**Stock Market Prediction using Artificial Neural Network**

Dissertation submitted in part fulfillment of the requirements

for the degree of

[MSc Data Analytics]

at Dublin Business School

[insert name]

[insert mat\_no]

# DECLARATION

I, **[insert name],** declare that this research is my original work and that it has never been presented to any institution or university for the award of Degree or Diploma. In addition, I have referenced correctly all literature and sources used in this work and this work is fully compliant with the Dublin Business School’s academic honesty policy.

Signed: [insert name]

Date: [insert date]

# ACKNOWLEDGEMENTS

I would like to express deepest gratitude to God [insert appreciation].

My deepest gratitude also goes to my supervisor [insert appreciation].

I would like to thank [insert name and appreciation].

I would also like to thank [insert postgraduate coordinator].

I would also like to thank [insert friends].

Finally, I dedicate my work to [].

# ABSTRACT

In today’s world, many people face the great challenge to predict the prices of stock in the stock market. This study investigates the use of LSTM (Long Short-Term Memory networks) in predicting the prices of stocks. Stock market is a market that is open to the public where individuals trade the fraction ownership of a company. Despite the facts that it is difficult to predict stock prices because market prices are influence by many data, this research define a model to do so. With the large excessive number of models that are available, getting the right one that suit this topic is a bit difficult, especially when considering the continuous flow of new models and learning skills. LSTM are unique kind of RNN(Recurrent Neural Network) which has the Ability to study long-term dependencies. LSTM are uniquely designed for removing the long-term dependency problem and difficulties that might occur. A supervised learning is used in training the RNN with a multilayer feedforward network. Mean Square Error and statistical analysis is used in having an extensive testing of the of the model just to identify the configurations. Regression analysis is used to find mathematical relationship between the data gotten so as to predict stocks going up or down. The result gotten shows that short-term perspective could be predicted perfectly for significant numbers of time relays and it does not change when the quantities of data was altered. While for the long-time perspective, no significant result could be gotten as the data patterns keeps changing.

Keywords: Recurrent Neural Network, Long Short-term Memory, stock Market, Moving Average, Facebook Prophet.

Table of Contents

DECLARATION………………………………………………………………………………………………………………………………………………………………….II

[ACKNOWLEDGEMENTS…………………………………………………………………………………………………………………………………………………..III](#_Toc47556958)

[ABSTRACT IV](#_Toc47556959)

TABLE OF CONTENT…………………………………………………………………………………………………………………………………………………………V

LIST OF FIGURES…………………………………………………………………………………………………………………………………………………………….VII

[Chapter One: Introduction 1](#_Toc47556960)

[1.1 Background of study 1](#_Toc47556961)

[1.1.1 What is Stock Market 1](#_Toc47556962)

[1.1.2 Stock Market 2](#_Toc47556963)

[1.1.2 History of stock market 2](#_Toc47556964)

[1.1.3 Types of Stocks 3](#_Toc47556965)

[1.1.4 Stock Trading 7](#_Toc47556966)

[1.1.5 Artificial Neural Network 7](#_Toc47556967)

[1.2 Problem Definition 14](#_Toc47556968)

[1.3 Research Purpose 14](#_Toc47556969)

[1.4 Research Question 15](#_Toc47556970)

[CHAPTER TWO: LITERATURE REVIEW 16](#_Toc47556971)

[2.0 Literature Review - Introduction 16](#_Toc47556972)

[2.1 STOCK MARKET PREDICTION WITH DIFFERENT APPROACH 16](#_Toc47556973)

[CHAPTER THREE: Methodology 21](#_Toc47556974)

[3.0 INTRODUCTION 21](#_Toc47556975)

[3.1 Data Source 22](#_Toc47556976)

[3.2 Data Exploration 22](#_Toc47556977)

[3.3 Data visualization 23](#_Toc47556978)

[3.4 Data modelling 34](#_Toc47556979)

[3.5 Facebook Prophet 34](#_Toc47556980)

[3.6 RNN Modeling 35](#_Toc47556981)

[3.6.1 Preprocessing 38](#_Toc47556982)

[3.6.2 Data Mining and Machine Learning Tools: 40](#_Toc47556983)

[3.7 Python in Data Science 40](#_Toc47556984)

[CHAPTER FOUR: RESULTS AND EVALUATION 41](#_Toc47556985)

[4.0 INTRODUCTION 41](#_Toc47556986)

[4.1 Experimental Setup 41](#_Toc47556987)

[4.2 Facebook Prophet 42](#_Toc47556988)

[4.3 RECURRENT NEURAL NETWORK(LSTM) 45](#_Toc47556989)

[4.3.1 ACCURACE 48](#_Toc47556990)

[4.3.2 Prediction 49](#_Toc47556991)

[CHAPTER FIVE: CONCLUSION AND RECOMMENDATION 52](#_Toc47556992)

[5.1 CONCLUSION 52](#_Toc47556993)

[5.2 Recommendation and Future works 53](#_Toc47556994)

[5.3 Contribution to Knowledge 53](#_Toc47556995)

[References 54](#_Toc47556996)

[APPENDIX 57](#_Toc47556997)

[Analysis started](#_Toc47556998)

List of Figures

Fig 1.0 Artificial neutral network description 9

Fig 1.1 Four input ANN 10

Fig1.2 Difference between FNN and RNN 12

Fig 3.0 Stock prices description 23

Fig 3.1 Closing price stock of each companies studied 24

Fig 3.2 Volume of stocks against time 25

Fig 3.3 Apple’s moving average 26

Fig 3.4 Amazon’s moving average 27

Fig 3.5 Facebook’s moving average 28

Fig 3.6 Google’s moving average 29

Fig 3.7 overall plot of moving average 30

Fig 3.8 Distribution of daily return 32

Fig 3.9 Reshaped moving average 33

Fig 3.10 Description of the LSTM model process 35

Fig 4.1 Table showing the statistics of the dataset of the 41

S&P 500 of the four companies

Fig 4.2 Flow chart 42

Fig 4.3 Facebook prophet on Google’s data 43

Fig 4.4 Facebook prophet on Apple’s data 43

Fig 4.5 Facebook prophet on Facebook data 44

Fig 4.6 Facebook prophet on Amazon data 44

Fig 4.7 Flow chart of Facebook prophet 45

Fig 4.8 LSTM data gotten using 60 data 48

Fig 4.9 The figures showing the resulting values, time used in 47

carrying out tasks and the losses as training vectors are used

Fig 4.10 Flow chart of LSTM 48

Fig 4.11 Moving average of the network for 10,20 and 50 days 49

Fig 4.12 Close price prediction graph of Apple stocks 50

Fig 4.13 Close price prediction graph of Google stocks 50

Fig 4.14 Close price prediction graph of Facebook stock 51

Fig 4.15 Close price prediction of Amazon stock 51

# 

# Chapter One: Introduction

### 1.1 Background of study

As many people who wants to invest their money for future use increase, there is need for humans to be able to predict what will happen to their money in the future. A lot of people wish to know which business deal will be profitable to their business, especially the stock market which is a volatile market where the stocks of a company can begin to go up and you might be gaining as your stock is appreciating.

### 1.1.1 What is Stock Market

Stock market is one of the lucrative and legitimate business a lot of people invest their money in, just to make profit when a particular company is making its profit. To make profit in stocks, an investor must be familiar with the stock, he/she is buying and the amount the stocks is offered. There are many types of stocks and what categories them from each other is quite different to different type of people.

In other to make accurate or close to accurate prediction on the stock market, there must be accurate study of previous data’s and see the similarities or discrepancy that can help to make a better decision in the near future. Artificial Neural Network is one of the major tools that most people who do machine learning, stock prices predictors use in their prediction of the future event or values of stocks.

For a protracted period of time we have a sequence prediction problem and they are though-about to be one of the toughest issues to unravel within the knowledge of science in Industries. These comprises of varying problems ranging from using market patterns to predict sales to understanding plots to patterns to prediction of keywords on phone when typing on keyboard (SRIVASTAVA, 2017). With recent breakthrough in data science prediction, research has shown that LSTM(Long Short Term Memory Networks) is one of the best tool to use to predict varying data effectively (Brownlee, 2017). LSTM had been discovered to be effective more than the widely used feed forward Neural Networks and RNN in so many ways. This is due to the facts that LSTM has a property that it uses for selectively recollecting patterns which is use for long duration of time.

This project makes use of LSTM to predict the prices of stocks in stock market over a Short and Long period of time and it is implemented by using python programming language with python GUI.

Science is the improvement of other scientific discoveries, so this section of this thesis is to review other people’s journals, article, books and so on as it relates with the topic Stock market prediction using Artificial neural network model. This section is written to help gain knowledge on how the stock market works and some of the factors that expert see, to make a prediction about the stock market.

This section also reviews some other scholar’s work on how they have developed an Artificial Neural Network Model to make predictions and some of their conclusions too is included in this thesis.

### 1.1.2 Stock Market

Stock market is a market that is open to the public where the buying, selling and trading of company’s stocks which trade on a stock exchange or over the counter. Stocks can also be referred to as equities which is owning of a fraction in a company and stock market is a geographical location where investors can either sell ownership of their investable assets. The stock market is so unique and important to the economics of a company and also to the nation At large. The Stock market gives individual or companies the legal backing to generate funds or accept capital from the general public.

### 1.1.2 History of stock market

Modern trading can be trace to the first trading of shares in the east india company in London while the history of trading stock can be traced way back to the mid-1500s in Antwerp (Stock Market , 2020). The sudden increase in the prices of stock in the 1990s and the crash that follows, beginning in the year 2000 are proof of the strong connections between stock markets across different countries. Recent literature documents the link between stock markets in the USA and the rest of the world (Eun, 1989) and (Susmel, 1994). The combination of European financial market could also be the reason we have strong correlation between prices of equity in individual European countries.

Charters was made available to a number of companies which east India company was part of them in the 1600 by three government which are the British, French and Dutch. All the good that are brought from the east are all transported through the sea, which is risky as pirates’ attacks is on the high side or ship reck could also. In other to lessen this risk, ship owners now look for investors to provide financing collateral for a voyage. While investors will now get just a portion of the monetary returns gotten only if the ship returned back successfully, with goods onboard that are for sale.

During early times, companies’ shares were issued to investors on paper which will help the investors to trade their equity with other investors, but regulated exchanges was not in existence then till the formation of the London Stock Exchange in the year 1773(Stock Exchange, 2020).

### 1.1.3 Types of Stocks

There different types of stocks according to their categories and we have common stock and preferred stocks.

#### 1.1.3.1 Common stock

Common stocks are types of stock which is the most popular types of stocks that many people trade. Common stock can be defined as the partial ownership in a company and if the company gets dissolved in the future, the shareholders will get a proportional share of the value of any remaining stock of the company. When it is time to elect board members which are individual who oversee the major decision made in the company, Common shareholders will only have one vote per share. Common shareholders get more returns than other investors over a long period of time through capital growth. Common stock provides his shareholders unlimited upside potential in written form, but if the company fail, they lose all, when the company don’t have any assets left. The creditors, bondholders and preferred stock investors will be paid first before crediting the common shareholders with what is left and usually, there is little remaining for them.

#### 1.1.3.2 Preferred stock

Preferred stock is quite different from that of common stock, because it gives its shareholders preference, as it gives back some amount of money to his shareholders, if the company fails or be dissolved and owners of this stocks don’t usually have same voting right when electing board members. The net result is that preferred stock when viewed as an investment can be linked to income bond investments rather than the regular common stock. Often an organization can supply solely Common stock. This is reasonable, because that is what shareholders wants to buy always. Preferred stocks are callable i.e. the company can buy back these shares from investor for any reason and at any time.

Other types of stocks according to their categories are as follows:

1. Stocks which are categorized according to their total worth of all their shares and this is called Market capitalization. Companies which has the biggest market capitalization are called **Large-cap** stock while the one smaller to that is called **Mid-Cap** stock and the smallest market capitalization is called **Small-cap** stock. Even though there are no precise thing to use to majorly say the difference or use to categorize it. Large-Cap stocks are safer generally and are more conservative as investment even though mid-caps and small-caps more capacity to grow in the future but it is very risky compared to large cap.
2. **Domestic stocks and International stock**: this type of stocks is categorized according to their geographical locations and most people use company’s official headquarters. The place where these companies headquarter is might be different to where the company sells its product or services to.
3. **Growth and Value Stocks**: Another method which is used to differentiate between two popular investment strategies is used to get the growth and Value stock. Growth investor are always interested in investing their resources in stocks of company that has a quick increase in sales and profit. While that of value investors are interested in shares of companies that is not expensive when compared to their past stock price or when compared with their peers. Growth stock is very risky but the returns that could likely be gotten from this stock is what make it very attractive. Successful growth stocks have businesses that are strong and increasing in demand among customers especially when I come to long term trends throughout the community of consumers that makes use of their product and services. There are many competitors, which are competing and any disruption cause by rivals could be devasting to the extent that it could cause a low sharp low price.
4. **IPO stocks:** IPO stocks (initial public Offering) as the name indicate, IPO means initial stocks of a company as at the time of the company going public. In other words, Initial public offering means the very first stock a company is offering to the public, the very first time it is going public. Disagreement could ensure among investors about what the company could do to make profit or grow its income with time. Many investors are always excited to invest in this company as the company had shown applaudable success and wishes to increase its companies’ capacity and income by going public. It is an IPO stock for at-least a year and it can go up to two or four years after it becomes public. Before investing in an IPO stock, the investors must go through this three steps (Bowman, 2020) which are: S-1 document of the company should be read and understood by the investor, look at the market the company is going into as it is outlined in the S-1 document and also look carefully at the growth the company had made, the motivations of those leading the company, tendence to make gain over time and advantages that could be gained if invested in.
5. **Dividend stocks and non-dividend stocks:** Many investors get back some amount of money from its stocks recurring at an interval of time which is known as dividend. Dividend stocks, just like the common stock is what many investors goes for as they get income at a regular interval. While non-dividend paying stocks do not pay income to its shareholders but rather re-invest the profit gotten thereby adding more to the holdings of its investor or shareholders.
6. **Income Stocks:** Dividend stocks is also known as income stocks as it provides regular income to its shareholders at a regular basis. Any investor that wishes to be to be getting income regularly goes for this kind of stock and companies that already have large customer base and that is well established offers this kind of stocks to investors. Reason being that their business could not easily goes out because there are buyers to buy their goods when they produce. Many investors that are close to their retirement age or even those in retirement goes for this type of stocks, so that they can have what to fall back to when the salary income is no more available.
7. **Safe Stocks:** Despite the fact that safe stocks have low risk, there is not nothing like 100 percent safe. Safe stocks have low rise and fall in its curve on the map of stock against the market price of stock in stock market. In Safe stock, even though investors do not make huge profit when compared to other stocks, they also do not make huge losses too. Investors who could not risk all of its investment always go for the safe stocks to be on a safer side.
8. **Cyclical Stocks and non-cyclical stocks:**  this type of stock is categories due to the effect of favorable and unfavorable conditions on the business of the company’s stocks. For examples, this period of corona virus pandemic, some businesses like Aviation Industries, manufacturing industries are greatly affected as people are in Lockdown period. People could not patronize these businesses and as a result the profit of this companies are affected. These business stock which are offered by this company is called Cyclical stocks. While the non-cyclical Stocks are stocks which do not easily rise or fall when there is a change in event, for example the food and Agricultural industries still make their money because despite the pandemic people still eat food. So, when there is market downturn, non-Cyclical stocks perform excellently well while the Cyclical stock do not perform well as a result of the sharp decrease in demand.

### 1.1.4 Stock Trading

Anyone who uses the daily fluctuation of stock price to always buy and sell stock is called a stock trader. This traders aim is to make money in few minutes, hours, days or month instead of buying the stock of a company and wait till the stocks grow over time that they will pass from one generation to another (Yochim, 2020). Stock Trading can be categorized based on some certain Characteristics which are stated below:

#### 1.1.4.1 Active Trading

When a trader places an average of 10 or more trading per month is called active trading. This kind of trader uses different strategies and different timing to make decision about buying and selling of stocks. They study the market over time and get use to the different market trends and could guess what event could shake the market that will either turned to profit or loss for them.

#### 1.1.4.2 Day Trading

As the name implies, this set of traders makes daily trading i.e. they buy and sell the same stock in just a day of trading, and they are not consigned about whether the stock could grow over time. All they want is to make gain immediately, and this make them to study the market very carefully as to know when to buy and what time in the day to sell the same stock. They capitalize so much on price fluctuation that occur in that very day (James Royal, 2020)

### 1.1.5 Artificial Neural Network

Artificial Neural Network (ANN) is the use of computer system to imitate the way the human brain process, analyze and produce data. It is the muse of artificial intelligence which the computer learns and makes decision based on what it had learnt over time. ANN helps to solve many Statistical problems using the way humans solve daily problems. For example before humans could predict that rain will fall the next day, they will have to consider many different factors like the wind, clouds, seasons and so on, so also the computer is supplied with this data and it uses ANN to predict rainfall for the next day, just as the human brain will do it (Graves & Schmidhuber, 2005). ANN has a feature of learning by itself which makes it effective to more data is supplied to the network.

Artificial Neural Network is the copy of the Human brain, just like the neutron in the human brain has different interconnected nodes, the ANN too has its own too. Each human cell that are called neurons are the cell body which sole work is to process data inwards which is known as input and away front the human brain as output. The ANN likewise also has hundreds or thousands of artificial neurons which are known as processing unit and the they are connected by the nodes. These Processing Unit consist of two units which are the input and output units. As the word input means the Input units of the processing unit takes in data in different forms and structures, and this is due to the internal weighting system and the way the networks make use to learn about the data to have an output result. Humans needs instructions and guidelines before they could come out with an output, the ANN also use a particular learning procedure called Backward Propagation Error which is used to get perfect or nice output.

At first the ANN has to goes through a coaching process of acknowledging the different patterns in data and these patterns could be visually, textually or aurally. During this learning process the ANN will compares the output expected to produce with the real output that the ANN produces. The backpropagation is now used to adjusted the difference in the two output. This suggest that the output unit is now used to get the input unit which means the ANN works in backward direction and during this process the ANN to regulate the burden of its connections around the two units till the distinction between the particular and wanted outcome produces the reduced attainable error. It is also observed that during supervisory and training stage, the ANN learnt what the output will be, the characteristics to look for using a yes or No question with binary numbers.

As shown in the figure below the Input layer consist of the neurons of the Input unit that transmit data to the hidden layer. While the hidden layer now sends this information to the output layer. Each of the neurons has its own weighted inputs synapses, one output and an activation function (which state the output when given input), and one output. Synapses are parameters which can be adjusted that changes a neural network into a parameterized system.



Figure 1.0 Artificial Neural network description Source: (Davydova, 2017)

For example: An Artificial neuron which consist of four inputs as shown in the figure below:

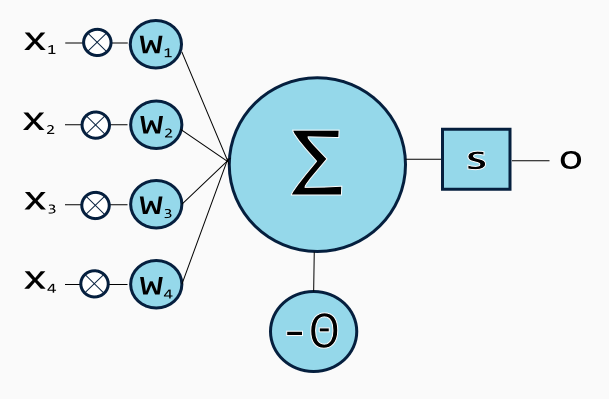


Figure 1.1 Four input ANN Source: (Davydova, 2017)

Activation signal is produced by the weighted sum of all the inputs which is then passed on to the activation function just to get the output from the neuron. Activation function that is popularly used is are the Linear function, step function, Rectified Linear unit function, sigmoid function, and tanh function which equation are stated below.

**Linear function**

**Step function**

**Logistic (Sigmoid) Function**

**Rectified linear Unit (ReLu) function**

**Tanh Function**

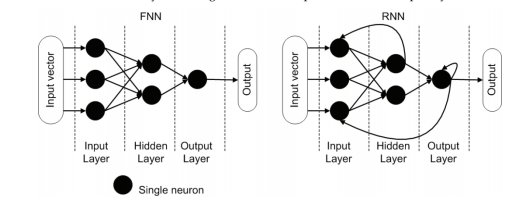
Training process is process of weights optimization whereby error of predictions is reduced and the network is made to get to a specific height of accuracy. Backpropagation which is define earlier in this literature review is used to calculate the loss function’s gradient and this method is the most commonly used method to get the error that each neuron contributes in the network. To make the network more flexible and powerful, an additional hidden layer is used. We have various types of ANN which are specific to the need of the user.

There are other different types of ANN that people use today to solve complex statistical problems which are:

1. Multilayer Perceptron (MLP)
2. Convolutional Neural Network (CNN)
3. Recursive Neural Network (RNN)
4. Recurrent Neural Network (RNN)
5. Long Short-Term Memory (LSTM)
6. Sequence-to-Sequence Models
7. Shallow Neural Networks
8. Feed-Forward artificial neural network

Feed-Forward artificial neural network (FNN) is a type of artificial neural network with feed-forward topology which means it has a condition which is, data move from input to output and cannot go the other direction i.e. no back-loops. In FNN the numbers of layer are unlimited, no limit also to the type of transfer function that is used in the individual artificial neuron and there is no limit also to the number of connections that can be in between artificial neurons (R., 1996). The easiest and simple feed-forward artificial neural network is that which consist of a single perceptron and also has the capability of learning Separable linear problems.

The figure below shows the FNN topology of the ANN and also the RNN which is the Recurrent Neural Network.

 Figure 1.2 Differences between FNN and RNN Source: (Andrej Krenker, 2009)

The Recurrent Artificial Neural Network is the artificial neural network with a recurrent topology. Just like the Feed-forward neural network (FNN) but does not have limitations as regards the back-loops. So therefore, data can be transmitted either forward or backwards. Internally the state of the network is allowed to behave temporarily dynamic. It can also use internal memory in processing any particular order of inputs.

Examples of Practical Applications of ANN

An Economic test hypothesis of the efficiency of the market was conducted by an economics name Halbert White (1988) and he was using daily stock returns from IBM. He developed a model which is a linear auto-regressive model and an auto regressive model by using a back-propagation neural network. A test on about 1000 daily returns were carried out on the daily with the linear auto-regressive model, the R-squared which was use for the equation was 0.008. at 10% level this approximate calculation is not remarkable at the 10% level and this make us to doubt whether the model truly capture the true structure in the price series. The Back-Propagation model on its own has it own R-squared of 0.175 which when considering the obvious it is splendid. On the other hand, as at the time the Back-Propagation model was tested out-of sample (both before and after the estimation sample), it was noticed that the correlation of the prediction.

However, when the back-propagation model was tested out-of-sample (both prior to and after the estimation sample), the correlation of the predictions with the actual returns was very small and insignificant. In fact, in one case, the predictions and actual returns were negatively correlated. In this case, the back-propagation network failed to generalize out-of sample. As White notes, the network may have discovered fleeting structures. These would have been actual features of the underlying process during the sample time period, but they were not part of the underlying process over the test sample time periods. More likely, the back-propagation network simply over-fitted the training observations which degraded its out-of sample performance. On the other hand, the prices may truly obey the efficient markets hypothesis and are thus unpredictable from observations on past behavior. One would expect this to be a particularly noisy problem domain and White did not attempt to use any techniques to improve the generalization ability of back propagation.

McMahon (1990) has compared the performance of a back-propagation network to an expert system in selecting maneuvers during simulated aircraft combat. The neural network was derived from, and compared against, an existing air combat expert system - the Air Combat Expert Simulation (ACES). ACES consist of thirty-eight production rules which produce twelve offensive and five defensive maneuvers. The system is supplied with twelve inputs representing the current battle situation (distance to enemy plane, relative position, etc.) on which to base its decision. McMahon trained a back-propagation network using the thirty-eight production rules as prototypes to supply network inputs and target. Each of the production rules selected a specific maneuver if a sub-set of the twelve inputs matched particular ranges in the rule. Where an input was not used by a production rule, McMahon set the input to a random value when training the network. For validation purposes, forty combat scenarios were presented to a group of expert fighter pilots; and, the maneuvers chosen by the pilots were used to assess the performance of the two systems. The trained neural network, Tactical Air Combat Intelligent Trainer (TACIT), outperformed the ACES system despite having only the ACES production rules as training input. ACES agreed with the pilots' maneuver selection on 25% of the scenarios (ten of forty); TACIT agreed with the pilots on 67% of the scenarios (27 of 40). Because both systems were based on the same production rules, the network's superior performance is based on a better resolution of mutually consistent production rules. In some cases, more than one production rule is valid for a given set of inputs. The resolution strategy in ACES to select among these competing rules proved inferior to resolutions made by the neural network.

## 1.2 Problem Definition

The stock market is a crucial part of the business world, it could be an opportunity to make a lot of money if traded at the right time, but there is a lot of dynamics in the stock market which makes it risky. Stock prices are influenced by a lot of factors which makes predictability almost impossible and investors are frantically searching for better ways to forecast stock prices. If investors can successfully find a way to make accurate stock prediction, it will improve their chances of making large gains in the stock market because it will give them an idea of which stocks can be traded at a particular time.

However, by studying the numerous factors influencing prices in the stock market, it is possible to predict prices in the market, but this is almost impossible to be done manually by human. So, it is necessary to harness the processing power of the computer to attempt to solve problem.

Neural networks have been proven to be very effective in making sense of large data such as the stock market data. The neural network (NN) learns from huge data by finding patterns in them, the patterned found forms the knowledge base which is used later on a test data to make predictions.

## 1.3 Research Purpose

This purpose of this research work is to experiment the effectiveness of Neural Network (NN) in making stock market prediction. The type of neural network to be used in this research is the Recursive Neural Network (RNN), the result of this will be compared with the result of the famous Facebook model (prophet)

## 1.4 Research Question

The purpose of this research is to find answers to the following questions:

1. How effective is artificial neural network in stock market prediction

2. Reveal the most important varibles that affects the price of stocks in the stock marget

3. What is the performance of the recursive neural network (RNN) to Facebook’s Prophet

# CHAPTER TWO: LITERATURE REVIEW

## 2.0 Literature Review - Introduction

Our research will not be complete without looking at past works in the area of stock market predictions. There have been numerous approaches to making stock market prediction, most of which are based on machine learning (Nadh & Prasad, 2018).

## 2.1 STOCK MARKET PREDICTION WITH DIFFERENT APPROACH

(Stock market prediction, 2020) highlighted three (3) categories of stock market prediction which are; fundamental analysis, technical analysis and machine learning. The fundamental analysis is focused on the value of the company while the technical analysis attempt to make prediction based on past trends in the data such as moving average methods. Machine learning on the other hand uses techniques such as Artificial neural network and Genetic algorithm.

The autoregressive integrated moving average (ARIMA) model is one of the popular and early methods of making stock predictions. (Adebiyi, Adewumi, & Ayo, 2014) implemented the ARIMA in their research work, they used the New York Stock Exchange (NYSE) and Nigeria Stock Exchange (NSE) data. They considered a lot of factors such as Bayesian or Schwarz Information Criterion (BIC), Square error of regression (SE) etc. Their research showed that the ARIMA can make good prediction on stock market data. However, the major shortfall of their model is that it can only do well on short-term.

Authors have adopted Fuzzy Logic to forecast the foreign ER and foreign data streams (Valavanis.K.P, 2008) (Mittal, 2009). Gradojevic [ (Gradojevic.N et al, 2015)] discussed about neuro-fuzzy (NF) decision-making in trading the foreign exchange and other applications. One of the findings made by this individual is that non-linear ANN ER microstructure mode when integrated with FL controller, develops a particular set of procedure of trading on earnings that have higher return rate, on average return rate when compared with the procedure of buy and hold. Other findings that was that after encompassing the costs of transactions, the gains gotten from Technology neuro-fuzzy did not reduce but increases or even maximize on some periods. It was also discovered that applying the model of neuro-fuzzy to the issue f tracking the market of foreign exchange as seen by the trading signals of chartists during strong depreciation periods. Kablan have expanded ANFI (Adaptive Neuro-Fuzzy Inference) systems for developing an expert system which is potential of adopting fuzzy reasoning integrated with the recognition of pattern capability of NN to be adopted in financial trading and forecasting (Kablan, 2009). David explained about topology of FL (Fuzzy Logic) expert advisor for market of foreign exchange. Expert advisor would act a robot for trading the foreign exchange using FL. It was indicated that the FL based expert advisor robot has the ability of making up to eighty per cent consistent profitable trades. It was demonstrated that emerging expert advisors that could reach a higher profitable trades percentage on the basis on this topology. Alizadeh et al (2009) presented an ANFI system for United States dollar or Japanese Yen ER forecasting. ANFI could be utilized to better solutions to clients rather than completely models of black-box like NN. The variables that was used for inputs layers are some of the fundamental and technical indexes as the proposed NF rules base system were in use. Fuzzy clustering of space output was adopted for generating Membership functions. 28 candidates’ variables were used as input for both currencies when testing the NF Model. For comparing SYM (Sugeno-Yasukawa Model), multiple regressions, feed forward multi-layer NN are benchmarked. Alizadeh et al concluded by saying that ANFI system shows its superiority in case of reducing the prediction error, robustness and flexibility. Liu (2008) combined rule-based FL with an ANFI system was used for building a model for forecasting the movement of foreign ER. Despite the facts that FL system models trained has a good strength in predicting, with ER dramatic their performance is not precise. It was suggested that the system requires background knowledge that would permit it to reinterpret and or integrate aspects in the data into new aspects that could lead to simpler or accurate patterns.

Li and Taiwo (2015) developed multi-objective GP based system for financial forecasting. It was constructed based on the prior tool for decision making in financial GP (Li, 1999). The Developed system improve financial GP in so many ways. The First and forecast, proposed system acquires data faster by the way of multiple conflicting objectives which is given to inherent property of multi-objective EA (evolutionary algorithms) which is that a set of Pareto front optimal solutions was gotten when algorithm is executed just one time. Second one, proposed system is simple and friendly from the view of users. The quantity of user- supplied variables which are firstly used by financial GP is removed and this is what makes it user. It becomes simple as for the customer it is not necessary to give a priori domain knowledge that is needed for proper usage of such parameters (Taiwo.S, 2015). Bylander and Schwaerzel (2015) have predicted financial time series (FTS) by GP with high-order statistics and trigonometric functions. When the performance analysis was considered for the extra FTS which is Financial Time series indicates that basic GP models plus the sets of valued-added function perform excellently okay than that of the Buy and Hold strategy, Akaike selected models of ARMA and GP model. It was recommended that deploying trigonometric functions and functions of high-order statistics would be valuable addon for other GP applications in same domain or other. It was also suggested that incorporating extra measures in statistics for example exponential moving average would further maximize the proposed approach. Diaz (2020) discussed about speculative strategies in foreign ER on the basis on predicting GP. From the conclusion of Diaz, it was observed that the predictive analysis gives proof against the unpredictability of the ER Evolution and also against the facts that ER makes use of the process of random walk. It was observed that GP performance was statistically excellent while forecasting than the model of random walk and more than Yen/Dollar forecasting one-period-ahead when both sign and point of prediction is considered. Thus, it can be understood that GP outperforms statistically better in terms of prediction when compared with the random walk model. Ravi et al (2012) have predicted foreign ER using methods of computational intelligence. During the prediction of foreign ER in paradigm of computational intelligence, six architectures of non-linear ensemble are proposed. BP neural network, Multivariate ARS (adaptive regression splines), Wavelet NN, SVR (support vector regression), genetic programming (GP), Group method (GM) of handling the data (HA), DENFIS (Dynamic evolving neuro-fuzzy inference system) are selected as the ensembles members. The ER data of US dollar for British pound, Japanese Yen and Deutsche mark is taken for checking the performance of the network and comparing the ensembles performances. GM of HA and GP outperformed well than other ensembles. It was also seen that assembling in paradigm of computational intelligence is a perfect substitute to the expanded techniques in forecasting foreign ER.

There are numerous researchers who employed ANNs to forecast the SM, market and financial indexes (Enke, 2005). Cao et al (2005) have compared the effectiveness of French and Fema model and ANN model for forecasting financial values in Chinese SM. It was observed that ANNs perform well in predictive power (Aamodt.R, 2010) than linear models. Pendharkar (2005) have adopted ANN applications for predicting the issue of bankruptcy. It was observed that ANNs perform well more than the SDA (statistical discriminant analysis) for hold-out samples and training. Sarker and Kamruzzaman (2003) compared ANN oriented models with ARIMA ((auto regressive integrated moving average) to predict the foreign ER. In this research, it was proposed and examined three ANN based models using SCG (Scaled conjugate gradient), SBP (Standard Back propagation) and BP with Baysian regularization for forecasting various currencies against dollar of Australia. It was observed that ARIMA model did not performed well than all ANN based models. It was also seen ANN based model could predict foreign ER almost precisely. Merh et al (2010) compared hybrid approaches of ARIMA and ANN for predicting future value of index and SM in India. It was shown in the journal that hybrid ANN\_ARIMA and BSE oil and gas ANN were able to control the input set of data and forecast future closing price while using the ARIMA\_ANN and ARIMA could not predict future values. Dannie and Rose (2006) developed a flexible model namely NCSTAR (Neuron Coefficient Smooth Transition Auto Regression), an ANN for predicting as well as modeling the non-linearities in monthly ER. Most of the studies have shown that ANN outperformed well than other models. On the other hand, Pacelli (2012) compared and analyzed potential of various mathematical models (MM) namely ANN, ARCH (Auto-regressive conditional heteroscedasticity) and GARCH models for forecasting the daily ER, the U.S/Euro dollar were the best variables used in determining the best performing models between these. When potential of different MM was compared such as ANN, GARCH and ARCH models, conventional indicators to evaluate the model’s ability indicated that GARCH and ARCH models in their static formulations, outperformed better than ANN. ARCH model demonstrated the best predictive potential with a static approach. At the same time, it was noticed that Goksu and Erdogan (2014) examined about forecasting Turkish Lira (TL) and Euro ER with ANN. From the analysis it was observed clearly that ANNs work well in predicting the future of Euro and TL ER. Bhatt and (2015) developed a model for predicting the foreign ER and compare develop efficiency of developed model with the prior techniques. Model was developed to forecast exchange from euro to dollar, rupee to dollar, pound to dollar exchange. Another Observation made was that ANN is faster than imagined in estimating the foreign ER. In addition to this, it was also noted that developed model provides better percentage in terms of accuracy. Further observation made was that a slight difference was seen between the predicted ER and real ER.

# CHAPTER THREE: Methodology

## 3.0 INTRODUCTION

There are several methods that other scholars used in predicting the stock prices of a particular company and in my investigation, which is included in my Literature review. It shows that most methods used behave very well i.e. they could predict values accurately when the data’s that is supplied as input are still few. While in a long run, this method fails woefully and could not predict data accurately when the input data is large.

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

For example, if someone wants to predict the next word in a sentence to autocomplete the sentence or to predict the next day stock price etc. by using ANN. In its simplest form the neural network will have an input layer which receives the input, a hidden layer where the activation is applied and an output layer where one finally receives the output. While for more complex forms, we will have multiple hidden layers which are; the input layer receives the input, the first hidden layer is activated, this activation is now sent to the next hidden layer, and each successive layer’s activations are sent through the layers to finally produce the output which is called the output layer. Weights and Bias of each hidden layers are unique to each hidden layer. For this, each layer behaves independently and, unless they have the same weights and bias, these hidden layers cannot be combined with one another.

The state of the previous input is stored in a recurrent neuron and this is joined or added to the input so as to have some relationship between previous input and current input

Long Short-Term Memory networks (LSTM) are a special kind of RNN which has the ability of learning long-term dependencies. LSTM can be describe as having a chain-like structure, but the repeating module has a slightly different structure. There are multiple layers which interact in a very special way. A common LSTM architecture is composed of a memory cell, an input gate, an output gate and a forget gate.

## 3.1 Data Source

The data which is used to train the network in this thesis are gotten from the yahoo finance on its website. The standard and poor’s 500 index can be defined as a market capitalization weighted index of largest US public traded companies which are 500 in numbers. But in this thesis only 4 of this companies’ data is used to train the network. To get the data of this companies such as the Open, High, Low, Volume and the Adjusted closing price, I used the API of yahoo finance and I stored each companies’ data on a csv file. The data features used are the stock prices of this companies from the year 2013 to 2018.

## 3.2 Data Exploration

As the data gotten were much as it spans from 2013 to 18 which is a 5 years period. We took a look at the data points in the dataset to see the trend in it.

The high and low can be regarded as the maximum and minimum prices of stocks in stock market at a given time period. The prices at which a stock began and ended trading in the same period are refer to as Open and close respectively i.e. closing stock and opening stock. Volume is the amount in total of all the trading activity. Adjusted closing price is the stock closing price after amendment to get the correct stock value after accounting for any corporate actions.

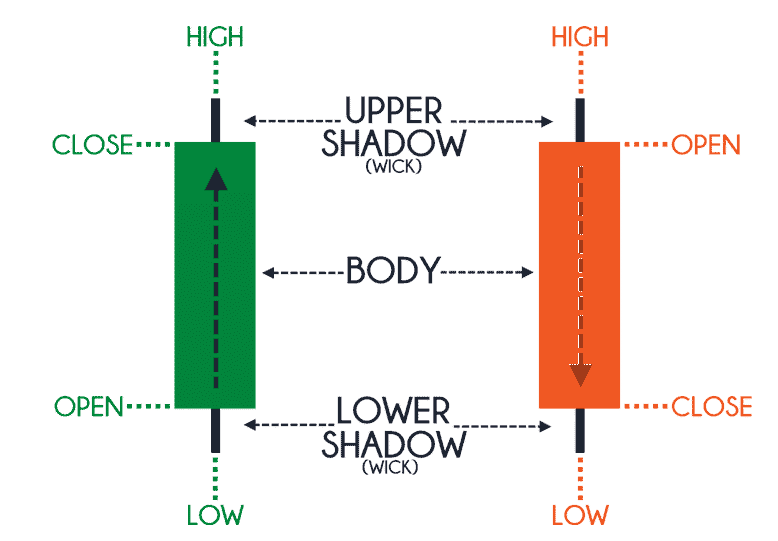


Figure 3.0 Stock prices Description (source: <https://analyzingalpha.com/open-high-low-close-stocks>)

## 3.3 Data visualization

To get a clearer view of the data, I had to plot the closing stock price against time, for the four companies as shown in figure 4.1. The closing stock price is seen to be rising for each of the companies’ data that is under consideration. And to get this done in python we had to go through three steps which are:

1. Import: Important libraries which are numpy that is used in python for handling arrays, the matplot libraries which is used in plotting the data and pandas which is a python library that is used in data analysis and for manipulating data.
2. Data: another syntax is used to get the data into the python program
3. Data visualization: to visualize the data as shown in figure 4.2, I plotted the graph using the plot library that was imported into the program. The graph was properly labelled for each company under consideration.

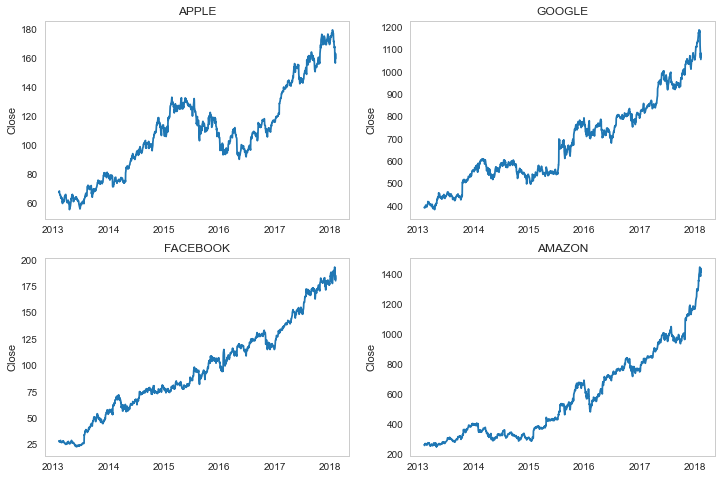


Figure 3.1 closing price stock of each companies studied

Then the Volume of the trading made by this companies for a 5 years period is plotted against time to visualize the junk of data gotten. The volume of trading is the quantity of the total sum of shares which is traded for a specific security and for a specific period of time. This is done to get the data patterns of this companies, with regards to their stock volume. This is shown in figure 3.2

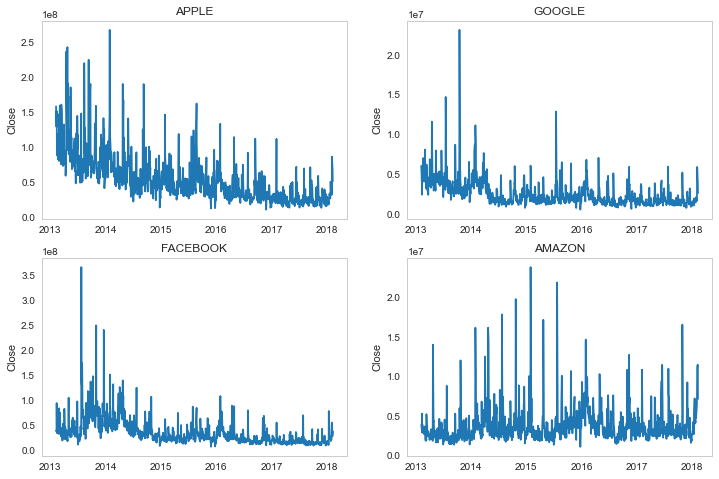


Figure 3.2 volume of stocks against time

The movingaverage (MA) is a simple technical analysis tool that smooths out price data by creating a constantly updated average price. The average is taken over a specific period of time, like 10 days, 20 minutes, 30 weeks or any time period the trader choose. We also plot the bar chart of the moving average using

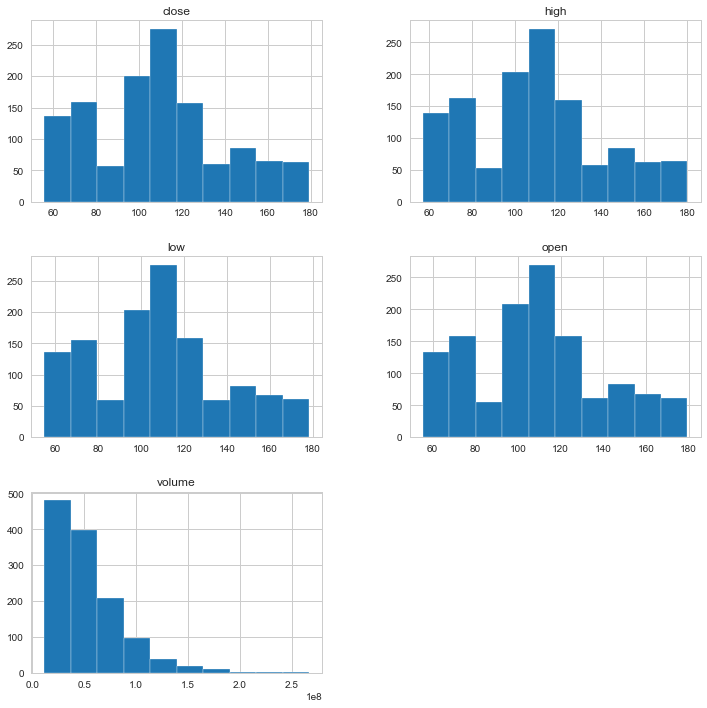


Figure 3.3 Apple’s Moving Average

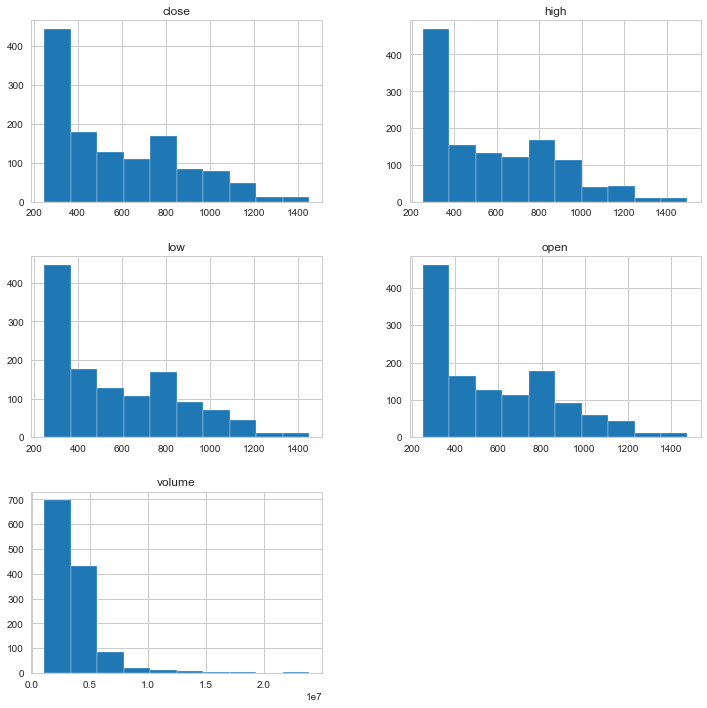


Figure 3.4 Amazon’s Moving Average

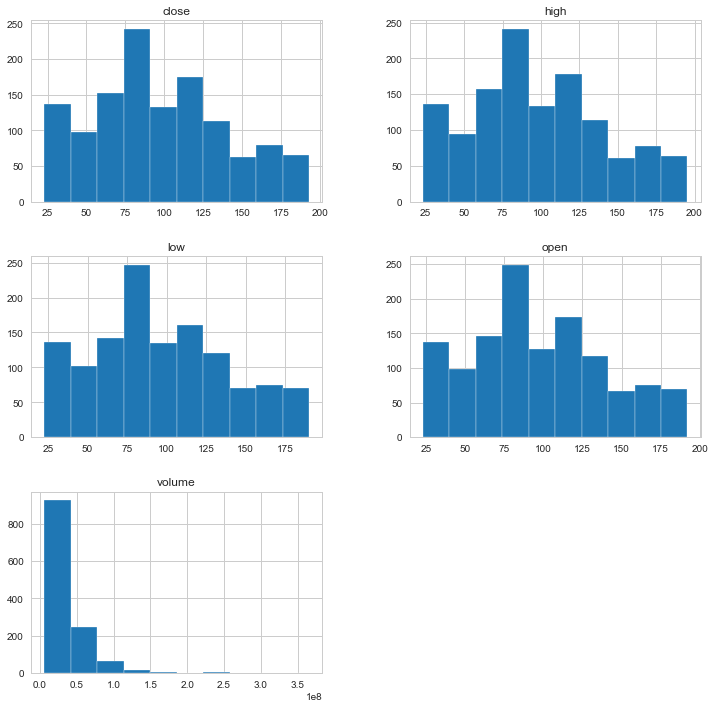


Figure 3.5 Facebook’s moving Average

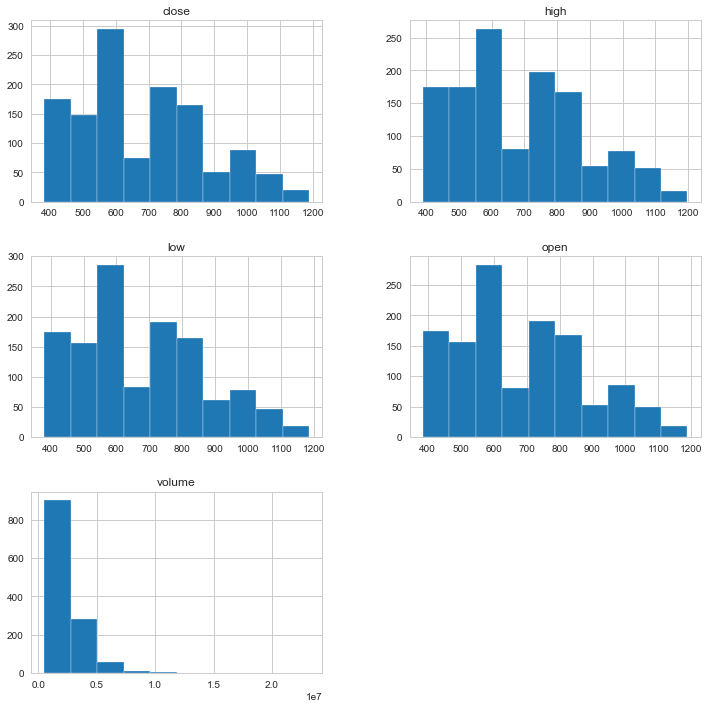


Figure 3.6: Google’s Moving Average



Figure 3.7: Overall plot of the moving average

We also visualize the risk of these companies’ stock. This is informed by the daily return on each stock. To calculate the daily return of stocks for each company we use the formula in python which is:

Where p1 is the current price and p0 is the initial price.

The shift method that is used to shift the index by the value provided as an argument and in this thesis, the values in the sp500 data column is shifted by one which means the current day price is divides by the previous day s&p 500 price. Then plotting graph as shown in the figure 3.7.

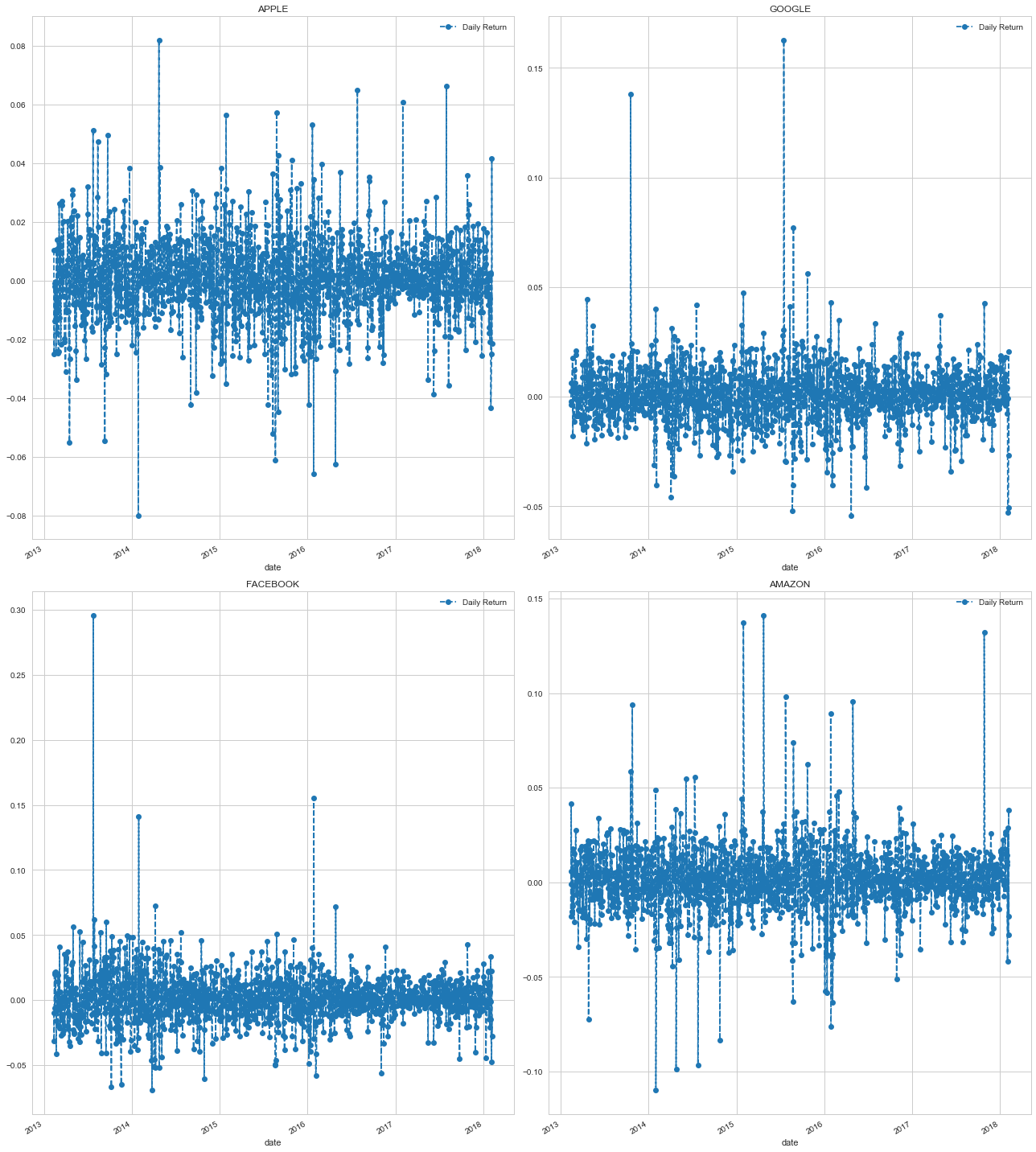


Figure 3.8: Distribution of daily return

Running the stock’s value of each company creates a plot that provides a clearer summary of the distribution of observations. It can be seen that the distribution is a little asymmetrical and perhaps a little pointy to be Gaussian.

Seeing a distribution like this may suggest later exploring statistical hypothesis tests to formally check if the distribution is Gaussian and perhaps data preparation techniques to reshape the distribution, like the Box-Cox transform.

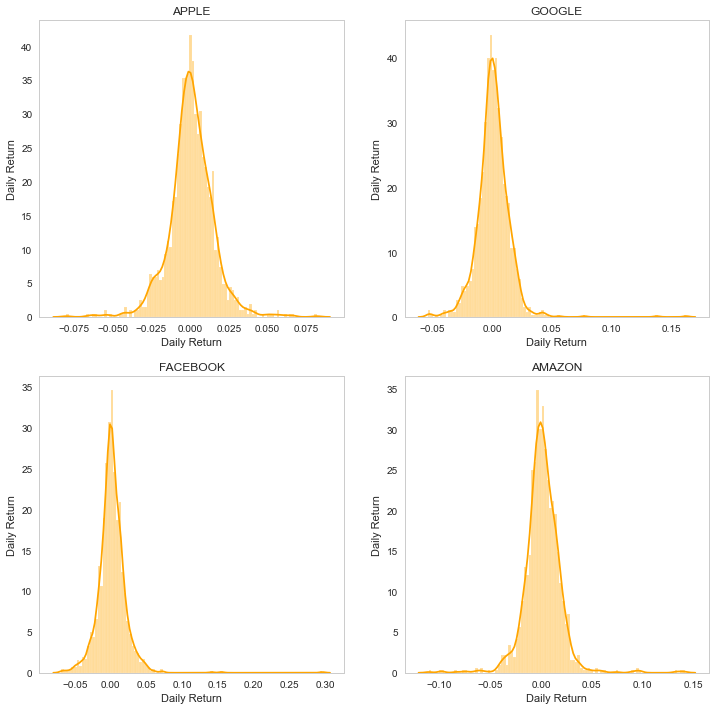


Figure 3.9 Reshaped Moving Average

## 3.4 Data modelling

In other to check the performance of our LSTM stock market forecasting, we develop our stock market prediction with Facebook prophet. The LSTM method is trained using the output of the Facebook prophet because LSTM go back and forth while training the network.

## 3.5 Facebook Prophet

Facebook prophet is an open source software that was released by Facebook for the forecasting time series data based on an additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects.

The following steps were taken to set-up the Facebook Prophet for stock market prediction.

Step 1: Importing the important python library for the data prediction like the pandas and the Facebook prophet.

Step 2: importing the csv of the stocks of this four companies which are used in this thesis and they are Google stocks, Facebook Stocks, Amazon stocks and Apple stocks. To test the first five lines imported, the Head() function was used as shown in the code below:

Step 3: to train the network the fit() method is used train the network and in this experiment 1000 closing price of each company’s stock is used in training.

Step 4: To make the Predictions on a dataframe and on a column ds which contains the dates for which the prediction is made. When getting the suitable dataframe that increase into the future, from 2017 to 2018 year and the helper method was used to achieve this. We use the tail() function to reveal the last number of rows.

Step 5: The predict method assign each row in future a predicted value which it names yhat. Then the historical dates is supplied, then an in-sample fit is provided. The forecast object here is a new dataframe that includes a column yhat with the forecast, as well as columns for components and uncertainty intervals.

Step 6: the forecast is plotted by calling the Prophet.plot method and passing in the forecast dataframe.

## 3.6 RNN Modeling

Recurrent means the output at the current time step becomes the input to the next time step. At each element of the sequence, the model will not just consider the current input, but what it remembers about the preceding elements

LSTM are the most powerful and well know subset of RNN, they are designed to recognize patterns in sequences of data, such as numerical times series data emanating from sensors, stock markets and government agencies (but also including text, genomes, handwriting and the spoken word). LSTM has 3 gates, the input gate, forget gate and output gate.

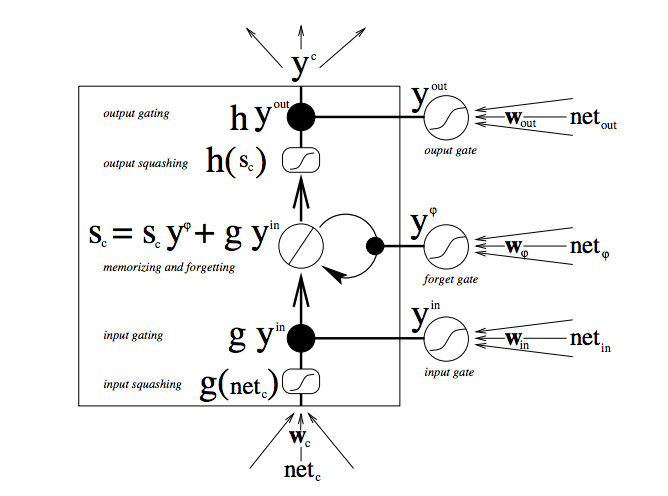


Figure 3.10 Description of the LSTM model Process

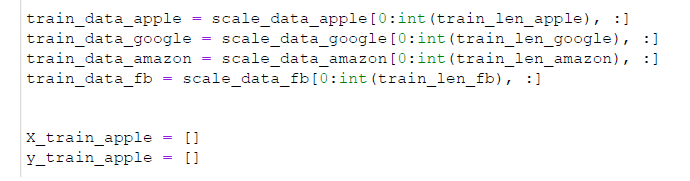
The architecture of our network is displayed below.

After visualizing the data as shown and discussed earlier, there is a need to train the network and the following steps were taken to achieve this:

Step1: the RNN needs to be updated with an input format which is done with the python code shown in the figure below. the structure of the calculations in the RNN, it is noted that there are 3 inputs within your training data.

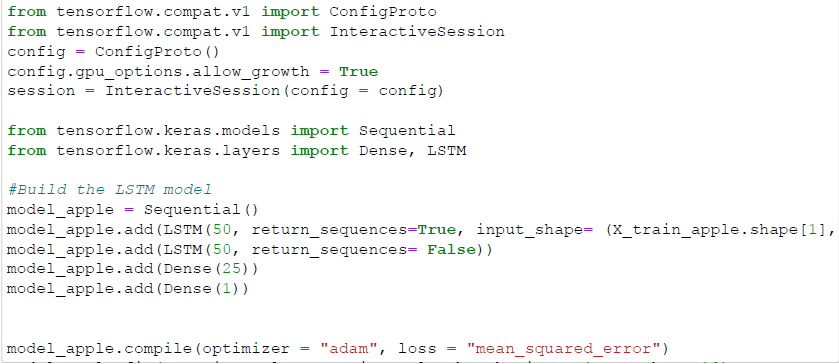
* The vector of input data
* The number of timesteps
* The number of indicators we are looking at

In this case we made use of 1000 sample of input data, the number of time steps is 60 and the number of indicators we are looking at is 1.

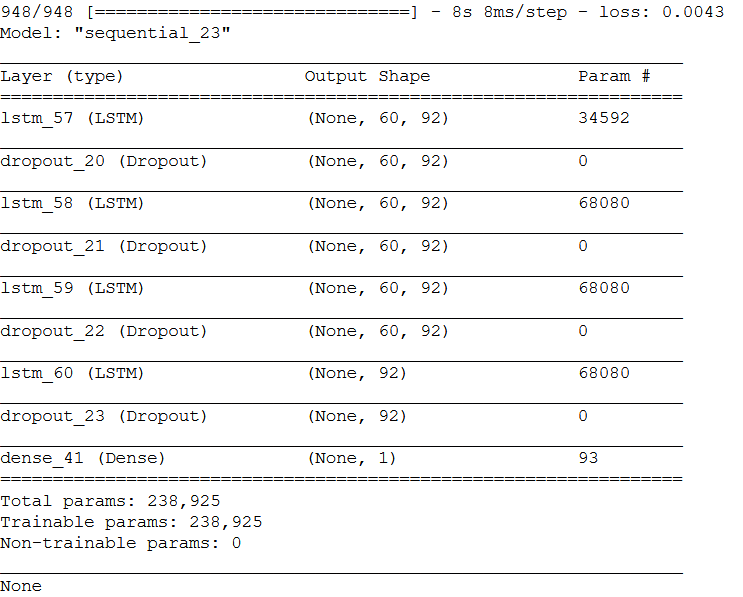


Step 2: Setting up, compiling and fitting the Recurrent Neural Network

In this stage the libraries that will be used to set up the RNN are declared and example of this is the kerars library, tensorflow.compat.v1. Then RNN was initialized and it was compiled as shown in the figure below;



The root mean square error is plotted to see the squared of the average difference between the estimated values and the actual value. So as to give the accurate or close to accurate prediction. This step is done for the close price of all the four companies that are chosen as indicated above. The architecture of the system is shown below:



Before fitting the model, LSTM accepts inputs in a format different from that of prophet. The input shape for lstm is (samples, time steps, features).

The number of epochs is the number of complete passes through the training dataset and three different epochs were used.

We can then proceed with the modeling. Modeling entails passing the training data through the architecture and predicting on the test data.

Step 3: making the predictions

So, we tried to predict 60-time step into the future.

To do this we need to preprocess our data and make it in the format above.

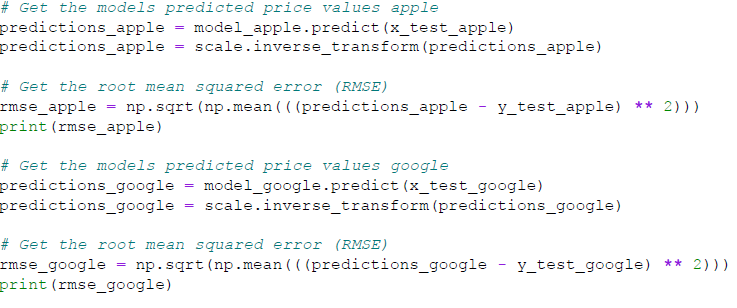
### 

### 3.6.1 Preprocessing

1. Scaling: Our dataset contains large values ranging from 0 to inf, hence we scale the data into the range of (0,1) such that the lowest value is 0 and the highest value is 1.
2. Reshape: After scaling, we reshape the data into the format above (sample, time-step, features)
3. Splitting: we split the data into train and test in the ratio of 80:20

All these steps are applied on the apple, google, Facebook and amazon data

As shown in the figure below:



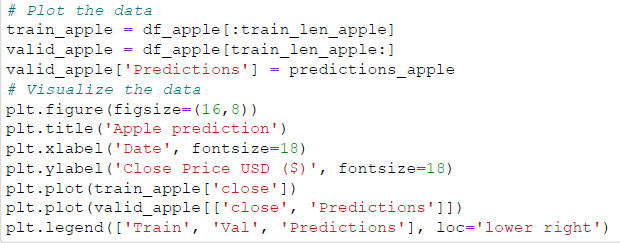
Hyperparameters used in compiling the model are:

Optimizer: **Adam optimizer**

Loss: **Mean\_Squared\_error**

**Step 4:** Visualizing the predicted data

The predicted data are now plotted using the matplot library in python so as to visualize the data and for the data to look presentable to a lay man. This process is done using the code shown below:



### 3.6.2 Data Mining and Machine Learning Tools:

The best python data mining tools are used in this thesis and examples of them are numpy, scipy and python plotting – Matplotlib.

The numpy library is used because, it provides a high-performance of multidimensional array and basic tools to compute with/and manipulate these arrays. While SciPy builds on this, and provides a large number of functions that operate on numpy arrays and are useful for different types of scientific and engineering applications and matplot is used to visualize the data.

## 3.7 Python in Data Science

Python is a very robust programming language

* the premier platform for data science competitions said that python In 2016 overtook R on Kaggle,.

# 

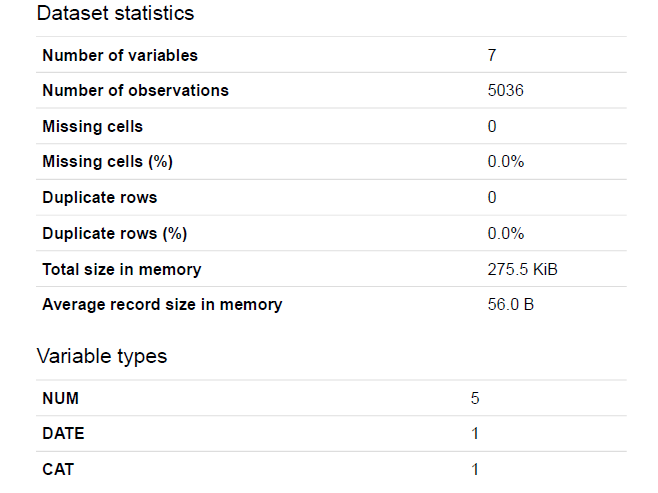
# CHAPTER FOUR: RESULTS AND EVALUATION

## 4.0 INTRODUCTION

This section shows the different tables and charts that we derived in predicting the closing stock price of each company stock (i.e. stock price of Facebook, Amazon, Apple and Google). The experiment was conducted with real stock price and the performance of the Facebook Prophet model from Facebook and the LSTM model which was designed in this thesis are analyzed in this section.

## 4.1 Experimental Setup

The experiment was conducted with real stock market data from standard and poor’s 500 index. The Experiment were conducted at different stages with the first stage is to setup the control experiment and the postulated model which is the Recurrent Neural Network of the Long- and Short-term Network is later setup to predict stock market prices.

 Fig 4.1 Table showing the statistics of the dataset of the S&P 500 of the four companies

## 

## Fig 4.2 Flow chart of RNN

## 4.2 Facebook Prophet

While using the Facebook prophet to predict the stock prices, the graph below shows the resulting predicted data in green and the data that is used to train the network in blue color and the actual data in orange color. We could see that the Facebook prophet could predict the behavior of the stock prices of each of the companies for the year 2017 correctly. As to the exact stock market close price, Facebook prophet could not predict the exact close price.

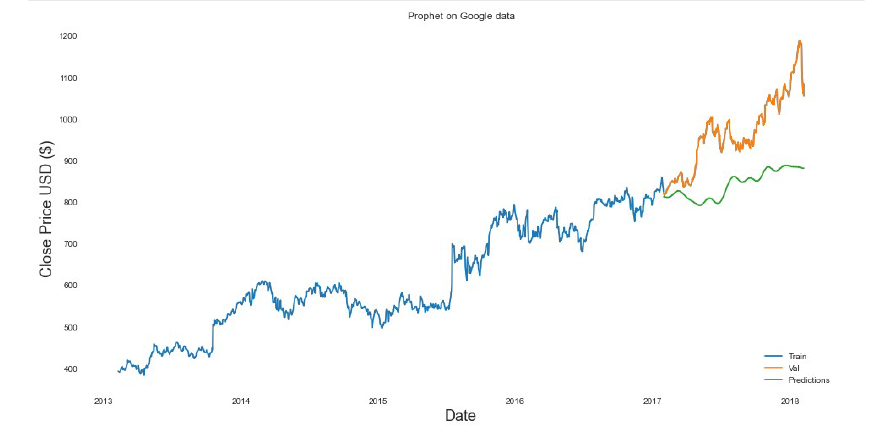


Fig 4.3 Facebook prophet graph on Google data for 2017

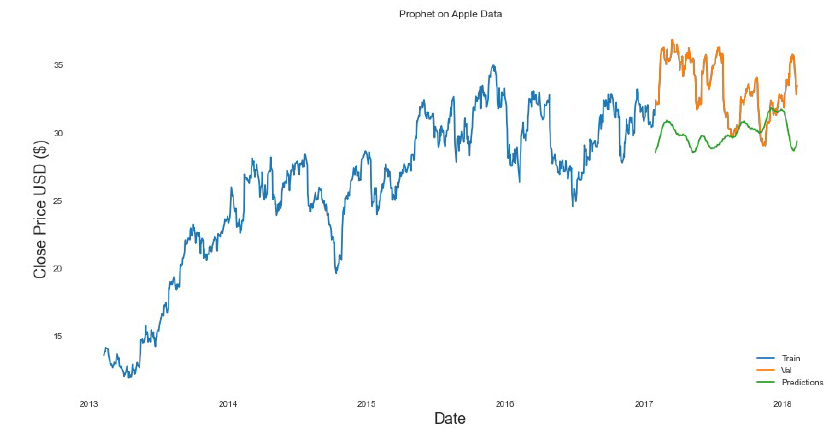


Fig 4.4 Facebook prophet graph on Apple data for 2017

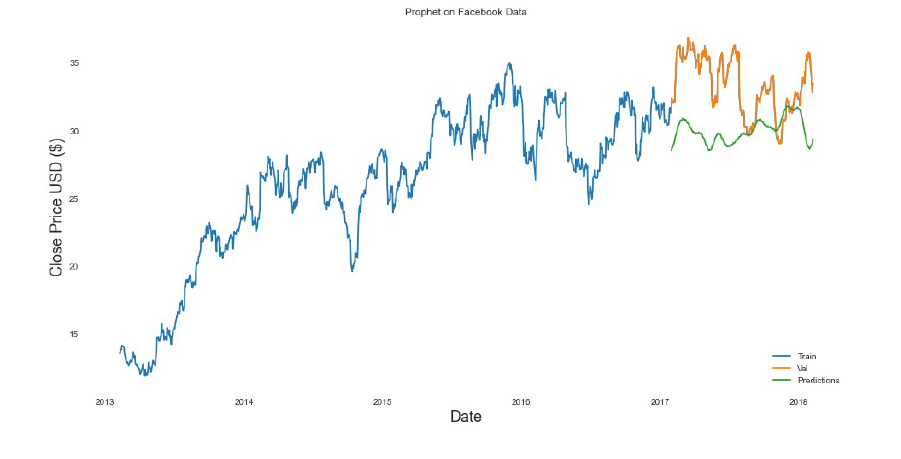
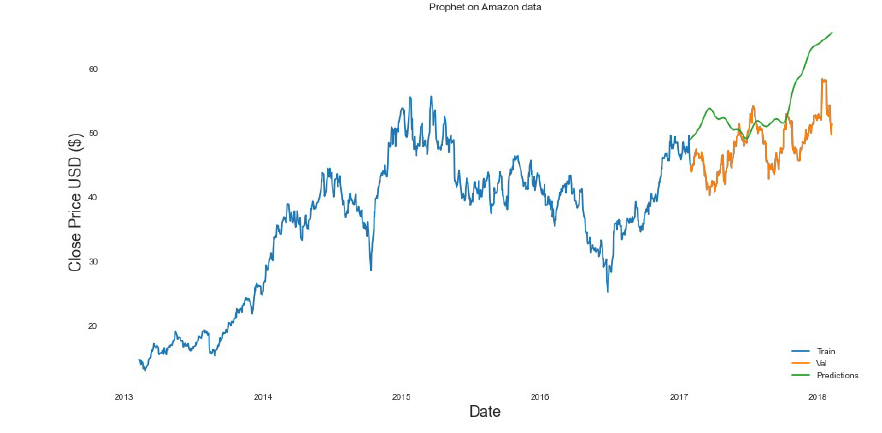


Fig 4.5 Facbook prophet graph on Facebook data for 2017

Fig 4.6 Facebook prophet graph on Amazon data for 2017

## 

Fig 4.7 Flow chart of Facebook prophet

## 4.3 RECURRENT NEURAL NETWORK(LSTM)

The LSTM was designed using the Jupiter programing environment and the programing language used is the python programming language.

The data shown below are the data gotten when the network was trained using 60 data.

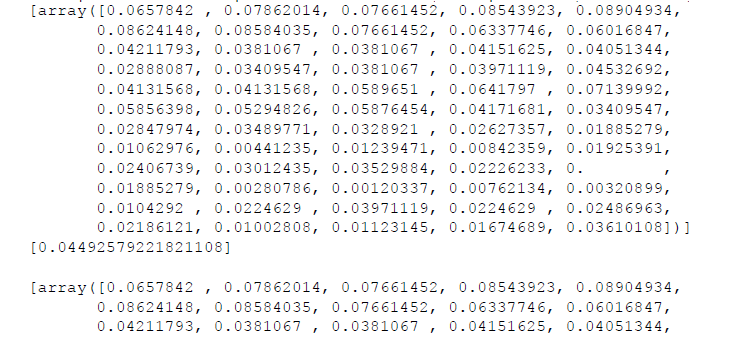
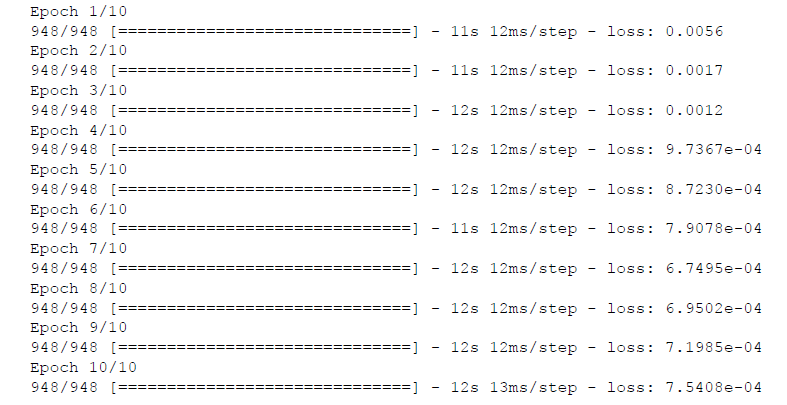
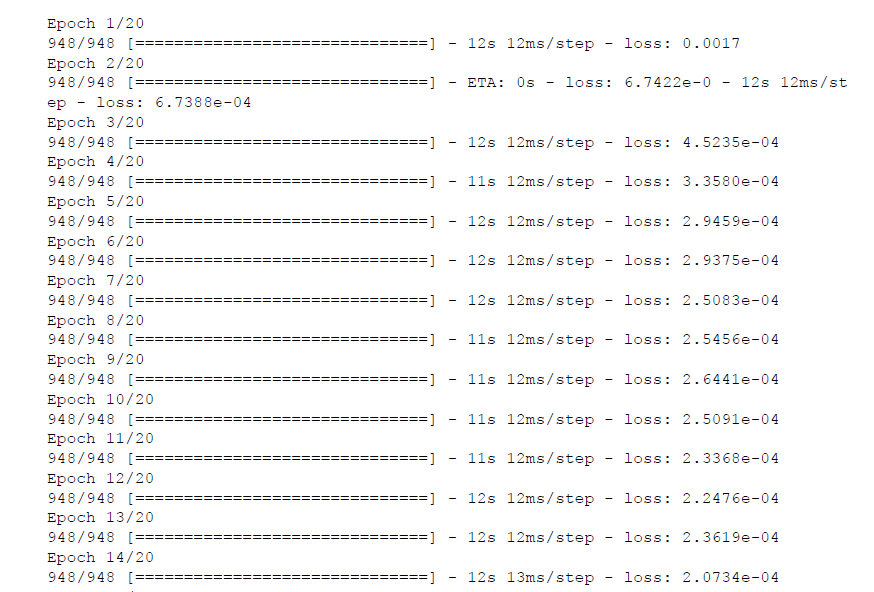


Fig 4.8 LSTM data gotten when the network was trained using 60 data

As the training vectors are used, we could see that the epoch is increasing. The figures below shows the resulting values, time used in carrying out the task and the losses. It is noticed that the error shown is so small that it can be overlooked.





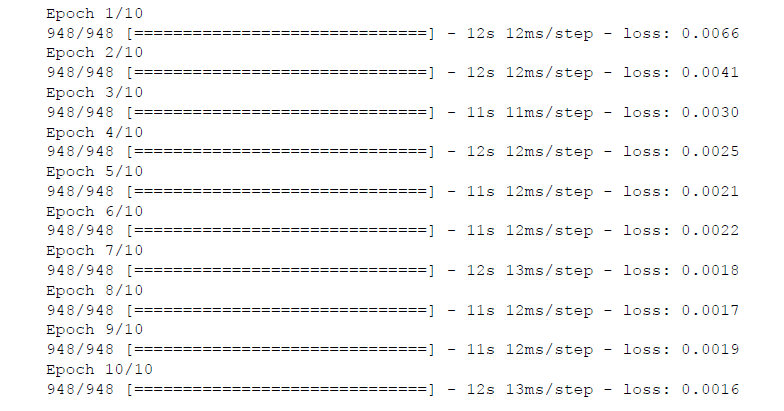


Fig 4.9 The figures showing the resulting values, time used in carrying out tasks and the losses as training vectors are used

### 

### Fig 4.10 Flow chart of LSTM

### 4.3.1 ACCURACE

To know the accuracy of the experiment carried out using the LSTM to predict data, we had to look for the Moving Average of the network as shown in figure below. The moving average for 10, 20 and 50 days. It is noticed that the moving average is between 29 to 34. This shows that the Recurrent network is effective in processing stock data, which is a model that can adaptively de-noise the noisy data and can reduce the dimensionality.

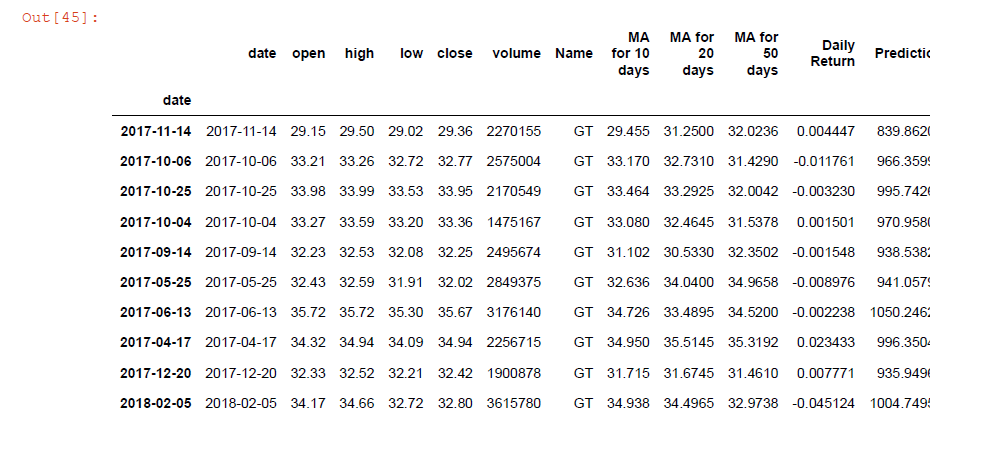


Fig 4.11 Moving average of the network for 10,20 and 50 days

### 4.3.2 Prediction

The prediction made by the application are shown on the graph below which is a plot of the closing price against the period of time. The close price prediction graph is plotted for the close stock price for Apple stocks, Google stocks, Facebook stocks and Amazon stocks. It is noticed that the previous data helps to train the network and the trained network is used to predict the real stock price and it is noticed that the accuracy is top notch. The values predicted are accurate, as it can be seen on the graph.



Fig 4.12 Close price prediction graph of apple stocks

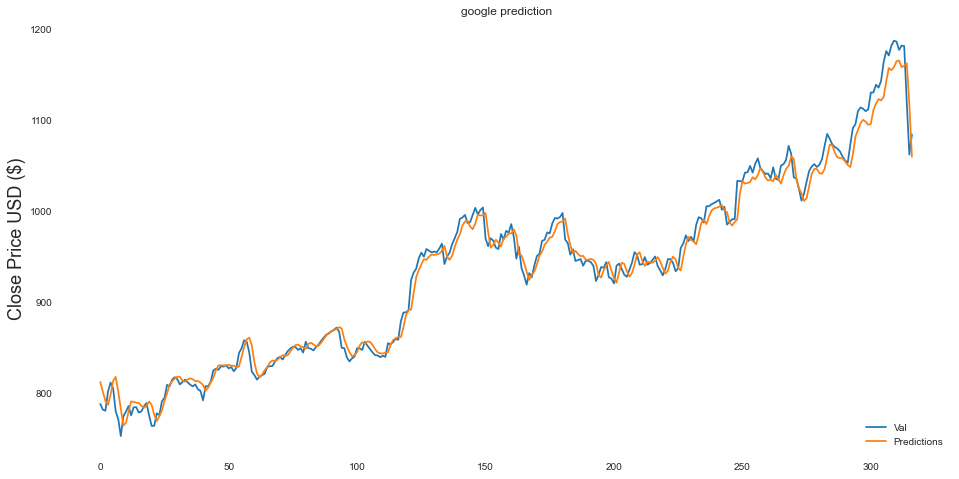


Fig 4.13 Close price prediction graph of google stocks

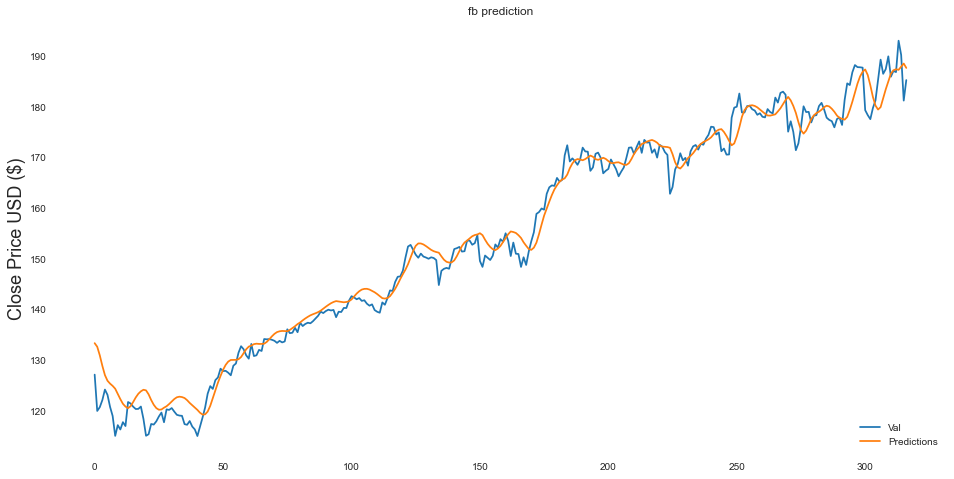


Fig 4.14 Close price prediction of facebook stock

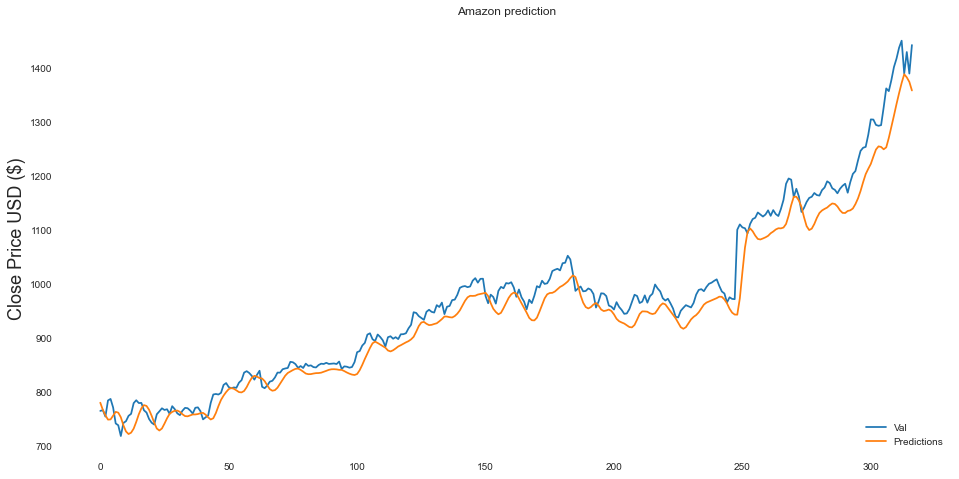


Fig 4.15 Close price prediction of Amazon stock

# CHAPTER FIVE: CONCLUSION AND RECOMMENDATION

## 5.1 CONCLUSION

Recurrent Neural Network (RNN), LSTM is unique and very powerful when it comes to sequence prediction and as it is shown in this thesis the results are accurate. In using the lstm which stores the data of the last in put to use it in the new input so as to predict an accurate data. The data used in this thesis were real data gotten from the S&P 500 index of the four companies which are Amazon, Facebook, Apple and google stocks. Despite the facts that other scholars had used different method to predict stock market. Bylander and Schwaerzel (2015) have predicted financial time series (FTS) by GP with high-order statistics and trigonometric functions. When the performance analysis was considered for the extra FTS which is Financial Time series indicates that basic GP models plus the sets of valued-added function perform excellently okay than that of the Buy and Hold strategy, Akaike selected models of ARMA and GP model. Merh et al (2010) compared hybrid approaches of ARIMA and ANN for predicting future value of index and SM in India. It was shown in the journal that hybrid ANN\_ARIMA and BSE oil and gas ANN were able to control the input set of data and forecast future closing price while using the ARIMA\_ANN and ARIMA could not predict future values. Deu to the facts that LSTM had a memory for storing previous hidden state output for each input time stem makes it unique in predicting stock market.

Data were gotten from the S&P 500 index which are real data of stock prices of this companies and data were plotted against time. The moving average were also plotted against time so as to see the data trends on graph. Facebook prophet which is a data analysis library from facebook was used to predict the stock prices over a of 2017 to 2018 and the resulted was plotted on graph to see the comparism of the actual data and the predicted data. Which serves as the control experiment for our experiment.

The LSTM model was declared in python Jupiter and the libraries that was used are all state in the methodology. The network was trained using data from the year 2013 to the year 2016 December. The moving average of the predicted value was calculated and shown in the data table. And we could notice and deduced that the LSTM could not only predict the stock market behavior but also predicted the real value of the closing price of stock market of this four companies.

## 5.2 Recommendation and Future works

The Artificial Neural Network that was used to developed from this research has proven that the efficient way of predicting stock market is to use the LSTM which is long shot time memory Recurrent neural network. So therefore, it is recommended for researchers which are in the financial sector to use in making decision about investment. It is also recommended for investors who wants to in company stocks, so as to see before hand how a company stock will grow positively or negatively in the nearest future.

Future works could find the accuracy of the predicted stocks and the rate of change of the predicted value to as to get the things that can destabilize the stock market that is stabilized. Evaluation of the system could be done on a larger dataset.

## 5.3 Contribution to Knowledge

This research had contributed greatly to my knowledge about stock market, python and how Artificial neural network works. This research has also contributed to knowledge by establishing that LSTM can be used to predict the future of stock prices of Large-scale companies and small-scale companies which investors can use as a tool when trying to make the decision on which stocks to invest their resources into.

# References

Aamodt.R. (2010). Using ANN to Forecast Financial Time Series. *Computers and Operations*, 32: 2399–410.

Adebiyi, A. A., Adewumi, A. O., & Ayo, C. K. (2014). Stock Price Prediction Using the ARIMA Model. *Institute of Electrical and Electronics Engineers* (pp. 106-112). Cambirdge: Institute of Electrical and Electronics Engineers.

al, A. e. (2009). Forecasting ER: A NF approach. *IFSA-EUSFLAT*.

al, D. e. (n.d.). FL Expert Advisor Topology for Foreign exchange market. *International Conference on Software Engineering and Intelligent Systems.*

al, M. e. (2010). Comparison between Hybrid Approaches of ANN and ARIMA for Indian Stock Trend Forecasting. *Business Intelligence Journal*, 23-43.

Andrej Krenker, J. B. (2009). *Introduction to the Artificial Neural Networks.* London: Routledge.

Bhatt.K, M. a. (2015). Developing a model for financial forecasting through ANN. *International Journal of Engineering Research and General Science*, vol 3, iss 2, pp:44-54.

Brownlee, J. (2017, May 24). *A Gentle Introduction to Long Short-Term Memory Networks by the Experts.* Retrieved from Machine Learning Mastery: https://machinelearningmastery.com/gentle-introduction-long-short-term-memory-networks-experts/

Cao, Q. K. (2005). A Comparison Between Fama and French‟s Model and Artificial Neural Networks in Predicting the Chinese Stock Market. *Computers and Operations Research*, 32: 2499–512.

Dannie, H. R. (2006). *Computational Genomics and Molecular Biology.* Pittsburgh: Carnegie Mello University.

Davydova, O. (2017, September 26). *7 types of Artificial Neural Networks for Natural Language Processing.* Retrieved from Data Monsters: https://medium.com/@datamonsters/artificial-neural-networks-for-natural-language-processing-part-1-64ca9ebfa3b2

Diaz.M. (2020, July 10). *Speculative Strategies in foreign ER based on GP Predictions.* Retrieved from Digital Access: http://dea.uib.es/digitalAssets/123/123154\_cvmarcosalvarez2.pdf

Enke, D. A. (2005). The Use Of Data Mining And Neural Networks For Forecasting Stock. *Expert Systems With Applications*, 29, 927–940.

Erdogan.O and Goksu.A. (2014). Forecasting Euro and Turkish Lira Exchange Rates with ANN. *International Journal of Academic Research in Accounting, Finance and Management Sciences*, vol 4, no 4, pp: 307-316,.

Eun, C. S. (1989). International transmission of stock. *Journal of Financial and Quantitative Analysis*, 24, 41-56.

Gradojevic.N et al. ( 2015, April 28th). *Neuro-Fuzzy Decision-making in Foreign Exchange Trading in Other Applications,*. Retrieved from Economics: http://economics.ca/2002/pdf/0100.pdf

Graves, A., & Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks.* , 18, (5–6), 602–610.

James Royal, P. (2020, June 16). *How to Day Trade.* Retrieved from Nedwallet : https://www.nerdwallet.com/article/investing/how-to-day-trade-safely

Kablan, A. (2009). Adaptive Neuro Fuzzy Inference Systems for High Frequency Financial Trading and Forcasting. *Third International Conference on Advanced Engineering Computing and Applications in* (pp. 105 –110,). Los Alamitos, CA, USA: IEEE Computer Society.

Kamruzzaman, S. a. (2003). *Comparing ANN based Models with ARIMA for Predicting of Forex Rates.*

Li, T. E. (1999). Improving technical analysis predictions: an application of genetic programming,. *12th International Florida AI Research Society Conference* (pp. 108-111). Florida: Orlando.

Liu.W. (2008). Forecasting ER change between USD and JPY Using Dynamic Adaptive Neuron-Fuzzy Logic. *Asia Pacific Journal of Finance and Banking Research*, vol 2, no 2, pp: 1-12.

Mittal, H. B. (2009). Software maintenance productivity assessment using fuzzy logic. *ACM SIGSOFT Software Engineering Notes*, pp. 1 -4.

Nadh, V. L., & Prasad, G. S. (2018). Stock Market Prediction Based on Machine Learning Approaches. *Computational Intelligence and Big Data Analytics SpringerBriefs in Applied Sciences and Technology*, 75-79.

Onwuka, E. N., Salihu, B. A., & Abdulrahman, I. A. (2017). Enhanced Subscriber Churn Prediction Model for the Mobile Telecommunication Industry. *ATBU, Journal of Science, Technology & Education*, 67-74.

Pacelli.V. (2012). Forecasting Exchange Rates: A Comparative Analysis. *International Journal of Business and Social Science*, vol 3, no 10, pp: 145-156,.

Pendharkar, P. (2005). Threshold-varying Artificial Neural Network Approach for Classification and its Application to Bankruptcy Prediction Problem. *Computers and Operations Research*, 2561–82.

R., L. R. (1996). Neural Networks: A systematic Introduction, Springer. *Springer*, 3-5.

Ravi.V et al. (2012). Foreign ER Prediction using Computational Intelligence Methods. *International Journal of Computer Information Systems and Industrial Management Applications*, vol 4, pp: 659-670,.

Samek, D., & Varacha, P. (2013). Time series prediction using artificial neural networks: single and multi-dimensional data . *INTERNATIONAL JOURNAL OF MATHEMATICAL MODELS AND METHODS IN APPLIED SCIENCES* , 38-46.

SRIVASTAVA, P. (2017, December 10). *Essentials of Deep Learning : Introduction to Long Short Term Memory.* Retrieved from Analytic Vidya: https://www.analyticsvidhya.com/blog/2017/12/fundamentals-of-deep-learning-introduction-to-lstm/

*Stock Market* . (2020). Retrieved from Corporate Finance Institute: https://corporatefinanceinstitute.com/resources/knowledge/trading-investing/stock-market/

*Stock market prediction*. (2020, 04 11). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Stock\_market\_prediction

Susmel, R. a. (1994). Hourly volatility spillovers between international equity markets. *Journal of International Money and Finance*, 13, 3-25.

Taiwo.S, L. a. (2015, April 28). *Enhancing Financial Decision Making using Multi-objective financial GP*. Retrieved from MOFGP-JinSope: http://www.cs.bham.ac.uk/~wbl/biblio/cache/http\_\_\_www.cs.bham.ac.uk\_\_jxl\_cercialink\_web\_publication

Valavanis.K.P, A. a. (2008). *Surveying Stock Market Forecasting Techniques, Soft Computing Methods, Expert System with Applications.*

Yochim, D. (2020, June 16). *Stock Trading: How to Begin, How to Survive.* Retrieved from Nerdwallet: https://www.nerdwallet.com/article/investing/stock-trading-how-to-begin

# APPENDIX

In [2]:

**import** numpy **as** np

**import** pandas **as** pd

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

**import** datetime **as** dt

**from** fbprophet **import** Prophet

*# Statsmodels widely known for forecasting than Prophet*

**import** statsmodels.api **as** sm

**from** scipy **import** stats

**import** plotly.offline **as** py*#visualization* py.init\_notebook\_mode()*#visualization* **import** plotly.graph\_objs **as** go*#visualization* **import** plotly.tools **as** tls*#visualization*

**import** plotly.figure\_factory **as** ff*#visualization*

**import** os *#init\_notebook\_mode(connected=True)* **import** warnings warnings.filterwarnings("ignore")

*# plt.style.available*

In [3]:

df **=** pd.read\_csv("Dataset/all\_stocks\_5yr.csv") df["date"] **=** pd.to\_datetime(df['date'])

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Out[3]: |  | | | | | | | |
|  |  | **date** | **open** | **high** | **low** | **close** | **volume** | **Name** |
|  | **0** | 2013-02-08 | 15.07 | 15.12 | 14.63 | 14.75 | 8407500 | AAL |
|  | **1** | 2013-02-11 | 14.89 | 15.01 | 14.26 | 14.46 | 8882000 | AAL |
|  | **2** | 2013-02-12 | 14.45 | 14.51 | 14.10 | 14.27 | 8126000 | AAL |
|  | **3** | 2013-02-13 | 14.30 | 14.94 | 14.25 | 14.66 | 10259500 | AAL |
|  | **4** | 2013-02-14 | 14.94 | 14.96 | 13.16 | 13.99 | 31879900 | AAL |
| In [4]:  Out[4]: |  |  |  |  |  |  |  |  |

array(['AAL', 'AAPL', 'AAP', 'ABBV', 'ABC', 'ABT', 'ACN', 'ADBE', 'ADI',

'ADM', 'ADP', 'ADSK', 'ADS', 'AEE', 'AEP', 'AES', 'AET', 'AFL',

'AGN', 'AIG', 'AIV', 'AIZ', 'AJG', 'AKAM', 'ALB', 'ALGN', 'ALK',

'ALLE', 'ALL', 'ALXN', 'AMAT', 'AMD', 'AME', 'AMGN', 'AMG', 'AMP',

'AMT', 'AMZN', 'ANDV', 'ANSS', 'ANTM', 'AON', 'AOS', 'APA', 'APC',

'APD', 'APH', 'APTV', 'ARE', 'ARNC', 'ATVI', 'AVB', 'AVGO', 'AVY',

'AWK', 'AXP', 'AYI', 'AZO', 'A', 'BAC', 'BAX', 'BA', 'BBT', 'BBY',

'BDX', 'BEN', 'BF.B', 'BHF', 'BHGE', 'BIIB', 'BK', 'BLK', 'BLL',

'BMY', 'BRK.B', 'BSX', 'BWA', 'BXP', 'CAG', 'CAH', 'CAT', 'CA',

'CBG', 'CBOE', 'CBS', 'CB', 'CCI', 'CCL', 'CDNS', 'CELG', 'CERN',

'CFG', 'CF', 'CHD', 'CHK', 'CHRW', 'CHTR', 'CINF', 'CI', 'CLX',

'CL', 'CMA', 'CMCSA', 'CME', 'CMG', 'CMI', 'CMS', 'CNC', 'CNP',

'COF', 'COG', 'COL', 'COO', 'COP', 'COST', 'COTY', 'CPB', 'CRM',

'CSCO', 'CSRA', 'CSX', 'CTAS', 'CTL', 'CTSH', 'CTXS', 'CVS', 'CVX',

'CXO', 'C', 'DAL', 'DE', 'DFS', 'DGX', 'DG', 'DHI', 'DHR', 'DISCA',

'DISCK', 'DISH', 'DIS', 'DLR', 'DLTR', 'DOV', 'DPS', 'DRE', 'DRI',

'DTE', 'DUK', 'DVA', 'DVN', 'DWDP', 'DXC', 'D', 'EA', 'EBAY',

'ECL', 'ED', 'EFX', 'EIX', 'EL', 'EMN', 'EMR', 'EOG', 'EQIX',

'EQR', 'EQT', 'ESRX', 'ESS', 'ES', 'ETFC', 'ETN', 'ETR', 'EVHC',

'EW', 'EXC', 'EXPD', 'EXPE', 'EXR', 'FAST', 'FBHS', 'FB', 'FCX',

'FDX', 'FE', 'FFIV', 'FISV', 'FIS', 'FITB', 'FLIR', 'FLR', 'FLS',

'FL', 'FMC', 'FOXA', 'FOX', 'FRT', 'FTI', 'FTV', 'F', 'GD', 'GE',

'GGP', 'GILD', 'GIS', 'GLW', 'GM', 'GOOGL', 'GOOG', 'GPC', 'GPN',

'GPS', 'GRMN', 'GS', 'GT', 'GWW', 'HAL', 'HAS', 'HBAN', 'HBI',

'HCA', 'HCN', 'HCP', 'HD', 'HES', 'HIG', 'HII', 'HLT', 'HOG',

'HOLX', 'HON', 'HPE', 'HPQ', 'HP', 'HRB', 'HRL', 'HRS', 'HSIC',

'HST', 'HSY', 'HUM', 'IBM', 'ICE', 'IDXX', 'IFF', 'ILMN', 'INCY',

'INFO', 'INTC', 'INTU', 'IPG', 'IP', 'IQV', 'IRM', 'IR', 'ISRG',

'ITW', 'IT', 'IVZ', 'JBHT', 'JCI', 'JEC', 'JNJ', 'JNPR', 'JPM',

'JWN', 'KEY', 'KHC', 'KIM', 'KLAC', 'KMB', 'KMI', 'KMX', 'KORS',

'KO', 'KR', 'KSS', 'KSU', 'K', 'LB', 'LEG', 'LEN', 'LH', 'LKQ',

In [5]:

<class 'pandas.core.frame.DataFrame'> RangeIndex: 619040 entries, 0 to 619039 Data columns (total 7 columns):

# Column Non-Null Count Dtype

1. date 619040 non-null datetime64[ns]
2. open 619029 non-null float64
3. high 619032 non-null float64
4. low 619032 non-null float64
5. close 619040 non-null float64
6. volume 619040 non-null int64
7. Name 619040 non-null object

dtypes: datetime64[ns](1), float64(4), int64(1), object(1) memory usage: 33.1+ MB

In [6]:

df\_apple **=** df[df["Name"] **==** "GT"] df\_apple.index **=** df\_apple["date"] df\_google **=** df[df["Name"] **==** "GOOGL"] df\_google.index **=** df\_google["date"] df\_fb **=** df[df["Name"] **==** "GS"] df\_fb.index **=** df\_fb["date"] df\_amazon **=** df[df["Name"] **==** "AAL"] df\_amazon.index **=** df\_amazon["date"]

company\_list **=** [df\_apple, df\_google, df\_fb, df\_amazon]

In [7]:

In [8]:

big\_4 **=** pd.concat([df\_apple, df\_google, df\_fb, df\_amazon], axis **=** 0)

Out[8]:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **date** | **date** | **open** | **high** | **low** | **close** | **volume** | **Name** |
| **2013-02-08** | 2013-02-08 | 13.62 | 13.780 | 13.32 | 13.59 | 5875887 | GT |
| **2013-02-11** | 2013-02-11 | 13.70 | 13.930 | 13.53 | 13.91 | 7967771 | GT |
| **2013-02-12** | 2013-02-12 | 13.55 | 14.150 | 13.10 | 13.86 | 12293423 | GT |
| **2013-02-13** | 2013-02-13 | 13.90 | 14.175 | 13.90 | 14.08 | 7092499 | GT |
| **2013-02-14** | 2013-02-14 | 14.00 | 14.200 | 13.93 | 14.17 | 4514455 | GT |

In [9]:

**import** pandas\_profiling **as** pf

Summarize dataset: 100% 21/21 [00:19<00:00, 1.10it/s, Completed]

Generate report structure: 100% 1/1 [00:13<00:00, 13.37s/it]

Render HTML: 100% 1/1 [00:03<00:00, 3.29s/it]

Overview

|  |  |  |
| --- | --- | --- |
| Dataset statistics |  | |
| **Number of variables** |  | 7 |
| **Number of observations** |  | 5036 |
| **Missing cells** |  | 0 |
| **Missing cells (%)** |  | 0.0% |
| **Duplicate rows** |  | 0 |
| **Duplicate rows (%)** |  | 0.0% |
| **Total size in memory** |  | 275.5 KiB |
| **Average record size in memory**  Variable types |  | 56.0 B |
| **NUM** | 5 |  |
| **DATE** | 1 |  |
| **CAT** | 1 |  |

Reproduction

**Analysis started**

**Analysis finished**

2020-07-26 14:19:37.542476

2020-07-26 14:19:56.051992

Out[9]:

In [10]:

*# plot of big\_4 close prices*

plt.figure(figsize**=**(12, 8))

*#plt.subplots\_adjust(top=1.25, bottom=1.2)*

**for** i, company **in** enumerate(company\_list, 1): plt.subplot(2, 2, i) plt.plot(company['date'], company["close"]) plt.ylabel('Close')

plt.xlabel(**None**)

**if** i **==** 1:

plt.title(f"APPLE")

**elif** i **==** 2:

plt.title(f"GOOGLE")

**elif** i **==** 3:

plt.title(f"FACEBOOK")

**else**:

plt.title(f"AMAZON")



In [11]:

*# plot of big\_4 close stock volume*

plt.figure(figsize**=**(12, 8))

*#plt.subplots\_adjust(top=1.25, bottom=1.2)*

**for** i, company **in** enumerate(company\_list, 1): plt.subplot(2, 2, i) plt.plot(company['date'], company["volume"]) plt.ylabel('Close')

plt.xlabel(**None**)

**if** i **==** 1:

plt.title(f"APPLE")

**elif** i **==** 2:

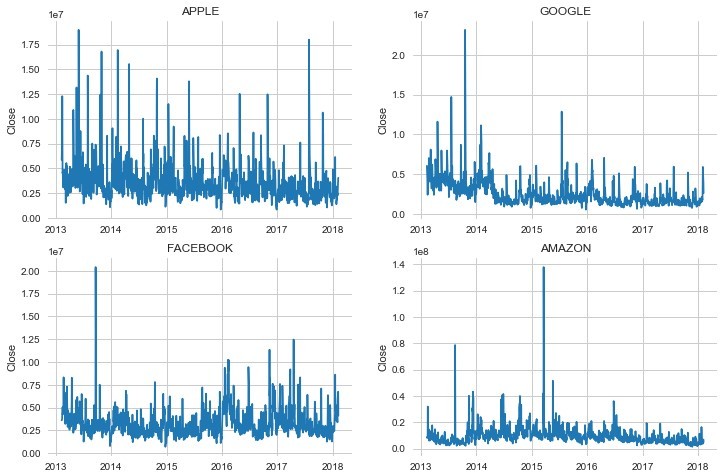
plt.title(f"GOOGLE")

**elif** i **==** 3:

plt.title(f"FACEBOOK")

**else**:

plt.title(f"AMAZON")



In [12]:

*# what are the moving averages for each company stock*

moving\_average **=** [10, 20, 50]

**for** ma **in** moving\_average:

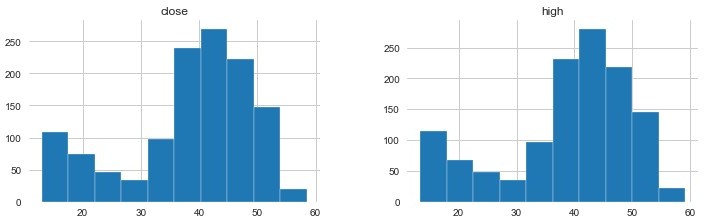
**for** company **in** company\_list: column\_name **=** f"MA for {ma} days"

company[column\_name] **=** company['close'].rolling(ma).mean()

Out[12]: Name

AAL [[AxesSubplot(0.125,0.670278;0.336957x0.209722... GOOGL [[AxesSubplot(0.125,0.670278;0.336957x0.209722... GS [[AxesSubplot(0.125,0.670278;0.336957x0.209722... GT [[AxesSubplot(0.125,0.670278;0.336957x0.209722...

dtype: object



In [13]:

fig, axes **=** plt.subplots(nrows**=**2, ncols**=**2) fig.set\_figheight(20)

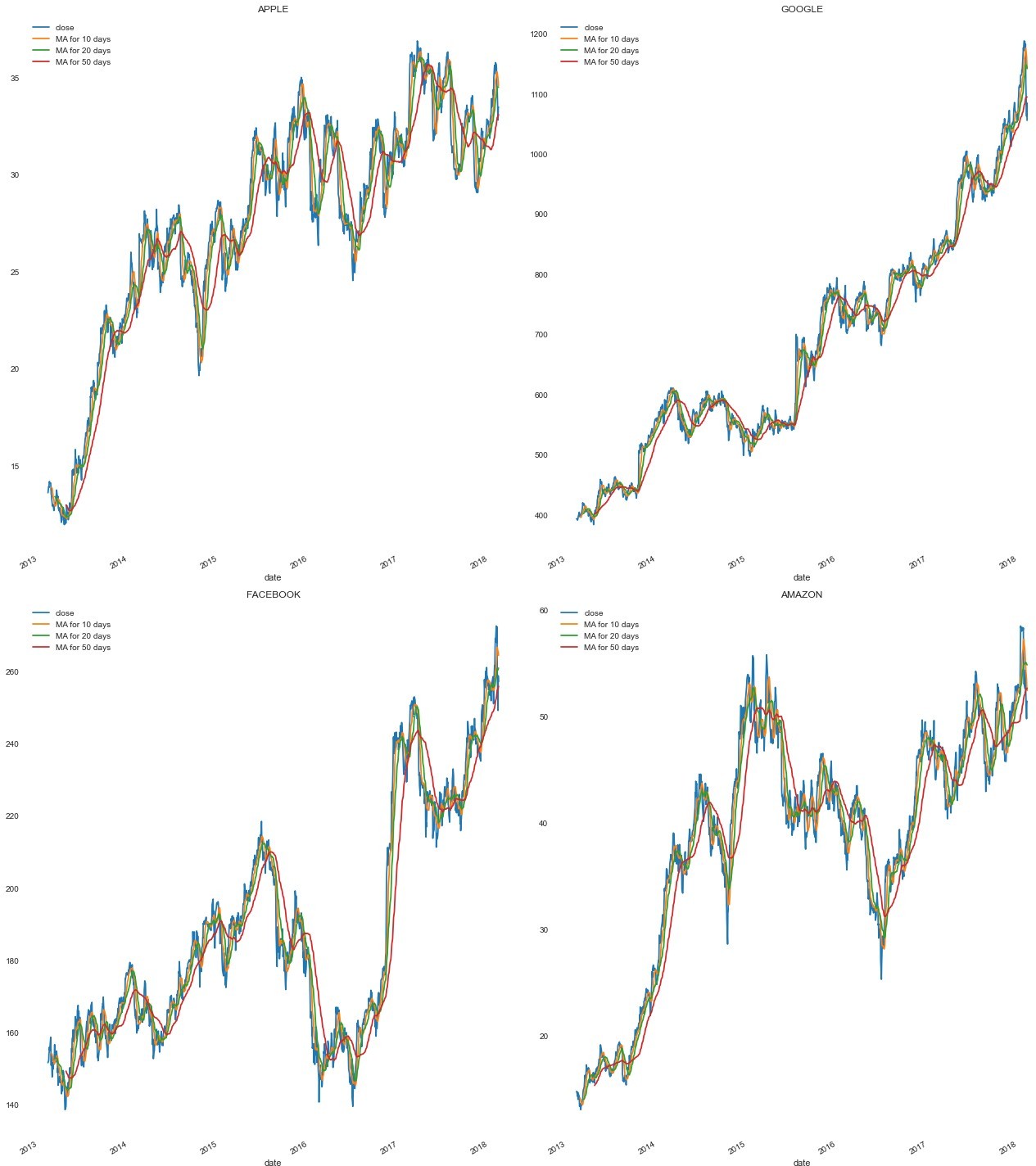
fig.set\_figwidth(18)

df\_apple[['close', 'MA for 10 days', 'MA for 20 days', 'MA for 50 days']].plot(ax**=**axe axes[0,0].set\_title('APPLE')

df\_google[['close', 'MA for 10 days', 'MA for 20 days', 'MA for 50 days']].plot(ax**=**ax axes[0,1].set\_title('GOOGLE')

df\_fb[['close', 'MA for 10 days', 'MA for 20 days', 'MA for 50 days']].plot(ax**=**axes[1 axes[1,0].set\_title('FACEBOOK')

df\_amazon[['close', 'MA for 10 days', 'MA for 20 days', 'MA for 50 days']].plot(ax**=**ax axes[1,1].set\_title('AMAZON')



In [14]:

*# We will analyze the risk of these companies stock*

**for** company **in** company\_list:

company['Daily Return'] **=** company['close'].pct\_change()

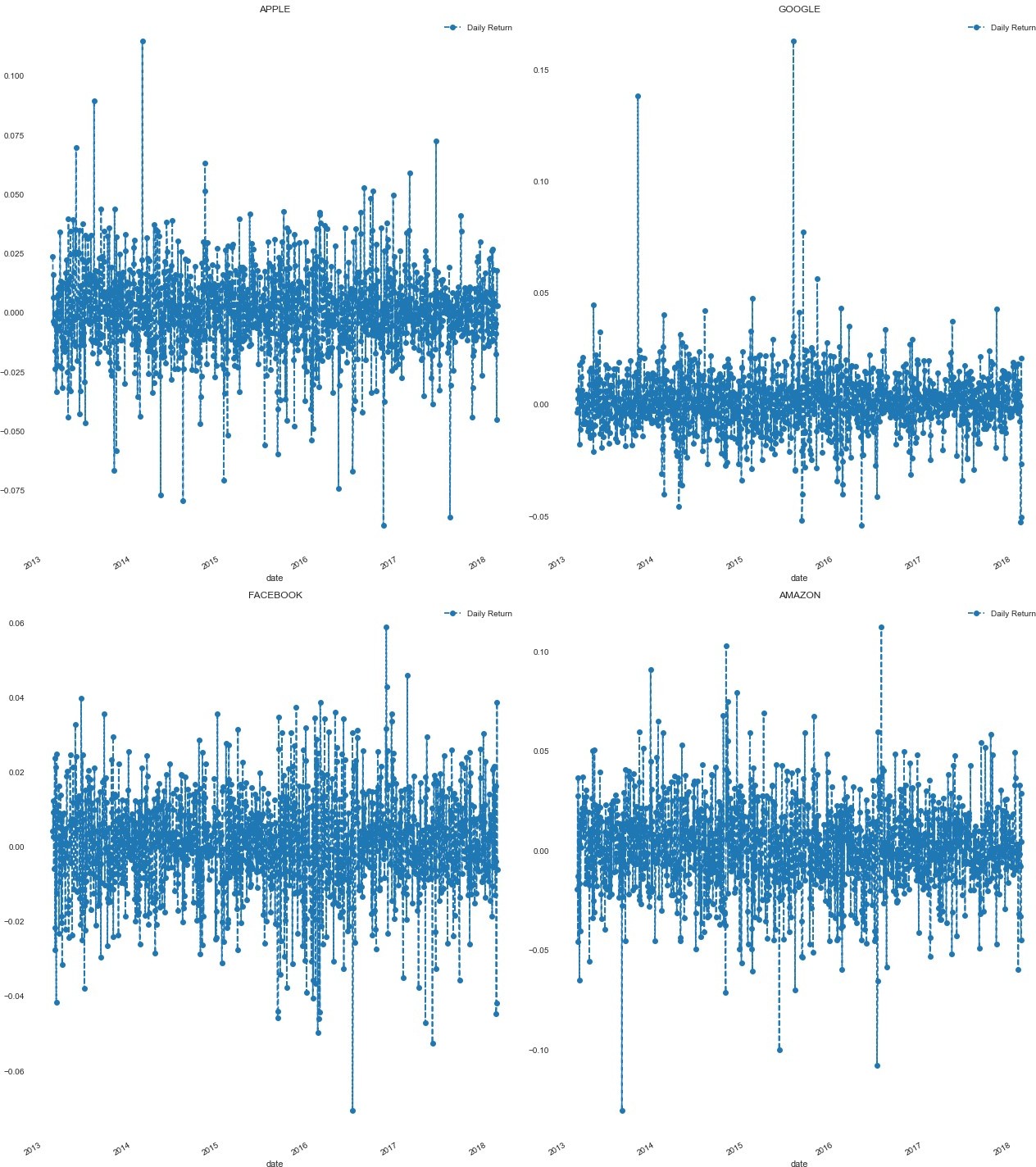
fig, axes **=** plt.subplots(nrows**=**2, ncols**=**2) fig.set\_figheight(20) fig.set\_figwidth(18)

df\_apple['Daily Return'].plot(ax**=**axes[0,0], legend**=True**, linestyle**=**'--', marker**=**'o') axes[0,0].set\_title('APPLE')

df\_google['Daily Return'].plot(ax**=**axes[0,1], legend**=True**, linestyle**=**'--', marker**=**'o') axes[0,1].set\_title('GOOGLE')

df\_fb['Daily Return'].plot(ax**=**axes[1,0], legend**=True**, linestyle**=**'--', marker**=**'o') axes[1,0].set\_title('FACEBOOK')

df\_amazon['Daily Return'].plot(ax**=**axes[1,1], legend**=True**, linestyle**=**'--', marker**=**'o') axes[1,1].set\_title('AMAZON')



In [15]:

**import** seaborn **as** sns plt.figure(figsize**=**(12, 12))

**for** i, company **in** enumerate(company\_list, 1): plt.subplot(2, 2, i)

sns.distplot(company['Daily Return'].dropna(), bins**=**100, color**=**'orange') plt.ylabel('Daily Return')

**if** i **==** 1:

plt.title(f"APPLE")

**elif** i **==** 2:

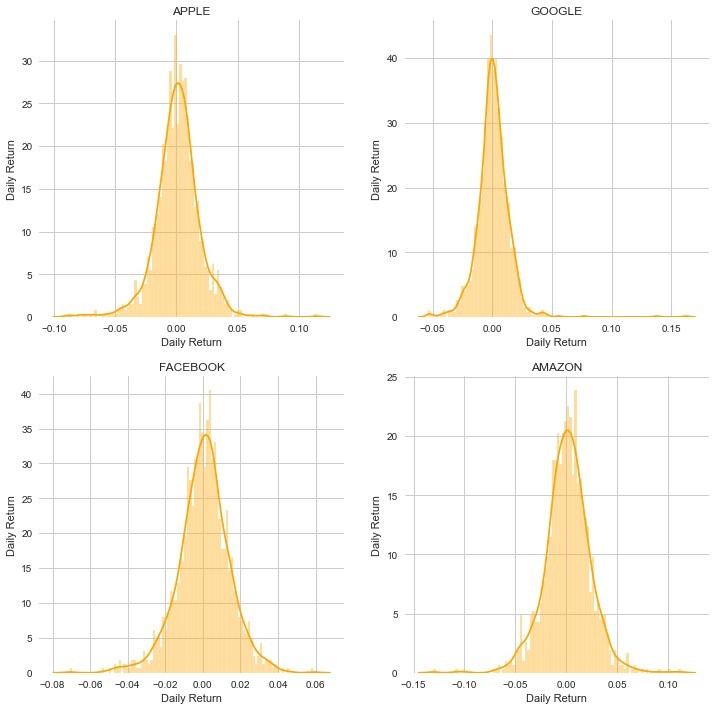
plt.title(f"GOOGLE")

**elif** i **==** 3:

plt.title(f"FACEBOOK")

**else**:

plt.title(f"AMAZON") plt.grid()



**FACEBOOK PROPHET**

In [16]:

*# Predicting Stock prices with LSTM*

data\_apple **=** df\_apple.filter(["close"]) data\_apple **=** data\_apple.values

data\_google **=** df\_google.filter(["close"]) data\_google **=** data\_google.values

data\_amazon **=** df\_amazon.filter(["close"]) data\_amazon **=** data\_amazon.values

data\_fb **=** df\_fb.filter(["close"]) data\_fb **=** data\_fb.values

train\_len **=** int(np.ceil(len(data\_apple) **\*** .8)) Out[16]: 1008

In [17]:

*#Predicting stock prices with Prophet # Apple Stock*

m\_apple **=** Prophet() m\_google **=** Prophet() m\_fb **=** Prophet() m\_amazon **=** Prophet()

*# Drop the columns*

ph\_df\_apple **=** df\_apple[["close", "date"]] ph\_df\_apple.rename(columns**=**{'close': 'y', 'date': 'ds'}, inplace**=True**) train\_df\_apple, y\_df\_apple **=** ph\_df\_apple[:1000], ph\_df\_apple[1000:]

ph\_df\_google **=** df\_google[["close", "date"]] ph\_df\_google.rename(columns**=**{'close': 'y', 'date': 'ds'}, inplace**=True**) train\_df\_google, y\_df\_google **=** ph\_df\_google[:1000], ph\_df\_google[1000:]

ph\_df\_amazon **=** df\_amazon[["close", "date"]] ph\_df\_amazon.rename(columns**=**{'close': 'y', 'date': 'ds'}, inplace**=True**) train\_df\_amazon, y\_df\_amazon **=** ph\_df\_amazon[:1000], ph\_df\_amazon[1000:]

ph\_df\_fb **=** df\_apple[["close", "date"]] ph\_df\_fb.rename(columns**=**{'close': 'y', 'date': 'ds'}, inplace**=True**) train\_df\_fb, y\_df\_fb **=** ph\_df\_fb[:1000], ph\_df\_fb[1000:]

Out[17]:

**date**

**y ds**

**2013-02-08** 13.59 2013-02-08

**2013-02-11** 13.91 2013-02-11

**y ds**

**date**

**2013-02-12** 13.86 2013-02-12

In [18]:

m\_apple.fit(train\_df\_apple) m\_google.fit(train\_df\_google) m\_fb.fit(train\_df\_fb)

INFO:fbprophet:Disabling daily seasonality. Run prophet with daily\_seasonality=Tru e to override this.

INFO:fbprophet:Disabling daily seasonality. Run prophet with daily\_seasonality=Tru e to override this.

INFO:fbprophet:Disabling daily seasonality. Run prophet with daily\_seasonality=Tru e to override this.

INFO:fbprophet:Disabling daily seasonality. Run prophet with daily\_seasonality=Tru e to override this.

Out[18]: <fbprophet.forecaster.Prophet at 0x1c42e8ecb48>

In [19]:

forcast\_apple **=** m\_apple.predict(y\_df\_apple[["ds"]]) forcast\_fb **=** m\_fb.predict(y\_df\_fb[["ds"]]) forcast\_google **=** m\_google.predict(y\_df\_google[["ds"]]) forcast\_amazon **=** m\_amazon.predict(y\_df\_amazon[["ds"]])

forcast\_apple.index **=** forcast\_apple['ds'] forcast\_google.index **=** forcast\_google['ds'] forcast\_fb.index **=** forcast\_fb['ds']

In [20]:

ph\_df\_y\_apple **=** ph\_df\_apple[["y"]] ph\_df\_y\_google **=** ph\_df\_google[["y"]] ph\_df\_y\_amazon **=** ph\_df\_amazon[["y"]]

In [21]:

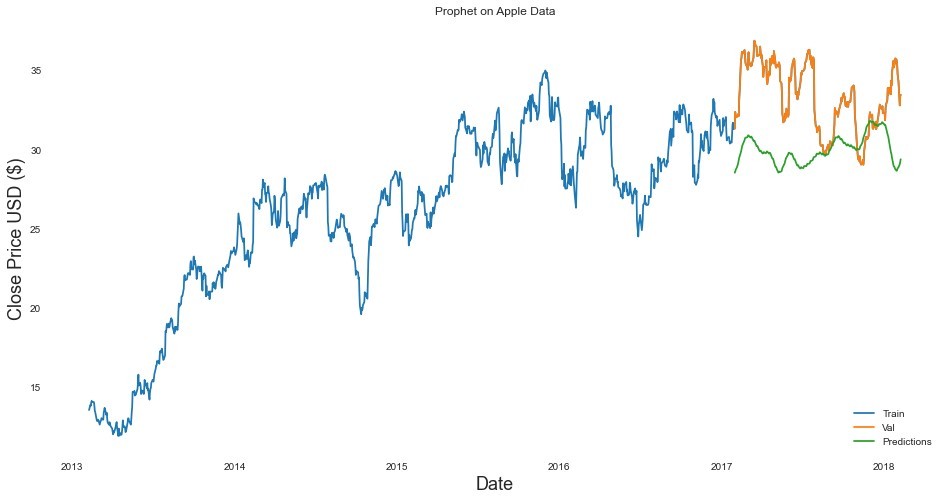
*# Plot the data*

train\_ph\_apple **=** ph\_df\_apple["y"].values valid\_ph\_apple **=** ph\_df\_apple['y'][1000:].values pred **=** forcast\_apple

*# Visualize the data* plt.figure(figsize**=**(16,8)) plt.title('Prophet on Apple Data') plt.xlabel('Date', fontsize**=**18)

plt.ylabel('Close Price USD ($)', fontsize**=**18) plt.plot(ph\_df\_y\_apple['y']) plt.plot(ph\_df\_y\_apple['y'][1000:]) plt.plot(pred["yhat"])

plt.legend(['Train', 'Val', 'Predictions'], loc**=**'lower right') plt.show()



In [22]:

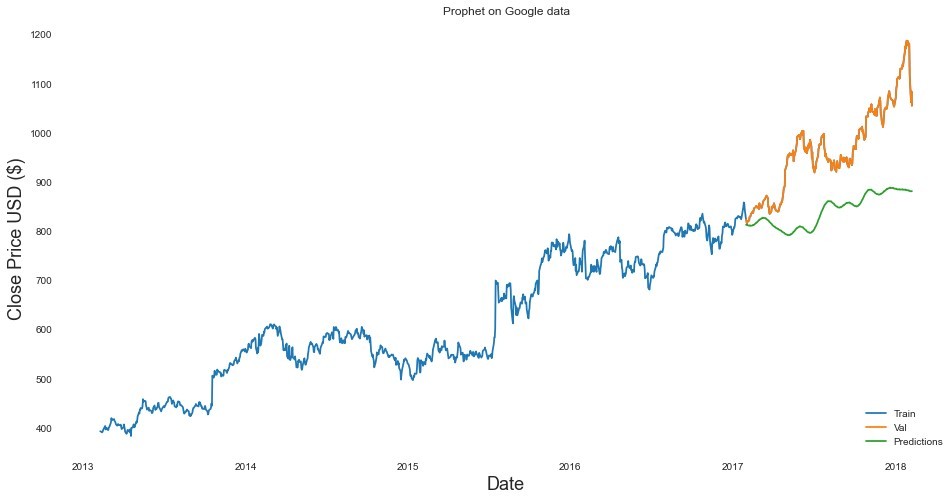
*# Plot the data*

train\_ph\_google **=** ph\_df\_google["y"].values valid\_ph\_google **=** ph\_df\_google['y'][1000:].values pred\_google **=** forcast\_google

*# Visualize the data* plt.figure(figsize**=**(16,8)) plt.title('Prophet on Google data') plt.xlabel('Date', fontsize**=**18)

plt.ylabel('Close Price USD ($)', fontsize**=**18) plt.plot(ph\_df\_y\_google['y']) plt.plot(ph\_df\_y\_google['y'][1000:]) plt.plot(pred\_google["yhat"])

plt.legend(['Train', 'Val', 'Predictions'], loc**=**'lower right')



In [23]:

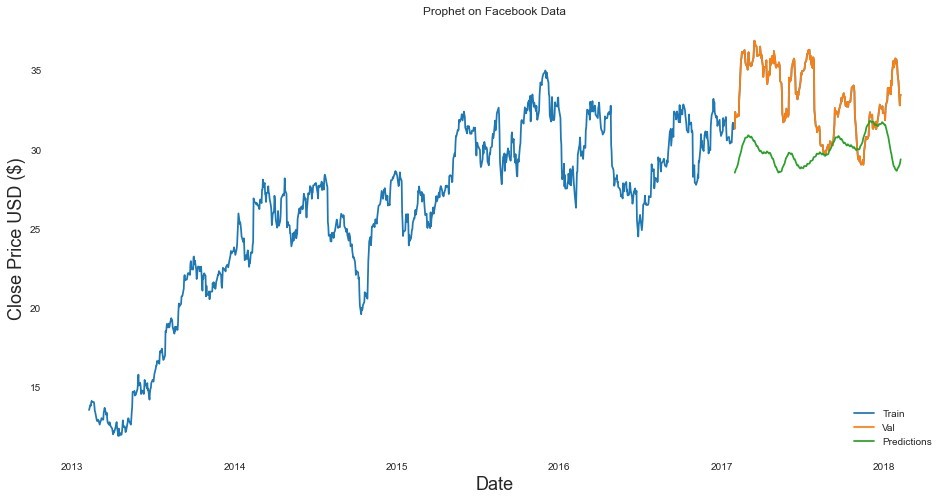
*# Plot the data*

train\_ph\_fb **=** ph\_df\_fb["y"].values valid\_ph\_fb **=** ph\_df\_fb['y'][1000:].values pred\_fb **=** forcast\_fb

*# Visualize the data* plt.figure(figsize**=**(16,8)) plt.title('Prophet on Facebook Data') plt.xlabel('Date', fontsize**=**18)

plt.ylabel('Close Price USD ($)', fontsize**=**18) plt.plot(ph\_df\_y\_fb['y']) plt.plot(ph\_df\_y\_fb['y'][1000:]) plt.plot(pred\_fb["yhat"])

plt.legend(['Train', 'Val', 'Predictions'], loc**=**'lower right')



In [24]:

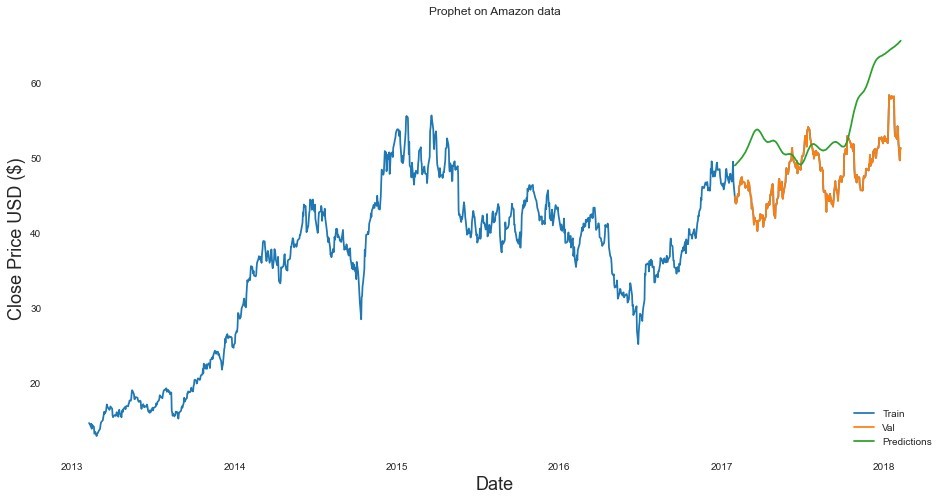
*# Plot the data*

train\_ph\_amazon **=** ph\_df\_amazon["y"].values valid\_ph\_amazon **=** ph\_df\_amazon['y'][1000:].values pred\_amazon **=** forcast\_amazon

*# Visualize the data* plt.figure(figsize**=**(16,8)) plt.title('Prophet on Amazon data') plt.xlabel('Date', fontsize**=**18)

plt.ylabel('Close Price USD ($)', fontsize**=**18) plt.plot(ph\_df\_y\_amazon['y']) plt.plot(ph\_df\_y\_amazon['y'][1000:]) plt.plot(pred\_amazon["yhat"])

plt.legend(['Train', 'Val', 'Predictions'], loc**=**'lower right')



**Reccurent Neural Network**

In [25]:

**from** sklearn.preprocessing **import** MinMaxScaler

train\_len\_apple **=** int(np.ceil(len(data\_apple) **\*** .8)) train\_len\_google **=** int(np.ceil(len(data\_google) **\*** .8)) train\_len\_fb **=** int(np.ceil(len(data\_fb) **\*** .8)) train\_len\_amazon **=** int(np.ceil(len(data\_amazon) **\*** .8))

scale **=** MinMaxScaler(feature\_range **=** (0, 1)) scale\_data\_apple **=** scale.fit\_transform(data\_apple) scale\_data\_fb **=** scale.fit\_transform(data\_fb) scale\_data\_amazon **=** scale.fit\_transform(data\_amazon) scale\_data\_google **=** scale.fit\_transform(data\_google)

train\_data\_apple **=** scale\_data\_apple[0:int(train\_len\_apple), :] train\_data\_google **=** scale\_data\_google[0:int(train\_len\_google), :] train\_data\_amazon **=** scale\_data\_amazon[0:int(train\_len\_amazon), :] train\_data\_fb **=** scale\_data\_fb[0:int(train\_len\_fb), :]

X\_train\_apple **=** [] y\_train\_apple **=** []

X\_train\_google **=** [] y\_train\_google **=** []

X\_train\_fb **=** [] y\_train\_fb **=** []

X\_train\_amazon **=** [] y\_train\_amazon **=** []

**for** i **in** range(60, len(train\_data\_apple)): X\_train\_apple.append(train\_data\_apple[i**-**60:i, 0])

y\_train\_apple.append(train\_data\_apple[i, 0])

**if** i**<=**61:

print(X\_train\_apple) print(y\_train\_apple) print()

**for** i **in** range(60, len(train\_data\_google)): X\_train\_google.append(train\_data\_google[i**-**60:i, 0])

y\_train\_google.append(train\_data\_google[i, 0])

**if** i**<=**61:

print(X\_train\_google) print(y\_train\_google) print()

**for** i **in** range(60, len(train\_data\_amazon)): X\_train\_amazon.append(train\_data\_amazon[i**-**60:i, 0])

y\_train\_amazon.append(train\_data\_amazon[i, 0])

**if** i**<=**61:

print(X\_train\_amazon) print(y\_train\_amazon) print()

**for** i **in** range(60, len(train\_data\_fb)): X\_train\_fb.append(train\_data\_fb[i**-**60:i, 0])

y\_train\_fb.append(train\_data\_fb[i, 0])

**if** i**<=**61:

print(X\_train\_fb) print(y\_train\_fb) print()

X\_train\_apple, y\_train\_apple **=** np.array(X\_train\_apple), np.array(y\_train\_apple); X\_train\_google, y\_train\_google **=** np.array(X\_train\_google), np.array(y\_train\_google); X\_train\_fb, y\_train\_fb **=** np.array(X\_train\_fb), np.array(y\_train\_fb);

[array([0.0657842 , 0.07862014, 0.07661452, 0.08543923, 0.08904934,

0.08624148, 0.08584035, 0.07661452, 0.06337746, 0.06016847,

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0.04211793, | 0.0381067 , | 0.0381067 | , | 0.04151625, | 0.04051344, |
| 0.02888087, | 0.03409547, | 0.0381067 | , | 0.03971119, | 0.04532692, |
| 0.04131568, | 0.04131568, | 0.0589651 | , | 0.0641797 , | 0.07139992, |
| 0.05856398, | 0.05294826, | 0.05876454, | | 0.04171681, | 0.03409547, |
| 0.02847974, | 0.03489771, | 0.0328921 , | | 0.02627357, | 0.01885279, |
| 0.01062976, | 0.00441235, | 0.01239471, | | 0.00842359, | 0.01925391, |
| 0.02406739, | 0.03012435, | 0.03529884, | | 0.02226233, | 0. , |
| 0.01885279, | 0.00280786, | 0.00120337, | | 0.00762134, | 0.00320899, |
| 0.0104292 , | 0.0224629 , | 0.03971119, | | 0.0224629 , | 0.02486963, |
| 0.02186121, | 0.01002808, | 0.01123145, | | 0.01674689, | 0.03610108])] |

[0.04492579221821108]

[array([0.0657842 , 0.07862014, 0.07661452, 0.08543923, 0.08904934,

|  |  |  |  |
| --- | --- | --- | --- |
| 0.08624148, 0.08584035, | 0.07661452, | 0.06337746, | 0.06016847, |
| 0.04211793, 0.0381067 , | 0.0381067 , | 0.04151625, | 0.04051344, |

0.02888087, 0.03409547, 0.0381067 , 0.03971119, 0.04532692,

0.04131568, 0.04131568, 0.0589651 , 0.0641797 , 0.07139992,

In [26]:

X\_train\_apple **=** np.reshape(X\_train\_apple, (X\_train\_apple.shape[0], X\_train\_apple.shap X\_train\_google **=** np.reshape(X\_train\_google, (X\_train\_google.shape[0], X\_train\_google. X\_train\_fb **=** np.reshape(X\_train\_fb, (X\_train\_fb.shape[0], X\_train\_fb.shape[1], 1))

In [27]:

Out[27]: (948, 60, 1)

In [28]:

**from** tensorflow.compat.v1 **import** ConfigProto

**from** tensorflow.compat.v1 **import** InteractiveSession config **=** ConfigProto() config.gpu\_options.allow\_growth **= True**

session **=** InteractiveSession(config **=** config)

**from** tensorflow.keras.models **import** Sequential

**from** tensorflow.keras.layers **import** Dense, LSTM

*#Build the LSTM model*

model\_apple **=** Sequential()

model\_apple.add(LSTM(50, return\_sequences**=True**, input\_shape**=** (X\_train\_apple.shape[1], model\_apple.add(LSTM(50, return\_sequences**= False**))

model\_apple.add(Dense(25)) model\_apple.add(Dense(1))

model\_apple.compile(optimizer **=** "adam", loss **=** "mean\_squared\_error")

Epoch 1/10

948/948 [==============================] - 11s 12ms/step - loss: 0.0056

Epoch 2/10

948/948 [==============================] - 11s 12ms/step - loss: 0.0017

Epoch 3/10

948/948 [==============================] - 12s 12ms/step - loss: 0.0012

Epoch 4/10

948/948 [==============================] - 12s 12ms/step - loss: 9.7367e-04

Epoch 5/10

948/948 [==============================] - 12s 12ms/step - loss: 8.7230e-04

Epoch 6/10

948/948 [==============================] - 11s 12ms/step - loss: 7.9078e-04

Epoch 7/10

948/948 [==============================] - 12s 12ms/step - loss: 6.7495e-04

Epoch 8/10

948/948 [==============================] - 12s 12ms/step - loss: 6.9502e-04

Epoch 9/10

948/948 [==============================] - 12s 12ms/step - loss: 7.1985e-04

Epoch 10/10

948/948 [==============================] - 12s 13ms/step - loss: 7.5408e-04

Out[28]: <tensorflow.python.keras.callbacks.History at 0x1c43ab9d108>

In [29]:

model\_google **=** Sequential()

model\_google.add(LSTM(50, return\_sequences**=True**, input\_shape**=** (X\_train\_google.shape[1 model\_google.add(LSTM(50, return\_sequences**= False**))

model\_google.add(Dense(25)) model\_google.add(Dense(1))

model\_google.compile(optimizer **=** "adam", loss **=** "mean\_squared\_error") model\_google.fit(X\_train\_google, y\_train\_google, batch\_size **=** 1, epochs **=** 20)

948/948 [==============================] - 12s 12ms/step - loss: 0.0017

Epoch 2/20

948/948 [==============================] - ETA: 0s - loss: 6.7422e-0 - 12s 12ms/st

ep - loss: 6.7388e-04 Epoch 3/20

948/948 [==============================] - 12s 12ms/step - loss: 4.5235e-04

Epoch 4/20

948/948 [==============================] - 11s 12ms/step - loss: 3.3580e-04

Epoch 5/20

948/948 [==============================] - 12s 12ms/step - loss: 2.9459e-04

Epoch 6/20

948/948 [==============================] - 12s 12ms/step - loss: 2.9375e-04

Epoch 7/20

948/948 [==============================] - 12s 12ms/step - loss: 2.5083e-04

Epoch 8/20

948/948 [==============================] - 11s 12ms/step - loss: 2.5456e-04

Epoch 9/20

948/948 [==============================] - 11s 12ms/step - loss: 2.6441e-04

Epoch 10/20

948/948 [==============================] - 11s 12ms/step - loss: 2.5091e-04

Epoch 11/20

948/948 [==============================] - 11s 12ms/step - loss: 2.3368e-04

Epoch 12/20

948/948 [==============================] - 12s 12ms/step - loss: 2.2476e-04

Epoch 13/20

948/948 [==============================] - 12s 12ms/step - loss: 2.3619e-04

Epoch 14/20

948/948 [==============================] - 12s 13ms/step - loss: 2.0734e-04

Epoch 15/20

948/948 [==============================] - 12s 12ms/step - loss: 2.3922e-04

Epoch 16/20

948/948 [==============================] - 11s 12ms/step - loss: 2.1930e-04

Epoch 17/20

948/948 [==============================] - 12s 12ms/step - loss: 2.2950e-04

Epoch 18/20

948/948 [==============================] - 11s 12ms/step - loss: 2.1901e-04

Epoch 19/20

948/948 [==============================] - 12s 13ms/step - loss: 1.9304e-04

Epoch 20/20

Out[29]: <tensorflow.python.keras.callbacks.History at 0x1c43d6ef5c8>

In [30]:

model\_fb **=** Sequential()

model\_fb.add(LSTM(50, return\_sequences**=True**, input\_shape**=** (X\_train\_fb.shape[1], 1))) model\_fb.add(LSTM(50, return\_sequences**= False**))

model\_fb.add(Dense(25)) model\_fb.add(Dense(1))

model\_fb.compile(optimizer **=** "adam", loss **=** "mean\_squared\_error")

In [76]:

Epoch 1/10

948/948 [==============================] - 12s 12ms/step - loss: 0.0031

Epoch 2/10

948/948 [==============================] - 12s 12ms/step - loss: 0.0013

Epoch 3/10

948/948 [==============================] - 12s 12ms/step - loss: 9.3762e-04

Epoch 4/10

948/948 [==============================] - 12s 13ms/step - loss: 7.3032e-0

**from** tensorflow.keras.layers **import** Activation, Dropout model\_amazon **=** Sequential()

model\_amazon.add(LSTM(50, return\_sequences**=True**, input\_shape**=** (X\_train\_amazon.shape[1 model\_amazon.add(Dropout(0.2))

model\_amazon.add(LSTM(128, return\_sequences**= False**)) model\_amazon.add(Dropout(0.2)) *#model\_amazon.add(Dense(25))* model\_amazon.add(Dense(1)) model\_amazon.add(Activation("linear"))

model\_amazon.compile(optimizer **=** "adam", loss **=** "mean\_squared\_error")

Epoch 1/10

|  |  |  |
| --- | --- | --- |
|  | 948/948 [==============================] - 12s 12ms/step - loss:  Epoch 2/10  948/948 [==============================] - 12s 12ms/step - loss: | 0.0066  0.0041 |
| Epoch 3/10 |  |
| 948/948 [==============================] - 11s 11ms/step - loss:  Epoch 4/10  948/948 [==============================] - 12s 12ms/step - loss: | 0.0030  0.0025 |
| Epoch 5/10  948/948 [==============================] - 11s 12ms/step - loss: | 0.0021 |
| Epoch 6/10  948/948 [==============================] - 11s 12ms/step - loss: | 0.0022 |
| Epoch 7/10  948/948 [==============================] - 12s 13ms/step - loss: | 0.0018 |
| Epoch 8/10  948/948 [==============================] - 11s 12ms/step - loss: | 0.0017 |
| Epoch 9/10  948/948 [==============================] - 11s 12ms/step - loss: | 0.0019 |
| Epoch 10/10 |  |
| 948/948 [==============================] - 12s 13ms/step - loss: | 0.0016 |
| Out[76]: | <tensorflow.python.keras.callbacks.History at 0x1c48d3bc648> |  |
| In [33]: |  |  |

*#Create the testing data set for apple*

*#Create a new array containing scaled values from index 1543 to 2002* test\_data\_apple **=** scale\_data\_apple[train\_len\_apple **-** 60:, :] *#Create the data sets x\_test and y\_test*

x\_test\_apple **=** []

y\_test\_apple **=** data\_apple[train\_len\_apple:, :]

**for** i **in** range(60, len(test\_data\_apple)): x\_test\_apple.append(test\_data\_apple[i**-**60:i, 0])

*# Convert the data to a numpy array*

x\_test\_apple **=** np.array(x\_test\_apple)

*# Reshape the data*

x\_test\_apple **=** np.reshape(x\_test\_apple, (x\_test\_apple.shape[0], x\_test\_apple.shape[1]

*#Create the testing data set for google*

*#Create a new array containing scaled values from index 1543 to 2002* test\_data\_google **=** scale\_data\_google[train\_len\_google **-** 60:, :] *#Create the data sets x\_test and y\_test*

x\_test\_google **=** []

y\_test\_google **=** data\_google[train\_len\_google:, :]

**for** i **in** range(60, len(test\_data\_google)): x\_test\_google.append(test\_data\_google[i**-**60:i, 0])

*# Convert the data to a numpy array*

x\_test\_google **=** np.array(x\_test\_google)

*# Reshape the data*

x\_test\_google **=** np.reshape(x\_test\_google, (x\_test\_google.shape[0], x\_test\_google.shap

*#Create the testing data set for google*

*#Create a new array containing scaled values from index 1543 to 2002*

test\_data\_fb **=** scale\_data\_fb[train\_len\_fb **-** 60:, :]

*#Create the data sets x\_test and y\_test*

x\_test\_fb **=** []

y\_test\_fb **=** data\_fb[train\_len\_fb:, :]

**for** i **in** range(60, len(test\_data\_fb)): x\_test\_fb.append(test\_data\_fb[i**-**60:i, 0])

*# Convert the data to a numpy array*

x\_test\_fb **=** np.array(x\_test\_fb)

*# Reshape the data*

x\_test\_fb **=** np.reshape(x\_test\_fb, (x\_test\_fb.shape[0], x\_test\_fb.shape[1], 1 ))

*#Create the testing data set for google*

*#Create a new array containing scaled values from index 1543 to 2002* test\_data\_amazon **=** scale\_data\_amazon[train\_len\_amazon **-** 60:, :] *#Create the data sets x\_test and y\_test*

x\_test\_amazon **=** []

y\_test\_amazon **=** data\_amazon[train\_len\_amazon:, :]

**for** i **in** range(60, len(test\_data\_amazon)): x\_test\_amazon.append(test\_data\_amazon[i**-**60:i, 0])

*# Convert the data to a numpy array*

x\_test\_amazon **=** np.array(x\_test\_amazon)

*# Reshape the data*

*# Get the models predicted price values apple* predictions\_apple **=** model\_apple.predict(x\_test\_apple) predictions\_apple **=** scale.inverse\_transform(predictions\_apple)

*# Get the root mean squared error (RMSE)*

rmse\_apple **=** np.sqrt(np.mean(((predictions\_apple **-** y\_test\_apple) **\*\*** 2))) print(rmse\_apple)

*# Get the models predicted price values google* predictions\_google **=** model\_google.predict(x\_test\_google) predictions\_google **=** scale.inverse\_transform(predictions\_google)

*# Get the root mean squared error (RMSE)*

rmse\_google **=** np.sqrt(np.mean(((predictions\_google **-** y\_test\_google) **\*\*** 2))) print(rmse\_google)

*# Get the models predicted price values fb*

predictions\_fb **=** model\_fb.predict(x\_test\_fb)

In [34]:

predictions\_fb **=** scale.inverse\_transform(predictions\_fb)

*# Get the root mean squared error (RMSE)*

rmse\_fb **=** np.sqrt(np.mean(((predictions\_fb **-** y\_test\_fb) **\*\*** 2))) print(rmse\_fb)

*# Get the models predicted price values amazon* predictions\_amazon **=** model\_amazon.predict(x\_test\_amazon) predictions\_amazon **=** scale.inverse\_transform(predictions\_amazon)

*# Get the root mean squared error (RMSE)*

rmse\_amazon **=** np.sqrt(np.mean(((predictions\_amazon **-** y\_test\_amazon) **\*\*** 2))) 1029.681798507289

19.465939596133772

717.0698019969451

Out[34]: 938.459676144034

In [45]:

Out[45]:

**date open high low close volume Name**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **date** |  |  |  |  | **days days** | **days** |  |
| **2017-11-14** 2017-11-14 | 29.15 | 29.50 29.02 | 29.36 2270155 | GT | 29.455 31.2500 | 32.0236 | 0.004447 839.862 |
| **2017-10-06** 2017-10-06 | 33.21 | 33.26 32.72 | 32.77 2575004 | GT | 33.170 32.7310 | 31.4290 | -0.011761 966.359 |
| **2017-10-25** 2017-10-25 | 33.98 | 33.99 33.53 | 33.95 2170549 | GT | 33.464 33.2925 | 32.0042 | -0.003230 995.742 |
| **2017-10-04** 2017-10-04 | 33.27 | 33.59 33.20 | 33.36 1475167 | GT | 33.080 32.4645 | 31.5378 | 0.001501 970.958 |
| **2017-09-14** 2017-09-14 | 32.23 | 32.53 32.08 | 32.25 2495674 | GT | 31.102 30.5330 | 32.3502 | -0.001548 938.538 |
| **2017-05-25** 2017-05-25 | 32.43 | 32.59 31.91 | 32.02 2849375 | GT | 32.636 34.0400 | 34.9658 | -0.008976 941.057 |
| **2017-06-13** 2017-06-13 | 35.72 | 35.72 35.30 | 35.67 3176140 | GT | 34.726 33.4895 | 34.5200 | -0.002238 1050.246 |
| **2017-04-17** 2017-04-17 | 34.32 | 34.94 34.09 | 34.94 2256715 | GT | 34.950 35.5145 | 35.3192 | 0.023433 996.350 |
| **2017-12-20** 2017-12-20 | 32.33 | 32.52 32.21 | 32.42 1900878 | GT | 31.715 31.6745 | 31.4610 | 0.007771 935.949 |
| **2018-02-05** 2018-02-05 | 34.17 | 34.66 32.72 | 32.80 3615780 | GT | 34.938 34.4965 | 32.9738 | -0.045124 1004.749 |

**MA**

**for 10**

**MA for**

**20**

**MA for**

**50**

**Daily Predicti Return**

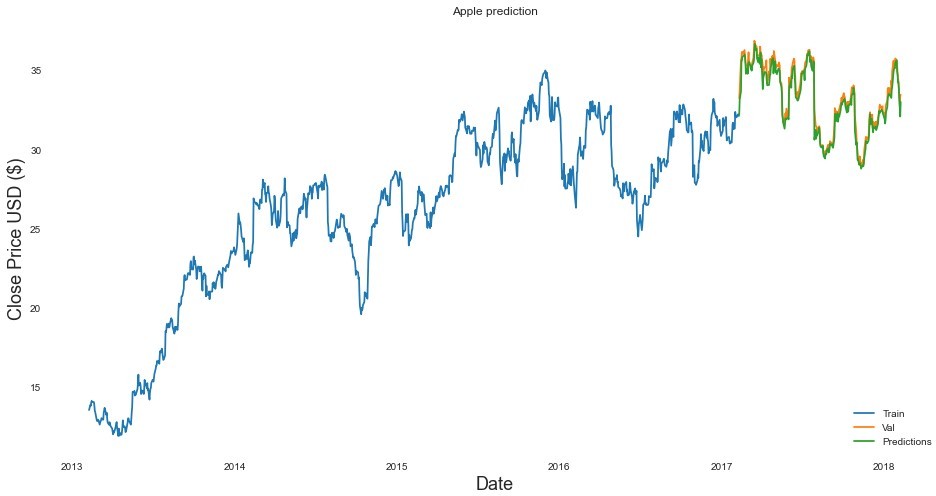
In [65]:

*# Plot the data*

train\_apple **=** df\_apple[:train\_len\_apple] valid\_apple **=** df\_apple[train\_len\_apple:] valid\_apple['Predictions'] **=** predictions\_apple *# Visualize the data* plt.figure(figsize**=**(16,8))

plt.title('Apple prediction') plt.xlabel('Date', fontsize**=**18) plt.ylabel('Close Price USD ($)', fontsize**=**18) plt.plot(train\_apple['close']) plt.plot(valid\_apple[['close', 'Predictions']])

plt.legend(['Train', 'Val', 'Predictions'], loc**=**'lower right')

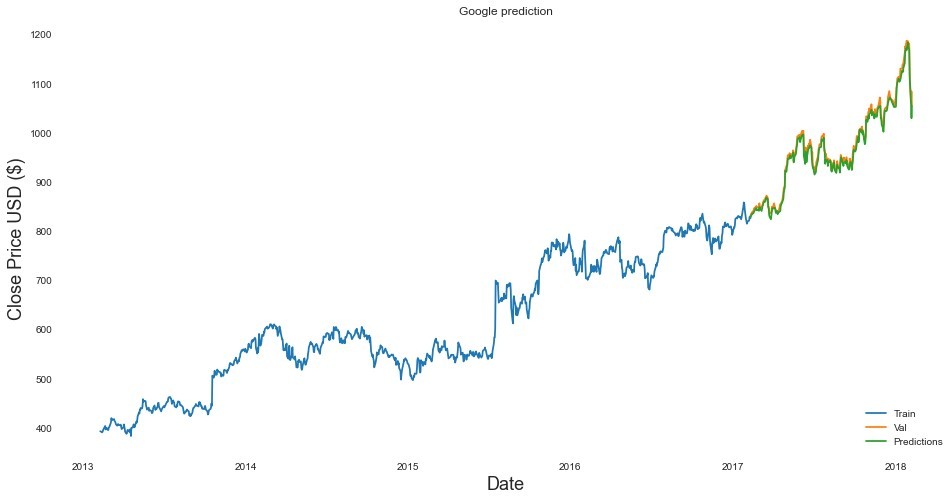


In [71]:

train\_google **=** df\_google[:train\_len\_google] valid\_google **=** df\_google[train\_len\_google:] valid\_google['Predictions'] **=** predictions\_google *# Visualize the data*

plt.figure(figsize**=**(16,8)) plt.title('Google prediction') plt.xlabel('Date', fontsize**=**18)

plt.ylabel('Close Price USD ($)', fontsize**=**18) plt.plot(train\_google['close']) plt.plot(valid\_google[['close', 'Predictions']]) plt.legend(['Train', 'Val', 'Predictions'], loc**=**'lower right')



In [73]:

train\_fb **=** df\_fb[:train\_len] valid\_fb **=** df\_fb[train\_len:]

valid\_fb['Predictions'] **=** predictions\_fb *# Visualize the data* plt.figure(figsize**=**(16,8)) plt.title('FaceBook prediction') plt.xlabel('Date', fontsize**=**18)

plt.ylabel('Close Price USD ($)', fontsize**=**18) plt.plot(train\_fb['close']) plt.plot(valid\_fb[['close', 'Predictions']])

plt.legend(['Train', 'Val', 'Predictions'], loc**=**'lower right')



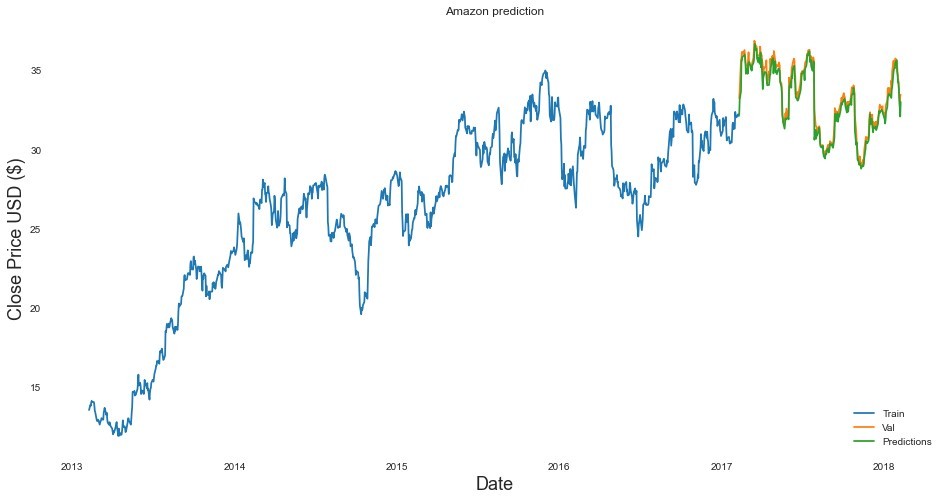
In [74]:

train\_amazon **=** df\_apple[:train\_len] valid\_amazon **=** df\_apple[train\_len:]

valid\_amazon['Predictions'] **=** predictions\_amazon

*# Visualize the data* plt.figure(figsize**=**(16,8)) plt.title('Amazon prediction') plt.xlabel('Date', fontsize**=**18)

plt.ylabel('Close Price USD ($)', fontsize**=**18) plt.plot(train\_amazon['close']) plt.plot(valid\_amazon[['close', 'Predictions']]) plt.legend(['Train', 'Val', 'Predictions'], loc**=**'lower right')



In [ ]:

In [ ]: