008 data points are for the first 4 years (2013 to 2016) and the last 251 data are for each day in 2017. In our first experiments, 1008 data points were used as a training set. For each of the five stocks, in the first experiment, each model predicted the 1009th data point. Next, data point 1009 was included in the training data, the first point was excluded, and data point 1010 was predicted. This process was continued to

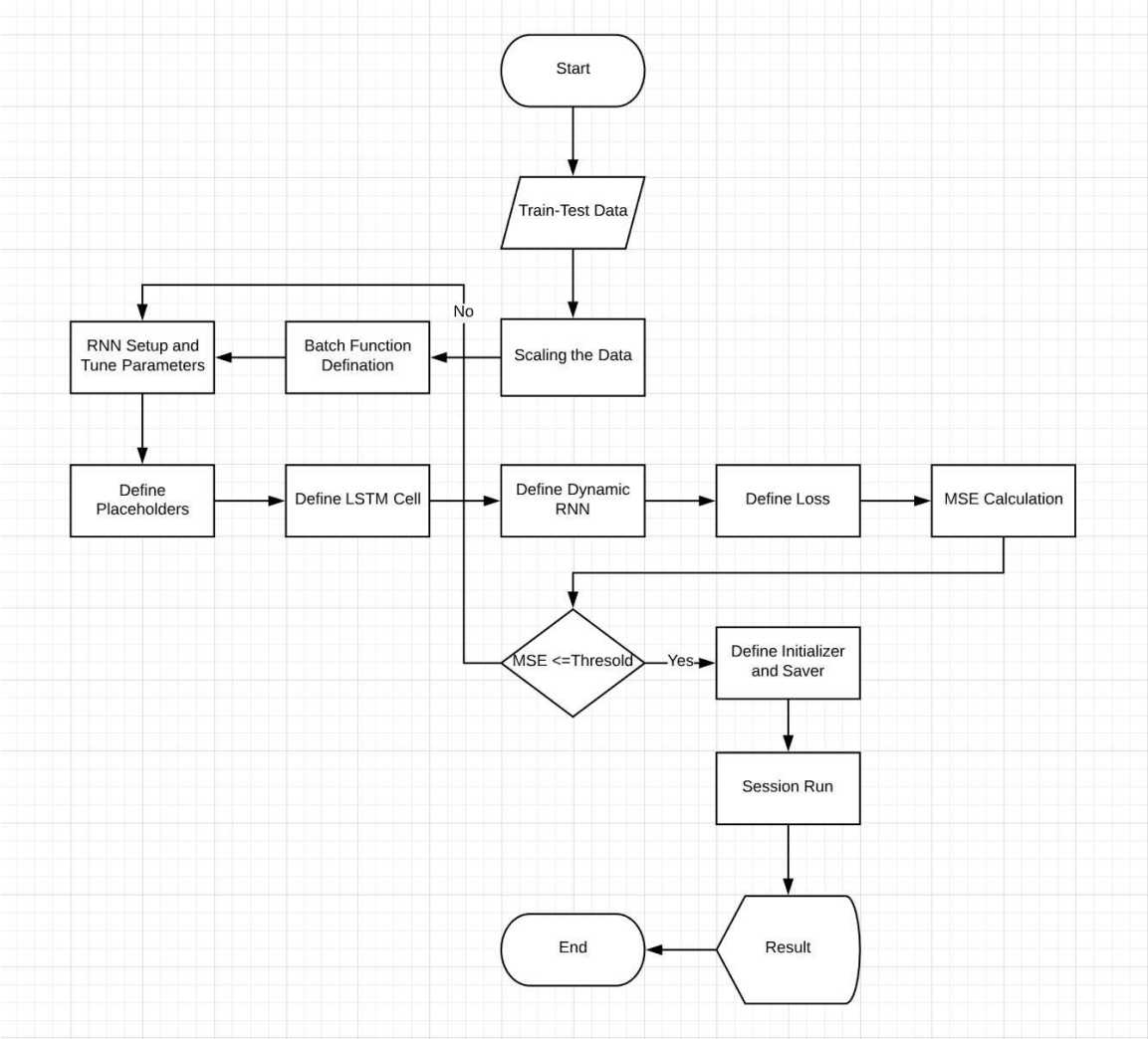

Figure 4. The design overview of RNN model

predict the stock price for all the days in 2017. In each case, 1008 data points were used to make the prediction. The actual test data for 2017 was compared with the predictions to validate the model.

The code that is used for splitting the training and testing data is in Figure 5

```
hold=np.zeros((251,2),dtype=np.float32)
hold1=np.ones((251,1),dtype=np.chararray)
for j in range(1008,1259,+1):
    hold[j-1008,0]=stock.values[j]
    hold1[j-1008]=stock.index[j]
```

Figure 5. Code to set training and testing data.

3.2.2. Scaling the Data

To standardize the range of independent variables or features of data, a feature scaling method was used for data processing. This is also known as data normalization, and is generally performed during the data pre-processing step. Here, MinMaxScaler was used to scale the data as a part of pre-processing. To normalize the data set, fit and transform were both used for training data, and only transform was used for testing data, because the scale of the testing data is already known. Normalizing or transforming the data means that the new scale variable will be between zero and one.

The code that is used for scaling the data is shown in Figure 6.

```
#sklearn
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
train_scaled = scaler.fit_transform(train_set)
#test_scaled = scaler.transform(test_set)
```

Figure 6. Code to scale the data.

3.2.3. Batch Function Definition

Here, the batch function is defined for getting the next_batch. The arguments of this function are training_data, batch_size, and steps. The data used to train the

model are training_data, the length of each batch is batch_size and steps are defined as the step size to move forward. Each successive input is called time step. Batch size limits the number of samples and the batch highest size will be equal to the number of training observations. It also defines how many times we want to predict at a time.

The code that is used for creating batch function of the the data is below in Figure 7.

```
#batch Defination
def next_batch(training_data,batch_size,steps):

    rand_start = np.random.randint(0,len(training_data)-steps)

    y_batch = np.array(training_data[rand_start:rand_start+steps+1]).reshape(1,steps+1)

    return y_batch[:, :-1].reshape(-1, steps, 1), y_batch[:, 1:].reshape(-1, steps, 1)
```

Figure 7. Code to define the batch function.

3.2.4. RNN Setup and Tuning Parameters

The most important task in predictive mode is to tune the parameters. The number of neurons are defined as num_neurons. A greater number of neurons can improve the model, but it might take a longer time to process. The learning_rate is also another important parameter that impacts the time needed to build the model. The number of iterations to train the model is defined as num_train_iterations. This is the time step and is incremented in every step. The code used for RNN setup and tuning the parameters is shown in Figure 8 below.

```
#RNN Setup
import tensorflow as tf

#Tuning parameters
num_inputs = 1
num_time_steps = 1
num_neurons = 500
num_outputs = 1
learning_rate = 0.002
num_train_iterations = 4000
batch_size = 4
```

Figure 8. Code for RNN setup and tuning parameters.

3.2.5. Define Placeholders

A placeholder is an empty node that needs a value to be provided in order to compute output. Here, X and y are used for the placeholder of the tensorflow [34]. The code which one is used for the placeholder is below in Figure 9.

```
#placeholder
X = tf.placeholder(tf.float32, [None, num_time_steps, num_inputs])
y = tf.placeholder(tf.float32, [None, num_time_steps, num_outputs])
```

Figure 9. Code to define placeholders.

3.2.6. Define LSTM Cell

A basic LSTM (Long Short Term Memory) cell is responsible for remembering values over arbitrary time intervals. An LSTM cell is a cell that has its own memory and which can behave like an intelligent human brain in making decisions. Cells that are a function of inputs from previous time steps are also known as memory cells.

To define the cost function of a recurrent neural network the ReLU (Rectified Liner Unit) is used. The code which one is used to define the LSTM Cell is in the below Figure 10. Here, a basic LSTM cell is used where num_units can be interpreted as the analogy of a hidden layer from the feed forward neural network where the number of nodes in the hidden layer of a feed forward neural network is equivalent to num_units number of LSTM units in an LSTM cell at every time step of the network.

The code that is used to define an LSTM cell is in Figure 10.

```
#LSTM Cell
cell = tf.contrib.rnn.OutputProjectionWrapper(
    tf.contrib.rnn.BasicLSTMCell(num_units=num_neurons, activation=tf.nn.relu),
    output_size=num_outputs)
```

Figure 10. Code to define LSTM cell.

3.2.7. Define Dynamic RNN

Here, Dynamic_Rnn is used to get the outputs and states of the cells. Basically, it means using a while loop operation to run over a cell to get the appropriate number of times. Dynamic RNN's allow for variable sequence lengths. It might have an input shape (batch_size, max_sequence_ length), but this will also allow the RNN to run for the correct number of time steps on those sequences that are shorter than max_sequence_length. The code which one is used for Dynamic RNN is shown below in Figure 11.

```
#Dynamic RNN
outputs, states = tf.nn.dynamic_rnn(cell, X, dtype = tf.float32)
```

Figure 11. Code to define dynamic RNN.

3.2.8. Define Loss

Here, using the cost function, loss that evaluates the quality of our model was reduced. Outputs need to be in the same format as inputs to compare the results

using the loss function. To execute the optimization, the Adam optimizer is used. Adam optimizer is a general-purpose optimizer that performs gradient descent via backpropagation through time. This allows faster convergence at the cost of more computation. Following the optimizer, optimizer.minimize(loss), is used to define the new train data. The below code in the Figure 12 is used for defining the loss function.

```
#Loss
loss = tf.reduce_mean(tf.square(outputs - y))
optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate)
train = optimizer.minimize(loss)
```

Figure 12. Code to define loss.

3.2.9. Define Initializer and Saver

Here, initializer and saver are defined to initialize all the variables and to save the model that is designed. The Initializer and Saver code are shown in Figure 13.

```
#initializer
init = tf.global_variables_initializer()
#saver
saver = tf.train.Saver()
#gpu_options
gpu_options = tf.GPUOptions(per_process_gpu_memory_fraction=0.9)
```

Figure 13. Code to define initializer and saver.

3.2.10. MSE Calculation

The MSE or Mean Squared Error is an estimator to measure of the quality. The result is always non-negative, and values closer to zero are better. It is calculated to get num_train_iterations times. The MSE is printed on every 100 times to see the improvement to train the model. Here, mean squared error (MSE) is used to minimize the difference between the actual and the predicted values. MSE Calculation code is in Figure 14.

```
#MSE CAlculation
with tf.Session(config=tf.ConfigProto(gpu_options=gpu_options)) as sess:
    sess.run(init)

    for iteration in range(num_train_iterations):

        X_batch, y_batch = next_batch(train_scaled,batch_size,num_time_steps)
        sess.run(train, feed_dict={X: X_batch, y: y_batch})

        if iteration % 100 == 0:

            mse = loss.eval(feed_dict={X: X_batch, y: y_batch})
            print(iteration, "\tMSE:", mse)

    saver.save(sess, "./rnn_stock_time_series_model")
```

Figure 14. Code for MSE calculation.

3.2.11. Session Run

Here, the session is restored from the saved session. X_batch and y_pred are defined to produce the train_seed.

The code that is used to run the session is in Figure 15

```
#Session Run
with tf.Session() as sess:

    saver.restore(sess, "./rnn_stock_time_series_model")

    train_seed = list(train_scaled[-num_time_steps:])
    for iteration in range(num_time_steps):
        X_batch = np.array(train_seed[-num_time_steps:]).reshape(1, num_time_steps, 1)
        y_pred = sess.run(outputs, feed_dict={X: X_batch})
        train_seed.append(y_pred[0, -1, 0])

#Print Train_seed
train_seed
```

Figure 15. Code to run the session.

3.2.12. Results

Finally, a result is generated and saved in the hold array which was defined earlier.

The code that is used to get the results is in Figure 16

```
#results
results = scaler.inverse_transform(np.array(train_seed[num_time_steps:]).reshape(num_time_steps,1))

#for k in range(0,num_time_steps,+1):
    #hold[k,1]=results[k]

hold[0,1]=results

hold
```

Figure 16. Code to get the results.

3.3. Define the Loops to Run the Model Continuously

After designing the predictive model, we found a predictive stock price on the first day of 2017. This price became the 1009th data point in this dataset. From here, as explained earlier, the training set is changed continuously using a for loop to predict all 251 days in 2017 individually. Then, the new predicted stock prices are saved in the array named hold. At the end, pd.ExcelWriter is used to convert all the actual and predicted stock prices to an excel file. Later, the excel file is converted to a data frame to generate all the figures and present the results.

The code that defines the loops to run the model continuously is in Figure 17

```
for i in range(1,251,+1):
    hold
    print("Pass ",i )

    train_set=stock[i:1008+i]

    #sklearn
    from sklearn.preprocessing import MinMaxScaler
    scaler = MinMaxScaler()
    train_scaled = scaler.fit_transform(train_set)
    #test_scaled = scaler.transform(test_set)
    #batch Defination
    #MSE CAlculation
    with tf.Session(config=tf.ConfigProto(gpu_options=gpu_options)) as sess:
        sess.run(init)

        for iteration in range(num_train_iterations):

            X_batch, y_batch = next_batch(train_scaled,batch_size,num_time_steps)
            sess.run(train, feed_dict={X: X_batch, y: y_batch})

            if iteration % 100 == 0:

                mse = loss.eval(feed_dict={X: X_batch, y: y_batch})
                print(iteration , "\tMSE:", mse)


        saver.save(sess, "./rnn_stock_time_series_model")
    #Session Run
    with tf.Session() as sess:

        saver.restore(sess, "./rnn_stock_time_series_model")

        train_seed = list(train_scaled[-num_time_steps:])
        for iteration in range(num_time_steps):
            X_batch = np.array(train_seed[-num_time_steps:]).reshape(1, num_time_steps, 1)
            y_pred = sess.run(outputs, feed_dict={X: X_batch})
            train_seed.append(y_pred[0, -1, 0])
    #Print Train_seed
    train_seed
    #results
    results = scaler.inverse_transform(np.array(train_seed[num_time_steps:]).reshape(num_time_steps,1))
    hold[i,1]=results
```

Figure 17. Code to define loops to run the model continuously.

CHAPTER 4. RESULTS AND DISCUSSION

4.1. Percentage of Error

The price of the five stocks was successfully predicted using the RNN-LSTM Model developed using Tensorflow [34]. The results are discussed below for each of the stocks. After implementing the model, the percentage of error was calculated using the following equation;

% of error = ((Actual Price - Predicted Price)/ Actual price) X 100

To nullify the negative value impact, the absolute value of the error was used. For the five of the companies the weekly, monthly and quarterly means were calculated and shown for representing all the business days' data for one year. Weekly means are calculated by all the business days' data of that particular week. Monthly means are calculated by all the business days' data of that particular month, and quarterly means are calculated by all the business days' data for each particular quarter of the year 2017.

4.1.1. Amazon Inc.

For Amazon, the weekly mean, monthly mean, and quarterly mean were calculated for both the actual price and the predicted price of AMZN for the year 2017. In order to make the first comparison between the predicted data and the actual data, the following bar chart plots are produced.

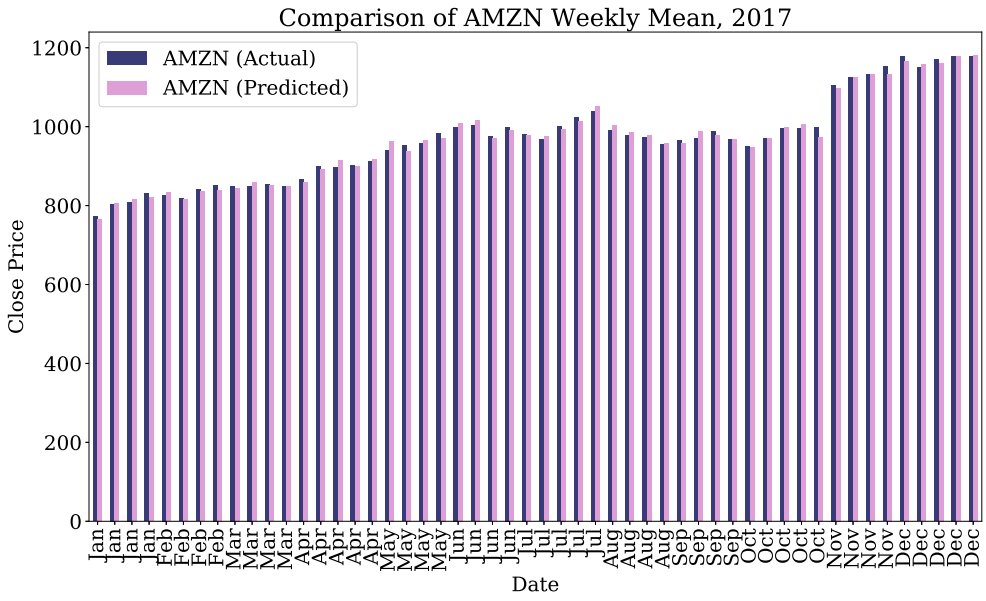

Figure 18. Comparison of AMZN Weekly Mean, 2017

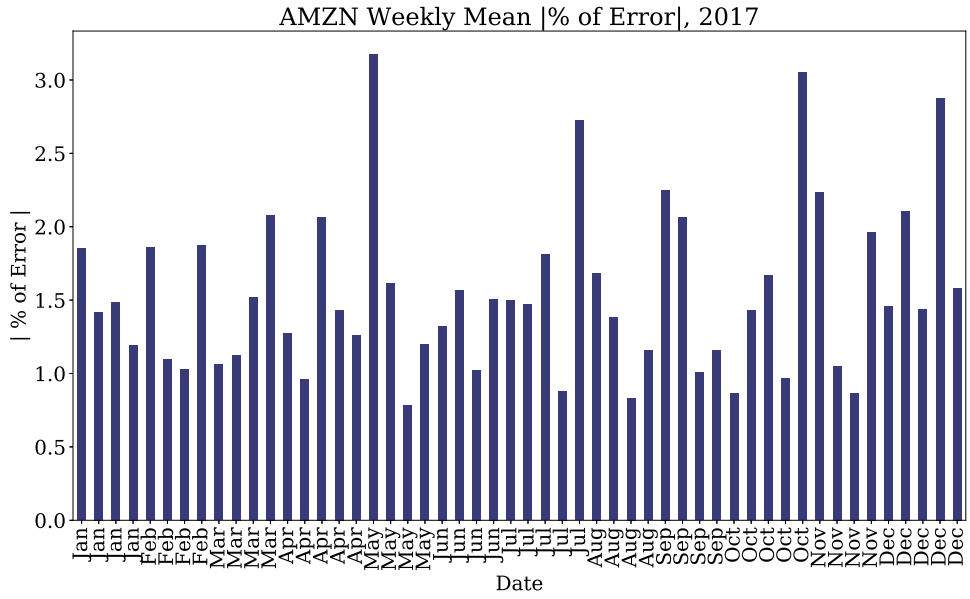

Figure 19. AMZN Weekly Mean |% of Error|, 2017

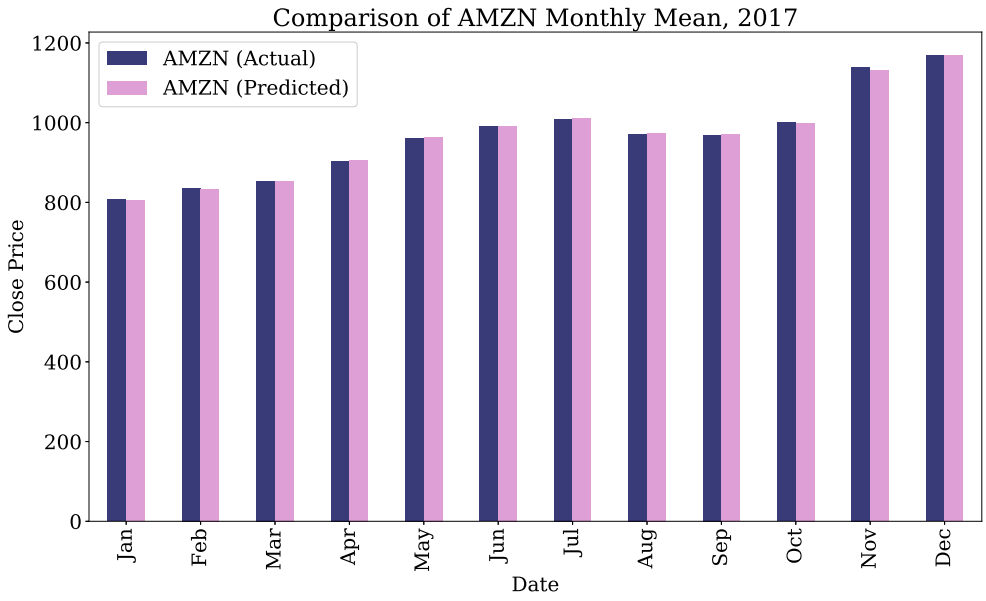

Figure 20. Comparison of AMZN Monthly Mean, 2017

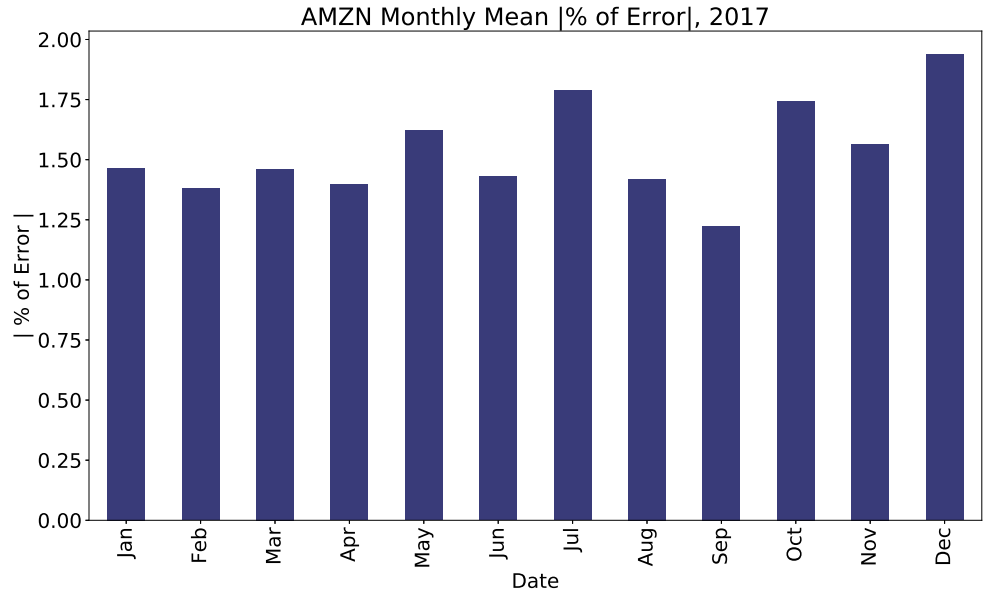

Figure 21. AMZN Monthly Mean |% of Error|, 2017

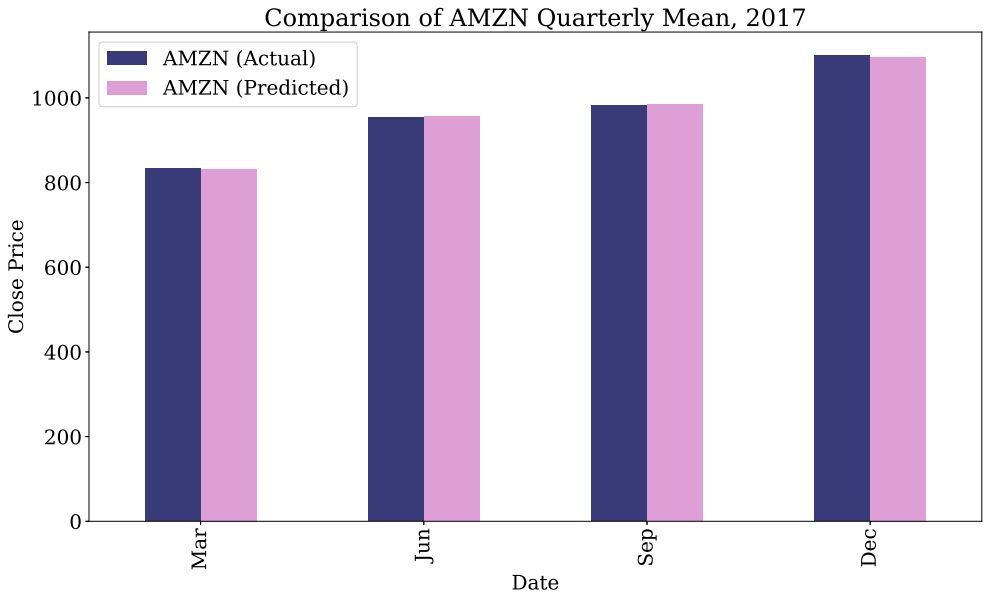

Figure 22. Comparison of AMZN Quarterly Mean, 2017

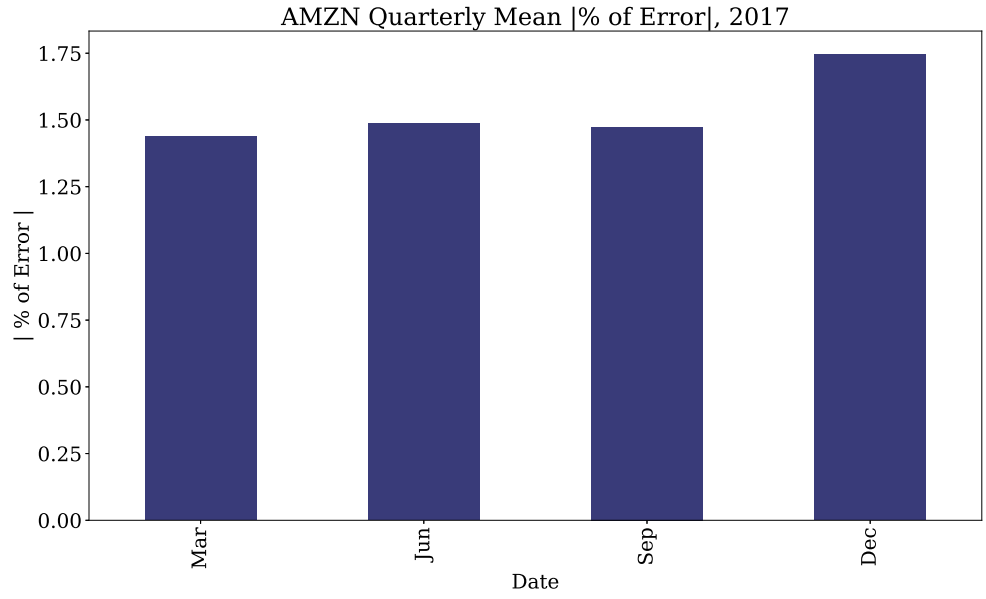

Figure 23. AMZN Quarterly Mean |% of Error|, 2017

In the Figures 18, 20 and 22, the comparison between the actual price and the predicted price was shown in three different periods of intervals. The absolute value

of the percentage errors of the most of the stocks are very close. In weekly mean values, as shown in Figure 19 highest error is 3 percent. In monthly mean values, highest error is 2 percent, as shown in Figure 21. In quarterly mean values, as shown in Figure 23 highest error is 1.75 percent. In the graphs, by comparing the actual price (weekly, monthly and quarterly mean) and predicted price (weekly, monthly and quarterly mean) all are so close that its in some graph its almost same. In the weekly means, we observed some uneven spike in the month of May, July, October and December. Those spikes are observed due to other impacting factors which are not considered in this study. Considering only the historical prices, all the values are less than 5 percent which is a good indicator that it validates the prediction model.

4.1.2. Facebook Inc.

Next for Facebook Inc; the weekly mean, monthly mean, and quarterly mean were calculated both for the actual and the predicted values for the year 2017. In order to make the first comparison between the predicted data and the actual data, bar chart plots are produced.

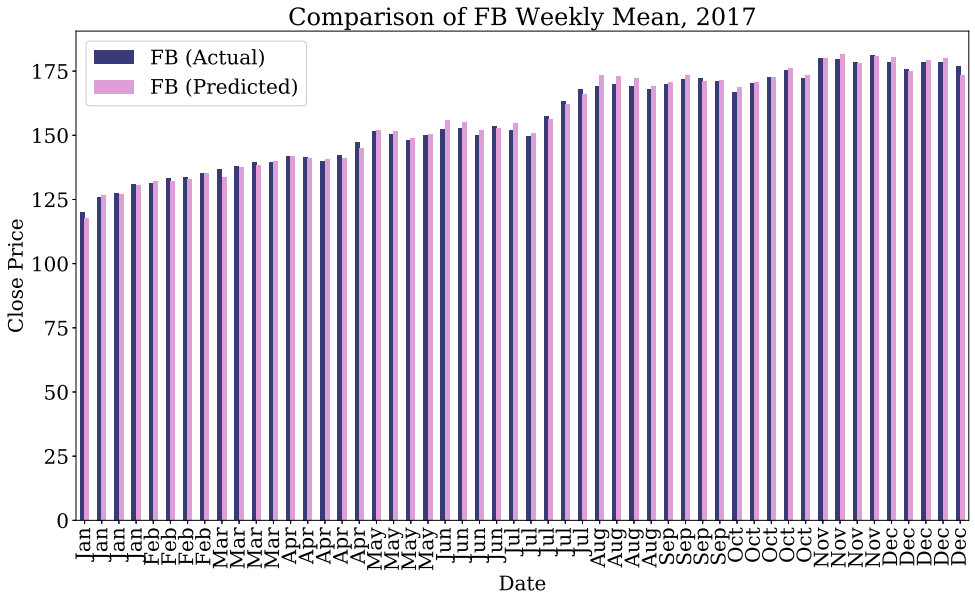

Figure 24. Comparison of FB Weekly Mean, 2017

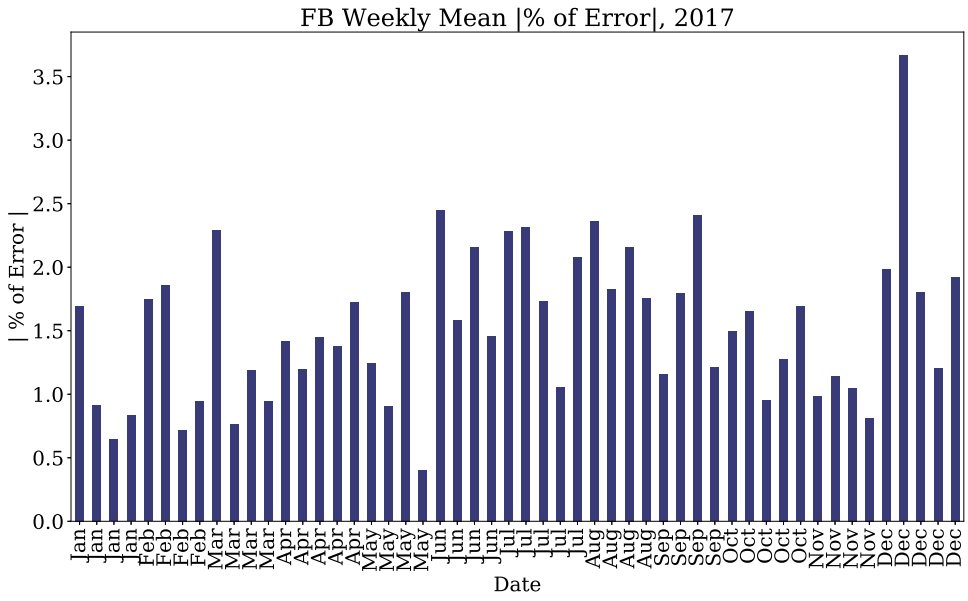

Figure 25. FB Weekly Mean |% of Error|, 2017

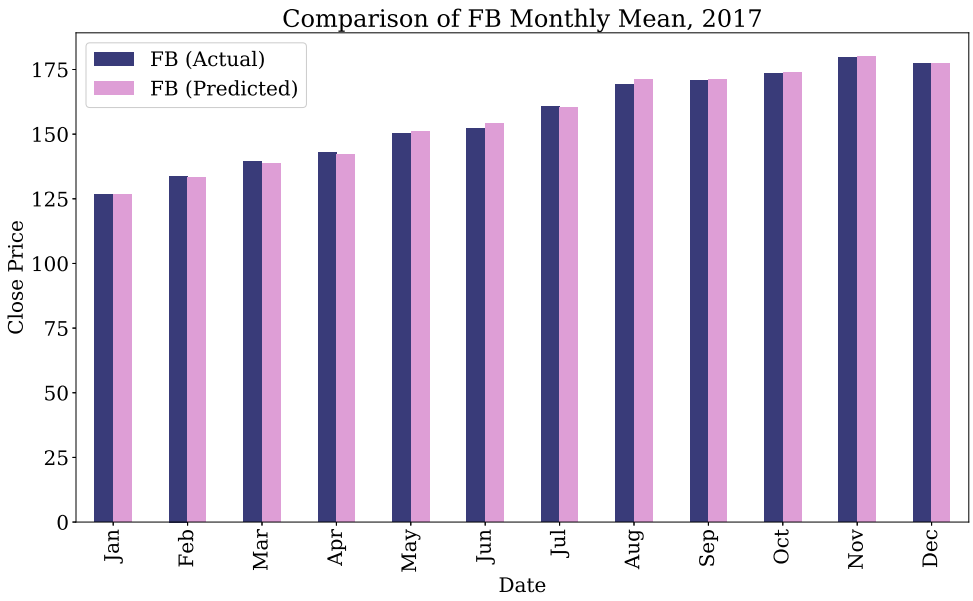

Figure 26. Comparison of FB Monthly Mean, 2017

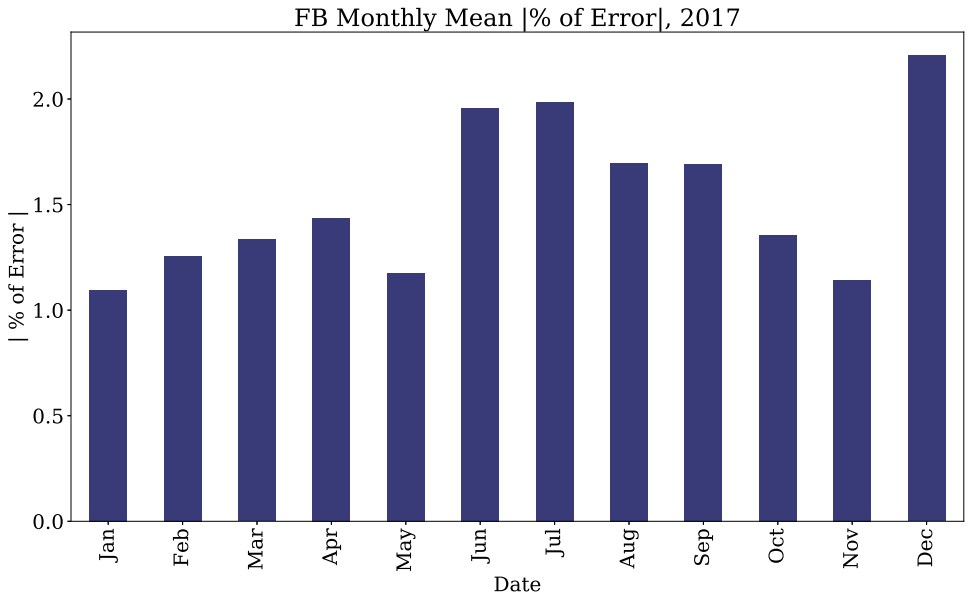

Figure 27. FB Monthly Mean |% of Error|, 2017

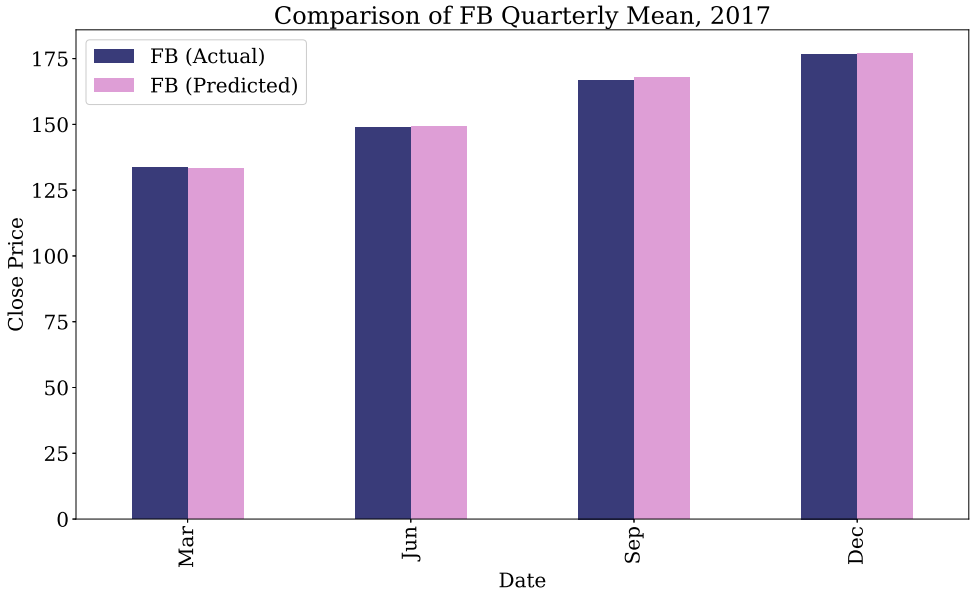

Figure 28. Comparison of FB Quarterly Mean, 2017

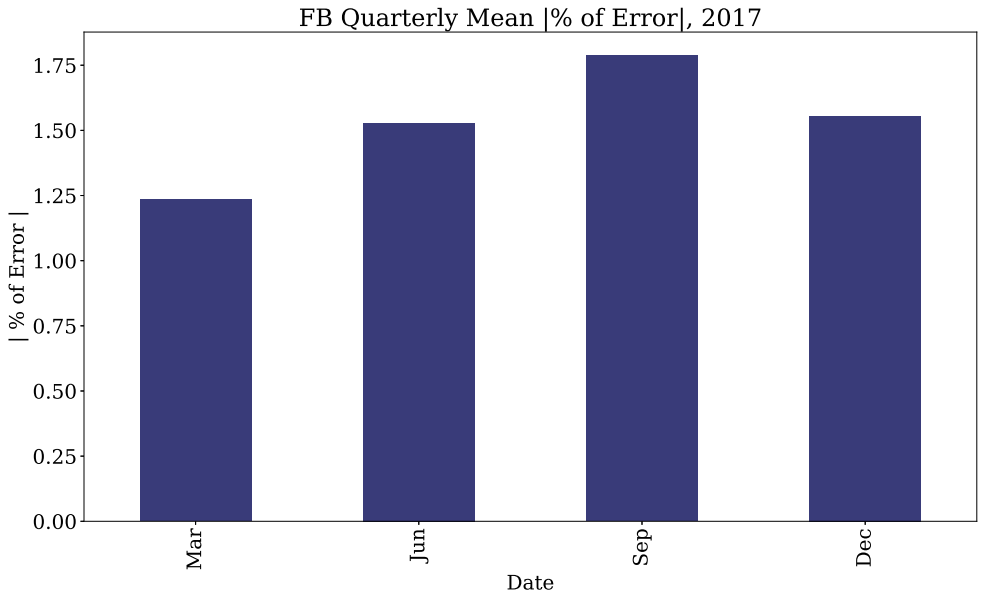

Figure 29. FB Quarterly Mean |% of Error|, 2017

In the Figures 24, 26 and 28, the comparison between the actual price and the predicted price was shown in three different periods of intervals and they are very close . In weekly mean values, as shown in Figure 25 highest error is 3.5 percent. In monthly mean values, highest error is 2.5 percent, as shown in Figure 27. In quarterly mean values, as shown in Figure 29 highest error is 1.75 percent. In the graphs, by comparing the actual price (weekly, monthly and quarterly mean) and predicted price (weekly, monthly and quarterly mean) all are so close that its in some graph its almost same. In the weekly means, we observed some uneven spike in the month of December. Those spikes are observed due to other impacting factors which are not considered in this study. Considering only the historical prices, all the values are less than 5 percent which is a good indicator that it validates the prediction.

4.1.3. Google Inc.

For Google Inc., the weekly mean, monthly mean and quarterly mean were calculated both for the actual and the predicted values for the year 2017. In order

to make the first comparison between the predicted data and the actual data, the following bar chart plots are produced.

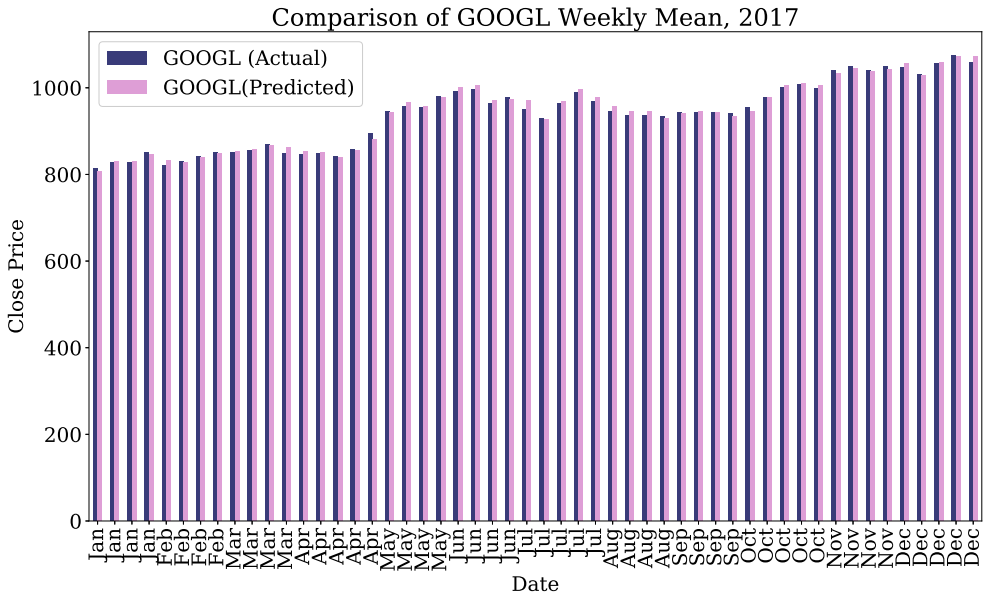

Figure 30. Comparison of GOOGL Weekly Mean, 2017

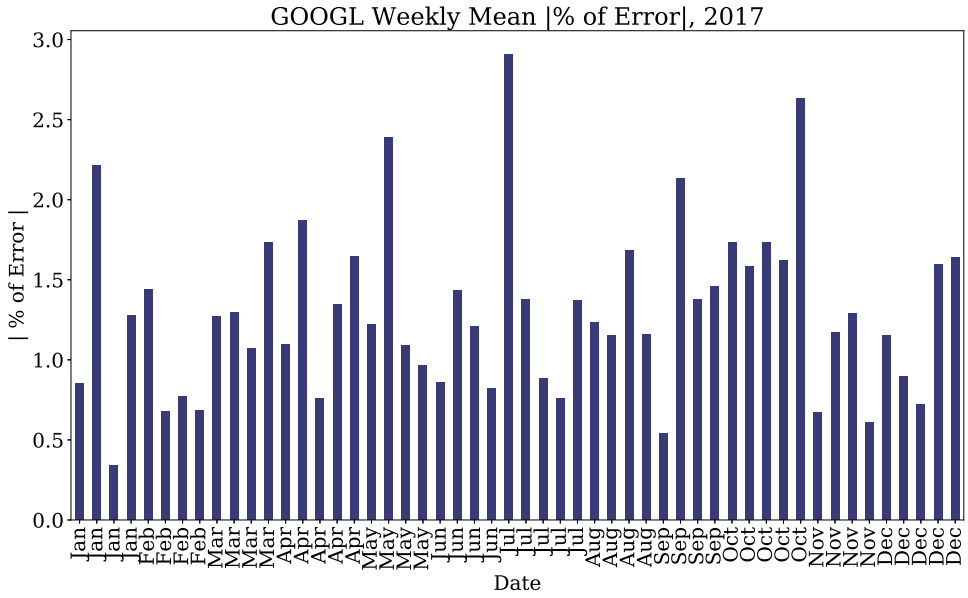

Figure 31. GOOGL Weekly Mean |% of Error|, 2017

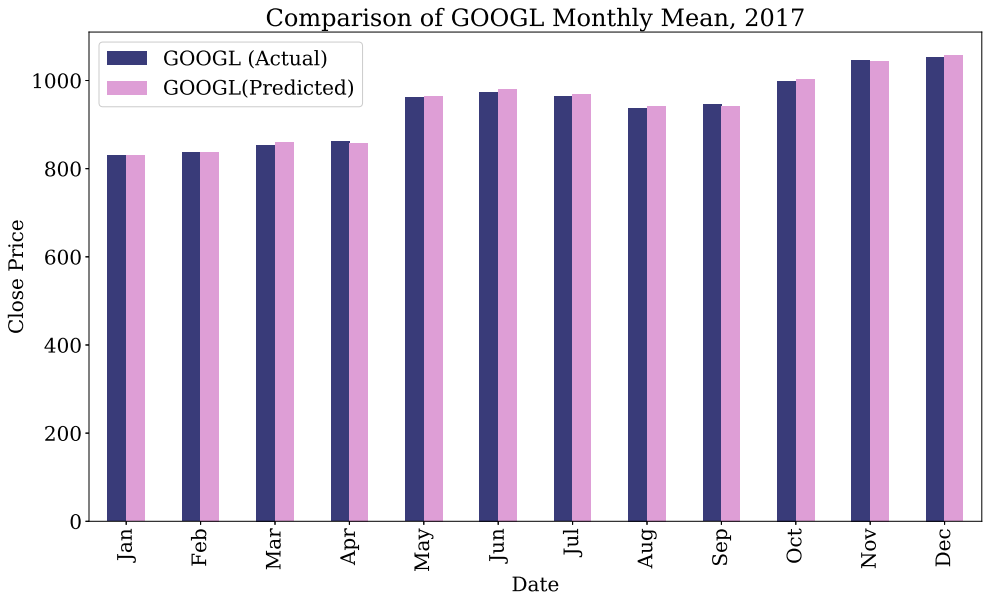

Figure 32. Comparison of GOOGL Monthly Mean, 2017

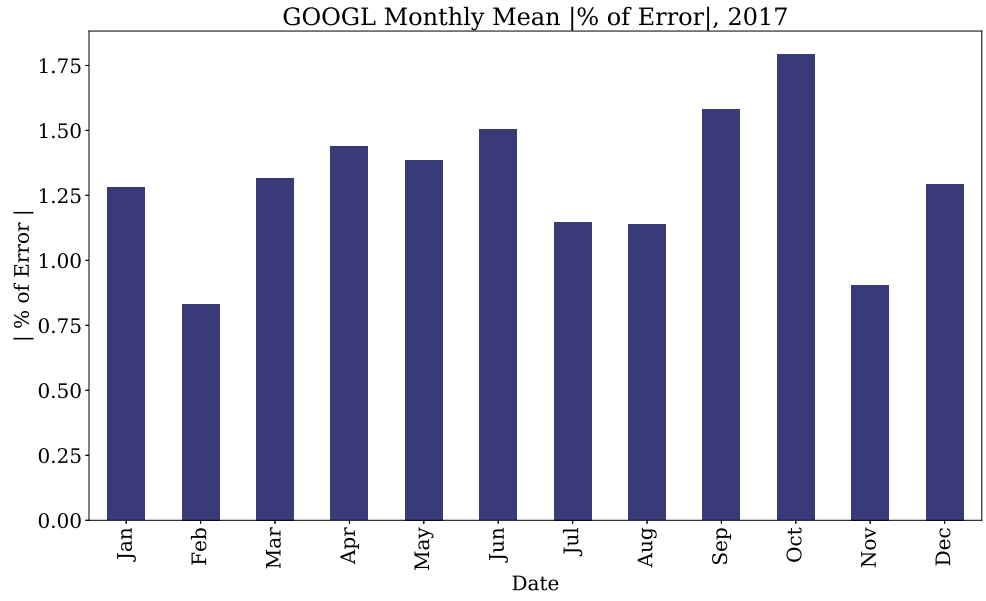

Figure 33. GOOGL Monthly Mean |% of Error|, 2017

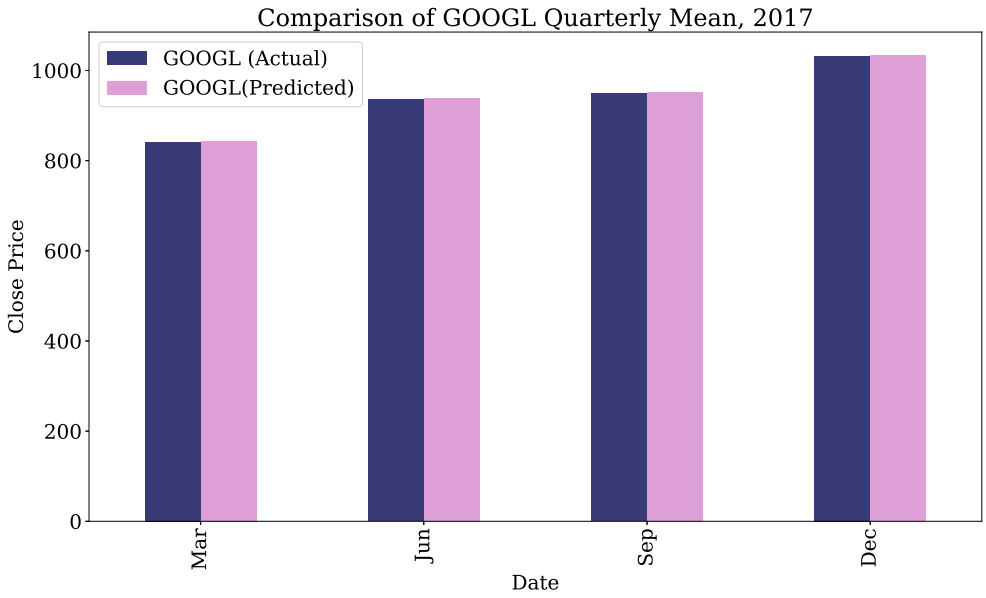

Figure 34. Comparison of GOOGL Quarterly Mean, 2017

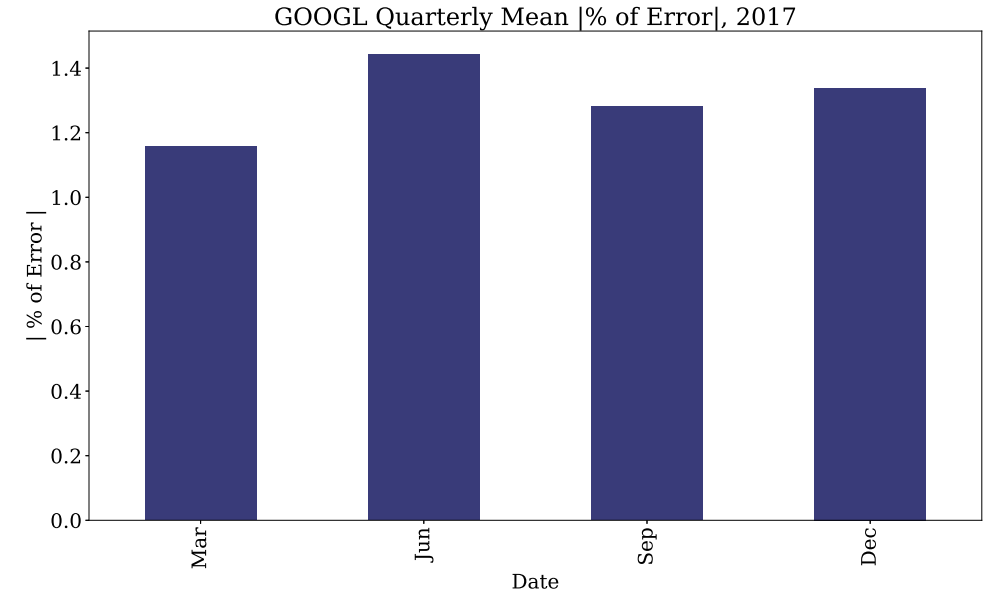

Figure 35. GOOGL Quarterly Mean |% of Error|, 2017

In the Figures 30, 32 and 34, a comparison between the actual price and the predicted price is shown in three different periods of intervals. In most of the cases

they are very close to the actual value. In weekly mean values, as shown in Figure 31 highest error is 3 percent. In monthly mean values, highest error is 1.75 percent, as shown in Figure 33. In quarterly mean values, as shown in Figure 35 highest error is 1.4 percent. In the graphs, by comparing the actual price (weekly, monthly and quarterly mean) and predicted price (weekly, monthly and quarterly mean) all are so close that its in some graph its almost same. In the weekly means, we observed some uneven spike in the month of January, May, July and October. Those spikes are observed due to other impacting factors which are not considered in this study. Considering only the historical prices, all the values are less than 5 percent which is a good indicator that it validates the prediction.

4.1.4. Microsoft Inc.

For Microsoft Inc., the weekly mean, monthly mean and quarterly mean were calculated for both the actual and the predicted values for the year 2017. In order to make the first comparison between the predicted data and the actual data, the following bar chart plots are produced.

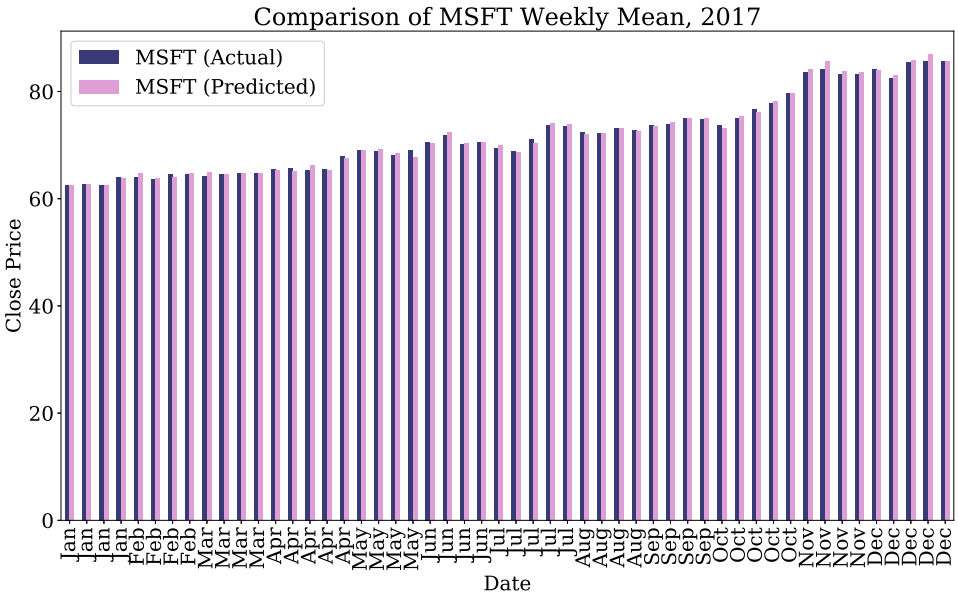

Figure 36. Comparison of MSFT Weekly Mean, 2017

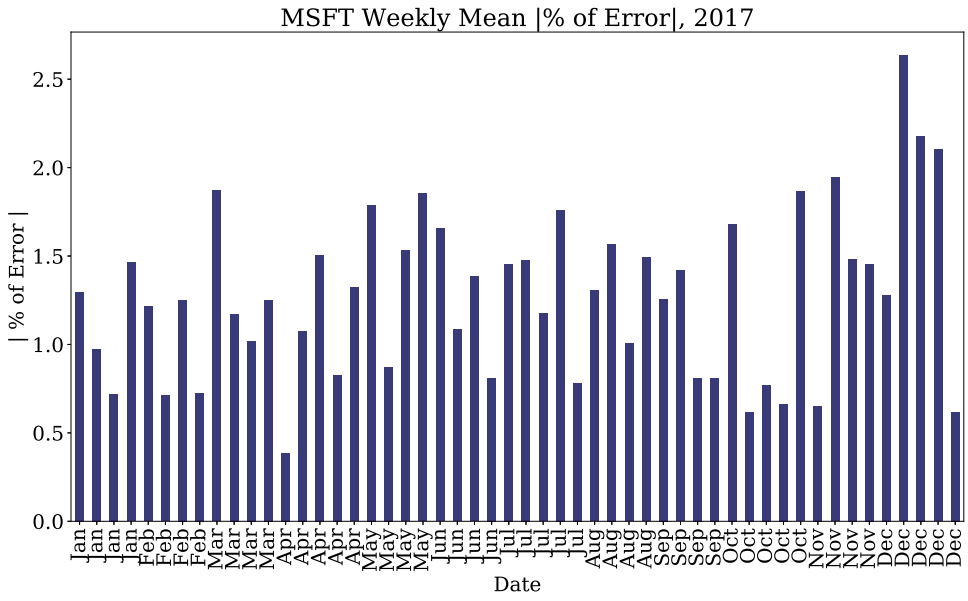

Figure 37. MSFT Weekly Mean |% of Error|, 2017

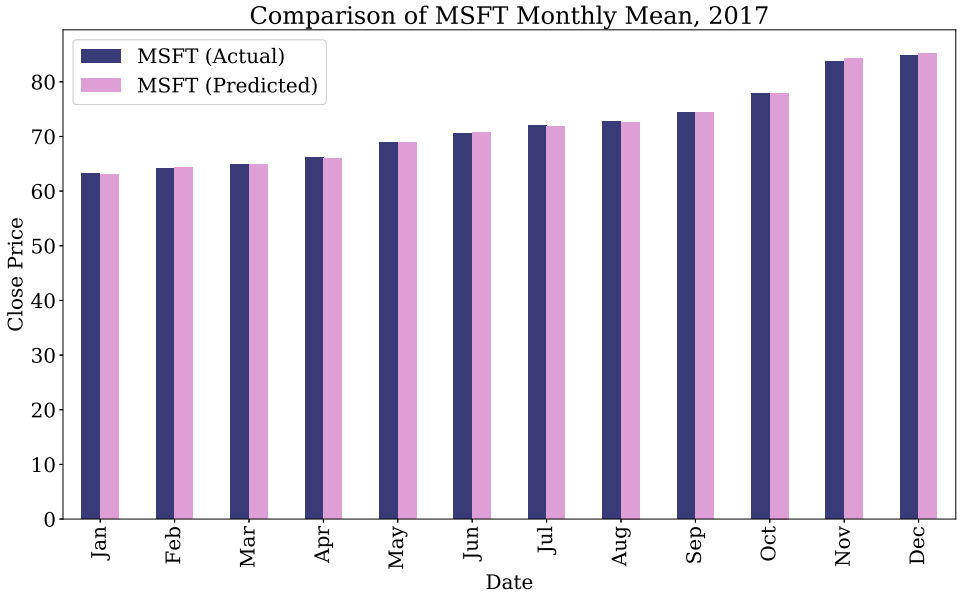

Figure 38. Comparison of MSFT Monthly Mean, 2017

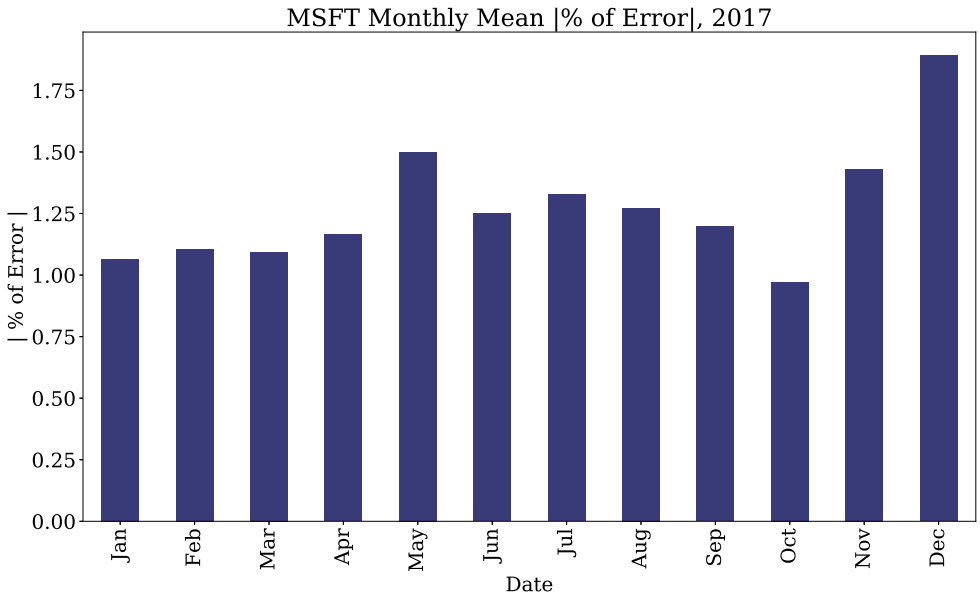

Figure 39. MSFT Monthly Mean |% of Error|, 2017

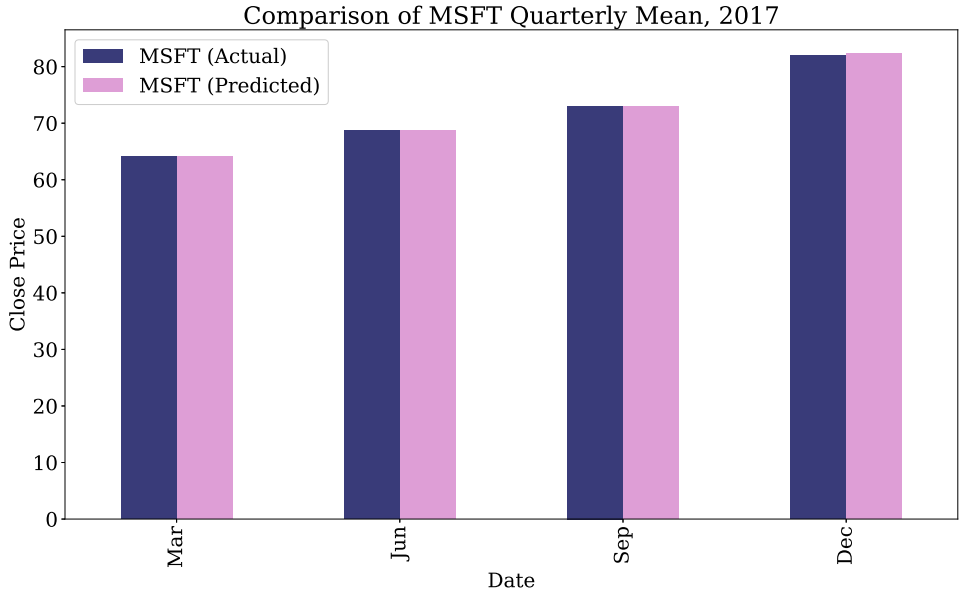

Figure 40. Comparison of MSFT Quarterly Mean, 2017

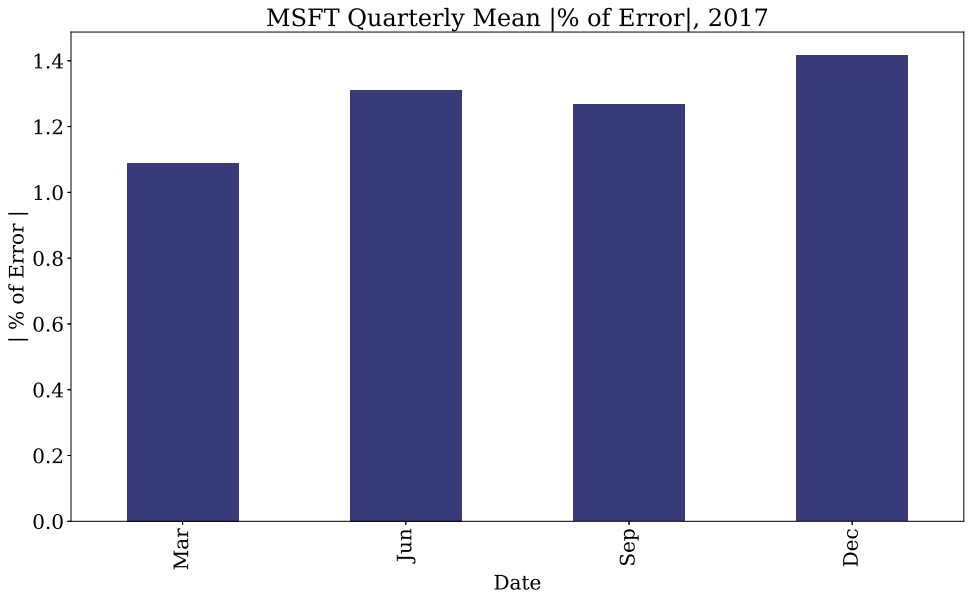

Figure 41. MSFT Quarterly Mean |% of Error|, 2017

In Figures 36, 38 and 40, the comparison between the actual price and the predicted price is shown in three different periods of intervals. In weekly mean values, as shown in Figure 37 highest error is 2.5 percent. In monthly mean values, highest error is 1.75 percent, as shown in Figure 39. In quarterly mean values, as shown in Figure 41 highest error is 1.45 percent. In the graphs, by comparing the actual price (weekly, monthly and quarterly mean) and predicted price (weekly, monthly and quarterly mean) all are so close that its in some graph its almost same. In the weekly means, we observed some uneven spike in the month of December. Those spikes are observed due to other impacting factors which are not considered in this study. Considering only the historical prices, all the values are less than 5 percent which is a good indicator that it validates the prediction.

4.1.5. Netflix Inc.

Finally, for Netflix Inc., the weekly means, monthly means and quarterly means were calculated for both the actual and predicted values for the year 2017. In order

to make the first comparison between the predicted data and the actual data, the following bar chart plots are produced.

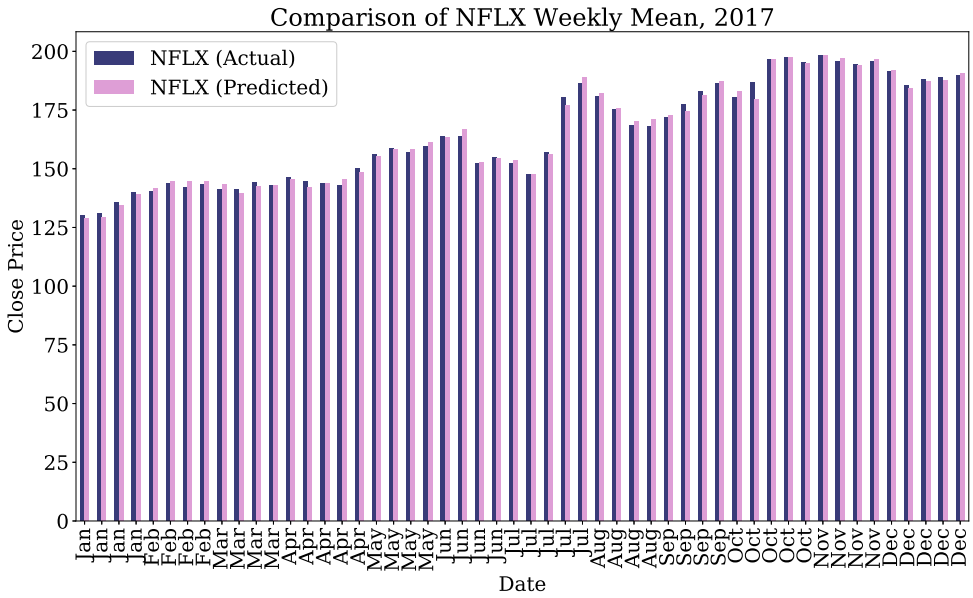

Figure 42. Comparison of NFLX Weekly Mean, 2017

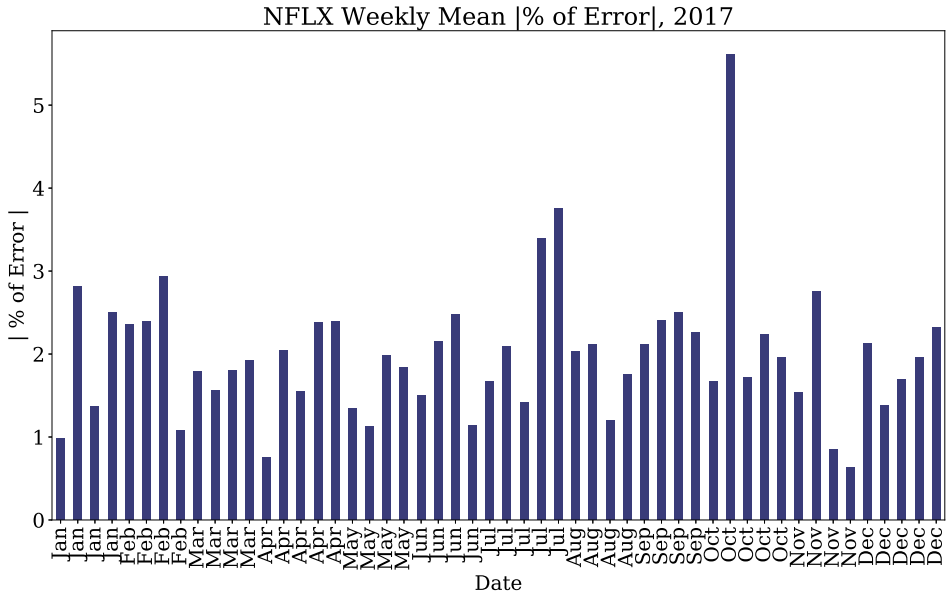

Figure 43. NFLX Weekly Mean |% of Error|, 2017

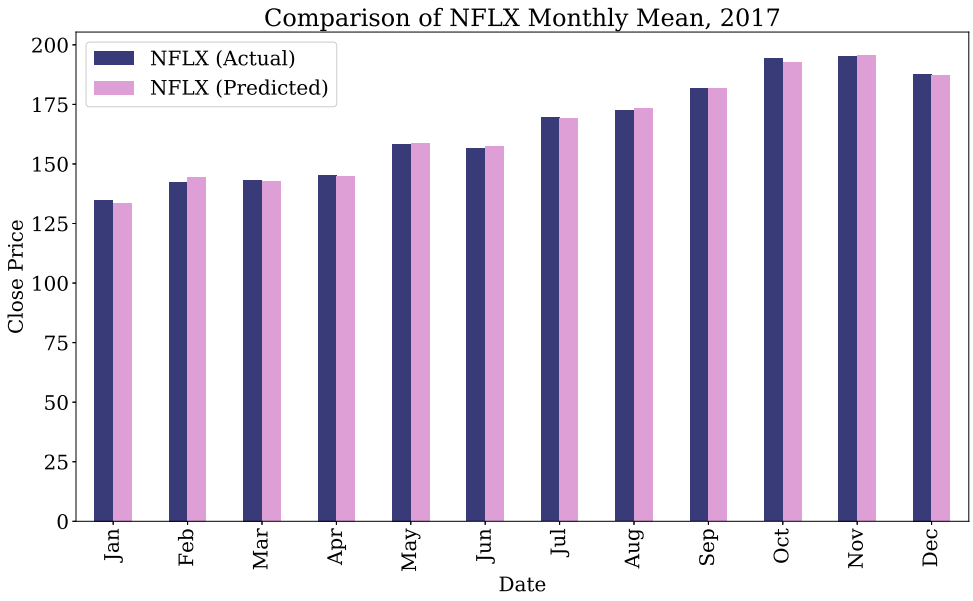

Figure 44. Comparison of NFLX Monthly Mean, 2017

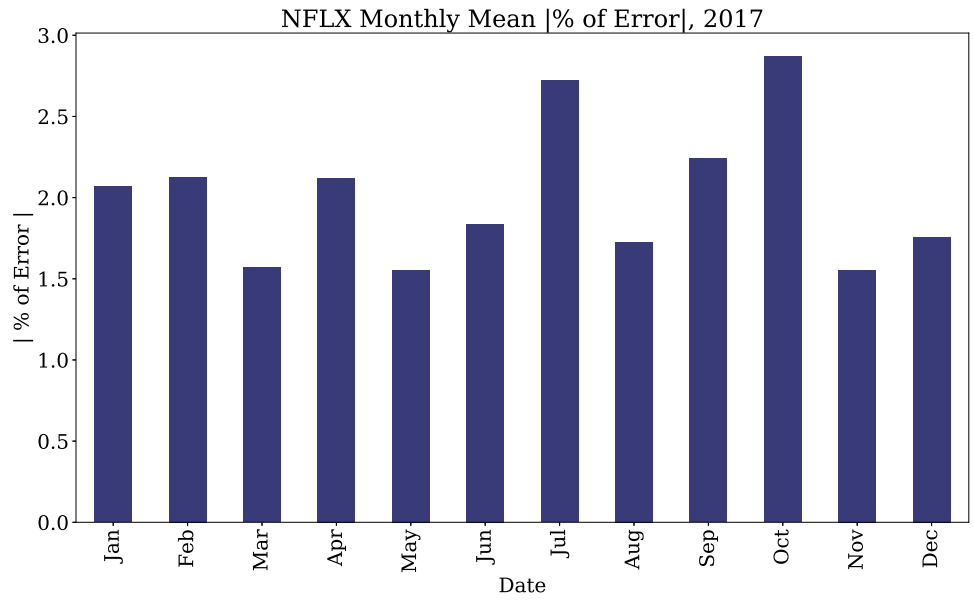

Figure 45. NFLX Monthly Mean |% of Error|, 2017

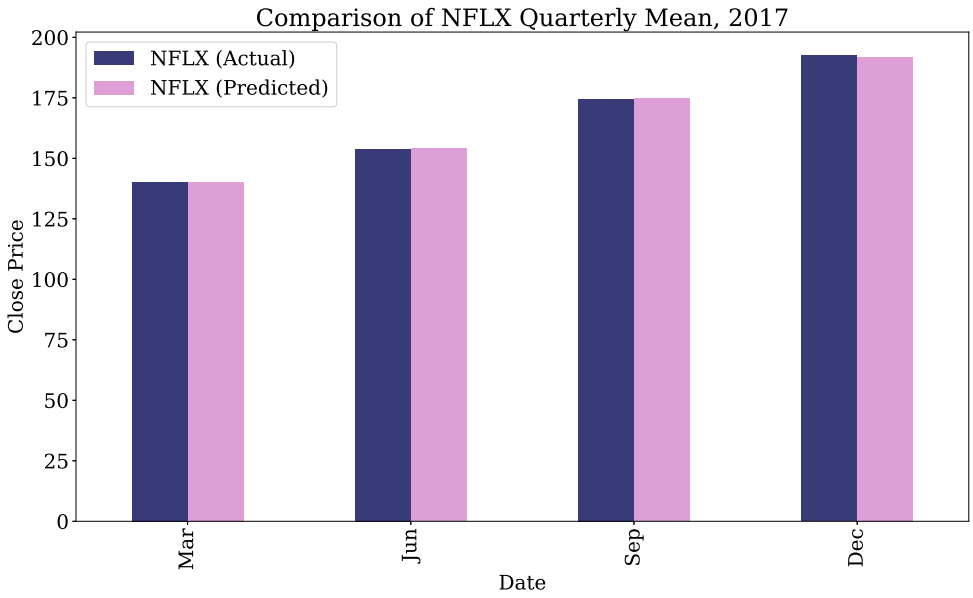

Figure 46. Comparison of NFLX Quarterly Mean, 2017

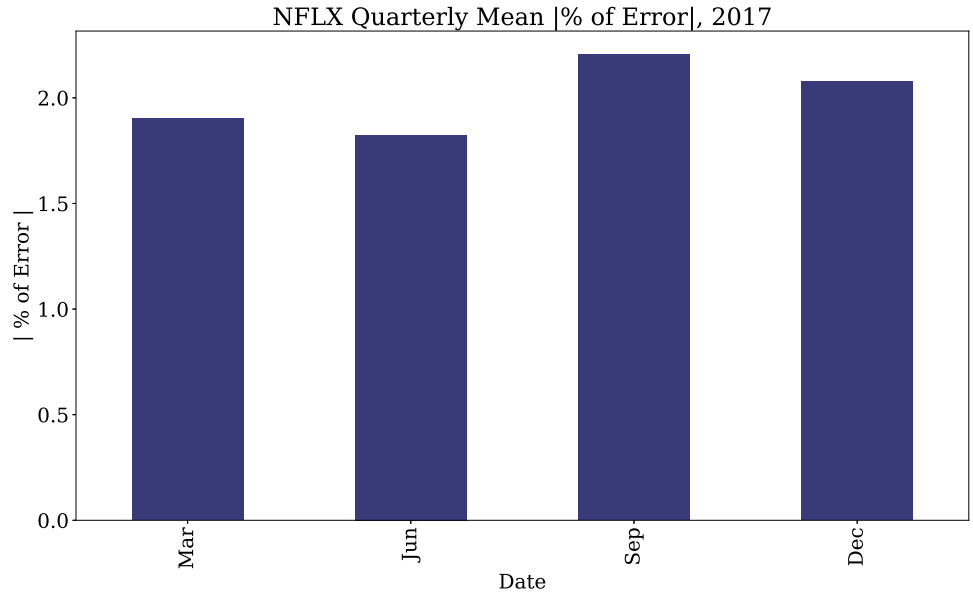

Figure 47. NFLX Quarterly Mean |% of Error|, 2017

In Figures 42, 44 and 46, the comparison between the actual price and the predicted price is shown in three different periods of intervals. In most of the cases,

they are very close. In weekly mean values, as shown in Figure 43 highest error is 4 percent. In monthly mean values, highest error is 3 percent, as shown in Figure 45. In quarterly mean values, as shown in Figure 47 highest error is 2.5 percent. In the graphs, by comparing the actual price (weekly, monthly and quarterly mean) and predicted price (weekly, monthly and quarterly mean) all are so close that its in some graph its almost same. In the weekly means, we observed some uneven spike in the month of October. Those spikes are observed due to other impacting factors which are not considered in this study. Considering only the historical prices, all the values are less than 5 percent which is a good indicator that it validates the prediction.

4.2. Model Performance Validation

Model performance validation is very important for measuring the accuracy and reliability of the model. Validation of model performance is based on many indicators, some of which are mentioned below. A total of 251 days of stock prices (251 data points) are considered in validating the performance of the model. Table 1 below summarizes the validation of the model performance.

Table 1. Performance Analysis of the RNN-LSTM Model

Comparison		AMZN	FB	GOOGL	MSFT	NFLX
Count		251	251	251	251	251
|% of Error|	mean	1.54	1.53	1.31	1.27	2.00
	std	1.32	1.25	1.08	1.05	1.54
	variance	1.74	1.57	1.17	1.10	2.38
	min	0.00	0.03	0.04	0.02	0.00
	25%	0.54	0.61	0.48	0.56	0.80
	50%	1.26	1.12	1.05	0.98	1.65
	75%	2.19	2.27	1.86	1.77	2.96
	max	9.40	7.32	6.09	6.48	9.25
Coefficient of Determination		0.9643	0.9669	0.9550	0.9692	0.9589
Pearson's Cor. Coefficient		0.9822	0.9847	0.9784	0.9857	0.9797
Spearman's Rank Cor. Coefficient		0.9639	0.9674	0.9663	0.9765	0.9694
Explained Variance Score		0.9643	0.9673	0.9557	0.9693	0.9589

- |% of Error|: This measures the absolute value of the percentage of error. Some sub categories are mentioned below.

    1. Mean: The mean represents the entire dataset with a single value which describes the average value of the entire dataset. The mean is essentially a model of entire data set. It is the is most common value in the data set. However, it is noticed that the mean is not often one of the actual values that anyone can observed in data set. One of its important properties is that, it minimizes error in the prediction of any one value in given data set. That is, it is the value that produces the lowest amount of error from all other values in the data set. In Table 1, among the five stocks the mean

of the data set are between 1.27 and 2.0. The lower value of the mean the better the result which means that its almost close to validate the model performance.

2. Standard Deviation (std): Std is a measure that is used to quantify the amount of variation or dispersion of a set of data values. In Table 1, the std of the data set are ranges from 1.05 to 1.54. Among them MSFT has the lowest std of 1.05 and NFLX has the highest std of 1.54. It tells us how the measurements for a group are spread out from the mean or average or from expected value. A low std means that the most of the values are close to the average value and high std means that the most of the values are spread out from the mean. So, the lower the value is the better that means how close the prediction is.

3. Variance: Variance is the expectation of the squared deviation of a random variable from its mean. Informally, it measures how far a set of (random) numbers are spread out from their average value. In Table 1, the variance of the data set is ranges from 1.1 to 2.38. A low variance means that the most of the values are close to the average value and high variance means that the most of the values are spread out from the mean. That's why, low variance is expected to achieve better prediction and this model is consistently reliable in predicting the values.

4. Minimum (min): The minimum is the smallest value in the entire data set. In Table 1, the minimum of the data set are between 0 to 0.04. AMZN and NFLX both have the lowest value of 0 and GOOGL has the highest value of 0.04 among the lowest value. A lower minimum is expected to have better prediction. All the values are so low which is a good indicator that the prediction is almost accurate.

5. 25 %: This determines the highest value of 25 % of the ascending ordered of dataset. In Table 1, this is ranges from 0.48 to 0.8. A lower value is expected to have better prediction. All the values are low which is a good indicator that validae the model and reliable for prediction.

6. 50 %: This determines the highest value of 50 % of the ascending ordered dataset. In Table 1, this is ranges from 0.98 to 1.65. Among them MSFT has the lowest value of .98 and NFLX is the highest value of 1.65. A lower value is expected to have better prediction. All the values are low which is a good indicator that validae the model and reliable for prediction.

7. 75%: This determines the highest value of 75 % of the ascending ordered dataset. In Table 1, this is ranged from 1.77 to 2.96. A lower value is expected to have better prediction. All the values are low which is a good indicator that validae the model and reliable for prediction.

8. Maximum (max): The maximum is the largest value in the data set. In Table 1, the maximum of the data set is ranges from 6.09 to 9.40. A lower maximum value is expected to have better prediction. All the values are low which is a good indicator that validae the model and reliable for prediction.

- Coefficient of Determination (R-squared) : R-squared is a statistical measure which shows how close the data are fitted in a regression line. It is known as the coefficient of determination, or the coefficient of multiple determination for multiple regression.

  - 0 % indicates that the model explains none of the variability of the response data around its mean.

- 100 % indicates that the model explains all the variability of the response data around its mean.

A value between 0 and 1 indicates the Coefficient of Determination is calculated as the square of the coefficient between actual and predicted data. A value of 1 means every point on the regression line fits the data. It commonly shows how accurately a regression model can predict the future price of a stock.

In Table 1, this value of the data set is ranges from 0.9550 to 0.9692. That means most of the values are close to 1 which is 100 % that indicates that the model explains 100% of the variability of the response data around its mean and also the the prediction is almost accurate by using RNN-LSTM model.

- Pearson's Correlation Coefficient: Pearson correlation coefficient (PCC) referred to as Pearson's r, the Pearson product-moment correlation coefficient (PPMCC) or the bivariate correlation, is a measure of the linear correlation between two variables X and Y. Pearson's r, can take a range of values from +1 to -1. If the value is 0 that indicates that there is no association or relationship between the two variables. To indicate a positive association; the value should be greater than 0 which means that as the value of one variable increases, so does the value of the other variable; and as one variable decreases, so does the value of the other. An inverse relationship between the two variables shows a negative relationship (would be indicated by a Pearsons r value of less than 0).

  In Table 1, these values of the data set are in ranges from 0.9784 to 0.9857. That means most of the values are close to 1 or 100 % which indicates that the model explains all the variability of the response data around its mean. Most of the values are close to 1, which indicates a positive association: as the value of one variable (actual price) increases, so does the value of the other variable

(predicted price). Likewise a decrease in one is accompanied by a corresponding decrease in the other.

- Spearman's Rank Correlation Coefficient: Spearman's Rank correlation coefficient is a technique which can be used to summarize the strength and positive or negative direction of a relationship between two variables. The result will always be between 1 and - 1. It is a number that shows how closely two sets of data are linked- strongly positive or strongly negative. It is a non-parametric version of the Pearson measure of the strength and direction of the association that exists between two variables.

  In Table 1, this value of the data set is ranges from 0.9639 to 0.9765. All the values are near to 1. That means most of the values are strongly coupled in a positive direction of a relationship between two variables (actual and predicted price) which validate the prediction model.

- Explained Variance Score: If X_p is the predicted target output, X_a is the corresponding (actual) target output, Var is the variance, the square of the standard deviation, then explained variance score is estimated as below:

  Explained_Variance (X_a , X_p)= 1 (Var(X_a X_p) / Var(X_a))

  The best possible score is 1 and the other values are worse. In Table 1, the value of the data set is ranged from 0.9557 to 0.9693. That means most of the values are closer to 1 and the larger number gives a good fit of the regression line in the proportion to which this developed model accounts for the variation or dispersion of a given data set.

4.3. Statistical Hypothesis Test

A statistical hypothesis, sometimes called confirmatory data analysis, is a hypothesis that is testable on the basis of observing a process that is modeled via a set

of random variables. A statistical hypothesis test is a method of statistical inference. Commonly, two statistical data sets are compared, or a data set obtained by sampling is compared against a synthetic data set from an idealized model. A hypothesis (the sample distribution is a beta distribution) is proposed for the statistical relationship between the two data sets, and this is compared as an alternative to an idealized null hypothesis that proposes no relationship between two data sets. The comparison is deemed statistically significant if the relationship between the data sets would be an unlikely realization of the null hypothesis according to a threshold probability called the significance level. Hypothesis tests are used in determining what outcomes of a study would lead to a rejection of the null hypothesis for a pre-specified level of significance. The process of distinguishing between the null hypothesis and the alternative hypothesis is aided by identifying two conceptual types of errors (type 1 and type 2), and by specifying parametric limits on e.g. on how much type 1 error will be permitted [35]. There are many statistical hypothesis tests that can be checked in this work. First of all, it is very important to know the sample distribution and probability density function.

4.3.1. KDE with Histogram and Probability Density Function

In statistics, kernel density estimation (KDE) is a non-parametric way to estimate the probability density function of random variables. Kernel density estimation is a fundamental data smoothing problem where inferences about the population are made, based on a finite data sample [35].

A histogram is an accurate representation of the distribution of numerical data. It is an estimate of the probability distribution of a continuous variable (quantitative variable) and was first introduced by Karl Pearson. It differs from a bar graph, in the sense that a bar graph relates two variables, but a histogram relates only one. To construct a histogram, the first step is to "bin" the range of values, that is, divide

the entire range of values into a series of intervalsand then count how many values fall into each interval. The bins are usually specified as consecutive, non-overlapping intervals of a variable. The bins (intervals) must be adjacent, and are often (but are not required to be) of equal size [35].

In probability theory and statistics, beta distribution is a family of continuous probability distributions defined on the interval [0, 1] parametrized by two positive shape parameters, denoted by α and β, that appear as exponents of the random variable and control the shape of the distribution [35]

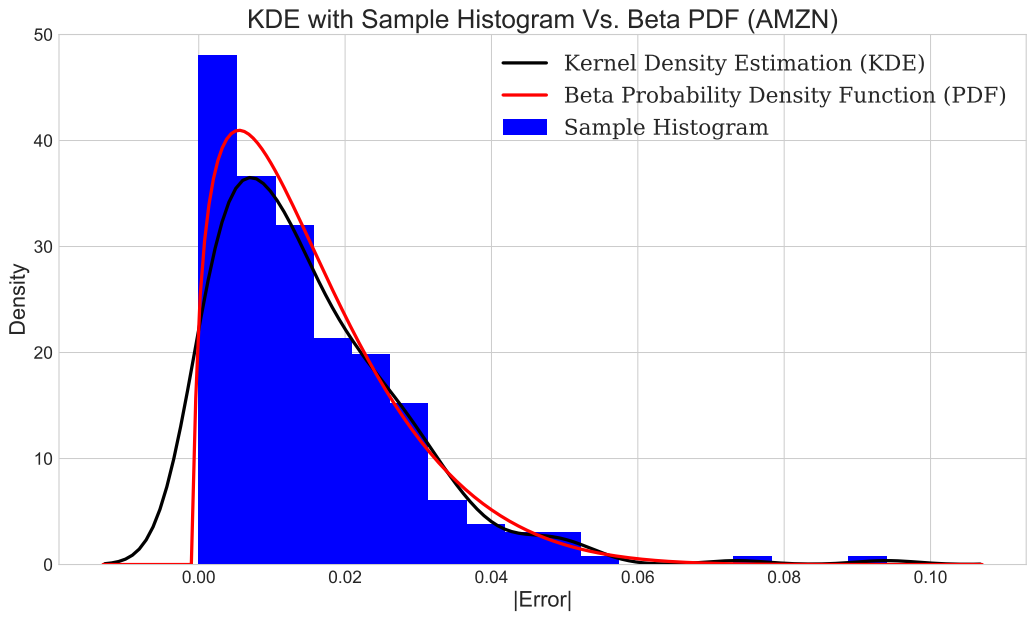

Figure 48. KDE with sample histogram vs. Beta PDF (AMZN)

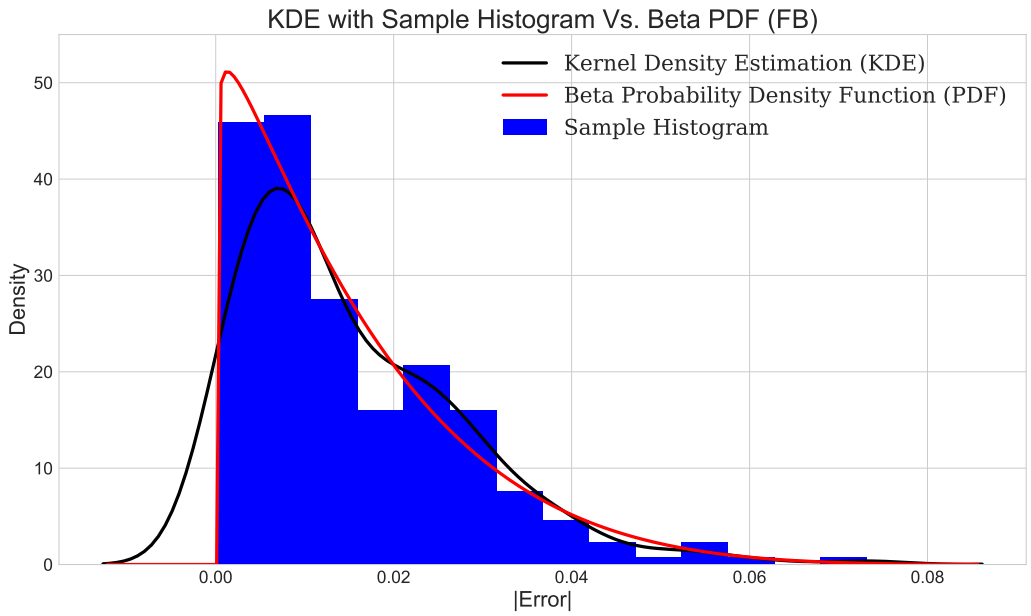

Figure 49. KDE with sample histogram vs. Beta PDF (FB)

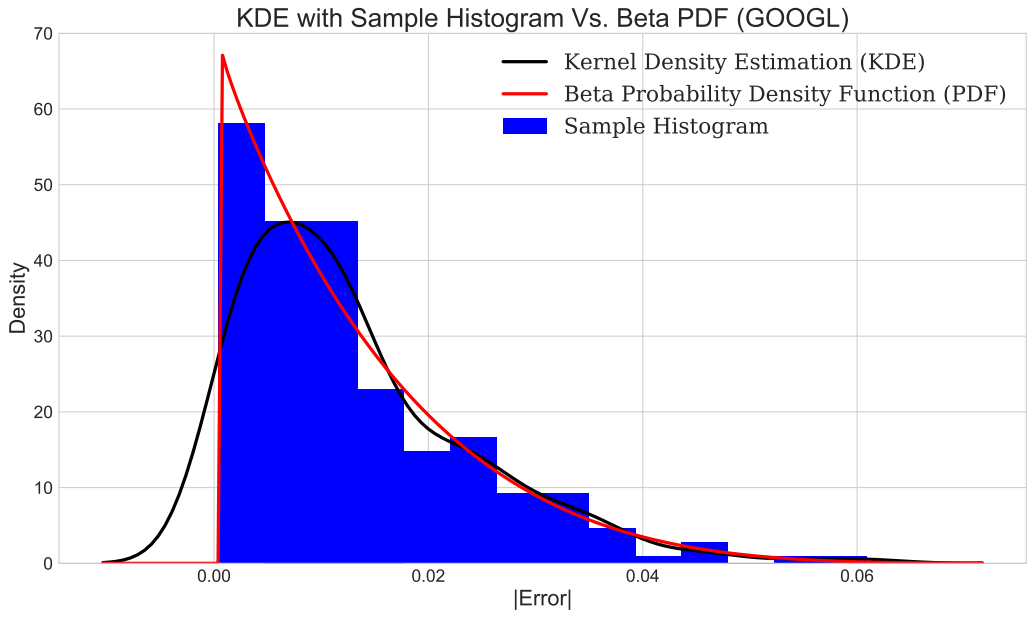

Figure 50. KDE with sample histogram vs. Beta PDF (GOOGL)

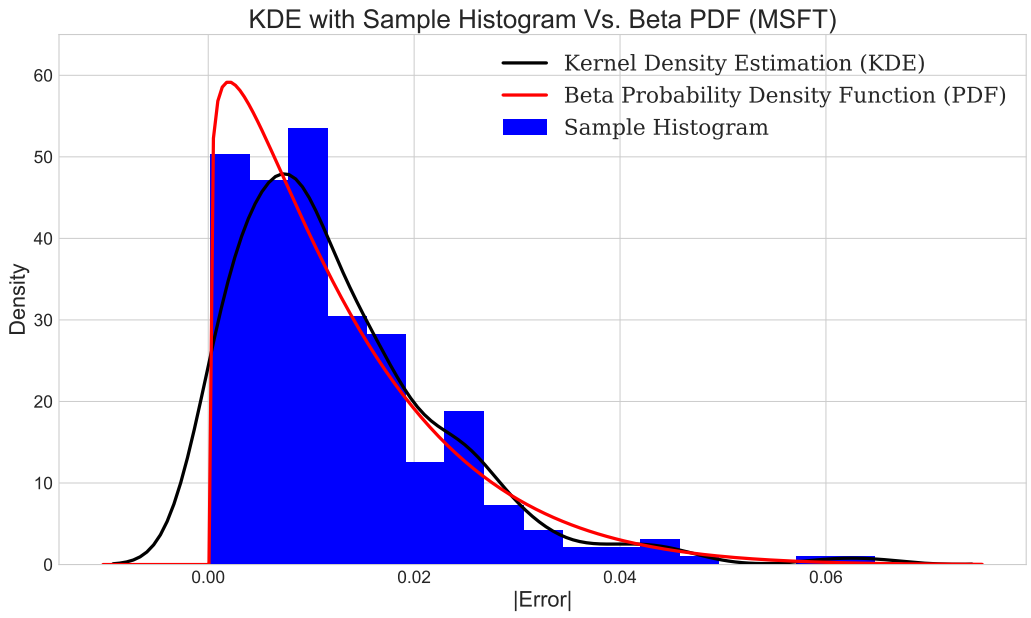

Figure 51. KDE with sample histogram vs. Beta PDF (MSFT)

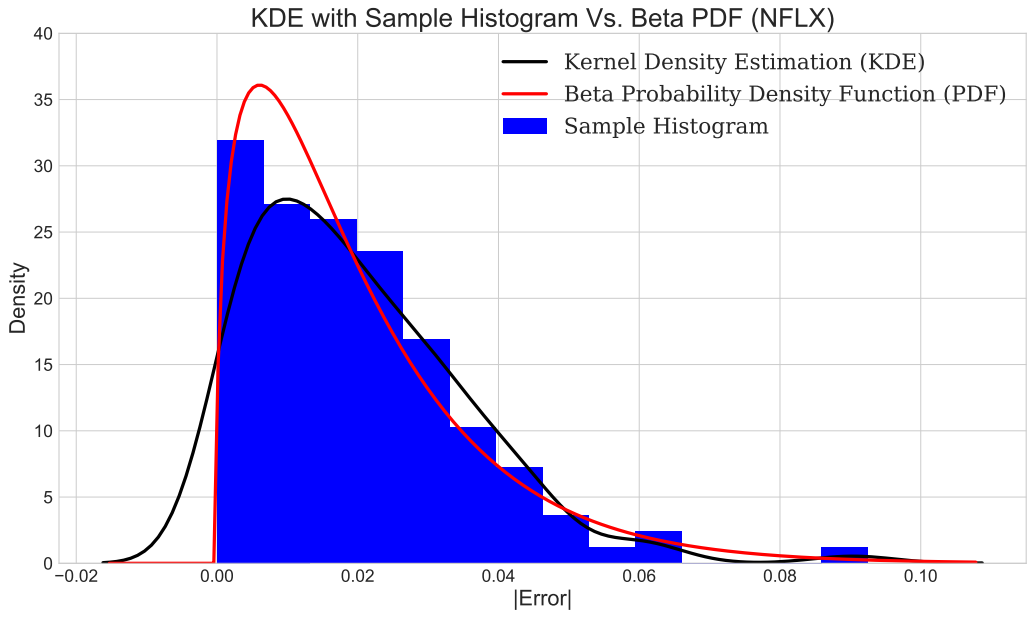

Figure 52. KDE with sample histogram vs. Beta PDF (NFLX)

Here, the python package Seaborn was used and a sample histogram with KDE was drawn. Then the sample histogram was fitted with beta distribution. In Figures 48, 49, 50, 51, and 52, Kernel Distribution Estimation (KDE) with sample histogram vs. Beta PDF was demonstrated on 5 stocks. A well-fitted beta distribution is observed in all 5 stocks. In a beta distribution, the probability of the lower values are higher compared to a normal distribution. As the absolute value of the error is fitted in beta distribution here, the probability of lower value of error is higher that domstrated a good prediction model.

The beta distribution is the conjugate pair probability distribution for bernoulli, binomial, negative binomial and geometric distribution. It is a suitable model for the random behavior of percentages and propositions.

4.3.2. Chi-Square Goodness of Fit Test

Chi-Square Goodness of Fit Test is used to determine whether sample data are consistent with a hypothesized distribution. The chi-square goodness of fit test is appropriate when the following conditions are met:

- The sampling method is simple random sampling.
- The variable under study is categorical.
- The expected value of the number of sample observations in each level of the variable is at least 5.

For this chi-square goodness of fit test, the hypotheses takes the following form.

- Null Hypothesis (H_0): The data are consistent with a beta distribution.
- Alternative Hypothesis (H_a): The data are not consistent with a beta distribution.

Table 2. Chi-Squared Goodness of Fit Test

Stock	Statistic
AMZN	2.8250633977200583
FB	2.5781650269970644
GOOGL	2.2426558774823278
MSFT	2.1644579617494211
NFLX	2.9700887211754785

A chi-square statistic is one way to show a relationship between two categorical variables. In statistics, there are two types of variables: numerical (countable) variables and non-numerical (categorical) variables. The chi-squared statistic is a single number that tells how much difference exists between the observed counts and the counts would expect if there were no relationship at all in the population. A low value for chi-square means there is a high correlation between two sets of data. In theory, if observed (KDE) and expected (Beta PDF) values were equal (no difference) then chi-square would be zero, an event that is unlikely to happen in real life. Deciding whether a chi-square test statistic is large enough to indicate a statistically significant difference isn't as easy as it seems. If the chi-square statistic is low, we can conclude that there is a high correlation. In Table 2, the chi-square statistic are ranged from 2.16 to 2.97 which indicates that they are highly related. Here, all the values are lower than critical value. Therefore, null hypothesis H_0 cannot be rejected.

4.3.3. Wilcoxon Signed Rank Test

The Wilcoxon signed-rank test is a non-parametric statistical hypothesis test used to compare two related samples, matched samples, or repeated measurements of a single sample, to assess whether their population mean ranks differ [35]. Here, all the possible pairs of 5 stocks are tested using the Wilcoxon signed rant test. The results are shown in Table 3.

Table 3. Wilcoxon Signed Rank Test

Stock	Statistic	P value
AMZN vs. FB	2222.0	0.23013934044341644
AMZN vs. GOOGL	2616.0	0.24821307898992373
AMZN vs. MSFT	2665.0	0.18192622259449343
AMZN vs. NFLX	3051.0	0.70545698611127339
FB vs. GOOGL	2730.0	1.0
FB vs. MSFT	3107.5	0.85010673913852586
FB vs. NFLX	2725.0	0.44141832678205362
GOOGL vs. MSFT	3334.5	0.85268368433464259
GOOGL vs. NFLX	2830.5	0.44560053289767665
MSFT vs. NFLX	2369.0	0.32210200812889567

For this Wilcoxon Signed Rank Test, the hypotheses takes the following form.

- Null Hypothesis (H_0): The difference between the pairs follows a symmetric distribution around zero.
- Alternative Hypothesis (H_a): The difference between the pairs doesn't follow a symmetric distribution around zero.

In Table 3, the statistics column shows the sum of the ranks of the differences above or below zero, whichever is smaller. This value is ranges from 2222 to 3334. Here, a 2-sided p-value is shown and the ranges from 0. 18 to 1 which means 18 percent to 100 percent. It is commonly seen that if the p-value is less than 0.05, then the null hypothesis H_0 will be rejected.the model consistently reliable in predicting the values. Based on the results, the null hypothesis H_0 is not rejected for all ten pairs the difference between the pairs follows a symmetric distribution around zero then the Wilcoxon signed-rank test is appropriate. In other words, we can say that a better

result in Wilcoxon signed rank test indicates the stocks of any tested pair follow each other in both increasing and decreasing trends.

CHAPTER 5. CONCLUSION

In this study a model is developed to predict accurate future stock prices using a Recurrent Neural Network with an LSTM cell for time series data. The predicted stock price is very close to the true price. Five different companies stock prices were sampled in this experiment. The mean of the absolute value of the percentage of error is below 2 % based on this model. The absolute value of the error was also plotted and fitted with a beta distribution. To validate the performance of the model, the absolute value of the percentage of error, the coefficient of determine (R- squared), Pearsons correlation coefficient, Spearmans rank correlation coefficient, and explained variance error were calculated. The calculated results demonstrated a very strong relation between the actual and predicted stock price. To test the statistical hypothesis, KDE with a sample histogram was drawn with the beta probability density function. Then, the chi-square goodness of fit and the Wilcoxon signed rank tests were performed to support the hypothesis. Overall, we can conclude that we can accurately predict the stock price one day ahead using this developed RNN-LSTM model.

In future work, I would like to introduce how investors sentiments on sensitive financial disclosure, when incorporated into price changes, make for an unevenness that is often very difficult to justify based on historical prices only.

REFERENCES

[1] King, B. F. (1966). Market and industry factors in stock price behavior. the Journal of Business, 39(1), 139-190.

[2] Chen, N. F., Roll, R., and Ross, S. A. (1986). Economic forces and the stock market. Journal of business, 383-403.

[3] Patel, S. D., Quadros, D., Patil, V., and Saxena, H. (2017). Stock Prediction using Neural Networks. International Journal of Engineering and Management Research (IJEMR), 7(2), 490-493.

[4] Kohara, K., Ishikawa, T., Fukuhara, Y., and Nakamura, Y. (1997). Stock price prediction using prior knowledge and neural networks. Intelligent systems in accounting, finance and management, 6(1), 11-22.

[5] Kim, K. J., and Han, I. (2000). Genetic algorithms approach to feature discretization in artificial neural networks for the prediction of stock price index. Expert systems with Applications, 19(2), 125-132.

[6] Trafalis, T. B., and Ince, H. (2000). Support vector machine for regression and applications to financial forecasting. In Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on (Vol. 6, pp. 348-353). IEEE.

[7] Lee, J. W. (2001). Stock price prediction using reinforcement learning. In Industrial Electronics, 2001. Proceedings. ISIE 2001. IEEE International Symposium on (Vol. 1, pp. 690-695). IEEE.

[8] Chang, P. C., and Liu, C. H. (2008). A TSK type fuzzy rule-based system for stock price prediction. Expert Systems with applications, 34(1), 135-144.

[9] Tsai, C. F., and Wang, S. P. (2009, March). Stock price forecasting by hybrid machine learning techniques. In Proceedings of the International MultiConference of Engineers and Computer Scientists (Vol. 1, No. 755, p. 60).

[10] Patel, J., Shah, S., Thakkar, P., and Kotecha, K. (2015). Predicting stock market index using fusion of machine learning techniques. Expert Systems with Applications, 42(4), 2162-2172.en

[11] Heaton, J. B., Polson, N. G., and Witte, J. H. (2016). Deep learning in finance. arXiv preprint arXiv:1602.06561.

[12] Li, X., Xie, H., Wang, R., Cai, Y., Cao, J., Wang, F., ... and Deng, X. (2016). Empirical analysis: stock market prediction via extreme learning machine. Neural Computing and Applications, 27(1), 67-78.

[13] Mullainathan, S., and Spiess, J. (2017). Machine learning: an applied econometric approach. Journal of Economic Perspectives, 31(2), 87-106.

[14] Rather, A. M., Agarwal, A., and Sastry, V. N. (2015). Recurrent neural network and a hybrid model for prediction of stock returns. Expert Systems with Applications, 42(6), 3234-3241.

[15] Jeon, S., Hong, B., and Chang, V. (2018). Pattern graph tracking-based stock price prediction using big data. Future Generation Computer Systems, 80, 171-187.

[16] Lee, M. S., Ahn, C. H., Kwahk, K. Y., and Ahn, H. (2018). Stock Market Prediction Using Convolutional Neural Network That Learns from a Graph. World Academy of Science, Engineering and Technology, International Journal of Business and Economics Engineering, 5(1).

[17] Tenti, P. (2017). Forecasting foreign exchange rates using recurrent neural networks. In Artificial Intelligence Applications on Wall Street (pp. 567-580). Routledge.

[18] Graves, A., Mohamed, A. R., and Hinton, G. (2013, May). Speech recognition with deep recurrent neural networks. In Acoustics, speech and signal processing (icassp), 2013 ieee international conference on (pp. 6645-6649). IEEE.

[19] Alpaydin, E. (2014). Introduction to machine learning. MIT press.

[20] Machine Learning, [online], http://www.expertsystem.com/machine-learning-definition/, last access date: 18th May,2018.

[21] Types of Machine Learning, [online], https://medium.com/deep-math-machine-learning-ai/different-types-of-machine-learning-and-their-types-34760b9128a2/,last access date: 18th May,2018.

[22] Machine Learning Model Assessments, [online], https://en.wikipedia.org/wiki Machine_learning_Model_assessments, last access date: 18th May, 2018.

[23] Kotsiantis, S. B., Zaharakis, I., and Pintelas, P. (2007). Supervised machine learning: A review of classification techniques. Emerging artificial intelligence applications in computer engineering, 160, 3-24.

[24] Blum, A. (2007). Machine learning theory. Carnegie Melon Universit, School of Computer Science, 26.

[25] Time series,[online], https://en.wikipedia.org/wiki/Time_series,last access date: 24th May,2018.

[26] Palit, A. K., and Popovic, D. (2006). Computational intelligence in time series forecasting: theory and engineering applications. Springer Science and Business Media.

[27] Bontempi, G., Taieb, S. B., and Le Borgne, Y. A. (2012, July). Machine learning strategies for time series forecasting. In European Business Intelligence Summer School (pp. 62-77). Springer, Berlin, Heidelberg.

[28] 6 Types of Artificial Neural Networks Currently Being Used in Machine Learning, [online], https://analyticsindiamag.com/6-types-of-artificial-neural-networks-currently-being-used-in-todays-technology/,last access date: 25th May,2018.

[29] https://Recurrent Neural Network (RNN),[online],www.techopedia.com/definition /32834/recurrent-neural-network-rnn, last access date: 25th May,2018.

[30] Fundamentals of Deep Learning Introduction to Recurrent Neural Networks,[online], https://www.analyticsvidhya.com/blog/2017/12/introduction-to-recurrent-neural-networks/, last access date: 26th May,2018.

[31] Long Short-Term Memory, [online], https://en.wikipedia.org/wiki/Long_short-term_memory, last access date: 28th May,2018.

[32] Yahoo Finance, [online], https://finance.yahoo.com/, last access date: 1st Jan,2018.

[33] Python Pandas, [online], https://pandas.pydata.org/, last acess date: 12 June, 2018

[34] Tensorflow, [online], https://www.tensorflow.org/, last acess date: 12 June, 2018

[35] Wikipedia, [online], https://en.wikipedia.org/, last access date: 1st Jan,2018.

APPENDIX A. PYTHON CODE

NFLX.ipynb File: The complete RNN-LSTM Model code for NFLX. Here, only RNN-LSTM model of NFLX is shown. For other stocks, only the read_csv file will be changed.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline

stock = pd.read_csv('NFLXdata.csv',index_col='Date',parse_dates=True)
stock.info()

hold=np.zeros((251,2),dtype=np.float32)
hold1=np.ones((251,1),dtype=np.chararray)

for j in range(1008,1259,+1):
    hold[j-1008,0]=stock.values[j]
    hold1[j-1008]=stock.index[j]

stock.plot()
stock.info()
train_set=stock[0:1008]

#sklearn
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
train_scaled = scaler.fit_transform(train_set)
#test_scaled = scaler.transform(test_set)
#batch Defination
```

```
def next_batch(training_data,batch_size,steps):

    rand_start = np.random.randint(0,len(training_data)-steps)

    y_batch = np.array(training_data[rand_start:rand_start+steps+1]).
    reshape(1,steps+1)

    return y_batch[:, :-1].reshape(-1, steps, 1), y_batch[:, 1:].
    reshape(-1, steps, 1)
#RNN Setup
import tensorflow as tf
#Tuning parameters

num_inputs = 1
num_time_steps = 1
num_neurons = 500
num_outputs = 1
learning_rate = 0.002
num_train_iterations = 4000
batch_size = 4
#placeholder
X = tf.placeholder(tf.float32, [None, num_time_steps, num_inputs])
y = tf.placeholder(tf.float32, [None, num_time_steps, num_outputs])
#LSTM Cell
cell = tf.contrib.rnn.OutputProjectionWrapper(
    tf.contrib.rnn.BasicLSTMCell(num_units=num_neurons, ...
        activation=tf.nn.relu),
    output_size=num_outputs)
#Dynamic RNN
outputs, states = tf.nn.dynamic_rnn(cell, X, dtype = tf.float32)
#Loss
loss = tf.reduce_mean(tf.square(outputs - y))
```

```
optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate)
train = optimizer.minimize(loss)
#initializer
init = tf.global_variables_initializer()
#saver
saver = tf.train.Saver()
#gpu_options
gpu_options = tf.GPUOptions(per_process_gpu_memory_fraction=0.9)
#MSE CAlculation
with tf.Session(config=tf.ConfigProto(gpu_options=gpu_options)) as sess:
    sess.run(init)

    for iteration in range(num_train_iterations):

        X_batch, y_batch = ...
           next_batch(train_scaled,batch_size,num_time_steps)
        sess.run(train, feed_dict={X: X_batch, y: y_batch})

        if iteration % 100 == 0:

            mse = loss.eval(feed_dict={X: X_batch, y: y_batch})
            print(iteration, "\tMSE:", mse)

    saver.save(sess, "./rnn_stock_time_series_model")
#Session Run
with tf.Session() as sess:

    saver.restore(sess, "./rnn_stock_time_series_model")

    train_seed = list(train_scaled[-num_time_steps:])
    for iteration in range(num_time_steps):
        X_batch = np.array(train_seed[-num_time_steps:]).
```

```
        reshape(1, num_time_steps, 1)
        y_pred = sess.run(outputs, feed_dict={X: X_batch})
        train_seed.append(y_pred[0, -1, 0])
#Print Train_seed
train_seed
#results
results = ...
   scaler.inverse_transform(np.array(train_seed[num_time_steps:]).
reshape(num_time_steps,1))

#for k in range(0,num_time_steps,+1):
    #hold[k,1]=results[k]

hold[0,1]=results

hold



for i in range(1,251,+1):
    hold
    print("Pass ",i )

    train_set=stock[i:1008+i]

    #sklearn
    from sklearn.preprocessing import MinMaxScaler
    scaler = MinMaxScaler()
    train_scaled = scaler.fit_transform(train_set)
    #test_scaled = scaler.transform(test_set)
    #batch Defination
    #MSE CAlculation
    with tf.Session(config=tf.ConfigProto(gpu_options=gpu_options)) ...
```

```
    as sess:
     sess.run(init)

     for iteration in range(num_train_iterations):

         X_batch, y_batch = ...
            next_batch(train_scaled,batch_size,num_time_steps)
         sess.run(train, feed_dict={X: X_batch, y: y_batch})

         if iteration % 100 == 0:

             mse = loss.eval(feed_dict={X: X_batch, y: y_batch})
             print(iteration , "\tMSE:", mse)



     saver.save(sess, "./rnn_stock_time_series_model")
#Session Run
with tf.Session() as sess:

     saver.restore(sess, "./rnn_stock_time_series_model")

     train_seed = list(train_scaled[-num_time_steps:])
     for iteration in range(num_time_steps):
         X_batch = np.array(train_seed[-num_time_steps:]).
         reshape(1, num_time_steps, 1)
         y_pred = sess.run(outputs, feed_dict={X: X_batch})
         train_seed.append(y_pred[0, -1, 0])
#Print Train_seed
train_seed
#results
results = ...
    scaler.inverse_transform(np.array(train_seed[num_time_steps:]).
```

```
    reshape(num_time_steps,1))
    hold[i,1]=results


np.set_printoptions(threshold=np.inf)
hold

df=pd.DataFrame(data=hold[0:,0:],index=hold1[0:,0],
columns=['Actual','Generated'])
df=df.assign(Δ=(df.Actual-df.Generated)/df.Actual)
df=df.assign(absoluteΔ=df.Δ.abs())

writer = pd.ExcelWriter('NFLX.xlsx')
df.to_excel(writer,'NFLX_Output')
Sum.to_excel(writer,'NFLX_Sum')
writer.save()
```

ResultsPlot.ipynb File: The code to generate the weekly, monthly and quarterly results. All the output of RNN-LSTM model were exported to Output.csv for convenience.

```
import numpy as np
import pandas as pd
import datetime
import matplotlib.pyplot as plt
%matplotlib inline
output = pd.read_csv('Output.csv',index_col='Date',parse_dates=True)
output.info()

params = {'legend.fontsize': 40,
          'legend.handlelength': 2}
```

```
#Monthly |% of Error|

AMZN=output['AMZN (|% of Error|)'].resample(rule='M').mean().
plot(kind='bar',figsize=(16,9),
fontsize=20,cmap='tab20b')
plt.title('AMZN Monthly Mean |% of Error|, 2017', fontsize=24)
plt.xlabel('Date', fontsize=20)
plt.ylabel('| % of Error |', fontsize=20)
plt.rcParams['font.family'] = 'serif'
f = lambda x: datetime.datetime.strptime(x, '%Y-%m-%d ...
    %H:%M:%S').strftime('%b')
AMZN.set_xticklabels([ f(x.get_text()) for x in AMZN.get_xticklabels()])

plt.savefig('AMZN1.eps')
plt.savefig('AMZN1.jpg')


FB=output['FB (|% of Error|)'].resample(rule='M').mean().
plot(kind='bar',figsize=(16,9),
fontsize=20,cmap='tab20b')
plt.title('FB Monthly Mean |% of Error|, 2017', fontsize=24)
plt.xlabel('Date', fontsize=20)
plt.ylabel('| % of Error |', fontsize=20)
plt.rcParams['font.family'] = 'serif'
f = lambda x: datetime.datetime.strptime(x, '%Y-%m-%d ...
    %H:%M:%S').strftime('%b')
FB.set_xticklabels([ f(x.get_text()) for x in FB.get_xticklabels()])

plt.savefig('FB1.eps')
plt.savefig('FB1.jpg')
```

```
GOOGL=output['GOOGL (|% of Error|)'].resample(rule='M').mean().
plot(kind='bar',figsize=(16,9),
fontsize=20,cmap='tab20b')
plt.title('GOOGL Monthly Mean |% of Error|, 2017', fontsize=24)
plt.xlabel('Date', fontsize=20)
plt.ylabel('| % of Error |', fontsize=20)
plt.rcParams['font.family'] = 'serif'
f = lambda x: datetime.datetime.strptime(x, '%Y-%m-%d ...
    %H:%M:%S').strftime('%b')
GOOGL.set_xticklabels([ f(x.get_text()) for x in ...
    GOOGL.get_xticklabels()])

plt.savefig('GOOGL1.eps')
plt.savefig('GOOGL1.jpg')

MSFT=output['MSFT (|% of Error|)'].resample(rule='M').mean().
plot(kind='bar',figsize=(16,9),
fontsize=20,cmap='tab20b')
plt.title('MSFT Monthly Mean |% of Error|, 2017', fontsize=24)
plt.xlabel('Date', fontsize=20)
plt.ylabel('| % of Error |', fontsize=20)
plt.rcParams['font.family'] = 'serif'
f = lambda x: datetime.datetime.strptime(x, '%Y-%m-%d ...
    %H:%M:%S').strftime('%b')
MSFT.set_xticklabels([ f(x.get_text()) for x in MSFT.get_xticklabels()])

plt.savefig('MSFT1.eps')
plt.savefig('MSFT1.jpg')

NFLX=output['NFLX (|% of Error|)'].resample(rule='M').mean().
plot(kind='bar',figsize=(16,9),
fontsize=20,cmap='tab20b')
```

```
plt.title('NFLX Monthly Mean |% of Error|, 2017', fontsize=24)
plt.xlabel('Date', fontsize=20)
plt.ylabel('| % of Error |', fontsize=20)
plt.rcParams['font.family'] = 'serif'
f = lambda x: datetime.datetime.strptime(x, '%Y-%m-%d ...
    %H:%M:%S').strftime('%b')
NFLX.set_xticklabels([ f(x.get_text()) for x in NFLX.get_xticklabels()])

plt.savefig('NFLX1.eps')
plt.savefig('NFLX1.jpg')

#Weekly |% of Error|

AMZN=output['AMZN (|% of Error|)'].resample(rule='W').mean().
plot(kind='bar',figsize=(16,9),
fontsize=20,cmap='tab20b')
plt.title('AMZN Weekly Mean |% of Error|, 2017', fontsize=24)
plt.xlabel('Date', fontsize=20)
plt.ylabel('| % of Error |', fontsize=20)
plt.rcParams['font.family'] = 'serif'
f = lambda x: datetime.datetime.strptime(x, '%Y-%m-%d ...
    %H:%M:%S').strftime('%b')
AMZN.set_xticklabels([ f(x.get_text()) for x in AMZN.get_xticklabels()])

plt.savefig('AMZN2.eps')
plt.savefig('AMZN2.jpg')

FB=output['FB (|% of ...
    Error|)'].resample(rule='W').mean().plot(kind='bar',figsize=(16,9),
fontsize=20,cmap='tab20b')
plt.title('FB Weekly Mean |% of Error|, 2017', fontsize=24)
plt.xlabel('Date', fontsize=20)
```

```
plt.ylabel('| % of Error |', fontsize=20)
plt.rcParams['font.family'] = 'serif'
f = lambda x: datetime.datetime.strptime(x, '%Y-%m-%d ...
    %H:%M:%S').strftime('%b')
FB.set_xticklabels([ f(x.get_text()) for x in FB.get_xticklabels()])

plt.savefig('FB2.eps')
plt.savefig('FB2.jpg')

GOOGL=output['GOOGL (|% of Error|)'].resample(rule='W').mean().
plot(kind='bar',figsize=(16,9),
fontsize=20,cmap='tab20b')
plt.title('GOOGL Weekly Mean |% of Error|, 2017', fontsize=24)
plt.xlabel('Date', fontsize=20)
plt.ylabel('| % of Error |', fontsize=20)
plt.rcParams['font.family'] = 'serif'
f = lambda x: datetime.datetime.strptime(x, '%Y-%m-%d ...
    %H:%M:%S').strftime('%b')
GOOGL.set_xticklabels([ f(x.get_text()) for x in ...
    GOOGL.get_xticklabels()])

plt.savefig('GOOGL2.eps')
plt.savefig('GOOGL2.jpg')

MSFT=output['MSFT (|% of Error|)'].resample(rule='W').mean().
plot(kind='bar',figsize=(16,9),
fontsize=20,cmap='tab20b')
plt.title('MSFT Weekly Mean |% of Error|, 2017', fontsize=24)
plt.xlabel('Date', fontsize=20)
plt.ylabel('| % of Error |', fontsize=20)
plt.rcParams['font.family'] = 'serif'
f = lambda x: datetime.datetime.strptime(x, '%Y-%m-%d ...
```

```
    %H:%M:%S').strftime('%b')
MSFT.set_xticklabels([ f(x.get_text()) for x in MSFT.get_xticklabels()])

plt.savefig('MSFT2.eps')
plt.savefig('MSFT2.jpg')

NFLX=output['NFLX (|% of Error|)'].resample(rule='W').mean().
plot(kind='bar',figsize=(16,9),
fontsize=20,cmap='tab20b')
plt.title('NFLX Weekly Mean |% of Error|, 2017', fontsize=24)
plt.xlabel('Date', fontsize=20)
plt.ylabel('| % of Error |', fontsize=20)
plt.rcParams['font.family'] = 'serif'
f = lambda x: datetime.datetime.strptime(x, '%Y-%m-%d ...
    %H:%M:%S').strftime('%b')
NFLX.set_xticklabels([ f(x.get_text()) for x in NFLX.get_xticklabels()])

plt.savefig('NFLX2.eps')
plt.savefig('NFLX2.jpg')

#Quarterly |% of Error|

AMZN=output['AMZN (|% of Error|)'].resample(rule='Q').mean().
plot(kind='bar',figsize=(16,9),
fontsize=20,cmap='tab20b')
plt.title('AMZN Quarterly Mean |% of Error|, 2017', fontsize=24)
plt.xlabel('Date', fontsize=20)
plt.ylabel('| % of Error |', fontsize=20)
plt.rcParams['font.family'] = 'serif'
f = lambda x: datetime.datetime.strptime(x, '%Y-%m-%d ...
    %H:%M:%S').strftime('%b')
AMZN.set_xticklabels([ f(x.get_text()) for x in AMZN.get_xticklabels()])
```

```
plt.savefig('AMZN3.eps')
plt.savefig('AMZN3.jpg')

FB=output['FB (|% of Error|)'].resample(rule='Q').mean().
plot(kind='bar',figsize=(16,9),
fontsize=20,cmap='tab20b')
plt.title('FB Quarterly Mean |% of Error|, 2017', fontsize=24)
plt.xlabel('Date', fontsize=20)
plt.ylabel('| % of Error |', fontsize=20)
plt.rcParams['font.family'] = 'serif'
f = lambda x: datetime.datetime.strptime(x, '%Y-%m-%d ...
    %H:%M:%S').strftime('%b')
FB.set_xticklabels([ f(x.get_text()) for x in FB.get_xticklabels()])

plt.savefig('FB3.eps')
plt.savefig('FB3.jpg')

GOOGL=output['GOOGL (|% of Error|)'].resample(rule='Q').mean().
plot(kind='bar',figsize=(16,9),
fontsize=20,cmap='tab20b')
plt.title('GOOGL Quarterly Mean |% of Error|, 2017', fontsize=24)
plt.xlabel('Date', fontsize=20)
plt.ylabel('| % of Error |', fontsize=20)
plt.rcParams['font.family'] = 'serif'
f = lambda x: datetime.datetime.strptime(x, '%Y-%m-%d ...
    %H:%M:%S').strftime('%b')
GOOGL.set_xticklabels([ f(x.get_text()) for x in ...
    GOOGL.get_xticklabels()])

plt.savefig('GOOGL3.eps')
plt.savefig('GOOGL3.jpg')
```

```
MSFT=output['MSFT (|% of Error|)'].resample(rule='Q').mean().
plot(kind='bar',figsize=(16,9),
fontsize=20,cmap='tab20b')
plt.title('MSFT Quarterly Mean |% of Error|, 2017', fontsize=24)
plt.xlabel('Date', fontsize=20)
plt.ylabel('| % of Error |', fontsize=20)
plt.rcParams['font.family'] = 'serif'
f = lambda x: datetime.datetime.strptime(x, '%Y-%m-%d ...
    %H:%M:%S').strftime('%b')
MSFT.set_xticklabels([ f(x.get_text()) for x in MSFT.get_xticklabels()])

plt.savefig('MSFT3.eps')
plt.savefig('MSFT3.jpg')

NFLX=output['NFLX (|% of Error|)'].resample(rule='Q').mean().
plot(kind='bar',figsize=(16,9),
fontsize=20,cmap='tab20b')
plt.title('NFLX Quarterly Mean |% of Error|, 2017', fontsize=24)
plt.xlabel('Date', fontsize=20)
plt.ylabel('| % of Error |', fontsize=20)
plt.rcParams['font.family'] = 'serif'
f = lambda x: datetime.datetime.strptime(x, '%Y-%m-%d ...
    %H:%M:%S').strftime('%b')
NFLX.set_xticklabels([ f(x.get_text()) for x in NFLX.get_xticklabels()])

plt.savefig('NFLX3.eps')
plt.savefig('NFLX3.jpg')

#Comparison (Monthly)

AMZN=output[['AMZN (Actual)', ...
```

```
    'AMZN(Predicted)']].resample(rule='M').mean().
plot(kind='bar',figsize=(16,9),
fontsize=20,cmap='tab20b')
plt.title('Comparison of AMZN Monthly Mean, 2017', fontsize=24)
plt.xlabel('Date', fontsize=20)
plt.ylabel('Close Price', fontsize=20)
plt.legend(fontsize=20)
plt.rcParams['font.family'] = 'serif'
f = lambda x: datetime.datetime.strptime(x, '%Y-%m-%d ...
    %H:%M:%S').strftime('%b')
AMZN.set_xticklabels([ f(x.get_text()) for x in AMZN.get_xticklabels()])

plt.savefig('AMZN4.eps')
plt.savefig('AMZN4.jpg')

FB=output[['FB (Actual)', 'FB(Predicted)']].resample(rule='M').mean().
plot(kind='bar',figsize=(16,9),
fontsize=20,cmap='tab20b')
plt.title('Comparison of FB Monthly Mean, 2017', fontsize=24)
plt.xlabel('Date', fontsize=20)
plt.ylabel('Close Price', fontsize=20)
plt.legend(fontsize=20)
plt.rcParams['font.family'] = 'serif'
f = lambda x: datetime.datetime.strptime(x, '%Y-%m-%d ...
    %H:%M:%S').strftime('%b')
FB.set_xticklabels([ f(x.get_text()) for x in FB.get_xticklabels()])

plt.savefig('FB4.eps')
plt.savefig('FB4.jpg')

GOOGL=output[['GOOGL (Actual)', ...
    'GOOGL(Predicted)']].resample(rule='M').mean().
```

```
plot(kind='bar',figsize=(16,9),fontsize=20,cmap='tab20b')
plt.title('Comparison of GOOGL Monthly Mean, 2017', fontsize=24)
plt.xlabel('Date', fontsize=20)
plt.ylabel('Close Price', fontsize=20)
plt.legend(fontsize=20)
plt.rcParams['font.family'] = 'serif'
f = lambda x: datetime.datetime.strptime(x, '%Y-%m-%d ...
    %H:%M:%S').strftime('%b')
GOOGL.set_xticklabels([ f(x.get_text()) for x in ...
    GOOGL.get_xticklabels()])

plt.savefig('GOOGL4.eps')
plt.savefig('GOOGL4.jpg')

MSFT=output[['MSFT (Actual)', 'MSFT ...
    (Predicted)']].resample(rule='M').mean().
plot(kind='bar',figsize=(16,9),fontsize=20,cmap='tab20b')
plt.title('Comparison of MSFT Monthly Mean, 2017', fontsize=24)
plt.xlabel('Date', fontsize=20)
plt.ylabel('Close Price', fontsize=20)
plt.legend(fontsize=20)
plt.rcParams['font.family'] = 'serif'
f = lambda x: datetime.datetime.strptime(x, '%Y-%m-%d ...
    %H:%M:%S').strftime('%b')
MSFT.set_xticklabels([ f(x.get_text()) for x in MSFT.get_xticklabels()])

plt.savefig('MSFT4.eps')
plt.savefig('MSFT4.jpg')

NFLX=output[['NFLX (Actual)', 'NFLX ...
    (Predicted)']].resample(rule='M').mean().
plot(kind='bar',figsize=(16,9),fontsize=20,cmap='tab20b')
```

```
plt.title('Comparison of NFLX Monthly Mean, 2017', fontsize=24)
plt.xlabel('Date', fontsize=20)
plt.ylabel('Close Price', fontsize=20)
plt.legend(fontsize=20)
plt.rcParams['font.family'] = 'serif'
f = lambda x: datetime.datetime.strptime(x, '%Y-%m-%d ...
    %H:%M:%S').strftime('%b')
NFLX.set_xticklabels([ f(x.get_text()) for x in NFLX.get_xticklabels()])

plt.savefig('NFLX4.eps')
plt.savefig('NFLX4.jpg')

#Comparison (Weekly)

AMZN=output[['AMZN (Actual)', 'AMZN ...
    (Predicted)']].resample(rule='W').mean().
plot(kind='bar',figsize=(16,9),fontsize=20,cmap='tab20b')
plt.title('Comparison of AMZN Weekly Mean, 2017', fontsize=24)
plt.xlabel('Date', fontsize=20)
plt.ylabel('Close Price', fontsize=20)
plt.legend(fontsize=20)
plt.rcParams['font.family'] = 'serif'
f = lambda x: datetime.datetime.strptime(x, '%Y-%m-%d ...
    %H:%M:%S').strftime('%b')
AMZN.set_xticklabels([ f(x.get_text()) for x in AMZN.get_xticklabels()])

plt.savefig('AMZN5.eps')
plt.savefig('AMZN5.jpg')

FB=output[['FB (Actual)', 'FB (Predicted)']].resample(rule='W').mean().
plot(kind='bar',figsize=(16,9),fontsize=20,cmap='tab20b')
plt.title('Comparison of FB Weekly Mean, 2017', fontsize=24)
```

```
plt.xlabel('Date', fontsize=20)
plt.ylabel('Close Price', fontsize=20)
plt.legend(fontsize=20)
plt.rcParams['font.family'] = 'serif'
f = lambda x: datetime.datetime.strptime(x, '%Y-%m-%d ...
    %H:%M:%S').strftime('%b')
FB.set_xticklabels([ f(x.get_text()) for x in FB.get_xticklabels()])

plt.savefig('FB5.eps')
plt.savefig('FB5.jpg')

GOOGL=output[['GOOGL (Actual)', 'GOOGL ...
    (Predicted)']].resample(rule='W').mean().
plot(kind='bar',figsize=(16,9),fontsize=20,cmap='tab20b')
plt.title('Comparison of GOOGL Weekly Mean, 2017', fontsize=24)
plt.xlabel('Date', fontsize=20)
plt.ylabel('Close Price', fontsize=20)
plt.legend(fontsize=20)
plt.rcParams['font.family'] = 'serif'
f = lambda x: datetime.datetime.strptime(x, '%Y-%m-%d ...
    %H:%M:%S').strftime('%b')
GOOGL.set_xticklabels([ f(x.get_text()) for x in ...
    GOOGL.get_xticklabels()])

plt.savefig('GOOGL5.eps')
plt.savefig('GOOGL5.jpg')

MSFT=output[['MSFT (Actual)', 'MSFT ...
    (Predicted)']].resample(rule='W').mean().
plot(kind='bar',figsize=(16,9),fontsize=20,cmap='tab20b')
plt.title('Comparison of MSFT Weekly Mean, 2017', fontsize=24)
plt.xlabel('Date', fontsize=20)
```

```
plt.ylabel('Close Price', fontsize=20)
plt.legend(fontsize=20)
plt.rcParams['font.family'] = 'serif'
f = lambda x: datetime.datetime.strptime(x, '%Y-%m-%d ...
    %H:%M:%S').strftime('%b')
MSFT.set_xticklabels([ f(x.get_text()) for x in MSFT.get_xticklabels()])

plt.savefig('MSFT5.eps')
plt.savefig('MSFT5.jpg')

NFLX=output[['NFLX (Actual)', 'NFLX ...
    (Predicted)']].resample(rule='W').mean().
plot(kind='bar',figsize=(16,9),fontsize=20,cmap='tab20b')
plt.title('Comparison of NFLX Weekly Mean, 2017', fontsize=24)
plt.xlabel('Date', fontsize=20)
plt.ylabel('Close Price', fontsize=20)
plt.legend(fontsize=20)
plt.rcParams['font.family'] = 'serif'
f = lambda x: datetime.datetime.strptime(x, '%Y-%m-%d ...
    %H:%M:%S').strftime('%b')
NFLX.set_xticklabels([ f(x.get_text()) for x in NFLX.get_xticklabels()])

plt.savefig('NFLX5.eps')
plt.savefig('NFLX5.jpg')

#Comparison (Quarterly)

AMZN=output[['AMZN (Actual)', 'AMZN ...
    (Predicted)']].resample(rule='Q').mean().
plot(kind='bar',figsize=(16,9),fontsize=20,cmap='tab20b')
plt.title('Comparison of AMZN Quarterly Mean, 2017', fontsize=24)
plt.xlabel('Date', fontsize=20)
```

```
plt.ylabel('Close Price', fontsize=20)
plt.legend(fontsize=20)
plt.rcParams['font.family'] = 'serif'
f = lambda x: datetime.datetime.strptime(x, '%Y-%m-%d ...
    %H:%M:%S').strftime('%b')
AMZN.set_xticklabels([ f(x.get_text()) for x in AMZN.get_xticklabels()])

plt.savefig('AMZN6.eps')
plt.savefig('AMZN6.jpg')

FB=output[['FB (Actual)', 'FB (Predicted)']].resample(rule='Q').mean().
plot(kind='bar',figsize=(16,9),fontsize=20,cmap='tab20b')
plt.title('Comparison of FB Quarterly Mean, 2017', fontsize=24)
plt.xlabel('Date', fontsize=20)
plt.ylabel('Close Price', fontsize=20)
plt.legend(fontsize=20)
plt.rcParams['font.family'] = 'serif'
f = lambda x: datetime.datetime.strptime(x, '%Y-%m-%d ...
    %H:%M:%S').strftime('%b')
FB.set_xticklabels([ f(x.get_text()) for x in FB.get_xticklabels()])

plt.savefig('FB6.eps')
plt.savefig('FB6.jpg')

GOOGL=output[['GOOGL (Actual)', 'GOOGL ...
    (Predicted)']].resample(rule='Q').mean().
plot(kind='bar',figsize=(16,9),fontsize=20,cmap='tab20b')
plt.title('Comparison of GOOGL Quarterly Mean, 2017', fontsize=24)
plt.xlabel('Date', fontsize=20)
plt.ylabel('Close Price', fontsize=20)
plt.legend(fontsize=20)
plt.rcParams['font.family'] = 'serif'
```

```
f = lambda x: datetime.datetime.strptime(x, '%Y-%m-%d ...
    %H:%M:%S').strftime('%b')
GOOGL.set_xticklabels([ f(x.get_text()) for x in ...
    GOOGL.get_xticklabels()])

plt.savefig('GOOGL6.eps')
plt.savefig('GOOGL6.jpg')

MSFT=output[['MSFT (Actual)', 'MSFT ...
    (Predicted)']].resample(rule='Q').mean().
plot(kind='bar',figsize=(16,9),fontsize=20,cmap='tab20b')
plt.title('Comparison of MSFT Quarterly Mean, 2017', fontsize=24)
plt.xlabel('Date', fontsize=20)
plt.ylabel('Close Price', fontsize=20)
plt.legend(fontsize=20)
plt.rcParams['font.family'] = 'serif'
f = lambda x: datetime.datetime.strptime(x, '%Y-%m-%d ...
    %H:%M:%S').strftime('%b')
MSFT.set_xticklabels([ f(x.get_text()) for x in MSFT.get_xticklabels()])

plt.savefig('MSFT6.eps')
plt.savefig('MSFT6.jpg')

NFLX=output[['NFLX (Actual)', 'NFLX ...
    (Predicted)']].resample(rule='Q').mean().
plot(kind='bar',figsize=(16,9),fontsize=20,cmap='tab20b')
plt.title('Comparison of NFLX Quarterly Mean, 2017', fontsize=24)
plt.xlabel('Date', fontsize=20)
plt.ylabel('Close Price', fontsize=20)
plt.legend(fontsize=20)
plt.rcParams['font.family'] = 'serif'
f = lambda x: datetime.datetime.strptime(x, '%Y-%m-%d ...
```

```
    %H:%M:%S').strftime('%b')
NFLX.set_xticklabels([ f(x.get_text()) for x in NFLX.get_xticklabels()])

plt.savefig('NFLX6.eps')
plt.savefig('NFLX6.jpg')
```

ResultsValidation.ipynb File: The code to validate the RNN-LSTM model which is develpoed to predict the stock price.

```
#Co-efficient of Determination (R-squared)

from sklearn.metrics import r2_score
a=output['AMZN (Actual)']
p=output['AMZN (Predicted)']
cod_AMZN = r2_score(a, p)
cod_AMZN

from sklearn.metrics import r2_score
a=output['FB (Actual)']
p=output['FB (Predicted)']
cod_FB = r2_score(a, p)
cod_FB

from sklearn.metrics import r2_score
a=output['GOOGL (Actual)']
p=output['GOOGL (Predicted)']
cod_GOOGL = r2_score(a, p)
cod_GOOGL

from sklearn.metrics import r2_score
a=output['MSFT (Actual)']
```

```
p=output['MSFT (Predicted)']
cod_MSFT = r2_score(a, p)
cod_MSFT

from sklearn.metrics import r2_score
a=output['NFLX (Actual)']
p=output['NFLX (Predicted)']
cod_NFLX = r2_score(a, p)
cod_NFLX

#Pearson's Co—relationCo—efficient

a=output['AMZN (Actual)']
p=output['AMZN (Predicted)']
pd.concat([a,p], axis=1).corr(method= "pearson")

a=output['FB (Actual)']
p=output['FB (Predicted)']
pd.concat([a,p], axis=1).corr(method= "pearson")

a=output['GOOGL (Actual)']
p=output['GOOGL(Predicted)']
pd.concat([a,p], axis=1).corr(method= "pearson")

a=output['MSFT (Actual)']
p=output['MSFT (Predicted)']
pd.concat([a,p], axis=1).corr(method= "pearson")

a=output['NFLX (Actual)']
p=output['NFLX (Predicted)']
pd.concat([a,p], axis=1).corr(method= "pearson")
```

```
#Spearman's Rank Co—relation Co—efficient

a=output['AMZN (Actual)']
p=output['AMZN (Predicted)']
pd.concat([a,p], axis=1).corr(method= "spearman")

a=output['FB (Actual)']
p=output['FB (Predicted)']
pd.concat([a,p], axis=1).corr(method= "spearman")

a=output['GOOGL (Actual)']
p=output['GOOGL(Predicted)']
pd.concat([a,p], axis=1).corr(method= "spearman")

a=output['MSFT (Actual)']
p=output['MSFT (Predicted)']
pd.concat([a,p], axis=1).corr(method= "spearman")

a=output['NFLX (Actual)']
p=output['NFLX (Predicted)']
pd.concat([a,p], axis=1).corr(method= "spearman")

#Explained Variance Error

a=output['AMZN (Actual)']
p=output['AMZN (Predicted)']
explained_variance_score(a, p)

a=output['FB (Actual)']
p=output['FB (Predicted)']
explained_variance_score(a, p)
```

```
a=output['GOOGL (Actual)']
p=output['GOOGL(Predicted)']
explained_variance_score(a, p)

a=output['MSFT (Actual)']
p=output['MSFT (Predicted)']
explained_variance_score(a, p)

a=output['NFLX (Actual)']
p=output['NFLX (Predicted)']
explained_variance_score(a, p)

a=output['GOOGL (Actual)']
p=output['GOOGL(Predicted)']
explained_variance_score(a, p)
```

ResultsDistribution.ipynb File: The code to test the Hypothesis

```
#Data Collection
import numpy as np
import pandas as pd
import datetime
import matplotlib.pyplot as plt
%matplotlib inline
from scipy import stats, special, optimize
import seaborn as sns; sns.set()

output = pd.read_csv('Output.csv',index_col='Date',parse_dates=True)
```

```
#KDE with Histogram Vs. PDF

#Get Data
e=output['AMZN (|Error|)']

#Plot Data
sns.set(style='whitegrid', rc={"grid.linewidth": ...
    0.25,'figure.figsize':(16,9)})
sns.set_context("paper", font_scale=2)
sns.distplot(e,fit=stats.beta,norm_hist=True,
             hist_kws={"histtype": "bar", "linewidth": 3,"alpha": ...
                 0.15, "color": "b","label":
             "Sample Histogram"},
             fit_kws= {"color": "r", "lw": 3, "label": "Beta ...
                 Probability Density Function (PDF)"},
             kde_kws={"color": "k", "lw": 3, "label": "Kernel ...
                 Density Estimation (KDE)"})
plt.title('KDE with Sample Histogram Vs. Beta PDF (AMZN)', fontsize=24)
plt.ylabel('Density', fontsize=20)
plt.xlabel('|Error|', fontsize=20)
plt.rcParams['font.family'] = 'serif'
plt.legend(fontsize=20)
plt.ylim(0, 50)

#Save Plot
plt.savefig('AMZNPDF.eps')
plt.savefig('AMZNPDF.jpg')

#Get Data
e=output['FB (|Error|)']

#Plot Data
```

```
sns.set(style='whitegrid', rc={"grid.linewidth": ...
    0.25,'figure.figsize':(16,9)})
sns.set_context("paper", font_scale=2)
sns.distplot(e,fit=stats.beta,norm_hist=True,
             hist_kws={"histtype": "bar", "linewidth": 3,"alpha": ...
                 0.15, "color": "b","label":
             "Sample Histogram"},
             fit_kws= {"color": "r", "lw": 3, "label": "Beta ...
                 Probability Density Function (PDF)"},
             kde_kws={"color": "k", "lw": 3, "label": "Kernel ...
                 Density Estimation (KDE)"})
plt.title('KDE with Sample Histogram Vs. Beta PDF (FB)', fontsize=24)
plt.ylabel('Density', fontsize=20)
plt.xlabel('|Error|', fontsize=20)
plt.rcParams['font.family'] = 'serif'
plt.legend(fontsize=20)
plt.ylim(0, 55)

#Save Plot
plt.savefig('FBPDF.eps')
plt.savefig('FBPDF.jpg')

#Get Data
e=output['GOOGL (| Error|)']

#Plot Data
sns.set(style='whitegrid', rc={"grid.linewidth": ...
    0.25,'figure.figsize':(16,9)})
sns.set_context("paper", font_scale=2)
sns.distplot(e,fit=stats.beta,norm_hist=True,
             hist_kws={"histtype": "bar", "linewidth": 3,"alpha": ...
                 0.15, "color": "b","label":
```

```
                "Sample Histogram"},
                fit_kws= {"color": "r", "lw": 3, "label": "Beta ...
                    Probability Density Function (PDF)"},
                kde_kws={"color": "k", "lw": 3, "label": "Kernel ...
                    Density Estimation (KDE)"})
plt.title('KDE with Sample Histogram Vs. Beta PDF (GOOGL)', fontsize=24)
plt.ylabel('Density', fontsize=20)
plt.xlabel('|Error|', fontsize=20)
plt.rcParams['font.family'] = 'serif'
plt.legend(fontsize=20)
plt.ylim(0, 70)

#Save Plot
plt.savefig('GOOGLPDF.eps')
plt.savefig('GOOGLPDF.jpg')

#Get Data
e=output['MSFT (|Error|)']

#Plot Data
sns.set(style='whitegrid', rc={"grid.linewidth": ...
    0.25,'figure.figsize':(16,9)})
sns.set_context("paper", font_scale=2)
sns.distplot(e,fit=stats.beta,norm_hist=True,
                hist_kws={"histtype": "bar", "linewidth": 3,"alpha": ...
                    0.15, "color": "b","label":
                "Sample Histogram"},
                fit_kws= {"color": "r", "lw": 3, "label": "Beta ...
                    Probability Density Function (PDF)"},
                kde_kws={"color": "k", "lw": 3, "label": "Kernel ...
                    Density Estimation (KDE)"})
plt.title('KDE with Sample Histogram Vs. Beta PDF (MSFT)', fontsize=24)
```

```
plt.ylabel('Density', fontsize=20)
plt.xlabel('|Error|', fontsize=20)
plt.rcParams['font.family'] = 'serif'
plt.legend(fontsize=20)
plt.ylim(0, 65)

#Save Plot
plt.savefig('MSFTPDF.eps')
plt.savefig('MSFTPDF.jpg')

#Get Data
e=output['NFLX (|Error|)']

#Plot Data
sns.set(style='whitegrid', rc={"grid.linewidth": ...
    0.25,'figure.figsize':(16,9)})
sns.set_context("paper", font_scale=2)
sns.distplot(e,fit=stats.beta,norm_hist=True,
             hist_kws={"histtype": "bar", "linewidth": 3,"alpha": ...
                 0.15, "color": "b","label":
             "Sample Histogram"},
             fit_kws= {"color": "r", "lw": 3, "label": "Beta ...
                 Probability Density Function (PDF)"},
             kde_kws={"color": "k", "lw": 3, "label": "Kernel ...
                 Density Estimation (KDE)"})
plt.title('KDE with Sample Histogram Vs. Beta PDF (NFLX)', fontsize=24)
plt.ylabel('Density', fontsize=20)
plt.xlabel('|Error|', fontsize=20)
plt.rcParams['font.family'] = 'serif'
plt.legend(fontsize=20)
plt.ylim(0, 40)
```

```
#Save Plot
plt.savefig('NFLXPDF.eps')
plt.savefig('NFLXPDF.jpg')

#Chi—Square Test
stats.chisquare(e)


#ChiSquare Test

stats.chisquare(e,f)

#Wilcoxon Signed Rank Test

x=output['AMZN (Change)']
y=output['FB (Change)']
stats.wilcoxon(x,y)

x=output['AMZN (Change)']
y=output['GOOGL (Change)']
stats.wilcoxon(x,y)

x=output['AMZN (Change)']
y=output['MSFT (Change)']
stats.wilcoxon(x,y)

x=output['AMZN (Change)']
y=output['NFLX (Change)']
stats.wilcoxon(x,y)

x=output['FB (Change)']
y=output['GOOGL (Change)']
```

```
stats.wilcoxon(x,y)

x=output['FB (Change)']
y=output['MSFT (Change)']
stats.wilcoxon(x,y)

x=output['FB (Change)']
y=output['NFLX (Change)']
stats.wilcoxon(x,y)

x=output['GOOGL (Change)']
y=output['MSFT (Change)']
stats.wilcoxon(x,y)

x=output['GOOGL (Change)']
y=output['NFLX (Change)']
stats.wilcoxon(x,y)

x=output['MSFT (Change)']
y=output['NFLX (Change)']
stats.wilcoxon(x,y)
```

APPENDIX B. RAW DATA

In this appendix, the raw data of the following stocks are given.

- Amazon Inc. (AMZN)
- FB Inc. (FB)
- Google Inc. (GOOGL)
- Microsoft Inc. (MSFT)
- Netflix Inc. (NFLX)

Date	AMZN (Actual)	AMZN (Predicted)	AMZN (|Error|)	AMZN (|% of Error|)	AMZN (Change)
1/3/17	753.6699829	763.130249	0.012552266	1.255226601	0
1/4/17	757.1799927	764.4590454	0.009613371	0.961337145	0
1/5/17	780.4500122	751.3500977	0.037286069	3.728606924	1
1/6/17	795.9899902	784.3565674	0.014615037	1.461503655	1
1/9/17	796.9199829	802.9025879	0.007507159	0.750715891	0
1/10/17	795.9000244	818.534729	0.028439131	2.843913063	0
1/11/17	799.0200195	801.7346191	0.003397411	0.33974112	0
1/12/17	813.6400146	788.4696045	0.030935561	3.093556128	1
1/13/17	817.1400146	816.5686035	0.000699282	0.069928181	1
1/17/17	809.7199707	816.5690918	0.008458629	0.845862925	0
1/18/17	807.4799805	831.2981567	0.029496925	2.949692495	0
1/19/17	809.039978	798.0313721	0.013606999	1.360699907	1
1/20/17	808.3300171	814.7600098	0.007954663	0.795466267	0
1/23/17	817.8800049	817.8069458	8.93E-05	0.008932738	1
1/24/17	822.4400024	823.7751465	0.001623394	0.162339385	0
1/25/17	836.5200195	813.0419312	0.02806638	2.806637995	1
1/26/17	839.1500244	839.1105347	4.71E-05	0.004705922	1
1/27/17	835.7700195	810.7426758	0.029945252	2.994525246	1
1/30/17	830.3800049	811.8262329	0.022343712	2.234371193	1
1/31/17	823.4799805	818.0549927	0.006587881	0.658788066	1
2/1/17	832.3499756	842.6679688	0.01239622	1.239622012	0
2/2/17	839.9500122	845.2011108	0.00625168	0.625167973	0
2/3/17	810.2000122	846.894104	0.045290165	4.529016465	0
2/6/17	807.6400146	812.270874	0.005733816	0.573381595	0
2/7/17	812.5	812.4961548	4.73E-06	0.000473257	1
2/8/17	819.710022	792.4151001	0.033298265	3.329826519	1
2/9/17	821.3599854	829.4816895	0.009888117	0.98881172	0
2/10/17	827.460022	832.3088379	0.005859879	0.585987931	0
2/13/17	836.5300293	818.8579712	0.021125432	2.11254321	1
2/14/17	836.3900146	845.7561646	0.011198305	1.119830459	0
2/15/17	842.7000122	840.7086792	0.002363039	0.236303895	1
2/16/17	844.1400146	844.2389526	0.000117206	0.011720566	0
2/17/17	845.0700073	830.9991455	0.016650528	1.665052772	1
2/21/17	856.4400024	815.4785767	0.047827549	4.782754928	1
2/22/17	855.6099854	860.4698486	0.005679999	0.567999855	0
2/23/17	852.1900024	847.303894	0.005733591	0.573359057	1
2/24/17	845.2399902	831.9319458	0.015744694	1.574469358	1
2/27/17	848.6400146	840.9008179	0.009119528	0.911952835	1
2/28/17	845.039978	851.8552856	0.008065071	0.806507096	0
3/1/17	853.0800171	847.0872803	0.007024824	0.702482369	1
3/2/17	848.9099731	855.1206665	0.00731608	0.731607992	0
3/3/17	849.8800049	831.5255127	0.02159657	2.159656957	1

Date	AMZN (Actual)	AMZN (Predicted)	AMZN (|Error|)	AMZN (|% of Error|)	AMZN (Change)
3/6/17	846.6099854	863.4970703	0.019946711	1.994671114	0
3/7/17	846.0200195	848.8410645	0.00333449	0.333448965	0
3/8/17	850.5	876.5911865	0.030677468	3.067746758	0
3/9/17	853	853.9431152	0.001105645	0.110564509	0
3/10/17	852.460022	851.3884277	0.001257061	0.125706103	1
3/13/17	854.5900269	845.8917847	0.010178263	1.01782633	1
3/14/17	852.5300293	850.7039795	0.002141919	0.214191852	1
3/15/17	852.9699707	838.0909424	0.017443789	1.744378917	1
3/16/17	853.4199829	880.8852539	0.032182597	3.218259662	0
3/17/17	852.3099976	840.3973389	0.013976908	1.397690829	1
3/20/17	856.9699707	834.6129761	0.026088422	2.608842216	1
3/21/17	843.2000122	824.9550781	0.02163773	2.16377303	1
3/22/17	848.0599976	863.3927002	0.018079737	1.80797372	0
3/23/17	847.3800049	843.7093506	0.004331769	0.433176896	1
3/24/17	845.6099854	874.3197632	0.033951558	3.395155817	0
3/27/17	846.8200073	850.5444946	0.004398204	0.439820439	0
3/28/17	856	842.7017212	0.015535372	1.553537231	1
3/29/17	874.3200073	874.1565552	0.000186948	0.018694774	1
3/30/17	876.3400269	880.4769897	0.004720728	0.472072791	0
3/31/17	886.539978	851.9805908	0.038982321	3.898232058	1
4/3/17	891.5100098	883.8270874	0.008617875	0.861787517	1
4/4/17	906.8300171	894.5923462	0.013495	1.349500008	1
4/5/17	909.2800293	913.8010254	0.004972061	0.497206114	0
4/6/17	898.2800293	896.8809814	0.001557474	0.155747402	1
4/7/17	894.8800049	877.430603	0.019499153	1.94991529	1
4/10/17	907.039978	904.3638916	0.002950351	0.295035122	1
4/11/17	902.3599854	927.8946533	0.028297652	2.829765156	0
4/12/17	896.2299805	897.4694824	0.001383018	0.138301775	0
4/13/17	884.6699829	928.9537964	0.050056875	5.005687475	0
4/17/17	901.9899902	878.7979126	0.025712123	2.571212314	1
4/18/17	903.7800293	907.9470215	0.004610626	0.461062649	0
4/19/17	899.2000122	903.168457	0.004413306	0.441330625	0
4/20/17	902.0599976	923.7561646	0.0240518	2.405180037	0
4/21/17	898.5300293	887.0002441	0.01283183	1.283183042	1
4/24/17	907.4099731	928.6276855	0.02338272	2.338271961	0
4/25/17	907.6199951	889.9719849	0.019444272	1.944427192	1
4/26/17	909.289978	916.4967041	0.007925663	0.792566314	0
4/27/17	918.3800049	925.4155273	0.007660797	0.766079687	0
4/28/17	924.9899902	920.807312	0.004521863	0.452186307	1
5/1/17	948.2299805	929.696228	0.01954563	1.954562962	1
5/2/17	946.9400024	969.9589233	0.024308743	2.430874296	0
5/3/17	941.0300293	1010.171021	0.073473737	7.347373664	0

Date	AMZN (Actual)	AMZN (Predicted)	AMZN (|Error|)	AMZN (|% of Error|)	AMZN (Change)
5/4/17	937.5300293	949.5662842	0.01283826	1.283826027	0
5/5/17	934.1500244	960.8296509	0.028560324	2.856032364	0
5/8/17	949.039978	906.0653687	0.045282193	4.528219253	1
5/9/17	952.8200073	936.8204956	0.016791746	1.679174602	1
5/10/17	948.9500122	945.0997314	0.004057412	0.405741157	1
5/11/17	947.6199951	950.973999	0.003539398	0.353939761	0
5/12/17	961.3499756	950.5669556	0.01121654	1.121653989	1
5/15/17	957.9699707	961.1419067	0.003311102	0.331110181	0
5/16/17	966.0700073	973.6856079	0.007883073	0.788307283	0
5/17/17	944.7600098	953.072998	0.008799047	0.879904721	0
5/18/17	958.4899902	967.6523438	0.009559155	0.955915451	0
5/19/17	959.8400269	969.2063599	0.009758223	0.975822285	0
5/22/17	970.6699829	968.6930542	0.002036664	0.203666417	1
5/23/17	971.539978	956.7996216	0.015172157	1.517215651	1
5/24/17	980.3499756	978.7015991	0.001681416	0.16814163	1
5/25/17	993.3800049	977.9364624	0.01554646	1.55464597	1
5/26/17	995.7800293	970.4677734	0.025419526	2.54195258	1
5/30/17	996.7000122	1008.200684	0.01153875	1.153874956	0
5/31/17	994.6199951	1000.96521	0.006379537	0.637953682	0
6/1/17	995.9500122	1026.75769	0.030932955	3.093295544	0
6/2/17	1006.72998	1002.545349	0.004156657	0.415665703	1
6/5/17	1011.340027	1025.108643	0.01361423	1.361422986	0
6/6/17	1003	1022.270447	0.019212808	1.921280846	0
6/7/17	1010.070007	1017.655701	0.007510067	0.751006696	0
6/8/17	1010.27002	1007.407288	0.00283363	0.283363042	1
6/9/17	978.3099976	1012.691711	0.035143986	3.514398634	0
6/12/17	964.9099731	978.3222046	0.013899982	1.389998198	0
6/13/17	980.789978	968.3820801	0.012650923	1.265092287	1
6/14/17	976.4699707	959.8396606	0.017031051	1.703105122	1
6/15/17	964.1699829	963.3013306	0.000900933	0.090093276	1
6/16/17	987.710022	981.1728516	0.006618512	0.661851186	1
6/19/17	995.1699829	966.1190796	0.0291919	2.919190004	1
6/20/17	992.5900269	1003.281799	0.010771589	1.077158935	0
6/21/17	1002.22998	988.0116577	0.014186687	1.418668684	1
6/22/17	1001.299988	1005.65448	0.004348839	0.43488387	0
6/23/17	1003.73999	986.776123	0.016900659	1.690065861	1
6/26/17	993.9799805	984.7826538	0.00925303	0.925302971	1
6/27/17	976.7800293	1003.590881	0.027448198	2.744819783	0
6/28/17	990.3300171	976.8543091	0.013607291	1.360729057	1
6/29/17	975.9299927	972.9718018	0.003031151	0.303115067	1
6/30/17	968	947.1744385	0.02151401	2.151400968	1
7/3/17	953.6599731	996.3812256	0.044797152	4.479715228	0

Date	AMZN (Actual)	AMZN (Predicted)	AMZN (|Error|)	AMZN (|% of Error|)	AMZN (Change)
7/5/17	971.4000244	966.8140259	0.004721019	0.472101942	1
7/6/17	965.1400146	956.4597168	0.008993822	0.899382215	1
7/7/17	978.7600098	979.0670166	0.000313669	0.031366915	0
7/10/17	996.4699707	986.7476196	0.009756793	0.975679327	1
7/11/17	994.1300049	971.2205811	0.023044696	2.304469608	1
7/12/17	1006.51001	978.1056519	0.02822064	2.82206405	1
7/13/17	1000.630005	1020.716797	0.020074146	2.007414587	0
7/14/17	1001.809998	1011.258484	0.009431415	0.943141524	0
7/17/17	1010.039978	999.7738647	0.010164066	1.016406622	1
7/18/17	1024.449951	1017.820618	0.006471115	0.647111516	1
7/19/17	1026.869995	1016.424744	0.010171932	1.017193217	1
7/20/17	1028.699951	1025.211304	0.003391317	0.339131686	1
7/21/17	1025.670044	1011.405029	0.013907996	1.390799601	1
7/24/17	1038.949951	1021.690918	0.016611997	1.661199704	1
7/25/17	1039.869995	1062.852173	0.022101011	2.210101113	0
7/26/17	1052.800049	1032.089966	0.019671431	1.967143081	1
7/27/17	1046	1071.435059	0.024316499	2.431649901	0
7/28/17	1020.039978	1074.588501	0.053476848	5.347684771	0
7/31/17	987.7800293	1015.483704	0.028046401	2.804640122	0
8/1/17	996.1900024	1001.295593	0.005125117	0.512511749	0
8/2/17	995.8900146	1005.307312	0.009456162	0.945616234	0
8/3/17	986.9199829	1015.753723	0.029215885	2.921588533	0
8/4/17	987.5800171	975.2684937	0.012466355	1.246635523	1
8/7/17	992.2700195	994.2727051	0.002018287	0.201828685	0
8/8/17	989.8400269	1018.798279	0.029255487	2.925548702	0
8/9/17	982.0100098	991.4373779	0.009600073	0.96000731	0
8/10/17	956.9199829	971.9935913	0.015752215	1.575221494	0
8/11/17	967.9899902	955.9691772	0.012418323	1.24183232	1
8/14/17	983.2999878	980.6886597	0.002655678	0.265567796	1
8/15/17	982.7399902	980.9034424	0.001868803	0.18688034	1
8/16/17	978.1799927	995.2000122	0.01739968	1.739967987	0
8/17/17	960.5700073	977.9898071	0.018134857	1.81348566	0
8/18/17	958.4699707	959.9901733	0.001586072	0.158607226	0
8/21/17	953.289978	973.7581787	0.021471117	2.147111669	0
8/22/17	966.9000244	962.4393311	0.004613397	0.461339671	1
8/23/17	958	946.3741455	0.012135548	1.213554759	1
8/24/17	952.4500122	967.3337402	0.015626781	1.562678069	0
8/25/17	945.2600098	941.4326172	0.004049037	0.404903665	1
8/28/17	946.0200195	969.6469727	0.02497511	2.497510985	0
8/29/17	954.0599976	920.8912964	0.034765843	3.47658433	1
8/30/17	967.5900269	954.3999634	0.013631872	1.36318719	1
8/31/17	980.5999756	952.824585	0.028324895	2.832489461	1

Date	AMZN (Actual)	AMZN (Predicted)	AMZN (|Error|)	AMZN (|% of Error|)	AMZN (Change)
9/1/17	978.25	988.8575439	0.010843388	1.08433878	0
9/5/17	965.2700195	974.9854126	0.010064948	1.006494835	0
9/6/17	967.7999878	964.374939	0.003539005	0.353900483	1
9/7/17	979.4699707	1005.500793	0.026576437	2.657643706	0
9/8/17	965.9000244	1006.817749	0.042362276	4.236227646	0
9/11/17	977.960022	965.5977783	0.012640848	1.264084782	1
9/12/17	982.5800171	979.4910889	0.003143691	0.314369122	1
9/13/17	999.5999756	986.2191772	0.013386153	1.338615268	1
9/14/17	992.210022	979.3566284	0.012954308	1.295430772	1
9/15/17	986.789978	978.7200317	0.008177978	0.817797799	1
9/18/17	974.1900024	986.1650391	0.012292301	1.229230128	0
9/19/17	969.8599854	970.1912842	0.000341594	0.034159448	0
9/20/17	973.210022	958.024292	0.015603755	1.560375467	1
9/21/17	964.6500244	954.8725586	0.010135765	1.013576519	1
9/22/17	955.0999756	973.7597656	0.019537002	1.953700185	0
9/25/17	939.789978	952.0856323	0.013083407	1.308340672	0
9/26/17	938.5999756	941.7966309	0.00340577	0.340576959	0
9/27/17	950.8699951	926.798645	0.02531508	2.531507984	1
9/28/17	956.4000244	955.098877	0.001360464	0.136046368	1
9/29/17	961.3499756	961.3855591	3.70E-05	0.003701409	0
10/2/17	959.1900024	972.4088745	0.013781287	1.378128678	0
10/3/17	957.0999756	959.2669067	0.002264059	0.226405938	0
10/4/17	965.4500122	951.1013794	0.014862119	1.486211922	1
10/5/17	980.8499756	961.0279541	0.020209026	2.020902559	1
10/6/17	989.5800171	1009.697144	0.020328954	2.032895386	0
10/9/17	990.9899902	994.0164795	0.003054006	0.305400579	0
######	987.2000122	1020.826294	0.034062278	3.406227753	0
######	995	971.8828125	0.023233354	2.323335409	1
######	1000.929993	989.7002563	0.011219302	1.121930219	1
######	1002.940002	1015.033813	0.01205836	1.205835957	0
######	1006.340027	1008.976379	0.002619743	0.261974335	0
######	1009.130005	1015.144897	0.005960473	0.596047333	0
######	997	994.8118896	0.002194694	0.219469448	1
######	986.6099854	1013.988892	0.027750487	2.775048651	0
######	982.9099731	992.7736206	0.010035149	1.003514882	0
######	966.2999878	985.026062	0.019379152	1.937915199	0
######	975.9000244	971.5202026	0.004487982	0.448798202	1
######	972.9099731	955.1259155	0.018279243	1.827924326	1
######	972.4299927	956.4935303	0.016388288	1.638828777	1
######	1100.949951	997.493103	0.09397053	9.397052974	1
######	1110.849976	1118.120361	0.006544885	0.654488523	0
######	1105.280029	1128.427002	0.020942179	2.094217949	0

Date	AMZN (Actual)	AMZN (Predicted)	AMZN (|Error|)	AMZN (|% of Error|)	AMZN (Change)
11/1/17	1103.680054	1062.948975	0.03690479	3.690478951	1
11/2/17	1094.219971	1109.209473	0.013698801	1.36988014	0
11/3/17	1111.599976	1074.227905	0.033620071	3.362007067	1
11/6/17	1120.660034	1128.997925	0.00744016	0.744016049	0
11/7/17	1123.170044	1114.375732	0.007829902	0.782990176	1
11/8/17	1132.880005	1120.778564	0.010682015	1.068201475	1
11/9/17	1129.130005	1148.93103	0.017536532	1.753653213	0
######	1125.349976	1115.034668	0.009166311	0.916631054	1
######	1129.170044	1126.49292	0.002370878	0.237087766	1
######	1136.839966	1136.296509	0.000478042	0.047804179	1
######	1126.689941	1112.486816	0.012606064	1.260606386	1
######	1137.290039	1132.618042	0.004108008	0.410800846	1
######	1129.880005	1156.672241	0.023712462	2.371246181	0
######	1126.310059	1133.246826	0.006158844	0.615884364	0
######	1139.48999	1142.009399	0.002210997	0.221099728	0
######	1156.160034	1133.315918	0.01975861	1.975861005	1
######	1186	1126.092651	0.050512098	5.051209778	1
######	1195.829956	1196.288696	0.000383617	0.03836166	0
######	1193.599976	1165.268066	0.02373652	2.373651974	1
######	1161.27002	1171.648682	0.008937337	0.893733744	0
######	1176.75	1133.504639	0.036749829	3.674982861	1
12/1/17	1162.349976	1166.02771	0.003164051	0.316405087	0
12/4/17	1133.949951	1168.765503	0.0307029	3.070290014	0
12/5/17	1141.569946	1115.21875	0.023083296	2.308329567	1
12/6/17	1152.349976	1136.410156	0.013832446	1.383244619	1
12/7/17	1159.790039	1203.229248	0.037454374	3.745437413	0
12/8/17	1162	1162.417358	0.000359172	0.035917247	0
######	1168.920044	1147.228516	0.018556897	1.85568966	1
######	1165.079956	1177.35437	0.010535255	1.053525507	0
######	1164.130005	1172.499756	0.007189705	0.718970457	0
######	1174.26001	1140.490967	0.028757721	2.875772119	1
######	1179.140015	1171.12085	0.006800859	0.680085924	1
######	1190.579956	1159.592651	0.026027067	2.602706663	1
######	1187.380005	1127.701416	0.050260734	5.026073381	1
######	1177.619995	1203.066772	0.021608649	2.160864882	0
######	1174.76001	1201.257446	0.022555618	2.255561762	0
######	1168.359985	1195.736084	0.023431219	2.343121916	0
######	1176.76001	1170.834229	0.005035675	0.503567513	1
######	1182.26001	1160.951416	0.01802361	1.802361012	1
######	1186.099976	1180.181152	0.004990155	0.499015534	1
######	1169.469971	1210.708984	0.035262994	3.526299447	0

Date	FB (Actual)	FB (Predicted)	FB (|Error|)	FB (|% of Error|)	FB (Change)
1/3/17	116.860001	115.7409134	0.009576307	0.957630668	1
1/4/17	118.690002	115.8662796	0.023790739	2.379073948	1
1/5/17	120.669998	117.299263	0.027933499	2.793349884	1
1/6/17	123.410004	122.6077805	0.006500471	0.650047138	1
1/9/17	124.900002	126.9867249	0.016707152	1.670715213	0
1/10/17	124.349999	126.1893387	0.014791639	1.479163859	0
1/11/17	126.089996	126.1412964	0.000406853	0.040685266	0
1/12/17	126.620003	127.452919	0.006578078	0.657807803	0
1/13/17	128.339996	127.3918152	0.007388041	0.738804089	1
1/17/17	127.870003	128.2363739	0.002865185	0.286518456	0
1/18/17	127.919998	127.4643021	0.003562352	0.356235239	1
1/19/17	127.550003	126.4541779	0.008591338	0.859133806	1
1/20/17	127.040001	125.6832428	0.010679771	1.067977119	1
1/23/17	128.929993	128.2625732	0.005176603	0.517660333	1
1/24/17	129.369995	130.5218506	0.008903575	0.890357513	0
1/25/17	131.479996	132.3295593	0.006461543	0.646154257	0
1/26/17	132.779999	130.9036713	0.014131101	1.413110085	1
1/27/17	132.179993	131.2668304	0.006908475	0.690847542	1
1/30/17	130.979996	134.9216919	0.030093879	3.009387851	0
1/31/17	130.320007	131.3752136	0.00809704	0.809703954	0
2/1/17	133.229996	129.585083	0.02735805	2.735804953	1
2/2/17	130.839996	131.8072052	0.007392303	0.739230262	0
2/3/17	130.979996	132.8684387	0.014417797	1.441779733	0
2/6/17	132.059998	130.4863892	0.01191586	1.191585977	1
2/7/17	131.839996	131.1713257	0.005071835	0.507183466	1
2/8/17	134.199997	127.7651978	0.047949325	4.794932529	1
2/9/17	134.139999	133.8646393	0.002052781	0.20527814	1
2/10/17	134.190002	137.6530914	0.025807355	2.58073546	0
2/13/17	134.050003	133.3416443	0.005284288	0.52842875	1
2/14/17	133.850006	131.8578186	0.014883731	1.488373149	1
2/15/17	133.440002	134.0540771	0.004601879	0.460187858	0
2/16/17	133.839996	134.0325012	0.001438321	0.143832108	0
2/17/17	133.529999	132.2260284	0.009765374	0.976537354	1
2/21/17	133.720001	132.6526489	0.007981994	0.798199419	1
2/22/17	136.119995	135.6128845	0.003725467	0.372546725	1
2/23/17	135.360001	138.4421692	0.022770157	2.277015708	0
2/24/17	135.440002	134.9802246	0.003394698	0.339469756	1
2/27/17	136.410004	136.02034	0.002856562	0.285656238	1
2/28/17	135.539993	132.8054657	0.02017506	2.017506026	1
3/1/17	137.419998	137.9808655	0.00408141	0.408140989	0
3/2/17	136.759995	132.2714691	0.032820456	3.282045573	1
3/3/17	137.169998	129.690094	0.054530177	5.453017727	1

Date	FB (Actual)	FB (Predicted)	FB (|Error|)	FB (|% of Error|)	FB (Change)
3/6/17	137.419998	136.2168274	0.008755427	0.875542685	1
3/7/17	137.300003	137.8221436	0.003802917	0.380291697	0
3/8/17	137.720001	139.4697113	0.012704836	1.27048362	0
3/9/17	138.240006	136.9315491	0.009465107	0.946510676	1
3/10/17	138.789993	138.2866364	0.003626752	0.362675241	1
3/13/17	139.600006	136.2445221	0.024036417	2.403641678	1
3/14/17	139.320007	140.2729034	0.006839621	0.683962135	0
3/15/17	139.720001	138.8584137	0.006166529	0.616652938	1
3/16/17	139.990006	136.9924469	0.021412661	2.14126613	1
3/17/17	139.839996	139.7127533	0.000909919	0.09099188	1
3/20/17	139.940002	140.6958923	0.005401528	0.540152844	0
3/21/17	138.509995	141.7794952	0.023604801	2.360480092	0
3/22/17	139.589996	137.6601715	0.013824951	1.38249509	1
3/23/17	139.529999	139.3592987	0.001223393	0.122339337	1
3/24/17	140.339996	139.9116821	0.003051975	0.305197528	1
3/27/17	140.320007	143.1473999	0.020149603	2.014960349	0
3/28/17	141.759995	140.1096497	0.011641824	1.164182369	1
3/29/17	142.649994	142.4546509	0.001369387	0.136938679	1
3/30/17	142.410004	144.8581543	0.017190861	1.719086058	0
3/31/17	142.050003	139.1359406	0.020514343	2.051434293	1
4/3/17	142.279999	145.5473175	0.022964006	2.296400629	0
4/4/17	141.729996	141.648468	0.000575233	0.057523255	1
4/5/17	141.850006	140.6783447	0.008259862	0.825986173	1
4/6/17	141.169998	138.4394379	0.019342355	1.934235543	1
4/7/17	140.779999	139.5671539	0.008615179	0.861517899	1
4/10/17	141.039993	139.1761475	0.013215017	1.321501657	1
4/11/17	139.919998	144.3532867	0.031684455	3.168445453	0
4/12/17	139.580002	140.4321136	0.006104827	0.610482739	0
4/13/17	139.389999	138.417572	0.006976306	0.697630644	1
4/17/17	141.419998	138.8720093	0.018017175	1.801717468	1
4/18/17	140.960007	141.843689	0.006269028	0.626902794	0
4/19/17	142.270004	137.569046	0.033042513	3.304251283	1
4/20/17	143.800003	144.9498444	0.007996115	0.79961149	0
4/21/17	143.679993	143.1814575	0.003469761	0.346976053	1
4/24/17	145.470001	143.2319031	0.01538529	1.538528968	1
4/25/17	146.490006	146.6445007	0.001054647	0.105464691	0
4/26/17	146.559998	146.0779114	0.003289344	0.328934356	1
4/27/17	147.699997	142.3853302	0.035982851	3.598285094	1
4/28/17	150.25	145.6749573	0.030449536	3.04495357	1
5/1/17	152.460007	149.0169525	0.022583326	2.258332632	1
5/2/17	152.779999	156.0624084	0.02148455	2.148455009	0
5/3/17	151.800003	152.9709625	0.007713831	0.771383056	0

Date	FB (Actual)	FB (Predicted)	FB (|Error|)	FB (|% of Error|)	FB (Change)
5/4/17	150.850006	150.5691833	0.001861603	0.186160253	1
5/5/17	150.240006	151.53685	0.008631819	0.863181893	0
5/8/17	151.059998	152.4618835	0.009280326	0.928032584	0
5/9/17	150.479996	153.7003021	0.02140023	2.140023001	0
5/10/17	150.289993	151.2331543	0.006275608	0.627560774	0
5/11/17	150.039993	150.418808	0.002524758	0.252475822	0
5/12/17	150.330002	149.466568	0.00574359	0.574358972	1
5/15/17	150.190002	153.0264282	0.018885583	1.888558269	0
5/16/17	149.779999	146.3975983	0.022582458	2.258245833	1
5/17/17	144.850006	150.161026	0.036665652	3.666565195	0
5/18/17	147.660004	147.9636688	0.002056516	0.205651601	0
5/19/17	148.059998	146.5673676	0.010081251	1.008125115	1
5/22/17	148.240006	148.0422668	0.001333909	0.133390876	1
5/23/17	148.070007	148.7198029	0.004388434	0.438843435	0
5/24/17	150.039993	150.1956482	0.001037423	0.10374228	0
5/25/17	151.960007	151.4501801	0.003355006	0.335500552	1
5/26/17	152.130005	153.6208801	0.009800008	0.980000757	0
5/30/17	152.380005	155.6179504	0.021249149	2.124914899	0
5/31/17	151.460007	154.4683533	0.019862317	1.986231655	0
6/1/17	151.529999	158.1447906	0.04365335	4.365335032	0
6/2/17	153.610001	155.6431122	0.013235542	1.323554199	0
6/5/17	153.630005	157.8868103	0.027708163	2.770816348	0
6/6/17	152.809998	153.5652161	0.004942206	0.49422062	0
6/7/17	153.119995	152.8159332	0.001985775	0.198577531	1
6/8/17	154.710007	155.0412292	0.002140925	0.214092503	0
6/9/17	149.600006	155.9375	0.042362925	4.236292467	0
6/12/17	148.440002	145.4183044	0.020356359	2.035635896	1
6/13/17	150.679993	151.4486389	0.005101183	0.510118296	0
6/14/17	150.25	157.0606079	0.045328505	4.532850534	0
6/15/17	149.800003	154.3975372	0.030691149	3.069114871	0
6/16/17	150.639999	151.5767517	0.006218483	0.621848321	0
6/19/17	152.869995	148.5122681	0.028506098	2.850609832	1
6/20/17	152.25	155.8279724	0.02350064	2.350063995	0
6/21/17	153.910004	154.2229004	0.002032985	0.203298498	0
6/22/17	153.399994	153.3282623	0.000467611	0.046761127	1
6/23/17	155.070007	152.2314453	0.018305035	1.830503531	1
6/26/17	153.589996	157.5576935	0.025833044	2.58330442	0
6/27/17	150.580002	154.9186554	0.028812947	2.881294675	0
6/28/17	153.240006	152.0103302	0.008024505	0.802450534	1
6/29/17	151.039993	156.479248	0.036012016	3.601201624	0
6/30/17	150.979996	153.311554	0.015442829	1.544282865	0
7/3/17	148.429993	153.4825897	0.034040269	3.404026851	0

Date	FB (Actual)	FB (Predicted)	FB (|Error|)	FB (|% of Error|)	FB (Change)
7/5/17	150.339996	145.6172485	0.031413782	3.141378239	1
7/6/17	148.820007	151.6641388	0.019111218	1.911121793	0
7/7/17	151.440002	152.656311	0.008031621	0.803162064	0
7/10/17	153.5	152.2414551	0.00819899	0.819898956	1
7/11/17	155.270004	153.3536987	0.012341763	1.234176289	1
7/12/17	158.899994	155.502655	0.021380359	2.138035931	1
7/13/17	159.259995	155.8804626	0.021220218	2.122021839	1
7/14/17	159.970001	163.7025299	0.02333268	2.333267964	0
7/17/17	159.729996	160.2161865	0.003043829	0.3043829	0
7/18/17	162.860001	159.155899	0.022744084	2.274408378	1
7/19/17	164.139999	161.411087	0.016625516	1.662551612	1
7/20/17	164.529999	166.1747894	0.009996904	0.999690406	0
7/21/17	164.429993	164.5129089	0.000504265	0.050426484	0
7/24/17	166	166.7672729	0.004622126	0.4622126	0
7/25/17	165.279999	168.8646698	0.021688474	2.168847434	0
7/26/17	165.610001	163.9824371	0.009827688	0.982768834	1
7/27/17	170.440002	165.4342804	0.029369408	2.936940826	1
7/28/17	172.449997	165.812088	0.038491789	3.849178925	1
7/31/17	169.25	179.6119385	0.06122268	6.122267991	0
8/1/17	169.860001	171.5118256	0.009724625	0.972462539	0
8/2/17	169.300003	174.1942139	0.02890851	2.890850976	0
8/3/17	168.589996	170.4302673	0.010915659	1.091565937	0
8/4/17	169.619995	170.8560638	0.007287282	0.728728203	0
8/7/17	171.979996	172.5254517	0.003171624	0.317162438	0
8/8/17	171.229996	171.9362335	0.004124498	0.412449799	0
8/9/17	171.179993	173.0683899	0.011031647	1.103164721	0
8/10/17	167.399994	176.220871	0.052693412	5.269341171	0
8/11/17	168.080002	171.5129852	0.020424699	2.042469941	0
8/14/17	170.75	169.2718506	0.008656804	0.865680445	1
8/15/17	171	175.6423645	0.027148331	2.714833058	0
8/16/17	170	174.2735748	0.025138676	2.513867617	0
8/17/17	166.910004	171.464859	0.02728929	2.728928998	0
8/18/17	167.410004	170.6984711	0.019643195	1.964319497	0
8/21/17	167.779999	167.2400818	0.003218006	0.321800564	1
8/22/17	169.639999	165.4485474	0.024707923	2.470792271	1
8/23/17	168.710007	173.1295624	0.026196169	2.619616874	0
8/24/17	167.740006	169.4833221	0.010392969	1.039296854	0
8/25/17	166.320007	170.1706085	0.023151761	2.315176092	0
8/28/17	167.240006	171.7588654	0.027020209	2.702020854	0
8/29/17	168.050003	168.5271149	0.002839106	0.283910637	0
8/30/17	169.919998	168.960495	0.005646794	0.564679364	1
8/31/17	171.970001	170.0516357	0.011155234	1.115523372	1

Date	FB (Actual)	FB (Predicted)	FB (|Error|)	FB (|% of Error|)	FB (Change)
9/1/17	172.020004	173.9538269	0.011241848	1.124184765	0
9/5/17	170.720001	174.3634644	0.021341747	2.134174667	0
9/6/17	172.089996	169.1266022	0.017220026	1.722002588	1
9/7/17	173.210007	176.3553162	0.018158937	1.815893687	0
9/8/17	170.949997	173.5134888	0.014995566	1.499556564	0
9/11/17	173.509995	166.5232544	0.040267076	4.026707634	1
9/12/17	172.960007	175.4431763	0.0143569	1.435690001	0
9/13/17	173.050003	171.4924622	0.009000525	0.900052488	1
9/14/17	170.960007	175.9155884	0.028986789	2.898678929	0
9/15/17	171.639999	166.8557281	0.027873872	2.787387185	1
9/18/17	170.009995	167.708725	0.013536084	1.353608351	1
9/19/17	172.520004	170.3464813	0.012598672	1.259867195	1
9/20/17	172.169998	174.894043	0.015821831	1.58218313	0
9/21/17	171.110001	173.416626	0.013480365	1.348036528	0
9/22/17	170.539993	171.4122314	0.005114567	0.511456653	0
9/25/17	162.869995	169.1841888	0.038768306	3.876830637	0
9/26/17	164.210007	164.2565613	0.000283506	0.028350626	0
9/27/17	167.679993	169.9528046	0.01355446	1.355446037	0
9/28/17	168.729996	167.4412079	0.007638167	0.763816666	1
9/29/17	170.869995	173.3357544	0.014430616	1.443061605	0
10/2/17	169.470001	169.2359467	0.001381097	0.138109736	1
10/3/17	169.960007	172.4425659	0.014606725	1.460672542	0
10/4/17	168.419998	174.3266296	0.035070844	3.507084399	0
10/5/17	171.240006	166.118454	0.029908616	2.99086161	1
10/6/17	172.229996	171.9496002	0.00162803	0.162802951	1
10/9/17	172.5	174.792511	0.013289919	1.328991912	0
10/10/17	171.589996	171.970932	0.002220034	0.222003437	0
10/11/17	172.740006	171.1615295	0.009137872	0.913787168	1
10/12/17	172.550003	174.1898956	0.009503868	0.950386841	0
10/13/17	173.740006	171.3681335	0.013651847	1.365184691	1
10/16/17	174.520004	173.3773499	0.006547412	0.654741237	1
10/17/17	176.110001	174.2709045	0.010442883	1.044288278	1
10/18/17	176.029999	177.4391632	0.008005251	0.800525118	0
10/19/17	174.559998	179.5988464	0.028866	2.886600047	0
10/20/17	174.979996	176.7285767	0.009993033	0.999303348	0
10/23/17	171.270004	177.844574	0.038387164	3.838716447	0
10/24/17	171.800003	171.8788605	0.000459007	0.04590071	0
10/25/17	170.600006	172.2867737	0.009887266	0.988726597	0
10/26/17	170.630005	172.4032745	0.010392484	1.039248426	0
10/27/17	177.880005	173.3524017	0.025453132	2.545313165	1
10/30/17	179.869995	177.6148987	0.012537369	1.253736857	1
10/31/17	180.059998	181.3451385	0.007137293	0.7137293	0

Date	FB (Actual)	FB (Predicted)	FB (|Error|)	FB (|% of Error|)	FB (Change)
11/1/17	182.660004	181.2472839	0.007734149	0.773414923	1
11/2/17	178.919998	182.1478729	0.018040882	1.804088242	0
11/3/17	178.919998	178.2452698	0.003771118	0.377111789	1
11/6/17	180.169998	180.9204865	0.004165445	0.416544545	0
11/7/17	180.25	180.0429688	0.001148578	0.114857836	1
11/8/17	179.559998	183.3317719	0.021005649	2.100564912	0
11/9/17	179.300003	184.0348511	0.026407406	2.640740573	0
11/10/17	178.460007	179.230896	0.004319675	0.431967527	0
11/13/17	178.770004	177.6967773	0.006003395	0.600339472	1
11/14/17	178.070007	173.0454102	0.028216977	2.821697667	1
11/15/17	177.949997	177.6199036	0.001854978	0.185497827	1
11/16/17	179.589996	180.4491425	0.004783931	0.478393072	0
11/17/17	179	181.0754547	0.011594719	1.159471925	0
11/20/17	178.740006	181.2171021	0.013858657	1.385865733	0
11/21/17	181.860001	180.4322052	0.007851069	0.785106886	1
11/22/17	180.869995	180.4962616	0.00206631	0.206631026	1
11/24/17	182.779999	181.1901855	0.008697961	0.869796053	1
11/27/17	183.029999	182.4376831	0.003236167	0.323616713	1
11/28/17	182.419998	183.7820892	0.007466786	0.746678608	0
11/29/17	175.130005	181.5463867	0.03663782	3.66378203	0
11/30/17	177.179993	173.4530792	0.021034619	2.103461884	1
12/1/17	175.100006	180.4819489	0.030736394	3.073639423	0
12/4/17	171.470001	184.0207825	0.073195204	7.319520414	0
12/5/17	172.830002	168.4253998	0.025485171	2.548517101	1
12/6/17	176.059998	169.0578003	0.039771654	3.977165371	1
12/7/17	180.139999	173.3974304	0.037429605	3.742960468	1
12/8/17	179	180.3405151	0.007488912	0.748891151	0
12/11/17	179.039993	185.155304	0.034156114	3.415611386	0
12/12/17	176.960007	178.9202576	0.011077367	1.107736677	0
12/13/17	178.300003	179.8986511	0.008966058	0.896605756	0
12/14/17	178.389999	174.6559601	0.020931887	2.093188651	1
12/15/17	180.179993	177.4613342	0.01508857	1.50885703	1
12/18/17	180.820007	183.4713593	0.014662934	1.466293447	0
12/19/17	179.509995	178.1056213	0.00782337	0.78233704	1
12/20/17	177.889999	178.1036224	0.001200872	0.120087154	0
12/21/17	177.449997	181.5638733	0.023183299	2.318329923	0
12/22/17	177.199997	179.5557556	0.013294349	1.329434942	0
12/26/17	175.990006	174.192627	0.010212958	1.021295786	1
12/27/17	177.619995	177.4072266	0.001197886	0.11978863	1
12/28/17	177.919998	175.7737122	0.012063208	1.206320804	1
12/29/17	176.460007	167.0413208	0.053375755	5.337575451	1

Date	GOOGL(Actual)	GOOGL(Predicted)	GOOGL(|Error|)	GOOGL(|% of Error|)	Change
1/3/17	808.0100098	796.7332764	0.013956181	1.395618077	1
1/4/17	807.7700195	804.2783813	0.004322565	0.43225647	1
1/5/17	813.0200195	814.4763184	0.001791221	0.179122132	0
1/6/17	825.210022	813.5779419	0.014095902	1.409590244	1
1/9/17	827.1799927	820.8075562	0.007703809	0.77038086	1
1/10/17	826.0100098	876.3110352	0.060896385	6.089638546	0
1/11/17	829.8599854	819.588562	0.012377297	1.237729657	1
1/12/17	829.5300293	828.3014526	0.001481051	0.148105144	1
1/13/17	830.9400024	807.4795532	0.028233627	2.823362686	1
1/17/17	827.460022	828.6383057	0.001423977	0.142397662	0
1/18/17	829.0200195	833.3598022	0.005234835	0.523483474	0
1/19/17	824.3699951	827.6546021	0.003984384	0.398438424	0
1/20/17	828.1699829	830.5946655	0.00292776	0.292775966	0
1/23/17	844.4299927	819.3115845	0.029745992	2.974599227	1
1/24/17	849.5300293	852.3380127	0.003305337	0.330533739	0
1/25/17	858.4500122	854.9851074	0.004036234	0.403623376	1
1/26/17	856.9799805	847.7573853	0.01076174	1.076173969	1
1/27/17	845.0300293	858.6442871	0.016110975	1.61109753	0
1/30/17	823.8300171	841.8127441	0.021828201	2.182820067	0
1/31/17	820.1900024	830.0163574	0.011980584	1.198058389	0
2/1/17	815.2399902	837.519165	0.027328363	2.732836269	0
2/2/17	818.2600098	824.0587769	0.007086705	0.708670495	0
2/3/17	820.1300049	823.0996704	0.003620969	0.362096936	0
2/6/17	821.6199951	830.2290039	0.01047809	1.047809049	0
2/7/17	829.2299805	823.203186	0.007267941	0.726794079	1
2/8/17	829.8800049	832.4108276	0.003049625	0.304962485	0
2/9/17	830.0599976	820.7436523	0.011223701	1.122370083	1
2/10/17	834.8499756	836.3353882	0.001779257	0.177925697	0
2/13/17	838.960022	838.6104126	0.000416718	0.041671755	1
2/14/17	840.0300293	846.2286987	0.007379104	0.737910438	0
2/15/17	837.3200073	842.6845703	0.006406826	0.64068255	0
2/16/17	842.1699829	830.871582	0.01341582	1.341581997	1
2/17/17	846.5499878	837.1973267	0.011047972	1.104797237	1
2/21/17	849.2700195	849.6443481	0.000440765	0.044076514	0
2/22/17	851.3599854	858.8212891	0.008763982	0.876398198	0
2/23/17	851	843.4725952	0.008845364	0.884536374	1
2/24/17	847.8099976	839.9894409	0.009224421	0.922442134	1
2/27/17	849.6699829	843.7781982	0.006934204	0.693420367	1
2/28/17	844.9299927	856.2427368	0.013388972	1.33889718	0
3/1/17	856.75	840.2311401	0.01928084	1.928083971	1
3/2/17	849.8499756	862.3322144	0.014687579	1.468757913	0
3/3/17	849.0800171	857.0435791	0.009379048	0.93790479	0

Date	GOOGL(Actual)	GOOGL(Predicted)	GOOGL(|Error|)	GOOGL(|% of Error|)	Change
3/6/17	847.2700195	849.887146	0.003088893	0.308889314	0
3/7/17	851.1500244	848.7125854	0.002863701	0.286370073	1
3/8/17	853.6400146	882.3069458	0.033581991	3.358199075	0
3/9/17	857.8400269	838.5354614	0.022503689	2.250368893	1
3/10/17	861.4099731	863.6542358	0.002605336	0.260533625	0
3/13/17	864.5800171	875.0154419	0.012069935	1.206993498	0
3/14/17	865.9099731	854.8306885	0.012794961	1.279496122	1
3/15/17	868.3900146	862.0369873	0.007315869	0.731586851	1
3/16/17	870	881.383667	0.013084674	1.308467425	0
3/17/17	872.3699951	865.0644531	0.008374362	0.837436225	1
3/20/17	867.9099731	873.7841797	0.006768221	0.676822104	0
3/21/17	850.1400146	881.6071777	0.037014093	3.701409325	0
3/22/17	849.7999878	869.9324341	0.023690805	2.369080484	0
3/23/17	839.6500244	851.7401123	0.014398961	1.439896133	0
3/24/17	835.1400146	839.2404785	0.004909912	0.490991166	0
3/27/17	838.5100098	845.895874	0.008808319	0.880831946	0
3/28/17	840.6300049	851.2601318	0.012645429	1.2645429	0
3/29/17	849.8699951	843.8880615	0.007038645	0.703864545	1
3/30/17	849.4799805	870.5228271	0.024771444	2.47714445	0
3/31/17	847.7999878	848.9904175	0.00140414	0.14041398	0
4/3/17	856.75	866.1425171	0.010962961	1.096296124	0
4/4/17	852.5700073	882.1934204	0.034746017	3.474601731	0
4/5/17	848.9099731	833.2479248	0.018449599	1.844959892	1
4/6/17	845.0999756	852.3867798	0.008622417	0.862241723	0
4/7/17	842.0999756	824.6082764	0.020771524	2.077152394	1
4/10/17	841.7000122	827.4481201	0.01693227	1.693226956	1
4/11/17	839.8800049	847.3632813	0.008909935	0.890993513	0
4/12/17	841.460022	838.5010376	0.003516488	0.351648848	1
4/13/17	840.1799927	840.9124146	0.000871744	0.087174401	0
4/17/17	855.1300049	834.7765503	0.023801591	2.380159125	1
4/18/17	853.9899902	863.9420776	0.011653635	1.165363472	0
4/19/17	856.5100098	846.7467041	0.011398939	1.139893942	1
4/20/17	860.0800171	876.4474487	0.019030126	1.903012581	0
4/21/17	858.9500122	860.1919556	0.001445886	0.144588551	0
4/24/17	878.9299927	861.944458	0.019325241	1.932524145	1
4/25/17	888.8400269	884.2749023	0.005136047	0.513604749	1
4/26/17	889.1400146	887.0826416	0.002313891	0.231389096	1
4/27/17	891.4400024	867.0234985	0.027389959	2.73899585	1
4/28/17	924.5200195	898.5906982	0.028046252	2.804625221	1
5/1/17	932.8200073	917.520752	0.01640108	1.640108041	1
5/2/17	937.0900269	925.4556885	0.01241539	1.241539046	1
5/3/17	948.4500122	950.8807983	0.002562904	0.25629038	0

Date	GOOGL(Actual)	GOOGL(Predicted)	GOOGL(|Error|)	GOOGL(|% of Error|)	Change
5/4/17	954.7199707	950.4754639	0.004445814	0.44458136	1
5/5/17	950.2800293	974.1028442	0.025069257	2.506925724	0
5/8/17	958.6900024	990.5875854	0.03327205	3.327205032	0
5/9/17	956.710022	932.9923706	0.024790846	2.479084581	1
5/10/17	954.8400269	992.3624878	0.039297119	3.929711878	0
5/11/17	955.8900146	968.5612793	0.013255986	1.325598639	0
5/12/17	955.1400146	946.8621216	0.00866668	0.866668019	1
5/15/17	959.2199707	954.9677734	0.004432974	0.443297392	1
5/16/17	964.6099854	956.1109619	0.00881084	0.881083962	1
5/17/17	942.1699829	973.5330811	0.033288151	3.328815103	0
5/18/17	950.5	944.8448486	0.00594966	0.594965974	1
5/19/17	954.6500244	956.5744019	0.002015794	0.20157937	0
5/22/17	964.0700073	962.3905029	0.001742098	0.174209801	1
5/23/17	970.5499878	956.6222534	0.014350353	1.435035281	1
5/24/17	977.6099854	982.3545532	0.004853232	0.485323183	0
5/25/17	991.8599854	977.3566895	0.014622321	1.462232135	1
5/26/17	993.2700195	1005.742493	0.012556981	1.25569813	0
5/30/17	996.1699829	994.8668823	0.001308111	0.130811066	1
5/31/17	987.0900269	1007.481323	0.02065799	2.065799013	0
6/1/17	988.289978	989.3932495	0.001116344	0.111634389	0
6/2/17	996.1199951	1007.220764	0.011144008	1.114400756	0
6/5/17	1003.880005	998.1275635	0.005730208	0.573020801	1
6/6/17	996.6799927	993.2376099	0.00345385	0.345384958	1
6/7/17	1001.5	999.7426147	0.001754753	0.175475318	1
6/8/17	1004.280029	1027.660278	0.023280608	2.328060754	0
6/9/17	970.1199951	1006.522217	0.037523422	3.752342239	0
6/12/17	961.8099976	977.9453735	0.016776053	1.677605323	0
6/13/17	970.5	980.4298096	0.010231643	1.023164298	0
6/14/17	967.9299927	980.8850098	0.01338425	1.338424999	0
6/15/17	960.1799927	966.4534302	0.006533606	0.653360551	0
6/16/17	958.6199951	945.7849731	0.013389061	1.338906121	1
6/19/17	975.2199707	959.9361572	0.01567217	1.567216963	1
6/20/17	968.9899902	964.1876221	0.004956055	0.495605543	1
6/21/17	978.5900269	987.5934448	0.009200398	0.920039788	0
6/22/17	976.6199951	975.0699463	0.001587157	0.158715656	1
6/23/17	986.0900269	976.7106934	0.00951164	0.95116403	1
6/26/17	972.0900269	995.2523804	0.023827374	2.382737398	0
6/27/17	948.0900269	991.2022705	0.045472734	4.547273368	0
6/28/17	961.0100098	943.0092163	0.018731119	1.873111911	1
6/29/17	937.8200073	979.1984253	0.044121917	4.412191734	0
6/30/17	929.6799927	942.0681763	0.013325213	1.332521252	0
7/3/17	919.460022	936.8736572	0.018938981	1.8938981	0

Date	GOOGL(Actual)	GOOGL(Predicted)	GOOGL(|Error|)	GOOGL(|% of Error|)	Change
7/5/17	932.2600098	923.3403931	0.009567735	0.956773479	1
7/6/17	927.6900024	922.2587891	0.005854556	0.585455634	1
7/7/17	940.8099976	921.2184448	0.020824132	2.082413249	1
7/10/17	951	942.123291	0.009334079	0.933407899	1
7/11/17	953.5300293	952.6435547	0.000929677	0.092967664	1
7/12/17	967.6599731	970.5852051	0.003022996	0.302299578	0
7/13/17	968.8499756	973.1146851	0.004401826	0.44018263	0
7/14/17	976.9099731	1002.766357	0.026467521	2.646752074	0
7/17/17	975.960022	997.6431885	0.02221727	2.221726999	0
7/18/17	986.9500122	990.6466064	0.003745473	0.374547252	0
7/19/17	992.7700195	993.4525757	0.000687527	0.068752695	0
7/20/17	992.1900024	998.0132446	0.00586908	0.586907985	0
7/21/17	993.8400269	999.2792969	0.005472983	0.547298323	0
7/24/17	998.3099976	988.9996948	0.009326064	0.932606403	1
7/25/17	969.0300293	1004.616394	0.036723696	3.67236957	0
7/26/17	965.3099976	971.2766724	0.006181097	0.618109712	0
7/27/17	952.5100098	967.5379028	0.01577715	1.577715017	0
7/28/17	958.3300171	957.9077148	0.000440665	0.044066473	1
7/31/17	945.5	967.7440186	0.023526195	2.352619544	0
8/1/17	946.5599976	952.303894	0.00606818	0.606817985	0
8/2/17	947.6400146	949.1997681	0.001645935	0.164593454	0
8/3/17	940.2999878	967.9934082	0.029451687	2.945168689	0
8/4/17	945.789978	944.9415283	0.00089708	0.089708046	1
8/7/17	945.75	954.836853	0.009608092	0.960809179	0
8/8/17	944.1900024	955.5303345	0.012010646	1.201064605	0
8/9/17	940.0800171	950.232666	0.010799771	1.079977117	0
8/10/17	923.5900269	943.9829102	0.022080017	2.208001725	0
8/11/17	930.0900269	927.3355103	0.002961559	0.296155922	1
8/14/17	938.9299927	923.326355	0.016618531	1.66185312	1
8/15/17	938.0800171	940.5541382	0.002637431	0.263743079	0
8/16/17	944.2700195	947.7922363	0.003730095	0.373009499	0
8/17/17	927.6599731	971.3974609	0.04714819	4.714819044	0
8/18/17	926.1799927	939.0266113	0.013870542	1.387054194	0
8/21/17	920.8699951	930.53125	0.010491443	1.049144287	0
8/22/17	940.4000244	907.5712891	0.03490933	3.490933031	1
8/23/17	942.5800171	934.7982788	0.008255785	0.825578533	1
8/24/17	936.8900146	939.6629639	0.002959738	0.295973825	0
8/25/17	930.5	931.5870972	0.001168294	0.116829353	0
8/28/17	928.1300049	921.2171631	0.00744814	0.74481396	1
8/29/17	935.75	943.3538208	0.008125911	0.812591054	0
8/30/17	943.6300049	936.7225342	0.007320105	0.73201051	1
8/31/17	955.2399902	956.7537231	0.001584662	0.158466236	0

Date	GOOGL(Actual)	GOOGL(Predicted)	GOOGL(|Error|)	GOOGL(|% of Error|)	Change
9/1/17	951.9899902	949.5429077	0.002570492	0.257049175	1
9/5/17	941.4799805	974.6622314	0.035244778	3.52447778	0
9/6/17	942.0200195	938.2761841	0.003974263	0.397426309	1
9/7/17	949.8900146	961.8035278	0.012541993	1.254199259	0
9/8/17	941.4099731	909.9007568	0.033470239	3.347023949	1
9/11/17	943.289978	955.7382813	0.013196687	1.319668721	0
9/12/17	946.6500244	934.4497681	0.012887822	1.288782153	1
9/13/17	950.4400024	933.2261963	0.018111408	1.811140776	1
9/14/17	940.1300049	960.7823486	0.02196754	2.196753956	0
9/15/17	935.289978	932.8556519	0.00260275	0.26027502	1
9/18/17	929.75	922.3469238	0.007962437	0.796243735	1
9/19/17	936.8599854	914.2409668	0.024143435	2.414343506	1
9/20/17	947.539978	935.5593872	0.01264389	1.264388952	1
9/21/17	947.5499878	937.5523071	0.010551085	1.055108476	1
9/22/17	943.2600098	959.7428589	0.017474344	1.7474344	0
9/25/17	934.2800293	956.4015503	0.023677614	2.367761359	0
9/26/17	937.4299927	914.539978	0.02441784	2.441783994	1
9/27/17	959.9000244	949.8955688	0.010422394	1.042239368	1
9/28/17	964.8099976	955.6419678	0.00950242	0.95024202	1
9/29/17	973.7199707	955.5092163	0.01870225	1.870224997	1
10/2/17	967.4699707	997.1953735	0.030724885	3.072488494	0
10/3/17	972.0800171	959.5913086	0.012847408	1.284740772	1
10/4/17	966.7800293	960.4296265	0.006568612	0.656861207	1
10/5/17	985.1900024	966.428833	0.0190432	1.904319972	1
10/6/17	993.6400146	1003.404907	0.009827395	0.982739497	0
10/9/17	992.3099976	1006.066345	0.013862954	1.386295352	0
10/10/17	987.7999878	992.3475952	0.004603773	0.460377336	0
10/11/17	1005.650024	976.3534546	0.029131973	2.913197316	1
10/12/17	1005.650024	1021.853088	0.016112031	1.611203142	0
10/13/17	1007.869995	1030.95813	0.022907849	2.29078494	0
10/16/17	1009.349976	996.4261475	0.01280411	1.28041096	1
10/17/17	1011	1026.805054	0.015633089	1.563308947	0
10/18/17	1012.73999	1009.621643	0.003079119	0.307911914	1
10/19/17	1001.840027	1033.994141	0.03209506	3.209505975	0
10/20/17	1005.070007	987.5241089	0.01745739	1.74573902	1
10/23/17	985.539978	1039.699951	0.054954618	5.495461822	0
10/24/17	988.4899902	1009.119263	0.02086948	2.086948045	0
10/25/17	991.460022	1000.111694	0.008726194	0.872619357	0
10/26/17	991.4199829	983.3980713	0.008091335	0.809133518	1
10/27/17	1033.670044	993.3442993	0.039012201	3.901220113	1
10/30/17	1033.130005	1028.430176	0.004549117	0.454911683	1
10/31/17	1033.040039	1021.180176	0.011480546	1.148054563	1

Date	GOOGL(Actual)	GOOGL(Predicted)	GOOGL(|Error|)	GOOGL(|% of Error|)	Change
11/1/17	1042.599976	1028.834473	0.013203053	1.32030528	1
11/2/17	1042.969971	1039.647827	0.003185272	0.318527245	1
11/3/17	1049.98999	1048.888794	0.001048768	0.10487684	1
11/6/17	1042.680054	1032.578125	0.009688426	0.968842581	1
11/7/17	1052.390015	1047.488159	0.004657832	0.465783151	1
11/8/17	1058.290039	1030.810425	0.025966052	2.596605197	1
11/9/17	1047.719971	1059.425171	0.011172069	1.117206924	0
11/10/17	1044.150024	1051.572144	0.007108288	0.710828789	0
11/13/17	1041.199951	1053.965942	0.012260845	1.226084493	0
11/14/17	1041.640015	1027.369263	0.013700272	1.370027196	1
11/15/17	1036.410034	1045.592041	0.008859434	0.885943417	0
11/16/17	1048.469971	1019.885681	0.027262859	2.726285905	1
11/17/17	1035.890015	1038.496826	0.002516495	0.25164946	0
11/20/17	1034.660034	1031.80249	0.002761819	0.276181917	1
11/21/17	1050.300049	1038.42041	0.01131071	1.131070964	1
11/22/17	1051.920044	1053.561157	0.001560112	0.156011223	0
11/24/17	1056.52002	1047.248291	0.008775724	0.87757241	1
11/27/17	1072.01001	1067.723633	0.003998449	0.399844861	1
11/28/17	1063.290039	1070.59436	0.006869548	0.686954753	0
11/29/17	1037.380005	1040.838257	0.003333641	0.333364052	0
11/30/17	1036.170044	1047.621826	0.01105203	1.105203014	0
12/1/17	1025.069946	1058.189819	0.032309867	3.230986744	0
12/4/17	1011.869995	1003.31134	0.008458256	0.845825579	1
12/5/17	1019.599976	1018.795349	0.000789159	0.078915898	1
12/6/17	1032.719971	1034.454102	0.001679188	0.167918787	0
12/7/17	1044.569946	1021.996155	0.021610608	2.161060832	1
12/8/17	1049.380005	1062.386719	0.012394665	1.239466481	0
12/11/17	1051.969971	1044.56604	0.007038158	0.70381579	1
12/12/17	1048.77002	1052.345825	0.003409523	0.340952305	0
12/13/17	1051.390015	1046.229614	0.00490817	0.490816962	1
12/14/17	1057.469971	1065.33606	0.007438593	0.743859308	0
12/15/17	1072	1086.182251	0.013229712	1.322971191	0
12/18/17	1085.089966	1068.866943	0.014950855	1.49508547	1
12/19/17	1079.780029	1053.001587	0.024799906	2.479990572	1
12/20/17	1073.560059	1064.29248	0.008632566	0.863256585	1
12/21/17	1070.849976	1104.034668	0.030989114	3.09891142	0
12/22/17	1068.859985	1069.348511	0.000457053	0.045705275	0
12/26/17	1065.849976	1057.603394	0.007737095	0.773709454	1
12/27/17	1060.199951	1065.355713	0.004863009	0.486300886	0
12/28/17	1055.949951	1082.536499	0.025177848	2.517784759	0
12/29/17	1053.400024	1082.7677	0.02787894	2.787894011	0

Date	MSFT (Actual)	MSFT (Predicted)	MSFT (|Error|)	MSFT (|% of Error|)	MSFT (Change)
1/3/17	62.58000183	61.89627457	0.010925652	1.092565153	1
1/4/17	62.29999924	62.81455231	0.008259279	0.825927872	0
1/5/17	62.29999924	63.425457	0.018065132	1.80651322	0
1/6/17	62.84000015	61.9218483	0.014610946	1.461094618	1
1/9/17	62.63999939	61.69145203	0.015142838	1.514283754	1
1/10/17	62.61999893	61.75377274	0.01383306	1.383305993	1
1/11/17	63.18999863	63.92245102	0.011591271	1.159127057	0
1/12/17	62.61000061	62.97895432	0.005892888	0.589288771	0
1/13/17	62.70000076	62.83665085	0.002179427	0.217942707	0
1/17/17	62.52999878	63.09840775	0.009090181	0.909018051	0
1/18/17	62.5	61.58094025	0.014704956	1.470495574	1
1/19/17	62.29999924	62.21004868	0.001443829	0.144382927	1
1/20/17	62.74000168	62.95913315	0.003492692	0.349269155	0
1/23/17	62.95999908	62.28789139	0.010675155	1.067515463	1
1/24/17	63.52000046	63.46764755	0.000824196	0.082419562	1
1/25/17	63.68000031	65.12814331	0.02274094	2.274093963	0
1/26/17	64.26999664	64.49494934	0.00350012	0.350012002	0
1/27/17	65.77999878	63.45140076	0.035399787	3.539978713	1
1/30/17	65.12999725	65.81659698	0.010541989	1.054198947	0
1/31/17	64.65000153	64.62751007	0.000347896	0.034789566	1
2/1/17	63.58000183	66.25559998	0.042082384	4.208238423	0
2/2/17	63.16999817	63.43405914	0.004180165	0.418016454	0
2/3/17	63.68000031	63.90782547	0.003577657	0.357765658	0
2/6/17	63.63999939	64.11683655	0.007492727	0.749272713	0
2/7/17	63.43000031	64.04175568	0.009644574	0.964457449	0
2/8/17	63.34000015	63.6484375	0.004869551	0.486955093	0
2/9/17	64.05999756	64.20334625	0.002237726	0.223772554	0
2/10/17	64	63.27847672	0.011273801	1.127380133	1
2/13/17	64.72000122	65.3219223	0.009300387	0.930038746	0
2/14/17	64.56999969	64.55182648	0.00028145	0.028144987	1
2/15/17	64.52999878	63.67999268	0.013172263	1.317226328	1
2/16/17	64.51999664	62.44430161	0.032171346	3.217134625	1
2/17/17	64.62000275	64.12158203	0.007713103	0.771310274	1
2/21/17	64.48999786	64.76701355	0.004295483	0.429548277	0
2/22/17	64.36000061	64.04832458	0.004842698	0.484269764	1
2/23/17	64.62000275	64.7117157	0.001419266	0.141926564	0
2/24/17	64.62000275	65.80903625	0.018400393	1.840039343	0
2/27/17	64.23000336	65.23384857	0.015628913	1.562891342	0
2/28/17	63.97999954	65.10862732	0.017640322	1.764032245	0
3/1/17	64.94000244	63.54917908	0.021417052	2.141705155	1
3/2/17	64.01000214	65.53239441	0.023783661	2.378366143	0
3/3/17	64.25	65.21574402	0.015031035	1.503103506	0

Date	MSFT (Actual)	MSFT (Predicted)	MSFT (|Error|)	MSFT (|% of Error|)	MSFT (Change)
3/6/17	64.26999664	64.49874115	0.003559118	0.355911814	0
3/7/17	64.40000153	65.89313507	0.023185303	2.318530343	0
3/8/17	64.98999786	63.82529831	0.017921213	1.792121306	1
3/9/17	64.73000336	64.14180756	0.009086912	0.908691157	1
3/10/17	64.93000031	64.61776733	0.004808763	0.480876304	1
3/13/17	64.70999908	64.66603088	0.000679465	0.06794653	1
3/14/17	64.41000366	64.76708221	0.005543837	0.554383686	0
3/15/17	64.75	65.30264282	0.008535025	0.853502471	0
3/16/17	64.63999939	65.81903839	0.018240083	1.824008301	0
3/17/17	64.87000275	63.70072174	0.018024988	1.802498847	1
3/20/17	64.93000031	65.5774231	0.009971089	0.997108873	0
3/21/17	64.20999908	64.86391449	0.010184012	1.018401235	0
3/22/17	65.02999878	63.57496262	0.022374846	2.237484604	1
3/23/17	64.87000275	63.9958992	0.013474695	1.347469538	1
3/24/17	64.98000336	65.41033173	0.006622474	0.662247371	0
3/27/17	65.09999847	65.20999908	0.001689718	0.168971752	0
3/28/17	65.29000092	65.12506104	0.002526266	0.252626557	1
3/29/17	65.47000122	64.59846497	0.013311994	1.331199426	1
3/30/17	65.70999908	65.64724731	0.000954981	0.095498055	1
3/31/17	65.86000061	65.91313171	0.000806728	0.080672797	0
4/3/17	65.55000305	65.56917572	0.000292489	0.029248922	0
4/4/17	65.73000336	64.78857422	0.01432267	1.432267018	1
4/5/17	65.55999756	65.03212738	0.008051711	0.805171113	1
4/6/17	65.73000336	66.03063965	0.004573806	0.457380619	0
4/7/17	65.68000031	63.93505478	0.026567381	2.656738088	1
4/10/17	65.52999878	65.58823395	0.00088868	0.088867953	0
4/11/17	65.48000336	65.92352295	0.006773359	0.677335914	0
4/12/17	65.23000336	66.81577301	0.024310434	2.431043424	0
4/13/17	64.94999695	66.78131104	0.028195754	2.819575369	0
4/17/17	65.48000336	64.58655548	0.013644591	1.364459097	1
4/18/17	65.38999939	65.93282318	0.008301327	0.830132701	0
4/19/17	65.04000092	64.44029236	0.009220611	0.92206113	1
4/20/17	65.5	64.86523438	0.009691078	0.969107822	1
4/21/17	66.40000153	66.43274689	0.000493153	0.0493153	0
4/24/17	67.52999878	65.77852631	0.025936214	2.593621425	1
4/25/17	67.91999817	66.56761169	0.019911462	1.991146244	1
4/26/17	67.83000183	68.55503082	0.010688913	1.068891305	0
4/27/17	68.26999664	68.17756653	0.001353891	0.135389075	1
4/28/17	68.45999908	69.02035522	0.008185161	0.818516128	0
5/1/17	69.41000366	66.76850128	0.038056508	3.805650771	1
5/2/17	69.30000305	69.75531769	0.006570197	0.657019671	0
5/3/17	69.08000183	70.81719208	0.025147513	2.514751256	0

Date	MSFT (Actual)	MSFT (Predicted)	MSFT (|Error|)	MSFT (|% of Error|)	MSFT (Change)
5/4/17	68.80999756	69.64761353	0.012172882	1.217288151	0
5/5/17	69	68.48374939	0.007481893	0.748189306	1
5/8/17	68.94000244	68.47808075	0.006700343	0.670034345	1
5/9/17	69.04000092	69.79380035	0.0109183	1.091829967	0
5/10/17	69.30999756	70.58966064	0.018462893	1.846289262	0
5/11/17	68.45999908	68.92640686	0.006812851	0.681285094	0
5/12/17	68.37999725	68.42753601	0.000695214	0.069521437	0
5/15/17	68.43000031	69.61994171	0.017389176	1.738917641	0
5/16/17	69.41000366	68.81194305	0.008616346	0.861634593	1
5/17/17	67.48000336	69.42006683	0.028750198	2.875019796	0
5/18/17	67.70999908	66.78681946	0.013634318	1.363431755	1
5/19/17	67.69000244	68.24058533	0.008133888	0.813388824	0
5/22/17	68.44999695	67.80644226	0.009401822	0.940182246	1
5/23/17	68.68000031	66.62255859	0.029956926	2.995692566	1
5/24/17	68.76999664	67.00691223	0.025637407	2.563740686	1
5/25/17	69.62000275	68.62908173	0.01423328	1.423327997	1
5/26/17	69.95999908	69.01828003	0.013460821	1.346082147	1
5/30/17	70.41000366	71.27845764	0.012334242	1.233424153	0
5/31/17	69.83999634	70.9333725	0.015655445	1.565544493	0
6/1/17	70.09999847	69.51557159	0.008337045	0.833704509	1
6/2/17	71.76000214	69.61206055	0.029932296	2.99322959	1
6/5/17	72.27999878	71.50898743	0.010667008	1.066700835	1
6/6/17	72.51999664	73.05635834	0.007396053	0.739605259	0
6/7/17	72.38999939	72.56227112	0.002379772	0.237977249	0
6/8/17	71.94999695	74.22955322	0.031682506	3.16825062	0
6/9/17	70.31999969	70.46849823	0.002111754	0.211175391	0
6/12/17	69.77999878	70.6239624	0.012094636	1.209463552	0
6/13/17	70.65000153	71.46740723	0.011569791	1.156979054	0
6/14/17	70.26999664	71.58966827	0.018780015	1.878001541	0
6/15/17	69.90000153	68.08822632	0.025919531	2.591953054	1
6/16/17	70	70.06085968	0.000869424	0.086942402	0
6/19/17	70.87000275	70.4184494	0.006371572	0.637157215	1
6/20/17	69.91000366	70.19979858	0.004145257	0.414525671	0
6/21/17	70.26999664	70.65536499	0.005484109	0.548410928	0
6/22/17	70.26000214	71.22541809	0.01374062	1.374061964	0
6/23/17	71.20999908	70.44786835	0.01070258	1.070258021	1
6/26/17	70.52999878	72.21508026	0.023891699	2.389169857	0
6/27/17	69.20999908	69.86981201	0.009533491	0.953349099	0
6/28/17	69.80000305	69.3963089	0.005783584	0.57835835	1
6/29/17	68.48999786	70.18003845	0.024675729	2.467572875	0
6/30/17	68.93000031	68.32240295	0.008814701	0.881470088	1
7/3/17	68.16999817	69.01529694	0.012399865	1.239986531	0

Date	MSFT (Actual)	MSFT (Predicted)	MSFT (|Error|)	MSFT (|% of Error|)	MSFT (Change)
7/5/17	69.08000183	67.2274704	0.026817188	2.681718767	1
7/6/17	68.56999969	69.51275635	0.013748821	1.374882087	0
7/7/17	69.45999908	69.0349884	0.006118783	0.611878326	1
7/10/17	69.98000336	68.76781464	0.017321929	1.732192934	1
7/11/17	69.98999786	69.6697464	0.004575675	0.457567489	1
7/12/17	71.15000153	69.12768555	0.028423274	2.842327394	1
7/13/17	71.76999664	71.35076141	0.005841372	0.58413716	1
7/14/17	72.77999878	72.96630859	0.002559904	0.255990401	0
7/17/17	73.34999847	72.876091	0.006460906	0.646090647	1
7/18/17	73.30000305	73.90126038	0.008202692	0.820269156	0
7/19/17	73.86000061	71.96214294	0.025695337	2.569533698	1
7/20/17	74.22000122	76.09300995	0.025235903	2.523590252	0
7/21/17	73.79000092	75.44593048	0.02244111	2.244110964	0
7/24/17	73.59999847	74.1791153	0.007868435	0.786843523	0
7/25/17	74.19000244	73.56970978	0.008360866	0.836086553	1
7/26/17	74.05000305	74.39282227	0.004629564	0.462956401	0
7/27/17	73.16000366	73.05452728	0.001441722	0.144172192	1
7/28/17	73.04000092	74.26817322	0.016815064	1.681506447	0
7/31/17	72.69999695	71.20131683	0.020614583	2.061458305	1
8/1/17	72.58000183	70.94645691	0.022506818	2.250681818	1
8/2/17	72.26000214	73.26321411	0.013883365	1.388336532	0
8/3/17	72.15000153	72.39591217	0.003408325	0.34083249	0
8/4/17	72.68000031	72.32997131	0.004816029	0.481602922	1
8/7/17	72.40000153	72.8651886	0.006425236	0.642523589	0
8/8/17	72.79000092	73.49243164	0.009650099	0.965009909	0
8/9/17	72.47000122	73.01979828	0.007586547	0.758654671	0
8/10/17	71.41000366	72.16196442	0.010530188	1.053018775	0
8/11/17	72.5	69.29514313	0.044204921	4.420492053	1
8/14/17	73.58999634	72.31260681	0.017358195	1.735819504	1
8/15/17	73.22000122	74.23241425	0.013827001	1.382700074	0
8/16/17	73.65000153	73.33419037	0.004287999	0.428799912	1
8/17/17	72.40000153	73.29541779	0.012367628	1.236762758	0
8/18/17	72.48999786	72.31651306	0.002393224	0.239322404	1
8/21/17	72.15000153	73.2988739	0.015923386	1.592338644	0
8/22/17	73.16000366	72.44969177	0.009709019	0.970901921	1
8/23/17	72.72000122	72.02944946	0.009496036	0.949603599	1
8/24/17	72.69000244	71.00566101	0.02317157	2.317157015	1
8/25/17	72.81999969	74.00667572	0.016296018	1.629601792	0
8/28/17	72.83000183	73.55397034	0.009940526	0.994052552	0
8/29/17	73.05000305	72.61424255	0.005965236	0.596523564	1
8/30/17	74.01000214	72.3632431	0.022250492	2.225049213	1
8/31/17	74.76999664	74.25831604	0.006843395	0.68433946	1

Date	MSFT (Actual)	MSFT (Predicted)	MSFT (|Error|)	MSFT (|% of Error|)	MSFT (Change)
9/1/17	73.94000244	75.2538681	0.017769348	1.776934788	0
9/5/17	73.61000061	75.74357605	0.028984858	2.898485772	0
9/6/17	73.40000153	74.17906189	0.010613902	1.061390154	0
9/7/17	74.33999634	73.19071198	0.015459839	1.545983925	1
9/8/17	73.98000336	73.85643005	0.001670361	0.167036091	1
9/11/17	74.76000214	74.65577698	0.00139413	0.139412994	1
9/12/17	74.68000031	75.16043091	0.00643319	0.643318985	0
9/13/17	75.20999908	74.72499084	0.00644872	0.644872012	1
9/14/17	74.76999664	76.144104	0.01837779	1.837779023	0
9/15/17	75.30999756	74.71405792	0.007913155	0.791315455	1
9/18/17	75.16000366	76.0353775	0.011646804	1.16468044	0
9/19/17	75.44000244	74.8110199	0.00833752	0.833752006	1
9/20/17	74.94000244	75.78826904	0.011319276	1.131927595	0
9/21/17	74.20999908	74.53412628	0.004367703	0.436770264	0
9/22/17	74.41000366	74.04825592	0.004861547	0.486154715	1
9/25/17	73.26000214	74.78177643	0.020772239	2.077223919	0
9/26/17	73.26000214	71.80723572	0.019830281	1.983028091	1
9/27/17	73.84999847	71.01264954	0.038420435	3.842043504	1
9/28/17	73.87000275	73.57827759	0.00394917	0.394916954	1
9/29/17	74.48999786	74.41226196	0.001043575	0.1043575	1
10/2/17	74.61000061	74.88022614	0.003621841	0.362184062	0
10/3/17	74.26000214	74.71336365	0.006105056	0.61050565	0
10/4/17	74.69000244	74.14030457	0.007359725	0.735972496	1
10/5/17	75.97000122	75.95484161	0.000199547	0.019954728	1
10/6/17	76	77.03775024	0.013654608	1.365460828	0
10/9/17	76.29000092	75.85917664	0.005647192	0.564719178	1
10/10/17	76.29000092	74.75278473	0.020149643	2.01496426	1
10/11/17	76.41999817	76.13236237	0.003763881	0.376388128	1
10/12/17	77.12000275	76.57681274	0.007043438	0.70434385	1
10/13/17	77.48999786	77.34629059	0.001854527	0.185452681	1
10/16/17	77.65000153	78.6955719	0.013465169	1.346516889	0
10/17/17	77.58999634	77.85002136	0.00335127	0.335126999	0
10/18/17	77.61000061	78.08831024	0.00616299	0.616298988	0
10/19/17	77.91000366	78.07904053	0.002169643	0.21696426	0
10/20/17	78.80999756	78.17435455	0.008065512	0.80655124	1
10/23/17	78.83000183	78.88605499	0.000711064	0.07110638	0
10/24/17	78.86000061	79.62807465	0.009739717	0.973971654	0
10/25/17	78.62999725	81.24241638	0.033224206	3.322420642	0
10/26/17	78.76000214	77.32929993	0.018165341	1.816534065	1
10/27/17	83.80999756	81.17913055	0.03139085	3.13908495	1
10/30/17	83.88999939	85.24002838	0.016092848	1.609284803	0
10/31/17	83.18000031	83.3249054	0.001742066	0.174206647	0

Date	MSFT (Actual)	MSFT (Predicted)	MSFT (|Error|)	MSFT (|% of Error|)	MSFT (Change)
11/1/17	83.18000031	84.00901794	0.00996655	0.996655039	0
11/2/17	84.05000305	84.1754303	0.001492293	0.14922932	0
11/3/17	84.13999939	84.41815948	0.00330592	0.330591993	0
11/6/17	84.47000122	83.76177216	0.008384386	0.838438608	1
11/7/17	84.26999664	89.72706604	0.064756967	6.475696713	0
11/8/17	84.55999756	85.19877625	0.007554147	0.755414739	0
11/9/17	84.08999634	85.43894958	0.01604178	1.604177989	0
11/10/17	83.87000275	83.82659149	0.000517602	0.051760167	1
11/13/17	83.93000031	84.04106903	0.00132335	0.132334954	0
11/14/17	84.05000305	82.75993347	0.015348834	1.534883399	1
11/15/17	82.98000336	86.36929321	0.040844657	4.084465653	0
11/16/17	83.19999695	84.01673889	0.00981661	0.981661025	0
11/17/17	82.40000153	81.8441925	0.006745255	0.674525509	1
11/20/17	82.52999878	83.84815216	0.01597181	1.597180963	0
11/21/17	83.72000122	82.26635742	0.017363161	1.736316085	1
11/22/17	83.11000061	85.08027649	0.023706844	2.370684408	0
11/24/17	83.26000214	83.16041565	0.00119609	0.119609036	1
11/27/17	83.87000275	85.12267303	0.014935856	1.493585575	0
11/28/17	84.87999725	83.33702087	0.018178327	1.817832701	1
11/29/17	83.33999634	84.63697052	0.015562446	1.556244586	0
11/30/17	84.16999817	83.55953217	0.007252774	0.72527742	1
12/1/17	84.26000214	83.58071899	0.008061751	0.806175079	1
12/4/17	81.08000183	85.96943665	0.060303833	6.030383334	0
12/5/17	81.58999634	79.53379822	0.025201596	2.520159632	1
12/6/17	82.77999878	81.57292938	0.014581655	1.458165515	1
12/7/17	82.48999786	84.53215027	0.024756365	2.475636452	0
12/8/17	84.16000366	83.57971191	0.006895101	0.689510116	1
12/11/17	85.23000336	85.07749939	0.001789323	0.178932259	1
12/12/17	85.58000183	87.64589691	0.024139928	2.41399277	0
12/13/17	85.34999847	86.41278839	0.012452138	1.245213766	0
12/14/17	84.69000244	86.94078827	0.026576759	2.657675929	0
12/15/17	86.84999847	83.03577423	0.043917377	4.391737655	1
12/18/17	86.37999725	87.13124847	0.008697051	0.869705062	0
12/19/17	85.83000183	89.87187195	0.047091577	4.70915772	0
12/20/17	85.51999664	86.2980957	0.009098446	0.9098446	0
12/21/17	85.5	83.92521667	0.018418519	1.841851883	1
12/22/17	85.51000214	87.3765564	0.021828491	2.182849124	0
12/26/17	85.40000153	85.1752243	0.002632052	0.263205171	1
12/27/17	85.70999908	85.89976501	0.002214047	0.221404666	0
12/28/17	85.72000122	84.93820953	0.009120295	0.912029482	1
12/29/17	85.54000092	86.45542145	0.010701667	1.070166659	0

Date	NFLX (Actual)	NFLX (Predicted)	NFLX (|Error|)	NFLX (|% of Error|)	NFLX (Change)
1/3/17	127.4899979	126.6228867	0.006801406	0.680140592	1
1/4/17	129.4100037	129.2349091	0.001353022	0.135302218	1
1/5/17	131.8099976	132.5314941	0.005473762	0.547376228	0
1/6/17	131.0700073	127.6811905	0.025855014	2.58550141	1
1/9/17	130.9499969	132.573288	0.012396266	1.239626575	0
1/10/17	129.8899994	133.15625	0.025146283	2.514628321	0
1/11/17	130.5	126.4863358	0.030756047	3.0756047	1
1/12/17	129.1799927	124.0897751	0.039404072	3.940407187	1
1/13/17	133.6999969	129.3023071	0.03289222	3.289221972	1
1/17/17	132.8899994	132.8445892	0.000341712	0.034171238	1
1/18/17	133.2599945	133.8703156	0.004579927	0.457992684	0
1/19/17	138.4100037	135.759079	0.019152695	1.915269531	1
1/20/17	138.6000061	134.3358307	0.030766055	3.076605499	1
1/23/17	137.3899994	135.2847595	0.015323094	1.532309409	1
1/24/17	140.1100006	136.0575714	0.028923197	2.892319672	1
1/25/17	139.5200043	145.2140045	0.040811352	4.081135243	0
1/26/17	138.9600067	139.9681396	0.007254842	0.72548422	0
1/27/17	142.4499969	137.8025055	0.032625422	3.262542188	1
1/30/17	141.2200012	143.9526367	0.019350201	1.935020089	0
1/31/17	140.7100067	135.8353271	0.034643445	3.464344516	1
2/1/17	140.7799988	143.3412018	0.018192947	1.819294691	0
2/2/17	139.1999969	142.2443695	0.021870494	2.187049389	0
2/3/17	140.25	143.6114807	0.023967776	2.396777645	0
2/6/17	140.9700012	142.927475	0.013885747	1.388574671	0
2/7/17	144	141.3853607	0.018157218	1.815721765	1
2/8/17	144.7400055	147.8432465	0.021440106	2.144010551	0
2/9/17	144.1399994	140.7177429	0.023742586	2.374258637	1
2/10/17	144.8200073	150.9224396	0.042138048	4.213804752	0
2/13/17	143.1999969	149.3195953	0.042734627	4.273462668	0
2/14/17	140.8200073	146.965744	0.043642495	4.364249483	0
2/15/17	142.2700043	142.4842072	0.001505608	0.15056082	0
2/16/17	142.0099945	137.8142242	0.029545598	2.954559773	1
2/17/17	142.2200012	146.4264374	0.029576967	2.957696654	0
2/21/17	142.6000061	143.131012	0.003723744	0.37237436	0
2/22/17	143.8600006	145.0519104	0.008285207	0.828520674	0
2/23/17	142.7799988	146.5361328	0.026307145	2.630714513	0
2/24/17	143.25	143.9396057	0.004814001	0.481400127	0
2/27/17	143.4100037	143.2304077	0.001252325	0.125232514	1
2/28/17	142.1300049	146.3037109	0.029365411	2.936541103	0
3/1/17	142.6499939	141.7709656	0.006162134	0.616213353	1
3/2/17	139.5299988	144.5496216	0.035975222	3.597522154	0
3/3/17	139.1399994	141.4329071	0.016479142	1.647914201	0

Date	NFLX (Actual)	NFLX (Predicted)	NFLX (|Error|)	NFLX (|% of Error|)	NFLX (Change)
3/6/17	141.9400024	139.2792511	0.018745607	1.874560677	1
3/7/17	141.4299927	143.2223053	0.01267279	1.267279033	0
3/8/17	140.3200073	138.0988922	0.015828926	1.582892612	1
3/9/17	140.5299988	139.3422699	0.008451782	0.845178217	1
3/10/17	140.8899994	137.7260895	0.022456598	2.245659754	1
3/13/17	143.5200043	140.973938	0.017740149	1.774014905	1
3/14/17	143.1900024	145.1336823	0.013574131	1.357413083	0
3/15/17	145.25	141.6377106	0.024869462	2.486946248	1
3/16/17	144.3899994	140.7735291	0.025046542	2.504654229	1
3/17/17	145.1100006	143.7921295	0.009081877	0.908187684	1
3/20/17	145.8300018	146.7678223	0.006430916	0.643091556	0
3/21/17	142.4199982	147.8109894	0.037852768	3.785276785	0
3/22/17	142.6499939	136.3748322	0.043989919	4.398991913	1
3/23/17	141.8399963	142.2825317	0.003119962	0.311996206	0
3/24/17	142.0200043	141.3531647	0.004695392	0.469539221	1
3/27/17	144.0599976	141.9808807	0.014432298	1.443229802	1
3/28/17	145.1699982	143.0938568	0.014301449	1.430144906	1
3/29/17	146.4700012	146.1141815	0.002429301	0.242930092	1
3/30/17	148.0599976	147.4442291	0.004158912	0.415891176	1
3/31/17	147.8099976	148.177597	0.002486973	0.248697307	0
4/3/17	146.9199982	145.1717682	0.011899197	1.189919654	1
4/4/17	145.5	141.5144958	0.027391782	2.739178203	1
4/5/17	143.6199951	142.646637	0.006777316	0.677731633	1
4/6/17	143.7400055	144.9924774	0.008713454	0.871345401	0
4/7/17	143.1100006	136.3287048	0.047385197	4.738519713	1
4/10/17	143.8500061	141.2879639	0.017810512	1.781051233	1
4/11/17	144.3500061	145.9952545	0.011397633	1.139763277	0
4/12/17	143.8300018	142.0061646	0.012680506	1.268050633	1
4/13/17	142.9199982	145.7920685	0.02009565	2.009565011	0
4/17/17	147.25	149.0023499	0.011900509	1.190050878	0
4/18/17	143.3600006	149.4926147	0.042777721	4.277772084	0
4/19/17	139.7599945	144.9112701	0.036858011	3.685801104	0
4/20/17	141.1799927	139.2859497	0.013415803	1.34158032	1
4/21/17	142.8699951	144.8699646	0.013998527	1.399852708	0
4/24/17	143.8300018	143.2020111	0.004366201	0.436620135	1
4/25/17	152.1600037	143.0026855	0.060182162	6.018216163	1
4/26/17	150.1699982	154.0653992	0.025939941	2.593994141	0
4/27/17	153.0800018	152.6529083	0.002790002	0.279000192	1
4/28/17	152.1999969	148.1923065	0.026331738	2.633173764	1
5/1/17	155.3500061	150.3279572	0.032327317	3.23273167	1
5/2/17	156.4499969	154.0806122	0.015144678	1.514467783	1
5/3/17	155.5899963	158.0455933	0.015782487	1.578248665	0

Date	NFLX (Actual)	NFLX (Predicted)	NFLX (|Error|)	NFLX (|% of Error|)	NFLX (Change)
5/4/17	157.25	157.4006042	0.000957738	0.095773768	0
5/5/17	156.6000061	157.036499	0.002787311	0.27873111	0
5/8/17	156.3800049	159.7161102	0.021333324	2.133332379	0
5/9/17	157.4600067	155.0500031	0.015305497	1.530549675	1
5/10/17	160.2799988	159.6871185	0.003699028	0.36990284	1
5/11/17	158.5399933	157.3652039	0.007410051	0.741005084	1
5/12/17	160.8099976	159.412323	0.008691465	0.869146548	1
5/15/17	160.0200043	159.5786285	0.002758253	0.275825337	1
5/16/17	159.4100037	164.2266235	0.030215293	3.021529317	0
5/17/17	153.1999969	158.6488953	0.03556722	3.55672203	0
5/18/17	155.6999969	151.0061646	0.030146644	3.014664352	1
5/19/17	157.0200043	156.9716797	0.000307761	0.03077607	1
5/22/17	157.1600037	161.9906311	0.030737003	3.073700331	0
5/23/17	157.9499969	154.6378174	0.020969799	2.096979879	1
5/24/17	157.75	160.3508301	0.016487038	1.648703776	0
5/25/17	163.0500031	162.4045258	0.003958769	0.395876914	1
5/26/17	162.4299927	165.6536407	0.019846383	1.984638348	0
5/30/17	163.2200012	165.0897369	0.01145531	1.145530958	0
5/31/17	163.0700073	165.7250061	0.016281344	1.6281344	0
6/1/17	162.9900055	160.6051483	0.014631922	1.463192236	1
6/2/17	165.1799927	162.2584381	0.017687097	1.768709719	1
6/5/17	165.0599976	163.4985657	0.009459783	0.945978332	1
6/6/17	165.1699982	168.6244507	0.020914529	2.091452852	0
6/7/17	165.6100006	171.7626953	0.037151709	3.71517092	0
6/8/17	165.8800049	167.2886047	0.00849168	0.849168003	0
6/9/17	158.0299988	163.0098267	0.031511914	3.151191399	0
6/12/17	151.4400024	160.6616516	0.060893085	6.089308485	0
6/13/17	152.7200012	152.2293243	0.003212918	0.321291829	1
6/14/17	152.1999969	154.1304321	0.012683542	1.268354245	0
6/15/17	151.7599945	150.2916412	0.009675496	0.967549626	1
6/16/17	152.3800049	146.7036591	0.037251253	3.725125268	1
6/19/17	153.3999939	155.0439911	0.010717061	1.071706135	0
6/20/17	152.0500031	153.7796783	0.0113757	1.137570012	0
6/21/17	155.0299988	150.7777252	0.027428715	2.742871456	1
6/22/17	154.8899994	155.2943878	0.00261081	0.261081033	0
6/23/17	158.0200043	157.2852631	0.004649672	0.464967219	1
6/26/17	157.5	156.123642	0.008738781	0.873878133	1
6/27/17	151.0299988	157.6658783	0.043937493	4.393749312	0
6/28/17	153.4100037	152.4407349	0.006318159	0.631815894	1
6/29/17	150.0899963	152.6575317	0.017106639	1.710663922	0
6/30/17	149.4100037	148.2837372	0.007538093	0.753809279	1
7/3/17	146.1699982	145.5192566	0.004451951	0.445195055	1

Date	NFLX (Actual)	NFLX (Predicted)	NFLX (|Error|)	NFLX (|% of Error|)	NFLX (Change)
7/5/17	147.6100006	145.1066132	0.01695947	1.695946977	1
7/6/17	146.25	152.1788025	0.040538821	4.053882137	0
7/7/17	150.1799927	146.8943787	0.02187784	2.187784016	1
7/10/17	152.6699982	148.9330139	0.024477528	2.447752841	1
7/11/17	154.3300018	150.941391	0.021956915	2.195691504	1
7/12/17	158.75	157.894516	0.005388875	0.538887549	1
7/13/17	158.2100067	161.1861877	0.018811585	1.881158538	0
7/14/17	161.1199951	161.1855316	0.000406756	0.040675583	0
7/17/17	161.6999969	158.1688538	0.02183762	2.183762006	1
7/18/17	183.6000061	167.3973846	0.088249572	8.824957162	1
7/19/17	183.8600006	188.4806671	0.02513144	2.513143979	0
7/20/17	183.6000061	185.6373444	0.011096613	1.109661348	0
7/21/17	188.5399933	184.1766357	0.023142876	2.31428761	1
7/24/17	187.9100037	189.2565765	0.007166052	0.716605224	0
7/25/17	186.9700012	199.1046753	0.064901717	6.490171701	0
7/26/17	189.0800018	187.0171356	0.010910018	1.091001835	1
7/27/17	182.6799927	193.3304901	0.058301389	5.830138922	0
7/28/17	184.0399933	175.5440369	0.046163641	4.616364092	1
7/31/17	181.6600037	187.5363007	0.032347776	3.2347776	0
8/1/17	182.0299988	179.5669861	0.013530807	1.353080664	1
8/2/17	180.7400055	176.8555756	0.021491811	2.149181068	1
8/3/17	179.2299957	185.1898804	0.03325272	3.325271979	0
8/4/17	180.2700043	180.4098663	0.000775848	0.077584764	0
8/7/17	181.3300018	183.6369934	0.012722614	1.272261422	0
8/8/17	178.3600006	183.4653473	0.028623832	2.862383239	0
8/9/17	175.7799988	173.0698547	0.015417818	1.541781798	1
8/10/17	169.1399994	172.5032959	0.019884691	1.988469064	0
8/11/17	171.3999939	166.449173	0.028884605	2.888460457	1
8/14/17	171	171.8626099	0.005044502	0.504450221	0
8/15/17	168.5	172.7445831	0.025190404	2.519040368	0
8/16/17	169.9799957	170.7086029	0.004286429	0.428642891	0
8/17/17	166.0899963	169.9378357	0.023167195	2.31671948	0
8/18/17	166.5399933	166.1129456	0.002564235	0.256423536	1
8/21/17	166.7599945	169.50177	0.016441446	1.64414458	0
8/22/17	169.3399963	173.4290924	0.024147255	2.414725535	0
8/23/17	169.0599976	173.2491455	0.024779061	2.477906086	0
8/24/17	168.1300049	169.2438507	0.006624908	0.662490819	0
8/25/17	165.9499969	168.5979004	0.015956031	1.595603116	0
8/28/17	167.1199951	171.1482086	0.02410372	2.410371974	0
8/29/17	168.8099976	167.0792694	0.010252521	1.025252137	1
8/30/17	174.6900024	168.6607513	0.034514003	3.45140025	1
8/31/17	174.7100067	175.6849976	0.005580624	0.55806241	0

Date	NFLX (Actual)	NFLX (Predicted)	NFLX (|Error|)	NFLX (|% of Error|)	NFLX (Change)
9/1/17	174.7400055	180.1957855	0.031222273	3.122227266	0
9/5/17	174.5200043	174.5248718	2.79E-05	0.002789109	0
9/6/17	179.25	170.5715332	0.048415434	4.841543362	1
9/7/17	179	173.2829132	0.031939033	3.193903342	1
9/8/17	176.4199982	179.2109985	0.015820203	1.582020335	0
9/11/17	181.7400055	174.6016235	0.039277989	3.927798942	1
9/12/17	185.1499939	186.7334137	0.008552092	0.855209213	0
9/13/17	183.6399994	174.9325256	0.047415998	4.741599783	1
9/14/17	182.6300049	186.3141174	0.020172548	2.017254755	0
9/15/17	182.3500061	184.0566559	0.009359198	0.935919769	0
9/18/17	184.6199951	181.1217957	0.018948108	1.894810796	1
9/19/17	185.6799927	180.6569061	0.027052384	2.705238387	1
9/20/17	185.5099945	192.5578766	0.037991926	3.799192607	0
9/21/17	188.7799988	189.8244019	0.005532382	0.553238206	0
9/22/17	187.3500061	191.7219543	0.023335725	2.333572507	0
9/25/17	178.5500031	186.9836884	0.047234304	4.723430425	0
9/26/17	179.3800049	181.3308716	0.010875609	1.087560877	0
9/27/17	181.9700012	180.7078094	0.006936263	0.693626283	1
9/28/17	180.6999969	183.1239471	0.013414224	1.341422368	0
9/29/17	181.3500061	182.236084	0.00488601	0.488600973	0
10/2/17	177.0099945	185.3088684	0.046883646	4.68836464	0
10/3/17	179.1900024	169.1156616	0.056221556	5.622155592	1
10/4/17	184.4499969	179.8739624	0.024809079	2.480907924	1
10/5/17	194.3899994	182.6546326	0.060370218	6.037021801	1
10/6/17	198.0200043	179.701416	0.092508778	9.250877798	1
10/9/17	196.8699951	199.8564758	0.015169811	1.516981144	0
10/10/17	195.0800018	200.22789	0.0263886	2.638860047	0
10/11/17	194.9499969	191.8933105	0.015679335	1.567933522	1
10/12/17	195.8600006	194.0236206	0.009375983	0.937598292	1
10/13/17	199.4900055	195.6799622	0.019098919	1.909891888	1
10/16/17	202.6799927	196.0722656	0.032601774	3.260177374	1
10/17/17	199.4799957	199.0961456	0.001924254	0.192425353	1
10/18/17	195.5399933	202.5524902	0.035862215	3.586221486	0
10/19/17	195.1300049	199.2865753	0.021301543	2.130154334	0
10/20/17	194.1600037	190.2632904	0.020069597	2.006959729	1
10/23/17	192.4700012	195.1351318	0.013846993	1.384699252	0
10/24/17	196.0200043	191.2304535	0.024433989	2.443398908	1
10/25/17	193.7700043	196.8167877	0.015723711	1.572371088	0
10/26/17	195.2100067	197.3453217	0.010938553	1.093855314	0
10/27/17	199.5399933	193.0090179	0.032730158	3.273015842	1
10/30/17	198.3699951	204.4160614	0.030478735	3.047873452	0
10/31/17	196.4299927	191.5178986	0.025006844	2.500684373	1

Date	NFLX (Actual)	NFLX (Predicted)	NFLX (|Error|)	NFLX (|% of Error|)	NFLX (Change)
11/1/17	198	194.8776703	0.015769342	1.576934196	1
11/2/17	199.3200073	200.0783997	0.003804898	0.380489812	0
11/3/17	200.0099945	199.6470032	0.001814866	0.181486597	1
11/6/17	200.1300049	201.6640472	0.007665229	0.766522903	0
11/7/17	195.8899994	203.0174866	0.036385152	3.638515249	0
11/8/17	196.4400024	204.3937531	0.040489465	4.0489465	0
11/9/17	193.8999939	184.6958618	0.04746845	4.746844992	1
11/10/17	192.0200043	190.9012146	0.005826422	0.582642248	1
11/13/17	195.0800018	192.1283875	0.015130277	1.513027679	1
11/14/17	195.7100067	195.4468231	0.001344763	0.134476309	1
11/15/17	192.1199951	190.4080811	0.00891065	0.891065039	1
11/16/17	195.5099945	196.9771118	0.007504053	0.750405295	0
11/17/17	193.1999969	195.0603333	0.00962907	0.962906983	0
11/20/17	194.1000061	193.6131134	0.002508463	0.250846311	1
11/21/17	196.2299957	196.1140289	0.000590974	0.059097388	1
11/22/17	196.3200073	197.0648804	0.003794178	0.37941779	0
11/24/17	195.75	199.3336029	0.01830704	1.830703951	0
11/27/17	195.0500031	194.4551849	0.003049567	0.304956734	1
11/28/17	199.1799927	191.1617126	0.040256452	4.025645182	1
11/29/17	188.1499939	196.4497375	0.044112377	4.411237687	0
11/30/17	187.5800018	189.7856445	0.01175841	1.175841037	0
12/1/17	186.8200073	188.1220703	0.006969612	0.696961209	0
12/4/17	184.0399933	187.7335663	0.020069404	2.006940357	0
12/5/17	184.2100067	182.5426178	0.009051565	0.905156508	1
12/6/17	185.3000031	185.0271301	0.001472601	0.147260074	1
12/7/17	185.1999969	185.2735901	0.000397371	0.039737116	0
12/8/17	188.5399933	181.3804779	0.037973456	3.797345608	1
12/11/17	186.2200012	191.727951	0.029577648	2.957764827	0
12/12/17	185.7299957	186.7776794	0.005640897	0.56408965	0
12/13/17	187.8600006	185.4616241	0.012766829	1.276682876	1
12/14/17	189.5599976	183.697525	0.030926738	3.092673793	1
12/15/17	190.1199951	189.0155487	0.005809207	0.580920698	1
12/18/17	190.4199982	184.7318878	0.029871391	2.987139113	1
12/19/17	187.0200043	192.9334106	0.031619113	3.161911294	0
12/20/17	188.8200073	186.6773071	0.011347845	1.13478452	1
12/21/17	188.6199951	185.6744537	0.015616274	1.561627351	1
12/22/17	189.9400024	188.1910553	0.009207892	0.920789223	1
12/26/17	187.7599945	194.8712463	0.037874158	3.787415847	0
12/27/17	186.2400055	188.9363556	0.014477825	1.447782479	0
12/28/17	192.7100067	185.3257599	0.038317923	3.83179225	1
12/29/17	191.9600067	192.402832	0.002306862	0.230686227	0