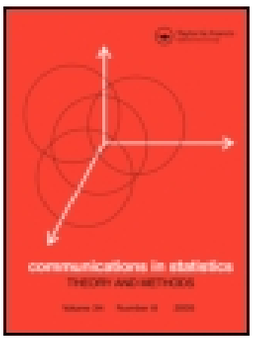

Taylor & Francis
Taylor & Francis Group

Communications in Statistics - Theory and Methods

ISSN: 0361-0926 (Print) 1532-415X (Online) Journal homepage: http://www.tandfonline.com/loi/lsta20

Back propagation neural network based big data analytics for a stock market challenge

V. P. Ramesh, Priyanga Baskaran, Aarthika Krishnamoorthy, Divya Damodaran & Preethi Sadasivam

To cite this article: V. P. Ramesh, Priyanga Baskaran, Aarthika Krishnamoorthy, Divya Damodaran & Preethi Sadasivam (2018): Back propagation neural network based big data analytics for a stock market challenge, Communications in Statistics - Theory and Methods

To link to this article: https://doi.org/10.1080/03610926.2018.1478103

Published online: 17 Nov 2018.

COMMUNICATIONS IN STATISTICS—THEORY AND METHODS
2018, VOL. 00, NO. 0, 1–21
https://doi.org/10.1080/03610926.2018.1478103

Back propagation neural network based big data analytics for a stock market challenge

V. P. Ramesh, Priyanga Baskaran, Aarthika Krishnamoorthy, Divya Damodaran, and Preethi Sadasivam

Mathematics and Analytics Center of Excellence (MACoE), School of Mathematics & Computer Sciences, Central University of Tamil Nadu, Thiruvarur, India

ABSTRACT

In this article we are presenting our methodology on solving a stock market challenge on predicting the intraday stock returns. We are presenting our complete approach on solving this challenge namely, the approaches to prepare the data from the unstructured data and the challenges on using back propagation neural network algorithm, namely the choice of activation function, learning rate and the number of neurons in the hidden layer. The validation of the approach is also presented demonstrating the effectiveness of back propagation neural network based model on predicting the stock returns. It was observed that the proposed algorithm was able to predict the stock returns with an maximum absolute error of 6×10^{-4} and therefore the prediction is very close to the actual value.

ARTICLE HISTORY
Received 7 January 2018
Accepted 8 May 2018

KEYWORDS
Big data; analytics; stock market; supervised machine learning; back propagation; artificial neural network

1. Introduction

Due to the exponential growth in the data, big data analytics has become one of the necessary technologies to gain insights from the digital media contents Parvathi and Rekha (2015), Jagadish et al. (2014), Sagiroglu and Sinanc (2013), Singh and Reddy (2014). In this article, we are proposing a machine learning algorithm for a big data analytics in stock market segment. Though big data analytics is a venerable technology, the exponential growth in computing power has created numerous applications of it in various engineering systems and thereby in many business sectors. Sagiroglu and Sinanc (2013), presents an overview of big data analytics and also touch upon the privacy issues. Singh and Reddy (2014), discuss a few available data processing platforms and also present the pros and cons of each of them. Parvathi and Rekha (2015), presents a survey of risk in big data analytics. Ramesh, Favas, and Joseph (2018) presents a detailed survey of big data analytics by summarizing multimodal information like journal articles, ventures funding's, startup companies, research labs etc. across the globe.

One of the key challenges in financial sector today is to predict the stock behavior and is also a venerable research area with numerous amount of literature. Benjamin, Dodd, and Cottle (1934) is one of the sources of today's stock market predictions and the motivation for investors or institutions to analyze stocks based on the trends. In fact many companies are making strategic move towards algorithms for analytics and high frequency trading by

CONTACT V. P. Ramesh vpramesh@gmail.com Mathematics and Analytics Center of Excellence (MACoE), School of Mathematics & Computer Sciences, Central University of Tamil Nadu, Thiruvarur 610005, Tamil Nadu, India.
Color versions of one or more of the figures in the article can be found online at www.tandfonline.com/lsta.

analyzing the big stock market data. Oztekin et al. (2016) presents the stock market prediction by using three machine learning algorithms namely, adaptive neuro-fuzzy inference systems, artificial neural network and support vector machines. Potthoff (2007) investigate the use of prediction markets to estimate the prior distributions for Bayesian inference. Atsalakis and Valavanis (2009) demonstrates a neuro-fuzzy adaptive based model predicting the next day's stock price. Xi et al. (2014) demonstrates the RBF neural network based algorithm to avoid over fitting of the neural network when jump discontinuity is present in the data. Enke, Grauer, and Mehdiyev (2011) presents the methodology to forecast the stock market price by using multiple regression and neural network based algorithms. Mo, Wang, and Niu (2016) presents a comparison study, to show the efficiency of the back propagation neural network algorithm for the stock market prediction by using exponential type of activation function.

In this paper we would like to present a methodology for real life big data problem in stock market segment. Predicting the stock behavior is one of the key analytics which every investor is interested in knowing to make profit. Though it's impossible to predict the stock prices accurately, the search for algorithms which can predict the stock prices reasonably closer is one of the key research area in stock market analytics. For further reading in this line reader may refer to Oztekin et al. (2016), Potthoff (2007), Atsalakis and Valavanis (2009), Robert and Peter F. (2016). Most of the activity in this segment is to work on large, noisy datasets to extract information that can help analyst to build successful trading systems for their clients.

The main objective of this article is to demonstrate the efficiency of the back propagation neural network based algorithm on predicting the intraday stock prices. The complete value chain of our analytics is presented with details of the methods and the process of reduction to practice. The other challenges namely the choice of activation function, learning rate and the number of neurons in the hidden layer are also explained. The computational complexity includes processing of 40,000 arbitrary stocks with 211 parameters to train and internally validate the model. An external validation of the model on a set of 1,20,000 arbitrary stocks were performed. All these computations were performed using cloud platform with Python.

1.1. Stock market challenge

This article is based on big data problem with main objective to predict the return of set of stocks, given historical stock performance of a few stocks. Given the sensitivity of stock market we are using the masked data for our analytics and prediction.

The dataset comprises of stocks behavior in 5-day time windows, namely D-2, D-1, D, D+1, and D+2. The stock returns were given in days D-2, D-1, and part of day D, and the aim was to to predict the returns in the rest of day D, and in days D+1 and D+2. During day D, the intraday returns for set of stock data were observed, which were the returns at different points of time in the day. They provided 180 minutes of data, from $t = 1$ to $t = 180$. In the training set we were given the full 180 minutes and in the test set just the first 120 minutes were provided. The aim here is to build a mathematical model based on the training set and validate the same using the test data.

The data was given in .csv format with 40,000 samples for the training set and as a measure of validation, test set was provided with 1,20,000 stocks to predict the returns based on the model developed. It was observed that 2,09,467 entries in the whole of training set data were missing, and reasonable number of outliers and wrong entries were observed in the data. The

methodology and various techniques used to solve this stock market challenge is presented in the following sections. For further techniques used to solve a stock market data the reader may refer to Cedric and Jacquier (2016), Tsay and Ando (2012), Dell'Aquila and Ronchetti (2006), Khansaa and Liginlalb (2011).

1.2. Back propagation neural network

Machine learning based analytics is a next-gen analytics technology in big data analytics. The recent trend in big data analytics is to create virtual human brain kind of a setup which can take the data as input and process it with a little support to help the organization to make automated or semi-automated analysis resulting to decisions almost in real time. This technology of creating an artificial human brain kind of setup is referred to us as Artificial Neural Network (ANN) Guresen, Kayakutlu, and Daim (2011), Nawi, Atomi, and Rehman (2013), Hill and Remus (1994), Wong, Bodnovich, and Selvi (1997). To make it very clear we are not talking about automated big data analytics which we guess is a future stage in the evolution of big data technologies. In other words, we could refer ANN as a semi-automated technology for big data analytics. Nowadays ANN is widely used in several business segments like Robotics, Aeronautics, Health care, Transportation, internet of things, media and communication, consumer & industrial, software etc. For further information on stock market prediction using ANN the readers may refer Kaastra and Boyd (1996), Bogaert, Ballings, and Poel (2016), Enke, Grauer, and Mehdiyev (2011), Xi et al. (2014), Mo, Wang, and Niu (2016), Giordano, Rocca, and Perna (2007), Maa et al. (2012). Multi-Layer Perceptron (MLP) is one of the very useful tools in artificial intelligence to induce a supervised learning system through the famous back propagation algorithm. The name is due to the back propagation of errors correcting the learnings. There will be forward propagation of inputs from one layer to another and the weights are fixed during the forward propagation of inputs. In this paper we have explained the complete end-to-end analytics of stock market data using back propagation neural network. For further reading on back propagation algorithm refer Haykin (1999), Piramuthu, Shaw and Gentry (1994).

1.3. Methodology

The big data analytics includes preparing the data, processing, visualizing, model development and validation. Our complete big data analytics on stock market challenge is divided into three phases, namely, understanding/preparing the data, model development and model validation. The value chain of our methodology is shown in Figure 1 with the list of challenges we faced and the tools used to conduct the analysis.

The complete experiments were conducted in open source environment R. The reduction to practice of all these challenges in various phases are demonstrated in the following sections.

2. Phase 1: Understanding/preparing the data

The purpose of this phase is to understand the data variables and prepare the data for model development and validation. In general the data provided in such big data analytics would be an unstructured one and therefore noise and errors are integral part of it. In this phase we devise strategies to identify and address these issues and create a structured data. Another

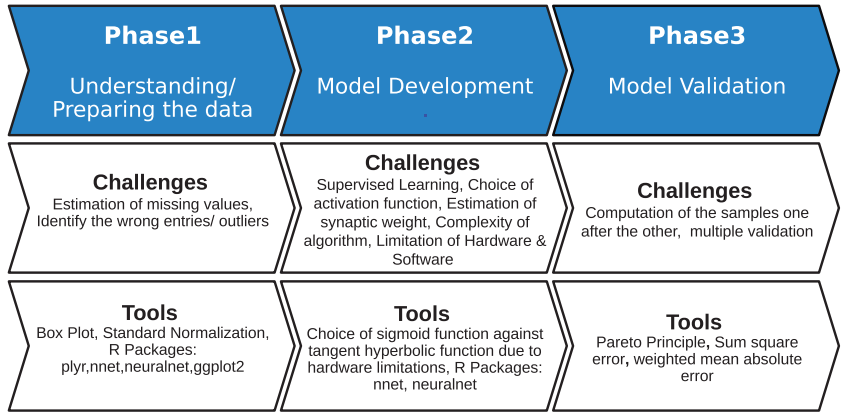

Figure 1. Value chain of our methodology.

important analysis we perform in this phase is to study the strength of correlation among the variables to identify the significant variables for the model development. We present a few important challenges and analysis on stock market data in this section.

2.1. 360° view of the data

The data were provided in two files, namely training set and test set which were in .csv format. The total size of the data was about 465.7MB (megabytes).

The training data was having information of 40,000 arbitrary stocks and 211 different information were given for all these 40,000 stocks. We would call this data as 211 dimensional data. These 211 dimensions includes the stock Id, 25 different features, stock returns *Ret_MinusTwo* & *Ret_MinusOne*, 179 dimensions were the stock returns at 179 different time stamps, next two dimensions were *Ret_PlusOne* & *Ret_PlusTwo* and the last two dimensions were the *Weight_Intraday* & *Weight_Daily*. These 25 feature variables were not mentioned specifically and however there was a disclaimer that these variables may not be useful in the analytics. The data variables and its description are given below

- *Feature_1 to Feature_25* : different features relevant to prediction
- *Ret_MinusTwo*: this is the return from the close of trading on day D-2 to the close of trading on day D-1 (i.e. 1 day)
- *Ret_MinusOne*: this is the return from the close of trading on day D-1 to the point at which the intraday returns start on day D (approximately 1/2 day)
- *Ret_2* to *Ret_120*: these are returns over approximately one minute on day D. *Ret_2* is the return between t = 1 and t = 2.
- *Ret_121* to *Ret_180*: intraday returns over approximately one minute on day D. These are the target variables you need to predict as id_1-60.
- *Ret_PlusOne*: this is the return from the time Ret_180 is measured on day D to the close of trading on day D+1. (approximately 1 day). This is a target variable you need to predict as id_61.
- *Weight_Intraday*: weight used to evaluate intraday return predictions Ret 121 to 180
- *Weight_Daily*: weight used to evaluate daily return predictions (*Ret_PlusOne* and *Ret_PlusTwo*).

The test file was comprised of 1,20,000 arbitrary stocks and the goal was to estimate the expected stock returns based on the developed model. The dimension of the test set was almost the same except the prediction variables namely, *Ret_121* to *Ret_180, Ret_PlusOne, Ret_PlusTwo* which were left empty to predict.

The understanding of this data and its variables were a challenge. Looks like due to confidentiality data provider had completely masked the twenty five features which made these features variables redundant. On the top of this there were huge number of missing entries, a reasonable number of wrong entries and outliers in the data. The variable *Feature_1* of the training set was observed to have the maximum number of missing values. It was estimated that 83.3% of 40,000 entries for *Feature_1* were missing. Preparing the data for analytics addressing the missing values and interpretation was a huge challenge and we have demonstrated our approach in the following section.

2.2. Significance of the dimensions: Feature_1 to Feature_25

As we discussed there were twenty five variables of every stock in the data without specifically mentioning the features and we assume that due to confidentiality and sensitivity of the stocks in the market, data provider had masked the identity of these variables. We wanted to analyze the significance of these variable in our developed model which will be discussed later. Here is a few univariate analysis on these variables and with this analysis we believe that these variables will have effects in our learning system and prediction. Due to space limitation we are presenting a few as an example for demonstration.

> summary(train$Feature_1)

Min.	*1st Qu.*	*Median*	*Mean*	*3rd Qu.*	*Max.*	*NA's*
1.00	*1.00*	*3.00*	*3.59*	*6.00*	*10.00*	*33313*

This variable, *Feature_1* has the maximum number of missing values, namely 33,313. It's clear from the above summary that the data is biased towards the lower side since 50% of the data are below 3 and the same may also be observed from the box plot in Figure 2. When we looked at the histogram in Figure 2, *Feature_1* is bias with values 1 and the rest of the values are occurring almost in the same frequency. It's also to be noted that any technique of estimating 83.3% of missing data from 16.7% of the given data will not give a good representation. Due to these analysis we decided to drop this variable. Let's look at another variable namely, *Feature_5*. *> summary(train$Feature_5)*

Min.	*1st Qu.*	*Median*	*Mean*	*3rd Qu.*	*Max.*
1.000	*2.000*	*6.000*	*5.483*	*8.000*	*10.000*

The variable, *Feature_5* has no missing values as see above and is also biased towards to the right side. The histogram in Figure 2 is giving a perception that this variable may have a significant impact in our proposed model due to its range of values.

Lets see two more variables namely *Feature_9* and *Feature_13*. Range of *Feature_9* is 1 to 36 and it has many outliers as you see in the box plot of the Figure 3. The range of *Feature_13* is 0 to 9 and seem to have uniform distribution. The sample seem to have almost uniform representation of stocks from every range which is evident from the histogram in the Figure 3.

> summary(train$Feature_9)

Min.	*1st Qu.*	*Median*	*Mean*	*3rd Qu.*	*Max.*	*NA's*
0.00	*9.00*	*11.00*	*10.68*	*12.00*	*36.00*	*1875*

> summary(train$Feature_13)

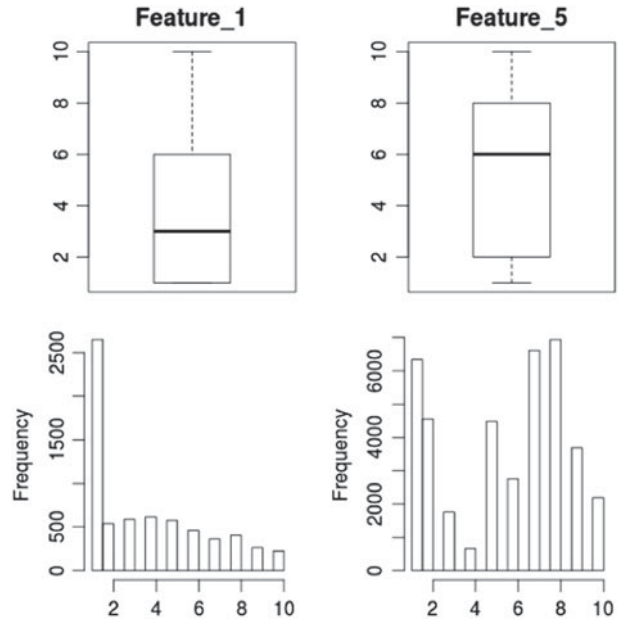

Figure 2. Box plot and histogram of variables Feature_1 and Feature_5.

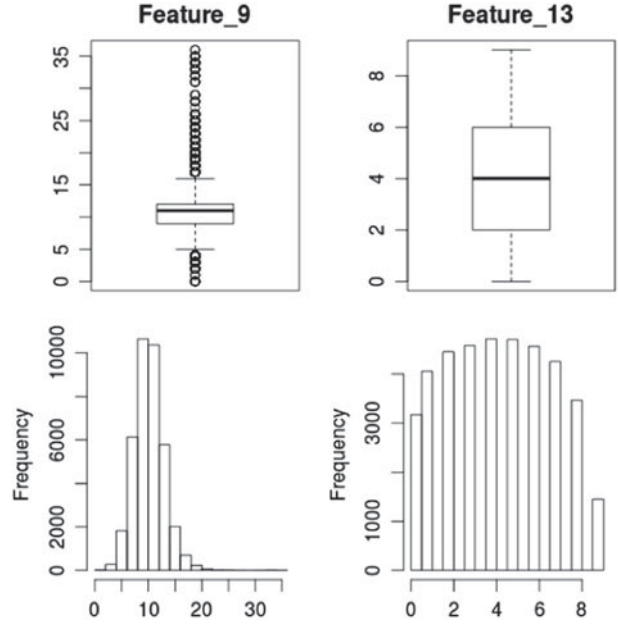

Figure 3. Box plot and histogram of features Feature_9 and Feature_13.

Min.	*1st Qu.*	*Median*	*Mean*	*3rd Qu.*	*Max.*	*NA's*
0.000	*2.000*	*4.000*	*4.238*	*6.000*	*9.000*	*594*

Similar analysis was performed on every variables and based on the analysis, it was decided that we will keep the variables such as *Feature_2* to *Feature_25* and drop the *Feature_1*.

Figure 4. Distribution of number of missing values of various variables namely 25 features and 183 Stock Returns.

2.3. Estimation of missing values and normalization

The training set had 211 variables and 40,000 arbitrary stocks as samples, hence totally 84,40,000 (211 × 40,000) values. Out of which 2,09,467 (2.5% of the total) values were missing. As discussed before, among the missing values about 0.4% (33,313) were of the variable *Feature_1* which is a dropped variable.

The Figure 4 shows the number of missing values for 208 dimensions of the data. The x axis was not label due to space constraints here. The x variables were in the following order, first 25 feature variables and then the remaining 183 stock returns. The distribution of missing values among the variables is clear from the Figure 4. It's also to be observed that *Feature_5* and *Feature_7* are not having any missing values. Similarly the variables *Ret_120* to *Ret_180*, *Ret_PlusOne* and *Ret_PlusTwo* are not having any missing values. So the data had 60+ variables which are error free and a good news for model development.

Now the challenge is to find a tool to estimate the remaining missing values, to be more precise we need to estimate 1,76,154 (2.1% of the total) values. We estimated the missing values and prepared the data using averaging technique.

In our data we had two types of attributes namely *Feature* and *Ret*. There were 25 arbitrary and masked *Features* and 183 *Ret* which were stock returns at different time line. All the 183 returns were in the range of $[-1, 1]$ whereas the feature variables were no so. So we normalized the feature values of the data to the interval $[-1, 1]$. If we use the data without normalizing it, then it may lead to useless results or it may lead to difficult learning process in our algorithm which is based on artificial neural network. For example during our experiments when we developed the model without normalizing the data, the learning process was complete messy and the algorithm ended up with as constant output on predicting the various stocks. In other words the learning part of the algorithm was unable to differentiate the different stocks and hence the prediction was same for every stock. Hence all these dimensions of the data were supposed to be normalized for uniformity and to have a better learning system. However the returns of the stocks were already normalized and this may again due to the confidentiality of the identity of stocks.

2.4. Preparing the data for model building

While considering the normalized stock returns (183 dimensions) the Figure 5 shows the distribution plots of *Ret_MinusOne, Ret_MinusTwo, Ret_2, Ret_22, Ret_130, Ret_150, Ret_PlusOne* and *Ret_PlusTwo*.

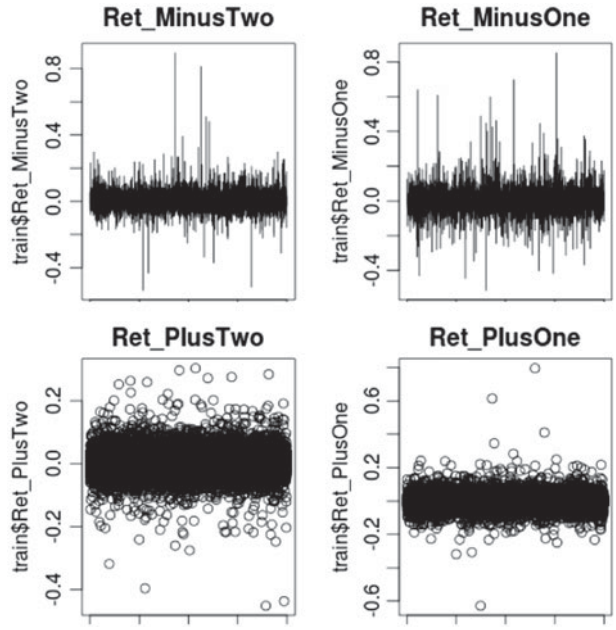

Figure 5. Distribution plot.

For the purpose of demonstration we have studied the univariate distribution of a few significant variables out of the 183 stock returns and given the same in Figure 5. From the plots it's clear that the distribution of the stock returns for these variables *Ret_MinusTwo, Ret_MinusOne, Ret_PlusTwo* and *Ret_PlusOne* were in the range from −0.2 to 0.2, similarly all other variables (183 stock returns) were in the interval −1 to +1 which confirmed that the given data were normalized. This is how we prepared the data for our neural net model building. We used Pareto principle, namely 80–20 rule for partitioning the data for building the model and validating the same. In this next section, we have demonstrated the model development using this prepared data and as discussed, we developed the model with 80% of the data and validated it with the remaining 20% of the data. One of the CTQs while developing neural network based learning system is to build and validated the model using multiple partitions and we will discuss about the same in the validation section.

According to us, in this data we didn't find any avoidable variables. The remaining variables represent the arbitrary stocks observed for every minute of a trading day for three hours and also the stock returns from the closure of two previous days, next two days respectively. So we didn't choose any particular variable as significant variable. The only eliminated variable in this data was *Feature_1*. In this data, 62 dependent variables were to predict for each arbitrary stock.

3. Phase 2: Model development

This is the second part of the overall execution. Now we have the data in a suitable format where we can built the model. Just to recall, the key activities in phase 1 includes estimation of the missing values, identification of significant variables and also normalizing the variables. In

Table 1. Variables used in back propagation algorithm.

Variables	Description
w	Weight
δ	Local gradient of neuron
ϕ	Activation function
η	Learning rate
d	Desired output
z	Actual output
ϵ	Tolerance of error

this phase we aim to develop a supervised learning system using artificial neural network and to be more precise we used back propagation algorithm. The aim of this phase is two folds, (i) We would like to demonstrate the reduction to practice of back propagation algorithm based learning system using the open source software R and (ii) we would like to solve the stock market data using the same approach. For further interpretation about the algorithm the reader may refer to Intrator and Intrator (2001), Rocca and Perna (2005), Li, Lin, and Peng (2014).

3.1. Design and analysis of back propagation algorithm

As seen in the introduction, Multi-Layer Perceptron (MLP) is one of the very useful tools in artificial intelligence to induce a supervised learning system through the famous back propagation algorithm. Refer to Figure 6 for an illustration of directional information flow in an MLP. The error get propagated backward (error = desired output − actual output) and the weights are adjusted based on some error correction methods so that the learning is better. This is an example of a supervised learning system and is referred to us as back propagation learning system.

We are presenting our design of back propagation algorithm based on which we have solved the stock market data of estimating the stock returns of about 1,20,000 samples in the test data. Table 1 list the variables and the description of the variables used in the back propagation algorithm. The algorithm is presented here

BPNN 1: [Input]

X - an l tuble element $(x_1, x_2, ...x_i, ...x_l)$

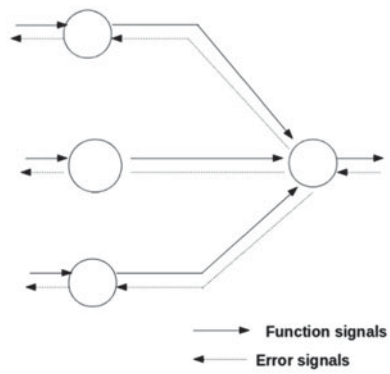

Figure 6. Demonstration of forward propagation of input and back propagation of error in a MLP.

x_i - an array of length n
D - an m tuble element $(d_1, d_2, ...d_k, ...d_m)$
d_k - an array of length n
ϕ - an array of activation function of length $2m$
ϵ - positive real number
η - a real number belonging to [0,1]
BPNN 2: [Initialise]
p $\leftarrow 1$; $x_0 \leftarrow 1$; $y_0 \leftarrow 1$
for $j \leftarrow 1$ to m **do**
  for $i \leftarrow 0$ to l **do**
    $w_{ji}(p) \leftarrow rand\big([-1, 1]\big)$
  end for
end for
for $k \leftarrow 1$ to m **do**
  for $j \leftarrow 0$ to m **do**
    $w_{kj}(p) \leftarrow rand\big([-1, 1]\big)$
  end for
end for
BPNN 3: [Normalisation]
for $i \leftarrow 1$ to l **do**
  for $q \leftarrow 1$ to n **do**
    $x_{iq} \leftarrow \frac{x_{iq}}{\sum x_{iq}}$
  end for
end for
BPNN 4: [Linear combination]
for $j \leftarrow 1$ to m **do**
  for $i \leftarrow 0$ to l **do**
    $v_j(p) \leftarrow \sum x_i w_{ji}(p)$
    $y_j(p) \leftarrow \phi_j(v_j(p))$ # Activation function
  end for
end for
BPNN 5: [Activation function]
for $k \leftarrow 1$ to m **do**
  for $j \leftarrow 0$ to m **do**
    $v_k(p) \leftarrow \sum y_j(p) w_{kj}(p)$ # linear combination of y_j
    $z_k(p) \leftarrow \phi_k(v_k(p))$ # Activation function
  end for
end for
BPNN 5: [Tolerance check]
for $k \leftarrow 1$ to m **do**
  $e_k(p) \leftarrow \big|d_k - z_k(p)\big|$
end for
BPNN 6: [Terminate]
if $\big|e_k(p)\big|_{k=1}^{m} < \epsilon$ **then**
  return z_k
end if

BPNN 7: [Local gradient - kth neuron of output layer]
for $k \leftarrow 1$ to m **do**
$\delta_k(p) \leftarrow e_k(p)\phi'_k(v_k(p))$
end for
BPNN 8: [Weights update - ouput layer]
for $k \leftarrow 1$ to m **do**
for $j \leftarrow 0$ to m **do**
$w_{kj}(p+1) \leftarrow \eta\delta_k(p)y_j(p)$
end for
end for
BPNN 9: [Local gradient - jth neuron of hidden layer]
for $j \leftarrow 1$ to m **do**
for $k \leftarrow 1$ to m **do**
$\delta_j(p) \leftarrow \phi'_j(v_j(p)) \sum \delta_k(p)w_{kj}(p)$
end for
end for
BPNN 10: [Weight update - input layer]
for $j \leftarrow 1$ to m **do**
for $i \leftarrow 0$ to l **do**
$w_{ji}(p+1) \leftarrow \eta\delta_j(p)x_i$
end for
end for
BPNN 11: [Increase]
$p \leftarrow p+1$
go to BPNN 4

3.2. Partitioning the data using Pareto principle

Pareto principle is also known as 80/20 rule or Pareto rule or Pareto law. The principle is named in order to honor Vilfredo Pareto. In order to build and evaluate a model it is necessary to split the data in to two sets, one to build the model and other to internally validate the model. Pareto principle is the main motivation to split the data in to 80–20. For further details about the Pareto principle the readers are suggested to read Ramesh et al. (2016). In our approach, we split the training set into two subsets one containing 80% of the data to build the model based on back propagation algorithm and other set containing 20% data to internally validate our model. We partitioned the data randomly into 80 and 20, respectively.

3.3. Learning model: A demo

Our model building was done using the package called *neuralnet* which is specially developed for various algorithms of neural network in R. For demonstration purpose and for better understanding we would like to show how this idea can be reduced to practice using R. For simplicity we restrict our self to only 10 variables, namely *Features_2* to *Features_11* and with 1 hidden layer having two neurons. We also assume to have one output neuron namely, *Ret_121*.

>library(neuralnet) >f1 = Ret_121 ~ Feature2 + Feature3 + Feature4 + Feature5 + Feature6 + Feature7 + Feature8 + Feature9 + Feature10 + Feature11 > modeldemo < − neuralnet

(formula = f1, data = train80, hidden = 2, threshold = 0.01, algorithm = "backprop", learningrate = 1e-05, err.fct = "sse", linear.output = TRUE)

>plot(modeldemo) We added the package *neuralnet* into the library in the first line. Second line was to declare the input and output variables. The function *neuralnet()* in the package, was to develop the model using the input and output as defined in the formula (f1) in the previous step. The command *data = train80* takes the data from the file *"train80"* which will be placed in the working directory. The *train80* was the output of the partition that we discussed in the previous subsection. The command *threshold = 0.01* was the difference between the desired output and the computed output. The command *algorithm = "backprop"* was for the choice of the algorithm and here we were using back propagation algorithm. The command *learningrate = 1e-05* was the learning rate of the back propagation algorithm which was denoted by η. The command *err.fct = "sse"* was the measure for computing the error namely sum square error(sse). Here we had used the default activation function namely sigmoid function. If we were to use a specific activation function then use the command, *act.fct = "function name"*. And *linear.output = TRUE* was to tell the model to apply the activation function only for the neurons of the hidden layer(s), which means the activation function for output neurons would be the identity function. The *linear.output = FALSE* refers to use the same activation function for all the neurons in both the hidden and output layers. The initial weight for the training was taken from the standard normal distribution which was the usual procedure in many neural network algorithms.

In the Figure 7 we have shown the visual of the model which was the output of the above computation with 10 inputs, 1 hidden layer with 2 neurons and one output neuron *Ret_121*. The synaptic weights shown on the edges were computed to get the desired output with an error = 0.018897. The model took 34 iterations to predict *Ret_121*. This computation took roughly 3 minutes in an i3 processor with 4GB RAM. It's to be noted that in the actual model there were predictions which took more than 24 hours.

3.4. Learning model: Back propagation algorithm on 80% of the data

We present the complete model development for the stock market challenge. The procedure is almost the same as we did in the demo.

The overall aim here was to develop a model based on 145 significant variables namely, *Feature_2* to *Feature_25* and stock returns *Ret_MinusOne*, *Ret _MinusTwo*, *Ret_2* to *Ret_120* (i.e., the stock returns up to 120th minute of current trading day). The logic of excluding the independent variables *Ret_121* to *Ret_180*, *Ret_PlusOne* and *Ret_PlusTwo* was because of the fact that in our test data (which is part of model validation) we will be predicting the stock returns from 121st minute of some trading day and then so on to 180th minute. So while iteratively predicting the stock returns at n^{th} minute(dependent variable) the algorithm considers the stock returns until $n - 1^{\text{th}}$ minute as the independent variables and compute the dependent variable.

The number of neurons in the hidden layer is roughly expected to be between 50–300% of the number of inputs. Here we opt for 139 neurons in the hidden layer which was within the expected range (139 neurons out of 146 inputs which is 105% of number of inputs).

Like we did in the demo case, the weight update was done by back propagation algorithm with a learning rate of 0.00001 which was chosen by trial and error method. Whenever the learning rate was greater than 10^{-5}, model was not close to the desired output i.e., the

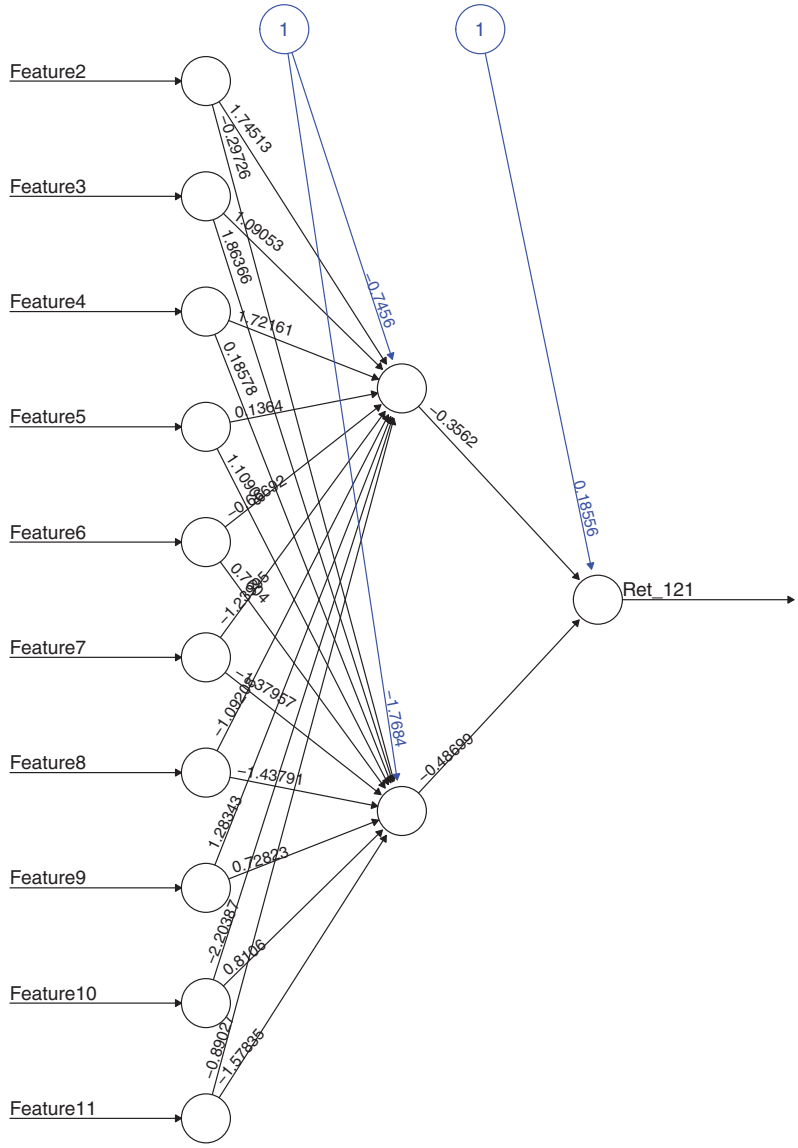

Figure 7. A sample network with 10 inputs, 1 hidden layer with 2 neurons and one output neuron *Ret_121*. The synaptic weights shown on the edges were computed to get the desired output with error = 0.018897 in 34 steps (iterations).

prediction was over fitting. The error tolerance that was the threshold used was 0.01 as a stopping criteria. The activation function applied to the hidden layer was sigmoid function which was the default function given in this package, we chose this because the computational time taken with other activation function namely "tanh" was too high and was not practically feasible. We have experimented with the activation function "tanh" and conformed that it was not practically feasible. We give the step by step instructions used in R for building the model. *>f1=Ret_121 Feature2 + Feature3 + Feature4 + Feature5 + Feature6 + Feature7 + Feature8 + Feature9 + Feature10 + Feature11 + Feature12 + Feature13 + Feature14 +*

Feature15 + Feature16 + Feature17 + Feature18 + Feature19 + Feature20 + Feature21 + Feature22 + Feature23 + Feature24 + Feature25 + Ret_MinusTwo + Ret_MinusOne + Ret_2 + RDirac1953888et_3 + Ret_4 + Ret_5 + Ret_6 + Ret_7 + Ret_8 + Ret_9 + Ret_10 + Ret_11 + Ret_12 + Ret_13 + Ret_14 + Ret_15 + Ret_16 + Ret_17 + Ret_18 + Ret_19 + Ret_20 + Ret_21 + Ret_22 + Ret_23 + Ret_24 + Ret_25 + Ret_26 + Ret_27 + Ret_28 + Ret_29 + Ret_30 + Ret_31 + Ret_32 + Ret_33 + Ret_34 + Ret_35 + Ret_36 + Ret_37 + Ret_38 + Ret_39 + Ret_40 + Ret_41 + Ret_42 + Ret_43 + Ret_44 + Ret_45 + Ret_46 + Ret_47 + Ret_48 + Ret_49 + Ret_50 + Ret_51 + Ret_52 + Ret_53 + Ret_54 + Ret_55 + Ret_56 + Ret_57 + Ret_58 + Ret_59 + Ret_60 + Ret_61 + Ret_62 + Ret_63 + Ret_64 + Ret_65 + Ret_66 + Ret_67 + Ret_68 + Ret_69 + Ret_70 + Ret_71 + Ret_72 + Ret_73 + Ret_74 + Ret_75 + Ret_76 + Ret_77 + Ret_78 + Ret_79 + Ret_80 + Ret_81 + Ret_82 + Ret_83 + Ret_84 + Ret_85 + Ret_86 + Ret_87 + Ret_88 + Ret_89 + Ret_90 + Ret_91 + Ret_92 + Ret_93 + Ret_94 + Ret_95 + Ret_96 + Ret_97 + Ret_98 + Ret_99 + Ret_100 + Ret_101 + Ret_102 + Ret_103 + Ret_104 + Ret_105 + Ret_106 + Ret_107 + Ret_108 + Ret_109 + Ret_110 + Ret_111 + Ret_112 + Ret_113 + Ret_114 + Ret_115 + Ret_116 + Ret_117 + Ret_118 + Ret_119 + Ret_120 > model1121 < − neuralnet(formula = f1, data = Fresh80, hidden = 139, threshold = 0.01, learningrate = 1e-05, algorithm = "backprop", err.fct = "sse", linear.output = TRUE)

> *model*121

```
1 repetition was calculated.

       Error Reached Threshold Steps
1 0.02340935       0.008299158    52
```

> summary(model1121)

```
> summary(model1121)
                   Length  Class      Mode
call                    9 -none-     call
response            32000 -none-     numeric
covariate         4640000 -none-     numeric
model.list              2 -none-     list
err.fct                 1 -none-     function
act.fct                 1 -none-     function
linear.output           1 -none-     logical
data                  209 data.frame list
net.result              1 -none-     list
weights                 1 -none-     list
startweights            1 -none-     list
generalized.weights     1 -none-     list
result.matrix       20437 -none-     numeric
```

In the summary, length of the *call* corresponds to the number of calling functions used in model building, namely formula, data, hidden, threshold, learningrate, algorithm, err.fct, linear.output and act.fct. The length of the response was 32,000 which corresponds to the number of samples in the data, namely "train80". The length of covariate was the number

values that goes on model building, i.e., 145 times 32,000 = 46,40,000. And length of the *data* corresponds to number of dimension in the data which was 209, namely 24 features, 183 stock returns, weight intraday and weight daily. The length of *result.matrix* was 20437 which was an one dimension output with step size, followed by error, the reached threshold value and then synaptic weights of the edges connecting the neurons namely, 139 neurons in the hidden layer connected to 146 inputs which creates 20,294 ($= 146 \times 139$) edges and 140 edges from the 139 neurons in the hidden layer and a bias connecting the output, hence $1+1+1+20, 294+140 = 20437$.

We have used this model to predict *Ret_121* as follow. The predicted values of the first few stocks were displayed as a sample. Similarly we can conduct the prediction for other variables like *Ret_122*, *Ret_123*, etc.,

[,1]
[1,] 6.879269e-05
[2,] −7.072335e-04
[3,] −2.538313e-05
[4,] 1.943846e-04
[5,] 1.033096e-04
[6,] 9.362994e-05

3.5. Generalized weights–learning indicator

The key challenge in any machine learning algorithm is to measure the learning, in other words the patterns acquired by the machine from the data. One way to measure the learning and to analyze the impact of any variable on the model is done by analyzing the weights that goes behind the variables while predicting a new dependent variable. It's to be noted that we are talking about two things (i) learning indicator and (ii) dependency / impact of an independent variable while computing a dependent variable.

For demonstration purpose let's take the variable *Feature_2* while predicting *Ret_121*. A simple univariate analysis on *Feature_2* before normalization gives the following,

> summary(Newfeaturesfile$Feature2)

Min.	*1st Qu.*	*Median*	*Mean*	*3rd Qu.*	*Max.*
–3.44100	*–0.79170*	*–0.09068*	*–0.11140*	*0.09154*	*4.17500*

The Figure 8 shows generalized weights against the normalized *Feature_2* values used for predicting *Ret_121* in the training set. A univariate on the generalized weights below shows that the minimum weight was –3,18,600 and the maximum value was 2,66,600 and the same can be seen in the Figure 8 as an outlier. There were few outliers in terms of these weights and we say that this features was having a high impact in the model prediction and has also challenged the learning process of the machine.

> summary(gendralizedweight1121$Feature2)

Min.	*1st Qu.*	*Median*	*Mean*	*3rd Qu.*	*Max.*
–318600.0	*–45.1*	*–13.2*	*–8.0*	*29.5*	*266600.0*

A simple univariate analysis on *Feature_13* before normalization gives the following,

> summary(Newfeaturesfile$Feature13)

Min.	*1st Qu.*	*Median*	*Mean*	*3rd Qu.*	*Max.*
0.000	*2.000*	*4.000*	*4.237*	*6.000*	*9.000*

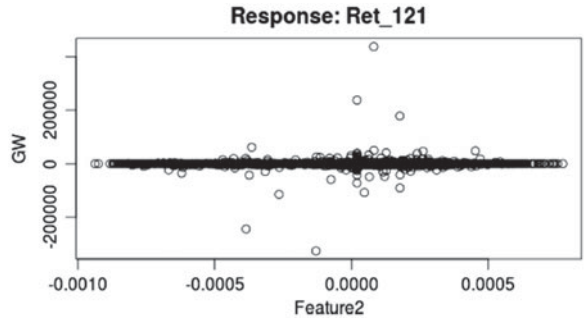

Figure 8. Generalized weights of *Feature_2* on predicting *Ret_121*.

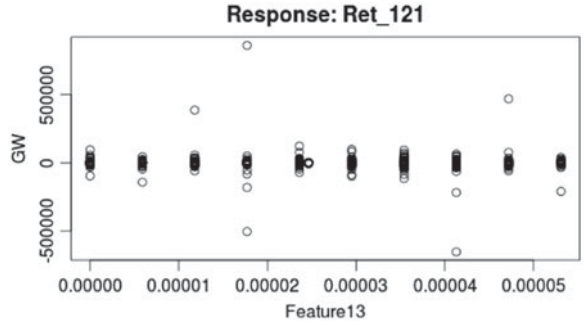

Figure 9. Generalized weights of *Feature_13* on predicting *Ret_121*.

The Figure 9 shows generalized weights against the normalized *Feature_13* values used for predicting *Ret_121* in the training set. A univariate on the generalized weights below shows that the minimum weight was –9,60,100 and the maximum value was 13,56,000 and the same can be seen in the Figure 9 as an outlier. It's to be observed from the Figure 9 that there generalized weights are different for a particular value of the *Feature_13*, thereby showing a relation in the figure. There were few outliers in terms of these weights and we say that this features was having a high impact in the model prediction and has also challenged the learning process of the model.

> summary(gendralizedweight1121$Feature13)

Min.	*1st Qu.*	*Median*	*Mean*	*3rd Qu.*	*Max.*
–960100.0	*–116.0*	*48.0*	*33.9*	*167.9*	*11356000.0*

For sake of completeness we would like the present the same analysis for one another variable namely, *Feature_22*. A simple univariate analysis on *Feature_22* before normalization gives the following,

> summary(Newfeaturesfile$Feature22)

Min.	*1st Qu.*	*Median*	*Mean*	*3rd Qu.*	*Max.*
–5.8200	*–1.7460*	*–0.7471*	*–0.7722*	*0.2485*	*2.2850*

The Figure 10 shows generalized weights against the normalized *Feature_22* values used for predicting *Ret_121* in the training set. A univariate on the generalized weights below shows that the minimum weight was –21,01,000 and the maximum value was 28,96,000 and the same can be seen in the Figure 10 as an outlier. It's to be observed from the Figure 10 that the generalized weights are zero for most of the values of the variable. There were few outliers

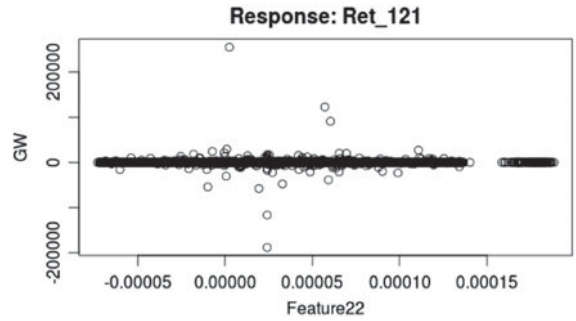

Figure 10. Generalized weights of *Feature_22* on predicting *Ret_121*.

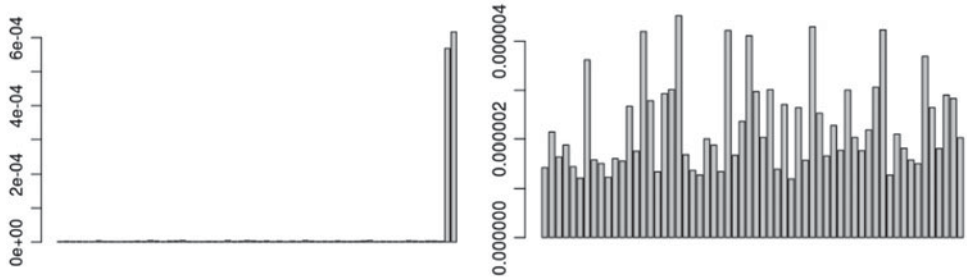

Figure 11. Estimated mean square error (MSE) for the predicted values (a) *Ret_121* to *Ret_180*, *Ret_PlusOne*, *Ret_PlusTwo* (b) *Ret_121* to *Ret_180*.

in terms of these weights and we say that this variable was having a reasonable impact in the model prediction and has also reasonably challenged the learning process of the machine.

> summary(generalizedweight1121$Feature22)

Min.	*1st Qu.*	*Median*	*Mean*	*3rd Qu.*	*Max.*
–2101000.0	*–232.8*	*100.7*	*71.3*	*345.4*	*2896000.0*

A similar analysis was carried out for all the variables and it was found that generalized weights are almost similar to one of these three variables demonstrated here and hence we have demonstrated using these variables namely *Feature_2*, *Feature_13* and *Feature_22*.

4. Phase 3: Model validation

In this phase, we present the model validation and also demonstrate the efficiency of the proposed algorithm on predicting the stock prices. One of the highly adapted validation techniques in big data analytics is the internal validation, which refers to the technique of internally keeping a portion of the data for validation. As we discussed in the previous phase, we have already kept aside 20% of training data to internally validate the model. We run the model on this 8000 stocks predicting the variables like *Ret_121* to *Ret_180*, *Ret_PlusOne*, *Ret_PlusTwo*.

We are using mean square technique to estimate the errors and are shown in the Figure 11. It's very clear from Figure 11a that the predicted stock prices are close to the actual stock prices except for the *Ret_PlusOne* and *Ret_PlusTwo* which are not the intraday stock prices

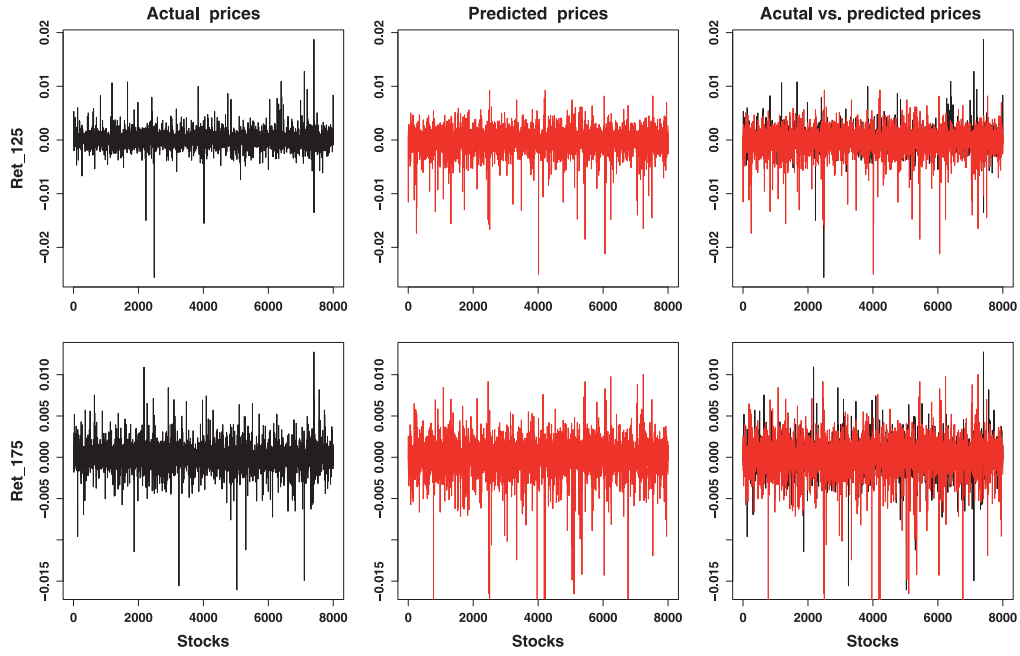

Figure 12. Actual and predicted stock prices of *Ret_125* and *Ret_175*.

but the stock price on the next and the day after it. The Figure 11b shows the error for the intraday stocks prices of *Ret_121* to *Ret_180* and its clear from the estimated errors that the predictions are very close to the actual stock prices. It's to be noted that the maximum error of the intraday stock prices is 4×10^{-6} and hence we could claim that the proposed algorithm works well for stock prices of intraday. Similarly, the maximum error of the stock prices for the next day and day after it is 6×10^{-4} which is also reasonable. The weighted mean absolute error that we got in this model was 14396.5 this is because the weights were given in lakhs in the *Weight_Intraday* and *Weight_Daily* columns.

To demonstrate the efficiency of the prediction we have shown in Figure 12 comparing the predicted stock prices with the actual prices of the dependent variables *Ret_125* and *Ret_175*. Further to test a few more dependent variables, Figure 13 shows the comparision of few more dependent variables namely, *Ret_121*, *Ret_131*, *Ret_141*, *Ret_151*, *Ret_161*, *Ret_171* and *Ret_180*. It's clear from these graphs that the proposed algorithm works well on capturing the intraday stock prices namely the dependent variables *Ret_121* to *Ret_180*.

Now, Figure 14 shows the comparison of predicted stock prices of the variables *Ret_PlusOne* and *Ret_PlusTwo*. As discussed before during the maximum error (MSE) this method works reaonable well for most of the stocks. From the Figure 14 we can observe that prediction of the stock prices with large time gap is difficult task than the intraday stock prices.

One of the CTQs given in the challenge was to predict the stock returns of a test set of data having 1,20,000 arbitrary stocks. We performed the prediction on the variables *Ret_121* to *Ret_180*, *Ret_PlusOne*, *Ret_PlusTwo* for these 1,20,000 arbitrary stocks. By this we conclude our validation of the developed model.

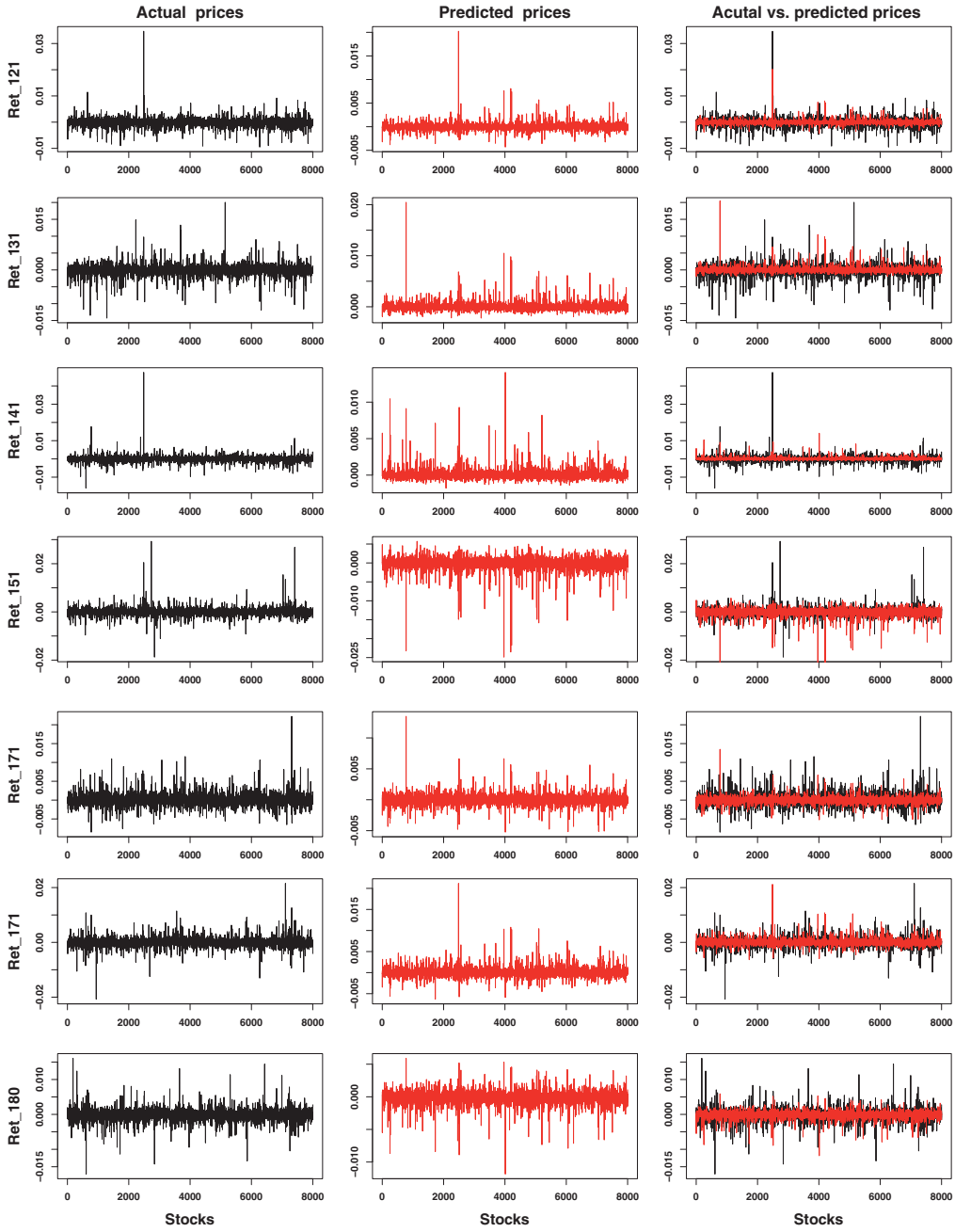

Figure 13. Actual and predicted stock price of *Ret_121*, *Ret_131*,*Ret_141*,*Ret_151*, *Ret_161*, *Ret_171* and *Ret_180*.

5. Conclusion

Here we have presented the demonstration of artificial neural network method for a real life problem. The stock market challenge of predicting stock returns was solved using back propagation algorithm and was demonstrated phase by phase in this article. It is also to be

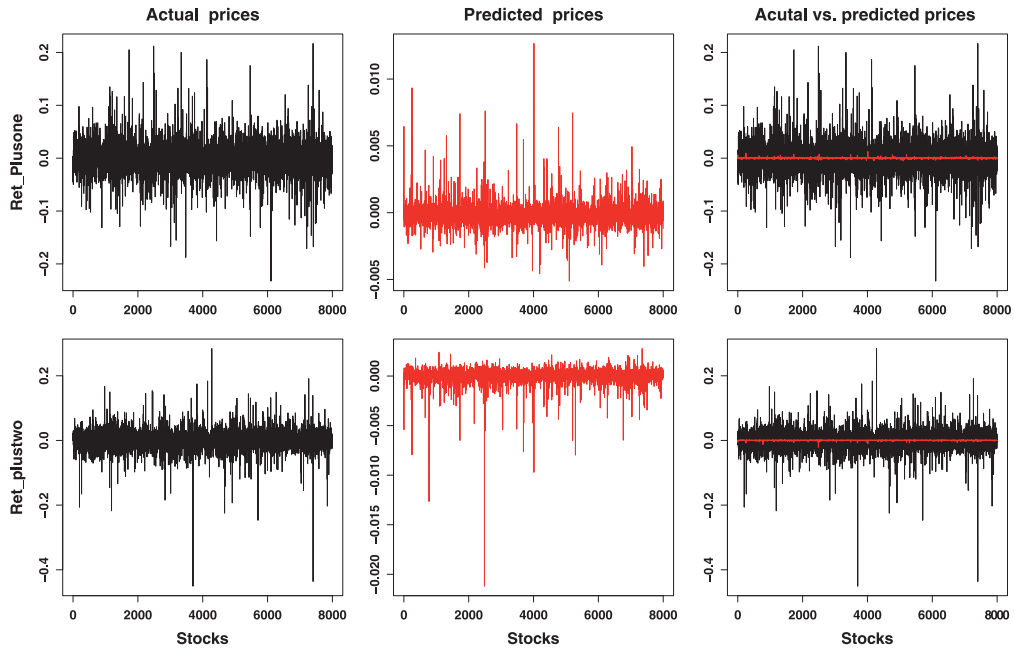

Figure 14. Actual and predicted stock price of *Ret_PlusOne* and *Ret_PlusTwo*.

noted that on a parallel computing environment back propagation algorithm may give better results which will be the future direction for us to explore. We also compared our results with multi-linear regression based model and it was observed that back propagation based model was giving better results on such difficult predictions.

Acknowledgments

Authors are grateful to the editor and the anonymous reviewers for their constructive comments which have helped us to improve the quality of the arguments at a great deal.

References

Atsalakis, G. S., and K. P. Valavanis. 2009. Forecasting stock market short-term trends using a neuro-fuzzy based methodology. *Expert Systems with Applications* 36:10696–707.

Benjamin, G., D. L. F. Dodd, and S. Cottle. 1934. *Security analysis*. New York: McGraw-Hill.

Bogaert, M., M. Ballings, and D. V. den Poel. 2016. The added value of Facebook friends data in event attendance prediction. *Decision Support Systems* 82:26–34.

Cedric, O., and E. Jacquier. 2016. Horizon effect in the term structure of long-run risk-return trade-offs. *Computational Statistics and Data Analysis* 100:445–66.

Dell'Aquila, R., E. Ronchetti. 2006. Stock and bond return predictability: the discrimination power of model selection criteria. *Computational Statistics and Data Analysis* 50:1478–95.

Enke, D., M. Grauer, and N. Mehdiyev. 2011. Stock market prediction with multiple regression, fuzzy type-2 clustering and neural networks. *Procedia Computer Science* 6:201–6.

Giordano, F., M. L. Rocca, and C. Perna. 2007. Forecasting nonlinear time series with neural network sieve bootstrap. *Computational Statistics and Data Analysis* 51:3871–84.

Guresen, E., G. Kayakutlu, and T. U. Daim. 2011. Using artificial neural network models in stock market index prediction. *Expert Systems with Applications* 38:10389–97.

Haykin, S. S. 1999. *Neural networks: A comprehensive foundation*. Prentice Hall PTR: Pearson Education.
Hill, T., and W. Remus. 1994. Neural network models for intelligent support of managerial decision making. *Decision Support Systems* 11:449–59.
Intrator, O., and N. Intrator. 2001. Interpreting neural-network results: A simulation study. *Computational Statistics and Data Analysis* 37:373–93.
Jagadish, H. V., J. Gehrke, A. Labrinidis, Y. Papakonstantinou, J. M. Patel, R. Ramakrishnan, and C. Shahabi. 2014. Big data and its technical challenges. *Review Article Communications of ACM* 57:86–94.
Kaastra, I., and M. Boyd. 1996. Designing a neural network for forecasting financial and economic time series. *Neurocomputing* 10:215–36.
Khansaa, L., and D. Liginlalb. 2011. Predicting stock market returns from malicious attacks: A comparative analysis of vector autoregression and time-delayed neural networks. *Decision Support Systems* 4:745–59.
Li, D.-C., L.-S. Lin, and L.-J. Peng. 2014. Improving learning accuracy by using synthetic samples for small datasets with non-linear attribute dependency. *Decision Support Systems* 59:286–95.
Maa, G.-Z., E. Songa, C.-C. Hungb, L. Sua, and D.-S. Huanga. 2012. Multiple costs based decision making with back-propagation neural networks. *Decision Support Systems* 3:657–63.
Mo, H., J. Wang, and H. Niu. 2016. Exponent back propagation neural network forecasting for financial cross-correlation relationship. *Expert Systems with Applications* 53:106–16.
Nawi, N. M., W. H. Atomi, and M. Z. Rehman. 2013. The effect of data pre-processing on optimized training of artificial neural networks. *Procedia Technology* 11:32–39.
Oztekin A., R. Kizilaslan, S. Freund, and A. Iseri. 2016. A data analytic approach to forecasting daily stock returns in an emerging market. *European Journal of Operational Research* 253:697–710.
Parvathi, R., and J. H. Rekha. 2015. Survey on software project risks and big data analytics. *Procedia Computer Science* 50:295–300.
Piramuthu, S., M. J. Shaw, and J. A. Gentry. 1994. A classification approach using multi-layered neural networks. *Decision Support Systems* 11:509–25.
Potthoff, R. F. 2007. Prediction markets, bayesian priors, and clinical trials. *Journal of Statistical Planning and Inference* 137:3706–21.
Ramesh, V. P., B. Priyanga, M. Priyadharshini, R. Sowmitha, and Y. V. Shiva Prakash. 2016. Health care big data analytics – predicting diabetes. *Sugyaan* 8 (1):13–30.
Ramesh, V. P., V. Favas, and M. E. Joseph. 2018. Technology landscape of big data analytics and convolutional neural network for image classification. *International Journal of Engineering Science, Advanced Computing and Bio-Technology*.
Robert L. M., and B. Peter F. 2016. System and method for executing synchronized trades in multiple exchanges. US patent application number: US20160035027A1, Assignee: Renaissance Technologies LLC.
Rocca, M. L., and C. Perna. 2005. Variable selection in neural network regression models with dependent data: a subsampling approach. *Computational Statistics and Data Analysis* 48:415–29.
Sagiroglu, S., and D. Sinanc. 2013. Big data: A review. In: Collaboration Technologies and Systems (CTS), 2013 International Conference on, IEEE. pp. 42–7.
Singh, D., and C. K. Reddy. 2014. A survey on platforms for big data analytics. *Journal of Big Data* 2:8.
Tsay, R. S., T. Ando. 2012. Bayesian panel data analysis for exploring the impact of subprime financial crisis on the us stock market. *Computational Statistics and Data Analysis* 56:3345–65.
Wong, B. K., T. A. Bodnovich, and Y. Selvi. 1997. Neural network applications in business: A review and analysis of the literature (1988–1995). *Decision Support Systems* 4:301–20.
Xi, L., H. Muzhou, M. H. Lee, J. Li, D. Wei, H. Hai, and Y. Wu. 2014. A new constructive neural network method for noise processing and its application on stock market prediction. *Applied Soft Computing* 15:57–66.