Stock Market Price Prediction

A report submitted in partial fulfilment of the

requirement for the award of degree of

BACHELORS OF ENGINEERING

in

COMPUTER SCIENCE ENGINEERING

By

Sahil Raja	**22BCT10012**
Shresth Varshney	**22BCT10013**
Krish Tyagi	**22BCT10014**
Siddharth Aasal	**22BCT10056**
Priyanshu Yadav	**22BCT10057**

under the guidance of

Dr. Harjeevan Singh

Assistant Professor

Computer Science Engineering

UIE, Chandigarh University

ACKNOWLEDGEMENT

We appreciate that we have such an opportunity to express our great gratitude and respect to the people who helped us during the completion of our B.E. project. Without their support and encouragement, it was not possible for us to complete the project successfully.

It is difficult to overstate my greatest gratitude to my project mentor Dr. Harjeevan Singh. Firstly, we would like to thank him for guiding and inspiring us patiently throughout our study period. Secondly, we highly appreciate his encouragement and support in our project work, which helped us build confidence and courage to overcome difficulties. Finally, we are grateful for his great insights and suggestions and for devoting so much time to the completion of the project.

Many thanks go to our other class teachers who have given their full effort in guiding the team in achieving our goal as well as their encouragement for completing the project timely. Our profound thanks go to all our classmates, especially to our friends for spending their time in helping and giving support whenever we needed it in our project.

At last, we would like to thank our family members for supporting and motivating us for completing this project.

ABSTRACT

In a financially volatile market, as the stock market, it is important to have a very precise prediction of a future trend. The recent trend in stock market prediction technologies is the use of machine learning which makes predictions based on the values of current stock market indices by training on their previous values. The project focuses on the use of Simple Moving Averages, Exponential Moving Averages, and LSTM based Machine Learning to predict stock values. The factor which is considered in this project is closing price of a stock to predict the prices of future stocks of the company,

LSTMs are very powerful in sequence prediction problems because they are able to store past information. This is important in our case because the previous price of a stock is crucial in predicting its future price. While predicting the actual price of a stock is an uphill climb, we can build a model that will predict whether the price will go up or down.

Keywords: Simple Moving Average, Exponential Moving Average, LSTM, ML, Trade Close.

CONTENTS

LIST OF SYMBOLS

Xt	Input at current state
X(t-1)	Input at Previous state
Ct	Current Cell State
C(t-1)	Previous Cell State
ht	Current hidden/output State
h(t-1)	Previous hidden/output State
σ	Sigmoid Function
tanh	Hyperbolic tangent function

LIST OF FIGURES

LIST OF ABBREVIATIONS

LSTM	Long Short-Term Memory
SMA	Simple Moving Average
EMA	Exponential Moving Average
ATS	Automated Trading System
ML	Machine Learning
SVM	Support Vector Machine
EMH	Efficient Market hypothesis
AI	Artificial Intelligence
NN	Neural Networks
ARMA	Autoregressive Moving Average
LMS	Least Mean Square
RMSE	Root Mean Squared Error
MAPE	Mean Absolute Percentage Error

CHAPTER 1: PROBLEM STATEMENT

Stock market prediction and analysis are some of the most difficult jobs to complete. There are numerous causes for this, including market volatility and a variety of other dependent and independent variables that influence the value of a certain stock in the market. These variables make it extremely difficult for any stock market expert to anticipate the rise and fall of the market with great precision.

Although humans can make the predictions themselves, automated trading systems (ATS) that are operated by the implementation of computer programs can perform better and with higher momentum in terms of speed and accuracy than any human. However, to evaluate and control the performance of ATSs, the implementation of risk strategies and safety measures applied based on human judgements are required. Many factors are incorporated and considered when developing a model to predict future stock prices, for instance, strategy to be adopted, complex mathematical functions that reflect the state of a specific stock, machine learning algorithms that enable the prediction of the future stock value, and specific news related to the stock being analysed.

In this project we attempt to implement various machine learning approaches to predict stock prices.

CHAPTER 2: LITERATURE SURVEY

2.1 Stock Market Prediction Using Machine Learning

The research work done by V Kranthi Sai Reddy Student, ECM, Sreenidhi Institute of Science and Technology, Hyderabad, India. In the finance world stock trading is one of the most important activities. Stock market prediction is an act of trying to determine the future value of a stock other financial instrument traded on a financial exchange. This paper explains the prediction of a stock using Machine Learning. The technical and fundamental or the time series analysis is used by the most of the stockbrokers while making the stock predictions. The programming language is used to predict the stock market using machine learning is Python. In this paper we propose a Machine Learning (ML) approach that will be trained from the available stocks data and gain intelligence and then uses the acquired knowledge for an accurate prediction. In this context this study uses a machine learning technique called Support Vector Machine (SVM) to predict stock prices for the large and small capitalizations and in the three different markets, employing prices with both daily and up-to-the-minute frequencies.

2.2 Forecasting the Stock Market Index Using Artificial Intelligence Techniques

The research work done by Lufuno Ronald Marwala A dissertation submitted to the Faculty of Engineering and the Built Environment, University of the Witwatersrand, Johannesburg, in fulfilment of the requirements for the degree of Master of Science in Engineering. The weak form of Efficient Market hypothesis (EMH) states that it is impossible to forecast the future price of an asset based on the information contained in the historical prices of an asset. This means that the market behaves as a random walk and as a result makes forecasting impossible. Furthermore, financial forecasting is a difficult task due to the intrinsic complexity of the financial system. The objective of this work was to use artificial intelligence (AI) techniques to model and predict the future price of a stock market index. Three artificial intelligence techniques, namely, neural networks (NN), support vector machines and neuro-fuzzy systems are implemented in forecasting the future price of a stock market index based on its historical price information. Artificial intelligence techniques have the ability to take into consideration financial system complexities and they are used as financial time series forecasting tools.

Two techniques are used to benchmark the AI techniques, namely, Autoregressive Moving Average (ARMA) which is linear modelling technique and random walk (RW) technique. The experimentation was performed on data obtained from the Johannesburg Stock Exchange. The data used was a series of past closing prices of the All Share Index. The results showed that the three techniques have the ability to predict the future price of the Index with an acceptable accuracy. All three artificial intelligence techniques outperformed the linear model. However, the random walk method out performed all the other techniques. These techniques show an ability to predict the future price however, because of the transaction costs of trading in the market, it is not possible to show that the three techniques can disprove the weak form of market efficiency. The results show that the ranking of performances support vector machines, neuro-fuzzy systems, multilayer perceptron neural networks is dependent on the accuracy measure used.

2.3 Indian stock market prediction using artificial neural networks on tick data

The research work done by Dharmaraja Selvamuthu, Vineet Kumar and Abhishek Mishra Department of Mathematics, Indian Institute of Technology Delhi, Hauz Khas, New Delhi 110016, India. A stock market is a platform for trading of a company's stocks and derivatives at an agreed price. Supply and demand of shares drive the stock market. In any country stock market is one of the most emerging sectors. Nowadays, many people are indirectly or directly related to this sector. Therefore, it becomes essential to know about market trends. Thus, with the development of the stock market, people are interested in forecasting stock price. But, due to dynamic nature and liable to quick changes in stock price, prediction of the stock price becomes a challenging task. Stock m Prior work has proposed effective methods to learn event representations that can capture syntactic and semantic information over text corpus, demonstrating their effectiveness for downstream tasks such as script event prediction. On the other hand, events extracted from raw texts lacks of common-sense knowledge, such as the intents and emotions of the event participants, which are useful for distinguishing event pairs when there are only subtle differences in their surface realizations. To address this issue, this paper proposes to leverage external common-sense knowledge about the intent and sentiment of the event.

2.4 Automated Stock Price Prediction Using Machine Learning

The research work done by Mariam Moukalled Wassim El-Hajj Mohamad Jaber Computer Science Department American University of Beirut. Traditionally and in order to predict market movement, investors used to analyse the stock prices and stock indicators in addition to the news related to these stocks. Hence, the importance of news on the stock price movement. Most of the previous work in this industry focused on either classifying the released market news as (positive, negative, neutral) and demonstrating their effect on the stock price or focused on the historical price movement and predicted their future movement. In this work, we propose an automated trading system that integrates mathematical functions, machine learning, and other external factors such as news' sentiments for the purpose of achieving better stock prediction accuracy and issuing profitable trades. Particularly, we aim to determine the price or the trend of a certain stock for the coming end-of-day considering the first several trading hours of the day. To achieve this goal, we trained traditional machine learning algorithms and created/trained multiple deep learning models taking into consideration the importance of the relevant news.

Various experiments were conducted, the highest accuracy (82.91%) of which was achieved using SVM for Apple Inc. (AAPL) stock.

2.5 Stock Price Correlation Coefficient Prediction with ARIMA- LSTM Hybrid Model

The research work done by Hyeong Kyu Choi, B.A Student Dept. of Business Administration Korea University Seoul, Korea. Predicting the price correlation of two assets for future time periods is important in portfolio optimization. We apply LSTM recurrent neural networks (RNN) in predicting the stock price correlation coefficient of two individual stocks. RNN's are competent in understanding temporal dependencies. The use of LSTM cells further enhances its long-term predictive properties. To encompass both linearity and nonlinearity in the model, we adopt the ARIMA model as well. The ARIMA model filters linear tendencies in the data and passes on the residual value to the LSTM model. The ARIMA-LSTM hybrid model is tested against other traditional predictive financial models such as the full historical model, constant correlation model, single-index model and the multi-group model. In our empirical study, the predictive ability of the ARIMA-LSTM model turned out superior to all other financial models by a significant scale. Our work implies that it is worth considering the ARIMA-LSTM model to forecast correlation coefficient for portfolio optimization.

CHAPTER 3: PROPOSED FRAMEWORK

3.1 PROPOSED SYSTEMS

The prediction methods can be roughly divided into two categories, statistical methods and artificial intelligence methods. Statistical methods included in this project are Simple Moving Averages and Exponential Moving Averages. Artificial Intelligence model used in the project is Long short-term memory network (LSTM).

3.2 SIMPLE MOVING AVERAGES

Simple Moving Average is the average price over the specified period. The average is called "moving" because it is plotted on the chart bar by bar, forming a line that moves along the chart as the average value changes**.**

Formula:

$$\text{Simple Moving Average} = \frac{(A_1 + A_2 + \ldots\ldots. + A_n)}{n}$$

$(A_{1,}\ A_2 \ldots\ldots.\ A_n)$ = Prices

n = The number of total periods

Figure 1

3.3 EXPONENTIAL MOVING AVERAGES

An exponential moving average (EMA) is a type of moving average (MA) that places a greater weight and significance on the most recent data points. The exponential moving average is also referred to as the exponentially weighted moving average.

- The EMA is a moving average that places a greater weight and significance on the most recent data points.
- Like all moving averages, this technical indicator is used to produce buy and sell signals based on crossovers and divergences from the historical average.
- Traders often use several different EMA lengths, such as 10-day, 50-day, and 200-day moving averages.

Formula:

$$\text{EMA}_{\text{Today}} = \text{Price Today} \times \left(\frac{\text{Smoothing}}{1 + \text{Days}}\right) + \text{EMA Yesterday} \left(1 - \left(\frac{\text{Smoothing}}{1 + \text{Days}}\right)\right)$$

Figure 2

3.4 LONG SHORT-TEM MEMORY NETWORK:

Long short-term memory network (LSTM) is a particular form of recurrent neural network (RNN).

3.5 Working of LSTM:

LSTM is a special network structure with three “gate” structures. The three gates in an LSTM unit, called input gate, forgetting gate and output gate. While information enters the LSTM’s network, it can be selected by rules. Only the information conforms to the algorithm will be left, and the information that does not conform will be forgotten through the forgetting gate..

3.6 LSTM ARCHITECTURE:

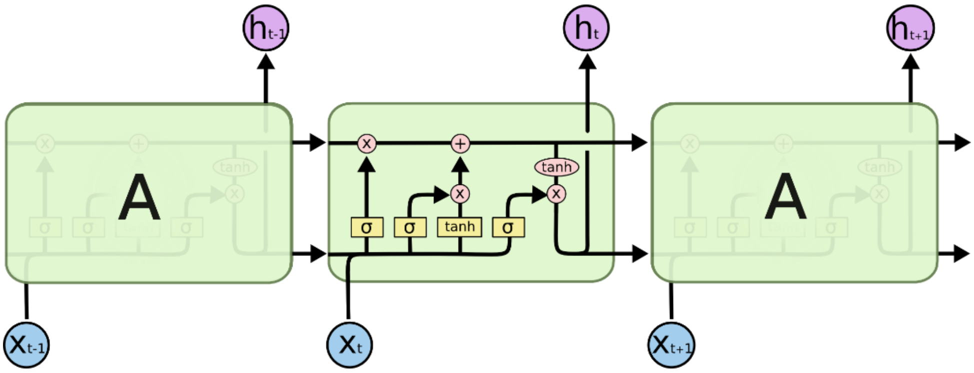

Figure 3

3.6.1 FORGET GATE:

A forget gate is responsible for removing information from the cell state.

- The information that is no longer required for the LSTM to understand things or the information that is of less importance is removed via multiplication of a filter.
- This is required for optimizing the performance of the LSTM network.
- This gate takes in two inputs; h_t-1 and x_t. h_t-1 is the hidden state from the previous cell or the output of the previous cell and x_t is the input at that particular time step.

3.6.2 INPUT GATE:

- Regulating what values need to be added to the cell state by involving a sigmoid function. This is basically very similar to the forget gate and acts as a filter for all the information from hi-1 and x_t.
- Creating a vector containing all possible values that can be added (as perceived from h_t-1 and x_t) to the cell state. This is done using the tanh function, which outputs values from -1 to +1.
- Multiplying the value of the regulatory filter (the sigmoid gate) to the created vector (the tanh function) and then adding this useful information to the cell state via addition operation.

3.6.3 OUTPUT GATE:

The functioning of an output gate can again be broken down to three steps:

- Creating a vector after applying tanh function to the cell state, thereby scaling the values to the range -1 to +1.
- Making a filter using the values of h_t-1 and x_t, such that it can regulate the values that need to be output from the vector created above. This filter again employs a sigmoid function.
- Multiplying the value of this regulatory filter to the vector created in step 1, and sending it out as a output and also to the hidden state of the next cell.

3.7 Hardware Requirements:

- RAM: 4 GB
- Storage: 500 GB
- CPU: 2 GHz or faster
- Architecture: 32-bit or 64-bit

3.8 Software Requirements:

- Python 3.5 in Google Colab is used for data pre-processing, model training and prediction.
- Operating System: windows 7 and above or Linux based OS or MAC OS.

3.9 Functional requirements

- The software shall accept the tw_spydata_raw.csv dataset as input.
- The software should shall do pre-processing (like verifying for missing data values) on input for model training.
- The model shall use LSTM ARCHITECTURE as main component of the software.
- It processes the given input data by producing the most possible outcomes of a CLOSING STOCK PRICE.

3.10 FLOW OF WORK IN LSTM:

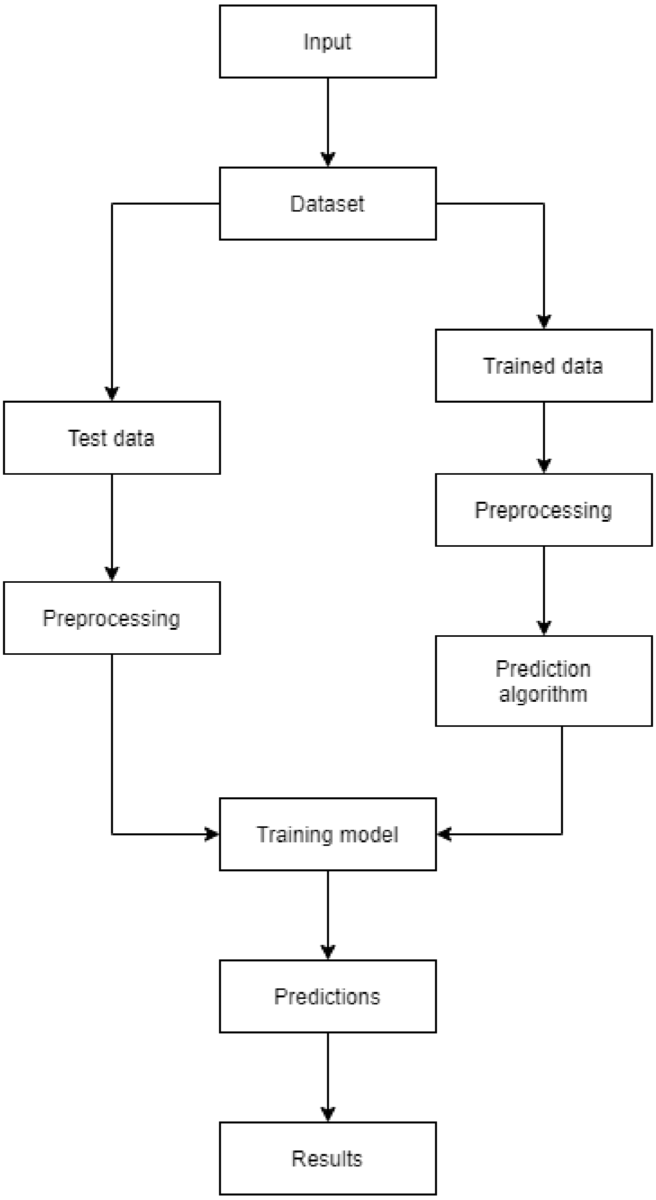

Figure 4

CHAPTER 4: CODE OF THE PROJECT:

```
[1]  !pip install yfinance
```

```
# Import all the packages for analysis
import os
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
from sklearn.preprocessing import MinMaxScaler, StandardScaler
import math
import yfinance as yf
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
```

```
stock_data = yf.download('AAPL', start='2016-11-01', end='2021-10-01')   #importing dataset from the web
print(stock_data.head())          #printing the head of dataset
stock_data.to_csv('Dataset.csv')  #downloading csv format of the dataset
```

```
[*********************100%***********************]  1 of 1 completed
                 Open       High        Low      Close  Adj Close     Volume
Date
2016-11-01  28.365000  28.442499  27.632500  27.872499  25.951244  175303200
2016-11-02  27.850000  28.087500  27.807501  27.897499  25.974527  113326800
2016-11-03  27.745001  27.865000  27.387501  27.457500  25.696110  107730400
2016-11-04  27.132500  27.562500  27.027500  27.209999  25.464489  123348000
2016-11-07  27.520000  27.627501  27.365000  27.602501  25.831810  130240000
```

```
df = pd.read_csv('Dataset.csv')    #df contains dataset
# Sort DataFrame by date
stockprices = df.sort_values('Date')
```

```
[ ]  # PLOTTING THE DATASET
     plt.figure(figsize=(15, 8))
     plt.title('Stock Prices History')
     plt.plot(stockprices['Close'])
     plt.xlabel('Date')
     plt.ylabel('Prices ($)')
```

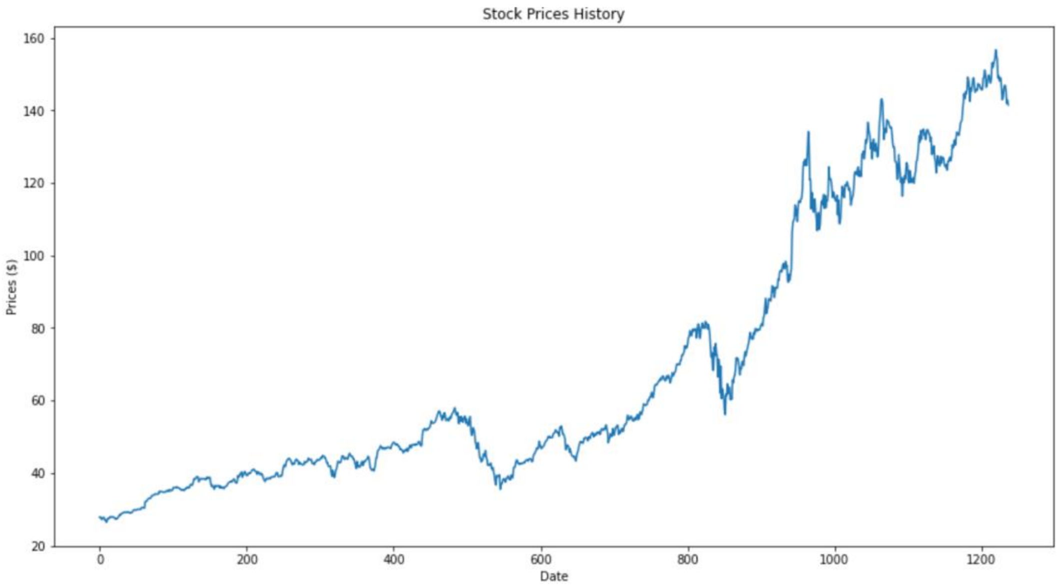

Figure 5

```
[ ]  # Train-Test split for time-series
     test_ratio = 0.2
     training_ratio = 1 - test_ratio

     train_size = int(training_ratio * len(stockprices))
     test_size = int(test_ratio * len(stockprices))

     print("train_size: " + str(train_size))
     print("test_size: " + str(test_size))

     train = stockprices[:train_size][['Date', 'Close']]
     test = stockprices[train_size:][['Date', 'Close']]

     # trainX = stockprices[:train_size][['Close']]
     # testX = stockprices[train_size:][['Close']]
```

```
train_size: 989
test_size: 247
```

```
#### Calculate the metrics RMSE and MAPE ####
def calculate_rmse(y_true, y_pred):
    """
    Calculate the Root Mean Squared Error (RMSE)
    """
    rmse = np.sqrt(np.mean((y_true-y_pred)**2))
    return rmse

def calculate_mape(y_true, y_pred):
    """
    Calculate the Mean Absolute Percentage Error (MAPE) %
    """
    y_pred, y_true = np.array(y_pred), np.array(y_true)
    mape = np.mean(np.abs((y_true-y_pred) / y_true))*100
    return mape
```

```python
#SIMPLE MOVING AVERAGES
stockprices['100days'] = stockprices['Close'].rolling(100).mean()

# RMSE and MAPE of Simple Moving Averages
rmse_SMA100 = calculate_rmse(np.array(stockprices[train_size:]['Close']), np.array(stockprices[train_size:]['100days']))
mape_SMA100 = calculate_mape(np.array(stockprices[train_size:]['Close']), np.array(stockprices[train_size:]['100days']))
print(rmse_SMA100)
print(mape_SMA100)
```

```
10.178922386102297
6.418805451354498
```

```python
#PLOTTING SIMPLE MOVING AVERAGES
plt.figure(figsize=(15, 8))
plt.title('100 Days MA')
plt.plot(stockprices['Close'])
plt.plot(stockprices['100days'])
plt.xlabel('Date')
plt.ylabel('Prices ($)')
```

```
Text(0, 0.5, 'Prices ($)')
```

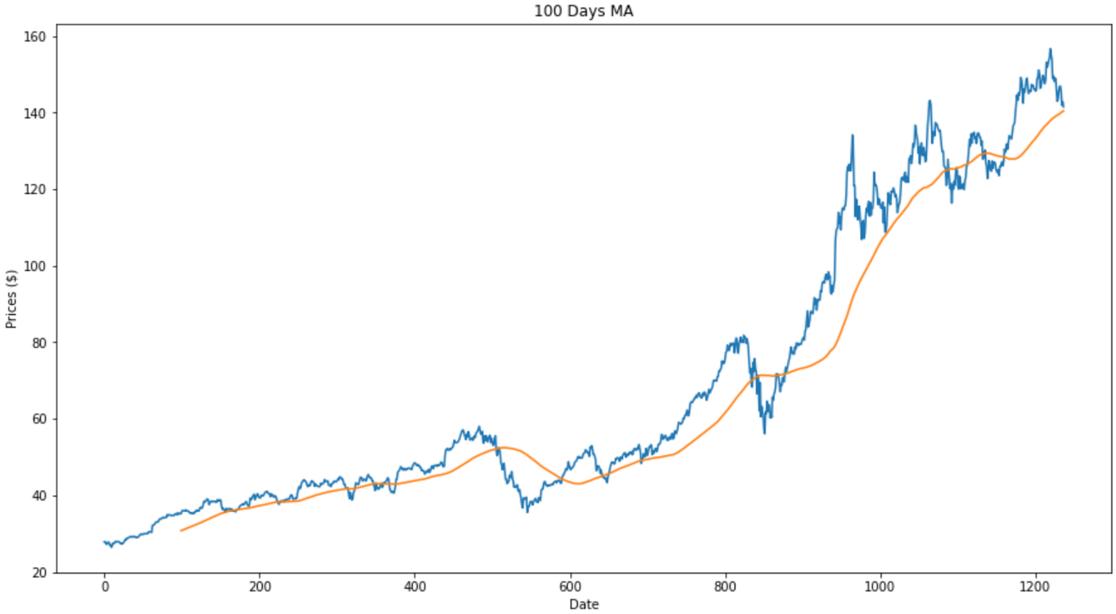

Figure 6

```python
#CALCULATING EXPONENTIAL MOVING AVERAGES
stockprices['100Days'] = stockprices['Close'].ewm(span=100, adjust=False).mean()

# RMSE and MAPE of Exponential Moving Averages
rmse_EMA100 = calculate_rmse(np.array(stockprices[train_size:]['Close']), np.array(stockprices[train_size:]['100Days']))
mape_EM Loading... culate_mape(np.array(stockprices[train_size:]['Close']), np.array(stockprices[train_size:]['100Days']))
print(rmse_EMA100)
print(mape_EMA100)
```

```
10.066498262993335
6.393949044641876
```

```python
#PLOTTING EXPONENTIAL MOVING AVERAGES
plt.figure(figsize=(15, 8))
plt.title('100 Days EMA')
plt.plot(stockprices['Close'])
plt.plot(stockprices['100Days'])
plt.xlabel('Date')
plt.ylabel('Prices ($)')
```

```python
#PLOTTING EXPONENTIAL MOVING AVERAGES
plt.figure(figsize=(15, 8))
plt.title('100 Days EMA')
plt.plot(stockprices['Close'])
plt.plot(stockprices['100Days'])
plt.xlabel('Date')
plt.ylabel('Prices ($)')
```

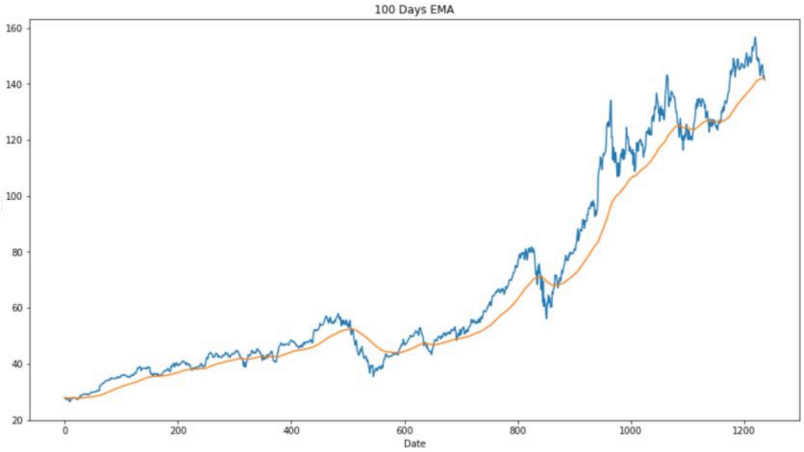

Figure 7

LSTM

```python
[ ]  # scale our dataset
     scaler =  MinMaxScaler(feature_range=(0,1))
     scaled_data = scaler.fit_transform(stockprices[['Close']])
     scaled_data_train = scaled_data[:train.shape[0]]
```

```python
[ ]  x_train = []
     y_train = []

     for i in range(60, scaled_data_train.shape[0]):
         x_train.append(scaled_data_train[i-60:i])
         y_train.append(scaled_data_train[i,0])

     x_train, y_train = np.array(x_train), np.array(y_train)
```

```python
[ ]  def preprocess_testdat(data=stockprices, scaler=scaler, window_size=60, test=test):
         raw = data['Close'][len(data) - len(test) - window_size:].values
         raw = raw.reshape(-1,1)
         raw = scaler.transform(raw)

         X_test = []
         for i in range(window_size, raw.shape[0]):
             X_test.append(raw[i-window_size:i, 0])

         X_test = np.array(X_test)

         X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))
         return X_test

     X_test = preprocess_testdat()
```

```
#ML
from keras.models import Sequential, Model
from keras.layers import Dense, Dropout, LSTM, Input, Activation, concatenate
```

+ Code + Text

```
model = keras.Sequential()
model.add(layers.LSTM(100, return_sequences=True, input_shape=(x_train.shape[1], 1)))
model.add(layers.LSTM(100, return_sequences=False))
model.add(layers.Dense(25))
model.add(layers.Dense(1))
model.summary()
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_2 (LSTM)               (None, 60, 100)           40800

 lstm_3 (LSTM)               (None, 100)               80400

 dense_2 (Dense)             (None, 25)                2525

 dense_3 (Dense)             (None, 1)                 26

=================================================================
Total params: 123,751
Trainable params: 123,751
Non-trainable params: 0
_________________________________________________________________
```

- Define a Sequential model which consists of a linear stack of layers.
- Add a LSTM layer by giving it 100 network units. Set the return_sequence to true so that the output of the layer will be another sequence of the same length.
- Add another LSTM layer with also 100 network units. But we set the return_sequence to false for this time to only return the last output in the output sequence.
- Add a densely connected neural network layer with 25 network units.
- At last, add a densely connected layer that specifies the output of 1 network unit.
- Show the summary of our LSTM network architecture.

```
model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(x_train, y_train, batch_size= 1, epochs=3)
```

```
Epoch 1/3
929/929 [==============================] - 20s 19ms/step - loss: 0.0017
Epoch 2/3
929/929 [==============================] - 18s 19ms/step - loss: 7.9089e-04
Epoch 3/3
929/929 [==============================] - 18s 19ms/step - loss: 4.7210e-04
<keras.callbacks.History at 0x7f7dc584fd00>
```

```
# predict stock prices using past window_size stock price

predicted_price_ = model.predict(X_test)
predicted_price = scaler.inverse_transform(predicted_price_)

test['Predictions_lstm'] = predicted_price
```

```
8/8 [==============================] - 1s 17ms/step
```

```
# RMSE and MAPE of LSTM MODEL
rmse_lstm = calculate_rmse(np.array(test['Close']), np.array(test['Predictions_lstm']))
mape_lstm = calculate_mape(np.array(test['Close']), np.array(test['Predictions_lstm']))
print(rmse_lstm)
print(mape_lstm)
```

```
2.707583768333171
1.6015668543708221
```

```
# Plot predicted price vs actual closing price
def plot_stock_trend_lstm(train, test, logNeptune=True):
    fig = plt.figure(figsize = (20,10))
    plt.plot(train['Date'], train['Close'], label = 'Train Closing Price')
    plt.plot(test['Date'], test['Close'], label = 'Test Closing Price')
    plt.plot(test['Date'], test['Predictions_lstm'], label = 'Predicted Closing Price')
    plt.title('LSTM Model')
    plt.xlabel('Date')
    plt.ylabel('Stock Price ($)')
    plt.legend(loc="upper left")

plot_stock_trend_lstm(train, test)
```

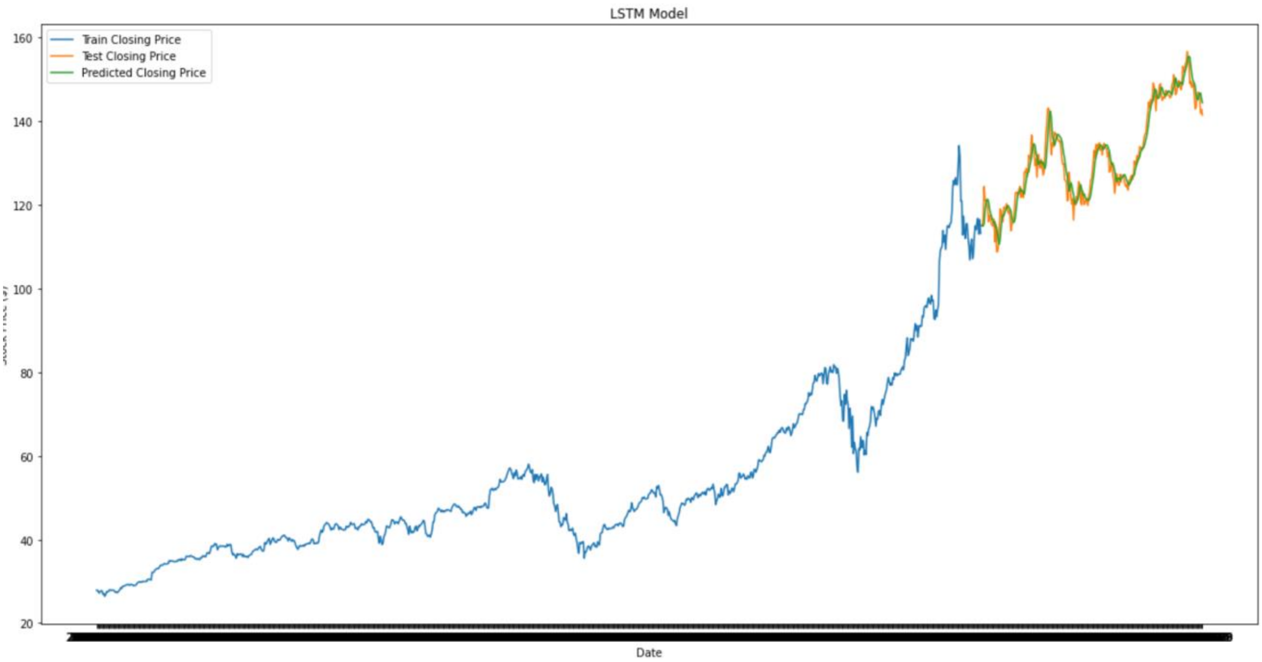

Figure 8

CHAPTER 5: RESULTS

5.1 MODEL COMPARISION:

MODEL	RMSE	MAPE
SMA	10.17	6.41
EMA	10.06	6.39
LSTM	2.80	1.65

5.2 The Difference Between EMA and SMA

The major difference between an EMA and an SMA is the sensitivity each one shows to changes in the data used in its calculation.

More specifically, the EMA gives higher weights to recent prices, while the SMA assigns equal weights to all values. The two averages are similar because they are interpreted in the same manner and are both commonly used for future stock prediction.

Since EMAs place a higher weighting on recent data than on older data, they are more responsive to the latest price changes than SMAs. That makes the results from EMAs more accurate.

5.3 Why LSTM performed better?

LSTMs are very powerful in sequence prediction problems because they’re able to store past information. This is important in our case because the previous price of a stock is crucial in predicting its future price.

5.4 LSTM Model Price Prediction of Apple Stocks:

Actual Price	Predicted Price
115.080002	114.515602
114.970001	114.481117
116.970001	115.101227
124.400002	117.893021
121.099998	...
...	146.132278
146.919998	146.873108
145.369995	146.955292
141.910004	145.756500
142.830002	144.747665
141.500000	

CONCLUSION:

The prices of Apple have been predicted through Simple Moving Averages, Exponential Moving Averages and Long Short-Term Memory models. On comparison the LSTM model showed least error and hence proved itself suitable for stock price predictions.

FUTURE WORK:

- We will extend this project into a fully-fledged web application.
- We will add sentiment analysis as well in the project in addition to the technical analysis for better prediction of the stock prices.

REFERENCES

- S. Selvin, R. Vinayakumar, E. A. Gopalkrishnan, V. K. Menon and K. P. Soman, "Stock price prediction using LSTM, RNN and CNN-sliding window model," in International Conference on Advances in Computing, Communications and Informatics, 2017.

- Murtaza Roondiwala, Harshal Patel, Shraddha Varma, “Predicting Stock Prices Using LSTM” in Undergraduate Engineering Students, Department of Information Technology, Mumbai University, 2015.

- Xiongwen Pang, Yanqiang Zhou, Pan Wang, Weiwei Lin, “An innovative neural network approach for stock market prediction”, 2018

- Ishita Parmar, Navanshu Agarwal, Sheirsh Saxena, Ridam Arora, Shikhin Gupta, Himanshu Dhiman, Lokesh Chouhan Department of Computer Science and Engineering National Institute of Technology, Hamirpur – 177005, INDIA - Stock Market Prediction Using Machine Learning.

- Pranav Bhat Electronics and Telecommunication Department, Maharashtra Institute of Technology, Pune. Savitribai Phule Pune University - A Machine Learning Model for Stock Market Prediction.

- Anurag Sinha Department of computer science, Student, Amity University Jharkhand Ranchi, Jharkhand (India), 834001 - Stock Market Prediction Using Machine Learning.

- V Kranthi Sai Reddy Student, ECM, Sreenidhi Institute of Science and Technology, Hyderabad, India - Stock Market Prediction Using Machine Learning.