Cryptocurrency Price Indicator

04.25.2018

Praveen Origanti

Udacity MLND Capstone Project

I. Definition

Project Overview

Cryptocurrency (or digital currency) is a revolutionary concept enabling peer-to-peer currency exchange without relying on a central trusted authority (like banks). Most cryptocurrencies use a decentralized/distributed ledger system, operating on the principles of cryptography, crowdsourcing and game theory. Satoshi Nakamoto first proposed these principles and defined the first blockchain database whose unit of currency is defined as bitcoin [1].

Bitcoin was created in 2009 and has mostly been limited to exploration by technological enthusiasts. The U.S. Treasury classified bitcoin as a convertible decentralized virtual currency in 2013 and the Commodity Futures Trading Commission, CFTC, classified bitcoin as a commodity in September 2015. Since then, it has garnered greater attention from general public and media. Mostly driven by speculative investments, it has seen an exponential rise in value in the last couple of years and several scholars expect the trend to continue, while others consider it to a bubble. Also, numerous other cryptocurrencies have been created as either variants of bitcoin or new platforms to support other applications. Ethereum, Litecoin and Ripple are a few among the popular ones, also referred to as alt-coins.

Most of the cryptocurrencies posted large gains over the last few years, and peaked in value in December 2017 (as of April 2018). This led to a strong interest in predicting the cryptocurrencies in both financial and academic circles.

Personally, I've been impressed with the technology for a while, but haven't made any investments. I would like to take this opportunity to understand the market (and technology) better and decide if it is a good time to invest in cryptocurrencies.

Problem Statement

The objective is to predict daily percentage returns and daily volatility of the two most popular coins - bitcoin and ethereum - on a given day using the past week's historical market data and appropriate google search trends. These are regression problems, as the goal is to predict continuous numerical values (both returns and volatility).

Since this is a time series problem, I've used recurrent neural networks (RNN), particularly LSTM (Long short-term memory) model, to predict the a given day's market behavior based on historical data. I've used Python3 and Keras package with Tensorflow backend. Also used sklearn's TimeSeriesSplit and GridSearchCV for cross validation and exploration of parameters space respectively.

Metrics

R^2 metric, Mean Squared Error (MSE) and Mean Absolute Error (MAE) are potential metrics for this problem. I've considered both MAE and MSE for the problems of predicting volatility and daily returns. To impose strong penalties on large errors and build aggressive models, I've chosen MSE over MAE as the primary metric for this project.

The model results would be considered satisfactory if the predicted results fare better than benchmark models.

II. Analysis & Methodology

Data Exploration

The historic price data of cryptocurrencies (along with other alt-coins) is available on coinmarketcap.com [3]. For each day from Aug 07 2015 to April 18 2018, I retrieved data from [3] for bitcoin, ethereum, litecoin and ripple that includes open/close/low/high prices, trading volume and market cap. This makes up 986 records of data for all four coins.

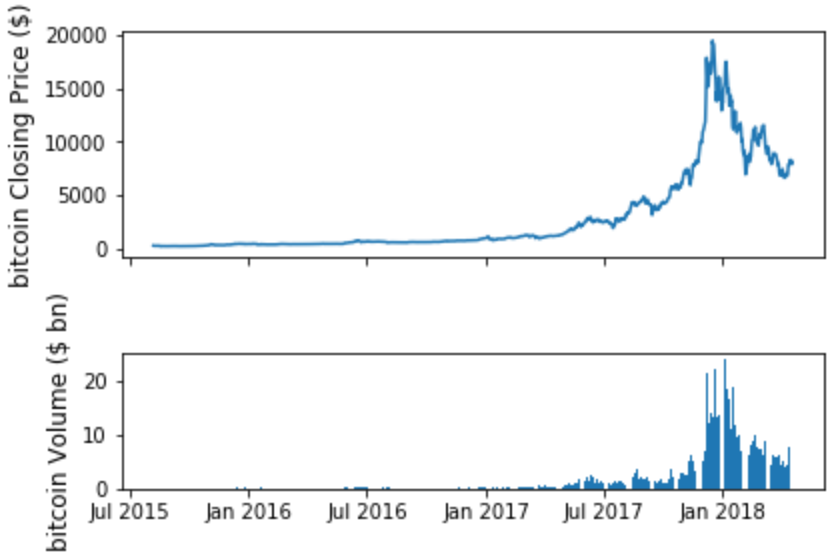

Image 1: Bitcoin's closing price and trading volume on full dataset

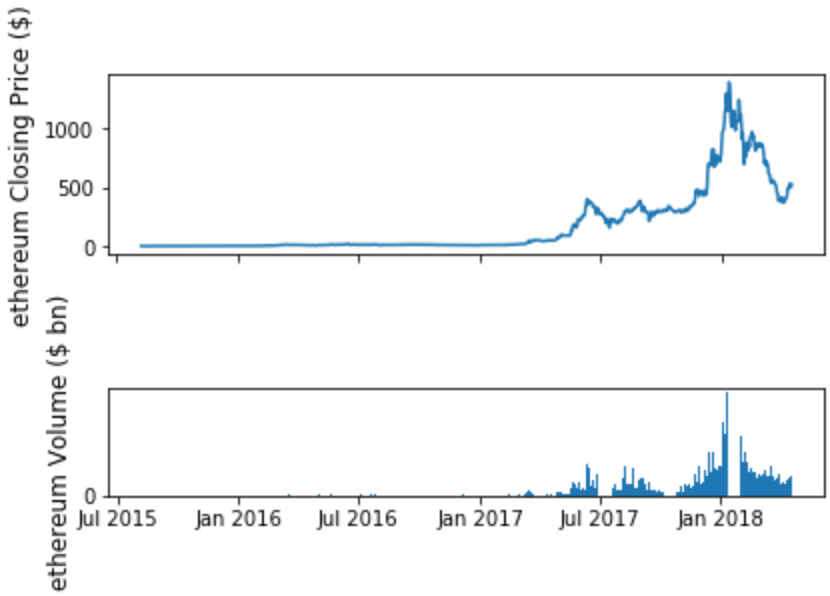

Image 2: Ethereum's closing price and trading volume on full dataset

As we see in Image 3, there is a strong correlation between Market Cap, Open, Close, High and Low prices of bitcoin. So, I've dropped Market Cap, Open, High and Low prices and selected these features - Close price and Volume.

I've included these new features based on available market data:

1. Daily returns: (Close - Open)/Open

2. Volatility: (High - Low)/Open

3. Close Off High: (High - Close)/(High - Low)

4. 7-day mean returns: 7-day average returns

5. 30-day mean returns: 30-day average returns

Image 3: Pearson Correlation for Bitcoin's market attributes

Using the python package pytrends [4], I've retrieved the search trends for these four terms - bitcoin, ethereum, litecoin and ripple. For the long duration of interest, Google Trends provided normalized data on a weekly basis. Instead, I've generated multiple queries on smaller time durations to retrieve daily data and merged these results by rescaling based on ten overlapped dates in successive queries. Both coinmarketcap and Google Trends use UTC timezone, so there are no issues in merging the datasets.

I've used the most recent ~10% as test set (89 records) and the rest as training set (897 records).

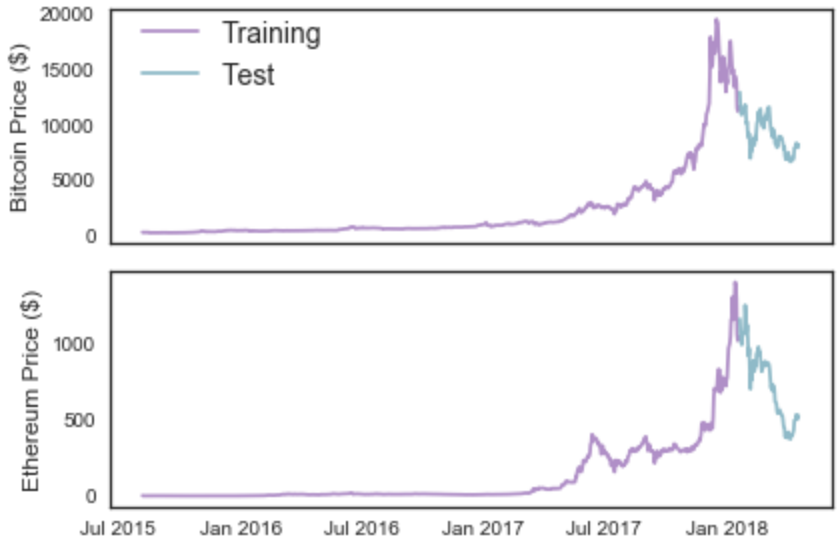

Image 4: Training and test data for Bitcoin and Ethereum (Close Price)

NOTE: Given the high volatility in these markets, it is hard to arrive at useful models just based on price data [2]. In this light, I also wanted to explore the possibility of including twitter/reddit sentiment over this period of time. However, I couldn't obtain freely accessible twitter/reddit/stocktwits data over this period of time. I could generate the required sentiment data on Quantopian platform [5], but Quantopian has strict limitations on using this data outside their platform. I failed to obtain permissions to use the data for this project. So, I've instead used Google Trends data on these search terms - bitcoin, ethereum, litecoin and ripple.

Target Data Exploration:

Instead of predicting the Close Price, I've looked at predicting the "Daily returns" of bitcoin and ethereum (Image 5), for the following reasons

- What matters most in any markets is the percentage returns and not the exact price of the commodity/stock/coin.
- We are mostly interested in the returns and not the Close Price
- Daily returns has a naturally gaussian distribution (Image 7)

In addition, I've also predicted "Daily Volatility" of bitcoin and ethereum (Image 6, 8) to see if there are stronger signals here.

Image 5: Training and test data for Bitcoin and Ethereum (Daily returns)

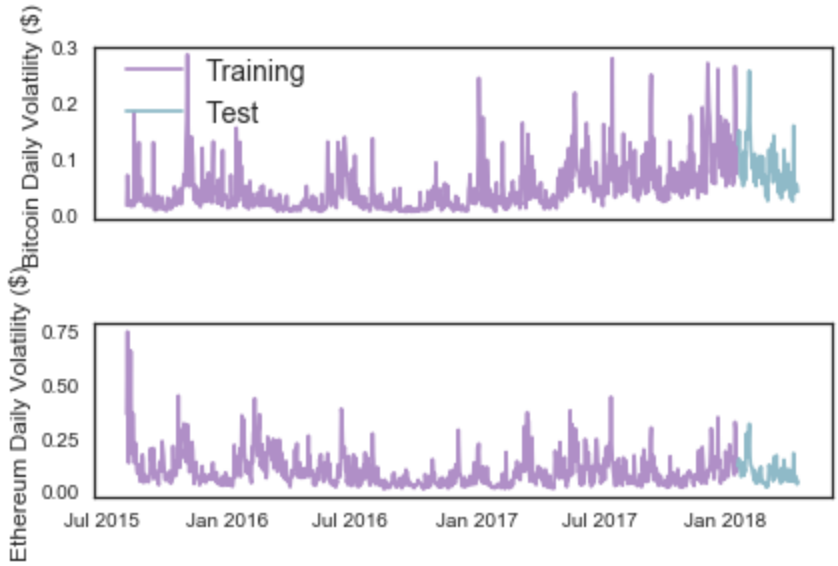

Image 6: Training and test data for Bitcoin and Ethereum (Volatility)

Given the long tail distributions of “Daily Volatility” of bitcoin and ethereum (Image 8), I’ve applied log transformation and observed closer to gaussian distributions (Image 9).

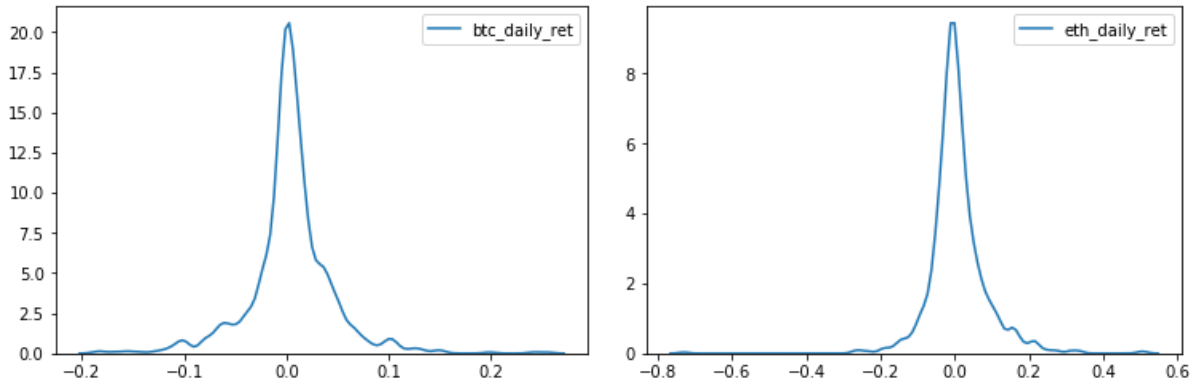

Image 7: Distribution of daily returns for Bitcoin (Left) and Ethereum (Right)

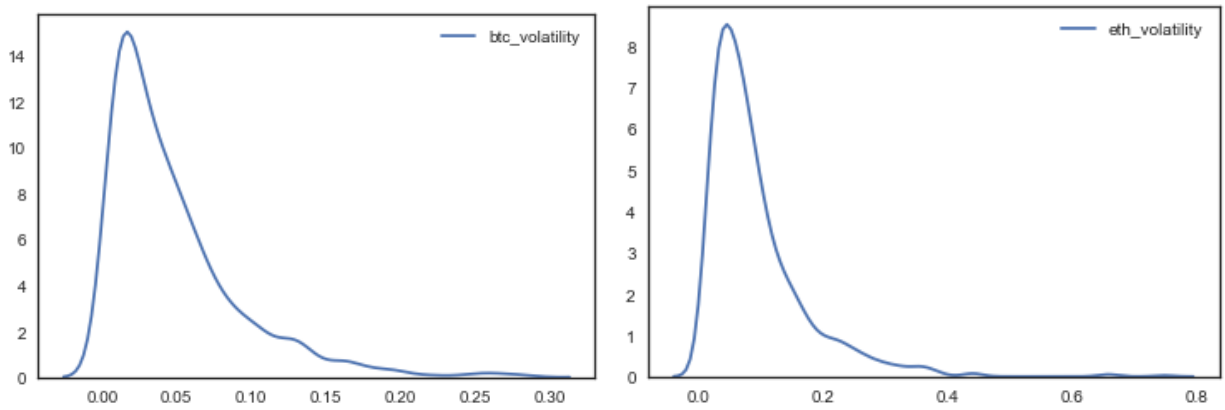

Image 8: Distribution of daily volatility for Bitcoin (Left) and Ethereum (Right)

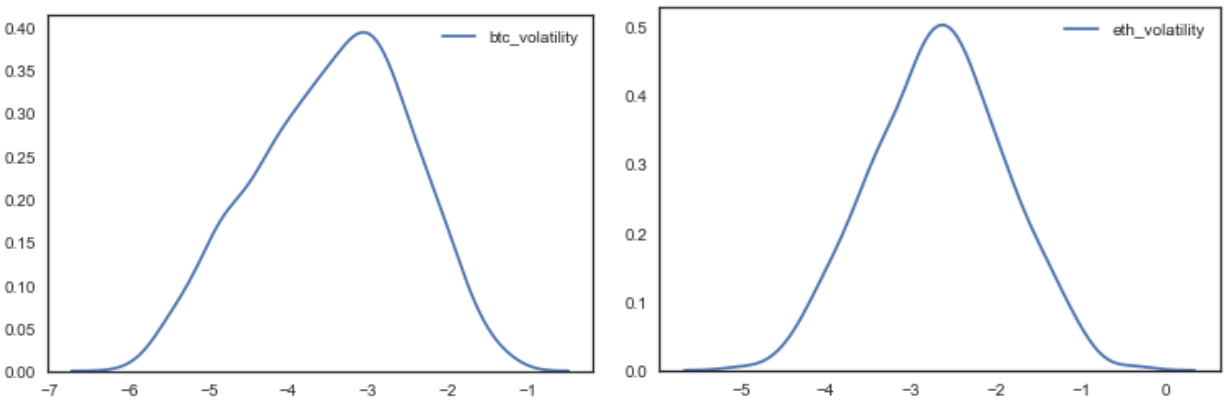

Image 9: Distribution of log(daily volatility) for Bitcoin (Left) and Ethereum (Right)

I've appended google search trends to the above set of features. Image 10 captures the pearson correlation of the bitcoin related features used:

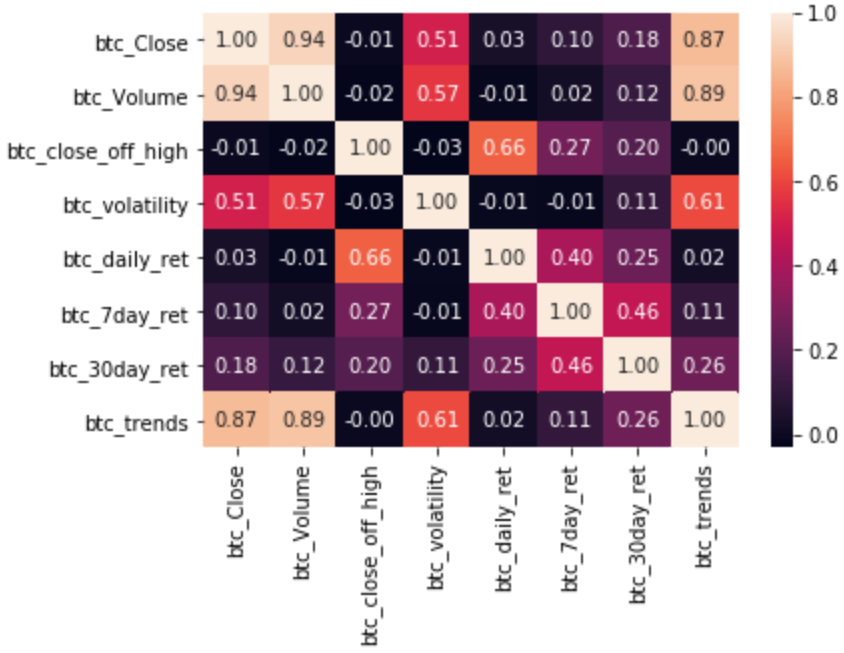

Image 10: Pearson Correlation for Bitcoin features used.

It can be seen that the search trends for bitcoin (btc_trends) strongly correlates to Close Price (btc_Close), but not much with daily returns. It is interesting to note that there is a decent correlation between btc_trends and btc_volatility (0.61)

Algorithms and Techniques

The objective is to predict the daily percentage returns and volatility of the two most popular coins - bitcoin and ethereum - on a given day using the past week's (window size = 7) historical market data and appropriate google search trends.

Since this is a time series, I've used LSTM (Long short-term memory) to predict the next day's market behavior using historical data. I've used Keras package with Tensorflow backend, sklearn's TimeSeriesSplit and GridSearchCV for cross validation and exploration of parameters space respectively

Normalization

Instead of using the MinMax scaling or z-scaling (zero mean, unit variance), I've followed the normalization approach described here [6] - for each sliding window, scale the values using the first record in the window as the reference such that the first record's value would be zero (after scaling)

I've used the following equations to normalise and subsequently de-normalise at the end of the prediction process to get a real world number out of the prediction:

n = normalised list [window] of price changes

p = raw list [window] of adjusted daily return prices

Normalisation:

$$n_i = (p_i/p_0)-1$$

De-Normalisation:

$$p_i = p_0(1+n_i)$$

The Close Price, Volume and Google Trends Data are normalised this way. I didn't have to de-normalise since these are only independent features for our prediction models.

GridSearch & Cross-Validation

Using sklearn's TimeSeriesSplit, five CV folds are created - successive training sets would be supersets of the previous ones. This ensures that the folds are not created randomly - and the principles of time series data are strictly adhered to.

Using keras.wrappers.scikit_learn.KerasRegressor, I've used sklearn's GridSearchCV to explore the search space and find the best hyperparameters:

# of neurons in the three LSTM layers: [[10, 8, 4], [20, 16, 8], [40, 32, 16], [80, 64, 32], [160, 128, 64]]

Dense layer activation Function: ['linear', 'sigmoid', 'tanh']

Dropout: [0.4, 0.5, 0.6, 0.7]

Optimizer: ['sgd', 'rmsprop', 'adam']

For the rest of Keras parameters, I've used default values [7]

I couldn't use GridSearchCV to find the best window size, but I've tried with different window lengths (1, 2, 7, 14, 30) iteratively to find the best fit.

Early stopping

I couldn't use Keras Early stopping with sklearn's cross-validation [8]. But in the initial trials with {train, validation} split, I am able to use EarlyStopping feature:

```
early_stopping_monitor = EarlyStopping(patience=2)
model.fit(...._split=0.2, epochs=100, shuffle=false, callbacks=[early_stopping_monitor])
```

Benchmarks

I have two benchmarks for each the four prediction problems:

1. Price-persistent model: This assumes the current day's price would stay the same as previous day's close price. In this model, both daily returns and volatility would be zero. This model is a good reference for predicting daily returns
2. Market-persistent model: This assumes the market would behave the same as previous day. In this model, both daily returns and volatility would be same as the previous day. This model is a good reference for predicting daily volatility

Implementation & Refinement

One note regarding preparing google trends data - I've coded up the function get_trends_data from scratch. Given that google only provides weekly data when the specified time duration is greater than 6 months, I initially interpolated the daily data based on the weekly data. Hoping to increase the data quality, I've used multiple queries using shorter durations to obtain the daily data. However, the results on each query are normalized. So, I ended up having an overlap of 10 days between successive queries and scaled the values while merging the results from successive queries.

Since this function is complex, I tried to validate it as much as I can, by cross-checking with the weekly results provided by a single query.

Further refinements included switching to Mean Squared Error (instead of Mean Absolute Error) to build aggressive models, using Litecoin and Ripple market & trends data additionally, tuning 'optimizer' parameter and trying with multiple window sizes.

I haven't changed the split data, always using the 10% latest data as test set.

III. Results

Model Evaluation and Validation

1. Predicting Daily Returns

For predicting daily returns of bitcoin and ethereum, based on the GridSearchCV results, the model in Image 11 obtained the best scores. However, the best MSE score was not satisfactory as it is about the same performance as benchmark-I (Table -I)

The predicted returns are observed to be pretty close to 0 most of the time, likely because there are no clear patterns that could be learned from this data.

	BTC Returns	ETH Returns	BTC Volatility	ETH Volatility
Benchmark-I	0.00306	0.00354	7.188	7.075
Benchmark-II	0.006	0.007	0.282	0.266
Best LSTM model	0.00305	0.00362	0.232	0.250

2. Predicting Daily Volatility

For predicting daily volatility of bitcoin/ethereum, a slightly more complex model (neurons: [20, 16, 8]) achieved the best score. The best MSE scores for bitcoin volatility and ethereum volatility are 17% and 6% better their benchmark-II scores (Table-I).

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_124 (LSTM)              (None, 2, 10)             1720
_________________________________________________________________
dropout_124 (Dropout)        (None, 2, 10)             0
_________________________________________________________________
lstm_125 (LSTM)              (None, 2, 8)              608
_________________________________________________________________
dropout_125 (Dropout)        (None, 2, 8)              0
_________________________________________________________________
lstm_126 (LSTM)              (None, 4)                 208
_________________________________________________________________
dropout_126 (Dropout)        (None, 4)                 0
_________________________________________________________________
dense_42 (Dense)             (None, 1)                 5
_________________________________________________________________
activation_42 (Activation)   (None, 1)                 0
=================================================================
Total params: 2,541
Trainable params: 2,541
Non-trainable params: 0
_________________________________________________________________
None
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_124 (LSTM)              (None, 2, 10)             1720
_________________________________________________________________
dropout_124 (Dropout)        (None, 2, 10)             0
_________________________________________________________________
lstm_125 (LSTM)              (None, 2, 8)              608
_________________________________________________________________
dropout_125 (Dropout)        (None, 2, 8)              0
_________________________________________________________________
lstm_126 (LSTM)              (None, 4)                 208
_________________________________________________________________
dropout_126 (Dropout)        (None, 4)                 0
_________________________________________________________________
dense_42 (Dense)             (None, 1)                 5
_________________________________________________________________
activation_42 (Activation)   (None, 1)                 0
=================================================================
Total params: 2,541
Trainable params: 2,541
Non-trainable params: 0
```

Image 11: The model summary used for Bitcoin & Ethereum's Daily Price Prediction.

I've tried with different random seeds for these models, and the variations in results are within +/- 3%, so I conclude the models are robust and the results are trustworthy.

Justification

The predictions on daily returns are barely useful given they just match benchmark model-I. Interestingly, the predictions on volatility are better than the best benchmarks indicating that it is worth pursuing in this direction. However, I am not sure if predicting market volatility is useful (for investors, i.e.).

I don't consider the results to be significant enough, but it can be attributed to several aspects - primarily the availability of patterns/signal in the data.

IV. Conclusions

Free-Form Visualization

The most interesting observation for me is that LSTM models for daily returns ended up predicting values close to 0, while the actual market returns have huge variations (Image 12). To me, this is a strong indicator that there is little signal in the available data to make deduce meaningful information.

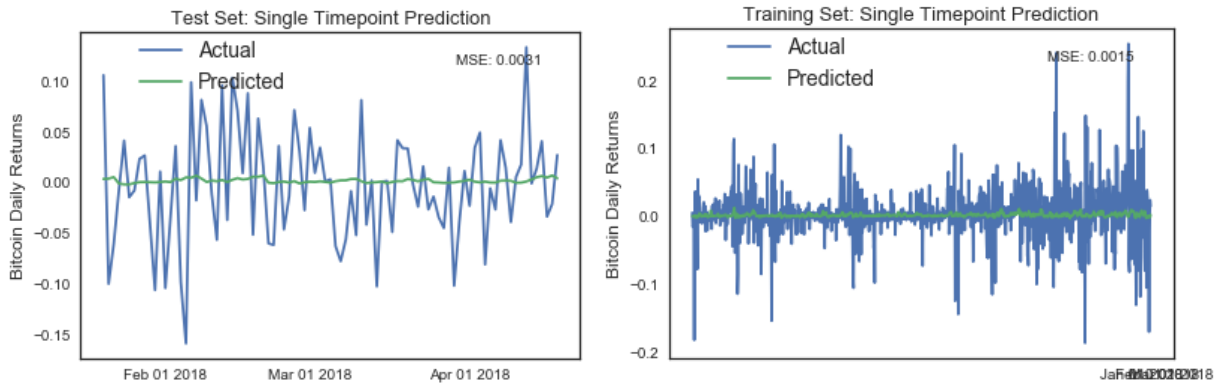

Image 12: Actual & Predicted daily returns of Bitcoin on test (Left) and train (Right) data

On the other hand, the predictions on volatility have a larger variance (still conservative compared to market behavior), indicating a potential signal in this data. (Image 13)

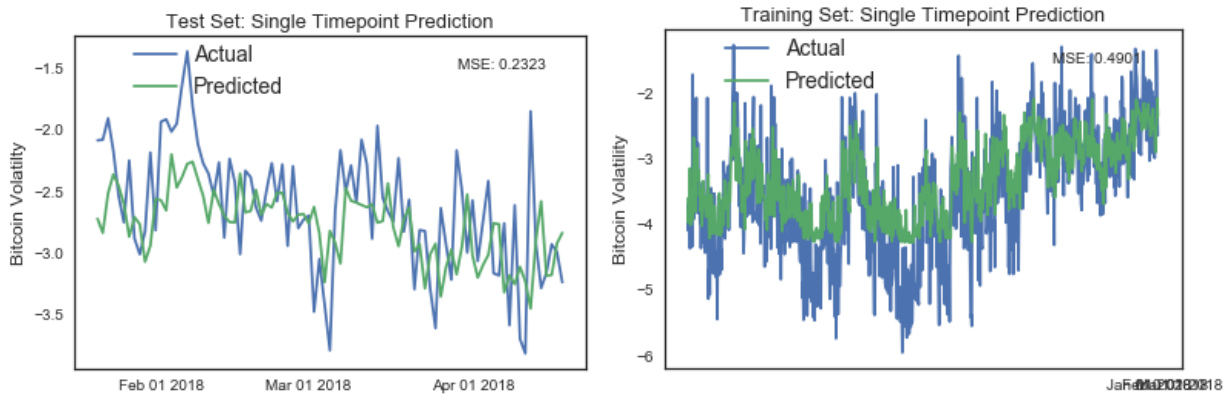

Image 13: Actual & Predicted daily volatility of Bitcoin on test (Left) and train (Right) data

Reflection

When I couldn't obtain sentiment information from twitter/stocktwits/reddit, I was a little concerned and regretted to have chosen this problem. I still believe using sentiment data might provide some insight into this problem.

However, I'm extremely happy to have explored this problem so far. In the process, I've learned a lot of new things - Google Trends, Google Correlate (this is an eye-opener), Google Colab and Quantopian platform for algorithmic trading.

I wouldn't propose using the project (in this state) in an investment setting. In the near future, I plan to import this work into Quantopian platform and leverage the additional sentiment data retrieved in there.

Scope for Improvement

A few aspects that can certainly improve the training process:

- GridSearchCV failed with njobs=-1. The parallelism helps explore better
- Batch Normalization layer, which I haven't used in these models
- Early Stopping - not integrated with GridSearchCV. But there must be a way
- Better visualization on train/validation error trends, to identify overfitting. Was able to produce such plots using keras model_history, but not with GridSearchCV

I believe better data (investor sentiment) can help. Based on hyperparameter tuning, more complex LSTM architectures did not provide better results, so I don't think using complex architectures would help

References

1. https://bitcoin.org/bitcoin.pdf
2. https://dashee87.github.io/deep%20learning/python/predicting-cryptocurrency-prices-with-deep-learning/
3. https://coinmarketcap.com/
4. https://github.com/GeneralMills/pytrends
5. https://www.quantopian.com
6. http://www.jakob-aungiers.com/articles/a/LSTM-Neural-Network-for-Time-Series-Prediction
7. https://keras.io/layers/recurrent
8. https://stackoverflow.com/questions/48127550/early-stopping-with-keras-and-sklearn-gridsearchcv-cross-validation