GAZİ ÜNİVERSİTESİ

Gazi

1926

COMPARISON OF DIFFERENT MACHINE LEARNING TECHNIQUES IN PREDICTING THE PRICE MOVEMENT OF BORSA ISTANBUL STOCK MARKET

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
GAZİ UNIVERSITY

BY

Tarik ZİYADOĞLU

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

DECEMBER 2018

The thesis study titled "COMPARISON OF DIFFERENT MACHINE LEARNING ECHNIQUES IN PREDICTING THE PRICE MOVEMENT OF BORSA ISTANBUL STOCK MARKET" is submitted by Tareq ALSHIKH WARAQ in partial fulfillment of the requirements for the degree of Master of Science in the Department of Computer Engineering, Gazi University by the following committee.

Supervisor: Assoc. Prof. Dr. Necaattin BARIŞÇI

Department of Computer Engineering, Gazi University

I certify that this thesis is a graduate thesis in terms of quality and content …..………..…………..

Chairman: Assoc. Prof. Dr. H. Murat ÜNVER

Department of Computer Engineering, Kırıkkale University

I certify that this thesis is a graduate thesis in terms of quality and content …..………..…………..

Member: Assoc. Prof. Dr. Hüseyin POLAT

Department of Computer Engineering, Gazi University

I certify that this thesis is a graduate thesis in terms of quality and content …..………..…………..

Date: 30/11/2018

I certify that this thesis, accepted by the committee, meets the requirements for being a Master of Science Thesis.

…………………….…….

Prof. Dr. Sena YAŞYERLİ

Dean of Graduate School of Natural and Applied Sciences

ETHICAL STATEMENT

I hereby declare that in this thesis study I prepared in accordance with thesis writing rules of Gazi University Graduate School of Natural and Applied Sciences;

- All data, information and documents presented in this thesis have been obtained within the scope of academic rules and ethical conduct,
- All information, documents, assessments and results have been presented in accordance with scientific ethical conduct and moral rules,
- All material used in this thesis that are not original to this work have been fully cited and referenced,
- No change has been made in the data used,
- The work presented in this thesis is original,

or else, I admit all loss of rights to be incurred against me.

Tarık ZİYADOĞLU

30/11/2018

İSTANBUL BORSASININ FİYAT HAREKETİNİ TAHMİN ETMEK İÇİN FARKLI MAKİNE ÖĞRENME TEKNİKLERİNİN KARŞILAŞTIRILMASI

(Yüksek Lisans Tezi)

Tarık ZIYADOĞLU

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

Aralık 2018

ÖZET

Borsa fiyatlarını öngörmek, piyasanın anlaşılmaması nedeniyle daima zorlu bir iştir. Bu çalışmada, BIST100 endeksinin (Borsa İstanbul; Türkiye borsasında ilk 100 şirketin endeksi) günlük dönüş hareketini tahmin etmek için çeşitli türlerde öğrenme modelleri kullanılmıştır. Algoritma olarak; Karar ağacı (DT), Rastgele Orman (RF), K-en yakın komşu (KNN), destek vektör regresyon (SVR), çok katmanlı algılayıcı (MLP) ve uzun - kısa süreli bellek (LSTM) kullanılmıştır. Kullanılan giriş özellikleri; on adet teknik gösterge, FBIST, Euro, Dolar ve altın günlük fiyatlarını içermektedir. Toplanan veriler 2014'ten 2015'e kadar yaklaşık 18 aylık BIST100 fiyatını içermektedir. RMSE bir performans ölçütü olarak kullanılmıştır. Bir sonraki gün, iki, beş ve on gün için tahmin yapılmıştır. Diğer algoritmaların SVR modelinden daha iyi performans göstermesine rağmen, sonuçlarımız diğer tüm algoritmalar üzerinde SVR modeli için istikrarlı bir performans göstermektedir.

Bilim Kodu	:	92432
Anahtar Kelimeler	:	BIST100, Tahmin, Teknik İndikatör, Karar Ağacı (DT), Rastgele Orman (RF), K- En Yakın Komşu (KNN), Destek Vektör regresyon (SVR) Çok katmanlı Perceptron (MLP), Uzun-Kısa Süreli Bellek (LSTM)
Sayfa Adedi	:	55
Danışman	:	Doç. Dr. Necaattin BARIŞÇI

COMPARISON OF DIFFERENT MACHINE LEARNING TECHNIQUES IN PREDICTING THE PRICE MOVEMENT OF BORSA ISTANBUL STOCK MARKET

(M. Sc. Thesis)

Tarik ZIYADOĞLU

GAZİ UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

June 2018

ABSTRACT

Predicting the stock market prices is always a challenging task due to lack understanding of the market. In this study, several kinds of machine learning models are employed for predicting the daily return movement of the BIST100 index; the index of the top 100 companies in the "Borsa Istanbul" Turkey stock market. Our used algorithms include; decision tree (DT), Random Forest (RF), K-nearest neighbor (KNN), support vector regression (SVT), multi-layer perceptron (MLP) and long short-term memory (LSTM). The used input features include; ten selected technical indicators, FBIST, Euro, Dollar and gold daily prices. The collected data contains about 18 months of the BIST100 prices from 2014 to 2015. The RMSE has been used as a performance metric. The prediction was made for the next one, two, five and ten days. Our results show a stable performance for the SVR model over all the other algorithms although other algorithms outperformed the SVR model.

Science Code	:	92432
Key Words	:	BIST100, Prediction, Technical Indicators, Decision Tree (DT), Random Forest (RF), K- Nearest Neighbor (KNN), Support Vector Regression (SVR), Multi-layer Perceptron (MLP), Long Short-Term Memory (LSTM)
Page Number	:	55
Supervisor	:	Assoc.Prof.Dr. Necaattin BARIŞÇI

ACKNOWLEDGEMENT

Foremost, from my heart, I would like to thank Assoc.Prof.Dr. Necaattin Barışçı for his all efforts and continuous support through all the working on this research. His guidelines, instructions, and advices were so helpful and precious. To all the committee of our faculty, helping us taking next levels in our academic life, thank you so much. Beside our faculty stuff, I would like to thank Dr. Andrew Ng, for his great courses in both machine learning and deep learning. His very simple style of explanation helped me overcome difficulties and understand really hard and confusing topics. Last but not the least, my thanks to my father, mother, and wife who surrounded me with all love, respect and trust.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SYMBOLS AND ABBREVIATIONS

The symbols and abbreviations used in this study are presented below along with explanations.

Abbrevıatıons	**Descriptions**
ADX	Trend Strength Indicator
CART	Classification And Regression Tree
DT	Decision Tree
EMA	Exponential Moving Average
FBIST	Istanbul Bond Index
KNN	K-Nearest Neighbors
LSTM	Long Short-Term Memory
MACD	Moving Average Convergence/Divergence Oscillator
MLP	Multi-Layer Perceptron
OBV	On-Balance volume
RBF	Radial Basis Function
ReLU	Rectified Linear Unit
RMSE	Root Mean Square Error
RNN	Recurrent Neural Network
ROC	Rate of Change indicator
RSI	Relative Strength Index
SMA	Simple Moving Average
SVR	Support Vector Regression

1. INTRODUCTION

Since the beginning of humanity, enhancing the quality of taken decisions always has a great care of human thinking, because the decisions made now shapes our lives today and our destiny in the future. For that, people always try to make right decisions, correct their wrongs, learn from mistakes and ask for help from who have an experience. One of the most shining tools nowadays that day after day has more interest in decision-making is the Artificial Intelligence. In this study, we are focusing on the decisions related to investment in stock market and revealing the hidden power of AI helping people to invest in the right stock, at the right time and protect their money and future.

The prediction of stock market movement is always a challenging task for traders and data analysts due to the complexity and the abundance factors inside and outside the company that affects the stock price, like the interest rate, the performance of the whole market, political changes, news and the psychology of investors themselves.

Generally, there are two broad categories of methods for choosing a stock invest in, the first one is the fundamental analysis which finds the intrinsic value of the company by looking at the business performance and financial statements that are filed quarterly, like earnings, dividends, cash flow, book value and so on. The investors here have a chance to buy stocks that are below its intrinsic value and vice versa. The second one is the technical analysis, where the value of a stock is evaluated by means of statistics that depends only on historical prices and volume of the stock; here technical analysts try to find trends or patterns in the stock movement. The fundamental analysis works better for long-term trading like years or months, while the technical analysis shines in the short-term trading like millisecond, minutes or days, where we focus on what happing in the stock exchange like the order book and momentum.

One of the most popular theory in the literature relating to stock prediction is the Efficient Market Hypothesis (EMH) introduced by Eugene Fama [1]. The EMH says that the price of the stock is intrinsic and it reflects all relevant information, meaning that the stock cannot be overbought or oversold, and no one can beat the market [2]. According to the hypothesis, the prices have a random walk pattern; the stock has the same probability 50% to go up and down. This theory is highly disputed and often controversial.

The critic of this theory is that the investors respond to the same coming information in different perspectives and different speed, and that affects the fair price of the stock. In addition, there were people who entered the jail because they stole the information from inside the company and used it illegally, meaning that the price does not always reflect all the available information.

Another criticism to EMH is that according to EMH, no investor can beat the market, but it has seen always investors who are beating the market consistently like the very famous investor Warren Buffett, and there is a numerous number of researchers relatively reached to accuracy more than 70% developing models to predict the movement of stocks.

Asil et al. [3] used neuro-fuzzy inference, neural network and support vector machine in predicting the daily BIST100 movement, they analyzed 8 years of data for building the model, their results showed that the support vector machine with accuracy 72% outperforms the other two models, they mentioned that their study is unique where only six features were used. Huang with his colleges [4] also used support vector machine for prediction the direction of NIKKEI 225 index, they showed that the support vector machine provides better results (75% hit ratio) when it combined with linear discriminant analysis, quadratic discriminant analysis, and Elman backpropagation Neural Networks.

In addition, Usmani and his colleges [5] showed that the KSE100 (Karachi Stock Exchange) can be predicted using machine learning techniques. They used single and multi-layer perceptron, Support Vector Machine and Radial Basis Function (RBF), but they showed - Contrary to the above - that the multi-layer perceptron outperforms the other techniques. In addition, they discovered that the oil prices are the most effective factors to the prices of KSE100; it has been considered that this study with its results could not be so dependent.

Patel et al. [6] Tried to predict the future prices of two index from Indian market named CNX Nifty and S&P BSE Sensex. They used 10 years of data for training, the prediction is made for 1, 10, 15 and 30 days in advance, where 10 technical indicators used as features for prediction. They noticed that when the number of days increases the accuracy is getting down, and for solving this issue they used two-stage fusion approach, where support vector regression (SVR) is in the first stage and Artificial Neural Network (ANN), Random Forest (RF) and SVR in the second stage. They showed that there is a significant improvement using two stages hybrid models (rather than single stage prediction models) in a case when ANN is hybridized with SVR, the model's RMSE equals 1.93 in prediction CNX NIFTY next closing price and also 1.96 for S&P BSE Sensex.

News, sentiments, and technical analysis also can be used together to predict the future stock prices according to Attigeri et al. [7]. Enke and Thawornwong [8] used neural network with information gain technique for evaluating the predictive relationships of numerous financial and economic variables. Kim [9] showed that the prediction performance of SVM algorithm is sensitive to the kernel parameter σ^2 and the value of the upper bound C, so it is important to find the optimal value for those two parameters.

Fischer and Krauss in their study [10] compared the performance of long short-term memory network (LSTM) in predicting the S&P500 index with some memory-free methods; random forest, standard deep neural network and standard logistic regression. They showed that the LSTM outperforms the other methods with returns of 0.46 percent per day, 0.43 percent for random forests, 0.32 percent for standard neural network and 0.26 percent for logistic regression.

Hakan et al. [11] Collected the hourly prices of 100 companies existed in BIST 100 index between 2011 and 2015 with 6705 number of hours. They trained the system using the first four years and did the test on the last one year. They used two classifiers in prediction the up and down movements; logistic classifier and convolutional neural network classifier. The parameters for two classifiers are different, in logistic regression they select 25 technical indicators as an input, and in convolutional network they used the raw time series data as an. The reason for using the raw data in CNN is that it can automatically extract the features and non-linear relations from data without the need of intervention from human, providing strong alternative to existing feature-based models. Their results showed an outperformance of convolutional network with 0.563 F-Measure rates, and 0.545 F-Measure rates for logistic regression.

Rajashree & Pradipta Dash [12] used a Specific type of neural network called computational efficient functional link artificial neural network (CEFLANN). This neural network has only one-layer structure but it showed a superior prediction performance over other machine learning techniques including; support vector machine (SVM), K-nearest neighbor (KNN), decision tree (DT) and Naïve Bayesian. The network in this study is trained using ELM instead of traditional back propagation algorithms, where the output weighs is obtained analytically using robust least squares solution including a parameter for regularization. The input data for a network is six technical indicators generated from five years prices of two stock indices (BSE SENSEX and S&P 500), where the output is three different classes; buy, sell and hold. The CEFLANN exceeds all other profits of the previously denoted algorithms with a profit of 47.2007 % for BSE SENSEX and 24.2872 % for S&P500.

Sasan et al. [13] Applied multiple diverse of classifiers to produce fusion models, where a set of diversity methods applied to create different ensembles, including AdaBoost, Boosting and Bagging. Their results showed that the Bagging outperforms the previously mentioned methods with maximum accuracy 83.6% using Decision Tree, LAD Tree and Rep Tree in prediction the return, and 88.2% accuracy with BF Tree, DTNB and LAD Tree in prediction the risk, the experiments are applied on Tehran Stock Exchange (TSE) data from 2002 to 2012 period.

In this research, the power of machine learning will be unveiled for predicting the closing price movements of the Borsa Istanbul Stock Exchange Index BIST100 using 10 chosen technical indicators, the prices of FBIST, Euro, Dollar and gold. Different machine learning techniques were used in building our models including; support vector Regression (SVR), k-nearest neighbor (KNN), decision tree (DS), random forest (RF), deep multilayer perceptron network (ANN) and recurrent neural network; long short-term memory (LSTMThe performance of each model is compared to other models separately.

The rest of this paper is organized as the following, in the material and methods part the data collection and data pre-processing has been explained in details, also each used technical factor one by one. After that, the used models have been shown and discussed. In the results and discussions part, the obtained results have been shown with all tables and numbers. At the end of this study, the suggested future work has been introduced to enhance this research.

2. MATERIAL AND METHODS

2.1. Data Collection & Date Pre-Processing

Our compiled data contains the volume, open, high, low and close prices of the BIST100 index for every single day between 2014-01-01 and 2015-6-1 (18 months). The all 10 used technical indicators are calculated using this data as explained in details within the next section. The other features contain the closing daily prices of FBIST (Istanbul Bond Index), Euro, Dollar and gold closing prices for the same previously mentioned date. All data are retrieved from the investing server.

In data cleaning phase, string data has been converted to numbers, like converting K to 1000, M to 1 000 000 and B to 1000 000 000 so it can be processed as numbers. Also the comma character "," has been deleted and replaced by "." dot character so our used language (Python) can understand it as an integer, e.g. converting the value from 101,218.30 to 101.21830

2.2. Selected Features

In feature selection 10 different technical indicators have been chosen, five of them are the most used indicators in trading [14], other features have been added due to their probable impact on the price of BIST100 including; the daily closing prices of FBIST, the daily closing prices of Euro, Dollar and gold in exchange with TL. The following part explains each used technical indicator with its equations and figures in details.

2.2.1. Momentum Indicator

The Momentum indicator is one of the simplest indicators exists, it measures the speed (or strength) of a movement for a specific stock [15], it can be positive when the price goes up or negative when the price goes down. Many traders look at momentum and see if it is positive, they will buy, and if it is negative, they will sell, because they expect that the momentum will continue. The formula of the momentum indicator is given as the following:

Momentum[t] = Price[t] – Price[t-n] (2.1)

Where n is the number of backward days and the price means the closing price. Figure 2.1. [16] shows two cases of momentum indicator when it is positive and negative.

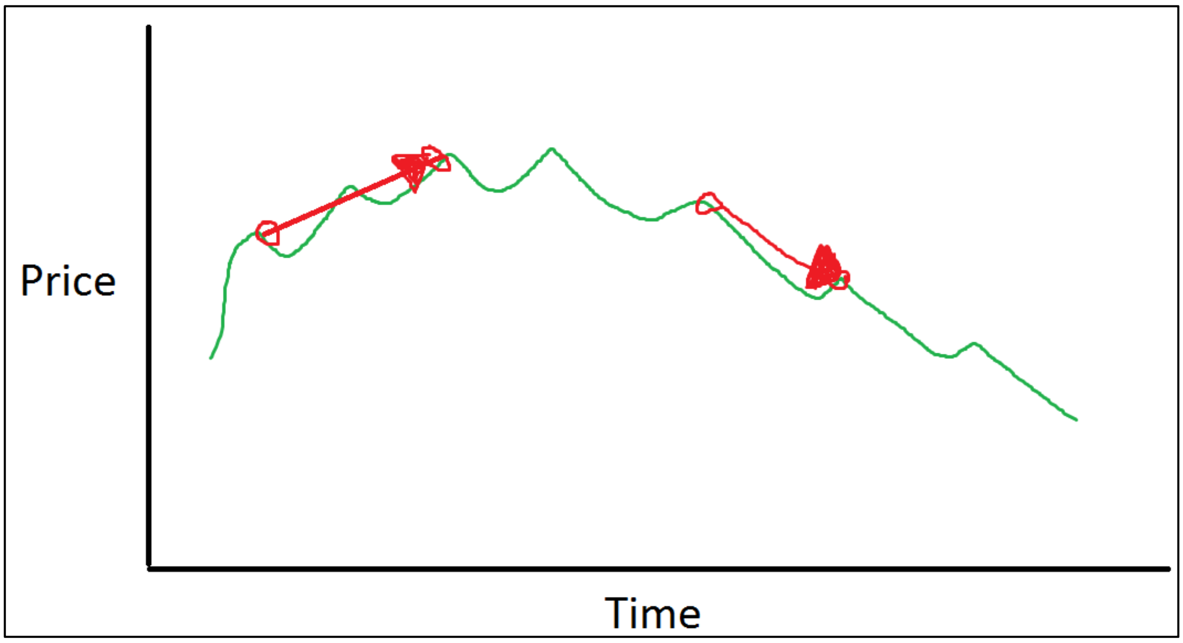

Figure 2.1. Momentum indicator

2.2.2. Simple Moving Average Indicator (SMA)

Moving averages are one of the most important indicators, where there are different kinds of them, moving averages do not predict the future price, but rather smooth the direction of the price and remove the noise in movement with lags in progress [17]. The simplest one is the Simple Moving Average; it is an average of closing prices of a series of data points over given period of time.

The formula of SMA is given by the following:

SMA[t] = mean (price [from t-n to t]) (2.2)

Where n is the number of backward days.

Figure 2.2. [18] shows the SMA line (in red) of the closing stock price (in black).

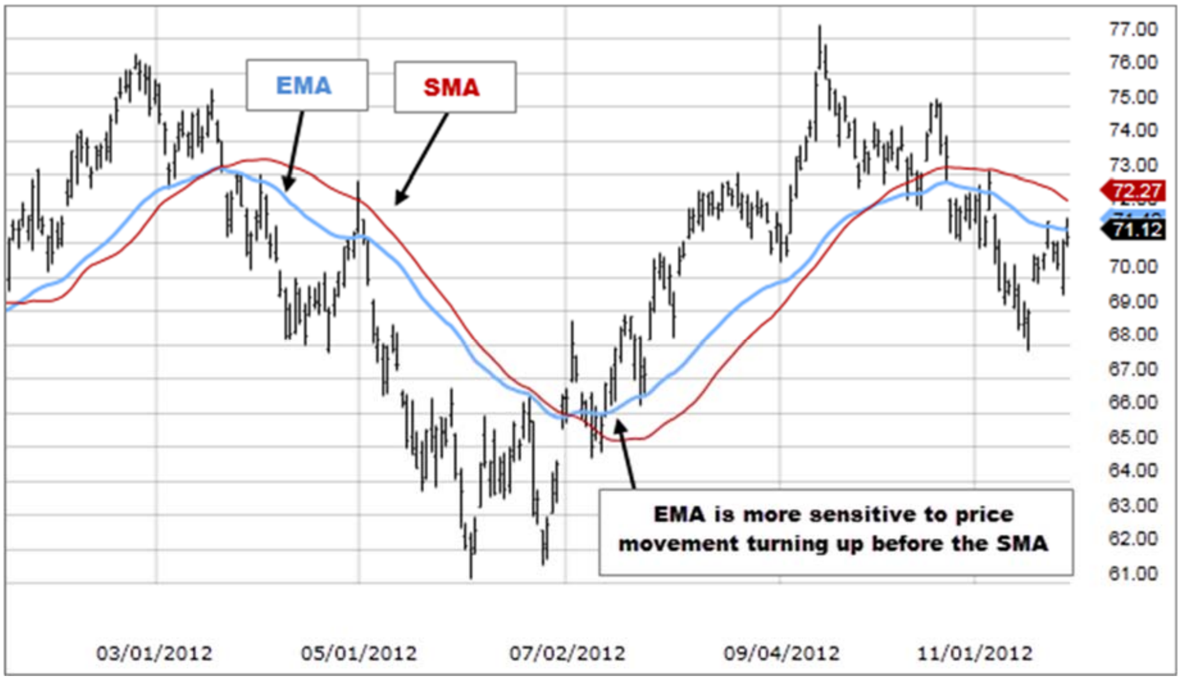

Figure 2.2. SMA with EMA indicator [18]

2.2.3. Exponential Moving Average (EMA)

In simple moving average, all data points have the same weight, the exponential moving average is similar to SMA wherein EMA more weight is given to the most recent days and less weight to the later ones, considering the most recent days more relevant and important. The EMA also responds faster to the changing in price than SMA (where the lag in price is reduced).

In calculating the EMA, the following steps has been applied [19]:

- The simple moving average (SMA) for the initial EMA value was calculated. (Like for 10 days).
- Then the weighting multiplier was also calculated.

$$\text{Multiplier} = (2 / (\text{Time periods} + 1)) \quad (2.3)$$

- For each data, the exponential moving average has been calculated by the following formula:

A = {Close - EMA (prev day)} X multiplier + EMA (prev day) (2.4)

Where the first value of EMA is SMA. Figure 2.2. [18] shows the EMA line alongside with SMA line and the stock price.

2.2.4. Bollinger Bands®

Invented by John Bollinger in the 1980s, where it is used as a measure to know when the deviation from the SMA line is significant enough to generate trading signals. Bollinger bands indicator consists of three lines: the first one is the normal simple moving average; the other two lines are upper and lower bands where the John added tow standard deviation up and down the SMA line for each band. The more the stock price is close to the upper band, the more the stock is probably overbought, and the more the stock price is close to the lower band, the more the stock is considered oversold. The calculation of the bands is given as the following:

The middle band = SMA for the price given selected time period (2.5)

The upper band = The middle band + (standard deviation of the price x 2) (2.6)

The lower band = The middle band - (standard deviation of the price x 2) (2.7)

Figure 2.3. [20] explains the Bollinger Band indicators showing the upper, lower and middle bands.

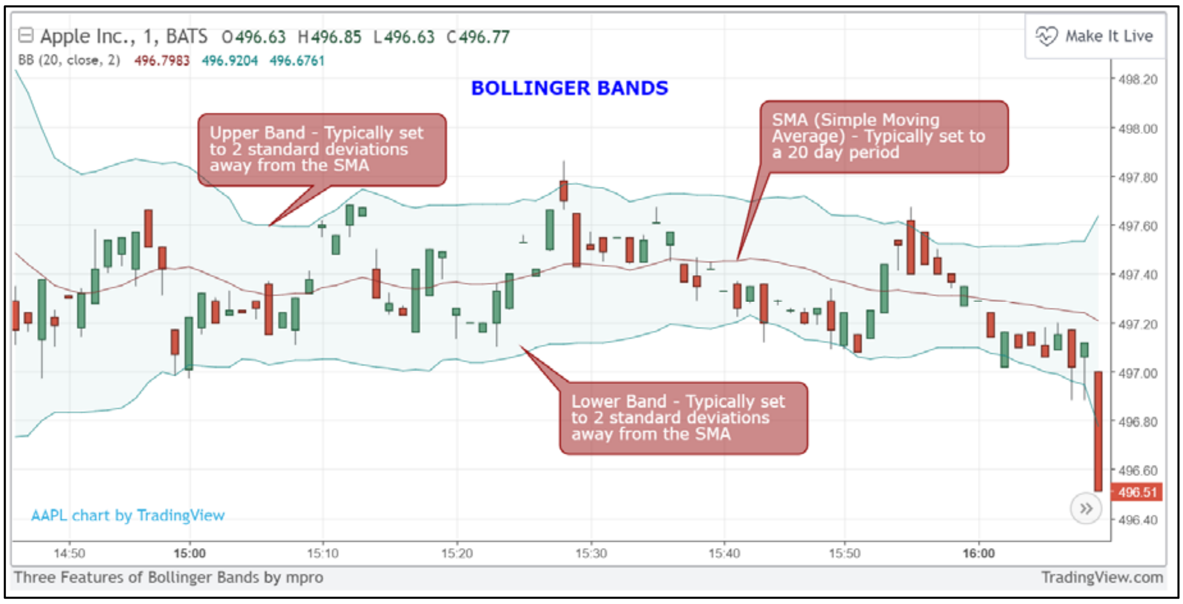

Figure 2.3. Bollinger Band indicator

2.2.5. The Trend Strength Indicator (ADX)

Developed by Welles Wilder in his book *New Concepts in Technical Trading Systems* in 1978. ADX combines three directional movement indicators [21]:

i. ADX: Average Directional Index
ii. -DI: Minus Directional Indicator
iii. +DI: Plus Directional Indicator

The ADX line measures the strength of the trend (not the direction of the trend if it is up or down) and has a value ranging from zero to 100. Many traders consider that when ADX is above 25, there is a trending and this is enough for trend-trading strategies, and when it is below 25, there is no trending [22].

When the ADX is 25 and higher and +DMI is above the -DMI, this indicates a strong uptrend, and when the ADX is 25 and higher and +DMI is below the -DMI, this indicates a strong downtrend. [23], Figure 2.4. [22] explains these cases:

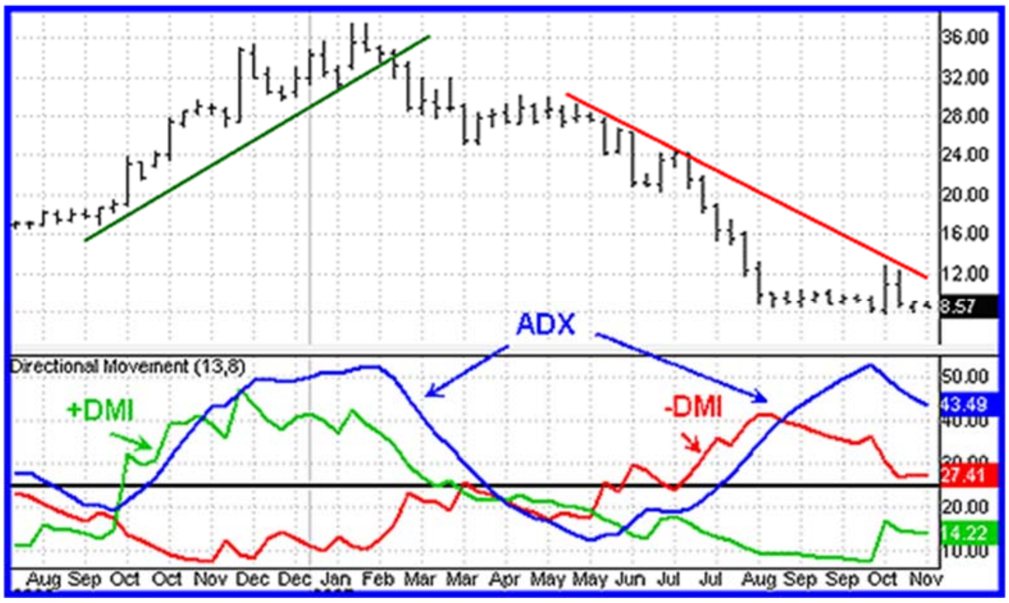

Figure 2.4. ADX indicator

The calculation of ADX indicator values for each period of time is explained as the followings:

$TR=$	$MAX\ [(High - low), ABS\ (High - Close_{prev}), ABS\ (Low - Close_{prev})]$	(2.8)
$+DM_1 =$	if $(High - High_{prev} > Low_{prev} - Low)$ then $Max\ ((High - High_{prev}), 0)$ else 0	(2.9)
$-DM_1 =$	if $(Low_{prev} - Low > High - High_{prev})$ then $Max\ ((Low_{prev} - Low), 0)$ else 0	(2.10)
First $TR_{14} =$	the sum of prev TR_{14} periods	(2.11)
Subsequent $TR_{14} =$	$TR_{14\ prev} - (TR_{14\ prev} / window) + TR_{14}$	(2.12)
First $+DM_{14} =$	the sum of prev $+DM_1$ periods	(2.13)
Subsequent $+DM_{14} =$	$+DM_{14\ prev} - (+DM_{14\ prev} / window) +DM_1$	(2.14)
First $-DM_{14} =$	the sum of prev $-DM_1$ periods	(2.15)
Subsequent $-DM_{14} =$	$-DM_{14\ prev} - (-DM_{14\ prev} / window) + -DM_1$	(2.16)
$+DI_{14} =$	$100 * (+DM_{14} / TR_{14})$	(2.17)
$-DI_{14} =$	$100 * (-DM_{14} / TR_{14})$	(2.18)
$DX =$	$100 * (ABS\ (+DI_{14} - -DI_{14}) / (+DI_{14} + -DI_{14}))$	(2.19)
First $ADX =$	the average of prev DX periods	(2.20)
Subsequent $ADX =$	$((ADX_{prev} * 13) + DX) / 14$	(2.21)

Where the default value for the window is 14 periods as recommended by Wilder and as used in this study, and Prev: means the previous day. Note that the details of calculating the Average True Range indicator are shown to give complete equations for computing the ADX indicator.

TR is the true range.

+DM is the plus directional movement for each period of time, and -DM is the negative one.
ATR_{14} is 14-days smoothed True Range.
$+DM_{14}$ is the smoothed plus directional movement for a given period of time and $-DM_{14}$ is the negative one.
$+DI_{14}$ is the plus directional indicator for a given period, and $-DI_{14}$ is the negative one.
DX is the directional movement index.
ADX is the average directional index.

2.2.6. Moving Average Convergence/Divergence Oscillator (MACD)

One of the most effective and simplest indicators exists, created in the late 1970s by Gerald Appel, MACD consists of three parts [24]:

i. The MACD line, which is calculated by subtracting the EMA (26) from EMA (9).
ii. The signal line, which is simply 9-day of EMA of the calculated MACD.
iii. MACD Histogram, which is the MACD line minus the signal line.

In MACD there is a bullish signal when MACD cross over the signal line, and bearish signal when MACD cross down the signal line, while the histogram explains how the MACD line and signal line converge from each other, the histogram gets bigger when there is a divergence and disappear when there is a cross between the mentioned lines [25]. Figure 2.5. [24] illustrates the MACD indicator system.

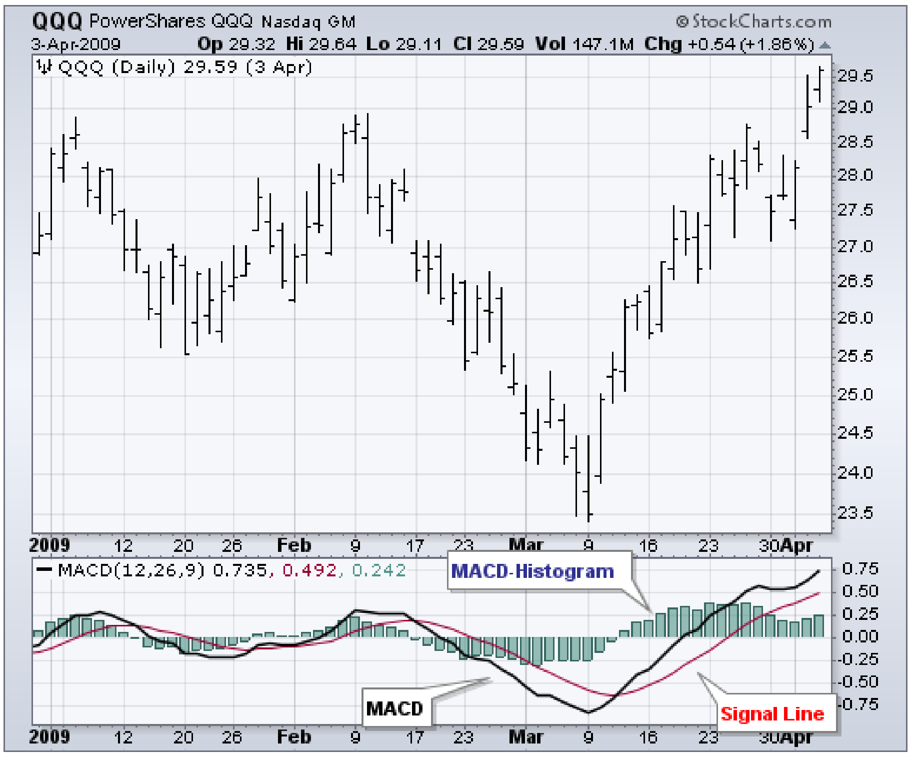

Figure 2.5. MACD indicator

The typical parameters for MACD are 12,26,9 and as used in this study, but other groups of value can be used like 5,35,5, which is more sensitive to signals, all that depends on the selected trading style and goals.

2.2.7. Rate of Change indicator (ROC)

The Rate of change (ROC) indicator is a pure momentum indicator that measures the stock changing speed over a specific period of time [26]. The calculation of ROC [27] is given in the equation below where ROC compares the current price with the price n periods ago. The commonly used time period for ROC indicator is 10 and 12. In this study 10 has been used as a compared period.

$$ROC = \frac{\text{close price} - \text{close price n periods ago}}{\text{close price n periods ago}} * 100 \qquad (2.22)$$

The Figure 2.6. [28] shows the close prices of stock and its ROC line at the bottom of the shape fluctuating around the zero, rising above the zero when the price goes up and falling below the zero when the price goes down.

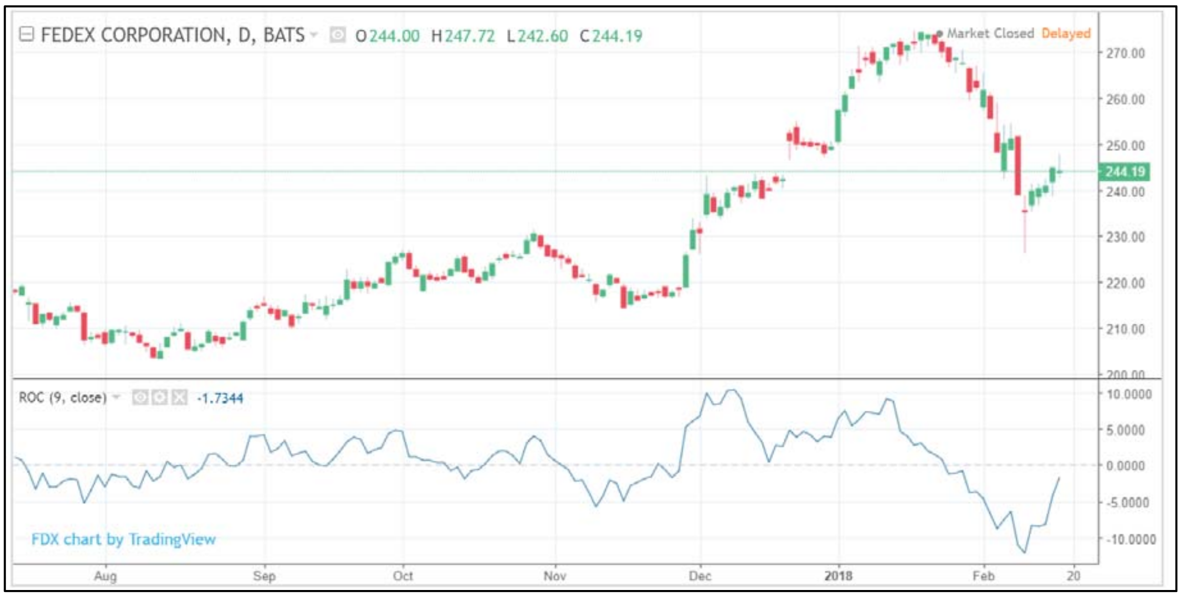

Figure 2.6. ROC indicator

2.2.8. Relative Strength Index (RSI)

Developed also by the well-known technical analyst Welles Wilder in his 1978 book, *New Concepts in Technical Trading Systems*. RSI is a very popular a momentum oscillator, measuring the speed and magnitude of price changes over time, it can have a value between zero and 100 where the price of a stock is considered overbought when RSI is above 70 and oversold when it is below 30 [29]. The calculation of RSI indicator for every day is given as the followings:

$$\text{Change} = \text{Close} - \text{Close}_{\text{prev}} \tag{2.23}$$

$$\text{Gain} = \text{if (Change} > 0\text{) then Change else } 0 \tag{2.24}$$

$$\text{Loss} = \text{if (Change} < 0\text{) then - Change else } 0 \tag{2.25}$$

$$\text{First Avg Gain} = \text{sum of all Gains over the Time Period / Time Period} \tag{2.26}$$

$$\text{Subsequent Avg Gain} = (\text{Gain}_{\text{prev}} * (\text{Time Period -1}) + \text{Gain}) / \text{Time Period}) \tag{2.27}$$

$$\text{First Avg Loss} = \text{sum of all Losses over the Time Period / Time Period} \tag{2.28}$$

Subsequent Avg Loss = (Loss $_{\text{prev}}$ * (Time Period -1) + Loss) / Time Period) (2.29)

RS = Average Gain / Average Loss (2.30)

RSI = 100 – (100/ 1 + RS) (2.31)

Where in this study 14 days has been used as a time period, prev means the previous day and RSI normalizes RS and make it oscillate between zero and 100. Figure 2.7. [30] shows the RSI indicator in the lower part explaining the overbought and oversold cases.

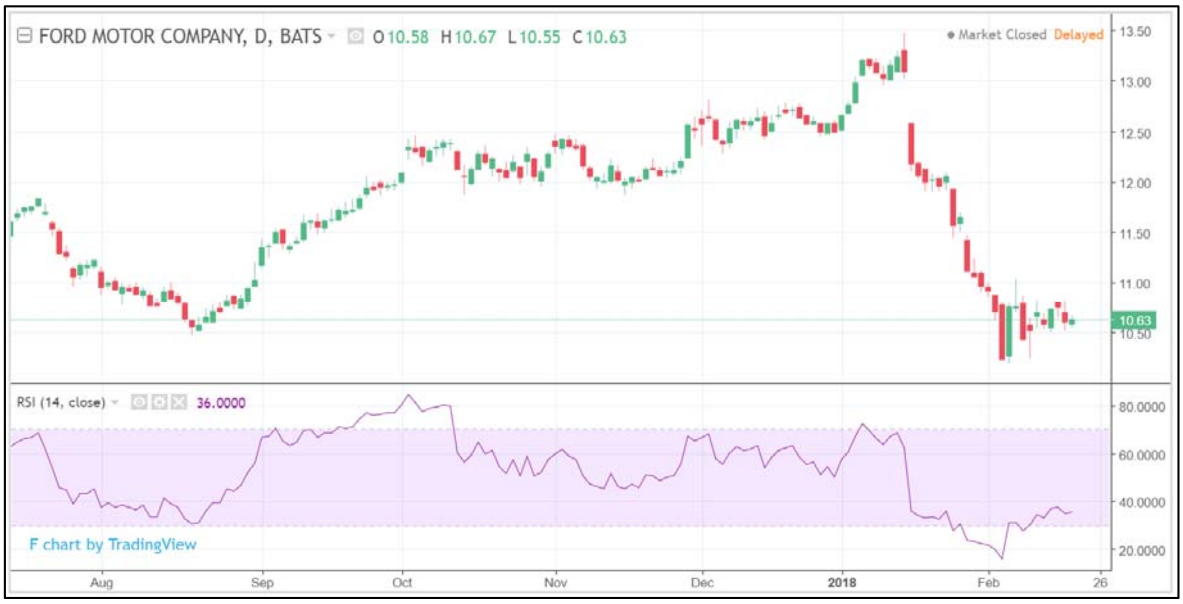

Figure 2.7. RSI indicator

2.2.9. Stochastic Oscillator

Designed by George C. Lane in the late 1950s, stochastic oscillator shows the location of the close price relative to the high and low range of a stock over a certain period of time, typically 14-day period [31]. Lane over his interviews always said that the stochastic oscillator follows only the speed or the momentum of a stock movement and nothing else like the price or volume or anything like that. Lane also mentioned that, as a rule, the momentum changes before the price. That is the reason why stochastic oscillator can be used to predict reversals when it shows bullish or bearish divergences, and this is the most important signal Lane identified.

Stochastic is a bounded oscillator, so it also can be used to predict overbought and oversold levels. The default settings consider that 80 as an overbought threshold and 20 as an oversold threshold, and these settings can be adjusted due to the chartists of a security. It is important to mention that the overbought signal is not necessarily bearish, because the security may generate an overbought signal and still overbought in a strong uptrend, and in the same way for oversold signal, the security may generate an oversold signal and still oversold in a strong downtrend.

Therefore, it is important to analyze bigger trend and act in a direction of this trend, where the trader may ignore frequent oversold readings and aim for infrequent readings in an uptrend, and similarly ignore frequent overbought readings and aim for infrequent readings in a downtrend. The calculation of the indicator is given as the followings for every day [32]:

Lowest Low = Min (Low price) over the past look-back period (2.32)

Highest High = Max (High price) over the past look-back period (2.33)

%K = (Current Close - Lowest Low) / (Highest High - Lowest Low) * 100 (2.34)

%D = 3-day simple moving average of %K (2.35)

Note that %D act as a signal line plotted with %K to identify oversold and overbought signals. The chosen look-back period for this study is the default 14 days.

Figure 2.8. [33] shows the Stochastic Oscillator with the %K and %D lines plotted.

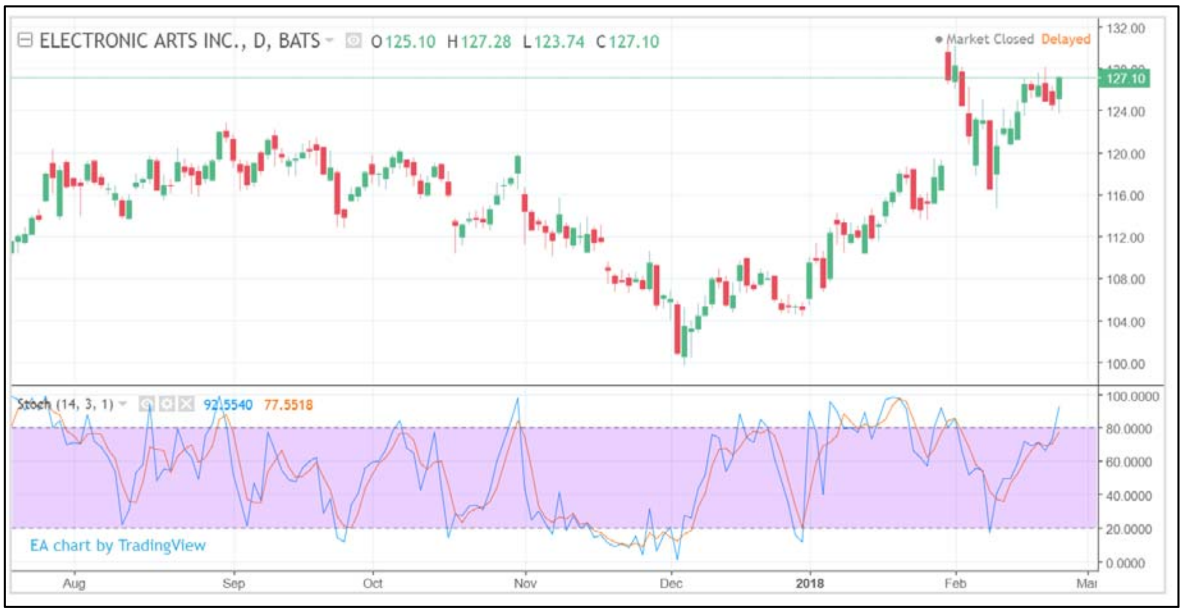

Figure 2.8. Stochastic oscillator

2.2.10. On-Balance volume (OBV)

According to the theory introduced by Charles Dow named Dow Theory, the volume increases if the price moves with the primary trend direction and decreases if the price moves against it. After the Dow Theory, more researchers were made in this subject and the On-Balanced Volume indicator (OBV) is developed [34]. Joe Granville in his 1963 book, *Granville's New Key to Stock Market Profits* Introduced OBV, one of the first indicators designed to measure the volume pressure flow [35], Joe believed that when there is a vast increase in the volume with no change in the price, then the price at some point will spring up or down [36]. Figure 2.9. [37] shows the OBV indicator on the downside of the figure. The calculations of OBV for every day are introduced as the followings:

For the first day:

$$OBV = Volume \tag{2.36}$$

For the next following days:

$$\text{If } (close > close_{prev}) \text{ then} \tag{2.37}$$

$$OBV = OBV_{prev} + Volume$$

Else if (close < close $_{prev}$) then

OBV = OBV $_{prev}$ – Volume

Else if (close = = close $_{prev}$) then (2.38)

OBV = OBV $_{prev}$

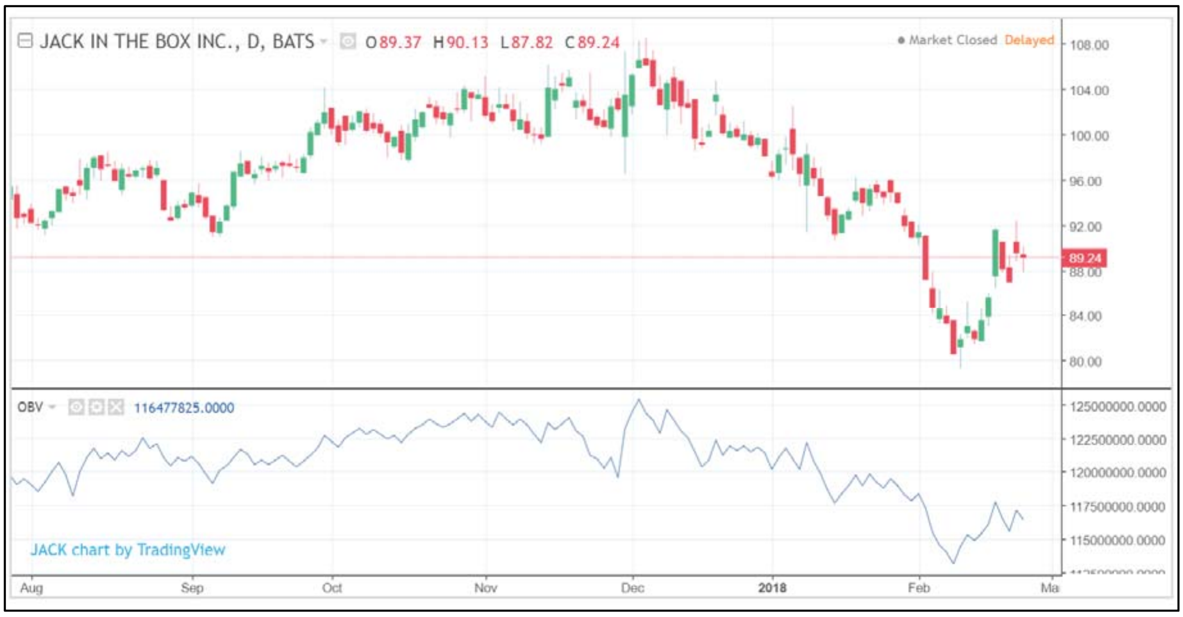

Figure 2.9. OBV indicator [37]

2.3. Selected Algorithms

In this section, the used algorithms for building our six different models will be explained. The Python programming language and two famous libraries have been used as the main building blocks for our systems, the first library named Scikit-Learn, and the second one named Keras on TensorFlow. Only the LSTM model is built using Keras on TensorFlow and the others are built using Scikit-Learn library.

2.3.1. Decision Tree (DT)

Decision trees are considered an intuitive and easy to understand machine learning algorithm, where it is based on partitioning the trained dataset at the first level to find the optimal split, and recursively repeat that for each level [38]. The target variable in decision trees can have categorical or contagious value [39]. The boundaries created by DT are always perpendicular to an axis [40] as illustrated in Figure 2.10. [41].

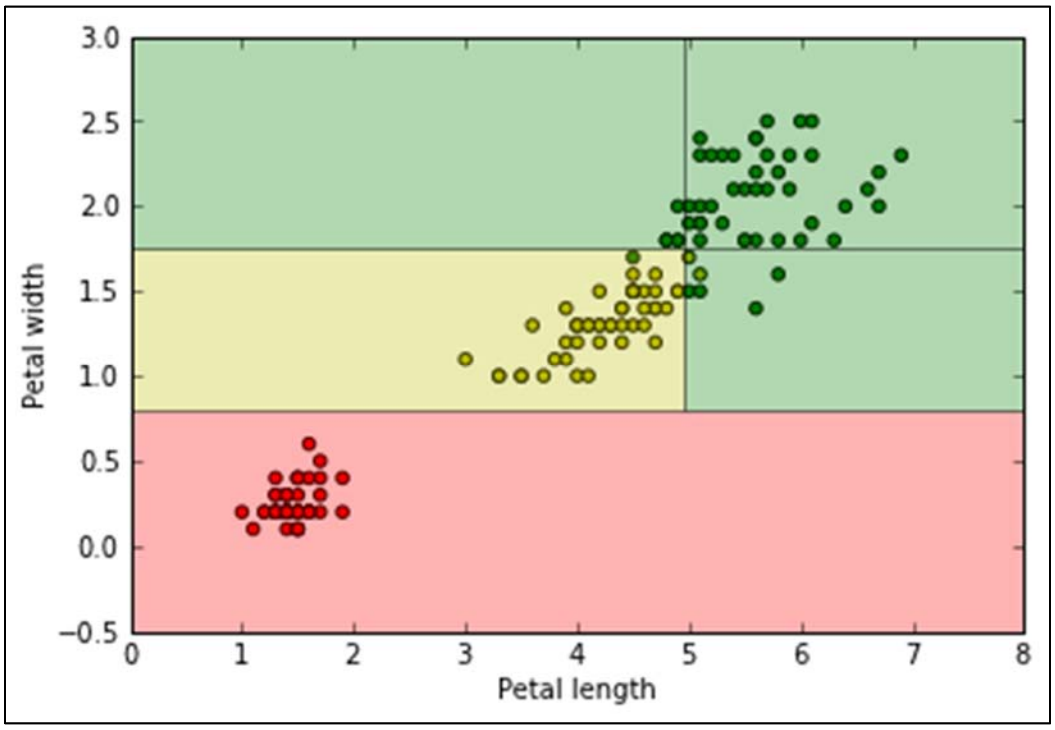

Figure 2.10. Orthogonal decision tree boundaries

Although DT is not balanced all the time, Our used framework Scikit-Learn tries to keep it in a balanced manner, so considering DT balanced, its complexity can be considered equals $O(\log_2(n_{samples})$ for query time and $O(n_{samples} * n_{features} \log_2(n_{samples}))$ for training time [42]. There are several existed algorithms for building decision trees in literature, the used one in this study is *Classification And Regression Tree (CART)* tree.

The mechanism of the CART algorithm can be explained as the followings: at each level, it uses greedy algorithm to find the best combination of feature k and threshold t_k that produces the purest two subsets, the process is repeated for each level. The algorithm stops when there is no split that can enhance the impurity. In addition, pre-defined condition can be used to stop the splitting like, the maximum allowed depth, the maximum leaf nodes or something like these conditions. The cost function that CART algorithms attempt to minimize is given by the following equation [40]:

$$f(k, \mathrm{tk}) = \frac{m_{left}}{m} G_{left} + \frac{m_{right}}{m} G_{right} \tag{2.39}$$

Where G left / right measure the impurity of the left/right branch.

In addition, m left / right is the number of instances in the left/right branch.

For measuring the splitting impurity in classification, the default measure chosen by the framework -as well as in this study- is Gini, also the Entropy measure can be used, but Gini may be more preferred because it is slightly faster than Entropy due to the absence of log calculation, the equation of Gini is given by the following:

$$H_i = \sum_{k=1}^{n} \mathrm{P}_{i,k}(1 - \mathrm{P}_{i,k}) \tag{2.40}$$

Where $\mathrm{P}_{i,k}$ is the ratio of class k instances among the training instances in the ith node.

In regression tasks, CART tries to predict a continuous value instead of a class, where the algorithm splits the data to make the predicted values as close as possible to the training instances. This is achieved by using measures such as mean squared error (MSE). The following equation shows the cost function that CART tries to minimize:

$$f(k, \mathrm{tk}) = \frac{m_{left}}{m} MSE_{left} + \frac{m_{right}}{m} MSE_{right} \tag{2.41}$$

Where

$$MSE_{node} = \sum_{i \in node} \left(\hat{y}_{node} - \mathrm{y}^{(\mathrm{i})}\right) \tag{2.42}$$

$$\hat{y}_{node} = \frac{1}{m_{node}} \sum_{i \in node} \left(\mathrm{y}^{(\mathrm{i})}\right) \tag{2.43}$$

Where m left / right is the number of instances in the left/right branch.

y is the true value and $\hat{y}$ is the predicted one.

DT and such intuitive models fall into the white box machine learning models, where the behavior and prediction steps taken by the algorithm can be easily understood and interpreted, in opposite to the black box models where it is really hard to understand the predictions rules made by the algorithm as the case in neural network [40].

2.3.2. Random Forest

Random Forest has been considered one of the best algorithms exists in machine learning, it can be used for classification and regression tasks, but before explaining the algorithm, a brief introduction of the ensemble and bagging concepts will be given.

Ensemble is a technique in machine learning that combines many classifiers to make more accurate one, where the performance of the combined model exceeds each individual classifier and has less overfitting [43]. In a classification task, the ensemble model selects the result that has most votes, but in regression, the mean of all produced results is calculated by the model [44]. The reason why ensemble model might work better than each individual classifier is that each algorithm might overfit to different aspect of the data, where the ensemble model averages all the mistakes made by each algorithm and produce a better performance [45].

Bagging is an ensemble learner invented by Leo Breiman in 1994 [46], in bagging the combined classifiers use the *same* training algorithm, but they are trained on different sub-set of data, where the sampling is done randomly with replacement; meaning the same model can have repeated samples. Bagging reduces the overfitting and can improve the performance of CART model a lot [47]. There is also a technique called pasting when the sampling of the data is done without replacement.

Random forest generally is an example of bagging technique (it can be sometimes trained via pasting), where it combines several decision trees in one model, each tree in the forest should do well in predicting the target value, but it should be also different from the other trees. This differencing in building these trees can be accomplished by;

i. Firstly, selecting the data randomly for each tree.

ii. Secondly, splitting each created tree by using randomly sub-set selected features from the set of all original features.

Random forests are less likely to overfit the data as the case in decision trees, it does not require extensive parameter tuning as the case in neural networks, also building random forests can be done easily on multiple CPUs [45].

One negative side of random forests is that the prediction structure is not easy to interpret, where the trainer may not fully understand the decisions taken by the algorithm, also random forests is not performing well in problems that have high dimensional sparse features like text classification [45], in comparison to algorithm called Bidirectional LSTM that has a better structure to do this task better. In this study, we used *max depth* technique for stopping the trees in the random forest going very deep and preventing overfitting. The chosen max depth is three.

2.3.3.   K-Nearest Neighbors (KNN)

KNN is one of the simplest and easiest to understand supervised machine learning algorithm. KNN represents an instance-based algorithm, where it does not learn the model or produce any leaning function about it, but instead, it stores all the given data permanently in memory. Each time the model is asked to do the prediction, it queries all the given instances to make the prediction, so both the training and predicting phases may be costly. In addition, KNN is considered non-parametric algorithm, where it does not make assumptions about the data [48] making it free to learn any functional form the training instances [49].

The detailed mechanism of KNN algorithm is illustrated as the followings: while the data points are stored in the memory in the training phase, the new "unseen" data point in the test phase is compared to every single element in the stored dataset using chosen kind of distance measure. The mean of most nearest K points (to the tested data point) is given as an output in the regression case, while the majority of votes is given in the classification case.

In measuring the distance between points, the Euclidian distance measure is the popular one, where its equation is given as the following:

$$d(x, x') = \sqrt{(x1 - x'1)^2 + (x2 - x'2)^2 + \cdots + (xn - x'n)^2} \tag{2.44}$$

While in this study, the Manhattan metric gives better results for our problem. The equation between two points using Manhattan Metric is given by the following [50]:

$$d = \sum_{i=1}^{n} |x_i - y_i| \tag{2.45}$$

Where n is the number of variables.

x_i and y_i are the values of the i_{th} variable, at points X and Y respectively.

Selecting a suitable value for K hyperparameter in KNN model has a crucial role in fitting the data set, where the K value controls the shape of the decision boundary of the model. Choosing a small value for K makes the boundary very jagged as shown in the figure 2.11. [40]. In this case, the model will have a low bias (under fitting) with high variance (overfitting).

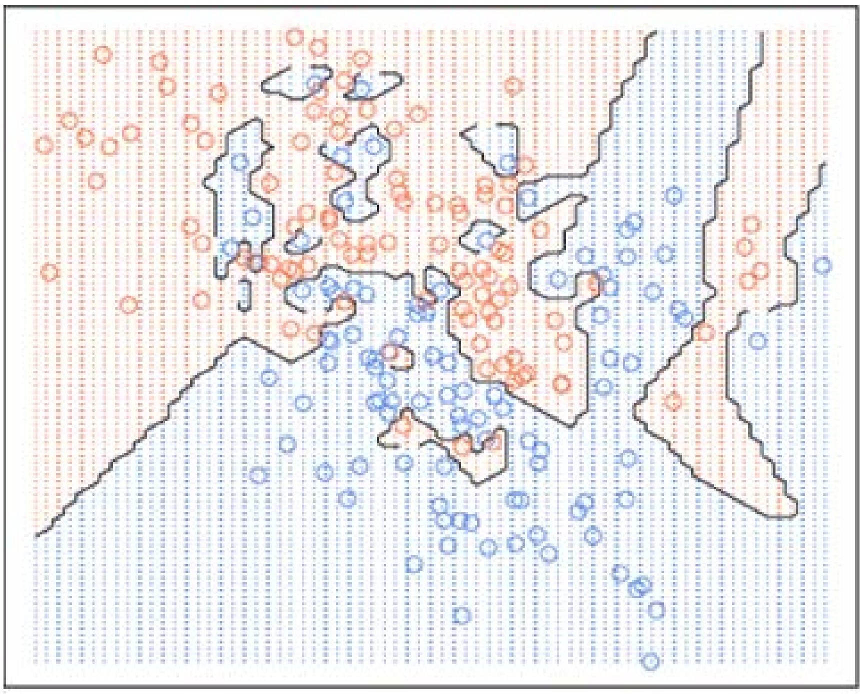

Figure 2.11. KNN decision boundary given K=1

When K becomes larger, the model can generalize better and becomes more impervious to outliers. In this case, the algorithm draws a smoother decision boundary as shown in the Figure 2.12. [40], the model, in this case, has a higher bias with lower variance. In this study, choosing six neighbors gives good results in predictions.

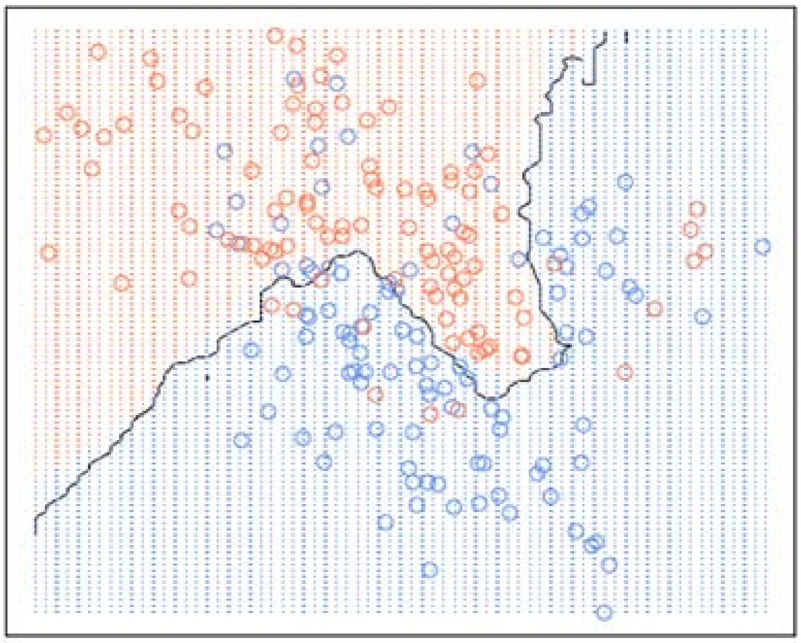

Figure 2.12. KNN decision boundary given K=20

It is hard to implement KNN model in applications that have a very large dataset and very frequent queries because the test time will become slow and unpractical. In contrast to the neural networks, where the test time is very faster than the training time. However, we find that KNN is doing well in stock price prediction where the dataset is not very big.

2.3.4. Support Vector Regression (SVR)

SVR is a very powerful algorithm that is widely used in both the academia and industry [51]. The *Support Vector Regression* is an extension to *Support Vector Classifier*, which is also an extension to intuitive classifier named *maximal margin classifier*. The last one can be applied only when the data is linearly separable, where the *Support Vector Classifier* can be used to classify non-linearly separable data. The most general one is the *Support Vector Regression* where it can construct non-linear class boundaries. Before getting deep into the details of SVR, a brief illustrating of some concepts that are important to make clear view about SVR will be given.

The first thing that should define is the *hyperplane*, in a space with p dimensions, the hyperplane is a flat affine subspace of dimension p -1, where the affine means that the subspace need not pass through the origin. For example, in two-dimension space, the hyperplane is a line, wherein three-dimension space the hyperplane is a plane with two dimensions. It is hard to visualize the hyperplane in three or more-dimensional space but the concept still works [52]. We can note that using the hyperplane for separations forms a linear decision boundary.

If the data is linearly separable, there are an infinite number of hyperplanes that can classify the data. The *maximal margin classifier* finds the **best** hyperplane that makes the margin as big as possible, where the *margin* is the minimal distance between each observation to the hyperplane. In case of two classes, the maximal margin hyperplane represents the midline of the widest street between the two classes, as illustrated in the Figure 2.13. [52].

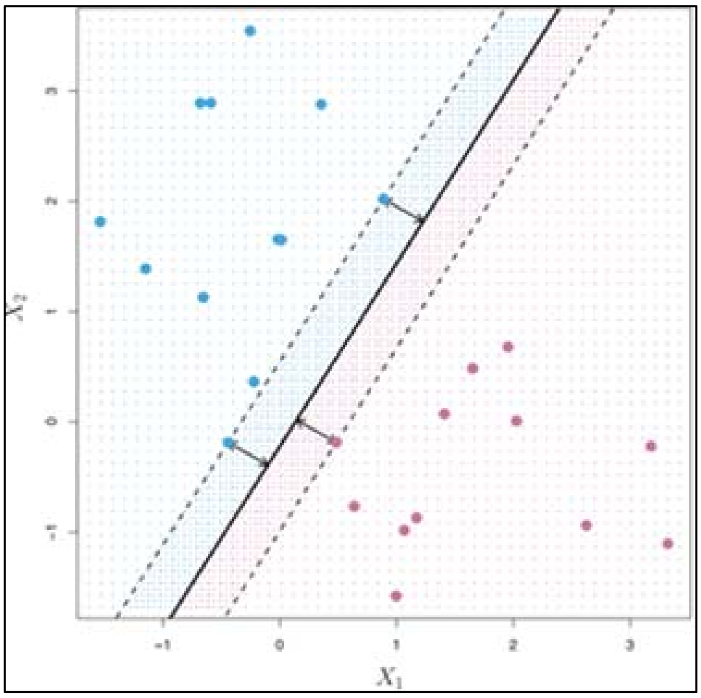

Figure 2.13. The maximal margin hyperplane that separates two classes

In Figure 2.13. the three points that lie on the two dashed lines determine the width of the margin, these points are called support vectors because they support creating maximal margin hyperplane, and they called vectors because the instances are vectors in p dimensional space. The created hyperplane depends only on these instances but not on other observations, where if these other observations change their location, there is no effect on the hyperplane.

The maximal margin hyperplane is a solution to the optimization problem as given by the following equations:

$$maximize\ M_{\beta_0,\beta_1,\dots,\beta_p} \tag{2.46}$$

$$subject\ to \sum_{j=1}^{P} \beta_j^2 = 1, \tag{2.47}$$

$$y_i(\beta + \beta_1 x_{i1} + \beta_2 x i_2 + \cdots + \beta_p x_{ip}) \geq M\ \forall\ i = 1, \dots, n. \tag{2.48}$$

Where M represents the margin of the hyperplane.

β_0, β_1, ... are the coefficient of maximal margin hyperplane, they will be chosen by the optimization algorithm to maximize the margin.

x_{i1}, x_{i1} … represent the features for every instance in the p dimensional space.

In the case that the data does not linearly separable as shown in the Figure 2.14. [52], the more generalized classifier called Support Vector Classifier can be applied. This classifier can develop hyperplane that does not *perfectly* classify the instances but *almost* do the classification. This generalization can occur by allowing observations to be in the wrong position inside the margin or even in the incorrect side of the hyperplane. This separation technique can help the classifier not overfitting the dataset.

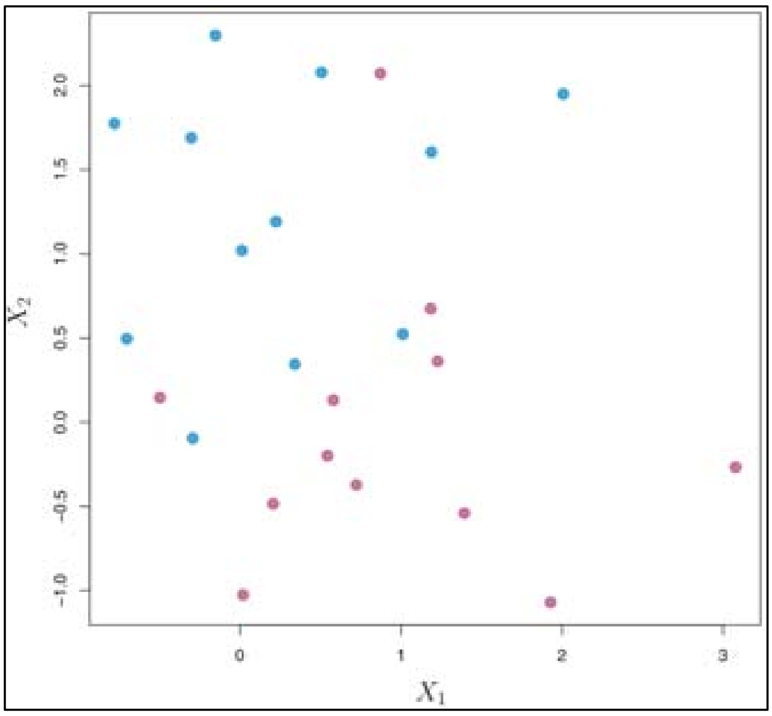

Figure 2.14. Non-linearly separable data points

In the case that the classes have non-linear decision boundary, the Support Vector Classifier cannot classify the dataset correctly as shown in the Figure 2.15. [52], while the Support Vector Regression can do this job more efficiently. In cases like shown, the feature space is enlarged by using something called Kernel Trick. Briefly, kernel trick are functions that take low dimensional feature space and map it to very high dimensional space, so the non-linear problems is converted to a linear problem, after that the support vector regression can classify the data points, and then the solution is backed to the original space ending with non-linear separation algorithm [53]. In the right hand of the Figure 2.16. [52], shows the SVR algorithm with radial kernel applied to non-linear data resulting in an appropriate classifier. It is worth noting that, kernels are not functions with feature space, but they are functions that quantify the similarity of two observations. In this study, SVR algorithm with the Radial Basis Function (RBF) kernel has been used.

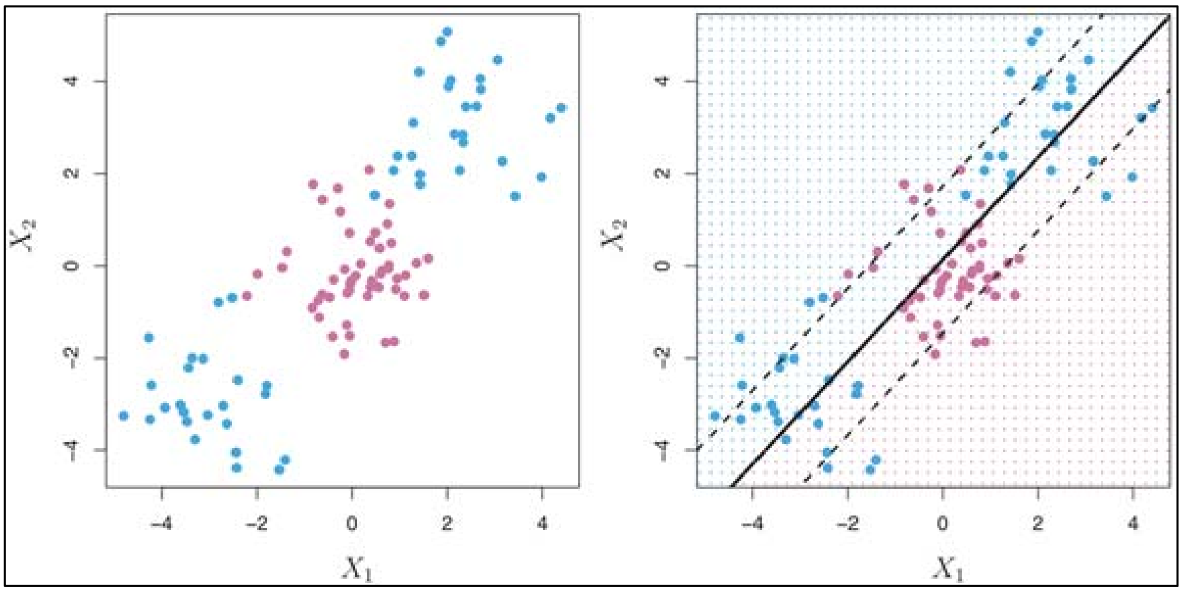

Figure 2.15. Non-linear class boundary.

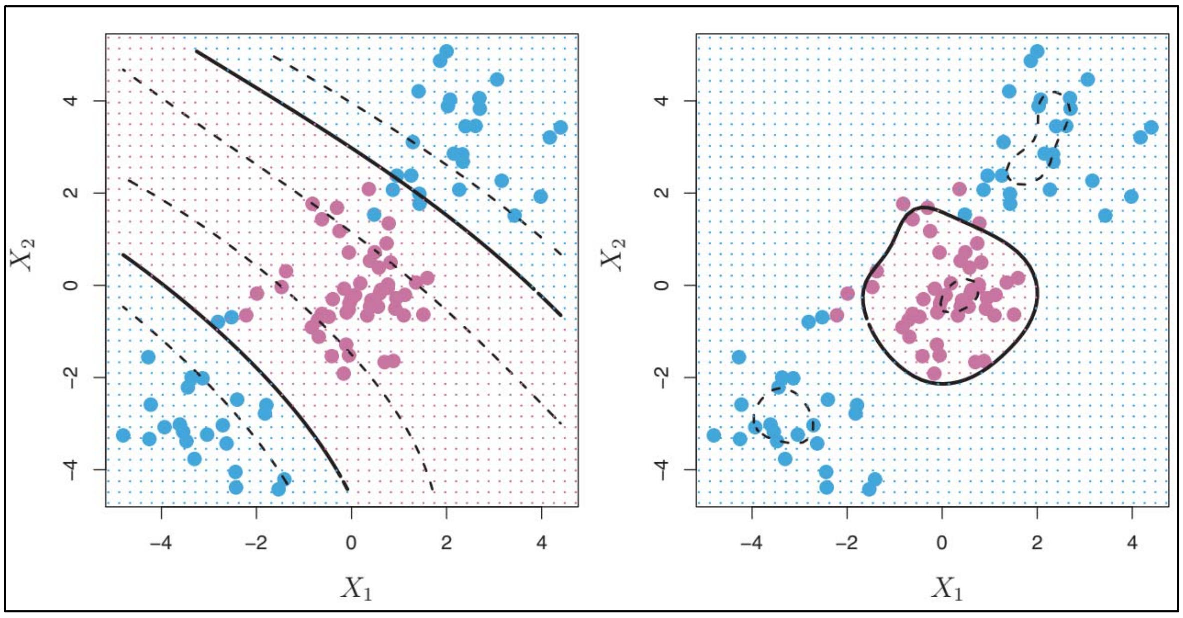

Figure 2.16. Radial kernel applied to non-linear dataset

2.3.5. Multi-Layer Perceptron (MLP)

MLP is a machine-learning algorithm that can be used to generate non-linear complex function from the data. MLP is one kind of feedforward artificial neural networks, where there is no cyclic connection between the formed units [54]. MLP composes at least three layers; the input layer, the output layer and one or more hidden layers, where each layer contains a set of nodes called neurons. All the nodes in every layer are connected with certain weight to every node in the next layer. Figure 2.17. shows MLP with one input layer, two hidden layers and one output layer [55].

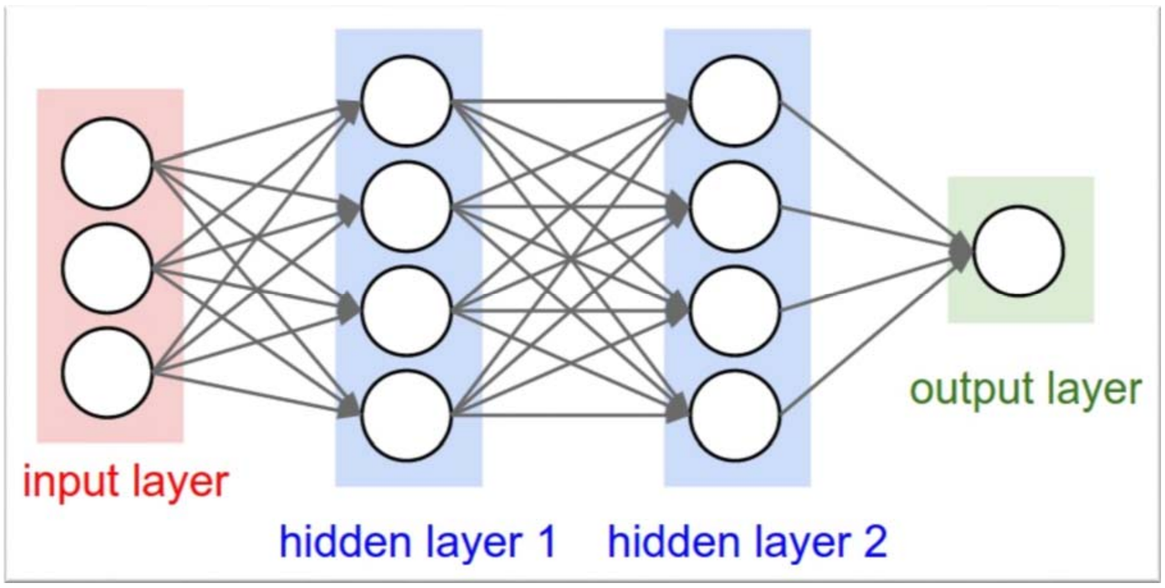

Figure 2.17. Two hidden multi-layer perceptron

The name of multilayer perceptron comes from the single classifier called perceptron (precursor to larger neural networks), where the perceptron consists of single neuron that categorizes the input linearly. The input is a vector multiplied by a certain weight and the bias is added to it as explained by the following equation [56]:

$$y = W * x + b \tag{2.49}$$

To address the issue of only linearly classification limitation in perceptron algorithm and form more complex function the MLP algorithm is used. MLP generally works as the flowing: the input data is vectorized and fed into the first layer, where it multiplied by randomly initialized weights, after that some biases are added, and an activation function is applied to the whole result. The output is passed to the next layer to repeat the same work, wherein each layer -except the first one- the input data comes from the previous layer. After the last layer is reached, the loss function is calculated as shown in the following equation

$$Loss(\hat{y}, y, W) = -y \ln \hat{y} - (1 - y) \ln(1 - \hat{y}) + \alpha \, \|w\|_2^2 \tag{2.50}$$

Where y is the correct output and $\hat{y}$ is the prediction.

$\alpha > 0$ is a non-negative hyperparameter to control the magnitude of the penalty.

The calculated error value is used to compute the partial derivative with respect to weight in each layer going backward recursively, the weights are updated with the calculated values, and the whole process is repeated until the error becomes as small as possible [57].

The neural networks are considered universal function approximators, and the activation functions used in each layer (May except the last one) cause the neural network to build a complex function that is nonlinear [57]. One of the most efficient activation functions exists is the Rectified Linear Units (ReLU), it has a greatly better performance than sigmoid or tanh activation functions, also it produces inexpensive operations that make it so popular nowadays. Krizhevsky et al. [58] showed that using ReLU has improved the convergence of a very big image classification problem 6 times faster compared with tanh unit. The ReLU unit is shown in Figure 2.18. [59]. In addition, it has the following equations:

$$f(x) = Max\ (0, x) \tag{2.51}$$

The equation denotes that the output value is zero when x is negative and linear with slop of one when x is positive.

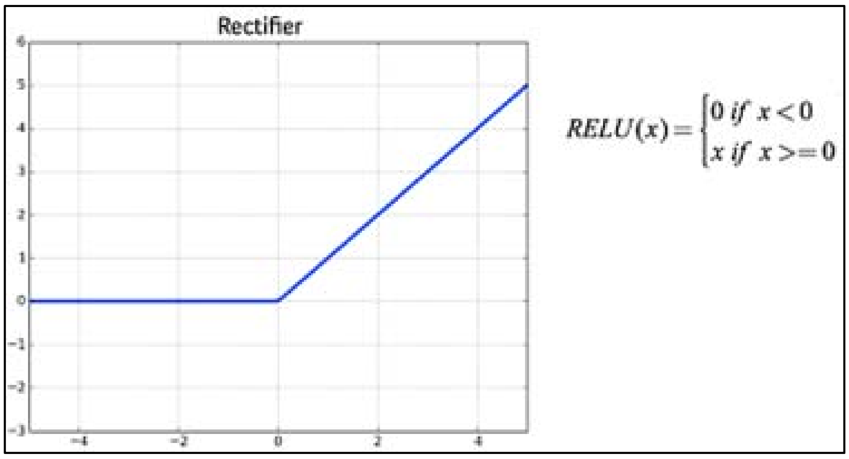

Figure 2.18. ReLU activation function

In this study, ReLU activation function has been chosen , our MLP consists of eleven layers, each layer has a set of neurons given as the followings in order:

50,44,40,35,44,30,64,20,13,12,11. In addition, 0.001 has been chosen as learning rate with maximum 2000 epochs witht the backpropogation algorithm used to calculate the gradients.

2.3.6. Long Short-Term Memory (LSTM)

The prediction mechanism used in feedforward neural network has some limitations making it less efficient than other models in some application. One of that restrictions is the requiring of the input x and output y data to be always with a fixed length, but it turns out that in some applications the input and output data needs be in variable lengths, like the case in machine translation and speech recognition. Another limitation is that the feedforward neural network cannot deal with data having an ordered structure, as the case in time series analysis where the data is ordered over time, also in natural language processing where the ordering of words is very important.

These shown previous problems can be handled using another type of neural network call Recurrent Neural Network (RNN). This model can use each unit to store memories about the previous input making it robust solution for handling sequential data, also it can handle the differences in the input-output lengths as shown the in the Figure 2.19. [60].

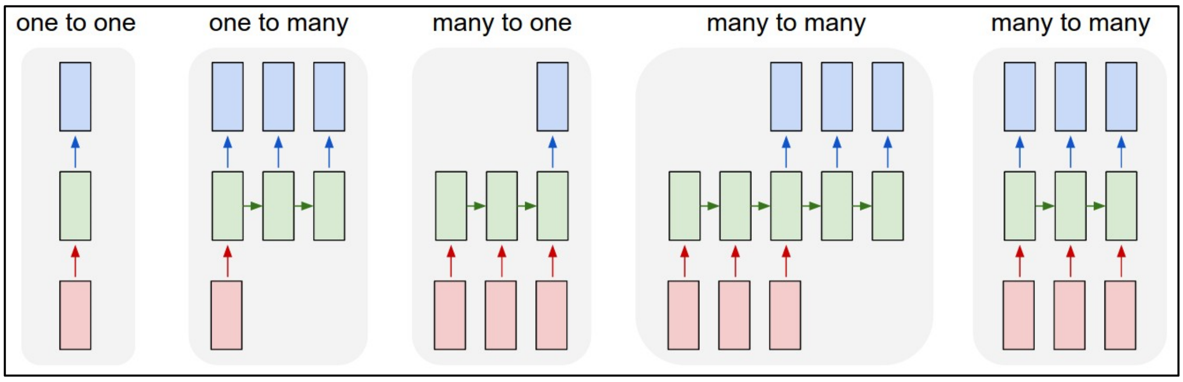

Figure 2.19. Several input-output lengths that RNN can handle

The basic type of RNN, unfortunately, cannot remember long input data backward, making it hard to capture long-range connections because of the vanishing gradient problem. One of the modifications of basic RNN hidden layer called Long Short-Term Memory (LSTM) can learn very large connection in a sequence and do so much better than the basic RNN, almost all the recent improvement in RNN comes by using LSTM models [61].

LSTM network was first introduced by Sepp Hochreiter and Jüurgen Schmidhuber in 1997 [62]. LSTM combines a set of cells that are connected to each other, where each cell consists of a number of gates. Manipulating data across gates makes the network able to learn what to store in the long-term state, what to throw away and what to read from it. This makes LSTM able to capture long-range connection and help a lot addressing vanishing gradient problem. Figure 2.20. [63] shows one LSTM cell with its gates:

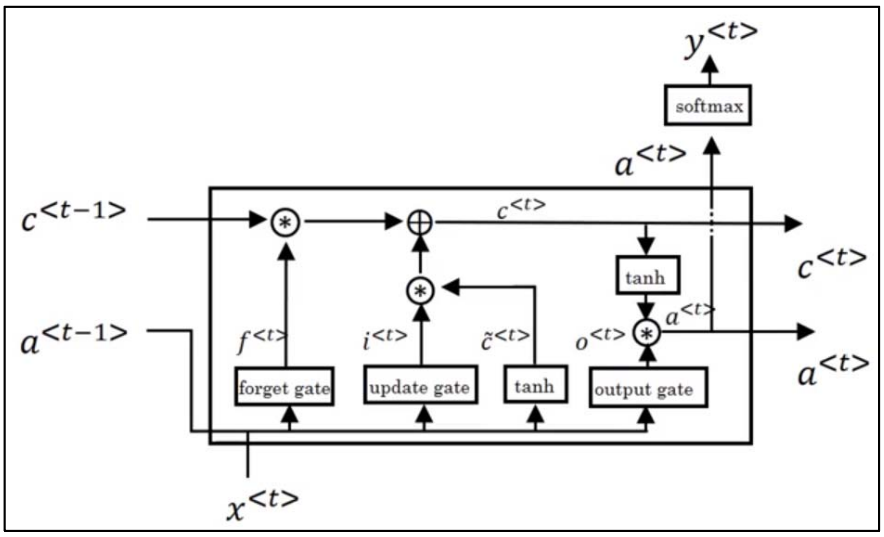

Figure 2.20. LSTM cell with gates illustrated

Each gate and output is given by the following equations:

$$\tilde{C}^{\langle t\rangle} = \tanh(W_C\left[a^{\langle t-1\rangle}, x^{\langle t\rangle}\right] + b_C) \tag{2.52}$$

$$\Gamma_u = \sigma(w_u\left[a^{\langle t-1\rangle}, x^{\langle t\rangle}\right] + b_u) \tag{2.53}$$

$$\Gamma_f = \sigma(w_f\left[a^{\langle t-1\rangle}, x^{\langle t\rangle}\right] + b_f) \tag{2.54}$$

$$\Gamma_o = \sigma(w_o\left[a^{\langle t-1\rangle}, x^{\langle t\rangle}\right] + b_o) \tag{2.55}$$

$$C^{\langle t\rangle} = \Gamma_u * \tilde{C}^{\langle t\rangle} + \Gamma_f * C^{\langle t-1\rangle} \tag{2.56}$$

$$a^{\langle t\rangle} = \Gamma_0 * \tanh c^{\langle t\rangle} \tag{2.57}$$

Where:

$c^{<t>}$ provides a memory for the cell at time t, used for Γ_u long-term reserving.

$c^{\sim<t>}$ which is a term multiplied by update gate to calculate the $c^{<t>}$ term.

Γ_u is the update gate used to calculate the new value of the memory cell $c^{<t>}$.

Γ_f is the forget gate which is responsible for forgetting unimportant data.

Γ_o is the output gate which is multiplied element-wise with $c^{<t>}$ to calculate the next activation function $a^{<t>}$

σ refers to the sigmoid activation function.

The choosing of optimization algorithm has a direct effect on the obtained results and training time. In machine learning, Adam optimization algorithm is one of the modern algorithms that stood up and proves its ability to work well in a wide range of applications. It was first introduced in 2014 by Diederik P. Kingma and Jimmy Ba Sebastian Ruder in their paper "Adam: A Method for Stochastic Optimization" [64]. The name of the algorithms is derived from adaptive moment estimation. .The authors showed that using the Adam algorithm combines the benefits of RMSProp and AdaGrad in handling sparse gradients and non-stationary data. Also, Adam requires little memory and it can be implemented straightforward. Sebastian Ruder [65] did a research for comparison between modern optimization algorithms; he showed that the using of Adam algorithm is a favorite choice over other modern algorithms including RMSprop and Adadelta. In addition, Adam algorithm is suggested to be a default choice for deep learning applications in the Stanford course " Convolutional Neural Networks for Visual Recognition" introduced by Fei-Fei Li et al. [66]

In this study, LSTM algorithm with 50 connected cells has been used, training it over 1500 epoch with the Adam optimization algorithm.

3. RESULTS AND DISCUSSIONS

The data of stock market is classified as non-stationary data, at a particular time there can be random walks, trend, cycles or combination of them, also the BIST100 values are affected by many different internal and external factors, which makes the stock prediction, not an easy task.

In this study, several factors that assumed to have an effect on BIST100 prices are combined together, cleaned, tuned and used in building six different models. Each model is built using one of the six algorithms and compared to other models in doing the regression task. The used algorithms include: decision tree (DT), Random Forest (RF), K-nearest neighbor (KNN), support vector regression (SVR), multi-layer perceptron (MLP) and long short-term memory (LSTM). The Root Mean Square Error (RMSE) measure is used to evaluate the performance of the classifiers, its equation is given as the following:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y} - y)}{n}} \tag{3.58}$$

Where the $\hat{y}$ represents the predicted value, y represents the true one and n is the total number of instances.

Each model is trained using 18 months of data, the test phase is done on an unseen data. To decrease the random sampling bias, the process is repeated 30 times, for each turn, the training data starts after few days from the last turn and the prediction is done for the next days. All models are trained and compared to the same data. The next 1, 2, 5 and 10 days, are predicted for each model, where the 5 days represents one trading week and 10 days represents two trading weeks. The number of predicted days are chosen not so forward in the future according to the ability of technical indicator to handle, as mention before. The random sampling is not used in this study because it cases a look-forward bias to occur in the prediction.

Table 3.1 shows the RMSE values for 1-day daily return prediction, where each model is trained and tested using the High, Low, Close and Volume values according to the "Data from" and "Data To" columns on the table. The daily return movement can be calculated as the following:

$$Daily\ Retrun\ for\ day\ T = \frac{close\ price\ of\ one\ day\ before\ T}{close\ price\ of\ the\ day\ T} * 100 \tag{3.59}$$

For each turn (represented as a row on the table), all the features are extracted and calculated to train the model. The values under the LSTM, SVR, MLP, RF, KNN, and DT show the results of testing the models according to the RMSE performance measurement. The bold row in the data represents the best results for the SVR found.

By investigating the Table 3.1, we can see that when the models are trained from 03/27/2014 to 08/25/2015 and the predicted date is 08/26/2015, four algorithms LSTM, SVR, KNN and DT have the biggest RMSE number, representing the worst expecting day between all 30 models. Going back in the past, it has been found that this period was when the elections in Turkey failed to create a new government and temporal one is appointed to manage the country, and the prediction becomes harder due to the scarcity of like examples in training dataset. In this period Turkey had a blurry political situation and BIST100 had 6 days of continuous dropping (from 76,922.44 TL in 08/17/2015 to 71,341.95 TL) in 24/08/2015 wherein the last day of dropping (08/24/2015) BIST100 loosed 3.33% from its value.

This denotes that the BIST100 and economy in Turkey are affected a lot by the political situations changes, where Turkey is classified as an emerged market not developed one according to the World Bank. Although Turkey is a member of G20 with strong growth from 2012 to 2013, but the Turkish Lira and BIST100 lost value so the relative GDP [3]; which is the total value of the finished services and products produced in a country normally for one year.

Table 3.1. The RMSE values comparison for 1 day

Training Data		Predicted day(s)	LSTM	SVR	MLP	RF	KNN	DT
Data From	Data To							
01/06/2014	06/03/2015	06/04/2015	1,818542	0,64694	1,453879	0,853893	0,595416	1,296467
01/11/2014	06/09/2015	06/10/2015	0,12863	1,290473	1,898405	0,81866	1,690147	1,276761
01/16/2014	06/14/2015	06/15/2015	0,196094	1,477702	1,154204	1,348005	1,672714	1,477363
01/21/2014	06/17/2015	06/18/2015	1,143101	0,408531	7,986257	0,369952	0,779242	0,189214
01/26/2014	06/24/2015	06/25/2015	0,399224	0,070572	0,029177	0,088056	0,032672	0,108837
01/31/2014	06/29/2015	06/30/2015	1,213846	0,910488	1,189644	0,893992	0,41612	0,630457
02/05/2014	07/02/2015	07/03/2015	0,03563	1,258259	1,613946	0,85005	1,648439	1,488116
02/10/2014	07/08/2015	07/09/2015	3,158445	0,460585	0,09501	0,467911	0,738512	0,667937
02/15/2014	07/14/2015	07/15/2015	0,63105	0,119369	0,432988	0,050336	0,465716	0,11977
02/20/2014	07/19/2015	07/20/2015	0,330644	0,169554	2,896264	0,123862	0,483397	0,387232
02/25/2014	07/22/2015	07/23/2015	0,225644	0,315645	0,916833	0,794111	1,422517	0,580214
03/02/2014	07/29/2015	07/30/2015	1,867229	1,901932	4,440253	1,764126	1,709545	1,889633
03/07/2014	08/03/2015	08/04/2015	0,223406	0,324004	0,014731	0,309824	0,325985	0,259095
03/12/2014	08/06/2015	08/07/2015	1,487421	1,383143	1,207781	1,339649	1,229619	1,316217
03/17/2014	08/12/2015	08/13/2015	0,250755	0,018909	0,031878	0,315525	0,542615	0,110567
03/22/2014	08/18/2015	08/19/2015	0,666472	0,901764	2,775402	1,273488	1,053053	0,761347
03/27/2014	08/23/2015	08/24/2015	6,084645	2,968468	5,03447	1,703453	4,232327	3,178135
04/01/2014	08/26/2015	08/27/2015	0,738261	0,507529	4,843253	0,303255	0,070946	0,158735
04/06/2014	09/02/2015	09/03/2015	0,086005	1,658507	0,139371	1,6505	1,518863	1,319527
04/11/2014	09/07/2015	09/08/2015	0,538883	0,221664	0,0064	0,278517	0,149397	0,164434
04/16/2014	09/10/2015	09/11/2015	4,242543	0,098046	1,799325	0,1269	0,216239	0,170732
04/21/2014	09/16/2015	09/17/2015	0,763236	0,087036	0,316736	0,408402	0,518804	0,126591
04/26/2014	09/21/2015	09/22/2015	0,737137	0,713138	0,06437	0,536466	0,688038	0,764738
05/01/2014	09/27/2015	09/28/2015	1,682881	1,327574	1,113422	1,139001	1,462236	1,03788
05/06/2014	09/30/2015	10/01/2015	0,790133	0,211884	0,778796	0,252665	0,109143	0,451158
05/11/2014	10/07/2015	10/08/2015	1,108484	0,521421	3,266049	0,576811	0,889345	0,346272
05/16/2014	10/12/2015	10/13/2015	2,974711	1,173833	0,515159	1,182918	0,986529	0,948898
05/21/2014	10/15/2015	10/16/2015	0,292975	1,611755	2,009623	1,550316	1,62776	1,247806
05/26/2014	10/21/2015	10/22/2015	0,41504	0,291258	0,868805	0,358164	0,177539	0,653259
05/31/2014	10/26/2015	10/27/2015	1,602934	0,029328	0,251318	0,042338	0,764455	0,191762
Average RMSE			1,194467	0,76931	1,638125	0,725705	0,940578	0,777305

The following Table 3.2. Table3.3 and Table 3.4. shows the same information of the table 3.1. but for two, five and ten trading days. The bold row in the data represents the best results for the SVR found.

Table 3.2. The RMSE values comparison for 2 days

Training Data		Predicted day(s)	LSTM	SVR	MLP	RF	KNN	DT
Data From	Data To							
01/06/2014	06/01/2015	06/02/2015 - 06/03/2015	4,363354	1,001278	1,814415	0,85361	0,910733	1,163592
01/11/2014	06/07/2015	06/08/2015 - 06/09/2015	0,626999	1,668991	0,426885	1,279421	2,323453	1,21971
01/16/2014	06/10/2015	06/11/2015 - 06/12/2015	2,363801	1,802366	5,570134	1,77196	2,015493	1,94772
01/21/2014	06/15/2015	06/16/2015 - 06/17/2015	1,134603	0,914021	1,617897	0,937585	1,883924	0,628819
01/26/2014	06/21/2015	06/22/2015 - 06/23/2015	1,430945	1,0216	1,926932	0,84906	0,849747	0,882725
01/31/2014	06/25/2015	06/26/2015 - 06/29/2015	0,917861	0,68128	1,422518	1,045681	0,751851	0,688557
02/05/2014	06/30/2015	07/01/2015 - 07/02/2015	1,181642	1,097489	1,407141	1,135413	1,009136	1,118115
02/10/2014	07/06/2015	07/07/2015 - 07/08/2015	2,156088	1,672385	4,682074	1,519861	2,197558	1,384544
02/15/2014	07/12/2015	07/13/2015 - 07/14/2015	1,00335	0,508751	0,656031	0,435462	0,350736	0,406479
02/20/2014	07/14/2015	07/15/2015 - 07/16/2015	1,046675	1,385355	0,995565	1,300681	1,438499	1,204481
02/25/2014	07/20/2015	07/21/2015 - 07/22/2015	2,318001	2,6382	2,558906	2,669947	2,716067	2,775216
03/02/2014	07/27/2015	07/28/2015 - 07/29/2015	0,37809	1,340834	0,565873	1,309826	1,597945	1,216233
03/07/2014	07/30/2015	07/31/2015 - 08/03/2015	1,760908	0,247317	2,111725	0,399057	0,198583	0,237129
03/12/2014	08/04/2015	08/05/2015 - 08/06/2015	2,038182	1,084963	2,514089	1,211208	1,056226	0,763247
03/17/2014	08/10/2015	08/11/2015 - 08/12/2015	1,670109	1,178614	3,006648	1,100806	1,531556	1,268742
03/22/2014	08/16/2015	08/17/2015 - 08/18/2015	0,516504	0,98853	1,015808	1,232889	0,670285	1,268054
03/27/2014	08/19/2015	08/20/2015 - 08/21/2015	2,636367	3,212225	2,958205	3,149972	3,321675	3,206994
04/01/2014	08/24/2015	08/25/2015 - 08/26/2015	5,707865	1,085078	3,022331	0,928274	1,518391	1,017502
04/06/2014	08/31/2015	09/01/2015 - 09/02/2015	2,551715	1,158932	1,328164	1,23939	0,950458	1,217165
04/11/2014	09/03/2015	09/04/2015 - 09/07/2015	2,821651	0,159063	1,337694	0,450749	0,706721	0,203429
04/16/2014	09/08/2015	09/09/2015 - 09/10/2015	0,101861	0,479004	1,113082	0,558084	0,310319	0,664654
04/21/2014	09/14/2015	09/15/2015 - 09/16/2015	0,859286	0,774712	2,085081	0,698688	1,325669	0,622102
04/26/2014	09/17/2015	09/18/2015 - 09/21/2015	1,70644	1,183159	2,654012	1,449135	0,808034	1,118981
05/01/2014	09/21/2015	09/22/2015 - 09/23/2015	1,581389	1,551122	1,863982	1,54469	1,513167	1,581677
05/06/2014	09/28/2015	09/29/2015 - 09/30/2015	0,445384	0,325198	0,703613	0,312978	0,395196	0,337537
05/11/2014	10/05/2015	10/06/2015 - 10/07/2015	1,269725	0,389025	1,80804	0,556588	0,605153	0,375969
05/16/2014	10/08/2015	10/09/2015 - 10/12/2015	1,253663	1,258101	1,568846	1,24812	1,245523	1,272704

Table 3.2. (devam) The RMSE values comparison for 2 days

05/21/2014	10/13/2015	10/14/2015 - 10/15/2015	1,458817	1,32268	1,758963	1,311058	1,301731	1,288834
05/26/2014	10/19/2015	10/20/2015 - 10/21/2015	0,476214	0,421553	2,326263	0,410476	0,475969	0,40207
05/31/2014	10/22/2015	10/23/2015 - 10/26/2015	0,492604	0,542521	0,432656	0,542012	0,687275	0,836523
Average RMSE			1,609003	1,103145	1,908452	1,115089	1,222236	1,077317

In predicting the next 2 days: SVR, RF, KNN and DT have the biggest RMSE (meaning the worst results) when they are trained using the prices between 03/27/2014 and 08/25/2015. That can be interrupted for the same reason(election one) mentioned in the prediction of the next day previously.

Table 3.3. The RMSE values comparison for 5 days

Training Data		Predicted day(s)	LSTM	SVR	MLP	RF	KNN	DT
Data From	Data To							
01/06/2014	05/24/2015	05/25/2015 To 05/29/2015	1,928654	1,919873	2,410607	1,907187	2,009219	1,915347
01/11/2014	05/28/2015	05/29/2015 To 06/04/2015	3,268606	2,579684	2,926903	2,645272	2,502098	2,531427
01/16/2014	06/02/2015	06/03/2015 To 06/09/2015	1,180911	1,576062	2,450369	1,937956	1,658636	1,627789
01/21/2014	06/07/2015	06/08/2015 To 06/12/2015	1,647193	1,32248	4,025459	1,506123	1,178517	1,402602
01/26/2014	06/14/2015	06/15/2015 To 06/19/2015	1,512656	0,903274	1,234034	0,891994	1,141767	0,987295
01/31/2014	06/17/2015	06/18/2015 To 06/24/2015	2,315179	1,373862	1,719194	1,345522	1,22012	1,454716
02/05/2014	06/22/2015	06/23/2015 To 06/29/2015	0,883912	0,804132	0,81082	0,845976	0,618276	0,928428
02/10/2014	06/28/2015	06/29/2015 To 07/03/2015	2,358038	1,429345	1,875402	1,461252	1,568739	1,569442
02/15/2014	07/02/2015	07/03/2015 To 07/09/2015	2,238121	0,492442	3,064729	0,477179	0,490186	0,570174
02/20/2014	07/06/2015	07/07/2015 To 07/13/2015	1,249137	0,997894	2,074271	1,087539	1,190883	0,931367
02/25/2014	07/09/2015	07/10/2015 To 07/16/2015	1,666613	1,884621	2,63881	1,968798	2,096139	2,061645
03/02/2014	07/19/2015	07/20/2015 To 07/24/2015	2,29376	1,203558	2,581702	1,248155	1,391222	1,243494
03/07/2014	07/22/2015	07/23/2015 To 07/29/2015	1,70282	1,408443	2,127897	1,504744	1,289833	1,364795
03/12/2014	07/27/2015	07/28/2015 To 08/03/2015	2,07963	0,872313	1,665175	0,627444	0,781433	0,674971
03/17/2014	08/02/2015	08/03/2015 To 08/07/2015	1,76397	1,826081	1,854667	1,770152	1,763907	1,771179
03/22/2014	08/06/2015	08/07/2015 To 08/13/2015	1,358083	0,991976	3,699964	0,722148	0,888664	0,892024
03/27/2014	08/11/2015	08/12/2015 To 08/18/2015	2,383055	2,23657	2,074084	2,119954	1,937809	2,21377
04/01/2014	08/16/2015	08/17/2015 To 08/21/2015	5,214975	2,129317	4,072869	2,094013	2,0214	2,136869
04/06/2014	08/23/2015	08/24/2015 To 08/28/2015	1,578	1,393144	5,853811	1,432728	1,216862	1,432452
04/11/2014	08/26/2015	08/27/2015 To 09/02/2015	1,480692	1,097969	1,902655	0,913573	0,703061	1,175556
04/16/2014	08/31/2015	09/01/2015 To 09/07/2015	0,510037	0,462459	1,141543	0,389269	0,475598	0,59174
04/21/2014	09/06/2015	09/07/2015 To 09/11/2015	4,394197	1,447922	2,831974	1,44879	1,83248	1,586036
04/26/2014	09/09/2015	09/10/2015 To 09/16/2015	1,440155	0,87766	1,766351	0,987538	0,944836	0,889514
05/01/2014	09/13/2015	09/14/2015 To 09/18/2015	2,478166	1,22322	1,259978	1,505021	1,314319	1,308066
05/06/2014	09/16/2015	09/17/2015 To 09/23/2015	2,381878	0,987287	1,700584	0,988892	1,173843	1,04484
05/11/2014	09/27/2015	09/28/2015 To 10/02/2015	3,040811	1,776769	1,099582	1,641937	2,021771	1,767143
05/16/2014	09/30/2015	10/01/2015 To 10/07/2015	1,368525	0,841926	3,320575	0,961634	1,076917	0,853651
05/21/2014	10/05/2015	10/06/2015 To 10/12/2015	1,287218	1,162918	1,160415	1,121109	1,131345	1.16365
05/26/2014	10/11/2015	10/12/2015 To 10/16/2015	2,583863	1,002445	2,749935	0,985358	1,03181	0.8832
05/31/2014	10/14/2015	10/15/2015 To 10/21/2015	0,590323	0,621091	0,614231	0,673825	0,816976	0.933254
Average RMSE			2,005973	1,294891	2,290286	1,307036	1,316289	1,330215

From the results of Table 3.3, it can be shown that the best result of the LSTM, SVR, RF, KNN and MLP when the data is trained between the 4/16/2014 and 9/14/2015. The reason for that was investigated and the news in that period of time is reviewed, but the inherent reason wasn't found, so founding the reason why the algorithm is not doing well prediction is easier than when the algorithm is not doing that. Note that the bold row in the data represents the best results for the SVR found.

Table 3.4. The RMSE values comparison for 10 days

Training Data		Predicted day(s)	LSTM	SVR	MLP	RF	KNN	DT
Data From	Data To							
01/06/2014	05/07/2015	05/08/2015 To 05/22/2015	2,131908	1,590917	1,682576	1,63763	1,787907	1,580633
01/11/2014	05/13/2015	05/14/2015 To 05/28/2015	2,514177	2,251837	3,221002	2,320566	2,345122	2,269626
01/16/2014	05/19/2015	05/20/2015 To 06/02/2015	2,527934	2,139584	2,230882	2,114869	2,186683	2,144163
01/21/2014	05/24/2015	05/25/2015 To 06/05/2015	2,941419	2,012663	1,615049	1,962441	2,133565	2,047978
01/26/2014	05/31/2015	06/01/2015 To 06/12/2015	1,54915	1,134944	1,06117	1,115878	1,061326	1,252335
01/31/2014	06/03/2015	06/04/2015 To 06/17/2015	1,128145	1,15253	3,78772	1,156009	1,183673	1,110655
02/05/2014	06/08/2015	06/09/2015 To 06/22/2015	2,11497	1,241466	2,813562	1,130489	1,189777	1,200214
02/10/2014	06/14/2015	06/15/2015 To 06/26/2015	1,876932	1,387628	2,31924	1,380025	1,403997	1,500147
02/15/2014	06/18/2015	06/19/2015 To 07/02/2015	1,470629	1,122392	1,627831	1,19712	1,170088	1,181077
02/20/2014	06/22/2015	06/23/2015 To 07/06/2015	1,443363	1,170471	1,500591	1,13028	1,148252	1,386449
02/25/2014	06/25/2015	06/26/2015 To 07/09/2015	1,07213	1,388352	2,025155	1,367326	1,394768	1,496952
03/02/2014	07/02/2015	07/03/2015 To 07/16/2015	1,279032	1,620195	1,91684	1,555227	1,523064	1,719275
03/07/2014	07/07/2015	07/08/2015 To 07/22/2015	2,083506	1,636911	1,1603	1,661027	1,447499	1,71375
03/12/2014	07/12/2015	07/13/2015 To 07/27/2015	2,084928	1,14122	1,457207	1,101601	1,075082	1,213198
03/17/2014	07/19/2015	07/20/2015 To 07/31/2015	2,511962	1,520689	1,729222	1,487564	1,398804	1,429177
03/22/2014	07/23/2015	07/24/2015 To 08/06/2015	5,004161	1,501947	2,022435	1,346449	1,196871	1,278693
03/27/2014	07/28/2015	07/29/2015 To 08/11/2015	3,514259	1,877741	3,226803	1,997029	1,617464	1,679952
04/01/2014	08/02/2015	08/03/2015 To 08/14/2015	2,599494	1,703538	2,649135	1,657274	1,549588	1,548894
04/06/2014	08/09/2015	08/10/2015 To 08/21/2015	3,985942	1,748983	1,330438	1,719666	1,759564	1,793159
04/11/2014	08/12/2015	08/13/2015 To 08/26/2015	7,279741	1,176576	1,681467	1,073659	1,523528	1,12425
04/16/2014	08/17/2015	08/18/2015 To 08/31/2015	4,401722	1,11857	2,042874	0,960287	0,956435	1,176111
04/21/2014	08/23/2015	08/24/2015 To 09/04/2015	3,859635	1,086748	3,037337	1,09604	1,06407	1,297722
04/26/2014	08/26/2015	08/27/2015 To 09/09/2015	1,997665	1,109707	1,15813	1,041965	1,183665	1,33066
05/01/2014	08/30/2015	08/31/2015 To 09/11/2015	2,380616	1,310955	1,702887	1,416327	1,464621	1,49149
05/06/2014	09/02/2015	09/03/2015 To 09/16/2015	3,467568	0,951386	1,597336	0,976665	1,036145	1,013708
05/11/2014	09/09/2015	09/10/2015 To 09/23/2015	1,490908	1,392841	2,089075	1,390075	1,36603	1,302737
05/16/2014	09/14/2015	09/15/2015 To 09/30/2015	2,414975	1,360354	1,145502	1,594018	1,285772	1,232534
05/21/2014	09/17/2015	09/18/2015 To 10/05/2015	3,785573	1,014704	1,208372	1,253827	1,107942	0,974935
05/26/2014	09/27/2015	09/28/2015 To 10/09/2015	1,465098	0,926686	1,607287	0,995532	1,292862	0,97951
05/31/2014	09/30/2015	10/01/2015 To 10/14/2015	2,165435	0,896875	1,046725	0,798452	1,367056	0,99674
Average RMSE			2,618099	1,389647	1,923138	1,387844	1,407374	1,415557

From Table 3.4, it can be shown that the algorithms SVR, RF, KNN and DT have the lowest RMSE when the data is trained between the 4/16/2014 and 9/14/2015 . Note that the bold row in the data represents the best results for the SVR found.

Table 3.5 shows a comparison between all models according to the RMSE measure. From this table, it can be shown that there is no model that can outperform the others in all cases, but the SVR model gives the most stable performance in most of the times. The "1 Day" column denotes the comparison between the models for one trading day, the same for the 2, 5 and 10 days columns. For each column, the models are sorted according to its performance from top to down, where the best performance models are on the upper side.

Table 3.5. Models comparison

1 Day	2 Days	5 Days	10 Days
RF	DT	SVR	RF
SVR	SVR	RF	SVR
DT	RF	KNN	KNN
KNN	KNN	DT	DT
MLP	LSTM	LSTM	MLP
LSTM	MLP	MLP	LSTM

Table 3.6 shows six test cases for a one-day daily-return prediction for each model, the cases are chosen according to the best-predicted cases for the SVR model, and each case is compared to other models. Every one row is bounded by the bold box to indicate that we have one shot- prediction. "Predicted Days" column represents the day predicted. The "True Value" column represents the actual value of the daily return for that day, and the other columns represent the predicted daily-return values for each model.

Table 3.6. Sample results for 1 day prediction

Predicated Days	True Value	LSTM	SVR	MLP	RF	KNN	DT
2015-08-13	0,289873	0,039118	0,308782	0,321751	-0,025652	-0,252742	0,40044
2015-10-27	-0,160458	1,442477	-0,13113	-0,411775	-0,202796	-0,924912	-0,352219
2015-06-25	0,322982	-0,076242	0,25241	0,352159	0,234926	0,355654	0,431819
2015-09-17	0,158001	-0,605235	0,245037	-0,158735	0,566403	-0,360803	0,284592
2015-09-11	-0,072094	4,314638	-0,17014	1,727231	-0,198995	-0,288334	-0,242826
2015-07-15	0,074848	-0,556202	0,194217	-0,35814	0,125184	-0,390868	-0,044922

Figure 3.1. shows a visual representation for one of the DT constructed models and how the prediction process is made. DT is considered a white-box algorithm, where the prediction reason can be interrupted, in contrast with algorithms like LSTM and MLP where they are considered black-box algorithm where the behavior of the algorithms is hard to analyze.

In Figure 3.1. X8 represents the MACD feature where it is used by the algorithm firstly because it gives the purest dividing, the X2 represents the Bollinger lower band, X7 represents the ADX winddow plus and X11 represenrs the stochastic K line's value. The MSE represents the Mean Square Error, it is doesn't used in building the models but the tool used SK-learn generates it, samples represents the number of instance in each node before the branching, and value represents the value of the Gini impurity function(it is preferred to divide where the Gini value is as less as possible and that expresses perfect . The value of a selected feature is used as a threshold for dividing, when the value is less than the threshold, the instance will go to the left side and vice versa. It is worth to note that the trees built by CART algorithm are always binary trees.

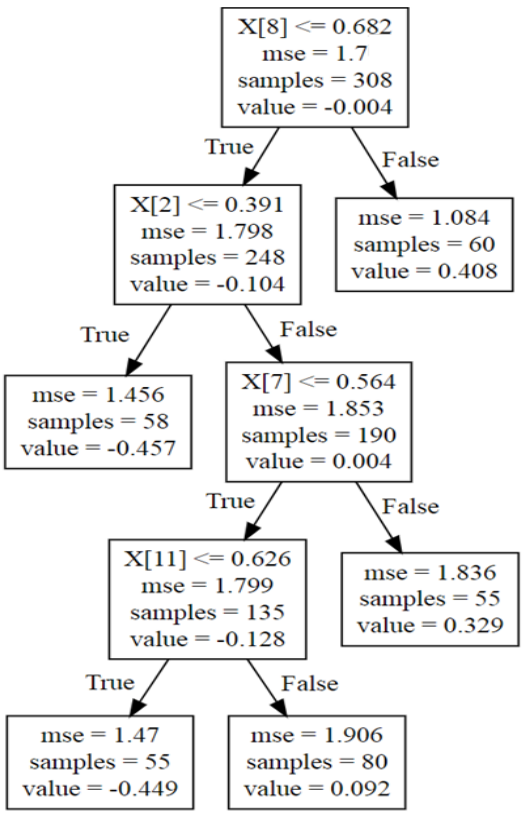

Figure 3.1. DT used to predict the daily return of 01-10-2015 day

Table 3.7 is similar to the Table 3.6, but the prediction is done for 2 days in the future as one shot, the table contains three different cases, where each case is represented by two bold rows. Every two rows are bounded by the bold box to indicate that we have one shot-prediction.

Table 3.7. Sample results for 2 days prediction

Predicated Days	True Value	LSTM	SVR	MLP	RF	KNN	DT
2015-09-04	0,258774	-1,746106	0,046744	-1,615998	0,335418	-0,623351	0,217467
2015-09-07	-0,067243	-3,517444	0,007895	-0,320390	0,565587	-0,537099	0,217467
2015-07-31	0,09054	2,396013	0,250688	1,890662	0,46542	0,370443	-0,229305
2015-08-03	-0,12851	0,812972	0,182430	2,254414	0,29334	-0,105604	-0,229305
2015-09-29	0,438633	-0,191218	0,030520	-0,555308	0,082158	-0,081853	0,264084
2015-09-30	-0,180207	-0,184799	0,031809	-0,133055	0,082158	0,023394	0,264084

Figure 3.2., 3.3. and 3.4. represents the test cases shown in the Tables 3.7, 3.8 and 3.9 respectively, where the red line represents the predicted value of BIST100 daily return for a certain period and the blue one represents the true value.

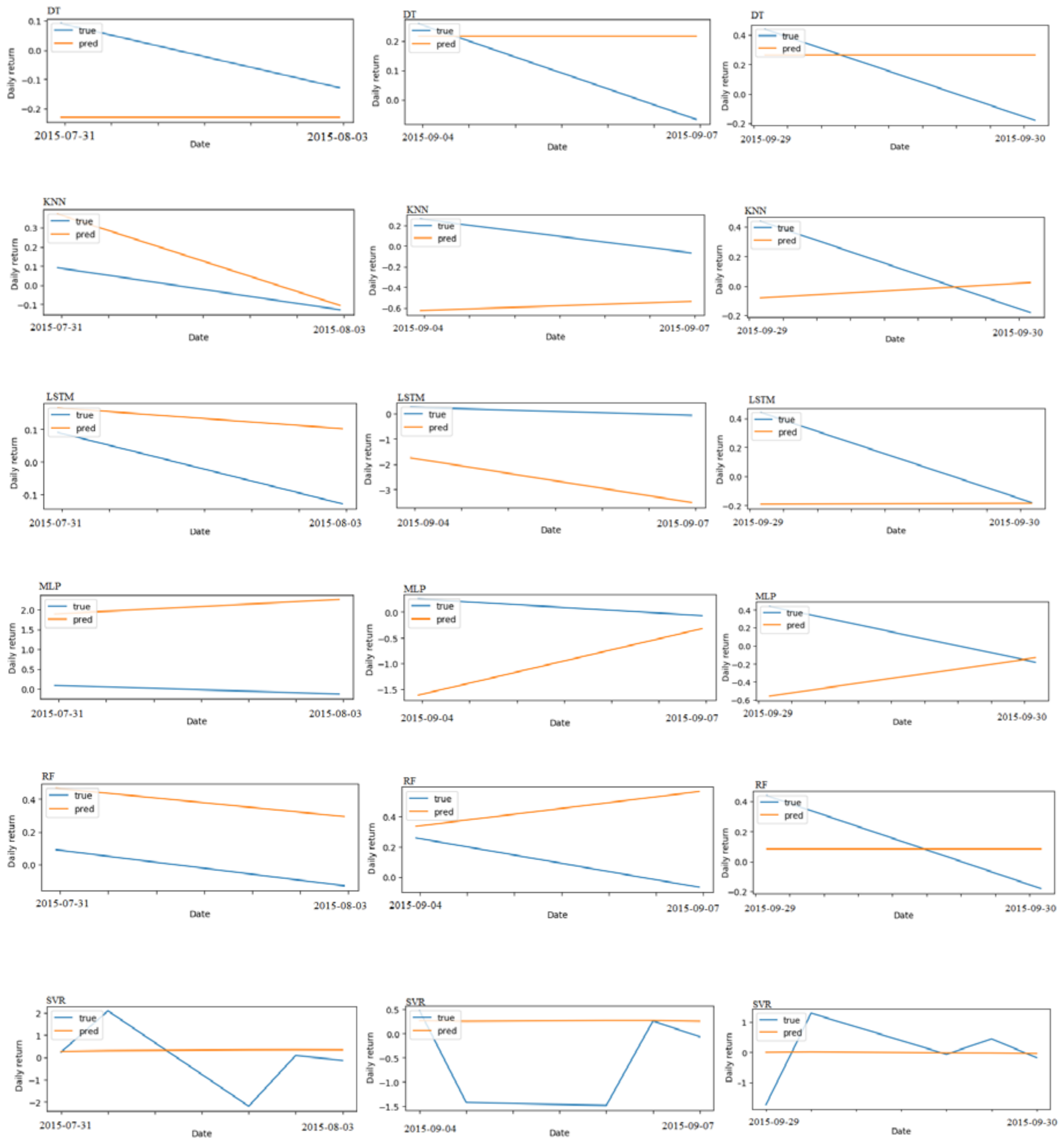

Figure 3.2. Sample chart results for 2 days prediction

Table 3.8 is similar to Table 3.6, but the prediction is done for 5 days in the future as one shot. Every five rows are bounded by bold box to indicate that we have one shot-prediction.

Table 3.8. Sample results for 5 days prediction

Predicated Days	True Value	LSTM	SVR	MLP	RF	KNN	DT
2015-09-01	0,258774	0,836602	0,223606	-0,962839	-0,171178	-0,479221	0,347329
2015-09-02	-0,067243	-0,152079	0,210758	-1,013467	-0,171178	-0,479221	0,347329
2015-09-03	-0,217795	-0,965748	0,197040	-1,446796	-0,171178	-0,479221	0,347329
2015-09-04	-0,689873	-0,535645	0,179404	-1,205806	0,015177	-0,479221	0,347329
2015-09-07	-0,072094	-0,685598	0,179078	1,461300	0,178348	-0,623351	0,347329
2015-07-03	0,614330	1,029017	0,196086	-0,108786	0,049263	0,385921	-0,051445
2015-07-06	0,720445	0,935523	0,169653	1,358269	0,104600	0,375241	-0,051445
2015-07-07	-0,416353	-1,420197	0,162067	-3,486192	0,104600	0,362399	0,142234
2015-07-08	-0,377247	-3,358350	0,241812	-5,885877	0,032211	0,238969	0,142234
2015-07-09	0,074848	-0,025632	0,202888	-2,427990	0,104600	0,284361	0,142234
2015-10-15	0,615092	-0,215228	-0,041293	0,927252	0,196297	0,302318	0,422108
2015-10-16	-0,163315	-0,159375	-0,025136	1,117152	0,080960	-0,112853	0,422108
2015-10-19	-1,076545	-0,849076	-0,058527	-0,723295	-0,041188	0,302318	0,422108
2015-10-20	-0,756665	-1,318006	-0,097459	-0,905021	0,196297	0,302318	0,422108
2015-10-21	-0,160458	-0,764435	-0,071472	-0,109805	0,074148	0,302318	0,422108

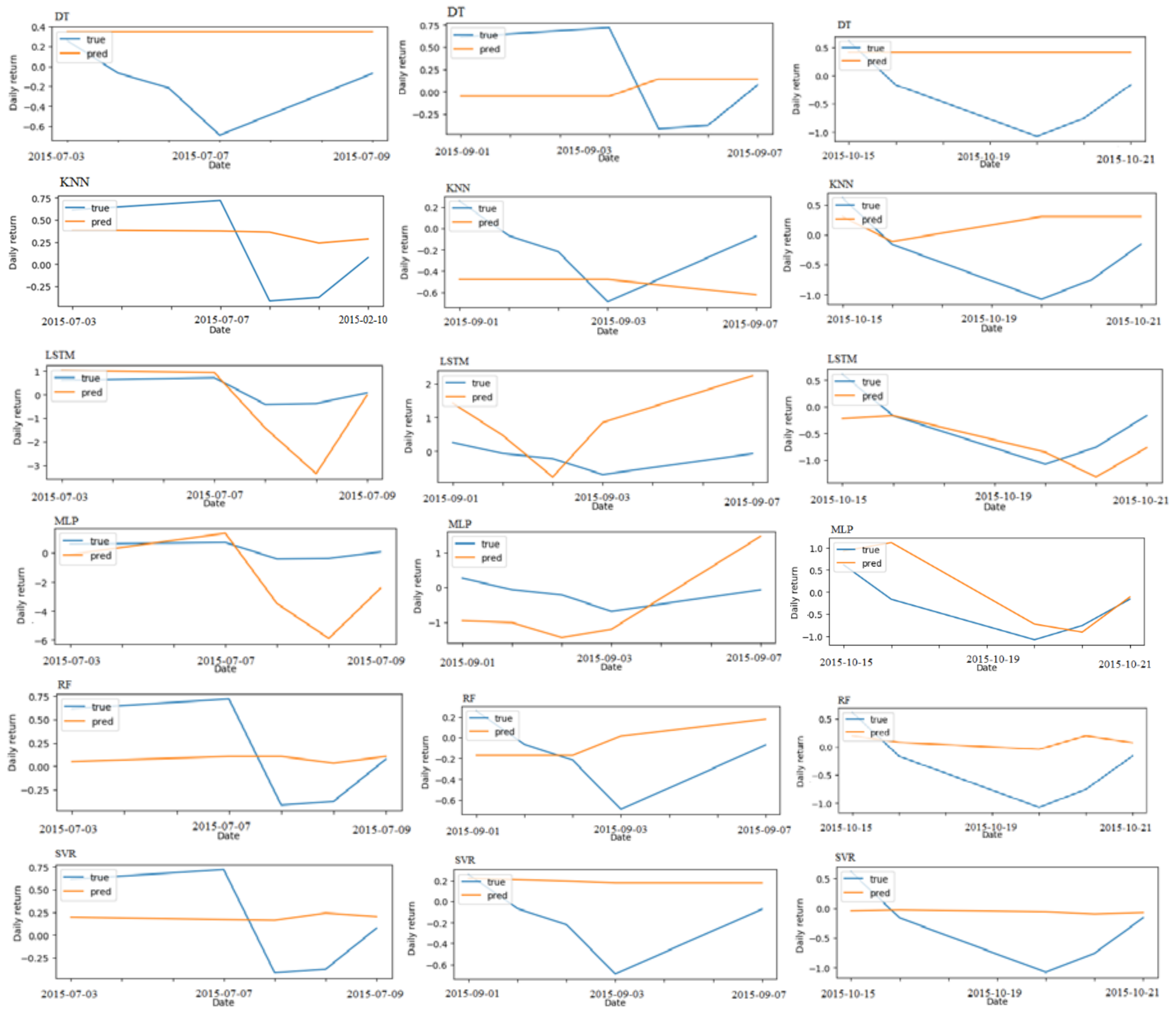

Figure 3.3. Sample chart results for 5 days prediction

Table 3.9 is similar to Table 3.6 but the prediction is done for 10 days in the future as one shot. Every ten rows are bounded by bold box to indicate that we have one shot-prediction.

Table 3.9. Sample results for 10 days prediction

Predicated Days	True Value	LSTM	SVR	MLP	RF	KNN	DT
2015-10-01	-0,147367	0,606576	0,205963	0,364545	0,137587	0,457392	0,407781
2015-10-02	-0,983697	0,481431	0,191920	0,218669	0,137587	0,457392	0,407781
2015-10-05	1,593952	1,738706	0,267675	1,375442	0,369074	0,516639	0,407781
2015-10-06	0,955596	1,878134	0,245035	1,123558	0,151600	0,516639	0,407781
2015-10-07	-0,908363	2,231241	0,235002	0,873441	-0,043208	1,275892	0,407781
2015-10-08	0,615092	2,328520	0,206464	0,546998	-0,157798	1,275892	0,407781
2015-10-09	-0,163315	2,357873	0,207338	0,604226	-0,128238	1,275892	0,407781
2015-10-12	-1,076545	2,451389	0,209159	0,715829	-0,128238	1,275892	0,407781
2015-10-13	-0,756665	1,776579	0,179234	0,363946	0,037010	0,516639	0,407781
2015-10-14	-0,160458	2,139772	0,203456	0,806851	-0,043208	0,516639	0,407781
2015-09-28	0,107505	1,164675	0,160451	-0,79639	0,544959	0,900754	0,425312
2015-09-29	-1,187237	1,184447	0,222896	1,328088	0,468423	1,02856	0,425312
2015-09-30	1,303788	1,067745	0,209686	0,862313	0,468423	1,02856	0,425312
2015-10-01	-0,147367	1,253506	0,21274	1,501424	0,468423	1,02856	0,425312
2015-10-02	-0,983697	1,041218	0,200384	1,455267	0,468423	1,02856	0,425312
2015-10-05	1,593952	1,42591	0,266897	1,485635	0,12593	1,02856	0,425312
2015-10-06	0,955596	1,423704	0,243455	1,411966	0,228096	1,02856	0,425312
2015-10-07	-0,908363	1,35509	0,23023	1,495954	0,093873	1,02856	0,425312
2015-10-08	0,615092	1,431791	0,201995	-0,39755	0,468423	1,02856	0,425312
2015-10-09	-0,163315	1,438751	0,200196	1,510505	0,123478	1,02856	0,425312
2015-09-03	1,120844	1,588208	0,166611	1,089413	0,311512	0,485867	0,427042
2015-09-03	0,158001	3,425909	0,138	1,600336	0,311512	0,485867	0,427042
2015-09-07	1,014221	4,022813	0,122449	1,657307	0,311512	0,541012	0,427042
2015-09-08	-1,14287	4,276845	0,149091	0,92441	0,311512	0,485867	0,427042
2015-09-09	-0,51139	4,520324	0,123682	1,567528	0,311512	0,485867	0,427042
2015-09-10	-1,73597	4,497237	0,150841	1,415342	0,311512	0,485867	0,427042
2015-09-11	1,285767	2,445825	0,064889	1,092938	0,234009	0,339932	0,427042
2015-09-14	-0,07027	0,581337	0,073121	1,403978	0,234009	0,207187	0,427042
2015-09-15	0,438633	2,638153	0,150029	0,448724	0,426518	0,485867	0,427042
2015-09-16	-0,18021	-0,55234	0,163606	-1,69115	0,247413	0,485867	0,427042

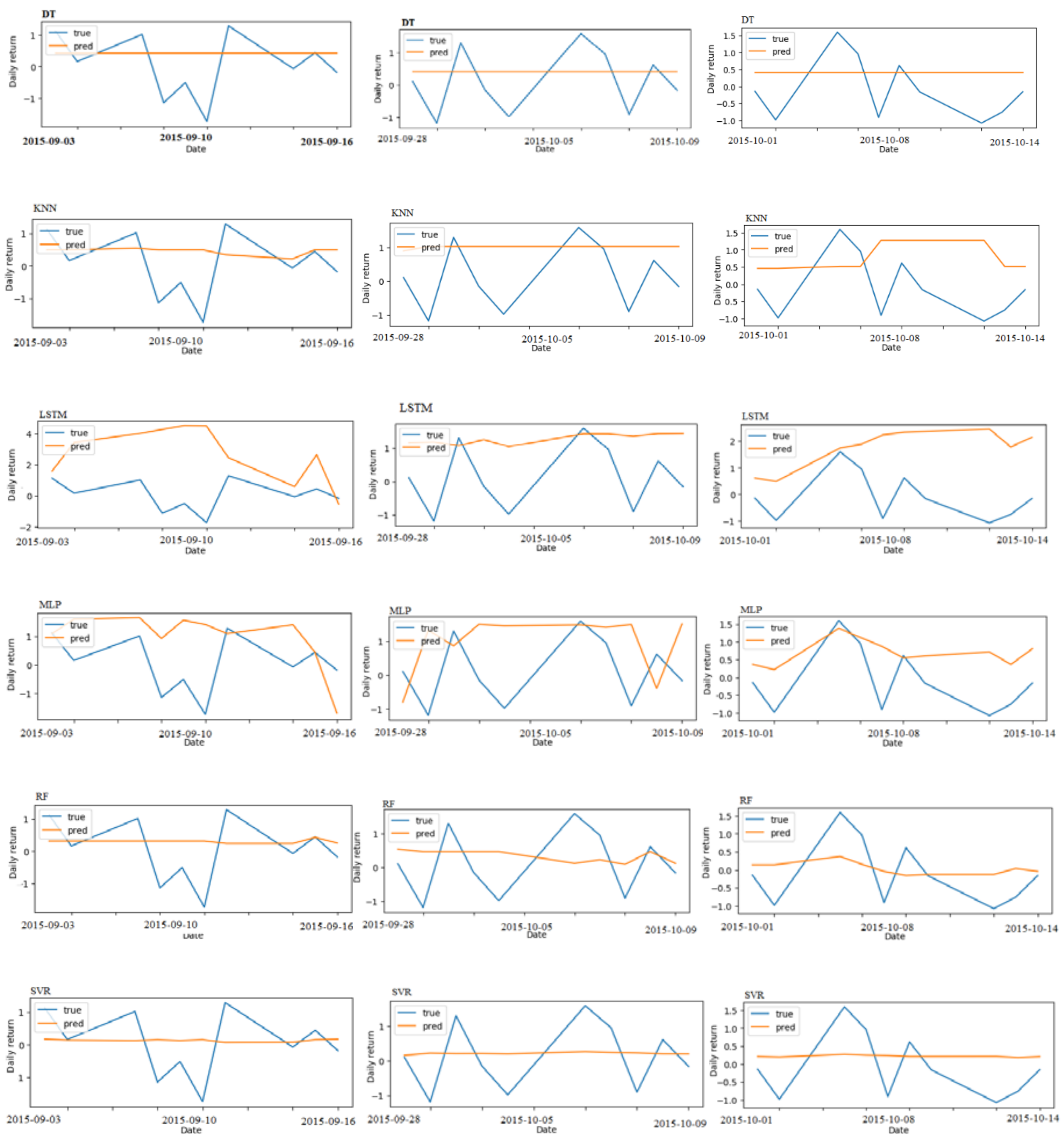

Figure 3.4. Sample chart results for 10 days prediction

From the shown chart Figures 3.2., 3.3. and 3.4. the SVR model may be considered having straight line prediction with the same result for each predicted day, but reviewing the results in the table shows that the SVR's result has no fixed prediction and it changes from day to day, and according to the chosen performance measure RMSE it gives the best result. In contrast by DT where a lot of predicted days have the same value, means the DT has a very low performance compared with other algorithms.

From the mentioned results, it can be observed that the SVR model gives the most stable performance with best results among other models, other studies also showed a superior performance of SVR over algorithms including; ANN and ANFIS [3],

Other very popular and strong algorithms including; LSTM and MLP are not doing very well, this can be interpreted due to the lacking data, It is known that these kinds of algorithms need huge data to shine and give good results, In the stock market movements prediction, looking backward so much in building the model, means that the training is done for different economic situations and different systems, For that, LSTM and MLP cannot take more data, One suggested method for solving this problem, is using data based on hours, minutes, seconds or even milliseconds, so the training can be done with very larger data on the same economic system.

The other algorithms including; RF, KNN and DT are in the middle performing area, with noticing that the RF has a better performance than DT, This confirms that the bagging has a better performance than normal individual methods.

In comparison with other studies, Asil [3] et al. have a notable study predicting the BIST100 index using different kinds of machine learning algorithms, but we found that theire results can't be compared with ours because they used classification method to predict the UP, Down movements of the BIST100 index, while we used regression method to predict the daily return changes.

Other study by Patel et al. [6] where they tried to predict the future prices of two index from Indian market named CNX Nifty and S&P BSE Sensex. They used 10 years of data for training, the prediction is made for 1, 10, 15 and 30 days in advance, where 10 technical indicators used as features for prediction. Although the data set used in this study is different than what we used, but both studies have similarities in predicting indices in the emerging markets and comparing the results using RMSE performance measure.

Our results show better predictions compared with of Patel et al. [6] Mentioned earlier, their prediction methodology consists of one stage also two stages prediction. In one stage prediction, they used single algorithm for each built model, and for the tow stages prediction, they used the result of one algorithm as an input for same/ another algorithm. Their best results show that for 1-day prediction, they have the single staged SVR model with the performance 1.26 RMSE for predicting CNX Nifty index and 1.25 for S&P BSE Sensex index. Our study has better results with 0.76 RMSE for 1-day prediction using SVR algorithm. Also, for tow days prediction, they have 1.40 RMSE for predicting CNX Nifty and 1.75 RMSE for S&P BSE Sensex using single stage SVR. Our results showed better performance with 1.10 RMSE for SVR algorithm. Also, for 5 days prediction, they have 2.92 for CNX Nifty using SVR and 2.83 for S&P BSE Sensex using SVR-SVR model. Our study showed better results with 1.29 RMSE for the SVR. For 10 days they have 3.99 RMSE for CNX Nifty using SVR-SVR and for the S&P BES Sensex they have 3.60 RMSE using SVR-ANN. Our study has better results with 1.38 RMSE using the SVR algorithms.

Also, our results showed better performance than the study done by Enke & Thawornwong [67] where they used a neural network for forecasting the stock market returns, their RMSE is 1.44 while we have 0.76 for 1-day prediction.

In comparison with study has better results than what we found, Atsalakis et al. [68], they used a neuro-fuzzy system composed of an ANFIS controller to predict stocks inside emerging and developed indices including Athens Stock Exchange (ASE) and the New York Stock Exchange (NYSE). They choose the same predicted variable we choose (the price changes) and the same performance measure (RMSE). Their results show very low RMSE 0.012200 for predicting the stock named TITAN and 0.0133 for IBM. We noted that these are too high results due to unshifting error happened in data, were we had like these results during the development of this thesis, and we noticed that there is no prediction in time series data -like stocks- if no shifting is used in data preparing, where the predicted variable column should be shifted one, for one step forward prediction (like predicting the next day if the data is daily based or predicting the next minute if the data is minutely based), and tow, for tow step prediction and so on.

4. CONCLUSION

The prediction of stock market prices is not an easy task to solve, even for the human mind, this is due to large number of factors that can affect the stock prices, these factors can be purely from inside the company, like its sales, dividends, and strategies, and also from outside the company like politics, news and the movement of the whole market,

In this study, six different machine learning models were deployed as a tool to help traders predict the daily return of the future price movements, It can be observed that the machine learning can help traders predicting the stock market movements, but this is affected a lot by the chosen features, collected data and the used algorithm.

In this study, the selected technical indicators and other features are fixed input, but in real life, some indicators may work better than others in certain circumstances, for this reason, it will be better in future works to use algorithms that can help choosing best features in best time like genetic algorithm, In addition, it will be better to do the analysis for smaller and bigger data as mentioned before like hours, minutes, seconds and milliseconds, Also, it will be better if the results and analysis stages are done using user-friendly graphical user interfaces (GUIs).

REFERENCES

1. Fama, E. F. (1965). Random walks in sstock market prices. *Financial Analysts Journal,* 51(1), 55-59.

2. İnternet: Bergen, J. V. (2017). Efficient market hypothesis: is the stock market efficient? .investopedia. web: http://www.webcitation.org/query?url=https%3A%2F%2Fwww.investopedia.com%2Farticles%2Fbasics%2F04%2F022004.asp+&date=2019-01-21 Accessed 25 Ağustos 2017.

3. Oztekin, A., Kizilaslan, R., Freund, S. ve lIseri, A. (2016). A data analytic approach to forecasting daily stock returns in an emerging market. *European Journal of Operational Research,* 253(3), 697-710.

4. Wei, H., Yoshiteru, N. ve Shou-Yang, W. (2005). Forecasting stock market movement direction with support vector machine. *Computer & Operation Research,* 32(10), 2513-2522.

5. Usmani, M., Adil Syed, H., Raza, K. ve Azhar Ali, S. S. (2016). *Stock market prediction using machine learning techniques.* 2016 3rd International Conference On Computer And Information Sciences, Kuala Lumpur, Malaysia.

6. Patel, J., Shah, S., Thakkar, P. ve Kotecha, K. (2015). Predicting stock market index using fusion of machine learning. *Expert Systems with Applications,* 42(4) 2162-2172

7. Attigeri, G. V., Pai, M., Pai, R. ve Nayak, A. (2015). *Stock market prediction: a big data approach.* TENCON 2015 - 2015 IEEE Region 10 Conference, Macao, China

8. Enke, D. ve Thawornwong, S. (2005). The use of data mining and neural networks for forecasting stock market returns. *Expert Systems with Applications,* 29(4), 927-940.

9. Kim, K. J. (2013). Financial time series forecasting using support vector machines. *Neurocomputing*, 55(1-2), 307–319.

10. Fischer, T. ve Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research,* 1, 1-16.

11. Gunduz, H., Yaslan, Y. ve Cataltepe, Z. (2017). *Intraday prediction of borsa Istanbul using convolutional neural networks and feature correlations.* Knowledge-Based Systems, 138-148.

12. Dash, R. ve Dash, P. K. (2016). A hybrid stock trading framework integrating technical analysis with machine learning techniques. *The Journal of Finance and Data Science* 2, 42-57.

13. Barak, S., Arjmand, A. ve Ortobelli, S. (2017). Fusion of multiple diverse predictors in stock market. *Information Fusion,* 90-102.

14. İnternet: Mitchell, C. (2018). The four most common indicators in trend trading. *investopedia.* web: http://www.webcitation.org/query?url=https%3A%2F%2Fwww.investopedia.com%

2Farticles%2Factive-trading%2F041814%2Ffour-most-commonlyused-indicators-trend-trading.asp+&date=2019-01-21 Accessed 28 February 2017.

15. İnternet: Cory Mitchell, C. (2017). How to trade with the momentum indicator. *Day Trading*. web: http://www.webcitation.org/query?url=https%3A%2F%2Fwww.thebalance.com%2Fhow-to-trade-with-the-momentum-indicator-1031195+&date=2019-01-21Accessed 28 February 2018.

16. İnternet: Balch, T. (2015). A few indicators momentum. *Udacity*. Web: http://www.webcitation.org/query?url=https%3A%2F%2Fwww.youtube.com%2Fwatch%3Fv%3DH7_J3jfiS5I%26index%3D130%26list%3DPLAwxTw4SYaPnIRwl6rad_mYwEk4Gmj7Mx%26t%3D0s+&date=2019-01-21Accessed 27 February 2018.

17. İnternet: (2018). Moving averages - simple and exponential. *stockcharts*. Web: http://www.webcitation.org/query?url=http%3A%2F%2Fstockcharts.com%2Fschool%2Fdoku.php%3Fid%3Dchart_school%3Atechnical_indicators%3Amoving_averages+Accessed+&date=2019-01-21, 27 February 2018.

18. İnternet: (2018). Exponential moving average (EMA). *Stockcharts*. Web: http://www.webcitation.org/query?url=https%3A%2F%2Fwww.fidelity.com%2Flearning-center%2Ftrading-investing%2Ftechnical-analysis%2Ftechnical-indicator-guide%2Fema%2C+&date=2019-01-21, Accessed 26 February 2018.

19. İnternet: (2018). Moving averages. *stockcharts*. Web: http://www.webcitation.org/query?url=http%3A%2F%2Fstockcharts.com%2Fschool%2Fdoku.php%3Fid%3Dchart_school%3Atechnical_indicators%3Amoving_averages&date=2019-01-21, Accessed 27 February 2018.

20. İnternet: (January 2018). Bollinger Bands (BB). *Tradingview*. Web: http://www.webcitation.org/query?url=https%3A%2F%2Fwww.tradingview.com%2Fwiki%2FBollinger_Bands_%28BB%29+&date=2019-01-21, Accessed 27 February 2018.

21. İnternet: Average Directional Index (ADX). *Stockcharts*. Web: http://www.webcitation.org/query?url=http%3A%2F%2Fwww.stockcharts.com%2Fschool%2Fdoku.php%3Fid%3Dchart_school%3Atechnical_indicators%3Aaverage_directional_index_adx+&date=2019-01-21, Accessed 19 February 2018.

22. İnternet: Schaap, C. (2018). ADX: the trend strength indicator. *Investopedia*. Web: http://www.webcitation.org/query?url=https%3A%2F%2Fwww.investopedia.com%2Farticles%2Ftrading%2F07%2Fadx-trend-indicator.asp+&date=2019-01-21, Accessed 19 February 2018.

23. İnternet: Chen, J. (2018). Wilder's DMI (ADX). *Investopedia*. Web: http://www.webcitation.org/query?url=https%3A%2F%2Fwww.investopedia.com%2Fterms%2Fw%2Fwilders-dmi-adx.asp+&date=2019-01-21 Accessed 06 January 2019.

24. İnternet: MACD Moving Average Convergence/Divergence Oscillator. *Stockcharts*. Web: http://www.webcitation.org/query?url=http%3A%2F%2Fstockcharts.com%2Fschool

%2Fdoku.php%3Fid%3Dchart_school%3Atechnical_indicators%3Amoving_average_convergence_divergence_macd.+&date=2019-01-21, Accessed 20 February 2018.

25. İnternet: Chen, J. (2018). Moving Average Convergence Divergence – MACD. Investopedia.Web: http://www.webcitation.org/query?url=https%3A%2F%2Fwww.investopedia.com%2Fterms%2Fm%2Fmacd.asp+&date=2019-01-21 Accessed 06 January 2019.

26. İnternet: (2014). Measure momentum change with ROC. *Investopedia.* Web: http://www.webcitation.org/query?url=https%3A%2F%2Fwww.investopedia.com%2Fterms%2Fm%2Fmacd.asp+&date=2019-01-21, asp Accessed 20 February 2018.

27. İnternet: Rate of Change (ROC). *Stockcharts*. Web: http://www.webcitation.org/query?url=http%3A%2F%2Fstockcharts.com%2Fschool%2Fdoku.php%3Fid%3Dchart_school%3Atechnical_indicators%3Arate_of_change_roc_and_momentum+&date=2019-01-21 Accessed 20 February 2018.

28. İnternet: Rate of Change (ROC). *Tradingview.* Web: http://www.webcitation.org/query?url=https%3A%2F%2Fwww.tradingview.com%2Fwiki%2FRate_of_Change_%28ROC%29+&date=2019-01-21, Accessed February 2 2018.

29. İnternet: Relative Strength Index (RSI). *Stockcharts*. Web: http://www.webcitation.org/query?url=http%3A%2F%2Fstockcharts.com%2Fschool%2Fdoku.php%3Fid%3Dchart_school%3Atechnical_indicators%3Arelative_strength_index_rsi.+&date=2019-01-21, Accessed 21 February 2018.

30. İnternet: Chen, J. (July 2018). Relative Strength Index (RSI). *Stockcharts*. Web: http://www.webcitation.org/query?url=https%3A%2F%2Fwww.tradingview.com%2Fwiki%2FRelative_Strength_Index_%28RSI&date=2019-01-21, Accessed 06 January 2019.

31. İnternet: Chen, J. (2018). Stochastic oscillator. *Investopedia*. Web: http://www.webcitation.org/query?url=https%3A%2F%2Fwww.investopedia.com%2Fterms%2Fs%2Fstochasticoscillator.asp+&date=2019-01-21, Accessed 06 January 2019.

32. İnternet: Stochastic Oscillator. *Stockcharts*. Web: http://www.webcitation.org/query?url=http%3A%2F%2Fstockcharts.com%2Fschool%2Fdoku.php%3Fid%3Dchart_school%3Atechnical_indicators%3Astochastic_oscillator_fast_slow_and_full+&date=2019-01-21 Accessed 24 2 2018.

33. İnternet: (May 2018). Stochastic stoch. *Tradingview*. Web: http://www.webcitation.org/query?url=https%3A%2F%2Fwww.tradingview.com%2Fwiki%2FStochastic_%28STOCH%29.+&date=2019-01-21, Accessed 06 January 2019.

34. İnternet: Jaiswal, A. (2017). On Balance Volume (OBV). *elearnmarkets*. Web: http://www.webcitation.org/query?url=https%3A%2F%2Fblog.elearnmarkets.com%2Fon-balance-volume%2F+&date=2019-01-21, Accessed 24 February 2018.

35. İnternet: On Balance Volume (OBV). *Stockcharts.* Web: http://www.webcitation.org/query?url=http%3A%2F%2Fstockcharts.com%2Fschool

%2Fdoku.php%3Fid%3Dchart_school%3Atechnical_indicators%3Aon_balance_volume_obv+&date=2019-01-21, Accessed 24 February 2018.

36. İnternet: Kenton, W. (2018). On-Balance Volume (OBV). *Investopedia*. Web: http://www.webcitation.org/query?url=https%3A%2F%2Fwww.investopedia.com%2Fterms%2Fo%2Fonbalancevolume.asp+&date=2019-01-21 Accessed 06 January 2019.

37. İnternet: (2018) On Balance Volume (OBV). *Tradingview*. Web: http://www.webcitation.org/query?url=https%3A%2F%2Fwww.tradingview.com%2Fwiki%2FOn_Balance_Volume_%28OBV%29+&date=2019-01-21 Accessed 24 February 2018.

38. İnternet: (2018) Decision tree learning. *Wikipedia*. Web: http://www.webcitation.org/query?url=https%3A%2F%2Fen.wikipedia.org%2Fwiki%2FDecision_tree_learning+&date=2019-01-21 Accessed 06 January 2019.

39. Bhattacharyya, D. k. ve Kalita, J. K. (2013). *Network anomaly detection: a machine learning perspective*. (First Edition). CRC Press: Chapman and Hall, 63-64.

40. Géron, A. (2017). Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems(First Edition). O'Reilly Media, 228-242.

41. İnternet: Taylor, J. (2001). Decision trees. *Stanford*. Web: http://www.webcitation.org/query?url=http%3A%2F%2Fstatweb.stanford.edu%2F%7Ejtaylo%2Fcourses%2Fstats202%2Ftrees.html&date=2019-01-21 Accessed 05 March 2018.

42. İnternet: Decision Trees. *scikit-learn*. Web: http://www.webcitation.org/query?url=http%3A%2F%2Fscikit-learn.org%2Fstable%2Fmodules%2Ftree.html%23regression+A&date=2019-01-21, ccessed 28 February 2018.

43. Mehryar, M., Rostamizadeh, A., Talwalkar, A. ve Bach, F. (2012). *Foundations of Machine Learning* (Second Edition). London: England: The MIT Press, 120.

44. İnternet: Demir, N. (2016). Ensemble Methods: Elegant Techniques to Produce Improved Machine Learning Results. *Kdnuggets*. Web: https://www.kdnuggets.com/2016/02/ensemble-methods-techniques-produce-improved-machine-learning.html Accessed 01 March 2018.

45. İnternet: Thompson, C. (2017). Applied Machine Learning in Python. *University of Michigan*. Web: http://www.webcitation.org/query?url=https%3A%2F%2Fwww.coursera.org%2Flearn%2Fpython-machine-learning+&date=2019-01-21 Accessed 01 January 2018.

46. İnternet: (2018) Bootstrap aggregating. *Wikipedia*. Web: http://www.webcitation.org/query?url=https%3A%2F%2Fen.wikipedia.org%2Fwiki%2FBootstrap_aggregating+&date=2019-01-21 Accessed 01 March 2018.

47. İnternet: Li, J. (2017). Bagging and Boosting: Brief Introduction. *The Pennsylvania State University*. Web: http://www.webcitation.org/query?url=http%3A%2F%2Fwww.personal.psu.edu%2F

jol2%2Fcourse%2Fstat597e%2Fnotes2%2Fbagging.pdf+&date=2019-01-21 Accessed 05 March 2018.

48. İnternet: Zakka, K. (2016). A Complete Guide to K-Nearest-Neighbors with Applications in Python and R. *kevinzakka*. Web: http://www.webcitation.org/query?url=https%3A%2F%2Fkevinzakka.github.io%2F2016%2F07%2F13%2Fk-nearest-neighbor+&date=2019-01-21, Accessed 05 March 2018.

49. İnternet: Brownlee, J. (2016). Parametric and Nonparametric Machine Learning Algorithms. *Machine Learning Mastery*. Web*:* http://www.webcitation.org/query?url=https%3A%2F%2Fmachinelearningmastery.com%2Fparametric-and-nonparametric-machine-learning-algorithms+&date=2019-01-21 Accessed 05 March 2018.

50. İnternet: Improved Outcomes Software. *Manhattan*. Web: http://www.webcitation.org/query?url=http%3A%2F%2Fwww.improvedoutcomes.com%2Fdocs%2FWebSiteDocs%2FClustering%2FClustering_Parameters%2FManhattan_Distance_Metric.htm.+&date=2019-01-21, Accessed 05 March 2018.

51. İnternet: Ng, A. (2016). Machine Learning. *Stanford University*. Web: http://www.webcitation.org/query?url=https%3A%2F%2Fwww.coursera.org%2Flearn%2Fmachine-learning+&date=2019-01-21 Accessed 05 March 2018.

52. James, D. Witten, T. ve Hastie R. (2013). *Tibshirani, An Introduction to Statistical Learning with Applications*. New York: Springer, 338.

53. İnternet: Thrun , S. ve Malon, K. (2015). Intro to Machine Learning. *Udacity*. Web: http://www.webcitation.org/query?url=https%3A%2F%2Fwww.udacity.com%2Fcourse%2Fintro-to-machine-learning--ud120+&date=2019-01-21, Access 05 March 2018.

54. İnternet: (2017). Feedforward neural network. *Wikipedia*. Web: http://www.webcitation.org/query?url=https%3A%2F%2Fen.wikipedia.org%2Fwiki%2FFeedforward_neural_network+&date=2019-01-21, Accessed 12 March 2018.

55. İnternet: (2018). Multilayer perceptron. *Wikipedia*, the free encyclopedia Web: http://www.webcitation.org/query?url=https%3A%2F%2Fen.wikipedia.org%2Fwiki%2FMultilayer_perceptron.+&date=2019-01-21, Accessed 12 March 2018.

56. İnternet: (2017). A Begginer's Guide to Multilayer Perceprons. *SkyMind*. Web*:* http://www.webcitation.org/query?url=https%3A%2F%2Fdeeplearning4j.org%2Fmultilayerperceptron+&date=2019-01-21 Accessed 12 March 2018.

57. İnternet: Raval , S. (2017). Which Activation Function Should I Use?. *Youtube*. Web: http://www.webcitation.org/query?url=https%3A%2F%2Fwww.youtube.com%2Fwatch%3Fv%3D-7scQpJT7uo+&date=2019-01-21Accessed 12 March 2018.

58. Krizhevsky, A., Sutskever, I. and Hinton, G. E. (2017). ImageNet Classification with Deep Convolutional. *Communications of the ACM*, 60(6), 84-90.

59. İnternet: wangjw, J. (2017). Introduction to Deep Learning. *Jiayiwangjw*.Web: http://www.webcitation.org/query?url=https%3A%2F%2Fjiayiwangjw.github.io%2F

2017%2F10%2F03%2FIntroduction-to-Deep-Learning-01+&date=2019-01-21, Accessed 12 March 2018.

60. İnternet: Karpathy, A. (2015). The Unreasonable Effectiveness of Recurrent Neural Networks. *Andrej Karpathy blog*. Web: http://www.webcitation.org/query?url=http%3A%2F%2Fkarpathy.github.io%2F2015%2F05%2F21%2Frnn-effectiveness%2F+&date=2019-01-21, Accessed 16 March 2018.

61. İnternet: Olah, C. (2015). Understanding LSTM Networks. *colah's blog*. Web: http://www.webcitation.org/query?url=http%3A%2F%2Fcolah.github.io%2Fposts%2F2015-08-Understanding-LSTMs+&date=2019-01-21, Accessed 16 March 2018.

62. Hochreiter, S. ve Schmidhuber, J. (1997). Long short-term memory. *Neural Computation,* 9(8)*,* 1735-1780.

63. İnternet: Andreaw, N. (2018). Deep Learning Specialization. *Coursera*. Web: http://www.webcitation.org/query?url=https%3A%2F%2Fwww.coursera.org%2Fspecializations%2Fdeep-learning+&date=2019-01-21 Accessed 16 March 2018.

64. İnternet: Kingma, D. P. ve Ba, J. (2017). Adam: A Method for Stochastic Optimization. *ICLR*. Web: http://www.webcitation.org/query?url=https%3A%2F%2Farxiv.org%2Fabs%2F1412.6980&date=2019-01-21

65. İnternet: Ruder, S. (2016). An overview of gradient descent optimization algorithms. *Insight Centre for Data Analytics NUI Galway*.

66. İnternet: F, F., Johnson, Li. J. and Yeung, S. (2017). Convolutional Neural Networks for Visual Recognition. *Stanford University*. Web: http://www.webcitation.org/query?url=http%3A%2F%2Fcs231n.stanford.edu%2Fsyllabus.html+&date=2019-01-21 Accessed 16 March 2018.

67. Enke, D. ve Thawornwong, S. (2005). The use of data mining and neural networks. *Expert Systems with Applications,* 29(4), 927-940.

68. Valavanis, K. P. ve Atsalakis, G. S. (2009). Forecasting stock market short-term trends using a neuro-fuzzy based methodology. *Expert Systems with Applications,* 36(7), 10696–10707.

69. İnternet: Balch, T. (2015). A few indicators Simple moving average. *Udacity*. Web: http://www.webcitation.org/query?url=https%3A%2F%2Fwww.youtube.com%2Fwatch%3Fv%3D4eKsY-KXEnc%26ind+ex%3D131%26t%3D0s%26list%3DPLAwxTw4SYaPnIRwl6rad_mYwEk4Gmj7Mx+&date=2019-01-21 Accessed 27 January 2018.

CURRICULUM VITAE

Personal Information

Last name, first name	: ZİYADOĞLU, Tarik
Nationality	: Turkish
Birth date and place	: 02,02,1989, Syria, Homs
Marital status	: Married
Mobile	: 0 (505) 105 06 33
E-mail	: tarekwarak@hotmail.com

Education

Degree	Graduate School	Graduation Date
Master	Gazi Üniversitesi / Teknoloji Fakültesi	Continuing
Bachelor's degree	Kalamoon University /IT Engineering	2012
High school	Khaleb bin Waleed high school	2007

Working Experience

Year	Place	Duty
2013-2015	Şam Inc.	Software Engineer

Foreign language

Arabic, English

Publications

Alshikh, Waraq. T., BARIŞÇI, N. (2017). Prediction of BIST100 index using technical analysis and machine learning techniques. *Fifth international symposium on engineering Artificial Intelligence and applications*, 1(1), 41.

Hobbies

Reading, Swimming and Water Polo

GAZİ GELECEKTİR,,,