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.

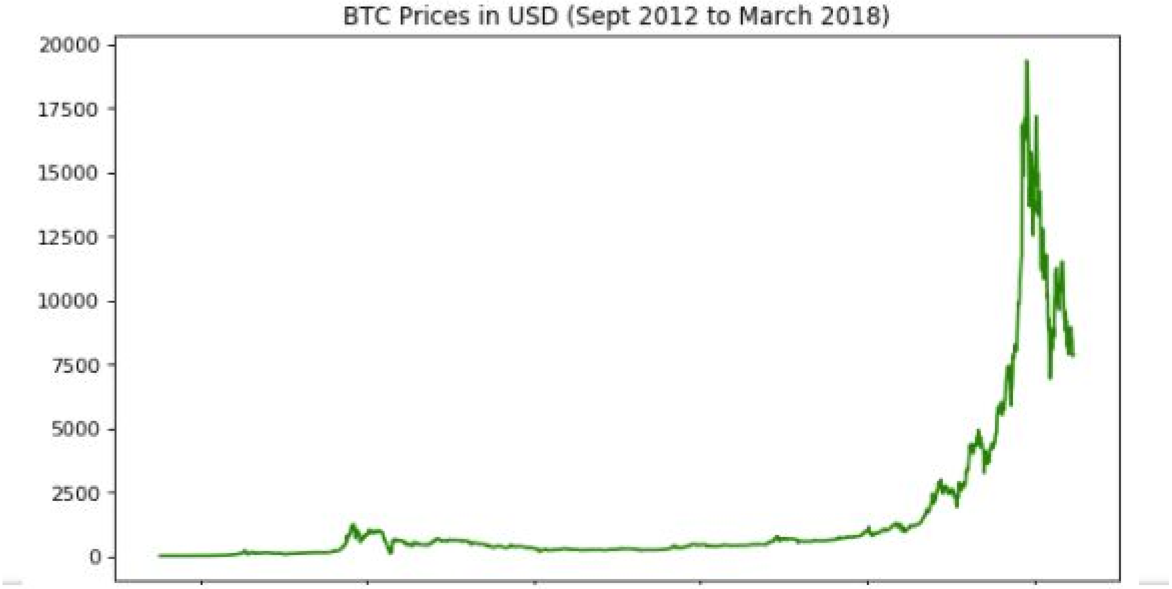
Problem:

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 -

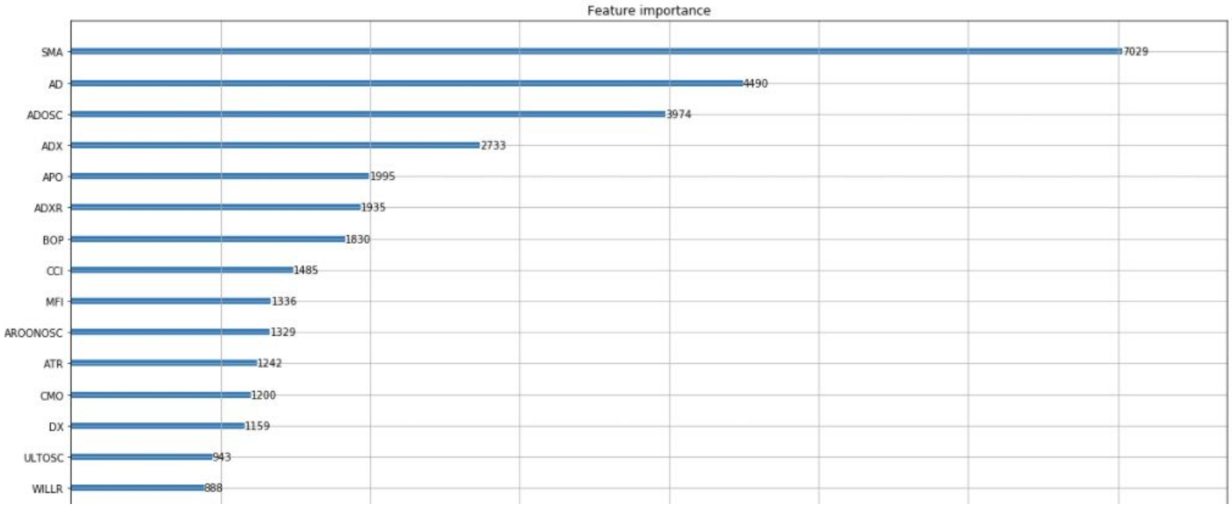
- Linear - basic regression model and we generally observe a smoothing effect in the prediction

- 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.

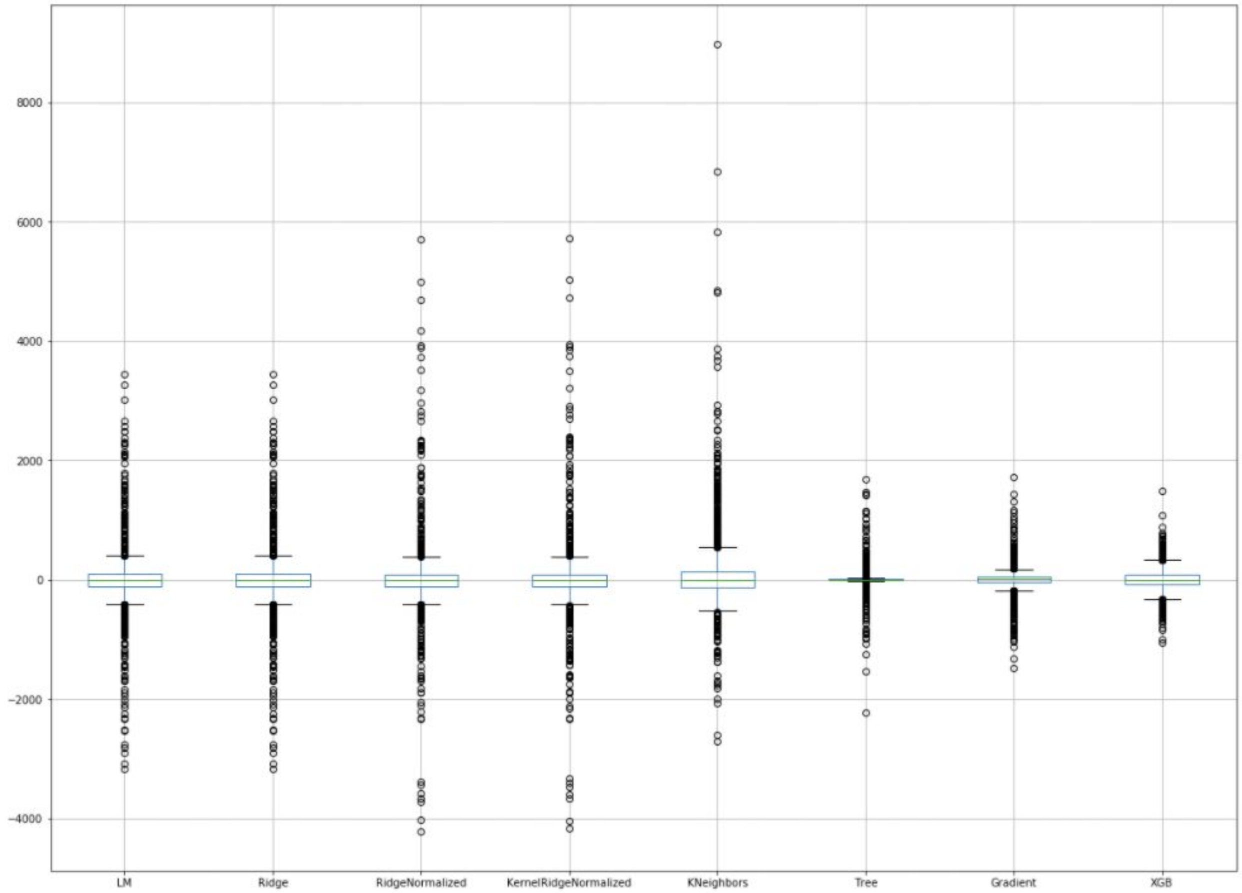
Here is a snapshot of my XGBoost plot_importance graph:



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.

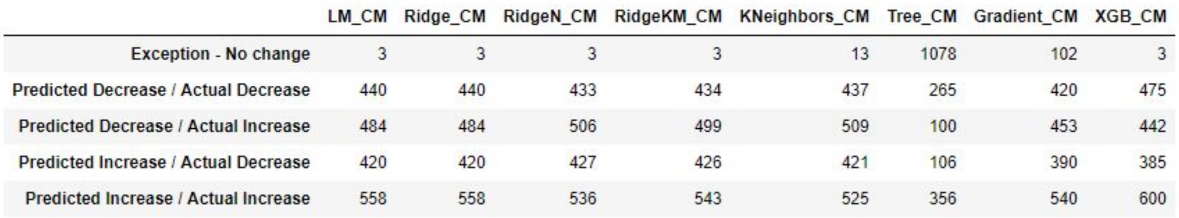
Finally for evaluation of the models, I looked at multiple things -

- 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
    - My models also appear to able to predict both price increase and decreases



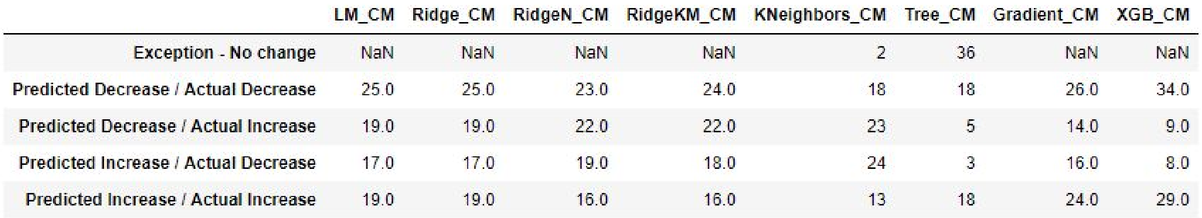
- I also calculated the models RMSE where lower would indicate a better model
- I created pseudo confusion matrixes to get a sense of when the model predicts correctly a change in direction of the price
    - Note this does not take into account the magnitude of the directional change
    - Confusion 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 the trade off being you will be losing out on observation points

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



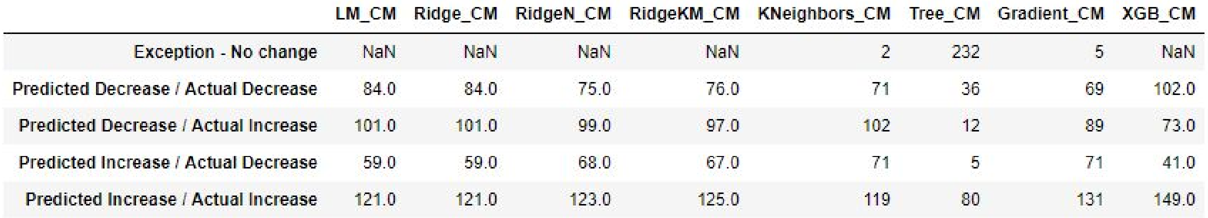
	LM_CM	Ridge_CM	RidgeN_CM	RidgeKM_CM	KNeighbors_CM	Tree_CM	Gradient_CM	XGB_CM
Exception - No change	3	3	3	3	13	1078	102	3
Predicted Decrease / Actual Decrease	440	440	433	434	437	265	420	475
Predicted Decrease / Actual Increase	484	484	506	499	509	100	453	442
Predicted Increase / Actual Decrease	420	420	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

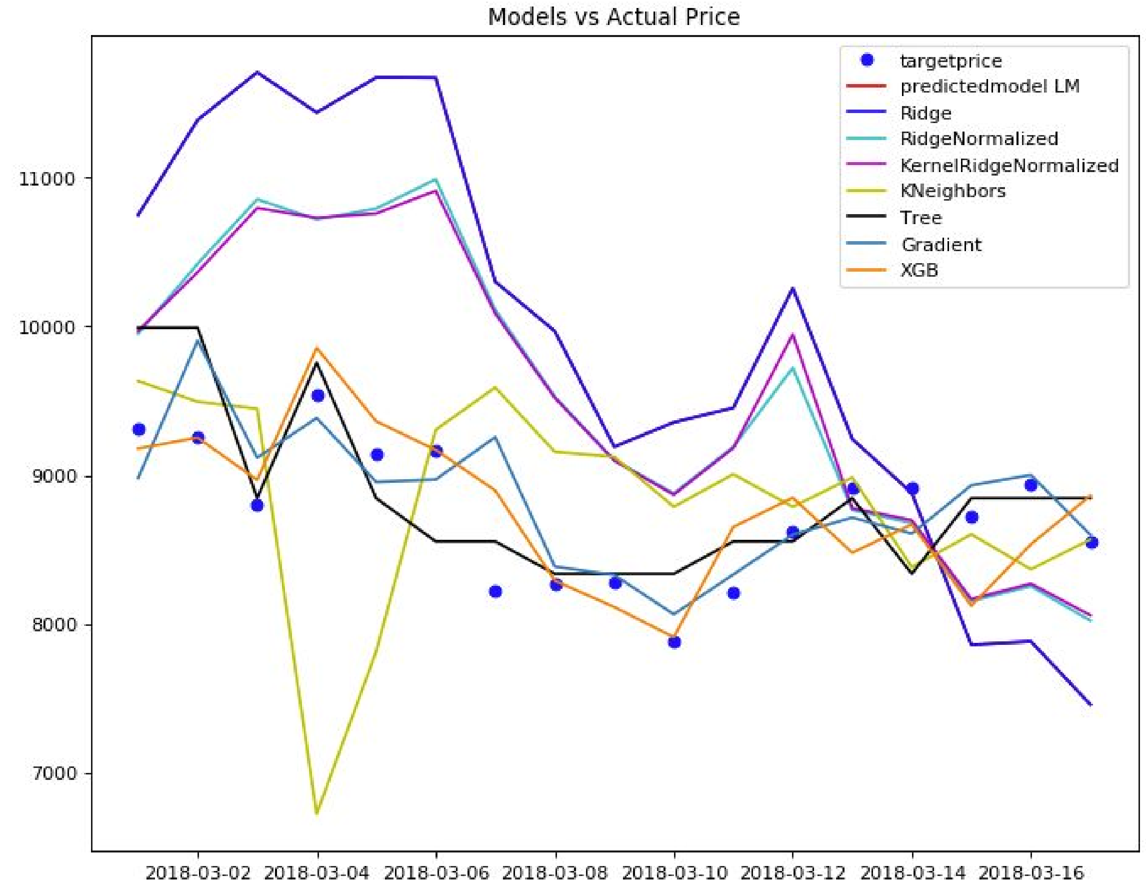


	LM_CM	Ridge_CM	RidgeN_CM	RidgeKM_CM	KNeighbors_CM	Tree_CM	Gradient_CM	XGB_CM
Exception - No change	NaN	NaN	NaN	NaN	2	36	NaN	NaN
Predicted Decrease / Actual Decrease	25.0	25.0	23.0	24.0	18	18	26.0	34.0
Predicted Decrease / Actual Increase	19.0	19.0	22.0	22.0	23	5	14.0	9.0
Predicted Increase / Actual Decrease	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.

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 following future few days. My tree based model, particularly XGBoost performs the best. 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.

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