STOCK MARKET PREDICTION USING MACHINE LEARNING

by

MANAV SINHA(1814310109)

MANSI DEWAL(1814310112)

KIRTI SHARMA(1814310103)

MANAS GUPTA(1814310107)

Department of Computer Science and Engineering

IMS Engineering College

NH-09, Adhyatmik Nagar, Near Dasna,

Distt: Ghaziabad, UP

June, 2022

STOCK MARKET PREDICTION USING MACHINE LEARNING

by

Manav Sinha (1814310109)

Mansi Dewal(1814310109)

Kirti Sharma(1814310103)

Manas Gupta(1814310107)

Submitted to the Department of Computer Science and Engineering

in partial fulfillment of the requirements

for the degree of

Bachelor of Technology

in

Computer Science and Engineering

IMS Engineering College

Dr. A.P.J Abdul Kalam Technical University, Uttar Pradesh, Lucknow.

June, 2022

Department Vision and Mission

Vision

To be recognized as a Center of Excellence imparting quality education and creating new opportunities for students to meet the challenges of technological development in Computer Science & Engineering.

Mission

- To promote technical proficiency by adopting effective teaching learning processes.
- To provide an environment & opportunity for students to bring out their inherent talents for all round development.
- To promote the latest technologies in Computer Science & Engineering and across disciplines in order to serve the needs of Industry, Government, Society, and the scientific community.
- To educate students to be Successful, Ethical and Effective problem-solvers and Life-Long learners who will contribute positively to the society.

PROGRAMME EDUCATIONAL OBJECTIVES:

1. Graduates of the program will be able to apply fundamental principles of engineering in problem solving and understand the role of computing in multiple disciplines.

2. Graduates will learn to apply various computational techniques & tools for developing solutions & projects in real world.

3. Be employed as computer science professionals beyond entry-level positions or be making satisfactory progress in graduate programs.

4. Demonstrate that they can function, communicate, collaborate and continue to learn effectively as ethically and socially responsible computer science professionals.

PROGRAMME OUTCOMES:

Engineering Graduates will be able to:

PO1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

PO6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change

PROGRAMME SPECIFIC OUTCOMES:

1. Foundation of Computer System: Ability to understand the principles and working of computer systems.
2. Foundations of Software development: Possess professional skills and knowledge of software design process. Familiarity and practical competence with a broad range of programming language and open-source platforms.
3. Foundation of mathematical concepts: Ability to apply mathematical methodologies to solve computation task, model real world problem using appropriate data structure and suitable algorithm.
4. Applications of Computing and Research Ability: Ability to use knowledge in various domains to identify research gaps and hence to provide solution to new ideas and innovations.

COURSE OUTCOMES

CO1	The students can effectively collaborate in groups to achieve a common goal.
CO2	Students can improve their capacity to communicate effectively with a diverse group of people.
CO3	Students learn how to design a software or hardware product by learning technical skills, conducting research, and responding ethically.
CO4	The students use what they've learned to create and implement a business plan for an entrepreneurial venture
CO5	Students build self-learning skills and apply them to lifelong learning.

CO-PO Mapping												
Course Outcome	**PO 1**	**PO 2**	**PO 3**	**PO 4**	**PO 5**	**PO 6**	**PO 7**	**PO 8**	**PO 9**	**PO10**	**PO11**	**PO12**
CO1	-	-	3	3	3	2	-	-	3	3	3	3
CO2	-	-	-	-	-	-	-	-	3	3	2	3
CO3	3	3	3	3	3	2	-	3	3	3	3	3
CO4	3	3	3	3	3	-	2	3	2	3	3	2
CO5	3	3	3	2	3	-	2	-	3	-	3	3
Avg.	**3**	**3**	**3**	**2.8**	**3**	**2**	**2**	**3**	**2.8**	**3**	**2.6**	**2.8**

CO-PSO Mapping				
COs	**PSO1**	**PSO2**	**PSO3**	**PSO4**
CO1	3	2	1	2
CO2	2	3	3	2
CO3	3	3	1	2
CO4	3	3	2	2
CO5	3	3	2	2
Avg.	**2.8**	**2.8**	**1.8**	**2**

DECLARATION

We hereby declare that this submission is our own work and that, to the best of our knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Signature:

Name : Manav Sinha

Roll No.: 1814310109

Date :

Signature:

Name : Mansi Dewal

Roll No.: 184310112

Date :

Signature:

Name : Kirti Sharma

Roll No.: 1814310103

Date :

Signature:

Name : Manas Gupta

Roll No.: 1814310108

Date :

CERTIFICATE

This is to certify that Project Report entitled “STOCK MARKET PREDICTION USING MACHINE LEARNING” which is submitted by Mr. Manav Sinha, Ms. Mansi Dewal, Ms. Kirti Sharma, Mr. Manas Gupta in partial fulfillment of the requirement for the award of degree B. Tech. in Department of Computer Science and Engineering of Dr. APJ Abdul Kalam Technical University, Uttar Pradesh, Lucknow is a record of the candidate’s own work carried out by him/her under my/our supervision. The matter embodied in this thesis is original and has not been submitted for the award of any other degree.

Supervisor : Dr. Sonali Mathur (Head of Department)

Date :

ACKNOWLEDGEMENT

It gives us a great sense of pleasure to present the report of the B. Tech Project undertaken during B. Tech. Final Year. We owe special debt of gratitude to Dr. Sonali Mathur, Department of Computer Science & Engineering, IMS Engineering College, Ghaziabad for her constant support and guidance throughout the course of our work. Her sincerity, thoroughness and perseverance have been a constant source of inspiration for us. It is only his cognizant efforts that our endeavors have seen light of the day.

We also take the opportunity to acknowledge the contribution of Dr. Sonali Mathur., Head, Department of Computer Science & Engineering, IMS Engineering College, Ghaziabad for her full support and assistance during the development of the project.

We also do not like to miss the opportunity to acknowledge the contribution of all faculty members of the department for their kind assistance and cooperation during the development of our project. Last but not the least, we acknowledge our friends for their contribution in the completion of the project.

Signature:

Name : Manav Sinha

Roll No.: 1814310109

Date :

Signature:

Name : Kirti Sharma

Roll No.:1 814310103

Date :

Signature:

Name : Mansi Dewal

Roll No . : 1814310112

Date :

Signature:

Name : Manas Gupta

Roll No . : 1814310107

Date :

ABSTRACT

Investment firms, hedge funds and even individuals have been using financial models to better understand market behavior and make profitable investments and trades. A wealth of information is available in the form of historical stock prices and company performance data, suitable for machine learning algorithms to process. Can we actually predict stock prices with machine learning? Investors make educated guesses by analyzing data. They'll read the news, study the company history, industry trends and other lots of data points that go into making a prediction. The prevailing theory is that stock prices are totally random and unpredictable but that raises the question why top firms like Morgan Stanley and Citigroup hire quantitative analysts to build predictive models. We have this idea of a trading floor being filled with adrenaline infused men with loose ties running around yelling something into a phone but these days they're more likely to see rows of machine learning experts quietly sitting in front of computer screens. In fact about 70% of all orders on Wall Street are now placed by software, we're now living in the age of the algorithm. 2 This project seeks to utilize Deep Learning models, Long-Short Term Memory (LSTM) Neural Network algorithm, to predict stock prices. For data with time frames recurrent neural networks (RNNs) come in handy but recent research has shown that LSTM networks are the most popular and useful variants of RNNs. I will use Keras to build a LSTM to predict stock prices using historical closing price and trading volume and visualize both the predicted price values over time and the optimal parameters for the model.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Project Overview

Investment firms, hedge funds and even individuals have been using financial models to better understand market behavior and make profitable investments and trades. A wealth of information is available in the form of historical stock prices and company performance data, suitable for machine learning algorithms to process. Can we actually predict stock prices with machine learning? Investors make educated guesses by analyzing data. They'll read the news, study the company history, industry trends and other lots of data points that go into making a prediction. The prevailing theory is that stock prices are totally random and unpredictable but that raises the question why top firms like Morgan Stanley and Citigroup hire quantitative analysts to build predictive models. We have this idea of a trading floor being filled with adrenaline infused men with loose ties running around yelling something into a phone but these days they're more likely to see rows of machine learning experts quietly sitting in front of computer screens. In fact about 70% of all orders on Wall Street are now placed by software, we're now living in the age of the algorithm. 2 This project seeks to utilize Deep Learning models, Long-Short Term Memory (LSTM) Neural Network algorithm, to predict stock prices. For data with time frames recurrent neural networks (RNNs) come in handy but recent research has shown that LSTM networks are the most popular and useful variants of RNNs. I will use Keras to build a LSTM to predict stock prices using historical closing price and trading volume and visualize both the predicted price values over time and the optimal parameters for the model.

1.2 Project Statement

The challenge of this project is to accurately predict the future closing value of a given stock across a given period of time in the future. For this project I will use a Long Short Term Memory network – usually just called “LSTMs” to predict the closing price of the S&P 500[2] using a dataset of past prices.

1.3 Metrics

For this project measure of performance will be using the Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) calculated as the difference between predicted and actual values of the target stock at adjusted close price and the delta between the performance of the benchmark model (Linear Regression) and our primary model (Deep Learning).

CHAPTER 2

Literature Survey

The goal of this project was to study time-series data and explore as many options as possible to accurately predict the Stock Price. Through my research we came to know about **Recurrent Neural Nets (RNN)** which are used specifically for sequence and pattern learning. As they are networks with loops in them, allowing information to persist and thus ability to memorize the data accurately. But Recurrent Neural Nets have a vanishing Gradient descent problem which does not allow it to learn from past data as was expected. The remedy of this problem was solved in **Long-Short Term Memory Networks**, usually referred to as LSTMs. These are a special kind of RNN, capable of learning long-term dependencies.

2.1 Recurrent Neural Networks

A recurrent neural network (RNN) as shown in figure 1 is a class of artificial neural networks where connections between nodes form a directed or undirected graph along a temporal sequence. This allows it to exhibit temporal dynamic behavior. Derived from feedforward neural networks, RNNs can use their internal state (memory) to process variable length sequences of inputs. This makes them applicable to tasks such as unsegmented, connected handwriting recognition or speech recognition. Recurrent neural networks are theoretically Turing complete and can run arbitrary programs to process arbitrary sequences of inputs.

The term "recurrent neural network" is used to refer to the class of networks with an infinite impulse response, whereas "convolutional neural network" refers to the class of finite impulse response. Both classes of networks exhibit temporal dynamic behavior. A finite impulse recurrent network is a directed acyclic graph that can be unrolled and replaced with a strictly feedforward neural network, while an infinite impulse recurrent network is a directed cyclic graph that can not be unrolled.

Both finite impulse and infinite impulse recurrent networks can have additional stored states, and the storage can be under direct control by the neural network. The storage can also be replaced by another network or graph if that incorporates time delays or has feedback loops. Such controlled states are referred to as gated state or gated memory, and are part of long

short-term memory networks (LSTMs) and gated recurrent units. This is also called Feedback Neural Network (FNN).

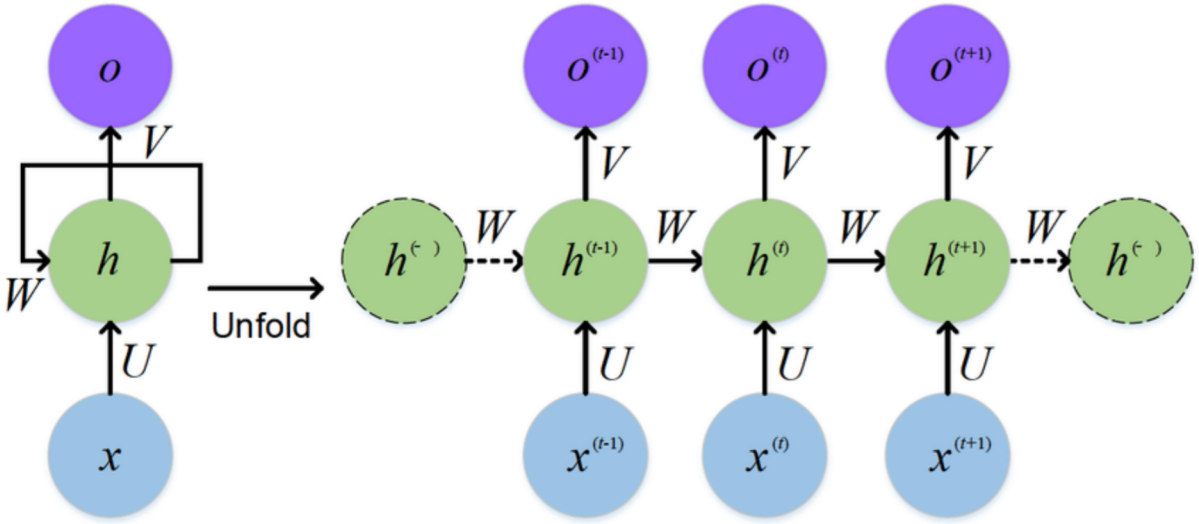


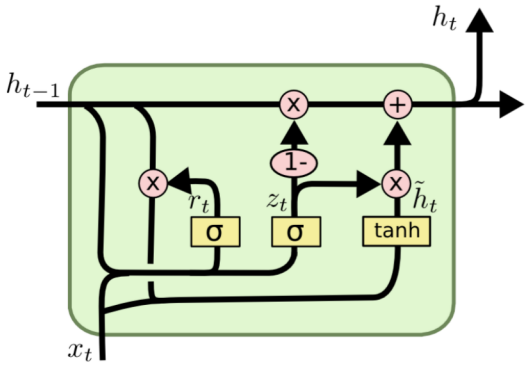
Figure 1 : Standard RNN

2.2 Long Short Term Memory

Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture used in the field of deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections. It can process not only single data points (such as images), but also entire sequences of data (such as speech or video). For example, LSTM is applicable to tasks such as unsegmented, connected handwriting recognition, speech recognition and anomaly detection in network traffic or IDSs (intrusion detection systems).

A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell.

LSTM networks are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series. LSTMs were developed to deal with the vanishing gradient problem that can be encountered when training traditional RNNs. Relative insensitivity to gap length is an advantage of LSTM over RNNs, hidden Markov models and other sequence learning methods in numerous applications.



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Fig 2: Standard LSTM Model

CHAPTER 3

System Design and Methodology

3.1 System Design

The data used in this project is of the famous tech giants MAANG(Meta, Apple, Amazon, Netflix, Google) from January 1,2008 to June 20, 2021, this is a series of data points indexed in time order or a time series. OUR goal was to predict the closing price for any given date after training. For ease of reproducibility and reusability, all data was pulled from the Yahoo Finance Python API.

The prediction has to be made for the Closing (Adjusted closing) price of the data. Since Yahoo Finance already adjusts the closing prices for us, we just need to make a prediction for the "CLOSE" price.

The dataset is of following form :

Table 1: Dataset Demo

Date	Open	High	Low	Close	Volume
30-Jun-17	943.99	945.00	929.61	929.68	2287662
29-Jun-17	951.35	951.66	929.60	937.82	3206674
28-Jun-17	950.66	963.24	936.16	961.01	2745568

The mean, standard deviation, maximum and minimum of the data was found to be following:

Table 2: Dataset Metrics

Feature	Open	High	Low	Close	Volume
Mean	382.5141	385.8720	378.7371	382.3502	4205707.8896
Std	213.4865	214.6022	212.08010	213.4359	3877483.0077
Max	1005.49	1008.61	1008.61	1004.28	41182889
Min	87.74	89.29	86.37	87.58	521141

We can infer from this dataset that date, high and low values are not important features of the data. As it does not matter at what was the highest prices of the stock for a particular day or what was the lowest trading prices. What matters is the opening price of the stock and closing prices of the stock. If at the end of the day we have higher closing prices than the opening prices that we have some profit otherwise we saw losses. Also volume of share is important as a rising market should see rising volume, i.e, increasing price and decreasing volume show lack of interest, and this is a warning of a potential reversal. A price drop (or rise) on large volume is a stronger signal that something in the stock has fundamentally changed.

Therefore we have removed Date, High and low features from the data set at the preprocessing step. The mean, standard deviation, maximum and minimum of the preprocessed data was found to be following:

Table 3: Preprocessed data

	Mean	Std	Max	Min
Open	0.3212	0.23261	1.0	0.0
Close	0.3215	0.2328	1.0	0.0
Volume	0.09061	0.0953	1.0	0.0

3.2 Exploratory Visualizations

To visualize the data we have used the matplotlib library. I have plotted Closing stock price of the data with the no of items(no of days) available.

Following is the snapshot of the plotted data for each Company's Dataset:

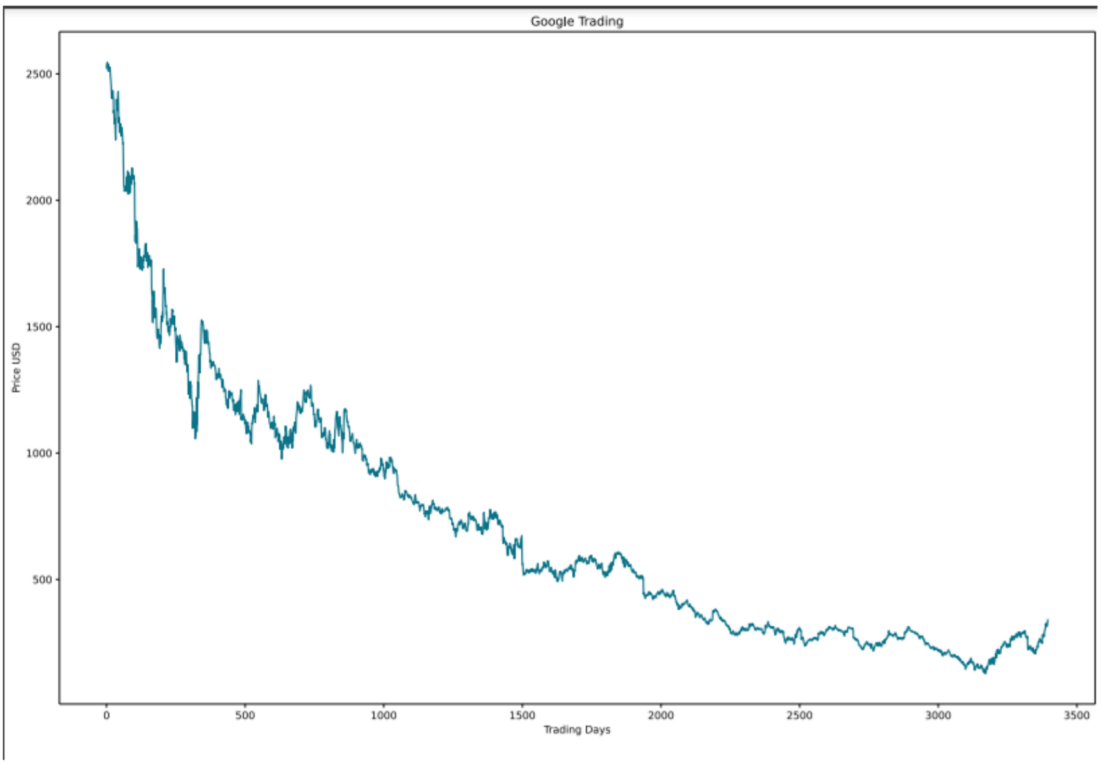


Figure 3 : Google Dataset Visualization

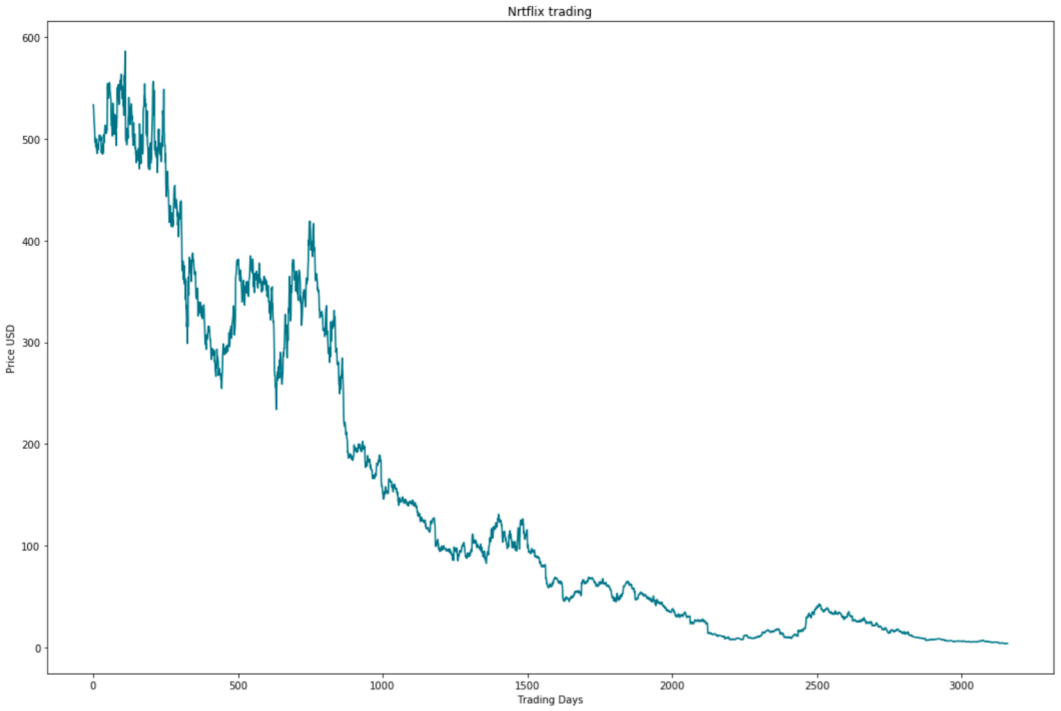


Figure 4: Netflix Dataset Visualization

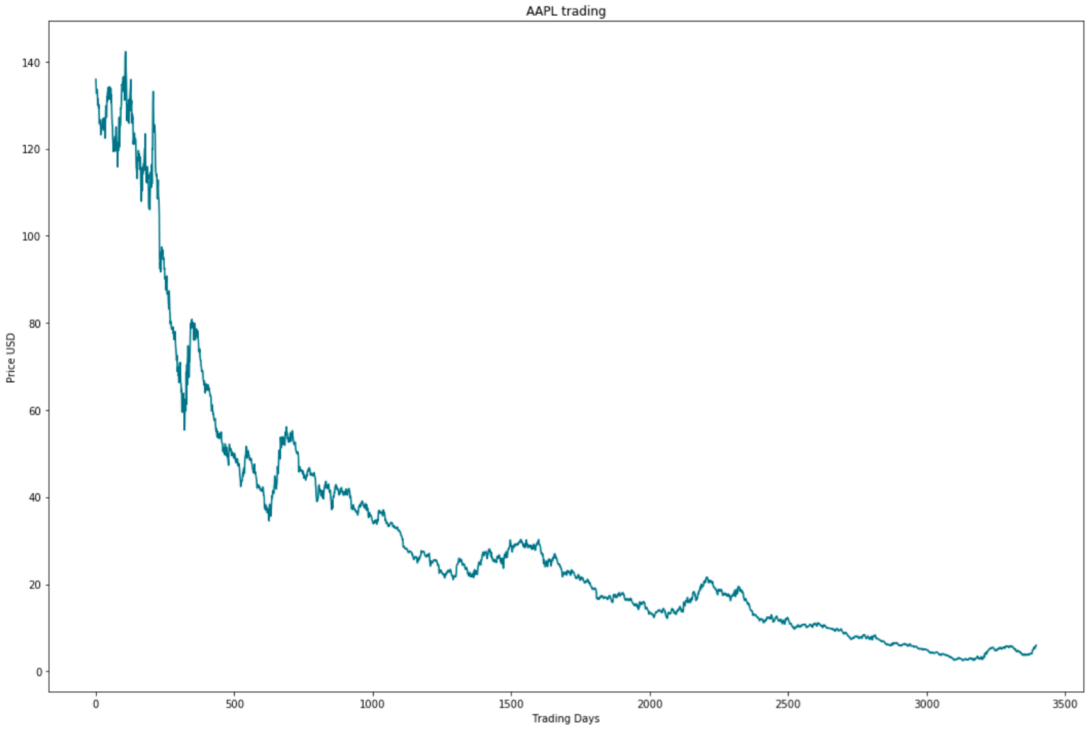


Figure 5: Apple Dataset Visualization

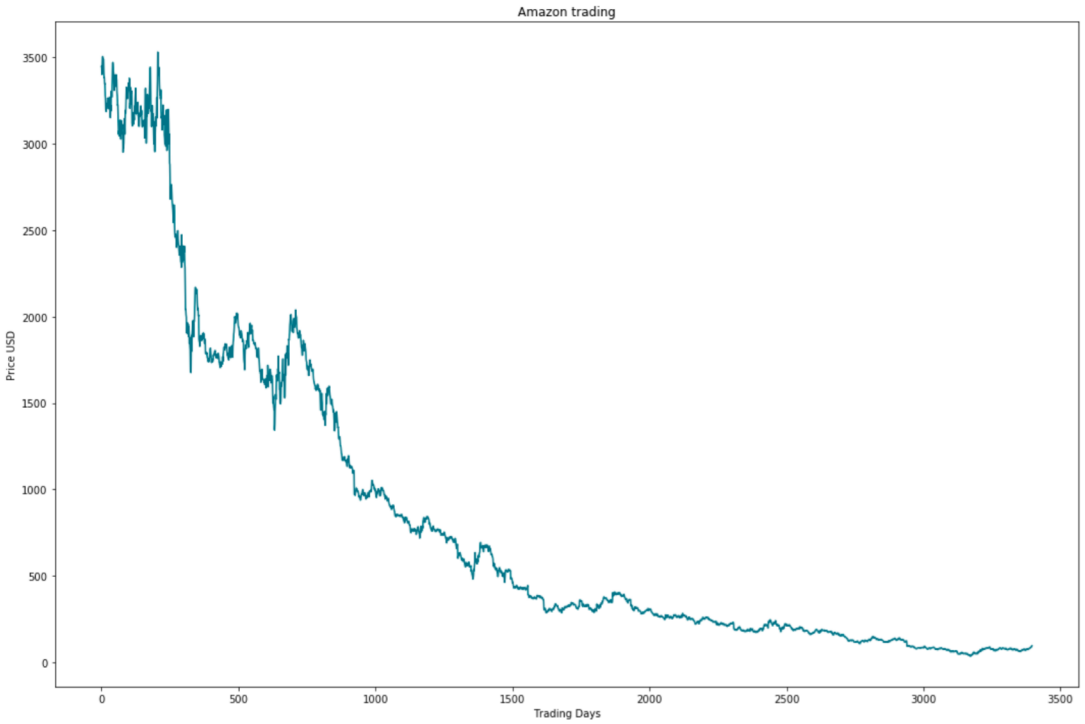


Figure 6 : Amazon Dataset Visualization

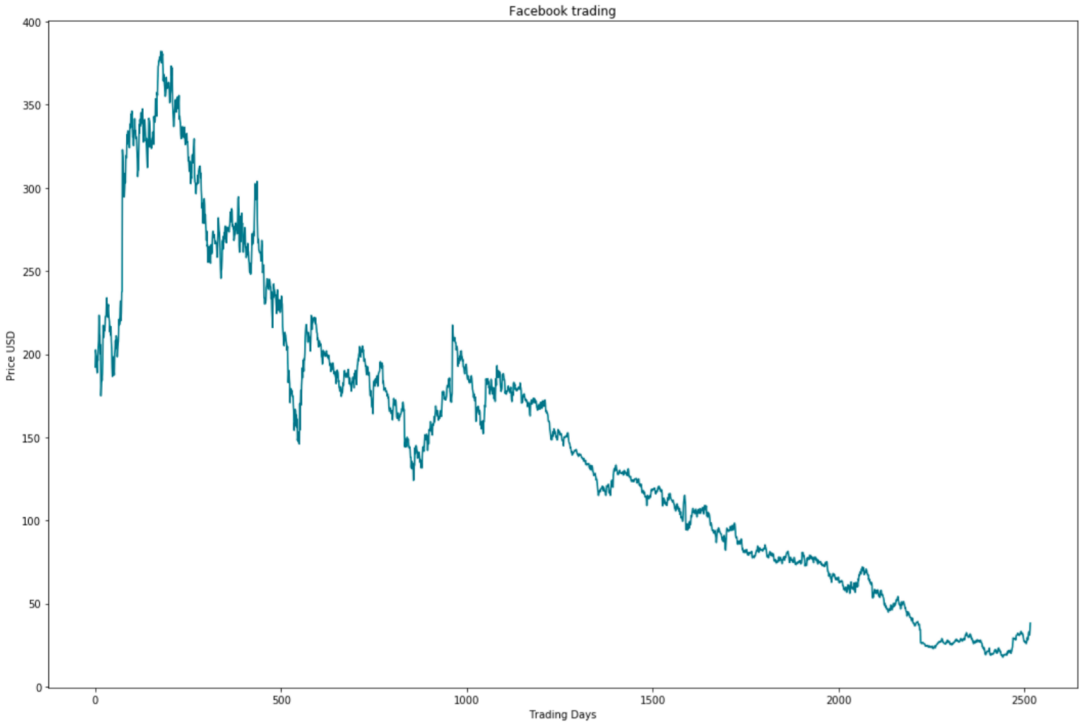


Figure 7 : Facebook Dataset Visualization

CHAPTER 4

Implementation and Results

4.1. Software and Hardware Requirements

Operating system : Windows 10.

Coding Language : PYTHON

System : Intel i5 .

Hard Disk : 40 GB

RAM : 4 GB (minimum)

4.2. Implementation Details

4.2.1 Data Preprocessing

Acquiring and preprocessing the data for this project occurs in following sequence, much of which has been modularized into the **preprocess.py** file for importing and use across all notebooks:

- Request the data from the Yahoo Finance Python API and save it in **google.csv** file in the following format.

Table 4: Featured Dataset

Date	Open	High	Low	Close	Volume
30-Jun-17	943.99	945.00	929.61	929.68	2287662
29-Jun-17	951.35	951.66	929.60	937.82	3206674
28-Jun-17	950.66	963.24	936.16	961.01	2745568

- Remove unimportant features(date, high and low) from the acquired data.

Table 5: Preprocessed Data without Normalization

Item	Open	Close	Volume
0	98.80	101.46	15860692
1	100.77	97.35	13762396
2	96.82	96.85	8239545
3	97.72	94.37	10389803

- Normalized the data using **MinMaxScaler** helper function from Scikit-Learn.

Table 6: Preprocessed Data with Normalization

Item	Open	Close	Volume
0	0.012051	0.015141	0.377248
1	0.014198	0.010658	0.325644
2	0.009894	0.010112	0.189820
3	0.010874	0.007407	0.242701

- Stored the normalized data in **google_preprocessed.csv** file for future reusability.
    -
- Splitted the dataset into the training (68.53%) and test (31.47%) datasets for linear regression model. The split was of following shape :
    - x_train (2155, 1)
    - y_train (2155, 1)
    - x_test (990, 1)
    - y_test (990, 1)
    -
- Splitted the dataset into the training (82.95%) and test (17.05%) datasets for the LSTM model. The Split was of following shape:
    - x_train (2841, 50, 3)
    - y_train (2841,)
    - x_test (446, 50, 3)
    - y_test (446,)

4.2.2 Implementation

Once the data has been downloaded and preprocessed, the implementation process occurs consistently through all three models as follow:

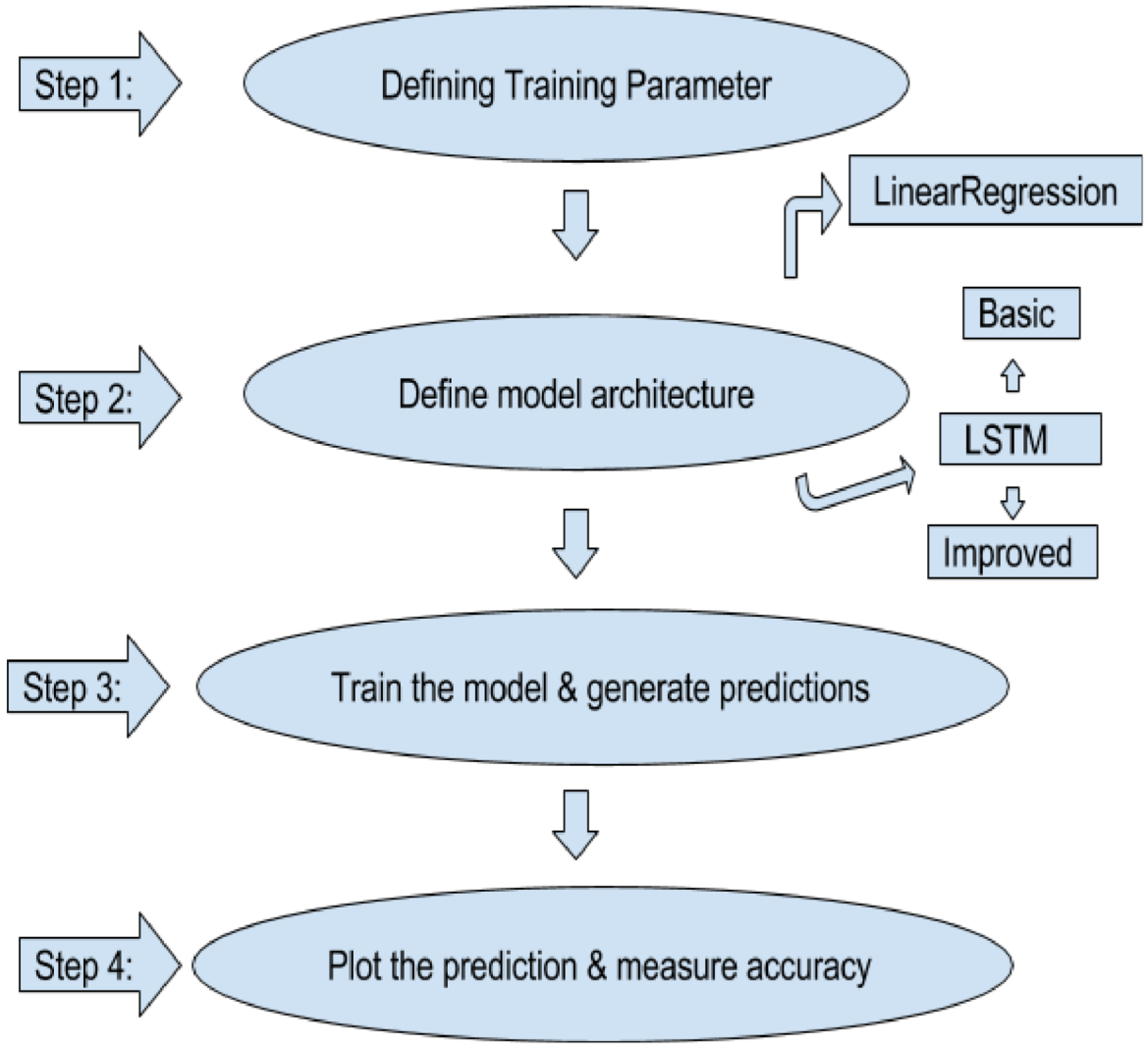


Figure 8 : Implementation Process

We have thoroughly specified all the steps to build, train and test the model and its predictions in the notebook itself.

Some code implementation insight for Different Models is shown ahead.

Benchmark model :

Step 1 : Split into train and test model :

```
X_train, X_test, y_train, y_test, label_range= sd.train_test_split_linear_regression(stocks)
```

Here We are calling a function defined in **'stock_data.py'** which splits the data for a linear regression model. The function is as follows :

```
def train_test_split_linear_regression(stocks):
    """
        Split the data set into training and testing feature for Linear Regression Model
        :param stocks: whole data set containing ['Open','Close','Volume'] features
        :return: X_train : training sets of feature
        :return: X_test : test sets of feature
        :return: y_train: training sets of label
        :return: y_test: test sets of label
        :return: label_range: scaled range of label used in predicting price,
    """
    # Create numpy arrays for features and targets
    feature = []
    label = []

    # Convert dataframe columns to numpy arrays for scikit learn
    for index, row in stocks.iterrows():
        # print([np.array(row['Item'])])
        feature.append([(row['Item'])])
        label.append([(row['Close'])])

    # Regularize the feature and target arrays and store min/max of input data for rescaling later
    feature_bounds = [min(feature), max(feature)]
    feature_bounds = [feature_bounds[0][0], feature_bounds[1][0]]
    label_bounds = [min(label), max(label)]
    label_bounds = [label_bounds[0][0], label_bounds[1][0]]

    feature_scaled, feature_range = scale_range(np.array(feature), input_range=feature_bounds, target_range=[-1.0, 1.0])
    label_scaled, label_range = scale_range(np.array(label), input_range=label_bounds, target_range=[-1.0, 1.0])

    # Define Test/Train Split 80/20
    split = .315
    split = int(math.floor(len(stocks['Item']) * split))

    # Set up training and test sets
    X_train = feature_scaled[:-split]
    X_test = feature_scaled[-split:]

    y_train = label_scaled[:-split]
    y_test = label_scaled[-split:]

    return X_train, X_test, y_train, y_test, label_range
```

Step 2: In this step model is built using **scikit-learn linear_model**[10] library.

```
model = LinearRegressionModel.build_model(X_train,y_train)
```

Here we are calling a function defined in **'LinearRegressionModel.py**' which builds the model for the project. The screenshot of the function is as follows:

```python
import numpy as np
import stock_data as sd


def build_model(X, y):
    """
    build a linear regression model using sklearn.linear_model
    :param X: Feature dataset
    :param y: label dataset
    :return: a linear regression model
    """
    linear_mod = linear_model.LinearRegression()  # defining the linear regression model
    X = np.reshape(X, (X.shape[0], 1))
    y = np.reshape(y, (y.shape[0], 1))
    linear_mod.fit(X, y)  # fitting the data points in the model

    return linear_mod
```

Step 3: Now it's time to predict the prices for given test datasets.

```python
predictions = LinearRegressionModel.predict_prices(model,X_test, label_range)
```

```python
def predict_prices(model, x, label_range):
    """
    Predict the label for given test sets
    :param model: a linear regression model
    :param x: testing features
    :param label_range: normalised range of label data
    :return: predicted labels for given features
    """
    x = np.reshape(x, (x.shape[0], 1))
    predicted_price = model.predict(x)
    predictions_rescaled, re_range = sd.scale_range(predicted_price, input_range=[-1.0, 1.0], target_range=label_range)

    return predictions_rescaled.flatten()
```

Step 4: Finally calculate the test score and plot the results of the benchmark model.

```python
trainScore = mean_squared_error(X_train, y_train)
print('Train Score: %.4f MSE (%.4f RMSE)' % (trainScore, math.sqrt(trainScore)

testScore = mean_squared_error(predictions, y_test)
print('Test Score: %.8f MSE (%.8f RMSE)' % (testScore, math.sqrt(testScore)))
```

✓ 0.8s

```
Train Score: 0.5961 MSE (0.7721 RMSE)
Test Score: 0.58359552 MSE (0.76393424 RMSE)
```

Improved LSTM model :

Step 1 : Split into train and test model :

Note : The same set of training and testing data is used for improved LSTM as is used with basic LSTM.

Step 2 : Build an improved LSTM model :

```
# Set up hyperparameters
batch_size = 512
epochs = 20

# build improved lstm model
model = lstm.build_improved_model( X_train.shape[-1],output_dim = unroll_length, return_sequences=True)
```

Here we are calling a function defined in **'lstm.py**' which builds the improved lstm model for the project. The screenshot of the function is as follows:

```
5
6   def build_improved_model(input_dim, output_dim, return_sequences):
7       """
8       Builds an improved Long Short term memory model using keras.layers.recurrent.lstm
9       :param input_dim: input dimension of model
10      :param output_dim: ouput dimension of model
11      :param return_sequences: return sequence for the model
12      :return: a 3 layered LSTM model
13      """
14      model = Sequential()
15      model.add(LSTM(
16          input_shape=(None, input_dim),
17          units=output_dim,
18          return_sequences=return_sequences))
19
20      model.add(Dropout(0.2))
21
22      model.add(LSTM(
23          144,
24          return_sequences=False))
25
26      model.add(Dropout(0.2))
27
28      model.add(Dense(
29          units=1))
30      model.add(Activation('linear'))
31
32      return model
33
```

We have increased the batch_size to 512 from 1 Epochs from 1 to 20 for our improved LSTM model library to implement LSTM Also in the function i have add increased the no of nodes in hidden layer to 144 from 100 and have added a drop out of 0.2 to all the

layers.

Step 3: We now need to train our model.

```
model.fit(X_train,
          y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=2,
          validation_split=0.05
          )
```

[126] ✓ 1m 52.7s

```
Output exceeds the size limit. Open the full output data in a text editor
Epoch 1/20
27/27 - 10s - loss: 0.0134 - val_loss: 2.5061e-04 - 10s/epoch - 367ms/step
Epoch 2/20
27/27 - 4s - loss: 0.0011 - val_loss: 1.0426e-04 - 4s/epoch - 164ms/step
Epoch 3/20
27/27 - 5s - loss: 8.1492e-04 - val_loss: 9.9920e-05 - 5s/epoch - 172ms/step
Epoch 4/20
27/27 - 4s - loss: 8.4560e-04 - val_loss: 8.5689e-05 - 4s/epoch - 151ms/step
Epoch 5/20
27/27 - 4s - loss: 8.5117e-04 - val_loss: 1.1919e-04 - 4s/epoch - 158ms/step
Epoch 6/20
27/27 - 4s - loss: 7.4393e-04 - val_loss: 5.0944e-05 - 4s/epoch - 151ms/step
Epoch 7/20
27/27 - 5s - loss: 7.3183e-04 - val_loss: 4.8081e-05 - 5s/epoch - 173ms/step
Epoch 8/20
27/27 - 4s - loss: 7.2166e-04 - val_loss: 1.0194e-04 - 4s/epoch - 161ms/step
Epoch 9/20
27/27 - 5s - loss: 6.4888e-04 - val_loss: 8.0082e-05 - 5s/epoch - 192ms/step
Epoch 10/20
27/27 - 6s - loss: 6.6330e-04 - val_loss: 7.6053e-05 - 6s/epoch - 240ms/step
Epoch 11/20
27/27 - 6s - loss: 7.0502e-04 - val_loss: 6.3308e-05 - 6s/epoch - 228ms/step
Epoch 12/20
27/27 - 6s - loss: 7.4710e-04 - val_loss: 3.3469e-05 - 6s/epoch - 240ms/step
Epoch 13/20
...
27/27 - 6s - loss: 6.0187e-04 - val_loss: 4.1638e-05 - 6s/epoch - 211ms/step
Epoch 19/20
27/27 - 6s - loss: 6.0920e-04 - val_loss: 7.1838e-05 - 6s/epoch - 204ms/step
Epoch 20/20
27/27 - 6s - loss: 5.8044e-04 - val_loss: 3.1831e-05 - 6s/epoch - 229ms/step
<keras.callbacks.History at 0x19b81def670>
```

We have used a built-in library function to train the model.

Step 4: Now it's time to predict the prices for given test datasets.

```
# Generate predictions
predictions = model.predict(X_test, batch_size=batch_size)
```

Step 5: Finally calculate the test score and plot the results of the improved LSTM model.

Step 5: Get the test score

```
trainScore = model.evaluate(X_train, y_train, verbose=0)
print('Train Score: %.8f MSE (%.8f RMSE)' % (trainScore, math.sqrt(trainScore)))

testScore = model.evaluate(X_test, y_test, verbose=0)
print('Test Score: %.8f MSE (%.8f RMSE)' % (testScore, math.sqrt(testScore)))
```

[19] ✓ 2.2s

```
Train Score: 0.00026132 MSE (0.01616543 RMSE)
Test Score: 0.00003914 MSE (0.00625607 RMSE)
```

4.2.3 Refinement

For this project we have worked on fine tuning parameters of LSTM to get better predictions. We did the improvement by testing and analyzing each parameter and then selecting the final value for each of them.

To improve LSTM we have done following:

- Increased the number of hidden nodes from 100 to 144.
- Added Dropout of 0.2 at each layer of LSTM
- Increased batch size from 1 to 512
- Increased epochs from 1 to 20
- Added verbose = 2
- Made prediction with the batch size

4.2.4 Results

With each model we have refined and fine tuned my predictions and have reduced mean squared error significantly.

We Used dataset for 5 companies but for sake of simplicity lets first just go through the results we recorded for Google dataset.

- For our first model using linear regression model:
    - Train Score: 0.5961 MSE (0.7721 RMSE)
    - Test Score: 0.58359552 MSE (0.76393424 RMSE

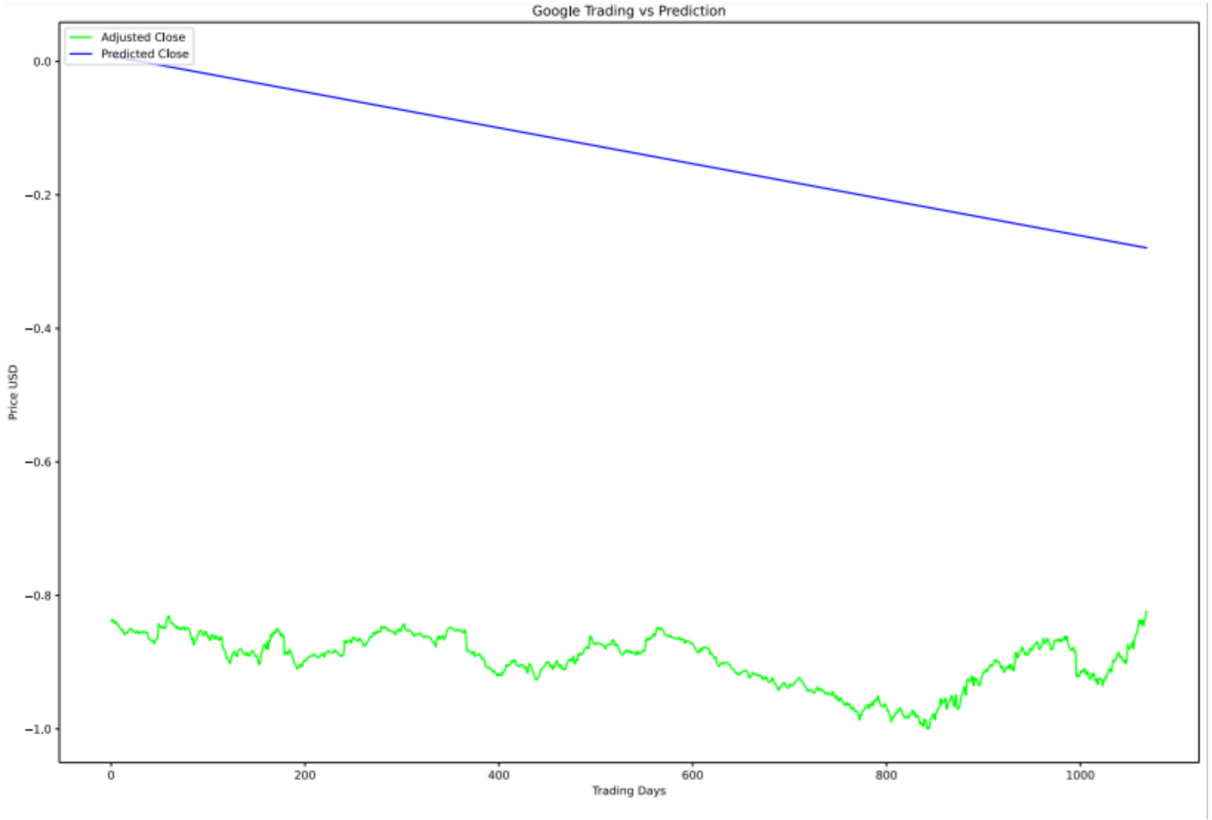


Figure 9: Plot of Linear Regression Model

- For our second model using basic Long-Short Term memory model:
    - Train Score: 0.00029000 MSE (0.01702941 RMSE)
    - Test Score: 0.00030443 MSE (0.01744785 RMSE)

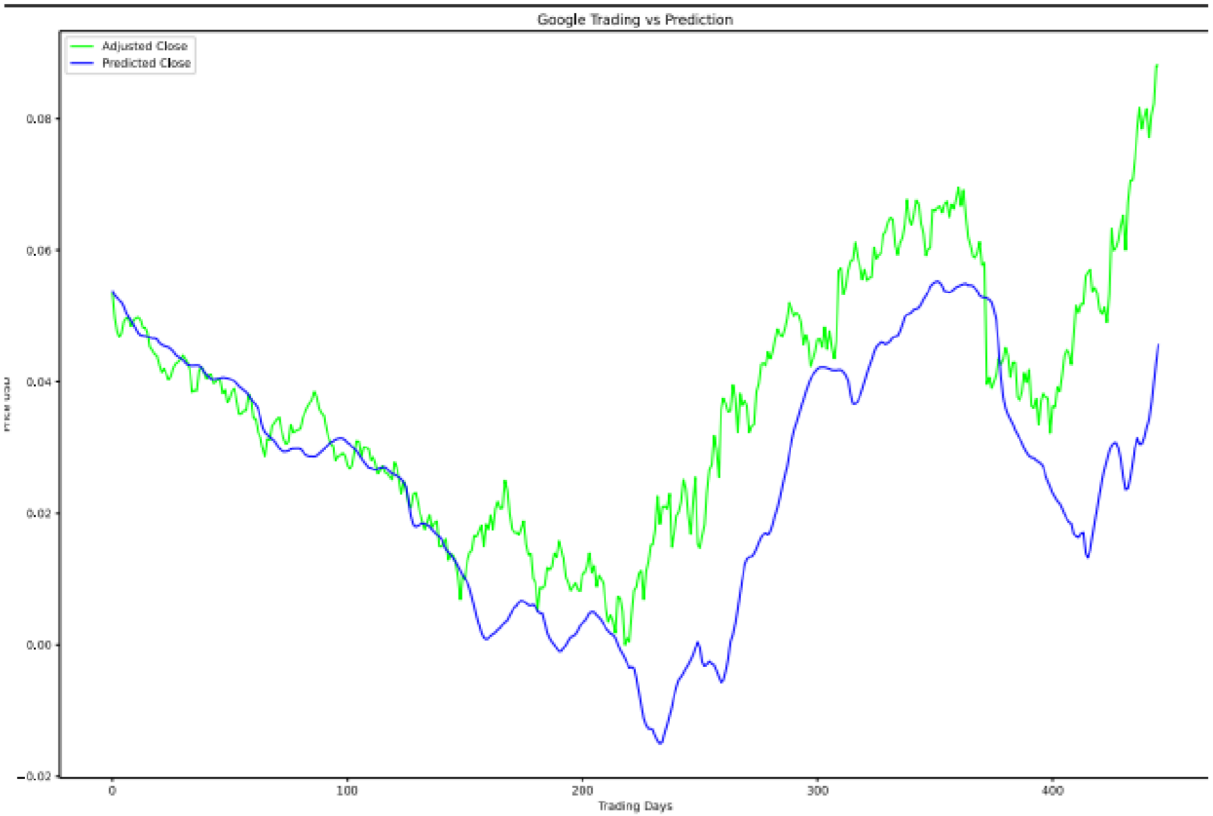


Figure 10: Plot of basic Long-Short Term Memory model

- For our third and final model, using improved Long-Short Term memory model:
    - Train Score: 0.00026132 MSE (0.01616543 RMSE)
    - Test Score: 0.00003914 MSE (0.00625607 RMSE)

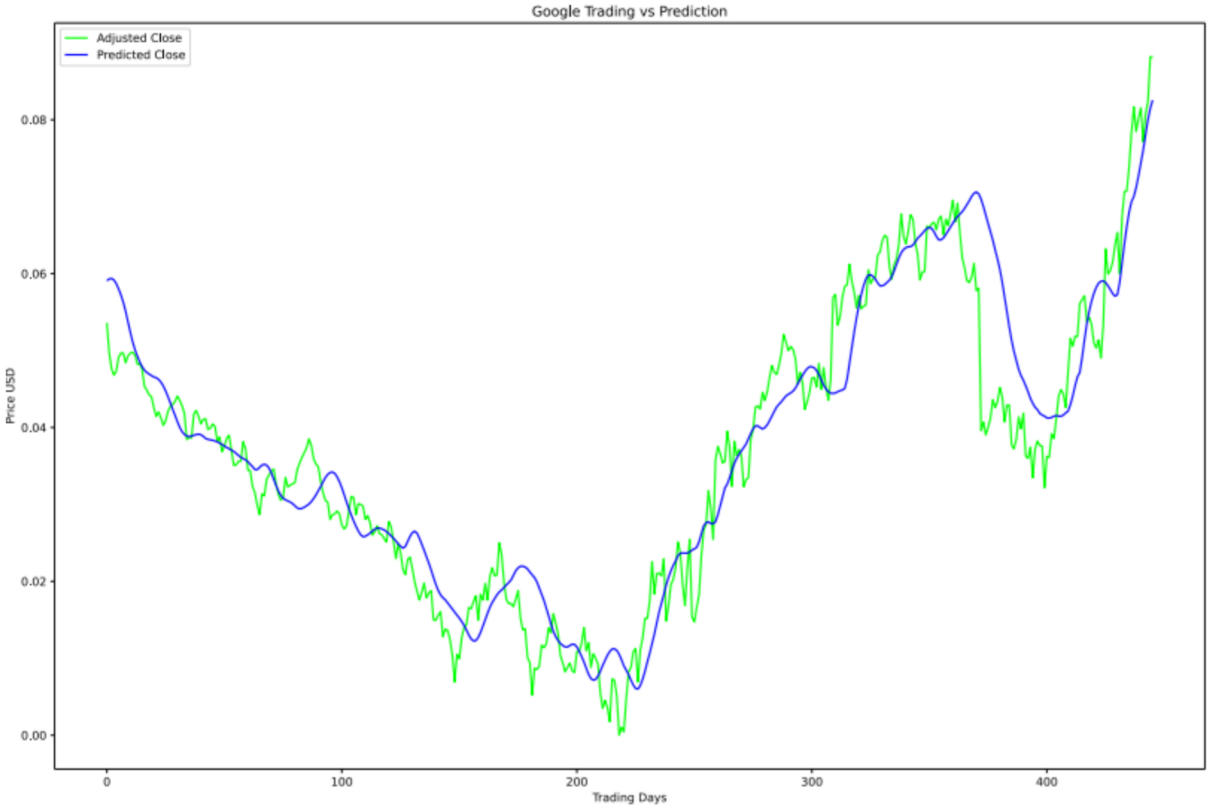


Figure 11: Plot of Improved Long-Short Term Memory model

Similarly for other datasets as we saw a similar trend , with change in model the prediction rate improved.

The final Graph representing prediction for other 4 companies using improved LSTM model:

Netflix:

- Train Score: 0.00078588 MSE (0.02803348 RMSE)
- Test Score: 0.00003464 MSE (0.00588518 RMSE)

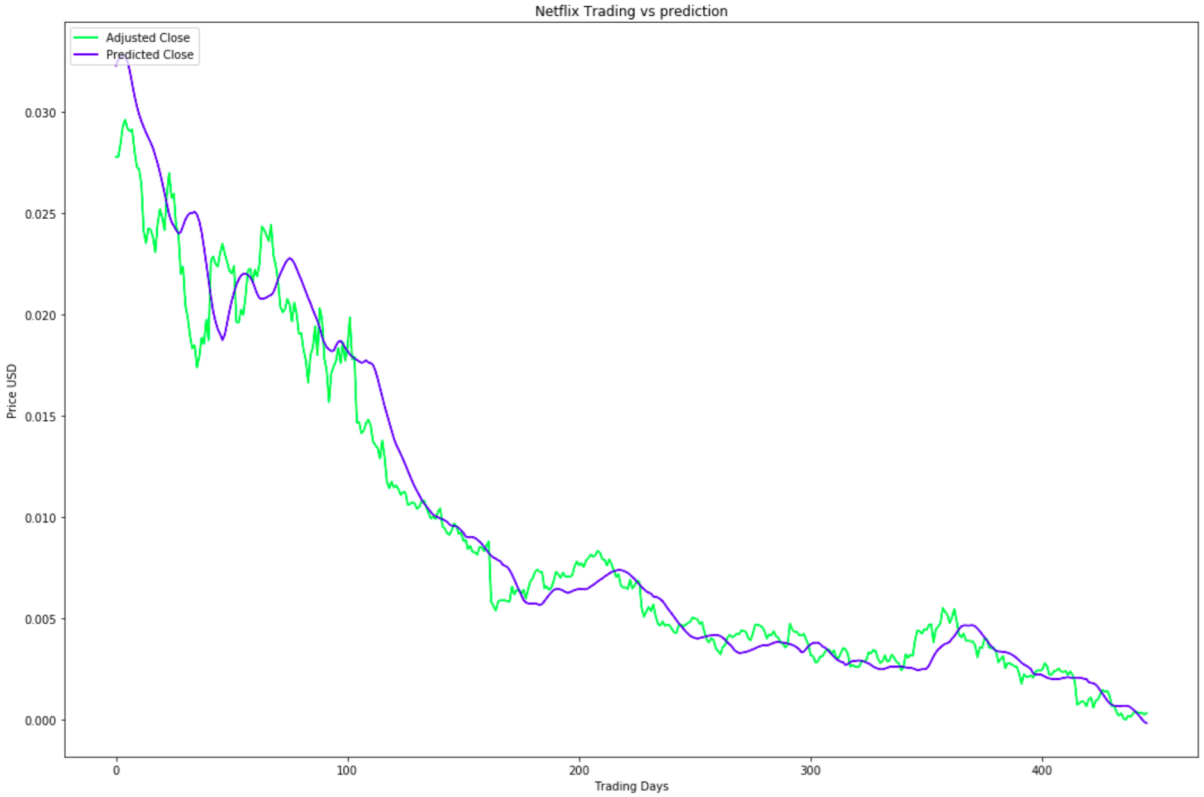


Figure 12: Plot of Improved Long-Short Term Memory model for Netflix

Facebook:

- Train Score: 0.00109912 MSE (0.03315295 RMSE)
- Test Score: 0.00021407 MSE (0.01463117 RMSE)

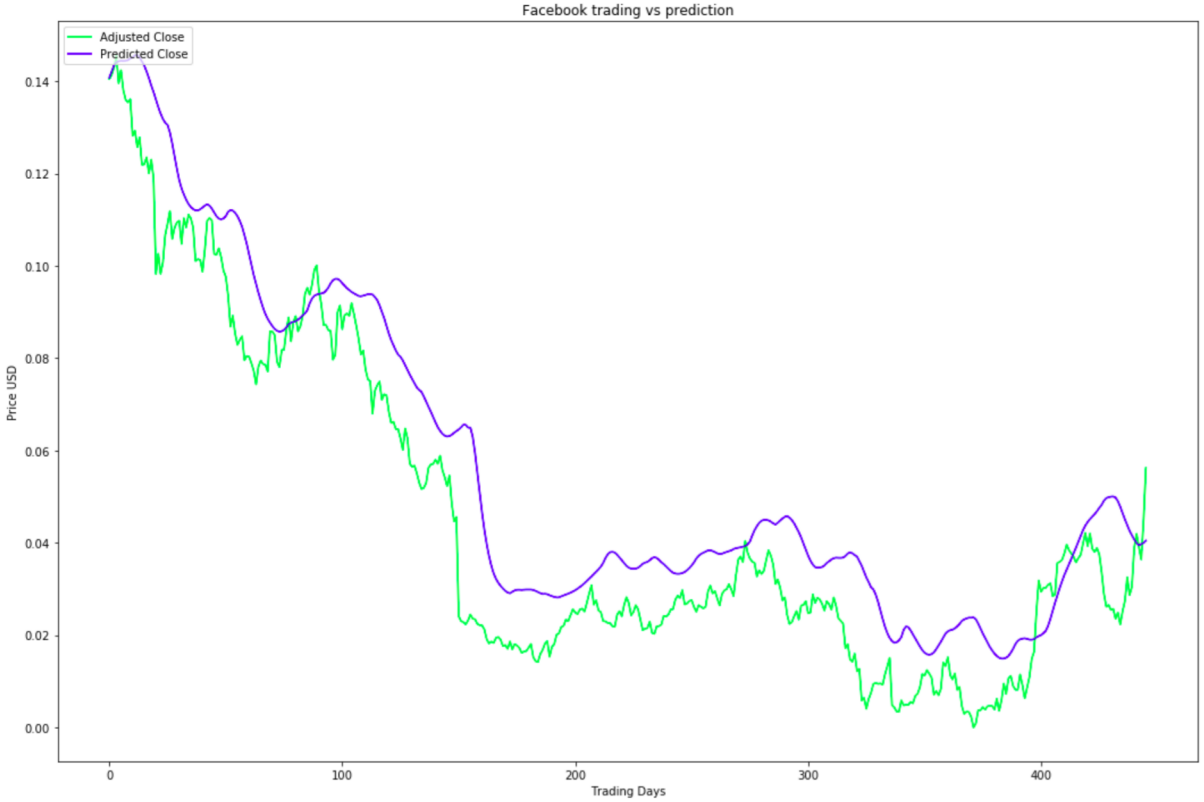


Figure 13: Plot of Improved Long-Short Term Memory model for Facebook

Amazon:

- Train Score: 0.00040049 MSE (0.2001224 RMSE)
- Test Score: 0.00001860 MSE (0.00431297 RMSE)

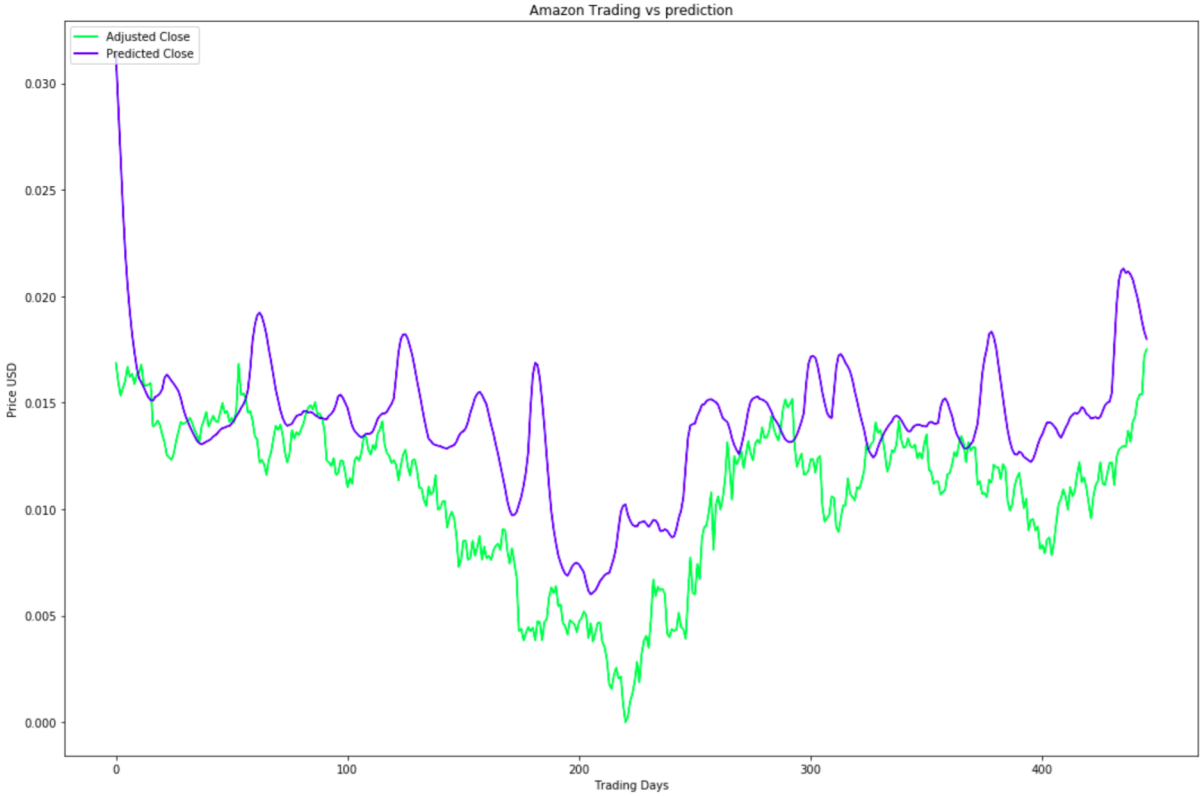


Figure 14: Plot of Improved Long-Short Term Memory model for Amazon

CHAPTER 5

Conclusion

5.1. Performance Evaluation

We have already discussed all the important features of the datasets and their visualization in one of the above sections. But to conclude my report we would choose my final model visualization, which is an improved version of LSTM by fine tuning parameters. As we were very impressed on seeing how close we have gotten to the actual data, with a mean square error of just 0.0009. It was an 'Aha!' moment for me as we had to poke around a lot . But it was fun working on this project.

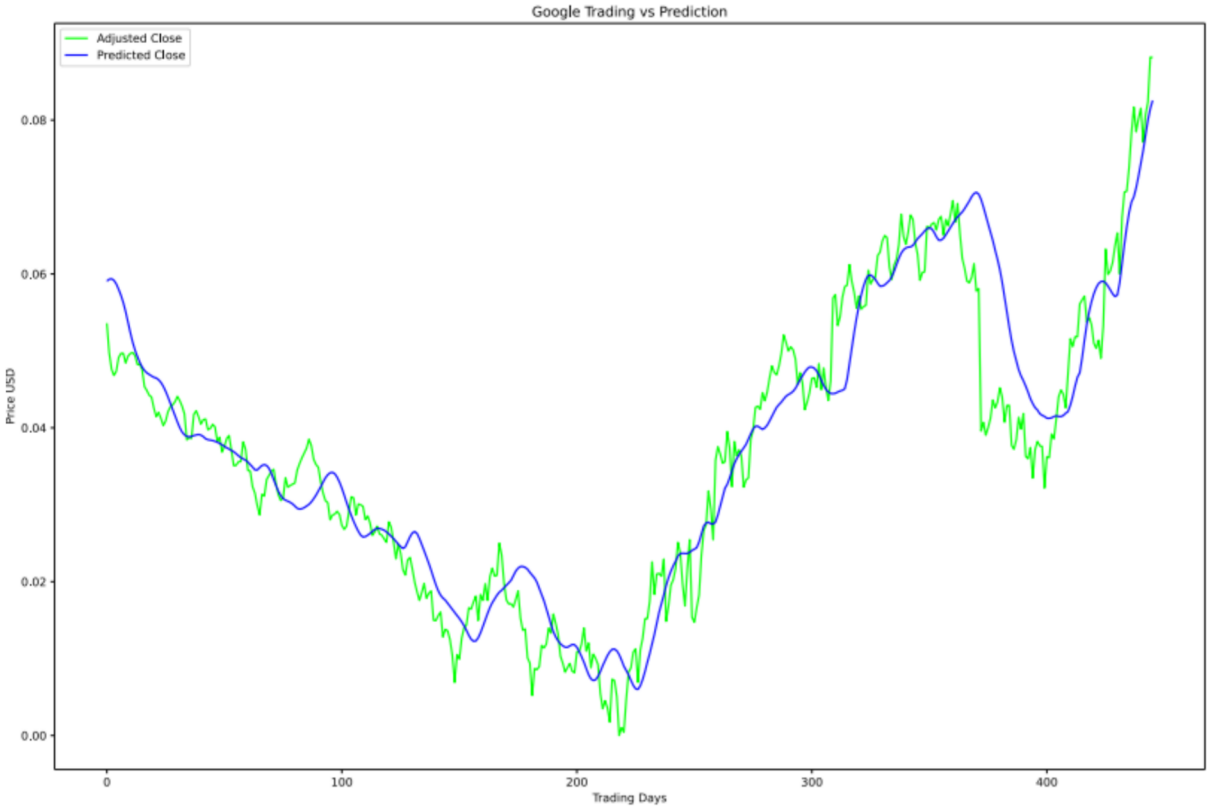


Figure16: Plot of Improved Long-Short Term Memory Model

5.2. Reflection

To recap, the process undertaken in this project:

- Set Up Infrastructure
  - iPython Notebook
  - Incorporate required Libraries (Keras, Tensor flow, Pandas, Matplotlib,

Sklearn, Numpy)
    - Git project organization
- Prepare Dataset
    - Incorporate data of Alphabet Inc company
    - Process the requested data into Pandas Dataframe
    - Develop function for normalizing data
    - Dataset used with a 80/20 split on training and test data across all models
- Develop Benchmark Model
    - Set up basic Linear Regression model with Scikit-Learn
    - Calibrate parameters
- Develop Basic LSTM Model
    - Set up basic LSTM model with Keras utilizing parameters from Benchmark Model
- Improve LSTM Model
    - Develop, document, and compare results using additional labels for the LSMT model 5. Document and Visualize Results
- Plot Actual, Benchmark Predicted Values, and LSTM Predicted Values per time series
- Analyze and describe results for the report.

We started this project with the hope to learn a completely new algorithm, i.e, Long-Short Term Memory and also to explore real time series data sets. The final model really exceeded my expectations and has worked remarkably well. I am greatly satisfied with these results.

5.3. Future Directions

Before starting this project we had little prior experience in python. In the beginning of this course to do everything with python, we had to google it. But now we have not only made 7 projects in python, we have explored many libraries along the way and can use them very comfortably.

And as there is scope of improvement in each individual, so is the case with this project. This project predicts closing prices with very minimum Mean Squared Error, still there are many things that are lagging in this project. Two of most important things are :

- There is no user interaction or interface provided in this project. A UI can be provided where users can check the

value for future dates.

- The stocks used for this project are only of Alphabet Inc, we can surely add more S&P 500 in the list so as to make this project more comprehensive.

We would definitely like to add these improvements to this project in future.

Appendix 1

Published paper

STOCK MARKET PREDICTION USING MACHINE LEARNING

Abstract— Investment firms, hedge funds and even individuals have been using financial models to better understand market behavior and make profitable investments and trades. A wealth of information is available in the form of historical stock prices and company performance data, suitable for machine learning algorithms to process. Can we actually predict stock prices with machine learning? Investors make educated guesses by analyzing data. They'll read the news, study the company history, industry trends and other lots of data points that go into making a prediction. The prevailing theory is that stock prices are totally random and unpredictable but that raises the question why top firms like Morgan Stanley and Citigroup hire quantitative analysts to build predictive models. We have this idea of a trading floor being filled with adrenaline infused men with loose ties running around yelling something into a phone but these days they're more likely to see rows of machine learning experts quietly sitting in front of computer screens. In fact about 70% of all orders on Wall Street are now placed by software, we're now living in the age of the algorithm. 2 This project seeks to utilize Deep Learning models, Long-Short Term Memory (LSTM) Neural Network algorithm, to predict stock prices. For data with time frames recurrent neural networks (RNNs) come in handy but recent research has shown that LSTM networks are the most popular and useful variants of RNNs. I will use Keras to build a LSTM to predict stock prices using historical closing price and trading volume and visualize both the predicted price values over time and the optimal parameters for the model.

Keywords— **Supervised learning, Machine Learning,Neural Networks, *Long Short Term Memory Model***

I. INTRODUCTION

The challenge of this project is to accurately predict the future closing value of a given

stock across a given period of time in the future. For this project I will use a Long Short Term Memory network – usually just called “LSTMs” to predict the closing price of the S&P 500 using a dataset of past prices.For this project measure of performance will be using the Mean Squared Error(MSE) and Root Mean Squared Error (RMSE) calculated as the difference between predicted and actual values of the target stock at adjusted close price and the delta between the performance of the benchmark model (Linear Regression) and our primary model (Deep Learning).

II. EXPERIMENT AND RESULT

For this project we have worked on fine tuning parameters of LSTM to get better predictions. We did the improvement by testing and analyzing each parameter and then selecting the final value for each of them.

To improve LSTM we have done following:

- Increased the number of hidden nodes from 100 to 144.
- Added Dropout of 0.2 at each layer of LSTM
- Increased batch size from 1 to 512
- Increased epochs from 1 to 20
- Added verbose = 2
- Made prediction with the batch size

With each model we have refined and fine tuned my predictions and have reduced mean squared error significantly.

We Used dataset for 5 companies but for sake of simplicity lets first just go through the results we recorded for Google dataset.

- For our first model using linear regression model:
  - Train Score: 0.5961 MSE (0.7721 RMSE)
  - Test Score: 0.58359552 MSE (0.76393424 RMSE

- For our second model using basic Long-Short Term memory model:
  - Train Score: 0.00029000 MSE (0.01702941 RMSE)
  - Test Score: 0.00030443 MSE (0.01744785 RMSE)
- For our third and final model, using improved Long-Short Term memory model:
  - Train Score: 0.00026132 MSE (0.01616543 RMSE)
  - Test Score: 0.00003914 MSE (0.00625607 RMSE)

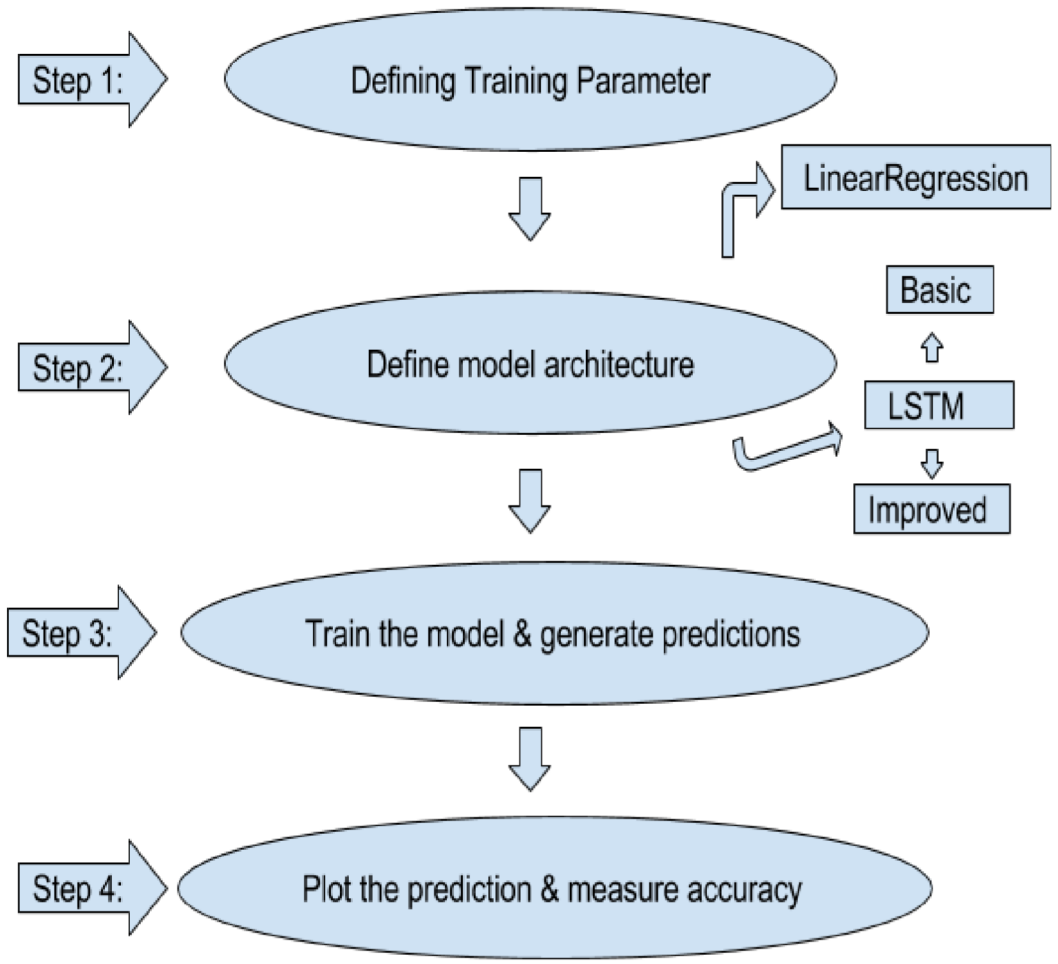


Figure .1. Flowchart

III. CONCLUSION

But to conclude my report we would choose my final model visualization, which is an improved version of LSTM by fine tuning parameters. As we were very impressed on seeing how close we have gotten to the actual data, with a mean

square error of just 0.0009. It was an ‘Aha!’ moment for me as we had to poke around a lot . But it was fun working on this project.

ACKNOWLEDGEMENT

Our sincere thanks to guide Dr. Sonali Mathur, Head of Department, Computer Science Department, IMS Engineering College, Ghaziabad, India for her patient guidance and productive suggestions for the research in the area of stock market prediction using machine learning. This research is made possible due to her willingness to devout time graciously.

IV. REFERENCES

1. Hu Z, Zhu J, Tse K. ,2013 “Stocks market prediction using support vector machines.” , pp. 115-118. IEEE.
2. Usmani M, Adil SH, Raza K, Ali SS. ,2016 “Stock market prediction using machine learning techniques” ,pp.322-327. IEEE.
3. Mankar T, Hotchandani T, Madhwani M, Chidrawar A, Lifna CS,2018,”Stock market prediction based on social sentiments using machine learning.” ,pp.1-3.IEEE.
4. Ouahilal M, El Mohajir M, Chahhou M, El Mohajir BE.,2016,” Optimizing stock market price prediction using a hybrid approach based on HP filter and support vector regression.” ,pp. 290-294. IEEE.
5. Huang H, Zhang W, Deng G, Chen J. ,2014 “Predicting stock trend using fourier transform and support vector regression.” , pp. 213-216. IEEE.
6. Sharma N, Juneja A.,2017 ,”Combining of random forest estimates using LSboost for stock market index prediction.” , pp.1199-1202. IEEE.
7. Tiwari S, Bharadwaj A, Gupta S. ,2017 ,”Stock price prediction using data analytics.”, pp. 1-5.IEEE.
8. Pun TB, Shahi TB. , 2018, “Nepal stock exchange prediction using support vector regression and neural net-works.”, pp. 1-6. IEEE.
9. Ramesh LS, Ganapathy S, Bhuvaneshwari R, Kulothungan K, Pandiyaraju V, Kannan A. ,2015, “Prediction of user interests for providing relevant information using relevance feedback and re-ranking.” International Journal of Intelligent Information Technologies (IJIIT). 2015 Oct 1;11(4):55-71.
10. Selvakumar K, Ramesh LS, Kannan A. Enhanced K-means clustering algorithm for evolving user groups. Indian Journal of Science and Technology. 2015 Sep 1;8(24):1-8.
11. Sabena S, Kalaiselvi S, Anusha B, Ramesh LS. An Multi-Label Classification with Label Correlation. Asian Journal of Research in Social Sciences and Humanities. 2016;6(9):373-86.
12. Aravind G, Uma GV, SaiRamesh L. Knowledge Management Process Model using IHS based on Detecting Process with Less Energy Consumption. Journal of Electrical Engineering. 2020;20(1):1-7.

13. SaiRamesh L, Ashok E, Sabena S, Ayyasamy A. Credit Card Fraud Detectıon in Retaıl Shopping Using Reinforcement Learning. In International Conference On Computational Vision and Bio Inspired Computing 2018 Nov 29 (pp. 1541-1549). Springer, Cham.
14. Xu X. Machine learning-based prediction of urban soil environment and corpus translation teaching.Arabian Journal of Geosciences. 2021 Jun;14(11):1-5.
15. Wang L. Urban land ecological evaluation and English translation model optimization based on machine learning. Arabian Journal of Geosciences. 2021 Jun;14(11):1-6.

Appendix 2

Source Code:

1. **Stock.py**

```
import numpy as np

import math

def scale_range(x, input_range, target_range):

    """

    Rescale a numpy array from input to target range

    :param x: data to scale

    :param input_range: optional input range for data: default 0.0:1.0

    :param target_range: optional target range for data: default 0.0:1.0

    :return: rescaled array, incoming range [min,max]

    """

    range = [np.amin(x), np.amax(x)]

    x_std = (x - input_range[0]) / (1.0*(input_range[1] - input_range[0]))

    x_scaled = x_std * (1.0*(target_range[1] - target_range[0])) + target_range[0]

    return x_scaled, range

def train_test_split_linear_regression(stocks):

    """

        Split the data set into training and testing feature for Linear Regression Model

        :param stocks: whole data set containing ['Open','Close','Volume'] features

        :return: X_train : training sets of feature

        :return: X_test : test sets of feature
```

```
        :return: y_train: training sets of label
        :return: y_test: test sets of label
        :return: label_range: scaled range of label used in predicting price,
    """
    # Create numpy arrays for features and targets
    feature = []
    label = []
    # Convert dataframe columns to numpy arrays for scikit learn
    for index, row in stocks.iterrows():
        # print([np.array(row['Item'])])
        feature.append([(row['Item'])])
        label.append([(row['Close'])])
    # Regularize the feature and target arrays and store min/max of input data for rescaling later
    feature_bounds = [min(feature), max(feature)]
    feature_bounds = [feature_bounds[0][0], feature_bounds[1][0]]
    label_bounds = [min(label), max(label)]
    label_bounds = [label_bounds[0][0], label_bounds[1][0]]
    feature_scaled, feature_range = scale_range(np.array(feature),
input_range=feature_bounds, target_range=[-1.0, 1.0])
    label_scaled, label_range = scale_range(np.array(label),
input_range=label_bounds, target_range=[-1.0, 1.0])
    # Define Test/Train Split 80/20
```

```
    split = .315
    split = int(math.floor(len(stocks['Item']) * split))
    # Set up training and test sets
    X_train = feature_scaled[:-split]
    X_test = feature_scaled[-split:]
    y_train = label_scaled[:-split]
    y_test = label_scaled[-split:]
    return X_train, X_test, y_train, y_test, label_range
def train_test_split_lstm(stocks, prediction_time=1, test_data_size=450,
unroll_length=50):
    """
        Split the data set into training and testing feature for Long Short
            Term Memory Model
        :param stocks: whole data set containing ['Open','Close','Volume'] features
        :param prediction_time: no of days
        :param test_data_size: size of test data to be used
        :param unroll_length: how long a window should be used for train test split
        :return: X_train : training sets of feature
        :return: X_test : test sets of feature
        :return: y_train: training sets of label
        :return: y_test: test sets of label
    """
    # training data
```

```
    test_data_cut = test_data_size + unroll_length + 1


    x_train = stocks[0:-prediction_time - test_data_cut].to_numpy()
    y_train = stocks[prediction_time:-test_data_cut]['Close'].to_numpy()
    # test data
    x_test = stocks[0 - test_data_cut:-prediction_time].to_numpy()
    y_test = stocks[prediction_time - test_data_cut:]['Close'].to_numpy()
    return x_train, x_test, y_train, y_test
def unroll(data, sequence_length=24):
    """
    use different windows for testing and training to stop from leak of information in the
data
    :param data: data set to be used for unrolling
    :param sequence_length: window length
    :return: data sets with different window.
    """
    result = []
    for index in range(len(data) - sequence_length):
        result.append(data[index: index + sequence_length])
    return np.asarray(result)
```

2. **LinearRegressionModel.py**

```
from sklearn import linear_model
import numpy as np
import stock_data as sd
def build_model(X, y):
    """
    build a linear regression model using sklearn.linear_model
    :param X: Feature dataset
    :param y: label dataset
    :return: a linear regression model
    """
    linear_mod = linear_model.LinearRegression()  # defining the linear regression model
    X = np.reshape(X, (X.shape[0], 1))
    y = np.reshape(y, (y.shape[0], 1))
    linear_mod.fit(X, y)  # fitting the data points in the model
    return linear_mod
def predict_prices(model, x, label_range):
    """
    Predict the label for given test sets
    :param model: a linear regression model
    :param x: testing features
    :param label_range: normalised range of label data
```

```
    :return: predicted labels for given features
    """
    x = np.reshape(x, (x.shape[0], 1))
    predicted_price = model.predict(x)
    predictions_rescaled, re_range = sd.scale_range(predicted_price, input_range=[-1.0, 1.0], target_range=label_range)
    return predictions_rescaled.flatten()
```

3. **lstm.py**

```
from keras.layers.core import Dense, Activation, Dropout
from keras.layers.recurrent import LSTM
from keras.models import Sequential
def build_improved_model(input_dim, output_dim, return_sequences):
    """
            Builds an improved Long Short term memory model using keras.layers.recurrent.lstm
    :param input_dim: input dimension of model
    :param output_dim: ouput dimension of model
    :param return_sequences: return sequence for the model
    :return: a 3 layered LSTM model
    """
    model = Sequential()
    model.add(LSTM(
        input_shape=(None, input_dim),
```

```
        units=output_dim,
        return_sequences=return_sequences))
    model.add(Dropout(0.2))
    model.add(LSTM(
        144,
        return_sequences=False))
    model.add(Dropout(0.2))
    model.add(Dense(
        units=1))
    model.add(Activation('linear'))
    return model
def build_basic_model(input_dim, output_dim, return_sequences):
    """
    Builds a basic lstm model
    :param input_dim: input dimension of the model
    :param output_dim: output dimension of the model
    :param return_sequences: return sequence of the model
    :return: a basic lstm model with 3 layers.
    """
    model = Sequential()
    model.add(LSTM(
        input_shape=(None, input_dim),
        units=output_dim,
```

```
        return_sequences=return_sequences))
    model.add(LSTM(
        100,
        return_sequences=False))
    model.add(Dense(
        units=1))
    model.add(Activation('linear'))
    return model
```

4. **Stock_Price_Predictor.ipynb**

```
import pandas as pd
def get_historical_data():
    col_names = ['Date','Open','High','Low','Close','Volume']
    stocks = pd.read_csv('./GOOG.csv', header=0, names=col_names)
    df = pd.DataFrame(stocks)
    return df
data = get_historical_data() # from January 1, 2008 to June 30, 2021
data.to_csv('google.csv',index = False)
import pandas as pd
import numpy as np
data = pd.read_csv('google.csv')
print(data.head())
print("\n")
```

```
print("Open --- mean :", np.mean(data['Open']), " \t Std: ", np.std(data['Open']), " \t Max: ", np.max(data['Open']), " \t Min: ", np.min(data['Open']))

print("High --- mean :", np.mean(data['High']), " \t Std: ", np.std(data['High']), " \t Max: ", np.max(data['High']), " \t Min: ", np.min(data['High']))

print("Low --- mean :", np.mean(data['Low']), " \t Std: ", np.std(data['Low']), " \t Max: ", np.max(data['Low']), " \t Min: ", np.min(data['Low']))

print("Close --- mean :", np.mean(data['Close']), " \t Std: ", np.std(data['Close']), " \t Max: ", np.max(data['Close']), " \t Min: ", np.min(data['Close']))

print("Volume --- mean :", np.mean(data['Volume']),"\t Std: ", np.std(data['Volume']),"\t Max: ", np.max(data['Volume']),"\t Min: ", np.min(data['Volume']))
```

References

1. Hu Z, Zhu J, Tse K. ,2013 “Stocks market prediction using support vector machines.” , pp. 115-118. IEEE.
2. Usmani M, Adil SH, Raza K, Ali SS. ,2016 “Stock market prediction using machine learning techniques” ,pp.322-327. IEEE.
3. Mankar T, Hotchandani T, Madhwani M, Chidrawar A, Lifna CS,2018,”Stock market prediction based on social sentiments using machine learning.” ,pp.1-3.IEEE.
4. Ouahilal M, El Mohajir M, Chahhou M, El Mohajir BE.,2016,” Optimizing stock market price prediction using a hybrid approach based on HP filter and support vector regression.” ,pp. 290-294. IEEE.
5. Huang H, Zhang W, Deng G, Chen J. ,2014 “Predicting stock trend using fourier transform and support vector regression.” , pp. 213-216. IEEE.
6. Sharma N, Juneja A.,2017 ,”Combining of random forest estimates using LSboost for stock market index prediction.” , pp.1199-1202. IEEE.
7. Tiwari S, Bharadwaj A, Gupta S. ,2017 ,”Stock price prediction using data analytics.”, pp. 1-5.IEEE.
8. Pun TB, Shahi TB. , 2018, “Nepal stock exchange prediction using support vector regression and neural net-works.”, pp. 1-6. IEEE.
9. Ramesh LS, Ganapathy S, Bhuvaneshwari R, Kulothungan K, Pandiyaraju V, Kannan A. ,2015, “Prediction of user interests for providing relevant information using relevance feedback and re-ranking.” International Journal of Intelligent Information Technologies (IJIIT). 2015 Oct 1;11(4):55-71.
10. Selvakumar K, Ramesh LS, Kannan A. Enhanced K-means clustering algorithm for evolving user groups. Indian Journal of Science and Technology. 2015 Sep 1;8(24):1-8.
11. Sabena S, Kalaiselvi S, Anusha B, Ramesh LS. An Multi-Label Classification with Label Correlation. Asian Journal of Research in Social Sciences and Humanities. 2016;6(9):373-86.
12. Aravind G, Uma GV, SaiRamesh L. Knowledge Management Process Model using IHS based on Detecting Process with Less Energy Consumption. Journal of Electrical Engineering. 2020;20(1):1-7.

13. SaiRamesh L, Ashok E, Sabena S, Ayyasamy A. Credit Card Fraud Detectıon in Retaıl Shopping Using Reinforcement Learning. In International Conference On Computational Vision and Bio Inspired Computing 2018 Nov 29 (pp. 1541-1549). Springer, Cham.
14. Xu X. Machine learning-based prediction of urban soil environment and corpus translation teaching.Arabian Journal of Geosciences. 2021 Jun;14(11):1-5.
15. Wang L. Urban land ecological evaluation and English translation model optimization based on machine learning. Arabian Journal of Geosciences. 2021 Jun;14(11):1-6.