

# Visvesvaraya Technological University

Jnana Sangama, Belgaum – 590014, Karnataka



A PROJECT REPORT

ON

## Stock Market Prediction Using Machine Learning

Submitted in partial fulfilment of the requirements for the award of the  
Degree of **Bachelor of Engineering in Computer Science & Engineering**  
for the academic year 2017-2018

**Submitted By**

**Akshay R Purohit**

**1ST14CS010**

**Aravind B**

**1ST14CS023**

**Arun Kumar R**

**1ST14CS025**

**Ashok S**

**1ST14CS027**

**Under the guidance of**

**Dr. T. John Peter**

H.O.D., Department of CSE



Department of Computer Science & Engineering

**Sambhram Institute of Technology**

**Lakshmipura Cross, M S Palya, Jalahalli East, Bengaluru – 560097**

**Sambhram Institute of Technology**  
**Lakshmipura Cross, M S Palya, Jalahalli East, Bengaluru – 560097**



Department of Computer Science & Engineering

**CERTIFICATE**

This is to certify that the project work entitled “**Stock Market Prediction Using Machine Learning**” has been carried out by **AKSHAY R P PUROHIT** (1ST14CS010), **ARAVIND B** (1ST14CS023), **ARUN KUMAR R** (1ST14CS025), **ASHOK S** (1ST14CS027), bonafide students of **Sambhram Institute of Technology** in partial fulfillment for the award of Bachelor of Engineering in Computer Science and Engineering of the **Visvesvaraya Technological University, Belgaum** during the year **2017-2018**. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the department library. The project report has been approved as it satisfies the academic requirements with respect to its work prescribed for the said degree.

Signature of the Guide

Signature of the HOD

Signature of the Principal

**Dr. T. John Peter**  
**Head, Dept. of CSE**  
Sambhram Institute of  
Technology

**Dr. T. John Peter**  
**Head, Dept. of CSE**  
Sambhram Institute  
of Technology

**Dr. H.G. Chandrakanth**  
**Principal**  
Sambhram Institute  
of Technology

**External Viva:**

**Name of the examiners**

**Signature with date**

**1**

**2.**

# ACKNOWLEDGEMENT

While presenting this Project on **Stock Market Prediction Using Machine Learning**, We felt that it is our duty to acknowledge the help rendered to us by various persons.

Firstly, we are grateful to our institution **Sambhram Institute of Technology** for providing us a congenial atmosphere to carry out the seminar successfully.

We would like to express our heartfelt gratitude to **Dr. H.G. Chandrakanth**, Principal, Bangalore, for extending his support.

We are very grateful to our guide, **Dr. T. John Peter**, **Head of Department**, Computer Science, for his able guidance and valuable advice at every stage of our project which helped us in the successful completion of my project.

We are also indebted to our parents and friends for their continued moral and material support throughout the course of project and helping us in finalize the presentation.

Our heartfelt thanks to all those have contributed bits, bytes and words to accomplish this Project

|                         |                   |
|-------------------------|-------------------|
| <b>Akshay R Purohit</b> | <b>1ST14CS010</b> |
| <b>Aravind B</b>        | <b>1ST14CS023</b> |
| <b>Arun Kumar R</b>     | <b>1ST14CS025</b> |
| <b>Ashok S</b>          | <b>1ST14CS027</b> |

## **ABSTRACT**

Time series forecasting has been widely used to determine the future prices of stock, and the analysis and modelling of finance time series importantly guide investors' decisions and trades. In addition, in a dynamic environment such as the stock market, the non-linearity of the time series is pronounced, immediately affecting the efficacy of stock price forecasts. Thus, this work proposes an intelligent time series prediction system that uses sliding-window metaheuristic optimization for the purpose of predicting the stock prices of multiple companies taken at random. It may be of great interest to home brokers who do not possess sufficient knowledge to invest in such companies. The system has a graphical user interface and functions as a stand-alone application. The developed hybrid system exhibited outstanding prediction performance and it improves overall profit for investment performance. The proposed model is a promising predictive technique for highly non-linear time series, whose patterns are difficult to capture by traditional models.

# TABLE OF CONTENTS

| <b>Chapters</b>                                | <b>Page No.</b> |
|------------------------------------------------|-----------------|
| <b>1. Introduction</b>                         |                 |
| 1.1    Stock Market Prediction                 | 1               |
| 1.2    Exiting System                          | 2               |
| 1.3    Disadvantages of the Existing System    | 2               |
| 1.4    Proposed System                         | 3               |
| 1.5    Advantages of the Proposed System       | 3               |
| <b>2. Literature Survey</b>                    |                 |
| 2.1    Time-series forecasting                 | 4               |
| 2.2    Support Vector Regression               | 4               |
| 2.3    Least Squares Support Vector Regression | 4               |
| 2.4    Hyperparameter Optimization             | 5               |
| 2.5    Firefly Algorithm                       | 5               |
| <b>3. System Requirement Specification</b>     |                 |
| 3.1    Non-functional requirements             | 6               |
| 3.2    Specific Requirements                   | 7               |
| 3.3    Software requirements                   | 7               |
| 3.4    Hardware requirements                   | 8               |
| <b>4. System Design</b>                        |                 |
| 4.1    Dataflow Diagram                        | 9               |
| 4.2    Use Case Diagram                        | 10              |
| 4.3    Sequence Diagram                        | 11              |
| 4.4    Flowchart                               | 13              |
| <b>5. Implementation</b>                       |                 |
| 5.1    Modules                                 | 14              |
| 5.2    Overall Implementation                  | 16              |

|                                |    |
|--------------------------------|----|
| <b>6. Testing</b>              |    |
| 6.1 Test Objectives            | 17 |
| 6.2 Testing Principles         | 17 |
| 6.3 Testing Design             | 17 |
| 6.4 Testing Strategies         | 18 |
| 6.5 Levels of Testing          | 19 |
| <b>7. Performance Analysis</b> |    |
| 7.1 IBM Stock Prediction       | 24 |
| 7.2 CISCO Stock Prediction     | 26 |
| 7.3 Multiple Stock Performance | 28 |
| 7.4 Overall System Performance | 29 |
| <b>8. Results</b>              | 30 |
| <b>9. Screenshots</b>          | 33 |
| <b>10. Future Enhancement</b>  |    |
| 10.1 Limitations               | 35 |
| 10.2 Future Enhancement        | 35 |
| <b>11. Conclusion</b>          | 36 |
| <b>References</b>              | 37 |

# LIST OF FIGURES AND TABLES

## FIGURES

## Page No.

|     |                                             |    |
|-----|---------------------------------------------|----|
| 4.1 | Dataflow diagram of proposed system         | 9  |
| 4.2 | Use case diagram of the proposed system     | 10 |
| 4.3 | Sequence diagram of the proposed system     | 12 |
| 4.4 | Flowchart of the proposed system            | 13 |
| 7.1 | IBM Stock Prediction for 90 days            | 24 |
| 7.2 | IBM: Day wise accuracy                      | 25 |
| 7.3 | IBM: Predictive Performance                 | 26 |
| 7.4 | Cisco Stock Prediction for 90 days          | 26 |
| 7.5 | Cisco Day wise accuracy                     | 27 |
| 7.6 | Cisco: Predictive Performance               | 27 |
| 7.7 | Individual Predictive Performance           | 28 |
| 7.8 | Overall System Performance                  | 29 |
| 8.1 | IBM Stock Prediction for 10 days            | 32 |
| 9.1 | Interface of the system                     | 33 |
| 9.2 | The screenshot of system                    | 34 |
| 9.3 | Predictive performance of the entire system | 34 |

## TABLES

|     |                                          |    |
|-----|------------------------------------------|----|
| 6.1 | Unit Test Case 1                         | 20 |
| 6.2 | Unit Test Case 2                         | 20 |
| 6.3 | Functional testing items                 | 21 |
| 7.1 | Input provided for performance analysis  | 24 |
| 8.1 | Requesting input for prediction          | 30 |
| 8.2 | Predicted closing stock prices           | 30 |
| 8.3 | Basic summary statistics                 | 31 |
| 8.4 | Results with actual and predicted prices | 31 |

# Chapter 1

## INTRODUCTION

### 1.1 Stock Market Prediction

Financial markets are highly volatile and generate huge amounts of data daily. Investment is a commitment of money or other resources to obtain benefits in the future. Stock is one type of securities. It is the most popular financial market instrument and its value changes quickly. It can be defined as a sign of capital participation by a person or an enterprise in a company or a limited liability company. The stock market provides opportunities for brokers and companies to make investments on neutral ground [1].

Stock prices are predicted to determine the future value of companies' stock or other financial instruments that are marketed on financial exchanges. However, the stock market is characterized by nonlinearities, discontinuities, and high-frequency multi-polynomial components because it interacts with many factors such as political events, general economic conditions, and traders' expectations. Therefore, making precise predictions of stock values are challenging [2].

Investors can buy stocks that are related to the construction firms that design infrastructure projects, hire contractors and handle paperwork, and decision-makers of construction firms can buy stocks from other companies. When the direction of the market is successfully predicted, investors may be better guided and monetary rewards will be substantial. The challenge in today's environment, where bad news can always be heard, is to forecast proactively, rather than reactively. Therefore, construction corporations are trying to predict stock prices which is important to be considered on a financial exchange, against sudden drops in the market.

Time series forecasting consists in a research area designed to solve various problems, mainly in the financial area. It is noteworthy that this area typically uses tools that assist in planning and making decisions to minimize investment risks. This objective is obvious when one wants to analyze financial markets and, for this reason, it is necessary to assure a good accuracy in forecasting tasks [3].



## 1.2 Existing System

Time series forecasting consists in a research area designed to solve various problems, mainly in the financial area. It is noteworthy that this area typically uses tools that assist in planning and making decisions to minimize investment risks. This objective is obvious when one wants to analyze financial markets and, for this reason, it is necessary to assure a good accuracy in forecasting tasks. [3]

Machine learning (ML) is coming into its own that can play a key in a wide range of critical applications. In machine learning, support vector machines (SVMs) have many advanced features that are reflected in their good generalization capacity and fast computation. They are also not very sensitive to assumptions about error terms and they can tolerate noise and chaotic components. Notably, SVMs are increasingly used in materials science, the design of engineering systems and financial risk prediction. [1]

Also, most methods that are in use are only applicable to a small portion of stock markets and usually such models do not generalize well to all stocks. Additionally, existing libraries are highly efficient in obtaining the optimal hyperparameters to be used in LSSVM and other algorithms.

## 1.3 Disadvantages of the Existing System

Since time series data can be formulated by regression analysis, LSSVR is very efficient when applied to the issue at hand. However, the efficacy of LSSVR strongly depends on its tuning hyperparameters, which are the regularization parameter and the kernel function. Inappropriate settings of these parameters may lead to significantly poor performance of the model. Therefore, the evaluation of such hyperparameters is a real-world optimization problem. [4]

Because the performance of SVR-based models strongly depends on the setting of its hyperparameters, they used to be set in advance based on the experience of practitioners, by trial-and-error, or using a grid search algorithm. Thus, finding the optimal values of regularization and kernel function parameters for SVR-based models is an important and time-consuming step. Therefore, a means of automatically finding the hyperparameters of SVR, while ensuring its generalization performance, is required. [5]

## 1.4 Proposed System

Decision to buy or sell a stock is very complicated since many factors can affect stock price. This work presents a novel approach, based on least squares support vector regression (LSSVR), to constructing a stock price forecasting expert system, with the aim of improving forecasting accuracy. The intelligent time series prediction system that uses sliding-window metaheuristic optimization is a graphical user interface that can be run as a stand-alone application. The system makes the prediction of stock market values simpler, involving fewer computations, than that using the other method that was mentioned above [1].

Additionally, the proposed system automatically fetches the latest stock data for any given company and date range.

## 1.5 Advantages of the Proposed System

To evaluate the proposed approach, it was applied to five datasets for stocks in Taiwan, and three other stock datasets that have been used in other papers.

- Firstly, to generalize the application of the proposed system, our work uses the proposed system to estimate other stocks in similar emerging markets and mature markets, such as Vietnam, Indonesia, China, Japan, Hong Kong, Korea, Singapore, Europe, USA and India.
- Secondly, the system can be extended to analyze multivariate time series data and import raw dataset directly.
- Thirdly, profit can be maximized even when the construction corporate stock market is bullish. Finally, the development of a web-based application has been considered to improve the user-friendliness and usability of the expert system.

In the proposed system, we import the data directly online using APIs, and since the LSSVM is standard algorithm, we make use of built-in functions and pass parameters directly and obtain the model with the help of the sliding-window method.

## Chapter 2

# LITERATURE SURVEY

Literature review is a text of a scholarly paper, which includes the current knowledge including substantive findings, as well as theoretical and methodological contributions to a particular topic. Literature reviews are secondary sources, and do not report new or original experimental work.

### 2.1 Time-series forecasting

According to Saini (2016), forecasting based on a time series represents a means of providing information and knowledge to support a subsequent decision [6]. Thus, the analysis of time series focuses on achieving dependency relationships among historical data. The two broad categories of forecasting models are linear and nonlinear. For many decades, traditional statistical forecasting models in financial engineering were linear. Some well-known statistical models can be used in time series forecasting [6].

### 2.2 Support Vector Regression

In machine learning, support vector regression (SVR) was developed by Vapnik *et al.* (1995) [7] and is a variant of the SVM. It is typically used to solve nonlinear regression problems by constructing the input-output mapping function. The least squares support vector regression (LSSVR) algorithm is a further development of SVR by Suykens (2001) [8] and involves equality instead of inequality constraints and works with a least squares objective function. The LSSVR approach considerably reduces computational complexity and increases efficiency compared to standard SVR. Hao et al. (2006) examined the feasibility of methods in stock composite index forecasting and improved the accuracy of parameter selection by SVR. They concluded that SVR has high prediction performance [9].

### 2.3 Least Squares Support Vector Regression

Some studies have demonstrated the superiority of LSSVR over standard support vector regression (SVR) for estimating product cost and energy utilization. LSSVR solves linear equations instead of a quadratic programming problem. It is preferred for large-scale regression problems that demand fast computation [8].

## 2.4 Hyperparameter Optimization

Since time series data can be formulated by regression analysis, LSSVR is very efficient when applied to the issue at hand. However, the efficacy of LSSVR strongly depends on its tuning hyperparameters, which are the regularization parameter and the kernel function. Inappropriate settings of these parameters may lead to significantly poor performance of the model. Therefore, the evaluation of such hyperparameters is a real-world optimization problem [4].

Optimization is one of the cornerstones of science and engineering. Recently, the field of nature-inspired optimization algorithms has grown incredibly fast. The algorithms are usually general-purpose and population-based. They are normally referred to as evolutionary algorithms because many of them are motivated by biological evolution. In a broad sense, evolutionary algorithms cover those that iteratively vary a group of solutions based on some nature-inspired operations.

## 2.5 Firefly Algorithm

The firefly algorithm (FA) [10], which is a nature-inspired metaheuristic method, has recently performed extremely well in solving various optimization problems such as stock price forecasting and electricity price prediction. The standard FA was developed by modeling the behavior of tropical fireflies. Notably, the smart firefly algorithm-based LSSVR has been demonstrated to be very effective in solving complex problems in civil engineering [11,12].

Recent research suggests that hybrid forecasting models can be usefully applied to the stock market's fluctuations, yielding satisfactory forecasting precision [2]. The authors used a hybrid model to capture the linear and non-linear characteristics of a stock price time series and confirmed that hybrid forecasting models are powerful tools for practitioners in management science. A review of the literature has indicated that enhancing the effectiveness capability of least squares support vector regression based on a nature-inspired metaheuristic optimization algorithm, such as the firefly algorithm is an unsolved problem in the field of stock price prediction

This work develops an intelligent time series prediction system using sliding-window metaheuristic optimization least squares support vector regression (LSSVR) to forecast the prices of construction corporate stocks.

## Chapter 3

# SYSTEM REQUIREMENT SPECIFICATION

To be used efficiently, all computer software needs certain hardware components or other software resources to be present on a computer. These prerequisites are known as (computer) system requirements and are often used as a guideline as opposed to an absolute rule. Most software defines two sets of system requirements: minimum and recommended. With increasing demand for higher processing power and resources in newer versions of software, system requirements tend to increase over time. Industry analysts suggest that this trend plays a bigger part in driving upgrades to existing computer systems than technological advancements.

Software requirements specification establishes the basis for an agreement between customers and contractors or suppliers on how the software product should function (in a market-driven project, these roles may be played by the marketing and development divisions). The software requirements specification lays out functional and non-functional requirements, and it may include a set of use cases that describe user interactions that the software must provide.

### 3.1 Non-functional requirements

Non-functional requirements are the functions offered by the system. It includes time constraints and constraints on the development process and standards. The non-functional requirements are as follows:

- **Speed:** The system should process the given input into output within appropriate time.
- **Ease of use:** The software should be user friendly. Then the customers can use easily, so it doesn't require much training time.
- **Reliability:** The rate of failures should be less then only the system is more reliable
- **Portability:** It should be easy to implement in any system.

### 3.1.1 Specific Requirements

The specific requirements are:

- **User Interfaces:** The external users are the clients. All the clients can use this software for indexing and searching.
- **Hardware Interfaces:** The external hardware interface used for indexing and searching is personal computers of the clients. The PC's may be laptops with wireless LAN as the internet connections provided will be wireless.
- **Software Interfaces:** The Operating Systems can be any version of Windows.
- **Performance Requirements:** The PC's used must be atleast Pentium 4 machines so that they can give optimum performance of the product.

### 3.2 Software requirements

Software requirements deal with defining software resource requirements and prerequisites that need to be installed on a computer to provide optimal functioning of an application.

These requirements or prerequisites are generally not included in the software installation package and need to be installed separately before the software is installed.

|                             |                          |
|-----------------------------|--------------------------|
| <b>Operating System</b>     | Window/ Ubuntu Linux     |
| <b>Programming Language</b> | R                        |
| <b>IDE/Editor</b>           | RStudio                  |
| <b>Required R Packages</b>  | Quandl, liquidSVM, caret |

### 3.3 Hardware requirements

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware. A hardware requirements list is often accompanied by a hardware compatibility list, especially in case of operating systems. An HCL lists tested, compatible, and sometimes incompatible hardware devices for a particular operating system or application. The following subsections discuss the various aspects of hardware requirements.

All computer operating systems are designed for a particular computer architecture. Most software applications are limited to particular operating systems running on particular architectures. Although architecture-independent operating systems and applications exist, most need to be recompiled to run on a new architecture.

The power of the central processing unit (CPU) is a fundamental system requirement for any software. Most software running on x86 architecture define processing power as the model and the clock speed of the CPU. Many other features of a CPU that influence its speed and power, like bus speed, cache, and MIPS are often ignored.

This definition of power is often erroneous, as AMD Athlon and Intel Pentium CPUs at similar clock speed often have different throughput speeds.

|                            |                                   |
|----------------------------|-----------------------------------|
| <b>System/Processor</b>    | Intel Core i3,5,7, 2.4-3.0 GHz    |
| <b>Hard Disk Space</b>     | 500 GB or more                    |
| <b>RAM</b>                 | 4 GB or more                      |
| <b>Internet Connection</b> | Required to auto-download dataset |

## CHAPTER 4

# SYSTEM DESIGN

System design is the process of defining the architecture, components, modules, interfaces and data for a system to satisfy specified requirements. One could see it as the application of systems theory to product development. There is some overlap with the disciplines of systems analysis, systems architecture and systems engineering. If the broader topic of product development "blends the perspective of marketing, design, and manufacturing into a single approach to product development," then design is the act of taking the marketing information and creating the design of the product to be manufactured. Systems design is therefore the process of defining and developing systems to satisfy specified requirements of the user.

### 4.1 Dataflow Diagram

A data flow diagram is a graphical representation of the "flow" of data through an information system, modeling its process aspects. Often, they are a preliminary step used to create an overview of the system which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design).

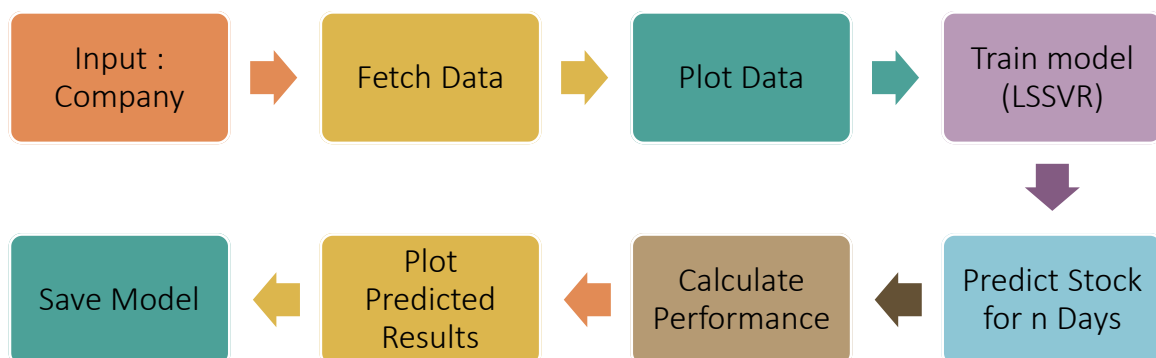


Figure 4.1 Dataflow diagram of the proposed system



The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of the input data to the system, various processing carried out on these data, and the output data is generated by the system.

After initialization the user is asked to select or input a company name using a Ticker. The system fetches the stock data online for a given input date range and plots the, on a graph. The data is sent to the train the LSSVR model. Stock Predictions for n-day ahead are performed and the model is saved for future use.

## 4.2 Use Case Diagram

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well.

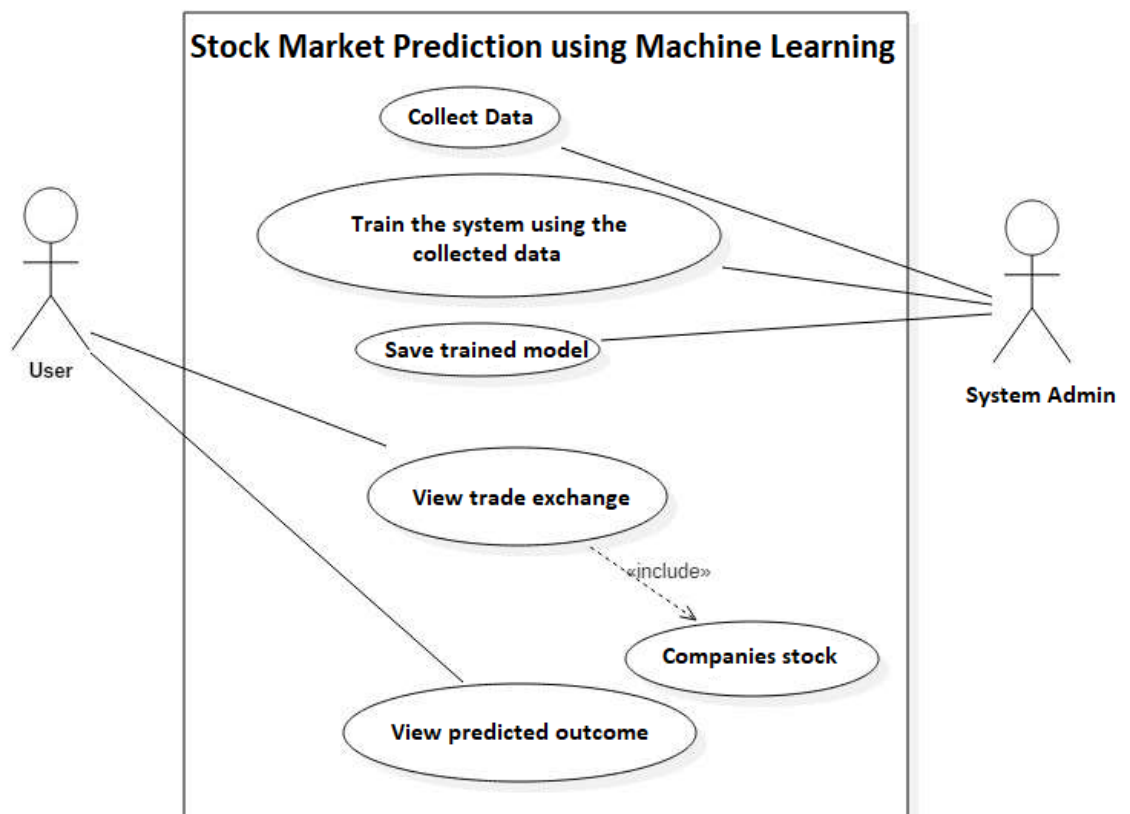


Figure 4.2 Use case diagram of the proposed system

The following steps are performed:

- Data is initially collected from online sources or the stock exchange, either by the system or the admin
- The data is then used to train the system
- Trained model is saved
- User views the trade exchange and stock of a company
- Using the model, closing prices are predicted

### **4.3 Sequence Diagram**

A sequence diagram in a UML is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams typically are associated with use case realizations in the Logical View of the system under development.

The steps performed are as follows:

- User visits the application
- Previously saved model is loaded
- User requests for a company's stock data
- He requests for prediction to be made
- The Stock Market Prediction System trains a model using the data from the database
- The model is saved for further use and closing price is predicted
- Result is displayed along with graph

The sequence diagram is shown below:

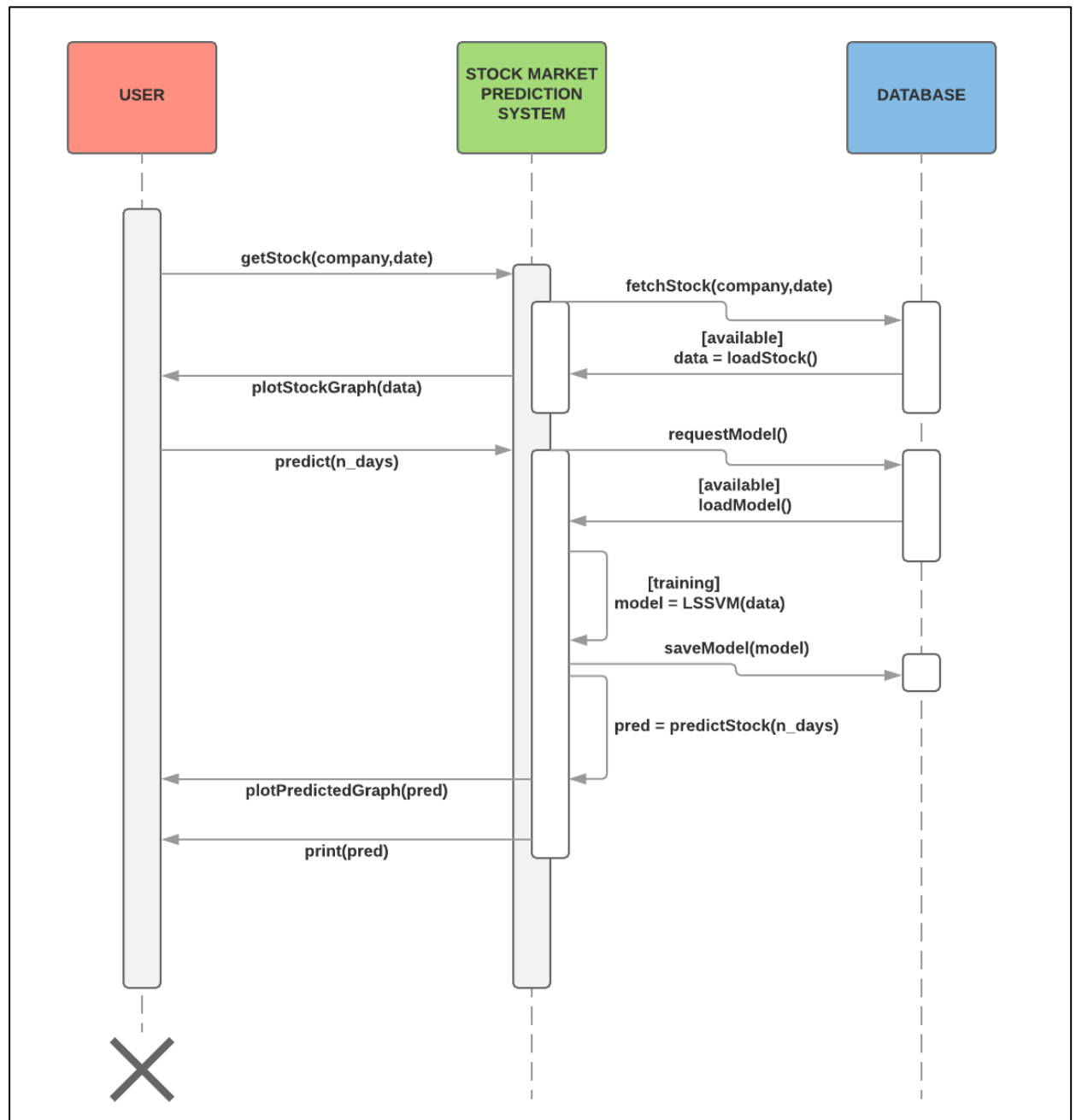


Figure 4.3 Sequence diagram of the proposed system

## 4.4 Flowchart

A flow chart is a graphical or symbolic representation of a process. Each step in the process is represented by a different symbol and contains a short description of the process step. The flow chart symbols are linked together with arrows showing the process flow direction.

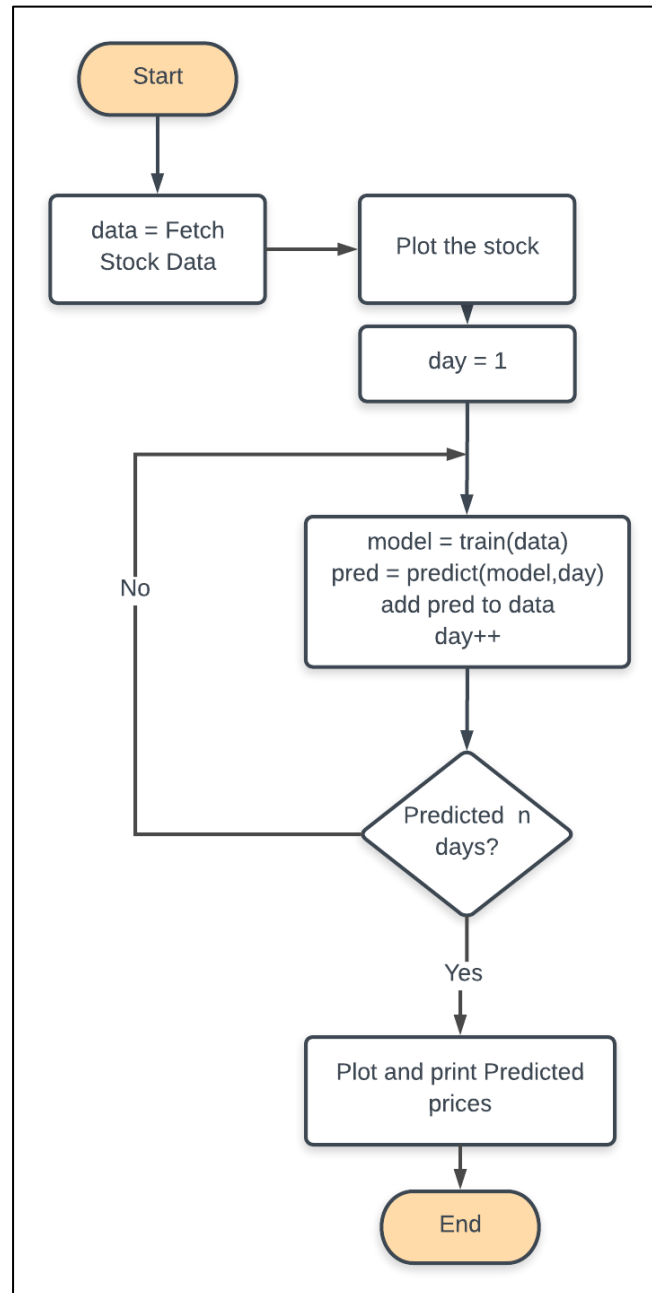


Figure 4.4 Flowchart of the proposed system

## CHAPTER 5

### IMPLEMENTATION

#### 5.1 Modules

The proposed system has mainly three modules that together form the main system implementation.

##### 5.1.1 Phase Space Reconstruction

In time series prediction, the time series are typically expanded into three or higher-dimensional space to exploit the information that is implicit in them. Selecting a suitable pairing of embedding dimension  $m$  (lag) and time delay  $\tau$  is very important for phase space reconstruction.

Consider a time series  $\bar{x} = \{x_1, x_2, x_3, \dots, x_n\}$ . The time-delay vectors can be reconstructed as follows, where  $X$  is the input matrix and  $Y$  is the corresponding output matrix. The output of the analysis is fed back to the input and future values are predicted from previous values in the time series. [1]

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} \xrightarrow{m} X = \begin{bmatrix} x_1 & x_2 & \dots & x_{m-1} & x_m \\ x_2 & x_3 & \dots & x_m & x_{m+1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{N-m-\tau+1} & x_{N-m-\tau+2} & \dots & x_{N-\tau-1} & x_{N-\tau} \end{bmatrix}, \quad Y = \begin{bmatrix} x_{m+\tau} \\ x_{m+1+\tau} \\ \vdots \\ x_N \end{bmatrix} \quad (5.1)$$

##### 5.1.2 Sliding-window method

As suggest in [1], the learning dataset used in this study was collected within a sliding-window. Fig. 1 depicts the sliding-window and phase space construction. Since the forecast is one step ahead (hence the term, “one-step ahead forecasting”), the forecast horizon is 1. In the first validation, the working window includes  $p$  historical observations  $(x_1, x_2, x_3, \dots, x_p)$  which are used to forecast the next value  $x_{p+1}$ . In the second validation, the oldest value  $x_1$  is removed from the window and the latest value  $x_{p+1}$  is added, keeping the length of the sliding window constant at  $p$ . The next forecast value will be  $x_{p+2}$ . The window continues to slide until the end of the dataset is reached. If the number of observations is  $N$ , then the total number of validations is  $(N-p)$ .

The algorithm for the sliding window is given as follows:

**Algorithm : slidingWindow**  
**Input** : *data* [stock data]  
**Output** : A data frame of a lagged stock data

1.  $LAG \leftarrow 1$
2.  $y \leftarrow$  remove first LAG rows from data
3. reset row indices of  $y$
4.  $x \leftarrow$  remove last LAG rows from data
5.  $train \leftarrow$  merge  $(x,y)$  into a dataframe
6. rename column name to  $x$  and  $y$
7. **return**  $train$

Algorithm 5.1 Sliding window algorithm

The above algorithm implemented as code in the R language is as follows:

```
slidingWindow = function(data)
{
  y = data[-(1:LAG),]
  rownames(y) = NULL
  x= data[1:(DATA_SIZE-LAG),]
  train =
data.frame(x[,2],y[,2])
  colnames(train) = c("x","y")
  return(train)
}
```

Snippet 5.1 Sliding window code

### 5.1.3 Least Squares Support Vector Regression

The LSSVR approach proposed by Suykens *et al.* (2002) [8] is a well-developed ML technique with many advanced features that support a high generalization capacity and fast computation. The LSSVR training process entails the use of a least squares cost function to obtain a linear set of equations in a dual space to minimize the computational cost. Accordingly, iterative methods, such as the conjugate gradient method are typically used to derive a solution by efficiently solving a set of linear equations. To reduce the computational burden of the LSSVR for function estimation, the regression model in this study uses a quadratic loss function.

The least squares version of the SVM classifier is obtained by reformulating the minimization problem as:

$$\min J_2(w, b, e) = \frac{\mu}{2} w^T w + \frac{\zeta}{2} \sum_{i=1}^N e_{c,i}^2, \quad (5.2)$$

For the kernel function  $K(x, x_i)$  one typically has the Radial Basis Function:

$$K(x, x_i) = \exp\left(-\|x - x_i\|^2 / \sigma^2\right), \quad (5.3)$$

The LSSVR involves equality instead of inequality constraints and works with a least squares objective function. The LSSVR approach considerably reduces computational complexity and increases efficiency compared to standard SVM. LSSVR solves linear equations instead of a quadratic programming problem.

## 5.2 Implementation

**Algorithm : StockPrediction**

**Input** : *COMP, D\_RANGE, N\_PRED* [company, date range, n-day predictions]

**Output** : A vector of predicted prices and graph, *RESULTS*

1. *data*  $\leftarrow$  fetch stock for *COMP* in date range *D\_RANGE*
2. **plot**(*data*)
3. *train\_data*  $\leftarrow$  *slidingWindow*(*data*)  
*//perform sliding window operation on data*
4. *RESULTS*  $\leftarrow$  0  
*//set accuracy vector to zeroes*
5. **for each** day in *N\_PRED* :  
*//pass the training data to LSSVM*
  1. *model*  $\leftarrow$  **LSSVM**(*train\_data*)  
*//predict the price given model and day*
  2. *pred*  $\leftarrow$  **predict**(*model*, day)  
*//removing last item and adding predicted value*
  3. remove first item from *train\_data*
  4. *train\_data*  $\leftarrow$  add *pred* to *train\_data*  
*//Last In, First Out.*
  5. *RESULTS*  $\leftarrow$  add *pred* to *RESULTS*
6. **end for**
7. **print**(*RESULTS*)
8. **plot**(*RESULTS*)
9. **return**

Algorithm 5.2 Overall Implementation

# CHAPTER 6

## TESTING

Testing is a critical element which assures quality and effectiveness of the proposed system in (satisfying) meeting its objectives. Testing is done at various stages in the System designing and implementation process with an objective of developing a transparent, flexible and secured system. Testing is an integral part of software development. Testing process, in a way certifies, whether the product, that is developed, complies with the standards, that it was designed to. Testing process involves building of test cases, against which, the product has to be tested.

### 6.1 Test Objectives

- Testing is a process of executing a program with the intent of finding an error.
- A good case is one that has a high probability of finding an undiscovered error.
- A successful test is one that uncovers a yet undiscovered error. If testing is conducted successfully (according to the objectives) it will uncover errors in the software.
- Testing can't show the absences of defects are present. It can only show that software defects are present.

### 6.2 Testing Principles

Before applying methods to design effective test cases, a software engineer must understand the basic principle that guides software testing. All the tests should be traceable to customer requirements.

### 6.3 Testing design

Any engineering product can be tested in one of two ways:

- White Box Testing
- Black Box Testing



### 6.3.1 White Box Testing

This testing is also called as glass box testing. In this testing, by knowing the specified function that a product has been designed to perform test can be conducted that demonstrates each function is fully operation at the same time searching for errors in each function. It is a test case design method that uses the control structure of the procedural design to derive test cases.

### 6.3.2 Black Box Testing

In this testing by knowing the internal operation of a product, tests can be conducted to ensure that “all gears mesh”, that is the internal operation performs according to specification and all internal components have been adequately exercised. It fundamentally focuses on the functional requirements of the software. The steps involved in black box test case design are:

- Graph based testing methods
- Equivalence partitioning
- Boundary value analysis
- Comparison testing

## 6.4 Testing Strategies

A software testing strategy provides a road map for the software developer. Testing is a set of activities that can be planned in advanced and conducted systematically. For this reason, a template for software testing a set of steps into which we can place specific test case design methods should be defined for software engineering process.

**Any software testing strategy should have the following characteristics:**

- Testing begins at the module level and works outward toward the integration of the entire computer-based system.
- Different testing techniques are appropriate at different points in time.
- The developer of the software and an independent test group conducts testing.
- Testing and debugging are different activities but debugging must be accommodated in any testing strategy.

## 6.5 Levels of Testing

Testing can be done in different levels of SDLC. They are:

- Unit Test
- Integration Test
- Functional Test

### 6.5.1 Unit Testing

The first level of testing is called unit testing. Unit testing verifies on the smallest unit of software designs-the module. The unit test is always white box oriented. In this, different modules are tested against the specifications produced during design for the modules. Unit testing is essentially for verification of the code produced during the coding phase, and hence the goal is to test the internal logic of the modules. It is typically done by the programmer of the module.

These types of tests are usually written by developers as they work on code (white-box style), to ensure that the specific function is working as expected. One function might have multiple tests, to catch corner cases or other branches in the code. Unit testing alone cannot verify the functionality of a piece of software, but rather is used to ensure that the building blocks of the software work independently from each other.

Unit testing is a software development process that involves synchronized application of a broad spectrum of defect prevention and detection strategies in order to reduce software development risks, time, and costs. It is performed by the software developer or engineer during the construction phase of the software development lifecycle. Rather than replace traditional quality analysis focuses, it augments it. Unit testing aims to eliminate construction errors before code is promoted to quality analysis; this strategy is intended to increase the quality of the resulting software as well as the efficiency of the overall development and quality analysis process.

Due to its close association with coding, the coding phase is frequently called “coding and unit testing.” The unit test can be conducted in parallel for multiple modules. We try testing various stock markets and predict the stock for each.

The Test cases in unit testing are as follows:

|                       |                                                                             |
|-----------------------|-----------------------------------------------------------------------------|
| Test Case ID          | Unit Test Case 1                                                            |
| Description           | IBM Stock Dataset                                                           |
| Input                 | <b>Date:</b> 01-01-2017 – 01-01-2018<br><b>Comp:</b> IBM<br><b>Days:</b> 90 |
| Expected output       | Prediction Successful                                                       |
| Actual Result/Remarks | Got the expected output                                                     |
| Passed (?)            | Yes                                                                         |

Table 6.1: Unit Test Case 1

In the next test case, we'll use the same date range but with Coca Cola stock

|                       |                                                                                   |
|-----------------------|-----------------------------------------------------------------------------------|
| Test Case ID          | Unit Test Case 2                                                                  |
| Description           | Coca Cola Stock Dataset                                                           |
| Input                 | <b>Date:</b> 01-01-2017 – 01-01-2018<br><b>Comp:</b> Coca Cola<br><b>Days:</b> 90 |
| Expected output       | Prediction Successful                                                             |
| Actual Result/Remarks | Got the expected output                                                           |
| Passed (?)            | Yes                                                                               |

Table 6.2: Unit Test Case 2

### 6.5.2 Integration Testing

The second level of testing is called integration testing. Integration testing is a systematic technique for constructing the program structure while conducting tests to uncover errors associated with interfacing. In this, many tested modules are combined into subsystems, which are then tested. The goal here is to see if all the modules can be integrated properly.

There are three types of integration testing:

- *Top-Down Integration*: Top down integration is an incremental approach to construction of program structures. Modules are integrated by moving downwards through the control hierarchy beginning with the main control module.
- *Bottom-Up Integration*: Bottom up integration as its name implies, begins construction and testing with automatic modules.
- *Regression Testing*: In this context of an integration test strategy, regression testing is the re execution of some subset of test that have already been conducted to ensure that changes have not propagated unintended side effects.

### 6.5.3 Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals. Functional testing is centered on the following items:

|               |                                                              |
|---------------|--------------------------------------------------------------|
| Valid Input   | Identified classes of valid input must be accepted.          |
| Invalid Input | Identified classes of invalid input must be rejected.        |
| Functions     | Identified functions must be exercised.                      |
| Output        | Identified classes of application outputs must be exercised. |

Table 6.3: Functional Testing items

**Systems/Procedures:** Interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

#### **6.5.4 Validation testing**

At the culmination of integration testing, software is completely assembled as a package; interfacing errors have been covered and corrected, and final series of software tests-validating testing may begin. Validation can be defined in many ways, but a simple definition is that validation succeeds when software functions in a manner that can be reasonably expected by customers. Reasonable expectation is defined in the software requirement specification- a document that describes all user visible attributes of the software. The specification contains a section title “validation criteria”. Information contained in that section forms the basis for validation testing approach

#### **6.5.5 Alpha testing**

It is virtually impossible for a software developer to foresee how the customer will really use a program. Instructions for use may be misinterpreted; strange combination of data may be regularly used and output that seemed clear to the tester may be unintelligible to a user in field.

When custom software is built for one customer, a series of acceptance tests are conducted to enable the customer to validate all requirements by the end user rather than system developer and acceptable test can range from an informal “test drive” to a planned and systematically executed series of tests. In fact, acceptance testing can be conducted over a period of weeks or months, thereby uncovering cumulative errors that might degrade the system over time. If software is developed as a product to be used by many customers, it is impractical to perform formal acceptance test with each one. Most software product builders use a process called alpha and beta testing to uncover errors that only the end user seems able to find.

A customer conducts the alpha test at the developer’s site. The software is used in a natural setting with the developer “Looking over the shoulder” of the user and recording errors and usage problems. Alpha tests are conducted in controlled environment.

### **6.5.6 Beta testing**

The beta test is conducted at one or more customer sites by the end user of the software. Unlike alpha testing, the developer is generally not present. Therefore, the beta test is a “live” application of the software in an environment that cannot be controlled by the developer. The customer records all problems that are encountered during beta testing and reports these to the developer at regular intervals. As a result of problems reported during beta test, the software developer makes modification and then prepares for release of the software product to the entire customer base.

### **6.5.7 System Testing and Acceptance Testing**

System testing is actually a series of different tests whose primary purpose is to fully exercise the computer-based system. Include recovery testing during crashes, security testing for unauthorized user, etc.

Acceptance testing is sometimes performed with realistic data of the client to demonstrate that the software is working satisfactorily. This testing in FDAC focuses on the external behavior of the system.

## CHAPTER 7

### PERFORMANCE ANALYSIS

Now that the system has been set up and the results have been obtained, we can do a performance analysis to observe the limits, reliability and accuracy of the proposed system. To achieve this, we provide the following inputs for various stock datasets available online.

|                     |                                                                              |
|---------------------|------------------------------------------------------------------------------|
| <b>Date Range</b>   | 01-01-2017 to 01-01-2018                                                     |
| <b>Companies</b>    | Microsoft, IBM, 3M, Coca-Cola, McDonald's, Intel, Nike, Apple, Cisco, Disney |
| <b>Forward Days</b> | <b>90 Days</b>                                                               |

Table 7.1 Input provided for performance analysis

We shall perform the analysis of all the companies and plot a performance graph. The analysis for the first two companies have been given in this literature.

#### 7.1 IBM Stock Prediction (IBM)

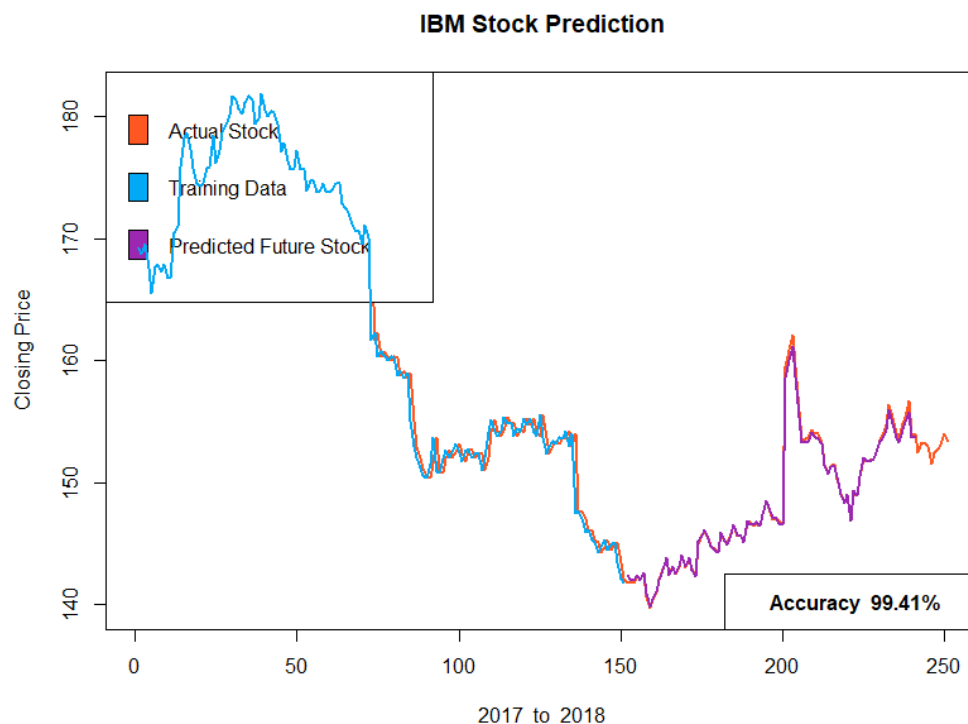


Figure 7.1 IBM Stock Prediction for 90 Days

As we can see from the above figure, we see the overall stock and also the predicted values. The system appears to perform extremely well with a 99.41% accuracy. The purple line almost perfectly follows the actual orange line.

During each prediction, the accuracy values are calculated and plotted separately as shown in Figure 7.2

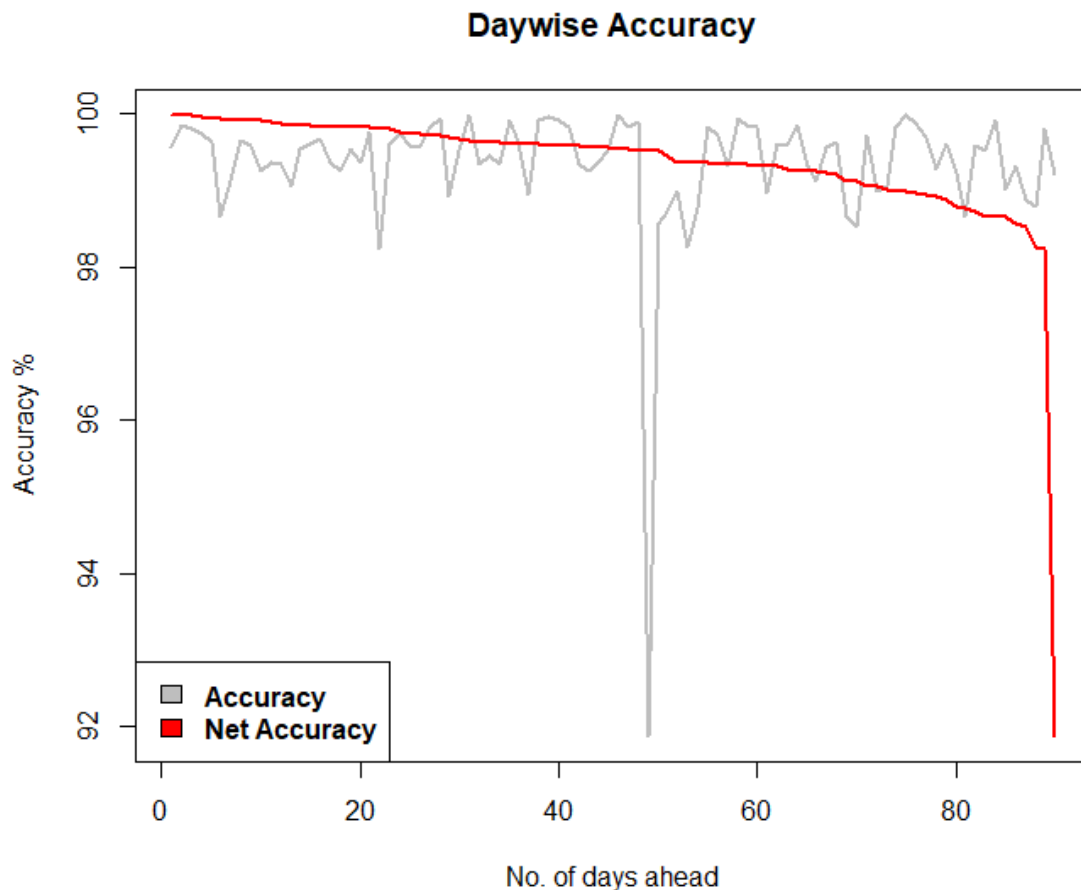


Figure 7.2 IBM: Day wise Accuracy

The above graph shows the day-wise accuracy of the predictions. The accuracy of the predicted and actual values was computed and plotted.

The prediction accuracy day-wise varies sometimes, and we plot them as the grey line. After sorting these values, we plot a red line to show the overall decline in accuracy.

The decline in accuracy over the days can be explained through a mean predictive accuracy graph. We can show this for the given IBM stock data.



The predictive overall performance of the entire prediction is shown in Figure 7.3

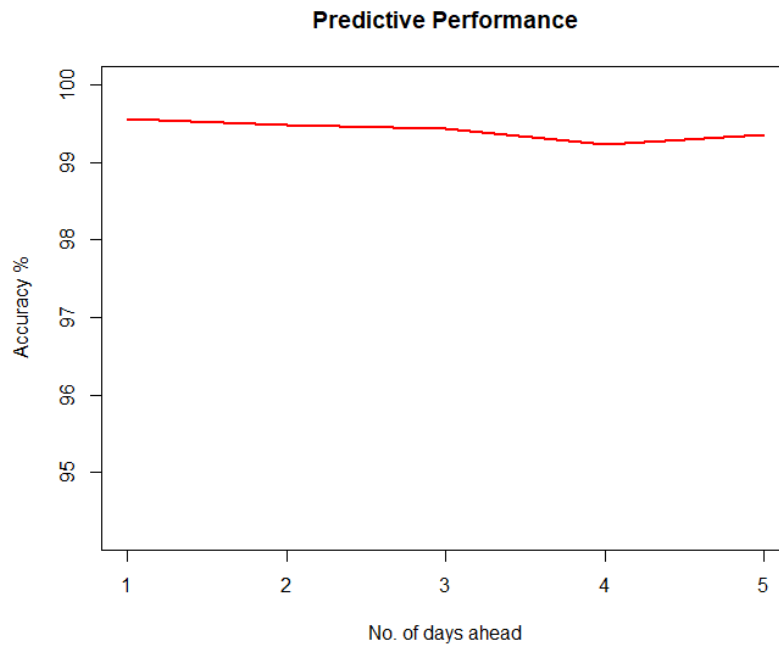


Figure 7.3 IBM: Predictive Performance

## 7.2 CISCO Stock Prediction (CSCO)

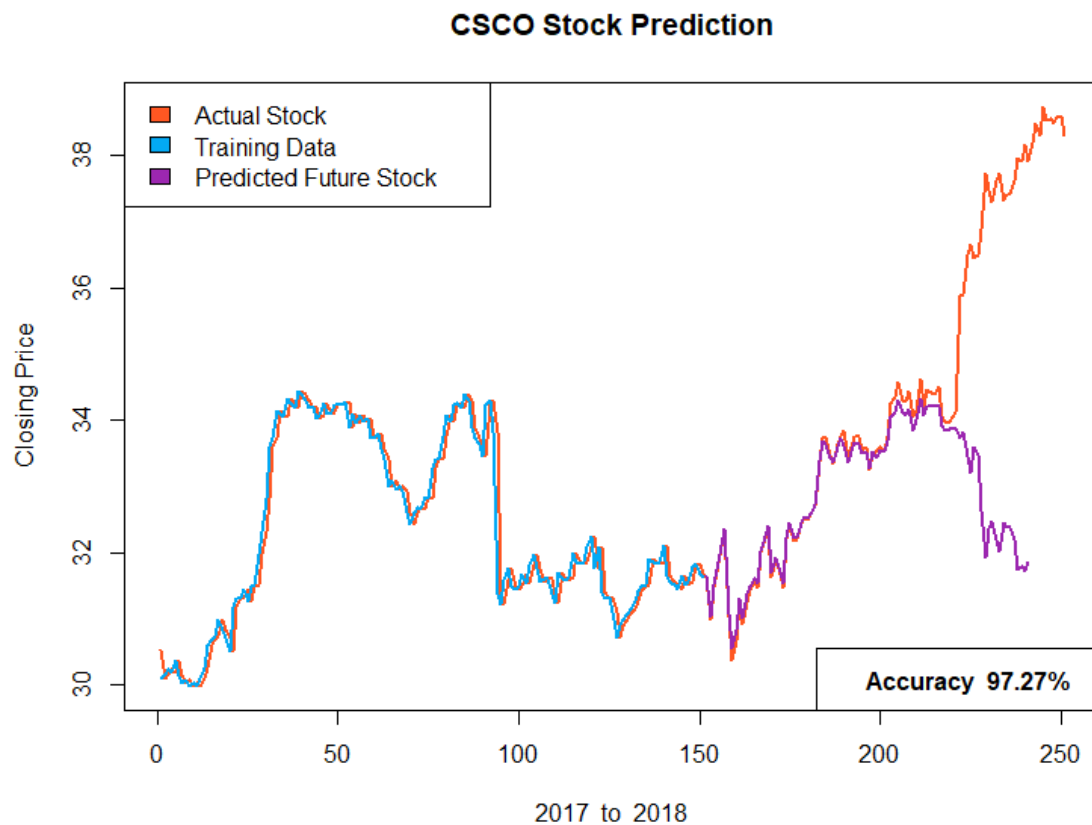


Figure 7.4 Cisco: Stock prediction for 90 days

In the Figure 7.4, we can see that the predicted stock declines steeply after a certain number of days. This sudden decline can be explained due to the external factors that influence stock market. These situations cannot be predicted by a learning system using past data. We can conclude that the performance of the proposed system is consistent only for a range of future days. We shall now see the performance decline in the prediction for CISCO by plotting a Mean Predictive Performance graph.

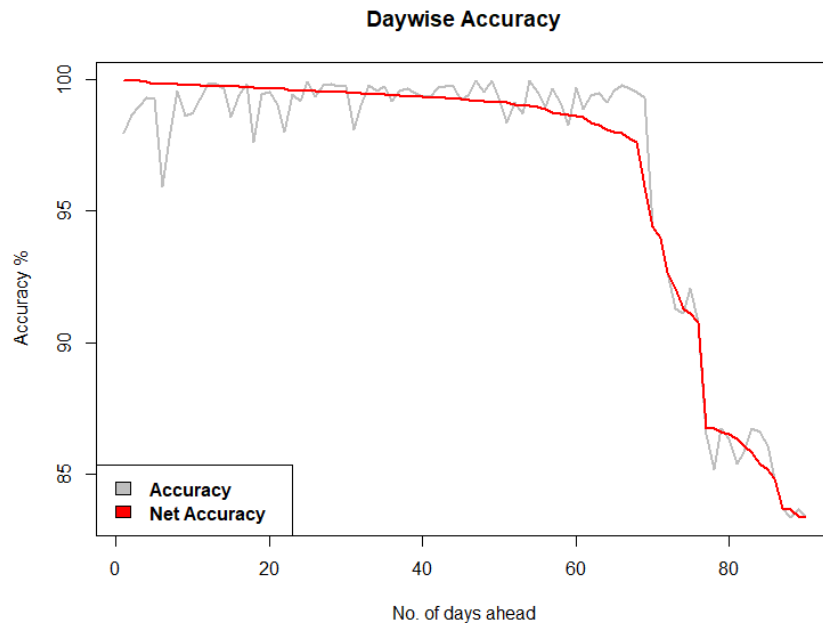


Figure 7.5 Cisco: Day wise accuracy

The following graph shows the overall performance over 90 days

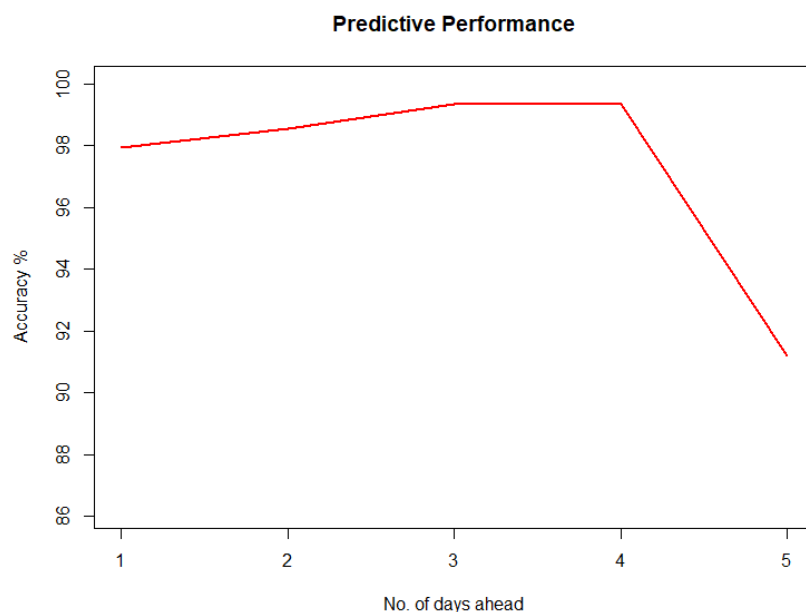


Figure 7.6 Cisco: Predictive Performance

### 7.3 Multiple Stock Performance

Similarly, we now do a complete performance over 10 datasets for a 90-day forward prediction. The resultant graph is as follows:

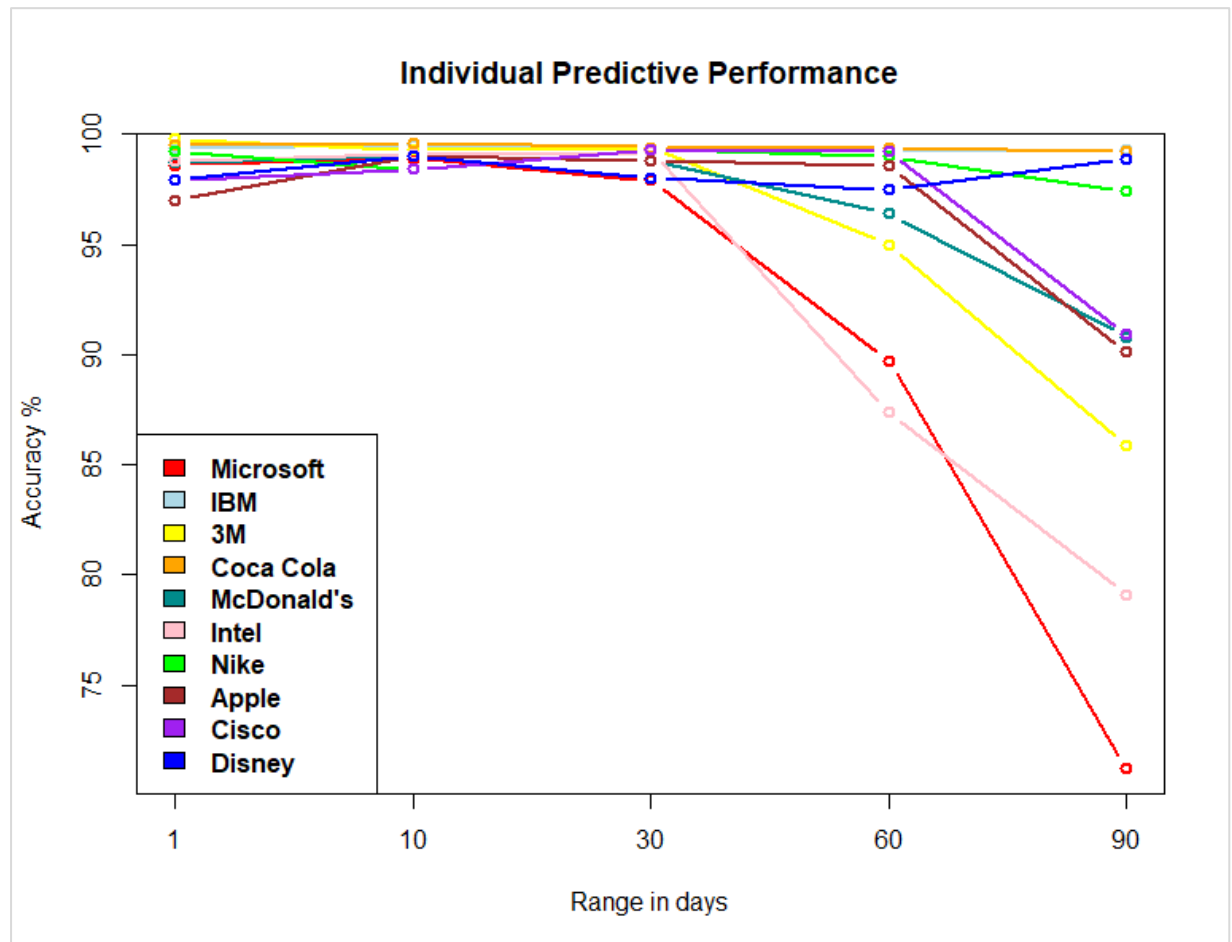


Figure 7.7 Individual Predictive Performance

### 7.3 Overall System Performance

We can now plot the universal performance of the proposed system by averaging the performance for all systems. The plotted graph is as shown in Figure 7.8.

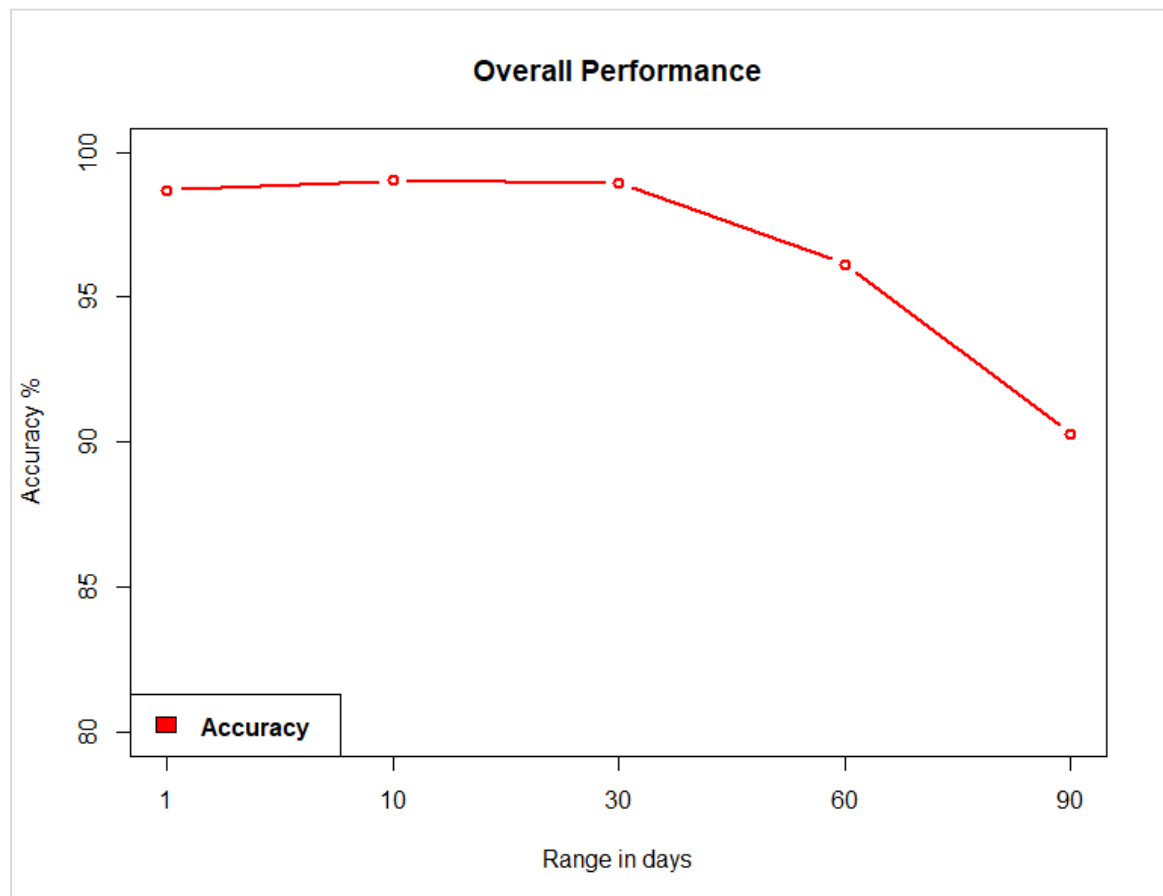


Figure 7.8 Overall System Performance

## CHAPTER 8

### RESULTS & DISCUSSIONS

We now observe the results obtained by the proposed system for each test case. When the user initializes the program, the following screen is displayed and the user enters the choice 3, for IBM in the date range 01-01-2017 to 01-01-2018 (which is the default range). The user predicts stock for the next 10 days. The input is given as follows:

```
Welcome to Stock Market Prediction
Choose the dataset you wish to predict the stock for

1. Microsoft      6. McDonald's
2. Apple          7. Intel Corp.
3. IBM            8. 3M
4. Nike           9. CocaCola
5. Cisco          10. Walt Disney

Enter your choice : 3
Do you wish to use default date range? (y|n) y
Fetching data for IBM
Predict stock for how many days in advance? (1 - 99 ) : 10
```

Table 8.1 Requesting input for prediction

Once these inputs have been given, the system fetches the data from Quandl or any other online source and feeds this data to the LSSVR and trains the model. When the training is complete, the predicted values are displayed for the next 10 days as follows:

| DAY    | DATE       | PREDICTED PRICE |
|--------|------------|-----------------|
| Day 01 | 2017-08-09 | 142.5203        |
| Day 02 | 2017-08-10 | 141.5793        |
| Day 03 | 2017-08-11 | 142.1064        |
| Day 04 | 2017-08-14 | 142.6604        |
| Day 05 | 2017-08-15 | 142.1869        |
| Day 06 | 2017-08-16 | 142.5381        |
| Day 07 | 2017-08-17 | 140.6312        |
| Day 08 | 2017-08-18 | 141.1423        |
| Day 09 | 2017-08-21 | 140.3994        |
| Day 10 | 2017-08-22 | 141.1157        |

Table 8.2 Predicted closing stock prices

After the above results are displayed, we also obtain the basic statistics and summary of the prediction for any given input set.

| -----Prediction Summary----- |        |
|------------------------------|--------|
| BASIC STATS                  |        |
| No. of days predicted        | 10     |
| Training data (days)         | 151    |
| Max. pred. accuracy          | 99.85% |
| Min. pred. accuracy          | 98.71% |
| Mean pred. accuracy          | 99.49% |
| Predictive Accuracy          | 99.5%  |

Table 8.3 Basic Summary Statistics

Now that we have the predicted values, we can later compare them to the actual closing stock prices by using the *my\_summary()* function.

|    | Date       | Actual | Predicted |
|----|------------|--------|-----------|
| 1  | 2017-08-09 | 141.84 | 142.5203  |
| 2  | 2017-08-10 | 141.84 | 141.5793  |
| 3  | 2017-08-11 | 142.32 | 142.1064  |
| 4  | 2017-08-14 | 142.07 | 142.6604  |
| 5  | 2017-08-15 | 142.50 | 142.1869  |
| 6  | 2017-08-16 | 140.70 | 142.5381  |
| 7  | 2017-08-17 | 139.70 | 140.6312  |
| 8  | 2017-08-18 | 140.33 | 141.1423  |
| 9  | 2017-08-21 | 141.01 | 140.3994  |
| 10 | 2017-08-22 | 142.14 | 141.1157  |

Table 8.4 Results with actual and predicted prices

From the table given above we see the 10-day forward prediction for the given test case, and the actual prices that were observed. The obtained and trained values are plotted on a graph as shown in the next section.

The trained data are in Blue colour and the predicted values are in purple. The actual values of the stock on any given day is given in Orange.

We now obtain the graph that is plotted along with actual, predicted and trained data. In the graph show in Figure x, we see the blue line represents the existing training data that is fed into the LSSVR to train the model. We shall assume that the end of the training set represents the “Current Day” and that the days that follow are the future.

The future closing prices are predicted and plotted as the purple line. The orange line represents the actual closing stock prices on that day. In Figure 8.1, the prediction is encircled in red.

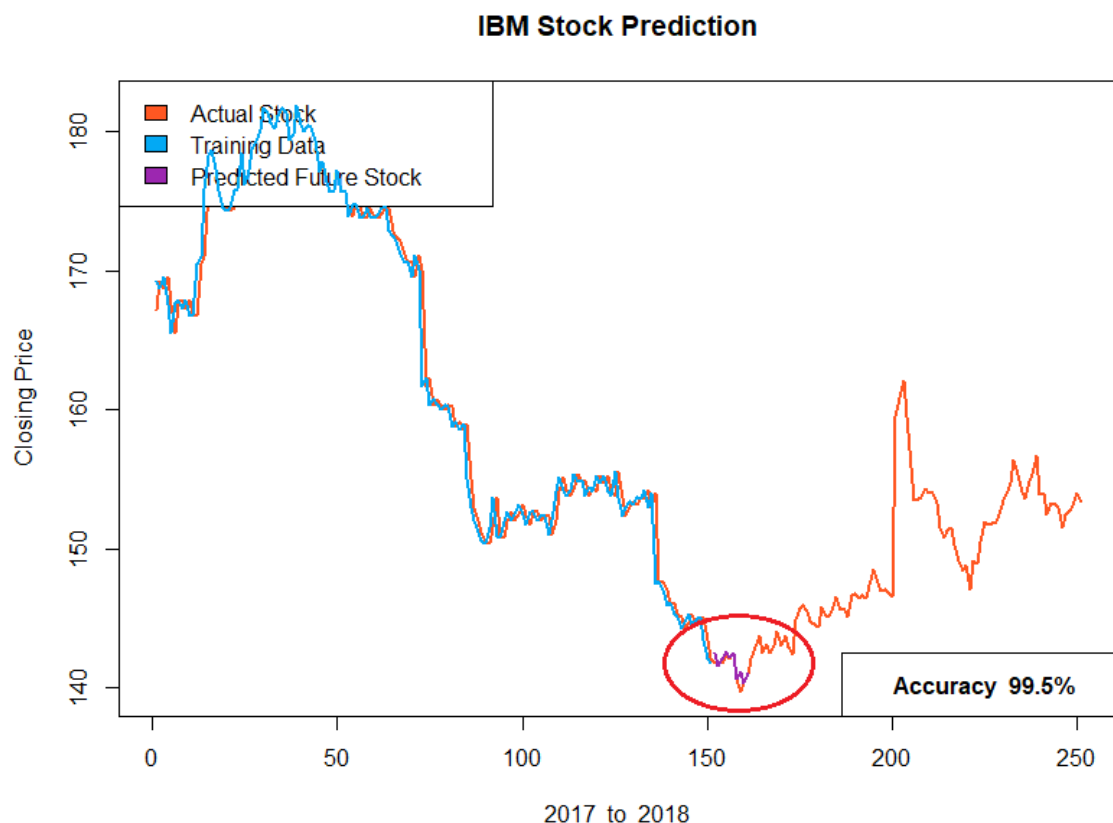


Figure 8.1 IBM Stock Prediction for 10 days

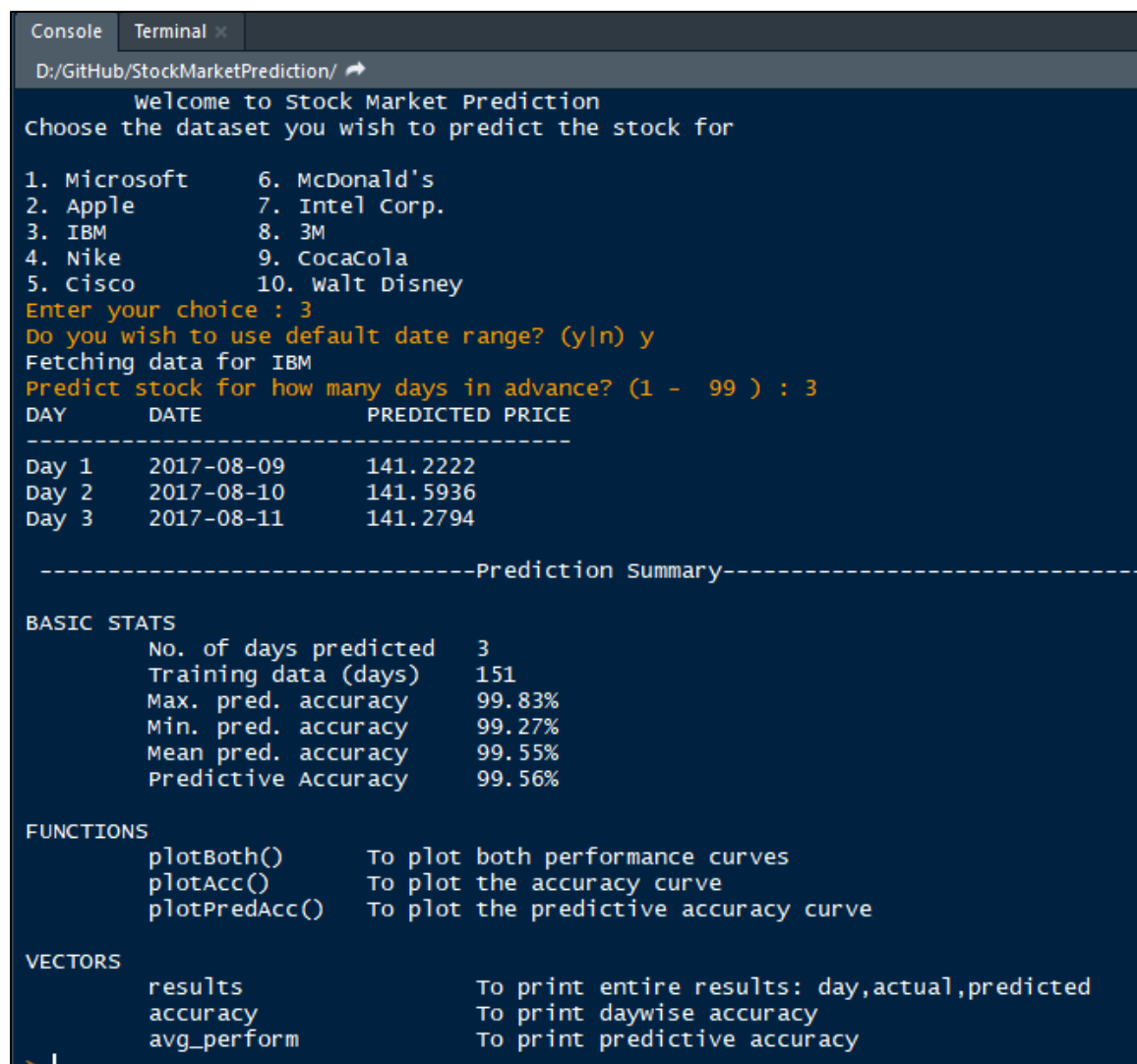
We see, that for 10 days, the mean prediction is given by dividing the actual and predicted values and finding the average. It has been observed for 10 days, the accuracy is around 99.5%.

We shall now do a performance review of the system with multiple dataset chosen at random in the same data range for a forward prediction of up to 90 days.

## CHAPTER 9

### SCREENSHOTS

An interface to the proposed system has been prepared and are shown in this section.



```
Console Terminal x
D:/GitHub/StockMarketPrediction/
welcome to Stock Market Prediction
Choose the dataset you wish to predict the stock for

1. Microsoft      6. McDonald's
2. Apple          7. Intel Corp.
3. IBM            8. 3M
4. Nike           9. CocaCola
5. Cisco          10. Walt Disney
Enter your choice : 3
Do you wish to use default date range? (y/n) y
Fetching data for IBM
Predict stock for how many days in advance? (1 - 99 ) : 3
DAY      DATE      PREDICTED PRICE
-----
Day 1     2017-08-09     141.2222
Day 2     2017-08-10     141.5936
Day 3     2017-08-11     141.2794

-----Prediction Summary-----

BASIC STATS
      No. of days predicted      3
      Training data (days)     151
      Max. pred. accuracy       99.83%
      Min. pred. accuracy       99.27%
      Mean pred. accuracy       99.55%
      Predictive Accuracy       99.56%

FUNCTIONS
      plotBoth()      To plot both performance curves
      plotAcc()       To plot the accuracy curve
      plotPredAcc()   To plot the predictive accuracy curve

VECTORS
      results         To print entire results: day,actual,predicted
      accuracy        To print daywise accuracy
      avg_perform     To print predictive accuracy

> |
```

Figure 9.1 Interface of the system

The above screenshot shows the user interface where the user enters the details. The user is free to choose any of the given options for stock after which he is asked for the date range. With a slight modification the user can also enter his own choice for Company ticker to fetch the stock.

A *my\_summary()* function is also provided that displays a summary of statistics, which can be used for further analysis such as plotting and forwarding results.



When the program is executed, the graph is displayed side-by-side instantly as soon as the stock is fetched and updated with the prediction when the training is complete.

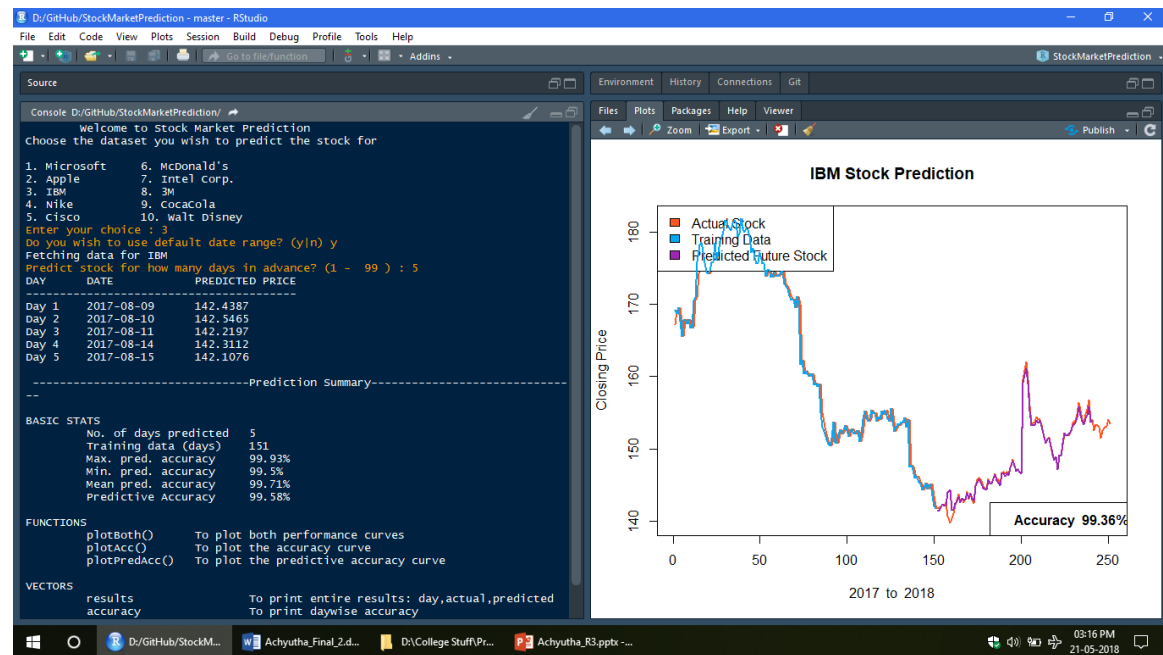


Figure 9.2 The screenshot of system

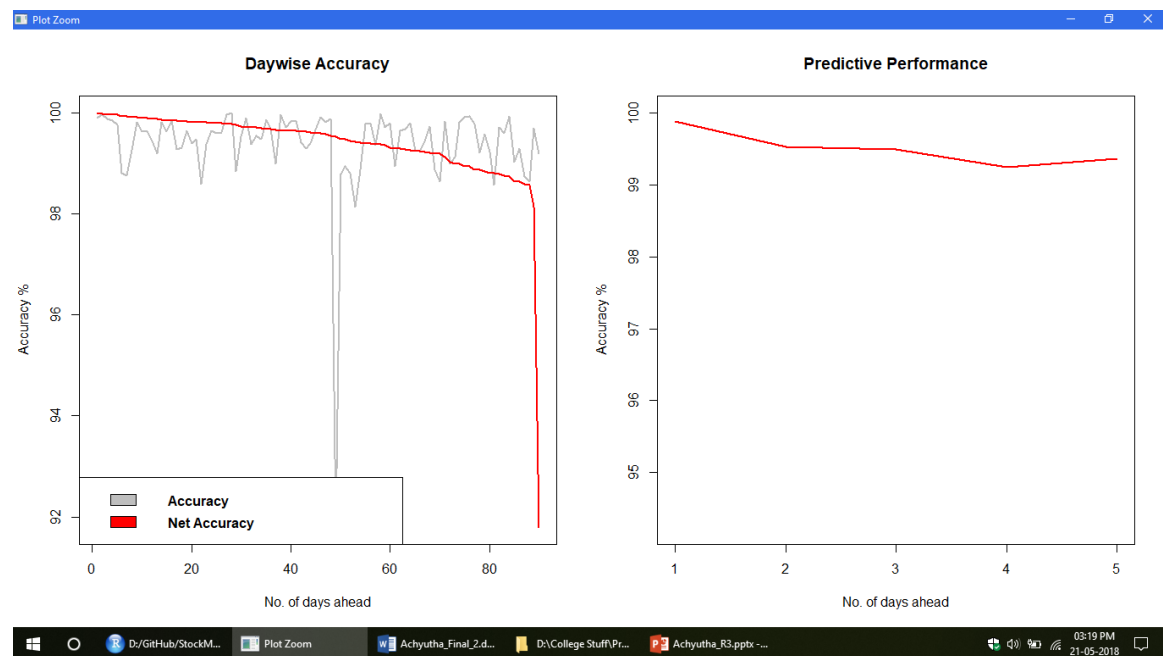


Figure 9.3 Predictive Performance of the system

## **CHAPTER 10**

### **FUTURE ENHANCEMENT**

#### **10.1 Limitations**

The proposed system has a number of limitations that have been observed such as its computational speed, especially with respect to sliding-window validation as the computational cost increases with the number of forward day predictions.

- LSSVR is computationally slower when used along with sliding-window method.
- Each time the window is moved across, the LSSVR must learn for the new dataset
- The LSSVR must be run each time the window slides and thus the window size affects the performance of the system.
- LSSVR uses many parameters that must be handled and taken care of.
- The system is best suited for short term prediction and error rate increases with number of days of prediction

#### **10.2 Future Enhancement**

- The proposed model does not predict well for sudden changes in the trend of stock data.
- This occurs due to external factors and real-world changes affecting the stock market.
- We can overcome this by implementing Sentiment Analysis and Neural Networks to enhance the proposed model.
- We can modify the same system to an online-learning system that adapts in real-time.

## CHAPTER 11

### CONCLUSION

Decision to buy or sell a stock is very complicated since many factors can affect stock price. This work presents a novel approach, based on LSSVR and Machine Learning to constructing a stock price forecasting expert system, with the aim of improving forecasting accuracy.

Thus, as we can see in our proposed method, we train the data using existing stock dataset that is available. We use this data to predict and forecast the stock price of n-days into the future.

The average performance of the model decreases with increase in number of days, due to unpredictable changes in trend as noted in the literature's limitations.

The current system can update its training set as each day passes so as to detect newer trends and behave like an online-learning system that predicts stock in real-time. The intelligent time series prediction system that uses sliding-window metaheuristic optimization is a graphical user interface that can be run as a stand-alone application. The system makes the prediction of stock market values simpler, involving fewer computations, than that using the other method that was mentioned above.

## REFERENCES

- [1] **Jui-Sheng Chou and Thi-Kha Nguyen**, Forward Forecast of Stock Price Using Sliding-window Metaheuristic-optimized Machine Learning Regression, IEEE Transactions on Industrial Informatics, 2018, DOI 10.1109/TII.2018.2794389
- [2] **M. Göçken, M. Özçalıcı, A. Boru, and A. T. Dosdoğru**, “Integrating metaheuristics and Artificial Neural Networks for improved stock price prediction,” Expert Systems with Applications, vol. 44, pp. 320-331, 2016.
- [3] **Y. Bao, Y. Lu, and J. Zhang**, "Forecasting Stock Price by SVMs Regression," Artificial Intelligence: Methodology, Systems, and Applications, C. Bussler and D. Fensel, eds., pp. 295-303, Berlin, Heidelberg: Springer Berlin Heidelberg, 2004
- [4] **K. Duan, S. S. Keerthi, and A. N. Poo**, “Evaluation of simple performance measures for tuning SVM hyperparameters,” Neurocomputing, vol. 51, pp. 41-59, 2003.
- [5] **T. Xiong, Y. Bao, and Z. Hu**, “Multiple-output support vector regression with a firefly algorithm for interval-valued stock price index forecasting,” Knowledge-Based Systems, vol. 55, pp. 87-100, 2014.
- [6] **D. Saini, A. Saxena, and R. C. Bansal**, "Electricity price forecasting by linear regression and SVM." pp. 1-7.
- [7] **V. N. Vapnik**, The nature of statistical learning theory: Springer-Verlag New York, Inc., 1995
- [8] **J. A. K. Suykens**, "Nonlinear modelling and support vector machines." pp. 287-294.
- [9] **W. Hao, and S. Yu**, "Support Vector Regression for Financial Time Series Forecasting," Knowledge Enterprise: Intelligent Strategies in Product Design, Manufacturing, and Management, K. Wang, G. L. Kovacs, M. Wozny and M. Fang, eds., pp. 825-830, Boston, MA: Springer US, 2006
- [10] **X.-S. Yang**, Nature-Inspired Metaheuristic Algorithms: Luniver Press, 2008.
- [11] **J.-S. Chou, and A.-D. Pham**, “Smart Artificial Firefly Colony Algorithm-Based Support Vector Regression for Enhanced Forecasting in Civil Engineering,” Computer-Aided Civil and Infrastructure Engineering, vol. 30, no. 9, pp. 715-732, 2015.
- [12] **J.-S. Chou, W. K. Chong, and D.-K. Bui**, “Nature-Inspired Metaheuristic Regression System: Programming and Implementation for Civil Engineering Applications,” Journal of Computing in Civil Engineering, vol. 30, no. 5, 2016.