PREDICTIVE PERFORMANCE EVALUATION OF DIFFERENT NEURAL NETWORKS USING STOCK PRICES

A Paper
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Deepika Nyavanandi

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Department:
Computer Science

May 2018

Fargo, North Dakota

North Dakota State University
Graduate School

Title

PREDICTIVE PERFORMANCE EVALUATION OF DIFFERENT NEURAL NETWORKS USING STOCK PRICES

By

Deepika Nyavanandi

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Dr. Simone Ludwig
Chair

Dr. Pratap Kotala

Dr. Azer Akhmedov

Approved:

05/07/2018
Date

Dr. Kendal Nygard
Department Chair

ABSTRACT

Forecasting stock market prices has been a challenging task due to its volatile nature and nonlinearity. Recently, artificial neural networks (ANNs) have become popular in solving a variety of scientific and financial problems including stock market price forecasting. ANNs have the ability to capture the underlying nonlinearity and complex relationship between the dependent and independent variables. This paper aims to compare the performance of various neural networks including Feed Forward Neural Networks (FNN), Vanilla Recurrent Neural Network (RNN), and Long Short-Term Memory (LSTM) neural networks by forecasting stock market prices of three different companies. Empirical results show that the LSTM neural network performed well in forecasting stock market prices compared to both vanilla RNN and FNN.

ACKNOWLEDGEMENTS

I would like to thank all the people who contributed in some way to the work described in this paper. First and foremost, I thank my adviser, professor Simone Ludwig, for her precious time and providing valuable advice to complete my research paper on time. I would also like to thank my committee members Professor Pratap Kotala, Professor Azer Akhmedov for their interest in my work.

Finally, I would like to thank my parents and my partner for constant love and support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

Machine learning has gained popularity in the last decade. The field of machine learning consists of different models including, but not limited to, support vector machines, random forests, and artificial neural networks. Few models are specialized to study classification problems while other models are specialized in studying both classification and regression problems. Artificial Neural Network (ANN) is a good candidate for solving both classification and prediction problems [1]. Hence, ANN is the most widely used machine learning model. Previous studies have indicated that ANNs can approximate any universal function and are applied to solve broad range of problems in many fields including sales forecasting, price forecasting, finance, medicine, engineering, and physics etc. [1, 2]. Forecasting using ANN has been an active research area over the past few years.

Primarily there are two types of machine learning techniques including supervised learning and unsupervised learning. Supervised learning consists of training the model with a predetermined explanatory (X) and explained (Y) variables. The unsupervised learning technique involves feeding the entire data to determine the structure of the data [3].

This project compares different neural networks including feed forward neural networks, and recurrent neural networks (both vanilla and LSTM) by forecasting stock market prices of Amazon, S&P, eBay companies. The data are obtained from the Yahoo finance website [4].

1.1. Forecasting

Forecasting is the process of predicting the future using past data and current data. It is a process involving uncertainty [5]. The selection and implementation of a forecasting method is important for many businesses, since most financial and marketing decisions are made based on

the accuracy of the forecast. Literature on the topic consists of two most prevalent forecasting techniques, which are causal forecasting, and time series forecasting.

1.1.1. Causal Forecasting

In causal forecasting, the independent variables (X_t) are used to predict the dependent variable (Y_t). Causal forecasting can be formulated as follows [5]:

$$Y_t = f(X_{1t}, X_{2t}, X_{3t}, X_{4t}, \dots.) \quad (1)$$

where Y_t is the dependent variable, and X_{it} are the independent variables. Expert knowledge is needed to select the appropriate independent variables that predict the dependent variables accurately enough [5].

1.1.2. Time Series Forecasting

A time series consists of data points obtained over time at equally spaced intervals (daily, monthly, quarterly, yearly) [6]. The fundamental difference between linear regression and time series is that time series data is time dependent whereas the linear regression model assumes that the data is independent of time, that means each observation is independent of each other. Time series data often contain seasonality and trend over time based on the frequency of the data [7]. There are two ways of analyzing time series. One is fundamental analysis, and the other is technical analysis. Fundamental analysis determines the future values based on the underlying factors that affect a company's real business and its future predictions. Whereas technical analysis determines the future values based on the historical values and its behavior over time [8]. Time series forecasting is a technique used to predict the future values (Y_{t+1}, Y_{t+2}, …) based on historical observations of the same variable (Y_t, Y_{t-1}, Y_{t-2}, …) and patterns that exist in the data [9]. Y_t is the value (output) at time t and inputs are the past observations of Y_t. Time series forecasting can be formulated as follows:

$$Y_t = f(Y_{t-1}, Y_{t-2}, Y_{t-3}, \dots) \quad (2)$$

Few examples of time series data include Dow-Jones Industrial Average, sunspot activity, number of births in a community, air temperature in a building, etc. As mentioned earlier, the time series data is disaggregated into trend, seasonality, and noise components [10]. Trend is an increasing or decreasing behavior of the series over time while seasonality is the repetitive patterns or cycles of the data over time. Finally, noise is the variation in the dependent variable that cannot be explained by the independent variables [10].

Predictions are made using two types of lags. If predictions are made using only past values, then that type of prediction is called one-lag prediction. On the other hand, if predictions are made by appending the predicted value to the current input to predict the next value then that type of prediction is called multi-lag prediction. For instance, in one-lag prediction, T1, T2, T3, T4 are used to predict the T5 value. Whereas in multi-lag prediction T2, T3, T4, T5 are used to predict the T6 value and T3, T4, T5, T6 are used to predict the T7 value [6]. In this paper, one lag prediction approach was used to forecast stock market prices of Amazon, S&P, eBay companies.

Forecasting models could be linear as well as nonlinear. Artificial neural networks (ANN) are examples of nonlinear forecasting models. A simple ANN contains an input layer, hidden layer, and an output layer. Activation (transfer) functions used in the hidden layer make the neural network nonlinear. The three main objectives of the project include: 1) highlight the differences between feed forward and recurrent neural network, 2) forecast time series and compare the results obtained using both the feed forward and recurrent neural networks in terms of RMSE, and 3) explain the neural network architecture in the context of time series forecasting.

Importance of Time Series Forecasting: The accuracy of forecasts is important for several reasons. First, the forecasts are used to make both short-term and long-term decisions. Second, the forecasts help to manage the uncertainty in the future. In many industries, most of the financial operations are based on the accuracy of forecast to make operative decisions in purchasing, marketing and advertising, etc. For instance, less accurate forecasts may lead the company to make wrong decisions and thereby will lead to loss in the revenues. Hence, the research to develop a good forecasting model and to improve the effectiveness of existing forecasting models has been an active area of research [11].

Importance of Stock Market Price Forecasting: Forecasting stock market price is important in financial and economic studies. It is an act of predicting the potential of the company. The successful prediction of a stock's future price could yield significant profits to the company. Most of the times, it is very difficult to predict stock market prices accurately just by looking at its price history because of the uncertainty in the market, economic and political factors instantaneously [8], and by nature stock market is nonlinear and unstable [12]. Even though many classical methods such as Box-Jenkins, ARMA and ARIMA have been developed to forecast time series they fail to address the major challenges such as uncertainty and nonlinearity of the financial data, as they often assume a linear relationship between input and output variables. On the other hand, neural network models have been proven to be good at approximating the nonlinear function without requiring any apriori information [1]**.**

1.2. Neural Networks

A neural network may be assumed as a nonlinear regression function describing the relationship between the dependent variable (target) ***Y*** and the independent variables (inputs) ***X*** [13]. ANNs are inspired by the biological nervous systems. A simple ANN is typically organized into 3 layers including an input layer, hidden layer, and an output layer. Layers are made up of a few interconnected nodes, which contain an activation function. Weights in the network determine how strong two neurons are connected [14].

1.2.1. Biological Motivation

ANNs are inspired from the biological nervous system [12]. The architecture and functionality of ANNs mimic the brain in the following way [2]. A neuron is a basic computational unit of a brain. Synapses connects all neurons in the human nervous system. Dendrites carry the signal to the cell body, where they all get summed up. The axon finally branches out and links via synapses to dendrites of other neurons. Each neuron gets input signals from its dendrites and produces an output signal along its axon [15]. Figure 1 shows the structures of both a biological neuron and the ANN mathematical model.

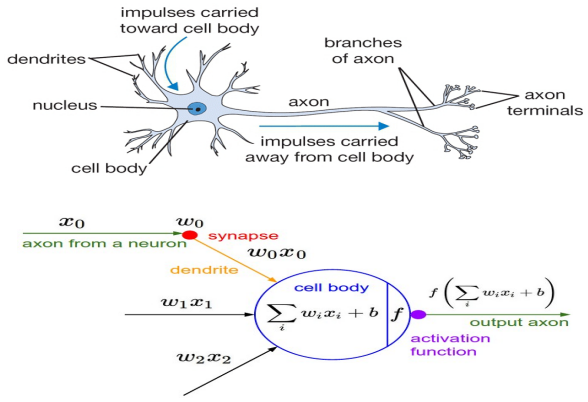

Figure 1. Structure of both biological neuron and ANN model[1]

[1] Source: http://cs231n.github.io/neural-networks-1/

In the ANN model, each neuron performs a dot product with the input (x_0) and its weight (w_0) along with the bias unit in the input layer. In the hidden layer, the result of the input layer is passed through a non-linear (or activation) function and the final result is passed on to the output layer [15].

The rest of the paper is organized in following way. Chapter 2 describes the related work of time series forecasting and stock market price forecasting. Chapter 3 describes the different types of Neural Network architectures and different backpropagation training algorithms. Chapter 4 describes the data set and preprocessing techniques applied before executing the neural network algorithm. Results, in the form of table and diagrams, are presented in Chapter 5. Chapter 6 concludes the paper.

2. RELATED WORK

In order to forecast time series, there are several linear statistical and econometric models are available including pure autoregressive (AR), pure moving average (MA), exponential smoothing and combined AR and MA (ARMA). However, the main disadvantage of all kinds of linear models is that they only capture the linear correlation and do not consider the nonlinear patterns that exist in the data [16]. To overcome the limitations of linear models some of the other nonlinear forecasting models such as bilinear model, the threshold autoregressive (TAR) model, and the autoregressive conditional heteroscedastic (ARCH) models have been developed but they can be applied to specific nonlinear problems. The applicability of these models to the general forecasting problems is limited [16].

Many linear statistical models for time series forecasting are computationally inexpensive and have proven to be inaccurate in addressing the nonlinear patterns in the data. Whereas nonlinear models such as neural network account for nonlinear patterns in the data and predict future values accurately [6].

For the first time in 1964, Hu stated his idea to use ANNs for prediction problems and used for forecasting weather without any learning algorithm [12]. In 1988, Werbos used a learning algorithm, which was introduced in 1980 to train the ANN, and stated that ANNs are better than regression methods and the Box-Jenkin model for prediction problems [12]. The main advantage of ANN's nonlinear model is that it is good at capturing nonlinear patterns that exist in the data [16]. In [16], Zhang applied both the ARIMA (linear) model and the ANN (nonlinear) model to forecast the time series to improve the forecasting accuracy compared to either of the model separately. Experimental results on real data have shown that the combined (hybrid) model achieved good accuracy compared to either of the model.

In the recent decade, many researchers have used neural networks to predict stock market changes. For the first time, Kimmoto and his colleagues used neural networks to predict the index of the Tokyo stock market [10]. Mizuno and his colleagues also used neural networks to predict the trade of stocks in the Tokyo stock market and achieved a 63% precision [12].
In [6], Chakraborty et al. used a feedforward neural network approach for multivariate time series analysis and predicted the monthly flour prices of Buffalo, Minneapolis and Kansas City over a period of a hundred months. Results show that the neural network approach leads to better results compared to the ARIMA model.

Due to some special characteristics of Artificial Neural Networks (ANN) such as generalization ability and capturing underlying non-linear patterns of the data, the application of ANN for prediction problems became popular [12]. In [13], Kuan and Liu proposed a two-step procedure to estimate and select feedforward and recurrent neural networks and investigated nonlinear patterns in foreign exchange data using selected networks in different out-of-sample periods. They also found that the predictive stochastic complexity (PSC) is a reasonable criterion to select networks. Finally, results show that there is no major difference between feedforward and recurrent neural networks in terms of out-of-sample MSE (mean squared errors) and sign predictions (i.e., forecasts of the direction of future changes).

In [1], Oancea, Bogdan, and Ciucu compared the performances of different neural networks such as feedforward and recurrent neural networks as well as using different training algorithms to predict the exchange rates of EUR/RON and USD/RON. Before applying the model, they applied preprocessing techniques to remove the correlation between data as well as to normalize the data. Based on the test results, Recurrent Neural Networks performed better than feed forward neural network.

In general, it is very difficult to predict stock market prices because more often, stock market prices react to news instantly. In [8], Xu and Yue predicted weekly changes in stock prices by combining conventional time series analysis with the information from the Google trend website and the Yahoo finance website and achieved better results compared to conventional time series analysis.

In [14], Gamboa and Borges reviewed and summarized few deep learning techniques and some applications on time series analysis and concluded that applying deep learning techniques for time series analysis yielded better results than existing time series analysis techniques.
In [17], Claveria and Torra evaluated the forecasting performance of artificial neural networks relative to different time series models such as autoregressive integrated moving average (ARIMA) models and self-exciting threshold auto regressions (SETAR) at the regional level by forecasting tourism demand. The results indicated that ARIMA models outperformed both SETAR and ANN models. RNNs have a long history, but their recent popularity is mostly due to the research works of Schmithuber, Hochreiter, and Graves. RNNs applications include areas ranging from speech recognition to driverless cars [27].

In summary, ANNs performed well compared to traditional statistical models in almost all of the cases.

3. APPROACH

3.1. Feed Forward Neural Network

A typical ANN consists of 3 layers. The first layer is input layer, the second layer is/are the hidden layer(s), and the last layer is the output layer [12]. In feed forward neural networks, the information flow is unidirectional from the input layer to the hidden layer and from the hidden layer to the output layer. Most feed forward neural networks are organized in layers and this architecture is known as a Multilayer Perceptron (MLP) [19]. Figure 2 is an example of a neural network topology with a single hidden layer.

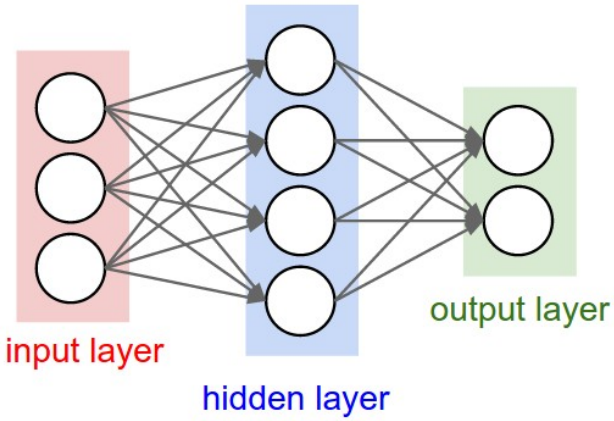

Figure 2. ANN and its architecture with three layers[2].

There are usually four types of neurons and three types of layers in a neural network [20]:

- **Input Neurons** - Each input neuron represents one element in the feature vector.
- **Hidden Neurons** - Hidden neurons allow the neural network to process the input and to forward the result to the output.
- **Output Neurons** - Each output neuron computes one part of the output.
- **Context Neurons** - Holds the state between calls of the neural network to predict.

[2] Source: http://cs231n.github.io/neural-networks-1/

- **Bias Neurons** - Works like the Y-intercept of a linear equation.

These neurons are grouped into layers [20]:

- **Input Layer** - The input layer accepts values from the dataset. Input layers usually have a bias neuron.
- **Output Layer** - This layer consists of output of the neural network. The output layer does not have a bias neuron and activation functions.
- **Hidden Layers** - Remain between the input and output layers. Each hidden layer will usually have a bias neuron.

The number of layers of an ANN varies based on the application. The best way to determine the neural network architecture is the trial and error approach. Every neural network contains a single input layer and a single output layer, but varying numbers of hidden layers. The ANN structure is one of the major factors in achieving satisfactory prediction results. A very simple network may not be able to approximate the function well. Also, a very complex network may overfit the data. However, there is no exact approach regarding the determination of the network complexity [13]. One must design an ANN with an appropriate number of layers, neurons and connections [12]. Other factors that affect the prediction results are activation functions, evaluation measures, and training algorithms [12]. If an ANN contains more than one hidden layer, then that ANN is usually referred to as a deep neural network. Typically, a deep neural network has more learning capacity in order to learn more complex behavior than shallow neural networks. As the number of layers in the network increases the number of learning parameters (weights and biases) also increase.

Selection of the number of neurons and the number of hidden layers is problem specific. For example, if one opts for more hidden layers for a simple problem then the network may

overfit the data. Overfitting occurs when the model fits the noise instead of the underlying relationship. Hence, one cannot claim that a deep neural network has always more learning capacity than a single hidden layer neural network.

In machine learning, the first step is to disaggregate the entire data into a training set and test set. Given the training set, a neural network can learn using a learning algorithm. Most commonly used learning algorithm is the backpropagation algorithm. The backpropagation algorithm forms a mapping between the input and the desired output by updating the weights that connect the network. While testing the network with the new data, the network predicts the values using weights updated during training [6]. Data are presented to the network via the input layer, which communicates to one or more hidden layers where the actual processing occurs using weights. The hidden layers are then connected to output layer.

Two important parameters are needed for building the architecture of the neural network. First, the number of hidden nodes in the hidden layer. Second, to determine the number of lagged values (past observations of the stock price), which are used as the inputs in the input layer. Using very few inputs and hidden nodes results in inadequate modeling, whereas, too many inputs and hidden nodes makes the model complicated [6]. Often a trial and error approach is used to determine the appropriate number of hidden nodes as well as the input nodes for the network.

The entire learning process of a neural network can be generalized in three steps:

1. Calculate the loss function
2. Back propagate the loss value and calculate the gradient
3. Update weights

Calculate loss function: The input data presented to the neural network goes through the whole network and then the accuracy between the predicted value and the desired value is compared. The loss is calculated as follows:

$$\frac{\text{Error}}{\text{loss}} = \text{desired output} - \text{actual output} \tag{3}$$

Backpropagation: At this stage, the gradients (derivative of loss function w.r.t to weights) are calculated. The backpropagation algorithm finds the minimum of the error/loss function in the weight space using gradient descent. The combination of weights, which minimizes the error function is considered as a solution of the learning problem [12].

Weight Update: Weights are updated using the gradients calculated in the backpropagation phase scaled by the learning rate. The weights of the network are adjusted based on the error calculated between the actual output and the output predicted by the network until the network converges [18].

$$\mathrm{W}' = \mathrm{W} - (\text{learning rate} * \text{gradient}) \tag{4}$$

where W is the initial weight, while W' is the updated weight.

The learning rate is an important hyper parameter in training the neural network because:

1. If the learning rate is too low, then near accurate results will be calculated but the training process is very slow.
2. If the learning rate is too high then it speeds up the training process, but the results may not be good enough.

Common values for learning rates are 0.1, 0.01, 0.001. Figure 3 depicts the effect of the learning rate on the loss function.

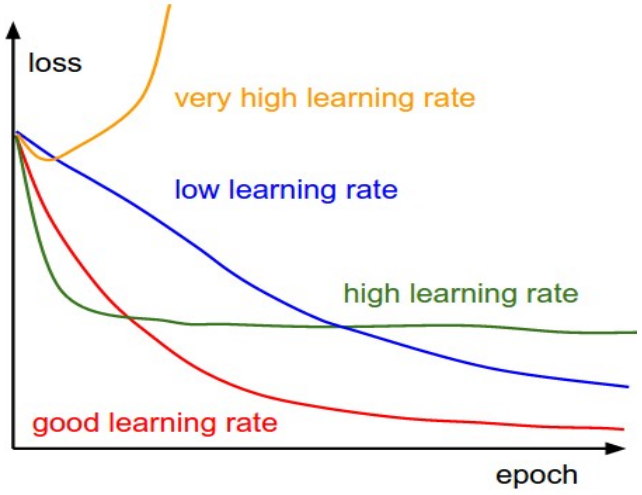

Figure 3. Pictorial representation of different learning rates and the loss[3]

3.2. Recurrent Neural network

As mentioned earlier, the feed forward neural network always has forward connections. The flow of the data in feed forward networks is always from the input layer to the output layer through the hidden layers. In contrast, the Recurrent Neural Networks (RNNs) have connections in both forward and backward directions. The recurrent connections in the RNN connects the neurons in the current layer to the following neurons [20]:

1. Neurons in the same layer
2. Neurons in the previous layer
3. Neurons itself

The following are the key differences between the feed forward neural networks (FNNs) and recurrent neural networks (RNNs). First, RNNs use sequential information, which means it receives a sequence of values as input and it also produces a sequence of values as output while FNNs assume that data is non-sequential, and that each data point is independent of one another

[3] Source: http://cs231n.github.io/neural-networks-3

[19]. As a result, the inputs are analyzed in isolation, which cause problems if there are dependencies in the data. Second, most RNNs maintain a state in the recurrent connections to capture previously computed information, whereas FNNs do not maintain any state. The state in the RNN acts as a short-term memory for the neural network. Thus, RNN will not produce the same output for a given input. When we unfold the RNN, each fold looks like a FNN with an extra input from its previous hidden state.

Figure 4 shows the architecture of the recurrent neural networks. In the figure, x_t is the input at time step *t*, s_t is the hidden state at time step *t*. s_t acts as a short-term memory for the network to capture the previously computed information. s_t is usually computed based on the previous hidden state and the current input. Usually, s_t is computed using the following formula [19]:

$$s_t = f(Ux_t + Ws_{t-1}) \tag{5}$$

Here, the function *f* is a non-linear function such as rectified linear unit (ReLU), *tanh*, or sigmoid. U, W are the parameters. Finally, o_t is the output at time step *t* [19].

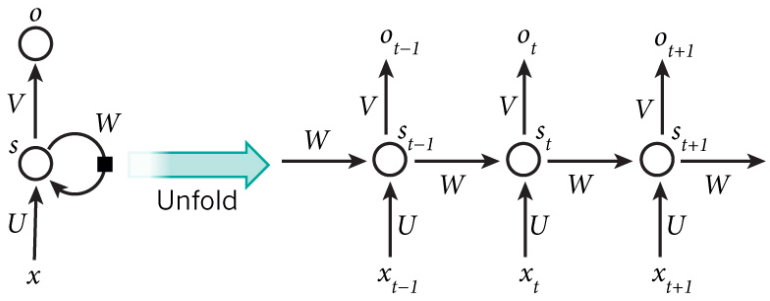

Figure 4. Recurrent neural networks and its architecture[4]

[4] Source: http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

A few applications of RNNs include language modeling, image captioning, machine translation, and speech recognition etc. Training a RNN is similar to traditional Neural Network with small changes to the backpropagation algorithm since parameters shared by all time steps in the network calculation of the gradient depends not only on the current time step but also on the previous time steps. This process is known as Backpropagation Through Time (BPTT) [19]. The main disadvantage of vanilla RNNs is that they are not good at accounting for long-term dependencies (dependencies between steps that are far apart) due to the vanishing/exploding gradient problems [19]. A vanishing gradient is a situation in which the gradient gets smaller as we move backward across the hidden layers. That is, the neurons in the earlier layers learn much more slowly than the neurons in later layers. While an exploding gradient is a situation in which the gradient gets much larger in earlier layers. Hence RNNs are useful in a situation where the gap between the relevant information and the place that it required is small since the gap increases RNNs incapable of learning the required information [21].

3.3. Long Short-Term Memory Neural Network

LSTMs are a special type of RNNs that are often used with deep neural networks. LSTMs were proposed by Hochreiter and Schmidhuber in late 1990s [9] and they are capable of learning long-term dependencies better than vanilla RNNs [19]. The only difference between LSTM, and vanilla RNNs architecture is that each one calculates the hidden state in a different way [14]. LSTMs have three different cells in the hidden state and they act as memory for the network. These cells take the previous hidden state (h_{t-1}) and the current input (x_t) together as input and decide the needed information in the memory. The number of neurons in the LSTM network defines the learning capacity.

LSTM uses two types of internal transfer functions. They are 1) sigmoid transfer function and 2) hyperbolic tangent (*tanh*) function. The sigmoid transfer function is used for the gated units inside of the unit, whereas the *tanh* transfer function is used to scale the output of the LSTM.

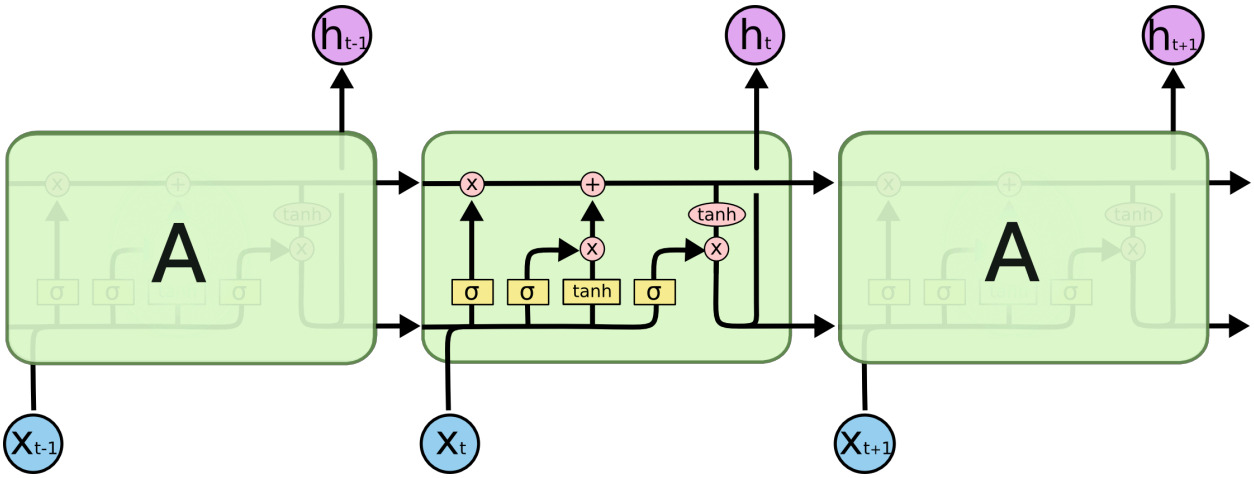

Figure 5. LSTM and its architecture[5]

The LSTM is made up of three gates:

- Forget Gate (f_t) – This gate decides the information to be retained from cell state. The sigmoid layer makes this decision and then it results in outputs between 0 and 1. The output "one" represents "completely keep this" while the output zero represents "completely forget this". The following formula represents the forget gate layer [21]:

$$f_t = \sigma[W_f(h_{t-1}, x_t) + b_f] \quad (6)$$

- Input Gate (i_t) – This gate decides when a value should be remembered by the context. The formula for input gate is shown below [21]:

$$i_t = \sigma[W_i(h_{t-1}, x_t) + b_i] \quad (7)$$

[5] Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

- Output Gate (o_t) – The output gate decides the parts of the cell state that are sent as output. The formula for the output gate looks as follows [21]:

$$o_t = \sigma[W_o(h_{t-1}, x_t) + b_o] \tag{8}$$

To create an update to the state, we use the candidate value along with the forget gate and the input gate. The candidate value is calculated using the following formula [21]:

$$C_t = \tanh\left[W_c(h_{t-1}, x_t) + b_c\right] \tag{9}$$

To update the old state, C_{t-1} to the new state C_t, we multiply the old state by f_t then we add the new state multiplied by i_t [21]:

$$C_t = (f_t \times C_{t-1}) + (i_t \times C_t) \tag{10}$$

To compute current hidden state h_t, we apply *tanh* activation function to the new state C_t and multiply it by o_t as follows [21].

$$h_t = o_t \times \tanh(C_t) \tag{11}$$

In all above formulas, W refers to the weight and b refers to the bias value. Over a period of time, a recurrent neural network tries to learn how much information to retain from the past, and how much information to keep from the present state, which makes the RNN powerful as compared to a simple FNN.

Optimization: It is the process of finding the appropriate weights, which minimizes the loss function [22].

Gradient descent: The process of repetitive evaluation of the gradient and then updating the weights is known as gradient descent. Although many different optimization algorithms exist, gradient descent is the most commonly used optimization algorithms to optimize the neural network loss functions [22].

Mini-batch Gradient Descent: If the training data size is large enough then it is inefficient to compute the gradient using the entire data to update a single parameter. The usual approach to address this problem is to calculate the gradient over batches of training data rather than the whole data. The gradient from a mini-batch is a good estimate of the gradient computed using the entire data. Hence, faster convergence is possible by calculating mini-batch gradients to perform more frequent parameter updates [22].

Stochastic Gradient Descent or online gradient Descent: The process of computing gradients using a single observation at a time instead of a batch or the whole data. In this process, the batch size is equal to one. This process is computationally efficient to calculate the gradient for 100 observations rather than calculating the gradient for an observation 100 times [22].

3.4. Backpropagation

The Backpropagation learning algorithm falls under the general category of gradient descent algorithm. The main objective of the gradient descent algorithms is to minimize the error function by iteratively moving in the direction of the negative slope of the function [1]. The main problem with the gradient descent algorithm is that it suffers from potentially getting trapped in a local minimum.

Momentum Backpropagation: It is one of the backpropagation techniques. In the momentum back propagation, another hyper parameter called momentum is used to escape the gradient from the local minima. In this process, the weights are updated not only using the learning rate but also using the momentum. The previous change in the weight is scaled by the momentum and added to the current weight as shown below [23]:

$$W' = W - (\text{learning rate} \times \text{gradient}) + (\text{momentum} \times \text{previous amount of weight change}) \quad (12)$$

The most commonly used value for the momentum is 0.9.

Batch Backpropagation: If the weights are updated based on the sum of the gradients over all the training set elements then that training process is called batch backpropagation [23].

Online Backpropagation: If the weights are updated based on gradients calculated from a single training set element then that training process is called online backpropagation [23].

Epoch: is the number of times the whole training set was processed [23].

Batch size: Batch size is smaller than the whole training set and usually 32, 64, etc. Batch size controls the frequency level of updating the weights of the network [23].

Steps or Iteration: Training set size/batch size [23].

For instance, if the training set size is 1000 and the batch size is 100 then the number of steps or iterations equal to 10. In addition to the above training algorithms, there exists few other training algorithms, which do not depend on the leaning rate and momentum values [23]:

- **Resilient Propagation** – It uses only the magnitude of the gradient and permits each neuron to learn at its own rate. There is no need of learning rate/momentum. But, the resilient propagation only works in full batch mode [23].
- **Nesterov accelerated gradient** – This helps to lessen the risk of choosing a bad mini-batch [23].
- **Adagrad** –This allows an automatically decaying per-weight learning rate and momentum. The drawback of Adagrad is that in case of deep learning, the monotonic learning rate usually proves to be too aggressive and stops learning too early [23].
- **Adadelta –** It is an extension of Adagrad that seeks to decrease its aggressive, monotonically decreasing learning rate.

- **Non-Gradient Methods** – In some cases, the non-gradient methods could outperform gradient-based backpropagation methods. Few non-gradient methods include simulated annealing, genetic algorithms, particle swarm optimization, and Nelder Mead [23].

3.5. Data preparation

Data set: For this experiment, the monthly stock market prices of three different companies have been collected from the Yahoo finance website with three different period ranges. The three companies' datasets include Amazon monthly stock prices (2001-2016), eBay monthly stock prices (2001-2016), and S&P monthly stock prices (1951-2016). Each company dataset contains different columns of the data including open, high, low, closing and adjusted closing prices. In this paper, the closing price of each company was selected for forecasting.

Data splitting: In machine learning models, the data are divided into train and test sets. The model learns the past behavior and patterns from the training data. Later, we can test the performance of the model by comparing the ground truth with the predicted test values [9]. The decision about splitting the data is based on the different factors including number of observations, quality of the data, and variation in the data. There is no hard and fast rule for data splitting. Depending on the number of observations in the dataset, the entire dataset could sometimes be split into three subsets including train, test, and validation sets. However, in this paper, 20% of the data was used as test data and the remaining 80% data was used as a training data for all datasets.

Data preprocessing: Before applying the algorithm, the usual practice is to explore the data for any missing values and outliers. If the data contain any missing values or outliers, then one should delete them using the appropriate data preprocessing technique as they impact the results significantly. In this project, the dataset does not have any outliers or missing values.

Therefore, the preprocessing technique was not needed. Usually, time series data are sequential data that contain patterns including seasonality, and trend components. Stationarity of the data is a precondition for forecasting. Few data preprocessing techniques applied in this paper are shown below.

Stationarity: Time series is a special type of data. Before fitting a model, one needs to convert the time series to stationary. We cannot fit the model to nonstationary data. When we make the data stationary the statistical properties of the data such as mean, variance, auto correlation, etc. are constant over time [7]. To make the data stationary, I have converted the data series in to time series object and removed the trend and seasonality.

There are a number of statistical tests that exist to check whether the data is stationary. Among them, Augmented Dickey-Fuller (ADF) test is the most widely used statistical test. The null hypothesis (H_0) of the test is that the time series is nonstationary. The alternate hypothesis (H_1) of the test is that the time series is stationary. The tests results are interpreted using the p-value. A p-value below a threshold (1% or 5%or 10%) indicates the rejection of the null hypothesis (stationary), otherwise accept the null hypothesis (non-stationary).

p-value>0.05: Accept the null hypothesis (H_0), the data is not stationary.

p-value<=0.05: Reject the null hypothesis (H_1), the data is stationary.

Normalization: For this data preprocessing technique, the data are normalized so that the entire dataset is uniform. If one fails to normalize the data, then the larger input value overwhelms the smaller input value. This may lead to wrong conclusions. Normalization of the data helps the optimizations algorithm to converge quickly and also reduces the prediction error [24].

Standardization or Z-score normalization: This is one of the common types of normalization techniques. In this process the input values will be rescaled, as a result features will have the properties of a standard normal distribution with mean zero and standard deviation one. Standard score can be calculated using the following formula [24].

$$Z = \frac{(x - \mu)}{\sigma} \tag{13}$$

In the above formula, x is the input value, μ is the mean and σ is the standard deviation.

Min-Max scaling: It is an alternative approach to standardization. In this process, the data are scaled to fixed range either from 0 to 1 or -1 to 1. Min-Max scaling is typically applied using the formula below [24]:

$$X_{norm} = \frac{(X - X_{min})}{(X_{max} - X_{min})} \tag{14}$$

In the above formula, X is the single data point, X_{min} is the minimum value, and X_{max} is the maximum value. Depending on the dataset, one needs to choose an appropriate normalization technique. In this project, z-score normalization was used to preprocess the data.

Weight initialization: Initializing the weights of the neural network is an important issue because if one initializes all weights with zero then every neuron in the network computes the same output. Also, the neurons compute the same gradients during backpropagation and update each weight with the same value. Thus, the network does not learn the underlying patterns. The best way to initialize the weights of the network is to randomly initialize the weights and make sure that they are close to zero but not exactly zero. The algorithm converges faster by computing different gradients and update each weight with a different value based on the input variable.

3.6. Activation functions

Each neuron or node in the neural network processes the information using an activation function. The choice of activation function has a big impact on the network's behavior. There are a number of activation functions used in artificial neural networks. Some of them are described in this section.

Sigmoid: One of the most popular activation functions used for the backpropagation network is sigmoid. Sigmoid activation function has the mathematical form as follows:

$$\frac{1}{1+e^{-Z}} \; where \; Z = w_i x_i + b_1 \tag{15}$$

The sigmoid function is a non-linear function (shown in Figure 6) in which it takes the real valued numbers as an input and results in values usually between zero and one. That is, large negative numbers become 0 and large positive numbers become 1 [15].

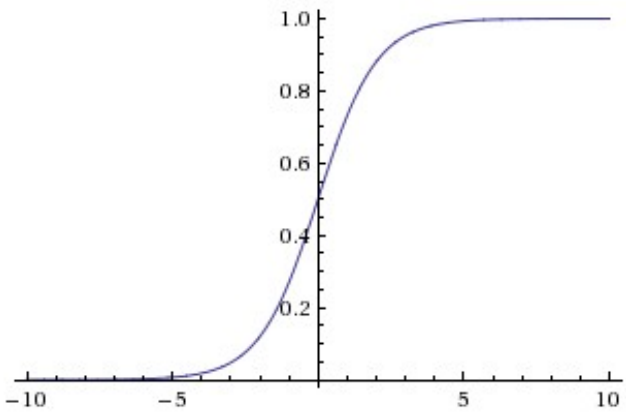

Figure 6. Pictorial representation of sigmoid function[6]

One of the major disadvantages of the sigmoid function is the vanishing gradient problem.

[6] Source: http://cs231n.github.io/neural-networks-1/

Vanishing gradient problem: When a neuron's activation saturates at either tail of 0 or 1, the gradient at these areas is almost zero. During backpropagation, local gradients will be multiplied with the gradient of this respective gates' output. Hence, if the local gradient is very small, it will ultimately vanish and as a result, no signal passes through the neuron to its weights. To prevent this problem, one should be careful with weight initialization and need to test with different initialization of the weights. For instance, if the initial weights are too large then most neurons would become saturated and the network does not learn [15].

Tanh: Tanh non-linear function is shown in Figure 7. This function reduces a real-valued number in the range [-1,1]. Like the sigmoid neuron, the neurons in tanh also face the vanishing gradient problem, however, unlike the sigmoid neuron its output is zero-centered. The mathematical form for Tanh function is as follows [15]:

$$F(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}, \; where \; z = w_i x_i + b_1 \tag{16}$$

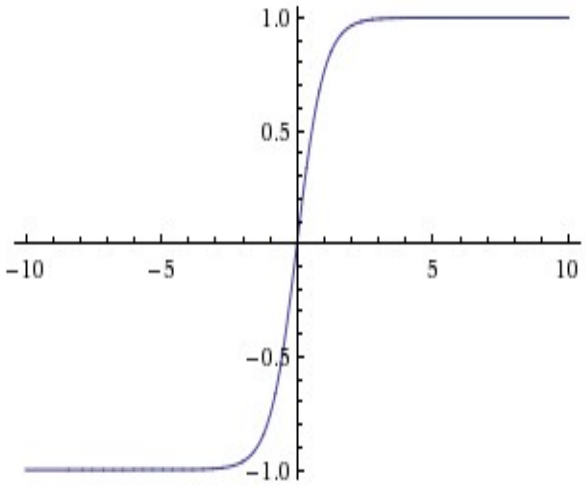

Figure 7. Pictorial representation of *tanh* function[7]

[7] Source: http://cs231n.github.io/neural-networks-1/

ReLU (Rectified Linear Unit): This function has become popular in the past few years. The mathematical function for ReLU is $f(x) = max(0, x)$ [15].

Advantages of ReLU:

1. ReLU accelerates the convergence of stochastic gradient descent compared to the sigmoid/tanh functions. It is argued that this is due to its linear, non-saturating form [19].
2. Sigmoid and Tanh neurons contain expensive procedures whereas ReLU can be implemented by simply thresholding a matrix of activations at zero [15].

Drawback: ReLU units can be unstable during training and can even die. For instance, a huge gradient flowing through a ReLU neuron could cause the weights to update in such a way that a neuron will never activate on any data point again [15].

Leaky ReLU: It is an effort to fix the drawback of ReLU, which is the dying ReLU problem. Rather than the function being zero when $x < 0$, a leaky ReLU will instead have a small negative slope (0.01 or so). That is, the function calculates $f(x) = 1(x < 0)(\alpha x) + 1(x >= 0)(x)$ where α is a small constant. The disadvantage of this activation function is that its results are not always consistent [15].

4. EXPERIMENTS AND RESULTS

Root Mean Square Error (RMSE) was used to evaluate the forecast performances of the three different neural networks. To calculate the RMSE, first residuals are calculated by taking the difference between the actual value and the predicted value, $y - \hat{y}$, where y is the actual value, and $\hat{y}$ is the predicted value. After that, the square of the differences is obtained in order to calculate the square root of the average values. RMSE can be calculated using the following formula:

$$\sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}} \tag{17}$$

In the above formula i is an individual observation, and n is the total number of observations. y_i is the actual value and $\hat{y}_i$ is a predicted value.

Figures 8, 12, and 16 show the closing prices of Amazon, S&P, and eBay, respectively. As shown in the figures, all three closing prices have an upward trend and seasonality. On the other hand, Figures 9, 13, and 17 show the decomposition of the closing price into trend, seasonality, and random/remainder components. In order to fit the model, the trend and seasonality components are removed from the original time series data. Figures 10, 14, and 18 show the architecture of the feed forward neural network with 3 layers, and Figures 11, 15 and 19 show the actual and predicted closing prices using FNN for Amazon, S & P and eBay companies, respectively. Figures 20, 21, and 22 show the actual and predicted prices using RNN for Amazon, S & P and eBay, respectively. Finally, Figures 23, 24, and 25 show the actual and the predicted prices using LSTM for Amazon, S & P and eBay, respectively.

4.1. Diagrams

4.1.1. FNN Diagrams

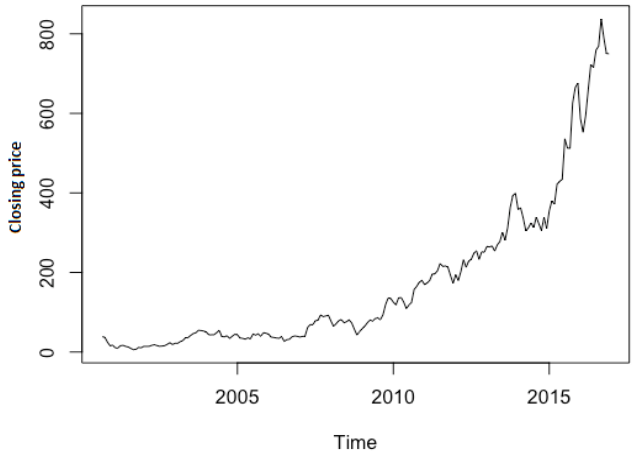

Figure 8. Closing price of Amazon

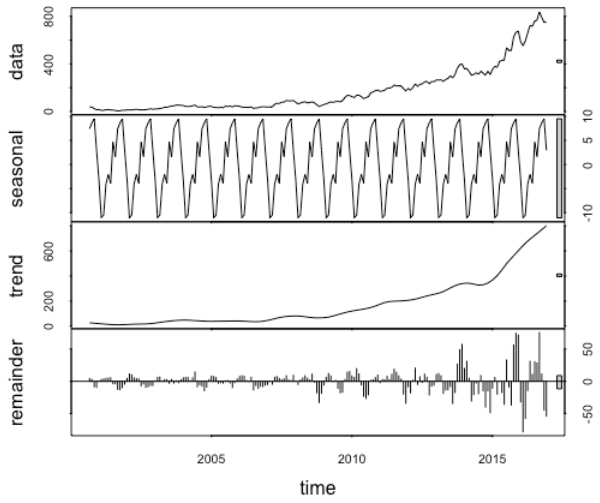

Figure 9. Decomposed Amazon closing price

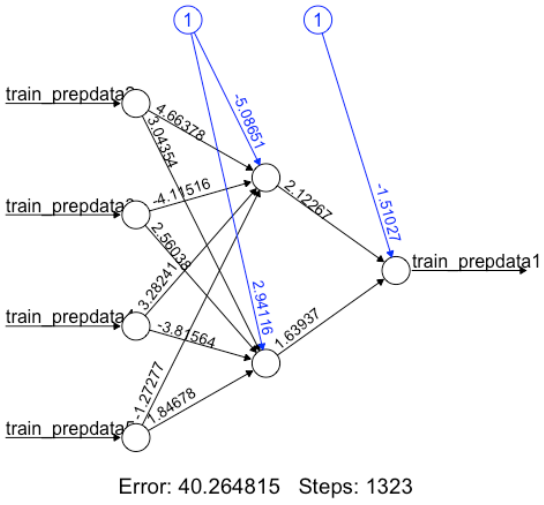

Figure 10. Feed forward neural network architecture for Amazon data

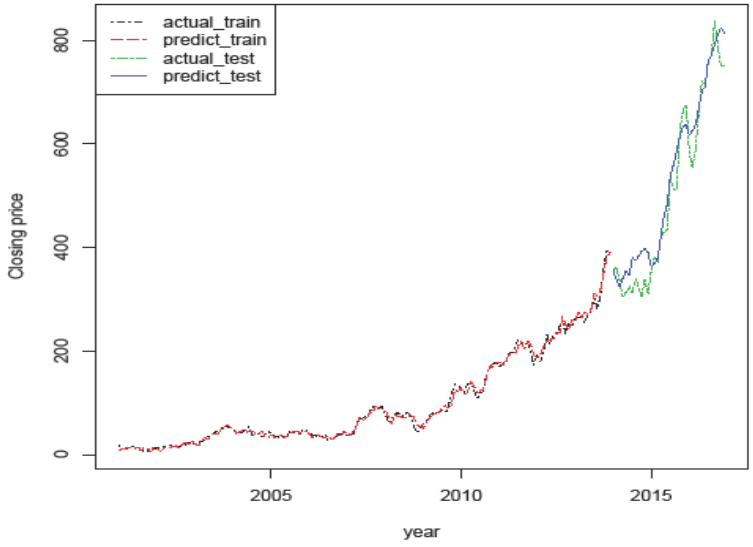

Figure 11. Actual vs Predicted values of FNN (Amazon closing price)

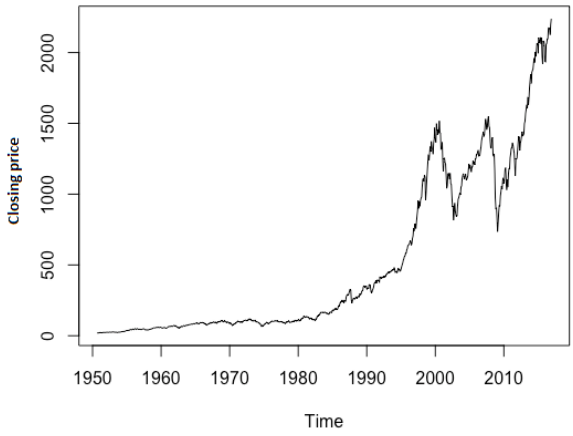

Figure 12. Closing price of S & P

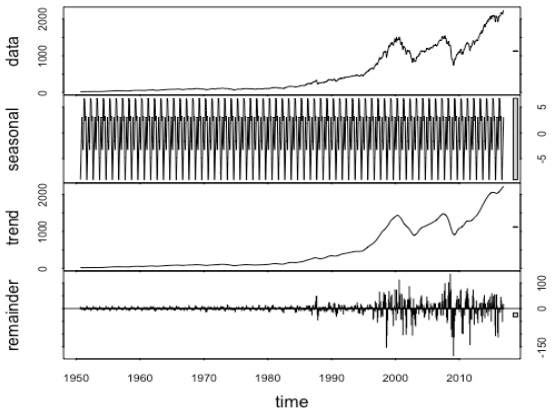

Figure 13. Decomposed S & P closing price

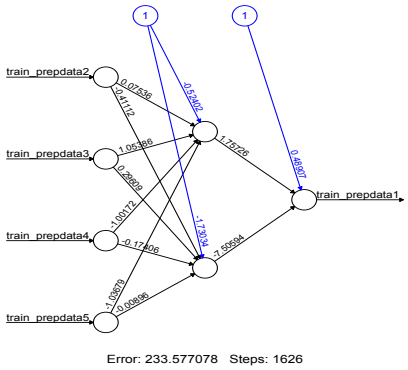

Figure 14. Feed forward neural network architecture for S & P data

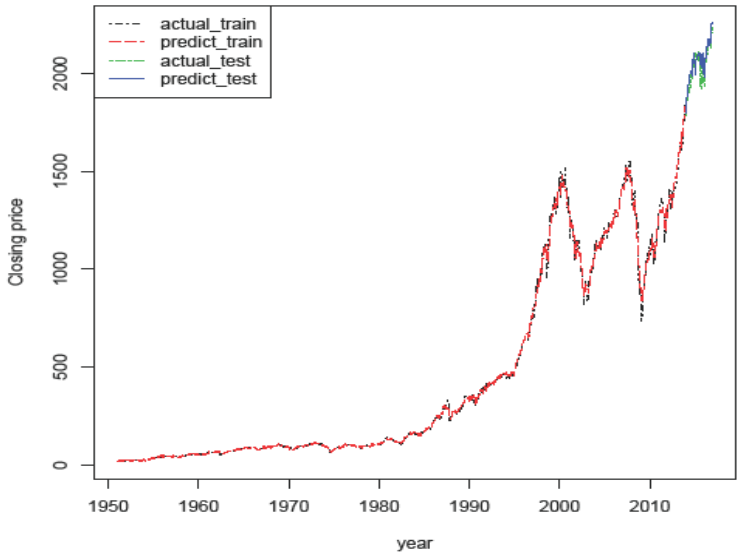

Figure 15. Actual vs Predicted values of FNN (S & P closing price)

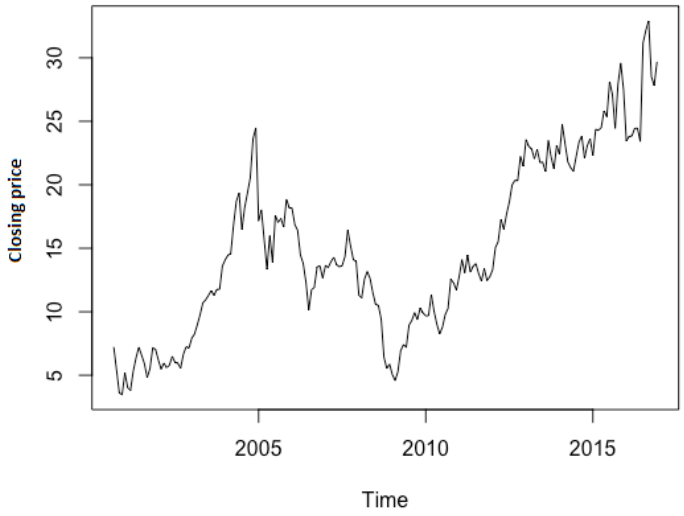

Figure 16. Closing price of eBay

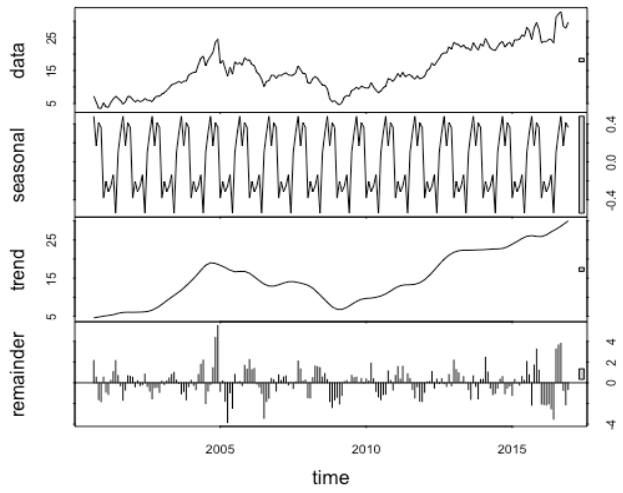

Figure 17. Decomposed eBay closing price

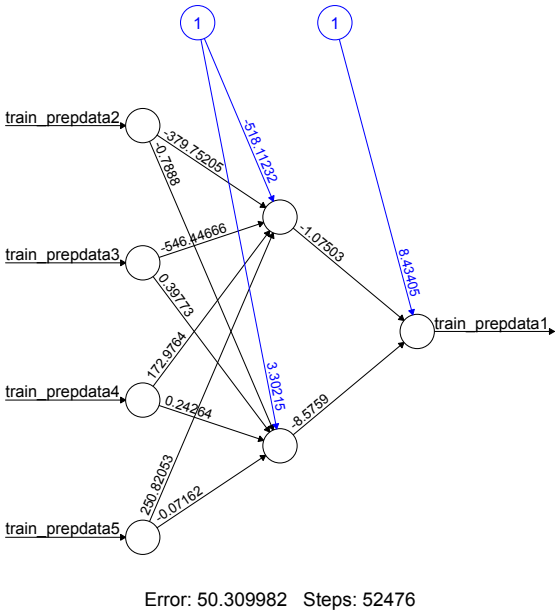

Figure 18. Feed forward neural network architecture for eBay data

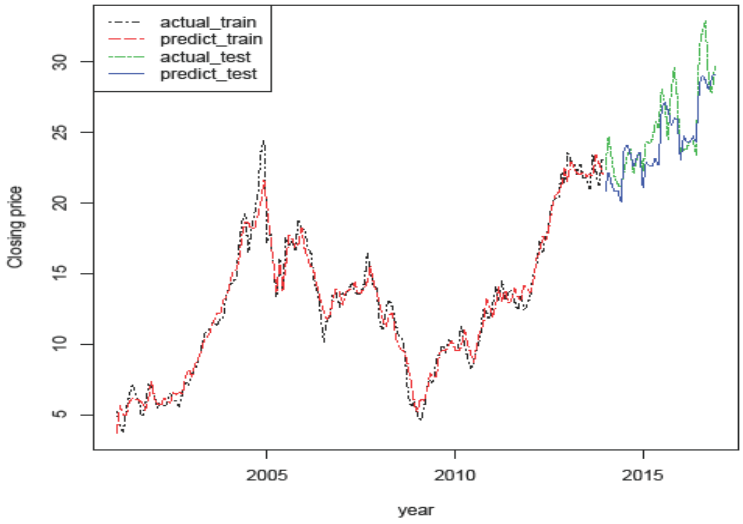

Figure 19. Actual vs Predicted values of FNN (eBay closing price)

4.1.2. RNN Diagrams

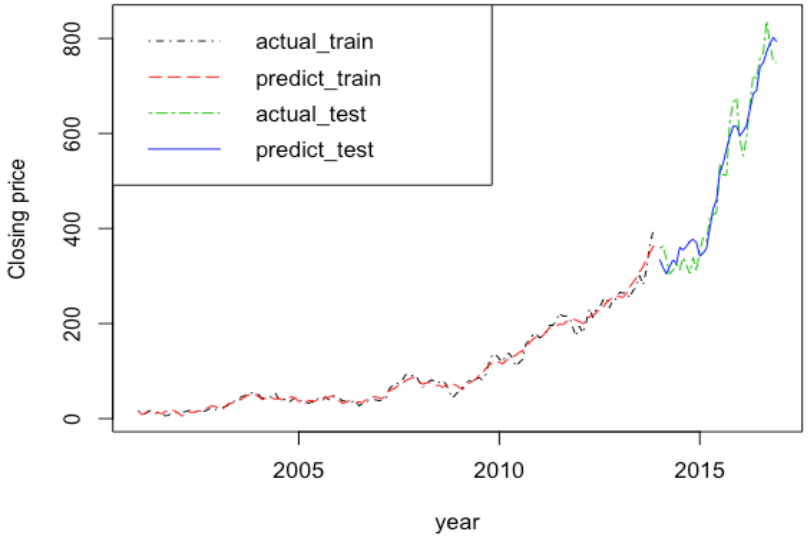

Figure 20. Actual vs Predicted values of RNN (Amazon closing price)

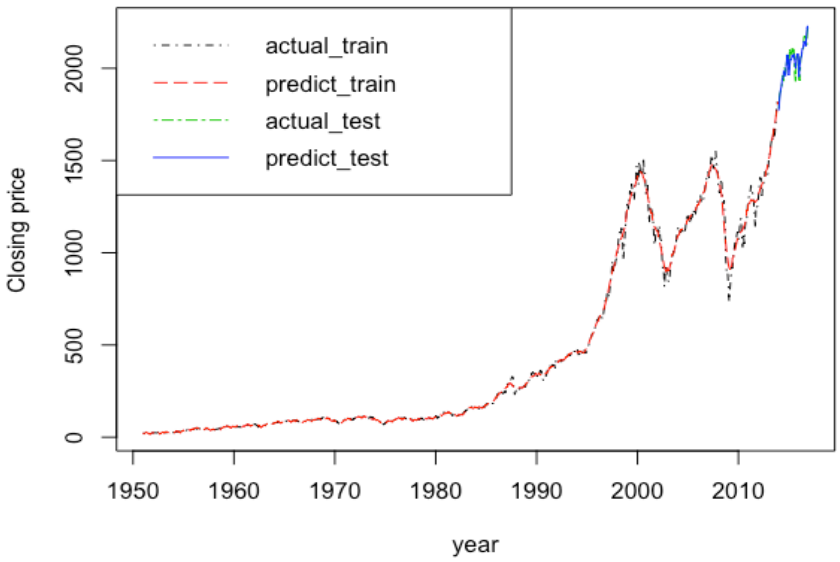

Figure 21. Actual vs Predicted values of RNN (S & P closing price)

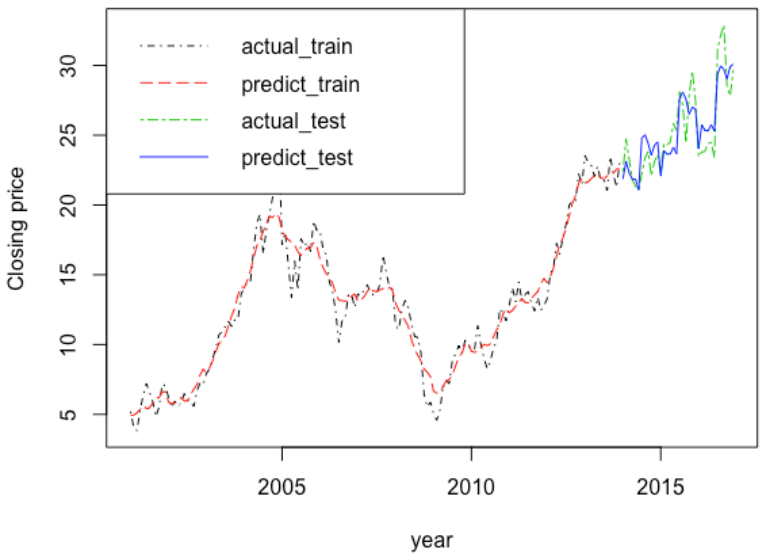

Figure 22. Actual vs Predicted values of RNN (eBay closing price)

4.1.3. LSTM Diagrams

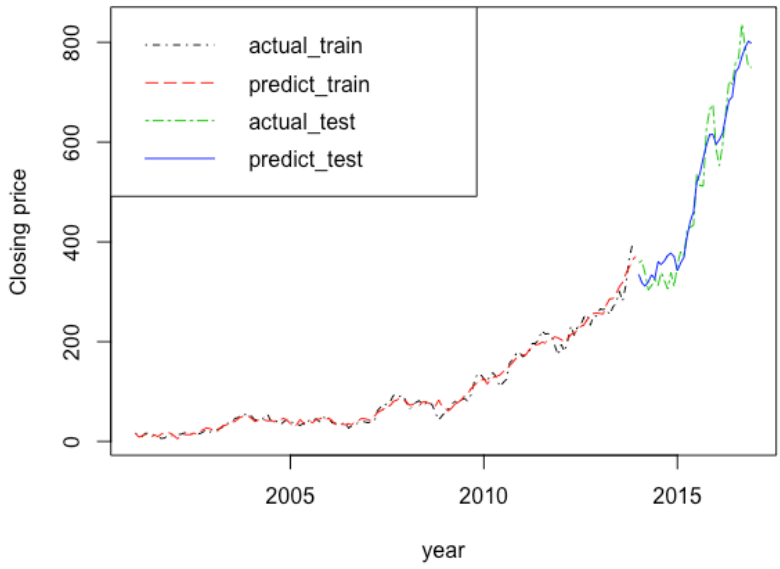

Figure 23. Actual vs Predicted values of LSTM (Amazon closing price)

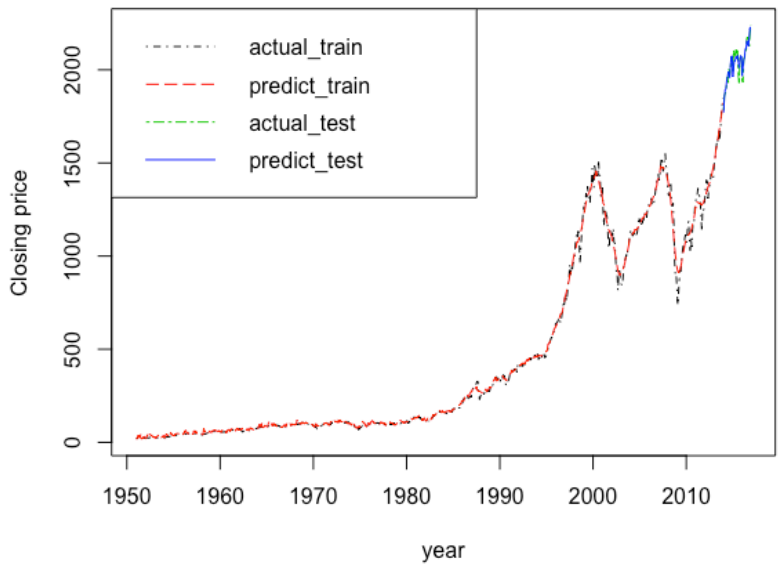

Figure 24. Actual vs Predicted values of LSTM (S & P closing price)

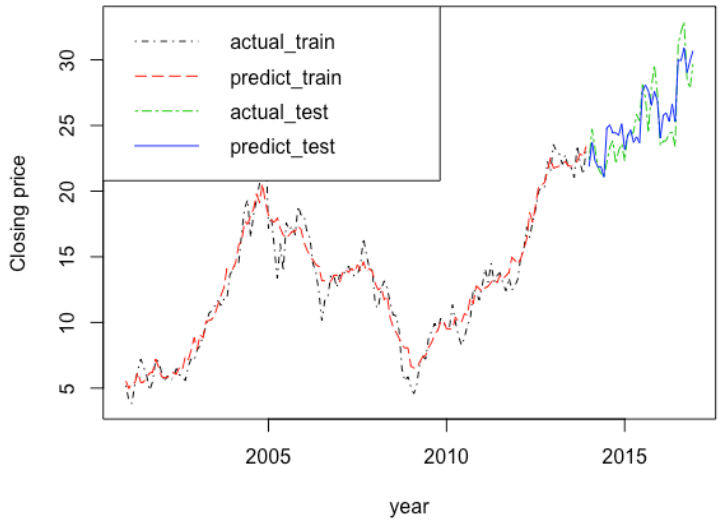

Figure 25. Actual vs Predicted values of LSTM (eBay closing price)

4.1.4. Results

First, a feedforward neural network was designed with three layers including an input layer, hidden layer and an output layer. Trial and error approach was used to choose the number of neurons in each layer. The best performing feed forward neural network has four input neurons, two hidden neurons and one output neuron as shown in the Figure 10. A logistic nonlinear activation function was used in the hidden layer.

Second, two recurrent neural networks were built including 1) vanilla recurrent neural network, and 2) Long Short-Term Memory (LSTM) recurrent neural network. Both recurrent neural networks have the same neural network structure and configuration.
The RMSE of the three different datasets executed with the three different networks are listed in Table 1. The table shows the results of three different neural networks. Based on the RMSE test error, the LSTM Neural Network performed well compared to both Feed Forward Neural Network and Recurrent Neural Network on the 3 different data sets. After

LSTM, the next best model is RNN and the least performing model is feed forward neural network. In the table 1 shown, the second column shows the datasets along with the years categorized into train and test sets. The third column shows the parameters on which the network is tuned. The fourth column shows the train error and finally the last column shows the test error. From each dataset, 80% of the data was used for training each type of neural network and 20% of the data was used for testing the trained neural network.

4.1.5. Experiments and Discussion

I performed a comparative study of the three different types of neural networks by forecasting stock market prices. Experiments were conducted on a Mac machine using the RStudio software to execute programs written in R language. Firstly, I have calculated the lag values for the closing prices of 3 different data sets. After that, I have divided the data into training (80%) and test (20%) data sets. Then, I have removed the trend and seasonality from the data and applied the z-score normalization before applying the model for faster convergence. While training the model, I have applied the logistic activation function and resilient backpropagation as the learning algorithm with 0.0001 as the learning rate. To get robust results, I ran the experiment 10 times and took the average of those results. Finally, I used the lowest Root Mean Square Error (RMSE) as the criteria to evaluate the trained model.

Table 1. Forecasting results of Amazon, S & P, and eBay companies closing prices using FNN, RNN, and LSTM

FNN	Amazon (2001-2016)	Testyears=3 learningrate=0.0001 hidden =2	Train Error 7.539903866	Test Error 41.79418785
	S&P(1951-2016)	Testyears=13 learningrate=0.0001 hidden =2	Train Error 15.55156718	Test Error 47.977658
	eBay(2001-2016)	Testyears=3 learningrate=0.0001 hidden =2	Train Error 0.9611230537	Test Error 1.655409189
RNN	Amazon (2001-2016)	Testyears=3 Learning rate=0.0001 Hidden_dim=4 Num_epochs=10 Batch_size=5 learningrate_decay =2	Train Error 10.78449	Test Error 36.29156
	S&P(1951-2016)	Testyears=13 Learning rate=0.01 Hidden_dim=10 Num_epochs=10 Batchsize=10 learningrate_decay =2	Train Error 18.7136384	Test Error 47.37440424
	eBay(2001-2016)	Testyears=3 Learning rate=0.01 Hidden_dim=10 Num_epochs=10 Batchsize=10 learningrate_decay =2	Train Error 1.271471719	Test Error 1.540293374
LSTM	Amazon (2001-2016)	Testyears=3 Learning rate=0.01 Hidden_dim=4 Num_epochs=10 learningrate_decay =2 Batch_size=5	Train Error 10.59769	Test Error 36.09783
	S&P(1951-2016)	Testyears=13 Learning rate=0.01 Hidden_dim=10 Num_epochs=10 Batchsize=10 learningrate_decay =2	Train Error 18.89344805	Test Error 47.28993449
	eBay(2001-2016)	Testyears=3 Learning rate=0.01 Hidden_dim=10 Num_epochs=10 Batchsize=10 learningrate_decay =2	Train Error 1.220629173	Test Error 1.503575794

5. CONCLUSION

This project compares the performance of different neural networks including feed forward neural networks, vanilla recurrent neural networks, and long short-term memory recurrent neural networks using different datasets. The datasets include the closing prices of Amazon, S&P, and eBay stocks.

In all the three data sets of different companies, the long short-term memory recurrent neural networks performed well compared to the others. The next best performing model is vanilla recurrent neural networks across all three closing prices of companies. The performance of the recurrent neural networks can be explained due to their nature of remembering the most recent past information in the recurrent states. In recurrent neural networks, long short memory networks performed well because it has the ability to remember both long-term information as well as short-term information of the price in consideration.

Future work could be focused on big data using more advanced neural networks such as deep neural networks.

REFERENCES

[1] Oancea, Bogdan, and Ştefan Cristian Ciucu. "Time series forecasting using neural networks." *arXiv preprint arXiv:1401.1333* (2014).

[2] Kar, Abhishek. "Stock prediction using artificial neural networks." *Dept. of Computer Science and Engineering, IIT Kanpur* (1990).

[3] Brownlee, Jason. "Supervised and unsupervised machine learning algorithms." *Machine Learning Mastery* (2016), machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/

[4] “Yahoo Finance - Business Finance, Stock Market, Quotes, News.” Yahoo! Finance, Yahoo!, finance.yahoo.com/

[5] “Forecasting.” Wikipedia, Wikimedia Foundation, 15 Feb. 2019, en.wikipedia.org/wiki/Forecasting.

[6] Chakraborty, Kanad, et al. "Forecasting the behavior of multivariate time series using neural networks." *Neural networks* 5.6 (1992): 961-970.

[7] Jain , Aarshay . " Complete guide to create a Time Series Forecast (with Codes in Python) ." *Analytics Community | Analytics Discussions | Big Data Discussion* (2016).

[8] Xu, Selene Yue, and C. U. Berkely. "Stock price forecasting using information from Yahoo finance and Google trend." *UC Brekley* (2014).

[9] Pant, Neelabh. " A Guide For Time Series Prediction Using Recurrent Neural Networks (LSTMs). " *Stats and Bots*, Stats and Bots (2017).

[10] Brownlee, Jason. "What Is Time Series Forecasting? - Machine Learning Mastery." *Machine Learning Mastery* (2016).

[11] Hiray, Jagdish. "Time-series methods of forecasting | All about Business and management." *All about Business and management | A blog devoted to business management and processes* (2008).

[12] Naeini, Mahdi Pakdaman, Hamidreza Taremian, and Homa Baradaran Hashemi. "Stock market value prediction using neural networks." *Computer Information Systems and Industrial Management Applications (CISIM), 2010 International Conference on*. IEEE, 2010.

[13] Kuan, Chung-Ming, and Tung Liu. "Forecasting exchange rates using feedforward and recurrent neural networks." *Journal of pplied econometrics* 10.4 (1995): 347-364.

[14] Gamboa, John Cristian Borges. "Deep Learning for Time-Series Analysis." *arXiv preprint arXiv:1701.01887* (2017).

[15] Karpathy, Andrej. "Cs231n convolutional neural networks for visual recognition." *Neural networks* 1 (2016).

[16] Zhang, G. Peter. "Time series forecasting using a hybrid ARIMA and neural network model." *Neurocomputing* 50 (2003): 159-175.

[17] Claveria, Oscar, and Salvador Torra. "Forecasting tourism demand to Catalonia: Neural networks vs. time series models." *Economic Modelling* 36 (2014): 220-228.

[18] Cheng, Haibin, et al. "Multistep-ahead time series prediction." *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, Berlin, Heidelberg, 2006.

[19] Britz, Denny. "Recurrent Neural Networks Tutorial, Part 1–Introduction to RNNs." (2015).

[20] Heaton, Jeff. "T81_558_deep_learning/t81_558_class3_tensor_flow.ipynb at master · jeffheaton/t81_558_deep_learning · GitHub." The world's leading software development platform · GitHub (2017).

[21] Olah, Christopher. "Understanding lstm networks–colah's blog." *Colah. github. io* (2015).

[22] Li, Fei-Fei, Andrej Karpathy, and Justin Johnson. "CS231n: Convolutional neural networks for visual recognition." *University Lecture* (2015).

[23] Heaton, Jeff. "T81_558_deep_learning/t81_558_class6_backpropagation.ipynb at master · jeffheaton/t81_558_deep_learning · GitHub." The world's leading software development platform · GitHub (2017).

[24] Raschka, Sebastian. "About feature scaling and normalization." *Sebastian Racha. Disques, nd Web. Dec* (2014).

[25] *RMS Error*, statweb.stanford.edu/~susan/courses/s60/split/node60.html.

[26] *CS231n Convolutional Neural Networks for Visual Recognition*, cs231n.github.io/neural-networks-3/

[27] "Deep Learning Fundamentals." *Cognitive Class*, 16 July 2018, cognitiveclass.ai/courses/introduction-deep-learning/.