Using Voting Regression to Predict Bitcoin Price

Karl Schmidt
karl.schmidt@colorado.edu
University of Colorado Boulder
Boulder, CO, USA

ABSTRACT

Over the last decade, Bitcoin has seen a repeated cycle in its long term price fluctuations. These cycles typically span over an approximate four year period where in the first year, Bitcoin experiences a cycle low in the first year followed by a cycle peak in the fourth year. While many studies have utilized various machine learning models to predict the price of Bitcoin, most tend to use data related to on-chain metrics, gold price, sentiment data, stock market data, and Bitcoin network activity. While some of these features may be good to include in predicting Bitcoin's price, macro economic data, such as the M2 money supply and ISM Manufacturing PMI, were not included. Additionally, during the investigtoin of related work, an ensemble of more than a couple of ML supervised models was not found. This paper focuses on both including macro economic data as features into the dataset and utilizing a Voting Regressor which ensembles AdaBoost, Random Forest, and Support Vector Machine models.

Keywords: Bitcoin, Price Prediction, Classification, Machine Learning

1 INTRODUCTION

Bitcoin has garnered popular and traditional finance support through recent years. Since its initial proposal in 2008, it has proven to be a volatile asset that cycles peaks and lows over a four year cycle. It should be noted that, in order to build a likely significantly more accurate model, several various methods should be analyzed which includes additional machine learning models beyond the regressive or classification models taught within this course. Additionally, quality and complete cycle data should be used to fully train models if the intent is to use them in actual trading. As such, it should be noted that this is a simple project intended to learn supervised learning methods and should not be taken as financial advice. It is likely that the model developed in this report will be far from worthy for making any sort of Bitcoin investment decisions.

2 RELATED WORK

Several studies have explored Bitcoin price prediction using machine learning and deep learning techniques. Chen (2023) [1] investigated various machine learning models for Bitcoin price forecasting, emphasizing their effectiveness in financial risk management. Ji et al. (2019) conducted a comparative study of deep learning approaches, demonstrating the advantages of neural networks over traditional models in capturing complex market patterns. Similarly, Chen et al. (2020) [2] proposed a novel method for sample dimension engineering to enhance the performance of machine learning models in Bitcoin price prediction, highlighting the impact of data preprocessing on forecasting accuracy. These studies collectively contribute to the growing body of research on cryptocurrency price prediction, showcasing different methodologies to improve predictive accuracy.

3 PROPOSED WORK

3.1 Problem Statement

The purpose of this project is to:

- Optimize AdaBoost, Random Forest, and SVM in predicting Bitcoin's price.
- Utilize these optimized models to construct and ensemble to improve the accuracy and decrease the variance of predictions.
- Identify key features influencing Bitcoin price movements.
- Evaluate the models based on accuracy and generalization to unseen data.
- Provide a foundation for further research in cryptocurrency price forecasting.

This project will limit its scope to hyperparameter tuning for each individual regression model, apply a simple mean aggregation of these models and perform an analysis of its performance by looking at mean absolute error (MAE), mean squared error (MSE), and the coefficient of determination or R-squared (R2). Given that this is a jupyter notebook, flow of the document may be broken up by code. The intent is to have cells with 'chunks' of code that are easier to read. As such, all description will be presented before the code blocks with the analysis writeup after the code blocks.

3.2 Data Preparation

For building successful models, data mining is imperative to its success by ensuring data is collected, clean, and preprocessed for use with each of the models before use. Data will be collected from various locations and most likely in the form of comma separated value (CSV) data files. Given data will come from different sources, it's time delta, size, shape, and time range were expected to be different. Data cleaning to concatenate these datasets into a single data source that includes features needed for training and testing the model shall be performed. The cleaned dataset can then be stored in a parquet columnar based file.

Once the dataset is cleaned, a exploration of the data needs to be performed before using the dataset with the models. This is where dataprocessing comes into affect. The exploratory data analysis (EDA) can be informative for understanding the correlation between features in order to find potential multi-colinearity issues within the data. This stage can also include adding data transformations to the final dataset. It is understood within the blockchain community, that Bitcoin is a highly volatile asset. This volatility could lead to overpredicting to the price noise. While some methods can be utilized to reduce overfitting, adding a trailing average of previous price data can normalize direction of price movement.

3.3 Price Prediction Methodology

Using AdaBoost, Random Forest, and Support Vector Machine models, a voting regressor will be used to ensemble these models into a final price prediction. The general methodology for tuning these individual models will be to utilize K-Fold and GridSearch to optimize a given set of parameters.

The implementation of these models will be done using the scikit-learn Python library.

3.3.1 AdaBoost. AdaBoost works by sequentially building models that correct the errors made by previous models. Initially, a model is trained on the dataset, and subsequent models are built to address the misclassifications of the prior models. This iterative process continues until errors are minimized, resulting in a robust classifier. AdaBoost enhances prediction accuracy by combining multiple weak learners into a strong learner, creating a powerful ensemble model. As an ensemble learning method, AdaBoost is particularly effective in improving the performance of machine learning models by refining predictions through adaptive boosting [8]. From the scikit-learn library, the AdaBoostRegressor class will be used.

3.3.2 Random Forest. Random Forest is an ensemble learning technique that improves upon bagging by introducing a small but crucial modification to decorrelate the decision trees. Like bagging, Random Forest constructs multiple decision trees using bootstrapped training samples. However, when building each tree, the algorithm randomly selects a subset of predictors (m) at each split, instead of considering all available predictors (p). This random selection ensures that no single predictor dominates the splits, which reduces the correlation between the trees. As a result, the predictions from the trees are more independent, leading to a greater reduction in variance and more reliable results. In situations with many correlated predictors, using a smaller subset of predictors (m) at each split helps improve model performance by preventing overfitting and increasing the diversity of the trees. This technique has been particularly effective in high-dimensional data, such as predicting cancer types based on gene expression data, where the model shows improved prediction accuracy over bagging [5]. RandomForestRegressor will be used.

3.3.3 Support Vector Machine. The support vector machine (SVM) is an advanced version of the maximal margin classifier, a simple and intuitive binary classifier that aims to separate two classes using a linear boundary. However, the maximal margin classifier is not suitable for most real-world datasets, as it requires the classes to be perfectly separable by a linear boundary. To address this limitation, the support vector classifier (SVC) was introduced as an extension, allowing for some flexibility by permitting misclassifications to improve robustness and generalizability. The support vector machine further extends the SVC by handling non-linear class boundaries, making it applicable to a wider range of datasets. Additionally, SVMs can be adapted for multi-class classification, and they share strong connections with other statistical methods like logistic regression, providing a powerful framework for binary and multi-class classification tasks [5]. For the SVM implementation, the SVR class will be used with a radial base function kernel.

3.3.4 Voting Regressor. A Voting Regressor is an ensemble learning technique that combines predictions from multiple regression models to improve the accuracy and robustness of predictions. It aggregates the outputs of various base models, typically through averaging, to create a stronger final prediction. This helps in reducing the risk of overfitting compared to relying on a single model. The Voting Regressor can accommodate different types of regressors (such as Linear Regression, Random Forest, and Support Vector Regression) and allows for weighted combinations of their predictions. By assigning different weights to each model based on its performance, the Voting Regressor can further enhance prediction accuracy. This ensemble method is effective in improving the generalization of machine learning models by leveraging the strengths of multiple base regressors [3].

4 EVALUATION

Utilizing a 80/15/15 training, test, and validation set, each model will be tuned by mean squared error (MSE) which helps interpret how our model performance from a variance perspective, mean absolute error (MAE) for evaluating precision, and the Coefficient of Determination (R2) to help understand how well the models outcomes describe the feature set. Cross Validation will also be used to show how well each individual model is performing. This could be a good indicator as each model will be tuned with scikit-learn's KFold cross-validation class with 5 folds each. In addition, each model will be given a set of hyperparameter values that will be run through the GridSearchCV for selecting the best combination. The best hyperparameters will then be evaluated based on MSE, MAE, and R2.

4.1 Data Preparation

4.2 Data Collection

Data was collected from multiple sources: the U.S. Dollar Index (DXY), Bitcoin data, and the M2 Money Supply (M2SL). Each dataset is retrieved from CSV files located in the data directory, and the date range for the data collection is restricted between **January 1, 2017** and **September 2, 2023**.

4.2.1 Bitcoin Data. Harris' Bitcoin network dataset [4] is a Kaggle dataset that contains several CSV files pertaining to Bitcoin on-chain, network, and price data. This project only uses the daily Bitcoin daily data which includes calculated values such as the Fear and Greed index, dervied from metrics within the Bitcoin network.

- **Granularity:** Daily (from 2009 to 2023)
- **Size:** 12 columns, 666 rows
- **Data Columns:**
  - datetime
  - **market_price_usd**: The Bitcoin price in USD
  - **total_supply**: The total supply of Bitcoin tokens
  - **market_cap_usd**: The total market capitalization of Bitcoin in USD
  - **realised_cap_usd**: The value of the Bitcoin network based on the price at which each coin last moved on-chain
  - **nupl**: Net Unrealized Profit/Loss, calculated as the market capitalization minus realized capitalization, divided by market capitalization

- **coin_days_destroyed**: A measure of the number of coins transacted and how long they remained unspent
- **active_addresses**: The number of daily active Bitcoin addresses
- **fear_greed_value**: A sentiment indicator measuring market emotions, ranging from extreme fear (024) to extreme greed (75100)
- **fear_greed_category**: The category of the Fear and Greed Index
- **lightning_nodes**: The number of nodes running on the Lightning Network
- **lightening_capacity_usd**: The total value of Bitcoin (in USD) locked in the Lightening Network

4.2.2 M2 Money Supply. The M2 Money Supply (M2S) is a broad measure of the total amount of dollars circulating in the economy. The Fred [6] dataset is included because the supply of money is often correlated with the appreciation of various assets, including Bitcoin, within macroeconomic trends. Typically, Bitcoin's price experiences high and low peaks around a four-year cycle, which may correlate with M2S, especially considering the effects of quantitative easing (QE) and quantitative tightening (QT) implemented by the Federal Reserve. QE generally occurs when the U.S. government extends the debt limit, often in response to the need to pay off national debt, which leads to inflation. The Federal Reserve typically responds to this inflation by removing money from circulation.

- **Granularity:** Monthly (from 1960 to 2025)
- **Size:** 2 columns, 770 rows
- **Data Columns:**
  - **observation_date**: The month for each M2S data reading
  - **M2SL**: The total money supply.

4.2.3 ISM Manufacturing PMI. The ISM Manufacturing PMI (Purchasing Managers' Index) is a key economic indicator that measures the economic health of the manufacturing sector in a country, specifically the United States. It is released monthly by the Institute for Supply Management (ISM). The U.S. Dollary Index (DXY) dataset [7] contains daily index value data.

- **Granularity:** Daily (from 2014 to 2023)
- **Size:** 4 columns, 2502 rows
- **Data Columns:**
  - **open**: The daily opening index value
  - **high**: The daily high index value
  - **low**: The daily low index value
  - **close**: The daily close index value

4.2.4 Data Cleaning. The data collection and preprocessing workflow involves multiple steps to ensure that the datasets are correctly formatted, synchronized, and cleaned before further analysis. The following steps summarize the data cleaning process:

Each dataset includes a date column, which is converted to a `datetime` format using `pandas.to_datetime()`. This ensures uniformity in the way dates are represented across the datasets. The specific format for each dataset was handled (e.g., MM/DD/YYYY format for the DXY data), and any invalid or malformed date entries were coerced to `NaT` (Not a Time).

After parsing the dates, each dataset was filtered to ensure that only data within the specified date range (from **January 1, 2017** to **September 2, 2023**) is retained. This ensures that the analysis is consistent and relevant to the chosen time period.

The date column names were standardized across all datasets. The `Date` column in the DXY dataset was renamed to `date`, and similarly, the `datetime` column in the Bitcoin dataset and the `observation_date` column in the M2SL dataset were renamed to `date` to facilitate merging and avoid potential issues with inconsistent column names.

The next step involved merging the individual datasets into a single dataframe. A date range dataframe was created to ensure that the data spans the full range of dates from **January 1, 2021** to **September 2, 2023**. This date range was used as a reference to merge all three datasets (DXY, Bitcoin, and M2SL) on the common `date` column. The merging strategy used was a **left join**, ensuring that no dates were missed in the final dataset.

After the merge, the datasets were forward-filled to replace any missing values with the last valid entry, using `ffill()`. This is particularly useful in time-series data where missing values are often caused by non-reporting on certain days. Subsequently, any remaining rows with missing values were removed using `dropna()`, ensuring the final dataset does not contain incomplete rows.

The result of the merging and cleaning process is a single, consolidated dataframe with synchronized, cleaned data from the DXY, Bitcoin, and M2SL datasets, ready for further analysis. The data is now aligned by date, with missing entries appropriately handled and all columns renamed consistently.

The cleaned data is now ready for subsequent analysis, ensuring the integrity of the datasets and their alignment across all data sources.

4.3 Exploratory Data Analysis

In this section, we provide a summary of the key statistics for several important features related to the market data, realised capitalization, network metrics, and macroeconomic factors. The tables below present the descriptive statistics for each feature, such as the count, mean, standard deviation, and percentiles.

Table 1 shows the summary statistics for the market data, including the market price, total supply, and market capitalization. The data spans across 2040 observations.

Table 2 continues with additional metrics related to the market, such as the realised capitalization, NUPL (Net Unrealized Profit and Loss), and coin days destroyed.

Table 3 provides an overview of the lightning network statistics, including the number of lightning nodes and their corresponding capacity.

Finally, Table 4 presents summary statistics for the on-chain metrics and macroeconomic variables, including active addresses, fear & greed index, DXY, and M2SL.

Figure **??** displays the price data of Bitcoin over time. This figure illustrates the fluctuations in Bitcoin's market price, which reflects the volatility characteristic of cryptocurrency markets. From this plot, we can observe periods of significant growth, as well as sharp declines, which are typical of the high-risk, high-reward nature of digital asset trading.

Figure 2 presents the trailing average of Bitcoin's price, which smooths out short-term fluctuations to provide a clearer picture of

Statistic	date	Market Price	Total Supply	Market Cap.
count	2040.00	2040.00	2040.00	2040.00
mean	.2f	21179.31	18388710.96	396949466128.52
min	.2f	3236.72	16839677.00	56399873657.69
25%	.2f	8115.03	17782030.25	142190217496.98
50%	.2f	16210.51	18545873.50	304886044680.10
75%	.2f	30464.50	19008497.00	589148933349.78
max	.2f	67492.00	19473739.00	1273420650592.18
std	nan	16228.93	745281.68	309020614744.35

Table 1: Summary Statistics: Market Data

Statistic	Realised Cap	NUPL	Coin Days Destroyed
count	2040.00	2040.00	2040.00
mean	239097911053.34	0.30	10637974.16
min	76222559953.09	-0.43	1313012.07
25%	91240764924.07	0.20	4581202.29
50%	129846243343.62	0.35	6724452.11
75%	392646325879.52	0.46	11456585.84
max	468557597880.42	0.75	397370509.67
std	153945860647.49	0.25	16222122.31

Table 2: Summary Statistics: Market Data continued

Statistic	Lightening Nodes	Lightening Capacity
count	2040.00	2040.00
mean	9904.04	57127638.51
min	0.00	0.00
25%	4569.00	6193676.69
50%	7716.00	16873160.32
75%	17087.00	111777070.67
max	20707.00	216470149.70
std	6541.46	60498632.08

Table 3: Summary Statistics: Lightning Network

Statistic	Active Addresses	Fear & Greed	DXY	M2SL
count	2040.00	2040.00	2040.00	2040.00
mean	857633.10	43.21	97.49	18178.32
min	410502.00	5.00	88.59	13907.30
25%	728823.00	25.00	93.58	14782.90
50%	874279.00	40.00	96.90	18972.90
75%	974841.00	57.00	100.48	21117.10
max	1368431.00	95.00	114.10	21723.20
std	169318.22	21.53	5.17	3005.79

Table 4: Summary Statistics: On-chain Metrics and Macro

the overall trend. By calculating the average over a specified time window, the figure helps in understanding the general direction of Bitcoin's market price, indicating potential periods of stability or volatility. This figure is crucial for analyzing market sentiment and trends.

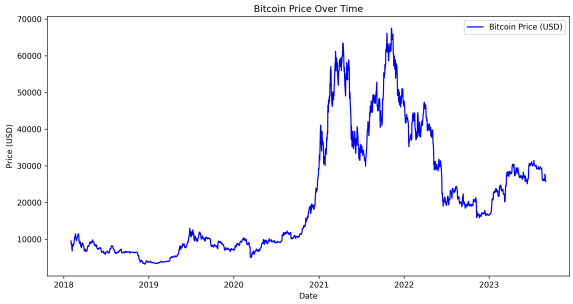

Figure 1: Bitcoin Price Data.

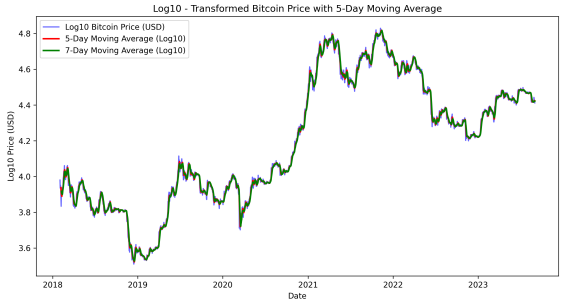

Figure 2: Price Trailing Average.

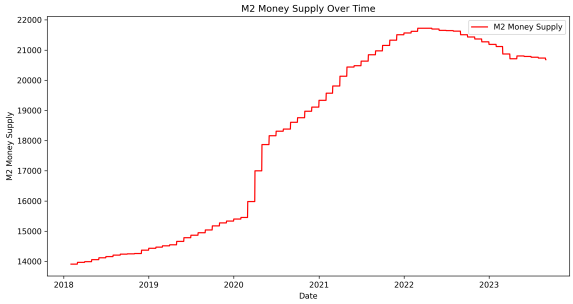

Figure 3: M2 Money Supply.

Figure 3 depicts the M2 Money Supply (M2SL) over time, showing the total supply of money in the economy, including currency, checking, and savings accounts, and other near money assets. The M2SL data is often used as a macroeconomic indicator to assess inflationary pressure or potential economic growth. This figure provides context on how changes in money supply correlate with the Bitcoin market, especially during times of economic uncertainty.

Figure 4 illustrates the Fear and Greed index, which measures the sentiment of investors in the market. The index ranges from extreme fear to extreme greed, offering insights into how market participants feel about Bitcoin's future prospects. Understanding this sentiment can aid in predicting potential price movements, as

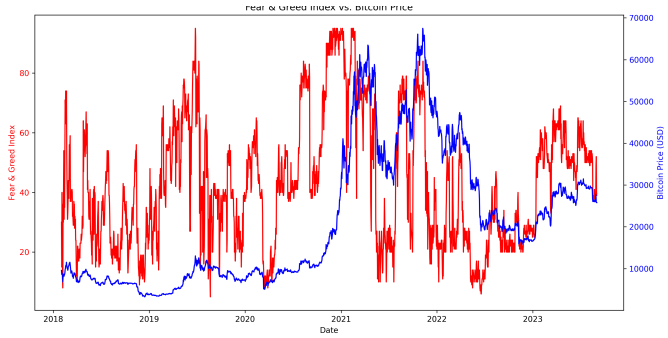

Figure 4: Fear and Greed.

extreme levels of fear or greed often signal market corrections or bubbles.

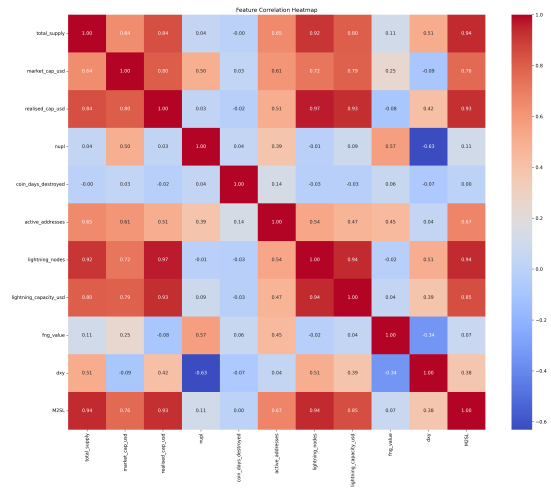

Figure 5: Correlation Matrix.

Figure 5 shows the correlation matrix between various factors influencing the Bitcoin market. The matrix visually represents how different variables such as Bitcoin price, M2SL, Fear and Greed index, and other metrics are related to one another. High correlations suggest that certain factors move together, which can provide important insights into the driving forces behind Bitcoin's price changes. The analysis of the correlation matrix is key in identifying significant predictors for Bitcoin price forecasting.

This collection of figures together provides a comprehensive view of the factors influencing Bitcoin price movements, sentiment, and economic context. Each figure serves as a piece of the puzzle in understanding how various macroeconomic and market-specific variables interact and impact Bitcoin's market performance.

4.4 Data Preprocessing

The data preprocessing process is a crucial step in preparing the raw dataset for further analysis and model training. The preprocessing pipeline is implemented within the `DataPreprocessing` class, which performs several key operations on the provided dataset.

The class initializes with a DataFrame (`data`) containing the raw dataset and two instances of `StandardScaler` from the `sklearn.preprocessing` module, which are used for normalizing the features and target variable.

4.4.1 Loading and Saving Data. To ensure smooth handling of data during the preprocessing, the class includes methods for saving and loading the processed data. The `save` method writes the processed data to a Parquet file format, which offers efficient storage. The `load` method reads in previously processed data from the same file format. The file is stored in the `../data/processed_data.parquet` directory.

4.4.2 Data Transformation. The core of the preprocessing happens in the `process` method, where the dataset undergoes several transformations:

(1) **Normalization of Dates:** The `date` column is first converted to represent the number of days' since the earliest date in the dataset. This is achieved by subtracting the minimum date (`reference_date`) from each date in the `date` column and converting the result to days. This transformation makes it easier to handle the temporal aspect of the data in machine learning models, which typically perform better with numerical input rather than date objects.
(2) **Separation of Features and Target Variable:** The dataset is split into features (denoted as `x_df`) and the target variable (denoted as `y_df`). The target variable in this case is `market_price_usd`, which represents the price of Bitcoin, while all other columns become the feature set.
(3) **Scaling of Features:** To ensure that all features are on a similar scale, which is critical for many machine learning algorithms, the features (`x_df`) are standardized using the `StandardScaler`. This scales the data such that the mean is 0 and the standard deviation is 1. The scaled data is stored in `df_x_scaled`.
(4) **Scaling of Target Variable:** Similarly, the target variable (`y_df`) is also scaled using a separate instance of `StandardScaler` to ensure that it is on the same scale as the features. The scaled target data is stored in `df_y_scaled`.

Finally, the scaled feature and target datasets are saved as instance variables `df_x_scaled` and `df_y_scaled`, which can be used for further analysis or fed into machine learning models.

This preprocessing pipeline prepares the data by transforming it into a format suitable for statistical modeling and machine learning, ensuring consistency and normalization across features and target variables.

4.5 Model Tuning

Model tuning is a critical step in optimizing machine learning models to achieve the best predictive performance. This section details the tuning process applied to AdaBoost, Random Forest, and

Support Vector Machine (SVM) regressors. The tuning process involves hyperparameter optimization using cross-validation and grid search, with model evaluation based on Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared (R2) scores.

4.5.1 Hyperparameter Tuning Strategy. Each model undergoes hyperparameter tuning using a systematic search over a predefined parameter grid. The models are evaluated using K-Fold cross-validation to ensure robustness and avoid overfitting.

Cross-Validation Setup

- **K-Fold Splitting:** A 3-fold cross-validation (K=3) is used
- **Scoring Metrics:**
  - **MAE:** Measures average absolute errors
  - **MSE:** Penalizes larger errors more significantly
  - **R2:** Measures goodness of fit
- **Parallel Processing:** The n_jobs parameter is set to self.nproc to utilize multiple CPU cores

4.5.2 AdaBoost Regressor Tuning. AdaBoost tuning involves selecting the best combination of base estimator depth, learning rate, and number of estimators.

Hyperparameters:

- n_estimators: [1, 200, 400, 600, 800]
- learning_rate: [0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1.0]
- loss: ['linear', 'square', 'exponential']

Optimization Process:

- A DecisionTreeRegressor is used as the base estimator with varying max_depth values
- A GridSearchCV is conducted to find the best combination of hyperparameters
- The best model is selected based on the lowest MSE

4.5.3 Random Forest Regressor Tuning. Random Forest tuning focuses on optimizing tree depth, number of estimators, feature selection, and leaf node constraints.

Hyperparameters:

- n_estimators: [1, 100, 200, 300, ..., 900]
- max_depth: [1, 3, 5]
- max_features: ['linear', 'square', 'exponential']
- max_leaf_nodes: [2, 3, 5]

Optimization Process:

- GridSearchCV is applied with K-Fold cross-validation
- The best model is selected based on MSE performance on test data

4.5.4 Support Vector Machine Regressor Tuning. AdaBoost tuning involves selecting the best combination of base estimator depth, learning rate, and number of estimators.

Hyperparameters:

- C: [1, 200, 400, 600, 800]
- kernel: [0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1.0]
- gamma: ['linear', 'square', 'exponential']

Optimization Process:

- GridSearchCV is performed across different kernel functions
- The best-performing model is selected based on the lowest MSE

4.6 Model Evaluation and Selection

Once the hyperparameter tuning is complete, the best models from each category are evaluated on test data. The final step involves selecting the best model for deployment by comparing performance metrics across all models. The tuned models are then combined into a weighted Voting Regressor, where individual models are weighted based on their inverse MSE values.

Hyperparameter tuning significantly improves model performance by optimizing key parameters. The AdaBoost, Random Forest, and SVM regressors undergo rigorous tuning and evaluation, ensuring that the final model ensemble provides the best possible predictions.

The performance of the tuned model was assessed using key evaluation metrics. The results indicate significant prediction errors, as reflected by the following values:

- **Mean Absolute Error (MAE):** 34,028.35
- **Mean Squared Error (MSE):** 1,174,315,995.49
- **R^2 Score:** -4,491,936,135.06

These results suggest that the model struggles to accurately capture the underlying patterns in the data. The high MAE and MSE values indicate substantial deviations between the predicted and actual values, while the negative R^2 score suggests that the model performs worse than a simple mean predictor. This highlights the need for further improvements, such as feature engineering, alternative model architectures, or additional data preprocessing techniques to enhance prediction accuracy.

From the results shown in Figure 6, the model is able to somewhat predict the general price movement; however, struggles at predicting the large swings in Bitcoin's price. The model may be underfitting the data too much resulting in highly inaccurat price predictions.

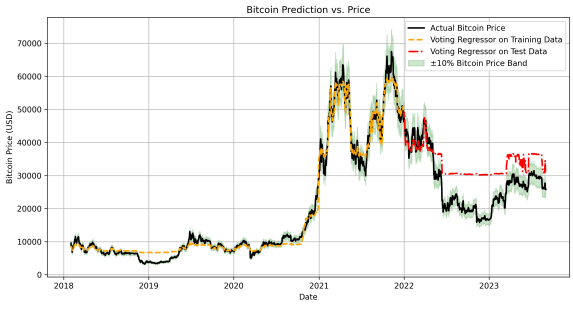

Figure 6: Bitcoin Price Data.

5 DISCUSSION

The model's performance indicates significant predictive errors, with a high Mean Absolute Error (MAE) and Mean Squared Error (MSE), alongside a negative R^2 score. These results suggest that

the model fails to capture meaningful relationships in the data, potentially due to feature limitations, inadequate hyperparameter tuning, or underlying data inconsistencies. The complexity of Bitcoin price prediction, influenced by various macroeconomic and sentiment-driven factors, may require a more sophisticated modeling approach. Incorporating additional features, such as global financial indicators or real-time sentiment analysis, could improve prediction accuracy.

6 CONCLUSION

The current model's poor performance highlights the need for further refinement, particularly in feature selection, model choice, and hyperparameter tuning. To enhance predictive accuracy, future efforts should focus on exploring additional data transformations, trying alternative regression models like XGBoost or deep learning, and utilizing more advanced optimization techniques such as Bayesian search. Additionally, error analysis and data augmentation may help mitigate inconsistencies and improve model generalization. By iterating on these improvements, model could become more reliable for Bitcoin price forecasting.

REFERENCES

[1] Junwei Chen. 2023. Analysis of Bitcoin Price Prediction Using Machine Learning. *J. Risk Financial Manag.* 16, 1 (2023), 51. https://doi.org/10.3390/jrfm16010051

[2] Zheshi Chen, Chunhong Li, and Wenjun Sun. 2020. Bitcoin price prediction using machine learning: An approach to sample dimension engineering. *J. Comput. Appl. Math.* 365 (February 2020), 112395. https://doi.org/10.1016/j.cam.2019.112395

[3] Data and Beyond. 2021. *Voting Regressor: Intuition and Implementation.* Medium. https://medium.com/data-and-beyond/voting-regressor-intuition-and-implementation-0359771b5204#:~:text=The%20Voting%20Regressor%20is%20a,better%20generalization%20to%20new%20data. Accessed: 2025-02-25.

[4] Alex Harris. 2024. *Bitcoin Network On-Chain Blockahin Data.* Kaggle. https://www.kaggle.com/datasets/aleexharris/bitcoin-network-on-chain-blockchain-data?select=blockchain_dot_com_daily_data.csv Accessed: 2025-02-25.

[5] Gareth James, Daniela Witten, and Trevor Hastie. 2023. *An Introduction to Statistical Learning: with Applications in Python* (3rd ed.). Springer, New York.

[6] Unknown. 2025. *M2SL.* Federal Reserve Bank of Saint Louis. https://fred.stlouisfed.org/series/M2SL Accessed: 2025-02-25.

[7] Unknown. 2025. *U.S. Dollar Index (DXY).* MarketWatch. https://www.marketwatch.com/investing/index/dxy/download-data?startDate=1/1/2024&endDate=12/31/2024 Accessed: 2025-02-25.

[8] Analytics Vidhya. 2021. *AdaBoost Algorithm: A Complete Guide for Beginners.* Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/09/adaboost-algorithm-a-complete-guide-for-beginners/#:~:text=AdaBoost%20algorithm%2C%20short%20for%20Adaptive,assigned%20to%20incorrectly%20classified%20instances. Accessed: 2025-02-25.