**Exploring and Predicting Cryptocurrency prices**

**Link to Project Code:**

https://github.com/peter05/Springboard2018/blob/master/Crypto%2BAnalysis%2B-%2B7%2BDay%2BPrediction%2B-%2BFinal.ipynb

**Data Set & Data Wrangling:**

For exploring the Cyprocurrency market data, I explored several possible data sources (APIs) including specific exchanges.

The trade-off for using one particular large exchange is that users may be skewed towards a specific country or region in the world. As some of my metrics take into account volume over time, I believe an aggregation across multiple large exchanges would be more ideal for my analysis.

For this purpose, I used cryptocompare API which aggregates data from over 90+ exchanges worldwide. I utilized functions to automate the download and saving of the particular coin or token of interest in my analysis.

Further, I used the Technical analysis library (Ta-lib) to re-create the most popular technical analysis indicators that I would using in my models. Many of the indicators were candlelight pattern identifiers and many of them returned all zero figures. I hypothesize this may be due to the fact that cryptocurrency markets are traded 24/7 while candlelight trading strategies were meant for trading in markets that had a daily open and close. As cryptocurrency markets are always continuous we see less variations in the high, low, open and close per time duration specified.

Nonetheless there were still candlestick indicators which were noted which can be a potential indicator of a bull/bear reversal in trend.

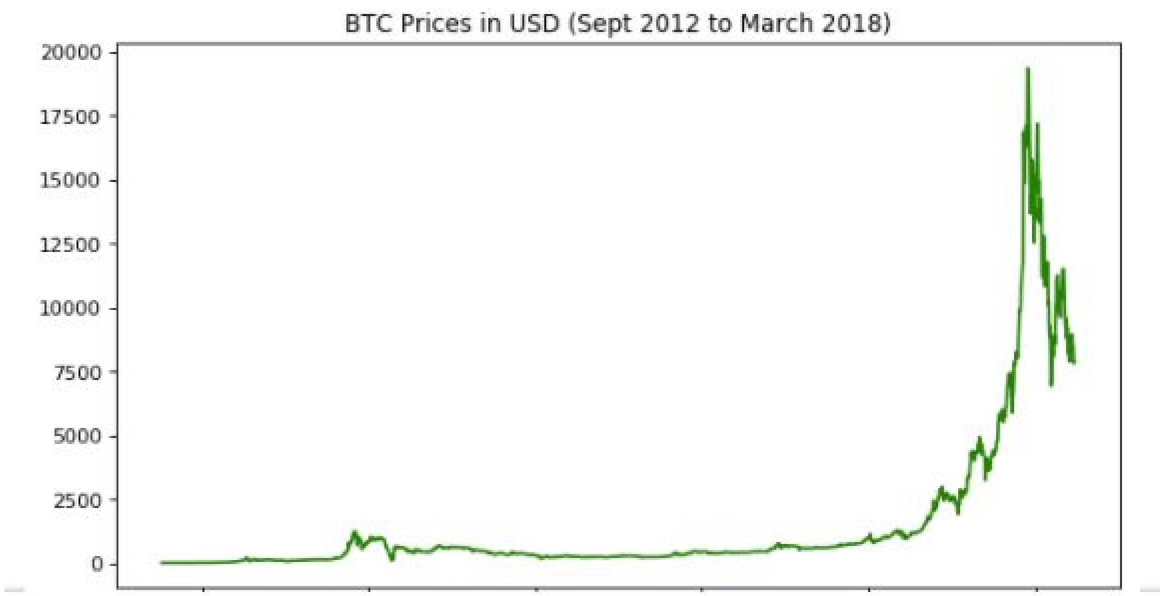
**Problem:**

When to sell and buy cryptocurrencies are typically a emotionally charged decision for many. Two common terms referenced by many in the crypto world are the acronyms ‘FOMO’ and ‘HODL’ which stand for ‘Fear of missing out’ and ‘Hold on to Dear Life’ respectively. My goal is to develop a prediction model using machine learning principles to help aid people in making a less biased decision on when one should buy or sell a particular cyptocurrency asset.

**Process:**

I will be reviewing my analysis for predicting prices 7 days ahead for Bitcoin but essentially my analysis can be replicated for different time periods and alternative coins. From my preliminary analysis 1 day ahead saw better accuracy while 30 day ahead saw little predictive power as one may expect. I also explored predicting 1 hour and 4 hour observed and saw mixed results as high volatility is observed and there may be outside factors that I am not taking into account.

I ingested and loaded data for BTC from September 2010 to March 2018.




From the chart we see exponential growth in 2017 and the current reversal we are observing in 2018. Notably, there was also a significant rise and fall price in 2014.

To prepare the data for my models -

- Created my feature variables using the Technical Analysis library
- Removed dependent variables/columns that were all 0's which removed many candlelight indicators which were not observed
  - This is mostly likely due to the fact that markets for cryptocurrencies are typically open 24/7, with the exception with exchanges being down for technical difficulties
- Created my target price variable which will be my independent variable by shifting the close price by 7 days to ensure I do not have access to future information
  - As a result, I also need to truncate some of the initial dates to run my models

I am examining a range of model algorithms to test their prediction power -

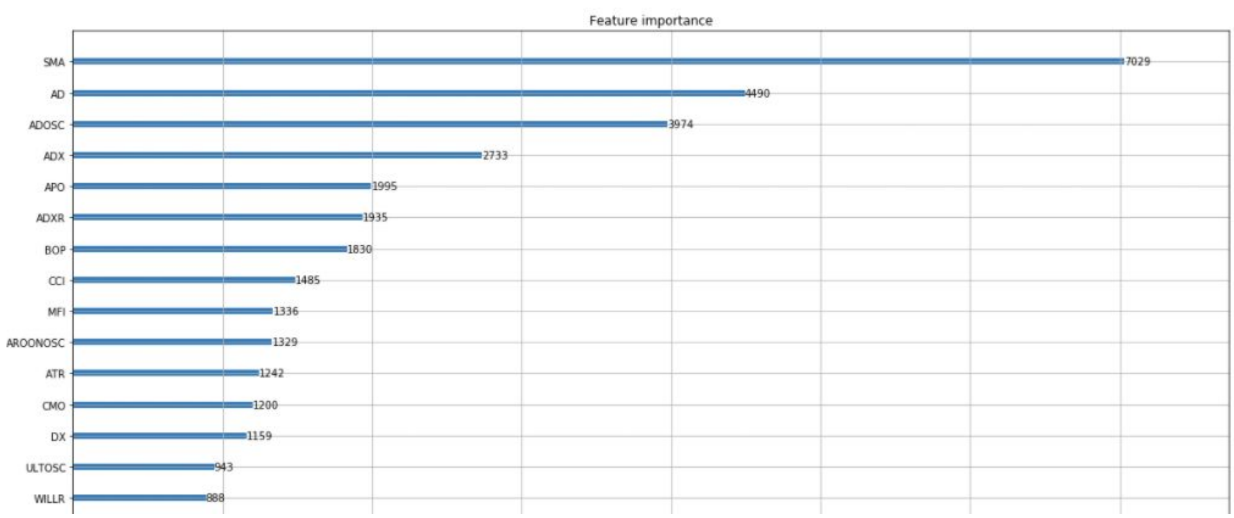
- Linear - basic regression model and we generally observe a smoothing effect in the prediction

- Ridge - introduces a penalty term to features which would help reduce overfitting
- Kernel Ridge - adds the kernel trick to ridge regression which allows calculation of an underlying relationship in a high dimensional space
    - A black box technique as high dimensional spaces (higher than 3-D) are not able to be visualized
- KNeighbors - common algorithm for predictions using similar points
- Tree - regular decision tree
- Gradient Boosting - Popular machine learning algorithm that is often used today, it is an ensemble modeling technique
    - Resource intensive to run and optimize so I would recommend using XGBoost over scikit-learn's gradient boosting
- XGBoost - Based on same principles of gradient boosting but observes significant improvement in speeds as it allows for parallel processing of some calculations

For the above model algorithms mentioned above, to take advantage of the additional processing power many people now have access to, I further hyper optimize parameters using Sk.Learn's GridsearchCV. And as I am dealing with time series I also utilize TimeSeriesSplit to ensure my splits are time sequential.

XGBoost also has a useful feature to look at relative importance of features by counting the number of times each feature is split on across all boosting rounds (trees) in the model. This is an advantage of XGBoost over a blackbox algorithm such as Kernel regression.

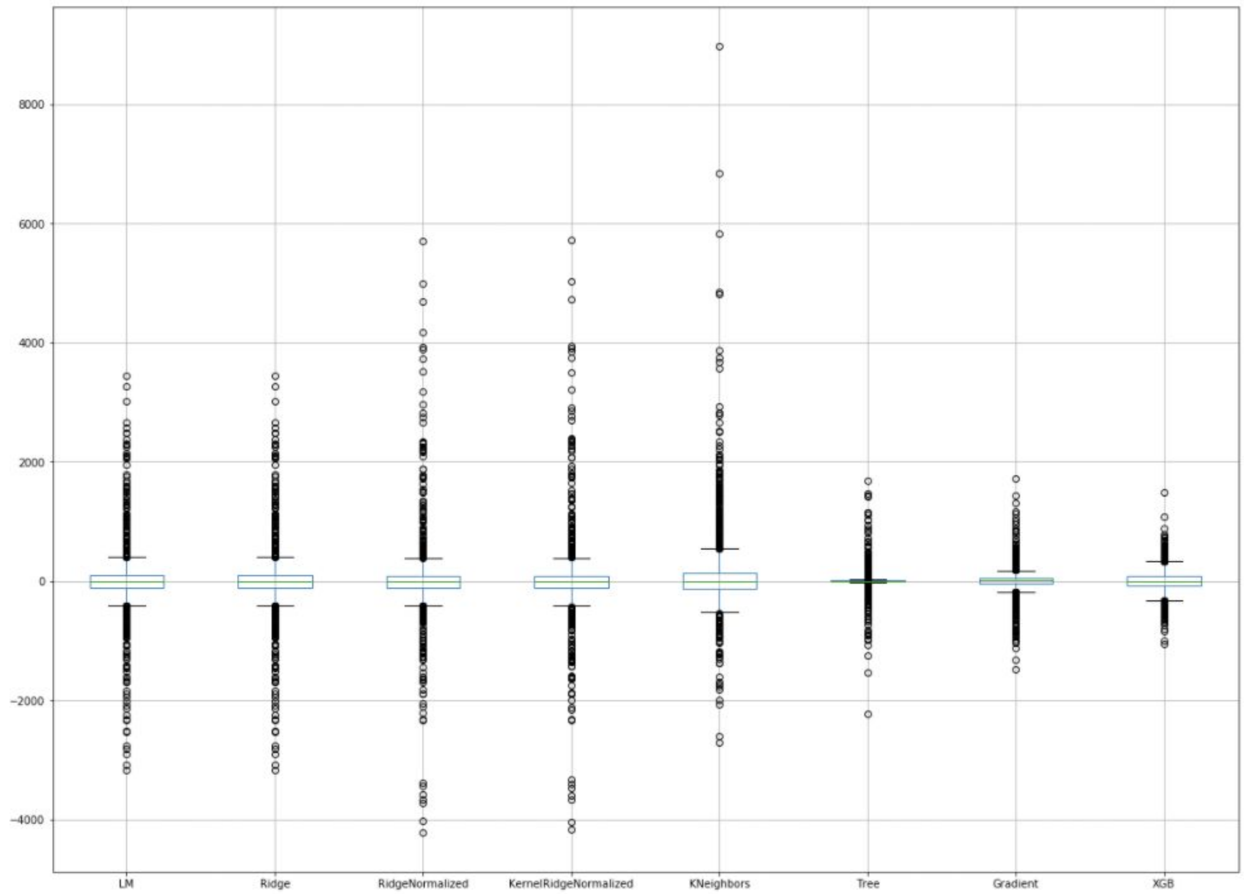
Here is a snapshot of my XGBoost plot_importance graph:




The most important features are related to moving averages and not shown the least frequent are candlestick indicators. This is not too surprising as the frequency of candlestick indicators present were sometimes low.

Finally for evaluation of the models, I looked at multiple things -

- Boxplots to see how the models perform against outliers and whether they perform better for price increases/decrease based on their distribution around 0
    - As observed below my tree based models perform the best with XGBoost performing the best
    - Models also appear to able to predict both price increase and decreases




- I also calculated the models RMSE where lower would indicate a better model
- I created pseudo confusion matrixes to get a sense of when the model predicts correctly a change in direction of the price
    - Note this does not take into account the magnitude of the directional change
    - Confusion matrix shows that our models perform better for data near the present versus historical numbers
        - This may be due to the fact that there were historical periods of bitcoin prices where it was relatively stable
        - It would be good to explore re-running models on shorter time periods while the trade off would be be losing out on observation points

Confusion Matrix - Total Counts of Accurate Prediction of Change in Price for BTC

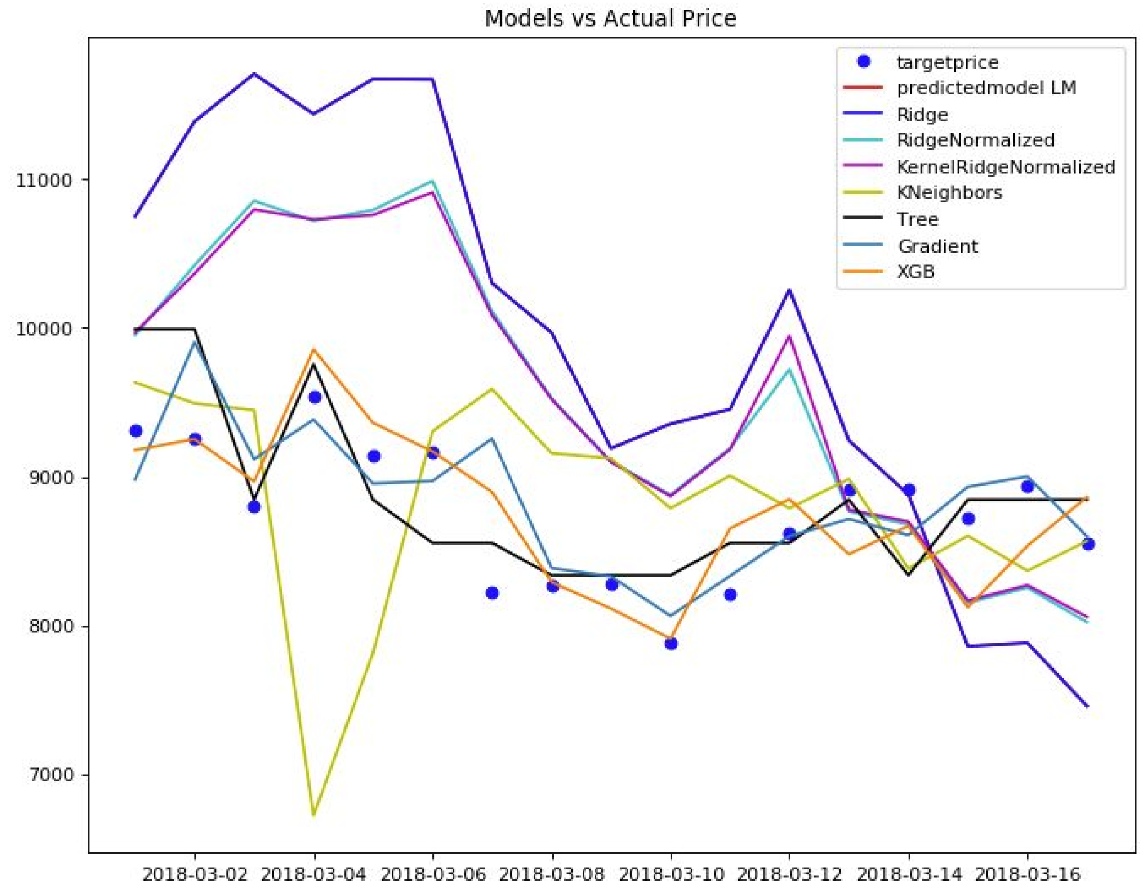
| | LM_CM | Ridge_CM | RidgeN_CM | RidgeKM_CM | KNeighbors_CM | Tree_CM | Gradient_CM | XGB_CM |
|---|---|---|---|---|---|---|---|---|
| Exception - No change | 3 | 3 | 3 | 3 | 13 | 1078 | 102 | 3 |
| Predicted Decrease / Actual Decrease | 440 | 440 | 433 | 434 | 437 | 265 | 420 | 475 |
| Predicted Decrease / Actual Increase | 484 | 484 | 506 | 499 | 509 | 100 | 453 | 442 |
| Predicted Increase / Actual Decrease | 420 | 420 | 427 | 426 | 421 | 106 | 390 | 385 |
| Predicted Increase / Actual Increase | 558 | 558 | 536 | 543 | 525 | 356 | 540 | 600 |

Confusion Matrix - Total Counts of Accurate Prediction of Change in Price in 2018

| | LM_CM | Ridge_CM | RidgeN_CM | RidgeKM_CM | KNeighbors_CM | Tree_CM | Gradient_CM | XGB_CM |
|---|---|---|---|---|---|---|---|---|
| Exception - No change | NaN | NaN | NaN | NaN | 2 | 36 | NaN | NaN |
| Predicted Decrease / Actual Decrease | 25.0 | 25.0 | 23.0 | 24.0 | 18 | 18 | 26.0 | 34.0 |
| Predicted Decrease / Actual Increase | 19.0 | 19.0 | 22.0 | 22.0 | 23 | 5 | 14.0 | 9.0 |
| Predicted Increase / Actual Decrease | 17.0 | 17.0 | 19.0 | 18.0 | 24 | 3 | 16.0 | 8.0 |
| Predicted Increase / Actual Increase | 19.0 | 19.0 | 16.0 | 16.0 | 13 | 18 | 24.0 | 29.0 |

Confusion Matrix - Total Counts of Accurate Prediction of Change in Price in 2017

| | LM_CM | Ridge_CM | RidgeN_CM | RidgeKM_CM | KNeighbors_CM | Tree_CM | Gradient_CM | XGB_CM |
|---|---|---|---|---|---|---|---|---|
| Exception - No change | NaN | NaN | NaN | NaN | 2 | 232 | 5 | NaN |
| Predicted Decrease / Actual Decrease | 84.0 | 84.0 | 75.0 | 76.0 | 71 | 36 | 69 | 102.0 |
| Predicted Decrease / Actual Increase | 101.0 | 101.0 | 99.0 | 97.0 | 102 | 12 | 89 | 73.0 |
| Predicted Increase / Actual Decrease | 59.0 | 59.0 | 68.0 | 67.0 | 71 | 5 | 71 | 41.0 |
| Predicted Increase / Actual Increase | 121.0 | 121.0 | 123.0 | 125.0 | 119 | 80 | 131 | 149.0 |




Above is a plot of the models against our target price for a range of time in March 2018, we can see that Tree models perform the best while the linear and ridge models over predicted. K-Nearest Neighbors performed poorly.

**Applying algorithms and concepts to another large “blue chip” coin - Ethereum**

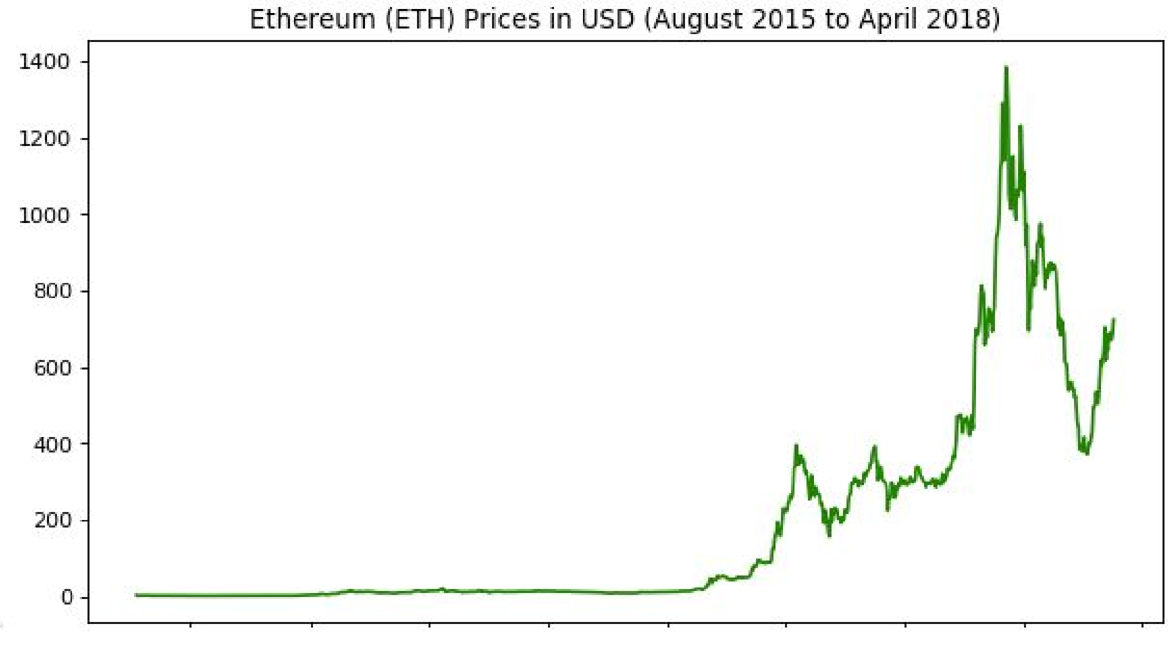
Utilizing my existing project code, I will be testing the applicability of my models on another popular cryptocurrency. Ethereum is another popular cryptocurrency which currently sits at #2 market cap among all coins. Ethereum is currently used to support many other tokens or other cryptocurrency projects and is utilized for it’s smart contract capabilities. Ethereum also have significant amount of fiat pairings similar to Bitcoin which may allow Ethereum to be less correlated with bitcoin price movement. Despite Ethereum having many direct fiat to coin pairings, Ethereum have at times been strongly correlated with bitcoin. As shown in correlation matrixes below, there were less correlation in 2017 when Ethereum experienced a large increase in price while a strong correlation in 2018 when whole market sentiment and prices appeared to move more in tandum. In general, I would expect my models and results with Ethereum to perform in a similar fashion to Bitcoin.

Correlation matrix of coins versus BTC and ETH in 2017

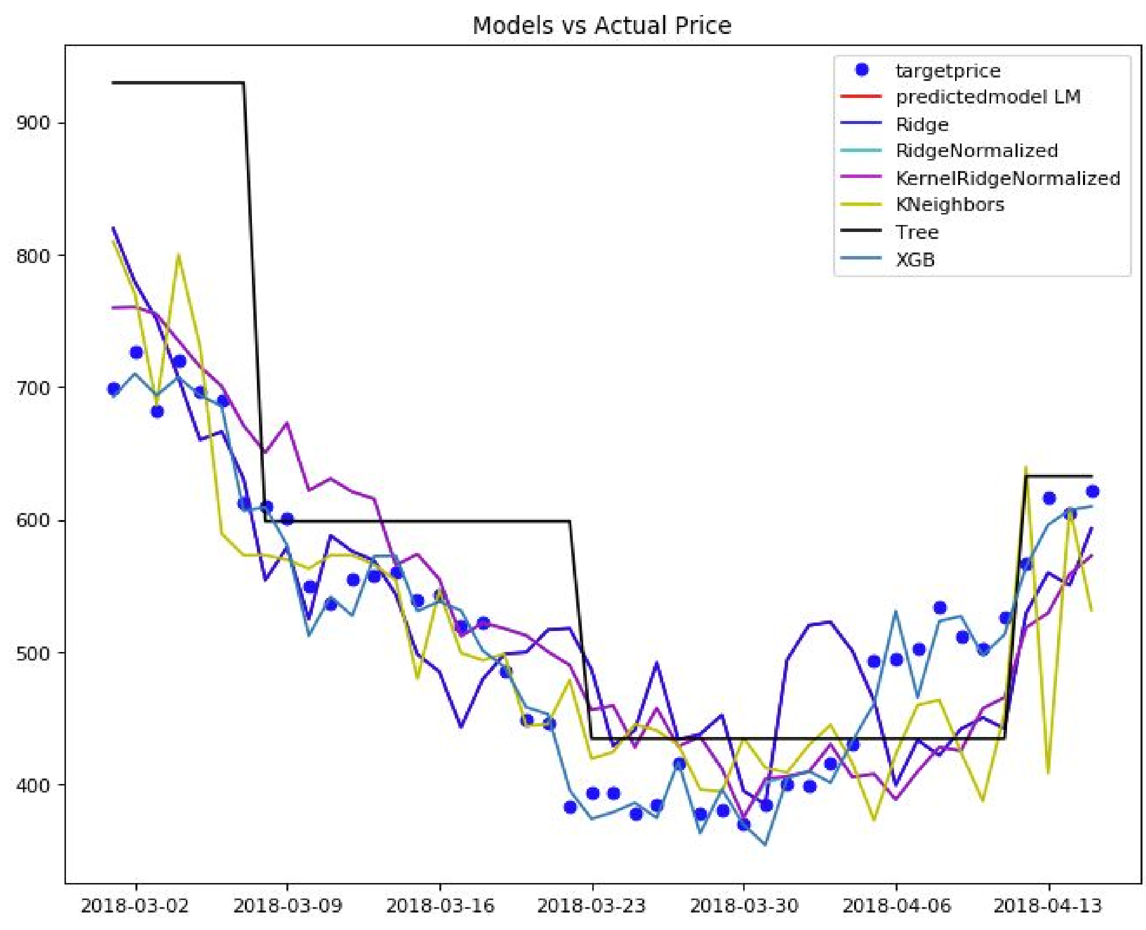
| | BCH | DASH | ETH | LTC | SC | STR | XEM | XMR | XRP | BTC |
|---|---|---|---|---|---|---|---|---|---|---|
| BCH | 1.000000 | 0.480873 | 0.308905 | 0.222741 | 0.217139 | 0.024506 | 0.187470 | 0.254298 | 0.059802 | 0.072468 |
| DASH | 0.480873 | 1.000000 | 0.582625 | 0.419035 | 0.347706 | 0.234419 | 0.408356 | 0.581108 | 0.191553 | 0.473970 |
| ETH | 0.308905 | 0.582625 | 1.000000 | 0.486109 | 0.408488 | 0.293128 | 0.460861 | 0.608007 | 0.279418 | 0.506469 |
| LTC | 0.222741 | 0.419035 | 0.486109 | 1.000000 | 0.372584 | 0.337519 | 0.423981 | 0.495404 | 0.372111 | 0.492560 |
| SC | 0.217139 | 0.347706 | 0.408488 | 0.372584 | 1.000000 | 0.420654 | 0.372832 | 0.417161 | 0.283258 | 0.375130 |
| STR | 0.024506 | 0.234419 | 0.293128 | 0.337519 | 0.420654 | 1.000000 | 0.373894 | 0.362818 | 0.529077 | 0.289252 |
| XEM | 0.187470 | 0.408356 | 0.460861 | 0.423981 | 0.372832 | 0.373894 | 1.000000 | 0.410027 | 0.324917 | 0.427295 |
| XMR | 0.254298 | 0.581108 | 0.608007 | 0.495404 | 0.417161 | 0.362818 | 0.410027 | 1.000000 | 0.300031 | 0.529819 |
| XRP | 0.059802 | 0.191553 | 0.279418 | 0.372111 | 0.283258 | 0.529077 | 0.324917 | 0.300031 | 1.000000 | 0.255953 |
| BTC | 0.072468 | 0.473970 | 0.506469 | 0.492560 | 0.375130 | 0.289252 | 0.427295 | 0.529819 | 0.255953 | 1.000000 |

Correlation matrix of coins versus BTC and ETH in 2018

| | BCH | DASH | ETH | LTC | SC | STR | XEM | XMR | XRP | BTC |
|---|---|---|---|---|---|---|---|---|---|---|
| BCH | 1.000000 | 0.877111 | 0.818380 | 0.777712 | 0.594802 | 0.575424 | 0.535669 | 0.820220 | 0.624868 | 0.857917 |
| DASH | 0.877111 | 1.000000 | 0.871950 | 0.869526 | 0.734408 | 0.714572 | 0.683570 | 0.893050 | 0.778367 | 0.906845 |
| ETH | 0.818380 | 0.871950 | 1.000000 | 0.769840 | 0.650680 | 0.650134 | 0.679346 | 0.830954 | 0.678463 | 0.836356 |
| LTC | 0.777712 | 0.869526 | 0.769840 | 1.000000 | 0.692478 | 0.620606 | 0.559748 | 0.781648 | 0.679025 | 0.853071 |
| SC | 0.594802 | 0.734408 | 0.650680 | 0.692478 | 1.000000 | 0.647803 | 0.664264 | 0.656861 | 0.729910 | 0.732648 |
| STR | 0.575424 | 0.714572 | 0.650134 | 0.620606 | 0.647803 | 1.000000 | 0.748364 | 0.670166 | 0.854022 | 0.665449 |
| XEM | 0.535669 | 0.683570 | 0.679346 | 0.559748 | 0.664264 | 0.748364 | 1.000000 | 0.643396 | 0.781174 | 0.639133 |
| XMR | 0.820220 | 0.893050 | 0.830954 | 0.781648 | 0.656861 | 0.670166 | 0.643396 | 1.000000 | 0.693810 | 0.889087 |
| XRP | 0.624868 | 0.778367 | 0.678463 | 0.679025 | 0.729910 | 0.854022 | 0.781174 | 0.693810 | 1.000000 | 0.724680 |
| BTC | 0.857917 | 0.906845 | 0.836356 | 0.853071 | 0.732648 | 0.665449 | 0.639133 | 0.889087 | 0.724680 | 1.000000 |

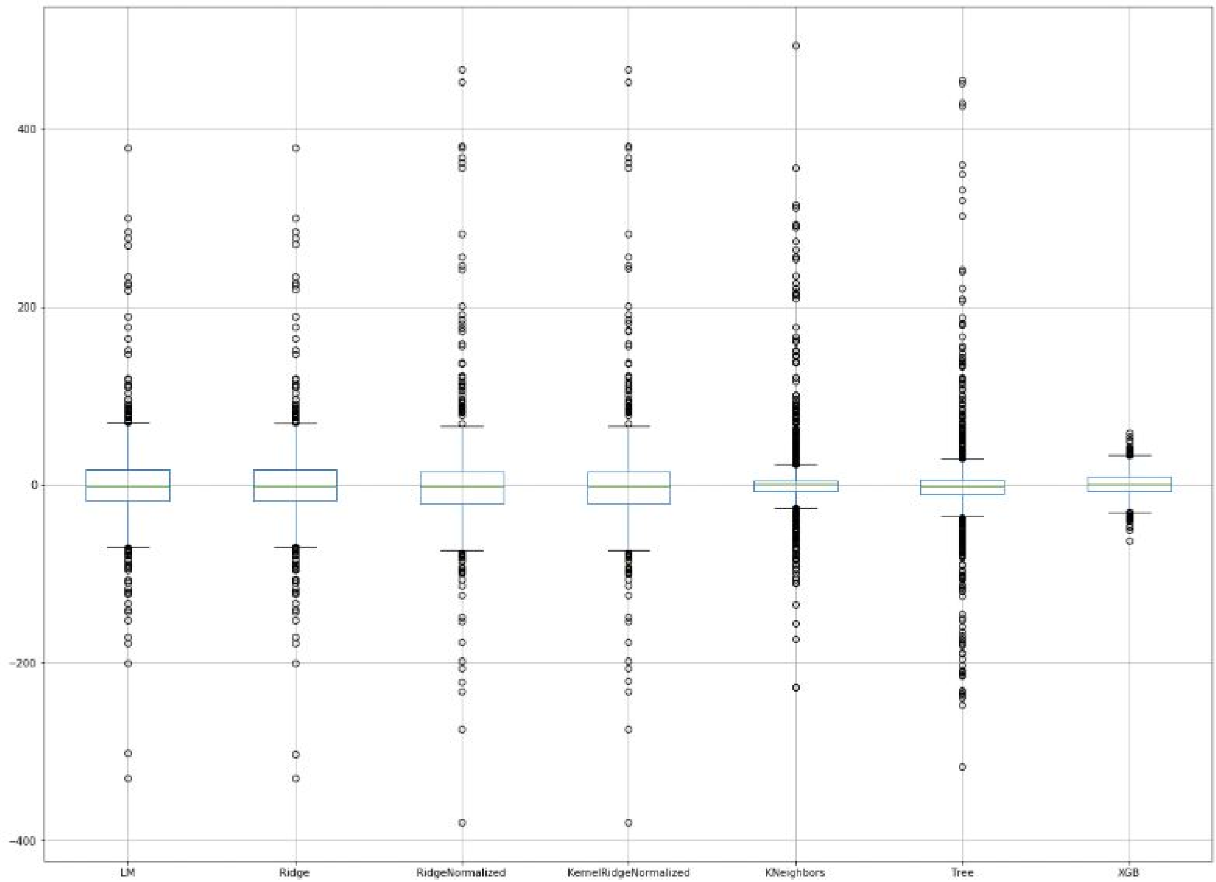


Ethereum (ETH) Prices in USD (August 2015 to April 2018)
1400
1200
1000
800
600
400
200
0

Models vs Actual Price
targetprice
predictedmodel LM
Ridge
RidgeNormalized
KernelRidgeNormalized
KNeighbors
Tree
XGB
900
800
700
600
500
400
2018-03-02
2018-03-09
2018-03-16
2018-03-23
2018-03-30
2018-04-06
2018-04-13

Residual Box Plot by Models




Confusion Matrix - Total Counts of Accurate Prediction of Change in Price for ETH

| | LM_CM | Ridge_CM | RidgeN_CM | RidgeKM_CM | KNeighbors_CM | Tree_CM | XGB_CM |
|---|---|---|---|---|---|---|---|
| **Exception - No change** | 12 | 12 | 12 | 12 | 20 | 884 | 21 |
| **Predicted Decrease / Actual Decrease** | 219 | 219 | 220 | 219 | 220 | 4 | 254 |
| **Predicted Decrease / Actual Increase** | 222 | 222 | 216 | 215 | 217 | 5 | 188 |
| **Predicted Increase / Actual Decrease** | 213 | 213 | 212 | 213 | 209 | 7 | 174 |
| **Predicted Increase / Actual Increase** | 239 | 239 | 245 | 246 | 239 | 5 | 268 |

Confusion Matrix - Total Counts of Accurate Prediction of Change in Price for ETH - 2018

| | LM_CM | Ridge_CM | RidgeN_CM | RidgeKM_CM | KNeighbors_CM | Tree_CM | XGB_CM |
|---|---|---|---|---|---|---|---|
| **Exception - No change** | NaN | NaN | NaN | NaN | 3 | 109 | NaN |
| **Predicted Decrease / Actual Decrease** | 29.0 | 29.0 | 30.0 | 30.0 | 32 | 3 | 45.0 |
| **Predicted Decrease / Actual Increase** | 31.0 | 31.0 | 24.0 | 24.0 | 29 | 1 | 17.0 |
| **Predicted Increase / Actual Decrease** | 27.0 | 27.0 | 26.0 | 26.0 | 22 | 1 | 11.0 |
| **Predicted Increase / Actual Increase** | 29.0 | 29.0 | 36.0 | 36.0 | 30 | 2 | 43.0 |

Confusion Matrix - Total Counts of Accurate Prediction of Change in Price for ETH - 2017

| | LM_CM | Ridge_CM | RidgeN_CM | RidgeKM_CM | KNeighbors_CM | Tree_CM | XGB_CM |
|---|---|---|---|---|---|---|---|
| Exception - No change | 4 | 4 | 4 | 4 | 7 | 351 | 8 |
| Predicted Decrease / Actual Decrease | 87 | 86 | 78 | 78 | 89 | 1 | 105 |
| Predicted Decrease / Actual Increase | 91 | 91 | 89 | 89 | 84 | 4 | 68 |
| Predicted Increase / Actual Decrease | 75 | 76 | 84 | 84 | 72 | 6 | 55 |
| Predicted Increase / Actual Increase | 108 | 108 | 110 | 110 | 113 | 3 | 129 |

From running the hyper-tuning parameters, I have noted that some algorithms are much more resource intensive. Notably the Scikit-SKLearn gradient boosting which takes the most time to run. For that reason I opt out of running the Scikit-sklearn gradient boosting in my second series of runs and instead rely on XGBoost a popular gradient boosting technique which produce similar results and runs more efficiently due to the ability for some of the calculations able to be run in parallel. This trade off of accuracy and speed/resource requirement should always be taken into account based on the type and goal of the model. For the purposes of predicting prices a few days in advance, speed is less of an important issue.

From the observed results with Ethereum, we see similar results as from our initial analysis using bitcoin. XGBoost performs the best. Notably, the regular tree model appear to be unstable which may due to higher volatility observed historically with Ethereum. I would hypothesis that smaller cap coins which would tend to observe worse performance in my models as large price movements can be observed when there is low volume and liquidity from buyers and sellers.

**Conclusion:**

Despite extremely high volatility there does appear to be leading indicators that can be used to help predict whether Bitcoin prices will rise or fall in the future few days. My tree based esemble model, particularly XGBoost performs the best for both Bitcoin and Ethereum. I would hypothesis that my tree models are performing better than for example a SVM based Kernel Ridge model due to there not being a clear underlying relationship or hyperplane identified which would benefit a Kernel Ridge regression. The flexibility of the tree and ensemble modeling seem to benefit XGBoost and Gradient Boosting models. However, I would caution on whether XGBBoost is overfitting in my models as well and a further test out of sample should be done to test viability of my models.

Further enhancements of the models -

- Run the models and predict using the log of the price which would help smooth out the exponential increase observed in 2017
- Run the models using shorter time periods if one believes the rules and patterns observed in the past year is significantly different from historical prices. For example, there were many periods were prices were relatively stable.

**Further Analysis/Ideas:**

- For my models I am only utilizing price history, movement and volume data while there are many other possible external factors at play. Possibly quantifying the amount of “buzz” surrounding specific cryptocurrency using data aggregated from social media (reddit, twitter, and google trends).
- Models need to be further validated on a further out test period to truly determine the predictive power of the models
- Applying some of the analysis and implement into a potential trading bot on an exchange if the edge provided is significant enough