Cryptocurrency Pairs Trading Final Project Report

Authors: Abhishek Sriram, Adam Slivinsky, Michael Collins, Phillip Jepkes

9 June 2022

Executive summary

The topic we chose for our project was a form of statistical arbitrage, in particular Pairs Trading applied to the cryptocurrency market.

The main portion of this project involves standard statistical methods to identify pairs of historically related cryptocurrencies and a construction of a neural network to exploit market differences in the price of two paired cryptocurrencies.

The bulk of our project was collecting data from the internet, developing a Python script to identify pairs of cryptocurrencies, and using a neural netowrk to predict future prices in order to exploit arbitrage opportunities. The neural network used historical data to train, in order to estimate exactly when to enter and exit trading positions using the cryptocurrency pairs.

In order to accomplish this task we had to build a web scraper using the BeautifulSoup Python library in order to allow for future conduction of trades. We also gathered historical data to select pairs and train the neural network. This historical data was from cryptodata-download.com. The historical data was 50 crypto coins over a 5 month period. This data was then fed into the python script which conducted statistical analysis to identify pairs that possessed similar price time-series behavior in the market.

For us to arrive that at these pairs, we had to use several python libraries. The libraries we picked in Python allowed us to use the clustering algorithm called OPTICS, perform a cointegration test and view the characteristic of the Hurst Exponent. We first use OPTICS to group cryptocurrency data. We then used the cointegration test on pairs within the clusters, to see if there exists a linear combination of the two cryptocurrency price time-series that is stationary. After this process was accomplished, we then used the HURST exponent, to evaluate if the spread, or in our naive appoach, the difference between the elements of the pair of two cointegrated cryptocurrencies, had a stationary time-series that reverted back to it's mean.

After the Python program ran successfully, we attained three pairs that we analyzed. We identified the cryptocurrency pairs of Celer and Doge, Compound and Solana, and Shiba and TRX. We then trained a neural network to predict when to enter and exit the market. Here the neural network we used was a Long Short-term Memory(LSTM) layer followed by a fully connected layer.

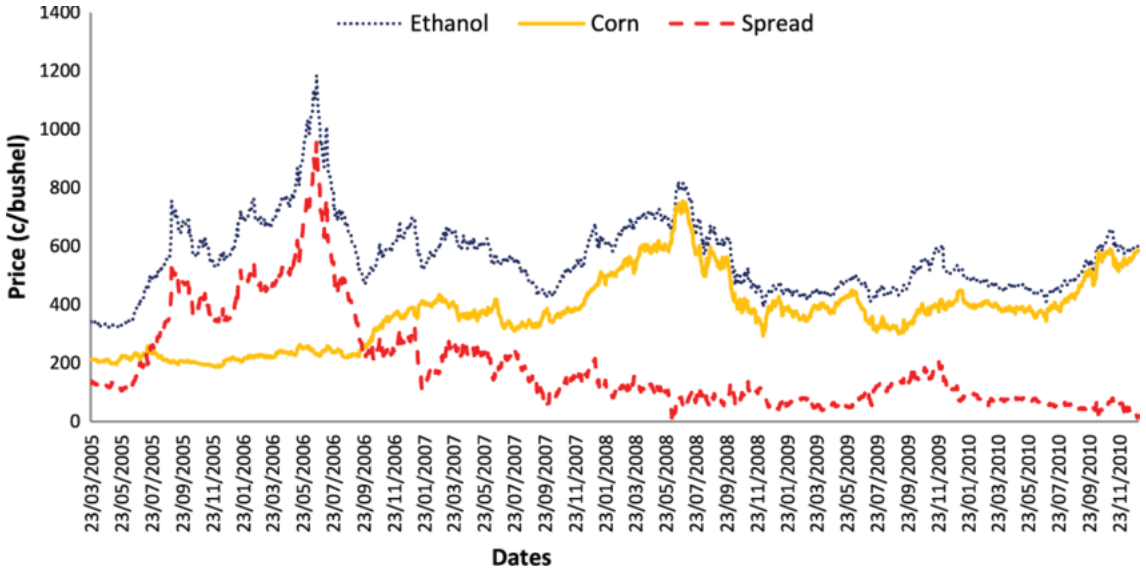
With the time that we had, the neural network was not able to successfully predict future pricing of paired cryptocurrencies effectively to net a profit from given market arbitrage opportunities.

We successfully completed our goal of identifying pairs, and we attempted to complete the goal of using the neural network to identify and make profitable trades. We believe more time and resources would have allowed for a better final outcome for the project. We also believe the substantial amount of programs that were completed set us up for future success when continuing to develop this project.

Introduction and Problem description

The problem that we chose to tackle is to conduct a well known method of statistical arbitrage known as Pairs Trading in the cryptocurrency market. This method of trading involves looking at historically related asset pairs who have desired attributes determined by a variety of statistical tests. These attributes are exploited by taking advantageous positions on each component of the pair. Going long on the undervalued component and short on the overvalued component.

The following is a Graphical representation of a pair in another market.



Our goal is to find pairs like Ethanol and Corn and enter long and short trade positions based on it's spread. We hope to find an ample amount of cryptocurrency pairs to create an investment portfolio.

In order to find long and short positions the pair has to have a set of desirable characteristics. We need the relationship between the pair to be a stationary time series. We need the time series of the spread to be mean reverting at a rate that allows for execution of trades. Also since the Cryptocurrency market contains coins valued below a dollar we need the coin's value in dollars to be around at least one cent to be able to execute trades while handling fees.

In order to find the desirable characteristics the following statistical methods will be used: OPTICS clustering, Cointegration and the Hurst Exponent.(Note PCA was intended to be a statistical method to be used but was omitted during the course of the project.) In order to create entry and exit points of trades and long/short positions we will use a neural network.

Historical data will be used to train the neural network. And more recent and further data collection from a web-scraper would allow for execution of trades.

Data description

We collected historical open prices on Binance of 50 popular cryptocurrencies over a 5 month period, from 2022-01-19 to 2022-05-22. This data was used to find pairs and train a simple LSTM Neural Network to predict the price of the coins. We also collected a sparse dataset of scraped prices from Binance, however, since we have missing days in the dataset and the overall time period that it was collected on was short, only being a two week period, we couldn't use this data to find additional pairs or make our predictive model more accurate. The scraper that was built with Python, however, would be a necessity in order to conduct trades.

Solution description

Selecting Pairs

After collecting the cryptocurrency price time-series data, and cleaning it for our data analysis purposes, we go about identifying pairs of cryptocurrencies that are related in price.
The process of identifying pairs for time-series has many different approaches, the most popular approaches use a combination of clustering algorithms, using distance, cointegration tests, and correlation tests.
For the purposes of selecting pairs for cryptocurrency price time-series', we chose to use the OPTICS, statsmodels, and hurst Python libraries for our clustering algorithm, a cointegration test, and a Hurst exponent test to assess mean-reversion.
Two cryptocurrencies are identified as a pair if they are in the same cluster after OPTICS clustering, if their two time-series price data can be cointegrated, and if the Hurst exponent of the time-series price spread reveals a mean-reverting character.

OPTICS

The OPTICS library clusters the cryptocurrency price time-series data into a set of connected components using Euclidean distance. This clustering algorithm is different than others because it generates an augmented cluster-ordering, ordering the individual cryptocurrencies in each cluster by two metrics: the distance of each point to the center of each cluster, and the distance of each point to all other points in the cluster. This clustering algorithm is more useful than others due to the amount of noise in our data set.
The Python code used to generate the clusters is briefly shown below:

```
X = coins[['date_delta','open']]

clusters = OPTICS().fit_predict(X)

coins['clusterLabel'] = clusters
```

Cointegration

The test to see if two cryptocurrency price time-series are cointegrated checks to see if there exists a linear combination of the two such that the resulting time series is stationary. Our source outlines the approach below:

Formally, considering two time series, y_t and x_t which are both I(1), cointegration implies there exist coefficients, μ and β such that

$$y_t - \beta x_t = u_t + \mu, \tag{2.7}$$

where u_t is a stationary series.

We implement this test by taking each cryptocurrency in a cluster, comparing its price time-series to other cryptocurrency price time-series, and seeing if such a cointegration exists. Our code is shown below:

```
for ele in clusteredCoins:
        #print(ele)
        for i in range(len(ele)-1):
                for coin2 in ele[i+1:]:
                        co1 = coins[coins['symbol'] == ele[i]]
                        co2 = coins[coins['symbol'] == coin2]
                        #print(co1)
                        #print(co2)
                        #print()
                        coint = stats.coint(co1['open'], co2['open'])

                        if coint[1] < 0.05:
                                cointegrated.append((ele[i],coin2))


co_pairs = set(cointegrated)
coH_pairs = []
print(co_pairs)
```

Hurst Exponent

The Hurst Exponent test is a numerical test to evaluate if the spread, or difference, of a cointegrated pair of cryptocurrency price time-series is mean-reverting. We have already verified by cointegration that a stationary time-series exists, and this is an additional test to narrow down the set of selected pairs into a stronger set of pairs. The Hurst Exponent checks whether the time-series' speed of diffusion from its initial value is slower than that of

a geometric random Hurst walk. Our source outlines the equation approach below:

> Formally, the speed of diffusion can be characterized by the variance as
>
> $$\mathrm{Var}(\tau) = \left\langle |z(t+\tau) - z(t)|^2 \right\rangle, \tag{2.2}$$
>
> where $z(t)$ is the logarithmic time series value at time t, τ is an arbitrary time lag and $\langle \cdot \rangle$ is the average across time.

For our purposes with price time-series', the Hurst Exponent is assumed to be 0.5. This means that if the calculated speed of diffusion is less than the 0.5 value of the Hurst Exponent, then the time-series of a cointegrated pair is said to be stationary or mean-reverting. We implemented the test with Python code below:

```
for pair in co_pairs:
        #print(coins[coins['symbol'] == pair[0]]['open'].to_numpy())
        #print(coins[coins['symbol'] == pair[1]]['open'].to_numpy())
        spread = np.subtract(coins[coins['symbol'] == pair[0]]['open'].to_numpy(), coins[coins['symbol'] == pair[1]]['open'].to_numpy())
        spread = np.absolute(spread)
        spread[spread==0] = np.finfo(np.float64).eps
        #print(spread)
        H, c, data = compute_Hc(spread, kind='price', simplified=True)
        if H < 0.5:
                coH_pairs.append(pair)

return coH_pairs
```

Neural Network

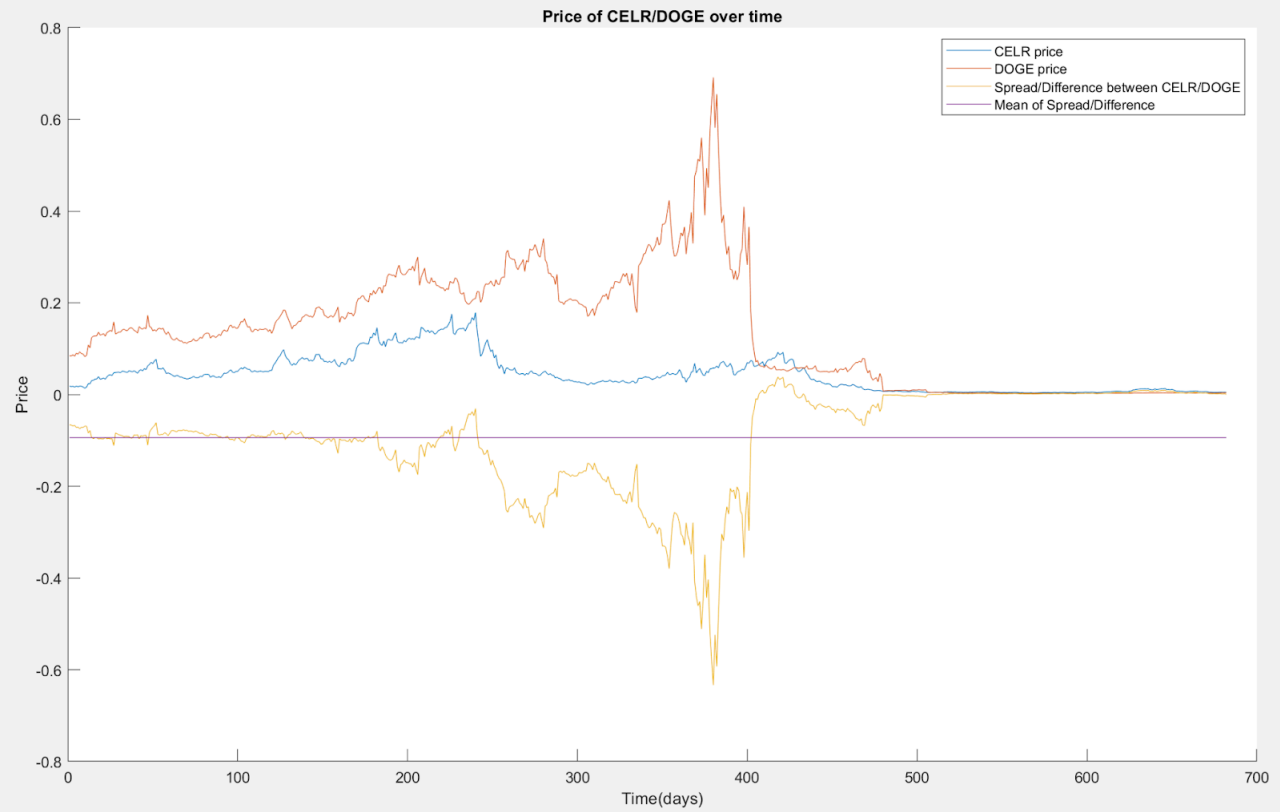
For our Neural Network we used a Long Short-term Memory(LSTM) layer followed by a fully connected layer. LSTM layers are especially good for time series predictions because they are a form of recurrent neural network meaning they use information from previous time steps to influence predictions of later time steps. LSTM layers also implement a type of memory gate that conditionally stores and forgets data throughout the training process, allowing for information learned early on into the process to be retained and not lost to a shrinking gradient. The goal of this network is to have a predictive model that can give insight into the near future of the market and inform when to open positions on a pair. Ideally, we create a probability distribution using the normalized difference between the predicted price and the current observed price to calculate price thresholds for when to open long or short positions.

Result analysis

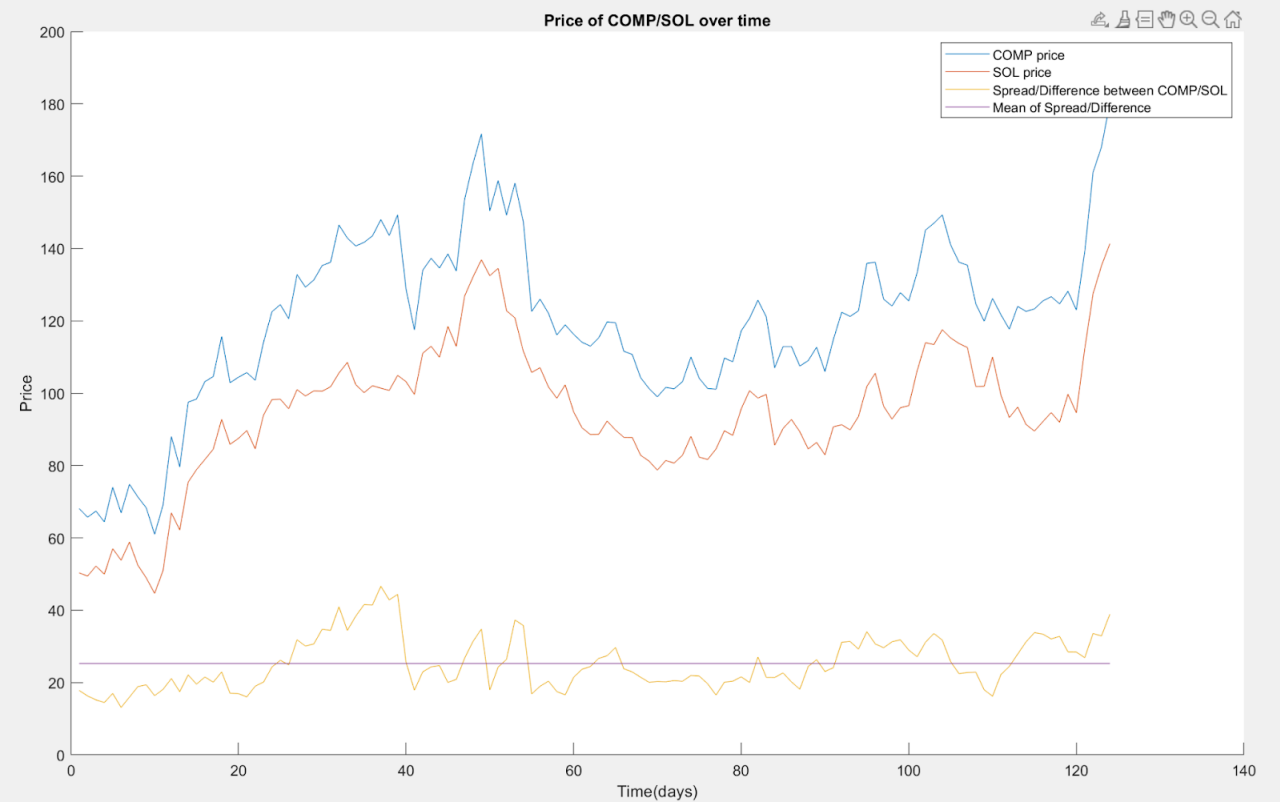
It should be taken into account in the following graphs that spread is calculated naively as the difference between the price time-series of the cryptocurrencies.

The first pair selected below in the following graph Celer and Doge coin. We were quite surprised that this pair was found because of the huge variance in the price of Doge for a

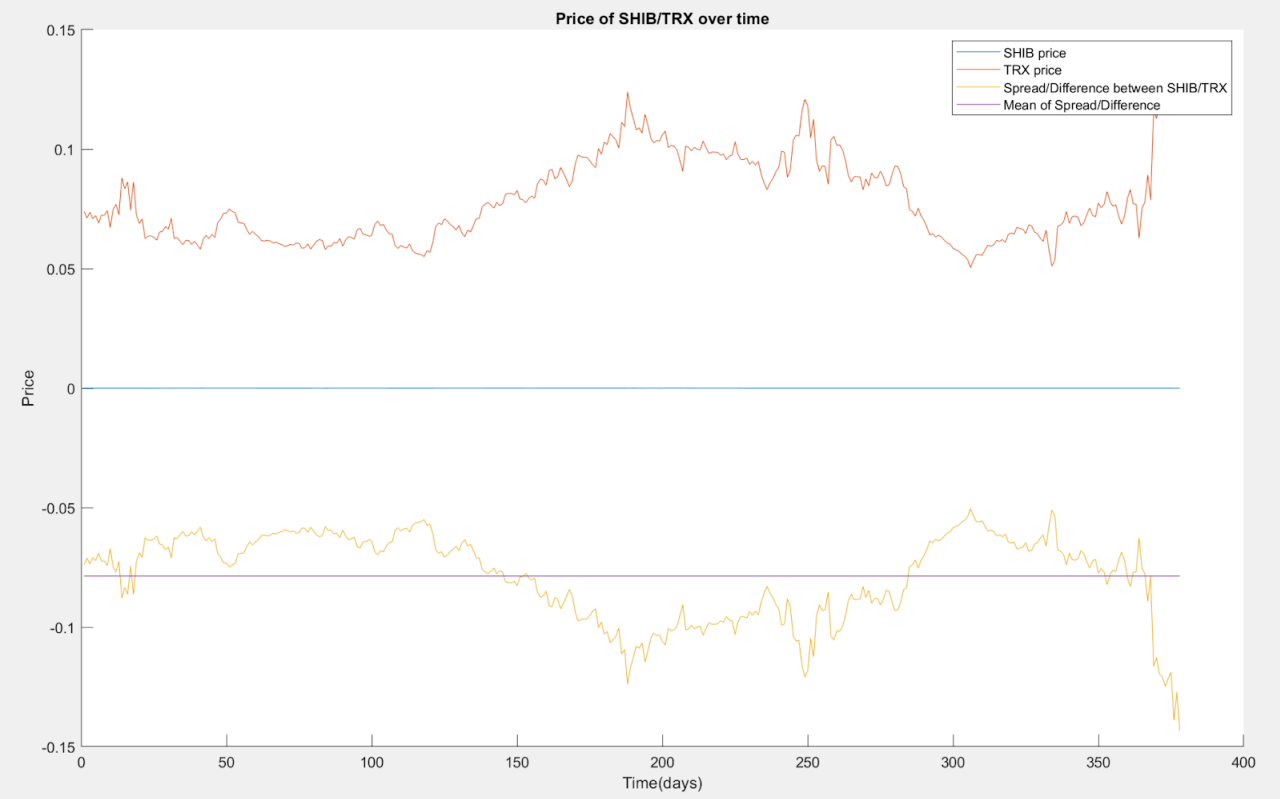
good duration of time. The pair we believe is at decent value. They currently hover around 1 cent and 8 cents respectively. We see decent movement of the spread around the mean that would give ample ability to execute trades. We would hope our neural network would be able to execute trades effectively during the time period Doge was overvalued. This pair would have to be traded at high volume.



The next pair we believe to be our strongest that we found: Compound and Solana. We see the spread move around the mean in a fashion that is desirable, as it visually looks very sinusoidal. The values of the coins currently sitting around 40 and 50 dollars make this the most effective pair to trade.

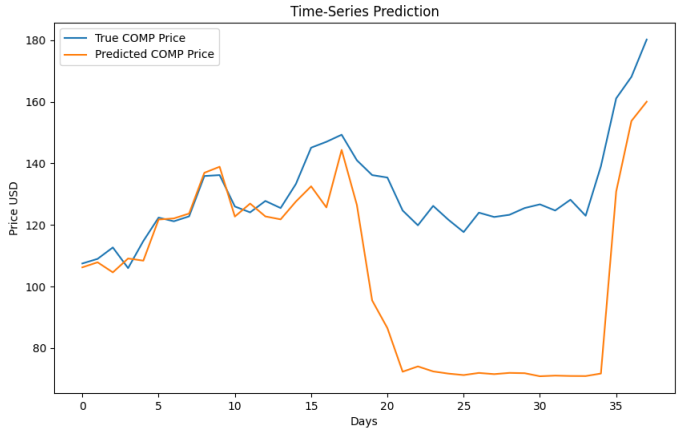


The next pair we found is Shiba coin and TRX. This is unfortunately not a real pair Shiba coin is fractions of a cent and as seen in the graph visually is a straight line. We were surprised that this pair was clustered together. We think Shiba coin happened to be in the epsilon distance from TRX and all the prices were basically stacked on top of each other. We also believe it passed the stationary tests because TRX is basically itself a stationary time series.

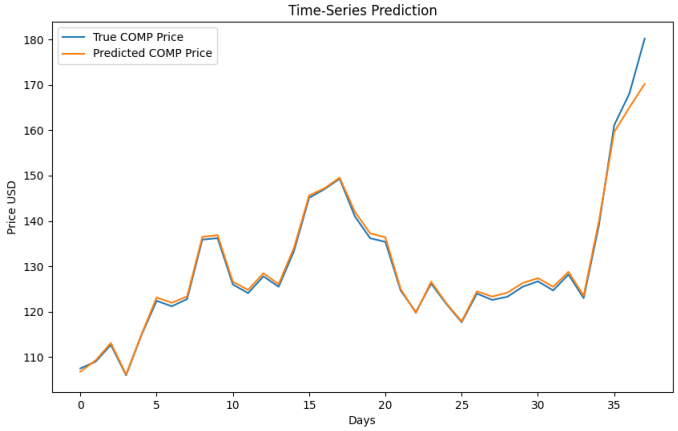


Validation analysis

The following graph is the neural network trained on the entire coin data set. It is attempting to predict the price of Compound. We see that it follows the general trend of the actual price but there is a huge dip in a section. This level of over-estimate is undesirable.



We decided to also train the neural network with just the pair.While visually this is appealing this is over fitting. This would not allow us to execute trades in a fashion that we want because we cannot trust our neural network here either.



We can clearly see from the neural network prediction graphs that allowing the current neural network to enter and exit trades would result in a loss. While it is a good sign that the graphs follow the trend we see the over estimates of the dip in the first graph would create points of exiting trades that would result in considerable loss. The graph that is over-fitted while visually appealing is bad for future predictions when given data that is more up to date.

In its current state the model does not work in the form of correctly allowing for profitable positions. We believe extending the project time and further developing the neural network would allow us to further refine and get more accurate results, as well as have the ability to develop trading portfolios with differing levels of risk for investors. We also believe that calculating spread in a non-naive could impact the amount of pairs being selected in a positive manner.

Original goals and goals met

Our goal to find related cryptocurrency pairs that allow for advantageous positions in the crypto market, using real-time and historical cryptocurrency market data was met. Our goal of developing investment portfolios with varying risk tolerance for an investor, and a log of the short/long sells based on pairs-trading was not met.

Original timeline and end result

Our original timeline was:

- Weeks 1 through 4: Develop Project idea and plan, meet and get to know group members and individual strengths.
- Week 5 and 6: Develop HTML Web Scraping tool to collect real-time cryptocurrency market data.
- Week 7 and 8: Develop implementation of cluster analysis on both the previous market

data and real-time market data to find pairs.

- Week 9 and 10: Construct a neural network to determine when and how much to short or sell long on a cryptocurrency pair, as well as entry and exit points.

Our original timeline we had set for ourselves was followed but overly ambitious. By week 5 and 6 we did have a completed web-scraper. By Week 7 and 8 we had conducted statistical tests to identify pairs but had omitted PCA as one of the statistical methods we were going to use. By Week 9 and 10 we did have a neural network in place but it did not have the predictive power to accurately conduct trades.

At the end though we do have programs that work and that can be re-used and refined. We have a Html- Web scraper that is an utomost necessity if you wish to conduct any trades. We have a program that selects pairs for us when given a data set. We also have a naive-neural network that can be refined and further tested in order to develop more accurate predictions and allow for the ability to execute trades.

Conclusions

We learned a lot about crypto markets and aspects of pairs trading and the statistical methods needed in order to conduct it over the course of the project. However we were not successful in our goal of conducting trades and having a neural network able to execute profitable trades. We think that the pairs selected and the monumental amount of programs we have created has set us up to further develop the project given more time.

We think due to the time constraint, significant lack of programming experience for the majority of the group members and hectic course loads for the entirety of the group we were not able to achieve some of our goals outlined in goals met. We did, however, accomplish a solid amount of our set goals.

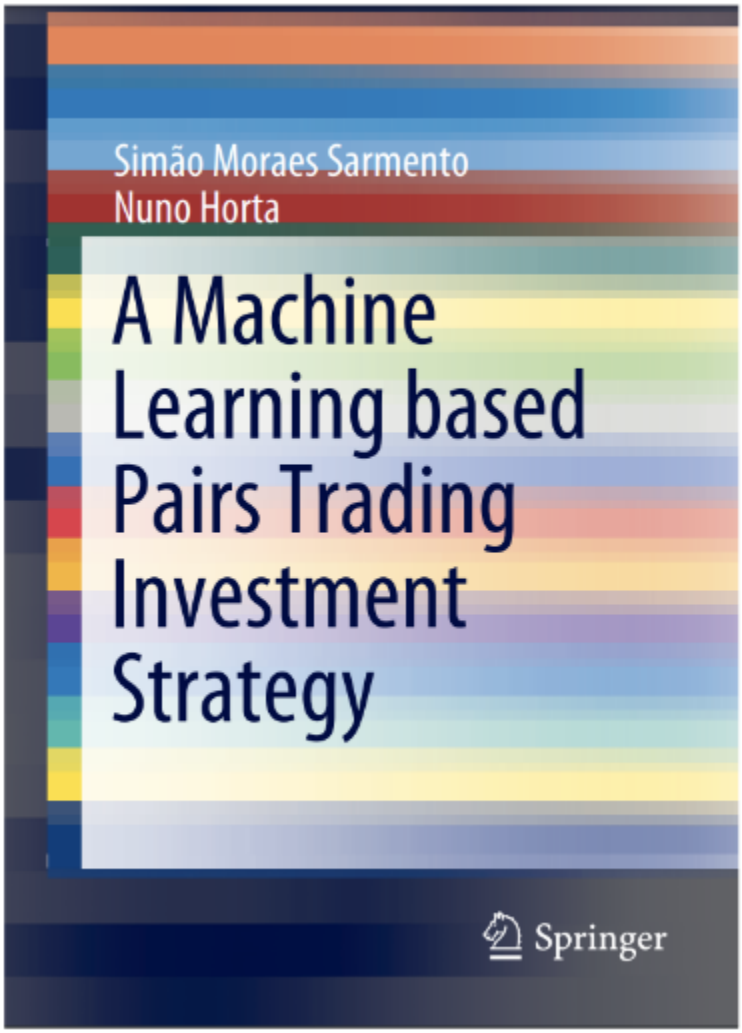
Most importantly we learned valuable information for future iterations of the course.If we were to suggest a project similar to this in the future for other students we would say make sure to have a light course load and make sure all group members have significant amount of programming experience. We would also say use stocks instead of cryptocurrencies. The added benefit of easy access to data stock markets have would allow for better training of the neural network and eliminate much of the work of data collection. Also we think it would result in more pairs being found. This would allow for a larger portfolio of pairs to be created.

We developed significant group bounds throughout the quarter and learned a lot of ideal strategies for when working in groups.We believe this skill set is immeasurable and our greatest accomplishment.

We are proud with what we achieved given the time frame given our levels of experience and we plan to refine the project in the future.

Resources used and References

We used the Python libraries listed above, OPTICS, statsmodels, and hurst. Our main source for the necessary knowledge and approach for selecting pairs and developing the Neural Network was "A Machine Learning based Pairs Trading Investment Strategy", a textbook by Simao Moraes Sarmento and Nuno Horta, published by Springer.



Moreover, we use multiple websites and videos to help us understand the material better and faster. Below is a list of those resources we used (These are all clickable links to the sources):

- Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S and P
- Hurst exponent evaluation and R/S-analysis
- How to Download Historical Market Data
- Pair Selection Framework in Pairs Trading using Unsupervised Machine Learning

- Pairs Trading with Cryptocurrencies
- A Practical Introduction to Web Scraping in Python
- How to scrape cryptocurrency market data for analysis and price tacking?

GitHub Repository:

This is the link to our Git repository that contains all the code to our project