Forecasting Stock Market Prices Using Advanced Tools of Machine Learning

by

Md. Tawhid Anwar
15301007
Saidur Rahman
16201004

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science and Engineering

Department of Computer Science and Engineering
Brac University
April 2019

Declaration

It is hereby declared that

1. The thesis submitted is our own original work while completing degree at BRAC University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Md. Tawhid Anwar	Saidur Rahman
15301007	16201004

Approval

The thesis titled "Forecasting Stock Market Prices Using Advanced Tools of Machine Learning" submitted by

1. Md. Tawhid Anwar (15301007)
2. Saidur Rahman (16201004)

Of Spring, 2019 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science and Engineering on April 25, 2019.

Examining Committee:

Supervisor:

Mahbub Majumdar, PhD
Professor
Department of Computer Science and Engineering
Brac University

Program Coordinator:

Jia Uddin, PhD
Associate Professor
Department of Computer Science and Engineering
Brac University

Head of Department:

Md. Abdul Mottalib, PhD
Professor and Chairperson
Department of Computer Science and Engineering
Brac University

Abstract

Stock market, a very unpredictable sector of finance, involves a large number of investors,buyers and sellers. Stock market prediction is the act of attempting to see the long run price of an organization stock or different money instrument listed on a monetary exchange. People invest in stock market supported some prediction. For predicting, the stock exchange prices people search such ways and tools which can increase their profits, whereas minimize their risks. Prediction plays a awfully necessary role available market business that is sophisticated and difficult method. A part of our thesis will be directed at assembling the required tools to aggregate financial data from various sources. Here, we proposed a model by applying machine-learning to technical analysis. Technical analysis reviews the past direction of prices using stock charts to anticipate the probable future direction of that security's price. In alternative words, technical analysis uses open, close, high and low prices, still as its volume information to construct stock chart to work out that direction the protection ought to take, supported its past information. An artificial trader can use the ensuing forecasting models to trade on any given stock market. The performance of the research is assessed using Dhaka Stock Exchange data.

Keywords: Stock Market; Dhaka Stock Exchange; Technical Analysis; Machine Learning; Neural Network; Prediction; Random Forest; Logistic Regression Analysis

Dedication

We would like to dedicate this thesis to our loving parents ...

Acknowledgement

We would like to express our gratitude to our supervisor, Professor Mahbub Majumdar of the Computer Science and Engineering department at Brac University. The door to Professor Majumdar's office was always open whenever we were in need of help or had questions about our research or writing. He consistently allowed this paper to be our own work, but steered us in the right direction whenever he thought we needed it.

We are also hugely indebted to Golam Rabiul Alam, Associate Professor of Brac University, for sharing his ideas and experiences, which helped us to make right choices in the implementation phase.

We would also like to thank our family for the support they provided me throughout our entire life. We will never forget that it is our parents' sacrifices that made our education possible.

Nevertheless, we would like to express our gratitude to Department of Computer Science and Engineering, Brac University and our teachers for helping us with all the necessary support.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

For several years, a primary topic of study for the money corporations and people has been "Predicting the future of stock prices". The financial goal behind the ability has been to predict how buying and selling stocks could be done at profitable positions. Historically, to measure stocks and generate trade signals there are 2 main classes. These are fundamental analysis and technical analysis. A third type of analysis, called sentiment analysis, has also contributed to get trade signals during recent years.

Company's basic information like the statement, record, cash-flow statement, news unleash and company policies are involved in Fundamental analysis. On the opposite hand, the underlying movement of the stock price and listed volume are evaluated in technical analysis. The users of this technique believe that each one accessible info and in addition to that, a company's fundamentals, is already contained at intervals the price history. Technical analysis will not be qualitative in nature. It's goal is distinguishing regularities by extracting patterns from uproarious knowledge and visually examining graphs of stock price movements, trendy breakthroughs in technology. Recently, machine learning algorithms have played a crucial role in this technique and into a lot of quantitative tool. Studies show that about 18th to 19th of polled professionals and individual investors have confidence a minimum of some variety of technical analysis. Sentiment analysis [19] [24]. incorporates the quantification of the the market reaction to clearly marked news events or earnings reports. This method has additionally recently shown promising value. Together, both technical analysis indicators and machine learning algorithms are used in this paper.

The lack of available operating trading methods during this area is most common due to the dearth of benefit to publish profitable systems. With bigger financial benefits from the exploiting or selling of such systems, there has always been an inherent negative bias in tutorial literature. Some people suggest that the reason for the scarcity of published operating models is because of the issue of business empirical results. These results are usually virtually statistically not very significant which leading to a possible "file drawer" bias in published studies [4]. Classical types of stock exchange prediction are not commonly profitable, because they will not be able to adapt to the dynamical conditions of market.

Indeed, markets do not remain stable and the indicators that have robust prophetical price over one period might stop to provide excess returns as presently as there is a

change in market conditions. There are 2 measures, which are proposed to fight this evolving market behavior. One, some trading systems area unit supported genetic algorithms. These algorithms rework the symptoms that are used as attributes over time [16]. Two, additional commonly, the data set is suitable nonlinear models using machine learning algorithms such as Artificial Neural Networks [24].

In this paper, we will take the latter approach and we will examine the accuracy and performance of assorted machine learning classification algorithms. We will be using each quantitative technical information as options. During the process of making a trading strategy that outperforms the market, we will be implementing feature generation, and also cross-validation to find out best parameters and regularizing parameters so that over-fitting is avoided. By aggregating a 10-year amount of Dhaka Stock Exchange market information, we train and then we optimize the different models to predict stock value movements. Finally, we then conduct out of sample tests to look at the profitability of the trading systems compared to an easy buy-and-hold strategy.

1.1 Methodology

From previous works, we have found various researchers using different methods to predict stock prices or to identify patterns of stock trends. To make improvements to the research works that we have studied we made several optimizations and improvements in order to make a substantial contribution towards this research. Most of the research has been done on either SP500 index or using data from Google Trends, and very few works have been done on Bangladesh stock data. We have taken a novel approach of using the advanced machine learning tools on Bangladesh stock market data, specifically Dhaka Stock Exchange data.

Our working methodology goes as follows: Selected technical indicators have been applied to the stock market data. An initial decision was obtained using the close price differences between consecutive days. The feature variables were the technical indicator values along with the initial column values of the stock. The target variable was the decision of either Buy, Sell or Hold. These feature and target variables were used to train the machine learning models. Models used include: Logistic Regression, SVM classifier, Random Forest, Deep Neural Network, RNN LSTM.

- **Stock Market Analysis:** We have leveraged the accurate technical indicators for our analysis of stock market. This is primarily because technical indicators have been well proven to accurately provide values to get stock market data trends not just in terms of stock price but also price volume.
- **Data Preparation:** The selected technical indicator formulae have been applied to the stock market data to produce the data set that will serve as the bedrock to our research. Each of the technical indicators provided us with a value for each entry of stock data. The technical indicator values were stored in columns. Each column corresponding to the data of a different technical indicator.
- **Feature Generation:** The technical indicator values along with the original stock data were used as the feature variable for the machine learning training.

Because we have used a supervised learning approach, therefore we needed a class label column. The class labels were calculated using cross-validation on the close price data.

- **Applied Different Algorithms:** Having pre-processed our data, we then split the whole data set into train and test set using 70 percent of the data for training set and the rest as the test set. This was done to avoid over fitting our machine learning models. Now, we scaled the dataset data because otherwise, the model would not be able to learn effectively from the data because each of the columns show a different distribution of data. We trained the above mentioned models to train the data and received the corresponding accuracy values.

- **Result Analysis:** Based on the accuracy value of the models, we found that the most accurate model for this research turned out to be the Deep Nerual network mode. It showed a staggering 94.29 percent accuracy value making it the most accurate compared to other models. Next up, the RNN LSTM model was the second best accurate model and it was followed by Random Forest, Logistic Regression and SVM models respectively in order of decreasing accuracy values. The least accurate was the SVM model with an accuracy value of 61.22 per cent.

1.2 Outline of this Paper

- **Chapter 2:** This chapter reviews previous research involving machine learning methods applied for forecasting stock price.
- **Chapter 3:** This chapter describes different machine learning algorithms used for the classification problem, detailing their working principle.
- **Chapter 4:** This chapter describes how technical analysis techniques used to generate trading system summarizing the general concepts behind each indicators.
- **Chapter 5:** This chapter explains how we prepared and generated the technical indicators and stock data-sets to be used for machine learning.
- **Chapter 6:** This chapter details the accuracy and performance of the different machine learning techniques on the training, validation and testing data sets.
- **Chapter 7:** This chapter compares different machine learning algorithms and shows the result.
- **Chapter 8:** This chapter concludes our findings and proposes future research to build a better trading model using machine learning algorithms

Chapter 2

Literature Review

A concise review of previous empirical studies that have employed machine learning algorithms to construct trading strategies will be presented in this section.

2.1 Previous Research

The Stock market prediction has been an important exertion in business and finance for many years. Correct prediction of stock market is very important for the investors to determine that if it would be better to buy any specific stock or not. There have been a significant number of studies and analysis done by many enthusiasts who applied previously established prediction models to acquire more accuracy.Artificial Neural Network based method is the first technique to be used for the stock market trend prediction [16]. In some cases,for complicated networks, reliability of results may decrease [15].

Neural networks, in short, NNs, have been a popular method in financial time-series analysis and forecasting. NNs also offer a great deal of flexibility in the type of architecture of the model and also in terms of the number of hidden nodes and hidden layers. Indeed, Pekkaya and Hamzacebi have taken an approach to conduct a comaprative study on the results from using a linear regression in comparison to a a NN model. The objective of their research was to forecast macro variables. Their intention was to show that the NN gives much better results compared to the linear regression. [15].

NNs have been employed in a number of studies and NNs have been successful in showing promising results in the financial markets. Grudnitski and Osburn have constructed NNs in order to forecast SP500 and Gold futures price directions. The findings of Grudnitski and Osburn show that they were able to correctly predict the direction of monthly price changes with 75% and 61% respectively [1]. We found that another study showed an NN-based model leads to greater arbitrage profits in comparison to cost of carry models [3]. Phua, Ming and Lin have constructed a NN using Singapore's stock market index. Their research work shows a forecasting accuracy of 81% [6]. In a similar research, Neural Network models which have been applied to weekly forecast of Germany's FAZ index have shown favorable predictive results in comparison to conventional statistical approaches [11].

In a recent research paper, NNs have been adapted to show improved performance

on financial time series forecasting. In their research, Shaoo et al. have showed that cascaded functional link artificial neural networks (CFLANN) have performed the best in FX markets [6]. Also, Egrioglu et al. have taken an approach to introduce a new method based on feed forward artificial neural networks. The purpose was to analyze multivariate high order fuzzy time series forecasting models [19]. In another research, Liao and Wang have used a stochastic time effective neural network model. Their model shows predictive results on the global stock indices [17]. A reseach conducted by Bildirici and Ersin have done the work of combining NNs with ARCH/GARCH and other volatility based models. This was do in order to produce a model that shows superior performance in contrast to ANNs or GARCH based models alone. [17]. Not only that, we also see Yudong and Lenan used bactrial chemotaxis optimization (BCO) and back-propagation NN on SP500 index. They have come to a conclsion that their hybrid model (IBCO-BP) offers less computational complexity, a far better prediction accuracy and much less training time [16].

From [26], we have gathered active experiences by victimisation the sample information sets offered and learning to use the Pandas library for improvement and data format the info as needed. In [25], the prediction performance of Fuzzy time-series models, a novel fuzzy statistic model for stock value prediction was bestowed, that model elaborates the conception of advanced network . First, the discourse is split into intervals victimisation fuzzy clustering technique with the midpoints of two adjacent cluster.To achieve prediction performance it follows the adjustive expectation model then supported the advanced networks to boost prediction accuracy, we tend to modify the prognostication value of the long run observes the error rate to regulate prognostication of future stock prices.

In [19] the forecasting was carried out by the the conventional time series analysis technique. The information was extracted from the Google trend website and the Yahoo finance website to predict weekly changes in stock price. The next step was that important news/events related to a particular stock of choice over a 5-year span are recorded. Then, the weekly Google trend index values on this stock are used to provide us with a measure of the strength or magnitude of these events. A significant correlation has been seen in the result of this experiment. The correlation was among the changes in weekly stock prices and the values of important events or news calculated from the Google trend website.

In order to further enhance the accuracy, some studies have employed the random forest algorithm for classification. This technique will be further discussed in the third chapter of our research. In another research, Booth et al. have shown that a regency-weighted ensemble of random forests have enabled them to get superior results when analyzed on a large sample of stocks from the DAX. The results were superior both in terms of both profitability and prediction accuracy, when compared with other ensemble techniques [22]. In a similar fashion, a gradient boosted random forest model was applied to Singapore's stock market. This model was able to produce excess returns in comparison with a buy-and-hold strategy [20]. A number of recent time researches have combined decision tree analysis with evolutionary algorithms. This is expected to enable the model to adapt to changing market conditions. In their research, Hsu et al. have presented constraint-based evolutionary

classification trees (CECT) and it has shown a strong predictability of a company's financial performance [13].

Another technique used in the prediction of market behaviour are Support Vector Machines (SVM). In a research conducted by Huang et al., they have compared SVM with other classification methods. The comparison was done with random Walk, linear discriminant analysis, quadratic discriminant analysis and elman back-propagation neural networks. It was found that SVM performed the best in forecasting weekly movements of the Nikkei 225 index [12]. In a similar fashion, Kim has taken an approach to compare SVM with NN and case-based reasoning. Kim found that SVM outperformed in forecasting the daily direction of change in the Korea composite stock price index (KOSPI) [9].

Similarly, a margin-varying Support Vector Regression model was used by Yang et al. and it has been able to show empirical results that have good predictive value for the Hang Seng Index [7]. A system was proposed by Nair et al. The system is a genetic algorithm optimized decision tree-support vector machine hybrid. The system validates its performance on the BSE-Sensex. It was found that its predictive accuracy is superior in comparison to not ony an NN but also a Naive bayes based model [18].

In [21], stock market forecast was done using technical analysis to improve the accuracy of the forecasting so that investors can take better decisions and to help them maximize their profits.There are 3 different ways that the improvement was done. Firstly, trend-based stock classification was done. Here, the stocks were differentiated into long-term and short-term investment categories. After classification of the stocks, these were classified further using series data capture of closing price, stock return calculation and two other methods. Secondly, adaptive stock market indicator selection was used. Here, quantitative index is used to analyze past price data and predict future trends effectively. Lastly, the third method is Stock Market Trading Signal Forecasting. Here, the technical indicators were normalized and then a sample training dataset was prepared with values obtained from the normalized indicator data and finally the forecasting model was ready for prediction.

Chapter 3

Machine Learning Algorithms

In chapter 6, we will be reviewing the algorithms of machine learning and we will generate the trading systems. The general concepts underlying each algorithm will be summarized and we will detail the free parameters which will be determined through the principle of cross validation.

3.1 Artificial Neural Network

Artificial Neural Networks are software implementation of the network of neurons present in the human brain [26]. The neurons in the human brain can be thought of as organic switches, since the neurons, depending on the strength of their electrical or chemical input, can change their output state. The neurons have millions of connections with other neighboring neurons. This extremely complex neural network of the human brain allows the human brain to carry out its learning function through the process of activation of particular neural connections over others. As a result. this will reinforce those connections. This learning also includes "feedback", meaning, when the expected outcome occurs, the neural connections because of which it occured, those connections will become strengthened [23].

Artificial Neural Networks (ANNs) used in Artificial Intelligence tries to mimic the aforementioned behavior of the human brain. We will be using supervised ANNs in our thesis paper to predict stock movement and provide buy/sell decision. In a supervised artificial neural network, the training process of the network is achieved by providing matched input and output data samples. The goal will be to get the Artificial Neural Network to provide a output that is desired for a given input. This learning will take place via adjusting of the weights of the Artificial Neural Network connections. For example: If we consider an e-mail spam filter. In this case, the input training data can be considered to be the count of various words in the body of the e-mail. Output training data can be a classification of whether the e-mail was truly spam or not. If many examples of e-mails are passed through the neural network this allows the network to learn what input data makes it likely that an e-mail is spam or not[5].

Figure 3.1 Shows 3 layers of a neural network. In this figure, Layer 1 (L1) represents the input layer, where the external input data enters the network. Layer 2 (L2) is called the hidden layer as this layer is not part of the input or output. Layer 3 (L3) is the output layer. As can be seen, each node in L1 has a connection to all

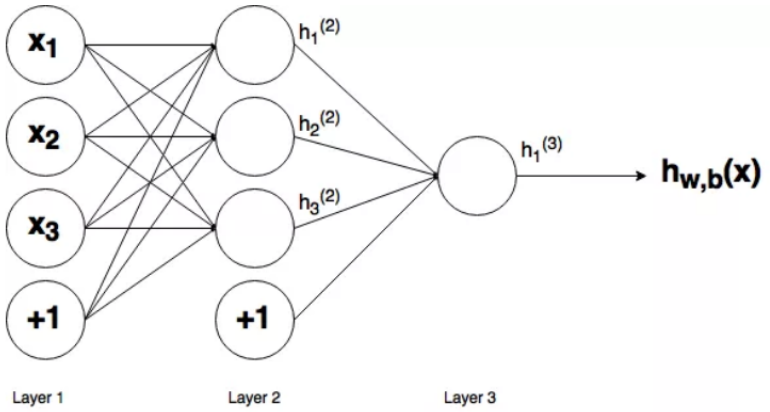


Figure 3.1: A three layer neural network

the nodes in L2. Likewise for the nodes in L2 to the single output node L3. Each of these connections will have an associated weight. In figure 3.1, the (+1) bias is connected to each of the nodes in the subsequent layer. So the bias in layer 1 is connected to the all the nodes in layer two. Because the bias is not a true node with an activation function, it has no inputs (it always outputs the value +1).
Output of the neural network is shown in equation 3.1

$$\begin{aligned} h_1^{(2)} &= f(w_{11}^{(1)}x_1 + w_{12}^{(1)}x_2 + w_{13}^{(1)}x_3 + b_1^{(1)}) \\ h_2^{(2)} &= f(w_{21}^{(1)}x_1 + w_{22}^{(1)}x_2 + w_{23}^{(1)}x_3 + b_2^{(1)}) \\ h_3^{(2)} &= f(w_{31}^{(1)}x_1 + w_{32}^{(1)}x_2 + w_{33}^{(1)}x_3 + b_3^{(1)}) \\ h_{W,b}^{(x)} = h_1^{(3)} &= f(w_{11}^{(2)}h_1^{(2)} + w_{12}^{(2)}h_2^{(2)} + w_{13}^{(2)}h_3^{(2)} + b_1^{(2)}) \end{aligned} \tag{3.1}$$

The function f refers to the node activation function. In this case the sigmoid function is used. This function is "activated" i.e. it moves from 0 to 1 when the input to the function is greater than a certain value. The sigmoid function is shown in equation 3.2

$$f(z) = \frac{1}{1 + exp(-z)} \tag{3.2}$$

3.1.1 Gradient Descent

In training the network with the (x,y) pairs, the goal is to get the neural network better and better at predicting the correct y given x [26]. This is performed by varying the weights so as to minimize the error. we know how to vary the weights, given an error in the output of the network by using the concept of gradient descent [23].
To proceed with this method, first the gradient of the error with respect to w is calculated at point "1". if it is positive with respect to an increase in w, a step in that direction will lead to an increase in the error. If it is negative with respect to an increase in w (as it is in the diagram above), a step in that will lead to a decrease in

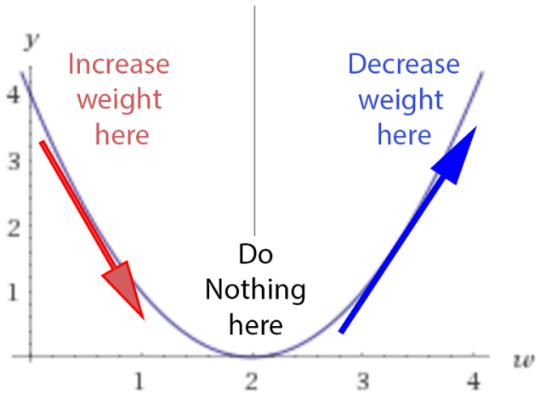


Figure 3.2: How weights are adjusted in a Neural Network

```
x_old = 0 # The value does not matter as long as abs(x_new - x_old) > precision
x_new = 6 # The algorithm starts at x=6
gamma = 0.01 # step size
precision = 0.00001

def df(x):
    y = 4 * x**3 - 9 * x**2
    return y

while abs(x_new - x_old) > precision:
    x_old = x_new
    x_new += -gamma * df(x_old)

print("The local minimum occurs at %f" % x_new)
```

Figure 3.3: The Gradient Descent implementation in Python code

the error. The gradient descent method uses the gradient to make an informed step change in w to lead it towards the minimum of the error curve. This is an iterative method that involves multiple steps. Each time, the w value is updated according to equation 3.3

$$w_{new} = w_{old} - \alpha \times \delta_{error} \tag{3.3}$$

The code we have used to implement Gradient Descent using Python code is shown in Figure 3.4

3.1.2 Backpropagation

Backpropagation is about determining how changing the weights impact the overall cost in the neural network. What we want to do is minimize the cost function $J(\theta)$ using the optimal set of values for θ (weights). Backpropagation is a method we use in order to compute the partial derivative of $J(\theta)$ This partial derivative value is then used in Gradient descent algorithm for calculating the θ values for the Neural Network that minimize the cost function $J(\theta)$.

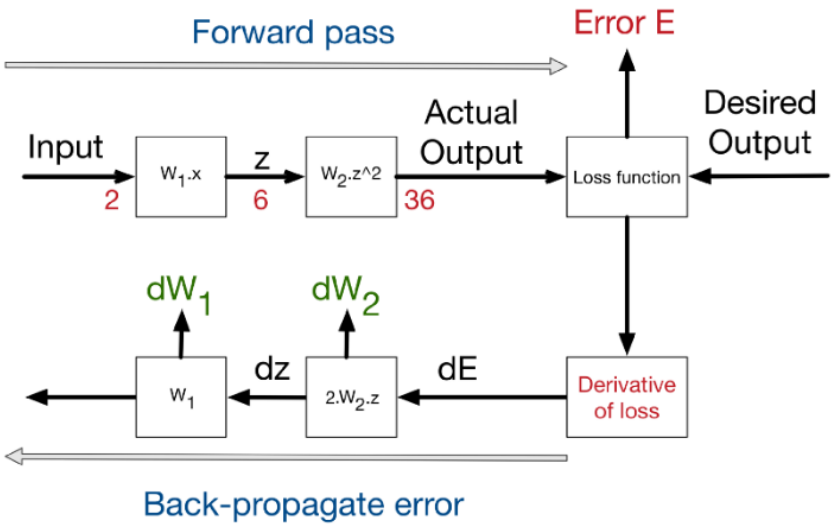


Figure 3.4: Diagram of Forward and Backward Paths

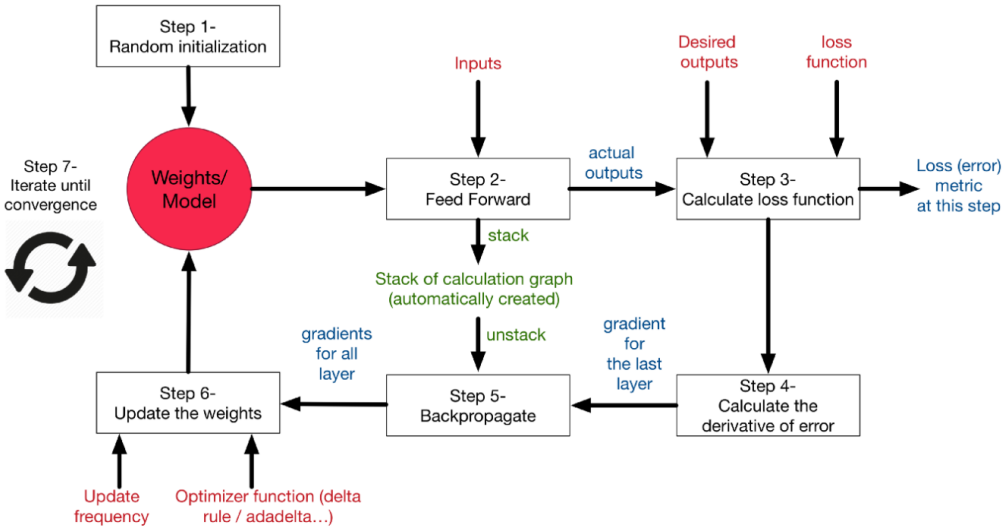


Figure 3.5: Summarized learning process on neural networks

Backpropagation algorithm has 5 steps:

1. Set $a(1) = X$; for the training examples
2. Perform forward propagation and compute $a(l)$ for the other layers (l = 2...L)
3. Use y and compute the delta value for the last layer $\delta(L) = h(x) - y$
4. Compute the $\delta(l)$ values backwards for each layer
5. Calculate derivative values $\delta(l) = (a(l))^T \cdot \delta(l+1)$ for each layer, which represent the derivative of cost $J(\theta)$ with respect to θ(l) for layer l

The forward and backward Paths taken in a Neural Network have been shown in Figure 3.5 and Figure 3.6 summarizes the entire mechanism of a neural network

3.2 Deep Neural Networks

Deep Neural Networks outperform shallow feed forward Neural Networks because shallow networks contain less number of hidden layers, and as a result, the learning is less accurate than a Deep Neural Network [5].

Deep Neural Network architecture design used in this paper

For our purpose we have implemented a Deep Neural Network using Tensorflow library. Out network consists of 61 input nodes (since this is the number of feature variables that we will be using) and it contains 2 hidden layers. One of the hidden layer contains 300 neurons and the other hidden layer contains 100 neurons respectively. The output layer contains 3 neurons since we have 3 class labels namely 0 (hold), 1 (sell), and 2 (buy). The Deep Neural Network that we have designed is shown in Figure 3.7

3.3 Recurrent Neural Networks

A recurrent neural network, at its most fundamental level, is simply a type of densely connected neural network. However, the key difference to normal feed forward networks is the introduction of time – in particular, the output of the hidden layer in a recurrent neural network is fed back into itself.
In figure 3.8, we have a simple recurrent neural network with three input nodes. These input nodes are fed into a hidden layer, with sigmoid activations, as per any normal densely connected neural network. The output of the hidden layer is fed back into the same hidden layer. The hidden layer outputs are passed through a conceptual delay block to allow the input of previous output ($h^{(t-1)}$into the hidden layer. As a result, we can now model time or sequence-dependent data. On the left-hand side of the above diagram, we have basically the same diagram as the first (the one which shows all the nodes explicitly) [26].
What the previous diagram neglected to show explicitly was that we in fact only ever supply finite length sequences to such networks – therefore we can unroll the network as shown on figure 3.9. This unrolled network shows how we can supply a stream of data to the recurrent neural network.

The problem with basic recurrent neural networks

Vanilla recurrent neural networks are not actually used very often in practice [26]. The main reason is the vanishing gradient problem. For recurrent neural networks, ideally, we would want to have long memories, so the network can connect data relationships at significant distances in time. That sort of network could make real progress in understanding how language and narrative works, how stock market events are correlated and so on. However, the more time steps we have, the more chance we have of back-propagation gradients either accumulating and exploding or vanishing down to nothing. Because the gradient will become basically zero when dealing with many prior time steps, the weights won't adjust to take into account these values, and therefore the network won't learn relationships separated by significant periods of time. This makes vanilla recurrent neural networks not

```
import tensorflow as tf

n_inputs = 61  # no. of features
n_hidden1 = 300
n_hidden2 = 100
n_outputs = 3
```

```
X = tf.placeholder(tf.float32, shape=(None, n_inputs), name="X")
y = tf.placeholder(tf.int32, shape=(None), name="y")
```

```
def neuron_layer(X, n_neurons, name, activation=None):
    with tf.name_scope(name):
        n_inputs = int(X.get_shape()[1])
        stddev = 2 / np.sqrt(n_inputs)
        init = tf.truncated_normal((n_inputs, n_neurons), stddev=stddev)
        W = tf.Variable(init, name="kernel")
        b = tf.Variable(tf.zeros([n_neurons]), name="bias")
        Z = tf.matmul(X, W) + b
        if activation is not None:
            return activation(Z)
        else:
            return Z
```

```
with tf.name_scope("dnn"):
    hidden1 = neuron_layer(X, n_hidden1, name="hidden1",
                           activation=tf.nn.relu)
    hidden2 = neuron_layer(hidden1, n_hidden2, name="hidden2",
                           activation=tf.nn.relu)
    logits = neuron_layer(hidden2, n_outputs, name="outputs")
```

```
with tf.name_scope("loss"):
    xentropy = tf.nn.sparse_softmax_cross_entropy_with_logits(labels=y,
                                                              logits=logits)
    loss = tf.reduce_mean(xentropy, name="loss")
```

```
learning_rate = 0.01

with tf.name_scope("train"):
    optimizer = tf.train.GradientDescentOptimizer(learning_rate)
    training_op = optimizer.minimize(loss)
```

```
with tf.name_scope("eval"):
    correct = tf.nn.in_top_k(logits, y, 1)
    accuracy = tf.reduce_mean(tf.cast(correct, tf.float32))
```

Figure 3.6: Deep Neural Network design used in this paper

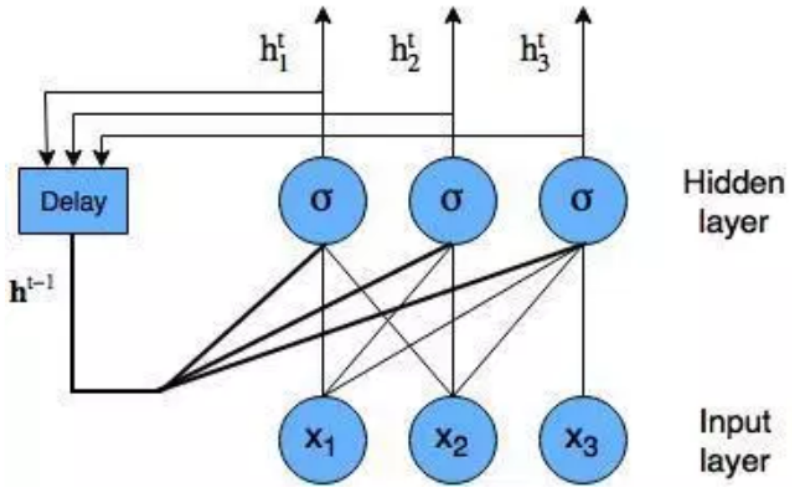


Figure 3.7: Recurrent Neural Network

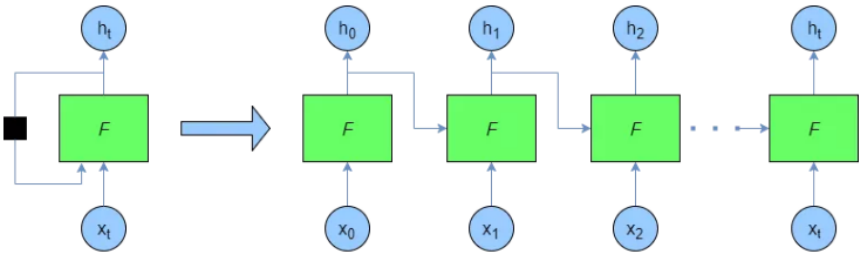


Figure 3.8: Unrolled Recurrent Neural Network

very useful. To overcome this vanishing/exploding gradient problem, we use LSTM cells in an RNN network. LSTM cells will be discussed in the next section.

3.4 Long Short Term Memory (LSTM)

Long Short-Term Memory (LSTM) networks are an extension for recurrent neural networks, which basically extends their memory. Therefore it is well suited to learn from important experiences that have very long time lags in between [5]. LSTM's enable RNN's to remember their inputs over a long period of time. This is because LSTM's contain their information in a memory that is much like the memory of a computer because the LSTM can read, write and delete information from its memory. In an LSTM you have three gates: input, forget and output gate. These gates determine whether or not to let new input in (input gate), delete the information because it isn't important (forget gate) or to let it impact the output at the current time step (output gate). An illustration of a LSTM RNN with its three gates is shown in figure 3.10

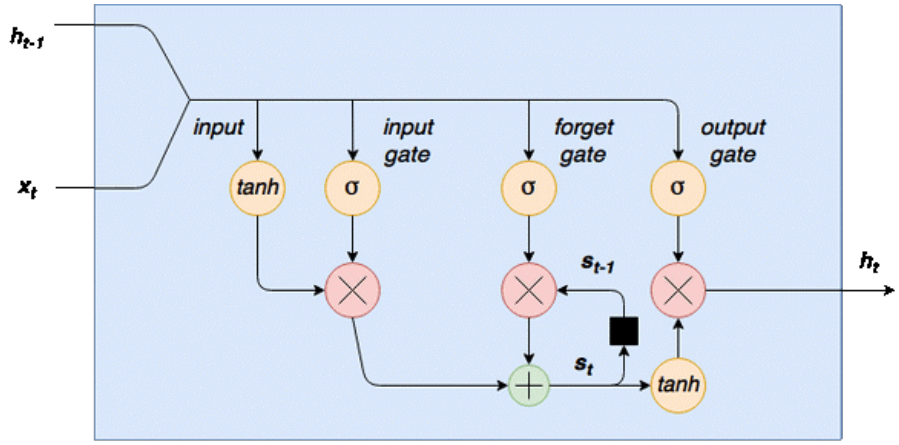


Figure 3.9: An LSTM cell with its inputs, gates and outputs

The input gate:

First, the input is squashed between -1 and 1 using a tanh activation function. This squashed input is then multiplied element-wise by the output of the input gate. The input gate is basically a hidden layer of sigmoid activated nodes, with weighted x(t) and h(t-1) input values, which outputs values of between 0 and 1 and when multiplied element-wise by the input determines which inputs are switched on and off.

The internal state and the forget gate:

Forget gate is again a sigmoid activated set of nodes which is element-wise multiplied by s(t-1) to determine which previous states should be remembered (i.e. forget gate output close to 1) and which should be forgotten (i.e. forget gate output close to 0). This allows the LSTM cell to learn appropriate context.

```
# The LSTM architecture
regressor = Sequential()
# First LSTM layer with Dropout regularisation
regressor.add(LSTM(units=50, return_sequences=True, input_shape=(X_train.shape[1],1)))
regressor.add(Dropout(0.2))
# Second LSTM layer
regressor.add(LSTM(units=50, return_sequences=True))
regressor.add(Dropout(0.2))
# Third LSTM layer
regressor.add(LSTM(units=50, return_sequences=True))
regressor.add(Dropout(0.2))
# Fourth LSTM layer
regressor.add(LSTM(units=50))
regressor.add(Dropout(0.2))
# The output layer
regressor.add(Dense(units=1))

# Compiling the RNN
regressor.compile(optimizer='rmsprop',loss='mean_squared_error')
# Fitting to the training set
regressor.fit(X_train,y_train,epochs=20,batch_size=32)
```

Figure 3.10: LSTM architecture used in this paper

The output gate:

The final stage of the LSTM cell is the output gate. The output gate has two components – another tanh squashing function and an output sigmoid gating function. The output sigmoid gating function, like the other gating functions in the cell, is multiplied by the squashed state's s(t) to determine which values of the state are output from the cell. As you can tell, the LSTM cell is very flexible, with gating functions controlling what is input, what is "remembered" in the internal state variable, and finally what is output from the LSTM cell. Figure 3.11 shows the LSTM architecture used in this paper along with the parameter values

3.5 Random Forest Algorithm

A random forest is a supervised classification algorithm. It creates a forest (many decision trees) and orders their nodes and splits randomly. The more trees in the forest, the better the results it can produce [26]. If you input a training dataset with targets and features into the decision tree, it will formulate some set of rules that can be used to perform predictions. Random forests are made of many decision trees. They are ensembles of decision trees, each decision tree created by using a subset of the attributes used to classify a given population .Those decision trees vote on how to classify a given instance of input data, and the random forest bootstraps those votes to choose the best prediction. This is done to prevent overfitting, a common flaw of decision trees. Figure 3.12 shows the Random Forest algorithm carries out its tasks using multiple decision trees

The out-of-bag (oob) error estimate

In random forests, there is no need for cross-validation or a separate test set to get an unbiased estimate of the test set error. It is estimated internally, during the run, as follows:

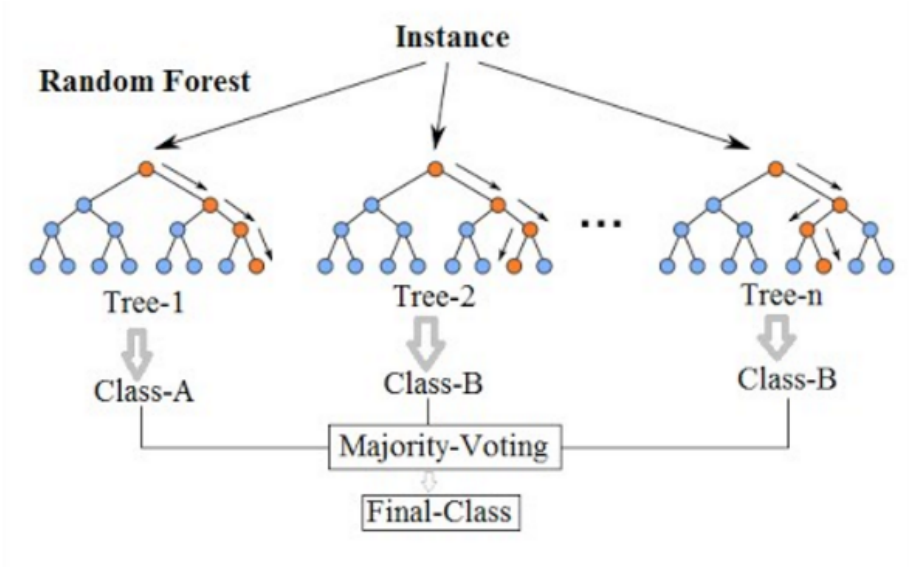


Figure 3.11: Random Forest algorithm in action

- Each tree is constructed using a different bootstrap sample from the original data. About one-third of the cases are left out of the bootstrap sample and not used in the construction of the kth tree.
- Put each case left out in the construction of the kth tree down the kth tree to get a classification. In this way, a test set classification is obtained for each case in about one-third of the trees. At the end of the run, take j to be the class that got most of the votes every time case n was oob. The proportion of times that j is not equal to the true class of n averaged over all cases is the oob error estimate. This has proven to be unbiased in many tests.

Variable importance

In every tree grown in the forest, put down the oob cases and count the number of votes cast for the correct class. Now randomly permute the values of variable m in the oob cases and put these cases down the tree [23]. Subtract the number of votes for the correct class in the variable-m-permuted oob data from the number of votes for the correct class in the untouched oob data. The average of this number over all trees in the forest is the raw importance score for variable m. If the values of this score from tree to tree are independent, then the standard error can be computed by a standard computation. The correlations of these scores between trees have been computed for a number of data sets and proved to be quite low, therefore we compute standard errors in the classical way, divide the raw score by its standard error to get a z-score, ands assign a significance level to the z-score assuming normality. If the number of variables is very large, forests can be run once with all the variables, then run again using only the most important variables from the first run. For each case, consider all the trees for which it is oob. Subtract the percentage of votes for the correct class in the variable-m-permuted oob data from the percentage of votes for the correct class in the untouched oob data. This is the local importance score for variable m for this case, and is used in the graphics program

3.6 Logistic Regression

Logistic regression is the type regression analysis, where the dependent variable is dichotomous or binary. In logistic regression, the probability that a binary target is True is modeled as a logistic function of a linear combination of features [14]. The following figure illustrates how logistic regression is used to train a 1-dimensional classifier. The training data consists of positive examples (depicted in blue) and negative examples (in orange). The decision boundary (depicted in pink) separates out the data into two classes.

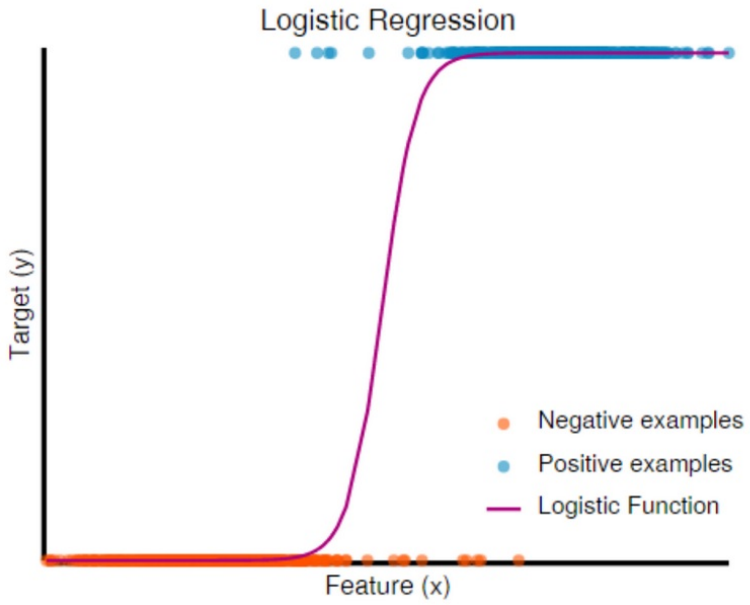


Figure 3.12: Logistic Regression

The main goal of logistic regression is to find the best fitted model to explain the relationship between the dependent variable and a set of independent variable [14]. This regression model is also known as binary logistic model as the outcome is 1 or 0. It allows one to say that the presence of a risk factor increases the odds of a given outcome by a specific factor. To illustrate logistic regression let us look at a diagram in the given diagram, it is illustrated how to train a 1-dimensional classifier. Here, the training data is divided into two segments, positive (in orange circles) and negative (in blue triangle), where a positive score means a probability of 1 and negative means probability of 0. The Pink line represents the decision boundary of the logistic regression which separates the data into two classes. The logistic regression can be understood simply as finding the parameters that best fit:

$$y = 1, if \beta_0 + \beta_1 x + \epsilon; y = 0, otherwise \tag{3.4}$$

Where, is an error distributed by the standard logistic regression. Logistic regression is named at the core of the method logistic function. Logistic function is S-shaped curve that can take any real value number and then, map it to a value that is between 0 and 1. But never exactly the limit values. The logistic function (t) is defined as follows:

$$\sigma(t) = \frac{1}{1+e^{-t}} = \frac{e^t}{1+e^t} \tag{3.5}$$

In Linear Regression, the output is the weighted sum of inputs [14]. Logistic Regression is a generalized Linear Regression in the sense that we don't output the weighted sum of inputs directly, but we pass it through a function that can map any real value between 0 and 1. We can see from the below figure that the output of the linear regression is passed through an activation function that can map any real value between 0 and 1.

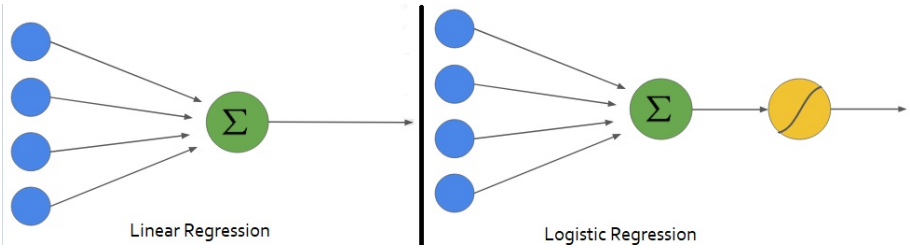


Figure 3.13: Linear and Logistic Regression

The activation function that is used is known as the sigmoid function. The plot of the sigmoid function looks like

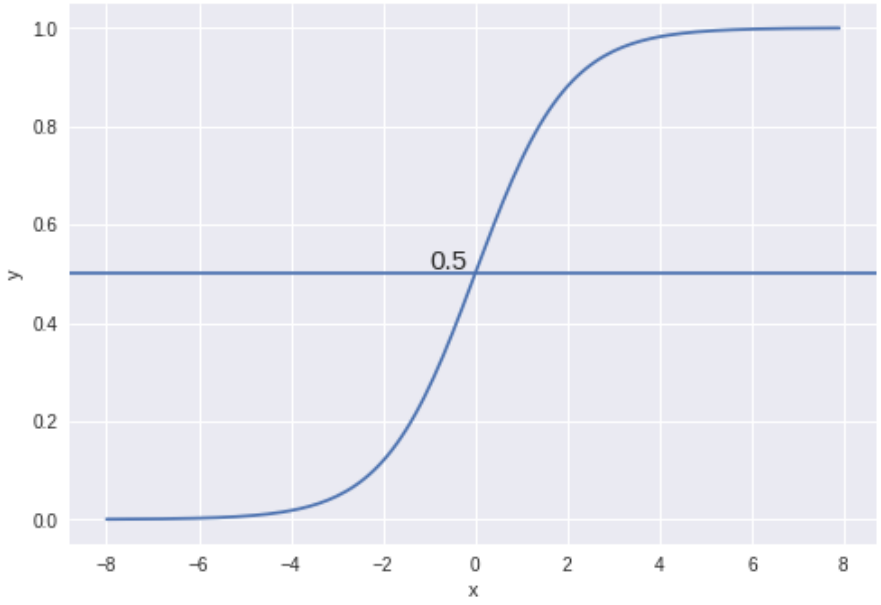


Figure 3.14: Sigmoid Function

We can see that the value of the sigmoid function always lies between 0 and 1. The value is exactly 0.5 at X=0. We can use 0.5 as the probability threshold to determine the classes. If the probability is greater than 0.5, we classify it as Class-1(Y=1) or else as Class-0(Y=0).

3.6.1 Confusion Matrix

Confusion matrix is different than the metrics that have been discussed so far. Confusion matrix returns a table layout that helps to visualize the performance of

an algorithm rather than producing a numerical value that indicates the goodness of the algorithm.
Confusion matrix is an X Y matrix where X dimension indicates the true classes and Y dimension indicates the predicted classes. The value of X and Y is equal, and they indicate the number of classes in a dataset.

3.7 Support Vector Machine (SVM)

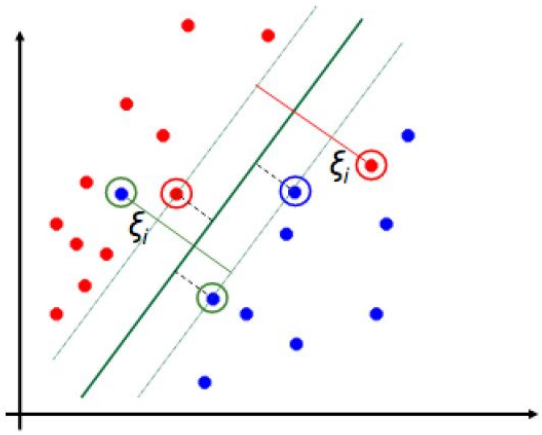


Figure 3.15: SVM with some missclassification

Support Vector Machine (SVM) follows a supervised machine learning algorithmic rule is be used for each classification or regression challenges [8]. However, it's principally utilized in classification issues. During this algorithmic rule, we have a tendency to plot every information item as some extent in n-dimensional area (where n is range of options we have) with the worth of every feature being the value of a specific coordinate. Then, we classifies by finding the hyper-plane that differentiate the 2 categories all right.
Hard SVM finds the maximum margin solution by solving:

$$min\frac{|\theta|^2}{2}, \text{such that } y^i(\theta^T\phi(x^i) + \theta_0 \geq 1 \tag{3.6}$$

with classifier

$$H(x) = \theta^T\phi(x) + \theta_0 \tag{3.7}$$

Where the term "hard" indicates that the formulation doesn't enable incorrect assignments. Often, the choice boundary could generalize higher if we tend to enable some area for error in assignments, accommodating for outliers within the dataset or non-separable dataset. Figure 3.4 shows an easy example wherever a soft linear SVM could manufacture a much better work than a kernelized hard SVM.
Also, we may wish to add regularization so that SVM doesn't over-fit the dataset by scaling. Hence, regularized soft SVM adds additional terms, C and i to allow for points inside the margin and to better generalize. Soft SVM hence solves the optimization problem:

$$min\frac{|\theta|^2}{2} + \sum_{N}^{i=1} \epsilon_i, \text{such that } y^i(\theta^T \phi(x^i) + \theta_0) \geq 1 - \epsilon_i \text{where } \epsilon_i \geq 0, \text{and } C > 0 \quad (3.8)$$

Thus, C controls the trade-off between the slack variable penalty and the margin [8]. It is analogous to the inverse of the regularization coefficient because it controls the trade-off between minimizing training errors and controlling model complexity. In this paper, we mainly using Gaussian/RBF kernel:

$$K(x, y) = exp(-\frac{|x - y|^2}{2\sigma^2}) \quad (3.9)$$

As it is a universal kernel and offers model flexibility via the free parameter . As determines the breadth of the Gaussian kernel, it defines however so much the influence of one training example reaches. Too little a worth implies that the radius of the area of influence of the support vectors solely includes the support vector itself. If is just too massive, the model is just too forced and can't capture the complex-ness or "shape" of the data.
To gain an improved intuition of applying kernel functions to SVM, we tend to specific the soft SVM optimization downside in dual form. And the classifiers becomes

$$H(x) = \sum_{N}^{i} \alpha_i y_j K(x_i, x) + \theta_0 \quad (3.10)$$

Hence, in the SVM model performed in chapter 6, we determine , the weights of each feature set, C , the regularization coefficient, and , the width of the Gaussian kernel.

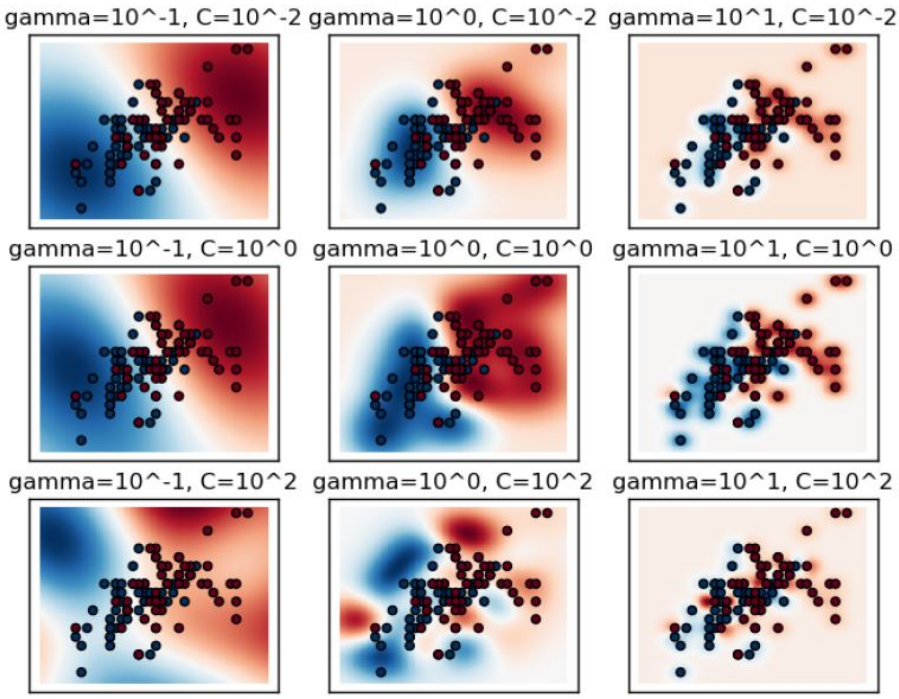


Figure 3.16: Gaussian kernel variation of SVM

Chapter 4

Stock Market Analysis

Market analysis can be three types. They are fundamental analysis, technical analysis and sentimental analysis. In this chapter we generate trading system reviewing technical analysis techniques. We explain general concepts of stock market indicators .

4.1 Technical Analysis

Technical analysis is a mercantilism discipline utilized to judge investments and establish trading opportunities by analyzing applied mathematics trends gathered from trading activity. It's the prediction of future monetary value movements supported an examination of past value movements. Technical analysis is used on any security with historical trading information [2]. This includes stocks, futures, commodities, invariable, currencies, and different securities.

Technical analysts believe that prices move in short, medium, and long trend. In different words, a stock value is a lot of doubtless to continue a past trend than move unpredictably. Most technical trading ways square measure are this assumption.

Traders and investors typically use a pair of classes of tools to create a choice what stocks to buy for and sell; basic and technical analysis, every of that aim at analyzing and predicting shifts in provide and demand If there unit of measurement extra sellers than customers for a stock the price got to fall, and equally, if there unit of measurement extra customers than sellers the price got to rise. Given the flexibleness to foresee these shifts in provide and demand thus offers the bargainer the flexibleness to work out profitable entry and exit positions, that's that the ultimate goal of stock analysis.

Whereas basic analysis involves the study of company fundamentals like revenues and expenses, market position, annual growth rates, and so on, technical analysis is entirely committed price and volume data, notably worth patterns and volume spikes. value and volume data is instantly getable in real time that produces technical analysis ideally suited to short swing trades. The underlying assumption in technical analysis is that stock prices evolve with a definite regularity, forming reliable and sure value and volume patterns.

4.2 Charting Techniques

A price chart shows how a stock's price has evolved over a given period of time, typically presented as in Figure 4.1 which plots the closing price of the company ACI over the beginning of 2010.

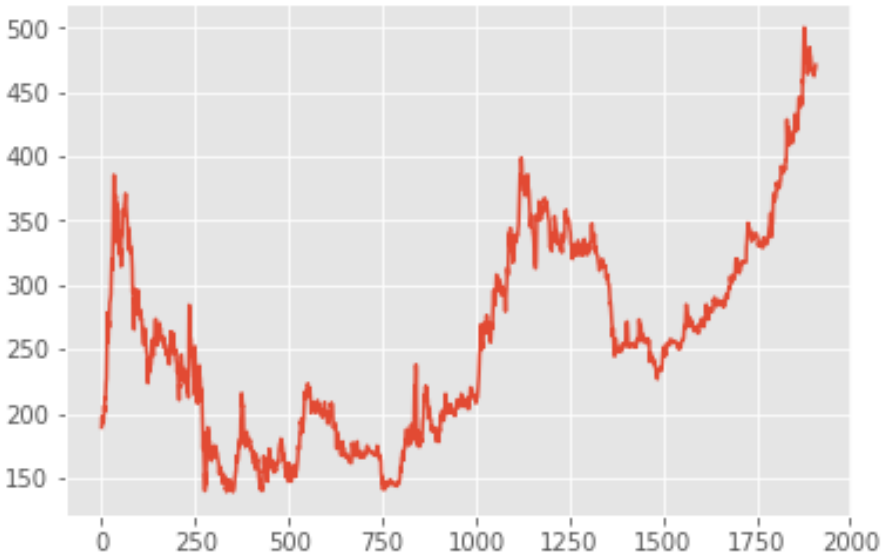


Figure 4.1: Closing Price Plot

Candlestick charts are designed to give the trader a quicker and more complete picture of price movements in a given time frame [10]. The candlestick entity (named so for its resemblance to a candle with a wick in both ends), as shown in Figure 4.2, is drawn with a rectangular "real body" that represents the range between the opening and closing price, and lower and upper "shadows" that represent the lowest

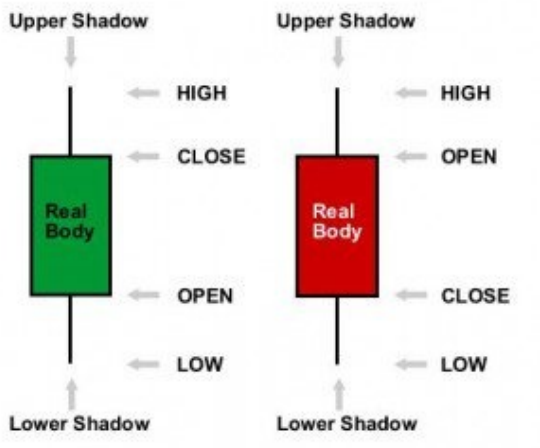


Figure 4.2: Candlestick Entity

and highest price obtained for the stock in the period. The real body is colored white or green if the closing price is above the opening price, and black or red if the closing price is below the opening price. Thus, green candlesticks represent positive price movements and red candlesticks represent negative price movements. In this

way, the candlestick charts provides a more visual and immediate impression of price movements. Candlestick charts can be used on any time scale, from intraday 15-minute

Figure 4.3: Candlestick Plot, GP

charts to daily or yearly charts. According to preference and scope of analysis, traders use dierent time scales when analyzing a stock. Figure 4.3 shows a daily candlestick chart over the same period as in Figure 4.1 shows the closing price of ACI.

4.3 Classical Chart Analysis

Charting the price of financial instruments creates a picture of the market "battle" between buyers (bulls) and sellers (bears). Market prices don't move in straight line but zigzag as price rise and falls depending on who is winning the battle. In order to decide if they should buy or sell, market players need tools to determine their trading strategy. One such tool is the market trend and is simply defined as the way the market is moving [10].

4.3.1 Uptrend

This is considered to be the time to buy or go long. A major uptrend known as a Bull market. Market players have the opportunity to profit by being bullish-buying and staying with the uptrend.

4.3.2 Downtrend

This is considered to be the time to sell or go short. A major downtrend is also known as a Bear market. Market players have the opportunity to profit by being bearish-selling with a view to buying later. In equity markets it is not always possible for normal investors to be short. They are either in or out of the market.

4.3.3 Sideways

This situation arises when there is no strong conviction by either bulls or bears. As a result market prices rise and fall in a more congested space. This type of pattern

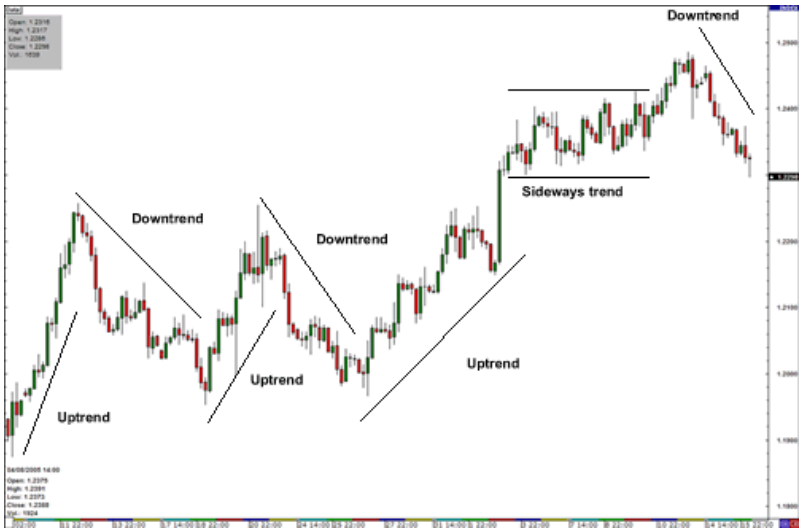


Figure 4.4: Uptrend, Downtrend , Sideways

is generally considered to be a signal to stay out of the market. However, some traders use congestion patterns as an opportunity for range trading.

4.3.4 Resistance Level

This is the opposite of support and is the level that resist market price action for a period of time. It is the level where selling interest is strong enough to overcome buying pressure so that market does not exceed that level. An idea essential to

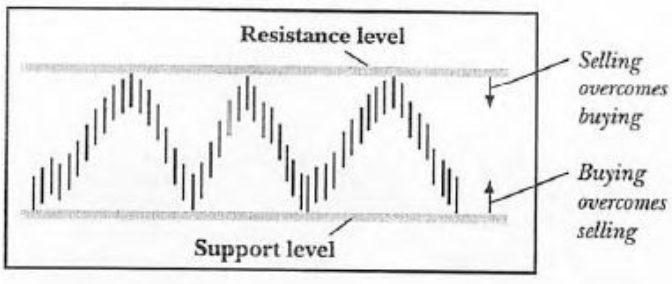


Figure 4.5: Support and Resistance

technical analysis is that the support and resistance levels are where the price movements stopped before. Traders expect that this is where prices will stop again and if they do not, then the trends have failed.

4.4 Technical Indicators

Indicators represent a numerical approach to technical analysis as critical a subjective approach. By gazing cash flow, trends, volatility, and momentum, they supply a secondary live to actual value movements and facilitate traders ensure the standard of chart patterns or kind their own purchase or sell signals.

4.5 Volume

4.5.1 Simple Moving Average (SMA)

We started off with a awfully straightforward approach, that calculates the moving average of stock price for last a hundred and three hundred days [2]. Moving average could be a wide used indicator that helps filtering price fluctuations. During this case, if we tend to compare the moving average of a hundred days and 300days, we will see if the market is increasing or shrinking. The strategy is pretty straightforward if moving average of a corporation for last a hundred days is higher, we tend to purchase stocks of that company with 100 percent of our total out there cash. Moving averages "smooth" price information by making one flowing line. The line represents the typical price over an amount of time. That moving average the dealer decides to use is decided by the timeframe within which he or she trades. For investors and semi-permanent trend followers, the 200-day, 100-day and 50-day straightforward moving average square measure standard selections.

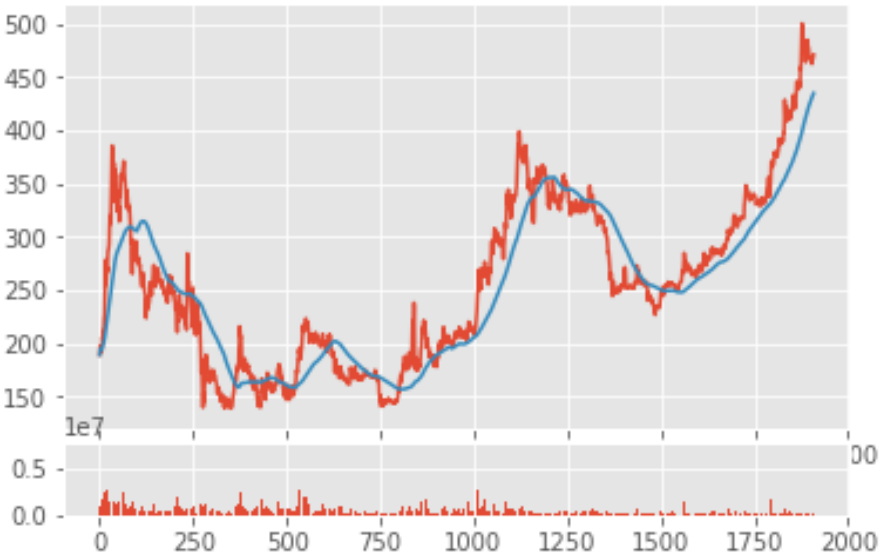


Figure 4.6: Simple moving average with closing price of ACI

The formula for SMA is

$$S = \frac{A_1 + A_2 + A_3 + ... + A_n}{n} \tag{4.1}$$

We can use simple moving averages in several ways. If it is mostly moving sideways for few days, then the price isn't trending, it is ranging. For an uptrend, the moving average line is angled up. Moving averages is not used for prediction; it simply shows what the price is doing, on average, over a period of time.

4.5.2 On Balance Volume (OBV)

On balance volume may be a straightforward indicator that uses volume and price to live shopping for pressure and commercialism pressure [10]. Buying pressure is clear once positive volume exceeds negative volume and also the OBV line rises.

Merchandising pressure is gift once negative volume exceeds positive volume and also the OBV line falls. Chartists will use OBV to substantiate the underlying trend or search for divergences which will forecast a value amendment. Like all indicators, it's necessary to use OBV in conjunction with alternative aspects of technical analysis. It's not a standalone indicator. OBV is combined with basic pattern analysis or to substantiate signals from momentum oscillators.
For example If today's closing price is higher than yesterday's closing price, then: Current OBV = Previous OBV + today's volume.

Figure 4.7: On Balance Volume In action

From figure 4.7 we see that when the value of on balance volume is greater than 0.0 then we will sell the stock. However if the value is less than 0.0 we will buy that otherwise we will hold.

4.5.3 Chaikin Money Flow (CMF)

The Chaikin cash Flow (CMF) is associate indicator created by brandy Chaikin within the Eighties to observe the buildup and distribution of a stock over a mere amount. The default CMF amount is twenty one days. The indicator readings vary between +1 and -1. Any crosses on top of or below zero are often accustomed establish changes in cash flow, further as shopping for or selling momentum. Shopping for pressure ends up in positive readings, above zero. Sustained selling of a stock pushes the indicator below zero. Once CMF oscillates right round the zero line, this means comparatively equal shopping for and selling pressure, and no clear trend [2]. Traders use the CMF system as a tool for distinctive and assessing the trends of stocks being listed.
Formula for CMF is

$$MoneyFlowMultiplier = \frac{((CloseValue\ -\ LowValue)-(HighValue\ -\ CloseValue))}{(HighValue\ -\ LowValue)} \tag{4.2}$$

Money Flow Volume = Money Flow Multiplier x Volume for the Period

CMF = 21-day Average of the Daily Money Flow / 21-day Average of the Volume From this above figure CMF will give a buy signal when volume of CMF will be

Figure 4.8: Chaikin Money Flow

greater than 0.5 or less than 0.3. For sell signal volume will be less than 0. Otherwise we will hold.

4.6 Volatility

4.6.1 Bollinger Bands (BB)

Bollinger Bands area unit a useful indicator, however they need variety of limitations [10]. Bollinger Bands area unit derived from an easy moving average that is that the average value over a precise variety of value bars. This suggests Bollinger Bands can forever react to price moves, however won't forecast them. Bollinger Bands area unit reactive, not prognostic ative. There are total three bands in Bollinger Band. Middle band, higher Band a Lower band. Formula for calculating these are

Middle Band = 20-day simple moving average (SMA)
Upper Band = 20-day SMA + (20-day standard deviation of price x 2)
Lower Band = 20-day SMA – (20-day standard deviation of price x 2)

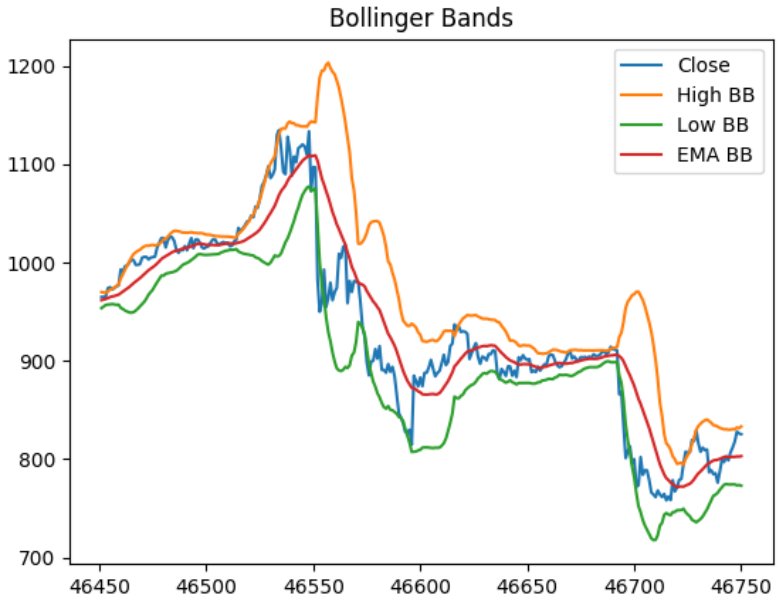


Figure 4.9: Bollinger Bands

Candlestick of the price will reach the upper band on a regular basis if an uptrend is strong. If a downtrend is strong it will reach the lower band on a regular basis. Reaching the lower band shows selling activity remains strong.

4.7 Trend

4.7.1 Moving Average Convergence Divergence (MACD)

Moving Average Convergence/Divergence generator (MACD) is one in all the best and only momentum indicators on the market [2]. The MACD turns 2 trend-following indicators, moving averages, into a momentum generator by subtracting the longer moving average from the shorter one. As a result, the MACD offers the

most effective of each worlds: trend following and momentum. The MACD fluctuates on top of and below the zero line because the moving averages converge, cross and diverge. Traders will explore for signal line crossovers, center line crossovers and divergences to get signals. As a result of the MACD is infinite, it's not significantly helpful for characteristic overbought and oversold levels.

Figure 4.10: MACD in Action

The formula for MACD is
MACD = 12 Period EMA – 26 Period EMA
Here EMA means exponential moving average. From figure 4.10 we can see that there is price plot of NHFIL Co. limited. We plotted the candlestick of that company. There are 2 lines. The red line is called MACD line and blue line is called Signal line. When the red line crosses blue line a buyer should buy the share. However, when blue line crosses the MACD line a buyer should sell the share.

4.7.2 Average Directional Movement Index (ADX)

Directional Movement (DM) is defined as the largest part of the current period's price range that lies outside the previous period's price range. For each period we have to calculate:

+DM = positive or plus DM = High - Previous High
-DM = negative or minus DM = Previous Low - Low

The smaller of the two values is reset to zero, i.e., if +DM ¿ -DM, then -DM = 0. On an inside bar (a lower high and higher low), both +DM and -DM are negative values, so both get reset to zero as there was no directional movement for that period. The True Range (TR) is calculated for each period, where:
TR = Max of (High - Low), (High -PreviousClose), (PreviousClose - Low)
The +DM, -DM and TR are each accumulated and smoothed using a custom smoothing method proposed by Wilder. For an n period smoothing, 1/n of each period's value is added to the total each period, similar to an exponential smoothing:

$$
\begin{aligned}
+DM_t &= (+DM_{t-1}/n)) + (+DM_t) \\
+DM_t &= (+DM_{t-1}/n)) + (+DM_t) \\
+DM_t &= (+DM_{t-1}/n)) + (+DM_t)
\end{aligned}
\tag{4.3}
$$

Figure 4.11: ADX in action

Now we have to compute the positive/negative Directional Indexes, +DI and -DI, as a percentage of the True Range:

$$
\begin{aligned}
+D_I &= (+D_M/T_R) * 100 \\
-D_I &= (-D_M/T_R) * 100
\end{aligned}
\tag{4.4}
$$

Then derive the directional difference as the absolute value of the differences: DIdiff = — ((+DI) - (-DI)) — Sum of the directional indicator values: DIsum = ((+DI) + (-DI)) . By calculating the Directional Movement index: DX = (DIdiff / DIsum) * 100 . The DX is always between 0 and 100. Finally, appling Wilder's smoothing technique to produce the final ADX value:

$$
ADX_t = \frac{(ADX_{t-1} \times (n-1)) + DX_t}{n} \tag{4.5}
$$

From the above figure, when the difference between (+)Average directional moving index and (-)Average direction moving index is greater than 3, we will buy the share. We will do the reverse technique of selling a share. We can also say that when green line is crossing the red line, buyer can go for a buy and when green line is crossing red line buyer can go for selling.

4.7.3 Mass Index (MI)

The Mass Index studies the distinction between high and low costs during a security over time [10]. It's thought of a study on volatility. Once the Mass Index moves on top of a definite level, then drops back below, it's normally understood that a trend modification may be forthcoming.
Formula for Mass Index is

$$\sum_{i}^{25} \frac{9-\text{day EMA of (High - Low)}}{9-\text{day EMA of a 9 - day EMA of (High - Low)}} \tag{4.6}$$

The Mass Index uses the high-low differential to produce a smoothened volatility live. The indicator usually fluctuates within the mid-20s; readings close to the high finish of the historical vary recommend increasing volatility that will increase the probabilities for a trend reversal. Though Dorsey set the bulge threshold at twenty seven, chartists ought to take into account a lower threshold to provide a lot of signals. Confine mind that the Mass Index doesn't have a directional bias. The directional bias depends on the present trend. Like all indicators, we've to use alternative analysis techniques to enrich the Mass Index.

4.8 Momentum

4.8.1 Money Flow Index (MFI)

Money Flow Index (MFI) is associate generator that uses each price and volume to live buying and marketing pressure [2]. MFI is additionally referred to as volume

Figure 4.12: Money Flow Index in MeghnaCem

weighted RSI. MFI starts with the everyday price for every amount. Cash flow is positive once the everyday price rises (buying pressure) and negative once the everyday price declines (selling pressure). A quantitative relation of positive associated

negative cash flow is then obstructed into associate RSI formula to form a generator that moves between zero and 100. As a momentum oscillator tied to volume, MFI is best suited to spot reversals and price extremes with a range of signals.
Formula for MFI is

$$\text{Typical Price} = (\text{High} + \text{Low} + \text{Close})/3$$
$$\text{Raw Money Flow} = \text{Typical Price x Volume}$$
$$\text{Money Flow Ratio} = (\text{14-period Positive Money Flow})/ (\text{14-period Negative Money Flow})$$
$$\text{MFI} = 100 - 100 / (1 + \text{Money Flow Ratio})$$

The blue line of the figure 4.12 is called MFI line. When the line is crossing 40, a buyer should wait to buy the share. If it crosses value 60 then it is the good time to buy the share. When the line hits 80 and come back to 60, it is perfect time to sell the share. However we should take help from other indicators for confirmation.

4.8.2 Relative Strength Index (RSI)

The relative strength index (RSI) usually a momentum indicator that measures the magnitude of recent price changes to judge overbought or oversold conditions within the price of a stock or different plus [2]. The RSI is displayed as associate degree generator (a line graph that moves between 2 extremes) and might have a reading from zero to one hundred.

$$RSI = 100 - \frac{100}{(1 + RS)} \tag{4.7}$$

Figure 4.13: RSI for IFIC

The very first calculations for average gain and average loss are simple 14-period averages:

$$\text{First Gain} = \text{Sum of Gains over the past 14 periods} / 14$$
$$\text{First Loss} = \text{Sum of Losses over the past 14 periods} / 14$$

The second, and subsequent, calculations are based on the prior averages and the current gain loss:

$$\text{Gain} = [(\text{previous Gain}) \text{ x } 13 + \text{current Gain}] \ / \ 14.$$
$$\text{Loss} = [(\text{previous Loss}) \text{ x } 13 + \text{current Loss}] \ / \ 14$$

Figure 4.13 says that if the signal is uptrend and it crosses values of 50 after crossing 30 then it is a buy signal. If the price is downtrend and crosses 50 from coming upper then it is a sell signal.

4.8.3 Stochastic Oscillator (SR)

Stochastic generator is called a momentum indicator that shows the placement of the shut relative to the high-low vary over a group range of periods. In keeping with associate interview with Lane, the random generator does not follow price, it does not follow volume or something like that. It follows the speed or the momentum of the price. As a rule the momentum changes direction before the price.

$$K = \frac{C - L14}{(H14 - L14)} * 100 \tag{4.8}$$

Where: C = the most recent closing price;
L14 = the lowest price traded of the 14 previous trading sessions;
H14 = the highest price traded during the same 14-day period; and
K = the current value of the stochastic indicator.
K is referred to sometimes as the slow stochastic indicator. The "fast" stochastic indicator is taken as D = 3-period moving average of K.

Figure 4.14: Stochastic Oscillator for HRTex

While momentum oscillators are best suited to commerce ranges, they will even be used with securities that trend, provided the trend takes on a zigzag format. Pull-backs are a part of uptrends that zigzag higher. Bounces are a part of downtrends

that zigzag lower. During this regard, the random generator will be wont to establish opportunities harmonic with the larger trend. Like all technical indicators, it's vital to use the random generator in conjunction with different technical analysis tools. From figure 4.14 we can see that this is a candlestick representation of HR textile. When the value of Stochastic Oscillator is above 20 that is a good time to buy a share. But buyer should not take buying decision only seeing this indicator. If the Stochastic line is crossing 50 or 40 from upper a buyer should sell the share otherwise hold [10].

4.8.4 True strength index (TSI)

The True Strength Index (TSI) is a distinctive indicator supported double ironed worth changes [2]. Price modification represents momentum in its truest kind. The double smoothing with 2 exponential moving averages reduces the noise associate degreed produces generator that tracks price quite well. Additionally to the standard generator signals, chartists will typically draw trend lines, support lines and resistance lines directly on TSI. These will then be wont to generate signals supported breakouts and breakdowns. Calculation for TSI is

Double Smoothed PC = Current Price minus Prior Price
First Smoothing = 25-period EMA of PC
Second Smoothing = 13-period EMA of 25-period EMA of PC
Double Smoothed Absolute PC
Absolute Price Change —PC— = Absolute Value of Current Price minus Prior Price
First Smoothing = 25-period EMA of —PC—
Second Smoothing = 13-period EMA of 25-period EMA of —PC—

TSI = 100 x (Double Smoothed PC / Double Smoothed Absolute PC)

The TSI should only be used in conjunction with other forms of analysis, such as price action analysis and other technical indicators.

Chapter 5

Data Processing

This chapter explains how we prepared and generated the technical indicators and stock datasets to be used for machine learning

5.1 Data Preparation

First, we read the GP stock's csv data using the Pandas library and we have the following dataframe. It contains the columns as shown in Figure 5.1

	Company	Date	Open	High	Low	Close	Volume
0	GP	3/1/2010	187.0	192.3	187.0	189.3	962200
1	GP	4/1/2010	190.0	198.9	190.0	197.9	2749600
2	GP	5/1/2010	199.0	202.0	195.2	196.5	2061800
3	GP	6/1/2010	196.0	197.8	195.0	196.4	1436200
4	GP	7/1/2010	197.0	197.0	193.5	193.8	1259600
5	GP	10/1/2010	193.9	194.4	190.2	192.3	1605400
6	GP	11/1/2010	193.9	197.4	192.8	196.8	1784800
7	GP	12/1/2010	197.4	207.0	197.4	206.0	5308600
8	GP	13-01-2010	206.5	209.0	197.6	201.2	2952200
9	GP	14-01-2010	203.0	213.9	203.0	211.7	3507000
10	GP	17-01-2010	214.0	215.0	210.2	211.0	3133200

Figure 5.1: GP stock data as we have collected from DSE

To get a closer look at the different columns in detail we have derived the information shown in Figure 5.2.

In figure 5.2, we can separately see the mean, standard deviation, maximum, minimum as well as the quartile values for each column which greatly facilitates us in getting a holistic view of the entire dataset. Afterwards, we add a new column called "Decision" which will contain the proposed decision about whether or not it is prof-

```
dfGP.describe()
```

	Open	High	Low	Close	Volume
count	1909.000000	1909.000000	1909.00000	1909.000000	1.909000e+03
mean	256.262074	259.302043	252.96988	255.841697	5.948112e+05
std	78.282818	78.844228	77.97213	78.385628	6.706939e+05
min	139.700000	140.900000	104.80000	139.100000	2.134000e+04
25%	186.900000	189.100000	182.00000	185.800000	1.985540e+05
50%	253.000000	254.900000	250.50000	252.600000	3.798000e+05
75%	319.000000	323.000000	316.30000	319.300000	7.242000e+05
max	500.000000	506.900000	488.00000	499.900000	7.492600e+06

Figure 5.2: GP stock data description of respective columns

```
dfGP['Decision']=0
for i in range(dfGP.shape[0]-1):
    if(i==0):
        pass
    else:
        if(dfGP['Close'][i+1] > dfGP['Close'][i]):
            dfGP['Decision'][i+1] = 1
        elif(dfGP['Close'][i+1] < dfGP['Close'][i]):
            dfGP['Decision'][i+1] = -1
        else:
            dfGP['Decision'][i+1] = 0
```

Figure 5.3: Decision label in GP stock data

itable for the user to “Buy” or “Sell” or simply “Hold” on a stock.The “Decision” column is calculated using the logic shown on figure 5.3.

Now that we have created our classification labels, we will apply the technical indicators on the “Close” price of our stock data as shown on figure 5.4 The “num feats” variable represents the number of features that we are to use for training our model and since we will be using a supervised model, so we also need to specify our classification labels during training which is provided by the decision column mention earlier. With our input and output features declared, now we need to transform the data as it is required for the training of the model. To reduce overfitting the model, we will split the entire dataset into 2 parts. One part will be used for training and the other for validating the model’s predictions.

The code on Figure 5.5 shows how we have implemented the train/test split. As the code above shows, 33% of our total dataset will be used for testing purpose and the rest of the dataset will be used for training the model.

```
num_feats = ['Open', 'High', 'Low', 'Close', 'Volume',
       'volume_adi', 'volume_obv', 'volume_obvm', 'volume_cmf', 'volume_fi',
       'volume_em', 'volume_vpt', 'volume_nvi', 'volatility_atr',
       'volatility_bbh', 'volatility_bbl', 'volatility_bbm', 'volatility_bbhi',
       'volatility_bbli', 'volatility_kcc', 'volatility_kch', 'volatility_kcl',
       'volatility_kchi', 'volatility_kcli', 'volatility_dch',
       'volatility_dcl', 'volatility_dchi', 'volatility_dcli', 'trend_macd',
       'trend_macd_signal', 'trend_macd_diff', 'trend_ema_fast',
       'trend_ema_slow', 'trend_adx', 'trend_adx_pos', 'trend_adx_neg',
       'trend_vortex_ind_pos', 'trend_vortex_ind_neg', 'trend_vortex_diff',
       'trend_trix', 'trend_mass_index', 'trend_cci', 'trend_dpo', 'trend_kst',
       'trend_kst_sig', 'trend_kst_diff', 'trend_ichimoku_a',
       'trend_ichimoku_b', 'trend_visual_ichimoku_a',
       'trend_visual_ichimoku_b', 'trend_aroon_up', 'trend_aroon_down',
       'trend_aroon_ind', 'momentum_rsi', 'momentum_mfi', 'momentum_tsi',
       'momentum_uo', 'momentum_stoch', 'momentum_stoch_signal', 'momentum_wr',
       'momentum_ao']
X = dfGP[num_feats]
y = dfGP['Decision']
X.head()
```

Figure 5.4: All respective columns of GP stock data

```
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.33)
```

```
X_train.shape
```

```
(1279, 61)
```

```
X_val.shape
```

```
(630, 61)
```

```
y_train.shape
```

```
(1279,)
```

```
y_val.shape
```

```
(630,)
```

Figure 5.5: GP stock feature variables split into train and test sets

```
from sklearn.preprocessing import StandardScaler

sc_X = StandardScaler()
```

```
X_train = sc_X.fit_transform(X_train)
X_valid = sc_X.transform(X_valid)
```

```
X_train
```

```
array([[ 0.99878375,  1.0165124 ,  1.02018267, ...,  0.52362405,
         0.9877135 , -0.06943706],
       [-0.23521721, -0.05816285, -0.19575847, ..., -0.73344271,
         0.62431166, -0.90982513],
       [-0.97510839, -0.98491222, -1.01491881, ...,  0.86239898,
         0.60256833, -0.87012576],
       ...,
       [-1.40809118, -1.41098935, -1.39250053, ..., -0.05943454,
         0.4631401 , -0.89993936],
       [-1.02732102, -1.058243  , -1.00851912, ...,  1.53127624,
         1.47503586,  0.12528915],
       [-1.10245627, -1.12019487, -1.08147559, ..., -0.94293238,
        -0.83065826,  0.19337592]])
```

Figure 5.6: GP stock data scaled for feeding to ML algorithms

Now, as a final step to our data preparation, we will need to scale the data as shown on Figure 5.6 This is because, if we look closely at the feature columns being used in the model, we can notice that there is a huge difference between the ranges of the columns. For example, the "Volume" column has values in 7 powers of ten, whereas the other columns have values in 3 powers of ten. This creates a problem since the model cannot treat them as separate column values. Now, we can see from the above results that all the values have been scaled using the "Standard Scaler" and they all fall within a range.

	Comapny	Date	Open	High	Low	Close	Volume
0	SQUARETEXT	3/1/2010	112.0	116.7	111.3	116.1	678050
1	SQUARETEXT	4/1/2010	115.4	117.5	114.5	116.4	503050
2	SQUARETEXT	5/1/2010	116.0	118.1	115.7	116.2	435400
3	SQUARETEXT	6/1/2010	116.0	122.4	116.0	120.7	880150
4	SQUARETEXT	7/1/2010	118.7	121.5	118.7	120.5	505550
5	SQUARETEXT	10/1/2010	121.0	121.1	118.5	119.8	410900
6	SQUARETEXT	11/1/2010	120.0	120.9	118.1	118.6	281450
7	SQUARETEXT	12/1/2010	119.8	119.9	117.9	118.2	277850
8	SQUARETEXT	13-01-2010	118.1	119.9	113.0	114.7	470800
9	SQUARETEXT	14-01-2010	117.9	119.8	115.5	118.2	201650
10	SQUARETEXT	17-01-2010	119.9	122.5	119.9	121.8	818550

Figure 5.7: SQUARE textile stock data description

Finally we also perform an additional test of our model using the stock data of another company. We have used "Square Textiles" data which looks as shown on

Figre 5.7 The same data cleaning steps that are carried out on GP data are also carried out on SQUARETEXT data but we do not train our model using its feature values, we just test to see the predicted decisions

5.2 Feature Generation

We will be using domain information from technical analysis in order to generate technical indicators that will be used as input variables for machine learning algorithms. Technical analysis field includes a very wide range of different techniques, and we will target the common ways and not the advanced or computationally expensive ones. We attempt in clustering and making a case for the variety of forms of indicators in Chapter 4. The following types of technical indicators are fed to the model input features:

- **Price Indicators:** Price indicators constitute unit supported moving averages of the daily closing price. Both simple moving averages and exponential moving averages will be used in our paper. Signals are generated the moment that the value moves outside of two customary deviations from the simple moving averages. The window sizes have been varied for being used for moving averages.

- **Momentum Indicators:** These indicators are supported, clearly visible trends and mean reversion patterns. Through the examination in the variations in prices and rate of change of value across completely different window sizes, trends are found.

```
In [16]: df['RSI_Label'] = 0

         df.loc[df['momentum_rsi'] > 63.0,'RSI_Label'] = '1'
         df.loc[df['momentum_rsi'] < 40.0,'RSI_Label'] = '-1'


         df['RSI_Label'] = df['RSI_Label'].astype(int)
```

Figure 5.8: Data Scaled for Momentum Features

Through the careful examination of the convergence and divergence of moving averages, mean reversion identification signals are generated.

```
In [10]: df['Stochastic_Oscillator_Label'] = 0

         df.loc[df['momentum_stoch'] > 80.0,'Stochastic_Oscillator_Label'] = '-1'
         df.loc[df['momentum_stoch'] < 20.0,'Stochastic_Oscillator_Label'] = '-1'

         df.loc[(df['momentum_stoch'] > 30.0) & (df['momentum_stoch'] < 70.0),'Stochastic_

         df['Stochastic_Oscillator_Label'] = df['Stochastic_Oscillator_Label'].astype(int)
```

Figure 5.9: Feature Extraction for Stochastic Oscillator

- **Stochastic Oscillator Indicators:** These indicators are the indicators that live the relative level of the close price in relevance with the utmost and the least of previous close prices. Once the comparisons indicate the market is overbought or oversold, signals are also generated.

- **Volatility Indicators:** This is the type of indicator that valuates the widening of range between high and low prices. Volatility indicators use models in order to cipher the calculable return over variation of the window sizes. Signals area unit generated with area unit huge increases or decreases in the measures.

- **Volume Indicators:** These indicators monitor positive and negative volume flows that area unit indicative of market reversal. A number of oscillators and volume based mostly in-dices produce signals for bull or bear markets at the moment that the indices cross moving averages.

We generated fifteen technical indicators features, including the daily volume, the open, high and close price,

Feature/Agents	Generated Values
Trend	NoTrend, Uptrend, Downtrend
Moving Average Crossover	Hold, Buy, Sell
Volume	Strong-Volume, Weak Volume
Stochastic	Hold, Buy, Sell
ADX	Strong-Trend, Weak Trend
Momentum	Hold, Buy, Sell

Table 5.1: Generated Features

We applied technical indicators and generated values as shown in table 5.1. Here Buy, Uptrend, Strong Volume and Strong trend signal means -1 in our data-set. Sell, Downtrend, Strong Volume and signal means 1. However Hold and No trend is denoted by 0.

```
In [6]: df
```

Out[6]:

	Company	Date	Open	High	Low	Close	Volume
1	GP	4/1/2010	190.0	198.9	190.0	197.9	2749600
2	GP	5/1/2010	199.0	202.0	195.2	196.5	2061800
3	GP	6/1/2010	196.0	197.8	195.0	196.4	1436200
4	GP	7/1/2010	197.0	197.0	193.5	193.8	1259600
5	GP	10/1/2010	193.9	194.4	190.2	192.3	1605400
6	GP	11/1/2010	193.9	197.4	192.8	196.8	1784800
7	GP	12/1/2010	197.4	207.0	197.4	206.0	5308600
8	GP	13-01-2010	206.5	209.0	197.6	201.2	2952200

Figure 5.10: Daily Data of GP Stock

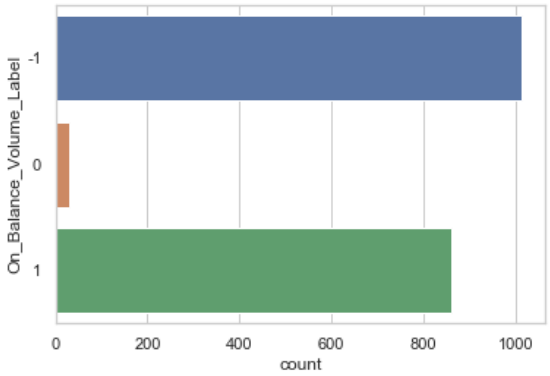


Figure 5.11: Generated Values using OVB Indicator

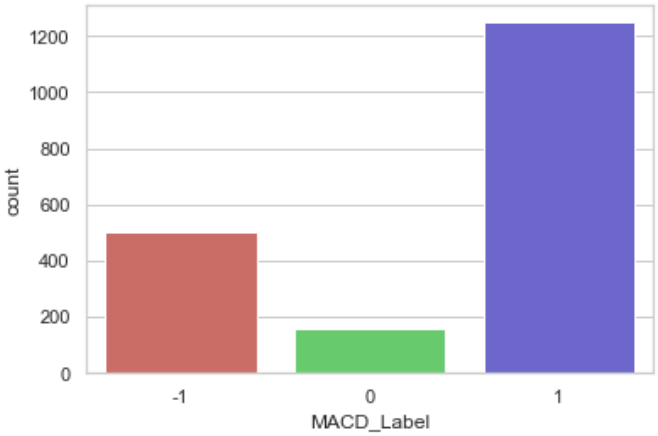


Figure 5.12: Generated Values using MACD Indicator

Chapter 6

Trading System Results

6.1 Results from Recurrent Neural Network (LSTM)

The stock data of GP has been split into Training and Test sets to avoid overfitting the model. The split has been done as shown in Figure 6.1

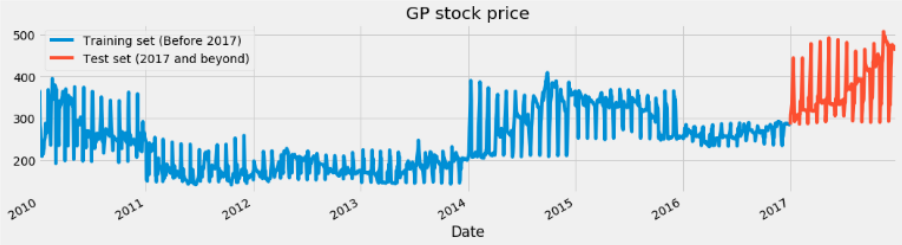


Figure 6.1: Train-test split of GP stock data

Figure 6.2 shows the gradual progression of the loss of the model per epoch as it is being trained. We run 20 epochs which is sufficient to get a an accurate model. The prediction of RNN LSTM model is plotted on Fig 6.3 and compared with the actual GP stock price. From this figure, we can see that our RNN LSTM model has been very accurate in correctly predicting the GP stock price movements and as a result, can be reliable to make future predictions about the movement of the stock thereby allowing the users of the information to make correct judgements about whether to buy, sell or hold on a stock.

6.2 Results from Neural Network

Figure 6.4 shows the gradual progression of the accuracy of the Neural Network model per epoch as it is being trained. We run 30 epochs which is sufficient to get a an accurate model in this case.
The prediction of Deep Neural Network model is plotted on Fig 6.5 and compared with the actual decisions about GP stock as obtained in our dataset. From this figure, we can see that our Deep Neural Network model has been very accurate in correctly predicting the GP stock movements and as a result, can be reliable to make future predictions about the movement of the stock thereby allowing the users of the information to make correct judgements about whether to buy, sell or hold on a stock.

```
Epoch 1/20
1603/1603 [==============================] - 25s 15ms/step - loss: 0.0186
Epoch 2/20
1603/1603 [==============================] - 22s 14ms/step - loss: 0.0094
Epoch 3/20
1603/1603 [==============================] - 22s 14ms/step - loss: 0.0084
Epoch 4/20
1603/1603 [==============================] - 22s 14ms/step - loss: 0.0073
Epoch 5/20
1603/1603 [==============================] - 22s 14ms/step - loss: 0.0070
Epoch 6/20
1603/1603 [==============================] - 22s 14ms/step - loss: 0.0059
Epoch 7/20
1603/1603 [==============================] - 22s 14ms/step - loss: 0.0056
Epoch 8/20
1603/1603 [==============================] - 22s 14ms/step - loss: 0.0045
Epoch 9/20
1603/1603 [==============================] - 22s 13ms/step - loss: 0.0046
Epoch 10/20
1603/1603 [==============================] - 22s 14ms/step - loss: 0.0044
Epoch 11/20
1603/1603 [==============================] - 22s 14ms/step - loss: 0.0044
Epoch 12/20
1603/1603 [==============================] - 22s 14ms/step - loss: 0.0040
Epoch 13/20
1603/1603 [==============================] - 24s 15ms/step - loss: 0.0035
Epoch 14/20
1603/1603 [==============================] - 22s 14ms/step - loss: 0.0035
Epoch 15/20
1603/1603 [==============================] - 25s 16ms/step - loss: 0.0035
Epoch 16/20
1603/1603 [==============================] - 22s 13ms/step - loss: 0.0033
Epoch 17/20
1603/1603 [==============================] - 22s 14ms/step - loss: 0.0030
Epoch 18/20
1603/1603 [==============================] - 22s 14ms/step - loss: 0.0027
Epoch 19/20
1603/1603 [==============================] - 22s 14ms/step - loss: 0.0031
Epoch 20/20
1603/1603 [==============================] - 22s 14ms/step - loss: 0.0028
```

Figure 6.2: Training phase showing the loss at each epoch

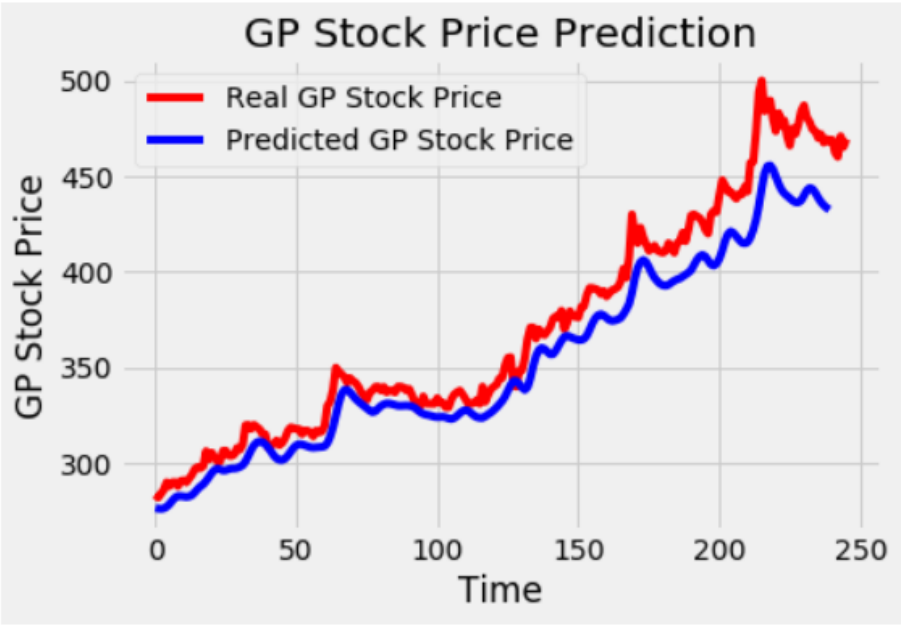


Figure 6.3: Final result of the RNN LSTM netowrk

```
0 Batch accuracy: 0.7254902 Val accuracy: 0.7
1 Batch accuracy: 0.9019608 Val accuracy: 0.7777778
2 Batch accuracy: 0.8627451 Val accuracy: 0.8111111
3 Batch accuracy: 0.8235294 Val accuracy: 0.84126985
4 Batch accuracy: 0.88235295 Val accuracy: 0.85238093
5 Batch accuracy: 0.8627451 Val accuracy: 0.87142855
6 Batch accuracy: 0.9607843 Val accuracy: 0.8888889
7 Batch accuracy: 0.98039216 Val accuracy: 0.8920635
8 Batch accuracy: 0.9607843 Val accuracy: 0.9015873
9 Batch accuracy: 0.9411765 Val accuracy: 0.9031746
10 Batch accuracy: 0.9607843 Val accuracy: 0.9174603
11 Batch accuracy: 0.92156863 Val accuracy: 0.9095238
12 Batch accuracy: 0.92156863 Val accuracy: 0.9126984
13 Batch accuracy: 0.9607843 Val accuracy: 0.9142857
14 Batch accuracy: 0.9607843 Val accuracy: 0.9174603
15 Batch accuracy: 0.98039216 Val accuracy: 0.9285714
16 Batch accuracy: 0.9607843 Val accuracy: 0.9126984
17 Batch accuracy: 0.98039216 Val accuracy: 0.9253968
18 Batch accuracy: 0.9607843 Val accuracy: 0.93650794
19 Batch accuracy: 0.98039216 Val accuracy: 0.93650794
20 Batch accuracy: 0.9607843 Val accuracy: 0.93650794
21 Batch accuracy: 1.0 Val accuracy: 0.9412698
22 Batch accuracy: 1.0 Val accuracy: 0.93333334
23 Batch accuracy: 0.9607843 Val accuracy: 0.93968254
24 Batch accuracy: 0.9411765 Val accuracy: 0.93968254
25 Batch accuracy: 0.98039216 Val accuracy: 0.93650794
26 Batch accuracy: 0.9607843 Val accuracy: 0.9380952
27 Batch accuracy: 0.9607843 Val accuracy: 0.9412698
28 Batch accuracy: 1.0 Val accuracy: 0.947619
29 Batch accuracy: 0.98039216 Val accuracy: 0.94285715
```

Figure 6.4: Training phase showing the validation accuracy at each epoch

```
print("Predicted classes:", y_pred)
print("Actual classes:   ", y_valid[:20])
```

```
Predicted classes: [2 1 2 1 2 2 2 2 1 2 2 1 2 2 1 2 2 1 1 1]
Actual classes:    [2 1 2 1 2 2 2 1 1 2 2 1 2 2 1 2 2 1 2 1]
```

Figure 6.5: A comparison in the class labels between predicted and actual classes

6.3 Results from Random Forest

Random Forest uses an ensemble of multiple decision trees to make it's predictions, and we know that decisions trees make split on the feature variables. As a result, one important metric to better understand the results from Random Forest algorithm is to analyze which feature was split on by the decision trees most often. This metric is represented by the variable importance parameter. We have plotted a table of all the feature variables along with their importance in Figure 6.6

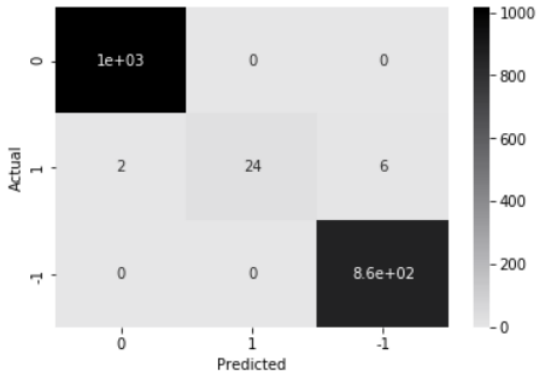
	variable	importance
6	volume_obv	0.480968
11	volume_vpt	0.064747
5	volume_adi	0.040032
57	momentum_stoch	0.031777
59	momentum_wr	0.031700
26	volatility_dchi	0.025064
53	momentum_rsi	0.023463
50	trend_aroon_up	0.021752
58	momentum_stoch_signal	0.017303
51	trend_aroon_down	0.016479
27	volatility_dcli	0.014166
41	trend_cci	0.012055
56	momentum_uo	0.008862
9	volume_fi	0.008300
7	volume_obvm	0.008242
54	momentum_mfi	0.007753
35	trend_adx_neg	0.007546
37	trend_vortex_ind_neg	0.007280
4	Volume	0.007257
36	trend_vortex_ind_pos	0.007122

Figure 6.6: Feature variables along with their importance in descending order. i.e. Most important variable first.

The results from Random Forest algorithm on GP stock data are shown on Figure 6.7. The algorithm outputs various probabilities for each of the decisions as can be seen in the table. We select the decision with the highest probability and append that to the Predicted Decision column.
The results have been plotted in a confusion matrix and a crosstab plot to better understand the accuracy of the Random Forest algorithm. The plots are shown on Figure 6.8.
To get a better evaluation of the Random Forest model's performance, we ran the model on the dataset of another company. We have used the dataset of Square

	Decision	-1	0	1	Predicted Decision
0	0	0.144	0.721	0.135	0
1	0	0.077	0.146	0.777	1
2	-1	0.668	0.077	0.255	-1
3	-1	0.972	0.007	0.021	-1
4	-1	0.977	0.005	0.018	-1
5	-1	0.966	0.022	0.012	-1
6	1	0.050	0.086	0.864	1
7	1	0.019	0.030	0.951	1
8	-1	0.674	0.059	0.267	-1
9	1	0.036	0.014	0.950	1
10	-1	0.868	0.042	0.090	-1

Figure 6.7: Random Forest algorithm prediction of Buy, Sell, Hold decision on GP stock

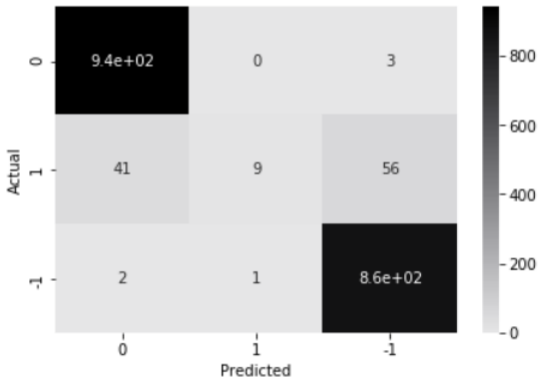


Predicted	-1	0	1	All
True				
-1	1015	0	0	1015
0	2	24	6	32
1	0	0	862	862
All	1017	24	868	1909

Figure 6.8: GP stock predictions by Random Forest algorithm on a confusion matrix and a crosstab plot

	Decision	-1	0	1	Predicted Decision
0	0	0.144	0.721	0.135	0
1	0	0.077	0.146	0.777	1
2	-1	0.668	0.077	0.255	-1
3	-1	0.972	0.007	0.021	-1
4	-1	0.977	0.005	0.018	-1
5	-1	0.966	0.022	0.012	-1
6	1	0.050	0.086	0.864	1
7	1	0.019	0.030	0.951	1
8	-1	0.674	0.059	0.267	-1
9	1	0.036	0.014	0.950	1
10	-1	0.868	0.042	0.090	-1

Figure 6.9: Random Forest algorithm prediction of Buy, Sell, Hold decision on SQUARETEXT stock



Predicted	-1	0	1	All
True				
-1	942	0	3	945
0	41	9	56	106
1	2	1	864	867
All	985	10	923	1918

Figure 6.10: SQUARETEXT stock predictions by Random Forest algorithm on a confusion matrix and a cross-tab plot

Textiles Ltd (SQUARETEXT). The results are shown on Fig. 6.9
Figure 6.10 shows the results which have been plotted in a confusion matrix and a crosstab plot to better understand the accuracy of the Random Forest algorithm on the SQUARETEXT dataset.

6.4 Results from Logistic Regression

The objective of Logistic regression is to find the relationship between one or more independent variable and a dependent variable. The results from Logistic Regression on GP stock data are shown in Figure 6.11. We have took the indicators decision as input feature to buy or sell stocks. In our data-set -1 means buy, 1 means sell and 0 means hold. We splitted our dataset in two part. Our training dataset was .75 and testing dataset was .25. We took fourteen technical indicators as X and one output column y.

accuracy
0.8029350104821803

	precision	recall	f1-score	support
-1	0.80	0.99	0.89	116
0	1.00	0.21	0.34	117
1	0.79	1.00	0.88	244
avg / total	0.84	0.80	0.75	477

Figure 6.11: Classification report of Logistic Regression

Accuracy for our logistic classification is 0.81. Precision is the measurement ratio of correctly identified positive occurrence to the total predicted positive occurrence. In other words it is defined as as the ratio between true positives to the sum of true and false positives. In our case the precision value was 0.84.

Recall for our regression is 0.80. Recall is the ratio of correctly predicted positive observations to the all observations in actual class - yes. In alternative words it is defined as the ratio of true positives to the sum of true positives and false negatives.

F1 Score is the weighted average of Precision and Recall. Generally speaking, F1 scores are lower than accuracy measures as they embed precision and recall into their computation. Here we got 0.75 as our f1-score.

Confusion Matrix	Buy(-1)	Hold(0)	Sell(1)
Buy (-1)	115	0	1
Hold (0)	28	24	65
Sell (1)	0	0	244

Figure 6.12: Confusion Matrix

This logistic classification generated 3x3 confusion matrix. Here we have correctly identified 115 buy signals, 24 hold signals and 244 sell signal. However 28 buy signals were wrongly identified as hold. 65 sell signals also wrongly identified.

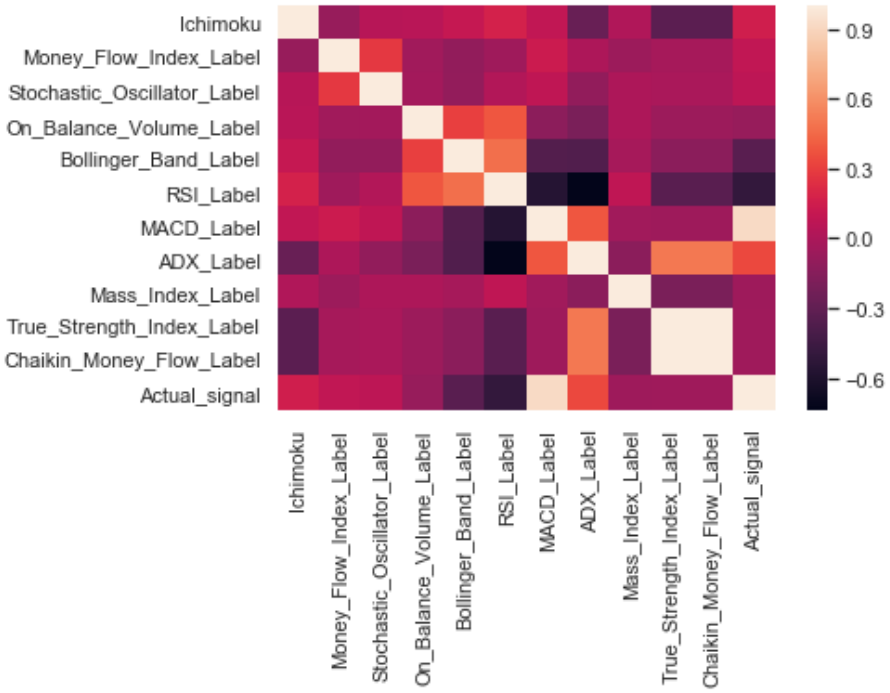


Figure 6.13: Correlation of Different Indicators

Data Correlation is a way to understand the relationship between multiple variables and attributes in dataset. Using Correlation, we can get one or multiple attributes are associated with other attributes.

When evaluating models, we frequently wish to assess however well it performs in predicting the target variable on totally different subsets of the data. One such technique for doing this is often k-fold cross-validation, that partitions the info into k equally sized segments. One fold is control out for validation whereas the opposite k-1 folds square measure accustomed train the model then accustomed predict the target variable in our testing information. This method is perennial k times, with the performance of every model in predicting the hold-out set being half-tracked employing a performance metric like accuracy. the foremost common variation of cross validation is 10-fold cross-validation. Cross-validation mean score for our model was 0.85.

6.5 Results from Support Vector Machine

The support vector machine (SVM) is a predictive analysis data-classification algorithm that assigns new data elements to one of labeled categories. SVM is, in most cases, a binary classifier; it assumes that the data in question contains two possible target values.
From fig 6.14 x-axis means the date and y axis denotes closing price. Here we have plotted last 4 years closing price of GP. We splitted or dataset in traning and testing part where .25 for testing and other for training.The training data set is used for the training of SVM. The algorithm uses the data from the training data set to learn rules for prediction.

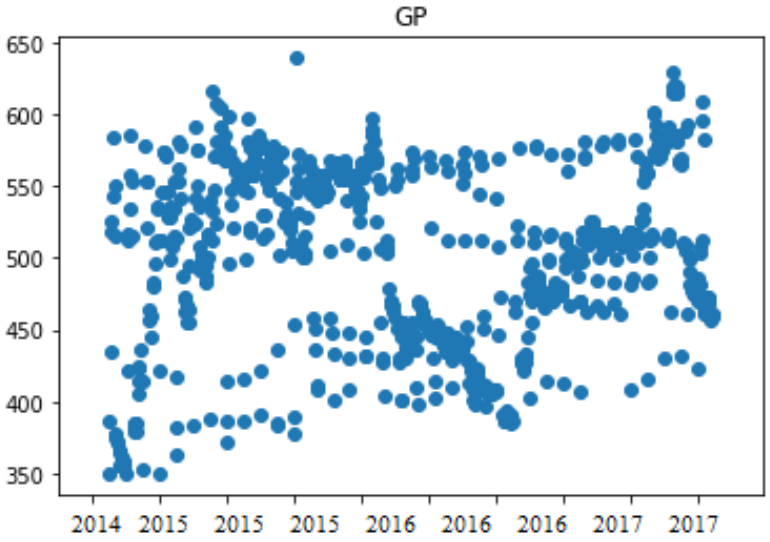


Figure 6.14: Scatter Plot of GP closing price

```
In [11]: window_lengths = [7,14,21,30,60,90,120,150,180]
         accuracys = {}
         reports ={}

         for l in window_lengths:
             print('window_length:',l)
             train_and_test(GP_prices,l,accuracys,reports)
```

Figure 6.15: Different Window Size

The testing data set is used for testing your model on data that was not used for training. So whether the rules learned by the training data set also apply to the testing data, therefore an error rate is computed.

We have run our algorithm for different window of our dataset. We have used different kernel rbf, poly and linear of SVM for comparing our results.

```
window_length: 14
The Accuracy of rbf : 0.581006
The Accuracy of poly : 0.441341
The Accuracy of linear : 0.536313
window_length: 21
The Accuracy of rbf : 0.587571
The Accuracy of poly : 0.451977
The Accuracy of linear : 0.412429
window_length: 30
The Accuracy of rbf : 0.577143
The Accuracy of poly : 0.485714
The Accuracy of linear : 0.457143
window_length: 60
The Accuracy of rbf : 0.619048
The Accuracy of poly : 0.464286
The Accuracy of linear : 0.505952
window_length: 90
The Accuracy of rbf : 0.568750
The Accuracy of poly : 0.481250
The Accuracy of linear : 0.475000
window_length: 120
The Accuracy of rbf : 0.614379
The Accuracy of poly : 0.535948
The Accuracy of linear : 0.542484
window_length: 150
The Accuracy of rbf : 0.600000
The Accuracy of poly : 0.531034
The Accuracy of linear : 0.517241
window_length: 180
The Accuracy of rbf : 0.594203
The Accuracy of poly : 0.463768
The Accuracy of linear : 0.485507
```

Figure 6.16: Accuracy of SVM for different window size of dataset

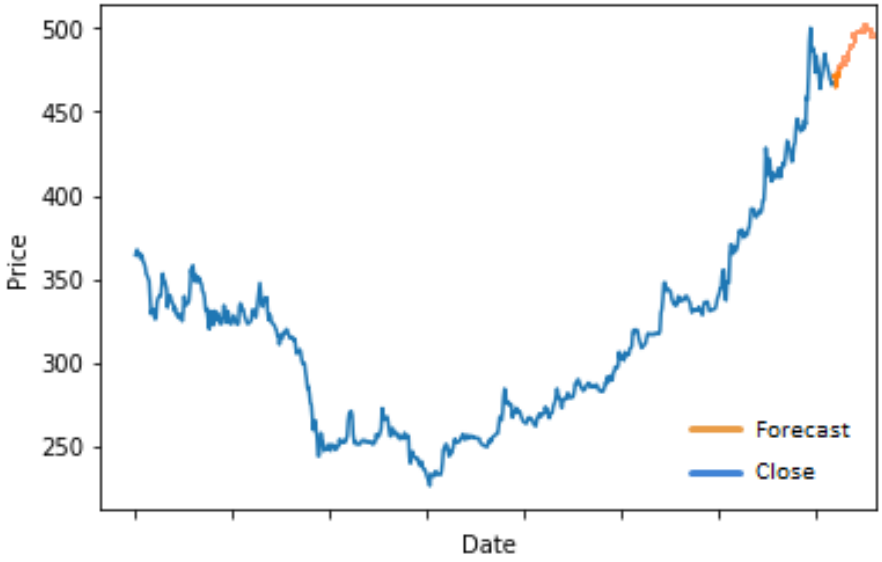


Figure 6.17: Trend prediction for future stock price

Chapter 7

Comparisons

7.1 Comparison of the Machine Learning Models

In this section, we perform an analysis on the comparison between the machine learning models that have been used in this paper. The bedrock of this analysis will be based on the accuracy of the respective models. The accuracy values have been shown on table 7.1 and a bar chart has been plotted for a quick comparison as shown on figure 7.1

As we can see in Table 7.1, Deep Neural Networks have given us the most accurate results of prediction which is a staggering 94.29% on the validation data. This can be explained because we have used a significant number of neurons in the hidden layer of the deep neural network which has, in turn, allowed the model to learn the patterns better and make state-of-the-art levels of accurate predictions.

Next up, the RNN LSTM model also shows considerable accuracy on its predictions but it is less accurate than the Deep Neural Network in our case. At the third place, Random Forests have shown promising accuracy of 83.30% which is also worth wile because further tuning of the hyperparameters would allow us to increase this accuracy figure even further.

Model	Accuracy(%)
Deep Neural Network	94.29
RNN LSTM	86.05
Random Forest	83.30
Logistic Regression	81.54
SVM	61.22

Table 7.1: Comparison among the models used on the basis of their accuracy

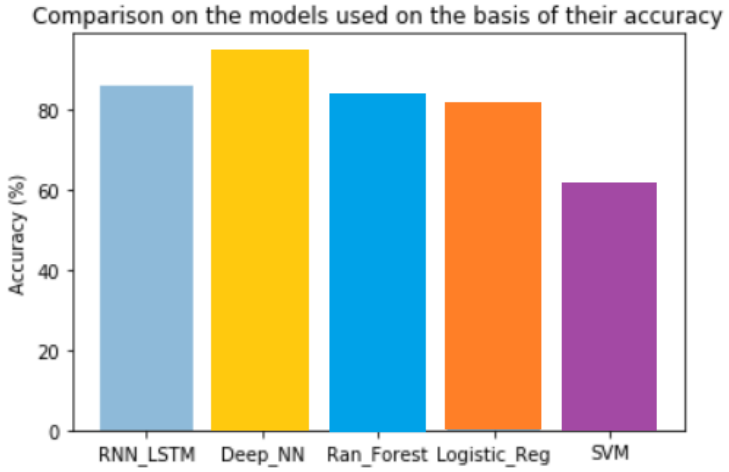


Figure 7.1: Bar chart showing comparison of accuracy among the models

7.2 Limitations of Research

In Bangladeshi context, building a predictive model to analyze stock market shifts and changes is a difficult task. To make a predictive model for Bangladesh, we have to work with Dhaka Stock Exchange (DSE) . But, there is no financial communication platform available in Bangladesh, where the shareholders are communicating with each other over the market share like United States share market. So, to work on a predictive stock market platform, the system in not well structured in context of Bangladesh. Twitter is not a popular social media in Bangladesh. Seldom people use twitter to share emotions via twitter.. Another problem arises through corruption. As investors are not always honest about their share returns, the data becomes corrupted as well as hard to determine some predictions. An alternative approach we thought was to use the Facebook API to get user data related to stock market in Bangladesh, because Facebook is more popular platform in Bangladesh than Twitter. But Facebook API has made its data sharing to a bare minimum in the recent times. So, getting the exact data we need to create a predictive model of stock market was not possible at this time. Moreover, there are not a lot of resource related to Bangladeshi stock market in Facebook. Most, of the resources are limited to different Facebook groups and using Facebook graph API, we can only extract information about an individual user data from that group and use only one group at a time. Facebook posts are not limited in words as well, so the sentiment analysis using those posts do not produce very accurate results.

Another problem is that, individual users do not post about stock markets in their regular posts, most of the posts are more like informative rather than expressive about certain stock market trends which also make sentiment analysis error prone. Therefore, using Facebook API did not seem a good idea. To actually make stock market predictive models based on Bangladesh data, we need a platform for stock related discussions among investors, which is at this time not available. We also need a platform of stock related information which will give access to not only the stock prices of that day only, but historical data of stock market in Bangladesh for an extended period of time. And, as we understood during our research, this might

take some time because most of the investors in Bangladesh are not used to this type of discussion in social media. So, even if someone build a platform, the investors might not use that at all. All of this limited us to not being able to work on stock market of Bangladesh. But, this same model could be used in future, if stock market data and user opinions are available then.

Chapter 8

Conclusion

In this chapter, we conclude the paper with a discussion of the results and limitations of Chapter 6 and recommend further machine learning algorithms for creating viable trading systems. Finally, we discuss the potential areas in which further research could be done.

8.1 Conclusion

The prime focus of this paper was to combine the machine learning algorithms' predictive power along with the technical indicators' pattern/trend detection abilities to come up with a superior model capable of identifying future market trends as well as predict future prices to allow the users of the data to benefit from increased returns and understand the market dynamics as well. This will allow for less riskier investments and more profits in the long and short run.

Our research has showed that text rich data contains information that can have a substantial effect on the market and if this data can be understood and structured to give quantitative value about the content context, it can be used to predict security movement in the market. Technical analysis of this data is one of the many ways these data can transformed into a numeric value, which can be used to gain profit from the market. Different widely used machine learning algorithms applied to financial data to create viable trading strategies. We examined the performance of these models on the out-of-sample test sets and our empirical results showed that.

8.2 Future Work

As we have established through our work that machine learning can stably predict the security movement, we think there is more scope to work on making the investment more dynamic and intelligently responsive to the market. The extension of the LSTM model can be the use of Gated Recurrent Units (GRUs) which have outperformed LSTM cells at certain times. Additionally, we also plan on using the approach of using Generative Adversarial Networks (GANs) to predict stock prices. Moreover, if the algorithms are optimized for minute frequency, it might produce very interesting results. Machine learning agents can be trained on the sentiment data to see how well they can perform using sentiment as feature. It can also be analyzed with different initial capital, to see how the algorithms behave.

Lastly and most importantly, our work is not in a condition to use as a general user. We are planning to build an online tool, which will update itself with new market information and users can perceive market trends at home and decide whether to buy, sell or retain any stock. All these remains areas of future research.

Bibliography

[1] G. Grudnitski and L. Osburn, *Forecasting sp and gold futures prices: An application of neural networks*. Journal of Futures Markets, 13:631–643, 1993.

[2] J. Murphy, *Technical analysis of the financial markets: a comprehensive guide to trading methods and applications*. Prentice Hall Press, 1999.

[3] S.-W. Yu, *Forecasting and arbitrage of the nikkei stock index futures: An application of backpropagation networks*. Asia-Pacific Financial Markets, 6:341–354, 1999.

[4] E. H. Miller-Keith L. Sorensen and C. K. Ooi, *The decision tree approach to stock selection*. Journal of Portfolio Management, 27:42, 2000.

[5] M. A. Dempster, T. W. Payne, Y. Romahi, and G. W. Thompson, "Computational learning techniques for intraday fx trading using popular technical indicators", *IEEE Transactions on neural networks*, vol. 12, no. 4, pp. 744–754, 2001.

[6] P. Ming D. Phua and W. Lin, *Neural network with genetically evolved algorithms for stocks prediction*. Asia-Pacific Journal of Operational Research, 18:103–107, 2001.

[7] H. Chang Laiwan Yang and I. King, *Support vector machine regression for volatile stock market prediction*. IDEAL, 2412:391–396, 2002.

[8] Y. I. Chang, *Boosting SVM Classifiers with Logistic Regression.*, 2003.

[9] K.-j. Kim, *Financial time series forecasting using support vector machines*. Neurocomputing, 55:307–319, 2003.

[10] Linløkken and S. F. olich, *Technical stock analysis: for reduced risks and increased returns*. Investtech.com, 2004.

[11] B. Freisleben, *Stock market prediction with backpropagation networks. In-dustrial and Engineering Applications of Artificial Intelligence and Exper Sys-tems*. 604:451–460, 2005.

[12] N. Yoshiteru Huang Wei and S.-Y. Wang, *Forecasting stock market movement direction with support vector machine*. Computers Operations Research, 32:2513–2522, 2005.

[13] C.-I. H. Yuan Lin Hsu and P. L. Hsu, *Financial performance prediction using constraint-based evolutionary classification tree (cect) approach*. Advances in Natural Computation, 3612:812–821, 2005.

[14] C. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[15] M. Pekkaya and C. HamzaÃğebi, *An application on forecasting exchange rate by using neural network*. Congress of YAEM, 27, 2007.

[16] Y. Zhang and L. Wu, *Stock market prediction of sp 500 via combi-nation of improved bco approach and bp neural network*. Expert Systems with Applications, 36:8849–8854, 2009.

[17] Z. Liao and J. Wang, *Forecasting model of global stock index by stochastic time effective neural network*. Expert Systems with Applications, 37:834–841, 2010.

[18] B. B. Modhandas V.P Nair and N. Sakthivel, *A genetic algorithm opti-mized decision tree-svm based stock market trend prediction system*. Interna-tional journal on computer science and engineering, 2:2981–2988, 2010.

[19] A. C. H. Erol Egrioglu and U. Yolcu, *Fuzzy time series forecasting with a novel hybrid approach combining fuzzy c-means and neural networks*. Expert Systems with Applications, 40:854–857, 2013.

[20] Q. W. Qing-Guo; Li Jin Qin and S. S. Ge, *Linear and nonlinear trading models with gradient boosted random forests and application to singa-pore stock market*. Journal of Intelligent Learning Systems and Applications, 5:1–10, 2013.

[21] Y. J. Chen, "Enhancement of stock market forecasting using a technical analysis-based approach", *2014 IEEE 5th International Conference on Software Engineering and Service Science, Beijing*, vol. 1, no. 4, pp. 702–705, 2014. [Online]. Available: :http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6933664&isnumber=6933501.

[22] A. Gerding Enrico Booth and F. McGroarty, *Automated trading with performance weighted random forests and seasonality*. Expert Systems with Applications, 41:3651–3661, 2014.

[23] J. Schmidhuber, "Deep learning in neural networks: An overview", *Neural networks*, vol. 61, pp. 85–117, 2015.

[24] J. Shah Shail Thakkar Priyank Patel and K. Kotecha, *Predicting stock and stock price index movement using trend deterministic data preparation and ma-chine learning techniques*. Expert Systems with Applications, 42:259–268, 2015.

[25] Y. Li, F. Wang, R. Sun, and R. Li, "A novel model for stock market forecasting", *2016 9th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, pp. 1995–1999, 2016.

[26] A. Géron, *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems.*" O'Reilly Media, Inc.", 2017.