NORTHEASTERN UNIVERSITY  
COLLEGE OF PROFESSIONAL STUDIES



**EAI 6010**

**Trend Prediction Using Binance API**

Spring 2021 Term

Group #6:

Benjamin Cooper

Mayank Kumar Pandey

Instructor: Sergiy Shevchenko

May 13, 2021

**INTRODUCTION**

Our dataset is retrieved using Binance API. It is open for any public usage/research while the source of the data is Binance ADA server. This dataset was collected as a part of machine learning project to determines what features best predict Trend for ADA bitcoin. It was last collected from on Jan 1, 2020 till May 13,2021. The dataset contains the ADA bitcoin K line historical data to predict trends correlated with ADA, which is saved and analyzed in Python’s Panda Dataframe directly. Altogether there are 7 columns and 266 rows.

We converted retrieved historical data and converted this into pandas dataframe using python libraries.

|  |  |
| --- | --- |
| **Variable** | **Description** |
| Time | Datetime64 |
| Open | object |
| High | object |
| Low | object |
| Close | object |
| Single\_Day\_Trend | int64 |
| Two\_Day\_Trend | Int64 |

Figure 2.1 Data Types

However, We have imported historical data using binance API which is in the format as follows:

1499040000000, # Open time  
0.01634790, # Open  
0.80000000, # High  
0.01575800, # Low  
0.01577100, # Close  
148976.11427815, # Volume  
1499644799999, # Close time  
2434.19055334, # Quote asset volume  
308, # Number of trades  
1756.87402397, # Taker buy base asset volume  
28.46694368, # Taker buy quote asset volume  
17928899.62484339 # Ignore

(Source: <https://sammchardy.github.io/historical-data-download-binance/>)

1. **EXPLANATORY DATA ANALYSIS**

In this section we provided an understanding of the data in the context of exploration and visualizations. Details about variables categories, how we manipulated the datasets will be covered in *III. Feature Engineering* section.

We conduct initial check on the dataset to see the data counts and data types.

Graphical user interface, text, application

Description automatically generated

Figure 2.2 Data Types

Graphical user interface, text, application

Description automatically generated

Figure 2.3 Sample Dataset

We checked missing values and there were no missing values in our dataset.

Text

Description automatically generated with medium confidence

Figure 2.3 Missing values

We checked duplicates and found no duplicate value in our dataset.

Graphical user interface, text, application

Description automatically generated

Figure 2.4 Duplicate Value

With this dataset, we are particularly interested in understanding what features had a significant impact on trends next day to predict if we can invest that particular day or not. The correlation matrix below shows the correlation coefficients between several variables related to satisfaction. The darker the color is, the stronger the correlation is. The light color of red means a string correlation and dark means the opposite.

Calendar

Description automatically generated

Figure 2.6 Correlation Matrix Heatmap

We also wanted to check differences in second day’s trend to check if our dataset is balanced or imbalanced. First, we look at the overall count between the two level of target variable and group by the second day’s trend.

Overall, there were more increasing trend than stable of decreasing. We can compare the proportions of these groups below:

Chart, bar chart

Description automatically generated

Figure 2.7 Target Variable (Two\_Day\_Trend) Count

Chart

Description automatically generated with medium confidence

1. **FEATURE ENGINEERING**

The task here is to prepare all dataset needed from raw historical data before various modeling techniques processing to predict the next day trend in price of the ADA bitcoin. We imported them first to get a full understanding of it in our analysis in the EDA part and converted it into pandas dataframe

Graphical user interface, text, application, email

Description automatically generated

Text

Description automatically generated with medium confidence

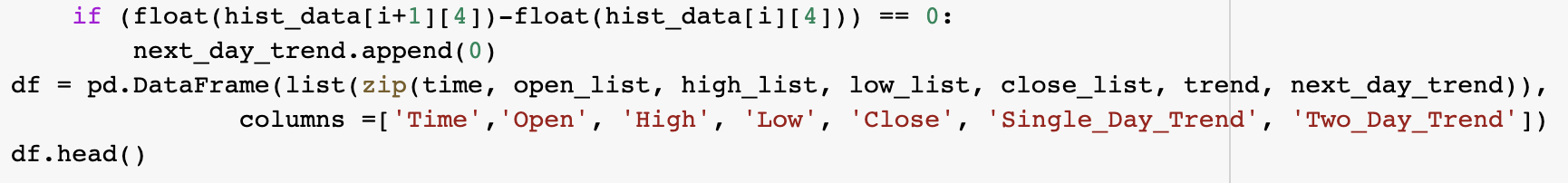
We used K line pull technique from binance API to fetch the data from repository

***A. Feature selection***

We found that data fetched were in the format which was not usable to predictive modeling. Hence, we converted this raw data into python dataframe using following codes:

Graphical user interface, text

Description automatically generated



We fetched every column from raw historical data into a separate list using a for loop and then we combined this data from list into pandas dataframe and made 7 columns out of it, which were

Time, Open, High, Low, Close, Single\_Day\_Trend, Two\_Day\_Trend.

**Time**: this column represents the timestamp of data when it was captured.

**Open**: this column tells us starting price of the bitcoin.

**High**: this column tells us highest price of the bitcoin in that particular day.

**Low**: this column tells us lowest price of the bitcoin in that particular day.

**Close**: this column tells us closing price of the bitcoin in that particular day

**Single\_Day\_Trend**: First, we took difference between opening price and closing price of the bitcoin for a particular day. If the difference is going up, we placed “1” in the column. If the difference is decreasing or neutral, we placed “0” in the column.

**Two\_Day\_Trend**: First, we took difference between closing price of this day and closing price of next day. If trend is going up, we assigned “1” else “0”.

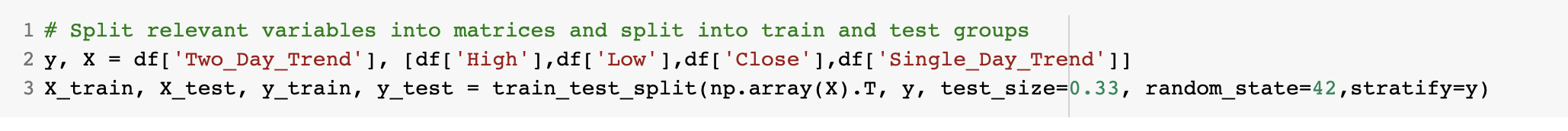
***B. Numerical Binning***

We did numerical binning on the target variable, when price increases – 1, when it goes down - 0.

Graphical user interface, text

Description automatically generated

These feature and target variables are separated for both the training and testing data in order to display our results.



***C. Imbalanced Data Handling***

We first checked if the data was imbalanced and found 53% of Target variable Two\_Day\_Trend is increasing, and rest is either neutral or decreasing. We need to make sure that we maintain the same percentage across both training and testing datasets. This kind of splitting the data maintaining the ratio is called "stratification". In this research, the dataset is balanced. When the data is imbalanced, there are some popular techniques such as handling the training set, K-fold Cross-Validation, ensembling different resampled datasets, resampling with different ratios and cluster the abundant class.

1. **MODELING TECHNIQUES**

After preparing the data, we can implement several machine learning methods and setperformance metrics to evaluate their performance with our historical data.

***A. Models***

*K-nearest Neighbors*

*Random Forest Algorithm*

***1.K-nearest Neighbor***

K nearest neighbor classification is a classification algorithm used to find the probability of success or failure of certain event. K-Nearest Neighbors algorithm (or KNN) is one of the most used learning algorithms due to its simplicity. With the help of K-NN, we can easily identify the class of a particular dataset. The data point is classified by a majority vote of its neighbors, with the data point being assigned to the class most common amongst its K nearest neighbors measured by a distance function.

***2.Random Forest***

A random forest model is an ensemble machine learning algorithm that creates multiple uncorrelated decision trees by taking a bagging approach to sampling along. It differs from bagging as it takes a random selection of features as opposed to all of them. The collective results from these trees are then used to make a classification for a given datapoint. Random forests is very accurate dealing with non-linear data as well as outliers and works well with large datasets. Conversely though, the training process is notably slow.

1. **EXPERIMENT SETTINGS**

There are some user defined functions that we have used in order to make it reusable and gauge the performance of our different models.

**Run\_model:** We have defined this function to run different classification functions and populate result with the time taken in the algorithm.

Graphical user interface, text, application

Description automatically generated

**Eval\_result:** This function has been defined in order to make various performance metrics and reports such as confusion matrix, accuracy score, F1 score etc.

Graphical user interface, application

Description automatically generated

1. *K nearest Neighbors*

Graphical user interface, text, application, email

Description automatically generated

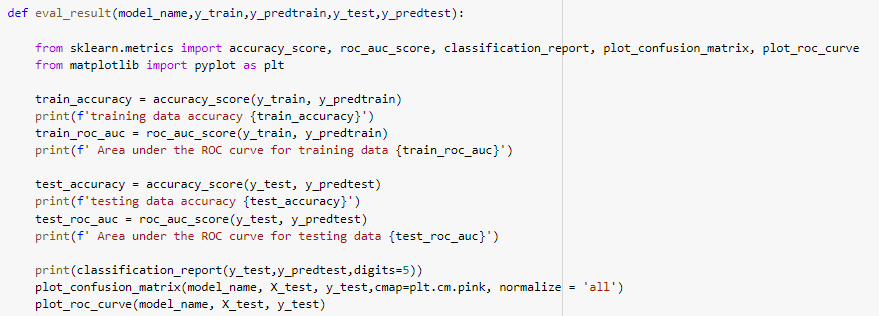
From the summary of the KNN classification, we see that the accuracy rate of the model was about 0.61, and the ROC are under the curve was about 0.60.

Table

Description automatically generated

1. *Random Forest*

A reproducible function is defined to easily call and evaluate our model performance by displaying accuracy information and plots. An example of this is shown below:



The number of trees to use is defined in the variable *n\_estimators\_hyp*, and the minimum number of samples required to split an internal node is defined in the variable *min\_sample\_leaf\_hyp*.

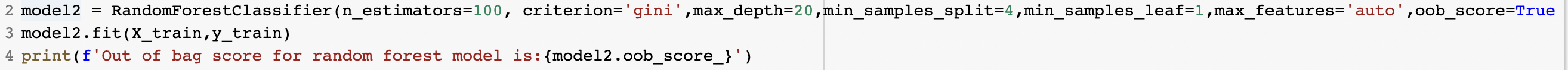


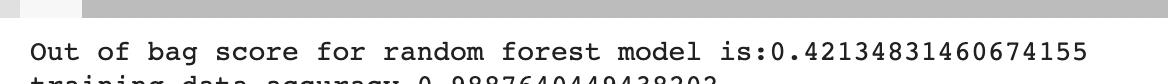
The *RandomForestClassifier* function from the *sklearn* library is called within a nested for-loop to create the models with each of the parameters. For example, a random forest with 100 trees is created with a minimum of 10 leaves. A few iterations of the models are collected and shown below with their Out-of-Bag score.

Graphical user interface, text, application

Description automatically generated

The OOB score represents the accuracy rate for based on the number of correctly predicted values from the out-of-bag sample. With this information, a random forest model is created using 100 trees and a minimum of sample leaf of 1 is stored. The OOB score for this model is approximately 0.42.





Calling our custom *eval\_result* function allows us to see the overall accuracy of this random forest model.

A picture containing text, receipt, screenshot

Description automatically generated

The training accuracy for the dataset is approximately 98% and the area under the ROC curve for it is about 98%, whereas testing accuracy for test dataset is 55% and area under ROC curve is 55%.

1. **RESULT AND DISCUSSION**

This experiment has been performed using two classification algorithms K nearest neighbor and random forest algorithm.

*A. Models performance Comparison*

|  |  |  |
| --- | --- | --- |
|  | Running time | Accuracy |
| K nearest neighbor | ~0.002 seconds | 0.61 |
| Random Forest | ~0.18 seconds | 0.56 |

B. Confusion Metrics:

1. *K nearest neighbors*

*Chart, treemap chart

Description automatically generated*

Approximately 61% of target variables were correctly identified.

Chart, line chart

Description automatically generated

We see a moderately strong ROC curve since the true positive rate is moderately high for this model. Area under the curve is 0.68

1. *Random Forest*

Chart, treemap chart

Description automatically generated

Approximately 56% of the target variables were correctly identified.

Chart, line chart

Description automatically generated

The AUC plot shows that the true positive rate is moderately high. This overview indicates that the random forest model was slightly less accurate in making these predictions. As we noted earlier, random forest models are moderately accurate. The training process is usually considerably slow though, especially since we had to create multiple random forests to find the optimal setting. Thus, it is important to consider the high accuracy and processing with a K nearest neighbor approach is sufficient in the analysis.

1. **CONCLUSION AND FUTURE SCOPE**

The K nearest neighbor was especially insightful in making predictive classifications since it was able to optimize the model. Going forward, we can use these models to better understand the initial problem of the dataset and how to best determine whether a day is suitable to invest or not.

From the comparison we could see that we are getting an accuracy of 61% for test dataset in KNN algorithm with time taken 0.002 seconds. Meanwhile, the Random Forest model also gets a moderate accuracy of 55% which is slightly less accuracy with more time taken as 0.18 second. Since it is important to consider high accuracy, in this project we would recommend predicting the next day trend using the KNN model method.

**REFERENCES**

* The Size and Quality of a Data Set. (2021). *Data Preparation and Feature Engineering for Machine Learning*. https://developers.google.com/machine-learning/data- prep/construct/collect/data-size-quality
* A Quick Introduction to KNN Algorithm. (2020).retrieved from <https://www.mygreatlearning.com/blog/knn-algorithm-introduction/>
* Yiu, T. (2019, July 12). *Understanding Random Forest*. Towards Data Science. Retrieved March 20, 2021 from https://towardsdatascience.com/understanding-random-forest- 58381e0602d2
* SAM MCHARDY. Save Historical data from Binance. Retrieved from <https://sammchardy.github.io/historical-data-download-binance/>