THE UNIVERSITY OF NAIROBI

SCHOOL OF COMPUTING AND INFORMATICS

COMPARISON OF ACTIVATION FUNCTIONS FOR NSE STOCK PRICE PREDICTION.

KIHONGE LABAN KARIUKI

A RESEARCH PROJECT REPORT SUBMITTED TO THE SCHOOL OF COMPUTING AND INFORMATICS IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF M.SC. COMPUTATIONAL INTELLIGENCE OF THE UNIVERSITY OF NAIROBI.

NOVEMBER 2017

DECLARATION

I declare that this research project, as presented in this report, is my original work and has not been presented for a degree in any other institution of higher learning and that all sources I have used or quoted have indicated and acknowledged by means of complete references.

Signed: ________________________ Date: ___________________________

Laban Kariuki Kihonge

Registration number: P52/72878/2014

This research project has been submitted for examination with my approval as University Supervisor.

Signed: ________________________ Date: ___________________________

Dr. E. T. Opiyo Omulo

Senior lecturer, SCI UON.

ACKNOWLEDGMENT

I would like to extend my gratitude to my supervisor, Dr. E. T. Opiyo Omulo for his persistent support, guidance, and encouragement during the undertaking of this research project. I am thankful for his willingness to share his time, advice, and expertise which helped me in carrying out my project idea from conception to completion. I am grateful for the many helpful interactions and discussions that I had with fellow students as they provided useful insight on how I should achieve my project milestones. I would also wish to thank my parents, brother and friends for their encouragement and spiritual support throughout the entire study period.

ABSTRACT

ANN timeseries prediction has been successfully implemented and tested on NSE stocks prediction by Barack (2014) and other stock exchanges (Safi & White 2017) however these studies focused on factors like training algorithm, network sizing and learning parameters bypassing selection of the activation functions. With the different type's activation functions currently available for use in ANN, further studies needed to be done to test if the activation functions or their combinations can improve performance of the ANN in stock price prediction. Research has proved that activation function to be one of the essential parameters of an artificial neural network (ANN) whose performance can be improved by use of various activation functions and their combinations. Ozkan and Erbek (2003), Sibi, Jones and Siddarth (2013). The study involved implementing of different networks containing varied activation functions being trained and tested on NSE data to measure their performance. RMSE and MSE were used as the basis of evaluating the accuracy of predictions. The study came to the conclusion that a network of S-SF-SF-S "sigmoid-softmax" performed best with minimal training i.e. 100 epochs and its performance degraded if the further trained. The activation functions of T-T-T-T "hyperbolic tangent", L-L-L-L "linear", S-T-T-S "sigmoid-hyperbolic tangent" and S-S-S-S "sigmoid" followed respectively according to the majority of the stocks tested though they required more training reaching up to 2000 epochs. Generally, it was observed that changing the activation function has either a positive or a negative effect on the performance of the ANN. Further comprehensive research on building better ANN models for other areas where ANNs are applied other than prediction i.e. image recognition, classification etc. as the research has proven changing the activation function can have a positive or negative effect on the ANN performance.

Table of Contents

List of figures

List of tables

List of abbreviations and acronyms

ANN	Artificial neural network
NSE	Nairobi securities exchange
KEGN	Kengen
EABL	East African Breweries Ltd
CMA	Capital market authority of Kenya
MSE	Mean square error
RMSE	Root mean square error
FE	Forecast error
NMG	Nation Media Group
CBK	Central bank of Kenya
EQTY	Equity banking
SASN	Sasini
SF	Softmax activation function
G	Gaussian activation function
T	Hyperbolic tangent (tanh) activation function
S	Sigmoid activation function.
L	Linear activation function

1.0 Introduction

1.1 Background

Nairobi Securities Exchange (NSE) is Kenya's main bourse offering an automated trading system (ATS) platform for companies to float and trade of their Shares and bonds with the aim of raising capital for their operations.

NSE operates under the jurisdiction of the Capital Markets Authority (CMA) of Kenya.

Shares at the NSE are generally grouped into eight main sectors which are automobile and accessories e.g. car & general Kenya limited, marshalls east Africa limited etc, Commercial and Services e.g. Kenya airways, express ltd, nation media group etc, energy & petroleum e.g. kenGen limited, kenol kobil etc Agriculture e.g. Eaagads ltd, kakuzi, limuru tea etc, Financial e.g. Barclays bank of Kenya, Kenya commercial bank, national industrial credit bank etc, construction & allied athi river mining, east African cables limited etc, insuarance e.g. britam holdings limited, Kenya reinsurance corporation limited etc and manufacturing & Allied e.g. BOC Kenya limited ,east Africa breweries, mumias sugar company limited etc sectors from which investors can choose invest to.

NSE generally operates on willing buyer willing seller concept where stock brokers and investors can sell and buy shares and bonds at the stock exchange in place of the buyers and sellers of shares. Stockbroker's act as middlemen in the stock market connecting buyers with sellers and earn a commission on trading activities and are also responsible for advising their investors on NSE trades.

Investors at NSE enjoy two main types of benefits of owning shares through:

1- Capital Gains – where shares are bought at a low share price and sold at a high share price.
2- Dividends – this is something of monetary value regularly e.g. quarterly or yearly given to shareholders of a company e.g. money or extra company shares.

Investors with Central Depository System accounts (CDS) can buy and sell stocks through stockbrokers from the NSE and CMA List licensed and approved trading participants who are listed on the central bank of Kenya (CBK) and NSE websites in minimum lots of 100 shares.

Investors may personally do research on stocks they intend to buy or rely on their stockbrokers opinions based on their stock analysis.

The Nairobi Securities Exchange (NSE) has been in Kenya since the British colonized Kenya with its operations been reported in as early as 1920's. (IFC/CBK, 1984).

The NSE was officially constituted in 1954 voluntary association of stockbrokers from an unofficial stock exchange involving gentlemans agreement. NSE continued to grow to be even recognized by London stock exchange (LSE) in 1993 and getting more responsibilities and rules in regulation of trading activities through CMA and societies acts. (NSE 2017)

NSE has three seven indices to provide a comprehensive measure of performance of the NSE which are:

NSE 25 share index

Financial times stock exchange NSE Kenya 15 index

NSE all share index

NSE 20 share index

Financial times stock exchange NSE Kenya 25 index

Financial times stock exchange NSE Kenyan shilling government bond index

Financial times stock exchange African securities exchanges Pan African index (NSE 2016)

1.2 Problem statement

Research by Ozkan and Erbek (2003) have proved that activation function to be one of the essential parameters of a neural network. Further research by Sibi, Jones and Siddarth (2013) also proved that artificial neural network (ANN) performance can be improved by use of various activation functions and their combinations.

While ANN timeseries prediction has been successfully implemented and tested on NSE stocks prediction by Barack (2014) and other stock exchanges (Safi & White 2017) these studies focused on factors like training algorithm, network sizing and learning parameters bypassing selection of activation function.

With the different type's activation functions currently available for use in ANN, further studies need to be done to test if the activation functions or their combinations can improve performance of the ANN.

1.3 Objectives

The main objective of this study is to improve ANN time series NSE stock prediction model by testing various transfer functions effects on the ANN time series prediction

1.3.1 Specific objects

- To do comparison of linear, hyperbolic tangent (tanh), logistic (sigmoid), Gaussian, linear and softmax activation functions in artificial neural network time series prediction on NSE stock price.
- Assessing the performance of hybrid activation functions in artificial neural network time series prediction on NSE stock price
- Identifying the optimal activation functions for use in ANN timeseries NSE stock price prediction.
- To apply the optimal neural network model to NSE with results accessible via a website

1.4 Assumption

This study assumption is trading at the stock market was based on the capitalistic system supply and demand, where there were willing buyers and willing seller with undue extreme external factors like political or economic locally and globally with a direct effect on stock trading.

1.5 significance

Data produced from this study will be used to show the comparison of various activation functions with the aim of furthering the accuracy and efficiency of ANN NSE stock price prediction. In addition this project is hoped will be a beginning of ongoing research into ANN application in NSE.

1.6 Justification

Technical analysis is favored by many NSE stock brokerage firms during decision making on buying and selling of stocks and non uses artificial intelligence especially neural networks. (Barack Wanjawa, 2014)

By experimenting with neural networks activation function the optimum configuration can be determined especially for the case of Nairobi securities exchange with the aim of attracting NSE stock brokerage firms towards ANN decision making.

1.7 methodology

I will implement the project through python pybrain neural networks library with the different activation function for testing with historical data bought from NSE authorized data vendors.

2.0 Literature review

2.1 NSE

Stock exchanges play a vital role in the economy of their countries providing measures e.g. stock prices for economic health of a country. Predicting economic indicators e.g. unemployment rate, stock prices, poverty rate etc. helps decision and policy makers in setting up policies.

NSE is critical and vital to the overall growth of the Kenya economically through encouraging of savings & investment, as well as helping local and international companies' access cost-effective capital.

Investment at the NSE in stocks involves buying and selling of shares with expectation of profit from capital gains when the share price rises or from dividends announced from listed companies. NSE is tracked and analyzed by Kenya National Bureau of Statistics (KNBS) specifically the NSE 20-share index which is then included the annual Kenya economic survey under money banking and finance section .(economic survey 2013,2014,2015,2016)

Investors who have CDS accounts can trade in shares through the 23 stock brokers accredited by the NSE, CMA, Central Depository and Settlement Corporation (CDSC) and CBK. (NSE 2017)

Investors with CDS accounts who want to buy or sell stocks may contact their stock brokerage firms to place an order of either sell or buy through stock broker who is an individual licensed by the CMA to trade securities at the NSE on instructions of their investors and he or she may earn a brokerage commission which is a preset percentage of value traded. The stock broker will in turn access a pool of offers for sale at NSE and place an offer for sale if the investor had placed an order for sale or the stock broker may purchase on behalf of the investor. The stock broker must then inform of the CDSC of the transactions so as the investor CDS account is update to reflect the shares. Trading of shares valued less than Kenya shilling (Kshs) 100,000 attract a charges which include brokerage commission and statutory charges of 1.85% while shares greater than Kshs 100,000 attract 2.12% charges in accordance with Capital Markets Act, Cap 485A.(Capital Markets Authority, 2013).

According to NSE amended equities securities trading rules (2017) there are four types of trades that can placed by the stock broker namely; "Immediate or cancel" where the offer to trade is available immediately and its cancelled immediately if there is not any matching offer," Good Till Cancelled (GTC)" where the offer is stays until 30 days lapse, "Good Till Day (GTD)" where the offer is valid for 5 days and finally there is the "Day Order (DO)" which is valid for the trading day only.(NSE 2017)

In a research case study of among teachers in Kisumu municipality, which attempting to understand the teachers thought process in during investing at the NSE Ndiege C O reached a conclusion that the teachers may try to be rational during decision making on which stock or shares to buy or sell but due to their limited or lack of knowledge on listed companies fundamentals and other factors that affect the stock price they or not able to understand and interpret available data optimally. Investment decision making was not based on company fundamentals or risk expected. (Ndiege C. O., November, 2012)

Individual investment decisions are usually influenced by several biases. They showed i.e. overconfidence, loss aversion, price changes, etc. highly influenced individual investment choosing with however mental accounting is less affected (Omery C.S., October, 2014)

There are generally two categories of investors at the NSE who are long term investors who time duration between buying and selling of shares is within a range one year or never selling hence leaning on value investing while short term investors the time duration between buying and selling of shares is within a range of few hours, days or weeks hence leaning on speculative trading. Short term investors' usually use work experience or analysis tools with the aim of predicting future stock price e.g. Machine learning, Fundamental/technical analysis etc.

2.2 neural network

The brain is one of the major components of the body which is consists of billions of interconnected neurons. A normal functional neuron consists several biological parts but the basic parts are the dendrites seen as connection points receiving input from previous neuron layers or act as the input layer, cell body which is seen as central command of the neuron and finally the axon which is a connecter of the central command to synapses. Inputs are received on dendrites from neighboring neurons or external environment, data is then processed independently in the neuron where if the processed data passes a certain threshold it is transmitted through the neurons axon to synapse then connected to other neurons.

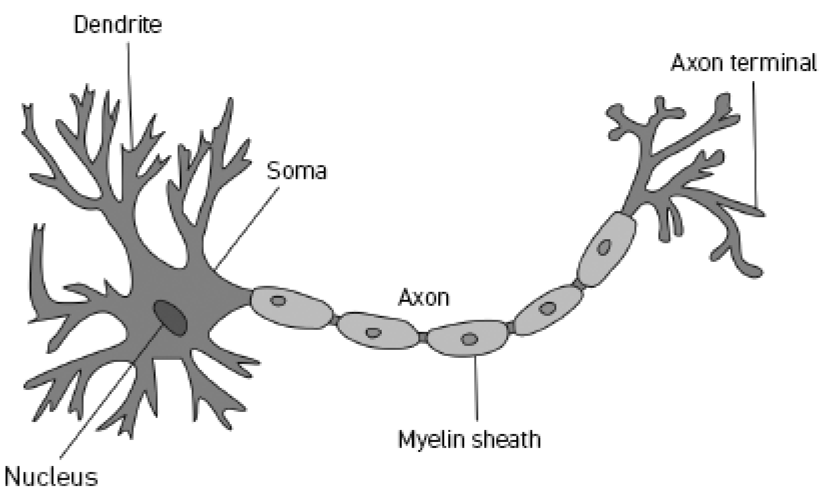


Figure 2.1 - neuron illustration (Quasar Jarosz, 2009)

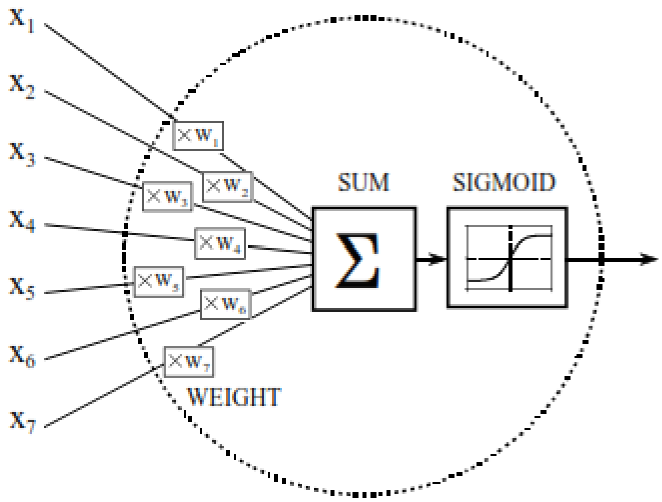
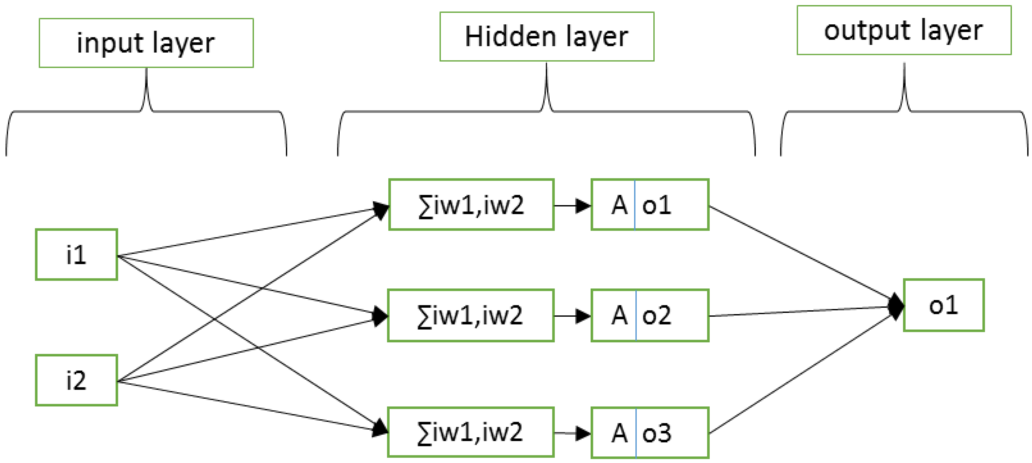


Figure 2.2 - structure of artificial neuron (**Steven Smith, 1997**)

The concept of ANN has been around since 1940's when Walter Pitts and Warren McCulloch produced a research paper together with an electrical circuit model showing how artificial neural networks may have worked.

The idea of using ANN in the field of prediction was first brought up by Hu (1964) and it was focused on weather forecasting but in 1980s did ANN gain significant use in scientific research in economics and finance when Halbert White (1980) published the first significant study on using ANN timeseries for stock prediction.

An ANN attempts to imitate a biological neuron where it has the first stage being an input layer which is connected to intermediary stage called the hidden layer which is finally connected to the last stage referred to as the output layer. In all stages the neurons in the layers are connected together with weights of different levels as illustrated on **figure 2.3** bellow.



Legend	
i	Input
w	Weight
A	Activation function
o	output

Figure 2.3 – a simple of ANN of 2:3:1. (Source: author)

2.2.3ANN major characteristics comparison to the brain.

An ANN attempts to imitate a biological neuron with the aim of gaining some characteristics of the brain in processing data which according to D. Kriesel, Priyanka W. and Sonali B. M. include:

1) Self-Organization and adaptive leaning: The brain can reorganize itself during its lifespan giving one the ability to learn by correcting errors. By ANN having a capability to learn it saves one from explicitly programming an ANN since it can learn from training examples finding reasonable solutions to similar problems in the same class as training examples.
2) Generalization capability: just as one can drive on new roads or a rats find the cheese in a rat maze the brain can generalize and associate past experience "training" hence can apply solution from past similar situation
3) Parallelism and Real time operation: due to the advantages capabilities and opportunities offered by ANN, specialized hardware components are been made which can take advantage of ANN abilities e.g. parallel ANN computation.
4) Fault Tolerance: just as neurons in the brain continually reorganize themselves or by external factors e.g. environmental influence, alcohol consumption etc. but the cognitive abilities of people are not significantly affected thus the brain is tolerant against internal and external errors, ANN has redundant information coding which helps retain some of the ANN abilities even with partial ANN network degradation (Kriesel D. 2007; Sonali B. et al 2014)

2.3 The working of ANN

ANN architecture mainly includes the input, output and hidden layers, transfer or activation functions and sometimes a bias which is a predetermined value and optional is summed with weighted sum of an ANN layer. The layered arrangement of neural nodes in layers is what generally determines the neural network architecture hence which area it's likely to be used in. The input layer does not perform any computation thus does not modify the data values it receives but its number of neurons provide an input data parameter ie a 5:21:21:1 neural network architecture will allow only five inputs to the ANN. The input layer nodes receive single value which it then transfers the value to all the nodes to the adjacent nodes depending on the ANN design.

In the active layers the values received previously are all multiplied by weights. The new values from multiplication of weights and inputs are all added up and passed through a mathematical function also referred to as activation function. This determines the output of the node

2.3.1 Basic artificial neuron workings

All layers in the neural network are constituted of neurons with the input layer neurons being passive as they only pass values received. Other neurons in the hidden and output layer are active where actual data processing takes place as they receive and modify data before transmitting its output to the next layer or as final output.

For neurons in the active layer, they receive input from previous layer where all their outputs are summed up, a bias may also be added for the hidden layer. The summed value is passed through an activation function where it either passes or misfires.

If the summed value it does pass the activation function threshold it is forwarded to all subsequent nodes in the following stage of neurons together with all the output from neurons in the same layer.

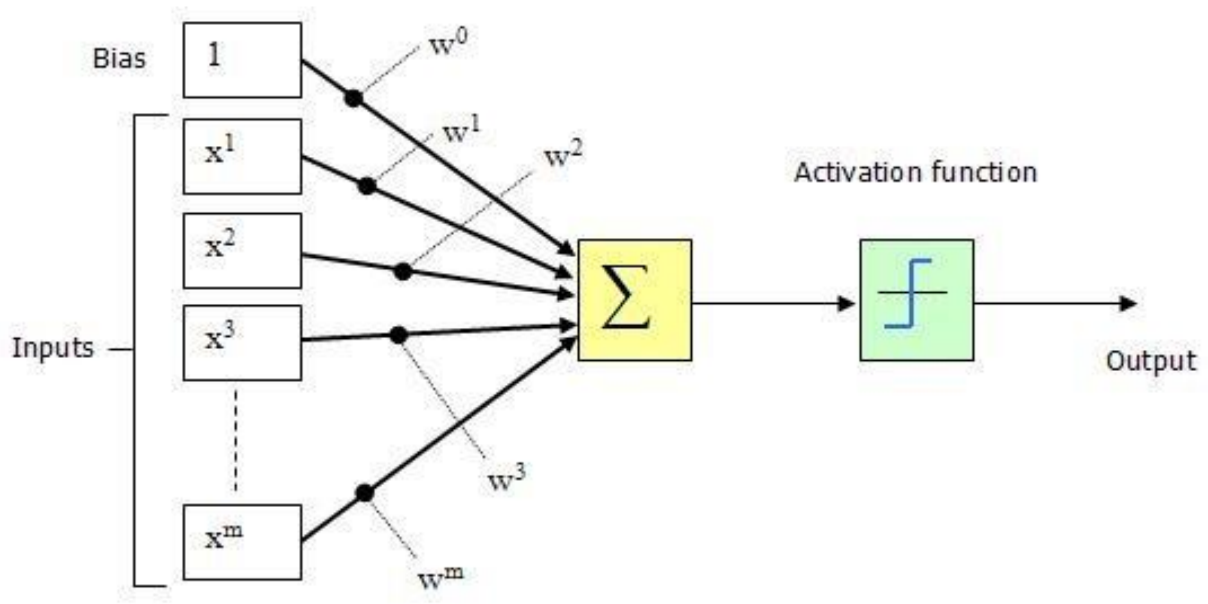


Figure 2.4 basic ANN neuron working

Multilayer perceptron (MLP) one of the most popular nonlinear network topologies usually with backpropagation training algorithm are accurate to a high degree in function approximation and learning. (Rohit R. D., 2012, White, 1992)

2.4 Data processing

Data used by the neural network needs to be cleaned and normalized for the optimal results to be obtained from the ANN.

This is done to prevent false positives during processing by the neuron hence increasing ANN accuracy.

Feature scaling through rescaling method could be used but it only scales to values between [1,0] one and zero while some activation functions require normalization of between negative one and positive one [-1,1]. While there are various complex methods that can be used for standardization and normalization, most didn't fit my problem or were unnecessarily complex for a simple problem thus I followed Occam's razor principle of choosing the simple process as explained in section **2.4.1** and **2.4.2**.

2.4.1 Preprocessing

Before data is entered to the neural network it is scaled to values between one and zero or negative one and positive one depending on the activation functions to be by the neurons in the ANN.

Formula steps used for scaling before the data is processed is:

Given some data for a function for scaling i.e. $LAB_{1,}LAB_{2,}LAB_{3,}LAB_{4,} \ldots., LAB_n$.

The solution is given to the function will be i.e. $lab_{1,}lab_{2,}lab_{3,}lab_{4,} \ldots., lab_n$. For $LAB_{1,}LAB_{2,}LAB_{3,}LAB_{4,} \ldots., LAB_n$.

Solution is given by $lab_1 + [(LAB - LAB_1) \propto]$ where LAB is $LAB_{1,}LAB_{2,}LAB_{3,}LAB_{4,} \ldots., LAB_n$

Where $\propto = (lab_n - lab_1) \,/\, (LAB_n - LAB_1)$

A practical example of application of scaling is;

Sample range data = [5, 6, 7, 8, 9, 10]

$LAB = [6]$ A sample value between the data range.

$lab_1 = [0]$ New range lower value.

$lab_n = [1]$ New range upper value.

lab = New value from the range

$\propto = (1 - 0)/(10 - 5) = 0.2$

$[(6 - 5)0.2] + 0$

$[1 * 0.2] + 0 \ = 0.2$

$0.2 + 0 \ = 0.2$

$lab = 0.2$

2.4.2 Post-processing

The output from the ANN is scaled to the original scale to be able to get the true value predicted. The reverse of scaling used to scale the data for input to the ANN is used to scale it back up again.

Formula steps used for scaling before the data is processed is:

Given some data for a function for scaling i.e. $lab_{1,} lab_{2,} lab_{3,} lab_{4,} \ldots., lab_n$.

The solution given to the function will be i.e. $LAB_{1,} LAB_{2,} LAB_{3,} LAB_{4,} \ldots., LAB_n$ for $lab_{1,} lab_{2,} lab_{3,} lab_{4,} \ldots., lab_n$.

The solution is given by $LAB_1 + [(lab - lab_1) \propto]$ where S is $lab_{1,} lab_{2,} lab_{3,} lab_{4,} \ldots., lab_n$.

Where $\propto = LAB_n - LAB_1 / lab_n - lab_1$

A practical example of application of scaling is;

S range data = [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]

X range data = [5, 6, 7, 8, 9, 10]

$lab = [0.2]$ A sample value between the data range.

$LAB_1 = [5]$ Original range lower value.

$LAB_n = [10]$ Original range upper value.

LAB = original value from scaled range.

$\propto = (10 - 5)/(1 - 0) = 5$

$[(0.2 - 0)5] + 5$

$[0.2 * 5] + 5$

$1 + 5 = 6$

$LAB = 6$

2.5 application of ANN

ANN applicability in non-linear machine learning problems makes it particularly useful in stock price prediction but neural networks are also used to compute various complex tasks which in the fields of prediction, classification, data processing, robotics and computer numerical control which include;

- **Detecting Extreme Weather in Climate Datasets -**deep Convolutional Neural Network (CNN) classification system have been used in detecting extreme weather events e.g. Tropical Cyclones, Atmospheric Rivers etc. in large climate datasets.(Yunjie Liu et al, 2016)

- **Measuring the Osteoarthritis Severity from x-ray-** X-rays taken are taken through image processing are used to classify rheumatology on the four grade of severity. Backpropagation ANN have been proven to identify the osteoarthritis severity by using the x-ray image features which include color and texture to high accuracies of up to 66 %. (Pratiwi D., et al, September 2013**)**

- **Automatic Machine Translation –** While language translation algorithms have around for a while automatic translation of text and images has been made possible by long short term memory architecture (LSTM) recurrent neural networks (RNN) in the Google translate mobile app. The Google translate mobile app allows instant camera translation in over 30 languages, translation between 103 languages while typing "52 languages while offline" and handwriting recognition through drawing characters instead of using keyboard. The application may be useful in translating information displayed e.g. hotel menus, tickets pricing, product labeling etc. from languages unknown to the user to known language by use of camera. The ANN have been designed and optimized to run on smartphone resources without uploading data to data center servers to be processed. (Otavio G, 2015 [Google research blog])

- breast cancer detection gigapixel Pathology Images. Detecting Cancer Metastases is usually done by human pathologist carefully going through biological tissues where human errors can occur due to various factors e.g. fatigue etc. Detecting Cancer Metastases on Gigapixel Pathology Images has been accomplished using convolutional neural network (CNN) architecture which Detected roughly 92 percent tumors compared to experienced pathologist whose search had a sensitivity levels of up to 73% on the same data. (https://arxiv.org/abs/1703.02442)

- autonomous cars - autonomous cars use various techniques to detect and analyze their environments e.g. GPS, laser light, cameras "computer vision" etc. which produce huge complex data. The data needs to be processed and interpreted fast enough by the autonomous car systems so as to determine their next course of action while driving. while autonomous cars have been existed from the early 1980s e.g. Carnegie Mellon University's "Navlab" controlled by ALVINN which had single hidden layer back-propagation network.(Pomerleau A. D., 1989) .since 2010 interest in autonomous cars has been high with many technology, automobile manufacturing companies and learning institutions collaborating to develop autonomous cars with automatic self-parking systems being one of the advantages currently to automobile manufacturers.

There exist two categories of robotic visual self driving systems which are mediated perception approach which involves parsing entire scenes and behavior reflex approach which involves blindly mapping an image directly to driving commands. Deep Convolution Networks architecture

Direct perception approach which uses deep convoluted neural network maps an Received data from external sensors ie cameras to ANNs that have been trained on road and traffic rules and tested and performed well in virtual and real environment. (chenyi C. et al, 2015)

2.6 activation functions

2.6.1 The logistic (sigmoid) function:

Sigmoid function has an output range of [0, 1] and its plotting produces an 'S' like curve as illustrated in figure2.5.

The mathematical equation for sigmoid function:

$$f(x) = \frac{1}{1 + e^{-x}}$$

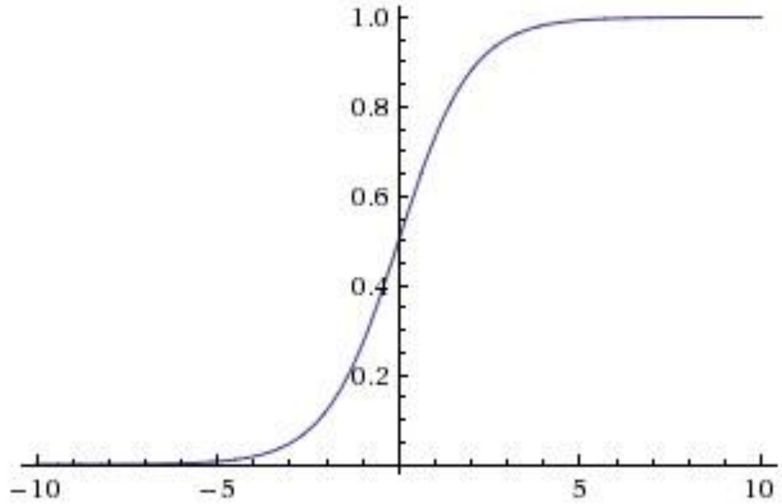


Figure 2.5 sigmoid function graphical representation

2.6.2 The hyperbolic tangent (tanh):

The tanh activation function is the ratio between cosine & sine mathematical functions. Tanh activation function works the same as sigmoid function but in a rescaled method whose range output is [-1, 1] unlike sigmoid functions whose output is between [0, 1] as shown in figure 2.6.

The equation for the tanh function is:

$$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$$

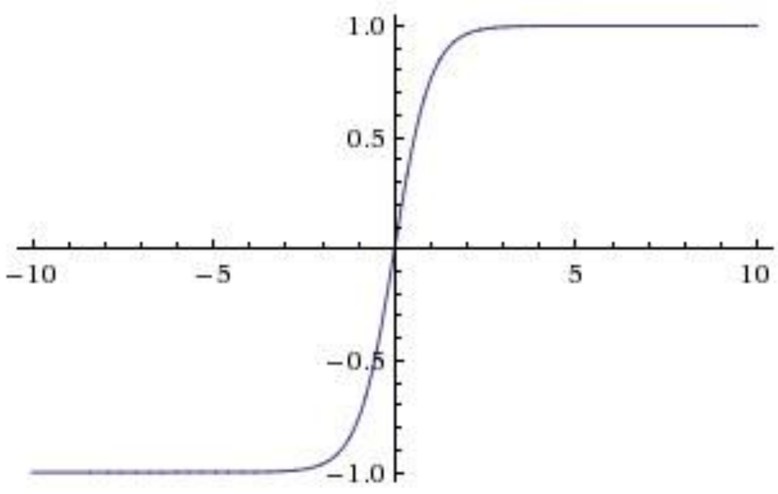


Figure 2.6 hyperbolic tangent function graphical representation

2.6.3. Softmax activation function:

Softmax is usually implemented a binary classifier with an output range of [0, 1].

The equation for softmax activation function is:

$$f(x)_i = \frac{e^{x_i}}{\sum_{k=1}^{K} e^{x_k}} \quad for\ i = 1, \ldots., K$$

2.6.4. The linear activation function:

Linear activation function output range is (-∞, ∞) and its equation is:

$$f(x) = x$$

2.6.5 Gaussian activation function

Gaussian function has an output range of [0, 1] as shown in figure 2.7 and its equation is:

$$f(x) = e^{-x^2}$$

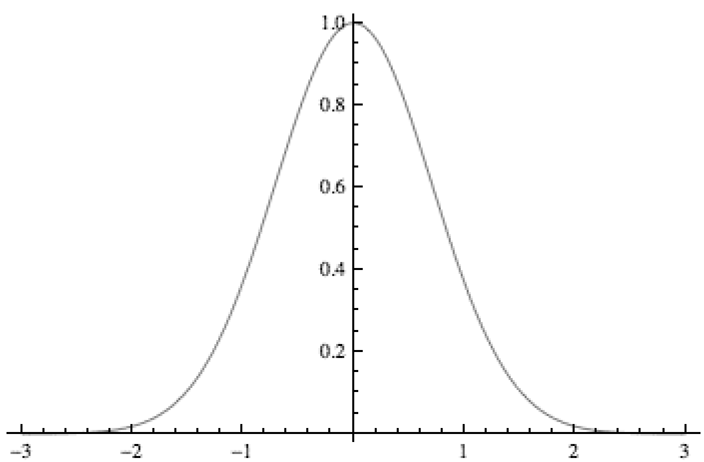


Figure 2.7 Gaussian activation function graphical representation

2.6 Number of iteration during training

Training in a supervised learning for ANN involves having available the desired output for the set of input data to train the neural network. It is from constant training repetition that the ANN will formulate hypothesis about the system being learned. During the training process of the ANN each complete a cycle of all training dataset, with the aim of appropriately changing weight values accordingly to form a training epoch (da Silva IN, 2017 pg. 26)

While training till convergence may seem reasonable, it may also lead to overfitting where the resultant trained neural network model may generalize on previously unseen examples while presenting excellent results on the training data. Small number of training iterations may lead to widely inaccurate neural network model. While cross validation may be used in i.e. getting the error rate of several neural network models variations and choosing the model with the least average errors, one may also decide to use early stopping during training to prevent overtraining hence overfitting. In early stopping one may use a conditional clause where for each cycle of training epoch, the errors i.e. validation errors and training error and are compared to the previous epoch's error so as to stop training when the error stops decreasing or starts increasing. Arbitrary maximum of epoch number for training the ANN can also be set as a safeguard for overtraining. Mahdi Pakdaman (2010) was able to achieve a prediction error of 1.5% on Tehran stock exchange stock price prediction with a ANN which had its maximum epoch set to 1000 during training. (Mahdi Pakdaman et al, 2010)

2.7 Neural network in stock exchange

ANN application in the financial sector and specifically in stock predication has been highly researched since Halbert White 1980 study on study on using ANN timeseries for stock prediction someof which include;

National stock exchange of India - Feedforward MLP neural network has successfully been proven to predict future stock price of companies in the Indian stock exchange LIX15 index to an overall output median normalized error of 0.05995 and a median correct direction of 51.06%. The study also achieved an MSE of 12.3372, RMSE 3.5124 for yes bank company stock prediction. (Mayankkumar B. P., Sunil R. Y. June 2014)

German stock index (Deutscher Aktienindex (DAX)) – Reza, Jamal and Esmaeil were able to achieve a MAPE average of 2.84 on their BNNMAS (bat artificial neural network multi agent system) by predicting Up to 8yrs of share prices at the DAX quarterly which not only involved DAX historical data but also evaluating DAX fundamental data e.g. companies cash flow, return on asset etc. (Hafezi R. et al 2015)

Sao paolo stock exchange (BM&F Bovespa) - ANN has been used to specifically to predict Petrobras stock (PETR4) which is traded on BM&F Bovespa. A prediction accuracy of 5.45% MAPE was achieved by including fundamental analysis of company fundamentals e.g. debt ratio, dividend yield etc. and macroeconomics e.g. Brazil energy commodities index etc. (Fagner A. 2013).

Libyan stock market- Research by Masoud and found that ANNs Accuracy reached to an average ninety one percent prediction rate on daily share price movement on Libyan stock exchange proving their viability in securities exchange predictions. (Masoud 2014)

Palestine Exchange (PEX) - study on Bank of Palestine and Stock of Jerusalem which are listed on the Palestine stock exchange by Safi and White was able to achieve a MAPE of 1.0416% and an RMSE of 0.0781 for long term data, MAPE of 2.2129% and an RMSE of 0.1507 for moderate term data and a MAPE of 2.0524 % and an RMSE of 0.1333 for short term data prediction. (Safi S., White A. 2017)

NSE Kenya- A study by Barack on ANN prediction in the NSE focused on three stocks namely standard bank, Kakuzi and Bamburi was able to achieve an average MAPE average of 1.37% on the three stock. The researcher was able to achieve a low MAPE of 0.77% on standard bank stock. (Barack W.2014)

2.8 Conceptual model

Literature review enabled the researcher to develop a conceptual model as shown in figure2.8 which shows the relationship between the research variables and identified targets in the study.

The main research problem is to come up with the optimal ANN time series stock prediction model by comparing effect of various activation functions hence helping increasing ANN NSE stock price prediction.

The testing of various activation functions will involve acquiring NSE historical data which will be processed to the required dimension and removal of noise. The data will be stored as it will be used in training and testing of all the different ANN models with different activation functions. Before the NSE dataset is fed to the ANN, it is first preprocessed by scaling it to fit with the activation function to be used. The processed data is divided in a ratio 3:1 between training dataset and testing dataset in favour of training dataset. Actual training of the neural network occurs using the training dataset with the errors recorded. Testing of the neural network with the testing dataset with the aim of identifying the accuracy of ANN model. With the completion of cycle the process is repeated with different activation functions while the previously tested ANN is saved and its error on accuracy being recorded for comparison with other ANN models.

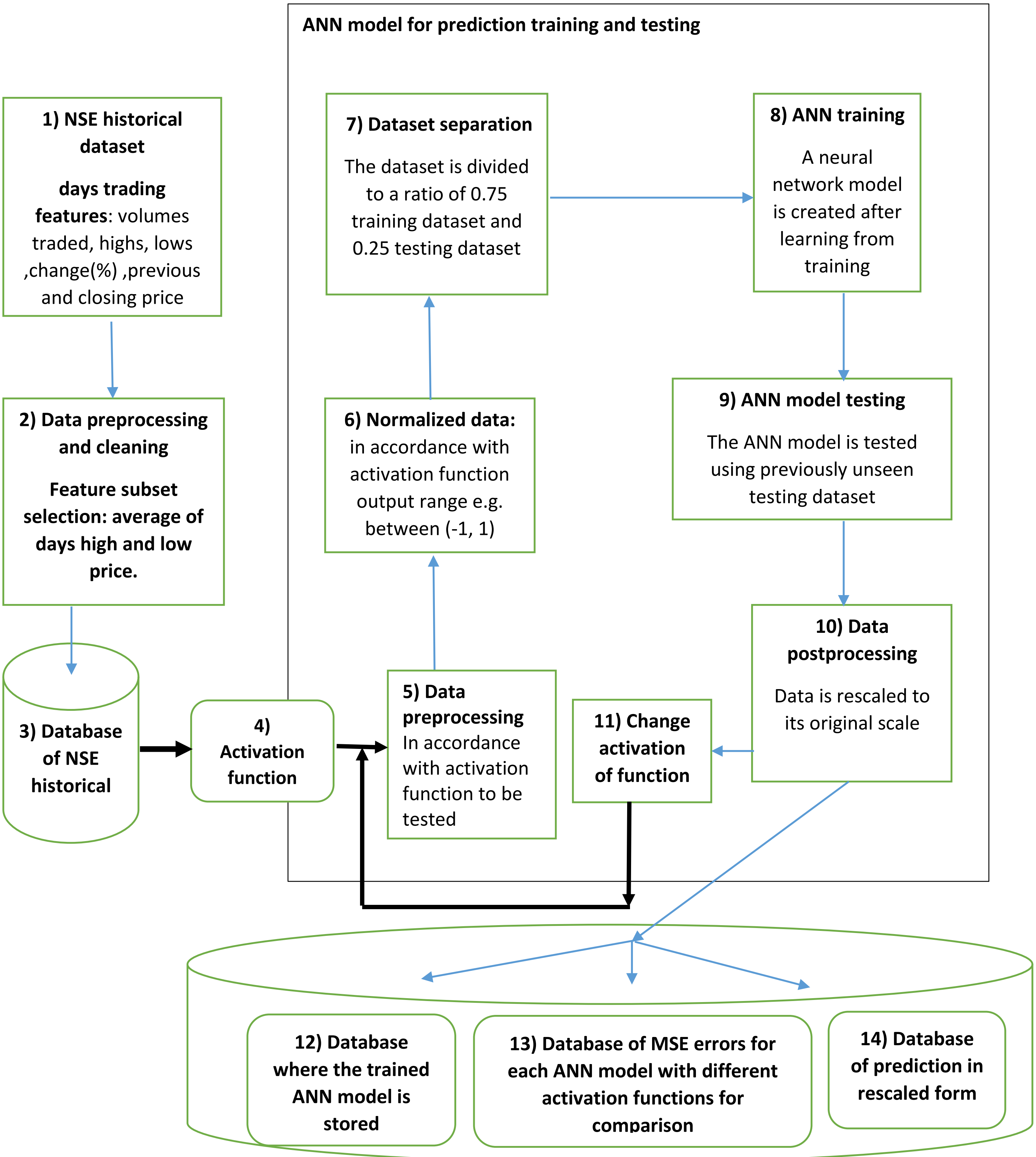


Figure 2.8 Conceptual model

3.0 Research methodology.

Since the proposed study will involve application development, the system development methodology to be used study is Incremental with Iterative Prototyping because it allows segmentation of the project for simpler implementation process and availability of an option to make changes during development, it will involve two phases as illustrated in figure3.1 namely:

-Data acquisition and preparation phase- where data will be acquired and preprocessed for use.

-ANN coding and comparison phase- where the neural networks will be designed, coded, tested and compared.

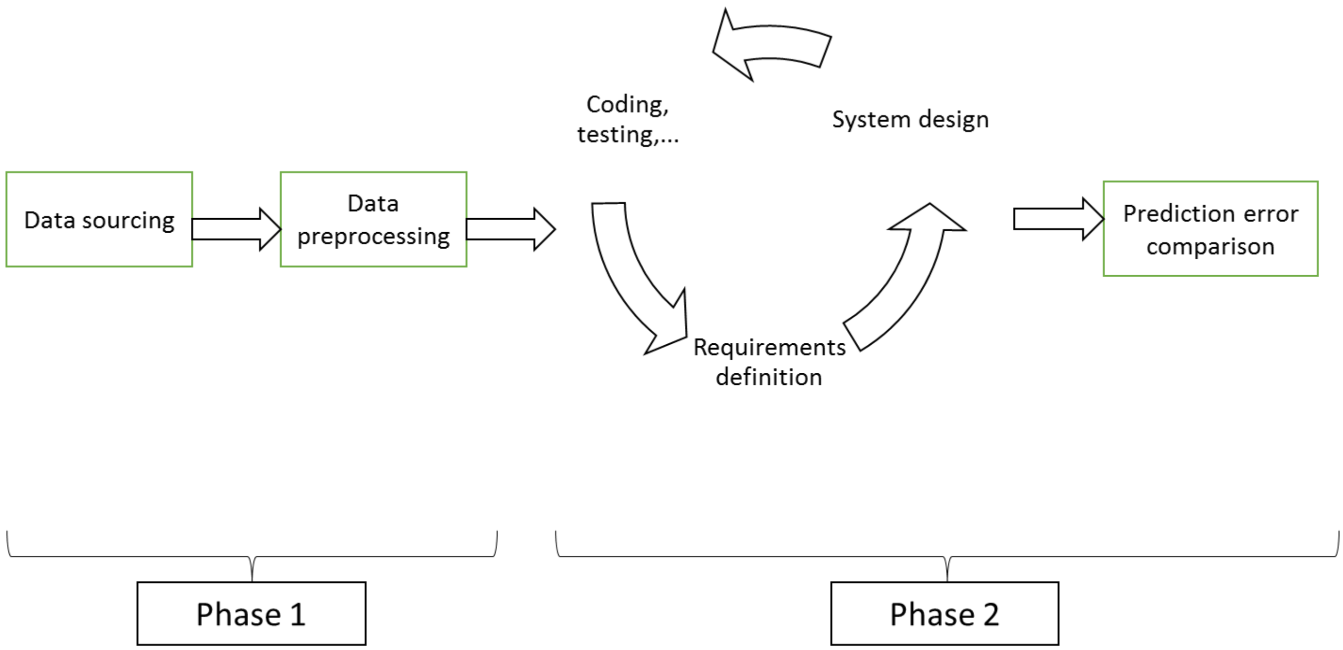


Figure 3.1 research methodology

3.1 Phase one

3.1.1 Sourcing of data

The proposed study will use NSE historical daily share data for training and testing the neural network. NSE data can be acquired directly from NSE or from authorized data vendors at a fee. (NSE, 2017)

This research will however use historical data from Synergy Systems Ltd who are NSE authorized data vendors through live.mystocks.co.ke/ due to ease of access and payment. Data will be in spreadsheets and embedded in html files.

3.1.2 Preprocessing of data

The NSE historical data acquired will need to be processed before any work can be done using the data. Some features in the NSE data which are not useful to training and testing of the neural network will be removed leaving only the date and the average share price of the day to be used during learning.

Python scripts will be used to perform data preprocessing through numpy and StringIO libraries to generate arrays for processing and saving data in csv(comma-separated value) files. SQLite and csv files will be used as the databases. These tools were selected due to their open-source nature, ease of use, and their familiarity to the researcher

3.2 Phase two

3.2.1 Requirements definition, designing, Coding and testing

Requirements for each activation function to be tested vary thus the need for the requirements to be reviewed for each activation function before designing and testing. Coding and testing will be done through python using:

-pybrain library for neural network designing, creation and implementation.

-numpy, StringIO and csv libraries for data extraction, transforming and loading.

-math library for MSE and RMSE error calculation.

-matplotlib library to generate plots and charts for graphical representation.

Text files will be used to temporarily store data SQLite will be used to permanently store data.

Xml files will be used trained ANN structure and data. Python and SQLite were selected due to their open-source nature, ease of use, and their familiarity to the researcher.

Prediction error and calculation and comparison will be done through mean squared error and the root mean square error.

The MSE mathematical equation is:

$$MSE = \frac{1}{o}\sum_{i=1}^{o}(actval_i - \widehat{preval}_i)^2$$

$actval_i$ = Actual value

$\widehat{preval}_i$ = Predicted value

o = Number of observations

The RMSE mathematical equation is:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{o}(\widehat{preval}_i - actval_i)^2}{o}}$$

$actval_i$ = Actual value

$\widehat{preval}_i$ = Predicted value

o = Number of observations

Both MSE and RMSE will be calculated from the prediction of the ANN through python script math library and plotted for better explanation through matplotlib libraries.

3.3 Data acquisition and preparation

The stocks chosen and used by the study were all constituents of the NSE 20 share index each representing a different section of the NSE companies as shown in table 3.1 below

Stock name	Ticker symbol	Sector
equity banking	EQTY	Banking
Sasini	SASN	Agriculture
Kengen	KEGN	Energy
East African Breweries Ltd	EABL	Manufacturing
Nation Media Group	NMG	Services

Table 3.1 stock to be used.

Data acquired from mystocks website was in HTML format and it was extracted and updated to an SQLITE3 database by the author's python script named nsescrapdata.py

The nsescrapdata.py code parses the HTML table containing the data where it was then looped through parsed data with instructions to insert it into the database.

Ps : the ">" character represents a command prompt terminal in windows cmd shell or PowerShell.

```
> Python nsescrapdata.py
```

SQLite manager was used for a better direct view and manipulation of the dataset through its query editor which is compatible with SQLite3 databases as shown in the figure 3.2.

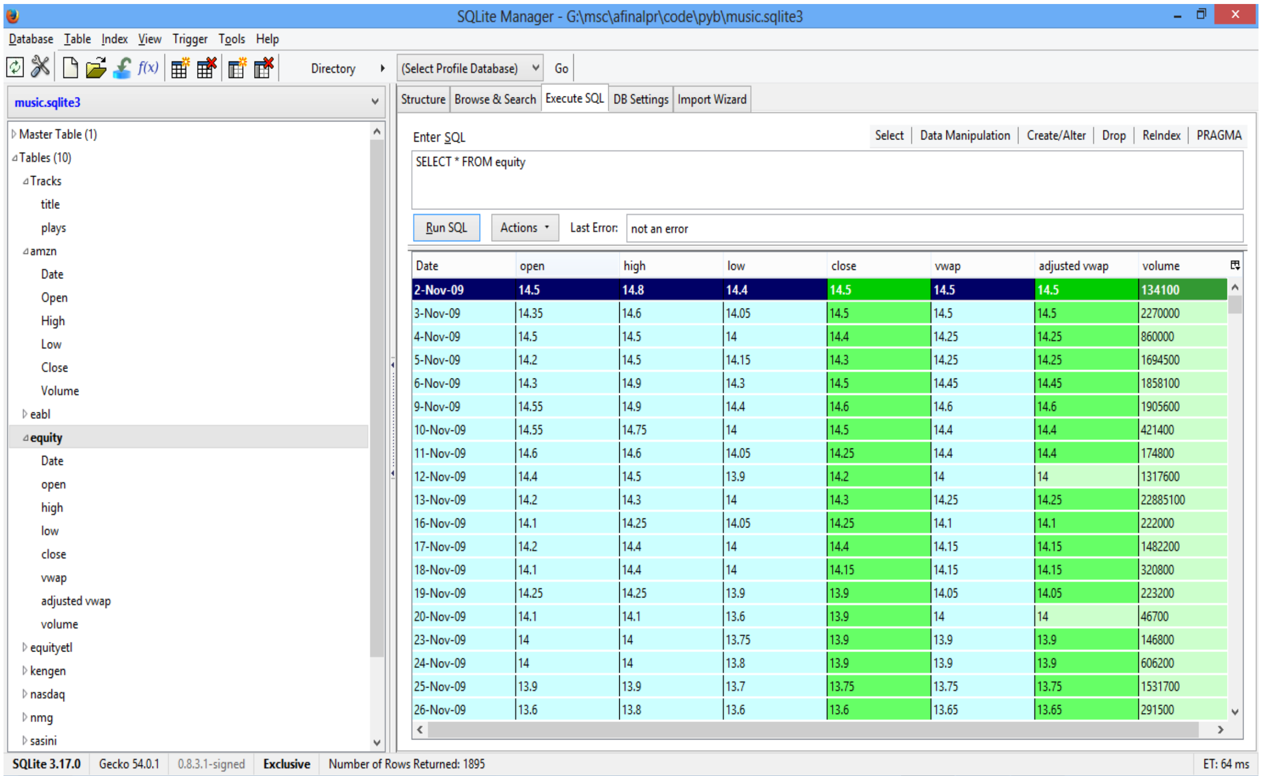


Figure 3.2 SQLite manager displaying raw equity stock data

The NSE raw data extracted came with attributes date, open, high, close, vwap, adjusted vwap and volume as shown in figure 3.2 previously.

The required data subsets of close attribute were fetched from the database using the dataetl.py python script which dumped the data to a text file called pybraindata.txt from where data scaling and rescaling occurred.

> Python dataetl.py

Since the activation function determines the range of scaling, the data normalization code was combined with the ANN training and testing code.

The code for data rescaling to original or expected figures was combined with the ANN code for convenience.

ANN activations functions tested

The activation functions that were tested were a total of five with twenty possible combinations tested which are.

Code	Name
S	Sigmoid
L	Linier
SF	Softmax
G	Gaussian
T	Hyperbolic Tangent

Legend

Activation function	HYBRID/homogenous
S-S-S-S	Plain
T-T-T-T	plain
SF-SF-SF-SF	plain
L-L-L-L	plain
G-G-G-G	plain
S-T-T-S	hybrid
S-SF-SF-S	Hybrid
S-G-G-S	Hybrid
S-L-L-S	Hybrid
T-S-S-T	hybrid
T-G-G-T	Hybrid
T-SF-SF-T	Hybrid
T-L-L-T	Hybrid
SF-S-S-SF	Hybrid
SF-T-T-SF	Hybrid
SF-L-L-SF	Hybrid
SF-G-G-SF	Hybrid
L-S-S-L	Hybrid
L-T-T-L	Hybrid
L-G-G-L	Hybrid
L-SF-SF-L	Hybrid
G-S-S-G	Hybrid
G-T-T-G	Hybrid
G-L-L-G	Hybrid
G-SF-SF-G	Hybrid

Table 3.2 Activation functions to be tested

Network saving

The trained networks were saved for later use through use of the network writer python library. The code to achieve saving the was incorporated into the training code to save the network but was also callable from different functions and programs

Network training and testing

Training of the network was achieved by calling a function with optional variables passed to them i.e. inside the python code in a file named caller.py calling the function, was done by importing the code with the training and testing capability while passing optional two variables for the scaling the data e.g. [0,1], the activation functions of input, first hidden, second hidden & output layers all represented by integers between 1 and 4 which are placeholder for activation functions eg 1 =sigmoid activation function.

> Python caller.py 0 1 3

User interface

The user interface and systems logic was created using the Django framework. The front end interface was coded in html styled using css and some of the functionality using JavaScript "chart JS" which includes generation of charts.

The backend was handled by python where stored neural networks were used to provide a prediction based on preprocessed data provided. By clicking on any stock on displayed on the table, on would be able to view an interactive chart of the select stock plus the prediction.

4.0 results and discussion

4.1 Activation functions results

The table 4.1 below shows the different MSE achieved by different activation functions

Network code	epoch	MSE	RMSE	Network type
T-SF-SF-T	92	0.000101	0.010027	Hybrid
T-T-T-T	1996	0.000102	0.010119	homogenous
L-L-L-L	25	0.00024	0.015489	homogenous
S-T-T-S	1791	0.000343	0.018534	hybrid
S-S-S-S	1983	0.000476	0.021828	Homogenous
L-T-T-L	1509	0.001383	0.037188	Hybrid
T-L-L-T	15	0.003411	0.058402	Hybrid
T-G-G-T	1465	0.00667	0.08167	Hybrid
L-SF-SF-L	0	0.011484	0.107163	Hybrid
L-S-S-L	21	0.0271	0.164621	Hybrid
T-S-S-T	14	0.028584	0.169068	hybrid
G-L-L-G	762	0.052549	0.229236	Hybrid
S-SF-SF-S	4	0.059002	0.242903	Hybrid
S-G-G-S	1385	0.06324	0.251476	Hybrid
G-SF-SF-G	1541	0.063367	0.251727	Hybrid
S-L-L-S	13	0.064416	0.253803	Hybrid
G-S-S-G	1256	0.06952	0.263666	Hybrid
L-G-G-L	4	0.098509	0.313862	Hybrid
G-T-T-G	889	0.224902	0.474238	Hybrid
SF-T-T-SF	all	0.363179	0.602644	Hybrid
SF-S-S-SF	all	0.412356	0.64215	Hybrid
SF-G-G-SF	all	0.427005	0.653457	Hybrid
G-G-G-G	4	4.000409	2.000102	Homogenous
SF-SF-SF-SF	Nan	NA	#VALUE!	Homogenous
SF-L-L-SF	Nan	nan	#VALUE!	Hybrid

Table 4.1 Activation functions test results

ANN with a homogenous activation functions of L-L-L-L “linier activation function” achieved the lowest MSE and RMSE making it the better alternative in stock price prediction as figures 4.1, 4.2

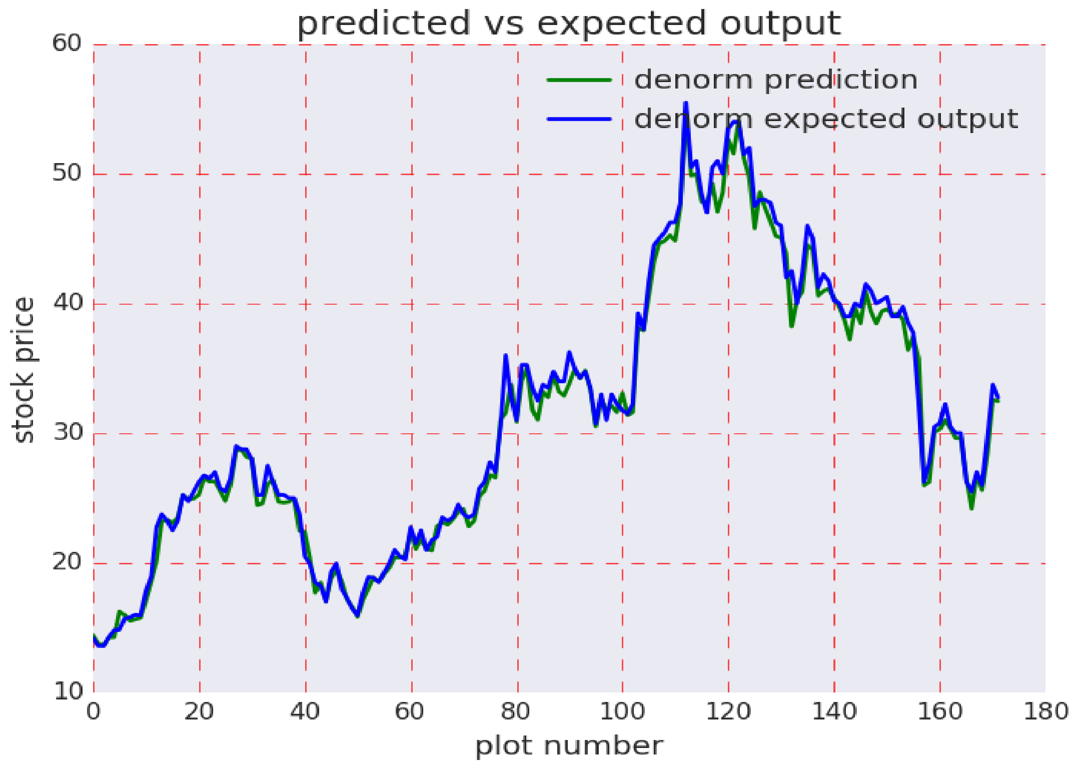


Figure 4.1 Linier activation function on unseen stock data

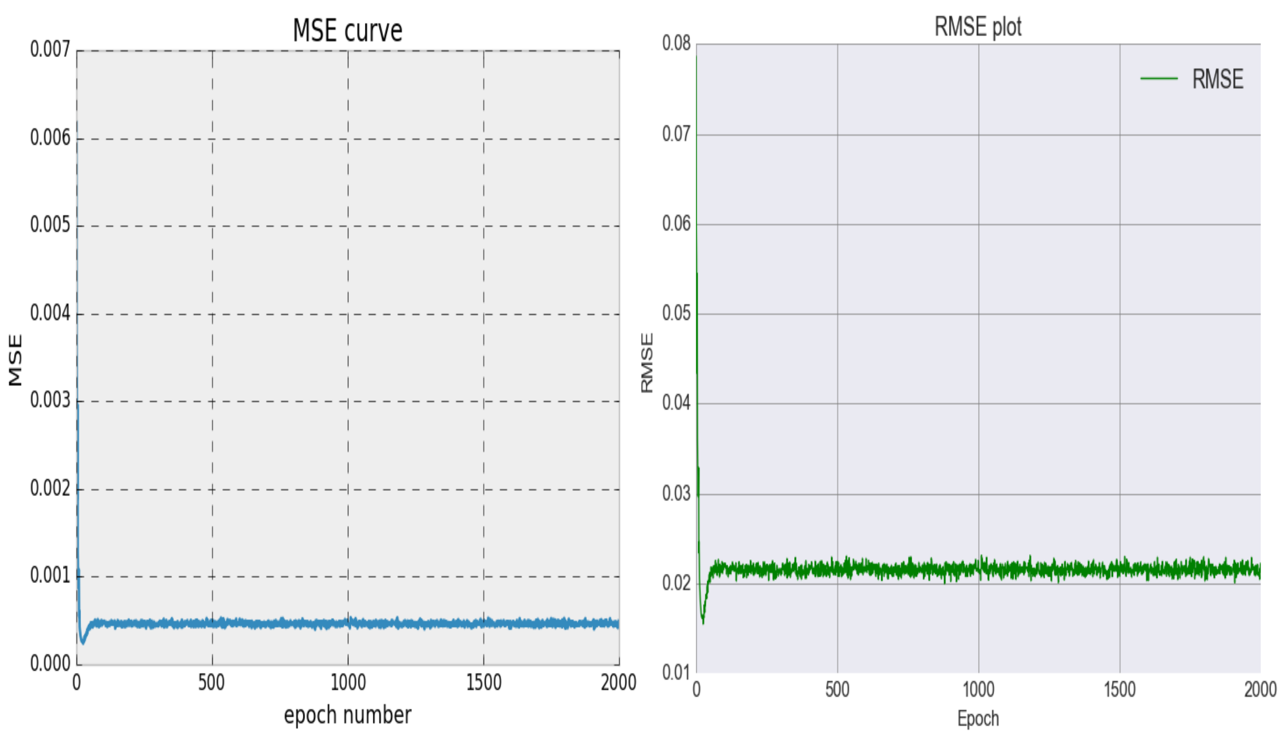


Figure 4.2 linier activation function MSE training results

A hybrid L-T-T-L “linier & hyperbolic tangent” type of ANN had the second lowest MSE [0.00138293] and RMSE [0.037187829] by the 1509 epoch of training as illustrated in figures 4.3,4.4.

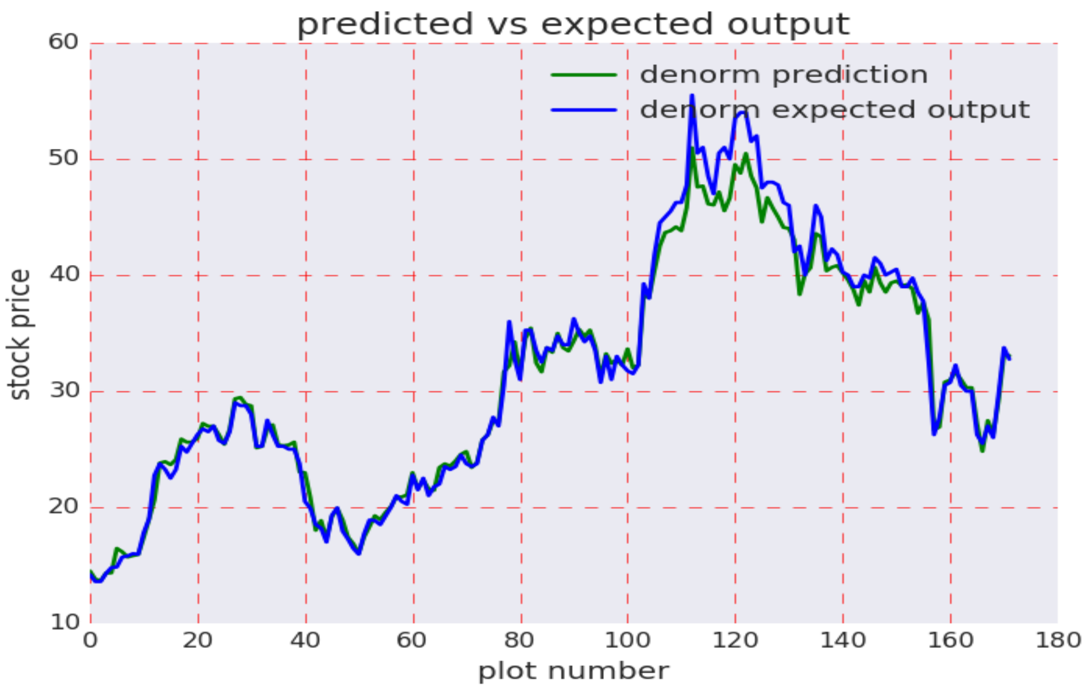


Figure 4.3 L-T-T-L prediction on unseen data

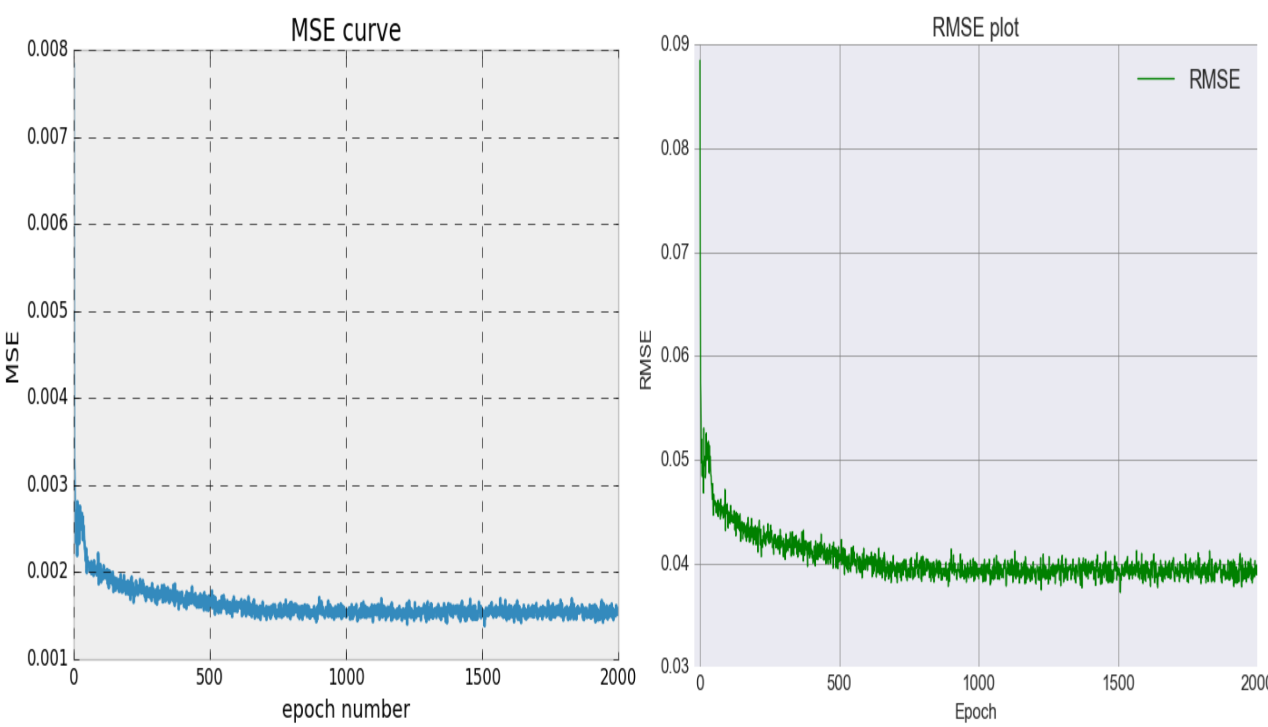


Figure 4.4 L-T-T-L training MSE curve

A hybrid T-L-L-T "hyperbolic tangent & linear " type of ANN had the third lowest MSE [0.00341081848081] and RMSE [0.0584022129787] by the 15 epoch of training as illustrated in figures 4.5,4.6.

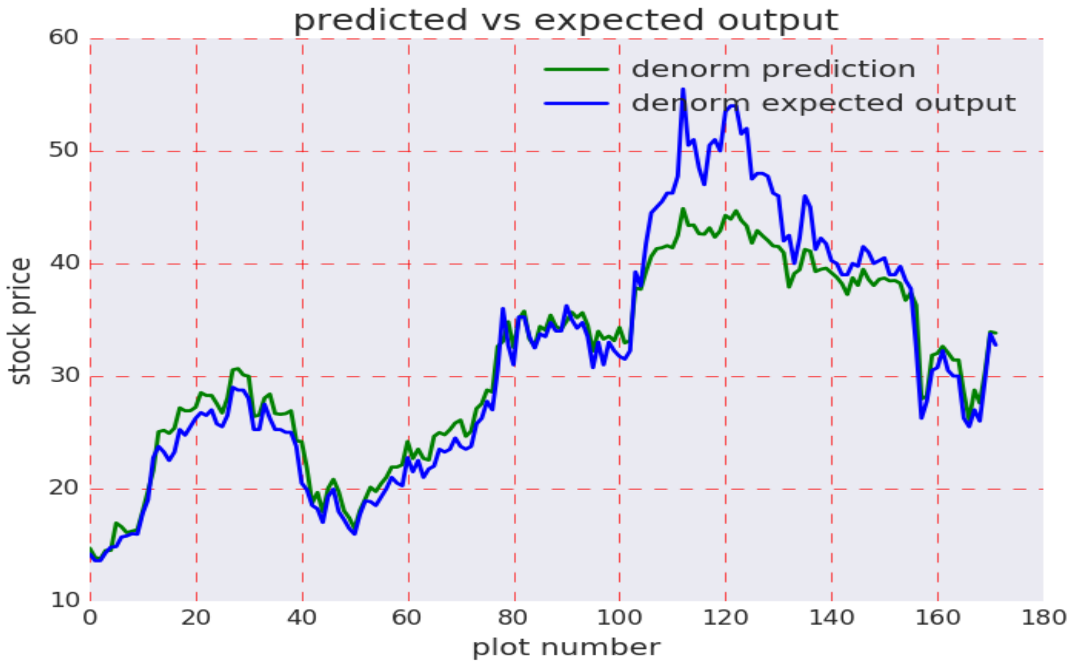


Figure 4.5 T-L-L-T prediction on unseen data

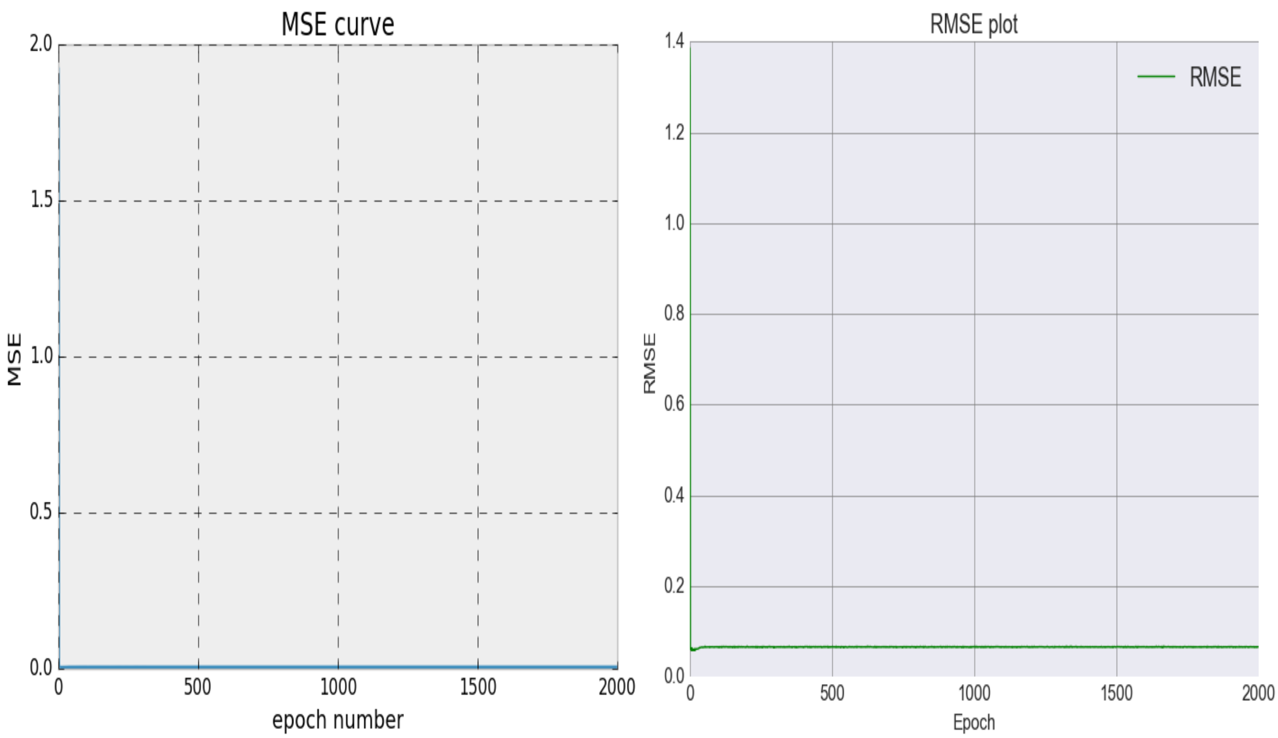


Figure 4.6 T-L-L-T training MSE curve

A homogenous T-T-T-T ”hyperbolic tangent” of ANN had a low MSE of [0.000102395169704] and an RMSE of [0.0101190498419] at the 1996 epoch as shown in figure 4.7 below

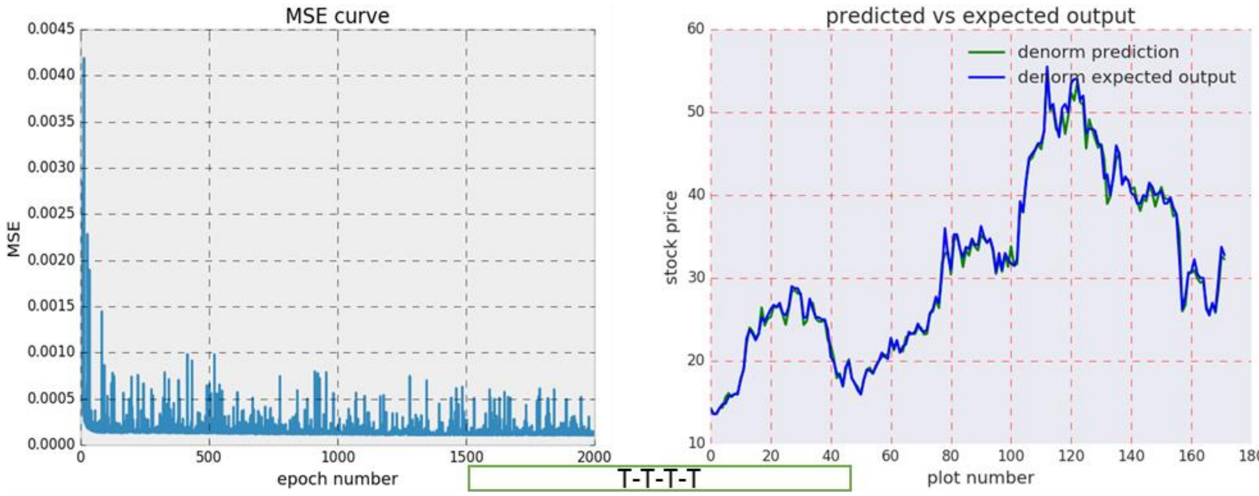


Figure 4.7 T-T-T-T training MSE and network test on unseen stock data

A homogenous S-S-S-S "sigmoid" type of ANN had a very low MSE [0.000476466] and RMSE [0.02182809] by the 1983 epoch of training as illustrated in figures 4.8

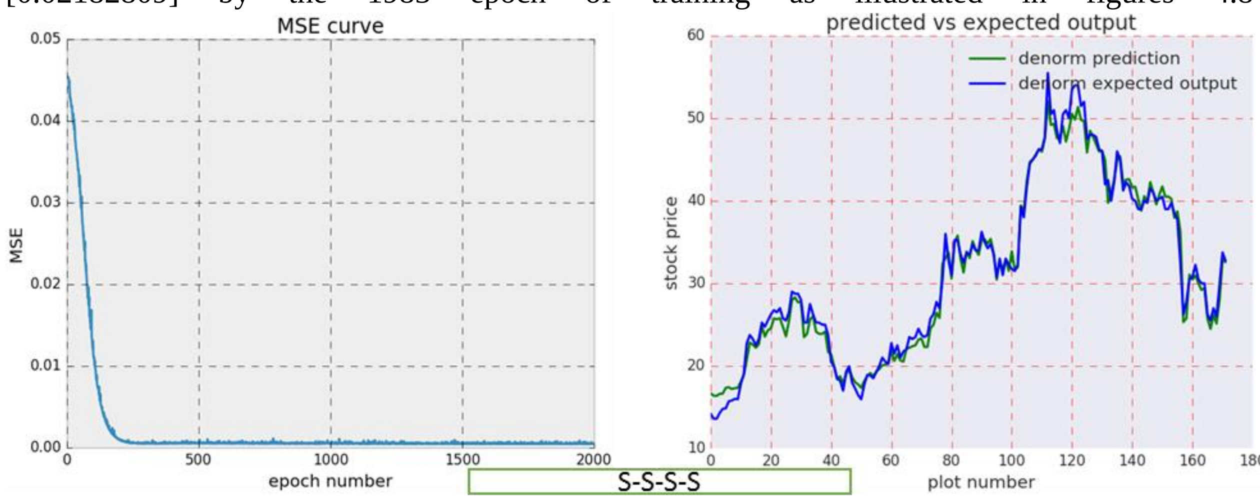


Figure 4.8 S-S-S-S training MSE and network test on unseen stock data

A hybrid S-T-T-S "sigmoid & hyperbolic tangent" type of ANN had a low MSE [0.000343495] and RMSE [0.0185336229225] at the 1791 epoch of training as illustrated in figures 4.9

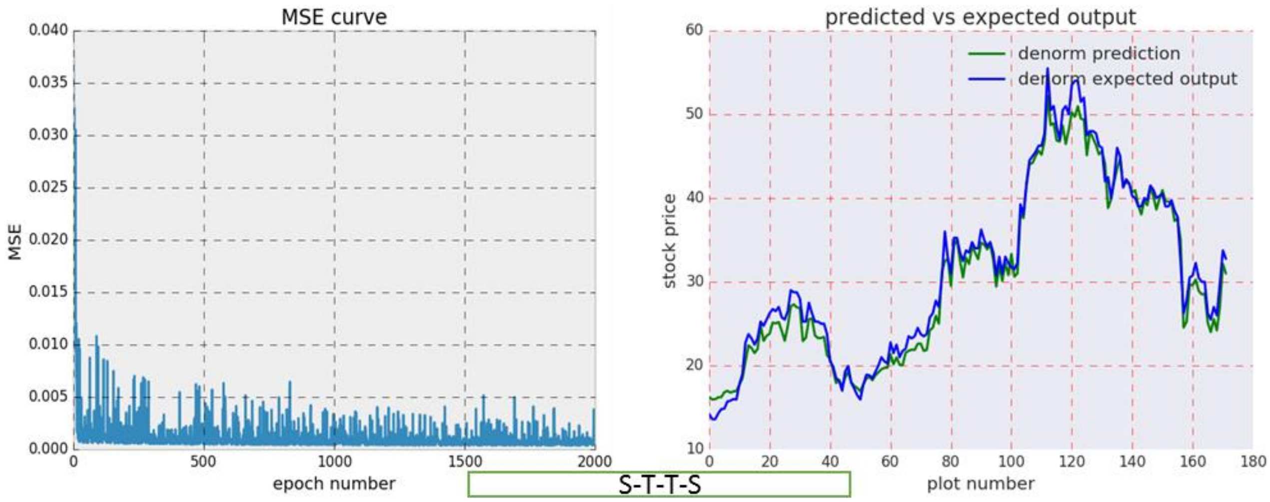


Figure 4.9 S-T-T-S training MSE and network test on unseen stock data

Softmax activation function did not perform well either in use in a homogenous or hybrid network with the exception of hyperbolic tangent in input/output layers and softmax function in the hidden layers which only worked with minimal number of training epochs eg 100 iterations. T-SF-SF-T network achieved a good result at the 92nd epoch of MSE [0.000100531] RMSE [0.010026535] as shown in figure 4.10 unlike other networks with softmax activation functions illustrated in figures 4.13,4.11 & 4.12

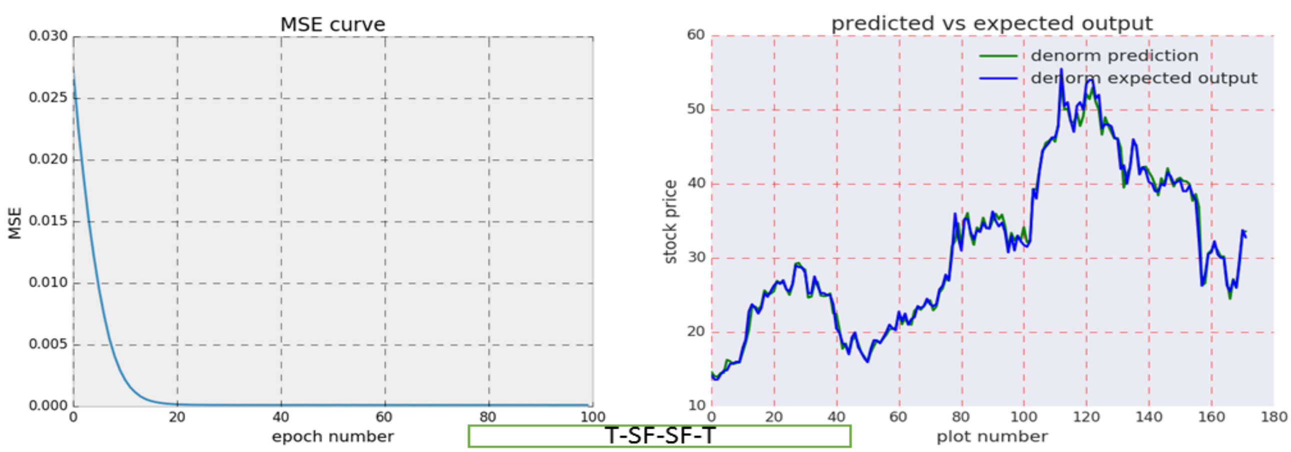


Figure 4.10 T-SF-SF-T training MSE curve and network test on unseen stock data

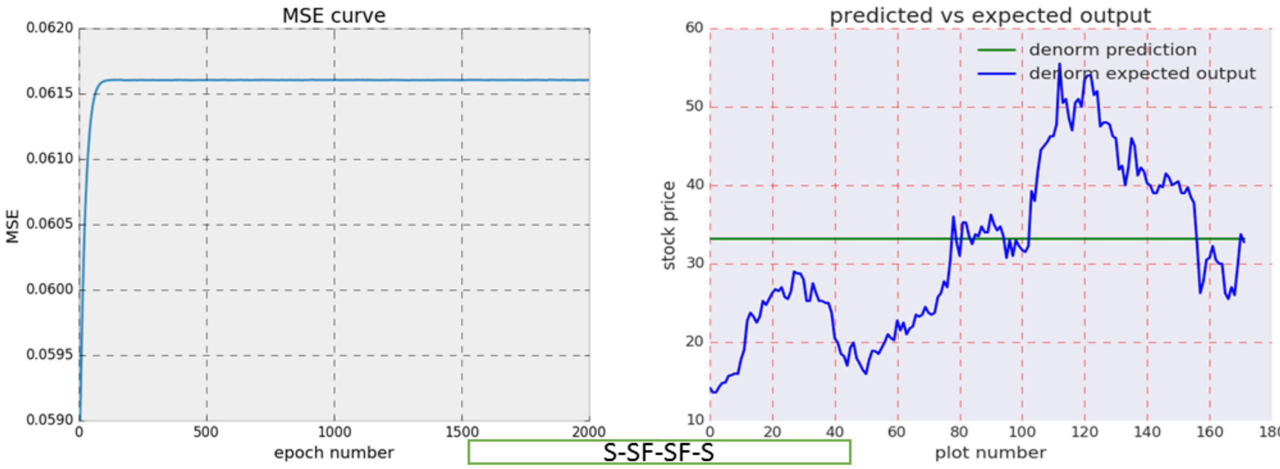


Figure 4.11 S-SF-SF-S training MSE curve and network test on unseen stock data

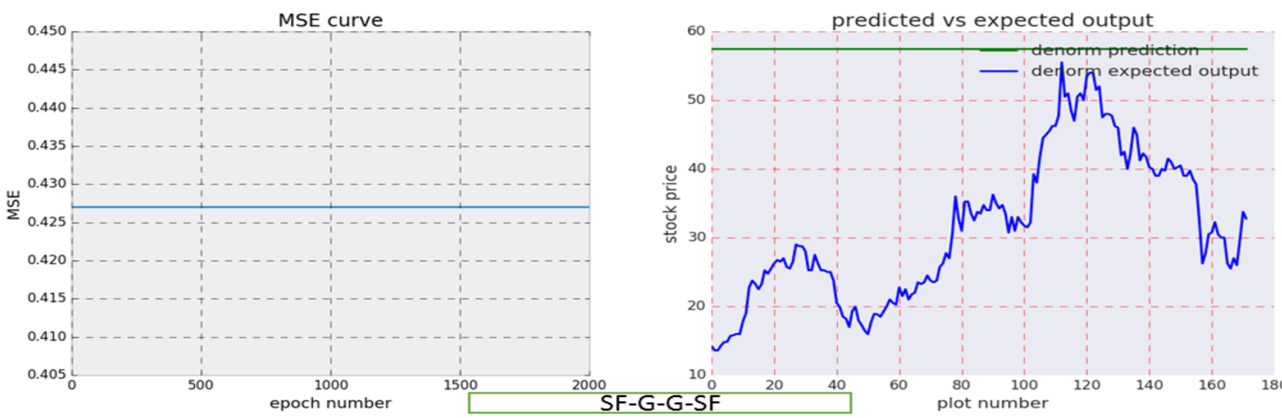


Figure 4.12 SF-G-G-SF training MSE curve and network test on unseen stock data

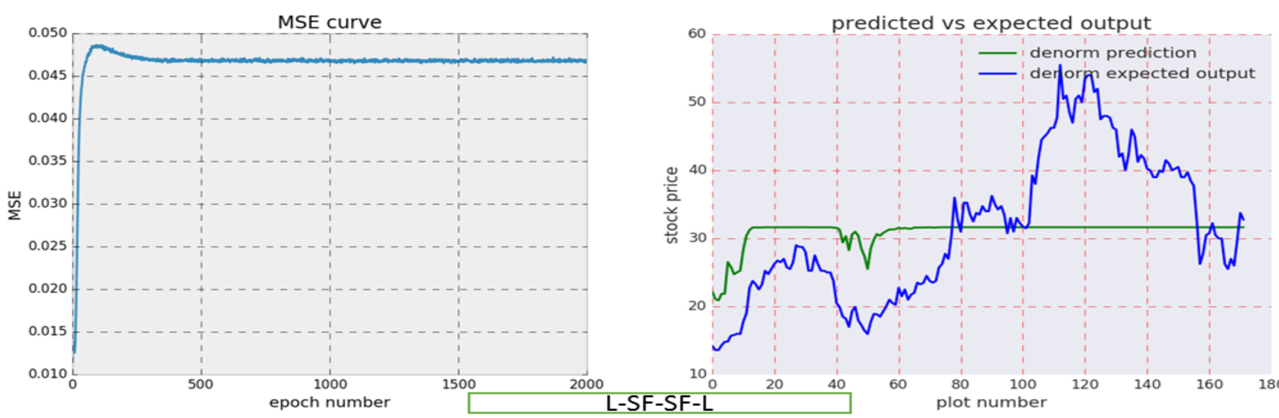


Figure 4.13 L-SF-SF-L training MSE curve and network test on unseen stock data

Gaussian activation function performed poorly in both homogenous and hybrid networks as shown in the figures 4.14, 4.15, 4.16, 4.17 & 4.18 below showing the training and testing performance.

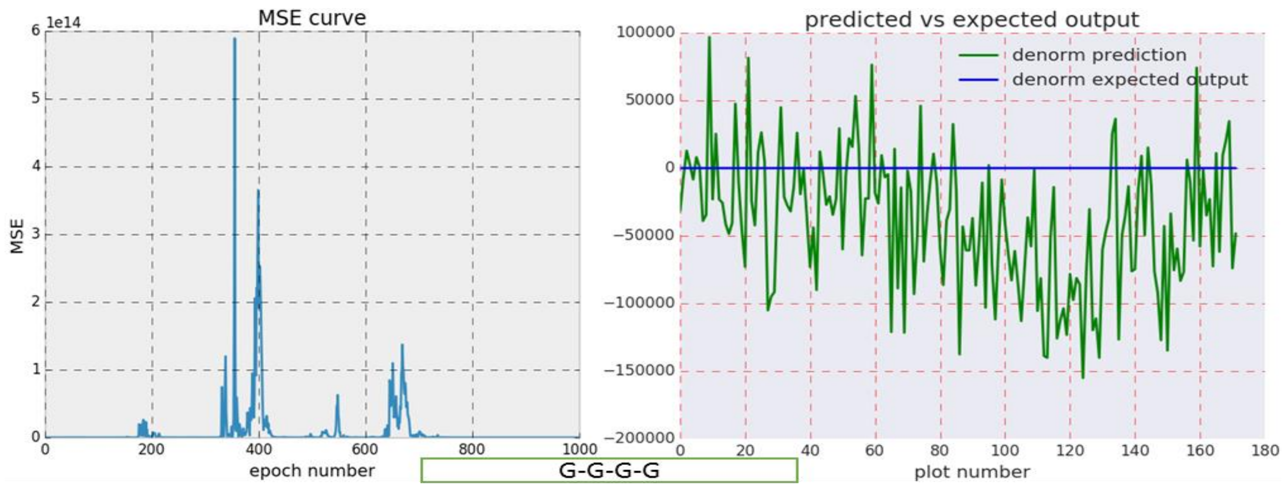


Figure 4.14 G-G-G-G training MSE curve and network test on unseen stock data.

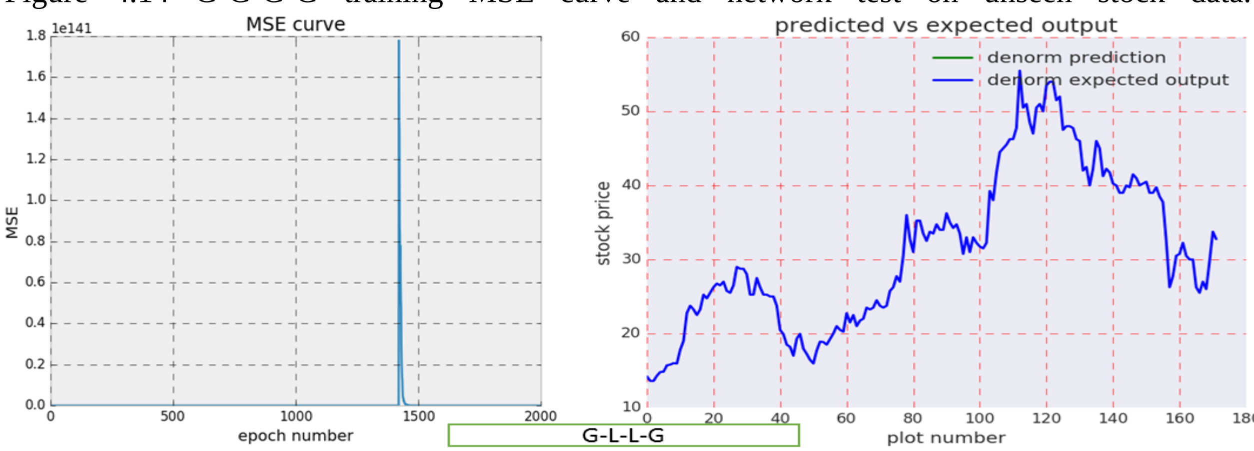


Figure 4.15 G-L-L-G training MSE curve and network test on unseen stock data.

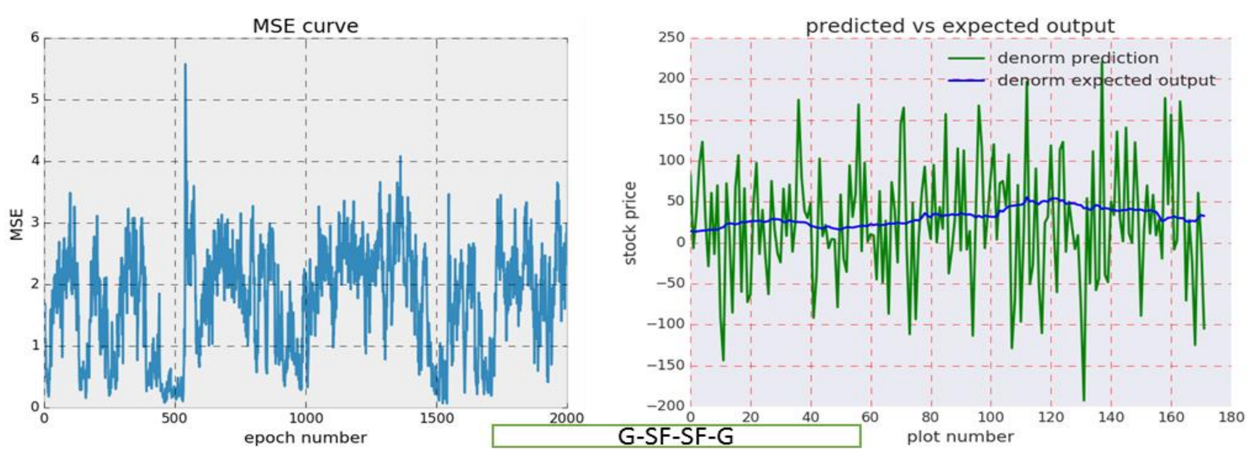


Figure 4.16 G-SF-SF-G training MSE curve and network test on unseen stock data.

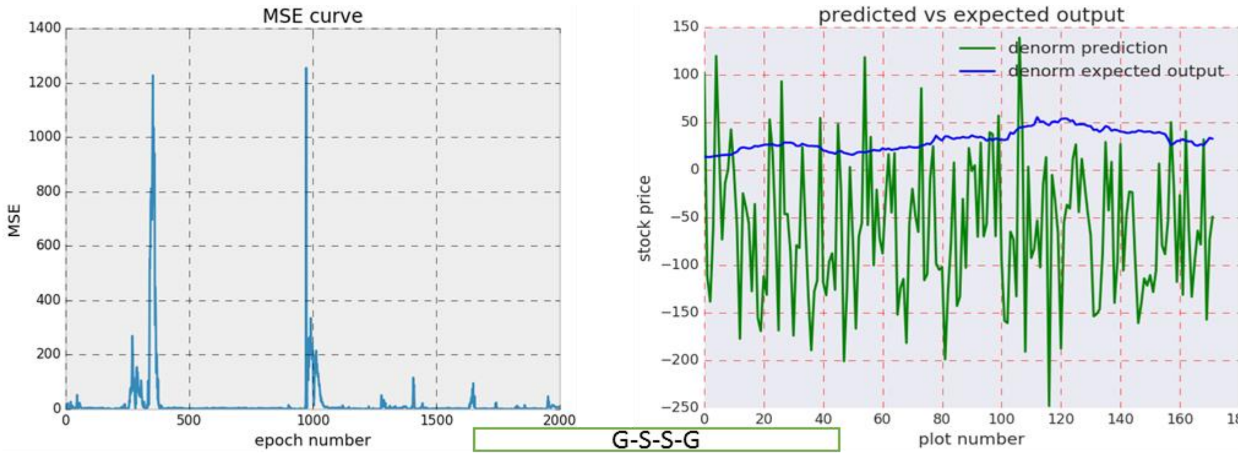


Figure 4.17 G-S-S-G training MSE curve and network test on unseen stock data.

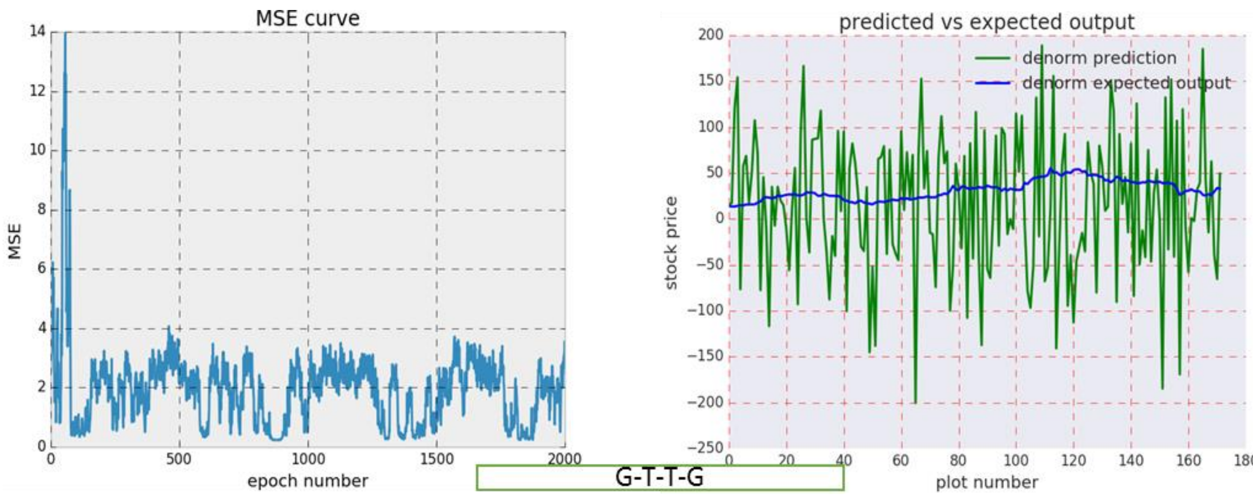


Figure 4.18 G-T-T-G training MSE curve and network test on unseen stock data.

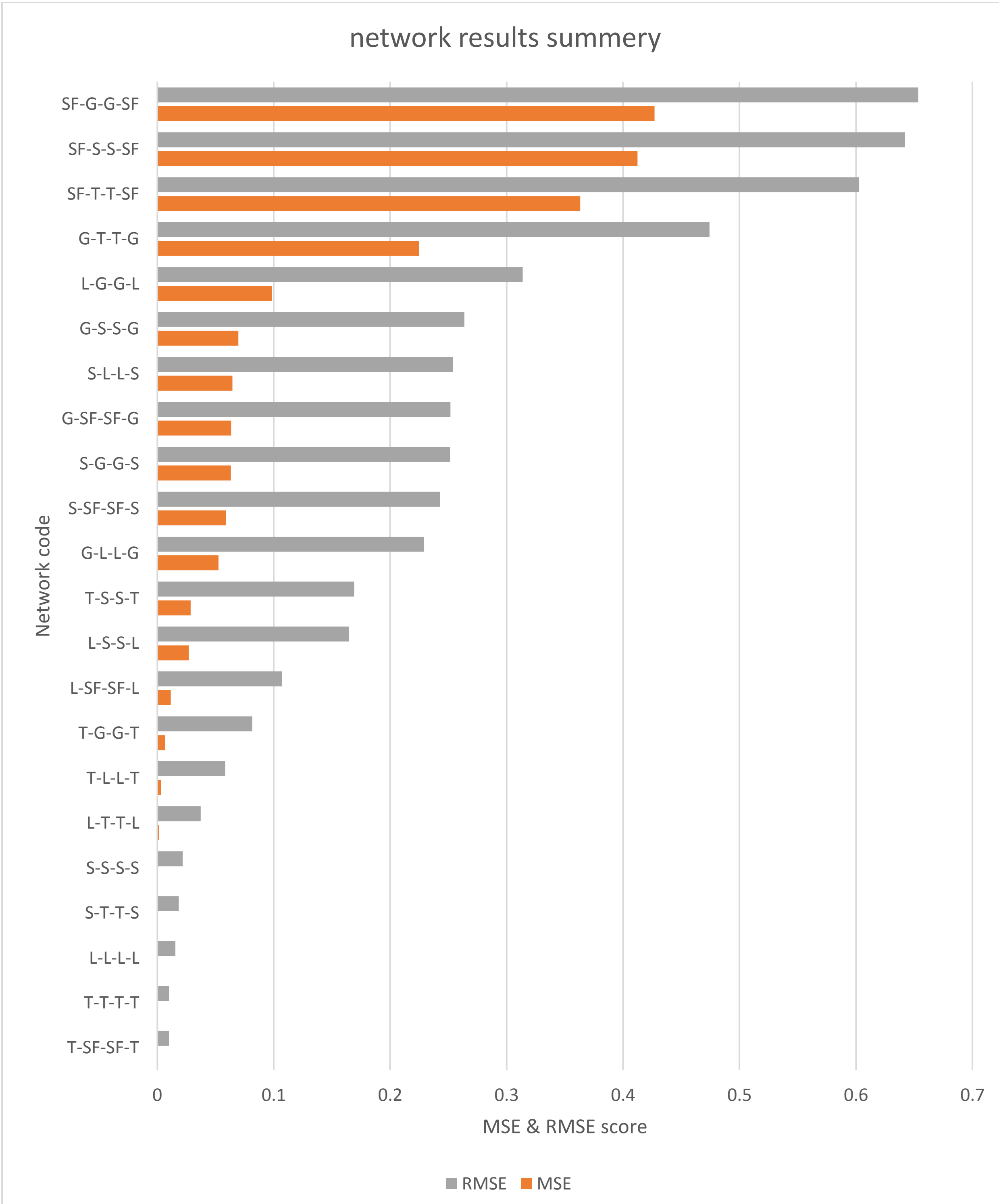


Figure 4.19 summery of MSE RMSE network performance.

Ps. shorter bar is more preferred.

Test result on selected stocks of the top five activation functions combinations

equity stocks			
network code	Epoch	MSE	RMSE
T-SF-SF-T	92	0.000100531	0.010026535
T-T-T-T	1996	0.000102395	0.01011905
L-L-L-L	25	0.00023992	0.015489438
S-T-T-S	1791	0.000343495	0.018533623
S-S-S-S	1983	0.000476466	0.02182809
EABL stocks			
network code	Epoch	MSE	RMSE
T-SF-SF-T	93	0.000241672	0.015545803
T-T-T-T	131	0.000311157	0.017639641
L-L-L-L	95	0.000379957	0.019492493
S-T-T-S	1962	0.000414071	0.02034874
S-S-S-S	1972	0.000590453	0.024299236
Kengen stocks			
network code	Epoch	MSE	RMSE
T-SF-SF-T	91	0.000147768	0.012155977
T-T-T-T	1876	0.000268425	0.016383692
L-L-L-L	50	0.000552808	0.023511877
S-S-S-S	1994	0.000683386	0.026141644
S-T-T-S	1929	0.000799378	0.02827327
NMG stocks			
network code	epoch	MSE	RMSE
T-T-T-T	906	0.0000202742	0.00450269
L-L-L-L	52	0.000100844	0.010042109
T-SF-SF-T	90	0.00011731	0.01083097
S-T-T-S	1999	0.000358273	0.018928106
S-S-S-S	1729	0.000763593	0.027633187
Sasini stocks			
network code	epoch	MSE	RMSE
T-SF-SF-T	81	0.000258714	0.016084598
T-T-T-T	827	0.000499748	0.022355053
L-L-L-L	321	0.000636524	0.025229431
S-T-T-S	1882	0.00118233	0.034385023
S-S-S-S	1992	0.001233084	0.035115294

Table 4.2 test results summery

4.2 Web application

The web application was developed as a three tiered structure as shown in figure 4.20

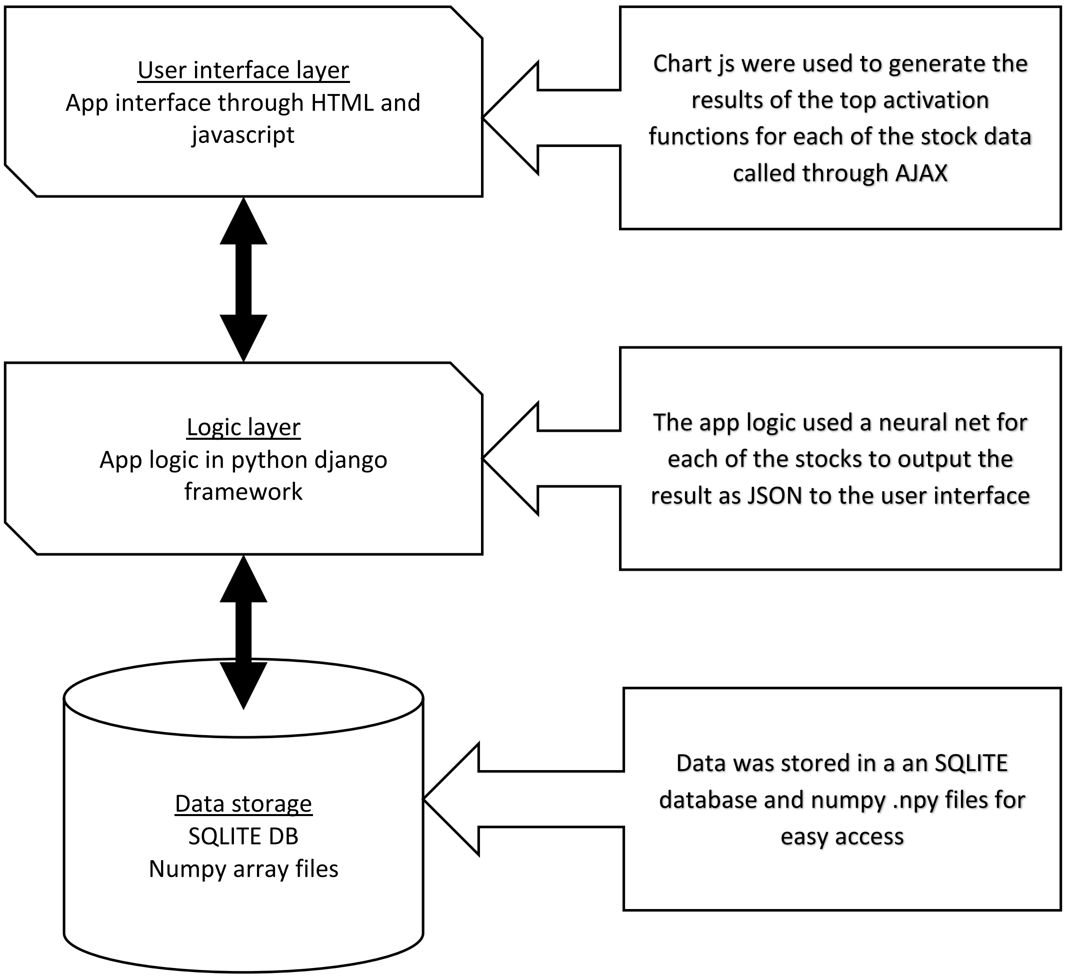


Figure 4.2.1 web application model

4.2.1 Data storage

Data scrapped for the select stocks was insert into the SQLITE database tables with each of the stocks having its own table named after the stock e.g kengen table etc. SQLITE manager Mozilla firefox plugin was particulary useful in extract transform, load and view operations on the database.

4.2.2 Logic layer

The logic layer was based on the python's Django model view controller "MVC" system with the model as the representation of the project data, view which is an interface for the user and controller which controls the flow of data or information between the model and view.

The application had two web APIs with JavaScript Object Notation "JSON". The first web API was used to populate a HTML table by looping through a JSON data dictionary as shown in figure 4.21 below.

stock	date	open	high	low	close	vwap	volume	stockfetchid
eabl	17-Jul-17	250	251	250	251	250	107,700	VIEW CHART
equity	28-Apr-17	33.75	34	32.75	33.75	33	8,614,600	VIEW CHART
kengen	17-Jul-17	7.9	8	7.8	7.85	7.85	358,000	VIEW CHART
nmg	17-Jul-17	105	111	105	110	107	76,200	VIEW CHART
sasini	17-Jul-17	25.5	25.5	25	25	25	10,500	VIEW CHART

Figure 4.2.2 stocks HTML table

The second web API was used in generating an interactive chart as shown in figure 4.22 below. The chart could be cleared and loaded with new stock data without refreshing the webpage by clicking on the view chart button in the HTML table.

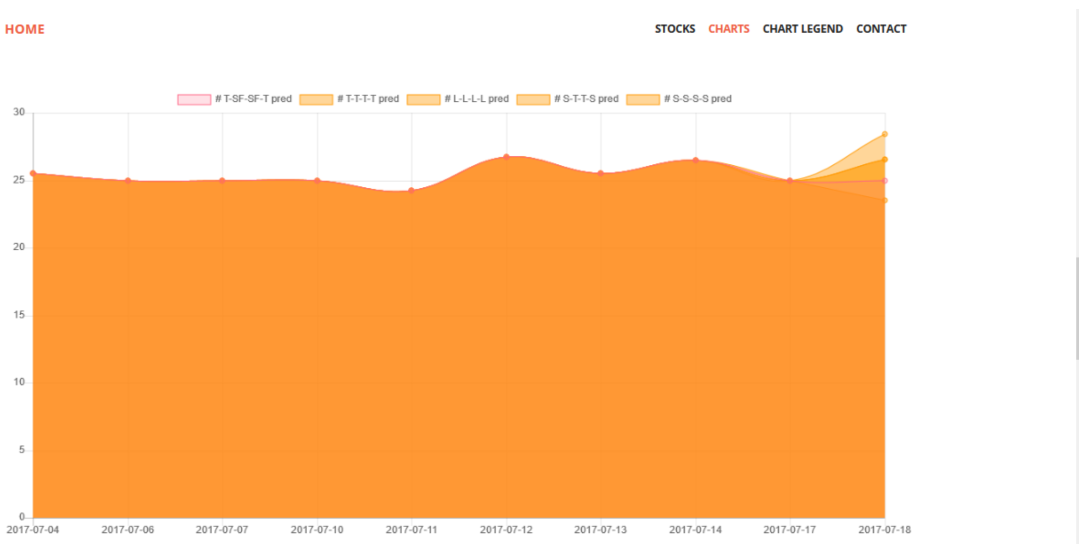


Figure 4.2.3 chart with prediction

The chart used the JSON response to generate the chart based on chart.js and jquery. The JSON data included prediction from the top five activation functions combinations in accordance to the study's results.

4.3 results summery

Test done on the rest of the selected stocks were very similar to results from Equity stocks with a few exceptions which are:

- For the NMG stocks unlike the other stock where a network of T-SF-SF-T “hyperbolic tangent – softmax – softmax - hyperbolic tangent” achieved the lowest score a network of T-T-T-T “hyperbolic tangent” on the 906^{th} epoch achieved an MSE of [0.0000202742] and RMSE of [0.00450269] making it the best performer for NMG stock prediction. Homogenous linear “L-L-L-L” network was the second best performer followed by a hybrid hyperbolic tangent, softmax network.
- For the Kengen stock prediction a homogenous sigmoid “S-S-S-S” network outperformed a hybrid sigmoid/ hyperbolic tangent “S-T-T-S” attaining an MSE of 0.000683386 & RMSE of 0.026141644 on the 1994^{th} epoch for S-S-S-S and MSE of 0.000799378 & RMSE of [0.02827327] on 1929^{th} epoch for “S-T-T-S” achieving 4^{th} and 5^{th} place respectively.

5.0 conclusion and recommendations

With the main objective of the study being identifying the best activation functions that should be used for NSE stock ANN prediction by testing their prediction accuracy level on five selected companies seven year data sets. A secondary objective was developing a web application through trained ANN can used and the results viewed interactively. The research objectives of this study were achieved.

ANN implementation of pybrain framework library was effective and efficient in prediction testing on NSE data by providing various customizable parameters to users to do various modifications to the neural network before training and saving it.

The testing done during the course of the study revealed a hybrid network of T-SF-SF-T "hyperbolic tangent – softmax – softmax - hyperbolic tangent" to be very good at stock prediction as long as during training ,less than roughly 100 epochs were used. Further training degrades the network leading to poor results. The homogenous networks of T-T-T-T "hyperbolic tangent" and linear "L-L-L-L" also proved they were good at ANN NSE stock price prediction.

For the web application which used the trained neural networks of the stocks running on Django framework, chart JS and jquery were used to provide interactive and colorful graphs. The use of a web platform would make it easier to access the information.

5.1 Recommendations

Further research and experimentation is needed to implement existing activation functions which have not been thoroughly tested in neural network works with the aim of improving ANNs accuracy on all fields i.e. classification, predictions etc.

Further research needs to be done in areas where ANNs are applied other than prediction i.e. image recognition etc. as the research has proven changing the activation function can have a positive or negative effect on the ANN performance.

3.0 References

1. Adebiyi, A. A., Ayo, C. K., Marion A. O., Otokiti, S. O. (January, 2012). Stock Price Prediction using Neural Network with Hybridized Market Indicators. Journal of Emerging Trends in Computing and Information Sciences, 3(1), ISSN 20798407, pp. 1-9. Retrieved from http://www.cisjournal.org. [Accessed 9 January 2017].

2. Bekir K & Vehbi A. O. (2010). Performance Analysis of Various Activation Functions in Generalized MLP Architectures of Neural Networks. International Journal of Artificial Intelligence and Expert Systems (IJAE), 1(4) pp. 111 – 122. Retrieved from http://www.cscjournals.org/index.php. [Accessed 20 January 2017].

3. Caroline C., Myszewski D., and Jimmy P. Stanford University: Neural networks history. [lecture notes] Retrieved December 14, 2016, from https://cs.stanford.edu/people/eroberts/courses/soco/projects/neural-networks/History/history1.html.

4. Chen C., Seff A., Kornhauser A. & Jianxiong X. (May 2015).DeepDriving: Learning Affordance for Direct Perception in Autonomous Driving. Princeton University. Retrieved from http://deepdriving.cs.princeton.edu, https://arxiv.org/abs/1505.00256 [Accessed 22 January 2017]

5. Da Silva, I.N., Hernane S.D., Andrade F.R.., Liboni, L.H.B. & dos Reis Alves S.F. (September 2016). Artificial Neural Networks, A Practical Course. (pp.25,26). Switzerland .Zurich, MT: Springer International Publishing Switzerland.

6. de Oliveira F. A., Nobre C. N., Zárate L. E (2013). Applying Artificial Neural Networks to prediction of stock price and improvement of the directional prediction index – Case study of PETR4, Petrobras, Brazil. Expert systems with applications. 40(18) pp.7596- 7606. Retrieved from www.elsevier.com/locate/eswa [Accessed 22 April 2017].

7. Hafezi (R., Shahrabi J. & Hadavandi E.(2015). A bat-neural network multi-agent system (BNNMAS) for stock price prediction: Case study of DAX stock price. Applied soft computing 29(), pp. 196-210. Retrieved from http://www.elsevier.com/locate/asoc [Accessed 21 April 2017].

8. Hu, M.J.C. (June 1964). Application of the Adaline System to Weather Forecasting (Master Thesis Technical Report 6775-1), Stanford Electronic Laboratories, Stanford, CA

9. Kenya National Bureau of Statistics (2016). Economic survey. Retrieved from **http://www.knbs.or.ke.** [Accessed 4 January 2017].

10. Kriesel D. 2007. A brief Introduction to Neural Networks. Retrieved from http://www.dkriesel.com/en/science/neural_networks. [Accessed September 17, 2016]

11. Liu Y., Gadepalli K., Norouzi M., George E. D., Timo K., Aleksey B., Subhashini V., Aleksei T., Philip Q. N., Greg S. C., Jason D. H., Lily P. & Martin C. S.(March 2017). Detecting Cancer Metastases on Gigapixel Pathology Images. arXiv:1703.02442v2. Retrieved from https://arxiv.org/ [Accessed 9 January 2017]

12. Masoud, N. (2014). Predicting direction of stock prices index movement using artificial Neural networks: The case of Libyan financial market. British Journal of Economics, Management & Trade, 4(4) pp. 597-619. Retrieved from **https://www.researchgate.net/** **.**[Accessed 22 April 2017].

13. Mayankkumar B. P. & Yalamalle R. S. (June 2014). Stock Price Prediction Using Artificial Neural Network. International Journal of Innovative Research in Science, Engineering and Technology IJIRSET, 3(6), pp. 13755- 13762. Retrieved from https://www.ijirset.com [Accessed 17 April 2017].

14. Nairobi Securities Exchange (n.d). NSE equities securities trading rules "amended". Retrieved from https://www.nse.co.ke/regulatory-framework/nairobi-securities-exchange.html. [Accessed 17 March 2017].

15. Nairobi Securities Exchange. (n.d.).List of Member Firms. Retrieved from https://www.nse.co.ke/member-firms/firms.html [Accessed 17 March 2017]

16. Navghane, A. P., & Patil, P. M. (October, 2013). Forecasting stock market movement: a neural network approach. International Journal of Electronics and Communication Engineering & Technology (IJECET), 4(5), ISSN 0976 –6464(print), ISSN 0976 – 6472(online) IAEME, pp. 117-125.

17. Otavio G. (July 29, 2015) How Google Translate squeezes deep learning onto a phone**.** [google research blog].Retrieved [17 March 2017] from https://research.googleblog.com/2015/07/how-google-translate-squeezes-deep.html

18. Pakdaman M. N., Taremian H. & Hashemi H. B. (May 2010).Stock market value prediction using neural networks. International Conference on Computer Information Systems and Industrial Management Applications (CISIM).pg135

19. Pomerleau D.A. (January 1989). ALVINN: An Autonomous Land Vehicle in a Neural Network. Advances in Neural Information Processing Systems 1 (NIPS). Retrieved from http://repository.cmu.edu/compsci [Accessed 21 January 2017].

20. Pratiwi D., Santika D. D. & Pardamean B. (September 2013). An Application of Backpropagation Artificial Neural Network Method for Measuring the Severity of Osteoarthritis. International Journal of Engineering & Technology IJET-IJENS, 11(03), pp. 102-105. Retrieved from https://arxiv.org/abs/1309.7522. [Accessed 21 February 2017].

21. Priyanka W., Sonali, B. M. (January 2014) Research Paper on Basic of Artificial Neural Network. International Journal on Recent and Innovation Trends in Computing and Communication (IJRITCC), 2(1). ISSN: 2321-8169, pp. 96 – 100. Retrieved from http://www.ijritcc.org. [Accessed 14 February 2017].

22. Rohit R. D. (January 2012). On The Rainfall Time Series Prediction Using Multilayer Perceptron Artificial Neural Network. International Journal of Emerging Technology and Advanced Engineering (IJETAE), 2(1). ISSN 2250-2459, pp. 148-153. Retrieved from http://www.ijetae.com/index.html. [Accessed 20 January 2017].

23. Rotich Kibet N. 2012 Master's Thesis. Forecasting of wind speeds and directions with artificial neural networks. Retrieved from **https://www.researchgate.net/publication/299507369_Rotich_M_Sc_Thesis.** [Accessed 21 march 2017].

24. Safi. S. & White. A. (April 2017). Short and long-term forecasting using artificial neural networks for stock prices in Palestine: a comparative study. Electronic Journal of Applied Statistical Analysis, 10(10), pp. 14-28. . Retrieved from http://siba-ese.unisalento.it/index.php/ejasa/index **.** [Accessed 22 April 2017].

25. Smith W. S. (1997) .The Scientist and Engineer's Guide to Digital Signal Processing, chapter 26, pg. 461. Retrieved from http://www.dspguide.com/. California, CA: California Technical Publishing. [Accessed 17 August 2016].

26. Stanford university (6 April 2013). Neural networks. [lecture notes] Retrieved from http://ufldl.stanford.edu/wiki/index.php/Neural_Networks. [Accessed 17 December 2016]

27. Wanjawa W.B. 2014 Master's Thesis. A Neural Network Model for Predicting Stock Market Prices at the Nairobi Securities Exchange Retrieved from https://www.researchgate.net/publication/269087026**.** [Accessed 6 October 2016].

28. White, H. (1988), economic prediction using neural networks: the case of IBM daily stock returns. Retrieved from: https://scholar.google.com/scholar?as_sdt=2005&hl=en&sciodt=0,5&cites=8960476038735531826&scipsc. [Accessed 10 January 2017].

29. Yunjie L., Joaquin C., Kunkel k., Racah E., Khosrowshahi A., Wehner M., Prabhat, David L. & Collins W. **(**4 May 2016**).**Application of Deep Convolutional Neural Networks for Detecting Extreme Weather in Climate Datasets. Retrieved from https://arxiv.org/abs/1605.01156 . [Accessed January 17, 2017]

Appendices

Appendix 1 NSE listed companies as at 2017

AGRICULTURAL
Eaagads Ltd Ord 1.25 AIMS
Kapchorua Tea Co. Ltd Ord Ord 5.00 AIMS
Kakuzi Ord.5.00
Limuru Tea Co. Ltd Ord 20.00
Rea Vipingo Plantations Ltd Ord 5.00
Sasini Ltd Ord 1.00
Williamson Tea Kenya Ltd Ord 5.00
AUTOMOBILES AND ACCESSORIES
Car and General (K) Ltd Ord 5.00
Sameer Africa Ltd Ord 5.00
BANKING
Barclays Bank Ltd Ord 0.50
CFC Stanbic Holdings Ltd ord.5.00
I&M Holdings Ltd Ord 1.00
Diamond Trust Bank Kenya Ltd Ord 4.00
HF Group Ltd Ord 5.00
KCB Group Ltd Ord 1.00
National Bank of Kenya Ltd Ord 5.00
NIC Bank Ltd 0rd 5.00
Standard Chartered Bank Ltd Ord 5.00
Equity Group Holdings Ord 0.50
The Co-operative Bank of Kenya Ltd Ord 1.00
COMMERCIAL AND SERVICES
Express Ltd Ord 5.00
Kenya Airways Ltd Ord 5.00
Nation Media Group Ord. 2.50
Standard Group Ltd Ord 5.00
TPS Eastern Africa (Serena) Ltd Ord 1.00
Scangroup Ltd Ord 1.00
Uchumi Supermarket Ltd Ord 5.00
Longhorn Publishers Ltd
Atlas Development and Support Services
Deacons (East Africa) Plc Ord 2.50
Nairobi Business Ventures Ltd
CONSTRUCTION AND ALLIED
Athi River Mining Ord 5.00
Bamburi Cement Ltd Ord 5.00
Crown Berger Ltd 0rd 5.00

E.A.Cables Ltd Ord 0.50
E.A.Portland Cement Ltd Ord 5.00
ENERGY AND PETROLEUM
KenolKobil Ltd Ord 0.05
Total Kenya Ltd Ord 5.00
KenGen Ltd Ord. 2.50
Kenya Power & Lighting Co Ltd
Umeme Ltd Ord 0.50
INSURANCE
Jubilee Holdings Ltd Ord 5.00
Sanlam Kenya PLC 0rd 5.00
Kenya Re-Insurance Corporation Ltd Ord 2.50
Liberty Kenya Holdings Ltd
Britam Holdings Ltd Ord 0.10
CIC Insurance Group Ltd Ord 1.00
INVESTMENT
Olympia Capital Holdings ltd Ord 5.00
Centum Investment Co Ltd Ord 0.50
Trans-Century Ltd
Home Afrika Ltd Ord 1.00
Kurwitu Ventures
INVESTMENT SERVICES
Nairobi Securities Exchange Ltd Ord 4.00
MANUFACTURING AND ALLIED
B.O.C Kenya Ltd Ord 5.00
British American Tobacco Kenya Ltd Ord 10.00
Carbacid Investments Ltd Ord 5.00
East African Breweries Ltd Ord 2.00
Mumias Sugar Co. Ltd Ord 2.00
Unga Group Ltd Ord 5.00
Eveready East Africa Ltd Ord.1.00
Kenya Orchards Ltd Ord 5.00
Flame Tree Group Holdings Ltd Ord 0.825
TELECOMMUNICATION AND TECHNOLOGY
Safaricom Ltd Ord 0.05

Table A1-1 NSE listed companies as at 2017,Source :NSE 2017

Appendix 2 screen shot of NSE website

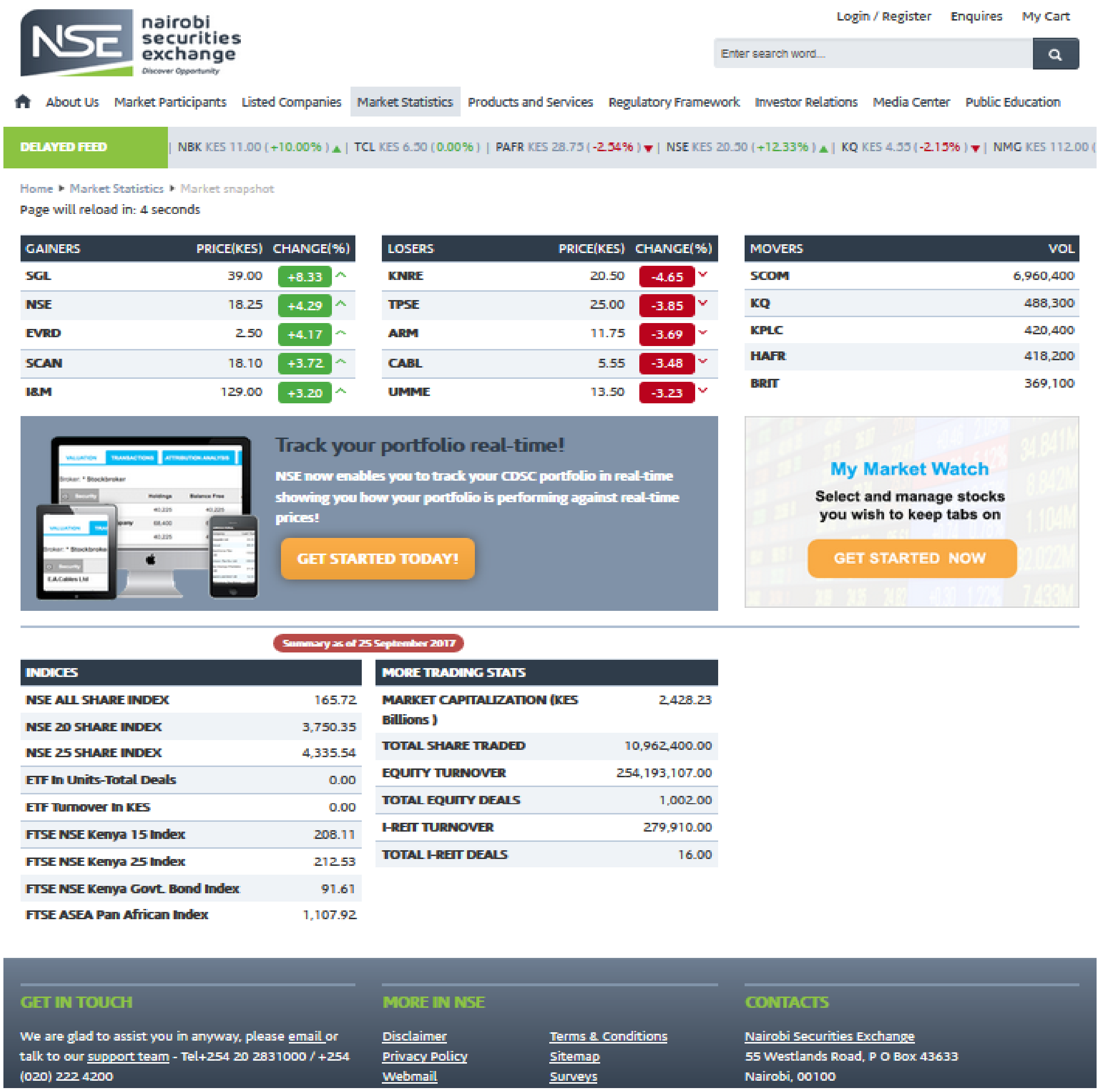


Figure A2.1 screen shot of NSE website

Source : NSE 2017

Appendix 3 screen shot of mystocks Kenya website

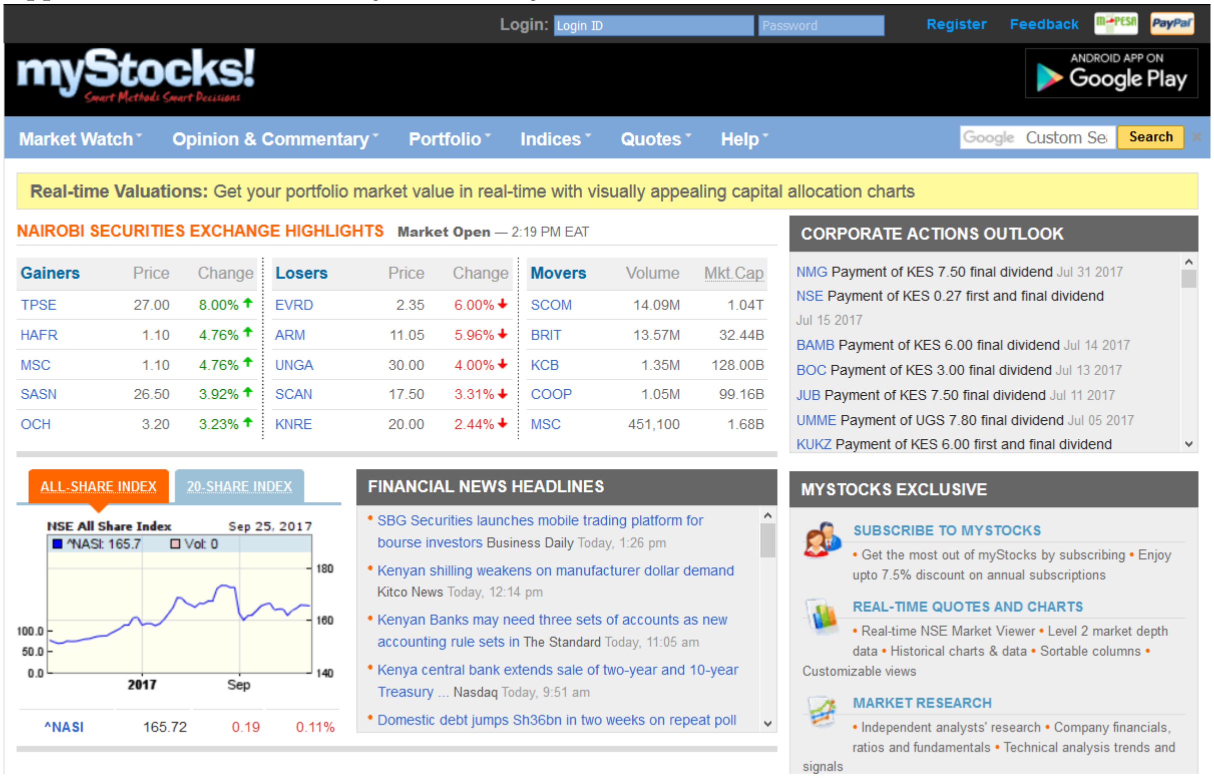


Figure A3.1 screen shot of mystocks Kenya website

Source : mystocks Kenya website

Appendix 4 test results of the top activation functions combinations

Table A4-1 EABL stock results

	t-sf-sf-t			t-t-t-t		l-l-l-l		s-t-t-s		s-s-s-s	
date	expected	prediction	forecast error	prediction	forecast error	Prediction	forecast error	prediction	forecast error	prediction	forecast error
23-Dec-16	230	228	0.008695652	230	0	229	0.004347826	223	0.030434783	224	0.026086957
3-Jan-17	245	233	0.048979592	250	-0.020408163	245	0	238	0.028571429	234	0.044897959
9-Jan-17	225	233	-0.035555556	231	-0.026666667	229	-0.017777778	224	0.004444444	227	-0.008888889
13-Jan-17	215	218	-0.013953488	222	-0.03255814	219	-0.018604651	217	-0.009302326	218	-0.013953488
19-Jan-17	225	218	0.031111111	221	0.017777778	221	0.017777778	216	0.04	217	0.035555556
25-Jan-17	218	221	-0.013761468	222	-0.018348624	221	-0.013761468	216	0.009174312	218	0
30-Jan-17	226	219	0.030973451	227	-0.004424779	224	0.008849558	219	0.030973451	219	0.030973451
2-Feb-17	230	225	0.02173913	231	-0.004347826	229	0.004347826	223	0.030434783	223	0.030434783
8-Feb-17	230	229	0.004347826	231	-0.004347826	229	0.004347826	223	0.030434783	224	0.026086957
14-Feb-17	222	224	-0.009009009	220	0.009009009	219	0.013513514	216	0.027027027	219	0.013513514
20-Feb-17	226	223	0.013274336	224	0.008849558	223	0.013274336	218	0.03539823	219	0.030973451
24-Feb-17	226	223	0.013274336	228	-0.008849558	226	0	220	0.026548673	221	0.022123894
2-Mar-17	219	225	-0.02739726	221	-0.00913242	221	-0.00913242	216	0.01369863	219	0
8-Mar-17	216	213	0.013888889	218	-0.009259259	217	-0.00462963	213	0.013888889	214	0.009259259
14-Mar-17	215	212	0.013953488	216	-0.004651163	214	0.004651163	212	0.013953488	213	0.009302326
20-Mar-17	220	215	0.022727273	218	0.009090909	217	0.013636364	214	0.027272727	215	0.022727273
24-Mar-17	221	221	0	222	-0.004524887	222	-0.004524887	217	0.018099548	218	0.013574661
30-Mar-17	226	222	0.017699115	225	0.004424779	223	0.013274336	219	0.030973451	220	0.026548673
5-Apr-17	227	226	0.004405286	227	0	226	0.004405286	221	0.026431718	222	0.022026432
11-Apr-17	245	229	0.065306122	231	0.057142857	229	0.065306122	224	0.085714286	224	0.085714286
18-Apr-17	240	240	0	243	-0.0125	240	0	233	0.029166667	234	0.025
20-Apr-17	240	240	0	241	-0.004166667	239	0.004166667	232	0.033333333	233	0.029166667
24-Apr-17	240	240	0	241	-0.004166667	239	0.004166667	232	0.033333333	232	0.033333333
26-Apr-17	239	239	0	240	-0.0041841	238	0.0041841	231	0.033472803	232	0.029288703
28-Apr-17	230	234	-0.017391304	231	-0.004347826	230	0	224	0.026086957	227	0.013043478
8-May-17	230	230	0	229	0.004347826	228	0.008695652	223	0.030434783	224	0.026086957
12-May-17	227	229	-0.008810573	236	-0.039647577	234	-0.030837004	227	0	227	0
18-May-17	235	225	0.042553191	233	0.008510638	230	0.021276596	224	0.046808511	224	0.046808511
24-May-17	235	233	0.008510638	233	0.008510638	231	0.017021277	225	0.042553191	227	0.034042553
30-May-17	241	241	0	243	-0.008298755	242	-0.004149378	234	0.029045643	233	0.033195021
6-Jun-17	233	240	-0.030042918	241	-0.034334764	239	-0.025751073	232	0.004291845	233	0
12-Jun-17	239	235	0.016736402	236	0.012552301	234	0.020920502	228	0.046025105	229	0.041841004
16-Jun-17	252	247	0.01984127	261	-0.035714286	258	-0.023809524	249	0.011904762	244	0.031746032
22-Jun-17	265	264	0.003773585	269	-0.01509434	267	-0.00754717	258	0.026415094	254	0.041509434
30-Jun-17	240	262	-0.091666667	261	-0.0875	258	-0.075	251	-0.045833333	251	-0.045833333
7-Jul-17	260	243	0.065384615	254	0.023076923	251	0.034615385	242	0.069230769	239	0.080769231
13-Jul-17	246	249	-0.012195122	250	-0.016260163	247	-0.004065041	240	0.024390244	241	0.020325203

Table A4-2 equity stocks results

		t-sf-sf-t		t-t-t-t		l-l-l-l		s-t-t-s		s-s-s-s	
date	expected	prediction	forecast error	prediction	forecast error	prediction	forecast error	prediction	forecast error	prediction	forecast error
29-Sep-16	31	27	0.129032258	28	0.096774194	27	0.129032258	26	0.161290323	27	0.129032258
5-Oct-16	30.75	31	-0.008130081	31	-0.008130081	31	-0.008130081	29	0.056910569	30	0.024390244
11-Oct-16	30.25	31	-0.024793388	31	-0.024793388	30	0.008264463	28	0.074380165	29	0.041322314
17-Oct-16	31	30	0.032258065	30	0.032258065	30	0.032258065	28	0.096774194	29	0.064516129
24-Oct-16	30.75	31	-0.008130081	30	0.024390244	30	0.024390244	28	0.089430894	29	0.056910569
28-Oct-16	30.75	31	-0.008130081	30	0.024390244	30	0.024390244	28	0.089430894	29	0.056910569
2-Nov-16	31	30	0.032258065	30	0.032258065	30	0.032258065	27	0.129032258	29	0.064516129
9-Nov-16	32	31	0.03125	30	0.0625	30	0.0625	28	0.125	29	0.09375
15-Nov-16	32.25	32	0.007751938	31	0.03875969	31	0.03875969	29	0.100775194	30	0.069767442
21-Nov-16	31.75	32	-0.007874016	31	0.023622047	31	0.023622047	29	0.086614173	30	0.05511811
25-Nov-16	30.75	31	-0.008130081	30	0.024390244	30	0.024390244	28	0.089430894	29	0.056910569
1-Dec-16	30.25	30	0.008264463	29	0.041322314	29	0.041322314	27	0.107438017	28	0.074380165
7-Dec-16	30.25	30	0.008264463	29	0.041322314	29	0.041322314	27	0.107438017	28	0.074380165
14-Dec-16	30	30	0	29	0.033333333	29	0.033333333	27	0.1	28	0.066666667
20-Dec-16	30.5	30	0.016393443	29	0.049180328	29	0.049180328	27	0.114754098	28	0.081967213
28-Dec-16	30	30	0	29	0.033333333	29	0.033333333	27	0.1	28	0.066666667
4-Jan-17	30	30	0	29	0.033333333	29	0.033333333	27	0.1	28	0.066666667
10-Jan-17	27.5	28	-0.018181818	27	0.018181818	28	-0.018181818	25	0.090909091	27	0.018181818
16-Jan-17	27	26	0.037037037	26	0.037037037	26	0.037037037	24	0.111111111	25	0.074074074
20-Jan-17	26	26	0	26	0	26	0	25	0.038461538	26	0
26-Jan-17	25.5	25	0.019607843	25	0.019607843	25	0.019607843	24	0.058823529	25	0.019607843
31-Jan-17	25.5	23	0.098039216	22	0.137254902	23	0.098039216	22	0.137254902	24	0.058823529
3-Feb-17	25.5	25	0.019607843	24	0.058823529	24	0.058823529	24	0.058823529	25	0.019607843
9-Feb-17	27.5	26	0.054545455	27	0.018181818	25	0.090909091	25	0.090909091	26	0.054545455
15-Feb-17	27.25	27	0.009174312	27	0.009174312	26	0.04587156	25	0.082568807	26	0.04587156
21-Feb-17	27	27	0	26	0.037037037	26	0.037037037	25	0.074074074	26	0.037037037
27-Feb-17	26.5	26	0.018867925	26	0.018867925	26	0.018867925	25	0.056603774	26	0.018867925
3-Mar-17	26	25	0.038461538	25	0.038461538	25	0.038461538	24	0.076923077	25	0.038461538
9-Mar-17	26.75	25	0.065420561	25	0.065420561	25	0.065420561	24	0.102803738	25	0.065420561
15-Mar-17	28	28	0	30	-0.071428571	28	0	27	0.035714286	28	0
21-Mar-17	29.75	29	0.025210084	29	0.025210084	28	0.058823529	27	0.092436975	28	0.058823529
27-Mar-17	30.75	30	0.024390244	30	0.024390244	29	0.056910569	28	0.089430894	29	0.056910569
31-Mar-17	33	32	0.03030303	33	0	31	0.060606061	30	0.090909091	31	0.060606061
6-Apr-17	34.75	33	0.050359712	33	0.050359712	32	0.079136691	30	0.136690647	31	0.107913669
12-Apr-17	33	34	-0.03030303	32	0.03030303	33	0	30	0.090909091	31	0.060606061
20-Apr-17	33	33	0	31	0.060606061	32	0.03030303	29	0.121212121	30	0.090909091
26-Apr-17	34	33	0.029411765	33	0.029411765	32	0.058823529	31	0.088235294	31	0.088235294

Table A4-3 nmg stocks results

		t-sf-sf-t		t-t-t-t		l-l-l-l		s-t-t-s		s-s-s-s	
date	expected	prediction	forecast error	Prediction	forecast error	prediction	forecast error	prediction	forecast error	prediction	forecast error
23-Dec-16	88	88.156484	-0.001778224	87.915122	0.000964522	87.343442	0.007460885	87.264765	0.008354946	96.354467	-0.0949
3-Jan-17	93	89.112835	0.041797476	91.27892	0.018506236	90.464965	0.027258438	88.353516	0.04996219	97.806845	-0.0517
9-Jan-17	87.5	88.966101	-0.016755436	88.466995	-0.011051376	88.095132	-0.00680151	87.981728	-0.005505461	96.253016	-0.1000
13-Jan-17	79.5	86.302082	-0.085560782	83.080395	-0.045036411	82.833722	-0.04193361	85.239998	-0.072201229	94.352967	-0.1868
19-Jan-17	79.5	79.527133	-0.000341294	78.177967	0.016629346	77.778056	0.021659679	74.173947	0.066994372	92.043722	-0.1578
25-Jan-17	75.5	81.080314	-0.073911448	78.42591	-0.038753773	78.20202	-0.035788345	77.723411	-0.029449146	92.520375	-0.2254
30-Jan-17	75.5	78.318255	-0.037327886	75.641724	-0.001877134	75.666969	-0.002211508	74.471318	0.013624932	91.583251	-0.2130
2-Feb-17	74	78.021179	-0.054340254	74.918545	-0.012412774	74.923843	-0.012484359	73.813752	0.002516863	91.281578	-0.2335
8-Feb-17	75.5	77.777217	-0.030161814	75.551976	-0.00068843	75.516059	-0.000212701	73.682851	0.024068205	91.570745	-0.2129
14-Feb-17	79	80.312775	-0.016617406	78.555582	0.005625544	78.104992	0.011329218	76.745923	0.028532622	92.720925	-0.1737
20-Feb-17	83.5	83.825499	-0.003898195	83.332023	0.002011698	83.612518	-0.001347521	84.926821	-0.017087671	95.192286	-0.1400
24-Feb-17	89	85.635786	0.037800154	85.673427	0.037377221	85.141999	0.043348327	84.173835	0.05422657	95.494138	-0.0730
2-Mar-17	85.5	88.067529	-0.030029575	86.564031	-0.012444806	85.99561	-0.005796613	86.768534	-0.014836653	95.681942	-0.1191
8-Mar-17	84	85.63046	-0.019410236	84.442138	-0.005263542	83.484613	0.006135558	81.956494	0.024327456	94.547545	-0.1256
14-Mar-17	85	85.03119	-0.000366944	83.551637	0.017039564	83.459605	0.018122296	84.302896	0.008201226	94.739654	-0.1146
20-Mar-17	85.5	86.43179	-0.010898126	85.945807	-0.005214122	85.633224	-0.00155817	85.905862	-0.004746927	95.716756	-0.1195
24-Mar-17	93	88.843752	0.044690836	90.687987	0.024860352	90.644749	0.025325283	91.307512	0.018198798	98.305582	-0.0570
30-Mar-17	98	98.327524	-0.003342084	100.36265	-0.02410872	99.25653	-0.012821734	102.48942	-0.045810421	103.34617	-0.0546
5-Apr-17	92	95.417746	-0.037149418	96.005055	-0.043533203	95.430224	-0.037285039	97.264619	-0.05722412	100.14489	-0.0885
11-Apr-17	89	89.901921	-0.010133946	89.659616	-0.007411413	89.243014	-0.002730492	89.772761	-0.008682709	97.03752	-0.0903
18-Apr-17	90	90.033014	-0.000366824	90.470647	-0.005229416	90.081363	-0.000904031	91.089537	-0.012105965	97.810921	-0.0868
20-Apr-17	93.5	91.183951	0.024770575	92.892955	0.006492458	92.171345	0.014210214	91.835553	0.017801571	98.776064	-0.0564
24-Apr-17	94	92.760105	0.013190371	93.989599	0.000110647	93.345108	0.006966941	93.760342	0.002549551	99.195795	-0.0553
26-Apr-17	94.5	94.038991	0.004878405	95.092513	-0.006269981	94.443936	0.000593275	95.863545	-0.014429044	99.921208	-0.0574
28-Apr-17	95.5	94.301093	0.012554	95.726663	-0.002373438	95.029394	0.004927815	96.042329	-0.00567884	100.16874	-0.0489
8-May-17	102	98.872277	0.030663954	101.68438	0.003094321	101.33683	0.006501707	105.36197	-0.032960483	104.31457	-0.0227
12-May-17	103	100.21557	0.02703332	102.36768	0.006139026	101.54056	0.014169367	103.58618	-0.005691025	103.59702	-0.0058
18-May-17	108	103.73093	0.039528464	107.12512	0.008100717	106.03009	0.018239934	108.63035	-0.005836568	106.65289	0.0125
24-May-17	116	108.64326	0.063420138	111.94094	0.03499191	111.12712	0.042007592	116.15131	-0.001304357	110.06888	0.0511
30-May-17	115	113.97676	0.008897719	114.84374	0.001358758	114.07839	0.008014022	121.30607	-0.054835405	111.793	0.0279
6-Jun-17	116	113.88689	0.018216502	117.61486	-0.013921213	116.73942	-0.00637428	122.14816	-0.053001383	113.61366	0.0206
12-Jun-17	103	113.44232	-0.101381748	116.07194	-0.126912086	114.99744	-0.11648	119.91393	-0.164212912	112.12774	-0.0886
16-Jun-17	105	103.12561	0.017851297	107.18501	-0.020809637	105.86281	-0.008217279	106.89036	-0.018003439	106.33945	-0.0128
22-Jun-17	107	105.14439	0.017342132	107.85523	-0.00799281	106.8799	0.001122449	110.5623	-0.033292496	107.17662	-0.0017
30-Jun-17	108	104.91275	0.028585656	107.44131	0.005173081	106.24923	0.016210879	108.60062	-0.005561326	106.35932	0.0152
7-Jul-17	109	106.21182	0.025579661	108.85417	0.001337846	108.20843	0.007262103	112.50042	-0.032113933	107.75543	0.0114
13-Jul-17	109	106.92809	0.019008372	108.55477	0.004084671	107.49896	0.013770982	111.54975	-0.023392229	107.26358	0.0159

Table A4-4 kengen stock results

		t-sf-sf-t		t-t-t-t		l-l-l-l		s-t-t-s		s-s-s-s	
date	expected	prediction	forecast error	prediction	forecast error	prediction	forecast error	prediction	forecast error	prediction	forecast error
23-Dec-16	5.9	5.9258853	-0.004387333	5.93781	-0.006408479	5.8943082	0.000964704	6.3471008	-0.075779793	6.346498	-0.075677569
3-Jan-17	5.75	5.941512	-0.033306442	5.8541118	-0.018106398	5.8356178	-0.014890049	6.3418145	-0.102924259	6.330359	-0.100932011
9-Jan-17	5.7	5.854363	-0.027081235	5.797535	-0.017111404	5.7695485	-0.012201492	6.2807002	-0.101877237	6.280577	-0.101855572
13-Jan-17	5.6	5.8049217	-0.036593155	5.7125088	-0.020090866	5.6328199	-0.005860689	6.2377312	-0.113880575	6.241052	-0.114473554
19-Jan-17	5.45	5.6927855	-0.044547805	5.5421204	-0.016902828	5.4673206	-0.003178085	6.1568136	-0.129690563	6.166564	-0.131479713
25-Jan-17	5	5.4166213	-0.083324257	5.0048256	-0.000965124	4.8850673	0.022986543	5.9559422	-0.191188443	5.973273	-0.194654561
30-Jan-17	5.1	5.3703205	-0.05300402	5.1086196	-0.001690122	5.1533755	-0.010465787	5.9817572	-0.172893562	6.014905	-0.179393167
2-Feb-17	5.5	5.5379196	-0.006894473	5.4560823	0.007985038	5.4453758	0.009931679	6.0918438	-0.107607957	6.123643	-0.113389585
8-Feb-17	5.9	5.9222583	-0.00377259	6.0373222	-0.023274949	5.7994803	0.017037232	6.3354783	-0.073809874	6.349547	-0.076194446
14-Feb-17	6.3	6.057325	0.038519835	6.1228315	0.028121984	6.1147933	0.029397893	6.4426084	-0.022636257	6.431128	-0.020813909
20-Feb-17	6.55	6.4636789	0.013178788	6.6776034	-0.019481432	6.4908282	0.009033866	6.7336755	-0.028042059	6.696615	-0.022383918
24-Feb-17	6.5	6.518409	-0.002832147	6.6760983	-0.027092054	6.5452363	-0.006959435	6.7578733	-0.039672815	6.70788	-0.031981497
2-Mar-17	6.25	6.2276559	0.003575049	6.2760751	-0.004172018	6.1638505	0.013783912	6.4918013	-0.038688208	6.460805	-0.033728804
8-Mar-17	6.25	6.2415797	0.001347248	6.3047608	-0.008761734	6.1811823	0.011010825	6.5430861	-0.046893775	6.514216	-0.042274482
14-Mar-17	6.45	6.3254031	0.019317345	6.5148549	-0.010055026	6.6805171	-0.035739082	6.6717138	-0.03437424	6.63635	-0.028891475
20-Mar-17	6.5	6.4721717	0.004281273	6.6381525	-0.021254231	6.4885509	0.001761401	6.7252143	-0.034648357	6.68151	-0.027924563
24-Mar-17	6.55	6.5147708	0.005378507	6.6914084	-0.021589063	6.538402	0.001770683	6.7579203	-0.031743552	6.71071	-0.024535813
30-Mar-17	6.55	6.5208304	0.004453381	6.6694442	-0.018235753	6.5680165	-0.002750613	6.7664398	-0.033044241	6.71469	-0.025143526
5-Apr-17	6.55	6.3641436	0.028375017	6.4664499	0.012755738	6.3461456	0.031122813	6.6101602	-0.009184764	6.570277	-0.00309568
11-Apr-17	6.5	6.518409	-0.002832147	6.6760983	-0.027092054	6.5452363	-0.006959435	6.7578733	-0.039672815	6.70788	-0.031981497
18-Apr-17	6.55	6.5147708	0.005378507	6.6914084	-0.021589063	6.538402	0.001770683	6.7579203	-0.031743552	6.71071	-0.024535813
20-Apr-17	6.35	6.4028754	-0.00832683	6.4549463	-0.016526971	6.307378	0.00671213	6.6452679	-0.046498883	6.598891	-0.039195496
24-Apr-17	6.5	6.4118317	0.01356436	6.595824	-0.014742153	6.5626036	-0.00963133	6.7038656	-0.031363944	6.664577	-0.025319497
26-Apr-17	6.45	6.4502331	-3.61455E-05	6.5721211	-0.018933511	6.430438	0.003032865	6.6982224	-0.038484093	6.652711	-0.03142806
28-Apr-17	6.45	6.4201428	0.004629026	6.5511864	-0.015687815	6.4674644	-0.002707659	6.6876825	-0.036850007	6.644378	-0.030136133
8-May-17	6.65	6.555165	0.014260903	6.7619335	-0.016832107	6.6201449	0.004489484	6.8086945	-0.023863833	6.759871	-0.016521943
12-May-17	6.65	6.5908936	0.008888182	6.7648592	-0.017272058	6.6668689	-0.002536683	6.8189303	-0.025403055	6.761542	-0.016773212
18-May-17	6.85	6.6864974	0.023868988	6.9126916	-0.009152057	6.7537525	0.014050736	6.9022826	-0.007632491	6.841507	0.001239908
24-May-17	7.15	6.9553772	0.027219974	7.2979866	-0.020697425	7.103288	0.006533148	7.1354223	0.00203884	7.057445	0.012944796
30-May-17	7.9	7.4885801	0.052078463	8.0394908	-0.017657058	7.9008917	-0.000112873	7.683318	0.027428104	7.563313	0.042618607
6-Jun-17	7.85	7.7482801	0.012957946	8.0749992	-0.028662321	7.9359275	-0.010946179	7.8070014	0.005477529	7.646807	0.025884418
12-Jun-17	8.25	7.8368897	0.050073973	8.2139101	0.004374534	8.0379071	0.025708235	7.9137664	0.040755592	7.749481	0.060668956
16-Jun-17	8.6	8.3211755	0.032421449	8.7516451	-0.017633146	8.4771894	0.014280298	8.369386	0.026815577	8.169175	0.050095908
22-Jun-17	8.75	8.7034204	0.005323387	9.1129013	-0.041474439	8.8269972	-0.008799684	8.7495678	4.93999E-05	8.515108	0.026844795
30-Jun-17	7.95	8.0221948	-0.009081103	8.0910832	-0.017746313	7.7813099	0.021218876	7.8562831	0.011788292	7.66811	0.035457857
7-Jul-17	7.95	7.7253646	0.028256028	8.0775514	-0.016044205	7.7833005	0.02096849	7.7236763	0.02846839	7.576759	0.046948538
13-Jul-17	7.85	7.730676	0.015200514	8.0296703	-0.022887934	7.8103239	0.005054277	7.7513775	0.01256337	7.595125	0.032468213

Table A4-5 sasini stocks results

		t-sf-sf-t		t-t-t-t		l-l-l-l		s-t-t-s		s-s-s-s	
date	expected	prediction	forecast error	prediction	forecast error	prediction	forecast error	prediction	forecast error	Prediction	forecast error
23-Dec-16	18.4	17.8835555	0.028067637	18.242871	0.008539612	18.32535	0.004057249	19.79265	-0.075687715	16.7228	0.091152019
3-Jan-17	19.9	19.3343937	0.028422428	19.226106	0.033864037	19.48091	0.021059935	21.10766	-0.06068622	17.90076	0.100464183
9-Jan-17	20.25	20.0652386	0.009124019	20.546699	-0.014651824	20.59346	-0.016961037	22.22706	-0.0976326	18.96491	0.063461015
13-Jan-17	18.25	19.6170937	-0.074909243	20.015452	-0.096737095	20.04607	-0.098414612	21.68654	-0.188303461	18.33409	-0.004607703
19-Jan-17	18.2	17.8599886	0.018681944	18.173009	0.001483048	18.2404	-0.002219885	19.7798	-0.086802404	16.58215	0.088893038
25-Jan-17	17.6	17.575136	0.001412726	17.553501	0.002641964	17.70437	-0.005929992	19.34663	-0.099240608	16.17162	0.081157774
30-Jan-17	17.6	17.5193131	0.004584481	17.527659	0.004110308	17.68375	-0.004758641	19.28952	-0.095995689	16.17775	0.080809425
2-Feb-17	19	18.0781178	0.048520117	18.436477	0.029659106	18.50836	0.025875644	19.99877	-0.052566622	16.89192	0.110951454
8-Feb-17	18.5	17.9846754	0.027855383	18.193535	0.016565674	18.2814	0.01181602	19.86232	-0.073639028	16.51593	0.107247289
14-Feb-17	19.35	19.4512349	-0.00523178	19.970954	-0.032090646	20.06136	-0.036762997	21.53845	-0.113098449	18.31085	0.053702686
20-Feb-17	19.85	19.165505	0.034483376	18.965598	0.04455424	19.23094	0.031187017	20.91092	-0.053446844	17.5531	0.115712842
24-Feb-17	20.75	20.139077	0.029442072	20.670377	0.00383725	20.75786	-0.000378639	22.34211	-0.07672802	18.97866	0.085365636
2-Mar-17	19	19.4195867	-0.022083509	19.138936	-0.007312405	19.38995	-0.020523916	21.16717	-0.11406149	17.66181	0.070431199
8-Mar-17	20.25	19.89556	0.017503212	20.30526	-0.002728873	20.36151	-0.005506832	21.97008	-0.084942266	18.84421	0.069421647
14-Mar-17	21.25	20.0503648	0.05645342	20.625889	0.029369926	20.6597	0.027778696	22.26405	-0.04771977	18.88213	0.111429132
20-Mar-17	21.75	20.991287	0.034883357	21.711041	0.001791217	21.73667	0.000612775	23.47068	-0.079111891	20.10375	0.075689795
24-Mar-17	22.5	21.6235926	0.038951441	23.17824	-0.030143989	23.06455	-0.025090928	24.74083	-0.099592412	21.42044	0.047980483
30-Mar-17	24.25	22.4306298	0.075025577	23.945451	0.012558709	23.95536	0.012150273	25.84076	-0.065598256	22.54817	0.070178727
5-Apr-17	25.75	23.3500459	0.0932021	25.760643	-0.000413303	25.78665	-0.00142339	27.77224	-0.078533714	24.27559	0.057258619
11-Apr-17	27.75	23.1908392	0.164294082	25.467868	0.082238974	25.26714	0.089472581	27.39089	0.012940991	23.92367	0.137885928
18-Apr-17	24	23.1366131	0.035974453	25.252436	-0.052184842	24.94555	-0.039397906	27.14649	-0.131103727	23.25853	0.030894601
20-Apr-17	28	23.8319564	0.148858699	27.236248	0.027276868	27.38581	0.021935523	29.27951	-0.045696779	26.03617	0.070136666
24-Apr-17	27	24.1923504	0.103987022	27.337781	-0.012510423	27.23635	-0.008753568	29.60724	-0.096564561	26.04165	0.035494291
26-Apr-17	27.5	24.1500129	0.121817712	27.484157	0.000576112	27.43184	0.002478551	29.68407	-0.07942084	26.21368	0.046775115
28-Apr-17	27	24.1529899	0.10544482	27.288634	-0.01069016	27.21875	-0.008101788	29.54328	-0.094195569	25.99576	0.037194057
8-May-17	26.75	23.8992949	0.106568414	26.676061	0.002764077	26.60087	0.005574807	28.86293	-0.078987899	25.208	0.057644756
12-May-17	27.25	23.5485586	0.135832711	26.017685	0.045222569	25.84798	0.05145035	28.06613	-0.029949543	24.44161	0.103060068
18-May-17	25	23.4968102	0.060127593	25.83146	-0.033258404	25.62188	-0.024875176	27.88642	-0.115456991	24.23423	0.030630683
24-May-17	27.75	23.6623798	0.147301629	26.690102	0.038194519	26.6626	0.039185715	28.66871	-0.033106607	25.22384	0.091032744
30-May-17	25.5	24.3729425	0.044198333	27.891632	-0.093789488	27.87307	-0.093061542	30.20599	-0.184548636	26.69119	-0.046713389
6-Jun-17	25.75	24.0685763	0.065298008	26.834443	-0.042114307	26.97973	-0.047756494	29.24488	-0.135723529	25.95891	-0.008112958
12-Jun-17	26.5	23.6936836	0.105898731	26.08958	0.01548754	26.09123	0.015425411	28.31708	-0.068569115	24.86153	0.061829083
16-Jun-17	25.75	23.7155297	0.079008554	26.167503	-0.016213718	26.09096	-0.01324113	28.36444	-0.101531645	24.86606	0.034327911
22-Jun-17	26.5	23.7856425	0.102428586	26.313686	0.007030728	26.33492	0.006229312	28.55951	-0.07771722	25.11391	0.052305169
30-Jun-17	26.5	23.8970975	0.098222735	27.012932	-0.019355913	26.93177	-0.016293284	29.08101	-0.097396685	25.45792	0.039323629
7-Jul-17	26.5	23.5252974	0.112252928	26.132125	0.013882073	25.95699	0.020491113	28.11073	-0.060782104	24.44762	0.077448462
13-Jul-17	25.5	23.3227299	0.085383143	25.566518	-0.002608543	25.36544	0.005276735	27.56443	-0.080957987	23.99276	0.059107574

Appendix 5 prototype website screenshot

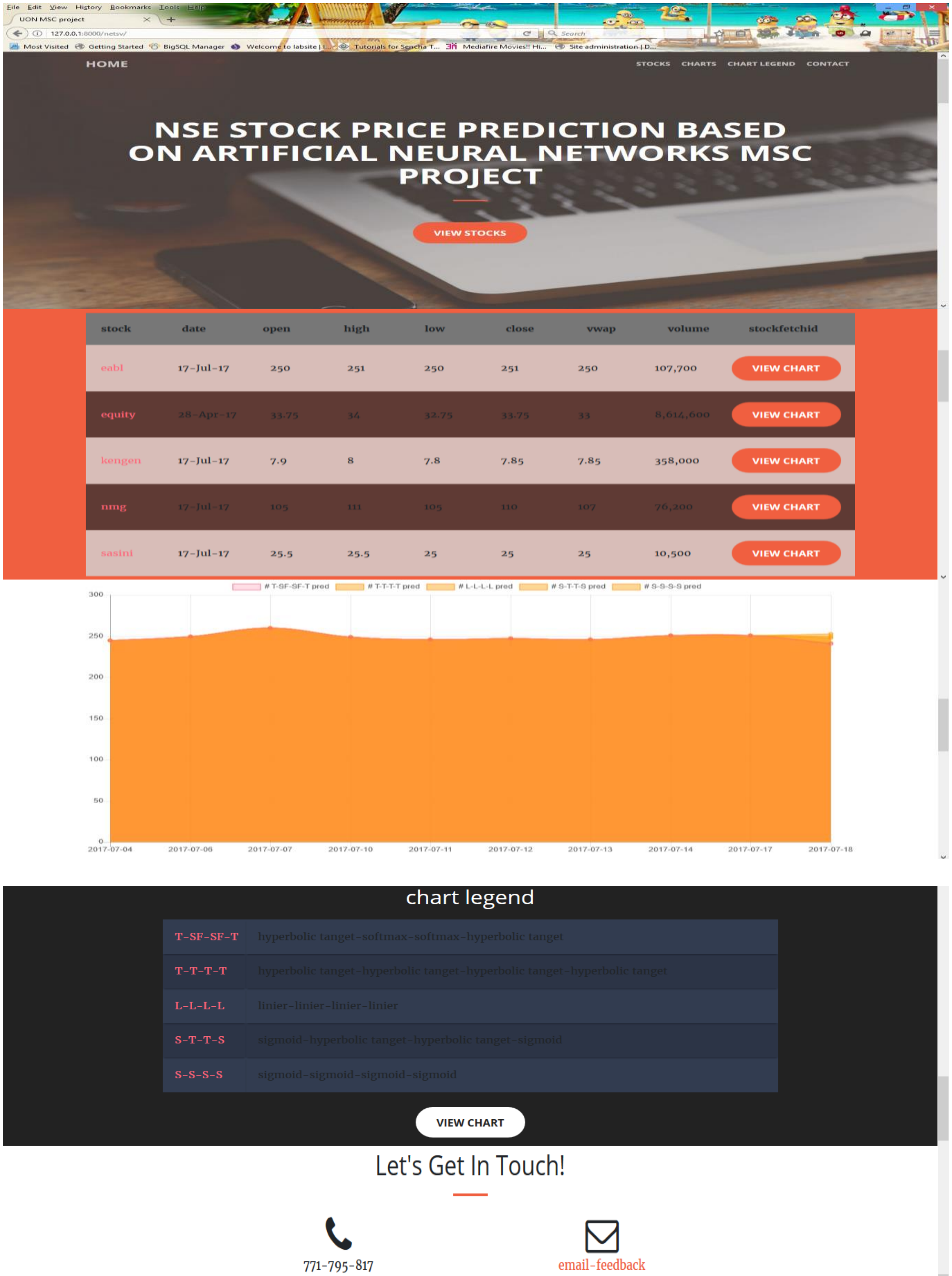


Figure A5.1 prototype website screenshot

Appendix 6 source code of the network training and testing

Training code

```
def apple(newminrx = 0,newmaxrx = 0.7, acfun =7 ):
	from pybrain.structure import TanhLayer
	from pybrain.structure import SoftmaxLayer
	from pybrain.structure import LinearLayer,
SigmoidLayer,GaussianLayer,LSTMLayer,MDLSTMLayer
	from pybrain.tools.shortcuts import buildNetwork
	from pybrain.datasets import SupervisedDataSet
	from pybrain.supervised.trainers import BackpropTrainer
	from matplotlib import pyplot as plt
	import math, os
	from pybrain.structure import FeedForwardNetwork
	from pybrain.structure import FullConnection
	import numpy as np
	from StringIO import StringIO
	#setting the base file path to be used in the code later
	scriptsloc = os.path.realpath(__file__)
	(filepath, filename) =os.path.split(scriptsloc)
	networkpath = os.path.join(filepath,"networksave")

	#getting data and normalizing the data in accordance to the minimum and maximum set
	#during calling the function
	try:
		os.makedirs(networkpath)
	except OSError as exception:
		pass
	xla = os.path.join(filepath, "pybraindatassasini.txt")
	datopen = open(xla,"r")
```

```
data =datopen.read()
input = np.genfromtxt(StringIO(data), delimiter=", " , usecols=(0,1,2,3,4,5,6,7,8,9),
autostrip=True)
target = np.genfromtxt(StringIO(data), delimiter="," ,usecols=(4), autostrip=True)
newmax =newmaxrx
newmin =newminrx
        if np.amax(input) >=np.amax(target):
        inmax = np.amax(input)
else:
        inmax = np.amax(target)
if np.amin(input) <=np.amin(target):
        inmin = np.amin(input)
else:
        inmin = np.amin(target)
inbtw = (newmax-newmin)/((inmax - inmin))
input = ((input - inmin) * inbtw)+newmin
target = ((target - inmin) * inbtw) +newmin
#adding data and splitting it in a propotion of 6:4 for training and testing
inpx = 4
outx = 1
ds = SupervisedDataSet(inpx, outx)
for ml in input:
        ds.addSample(ml[0:4],ml[4:5])
tstdata, trndata = ds.splitWithProportion( 0.25 )
#setting the activation function in accordance to the values used in calling the function
if acfun == 1:
        realfun = SigmoidLayer
elif acfun == 2:
        realfun = LinearLayer
```

```
	elif acfun == 3:
		realfun = GaussianLayer
	elif acfun == 4:
		realfun = LSTMLayer
	elif acfun == 5:
		realfun = MDLSTMLayer
	elif acfun == 6:
		realfun = SoftmaxLayer
	elif acfun == 7:
		realfun = TanhLayer
	#defining the network and the parameters of all its layers
	net = FeedForwardNetwork()
	labanin = realfun(4)
	labanhidden = realfun(8)
	labanhiddensecond = realfun(8)
	labanout = realfun(1)
	net.addInputModule(labanin)
	net.addModule(labanhidden)
	net.addModule(labanhiddensecond)
	net.addOutputModule(labanout)
	#connecting the different components of the network together and setting additional
parameters
	net.addConnection(FullConnection(labanin, labanhidden))
	net.addConnection(FullConnection(labanhidden, labanhiddensecond))
	net.addConnection(FullConnection(labanhiddensecond,labanout))
	net.sortModules()
	trainer = BackpropTrainer(net, dataset=trndata, learningrate=0.19  ,momentum=0.0,
weightdecay=0.0)
	#training of the network while getting the MSE and RMSE to an array
```

```
from pybrain.tools.validation import ModuleValidator
from pybrain.tools.validation import CrossValidator
modval = ModuleValidator()
msearray =[]
rmsearray =[]
epochnumbers =2000
for i in range(epochnumbers):
    trainer.trainEpochs(1)
    trainer.trainOnDataset(dataset=trndata)
    cv = CrossValidator( trainer, tstdata, n_folds=5, valfunc=modval.MSE)
    if cv.validate() < 1:
        rmsearray.append(math.sqrt(cv.validate()))
        msearray.append(cv.validate())
#saving the MSE error array to an .npy file using numpy
msesavepath = str(labanout).replace('<','mse').replace('>','')
+str(labanhidden).replace('<','').replace('>','')
msesave = os.path.join(networkpath,msesavepath)
rmsesave = os.path.join(networkpath,"r" + msesavepath)
np.save(msesave,msearray)
np.save(rmsesave,rmsearray)
xaxis =np.arange(epochnumbers)

#plotting for MSE errors curve and saving the graph
from matplotlib import style
style.use('bmh')
plt.xlabel('epoch number')
plt.ylabel('MSE')
plt.title('MSE curve')
plt.legend()
```

```
        plt.grid(b= True, which='major', color='black', linestyle='--')
        plt.plot(xaxis,msearray)
        graphs = str(labanout).replace('<','').replace('>','')
+str(labanhidden).replace('<','').replace('>','.png')
        graphsave = os.path.join(networkpath,"mse" +graphs)
        plt.savefig(graphsave)
        plt.gcf().clear()
        p = net.activateOnDataset( ds )
        mymsetest =0
        xcc= 0
        pytsout =[]
        pytsreal= []
        for ml in input:
                tsout = net.activate(ml[4:8])
                tsreal = (ml[8:9])
                pytsout.append(tsout)
                pytsreal.append(tsreal)
                xcc = xcc + 1
                mymsetest= mymsetest + ((tsreal-tsout)*(tsreal-tsout))
        x =np.arange(xcc)
        mymsetest =mymsetest / xcc
        from numpy import mean, sqrt, square
        from sklearn.metrics import mean_squared_error
        ax=1
        skmse=[]
        skmse= ((np.array(pytsout) * np.array(pytsreal))**2).mean(axis=ax)
        #for ploting prediction vs expected output before rescaling the data
        from matplotlib import style
        style.use('seaborn-darkgrid')
```

```
plt.plot(x,pytsout, 'g', label='prediction')
plt.plot(x,pytsreal, 'b', label='expected output')
plt.xlabel('plot number')
plt.ylabel('sclaled stock price')
plt.title('predicted vs expected output')
plt.legend()
plt.grid(b= True, which='major', color='grey', linestyle='-')
#saving the plotted graph
graphsave = os.path.join(networkpath,"denormpred" +graphs)
plt.savefig(graphsave)
plt.gcf().clear()
#rescaling the data
inbtw = ((inmax - inmin)/(newmax-newmin))
pytsout =np.float32(pytsout)
pytsreal =np.float32(pytsreal)
pytsout = ((pytsout - newmin ) * inbtw) + inmin
pytsreal = ((pytsreal -newmin ) * inbtw) + inmin
msearraydif = tsreal-tsout
msearraydif =np.array(msearraydif)**2

#ploting the origanal data scale predction vs expected outpu
plt.plot(x,pytsout, 'g', label='denorm prediction')
plt.plot(x,pytsreal, 'b', label='denorm expected output')
plt.xlabel('plot number')
plt.ylabel('stock price')
plt.title('predicted vs expected output')
plt.legend()
plt.grid(b= True, which='major', color='red', linestyle='--')
#saving the plotted graph
```

```
        graphsave = os.path.join(networkpath,"normpred" +graphs)
        plt.savefig(graphsave)
        #saving neural network using newtwork writer
        from pybrain.tools.customxml import NetworkWriter
        from pybrain.tools.customxml import NetworkReader
        networksave = os.path.join(networkpath,str(acfun) + "nnsave.xml")
        print networksave
        NetworkWriter.writeToFile(net,networksave)
if __name__ == "__main__":
        apple()
```

Testing code

```
import os
from pybrain.tools.customxml import NetworkReader
import numpy as np
from StringIO import StringIO
import csv,sqlite3
#to open db get total number of rows and minus 150 to get rows you will loop through
conn = sqlite3.connect(r"G:\msc\afinalpr\code\pyb\music.sqlite3")
conn.text_factory=str
cur = conn.cursor()
stocks = 'kengenc'
allrowcount = cur.execute('SELECT COUNT(*) FROM ({stocks})'. format (stocks =stocks))
.fetchone()[0]
looprows = allrowcount - 150
#create csv and append data
scriptsloc = os.path.realpath(__file__)
(filepath, filename) =os.path.split(scriptsloc)
networkpath = os.path.join(filepath,"networksave")
try:
```

```
        os.makedirs(networkpath)
except OSError as exception:
        pass
xla = os.path.join(filepath, "csvtest.csv")
csvopen = open(xla,"a")
spamwriter = csv.writer(csvopen,delimiter=',', lineterminator='\n',
quoting=csv.QUOTE_MINIMAL)
#get the saved neural network
nrd =NetworkReader.readFrom(r"G:\msc\afinalpr\code\pyb\networksave\kengen\s-t-t-
s\7nnsave.xml")
#getting the max value and min value of stock and calculation the constant value for scaling
operation
newmax =1
newmin =0
inmax = float(cur.execute('SELECT MAX(VWAP) FROM ({stocks})'. format (stocks =stocks))
.fetchone()[0])
inmin =         float(cur.execute('SELECT MIN(VWAP) FROM ({stocks})'. format (stocks
=stocks)) .fetchone()[0])
inbtw = (newmax-newmin)/((inmax - inmin))
reinbtw = ((inmax - inmin)/(newmax-newmin))
#to loop while performing activation
actarr =[]
mydata =[]
while looprows < allrowcount-3:
        print looprows, allrowcount
        innerloop =0
        #get stock values from database to a single array
        while innerloop < 4:
                valuehold =     float(cur.execute('SELECT VWAP FROM ({stocks}) where sid =
({looprows})'. format (stocks =stocks, looprows =int(looprows))) .fetchone()[0])
                valuehold = ((valuehold - inmin) * inbtw)+newmin
```

```
                actarr.insert(len(actarr),valuehold)
                innerloop +=1
                looprows +=1
                valuehold =''
        #using the array values in the array to use the activation function and empty the array
        vc =nrd.activate(actarr)
        vc = float(((vc - newmin ) * reinbtw) + inmin)
        actarr =[]
        # fetching the date and expected value
        fifthhold =     cur.execute('SELECT VWAP FROM ({stocks}) where sid =
({looprows})'. format (stocks =stocks, looprows =int(looprows))) .fetchone()[0]
        fifthdate =     cur.execute('SELECT Date FROM ({stocks}) where sid = ({looprows})'.
format (stocks =stocks, looprows =int(looprows))) .fetchone()[0]
        #putting the date ,expected value and predicted value to a single array
        mydata.insert(len(mydata),fifthdate)
        mydata.insert(len(mydata),vc)
        mydata.insert(len(mydata),fifthhold)
        #saving the date ,expected values, and predicted value array to a .csv file
        spamwriter.writerow(mydata)
        mydata =[]
csvopen.close();
```

Appendix 7 source code of the web application

Model

```
from __future__ import unicode_literals
from django.db import models
import datetime from django.utils
#for creating the eabl stock table with the columns below
class eabl(models.Model):
        SID =models.IntegerField(default=0)
```

```
        Date = models.DateTimeField(auto_now_add=False)
        Open =models.CharField(max_length=100)
        High =models.CharField(max_length=100)
        Close =models.CharField(max_length=100)
        Close =models.CharField(max_length=100)
        VWAP =models.IntegerField(default=0)
        Adjusted_VWAP =models.IntegerField(default=0)
        Volume =models.CharField(max_length=255)
#for creating the equity stock table with the columns below
class equity(models.Model):
        SID =models.IntegerField(default=0)
        Date = models.DateTimeField(auto_now_add=False)
        Open =models.CharField(max_length=100)
        High =models.CharField(max_length=100)
        Close =models.CharField(max_length=100)
        Close =models.CharField(max_length=100)
        VWAP =models.IntegerField(default=0)
        Adjusted_VWAP =models.IntegerField(default=0)
        Volume =models.CharField(max_length=255)
#for creating the kengen stock table with the columns below
class kengen(models.Model):
        SID =models.IntegerField(default=0)
        Date = models.DateField(max_length=200)
        Open =models.CharField(max_length=100)
        High =models.CharField(max_length=100)
        Close =models.CharField(max_length=100)
        Close =models.CharField(max_length=100)
        VWAP =models.IntegerField(default=0)
        Adjusted_VWAP =models.IntegerField(default=0)
```

```
	Volume =models.CharField(max_length=255)
#for creating the nmg (nation media group) stock table with the columns below
class nmg(models.Model):
	SID =models.IntegerField(default=0)
	Date = models.DateField(max_length=200)
	Open =models.CharField(max_length=100)
	High =models.CharField(max_length=100)
	Close =models.CharField(max_length=100)
	Close =models.CharField(max_length=100)
	VWAP =models.IntegerField(default=0)
	Adjusted_VWAP =models.IntegerField(default=0)
	Volume =models.CharField(max_length=255)
#for creating the sasini stock table with the columns below
class sasini(models.Model):
	SID =models.IntegerField(default=0)
	Date = models.DateTimeField(auto_now_add=False)
	Open =models.CharField(max_length=100)
	High =models.CharField(max_length=100)
	Close =models.CharField(max_length=100)
	Close =models.CharField(max_length=100)
	VWAP =models.IntegerField(default=0)
	Adjusted_VWAP =models.IntegerField(default=0)
	Volume =models.CharField(max_length=255)
```

Controller

Settings

```
import os
# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
SECRET_KEY = 'mysecret'
```

```
DEBUG = False
ALLOWED_HOSTS = []
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
        'raw',
]
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
ROOT_URLCONF = 'rawtest.urls'
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
```

```
            'django.template.context_processors.request',
            'django.contrib.auth.context_processors.auth',
            'django.contrib.messages.context_processors.messages',
        ],
    },
  },
]
WSGI_APPLICATION = 'rawtest.wsgi.application'
# Database
DATABASES = {
  'default': {
    'ENGINE': 'django.db.backends.sqlite3',
    'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
  }
}
AUTH_PASSWORD_VALIDATORS = [
  {
    'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
  },
  {
    'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
  },
  {
    'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
  },
  {
    'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
  },
]
```

```
# Internationalization
# https://docs.djangoproject.com/en/1.11/topics/i18n/
LANGUAGE_CODE = 'en-us'
TIME_ZONE = 'UTC'
USE_I18N = True
USE_L10N = True
USE_TZ = True
# Static files (CSS, JavaScript, Images)
STATIC_URL = '/static/'
ROOT_PATH = os.path.dirname(__file__)
STATICFILES_DIR =[
        os.path.join(ROOT_PATH, "static"),
        ]
```

URLs

```
from django.conf.urls import url
from django.contrib import admin
from raw import views as v
from django.conf.urls.static import static
from django.conf import settings
urlpatterns = [
  url(r'^admin/', admin.site.urls),
        #returns JSON data for populating a HTML table
        url(r'^netsv/json/$', v.netjs, name='netjs'),
        #accepts an integer value that is used to call for json data of specific stock
        url(r'^netsv/json/(?P<stockid>\d+)/$', v.netjs, name='netjs'),
        #goes to view and gets html page and its instructtions when user enters the url
        url(r'^netsv/$', v.netsv, name='netsv'),
```

```
	#used to get os file path
] + static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)
```

View

```
# -*- coding: utf-8 -*-
from __future__ import unicode_literals
import os, sqlite3
from pybrain.tools.customxml import NetworkReader
import numpy as np
from StringIO import StringIO
from django.shortcuts import render,render_to_response
from .models import *
from django.db import connection
from django.http import JsonResponse
from django.conf import settings
# Create your views here.
ROOT_PATH = os.path.dirname(__file__)
dateload = np.load(os.path.join(ROOT_PATH, "templates","network","datesave.npy"))
def netsv(request):
	#connecting to database
	conn = sqlite3.connect(r"G:\msc\afinalpr\code\pyb\music.sqlite3")
	conn.text_factory=str
	cur = conn.cursor()
	stocktables =['eabl','equity','kengen','nmg','sasini']
	stocksdata=[]
	holda =[]
	stockfetchid=1
	#loop to open the different stock tables and get total number of rows
	for stocks in stocktables:
```

```
                allrowcount = cur.execute('SELECT COUNT(*) FROM ({stocks})'. format
(stocks =stocks)) .fetchone()[0]

                co=0

                holda.insert(len(holda),stocks)

                sname =['stockname','date', 'open', 'high', 'low', 'close' , 'vwap',
'volume','stockfetchid']

                #loop to get data from different stock tables and provide values for keys during
dictionary making

                while co <= 6:

                        getdate= cur.execute('SELECT date, open, high, low, close , vwap,
volume FROM ({stocks}) where sid =({rowcounter})'\

                                . format (stocks =stocks, rowcounter
=int(allrowcount))).fetchone()[co]

                        holda.insert(len(holda),getdate)

                        co =co+1

                #creating dictionay from with stock name, date, open, high, low, close , vwap,
volume as keys respectivly as values

                holda.insert(len(holda),stockfetchid)

                stockfetchid=stockfetchid+1

                mydc = dict(zip(sname,holda))

                stocksdata.insert(len(stocksdata),mydc)

                holda =[]

        stockfetchid=1

        conn.close()

        question =
{"sales":1,"customers":"laban","inc":1},{"sales":4,"customers":"kkk","inc":2},{"sales":6,"custo
mers":"sff","inc":3},{"sales":4,"customers":"sff","inc":4},{"sales":10,"customers":"sff","inc":4}

        return render_to_response('viewnet.html', {'stocksdata': stocksdata})

def netjs(request,stockid =1 ,*args, **kwargs):

        #normalization range

        newmax =0.5
```

```
        newmin =0

        # to load data stored in .npy numpy file and calculating minimum and maximum from
array for normalization

        cc = os.path.join(ROOT_PATH, "templates","network",str(stockid),"data.npy")

        eqdata = np.load(cc)

        consteqdata = eqdata

        inmax =max(eqdata)

        inmin =min(eqdata)

        inbtw = (newmax-newmin)/((inmax - inmin))

        normdata = ((eqdata - inmin) * inbtw)+newmin

        networktype =1

        #to loop through the activation functions i folder of selected stockid

        while networktype <= 5:

                #loading the saved network and using it to perform prediction from scaled data

                nrd = NetworkReader.readFrom(os.path.join(ROOT_PATH,
"templates","network",str(stockid),str(networktype) + "nnsave.xml"))

                vc =nrd.activate(normdata[-5:-1])

                #returning scaled output to original expected size and inserted at the end of array

                inbtw = ((inmax - inmin)/(newmax-newmin))

                pytsout =np.float32(vc)

                pytsout = ((pytsout - newmin ) * inbtw) + inmin

                eqdata= np.append(eqdata,pytsout)

                fnaldata =np.append(eqdata,pytsout)

                #loop increment

                networktype = networktype+1

        networktype =1

        #select last 10 values of date and 9 for stocks for creating dictionary

        numbersongraph =-10

        numbersongraphhold=abs(numbersongraph) + 5
```

```
fnal =consteqdata[numbersongraph +1:]
tsfhold =fnal
tsfhold=np.append(tsfhold,eqdata[-1])
tthold =fnal
tthold=np.append(tthold,eqdata[-2])
llhold =fnal
llhold=np.append(llhold,eqdata[-3])
sthold =fnal
sthold=np.append(sthold,eqdata[-4])
sshold =fnal
sshold=np.append(sshold,eqdata[-5])
#mm = dateload[numbersongraph:]
conn = sqlite3.connect(r"G:\msc\afinalpr\code\pyb\music.sqlite3")
conn.text_factory=str
cur = conn.cursor()
#to get stock name table name using stock id provided
stockname =['eablc','equityc','kengenc','nmgc','sasinic']
#stockid =1
#getting total number of rows from tables
allrowcount = cur.execute('SELECT COUNT(*) FROM ({stockname})'. format
(stockname = stockname[int(stockid)-1])).fetchone()[0]
#getting the last nine dates on db
co=abs(numbersongraph)-2
mm= []
while co >= 0:
    getdate= cur.execute('select date FROM ({stockname}) where sid =({daterow})'\
        . format (stockname = stockname[int(stockid)-1],daterow
=int(allrowcount-co))).fetchone()[0]
    mm.insert(len(mm),getdate)
```

```
            co =co-1
        #get the last date and add +1 days
        getdate= cur.execute('select date(Date, """+1 days""") from ({stocknameget}) where sid
=({rowcounter})'\
            . format (stocknameget = stockname[int(stockid)-1], rowcounter
=int(allrowcount))).fetchone()[0]
        mm.insert(len(mm),getdate)
        predloop = 0
        data =[]
        for stockdate in mm:
            dataholdkey =[]
            dataholdvalue =[]
            dataholdkey.append("stock price tsf")
            dataholdvalue.append(str(tsfhold[predloop]))
            dataholdkey.append("stock price tt")
            dataholdvalue.append(str(tthold[predloop]))
            dataholdkey.append("stock price ll")
            dataholdvalue.append(str(llhold[predloop]))
            dataholdkey.append("stock price st")
            dataholdvalue.append(str(sthold[predloop]))
            dataholdkey.append("stock price ss")
            dataholdvalue.append(str(sshold[predloop]))
            dataholdkey.append("date")
            dataholdvalue.append(stockdate)
            predloop = predloop +1
            mydc = dict(zip(dataholdkey,dataholdvalue))
            data.append(mydc)
        predloop =0
        return JsonResponse(data, safe=False)
```

HTML code

Navigation bar code

```
{% load staticfiles %}
<head>
   <title>UON MSC project</title>
   <!-- Bootstrap core CSS -->
   <link href="{% static "vendor/bootstrap/css/bootstrap.min.css" %}" rel="stylesheet">
   <!-- Custom fonts for this template -->
   <link href="{% static "vendor/font-awesome/css/font-awesome.min.css" %}" rel="stylesheet" type="text/css">
   <link href='https://fonts.googleapis.com/css?family=Open+Sans:300italic,400italic,600italic,700italic,800italic,400,300,600,700,800' rel='stylesheet' type='text/css'>
   <link href='https://fonts.googleapis.com/css?family=Merriweather:400,300,300italic,400italic,700,700italic,900,900italic' rel='stylesheet' type='text/css'>
   <!-- Plugin CSS -->
   <link href="{% static "vendor/magnific-popup/magnific-popup.css" %}" rel="stylesheet">
   <!-- Custom styles for this template -->
   <link href="{% static "css/creative.min.css" %}" rel="stylesheet">
   <link href="{% static "css/customtable.css" %}" rel="stylesheet">
   <link href="{% static "css/tabletwostyle.css" %}" rel="stylesheet">
  </head>  <body id="page-top">
   <nav class="navbar navbar-expand-lg navbar-light fixed-top" id="mainNav">
     <div class="container">
       <a class="navbar-brand js-scroll-trigger" href="#page-top">Home</a>
       <button class="navbar-toggler navbar-toggler-right" type="button" data-toggle="collapse" data-target="#navbarResponsive" aria-controls="navbarResponsive" aria-expanded="false" aria-label="Toggle navigation">
         <span class="navbar-toggler-icon"></span> </button>
```

```
    <div class="collapse navbar-collapse" id="navbarResponsive">
     <ul class="navbar-nav ml-auto">
      <li class="nav-item">
       <a class="nav-link js-scroll-trigger" href="#about">Stocks</a></li>
      <li class="nav-item">
       <a class="nav-link js-scroll-trigger" href="#services">Charts</a></li>
      <li class="nav-item">
       <a class="nav-link js-scroll-trigger" href="#portfolio">chart legend</a></li>
      <li class="nav-item">
       <a class="nav-link js-scroll-trigger" href="#contact">Contact</a></li></ul></div>
   </div></nav>
```

<u>Html body code</u>

```
{% load staticfiles %}
{% include "navbar.html" %}
<!DOCTYPE html>
<html lang="en">
<script src="{% static "js/jquery-3.2.1.min.js" %}"></script>
<!-- Theme chartjs -->
<script src="{% static "js/Chart.min.js" %}" />
<script>
var ctx = document.getElementById("myChart").getContext('2d');
var myChart = new Chart(ctx, {});
</script>
<header class="masthead"><div class="header-content"><div class="header-content-inner">
 <h1 id="homeHeading">NSE stock price prediction based on Artificial Neural Networks MSc project</h1>
  <hr><a class="btn btn-primary btn-xl js-scroll-trigger" href="#about">view stocks</a>
    </div></div></header><section class="bg-primary" id="about">
```

```
<div class="container" style="width:98%;"><div class="tbl-header"><table cellpadding="0" cellspacing="0" border="0" style="width: 100%">
<colgroup>
<col span="1" style ="width: 10%;"><col span="1" style ="width: 10%;">
<col span="1" style ="width: 10%;"><col span="1" style ="width: 10%;">
<col span="1" style ="width: 10%;"><col span="1" style ="width: 10%;">
<col span="1" style ="width: 10%;"><col span="1" style ="width: 10%;">
<col span="1" style ="width: 15%;">
</colgroup>
<thead>
<tr style="background-color: #777"><th >stock</th>
<th>date</th><th>open</th><th>high</th><th>low</th><th>close</th><th>vwap</th><th>volume</th><th>stockfetchid</th><?></tr>
</thead>  </table><div class="tbl-content"><table cellpadding="0" cellspacing="0" border="0" style="width:100%">
<colgroup>
<col span="1" style ="width: 10%;"><col span="1" style ="width: 12%;">
<col span="1" style ="width: 10%;"><col span="1" style ="width: 10%;">
<col span="1" style ="width: 10%;"><col span="1" style ="width: 10%;">
<col span="1" style ="width: 10%;"><col span="1" style ="width: 10%;">
<col span="1" style ="width: 20%;"></colgroup>
<tbody>
{% for dataloop in stocksdata %}
<tr><td>{{ dataloop.stockname }}</td><td>{{ dataloop.date }}</td><td>{{ dataloop.open }}</td>
<td>{{ dataloop.high }}</td><td>{{ dataloop.low }}</td><td>{{ dataloop.close }}</td>
<td>{{ dataloop.vwap }}</td><td>{{ dataloop.volume }}</td><td>
<a class="btn btn-primary btn-xl js-scroll-trigger" href="#services" onclick ="myHandler({{ dataloop.stockfetchid }})">view chart</a></div></td></tr>
{% endfor %}
```

```
</tbody></table></div></div>
   </section><section id="services"><div class="container" style="width:98%;">
<div id="chart_div"></div>
<div id="chartContainer" style="height: 70%; width: 100%;"> <canvas id="myChart"></canvas>
  </div><div id="show-data"></div></div> </section>
<script type="text/javascript">
url ='/netsv/json/'
var result ={name: 1}
         function myHandler(name) {
         $.ajax({
   type: 'GET',
   cache: false,
   data: {get_param:'value'},
         dataType:'json',
   url:"/netsv/json/"+name ,
   success:function(data){
                  Array.prototype.mapProperty = function(property) {
                  return this.map(function (obj) {
                  return obj[property];
                  });
                  };
<!-- var ctx = document.getElementById("myChart").getContext('2d'); -->
$('#myChart').remove();
$('iframe.chartjs-hidden-iframe').remove();
$('#chartContainer').append('<canvas id="myChart"><canvas>');
var ctx = $("#myChart");
var myChart = new Chart(ctx, {
   type: 'line',
   data: {
```

```
labels: data.mapProperty('date'),
datasets: [{
   label: '# T-SF-SF-T pred',
   data: data.mapProperty('stock price tsf'),
   backgroundColor: ['rgba(255, 99, 132, 0.2)'],
   borderColor: ['rgba(255,99,132,1)'],
   borderWidth: 1
                  },{
                  label: '# T-T-T-T pred',
   data: data.mapProperty('stock price tt'),
   backgroundColor: ["rgba(255,153,0,0.4)"],
   borderColor: ['rgba(255,153,0,1)'],
   borderWidth: 1
                  },{
                  label: '# L-L-L-L pred',
   data: data.mapProperty('stock price ll'),
   backgroundColor: ["rgba(255,153,0,0.4)"],
   borderColor: ['rgba(255,153,0,1)'],
   borderWidth: 1
                  },{
                  label: '# S-T-T-S pred',
   data: data.mapProperty('stock price st'),
   backgroundColor: ["rgba(255,153,0,0.4)"],
   borderColor: ['rgba(255,153,0,1)'],
   borderWidth: 1
                  },{
                  label: '# S-S-S-S pred',
   data: data.mapProperty('stock price ss'),
   backgroundColor: ["rgba(255,153,0,0.4)"],
```

```
          borderColor: ['rgba(255,153,0,1)'],
          borderWidth: 1
                                }] },

    options: {
        scales: {
            yAxes: [{
                ticks: {
                    beginAtZero:true
                } }] } }});} });};
window.onload = myHandler(1);</script>
    <section id="portfolio"><div class="call-to-action bg-dark">
      <div class="container text-center"><h2>chart legend</h2><table class="container2">
      <tr><td>T-SF-SF-T   </td> <td>hyperbolic tanget-softmax-softmax-hyperbolic tanget</td>
</tr>
      <tr><td>T-T-T-T   </td>   <td>   hyperbolic   tanget-hyperbolic   tanget-hyperbolic   tanget-
hyperbolic tanget</td></tr>
      <tr><td>L-L-L-L   </td>   <td>   linier-linier-linier-linier</td></tr><tr><td>S-T-T-S   </td>
<td>sigmoid-hyperbolic tanget-hyperbolic tanget-sigmoid</td></tr>
      <tr><td>S-S-S-S </td><td>sigmoid-sigmoid-sigmoid-sigmoid</td></tr>        </table><br/>
       <a class="btn btn-default btn-xl sr-button" href="#services">view chart</a>
      </div></div></section><br/>
    <section id="contact"> <div class="container">
        <div class="row"><div class="col-lg-8 mx-auto text-center">
          <h2 class="section-heading">Let's Get In Touch!</h2>
          <hr class="primary">
          <p></p></div></div><div class="row"><div class="col-lg-4 ml-auto text-center">
          <i class="fa fa-phone fa-3x sr-contact"></i><p>771-795-817</p></div>
         <div class="col-lg-4 mr-auto text-center"><i class="fa fa-envelope-o fa-3x sr-contact"></i>
          <p><a href="mailto:kklaban@gmail.com">email-feedback</a>
```

```html
        </p></div></div></div> </section>
    <!-- jQuery -->
    <!-- Bootstrap core JavaScript -->
    <script src="{% static "vendor/jquery/jquery.min.js" %}"></script>
    <script src="{% static "vendor/popper/popper.min.js" %}"></script>
    <script src="{% static "vendor/bootstrap/js/bootstrap.min.js" %}"></script>
    <!-- Plugin JavaScript -->
    <script src="{% static "vendor/jquery-easing/jquery.easing.min.js" %}"></script>
    <script src="{% static "vendor/scrollreveal/scrollreveal.min.js" %}"></script>
    <script src="{% static "vendor/magnific-popup/jquery.magnific-popup.min.js" %}"></script>
    <!-- Custom scripts for this template -->
    <script src="{% static "js/creative.min.js" %}"></script>
  </body>
  </html>
```