# STOCK PRICE ANALYSIS

## B.TECH PROJECT


**Submitted by**

Rebaka Deepika Manaswi

Gunupuru Dheeraj Bhargav

Vurukuti Dheeraj

Donthu Gayathri Veni



**Guided by**
Prof. V. VALLI KUMARI

**Submitted to**


DEPARTMENT OF COMPUTER SCIENCE AND SYSTEMS ENGINEERING
ANDHRA UNIVERSITY COLLEGE OF ENGINEERING (A)
ANDHRA UNIVERSITY
VISAKHAPATNAM


May, 2022

**DEPARTMENT OF COMPUTER SCIENCE AND SYSTEMS ENGINEERING**
**ANDHRA UNIVERSITY COLLEGE OF ENGINEERING (A)**
**ANDHRA UNIVERSITY**
**VISAKHAPATNAM**

## CERTIFICATE

This is to certify that this is a bonafide project work entitled "**STOCK PRICE ANALYSIS"** done by **Rebaka Deepika Manaswi (Reg.No:318506402005**) **Gunupuru Dheeraj Bhargav (Reg.No:318506402006)**, **Vurukuti Dheeraj (Reg.No:318506402007)**, **Donthu Gayathri Veni (Reg.No:318506402008**) students of Department of Computer Science & Systems Engineering, Andhra University College of Engineering(A) during the period 2018-2022 in the partial fulfillment of the requirements for the award of degree of **Bachelor of Technology** in **Computer Science and Engineering**.

**Prof. KUNJAM NAGESWARA RAO**
**Head of the Department**

**Prof. V. VALLI KUMARI**
**Project guide**

# DECLARATION

We hereby declare that the project entitled **"STOCK PRICE ANALYSIS"** has been prepared by us during the period Feb 2022 – May 2022 in partial fulfillment of the requirements for the award of a degree of **Bachelor of Technology** in **Computer Science and Engineering**.

**Rebaka Deepika Manaswi (Reg.No- 318506402005)**

**Gunupuru Dheeraj Bhargav (Reg.No- 318506402006)**

**Vurukuti Dheeraj (Reg.No- 318506402007)**

**Donthu Gayathri Veni (Reg.No- 318506402008)**

**PLACE:** Visakhapatnam

**DATE:**

# ACKNOWLEDGEMENT

We would like to convey our heartfelt gratitude to **Prof. V. VALLI KUMARI**, our Project Guide, for her incredibly motivating nature and intellectual leadership. Despite her busy schedule, she has been gracious enough to spare her time and provide us with the required advice and assistance at every level of the planning and creation of this work. We convey our heartfelt gratitude for her permission to work on this project. and for graciously assisting us during the completion of this project.

We express sincere Thanks to **Prof. KUNJAM NAGESHWARA RAO**, Head of the Department of Computer Science and Systems Engineering, Andhra University College of Engineering(A) for his keen interest and for providing the necessary facilities for this project study.

We express sincere Thanks to **Prof. P. SRINIVAS RAO**, Principal, Computer Science and Systems Engineering, Andhra University College of Engineering(A) for his keen interest and for providing the necessary facilities for this project study.

We express sincere gratitude to **Prof. P.V.G.D. PRASAD REDDY**, Vice-Chancellor, Andhra University College of Engineering(A) for his keen interest and for providing necessary facilities for this project study.

We would like to thank our parents, friends, and classmates for their encouragement throughout our project period. At last but not the least, we thank everyone for supporting us directly or indirectly in completing this project successfully.

# TABLE OF CONTENTS


**ABSTRACT** **1**

**1. INTRODUCTION** **2-3**

1.1 Motivation for work 3

1.2 Problem Statement 3

**2. LITERATURE SURVEY** **4-11**

2.1 Stock Market Prediction Using Machine Learning 4

2.2 Forecasting the Stock Market Index Using AI Techniques 4

2.3 Indian stock market prediction using artificial neural networks on tick data 5

2.4 The Stock Market and Investment 6

2.5 Automated Stock Price Prediction Using Machine Learning 7

2.6 Stock Price Correlation Coefficient Prediction with ARIMA LSTM Hybrid Model 7

2.7 Event Representation Learning Enhanced with External Common-sense Knowledge 8

2.8 Forecasting directional movements of stock prices for intraday trading using LSTM and random forests 8

2.9 Deep Reinforcement Learning Library for Automated Stock Trading in Quantitative Finance 9

2.10 An innovative neural network approach for stock market prediction 10

2.11 An Intelligent Technique for Stock Market Prediction 11

**3. REQUIREMENTS ANALYSIS** **12-30**

3.1 Software Requirement Analysis 12

3.1.1 Functional Requirements 12

3.1.2 Non Functional Requirements 13

3.2 System Requirements 13

3.2.1 Software Requirements 13

3.2.2 Hardware Requirements 13

3.3 Technologies 14-16

3.3.1 Python 14

3.3.2 Numpy 15

3.3.3 Scikit Learn 15

3.3.4 Tensorflow 15
3.3.5 Keras 16
3.3.6 Compiler Option 16
3.4 Neural Networks 17-30
3.4.1 Recurrent Neural Networks 17
3.4.2 Long Short Term Memory 25

**4. SYSTEM ARCHITECTURE 31**
4.1 Preprocessing of Data 31
4.2 Overall Architecture 31

**5. DESIGN AND IMPLEMENTATION 32-50**
5.1 Proposed System 32
5.2 Uml Diagrams 33-38
5.2.1 Use Case Diagram 34
5.2.2 Sequence Diagram 35
5.2.3 Collaboration Diagram 36
5.2.4 Flow Chart 37
5.2.5 Component Diagram 38
5.3 Sample Code 39-50

**6. RESULTS 51-54**
6.1 Tesla Data Analysis 51
6.2 Microsoft Data Analysis 52
6.3 Google Data Analysis 53
6.4 Netflix Data Analysis 54

**7. CONCLUSION 55**

**8. REFERENCES 56-58**

# ABSTRACT

For many business analysts and academics, forecasting stock market values has always been a difficult undertaking. Indeed, stock market price forecasting is a fascinating field of study for investors. Many investors want to know about the market's future scenario in order to make profitable investments. Effective prediction systems assist traders indirectly by giving supportive information such as market direction in the future. By applying multiple algorithms to data, data mining techniques are excellent for projecting the future.

This project tries to increase the quality of output by anticipating stock market movements using financial news, analyst opinions, and quotes. It presents a revolutionary method for predicting the closing price of the stock market. Many researchers have contributed in some way to the field of chaotic forecasting. Fundamental and technical analyses are the traditional approaches so far. For stock market prediction, another common method for identifying unknown and hidden patterns in data is ANN.

We investigate the problem of stock market forecasting utilizing a Recurrent Neural Network (RNN) with Long Short-Term Memory (LSTM) in this project. The goal of this study is to look at the feasibility and performance of using LSTM to forecast stock market movements. We optimize the LSTM model by experimenting with various configurations, that is a multi-layered feed-forward neural network is built using a combination of data mining and machine learning. The Backpropagation Algorithm, which is used to predict share market closing prices, is utilized to train the Neural Network on stock quotes. Various out of sample performance measurements are used to compare the accuracy of the neural network's performance. This report will also cover Modeling techniques and the architecture of the Recurrent Neural Network .

# CHAPTER 1

# INTRODUCTION

Machine learning is a type of data analysis that automates the creation of analytical models. It is a subset of artificial intelligence predicated on the premise that computers can learn from data, recognise patterns, and make judgments with minimal human intervention.

Financial markets are a dynamic and aggregated system where anyone can buy and sell currencies, stocks, shares and derivatives on virtual platforms backed by brokers. The stock market allows investors to own shares of public companies through exchanges or over-the-counter purchases. This market has given investors the opportunity to make money and have a prosperous life by investing small initial amounts, low risk compared to the risk of opening a new business or business needs. high salary. The stock market is influenced by many factors that cause uncertainty and high market volatility. The stock market is essentially a collection of many different buyers and sellers of stocks. Stocks (also known as shares more commonly) typically represent ownership in the business of a particular individual or group of people. The attempt to determine the future value of the stock market is known as  stock market prediction. Predictions must be robust, accurate, and efficient. The system must behave according to real-life situations and must be well-adapted to real-world settings.

The main goal is to study and apply deep learning techniques to the stock market in order to predict stock behavior and, as a result, act on such predictions to reduce investment risk and make profit. The goal is to use transfer learning to take advantage of pre-built neural network models to achieve the goal. For short time periods, stock market price prediction looks to be a random process. Over a long period of time, the price of a stock usually forms a linear curve. People like to invest in equities that are predicted to rise in

value in the near future. People avoid investing in equities because of the stock market's instability.

Furthermore, in the case of time series forecasting, this project will cover a simple example of the most fitting model (the one that produces the best results), which is the LSTM model, which stands for Long Short Term Memory. Its efficacy is due to the addition of a vital component in time series predictions, the memory component, as compared to a typical deep neural network. In addition, this report will cover a more advanced example of LSTM models that use nested LSTM networks. We will also see and explain how the more complex model outperforms the simpler model and achieves promising outcomes.

## 1.1 MOTIVATION FOR WORK

The financial market piques the interest of many people. And they require direction and reliable forecasts in order to invest effectively. Investors are constantly on the lookout for reliable future results. Many applications attempt to forecast stock prices, but they do not provide detailed information on the forecast. We strive to acquire insight into market behavior over time using a successful stock prediction model.

## 1.2 PROBLEM STATEMENT

Every day, the stock market makes headlines. Every time it achieves a new high or low, you hear about it. If an effective algorithm could be established to anticipate the short term price of an individual stock, the rate of investment and business prospects in the stock market may increase. Artificial Neural Networks and Convolutional Neural Networks have been used in the past to predict stock prices, with an average error loss of 20%. In this report, we will investigate if a Recurrent Neural Network model may be used to predict stock price with a lower percentage of inaccuracy.

# CHAPTER 2

# LITERATURE SURVEY

The following papers were studied in order to get an overview of the techniques that were applied earlier to predict the stock market.

## 2.1 Stock Market Prediction Using Machine Learning

The act of attempting to anticipate the future value of a stock or other financial instrument traded on a financial exchange is known as stock market prediction. The prediction of a stock using Machine Learning is explained in this study. Most stockbrokers utilize technical and fundamental analysis, as well as time series analysis, when making stock predictions. Python is the programming language used to apply machine learning to predict the stock market. In this research, we present a Machine Learning (ML) approach that will be trained from the available stock data and gain intelligence and then use the acquired knowledge for an accurate prediction. In this context this study uses a machine learning technique called Support Vector Machine (SVM) to predict stock prices for the large and small capitalizations and in the three different markets, employing prices with both daily and up-to-the-minute frequencies.

## 2.2 Forecasting the Stock Market Index Using AI Techniques

The weak form of the Efficient Market Hypothesis (EMH) states that it is difficult to predict an asset's future price based on knowledge contained in historical prices. As a result, the market operates like a random walk, rendering forecasting impossible. Furthermore, due to the inherent complexity of the financial system, financial forecasting is a challenging undertaking. The goal of this project was to model and predict the future price of a stock market index using artificial intelligence (AI) techniques. In order to

estimate the future price of a stock market index based on previous price data, three artificial intelligence approaches are used: neural networks (NN), support vector machines (SVM), and neuro-fuzzy systems. Artificial intelligence approaches are employed as financial time series forecasting tools because they can account for financial system complexities. The AI algorithms are benchmarked using two techniques: Autoregressive Moving Average (ARMA), a linear modeling methodology, and random walk (RW), a random walk technique. The research was conducted using data from the Johannesburg Stock Exchange. The data used was a series of historical All Share Index closing prices. The results demonstrated that all three strategies can estimate the Index's future price with a reasonable degree of accuracy. The linear model was surpassed by all three artificial intelligence algorithms. The random walk method, on the other hand, outperformed all of the other methods. These strategies demonstrate the ability to predict future prices, but due to market transaction costs, it is not possible to demonstrate that the three techniques can contradict the weak version of market efficiency. The results reveal that the accuracy metric utilized affects the ranking of performances for vector machines, neuro-fuzzy systems, and multilayer perceptron neural networks.

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

A stock market is a marketplace where companies' stocks and derivatives can be traded at a set price. The stock market is driven by supply and demand for shares. The stock market is one of the most developing sectors in any country. Many people are now involved in this industry, either indirectly or directly. As a result, understanding market trends becomes critical. People are increasingly interested in stock price predictions as the stock market develops. However, because of its dynamic nature and vulnerability to rapid fluctuations in stock price, stock price prediction is a difficult assignment. Prior work has proposed effective methods to learn event representations that can capture syntactic and semantic information over text corpus, demonstrating their effectiveness for

downstream tasks such as script event prediction. On the other hand, events extracted from raw texts lack common-sense knowledge, such as the intentions and emotions of the event participants, which are useful for distinguishing event pairs when there are only subtle differences in their surface realizations. To address this issue, this paper proposes to leverage external common-sense knowledge about the intent and sentiment of the event. Markets are mostly a nonparametric, non-linear, noisy and deterministic chaotic system (Ahangar et al. 2010). As the technology is increasing, stock traders are moving towards using Intelligent Trading Systems rather than fundamental analysis for predicting prices of stocks, which helps them to make immediate investment decisions. One of the main aims of a trader is to predict the stock price such that he can sell it before its value declines, or buy the stock before the price rises. The efficient market hypothesis states that it is not possible to predict stock prices and that stock behaves in the random walk. It seems to be very difficult to replace the professionalism of an experienced trader for predicting the stock price. But because of the availability of a remarkable amount of data and technological advancements we can now formulate an appropriate algorithm for prediction whose results can increase the profits for traders or investment firms. Thus, the accuracy of an algorithm is directly proportional to gains made by using the algorithm.

## 2.4 The Stock Market and Investment

As European financial markets become more integrated, share prices in different European countries are likely to become even more correlated. If stock market movements influence actual economic components like investment and consumption, this process can lead to convergence in economic development across European countries. Indeed, our vector autoregressive models imply that there is a large positive association between changes in equity prices and investment. As a result, monetary authorities should

keep an eye on stock market reactions to monetary policy and their implications for the business cycle.

## 2.5 Automated Stock Price Prediction Using Machine Learning

Investors used to analyze stock prices and stock indicators, as well as news about these stocks, in order to predict market movement in the past. As a result, news has a significant impact on stock price fluctuation. The majority of past research in this field concentrated on either defining disclosed market news as (positive, negative, or neutral) and establishing their impact on the stock price, or on historical price movement and forecasting future movement. We present an automated trading system that blends mathematical functions, machine learning, and other external inputs such as news feelings to improve stock forecast accuracy and issue lucrative trades in this paper. We want to know the price or trend of a particular stock for the coming end-of-day by looking at the first few trading hours of the day. We used classic machine learning algorithms and created/trained numerous deep learning models to achieve this goal, taking into account the value of the relevant news. Several experiments were carried out, with the maximum accuracy (82.91%) attained using SVM for Apple Inc. (AAPL) stock.

## 2.6 Stock Price Correlation Coefficient Prediction with ARIMA LSTM Hybrid Model

In portfolio optimization, predicting the price correlation of two assets for future time periods is critical. The stock price correlation coefficient of two separate stocks is predicted using LSTM recurrent neural networks (RNN). RNNs are capable of recognising temporal dependencies. The use of LSTM cells improves its long-term prediction abilities even further. We use the ARIMA model to account for both linearity and nonlinearity in the model. The ARIMA model filters data for linear tendencies and sends the residual value to the LSTM model. Other traditional predictive financial models

such as the whole historical model, constant correlation model, single-index model, and multi-group model are compared to the ARIMA-LSTM hybrid model. The ARIMA-LSTM model's forecasting capacity outperformed all other financial models by a significant margin in our empirical investigation. Our findings suggest that the ARIMA LSTM model should be considered for forecasting correlation coefficients for portfolio optimization.

## 2.7 Event Representation Learning Enhanced with External Common-sense Knowledge

Prior work has proposed effective methods to learn event representations that can capture syntactic and semantic information over text corpus, demonstrating their effectiveness for downstream tasks such as script event prediction. On the other hand, events extracted from raw texts lack common-sense knowledge, such as the intentions and emotions of the event participants, which are useful for distinguishing event pairs when there are only subtle differences in their surface realizations. This work attempts to use external common-sense knowledge about the event's intent and sentiment to address this problem. Experiments on three event-related tasks, namely event similarity, script event prediction, and stock market prediction, show that our model obtains much better event embeddings for the tasks, with 78% improvements on the hard similarity task, yielding more precise inferences on subsequent events under given contexts, and better accuracies in predicting stock market volatilities.

## 2.8 Forecasting directional movements of stock prices for intraday trading using LSTM and random forests

We employ both random forests and LSTM networks (more precisely CuDNNLSTM) as training methodologies to analyze their effectiveness in forecasting out-of-sample directional movements of constituent stocks of the S&P 500 from January 1993 till

December 2018 for intraday trading. We introduce a multi-feature option that includes returns not just relative to closing prices, but also relative to opening prices and intraday returns. We utilize Krauss et al. (2017) and Fischer & Krauss (2018) as a benchmark and buy the 10 stocks with the highest probability and short sell the 10 stocks with the lowest probability to exceed the market in terms of intraday returns – all with equal monetary weight – on each trading day. Our empirical findings reveal that, before transaction costs, the multi-feature option gives a daily return of 0.64 percent for LSTM networks and 0.54 percent for random forests. As a result, we surpass Fischer & Krauss (2018) and Krauss et al. (2017)'s single feature setup consisting simply of daily returns with respect to closing prices, with comparable daily returns of 0.41 percent and 0.39 percent for LSTM and random forests, respectively. 1 Random forest, LSTM, Forecasting, Statistical Arbitrage, Machine learning, Intraday trading are some of the terms used.

## 2.9 Deep Reinforcement Learning Library for Automated Stock Trading in Quantitative Finance

Getting hands-on experience is appealing to beginners since deep reinforcement learning (DRL) has been recognised as an effective strategy in quantitative finance. However, developing and debugging a practical DRL trading agent that selects where to trade, at what price, and in what quantity is error-prone and time-consuming. In this paper, we introduce FinRL, a DRL library that helps beginners learn about quantitative finance and design their own stock trading methods. The FinRL library provides users with easy-to-follow tutorials as well as the ability to simplify their own developments and compare them to current schemes. FinRL uses stock market datasets to build virtual environments, neural networks to train trading agents, and comprehensive back testing to analyze trading performance. It also takes into account significant trading constraints including transaction costs, market liquidity, and the risk aversion of the investor. FinRL's completeness, hands-on teaching, and reproducibility make it ideal for beginners: (i) at multiple levels of time granularity, FinRL simulated trading environments across various

stock markets, including NASDAQ-100, DJIA, S&P 500, HSI, SSE 50, and CSI 300; (ii) organized in a layered architecture with modular structure, FinRL provides finetuned state-of-the-art DRL algorithms (DQN, DDPG, PPO, SAC, A2C, TD3, etc.), commonly used reward functions and standard evaluation baselines to alleviate the debugging workloads and promote the reproducibility, and (iii) being highly extendable, FinRL reserves a complete set of user-import interfaces. Furthermore, We also included three application demos, including single stock trading, multiple stock trading, and portfolio allocation.

## 2.10 An innovative neural network approach for stock market prediction

To develop an innovative neural network approach to achieve better stock market predictions. To illustrate the Internet of Multimedia of Things for stock analysis, data was acquired from the live stock market for real-time and off-line analysis, as well as the outcomes of visualizations and analyses. Traditional neural network algorithms may inaccurately anticipate the stock market while studying the influence of market variables on stock prices, because the initial weight of the random selection issue is easily prone to false predictions. We show the notion of "stock vector" based on the growth of word vectors in deep learning. The input is multi-stock high-dimensional historical data, rather than a single index or single stock index. To anticipate the stock market, we propose the deep long short-term memory neural network (LSTM) with embedded layers and the long short-term memory neural network with automatic encoder. We vectorize the data in these two models using the embedded layer and the automatic encoder, respectively, in order to forecast the stock using a long short-term memory neural network. The deep LSTM with embedded layers performs better in the experiments. For the Shanghai A-shares composite index, the accuracy of two models is 57.2 and 56.9 percent, respectively. Furthermore, for individual equities, they are 52.4 and 52.5 percent,

respectively. We present research contributions in IMMT for financial analysis using neural networks.

## 2.11 An Intelligent Technique for Stock Market Prediction

A stock market is a loose network of economic transactions centered on stocks, also known as shares, between buyers and sellers. Stocks reflect ownership claims on enterprises in stock markets. These can include securities that are publicly traded as well as those that are exclusively traded privately. A stock exchange is a market where brokers can purchase and sell stocks, bonds, and other financial instruments. Because of its volatility, the stock market is a risky area to invest in. We had major financial troubles in the recent past as a result of the massive collapse in stock market prices around the world. This occurrence had a significant impact on both the international and domestic financial systems. On the stock market, many people lost their last savings. The Bangladeshi stock market crashed dramatically in the 2010–2011 financial year. This phenomenon can be brought under control especially by strict monitoring and stock market analysis. If we can accurately analyze the stock market at the right time, it can become a lucrative sector for investors and make them less vulnerable. The stock market is all about generating predictions and making quick investment decisions, which requires detailed market analysis. If we can accurately predict the stock market using historical data, we can avoid the effects of a severe market crash and take the necessary actions to make the market immune to such events.

# CHAPTER 3

# REQUIREMENTS ANALYSIS

## 3.1 SOFTWARE REQUIREMENT SPECIFICATIONS

### 3.1.1 FUNCTIONAL REQUIREMENTS

The software's functional requirements outline what it should be able to perform (the functions). Consider the fundamental operations. Functional requirements should be written in the future tense because "functions" are developed before development. Some of the functional requirements for designing Stock Price Prediction software could include:

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

### 3.1.2 NON FUNCTIONAL REQUIREMENTS

**Product properties:**

- **Usability:** It specifies the software's user interface in terms of how easy it is to understand the user interface of stock prediction software for any type of stock trader and other stock market stakeholders.
- **Efficiency:** Maintaining the highest possible accuracy in closing stock prices in the shortest time possible with the data available.
- **Performance:** It is a quality attribute of the stock prediction software that describes how responsive it is to different user interactions.

## 3.2 SYSTEM REQUIREMENTS

### 3.2.1 SOFTWARE REQUIREMENTS

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

### 3.2.2 HARDWARE REQUIREMENTS

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

## 3.3 TECHNOLOGIES

### 3.3.1 PYTHON

Python was the language of choice for this project. This was an easy decision for multiple reasons.

- Python as a programming language has a sizable user base. A trip to Stack Overflow can easily fix any problems that arise. Python is one of the most popular languages on the site, so there's a good chance you'll get a direct answer to your question.
- Python comes with a variety of scientifically sound computing capabilities. Numpy, Pandas, and SciPy are examples of free and well-documented Python packages. Packages like this can drastically decrease and simplify the amount of code required to develop an application. This allows for speedy iteration.
- Python is a forgiving language that permits pseudocode to be used in programming. When pseudocode from academic publications needs to be implemented and checked, this is useful. This step is usually rather simple when using Python.

Python, on the other hand, is not without flaws. The language is dynamically typed, and duck typing is common among packages. This can be aggravating when a package function produces something that looks like an array but isn't actually an array. When combined with the fact that standard Python documentation does not explicitly identify a method's return type, this can result in a lot of trial and error testing that would not occur in a highly typed language. This is a problem that makes learning a new Python package or library more difficult than it would otherwise be.

### 3.3.2 NUMPY

Numpy is a collection of Python modules that enable scientific and higher-level mathematical abstractions. We can't employ mathematical abstractions like f(x) in most programming languages since they would change the code's semantics and syntax. However, we may use Numpy to make use of such functions in our code. Numpy's array type provides an efficient data structure for numerical computations, such as manipulating matrices, in the Python language.

### 3.3.3 SCIKIT LEARN

Scikit-learn is a Python-based machine learning library that is available for free. Support vector machine, random forest, gradient boosting, k-means, and other classification, regression, and clustering algorithms are included. It's primarily intended to work with Python's NumPy and SciPy numerical and scientific libraries. Scikit-learn is primarily written in Python, with certain performance-oriented core algorithms written in Cython. Support vector machines are built using a Cython wrapper around LIBSVM, similar to how logistic regression and linear support vector machines are implemented using LIBLINEAR.

### 3.3.4 TENSORFLOW

TensorFlow is an open source software library that uses data flow graphs to do numerical computations. The graph's nodes represent mathematical operations, while the graph's edges represent the multidimensional data arrays (tensors) that are exchanged between them. With the flexible architecture, you may use a single API to deploy compute to one or more CPUs or GPUs in a desktop, server, or mobile device. TensorFlow was created by Google's Machine Intelligence research group for the aim of conducting machine learning and deep neural network research, although the technology is broad enough to be used in a variety of different disciplines.

TensorFlow is Google Brain's second-generation system. While the reference implementation runs on single devices, TensorFlow can run on multiple CPUs and GPUs (with optional CUDA and SYCL extensions for general-purpose computing on graphics processing units). TensorFlow is available on 64-bit Linux, macOS, Windows, and mobile computing platforms including Android and iOS.

### 3.3.5 KERAS

Keras is a Python-based high-level neural network API that can be used with TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay is key to doing good research. Keras allows for easy and fast prototyping (through user friendliness, modularity, and extensibility). Supports both convolutional networks and recurrent networks, as well as combinations of the two. Runs seamlessly on CPU and GPU. The library includes a variety of implementations of common neural network building components like layers, objectives, activation functions, optimizers, and a variety of other tools to make dealing with picture and text data easier.

### 3.3.6 COMPILER OPTION

Google has released Colaboratory: a web IDE for python, to enable Machine Learning with storage on the cloud. Colaboratory, or “Colab” for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education.

## 3.4 NEURAL NETWORKS

A Neural Network is made up of layers that are connected to operate on the structure and function of the human brain. It learns from massive amounts of data and trains a neural network using complex algorithms.

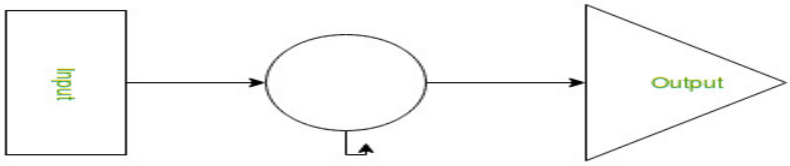
Several neural networks can help solve different business problems. Let's look at a few of them.

- **Feed-Forward Neural Network**: Used for general Regression and Classification problems.
- **Convolutional Neural Network**: Used for object detection and image classification.
- **Deep Belief Network**: Used in healthcare sectors for cancer detection.
- **RNN**: Used for speech recognition, voice recognition, time series prediction, and natural language processing.

### 3.4.1 RECURRENT NEURAL NETWORK (RNN)

Recurrent Neural Network(RNN) is a type of Neural Network where the output from previous step RNNs are a sort of Neural Network in which the output from the previous step is used as input in the next stage. Traditional neural networks have inputs and outputs that are independent of one another, however in some circumstances, such as when predicting the next word of a phrase, the prior words are necessary, and so the previous words must be remembered. As a result, RNN was created, which used a Hidden Layer to overcome the problem. The hidden state, which remembers certain information about a sequence, is the most significant element of RNN. They have a "memory" that stores all information about the calculations. It employs the same settings for each input since it produces the same outcome by performing the same task on all

inputs or hidden layers. Unlike other neural networks, this decreases the complexity of parameters.




## Converting Feed-Forward Neural Network into a Recurrent Neural Network

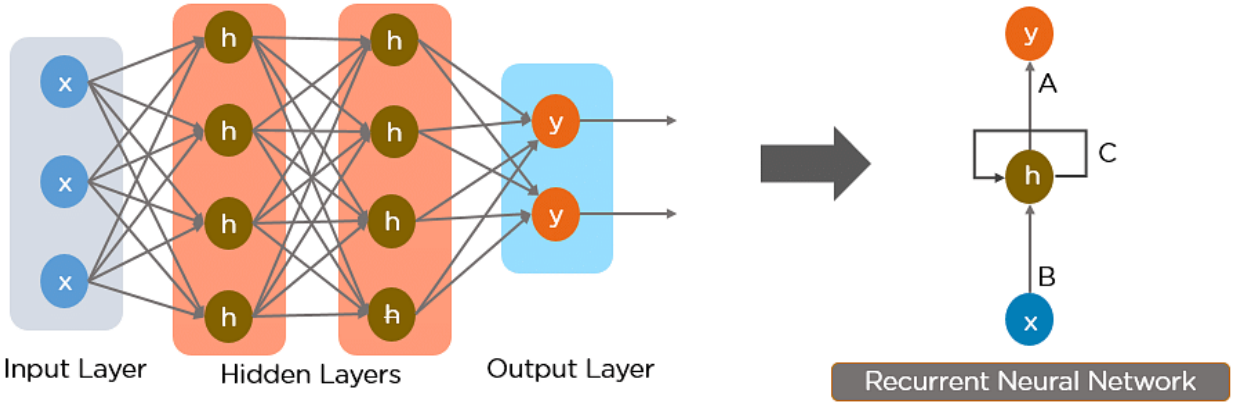



**Fig: Simple Recurrent Neural Network**

The nodes in different layers of the neural network are compressed to form a single layer of recurrent neural networks. A, B, and C are the parameters of the network. Here, "x" is the input layer, "h" is the hidden layer, and "y" is the output layer. A, B, and C are the network parameters used to improve the output of the model. At any given time t, the current input is a combination of input at x(t) and x(t-1). The output at any given time is fetched back to the network to improve on the output.

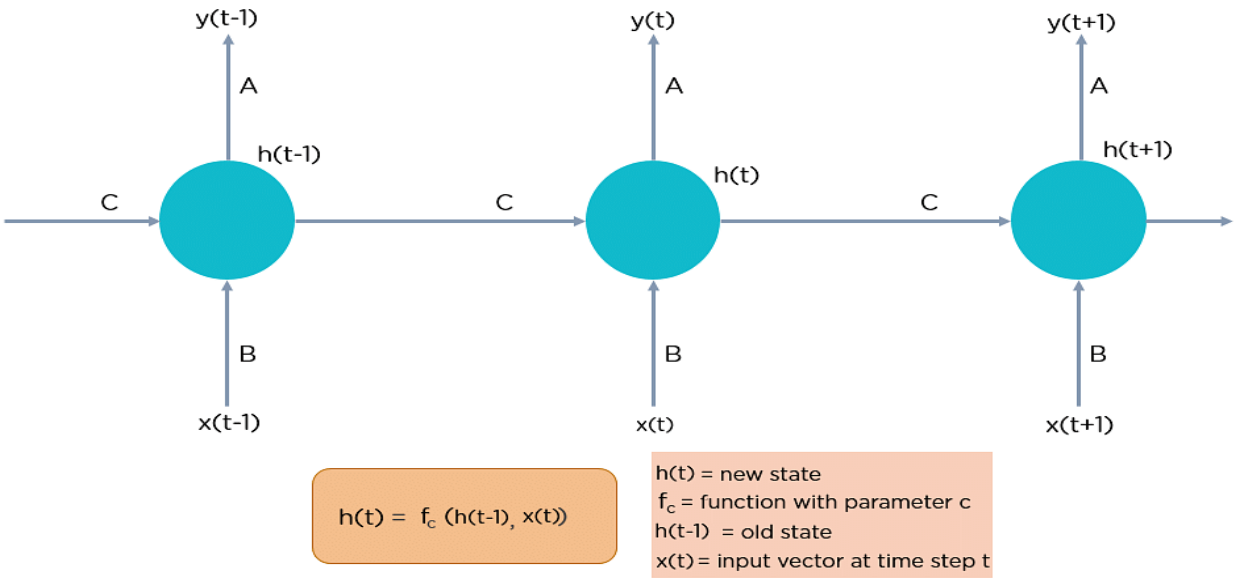



**Fig: Fully connected Recurrent Neural Network**

## Recurrent Neural Networks Working

In Recurrent Neural networks, the information cycles through a loop to the middle hidden layer.

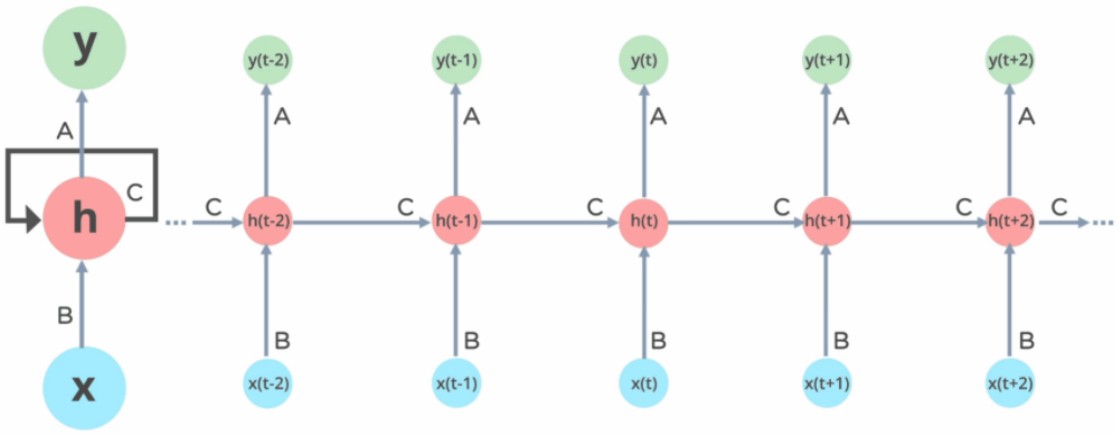



**Fig: Working of Recurrent Neural Network**

The input layer ‘x’ takes in the input to the neural network and processes it and passes it onto the middle layer.

The middle layer ‘h’ can consist of multiple hidden layers, each with its own activation functions and weights and biases. You can utilize a recurrent neural network if the various parameters of different hidden layers are not affected by the preceding layer, i.e. the neural network does not have memory.

The different activation functions, weights, and biases will be standardized by the Recurrent Neural Network so that each hidden layer has the same parameters. Then, rather than producing numerous hidden layers, it will generate only one and loop over it as many times as necessary.

## Types of RNN

The four commonly used types of Recurrent Neural Networks are:

### 1. One-to-One :

The simplest type of RNN is One-to-One, which allows a single input and a single output. It is a standard neural network with set input and output sizes. The One-to-One application can be found in Image Classification.


**Fig: One-to-One**

## 2. One-to-Many :

One-to-Many is a type of RNN that gives multiple outputs when given a single input. It takes a fixed input size and gives a sequence of data outputs. Its applications can be found in Music Generation and Image Captioning.

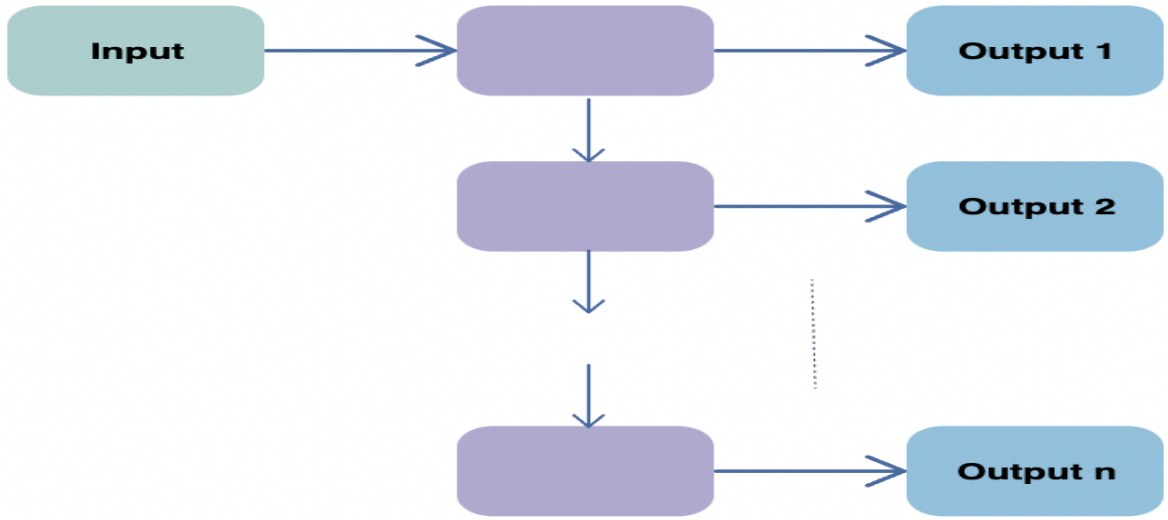



**Fig: One-to-Many**

## 3. Many-to-One :

**Many-to-One** is used when a single output is required from multiple input units or a sequence of them. It takes a sequence of inputs to display a fixed output. *Sentiment Analysis* is a common example of this type of Recurrent Neural Network.

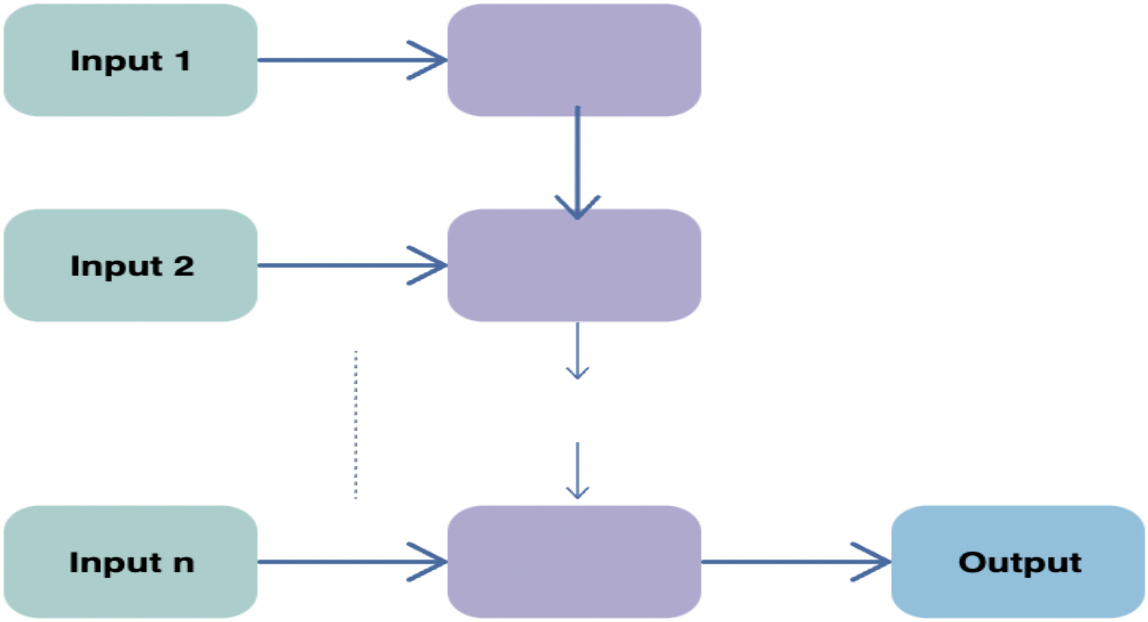



**Fig: Many-to-One**

## 4. Many-to-Many :

**Many-to-Many** is used to generate a sequence of output data from a sequence of input units.

This type of RNN is further divided into the following two subcategories:

**a. Equal Unit Size:** In this case, the number of both the input and output units is the same. A common application can be found in Name-Entity Recognition.

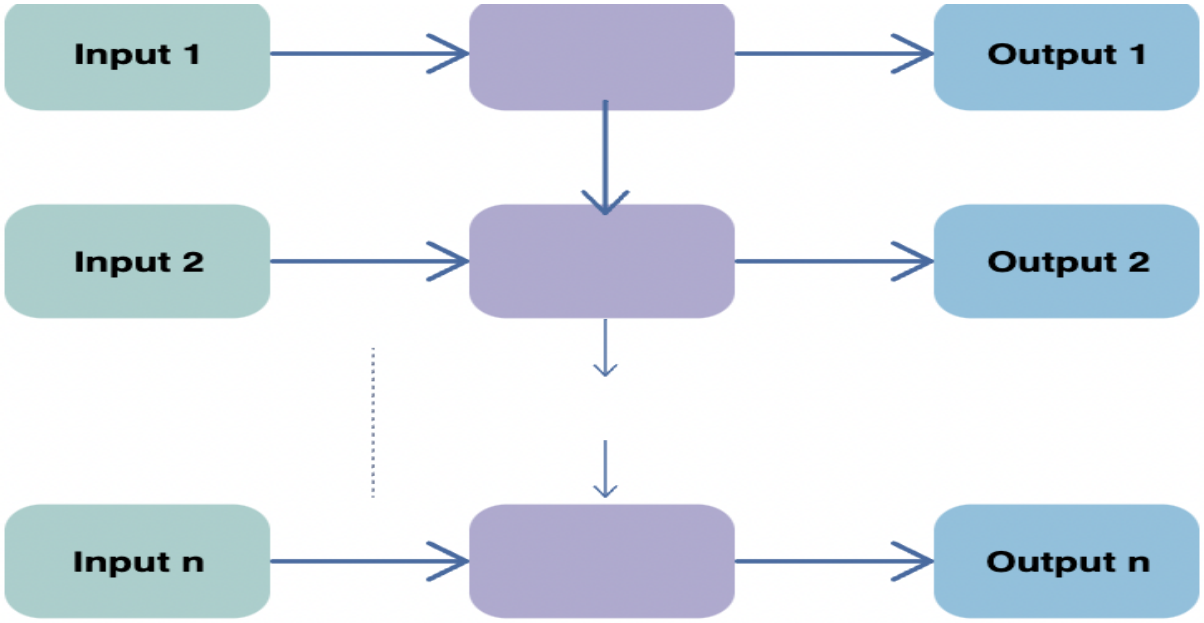



**Fig: Many-to-Many (Equal)**

**b. Unequal Unit Size:** In this case, inputs and outputs have different numbers of units. Its application can be found in Machine Translation.

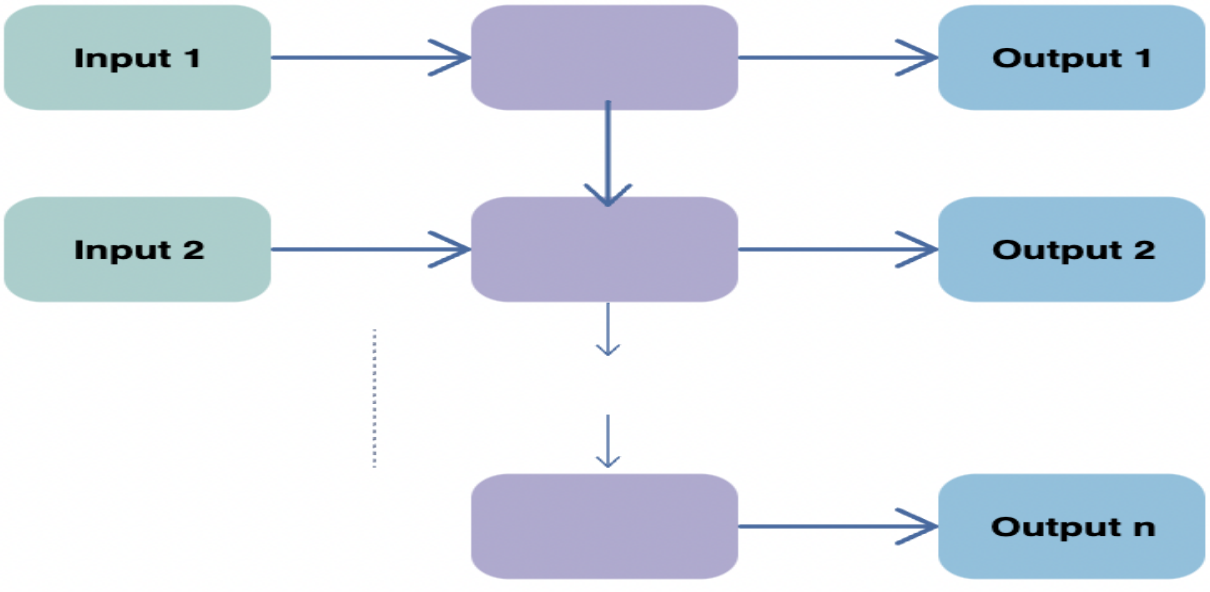



**Fig: Many-to-Many (Unequal)**

## Applications of Recurrent Neural Networks

**Image Captioning:** By assessing the activities included in an image, RNNs are utilized to caption it.

**Time Series Prediction:** An RNN can be used to tackle any time series problem, such as predicting stock prices in a specific month.

**Natural Language Processing:** An RNN for Natural Language Processing can be used to do text mining and sentiment analysis (NLP).

**Machine Translation:** RNNs can be used to translate a single language input into multiple languages as an output.

## Benefits of RNN

- Processes sequential data
- Can memorize and store previous results
- Takes into account both the current and the previous results in the computation of new results
- Regardless of the increasing size of the input, the model size remains fixed
- It shares weights to other units across time

## 3.4.2 LONG SHORT TERM MEMORY(LSTM)

Long short-term memory networks (LSTMs) are an extension of Recurrent Neural Networks that expand the memory capacity. As a result, it is highly suited to learning from significant experiences separated by long time lags. The units of an LSTM are used as building units for the layers of a RNN, often called an LSTM network. RNNs can recall inputs for a long time because of LSTMs. This is because LSTMs store information in a memory similar to a computer's memory. The LSTM has the ability to read, write, and delete data from its memory.

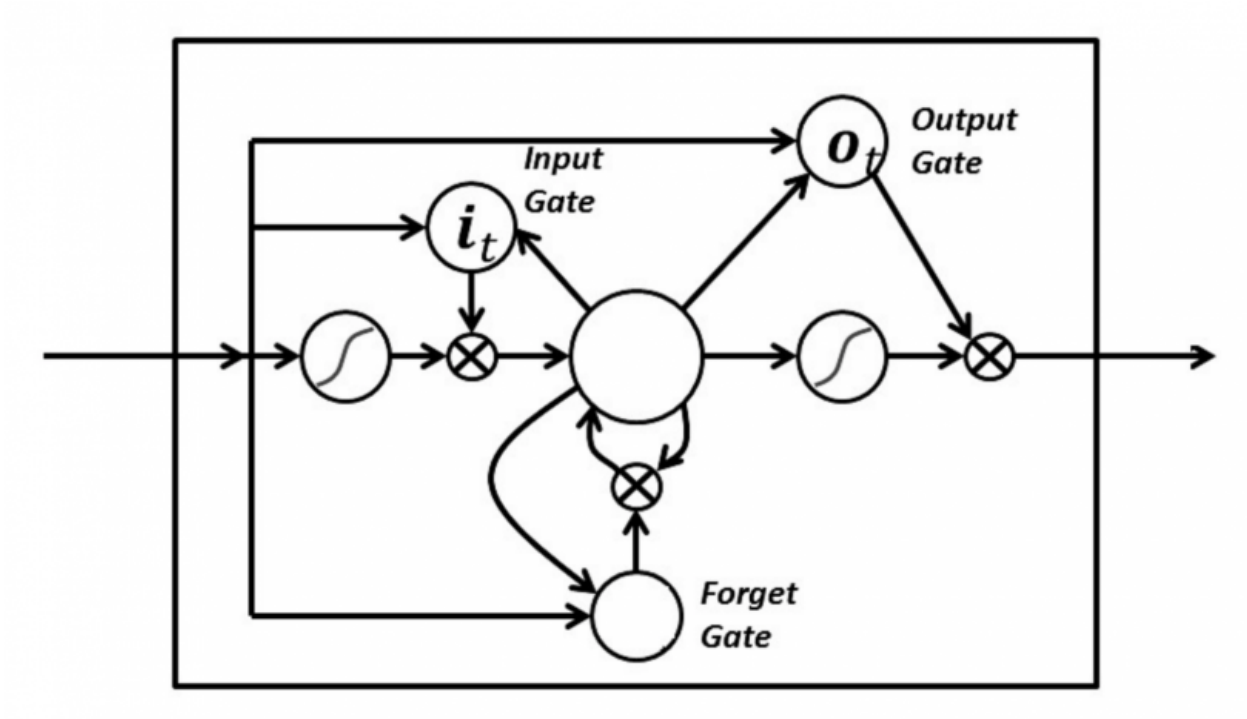
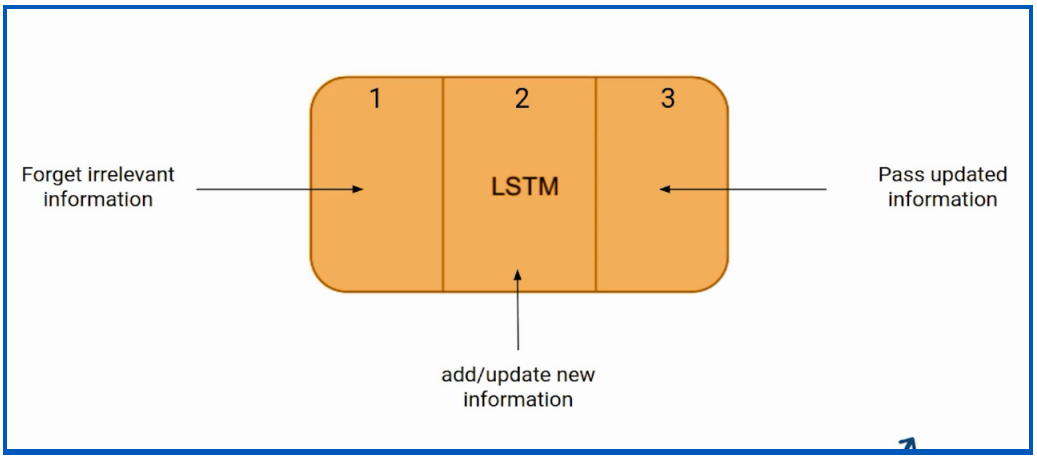



**Fig: LSTM**

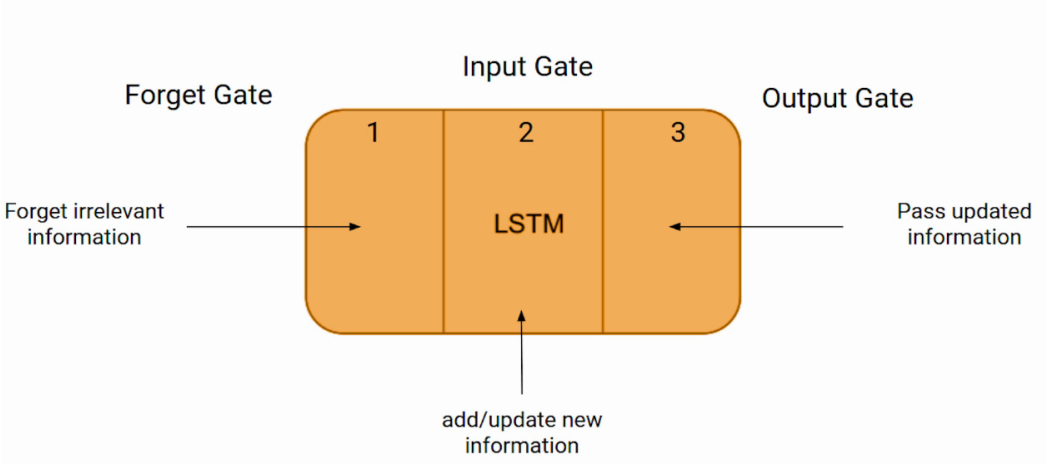
**LSTM ARCHITECTURE**

At a high-level LSTM works very much like an RNN cell. Here is the internal functioning of the LSTM network. The LSTM consists of three parts, as shown in the image below and each part performs an individual function.

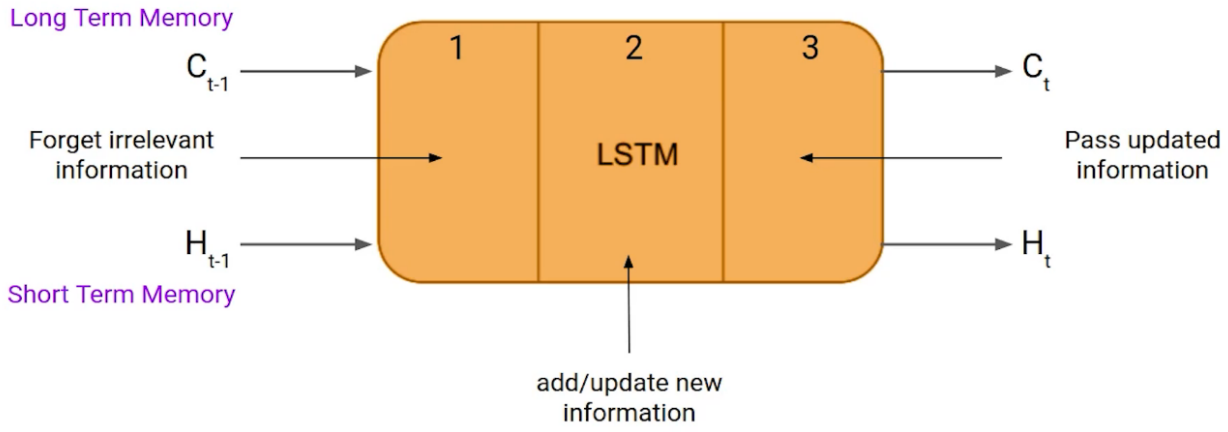



The first part determines whether the information from the preceding timestamp is important to remember or can be ignored. The cell attempts to learn new information from the input in the second part. Finally, the cell passes updated information from the current timestamp to the next timestamp in the third part.

These three parts of an LSTM cell are known as gates. The first part is called Forget gate, the second part is known as the Input gate and the last one is the Output gate.



Just like a simple RNN, an LSTM also has a hidden state where H(t-1) represents the hidden state of the previous timestamp and Ht is the hidden state of the current timestamp. LSTM additionally contains a cell state for prior and current timestamps, which is represented by C(t-1) and C(t), respectively. Short term memory refers to the hidden state, whereas Long term memory refers to the cell state. Here the cell state carries the information along with all the timestamps. Please see the illustration below.




## FORGET GATE

In a cell of the LSTM network, the first step is to decide whether we should keep the information from the previous timestamp or forget it. Here is the equation for the forget gate.

- $f_t = \sigma\left(x_t * U_f + H_{t-1} * W_f\right)$

Let's try to understand the equation, here

- Xt: input to the current timestamp.
- Uf: weight associated with the input
- Ht-1: The hidden state of the previous timestamp
- Wf: It is the weight matrix associated with hidden state

Later, a sigmoid function is applied over it. That will make it a number between 0 and 1. This ft is later multiplied with the cell state of the previous timestamp. If ft is 0 then the network will forget everything and if the value of ft is 1 it will forget nothing.

$$C_{t-1} * f_t = 0 \quad \text{...if } f_t = 0 \text{ (forget everything)}$$

$$C_{t-1} * f_t = C_{t-1} \quad \text{...if } f_t = 1 \text{ (forget nothing)}$$

**INPUT GATE**

Input gate is used to quantify the importance of the new information carried by the input. Here is the equation of the input gate.

- $i_t = \sigma (x_t * U_i + H_{t-1} * W_i)$

Here,

- Xt: Input at the current timestamp t
- Ui: weight matrix of input

- Ht-1: A hidden state at the previous timestamp
- Wi: Weight matrix of input associated with hidden state

Again we have applied a sigmoid function over it. As a result, the value of I at timestamp t will be between 0 and 1.

**New information**

- $N_t = \tanh(x_t * U_c + H_{t-1} * W_c)$ (new information)

The new information to be transmitted to the cell state is now a function of a concealed state at timestamp t-1 and input x at timestamp t. Tanh is the activation function here. The value of new information will be between -1 and 1 due to the tanh function. If Nt is negative, data is subtracted from the cell state; otherwise, data is added to the cell state at the current timestamp. However, the Nt won't be added directly to the cell state. Here comes the updated equation.

$C_t = f_t * C_{t-1} + i_t * N_t$ (updating cell state)

Here, Ct-1 is the cell state at the current timestamp and others are the values we have calculated previously.

## OUTPUT GATE

The Output gate has an equation that is very similar to the two prior gates.

- $o_t = \sigma(x_t * U_o + H_{t-1} * W_o)$

Because of the sigmoid function, its value will also be between 0 and 1. We will now utilize the modified cell state's Ot and tanh to determine the current hidden state. As illustrated below.

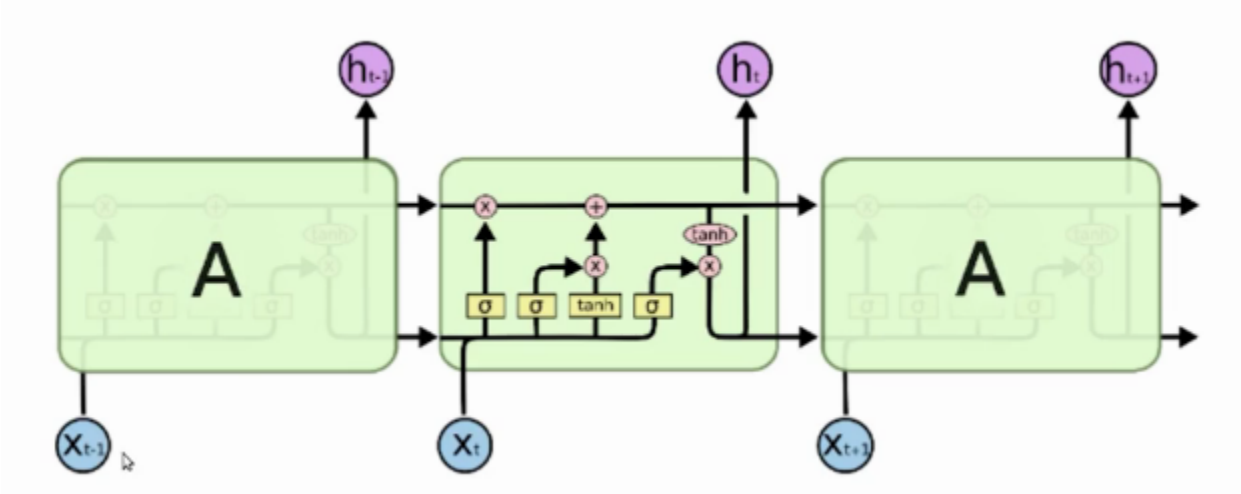
$$H_t = o_t * \tanh(C_t)$$

The hidden state turns out to be a function of long-term memory (Ct) and current output. Apply the SoftMax activation to hidden state Ht if you need the output of the current timestamp.

$$\text{Output} = \text{Softmax}(H_t)$$

Here the token with the maximum score in the output is the prediction.

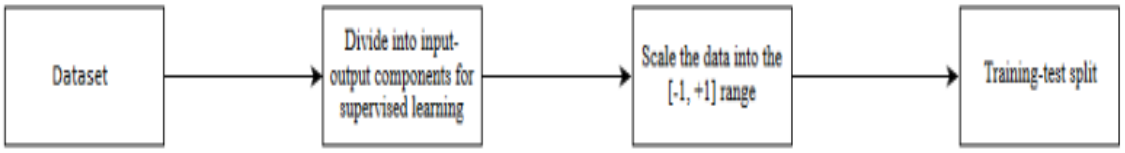
**More intuitive diagram of the LSTM network :**



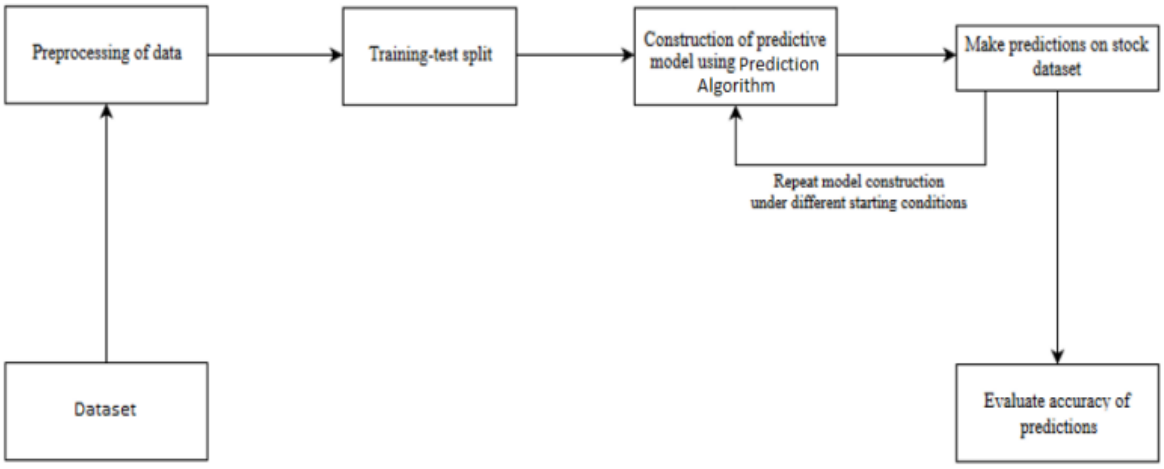
# CHAPTER 4

# SYSTEM ARCHITECTURE

## 4.1 PREPROCESSING OF DATA




## 4.2 OVERALL ARCHITECTURE



# CHAPTER 5

# DESIGN AND IMPLEMENTATION

## 5.1 PROPOSED SYSTEM

In this proposed system, we focus on predicting the stock values using machine learning algorithms like Long Short Term Memory (LSTM). We were able to train the computer using multiple data points from the past to produce a future forecast in our proposed method. To train the algorithm, we used data from the previous year's stocks. To solve the challenge, we primarily used two machine-learning packages. The first was numpy, which was used to clean and alter the data before putting it into a format that could be analyzed. Scikit, on the other hand, was employed for real-time analysis and prediction. We used data from prior years' stock markets, which we obtained from a public database accessible online. We used 80% of the data to train the machine and 20% to test the data. The supervised learning model's core approach is to learn patterns and correlations in data from the training set and then replicate them for the test data.For data processing, we used the Python pandas module, which merged many datasets into a single data frame. We were able to prepare the data for feature extraction using the tuned dataframe. The date and the closing price for a specific day were the data frame's features. We used all of these characteristics to train the computer using a random forest model and predict the object variable, which is the price for a particular day. We also calculated the accuracy by comparing the test set predictions to the actual values. The proposed systems cover a wide range of topics, including data pre-processing and many more.

## 5.2 UML DIAGRAMS

A UML diagram is a partial graphical representation (view) of a model of a system under design, implementation, or already in existence. The 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 basic graphical symbols depicted on the diagram identify the diagram's type. A class diagram, for example, is one in which the major symbols in the contents area are classes. Use case diagram is a diagram that depicts use cases and actors. A sequence diagram depicts the order in which messages are exchanged between lifelines.

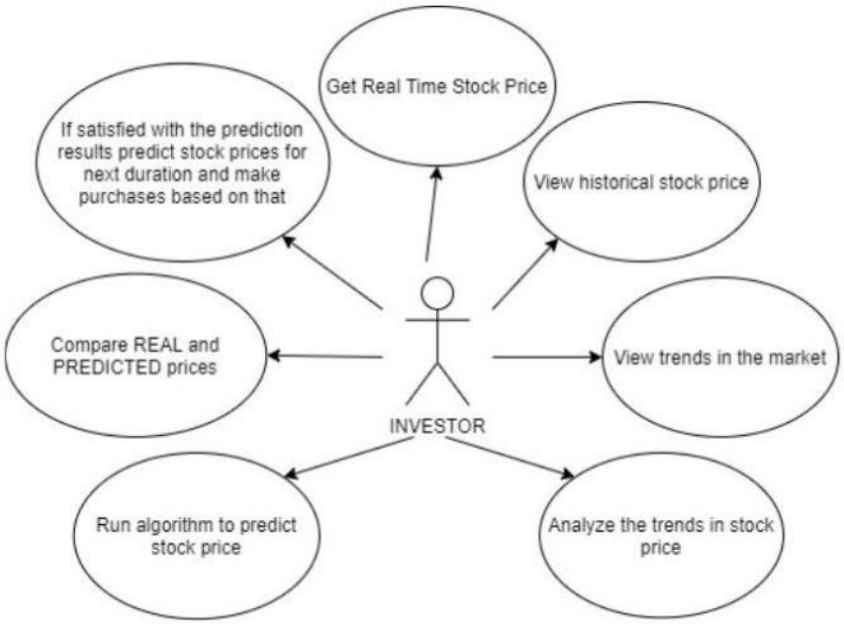
The UML specification allows for the merging of multiple types of diagrams, for example, to represent a state machine nested inside a use case by combining structural and behavioral elements. As a result, the distinctions between different types of diagrams are rarely tightly enforced. However, certain UML Tools limit the number of graphical elements that can be utilized when working on a given type of diagram.

UML specification defines two major kinds of UML diagrams: structure diagrams and behavior diagrams. Structure diagrams depict the system's static structure and its components at various abstraction and implementation levels, as well as their relationships. Behavior diagrams depict the items in a system's dynamic behavior, which can be represented as a series of changes over time.

### 5.2.1 USE CASE DIAGRAM

A use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system in the Unified Modeling Language (UML). You'll need a collection of specialized symbols and connectors to construct one. A good use case diagram can assist your team in discussing and representing.

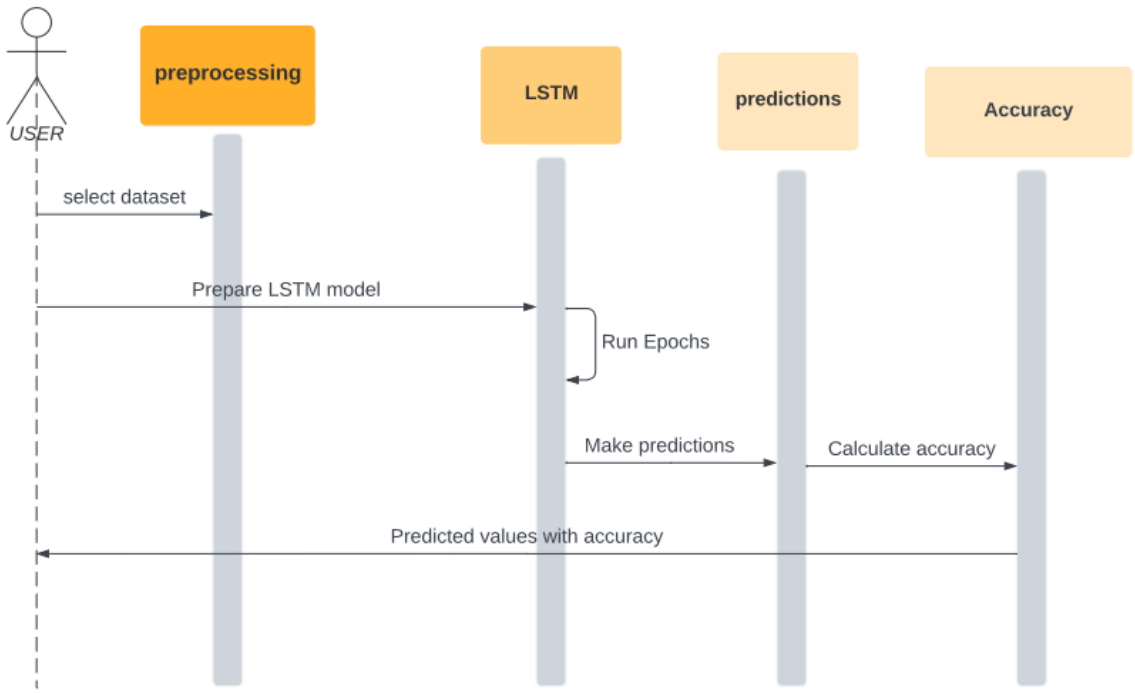
- Scenarios in which your system or application interacts with people, organizations, or external systems.
- Goals that your system or application helps those entities (known as actors) achieve.
- The scope of your system.



## 5.2.2 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.Software engineers and business experts use these diagrams to understand the requirements for a new system or to describe an existing process. Event diagrams and event scenarios are other names for sequence diagrams. Sequence diagrams can be useful references for businesses and other organizations. Try drawing a sequence diagram to:

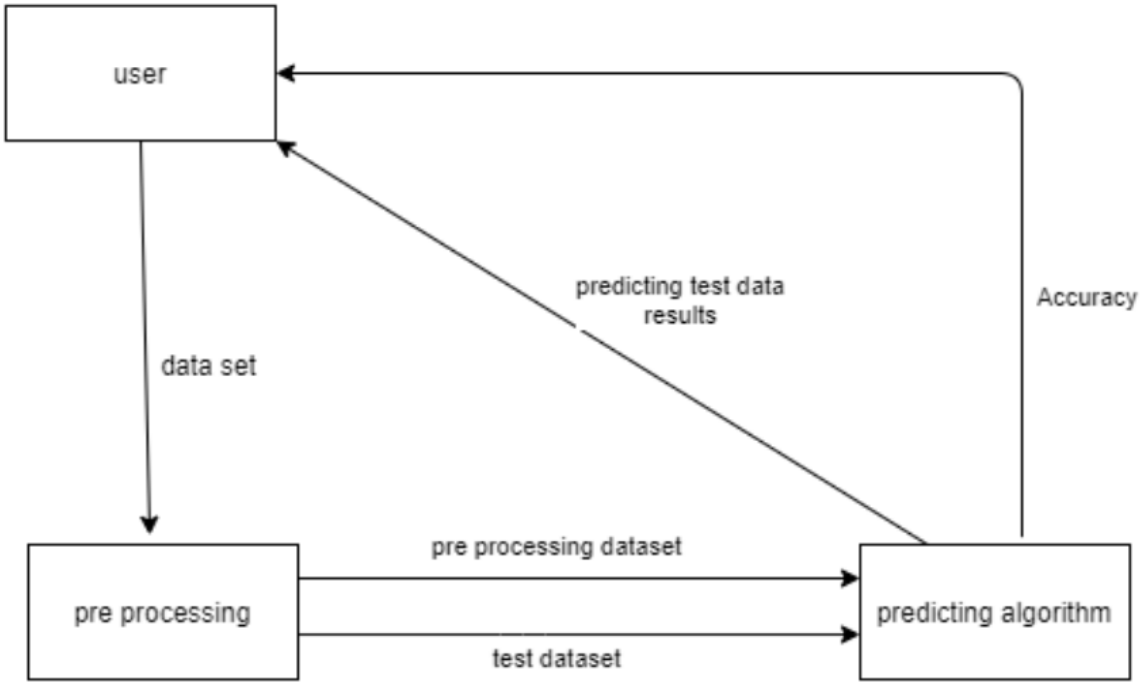
- Represent the details of a UML use case.
- Model the logic of a sophisticated procedure, function, or operation.
- Examine how items and components work together to execute a task.
- Plan and understand the detailed functionality of an existing or future scenario.



### 5.2.3 COLLABORATION DIAGRAM

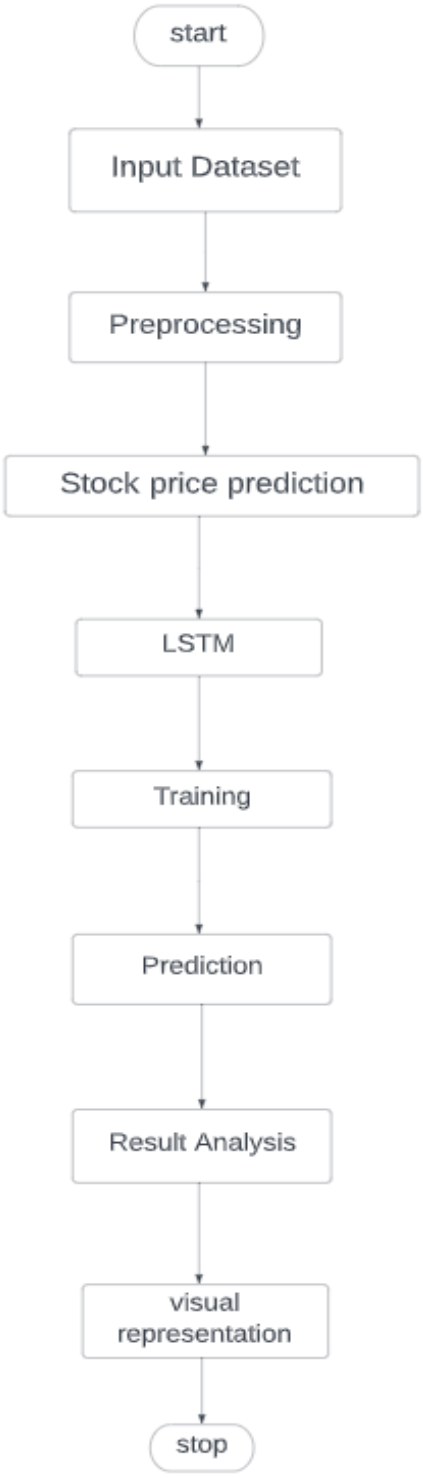
Collaboration diagrams are used to depict how objects interact to carry out the behavior of a specific use case, or a subset of a use case. Designers employ collaboration, together with sequence diagrams, to define and clarify the roles of the objects that perform a certain flow of actions in a use case. They are the most important source of data for determining class responsibilities and interfaces.

When it is necessary to illustrate the relationship between the objects, collaborations are used. The sequence and collaboration diagrams both depict the same information, but they do so in quite different ways. Use cases are best analyzed using collaboration diagrams.



### 5.2.4 FLOW CHART

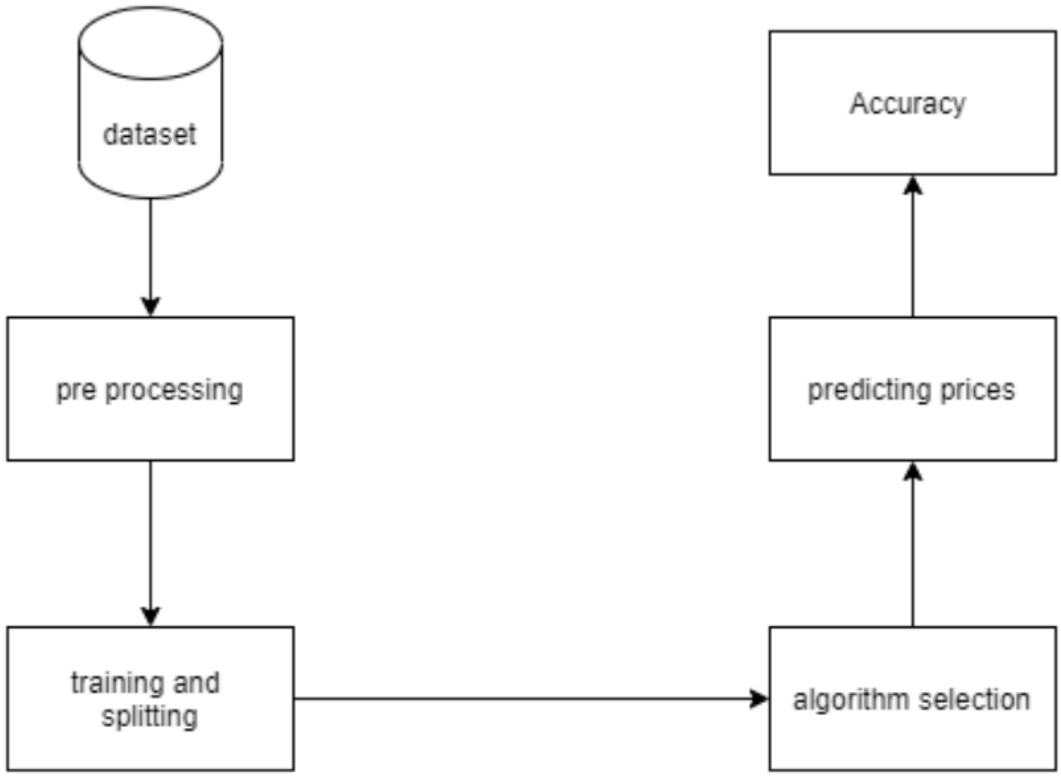
A flowchart is a diagram that shows how a workflow or process works. A flowchart is a diagrammatic representation of an algorithm, or a step-by-step procedure for completing a task. The flowchart depicts the steps as various types of boxes, with arrows linking in order.



### 5.2.5 COMPONENT DIAGRAM

In UML, a component diagram is a special type of diagram. The goal is also distinct from all of the previous diagrams. It does not explain the system's functionality; rather, it describes the components that enable such functions.

Component diagrams are used to depict the physical aspects of object-oriented systems. They are used for visualizing, describing, and documenting component-based systems, as well as for forward and reverse engineering to create executable systems. Component diagrams are essentially class diagrams that focus on the components of a system and are frequently used to describe the system's static implementation view.



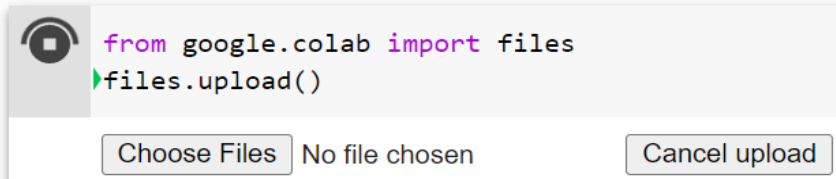
## 5.3 SAMPLE CODE

- Import the libraries

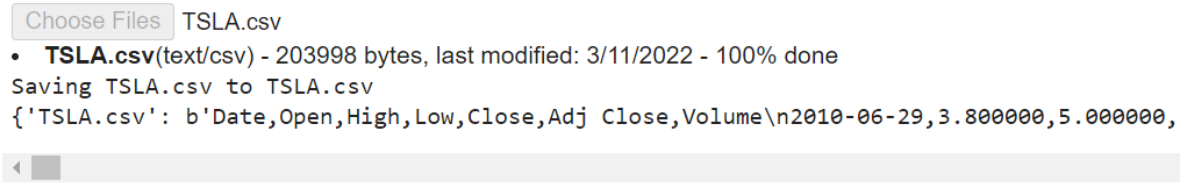
```
[1] import math
    import numpy as np
    import pandas as pd
    from sklearn.preprocessing import MinMaxScaler
    import matplotlib.pyplot as plt
```

- Get the stock data

```
from google.colab import files
files.upload()
```




```
[2] from google.colab import files
    files.upload()
```



- Show the data

```
[3]  df=pd.read_csv('TSLA.csv')
     df = df.set_index(pd.DatetimeIndex(df['Date'].values))
     df.index.name = 'Date'
     df
```

| Date | Date | Open | High | Low | Close | Adj Close | Volume |
|---|---|---|---|---|---|---|---|
| **2010-06-29** | 2010-06-29 | 3.800000 | 5.000000 | 3.508000 | 4.778000 | 4.778000 | 93831500 |
| **2010-06-30** | 2010-06-30 | 5.158000 | 6.084000 | 4.660000 | 4.766000 | 4.766000 | 85935500 |
| **2010-07-01** | 2010-07-01 | 5.000000 | 5.184000 | 4.054000 | 4.392000 | 4.392000 | 41094000 |
| **2010-07-02** | 2010-07-02 | 4.600000 | 4.620000 | 3.742000 | 3.840000 | 3.840000 | 25699000 |
| **2010-07-06** | 2010-07-06 | 4.000000 | 4.000000 | 3.166000 | 3.222000 | 3.222000 | 34334500 |
| **...** | ... | ... | ... | ... | ... | ... | ... |
| **2022-02-10** | 2022-02-10 | 908.369995 | 943.809998 | 896.700012 | 904.549988 | 904.549988 | 22042300 |
| **2022-02-11** | 2022-02-11 | 909.630005 | 915.960022 | 850.700012 | 860.000000 | 860.000000 | 26492700 |
| **2022-02-14** | 2022-02-14 | 861.570007 | 898.880005 | 853.150024 | 875.760010 | 875.760010 | 22515100 |
| **2022-02-15** | 2022-02-15 | 900.000000 | 923.000000 | 893.380005 | 922.429993 | 922.429993 | 19095400 |
| **2022-02-16** | 2022-02-16 | 914.049988 | 917.659973 | 901.219971 | 902.000000 | 902.000000 | 9167269 |

2931 rows × 7 columns

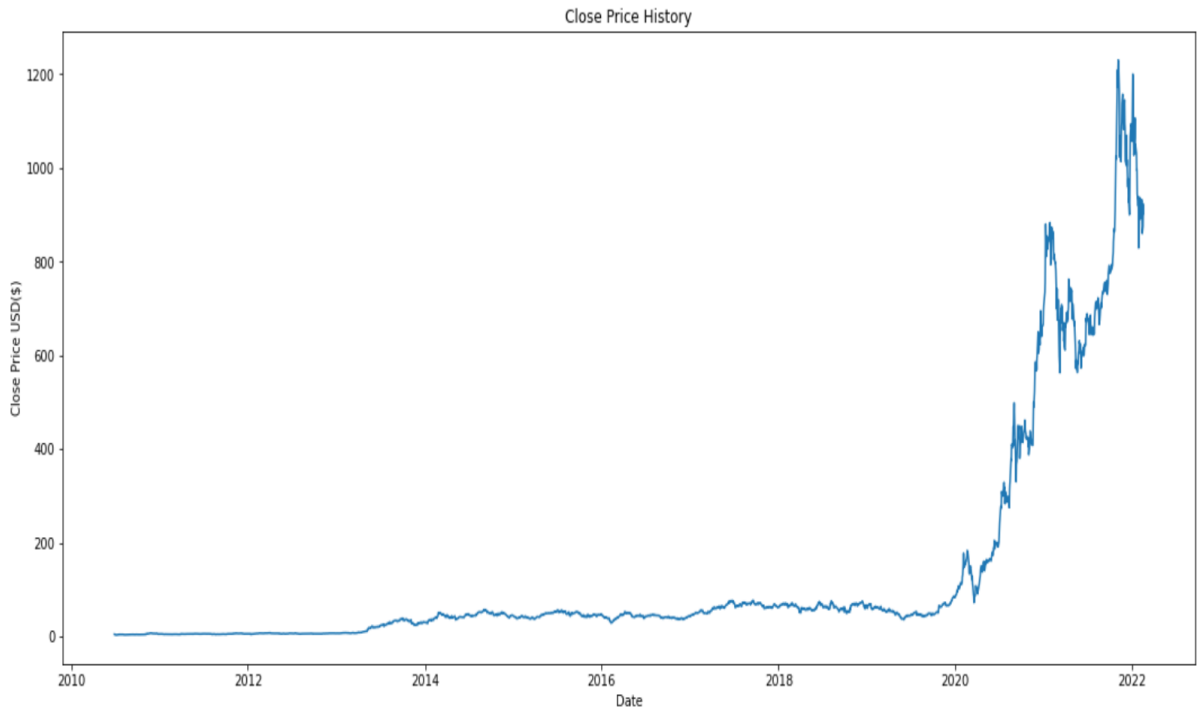
- Get the number of rows and columns in data set

```
[4]  df.shape
```

```
(2931, 7)
```

- Visualize the closing price history

```
[5]  plt.figure(figsize=(18,8))
     plt.title('Close Price History')
     plt.plot(df['Close'])
     plt.xlabel('Date')
     plt.ylabel('Close Price USD($)')
     plt.show()
```



- Create a new dataframe with only the ‘close’ column
- Convert the dataframe to numpy array
- Get the number of rows to train the model on

```
[6]  data=df.filter(['Close'])
     dataset=data.values
     training_data_len=math.ceil(len(dataset) * .8)
     training_data_len
```

```
2345
```

- Scale the data

```
[7]  scaler=MinMaxScaler(feature_range=(0,1))
     scaled_data=scaler.fit_transform(data)
     scaled_data
```

```
array([[0.00131893],
       [0.00130915],
       [0.00100428],
       ...,
       [0.71131036],
       [0.74935396],
       [0.7327002 ]])
```

- Create the training dataset
- Split the data into x_train and y_train data sets

```
[8]  train_data= scaled_data[0:training_data_len, :]
     x_train = []
     y_train = []
     for i in range(60, len(train_data)):
       x_train.append(train_data[i-60:i,0])
       y_train.append(train_data[i,0])
       if i<= 61:
         print(x_train)
         print(y_train)
         print()
```

```
[array([1.31893210e-03, 1.30915016e-03, 1.00427957e-03, 5.54310154e-04,
       5.05400434e-05, 0.00000000e+00, 2.70633781e-04, 2.60851837e-04,
       2.03790498e-04, 3.81495812e-04, 6.58650889e-04, 6.66802509e-04,
       7.89076807e-04, 9.96127953e-04, 7.33645792e-04, 7.20603200e-04,
       8.47768470e-04, 8.95047866e-04, 8.39616851e-04, 7.74403891e-04,
       8.02119399e-04, 7.41797412e-04, 6.74954128e-04, 8.34725879e-04,
       1.00264925e-03, 8.90156894e-04, 7.58100651e-04, 6.17892789e-04,
       6.19523113e-04, 5.26594646e-04, 3.42368036e-04, 2.93458317e-04,
       4.10841643e-04, 4.85836547e-04, 5.46158534e-04, 4.84206223e-04,
       4.87466871e-04, 5.38006914e-04, 7.05930284e-04, 5.54310154e-04,
       6.68432833e-04, 6.43977973e-04, 6.35826353e-04, 6.63541861e-04,
       5.99959225e-04, 7.58100651e-04, 8.57550414e-04, 8.55920090e-04,
       7.72773567e-04, 8.31465231e-04, 8.00489075e-04, 7.12451580e-04,
       8.02119399e-04, 8.67332358e-04, 1.00754022e-03, 8.37986527e-04,
       7.22233524e-04, 8.57550414e-04, 8.10271019e-04, 6.63541861e-04])]
[0.0006130018171248731]
```

```
[8] [array([1.31893210e-03, 1.30915016e-03, 1.00427957e-03, 5.54310154e-04,
        5.05400434e-05, 0.00000000e+00, 2.70633781e-04, 2.60851837e-04,
        2.03790498e-04, 3.81495812e-04, 6.58650889e-04, 6.66802509e-04,
        7.89076807e-04, 9.96127953e-04, 7.33645792e-04, 7.20603200e-04,
        8.47768470e-04, 8.95047866e-04, 8.39616851e-04, 7.74403891e-04,
        8.02119399e-04, 7.41797412e-04, 6.74954128e-04, 8.34725879e-04,
        1.00264925e-03, 8.90156894e-04, 7.58100651e-04, 6.17892789e-04,
        6.19523113e-04, 5.26594646e-04, 3.42368036e-04, 2.93458317e-04,
        4.10841643e-04, 4.85836547e-04, 5.46158534e-04, 4.84206223e-04,
        4.87466871e-04, 5.38006914e-04, 7.05930284e-04, 5.54310154e-04,
        6.68432833e-04, 6.43977973e-04, 6.35826353e-04, 6.63541861e-04,
        5.99959225e-04, 7.58100651e-04, 8.57550414e-04, 8.55920090e-04,
        7.72773567e-04, 8.31465231e-04, 8.00489075e-04, 7.12451580e-04,
        8.02119399e-04, 8.67332358e-04, 1.00754022e-03, 8.37986527e-04,
        7.22233524e-04, 8.57550414e-04, 8.10271019e-04, 6.63541861e-04]), array([1.30915016e-03, 1.00427957e-03, 5.54310154e-04, 5.05400434e-05,
        0.00000000e+00, 2.70633781e-04, 2.60851837e-04, 2.03790498e-04,
        3.81495812e-04, 6.58650889e-04, 6.66802509e-04, 7.89076807e-04,
        9.96127953e-04, 7.33645792e-04, 7.20603200e-04, 8.47768470e-04,
        8.95047866e-04, 8.39616851e-04, 7.74403891e-04, 8.02119399e-04,
        7.41797412e-04, 6.74954128e-04, 8.34725879e-04, 1.00264925e-03,
        8.90156894e-04, 7.58100651e-04, 6.17892789e-04, 6.19523113e-04,
        5.26594646e-04, 3.42368036e-04, 2.93458317e-04, 4.10841643e-04,
        4.85836547e-04, 5.46158534e-04, 4.84206223e-04, 4.87466871e-04,
        5.38006914e-04, 7.05930284e-04, 5.54310154e-04, 6.68432833e-04,
        6.43977973e-04, 6.35826353e-04, 6.63541861e-04, 5.99959225e-04,
        7.58100651e-04, 8.57550414e-04, 8.55920090e-04, 7.72773567e-04,
        8.31465231e-04, 8.00489075e-04, 7.12451580e-04, 8.02119399e-04,
        8.67332358e-04, 1.00754022e-03, 8.37986527e-04, 7.22233524e-04,
        8.57550414e-04, 8.10271019e-04, 6.63541861e-04, 6.13001817e-04])]
    [0.0006130018171248731, 0.0007010393121374878]
```

- Convert the x_train and y_train to numpy arrays

```
[9] x_train,y_train = np.array(x_train),np.array(y_train)
```

- Reshape the data

```
[10] x_train = np.reshape(x_train,(x_train.shape[0],x_train.shape[1], 1))
     x_train.shape
```

```
(2285, 60, 1)
```

- Import the libraries required for lstm model

```
[11] from keras.models import Sequential
     from keras.layers import Dense,LSTM
```

- Build the LSTM model

```
[12] model=Sequential()
     model.add(LSTM(50, return_sequences=True, input_shape=(x_train.shape[1], 1)))
     model.add(LSTM(50, return_sequences=False))
     model.add(Dense(25))
     model.add(Dense(1))
```

- View the summary of LSTM model

```
[13] model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 60, 50)            10400

 lstm_1 (LSTM)               (None, 50)                20200

 dense (Dense)               (None, 25)                1275

 dense_1 (Dense)             (None, 1)                 26

=================================================================
Total params: 31,901
Trainable params: 31,901
Non-trainable params: 0
_________________________________________________________________
```

- Compile the model

```
[14] model.compile(optimizer='adam',loss='mean_squared_error')
```

- Train the model

```
[15] model.fit(x_train,y_train,batch_size=1,epochs=1)
```

```
2285/2285 [==============================] - 75s 31ms/step - loss: 2.0028e-05
<keras.callbacks.History at 0x7fdba0a1ed50>
```

- Create the testing data set

```
[16] test_data = scaled_data[training_data_len - 60: , :]
     x_test = []
     y_test = dataset[training_data_len:, :]
     for i in range(60,len(test_data)):
       x_test.append(test_data[i-60:i, 0])
```

- Convert the data to a numpy array

```
[17] x_test = np.array(x_test)
```

- Reshape the data

```
[18] x_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1], 1))
     x_test.shape
```

```
(586, 60, 1)
```

- Get the models predicted price values:

```
[19] predictions = model.predict(x_test)
     predictions = scaler.inverse_transform(predictions)
```

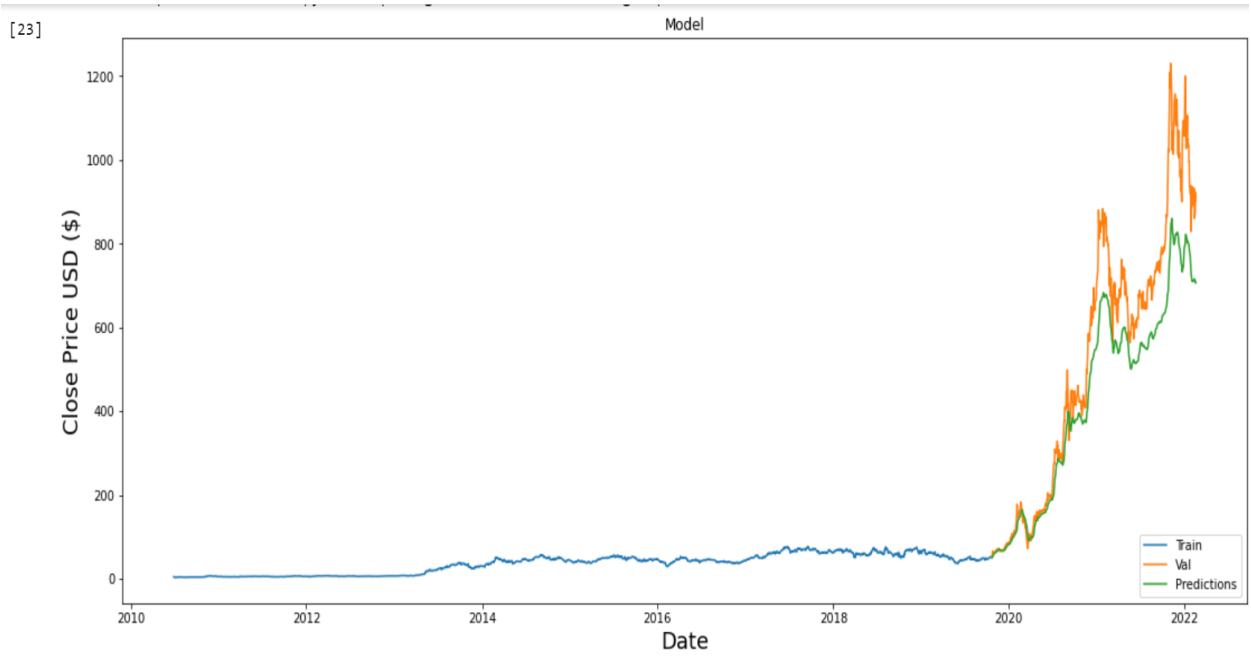
- Get the root mean squared error(RMSE)**:**

```
[20] rmse = np.sqrt( np.mean( predictions - y_test)**2)
     rmse
```

```
93.71412088547221
```

- Plot the data

```
[23] train = data[:training_data_len]
     valid = data[training_data_len:]
     valid['Predictions'] = predictions
     plt.figure(figsize=(18,8))
     plt.title('Model')
     plt.xlabel('Date', fontsize=18)
     plt.ylabel('Close Price USD ($)',fontsize=18)
     plt.plot(train['Close'])
     plt.plot(valid[['Close','Predictions']])
     plt.legend(['Train','Val','Predictions'],loc='lower right')
     plt.show()
```



- Show the valid and predicted prices

```
[24] valid
```

| Date | Close | Predictions |
|---|---|---|
| **2019-10-22** | 51.116001 | 51.190155 |
| **2019-10-23** | 50.936001 | 51.267265 |
| **2019-10-24** | 59.936001 | 51.287018 |
| **2019-10-25** | 65.625999 | 52.349812 |
| **2019-10-28** | 65.542000 | 54.420269 |
| **...** | ... | ... |
| **2022-02-10** | 904.549988 | 715.223511 |
| **2022-02-11** | 860.000000 | 715.358032 |
| **2022-02-14** | 875.760010 | 710.637695 |
| **2022-02-15** | 922.429993 | 706.034302 |
| **2022-02-16** | 902.000000 | 706.226013 |

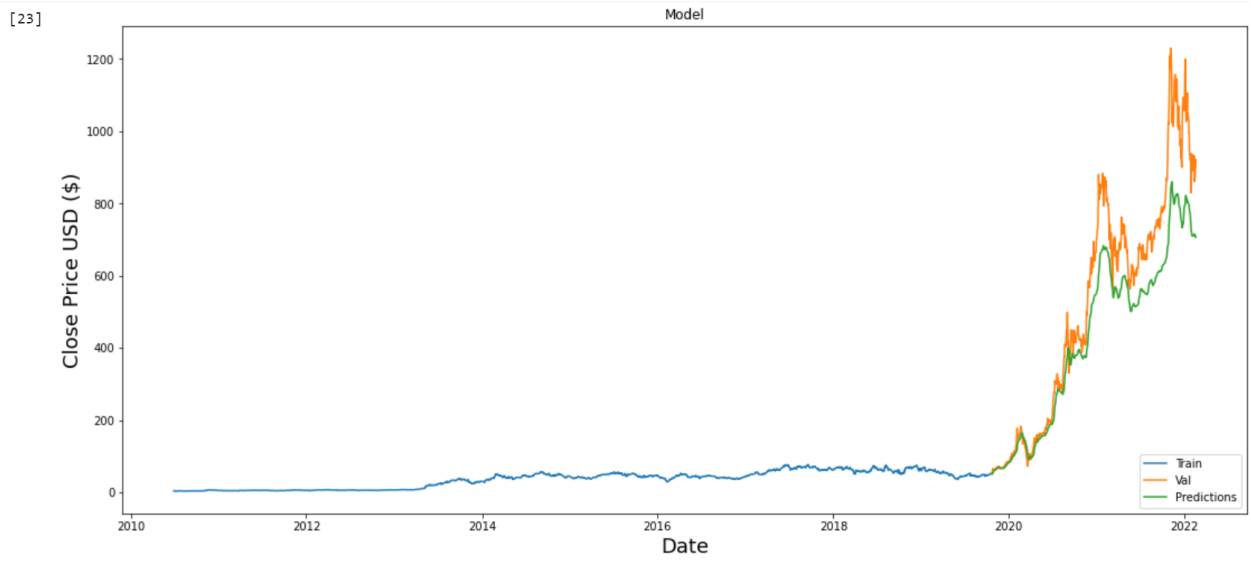
586 rows × 2 columns

# CHAPTER 6

# RESULTS

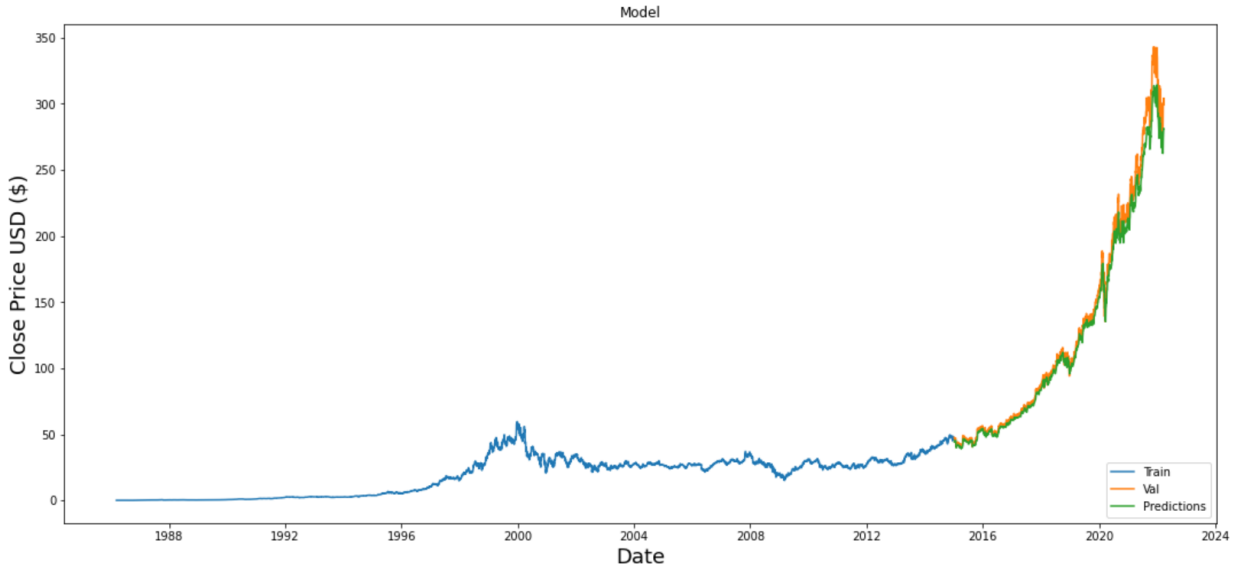
Eventually, after enough training epochs, it will produce better and better results over the time. This is how you would tackle a sequence prediction problem with LSTM. LSTMs are a viable answer for problems involving sequences and time series.

## 6.1 TESLA DATA ANALYSIS



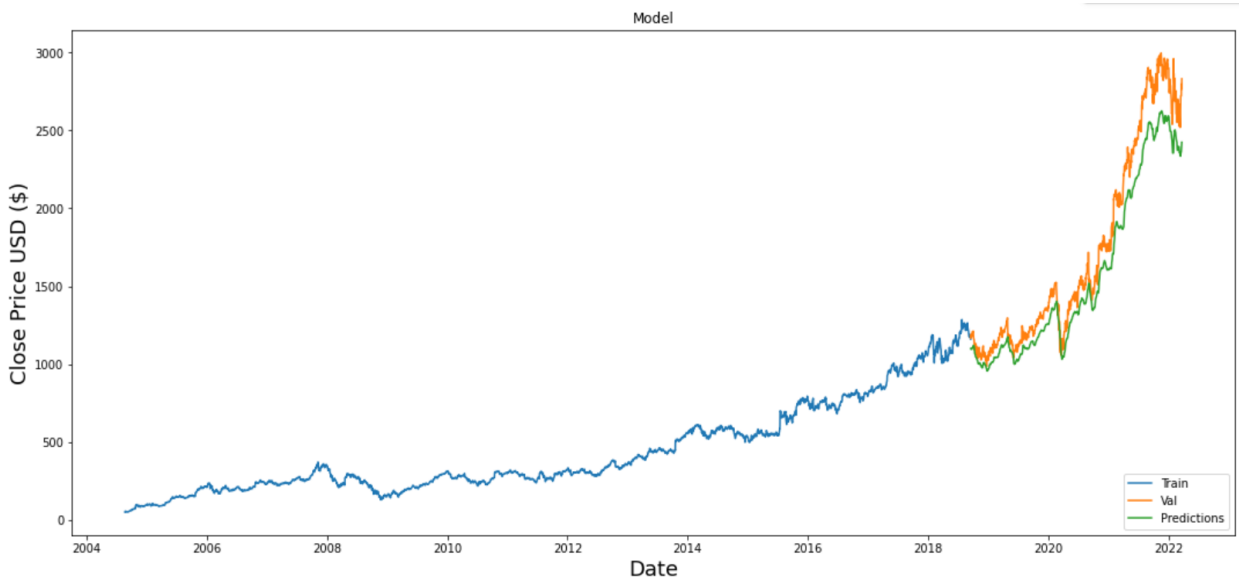

| EPOCHS | RMSE(ROOT MEAN SQUARE ERROR) |
|---|---|
| 1 | 93.71412088547221 |
| 10 | 101.17009283440183 |
| 20 | 162.25310609553938 |

## 6.2 MICROSOFT DATA ANALYSIS



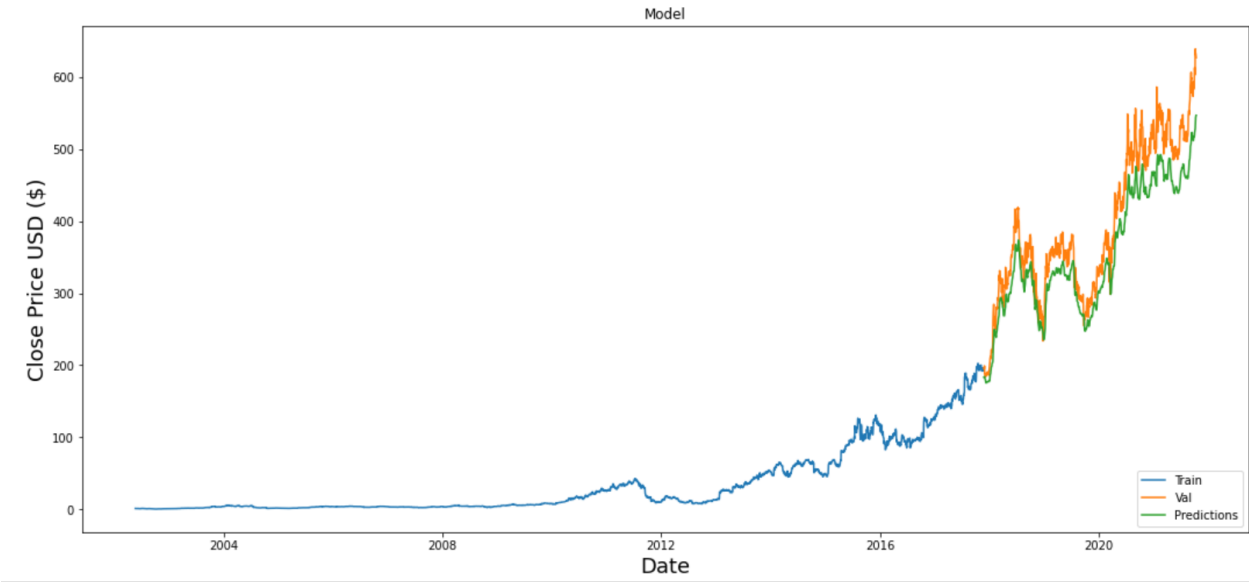

| EPOCHS | RMSE(ROOT MEAN SQUARE ERROR) |
|---|---|
| 1 | 6.065790770447181 |
| 10 | 26.74183326419175 |
| 20 | 26.433530479385443 |

## 6.3 GOOGLE DATA ANALYSIS




| EPOCHS | RMSE(ROOT MEAN SQUARE ERROR) |
|---|---|
| 1 | 155.7935920469552 |
| 10 | 1.5648574256004875 |
| 20 | 81.03911312735875 |

## 6.4 NETFLIX DATA ANALYSIS




| EPOCHS | RMSE(ROOT MEAN SQUARE ERROR) |
|---|---|
| 1 | 38.71309192595 |
| 10 | 2.509939193725583 |
| 20 | 51.07455377109715 |

# CHAPTER 7

# CONCLUSION

There are various methods for predicting the stock market at the moment, however they are less accurate. We suggested a more accurate model that employs RNN and LSTM to forecast the trend in stock prices. In the buried layer of the network, LSTM introduces the memory cell, a computational unit that substitutes typical artificial neurons. The accuracy of prediction is improved in this work by increasing the Epochs and batch size. In the suggested method, we use test data to forecast, which results in more accurate outcomes using the test data. The proposed method is capable of stock market tracing and prediction, and the forecast will yield higher and more accurate outcomes. In our above model we are getting accurate results which will be more useful to stock analysts, Business analysts, Stock Market Investors.

# CHAPTER 8

# REFERENCES

1. Stock Price Prediction Using LSTM on the Indian Share Market by Achyut Ghosh,Soumik Bose1, Giridhar Maji, Narayan C. Debnath, Soumya Sen.

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

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. Durmagambetov currently works at CNTFI. He does research in Theory of Computation and Computing in Mathematics, Natural Science, Engineering and Medicine. Their current project is 'The Riemann Hypothesis-Millennium Prize Problems' - stock market predictions

7. Mariam Moukalled Wassim El-Hajj Mohamad Jaber Computer Science Department American University of Beirut - Automated Stock Price Prediction Using Machine Learning.

8. 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.

9. Lufuno Ronald Marwala A dissertation submitted to the Faculty of Engineering and the Built Environment, University of the Witwatersrand, Johannesburg, in fulfillment of the requirements for the degree of Master of Science in Engineering - Forecasting the Stock Market Index Using Artificial Intelligence Techniques.

10. Xiao-Yang Liu1 Hongyang Yang,Qian Chen4,Runjia ZhangLiuqing Yang Bowen Xiao Christina Dan Wang Electrical Engineering, 2Department of Statistics, 3 Computer Science, Columbia University, 3A4 Finance LLC., USA, Ion Media Networks, USA, Department of Computing, Imperial College, 6New York University (Shanghai) - A Deep Reinforcement Learning Library for Automated Stock Trading in Quantitative Finance

11. Pushpendu Ghosh, Ariel Neufeld, Jajati Keshari SahooDepartment of Computer Science & Information Systems, BITS Pilani K.K.Birla Goa campus, India bDivision of Mathematical Sciences, Nanyang Technological University, Singapore cDepartment of Mathematics, BITS Pilani K.K.Birla Goa campus, India - Forecasting directional movements of stock prices for intraday trading using LSTM and random forests.

12. Xiao Ding, Kuo Liao, Ting Liu, Zhongyang Li, Junwen Duan Research Center for Social Computing and Information Retrieval Harbin Institute of Technology, China - Event Representation Learning Enhanced with External Commonsense Knowledge.

13. 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.

14. Hyeong Kyu Choi, B.A Student Dept. of Business Administration Korea University Seoul, Korea = Stock Price Correlation Coefficient Prediction with ARIMA-LSTM Hybrid Model.

15. M. Nabipour Faculty of Mechanical Engineering, Tarbiat Modares University, 14115-143 Tehran, Iran; Mojtaba.nabipour@modares.ac.ir - Deep Learning for Stock Market prediction.

16. 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.

17. https://builtin.com/data-science/recurrent-neural-networks-and-lstm

18. https://colah.github.io/posts/2015-08-Understanding-LSTMs/