

**VISVESVARAYA TECHNOLOGICAL UNIVERSITY
BELGAUM-590014**



PROJECT ENTITLED

**“TRADING SIMULATION AND STOCK
MARKET PREDICTION”**

Submitted in partial fulfillment of the requirements for the award of degree of

**BACHELOR OF ENGINEERING
In
COMPUTER SCIENCE AND ENGINEERING**

For the Academic year 2012-2013

Submitted by:

Arun S.	1MV09CS020
Darshan M.S.	1MV09CS031
Sneha Priscilla M.	1MV09CS098
Vivek John George	1MV09CS109

Project carried out at
**Sir M. Visvesvaraya Institute of Technology
Bangalore-562157**

Under the Guidance of
Mrs Ch. Vani Priya
Lecturer, Department of CSE
Sir M Vivesvaraya Institute of Technology, Bangalore



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
SIR M. VISVESVARAYA INSTITUTE OF TECHNOLOGY
HUNASAMARANAHALLI, BANGALORE-562157**

Sir M. Visvesvaraya Institute of Technology
BANGALORE – 562157

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING



Certificate

Certified that the project work entitled “**TRADING SIMULATION AND STOCK MARKET PREDICTION**” is a bonafide work carried out by **Arun S. (1MV09CS020)**, **Darshan M.S. (1MV09CS031)**, **Sneha Priscilla M. (1MV09CS098)**, & **Vivek John George (1MV09CS0)** in partial fulfillment for the award of Degree of Bachelor of Engineering in Computer Science and Engineering of the Visvesvaraya Technological University, Belgaum during the year 2012-2013. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the Bachelor of Engineering Degree.

Signature of the Guide

Signature of the HOD

Signature of the Principal

Mrs Ch. Vani Priya
Lecturer, Dept of CSE
Sir MVIT

Prof. Dilip K Sen
HOD, Dept of CSE
Sir MVIT

Dr. M S Indira
Principal
Sir MVIT

External Viva:

Name of the examiners

Signature with Date

1)

2)

DECLARATION

We hereby declare that the entire project work embodied in this dissertation has been carried out by us and no part has been submitted for any degree or diploma of any institution previously.

Place: Bangalore

Date:

Signature of students

ARUN S. (1MV09CS020)

DARSHAN M.S. (1MV09CS031)

SNEHA PRISCILLA (1MV09CS098)

VIVEK JOHN GEORGE (1MV09CS109)

ABSTRACT

In certain applications involving big data, data sets get so large and complex that it becomes difficult to analyze using traditional data processing applications.

In order to overcome these challenges, we can extract useful information from big data to an understandable structure using Data Mining. We can also use algorithms that learn from this data and automatically predict further trends. This branch of Artificial Intelligence is called Machine Learning and Artificial Neural Networks is the approach we are using to implement this.

The stock market is a platform where an enormous amount of data exists and constantly needs to be scrutinized for business opportunities. Therefore, we are applying these aforementioned methods to simulate a brokerage system and analyze the stock market while at the same time learning the fundamentals of investment, without risking your own money.

ACKNOWLEDGMENT

It gives us immense pleasure to express our sincere gratitude to the management of **Sir M. Visvesvaraya Institute of Technology**, Bangalore for providing the opportunity and the resources to accomplish our project work in their premises.

On the path of learning, the presence of an experienced guide is indispensable and we would like to thank our guide **Mrs Ch. Vani Priya**, Assistant Professor, Dept. of Computer Science and Engineering, for his invaluable help and guidance.

We would also like to convey my regards and sincere thanks to **Prof. Dilip K Sen**, Head of the Department, Dept of Computer Science and Engineering for his suggestions, constant support and encouragement. Heartfelt and sincere thanks to **Dr. M S Indira**, Principal, Sir. MVIT for providing us with the infrastructure and facilities needed to develop our project.

We would also like to thank the staff of Department of Computer Science and Engineering and lab-in-charges for their co-operation and suggestions. Last but not the least we would like to thank all our friends for their help and suggestions without which completing this project would have been impossible.

-ARUN S. (1MV09CS020)

-DARSHAN M.S. (1MV09CS031)

-SNEHA PRISCILLA (1MV09CS098)

-VIVEK JOHN GEORGE (1MV09CS109)

TABLE OF CONTENTS

Title Page	I
Certificate	II
Declaration	III
Abstract	IV
Acknowledgement	V
Table of Contents	VI
List of Figures	IX
Chapter 1 Introduction	1
1.1 General Introduction	1
1.2 Statement of the problem	2
1.3 Objectives of the project	3
Chapter 2 Literature Survey	4
2.1 Current scope	4
2.2 Literature Survey	4
Chapter 3 Neural Networks	6
3.1 The Feedforward Backpropagation Algorithm	6
3.2 Neural Network Basics	6
3.3 Perceptrons	8
3.4 The Delta Rule	9
3.5 Multi-Layer Networks and Backpropagation	14
3.6 Network Terminology	14
3.7 The Sigma Function	15
3.8 The Backpropagation Algorithm	16
3.8 Bias	17
3.9 Network Topology	18

Chapter 4	Implementation	20
	4.1 Database	20
	4.1.1 SQLite	20
	4.1.2 Database design	21
	4.2 Extraction of Stock Data	22
	4.3 Neuroph	23
	4.4 Eclipse IDE	24
	4.5 Java	26
	4.5.1 Java Platform, Enterprise Edition	27
	4.5.2 Web Applications Container	27
	4.5.3 Java Web application	27
	4.5.4 Servlets	27
	4.5.5 Java Server Pages	28
	4.5.6 Apache Tomcat	28
	4. 6 User Interface Implementation	29
Chapter 5	Testing	36
	5.1 Testing process	36
	5.2 Testing Objectives	36
	5.3 Levels of testing	37
	5.3.1 Unit testing	37
	5.3.1.1 User Input	37
	5.3.1.2 Error Handling	37
	5.3.2 Integration Testing	38
	5.3.3 System Testing	39
	5.4 Test Results for the predictions	39
Chapter 6	Sentiment Analysis	42
	6.1 Basics	42

6.2 R (programming language)	44
6.3 Sentiment analysis with R	45
6.3.1 Text Cleaning	46
6.3.2 Extract the Sentiment	46
6.4 Neural Network and Sentiment Analysis.	47
6.5 Results	48
Chapter 7 Snap Shots	50
Chapter 8 Conclusion	53
Chapter 9 Future Enhancement	54
Bibliography	55

List of Figures

Figure no	Description	Page No
Fig 3.1	A typical feedforward neural network.	7
Fig 3.2	Comparison between a biological neuron and an artificial neuron.	8
Fig.3.3	Delta Rule	10
Fig 3.4	Error Function	11
Fig 3.5	Bias	18
Fig 3.6	Examples of Neural Network	19
Fig 4.1	Database schema	21
Fig 4.2	Neuroph framework	23
Fig 4.3	Basic concepts in Neuroph Framework	24
Fig 4.4	Java Runtime Environment	26
Fig 4.5	Carousel	30
Fig 4.6	Jumbrotron Subhead	31
Fig 4.7	Navbar	31
Fig 4.8	Cloud Overlay Effect.	31

Fig 4.9	Validating forms	32
Fig 4.10	AutoComplete	33
Fig 4.11	Datatables	34
Fig 4.12	Highcharts	35
Fig 5.1	Prediction Performance 1	39
Fig 5.2	Prediction Performance 2	40
Fig 5.3	Prediction Performance 3	40
Fig 5.4	Prediction Performance 4	40
Fig 5.5	Prediction Performance 5	41
Fig 6.1	Sentiment Analyzer framework	47
Fig 6.2	Neural Network Layout	48
Fig 6.3	Predictor Framework	48

CHAPTER 1

INTRODUCTION

1.1 General Introduction

For a new investor, the stock market can feel a lot like legalized gambling. "Ladies and gentlemen, place your bets! Randomly choose a stock based on gut instinct. If the price of your stock goes up -- you win! If it drops, you lose!" Not exactly. The stock market can be intimidating, but the more you learn about stocks, and the more you understand the true nature of stock market investment, the better and smarter you'll manage your money.

Terms:

- A stock of a company constitutes the equity stake of all shareholders.
- A share of stock is literally a share in the ownership of a company. When you buy a share of stock, you're entitled to a small fraction of the assets and earnings of that company.
- Assets include everything the company owns (buildings, equipment, trademarks)
- Stocks in publicly traded companies are bought and sold at a stock market or a stock exchange.

These are some examples of popular stock exchanges:

- NYSE - New York Stock Exchange
- NASDAQ - National Association of Securities Dealers
- NSE – National Stock Exchange(India)
- BSE – Bombay Stock Exchange

The truth is there is no magical way to predict the stock market. Many issues affect rises and falls in share prices, whether gradual changes or sharp spikes. The best way to understand how the market fluctuates is to study trends.

Stock market trends are like the behavior of a person. After you study how a person reacts to different situations, you can make predictions about how that person will react to an event. Similarly, recognizing a trend in the stock market or in an individual stock will enable you to choose the best times to buy and sell.

Prediction methods are of the following categories:

i. Fundamental Analysis:

Fundamental Analysts are concerned with the company that underlies the stock itself. They evaluate a company's past performance as well as the credibility of its accounts.

ii. Technical Analysis:

Technical analysts or chartists are not concerned with any of the company's fundamentals. They seek to determine the future price of a stock based solely on the (potential) trends of the past price. The most prominent technique involves the use of artificial neural networks (ANNs).

This is the technology we are using for prediction in our project apart from creating a virtual trading system where users can buy and sell stocks in a virtual environment and test the waters a bit before investing with real money.

1.2 Statement of the Problem

When you buy a stock, you place a bet on how that stock will perform. In a perfect world, we can easily determine where to invest based on previous data. But what happens when the volume of data used to make decisions increases 100 million times, and trading volumes increase 100 million times, and trades can be transacted over 100 million times a second?

The proliferation of mobile phones, social media, machine data, and web logs has led to massive amounts of data being processed, and this volume is increasing exponentially with the digital shift from offline to online making data more expensive and complex to manage.

Inside these rapidly expanding data pools are millions of tiny little “tells” that can be extracted and combined with the emerging science of anticipatory computing into very predictable movement indicators.

Some examples of jaw dropping stats are:

- 340 million tweets are sent per day. That’s nearly 4,000 tweets per second.

- 247 billion emails are sent every day (80% is spam!)
- 10,000 payment card transactions are made every second around the world.
- Wal-Mart averages more than 1 million customer transactions every hour.
- 30 billion pieces of content are shared on Facebook every month.

Big Data is indeed ...big! And getting bigger.

1.3 Objectives of the project

- View the current status of the stock market by providing charts, graphs , news feeds and other research tools.
- Be able to buy and sell stocks and analyze profits and losses made.
- Predicting future trends and therefore making informed decisions on that basis.
- Learning and educating oneself about the market before investing with real money.
- Provide a simplistic user interface which guides and assists a new user to understand the stock market with ease.

CHAPTER 2

LITERATURE SURVEY

2.1 Current Scope

This application can be used to retrieve the current market scenario at any given point of time and allows a user to trade virtual money using real time data. By analyzing historical data as well as user's portfolio, it guides the user while buying stocks by predicting future trends in the stock market on a day to day basis.

Currently, it can be successfully hosted on a web server and serve as a virtual stock market trading platform

2.2 Literature Survey

Takashi Kimoto and Kazuo Asakawa Computer-based Systems Laboratory FUJITSU LABORATORIES LTD., KAWASAKI and Morio Yoda and Masakazu Takeoka INVESTMENT TECHNOLOGY & RESEARCH DIVISION The Nikko Securities Co., Ltd. Japan proposed buying and selling timing prediction system for stocks on the Tokyo Stock Exchange and analysis of internal representation. It is based on modular neural networks. They developed a number of learning algorithms and prediction methods for the TOPIX (Tokyo Stock Exchange Prices Indexes) prediction system. The prediction system achieved accurate predictions and the simulation on stocks trading showed an excellent profit. The prediction system was developed by Fujitsu and Nikko Securities.

Ramon Lawrence, Department of Computer Science University of Manitoba ,his paper is a survey on the application of neural networks in forecasting stock market prices. With their ability to discover patterns in nonlinear and chaotic systems, neural networks offer the ability to predict market directions more accurately than current techniques. Common market analysis techniques such as technical analysis, fundamental analysis, and regression are discussed and compared with neural network performance. Also, the Efficient Market Hypothesis (EMH) is presented and contrasted with chaos theory and neural networks. This paper refutes the EMH based on previous neural network work.Finally, future directions for applying neural networks to the financial markets are discussed.

Xue Zhang, Hauke Fuehres , Peter A. Gloor from National University of Defense Technology, Changsha, Hunan, China and MIT Center for Collective Intelligence, Cambridge MA, USA .Their work describes early work trying to predict stock market indicators such as Dow Jones, NASDAQ and S&P 500 by analyzing Twitter posts. We collected the twitter feeds for six months and got a randomized subsample of about one hundredth of the full volume of all tweets. We measured collective hope and fear on each day and analyzed the correlation between these indices and the stock market indicators. We found that emotional tweet percentage significantly negatively correlated with Dow Jones, NASDAQ and S&P 500, but displayed significant positive correlation to VIX. It therefore seems that just checking on twitter for emotional outbursts of any kind gives a predictor of how the stock market will be doing the next day.

CHAPTER 3

NEURAL NETWORKS

Computational neurobiologists have constructed very elaborate computer models of neurons in order to run detailed simulations of particular circuits in the brain. As Computer Scientists, we are more interested in the general properties of neural networks, independent of how they are actually "implemented" in the brain. This means that we can use much simpler, abstract "neurons", which (hopefully) capture the essence of neural computation even if they leave out much of the details of how biological neurons work.

People have implemented model neurons in hardware as electronic circuits, often integrated on VLSI chips. Remember though that computers run much faster than brains - we can therefore run fairly large networks of simple model neurons as software simulations in reasonable time. This has obvious advantages over having to use special "neural" computer hardware.

3.1 The Feedforward Backpropagation Algorithm

Although the long-term goal of the neural-network community remains the design of autonomous machine intelligence, the main modern application of artificial neural networks is in the field of pattern recognition. In the sub-field of data classification, neural-network methods have been found to be useful alternatives to statistical techniques such as those which involve regression analysis or probability density estimation. The potential utility of neural networks in the classification of multisource satellite-imagery databases has been recognized for well over a decade, and today neural networks are an established tool in the field of remote sensing. The most widely applied neural network algorithm in image classification remains the feedforward backpropagation algorithm. This web page is devoted to explaining the basic nature of this classification routine.

3.2 Neural Network Basics

Neural networks are members of a family of computational architectures inspired by biological brains. Such architectures are commonly called "connectionist systems", and are composed of interconnected and interacting components called nodes or neurons (these terms are generally considered synonyms in connectionist terminology, and are used interchangeably here). Neural networks are characterized by a lack of explicit

representation of knowledge; there are no symbols or values that directly correspond to classes of interest. Rather, knowledge is implicitly represented in the patterns of interactions between network components. A graphical depiction of a typical feedforward neural network is given in Fig 3.1. The term “feedforward” indicates that the network has links that extend in only one direction. Except during training, there are no backward links in a feedforward network; all links proceed from input nodes toward output nodes.

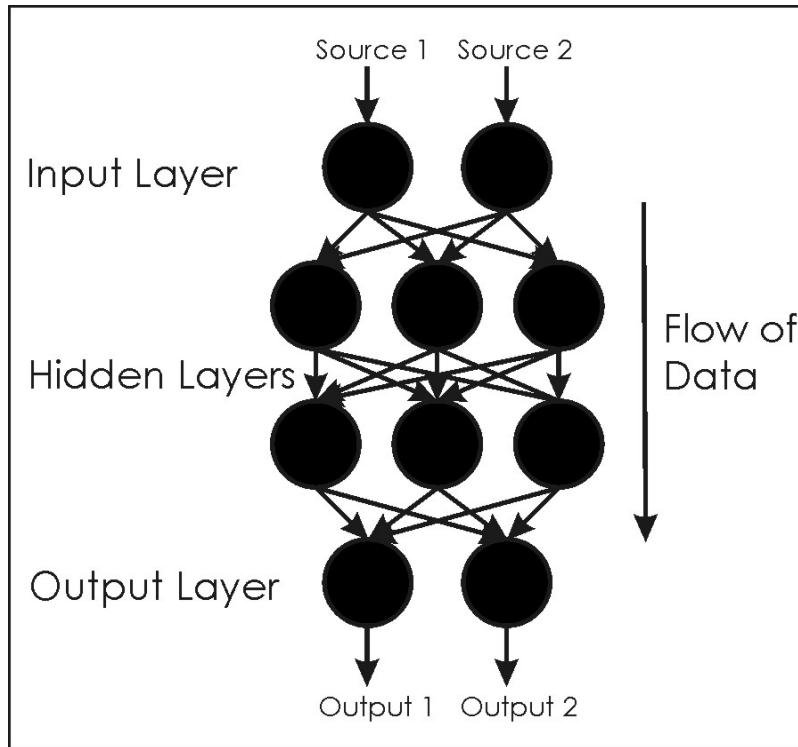


Fig 3.1: A typical feedforward neural network.

Individual nodes in a neural network emulate biological neurons by taking input data and performing simple operations on the data, selectively passing the results on to other neurons (Fig 3.2). The output of each node is called its "activation" (the terms "node values" and "activations" are used interchangeably here). Weight values are associated with each vector and node in the network, and these values constrain how input data (e.g., satellite image values) are related to output data (e.g., land-cover classes). Weight values associated with individual nodes are also known as biases. Weight values are determined by the iterative flow of training data through the network (i.e., weight values are established during a training phase in which the network learns how to identify particular classes by their typical input data characteristics). Once trained, the neural network can be applied toward the classification of new data. Classifications are performed by trained

networks through 1) the activation of network input nodes by relevant data sources [these data sources must directly match those used in the training of the network], 2) the forward flow of this data through the network, and 3) the ultimate activation of the output nodes. The pattern of activation of the network's output nodes determines the outcome of each pixel's classification.

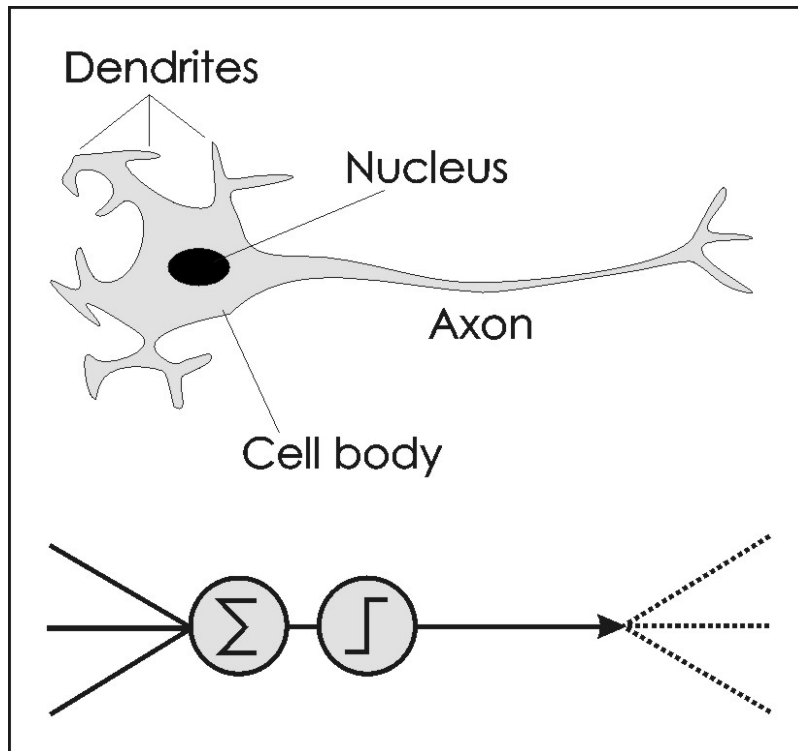


Fig 3.2 Schematic comparison between a biological neuron and an artificial neuron. For the biological neuron, electrical signals from other neurons are conveyed to the cell body by dendrites; resultant electrical signals are sent along the axon to be distributed to other neurons. The operation of the artificial neuron is analogous to (though much simpler than) the operation of the biological neuron: activations from other neurons are summed at the neuron and passed through an activation function, after which the value is sent to other neurons.

3.3 Perceptrons

The development of a connectionist system capable of limited learning occurred in the late 1950's, when Rosenblatt created a system known as a perceptron. Again, this system consists of binary activations (inputs and outputs). In common with the McCulloch-Pitts neuron described above, the perceptron's binary output is determined by summing the products of inputs and their respective weight values. In the perceptron implementation, a

variable threshold value is used (whereas in the McCulloch-Pitts network, this threshold is fixed at 0): if the linear sum of the input/weight products is greater than a threshold value (θ), the output of the system is 1 (otherwise, a 0 is returned). The output unit is thus said to be, like the perceptron output unit, a linear threshold unit. To summarize, the perceptron “classifies” input values as either 1 or 0, according to the following rule.

3.4 The Delta Rule

The development of the perceptron was a large step toward the goal of creating useful connectionist networks capable of learning complex relations between inputs and outputs. In the late 1950's, the connectionist community understood that what was needed for the further development of connectionist models was a mathematically-derived (and thus potentially more flexible and powerful) rule for learning. By the early 1960's, the Delta Rule was invented. This rule is similar to the perceptron learning rule above, but is also characterized by a mathematical utility and elegance missing in the perceptron and other early learning rules. The Delta Rule uses the difference between target activation (i.e., target output values) and obtained activation to drive learning. For reasons discussed below, the use of a threshold activation function (as used in both the McCulloch-Pitts network and the perceptron) is dropped; instead, a linear sum of products is used to calculate the activation of the output neuron (alternative activation functions can also be applied). Thus, the activation function in this case is called a linear activation function, in which the output node's activation is simply equal to the sum of the network's respective input/weight products. The strengths of network's connections (i.e., the values of the weights) are adjusted to reduce the difference between target and actual output activation (i.e., error). A graphical depiction of a simple two-layer network capable of employing the Delta Rule is given in Fig 3. Note that such a network is not limited to having only one output node.

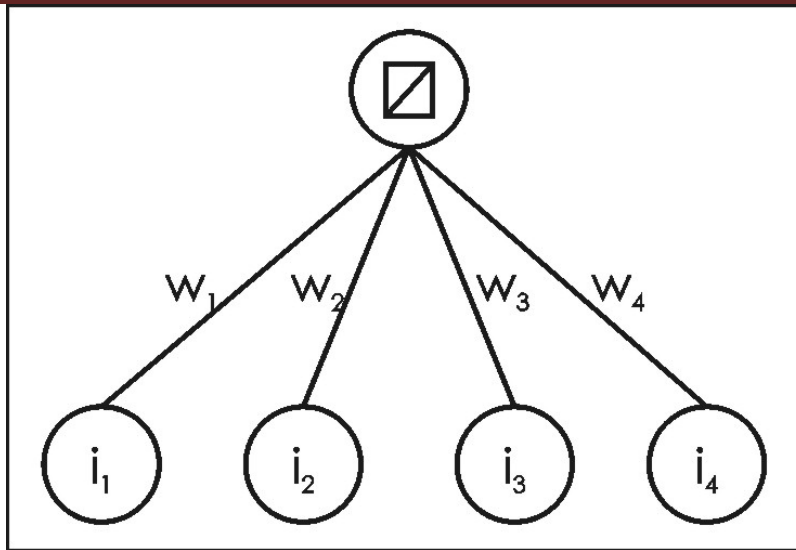


Fig 3.3 A network capable of implementing the Delta Rule. Non-binary values may be used. Weights are identified by w's, and inputs are identified by i's. A simple linear sum of products (represented by the symbol at top) is used as the activation function at the output node of the network shown here.

During forward propagation through a network, the output (activation) of a given node is a function of its inputs. The inputs to a node, which are simply the products of the output of preceding nodes with their associated weights, are summed and then passed through an activation function before being sent out from the node. Thus, we have the following:

$$S_j = \sum_i w_{ij} a_i \quad (3.1)$$

and

$$a_j = f(S_j) \quad (3.2)$$

where S_j is the sum of all relevant products of weights and outputs from the previous layer i , w_{ij} represents the relevant weights connecting layer i with layer j , a_i represents the activations of the nodes in the previous layer i , a_j is the activation of the node at hand, and f is the activation function.

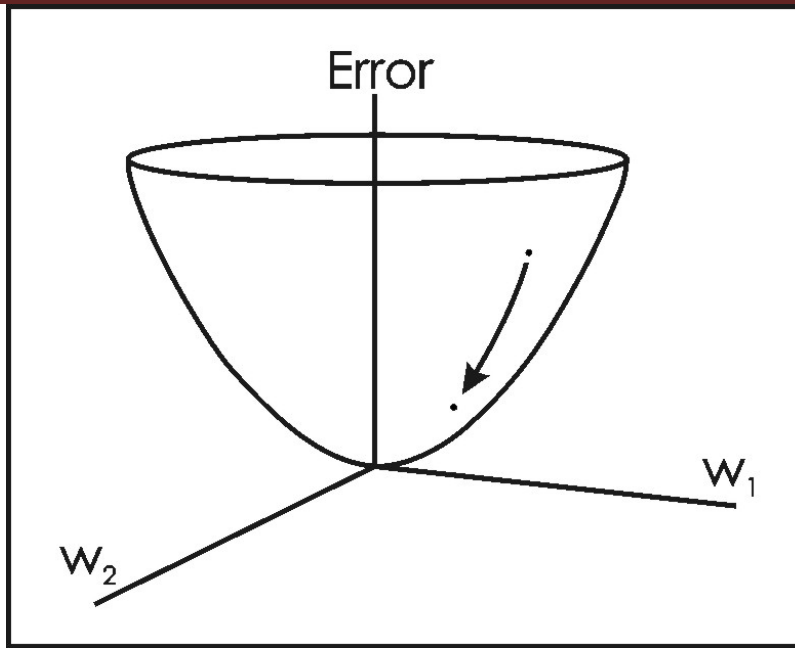


Fig 3.4: Schematic representation of an error function for a network containing only two weights (w_1 and w_2). Any given combination of weights will be associated with a particular error measure. The Delta Rule uses gradient descent learning to iteratively change network weights to minimize error (i.e., to locate the global minimum in the error surface).

For any given set of input data and weights, there will be an associated magnitude of error, which is measured by an error function (also known as a cost function) (Fig 3. 4). The Delta Rule employs the error function for what is known as gradient descent learning, which involves the modification of weights along the most direct path in weight-space to minimize error; change applied to a given weight is proportional to the negative of the derivative of the error with respect to that weight. The error function is commonly given as the sum of the squares of the differences between all target and actual node activations for the output layer. For a particular training pattern (i.e., training case), error is thus given by:

$$E_p = \frac{1}{2} \sum_n (t_{j_n} - a_{j_n})^2 \quad (3.3)$$

where E_p is total error over the training pattern, $\frac{1}{2}$ is a value applied to simplify the function's derivative, n represents all output nodes for a given training pattern, $t_{j \text{ sub } n}$ represents the target value for node n in output layer j , and $a_{j \text{ sub } n}$ represents the actual

activation for the same node. This particular error measure is attractive because its derivative, whose value is needed in the employment of the Delta Rule, is easily calculated. Error over an entire set of training patterns (i.e., over one iteration, or epoch) is calculated by summing all E_p :

$$E = \sum_p E_p = \frac{1}{2} \sum_p \sum_n (t_{j_n} - a_{j_n})^2 \quad (3.4)$$

where E is total error, and p represents all training patterns. An equivalent term for E in Equation 3.4 is sum-of-squares error. A normalized version of Equation 3.4 is given by the mean squared error (MSE) equation:

$$MSE = \frac{1}{2PN} \sum_p \sum_n (t_{j_n} - a_{j_n})^2 \quad (3.5)$$

where P and N are the total number of training patterns and output nodes, respectively. It is the error of Equations 3.4 and 3.5 that gradient descent attempts to minimize (in fact, this is not strictly true if weights are changed after each input pattern is submitted to the network). Error over a given training pattern is commonly expressed in terms of the total sum of squares (“tss”) error, which is simply equal to the sum of all squared errors overall output nodes and all training patterns. The negative of the derivative of the error function is required in order to perform gradient descent learning. The derivative of Equation 4a (which measures error for a given pattern p), with respect to a particular weight w_{ij} sub x, is given by the chain rule as:

$$\frac{\delta E_p}{\delta w_{ij_x}} = \frac{\delta E_p}{\delta a_{j_z}} \frac{\delta a_{j_z}}{\delta w_{ij_x}} \quad (3.6)$$

where a_{j_z} is the activation of the node in the output layer that corresponds to the weight w_{ij} sub x (note: subscripts refer to particular layers of nodes or weights, and the “sub-subscripts” simply refer to individual weights and nodes within these layers). It follows that

(3.7)

$$\frac{\delta E_p}{\delta a_{j_z}} = (2)\left(\frac{1}{2}\right)(t_{j_z} - a_{j_z})(-1) = -(t_{j_z} - a_{j_z})$$

and

(3.8)

$$\frac{\delta a_{j_z}}{\delta w_{ij_x}} = \frac{\delta}{\delta w_{ij_x}} \sum_n (w_{ij_n} a_{i_n}) = \frac{\delta}{\delta w_{ij_x}} (w_{ij_0} a_{i_0} + w_{ij_1} a_{i_1} \dots w_{ij_n} a_{i_n}) = a_{i_x}$$

Thus, the derivative of the error over an individual training pattern is given by the product of the derivatives of Equation 3.6:

(3.9)

$$\frac{\delta E_p}{\delta w_{ij_x}} = -(t_{j_z} - a_{j_z})(a_{i_x})$$

Because gradient descent learning requires that any change in a particular weight be proportional to the negative of the derivative of the error, the change in a given weight must be proportional to the negative of equation 3.9. Replacing the difference between the target and actual activation of the relevant output node by d , and introducing a learning rate ϵ , Equation 3.9 can be re-written in the final form of the delta rule:

(3.10)

$$\Delta w_{ij_x} = -\epsilon \frac{\delta E}{\delta w_{ij}} = \epsilon \delta a_{i_x}$$

The reasoning behind the use of a linear activation function here instead of a threshold activation function can now be justified: the threshold activation function that characterizes both the McCulloch and Pitts network and the perceptron is not differentiable at the transition between the activations of 0 and 1 (slope = infinity), and its derivative is 0 over the remainder of the function. As such, the threshold activation function cannot be used in gradient descent learning. In contrast, a linear activation

function (or any other function that is differentiable) allows the derivative of the error to be calculated.

Equation 3.10 is the Delta Rule in its simplest form. From Equation 3.10 it can be seen that the change in any particular weight is equal to the products of 1) the learning rate ϵ , 2) the difference between the target and actual activation of the output node $[d]$, and 3) the activation of the input node associated with the weight in question. A higher value for ϵ will necessarily result in a greater magnitude of change. Because each weight update can reduce error only slightly, much iteration is required in order to satisfactorily minimize error.

3.5 Multi-Layer Networks and Backpropagation

Eventually, despite the apprehensions of earlier workers, a powerful algorithm for apportioning error responsibility through a multi-layer network was formulated in the form of the backpropagation algorithm. The backpropagation algorithm employs the Delta Rule, calculating error at output units in a manner analogous, while error at neurons in the layer directly preceding the output layer is a function of the errors on all units that use its output. The effects of error in the output node(s) are propagated backward through the network after each training case. The essential idea of backpropagation is to combine a non-linear multi-layer perceptron-like system capable of making decisions with the objective error function of the Delta Rule.

3.6 Network Terminology

A multi-layer, feedforward, backpropagation neural network is composed of 1) an input layer of nodes, 2) one or more intermediate (hidden) layers of nodes, and 3) an output layer of nodes (Fig 3.1). The output layer can consist of one or more nodes, depending on the problem at hand. In most classification applications, there will either be a single output node (the value of which will identify a predicted class), or the same number of nodes in the output layer as there are classes (under this latter scheme, the predicted class for a given set of input data will correspond to that class associated with the output node with the highest activation). It is important to recognize that the term “multi-layer” is often used to refer to multiple layers of weights. This contrasts with the usual meaning of “layer”, which refers to a row of nodes. For clarity, it is often best to describe a particular network by its number of layers, and the number of nodes in each layer (e.g., a “4-3-5”

network has an input layer with 4 nodes, a hidden layer with 3 nodes, and an output layer with 5 nodes).

3.7 The Sigma Function

The use of a smooth, non-linear activation function is essential for use in a multi-layer network employing gradient-descent learning. An activation function commonly used in backpropagation networks is the sigma (or sigmoid) function:

$$a_{j_m} = \frac{1}{(1 + e^{-S_{j_m}})} \quad \text{where } S_{j_m} = \sum_{x=0}^n w_{ij_x} a_{i_x} \quad (3.11)$$

where a_{j_m} is the activation of a particular “receiving” node m in layer j , S_j is the sum of the products of the activations of all relevant “emitting” nodes (i.e., the nodes in the preceding layer i) by their respective weights, and w_{ij} is the set of all weights between layers i and j that are associated with vectors that feed into node m of layer j . This function maps all sums into $[0, 1]$ (Fig 3.1). If the sum of the products is 0, the sigma function returns 0.5. As the sum gets larger the sigma function returns values closer to 1, while the function returns values closer to 0 as the sum gets increasingly negative. The derivative of the sigma function with respect to S_{j_m} is conveniently simple as:

$$\frac{d}{d(S_{j_m})} (1 + e^{-S_{j_m}})^{-1} = -1 (1 + e^{-S_{j_m}})^{-2} e^{-S_{j_m}} (-1) = \frac{1}{1 + e^{-S_{j_m}}} \left(1 - \frac{1}{1 + e^{-S_{j_m}}}\right) = a_{j_m} (1 - a_{j_m}) \quad (3.12)$$

The sigma function applies to all nodes in the network, except the input nodes, whose values are assigned input values. The sigma function superficially compares to the threshold function (which is used in the perceptron) as shown in Fig 3.10. Note that the derivative of the sigma function reaches its maximum at 0.5, and approaches its minimum with values approaching 0 or 1. Thus, the greatest change in weights will occur with values near 0.5, while the least change will occur with values near 0 or 1.

3.8 The Backpropagation Algorithm

In the employment of the backpropagation algorithm, each iteration of training involves the following steps: 1) a particular case of training data is fed through the network in a forward direction, producing results at the output layer, 2) error is calculated at the output nodes based on known target information, and the necessary changes to the weights that lead into the output layer are determined based upon this error calculation, 3) the changes to the weights that lead to the preceding network layers are determined as a function of the properties of the neurons to which they directly connect (weight changes are calculated, layer by layer, as a function of the errors determined for all subsequent layers, working backward toward the input layer) until all necessary weight changes are calculated for the entire network. The calculated weight changes are then implemented throughout the network, the next iteration begins, and the entire procedure is repeated using the next training pattern. In the case of a neural network with hidden layers, the backpropagation algorithm is given by the following three equations, where i is the “emitting” or “preceding” layer of nodes, j is the “receiving” or “subsequent” layer of nodes, k is the layer of nodes that follows j (if such a layer exists for the case at hand), ij is the layer of weights between node layers i and j , jk is the layer of weights between node layers j and k , weights are specified by w , node activations are specified by a , delta values for nodes are specified by δ , subscripts refer to particular layers of nodes (i, j, k) or weights (ij, jk), “sub-subscripts” refer to individual weights and nodes in their respective layers, and epsilon is the learning rate:

(3.13)

$$\Delta w_{ij_m} = \varepsilon \delta_{j_p} a_{i_q}$$

(3.14)

$$\text{where } \delta_{j_p} = a_{j_p} (1 - a_{j_p}) (t_{j_p} - a_{j_p}) \text{ if output node}$$

(3.15)

$$\text{where } \delta_{j_p} = a_{j_p} (1 - a_{j_p}) \sum_{x=0}^n \delta_{k_x} w_{jk_x} \text{ if intermediate node}$$

Being based on the generalized Delta Rule, it is not surprising that Equation (3.13) has the same form as Equation (3.10). Equation (3.13) states that the change in a given weight m located between layers i and j is equal to the products of: 1) the learning rate (epsilon); 2) the delta value for node p in layer j [where node p is the node to which the vector associated with weight m leads]; and 3) the activation of node q in layer i [where node q is the node from which the vector associated with weight m leads]. In practice, the learning rate (epsilon) is typically given a value of 0.1 or less; higher values may provide faster convergence on a solution, but may also increase instability and may lead to a failure to converge. The delta value for node p in layer j in Equation (3.13) is given either by Equation (3.14) or by Equation (3.15), depending on whether or not the node is in an output or intermediate layer. Equation (3.14) gives the delta value for node p of layer j if node p is an output node. Together, Equations (3.13) and (3.14) were derived through exactly the same procedure as Equation 3.10, with the understanding that a sigma activation function is used here instead of a simple linear activation function (use of a different activation function will typically change the value of d). Both sets of equations were determined by finding the derivative of the respective error functions with respect to any particular weight. Equation (3.15) gives the delta value for node p of layer j if node p is an intermediate node (i.e., if node p is in a hidden layer). This equation states that the delta value of a given node of interest is a function of the activation at that node ($a_{j \text{ sub } p}$), as well as the sum of the products of the delta values of relevant nodes in the subsequent layer with the weights associated with the vectors that connect the nodes.

3.8 Bias

Equations (3.13), (3.14), and (3.15) describe the main implementation of the backpropagation algorithm for multi-layer, feedforward neural networks. It should be noted, however, that most implementations of this algorithm employ an additional class of weights known as biases. Biases are values that are added to the sums calculated at each node (except input nodes) during the feedforward phase. That is, the bias associated with a particular node is added to the term S_j in Equation (3.1), prior to the use of the activation function at that same node. The negative of a bias is sometimes called a threshold.

For simplicity, biases are commonly visualized simply as values associated with each node in the intermediate and output layers of a network, but in practice are treated in

exactly the same manner as other weights, with all biases simply being weights associated with vectors that lead from a single node whose location is outside of the main network and whose activation is always 1 (Fig 5). The change in a bias for a given training iteration is calculated like that for any other weight [using Equations (3.13), (3.14), and (3.15)], with the understanding that a_i sub m in Equation (3.13) will always be equal to 1 for all biases in the network. The use of biases in a neural network increases the capacity of the network to solve problems by allowing the hyperplanes that separate individual classes to be offset for superior positioning.

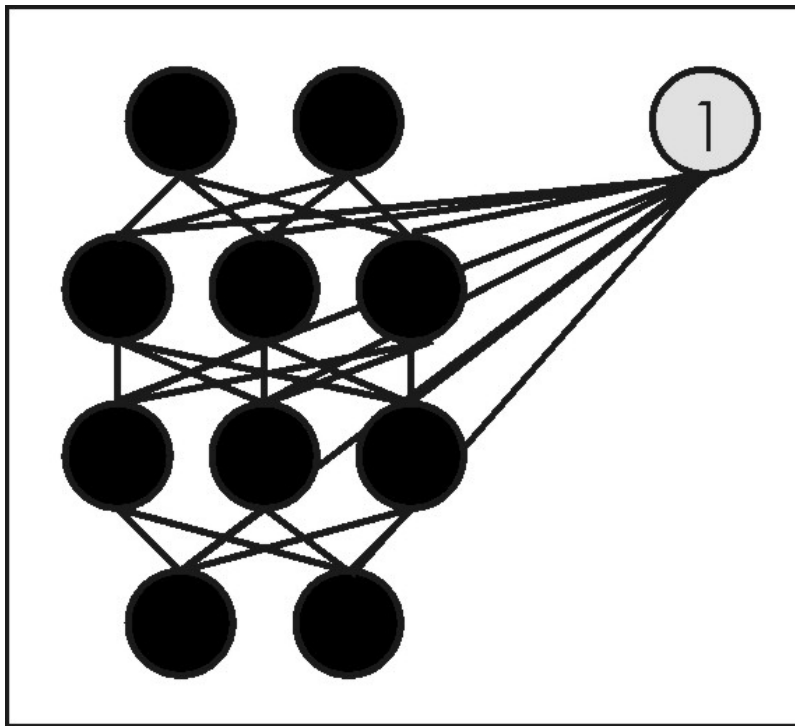


Fig 3.5: Biases are weights associated with vectors that lead from a single node whose location is outside of the main network and whose activation is always 1.

3.9 Network Topology

The precise network topology required to solve a particular problem usually cannot be determined, although research efforts continue in this regard. This is a critical problem in the neural-network field, since a network that is too small or too large for the problem at hand may produce poor results. This is analogous to the problem of curve fitting using polynomials: a polynomial with too few coefficients cannot evaluate a function of interest, while a polynomial with too many coefficients will fit the noise in the data and produce a poor representation of the function. General “rules of thumb” regarding

network topology are commonly used. At least one intermediate layer is always used, even simple problems such as the exclusive-OR problem cannot be solved without intermediate layers. Many applications of the backpropagation algorithm involve the use of networks consisting of only one intermediate layer of nodes, although the use of two intermediate layers can generate superior results for certain problems in which higher order functions are involved. The number of nodes used in each intermediate layer is typically between the number of nodes used for the input and output layers. An experimental means for determining an appropriate topology for solving a particular problem involves the training of a larger-than-necessary network, and the subsequent removal of unnecessary weights and nodes during training. This approach, called pruning, requires advance knowledge of initial network size, but such upper bounds may not be difficult to estimate. An alternative means for determining appropriate network topology involves algorithms which start with a small network and build it larger; such algorithms are known as constructive algorithms. Additionally, much neural network research remains focussed on the use of evolutionary and genetic algorithms, based on simplified principles of biological evolution, to determine network topology, weights, and overall network behavior.

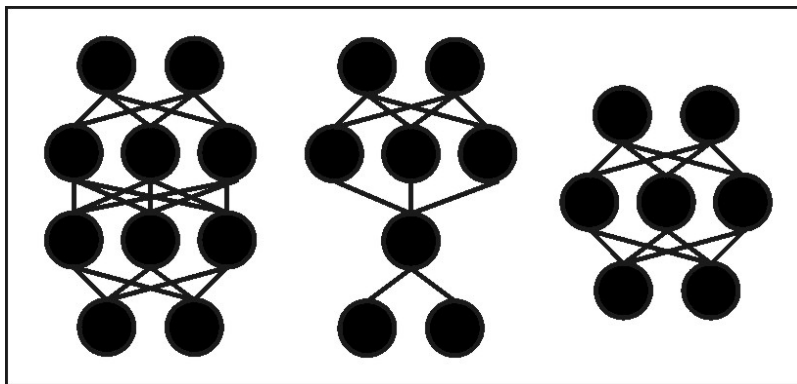


Fig 3.6: Three of an infinite number of possible network topologies that could be used to relate two inputs to two outputs.

CHAPTER 4

IMPLEMENTATION

Implementation literally means to put into effect or to carry out. The system implementation phase of the software deals with the translation of the design specifications into the source code. The ultimate goal of the implementation is to write the source code and the internal documentation so that it can be verified easily. The code and documentation should be written in a manner that eases debugging, testing and modification. A post-implementation review is an evaluation of the extent to which the system accomplishes stated objectives and actual project costs exceed initial estimates. It is usually a review of major problems that need converting and those that surfaced during the implementation phase.

After the system is implemented and conversion is complete, a review should be conducted to determine whether the system is meeting expectations and where improvements are needed. A post implementation review measures the systems performance against predetermined requirements. It determines how well the system continues to meet performance specifications. It also provides information to determine whether major re-design or modification is required.

4.1 DATABASE

4.1.1 SQLite

SQLite is a software library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. SQLite is the most widely deployed SQL database engine in the world.

It is a relational database management system contained in a small (~350 KB)^[4] C programming library. In contrast to other database management systems, SQLite is not a separate process that is accessed from the client application, but an integral part of it.

SQLite is ACID-compliant and implements most of the SQL standard, using a dynamically and weakly typed SQL syntax that does not guarantee the domain integrity.

SQLite is a popular choice as embedded database for local/client storage in application software such as web browsers. It is arguably the most widely deployed database engine, as it is used today by several widespread browsers, operating systems, and embedded systems, among others.^[5] SQLite has many bindings to programming languages.

SQLiteManager is a powerful database management system for sqlite databases, it combines an easy to use interface with blazing speed and advanced features. SQLiteManager allows you to work with a wide range of sqlite 3 databases (like plain databases, in memory databases, AES 128/256/RC4 encrypted databases and also with cubeSQL server databases). You can perform basic operations like create and browse tables, views, triggers and indexes in a very powerful and easy to use GUI. SQLiteManager's built-in Lua scripting language engine is flexible enough to let you generate reports or interact with sqlite databases in just about any way you can imagine.

4.1.2 Database design

Database is used to store the user details, the stocks he has bought or sold, the transactions he has performed and also used to historical values of the stock values for a set of 19 stocks.

The database schema design is shown below.

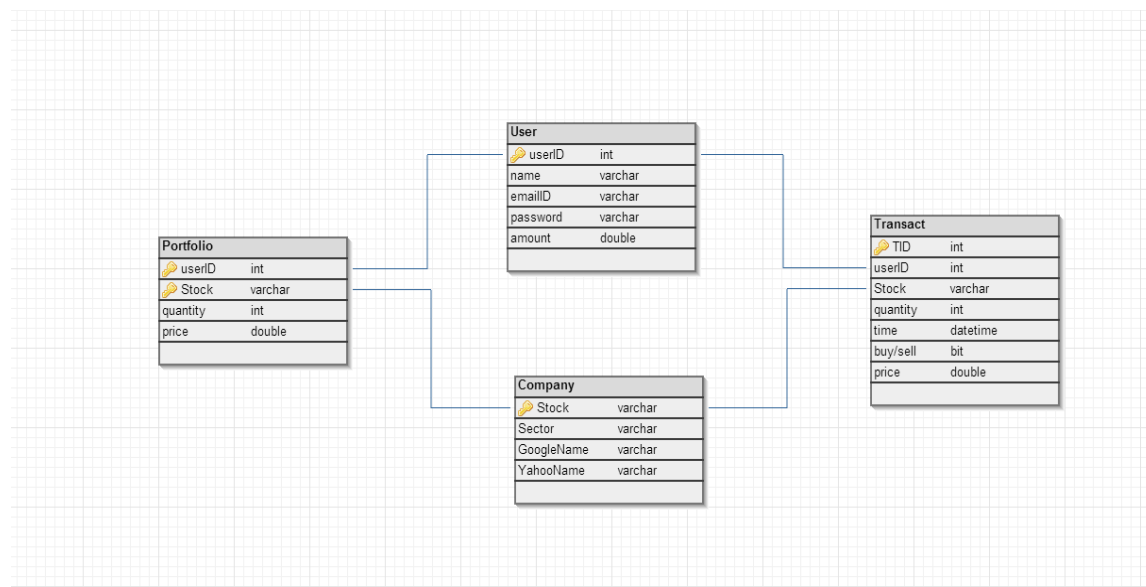


Fig 4.1 Database schema

4.2 Extraction of Stock Data

The extraction of historical prices was done by using the set of tools provided by finance.yahoo.com. The historical prices were obtained by framing the required url and using the java get URL code. The historical prices were downloaded in the csv format and then parsed to store the required data on to the database. Following are the set of variables used in extracting the historical prices

Start

To get the historical data we first start with the default base URL for historical quotes

<http://ichart.yahoo.com/table.csv?s=>

ID

Now, the ID of the stock or index required to be receive must be set. Every stock or index has their own ID. Also special characters have to be converted into the correct URL format. Historical quotes of several stocks or indices cannot be downloaded at once.

<http://ichart.yahoo.com/table.csv?s=GOOG>

From Date

The from date specifies the date from which the historical values must be downloaded. At first we have to add the number of the month minus 1.

<http://ichart.yahoo.com/table.csv?s=GOOG&a=2> .

Then add the number of the day.

<http://ichart.yahoo.com/table.csv?s=GOOG&a=0&b=15>

At last add the year.

<http://ichart.yahoo.com/table.csv?s=GOOG&a=0&b=1&c=2000>

To Date

Adding the "To Date" data uses the same design construct as from date.

Month minus 1 : <http://ichart.yahoo.com/table.csv?s=GOOG&a=0&b=1&c=2000&d=0>

Day. <http://ichart.yahoo.com/table.csv?s=GOOG&a=0&b=1&c=2000&d=0&e=31>

And the Year.

<http://ichart.yahoo.com/table.csv?s=GOOG&a=0&b=1&c=2000&d=0&e=31&f=2010>

4.3 Neuroph

Neuroph is lightweight Java neural network framework to develop common neural network architectures. It contains well designed, open source Java library with small number of basic classes which correspond to basic NN concepts. Also has nice GUI neural network editor to quickly create Java neural network components. It has been released as open source under the Apache 2.0 license.

Neuroph simplifies the development of neural networks by providing Java neural network library and GUI tool that supports creating, training and saving neural networks. Neuroph started as a graduate thesis project, after that a part of master theses, and on September 2008. it became open source project on SourceForge. After the initial release, development continued and several people helped to improve it in many ways. After a few months the version 2 was released, with many new features, optimized and cleaned code. After that it has been adopted for teaching neural networks during the Intelligent Systems course at Faculty of Organisational Sciences in Belgrade.

Diagram of Neuroph framework .This is the ' big picture' of framework, and it will help you understand what its basic components are and how it is organized.

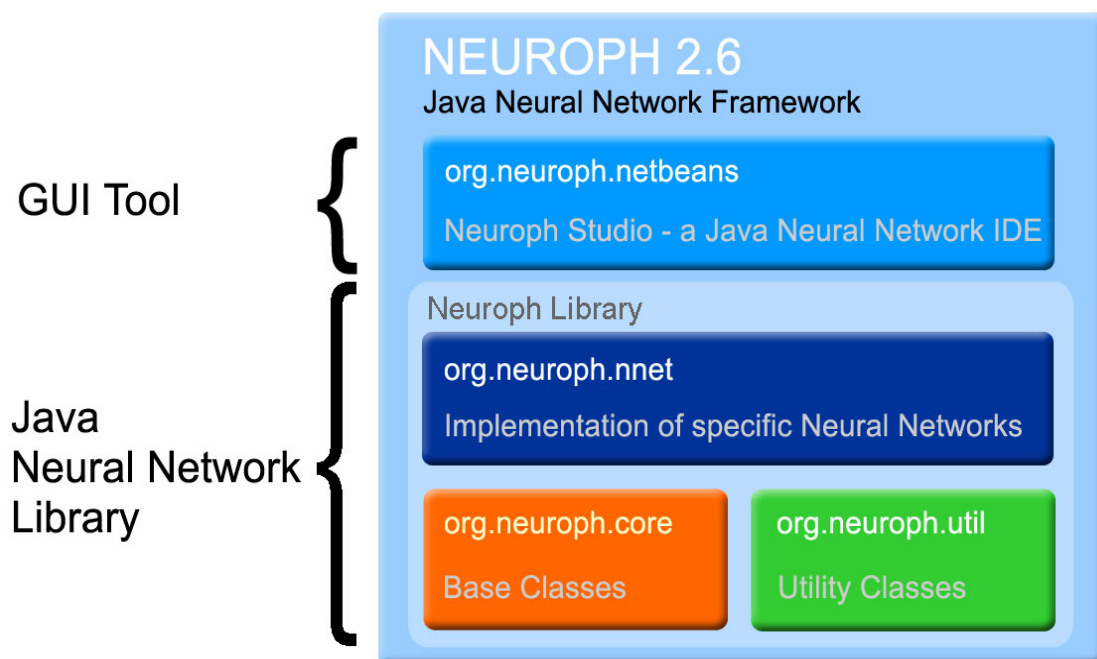


Fig. 4.2 Neuroph framework.

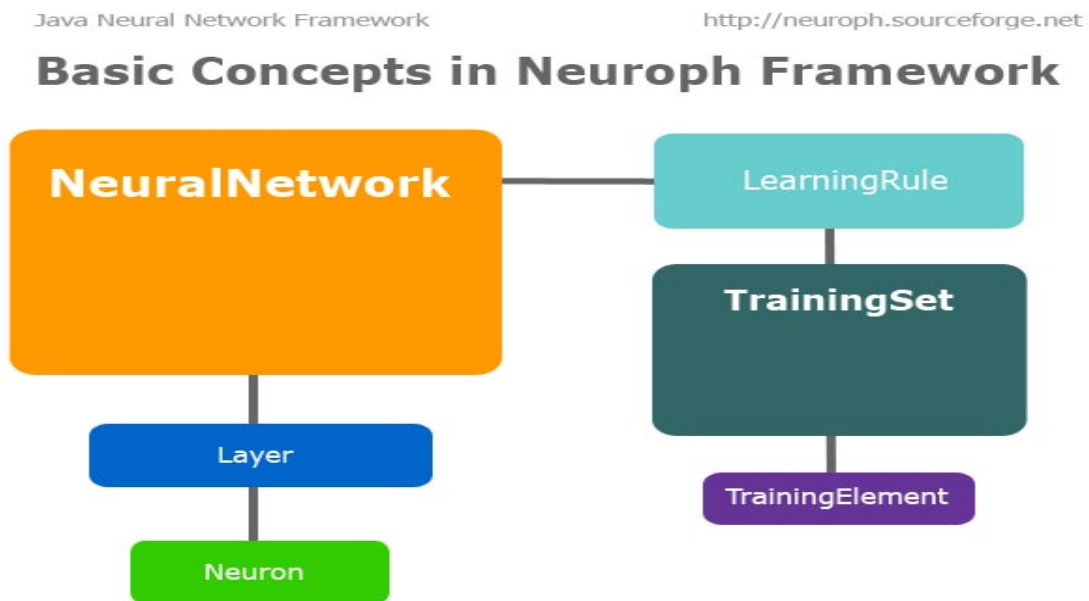


Fig. 4.3 Basic concepts in Neuroph Framework

4.4 Eclipse IDE

Eclipse is a multi-language software development environment comprising an integrated development environment (IDE) and an extensible plug-in system. It is written primarily in Java and can be used to develop applications in Java and, by means of the various plug-ins, in other languages as well, including C, C++, COBOL, Python, Perl, PHP, and others. The IDE is often called Eclipse ADT for Ada, Eclipse CDT for C, Eclipse JDT for Java and Eclipse PDT for PHP. The initial codebase originated from Visual Age. In its default form it is meant for Java developers, consisting of the Java Development Tools (JDT). Users can extend its capabilities by installing plug-ins written for the Eclipse software framework, such as development toolkits for other programming languages, and can write and contribute their own plug-in modules. Released under the terms of the Eclipse Public License, Eclipse is free and open source software.

IDE 1.0: Code completion is a very popular feature of modern IDEs, a life without which many developers find hard to imagine. One major reason for its popularity is that developers are frequently unaware of what methods they can invoke on a given variable. Here, code completion systems (CCSs) serve as an API browser, allowing developers to browse methods and select the appropriate one from the list of proposals. However, current completions are either computed by rather simplistic reasoning systems or are

simply hard-coded. For instance, for method completion, CCSs only consider the receiver's declared type. This often leads to an overwhelming number of proposals. Triggering code completion on a variable of `javax.swing.JButton` results in 381 method proposals. Clearly, developers only need a fraction of the proposed methods to make their code work. Code templates are an example for hard-coded proposals. Templates (like the Eclipse SWT Code Templates) serve as shortcuts and documentation for developers. Manual proposal definitions are labor intensive and error prone.

IDE 1.5: Researchers have recognized these issues. For instance, approaches exist that analyse client code to learn which methods the clients frequently use in certain contexts, and rearrange method proposals according to this notion of relevance [2]. Tools like XSnippet, Prospector and Parseweb [7 9 10] attempt to solve the issue of hard-coded code templates by also analyzing source code, identifying common patterns in code. Although obviously useful, these systems didn't made it into current IDEs. We argue that the primary reason for this is the lack of a continuously growing knowledge base. To build reliable models, source-code based approaches require example applications and full knowledge about the execution environment (i.e., classpath, library versions etc.). However, finding a sufficiently large set of example projects is difficult and tedious, and creating models for new frameworks is too time-consuming yet. While such approaches can sufficiently support a few selected APIs, we argue that they do not scale when tens of thousands of APIs should be supported.

IDE 2.0: So, how can we build continuously improving code completion systems then? To solve the scalability problem, code completion systems must allow users to share usage information among each other in an anonymized and automated way—from within the developer's IDE. This continuous data sharing allows recommender systems to learn models for every API that developers actually use. IDEs are very powerful when it comes to extracting information: they have access to information about the execution environment and about user interactions, even with respect to certain APIs. But the new, massive data sets derived from this information pose a challenge. We will likely require new algorithms to find reliable and valuable patterns in this data. Whatever means future code completion systems will use to build better recommendation models, the systems will be based on shared data. It will be the users who provide this data, and it is important to realize that, as the user base grows, the recommendation systems will be able to continuously improve over time, making intelligent completions that are useful for novice developers and experts alike.

4.5 Java

Java is a programming language originally developed by James Gosling at Sun Microsystems (which is now a subsidiary of Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++ but has a simpler object Model and fewer low-level facilities. Java applications are typically compiled to byte code (class file) that can run on any Java Virtual Machine (JVM) regardless of computer Architecture.

Java is general-purpose, concurrent, class-based, and object-oriented, and is specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere". Java is considered by many as one of the most influential programming languages of the 20th century, and is widely used from Application software to web applications.

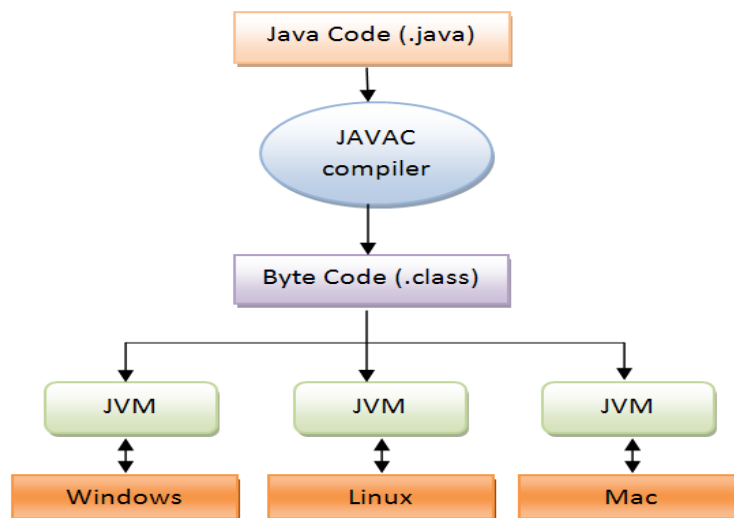


Fig 4.4: Java runtime environment

Java code is written in .java file. This code contains one or more Java language attributes like Classes, Methods, Variable, and Objects etc. Javac is used to compile this code and to generate .class file. Class file is also known as “byte code“. The name byte code is given may be because of the structure of the instruction set of Java program. We will see more about the instruction set later in this tutorial. Java byte code is an input to Java Virtual Machine. JVM read this code and interpret it and executes the program.

There were five primary goals in the creation of the Java language:

- It should be "simple, object-oriented and familiar"
- It should be "robust and secure"
- It should be "architecture-neutral and portable"
- It should execute with "high performance"
- It should be "interpreted, threaded, and dynamic"

4.5.1 Java Platform, Enterprise Edition

Java Platform, Enterprise Edition or Java EE is Oracle's enterprise Java computing platform. The platform provides an API and runtime environment for developing and running enterprise software, including network and web services, and other large-scale, multi-tiered, scalable, reliable, and secure network applications. Java EE extends the Java Platform, Standard Edition providing and web services. The platform incorporates a design based largely on modular components running on an application server. Software for Java EE is primarily developed in the Java programming language. The platform emphasizes Convention over configuration and annotations for configuration.

4.5.2 Web Applications Container

A Web application contains an application's resources, such as servlets, JavaServer Pages (JSPs), JSP tag libraries, and any static resources such as HTML pages and image files. The main role of the container is to facilitate the various services of dynamic web applications.

4.5.3 Java Web application

A Java web application is a collection of dynamic resources (such as Servlets, JavaServer Pages, Java classes and jars) and static resources (HTML pages and pictures). A Java web application can be deployed as a ".war" file. The ".war" file is a zip file which contains the complete content of the corresponding web application.

4.5.4 Servlets

A servlet is a Java class that runs in a Java-enabled server. An HTTP servlet is a special type of servlet that handles an HTTP request and provides an HTTP response, usually in the form of an HTML page. The most common use of HTTP servlets is to create

interactive applications using standard Web browsers for the client-side presentation while Server handles the business logic as a server-side process. HTTP servlets can access databases, Enterprise JavaBeans, messaging APIs, HTTP sessions, and other facilities of Server.

4.5.5 Java Server Pages

JavaServer Pages (JSPs) are a specification for combining Java with HTML to provide dynamic content for Web pages. When you create dynamic content, JSPs are more convenient to write than HTTP servlets because they allow you to embed Java code directly into your HTML pages, in contrast with HTTP servlets, in which you embed HTML inside Java code.

JSPs are Web pages coded with an extended HTML that makes it possible to embed Java code in a Web page. JSPs can call custom Java classes, called taglibs, using HTML-like tags. The

JSPs enable you to separate the dynamic content of a Web page from its presentation. It caters to two different types of developers: HTML developers, who are responsible for the graphical design of the page, and Java developers, who handle the development of software to create the dynamic content.

Overview of How JSP Requests Are Handled

- WebLogic Server handles JSP requests in the following sequence:
- A browser requests a page with a .jsp file extension from WebLogic Server.
- WebLogic Server reads the request.
- Using the JSP compiler, WebLogic Server converts the JSP into a servlet class that implements the `javax.servlet.jsp.JspPage` interface. The JSP file is compiled only when the page is first requested, or when the JSP file has been changed. Otherwise, the previously compiled JSP servlet class is re-used, making subsequent responses much quicker.
- The generated `JspPage` servlet class is invoked to handle the browser request.

4.5.6 Apache Tomcat

Apache Tomcat is an open source web server and servlet container developed by the Apache Software Foundation (ASF). Tomcat implements the Java Servlet and the JavaServer Pages (JSP) specifications from Sun Microsystems, and provides a "pure Java" HTTP web server environment for Java code to run.

Apache Tomcat includes tools for configuration and management, but can also be configured by editing XML configuration files.

Tomcat started off as a servlet reference implementation by James Duncan Davidson, a software architect at Sun Microsystems. He later helped make the project open source and played a key role in its donation by Sun Microsystems to the Apache Software Foundation.

4. 6 User Interface Implementation

The user interface was designed primarily keeping simplicity and the user in mind. Since our programming language is in Java, the UI files are in the form of JSP files for ease of access and compatibility with Java servlets. The user interface includes the following components:

Component	Functionality
Web Server	Apache Tomcat
Java Servlet Pages	Used for the web pages
HTML	Used within JSP files
CSS	Style sheets used. The main one is Bootstrap.
Bootstrap	A front end framework offered by Twitter which eases web development. Used here for providing design and ease of access in all pages.
Javascript	Used as a scripting language to provide various functionality including validating forms and interacting with the server.
Jquery	A Javascript library which further eases JS programming .
Datatables	A plugin to the Jquery library which provides

	flexible tables with interactive controls. Used for the user portfolio.
HighCharts	A JS library used to provide interactive graphs and pie charts . Used to provide graphs for the entire stock market as well as analyze the user portfolio.

Table 4.1 Components and Functionalities

CSS - Twitter Bootstrap:

Twitter offers a powerful and flexible web development framework called Bootstrap.

Most of the websites today in the WWW use Bootstrap for their web page design.

Use in Project:

We've used several elements of Twitter Bootstrap such as:

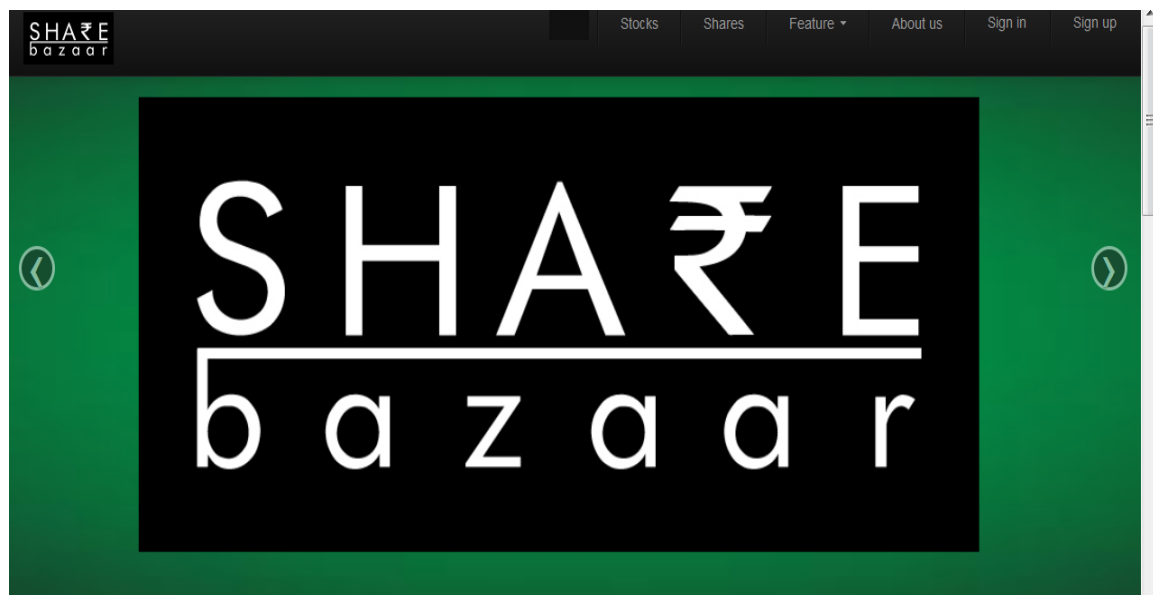


Fig.4.5 Carousel: Used to create a slideshow displaying various features about our application

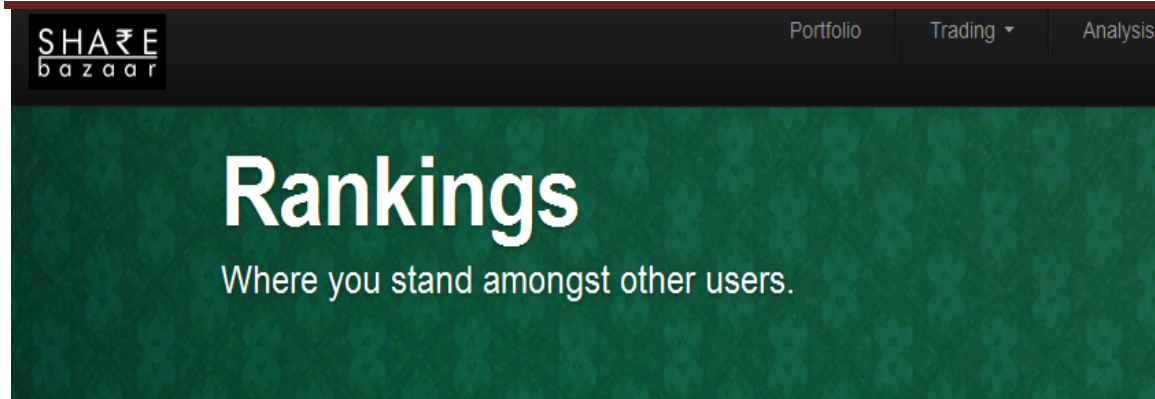


Fig 4.6 Jumbrotron Subhead: Used to display headings of various pages.

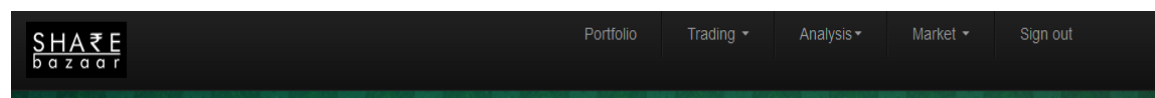


Fig 4.7 Navbar.

Other CSS:

We've used an alternative CSS file to provide the dynamic stock viewing option which is called the Cloud Overlay Effect. This displays all stocks in a blocks format along with their current market price and recent changes(+ or -) and these values are updated every 30 seconds.

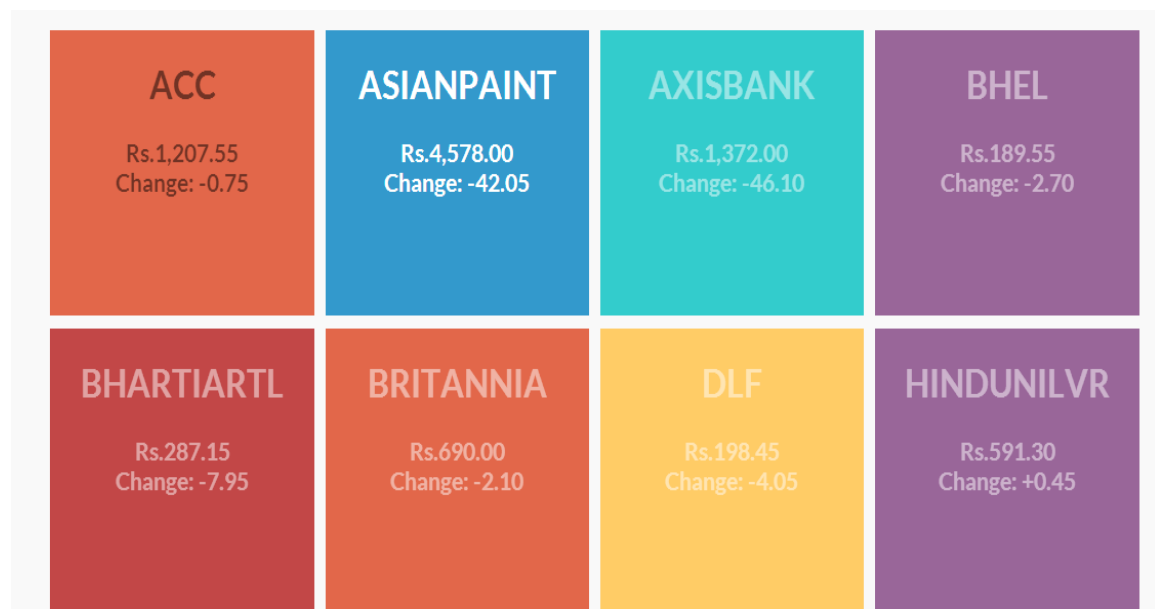


Fig 4.8. Cloud Overlay Effect.

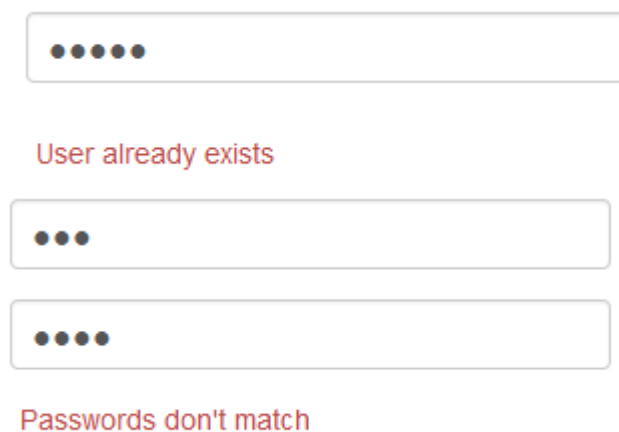
JavaScript:

JavaScript is a scripting language. A scripting language is a lightweight programming language so basically, JS is programming code that can be inserted into HTML pages.

This can be executed by all modern web browsers. They provide code to interact with the server, validate forms, hide/display a container, change HTML styles and much more.

Use in project:

We've used JavaScript in almost all our web pages. Bootstrap offers a JS file to provide functionality for its UI features. jQuery and jQuery UI also provide their own JS files. We have included JS to validate sign up, sign in forms and obtain the quote for a stock and retrieve the news feed etc.



The image shows a web form with three input fields. The first field has a red error message 'User already exists' below it. The second and third fields are password inputs, each with a red error message 'Passwords don't match' below them. The input fields are white with a light gray border and contain five dots representing masked characters.

Fig 4.9. Validating forms:

jQuery:

jQuery is a JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. jQuery UI is a set of widgets and interaction elements such as date pickers, autocomplete, accordion boxes built on top of the jQuery library.

Eg:

```
<script>
$(document).ready(function () {
    readValues();
});
</script>
```

This jQuery calls the function readValues() as soon as the page loads.

Eg:

Another JQuery UI feature is to provide “auto-complete” for text boxes , especially useful when a user doesn’t know the exact name of a stock . As soon as the user types in a letter, all companies including that letter are displayed for the user to select.

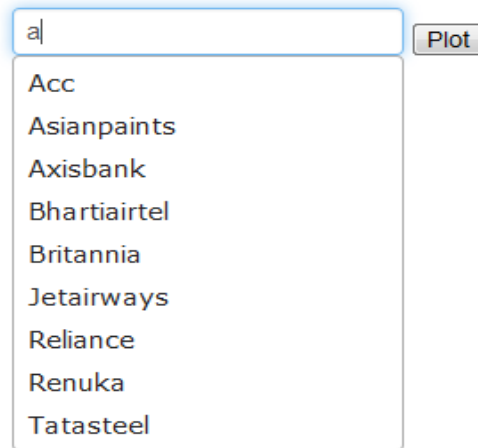


Fig. 4.10 AutoComplete

DataTables is a plug-in for the jQuery Javascript library. It is a highly flexible tool, based upon the foundations of progressive enhancement, which will add advanced interaction controls to any HTML table. Key features:

- Variable length pagination
- On-the-fly filtering
- Multi-column sorting with data type detection
- Smart handling of column widths
- Display data from almost any data source

10
▼
records per page
Search:

Number	User Id	Name	Quantity	Price
1	12	Asianpaint	55	4678.02490234375
2	12	Dlf	20	178.79249572753906
3	12	Infy	11	41.279998779296875
4	12	Tatasteel	5	305.20001220703125
5	12	aapl	100	442.7799987792969
6	12	goog	10	824.5700073242188
7	12	ibm	2	202.5399932861328

Showing 1 to 7 of 7 entries

← Previous
1
Next →

10
▼
records per page
Search:

Name	Predicted value	Rise/Fall
ACC	1213.510109387353	▲
Airtel	306.2286811798481	▼
ASIANPAINTS	4777.980210732874	▲
AxisBank	1393.7349891132185	▼
BHEL	196.15956426995479	▲
BRITANNIA	540.1419010188855	▼
DLF	257.56133321853304	▲
HIND_UNILEVER	555.8353479946852	▼
INFY	2658.8413822481825	▲
JET	584.8019669298745	▲

Showing 1 to 10 of 19 entries

← Previous
1
2
Next →

Fig. 4.11 Datatables

Highcharts is a charting library written in pure HTML5/JavaScript, offering intuitive, interactive charts to a web site or web application. Highcharts currently supports line, spline, area, areaspline, column, bar, pie, scatter, angular gauges, arearange, areasplinerange, columnrange, bubble, box plot, error bars, funnel, waterfall and polar chart types.

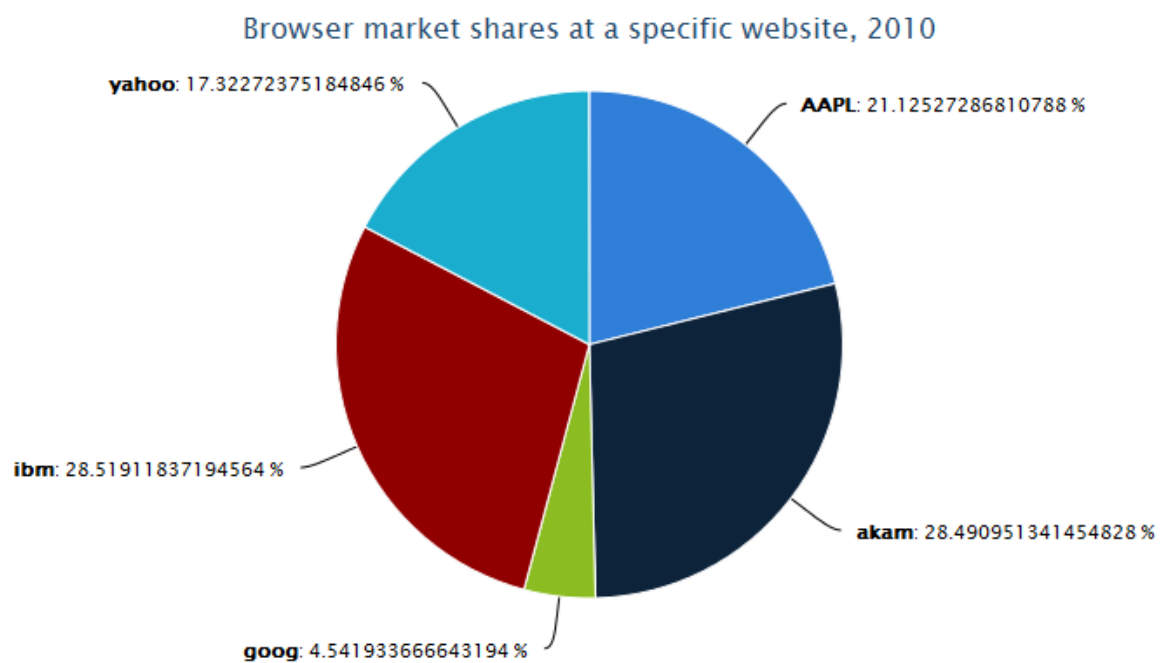


Fig. 4.12 Highcharts

CHAPTER 5

TESTING

5.1 TESTING PROCESS

After designing phase there is the coding phase. In this phase, every module identified and specified in the design document is independently Coded and Unit tested .Unit testing (or module testing) is the testing of different units or modules of a system. In this phase, the physical design of the system is converted into the logical programming language.

5.2 TESTING OBJECTIVES

The coding is done in java before starting of the coding, we have tried to follow some coding standards and Guidelines.

The coding standards are: -

- Naming standards for the java Classes and variables etc.
- Screen design standards.
- Validation and checks that need to be implemented.

The Guidelines are: -

- Code should be well document.
- Coding style should be simple.
- Length of function should short.

5.3 LEVELS OF TESTING

5.3.1 UNIT TESTING

In this, the programs that made up the system were tested. This is also called as program testing. This level of testing focuses on the modules, independently of one another.

The purpose of unit testing is to determine the correct working of the individual modules. For unit testing, we first adopted the code testing strategy, which examined the logic of program. During the development process itself all the syntax errors etc. got rooted out. For this we developed test case that results in executing every instruction in

the program or module i.e. every path through program was tested. (Test cases are data chosen at random to check every possible branch after all the loops.).Unit testing involves a precise definition of test cases, testing criteria, and management of test cases.

5.3.1.1 USER INPUT

In User interface the data entry is done through GUI and tested. Each element is tested for valid range and invalid range of data.

5.3.1.2 ERROR HANDLING

In this system we have tried to handle all the errors that are occurred while running the GUI forms. The common errors we saw are reading the empty record and displaying a compiler message, etc.

We have handled errors using exception handling. Many errors have been handled using try catch blocks provided by java. Most of the errors occurred while committing data into database these have been handled with utmost care.

5.3.2 INTEGRATION TESTING

In this the different modules of a system are integrated using an integration plan. The integration plan specifies the steps and the order in which modules are combined to realize the full system. After each integration step, the partially integrated system is tested. The primary objective of integration testing is to test the module interface.

An important factor that guides the integration plan is the module dependency graph. The module dependency graph denotes the order in which different modules call each other. A structure chart is a form of a module dependency graph. Thus, by examining the structure chart the integration plan can be developed based on any of the following approaches:

- Big-bang approach.
- Top-down approach.
- Bottom-up approach.
- Mixed approach.

5.3.2.1 BOTTOM UP INTEGRATION TESTING

In this approach, each subsystem is tested separately and then the full system is tested. A subsystem might consist of many modules, which communicate among each other through well-defined interfaces. The primary purpose of testing each subsystem is to test the interfaces among various modules making up the subsystem. Both control and data interface is tested. A principal advantage of bottom-up integration testing is that several disjoint subsystems can be tested simultaneously. A disadvantage of bottom-up testing is the complexity that occurs when the system is made up of large number of small subsystems.

In Main module, we have tested all the individual programs first and after having successful results in the individual program testing we moved further for the integration .We have combined some programs and then tested it, after having good results; we have combined all the programs together and started for system testing.

5.3.3 SYSTEM TESTING

Once we are satisfied that all the modules work well in themselves and there are no problems, we do in to how the system will work or perform once all the modules are put together. The main objective is to find discrepancies between the system and its original objective, current specifications, and system documentation. Analysts try to find molds that have been designed with different specifications, which could cause incompatibility.

At this stage the system is used experimentally to ensure that all the requirements of the user are fulfilled. At this point of the testing takes place at different levels so as to ensure that the system is free from failure. Testing is vital to success of the system. System testing makes a logical assumption that whether all parts of the system are correct. Initially the system was given to the user for entry validation was provided at each and every stage.

So that the user is not allowed to enter unrelated data. The training is given to user about how to make an entry. While implementing the system it was observed that the user was initially resisting the change, however the system being the need of the hour and user friendly, the fear was overcome. Entering live data of the past months records was little tedious, prior to the actual day to day transaction

The best test made on the system was whether it produces the correct outputs. All the outputs were checked out and were found to be correct. Feedback sessions were

conducted and the suggested changes given by the user were made before the acceptance test. Finally the system is being accepted and made to run with live data. System tests are designed to validate a fully developed system with a view to assuring that it meets its requirements.

There are three main kinds of system testing:

- Alpha Testing.
- Beta Testing.
- Acceptance Testing.

5.3.3.1 ALPHA TESTING: This refers to the system testing that is carried out by the test team with the organization.

5.3.3.2 BETA TESTING: This refers to the system testing that is performed by a select group of friendly customers.

5.3.3.3 ACCEPTANCE TESTING: This refers to the system testing that is performed by the customer to determine whether or not to accept the delivery of the system.

5.4 TEST RESULT FOR THE PREDICTIONS.

In the following graphs blue line is the line plotted from actual values and black line is the line plotted from the predicted values. This is done only using the historical values.

Prediction for the year of 2012. Stock – Airtel.



Fig 5.1 Prediction Performance 1

Prediction for the year of 2012. Stock –BHEL.



Fig 5.2 Prediction Performance 2

Prediction for the year of 2012. Stock –DLF.



Fig 5.3 Prediction Performance 3

Prediction for the year of 2012. Stock –RENUKA.

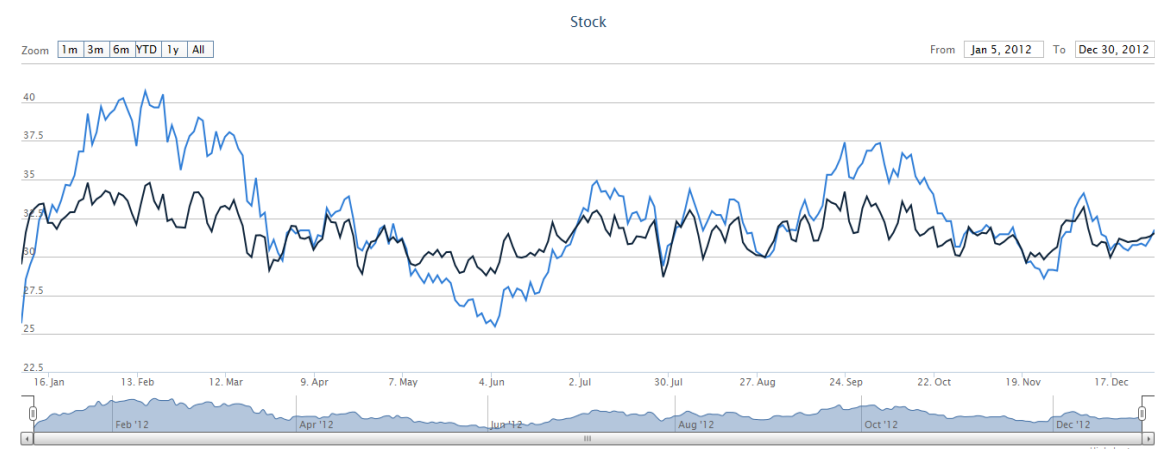


Fig 5.4 Prediction Performance 4

Prediction for the year of 2012. Stock –WIPRO.**Fig 5.5 Prediction Performance 5**

CHAPTER 6

SENTIMENT ANALYSIS

Sentiment scoring is emotion! Is the text positive or negative? Good or bad? How will you know if you don't look? How can you look by hand if its millions of comments per hour?

The obvious limitations of manual sentiment scoring and sentiment analysis led to the development of machine scoring. Once you have reliable, consistent machine-based sentiment scoring, there are a number of easy applications; reputation management (the problem every marketing person faces), “voice of customer” (listen to how they’re saying what they say, don’t constrain them to closed-ended questions), eDiscovery (was there a wave of negative emails before a certain crisis hit?), etc...

Lexalytics has provided text mining and sentiment analysis for over nine years now. As such, we have a substantial amount of experience and know where it works well. Unfortunately, many vendors haven’t been doing it as long as we have, so you need to be wary of over-reaching claims and confusing descriptions by the vendors rolling out early sentiment analysis engines.

Humans seemingly have no trouble reading a sentence and mentally scoring the sentiment. We humans use a process of reading and understanding the descriptors placed on the subject of a sentence.

Consider these sentences:

- A horrible pitching performance resulted in another devastating loss.
- Sub-par pitching and superb hitting combined to cost us another close game.

They both have the same basic subject, the loss of a baseball game, but obviously (to you!) the first sentence is contextually much more negative. The keys humans that humans use to discern this are to focus on the emotive phrases "horrible pitching" and "devastating loss". The sentiment system developed by Lexalytics does exactly the same thing.

Our code identifies the emotive phrases within a document and then scores these phrases (roughly -1 to +1) and then combines them to discern the overall sentiment of the sentence. Importantly, our sentiment scoring will score those sentences the same every time they're exposed to them – our software is not affected by whether or not it has had its morning coffee, or whether it is a fan of the team that lost (or that won!). That's consistency, and that's important.

Sentiment Analysis

Sentiment analysis aims to automatically identify the feeling, emotion, or intent behind a given text using various text-mining techniques. The most obvious advantage of these automated tools is their ability to evaluate large quantities of text, without manual intervention.

The academic field of sentiment analysis, and various associated software tools, are not a new phenomenon, and have been in mainstream commercial use since around 2001. Initially, these tools were used for analysis of sentiment surrounding specific products on popular product review sites. Later, they were applied to evaluating brand value for corporations. Most recently, tools to measure sentiment have been successfully and commercially applied to the stock market advice domain.

What is Sentiment?

Sentiment is defined as a thought, view, or attitude. Arguably, what people feel - and especially what this impels them to do - is the driving force behind human economics, and the glue that binds human society itself.

Thus, the importance of sentiment analysis – which enables large-scale understanding and clarity regarding the feelings of a group of people on a given subject – cannot be underestimated. For political, commercial, and even religious purposes, sentiment analysis tools can offer insight that is valuable both monetarily and socially.

Where is Sentiment?

Quantifiable sentiment analysis, based on a large-scale sentiment dictionary, has of course been facilitated by the ready availability of the immense and classifiable body of data that is the Internet.

In fact, the Internet has not only facilitated the development of sentiment analysis tools, it has actually necessitated it. For any given subject, there exists such a vast amount of information on the Internet as to make manual review and processing infeasible. Thus, in a very real sense, one of the few ways we can make sense of any subject based on web-originated data is via tools that measure sentiment.

How is Sentiment Analysis Used?

As mentioned above, there are number commercial uses for sentiment analysis tools. One field is stock market advice, in which these tools have been especially effective in sorting through and making sense of reams of data.

The stock market is an excellent example of a closed and measurable system that is almost entirely sentiment-driven. Investors know that the stock market isn't just about P/E, EPS, or Market Cap. It is the unstructured dinner party chat, the gossip, the rumor, the opinion, and the gut feelings that shape the market.

For this reason, software like The Stock Sonar has shown such promise in harnessing the fiscal power of sentiment. By effectively and accurately measuring the prevailing sentiment in forums, blogs, documents, Tweets, and mainstream and niche media, sentiment analysis tools like The Stock Sonar examine unstructured data from multiple sources, and then intelligently weigh and score the findings on a numeric scale. This ultimately offers investors an effective method to mine insights from analysts, media, social media, and company announcements – granting them a clearer understanding of the effect of sentiment on stock values, and delivering a competitive investing edge.

6.2 R (programming language)

R is a language and environment for statistical computing and graphics. It is a GNU project which is similar to the S language and environment which was developed at Bell Laboratories (formerly AT&T, now Lucent Technologies) by John Chambers and

colleagues. R can be considered as a different implementation of S. There are some important differences, but much code written for S runs unaltered under R.

R provides a wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering) and graphical techniques, and is highly extensible. The S language is often the vehicle of choice for research in statistical methodology, and R provides an Open Source route to participation in that activity.

R is an integrated suite of software facilities for data manipulation, calculation and graphical display. It includes

- an effective data handling and storage facility,
- a suite of operators for calculations on arrays, in particular matrices,
- a large, coherent, integrated collection of intermediate tools for data analysis,
- graphical facilities for data analysis and display either on-screen or on hardcopy, and
- a well-developed, simple and effective programming language which includes conditionals, loops, user-defined recursive functions and input and output facilities.

The term "environment" is intended to characterize it as a fully planned and coherent system, rather than an incremental accretion of very specific and inflexible tools, as is frequently the case with other data analysis software.

R, like S, is designed around a true computer language, and it allows users to add additional functionality by defining new functions. Much of the system is itself written in the R dialect of S, which makes it easy for users to follow the algorithmic choices made. For computationally-intensive tasks, C, C++ and Fortran code can be linked and called at run time. Advanced users can write C code to manipulate R objects directly.

Many users think of R as a statistics system. We prefer to think of it of an environment within which statistical techniques are implemented. R can be extended (easily) via *packages*. There are about eight packages supplied with the R distribution and many more are available through the CRAN family of Internet sites covering a very wide range of modern statistics.

R has its own LaTeX-like documentation format, which is used to supply comprehensive documentation, both on-line in a number of formats and in hardcopy.

6.3 Sentiment analysis with R

We have used R language to extract the news articles from moneycontrol.com, it contains important news for individual stocks. Moneycontrol is India's leading financial information source. Manage your finance with our online Investment Portfolio, Live Stock Price, Stock Trading news etc. The corpus is taken for INFOSYS for one year from Jan 2012 to Dec 2012.

6.3.1 Text Cleaning

It is mostly heuristic based and case specific. By this what we mean is to identify the unwanted portions in the extracted contents with respect to different kinds of web documents (e.g. News article, Blogs, Review, Micro Blogs etc) and then write simple cleanup codes based on that learning what , which will remove such unwanted portions with high accuracy.

6.3.2 Extract the Sentiment

We have a list of positive negative lexicon words Sentiment Analysis of web documents can be defined as the consumer opinion expressed through online medium e.g. Blog or review. Now days a consumer can choose to post his/her sentiment about a particular brand/product/feature online which can be categorized broadly as Positive, Negative or Neutral. The web documents where such sentiment has been expressed can be referenced for various analytical/actionable causes by that brand representative. For example, considerable amount of negative sentiment expressed by consumers about customer service of ICICI bank can be actionable insight for ICICI bank, in which case ICICI might want to restructure its customer service to give better customer satisfaction and thus tend to reduce negative sentiment about it in the net. Sentiment analysis can be done on the clean extracted web documents in two manners - manual rating or automated rating of such web documents. While manual rating is a near perfect method to do it, but it is a slow process when the volume of web documents is too high. Whereas automated system

will be much faster method, but is bound to lack accuracy since it is effectively machine learning and deriving human sentiments through user generated content. Also, language barrier is a major challenge for automated sentiment analysis. Nevertheless, extensive research work on Natural Language Processing has addressed such challenges well and reasonably high performance machine learning techniques have evolved which can do sentiment analysis of web documents. We compare the words in the text document with the list of positive and negative words. The number of positive or negative matches is calculated and the score for the text document is calculated.

The score is calculated according to the following formula:

$$\text{SCORE} = \frac{\sum(\text{positive matches}) - \sum(\text{negative matches})}{\sum(\text{positive matches}) + \sum(\text{negative matches})} \quad (8.3.1)$$

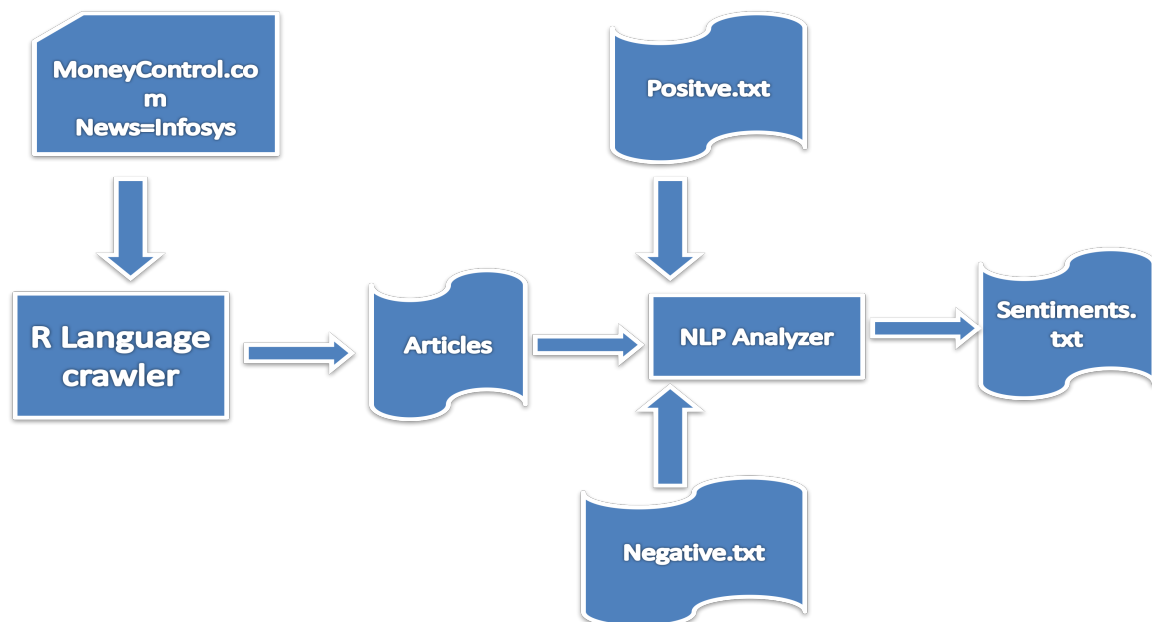


Fig. 6.1 Sentiment Analyzer framework

6.4 Neural Network and Sentiment Analysis.

We use the neuroph java framework to combine the sentiment and historical values to predict the next day's value. The neural network representation used for this is shown below.

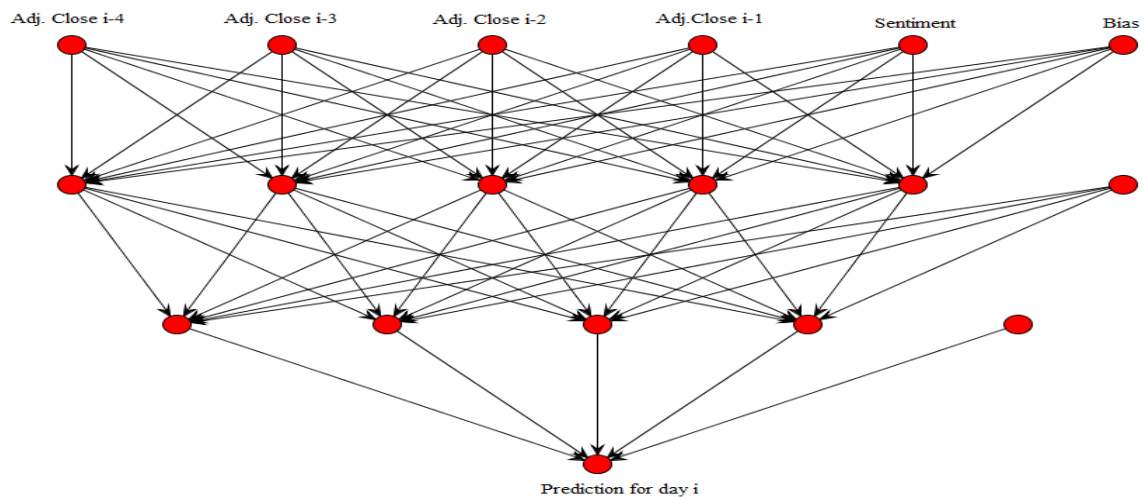


Fig. 6.2 Neural Network Layout

The complete process of using the news articles and the historical values is shown in the below figure.

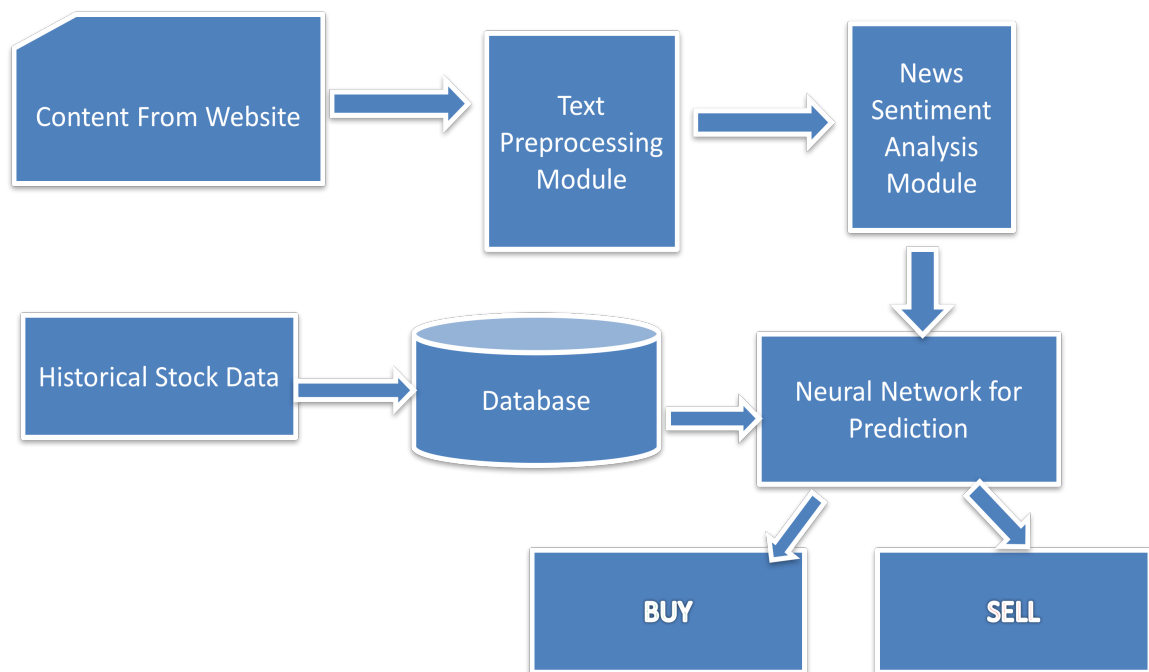


Fig. 6.3 Predictor Framework

6.5 RESULTS

The predicted gain or fall using the historical values and sentiments for the year of 2012 and for the stock of INFY was calculated for different number of nodes in the hidden layers is shown below. We have one input layer with 5 inputs and one output layer with

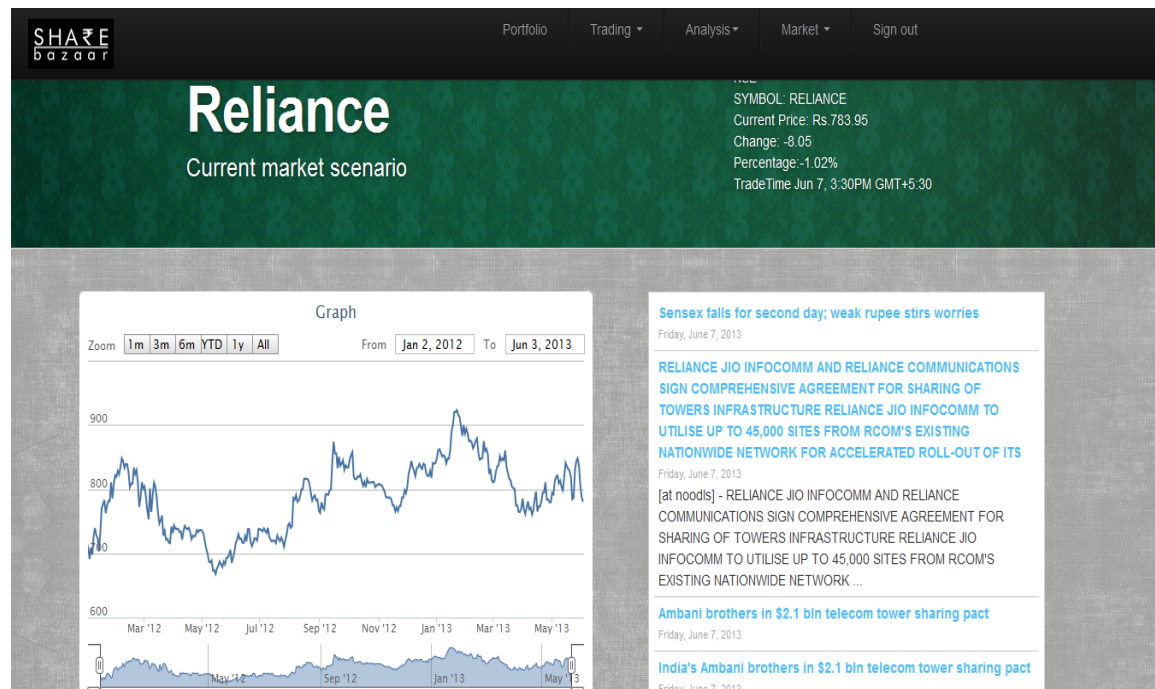
one output. The neural network is trained using 150 trading days and the network is tested for 107 days.

No of nodes in the 1 st Hidden layer	No of nodes in the 2 nd Hidden layer	Training set accuracy	Testing set accuracy
20	10	80.12%	79%
5	4	80.12%	71%
10	10	78%	74.756%
30	25	52%	58%

Table 6.1 Results

CHAPTER 7

SNAPSHOTS



SYMBOL: ACC
Current Price: Rs.1,207.55
Change: -0.75
Percentage:-0.06%
TradeTime Jun 7, 3:30PM GMT+5:30

Buy

Acc

Buy:

Number of stocks		Current Value		Total price
<input type="text" value="10"/>	×	<input type="text" value="1207.55"/>	=	<input type="text" value="12075.5"/>

SHARDE
BOARD

PortfolioTradingAnalysisMarketSign out

Trade

Buy or sell stocks.

Your buying power is
186347.16

Sell

View the company whose stocks you want to sell:

Asianpaint

Get Current Value

Sell:

Number of stocks		Current Value		Total price
12	×	4578	=	54936
<div>Sell!</div>				

Rankings

Where you stand amongst other users.

10 records per page Search:

User ID	Name	Amount	Email ID
21	Rohit	500000.0	sharma@gmail.com
22	Shikar	500000.0	dhawan@yahoo.com
23	Rahul	500000.0	dravid@aol.com
24	Lalit	500000.0	modi@gmail.com
25	Sidhartha	500000.0	sid@yahoo.com
26	David	500000.0	lloyd@microsoft.com
27	Nasser	500000.0	hussain@facebook.com

Stock prediction for today.

10 records per page Search:

Name	Predicted value	Rise/Fall
JindalSteel	364.2023516010594	▲
JUBILANT	1212.4777281687127	▲
LT	1503.359328544947	▲
MM	45.71691465953247	▼
PANTALOON	150.1155256642329	▼
RELIANCE	797.5904924989775	▼
RENUKA	22.392004317069887	▲
TataSteel	303.8677302192034	▲
WIPRO	335.78521214944635	▲

CHAPTER 8

CONCLUSION

This project has been an excellent learning experience for all of us, it helped us to work with various new software and it leads us to understand each phase of Software Development Lifecycle.

We were able to extract large amounts of data in the field of stock market through the Internet at runtime for the purpose of analysis and simulating a trade environment.

We also performed analysis using algorithms related to the field of data mining, machine learning and neural networks to predict the future trends.

The simulated trade system is also implemented by developing a portfolio management system to help train users in the stock market world and also for the purpose of understanding and testing the algorithms developed. Finally, this helps clients to get into the driver's seat by broadening their market knowledge and confidence so that they can make informed trading and investment decisions in the future.

We were also able to publish a paper named “INDIAN STOCK MARKET PREDICTOR SYSTEM” in The ASAR-International Conference on Electrical, Electronics and Computer Engineering (ASAR-ICEECE-2013), which was held at Mysore on May 12th, 2013

CHAPTER 9

FUTURE ENHANCEMENT

Using neural networks to forecast stock market prices will be a continuing area of research as researchers and investors strive to outperform the market, with the ultimate goal of bettering their returns. It is unlikely that new theoretical ideas will come out of this applied work. However, interesting results and validation of theories will occur as neural networks are applied to more complicated problems. For example, network pruning and training optimization are two very important research topics which impact the implementation of financial neural networks. Financial neural networks must be trained to learn the data and generalize, while being prevented from over training and memorizing the data. Also, due to their large number of inputs, network pruning is important to remove redundant input nodes and speed-up training and recall. The major research thrust in this area should be determining better network architectures. The commonly used backpropagation network offers good performance, but this performance could be improved by using recurrence or reusing past inputs and outputs. The architecture combining neural networks and expert systems shows potential. Currently, until we more fully understand the dynamics behind such chaotic systems, the best we can hope for is to model them as accurately as possible. Neural networks appear to be the best modeling method currently available as they capture nonlinearities in the system without human intervention. Continued work on improving neural network performance may lead to more insights in the chaotic nature of the systems they model. However, it is unlikely a neural network will ever be the perfect prediction device that is desired because the factors in a large dynamic system, like the stock market, are too complex to be understood for a long time.

BIBLIOGRAPHY

- <http://neuroph.sourceforge.net/>
- cran.r-project.org/
- www.r-bloggers.com/
- <http://andybromberg.com/sentiment-analysis/>
- jqueryui.com/
- <http://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html>
- <http://javabrainz.koushik.org/p/jsps-and-servlets.html>
- www.eclipse.org/