Milestone 3, CIS700 – Monday @ 8:00pm

Kevin Martin

Syracuse University

The dataset I chose was one involving historical cryptocurrency pricing, with the goal being to successfully predict currency prices given historic trends and available features. I set two research goals for myself: one, determine which features of crytpocurrency trading will best predict future price movements, and can we predict the magnitude of such movements, and two, can we find an optimal combination to successfully predict future price moves.

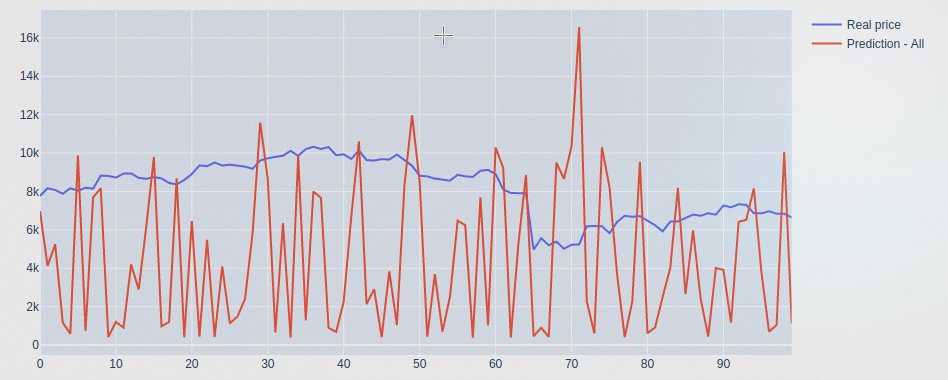
Upon initial observation, the data looked more than sufficient to accomplish these tasks. However, after deeper investigation, that was not necessarily the case. The data had 17 columns (features) to choose from, which seemed ample. But many of these columns were completely useless, and some were redundant. So using a strictly feature based approach was going to be difficult.

Before continuing, the first step was just to understand the data. There are hundreds of available digital coins being tracked for this dataset. While more data is better than less, this could potentially create a lot of noise. So in order to test some of the algorithms, the focus was shifted specifically to Bitcoin. Bitcoin is of course the most popular, widely traded, and widely known of all crytpocurrencies. It is in the news often, for various reasons, and has (as the data also supported) the highest absolute price. That makes the moves more apparent, and easier to study.

Digging further into the data, many of those coins did not even have complete records. At the time I pulled the data, the major coins all had ~1,554 days of records available. This dataset gets updated regularly, so more transactions get added all the time. Many coins had less than 1,000 days available, and would not be suitable for training/testing purposes. In regards to features, as previously mentioned many were unusable. In Milestone 1, I worked to eliminate the unnecessary ones and determine how to get a clean dataset. Originally, I added a binary flag “IsBit” and used classification algorithms to see how well they could determine whether a transaction was or was not in Bitcoin. This proved to be very successful, with most algorithms achieving over 95% accuracy.

Unfortunately, being able to determine the specific coin is not really that useful. One of the features included was price (or price\_usd specifically), and because Bitcoin trades in such higher absolute amounts, I believe this made it very easy for the classifiers to determine that a transaction was in Bitcoin. This was still a useful exercise because it allowed for a clean start on subsequent testing.

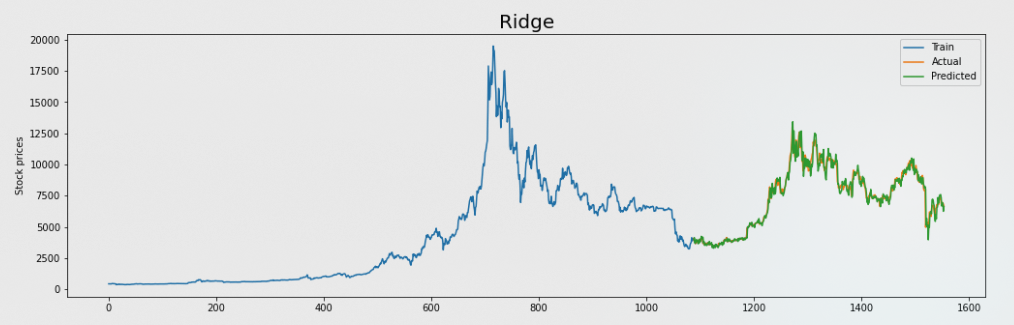
When determining price, I tried two approaches. First, I followed this feature-based approach. In Milestone 2 I used PCA analysis to determine what the top features were, and then went on to attempt to classify the price based on only these attributes. Now the target had shifted from a binary “IsBit” flag to a continuous variable of daily price. The most appropriate algorithm in this case was gradient boosting. There is a packaged version of gradient boosting that I used in several different ways called XGBoost. It includes not only a “pre-tuned” gradient boosting algorithm, but also some anciliary functions that were quite useful. In fact, when running my PCA analysis, the “dmatrix” function allowed for a quick statistical comparison of all features and helped determine the three most useful ones. I used those features and tried to predict price using XGBoost. While successful, the accuracy was not great. See below the best performing approach:



At this point, I believed that my feature-based approach had run its course. My dataset had only three real, usable columns, and predicting daily price is a very tough request for a classification algorithm. I included the results of these tests in the final exam, but did not include them in my Milestone 3 workbook. To get better results, the best option was to use the right tool for the job. In this case, that meant turning the focus to regression algorithms.

I first tried what I thought would be the simplest and least effective algorithms, linear regression and variations thereof. This turned out not be the case as these performed best, but that will be discussed at the end. The linear regression approaches just seem to work for this type of data. I ran a regular linear model, a ridge regression, and a linear support vector regression (SVR).

Ridge regression helps in cases of multicollinearity, where there exists near-linear relationships among the independent variables. This approach is best suited with multiple feature variables exhibiting this phenomenon, however I thought it might be appropriate here as well. By looking at each day-over-day price, I suspected that could create a linear relationship, and thus be subject to this situation. After further testing, I do not believe this to be the case. I believe that multi-variable problems are susesptible, and that pricing is not. The ridge regression did perform slightly better than the standard linear regression, and I believe that is due to simply standardizing the data a bit. Ridge regression works by first subtracting the mean of each variable and dividing by its standard deviation. In the case of some very volatile price movements, this had a positive effect on the predictive prowess. Observe how much better it does than the feature-based approach:

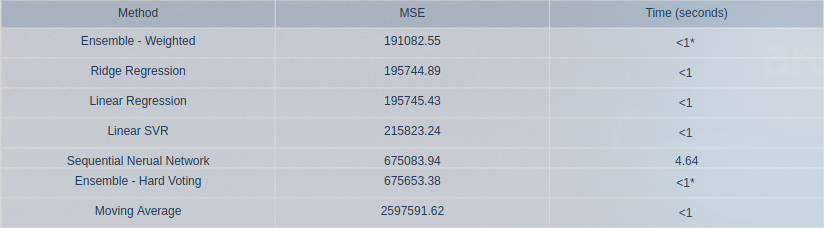


I was most excited to try the third algorithm in this class, linear SVR. Basically, it applies linear regression but also with the familiar support planes found in a regular support vector machine. I had never heard of this method before, but it appeared to be suitable for this type of dataset. Unfortunately, or at least dissapointingly, it did not beat the ridge regression on Bitcoin pricing. My thoughts are while this a novel approach, the standardization mentioned above most likely had a stronger effect on the large price swings.

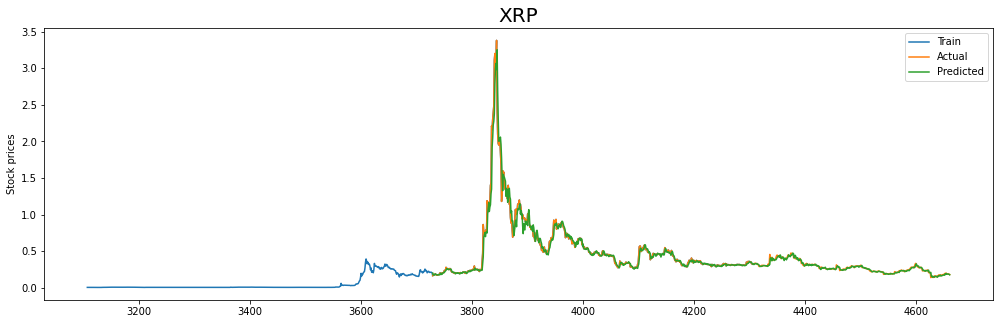
The second approach I tried was discussed in Milestone 2, and really it came down to a neural network. I thought that would have been the most sophisticated and best performer of all, so that was were I wanted to drive the research. I tested a simple moving average, because this type of momentum approach is actually common in the stock pricing world. Finance professionals will cite the relationship between moving averages of two different stocks, or a single stock and two different periods, in order to make predictions. In looking at the other available options, this seems like a wildly misguided approach. It lacks sophistication and any real predictive power, as evidenced by the graph and MSE.

For a neural network, I also chose one that I thought would be appropriate for this dataset, the sequential nerual net (SNN). Again, the data and target are sequential in nature, and would benefit from this type of approach. I was also able to compile it by focusing on both MSE and MAE, which are the metrics I have been looking at with other regression techniques. The results were also dissapointing here, where it underperformed all three linear regression techniques by a substantial margin. Here I believe that more hyperparameters could be tuned to optimize, or perhaps even a different algorithm altogether. I am still of the belief that a deep neural network should outperform simple regression, especially on a dataset like this one.

Finally, I employed two ensemble techniques to try and improve things. I chose both hard voting and weighted approaches. The hard voting approach basically takes the mode from each model described above, for every single day of prediction, and records the results in an array. Then it graphs the array and checks the MSE. For the weighted approach, I gave arbirtrary weights to all the models included, with extra weight to the better performing models. Of the two approaches, the weighted ensemble performed better, and in fact almost as good as the best performing ridge regression. I believe it was dragged down by some of the poor choices from the less optimal models. Here is a quick summary of the models disucssed:



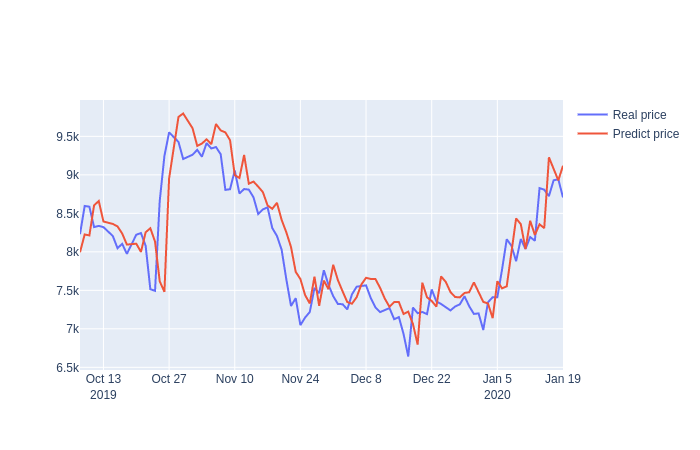
Those were really the goals I set out to accomplish with this project, but I felt there was still more to investigate. Given that the dataset had so many coins, I wanted to see how well I could predict the price of ones other than Bitcoin. I created a new driver function and passed my best performing model, ridge regression, up to it with a coin of my choosing. I tested the top few coins (in terms of trading days available), and results were very good. They were, in fact, much better than Bitcoin. Even the most popular of cryptocurrencies pale in comparison to how robust Bitcoin is, so their prices are so much smaller. When looking at the volatility, they are subject to large swings, but it is all in denomentations far less than Bitcoin. As such, the models were able to achieve statistic superiority on these lesser coins. To be quite honest, I did not expect this at all. I thought the percentage changes would throw off the predictive ability, but that did not occur. See below an excellent example, one called XRP, which had just as many records as Bitcoin but traded for less than $4 during the entire relevant period:



The MSE here was 0.01, several orders of magnitude “better” than Bitcoin. Also, just for exposition, I shortened the training days and the models were still far superior.

In closing, I achieved the two goals I set out do. I found the most relevant features, and was able to generate accurate models to help predict not only the price of Bitcoin, but other coins as well. I was able to explore numerous different techniques, weigh the pros and cons, and arrive and a much more educated explanation about what is going on. I would like to end with one final comparison, between myself and the other kernel on Kaggle. The author used a completely different approach, training models using the Long Short-Term Memory (LSTM) method. I have read about this previously, and how it is useful for stock/asset pricing, but wanted to go a different route. In the end, the author’s best MAE (his metric of choice) was 273, and my best was 306. I am extremely pleased with how close I got, especially considering that the data has moved since the author ran his approach. It is possible the data was better or worse in that prior period, but regardless I think this speaks strongly to both of our work. See below the other kerne’s last 100 days of pricing, as compared to my last 100 days of pricing:

Kaggle:



Milestone 3:

