

STOCK MARKET PREDICTION ANALYSIS

*A Project Report submitted in partial fulfilment
Of the requirement for the award of the degree of*

BACHELOR OF TECHNOLOGY

In

Instrumentation and Control Engineering

Submitted by

Abhishek Shukla - 160921066
Uttaran Tribedi - 160921118

Under the guidance of

DR RENUKA A
Professor
Department of
Computer Science And Engineering

&

DR SANDRA D'SOUZA
Associate Professor
Department of Instrumentation
And Control Engineering

DEPARTMENT OF INSTRUMENTATION AND CONTROL ENGINEERING



MANIPAL INSTITUTE OF TECHNOLOGY

MANIPAL

(A constituent unit of MAHE, Manipal)

JUNE 2020



DEPARTMENT OF INSTRUMENTATION AND CONTROL ENGINEERING

MANIPAL INSTITUTE OF TECHNOLOGY

(A Constituent Institution of Manipal Academy of Higher Education)

MANIPAL – 576 104 (KARNATAKA), INDIA

Manipal
31st May 2020

CERTIFICATE

This is to certify that the project titled **STOCK MARKET PREDICTION ANALYSIS** is a record of the bonafide work done by **UTTARAN TRIBEDI** (*Reg. No.160921118*) and **ABHISHEK SHUKLA** (*Reg. No. 160921066*) submitted in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology (BTech) in **INSTRUMENTATION AND CONTROL ENGINEERING** of Manipal Institute of Technology, Manipal, Karnataka, (A Constituent Institution of Manipal Academy of Higher Education), during the academic year 2019-2020.

Dr Renuka A

Project Guide

Dr Sandra D'Souza

Project Guide

Prof. Dr Shreesha C

HOD, ICE.

MIT, MANIPAL

ACKNOWLEDGEMENT

For this work to be successful, there are many people to whom we are thankful. Foremost, we thank our guides Dr Renuka A, Professor, Department of Computer Science and Engineering and Dr Sandra D'Souza, Associate Professor, Department of Instrumentation and Control Engineering for being extremely patient, for providing valuable guidance to tiniest of the queries, encouraging continuously throughout this project. We are very much honoured to have worked under both of them.

We extend our heartfelt thanks to Prof. Dr Shreesha C, Head of the Department, Instrumentation and Control Engineering, Manipal Institute of Technology for allowing us to showcase our academic abilities and allowing us to have benefits of the well-equipped laboratories.

Our sincere gratitude to Dr D.Shrikanth Rao, Director, Manipal Institute of Technology for providing ample amount of working space and wonderful academic availabilities for all the students.

We are immensely grateful to Prof. Santhosh KV and Prof. P Chenchu Saibabu from the Department of Instrumentation and Control Engineering, Manipal Institute of Technology for providing insightful and timely guidance for the completion of the project.

We are indebted to the whole faculty of the Department of Instrumentation and Control Engineering for always being immensely supportive and encouraging throughout our B.Tech course. We are incredibly grateful to all our classmates for providing words of encouragement and for always being there. Last but not least, we are also thankful to our Department Laboratory in-charge, who has always been very helpful in matters regarding the lab. We wish all the best and immense success to everyone who has been a helping hand in our entire B.Tech course.

ABSTRACT

This project aims to analyse the financial ecosystem of the volatile stock market and extract relevant information and trends that may help generate valuable insights into the inner workings of the platform as well as obtain accurate results that would aid in making profitable predictions and decisions for the various players involved. Predicting the closing price of a company's stock with high accuracy can provide the analysts involved with the means to make financially sound decisions and serves as the primary objective of this project.

This project uses machine learning and neural networking to analyse and study past stock data of a given company, using this data as a means to build and train accurate prediction models which would later serve to predict the closing price of a given company with high precision.

The code extracts stock parameters namely opening price, closing price, the volume of stocks traded, highest price and lowest price of the company stock from the finance website Yahoo Finance in real-time, converting it into tables and storing it in an excel sheet. Different technical parameters are used to gain insights into the extracted data. Then the data is applied to two of the most accurate machine learning models to analyse and gain insights on both the different techniques.

The results reflect the uncertainties of working with stock prediction but does show a light at the end of the tunnel. The conclusions made out of this work will help people understand more about the nature of the two models and most importantly, which one to use when. This work can therefore truly benefit the ones willing to take help of technology for stock market prediction and enhance their financial gains emerging from it.

LIST OF FIGURES

Figure No	Figure Title	Page No
2.1	Plot of stock trend for Halliburton	12
2.2	RSI, Close and MACD graph	13
2.3	Basic depiction of an LSTM unit	20
3.1	Depiction of terminal UI in the program	22
3.2	Data extracted by web scraper	23
3.3	Plot of stock trend for Amazon	24
3.4	Candlestick graph – type1	25
3.5	Candlestick graph – type 2	25
3.6	Plot of MA vs Close graph	26
3.7	Plot of 20-day MA vs 200-day MA.	27
3.8	Plot of 20-day EMA vs 20-day SMA	27
3.9	Plot of RSI	28
3.10	Plot of MACD	29
3.11	Plot of RSI-Close-MACD	30
3.12	Plot of Bollinger Bands	31
3.13	Pie Chart of Sentiment Index	33
3.14	Plot of Standard and Rolling mean	34
3.15	Results of Dicky Fuller test	33
3.16	Inputs to the LSTM model	35
3.17	Dimension of the data used in LSTM model	35
4.1	Summary of the ARIMA model	37
4.2	ARIMA – Predicted vs True values plot (Microsoft)	38
4.3	ARIMA – Closing Price forecast	39
4.4	ARIMA – Predicted vs True values plot (Tesla)	38
4.5	LSTM model information	41
4.6	LSTM model summary	42
4.7	LSTM – Original vs Predicted values	42

LIST OF TABLES

Table No	Table Caption	Page No
1	Comparison of results	42

Contents		
		Page No
Acknowledgement		ii
Abstract		iii
List Of Figures		iv
List Of Tables		v
Chapter 1	INTRODUCTION	
1.1	Introduction	7
1.2	Motivation	8
1.3	Organization of Report	9
Chapter 2	BACKGROUND THEORY and/or LITERATURE REVIEW	
2.1	Introduction	10
2.2	Theoretical discussion	12
Chapter 3	METHODOLOGY	
3.1	Introduction	22
3.2	Methodology	22
Chapter 4	RESULT ANALYSIS	
4.1	Auto Regressive Integrated Moving Averages	38
4.2	Long Short Term Memory Network	42
Chapter 5	CONCLUSION AND FUTURE SCOPE	
5.1	Work Conclusion	46
5.2	Future Scope of Work	47
REFERENCES		48
PROJECT DETAILS		49

CHAPTER 1

INTRODUCTION

1.1 Introduction:

A stock market or share market is the aggregation of buyers and sellers of stocks/shares. In other words, it can be defined as a loose network of economic transactions of publicly held companies conducted through institutionalised formal exchanges which operate under a defined set of regulations. The stock market may also comprise of stock that is traded privately, such as shares of private companies which are sold to investors through equity crowdfunding platforms. Investment in the stock market is most often made via stock brokerages and electronic trading platforms. Investment is usually made with an investment strategy in mind.

High level of accuracy and precision is the crucial factor in predicting a stock market. The technical, fundamental or the time series analysis is used by most of the stockbrokers while making the predictions. Nevertheless, these methods cannot be trusted entirely, so there is a necessity to provide a supportive method for stock market prediction.

The project aims to build analytical and predictive models using various machine learning methods to forecast trends in the stock market ecosystem and yield results that may prove beneficial and profitable for all players involved. It should be noted that the stock market is volatile and prone to many fluctuations, since the market depends on the mindset of the countless traders involved, therefore to counter this hindrance, the project aims to emulate several technical indicators that provide an overview of prevailing conditions in the market and act as a litmus to the predictions borne out of the machine learning models.

1.2 Brief present-day scenario concerning the work area:

There are various research papers out there that aim to predict the closing price of a company's stock based on historical data. The stock market is a very productive environment, and, naturally, various analysts would try to develop mathematical models to make decisions and transactions more comfortable, efficient and beneficial for all involved. The major factors which differentiate the various attempts at building accurate models are the methods used, and the accuracies obtained. A 100% accuracy rate is not a feasible goal on account of the volatility involved in the day to day market trading. Instead, it is the overall trends that play an important role in trading decisions. In that regard, this project aims to build models that could effectively and accurately forecast upcoming trends and allow traders to make profitable choices as per insights generated on their parts.

1.3 Motivation:

From an economic perspective, traders, stockbrokers, and other parties involved viewing the stock market as a volatile financial ecosystem. Thus, given the unpredictability of the market, the best course of action is to categorise and analyse all the variables involved and acquire an optimal time to trade goods.

Since the stock market depends so heavily on human resources, human errors are bound to creep in. On account of the massive number of traders involved, mathematical models have been built in order to aid the traders in their task and streamline their decision-making process. These models are referred to as technical indicators, and their primary purpose is to aid traders in making decisions, display trends, predict volatility bands and analyse risk changes. These models are generated from four essential variables that are heavily involved in the daily stock market transactions, namely: Open Price, Close Price, High Price, Low Price. These variables depict the stages of the price depending upon the sampling time. The opening price is the price at which a stock first trades upon the opening of an exchange on a given trading day. The closing price is the final price at which a stock is traded on a given day (sampling time). High and low prices represent the maximum and minimum a price can reach on a given day. Optimal decisions are given when a technical indicator makes an entry or exit decision that generates profit. Because of the high number of technical indicators, it is always beneficial to determine which indicators are more accurate and minimising the number of relevant features.

1.4 Objectives:

Primary Objective:

The primary objective of this project is to construct analytical tools and build prediction models upon stock data extracted from the web to predict future closing price trends of the company with the highest accuracy possible.

Secondary Objective:

The secondary objective of this project is to compare results generated from both models and determine the more accurate method in time series forecasting concerning the stock market.

1.5 Target Specifications:

This project serves to predict closing price trends with maximum accuracy values in regards to a company's stock based on past data. A high accuracy value will play a significant role in making more informed decisions in day to day stock trading that could be profitable and beneficial to all players involved. Apart from that, the ability to predict a stock's future trends can offer valuable insights into its performance in the financial domain and highlight trends that could prove useful in future analysis.

1.6 Project Work schedule:

Jan 2020 - Feb 2020: Building of Web scraper

Feb 2020 - March 2020: Constructing stock market indicators

March 2020 – April 2020: Building Prediction models

April 2020 – May 2020: Integration of all the different parts of the code

1.7 Organisation of the project report (chapter wise):

Chapter 1 (Introduction):

Chapter 1 deals with the introduction to the project and provides the motivation behind undertaking this project along with the objectives this project aims to full fill and a basic timeline of the duration during which this project was conducted.

Chapter 2 (Background Theory):

Chapter 2 deals with the background theory necessary to understand the concepts explored in the project. It provides a brief literature review to explore previous work put into the domain, theoretical discussions centred around the core concepts of the project and a more in-depth explanation of the prediction models employed in this project.

Chapter 3 (Methodology):

Chapter 3 deals with the work that went into achieving the desired results this project aims to accomplish. It provides explanations behind the methods used and displays some of the results achieved by running the program on which this project is built along with a brief overview of how certain snippets of the code function.

Chapter 4 (Result Analysis):

Chapter 4 deals with the explanation of the results obtained in this project. It gives an individual summary of the prediction models that were used in this project and also gives a side by side comparison of the results acquired through the work put into the project.

Chapter 5 (Conclusion):

Chapter 5 deals with the conclusions obtained from analysing the results and a summary of the future scope and potential with regards to the work accomplished in this project.

CHAPTER 2

BACKGROUND THEORY

2.1 Introduction:

This section will deal with all the details regarding the same topic which has already been done or accomplished and how our approach differs from those.

Stock Market Prediction primarily refers to predicting the price of the desired stocks as accurately as possible. It is the act of trying to determine the future value of company stock or other financial instrument traded on an exchange. The successful prediction of a stock's future trend could yield a significant profit. For that, we use a combination of different Machine Learning methods and various technical indicators.

Presently there have been several methods and algorithms that have been applied and have yielded some result although they have not been as accurate or as reliable as desired.

2.2 Literature review:

The stock market prediction has become an increasingly important issue in the present time. One of the methods employed is technical analysis, but such methods do not always yield accurate results. So it is vital to develop methods for a more accurate prediction. Generally, investments are made using predictions that are obtained from the stock price after considering all the factors that might affect it. The technique that was employed in this instance was statistical machine learning and neural networks. Since financial stock marks generate enormous amounts of data at any given time, a significant volume of data needs to undergo analysis before a prediction can be made. Each of the techniques used has its advantages and limitations over its other counterparts. Each of the techniques used has its advantages and limitations over its other counterparts. As for the research done by Yuzheng Zhai, Arthur Hsu and Saman K Halgamuge [1], the existing methods neglect the impact from mass media which significantly affects the behaviour of the investors of the stock.

The methods commonly employed in this field are as follows:

Last Value: In the Last Value method, the prediction is set to the last observed value. The current adjusted closing price is set as the previous day's adjusted closing price. This is the most cost-effective prediction model and is commonly used as a benchmark against which more sophisticated models can be compared. There are no hyperparameters to be tuned here.

Moving Average: Here, the predicted value is the mean of the previous N values. This means we set the presently adjusted closing price as the mean of the adjusted closing price of the previous N days. The hyperparameter N needs to be tuned. Simple moving average, weighted

moving average, exponential moving average are some examples of moving averages commonly employed in the stock market.

Regression: Regression is the approach to modelling the relationship between a dependent variable and one or more independent variables. Regression is a useful technique that is widely employed to make accurate future predictions and is utilised in many forms such as Linear, Logistic, Multivariate, Auto-Regressive.

Extreme Gradient Boosting (XGBoost): The name XGBoost refers to the engineering goal to push the limit of computational resources for boosted tree algorithms. It is a more efficient version of gradient boosting framework containing both a linear model solver and tree learning algorithms. The reason behind its efficiency is its capacity to do parallel computing on a single machine.

Support Vector Machines: A Support Vector Machine is a method usually used for performing classification tasks, that uses a separating hyperplane in multidimensional space to perform a given task. Support vector machines find a hyperplane by maximising the distance between the plane and nearest input data points. Support Vector Machines were a very popular decade back, but now they have fallen out of favour because of them not considering many factors as per the findings of Youxun Lei, Kaiyue Zhou and Yuchen Liu [2].

The methods mentioned above serve to predict the trends and fluctuations in the closing price of a company's stock and require a lot of feature engineering and pre-processing before the data becomes usable for building prediction models. Feature engineering is a crucial aspect of predictive analysis where parameters included in the data are utilised to derive more specialised variables that are closely correlated to the target variable and serve as predictors in forecasting trends via machine learning. These previously conducted methods offer vital insights into modelling raw data into a form geared explicitly towards being efficiently utilised by machine learning models to make the most accurate forecasts possible. Many previously published papers derive such variables from the data available, using these variables to enhance their results, and serve as a means of acquiring more accuracy in predictions. These papers were instrumental in performing feature engineering on the extracted variables and deriving more parameters in hopes of increasing accuracy the models built.

The recent studies provide a well-grounded proof that most of the previously used models are inefficient like SVM and are now replaced by neural network based on ones in accordance with Youxun Lei [3]. The reason for this inefficiency was parameter instability and model uncertainty. The studies also concluded the traditional strategies that promise to solve this problem. ARIMA, short for 'Auto-Regressive Integrated Moving Average' is a class of models that 'explains' a given time series based on its past values, that is, its lags and the lagged forecast errors, which can be used to forecast future values. Therefore it is best for forecasting one-step out-of-sample forecast in accordance with the findings of Sia Siami Namini, Neda Tavakoli and Akbar Siami Namin [4]. ARIMA models are known to be robust and efficient in financial time series forecasting especially short-term prediction than even the most popular ANNs

techniques in accordance with Adebisi A. Ariyo, Adewumi O. Adewumi and Charles K. Ayo [5].

Any ‘non-seasonal’ time series exhibiting patterns and not a random white noise can be modelled with ARIMA models. ANNs, a category under which LSTM falls, have a series of interconnected nodes that simulate individual neurons and are organised into different layers based on function (input layer, processing layer, output layer, etc.). The ANN designates weights to connections, and the output is computed based on the inputs and the weights. As the machine trains, it notices patterns in the training data and reassigns the weights. Previous case studies have demonstrated that ANNs are entirely accurate when the data does not have sudden variations. Stock value although varies like Brownian motion, but historical data cannot be entirely ignored as they may contain some hidden pattern or trends that can be captured by advanced neural network-based models according to the findings by Edwin.M.Torralba [6]. The results by Dou Wei [5] also affirm that although LSTM neural network model has some limitations, such as the time lag of prediction, with attention layer, it can predict stock prices.

2.3 Theoretical discussions:

Time Series Forecasting:

Predicting future trends in a company’s stock based on its historical data falls under the bracket of time series forecasting. A time series is a series of numerical data points in consecutive order. In the stock market, a time series tracks the movement of the chosen data points, such as a stock’s price, over a specified period with data points recorded at regular intervals. There are no extreme time frames that must be included, allowing the data to be collected in a way that provides the information being sought by the analyst examining the activity. A time series can be noted on any variable that changes over time. In the stock market, it is common to use a time series to track the price of a stock’s closing price over time. This can be followed over the short term, such as the price of a security on the hour over a business day, or the long term, such as the price of a stock at close on the last day of every month over five years.

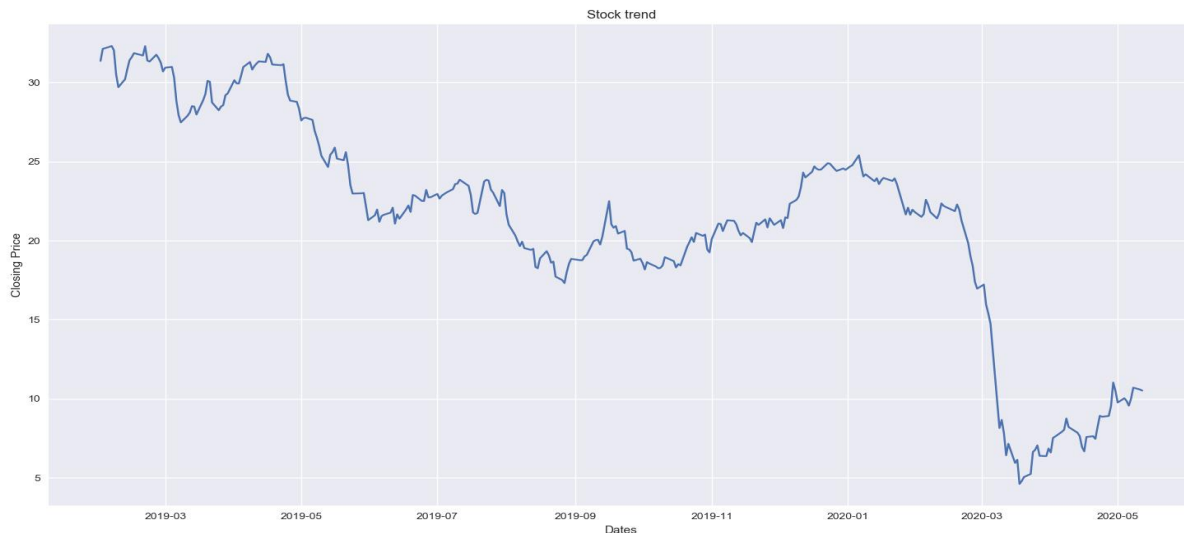


Figure 2.1 Plot of Stock Trend for Halliburton

Figure 2.1 displays a typical time series, stock data for Halliburton plotted over 14 months from March 2019 to April 2020. Time series analysis is beneficial for analysing how a given asset, security, or economic variable changes over time. It can also be for examining how the fluctuations associated with the chosen data point correlates to shifts in other variables over the same period. A time series, in general, is supposed to be affected by four main components, which can be separated from the observed data. These components are Trend, Cyclical, Seasonal and Irregular components. A brief description of these four components is provided below:

Trend: The general tendency of a time series to increase, decrease or stagnate over a long period is termed as Secular Trend or simply Trend. Thus, it can be said that trend is a long term movement in a time series.

Seasonality: Seasonal variations in a time series, it may manifest in the form of fluctuations within a year during the season or any other periodic unit of time.

Cyclical: The cyclical variation in a time series describes the medium-term changes in the series, caused by circumstances, which repeat in cycles. The duration of a cycle spans over a more extended period, usually two or more years.

Irregular: Irregular or random variations in a time series are caused by unforeseen influences, which are not regular and also do not replicate in a particular pattern. These variations are caused by incidences such as war, strike, earthquake, flood.

In time series forecasting, past observations are accumulated and analysed to develop a suitable mathematical model which arrests the underlying data generating process for the series. The future events are then predicted using the model. This approach is specifically useful when there is not much knowledge about the statistical pattern followed by the successive observations or when there is a lack of a satisfactory explanatory model. Time series forecasting has essential applications in various fields. Often valuable strategic decisions and precautionary measures

are taken based on the forecast results. Thus making a good forecast, i.e. fitting an adequate model to a time series is very important for the sake of acquiring accurate results.

A time series is non-deterministic, i.e. we cannot predict with certainty what will occur in future. The mathematical expression explaining the probability structure of a time series is termed as a stochastic process. Thus the sequence of observations of the series is a sample realisation of the stochastic process that produced it.

Stationarity is an important concept that is crucial to time series forecasting. The logic of stationarity of a stochastic process can be visualised as a form of statistical equilibrium. The statistical properties like mean and variance of a stationary process are independent of time. It is an essential condition for building a time series model which is useful for future prediction. Further, the mathematical complexity of the fitted model reduces with this assumption. The concept of stationarity is a mathematical idea constructed to simplify the theoretical and practical development of stochastic processes.

Technical Indicators:

Technical indicators are heuristic or pattern-based signals produced by the price, volume, and open interest of a security or contract used by traders who follow technical analysis. By analysing historical data, technical analysts use indicators to predict future price movements. Examples of standard technical indicators include the Relative Strength Index, Money Flow Index, Stochastics, MACD and Bollinger Bands.

Technical analysis is a trading discipline that is used to evaluate investments and identify trading opportunities by examining statistical trends collected from trading activity, such as price movement and volume. Unlike fundamental analysts, who attempt to evaluate stocks intrinsic value based on financial or economic data, technical analysis focus on patterns of price movements, trading signals and various other analytical charting tools to evaluate a security's strength or weakness.

There are two basic types of indicators :

Overlays: Technical indicators which use the same scale as prices are plotted over the top of the prices on a stock chart. Examples include moving averages and Bollinger Bands.

Oscillators: Technical indicators that fluctuate between a local minimum and maximum are plotted above or below a price chart. Examples include the MACD or RSI.

Often many different technical indicators are utilised when analysing stock data. With thousands of different options, the appropriate indicators must be chosen, indicators that work best for the situation and help in devising accurate results. Many different technical indicators may be combined with more subjective forms of technical analysis, such as looking at chart patterns, to emerge with trends and strategies. Technical indicators can also be involved in automated trading systems, given their quantitative nature.

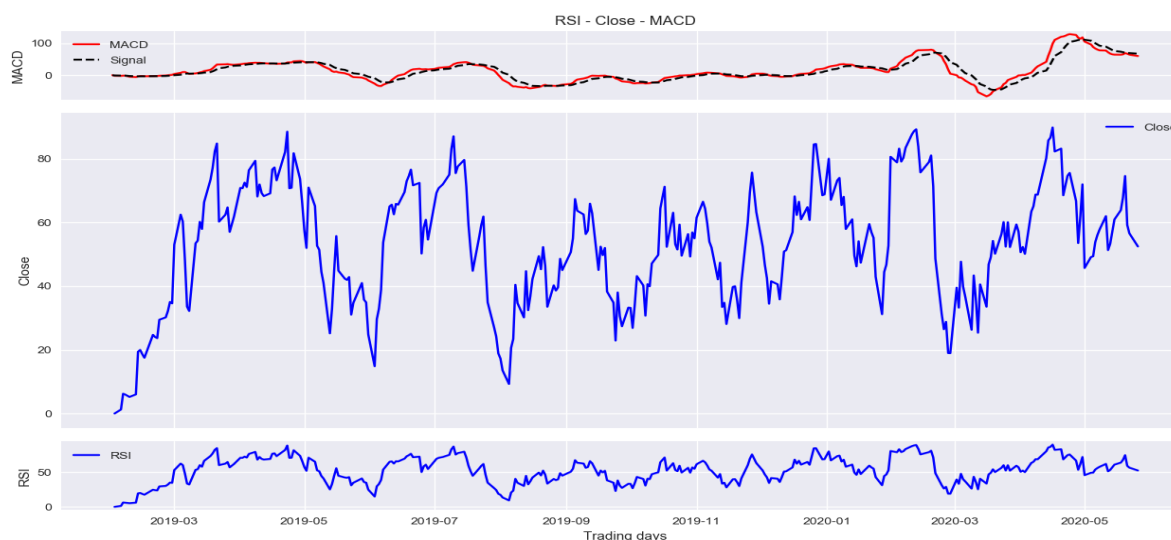


Figure 2.2 Plot of RSI-Close-MACD

Technical Analysis is a blanket terminology for a variety of strategies that are depending upon the interpretation of price action in a stock. Most technical analysis is concentrated on determining whether or not a current trend will continue and, if not when it will reverse. Some forms of analysis use trends for predictions, some use candlestick formations, and others prefer bands and boxes created through a mathematical visualisation. Most technical analysts use some a combination of tools to recognise potential entry and exit points for trades. A chart formation may indicate an entry point for a short seller, for example. However, the trader will look at moving averages for different periods to confirm that a breakdown is likely.

Before proceeding further, it is essential dwelling into surface-level concepts that currently dominate stock market trading and will be paramount in comprehending the technical indicators emulated in this project. It should be noted that this project is more focused on the programming and computer science aspect of analysis, and therefore most statistical and financial concepts mentioned in this report will only see a brief description, a simplified overview, instead of going deeper into the theory, which is a whole separate topic subject to multiple research papers.

When a stock is overbought, it implies that buying has pushed the price too high above and a reaction, called a price pullback, is contemplated. When a stock is oversold, the indication is that selling has pushed the price extremely low and a reaction, which is known as a price bounce, is expected. Just because stocks have gone up or down extremes does not mean that they still cannot change either way. Oversold and overbought stocks are often opinions that reflect someone's view of the market, which may or may not be accurate. A bull market is one that is on the rise and where the economy is sound, whereas a bear market exists in an economy that is receding, where most stocks are declining in value.

The indicators emulated in this project are as follows:

RSI: The relative strength index or RSI is a momentum index used in technical analysis that measures the recent price fluctuations quantitatively to calculate overbought or oversold conditions in the price of a stock or other asset. The RSI is displayed as an oscillator, in the form of a line graph that shifts between two extremes, and can have any reading from 0 to 100.

Conventional interpretation and usage of the RSI are that values of 70 or above indicating that a security is tending towards being overbought or overvalued and can probably be primed for a trend reversal or corrective pullback in the price—an RSI. Reading of 30 or below shows an oversold or undervalued condition—the RSI. Provides technical traders signals about bullish and bearish price momentum, and it is often plotted beneath the graph of a stock's price.

Formula:

First the price data is averaged and the average of the upward prices is divided by the average down price:

$$\text{Where RS} = \frac{\text{Smoothed Average* of 14 days Up closes}}{\text{Smoothed Average* of 14 days Down closes}}$$

The resulting product is then converted to a value between 1 and 100:

$$\text{RSI} = 100 - \left(\frac{100}{1 + \text{RS}} \right) \quad (3.1)$$

The formula above is commonly used to calculate RSI. The primary trend of the stock is an essential tool in making sure the indicator's readings are correctly understood. Analysts will commonly apply a horizontal trendline between 30% and 70% levels when strong trends are in place to better identify extremes and hence identify the overbought or oversold status.

Moving Averages: A simple moving average is an arithmetic moving average calculated by adding recent prices and then dividing that by the number of periods in the calculation average. Short-term averages respond quickly to changes in the price of the underlying, while long-term averages are slower to react. Moving averages may be calculated in various forms such as 50 day, 100 day and 200-day moving averages. 200-day moving averages are commonly employed as indicators in day to day stock trading. The different moving averages may even be plotted against each other, where every crossover depicts the signal to buy/sell shares.

$$\begin{aligned} \bar{p}_{\text{SM}} &= \frac{p_M + p_{M-1} + \cdots + p_{M-(n-1)}}{n} \\ &= \frac{1}{n} \sum_{i=0}^{n-1} p_{M-i}. \end{aligned} \quad (3.2)$$

MACD: MACD, which is the Moving Average Convergence Divergence is a trend-following momentum indicator showing the relationship between two moving averages of a security's price. The MACD is calculated by deducting the 26-period Exponential Moving Average (EMA) from the 12-period EMA.

The output of that calculation is the MACD line. A nine-day EMA of the MACD called the "signal line," is then plotted on top of the MACD line, which can function as a trigger for buy and sell signals. Traders may buy the stock when the MACD crosses above its signal line and

sell the stock when the MACD crosses below the signal line. Moving Average Convergence Divergence (MACD) indicators can be explained in many ways, but more common methods are crossovers, divergences, and rapid rises/falls.

$$\text{EMA} = \text{Price (t)} \times k + \text{EMA (y)} \times (1-k)$$

Where:

t = today

y = yesterday

N = number of days in EMA

$$k = 2 \div (N+1) \tag{3.3}$$

The formula (3.3) above shows the calculation of N day Exponential Moving Average. For calculating MACD, 26 day, 12 day and 9 day EMA data are calculated separately.

Bollinger Bands: A Bollinger Band is an analysis tool described by a set of trendlines plotted two standard deviations, positively and negatively, away from a simple moving average of a stock's price. Bollinger Bands are a highly popular technique. The common consensus is that the closer the prices move to the upper band, greater overbought the market is, and the closer the prices move to the lower band, the more oversold the market is. When the bands come near to each other, constricting the moving average, it is called a squeeze. A squeeze indicates a time frame of low volatility and is considered to be a potential sign of future increased volatility and probable trading opportunities.

Conversely, wider apart the bands move, more the probability of a decrease in volatility and higher the possibility of exiting a trade. However, these conditions are not absolute trading signals. The bands give no definite indication when the change may take place or which direction price could move.

Sentiment Index: A sentiment indicator refers to a graphical or numerical indicator for showing how a group feels about the market or economy. A sentiment index seeks to quantify how current beliefs and positions affect future behaviour. Sentiment indicators show how optimistic or pessimistic a group of people are, which may help forecast this group's future behaviour, often in a contrarian way. Like when investors are extremely bearish, that is often a contrary signal to sentiment indicator traders that market prices could start heading higher soon. The data is subject to interpretation. A high reading shows that consumers are upbeat. However, from a high reading, some feel it will likely head lower over time. A low reading shows consumers are disappointed; from there, things are likely to improve. These indicators are just one piece of data and are not meant to be a timing signal for taking action.

Predictive Models:

The Autoregressive Integrated Moving Average (ARIMA) Model:

ARIMA stands for Auto-Regressive Integrated Moving Average. A stands for a model that uses the dependent relationship between an observation and some number of lagged observations. I stands for Integrated. The use of differencing of raw observations in order to make the time series stationary. MA stands for Moving Average. A model that uses the dependency between observations and residual errors from a moving average model applied to lagged observations. Each of these components is explicitly specified in the model as a parameter. AR and MA are two commonly used linear models that work on stationary time series, and I am a pre-processing procedure to make the time series stationary if required.

A statistical model is autoregressive(AR) if it predicts future values based on past values. Autoregressive models implicitly take for granted that the future will mimic the past. Therefore, they can prove inaccurate under certain market circumstances, such as financial crises or periods of rapid technological change. They work under the premise that past values have an effect on current values, which makes the statistical technique accessible for analysing nature, economics, and other processes that vary over time. An AR(1) process has its current value based on the immediately preceding value, while an AR(2) process is one where the current value is dependent on the last two values. An AR(0) process is utilised for white noise and has no dependence between the terms. These techniques are used by technical analysts to forecast security prices. But since autoregressive models base their predictions only on previous data, they implicitly take for granted that the fundamental forces that influenced the past prices will not change over time. This can lead to surprising and inaccurate predictions if the underlying forces in question are changing, such as if an industry is undergoing a rapid and unprecedented technological transformation.

An ARIMA model is characterised by three variables: p, d, q where:

p is the order of the AR term

q is the order of the MA term

d is the number of differences required for making the time series stationary

The Autoregressive part of the ARIMA model is depicted below:

$$Y_t = \alpha + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + \dots + \beta_p Y_{t-p} + \epsilon_1 \quad (3.4)$$

Where Y_{t-1} is the lag 1 of the series, β_1 is the coefficient of lag1 that the model predicts and α is the intercept term, also predicted by the model.

A pure Moving Average model is one where Y_t depends only on the lagged forecast errors, as depicted below:

$$Y_t = \alpha + \epsilon_t + \phi_1 \epsilon_{t-1} + \phi_2 \epsilon_{t-2} + \dots + \phi_q \epsilon_{t-q} \quad (3.5)$$

Where error terms are the errors of the autoregressive models of the respective lag.

In an ARIMA model, the time series is differenced at least once to make it stationary, followed by a combination of the AR and the MA model. The equation is depicted below:

$$Y_t = \alpha + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + \dots + \beta_p Y_{t-p} + \epsilon_t + \phi_1 \epsilon_{t-1} + \phi_2 \epsilon_{t-2} + \dots + \phi_q \epsilon_{t-q} \quad (3.6)$$

Equation (3.6) is interpreted as

Predicted Y_t = Constant + Linear combination Lags of Y (up to p lags) + Linear Combination of Lagged forecast errors (up to q lags)

Long Short Term Memory (LSTM) Model:

LSTM is a recurrent neural network (RNN) architecture that remembers values over arbitrary intervals. It is well-suited to classify, process and predict time series given time lags of unknown duration. Relative insensitivity to gap length gives an advantage to LSTM over alternative RNNs, hidden Markov models and other sequence learning methods. LSTM, in a nutshell, is an RNN with a better recurring unit.

The idea behind all gated RNN architectures is to make the network hang on to essential memories from many time steps ago, capture long-distance dependencies, and allow error messages to flow at different strengths based on the input. RNNs are a broad category of neural networks with feedback loops and are capable of “remembering” and using information about previous inputs.

LSTM’s ability to erase, write and read information is what helps it alleviate, to a degree, the impact of the vanishing/exploding gradients and hold on to information over many time steps (something vanilla RNNs struggle with). In vanilla RNNs, we update hidden state each time a new input is being processed, no matter how insignificant it might be to the overall prediction. LSTM, however, can decide what information to throw out. To understand why LSTM has become the key player in the field of RNNs, we need to understand how neural networks are trained. Most of the networks are trained through a variant of gradient descent which is an iterative process. The process comprises of steps as following- calculating the output or forward propagation, followed by measuring the error from the output, followed by using the error to calculate the gradient (like by using backpropagation) and finally repeating all these steps multiple times with different outputs.

But unfortunately, the gradient in deep neural networks is unstable. As earlier gradients are the product of later gradients, they tend to either increase or decrease exponentially. This is called the vanishing/exploding gradient problem. This problem with unstable gradients is much worse in RNNs. This is because when we train RNNs, we are not just calculating the gradient through all the different layers, but also through time. Thus, although “vanilla” RNNs should theoretically be able to use information from the distant past, they have been unable to do so in practice (like the case of remembering a word in a previous sentence). Now let us see the scenario when LSTM comes into the picture. LSTMs introduce the concept of cell states, which provide “highways” for the gradient to flow backwards through time freely, thereby making it more resistant to the vanishing gradient problem. The cell state can be thought of almost like data stored in a computer’s memory. LSTMs can “remember” or “forget” information in the cell state by using specialised neurons called “gates”. This way, LSTMs can retain long-term dependencies and connect information from the past to the present. There are three gates:

1. forget gate: decide what information from previous inputs to forget
2. input gate: decide what new information to remember
3. output gate: decide which part of the cell state to output (this is what we see coming out of the LSTM)

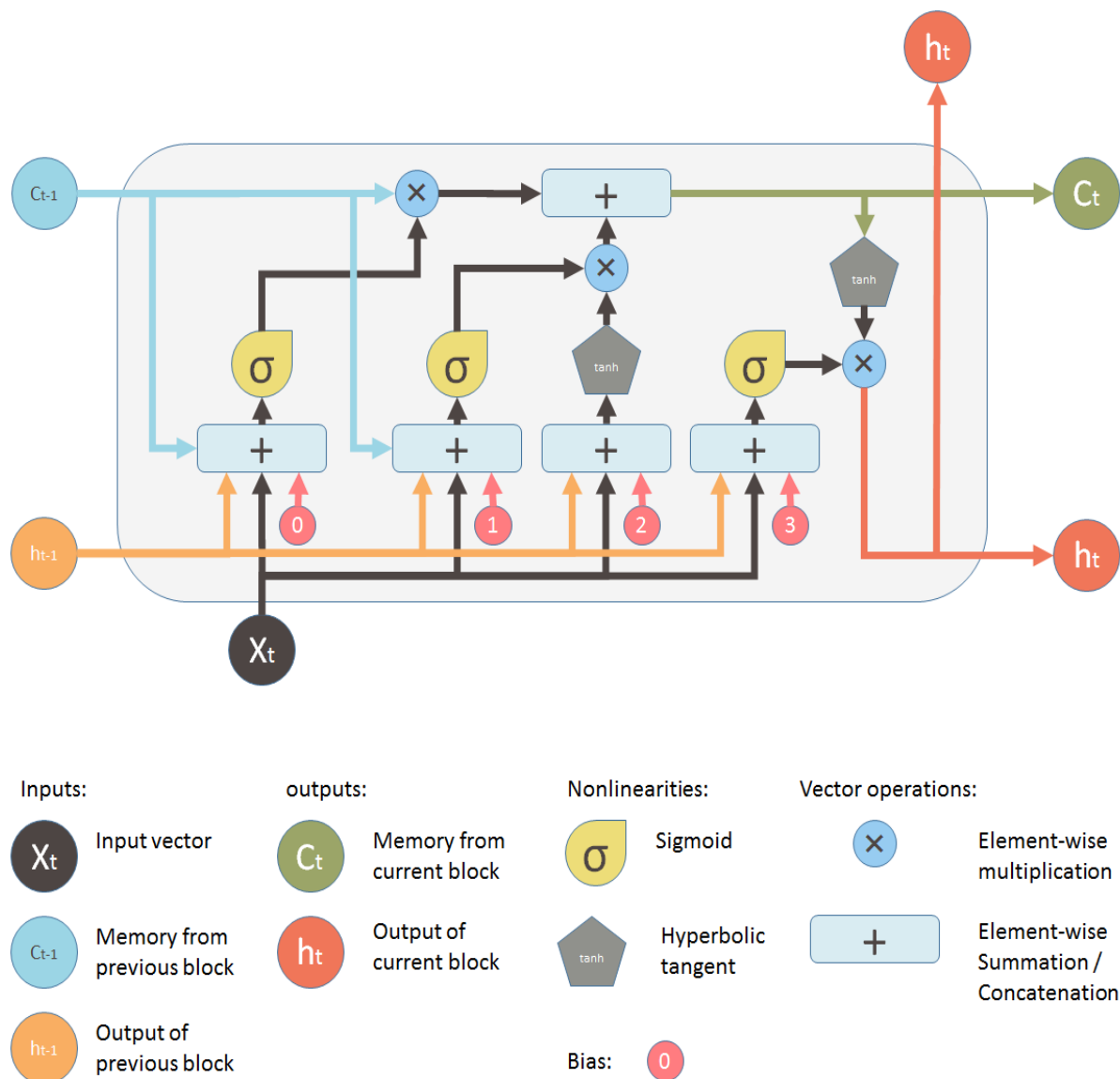


Figure 2.3 Basic depiction of an LSTM unit

The math for backpropagation (measuring the gradient) in LSTMs is much more complicated. However, it conveniently works out such that information in our cell state can be unaffected by the vanishing gradient problem. Thus, LSTMs much more resistant to vanishing gradients.

CHAPTER 3

METHODOLOGY

3.1 Introduction:

This section deals with the working of the code and the methods used to achieve the results obtained so far. The code was written in Python(version 3.xx) primarily for the vast number of libraries that make such an analysis less taxing and the machine learning emphasis that is the speciality of this language on account of the functions, libraries and frameworks available to finetune results and achieve maximum accuracy. It should be noted that the first part of the project, up until the building of the prediction models, uses Amazon stock history as its dataset at which point Microsoft is used to depict the differences in the machine learning methods.

3.2 Methodology:

This section details the process used to construct the technical indicators and prediction models utilised in this report. The section has been split into sub-topics, namely, Web scraper, Technical Indicators and Prediction models to distinguish the various parts of the project according to the functions they perform. The Data Extraction section deals with the construction of the web scraper that was used to extract the data that forms the basis of this project, the Technical Indicators section details the various indicators emulated and a summary of the work that went into programming and calculating required parameters and the Prediction models section deals with the building and explanation of the machine learning models constructed in this project.

Data Extraction:

The first part of the project deals with the extraction of past stock data for a given company from a financial website for the purpose of providing the base upon which prediction models are to be trained and built. The financial website in question is required to be accurate, reputable and updated on a daily basis to provide maximum data for usage, and for these purposes, Yahoo Finance was selected as the source of all the information regarding stock data of the companies. The extraction of relevant data was facilitated by the construction of a Web Scraper. Web scraping or web harvesting refers to data scraping used to extract all the relevant information from a website. Data scraping from Yahoo Finance was achieved through the means of the `pandas_datareader` library in Python, which provides functions directly sourced from the Yahoo Finance API to make the extraction a much more efficient process. The data is extracted as it is hosted on Yahoo Finance, with all its parameters exported in the same form into an excel sheet. First, a connection to the website is established, then the relevant page, in this case, historical stock data for the company, is accessed. Finally, all of that data is taken and exported to an excel sheet with the `to_excel()` function provided as part of the Pandas library. The program allows the user to enter the name of any company, and after receiving input in parses Yahoo Finance for all the historical data available.

```
Enter company name:
Amazon
Downloading stock data for company...

Process finished with exit code 0
|
```

Figure 3.1 Depiction of terminal UI in the program

Figure 3.1 depicts the process of downloading the data. The code makes use of these functions to parse the website and extract all the significant parameters that will influence the accuracy of the prediction models, namely Opening Price, Highest Price, Lowest Price, Closing Price, Volume of Stocks traded and Adjusted Closing Price. All this information is readily available for viewing on the website itself; the code simply extracts all the information under these parameters and exports them to an excel file stored in the relevant directories. Figure 11 depicts the data as extracted by the web scraper from Yahoo Finance, from 1999 to 2020, along with all the required parameters. A short exploratory analysis of the dataset reveals that it contains no empty cells or junk values, which means it does not need to undergo any post-extraction processing and is already in a format conducive to further analysis.

	A	B	C	D	E	F	G	H
1	Date	High	Low	Open	Close	Volume	Adj Close	
2	31-12-1999	79.375	76	79.3125	76.125	7270700	76.125	
3	03-01-2000	89.5625	79.04688	81.5	89.375	16117600	89.375	
4	04-01-2000	91.5	81.75	85.375	81.9375	17487400	81.9375	
5	05-01-2000	75.125	68	70.5	69.75	38457400	69.75	
6	06-01-2000	72.6875	64	71.3125	65.5625	18752000	65.5625	
7	07-01-2000	70.5	66.1875	67	69.5625	10505400	69.5625	
8	10-01-2000	72.625	65.5625	72.5625	69.1875	14757900	69.1875	
9	11-01-2000	70	65	66.875	66.75	10532700	66.75	
10	12-01-2000	68	63	67.875	63.5625	10804500	63.5625	
11	13-01-2000	67.1875	63.125	64.9375	65.9375	10448100	65.9375	
12	14-01-2000	68.625	64	66.75	64.25	6853600	64.25	
13	18-01-2000	65.1875	63	63.4375	64.125	5384900	64.125	
14	19-01-2000	67.5	63	64.125	66.8125	8245500	66.8125	
15	20-01-2000	67	63.9375	66.9375	64.75	5978000	64.75	
16	21-01-2000	64.625	60	64.625	62.0625	11461900	62.0625	
17	24-01-2000	73.375	67.5	67.5625	70.125	29170200	70.125	
18	25-01-2000	71.25	66	70	69.25	9434100	69.25	
19	26-01-2000	70	64.75	68.625	64.8125	6558000	64.8125	
20	27-01-2000	67.75	64.625	65.1875	66.9375	6784000	66.9375	
21	28-01-2000	66.4375	60	65	61.6875	13777900	61.6875	
22	31-01-2000	64.75	58.4375	60.375	64.5625	10697900	64.5625	
23	01-02-2000	70.625	64.375	67.5	67.4375	13404600	67.4375	
24	02-02-2000	72.25	67.75	67.9375	69.4375	14025900	69.4375	
25	03-02-2000	85.9375	77.375	81.125	84.1875	43750000	84.1875	
26	04-02-2000	82.75	77.875	82.75	78.5625	11023000	78.5625	
27	07-02-2000	76.875	73.125	76.1875	75	10129000	75	
28	08-02-2000	84	73.4375	74	83.125	19472800	83.125	
29	09-02-2000	83.73438	80.01563	80.54688	80.25	9580400	80.25	
30	10-02-2000	79.875	75.5	78.5	76.1875	9979600	76.1875	
31	11-02-2000	79.125	75.5	77.125	76.1875	8309000	76.1875	

AMZN
+

Ready

Figure 3.2 Data extracted by web scraper

Figure 3.2 depicts the data as extracted from the website. It should be noted that the data is already available in a clean and processed format, eliminating any need for pre-processing. The function also takes in starting date and ending date as its parameters, this refers to the time period during which the web scraper is supposed to extract all the information in. For the purpose of this project the starting date was set as 31st December 1999 and the ending date was set as 5th March 2020, to provide the maximum amount of data possible with regards to model construction, since more information correlates to more accuracy.

Construction of Technical Indicators:

The second part of this project deals with the construction and emulation of the aforementioned technical indicators using the data extracted. These technical indicators are dependent on

calculations performed on the extracted parameters, generating new variables that are then plotted to depict trends that could be used in analysing the future movement of the stock price. This is facilitated entirely by the use of the Matplotlib library available in Python. Matplotlib allows in-depth customisation of plots with the various functions available in its library, which were extensively made use of when plotting the indicators

- **Stock Trend:**



Figure 3.3 Plot of Stock Trend for Amazon

Figure 3.3 depicts the historical trend in Amazon's closing price over a period of 20 years. The trend is plotted by passing the Closing Price over the years and the dates into the plot function provided by Matplotlib as x-axis and y-axis values respectively. The graph type is set to line, and upon running the function, Fig 11 is plotted on the console screen.

Overall the trend seems to be irregular, but the graph depicts a definite trend. This makes the data non-stationary in nature. Another insight that can be gathered from looking at the trend is the sudden uptick in March 2020. This phenomenon is testament to the fact that a stock's closing prices are influenced entirely by outside factors and hence it is extremely difficult to develop a model that can account for anomalies like this since most models operate on the presumption that the future values of the stock's price are dependent on their past prices, and even though a relationship can be observed between past and future values, the prices are completely independent of their previous values and are instead only dependent upon the transactions of the traders and analysts involved, which is a factor outside the control of any machine learning model.

■ Candlestick Chart:

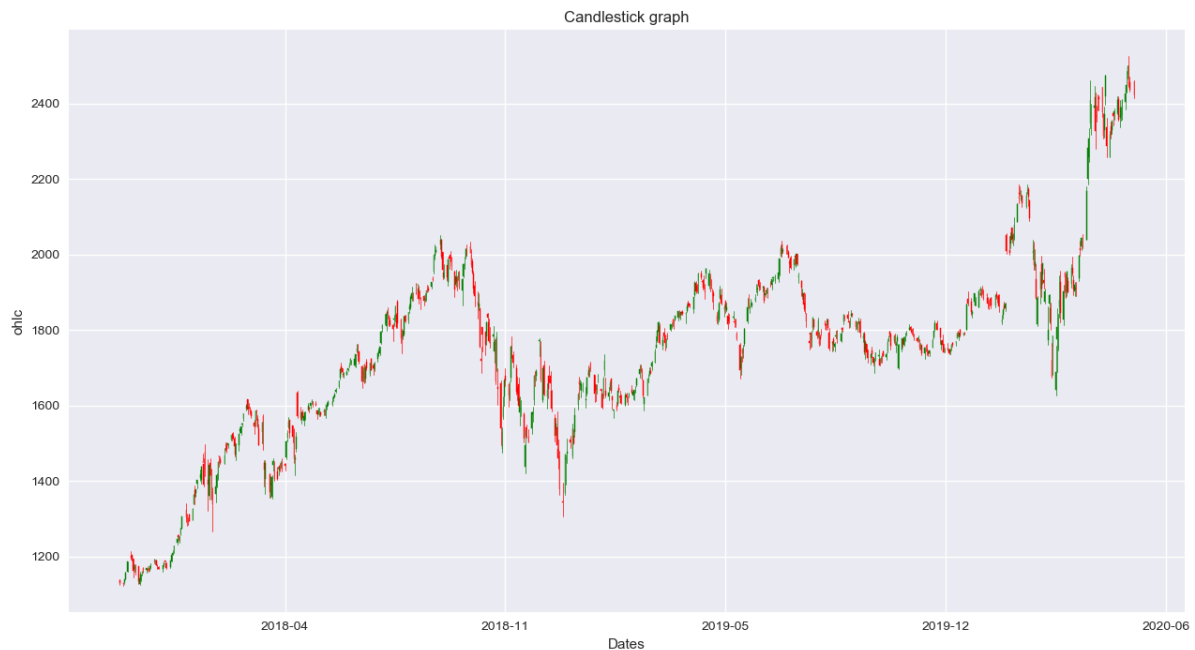


Figure 3.4 Candlestick graph- type 1

Figure 3.4 depicts a Candlestick graph. A Candlestick graph is a style of financial chart used to describe price movements of a stock or currency. The candlestick graph is plotted by an entirely separate function provided as part of the Matplotlib.finance library which takes in the Opening Price, Closing Price, High and Low values and returns the corresponding candlestick plot, which is then ported to the console. In the above-plotted graph, each “candlestick” typically shows one day, with all the relevant stock information, namely, Open High Low and Close depicted in a single candlestick. Candlestick charts can also be built using intervals shorter or longer than one day.

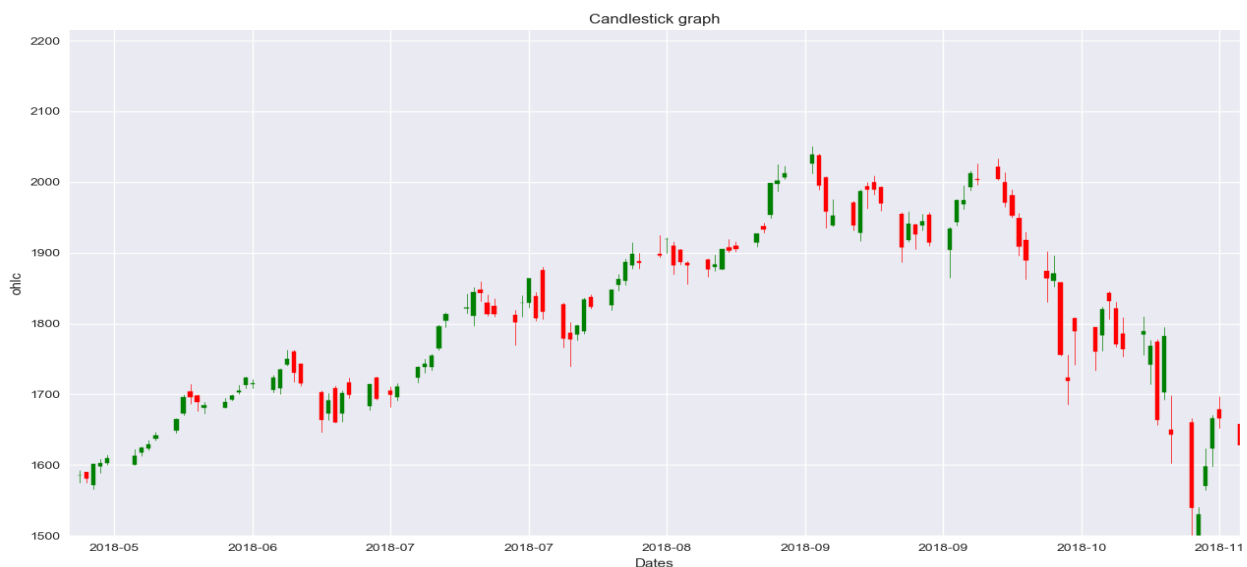


Figure 3.5 Candlestick graph- type 2

Figure 3.5 depicts a zoomed-in version of Figure 3.4. As is visible in the graph, each candlestick displays a condensed version of stock parameters, and the colour scheme displays the comparison of the next day's closing price with the previous day closing price. In this way, a candlestick chart manages to display much information in its plots and is therefore highly utilised by traders looking for a brief and quick overview regarding a company's stock performance.

▪ Moving Averages:

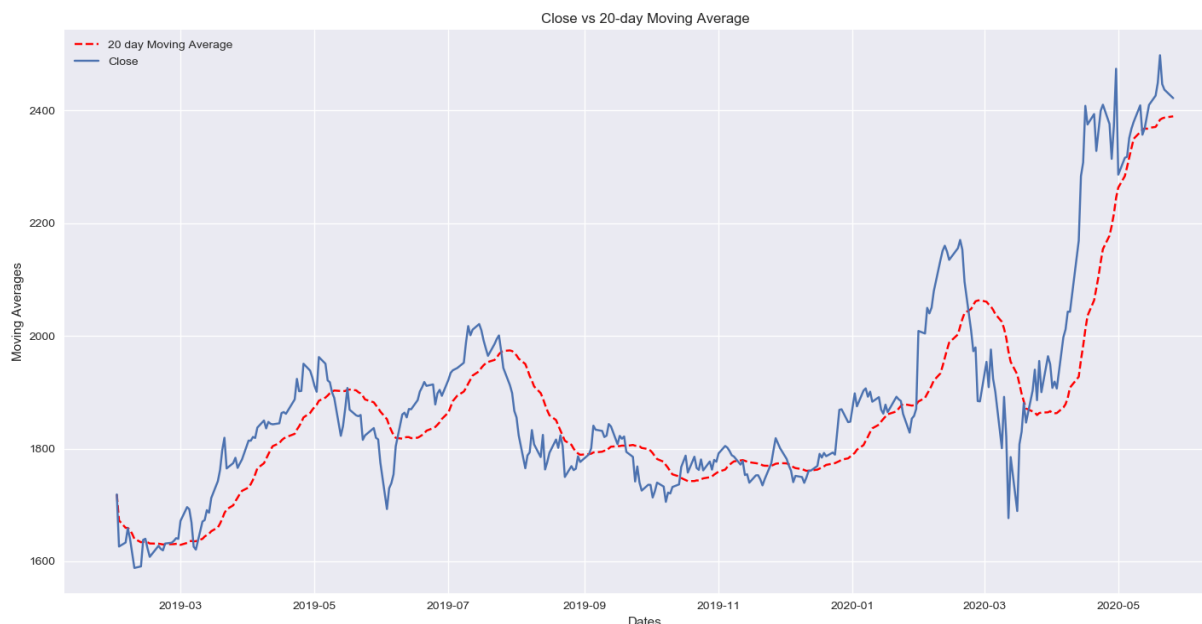


Figure 3.6 Plot of 20-day Moving Average vs Closing Price

A moving average (MA) is a technical indicator that smooths out price trends by filtering out the “noise” from random short-term price fluctuations. Figure 3.6 depicts a 20-day moving average plotted around the closing price of Amazon's stock. Moving averages with a smaller window (20 day MA) will react faster to changes in the price as opposed to moving averages with larger windows (200 day MA). The Close line crossing over the MA line during a downtrend is generally interpreted as a signal of the stock is overbought. In contrast, the crossing over of the Close line over the MA line in an uptrend is generally accepted as a sign of the stock being oversold. Another interpretation of the graph is that if the close is above the MA, the price is up and vice versa for the price being down.

To plot a moving average, it is first calculated as an entirely new parameter with the help of the Closing Price. The rolling function provided by pandas is used in conjecture with the mean function to set a window of days (N) and calculate the moving average for those corresponding days. This value is then exported to the plot function along with the Closing price, which is then graphed and displayed on the console screen.

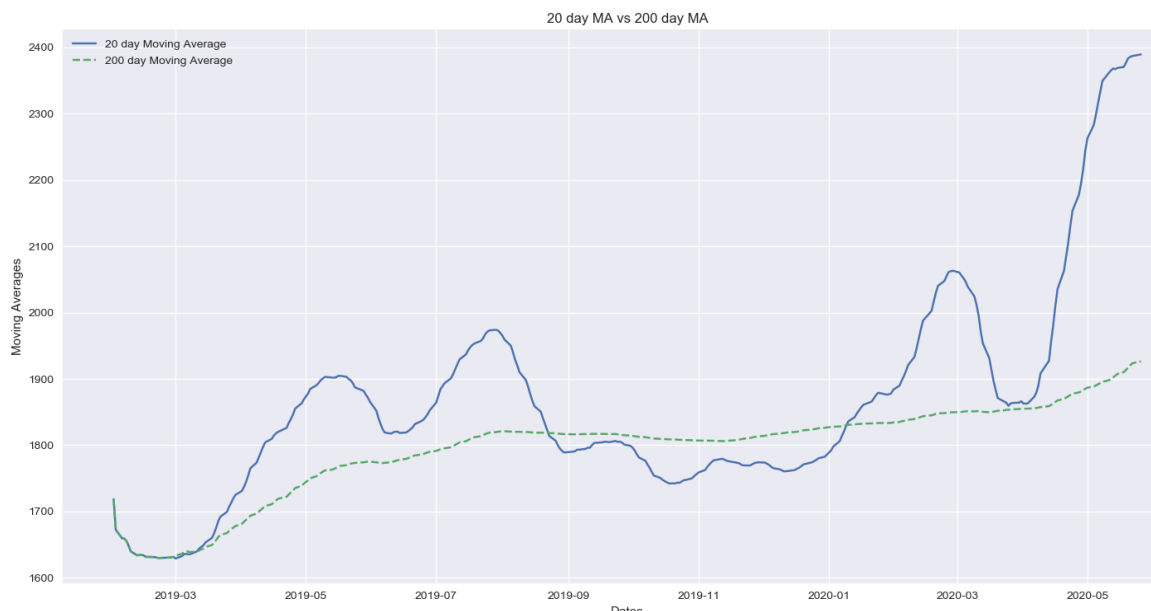


Figure 3.7 Plot of 20-day vs 200-day Moving averages

Figure 3.7 depicts the 20 day vs 200-day Moving average graph. It is a popular technical indicator in the stock market ecosystem and is used to predict possible trends of the closing price. While most of these indicators are open to interpretation, it is generally accepted that the 20 day MA being above the 200 days MA is a sign of the price going up and a signal to buy.

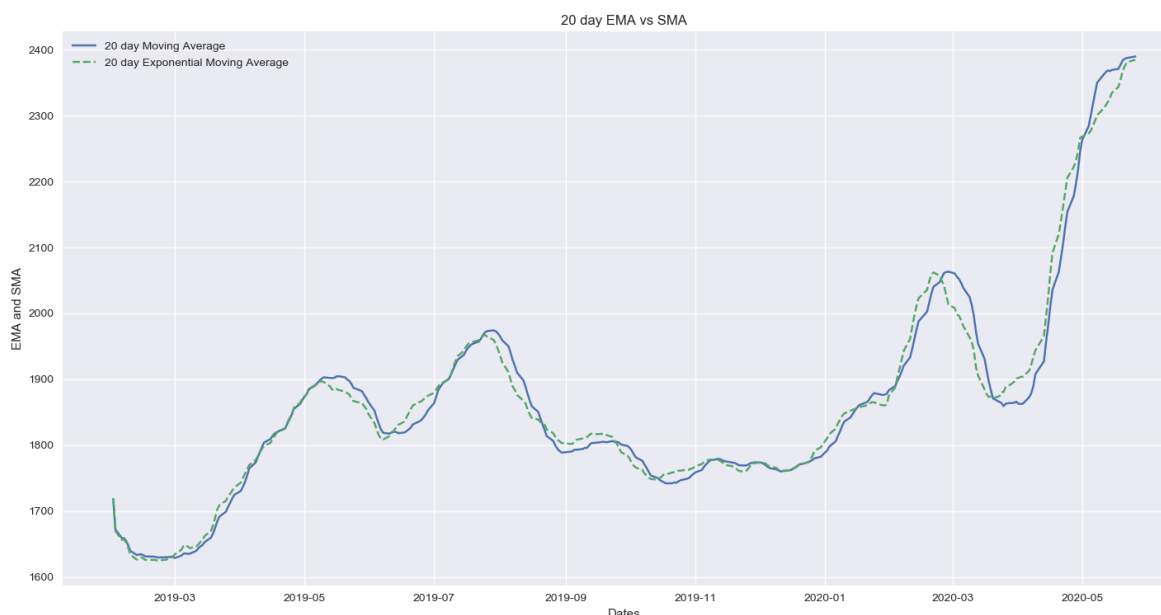


Figure 3.8 Plot of 20 day EMA vs 20 day SMA

The SMA vs EMA (Exponential Moving Average) was plotted for the sole reason of observing their reactions to the closing price trend. Both these indicators enjoy heavy usage in the stock market ecosystem, and they both have their tradeoffs. Exponential moving averages have lower lag and are hence more sensitive to recent prices - and recent price changes. Exponential moving averages will, therefore, turn before simple moving averages. Simple moving averages, meanwhile, represent an actual average of prices for the entire period. They may be better suited to identify support or resistance levels.

▪ Relative Strength Index (RSI):

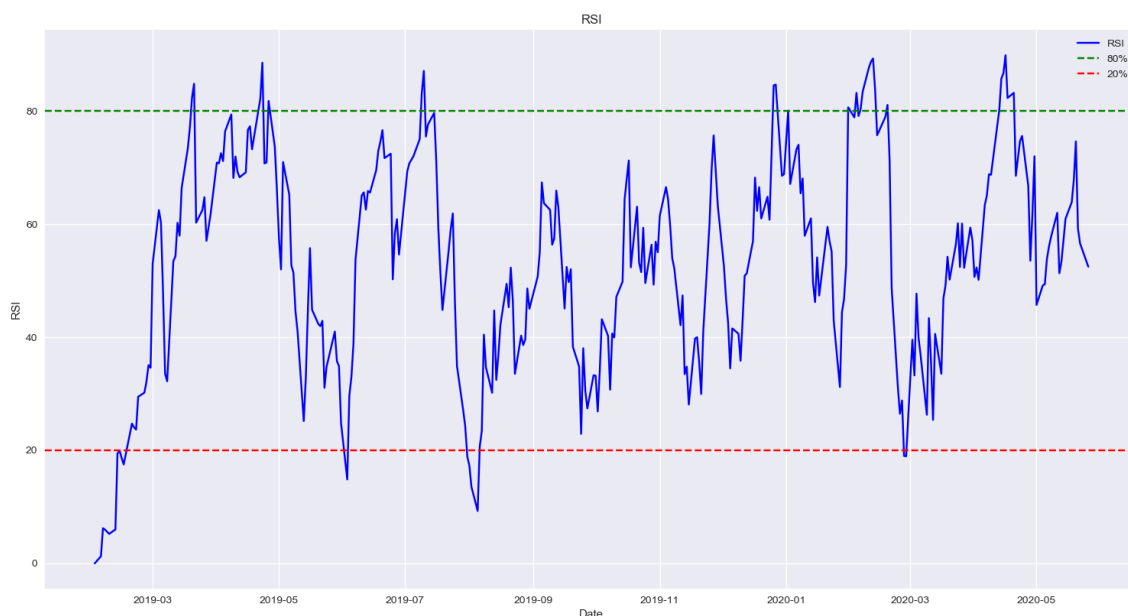


Figure 3.9 Plot of RSI for Closing Prices

Figure 3.9 displays the RSI for the company stock data. The relative strength index (RSI) is benchmark measuring the degree of recent price changes to find overbought or oversold conditions in the price of a stock or other asset. An asset is considered overbought when the RSI is above 70% and oversold when it is below 30%, though for this analysis threshold levels of 80 and 30 were selected.

The calculations for the RSI are enclosed within a custom function that is called only when it needs to be graphed on to the console. The function uses the Closing Price as its base, calculates the difference using consecutive values, segregates these values on the basis of them being positive or negative, and finally calculates the 14 day mean, using the rolling function. RSI is calculated as its own parameter, which is then used to produce a plot on the screen with the appropriate threshold values overlaid on it.

The basic trend of the stock or asset is an essential tool in making sure the indicator's readings are adequately understood. When the RSI values cross the 80 threshold, the stock is considered overbought, and when the RSI values cross the 30 threshold, the stock is considered oversold.

- **Moving Average Convergence Divergence (MACD):**

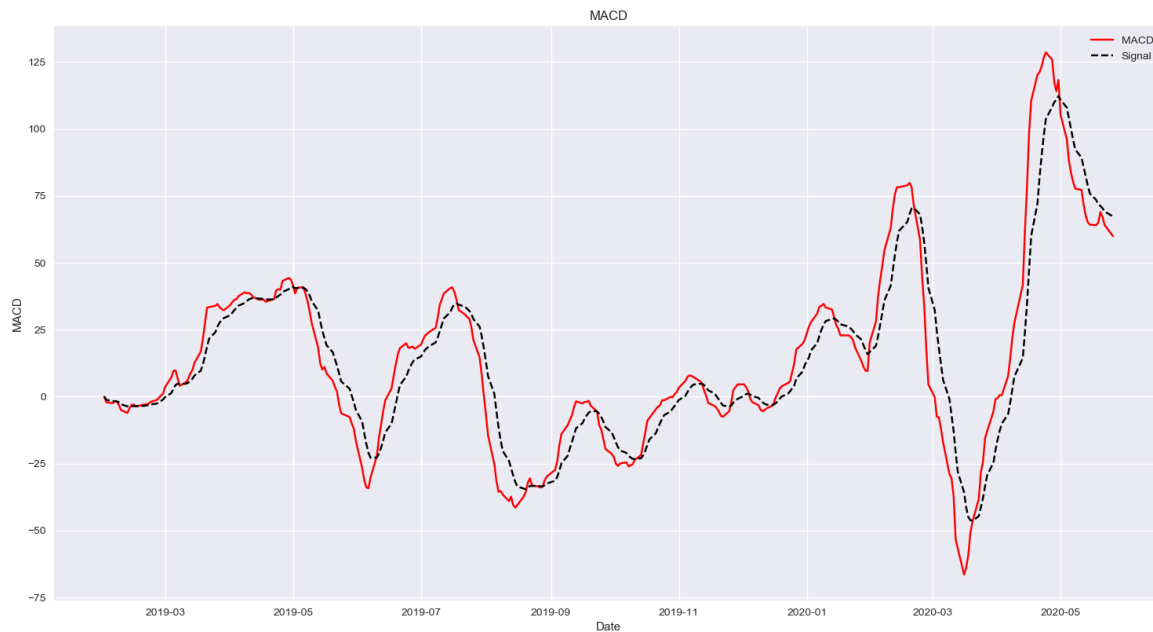


Figure 3.10 Plot of MACD for Closing Price values

Figure 3.10 depicts the MACD plot for the stock. Moving Average Convergence Divergence (MACD) shows the relationship between two moving averages of a security's price. The MACD is calculated by deducting the 26-period Exponential Moving Average (EMA) from the 12-period EMA. The value of that calculation is the MACD line. A signal line which is the nine-day EMA of the MACD is then drawn on top of the MACD line, which can function as an indicator for buy and sell signals. The MACD has a positive value whenever the 12-period EMA (blue) is on top of the 26-period EMA (red) and a negative value when the 12-period EMA is below the 26-period EMA.

The calculation of the MACD is also outsourced to its own function. MACD is heavily dependent on the exponential moving average of the closing price parameter; therefore all the appropriate N-day exponential moving averages are calculated via the rolling function and the `ewm()` function provided by the pandas library. These values are attached as their own parameters to the data and then used further to calculate the MACD value for each day. Similarly, the Signal values are calculated and stored as a list for further use. Finally, both these values are simultaneously graphed on the console screen to depict the MACD plot.

- **RSI-Close-MACD plot:**

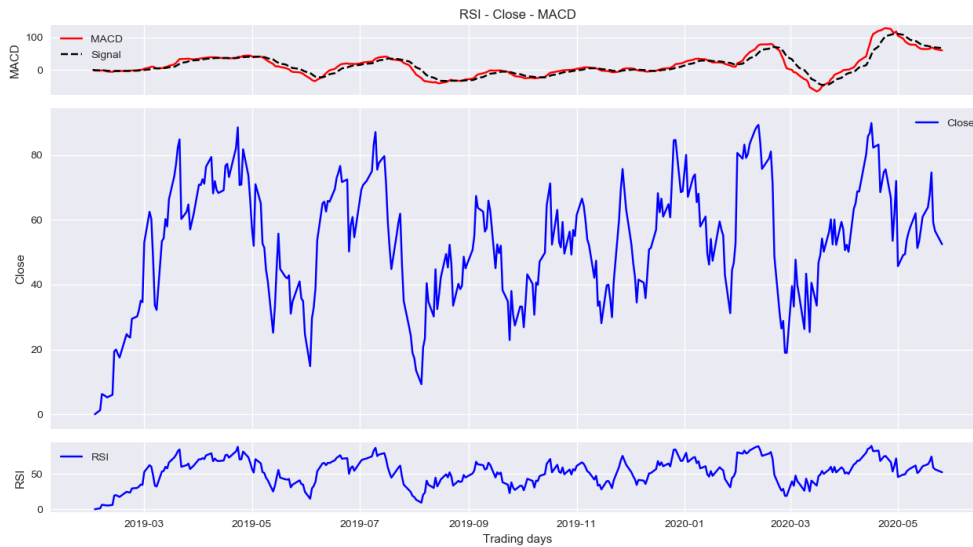


Figure 3.11 Plot of RSI-Close-MACD

Figure 3.11 shows the most common depiction of RSI and MACD, used as indicators alongside the Closing price.

The relative strength indicator (RSI) aims to indicate whether a market is considered to be overbought or oversold concerning recent price levels. It is an oscillator that calculates average price gains and losses over a given period; the default time frame is 14 periods with values ranging from 0 to 100.

MACD measures the relationship between two EMAs, while the RSI measures price change concerning recent price highs and lows. These two parameters are often used together to provide analysts with a comprehensive scene of a market.

Matplotlib provides various utilities that allow customisation of the plots being displayed. The program takes advantages of those utilities, such as customising the number of plots on the console screen, setting a common x-axis for all the plots being graphed separately etc. to display the plot in Figure 17.

▪ Bollinger Bands:

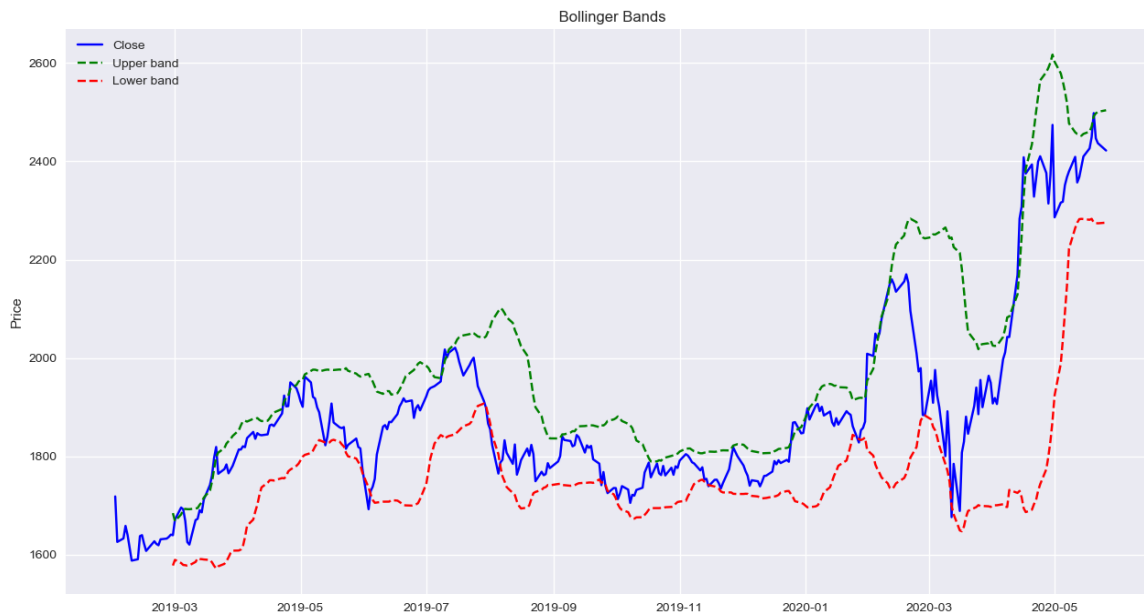


Figure 3.12 Plot of Bollinger Bands

Figure 3.12 displays the Bollinger Bands for the stock. Bollinger Bands consist of a centre line and two price channels (bands) above and below it. The centre line is the closing price; the price channels are the standard deviations of the stock being studied. The bands expand, and contracts as the price action of stock become volatile or become bound into a tight trading pattern. Upper resistance and lower support lines are initially drawn and then extrapolated to form channels within which the prices are expected to be contained. The general consensus is that nearer the prices move to the upper band, the greater overbought the market, and the nearer the prices shift towards, the lower band, the more oversold the market.

Bollinger bands required the calculations of the upper and lower bands separately so as to allow them to be graphed alongside the closing price. These bands were calculated by taking the standard deviation of the 20 day moving average that was attached as a user-defined parameter. Pandas allow these calculations performed for individual values to be reflected across the dataset and so this was leveraged to quickly produce a new column of values that could be graphed later on.

▪ Sentiment Index:

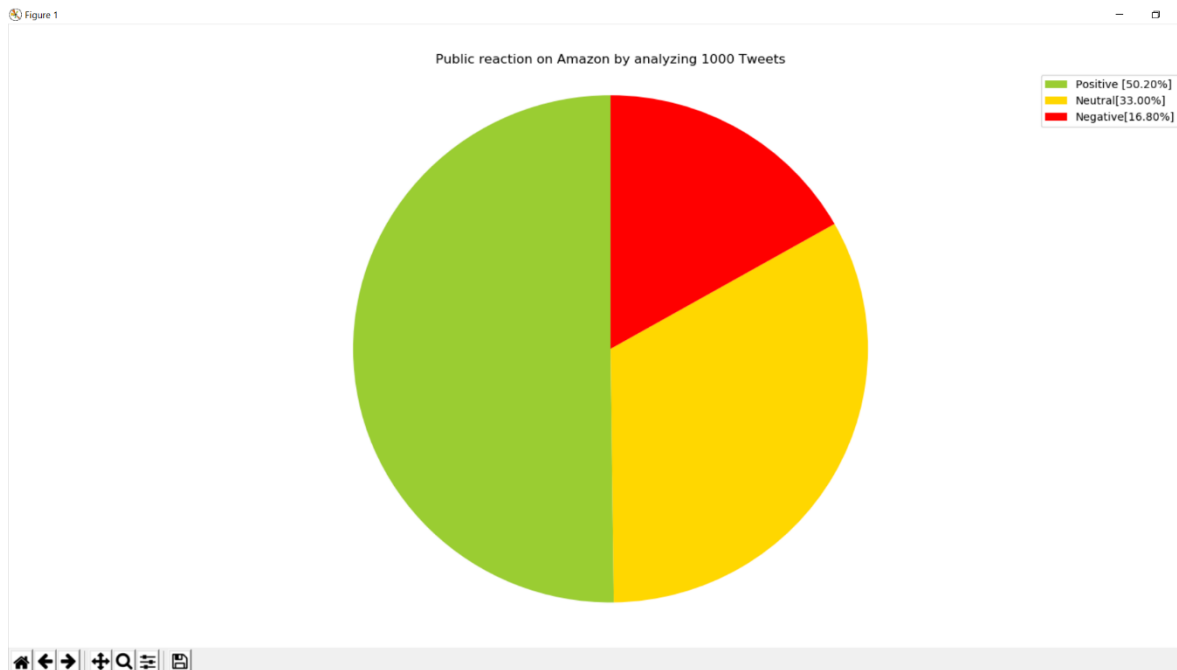


Figure 3.13 Pie Chart of Sentiment Index

The above sentiment index is extracted from analysing a thousand tweets over a seven day period. The purpose of conducting a Sentiment Analysis is purely to test the effect of the public mood on stock prices. The sentiment index is an important indicator as the day to day trading depends heavily on the trading of the players involved and an insight into their sentiments regarding a company's financial situation can serve as an additional indicator to be interpreted with various others. The code utilised in this project makes use of Twitter API keys to access the website and begin extraction of the data by importing the tweepy library and using the OAuthHandler function within it to automate the authentication part of the process. The TextBlob function from this library will break down the tweets into a number of words. This process is referred to as Tokenisation. The code makes use of a loop to parse through all the tweets stored in the tweepy object and runs the TextBlob function to break down each tweet into its word component. After Tokenisation, every word in each tweet is compared to the xml file holding all the required polarity values in the library and based on the words included in the tweet, calculations are performed, and a polarity value is generated. Finally, all these values are then graphed into a pie chart to demonstrate the aggregated result of public sentiment with regards to company stock.

Prediction Models:

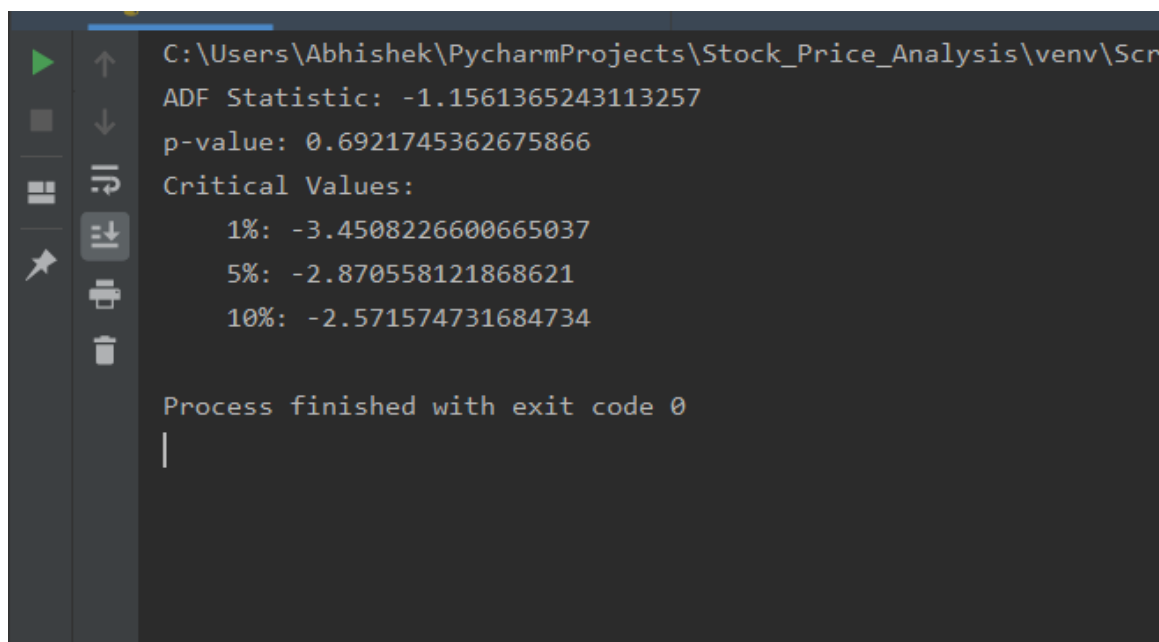
ARIMA:

As mentioned previously, predicting stock price on the basis of historical stock data falls under the domain of time series forecasting. To apply the ARIMA statistical model for predictive purposes, it must first be ensured that the time series is stationary in nature. To check the stationarity of the series, a simple graph depicting the rolling mean and standard deviation of the data is plotted in Figure 3.14.



Figure 3.14 Plot of Standard and Rolling Mean

As it is evident from the graph, the mean and the standard deviation show irregular trends, proving that the data is non-stationary in nature. The Dicky Fuller test is another statistical test that checks the stationarity of the data in question. The results of applying the test on the dataset are depicted in Figure 3.15.



```
C:\Users\Abhishek\PycharmProjects\Stock_Price_Analysis\venv\Scr
ADF Statistic: -1.1561365243113257
p-value: 0.6921745362675866
Critical Values:
    1%: -3.4508226600665037
    5%: -2.870558121868621
   10%: -2.571574731684734

Process finished with exit code 0
|
```

Figure 3.15 Results of Dicky Fuller test

In statistics, the Dickey-Fuller test tests the null hypothesis that a unit root is present in an autoregressive model, making it non-stationary. The alternative hypothesis is that there is no unit root present, making the data in question stationary.

The Dicky Fuller test is performed with the help of the `adfuller()` function provided as part of the `stats.model` library in Python. This function returns a list of values, which are then formatted to display their corresponding information. The formatted representation of the Dicky Fuller test is displayed in Figure 18.

The p-value of this test should be below 0.05, which would prove that the null hypothesis is false, and the data is indeed stationary. After running the test on Amazon's historical stock data, a p-value of 0.69 is received, meaning that the dataset is non-stationary in its present form. Furthermore, the value of the ADF statistic is higher than the Critical values, cementing the non-stationary status of the dataset.

A non-stationary dataset is not conducive to an accurate prediction model on account of the noise present in the dataset. To apply the ARIMA model, the dataset needs to be converted into a stationary form. One of the most used ways of converting a non-stationary series into stationary series is taking the difference between the values at $t+1$ and t , which removes most of the noise and gets rid of the trends in the moving average. Another common way is by taking a log of the mean; the trends are forced to become stationary. Sometimes both these operations are performed together to acquire the desired results. To build the ARIMA model, the dataset in question was differenced to get rid of the non-stationarity present in it.

After the necessary operations were performed, the dataset is ready to be fed into the ARIMA model. The ARIMA model relies upon the parameters of p , d and q as mentioned before. Here, since the dataset has been differenced once already, the d value is set to 0, as there is no need for it to be differenced again. There are several ways to tune the p and q values for ARIMA, the most common being hyperparameter analysis, where the ARIMA model is repeatedly executed

with several p and d parameters by means of a loop, and the most favourable parameters are picked for prediction. However, for the purpose of this project, a trial and error based approach was adopted, since stock data tend only to have a few p and q values appropriate for it as proved by research papers published before. For this model, $p=1$ $d=0$ $q=1$ were the values chosen and fed into the ARIMA function. The dataset was divided into a train and test set, in the ratio of 0.7 to 0.3, to provide maximum values to the ARIMA model for accuracy. Once the model is built upon the train set, it is fitted onto the test set using the `.fit()` function, and the resulting plot detailing the true and predicted values are graphed on the screen.

LSTM:

In LSTM, both the historical data and the current data is used to make a prediction. This is attained by the use of a memory cell, where all the relevant past information is stored. Since the memory cell is controlled by three gates, LSTM solves the problem of long term dependencies, i.e. LSTM performs well in cases where information way back in the past is needed to make the future prediction. Hence LSTM generates outputs taking into account the previous data as well as the current data, thereby giving it a sense of time context and making more accurate predictions. For stock market prediction, the data will be a time series. In order to make accurate predictions, the past events should be taken into account as events happening in the past affect the present. All the relevant past information and events will be stored in the LSTM memory cell and taken into account in addition to the current data points, leading to “contextual” and more accurate predictions.

LSTMs and other recurrent network architectures are good at finding relationships between consecutive data points, often over varying lengths of timeframes. They can identify patterns resulting from seasonality very well. They have an advantage over other regression models as they always look at recent past data, compared to, let us say, a random forest regressor, which works well with categories in the data and can detect some seasonality, but overall looks at one row at a time independent of the previous points.

We first load the dataset, perform exploratory data analysis on it following which we visualise the same plotting it on a graph. We try to create an imaginary funnel for the supply of input data. This can be thought of all the different types of input data combined in a single pathway, uniformly. This process involves Normalisation and standardisation.

For Normalisation, we need to define a scalar to define the data; hence we use the `MinMaxScalar`.

```

array([[0.13770717, 0.82599163, 0.78349733, 0.7503209 ],
       [0.17230424, 0.76945686, 0.81983423, 0.8074455 ],
       [1.0000001 , 0.89574146, 0.9326272 , 0.90885735],
       [0.23536322, 0.824523 , 0.9031043 , 0.82605886],
       [0.15407464, 0.9544792 , 0.9182439 , 0.92426157],
       [0.09631124, 0.93025017, 0.91521597, 0.92361975],
       [0.09176993, 0.95301056, 0.9038615 , 0.95314455],
       [0.13630316, 0.9096918 , 0.8781228 , 0.9197688 ],
       [0.08814317, 0.9508076 , 0.9621501 , 0.98716307],
       [0.12131184, 0.99118996, 0.97426176, 0.96854925],
       [0.04920864, 1. , 1.0000005 , 1. ],
       [0.36578724, 0.94052887, 0.8925061 , 0.7881899 ],
       [0.26557574, 0.7951536 , 0.77214193, 0.72464705],
       [0.27553454, 0.7239361 , 0.68584394, 0.64120626],
       [0.21557629, 0.6857567 , 0.76532984, 0.7349167 ],
       [0.8283578 , 0.69309807, 0.64572287, 0.41335058],
       [0.87239456, 0.26064587, 0.4511733 , 0.31129646],
       [0.5908555 , 0.5293684 , 0.62604094, 0.49871635],
       [0.31782064, 0.52202654, 0.49280882, 0.4878044 ],
       [0.30984417, 0.5682821 , 0.6389098 , 0.6161742 ],
       [0.22166616, 0.7254038 , 0.66918993, 0.6546855 ],
       [0.32545218, 0.6093979 , 0.57229376, 0.5442877 ],
       [0.33031747, 0.5234947 , 0.59727526, 0.5686779 ],
       [0.22162089, 0.5521288 , 0.573051 , 0.5706034 ],
       [0.52167 , 0.43832588, 0.45420122, 0.36970472],
       [0.8722813 , 0.4853158 , 0.41786528, 0.14377403],
       [0.8330698 , 0.34875202, 0.47691107, 0.43645716],
       [0.7263974 , 0.28560972, 0.4375472 , 0.34723997],
       [0.7201072 , 0.46328926, 0.43376207, 0.14634132],
       [0.89759374, 0.13656378, 0.10673714, 0.04878044],

```

Figure 3.16 Picture of inputs to the model

Figure 3.16 showing a set of all different inputs like Volume, Opening price, Highest price and Lowest price taken from the dataset that are all normalised, homogeneous and are ready to be fed to our LSTM model.

```

print("shape of X_train:",X_train.shape)
print("shape of X_test:",X_test.shape)
print("shape of y_train:",y_train.shape)
print("shape of y_:",y_test.shape)

```

```

shape of X_train: (42, 4)
shape of X_test: (22, 4)
shape of y_train: (42,)
shape of y_: (22,)

```

Fig 3.17 Dimension of the data used in LSTM model

Following the splitting of data for training and testing, we print the shape of both X and Y train and test variable. This is particularly important because it helps us to decide the dimension of input data to the different layers of the LSTM. Following this, we define the model.compile function and specify the optimiser type, the loss function and the other metrics like accuracy. Then we fit the model using the model.fit function where specify the parameters of batch size, the x and y train variables, the number of epochs and the number of verbose before finally training the model.

Software/libraries utilised in the project:

Python 3.6 was utilised as the base language upon which this project was programmed. For the Interactive Development Environment (IDE), JetBrains Pycharm and Google Colaboratory were used interchangeably. A number of popular python libraries were utilised for building models and visualising results, foremost being Matplotlib for graphical purposes, Pandas for data manipulation and statsmodel, Keras for machine learning models. Twitter API was utilised for Sentiment Analysis, and the Yahoo Finance API was utilised for data extraction from the Yahoo Finance website.

CHAPTER 4

RESULT ANALYSIS

4.1 Introduction:

This section will deal with the results obtained from running the dataset through the models built in the program. The results are obtained in the form of graphs to help visually depict the performance of the models, along with numerical figures that could help establish a comparison between the two models utilised in this project.

4.2 Result Analysis:

ARIMA:

The ARIMA model requires p, d and q parameters, as explained in the previous sections. For this analysis, ARIMA of the order (1,0,1) was constructed, as it best fits the dataset in question. These parameters were obtained through trial and error, and by comparison of the results obtained from tuning the model with different values.

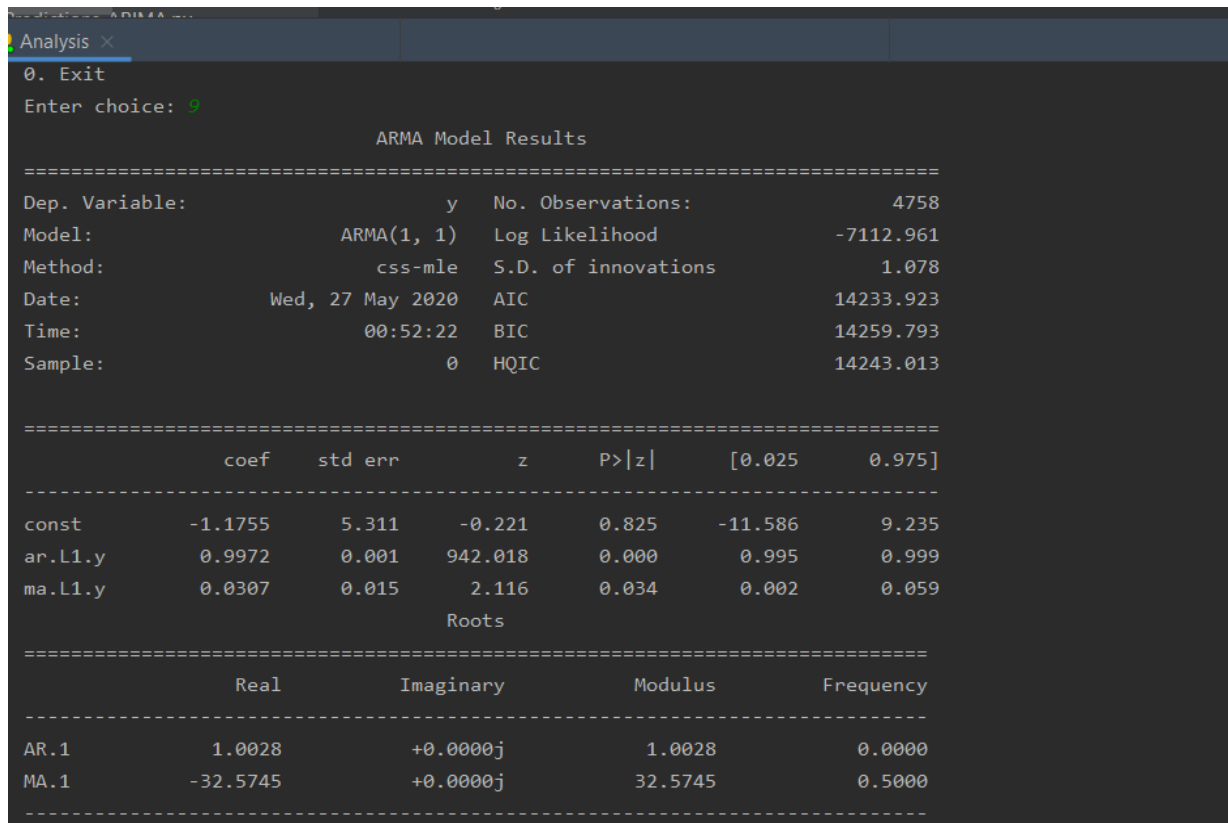


Figure 4.1 Results of the ARIMA model

Figure 4.1 depicts the results of running the model fitted to the dataset through a summary() function, which gives a brief overview of the process and provides values that can help finetune

the model for maximum accuracy. From Figure 21, 3 values are of utmost importance, namely the AIC value and the p values for $ar.L1.y$ and $ma.L1.y$. Before dwelling into these numbers, it should be noted that the value of d was taken as 0 since the dataset was already subject to the difference before running it through the model to make it stationary. Since the value of d was 0, the ARIMA function assumes no integration was required and hence returns the results under the title of the ARMA model, a model only suited to stationary time series. However, in reality, the order of this model is (1,1,1) since the data was already differenced before, and this model is, therefore, an ARIMA model. AIC stands for Akaike information criterion and is commonly used as a benchmark in statistics to compare various models against each other for the purpose of picking the best one. The AIC value of ARIMA(1,0,1) was the least of all the various other orders of the ARIMA function. A lower AIC value shows a better model, and therefore ARIMA(1,0,1) was chosen as the best fit for the dataset. The other variable that shows the performance of the model is the p-value. A p-value lesser than 0.05 shows that the series fed into this model is non-stationary, as discussed in the previous sections, and the lower the p-value of the models, the better the chances of the results being extremely accurate. Since both the autoregressive and moving average part of the model display a p-value lesser than 0.05, it can be assumed that the model fitted to the data will perform reasonably well.



Figure 4.2 Plot of Predicted vs True values for Microsoft

Figure 4.2 depicts the results obtained from applying the ARIMA(1,0,1) model built upon the training set to the testing set. It can be seen in the graph that the predicted values closely mirror that trend of the original values, with a small lag. This lag is introduced because of the differencing required to make the series stationary. However, overall the model serves as a good predictor of the future trends regarding the closing price of a company's stock.



Figure 4.3 10 day forecast of company stock prices

Figure 4.3 depicts the ten-day forecast of Microsoft's closing price based on the data fed into the model. The accuracy of the model is higher for a lesser period of days, but as the number increases the accuracy deteriorates. It should be noted that the forecast values are for the future; there is no test set to compare the accuracy of the results against. The part of the graph in red depicts the forecast, where the part in blue is the data it was trained upon, cropped to a certain point to display more precise graphs.

RMSE is the accuracy indicator that is used for comparing the results of both the models. The RMSE (Root Mean Square Error) value obtained is 2.357, which shows that it is a reasonably accurate model.

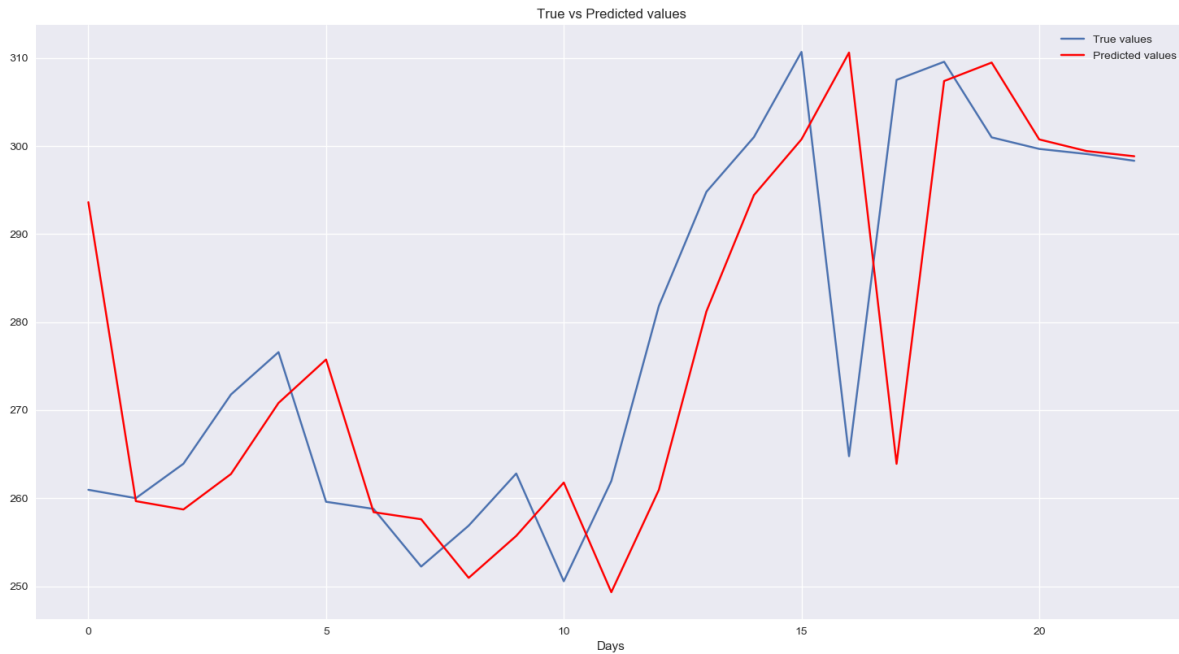


Figure 4.4 Plot of Predicted vs True values for Tesla

Figure 4.4 depicts the application of the ARIMA model upon Tesla's historical data. The test RMSE value, in this case, was found to be 17.06, a value much higher than the previous RMSE for Microsoft, which falls in the same category of time series. This value can be explained by the common fluctuations in the historical price of Tesla's stock, which makes the time series more non-stationary, more volatile and hence less suited to accurate predictions.

LSTM:

Before going into building the model, we need to prepare the data and most importantly create an Input data pipeline. This is done by merging the different input parameters into a single array list of float and then normalising them. The train test data was split in the ratio of 67-33% respectively.

Although many different types of LSTMs available can be used for model building, the Stacked LSTM has been used in this case because of two reasons: the type of use-case we are solving, and secondly, we require a two-dimensional result as our output. It is nothing but multiple hidden stacked LSTM layers one on top of another. An LSTM layer requires a three-dimensional input and LSTMs by default will produce a two-dimensional output as an interpretation from the end of the sequence.

We can address this by having the LSTM output a value for each time step in the input data by setting the return sequences variable as True argument on the layer. This allows us to have our desired output from hidden LSTM layer as input to the next.

The optimiser used is SGD or Stochastic Gradient Descent amongst various others like RMSprop, Adam, Adadelta, Adagrad, Adamax, Nadam and Ftrl.

This is because the other gradient descents perform redundant computations for bigger datasets, as it recomputes gradients for similar examples before each parameter update. SGD solves this redundancy by performing one update at a time. It is hence much faster and can also be used to train online.

The loss is computed through the Mean Squared error parameter because the MSE criterion gives us the perfect tradeoff between (squared) bias and variance. It also decreases as our algorithm gets more and more trained.

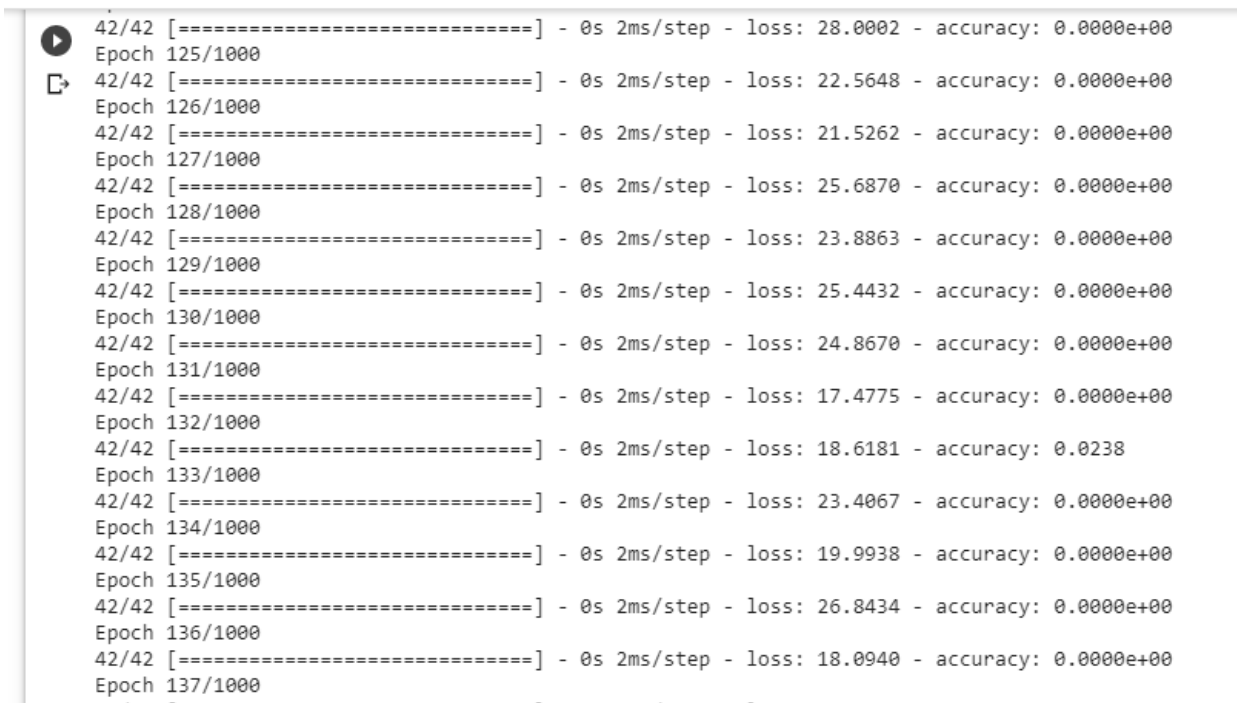


Fig 4.5 Information derived from LSTM model

Figure 4.5 showing a screenshot taken during the training of the model. The data values seen in the above pic suggests that on average, the loss decreases with the increase in an increasing number of epochs, as mentioned before.

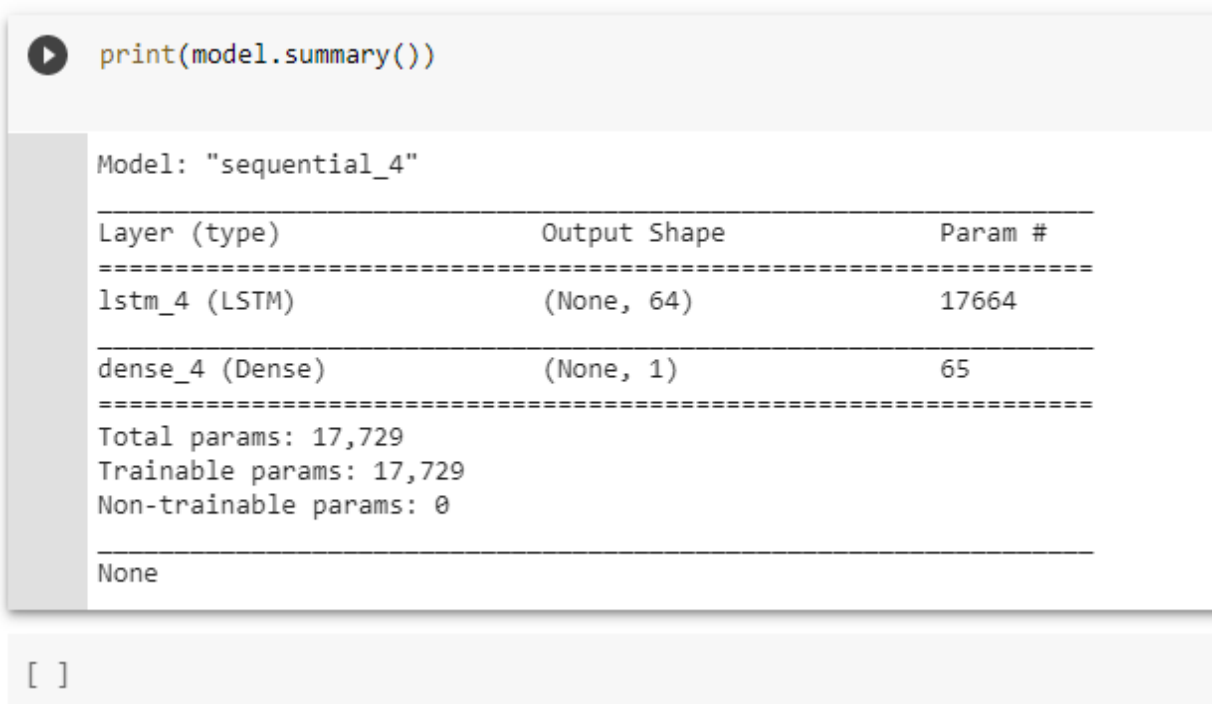


Fig 4.6 LSTM model summary

Figure 4.6 shows the summary of the LSTM model- it's hidden layers and the number of trainable and non-trainable params.

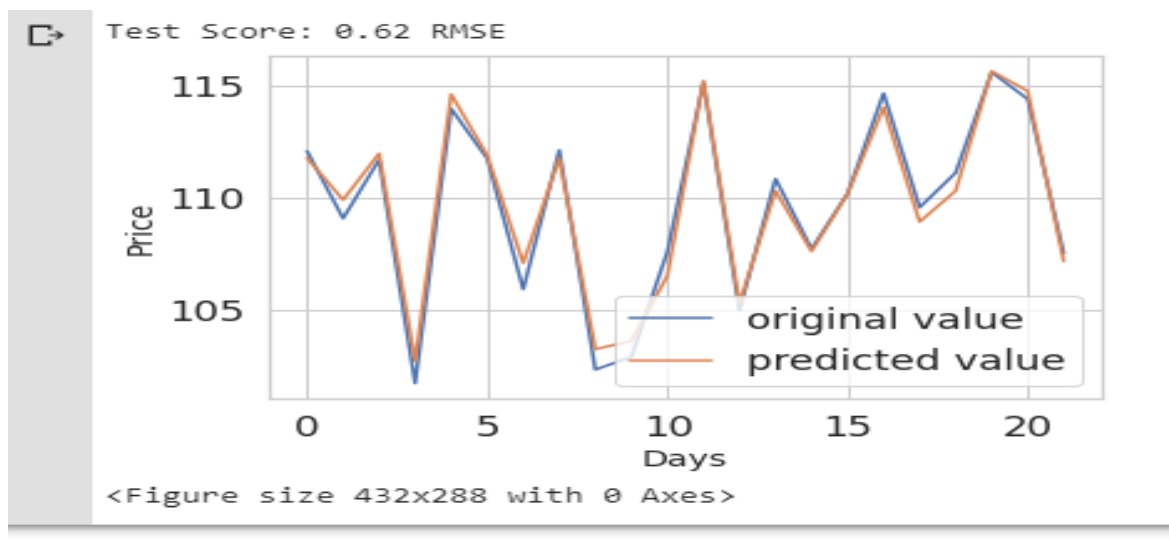


Fig 4.7 Plot of LSTM Predicted vs True values for Microsoft

Figure 4.7 shows the graph of comparison off the original values with the predicted values for Microsoft Inc. and also the RMSE value obtained. Applying this model to Tesla's dataset yields the following results.

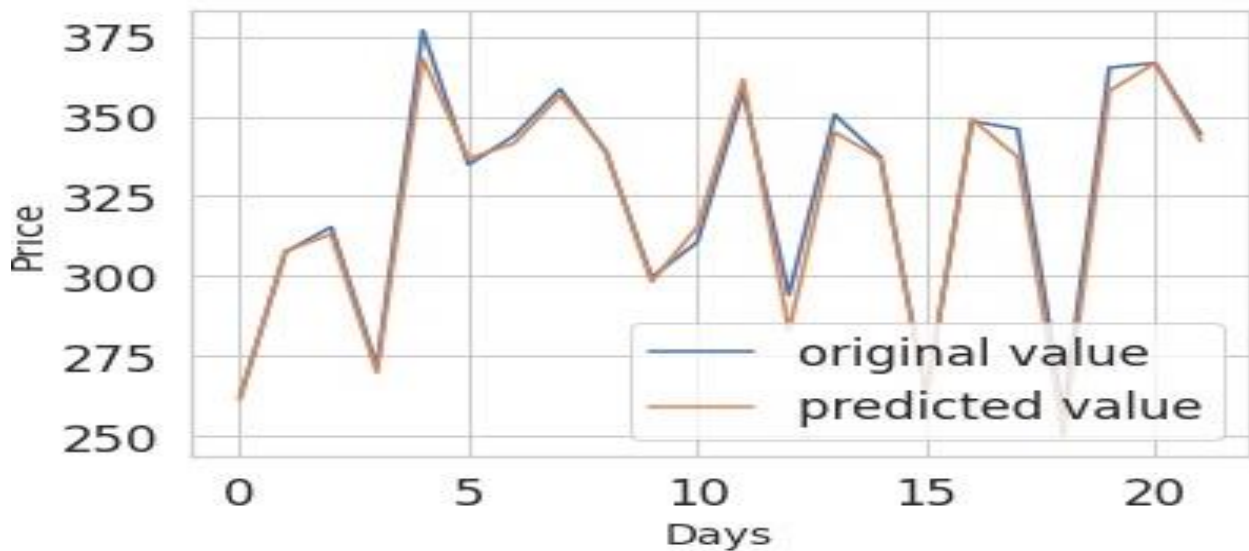


Figure 4.8 – LSTM plot of Predicted vs True values for Tesla

The RMSE in the above figure is 4.65, which is more than the RMSE value obtained for Microsoft, indicating predictions made for this dataset will be less accurate than the ones made for Microsoft.

Table 1 – Comparison of results

	Microsoft Dataset	Tesla Dataset
ARIMA	2.35	17.06
LSTM	0.62	4.65

The table above gives a summary of the RMSE values obtained after applying the models to their respective datasets. One observation that is apparent is that the models perform worse for the Tesla dataset as compared to the Microsoft dataset. This can be explained by the volatility in Tesla stock trend, which causes noise to be generated in the models built, causing a reduction in accuracy. Another observation is that the ARIMA model is far more adversely affected compared to the LSTM model. This is on account of the fact that LSTM is an implementation of Neural Networks and is hence more adaptable to the randomness in the dataset compared to ARIMA, which suffers accuracy wise greatly.

CHAPTER 5

CONCLUSION AND FUTURE SCOPE OF WORK

5.1 Summary:

The objective of this project was to analyse past stock information of a given company and build a prediction model that can effectively predict the closing price trends of the company's stock on a given day. Since it is not possible to achieve a 100% accuracy rate given the volatility of the stock market ecosystem the project sets out to achieve the maximum possible precision through tuning parameters and factoring in technical indicators that can be interpreted in various ways to help predict the future trends of the prices.

5.2 Conclusions:

Web scraping stock data from the internet has facilitated the creation of a dataset that will be the base upon which prediction models will be trained and tested. The web scraper ensures that the models have enough data to be as accurately trained as possible, which is paramount to achieving the maximum possible precision with regards to the closing price of a company's stock.

Technical analysis has been included as part of the project to serve as a confirmation on the trends forecasted by the machine learning models. The technical indicators display much-condensed information that if interpreted in the right manner, could complement the results displayed by the machine learning models. Since the machine learning models are entirely dependent on past data, they do not account for outside anomalies that may throw off any predictions made by them. This is where technical indicators are useful, as they can adapt to the rapidly changing ecosystem by offering up insights to the user who may then learn to predict the trends on their own by correctly interpreting the information provided by the various indicators.

As far as the accuracies of both the models are concerned, both are quite accurate because we are here predicting stock prices which itself is a highly fluctuating game and is impossible for anyone to predict. However, quantitatively, the LSTM (RMSE value 0.67) is better and more accurate than ARIMA (RMSE value 2.3570) in this case.

Although there is no doubt that both LSTM and ARIMA are one of the most accurate machine learning models that exist; however it cannot always be said with certainty that LSTM will be more accurate than ARIMA or vice versa for stock price prediction. That is more or less dependent on the nature of the use-case that we are solving and the nature and size of the data that we have with us. There is also ambiguity on the correlation between the number of epochs and the accuracy of the model. Also, the cost of computing has to be considered. It has to be

noted that ARIMA is a simpler model to implement, and LSTM is more complex (computationally). It, therefore, requires some amount of experience, experimenting with the two models to always get the required answer.

However we can safely conclude that an LSTM model will be wiser to use in similar circumstances like ours (like for the similar type of companies) as they will undoubtedly be more reliable and accurate, but this comes at the cost of more resources, due to more complexity.

5.3 Future scope of work:

The future scope of work will involve implementing more and more dependable machine learning models with even better accuracies which will be as much reliable irrespective of the underlying conditions including the type of companies for which the prediction is made for. After all there are millions of people who are directly and indirectly associated with the stock market, and a useful product like a stock predictor would benefit the entire ecosystem in a multitude of ways.

REFERENCES

- [1] Yuzheng Zhai, Arthur Hsu, Saman K Halgamuge, 'Combining News and Technical Indicators in Daily Stock Price Prediction' International Symposium on Neural Networks: Advances in Neural Networks-ISNN 2007
- [2] Youxun Lei, Kaiyue Zhou, Yuchen Liu, 'Multi Category Events Driven Stock Price Trends Prediction' 5th IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS), 2018
- [3] Sia Siami Namini, Neda Tavakoli, Akbar Siami Namin, 'A Comparison of ARIMA and LSTM in Forecasting Time Series' 17th IEEE International Conference on Machine Learning and Applications, 2017
- [4] Adebiyi A. Ariyo, Adewumi O. Adewumi, Charles K. Ayo, 'Stock Price Prediction Using the ARIMA Model' 16th International Conference on Computer Modelling and Simulation, UKSim-AMSS 2014
- [5] Dou Wei, 'Prediction of Stock Prices Based on LSTM Neural Network', International Conference on Artificial Intelligence and Advanced Manufacturing (AIAM), 2019
- [6] Edwin.M.Torralba, 'Development of a Deep Learning-LSTM Trend Prediction Model of Stock Prices' International Conference on Management Science and Industrial Engineering, May 2019
- [7] Technical indicators, investopedia.com
- [8] Dr S Lovelyn Rose, Dr L Ashok Kumar, "Deep Learning Using Python", WILEY, 2019, 978-81-265-7991-4

PROJECT DETAILS

<i>Student Details</i>			
Student Name	Uttaran Tribedi		
Register Number	160921118	Section / Roll No	B-23
Email Address	utribedi18@gmail.com	Phone No (M)	8073718219
Student Name	Abhishek Shukla		
Register Number	160921066	Section / Roll No	B-10
Email Address	shukla.abhishek2402@gmail.com	Phone No (M)	9920980072
<i>Project Details</i>			
Project Title	STOCK MARKET PREDICTION ANALYSIS		
Project Duration	4 months	Date of reporting	12-01-2020
Expected date of completion of project	29-04-2020		
<i>Organisation Details</i>			
Organisation Name	Manipal Institute of Technology		
Full postal address with pin code	Manipal Institute of Technology, Manipal – 576 104 (Karnataka State), INDIA		
Website address	www.manipal.edu		
<i>Internal Guide Details</i>			
Faculty Name	Dr Sandra D'Souza		
Full contact address with pin code	Dept. of Instrumentation & Control Engineering, Manipal Institute of Technology, Manipal – 576 104 (Karnataka State), INDIA		
Email address	sandra.dsouza@manipal.edu		
<i>Co- Guide Details</i>			
Faculty Name	Dr Renuka A		
Full contact address with pin code	Dept. of Computer Science & Engineering, Manipal Institute of Technology, Manipal – 576 104 (Karnataka State), INDIA		
Email address	renuka.prabhu@manipal.edu		