Faculty of
Mathematics
and Informatics

VILNIUS UNIVERSITY
FACULTY OF MATHEMATICS AND INFORMATICS
MODELLING AND DATA ANALYSIS
MASTER'S STUDY PROGRAMME

APPLICATIONS OF TIME-SERIES, AND MACHINE LEARNING MODELS IN CRYPTOCURRENCY MARKETS

Master's thesis

Author: Şükrü Yavuz
VU email address: sukru.yavuz@mif.stud.vu.lt
Supervisor: prof. dr. Igoris Belovas

Vilnius

2022

Abstract

In recent years, with the increase in the prevalence of cryptocurrencies, the interest in the problem of predicting cryptocurrencies has also started to increase. This study aims to predict Bitcoin's return using different machine learning and time series models. Different training data intervals, cross-validation, feature engineering, and various statistical tests are used to achieve high performance in each model. The training data is taken from Binance API. The results show that machine learning algorithms outperform time-series models. Machine learning algorithms blending helps to obtain even better results.

Keywords: time series, bitcoin, ARIMA, VAR, VARMA, SVM, LightGBM, XGBoost, Random Forest, Decision Tree, Adaboost, blending, time-series forecasting, machine learning, cryptocurrency, permutation feature importance, Extra Trees, cross-validation, feature selection

Santrauka

Pastaraisiais metais, didėjant kriptovaliutų paplitimui, ėmė didėti ir susidomėjimas kriptovaliutų prognozavimo problema. Šio tyrimo tikslas yra numatyti Bitcoin grąžą naudojant skirtingus mašininio mokymosi ir laiko eilučių modelius. Norint pasiekti aukštą kiekvieno modelio našumą, naudojami skirtingi mokymo intervalai, kryžminis patvirtinimas, funkcijų inžinerija ir įvairūs statistiniai testai. Empiriniai duomenys paimti iš Binance API. Rezultatai parodė, kad mašininio mokymosi algoritmai pranoksta laiko eilučių modelius. Be to, mašininio mokymosi algoritmų derinimas leidžia pasiekti dar geresnių rezultatų.

Raktiniai žodžiai: laiko eilutė, bitkoinas, ARIMA, VAR, VARMA, SVM, LightGBM, XGBoost, atsitiktinis miškas, sprendimų medis, Adaboost, maišymas, laiko eilučių prognozavimas, mašininis mokymasis, kriptovaliuta, permutacijos funkcijos svarba, papildomi medžiai, kryžminis patvirtinimas, funkcijų pasirinkimas

Contents

List of Figures

List of Tables

Introduction

In today's global world, where the economic borders between countries have disappeared, and social, cultural, and social values are constantly changing, "decision making" represents a challenging process for states, companies, and individuals from all segments of society. Because the continually changing conditions and the presence of ambiguous data negatively affect the decision-making process, this process involves the ability of investors to invest by making the right decision, or in other words, to make portfolio management correctly. For this reason, in the decision-making process, which is also crucial for investors, investors need to access information such as the trading volume, transaction amount, market value of the stock they invest in and predict the stock's future prices.

The future price prediction of stocks, which is essential for portfolio management, when done correctly, can help investors achieve higher profits than their current profits or reduce or eliminate their losses. Due to this importance, many studies have been carried out on stock future price prediction, and many studies have been carried out on stock price prediction, and many methods, such as Artificial Neural Networks (ANN), Autoregression (AR), Moving Average (MA), Autoregressive Moving Average (ARMA), Autoregressive Integrated Moving Average (ARIMA), Support Vector Machines (SVM), Linear Regression, Decision Trees, Random Forest, Markov Chains have been proposed [1–5].

Uncertainty, chaotic market movements, and the non-linear dynamic structure of financial time series make precise estimations very difficult. Moreover, the fact that stock market indices are influenced by many macroeconomic factors, such as political shifts, the general stance of the economy, investors' expectations, and investment preferences, makes index estimations both complicated and attractive. Thus, various factors should be considered, and an accurate estimation approach must be chosen. Under all these circumstances, approaching the stock price prediction problem with only one model seems inappropriate. In the present study, we employ and compare Machine Learning and Time-Series Models.

Approaching the forecasting problem with different models requires different ways to prepare the empirical data. First, in the world of machine learning, stock price prediction is a regression problem. Hence, to forecast with Machine Learning Algorithms, independent variables are necessary. Therefore, the first challenge of using Machine Learning algorithms is preparing the target variable, the stock's closing price, for each time interval. A one-step-ahead forecasting approach is used to deal with this problem. Yesterday's independent variables (open, high, low) aligned with the target variable of today's closing price. This strategy allows models to predict the price for tomorrow with today's input data. Other preprocessing techniques such as differencing, deseasonalizing, detrending, and combination of all three were tested and applied. To increase the performance of machine learning models, feature engineering is used. Details about generated features are given in chapter 3.3 on page 33. Generated features are tested with Permutation Feature Importance. Machine learning algorithms that are used in this study are Linear Regression, Lasso Regression, Ridge Regression, Huber Regressor, Support Vector Machines [7], K-Nearest Neighbors, Decision Trees [8], Random Forest [9], Extra Trees Regressor [10], Adaboost Regressor [11], Gradient Boosting Regressor [12], Extreme Gradient Boosting [13], Light Gradient Boosting [14], and Catboost Regressor [15]. In addition, Hyperparameter tuning [16] and Cross-Validation [17] techniques (with Hold-out sets) are used to ensure each model is robust and

performs at its best. The performance of Machine learning algorithms is evaluated with MAPE and RMSE. Best performance based on MAPE was achieved with Blended Machine learning Models. There are four different machine learning models under this blended model. These models are:

- Extra Trees
- Linear Regression
- Ridge Regression
- Huber Regression
- Adaboost Regression

The performance of Machine Learning algorithms is evaluated with Root Mean Squared Error (RMSE). The data preparation stage for time-series models starts with decomposing for further analysis such as trend, seasonality, and error, followed by Augmented Dickey-Fuller Test (ADF) for stationarity, Akaike Information Criterion (AIC) for determination of the parameters of Time Series algorithms. Both Univariate and Multivariate time-series methods are used in this study. For Univariate methods, the date and return of the stock; for Multivariate methods, opening price, minimum price, maximum price, and other generated features are used. These features are mentioned in 3.3 on page 33. Univariate time-series methods used in this study are ARIMA, Exponential Smoothing, Seasonal Naïve Forecaster, Naïve Forecaster, ETS. Multivariate time-series methods are Vector Autoregression (VAR) [60], Vector Autoregression Moving-Average (VARMA). In addition, Time-Series Cross-Validation is applied for this case. All forecasting methods are evaluated with RMSE. The best performing models are:

- Seasonal Naive Forecaster
- VAR

The importance of training data is also covered in this study. In addition to comparing machine learning and time series prediction models and combining their strengths in this study, the effect of the training data's date range on the prediction results will be investigated, the best results for the date range will be identified, and all models will be learned using this training data. While the machine learning approach covers how the skill was acquired, the most appropriate date range for the training data addresses where the skill was learned. Different time intervals as training data were used and tested. The mean and standard deviation MAPE were calculated for every model with 10-fold cross-validation during the testing of training data. The dataset used in this study is collected through Binance's Application Programming Interface (API) with 15-minute intervals, and the optimal date range is July 1st, 2021 – December 13th, 2021 for Bitcoin.

The rest of the workflow is as follows; in Chapter 1, literature research is illustrated by analyzing works related to this study's subject. The following chapter demonstrates definitions and theoretical concepts of Machine learning, Time Series Models in this study's practical section. Next, mentioned models' real-world application, evaluation, comparison, and combination are demonstrated with experiments of selecting the right training dataset in chapter 3. Finally, the conclusions are given in Chapter 4.

The flowchart of workflow is given below:

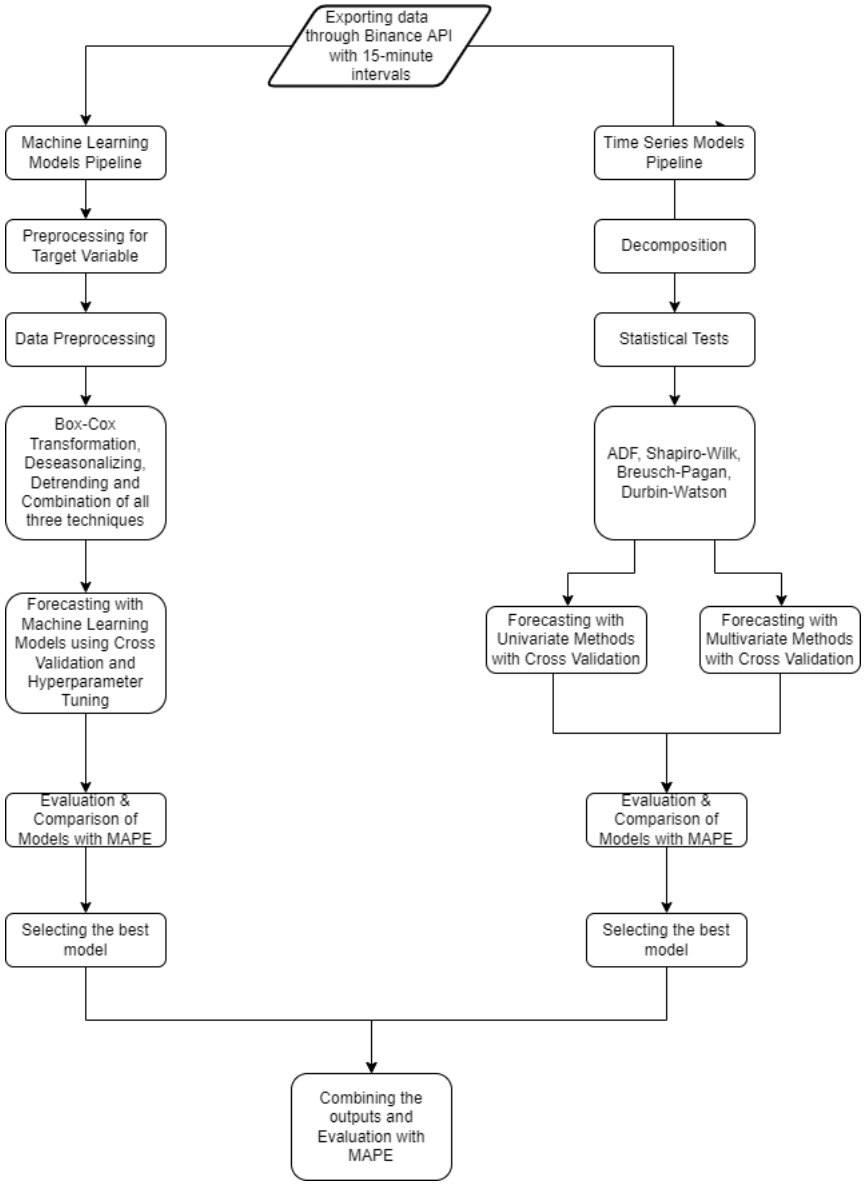


Figure 1: The flowchart of workflow

1 Literature Review

Financial markets, by their nature, can be closely associated with price volatility and the ability to predict prices. In this context, many econometric, statistical, and mathematical models are used in price predictions. However, when the studies in the literature on the prediction of all cryptocurrencies are examined, it is seen that the vast majority predict mostly Bitcoin prices and returns by using different methods and approaches.

Almeida et al. [20], using ANNs, tried to predict the future Bitcoin price using daily USD price for the next 730 steps equal to two years. Before predicting the price of Bitcoin, to avoid overfitting and increase the performance of ANNs, min-max scaling was applied. The artificial neural networks model was used in the study, and as a conclusion, it is mentioned that ANNs are quite successful in predicting the price of Bitcoin.

Amjad and Shah [21] predicted the movement of Bitcoin prices using classification algorithms and obtained more accurate results than the ARIMA technique. Three possible outcomes for the Bitcoin price were considered: an increase, a decrease, and the assumption that the price does not change. Rather than utilizing regression to solve Time Series issues with stationary data, the performance of classification techniques demonstrated that they should be used.

Sean McNally [22] aimed to determine how accurately the trend of Bitcoin price (in USD) can be predicted. In this context, the author used both classification and Time Series predictions. To evaluate the performance of the algorithms, K-Fold Cross Validation, RMSE, and classification metrics such as Accuracy, Sensitivity, Specificity, and Recall were used. For feature engineering, Simple Moving Average (SMA) indicator was used. To eliminate and reduce the number of features, the Boruta algorithm was used. Principal Component Analysis was also conducted, but it was not included in the final results. As a result of the study, they concluded that nonlinear Deep Learning methods outperform the estimation made by the ARIMA. Long Short-Term Memory (LSTM) was the best performing among all algorithms used.

Georgoula et al. [23] explored the factors of the Bitcoin price using Sentiment Analysis and Support Vector Machines (SVM). According to the authors, there is a positive correlation between Wikipedia traffic and the network's hash rate. On the other side, a negative link is discovered between the USD/EURO exchange rate and bitcoin price.

Madan et al. [24], utilized the Bitcoin exchange and payment data to develop an algorithmic trading system, with the issue handled as a classification problem. The authors separated the investigation into two phases: in phase one, they added 25 extra characteristics to boost classification algorithms' effectiveness, and they tested with various time intervals; in phase two, they just used the closing price and produced predictions. The first phase with data collected at ten-minute intervals was the preferred technique. Authors employed Binomial Generalized Linear Models (GLM), SVM, and Random Forest (RF).

Ji et al. [25] applied Deep Neural Networks (DNN), LSTM, Convolutional Neural Networks (CNN), ResNet, Convolutional Recurrent Neural Network (CRNN), and ensembling in their study. Both classification and regression models were used and evaluated. The results showed LSTM slightly outperformed other models for regression, DNN outperformed for classification. A comparison of those

models is made using a simplified trading strategy. The authors found out that classification models outperformed regression models.

Jang and Lee [26] aimed to reveal the effect of Bayesian neural networks (BNN) by analyzing the time series of Bitcoin prices. Additionally, they constructed models by extracting the most relevant elements from the Blockchain data underlying Bitcoin supply and demand. On the other hand, compared the Bayesian neural network to other linear and non-linear models in models for predicting the Bitcoin price time process and discovered that the BNN performs well in predicting the Bitcoin price time series and explaining the recent Bitcoin price's high volatility.

Lahmiri and Bekiros [27] used deep learning techniques to predict the price of the three most traded digital currencies, namely Bitcoin, Digital Cash, and Ripple, based on historical data. They concluded that deep learning is highly effective in predicting the natural chaotic dynamics of cryptocurrency markets.

Simon and Geetha [28] used linear regression and polynomial regression to predict the next block reward to the miner. The Long Short Term Memory (LSTM) algorithm was used to forecast the next ether price in the market.

Phaladisailoed and Numnonda [29] aimed to discover the most efficient and most accurate model for predicting Bitcoin prices within various Machine learning algorithms. They tried different regression models with Scikit-Learn and Keras libraries, using 1-minute interval trading data on the Bitcoin exchange website Bitsamp from January 1, 2012, to January 8, 2018. The best results showed Mean Squared Error (MSE) as low as 0.00002 and R-Squared (R2) as high as 99.2.

Jaquart et al. [30] compared six different Machine learning models which are trained on nine months of cryptocurrency-related data. They used RNNs and XGBoost that surpassed arbitrary classification. Furthermore, they employed blockchain-based and technical features for various time horizons from one to sixty minutes. After testing the importance of features, the authors observed that technical features are most relevant for the algorithms.

Xiaoleia et al. [31] made predictions with various economic indicators by using three different Machine learning methods: SVM, Random Forest, and LightGBM for 42 different features. In the study, daily data from 01.01.2018 - 06.30.2018 were used. Stock market indices of various countries, US Dollar Index and Crude Oil WTI Futures Price, were used as economic indicators. Forecasts are made to determine whether prices will increase or decrease, not prices. The methods used to give the best results for the medium term (2 weeks). In addition, more successful predictions have been obtained for more widely used cryptocurrencies. One of the study's essential findings is that LightGBM gives better resistance results than SVM and Random Forest and is an effectively uses method for many data and variables.

Hattori [32] made a representative volatility indicator for the Bitcoin market and modeled volatility using GARCH, GJR-GARCH, EGARCH, APARCH, and IGARCH models. The data obtained by the author in the 5-minute frequency range from January 2016 to July 2018 were used. In the study, "performed volatility," calculated by the squares of the daily return, is used as an indicator of volatility. The EGARCH and APARCH asymmetric volatility models provided more successful estimations. Normally distributed volatility models provided more successful results than models with thick-tailed distributions such as t-skewed.

Mallqui and Fernandes [40] aimed to determine the highest and lowest values, direction, and closing price of Bitcoin price in their study. The data used to belong to August 19, 2013 - July 19, 2016. The study used collective algorithms based on SVM, ANN and RNN, and K-Means clustering algorithms. The study used 80 – 20 (range 1) and 75 – 25 (range 2) rates as training and test data. For estimation according to the regression trials, the most successful results were obtained with SVM in predicting the largest, smallest, and closing prices for both ranges. Similarly, SVM achieved the best results in predicting closing prices.

Aalborg et al. [34] aimed to determine which variables should be used to predict Bitcoin's return, volatility, and trading volume. Variables used are trading volume, the number of unique user addresses, the VIX index, and the number of Google searches. The multivariate regression model was used in the study. Daily and weekly data from March 1, 2012, to March 19, 2019, were used in the study. Models are insufficient in estimating Bitcoin returns; Although relatively successful results are obtained in daily volatility estimation, it has been determined that the models are inadequate for weekly volatility estimation. In addition, only Google searches increased the performance of the models for trade volume.

In their work, Adcock and Gradojevic [35] made price and density predictions for Bitcoin returns. ARIMAX, GARCH-M, linear regression, quantile regression ANN and kernel regression with the date range 19 July, 2010 - 5 March, 2010 were used in the study. The study used a 50-day buy-sell signal, 200-day buy-sell signal, and VIX volatility index. In the study, the predictive performance of all models was compared with that of the random walk model. Only recurrent neural networks gave better results than the random walk model of the models used. Similarly, the ANN model provided the most relevant results for density prediction.

Kristjanpoller and Minutolo [36] made volatility predictions for Bitcoin price with the ANN – GARCH model, a hybrid of artificial neural networks, and the GARCH model. The study used daily data from 13 September 2011 to 26 August 2017. Volatility is represented by a 22-day (1-month) return variance. In addition, 10-day and 44-day variances with seven technical indicators were used for 12 different models with different combinations of hyperparameters. Using technical indicators increased all models' performance with MSE values. The best hybrid models in this group are those with the fewest layers and neurons. The authors also used PCA on technical indexes, which helped increase the models' performance.

Walther et al. [37] made daily, weekly, and monthly volatility predictions for Bitcoin, Etherium, Litecoin, Ripple, Stellar, and the cryptocurrency index (CRIX). Seventeen external financial and economic indicators were used in forecasting models in the study. Daily prices consist of data until 31 July 2019, although the starting date varies according to currencies. GARCH and GARCH-MIDAS models were used in the study. The best predictions in the study were obtained with the model in which the global actual economic activity index is used externally. The second best prediction was obtained with the prediction averages of all models. Roy and Ojha stated that Twitter is a big gold mine where people share their instant feelings and thoughts, and based on this, they carried out a sentiment analysis on Twitter. Three deep learning models were created and compared for sentiment analysis. Google Bidirectional Encoder Representations from Transformers (BERT) [38], LSTM, and CNN algorithms were used, and it was determined that the BERT model outperformed the others. The language of the study was English, which was an increasing factor for the accuracy rate.

Mallqui and Fernandes [40] aimed to determine the machine learning algorithms that achieved the best results in price predictions with the help of Bitcoin daily prices between the years 2010-2013. They tried to predict the direction and severity of Bitcoin price movement with the help of Artificial Neural Networks (ANN), SVM, and Ensemble algorithms (based on Recurrent Neural Networks (RNN) and k-means clustering method). Similarly, ANN and SVM are used for Bitcoin's maximum, minimum, and closing prices regression model. As result, it was revealed that the SVM algorithm gave the best results for all predictions (maximum, minimum, and closing prices) and both ranges.

Briere et al. [41], using weekly data for the period 2010-2013 in their study, analyzed Bitcoin investments for the US investor who has a diversified portfolio with both traditional assets (stocks, bonds, fixed currencies) and alternative investments (including commodities, hedge funds, real estate). During the mentioned period, Bitcoin investments were found to have very distinctive features, including the exceptionally high average return and volatility. They were found to have a very low correlation with other assets.

Catania [42] compared various alternative univariate and multivariate models to predict the prices of cryptocurrencies with the largest trading volume, Bitcoin, Litecoin, Ripple, and Ethereum. In the study, a series of cryptography experiments were applied to combine the predictors, and combinations of univariate and multivariate models were proposed. The results showed statistically significant changes when using combinations of univariate models.

Kodama et al. [43] analyzed the time series dataset of Bitcoin prices, which recorded individual transactions in Euro in the Coinbase market between April 23, 2015, and August 15, 2016. The study applied the Markov technique to classify regions of variable volatility represented by three latent state regimes using the univariate autoregressive and the dependent mix models. At the end of the study, they found that cryptocurrency series can also be applied to causality inference of iterative neural networks, such as exchange rate time series data.

Karakoyun [44], predicted Apple stock and Bitcoin prices, applying LSTM and ARIMA as deep learning architectures. The author compared the performance of these two architectures. Apple stock pricing data from September 7, 1984 to February 28, 2018 were used in the study. The data included 8428 Apple stock prices in total, belonging to the daily closing prices of the relevant stock. The dataset used for training deep learning models included Bitcoin prices between April 28, 2013, and October 29, 2017. The study collected 1646 price data, and the deep learning model was asked to make a pricing forecast for the next 30 days. When Apple stock price prediction is made with the ARIMA model, the result obtained in the RMSE metric is 34.04762. To make predictions with the LSTM model using the same data set, LSTM with 64 neurons in the input layer, LSTM with 128 neurons in the second layer, MLP with 32 neurons in the third layer, and MLP with one neuron in the output layer was used. In the results obtained with the LSTM obtained with the same data, the value obtained in the RMSE metric was 9.61. A lower RMSE means a more successful model, and the LSTM model was more successful in predicting Apple stock prices. In the prediction of Bitcoin prices, the ARIMA model took 1146.067 RMSE. In the prediction of Bitcoin prices with LSTM, LSTM with 64 neurons in the input layer, LSTM with 128 neurons in the second and third layers, MLP with 64 neurons in the fourth and fifth layers, an MLP with one neuron in the output layer were used, and RMSE of 93.27 was taken. These results have proven that the LSTM model is more successful than traditional statistical models.

Namini and Namin [45] revealed how advanced new algorithms, emerged with the development of more complex structures such as deep learning, with the increase in the processing capacity of the computer, compared to traditional algorithms in terms of success rate and margin of error. The study showed that the LSTM model obtained results from the output layer with an 85 lower error margin than the ARIMA model.

Björklund and Uhlin [46] reviewed the existing properties of financial time series and conducted a study on the efficient market hypothesis, in which potential anomalies were emphasized by using their statistical properties in an artificial neural network. Based on the theoretical review, an interdisciplinary approach between machine learning and finance has evolved into a quarterly prediction of the expected time series. In a related context, they used exchange rates and indices from different asset classes, such as stocks and commodities, using the stock optimization model to evaluate the use of future return forecasts.

Grachev [47] examined the efficiency and limitations of time series models such as ARIMA, GARCH, and ARMA-GARCH for the stock market. The study first assesses the unique features of financial data, particularly the fluctuation, clustering, and excesses in the return distribution. It addresses the limitations of using autoregressive integrated moving average (ARIMA) models in financial economics. Second, the application of ARMA-GARCH models for estimating both conditional means and the conditional variance of returns is examined. Finally, the prediction performance of various candidate ARMA-GARCH models is discussed using standard model selection criteria such as AIC, BIC, SIC, and HQIC. For MSCI World Index excess return and Fama-French 3-factor excess return models, an ARMA (1.0) + GARCH (1.1) yielded the best results in both groups in the same period.

Moreno [48] presented a description and comparison of the leading models of ANNs that have proven helpful in time series estimation and a standard list of steps for the practical application of ANN in this type of task. MLP, Radial Basis Function (RBF), Generalized Regression Neural Network (GRNN), and RNN models were analyzed. For this purpose, they used a time series consisting of 244-time points. Their comparative study showed that the error made by the four neural network models analyzed was less than 10. The model with the best performance is RBF, followed by RNN and MLP. The GRNN model has the worst performance.

Fakhr and Baasher [49] studied the foreign exchange market (FOREX), a highly volatile complex time series, for which forecasting the daily trend is a challenging problem. The high exchange rate daily trend is taken as a binary classification problem with the results of uptrend and downtrend. Many key features resulting from time-series data, including technical analysis features, were created using multiple historical time windows. Various feature selection and feature extraction techniques were used to find the best subsets for the classification problem. RBF, MLP, and SVM machine learning methods were tested and analyzed for each feature subset. While analyzing four major FOREX currency pairs, it was found that the results showed consistent success in daily forecasts and expected profits.

2 Scientific Research

This chapter demonstrates the definition and formulas of Machine learning Models, Time Series Models, Model Evaluation Techniques, Feature Selection Techniques.

2.1 Machine learning Models

Machine learning (ML) is the technique of assisting a computer in learning without receiving strict orders. This is a subcategory of Artificial Intelligence (AI). Machine learning algorithms are used to discover patterns in data. Additionally, these patterns are utilized to build a predictive data model. Just as people grow with practice, machine learning outcomes progress with more data and expertise. Due to its versatility, machine learning is an excellent alternative in situations when data, demands, or tasks are continually changing, or where coding a solution properly is not feasible. Basic concepts of ML are:

- **Dependent Variable**: This is the primary feature in the machine learning task that we are attempting to forecast. (e.g. return of the stock)
- **Independent Variable**: The independent variable is the one that assists us in estimating the dependent variable. (opening price, closing price)
- **Overfitting**: It is possible that the data set is learnt really effectively, almost to the point of memorizing, but makes errors while attempting to make predictions on the data set just observed.
- **Underfitting**: Overfitting is the polar opposite of underfitting. In this instance, models are unable to capture critical characteristics in the data set and so cannot do the essential learning.

Algorithms of ML can be categorized as two: Supervised Learning, and Unsupervised Learning [71].

- **Supervised Learning**: In supervised learning, the data set and the intended output are known. Supervised learning enables the extraction of a function (a match between the input and result data) from this information by feeding the data and the results back to the computer. As a result, the machine gains an understanding of the link between the dependent and independent variables [71]. There are two subfields of Supervised Learning: Classification, and Regression.
  - **Classification**: Classification is a supervised learning technique used in machine learning and statistics in which the computer software learns from the input data and then utilizes this knowledge to categorize fresh observations. These datasets may be binary in nature (for example, detecting whether a person is male or female, or whether an email is spam or not) or multi-class in nature (multiclass classification in machine learning is a problem of classifying instances into one of three or more classes) [72].
  - **Regression**: Cause-effect connections are often examined in regression analysis using statistical models comprising two or more independent variables. In other words, the degree to which one or more factors influence one or more other variables is investigated. If there is a connection between the variables, the relationship's degree is indicated mathematically.

This is referred to as the regression function. In a regression issue, we attempt to forecast the values of continuous output variables; that is, we attempt to map the input data to some continuous function [72].

- **Unsupervised Learning**: Unsupervised learning is a kind of machine learning approach in which the model is not supervised. Rather than that, model finds out about the data on its own. In comparison to supervised learning, unsupervised learning algorithms enable to do more complicated processing tasks. Unsupervised learning does not need the system to be taught; instead, it learns from the data. Unsupervised machine learning discovers any previously undiscovered pattern in data. Clustering is one of the most popular Unsupervised Learning techniques [71].
  - **Clustering**: Clustering is a statistical technique that is used in unsupervised learning, a subset of machine learning. Unlabeled data used in unsupervised learning are categorized using the clustering approach according to the features they exhibit. Consider the following scenario: you have a data repository; thousands of data are clustered according to their common features; as a consequence, a particular number of datasets are produced based on the data's closeness to one another [73].

In the following sections, the foundations of the ML regression algorithms that are used in this study is given.

2.1.1 Decision Trees

In many studies, a number of methods are needed when performing certain operations in order to make the best decision about the problem at hand. Among these methods, the decision tree method is one of the most widely used classification and regression models, which provides convenience to the decision-maker in terms of use, has high reliability and has been used recently [50]. Decision Tree is a graphical technique that evaluates all possible action options, all possible factors that may affect these action options, and each possible outcome based on all these factors, depending on the data, and that facilitates the decision-maker to understand the problem through the use of geometric symbols such as lines, squares, and circles. Tree-based approaches provide a high degree of accuracy, stability, and interpretability. Unlike linear models, they can also map nonlinear relationships quite well. Classification or regression can be adapted to solve any given problem. Methods such as decision trees, random forest, gradient reinforcement are widely used in all kinds of data science problems.

Decision tree learning is a technique for approximating discrete-valued target functions that uses a decision tree to represent the learnt function. Learned trees can be represented as if-then rule sets to increase human readability. The decision tree creates classification or regression models in the form of a tree structure. When dividing a dataset into smaller and smaller subsets, an associated decision tree is progressively developed at the same time. The terminology of Decision Trees is given below:

- Root Node: It represents the entire instance, and this node is then split into two or more subsets.
- Splitting: Dividing one node within two or more sub-nodes.
- Decision Node: If a child node is divided into other child nodes, it is called a decision node.

- Leaf Node: Nodes that do not divide are called Leaf or Terminal nodes.
- Pruning: When child nodes of the decision node is removed, it is called pruning. In other words, it is vice versa of the splitting process.
- Subtree: A sub-section of the entire tree is called a branch or sub-tree.
- Parent and Child Node: A node which is divided into child nodes is called the parent node and the child nodes of the child nodes are called the child node.

The goal in decision trees is to create as pure, homogeneous nodes as possible. In other words, decision trees use multiple algorithms to decide whether a node (attribute) should be split into two or more child nodes. With the creation of sub-nodes, the data is divided into sub-nodes in a more homogeneous and understandable way. Various methods are used to split nodes.

- Continuous Target Variable
  - Reduction in Variance
- Categorical Target Variable
  - Gini Impurity
  - Information Gain
  - Chi-Square

This study aims to predict the numerical output. Thus, only Reduction in Variance will be explained; Reduction in Variance is an algorithm used for continuous target variables (regression problems). It is used to explore features that effectively reduce the variance of the numerical output variable, and uses the standard variance formula to choose the best split. The formula is given below:

$$\sigma^2 = \frac{\sum_{i=1}^{n}(x_i - \mu)^2}{n}$$

Reduction in Variance follows this steps throughout the learning process of the algorithm:

- Calculating the variance for the node.
- Each split's variance is derived as the weighted average of the variances of each node.

2.1.2 Random Forest

The Random Forest algorithm is a method consisting of decision trees and nodes developed by Breiman [9]. As the name suggests anyway, it creates a forest and does it somehow randomly. The "forest" algorithm builts is a collection of decision trees that are mostly trained by the "bagging" method. Bagging is short for bootstrap aggregation. It means bringing together the trees obtained by the resampling method. In this respect, it opened up a completely different perspective to single tree structures and moved the world of machine learning ahead [50].

In this method, new trees are formed by continually drawing representatives from the sample data set to substitute them, and a community arises from these trees. Based on this, they are referred as ensemble methods. All of the resulting trees are used for modeling, and each one is asked for their opinion. The result of the whole process is given as a single output by taking a vote/average opinion by evaluating the opinion of each. Bagging should be seen as a methodology. For example, when the CART algorithm is secured, a bagged tree emerges. When the CART algorithm is connected on the basis of both observations and variables, Random Forests emerge. With Random Forest, regression and classification analyzes can be performed. In order to branch the nodes according to the method, the best random value in the nodes is selected, and certain weights are given to the created decision trees. These weights are determined according to the internal errors of the decision trees, and the decision tree with the lowest error is given the highest weight, and the decision tree with the highest error is given the lowest weight. These given weights are used for voting in class estimation. Afterward, these votes are collected, and the final decision is made. Thus, with the random forest algorithm, more reliable predictions are made by combining the results of multiple decision trees and making a single decision on behalf of the forest. The steps of this algorithm is as follows:

- Step 1: In the first step, the number of decision trees (n) to be created according to the data properties is determined.
- Step 2: At each node in the created decision trees, m random variables are selected, and the best branch is determined by calculating with the Gini index.
- Step 3: The best branch determined in the previous step is divided into two sub-branches. This process is continued until the Gini index becomes zero, in other words, until there is only one class left in each node.
- Step 4: In the last stage, if it is a classification problem, the class with the highest number of votes is chosen as the final decision, if it is a regression problem, the average of individual trees' output is returned as estimation among the predictions made by n decision trees.

2.1.3 LightGBM

With the rapid increase in data size and diversity in recent years, the importance given to algorithm optimizations is increasing. For this reason, as an alternative to the Gradient Boosting algorithm, algorithms such as XGBoost, LightGBM, Catboost that can be considered versions of Gradient Boosting have been developed. With these algorithms, it is aimed to achieve faster training and higher accuracy. LightGBM is a boosting algorithm developed in 2017 as part of the Microsoft DMTK (Distributed Machine learning Toolkit) project. Compared to other boosting algorithms, it has advantages such as high processing speed, large data processing, less resource (RAM) usage, high prediction rate, parallel learning and GPU learning support. According to the article "LightGBM: A Highly Efficient Gradient Boosting Decision Tree" in which the model is introduced, studies have concluded that LightGBM is 20 times faster than other models [14].

There are two strategies, level-wise or depth-wise, or leaf-wise, can be used in learning decision trees. In the level-oriented strategy, the tree's balance is maintained as the tree grows. In the leaf-

focused strategy, on the other hand, the division process continues from the leaves, which reduces the loss. Thanks to this feature, LightGBM differs from other boosting algorithms. The model has less error rate and learns faster with the leaf-oriented strategy. However, a leaf-oriented growth strategy causes the model to be prone to over-learning in cases where the number of data is low. As a result, the methodology is better suited for usage with large amounts of data. In addition, parameters such as tree depth and the number of leaves can be optimized to prevent overfitting.

LightGBM also uses two different techniques from other algorithms. These include Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB), which both address the issue of the quantity of data samples and variables [14].

- GOSS: GOSS aims to reduce the number of data while maintaining the accuracy of decision trees. Traditional Gradient Boosting scans all data samples to calculate the information gain for each feature, but GOSS uses only the important data. Thus, the number of data is reduced without affecting the distribution of the data too much.

- EFB: EFB aims to reduce the number of variables without harming the accuracy rate and accordingly increase the efficiency of model training. EFB has two processing steps. These are creating packages and merging variables into the same package. With EFB, sparse features are combined to create denser features. This leads to reduced complexity and a faster training process with lower memory consumption. In summary, GOSS reduces data size to calculate information gain, omitting data that may be considered less important, while EFB aggregates variables to reduce dimensionality. With these two functions, LightGBM increases the efficiency of the training process.

2.1.4 XGBoost

XGBoost(eXtreme Gradient Boosting) is a high-performance version of the Gradient Boosting algorithm optimized with various modifications. It became a part of our lives with the article titled "XGBoost: A Scalable Tree Boosting System" published by Tianqi Chen and Carlos Guestrin in 2016. The most important features of the algorithm are that it can achieve high predictive power, prevent over-learning, manage empty data and do them quickly. According to Tianqi, XGBoost runs ten times faster than other popular algorithms.

The first step in XGBoost is to forecast the initial score. This estimate can be any number as the correct result will be reached by converging with the actions to be taken in the next steps. This number is 0.5 by default. The evaluation of these estimations is examined with the residuals of the model. These errors are found by subtracting the estimated value from the observed value. The next step is to construct a decision tree that predicts errors. The aim here is to learn from the errors and get closer to the correct prediction.

The similarity score is calculated for each branch of the tree created. The similarity score indicates how well the data are grouped into branches. After the similarity scores are calculated, the next question is whether a better guess can be made. Trees of all possible possibilities are constructed to answer this question. Similarity scores are calculated for all of them. The gain will be calculated in order to decide which tree is better. While branches are evaluated with similarity score, the whole tree

is evaluated with gain. The first tree is completed with the above-mentioned operations (establishing trees that predict errors, calculating similarity and gain scores for trees, pruning, and obtaining model outputs). After the first tree is completed, with the first prediction, learning rate, and first tree, the second forecast is made. In this way, trees are calculated, and the predictions continue to be corrected. These processes will continue until the errors are very low or the specified number of trees is reached. By default, branches are split down six times.

2.1.5 Extra Trees

The Extra Trees algorithm is a different version of the Random Forest method. As in the Random Forest method, the model is trained using copies of the data set, but in the branching phase of the nodes, random branching is done instead of using the decision criterion to make the optimum separation. Although this idea reduces the complexity and processing load in solving some data analysis problems, it has poor performance in analyzing large data sets with high noise. When evaluated statistically, this method generally causes an increase in bias and decreases the variance [51]. On the one hand, bootstrap representative of the training dataset forms each tree during the training of the RF algorithm. During the training of the Extra Trees algorithm, the whole training dataset is fitted to each decision tree. Extra Trees also randomly samples the features during the splitting of a tree. On the other hand, unlike Random Forest, the Extra Trees algorithm picks a splitting point randomly. This random selection makes trees throughout the ensemble less correlated, at the same time, it increases algorithm's variance [51].

2.1.6 AdaBoost Regressor

Boosting, one of the learning algorithms, is an ensemble algorithm based on machine learning, which was developed by Schapire in 1990 and used in solving both classification and regression problems. The amplification algorithms were mainly developed to answer the question: can a single strong learner form from a set of weak learners? These algorithms work iteratively. In this method, a strong learner is created by adding a new weak learner at each iteration. In other words, a strong learner is formed by gathering weak learners. Individual models do not perform well on the entire dataset, but work well for a portion of the dataset. Therefore, each model actually improves the performance of the ensemble. One of the best known and most successful boosting algorithms is AdaBoost (Adaptive Boosting) algorithm developed by Freund and Shapire in 1996. It consists of an iterative process in which new models are added to create a community. It is adaptive in a sense. With each new iteration of the algorithm, new models are built to overcome the mistakes made in previous iterations. In this method, a sequential process is performed in which each subsequent model tries to correct the errors in the previous model. That is, the later models are dependent on the previous model. Although the AdaBoost algorithm is mainly developed for solving classification problems, it can also be used to solve regression problems. The workflow of AdaBoost is given below:

- A subset of the original dataset is constructed.
- At first, all data points are weighted equally.

- A base model is created in this subset.
- This model is used to make predictions on the entire dataset.
- Errors are calculated using observation and prediction values.
- Incorrectly predicted observations are given higher weight.
- On the dataset, another model is developed and predictions are generated.
- Similarly, many models are produced, with each one rectifying the preceding model's flaws.
- The final model (strong learner) is the weighted average of all models (weak learners).

2.1.7 Catboost Regressor

Catboost is an open-source machine learning algorithm based on Gradient Boosting developed by the Yandex company. It was introduced in April 2017 with the article "CatBoost: unbiased boosting with categorical features" as an alternative to XGBoost and LightGBM, which were developed to increase the performance of Gradient Boosting. Its name comes from a combination of the words "Category" and "Boosting". High learning speed, ability to work with both numerical, categorical, and text data, GPU support, and visualization options are the most distinguishing features from other algorithms. CatBoost has an important position by shortening the data preparation phase. It can deal with empty data, encoding categorical data [15].

The reason why it can work with categorical data with high performance is that it has a unique coding method. In other words, there is no need for a separate coding process while preparing the data. In fact, it is highly recommended not to code. This will affect both the learning speed and the quality of the results. Also, Catboost builds symmetrical trees. In this way, it achieves a high prediction rate without establishing very deep trees and overcomes the over-learning problem. If over-fitting occurs, the algorithm stops learning before it reaches the properties specified in the parameters (for example, the specified number of trees). This option is set from the initial parameters [15].

2.1.8 Huber Regressor

For the M- Estimation method developed by Huber (1964), the objective function is known as;

$$\rho(r) = \begin{cases} \frac{1}{2}r^2 & , |r| < k \\ k|r| - \frac{1}{2}k^2, & |r| \geq k \end{cases}$$

The constant k in this function is called the tuning constant and its value is used as 1.345. Here, the purpose of k is to ensure that the efficiency of the function used is approximately 95 if the data really has a normal distribution. The most important feature of Huber M- Estimators is that the function behaves like a normal in the $(-k, k)$ interval, while in other places it behaves like a Laplace (Double Exponential) distribution. For the Huber M-estimator, the Impact function and the Weight function

are denoted by

$$\psi(r) = \begin{cases} r & ,|r| < k \\ ksign(r) & ,|r| \geq k \end{cases}$$

$$w(r) = \begin{cases} 1 & ,|r| < k \\ \frac{k}{|r|} & ,|r| \geq k \end{cases}$$

2.1.9 Multiple Linear Regression

A linear regression refers to a regression model that consists entirely of linear variables. It may aid in the comprehension and prediction of the behavior of complex systems, as well as the analysis of experimental, financial, and biological data. A linear regression model's general equation is:

$$y = k + \beta_1 x_1 + \beta_2 x_2 + \ldots . + \beta_n x_n \tag{1}$$

Multiple Linear Regression is an analysis to reveal the relationship between a dependent variable and a series of independent variables associated with it. Multiple linear regression examines the linear relationship between two or more independent variables and one dependent variable. It helps to estimate the dependent variable when the independent variable is a known factor after the results are obtained [52]. Multiple Linear Regression is denoted as:

$$\mathrm{Y_i} = \beta_0 + \beta_1 \mathrm{X_{1i}} + \beta_2 \mathrm{X_{2i}} + \ldots + \beta_\mathrm{k} \mathrm{X_{ki}} + \varepsilon_\mathrm{i} \tag{2}$$

where :
$\mathrm{Y_i} = i$ th observation of the dependent variable $\mathrm{Y, i} = 1, 2, \ldots, \mathrm{n}$
$\mathrm{X_j} =$ independent variables, $\mathrm{j} = 1, 2, \ldots, \mathrm{k}$
$\mathrm{X_{ji}} = i$ th observation of the j th independent variable
$\beta_0 =$ intercept term
$\beta_\mathrm{j} =$ slope coefficient for each of the independent variables
$\varepsilon_\mathrm{i} =$ error term for the i th observation
$\mathrm{n} =$ number of observations
$\mathrm{k} =$ number of independent variables

2.1.10 Ridge Regression

Ridge Regression (also known as Tikhonov regularization) is obtained by adding a regularization term ($\alpha \sum_{i=1}^{n} \theta_i^2$) to the cost function in linear regression. With this addition, the learning algorithm both learns the data and tries to keep the model weights as small as possible. The α hyper-parameter controls the amount of regularization on the model. Ridge regression for $\alpha = 0$ is the same as linear regression. The solution technique of Ridge regression is similar to the simple least-squares method. In the Ridge regression method, the regression is performed by adding a small and positive constant to the diagonal elements of the $(X'X)$ matrix formed by the variables in standard form before calculating

the coefficient estimates [53]. Accordingly, the ridge regression is denoted as:

$$\text{argmin}_\beta \sum_i \left(y_i - \boldsymbol{\beta}' \boldsymbol{x}_i\right)^2 + \lambda \sum_{k=1}^{K} \beta_k^2 \tag{3}$$

Ridge regression is resistant to over-fitting and offers a solution to multidimensionality by creating a model with all variables, does not remove irrelevant variables, only brings their coefficients closer to zero. It is necessary to find a good value for α (penalty) when building the model [53]. To do so, cross-validation is used. The details of cross-validation is explained in 2.4.3 on page 32.

2.1.11 Lasso Regression

Lasso regression (Least Absolute Shrinkage and Selection Operator Regression) is another regularized variant of linear regression: $\alpha \sum_{i=1}^{n} |\theta_i|$ is added. An important characteristic of Lasso regression is that it eliminates the weights of the least important features (for example, making it zero). In other words, Lasso regression employs L1 regularization, which imposes a penalty proportional to the amount of the coefficients' absolute value. Regularization of this approach might lead to sparse models with few coefficients; certain coefficients may become zero and are therefore deleted from the model. Increased penalties result in closer-to-zero coefficient values, which is suitable for constructing simpler models. [54]. Lasso regression is denoted as:

$$\hat{\beta}^{\text{lasso}} = \arg\min_{\beta} \sum_{i=1}^{n} \left(\mathbf{y}_i - \left(\beta_0 + \beta^{\mathbf{T}} \mathbf{x_i}\right)\right)^2 + \lambda \|\beta\|_1 \tag{4}$$

2.1.12 Support Vector Regression

Support Vector Machines (SVM) is a supervised learning method developed by Vapnik and is used for classification and regression. Points outdoor the tube are support vectors due to the fact they are helping the shape or formation of the tube. These characteristic vectors had been named support vectors due to the fact intuitively you may say that they "aid" the keeping apart hyper-plane, or it might be said that for the keeping apart hyper-plane, the support vectors play an equal function similar as the pillars to a building. Compared to other traditional learning methods, this method has much better performance and ability to solve nonlinear problems [7].

In classification problems, the aim is to find the appropriate classifier function that separates the two classes. The goal is to find the best linear classifier plane even for nonexistent data. In the case of SVMs, the maximum distance between two classes is called the margin. There are infinitely many lines to separate these two classes. But there is only one line that maximizes the margin. This line is called the best separating hyperplane. In SVM, the goal is to transform the classification problem into a quadratic programming problem; to get the best hyper is to find the global minimum that represents the plane.

$$\hat{y} = \begin{cases} 0 \text{ if } \mathbf{w}^T \cdot \mathbf{x} + b < 0 \\ 1 \text{ if } \mathbf{w}^T \cdot \mathbf{x} + b \geq 0 \end{cases} \tag{5}$$

SVM, which has been successfully applied in classification problems, can be applied to regression

problems by giving an alternative loss function. In some cases, the data can't be separated with a linear line. But, if one of the classes in the data set is lifted a little higher, it will be possible to separate the two classes with a plane to be placed between the lower class and the upper class. Here, the process of raising one of the classes a little bit is done by means of kernels. Many classes that are intertwined with the same logic can be easily separated from each other by means of kernels. Thanks to this solution, SVMs are preferred for classification processes in many different application areas. In this way, it is possible to increase the size of the classes that cannot be separated by a straight line and to separate them by a plane. There are different types of kernel functions used in the literature for this purpose [55].

The kernel aids in the discovery of hyperplanes in higher dimensional space without requiring more computing effort. Generally, the larger the dimension of the data, the more computational complexity. This dimension increase is necessary when the separation hyperplane cannot be found in a particular dimension and needs to be moved to a higher dimension.

- Polynomial Kernel: In this method, to solve the problem, it is acted as if it is going out of two dimensions and operating in three or more dimensions. We cannot classify the left (2 dimensions) distribution with a straight line. For this, the Polynomial Kernel can be used for such problems. When operating in the third dimension, a plane is used instead of a straight line to classify and can be classified higher margin [7]. The equation of Polynomial Kernel is given below:

$$k\left(\mathbf{x_i}, \mathbf{x_j}\right) = \left(\mathbf{x_i} \cdot \mathbf{x_j} + 1\right)^d \tag{6}$$

- Radial Basis Function: The RBF kernel is also known as the Gaussian kernel. The feature space has an infinite number of dimensions because it can be expanded in the Taylor series. In the following form, the parameter y defines how much impact a single training example has. The larger it is, the closer the other examples need to be considered. This is a universal kernel. It is used when there is no prior knowledge of the data [7]. The equation is given below:

$$k\left(\mathbf{x_i}, \mathbf{x_j}\right) = \exp\left(-\gamma \left\|\mathbf{x_i} - \mathbf{x_j}\right\|^2\right) \tag{7}$$

- Sigmoid Kernel: The hyperbolic tangent kernel is also known as the sigmoid kernel or multilayer perceptron (MLP) kernel. The sigmoid kernel comes from the realm of neural networks. Here, the bipolar sigmoid function is often used as the activation function for artificial neurons. It can be used as a proxy for neural networks [7]. Sigmoid Kernel is denoted as:

$$k(x, y) = \tanh\left(\alpha x^T y + c\right) \tag{8}$$

2.1.13 K-Nearest Neighbor Regressor

Unlike other Supervised Learning algorithms, K-Nearest Neighbor does not have a training phase. Training and testing are pretty much the same thing. It is a lazy type of learning. Therefore, kNN is not an ideal candidate as the algorithm required to process large dataset. With KNN, KNN searches for the points that are nearest to the new point. K is the number of the unknown point's closest neighbors. The

algorithm's k quantity (usually an odd number) is chosen to predict the results. In pattern recognition, kNN is a non-parametric method used for classification and regression. Both scenarios use the k closest training examples in the feature space as the input. The output depends on whether the kNN will be used for classification or regression [57]. The formulas for distance calculations that can be used are given below.

- Euclidean Distance

$$d(p,q) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2} \tag{9}$$

- Manhattan Distance

$$\sum_{i=1}^{n} |x_i - y_i| \tag{10}$$

- Minkowski Distance

$$\left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p} \tag{11}$$

2.2 Time-Series

A time series is a sequence of observations made over period of well-defined data elements. For instance, a time series might be created by calculating the value of retail sales for each month of the year. This is because sales income is properly defined and recorded at regular periods. Time series are not data that are gathered seldom or just once [68].

Three components may be identified in an observed time series: the trend (long-term direction), the seasonal (systematic, calendar-related motions), and the irregular (unsystematic, short term fluctuations).

2.2.1 Time-Series Components

- **Seasonal Variation**: is often a yearly period that happens for a large number of measurement series where comparable behavior patterns are seen at certain periods of the year, such as weekly, monthly, or quarterly. The ice cream sales model series is a good illustration of this, since it is usually popular throughout the summer. If a time series is only monitored yearly (for example, once a year), seasonal variation cannot be seen [68].

- **Trend**: When a series demonstrates a consistent upward or downward growth or drop throughout at least many successive time periods, this is referred to be a trend. While a trend may be described as "average long-term change," there is no mathematical description that is completely adequate. Perception of trends is somewhat determined by the duration of the observed series [68].

- **Cyclic Variation**: This term refers to periodic cyclic variation that occurs over a longer length of time than one year. As examples, five-year business cycles and the daily rhythm of biological activity of living things might be cited [68].

- **Irregular Fluctations**: The phrase irregular fluctuations is often used to refer to "leftover" variations that remain after trend, seasonality, and other systematic variables are eliminated. As a result, they might be utterly unexpected and random [68].

Taking into account the impacts of these four components, two distinct kinds of time series models are often used: additive and multiplicative. These models are as follows for components at time t: y_t: observation, T_t: trend, S_t: seasonal, C_t: cyclical, and I_t: irregular fluctation:

$$\textbf{Additive Model: } y_t = T_t + S_t + C_t + I_t \tag{12}$$

$$\textbf{Multiplitive Model: } y_t = T_t + S_t + C_t + I_t \tag{13}$$

The multiplicative model is predicated on the premise that the four components of a time series are not always independent of one another and may interact. The four components of the additive model may be completely independent of one another.

2.2.2 Stationary Time Series

A stationary time series is one whose attributes remain constant regardless of the time interval over which it has been monitored. Thus, time series that exhibit trends or seasonality are not stationary – the trend and seasonality will impact the time series' value at various points in time. On the other hand, a white noise series is stationary – regardless of when it is seen, it should seem almost identical at any particular time [67].

2.2.3 Autocovariance, Autocorrelation and Partial Correlation Functions

If a time series is stationary, this means that the joint probability y_t and y_{t+k} distribution of and its observation is the same for time periods t and $t+k$ separated by the same interval k. Useful information about the nature of these time series may be created by drawing a scatter diagram of all of the y_t, y_{t+k} data pairs separated by the same k interval. The k interval is called lag [68].

The autocovariance at the k delay indicated by the equation below is the covariance among y and its value y_{t+k} in an another time period.

$$\gamma_k = \text{Cov}\left(y_t, y_{t+k}\right) = E\left[\left(y_t - \mu\right)\left(y_{t+k} - \mu\right)\right] \tag{14}$$

$gamma_k$ obtained for $k = 0, 1, 2...$ is called autocovariance function [68].

A specific statistical procedure examines the relationship between two random variables at distinct times in time. An alternative method for determining the link between y_t and y_{t+r} is to filter away the linear influence of the random variables and then compute the transformed random variables y_{t+r}, y_{t+r-1}. This phenomenon is referred known as partial autocorrelation [68].

$$\eta_{kk} = \text{Corr}\left(y_t - \Gamma\left(y_t \mid y_{t+1}, \ldots, y_{t+k-1}\right), y_{t+k} - \Gamma\left(y_{t+k} \mid y_{t+1}, \ldots, y_{t+k-1}\right)\right) \tag{15}$$

2.2.4 Differencing

Calculating the differences between successive observations is one approach to make a time series stable. This is referred to as distinguishing. Differentiation eliminates level variations in a time series and hence aids in the stabilization of the mean by controlling (or eliminating) trend and seasonality. Apart from visualizing the data's chronology, the autocorrelation function graph may also be used to detect non-stationary time series. The autocorrelation function of a stationary time series approaches zero relatively fast, but the autocorrelation function of non-stationary data decreases slowly [69].

2.2.5 Unit Root Tests

Unit root testing is one method for objectively determining the requirement for differentiation. These are stationarity statistical hypothesis tests that are used to establish whether or not a difference must be taken. There are several unit root tests, each with its own set of assumptions [69].

- **Augmented Dickey-Fuller Test**: This test is denoted as:

$$y'_t = \alpha + \beta_t + \varphi y_{t-1} + \gamma_1 y'_{t-1} + \gamma_2 y'_{t-2} + \ldots + \gamma_k y'_{t-k} \tag{16}$$

- y'_t specifies the series taken first difference.
- $y'_t = y_t - y_{t-1}$ and k is the number of lags to include into the regression and is often set to three.

2.3 Time-Series Forecasting

Time series forecasting is the process of analyzing time series data with the use of statistics and modeling in order to provide predictions and improve strategic decision-making. It is not always an accurate prediction, and forecasting probabilities may vary dramatically; particularly when dealing with regularly changing variables in time series data as well as external events.

Forecasting is used in a variety of sectors. It is useful for a variety of purposes, including weather forecasting, climate forecasting, economic forecasting, healthcare forecasting, technical forecasting, financial forecasting, retail forecasting, commercial forecasting, environmental studies forecasting, and social studies forecasting. Essentially, anybody with reliable historical data may examine it using time series analysis techniques and then model, forecast, and predict it. The whole purpose of time series analysis in certain businesses is to aid with predicting. Developed on the idea that time-dependent events are random and that time series relating to these occurrences are stochastic processes, this approach assumes that the time series to which it is applied is a discrete and stationary series with evenly spaced observation values. However, in fact, the mean and variance of the time series alter throughout time. This shift, which is often seen in non-stationary time series, is caused by trend, regular, irregular, and random fluctuations [68]. To use the Box-Jenkins technique to forecast non-stationary time series, the series must first be transformed to become stationary. Prediction occurs in four steps using the Box-Jenkins Method [70].

1. **Model Determination**: The Box-Jenkins model suitable for the time series is determined at this stage.

2. **Parameter Estimation**: It is the stage where the parameters related to the model determined during the model determination phase are estimated.

3. **Test of Suitability**: At this stage, where the suitability of the model to the data set is tested with statistical methods, if the model is found suitable, the final stage is passed, if not, the first stage is returned to determine another model.

4. **Prediction**: The selected best fit model is used for estimation.

In the following chapters, with their equations, different methods of forecasting is given.

2.3.1 Naive Forecaster

The naive method takes into account what happened in the preceding period and forecasts that it will happen again. This method is denoted as:

$$\hat{y}_{T+h|T} = y_T \tag{17}$$

Naive Forecaster may give good results when examined with metrics, but they may not be very useful in cases where decisions need to be made based on the estimates made.

2.3.2 Seasonal Naive Forecaster

Seasonal Naive Forecasting is a scenario in which each prediction is equal to the most recent measured value from the same year's season (e.g., the same month of the previous year)

$$\hat{y}_{T+h|T} = y_{T+h-m(k+1)} \tag{18}$$

2.3.3 Exponential Smoothing

The exponential smoothing method is a method in which estimates or predictions are constantly updated, taking into account the latest changes and jumps in the data. In this context, exponential smoothing of time series data is the assignment of exponentially decreasing weights for the newest and oldest observations. In other words, the larger the data, the less priority weight is given to the data. Newer data is seen as more relevant and given more weight [58]. Exponential Smoothing is denoted as:

$$\hat{y}_{T+1|T} = \alpha y_T + \alpha(1-\alpha)y_{T-1} + \alpha(1-\alpha)^2 y_{T-2} \tag{19}$$

The forecast for time $T+1$ is built on a weighted average of all observations in the series $y_1, ..., y_T$. The parameter depends on the rate at which the weights drop.

2.3.4 Autoregressive Models (AR)

The AR(p) model does not reveal the relationship between a dependent variable and the independent variables that explain this variable, as in the multiple regression model. Still, it is different from the multiple regression model because it explains the relationship between the observed value of the same

variable for a certain t-period and the observation values of the previous periods. Autoregressive processes make intuitive sense since the next observed value is a tiny modification of the most recent observations; in other words, the current value of the series is linearly dependent on its previous value, with some random error. Because y_t is regressed on itself, this model is named an AR model. [61]. Denoted as:

$$y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} ... + \phi_p y_{t-p} + w_t, \tag{20}$$

Can be rephrased as:

$$y_t = \sum_{j=1}^{p} \phi_j y_{t-j} + w_t, \tag{21}$$

- $\phi_1, \phi_2, ..., \phi_p$ are fixed constants.
- w_t random variable with mean 0 and variance.

2.3.5 Moving Average (MA)

The Moving Average (MA) approach is the most straightforward and fundamental of all-time series forecasting algorithms. This model is used to analyze a univariate time series (which has just one variable). The output (or future) variable is supposed to have a linear relationship with an MA model's present and previous values. As a result, the new series is constructed using the average of the previous data. MA models are well-suited for recognizing and emphasizing trends and trend cycles. MA models are models in which the observation value in any period of a time series is expressed as a linear combination of the error terms of the same period and the error terms of a certain number of past periods. MA models are called first-order, second-order, and generally q-order MA models according to the number of past error terms they contain [62]. MA(q) is denoted as:

$$y_t = w_t + \theta_1 w_{t-1} + ... + \theta_q w_{t-q} \tag{22}$$

May be rephrased as:

$$y_t = \sum_{j=0}^{q} \theta_j w_{t-j} \tag{23}$$

- $\theta_1, \theta_2, ..., \theta_q$ fixed contants.
- w_t random variable with mean 0 and variance.

Considering the white noise series as a collection of stochastically uncorrelated data points that emerge at each time step and are merged with other data points to create the observation at time. Thus, moving average models are advantageous for modeling series that are impacted by a number of unexpected events that have both immediate and probable long-term consequences [62].

2.3.6 Autoregressive Moving Average (ARMA)

ARMA models are used to model stationary time series and combine AR and MA models. In these models, the observation value for any period of a time series is expressed as a linear combination of a

certain number of previous observation values and the error term. If the ARMA model is a combination of the p-term AR and the q-term MA model, it contains p+q terms and is written as ARMA(p,q) [63]. It is denoted as:

$$y_t = \underbrace{\phi_1 y_{t-1} + ... + \phi_p y_{t-p}}_{\text{AR}} + \underbrace{\theta_i w_{t-1} + ... + \theta_q w_{t-q}}_{\text{MA}} \tag{24}$$

2.3.7 Autoregressive Integrated Moving Average (ARIMA)

ARIMA models are applied to non-stationary series but converted to stationary by difference-taking. Models used to non-stationary series but converted to stationary by difference-taking are called non-stationary linear stochastic models. These models are AR, applied to series with a d-degree difference. The variable's value in the t-period is expressed as a linear function of a certain number of back-period values and the error term in the same period. The variable's value in the t-period is the error term in the same period and a certain number of back-period error terms. It is a combination of MA models in which it is expressed as a linear function [59]. The general representation of the models is ARIMA (p, d, q). Here, p and q are the degrees of the autoregressive (AR) model and the moving average (MA) model, respectively, and d is the degree of difference. ARIMA is denoted as:

$$y'_t = \phi_1 y'_{t-1} + ... + \phi_p y'_{t-p} + \theta_i w_{t-1} + ... + \theta_q w_{t-q} \tag{25}$$

- y'_t is differenced series

2.3.8 Vector Autoregression (VAR)

VAR is an econometric model that generalizes univariate AR models, giving the evolution and inter-dependence between multiple time series. VAR models are distinguished from univariate autoregressive models in that they may be used to evaluate and predict time series data with multiple variables. The fundamental distinction between the ARIMA family and VAR models is that all ARIMA models apply to univariate time series, while VAR models apply to multivariate time series. Additionally, ARIMA models are unidirectional, implying that the dependent variables are impacted only by their past or lag values. In contrast, VAR models are bidirectional, implying that a dependent variable is influenced either by its previous value or by the value of another variable, or by both [64]. If we consider the Y_t and X_t series, the VAR model is denoted as:

$$Y_t = \alpha + \sum_{i=1}^{m} \beta_j Y_{t-j} + \sum_{j=1}^{m} \delta_j X_{t-j} + \varepsilon_{1t} \tag{26}$$

$$X_t = \alpha + \sum_{j=1}^{m} \theta_j Y_{t-j} + \sum_{j=1}^{m} \vartheta_j X_{t-j} + \varepsilon_{2t} \tag{27}$$

2.3.9 Vector Autoregressive Moving Average (VARMA)

VARMA model is a combination of VAR and Vector Moving Average (VMA) models. It is possible to say that VARMA is a generalized version of ARMA for multivariate time series forecasting. As

ARMA, VARMA models are also represented with p, and q [65]. They can be denoted as:

$$y_t = c + \sum_{j=1}^{p} \Phi_j y_{t-j} + \sum_{k=1}^{q} \Theta_k \varepsilon_{t-k} + \varepsilon_t \tag{28}$$

2.3.10 Vector Autoregressive Moving Average with Exogenous Regressors (VARMAX)

Economic and financial variables are frequently contemporaneously connected and with each other's previous values. These sorts of temporal correlations may be modeled using the VARMAX technique. Factors outside the underlying process impact the variables of interest in many economic and financial applications. The VARMAX technique may be used to simulate the temporal relationship among dependent and independent variables as well as among dependent and independent variables [66]. VARMAX models are generally denoted as:

$$Y_t = \sum_{i=1}^{p} \Phi_i Y_{t-i} + \sum_{i=0}^{b-1} B_i X_{t-i} + \sum_{i=1}^{q} \Theta_i E_{t-i} + C + E_t \tag{29}$$

- Y_t denotes a vector containing n response variables
- X_t denotes a vector containing m exogenous variables
- p the number of preceding periods for the endogenous variables in the model
- q the number of preceding periods included in the moving average
- b the number of preceding periods of exogenous variables included
- Φ_i is an n^*n matrix of autoregressive parameters
- B_i is an $n * m$ matrix of exogenous variable parameters
- Θ_i is an n^*n matrix of moving average parameters
- E_t is the residual Y_t, $\left(Y_t - \hat{Y}_t\right)$.

2.3.11 Prophet

The Prophet model is an open-source forecasting application developed by the Facebook data science team. It includes procedures that enable making annual, quarterly, weekly, and daily forecasts on non-linear time series data. In addition, the Prophet model can handle data loss and extreme values successfully. The model is actively used in many applications within Facebook today [19]. The model has three main components: seasonality, holidays, and trend. These components' combination under the Prophet model is denoted as:

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t \tag{30}$$

Where:

- $g(t)$ shifts in non-periodic manner
- $s(t)$ periodical shifts
- $h(t)$ effects of holidays
- ϵ_t idiosyncratic shifts

2.4 Model Selection and Evaluation

After modeling the empirical data using Machine Learning, we estimate the adequacy of our model. Numerous performance assessment techniques have been suggested. Note that measures must employ different facts depending on the dataset [59]. We apply two evaluation metrics for both Machine Learning and Time-Series models: Mean Absolute Percentage Error (MAPE) and Root Mean Squared Error (RMSE). On the other hand, a Cross-Validation technique is used rather than testing the model's fit on one set of observations. Cross-validation is a model selection approach used to estimate the error of a test run on a machine learning model. Cross-validation is a technique that generates sample observation segments, referred to as validation sets, from the training data set. After fitting a model to the training data, its performance is evaluated against each subsequent validation set, providing a more accurate estimate of the model's performance when attempting to predict new observations. The number of partitions required depends on the number of observations in the sample dataset. The choice about bias-variance balance alters as more divides result in a lower bias and more variance.

2.4.1 RMSE

The root mean square error (RMSE) is the standard deviation of the estimate errors (residues) [59],

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}} \tag{31}$$

- $\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n$ are predicted values
- $y_1, y_2, \ldots, y_n$ are observed values
- n is the number of observation

2.4.2 MAPE

The mean absolute percentage error (MAPE) is widely used to quantify prediction accuracy in regression and time series models. When MAPE is used to compare the accuracy of estimators, it is biased in that it consistently selects a technique with very low estimates. This tiny but significant issue may be resolved by calculating the ratio of expected to actual values [59].

$$MAPE = \frac{100\%}{n} \sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{y_i} \right| \tag{32}$$

- n is the number of observation

- $\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n$ are predicted values
- $y_1, y_2, \ldots, y_n$ are observed values

2.4.3 Time Series Cross Validation

Time series cross-validation is a more complicated variant of training/test sets. This process consists of several test sets, each containing a single observation. The equivalent training set contains only observations that arise before the test set's observations. As a result, forecasts cannot be generated using future data. The first observations are not included in the test set since a valid estimate cannot be obtained from a tiny training sample [74]. The following graphic depicts a series of training and test sets, with blue observations representing the training sets and orange observations representing the test sets:

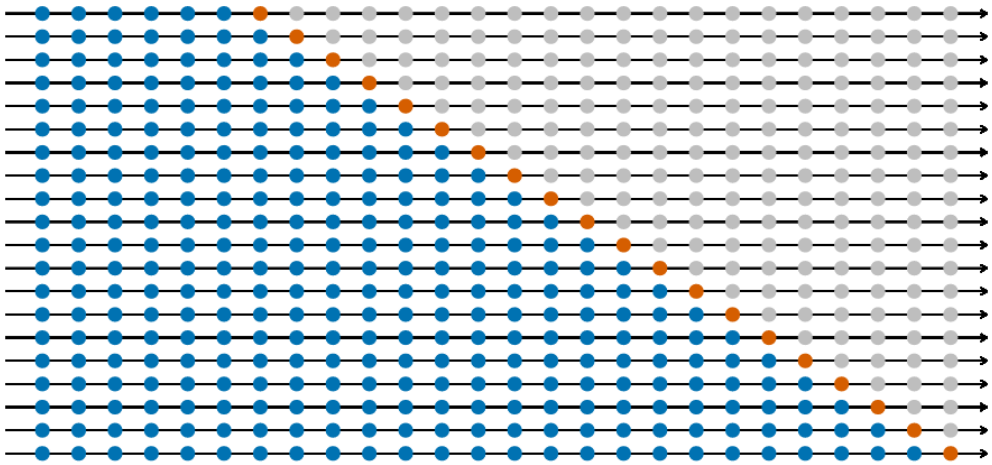


Figure 2: Time Series Cross Validation

2.5 Feature Selection

The process of choosing and locating the most valuable characteristics in a data collection is called feature selection. This technique has a significant impact on the machine learning model's performance. There are other ways for this selection process. However, the one employed out this research is Permutation Feature Selection.

2.5.1 Permutation Feature Importance

The feature's relevance is determined by the increase in the model's prediction error when the order of the data underneath the feature is shuffled. A feature is considered important if combining its values raises a model error since the model relies on the feature for prediction in this situation. A feature is trivial if mixing its values results in no change in the model error since it then dismisses the feature for forecasts. Breiman pioneered the use of permutation features to determine the relevance of Random Forests [9].

3 Experiments and Results

The findings of the studies on the literature review and the methodologies specified will be shown and discussed in this section of the study. Python was used to find the best date range for the learning set, to analyze the results and accomplishments of all the models stated above, to find the best feature sets, and to choose the best hyperparameters for all the models.

3.1 Dataset

The dataset used in this study has been taken through Binance API with Python. The features and their explanations are given below:

1. **Open Time**: The starting time of the time interval
2. **Open**: The opening price at the beginning of that specific time interval
3. **High**: The maximum price of the stock within that specific time interval
4. **Low**: The maximum price of the stock within that specific time interval
5. **Close**: The last price of the stock within that time interval
6. **Volume**: The trading volume of the stock in that time interval
7. **Close Time**: Closing time of the time interval
8. **Number of Trades**: Number of trades in that time interval

3.2 Target (Independent) Variable

As the target variable, Close variable will be used to calculate the return. A return is the change in price of a commodity, investment over time, expressed in either money amount terms or terms of percentage. Return is denoted as:

$$\text{Return } = \frac{EndingPrice - StartingPrice}{StartingPrice} \tag{33}$$

3.3 Feature Engineering

In machine learning, the real-world data provided to the models is in the form of feature vectors, and these feature vectors are retrieved from the raw data in order to train the models. Feature engineering is the practice of identifying the most useful features from data in order to improve the performance of a machine learning model or job. The number of features is just as critical as the features themselves; if the features in the feature vectors that represent the model's input data are insufficient, the model will fail to perform its primary task; if an attribute vector contains redundant and irrelevant features, the model will still fail to produce correct results. Not only the model, but also the features used to describe the data are chosen throughout the machine learning process. While well chosen features simplify future modeling phases and enhance the capacity of the eventual model to accomplish the

target job, if features are not chosen appropriately, a considerably more complicated model may be necessary to attain the same level of performance. The list of created features with their equations are given below:

- **Day of Week**: The number of the day is extracted from "Open Time" column. The numbers vary from 0 to 6.
  - 0 - Monday
  - 1 - Tuesday
  - 2 - Wednesday
  - 3 - Thursday
  - 4 - Friday
  - 5 - Saturday
  - 6 - Sunday
- **Week Number**: Number of the week for the year. Numbers vary from 0 to 51
- **Hour**: Hour within the day. Numbers vary from 0 to 23
- **Rolling Mean Hourly**: The average of last n values in unweighted manner, and n is based on time intervals. For example, if the range between two data ticks is 5 minutes:

$$n = \frac{60}{5} \tag{34}$$

- **Rolling Median Hourly**: The median of last n values in unweighted manner. n is decided the same way.
- **Rolling Max Hourly**: The maximum of the last n values. n is decided the same way.
- **Rolling Standard Deviation Hourly**: The standard deviation of the last n values. n is decided the same way.
- **Hourly Range Close**: Hourly range within the last n data ticks. This might give idea about how volatile was the cryptocurrency within the hour.

$$HourlyRangeClose = \max\left(X_t, X_{t-1}, ..., X_{t-n}\right) - \min X_t, X_{t-1}, ..., X_{t-n} \tag{35}$$

  - X denotes Close value.
  - t denotes time point.
  - n denotes previous number of time points.
- **Hourly Range Number of Trades**: Hourly range of number of trades within the hour. Also this might give idea about the external factors that might effect the number of trades such as

Telegram groups or Social Media Platforms. A sudden increase in the number of trades might mean that there is an update or some news about the cryptocurrency.

$$= HourlyRangeNumberofTrades = \max X_t, X_{t-1}, X_{t-2}, X_{t-3} - \min X_t, X_{t-1}, X_{t-2}, X_{t-3} \tag{36}$$

  - X denotes number of trades.
  - t denotes time point.
  - n denotes previous number of time points.

- **Number of Trades Percentage Change**: This feature gives a number of percental change in trades within last tick. The aim of this feature is to catch sudden changes.

$$NumberofTradesPercentageChange = \frac{X_t - X_{t-1}}{X_t} \times 100 \tag{37}$$

  - X_t denotes the number of trades in the current data point
  - X_{t-1} denotes the number of trades in the previous data point

- **Volume Percentage Change**: This feature gives a number of percental change in volume within last tick. The aim of this feature is to catch sudden changes.

$$VolumePercentageChange = \frac{X_t - X_{t-1}}{X_t} \times 100 \tag{38}$$

  - X_t denotes the amount of volume in the current data point
  - X_{t-1} denotes the amount of volume in the previous data point

- **Month Start Flag**: This feature gives information about if the date is around the start (first week) of the month. The aim is try to catch if number of trades or amount of volume is increasing during the first days of the month. Numbers vary from 0 to 1, subject to True and False.

- **Month End Flag**: This feature gives information about if the date is around the end (last week) of the month. The aim is try to catch if number of trades or amount of volume is increasing or decreasing through the last days of the month. Numbers vary from 0 to 1, subject to True and False.

- **Weekend Flag**: This feature gives information about whether the date is at the weekend or in the weekdays. Numbers vary from 0 to 1, subject to True and False.

- **Business Hours**: This feature gives information about whether it is business hours (9-17). Numbers vary from 0 to 1, subject to True and False.

- **Weighted Average of Close**: The logic behind this feature is very similar to Exponential Smoothing. Calculating the average of last n values by assigning more weight to recent values.

$$\bar{x} = \sum_{i=1}^{n} w_i x_i \tag{39}$$

3.4 Optimal Training Set Experiments

Apart from comparing machine learning and time series prediction models and combining their strengths, the effect of the date range of the selected training data on the prediction results will be examined in this study. The best results for the date range will be identified, and all models will be learned on this training data. While the machine learning technique addresses the issue of how it was learned, the ideal date range for the training data addresses the question of where it was taught. To ensure that all the models perform at their highest level, which will also provide of selecting the optimal dataset for maximum performance, time-series cross-validation (mentioned in 2.4.3 on page 32), and Permutation Feature Importance (mentioned in 2.5.1 on page 32) were employed. Experimented date ranges for training data are given below:

- December 1, 2020 - December 1, 2021 (15 - minute intervals)
- January 1, 2021 - December 1, 2021 (15 - minute intervals)
- Februar 1, 2021 - December 1, 2021 (15 - minute intervals)
- March 1, 2021 - December 1, 2021 (15 - minute intervals)
- April 1, 2021 - December 1, 2021 (15 - minute intervals)
- May 1, 2021 - December 1, 2021 (15 - minute intervals)
- June 1, 2021 - December 1, 2021 (15 - minute intervals)
- July 1, 2021 - December 1, 2021 (15 - minute intervals)
- August 1, 2021 - December 1, 2021 (15 - minute intervals)
- September 1, 2021 - December 1, 2021 (15 - minute intervals)

The results for each time interval is given the following chapters.

3.4.1 Experiments for December, 2020 - December, 2021

The table below shows the results of the experiments for models trained on the historical data between December 1, 2020 - December 1, 2021. The time step in this training dataset is 15 minutes. With this date interval and time steps, this dataset has 35040 observations. All models are evaluated with both MAPE and RMSE.

Model	RMSE	MAPE
Lasso Regression	0.0044	1.1504
Extra Trees Regressor	0.0034	2.9856
Random Forest Regressor	0.0033	3.0766
Light Gradient Boosting Machine	0.0034	3.1015
Gradient Boosting Regressor	0.0036	3.2124
K Neighbors Regressor	0.0039	3.2272
Huber Regressor	0.0039	3.5873
AdaBoost Regressor	0.0043	3.588
Ridge Regression	0.0037	3.6113
Linear Regression	0.004	4.4087
Decision Tree Regressor	0.0048	5.9403
Mean of the Models	0.0038	3.4444
Standard Deviation of the Models	0.0004	1.0894

Table 1: Experiment of Models for the date range December 1, 2020 - December 1, 2021 (with 15-minute time steps)

3.4.2 Experiments for January 1, 2021 - December 1, 2021

The table below shows the results of the experiments for models trained on the historical data between January 1, 2021 - December 1, 2021. The time step in this training dataset is 15 minutes. With this date interval and time steps, this dataset has 32064 observations. All models are evaluated with both MAPE and RMSE.

Model	RMSE	MAPE
Lasso Regression	0.0044	1.1164
Extra Trees Regressor	0.0039	2.8993
Random Forest Regressor	0.0033	3.15
Light Gradient Boosting Machine	0.0036	3.2039
Gradient Boosting Regressor	0.0034	3.2569
K Neighbors Regressor	0.0034	3.3261
Huber Regressor	0.0038	3.4976
AdaBoost Regressor	0.0042	3.6177
Ridge Regression	0.0037	3.6275
Linear Regression	0.0045	4.9957
Decision Tree Regressor	0.0048	6.5365
Mean of the Models	0.0039	3.5661
Standard Deviation of the Models	0.0004	1.2714

Table 2: Experiment of Models for the date range January 1, 2021 - December 1, 2021 (with 15-minute time steps)

3.4.3 Experiments for February 1, 2021 - December 1, 2021

The table below shows the results of the experiments for models trained on the historical data between February 1, 2021 - December 1, 2021. The time step in this training dataset is 15 minutes. With this date interval and time steps, this dataset has 29088 observations. All models are evaluated with both MAPE and RMSE.

Model	**RMSE**	**MAPE**
Lasso Regression	0.0044	1.095
Extra Trees Regressor	0.0034	3.064
Random Forest Regressor	0.0035	3.2856
Light Gradient Boosting Machine	0.0036	3.3391
Gradient Boosting Regressor	0.0035	3.3941
K Neighbors Regressor	0.0044	3.467
Huber Regressor	0.0039	3.5919
AdaBoost Regressor	0.0038	3.6226
Ridge Regression	0.0038	3.8857
Linear Regression	0.0051	4.1797
Decision Tree Regressor	0.0049	6.4051
Mean of the Models	0.0040	3.5754
Standard Deviation of the Models	0.0005	1.1704

Table 3: Experiment of Models for the date range February 1, 2021 - December 1, 2021 (with 15-minute time steps)

3.4.4 Experiments for March 1, 2021 - December, 2021

The table below shows the results of the experiments for models trained on the historical data between March 1, 2021 - December 1, 2021. The time step in this training dataset is 15 minutes. With this date interval and time steps, this dataset has 26400 observations. All models are evaluated with both MAPE and RMSE.

Model	RMSE	MAPE
Lasso Regression	0.0045	1.0429
Extra Trees Regressor	0.0034	2.9168
Random Forest Regressor	0.0035	3.0237
Light Gradient Boosting Machine	0.0035	3.0974
Gradient Boosting Regressor	0.0044	3.5433
K Neighbors Regressor	0.0037	3.6776
Huber Regressor	0.0039	3.6842
AdaBoost Regressor	0.0039	3.8827
Ridge Regression	0.0038	3.9043
Linear Regression	0.0102	3.9537
Decision Tree Regressor	0.0047	5.7086
Mean of the Models	0.0045	3.4941
Standard Deviation of the Models	0.0018	1.0543

Table 4: Experiment of Models for the date range March 1, 2021 - December 1, 2021 (with 15-minute time steps)

3.4.5 Experiments for April 1, 2021 - December 1, 2021

The table below shows the results of the experiments for models trained on the historical data between April 1, 2021 - December 1, 2021. The time step in this training dataset is 15 minutes. With this date interval and time steps, this dataset has 23424 observations. All models are evaluated with both MAPE and RMSE.

Model	RMSE	MAPE
Lasso Regression	0.0038	1.0414
Extra Trees Regressor	0.0029	2.3807
Random Forest Regressor	0.0028	2.4718
Light Gradient Boosting Machine	0.0029	2.5219
Gradient Boosting Regressor	0.0032	2.643
K Neighbors Regressor	0.003	2.6469
Huber Regressor	0.0034	2.6677
AdaBoost Regressor	0.0032	2.7276
Ridge Regression	0.0037	3.3061
Linear Regression	0.0033	3.7387
Decision Tree Regressor	0.0042	4.9629
Mean of the Models	0.0033	2.8280
Standard Deviation of the Models	0.0004	0.9210

Table 5: Experiment of Models for the date range April 1, 2021 - December 1, 2021 (with 15-minute time steps)

3.4.6 Experiments for May 1, 2021 - December 1, 2021

The table below shows the results of the experiments for models trained on the historical data between May 1, 2021 - December 1, 2021. The time step in this training dataset is 15 minutes. With this date interval and time steps, this dataset has 20544 observations. All models are evaluated with both MAPE and RMSE.

Model	**RMSE**	**MAPE**
Lasso Regression	0.0036	1.065
Extra Trees Regressor	0.0032	1.7993
Random Forest Regressor	0.0028	2.4127
Light Gradient Boosting Machine	0.0027	2.5298
Gradient Boosting Regressor	0.0028	2.5918
K Neighbors Regressor	0.0031	2.6097
Huber Regressor	0.0029	2.7443
AdaBoost Regressor	0.0031	2.9846
Ridge Regression	0.0031	3.1443
Linear Regression	0.0036	3.8844
Decision Tree Regressor	0.004	5.4612
Mean of the Models	0.0031	2.8388
Standard Deviation of the Models	0.0003	1.0749

Table 6: Experiment of Models for the date range May 1, 2021 - December 1, 2021 (with 15-minute time steps)

3.4.7 Experiments for June 1, 2021 - December 1, 2021

The table below shows the results of the experiments for models trained on the historical data between June 1, 2021 - December 1, 2021. The time step in this training dataset is 15 minutes. With this date interval and time steps, this dataset has 17568 observations. All models are evaluated with both MAPE and RMSE.

Model	RMSE	MAPE
Lasso Regression	0.0036	1.0703
Extra Trees Regressor	0.0031	2.1566
Random Forest Regressor	0.0027	2.4321
Light Gradient Boosting Machine	0.0028	2.4517
Gradient Boosting Regressor	0.0026	2.5219
K Neighbors Regressor	0.0031	2.5426
Huber Regressor	0.0027	2.5899
AdaBoost Regressor	0.0035	2.8051
Ridge Regression	0.0031	3.0622
Linear Regression	0.0032	3.292
Decision Tree Regressor	0.0039	4.7022
Mean of the Models	0.0031	2.6933
Standard Deviation of the Models	0.0003	0.8341

Table 7: Experiment of Models for the date range June 1, 2021 - December 1, 2021 (with 15-minute time steps)

3.4.8 Experiments for July 1, 2021 - December 1, 2021

The table below shows the results of the experiments for models trained on the historical data between July 1, 2021 - December 1, 2021. The time step in this training dataset is 15 minutes. With this date interval and time steps, this dataset has 14688 observations. All models are evaluated with both MAPE and RMSE.

Model	RMSE	MAPE
Lasso Regression	0.0035	1.1298
Extra Trees Regressor	0.0027	2.462
Random Forest Regressor	0.0027	2.4806
Light Gradient Boosting Machine	0.0028	2.495
Gradient Boosting Regressor	0.0026	2.5105
K Neighbors Regressor	0.0031	2.5765
Huber Regressor	0.003	2.7259
AdaBoost Regressor	0.0035	2.8468
Ridge Regression	0.0035	3.1523
Linear Regression	0.003	3.2987
Decision Tree Regressor	0.0038	4.9868
Mean of the Models	0.0031	2.7877
Standard Deviation of the Models	0.0003	0.8735

Table 8: Experiment of Models for the date range July 1, 2021 - December 1, 2021 (with 15-minute time steps)

3.4.9 Experiments for August 1, 2021 - December 1, 2021

The table below shows the results of the experiments for models trained on the historical data between August 1, 2021 - December 1, 2021. The time step in this training dataset is 15 minutes. With this date interval and time steps, this dataset has 11712 observations. All models are evaluated with both MAPE and RMSE.

Model	**RMSE**	**MAPE**
Lasso Regression	0.0036	1.0642
Extra Trees Regressor	0.0031	2.2711
Random Forest Regressor	0.0031	2.4019
Light Gradient Boosting Machine	0.003	2.5228
Gradient Boosting Regressor	0.0028	2.6788
K Neighbors Regressor	0.0029	2.6895
Huber Regressor	0.0028	2.755
AdaBoost Regressor	0.0029	2.7901
Ridge Regression	0.003	3.1227
Linear Regression	0.0035	3.3862
Decision Tree Regressor	0.004	5.7802
Mean of the Models	0.0031	2.8602
Standard Deviation of the Models	0.0003	1.0806

Table 9: Experiment of Models for the date range August 1, 2021 - December 1, 2021 (with 5-minute time steps)

3.4.10 Experiments for September 1, 2021 - December 1, 2021

The table below shows the results of the experiments for models trained on the historical data between September 1, 2021 - December 1, 2021. The time step in this training dataset is 15 minutes. With this date interval and time steps, this dataset has 8736 observations. All models are evaluated with both MAPE and RMSE.

Model	RMSE	MAPE
Lasso Regression	0.0035	1.1378
Extra Trees Regressor	0.003	1.8434
Random Forest Regressor	0.003	2.6791
Light Gradient Boosting Machine	0.0028	2.8631
Gradient Boosting Regressor	0.0028	2.985
K Neighbors Regressor	0.003	3.001
Huber Regressor	0.003	3.0278
AdaBoost Regressor	0.0028	3.1425
Ridge Regression	0.0028	3.1452
Linear Regression	0.0035	3.4628
Decision Tree Regressor	0.0039	5.6433
Mean of the Models	0.0031	2.9937
Standard Deviation of the Models	0.0003	1.0518

Table 10: Experiment of Models for the date range September 1, 2021 - December 1, 2021 (with 5-minute time steps)

3.4.11 Results of Experiments for Training Set

MAPE metric was used to determine the optimal interval for the training set. The mean and standard deviation of MAPE for each month were evaluated. After the evaluation, it is determined that the best interval for training data is June 1, 2021 - December 1, 2021, which has 17568 observations. The results are given in the table below:

Date Interval	Mean MAPE	Standard Deviation MAPE
December 2020 - December 2021	3.4444	1.0894
January 2021 - December 2021	3.5661	1.2714
February 2021 - December 2021	3.5754	1.1704
March 2021 - December 2021	3.4941	1.0543
April 2021 - December 2021	2.8280	0.9210
May 2021 - December 2021	2.8388	1.0749
June 2021 - December 2021	**2.6933**	**0.8341**
July 2021 - December 2021	2.7877	0.8735
August 2021 - December 2021	2.8602	1.0806
September 2021 - December 2021	2.9937	1.0518

Table 11: Mean and Standard Deviation of MAPE for Each Month Interval

3.5 Experiments for Machine Learning Models

Results of all the Machine Learning models that were mentioned in 2.1 on page 14 will be demonstrated in this section. To make the evaluation step fairly, models were trained with the historical

data of June 1, 2021 - December 1, 2021, and tested on December 13, 2021 - December 14, 2021, with 10-folds time-series cross-validation. The aim of evaluating all models with different training and test sets is to ensure that the model is robust and will perform well on unseen data. 10-folds time-series cross-validation is also employed throughout the application of Permutation Feature Importance, and the most valuable features are given below. All of the features are explained in chapter 3.3 on page 33.

- Rolling Median Hourly
- Weighted Average
- Volume Change Percentage
- Number of Trades
- Volume
- Number of Trades Change Percentage
- Rolling Standard Deviation Hourly
- Number of Trades - Min Max Range
- Close Price - Min Max Range
- Rolling Mean Hourly
- Hour
- Week Number
- Day of Week

All Machine Learning models were trained with 13 features and 17568 observations. In the following sections, MAPE and RMSE of the results and visualizations of the results will be demonstrated for each model. The top five best-performing models are selected and blended.

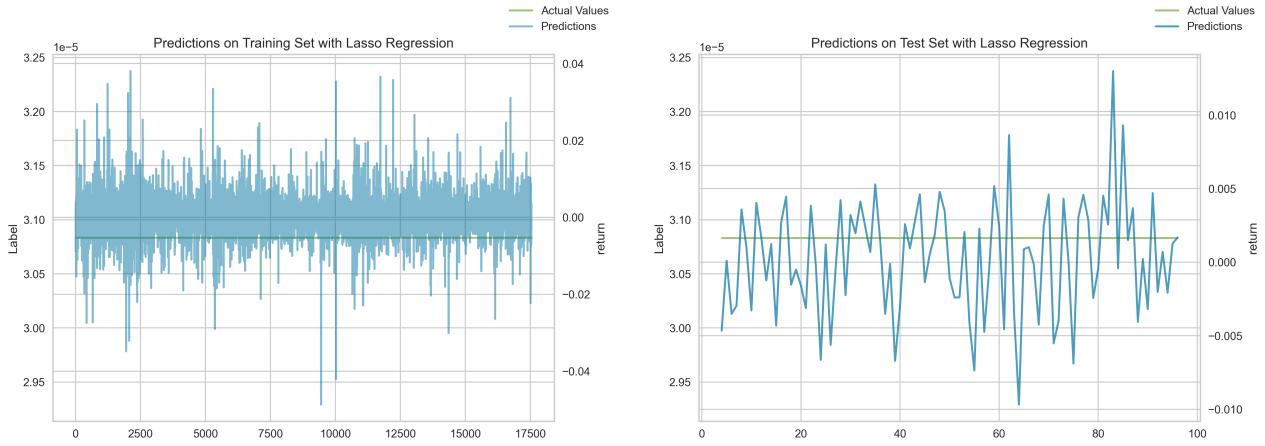
3.5.1 Experiment of Lasso Regression

The model is trained with 13 features and 17568 observations with 15-minute time steps. The model's performance is evaluated with MAPE and RMSE throughout all 10-folds. Mean MAPE, RMSE, and Standard Deviation of MAPE and RMSE were calculated for evaluation. The aim of calculating the model's mean and standard deviation is to ensure that the model is robust and stable within each fold. Results of Lasso Regression are given below:

Fold	RMSE	MAPE
0	0.0053	1.004
1	0.003	1.0219
2	0.0041	1.0461
3	0.0035	1.0293
4	0.0033	1.0669
5	0.004	1.029
6	0.0035	1.0194
7	0.0036	1.0945
8	0.003	1.2125
9	0.0037	1.0256
Mean of the Models	0.0037	1.0549
Standard Deviation of the Models	0.0006	0.0580

Table 12: RMSE and MAPE scores of Lasso Regression for Each Fold

The table above shows good performance within MAPE metrics. But RMSE is not very promising. Thus, additional visualizations are used to ensure the model is performing well.



(a) Lasso Regression on Training Set.

(b) Lasso Regression on Test Set.

Figure 3: Lasso Regression's Performance on both Training and Test Sets

By looking at the figure above, it is possible to see that Lasso Regression predicted the same constant value for each observation. From this, it can be understood that just looking at the metrics will not be enough: visualizations should support. Thus, Lasso Regression is not used in further steps.

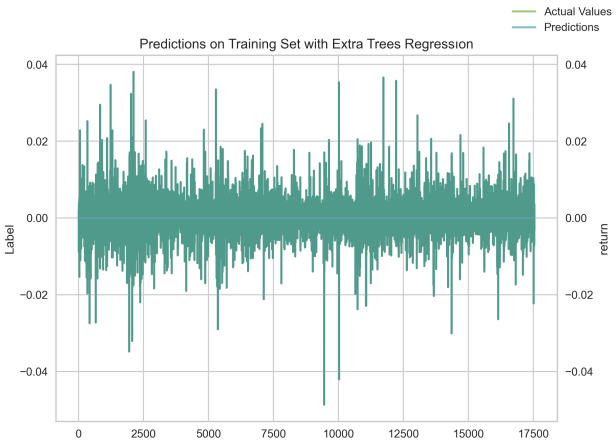
3.5.2 Experiment of Extra Trees

The model is trained with 13 features and 17568 observations with 15-minute time steps. The model's performance is evaluated with MAPE and RMSE throughout all 10-folds. Mean MAPE, RMSE,

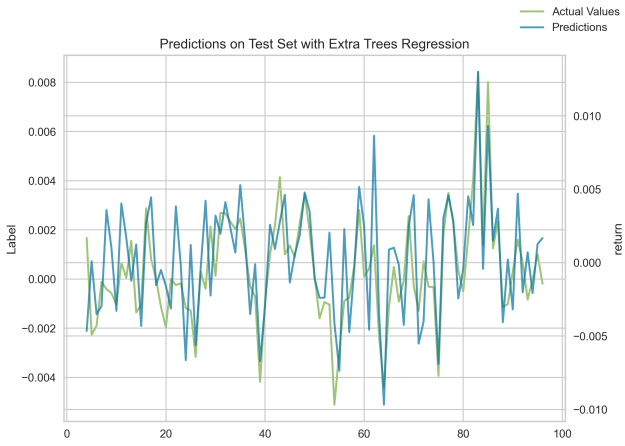
Standard Deviation of MAPE, and RMSE were calculated for evaluation. The aim of calculating the mean and standard deviation of the model is to make sure that the model is robust and stable within each fold. Results of Extra Trees are given below:

Fold	**RMSE**	**MAPE**
0	0.0042	2.3539
1	0.0024	3.2646
2	0.003	2.2322
3	0.0026	2.8609
4	0.0023	2.1529
5	0.0031	2.4872
6	0.0024	2.6247
7	0.0027	2.4403
8	0.0022	4.029
9	0.0026	1.9028
Mean of the Models	0.0027	2.6348
Standard Deviation of the Models	0.0005	0.5882

Table 13: RMSE and MAPE scores of Extra Trees Regression for Each Fold



(a) Extra Trees Regression on Training Set.



(b) Extra Trees Regression on Test Set.

Figure 4: Extra Trees Regression's Performance on both Training and Test Sets

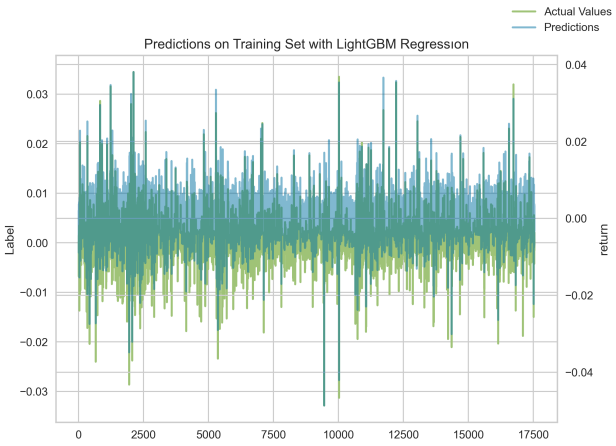
In figure 4a it is impossible to separate the lines of predictions and actual values, which is a good sign. But it is not possible to be sure about the model's performance without checking the performance on the test set. Thus, in figure 4b, we can see that the Extra Trees Regressor is performing well on the test set too.

3.5.3 Experiment of LightGBM

The model is trained with 13 features and 17568 observations with 15-minute time steps. The model's performance is evaluated with MAPE and RMSE throughout all 10-folds. Mean MAPE, RMSE, Standard Deviation of MAPE, and RMSE were calculated for evaluation. The aim of calculating the mean and standard deviation of the model is to make sure that the model is robust and stable within each fold. Results of LightGBM are given below:

Fold	**RMSE**	**MAPE**
0	0.0044	2.8626
1	0.0024	5.1535
2	0.003	2.4974
3	0.0026	2.8452
4	0.0025	2.1109
5	0.0031	2.3745
6	0.0024	2.1937
7	0.0028	2.2953
8	0.0023	5.4166
9	0.0027	1.9802
Mean of the Models	0.0028	2.9729
Standard Deviation of the Models	0.0005	1.1888

Table 14: RMSE and MAPE scores of LightGBM Regression for Each Fold



(a) LightGBM Regression on Training Set.

(b) LightGBM Regression on Test Set.

Figure 5: LightGBM Regression's Performance on both Training and Test Sets

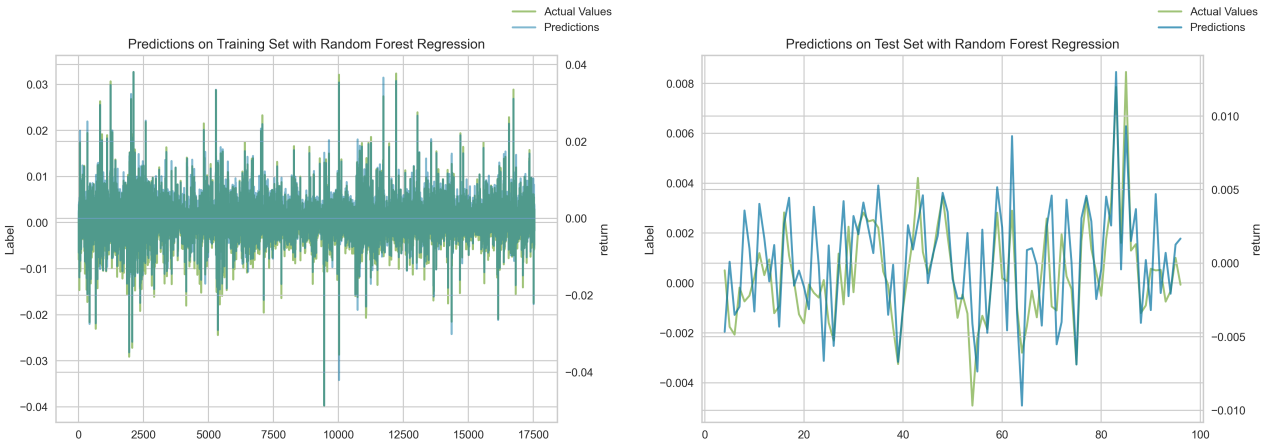
Standard deviation must be low within the folds to perform at a high level with these algorithms. Each time models make predictions on unseen data, models should perform at a good but stable level to be robust. In figure 5a and 5b, LightGBM performs well in both training and test data.

3.5.4 Experiment of Random Forest

The model is trained with 13 features and 17568 observations with 15-minute time steps. The model's performance is evaluated with MAPE and RMSE throughout all 10-folds. Mean MAPE, RMSE, Standard Deviation of MAPE, and RMSE were calculated for evaluation. The aim of calculating the mean and standard deviation of the model is to make sure that the model is robust and stable within each fold. Results of Random Forest are given below:

Fold	**RMSE**	**MAPE**
0	0.0042	2.5888
1	0.0024	3.8524
2	0.003	1.9868
3	0.0026	2.5365
4	0.0024	2.1586
5	0.0032	2.4872
6	0.0025	2.9399
7	0.0028	2.7956
8	0.0023	3.7288
9	0.0027	1.971
Mean of the Models	0.0028	2.7045
Standard Deviation of the Models	0.0005	0.6221

Table 15: Random Forest Regression's Performance on both Training and Test Sets



(a) Random Forest Regression on Training Set.

(b) Random Forest Regression on Test Set.

Figure 6: Random Forest Regression's Performance on both Training and Test Sets

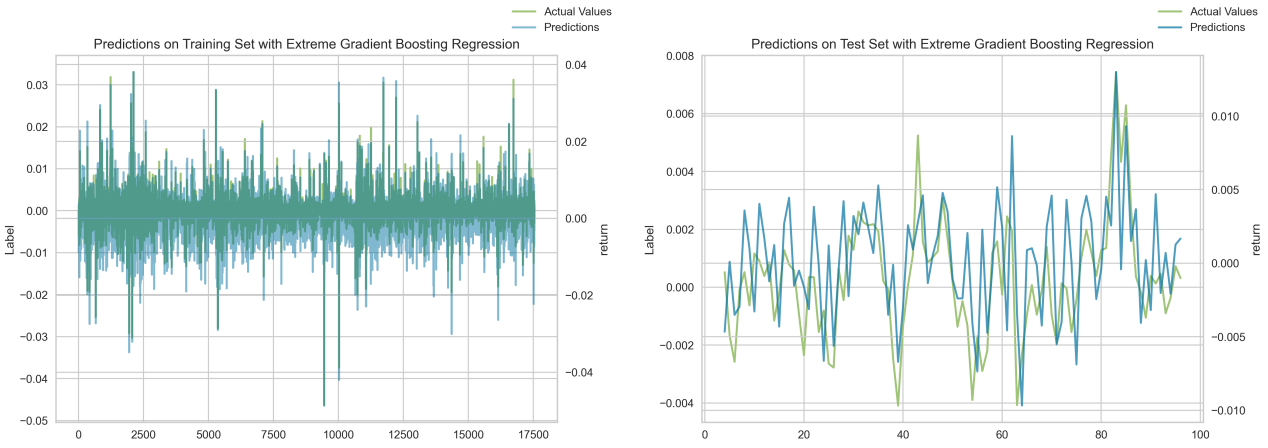
Random Forest Regression also performs well on both training data and test data. Both in figure **??** and **??**, the lines are overlapping which means the model is following the trend in both plots.

3.5.5 Experiment of Extreme Gradient Boosting

The model is trained with 13 features and 17568 observations with 15-minute time steps. The model's performance is evaluated with MAPE and RMSE throughout all 10-folds. Mean MAPE, RMSE, Standard Deviation of MAPE, and RMSE were calculated for evaluation. The aim of calculating the mean and standard deviation of the model is to make sure that the model is robust and stable within each fold. Results of Extreme Gradient Boosting are given below:

Fold	**RMSE**	**MAPE**
0	0.0042	2.4969
1	0.0024	5.0354
2	0.0031	2.0864
3	0.0027	2.8286
4	0.0025	2.2181
5	0.0032	2.5816
6	0.0027	2.7106
7	0.0029	2.2475
8	0.0024	4.1457
9	0.0028	2.0715
Mean of the Models	0.0028	2.8422
Standard Deviation of the Models	0.0005	0.9285

Table 16: Extreme Gradient Boosting's Performance on both Training and Test Sets



(a) Extreme Gradient Boosting on Training Set.

(b) Extreme Gradient Boosting on Test Set.

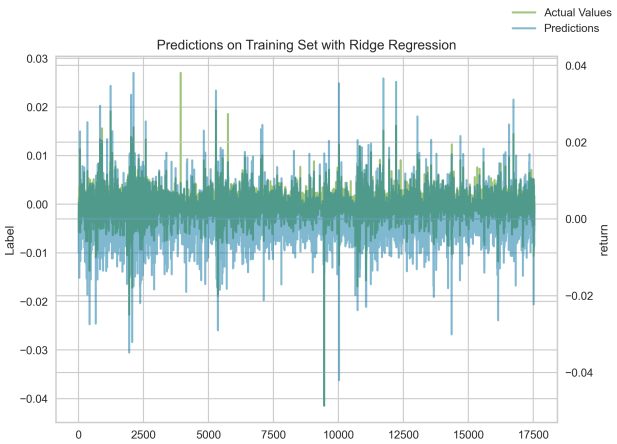
Figure 7: Extreme Gradient Boosting's Performance on both Training and Test Sets

With the Standard Deviation MAPE 0.9285 and RMSE with 0.0005, and overlapping lines in figure 7a with tight lines in figure **??**, the Extreme Gradient Boosting algorithm is performing well on both seen and unseen data.

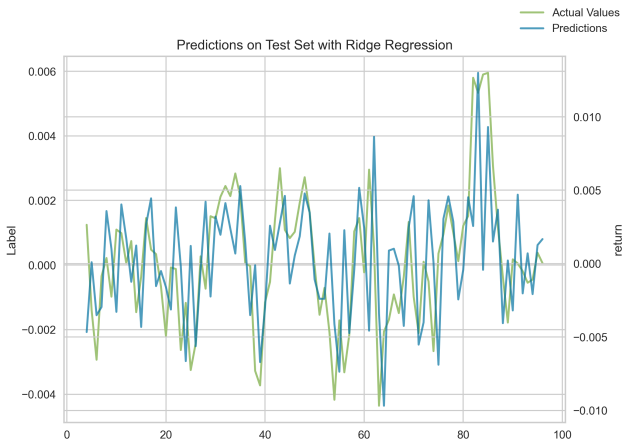
3.5.6 Experiment of Ridge Regression

The model is trained with 13 features and 17568 observations with 15-minute time steps. The model's performance is evaluated with MAPE and RMSE throughout all 10-folds. Mean MAPE, RMSE, Standard Deviation of MAPE, and RMSE were calculated for evaluation. The aim of calculating the mean and standard deviation of the model is to make sure that the model is robust and stable within each fold. Results of Ridge Regression are given below:

Fold	**RMSE**	**MAPE**
0	0.0044	2.3056
1	0.0029	4.3775
2	0.0034	3.1719
3	0.003	2.9904
4	0.0028	2.3448
5	0.0035	2.8321
6	0.0029	2.764
7	0.0031	2.0848
8	0.0026	3.642
9	0.0031	2.2735
Mean of the Models	0.0031	2.8786
Standard Deviation of the Models	0.0004	0.6747



(a) Ridge Regression on Training Set.



(b) Ridge Regression on Test Set.

Figure 8: Ridge Regression's Performance on both Training and Test Sets

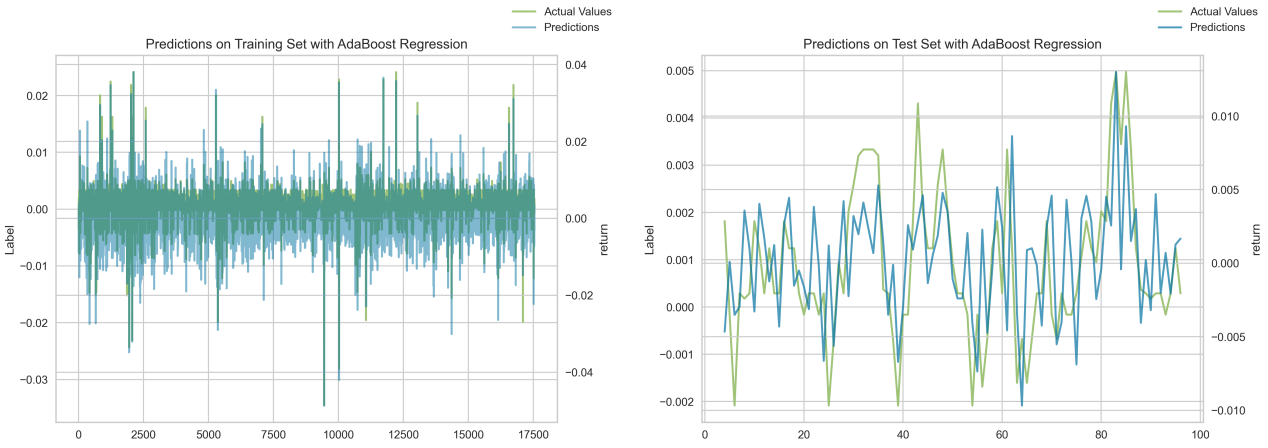
With the Standard Deviation MAPE 0.6747 and RMSE with 0.0004, and overlapping lines in figure 8a with very tight lines in figure 8b, the Ridge Regression algorithm is performing well on both seen and unseen data.

3.5.7 Experiment of AdaBoost Regression

The model is trained with 13 features and 17568 observations with 15-minute time steps. The model's performance is evaluated with MAPE and RMSE throughout all 10-folds. Mean MAPE, RMSE, Standard Deviation of MAPE, and RMSE were calculated for evaluation. The aim of calculating the mean and standard deviation of the model is to make sure that the model is robust and stable within each fold. Results of AdaBoost Regression are given below:

Fold	**RMSE**	**MAPE**
0	0.0045	2.6796
1	0.0027	4.5383
2	0.0034	1.976
3	0.0031	2.3948
4	0.0028	2.593
5	0.0035	2.1883
6	0.003	3.5833
7	0.0032	2.2197
8	0.0026	3.3237
9	0.0032	2.4064
Mean of the Models	0.0032	2.7903
Standard Deviation of the Models	0.0005	0.7533

Table 17: AdaBoost Regression's Performance on both Training and Test Sets



(a) AdaBoost Regression on Training Set.

(b) AdaBoost Regression on Test Set.

Figure 9: AdaBoost Regression's Performance on both Training and Test Sets

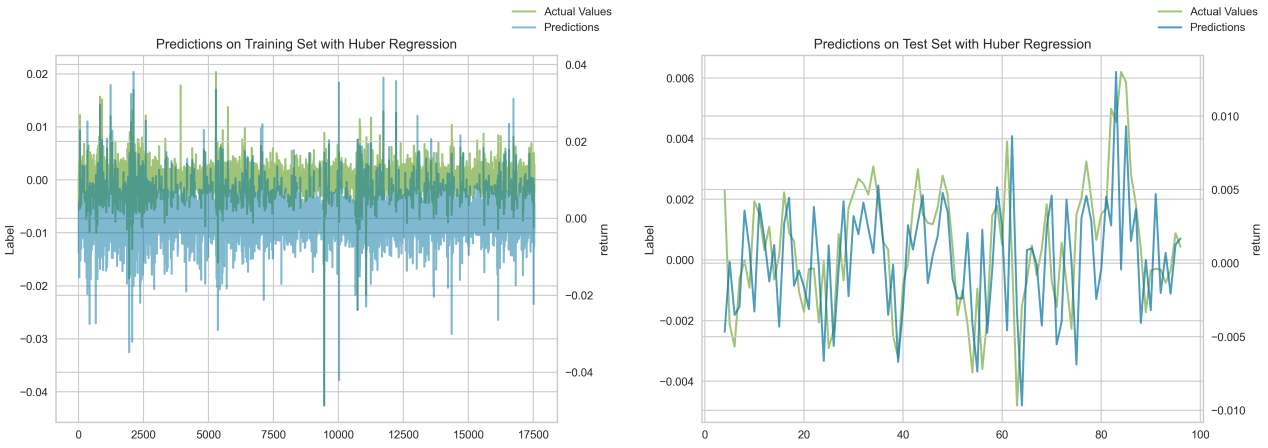
With the Standard Deviation MAPE 0.7533 and RMSE with 0.0005, and overlapping lines in figure 9a with slightly close lines in figure 9b, AdaBoost Regression algorithm is performing well on both seen and unseen data.

3.5.8 Experiment of Huber Regression

The model is trained with 13 features and 17568 observations with 15-minute time steps. The model's performance is evaluated with MAPE and RMSE throughout all 10-folds. Mean MAPE, RMSE, Standard Deviation of MAPE, and RMSE were calculated for evaluation. The aim of calculating the mean and standard deviation of the model is to make sure that the model is robust and stable within each fold. Results of Huber Regression are given below:

Fold	**RMSE**	**MAPE**
0	0.0045	2.1008
1	0.0029	3.9094
2	0.0034	2.2361
3	0.003	2.5354
4	0.0029	2.1278
5	0.0036	2.3858
6	0.0029	2.3399
7	0.0032	2.2825
8	0.0026	4.1732
9	0.0032	2.1258
Mean of the Models	0.0032	2.6216
Standard Deviation of the Models	0.0005	0.7233

Table 18: Huber Regression's Performance on both Training and Test Sets



(a) Huber Regression on Training Set.

(b) Huber Regression on Test Set.

Figure 10: Huber Regression's Performance on both Training and Test Sets

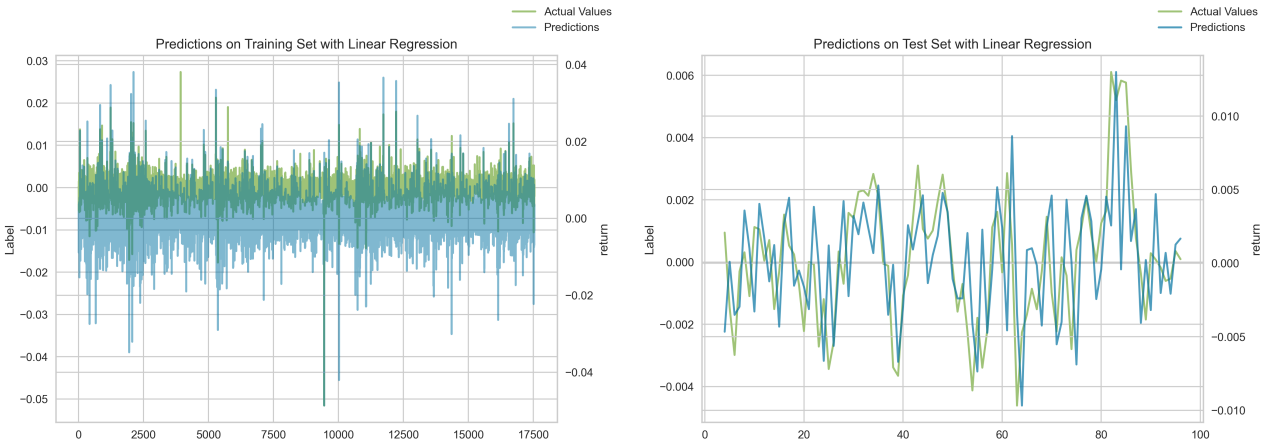
With the Standard Deviation MAPE 0.7233 and RMSE with 0.0005, and overlapping lines in figure 10a with very close lines in figure 10b, the Huber Regression algorithm is performing very well on both seen and unseen data.

3.5.9 Experiment of Linear Regression

The model is trained with 13 features and 17568 observations with 15-minute time steps. The model's performance is evaluated with MAPE and RMSE throughout all 10-folds. Mean MAPE, RMSE, Standard Deviation of MAPE, and RMSE were calculated for evaluation. The aim of calculating the mean and standard deviation of the model is to make sure that the model is robust and stable within each fold. Results of Linear Regression are given below:

Fold	**RMSE**	**MAPE**
0	0.0045	2.5568
1	0.003	4.1591
2	0.0034	3.5073
3	0.003	3.0731
4	0.0051	2.5023
5	0.0036	2.9483
6	0.0029	2.8338
7	0.0031	2.1013
8	0.0026	3.9432
9	0.0031	2.3031
Mean of the Models	0.0034	2.9928
Standard Deviation of the Models	0.0007	0.6529

Table 19: Linear Regression's Performance on both Training and Test Sets



(a) Linear Regression on Training Set.

(b) Linear Regression on Test Set.

Figure 11: Linear Regression's Performance on both Training and Test Sets

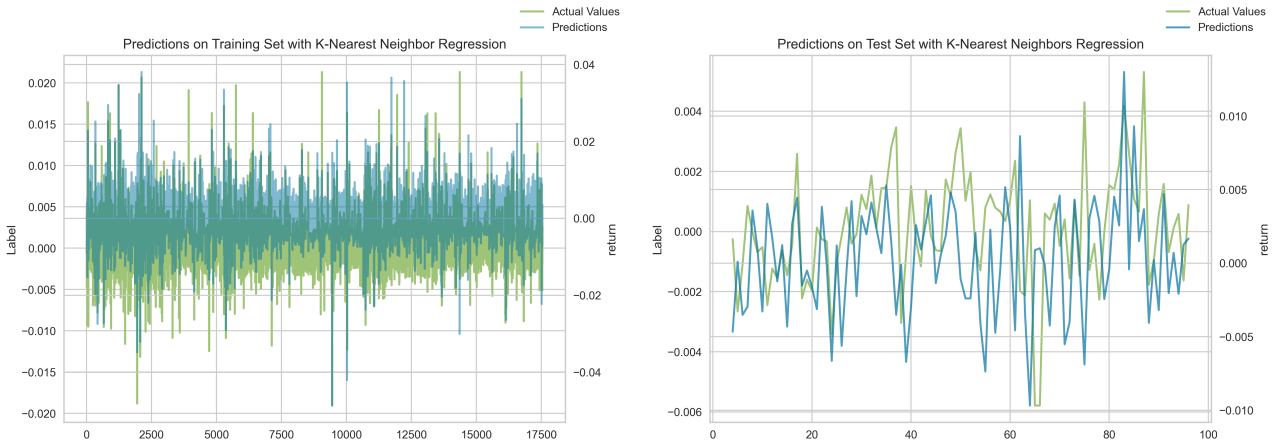
With the Standard Devation MAPE 0.6529 and RMSE with 0.0007, and overlapping lines in figure 11a with very close lines in figure 11b, Linear Regression algorithm is performing very well on both seen and unseen data.

3.5.10 Experiment of K - Nearest Neighbors

The model is trained with 13 features and 17568 observations with 15-minute time steps. The performance of the model is evaluated with MAPE and RMSE throughout all 10-folds. Mean MAPE, RMSE, and Standard Deviation of MAPE, and RMSE were calculated for evaluation. The aim of calculating the mean and standard deviation of the model is making sure that the model is robust and stable within each folds. Results of K - Nearest Neighbors are given below:

Fold	**RMSE**	**MAPE**
0	0.0053	1.9161
1	0.0032	3.3137
2	0.004	3.0678
3	0.0035	2.9494
4	0.0033	2.7757
5	0.004	2.64
6	0.0034	4.4082
7	0.0036	3.0586
8	0.003	3.4244
9	0.0037	2.4048
Mean of the Models	0.0037	2.9958
Standard Deviation of the Models	0.0006	0.6309

Table 20: K-Nearest Neighbor Regression's Performance on both Training and Test Sets



(a) K-Nearest Neighbor Regression on Training Set.

(b) K-Nearest Neighbor Regression on Test Set.

Figure 12: K-Nearest Neighbor Regression's Performance on both Training and Test Sets

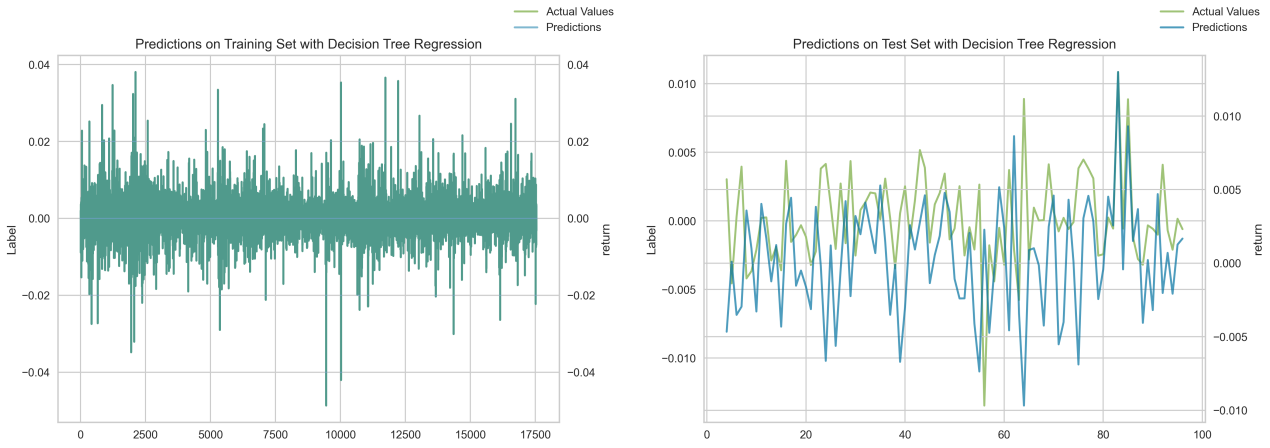
With the Standard Devation MAPE 0.6309 and RMSE with 0.0006, and overlapping lines in figure 12a with slightly close lines in figure 12b, K-Nearest Neighbor Regression algorithm is performing very well on both seen and unseen data.

3.5.11 Experiment of Decision Trees

The model is trained with 13 features and 17568 observations with 15-minute time steps. The performance of the model is evaluated with MAPE and RMSE throughout all 10-folds. Mean MAPE, RMSE, and Standard Deviation of MAPE, and RMSE were calculated for evaluation. The aim of calculating the mean and standard deviation of the model is making sure that the model is robust and stable within each folds. Results of Decision Tree Regressor are given below:

Fold	**RMSE**	**MAPE**
0	0.0055	4.3292
1	0.0035	8.9912
2	0.0045	4.0901
3	0.0036	4.0419
4	0.0032	4.0229
5	0.0046	4.7191
6	0.0036	3.762
7	0.0038	4.0524
8	0.0032	8.9865
9	0.0039	3.3371
Mean of the Models	0.0039	5.0332
Standard Deviation of the Models	0.0006	2.0058

Table 21: Decision Tree Regression's Performance on both Training and Test Sets



(a) Decision Tree Regression on Training Set.

(b) Decision Tree Regression on Test Set.

Figure 13: Decision Tree Regression Performance on both Training and Test Sets

With the Standard Devation MAPE 2.0058 and RMSE with 0.0006, and overlapping lines in figure 13a with slightly close lines in figure 13b, Decision Tree Regression algorithm is performing not well on both seen and unseen data. Having bigger standard deviation means that the model is not very stable.

Thus, Decision Tree will not be used in further steps.

3.6 Evaluation of All Algorithms

To compare all the algorithms, all the results are combined and evaluated with MAPE. Training conditions for all algorithms are:

- Time-Series Cross Validation with 10 folds
- Training data is from July 1, 2021 to December 1, 2021 with 17568 observations with 13 features.
- Test data is from December 14, 2021 to December 15, 2021 with 96 observations.
- All algorithms were evaluated with Line Plots, MAPE and RMSE.

The combined results of each Machine Learning model is given in the table below:

Model	**MEAN MAPE**	**Standard Deviation MAPE**
Lasso Regression	1.0549	0.058
Extra Trees	2.6348	0.5882
LightGBM	2.9729	1.1888
Random Forest	2.7045	0.6221
Extreme Gradient Boosting	2.8422	0.9285
Ridge Regression	2.8786	0.6747
AdaBoost Regression	2.7903	0.7533
Huber Regression	2.6216	0.7233
Linear Regression	2.9928	0.6529
K - Nearest Neighbors	2.9958	0.6309
Decision Trees	5.0332	20058

Table 22: MAPE (with Mean and Standard Deviation) Results of Each Algorithm

Throughout the experimentation step, Lasso Regression and Decision Trees were problematic. These 2 algorithms did not fit the data well as it can be seen in figure 3a on page 45, and 3b on page 45 for Lasso Regression, and it can be seen that Decision Trees are overfitted on training data in figure 13a, thus, they did not fit well on training data on figure 13b. On the other hand, best performing algorithms are:

- Extra Trees
- Ridge Regression
- Linear Regression
- Huber Regression
- AdaBoost Regression

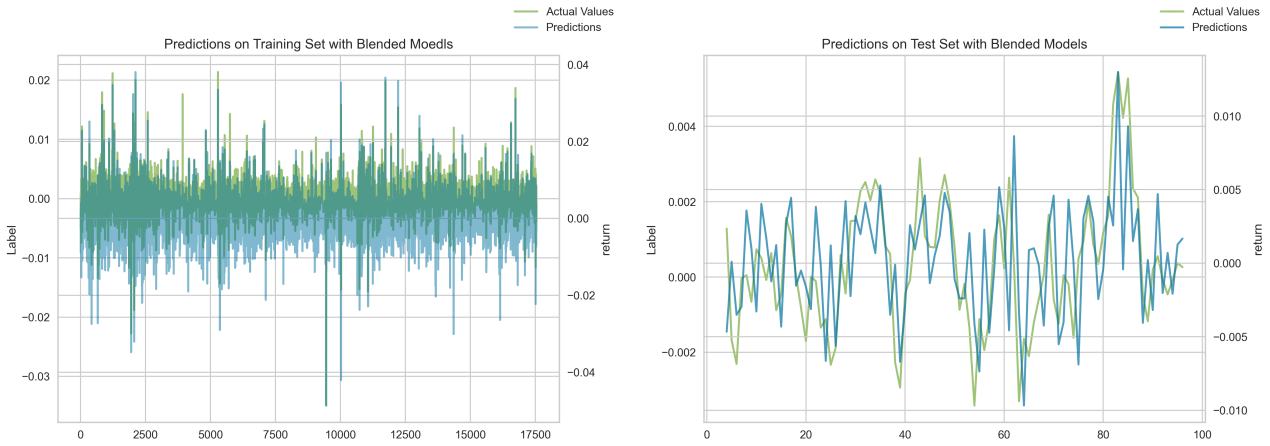
3.7 Blending of Best Models

Using blending technique, these models will be combined and used as the champion model. The model is trained with 13 features and 17568 observations with 15-minute time steps. The performance of the model is evaluated with MAPE and RMSE throughout all 10-folds. Mean MAPE, RMSE, and Standard Deviation of MAPE, and RMSE were calculated for evaluation. The aim of calculating the mean and standard deviation of the model is making sure that the model is robust and stable within each folds. Results of Blended Models are given below:

Fold	**RMSE**	**MAPE**
0	0.0043	2.0794
1	0.0027	3.4056
2	0.0033	2.1452
3	0.0029	2.1757
4	0.0025	1.9657
5	0.0034	2.3064
6	0.0028	2.8384
7	0.003	1.9548
8	0.0025	3.4089
9	0.003	2.0125
Mean of the Models	0.00304	2.42926
Standard Deviation of the Models	0.0005	0.5455

Table 23: MAPE (with Mean and Standard Deviation) Results of Blended Models

The table above shows that blending increased the overall performance of models. The best results of MAPE (with standard deviation and mean) is achieved. The predictions on training and test set are given in the plots below:



(a) Blended Models on Training Set.

(b) Blended Models on Test Set.

Figure 14: Blended Models' Performance on both Training and Test Sets

In figure 14a, it is possible to see that the blended model was successful to catch the trend, but as mentioned before, the predictions on the test results must be similar or close to make sure that model is fair and robust. In figure 14b, the predictions are very close to the actual values. Thus, blending is successful. This blended model will be used in the further steps.

3.8 Experiments for Univariate Time Series Models

All models that were described in chapter 2.3 on 26 are used throughout the experiments. For these models, the training data interval is from July 1, 2021 to December 13, 2021, with 15-minute time steps. All models are trained with 15814 observations. Univariate models are trained with 1 feature, multivariate models are trained with 13 features as in Machine learning models. Models are evaluated with MAPE and RMSE. 10-fold time series cross-validation was used. ADF test was employed to test for stationarity. Since this study forecasts the return rather than the closing price, the data is stationary with $p - value = 0$. To find the best parameters for the ARIMA model's p, d, q, AIC test was employed. The results of the AIC test are given below:

3.8.1 Autocorrelation

To have a better understanding of the data, the experiments started with Autocorrelation plots. An autocorrelation graph is used to determine whether the components of a time series are positively, negatively, or independently associated. The graph is given below:

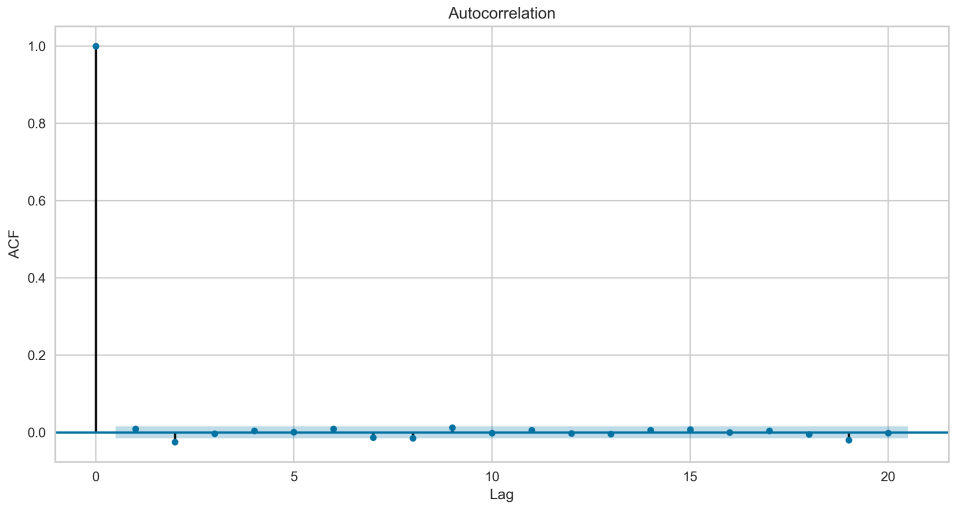


Figure 15: Autocorrelation Plot of Bitcoin Return Prices with 15-minute Time Steps

The autocorrelation plot for daily returns on Apple stock demonstrates that the majority of peaks are statistically insignificant. This means that, as seen above, the returns are not significantly correlated.

Model Parameters	AIC
ARIMA(2,0,2)	-132873.412
ARIMA(0,0,0)	-132869.721
ARIMA(1,0,0)	-132868.941
ARIMA(0,0,1)	-132869.005
ARIMA(0,0,0)	-132870.664
ARIMA(1,0,2)	-132875.173
ARIMA(0,0,2)	-132877.035
ARIMA(0,0,3)	-132875.235
ARIMA(1,0,1)	-132868.270
ARIMA(1,0,3)	-132861.712
ARIMA(0,0,2)	-132877.942
ARIMA(0,0,1)	-132869.967
ARIMA(1,0,2)	-132876.085
ARIMA(0,0,3)	-132876.136
ARIMA(1,0,1)	-132869.276
ARIMA(1,0,3)	-132862.658

Table 24: Parameter Selection for ARIMA models with Akaike Information Criterion Test

The table above shows that the best model is ARIMA(0,0,2) with the AIC value of -132877.035. These parameters are used for ARIMA models in further steps.

3.8.2 Naive Forecaster

Model is trained with 15814 observations from July 1, 2021 to December 13,2021 with 15-minute time steps. The aim of calculating the mean and standard deviation of the model is making sure that the model is robust and stable within each fold. Results of Naive Forecaster are given below:

Fold	RMSE	MAPE
0	0.0041	4.1483
1	0.0034	1.9281
2	0.0111	3.6892
3	0.0055	3.5867
4	0.0122	20.7644
5	0.0048	12.2264
6	0.0042	2.3761
7	0.0079	6.7995
8	0.0098	12.5297
9	0.0057	4.7343
Mean of the Models	0.0068	7.2783
Standard Deviation of the Models	0.0030	5.7411

Table 25: RMSE and MAPE scores of Naive Forecaster for Each Fold

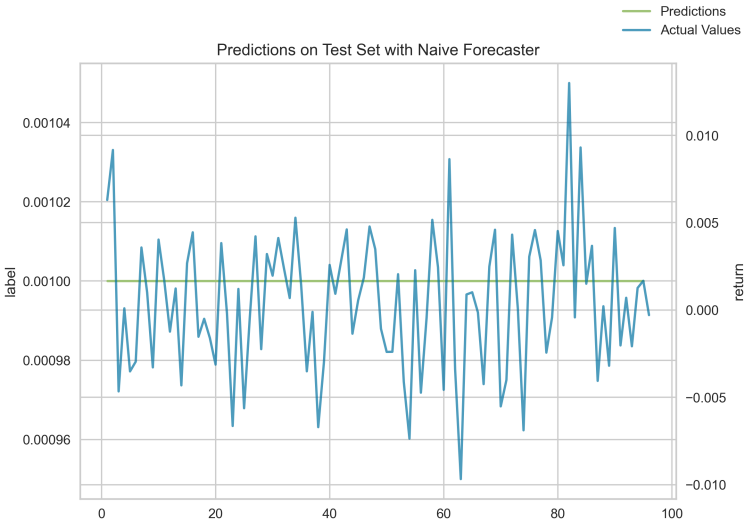


Figure 16: Naive Forecaster Predictions on Test Data

Naive Forecaster predicted a constant value for each observation. Thus, this model is not used in further steps.

3.8.3 Seasonal Naive Forecaster

Model is trained with 15814 observations from July 1, 2021 to December 13,2021 with 15-minute time steps. The aim of calculating the mean and standard deviation of the model is making sure that the model is robust and stable within each fold. Results of Seasonal Naive Forecaster are given below:

Fold	RMSE	MAPE
0	0.005	4.7872
1	0.0048	7.8029
2	0.0104	2.4609
3	0.0067	5.1401
4	0.0065	7.4647
5	0.006	4.7211
6	0.0051	5.1075
7	0.006	3.433
8	0.0059	3.8407
9	0.0073	6.1684
Mean of the Models	0.0063	5.0927
Standard Deviation of the Models	0.0015	1.6003

Table 26: RMSE and MAPE scores of Seasonal Naive Forecaster for Each Fold

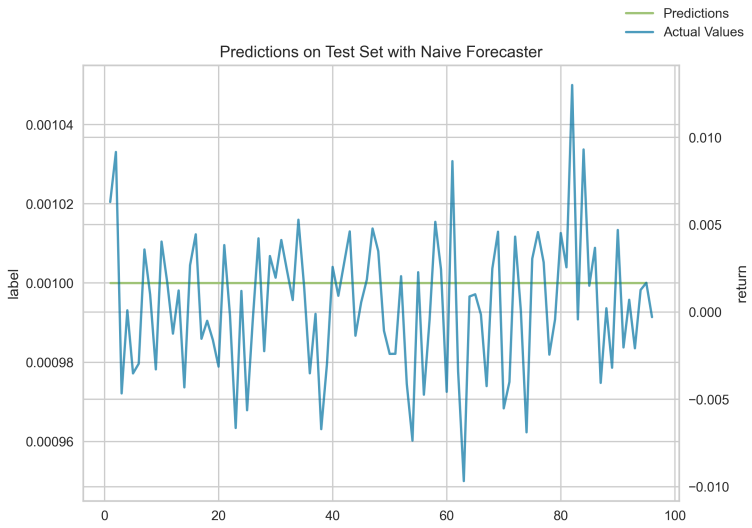


Figure 17: Seasonal Naive Forecaster Predictions on Test Data

Forecasting results of Seasonal Naive Forecaster looks promising. The lines for predicted values are close to the actual values' lines. Thus, Seasonal Naive Forecaster is used in the further steps.

3.8.4 ETS

Model is trained with 15814 observations from July 1, 2021 to December 13, 2021 with 15-minute time steps. The aim of calculating the model's mean and standard deviation is to make sure that the model is robust and stable within each fold. Results of ETS are given below:

Fold	RMSE	MAPE
0	0.0036	1.0019
1	0.0032	1.0016
2	0.0102	0.00009994
3	0.0047	1.0113
4	0.0045	1.3758
5	0.0027	1.0117
6	0.0039	1.0078
7	0.0037	1.0008
8	0.0045	0.00009818
9	0.0043	0.00009881
Mean of the Models	0.0045	1.038
Standard Deviation of the Models	0.0019	0.00001

Table 27: RMSE and MAPE scores of Seasonal Naive Forecaster for Each Fold

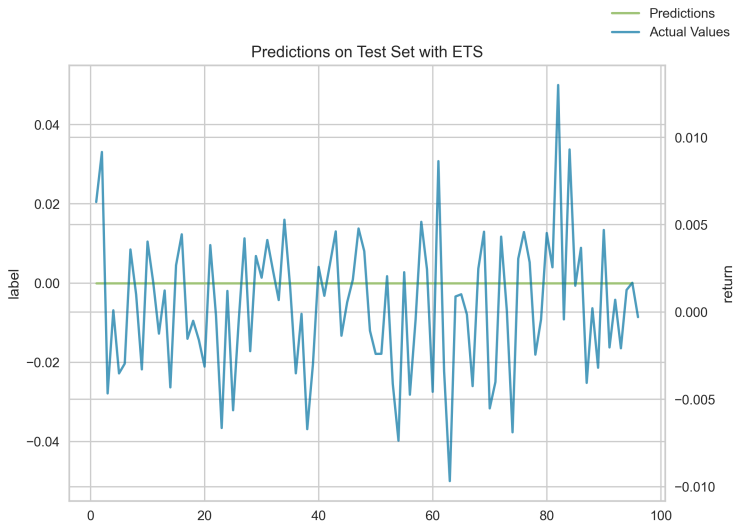


Figure 18: ETS Predictions on Test Data

Even though ETS has low Mean and Standard Deviation MAPE, the graphic above shows that it predicted a constant value for each observation. Thus, it will not be used in further experiments.

3.8.5 ARIMA

Model is trained with 15814 observations from July 1, 2021 to December 13,2021 with 15-minute time steps. The aim of calculating the model's mean and standard deviation is to make sure that the model is robust and stable within each fold. Results of ARIMA are given below:

Fold	RMSE	MAPE
0	0.0036	1.0292
1	0.0032	1.0442
2	0.0102	1.0009
3	0.0047	1.0059
4	0.0045	0.00009849
5	0.0027	1.0603
6	0.0039	1.0012
7	0.0037	0.00009994
8	0.0045	1.0116
9	0.0043	1.0111
Mean of the Models	0.00453	1.0149
Standard Deviation of the Models	0.0019	0.00002

Table 28: RMSE and MAPE scores of ARIMA for Each Fold

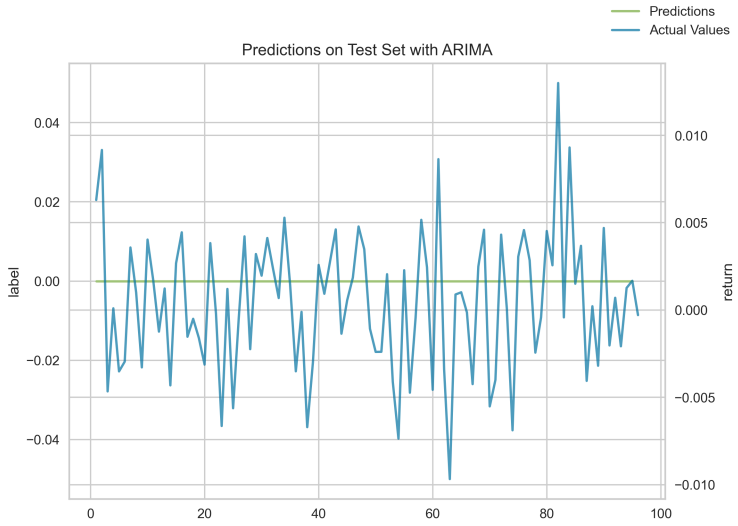


Figure 19: ETS Predictions on Test Data

Even though ARIMA has low Mean and Standard Deviation MAPE, the graphic above shows that it predicted a constant value for each observation. Thus, it will not be used in further experiments.

3.8.6 Exponential Smoothing

Model is trained with 15814 observations from July 1, 2021 to December 13,2021 with 15-minute time steps. The aim of calculating the model's mean and standard deviation is to make sure that the model is robust and stable within each fold. Results of Exponential Smoothing is given below:

Fold	RMSE	MAPE
0	0.0036	0.00009843
1	0.0032	1.007
2	0.0102	0.00009991
3	0.0047	1.0113
4	0.0045	1.0201
5	0.0027	1.1706
6	0.0039	1.0097
7	0.0037	1.002
8	0.0045	1.0427
9	0.0044	1.0363
Mean of the Models	0.00454	1.0283
Standard Deviation of the Models	0.0019	0.0005

Table 29: RMSE and MAPE scores of Exponential Smoothing for Each Fold

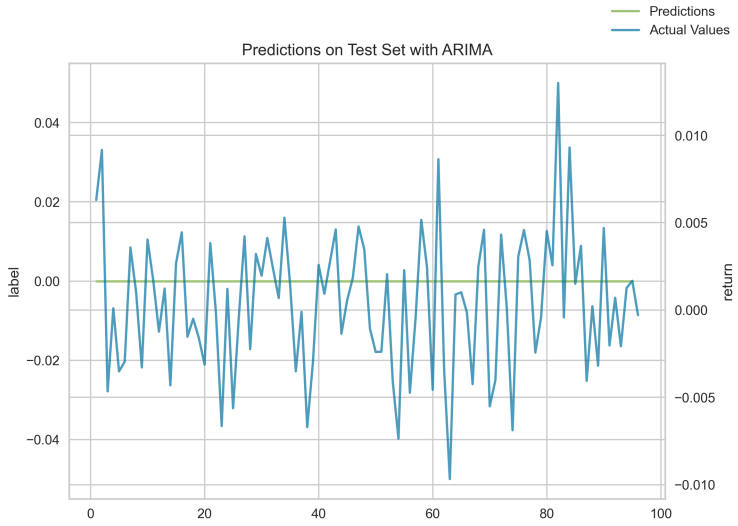


Figure 20: ETS Predictions on Test Data

Even though ETS has low Mean and Standard Deviation MAPE, the graphic above shows that it predicted a constant value for each observation. Thus, it will not be used in further experiments.

3.9 Experiments for Multivariate Time Series Models

All models that were described in chapter 2.3 on 26 are used throughout the experiments. For these models, the training data interval is from July 1, 2021 to December 13, 2021 with 15-minute time steps. All models are trained with 15814 observations. Multivariate models are trained with 1 feature, multivariate models are trained with 15 features. Models are evaluated with MAPE and RMSE. 10-

fold time series cross-validation was used. ADF test was employed to test for stationarity. During the experiments of Univariate models, only one ADF test was conducted. Since there are 15 numeric features for Multivariate models, 15 tests are run. The results of the ADF tests are given below.

Test Name	**Column Name**	**P Value**	**Stationarity**
ADF	Open	0.498519	Non-Stationary
ADF	High	0.511696	Non-Stationary
ADF	Low	0.442089	Non-Stationary
ADF	Close	0.497759	Non-Stationary
ADF	Volume	2.56E-21	Stationary
ADF	Number of Trades	2.33E-20	Stationary
ADF	Rolling Mean Hourly	0.468676	Non-Stationary
ADF	Rolling Median Hourly	0.470616	Non-Stationary
ADF	Rolling Max Hourly	0.484866	Non-Stationary
ADF	Rolling Std Hourly	0	Stationary
ADF	Hourly Range Min - Max	4.09E-22	Stationary
ADF	Number of Trades Hourly Range	3.61E-22	Stationary
ADF	Number of Trades Percental Change	0	Stationary
ADF	Percental Change of Volume	0	Stationary
ADF	Weighted Average	0.485478	Non-Stationary

Table 30: ADF Test for Exogenous Variables

The table above shows that any calculations that include percentages remove stationarity. The features that are not stationary will be transformed to stationary using differencing. The table below shows the results of the ADF test after first-order differencing.

Test Name	Column Name	P Value	Stationarity
ADF	Open	0	Stationary
ADF	High	0	Stationary
ADF	Low	0	Stationary
ADF	Close	0	Stationary
ADF	Volume	0	Stationary
ADF	Number of Trades	0	Stationary
ADF	Rolling Mean Hourly	4.15E-24	Stationary
ADF	Rolling Median Hourly	0	Stationary
ADF	Rolling Max Hourly	0	Stationary
ADF	Rolling Std Hourly	0	Stationary
ADF	Hourly Range Min - Max	0	Stationary
ADF	Number of Trades Hourly Range	0	Stationary
ADF	Number of Trades Percental Change	0	Stationary
ADF	Percental Change of Volume	0	Stationary
ADF	Weighted Average	0	Stationary

Table 31: ADF Test for Exogenous Variables after First-Order Differencing

After first-order differencing, every exogenous variable is stationary.

3.9.1 Vector Autoregression

All models that were described in chapter 2.3 on 26 are used throughout the experiments. For these models, the training data interval is from July 1, 2021 to December 13, 2021 with 15-minute time steps. All models are trained with 15814 observations. Multivariate models are trained with 1 feature, multivariate models are trained with 15 features. Models are evaluated with MAPE and RMSE. 10-fold time series cross-validation was used. ADF test was employed to test for stationarity. During the experiments of Univariate models, only one ADF test was conducted. Since there are 15 numeric features for Multivariate models, 15 tests are conducted, and non-stationary columns are transformed to stationary. After finishing the preprocessing, to decide the p of VAR, the AIC test is applied. The test results are given in the table below:

Test Name	Order	Value
AIC	1	109.662
AIC	2	99.963
AIC	3	70.524
AIC	4	71.864

Table 32: AIC for each order

In the table, the highest lag order 1 has the highest AIC. Thus, p is 1. With the p value of 1, the results of VAR is given below:

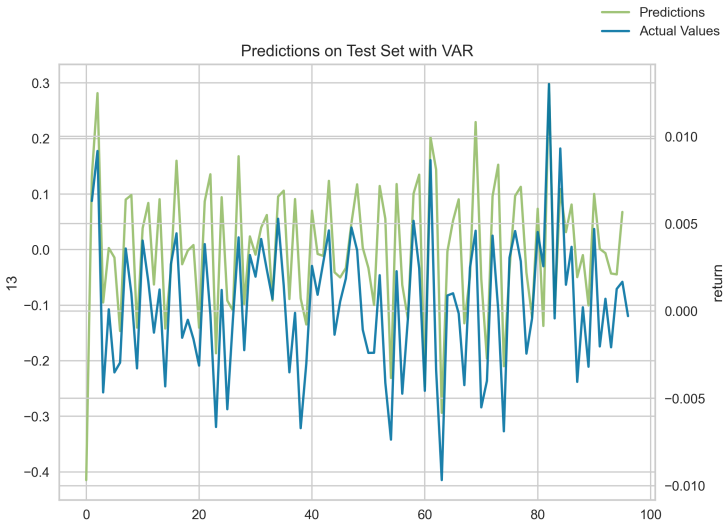


Figure 21: VAR Predictions on Test Data

Forecasting results of VAR looks promising. The lines for predicted values are close to the actual values' lines. Thus, VAR is used in the further steps.

3.9.2 Vector Autoregression Moving Average

All models that were described in chapter 2.3 on 26 are used throughout the experiments. For these models, the training data interval is from July 1, 2021 to December 13, 2021 with 15-minute time steps. All models are trained with 15814 observations. Multivariate models are trained with 1 feature, multivariate models are trained with 15 features. Models are evaluated with MAPE and RMSE. 10-fold time series cross-validation was used. ADF test was employed to test for stationarity. During the experiments of Univariate models, only one ADF test was conducted. Since there are 15 numeric features for Multivariate models, 15 tests are conducted, and non-stationary columns are transformed to stationary. The result of VARMA is given below:

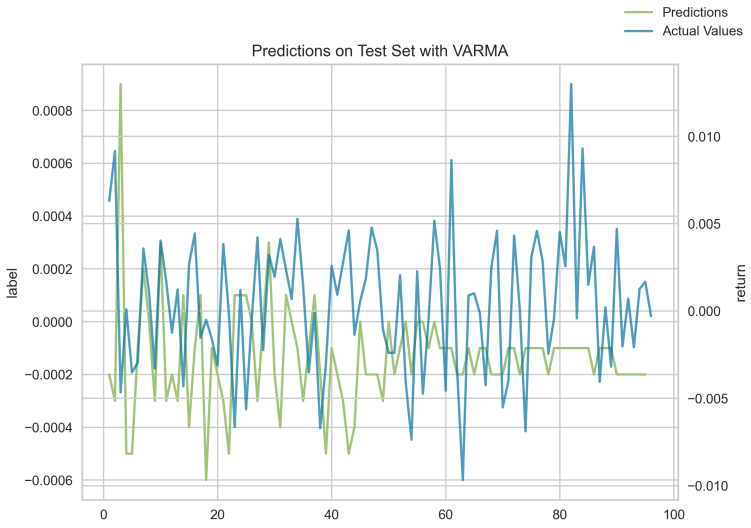


Figure 22: VARMA Predictions on Test Data

Forecasting results of VARMA is not looking very good. Thus, VARMA is used in the further steps.

3.10 Blending of Best Time Series Models

After the experiments, the superior Time Series models are Seasonal Naive Forecaster and VAR. Thus, these models will be blended. The results are given in the plot below:

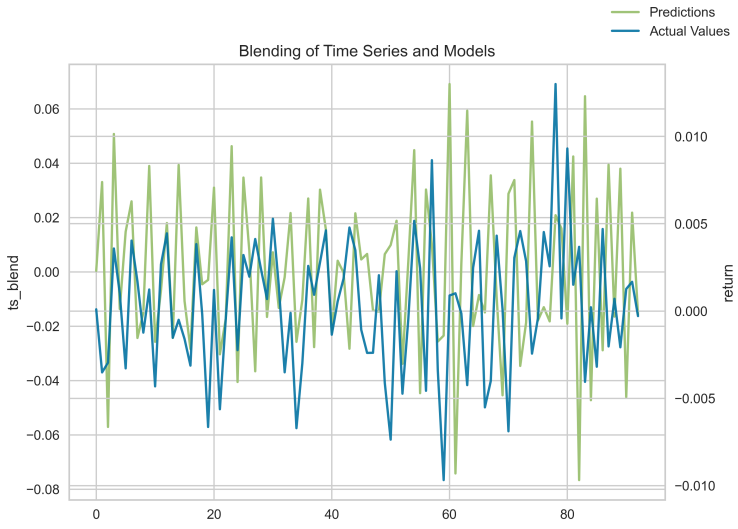


Figure 23: Blending of Time Series Models

The blending of Time Series models looks better than single Time Series models. But, the performance of Machine learning models looked better. In the final steps, both Time Series and Machine

learning models will be blended and evaluated.

3.11 Blending of All Models

In this chapter, both Machine learning and Time Series models will be blended. On the other hand, their MAPE will be evaluated to see if any of those models perform better. MAPE results of each type of model are given in the table below:

Models	**MAPE**
Machine learning - Blended	2.429
Time Series - Blended	9.285
Time Series and Machine learning Blended	5.729

Table 33: Results of Blended Models

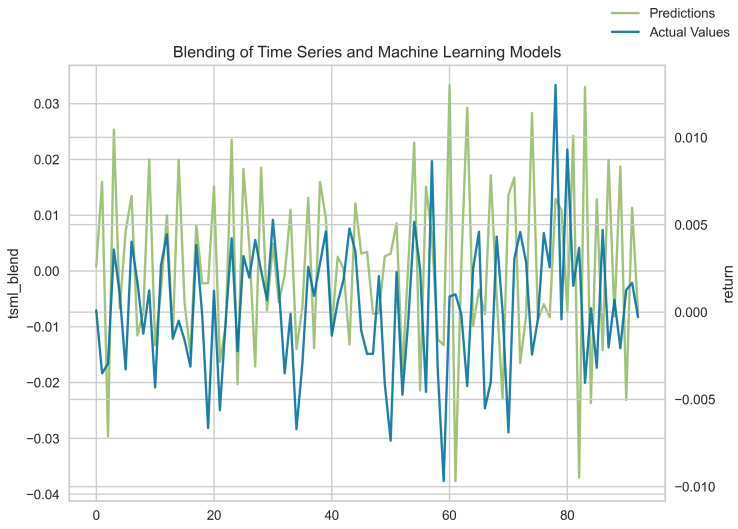


Figure 24: Blending Results of Time Series and Machine learning Models

Results show that the Blended Machine learning model was the superior model. The list of blended models is as follows:

- Extra Trees
- Ridge Regression
- Linear Regression
- Huber Regression
- Adaboost Regression

4 Conclusions

The purpose of this research was to evaluate various models and to guarantee that each model performs to its full capacity and then to blend these models to produce even more excellent outcomes. To maximize the potential of machine learning models, feature engineering, permutation feature importance, and cross-validation were used; methods such as difference taking, AIC test, ADF test, and extra features were used to maximize the potential of time series models were employed. Detailed information about feature engineering is in chapter 3.3 on page 33 Additionally, experiments for the ideal training data were undertaken to forecast the time between December 14 and December 15 to enhance the model performance. June 1, 2021 to December 1, 2021 is the optimal training data range. The results for each date interval can be found on chapter **??** on page **??**. This time period was used to train the models used in the research. Time-series cross-validation using mean and standard deviation of MAPE and root mean square error (RMSE) were utilized to assess model performance and choose the optimal training data range for this study. Hyperparameter tuning was not employed because the default values produced better results than tuned models. Following the evaluation of both models, the following models were found to be the most accurate:

- Best performing machine learning models:
  - Extra Trees Regression
  - Ridge Regression
  - Linear Regression
  - Huber Regression
  - Adaboost
- Best performing time-series models:
  - Seasonal Naive Forecaster
  - Vector Autoregression

Machine learning models and time-series models were evaluated independently, then blending was applied to each model once it had been evaluated individually. The performance of blended models outperformed separate ones, while the performance of the time-series model was unsatisfactory. Most time-series models predicted a constant value and required a significant amount of time to train. On the other hand, blending time-series models improved their performance, but it was still insufficient compared to machine learning models in terms of accuracy. The champion model, which was created by combining five machine learning algorithms, had a MAPE of 2.42% and was selected as the final model.

Future work

The flaws and limits methodologies established throughout the research study have led to the identification of the following topics that should be investigated further in the future.

- Time restrictions prevented us from using approaches and other indicators that may have been utilized to improve performance in this research because of the large number of machine learning experiments that were completed.
- Due to the fact that a lot of time was spent on the learning data for the time series models, not enough effort was spent on the parameters of the time series models.
- Since cryptocurrency market is highly volatile and manipulative through social media and other platforms, text data with support of sentiment analysis can be used for machine learning models.
- To reduce the time of training for time-series models, GPU support or parallel processing can be employed.
- Training data set experimentations were conducted for only machine learning models due to time restrictions. Best performing date interval for time-series models can be experimented.
- Rather than time-series cross-validation, other cross-validation techniques can be used such as Sliding Window Splitter, Expanding Window Splitter.
- Due to time restrictions, and training time; VARMAX, VARIMAX, SARIMAX could not be implemented.
- In this study, time-steps are 15 minutes; for further research, other time-steps can be experimented.
- Bitcoin is the most popular cryptocurrency and number of transactions are really high. Another cryptocurrency can be selected to have better results.
- While evaluating the models, monthly RMSE or MAPE might be calculated. During the calculation, weighted average can be used to make sure that models have less errors as gets closer to the unseen data.

References

[1] Amin Azari. *Bitcoin Price Prediction: An ARIMA Approach*, CoRR, 2019, 1904.05315, https://arxiv.org/pdf/1904.05315.pdf

[2] Yike Xu *Bitcoin Price Forecast Using LSTM and GRU Recurrent networks, and Hidden Markov Model*, UCLA, 2020, https://escholarship.org/uc/item/70d9n5sd

[3] Yuankai Guo, Yangyang Li, Yuan Xu. *Study on the application of LSTM-LightGBM Model in stock rise and fall prediction*, International Conference on Computer Science Communication and Network Security (CSCNS2020), 2021, 2, 6, https://doi.org/10.1051/matecconf/202133605011

[4] Esam Mahdi, Víctor Leiva, Saed Mara'Beh, Carlos Martin-Barreiro. *A New Approach to Predicting Cryptocurrency Returns Based on the Gold Prices with Support Vector Machines during the COVID-19 Pandemic Using Sensor-Related Data*, Sensors, 2021, 21(18), https://doi.org/10.3390/s21186319

[5] Ziaul Haque Munim, Mohammad Hassan Shakil, Ilan Alon. *Next-Day Bitcoin Price Forecast*, Journal of Risk and Financial Management, 2019, 12(2), https://doi.org/10.3390/jrfm12020103

[6] Kaby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, Yoram Singer. *Online Passive-Aggressive Algorithms* Journal of Machine learning Research, 2006, 7(19), http://jmlr.org/papers/v7/crammer06a.html

[7] Harris Drucker, Chris J.C. Burges, Linda Kaufman, Alex Smola, Vladimir Vapnik. *Support Vector Rector Machines*, 1965

[8] J.R. Quinlan. *Induction of Decision Trees*, Kluwer Academic Publishers, 1986

[9] Leo Breiman. *Random Forests*, Machine Learning, 2001, 45, pp. 5-32, http://dx.doi.org/10.1023/A3A1010933404324

[10] Pierre Geurts, Damien Ernst, Louis Wehenke. *Extremely Randomized Trees*, Machine Learning, 2006, 63(1), pp. 3-42, http://dblp.uni-trier.de/db/journals/ml/ml63.html/GeurtsEW06

[11] Dimitri P Solomatine, Durga Shrestha. *AdaBoost.RT: A boosting algorithm for regression problems*, IEEE International Joint Conference on Neural Networks, 2004

[12] Jerome H. Friedman. *Greedy Function Approximation: A Gradient Boosting Machine*, Annals of Statistics, 2001, 23(5), pp. 1189 - 1232

[13] Tianqi Chen, Carlos Guestrin. *XGBoost: A Scalable Tree Boosting System*, 2016, DOI: 10.1145/2939672.2939785

[14] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, Tie-Yan Liu. *LightGBM: A Highly Efficient Gradient Boosting Decision Tree*, Advances in Neural Information Processing Systems 30, 2017, pp. 3146-3154, http://papers.nips.cc/paper/6907-lightgbm-a-highly-efficient-gradient-boosting-decision-tree.pdf

[15] Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. & Gulin, A. CatBoost: unbiased boosting with categorical features.. *NeurIPS*. pp. 6639-6649 (2018), http://dblp.uni-trier.de/db/conf/nips/nips2018.html/ProkhorenkovaGV18

[16] Bergstra, J. & Bengio, Y. Random Search for Hyper-Parameter Optimization. *Journal Of Machine Learning Research*. **13** pp. 281-305 (2012)

[17] Larson S. *The shrinkage of the coefficient of multiple correlation*, 1931

[18] Assimakopoulos, V. & Nikolopoulos, K. The theta model: a decomposition approach to forecasting. *International Journal Of Forecasting*. **16**, 521-530 (2000), http://www.sciencedirect.com/science/article/B6V92-41J6944-9/1/2e47ffd682aba83ec23c3243e3359d09

[19] Taylor, S. & Letham, B. Forecasting at Scale.. *PeerJ PrePrints*. **5** pp. e3190 (2017), http://dblp.uni-trier.de/db/journals/peerjpre/peerjpre5.html

[20] João Almeida, Shravan Tata, Andreas Moser, Vikko Smit. *Bitcoin prediction using ANN, IN4015: Neural Networks*, 2015

[21] Amjad, M. & Shah, D. Trading Bitcoin and Online Time Series Prediction.. *NIPS Time Series Workshop*. **55** pp. 1-15 (2016), http://dblp.uni-trier.de/db/conf/nips/tsw2016.html

[22] McNally, S., Roche, J. & Caton, S. Predicting the Price of Bitcoin Using Machine Learning.. *PDP*. pp. 339-343 (2018), http://dblp.uni-trier.de/db/conf/pdp/pdp2018.html

[23] Georgoula, I., Pournarakis, D., Bilanakos, C., Sotiropoulos, D. & Giaglis, G. Using Time-Series and Sentiment Analysis to Detect the Determinants of Bitcoin Prices.. *MCIS*. pp. 20 (2015), http://dblp.uni-trier.de/db/conf/mcis/mcis2015.html

[24] Madan, I. Automated Bitcoin Trading via Machine Learning Algorithms. (2014)

[25] Ji, S., Kim, J. & Im, H. A Comparative Study of Bitcoin Price Prediction Using Deep Learning. *Mathematics*. **7** (2019), https://www.mdpi.com/2227-7390/7/10/898

[26] Jang, H. & Lee, J. An Empirical Study on Modeling and Prediction of Bitcoin Prices With Bayesian Neural Networks Based on Blockchain Information.. *IEEE Access*. **6** pp. 5427-5437 (2018), http://dblp.uni-trier.de/db/journals/access/access6.html

[27] Lahmiri, S. & Bekiros, S. Cryptocurrency forecasting with deep learning chaotic neural networks. *Chaos, Solitons and Fractals*. **118** pp. 35-40 (2019), https://www.sciencedirect.com/science/article/pii/S0960077918310233

[28] Simon, Jeyasheela Rakkini & Geetha, K Block Mining reward prediction with Polynomial Regression, Long short-term memory, and Prophet API for Ethereum blockchain miners. *ITM Web Conf.*. **37** pp. 01004 (2021), https://doi.org/10.1051/itmconf/20213701004

[29] T. Phaladisailoed and T. Numnonda, "Machine Learning Models Comparison for Bitcoin Price Prediction," 2018 10th International Conference on Information Technology and Electrical Engineering (ICITEE), 2018, pp. 506-511, doi: 10.1109/ICITEED.2018.8534911.

[30] Jaquart, P., Dann, D. & Weinhardt, C. Short-term bitcoin market prediction via machine learning. *The Journal Of Finance And Data Science*. **7** pp. 45-66 (2021), https://www.sciencedirect.com/science/article/pii/S2405918821000027

[31] Sun, X., Liu, M. & Sima, Z. A novel cryptocurrency price trend forecasting model based on LightGBM. *Finance Research Letters*. **32** pp. 101084 (2020), https://www.sciencedirect.com/science/article/pii/S1544612318307918

[32] Hattori, T. A forecast comparison of volatility models using realized volatility: evidence from the Bitcoin market. *Applied Economics Letters*. **27**, 591-595 (2020), https://doi.org/10.1080/13504851.2019.1644421

[33] Mallqui, D. & Fernandes, R. Predicting the direction, maximum, minimum and closing prices of daily Bitcoin exchange rate using machine learning techniques.. *Appl. Soft Comput.*. **75** pp. 596-606 (2019), http://dblp.uni-trier.de/db/journals/asc/asc75.html

[34] Aalborg, H., Molnár, P. & De Vries, J. What can explain the price, volatility and trading volume of Bitcoin?. *Finance Research Letters*. **29** pp. 255-265 (2019), https://www.sciencedirect.com/science/article/pii/S1544612318302058

[35] Robert Adcock, Nikola Gradojevic. Non-fundamental, non-parametric Bitcoin forecasting. Physica A, Elsevier, 2019, 531, pp.121727 -. ff10.1016/j.physa.2019.121727f

[36] Kristjanpoller, W. & Minutolo, M. A hybrid volatility forecasting framework integrating GARCH, artificial neural network, technical analysis and principal components analysis.. *Expert Syst. Appl.*. **109** pp. 1-11 (2018), http://dblp.uni-trier.de/db/journals/eswa/eswa109.html

[37] Walther, T., Klein, T. & Bouri, E. Exogenous drivers of Bitcoin and Cryptocurrency volatility – A mixed data sampling approach to forecasting. *Journal Of International Financial Markets, Institutions And Money*. **63** pp. 101133 (2019), https://www.sciencedirect.com/science/article/pii/S1042443119302446

[38] Devlin, J., Chang, M., Lee, K. & Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR*. **abs/1810.04805** (2018), http://arxiv.org/abs/1810.04805

[39] Roy, A. & Ojha, M. Twitter sentiment analysis using deep learning models. *2020 IEEE 17th India Council International Conference (INDICON)*. pp. 1-6 (2020)

[40] Mallqui, D. & Fernandes, R. Predicting the direction, maximum, minimum and closing prices of daily Bitcoin exchange rate using machine learning techniques.. *Appl. Soft Comput.*. **75** pp. 596-606 (2019), http://dblp.uni-trier.de/db/journals/asc/asc75

[41] Briere, Marie and Oosterlinck, Kim and Szafarz, Ariane, Virtual Currency, Tangible Return: Portfolio Diversification with Bitcoin (2015). Journal of Asset Management, 16, 6, 365-373, doi:10.1057/jam.2015.5 , Available at SSRN: https://ssrn.com/abstract=2324780 or http://dx.doi.org/10.2139/ssrn.2324780

[42] Catania, L., Grassi, S. & Ravazzolo, F. Forecasting Cryptocurrencies Financial Time Series. (Centre for Applied Macro-,2018,3), https://ideas.repec.org/p/bny/wpaper/0063.html

[43] Kodama, O., Pichl, L. & Kaizoji, T. REGIME CHANGE AND TREND PREDICTION FOR BITCOIN TIME SERIES DATA. *CBU International Conference Proceedings*. **5** pp. 384 (2017,9)

[44] Ebru Şeyma Karakoyun. *Real-Time Prediction of Time-Series with Deep Learning*, 2018

[45] Siami-Namini, S. & Namin, A. Forecasting Economics and Financial Time Series: ARIMA vs. LSTM. *CoRR*. **abs/1803.06386** (2018), http://arxiv.org/abs/1803.06386

[46] Briere, Marie and Oosterlinck, Kim and Szafarz, Ariane, Virtual Currency, Tangible Return: Portfolio Diversification with Bitcoin (2015). Journal of Asset Management, 16, 6, 365-373, doi:10.1057/jam.2015.5 , Available at SSRN: https://ssrn.com/abstract=2324780 or http://dx.doi.org/10.2139/ssrn.2324780

[47] Grachev, O. Application of Time Series Models (ARIMA, GARCH, and ARMA-GARCH) for Stock Market Forecasting. (2017)

[48] Montaño Moreno JJ, Palmer Pol A, Muñoz Gracia P. Artificial neural networks applied to forecasting time series. Psicothema. 2011 Apr;23(2):322-9. PMID: 21504688.

[49] Baasher, A. & Fakhr, M. FOREX Daily Trend Prediction using Machine Learning Techniques. (2011,11)

[50] Biau, G., Scornet, E. A random forest guided tour. TEST 25, 197–227 (2016). https://doi.org/10.1007/s11749-016-0481-7

[51] Geurts, P., Ernst, D. & Wehenkel, L. Extremely randomized trees.. *Mach. Learn.*. **63**, 3-42 (2006), http://dblp.uni-trier.de/db/journals/ml/ml63.html

[52] M. Tranmer and M. Elliot, Multiple Linear Regression, The Cathie Marsh Centre for Census and Survey Research (CCSR) 2008.

[53] Hoerl, Arthur E., and Robert W. Kennard. "Ridge Regression: Biased Estimation for Nonorthogonal Problems." Technometrics, vol. 42, no. 1, [Taylor & Francis, Ltd., American Statistical Association, American Society for Quality], 2000, pp. 80–86, https://doi.org/10.2307/1271436.

[54] Tibshirani, R. Regression shrinkage and selection via the Lasso. *Journal Of The Royal Statistical Society. Series B (Methodological)*. pp. 267-288 (1996)

[55] Zhou, D. & Jetter, K. Approximation with polynomial kernels and SVM classifiers.. *Adv. Comput. Math.*. **25**, 323-344 (2007,2,7), http://dblp.uni-trier.de/db/journals/adcm/adcm25.html

[56] Rumelhart, D., Hinton, G. & Williams, R. Learning representations by back-propagating errors. Nature 323, 533–536 (1986). https://doi.org/10.1038/323533a0

[57] Cunningham, P. & Delany, S. k-Nearest Neighbour Classifiers: 2nd Edition (with Python examples). *CoRR.* **abs/2004.04523** (2020), https://arxiv.org/abs/2004.04523

[58] Everette S. Gardner, Jr. *Exponential Smoothing: The State of the Art*, Journal of Forecasting, 1985, 4(1), pp. 1-18, https://doi.org/10.1002/for.3980040103

[59] Adhikari, R. & Agrawal, R. An Introductory Study on Time Series Modeling and Forecasting. *CoRR.* **abs/1302.6613** (2013), http://dblp.uni-trier.de/db/journals/corr/corr1302.html

[60] Stock, J. & Watson, M. Vector Autoregressions. *Journal Of Economic Perspectives.* **15**, 101-115 (2001,12), https://www.aeaweb.org/articles?id=10.1257/jep.15.4.101

[61] Canova, Fabio and Ciccarelli, Matteo, Panel Vector Autoregressive Models: A Survey (January 16, 2013). ECB Working Paper No. 1507, Available at SSRN: https://ssrn.com/abstract=2201610 or http://dx.doi.org/10.2139/ssrn.2201610

[62] Li, D., Ling, S. & Tong, H. On moving-average models with feedback. *Bernoulli.* **18** (2012,5)

[63] Burlando, P., Rosso, R., Cadavid, L. & Salas, J. Forecasting of short-term rainfall using ARMA models. *Journal Of Hydrology.* **144**, 193-211 (1993), https://www.sciencedirect.com/science/article/pii/0022169493901726

[64] Abrigo, M. & Love, I. Estimation of Panel Vector Autoregression in Stata. *The Stata Journal.* **16**, 778-804 (2016), doi:10.1177/1536867X1601600314

[65] Isufi, E., Loukas, A., Perraudin, N. & Leus, G. Forecasting Time Series With VARMA Recursions on Graphs.. *IEEE Trans. Signal Process..* **67**, 4870-4885 (2019), http://dblp.uni-trier.de/db/journals/tsp/tsp67.html

[66] Östermark, R. & Saxén, H. VARMAX-modelling of blast furnace process variables. *European Journal Of Operational Research.* **90**, 85-101 (1996), https://www.sciencedirect.com/science/article/pii/0377221794003041

[67] Kwiatkowski, D., Phillips, P., Schmidt, P. & Shin, Y. Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root?. *Journal Of Econometrics.* **54**, 159-178 (1992), https://www.sciencedirect.com/science/article/pii/030440769290104Y

[68] Montgomery, D.C., Jennings, C.L. and Kulachi, M. *Introduction to Time Series Analysis and Forecasting*, The Journal of Infectious Diseases, 2008

[69] Hyndman, R. & Athanasopoulos, G. Forecasting: Principles and Practice. (OTexts,2018)

[70] Newbold, P. The Principles of the Box-Jenkins Approach. *Operational Research Quarterly (1970-1977).* **26**, 397-412 (1975), http://www.jstor.org/stable/3007750

[71] Mahesh, B. Machine Learning Algorithms -A Review. (2019,1)

[72] Liaw, A. & Wiener, M. Classification and Regression by RandomForest. *Forest.* **23** (2001,11)

[73] Ghahramani Z. (2004) Unsupervised Learning. In: Bousquet O., von Luxburg U., Rätsch G. (eds) Advanced Lectures on Machine Learning. ML 2003. Lecture Notes in Computer Science, vol 3176. Springer, Berlin, Heidelberg.

[74] Jong, P. A Cross-Validation Filter for Time Series Models. *Biometrika.* **75**, 594-600 (1988), http://www.jstor.org/stable/2336613