Morahabadi, Ranchi-834001

A Project

On

**Stock Market Price Prediction**

MASTER OF COMPUTER APPLICATION

(2022)



***Teacher’s Signature:***

***Submitted by:***

**Name:**

**Sawan**

**Kumar**

**Roll.no:**

**48**

**Exam Roll:**

**19MC513672**

**Session:**

**2019-2022**

**Department:**

**MCA sem-6**

**Table of Contents**

[**ACKNOWLEDGEMENT** 3](#_Toc107598501)

[**Professional Exposure** 4](#_Toc107598502)

[Technical Skills 5](#_Toc107598503)

[Professional Experience: 5](#_Toc107598504)

[Declaration 7](#_Toc107598505)

[BONAFIDECERTIFICATE 8](#_Toc107598506)

[EXAMINER’S CERTIFICATION 9](#_Toc107598507)

[Chapter 1 Introduction 11](#_Toc107598508)

[**Purpose** 11](#_Toc107598509)

[**Scope** 11](#_Toc107598510)

[**Stock analysis: fundamental analysis vs. technical analysis** 11](#_Toc107598511)

[**Problem Statement** 12](#_Toc107598512)

[Chapter 2 Literature Survey 13](#_Toc107598513)

[2.1 Introduction to Literature Review 13](#_Toc107598514)

[Chapter 3 Methodology 21](#_Toc107598515)

[3.1 PROPOSED SYSTEMS 21](#_Toc107598516)

[3.1.1 Time Series Analysis 21](#_Toc107598517)

[3.1.2 Long short-term memory network: 27](#_Toc107598518)

[3.2 SYSTEM ARCHITECTURE 31](#_Toc107598519)

[Chapter 4 Design 32](#_Toc107598520)

[4.1 Structure Chart 32](#_Toc107598521)

[4.2 UML Diagrams 33](#_Toc107598522)

[4.2.1 Use Case Diagram 34](#_Toc107598523)

[4.2.2 Sequence Diagram 35](#_Toc107598524)

[4.2.3 Activity Diagram 36](#_Toc107598525)

[4.2.4 Collaboration Diagram 37](#_Toc107598526)

[4.2.5 Flow Chart 38](#_Toc107598527)

[4.2.6 Component Diagram 39](#_Toc107598528)

[Chapter 5 Experiment Analysis 40](#_Toc107598529)

[5.1 System configuration 40](#_Toc107598530)

[5.2 Sample code 41](#_Toc107598531)

[Chapter 6 Conclusion And Future Work 91](#_Toc107598532)

[6.1 Conclusion 91](#_Toc107598533)

[6.2 Future work 91](#_Toc107598534)

[Chapter 7 Bibliography 92](#_Toc107598535)

# 

# ACKNOWLEDGEMENT

It is a great pleasure for us to acknowledge all those who have contributed towards the conception, origin and nurturing of this project that is on System analysis the “***Stock Market Price Prediction”.***

The way can’t walk itself. We have to walk on it. For that we must have a guide. Many guides have contributed to the successful completion of the project. We would like to place on record my grateful thanks to each one of them who help us in this project.

Before we get into thick of the thing, we would like to add a few heartfelt words for the people who gave us unending time support whichever and whenever necessary.

Our grateful thanks go to our Dept., which provides us an opportunity as a project subject in **6th Semester** to develop a report work skill in this System analyzing.

We would like to thank our parents & friends for giving us full feedback when we are in trouble. Our special thanks go to **Mr. Indra Nath Sahu** to give their expert guidance to us whenever necessary.

Last but not the least; I heartily thank our H.O.D. **Mrs. Anita Kumari.**

Date:

SAWAN KUMAR

19MC513672

**SAURAV SINHA**

|  |
| --- |
| **Profile** |

7 years of experience in System server engineer desktop system and network. Support engineer looking for the opportunity to utilize my skills and abilities in new organization that offers me professional growth while being resourceful. Innovative and flexible in order to develop myself and organization concerned. Strong in learning and adapting new technology new environment.

|  |
| --- |
| **Professional Exposure** |

Total 3 years of experience in IT industry 9 month working on supporting engineer.1 year 9 month working in Server engineer. Server related issues and manage all server such as, DNS DHCP, SCCM, AD, FSMO ROLLES, Exchange server Install Configure monitoring server with solar winds.

Fast learner, self-motivated, applying methods and technical updates in my day-to-day work with a high degree of attention and commitment to work.

**Summary**

* Expertise in working and Supporting client OS Windows XP, Vista and Windows 7,8.
* Attending Server Hardware, Software calls in a 24x7 availability environment at site & resolving the Server issue On SLA Based.
* Working on Remedy Tool
* Provide permissions to the OU for gaining access to use network resources.
* Installation of Active Directory Services, joining systems on domain.
* Installation & configure exchange server 2007,
* Manage a mail box size and modify.
* Managing storage group.
* Installation Migrate DHCP, DNS server and create fail overs cluster,
* Installation & configuration of printer server in Network.
* Installation manage DNS & DHCP server.
* Create new scope modify and DNS server.
* Create a new zone in DNS server and modify.
* Create GPO policy.
* Create a backup server in schedule.
* Installation, Configuration & Troubleshooting of Microsoft office, Outlook 2003, 2007, 2010.
* Installation & configuration of Symantec anti-virus server, McAfee anti-virus & downloading updates.

|  |
| --- |
| Technical Skills |

* Operating System : WIN-2003, 2008, 2012 Server, WIN-XP/VISTA, 7, 8,10.
* Microsoft Technology : Active Directory Services, DNS, DHCP, VPN, RRAS.
* Area of Knowledge : Knowledge of Computer Networking & server.
* Configuration : Server AD, DHCP, DNS, RRAS, EXCHANGE-SERVER, SCCM, All server
* Mail Application : MS Outlook and Windows Mail

|  |
| --- |
| Professional Experience: |

* **SHANTI BUILDERS AND DEVELOPERS PVT. LTD. AS A Server Engineer**
* 12 August 2014 TO TILL DATE
* Assisted company personnel with server operation both hardware and software related issue.
* Installation Configuration of Windows 2008 r2/2012 File folder and Print Server
* Create group policy and giving the permission to access AD object.
* Create DNS Zone and manage Zone.
* Working with SCCM 2007.
* Installation Configuration Terminal Services and Remote Desktops.
* Installation of Active Directory Services, systems on domain.
* Installation & configure exchange server 2007,2010.
* Manage a mail box size and create a managing public folder database.
* Managing storage group and policy exchange server 2007&2010.
* Monitoring & reporting to all exchange server massage and troubleshooting with ETA
* Backup and restore windows server and create a schedule in daily based.
* Installation Migrate DHCP, DNS server and create fail overs cluster.
* Create a backup server in schedule.
* Managing Data Backups and Restores AD.
* Monitoring server & networking in tools. Like SOLAR-WINDS.
* Disaster Recovery and Business Continuity Processes.
* Provide Support for Network Configuration.
* Cable patching in data center
* Assisted company personnel with server operation both hardware and software related issue.
* Anti-Virus installation Norton Antivirus (Professional & Corporate Edition) Mac fee, AVG, Avast etc. Virus Cleaning, Service Pack & Patch Installation in windows server.
* Monitoring server & networking in tools. Like SOLAR-WINDS
* Installation of Active Directory Services, joining systems on domain.
* Installation & configure exchange server 2007.
* Manage a mail box size and create a DL.
* Managing storage group and policy in exchange server 2007&2010.
* Backup and recovery windows and create a schedule in daily based.
* Configuration and Troubleshooting of Network Printers
* Solving Issues with Routers, Desktop, Laptop and other Devices.
* Taking Remote Access and creating Network and Physical configuration and Troubleshoot system issues.
* Create a backup server in schedule.
* Installation, Configuration & Troubleshooting of Microsoft office, Outlook 2003, 2007, 2010.
* Installation & configuration of Symantec anti-virus server, McAfee anti-virus & downloading updates.
* Provide permissions to the OU for gaining access to use network resources.
* Installation of networking equipment’s like Switch, Hub, and LAN card, Modem card.

|  |
| --- |
| **Educational Qualification:** |

|  |  |  |  |
| --- | --- | --- | --- |
| CLASS | PERCENTILE | YEAR | BOARD/University |
| X th | 50 | 2005 | I.C.S.E |
| XIIth | 66 | 2008 | N.I.O.S |
| Bachelor of Engineering | 67 | 2009-2013 | R.G.P.V. (MADHYA PRADESH) |
| M.Tech | 76 | 2017 | KUT |

**I hereby declare that the above-mentioned information is true to the best of my knowledge.**

**References are available upon request.**

**(SAURAV SINHA)**

# Declaration

I, **Sawan Kumar** hereby declare that the project report “**Stock Market Price Prediction** “is based on my own work carried out during the course of our study under the supervision of Pooja Jha Mam**, Department of MCA, DSPMU RANCHI**.

I assert the statements made and conclusions drawn are an outcome of my research work. I further certify that

1. The work contained in the report is original and has been done by me under the general supervision of my supervisor.
2. The work has not been submitted to any other Institution for any other degree/diploma/certificate in this university or any other University of India or abroad.
3. We have followed the guidelines provided by the university in writing the report.
4. Whenever we have used materials (data, theoretical analysis, and text) from other sources, we have given due credit to them in the text of the report and giving their details in the references.

# BONAFIDECERTIFICATE

This is to certify that this project titled **Stock Market Price Prediction** is the bonafide work of **Sawan Kumar** 6th semester Department of MCA, DSPMU RANCHI that carried out the project work under my supervision.

Signature

SAURAV SINHA(GUIDE)

# EXAMINER’S CERTIFICATION

The project report entitled “**Stock Market Prediction”** submitted by **Sawan Kumar (19MC513672)** in partial fulfilment of requirement of the award of the **Degree Master of Computer Application in P.G., Department of MCA, Dr. Shyama Prasad Mukherjee University RANCHI**.Is approved and is acceptable in quality and form.

Is approved and is acceptable in quality and form.

Signature

Name:

(External Examiner-2)

Signature

Name:

(External Examiner-1)

**LIST OF ABBREVATIONS**

|  |  |
| --- | --- |
| LSTM | Long Short-Term Memory |
| ATS | Automated Trading System |
| GRU | Gated Recurrent Unit |
| ML | Machine Learning |
| SVM | Support Vector Machine |
| EMH | Efficient Market hypothesis |
| AI | Artificial Intelligence |
| NN | Neural Networks |
| ARMA | Autoregressive Moving Average |
| DRL | Deep Reinforcement Learning |
| LMS | Least Mean Square |
| UML | Unified modelling Language |
| MSE | Mean Squared Error |
| RMSE | Root Mean Squared Error |

# Chapter 1 Introduction

## **Purpose**

Stock Market Price is known for being volatile, dynamic and nonlinear. Accurate stock price prediction is extremely challenging because of multiple (macro and micro) factors, such as politics, global economic conditions, unexpected events, a company’s financial performance, and so on.

But, all of this also means that there’s a lot of data to find patterns in. So, financial analysts, researchers, and data scientists keep exploring analytics techniques to detect stock market trends. This gave rise to the concept of algorithmic trading, which uses automated, pre-programmed trading strategies to execute orders.

## **Scope**

Despite the volatility, stock prices aren’t just randomly generated numbers. So, they can be analysed as a sequence of discrete-time data; in other words, time-series observations taken at successive points in time (usually on a daily basis). Time series forecasting (predicting future values based on historical values) applies well to stock forecasting.

Because of the sequential nature of time-series data, we need a way to aggregate this sequence of information. From all the potential techniques, the most intuitive one is MA with the ability to smooth out short-term fluctuations.

## **Stock analysis: fundamental analysis vs. technical analysis**

When it comes to stocks, fundamental and technical analyses are at opposite ends of the market analysis spectrum.

* Fundamental analysis:
  + Evaluates a company’s stock by examining its intrinsic value, including but not limited to tangible assets, financial statements, management effectiveness, strategic initiatives, and consumer behaviors; essentially all the basics of a company.
  + Being a relevant indicator for long-term investment, the fundamental analysis relies on both historical and present data to measure revenues, assets, costs, liabilities, and so on.
  + Generally speaking, the results from fundamental analysis don’t change with short-term news.
* Technical analysis:
  + Analyzes measurable data from stock market activities, such as stock prices, historical returns, and volume of historical trades; i.e. quantitative information that could identify trading signals and capture the movement patterns of the stock market.
  + Technical analysis focuses on historical data and current data just like fundamental analysis, but it’s mainly used for short-term trading purposes.
  + Due to its short-term nature, technical analysis results are easily influenced by news.
  + Popular technical analysis methodologies include moving average (MA), support and resistance levels, as well as trend lines and channels.

For our exercise, we’ll be looking at technical analysis solely and focusing on the Simple MA and Exponential MA techniques to predict stock prices. Additionally, we’ll utilize LSTM (Long Short-Term Memory), a deep learning framework for time-series, to build a predictive model and compare its performance against our technical analysis.

As stated in the disclaimer, stock trading strategy is not in the scope of this article. I’ll be using trading/investment terms only to help you better understand the analysis, but this is not financial advice. We’ll be using terms like:

* trend indicators: statistics that represent the trend of stock prices,
* medium-term movements: the 50-day movement trend of stock prices.

## **Problem Statement**

Time Series forecasting & modelling plays an important role in data analysis. Time series analysis is a specialized branch of statistics used extensively in fields such as Econometrics & Operation Research. Time Series is being widely used in analytics & data science. Stock prices are volatile in nature and price depends on various factors. The main aim of this project is to predict stock prices using Long short-term memory (LSTM).

# Chapter 2 Literature Survey

In this section we will review Key Terms, Technical Indicators and probabilistic models in depth.

### 2.1 Introduction to Literature Review

Stock market prediction is an area of research which walks hand in hand with all the other businesses in the world. Hence, it is known as 'Mother of all Businesses’. Before trying to jump into analysis we need to understand the following terms.

#### **2.1.1 Key Terms**

##### **2.1.1.1 Interval**

Millions of trades take place every second. Hence, for analysis we need to classify these trades based on the interval at which they took place. We can divide these trades into intra-day and long-term intervals. Intra-day can be sub-classified into 1 minute, 2, 5, 10, 15, 30 and 60 minutes. Long term intervals can be classified into daily, weekly, monthly and so on.

##### **2.1.1.2 Tick Prices**

At every interval any stock will have 4 types of prices associated with it. High price, Low price, Open Price and Close price. High price is the highest value at which it was traded in that particular interval. Low price is the lowest price which it reached in that interval. Open price is the first trade which took place at that particular interval and Close price the price at which the stock was last traded at that interval

##### **2.1.1.3 Trend**

At any instance of time, any stock will have either higher demand than supply or lower demand than available supply. Hence, we can classify the Trend into two types namely Bearish and Bullish. A stock is said to be in Bullish Trend if it has higher demand than its supply at that instance of time. If the stock has higher available supply when compared with demand, it is said it be in Bearish Trend.

##### **2.1.2 Data Representation**

Data need to be represented in some format before we can start analyzing it. In this section we will discuss how can we represent data for better interpretation.

###### **2.1.2.1 Japanese Candle**

Diagram, schematic

Description automatically generated

Figure 1 Japanese Candle

Figure 1 represents two different candle sticks. The green candle represents gain, and red candle represents loss. This candle stick always contains all attributes associated with an interval (Example: tick prices). Since it represents an interval what we are looking at, we will refer to a candle stick as an interval in the rest of this report.

###### **2.1.2.2 Line Graph**

Line graph is a graphical way of representing data points in stock market based only on close price or current trading price of the stock.



Figure 2. Line Graph

##### **2.1.3 Accuracy**

Accuracy is defined as the result of number of right decisions made divided by total number of decisions.

###### **2.1.3.1 Accuracy based on Close Price**

When we calculated accuracy, if we relied on next interval's close price for decision verification, we have mentioned it as Accuracy based on Close price in the rest of the report. When we make a Buy call, to verify based on close price, we check if next interval's close price is above current interval's close price. If so, we consider the decision as a right decision. Also, if we make a short sell, to verify the decision based on close price, we check if next interval's close price is lesser than that of current interval's close price. If so, we consider the decision as right decision.

###### **2.1.3.2 Accuracy based on Low/High Price**

When calculating accuracy, if relied on next interval's low/high price for decision verification, we have mentioned it as Accuracy based on high/low price in the rest of the report. When we make a Buy call, to verify the decision based on high price, we check if current interval's close price is lesser than next interval's high price. [4] If so, we consider the decision as a right decision. Similarly, if we make a short sell, to verify the decision based on low price, we check if next interval's low price is lesser than current interval's close price. If so, we consider this decision as right decision.

###### **2.1.3.3 Short Sell**

Short sell is a type of trade where we sell the stock first, before buying it and buy it at a later point of time. Investors perform this trade if they think stock market is going to follow Bearish trend.

###### **2.1.3.4 Buy Call**

Buy call is a type of trade where investors invest by buying the stock first and selling it at a later point of time. They perform this trade if they think, the stock in which they are investing will follow Bullish trend and can yield them profit.

#### **2.1.4 Technical Indicators**

Technical Indicators are the properties associated with any stock based on its tick prices (2.1.1.2). Calculation of each indicator is mentioned in the following subsections.

Interpretation of each of the indicator is explained in detail in section 3.0.

###### **2.1.4.1 Simple Moving Average (SMA)**

Simple Moving Average is calculated exclusively based on close price of the stock which we are trying to analyze. For instance, if we need to calculate SMA of 'x' intervals we need to get close prices of previous 20 intervals and divide it by 'x'. Hence the first available SMA value will correspond to xth interval. To calculate we add all close prices starting from the current interval looking back for n number of intervals. In this case n stands for number of intervals for which we need to calculate SMA.

###### **2.1.4.2 EXPONENTIAL MOVING AVERAGE (EMA)**

Exponential Moving Average is also calculated based on close price of the stock which we are analyzing. If we need to calculate EMA of 'x' intervals, we first need to calculate SMA (2.1.2.1) till xth interval and for every subsequent interval we calculate EMA based on the following formula.

EMS =

Hence, the first available EMA value will be corresponding to xth interval. EMA

moves hand in hand with price of stock. Higher the number of intervals we choose to calculate EMA, higher the stability of EMA. [18] Hence, we chose 2, 5, 10, 20, 50 and 100 intervals for calculating EMA.

###### **2.1.4.3 Relative Strength Index (RSI)**

Relative Strength Index is calculated based on SMA and close price of the stock for the given interval. We must get familiar with the following terms to better understand the calculation of RSI: Gain, Loss. If the close price of a stock at a given interval is greater than its open price, the stock resulted in Gain, vice versa it resulted in Loss. [18]

Here are the formulae to calculate RS and RSI. [3, 4]

***RSI=100 -***

RSI indicates the strength of the current trend. Higher the value of interval we choose, we get stable RSI values. We need to find out a threshold value. If RSI falls below its threshold, it is an indication of sellers taking over buyers. If RSI value rises over its threshold, it indicates that buyers are taking over sellers and stock prices will go high. [14]

###### **2.1.4.4 Bollinger Band (BB)**

Bollinger Band is calculated based on Standard Deviation (2.1.2.3) and close price of the stock at a given interval. [12] Bollinger Bands are calculated based on the following formulae:

Upper Band =SMA [20] +(2\*SD [20])

Lower Band=SMA [20] -(2\*SD [20])

Bollinger Band’s values provide an insight on how much more the stock can rise if it is in Bullish trend. Or, how much can it fall, if the stock is in Bearish trend.

###### 2**.1.4.5 Fast Stochastic (FS)**

Fast Stochastic is the technical indicator which involves two values, namely, %k and %d. For a given interval these values are calculated based on current close price of the stock,[18] lowest low price in the look-back period and highest high price of the stock in the look-back period. Lowest low and highest high are the lowest price and highest price at which the stock was traded in the given look-back period with respect to given interval respectively.

Here are the formulae to calculate %k and %d:

**%k=**

At a given interval %k’s value falls below %d’s value, market takes Bearish trend.

Otherwise, if %k’s value rises above %d’s value it indicates Bullish trend. However, it is very difficult to identify how long will the trend predicted from FS is valid.

###### **2.1.4.6 Moving Average Convergence Divergence (MACD**)

This indicator is calculated based on exponential moving average. To calculate MACD we first need to compute EMA of 9 intervals and EMA of 26 intervals. Now starting from 27th interval till 35th interval subtract EMA (26) from EMA (9) and store the result for further processing.[4] This value represents upper MACD indicator. With these values now calculate EMA of previously stored results starting from 36th interval.

This represents lower MACD indicator. Minimum number of intervals needed for

MACD is 35 and is its only limitation. [5]

Whenever lower MACD value falls below upper MACD value it shows a trend reversal from Bullish to Bearish. If lower MACD value rises above its upper MACD value it indicates a trend reversal from Bearish to Bullish.

###### **2.1.4.7 Williams Average (WA)**

Williams Average is calculated based on current close, highest high and lowest low (2.1.2.5). Here is the formula to calculate Williams Average: In order to calculate this indicator, we first subtract current close from highest high and store the result and call it HC. Now we subtract lowest low from highest high, store it and call it HL. With these results we now calculate Williams Average by the following formula.

***WA=\*-100***

WA indicates oversold and overbought conditions. [4] Whenever it reaches close to 0 it shows oversold condition. If WA value is closer to 100 it shows overbought condition. [12] We need to identify the optimal threshold for which this indicator gives accurate results.

###### **2.1.4.8 Fibonacci Series**

Fibonacci Series is calculated using the following formula: [4, 5, 16]

***nth Fibonacci Number =***

Fibonacci series helps us to identify support level or resistance level of a given stock. Whenever we make a decision based on certain indicator, we can use this indicator to identify we made the right decision before making a trade.

###### **2.1.4.9 Rate of Change (ROC)**

To calculate ROC, we need to know the current trading price of the current interval and close price of the interval which is look-ahead parameter intervals previous to current interval. Here is the formula to calculate ROC.

***Rate of Change=\*100***

Rate of change is directly proportional to the trend of the stock market. If ROC is lesser than 20 it is an indication that the market is in bearish trend. If ROC is greater than 80 it is a strong signal to buy stocks.

###### **2.1.4.10 Artificial Neural Networks (ANN)**

This model is a multi-level perceptron model which feeds forward the result of every computation to the next level of computation. The author [7] has modified the approach of feed-forward model by adding pseudo input variables into ANN. First version of ANN propagates back and forth to provide final results. [7] Author in his approach has argued that, by modifying the model he could minimize the back-and-forth movement of the data to increase optimization of computer’s computational power. ANN’s are always very less susceptible to noise points in the data. If a model is trained based on noise points accuracy of the prediction would decrease drastically due to increase in false positives. Using this model author is making the decision to hold, sell or buy stocks at given time. He used a Genetic algorithm to select the data from the given data set.[20] This selected data will be given as an input to ANN which would classify the labels based on the weight of computed results. In his results he proves that 2 layer ANN is more feasible than a single layer perceptron. The author [9] has improved the model by embedding pattern matching with basic ANN. Pattern matching algorithms are always highly susceptible to noise.[13] However, for given patterns the algorithm will detect the re-occurrence if any. Hence, by embedding this advantage with ANN author has opened the doors of fortunes. This improved model can now be used for prediction on data that contains extreme noise (stock market often is the case). He has also proved the same with his results in the publication. [9]

###### **2.1.4.11 Hidden Markov Model (HMM)**

Hidden Markov Models are statistical models that are used to determine the most likely sequence of occurrence given a sequence of data. This model consists of transition matrix, observation matrix and initial set of observation distribution, denoted as A, B and pi respectively [21]. This model is used to uncover hidden information behind any set of sequences that has happened. In stock market ticks of respective intervals are taken into consideration for computing initial set of observations and observation distribution. Based on the computed results we then calculate the transition matrix. It is the transition matrix, which author relied on to calculate the likeliness of the trading sequence. When he trained this model with ticks whose results we already know since they are training records. He then calculated its likeliness. [10] Now he calculates likeliness for a sequence whose results we don’t know yet. It is based on this likeliness; we predict the future trend for that particular stock [6]. The author [10] has used tick prices of current interval as 4 of his observations. He also considered close price of next interval, relative to the current interval, as fifth observation for training his model. Based on these data he computes A, B and pi as discussed previously. This model now can be used to predict next most likely observation. [10, 22, 23]

# Chapter 3 Methodology

### PROPOSED SYSTEMS

The prediction methods can be roughly divided into two categories, statistical methods and artificial intelligence methods. Statistical methods include logistic regression model, ARCH model, etc. Artificial intelligence methods include multi-layer perceptron, convolutional neural network, naive Bayes network, back propagation network, single-layer LSTM, support vector machine, recurrent neural network, etc. They used Long short-term memory network (LSTM).

### 3.1.1 Time Series Analysis

Time series analysis involves the sequential plotting of a set of observations or data points at regular intervals of time. By studying previous outcomes and their progression over time, we can make future predictions with the help of these studied observations.

The approach to problems related to time series analysis is unique in its own way. Most machine learning problems utilize a dataset with certain outcomes to be predicted, such as class labels. This statement means that most machine learning tasks usually utilize an independent and dependent variable for the computation of a particular question.

This procedure involves machine learning algorithms analysing a dataset (say, XX) and using the predictions (YY) to form an assessment and approach to solving the problem statement. Most supervised machine learning algorithms perform in this manner. However, time series analysis is unique because it has only one variable: *time*. We will dive deeper into how to solve the stock market price prediction task with deep learning in the next part of this article. For now, our primary objective will be understanding the terms and important concepts required for approaching this task. Let us begin!

The ordered sequence of data points in regular intervals of time constitutes the primary elements of time series analysis. Time series analysis has a wide range of applications in the practical world. Hence, the study of time series is significant to understand, or at the very least, to gain basic knowledge about what you can expect in the near future. Apart from the stock market price prediction model that we will build in the next part of these articles, time series analysis has applications in economic forecasting, sales forecasting, budgetary analysis, process and quality control, detection of weather patterns, inventory and utility studies, and census analysis, among many others.

Understanding previous behavioural patterns of data elements is critical. Consider an example of business or economic forecasting. When you can extract useful information from the previous patterns, you can plan the future accordingly. More often than not, the predictions made with the help of time series analysis and forecasting yield good results. These will help users to plan and evaluate current accomplishments.

Essential components of time series analysis

Time series analysis is all about the collection of previous data to study the patterns of various trends. By conducting a detailed analysis of these time series forecasting patterns, we can determine the future outcomes with the help of our constructive deep learning models. While there are other methods to determine the realistic results of future trends, deep learning models are an outstanding approach to receive some of the best predictions for each concept.

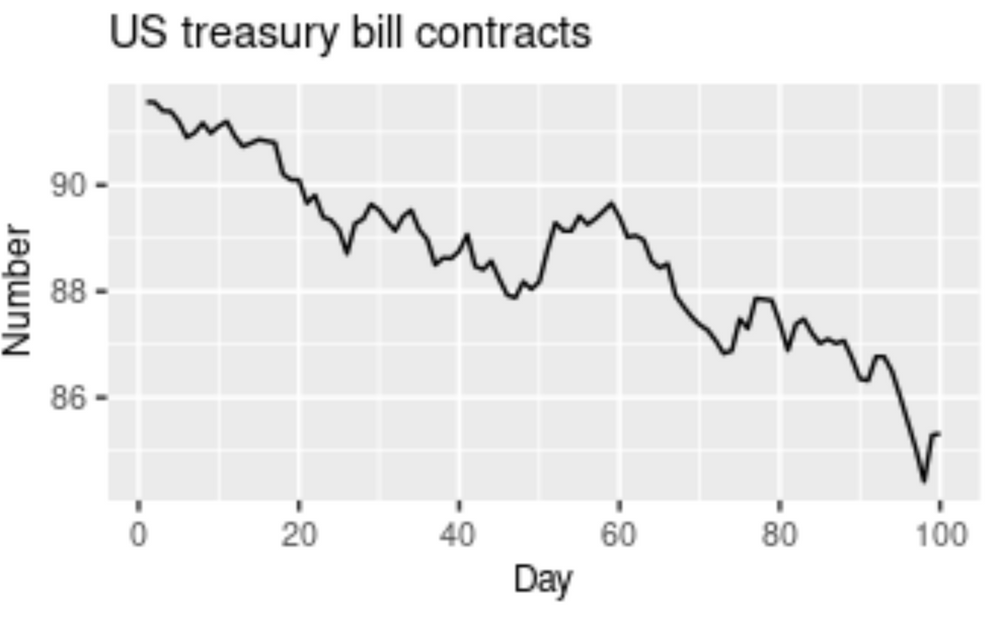
Let's now focus on the components of time series forecasting, namely Trend, Seasonality, Cyclicity, and Irregularity. These four components of time series analysis refer to types of variations in their graphical representations. Let us look at each of these concepts individually and try to gain more intuition behind them with the help of some realistic examples.

**Trend**

A Trend in time series forecasting is defined as a long period of time with a consistent increase or decrease in the data. There may be slight fluctuations in the data points and elements at various instances of time, but the overall variation and direction of change remain constant for a longer duration of time. When a Trend goes from a long duration of constant increase to a long duration of constant decrease, this is often referred to as a "Changing Direction" trend.

There a few terminologies used to define the type of trend that we are dealing with in time series forecasting. A slightly or moderately increasing trend over a long duration of time can be referred to as an uptrend, while a slightly or moderately decreasing trend over a long duration of time is called a downtrend. If the trend is following a consistent pattern of gradually increasing or gradually decreasing, and there is not much effect in the overall pattern of the graph, then this trend can be referred to as a horizontal or stationary trend. The graphical representation shown in the above image has a downward trajectory over a long period of time. Hence, this image shows the representation of a downward trend, also called a downtrend.

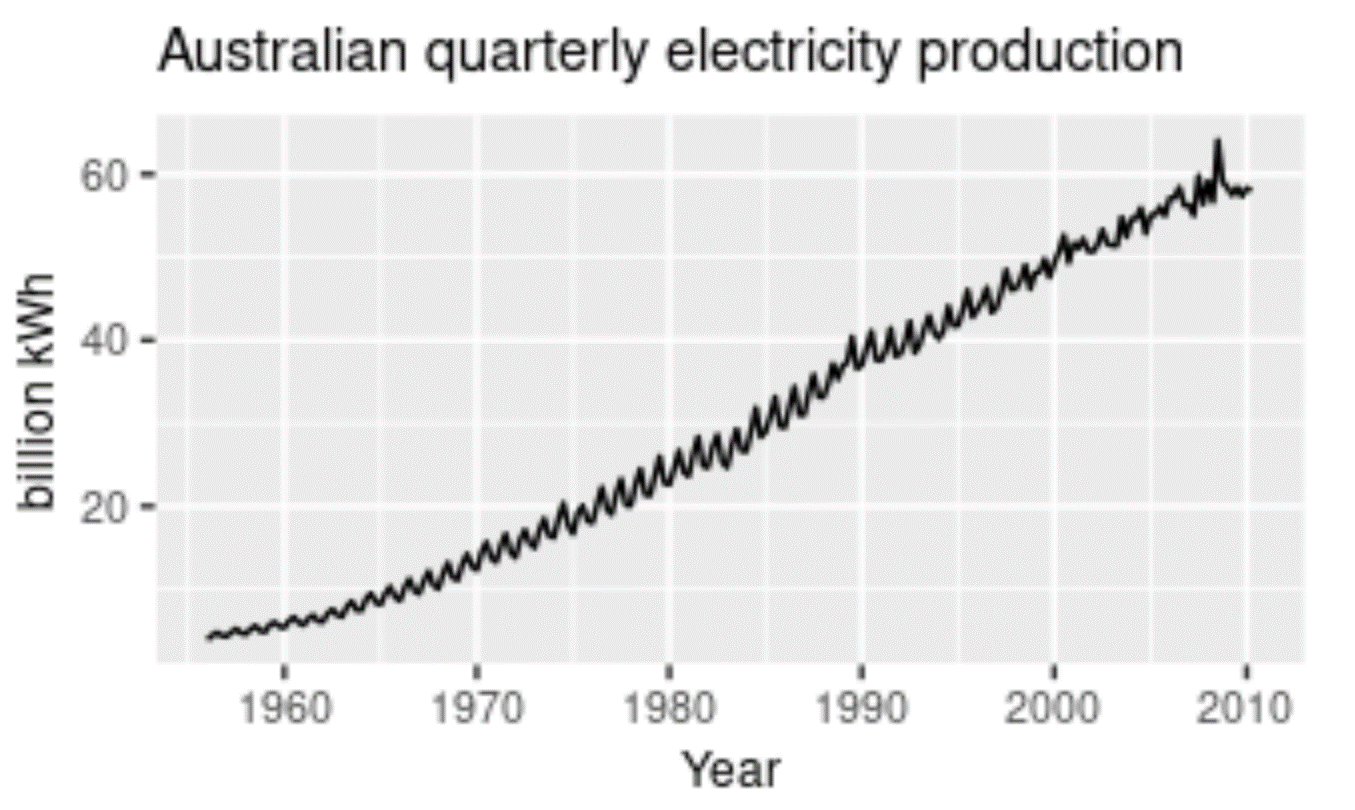
Let us consider an example to understand the concept of trend better. Successful companies like Amazon, Apple, Microsoft, Tesla, and similar tech giants have a reasonably performing upward stock price curve. We have learned that we can determine this upward rise in stock prices as an uptrend. While successful companies have an uptrend, some companies that are not performing that well in the stock market have downward trajectories in stock prices, or a downtrend. Companies that are making a neutral profit rate with decent amounts of profits and losses at regular intervals are determined as a horizontal or stationary trend.



**Seasonality**

A pattern that is affected or influenced by seasonal frequency variations are termed to be seasonal. This component of time series analysis can vary from time stamps like quarterly, monthly, or half-yearly. However, an important point to note is that these fluctuations usually take place within the period of a year. Whenever the frequency is fixed and known and also occurs on a timely basis, usually within a year, this component of time series analysis is known as seasonality.

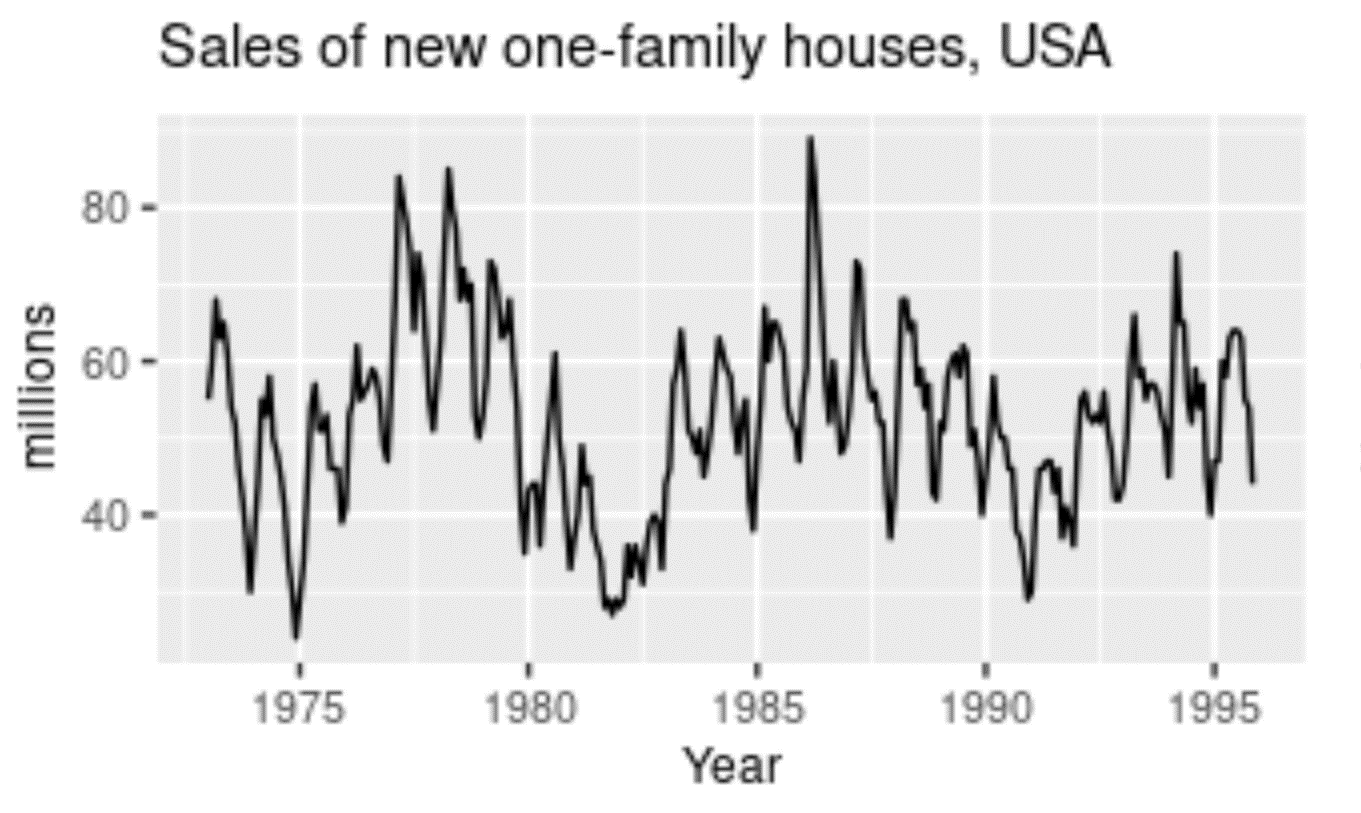
To consider a realistic example, think about the sale of certain seasonal fruits. Fruits like watermelon will have increased sales during the summer season, while during the winter seasons, the sales of watermelons will gradually reduce. Similarly, seasonal fruits like apple have higher sales during the winter seasons in comparison to the other seasons. Ice cream and tender coconuts are other examples of food products that have increased sales during the summer season while experiencing a dip in sales during other seasons. Apart from food products, a period or month like April might have higher shares of investment and experience a dip in shares until six months later in October, where it has a peak rise. This pattern can also be regarded as a seasonality.



**Cyclicity**

When a pattern exhibits a rise and fall of mixed frequencies, and the graphical representation has peaks and troughs that occur randomly over a period of time, it is called a cyclic component. The duration of these occurrences usually ranges over the period of at least one year. Stock prices of certain companies that are hard to predict usually have cyclic patterns where they flourish during a certain period, while having lower profits at other times. Cyclic trends are some of the hardest for our deep learning models to predict. The graphical representation above shows the cyclic behaviour of house sales over the span of two decades.

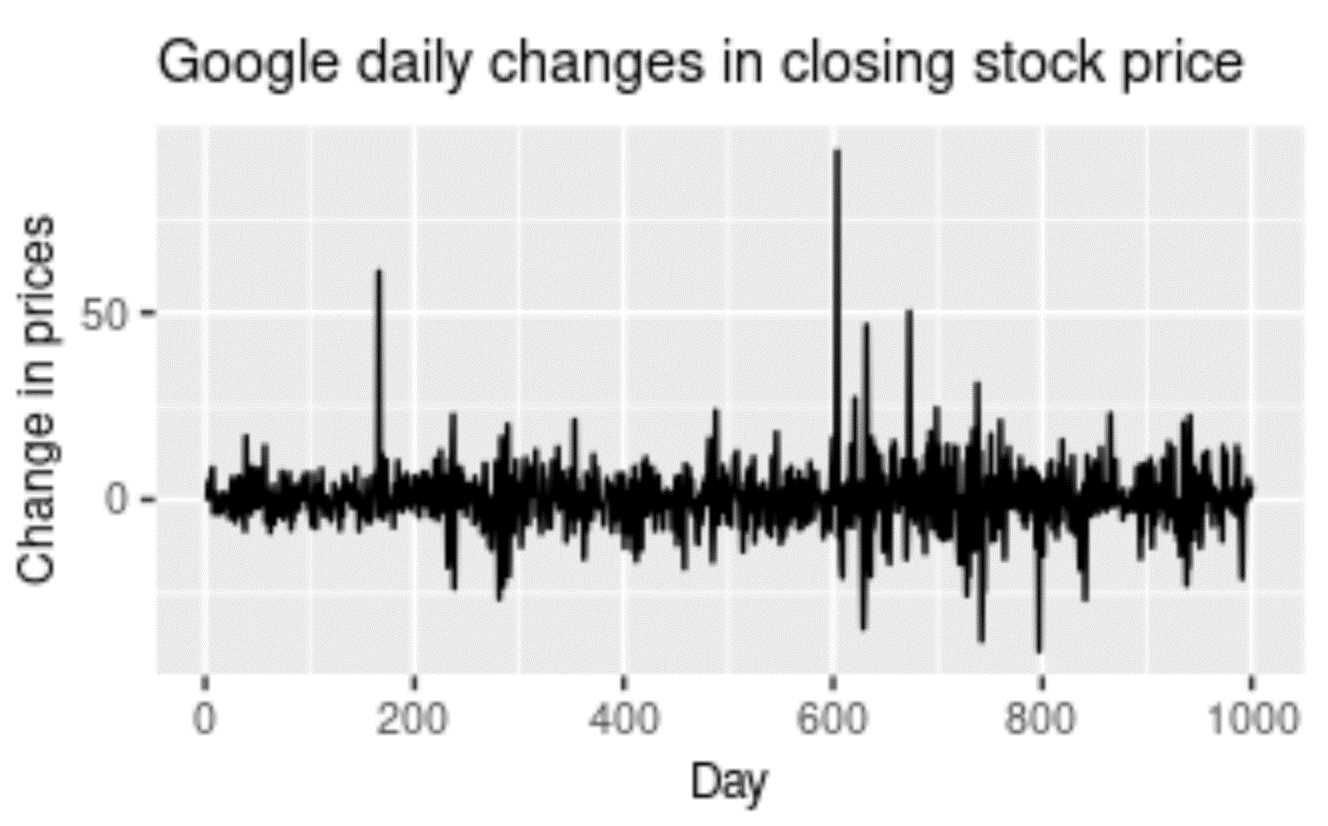
A great realistic example of cyclic behavioural patterns is when a person decides to invest in their own start-up. During the set-up and progression of start-ups, every business experiences a cyclic phase. These cyclic fluctuations often occur in business cycles. Usually, the phases of a start-up would include the investment stage, which would have a slightly negative impact on our prices. The next phases could include your marketing and profitable stages, where you start to earn profits from your successful start-up. Here, you experience an increase in the graphical curve. However, you will eventually experience a depreciation phase as well. These will show lesser profits until you make constant improvements and investments. This procedure begins the cyclic stage again, lasting for periods of a few years. Rinse and repeat.



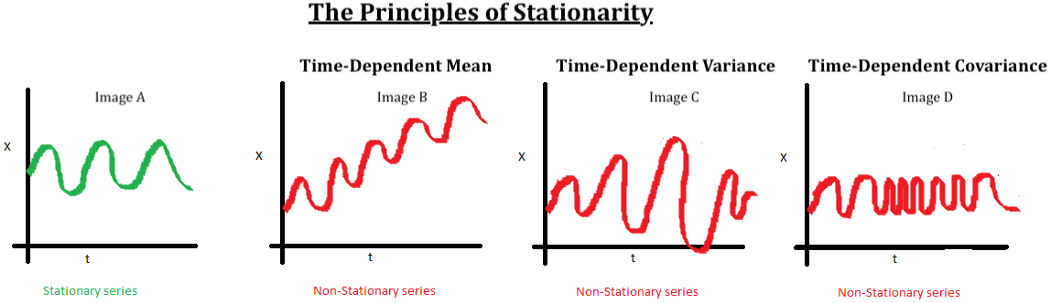
**Irregularity**

Irregularity is a component that is almost impossible to make accurate predictions for with a deep learning model. Irregularity, or random variations (as the name suggests), involves an abnormal or atypical pattern where it becomes hard to deduce the occurrences of the data elements with respect to time. Above is a graphical representation of the Google Stock Price changing rapidly with an irregular and random pattern; this is hard to read. Despite the information and data patterns present, the modelling procedure for such kinds of representations will be hard to crack. The primary objective of the model is to predict future possibilities based on previous outcomes. Hence, models for irregular patterns are slightly harder to construct.

To provide a realistic example for irregular patterns, let us analyze the current status of the world, where many businesses and other industries are affected on a large scale. The global pandemic is a great example of irregular activity that is impossible for anyone to predict. These disturbances that occur due to a natural calamity or phenomenon will affect the trading prices, stock price charts, companies, and businesses. It is not possible for the model you construct to detect the occurrences of these situational tragedies. Hence, irregular patterns are an interesting component of time series forecasting to analyze and study.



Understanding stationary and non-stationary series



Another significant aspect of time series analysis that we will encounter in this article is the concept of stationarity. When we have a certain number of data points or elements, and the mean and variance of all these data points remain constant with time, then we call it a stationary series. However, if these elements vary with time, we call it a non-stationary series. Most forecasting tasks, including stock market prices or cryptocurrency graphs, are non-stationary. Unfortunately, the results obtained by non-stationary patterns are not efficient. Hence, they need to be converted into a stationary pattern.

The entire data preparation process for stock market price prediction in the next article is explained with code snippets. Here, we will discuss a few other topics of importance. Several methods for converting non-stationary series into stationary series include *differencing* and *transformation*. Differencing involves subtracting two consecutive data points from the higher to lower order, while transformation involves transforming or diverging the series. Typically, a log transform is used for this process. For further information on this topic, refer to the link provided in the image source above.

To test for stationarity in Python, the two main methods that are utilized include *rolling statistics* and *the ADCF Test*. Rolling statistics are more of a visual technique that involves plotting the moving average or moving variance to check if it varies with time. *The Augmented Dickey-Fuller test (ADCF)* is used to give us various values that can help in identifying stationarity. These test results comprise some statistics and critical values. They are used to verify stationarity.

### 3.1.2 Long short-term memory network:

Recurrent neural networks, also known as RNNs, are a class of neural networks that allow previous outputs to be used as inputs while having hidden states. RNNs have issues with the transfer of long-term data elements due to the possibility of exploding and vanishing gradients. The fix to these issues is offered by the Long short-term memory (LSTM) model, which is also a recurrent neural network (RNN) architecture that is used to solve many complex deep learning problems. LSTMs are especially useful for our task of Stock Price Prediction Using Deep Learning.

The LSTM architecture, with its effective mechanism of memory cells, is extremely useful for solving complex problems and making overall efficient predictions with higher accuracy. LSTMs learn when to forget and when to remember the information provided to them. The basic anatomy of the LSTM structure can be viewed in three main steps.

The first stage is the cell state that contains the basic and initial data to be recollected. The second stage is the hidden state that consists of mainly three gates: the forget gate, input gate, and output gate. We will discuss these gates shortly. Finally, we have a looping stage that reconnects the data elements for computation at the end of each time step.

#### **Working of LSTM:**

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

The experimental data in this paper are the actual historical data downloaded from the Internet. Three data sets were used in the experiments. It is needed to find an optimization algorithm that requires less resources and has faster convergence speed.

* Used Long Short-term Memory (LSTM) with embedded layer and the LSTM neural network with automatic encoder.
* LSTM is used instead of RNN to avoid exploding and vanishing gradients.
* The historical stock data table contains the information of opening price, the highest price, lowest price, closing price, transaction date, volume and so on.
* The accuracy of this LSTM model used in this project is 57%.

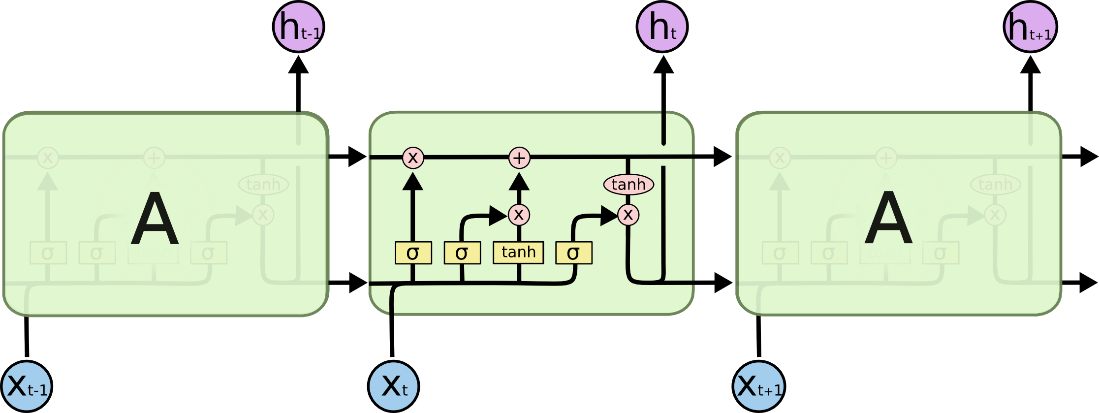


Fig. 4: LSTM Architecture

#### **Forget Gate:**

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

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

#### **Input Gate:**

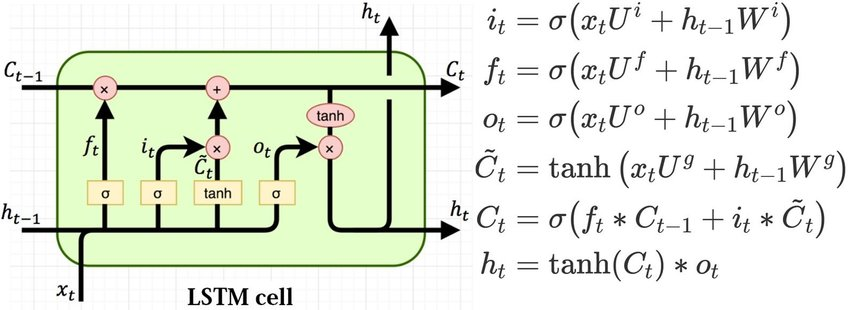
* + 1. Regulating what values need to be added to the cell state by involving a sigmoid function. This is basically very similar to the forget gate and acts as a filter for all the information from hi-1 and x\_t.
    2. Creating a vector containing all possible values that can be added (as perceived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

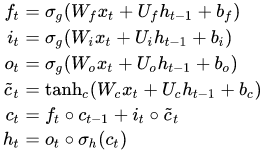
#### **Output Gate:**

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

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

The mathematical computations for LSTMs can be interpreted in many ways. The images shown below are two of the methods in which the mathematical calculations of LSTMs are carried out. Sometimes the bias function is ignored, as shown in the second image. However, we will not cover the intricate details of the first image. You can check out more on that representation from the link provided in the image source. The mathematical equations in consideration of the first LSTM image are pretty accurate for computation purposes.





Where:

X(t)X(t): input vector to the LSTM unit

• F(t)F(t): forget gate's activation vector

• I(t)I(t): input/update gate's activation vector

• O(t)O(t): output gate's activation vector

• H(t)H(t): hidden state vector, also known as output vector of the LSTM unit

• c˜(t)c̃(t): cell input activation vector

• C(t)C(t): cell state vector

• WW, UU, and BB: weight matrices and bias vector parameters which need to be learned during training

* # LSTM
* Inputs: dataset
* Outputs: RMSE of the forecasted data

•

* # Split dataset into 75% training and 25% testing data
* size = length(dataset) \* 0.75
* train = dataset [0 to size]
* test = dataset [size to length(dataset)]

•

* # Procedure to fit the LSTM model
* # Procedure LSTMAlgorithm (train, test, train\_size, epochs)
* X = train
* y = test
* model = Sequential ()
* model.add (LSTM (50), stateful=True)
* model. compile (optimizer='adam', loss='mse')
* model.fit (X, y, epochs=epochs, validation\_split=0.2)
* return model

•

* # Procedure to make predictions
* # Procedure getPredictonsFromModel (model, X)
* predictions = model.predict(X)
* return predictions

•

* epochs = 100
* neurons = 50
* predictions = empty

•

•

•

•

•

•

•

•

•

•

•

•

# Fit the LSTM model

model = LSTMAlgorithm (train, epoch, neurons)

# Make predictions

pred = model.predict(train)

# Validate the model n = len(dataset)

error = 0

for i in range(n): error += (abs(real[i] - pred[i])/real[i]) \* 100 accuracy = 100 - error/n

### SYSTEM ARCHITECTURE

* + - * Preprocessing of data



Fig. 5: Pre-processing of data

* + - * Overall Architecture

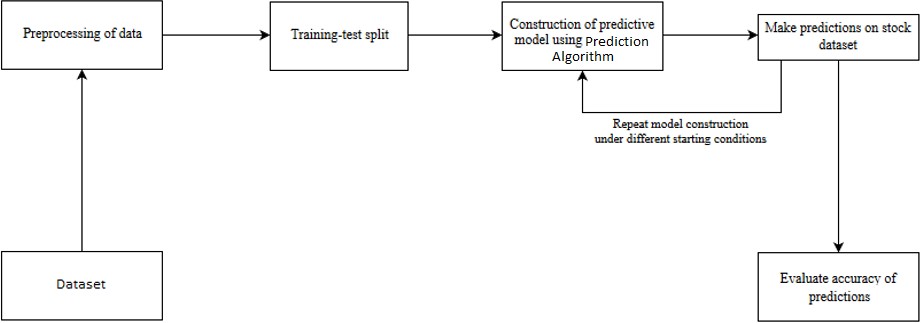


Fig. 6: Overall Architecture

# Chapter 4 Design

### Structure Chart

A structure chart (SC) in software engineering and organizational theory is a chart which shows the breakdown of a system to its lowest manageable levels. They are used in structured programming to arrange program modules into a tree. Each module is represented by a box, which contains the module's name.

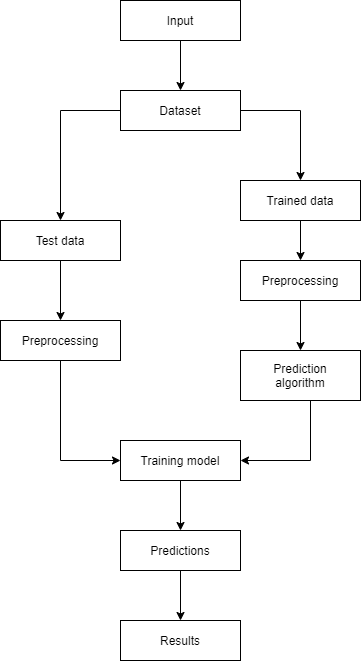


Fig. 7: Training and prediction

### UML Diagrams

A UML diagram is a partial graphical representation (view) of a model of a system under design, implementation, or already in existence. UML diagram contains graphical elements (symbols) - UML nodes connected with edges (also known as paths or flows) - that represent elements in the UML model of the designed system. The UML model of the system might also contain other documentation such as use cases written as templated texts.

The kind of the diagram is defined by the primary graphical symbols shown on the diagram. For example, a diagram where the primary symbols in the contents area are classes is class diagram. A diagram which shows use cases and actors is use case diagram. A sequence diagram shows sequence of message exchanges between lifelines.

UML specification does not preclude mixing of different kinds of diagrams, e.g., to combine structural and behavioral elements to show a state machine nested inside a use case. Consequently, the boundaries between the various kinds of diagrams are not strictly enforced. At the same time, some UML Tools do restrict set of available graphical elements which could be used when working on specific type of diagram.

UML specification defines two major kinds of UML diagram: structure diagrams and behavior diagrams.

Structure diagrams show the static structure of the system and its parts on different abstraction and implementation levels and how they are related to each other. The elements in a structure diagram represent the meaningful concepts of a system, and may include abstract, real world and implementation concepts.

Behavior diagrams show the dynamic behavior of the objects in a system, which can be described as a series of changes to the system over time.

### Use Case Diagram

In the Unified Modelling Language (UML), a use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. To build one, you'll use a set of specialized symbols and connectors. An effective use case diagram can help your team discuss and represent:

* + - * Scenarios in which your system or application interacts with people, organizations, or external systems.
      * Goals that your system or application helps those entities (known as actors) achieve.
      * The scope of your system.

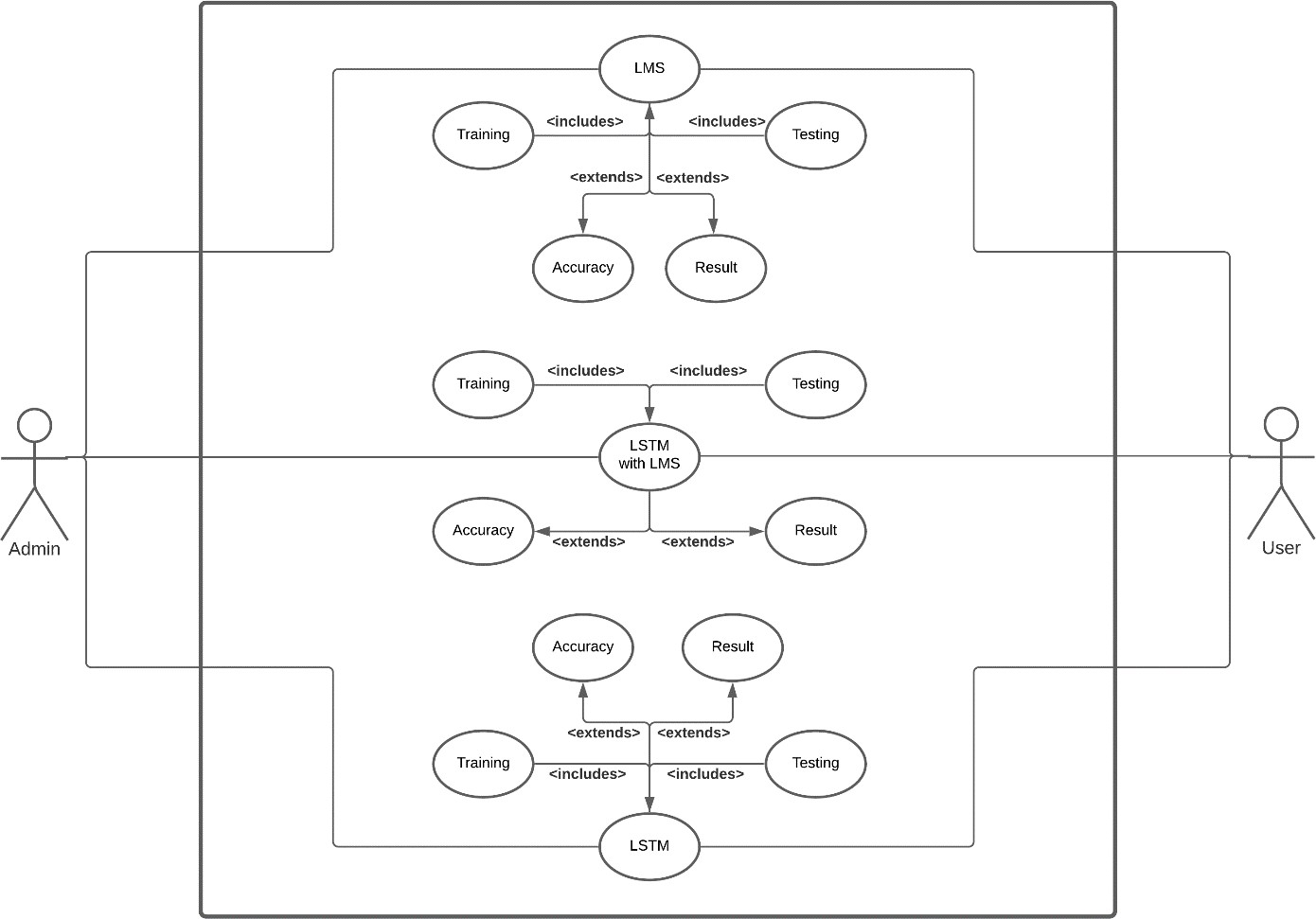


Fig. 8: Using LMS, LSTM and LSTM with LMS in the system

### Sequence Diagram

A sequence diagram is a type of interaction diagram because it describes how and in what order a group of objects works together. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process. Sequence diagrams are sometimes known as event diagrams or event scenarios.

Sequence diagrams can be useful references for businesses and other organizations. Try drawing a sequence diagram to:

* + - * Represent the details of a UML use case.
      * Model the logic of a sophisticated procedure, function, or operation.
      * See how objects and components interact with each other to complete a process.
      * Plan and understand the detailed functionality of an existing or future scenario.

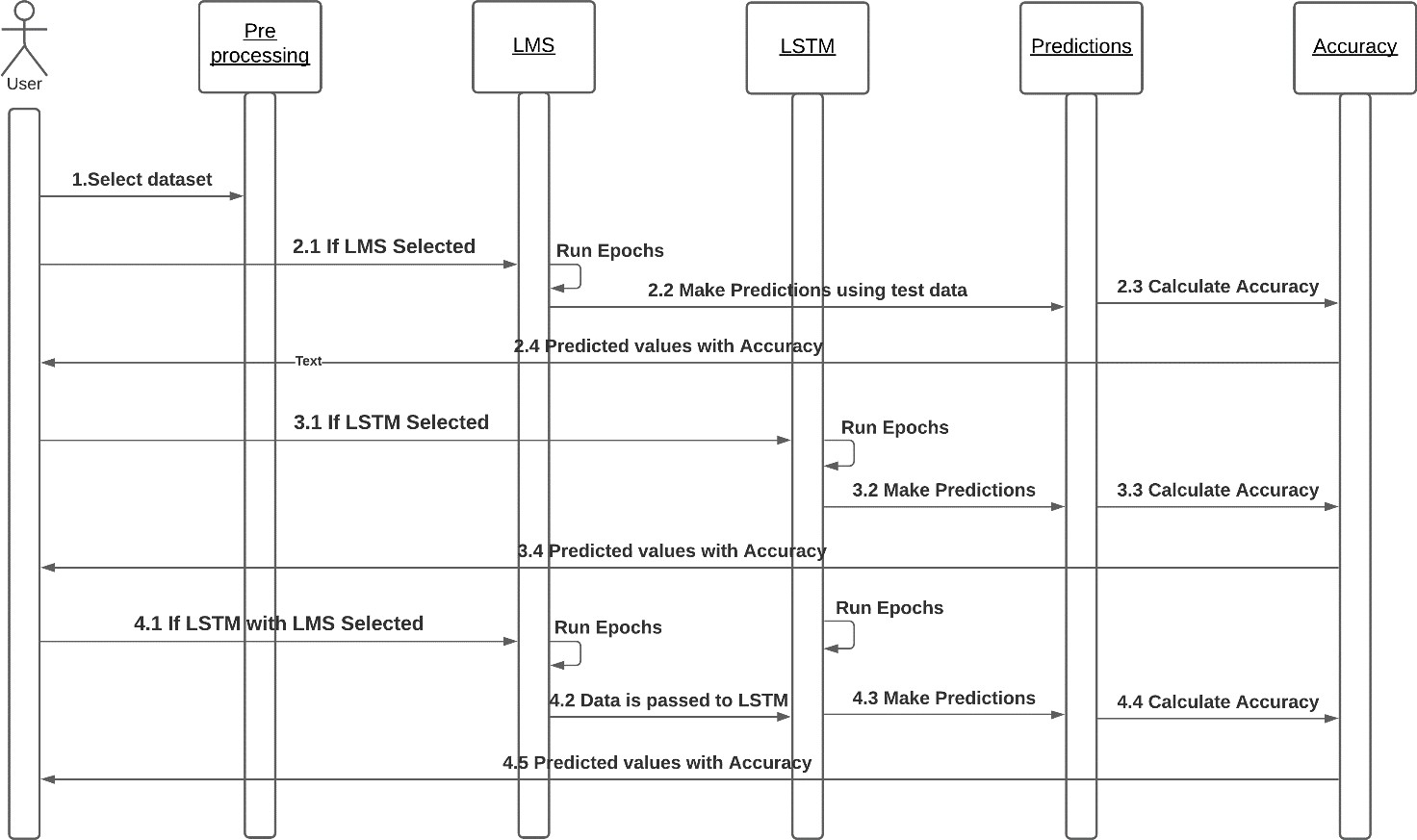


Fig. 9: Execution based on model selection

### Activity Diagram

An activity diagram is a behavioral diagram i.e., it depicts the behavior of a system.

An activity diagram portrays the control flow from a start point to a finish point showing the various decision paths that exist while the activity is being executed.

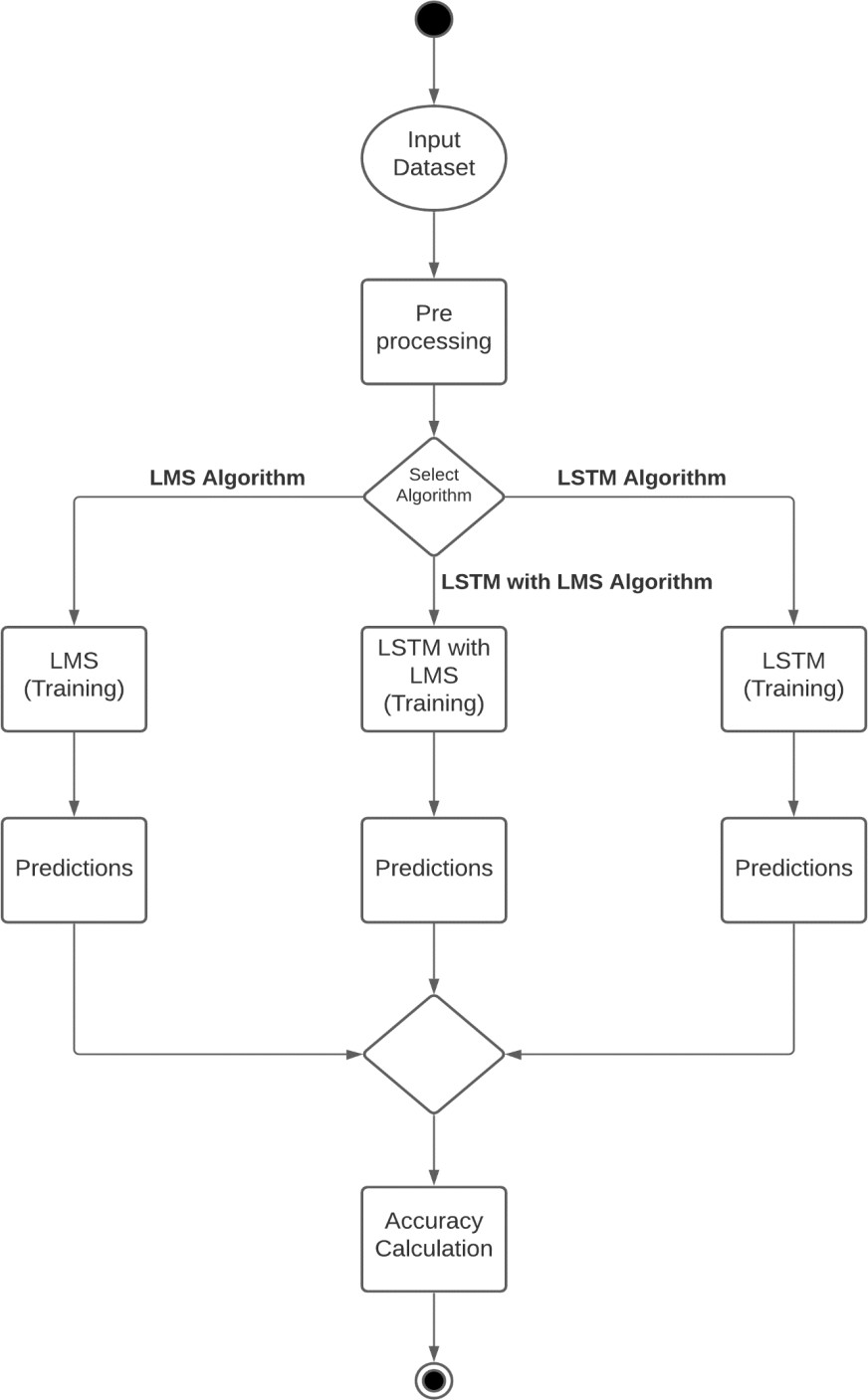


Fig. 10: Execution based on algorithm selection

### Collaboration Diagram

Collaboration diagrams are used to show how objects interact to perform the behavior of a particular use case, or a part of a use case. Along with sequence diagrams, collaboration are used by designers to define and clarify the roles of the objects that perform a particular flow of events of a use case. They are the primary source of information used to determining class responsibilities and interfaces.

The collaborations are used when it is essential to depict the relationship between the object. Both the sequence and collaboration diagrams represent the same information, but the way of portraying it quite different. The collaboration diagrams are best suited for analyzing use cases.

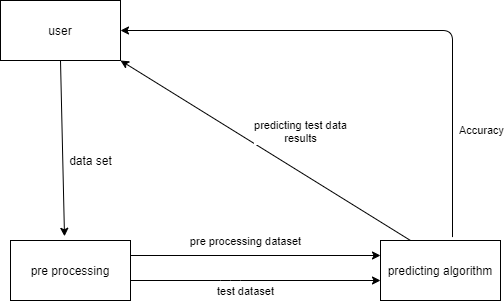


Fig. 11: Data transfer between modules

### Flow Chart

A flowchart is a type of diagram that represents a workflow or process. A flowchart can also be defined as a diagrammatic representation of an algorithm, a step-by-step approach to solving a task. The flowchart shows the steps as boxes of various kinds, and their order by connecting the boxes with arrows.

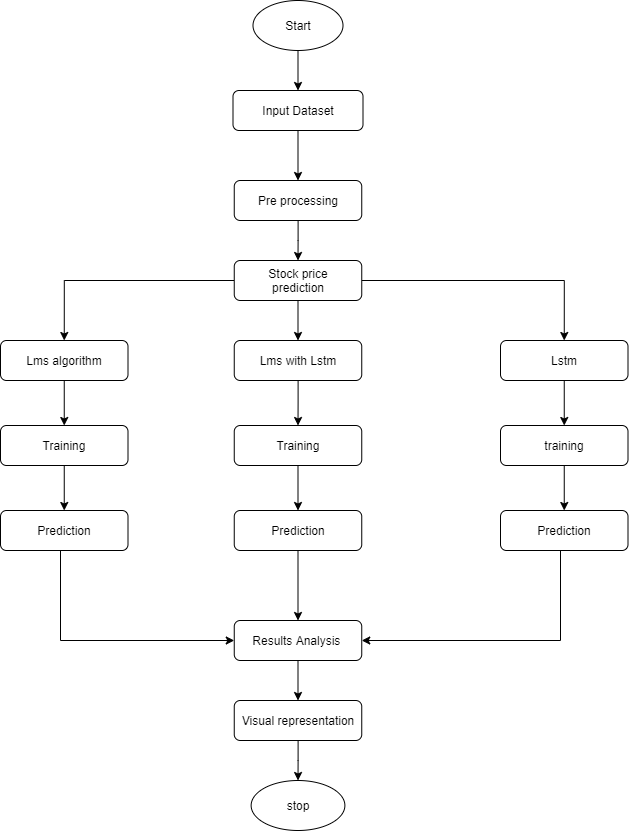


Fig. 12: Flow of execution

### Component Diagram

Component diagram is a special kind of diagram in UML. The purpose is also different from all other diagrams discussed so far. It does not describe the functionality of the system but it describes the components used to make those functionalities.

Component diagrams are used in modeling the physical aspects of object-oriented systems that are used for visualizing, specifying, and documenting component-based systems and also for constructing executable systems through forward and reverse engineering. Component diagrams are essentially class diagrams that focus on a system's components that often used to model the static implementation view of a system.

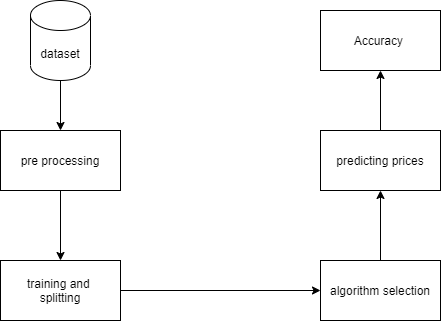


Fig. 13: Components present in the system

# Chapter 5 Experiment Analysis

### System configuration

This project can run on commodity hardware. We ran entire project on an Intel I5 processor with 8 GB Ram, 2 GB Nvidia Graphic Processor, It also has 2 cores which runs at 1.7 GHz, 2.1 GHz respectively. First part of the is training phase which takes 10-15 mins of time and the second part is testing part which only takes few seconds to make predictions and calculate accuracy.

##### **5.1.1 Hardware Requirements:**

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

##### **5.1.2. Software Requirements:**

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

##### **5.1.3 Functional requirements**

Functional requirements describe what the software should do

(the functions). Think about the core operations.

Because the “functions” are established before development, functional requirements should be written in the future tense. In developing the software for Stock Price Prediction, some of the functional requirements could include:

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

Notice that each requirement is directly related to what we expect the software to do. They represent some of the core functions.

##### **5.1.4 Non-Functional requirements**

Product properties

* Usability: It defines the user interface of the software in terms of simplicity of understanding the user interface of stock prediction software, for any kind of stock trader and other stakeholders in stock market.
* Efficiency: maintaining the possible highest accuracy in the closing stock prices in shortest time with available data.

Performance: It is a quality attribute of the stock prediction software that describes the responsiveness to various user interactions with it.

### Sample code

###### **5.2.1 Model Code**

**#!/usr/bin/env python**

**# coding: utf-8**

# In[1]:

import numpy as np

import pandas as pd

import seaborn as sns

from datetime import datetime

import matplotlib.pyplot as plt

from keras.models import Sequential

from keras.layers import Dense,Dropout,LSTM

from sklearn.preprocessing import MinMaxScaler

# In[2]:

AAPL=pd.read\_csv("Database/AAPL.csv")

TSLA=pd.read\_csv("Database/TSLA.csv")

GOOG=pd.read\_csv("Database/GOOG.csv")

MSFT=pd.read\_csv("Database/MSFT.csv")

AMZN=pd.read\_csv("Database/AMZN.csv")

# In[3]:

AAPL.head()

# In[4]:

AAPL.info()

# In[5]:

AAPL.dropna(inplace=True)

# In[6]:

#**change Dtype of column**

AAPL["Date"]=pd.to\_datetime(AAPL["Date"])

AAPL["Volume"]=AAPL["Volume"].str.replace(',','')

AAPL=AAPL.astype({"Open":float,"Volume":float})

# In[7]:

#**sort the dataset by date**

AAPL=AAPL.sort\_values(by='Date',ignore\_index=True)

# In[8]:

AAPL

# In[9]:

#**Drop rows having Date < '2015-01-01'**

AAPL=AAPL[AAPL['Date']>='2015-01-01'].reset\_index(drop=True)

# In[10]:

AAPL.describe()

# In[11]:

TSLA.dropna(inplace=True)

GOOG.dropna(inplace=True)

AMZN.dropna(inplace=True)

MSFT.dropna(inplace=True)

**# ### change Dtype of column**

# In[12]:

TSLA["Date"]=pd.to\_datetime(TSLA["Date"])

TSLA["Volume"]=TSLA["Volume"].str.replace(',','')

TSLA=TSLA.astype({"Open":float,"Volume":float})

# In[13]:

GOOG["Date"]=pd.to\_datetime(GOOG["Date"])

for col in ["Open","High","Low","Close","Adj. Close","Volume"]:

GOOG[col]=GOOG[col].str.replace(',','')

GOOG=GOOG.astype({"Open":float,"High":float,

"Low":float,'Close':float,

'Adj. Close':float,"Volume":float})

# In[14]:

MSFT["Date"]=pd.to\_datetime(MSFT["Date"])

MSFT['Open']=MSFT['Open'].str.replace(',','')

MSFT["Volume"]=MSFT["Volume"].str.replace(',','')

MSFT=MSFT.astype({"Open":float,"Volume":float})

# In[15]:

AMZN["Date"]=pd.to\_datetime(MSFT["Date"])

for col in ['Open','High','Low','Close','Adj. Close','Volume']:

AMZN[col]=AMZN[col].str.replace(',','')

AMZN=AMZN.astype({"Open":float,"High":float,

"Low":float,"Close":float,

"Adj. Close":float,"Volume":float})

**# ### sort the dataset by date**

# In[16]:

AMZN=AMZN.sort\_values(by='Date',ignore\_index=True)

MSFT=MSFT.sort\_values(by='Date',ignore\_index=True)

GOOG=GOOG.sort\_values(by='Date',ignore\_index=True)

TSLA=TSLA.sort\_values(by='Date',ignore\_index=True)

**# ### Drop rows having Date < '2015-01-01'**

# In[17]:

TSLA=TSLA[TSLA['Date']>='2015-01-01'].reset\_index(drop=True)

GOOG=GOOG[GOOG['Date']>='2015-01-01'].reset\_index(drop=True)

MSFT=MSFT[MSFT['Date']>='2015-01-01'].reset\_index(drop=True)

AMZN=AMZN[AMZN['Date']>='2015-01-01'].reset\_index(drop=True)

**# # Exploratory Data Analysis**

# In[18]:

Companies=[AAPL,TSLA,GOOG,MSFT,AMZN]

Companies\_Title=['Apple','Tesla','Google','Microsoft','Amazon']

# In[19]:

plt.figure(figsize=(20,20))

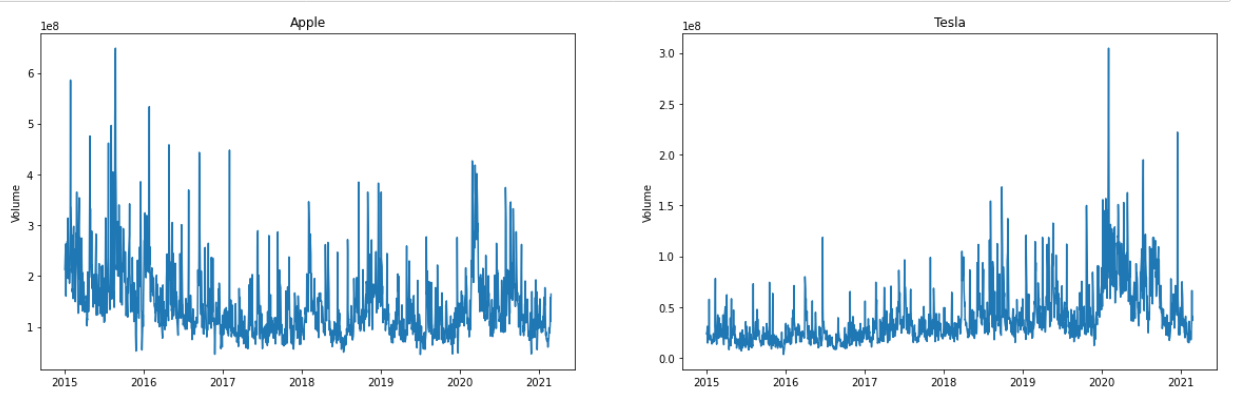
for index,company in enumerate(Companies):

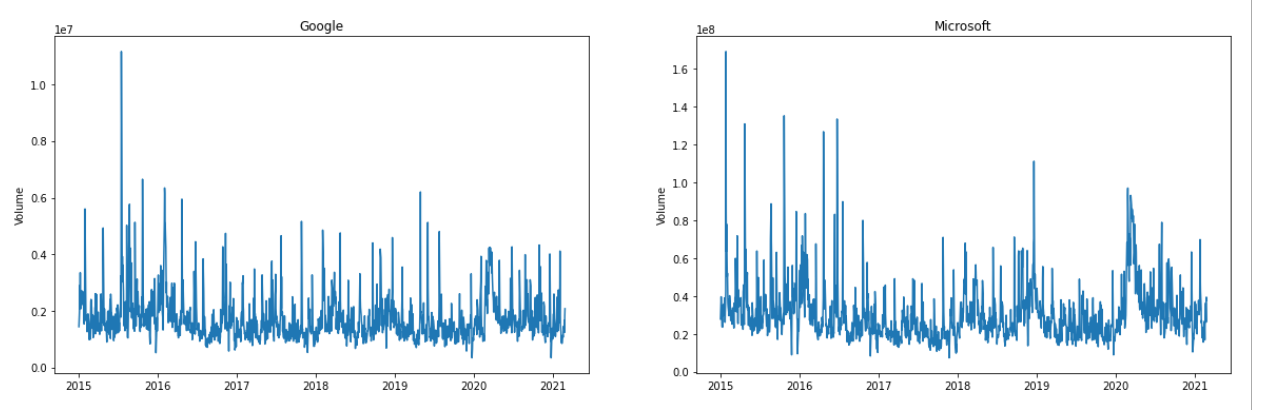
plt.subplot(3,2,index+1)

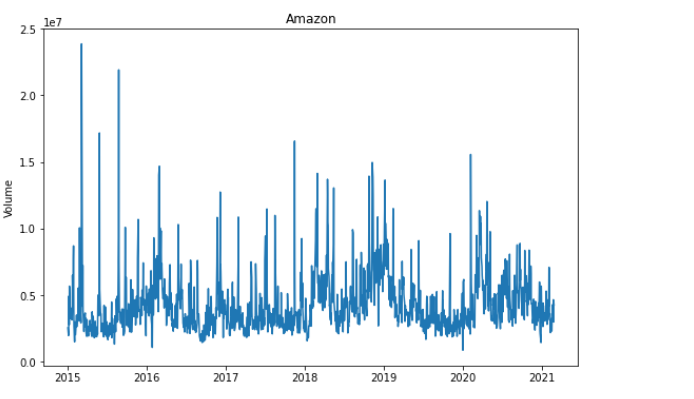
plt.plot(company['Date'],company['Volume'])

plt.title(Companies\_Title[index])

plt.ylabel('Volume')







**# ### What was the moving average of the various stocks ?**

# In[20]:

Moving\_Average\_Days=[10,20,50]

for Moving\_Average in Moving\_Average\_Days:

for company in Companies:

column\_name=f'Moving Average for {Moving\_Average} days'

company[column\_name]=company['Adj. Close'].

rolling(Moving\_Average).mean()

# In[21]:

plt.figure(figsize=(20,20))

for index,company in enumerate(Companies):

plt.subplot(3,2,index+1)

plt.plot(company['Date'],company['Adj. Close'])

plt.plot(company['Date'],company['Moving Average for 10 days'])

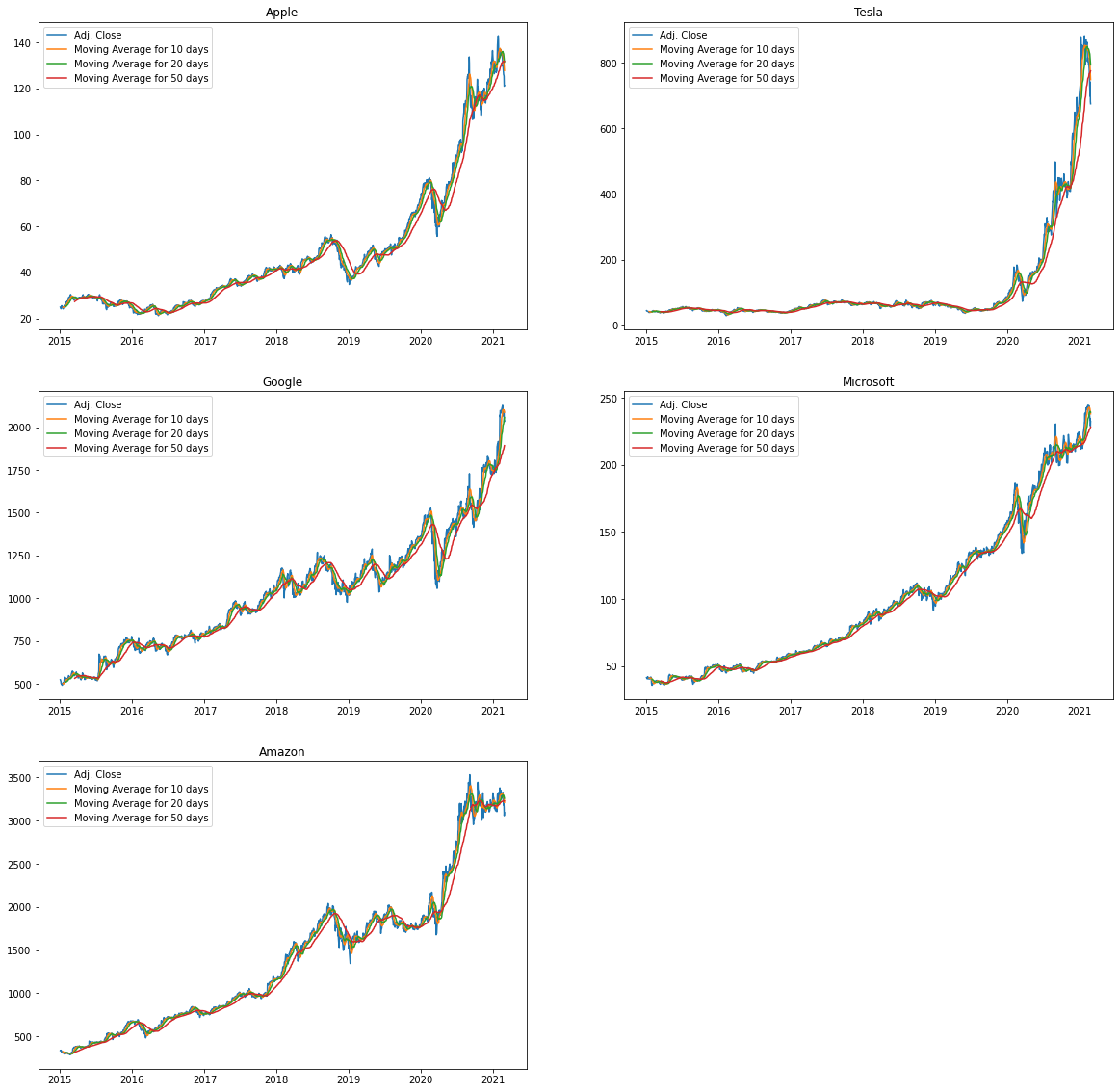
plt.plot(company['Date'],company['Moving Average for 20 days'])

plt.plot(company['Date'],company['Moving Average for 50 days'])

plt.title(Companies\_Title[index])

plt.legend(('Adj. Close','Moving Average for 10 days',

'Moving Average for 20 days','Moving Average for 50 days'))



# In[22]:

Moving\_Average\_Days\_2=[100,200,500]

for Moving\_Average in Moving\_Average\_Days\_2:

for company in Companies:

column\_name=f'Moving Average for {Moving\_Average} days'

company[column\_name]=company['Adj. Close'].

rolling(Moving\_Average).mean()

# In[23]:

plt.figure(figsize=(20,20))

for index,company in enumerate(Companies):

plt.subplot(3,2,index+1)

plt.plot(company['Date'],company['Adj. Close'])

plt.plot(company['Date'],company['Moving Average for 100 days'])

plt.plot(company['Date'],company['Moving Average for 200 days'])

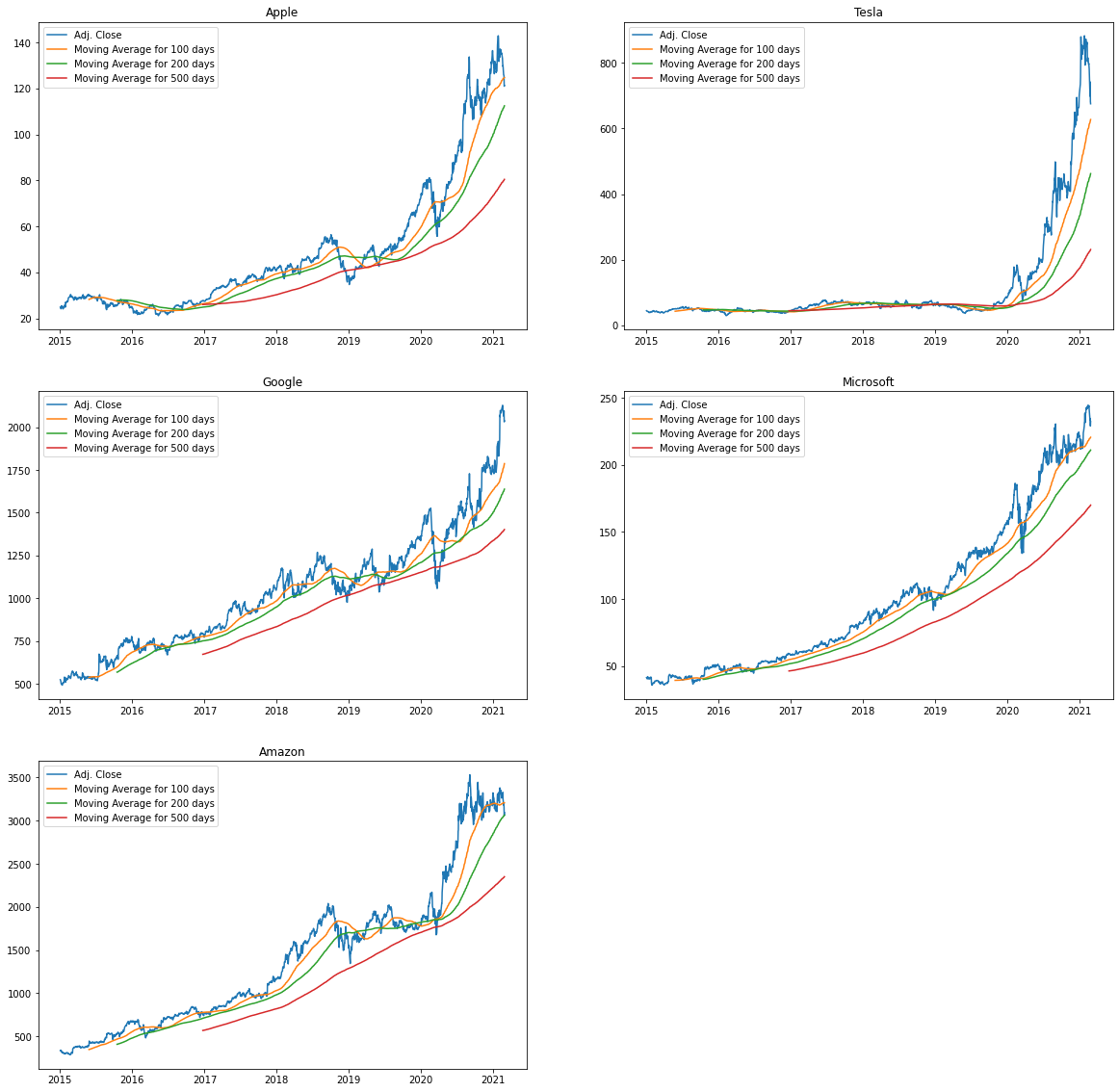
plt.plot(company['Date'],company['Moving Average for 500 days'])

plt.title(Companies\_Title[index])

plt.legend(('Adj. Close','Moving Average for 100 days',

'Moving Average for 200 days',

'Moving Average for 500 days'))



**# ### What was the daily return of the stock on average ?**

# In[24]:

for company in Companies:

company['Daily Return']=company['Adj. Close'].pct\_change()

# In[25]:

plt.figure(figsize=(20,20))

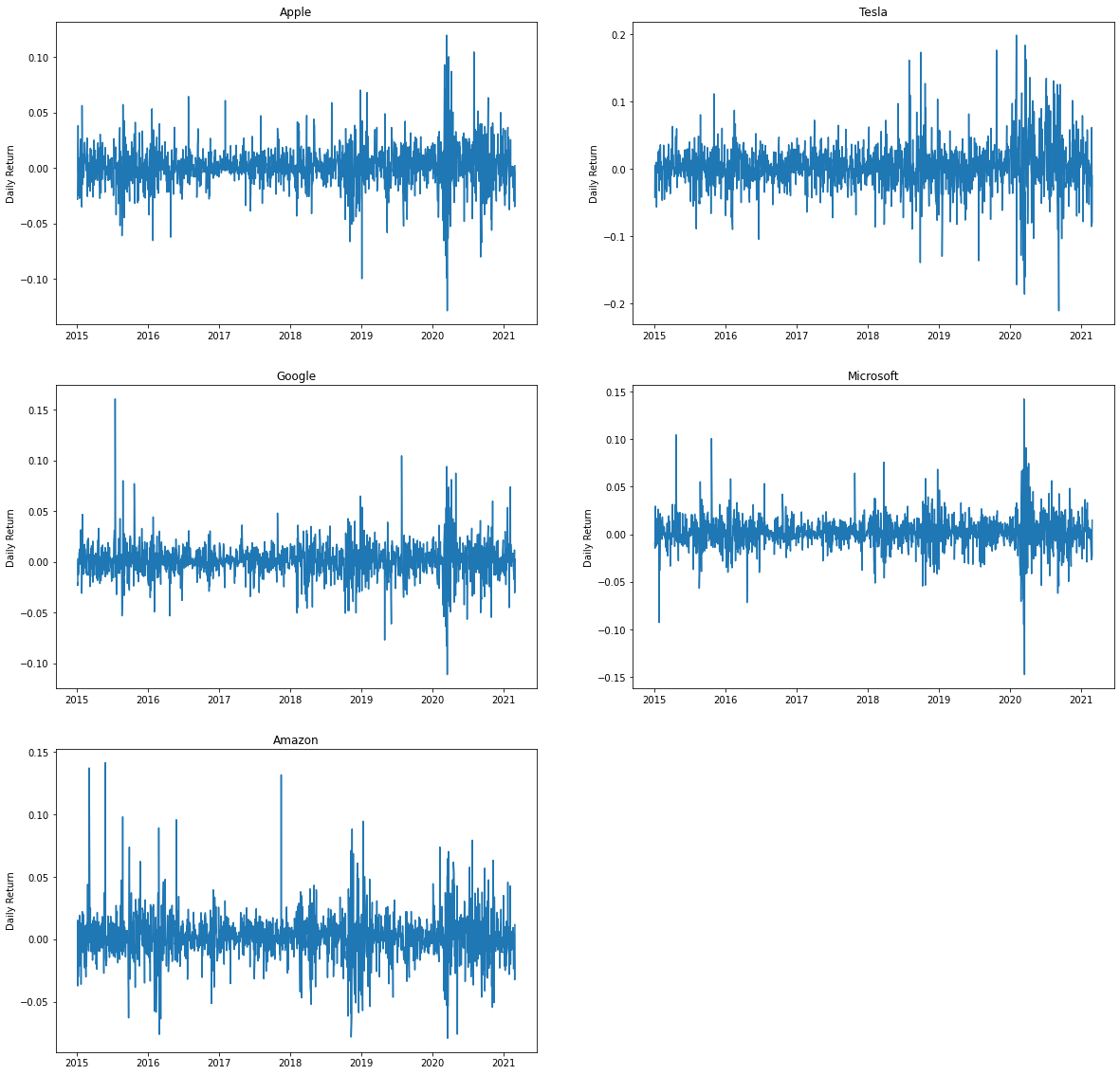
for index,company in enumerate(Companies):

plt.subplot(3,2,index+1)

plt.plot(company['Date'],company['Daily Return'])

plt.title(Companies\_Title[index])

plt.ylabel('Daily Return')



# In[26]:

import warnings

warnings.filterwarnings('ignore')

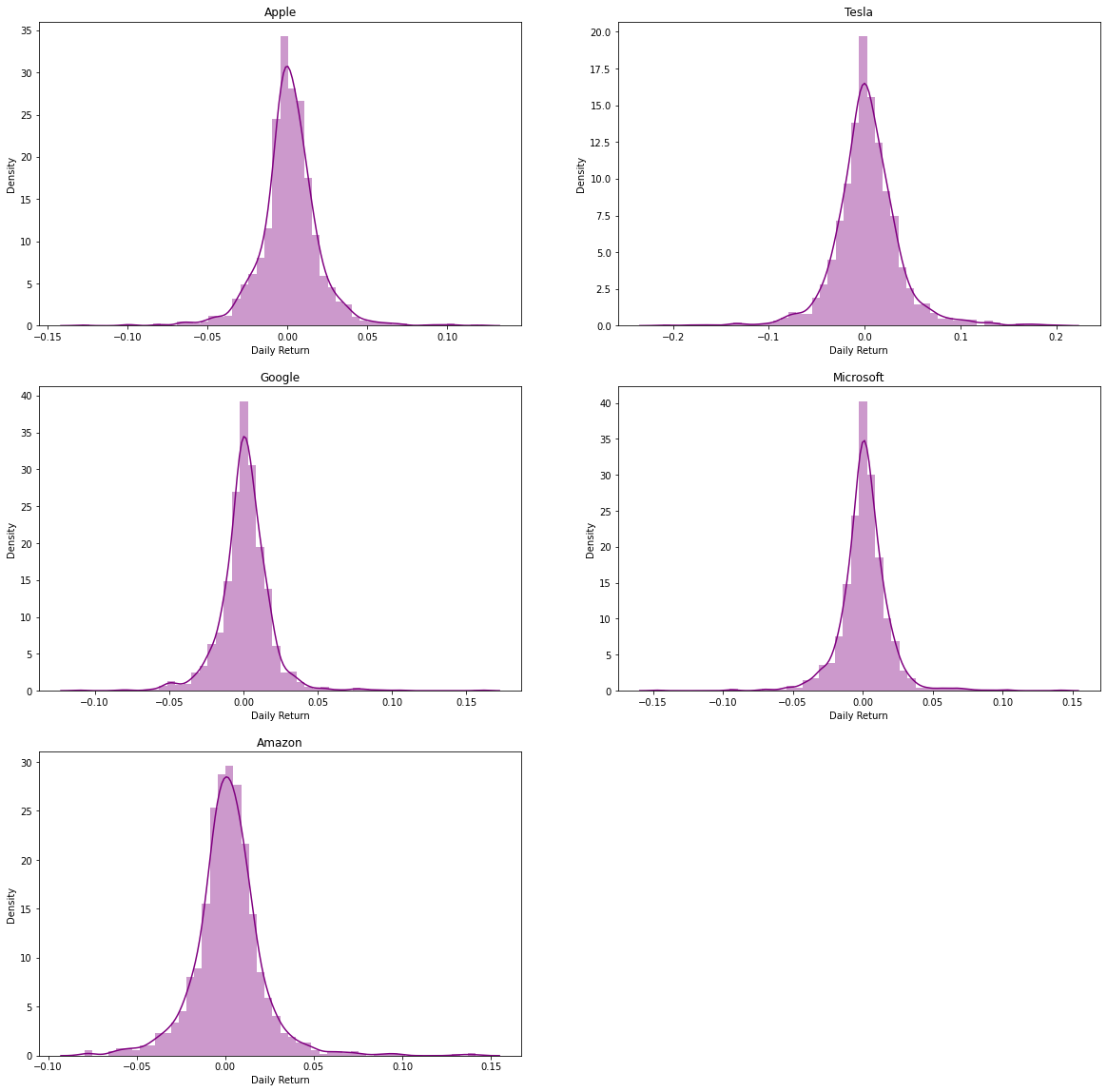
plt.figure(figsize=(20,20))

for index,company in enumerate(Companies):

plt.subplot(3,2,index+1)

sns.distplot(company['Daily Return'].dropna(),color='purple')

plt.title(Companies\_Title[index])



# In[27]:

print("Kurtosis value")

for index,company in enumerate(Companies):

print(f'{Companies\_Title[index]}: {company["Daily Return"].kurtosis()}')

**# ### What was the correlation between different stocks closing price?**

# In[28]:

Companies\_returns=pd.DataFrame()

Companies\_returns['AAPL']=AAPL['Adj. Close']

Companies\_returns['TSLA']=TSLA['Adj. Close']

Companies\_returns['MSFT']=MSFT['Adj. Close']

Companies\_returns['GOOG']=GOOG['Adj. Close']

Companies\_returns['AMZN']=AMZN['Adj. Close']

# In[29]:

Companies\_returns.head()

|  | **AAPL** | **TSLA** | **MSFT** | **GOOG** | **AMZN** |
| --- | --- | --- | --- | --- | --- |
| **0** | 24.86 | 43.86 | 41.44 | 523.37 | 335.04 |
| **1** | 24.16 | 42.02 | 41.06 | 512.46 | 333.57 |
| **2** | 24.16 | 42.26 | 40.46 | 500.59 | 338.64 |
| **3** | 24.50 | 42.19 | 40.97 | 499.73 | 326.00 |
| **4** | 25.44 | 42.12 | 42.18 | 501.30 | 326.31 |

# In[30]:

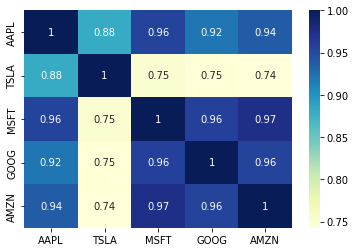
Companies\_Daily\_returns=Companies\_returns.pct\_change()

Companies\_Daily\_returns.head()

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **AAPL** | **TSLA** | **MSFT** | **GOOG** | **AMZN** |
| **0** | NaN | NaN | NaN | NaN | NaN |
| **1** | -0.028158 | -0.041952 | -0.009170 | -0.020846 | -0.004388 |
| **2** | 0.000000 | 0.005712 | -0.014613 | -0.023163 | 0.015199 |
| **3** | 0.014073 | -0.001656 | 0.012605 | -0.001718 | -0.037326 |
| **4** | 0.038367 | -0.001659 | 0.029534 | 0.003142 | 0.000951 |

# In[31]:

sns.heatmap(Companies\_returns.corr(),annot=True,cmap='YlGnBu')



# In[32]:

Companies\_returns.corr()

AAPL         TSLA         MSFT         GOOG         AMZN

AAPL       1.000000        0.881326        0.956366        0.919717        0.937317

TSLA         0.881326        1.000000        0.751496        0.745301        0.742543

MSFT         0.956366        0.751496        1.000000        0.957998        0.974242

GOOG         0.919717        0.745301        0.957998        1.000000        0.956704

AMZN        0.937317        0.742543        0.974242        0.956704        1.000000

*# In[33]:*

*Companies\_Daily\_returns.corr()*

AAPL TSLA MSFT GOOG AMZN

AAPL 1.000000 0.379490 0.682408 0.599042 -0.029361

TSLA 0.379490 1.000000 0.394718 0.349043 0.020527

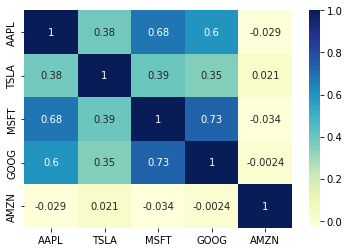
MSFT 0.682408 0.394718 1.000000 0.727382 -0.033594

GOOG 0.599042 0.349043 0.727382 1.000000 -0.002388

AMZN -0.029361 0.020527 -0.033594 -0.002388 1.000000

# In[34]:

sns.heatmap(Companies\_Daily\_returns.corr(),annot=True,cmap='YlGnBu')



# In[35]:

Return=Companies\_Daily\_returns.dropna()

plt.figure(figsize=(20,5))

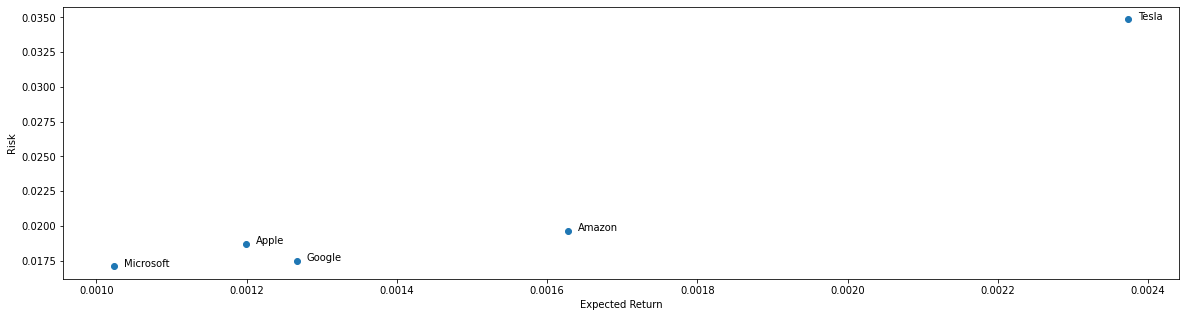
plt.scatter(Return.mean(),Return.std())

plt.xlabel('Expected Return')

plt.ylabel('Risk')

for label,x,y in zip(Companies\_Title,Return.mean(),Return.std()):

plt.annotate(label,xy=(x,y),xytext=(10,0),textcoords='offset points')



**Arima Model**

from statsmodels.tsa.stattools import adfuller

from statsmodels.tsa.arima\_model import ARIMA

from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error

import scipy.signal.signaltools

import warnings

warnings.filterwarnings('ignore')

# In[36]:

from statsmodels.tsa.stattools import adfuller

from statsmodels.tsa.arima\_model import ARIMA

from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error

import scipy.signal.signaltools

import warnings

warnings.filterwarnings('ignore')

# In[37]:

def \_centered(arr, newsize):

# Return the center newsize portion of the array.

newsize = np.asarray(newsize)

currsize = np.array(arr.shape)

startind = (currsize - newsize) // 2

endind = startind + newsize

myslice = [slice(startind[k], endind[k]) for k in range(len(endind))]

return arr[tuple(myslice)]

scipy.signal.signaltools.\_centered = \_centered

# In[38]:

from pmdarima.arima import auto\_arima

# In[39]:

Pre\_Processed\_AAPL=pd.read\_csv("Database/Pre\_Processed\_AAPL.csv")

Pre\_Processed\_TSLA=pd.read\_csv("Database/Pre\_Processed\_TSLA.csv")

Pre\_Processed\_GOOG=pd.read\_csv("Database/Pre\_Processed\_GOOG.csv")

Pre\_Processed\_MSFT=pd.read\_csv("Database/Pre\_Processed\_MSFT.csv")

Pre\_Processed\_AMZN=pd.read\_csv("Database/Pre\_Processed\_AMZN.csv")

# In[40]:

Pre\_Processed\_AAPL

Date Open High Low Close Adj. Close Volume

0 2015-01-02 27.85 27.86 26.84 27.33 24.86 212818400.0

1 2015-01-05 27.07 27.16 26.35 26.56 24.16 257142000.0

2 2015-01-06 26.64 26.86 26.16 26.57 24.16 263188400.0

3 2015-01-07 26.80 27.05 26.67 26.94 24.50 160423600.0

4 2015-01-08 27.31 28.04 27.17 27.97 25.44 237458000.0

... ... ... ... ... ... ... ...

1544 2021-02-22 128.01 129.72 125.60 126.00 126.00 103607600.0

1545 2021-02-23 123.76 126.71 118.39 125.86 125.86 157859100.0

1546 2021-02-24 124.94 125.56 122.23 125.35 125.35 110691500.0

1547 2021-02-25 124.68 126.46 120.54 120.99 120.99 147306500.0

1548 2021-02-26 122.59 124.85 121.20 121.26 121.26 164320000.0

# In[41]:

AAPL

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Date** | **Open** | **High** | **Low** | **Close** | **Adj. Close** | **Volume** | **Moving Average for 10 days** | **Moving Average for 20 days** | **Moving Average for 50 days** | **Moving Average for 100 days** | **Moving Average for 200 days** | **Moving Average for 500 days** | **Daily Return** |  |
| **0** | 2015-01-02 | 27.85 | 27.86 | 26.84 | 27.33 | 24.86 | 212818400.0 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **1** | 2015-01-05 | 27.07 | 27.16 | 26.35 | 26.56 | 24.16 | 257142000.0 | NaN | NaN | NaN | NaN | NaN | NaN | -0.028158 |
| **2** | 2015-01-06 | 26.64 | 26.86 | 26.16 | 26.57 | 24.16 | 263188400.0 | NaN | NaN | NaN | NaN | NaN | NaN | 0.000000 |
| **3** | 2015-01-07 | 26.80 | 27.05 | 26.67 | 26.94 | 24.50 | 160423600.0 | NaN | NaN | NaN | NaN | NaN | NaN | 0.014073 |
| **4** | 2015-01-08 | 27.31 | 28.04 | 27.17 | 27.97 | 25.44 | 237458000.0 | NaN | NaN | NaN | NaN | NaN | NaN | 0.038367 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **1544** | 2021-02-22 | 128.01 | 129.72 | 125.60 | 126.00 | 126.00 | 103607600.0 | 132.842 | 135.0495 | 131.7392 | 124.3926 | 111.53640 | 79.81000 | -0.029799 |
| **1545** | 2021-02-23 | 123.76 | 126.71 | 118.39 | 125.86 | 125.86 | 157859100.0 | 131.737 | 134.2070 | 131.8244 | 124.5140 | 111.78895 | 79.97692 | -0.001111 |
| **1546** | 2021-02-24 | 124.94 | 125.56 | 122.23 | 125.35 | 125.35 | 110691500.0 | 130.671 | 133.3270 | 131.8702 | 124.6131 | 112.03000 | 80.14192 | -0.004052 |
| **1547** | 2021-02-25 | 124.68 | 126.46 | 120.54 | 120.99 | 120.99 | 147306500.0 | 129.231 | 132.2840 | 131.8454 | 124.6589 | 112.24315 | 80.29778 | -0.034783 |
| **1548** | 2021-02-26 | 122.59 | 124.85 | 121.20 | 121.26 | 121.26 | 164320000.0 | 127.844 | 131.5025 | 131.8386 | 124.7449 | 112.46215 | 80.45434 | 0.002232 |

1549 rows × 14 columns

**Ad-fuller Test**

# In[42]:

def Test\_Stationarity(timeseries):

result=adfuller(timeseries['Adj. Close'],autolag='AIC')

print('Results of Dickey Fuller Test')

print(f'Test Statistics: {result[0]}')

print(f'p-value: {result[1]}')

print(f'Number of legs used: {result[2]}')

print(f'Number of Observations used: {result[3]}')

for key,value in result[4].items():

print(f'critical value ({key}):{value}')

*# In[43]:*

*Pre\_Processed\_AAPL.info()*

*<class 'pandas.core.frame.DataFrame'>*

*RangeIndex: 1549 entries, 0 to 1548*

*Data columns (total 7 columns):*

*# Column Non-Null Count Dtype*

*--- ------ -------------- -----*

*0 Date 1549 non-null object*

*1 Open 1549 non-null float64*

*2 High 1549 non-null float64*

*3 Low 1549 non-null float64*

*4 Close 1549 non-null float64*

*5 Adj. Close 1549 non-null float64*

*6 Volume 1549 non-null float64*

*dtypes: float64(6), object(1)*

*memory usage: 84.8+ KB*

# In[44]:

***Pre\_Processed\_AAPL.head()***

Date Open High Low Close Adj. Close Volume

0 2015-01-02 27.85 27.86 26.84 27.33 24.86 212818400.0

1 2015-01-05 27.07 27.16 26.35 26.56 24.16 257142000.0

2 2015-01-06 26.64 26.86 26.16 26.57 24.16 263188400.0

3 2015-01-07 26.80 27.05 26.67 26.94 24.50 160423600.0

4 2015-01-08 27.31 28.04 27.17 27.97 25.44 237458000.0

# In[45]:

***Pre\_Processed\_AAPL['Date']=pd.to\_datetime(Pre\_Processed\_AAPL["Date"])***

# In[46]:

***Pre\_Processed\_AAPL.head()***

Date Open High Low Close Adj. Close Volume

0 2015-01-02 27.85 27.86 26.84 27.33 24.86 212818400.0

1 2015-01-05 27.07 27.16 26.35 26.56 24.16 257142000.0

2 2015-01-06 26.64 26.86 26.16 26.57 24.16 263188400.0

3 2015-01-07 26.80 27.05 26.67 26.94 24.50 160423600.0

4 2015-01-08 27.31 28.04 27.17 27.97 25.44 237458000.0

*# In[47]:*

*Pre\_Processed\_AAPL*

*# In[48]:*

*Test\_Stationarity(Pre\_Processed\_AAPL)*

*Results of Dickey Fuller Test*

*Test Statistics: 1.1207222064433595*

*p-value: 0.9953896550774386*

*Number of legs used: 24*

*Number of Observations used: 1524*

*critical value (1%):-3.4346481290263013*

*critical value (5%):-2.8634383565975736*

*critical value (10%):-2.567780658251872*

*# In[49]:*

*Pre\_Processed\_AAPL['log Adj Close']=np.log(Pre\_Processed\_AAPL['Adj. Close'])*

*AAPL\_log\_moving\_avg=Pre\_Processed\_AAPL['log Adj Close'].rolling(12).mean()*

*AAPL\_log\_std=Pre\_Processed\_AAPL['log Adj Close'].rolling(12).std()*

*plt.figure(figsize=(10,5))*

*plt.plot(Pre\_Processed\_AAPL['Date'],AAPL\_log\_moving\_avg,label='Rolling Mean')*

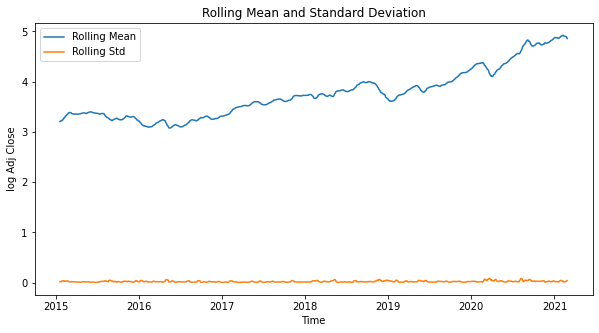
*plt.plot(Pre\_Processed\_AAPL['Date'],AAPL\_log\_std,label='Rolling Std')*

*plt.xlabel('Time')*

*plt.ylabel('log Adj Close')*

*plt.legend(loc='best')*

*plt.title('Rolling Mean and Standard Deviation')*



***# ### Split the data into training and test set***

***#***

***# Training Period: 2015-01-02 - 2020-09-30***

***#***

***# Testing Period: 2020-10-01 - 2021-02-26***

*# In[50]:*

*AAPL\_Train\_Data=Pre\_Processed\_AAPL[Pre\_Processed\_AAPL['Date']<'2020-10-01']*

*AAPL\_Test\_Data=Pre\_Processed\_AAPL[Pre\_Processed\_AAPL['Date']>='2020-10-01'].reset\_index(drop=True)*

*plt.figure(figsize=(10,5))*

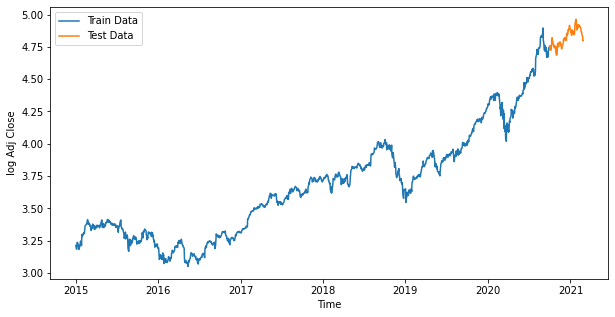
*plt.plot(AAPL\_Train\_Data['Date'],AAPL\_Train\_Data['log Adj Close'],label='Train Data')*

*plt.plot(AAPL\_Test\_Data['Date'],AAPL\_Test\_Data['log Adj Close'],label='Test Data')*

*plt.xlabel('Time')*

*plt.ylabel('log Adj Close')*

*plt.legend(loc='best')*



***# ## Arima Modeling***

*# In[51]:*

*AAPL\_Auto\_ARIMA\_Model=auto\_arima(AAPL\_Train\_Data['log Adj Close'],seasonal=False,*

*error\_action='ignore',suppress\_warnings=True)*

*print(AAPL\_Auto\_ARIMA\_Model.summary())*

*SARIMAX Results*

*==============================================================================*

*Dep. Variable: y No. Observations: 1447*

*Model: SARIMAX(1, 1, 0) Log Likelihood 3721.627*

*Date: Sat, 30 Apr 2022 AIC -7437.254*

*Time: 07:42:22 BIC -7421.424*

*Sample: 0 HQIC -7431.346*

*- 1447*

*Covariance Type: opg*

*==============================================================================*

*coef std err z P>|z| [0.025 0.975]*

*------------------------------------------------------------------------------*

*intercept 0.0012 0.000 2.366 0.018 0.000 0.002*

*ar.L1 -0.0981 0.015 -6.535 0.000 -0.127 -0.069*

*sigma2 0.0003 6.51e-06 52.314 0.000 0.000 0.000*

*===================================================================================*

*Ljung-Box (L1) (Q): 0.00 Jarque-Bera (JB): 2680.38*

*Prob(Q): 0.98 Prob(JB): 0.00*

*Heteroskedasticity (H): 2.25 Skew: -0.40*

*Prob(H) (two-sided): 0.00 Kurtosis: 9.62*

*===================================================================================*

*Warnings:*

*[1] Covariance matrix calculated using the outer product of gradients (complex-step).*

*# In[52]:*

*AAPL\_ARIMA\_Model=ARIMA(AAPL\_Train\_Data['log Adj Close'],order=(1,1,0))*

*AAPL\_ARIMA\_fit=AAPL\_ARIMA\_Model.fit()*

*print(AAPL\_ARIMA\_fit.summary())*

***ARIMA Model Results***

*==============================================================================*

*Dep. Variable: D.log Adj Close No. Observations: 1446*

*Model: ARIMA(1, 1, 0) Log Likelihood 3721.627*

*Method: css-mle S.D. of innovations 0.018*

*Date: Sat, 30 Apr 2022 AIC -7437.254*

*Time: 07:42:22 BIC -7421.424*

*Sample: 1 HQIC -7431.346*

*=========================================================================================*

*coef std err z P>|z| [0.025 0.975]*

*-----------------------------------------------------------------------------------------*

*const 0.0011 0.000 2.405 0.016 0.000 0.002*

*ar.L1.D.log Adj Close -0.0982 0.026 -3.748 0.000 -0.150 -0.047*

*Roots*

*=============================================================================*

*Real Imaginary Modulus Frequency*

*-----------------------------------------------------------------------------*

*AR.1 -10.1863 +0.0000j 10.1863 0.5000*

*-----------------------------------------------------------------------------*

*# In[53]:*

*AAPL\_Output=AAPL\_ARIMA\_fit.forecast(102,alpha=0.05)*

*AAPL\_predictions=np.exp(AAPL\_Output[0])*

*# In[54]:*

*AAPL\_predictions*

*# In[55]:*

*AAPL\_predictions.shape*

*# In[56]:*

*plt.figure(figsize=(10,5))*

*plt.plot(AAPL\_Train\_Data['Date'], AAPL\_Train\_Data['Adj. Close'], label = 'Training')*

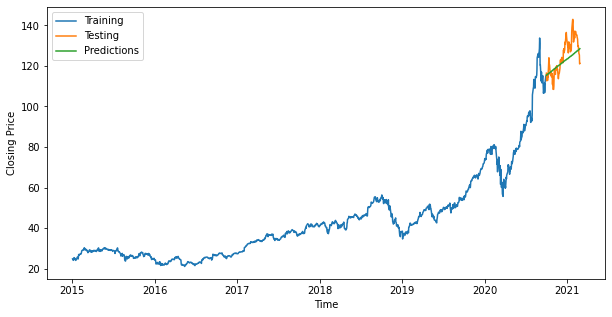
*plt.plot(AAPL\_Test\_Data['Date'], AAPL\_Test\_Data['Adj. Close'], label = 'Testing')*

*plt.plot(AAPL\_Test\_Data['Date'], AAPL\_predictions, label = 'Predictions')*

*plt.xlabel('Time')*

*plt.ylabel('Closing Price')*

*plt.legend()*



*# In[57]:*

*import math*

*rmse = math.sqrt(mean\_squared\_error(AAPL\_Test\_Data['Adj. Close'], AAPL\_predictions))*

*mape = np.mean(np.abs(AAPL\_predictions - AAPL\_Test\_Data['Adj. Close'])/np.abs(AAPL\_Test\_Data['Adj. Close']))*

*print(f'RMSE: {rmse}')*

*print(f'MAPE: {mape}')*

*# In[58]:*

*AAPL\_Test\_Data.shape*

*# In[59]:*

*AAPL\_Test\_Data['Adj. Close']*

*# In[60]:*

*AAPL['Adj. Close']*

***# ### Change Dtype of Date column***

*# In[61]:*

*Pre\_Processed\_AMZN['Date']=pd.to\_datetime(Pre\_Processed\_AAPL['Date'])*

*Pre\_Processed\_GOOG['Date']=pd.to\_datetime(Pre\_Processed\_GOOG['Date'])*

*Pre\_Processed\_MSFT['Date']=pd.to\_datetime(Pre\_Processed\_MSFT['Date'])*

*Pre\_Processed\_TSLA['Date']=pd.to\_datetime(Pre\_Processed\_TSLA['Date'])*

***# ## Dickey-Fuller Test or adfuller Test***

*# In[62]:*

*Test\_Stationarity(Pre\_Processed\_AMZN)*

*# In[63]:*

*Test\_Stationarity(Pre\_Processed\_GOOG)*

*# In[64]:*

*Test\_Stationarity(Pre\_Processed\_MSFT)*

*# In[65]:*

*Test\_Stationarity(Pre\_Processed\_TSLA)*

*# In[66]:*

*Pre\_Processed\_AMZN['log Adj Close']=np.log(Pre\_Processed\_AMZN['Adj. Close'])*

*AMZN\_log\_moving\_avg=Pre\_Processed\_AMZN['log Adj Close'].rolling(12).mean()*

*AMZN\_log\_std=Pre\_Processed\_AMZN['log Adj Close'].rolling(12).std()*

*plt.figure(figsize=(10,5))*

*plt.plot(Pre\_Processed\_AMZN['Date'],AMZN\_log\_moving\_avg,label='Rolling Mean')*

*plt.plot(Pre\_Processed\_AMZN['Date'],AMZN\_log\_std,label='Rolling Std')*

*plt.xlabel('Time')*

*plt.ylabel('log Adj Close')*

*plt.legend(loc='best')*

*plt.title('Rolling Mean and Standard Deviation')*

*# In[67]:*

*Pre\_Processed\_GOOG['log Adj Close']=np.log(Pre\_Processed\_GOOG['Adj. Close'])*

*GOOG\_log\_moving\_avg=Pre\_Processed\_GOOG['log Adj Close'].rolling(12).mean()*

*GOOG\_log\_std=Pre\_Processed\_GOOG['log Adj Close'].rolling(12).std()*

*plt.figure(figsize=(10,5))*

*plt.plot(Pre\_Processed\_GOOG['Date'],GOOG\_log\_moving\_avg,label='Rolling Mean')*

*plt.plot(Pre\_Processed\_GOOG['Date'],GOOG\_log\_std,label='Rolling Std')*

*plt.xlabel('Time')*

*plt.ylabel('log Adj Close')*

*plt.legend(loc='best')*

*plt.title('Rolling Mean and Standard Deviation')*

# In[68]:

*Pre\_Processed\_MSFT['log Adj Close']=np.log(Pre\_Processed\_MSFT['Adj. Close'])*

*MSFT\_log\_moving\_avg=Pre\_Processed\_MSFT['log Adj Close'].rolling(12).mean()*

*MSFT\_log\_std=Pre\_Processed\_MSFT['log Adj Close'].rolling(12).std()*

*plt.figure(figsize=(10,5))*

*plt.plot(Pre\_Processed\_MSFT['Date'],MSFT\_log\_moving\_avg,label='Rolling Mean')*

*plt.plot(Pre\_Processed\_MSFT['Date'],MSFT\_log\_std,label='Rolling Std')*

*plt.xlabel('Time')*

*plt.ylabel('log Adj Close')*

*plt.legend(loc='best')*

*plt.title('Rolling Mean and Standard Deviation')*

*# In[69]:*

*Pre\_Processed\_TSLA['log Adj Close']=np.log(Pre\_Processed\_TSLA['Adj. Close'])*

*TSLA\_log\_moving\_avg=Pre\_Processed\_TSLA['log Adj Close'].rolling(12).mean()*

*TSLA\_log\_std=Pre\_Processed\_TSLA['log Adj Close'].rolling(12).std()*

*plt.figure(figsize=(10,5))*

*plt.plot(Pre\_Processed\_TSLA['Date'],TSLA\_log\_moving\_avg,label='Rolling Mean')*

*plt.plot(Pre\_Processed\_TSLA['Date'],TSLA\_log\_std,label='Rolling Std')*

*plt.xlabel('Time')*

*plt.ylabel('log Adj Close')*

*plt.legend(loc='best')*

*plt.title('Rolling Mean and Standard Deviation')*

***# ### Split the data into training and test set***

***# Training Period: 2015-01-02 - 2020-09-30***

***#***

***# Testing Period: 2020-10-01 - 2021-02-26***

*# In[70]:*

*AMZN\_Train\_Data=Pre\_Processed\_AMZN[Pre\_Processed\_AMZN['Date']<'2020-10-01']*

*AMZN\_Test\_Data=Pre\_Processed\_AMZN[Pre\_Processed\_AMZN['Date']>='2020-10-01'].reset\_index(drop=True)*

*plt.figure(figsize=(10,5))*

*plt.plot(AMZN\_Train\_Data['Date'],AMZN\_Train\_Data['log Adj Close'],label='Train Data')*

*plt.plot(AMZN\_Test\_Data['Date'],AMZN\_Test\_Data['log Adj Close'],label='Test Data')*

*plt.xlabel('Time')*

*plt.ylabel('log Adj Close')*

*plt.legend(loc='best')*

*# In[71]:*

*GOOG\_Train\_Data=Pre\_Processed\_GOOG[Pre\_Processed\_GOOG['Date']<'2020-10-01']*

*GOOG\_Test\_Data=Pre\_Processed\_GOOG[Pre\_Processed\_GOOG['Date']>='2020-10-01'].reset\_index(drop=True)*

*plt.figure(figsize=(10,5))*

*plt.plot(GOOG\_Train\_Data['Date'],GOOG\_Train\_Data['log Adj Close'],label='Train Data')*

*plt.plot(GOOG\_Test\_Data['Date'],GOOG\_Test\_Data['log Adj Close'],label='Test Data')*

*plt.xlabel('Time')*

*plt.ylabel('log Adj Close')*

*plt.legend(loc='best')*

*# In[72]:*

*MSFT\_Train\_Data=Pre\_Processed\_MSFT[Pre\_Processed\_MSFT['Date']<'2020-10-01']*

*MSFT\_Test\_Data=Pre\_Processed\_MSFT[Pre\_Processed\_MSFT['Date']>='2020-10-01'].reset\_index(drop=True)*

*plt.figure(figsize=(10,5))*

*plt.plot(MSFT\_Train\_Data['Date'],MSFT\_Train\_Data['log Adj Close'],label='Train Data')*

*plt.plot(MSFT\_Test\_Data['Date'],MSFT\_Test\_Data['log Adj Close'],label='Test Data')*

*plt.xlabel('Time')*

*plt.ylabel('log Adj Close')*

*plt.legend(loc='best')*

*# In[73]:*

*TSLA\_Train\_Data=Pre\_Processed\_TSLA[Pre\_Processed\_TSLA['Date']<'2020-10-01']*

*TSLA\_Test\_Data=Pre\_Processed\_TSLA[Pre\_Processed\_TSLA['Date']>='2020-10-01'].reset\_index(drop=True)*

*plt.figure(figsize=(10,5))*

*plt.plot(TSLA\_Train\_Data['Date'],TSLA\_Train\_Data['log Adj Close'],label='Train Data')*

*plt.plot(TSLA\_Test\_Data['Date'],TSLA\_Test\_Data['log Adj Close'],label='Test Data')*

*plt.xlabel('Time')*

*plt.ylabel('log Adj Close')*

*plt.legend(loc='best')*

***# ## ARIMA Modeling***

*# In[74]:*

*AMZN\_Auto\_ARIMA\_Model=auto\_arima(AMZN\_Train\_Data['log Adj Close'],seasonal=False, error\_action='ignore',supress\_warnings=True)*

*print(AMZN\_Auto\_ARIMA\_Model.summary())*

*# In[75]:*

*AMZN\_ARIMA\_Model=ARIMA(AMZN\_Train\_Data['log Adj Close'],order=(1,1,0))*

*AMZN\_ARIMA\_Model\_fit=AMZN\_ARIMA\_Model.fit()*

*print(AMZN\_ARIMA\_Model\_fit.summary())*

*# In[76]:*

*MSFT\_Auto\_ARIMA\_Model=auto\_arima(MSFT\_Train\_Data['log Adj Close'],seasonal=False,*

*error\_action='ignore',supress\_warnings=True)*

*print(MSFT\_Auto\_ARIMA\_Model.summary())*

*# In[77]:*

*MSFT\_ARIMA\_Model=ARIMA(MSFT\_Train\_Data['log Adj Close'],order=(1,1,0))*

*MSFT\_ARIMA\_Model\_fit=MSFT\_ARIMA\_Model.fit()*

*print(MSFT\_ARIMA\_Model\_fit.summary())*

*# In[78]:*

*GOOG\_Auto\_ARIMA\_Model=auto\_arima(GOOG\_Train\_Data['log Adj Close'],seasonal=False,*

*error\_action='ignore',supress\_warnings=True)*

*print(GOOG\_Auto\_ARIMA\_Model.summary())*

*# In[79]:*

*GOOG\_ARIMA\_Model=ARIMA(GOOG\_Train\_Data['log Adj Close'],order=(1,1,0))*

*GOOG\_ARIMA\_Model\_fit=GOOG\_ARIMA\_Model.fit()*

*print(GOOG\_ARIMA\_Model\_fit.summary())*

*# In[80]:*

*TSLA\_Auto\_ARIMA\_Model=auto\_arima(TSLA\_Train\_Data['log Adj Close'],seasonal=False,*

*error\_action='ignore',supress\_warnings=True)*

*print(TSLA\_Auto\_ARIMA\_Model.summary())*

*# In[81]:*

*TSLA\_ARIMA\_Model=ARIMA(TSLA\_Train\_Data['log Adj Close'],order=(1,1,0))*

*TSLA\_ARIMA\_Model\_fit=TSLA\_ARIMA\_Model.fit()*

*print(TSLA\_ARIMA\_Model\_fit.summary())*

***# ## Predicting the closing stock price***

*# In[82]:*

*AMZN\_output=AMZN\_ARIMA\_Model\_fit.forecast(102,alpha=0.05)*

*AMZN\_predictions=np.exp(AMZN\_output[0])*

*plt.figure(figsize=(10,5))*

*plt.plot(AMZN\_Train\_Data['Date'],AMZN\_Train\_Data['Adj. Close'],label='Training Data')*

*plt.plot(AMZN\_Test\_Data['Date'],AMZN\_Test\_Data['Adj. Close'],label='Testing Data')*

*plt.plot(AMZN\_Test\_Data['Date'],AMZN\_predictions,label='Predictions')*

*plt.xlabel('Time')*

*plt.ylabel('Closing Price')*

*plt.legend()*

*# In[83]:*

*MSFT\_output=MSFT\_ARIMA\_Model\_fit.forecast(102,alpha=0.05)*

*MSFT\_predictions=np.exp(MSFT\_output[0])*

*plt.figure(figsize=(10,5))*

*plt.plot(MSFT\_Train\_Data['Date'],MSFT\_Train\_Data['Adj. Close'],label='Training Data')*

*plt.plot(MSFT\_Test\_Data['Date'],MSFT\_Test\_Data['Adj. Close'],label='Testing Data')*

*plt.plot(MSFT\_Test\_Data['Date'],MSFT\_predictions,label='Predictions')*

*plt.xlabel('Time')*

*plt.ylabel('Closing Price')*

*plt.legend()*

*# In[84]:*

*GOOG\_output=GOOG\_ARIMA\_Model\_fit.forecast(102,alpha=0.05)*

*GOOG\_predictions=np.exp(GOOG\_output[0])*

*plt.figure(figsize=(10,5))*

*plt.plot(GOOG\_Train\_Data['Date'],GOOG\_Train\_Data['Adj. Close'],label='Training Data')*

*plt.plot(GOOG\_Test\_Data['Date'],GOOG\_Test\_Data['Adj. Close'],label='Testing Data')*

*plt.plot(GOOG\_Test\_Data['Date'],GOOG\_predictions,label='Predictions')*

*plt.xlabel('Time')*

*plt.ylabel('Closing Price')*

*plt.legend()*

*# In[85]:*

*TSLA\_output=TSLA\_ARIMA\_Model\_fit.forecast(102,alpha=0.05)*

*TSLA\_predictions=np.exp(TSLA\_output[0])*

*plt.figure(figsize=(10,5))*

*plt.plot(TSLA\_Train\_Data['Date'],TSLA\_Train\_Data['Adj. Close'],label='Training Data')*

*plt.plot(TSLA\_Test\_Data['Date'],TSLA\_Test\_Data['Adj. Close'],label='Testing Data')*

*plt.plot(TSLA\_Test\_Data['Date'],TSLA\_predictions,label='Predictions')*

*plt.xlabel('Time')*

*plt.ylabel('Closing Price')*

*plt.legend()*

*# In[86]:*

*AMZN\_rmse=math.sqrt(mean\_squared\_error(AMZN\_Test\_Data['Adj. Close'],AMZN\_predictions))*

*AMZN\_mape=np.mean(np.abs(AMZN\_predictions-AMZN\_Test\_Data['Adj. Close'])/np.abs(AMZN\_Test\_Data['Adj. Close']))*

*print(f'AMZN\_rmse: {AMZN\_rmse}')*

*print(f'AMZN\_mape: {AMZN\_mape}')*

*# In[87]:*

*MSFT\_rmse=math.sqrt(mean\_squared\_error(MSFT\_Test\_Data['Adj. Close'],MSFT\_predictions))*

*MSFT\_mape=np.mean(np.abs(MSFT\_predictions-MSFT\_Test\_Data['Adj. Close'])/np.abs(MSFT\_Test\_Data['Adj. Close']))*

*print(f'MSFT\_rmse: {MSFT\_rmse}')*

*print(f'MSFT\_mape: {MSFT\_mape}')*

*# In[88]:*

*GOOG\_rmse=math.sqrt(mean\_squared\_error(GOOG\_Test\_Data['Adj. Close'],GOOG\_predictions))*

*GOOG\_mape=np.mean(np.abs(GOOG\_predictions-GOOG\_Test\_Data['Adj. Close'])/np.abs(GOOG\_Test\_Data['Adj. Close']))*

*print(f'GOOG\_rmse: {GOOG\_rmse}')*

*print(f'GOOG\_mape: {GOOG\_mape}')*

*# In[89]:*

*TSLA\_rmse=math.sqrt(mean\_squared\_error(TSLA\_Test\_Data['Adj. Close'],TSLA\_predictions))*

*TSLA\_mape=np.mean(np.abs(TSLA\_predictions-TSLA\_Test\_Data['Adj. Close'])/np.abs(TSLA\_Test\_Data['Adj. Close']))*

*print(f'TSLA\_rmse: {TSLA\_rmse}')*

*print(f'TSLA\_mape: {TSLA\_mape}')*

***Machine learning Model***

*# In[90]:*

*import tensorflow as tf*

*# In[91]:*

*def Dataset(Data,Date):*

*Train\_Data=Data['Adj. Close'][Data['Date']<Date].to\_numpy()*

*Data\_Train=[]*

*Data\_Train\_X=[]*

*Data\_Train\_Y=[]*

*for i in range(0,len(Train\_Data),5):*

*try:*

*Data\_Train.append(Train\_Data[i:i+5])*

*except:*

*pass*

*if len(Data\_Train[-1])<5:*

*Data\_Train.pop(-1)*

*Data\_Train\_X=Data\_Train[0:-1]*

*Data\_Train\_X=np.array(Data\_Train\_X)*

*Data\_Train\_X=Data\_Train\_X.reshape((-1,5,1))*

*Data\_Train\_Y=Data\_Train[1:len(Data\_Train)]*

*Data\_Train\_Y=np.array(Data\_Train\_Y)*

*Data\_Train\_Y=Data\_Train\_Y.reshape((-1,5,1))*

*Test\_Data=Data['Adj. Close'][Data['Date']>=Date].to\_numpy()*

*Data\_Test=[]*

*Data\_Test\_X= []*

*Data\_Test\_Y= []*

*for i in range (0, len(Test\_Data),5):*

*try:*

*Data\_Test.append(Test\_Data[i:i+5])*

*except:*

*pass*

*if len(Data\_Test[-1])<5:*

*Data\_Test.pop(-1)*

*Data\_Test\_X=Data\_Test[0:-1]*

*Data\_Test\_X=np.array(Data\_Test\_X)*

*Data\_Test\_X=Data\_Test\_X.reshape((-1,5,1))*

*Data\_Test\_Y=Data\_Test[1:len(Data\_Test)]*

*Data\_Test\_Y=np.array(Data\_Test\_Y)*

*Data\_Test\_Y=Data\_Test\_Y.reshape((-1,5,1))*

*return Data\_Train\_X,Data\_Train\_Y,Data\_Test\_X,Data\_Test\_Y*

***# # Deep Learning Model***

*# In[92]:*

*def Model():*

*model=tf.keras.models.Sequential([*

*tf.keras.layers.LSTM(200,input\_shape=(5,1),activation=tf.nn.leaky\_relu,return\_sequences=True),*

*tf.keras.layers.LSTM(200,activation=tf.nn.leaky\_relu),*

*tf.keras.layers.Dense(200,activation=tf.nn.leaky\_relu),*

*tf.keras.layers.Dense(100,activation=tf.nn.leaky\_relu),*

*tf.keras.layers.Dense(50,activation=tf.nn.leaky\_relu),*

*tf.keras.layers.Dense(5,activation=tf.nn.leaky\_relu)*

*])*

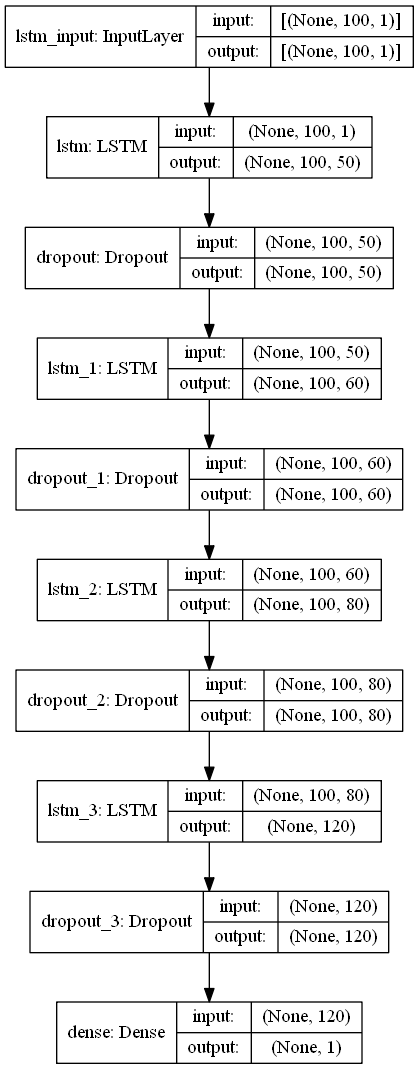
*return model*

*# In[93]:*

*model=Model()*

*# In[94]:*

*tf.keras.utils.plot\_model(model,show\_shapes=True)*



# In[95]:

*model.summary()*

*Model: "sequential"*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Layer (type) Output Shape Param #*

*=================================================================*

*lstm (LSTM) (None, 5, 200) 161600*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*lstm\_1 (LSTM) (None, 200) 320800*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*dense (Dense) (None, 200) 40200*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*dense\_1 (Dense) (None, 100) 20100*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*dense\_2 (Dense) (None, 50) 5050*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*dense\_3 (Dense) (None, 5) 255*

*=================================================================*

*Total params: 548,005*

*Trainable params: 548,005*

*Non-trainable params: 0*

***# ## Custom Learning Rate***

*# In[96]:*

*def scheduler(epoch):*

*if epoch <= 150:*

*lrate=(10 \*\* -5)\*(epoch / 150)*

*elif epoch <= 400:*

*initial\_lrate=(10 \*\* -5)*

*k=0.01*

*lrate=initial\_lrate \* math.exp(-k \* (epoch - 150))*

*else:*

*lrate=(10 \*\* -6)*

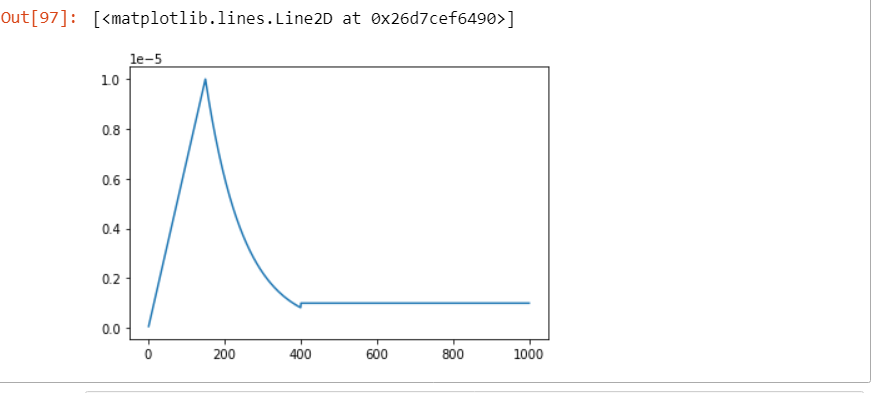
*return lrate*

*# In[97]:*

*epochs=[i for i in range(1,1001,1)]*

*lrate=[scheduler(i) for i in range(1,1001,1)]*

*plt.plot(epochs,lrate)*



# In[98]:

*callback=tf.keras.callbacks.LearningRateScheduler(scheduler)*

# In[99]:

*Pre\_Processed\_AAPL.head()*

*# In[100]:*

*Pre\_Processed\_AAPL.info()*

***# ### Split the Data into Training and Test set***

***# Training Period: 2015-01-02 - 2020-09-30***

***#***

***# Testing Period: 2020-10-01 - 2021-02-26***

*# In[101]:*

*AAPL\_Date='2020-10-01'*

*AAPL\_Train\_X,AAPL\_Train\_Y,AAPL\_Test\_X,AAPL\_Test\_Y=*

*Dataset(Pre\_Processed\_AAPL,AAPL\_Date)*

*# In[102]:*

*AMZN\_Date='2020-10-01'*

*AMZN\_Train\_X,AMZN\_Train\_Y,AMZN\_Test\_X,AMZN\_Test\_Y=Dataset(Pre\_Processed\_AMZN,AMZN\_Date)*

*# In[103]:*

*MSFT\_Date='2020-10-01'*

*MSFT\_Train\_X,MSFT\_Train\_Y,MSFT\_Test\_X,MSFT\_Test\_Y=Dataset(Pre\_Processed\_MSFT,MSFT\_Date)*

*# In[104]:*

*GOOG\_Date='2020-10-01'*

*GOOG\_Train\_X,GOOG\_Train\_Y,GOOG\_Test\_X,GOOG\_Test\_Y=Dataset(Pre\_Processed\_GOOG,GOOG\_Date)*

*# In[105]:*

*TSLA\_Date='2020-10-01'*

*TSLA\_Train\_X,TSLA\_Train\_Y,TSLA\_Test\_X,TSLA\_Test\_Y=Dataset(Pre\_Processed\_TSLA,TSLA\_Date)*

***# ## Model Fitting***

***# ### Apple***

***Model Fitting***

***Compile model***

*# In[106]:*

*AAPL\_Model=Model()*

*# In[107]:*

*AAPL\_Model.compile(optimizer=tf.keras.optimizers.Adam(),*

*loss='mse',*

*metrics=tf.keras.metrics.RootMeanSquaredError())*

*# In[108]:*

*AAPL\_hist=AAPL\_Model.fit(AAPL\_Train\_X,AAPL\_Train\_Y,*

*epochs=1000,*

*validation\_data=(AAPL\_Test\_X,AAPL\_Test\_Y),*

*callbacks=[callback])*

*Epoch 1/1000*

*9/9 [==============================] - 3s 77ms/step - loss: 2304.6030 - root\_mean\_squared\_error: 48.0063 - val\_loss: 15650.6953 - val\_root\_mean\_squared\_error: 125.1027*

*Epoch 2/1000*

*9/9 [==============================] - 0s 20ms/step - loss: 2304.5889 - root\_mean\_squared\_error: 48.0061 - val\_loss: 15650.4141 - val\_root\_mean\_squared\_error: 125.1016*

*Epoch 3/1000*

*9/9 [==============================] - 0s 20ms/step - loss: 2304.5439 - root\_mean\_squared\_error: 48.0057 - val\_loss: 15649.8486 - val\_root\_mean\_squared\_error: 125.0994*

*Epoch 4/1000*

*9/9 [==============================] - 0s 18ms/step - loss: 2304.4719 - root\_mean\_squared\_error: 48.0049 - val\_loss: 15648.9766 - val\_root\_mean\_squared\_error: 125.0959*

*Epoch 5/1000*

*9/9 [==============================] - 0s 20ms/step - loss: 2304.3760 - root\_mean\_squared\_error: 48.0039 - val\_loss: 15647.7432 - val\_root\_mean\_squared\_error: 125.0909*

*Epoch 6/1000*

*9/9 [==============================] - 0s 21ms/step - loss: 2304.2332 - root\_mean\_squared\_error: 48.0024 - val\_loss: 15646.3096 - val\_root\_mean\_squared\_error: 125.0852*

*Epoch 7/1000*

*9/9 [==============================] - 0s 18ms/step - loss: 2304.0652 - root\_mean\_squared\_error: 48.0007 - val\_loss: 15644.5264 - val\_root\_mean\_squared\_error: 125.0781*

*# In[109]:*

*AAPL\_history\_dict=AAPL\_hist.history*

*AAPL\_loss=AAPL\_history\_dict['loss']*

*AAPL\_root\_mean\_squared\_error=AAPL\_history\_dict['root\_mean\_squared\_error']*

*AAPL\_val\_loss=AAPL\_history\_dict['val\_loss']*

*AAPL\_val\_root\_mean\_squared\_error=AAPL\_history\_dict['val\_root\_mean\_squared\_error']*

*AAPL\_epochs=range(1,len(AAPL\_loss)+1)*

*# In[110]:*

*fig,(ax1,ax2)=plt.subplots(1,2)*

*fig.set\_figheight(5)*

*fig.set\_figwidth(15)*

*ax1.plot(AAPL\_epochs,AAPL\_loss,label='Training Loss')*

*ax1.plot(AAPL\_epochs,AAPL\_val\_loss,label='Validation Loss')*

*ax1.set(xlabel='Epochs',ylabel='Loss')*

*ax1.legend()*

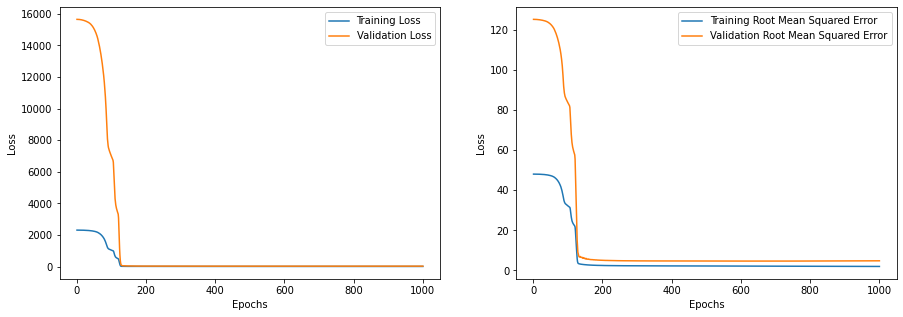
*ax2.plot(AAPL\_epochs,AAPL\_root\_mean\_squared\_error,label='Training Root Mean Squared Error')*

*ax2.plot(AAPL\_epochs,AAPL\_val\_root\_mean\_squared\_error,label='Validation Root Mean Squared Error')*

*ax2.set(xlabel='Epochs',ylabel='Loss')*

*ax2.legend()*

*plt.show()*



***# ## Amazon***

*# In[111]:*

*AMZN\_Model=Model()*

*# In[112]:*

*AMZN\_Model.compile(optimizer=tf.keras.optimizers.Adam(),loss='mse',metrics=tf.keras.metrics.RootMeanSquaredError())*

*# In[113]:*

*AMZN\_hist=AMZN\_Model.fit(AMZN\_Train\_X,AMZN\_Train\_Y,epochs=200,validation\_data=(AMZN\_Test\_X,AMZN\_Test\_Y),callbacks=[callback])*

*# In[114]:*

*AMZN\_history\_dict=AMZN\_hist.history*

*AMZN\_loss=AMZN\_history\_dict['loss']*

*AMZN\_root\_mean\_squared\_error=AMZN\_history\_dict['root\_mean\_squared\_error']*

*AMZN\_val\_loss=AMZN\_history\_dict['val\_loss']*

*AMZN\_val\_root\_mean\_squared\_error=AMZN\_history\_dict['val\_root\_mean\_squared\_error']*

*AMZN\_epochs=range(1,len(AMZN\_loss)+1)*

*# In[115]:*

*fig,(ax1,ax2)=plt.subplots(1,2)*

*fig.set\_figheight(5)*

*fig.set\_figwidth(15)*

*ax1.plot(AMZN\_epochs,AMZN\_loss,label='Training Loss')*

*ax1.plot(AMZN\_epochs,AMZN\_val\_loss,label='Validation Loss')*

*ax1.set(xlabel='Epochs',ylabel='Loss')*

*ax1.legend()*

*ax2.plot(AMZN\_epochs,AMZN\_root\_mean\_squared\_error,label='Training Root Mean Squared Error')*

*ax2.plot(AMZN\_epochs,AMZN\_val\_root\_mean\_squared\_error,label='Validation Root Mean Squared Error')*

*ax2.set(xlabel='Epochs',ylabel='Loss')*

*ax2.legend()*

*plt.show()*

***# ## Microsoft***

*# In[116]:*

*MSFT\_Model=Model()*

*# In[117]:*

*MSFT\_Model.compile(optimizer=tf.keras.optimizers.Adam(),loss='mse',metrics=tf.keras.metrics.RootMeanSquaredError())*

*# In[118]:*

*MSFT\_hist=MSFT\_Model.fit(MSFT\_Train\_X,MSFT\_Train\_Y,epochs=1000,validation\_data=(MSFT\_Test\_X,MSFT\_Test\_Y),callbacks=[callback])*

*# In[119]:*

*MSFT\_history\_dict=MSFT\_hist.history*

*MSFT\_loss=MSFT\_history\_dict['loss']*

*MSFT\_root\_mean\_squared\_error=MSFT\_history\_dict['root\_mean\_squared\_error']*

*MSFT\_val\_loss=MSFT\_history\_dict['val\_loss']*

*MSFT\_val\_root\_mean\_squared\_error=MSFT\_history\_dict['val\_root\_mean\_squared\_error']*

*MSFT\_epochs=range(1,len(MSFT\_loss)+1)*

*# In[120]:*

*fig,(ax1,ax2)=plt.subplots(1,2)*

*fig.set\_figheight(5)*

*fig.set\_figwidth(15)*

*ax1.plot(MSFT\_epochs,MSFT\_loss,label='Training Loss')*

*ax1.plot(MSFT\_epochs,MSFT\_val\_loss,label='Validation Loss')*

*ax1.set(xlabel='Epochs',ylabel='Loss')*

*ax1.legend()*

*ax2.plot(MSFT\_epochs,MSFT\_root\_mean\_squared\_error,label='Training Root Mean Squared Error')*

*ax2.plot(MSFT\_epochs,MSFT\_val\_root\_mean\_squared\_error,label='Validation Root Mean Squared Error')*

*ax2.set(xlabel='Epochs',ylabel='Loss')*

*ax2.legend()*

*plt.show()*

***# # GOOGle***

*# In[121]:*

*GOOG\_Model=Model()*

*# In[122]:*

*GOOG\_Model.compile(optimizer=tf.keras.optimizers.Adam(),loss='mse',metrics=tf.keras.metrics.RootMeanSquaredError())*

*# In[123]:*

*GOOG\_hist=GOOG\_Model.fit(GOOG\_Train\_X,GOOG\_Train\_Y,epochs=1000,validation\_data=(GOOG\_Test\_X,GOOG\_Test\_Y),callbacks=[callback])*

*# In[124]:*

*GOOG\_history\_dict=GOOG\_hist.history*

*GOOG\_loss=GOOG\_history\_dict['loss']*

*GOOG\_root\_mean\_squared\_error=GOOG\_history\_dict['root\_mean\_squared\_error']*

*GOOG\_val\_loss=GOOG\_history\_dict['val\_loss']*

*GOOG\_val\_root\_mean\_squared\_error=GOOG\_history\_dict['val\_root\_mean\_squared\_error']*

*GOOG\_epochs=range(1,len(GOOG\_loss)+1)*

*# In[125]:*

*fig,(ax1,ax2)=plt.subplots(1,2)*

*fig.set\_figheight(5)*

*fig.set\_figwidth(15)*

*ax1.plot(GOOG\_epochs,GOOG\_loss,label='Training Loss')*

*ax1.plot(GOOG\_epochs,GOOG\_val\_loss,label='Validation Loss')*

*ax1.set(xlabel='Epochs',ylabel='Loss')*

*ax1.legend()*

*ax2.plot(GOOG\_epochs,GOOG\_root\_mean\_squared\_error,label='Training Root Mean Squared Error')*

*ax2.plot(GOOG\_epochs,GOOG\_val\_root\_mean\_squared\_error,label='Validation Root Mean Squared Error')*

*ax2.set(xlabel='Epochs',ylabel='Loss')*

*ax2.legend()*

*plt.show()*

***# ## TeSLA***

*# In[126]:*

*TSLA\_Model=Model()*

*# In[127]:*

*TSLA\_Model.compile(optimizer=tf.keras.optimizers.Adam(),loss='mse',metrics=tf.keras.metrics.RootMeanSquaredError())*

*# In[128]:*

*TSLA\_hist=TSLA\_Model.fit(TSLA\_Train\_X,TSLA\_Train\_Y,epochs=200,validation\_data=(TSLA\_Test\_X,TSLA\_Test\_Y),callbacks=[callback])*

*# In[129]:*

*TSLA\_history\_dict=TSLA\_hist.history*

*TSLA\_loss=TSLA\_history\_dict['loss']*

*TSLA\_root\_mean\_squared\_error=TSLA\_history\_dict['root\_mean\_squared\_error']*

*TSLA\_val\_loss=TSLA\_history\_dict['val\_loss']*

*TSLA\_val\_root\_mean\_squared\_error=TSLA\_history\_dict['val\_root\_mean\_squared\_error']*

*TSLA\_epochs=range(1,len(TSLA\_loss)+1)*

*# In[130]:*

*fig,(ax1,ax2)=plt.subplots(1,2)*

*fig.set\_figheight(5)*

*fig.set\_figwidth(15)*

*ax1.plot(TSLA\_epochs,TSLA\_loss,label='Training Loss')*

*ax1.plot(TSLA\_epochs,TSLA\_val\_loss,label='Validation Loss')*

*ax1.set(xlabel='Epochs',ylabel='Loss')*

*ax1.legend()*

*ax2.plot(TSLA\_epochs,TSLA\_root\_mean\_squared\_error,label='Training Root Mean Squared Error')*

*ax2.plot(TSLA\_epochs,TSLA\_val\_root\_mean\_squared\_error,label='Validation Root Mean Squared Error')*

*ax2.set(xlabel='Epochs',ylabel='Loss')*

*ax2.legend()*

*plt.show()*

***# ## Prediction the Closing stock price***

*# In[131]:*

*AAPL\_Prediction=AAPL\_Model.predict(AAPL\_Test\_X)*

*# In[132]:*

*plt.figure(figsize=(10,5))*

*plt.plot(Pre\_Processed\_AAPL['Date'][Pre\_Processed\_AAPL['Date']<'2020-10-12'],*

*Pre\_Processed\_AAPL['Adj. Close'][Pre\_Processed\_AAPL['Date']<'2020-10-12'],*

*label='Training')*

*plt.plot(Pre\_Processed\_AAPL['Date'][Pre\_Processed\_AAPL['Date']>='2020-10-09'],*

*Pre\_Processed\_AAPL['Adj. Close'][Pre\_Processed\_AAPL['Date']>='2020-10-09'],*

*label='Testing')*

*plt.plot(Pre\_Processed\_AAPL['Date'][Pre\_Processed\_AAPL['Date']>='2020-10-12'],*

*AAPL\_Prediction.reshape(-1),*

*label='Predictions')*

*plt.xlabel('Time')*

*plt.ylabel('Closing Price')*

*plt.legend(loc='best')*

*# In[133]:*

*AAPL\_rmse=math.sqrt(mean\_squared\_error(AAPL\_Test\_Y.reshape(-1,5),AAPL\_Prediction))*

*AAPL\_mape=np.mean(np.abs(AAPL\_Prediction-AAPL\_Test\_Y.reshape(-1,5))/np.abs(AAPL\_Test\_Y.reshape(-1,5)))*

*print(f'RMSE: {AAPL\_rmse}')*

*print(f'RMSE: {AAPL\_mape}')*

*# In[134]:*

*AMZN\_Prediction=AMZN\_Model.predict(AMZN\_Test\_X)*

*# In[135]:*

*plt.figure(figsize=(10,5))*

*plt.plot(Pre\_Processed\_AMZN['Date'][Pre\_Processed\_AMZN['Date']<'2020-10-12'],*

*Pre\_Processed\_AMZN['Adj. Close'][Pre\_Processed\_AMZN['Date']<'2020-10-12'],*

*label='Training')*

*plt.plot(Pre\_Processed\_AMZN['Date'][Pre\_Processed\_AMZN['Date']>='2020-10-09'],*

*Pre\_Processed\_AMZN['Adj. Close'][Pre\_Processed\_AMZN['Date']>='2020-10-09'],*

*label='Testing')*

*plt.plot(Pre\_Processed\_AMZN['Date'][Pre\_Processed\_AMZN['Date']>='2020-10-12'],*

*AMZN\_Prediction.reshape(-1),*

*label='Predictions')*

*plt.xlabel('Time')*

*plt.ylabel('Closing Price')*

*plt.legend(loc='best')*

*# In[136]:*

*AMZN\_rmse=math.sqrt(mean\_squared\_error(AMZN\_Test\_Y.reshape(-1,5),AMZN\_Prediction))*

*AMZN\_mape=np.mean(np.abs(AMZN\_Prediction-AMZN\_Test\_Y.reshape(-1,5))/np.abs(AMZN\_Test\_Y.reshape(-1,5)))*

*print(f'RMSE: {AMZN\_rmse}')*

*print(f'RMSE: {AMZN\_mape}')*

*# In[137]:*

*GOOG\_Prediction=GOOG\_Model.predict(GOOG\_Test\_X)*

*# In[138]:*

*plt.figure(figsize=(10,5))*

*plt.plot(Pre\_Processed\_GOOG['Date'][Pre\_Processed\_GOOG['Date']<'2020-10-12'],*

*Pre\_Processed\_GOOG['Adj. Close'][Pre\_Processed\_GOOG['Date']<'2020-10-12'],*

*label='Training')*

*plt.plot(Pre\_Processed\_GOOG['Date'][Pre\_Processed\_GOOG['Date']>='2020-10-09'],*

*Pre\_Processed\_GOOG['Adj. Close'][Pre\_Processed\_GOOG['Date']>='2020-10-09'],*

*label='Testing')*

*plt.plot(Pre\_Processed\_GOOG['Date'][Pre\_Processed\_GOOG['Date']>='2020-10-12'],*

*GOOG\_Prediction.reshape(-1),*

*label='Predictions')*

*plt.xlabel('Time')*

*plt.ylabel('Closing Price')*

*plt.legend(loc='best')*

*# In[139]:*

*GOOG\_rmse=math.sqrt(mean\_squared\_error(GOOG\_Test\_Y.reshape(-1,5),GOOG\_Prediction))*

*GOOG\_mape=np.mean(np.abs(GOOG\_Prediction-GOOG\_Test\_Y.reshape(-1,5))/np.abs(GOOG\_Test\_Y.reshape(-1,5)))*

*print(f'RMSE: {GOOG\_rmse}')*

*print(f'RMSE: {GOOG\_mape}')*

*# In[140]:*

*MSFT\_Prediction=MSFT\_Model.predict(MSFT\_Test\_X)*

*# In[141]:*

*plt.figure(figsize=(10,5))*

*plt.plot(Pre\_Processed\_MSFT['Date'][Pre\_Processed\_MSFT['Date']<'2020-10-12'],*

*Pre\_Processed\_MSFT['Adj. Close'][Pre\_Processed\_MSFT['Date']<'2020-10-12'],*

*label='Training')*

*plt.plot(Pre\_Processed\_MSFT['Date'][Pre\_Processed\_MSFT['Date']>='2020-10-09'],*

*Pre\_Processed\_MSFT['Adj. Close'][Pre\_Processed\_MSFT['Date']>='2020-10-09'],*

*label='Testing')*

*plt.plot(Pre\_Processed\_MSFT['Date'][Pre\_Processed\_MSFT['Date']>='2020-10-12'],*

*MSFT\_Prediction.reshape(-1),*

*label='Predictions')*

*plt.xlabel('Time')*

*plt.ylabel('Closing Price')*

*plt.legend(loc='best')*

*# In[142]:*

*MSFT\_rmse=math.sqrt(mean\_squared\_error(MSFT\_Test\_Y.reshape(-1,5),MSFT\_Prediction))*

*MSFT\_mape=np.mean(np.abs(MSFT\_Prediction-MSFT\_Test\_Y.reshape(-1,5))/np.abs(MSFT\_Test\_Y.reshape(-1,5)))*

*print(f'RMSE: {GOOG\_rmse}')*

*print(f'RMSE: {GOOG\_mape}')*

*# In[143]:*

*TSLA\_Prediction=TSLA\_Model.predict(TSLA\_Test\_X)*

*# In[144]:*

*plt.figure(figsize=(10,5))*

*plt.plot(Pre\_Processed\_TSLA['Date'][Pre\_Processed\_TSLA['Date']<'2020-10-12'],*

*Pre\_Processed\_TSLA['Adj. Close'][Pre\_Processed\_TSLA['Date']<'2020-10-12'],*

*label='Training')*

*plt.plot(Pre\_Processed\_TSLA['Date'][Pre\_Processed\_TSLA['Date']>='2020-10-09'],*

*Pre\_Processed\_TSLA['Adj. Close'][Pre\_Processed\_TSLA['Date']>='2020-10-09'],*

*label='Testing')*

*plt.plot(Pre\_Processed\_TSLA['Date'][Pre\_Processed\_TSLA['Date']>='2020-10-12'],*

*TSLA\_Prediction.reshape(-1),*

*label='Predictions')*

*plt.xlabel('Time')*

*plt.ylabel('Closing Price')*

*plt.legend(loc='best')*

*# In[145]:*

*TSLA\_rmse=math.sqrt(mean\_squared\_error(TSLA\_Test\_Y.reshape(-1,5),TSLA\_Prediction))*

*TSLA\_mape=np.mean(np.abs(TSLA\_Prediction-TSLA\_Test\_Y.reshape(-1,5))/np.abs(TSLA\_Test\_Y.reshape(-1,5)))*

*print(f'RMSE: {TSLA\_rmse}')*

*print(f'RMSE: {TSLA\_mape}')*

***Save model in folder***

*# In[146 ]:*

*AAPL\_Model.save('karas\_model.h5')*

###### ***5.2.2 Webpage Code***

*from unicodedata import name*

*import streamlit as st*

*import yfinance as yf*

*import pandas as pd*

*import numpy as np*

*import cufflinks as cf*

*from prophet import Prophet*

*from datetime import datetime*

*import matplotlib.pyplot as plt*

*from keras.models import load\_model*

*import pandas\_datareader.data as web*

*from plotly import graph\_objects as go*

*from prophet.plot import plot\_plotly, plot\_components\_plotly*

*import datetime*

*#* ***App title***

*st.set\_page\_config(page\_title="Stock Prediction App")*

*st.markdown("""<style>#root>div:nth-child(1)>div>div>div>div>section>div {color:red}*

*</style>*

*""",unsafe\_allow\_html=True)*

*st.markdown('''*

*# Stock Price App*

*Shown are the stock price data for query companies!*

*\*\*Credits\*\**

*- Built in `Python` using `streamlit`,`yfinance`, `cufflinks`, `pandas` and `datetime`*

*''')*

*st.write('---')*

***# Sidebar***

*st.sidebar.subheader('Query parameters')*

*start\_date = st.sidebar.date\_input("Start date", datetime.date(2005, 1, 1))*

*end\_date = st.sidebar.date\_input("End date", datetime.date(2022, 4, 30))*

*n\_years=st.slider("years of predictions: ",1,4)*

*period=n\_years\*365*

***# Retrieving tickers data***

*ticker\_list = pd.read\_csv('https://raw.githubusercontent.com/dataprofessor/s-and-p-500-companies/master/data/constituents\_symbols.txt')*

*tickerSymbol = st.sidebar.selectbox('Stock ticker', ticker\_list)*

***# Select ticker symbol***

*tickerData = yf.Ticker(tickerSymbol) # Get ticker data*

*tickerDf = tickerData.history(period='1d', start=start\_date, end=end\_date) #get the historical prices for this ticker*

***# Ticker information***

*string\_logo = '<img src=%s>' % tickerData.info['logo\_url']*

*st.markdown(string\_logo, unsafe\_allow\_html=True)*

*string\_name = tickerData.info['longName']*

*st.header('\*\*%s\*\*' % string\_name)*

*string\_summary = tickerData.info['longBusinessSummary']*

*st.info(string\_summary)*

***# Ticker data***

*st.header('\*\*Ticker data\*\*')*

*st.write(tickerDf)*

*@st.cache*

*def load\_data(ticker):*

*data=web.DataReader(ticker,data\_source='yahoo',start=start\_date)*

*data.reset\_index(inplace=True)*

*return data*

*data=load\_data(tickerSymbol)*

***#describing data***

*st.subheader('Data from 2005 - Till Now')*

*st.write(data.describe())*

***# Time series Data***

*def plot\_row\_data():*

*fig=go.Figure(layout=go.Layout(height=600,width=900)) fig.add\_trace(go.Scatter(x=data['Date'],y=data['Open'],name='stock\_open'))*

*fig.add\_trace(go.Scatter(x=data['Date'],y=data['Close'],name='stock\_close'))*

*fig.layout.update(title\_text="Time Series Data",xaxis\_rangeslider\_visible=True)*

*st.plotly\_chart(fig)*

*plot\_row\_data()*

*st.subheader('Closing Price vs time chart 100MA ')*

*ma100=data.Close.rolling(100).mean()*

*fig=plt.figure(figsize=(12,6))*

*plt.plot(ma100)*

*plt.plot(data.Close)*

*plt.legend()*

*st.pyplot(fig)*

*st.subheader('Closing Price vs time chart 100MA & 200MA')*

*ma100=data.Close.rolling(100).mean()*

*ma200=data.Close.rolling(200).mean()*

*fig=plt.figure(figsize=(12,6))*

*plt.plot(ma100,'r')*

*plt.plot(ma200,'g')*

*plt.plot(data.Close,'b')*

*plt.legend()*

*st.pyplot(fig)*

***# Bollinger bands***

*st.header('\*\*Bollinger Bands\*\*')*

*qf=cf.QuantFig(tickerDf,title='First Quant Figure',legend='top',name='GS')*

*qf.add\_bollinger\_bands(periods=20,boll\_std=2,colors=['cyan','gray'],fill=True)*

*# qf.add\_volume(name='volume',up\_color='green',down\_color='red')*

*fig = qf.iplot(asFigure=True)*

*fig.layout.update(title\_text="Time Series Data",xaxis\_rangeslider\_visible=True)*

*st.plotly\_chart(fig)*

***#forecasting***

*df\_train=data[['Date','Close']]*

*df\_train=df\_train.rename(columns={"Date":"ds","Close":"y"})*

*m=Prophet()*

*m.fit(df\_train)*

*future=m.make\_future\_dataframe(periods=period)*

*forecast=m.predict(future)*

*st.subheader("Forecast data")*

*st.write(forecast.tail())*

*st.write("Forecast data")*

*fig1=plot\_plotly(m, forecast)*

*st.plotly\_chart(fig1)*

*st.write("Forecast components")*

*fig2=plot\_components\_plotly(m,forecast)*

*st.plotly\_chart(fig2)*

*data\_training=pd.DataFrame(data['Close'][0:int(len(data)\*0.70)])*

*data\_testing=pd.DataFrame(data['Close'][int(len(data)\*0.70):int(len(data))])*

*from sklearn.preprocessing import MinMaxScaler*

*scaler=MinMaxScaler(feature\_range=(0,1))*

*data\_training\_array=scaler.fit\_transform(data\_training)*

***#load model***

*model =load\_model('keras\_model.h5')*

***#testing part***

*past\_100\_days=data\_training.tail(100)*

*final\_df=past\_100\_days.append(data\_testing,ignore\_index=True)*

*input\_data=scaler.fit\_transform(final\_df)*

*x\_test=[]*

*y\_test=[]*

*for i in range(100,input\_data.shape[0]):*

*x\_test.append(input\_data[i-100:i])*

*y\_test.append(input\_data[i,0])*

*x\_test,y\_test=np.array(x\_test),np.array(y\_test)*

***# making predictions***

*y\_predicted=model.predict(x\_test)*

*scaler=scaler.scale\_*

*scaler\_factor=1/scaler[0]*

*y\_predicted=y\_predicted\*scaler\_factor*

*y\_test=y\_test\*scaler\_factor*

***#final graph***

*st.subheader('Predictions vs Original ')*

*fig2=plt.figure(figsize=(12,6))*

*plt.plot(y\_test,'b',label='Original Price')*

*plt.plot(y\_predicted,'r',label='Predicted Price')*

*plt.xlabel('Time')*

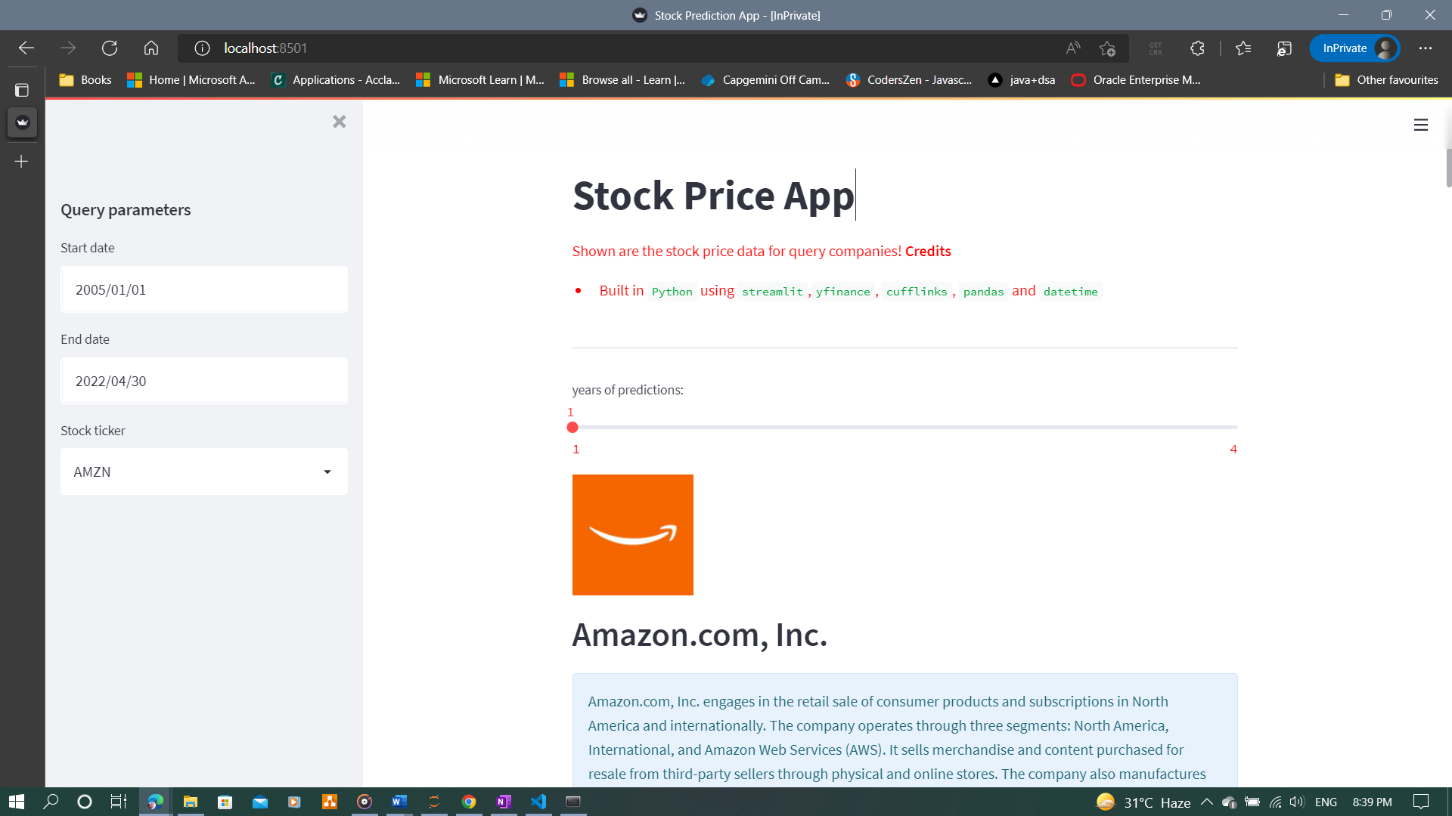
*plt.ylabel('Price')*

*plt.legend()*

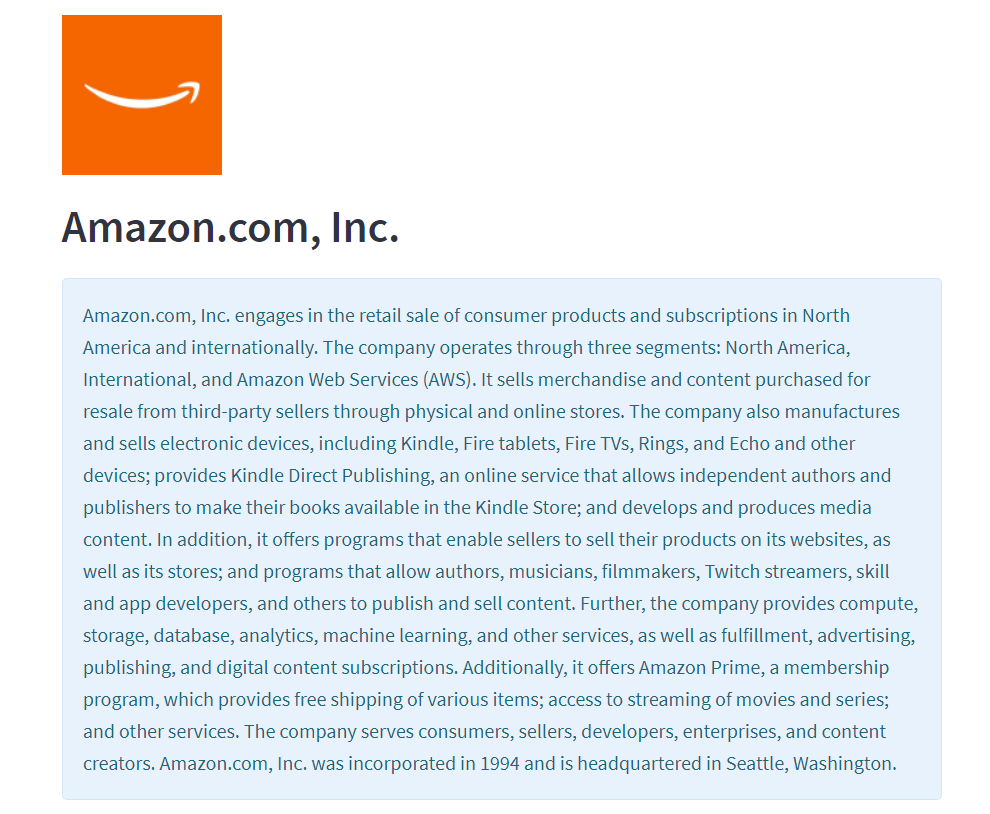
*st.pyplot(fig2)*

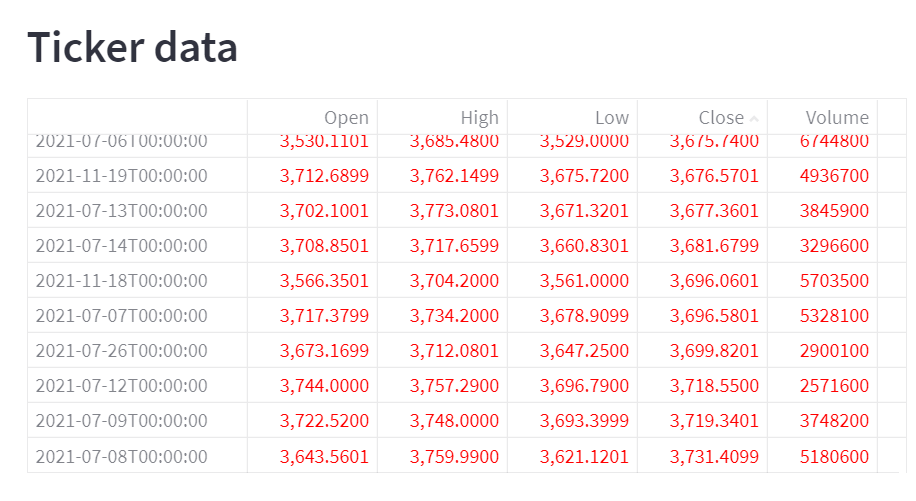
###### **5.2.2 Web Screen**

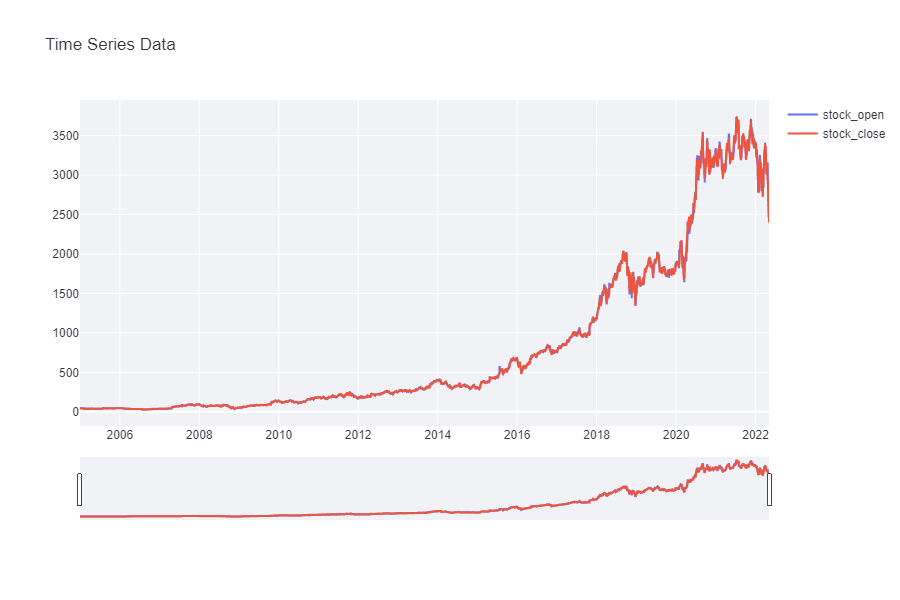
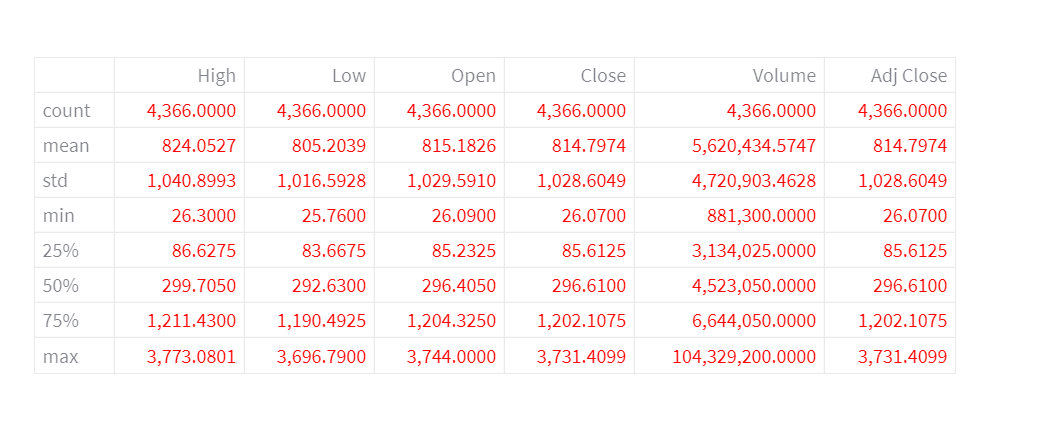
* Main Screen

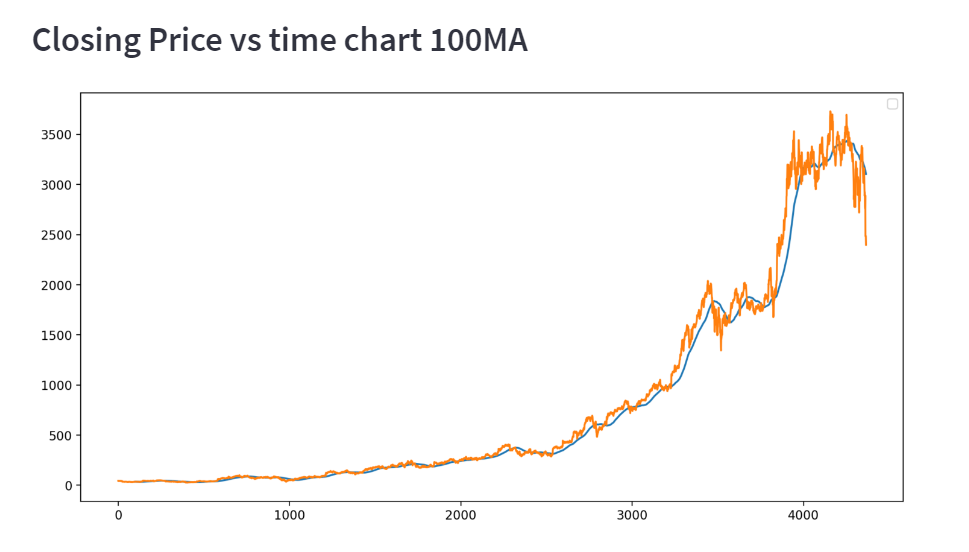


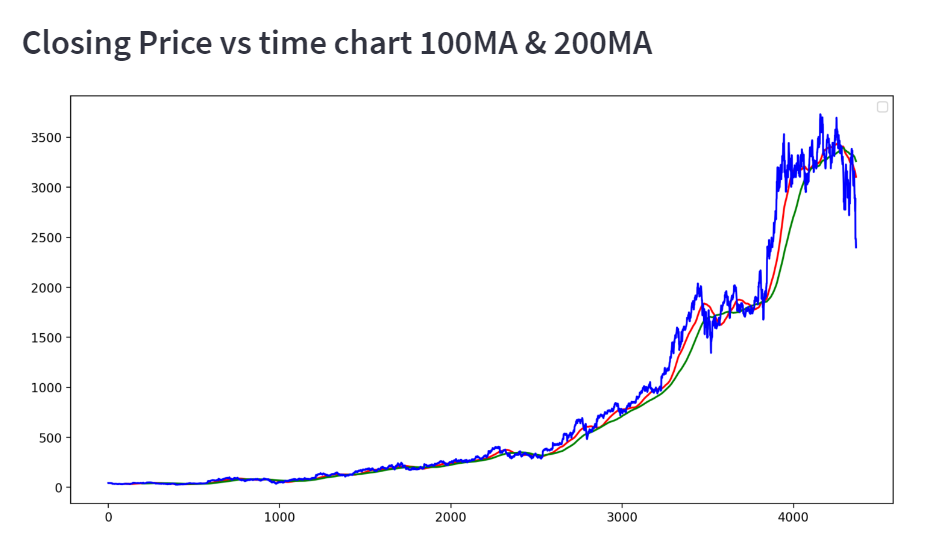
* Company Information

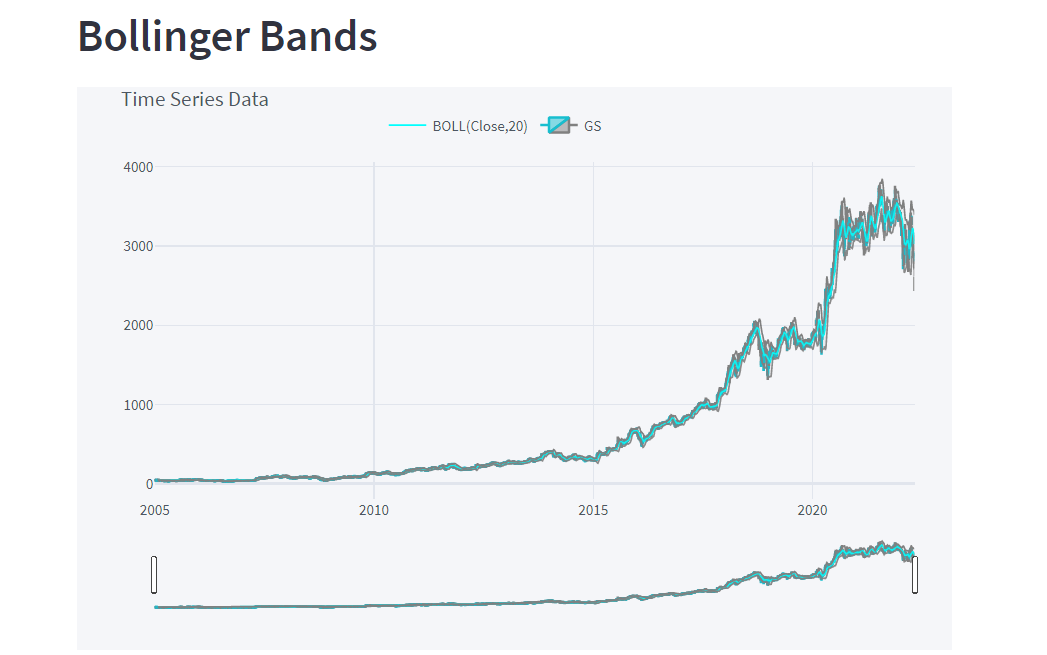




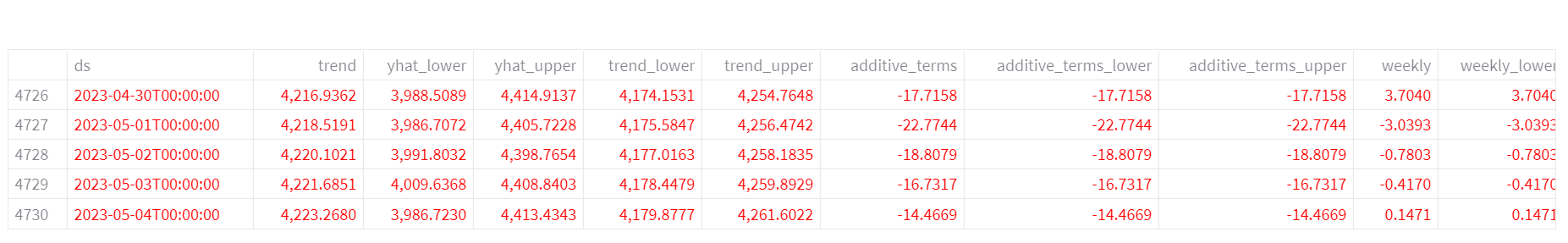


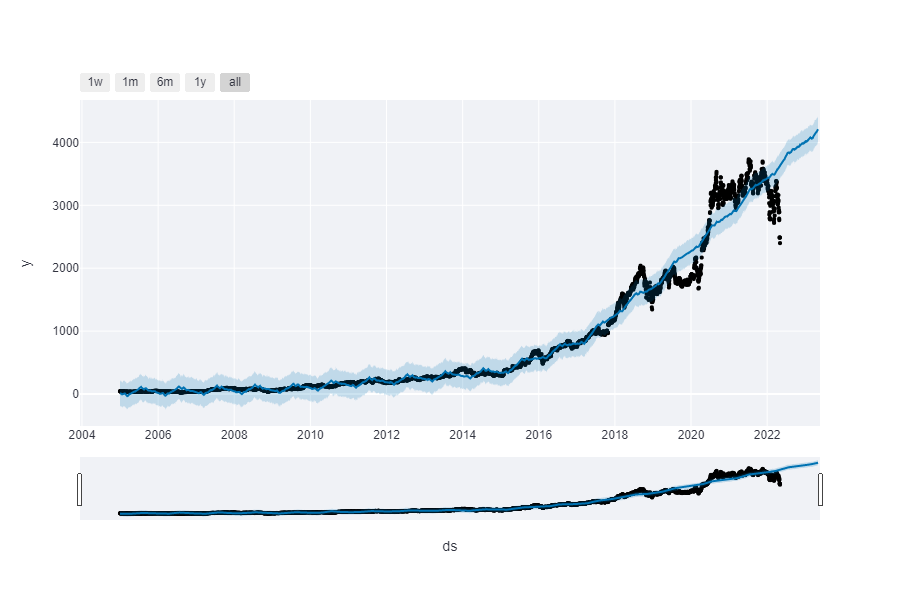




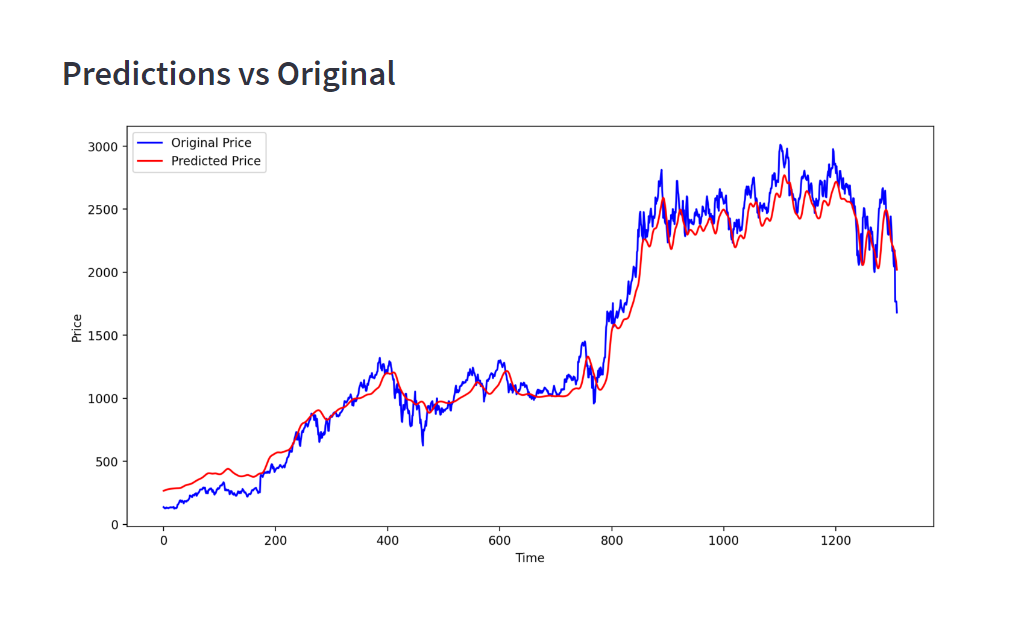


* Forecast Component





* Forecast Component



# Chapter 6 Conclusion And Future Work

### Conclusion

In this project, we are predicting closing stock price of any given organization, we developed a web application for predicting close stock price using LMS and LSTM algorithms for prediction. We have applied datasets belonging to Google, Nifty50, TCS, Infosys and Reliance Stocks and achieved above 95% accuracy for these datasets.

### Future work

* + - We want to extend this application for predicting cryptocurrency trading.
    - We want to add sentiment analysis for better analysis.

# Chapter 7 Bibliography

1. Stock Price Prediction Using LSTM on Indian Share Market by Achyut Ghosh, Soumik Bose1, Girdhar Maji, Narayan C. Debnath, Soumya Sen
2. S. Selvin, R. Vinay Kumar, E. A. Gopal Krishnan, V. K. Menon and K. P. Soman - Stock Price Prediction using LSTM, RNN and CNN-sliding window model - 2017.
3. Murtaza Roondiwala, Harshal Patel, Shraddha Varma, “Predicting Stock Prices Using LSTM” in Undergraduate Engineering Students, Department of Information Technology, Mumbai University, 2015.
4. Xiongwen Pang, Yanqiang Zhou, Pan Wang, Weiwei Lin, “An innovative neural network approach for stock market prediction”, 2018
5. Ishita Parmar, Navanshu Agarwal, Sheirsh Saxena, Ridam Arora, Shikhin Gupta, Himanshu Dhiman, Lokesh Chouhan Department of Computer Science and Engineering National Institute of Technology, Hamirpur – 177005, INDIA - Stock Market Prediction Using Machine Learning.
6. Pranav Bhat Electronics and Telecommunication Department, Maharashtra Institute of Technology, Pune. Savitribai Phule Pune University - A Machine Learning Model for Stock Market Prediction.
7. Anurag Sinha Department of computer science, Student, Amity University Jharkhand Ranchi, Jharkhand (India), 834001 - Stock Market Prediction Using Machine Learning.
8. V Kranthi Sai Reddy Student, ECM, Sreenidhi Institute of Science and Technology, Hyderabad, India

- Stock Market Prediction Using Machine Learning.

1. Asset Durmagambetov currently works at the mathematics, CNTFI- 'The Riemann Hypothesis- Millennium Prize Problems' - stock market predictions.
2. Mariam Moukalled Wassim El-Hajj Mohamad Jaber Computer Science Department American University of Beirut - Automated Stock Price Prediction Using Machine Learning.
3. Manh Ha Duong Boriss Siliverstovs June 2006 - The Stock Market and Investment.
4. Dharmaraja Selvamuthu, Vineet Kumar and Abhishek Mishra Department of Mathematics, Indian Institute of Technology Delhi, Hauz Khas, New Delhi 110016, India - Indian stock market prediction using artificial neural networks on tick data.
5. Lufuno Ronald Marwala A dissertation submitted to the Faculty of Engineering and the Built Environment, University of the Witwatersrand, Johannesburg, in fulfilment of the requirements for the degree of Master of Science in Engineering - Forecasting the Stock Market Index Using Artificial Intelligence Techniques.
6. Xiao Ding, Kuo Liao, Ting Liu, Zhongyang Li, Junwen Duan Research Centre for Social Computing and Information Retrieval Harbin Institute of Technology, China - Event Representation Learning Enhanced with External Common-sense Knowledge.
7. Huicheng Liu Department of Electrical and Computer Engineering Queen’s University, Canada - Leveraging Financial News for Stock Trend Prediction with Attention-Based Recurrent Neural Network.
8. Hyeong Kyu Choi, B.A Student Dept. of Business Administration Korea University Seoul, Korea = Stock Price Correlation Coefficient Prediction with ARIMA-LSTM Hybrid Model.
9. M. Nabipour Faculty of Mechanical Engineering, Tarbiat Modares University, 14115-143 Tehran, Iran[;Mojtaba.nabipour@modares.ac.ir](mailto:Mojtaba.nabipour@modares.ac.ir) - Deep Learning for Stock Market prediction.
10. Lavanya Ra SRM Institute of Science and Technology | SRM · Department of Computer Science - Stock Market Prediction.
11. M. Mekayel Anik · M. Shamsul Arefin (B) Department of Computer Science and Engineering, Chittagong University of Engineering and Technology, Chittagong, Bangladesh - An Intelligent Technique for Stock Market Prediction.
12. Osman Hegazy, Omar S. Soliman and Mustafa Abdul Salam Faculty of Computers and Informatics, Cairo University, Egypt - A Machine Learning Model for Stock Market Prediction
13. Sharanya Banerjee, Neha Dabeeru, R. Lavanya SRM Institute of Science and Technology, Chennai in Computer Science and Engineering - Stock Market Prediction
14. Fahad Almudhaf, Yaser A. Alkulaib from Kuwait University - Are Civets Stock Markets Predictable?
15. Mohammad Mekayel Anik, Mohammad Shamsul Arefin and M. Ali Akber Dewan, Department of Computer Science and Engineering - An Intelligent Technique for Stock Market Prediction
16. Asep Juarna, Departement of Informatics, Gunadarma University, Jakarta-Indonesia - ONE YEAR STOCK PRICE PREDICTION AND ITS VALIDITY USING LEAST SQUARE METHOD IN MATLAB
17. Giovanis, Eleftherios - Applications of Least Mean Square (LMS) Algorithm Regression in Time-Series Analysis
18. Vivek Rajput and Sarika Bobde - Stock market prediction using hybrid approach
19. James C. Van Horne and George G. C. Parker – The Random-walk Theory: An Empirical Test