Predicting Bitcoin in the Short Term

Manuel Hanuch, Haobing Meng, Che-Wei Lou

December 6, 2021

Abstract

Cryptocurrencies have been a prevalent and novel investment trend in recent years. Trying to predict the price has the potential to be lucrative and worthwhile even though it is quite difficult due to extremely high volatility relative to other traditional assets. In this paper, our goal is to use several machine learning algorithms to predict the price or direction of Bitcoin. After the linear models on daily data, we transfer to binary classification problem with non-linear models on hour/minute level data, among which Random Forest has the best performance. Then we combine the model results with a long-only trading strategy that achieves a respectable Sharpe ratio of 2.19.

1 Introduction

The traditional finance theories, such as CAPM, Fama-French models, have laid the foundations for predicting asset prices. Multi-factor models like these are essentially supervised learning problems - given a set of features, what will be the return of an asset at some future point in time? Will it go up, or down? We want to see if multi-factor models can be used to predict the price of Bitcoin. This question is the concern of many investors, including giant prop-trading firms like Jane Street and other private investors. However, predicting the price of Bitcoin has unique challenges: higher volatility, 24/7 trading, and no true fundamental factors (e.g. P/E ratio, Free Cash Flow to Equity). Nonetheless, these challenges are what make this problem interesting. We divide the factors of Bitcoin into three categories: Technical factors (momentum, moving averages, and Bitcoin order flow), Fundamental factors (Gold price, Hash rate, GPU price, and returns of technology stocks (or some other basket of equities)), and Sentimental factors (Google trend of Bitcoin). There is much empirical evidence showing that using such factors could make better prediction on crypto. Our final goal is to combine these features and machine learning techniques to predict the price of Bitcoin at some future time and make appropriate trading decisions based on our model.

2 EDA

2.1 Data

Our core Bitcoin price and volume data is from CryptoDataDownload.com. The raw data contains timestamped Open, High, Low, Close, and Volume in BTC and USD at the minute level from 2017-present day. We used data from 2017 to 2020 as the training set and data from 2021 as the test set. At the minute level, we have about 2.7 million rows of data.

2.2 Features

There are hundreds of features to consider when it comes to predicting an asset price. Our approach to feature selection is to cast a wide net and refine our search depending on performance and limitations of the data we can obtain. We elect to consider three broad categories of features which we believe may be useful in forecasting the price of Bitcoin: Technical, Fundamental, and Sentimental.

2.2.1 Technical

Technical Features are derived from the price and volume of an asset. They are often used to identify price trends and gauge how other market participants may react at certain price levels. In general, there is debate about whether or not technical analysis is valid since it does not rely on any fundamental aspects of the asset and because many technical indicators seem arbitrary ("just drawing lines on charts"). However, technical indicators may prove useful in the context of using ML models to predict Bitcoin for two reasons. First, ML models will allow us to consider technical indicators systematically. Second, unlike tradition assets, Bitcoin has no intrinsic value (i.e. it is not associated with any cash flows, similar to gold), so the supply and demand for Bitcoin becomes significantly more important. We obtain popular technical features using the python library TA, outlined below. Follow the link for further information on these technical indicators.

Table 1: Technical features

Feature Name	Description
RSI	Compares the magnitude of recent gains and losses
TSI	Shows both trend direction and overbought/oversold conditions
Volatility	The standard deviation of returns over a specified time period
MA	The average price over a specified time period
MFI	Uses both price and volume to measure buying and selling pressure
OBV	Measures relationship between price and volume
ATR	The average range between the high and low price in given time interval
Aroon	Measures the time between highs and the time between lows over a time period

2.2.2 Fundamental

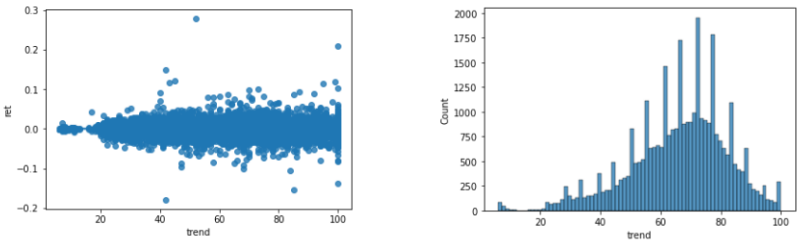
This category of features captures features that are related to bitcoin in some way other than its market dynamics.

- Bitcoin's hash rate, which is a measure of the yield from mining bitcoin relative to the cost of mining it (the higher the better). (Source: Nasdaq.com)
- The price of GPUs used to mine bitcoin. We used Nvidia stock price as a proxy variable of GPU price. (Source: Yahoo Finance)
- The price of other assets that bitcoin traders may be interested in, such as Microstrategy and the price of gold. (Source: Yahoo Finance)

A huge issue with some of these features, particularly bitcoin has rate and GPU price, is that the data is not as readily available (at least for free) as most of our other features. We were only able to find daily level data, which dramatically decreases the amount of data we can use from our other features.

2.2.3 Sentimental

Since Bitcoins price is, in theory, entirely driven by supply and demand, it could be extremely useful to gauge interest in Buying or selling bitcoin before any transactions actually happen. One way to potentially capture this interest is by looking at Google Trend data for the word 'Bitcoin'. Google trend scores work by assigning a number between 0 and 100 that represents the relative popularity of the search term in question. We were only able to obtain this data at an hourly level.
To further explore this feature, lets look at the Google Trend distribution and relationship between Google trend score at time t and bitcoin return at time $t+1$, where time intervals are by the hour, shown in Figure 1.
There does not seem to be a relationship, and this is confirmed by a Pearson correlation coefficient of about -0.008. Despite a seemingly weak relationship, we believe it is still worth including in our models. Bitcoin is known to rise and fall on the hype of retail investors, so it would prove extremely valuable to have a feature that may gauge interest in bitcoin ex-ante.



(a) Google trend on bitcoin returns (b) Google trend distribution

Figure 1: Minute Level Google Trend Data

2.2.4 Target Variable Distributions

Looking at target variable distributions helped us decide which target variable to use. our target could be the actual price of Bitcoin, the returns of bitcoin, or two classes, price closed up or down. These distributions are shown in Figure 2. The price distribution is ugly, but the returns distribution looks quite normal. For the binary class target variable, it up and down are just about even, with Bitcoin going up 51.5% of the time. Although we tried it later in this report, ultimately does not make sense to try to predict the price of Bitcoin. Predicting the price now does not translate well for predicting into the future because prices trend. To illustrate, our data contains points when bitcoin was at $10,000, but bitcoin has not been anywhere near that price in years. Trying to predict returns or direction makes much more sense.

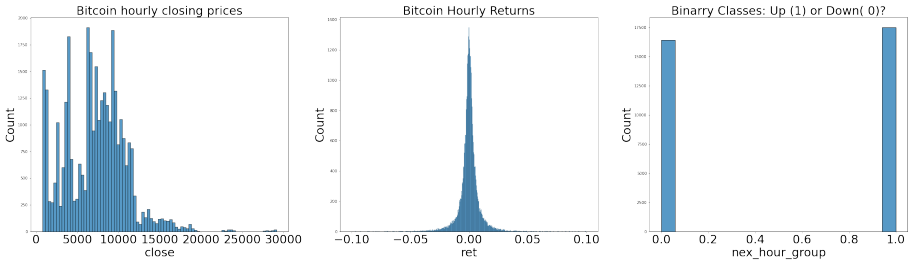


Figure 2: Target Variable Distributions

2.2.5 Feature Correlations

We continue by looking at feature correlations. Ensuring that we do not include highly correlated features in our models is important by virtue of Occam's Razor. Simpler models are generally more preferable, correlated features in general do not improve models. Furthermore, for Linear models, multicollinearity can generate results that vary greatly and are unstable. For tree based models like Random Forest, highly correlated features can mask important interactions identified by the model. Based on Figure 3, it does not seem like correlated features will pose an issue.

3 Preliminary Analysis with Linear Models

There are two major choices to make when attempting to predict the future price of bitcoin: the target variable (the actual price, return, up or down) and how far into the future to predict. Our first pass attempt was to try to predict the price 24 hours into the future since that would allow us to use all of our features, including our fundamental features. We start by using OLS, but it does not generalize well. We decide to try ℓ_2 regularization to improve generalization and reduce variance. However, This does not work very well either. We also formulate the prediction problem into a binary classification problem and simply try to predict whether the price will go up or down by using logistic regression. Results below.

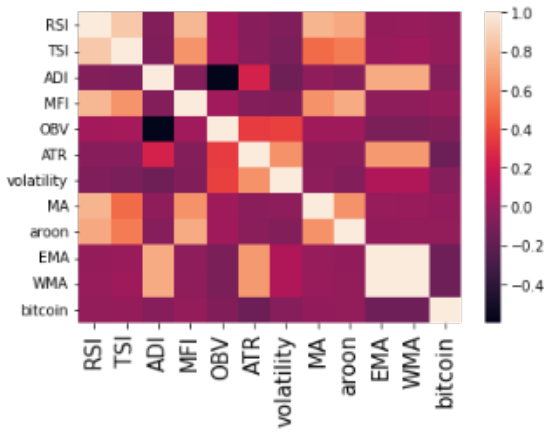


Figure 3: Daily Level Feature Correlation Heatmap

Table 2: Daily level Predictions

Model	Train MAE/accuracy	Test MAE/accuracy
OLS	4513.31	10427.05
Ridge	257.45	10436.05
Logistic	55.2%	46.9%

OLS and Ridge regression perform quite poorly when trying to predict the actual bitcoin price. This is unsurprising as prices are extremely noisy. Logistic regression does poorly in the classification version of our problem, but not as poorly as the regressors.

Recall that the final application of our models will be to inform a trading strategy. Its clear both intuitively and from the preliminary models that trying to trade based on predictions of the actual price may lead to outsized losses since it is virtually impossible to know where exactly the price is going to be in the future. A much more practical task is to try to predict whether the price will go up or down and trade based on that prediction. Furthermore, models used in classification can output actual probabilities of going up or down, which can prove extremely useful in the context of a trading strategy.

Our preliminary models try to predict the price 24 hours into the future, but they do not perform very well. We think that one reason is because we are trying to predict too far out into the future. It is possible that information captured by our features is being priced into the market by the time we want to trade. It may be worth sacrificing some of our interesting features in order to make predictions in the nearer term, such as predicting price in the next minute or hour. That way, information contained in our features will still be useful.

4 Further Analysis

4.1 More Models on Higher-Frequency Data

4.1.1 Further Analysis

As stated above, rather than predicting the daily price of Bitcoin or the movement of the daily returns of Bitcoin, we want to try more complex models on higher-frequency data. However, since the

frequencies of our features don't match, there are less features we can use when we train more complex models with higher frequency data. From the table below, if we want to use hourly data, we have to abandon fundamental features. We tried four different machine learning models including Support Vector Machines, Random Forest, Gradient Boosting and XGBoost. All of these models are powerful classification algorithms. Support Vector Machines are great at capturing non-linear relationships between label and features. Random Forest, Gradient Boosting and XGBoost are powerful ensemble methods that incorporate many weak learners and can prevent overfitting.

Table 3: Usability of features

Feature Type	Technical	Fundamental	Sentiment
Frequency	Minute-level	Daily-level	Hour-level

4.1.2 Support Vector Machines

Support Vector Machines use hinge loss as loss functions and are formulated like this. Given $x \in R^{m \times n}$, $y \in \{1, 0\}^m$, SVM solves the following problem. ϕ is the kernel function. In our project, we used the rbf kernel as rbf is better at capturing non-linear relationships and can solve harder classification problems. C is the regularization term. In our project, we found the best value of C using cross validation on the validation set. The higher C is, the less examples are allowed to be mistakenly classified and the more likely to overfit. In cross validation, the best parameter for C is $C = 0.9$.

$$min_{w,b,\xi} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \xi_i$$

,

$$s.t.\ y_i(w^T \phi(x_i) + b) \geq 1 - \xi_i$$

$$where\ \xi_i \geq 0, i = 1, \ldots, n$$

4.1.3 Random Forest

Random Forest is a powerful ensemble method that involves the technique of bootstrap aggregating, or bagging. The parameters we tuned for Random Forest are n_estimator and max_depth. Specifically, Random Forest algorithms are formulated like this:
For i = 1,2,. . . , n_estimator:

1. Sample training examples with replacement from the training set X_i, Y_i
2. Train a decision tree T_i on X_i, Y_i with the max_depth constraint

Then the label can be decided as the major vote from all n_estimator decision trees. Note that with higher value of n_estimator, it is less likely for the model to overfit. With higher value of max_depth, each single weak learner is more complicated, thus more likely to overfit. After cross validation, the best parameters are n_estimator = 33, max_depth = 10.

4.1.4 Gradient Boosting

Gradient Boosting is an ensemble method that involves the technique called boosting. Multiple models are used to predict the residuals or errors of prior models and then added together to make the final prediction. The parameters we tuned for Gradient Boosting are also n_estimator and max_depth. Specifically, Gradient Boosting algorithms are formulated like this: For m = 1,2,. . . , n_estimator:

1. Compute gradient term for each data point and fit a weak learner h_m on the dataset $\{x_i, r_{im}\}_{i=1}^n$ with with the max_depth constraint

$$r_{im} = -[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}]_{F(x)} = F_{m-1}(x)$$

2. Compute the step direction

$$\gamma_m = argmin_\gamma \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i))$$

3. Update model:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x_i)$$

Then the label can be decided using the final tree which learns the errors of prior models. As we can see from the algorithm, with higher value of n_estimator, the model fits better on the training set, thus more likely to overfit. With higher value of max_depth, each single weak learner is more complicated, thus more likely to overfit. After cross validation, the best parameters are n_estimator = 18, max_depth = 10.

4.1.5 XGBoost

XGBoost stands for Extreme Gradient Boosting. It is similar to Gradient Boosting, but uses second order of Taylor's expansion when estimating the error, rather than only the first order in Gradient Boosting. Therefore, XGBoost is more accurate when estimating the errors/residuals of previous models. A more detailed introduction of XGBoost can be found at: XGBoost: A Scalable Tree Boosting System.
Similarly to gradient boosting, the larger n_estimator is, the higher probability to overfit. The same principle also applies to max_depth. Moreover, gamma is the minimum loss reduction required to make a further partition on a leaf node of the tree. The larger gamma is, the shallower trees will be split. The last parameter is learning rate which shrinks the feature weights to make the boosting process more conservative. After cross validation, the best parameters are n_estimator = 19, max_depth = 5, gamma = 0.01, learning rate = 0.01.

4.2 Results

After cross validation on the validation set, we chose the best parameters and trained the model on the training set. We report three metrics including training accuracy, test accuracy and precision. Denote True Positive as TP, True negative as TN, Predicted Positive as PP, and Predicted Negative as PN. The metrics are calculated as follows. $Accuracy = \frac{TP+TN}{PP+PN}$, $Precision = \frac{TP}{PP}$. The reason why we also care about precision is that precision is the accuracy in regards to true predictions, so this can help us estimate how our models will behave in long only trading strategies. Long only means that only buy Bitcoin when the model predicts a positive return and never short the asset. So we attach more significance to accuracy among true predictions.
From the table below, we can see in general more complex models perform better than logistic regression on the test set. Among them, XGBoost has the highest test accuracy of 53.7%. In terms of precision, Random Forest has a remarkable accuracy of 56.1%.

Table 4: Accuracy Results

Algorithm	Frequency	Training Accuracy	Test Accuracy	Precision
Logistic Regression	Hour	50.1%	48.2%	48.9%
SVM	Hour	56.2%	50.9%	51.0%
Random Forest	Hour	68.9%	53.2%	56.1%
Gradient Boosting	Hour	75.7%	50.8%	51.9%
XGBoost	Hour	54.8%	53.7%	54.1%
Gradient Boosting	Minute	56.2%	51.4%	49.6%

5 Applying Our Models

5.1 Trading Strategy

From a practical perspective, our classification models output predictions of the movements of Bitcoin, so we could consider generating a trading strategy based on the model results. To be more specific, when we use the hourly data, if the model predicts that the next-hour Bitcoin return to be positive, we would buy and hold Bitcoin for the next hour. Similarly, if the model predicts a negative return, we would short Bitcoin for the next hour. This is called a long-short trading strategy. If we only long Bitcoins when the model predicts a positive return, this is called a long-only trading strategy.
From the last section, we can see that among all the reported results, Random Forest trained on the hourly data has the highest precision with 56.1%. What is more, most models have higher precision than overall accuracy on the test set. This means it makes more sense for us to have a long only trading strategy than long short trading strategy. Therefore, we chose the Random Forest model as an attempt to build a long only trading strategy.
A general idea to build long only trading strategies using classification models is that when the model predicts a positive return for the next hour, we would buy Bitcoin at the beginning of the next hour and sell it at the end of the next hour. Note that classification models also output the probabilities of the example belonging to which category. So we can also make use of these two probabilities. The higher the probability that the Bitcoin would have positive result is, the more confidence we would have in profiting from holding Bitcoin for the next hour. So when we build long only trading strategies, we should also consider the two probabilities.

5.2 Backtesting Our Strategy

We want to test the performance of our model as if it is in real trading. For the backtest, we have hourly data from 2021.1 to 2021.8. For each hour, we feed the features in the model and decide whether to buy and hold Bitcoin for the next hour based on the probability that the price of Bitcoin would go up. If it exceeds our probability threshold, we would buy Bitcoin at the beginning of the next hour and sell it at the end. In order to simplify the question, transaction costs and bid-ask spread are not considered. A common criterion of the profitability of a trading strategy is called Sharpe Ratio. $SharpeRatio = \frac{average\ return - risk\ free\ rate}{standard\ deviation\ of\ returns}$. A higher Sharpe Ratio means higher average return and lower volatility.

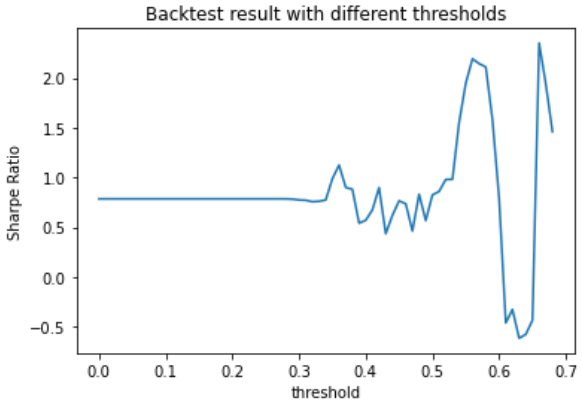
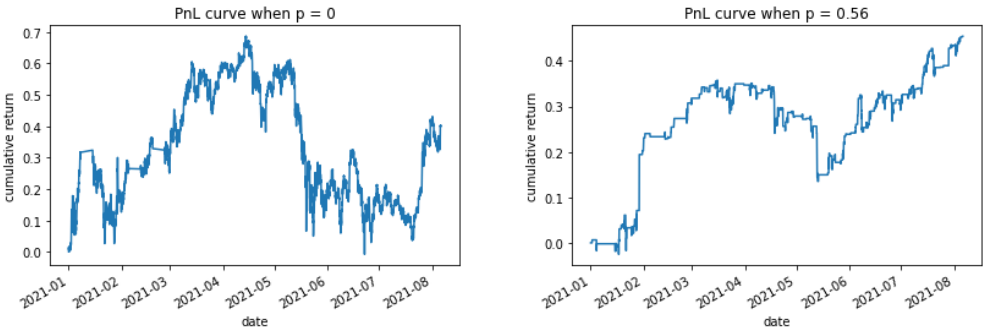


Figure 4: Backtest result with different thresholds

From figure 4, we can see if we increase the threshold, we can firstly achieve higher Sharpe Ratio but Sharpe Ratio begins to decrease when threshold is greater than 0.6. This is because when we are too strict with our longing condition, few trades are left so this is also not profitable. From figure 5, we can see that compared with p = 0, which is a simple buy and hold strategy because you would buy the asset no matter what the output is, using the Random Forest can significantly increase the performance, with higher Sharpe Ratio and lower volatility. This means that our strategy can make money in the long run with the help of machine learning models.



(a) p = 0 (b) p = 0.56

Figure 5: PnL Curves with different probability thresholds

6 Conclusion and Discussion

6.1 Conclusion

Predicting Bitcoin price is definitely not an easy thing because of high volatility and lack of fundamental data compared to stock market. In this project, we tried to use machine learning techniques with some selected features to conduct predictions. In the preliminary analysis, the results of linear regression and ridge regression on daily data are unsatisfying. Thus we turned to the binary classification problem to predict the trends of Bitcoin instead of the price. In the meantime, hour/minute level data replace the daily one to introduce more data points and further more information. The result shows that tree models perform better than others like SVM, logistic regression. Specifically, Random Forest on hourly data is the best model with highest precision (TP/PP). Finally, the long-only strategy with the results of Random Forest has an acceptable Sharpe ratio 2.19, which means we could make money in the long run if the model performance persists. It proves that the prediction of Bitcoin is practical with machine learning techniques.

6.2 Discussion

The application of machine learning techniques may introduce some negative impact to society. In this part, we discuss the weapons of math destruction and fairness. The limitations and future improvements of our model are also included.

Weapons of Math Destruction and Fairness. If the performance of the model persists and it is widely used in trading, then it will exert influence the market. In other words, if the model predicts that the price will increase, many investors will purchase bitcoin, leading to further growth in price and vice versa. Such influence is unhealthy to the markets, which might increase inequality among investors. Fairness in this project is not an important criterion. The only weapon of math destruction this model could create is one that will destruct our wallets.

Limitations and Future Improvements. To the hourly model, due to the lack of accessible data, we only used some technical factors and Google trends as features, which is comparatively naive since there are many other features related to currency market such as political factors. Thus, more features could be considered in the future. Moreover, because of high non-linearity among the data, more advanced machine learning including neural network might be utilized to capture more information in the market.