Bitcoin Algorithmic Trading

Joey Horner
Computer Science
William & Mary
Williamsburg, United States of America
jbhorner@wm.edu

***Abstract*—For this project I had models predict the direction Bitcoin prices would move 1 day, 7 days, or 30 days in the future relative to the datapoints being predicted on. This project is based on the work of Crone et al.'s ”EXPLORATION OF ALGORITHMIC TRADING STRATEGIES FOR THE BITCOIN MARKET”, in which they explored next day prediction accuracy of models on Bitcoin data. In their work they originally predicted the binary classification task of whether Bitcoin prices would go up or down the next day, but I decided to explore how well models predict 7 days and 30 days in the future relative to a given data point given Crone et al.'s accuracy scores were fairly low and predicting next day accuracy seemed like it would be a very difficult task in the context of the Bitcoin market and longer time frames seemed like they would be more appropriate.**

I. Introduction and Background

Algorithmic trading is a practice which ”involves employing process- and rules-based computational formulas for executing trades”[1] and ”uses complex mathematical models with human oversight to make decisions to trade securities”[1]. Effective Algorithms for trading in financial markets are highly valuable as machines possess many advantages over human traders. Machines are more capable of monitoring multiple financial markets simultaneously and constantly, and can perform computation and process data much faster than humans. They also are not swayed by emotion, whereas human traders often can make bad decisions as emotional responses to events in financial markets and to gaining or losing money in financial markets. For this project I did not delve into giving an algorithm a complex set of rules for when to buy or sell, rather I sought to explore how well different machine learning models could predict how Bitcoin prices would change after 1 day, 7 day and 30 day time intervals. In order to develop a strong and effective algorithm for trading in financial markets, it is essential for the algorithm to have a strong grasp of how the prices of a financial investment will change over time, as an algorithm that cannot predict how prices will move fairly well likely would not give a trader much success and would likely cause a trader to lose money, which is why my project focuses mainly on the accuracy of predicting price movements over different time intervals.

II. Summary of Contributions

The original experiment by Crone et al. focuses on next day prediction accuracy, but does not explore the prediction accuracy on other time intervals. As I believe it is essential

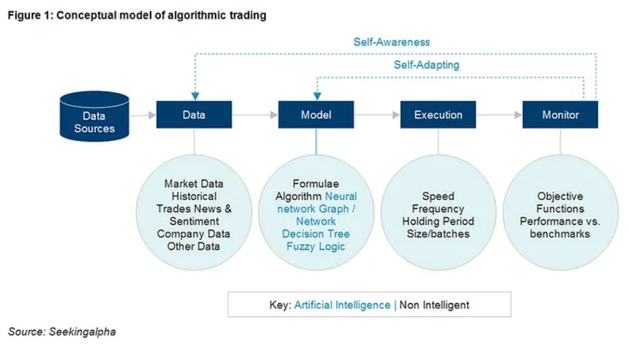


Figure 1: Conceptual model of algorithmic trading
Self-Awareness
Self-Adapting
Data Sources
Data
Model
Execution
Monitor
Market Data Historical Trades News & Sentiment Company Data Other Data
Formulae Algorithm Neural network Graph / Network Decision Tree Fuzzy Logic
Speed Frequency Holding Period Size/batches
Objective Functions Performance vs. benchmarks
Key: Artificial Intelligence | Non Intelligent
Source: Seekingalpha

Fig. 1. Diagram of a Possible Algorithmic Trading Framework and the Capabilities of Algorithmic Trading Source: [6]

to understand over what time frames models will predict most accurately on in order to consider how to make an algorithm most profitable, I decided to add the binary classification tasks of having models predict whether Bitcoin prices will go up or down 7 days and 30 days in the future relative to dates of data points in the dataset. I explored some of the models used by Crone et al., as well as some additional models to get a sense of what models are most suited to the task and the dataset.

III. Related Work

This project builds off ”EXPLORATION OF ALGORITHMIC TRADING STRATEGIES FOR THE BITCOIN MARKET” by Nathan Crone, Eoin Brophy, and Tomás Ward. I built off their code base [4] which can be found on GitHub. In their experiment they focused on having models predict whether prices would go up or down the next day, but their next day prediction accuracies were low given the task is a binary classification task.

A. Crone et al. Experiment Design

As my project builds off Crone, Brophy and Ward's work, my experiment structure is largely similar to theirs as well. In their experiment, they collect data from many sources and include a broad array of features, and they incorporated data to understand markets from across the world. For example, their Yahoo Finance commodity futures, stock indices and currency exchange rates data include stock indices from many regions like Turkey, Taiwan, Buenos Aires, and Toronto [2]. They cover many distinct investments, as their commodity futures

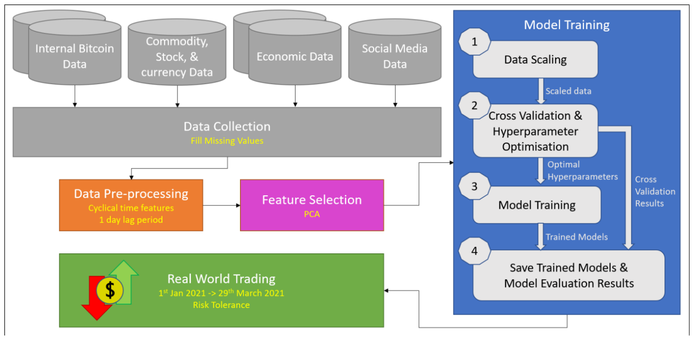


Fig. 2. Crone et al. experiment structure [2]

Metrics	Expl var (%)	SVM	XGB	RFC
Acc. (%)	80	55.9	54.7	54
	90	54.9	55	55.7
	95	**56**	53.9	55.7

Fig. 3. Crone et al. cross validation accuracy results [2]

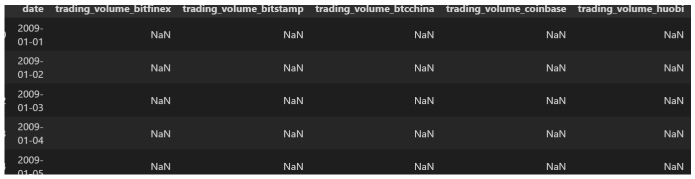
in this data include distinctly different commodities such as Crude Oil, Cotton, and Lean Hogs [2]. They also delved into the effects of social media on the market, incorporating Elon Musk tweets into their dataset to allow models to capture the effect of Elon Musk tweets on Bitcoin. Due to trouble during the data collection process, I did not collect data from all of the sources Crone, Brophy and Ward used and didn't collect data from Twitter.

B. *Crone et al. results*

When performing an evaluation on three datasets and collecting cross-validation data, Crone et al found ”SVM to perform the best, achieving an accuracy score of 56% and an F1 score of 0.716 when trained on the dataset explaining 95% of the original data's variance”[2]. With accuracy scores consistently in the mid 50s in a binary classification task, it seems essential to figure out a better way to predict the direction Bitcoin prices will move the next day, or to figure out a better time frame to predict on that may be more accurate and possibly would facilitate more profit as a result.

IV. Research Gaps

A crucial deficiency in Crone et al. is the low prediction accuracy results. As mentioned earlier and is depicted by figure 3, Support Vector Machines were considered one of the strong models in Crone et al.'s cross-validation data analysis, despite having a cross validation accuracy score of 56% in a binary classification task. Additionally, Crone et al. attempted to recreate the work of other researchers, Mudassir et al., who claimed to achieve a 60% accuracy score a next day price prediction classification task, but it was found by Crone et al. that the model that achieved this was overfitted and Crone et al. achieved an accuracy score of 54% with that model after fixing the problem of overfitting [2]. Therefore achieving good next day prediction accuracy is something that current research struggles to achieve.



date	trading_volume_bitfinex	trading_volume_bitstamp	trading_volume_btcchina	trading_volume_coinbase	trading_volume_huobi
2009-01-01	NaN	NaN	NaN	NaN	NaN
2009-01-02	NaN	NaN	NaN	NaN	NaN
2009-01-03	NaN	NaN	NaN	NaN	NaN
2009-01-04	NaN	NaN	NaN	NaN	NaN
2009-01-05	NaN	NaN	NaN	NaN	NaN

Fig. 4. Head of bitcoinity dataframe, dataframe containing features used to generate old dataset targets, filled with many NaN values

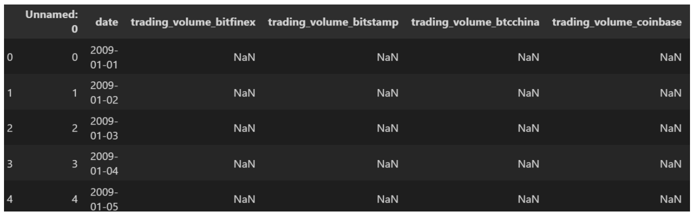
V. Dataset Description

The datasets used in this project to collect results were collected mostly by scraping data from various websites using code from the original Crone et al. experiment. I encountered quite a bit of trouble with Crone et al's code, as it is a few years old and did not seem to work with more current libraries. Because of this, I did not collect data from all of the sources from which Crone et al. collected data, notably I did not get to collecting Twitter data such as Elon Musk tweets. I realized my original dataset has a flaw which should not be overlooked- namely that targets for the dataset were generated using Bitcoin prices at various currency exchanges, which were features in the dataset. The problem with this is that the initial dataset contained a significant amount of NaN values, which I filled in through various methods such as forward filling, backward filling, and interpolation. These price features at various currency exchanges were used to generate a new feature, an average of the Bitcoin prices at these currency exchanges. The average price features were compared to average price features 1 day, 7 days, and 30 days in the future relative to a given datapoint to generate the datapoint's targets. In order to confirm the strong results of some of the models in certain prediction time frames did not occur simply due to the dataset having many target values inaccurately filled in as a result of the fact that features that may previously have had an NaN value were part of generating the targets for the dataset, I decided to retest my most successful models, XGB and AdaBoost with Decision Trees, on a new dataset in which I dropped the old target columns and the averaged_btc_price feature, and collected data from investing.com, using the open prices from that dataset to generate targets for the new dataset. The new dataset consisted of most of the features from the old dataset aside from the aforementioned averaged_btc_price feature that was dropped, a few new features from the investing.com dataset, and the newly generated targets based on the investing.com dataset's open price data which did not have NaN values.

VI. Methodology Description

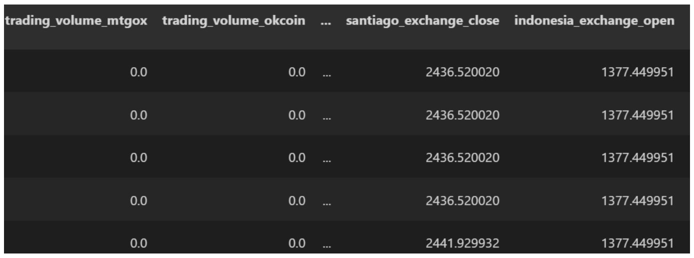
A. *Data Collection*

I first collect data for the datasets by scraping it from the following sources: Bitcoinity, BitInfoCharts, FRED Economic Data, Yahoo commodity data, Yahoo currency exchange data,



	Unnamed: 0	date	trading_volume_bitfinex	trading_volume_bitstamp	trading_volume_btcchina	trading_volume_coinbase
0	0	2009-01-01	NaN	NaN	NaN	NaN
1	1	2009-01-02	NaN	NaN	NaN	NaN
2	2	2009-01-03	NaN	NaN	NaN	NaN
3	3	2009-01-04	NaN	NaN	NaN	NaN
4	4	2009-01-05	NaN	NaN	NaN	NaN

Fig. 5. Head of combined_data_csv dataframe, filled with many NaN values



trading_volume_mtgox	trading_volume_okcoin	...	santiago_exchange_close	indonesia_exchange_open
0.0	0.0	...	2436.520020	1377.449951
0.0	0.0	...	2436.520020	1377.449951
0.0	0.0	...	2436.520020	1377.449951
0.0	0.0	...	2436.520020	1377.449951
0.0	0.0	...	2441.929932	1377.449951

Fig. 6. portion of data from CombinedNaNsRemovedDatav2.csv

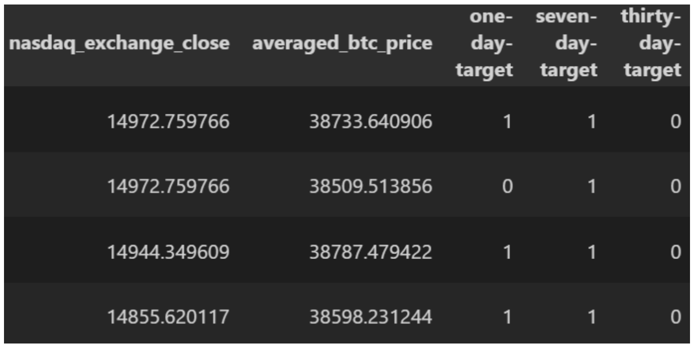
and Yahoo stock data. I merge the data from the resulting datasets into one dataset which contains datapoints corresponding to a date in the time period of January 1, 2009 to April 17, 2024. This dataset contains 5586 datapoints and a little over 1300 columns.

B. Data Processing

After having trouble with Crone et al.'s data processing code not working properly, I decided to make my own data processing file, with my data processing methods influenced by Crone et. al's methods. I take the merged dataframe, which can be found in the "Data\Scraped_data section of the code and is titled "combined_data_csv". Like Crone et al., I used interpolation to fill in missing values in the dataset that could be interpolated [2]. This did not fill in all missing values, however, so I used the df.fillna() method, forward filling and backward filling to fill in values that were still NaN after attempting interpolation. I used a similar process to Crone et al., in that I filled missing volume data with zeros as they did [2]. I adapted and reworked code from the fill_in_missing_values(df) function in Crone et al.'s data_processing_functions.py code, which originally did not use backward filling, but I found there were still missing values after the fillna and forward filling so I added backward filling. I began the process of generating targets for the dataset by dropping the features of the Bitcoin exchanges with the highest and lowest summed prices to remove potential outliers, added a new feature column of averaged Bitcoin price at the remaining currency exchanges, and used this new column to generate targets. I generated targets by initializing the targets to 0 and comparing the value of a data point's average Bitcoin price feature to that of data points 1 day, 7 days, and 30 days after it. If the future average Bitcoin price feature is less than the current one, the appropriate target based on how many days in the future the data point with the lesser averaged Bitcoin was would be given a 1 value for the current data point (go back to this). Then, the last 30 days in the dataframe are cut out as not all targets can be generated for these points. After this the new "CombinedNaNsRemovedDatav2.csv" file is generated from the resulting dataframe after the described data processing.

C. Feature Selection

I created a new file titled "Short_Feature_Select_and_drop.ipynb for feature selection, and employed methods based on Crone et al.'s code. Based



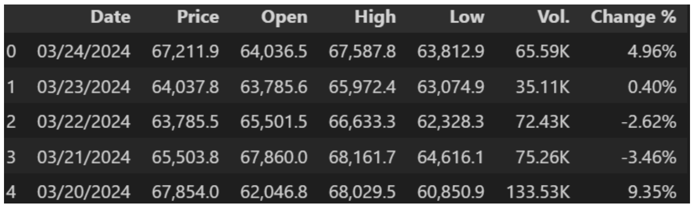
nasdaq_exchange_close	averaged_btc_price	one-day-target	seven-day-target	thirty-day-target
14972.759766	38733.640906	1	1	0
14972.759766	38509.513856	0	1	0
14944.349609	38787.479422	1	1	0
14855.620117	38598.231244	1	1	0

Fig. 7. portion of dataframe from post_feature_selection.csv , the file containing the dataset I originally used to collect results

on code from the use_rand_forest_to_get_feature_importance method from the feature_importance.py file of the original code base, I employed a random forest to get feature importances. The original code base's Feature_importance.ipynb file has an "Analyse feature importance" section which has a subsection "Summarise top 200 features", and though this subsection is empty it seemed appropriate to select the top 200 features and drop features outside of the top 200. Therefore, in my experiment after feature importances are obtained with the Random Forest Classifier, those outside the top 200 are dropped. The dataframe with features outside of the top 200 dropped is used to generate the "post_feature_selection.csv" file, which is the original dataset I used to collect experiment results after training and testing models on it.

D. Developing and Testing the Models

For this project, models were created using the sklearn [7] and xgboost [8] libraries. Selected models include Logistic Regression, Support Vector Machines, XGB, AdaBoost with Logistic Regression, AdaBoost with Support Vector Machines, and AdaBoost with Decision Trees. A Support Vector Machine run with Grid Search was performed with Grid search only for the next day prediction on the old dataset, as grid search proved very computationally and time demanding. I experimented with manually tweaking the settings of models to try to ensure models were fairly optimized. I performed testing with the models using the post_feature_selection.csv data and collected results.



	Date	Price	Open	High	Low	Vol.	Change %
0	03/24/2024	67,211.9	64,036.5	67,587.8	63,812.9	65.59K	4.96%
1	03/23/2024	64,037.8	63,785.6	65,972.4	63,074.9	35.11K	0.40%
2	03/22/2024	63,785.5	65,501.5	66,633.3	62,328.3	72.43K	-2.62%
3	03/21/2024	65,503.8	67,860.0	68,161.7	64,616.1	75.26K	-3.46%
4	03/20/2024	67,854.0	62,046.8	68,029.5	60,850.9	133.53K	9.35%

Fig. 8. portion of Bitcoin Historical Data dataframe obtained from csv file from investing.com. The data in said csv file was used to generate new targets

E. Improving the Dataset and Retesting

My original dataset had a notable flaw, namely that there were many NaN values originally in the dataset and since targets were generated using features in the dataset, likely very many price targets were generated based on the values of features that were previously NaN. As a result, I decided to improve the dataset by getting a ”Bitcoin Historical Data” dataset from investing.com [5] and using the open column, as I figured the open column was associated with a price at a certain open time so making predictions of price on the open column makes sense as you get a sense of at exactly what time models are predicting prices will be higher or lower relative to that time at the current data point, which gives a trader an idea of what time they should act rather than using a price that is expected to occur at some random point within the day. After dropping the ”averaged_btc_price” and the old targets and replacing them with the new investing.com data and newly generated targets, I turn the resulting dataframe into a new csv file. For this project, I performed testing on the new file using the two models that performed best on the old file: XGB and AdaBoost with decision trees.

VII. Model Settings

Note: This chart does not account for all of the settings of the models, as models have default settings. These are some of the settings I manually coded in.

model	Settings
SVM(grid search)	grid settings: param_grid = ’C’: [0.1, 1, 10, 100], ’gamma’: [1, 0.1, 0.01, 0.001], ’kernel’: [’rbf’, ’linear’] , grid = GridSearchCV(svm.SVC(), param_grid, refit=True, verbose=2) , model = svm.SVC(**grid.best_params_)
SVM(no grid search)	SVMModel = svm.SVC(kernel=’linear’, class_weight=’balanced’, random_state=42)
AdaBoost(Logistic Regr.)	LogisticRegressor= LogisticRegression(max_iter=5000) logrAda = AdaBoostClassifier(estimator=LogisticRegressor, n_estimators=100)
AdaBoost(SVM)	SVMModel = svm.SVC(kernel=’linear’, class_weight=’balanced’, random_state=42) svmAda = AdaBoostClassifier(estimator= SVMModel, n_estimators=100, algorithm=’SAMME’)
AdaBoost(Decision Trees)	decisionModel = DecisionTreeClassifier(max_depth=2) decisionAda = AdaBoostClassifier(estimator=decisionModel, n_estimators=100)
Logistic Regression	LogisticRegressor= LogisticRegression(max_iter=50000)
XGB	xgbModel = XGBClassifier(n_estimators=30000, early_stopping_rounds=7, learning_rate=0.6)

VIII. Experiment Results

A. Old Dataset

model	acc (1 day)	MajorityClass%(1 day)
SVM(grid search)	63.73%	62.09%
SVM(no grid search)	63.27%	62.09%
AdaBoost(Logistic Regr.)	62.64%	62.09%
AdaBoost(SVM)	40.91%	62.09%
AdaBoost(Decision Trees)	64.36%	62.09%
Logistic Regression	42.27%	62.09%
XGB	65.27%	62.09%

model	acc (7 day)	MajorityClass%(7 day)
SVM(no grid search)	66.55%	63.36%
AdaBoost(Logistic Regr.)	60.36%	63.36
AdaBoost(SVM)	39.18%	63.36%
AdaBoost(Decision Trees)	77.36%	63.36%
Logistic Regression	39.00%	63.36%
XGB	76.73%	63.36%

model	acc (30 day)	MajorityClass%(30 day)
SVM(no grid search)	74.91%	65.91%
AdaBoost(Logistic Regr.)	40.91%	65.91%
AdaBoost(SVM)	36.73%	65.91%
AdaBoost(Decision Trees)	92.49%(3 run avg)	65.91%
Logistic Regression	37.36%	65.91%
XGB	93.82%	65.91%

B. New Dataset

model	acc (1 day)	MajorityClass%(1 day)
AdaBoost(Decision Trees)	57.59%	57.49%
XGB	61.98%	57.49%

model	acc (7 day)	MajorityClass%(7 day)
AdaBoost(Decision Trees)	66.97%	59.33%
XGB	64.73%	59.33%

model	acc (30 day)	MajorityClass%(30 day)
AdaBoost(Decision Trees)	86.54%	57.90%
XGB	86.95%	57.90%

IX. Observation and Discussion

A. Old Dataset

For my old dataset with the flawed generation of targets, I found that the models overall were not able to achieve prediction accuracies very significantly higher than the percent of data points with the majority class target. My best models in this next day prediction task were the AdaBoost with Decision Trees and XGB with accuracy scores of 64.36% and 65.27% respectively. Given the fact that the most common target accounted for 62.09% of the datapoints, this means that the best model at this task, XGB, only outperformed the majority class percent by a little over 3%. This suggests that the models likely struggle to learn to distinguish between classes well in the next day task. The dataset has a data point for each day, and this likely does not facilitate good performance on the next day prediction task. The models would likely benefit from a model which collects datapoints over smaller time intervals, for example having a datapoint for every hour, to make next day predictions more accurate.

B. New Dataset

After creating a new dataset without the flawed target creation method, I evaluated the performance of the models which had previously performed the best on the old dataset: XGB and AdaBoost with Decision Trees. The distribution of data with the new dataset is different, with the old dataset having consistently more than 60% of the data points accounted for by the majority class target, while the new dataset was more evenly distributed with the majority class accounting for percentages of the datapoints in the high 50s. It is worth noting that the time frame of the new dataset is different than the old dataset, with the new dataset having dates from 07/18/2010 to 12/19/2023 while the old dataset has dates from 01/01/2009 to 01/18/2024. This most likely effects the data distribution and the percent of data points accounted for by each prediction time frame target's majority class. On the new dataset, AdaBoost with Decision Trees seems to be less able to learn to distinguish between classes well on next day predictions, with the difference between the accuracy score and majority class % being so small that it is likely not significant and it cannot be concluded that AdaBoost with Decision trees would most likely outperform the accuracy of choosing the majority class target every time if it were run on new testing data. XGB, however, outperformed the majority class by over 4%, which is even more than it outperformed the majority class in next day predictions on the old dataset. That difference of a bit over 4% is small, but could possibly provide notable increases to a model's profitability over the long term. Over the 7 day prediction time frame on the new dataset, accuracy scores outperformed the majority class by notably less than they did on the old dataset. On the old dataset the strongest model, AdaBoost with Decision Trees, outperformed the majority class by 14%. On the new dataset, it still outperformed XGB and outperformed the majority class % by a little over 7%. These results are still promising, but suggest using XGB on a next day prediction task may be more worthwhile than using AdaBoost on a 7 day prediction task, as the potential for more frequent trading offers more potential for compounding investments and investing profit from trades in future trades, as profit could be collected and invested more often. Based on the results on the new dataset, it is hard to conclude whether XGB on next day predictions or AdaBoost with Decision trees over 7 day prediction intervals is a better choice as the AdaBoost with Decision Trees 7 day prediction accuracy is only slightly better than the XGB next day prediction accuracy, while trading over 1 day intervals offers more growth potential than trading over 7 day intervals, assuming the models would actually be profitable. Both the AdaBoost with Decision Trees and XGB models performed strongly on the next 30 day prediction task on the new dataset, achieving accuracy scores over 86%. The XGB model outperformed the majority class by about 29%, suggesting the model learns important distinctions between classes on this prediction time frame. With the high level of accuracy of models over this time frame and their strongly outperforming the majority class %, this seems to be a good prediction interval for those interested in algorithmic trading to focus on, especially if they have a low risk tolerance. Given the performance of XGB and AdaBoost with Decision Trees on the 30 day prediction task on the new dataset, it is likely that these models would be able to warn traders of period where prices will decrease and occasionally help protect traders from losing money. They would also likely be able to indicate periods where prices are expected to go up 30 days afterward, allowing traders to profit off some such periods.

X. Conclusion and Future Work

The experiment results suggest a trend in which models become more capable of predicting how Bitcoin prices will change over larger time frames. Based on the results, this is not simply due to datasets becoming less evenly distributed over longer prediction time frames, as the majority class percents remain fairly consistent, while prediction accuracy begins to very strongly outperform the majority class % over the 30 day prediction time frame. Due to flaws of the original dataset I had created and used to collect testing results, the results on the new dataset are likely more realistic and reflective of how models may perform in the real world. However, the new dataset prediction results reflect the same trend as the prediction results on the old dataset do of models becoming more accurate and outperforming majority class percents by larger amounts when predicting how Bitcoin prices will differ further into the future rather than predicting how prices will change over a short time frame. The results suggest choosing the 30 day time period with XGB or AdaBoost with Decision Trees is a good choice for traders wanting an effective and accurate algorithm that does not expose them to too much risk of incorrect predictions. The 30 day time frame stands out as it offers the potential for a fairly safe and reliable algorithm, but it is difficult to say which algorithm offers the most profit potential. Shorter time frames offer more opportunity to compound money and due to this may offer more profitability even with lower prediction accuracy. With more frequent trading, assuming that trading is effective and profitable overall, traders can grow their money and reinvest profits in their next trade. Based on my next day prediction results as well as Crone et al.'s, trading based on next day predictions would likely be highly risky given the current prediction accuracies, and may not be the optimal choice without datasets and models that facilitate better prediction accuracy over this time frame. Further exploration into increasing next day prediction accuracy would be very valuable to develop effective Bitcoin trading algorithms. Exploration into next day prediction accuracy on datasets where data is collected hourly rather than having one data point for each day may help models better understand how Bitcoin prices will move over such a short time frame. Additionally, it is essential to explore Bitcoin price movement predictions as a regression task, as being able to predict whether Bitcoin price will go up or down after a certain time period with a high degree of accuracy does not guarantee an algorithm will be profitable, as it is possible that days Bitcoin price goes down may tend to have steeper price changes then days Bitcoin price goes up, and in such a case the punishment for misclassifying and choosing to buy Bitcoin when it is expected to go down in the future would be more significant than the reward for correctly deciding to buy Bitcoin when it will go up after a time frame elapses. Such details should be accounted for to make algorithms as effective as possible. To expand upon this project and turn this work into a more useful and effective algorithm, one could broaden the scope of this project by adding more evaluation metrics and targets and developing a more complex rule system to decide when it is appropriate to buy or sell. I originally wanted to add more technical indicators to the dataset as Crone, Brophy, and Ward suggested for future work[2], but didn't really delve into this after trouble during the data collection phase, so adding new technical indicators to the dataset would be useful in the future. I also wanted to explore Reinforcement Learning models in this project, but didn't do this for this project so it would be useful to incorporate Reinforcement Learning models in the future. If adding more complex models to this project in the future, it may become useful to explore cloud computing options to deal with higher time and computing resource requirements the new models may have.

References

[1] Chen, James. "Algorithmic Trading: Definition, How It Works, Pros & Cons." Edited by Gordon Scott and Hans Daniel Jasperson, Investopedia, 2024, www.investopedia.com/terms/a/algorithmictrading.asp.

[2] Crone, Nathan, Brophy, Eoin, and Ward, Tomàs. "EXPLORATION OF ALGORITHMIC TRADING STRATEGIES FOR THE BITCOIN MARKET." Paperswithcode, 2021, arxiv.org/pdf/2110.14936v1.pdf.

[3] Github Copilot

[4] Crone1. "Crone1/Bitcoin-Algorithmic-Trading-Paper." GitHub, github.com/Crone1/Bitcoin-Algorithmic-Trading-Paper. Accessed 2024.

[5] "Bitcoin Historical Data." Investing.Com, www.investing.com/crypto/bitcoin/historical-data. Accessed 2024.

[6] Zach, Christopher. "Can Algo Trading Transform the Rhythm of Capital Markets?" Medium, Techiefinance, 17 Sept. 2019, medium.com/techiebaba/can-algo-trading-transform-the-rhythm-of-capital-markets-ad9e1192e170.

[7] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011

[8] xgboost developers. "XGBoost Documentation." Dmlc XGBoost, xgboost.readthedocs.io/en/stable/. Accessed 2024. https://xgboost.readthedocs.io/en/stable/