CCFEA
Centre for Computational Finance and Economic Agents

University of Essex

USING ARTIFICIAL NEURAL NETWORK TO PREDICT STOCK MARKET RETURNS

Submitted in partial fulfilment for the award of

MSc in Computational Finance

By

ARPIT AGRAWAL

PG Number: 0943428

Under the guidance of

Dr. John Foster

The Centre for Computational Finance and Economic Agents

University of Essex

Abstract

This research investigates the use of neural networks in finding a relation between two successive overshoots leading to directional changes in the market trend. The algorithm given by Glattfelder, Dupuis and Olsen which observes the directional changes, given a particular threshold, was studied and used to find the actual directional changes and overshoots in the historical data for fourteen major stocks form the Indian market. This data was then used to train a neural network using the FANN library and the overshoots in the market trend after every directional change were predicted. It was observed that although the neural network did not predict every overshoot correctly, there was an overall improvement in net returns from the trades done at directional change.

Acknowledgement

I show my gratitude and indebtedness towards my project supervisor Dr. John Foster whose professional guidance, support and ideas encouraged me to study the topic further and complete this research.

I would also like to thank my faculty at CCFEA, University of Essex and a special mention of Professor Edward Tsang, Dr Wing Lon Ng and Dr Richard Olsen who motivated me and inspired me largely for this research.

Thanks to my friend Arjun Koppa for his comments and inputs that gave a new dimention to this research.

Last but not the least I would like to thank my family for their continuing emotional and moral support and would like to dedicate this piece of work to them.

Contents

List of Figures

List of Tables

Chapter 1 – Introduction

1.1 Background

There have been a number of researches in the past regarding forecasting of stock market returns, all with a common motive of increasing the profitability of investing and trading[1]. People have come up with various numeric and machine learning techniques in order to successfully predict the market using various data models. There has also been a strong focus on the strategy of timing and pricing of the order for a trade in the market. Glattfelder, Dupuis and Olsen proposed an algorithm that studies the price curve of the market in real time and tries to capture the directional changes in the market trend for a given threshold λ and trade accordingly. They sold on observing a downtrend and bought on observing an uptrend and made a profit depending upon the overshoot after every directional change[2]. In this research we try to extend the idea given by Glattfelder, Dupuis and Olsen by investigating the scope of Artificial Neural Network in establishing a relationship between two successive overshoots in the market and proposing a system in which the overshoot before the next directional change occurs can be predicted using the previous overshoot.

1.2 Aim

To investigate the use of Artificial Neural Networks in finding a relation between two successive overshoot after directional changes occur in the price for various stocks from the Indian market.

1.3 Objectives

- The objectives of this research include the study of historical data for important stocks selected from different sectors from the Indian National Stock Exchange. This includes calculating the total potential profit given perfect foresight of the market, given a particular threshold for the directional change and taking into account all the transaction costs. This will be used as a benchmark to compare our outputs and measure the profitability of the system.
- Study of various numerical and machine learning techniques used by researchers in the past to forecast stock market returns. Our main focus will be on study of the algorithm proposed by Glattfelder, Dupuis and Olsen for observing directional

[1] (Majumdar & Hussain, 2010)
[2] (Glattfelder, Dupuis, & Olsen, 2010)

changes in the market returns and the study of various of artificial neural networks used by researchers over the years to predict market variables.

- Using FANN library to try to predict the next overshoot in the market when a directional change occurs. The previous overshoot will be used as the input to the neural network.
- To calculate the prices at which the predicted directional changes take place using the overshoots predicted by the FANN neural network and to compare them with the actual directional changes that took place in the market.

1.4 Artificial Neural Network

Many researchers claim that the stock market is a dynamic, non-linear, deterministic system which is complicated and only appears random because of its irregular fluctuations; in other words, stock market is a chaos system[3]. It is not very easy to deal with chaos systems using the normal analytical methods. An effective way of learning these systems, as explained by Majumdar and Hussain (2010) is by using artificial neural networks because they make very few assumptions about the functional form of the underlying dynamic dependencies and their initial conditions[3]. *'A neural network is a massively parallel distributed processor made up of simple processing unit, known as neurons, which has a natural propensity for storing experimental knowledge and making it available for use.'*[4] The neural network structure consists of neurons in two or more layers with weighted connections between neurons which are often non linear scalar transformations. Each neuron has a weight assigned to it. A feed-Forward Backpropagation is a popular weight adaption algorithm that is used to minimise a nonlinear function and train the neural network[5]. In this research we aim to use the feed forward Backpropagation artificial neural network to try to predict the overshoots in the market before the next directional change occurs.

[3] (Majumdar & Hussain, 2010)
[4] (Majumdar & Hussain, 2010)
[5] (Wilde, 1997)

Chapter 2 – Literature Review

2.1 Market Prediction Models

Although many traders feel threatened by the idea that algorithmic models can achieve better results, technology is still penetrating trading floors across the world. Resistive reactions to technical advancements are common human behavior and are often observed in every industry, for example autopilots were initially opposed when they were introduced in early 20th century but actually they proved to be more efficient and precise than human pilots. Similarly people around the world have proposed various models and systems over the past two decades which are expected to forecast stock market returns and trade accordingly with efficiency and precision[6].

Many researchers and practitioners have proposed various prediction models using Fundamental and Technical analytical techniques. Not all of them have proven to predict precisely but there has always been some contribution to research in this field over time.

"Fundamental analysis involves the in-depth study of the changes in stock price in terms of exogenous macroeconomic variables.[7]" It is purely an information based model and assumes the stock price to be a function of intrinsic value of the share and the expected returns of the investor. Since the interpretation of the available information is highly subjective and totally lays on the intellectuality of the analyst and availability of the information, it is not easy to derive an objective system that could forecast the stock prices or returns on the basis of fundamental analysis[8].

Because system based on fundamental information would not be too much of a help in designing a predictor, a rule based system can be thought of as a good idea. Like fundamental analysis, technical Analysis also studies the information presents in the market. The same news can affect different people in different ways. The sum of the attitude of all the participants and potential participants is reflected as the price. As Martin J.Pring quotes Garfield Drew in his book, "They never sell for what they are worth but for what people think they are worth." In Technical Analysis we study various market indicators that provide

[6] (J.Deboeck, 1994)
[7] (Majumdar & Hussain, 2010)
[8] (Majumdar & Hussain, 2010)

evidence to the information present in the market and how people react to it and try to predict the trend change or directional changes in the market before they actually occur[9].

Time series prediction models are designed by using various technical indicators and the prediction variables by studying the past data and capturing trends. There are two main approaches of time series modeling and forecasting as explained by Majumdar & Hussain, linear and non-linear. Linear models like moving average, exponential smoothing, regression, Autoregressive Moving Average (ARMA) and Autoregressive Integrated Moving Average (ARIMA) are popular and commonly used by traders and analysts around the world for forecasting market returns and stock prices. But market returns have not really been observed to be linear and as a consequence these systems must be frequently fine-tuned according to changing market parameters. For this reason some parametric non-linear models such as Autoregressive conditional heteroskedasticity (ARCH) and General Autoregressive conditional heteroskedasticity (GARCH) have been used for financial forecasting. But most of these non-linear techniques require the models to be specified beforehand which means the relationship between the input parameters and the output variable to be predicted should be specified explicitly before any estimation of the prediction variable is done[10].

2.2 Using an event based approach

Prices in any financial market evolve as events occur. Events can be market transactions, trades or any news or political announcements. We know that neither the news nor the price change occur at regular intervals in the market. Therefore an event based approach was proposed by Bisig, Dupuis, Impagliazzo and Olsen in their paper that considers the sequence of price directional changes of a fixed magnitude (say λ) as an event in the market. Every directional change is often followed by an overshoot (say ω). The overshoot ω can be thought of as a measure of market activity. Every directional change triggers a new overshoot and the overshoot continues until the price again changes by $-\lambda$ (changes by λ in a direction opposite to the current market trend) triggering the next directional change event. Therefore the value of ω can be thought of swinging between $-\lambda$ and any positive value depending upon the market conditions[11]. This can be easily understood by the figure given below (figure1).

[9] (Pring, 2002)
[10] (Majumdar & Hussain, 2010)
[11] (Bisig, Dupuis, Impagliazzo, & Olsen, 2009)

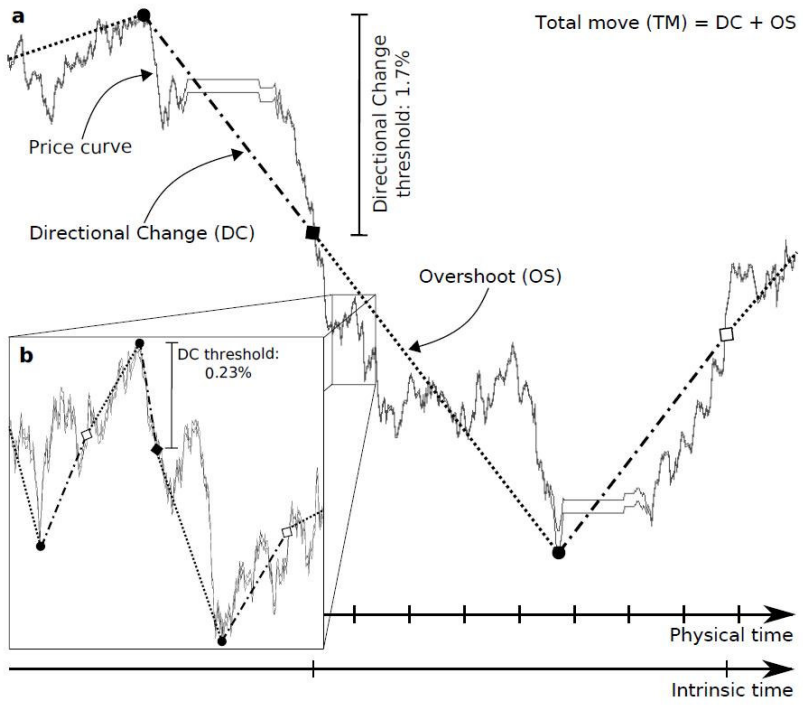

Figure 1: Projection of a (a) two-week, (b) zoomed-in 36 hour price sample onto a reduced set of so-called directional change events defined by a threshold (a) λ= 1.7%, (b) λ= 0.23%. These directional change events (diamonds) act as natural dissection points, decomposing a total price move between two extremal price levels (bullets) into so-called directional change (solid lines) and overshoot (dashed lines) sections. Time scales depict physical time ticking evenly across different price curve activity regimes, whereas intrinsic time triggers only at directional change events. Note: This figure and its description are taken directly from [12]

A directional change depicts news in the market and the overshoot ω measures the amount of activity in the market. For this reason, ω remain of great interest to my research. Predicting the overshoot before the next directional change occurs in the market can be thought of as an opportunity to make money.

Glattfelder, Dupuis and Olsen studies a set of five year (from December 1, 2002 to December 1, 2007) tick data for fourteen different currency pairs from the foreign exchange market and measured the market coastline by taking different thresholds λ[13]. We will be using the same directional change algorithm used by them to find the market coastline and determine the potential profit that a trader can make in the market given a perfect foresight and a particular

[12] (Bisig, Dupuis, Impagliazzo, & Olsen, 2009)
[13] (Glattfelder, Dupuis, & Olsen, 2010)

threshold λ for the directional change. This will be used as a benchmark to compare our result.

2.3 Artificial Neural Network

Apart from market prediction models described above i.e. fundamental and technical analysis, researchers and traders have also used various non-linear non-parametric data-driven systems which are actually model free and this gives them an upper hand above the rest. One such widely used data processing technique is the Artificial Neural Network.

Neural Networks, as explained by Azoff(1994), are non-linear models that can be trained to map a function between the input parameters and output variables thereby extracting hidden structure and relationships governing data. Over time people have used neural networks to predict future prices in a time series using the past values and other derived variables. The Neural Network research has gained huge importance in the past few years due to its powerful pattern recognition capabilities and also because of its excellent computation speed and accuracy. The network is trained using input parameters for any possible relationships with the corresponding outputs, without specifying and model constraints, driven and shaped completely by input data. Neural Networks are math, not magic and they have strong ties to statistics[14]. Based on their application, neural network can be divided into four types: Classification, Association, Codification and Simulation[15]. The simulation type is the most important to us here because we will be trying to predict the next overshoot (ω) in the directional change algorithm explained above.

People have used Neural Networks over the years to predict various parameters and variables related to Stock Market Returns and price movements. There are various neural network architecture designs or neural paradigms and they can be broadly classified into three categories on the basis of technique used to iteratively train the weights in the network namely supervised learning, reinforcement learning and unsupervised learning. In supervised learning the network output targets are known during training and actual output may be fed back to the network for improving performance[16]. Backpropagation is a supervised learning method that tunes and retunes the weights of the neural network by finding the error while training by comparing the actual and the predicted value of the output variable[17].

[14] (Klimasauskas, 1994)
[15] (Azoff, 1994)
[16] (Azoff, 1994)
[17] (Majumdar & Hussain, 2010)

Backpropagation algorithm has been used in our research to train the neural network with the values of successive overshoots in the directional changes algorithm proposed by Bisig, Dupuis, Impagliazzo, and Olsen.

Yoda, a researcher at Nikko Securities Company in Tokyo used neural network to predict the Tokyo stock exchange price index (TOPIX). He studied various factors affecting the Tokyo Stock Market like the weekly Dow Jones Index, Moving average of interest rates of Japanese Government Bonds (JGB), vector curves for JGB and other technical indicators like IT Radar (developed by Nikko). The system developed by Yoda consisted of several neural network modules that used historical data to learn the relationship between various technical and economic indexes and the timing for when to buy and sell on the TOPIX. An arithmetic average of the outputs of all the modules was taken as the final output. The simulation and the actual experiment performed remarkably while predicting timing of buying and selling stocks. The system produced 145 predictions from September 1989 to October 1992 with a correct prediction rate of 62.1% for both rising and falling markets[18].

In other research, Gia-Shuh Jang and Feipei Lai from National Taiwan University presented a dual adaptive neural network that can predict the short-term trends of price movements as well as recognize major trend reversals in the market and used it to develop an innovative, intelligent and profitable stock trading system for the Taiwan stock market. Learning processes in a neural network sometimes get trapped in a local minimum point of error surface and the network cannot really produce the desired accuracy. To prevent this, it is important that the structure of the network should be readjusted and updated as and when the characteristics of the market change. Therefore by using learning algorithm with structure level adaption ability, Gia-Shuh Jang and Feipei Lai ensured their model's accuracy by taking into account its tolerance to ephemeral events like tax and trade policies and singular international events etc. They used the latest 'm' training patterns at any time to tune the weights in the neural network; this was called the moving window training system. The structure of their Dual-adaptive system was self synthesized based on training patterns collected from the windows of different sizes and the system continuously adapted its structure to follow any statistical changes in the problem domain. The annual rates of return achieved by this system during the test period in 1990 and 1991 were much superior to the

[18] (Yoda, 1994)

performance buy and hold strategy and of closed-end funds that were focused on Taiwan and were managed by human experts[19].

In another more recent paper by Majumdar and Hussain published at the National Stock Exchange of India, they use artificial neural network for predicting the daily closing value of the S&P CNX Nifty-50 Index which is the main index of the National Stock Exchange in India using historical data from 1st January 2000 to 31st December 2009. They proposed a three layer feed-forward back propagation neural network with 10 input neurons and a hidden layer with 5 neurons and 1 output neuron with tan-sigmoidal and linear transfer function in the hidden and the output layer respectively. Accuracy of their neural network was compared using various out of sample performance measures. The highest performance of the network in terms of accuracy in predicting the direction of closing value of the index was reported at 89.65% with an average accuracy of 69.72% over a period of four years[20].

Mohan, Jha, Laha, & Dutta (2006) studied efficacy of artificial neural network in modelling the the weekly closing value of Sensex the main index at the Bombay Stock Exchange, a leading Stock Exchange in India. Sensex data of 250 weeks starting from January 1997 was taken and was fed to two networks ANN1 and ANN2. ANN1 takes as its inputs the weekly closing value, 52-week moving average of the weekly closing SENSEX values, 5-week moving average of the same, and the 10-week Oscillator for the past 200 weeks. ANN2 takes as its inputs the weekly closing value, 52-week moving average of the weekly closing SENSEX values, 5-week moving average of the same and the 5-week volatility for the past 200 weeks. Weekly closing values of sensex for a period of two years beginning January 2002 were predcted successfully with root mean square error (RMSE) of 4.82 per cent and mean absolute error (MAE) of 3.93 per cent achieved by ANN1 while RMSE of 6.87 per cent and MAE of 5.52 per cent achieved by ANN2[21].

Panda and Narasimhan(2006) in another article used artificial neural network for the forecasting of the daily Bombay Stock Exchange Sensex returns. Their data consisted of total of 2553 observations of daily closing values of Sensex starting from January 1994 to December 2004 out of which 2100 observations were used for training and the rest for testing their network. They compared the performance using six different performance measures and found that the neural network out-performed linear autoregressive and random walk by all

[19] (Jang & Lai, 1994)
[20] (Majumdar & Hussain, 2010)
[21] (Mohan, Jha, Laha, & Dutta, 2006)

performance measures in both in-sample and out-of-sample forecasting of daily BSE Sensex returns and they showed that the neural network is capable of capturing non-linearity contained in stock returns. Based on their findings they also suggested that the stock markets do not follow a random walk and there exists a possibility of predicting stock returns[22].

2.4 FANN Library

Fast Artificial Neural Network (FANN) Library is a free open source neural network library, which implements multilayer artificial neural networks in C with support for both fully connected and sparsely connected networks. This library will be used in this research to investigate whether neural networks can successfully predict the overshoot before the next directional change occur. The portable graphical user environment, FannExplorer and the multithreaded FannKernal from the Fann 2.1 library can be used for developing, training and testing neural networks. It uses two main input files which contain the training and the testing data. The extensions for these files should be .train (example: *abc.train*) and .test (example: *abc.test*) respectively. Once the FannKernal is fired up, one can easily create new networks and train and test them using the data supplied as input files. A detailed reference manual and a tutorial are provided on the official website of the FANN library that helps in development, training and testing up of the network[23].

[22] (Panda & Narasimhan, 2006)
[23] (Fast Artificial Neural Network , 2010)

Chapter 3 – Data and Methodology

3.1 Data

In order to help researchers and market participants to undertake research work the National Stock Exchange of India regularly releases high frequency tick-by-tick historical data that can be bought from their website http://www.nseindia.com in the form of DVDs. Fourteen stocks from various sectors of the industry were chosen and their high frequency trade information from 1st of December 2008 to 6th of January 2009 was studied under this research. The names, symbols and brief information about the market sector they belong to are given below in Table 1.

Sno.	Symbol	Name of the Company	Industry
1	AXISBANK	Axis Bank	Banking, Financial Services Firm
2	HDFC	HDFC Bank	Banking, Financial Services Firm
3	RELCAPITAL	Reliance Capital Limited	Non Banking Financial Company
4	DLF	Delhi Land and Finance Ltd.	India's largest real estate developers
5	UNITECH	Unitech Group	Real estate investment company
6	INFOSYS	Infosys	IT services and consulting
7	WIPRO	Wipro Limited	IT services and consulting
8	SATYAMCOMP	Satyam Computers (Now Mahindra Satyam)	IT services and consulting
9	ESSAROIL	Essar Oil Ltd.	Petroleum and Gas company
10	RELIANCE	Reliance Industries Limited	India's largest private sector conglomerate company
11	TATASTEEL	Tata Steel	India's largest private sector steel company
12	BHARTIARTL	Bharti Airtel Ltd.	Telecommunication services
13	HEROHONDA	Hero Honda Motors Ltd.	World's biggest 2-wheeler motorcycle manufacturer.
14	PANTALOONR	Pantaloon Retail (India) Limited	India's largest retail brand

Table 1: List of stocks chosen for study in this research[24].

The data was purchased from the National Stock Exchange of India and it contained trade information about every security traded throughout the day. The format of the data as obtained from NSE is given below[25]:

1. Trade ID number- A unique number for each trade, the files on the CD/DVD are sorted by this trade ID.
2. Symbol - The symbol of the security traded.
3. Series - The series of the security traded.
4. Timestamp - The time at which the trade took place, formatted as hh:mm:ss.
5. Price - The price at this trade.
6. Quantity traded - The number of shares transacted in this trade.

Data was present in .trd formatted data files with one file containing all the trades for one day. The files provided by NSE were named in the format 'yyyymmdd.trd' for example '20081201.trd' for the file containing trade information for 1st December 2008. Files corresponding to dates from 1st December 2008 to 5th January 2009 were imported to MsAccess database. SQL queries were then used to extract the price curves for all the fourteen stocks. A sample query for the same is shown below:

*select DISTINCT (left(timeStamp,2)*3600 + right(left(timeStamp,5),2)*60 + right(timeStamp,2)) as time_, price into 20081201AXISBANK from 20081201 where symbol = 'AXISBANK';*

We can see in the query above the time has been converted from a string format to numeric format with the new field "time_" containing the number of seconds since the beginning of that day. Also the redundant entries or the entries where the duplicate trades were entered at the same instant were also removed using the DISTINCT clause in the query. The data from 1 day (1/12/2008 in this example) was saved in separate tables for each of the fourteen stocks. (In this example we can see a new table 20081201AXISBANK stores the trade information for 1/12/2008 for all the trades where symbol='AXISBANK')

[24] (National Stock Exchange of India, 2010)
[25] (National Stock Exchange of India, 2010)

3.2 Measuring the Coastline

To find the potential profit that a trader with perfect foresight can make given a threshold λ, and to calculate the profit made by the algorithm given by Glattfelder, Dupuis, & Olsen, (2010) we use the algorithm given by the following pseudo code. The pseudo code is inspired by the algorithm given originally by Glattfelder, Dupuis and Olsen in their paper[26].

- *Declare two lists* ***list_of_DC***, ***list_of_olsen_DC***
- set ***Trend*** = *'up'*
- *set* ***Position*** = *0*
- *set* ***benchmark_price*** = *0*
- ***For each*** *trade(price=P) that occurred in the market:*
    - *If* ***Trend*** = *'up' then*
        - *If P >* ***benchmark_price*** *then*
            - *Update* ***benchmark_price***
        - *Else If price falls by* λ then
            - *DIRECTIONAL CHANGE!!!!*
            - *set* ***trend*** = *'down'*
            - ***list_of_DC*** ← ***benchmark_price***
            - ***list_of_olsen_DC*** ← *P*
            - ***Position=Position****+1*
            - *update* ***benchmark_price***
    - *If* ***Trend*** = *'down' then*
        - *If P <* ***benchmark_price*** *then*
            - *Update* ***benchmark_price***
        - *Else If price rises by* λ then
            - *DIRECTIONAL CHANGE!!!!*
            - *set* ***Trend*** = *'up'*
            - ***list_of_DC*** ← -(***benchmark_price***)[negative because buy]
            - ***list_of_olsen_DC*** ← ***-****(P)* [negative because buy]
            - ***Position=Position****-1*
            - *update* ***benchmark_price***

[26] (Glattfelder, Dupuis, & Olsen, 2010)

The code for the above algorithm is given in Appendix A. At this point there are two lists with us. ***list_of_DC*** and ***list_of_Olsen_DC*** and as the names suggest they contain the benchmark prices at which a potential trader with perfect foresight and a trader using Olsen's algorithm respectively will discover the directional changes. The variable **Position** contains 0 if the trader is does not hold any stock or debt in other words position of the trader is closed. If the **Position** = +1 then the trader is one share long and if the **Position** = -1 then the trader is one share short. We assume that the trader will close its position at the end of the day. If we sum up all the values in the ***list_of_DC***, it'll give us the profit made by a trader with perfect foresight or coastline of the market in other words.

Potential Profit = sum (*list_of_DC)* …….... (eq.1)

Similarly the profit calculated by the algorithm given by Glattfelder, Dupuis, & Olsen can be given by the equation given below:

Olsen's Profit = sum (*list_of_Olsen_DC)* …….... (eq.2)

But note that the above profit does not take into account any transaction costs, brokerage charges or taxes etc. The figures for all the costs that any trader would have to incur in order to trade on the National Stock Exchange of India are given in the Table 2. These values have been taken from a market participant and a leading stock trading company "Interactive Brokers"[27].

[27] (Interactive Brokers Group, Inc.'s, 2010)

Cost Factor	Value (%)
Commission	0.05[*^]
Exchange Transaction Charges	0.00325
Service Tax	0.00548[†]
SEBI Turnover charges	0.0001
Stamp Duty	0.002[‡¥]
Securities Transaction Tax	0.0125[¥]
Total	0.07333

Table 2: Costs incurred when trading on National Stock Exchange of India through Interactive Brokers[26].

* Minimum = Rs.50 (Minimum = 2.5% If Rs.50 exceeds the 2.5% of the total trade value which is the maximum limit set by the National Stock Exchange.)

^ In the case of intra-day trades on the NSE, in which a position is opened and closed on the same day, commissions will only be charged on the opening leg.

† Service tax is charged on commission amount and exchange transaction charges at prevailing rate (10.30% currently) approximated to be equal to 0.00548%

‡ Stamp duty given above is for the state of Maharashtra and it varies from state to state.

¥ These values are for cash intra-day buy and sell trades and do not apply to cash delivery trades.

Note that the total cost, calculated to be equal to 0.07333% is only an approximation and may occasionally vary based on the actual value of the trade. Also the above value holds for the opening leg in an intra-day trade and the value for the corresponding closing leg will be equal to 0.01493%. Because we assume that the trader will close its position at the end of the day, therefore we can assume that half of the trades during the day will be the opening leg (transaction costs = 0.07333%) and the rest will be the closing legs (transaction costs = 0.01493%) of intra-day trades. Therefore we can assume that the trader pays an average of **0.04413%** as the total transaction costs per trade.

From the above calculation and the algorithm given before we can say that the total profit after deducting costs is equal to total profit as calculated in the equation (1) minus 0.04413%. This can be written as the following equation:

(Potential Profit) $_{\textbf{minus costs}}$ **= sum (list_of_DC) - 0.0004413* sum (abs(list_of_DC))**

…….... (eq.3)

Similarly the profit calculated by the algorithm given by Glattfelder, Dupuis, & Olsen can be given by the equation given below:

(Olsen's Profit) $_{\textbf{minus costs}}$ **=**

sum (list_of_Olsen_DC) -0.0004413* sum (abs(list_of_Olsen_DC))... (eq.4)

The overshoot ω as explained before is calculated for each value in the list_of_DC. This gives us the actual overshoot before the next directional change occurs in the market. Overshoots were calculated using simple arithmetic as follows:

$$1+\omega = \frac{\text{abs}(\text{abs}(\textbf{list\_of\_DC}(i)) - \text{abs}(\textbf{list\_of\_DC}(i-1)))}{\lambda * \text{abs}(\textbf{list\_of\_DC}\ (i - 1))} \quad \text{…… (eq.5)}$$

What you see above is nothing but the return of two consecutive trades divided by λ. This gives us the overshoot (ω) in terms of number of λ(s).

3.3 Prediction using FANN

The executables for FannExplorer and FannKernal were downloaded and extracted from their official website http://leenissen.dk/fann/ (August, 1st 2010). The minimum requirements of FannExplorer namely, a web browser with JavaScript, Flash version 7 or later and FannKernal were installed to ensure its success. It is now important to create files in a format that the Fann Library can understand for the testing and training data as explained in the reference manual provided on the Fann website[28].

We wanted our neural network to investigate the presence of any relationship between the successive overshoots. For each of the fourteen stocks that have been studied in this research, we calculate the overshoots between successive directional changes using the eq.5 for the data taken from 1st of December 2008 to 2nd of January 2009. This is a sample of 23 days of high frequency tick-by-tick trading data as obtained by NSE and stored in the tables as explained in the previous sections. Threshold λ for each stock was carefully selected so that there were just enough trades during one trading day. Table 3 shows the list of all the fourteen stocks with the threshold used for each to calculate directional changes.

Symbol	Threshold λ (%)	Training File	Testing Files
AXISBANK	1	AXISBANK.train	AXISBANK.test
HDFC	1	HDFC.train	HDFC.test
RELCAPITAL	1	RELCAPITAL.train	RELCAPITAL.test
DLF	1.25	DLF.train	DLF.test
UNITECH	2	UNITECH.train	UNITECH.test
INFOSYS	1	INFOSYS.train	INFOSYS.test
WIPRO	1	WIPRO.train	WIPRO.test
SATYAMCOMP	2	SATYAMCOMP.train	SATYAMCOMP.test
ESSAROIL	1	ESSAROIL.train	ESSAROIL.test
RELIANCE	0.7	RELIANCE.train	RELIANCE.test
TATASTEEL	1.25	TATASTEEL.train	TATASTEEL.test
BHARTIARTL	1	BHARTIARTL.train	BHARTIARTL.test
HEROHONDA	0.6	HEROHONDA.train	HEROHONDA.test
PANTALOONR	1	PANTALOONR.train	PANTALOONR.test

[28] (Fast Artificial Neural Network , 2010)

Table 3: Thresholds used to calculate directional change for each stock and the names of files containing training and testing data

After calculating the directional changes as defined by thresholds given in the Table 3, we now normalize them between 0 and 1 and then save them as a list of pairs of successive overshoots. Normalization is important to make sure that the network is consistent for all the stocks and understands the inputs well. Simple arithmetic division by 6 was used as a function to normalize the overshoots after observation of all the fourteen stocks because the overshoots always lay between 0 and 6. For example say there were four overshoots given by 1.811, 1.821, 2.039 and 1.666 then after normalization they will be 0.302, 0.304, 0.339 and 0.278. And to save them as training data we will be using a list of pairs of consecutive overshoots as given below:

```
3 1 1
0.302
0.304
0.304
0.339
0.339
0.278
```

The numbers 3,1,1 in the first row represent the number of samples in the data, the number of inputs and the number of outputs respectively. Each pair of rows following the first row represents the input-output pairs of successive overshoots. Files like the above were created for each of the fourteen stocks containing training data for 23 days and stored as the filenames given in Table 3.

The overshoots for the 5th of January 2009 were taken as the testing data. The testing data was formatted in the similar way as the training data and saved in fourteen different files, one for each stock, with filenames given in Table 3.

FannExplorer and FannKernal were downloaded and extracted from the Fann website[29]. The fourteen training files and the fourteen testing files were saved in the "net" folder present in the Fann installation directory. The FannKernal was executed (screenshot given as Figure 2) and it gives us a url on the localhost where we can now access the FannExplorer.

[29] (Fast Artificial Neural Network , 2010)

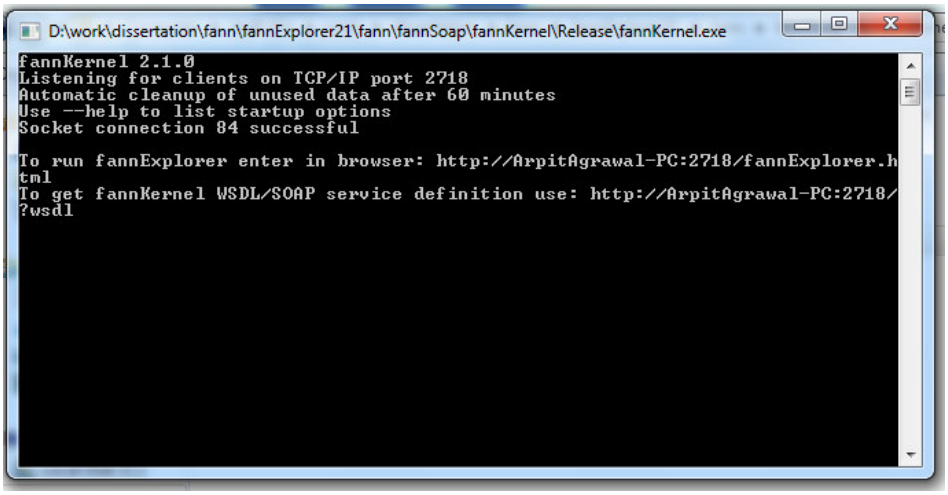

Figure 2: Screenshot of FannKernal.exe; we can see in the figure that the URL for accessing the FannExplorer was given by FannKernal as *http://arpitagrawal-pc:2718/FannExplorer.html*

FannExplorer was opened using the URL provided by the FannKernal and its screenshot is given in the Figure 3.

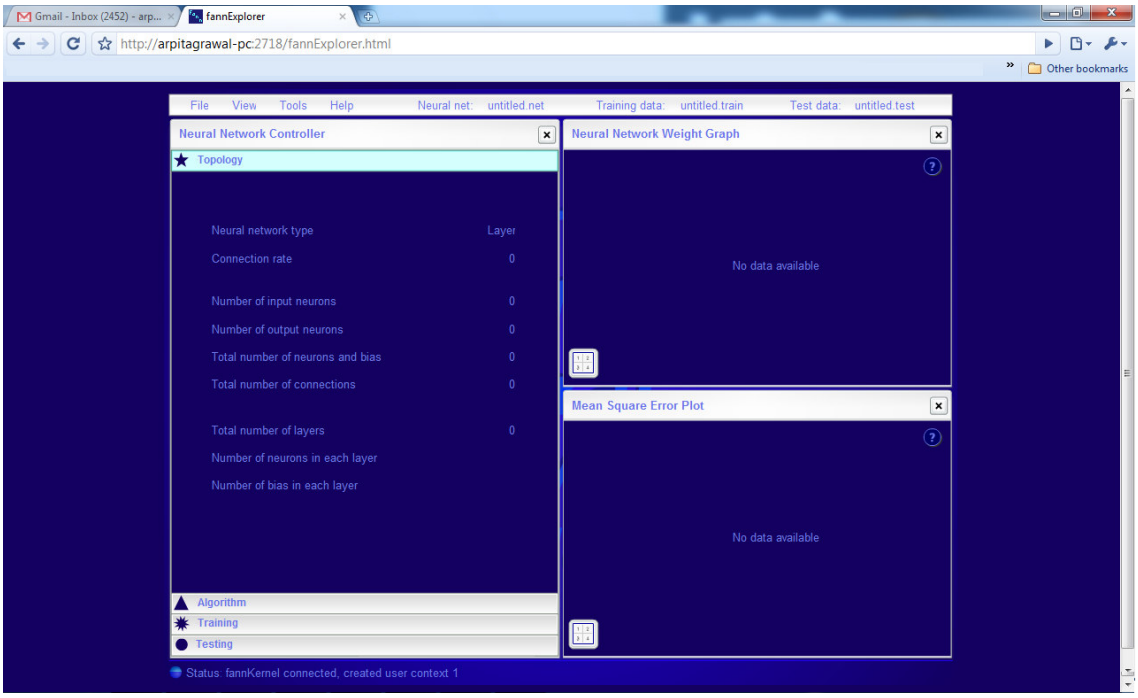

Figure 3: screenshot of the FannExplorer

Now, a new neural network was created. To do so, we select **file->new neural network** and enter the desired settings for the network in the dialog box that pops up (screenshot in Figure 4)

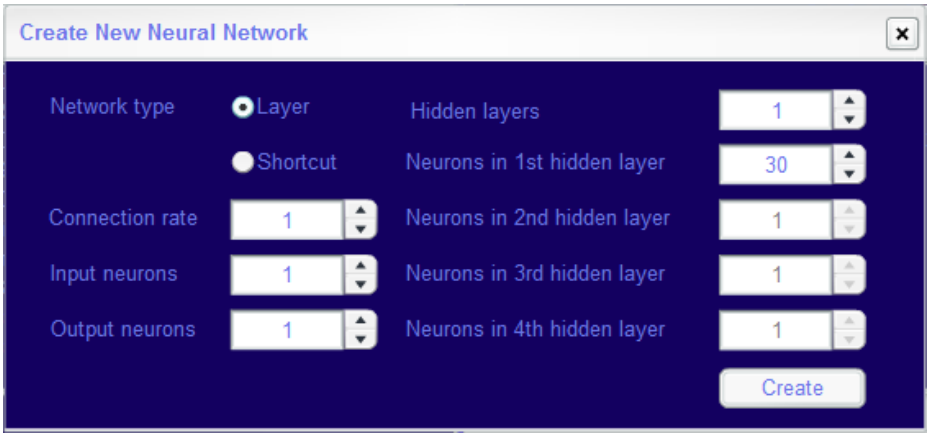

Figure 4: screenshot of "Create New Neural Network" dialog box. Here we can see all the settings for the neural network. Our neural network has 1 input neuron, 1 output neuron and a hidden layer was selected with 30 neurons.

Different combinations of number of hidden layers and number of neurons in the hidden layer(s) were tried for the neural network. Although there were very negligible variations in the result, the network with single hidden layer with 30 neurons was found to be most efficient in accordance to our data and was therefore selected for this research.

Once the network is created, the training and testing data files were loaded using **load training data** and **load testing data** options from the **file** menu. These menus open dialog boxes through which you can easily select the training and testing files, screenshots for the same given in Figure 5 and Figure 6.

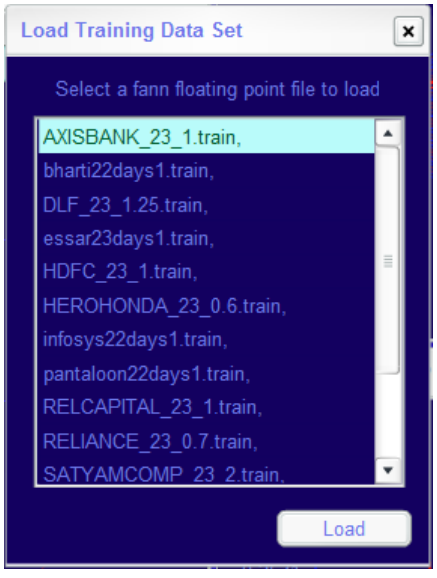

Figure 5: load training data dialog box

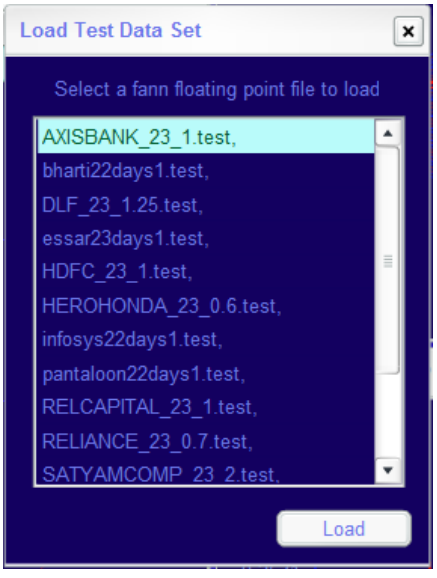

Figure 6: load test data screenshot

The network is now loaded with all the data. FANN initializes all the other important parameters of the network automatically to its default settings. To train the data we go to the training tab (as shown in figure7) and click on **Shuffle training data** and then click on **Initialize**, this shuffles the training data, initializes the network with the weights from the training data using Widrow & Nguyen's algorithm which attempts to set up the weights to speed up the training[30]. The number of training epochs was set to 200 by default and we found that this was enough because even when the number was raised to 1000, the system trained the network to the same level of mean square error. Minimum mean square error on training was set to 0.001 although the network didn't achieve it for any of the stocks.

Now that all the initial settings have been made, we click on the **Train** button. This trains the neural network using the training data provided and a plot of the mean square error can be seen on the bottom right (see figure7). Once the network has been trained, we now test it for the testing data sample. For this we go to the **Testing** tab and click on **Execute**. As soon as we click on **Execute**, a plot as shown in Figure 8 is obtained. The plot shows two lines, red(top) and green(top). The green line shows the desired overshoots, these are the actual overshoots from the data taken from the market. The red line actually shows us the output of the neural network. These are the overshoots as predicted by the network. The predicted values of the overshoots can be obtained from the **Tools->Status** menu. This opens a dialog box with the current status of the neural network system which currently shows the outputs from the network (as seen in Figure 9)

[30] (Fast Artificial Neural Network , 2010)

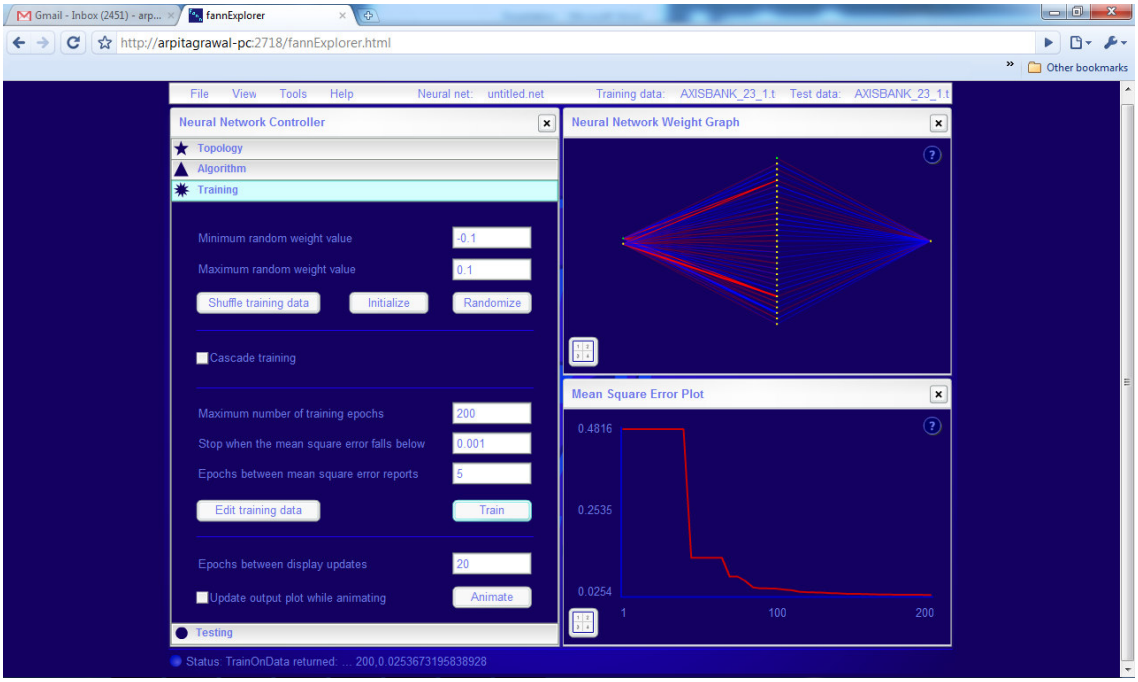

Figure 7: Screenshot of the FannExplorer showing the training of the network with data for '*AXISBANK*'. The mean square error plot can be seen on the bottom right of the screen.

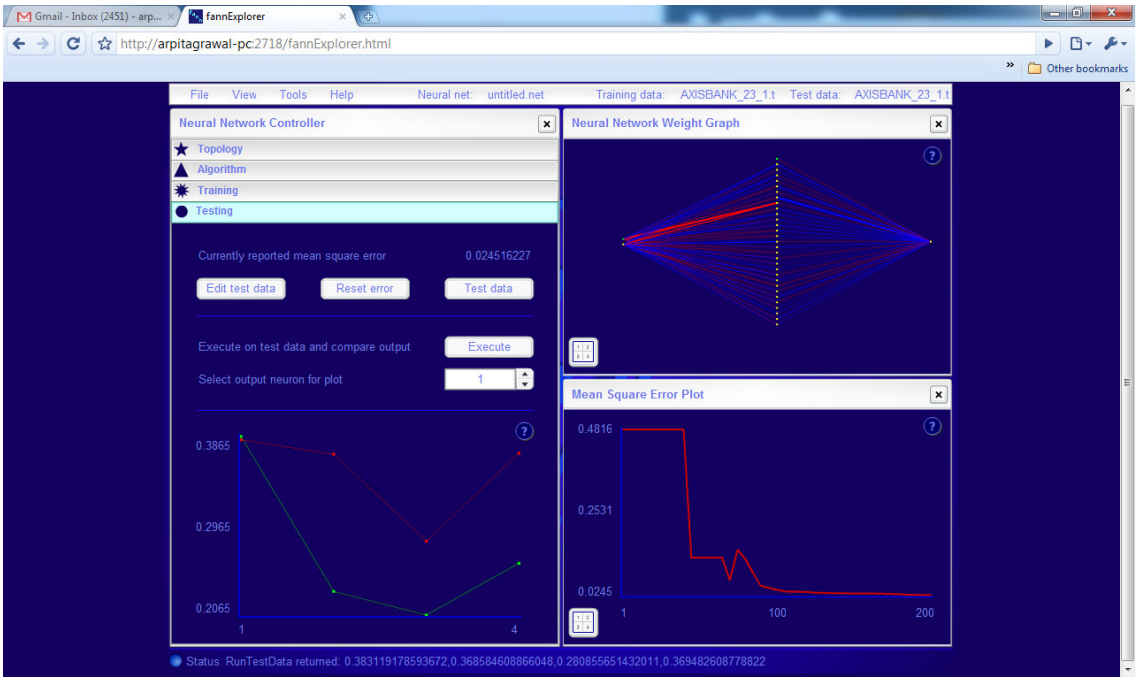

Figure 8: Screenshot of the FannExplorer showing the testing of the network with data for '*AXISBANK'*. The output plot can be seen in the bottom left showing the desired and the obtained values of the overshoots.

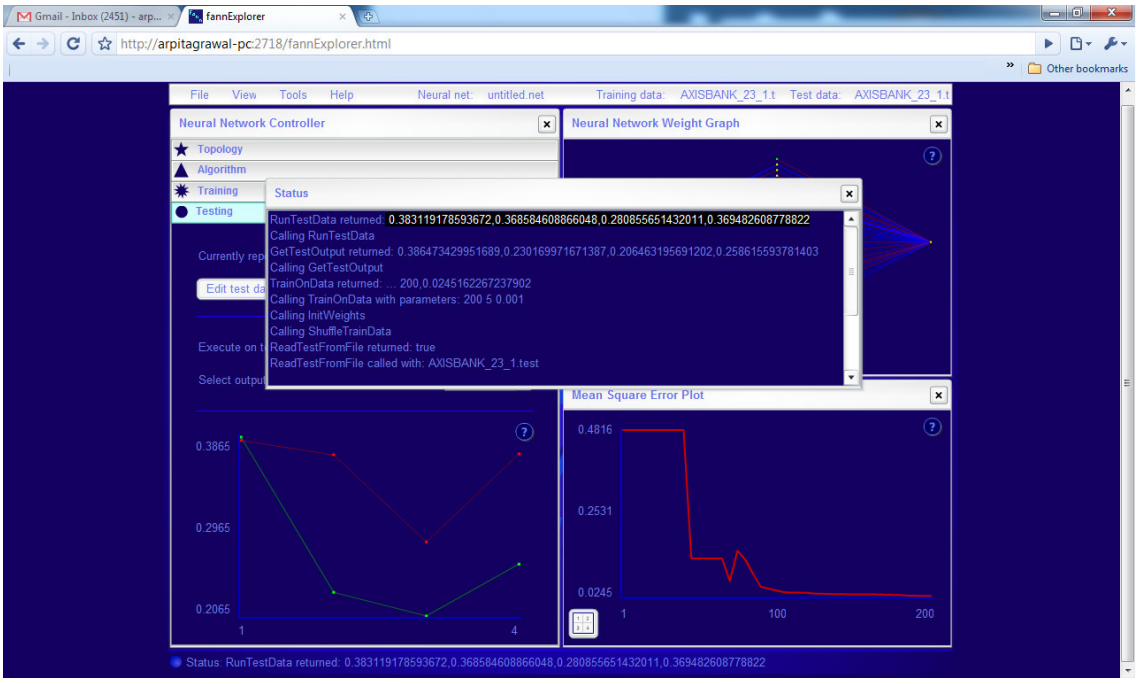

Figure 9: Screenshot of the FannExplorer showing the testing of the network with data for '*AXISBANK'*. The output values from the neural network can be copied from the status box.

The network is trained and tested for all the fourteen stocks separately and the results are obtained. Once we have the result we now find out the trading price using the predicted

overshoot. We can then compare those prices to the original algorithm to see how efficient the output of the neural network is. The overshoots and the predicted prices from the neural network are given in the result section.

Chapter 4 – Results

The directional changes were calculated using the overshoots predicted by the neural network and were plotted against the actual directional changes that took place in the market. The actual directional changes and the predicted values were also compared by those observed by the algorithm given by Glattfelder, Dupuis and Olsen (2010). Below you can find graphs and tables for each of the fourteen stocks showing the plot and the values for the actual directional changes, the directional changes as observed Olsen's algorithm and the directional changes as predicted by our artificial neural network. The last row in the table shows the net profit by each of the three strategies after deducting the transaction costs. You may find that the number of predictions are not as many as the total number of directional changes that took place. This is because at the points where the predictions are missing, the directional change took place before the predicted price occurred in the market.

Prediction for ESSAROIL

Potential	Olsen	Prediction
94.35	93.3	
93.1	94.05	
94.25	93.25	
93.2	94.15	
94.3	93.3	
91.55	92.5	92.4422
92.9	91.5	
91.15	92.1	
92.5	91.5	
91.4	92.35	
93.5	92.5	93.20037
93.05	93.05	
7.858	**-3.341**	**0.676**

Table 4:DC for ESSAROIL

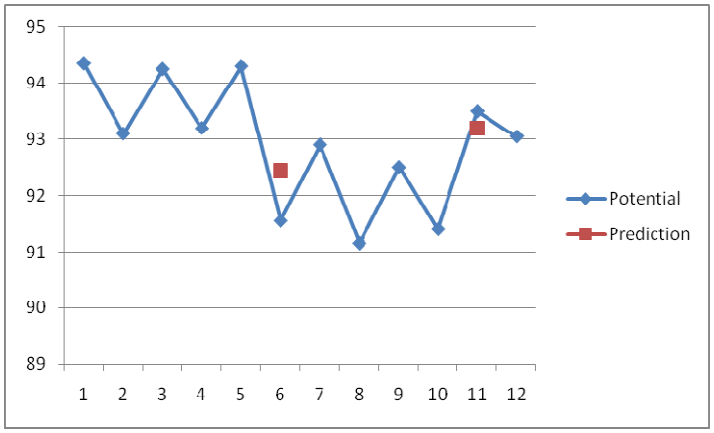

Figure 10: Plot of the Directional Changes for ESSAROIL

Predictions for AXISBANK

Potential	Olsen	Prediction
558.8	552.95	
552	557.6	
564.8	559.1	561.3014
557	562.6	
563.9	558	
555.15	560.75	555.15
21.971	**-12.379**	**5.904**

Table 5: DC for AXISBANK

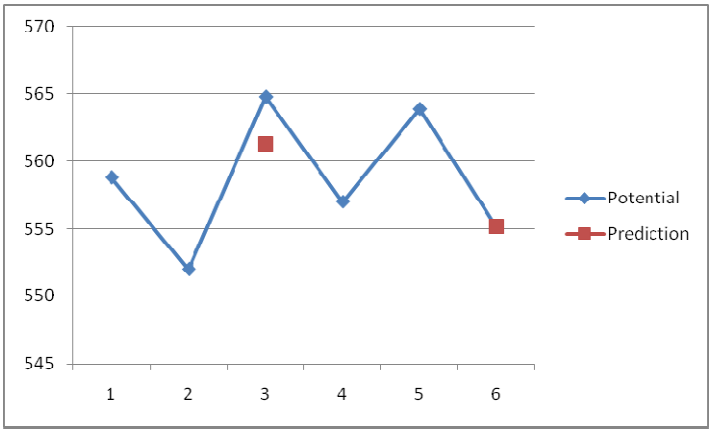

Figure 11: Plot of the Directional Changes for AXISBANK

Predictions for PANTALOONR

Potential	Olsen	Prediction
228.95	225.05	
225.05	228.95	
229.9	227.05	229.6678
227.05	229.75	
229.75	227.35	
226.1	228.7	
237.5	235.1	
235	237.5	
237.5	235.05	
235.05	237.5	
237.5	235	
234.5	236.95	234.5
17.121	**-15.978**	**-4.933**

Table 6: DC for PANTALOONR

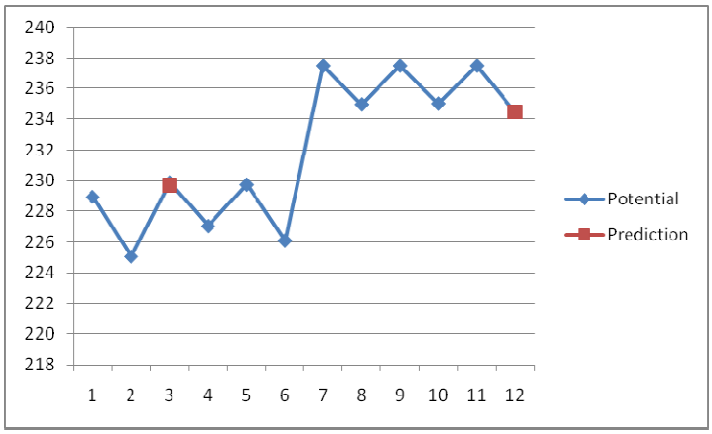

Figure 12:Plot of the Directional Changes for PANTALOONR

Predictions for RELCAPITAL

Potential	Olsen	Prediction
583.6	577.5	
577.5	583.3	
585.9	580	
578.2	584	
596	590	589.77352
589.2	595.1	
604.5	598.35	
592	597.95	593.19651
602	595.85	
593.15	599.1	
609.75	603.65	605.10994
602.1	608.2	
609	602.55	
605.6	605.6	605.6
49.324	**-29.022**	**-4.702**

Table 7: DC for RELCAPITAL

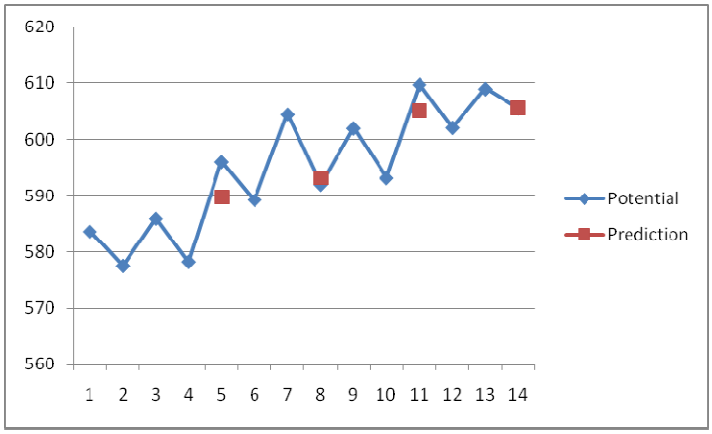

Figure 13: Plot of the Directional Changes for RELCAPITAL

Predictions for HDFC

Potential	Olsen	Prediction
1579.8	1562.35	
1545.55	1561.2	
1614.25	1597	1575.8266
1591	1607	
1649	1632.5	
1630.15	1630.15	1630.15
72.109	**-10.732**	**-55.019**

Table 8: DC for HDFC

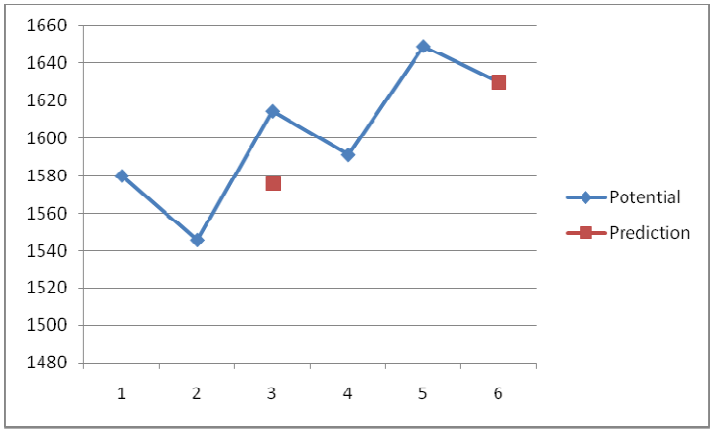

Figure 14: Plot of the Directional Changes for HDFC

Predictions for DLF

Potential	Olsen	Prediction
310	305.5	
305.5	309.4	
311.5	307	
294.65	298.35	303.3971
301	297.2	
295.1	298.8	
299	295.2	
293.25	296.95	
297	293	
292.5	296.2	
298.7	294	
295.95	295.95	295.95
38.664	**-5.333**	**-7.581**

Table 9: DC for DLF

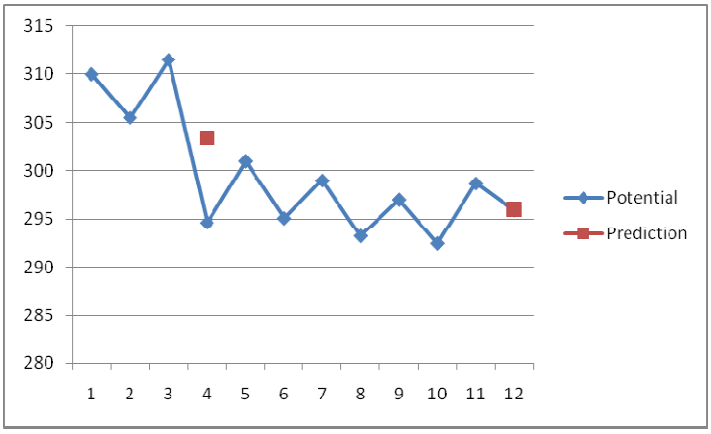

Figure 15: Plot of the Directional Changes for DLF

Predictions for INFOSYS

Potential	Olsen	Prediction
1165	1152.1	
1150	1161.5	
1184.7	1172.8	1169.7048
1172.8	1172.8	1172.8
24.838	**-11.456**	**-3.611**

Table 10: DC for INFOSYS

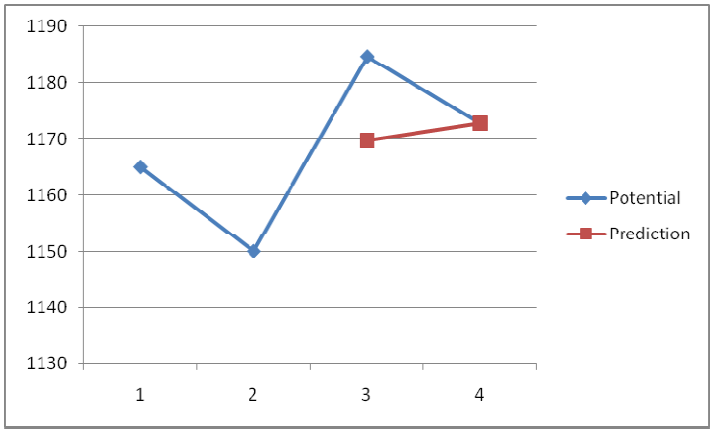

Figure 16: Plot of the Directional Changes for INFOSYS

Predictions for BHARTIARTL

Potential	Olsen	Prediction
714	706.55	
677.25	684.05	
685.1	677.7	682.9884
677.7	684.5	677.7
42.935	**14.485**	**4.987**

Table 11: DC for BHARTIARTL

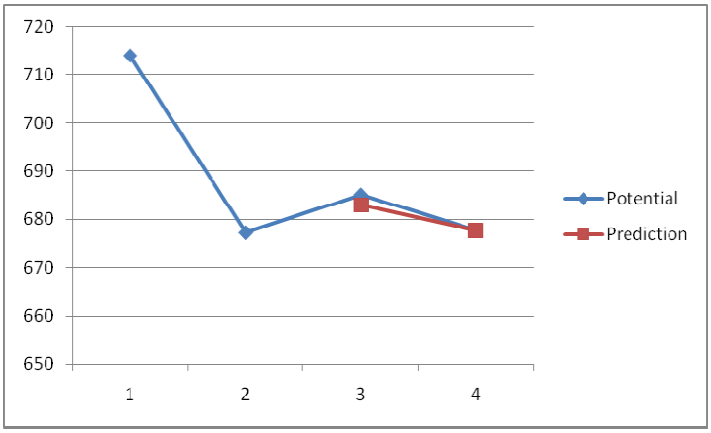

Figure 17: Plot of the Directional Changes for BHARTIARTL

Predictions for HEROHONDA

Potential	Olsen	Prediction
802.9	798	
774	779	
786	778	
778	783	
785.35	777.9	
777.05	781.8	
781.9	777.05	
776.4	781.4	
785.75	781	785.17169
778.5	783.5	
783.9	779.15	
776	780.9	
781.45	776.2	
770.05	774.7	772.63109
782.95	778	777.95681
776.1	781	
782	777	
778.45	778.45	778.45
81.447	**-7.649**	**11.017**

Table 12: DC for HEROHONDA

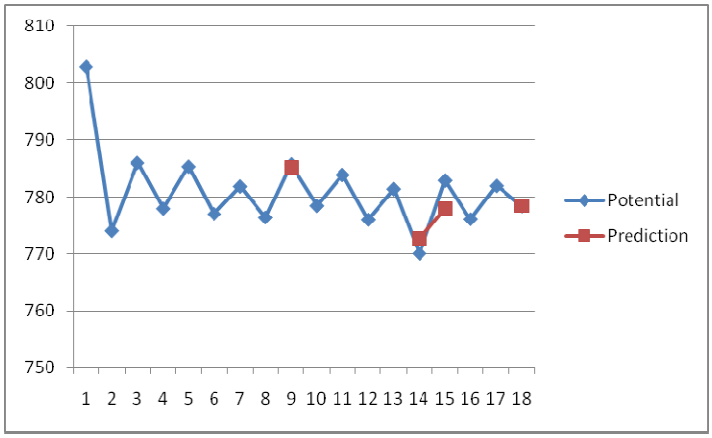

Figure 18: Plot of the Directional Changes for HEROHONDA

Predictions for RELIANCE

Potential	Olsen	Prediction
1306.7	1297.35	
1296.55	1305.8	
1322	1312	1313.1059
1312	1321.2	
1324	1314.7	
1312.2	1321.4	
1343.7	1333.3	
1332.3	1341.8	
1347.5	1338	
1334.55	1343.9	
1348	1337.2	
1337	1346.4	
1377	1365.85	
1365.85	1365.85	1365.85
70.216	**-56.178**	**-53.323**

Table 13: DC for RELIANCE

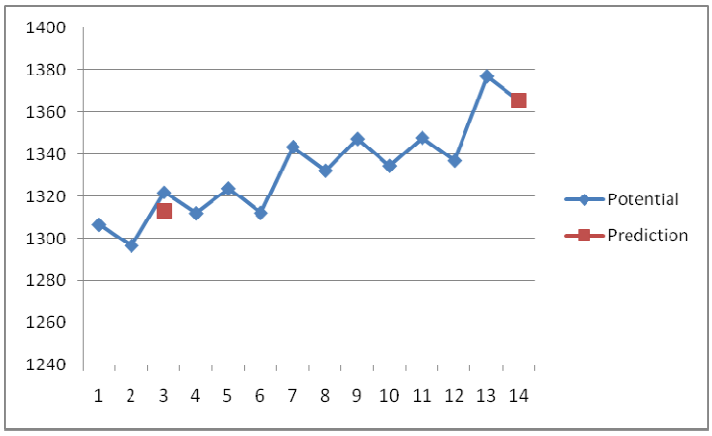

Figure 19: Plot of the Directional Changes for RELIANCE

Predictions for WIPRO

Potential	Olsen	Prediction
248	245.25	
245.25	247.9	
249.8	247	
246.05	248.65	
250.95	248.35	250.60221
243.7	246.15	246.29361
247.4	244.8	
241.75	244.25	242.83282
245.4	242.9	243.75558
242.15	244.6	
245.4	242.7	
242.55	245	
24.199	**-6.851**	**3.699**

Table 14: DC for WIPRO

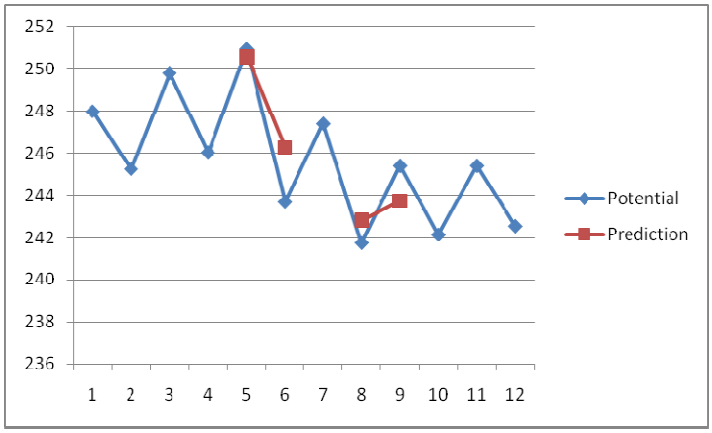

Figure 20: Plot of the Directional Changes for WIPRO

Predictions for TATASTEEL

Potential	Olsen	Prediction
239.25	236.15	
235.6	238.55	
239	236	
233.5	236.45	
240	236.55	239.68101
236.55	239.55	238.77023
11.972	-6.478	3.025

Table 15: DC for TATASTEEL

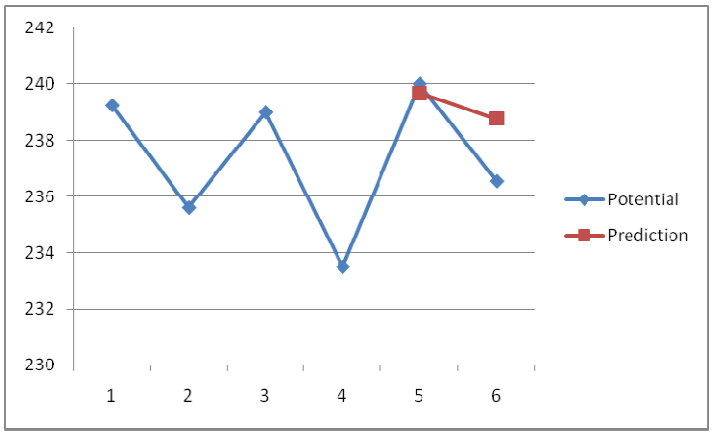

Figure 21: Plot of the Directional Changes for TATASTEEL

Predictions for UNITECH

Potential	Olsen	Prediction
48.85	47.85	
47.8	48.8	
50.15	49.1	49.531795
45.55	46.5	48.322794
47.6	46.55	47.407981
46.5	47.45	47.227831
6.624	**0.624**	**2.053**

Table 16: DC for UNITECH

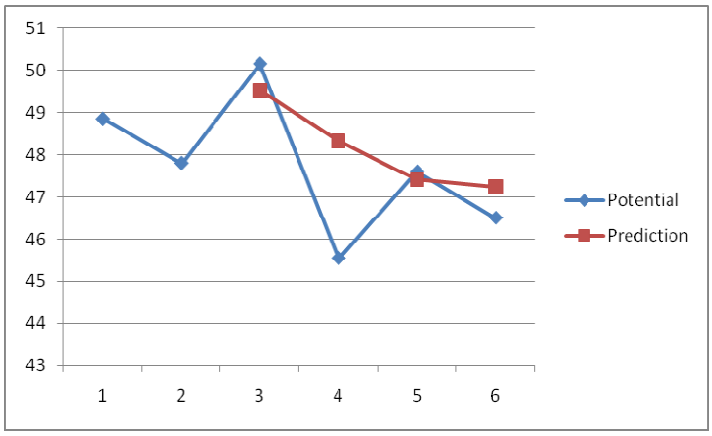

Figure 22: Plot of the Directional Changes for UNITECH

Predictions for SATYAMCOMP

Potential	Olsen	Prediction
181	175.05	
175.05	179	
180	176	
170.1	173.55	174.51277
173.9	170.3	
161.6	164.85	161.6
27.69	**3.491**	**-12.989**

Table 17: DC for SATYAMCOMP

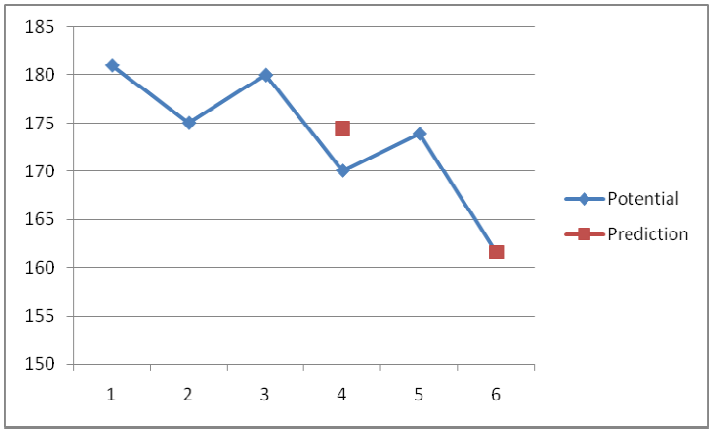

Figure 23: Plot of the Directional Changes for SATYAMCOMP

The summary of all the above fourteen stocks is given by Table 18 and the graph shown in Figure 24. This table and the graph shows the total profit made by each of the fourteen stocks using the algorithm given by Glattfelder, Dupuis and Olsen and the profit made by the predictions made by the artificial neural network. The values have also been compared to the potential profit that could have made by somebody having a perfect foresight of the market.

	Potential	**Olsen**	**Predicted**
ESSAR	7.858	-3.341	0.676
AXISBANK	21.971	-12.379	5.904
BHARTIARTL	42.935	14.485	4.987
PANTALLOONR	17.121	-15.978	-4.933
DLF	38.664	-5.333	-7.581
HDFC	72.109	-10.732	-55.019
HEROHONDA	81.447	-7.649	11.017
INFOSYS	24.838	-11.456	-3.611
RELCAPIRAL	49.324	-29.022	-4.702
RELIANCE	70.216	-56.178	-53.323
SATYAMCOMP	27.69	3.491	-12.989
TATASTEEL	11.972	-6.478	3.025
UNITECH	6.624	0.624	2.053
WIPRO	24.199	-6.851	3.699
Total	**496.968**	**-146.797**	**-110.797**
Average	**35.49771**	**-10.4855**	**-7.91407**

Table 18: net profit made by each stock using Olsen's algorithm and the artificial neural network

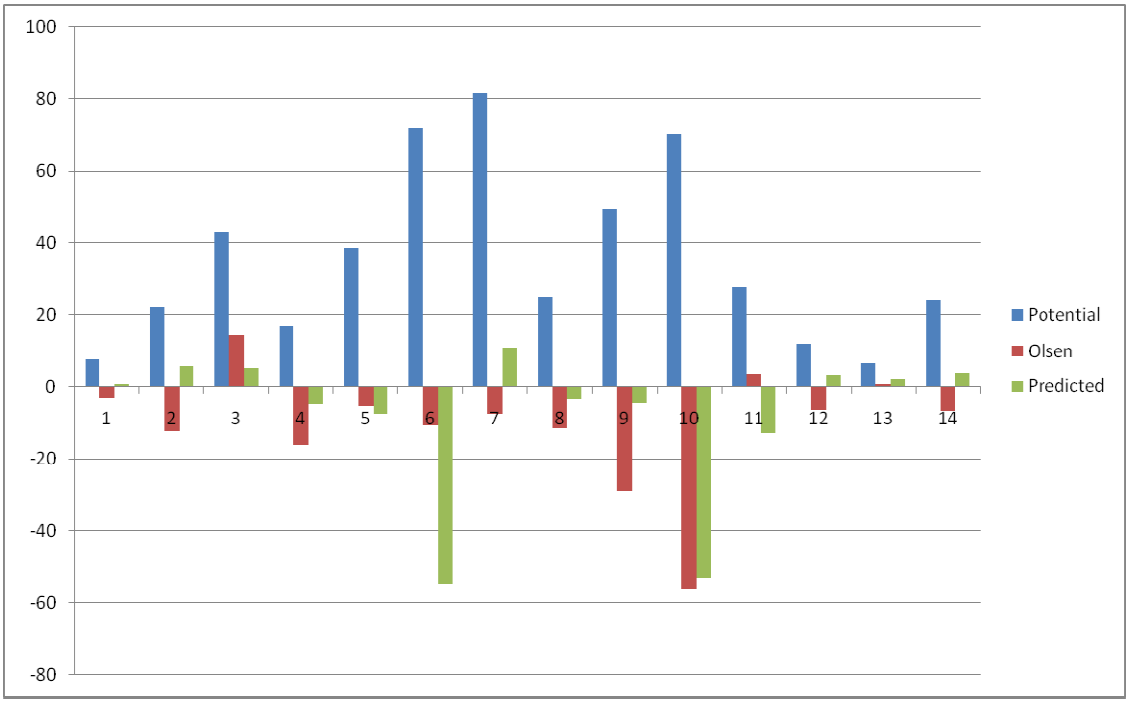

Figure 24: net profit made by each stock using Olsen's algorithm and the artificial neural network.

Chapter 5 – Conclusion & Further work

5.1 Conclusion

The uniqueness of this research comes from the fact that this will help in developing a new forecasting tool for the highly volatile Indian markets using the artificial neural network. In this paper we studied the directional changes that occur in the market for any given threshold and calculated the overshoot for every pair of successive directional changes. We attempted to investigate the use of an artificial neural network to find a relationship between two successive overshoots. A three layer feed forward-back propagation neural network with one input neuron, a hidden layer with 30 neuron and one output neuron was considered as the optimal network structure. The structure was tested with twenty two days of data for fourteen important stocks from different sectors of the market. Overshoot before the next directional change was taken as the input to the network and the next overshoot was predicted.

The output of the neural network is provided in the results. It was observed that although the neural network did not predict the overshoots accurately, for ten out of fourteen stocks, it made more profit (or less loss) than the original algorithm given by Glattfelder, Dupuis and Olsen which observed the directional change after the threshold λ was reached. Also we observed that the total return made by the neural network was better than the total return observed by the original algorithm.

All the above results show that the neural network gives a promising direction to the study of prediction of stock market trends.

5.2 Further work

This research has proposed a method to forecast overshoots in the price of a stock before the next directional change occurs in the market. The research uses artificial neural networks and compares the result with the potential profit in the market and the existing algorithm.

The methodology used in this research leaves much scope for improvement. A neural network with better structure having different numbers of neurons in input or hidden layer can be developed. Other market variables or their combinations can be used as inputs to the neural network to improve the predictability of the overshoots. The network that we

developed was trained only for 22 days of data because of the exorbitant time required to import and process the data before it could be used for training. Also the network was tested for fourteen stocks from the market. A system can be better trained in the future with more number of stocks and trained for more number of days.

Bibliography

1. Azoff, E. M. (1994). *Neural Network Time Series Forecasting of Financial Markets.* West Sussex, U.K.: John Wiley & Sons Ltd.

2. Bisig, T., Dupuis, A., Impagliazzo, V., & Olsen, R. B. (2009). *The scale of Market Quakes.* Working paper, arXiv:0909.1690v1.

3. Fast Artificial Neural Network . (2010, september 03). *Fast Artificial Neural Network – FANN 2.1*. Retrieved from http://leenissen.dk/fann/

4. Glattfelder, J. B., Dupuis, A., & Olsen, R. B. (2010). *Patterns in high-frequency FX data: Discovery of 12 empirical scaling laws.* Working paper, arXiv:0809.1040v2.

5. Interactive Brokers Group, Inc.'s. (2010, september 3). *Interactive Brokers-Commissions*. Retrieved from Interactive Brokers: http://www.interactivebrokers.co.in/en/p.php?f=commission

6. J.Deboeck, G. (1994). *Trading on the Edge.* (G. J.Deboeck, Ed.) USA: John Wiley & Sons, Inc.

7. Jang, G.-S., & Lai, F. (1994). *Trading on the Edge - Intelligent Trading of an Emerging Market.* (G. J. Deboeck, Ed.) USA: John Wiley & Sons, Inc.

8. Klimasauskas, C. C. (1994). *Trading on the Edge- Neural Network Techniques.* (G. J. Deboeck, Ed.) USA: John Wiley & Sons, Inc.

9. Majumdar, M., & Hussain, M. A. (2010). *Forecasting ofIndian Stock Market Index using Artificial Neural Network.* Mumbai, India: National Stock Exchange of India.

10. Mohan, N., Jha, P., Laha, A. K., & Dutta, G. (2006). Artificial Neural Network Models for Forecasting Stock Price Price Index in Bombay Stock Exchange. *Journal of Emerging Market Finance* .

11. National Stock Exchange of India. (2010). http://www.nseindia.com.

12. Panda, C., & Narasimhan, V. (2006). Predicting Stock Returns: An Experiment of Artificial Neural Network in Indian Stock Market. *South Asia Economic Journal* .

13. Pring, M. J. (2002). *Technical analysis explained : the successful investor's guide to spotting investment trends and turning points.* New York: McGraw-Hill.

14. Wilde, P. D. (1997). *Neural Network Models.* London: Springer-Verlag London Limited.

15. Yoda, M. (1994). *Trading on the Edge - Predicting the Tokyo Stock Market.* (G. J. Deboeck, Ed.) USA: John Wley & Sons, Inc.

Appendix A

Codes developed in VB.NET for processing of the data are given below. (The data has already been imported to the database using the sql queries as explained in Chapter 3)

The code for the function **calculate_total_profit** is given below. The function calculates the directional changes observed by a trader having a perfect foresight and a trader using the algorithm given by Glattfelder, Dupuis and Olsen. The function takes two inputs, **fl** which is the name of the table (for example **20081201AXISBANK**) from which the data is to be extracted and **connec** which stores the Connection to the database.

```
Sub calculate_total_profit(ByVal fl As String, ByVal connec As OdbcConnection)
    Dim position As Integer = 0                     '0 means position closed,
    '                                               'positive means we have bought extra (long position)
    '                                               'negative means we have sold more (short position)
    Dim trend As Char = CChar("u")                  'Start with an uptrend
    Dim bp As Double = 0                            'Benchmark price
    Dim oda As New OdbcDataAdapter("select * from " + fl, connec)
    Dim ds As New DataSet
    oda.Fill(ds)                                    'data set contains time n price for 1 day
    Dim i As Integer
    For i = 0 To ds.Tables(0).Rows.Count - 1        'For each trade
        If trend = CChar("u") Then                  'If uptrend
            If ds.Tables(0).Rows(i)(1) > bp Then    'If price moves further up
                bp = ds.Tables(0).Rows(i)(1)        'Update Benchmark Price
            ElseIf ds.Tables(0).Rows(i)(1) <= bp * (1 - delta) Then 'If price falls by delta => DC..!!
                trend = CChar("d")                  'Trend change to downtrend
                benchmark.Add(bp)                  'Positive cuz sell...!!! (perfect forsight)
                olsen.Add(ds.Tables(0).Rows(i)(1)) 'Positive cuz sell...!!! (Olsen's algo)
                position = position + 1             'Open Position
                bp = ds.Tables(0).Rows(i)(1)        'Set new Benchmark (Now this is the new minimum point)
            End If
        ElseIf trend = CChar("d") Then              'If downtrend
            If ds.Tables(0).Rows(i)(1) < bp Then    'If price moves further down
```

```
                bp = ds.Tables(0).Rows(i)(1)        'Update Benchmark Price
            ElseIf ds.Tables(0).Rows(i)(1) >= bp * (1 + delta) Then 'If price rises by delta => DC..!!
                trend = CChar("u")                  'Trend change to Uptrend
                benchmark.Add(-bp)                   'Negative cuz buy...!!! (perfect forsight)
                olsen.Add(-ds.Tables(0).Rows(i)(1))  'Negative cuz buy...!!! (Olsen's algo)
                position = position - 1             'close/shortsell Position
                bp = ds.Tables(0).Rows(i)(1)        'Set new Benchmark (Now this is the new maxima point)
            End If
        End If
    Next
    'End of day position close---->
    If position <> 0 Then
        benchmark.Add((-1)*position * ds.Tables(0).Rows(i - 1)(1))
        olsen.Add((-1)*position * ds.Tables(0).Rows(i - 1)(1))
    End If
    benchmark.Add(-1)
End Sub
```

Another function **exportit** that receives the Symbol of the security (for example **AXISBANK**) and exports the training or the testing files is given below.

```
Sub exportit(ByVal symbol As String)
    Dim i As Integer
    Dim num As Integer = 0
    Dim ii As Integer
    Dim sum As Double = 0               'Profit using perfect forsight
    Dim sum1 As Double = 0              'Profit using olsen's algo

    ''''''A small adjustment to remove the overnight returns
    For ii = 1 To benchmark.Count - 1
        If benchmark(ii) = -1 Then
            num = num - 3
        End If
    Next
    num = num + benchmark.Count + 1
    ''''''
    Dim train As String = num.ToString() + Chr(9) + "1" + Chr(9) + "1"
    For i = 2 To benchmark.Count - 3
        If benchmark.Item(i + 1) = -1 Then      'ending of a day
```

```
            train = train + Chr(13) + (Math.Abs(((Math.Abs(benchmark(i)) - Math.Abs(benchmark(i - 1))) / (6 *
                                                                delta * Math.Abs(benchmark(i - 1)))))).ToString()
        ElseIf benchmark.Item(i) = -1 Then      'The last element of the day
        ElseIf benchmark.Item(i - 1) = -1 Then  '"-1" used as a day breaker
        ElseIf benchmark.Item(i - 2) = -1 Then  'The begining of a new day
            train = train + Chr(13) + (Math.Abs(((Math.Abs(benchmark(i)) - Math.Abs(benchmark(i - 1))) / (6 *
                                                                delta * Math.Abs(benchmark(i - 1)))))).ToString()
        Else
            'number of deltas in an overshoot untill next DC devided by six
            'devided by six because the output is typically between 0 to 6 and we want it to be between 0 and 1
            'and now display in terms of alternate input/output for the FANN
            train = train + Chr(13) + (Math.Abs(((Math.Abs(benchmark(i)) - Math.Abs(benchmark(i - 1))) / (6 *
                                                                delta * Math.Abs(benchmark(i - 1)))))).ToString()
            train = train + Chr(13) + (Math.Abs(((Math.Abs(benchmark(i)) - Math.Abs(benchmark(i - 1))) / (6 *
                                                                delta * Math.Abs(benchmark(i - 1)))))).ToString()
        End If

       ' Display Potential Trade Price
        RichTextBox1.Text = RichTextBox1.Text + Chr(13) + Math.Abs(benchmark.Item(i - 1)).ToString()
        'Potential Total Profit minus the transactional costs
        sum = sum + benchmark(i - 1) * (1 + (Math.Sign(benchmark(i - 1)) * 0.0004413))
        'Olsen's Trade Price
        RichTextBox3.Text = RichTextBox3.Text + Chr(13) + Math.Abs(olsen(i - 1)).ToString()
        'Olsen's Total Profit minus the transactional costs
        sum1 = sum1 + olsen(i - 1) * (1 + (Math.Sign(olsen(i - 1)) * 0.0004413))
    Next
    Label1.Text = Label1.Text + sum
    Label2.Text = Label2.Text + sum1
    Dim outfile As StreamWriter
    'export train/test files
    outfile = File.CreateText("D:\work\dissertation\training_files\" + symbol.Trim() + ".train")
    outfile.Flush()
    outfile.Write(train)
    outfile.Close()
End Sub
```

RichtextBox1 and RichtextBox3 now contain the directional changes as observed by a trader with perfect foresight and one using Olsen's algorithm.

To export training and testing files of different stocks we have to execute the functions given above many a number of times.

For example to export the training files for **AXISBANK**, we need to run `calculate_total_profit` for 22 days for **AXISBANK** data and then call `exportit`. Similarly to export the testing file, we need to run `calculate_total_profit` for the 23rd day of **AXISBANK** data and then call `exportit` to export the testing file.

(Note: we have to change the extension of the file from .train to .test while exporting it)

Once the training and testing files are exported, we can use the FANN library to get the outputs of a neural network and use the function given below to predict the directional changes

```
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click
    Dim str As String()
    str = RichTextBox5.Text.Split(",")      'FANN gives comma saperated outputs
    Dim prediction As New List(Of Double)   'Output of the ANN
    Dim PredictedProfit As Double = 0       'Total profit when u trade using the predictions
    Dim s As String
    For Each s In str
        prediction.Add(CType(s, Double))    'Prediction() now contains the predictions by the ANN
    Next
    Dim i As Integer = 0
    Dim p As Double = 0
    Dim position As Integer = 0
    Dim newp As Double = 0
    RichTextBox4.Text = "Prediction" + Chr(13)
    'For the first trade
    'First, we consider if we would have bought at benchmark then how would we have proceeded??
    p = (benchmark(i + 1))
    For i = 0 To prediction.Count - 1
        'p=previous buy or sell price
        'newp=predicted price using FANN outputs and the actual previous observed overshoot
```

```
            newp = -(benchmark(i + 1) + (Math.Abs(benchmark(i + 1)) * prediction(i) * 6 * delta))
            RichTextBox4.Text = RichTextBox4.Text + Chr(13)
            If (benchmark(i + 2) < newp) Then
                'trade at prediction
                If Math.Abs(position + 1 * Math.Sign(newp)) < 2 Then    'To prevent excessive loss
                    position = position + 1 * Math.Sign(newp)
                    PredictedProfit = PredictedProfit + (newp * (1 + Math.Sign(newp) * 0.0004413))
                    p = newp
                    RichTextBox4.Text = RichTextBox4.Text + Math.Abs(p).ToString()
                    If p > 0 Then   'If cost positive
                        RichTextBox4.Text = RichTextBox4.Text + Chr(9) + "BUY"
                    Else            'cost negative
                        RichTextBox4.Text = RichTextBox4.Text + Chr(9) + "SELL"
                    End If
                End If
            End If

        Next

        'Close position at day end
        If position <> 0 Then
            RichTextBox4.Text = RichTextBox4.Text + Chr(13) + (Math.Abs(benchmark(benchmark.Count - 1))).ToString()
            PredictedProfit = PredictedProfit - position * Math.Abs(benchmark(benchmark.Count - 1))
        End If
        Label7.Text = PredictedProfit
    End Sub
```

RichtextBox4 (a rich text box control) now has the predicted directional changes.

RichtextBox1, RichtextBox2 and RichtextBox4 can now be compared to get the final results.

Also label1, label2, and label7 can be compared to get the total profits observed by the three algorithms involved in this research.