

**Midterm Paper**

**Predicting Ethereum Prices:**

**Applications of XGBoost and Linear Regressions**

For

Capstone

Daniel Valverde

University of Denver, Ritchie School of Engineering

20 July 2022

Claudio Delrieux

Introduction: Purpose and Significance

The purpose of my project is to determine to what extent can I predict prices of Ethereum with XGBoost or Linear Regression. The ability to predict price movement could be very lucrative and it seemed like a fun problem to try and solve. There are an endless number of different approaches that can be used to try and solve this problem. I have been interested in predicting price movements in stocks and/or cryptocurrencies for almost 7 years now and this is a project I will be spending more time on when I am done with this program.

Research Question

Which feature set works better with XGBoost or Linear Regression at predicting future prices of Ethereum

**Description of the Dataset**

In order to get up to date data I connected to the API of coingecko.com. This API will allow you to pull almost any available cryptocurrency and you can check the price from almost any exchange. The nice thing about this site is that it allows you to pull historic price data. For the purposes of this project, I pulled 5 years’ worth of data for price, and total volume. It returns the daily price at a specific time each day of Ethereum which is the second largest cryptocurrency. The price is quite volatile at times, which makes this dataset interesting.

**Data Preprocessing**

I used two different feature sets to see what would perform better with XGboost and Linear Regression at predicting future prices of Ethereum.

**The features using manually calculated fields**

Using the dataset that I pull from coingecko I create some manually calculated features which are some commonly used stock market trading indicators.

Manually Calculated Features:

* Relative Strength Index (RSI)
* Exponential Weighted 14 day Moving Average (EWM)
* Difference- change in price from yesterday
* MACD (Moving Average Convergence Divergence) – difference between the 12 and 26 day moving averages
* Macd\_s – the 9 day moving average of the macd for the trigger line
* Macd\_h -the difference between macd and macd\_s

Other Features: Volume

The target Variable that I am trying to predict is the price of Ethereum at a specific time on that day.

When creating some of these moving average based features this Created NaN’s for the beginning rows of data so after making the calculations I needed to drop all rows containing NaN’s.

**The features using a 7 day sliding window**

The other features that I tried making predictions with was using a sliding window of 7 days. Which means that the target variable was that days price and the features were the day priors’ price through 7 days prior to the target price.

Table

Description automatically generated

**Exploratory Data Analysis**

The first thing that I did was check for trends and Seasonality. Over the period that I was looking at there is no seasonality or trends. In order to analyze this I utilized the statsmodel library’s seasonal\_decompose function on the target variable which is the price of Ethereum over time.

**Timeline

Description automatically generated**

The next thing that I did was I created a correlation heatmap using the seaborn library. Below you can see that not many of these calculated features have a high correlation with price except for the exponential weighted moving average has an extremely high correlation and volume also has a higher correlation with price.

Chart, treemap chart

Description automatically generated

I created the same correlation matrix with the sliding window feature set but all the features had approximately .99-.98 correlation. The correlation decreased as the feature price is further days away from the target prices date. Saying that 7 days away was less correlated than 1 day away from the target price.

**Data Splitting**

On the calculated features set, first I used a standard scaler on the features. Then I shifted the feature data by 1 day so that we are making predictions using the data from the day prior. Lastly I performed a train and test split utilizing the most recent 60 days as the test set and all the data prior to that as the train set.

On the Sliding Window of 7 days features set I used a standard scaler and also used the most recent 60 days as the test set. And All the data prior to 60 days as the test set.

Model Building And Evaluation

In the first XGBoost model using the calculated feature set it did not perform well according to the RMSE or R Squared values. After performing a grid search for hyper parameter tuning it improved it substantially as can be seen in the R Squared and RMSE values of the Xgb\_calc\_grid column in the table below. The Linear Regression using the calculated features performed quite well at first glance but when I dove deeper into which features it was heavily weighted towards the feature exponential weighted moving average. I then decided to see what would happen if I removed that feature and it through the entire model off. This led me to believe that the other features were not good features for the linear regression model.

I then decided to try new a new feature set using a sliding window of seven days. I performed an XGBoost model on this new feature set and it performed quite poorly according to the RMSE and R Squared values which can be seen below in the column Xgb\_sw. I then performed a grid search on the XGBoost model to tune the hyperparameters. This improved the results substantially but still not enough for my liking.

Table

Description automatically generated

The Linear Regression on the Sliding Window features performed pretty well so I dove deeper into this. After attempting many different variations on the sliding window feature set the Linear Regression using 360 days as the test set on a sliding window of 6 days through 15 days had the most interesting results. The predictions seem to be shifted to the left on the test set as can be seen on the graph below. The RMSE of the train and test was 137 and 327. The R squared was .959 and .866 on the train and test sets respectively.

Chart, histogram

Description automatically generated

Conclusion

XGBoost seemed to perform alright on the calculated features set but I would not necessarily use this model to make trades given the erratic behavior it had sometimes.

Linear Regression on the Sliding Window features set when I used price features from 6 to 15 days prior seemed like it would be good to use for trading signals at first glance. Although it is still hard to tell just using these metrics and I would need to build a system that simulates making trades given a model and calculates profitability over time. I could potentially implement streaming with pyspark moving forward.

Overall this was a very fun problem to solve and there are many more models and features out there to try and use.