Project Report

# GitHub URL

https://github.com/amaykmr/UCDPA\_Amay\_Kumar

# Abstract

The project aims to provide predictions of a cryptocurrency Chainlink (LINK) based on the prices of 4 larger (greater market capitalisation) cryptocurrencies (Bitcoin (BTC), Ethereum (ETH), Binance Coin (BNB) and Cardano (ADA)). For the project, we use averaged price action or an average of OHLC (open, high, low, close) values) on an hourly interval and use different machine learning models namely Multiple Linear Regression, Support Vector Regression and AdaBoost Regression to determine the price of LINK. The best performing model with a high R2  value of 0.89 on the test data was the Support Vector Regression model (SVR).

# Introduction

I wanted to work with financial data as I trade securities as a secondary source of income. I currently swing trade based on Technical Analysis. Swing-trading strategy is when one takes positions for a short period of time (usually longer than a day and up to a few weeks).

However, given the variability in cryptocurrency prices within a day (they can move as much as 10% in a day) I wanted to explore the idea of a data-driven scalping trading strategy, where I would base the price of a cryptocurrency on the prices of 4 other (larger) cryptocurrencies. Scalping strategy looks to earn 0.1%-2% on trades and occurs more frequently, where one holds onto an asset for a very short time period (hours to a day).

I further wanted to base the prediction model on the prices of other currencies and not other factors such as DateTime or trading volume as the price of one cryptocurrency strongly affects the price of the other.

# Dataset

The datasets were downloaded from the cryptodatadownload.com website. This website contains historical data from many exchanges and I chose the prices from the Binance exchange which is the largest in the world. The website is an established name in the market running a business that gets cited in academic journals as a cryptocurrency pricing data provider.

I required hourly data because I wanted a large sample space. I chose this source since the data was free and easy to procure (only requires sign-up) and required moderate cleaning to showcase my data wrangling skills. URL of the downloaded data: <https://www.cryptodatadownload.com/data/binance/>

There were 5 datasets downloaded from the website, namely:

1. BTC/USDT
2. ETH/USDT
3. BNB/USDT
4. ADA/USDT
5. LINK/USDT

And each dataset contains the following columns:

1. Unix Timestamp - This is the unix timestamp or also known as "Epoch Time". Use this to convert to your local timezone
2. Date - This timestamp is converted to NY EST Standard Time
3. Symbol - The symbol for which the timeseries data refers
4. Open - This is the opening price of the time period
5. High - This is the highest price of the time period
6. Low - This is the lowest price of the time period
7. Close - This is the closing price of the time period
8. Volume (Crypto) - This is the volume in the transacted Ccy. Ie. For BTC/USDT, this is in BTC amount
9. Volume Base Ccy - This is the volume in the base/converted ccy. Ie. For BTC/USDT, this is in USDT amount
10. Trade Count - This is the unique number of trades for the given time period

The aim of the cleaning process would be to get 2 columns each from the 5 datasets, the timestamp (DateTime) and an averaged value of the ‘Open’, ‘High’, ‘Low’ and ‘Close’ values for each cryptocurrency and then merge them along the timestamp.

# Implementation Process

Once the datasets had been loaded onto the Jupyter notebook, I had 4 major tasks to carry out:

1. **Data cleaning and merging**

* Datasets exploration: Using .head and .info method, get an idea of the dataset.
* Remove multi-index: Datasets came through as a multi-index and only one column where the column name was the data source.
* Clean data function 1: Use a custom function to reset the multi-index, rename columns using a dictionary and drop the first row (as this is the same as column names).
* Check for missing values: use isnull() method with sum() method to calculate the name of null values that might require transformation.
* Check for duplicate values: use .value\_counts() method to calculate the number of duplicate values.
* Get average of data OHLC: Combine OHLC (open, high, low, close) into one figure that can be used for analysis using a custom function. All columns are objects, ‘date’ will need to be converted into datetime object while ‘open’, ‘high’, ‘low’, and ‘close’ columns will need to be converted into floats. Once ‘open’, ‘high’, ‘low’, and ‘close’ columns have been averaged out, they will be dropped along with the ‘symbol’ column.
* Dropping null values: During conversion, 25,936 ‘date’ values did not convert to datetime object in df\_btc (BTC) and df\_eth (ETH) datasets. These returned ‘NaT’ null value. The remaining values (15,248) also happen to be the exact number of rows that are found in the df\_bnb (BNB), df\_ada (ADA) and df\_link (LINK) datasets. This is likely due to the datasource changing the datatime format at the moment. Since all datasets would require equal values to merge and any NaT values would drop anyway, we drop these values at this moment.
* Renaming ‘average’ column names: We change the column name ‘average’ in all datasets to the corresponding asset name using a dictionary ('average' column in df\_btc becomes 'average\_btc', 'average' column in df\_eth becomes 'average\_eth' etc.).
* Merging datasets: Use a custom function to merge all 5 datasets into a single dataset namely df)

1. **Exploratory Data Analysis**

* Plot datasets: Separately plot the 5 datasets with datetime on the x-axis and average asset price on the y-axis.
* Comparative datetime plot: Plot all 5 datasets together with datetime on the x-axis and average asset price on the y-axis.
* Summary statistics: Get the summary statistics of the average for all 5 datasets and assets.
* Correlation matrix: Develop a correlation matrix for the 5 assets.

1. **Data preparation and model training**

* Segregation of feature and target variables: ‘df’ columns of 'average\_btc', 'average\_eth', 'average\_bnb', 'average\_ada' were added as feature variables ‘X’ dataset and the target variable of ‘average\_link’ was assigned the ‘y’ dataset.
* Train-test split: The two datasets ‘X’ and ‘y’ are split using a 70:30 train-test split ratio.
* Multiple Linear Regression: The multiple regression model is called and fit to the training data, before being used to predict the test ‘X’ dataset. Then, various regression metrics are calculated for grading.
* Support Vector Regression: The feature variables are transformed using a minimax scaler. We use a boxplot to see the resulting transformed variables. Then the support regression model is called and fit to the training data, before being used to predict the test dataset of feature variables. Then, various regression metrics are calculated for grading.
* AdaBoost Regression: The AdaBoost regression model is called and fit to the training data, before being used to predict the test ‘X’ dataset. Then, various regression metrics are calculated for grading.

1. **Developing Insights**

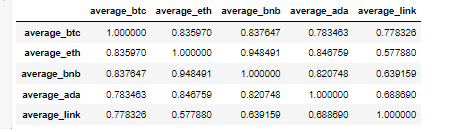
* Correlation Matrix: Pearson correlation method was called on the dataframe to get an understanding of the correlation between the 5 cryptocurrencies.
* Multiple Regression Plot: The true values of the test data were plotted on the x-axis with the predicted values on the y-axis on a scatter plot to look at the differences between the two values. The regression metrics are added below the plot to have them available to develop insights over the model.
* Support Vector Regression Plot: The true and predicted values of the test data were charted on the scatter plot to look at the differences between the two values. Regression metrics were added as above.
* Multiple Regression Plot: The true and predicted values of the test data were charted to the scatter plot to look at the differences between the two values. Regression metrics were added as above.

To conclude, data was loaded onto the Jupyter notebook, it was then cleaned to get one dataframe that contained the timestamp and the averaged OHLC values of the 5 assets before fitting 3 different machine learning models and then measuring them on various metrics. Finally, insights were developed on the correlation

# Results

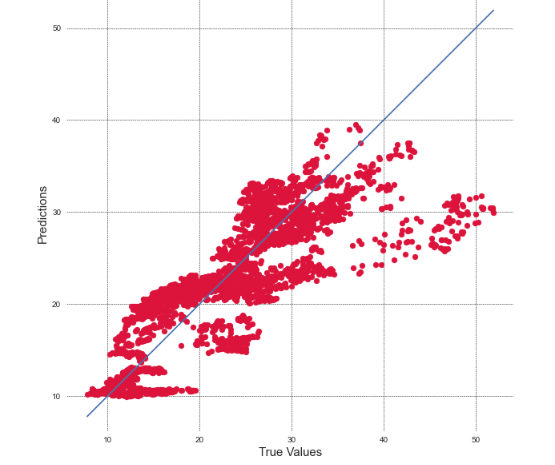
The results from the exploratory data analysis and prediction modelling are below:

1. Correlation Matrix



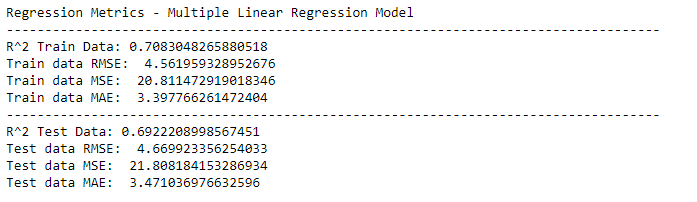
‘average\_link’ prices are positively correlated to all of the other cryptocurrencies, with it being most correlated to the ‘average\_btc’ price (0.78) and least correlated to the ‘average\_eth’ price (0.56). Other cryptocurrencies also share a greater correlation, with the ‘average\_btc’ and ‘average\_bnb’ showing the most positive correlation using the Pearson method.

1. Multiple Linear Regression (MLR)
   1. Graph



The multiple linear regression scatter plot shows the True Values of the test on the x-axis and the predicted values on the y-axis. We can see that the model is poor at higher true values of ‘average\_link’ but overall the model undervalues the ‘average\_link’ price consistently.

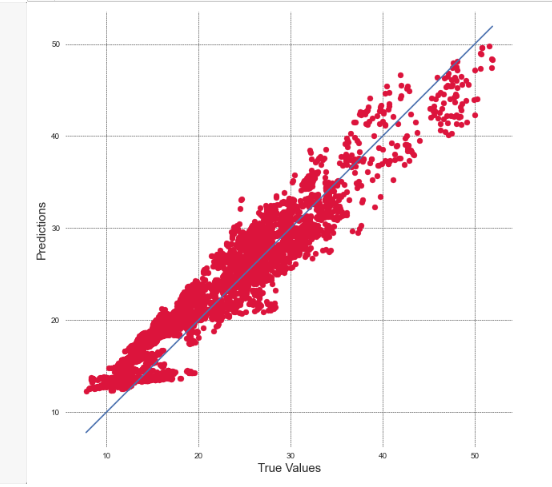
* 1. Metrics



The multiple linear regression model gives has an R2  score of 0.69 on the test data, a Root Mean Squared Error of 4.66, Mean Squared Error of 21.81 and Mean Absoulte Error of 3.4.

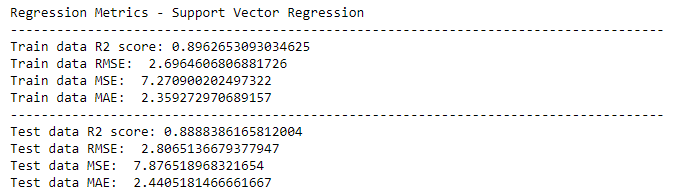
Scores on the Training Data were marginally better with slightly higher R2 and lower RMSE, MSE and MAE.

1. Support Vector Regression (SVR)
   1. Graph



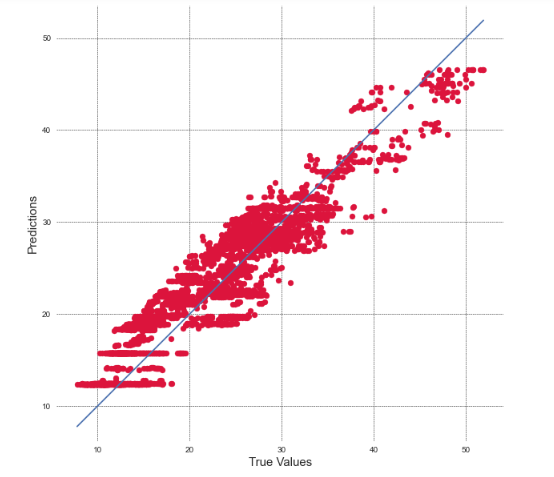
The Support Vector Regression scatterplot gives values that are more closely located around the line of fit, and the differences between the true values and the predicted values are less in comparison to the Multiple Linear Regression model.

* 1. Metrics



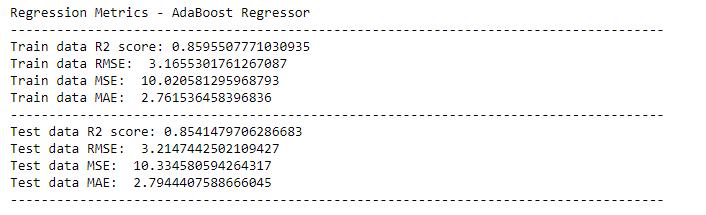
The model has a test data R2  score of 0.89, with an RMSE of 2.81, MSE of 7.87 and MAE of 2.44. Scores on the support vector regression model are much higher in comparison to the baseline model with higher R2 and lower RMSE, MSE and MAE. The training data scores marginally better.

1. Adaboost Linear Regression (ABR)
   1. Graph



The Adaboost Regressor scatterplot gives values that are closely located around the line of fit, and the differences between the true values and the predicted values are similar to the Support Vector Regressor.

* 1. Metrics



The model has a test data R2  score of 0.85, with an RMSE of 3.21, MSE of 10.33 and MAE of 2.79. This scores the Adaboost Regression model between the baseline model of linear regression but closer to the Support Vector Regression model albeit with slighter lower accuracy scores. The training data scores were better except for the R2 which was marginally lower.

# Insights

* Overall, the Support Vector Regression model performs the best against the accuracy metrics in comparison to the other two models. It provides the most accurate predictions between the three models.
* ‘average\_link’ prices are positively correlated to all of the other cryptocurrencies, with it being most correlated to the ‘average\_btc’ price (0.78) and least correlated to the ‘average\_eth’ price (0.56). Other cryptocurrencies share a greater correlation, with the ‘average\_btc’ and ‘average\_bnb’ showing the most positive correlation using the Pearson method.
* The multiple linear regression model acts as a baseline, explaining about 70% of the variance in LINK overall. Despite a small RMSE of about 4.7, it is the worst performing model. It expectedly fails to capture the complex data generating process lying beneath the price mechanism.
* The Support vector model makes improvements on the linear model, successfully explaining about 90% of the variance in LINK. As can be seen with the smallest accuracy metrics (RMSE, MSE, MAE) on the test data, it is the best performing model. Further, there doesn't seem to be overfitting in the model either, with similar R-squared metrics on the training and test data.
* Ada boosting surprisingly did not lead to the expected outcome. And for the present context would be ranked below SVR. With RMSE, MSE and MAE on average being 24% larger on the training data, and being 20% larger on the test data, than that compared to SVR.