İSTANBUL BİLGİ UNIVERSITY

INSTITUTE OF GRADAUTE PROGRAMS

FINANCIAL ECONOMICS MASTER'S DEGREE PROGRAM

COMPARISON OF THE PERFORMANCES OF BIST 30 PORTFOLIOS BY USING MACHINE LEARNING ALGORITHMS

Fatih ERİK
118620016

Assoc. Prof. Serda Selin ÖZTÜRK

İSTANBUL
2021

Comparison of the Performances of BIST 30 Portfolios By Using Machine Learning Algorithms

Makine Öğrenmesi Algoritmaları ile Oluşturulan BİST 30 Portföylerinin Performanslarının Karşılaştırılması

Fatih Erik

118620016

Tez Danışmanı:	Doç. Dr. Serda Selin Öztürk İstanbul Bilgi Üniversitesi	(İmza): ……
Jüri Üyesi:	Dr. Öğr. Gör. Fatma Didin Sönmez İstanbul Bilgi Üniversitesi	(İmza):…….
	Doç. Dr. Ender Demir İstanbul Medeniyet Üniversitesi	(İmza):……..…

Tezin Onaylandığı Tarih : 24.06.2021
Toplam Sayfa Sayısı : 71

Anahtar Kelimeler:	Keywords:
1) Makine Öğrenmesi Algoritmaları	1) Machine Learning Algorithms
2) Aktif/Pasif Portföy Yönetimi	2) Active/Passive Portfolio Management
3) Yatırım Stratejisi	3) Invesment Strategy
4) Fiyat Tahmini	4) Price Prediction
5) BİST 30	5) BIST 30

ACKNOWLEDGEMENT

First of all, I am happy to be able to achieve what I had imagined in the beginning of this study and to contribute to the literature in this field that I have a passion for.

Taking this opportunity, I would like to thank my thesis advisor Assoc. Prof. Serda Selin Öztürk for her support and close collaboration in this challenging period of the pandemic.

I would also like to thank my father and my mother due to their guidance and efforts throughout my education.

Last but not least, a special thanks to my dear wife who has always been there to support me along the way.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

AI: Artificial Intelligence

BIST : Borsa İstanbul

EDA: Exploratory Data Analysis

LSTM:Long-Short Term Memory

MAE : Mean Absolute Error

MAPE : Mean Absolute Percentage Error

MSE : Mean Squared Error

ML : Machine Learning

NN : Neural Network

RMSE : Root Mean Square Error

RNN : Recurrent Neural Network

SD : Standard Deviation

SVM : Support Vector Machine

XGBoost : Extreme Gradient Boosting

USD: US Dollar

TRY: Turkish Lira

EUR: Euro

XAU: Gold

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

In this study, the prices of BIST 30 stocks are estimated using machine learning algorithms such as linear regression, decision tree, support vector machines, long-short term memory and XGBoost, and based on these predicted prices, various portfolios have been created. Portfolios are generally managed in two ways, active and passive. The most important factor while choosing one is their return. Current studies show that the returns of funds traded on Borsa Istanbul Stock Exchange performed below the market, in other words, passive management beat active management. On the other hand, there are also studies showing that portfolios to be created with certain strategies yielded higher returns than the market. In this study, first of all, the prices of BIST 30 stocks for 1 month, 2 months, 3 months, 4 months, 6 months and 12 months are predicted by using machine learning algorithms and according to these revision periods, 6 strategies were created. In the second stage, these strategies were diversified by changing the portfolio sizes (5-10-15 shares). As a result, a total of 90 strategies were formed with 3 different sizes, 6 different timeframes and 5 different algorithms. As a result of the study, it was determined that the returns of portfolios created by using machine learning algorithms are generally above the return of the BIST 30 index. Also, the LSTM algorithm generally makes more successful predictions, 12-month strategies yield higher returns than other strategies, and portfolios including 5 shares are more successful than other portfolios. The result table shows that the most successful portfolio is the portfolio with 5 shares, revised every 12 month and created by using the XGBoost algorithm.

Keywords: Machine Learning Algorithms, Active/Passive Portfolio Management, BIST 30

ÖZET

Bu çalışmada makine ogrenmesi algoritmalarindan lojistik regresyon, karar ağacı (decision tree), sektor vektor makineleri (support vector machine), uzun-kisa sureli bellek (long-short term memory) ve XGBoost kullanilarak BIST 30 hisse senetlerinin fiyatlari tahmin edilmis ve bu tahmin fiyatlari baz alinarak cesitli buyukluklerde portfoyler olusturulmustur. Portföyler genel olarak aktif ve pasif olmak üzere iki şekilde yönetilmektedir. Yatırımcıların bunlardan hangisini seçeceğini belirleyen en önemli faktör sağladıkları getirilerdir. Mevcut çalışmalar, BİST'te işlem gören fonların getirilerinin piyasanın altında performans sergilediğini, diğer bir deyişle pasif yönetimin aktif yönetimi yendiğini göstermektedir. Bununla beraber belirli stratejiler kapsamında oluşturulacak portföylerin, endeksin üzerinde getiri sağladığını gösteren çalışmalar da mevcuttur. Bu çalışmada ilk önce makine öğrenmesi yöntemleri uygulanarak BIST 30 hisselerinin 1 ay, 2, ay, 3 ay, 4 ay, 6 ay ve 12 aylik fiyatlari tahmin edilmis ve bu aylara gore portfoy revizyonunun yapildigi 6 strateji olusturulmustur. İkinci aşamada ise sozkonusu stratejiler portfoy büyüklükleri (5-10-15 hisse) değiştirilerek çeşitlendirilmiştir. Sonuc olarak 5 farklı algoritmadan, 6 farklı zaman diliminde kendini revize eden, 3 farkli buyuklukte toplam 90 strateji bulunmustur. Çalışmanın sonucunda, makine öğrenmesi algoritmaları kullanılarak oluşturulan portföylerin getirilerinin genel olarak BIST 30 endeksinin getirisinin uzerinde oldugu tespit edilmistir. Ayrica LSTM algoritmasinin genel olarak daha basarili tahminler yaptigi, 12 aylik stratejilerin diger stratejilerden daha yüksek getirili oldugu ve 5 hisseden oluşan portföylerin diger portföylerden daha basarili oldugu gorulmustur. Sonuc tablosu en basarili olan portföyün XGBoost algoritmasını kullanan, 12 ayda revize edilen ve 5 hisseden oluşan portföy oldugunu gostermektedir.

Anahtar Sözcükler: Makine Öğrenmesi Algoritmaları, Aktif/Pasif Portföy Yönetimi, BİST 30

INTRODUCTION

Along with the advance of technology, machine learning algorithms are widely used in almost every field of our daily lives as healthcare, construction, transportation, marketing, management and finance. Specifically these kinds of algorithms are used in the financial market in order to make predictions of the prices of financial instruments and to make portfolios as well.

Investors form portfolios not only because there are a variety of investment instruments but also because investing in only one instrument is quite risky. Generally the goal is minimizing the risk while getting the maximum return from the portfolio.

While forming a portfolio, the assets included and the ways of managing it are two important factors that affect the return of the portfolio. There are two methods that investors can choose between, while managing their portfolios. These are active and passive portfolio management.

Passive management assumes that markets are efficient, that securities reflect their real prices (that they are not expensive or cheap), thus trying to find cheap securities through analysis and trading often do not work. In this method, investors usually invest in a certain index without trading.

On the other hand, active management supports that markets are not efficient, including low-price securities in the portfolio at the right time can increase return. The securities in the portfolio are revised in accordance with the determined strategy and within the determined period in order to maximize return.

In the literature, there are studies that report that passive management is more profitable in the long run. On the other hand, there are also studies that support that active management conducted with certain strategies has more return than market return.

In most studies regarding BIST that support the passive management, the performance of A and B type mutual funds were measured by using Sharpe, Treynor

and Jensen criteria. It was seen that fund return usually was under market return. (Gökgöz & Günel, 2012).

In other studies like Okur (2009) and Zengin (2006), it was reported that portfolios that were formed with certain criteria (e.g. dividend yield) gained over market return. In literature, there are relatively few studies showing the superiority of the active management by using ML algorithms in the portfolio making process. These strategies are generally on foreign exchanges. In their studies on cryptocurrency market and S&P500, Jiang et al. (2017) and Kaczmarek and Perez (2020) found that the return of the portfolios created by using ML outperformed the market.

The main goal of this study is to find answers for these main three questions: i) Are machine learning algorithms really good at making predictions and forming portfolios? ii) What is the optimum number of stocks in a portfolio? iii) What is the optimum revision period to get the maximum return?

BIST 30 stocks are used while making portfolios and along with their daily close price, dataset includes BIST100 index, BIST 30 index, S&P 500 index, 1 year bond, 10 year bond, CDS, USD/TRY, EUR/USD, XAU/USD and oil prices. The dataset consists of data in the period of 1 January 2016-31 December 2020 (1256 days).

In the first phase of this study, price predictions are made by using 5 machine learning algorithms (Linear regression, decision tree, support vector machine, long-short term memory and XGboost) and the accuracy scores are compared with each other. In the second phase, various portfolios are formed with 5 different algorithms, 6 different revision periods and 3 different sizes. At the end the results are compared with each other and with the return of the BIST 30 index for the same period.

The structure of the current study is as follows: in the first two sections machine learning algorithms and portfolio management strategies will be explained. In the third section studies in the literature will be explained. On the other hand, in the third and fourth sections the dataset and method will be introduced. Conclusion will be the last section of the study.

1. MACHINE LEARNING (ML) AND ML ALGORITHMS

1.1.Machine Learning

Machine learning (ML) is a form of Artificial Intelligence (AI) that enables computers to recognize patterns in data so as to build algorithms to make predictions. The accuracy rate of the prediction is directly related to the level of development of the algorithm, which increases along with the amount of data.

ML enables us to identify patterns in the data and to understand the data easily. Also, it increases data integrity, lowers cost with automation and improves user experience. Today ML is used in various sectors such as healthcare, transformation, retail, agriculture, banking and finance. Risk management, price prediction and fraud detection are some of the fields where ML is widely used in finance (Heaton et al., 2017).

There are some important steps while applying ML algorithms on a problem. First, data is compiled and exploratory data analysis (EDA) methods are used to analyze the dataset. In this process the patterns, anomalies and the features of the data can be identified. Second, the data is divided into two parts as train dataset and test dataset. Train set is used to train the model. Third, the trained model is applied on the test data and the performance of the model is observed. Finally, the prediction is made and interpretation can be made on it.

1.1.1. Supervised Learning

In this method, the dataset has labeled data and the algorithm learns and makes connections according to this labeled data. It has two categories:

- Regression

- Classification

1.1.2. Unsupervised Learning

In this method, the dataset does not have labeled data as supervised learning. The model clusters the data without labeled data. It has three categories:

- Clustering

- Association Rule Mining

- Dimensionality Reduction

1.1.3. Reinforcement Learning

In this method, the algorithm learns based on the feedback of its trial. It makes trials and classifies the results as positive and negative. It does not repeat the same method giving negative results.

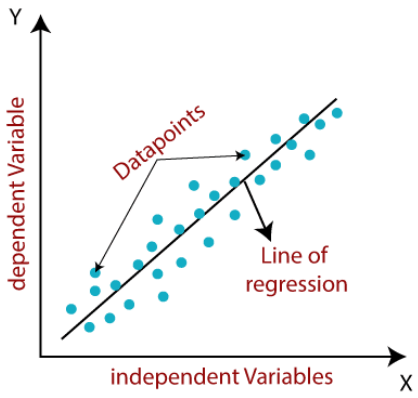
1.2. Machine Learning Algorithms used in this study

In this study 5 different machine learning algorithms are used while predicting the stock price. These are linear regression, decision tree, support vector machine (svm), long-short term memory (lstm) and xgboost.

1.2.1. Linear Regression

Linear regression is one of the supervised ML algorithms that is used for building a linear relationship between dependent and independent values (Kavitha et al., 2016). It is generally used for making predictions for continuous values. The model gives a straight line according to the changes of the dependent and independent values.

Figure 1.1: Line of Regression (Javatpoint, n.d.)



y= w0+w1x

y: output (dependent value)

x: input (independent value)

w0:intercept

w1: coefficient of the model (slope of the line)

1.2.1.1. Types of Linear Regression

Simple Linear Regression: There is only one independent variable (x) in the regression model.

y= w0+w1x

Multiple Linear regression: In the regression model there are at least two or more independent variables (x).

f(x,y,z) =w0+w1x+w2y+w3z

1.2.1.2. Cost function

The model forms an equation like above and tries to find w0 and w1 in order to identify the relationship between variables. Different w0 and w1 forms different regression lines and it gives different residuals which means the difference between the predicted values (y_pred) and actual values (y). In order to make a comparison between various regression lines, the model requires a cost function and for linear regression the cost function is mean squared error (MSE). MSE is the average squared error between the actual and predicted values.

Therefore the model calculates MSE for each different regression line and tries to minimize the cost function. Actually it aims to find the equation below:

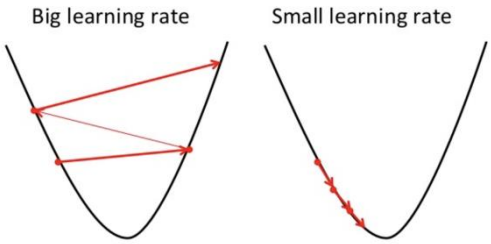
$$minimize \frac{1}{n}\sum_{i=1}^{n}(pred_i - y_i)^2$$

$$J = \frac{1}{n}\sum_{i=1}^{n}(pred_i - y_i)^2$$

1.2.1.3. Gradient Descent

Gradient descent is an algorithm which linear regression model uses to minimize the cost function (MSE). The coefficients of the equation are updated by gradient descent in each iteration in order to find the minimum MSE. The iteration continues until the minimum MSE is found. There is an important concept while using gradient descent: learning rate. Learning rate means the size of changes in each iteration. With a big learning rate, there is possibility to not converge. On the other hand, with a small learning rate, it takes a long time to find the minimum error.

Figure 1.2: Learning Rate (Agrawal, 2021)



1.2.1.4. Assumption of the Linear Regression

-- There is a linear relationship between y and x.

-- There is no or little multicollinearity between the independent variables.

-- There is no autocorrelation in error terms (residuals).

-- The error terms are distributed normally.

-- The error terms are homoscedastic.

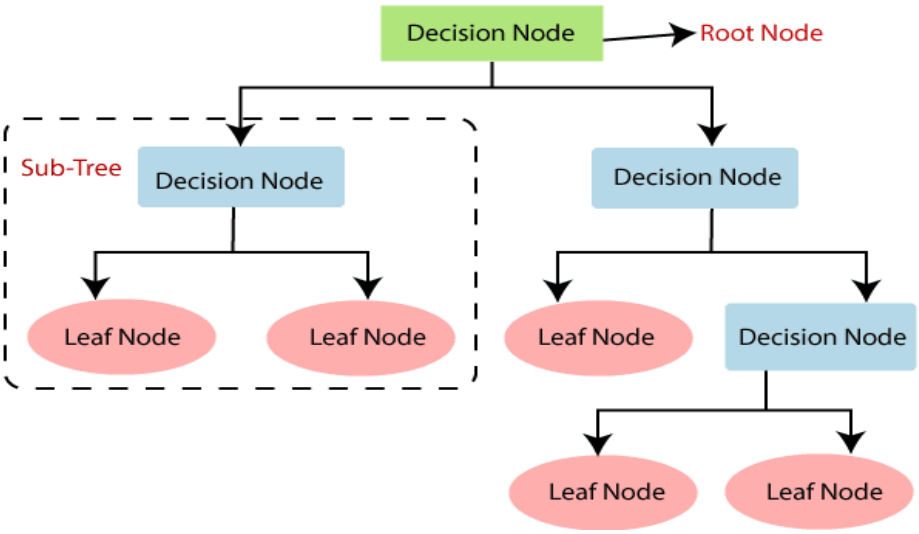
1.2.2. Decision Tree

Decision tree algorithm is one of the supervised ML algorithms that is used for regression and classification. It is called a tree because it uses nodes and branches while making decisions. It has two categories. Classification tree is used when making classification, on the other hand regression tree is used when making prediction out of the dataset (Osisanwo et al., 2017).

There are some important terms about decision tree algorithms.

- **Root node**: It is where the tree starts to divide. It includes the whole dataset.
- **Decision Node:** It refers to the nodes that can be splitted.
- **Leaf node:** It is the output of a node. After the leaf node there is no further splitting.
- **Splitting:** Forming sub-nodes from a single node.
- **Sub-tree:** Small trees comprising the big tree.
- **Pruning:** Deleting the branches of the decision tree.
- **Parent/Child node:** Parent node represents the root node, on the other hand, child node represents sub-nodes.

Figure 1.3: Decision Tree (Javatpoint, n.d.)



While making classification, the model starts from the root node and in every node, it compares the attribute value with the subnodes value and this process continues till the model reaches the leaf node. However, when it decides first at the root node, it needs a feature, since each feature brings the model to a different tree. Therefore the model should select the best attribute in order to find the optimum tree. In this point the model uses Attribute Selection Measure (ASM) while deciding the best attribute. There are two important widely used techniques in ASM for this purpose.

1.2.2.1. Information Gain

Information gain is a value that refers to the difference in entropy after splitting of a node. The model tried to find the biggest information gain when deciding the best attribute for splitting. It can be calculated by the equation below:

Information Gain= Entropy(S)- [(Weighted Avg) *Entropy(each feature)

Entropy(s)= -P(yes)log2 P(yes)- P(no) log2 P(no)

S= the size of samples

P(yes)= probability of yes

P(no)= probability of no

1.2.2.2. Gini Index

Gini Index can be considered as a cost function that is used for determining the inequality of the split. So. higher Gini index means higher inequality. An attribute giving low Gini Index should be chosen for the model.

$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2$$

Decision tree model is easy to understand since it is working like the human decision making process. On the other hand, it is prone to overfitting since it continues splitting till it reaches the leaf node. It means that the model learns the training dataset perfectly and the accuracy of the test set will show low performance. However, pruning the tree and using the Random Forest algorithm are two basic solutions for the problem of decision tree overfitting.

1.2.3. Support Vector Machine (SVM)

Support Vector Machine (SVM) is one of the supervised ML algorithms that is used for both regression and classification problems. The algorithm uses a hyperplane (decision boundary) to split the data and to make classification (Osisanwo, 2017). The model decides the best hyperplane according to the support

vectors. Support vectors are the data points which are closest to the decision boundary (hyperplane).

Figure 1.4: Classification with Linear SVM (Javatpoint, n.d.)

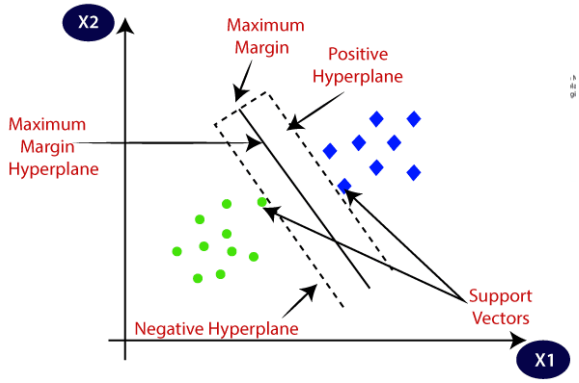
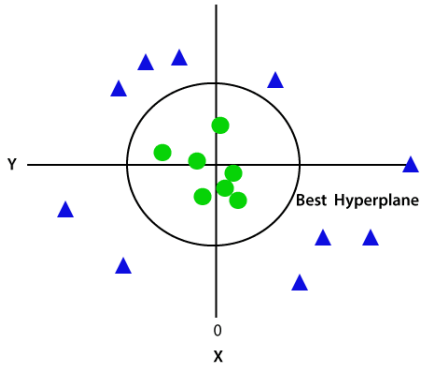


Figure 1.5: Non-linear SVM (Javatpoint, n.d.)



The model tries to find the optimum hyperplane to make classification. First, it finds the extreme points of the classes and draws the support vectors as above. Later, it draws a hyperplane between these support vectors and calculates the distance between the support vectors and hyperplane. This distance is called margin. As it may be imagined, it can be drawn more than one hyperplane between these support vectors. Therefore the model decides the hyperplane as the optimum hyperplane which maximizes the margin. When the data is splitted with a hyperplane, it is easy to decide the class of the new data.

1.2.3.1. Types of SVM

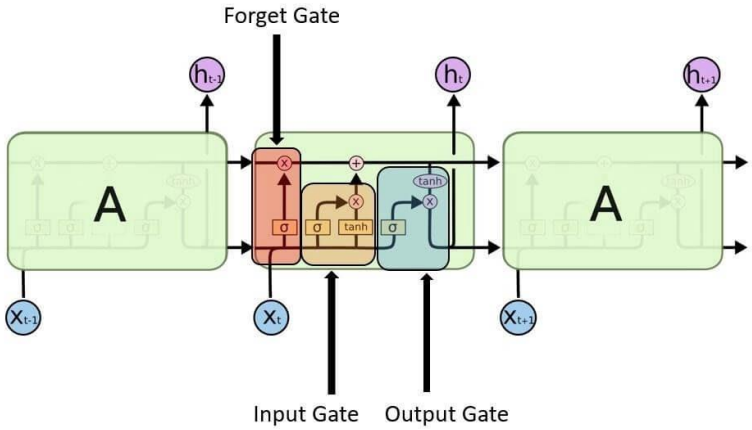
-- Linear SVM: If there are two classes, the data can be splitted with a straight line. Therefore the hyperplane would be a line.

-- Non-linear SVM: If there are more than two classes, a straight line can not split the data and it needs a hyperplane with n-dimension plane.

1.2.4. Long Short Term Memory (LSTM)

LSTM is one the artificial recurrent neural network algorithms that is used for classification, process and prediction (Heaton et al., 2018). Created by Sepp Hochreiter and Jurgen Schmidhuber in 1997, it uses cells containing gates (input gate, output gate and forget gate). These gates enable the model to remember the values in the memory and to process multiple data points at the same time. The output of the previous cell is used as an input for the next cell.

Figure 1.6: LSTM Gates (Javatpoint, n.d.)



1.2.4.1.Gates

-- **Input gate:** Input gate decides the value used in the memory. One one hand, sigmoid function decides the values by labeling them as 0 or 1 and on the other hand, tanh function decides the weight of the values (from -1 to 1).

$$i_t = \sigma(W_i . [h_t - 1, x_t] + bi)$$
$$C_t = \tanh\,(W_C . [h_t - 1, x_t] + b_C)$$

-- **Forget gate:** In this gate sigmoid function is used to decide the value that should be ignored by labeling them as 0 (omit) or 1 (keep).

$$f_t = \sigma(W_f . [h_t - 1, x_t] + b_f)$$

-- **Output gate**: It decides the output by using the results of the input gate and forget gate. Sigmoid and tanh functions are used in the same way explained above.

$$O_t = \sigma(W_o[h_t - 1, x_t] + b_o$$
$$h_t = o_t * \tanh(C_t)$$

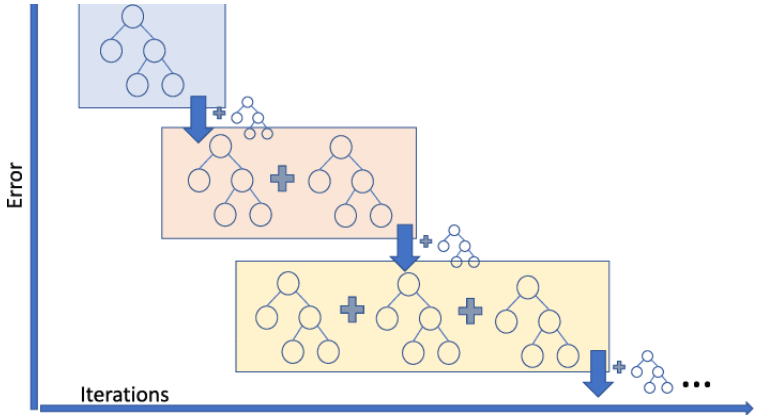
1.2.4.2. Application of the model

1) Creating LSTM layers,
2) Building a function to calculate the loss function, optimizer and accuracy rate,
3) Training the model
4) Finding out the results of training loss, training accuracy and validation accuracy for each epoch.
5) Testing model
6) Compare the test accuracy score with the results of the training period.

1.2.5. XGBoost

XGBoost is one of the ML algorithms that can be used for making classification and prediction. Actually XGBoost is an implementation of Gradient Boosting algorithm. Gradient Boosting can be identified as the advanced version of the decision tree algorithm. Basically Gradient Boosting algorithms form decision trees and, as in all machine learning algorithms, they aim to minimize errors (Gümüş & Kıran, 2017).

Figure 1.7: Gradient Boosting (Baturynska,2021)



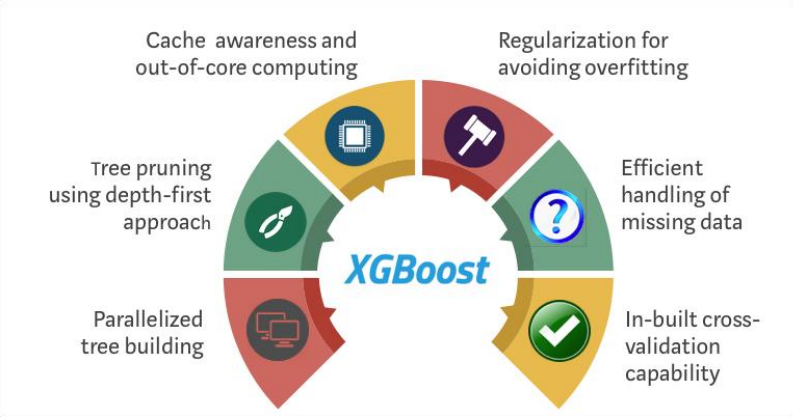
However in the Gradient Boosting method, unlike decision trees, the model proceed by adding decision trees on top of each other and this process continues until the minimum error is found. One of the most important factors in Gradient Boosting method is, like in the decision trees, to decide how the parent node splits into children nodes. In the XGBoost model, two equations are used for this purpose. gi refers to the first order differential of the loss function, while hi refers to the second order.

$$gi = \partial\, \hat{y}i\ (t-1)\ l(yi\ ,\ \hat{y}i\ (t-1)\ )$$

$$hi = \partial\ 2\ \hat{y}i\ (t-1)\ l(yi\ ,\ \hat{y}i\ (t-1)\ )$$

XGBoost models perform better when compared with the other Gradient Boosting methods. One of the reasons is that XGBoost is pruning. If the branches exceed the decided level, the model prunes them in order to prevent the tree from being too complicated. Furthermore, the algorithm enables us to tune the parameter and find the best model giving the highest result. Additionally, the model is so fast, because it is making parallelization while running and this process uses all CPU of the computer.

Figure 1.8: XGBoost Features (Programmersought, n.d.)



2. PORTFOLIO MANAGEMENT STRATEGIES

2.1. Portfolio Management

Portfolio management is the method of managing a group of financial instruments in order to get the optimum return within a certain risk level (Kevin, 2015). For achieving this goal, portfolio managers need to understand all opportunities and threats regarding the instruments in a financial market.

2.1.1. Asset Allocation

In order to build an effective portfolio, managers use various financial instruments from different asset classes. These asset classes may include stocks, bonds, derivatives etc. The main reason is that different asset classes react differently to the developments in the market, their volatilities are different in other words, and this approach protects the portfolio from the market risk. Some instruments have high return opportunity / high volatility and aggressive investors may choose this kind of investment strategy. On the other hand, conservative investors may invest low return opportunity / low risk instruments. Therefore, the character of the portfolio reflects the character of the investors.

2.1.2. Diversification

Investors form portfolios not only because there are a variety of investment instruments but also because investing in only one instrument is quite risky. Generally the goal is minimizing the risk while getting the maximum return from the portfolio (Brentani, 2004). Because of the fact that managers can not always know which instrument outperforms the market, diversification is made to get a return above the market while reducing the risk. Managers can select various instruments in the same asset class when building a portfolio. However, the effective way would be choosing them from different asset classes.

2.1.3. Rebalancing

While building a portfolio, it is important for the managers to decide the weight of each instrument in the portfolio. Because these weights directly affect the portfolio

return. Managers can change the weights according to the situation of the asset classes. When the stock market is in rally, increasing the weight of stocks in the portfolio can be decided. This kind of shift may result in a positive result, however it also changes the risk of the portfolio and it requires rebalancing when the rally finishes.

2.1.4. Portfolio Management Process

While building a portfolio and managing it, there are some critical steps that need to be followed by the portfolio managers (Chandra, 2017). These includes,

-- Deciding the goal of the investment,

-- Making market analysis in order to understand the expected return and the risk,

-- Deciding the asset classes,

-- Creating a portfolio strategy,

-- Selecting the instrument,

-- Managing the portfolio according to the criteria decided before,

--Evaluating the results at the end of the period,

-- Rebalancing it if needed.

2.2. Portfolio Management Strategies

Mainly there are two portfolio management strategies.

2.2.1. Passive Management

In passive management, managers buy an instrument at the beginning of the investment period and keep it till the period ends. This investment strategy is also called a buy-and-hold strategy. Within this period, managers do not sell the investment even when the prices go down (Cox, 2017).

It can be decided to invest in a portfolio rather than invest in one instrument. In this case managers choose an index and buy the instrument with the same weight as

they are in the index. With this way, managers aim to get the same return with the index. Generally investors accepting that Efficient Market Theory prevails in the market prefer to use this management strategy. Theory suggests that financial markets work efficiently and in the long run it is impossible to outperform the market return. Therefore changing the investment in the portfolio will not work.

The other strategy in passive management is keeping buying an investment in the long run regardless of its prices. In this strategy, investors keep buying even when the prices decrease and that reduces the cost of the portfolio. Investors using this strategy aim to make profit when the prices go up again due to its low portfolio cost.

Table 2.1: Advantages and Disadvantages of Passive Management

Advantages	**Disadvantages**
Low fees	Limited passive funds
Transparency	Small return
Tax-efficiency	

2.2.2. Active Management

In this management style, investors believe that financial markets do not work efficiently and beating the market can be possible by changing the portfolio according to certain strategies. Investors try to outperform the market by taking advantage of price fluctuations (Cox, 2017). Because the portfolio changes based on a strategy, the return depends on the success of this strategy. Therefore while comparing with passive management, it can be said that active management requires more expertise. Various strategies can be formed by using technical analysis, fundamental analysis, price prediction etc.

Managers using this method believe that there are some irregularities in the market and if the appropriate time is found, it is possible to buy low and sell high and get profit from these inefficiencies. Because more trading is made in this strategy, transaction cost is higher than passive management. However, the choices are not limited as in the passive strategy as it does not need to be invested in only indexes.

On the other hand, the instrument selection process also increases the risk of the portfolio due the fact that instruments with high return opportunity also have high volatility or high risk. However, there are some methods that limit the risks and losses like hedging.

Table 2.2: Advantages and Disadvantages of Active Management

Advantages	**Disadvantages**
Flexibility	High transaction costs
Hedging	Active risk
Tax-management	

3. LITERATURE REVIEW

While looking at the literature, the studies regarding active and passive management on BIST are examined firstly. Afterwards, other studies including ML applications on price prediction and portfolio optimization are analyzed.

3.1. Studies over Portfolio Management Strategies

In literature there are lots of studies supporting active or passive management on both BIST and foreign exchanges. However, it can be asserted that studies demonstrating the superiority of active management are fewer than those of passive management.

In most studies that support the passive management, various portfolios have been formed and the results were compared with the return of the corresponding indexes.

In Terzi and Saldanlı's (2019) study, 12 investment strategies were determined by including the daily closing prices of the companies in the BIST 30 index between 01.01.2016 and 31.12.2017. As a result of the analysis, it was found that passive investment strategies were suitable for individual investors. Zengin (2006) created 7 portfolios by taking into account the 2005 prices of 5 stocks traded on the BIST (named IMKB at the time) and compared their performances. As a result, it was concluded that the "buy-and-hold" strategy provided the biggest profit with 16%.

Additionally, there are also studies observing the performance of A and B type mutual funds by using Sharpe, Treynor and Jensen criteria. Generally it was seen that the return of these funds was below market return.

Arslan (2005) evaluated the performance of 45 A-Type mutual funds between 2002 and 2005 with the performance measurements of Sharpe, M2, Jensen and Treynor. The study concluded that most of the funds performed poorly compared to the market.

On the other hand, some studies emphasizing that portfolios that were formed with certain criteria (e.g. dividend yield) gained over market return, also exist.

Okur (2009) created portfolios by calculating the average monthly dividend yield of stocks traded on the IMKB and revised them quarterly between January 2005 and March 2009. As a result, it was observed that the inclusion of stocks with high dividend yield into the portfolio provided a higher return compared to the index.

In the study of Kılıç et al. (2014), it was concluded that the BIST 100 index acted according to a certain pattern and the market was not efficient.

3.2. Studies over Price Prediction and Portfolio Optimization by using Machine Learning

Emerson et al. stated in a study (2019) that the studies in the literature on the applications of ML techniques in finance were generally concerning price estimation, portfolio creation and risk modelling.

3.2.1. Studies over BIST

Avcı (2009) made a study with 5 shares in BIST 30 and obtained better results than the buy-and-hold strategy with artificial neural networks.

Kara et al. (2011) aimed to estimate the direction of daily prices of the BIST 100 index. In this study, artificial neural networks and support vector machines (SVM) were compared with each other. An accuracy rate of 75.74 % with the artificial neural network model and a rate of 71.52 % with support vector machines were obtained.

In Kara and Ecer's study (2018), the prediction of the movement direction of the stock market index was studied by using the classification methods with a total of 5750 daily data belonging to the BIST Bank index between the years 1995-2018. As a result, it was determined that the artificial neural network model performed better than other models with a rate of 81.74%.

Hatipoğlu and Uyar (2018) investigated the stock returns of 94 companies operating without cease in the BIST100 index for the period of 2011-2016 Bayesian network models were used and qualitative and quantitative information that investors could use in portfolio selection were obtained.

Uyar (2019) created portfolios from BIST, FTSE and DAX markets using machine learning and HRP methods, updated them every 12 months and examined the performances of these portfolios between July 2005 and June 2017. As a result of the analysis, it was concluded that the HRP method was not successful in BIST and FTS markets, but performed successfully in DAX markets.

Gündüz et al. (2017), studied the estimation of the daily closing prices of BIST shares of Garanti Bank, Turkish Airlines and İş Bank companies. Better results were achieved through use of the gradient boosting machine compared to other methods.

In Erkartal's (2018) study, Garanti Bank stock trading signals for 5 days afterwards were created. Decision tree, support vector machines and recurrent neural network models produced predictions for these signals. With a performance of 80%, the decision tree model yielded better results.

3.2.2. Studies over Foreign Exchanges

Most of the articles reviewed focus on the estimated returns of a single stock market index or multiple stock market indices. Only several of them study the prediction of a single stock or multiple stock behavior (Atsalakis & Valavanis, 2009). Historical prices form the basis of these stock-oriented forecasts, which produce the direction of the price.

Jiang et al. (2017) aimed to predict the 30-minute prices of the most traded cryptocurrencies in the financial markets, and according to these predictions, they created a portfolio including 12 cryptocurrencies to be revised every 30 minutes. As a result of the study, they determined that artificial neural network algorithms gave better results than other models.

In Kaczmarek and Perez's (2020) study, using the random forest algorithm, the 30-day prices of the S&P 500 shares for the years 1999-2019 were predicted. Using 1 / N, mean-variance and HRP, 10 different portfolios from 25 to 250 shares (incrementing by 25) were created and the results were compared. It was concluded that the most successful portfolios were those with a high number of shares created

through mean-variance. The results of the other 3 methods outperformed the market.

In their study, Gu et al. (2020) made a monthly price estimation by using 60-year data of stocks traded in US stock exchanges and created equally weighted portfolios. As a result of the study, it was observed that machine learning algorithms were more successful than linear models, especially artificial neural networks and random forest algorithms gave better results.

In the study by Kinn (2018), a portfolio using the data of 10 stocks in the S&P 500 between 2012-2017 was created and optimized by means of ML algorithms. It was concluded that the risk of the portfolio created using ML was lower than traditional methods and equally weighted portfolios.

Jain and Jain (2019) created 5 portfolios, each containing 10 stocks from the National Stock Exchange of India, and weighted these portfolios with various methods. Consequently, they concluded that portfolios weighted using ML techniques performed better than those weighted using traditional methods.

Dey et al. (2016) aimed to predict the trends of the stock prices of Apple and Yahoo companies with a machine learning model. eXtreme Gradient Boosting (XGBoost) model was used to this end. An accuracy rate of 87% for periods of 60 and 90 days in the models was attained. It was stated that significantly more successful results were reached in comparison with other models.

Jiao and Jakubowicz (2017) made a study to forecast the direction of the S&P 500 shares. He used various macroeconomic data such as the interest rate. Various machine learning models along with artificial neural networks and gradient enhancement methods were utilized. ML methods provided better results than others in predicting the S&P 500 index.

4. DATASET AND PERFORMANCE MEASUREMENTS

4.1. Dataset

Data related to BIST100 and BIST30 index stock values, S&P 500, 1-year and 10-year T-bills, CDS, USD/TRY, EUR/TRY, Brent petrol, gold between 04.01.2016-31.12.2020 are used for the study.

Stock Prices

In the study, daily closing prices of companies traded in the BIST 30 index at the beginning of 2019 were used. Daily closing prices are taken from investing.com website. The companies included in the BIST 30 index at the beginning of 2019 are given below.

Table 4.1: List of the stocks traded in BİST 30 index as of 01.01.2020

Stock Code	**Company Name**
AKBNK	Akbank T.A.Ş
ARCLK	Arçelik A.Ş
ASELS	Aselsan Elektronik Sanayi ve Ticaret A.Ş
BIMAS	Bim Birleşik Mağazalar A.Ş.
DOHOL	Doğan Şirketler Grubu Holding A.Ş.
EKGYO	Emlak Konut Gayrimenkul Yatırım Ortaklığı A.Ş
ENKAI	Enka İnşaat ve Sanayi A.Ş
EREGL	Ereğli Demir ve Çelik Fabrikaları T.A.Ş.
FROTO	Ford Otomotiv Sanayii A.Ş.
GARAN	T. Garanti Bankası A.Ş
HALKB	Türkiye Halk Bankası A.Ş.
ISCTR	T. İş Bankası A.Ş.

KCHOL	Koç Holding A.Ş.
KOZAA	Koza Anadolu Metal Madencilik İşletmeleri A.Ş.
KOZAL	Koza Altın İşletmeleri A.Ş.
KRDMD	Kardemir Karabük Demir Çelik Sanayi ve Ticaret A.Ş.
PETKM	Petkim Petrokimya Holding A.Ş.
PGSUS	Pegasus Hava Taşımacılığı A.Ş.
SAHOL	Hacı Ömer Sabancı Holding A.Ş.
SISE	T. Şişe ve Cam Fabrikaları A.Ş.
SODA	Soda Sanayii A.Ş.
TAVHL	Tav Havalimanları Holding A.Ş.
TCELL	Turkcell İletişim Hizmetleri A.Ş.
THYAO	Türk Hava Yolları A.O.
TKFEN	Tekfen Holding A.Ş.
TOASO	Tofaş Türk Otomobil Fabrikası A.Ş.
TTKOM	Türk Telekomünikasyon A.Ş.
TUPRS	Tüpraş -Türkiye Petrol Rafinerileri A.Ş.
VAKBN	Türkiye Vakıflar Bankası T.A.O
YKBNK	Yapı ve Kredi Bankası A.Ş.

BIST 100 Index

Borsa İstanbul 100 index (BIST100) is an index consisting of 100 stocks traded in the Turkish capital market. It is capitalization-weighted, which means that the overall index performance is weighted by the company valuation of each stock. Companies traded on the BIST Stars and the BIST Main markets, as well as real estate investment trusts and venture capital trusts tradable on Collective and Structured Products-Market are included in BIST100. BIST100 index covers the stocks in BIST50 and BIST30.

BIST 30 Index

BIST30 index contains the 30 stocks with the largest market value and trading volume in BIST 100. The stocks listed in the BİST 30 index are reviewed quarterly. The index includes 7 banking, 23 non-banking stocks.

Data related to BIST100 and BIST30 indices are retrieved as to 02.01.2020, the date of the start of the testing period.

S&P 500 Index

It is the stock index introduced by Standard & Poor's. The index consists of the 500 largest companies in the U.S, covering approximately 80% of the U.S. stocks. Therefore, it is a convenient economic indicator of the performance of the US market.

Turkish Treasury Bills and Government Bonds

Treasury bills and government bonds are domestic government debt securities issued by the Turkish Treasury. Treasury bills have a maturity of less than 1 year and government bonds have a maturity of longer than 1 year. Government bonds are included in the risk-free investment category, therefore their yields are lower than corporate bonds.

Benchmark bonds are government bonds used as a reference in measuring the performance of other bonds. It is the highest traded bond remaining 2 years to maturity in the secondary market with the highest liquidity. The interest rate of this bond is acknowledged as the market interest rate.

CDS

Credit Default Swap (CDS) is a financial instrument that eliminates the risk of default of a loan. A country's CDS rate insures a loan to be used in that country and is an indicator of the country's risk premium for foreign investors. Turkey 5-year-CDS is included in the data set.

USD/TRY

The US dollar is a convertible international payment instrument in the world. Governments, including Turkey, hold USD as the foreign reserve currency in their central banks. For the study, USD/TRY exchange rate is included in the data set.

EUR/USD

The euro is the second most traded currency in the world. The EUR/USD covers European economies as well as American, making up more than half of the Forex trading volume. EUR/USD exchange rate is included in the data set of the study.

Brent Crude

Brent Crude is the most traded oil, accounting for approximately two-thirds of the world's traded crude oil supplies. Hence, it is the most important oil benchmark. It is included in the data set in terms of USD rate.

Gold

Gold is one of the most important of the central banks. Its price increases during periods of uncertainty in financial markets. Data regarding gold is included in terms of USD rate.

4.2. Performance Measurements

4.2.1. Performance Measurements of ML models

Performance measurements commonly used in the literature were used in comparing the ML models. These measurements include Mean Absolute Error (MAE), Mean Square Error (MSE) and Root Mean Square Error (RMSE).

4.2.1.1. Mean Absolute Error (MAE)

The Mean absolute error represents the average of the absolute difference between the actual and predicted values in the dataset. It measures the average of the residuals in the dataset.

$$MAE = \frac{1}{n}\sum_{j=1}^{n} |e_j|$$

4.2.1.2. Mean Squared Error (MSE)

Mean Squared Error represents the average of the squared difference between the original and predicted values in the data set. It measures the variance of the residuals.

$$MSE = \frac{1}{n}\sum_{j=1}^{n} e_j^2$$

4.2.1.3. Root Mean Squared Error (RMSE)

Root Mean Squared Error is the square root of Mean Squared error. It measures the standard deviation of residuals.

$$RMSE = \sqrt{\frac{\sum_{j=1}^{n} e_j^2}{n}}$$

$$RMSE = \sqrt{MSE}$$

4.2.2. Performance Measurement of Portfolios

In this study Sharpe ratio was used while comparing the performances of the portfolios.

4.2.2.1. Sharpe Ratio

The Sharpe Ratio is a financial measurement used while comparing the performances of the portfolios. It represents the risk-adjusted return or the excess return per unit of risk. While finding the excess return, risk-free rate is subtracted from the return of the portfolio. Generally, the Treasury bond's rate is used as a risk-free rate. The standard deviation of a portfolio's excess return helps to understand the volatility of the portfolio.

$$Sharpe\ Ratio = \frac{R_p - R_f}{\sigma_p}$$

The portfolio with a higher sharpe ratio is considered a better portfolio while making comparison. This means for the same risk, a portfolio with higher excess return or for the same excess return a portfolio with lower risk should be selected.

From Sharpe ratio perspective it is not a wise decision to add a financial product to a portfolio due to only its additional return if it comes with too much additional risk. Generally speaking, a sharpe ratio of 1.00 and above can be interpreted as successful but it would be good to compare a portfolio with its peers.

Even if it is widely used in the financial market today, Sharpe ratio also has some drawbacks. One of them is that, due to using standard deviation in the formula, it assumes that the return is distributed normally which is not always the case in the financial market.

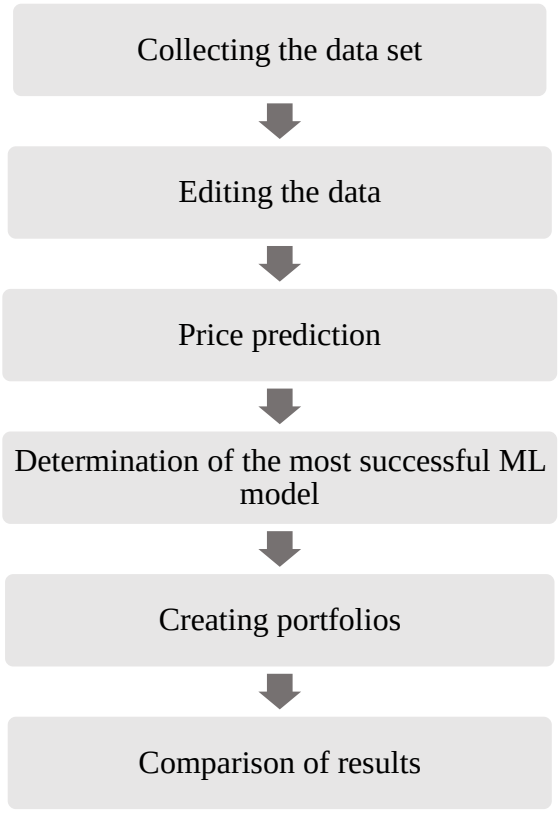
5. APPLICATION ON BIST 30 STOCKS

5.1. Method

The purpose of this study is to predict the prices of stocks that make up the BIST 30 index using ML algorithms and to create various portfolios according to these price estimates. In the first stage, price predictions made with five ML algorithms were compared and the most successful ML model was determined. In the second stage, various portfolios have been created by using the price predictions made in the previous stage. As a result, the returns of the portfolios are compared with each other and the performance of the BIST-30 index.

Python programming language was generally used in the study. Pandas library for data editing, scikit-learn and tensorflow libraries for price prediction and Matplotlib library for visualizing the results, were used.

The method of this study can be depicted as follows:



5.2. Data Preparation

In this study, daily data of the stocks in the BIST 30 index between 4 January 2016 and 31 December 2020 (1256 days) were used. As is known, BIST 30 index is updated periodically. In the study, the BIST 30 index on 2 January 2020 was taken as a basis and the data of 30 stocks included in the index as of the mentioned date were used. Daily price data were retrieved from the Matrix data terminal.

In addition to daily stock data, other economic indicators were also used in the study. In this regard, 5-year closing data of BIST100 and BIST30 index stock values, S&P 500, 1-year and 10-year T-bills, USD / TRY, EUR / TRY, Brent oil and gold were also added to the data set. Closing prices of these financial instruments were retrieved from investing.com website.

It is possible to predict the price of a share based solely on historical data, but in today's global financial markets, the prices of stocks and national indices are affected by many factors, some of which are mentioned above. It was considered that the inclusion of these factors in the data set would enable better price predictions of ML algorithms.

However, especially the addition of foreign financial data to the data set brought about some data incompatibilities, for the reason that the opening days of BIST and foreign financial markets differ. Therefore, some arrangements were made in the data set. Data from global markets were consolidated according to the dates Turkish markets were open. To this end, global market data on dates when Turkish markets were closed were eliminated from the data set. Accordingly, for the dates when Turkish markets were open and global markets were closed, the immediate preceding global market data were added to the data set.

5.3. Predicting Prices by Using ML Models

In the study, the data between 4 January 2016 and 31 December 2019 (1003 days) were used as the train set, and the data between 2 January 2020 and 31 December 2020 (253 days) were used as the test set. After ML algorithms were trained with 4

years of data, 1-2-3-4-6-12-month price predictions were made for each share as of 2 January 2020.

While making price predictions, 5 ML algorithms were used, including linear regression, decision tree, support vector machines, long-short term memory (LSTM) and XGBoost. As it is known, development levels and the parameters used by each algorithm differ from each other. In the current study, the default parameters of the algorithms were used and no tuning was performed.

Below are the graphs of the one-month-price forecast of AKBNK stock by using 5 ML algorithms.

Table 5.1: AKBNK- One-month price prediction with Linear Regression

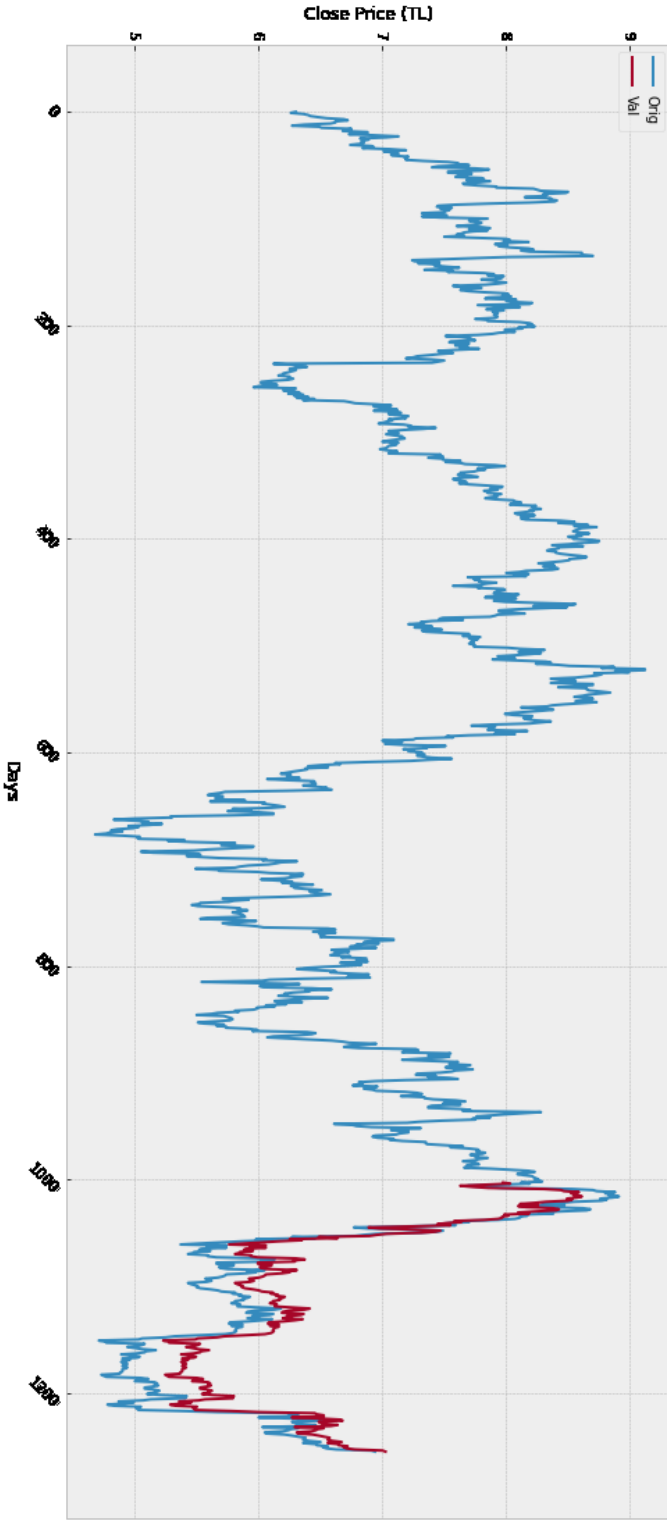


Table 5.2: AKBNK- One month price prediction with Decision Tree

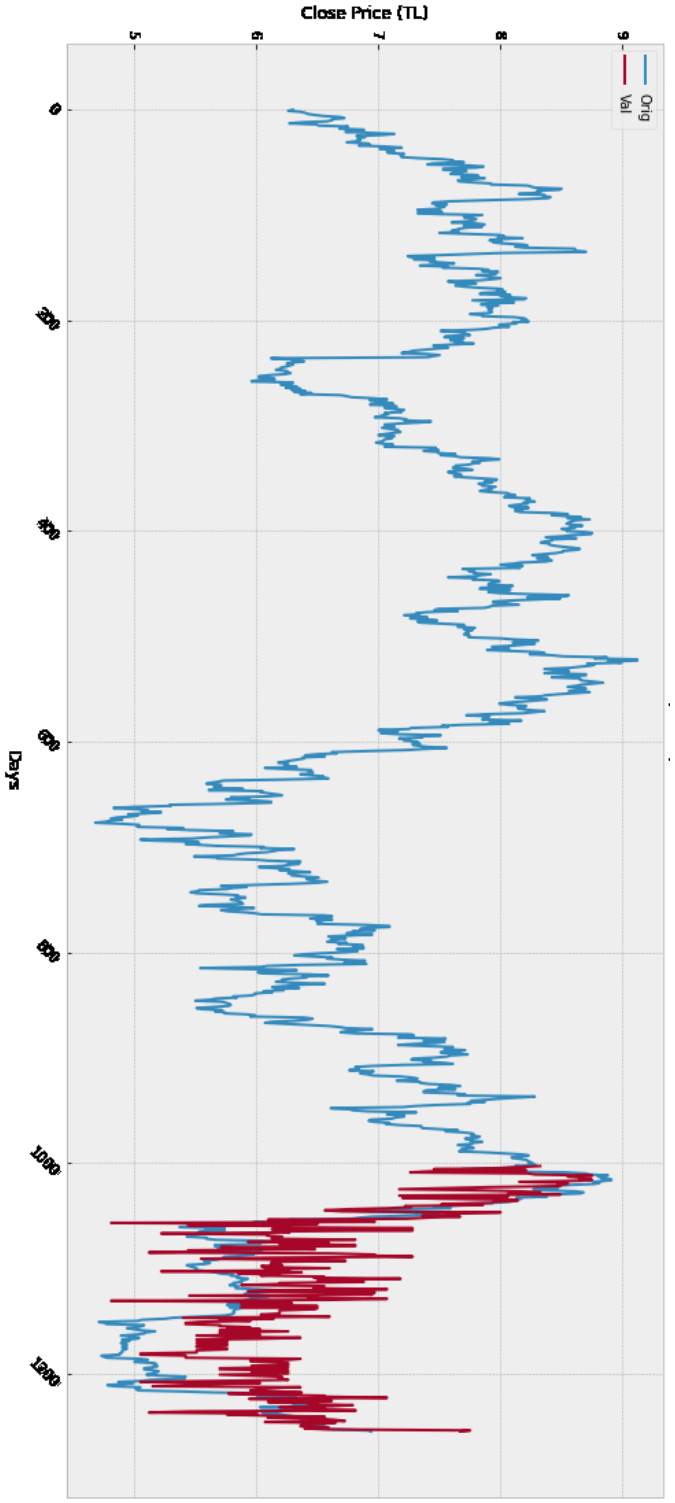


Table 5.3: AKBNK - One month price prediction with SVR

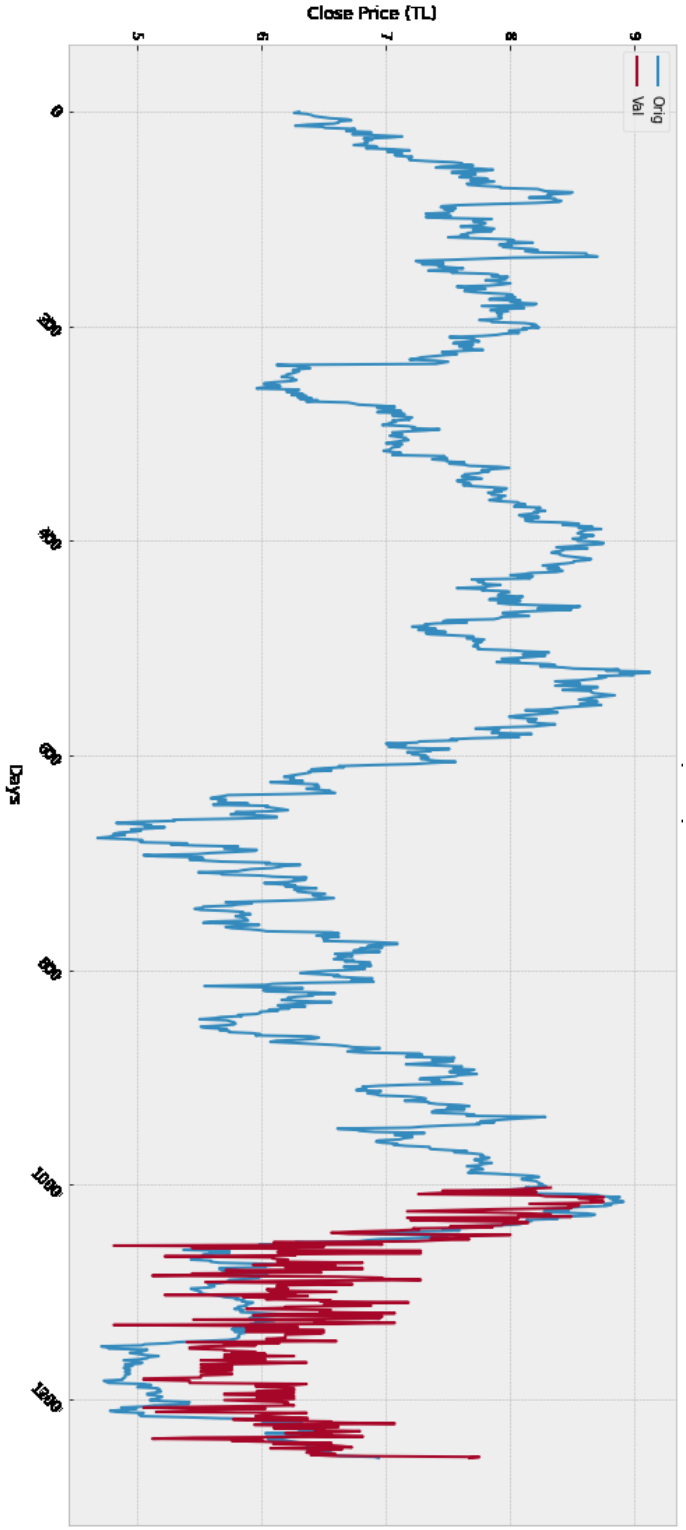


Table 5.4: AKBNK - One month price prediction with LSTM

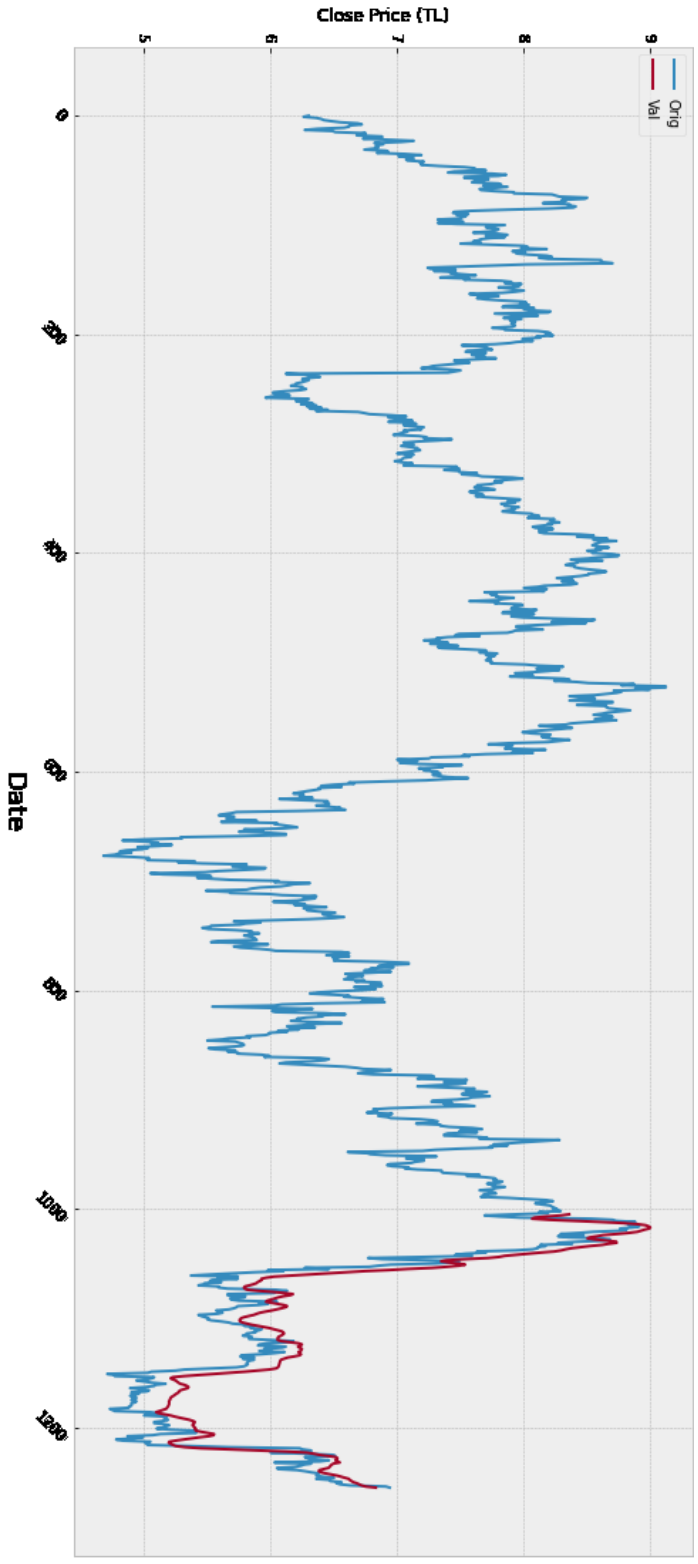
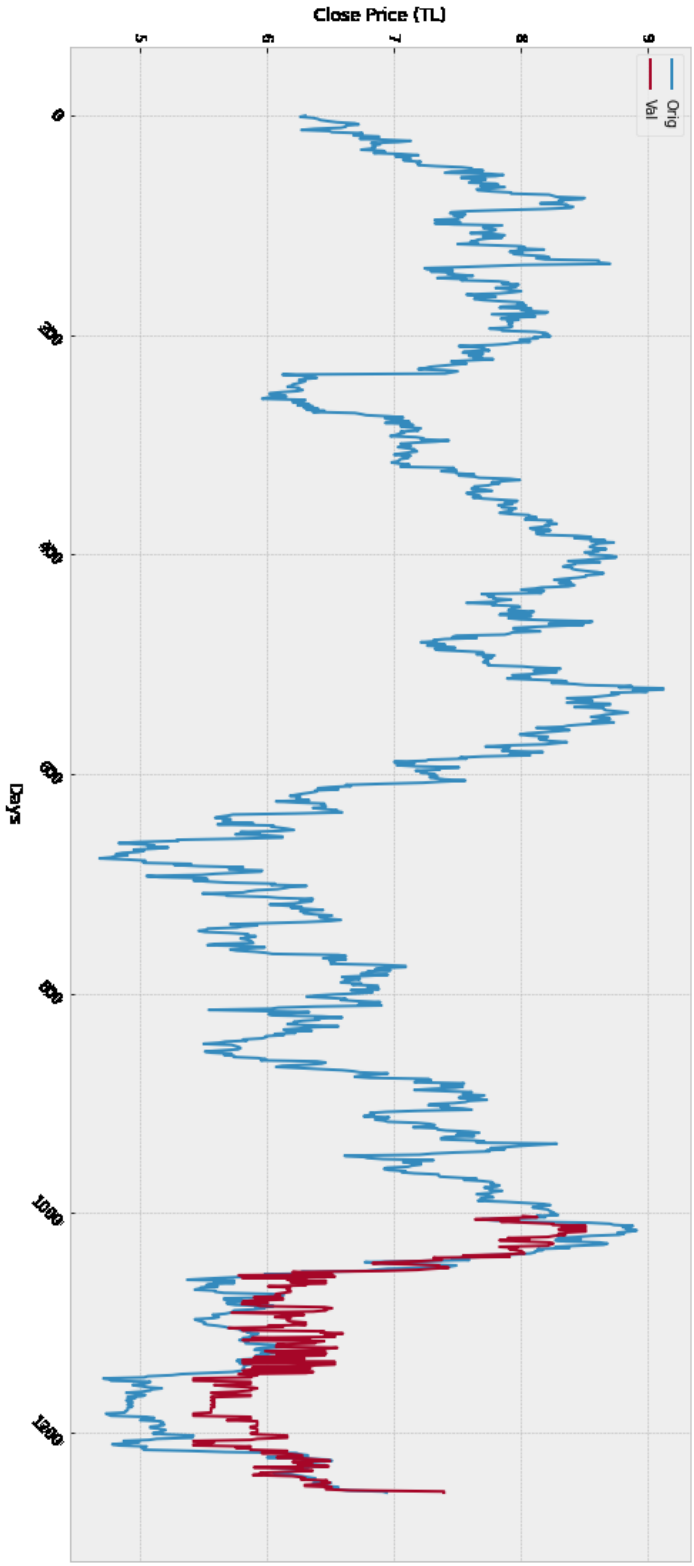


Table 5.5: AKBNK - One month price prediction with XGBoost



After making a price estimation using 4-year data, the estimates were compared with real prices and the accuracy rates of the algorithms were calculated. While finding the accuracy rates, MAE, MSE, RMSE, MAPE were used as criteria. For instance, while making a monthly price estimation, MAE, MSE, RMSE, MAPE values of 30 shares were calculated separately and the arithmetic mean was determined.

The table below shows the accuracy of the price predictions made by the algorithms for different time frames. Examining the table, it will be seen that the LSTM algorithm makes the best price estimation for all time frames.

Table 5.6: Accuracy rates of the 1-month price predictions for each algorithm

Timeframe	**Algorithm**	**MAE**	**MSE**	**RMSE**
1-month	Linear Regression	0,491	0,340	0,583
1-month	Decision tree	0,587	0,566	0,752
1-month	SVR	0,492	0,363	0,602
1-month	**LSTM**	**0,188**	**0,072**	**0,268**
1-month	XGBoost	0,486	0,360	0,600

Table 5.7: Accuracy rates of the 2-month price predictions for each algorithm

Timeframe	**Algorithm**	**MAE**	**MSE**	**RMSE**
2-month	Linear Regression	0,626	0,644	0,802
2-month	Decision tree	0,707	0,825	0,908
2-month	SVR	0,520	0,478	0,691
2-month	**LSTM**	**0,245**	**0,097**	**0,311**
2-month	XGBoost	0,617	0,630	0,794

Table 5.8: Accuracy rates of the 3-month price predictions for each algorithm

Timeframe	**Algorithm**	**MAE**	**MSE**	**RMSE**
3-month	Linear Regression	0,720	0,783	0,885
3-month	Decision tree	0,749	0,952	0,976
3-month	SVR	0,657	0,782	0,884
3-month	**LSTM**	**0,207**	**0,086**	**0,293**
3-month	XGBoost	0,666	0,705	0,840

Table 5.9: Accuracy rates of the 4-month price predictions for each algorithm

Timeframe	**Algorithm**	**MAE**	**MSE**	**RMSE**
4-month	Linear Regression	0,773	0,860	0,927
4-month	Decision tree	0,827	1,113	1,055
4-month	SVR	0,682	0,839	0,916
4-month	**LSTM**	**0,209**	**0,078**	**0,279**
4-month	XGBoost	0,740	0,899	0,948

Table 5.10: Accuracy rates of the 6-month price predictions for each algorithm

Timeframe	**Algorithm**	**MAE**	**MSE**	**RMSE**
6-month	Linear Regression	0,853	0,983	0,991
6-month	Decision tree	0,955	1,346	1,160
6-month	SVR	0,745	0,886	0,941
6-month	**LSTM**	**0,204**	**0,075**	**0,273**
6-month	XGBoost	0,880	1,065	1,032

Table 5.11: Accuracy rates of the 12-month price predictions for each algorithm

Timeframe	**Algorithm**	**MAE**	**MSE**	**RMSE**
12-month	Linear Regression	0,950	1,220	1,104
12-month	Decision tree	1,168	2,113	1,454
12-month	SVR	0,921	1,224	1,106
12-month	**LSTM**	**0,205**	**0,076**	**0,275**
12-month	XGBoost	0,900	1,182	1,087

5.4. Building Portfolios

For the study, stock portfolios were created and revised every 1-2-3-4-6-12 months. The stocks that were expected to show the best performance at the end of each period were included in the portfolio. Shares were equally weighted while being added to the portfolio. According to Kaczmarek and Perez's study (2020), most study in literature focuses on equal-weighted portfolios as well.

In addition, the number of shares in the portfolios was switched (5-10-15) for the said periods and the effect of this approach on the portfolio return was measured. As a result, a total of 90 portfolios using 5 different algorithms, of 3 different sizes, to be revised in 6 different time frames were created.

Below is an explanation of how to calculate the annual return of a portfolio that includes 5 stocks, which is revised each month and whose price is estimated by linear regression.

First of all, the linear regression algorithm was trained with 4-year data of the stocks, and a monthly price prediction (the price on 31 January 2020) for each stock in BIST 30 was made on December 31, 2019. Afterwards,, according to these estimates, the expected return of each stock was found and the stocks were sorted in descending order according to the expected return.

Table 5.12: Expected and actual return of BIST 30 stocks (ordered by expected return)

Stock	Date	Close	21 days prediction	Expected return (%)	21 days actual	Actual return (%)	MSE	RMSE	MAE
ASELS	Dec 31, 2019	10,41	12,67	21,71	11,66	12,01	5,084	2,255	1,358
VAKBN	Dec 31, 2019	5,51	6,48	17,66	6,6	19,78	0,230	0,479	0,374
HALKB	Dec 31, 2019	5,91	6,85	15,83	6,92	17,09	0,629	0,793	0,648
KRDMD	Dec 31, 2019	2,66	3,06	14,99	2,82	6,02	0,068	0,261	0,200
YKBNK	Dec 31, 2019	2,48	2,80	12,97	2,97	19,76	0,058	0,241	0,179
GARAN	Dec 31, 2019	11,14	12,42	11,50	11,9	6,82	0,391	0,625	0,482
ISCTR	Dec 31, 2019	6,41	7,14	11,43	7,25	13,10	0,167	0,408	0,302
SAHOL	Dec 31, 2019	9,54	10,49	9,99	9,75	2,20	0,284	0,533	0,413
TSKB	Dec 31, 2019	1,21	1,32	9,28	1,37	13,22	0,005	0,071	0,054
KOZAA	Dec 31, 2019	10,03	10,92	8,89	10,59	5,58	0,587	0,766	0,584
MGROS	Dec 31, 2019	24,22	26,35	8,77	25,2	4,05	2,797	1,672	1,266

EKGYO	Dec 31, 2019	1,46	1,59	8,67	1,67	14,38	0,021	0,145	0,119
AKBNK	Dec 31, 2019	8,11	8,81	8,63	8,26	1,85	0,182	0,427	0,343
TTKOM	Dec 31, 2019	7,37	7,99	8,36	7,75	5,16	0,176	0,420	0,338
PETKM	Dec 31, 2019	3,175	3,43	7,94	3,358	5,76	0,124	0,353	0,278
EREGL	Dec 31, 2019	9,04	9,74	7,70	9,33	3,21	0,260	0,510	0,422
ARCLK	Dec 31, 2019	20,84	22,37	7,34	20,9	0,29	1,358	1,165	0,936
KOZAL	Dec 31, 2019	74,05	79,37	7,19	80,15	8,24	13,329	3,651	2,860
TCELL	Dec 31, 2019	13,8	14,72	6,70	14,1	2,17	0,626	0,791	0,617
DOHOL	Dec 31, 2019	1,84	1,95	6,17	2,04	10,87	0,010	0,100	0,078
KCHOL	Dec 31, 2019	20,32	21,42	5,43	19,54	-3,84	0,836	0,915	0,750
THYAO	Dec 31, 2019	14,46	15,23	5,30	13,68	-5,39	1,492	1,221	0,909
ENKAI	Dec 31, 2019	5,582	5,79	3,72	6,13	9,82	0,056	0,237	0,181

OYAKC	Dec 31, 2019	5,92	6,08	2,67	5,47	-7,60	0,087	0,295	0,191
TAVHL	Dec 31, 2019	29,18	29,18	0,01	27,24	-6,65	2,136	1,461	1,133
TUPRS	Dec 31, 2019	126,8	126,47	-0,26	112,6	-11,20	32,286	5,682	4,286
BIMAS	Dec 31, 2019	46,66	46,15	-1,09	48,78	4,54	25,618	5,061	3,157
PGSUS	Dec 31, 2019	86,4	85,46	-1,09	68,6	-20,60	16,293	4,036	3,066
TKFEN	Dec 31, 2019	19,32	19,05	-1,41	19,25	-0,36	1,595	1,263	0,982
SISE	Dec 31, 2019	5,27	5,16	-2,03	5,56	5,50	0,098	0,312	0,236

Later, since it was decided to have 5 shares in the portfolio, the first 5 shares with the highest expected return were included in the portfolio and kept until 31 January 2020. As of this date the closing prices of the shares in the portfolio have been checked and the real return of the portfolio has been calculated.

Table 5.13: The average return of the portfolio for January 2020

Algorithm	**Time**	**Portfolio**	**Average Return (%)**
Linear Regression	January 2020	['ASELS', 'VAKBN', 'HALKB', 'KRDMD', 'YKBNK']	14,93

The process mentioned above was repeated on a monthly basis for 12 months and 12 portfolios were created.

Table 5.14: The monthly average return of the portfolio for the year 2020

Algorithm	Time	Portfolio	Average Return (%)
Lineer Regression	January 2020	['ASELS', 'VAKBN', 'HALKB', 'KRDMD', 'YKBNK']	14,93
Lineer Regression	February 2020	['ASELS', 'HALKB', 'AKBNK', 'VAKBN', 'YKBNK']	-11,42
Lineer Regression	March 2020	['YKBNK', 'AKBNK', 'HALKB', 'GARAN', 'TAVHL']	-16,75
Lineer Regression	April 2020	['YKBNK', 'AKBNK', 'TAVHL', 'TUPRS', 'ARCLK']	13,52
Lineer Regression	May 2020	['AKBNK', 'YKBNK', 'TAVHL', 'GARAN', 'TSKB']	3,84
Lineer Regression	June 2020	['AKBNK', 'YKBNK', 'GARAN', 'TAVHL', 'HALKB']	7,22
Lineer Regression	July 2020	['AKBNK', 'YKBNK', 'GARAN', 'TAVHL', 'VAKBN']	-14,60
Lineer Regression	August 2020	['AKBNK', 'YKBNK', 'GARAN', 'TAVHL', 'VAKBN']	-6,94
Lineer Regression	September 2020	['AKBNK', 'YKBNK', 'GARAN', 'TAVHL', 'VAKBN']	3,95
Lineer Regression	October 2020	['AKBNK', 'YKBNK', 'GARAN', 'TAVHL', 'VAKBN']	-6,39
Lineer Regression	November 2020	['AKBNK', 'GARAN', 'YKBNK', 'TAVHL', 'VAKBN']	29,73
Lineer Regression	December 2020	['AKBNK', 'VAKBN', 'HALKB', 'GARAN', 'YKBNK']	9,80

Then, the annual return of this strategy was calculated by summing the returns of these 12 portfolios.

Table 5.15: The annual return of the portfolio in 2020

Algorithm	Strategy type	# of Stocks	Average Return (%)	Annual Return (%)	Std dev	Sharpe
Lineer Regression	1-month	5	2,24	26,87	13,296	0,168

5.5. Comparison of Results

The above mentioned portfolio creation process was repeated 90 times by changing the algorithm, revision period and portfolio size. As a result, the following table was created.

Table 5.16: The annual returns of 90 portfolios

Portfolio	Algorithm	Strategy type	# of Stocks	Average Return (%)	Annual Return (%)	Std dev	Sharpe
1	Linear Regression	1-month	5	2,72	32,59	12,964	0,209
2	Linear Regression	1-month	10	3,59	43,04	11,682	0,307
3	Linear Regression	1-month	15	3,33	39,98	11,073	0,301
4	Linear Regression	2-month	5	11,60	69,62	18,166	0,639
5	Linear Regression	2-month	10	7,39	44,37	16,402	0,451
6	Linear Regression	2-month	15	7,68	46,07	17,051	0,450
7	Linear Regression	3-month	5	12,88	51,52	23,361	0,551
8	Linear Regression	3-month	10	9,31	37,26	21,222	0,439

9	Linear Regression	3-month	15	10,08	40,31	21,694	0,465
10	Linear Regression	4-month	5	15,59	46,77	10,159	1,535
11	Linear Regression	4-month	10	15,29	45,86	12,443	1,228
12	Linear Regression	4-month	15	13,54	40,63	13,398	1,011
13	Linear Regression	6-month	5	28,26	56,52	8,130	3,476
14	Linear Regression	6-month	10	28,34	56,68	2,424	11,692
15	Linear Regression	6-month	15	24,68	49,37	1,471	16,779
16	**Linear Regression**	**1-year**	**5**	**75,65**	**75,65**	--	--
17	Linear Regression	1-year	10	66,02	66,02	--	--
18	Linear Regression	1-year	15	56,65	56,65	--	--
19	Decision Tree	1-month	5	3,21	38,46	14,784	0,217
20	Decision Tree	1-month	10	2,45	29,45	12,510	0,196
21	Decision Tree	1-month	15	3,33	40,00	11,593	0,288
22	Decision Tree	2-month	5	5,53	33,15	17,172	0,322
23	Decision Tree	2-month	10	4,71	28,29	18,593	0,254

24	Decision Tree	2-month	15	5,59	33,55	18,850	0,297
25	Decision Tree	3-month	5	12,96	51,83	22,756	0,569
26	Decision Tree	3-month	10	12,87	51,48	20,346	0,633
27	Decision Tree	3-month	15	11,85	47,42	20,029	0,592
28	Decision Tree	4-month	5	16,00	48,00	7,587	2,109
29	Decision Tree	4-month	10	16,85	50,54	14,076	1,197
30	Decision Tree	4-month	15	15,11	45,33	15,875	0,952
31	Decision Tree	6-month	5	30,75	61,50	16,018	1,920
32	Decision Tree	6-month	10	27,73	55,47	4,189	6,621
33	Decision Tree	6-month	15	26,05	52,11	1,541	16,903
34	**Decision Tree**	**1-year**	**5**	**80,25**	**80,25**	**--**	**--**
35	Decision Tree	1-year	10	66,69	66,69	--	--
36	Decision Tree	1-year	15	57,57	57,57	--	--
37	SVM	1-month	5	4,36	52,31	12,684	0,344
38	SVM	1-month	10	2,99	35,85	12,282	0,243

39	SVM	1-month	15	3,47	41,70	11,061	0,314
40	SVM	2-month	5	6,49	38,95	19,844	0,327
41	SVM	2-month	10	5,10	30,59	18,576	0,274
42	SVM	2-month	15	4,14	24,82	17,669	0,234
43	SVM	3-month	5	11,95	47,78	25,754	0,464
44	SVM	3-month	10	9,21	36,84	23,751	0,388
45	SVM	3-month	15	10,30	41,22	24,236	0,425
46	SVM	4-month	5	13,04	39,12	15,293	0,853
47	SVM	4-month	10	11,49	34,47	18,534	0,620
48	SVM	4-month	15	10,23	30,69	17,271	0,592
49	SVM	6-month	5	18,39	36,78	4,505	4,082
50	SVM	6-month	10	16,46	32,92	0,382	43,087
51	SVM	6-month	15	14,85	29,71	0,395	37,646
52	**SVM**	**1-year**	**5**	**58,02**	**58,02**	**--**	**--**
53	SVM	1-year	10	48,96	48,96	--	--

54	SVM	1-year	15	41,07	41,07	--	--
55	XGBoost	1-month	5	2,03	24,39	13,739	0,148
56	XGBoost	1-month	10	2,02	24,24	13,224	0,153
57	XGBoost	1-month	15	2,46	29,55	12,696	0,194
58	XGBoost	2-month	5	6,49	38,95	21,252	0,305
59	XGBoost	2-month	10	5,63	33,79	19,491	0,289
60	XGBoost	2-month	15	6,40	38,41	19,314	0,331
61	XGBoost	3-month	5	16,98	67,91	23,310	0,728
62	XGBoost	3-month	10	13,78	55,12	21,802	0,632
63	XGBoost	3-month	15	14,29	57,17	22,730	0,629
64	XGBoost	4-month	5	15,01	45,04	8,529	1,760
65	XGBoost	4-month	10	14,98	44,95	11,379	1,317
66	XGBoost	4-month	15	15,34	46,02	17,988	0,853
67	XGBoost	6-month	5	30,21	60,42	16,557	1,825
68	XGBoost	6-month	10	25,51	51,02	6,384	3,995

69	XGBoost	6-month	15	22,79	45,58	1,597	14,273
70	**XGBoost**	**1-year**	**5**	**80,26**	**80,26**	**--**	**--**
71	XGBoost	1-year	10	66,81	66,81	--	--
72	XGBoost	1-year	15	57,57	57,57	--	--
73	**LSTM**	**1-month**	**5**	**4,50**	**53,99**	**9,983**	**0,451**
74	LSTM	1-month	10	4,40	52,86	10,214	0,431
75	LSTM	1-month	15	3,55	42,66	11,315	0,314
76	LSTM	2-month	5	7,66	45,94	22,339	0,343
77	LSTM	2-month	10	6,23	37,37	20,500	0,304
78	LSTM	2-month	15	5,69	34,16	19,557	0,291
79	LSTM	3-month	5	10,34	41,34	21,516	0,480
80	LSTM	3-month	10	10,78	43,10	22,823	0,472
81	LSTM	3-month	15	11,48	45,94	24,650	0,466
82	LSTM	4-month	5	10,10	30,31	18,716	0,540
83	LSTM	4-month	10	10,23	30,68	18,661	0,548

84	LSTM	4-month	15	11,60	34,81	20,367	0,570
85	LSTM	6-month	5	9,14	18,28	1,153	7,928
86	LSTM	6-month	10	9,01	18,02	2,743	2,803
87	LSTM	6-month	15	10,72	21,44	1,832	5,851
88	LSTM	1-year	5	32,31	32,31	--	--
89	LSTM	1-year	10	29,31	29,31	--	--
90	LSTM	1-year	15	27,63	27,63	--	--

The following inferences could be reached through examining the table.

i) In the period from 2 January 2020 to 31 December 2020, during which the portfolios were created, the yield of BIST 30 was 17.8%. The table demonstrates that the lowest return is 18.02%. In other words, the return of any portfolio made and actively managed using the ML algorithm was higher than the return of the passively managed portfolio. Active has defeated passive management.

ii) Considering the revision periods of the strategies, it is observed that the portfolios revised in 1 year are generally more profitable compared to other periods. Average returns for the revision periods are depicted below.

Table 5.17: Comparison table according to revision period

Strategy type	# of portfolio	Min Annual Return (%)	Max Annual Return (%)	Average Annual Return (%)
1-month	15	24,24	53,99	38,74
2-month	15	24,82	69,62	38,53
3-month	15	36,84	67,91	47,75
4-month	15	30,31	50,54	40,88
6-month	15	15,37	61,5	42,88
1-year	**15**	**27,63**	**80,26**	**56,32**

iii) Exploring the number of shares in the portfolio, it is observed that portfolios with 5 shares are mostly more profitable than others. Below are the average returns for the number of shares.

Table 5.18: Comparison table according to number of stocks in the portfolio

# of Stocks	# of Portfolio	Min Annual Return (%)	Max Annual Return (%)	Average Annual Return (%)
5	**30**	**18,28**	**80,26**	**48,6**
10	30	15,37	66,81	42,65
15	30	21,44	57,57	41,3

iv) The most successful portfolio for the period from 2 January 2020 to 31 December 2020 has a 80.26% rate of return and is the one using the algorithm XGBoost, revised in 1-year and containing 5 shares.

6. CONCLUSION

In this study, the prices of the shares in the BIST 30 index were estimated using ML algorithms and various portfolios were created according to these price estimates. Afterwards, the returns of the portfolios were calculated and compared for the 2019-2020 period. In addition, by looking into the yield of the BIST 30 index in the same period, a comparison between active and passive investments was made to see which one was more successful.

As a result of the study, it was observed that the LSTM algorithm made more successful predictions, portfolios revised in 1-year had better returns and portfolios with 5 stocks yielded more successful results. In addition, it has been observed that the returns of all the created portfolios were higher than the 17.8% return of the BIST 30 index in the same period, in other words, active investment defeated passive investment.

Looking into the literature regarding the applications of ML in financial investment, it is seen that ML algorithms are generally used in price estimation and weighting in the portfolio building process. In the studies on price estimation, national indices were generally used instead of stocks, and the next day opening values or directions were predicted. Studies that make relatively long-term price estimates of 3-6-12 months for stocks are limited. There are even fewer studies that utilize the method of portfolio creation based on price estimates. No study of the same type concerning BIST has been found. The study is expected to fill this gap in the literature.

Although the portfolios in which ML algorithms were used gave a better result than the BIST 30 index, a limitation that may have affected the portfolio returns and price prediction accuracy is worth mentioning. There was the pandemic of COVID-19 during the test period of 2019-2020. As a result, some of the stocks fell sharply in the first half of 2020. Since ML algorithms have been learning during the 4 years up to the beginning of 2020, it is possible to say that they were less successful than expected in their predictions during the test period and this affected their portfolio returns negatively.

In addition, certain assumptions were made during the study. In future studies, better performance can be achieved by changing these assumptions. For example, the default values of the algorithms were used in the study. Better results can be obtained by tuning these values with methods such as GridSearchCV. In addition, as mentioned above, there was a pandemic during the test period and therefore the price prediction accuracy was negatively affected. Cross Validation method can be applied to minimize this effect. Finally, it was assumed in the study that there is zero market impact and zero slippage. More successful comparisons can be made by using real trade volumes in the study. It can be useful to change these assumptions for comparison of portfolios in future studies.

REFERENCES

Agrawal, Prachi (2021). [Linear Regression]. Retrieved May 2, 2021, from https://medium.com/analytics-vidhya/linear-regression-6b642119533a

Arslan, M. (2005). A Tipi yatırım Fonlarında Yöneticilerin Zamanlama Kabiliyeti ve Performans İlişkisi Analizi: 2002-2005 Dönemi Bir Uygulama. *Ticaret ve Turizm Eğitim Fakültesi Dergisi (2)*, 1-23.

Atsalakis, G. S., & Valavanis, K. P. (2009). Forecasting stock market short-term trends using a neuro-fuzzy based methodology. *Expert Systems with Applications*, *36*(7), 10696–10707. https://doi.org/10.1016/j.eswa.2009.02.043

Avcı, E. (2009). Stock Return Forecasts With Artificial Neural Network Models. *Marmara Üniversitesi İ.İ.B.F. Dergisi (C.XXVI)*1, 443 - 461.

Brentani, S. (2004). *Portfolio management in practice: Essential capital markets*. Elsevier.

Baturynska, I. (2021). *Gradient Boosting* [Image]. ResearchGate. https://www.researchgate.net/publication/340524896_Prediction_of_geometry_deviations_in_additive_manufactured_parts_comparison_of_linear_regression_with_machine_learning_algorithms

Chandra, P. (2017). *Investment Analysis and Portfolio Management*. (5th ed.). McGraw-Hill Education.

Cox, C. (2017). *A comparison of active and passive portfolio management* [Dissertation, University of Tennessee]. https://trace.tennessee.edu/cgi/viewcontent.cgi?article=3080&context=utk_chanhonoproj

Dey, S., Kumar, Y., Saha, S. & Basak, S. (2016). *Forecasting to classification: Predicting the direction of stock market price using Xtreme Gradient Boosting*. https://doi.org/10.13140/RG.2.2.15294.48968

Emerson, S., Kennedy, R., O'Shea, L., & O'Brien, J. (2019). Trends and Applications of Machine Learning in Quantitative Finance. *8th International Conference on Economics and Finance Research (ICEFR 2019)* Retrieved April 22, 2021, from https://ssrn.com/abstract=3397005

Erkartal, R. B (2018). *Forecasting İstanbul stock exchange* (Publication No. 514536) [Master's Thesis, Yeditepe University]. https://tez.yok.gov.tr/UlusalTezMerkezi/tezDetay.jsp?id=lTIusjqMb0soYLI3TxuESQ&no=I4N_hkZBcFq4TMbAp2SIOw

Gu, S., Kelly, B. & Xiu, D. (2020). Empirical asset pricing via machine learning. *The Review of Financial Studies, 33*(5), 2223–2273. https://doi.org/10.1093/rfs/hhaa009

Gökgöz, F. & Günel, M.O. (2012). Türk yatırım fonlarının portföy performanslarının analizi. *Ankyra: Ankara Üniversitesi Sosyal Bilimler Enstitüsü Dergisi 3(2),* 9-16. https://doi.org/10.1501/sbeder_0000000043

Gümüş, M. & Kıran, M. S. (2017). Crude oil price forecasting using XGBoost. *2017 International Conference on Computer Science and Engineering (UBMK)*, 1100-1103, https://doi.org/ 10.1109/UBMK.2017.8093500.

Gündüz, H., Çataltepe, Z. & Yaslan, Y. (2017). Stock daily return prediction using expanded features and feature selection. T*urkish Journal of Electrical Engineering & Computer Sciences*, 25, 4829 - 4840.

Hatipoğlu, B., & Uyar, U. (2018). Bayes ağ modelleri ile hisse senedi getirilerinin karşılıklı dinamik ilişkileri. *International Journal of Economic and Administrative Studies, 18. EYİ Özel Sayısı,* 709-720.

Heaton, J. B., Polson, N. G., & Witte, J. H. (2017). Deep learning for finance: deep portfolios. *Applied Stochastic Models Bus. Industry, 33*(1), 3–12. https://doi.org/10.1002/asmb.2209.

Jain, P. & Jain, S. (2019). Can machine learning-based portfolios outperform traditional risk-based portfolios? The need to account for covariance misspecification. *Risks, 7*(3). https://doi.org/10.3390/risks7030074

Javatpoint (n.d.). [Linear Regression]. Retrieved May 2, 2021, from https://www.javatpoint.com/linear-regression-in-machine-learning

Jiao, Y. & Jakubowicz, J. (2017). Predicting stock movement direction with machine learning: an extensive study on S&P 500 stocks. *IEEE International Conference on Big Data*.

Jiang, Z., Xu, D. & Liang, J. (2017). A deep reinforcement learning framework for the financial portfolio management problem, Retrieved April 18, 2021, from https://arxiv.org/abs/1706.10059

Kaczmarek, T. & Perez, K. (2020). Building portfolios based on machine learning predictions *Economic Research-Ekonomska Istraživanja.* https://doi.org/10.1080/1331677X.2021.1875865

Kara, Y., Boyacioglu, M. A. & Baykan, Ö. K. (2011). Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the Istanbul Stock Exchange. *Expert Systems with Applications, 38,* 5311 – 5319.

Kara, İ., & Ecer, F. (2018). BİST endeks hareket yönünün tahmininde sınıflandırma yöntemlerinin performanslarının karşılaştırılması. *The Journal of Academic Social Science, 6* (83), 514-524

Kavitha, S., Varuna, S. & Ramya, R. (2016). A comparative analysis on linear regression and support vector regression. *2016 Online International Conference on Green Engineering and Technologies (IC-GET),* 1-5, https://doi.org/10.1109/GET.2016.7916627.

Kevin, S. (2015). *Security analysis and portfolio management.* (2nd ed.). PHI Learning Pvt. Ltd.

Kılıç, S. B., Paksoy, S., & Genç, T. (2014). Forecasting the direction of BİST 100 returns with artificial neural network models. *International Journal Of Latest Trends İn Finance And Economic Sciences, 4*(3).759 - 765.

Kinn, D. D. (2018) Reducing estimation risk in mean-variance portfolios with machine learning, Retrieved May 3, 2021, from https://arxiv.org/abs/1804.01764v2

Okur, Y. (2009). *Kar payı verimliliği çerçevesinde aktif portföy yönetimi ve İMKB uygulaması,* [Master's thesis, Dokuz Eylül University]. https://acikerisim.deu.edu.tr/xmlui/handle/20.500.12397/10989?show=full

Osisanwo, F.Y., Akinsola, J.E.T., Awodele, O., Hinmikaiye, J. O. , Olakanmi, O. & Akinjobi, J. (2017). Supervised Machine Learning Algorithms: Classification and Comparison. *International Journal of Computer Trends*

and Technology (IJCTT), 48. 128 - 138. https://doi.org/10.14445/22312803/IJCTT-V48P126

Programmersought (n.d.). [Integrated model XGBoost]. Retrieved May 2, 2021, from https://www.programmersought.com/article/46524622557/

Terzi, S. & Saldanlı, A. (2019). Yatırımcılar için uygun zamanlama stratejileri; BİST 30'daki hisse senetleri üzerine bir uygulama, *Uluslararası Ekonomi İşletme ve Politika Dergisi, 3(2), 253-270. https://doi.org/10.29216/ueip.599211*

Uyar, U. (2019). Makine öğrenmesi ile portföy optimizasyonu: FTSE, DAX ve BİST uygulamaları. *23. Finans Sempozyumu*, 161-175.

Zengin, E. (2006). *Hisse senedi portföylerinin yönetiminde pratik yaklaşımlar ve İMKB uygulaması* (Publication No. 189922), [Master's thesis, Dokuz Eylül University]. https://tez.yok.gov.tr/UlusalTezMerkezi/tezDetay.jsp?id=ZZAHFklH_tqXSD7bxojqhA&no=4U1roCJ6UUzN6t-DufsScA

APPENDIXES

Appendix 1: Codes to build data set (excluding stock data)

```
df=pd.read_csv(data_path_name[0])
data = {'Date':df.Date[::-1]}
df_final = pd.DataFrame(data=data)
df_final.index=[i for i in range(len(df))]

for i in data_path_name:

    column_name= i.split('\\')[-1][:-20]

    df=pd.read_csv(i)
    data = {'Date':df.Date[::-1], column_name:df.Price[::-1]}
    dff = pd.DataFrame(data=data)

    df_final = pd.merge(df_final,dff, on='Date', how='left')
    df_final = df_final.rename(columns={'Brent Oil Futures':'Brent','Turkey 1-Year
Bond Yield':'1Y Bond','Turkey 10-Year Bond Yield':'10Y Bond'})

df_final = df_final.ffill()

df_final['BIST 100'] = df_final['BIST 100'].apply(lambda x: x.replace(',',''))
df_final['BIST 30'] = df_final['BIST 30'].apply(lambda x: x.replace(',',''))
df_final['S&P 500'] = df_final['S&P 500'].apply(lambda x: x.replace(',',''))
df_final['XAU_USD'] = df_final['XAU_USD'].apply(lambda x: x.replace(',',''))

df_final['BIST 100'] = df_final['BIST 100'].astype('float64')
df_final['BIST 30'] = df_final['BIST 30'].astype('float64')
df_final['S&P 500'] = df_final['S&P 500'].astype('float64')
df_final['XAU_USD'] = df_final['XAU_USD'].astype('float64')
# df_final.dtypes
# df_final.to_excel("tum_veriler.xlsx")
df_final
```

Appendix 2: Application for Portfolio Results

```
result = pd.DataFrame(columns=['Algorithm', 'Strategy_type', '# of Stocks','Average Return(%)','Annual Return(%)','Std dev','Sharpe'])

# algorithm = 'Linear Regression'
# algorithm = 'Decision Tree'
# algorithm = 'SVM'
# algorithm = 'XGBoost'
algorithm='LTSM'
intervals = [21,42,63,84,126,252]
portfolio_size = [5, 10, 15]


for interval in intervals:

    x_days_prediction = str(interval)+' days prediction'
    x_days_actual = str(interval)+' days actual'

    first_month = pd.DataFrame(columns=['Date','Close',x_days_prediction,'Expected return(%)',x_days_actual,'Actual return(%)','Stock'])
    second_month = pd.DataFrame(columns=['Date','Close',x_days_prediction,'Expected return(%)',x_days_actual,'Actual return(%)','Stock'])
    third_month = pd.DataFrame(columns=['Date','Close',x_days_prediction,'Expected return(%)',x_days_actual,'Actual return(%)','Stock'])
    fourth_month = pd.DataFrame(columns=['Date','Close',x_days_prediction,'Expected return(%)',x_days_actual,'Actual return(%)','Stock'])
    fifth_month = pd.DataFrame(columns=['Date','Close',x_days_prediction,'Expected return(%)',x_days_actual,'Actual return(%)','Stock'])
    sixth_month = pd.DataFrame(columns=['Date','Close',x_days_prediction,'Expected return(%)',x_days_actual,'Actual return(%)','Stock'])
    seventh_month = pd.DataFrame(columns=['Date','Close',x_days_prediction,'Expected return(%)',x_days_actual,'Actual return(%)','Stock'])
    eighth_month = pd.DataFrame(columns=['Date','Close',x_days_prediction,'Expected return(%)',x_days_actual,'Actual return(%)','Stock'])
```

```
    ninth_month                                                    =
pd.DataFrame(columns=['Date','Close',x_days_prediction,'Expected
return(%)',x_days_actual,'Actual return(%)','Stock'])
    tenth_month                                                    =
pd.DataFrame(columns=['Date','Close',x_days_prediction,'Expected
return(%)',x_days_actual,'Actual return(%)','Stock'])
    eleventh_month                                                 =
pd.DataFrame(columns=['Date','Close',x_days_prediction,'Expected
return(%)',x_days_actual,'Actual return(%)','Stock'])
    twelfth_month                                                  =
pd.DataFrame(columns=['Date','Close',x_days_prediction,'Expected
return(%)',x_days_actual,'Actual return(%)','Stock'])

    for i in file_path_name:

        stock_name = i.split('\\')[-1].split(' ')[0]
        stock=pd.read_csv(i)

        data = {'Date':stock.Date[::-1], 'Close':stock.Price[::-1]}
        dff = pd.DataFrame(data=data)

#         df = dff[['Close']]
#         future_days=int(interval)
#         df['Prediction']=df[['Close']].shift(-future_days)
#         X=np.array(df[['Close']])[:-253]
#         y=np.array(df['Prediction'])[:-253]

        dff = pd.merge(dff,df_final, on='Date', how='left')
        dff_new = dff
        dff_new = dff.drop('Date', axis=1)
        future_days=int(interval)
        dff_new['Prediction']=dff_new[['Close']].shift(-future_days)
        X=np.array(dff_new)[:-253,:-1]
        y=np.array(dff_new['Prediction'])[:-253]

        x_train,x_test,y_train,y_test=train_test_split(X,y,test_size=0.20)
        # x_train=X
        # y_train=y

#         lr=LinearRegression().fit(x_train,y_train)
#         y_pred=lr.predict(x_test)

#         tree=DecisionTreeRegressor().fit(x_train,y_train)
#         y_pred=tree.predict(x_test)

#         svr_rbf = SVR(kernel='rbf', C=1e3, gamma=0.1)
```

```
#       svr_rbf.fit(x_train, y_train)
#       y_pred=svr_rbf.predict(x_test)

      model = xgb.XGBRegressor(learning_rate=0.1, max_depth=3,
min_samples_split=2, n_estimators=100, subsample=0.6)
      model.fit(x_train, y_train)
      y_pred=model.predict(x_test)

      # r2':metrics.r2_score(y_test,y_pred)
      prediction_rates={'mse':mean_squared_error(y_test,y_pred),
                  'rmse':np.sqrt(mean_squared_error(y_test,y_pred)),
                  'mae':mean_absolute_error(y_test,y_pred)}


#       x_future=np.array(df[['Close']])[-253:]
#       y_future=lr.predict(x_future)
#       valid=dff[X.shape[0]:]
#       prediction_name = str(future_days)+' days prediction'
#       valid[prediction_name]=y_future

      x_future=dff_new.iloc[-253:,:-1]
#       y_future=lr.predict(x_future)
#       y_future=tree.predict(x_future)
#       y_future=svr_rbf.predict(x_future)
      y_future=model.predict(x_future)
      valid=dff[['Date','Close']][X.shape[0]:]
      prediction_name = str(future_days)+' days prediction'
      valid[prediction_name]=y_future

      ########################

#       stock_name = i.split('\\')[-1].split(' ')[0]
#       stock=pd.read_csv(i)

#       data = {'Date':stock.Date[::-1], 'Close':stock.Price[::-1]}
#       dff = pd.DataFrame(data=data)
#       dff.index=[i for i in range(len(dff))]

#       data=dff.filter(['Close'])
#       dataset=data.values
# #       training_data_len=math.ceil(len(dataset)*0.8)
#       training_data_len=990

#       future_days=int(interval)

#       scaler=MinMaxScaler(feature_range=(0,1))
```

```
#       scaled_data=scaler.fit_transform(dataset)

#       train_data=scaled_data[0:training_data_len,:]
#       x_train=[]
#       y_train=[]

#       for i in range(future_days, len(train_data)):
#          x_train.append(train_data[i-future_days:i,0])
#          y_train.append(train_data[i,0])

#       x_train,y_train=np.array(x_train),np.array(y_train)
#       x_train=np.reshape(x_train,(x_train.shape[0],x_train.shape[1],1))

#       model=Sequential()
#                                   model.add(LSTM(50,       return_sequences=True,
input_shape=(x_train.shape[1],1)))
#       model.add(LSTM(50, return_sequences=False))
#       model.add(Dense(25))
#       model.add(Dense(1))

#       model.compile(optimizer='adam',loss='mean_squared_error')

#       model.fit(x_train, y_train, batch_size=1, epochs=1)

#       test_data=scaled_data[training_data_len-future_days:,:]
#       x_test=[]
#       y_test=dataset[training_data_len:,:]

#       for i in range(future_days, len(test_data)):
#          x_test.append(test_data[i-future_days:i,0])

#       x_test=np.array(x_test)
#       x_test=np.reshape(x_test,(x_test.shape[0],x_test.shape[1],1))

#       y_pred=model.predict(x_test)
#       y_pred=scaler.inverse_transform(y_pred)

#       prediction_rates_lstm={'r2':r2_score(y_test,y_pred),
#                   'mse':mean_squared_error(y_test,y_pred),
#                   'rmse':np.sqrt(mean_squared_error(y_test,y_pred)),
#                   'mae':mean_absolute_error(y_test,y_pred)}

#       train=data[:training_data_len]
#       valid=data[training_data_len:]
#       prediction_name = str(future_days)+' days prediction'
#       valid[prediction_name]=y_pred
```

```
#       valid['Date']=dff['Date'][training_data_len:]

# #       x_future=dff_new.iloc[-253:,:-1]
# #       y_future=model.predict(x_future)
# #       valid=dff[['Date','Close']][X.shape[0]:]
# #       prediction_name = str(future_days)+' days prediction'
# #       valid[prediction_name]=y_future

#########################

        first_month_index=valid[valid['Date']=='Dec 31, 2019'].index[0]
        second_month_index=valid[valid['Date']=='Jan 31, 2020'].index[0]
        third_month_index=valid[valid['Date']=='Feb 28, 2020'].index[0]
        fourth_month_index=valid[valid['Date']=='Mar 31, 2020'].index[0]
        fifth_month_index=valid[valid['Date']=='Apr 30, 2020'].index[0]
        sixth_month_index=valid[valid['Date']=='May 29, 2020'].index[0]
        seventh_month_index=valid[valid['Date']=='Jun 30, 2020'].index[0]
        eighth_month_index=valid[valid['Date']=='Jul 30, 2020'].index[0]
        ninth_month_index=valid[valid['Date']=='Aug 31, 2020'].index[0]
        tenth_month_index=valid[valid['Date']=='Sep 30, 2020'].index[0]
        eleventh_month_index=valid[valid['Date']=='Oct 30, 2020'].index[0]
        twelfth_month_index=valid[valid['Date']=='Nov 30, 2020'].index[0]
        last_index=valid[valid['Date']=='Dec 31, 2020'].index[0]

        if interval==21:

            strategy_type = '1-month'

            list1 = [first_month_index,
              second_month_index,
              third_month_index,
              fourth_month_index,
              fifth_month_index,
              sixth_month_index,
              seventh_month_index,
              eighth_month_index,
              ninth_month_index,
              tenth_month_index,
              eleventh_month_index,
              twelfth_month_index,
              last_index]

            return_table = valid.loc[list1,:]
            return_table['Expected return(%)'] = 100*((return_table['21 days
prediction']-return_table['Close'])/return_table['Close'])
            return_table['21 days actual'] = return_table['Close'].shift(-1)
```

```
        return_table['Actual return(%)'] = 100*((return_table['21 days actual']-
return_table['Close'])/return_table['Close'])
        return_table['Stock'] = stock_name

        first_month                                                    =
pd.concat([first_month,return_table.loc[[first_month_index],:]],axis=0)
        second_month                                                   =
pd.concat([second_month,return_table.loc[[second_month_index],:]],axis=0)
        third_month                                                    =
pd.concat([third_month,return_table.loc[[third_month_index],:]],axis=0)
        fourth_month                                                   =
pd.concat([fourth_month,return_table.loc[[fourth_month_index],:]],axis=0)
        fifth_month                                                    =
pd.concat([fifth_month,return_table.loc[[fifth_month_index],:]],axis=0)
        sixth_month                                                    =
pd.concat([sixth_month,return_table.loc[[sixth_month_index],:]],axis=0)
        seventh_month                                                  =
pd.concat([seventh_month,return_table.loc[[seventh_month_index],:]],axis=0)
        eighth_month                                                   =
pd.concat([eighth_month,return_table.loc[[eighth_month_index],:]],axis=0)
        ninth_month                                                    =
pd.concat([ninth_month,return_table.loc[[ninth_month_index],:]],axis=0)
        tenth_month                                                    =
pd.concat([tenth_month,return_table.loc[[tenth_month_index],:]],axis=0)
        eleventh_month                                                 =
pd.concat([eleventh_month,return_table.loc[[eleventh_month_index],:]],axis=0)
        twelfth_month                                                  =
pd.concat([twelfth_month,return_table.loc[[twelfth_month_index],:]],axis=0)

        month_list=[first_month,
                    second_month,
                    third_month,
                    fourth_month,
                    fifth_month,
                    sixth_month,
                    seventh_month,
                    eighth_month,
                    ninth_month,
                    tenth_month,
                    eleventh_month,
                    twelfth_month]

    elif interval==42:

        strategy_type = '2-month'
```

```
            list2 = [first_month_index,
              third_month_index,
              fifth_month_index,
              seventh_month_index,
              ninth_month_index,
              eleventh_month_index,
              last_index]

            return_table = valid.loc[list2,:]
            return_table['Expected return(%)'] = 100*((return_table['42 days
prediction']-return_table['Close'])/return_table['Close'])
            return_table['42 days actual'] = return_table['Close'].shift(-1)
            return_table['Actual return(%)'] = 100*((return_table['42 days actual']-
return_table['Close'])/return_table['Close'])
            return_table['MSE'] = prediction_rates['mse']
            return_table['RMSE'] = prediction_rates['rmse']
            return_table['MAE'] = prediction_rates['mae']
            return_table['Stock'] = stock_name

            first_month                                                    =
pd.concat([first_month,return_table.loc[[first_month_index],:]],axis=0)
            third_month                                                    =
pd.concat([third_month,return_table.loc[[third_month_index],:]],axis=0)
            fifth_month                                                    =
pd.concat([fifth_month,return_table.loc[[fifth_month_index],:]],axis=0)
            seventh_month                                                  =
pd.concat([seventh_month,return_table.loc[[seventh_month_index],:]],axis=0)
            ninth_month                                                    =
pd.concat([ninth_month,return_table.loc[[ninth_month_index],:]],axis=0)
            eleventh_month                                                 =
pd.concat([eleventh_month,return_table.loc[[eleventh_month_index],:]],axis=0)

            month_list=[first_month,
                    third_month,
                    fifth_month,
                    seventh_month,
                    ninth_month,
                    eleventh_month]

        elif interval==63:

            strategy_type = '3-month'

            list3 = [first_month_index,
              fourth_month_index,
              seventh_month_index,
```

```
            tenth_month_index,
            last_index]

        return_table = valid.loc[list3,:]
        return_table['Expected return(%)'] = 100*((return_table['63 days prediction']-return_table['Close'])/return_table['Close'])
        return_table['63 days actual'] = return_table['Close'].shift(-1)
        return_table['Actual return(%)'] = 100*((return_table['63 days actual']-return_table['Close'])/return_table['Close'])
        return_table['Stock'] = stock_name

        first_month = pd.concat([first_month,return_table.loc[[first_month_index],:]],axis=0)
        fourth_month = pd.concat([fourth_month,return_table.loc[[fourth_month_index],:]],axis=0)
        seventh_month = pd.concat([seventh_month,return_table.loc[[seventh_month_index],:]],axis=0)
        tenth_month = pd.concat([tenth_month,return_table.loc[[tenth_month_index],:]],axis=0)

        month_list=[first_month,
                fourth_month,
                seventh_month,
                tenth_month]

    elif interval==84:

        strategy_type = '4-month'

        list4 = [first_month_index,
          fifth_month_index,
          ninth_month_index,
          last_index]

        return_table = valid.loc[list4,:]
        return_table['Expected return(%)'] = 100*((return_table['84 days prediction']-return_table['Close'])/return_table['Close'])
        return_table['84 days actual'] = return_table['Close'].shift(-1)
        return_table['Actual return(%)'] = 100*((return_table['84 days actual']-return_table['Close'])/return_table['Close'])
        return_table['MSE'] = prediction_rates['mse']
        return_table['RMSE'] = prediction_rates['rmse']
        return_table['MAE'] = prediction_rates['mae']
        return_table['Stock'] = stock_name
```

```
        first_month = pd.concat([first_month,return_table.loc[[first_month_index],:]],axis=0)
        fifth_month = pd.concat([fifth_month,return_table.loc[[fifth_month_index],:]],axis=0)
        ninth_month = pd.concat([ninth_month,return_table.loc[[ninth_month_index],:]],axis=0)

        month_list=[first_month,
                fifth_month,
                ninth_month]

    elif interval==126:

        strategy_type = '6-month'

        list5 = [first_month_index,
          seventh_month_index,
          last_index]

        return_table = valid.loc[list5,:]
        return_table['Expected return(%)'] = 100*((return_table['126 days prediction']-return_table['Close'])/return_table['Close'])
        return_table['126 days actual'] = return_table['Close'].shift(-1)
        return_table['Actual return(%)'] = 100*((return_table['126 days actual']-return_table['Close'])/return_table['Close'])
        return_table['Stock'] = stock_name

        first_month = pd.concat([first_month,return_table.loc[[first_month_index],:]],axis=0)
        seventh_month = pd.concat([seventh_month,return_table.loc[[seventh_month_index],:]],axis=0)

        month_list=[first_month,
                seventh_month]

    elif interval==252:

        strategy_type = '1-year'

        list6 = [first_month_index,
          last_index]

        return_table = valid.loc[list6,:]
        return_table['Expected return(%)'] = 100*((return_table['252 days prediction']-return_table['Close'])/return_table['Close'])
        return_table['252 days actual'] = return_table['Close'].shift(-1)
```

```
        return_table['Actual return(%)'] = 100*((return_table['252 days actual']-
return_table['Close'])/return_table['Close'])
        return_table['Stock'] = stock_name

        first_month =
pd.concat([first_month,return_table.loc[[first_month_index],:]],axis=0)

        month_list=[first_month]

    for size in portfolio_size:

      one_month_strategy = pd.DataFrame(columns=['Type', 'Portfolio', 'Average
Return(%)'])
      two_month_strategy = pd.DataFrame(columns=['Type', 'Portfolio', 'Average
Return(%)'])
      three_month_strategy = pd.DataFrame(columns=['Type', 'Portfolio', 'Average
Return(%)'])
      four_month_strategy = pd.DataFrame(columns=['Type', 'Portfolio', 'Average
Return(%)'])
      six_month_strategy = pd.DataFrame(columns=['Type', 'Portfolio', 'Average
Return(%)'])
      one_year_strategy = pd.DataFrame(columns=['Type', 'Portfolio', 'Average
Return(%)'])

      for i in range(len(month_list)):

        month_list[i] = month_list[i].sort_values(by=['Expected return(%)'],
ascending=False)
        portfolio = month_list[i][:size]
        stocks=[]
        for k in range(size):
          stock = portfolio['Stock'].iloc[k]
          stocks.append(stock)

        x_portfolio = pd.DataFrame(columns=['Type', 'Portfolio', 'Average
Return(%)'])
        x_portfolio['Type']=[strategy_type]
        x_portfolio['Portfolio']=[stocks]
        x_portfolio['Average Return(%)']=np.mean(portfolio['Actual return(%)'])

        if interval==21:
          one_month_strategy =
pd.concat([one_month_strategy,x_portfolio],axis=0)
        elif interval==42:
          two_month_strategy =
pd.concat([two_month_strategy,x_portfolio],axis=0)
```

```
        elif interval==63:
            three_month_strategy = pd.concat([three_month_strategy,x_portfolio],axis=0)
        elif interval==84:
            four_month_strategy = pd.concat([four_month_strategy,x_portfolio],axis=0)
        elif interval==126:
            six_month_strategy = pd.concat([six_month_strategy,x_portfolio],axis=0)
        elif interval==252:
            one_year_strategy = pd.concat([one_year_strategy,x_portfolio],axis=0)

    annual_results = pd.DataFrame(columns=['Algorithm', 'Strategy_type', '# of Stocks','Average Return(%)','Annual Return(%)','Std dev','Sharpe'])
    annual_results['Algorithm'] = [algorithm]
    annual_results['Strategy_type'] = [strategy_type]
    annual_results['# of Stocks'] = [size]

    if interval==21:
        annual_results['Annual Return(%)'] = np.sum(one_month_strategy['Average Return(%)'])
        annual_results['Average Return(%)'] = np.mean(one_month_strategy['Average Return(%)'])
        annual_results['Std dev'] = np.std(one_month_strategy['Average Return(%)'])
        annual_results['Sharpe'] = annual_results['Average Return(%)']/annual_results['Std dev']
    elif interval==42:
        annual_results['Annual Return(%)'] = np.sum(two_month_strategy['Average Return(%)'])
        annual_results['Average Return(%)'] = np.mean(two_month_strategy['Average Return(%)'])
        annual_results['Std dev'] = np.std(two_month_strategy['Average Return(%)'])
        annual_results['Sharpe'] = annual_results['Average Return(%)']/annual_results['Std dev']
    elif interval==63:
        annual_results['Annual Return(%)'] = np.sum(three_month_strategy['Average Return(%)'])
        annual_results['Average Return(%)'] = np.mean(three_month_strategy['Average Return(%)'])
        annual_results['Std dev'] = np.std(three_month_strategy['Average Return(%)'])
        annual_results['Sharpe'] = annual_results['Average Return(%)']/annual_results['Std dev']
    elif interval==84:
```

```
        annual_results['Annual Return(%)'] = np.sum(four_month_strategy['Average Return(%)'])
        annual_results['Average Return(%)'] = np.mean(four_month_strategy['Average Return(%)'])
        annual_results['Std dev'] = np.std(four_month_strategy['Average Return(%)'])
        annual_results['Sharpe'] = annual_results['Average Return(%)']/annual_results['Std dev']
    elif interval==126:
        annual_results['Annual Return(%)'] = np.sum(six_month_strategy['Average Return(%)'])
        annual_results['Average Return(%)'] = np.mean(six_month_strategy['Average Return(%)'])
        annual_results['Std dev'] = np.std(six_month_strategy['Average Return(%)'])
        annual_results['Sharpe'] = annual_results['Average Return(%)']/annual_results['Std dev']
    elif interval==252:
        annual_results['Annual Return(%)'] = np.sum(one_year_strategy['Average Return(%)'])
        annual_results['Average Return(%)'] = np.mean(one_year_strategy['Average Return(%)'])
        annual_results['Std dev'] = ['--']
        annual_results['Sharpe'] = ['--']

    result = pd.concat([result,annual_results],axis=0)
```

Appendix 3: Accuracy Rates of ML Algorithms

```
df_prediction_rates_linear =
pd.DataFrame(columns=['Algorithm','R2','MAE','MSE','RMSE'])
df_prediction_rates_linear['Algorithm'] = ['Linear Regression']
df_prediction_rates_linear['R2'] = [prediction_rates_linear['r2']]
df_prediction_rates_linear['MAE'] = [prediction_rates_linear['mae']]
df_prediction_rates_linear['MSE'] = [prediction_rates_linear['mse']]
df_prediction_rates_linear['RMSE'] = [prediction_rates_linear['rmse']]

df_prediction_rates_decision_tree =
pd.DataFrame(columns=['Algorithm','R2','MAE','MSE','RMSE'])
df_prediction_rates_decision_tree['Algorithm'] = ['Decision tree']
df_prediction_rates_decision_tree['R2'] = [prediction_rates_decision_tree['r2']]
df_prediction_rates_decision_tree['MAE'] =
[prediction_rates_decision_tree['mae']]
df_prediction_rates_decision_tree['MSE'] =
[prediction_rates_decision_tree['mse']]
df_prediction_rates_decision_tree['RMSE'] =
[prediction_rates_decision_tree['rmse']]

df_prediction_rates_svr =
pd.DataFrame(columns=['Algorithm','R2','MAE','MSE','RMSE'])
df_prediction_rates_svr['Algorithm'] = ['SVR']
df_prediction_rates_svr['R2'] = [prediction_rates_svr['r2']]
df_prediction_rates_svr['MAE'] = [prediction_rates_svr['mae']]
df_prediction_rates_svr['MSE'] = [prediction_rates_svr['mse']]
df_prediction_rates_svr['RMSE'] = [prediction_rates_svr['rmse']]

df_prediction_rates_lstm =
pd.DataFrame(columns=['Algorithm','R2','MAE','MSE','RMSE'])
df_prediction_rates_lstm['Algorithm'] = ['LSTM']
df_prediction_rates_lstm['R2'] = [prediction_rates_lstm['r2']]
df_prediction_rates_lstm['MAE'] = [prediction_rates_lstm['mae']]
df_prediction_rates_lstm['MSE'] = [prediction_rates_lstm['mse']]
df_prediction_rates_lstm['RMSE'] = [prediction_rates_lstm['rmse']]

df_prediction_rates_xgboost =
pd.DataFrame(columns=['Algorithm','R2','MAE','MSE','RMSE'])
df_prediction_rates_xgboost['Algorithm'] = ['XGBoost']
df_prediction_rates_xgboost['R2'] = [prediction_rates_xgboost['r2']]
df_prediction_rates_xgboost['MAE'] = [prediction_rates_xgboost['mae']]
df_prediction_rates_xgboost['MSE'] = [prediction_rates_xgboost['mse']]
df_prediction_rates_xgboost['RMSE'] = [prediction_rates_xgboost['rmse']]
```

```
prediction_rates = pd.concat([prediction_rates,df_prediction_rates_linear],axis=0)
prediction_rates =
pd.concat([prediction_rates,df_prediction_rates_decision_tree],axis=0)
prediction_rates = pd.concat([prediction_rates,df_prediction_rates_svr],axis=0)
prediction_rates = pd.concat([prediction_rates,df_prediction_rates_lstm],axis=0)
prediction_rates =
pd.concat([prediction_rates,df_prediction_rates_xgboost],axis=0)

prediction_rates.to_excel("12-month.xlsx")
prediction_rates
```