Predicting Price Changes in Ethereum

Matthew Chen, Neha Narwal and Mila Schultz
Stanford University
Stanford, CA 94305
[mchenja, nnarwal, milafaye]@stanford.edu

1 Introduction

The market capitalization of publicly traded cryptocurrencies is currently above $230 billion (Bovaird, 2017). Bitcoin, the most valuable cryptocurrency, serves primarily as a digital store of value (Van Alstyne, 2014), and its price predictability has been well-studied (Hegazy and Mumford, 2016). However, Ethereum has the second-highest market capitalization and supports much more functionality than Bitcoin. While its price predictability is sparsely covered in published literature, the technology's additional functionality may cause Ether's price predictability to differ significantly from that of Bitcoin. These characteristics are outlined in the following subsection; the underlying details of Bitcoin (Nakamoto, 2008) and Ethereum (Buterin, 2013) are elided, as they are described in depth in the cited papers.

1.1 Price Predictability of Asset Classes

The financial concept of volatility, which is the standard deviation of price returns, represents a measure of predictability. For example, the price of an asset with zero volatility can be predicted with 100% accuracy, even as the price changes over time.

Analysis of price data from Coinbase (2017) shows that between August 30th, 2015 and October 19th, 2017, Bitcoin had a monthly volatility of 21.73%. Over that same time span, Ethereum had a monthly volatility of 77.91%. For comparison, the S&P 500 has a historical monthly volatility of about 14%, suggesting that the price of Ether is significantly less predictable than that of either Bitcoin or common stock.

This difference persists after accounting for the fact that Ethereum was created in 2015; analyzing Ether prices in 2017, for example, yields a monthly volatility of 89.15%. And unitwise, more Ether than Bitcoin is traded each month, so low Ether trading volumes are not the cause of this volatility either.

A logical explanation for the high volatility of Ether, relative to Bitcoin and common stock, is that Ether is transferred or traded in a way in which Bitcoin and stock are not. More specifically, Ethereum has contract accounts which can cause Ether to be transferred between accounts in an unpredictable manner. (While Bitcoin and common stock are also traded by algorithms, the distinction is that anyone - even someone writing a *hello world* program - can instantiate a contract account on Ethereum. Meanwhile, algorithmically trading Bitcoin or stock requires more technical sophistication.)

In recognition of Ether's high volatility, this project aims to predict the directionality of Ether price changes. In other words, this project will aim to apply machine learning techniques to answer the question, "Will Ether increase in price tomorrow?"

We are pursuing our project in conjunction with CS 221. Some of our dataset loading and cleaning code is shared; one of our logistic regression classifiers is shared and then used as input to a portfolio management strategy for Ethereum, which is our problem statement for CS 221.

2 Related Work

The algorithms detailed in §4 draw inspiration from prior work on predicting Bitcoin prices.

2.1 Predicting Price Changes in Bitcoin

Previous work on predicting the directionality of Bitcoin prices has shown that significant signal exists in the price of the cryptocurrency. Hegazy and Mumford (2016) compute an exponentially-

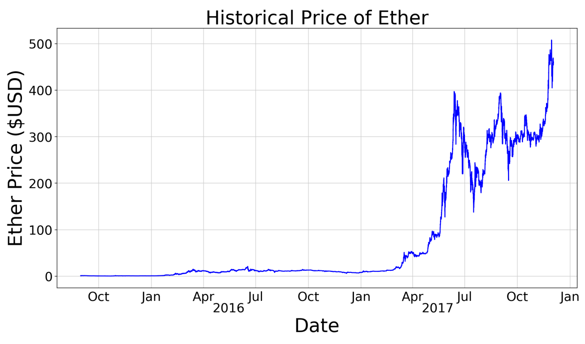

Figure 1: Ether Price History

smoothed Bitcoin price every eight minutes; using the first five left derivatives of this price as features in a decision-tree based algorithm, they predict the direction of the next change in Bitcoin price with 57.11% accuracy.

Their results substantiate earlier research done by Madan, Saluja, and Zhao (2014), who found that by using the Bitcoin price sampled every 10 minutes as the primary feature for a random-forest model, they could predict the direction of the next change in Bitcoin price with 57.4% accuracy.

An alternative model was used by Sebastian, Katabarwa, and Li (2014), who use the Bitcoin price sampled every minute as the primary feature for a forward-feed neural network. Their results suggest that this system predicts future Bitcoin price directionality with 60% accuracy.

3 Datasets

The primary dataset consists of the price of Ether sampled at approximately one-hour intervals between August 30, 2015 and December 2, 2017 (Etherchain, 2017). This dataset is plotted in Figure 1, and summary statistics for this dataset are listed in Figure 2.

Sample Size	19757
Minimum	0.41
1st Quartile	7.81
Median	11.7
3rd Quartile	88.56
Maximum	507.94
Mean	79.4737
Standard Deviation	122.95

Figure 2: Statistics of Ether Prices

3.1 Dataset Truncation

The variance of the dataset is large, relative to its mean, and so we initially attempted to reduce variance by truncating the dataset to only include data points occurring after February 26th, 2017. This date was chosen based on the analysis of Bovaird (2017), which suggested that the recent Ether price increases were institutionally-driven.

On February 27th, 2017, the *New York Times* had reported the forming of the Enterprise Ethereum Alliance, a consortium of 30+ significant institutions that agreed to collaborate on Ethereum development (Popper, 2017). This date marks a natural inflection point in Ethereum's price history, which seemed to support Bovaird's hypothesis, and roughly indicates when large institutions increased their interest in Ethereum (since if there was no such interest, the institutions would not have announced the consortium's formation.)

Sample Size	6667
Mean	220.0679
Standard Deviation	122.14

Figure 3: Statistics of Ether Prices (Truncated)

However, truncating the dataset in this way did not significantly change results. The variance of the truncated dataset is unchanged (although it it smaller relative to the mean) and the accuracy of prediction models presented in later sections did not change in any significant way. All this analysis was done after final results were established, however, as moving from the truncated dataset back to the full dataset can cause information from the test set for the truncated dataset to leak into the development set for the full dataset. As such, only results obtained on the truncated dataset are described in this paper.

3.2 Data Splitting

Following standard practice, the full dataset is split into portions of 80%, 10%, and 10%, which represent the training set, the development (or validation) set, and the test set. The splits are done by time, such that the earliest data point in the test set is in the future relative to the latest data point in the development set.

Hyperparameters are tuned by training on the training set and testing on the development set. Final results are obtained by training on the training and development sets and testing on the test set.[1]

3.3 Feature Selection

Features were generated by grouping the original data points, which contained Ether prices, into series of six points, such that each point was separated from its neighbors by one hour. (The original dataset included gaps between some points, such that not every point was separated by an hour from its neighbors. As these gaps were rare and widely spaced, the grouping mechanism ensured the produced features were valid by simply disregarding points which were too close to one of these gaps.)

More precisely, an input to the classifiers is $(p_{t-5}, p_{t-4}, p_{t-3}, p_{t-2}, p_{t-1}, p_t)$, where t is the time, and p_t is the price at time t. The state consists of the price at time t along with the 5 previous time points, for a total of 6 time points. The target prediction is the sign of $p_{t+1} - p_t$, an element of $\{+1, -1\}$.

The length of each feature vector was chosen to be 6 because these vectors were also used as input for a Markov Decision Process-based model in an associated project. For completion, alternative-length features were generated and tested using the process described above, and notable results are included in §5. Other types of features were also tested, including the price change between time points and the sign of the price change between time points, as well as normalized and standard versions of all the features already described.

4 Methods

Multiple models were assessed on the task of predicting the directionality of change in Ether price. Most of these models used 6 price points as the input feature and were based on binomial classification algorithms, including Logistic Regression, Support Vector Machine, Random Forest and Naive Bayes. Other models were based on regression algorithms, such as the autoregressive integrated moving average (ARIMA). Models based on a recurrent neural network (RNN) and a Neural Network (NN) were also implemented and tested.

All of the models were assessed on how well they performed on the task, and these results are given in §5. The impetus for trying such a large number of models was to analyze how the assumptions underlying each of the respective models could affect the models' performance. The methods underlying these models and their assumptions are briefly summarized below. Our implementations made use of Scikit-Learn (Pedregosa et al., 2011).

[1]In the milestone version of this paper, an uncaught error caused the models to train on both the training and development sets before being tested on the development set. This error, which exaggerated model accuracy, was caught during cross-validation; our results have been amended.

4.1 Logistic Regression

Logistic regression is a binary classification model which makes very few assumptions about the dataset. Its hypothesis function has the form

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$

where x is the input and θ the parameter that must be learned. This model also employs ℓ_2-regularization to minimize overfitting.

4.2 Naive Bayes

Naive Bayes is a binary classification model which makes a strong assumption about the conditional independence of the input features. Specifically, it assumes that the input features (i.e., the groups of 6 price points) are conditionally independent given the label (i.e., a positive price change [+1] or a negative price change [-1]). The classifier is

$$\hat{y} = \arg\max_y P(y) \prod_{j=1}^{n} P(x_i \mid y)$$

where n is the number of features, x is the input, and y is the class label. We use Gaussian Naive Bayes, where the feature likelihood is given by

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi}|\sigma_y|} \exp\Big(- \frac{(x_i - \mu_y)^2}{2\sigma_y^2} \Big)$$

and assumed to be Gaussian, with $\mu_{-1}, \mu_1, \sigma_{-1}, \sigma_1$ computed with maximum likelihood estimation.

4.3 Support Vector Machine

Like logistic regression, the support vector machine algorithm yields a binary classification model while making very few assumptions about the dataset. The classifier is obtained by optimizing:

$$\min_{\gamma, w, b} \ \frac{1}{2}||w||^2$$
$$\text{s.t.} \quad y^{(i)}(w^T x^{(i)} + b) \geq 1, \ i = 1, \ldots, m$$

where x is the input and w, b are parameters that must be learned. Predictions are made by analyzing the value of $w^T x + b$.

4.4 Random Forest

Random forests is a modification of bagging that builds a large collection of de-correlated trees, and then averages them. The essential idea in bagging is to average many noisy but approximately unbiased models, and hence reduce the variance. When used for classification, a random forest obtains a class vote from each tree, and then classifies using majority vote. Random forests approximate the expectation

$$\hat{f}_{rf} = E_\theta T(x;\theta) = \lim_{B\to\infty} \hat{f}(x)^B_{rf}$$

with an average over B realizations of θ.

4.5 Auto Regressive Integrated Moving Average (ARIMA)

ARIMA is a model used for time series analysis and forecasting. The model is used on time series data which will be transformed into a stationary time series; the predictions are a linear regression upon features including time differences and moving averages. The implementation used is from the Statsmodels package (Seabold and Perktold, 2010). In ARIMA, the data is differenced; that is, the price features are transformed to the difference between prices. Let L be the lag operator, then the ARIMA equations are

$$\left(1 - \sum_{k=1}^{p} \alpha_k L^k\right)(1-L)^d X_t = \left(1 - \sum_{k=1}^{q} \beta_k L^k\right)\epsilon_t$$

and p, d, q are hyper-parameters over which we optimized. At each time t, we train a model using the price history to predict the price at time t and use the sign of the change in price as a prediction.

4.6 Recurrent Neural Network (RNN)

Recurrent neural networks have been used to predict time series data. We used a two-layer RNN with a fully connected layer with softmax activation. The RNN layers consist of Gated Recurrent Unit (GRU) cells. We also used Long short-term memory (LSTM) cells. We tried different numbers of units for the layers, training times, and batch sizes. We have implemented the neural networks with both Keras (Chollet et al., 2015) and TensorFlow (Abadi et al., 2015) directly.

5 Results

The ratio of positive to negative price changes in the dataset is almost 1:1; as such, the models are evaluated based on their prediction accuracy. (If the dataset had a something like a 80%/20% split between positive and negative price changes, however, then F_1 score would have been used as a metric instead of accuracy.)

Method	Accuracy
Logistic Regression	53.40 %
Logistic Regression [Binary]	56.94 %
Naive Bayes	51.78 %
Support Vector Machines	51.29 %
Support Vector Machines [Change]	52.59 %
Support Vector Machines [Binary]	55.99 %
Random Forest	50.81 %
ARIMA	61.17 %
Recurrent Neural Network	52.43 %
Neural Network	52.18 %

Figure 4: Ether Price Change Predictor Accuracies

The test set prediction accuracies of the best models are reported in Figure 4. The features input to the model are indicated in brackets. An absence of brackets means that the input features were the 6 price points previously mentioned, while [Change] means that the features were the previous real-number changes in price and [Binary] means that the features were the previous sign changes in price. (For reference, a naive model which takes no input and always predicts the price will increase yields a baseline accuracy of 55.8%.)

We interrogated the dataset with TSNE, LDA, and PCA to discover that the classes are not qualitatively well-separated by any of those methods. Figure 5 shows the PCA representation with two principal components; the classes have much overlap, indicating that the classification task may be challenging.

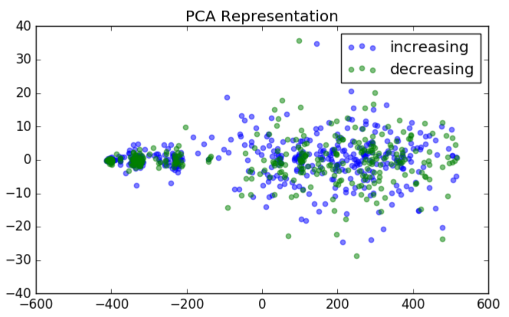

Figure 5: PCA with Two Principal Components

5.1 Best Performance

The ARIMA algorithm had the best performance. The price of Ether is not stationary; like many cryptocurrencies, it is volatile and also trending upward or trending downward. In this case, the prices are trending downward, and the integrative part of the algorithm which converts the features into differences accounts for the trend and allows the rest of the model to learn on stationary features. Additionally, the ARIMA model is trained every time step with all observed data. Hence, when it is making a prediction, it has all history before the predicted time. This data is to the advantage of the performance. The confusion matrix in Figure 6 indicates that the model has learned to predict both increasing and decreasing prices rather than more accurately predicting one class over another.

	Increase	Decrease
Increase	29	19
Decrease	21	34

Figure 6: A confusion matrix for the ARIMA model. The row indicates the predicted change in Ether price, while the column indicates the actual change.

5.2 Other Models

The other models tested are not designed specifically to work with time-series data, unlike the ARIMA-based model. As such, it is unsurprising that those algorithms underperform relative to ARIMA.

However, even after attempting to account for the time-series aspect of the data, the binomial classifiers still underperform. For example, a reasonable assumption could be made that the distribution of price differences does not change over time. (i.e., even as the price of Ethereum might vary, the distribution describing the magnitude of the price changes between iterations would stay constant.) This assumption guided our idea to use price changes (and the sign of the price change) as input features into a SVM-based model, but the model underperformed the ARIMA-based model all the same, even when the data was standardized and/or normalized.

The logical conclusion is that when working with time-series data, a model which is explicitly designed for such an input is likely to yield better results than a model which is not, even when the time-series data is massaged into a form that should be time-invariant.

Of course, the assumptions that each model makes are important as well. The Naive Bayes-based model barely outperforms a coin flip; this underperformance is most likely due to the fact that our data is quite far away from being normally distributed.

While the logistic regression-based model's assumptions were not violated, it is only able to classify accurately if a separable hyperplane exists. The poor accuracy yielded by this model suggests that the data is not well-separable, a result which is backed up by the poor performance of the SVM-based models. (This is also most-likely due to its time-series nature.)

As for the random forest-based classifier, a reasonable explanation for its underperformance is the continuous nature of our dataset. Combined with the fact that it is time series data, with price features that may not necessarily repeat, it is infeasible for the algorithm to explore the entire feature space.

Similarly, the neural-network based models may also have not ran for enough iterations to facilitate convergence to the global minima of their objective functions. More training time and perhaps a more structured dataset might be necessary to ensure convergence.

6 Conclusion

The task of classifying price sign change is non-trivial, as demonstrated by the lack of obvious class separation. While all methods achieved above 50% accuracy, the best performance was achieved by the Auto Regressive Integrated Moving Average (ARIMA) model, which is attributed to its features and suitability to time-series data. Other methods fell short due to lack of data, assumptions made by the models about the data, and non-convergence of the models. We are encouraged by the results thus far and will continue to study the dynamics of cryptocurrency markets as the field grows.

7 Contributions

Mila has contributed to: code to clean and divide the data, code to extract features, code for logistic regression, SVM, Naive Bayes, RNN, code to visualize features and results including PCA, TSNE, LDA, code for ARIMA, the proposal, milestone, and final written reports, and the poster.

Matthew has contributed to: code to clean and divide the data; all the code for our baseline and oracle experiments; code for generating statistics for Ether and other asset classes; debugging the exaggerated model accuracy issue; the poster design; the proposal, milestone, and final written reports.

Neha has contributed to:code for implementation of Naive Bayes, Support Vector Machine, Random Forest classifiers, code for Neural Network training and its hyper-parameter tuning, time series analysis for stationarity and auto-correlation in R; the proposal, milestone and the final written reports.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org. https://www.tensorflow.org/.

Charles Bovaird. 2017. Why the crypto market has appreciated more than 1,200% this year. *Forbes Magazine* [Online; posted 17-November-2017]. https://www.forbes.com/sites/cbovaird/2017/11/17/why-the-crypto-market-has-appreciated-more-than-1200-this-year.

Vitalik Buterin. 2013. Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform. https://github.com/ethereum/wiki/wiki/White-Paper.

François Chollet et al. 2015. Keras. `https://github.com/fchollet/keras`.

Etherchain. 2017. Ethereum historical prices. https://etherchain.org/api/statistics/price. [Online; accessed 02-December-2017].

Kareem Hegazy and Samuel Mumford. 2016. Comparitive automated bitcoin trading strategies. http://cs229.stanford.edu/proj2016/report/MumfordHegazy-ComparitiveAutomatedBitcoinTradingStrategies-report.pdf.

Isaac Madan, Shaurya Saluja, and Aojia Zhao. 2014. Automated bitcoin trading via machine learning algorithms.

Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

Nathaniel Popper. 2017. Business giants to announce creation of a computing system based on ethereum. *The New York Times* [Online; posted 27-February-2017]. https://nyti.ms/2lNBaWf.

Skipper Seabold and Josef Perktold. 2010. Statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*.

Ellen Sebastian, Abaho Katabarwa, and Guoxing Li. 2014. How to succeed with bitcoin without really trying.

Marshall Van Alstyne. 2014. Why bitcoin has value. *Communications of the ACM* 57(5):30–32. https://doi.org/10.1145/2594288.

Zielak. 2017. Bitcoin historical data. https://www.kaggle.com/mczielinski/bitcoin-historical-data. [Online; accessed 15-December-2017].