MACHINE LEARNING PROJECT (2021-22)

Stock Price Prediction

REPORT

Institute of Engineering & Technology

Submitted by

Vishal Gaur(181500803)

Kanhaiya Sharma(181500306)

Supervised By: -

Mr. Neeraj Gupta

Department of Computer Engineering and Applications
GLA UNIVERSITY MATHURA
17 KM. STONE NH-2,MATHURA-DELHI ROAD,P.O.-CHAUMUHA
MATHURA-281406

Declaration

I Hereby declare that the work which is being presented in the Machine Learning Project **"Stock Price Prediction",** in partial fulfillment of the requirements for Machine Learning Project Lab is an authentic record of my own work carried under the supervision of **Mr. Neeraj Gupta.**

Vishal Gaur

Kanhaiya Sharma

Department of Computer Engineering and Applications
GLA UNIVERSITY MATHURA
17 KM. STONE NH-2,MATHURA-DELHI ROAD,P.O.-CHAUMUHA
MATHURA-281406

Certificate

This is to certify that the project entitled – **STOCK PRICE PREDICTION** carried out in Machine Learning Project Lab is a bonafide work done by Vishal Gaur (181500803) and Kanhaiya Sharma (181500306) and is submitted in partial fulfillment of the requirements for the award of the degree Bachelor of Technology (ComputerScience & Engineering).

Signature of Supervisor:

Name of Supervisor: Mr. Neeraj Gupta

Date: 27th, June 2021

Department of Computer Engineering and Applications
GLA UNIVERSITY MATHURA
17 KM. STONE NH-2,MATHURA-DELHI ROAD,P.O.-CHAUMUHA
MATHURA-281406

ACKNOWLEDGEMENT

It gives us a great sense of pleasure to present the report of the B.Tech Machine Learning Project undertaken during B. Tech. Third Year. This project in itself is an acknowledgement to the inspiration, drive and technical assistance contributed to it by many individuals. This project would never have seen the light of the day without the help and guidance that we have received.

Our heartiest thanks to Dr. (Prof).Anand Singh Jalal, Head of Dept., Department of CEA for providing us with an encouraging platform to develop this project, which thus helped us in shaping our abilities towards a constructive goal.

We owe a special debt of gratitude to Mr. Neeraj Gupta, Machine Learning faculty, for his constant support and guidance throughout the course of our work. His sincerity, thoroughness and perseverance have been a constant source of inspiration for us. He has showered us with all his extensively experienced ideas and insightful comments at virtually all stages of the project & has also taught us about the latest industry-oriented technologies.

We also do not like to miss the opportunity to acknowledge the contribution of all faculty members of the department for their kind guidance and cooperation during the development of our project. Last but not the least, we acknowledge our friends for their contribution in the completion of the project.

Vishal Gaur

Kanhaiya Sharma

Department of Computer Engineering and Applications
GLA UNIVERSITY MATHURA
17 KM. STONE NH-2,MATHURA-DELHI ROAD,P.O.-CHAUMUHA
MATHURA-281406

Abstract

In the finance world stock trading is one of the most important activities. Stock market prediction is an act of trying to determine the future value of a stock or other financial instrument traded on a financial exchange. This paper explains the prediction of a stock using Machine Learning. The technical and fundamental or the time series analysis is used by most of the stockbrokers while making the stock predictions. The programming language used to predict the stock market using machine learning is Python. In this paper we propose a Machine Learning (ML) approach that will be trained from the available stocks data and gain intelligence and then 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.

Department of Computer Engineering and Applications
GLA UNIVERSITY MATHURA
17 KM. STONE NH-2,MATHURA-DELHI ROAD,P.O.-CHAUMUHA
MATHURA-281406

Table of Contents

Department of Computer Engineering and Applications
GLA UNIVERSITY MATHURA
17 KM. STONE NH-2,MATHURA-DELHI ROAD,P.O.-CHAUMUHA
MATHURA-281406

Introduction

Sequence prediction problems have been around for a long time. They are considered as one of the hardest problems to solve in the data science industry. These include a wide range of problems; from predicting sales to finding patterns in stock markets' data, from understanding movie plots to recognizing your way of speech, from language translations to predicting your next word on your iPhone's keyboard.

With the recent breakthroughs that have been happening in data science, it is found that for almost all of these sequence prediction problems, Long Short Term Memory networks, a.k.a LSTMs, have been observed as the most effective solution.

LSTMs have an edge over conventional feed-forward neural networks and RNN in many ways. This is because of their property of selectively remembering patterns for long durations of time. The purpose of this article is to explain LSTM and enable you to use it in real life problems.

Department of Computer Engineering and Applications
GLA UNIVERSITY MATHURA
17 KM. STONE NH-2,MATHURA-DELHI ROAD,P.O.-CHAUMUHA
MATHURA-281406

EXISTING SYSTEM

In the existing system, we tend to propose that a company's performance, in terms of its stock worth movement, is foreseen by internal communication patterns. to get early warning signals, we tend to believe that it's vital for patterns in company communication networks to be detected earlier for the pre- diction of sirius stock worth movement to avoid attainable adversities that an organization could face within the securities market in order that stakeholders' interests is protected the maximum amount as attainable. Despite the potential importance of such data regarding corporate communication, very little work has been tied to this vital direction. We attempt to bridge these research gaps by employing a data-mining method to examine the linkage between a firm's communication data and its share price. As Enron Corporation's e-mail messages constitute the only corpus available to the public, we make use of Enron's email corpus as the training and testing data for our proposed algorithm.

DISADVANTAGE

- However accuracy would decrease when setting more levels of stock market movement.
- The average of prediction accuracies using Decision Tree as the classifier are 43.44%, 31.92%, and 12.06% for “two levels,” “three levels,” and “five levels,” respectively.
- These results indicate that the stock price is unpredictable when a traditional classifier is used.

Department of Computer Engineering and Applications
GLA UNIVERSITY MATHURA
17 KM. STONE NH-2,MATHURA-DELHI ROAD,P.O.-CHAUMUHA
MATHURA-281406

PROPOSED SYSTEM

Accuracy plays an important role in stock market prediction. Although many algorithms are available for this purpose, selecting the most accurate one continues to be the fundamental task in getting the best results. In order to achieve this, in this paper we have compared and analysed the performance of various available algorithms such as Logistic regression, SVM, Random Forest, etc. This involves training the algorithms, executing them, getting the results, comparing various performance parameters of these algorithms and finally obtaining the most accurate one.

Department of Computer Engineering and Applications
GLA UNIVERSITY MATHURA
17 KM. STONE NH-2,MATHURA-DELHI ROAD,P.O.-CHAUMUHA
MATHURA-281406

ADVANTAGE

- The successful prediction will maximize the benefit of the customer.
- In this paper we have discussed various algorithms to predict the same.
- In this paper we used stock data of five companies from the Huge Stock market dataset consisting of data ranging from 2011 to 2017 to train different machine learning algorithms.
- Hence we compared the accuracy of different machine learning algorithms

Department of Computer Engineering and Applications
GLA UNIVERSITY MATHURA
17 KM. STONE NH-2,MATHURA-DELHI ROAD,P.O.-CHAUMUHA
MATHURA-281406

PERFORMANCE ANALYSIS OF ALGORITHMS

In this report we compared machine learning algorithms using stock market dataset. We performed experiments with various algorithms on stock market dataset and observed the mean square error to predict accuracy using four algorithms namely Logistic regression, SVM and Random Forest. We used Python libraries such as pandas, numpy to load the dataset and to perform mathematical calculations respectively and we used sklearn to model different machine learning algorithms. We used 0.20 of our whole dataset to test our model. We can conclude that SVM, Random Forest and Logistic Regression have better accuracies. We have plotted three of five best performing algorithms' mean square error for each company to compare performance of each algorithm. SVM, Random Forest and Logistic Regression yielded best results. The result can further be enhanced by processing data properly as there are large fluctuations in the stock market.

Table -1: Sample Table format

Sr. No	Attribute	Description
1	Date	Date of the stock value
2	High	Highest point of the price of a stock at the exchange
3	Low	Lowest point of the price of a stock at the exchange
4	Open	Opening price of a stock at the exchange
5	Close	Closing price of a stock at the exchange
6	Volume	Volume of stock is average of total traded stocks at the exchange over a period of time.

PROGRAMMING LANGUAGE

- Python

TOOL

- Python IDLE
- Google Colab

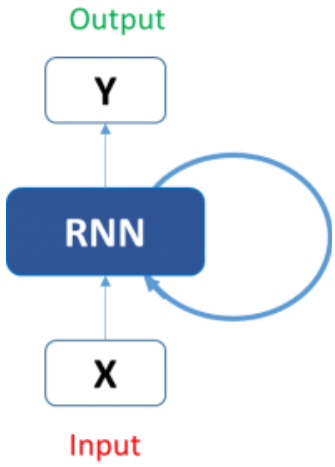
Department of Computer Engineering and Applications
GLA UNIVERSITY MATHURA
17 KM. STONE NH-2,MATHURA-DELHI ROAD,P.O.-CHAUMUHA
MATHURA-281406

1. Flashback: A look into Recurrent Neural Networks (RNN)

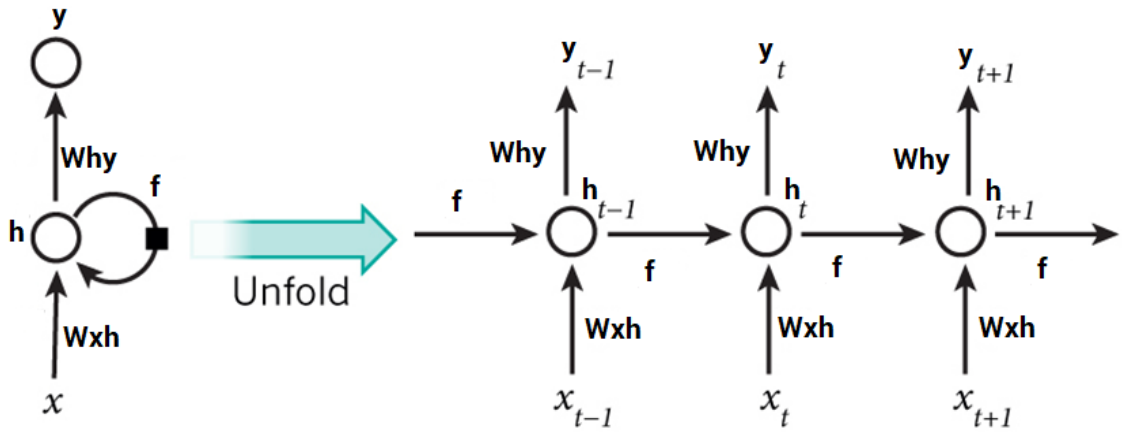
Take an example of sequential data, which can be the stock market's data for a particular stock. A simple machine learning model or an Artificial Neural Network may learn to predict the stock prices based on a number of features: the volume of the stock, the opening value etc. While the price of the stock depends on these features, it is also largely dependent on the stock values in the previous days. In fact for a trader, these values in the previous days (or the trend) is one major deciding factor for predictions.

In the conventional feed-forward neural networks, all test cases are considered to be independent. That is when fitting the model for a particular day, there is no consideration for the stock prices on the previous days.

This dependency on time is achieved via Recurrent Neural Networks. A typical RNN looks like:



This may be intimidating at first sight, but once unfolded, it looks a lot simpler:



Now it is easier for us to visualize how these networks are considering the trend of stock prices, before predicting the stock prices for today. Here every prediction at time t (h_t) is dependent on all previous predictions and the information learned from them.

RNNs can solve our purpose of sequence handling to a great extent but not entirely. We want our computers to be good enough to write Shakespearean sonnets. Now RNNs are great when it comes to short contexts, but in order to be able to build a story and remember it, we need our models to be able to understand and remember the context behind the sequences, just like a human brain. This is not possible with a simple RNN.

2. Limitations of RNNs

Recurrent Neural Networks work just fine when we are dealing with short-term dependencies. That is when applied to problems like:

The colour of the sky is _____.

RNNs turn out to be quite effective. This is because this problem has nothing to do with the context of the statement. The RNN need not remember what was said before this, or what was its meaning, all they need to know is that in most cases the sky is blue. Thus the prediction would be:

The colour of the sky is blue.

However, vanilla RNNs fail to understand the context behind an input. Something that was said long before, cannot be recalled when making predictions in the present. Let's understand this as an example:

I spent 20 long years working for the under-privileged kids in Spain. I then moved to Africa.

.....

I can speak fluent _____.

Here, we can understand that since the author has worked in Spain for 20 years, it is very likely that he may possess a good command over Spanish. But, to make a proper prediction, the RNN needs to remember this context. The relevant information may be

separated from the point where it is needed, by a huge load of irrelevant data. This is where a Recurrent Neural Network fails!

The reason behind this is the problem of Vanishing Gradient. In order to understand this, you'll need to have some knowledge about how a feed-forward neural network learns. We know that for a conventional feed-forward neural network, the weight updating that is applied on a particular layer is a multiple of the learning rate, the error term from the previous layer and the input to that layer. Thus, the error term for a particular layer is somewhere a product of all previous layers' errors. When dealing with activation functions like the sigmoid function, the small values of its derivatives (occurring in the error function) get multiplied multiple times as we move towards the starting layers. As a result of this, the gradient almost vanishes as we move towards the starting layers, and it becomes difficult to train these layers.

A similar case is observed in Recurrent Neural Networks. RNN remembers things for just small durations of time, i.e. if we need the information after a small time it may be reproducible, but once a lot of words are fed in, this information gets lost somewhere. This issue can be resolved by applying a slightly tweaked version of RNNs – the Long Short-Term Memory Networks.

3. Improvement over RNN: LSTM (Long Short-Term Memory) Networks

When we arrange our calendar for the day, we prioritize our appointments right? If in case we need to make some space for anything important we know which meeting could be canceled to accommodate a possible meeting.

Turns out that an RNN doesn't do so. In order to add new information, it transforms

the existing information completely by applying a function. Because of this, the entire information is modified, on the whole, i. e. there is no consideration for *'important'* information and *'not so important'* information.

LSTMs on the other hand, make small modifications to the information by multiplications and additions. With LSTMs, the information flows through a mechanism known as cell states. This way, LSTMs can selectively remember or forget things. The information at a particular cell state has three different dependencies.

We'll visualize this with an example. Let's take the example of predicting stock prices for a particular stock. The stock price of today will depend upon:

1. The trend that the stock has been following in the previous days, maybe a downtrend or an uptrend.
2. The price of the stock on the previous day, because many traders compare the stock's previous day price before buying it.
3. The factors that can affect the price of the stock for today. This can be a new company policy that is being criticized widely, or a drop in the company's profit, or maybe an unexpected change in the senior leadership of the company.

These dependencies can be generalized to any problem as:

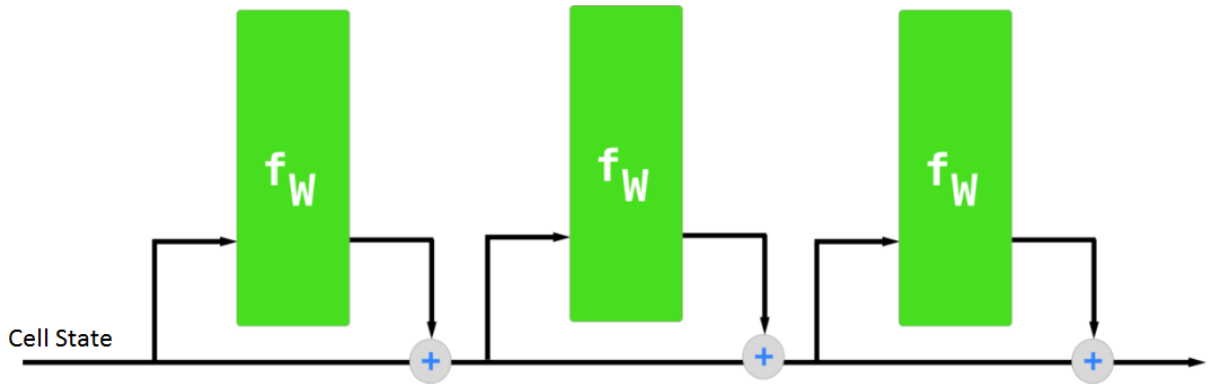
1. The previous cell state *(i.e. the information that was present in the memory after the previous time step)*
2. The previous hidden state *(i.e. this is the same as the output of the previous cell)*
3. The input at the current time step *(i.e. the new information that is being fed in at that moment)*

Another important feature of LSTM is its analogy with conveyor belts!

Industries use them to move products around for different processes. LSTMs use this mechanism to move information around.

We may have some addition, modification or removal of information as it flows through the different layers, just like a product may be molded, painted or packed while it is on a conveyor belt.

The following diagram explains the close relationship of LSTMs and conveyor belts.



Although this diagram is not even close to the actual architecture of an LSTM, it solves our purpose for now.

Just because of this property of LSTMs, where they do not manipulate the entire information but rather modify them slightly, they are able to *forget* and *remember* things selectively.

4. Architecture of LSTMs

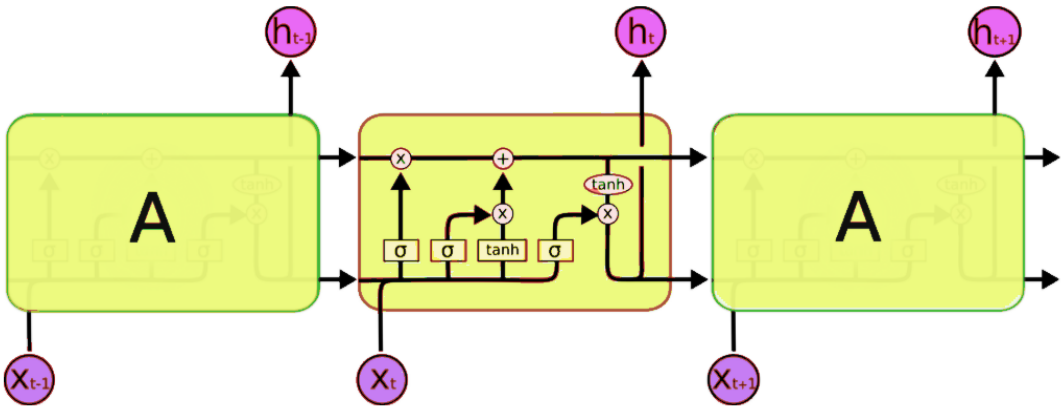
The functioning of LSTM can be visualized by understanding the functioning of a news channel's team covering a murder story. Now, a news story is built around facts, evidence and statements of many people. Whenever a new event occurs you take either of the three steps.

Let's say, we were assuming that the murder was done by 'poisoning' the victim, but the autopsy report that just came in said that the cause of death was 'an impact on the head'. Being a part of this news team what do you do? You immediately forget the previous cause of death and all stories that were woven around this fact.

What, if an entirely new suspect is introduced into the picture. A person who had grudges with the victim and could be the murderer? You input this information into your news feed, right?

Now all these broken pieces of information cannot be served on mainstream media. So, after a certain time interval, you need to summarize this information and output the relevant things to your audience. Maybe in the form of "*XYZ turns out to be the prime suspect.*".

Now let's get into the details of the architecture of LSTM network:



Now, this is nowhere close to the simplified version which we saw before, but let me walk you through it. A typical LSTM network is composed of different memory blocks called cells.

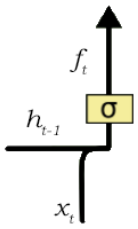
(the rectangles that we see in the image). There are two states that are being transferred to the next cell; the cell state and the hidden state. The memory blocks are responsible for remembering things and manipulations to this memory are done through three major mechanisms, called gates. Each of them is being discussed below.

4.1 Forget Gate

Taking the example of a text prediction problem. Let's assume an LSTM is fed in, the following sentence:

Bob is a nice person. Dan on the other hand is evil.

As soon as the first full stop after "*person*" is encountered, the forget gate realizes that there may be a change of context in the next sentence. As a result of this, the *subject* of the sentence is *forgotten* and the place for the subject is vacated. And when we start speaking about "*Dan*" this position of the subject is allocated to "*Dan*". This process of forgetting the subject is brought about by the 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. The given inputs are multiplied by the weight matrices and a bias is added. Following this, the sigmoid function is applied to this value. The sigmoid function outputs a vector, with values ranging from 0 to 1, corresponding to each number in the cell state. Basically, the sigmoid function is responsible for deciding which values to keep and which to discard. If a '0' is output for a particular value in the cell state, it means that the forget gate wants the cell state to forget that piece of information completely. Similarly, a '1' means that the forget

gate wants to remember that entire piece of information. This vector output from the sigmoid function is multiplied to the cell state.

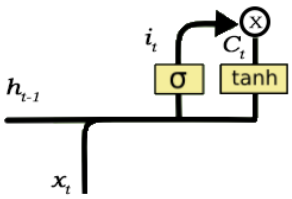
4.2 Input Gate

Okay, let's take another example where the LSTM is analyzing a sentence:

Bob knows swimming. He told me over the phone that he had served the navy for 4 long years.

Now the important information here is that "Bob" knows swimming and that he has served the Navy for four years. This can be added to the cell state, however, the fact that he told all this over the phone is a less important fact and can be ignored. This process of adding some new information can be done via the input gate.

Here is its structure:



The input gate is responsible for the addition of information to the cell state. This addition of information is basically a three-step process as seen from the diagram above.

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 h_t-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.

Once this three-step process is done with, we ensure that only that information is added to the cell state that is *important* and is not *redundant.*

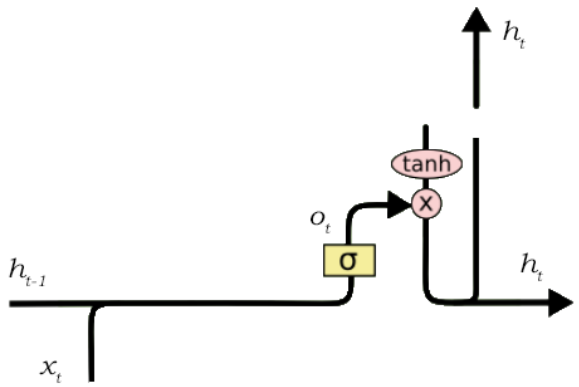
4.3 Output Gate

Not all information that runs along the cell state is fit for being output at a certain time. We'll visualize this with an example:

Bob fought single handedly with the enemy and died for his country. For his contributions brave _____.

In this phrase, there could be a number of options for the empty space. But we know that the current input of *'brave',* is an adjective that is used to describe a noun. Thus, whatever word follows, has a strong tendency of being a noun. And thus, Bob could be an apt output.

This job of selecting useful information from the current cell state and showing it out as an output is done via the output gate. Here is its structure:



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

1. Creating a vector after applying tanh function to the cell state, thereby scaling the values to the range -1 to +1.
2. 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.
3. Multiplying the value of this regulatory filter to the vector created in step 1, and sending it out as an output and also to the hidden state of the next cell.

The filter in the above example will make sure that it diminishes all other values but

'Bob'. Thus the filter needs to be built on the input and hidden state values and be applied on the cell state vector.

5. Text generation using LSTMs

We have had enough of theoretical concepts and functioning of LSTMs. Now we would be trying to build a model that can predict some *n* number of characters after the original text of Macbeth. Most of the classical texts are no longer protected under copyright. We will use the library Keras, which is a high-level API for neural networks and works on top of TensorFlow or Theano. So make sure that before diving into this code you have Keras installed and functional.

Department of Computer Engineering and Applications
GLA UNIVERSITY MATHURA
17 KM. STONE NH-2,MATHURA-DELHI ROAD,P.O.-CHAUMUHA
MATHURA-281406

BIBLIOGRAPHY& REFERENCES

To develop this project for Stock Price Prediction, we used python and used google colab for executing our code. We take some knowledge towards automation system from some books that are given below:

References:

[1]. https://data-flair.training.com

[2]. https://www.analyticsvidhya.com

[3]. www.stackoverflow.com