Cryptocurrency Price Prediction and Analysis

Diane Onguetou and Oleg Nikolsky

September 26th, 2021

Introduction

We started with the idea of analyzing cryptocurrencies in the hopes of uncovering interesting insights. Our first idea was to try to predict if a given coin would become successful. On the face of it, this seems like a simple question, but gets complicated quickly. Before we started we were able to meet with Professor Robert Dittmar from the Ross School of Business here at the University of Michigan. He teaches a course, Blockchain and Cryptocurrency Explained, on Coursera. We knew a little about cryptocurrency coming in, but it was interesting to talk to someone much more knowledgeable about it and finance in general. We came away thinking that market capitalization could be an acceptable measure of success. The more we started thinking about this, the more difficult it became, because all of our data was time series. We ended up meeting with Professor Qiaozhu Mei and Yumou Wei to point us in the right direction, so a big thank you to them! Thank you to Professors Kevyn Collins-Thompson and Eric Gilbert for their discussion too!

In the end we decided to try to do next-day price prediction for the supervised learning part because it was a simpler goal for us to understand. We used two broad types of methods for this, a deep neural network and a regressor. Some were more effective than others, maybe suspiciously so. For the unsupervised learning part of the project, we wanted to see if we could find families of coins whose prices tend to move together. This was accomplished by finding similar price movement through dynamic time warping and clustering, with the result being quite reasonable. We'd still like to return to predicting something like a coin reaching a certain market capitalization threshold as a signal of success when we get more experience.

Data Source

We browsed different cryptocurrency websites with the objective of finding price histories for multiple coins along with their market volume, market capitalization, transactions, coins in wallet, social media data, etc. For the analysis, we ended up selecting Coinmetrics and LunarCrush, which cover over a thousand cryptocurrencies providing real-time, daily & weekly updates, as well as historical data. Coinmetrics archives cryptocurrencies since January 2010, meaning data are available almost from creation for almost all existing coins. LunarCrush makes available the last two years worth of data, mostly having to do with social media.

Both Coinmetrics and LunarCrush provide free APIs that facilitate historical data downloading.

Regarding data downloading, Coinmetrics API required the use of a proprietary library while LunarCrush was directly accessible with Python. Data are essentially time series retrieved in JSON format, then converted into pandas dataframes and saved into csv files provided with this report. The table below summarizes data acquisition; further data preprocessing is discussed directly in supervised learning and unsupervised learning sections.

API Resource	**History**	**Records**	**Format delivery**
Coinmetrics https://coinmetrics.io	Since January 2010	138 features for BTC, ETH, LTC, DOGE ~3000-4000 observation for each	Data were essentially time series, retrieved in JSON format and processed & made available into csv files
LunarCrush https://lunarcrush.com/markets	Two years	58 features for 1012 coins	

Supervised Learning

LSTM Price prediction based on historical price values

As stated earlier, our supervised learning objective is to predict the future price for cryptocurrencies. We predict the daily close price of a given coin at time t, based on previous daily close prices from t-T to t-1 for the coin in question. This is a univariate time series with a training window of size T.

- The Model

One approach is to implement a recurrent neural network using Long Short-term Memory (LSTM) layers to predict future cryptocurrency values based on historical prices. LSTM is actually recognized for such time series forecasting applications. The method is known for maintaining an internal state that keeps track of the data previously seen, thereby providing a convenient way to model long-term dependencies. LSTM requires a 3D tensor input with batch size, timesteps (i.e. T), and input dimension equals 1 in our case since we first focus on a single feature (i.e. price).

Data also requires some preprocessing prior to LSTM application. Preliminary clean up tasks include deleting nan observations and sorting data in ascending order by date; this ensures predictions only use past data to predict the future. Coins are so volatile that it is also safe to normalize price values, which is particularly helpful from LSTM computation perspective. The key with normalization is to fit the scaler to training data only, before transforming the entire dataset. This mimics reality where we don't have a clue about the future. Another important step of data preprocessing is to reorganize the data in a way that a sequence of the values in previous T days is used to predict the value at time t. This is done by recording every T+1 consecutive prices, starting at 0 and incrementing by 1 until the (T+1)th last (i.e. from the end); then a historical array keeps the first T elements and a target vector records the last element from the T+1 sequence under consideration. These are then split for training and testing.

At this stage, we use the *tensorflow* Python library to build a neural network of 3 LSTM and 1 dense layers. We compile the model using a tensorflow "adam" optimizer, recommended in literature for regression tasks, and a mean squared error (MSE) loss function. The last step is to fit the model. We mainly use hyper parameters available in the literature, with light tuning based on observations. The reader may want to refer to the lstm notebook for the model summary.

- Evaluation

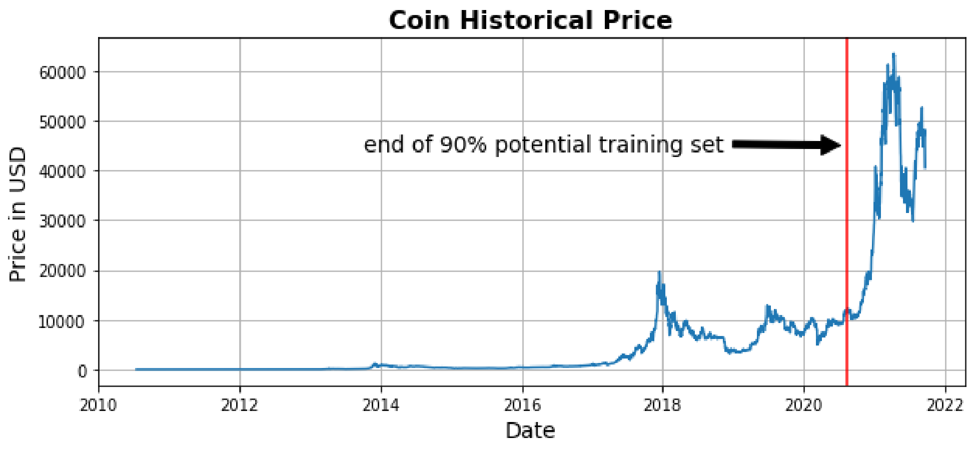

Figure 1 BTC Historical Price with Training Boundary

The above described neural network model is applied to different coins including Bitcoin(BTC), Ethereum(ETH), Litecoin(LTC) and Dogecoin(DOGE). For example, Figure 1 plots BTC dataset and Figure2 provides predictions using different time windows. The model effectiveness is guaranteed by MSE learning curves. The reader may want to look at the notebook to see how our model converges, typically in less than 15 epochs.

In Figure 2, predictions follow the same trend as the actual price values. However, except for the naive prediction model that just assigns t-1 price value to t, the other attempts show a higher mean absolute error (MAE) and give the impression that there is a cap that predictions cannot overtake. MAE was chosen for its simple interpretability, as it has the units of the dependent variable. Even predictions for a 1 day window that could be expected to be similar to the naive prediction screw up during high spikes. In our opinion, this is caused by the nature of the data and the normalization scaler that is set on the basis of training data only, where it is not possible to anticipate the spike observed in mid 2020.

Expanding the training set to encompass this might create other issues because this will represent 96 to 97% of the data; using only 3% of data for validation would certainly make MAE noisy. Another idea to fix the scaler was to add a very big, temporary value when fitting (e.g. $80000); this didn't fix the issue but playing with larger windows that could maintain in memory spikes (even much smaller) from the past until current prediction cases, and other coins where similar peaks had been experienced in training data greatly improved MAE (i.e. approximately $2000 for 200 days) and almost perfect predictions for LTC. The reader would certainly want to refer to the notebook for those improvements.

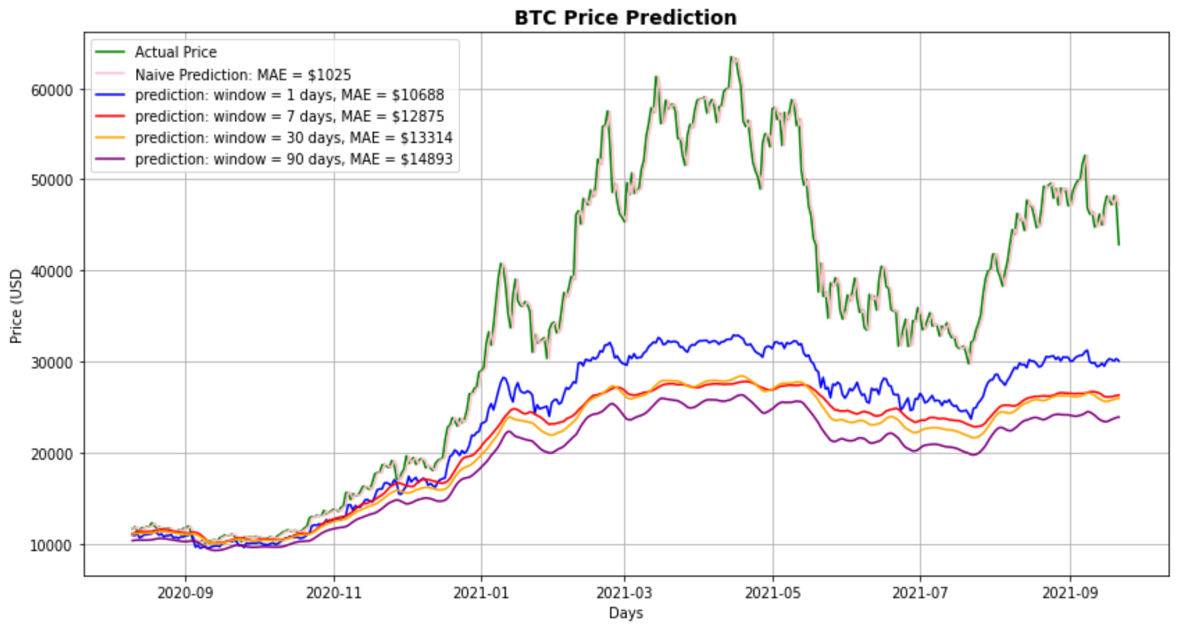

Figure 2 BTC Price Predictions for Different Window Sizes

- Failure analysis

DOGE coin prediction was a complete failure, as shown in Figure 4. The problem here is not our model but in the data; looking at the dataset plot in Figure 3, we see that there is a total break with the past. Nothing similar to the future is observed in the past and sent to the model.

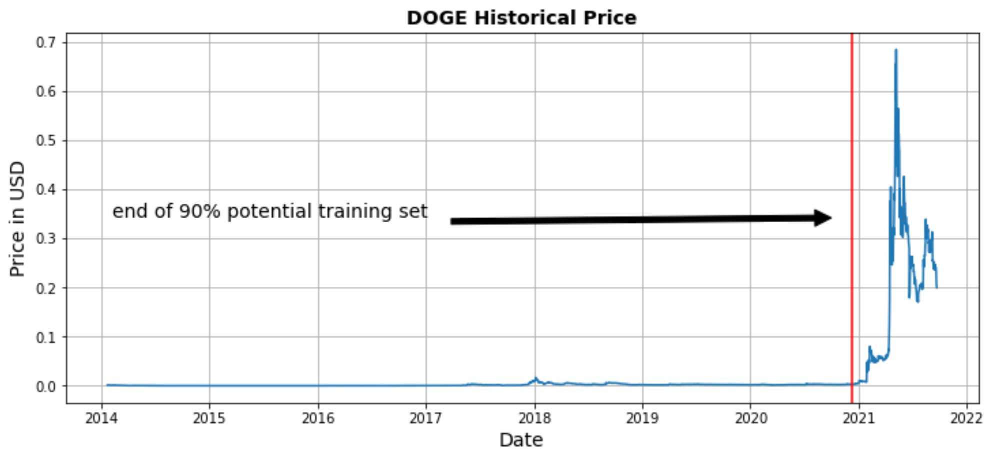

Figure 3 DOGE Historical Price with Training Boundary

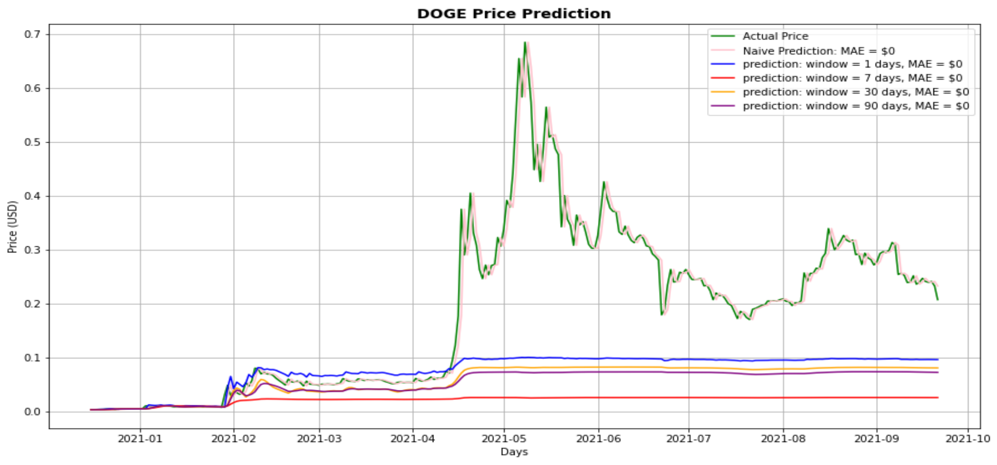

Figure 4 DOGE Prediction Failure

DNN Price prediction based on historical price values

For a second method, we tried a simpler Dense Neural Network with *tensorflow*, with just two and three layers. LSTM is made to work with temporal data, so it should be the better choice, but we were curious about the results we would get with just a deep neural network. We were able to tune this model to give predictions very close to the naive prediction, resulting in a very similar line. This seems suspicious, either due to data leakage or too much reliance on the previous day's price. The first layer had a size of 40, to match the size of the training window, and output a single value in the last layer. Having a smaller first layer gave worse results.

An optimal learning rate was found through a plot of learning rate versus loss. The input is only a time series of the previous forty days of prices, so there should not be any data leakage, but just in case we tried a similar prediction for the second day beyond the window instead of just the next day, achieving similar suspicious results. For one day ahead, the MAE was 1168 vs 1030 for the naive prediction, and for two days ahead, MAE was 1538 vs 1518. After tuning the learning rate, the number of epochs was increased from 100 to 200 for the final run.

An attempt was made to continue predicting from only newly predicted values added onto the end of the training values, but this resulted in a new prediction line that just continued up. Having as much valid price data as possible in the forty day window was very important. Only BTC prices were used for this analysis.

Random Forest Regressor for price prediction based on non-price features

- The Method

In the LSTM method, we deliberately used past price values only to make price predictions. In reality, price value can be affected by many other factors. In this third attempt to predict prices, we used coin features other than price in order to predict the price value of the coin in question. The initial idea came from the Mads Money project (Drumheller, 2021). Here after cleaning up the dataset and splitting into train and set, we manually excluded the features that were too correlated with price such as price itself and market cap numbers. These were filtered out with a combination of looking at very high importance scores and common sense. This was done with the Lunar Crush and Coinmetrics datasets separately and together. Scaling was not found to have made a difference on feature weights. The Coinmetrics features were more important than the social media data from Lunar Crush, which was surprising. This is in contrast to the neural networks where we couldn't interpret features at all.

We tried a few different regressors, such as *AdaBoostRegressor()*, *GradientBoostingRegressor()*, and *ExtraTreesRegressor()*, but found that *RandomForestRegressor()* worked the best. One problem we had was tuning the regressor. A grid search was used to try to find the best parameters to use, but these didn't turn out to work well in the end. We manually tuned *max_depth* and *n_estimators* instead, which took some time.

We next wanted to find a smaller set of features that could be used for the regressor. With the Coinmetrics and Lunar Crush having 138 and 58 features respectively, we took those that were found to have 90% of the importance. This resulted in 39 for Coinmetrics and 13 for Lunar Crush. For the amount of data we had this didn’t make much difference in processing time, but it does make the number more manageable. The number of BTC wallet addresses with high balances were found to be the most important features from Coinmetrics, and social contributors and Tweet sentiment were found to be the most important variables from Lunar Crush.

- Evaluation

We used Mean Absolute Error to evaluate the different models and parameters, and Residuals vs Fit plot evaluate performance. The R^2 values for the regressions were very high, but overfit. This was confirmed by a Residuals vs Fit plot and showed that the distribution of predictions wasn’t random about zero as would be expected for a well fit line. This is seen in the plot below, as the orange predicted values are much further below the actual values. We believe this failure is due to the fact that a RandomForestRegressor cannot extrapolate predictions outside those it has seen.

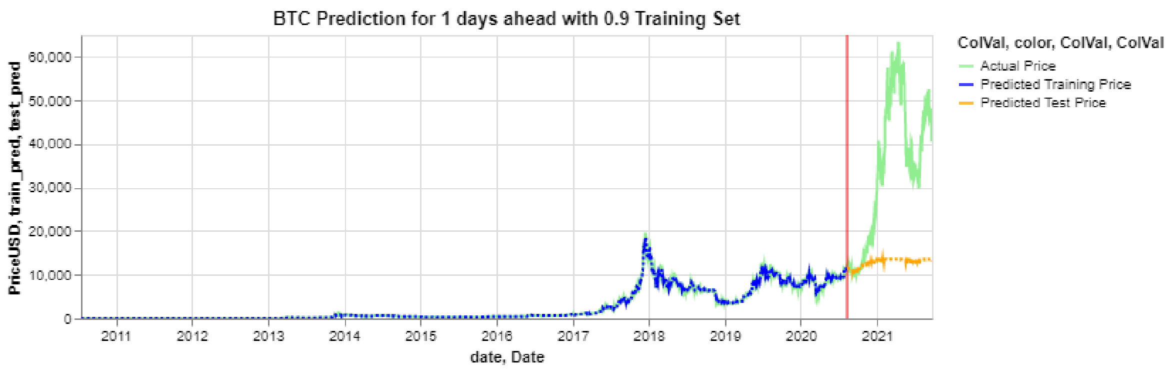

Figure 5 BTC Price Prediction using Random Forest Regressor

Better performance was achieved with smaller training sets. They looked much better in the plot, but left less data to actually validate against, and did allow the model to see much higher price values.

Unsupervised Learning

- Methods

In order to find coins which had similar price movements, we needed to get pricing data. Lunar Crush was the source we selected because even though it provides only two years of historical data, it has data on 1012 coins. After removing those coins which had missing data, we were left with 502 coins. Not much other cleaning was needed. The first problem we needed to solve is about how to find similarity between coins with widely different prices. Bitcoin(BTC) has been as high as $60,000 while other lesser-known coins struggle to reach a penny. A solution to this was to normalize each coin's price to a z-score. (Mueen, 2016) A z-score is found by subtracting the mean of a series from each value, and dividing by the standard deviation. This basically returns how many standard deviations from the mean each element in the series is, and this was calculated for each coin.

The next step was to find the pairwise distances between each time series, for which dynamic time warping(dtw) is created. For a small collection of time series dtw is reasonable to calculate, but with over 500 we had to search for a faster library. We ended up using *pytw* from *cdtw* which accomplished this in a few minutes, where standard libraries still hadn't finished after a couple hours.

The last step was to visualize clusters. A simple but ineffective way to look at the resulting distance matrix using *pcolormesh* from *matplotlib*, but this results in a square image like a confusion matrix, but only slightly organized. It is difficult to see how many families of coins there may be as nearby coins may not be shown together. There are methods for helping visualize these relationships, and one we liked is called Ward's method. It tended to group similar coins together in a more organized way. The second visualization we used was a dendrogram from *scipy*, again favoring Ward's method of the available sorting methods.

- Evaluation

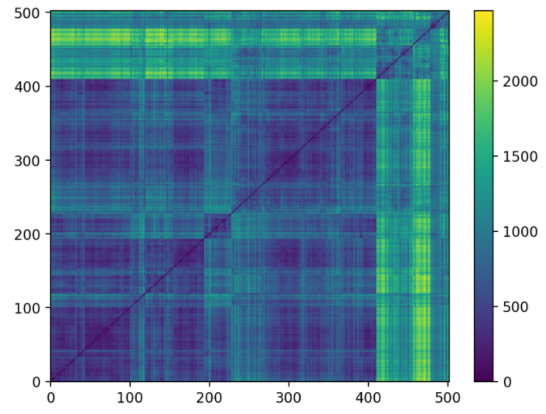

Figure 6 pcolormesh plot for 502 coins

The confusion matrix resulting from Ward's method was pretty confusing, but we were able to see two big families of coins, and then quite a few other smaller families within them. This is shown by the light and dark sets of bars and helps get an idea of what to expect, though we left it unlabeled as it shows a lot of redundant data and would've been huge on the screen.

The dendrogram was much more interesting, though labeling it required it being quite large too. The original two families of coins were visible with a cutoff of 5100. The smaller families were visible below these. We created a plot of cutoff vs number of clusters to see what the trend was, and decided on a cutoff of 2400 as this resulted in 11 clusters, where 2500 left 8.

The other sorting methods weren't as pleasing to look at for the confusion matrix or the dendrogram, as they resulted in uneven cluster sizes at the different cutoff limits we tried.

Given that there are so many different kinds of cryptocurrencies, we were surprised that Ethereum Classic(ETC) and Dogecoin(DOGE) were so related by price movements. Ethereum(ETH), which split from ETC, was not similar to either. BTC, DOGE, ETH, and LTC were all in the larger of the two big families of coins.

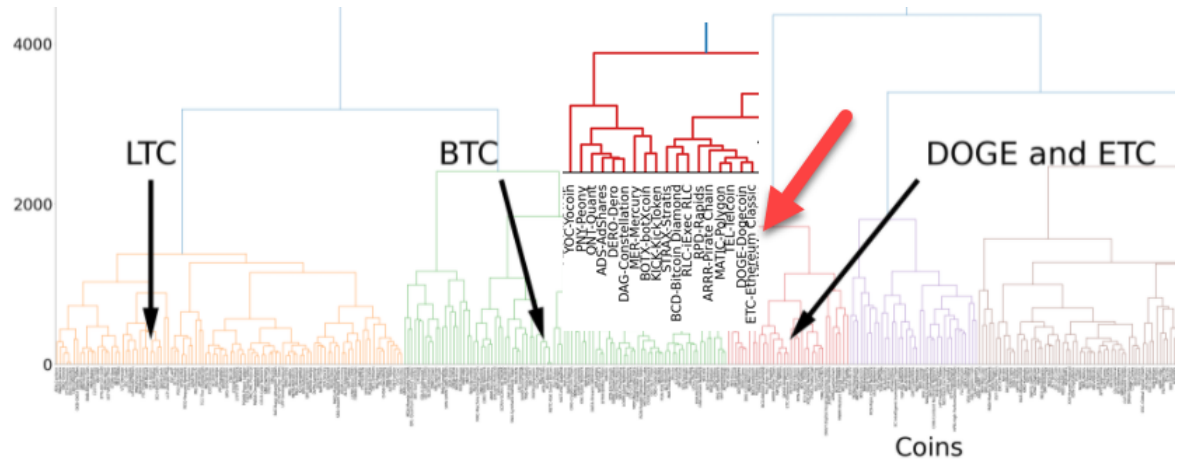

Figure 7 Zoomed-in Portion of the Dendrogram for Coins Clustering

After the dendrogram, we plotted the actual coin prices of those that were found to be similar to confirm this was so.

We had wanted to try clustering each coin by taking into account not only each coin's price, but also data from its associated parallel time series. We weren't able to make a useful table of summary statistics from each of the parallel time series, as there were too many time series to choose from, and we weren't familiar with which summary statistics would be most relevant. This would be an interesting task to complete in the future.

Discussion

This was our first time working with time series and an API, and we were a little overwhelmed when we realized that that's what we were really working with. Each coin had many parallel 'sibling' time series, which made it difficult to see which way to go at first. Talking with professors before starting the project helped us understand what we needed to do, and is a great first step before starting any new project!

This was our first real world experience with machine learning outside class assignments. We had many choices to make when creating our models. We learned a few different approaches to predicting time series. The time window was a new but obvious approach for the deep learning models, and we were surprised that some of them basically predicted the future by using the previous day's value. If we adjusted the percentage used for training data, our accuracy did increase, but at the cost of having less training data to evaluate on. It would be interesting to add the ability to change the training size percentage in the streamlit page as a future improvement.

If we had more time, exploring more about the different features available in tensorflow would have been interesting. For example, it is towards the end that we realized the third input parameter to the LSTM allows for multivariate analysis; this is a perfect opportunity to include multiple features such as market or social media data to the LSTM price prediction and have a better comparison with the regressor. On the other hand, price predictions we made here are for the next day only. In the real world, a more realistic goal is to forecast weeks or even months ahead; the idea behind is to plan more confidently for investments.

We didn’t use any personally identifiable information, so we didn’t run into those kinds of ethical issues, but I could see issues arising about the interpretation of results. People could really rely on them, and how we display and explain them could make a difference. Adding uncertainty bars to any stated predictions would help.

For the unsupervised learning part, we learned about how much it takes to create a distance matrix for so many time series, the kinds of ways this can be represented. Dynamic time warping was a great tool, and in the referenced paper has lots of other applications as well, which would be interesting to explore. One improvement that could be done with the clusters in the dendrogram would be to shade the different coins in each cluster by market capitalization, which would be an easy way to show the leaders in each grouping. This would make it much easier to find the most important coins out of the 502 available.

An ethical issue that could arise would be interpretation of the clusters and giving reasons as to what might be the underlying reasons for this. This would be interesting to examine, but we would have to be careful with only giving reasons in which we have reasonable confidence. There are outside influences on crypto prices such as China making most crypto currencies illegal. Our project doesn’t have an obvious connection to this, but we should remain aware. Taxes and regulations are also an issue as this industry is still very new.

Statement of Work

Oleg: Deepnote collaborative environment, API for data downloading, clustering for unsupervised learning, DNN, random forest regressor, report, streamlit for publishing results

Diane: Coming up with idea of cryptocurrency analysis, LSTM for supervised learning, report, streamlit, finding cryptocurrency Coursera class

Related Resources

Source Code: https://github.com/legolego/MADS695

Further visualizations:
https://share.streamlit.io/legolego/mads695/main/streamlit/app.py

References

Mueen, A., & Keogh, E. (2016, August 11). *Extracting Optimal Performance from Dynamic Time Warping.* University of New Mexico. Retrieved September 2021, from https://www.cs.unm.edu/~mueen/DTW.pdf, Pg. 46

Drumheller, T., Garden, J., & Kessler, J. (2021, August). Mads Money Dashboard. Mads Money. Retrieved September 2021, from madsmoney.io.

Soner Yildirim, (2020, April 15). *Cryptocurrency Prediction with LSTM, How to predict the trend of currency rates*, Retrieved September 2021 from https://towardsdatascience.com/cryptocurrency-prediction-with-lstm-4cc369c43d1b

Mauro Di Pietro, (2020, March 9). *Time Series Forecasting: ARIMA vs LSTM vs Prophet*, Retrieved September 2021, https://medium.com/analytics-vidhya/time-series-forecasting-arima-vs-lstm-vs-prophet-62241c203a3b