# Cryptocurrency Predictive Analysis

Minjun Park[1], Erin Song[1], Hao Xiang[1], Ying Xiong[1]

*Abstract*— **The goal of this project is to predict the nonlinear nature of cryptocurrency price with historical data of four features - open, close, high, and low - using deep learning models. To capture the general trend of the price, we applied sliding window method with various window sizes. In the model selection part, we used a hyperparameter search method to find the best combination of the parameters. In addition to the above, logarithm transformation was also used to provide the log return. Our result shows the model is able to give reasonable predictions for the cryptocurrency high and low prices. In addition, we also discovered that, due to the unpredictable nature of cryptocurrency, the same trained model was unable to make good predictions on the stock market price prediction task.**

## I. Background and Motivation

Time series prediction is a well-known problem and extended research has been done on stock market prediction [3]. Cryptocurrency payment systems such as Bitcoin and Ethereum are gaining a lot of attention and adoption since their first emergence around the year 2009. Bitcoin has a current market capitalization of 9 billion USD and has over 250,000 transactions taking place per day. The decentralization of cryptocurrencies has greatly reduced the level of central control over them, impacting international relations and trade. Cryptocurrency predictive analysis is an interesting time series prediction problem as the market is still in it transient stage.

## II. State of the art methods

### A. *Overview of the existing methods*

Research on cryptocurrency price forecasts is at an early stage. Various methods are adopted to analyze the price behaviors of cryptocurrency; method to predict cryptocurrency price by considering various factors such as market cap, volume, circulating supply, and maximum supply based on deep learning techniques such as the recurrent neural network (RNN) and the long short-term memory (LSTM), which are effective learning models for training data, with the LSTM being better at recognizing longer-term associations.

### B. *Challenges and Limitations*

**High Volatility:** Due to high variance, validating machine learning predictions onto Bitcoin data remains a difficult task. In particular, price behavior of cryptocurrency is oftentimes counter-intuitive, making the price prediction challenging.

**Lack of seasonality:** Classical approaches such as Holt-Winters exponential smoothing for time series prediction problems are dependent on linear assumptions, and these approaches may not be useful for predicting cryptocurrency price since there is no seasonal effect in cryptocurrencies, which are highly volatile in nature.

## III. Broader Scope and Goal

We wish to implement a deep learning model, which has two input layers followed by Long Short-Term Memory(LSTM) layers, a concatenate layer, dropout layer, and two output layers. This model predicts the nonlinear nature of two cryptocurrency price return (Hourly/Daily Bitcoin and Ethereum data). In addition, the following two goals are also listed.

1) Identify useful parameters, such as the window size of sliding window method, that lead to higher prediction accuracy.
2) Investigate if trained cryptocurrency model will make reasonable predictions on the stock market (SP 500).

## IV. Proposed solution and major contributions

Based on the papers we reviewed, the main contributions of this project includes:

1) We use sliding window technique which is an optional technique for time series data that uses previous timestamps to predict the next timestamp.
2) We predict logarithmic return instead of the price. The benefit of using log return is normalization, measuring all variables in comparable metric.
3) We extend the analysis to explore whether the bitcoin predictive model can be used to predict the SP's 500 index, which has a long run negative correlation to bitcoin price [7].
4) We propose a novel approach to our model by passing two separate inputs in pairwise fashion – open and close as one pair, and high and low as another pair – in order to train the model with features with similar price types.

## V. Proposed experiments-datasets, tasks, architectures, expected results

### A. *Issue and hypothesis*

The nonlinear nature of cryptocurrency price makes prediction a difficult problem. By using deep learning techniques and understanding the dataset through exploratory analysis, we can predict the future price with acceptable accuracy measure.

[1]Rice University, Houston TX

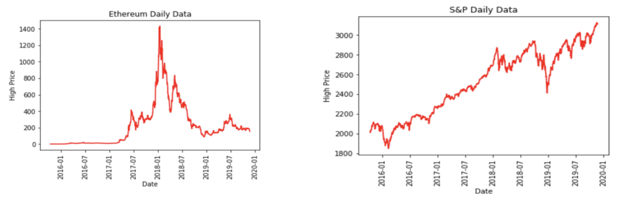

**Fig. 1:** training and testing dataset for Ethereum and stock daily prices

## B. Experimental Setup

*1) Dataset:* As shown in fig. 1, our dataset contains the historical price information (from October 2015 to November 2019) of two top cryptocurrencies: Bitcoin and Ethereum. The dataset is obtained from www.cryptodatadownload.com and originated from coinmarketcap, Blockchain info and Etherscan [5]. For each type of cryptocurrency, we have both and the hourly and daily data (the files are stored in csv format); for stock data, we have daily information. In addition, we also have the following information for a given day: The opening price, The highest price, The lowest price, The closing price, and Market capitalization in USD.

*2) Data Exploratory Analysis:* For this part of the experiment, we began by doing some literature review of similar topics. Since price data can be stored either in daily, hourly or minute, choosing suitable data formats is necessary to start. After some research, we decided to use both daily as well as hourly data for the cryptocurrencies based on the research papers we read [1]. For the data preprocessing steps, we began by checking for null values and removing them if they are present. In addition, we implemented the min/max scaling method. Normalizing the data is important since it evens the learning weights and help prevent the network from focusing only on the greatest values due to scale. Logarithm normalization was also implemented for two reasons: it is often used in time-series data such as stock prices, and it can be used as comparison to the min/max scaling method. As mentioned previously, sliding window is also used in the preprocessing step for handling the training and testing data. Finally, for variable selection, we have the option to use either all four features (open, close, high, low) or one pair. For this project, we chose high and low for our features after doing some preliminary selection.

*3) Model and method:* Model architecture is shown in fig. 3. Because variables are dependent on each other time series data, models that can account for the long term price changes. Popular models include Recurrent Neural Network (RNN) and LSTM models. In addition, modeling construction may go from basic neural network, to RNN with LSTM (or GRU) layer, to RNN with GRU layers with various ways of dropout. Different from the current literature, we initiate the architecture with two individual input layers to pair the inputs. Hidden layers are constructed by multi-input LSTM layers with twenty neurons, a concatenate layer that combines two LSTM layers into one vector, dropout layer to prevent overfitting, and two output layers that output low

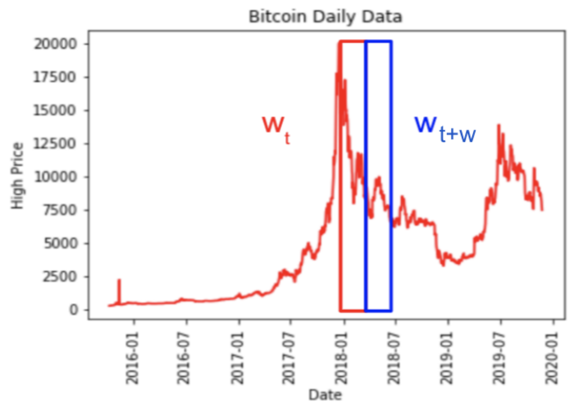

**Fig. 2:** Sliding window method with window size w at time t

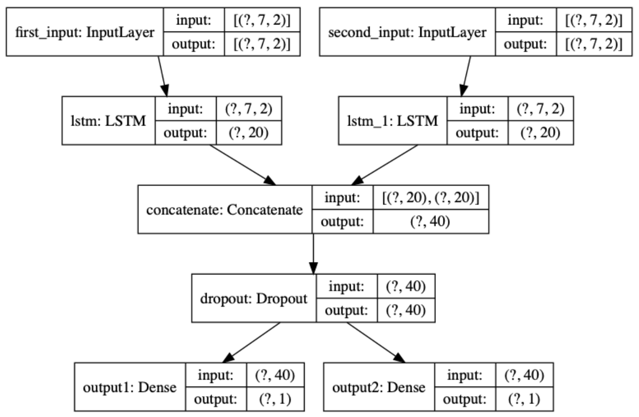
first_input: InputLayer | input: [(?, 7, 2)] | output: [(?, 7, 2)]
second_input: InputLayer | input: [(?, 7, 2)] | output: [(?, 7, 2)]
lstm: LSTM | input: (?, 7, 2) | output: (?, 20)
lstm_1: LSTM | input: (?, 7, 2) | output: (?, 20)
concatenate: Concatenate | input: [(?, 20), (?, 20)] | output: (?, 40)
dropout: Dropout | input: (?, 40) | output: (?, 40)
output1: Dense | input: (?, 40) | output: (?, 1)
output2: Dense | input: (?, 40) | output: (?, 1)

**Fig. 3:** This model takes two input layers and passed to two individual LSTM layers. After concatenation and dropout layers, the model predicts two output layers

and high prices for the next window.

Furthermore, we applied sliding window method by averaging prices in given window size to identify more general trends of the price movements. Shown in fig. 2, This method allows the model to predict the length of the window size, and because cryptocurrencies are highly volatile, sliding window method is able to minimize the variance.

*4) Hyperparameter Tuning:* When it comes to creating the LSTM model, plenty of experimentation will be necessary in order to find the best combination of parameters.We methodically established ranges for candidate hyperparameters.The parameter tuning is done using the Hyperopt library, which finds the best set of parameters that minimize the loss value given the maximum number of iterations, which we set to 100 iterations.

**Number of Neurons:** We chose to try 64, 128 and 256 neurons in the hidden layers. It costs a lot to have more neurons, as the training process will last longer, and we did not find trying a larger number of neurons provide improved results.

**Epochs:** Rather arbitrarily, we decided to try 10, 20, 30 for the number of epochs. As with the number of hidden layer neurons, the more epochs, the more time it takes for training

| Parameter | Values |
|---|---|
| Number of Neurons | 64, 128, 256 |
| Epoch | 10, 20, 30 |
| Batch Size | 100,200, 500, 1000 |
| Optimizer | SGD, Adam |
| Window Size | 3, 7, 10 |
| Activation Function | sigmoid, tanh,ReLu |
| Dropout Rate | 0.25, 0.21 |
| Loss Function | MAE, MSE |

**Fig. 4:** Table for hyperparameters used for tuning

to finish, since one epoch is a full iteration over the training data. Also, it may overfit the model.

**Batch Size:** With respect to the batch size.We tried several values:100, 200, 500, 1000.

**Optimizer:** While Stochastic Gradient Descent is used in many Neural Network problems, it has the problem of converging to a local minimum. For this experiment, we tested a few variations of adaptive learning optimizers such as Adam, Adagrad and RMSProp.

**Window Size:** It is an integer to be used as the look back window for creating a single input sample.We have tried 3, 7, 10 for the window length.

**Activation function:** The choice for activation function was not very difficult. The most popular are sigmoid, tanh, and ReLu.

**Dropout Rate:** Its value represents the percentage of disabled neurons in the preceding layer and ranges from 0 to 1. We have tried 0.25 and 0.21.

**Loss function:** The performance measure for regression problems, will typically be either RMSE (Root Mean Square Error) or MAE (Mean Absolute Error).

### C. *Results and Findings*

Using the Hyperopt library, we found two sets of best-tuned parameter values for Bitcoin Daily Data and Ethereum Daily Data. Amongst the adaptive learning optimizers that we tested, Adam was found to work slightly better than the rest. Both models have many similar optimal parameters, such as activation function, dropout rate, epoch and loss function. The main difference is that more neurons and batch size will work better for Bitcoin data compared to that of the Ethereum data. The best combinations are shown in fig. 5.

After tuning the model for the datasets, we then conducted the experiment in two ways. We first chose to scale the data with min-max scaling and making predictions on the actual price. The second experiment is using logarithm scaling and

| Parameter | Value |
|---|---|
| Number of Neurons | 256 |
| Epochs | 30 |
| Batch Size | 200 |
| Optimizer | Adam |
| Window Size | 7 |
| Activation Function | tanh |
| Dropout Rate | 0.21 |
| Loss Function | MSE |

Parameters of the best tuned model for Bitcoin Daily Data

| Parameter | Value |
|---|---|
| Number of Neurons | 64 |
| Epochs | 30 |
| Batch Size | 100 |
| Optimizer | Adam |
| Window Size | 3 |
| Activation Function | tanh |
| Dropout Rate | 0.21 |
| Loss Function | MSE |

Parameters of the best tuned model for Ethereum Daily Data

**Fig. 5:** The most optimal combinations after hyperparameter searching

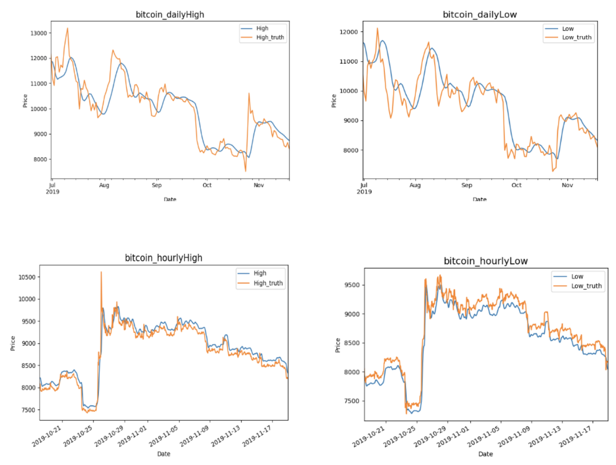

**Fig. 6:** Min-max scaling with price: Actual vs. predicted (high and low) prices for Bitcoin daily and hourly data

calculating the log return of the model. Both models use MSE as the loss function. The results for two experiments are shown below.

For both types of cryptocurrencies, the model performance is decent with very low loss on both the daily as well as hourly data. From the plots above, it seems to capture the signals and is able to make reasonable predictions with a small lag to the actual price. For the stock data however, it performed poorly compared to the cryptocurrencies. The price predictions for both the high and low daily values is significantly lower than the actual price ( $700 less). This result was somewhat expected, since the model was trained on the cryptocurrencies rather than the stock dataset.

Similar to the results using min-max normalization, the model trained with logarithm scaled data yielded decent performance for the cryptocurrencies. Instead of using price as the response variable, we use log-return. For the log-return equation, Pt is the lowest and highest price at time t calculated in the previous time period from t - $\Delta t$ to t, and t is the window size. The equation is the following :

$$Rt = \log(P_t) - \log(P_{t-\Delta t}) \tag{1}$$

Since we're using the sliding window method, the model's predictions resembles a smoothed curve that captures the signals of previous values with some lag. For both the daily and hourly data, the low and high log return provided by

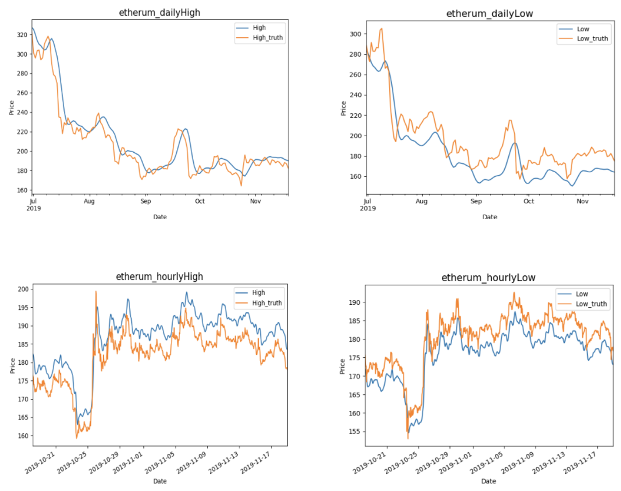

**Fig. 7:** Min-max scaling with price: Actual vs. predicted (high and low) prices for Ethereum daily and hourly data

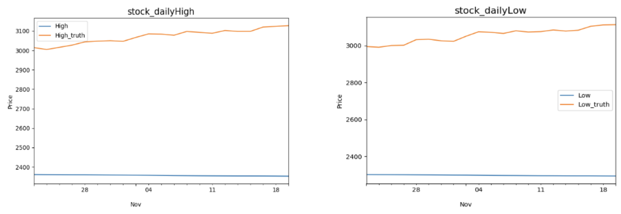

**Fig. 8:** Min-max scaling with price: Actual vs. predicted (high and low) prices for S&P 500 daily data

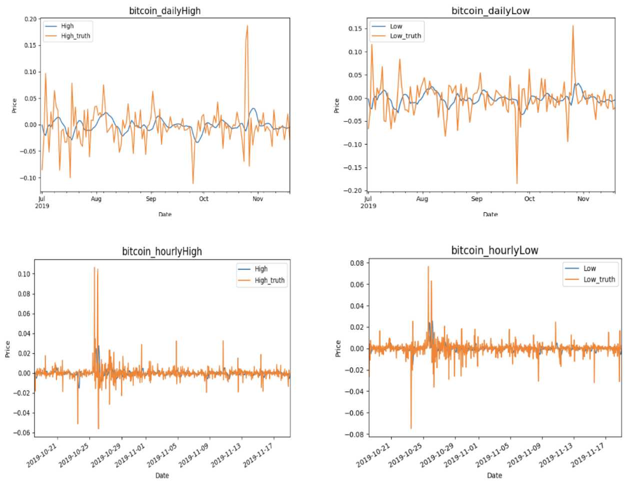

**Fig. 9:** Logarithm scaling with log return: Actual vs. predicted (high and low) prices for Bitcoin daily and hourly data

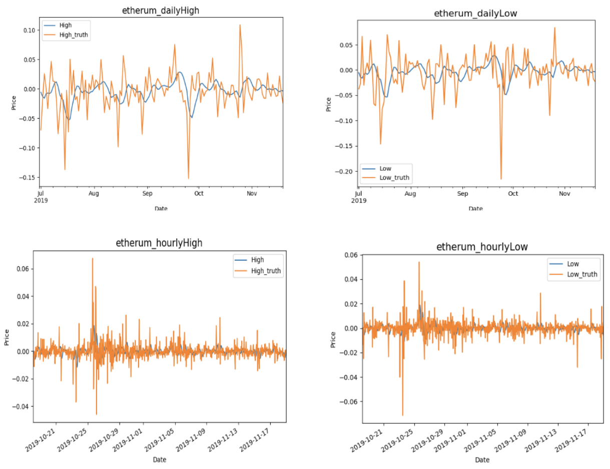

**Fig. 10:** Logarithm scaling with log return: Actual vs. predicted (high and low) prices for Ethereum daily and hourly data

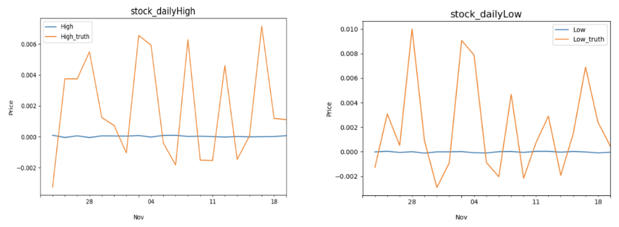

**Fig. 11:** Logarithm scaling with log return: Actual vs. predicted (high and low) prices for S&P 500 daily data

the model yielded reasonable predictions to the ground truth. The stock market prediction on the other hand, similar to the previous experiment, is not nearly as well due to the same reasons.

The deep neural networks used in this experiment has provided us with a better understanding of cryptocurrencies, how they compare to the stock market, and the LSTM architecture. As a next step, other network architectures (such as CNN) can be explored and compared to the current model. In addition, increasing the depth of the network or adding other features can be considered. From our experiments with Bitcoin, more features have not always led to better results, but it would be a topic worth investigating deeper into.

To sum up, at a technical level, we hope to expand upon and improve our approach with other architectures and more rigorous hyperparameter optimization (stochastic grid or bayesian search). Lastly, an increase in automation is desired to avoid large amounts of manual tuning for similar unseen datasets.

## VI. Potential Impact

Potential impacts of this project include formulating trading strategies on how much to buy or sell cryptocurrency on given window using high and low cryptocurrency price predictions. An ideal situation for the investor is so called, 'sell high, buy low' situation. With high accuracy in predicting high/low prices, investors can formulate strategies when to buy and sell the cryptocurrencies during the predicted window size. We were also able to compare the cryptocurrency market vs. stock market from a modeling perspective. Although both cryptocurrencies and stocks are known to be correlated, the model trained on cryptocurrency data did not actually yield reasonable results on the stock market. Due to high volatility in both markets, the model seems to miss features for both prices. Our next step is to find the best algorithmic mechanisms which help anticipate the short-term price behaviors of the cryptocurrency market. And we will further design a simple trading strategy assisted by state-of-the-art machine learning algorithms which could outperform standard benchmarks.

## VII. Contribution

- Report & Poster - Everyone
- Model & Pipeline - Minjun Park, Ying Xiong
- Data Preprocessing - Ying Xiong, Erin Song
- Parameter Tuning - Hao Xiang, Erin Song

## References

[1] Mcnally, S., Roche, J., & Caton, S. (2018). Predicting the Price of Bitcoin Using Machine Learning. 2018 26th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP). doi: 10.1109/pdp2018.2018.00060

[2] Madan, I., Saluja, S., Zhao, A. (n.d.). Automated Bitcoin Trading via Machine Learning Algorithms. Retrieved from http://cs229.stanford.edu/proj2014/Isaac Madan, Shaurya Saluja, Aojia Zhao,Automated Bitcoin Trading via Machine Learning Algorithms.pdf

[3] Yao, Y., Yi, J., Zhai, S., Lin, Y., Kim, T., Zhang, G., Lee, L. Y. (2018). Predictive Analysis of Cryptocurrency Price Using Deep Learning. International Journal of Engineering Technology, 7(3.27), 258. doi: 10.14419/ijet.v7i3.27.17889

[4] Sun, J., Zhou, Y., Lin, J. (2019). Using machine learning for cryptocurrency trading. 2019 IEEE International Conference on Industrial Cyber Physical Systems (ICPS). doi: 10.1109/icphys.2019.8780358

[5] `https://www.cryptodatadownload.com/data/`

[6] `https://hackernoon.com/dont-be-fooled-deceptive-cryptocurrency-price-predictions-using-deep-learning-bf27e4837151`

[7] Shah, D., & Zhang, K. (2014). Bayesian regression and Bitcoin. 2014 52nd Annual Allerton Conference on Communication, Control, and Computing (Allerton). doi: 10.1109/allerton.2014.7028484

[8] Georgoula, I., Pournarakis, D., Bilanakos, C., Sotiropoulos, D. N., & Giaglis, G. M. (2015). Using Time-Series and Sentiment Analysis to Detect the Determinants of Bitcoin Prices. SSRN Electronic Journal. doi: 10.2139/ssrn.2607167