# Classification of Bitcoin Price Data: An Experimental Analysis

Deepak Binkam
Towson University

***Abstract*—This paper presents an experimental analysis of Bitcoin price data classification using multiple machine learning models. Exploratory data analysis (EDA) and feature engineering techniques were used to incorporate time based data by extracting date-based features and creating lag features. The study compares the performance of Logistic Regression, Decision Trees, and k-Nearest Neighbors before and after using temporal features. The results demonstrate that incorporating date-related features improves the performance of certain models, particularly Logistic Regression, and highlights the importance of preserving the temporal order in financial time series data.**

***Index Terms*—Bitcoin, Classification, Time Series, Feature Engineering, Logistic Regression, Decision Trees, k-Nearest Neighbors.**

## I. Introduction

Predicting Bitcoin prices is a challenging task because of the volatility and noise of financial time series data. This study presents a classification approach to determine whether daily Bitcoin prices move upward or not, by assessing if the closing price exceeds the opening price or not. A big challenge is capturing the time based features within the data. To address this issue, date-based features (such as day-of-week and month), lag features, and rolling averages are incorporated into the modeling process. Section II describes the dataset, and Section III details the methodology, exploratory data analysis, and feature engineering. Section IV presents the experimental results, and Section V concludes the paper.

## II. Dataset Description

The dataset employed in this study is a historical collection of Bitcoin price data sourced from Kaggle. It provides a comprehensive record of daily market activity and includes the following key variables:

- **Date:** This variable represents the specific trading day. It is important for capturing time-based trends and seasonality patterns in Bitcoin prices. The Date variable is converted from string format to a datetime object.
- **Open:** The opening price is the first price of Bitcoin on a given day. It is the baseline for daily price changes and is used with other data to assess the momentum of the price.
- **High:** The highest recorded price of Bitcoin during the trading day.
- **Low:** The lowest recorded price during the day.
- **Close:** The closing price is the last traded price at the end of the trading day. It is used with Open to determine the daily price movement, and is the basis for the target variable Movement.
- **Volume:** This variable is the total number of Bitcoin units traded during the day.
- **MarketCap:** Market capitalization is calculated by multiplying the closing price by the supply of Bitcoin in circulation.

The target variable, *Movement*, is defined as a value of 1 if the closing price is higher than the opening price (indicating a positive movement that day), and 0 if it is not higher. Before analysis, data cleaning was performed, which included the conversion of date strings to datetime objects, the removal of extra characters (such as commas and hyphens) from numerical fields, and sorting the data in chronological order to preserve the temporal structure essential for time series analysis.

### A. Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) was conducted to gain a thorough understanding of the dataset's structure and underlying patterns. The following steps were done:

**Data Cleaning and Preparation:**

- **Date Conversion and Sorting:** The Date column was converted from a string into a datetime object. Sorting the data set chronologically is essential to preserve the effect of time on the data.
- **Numeric Data Cleaning:** Numeric fields such as *Volume* and *MarketCap* were cleaned by removing extra characters and replacing hyphens in the data with zero.

**Descriptive Statistics:** Summary statistics (mean, median, standard deviation, minimum and maximum) were calculated for key price-related variables (*Open, High, Low, Close*). These statistics provided the initial insight into the central tendencies and dispersion of the data.

TABLE I
Training Data Statistics

| Statistic | Date | Open | High | Low | Close | Volume | MarketCap | Movement |
|---|---|---|---|---|---|---|---|---|
| Count | 1556 | 1556 | 1556 | 1556 | 1556 | 1556 | 1556 | 1556 |
| Mean | 2015-06-14 12:00:00 | 582.63 | 597.99 | 567.85 | 584.24 | $1.25 \times 10^8$ | $8.69 \times 10^9$ | 0.54 |
| Min | 2013-04-28 00:00:00 | 68.50 | 74.56 | 65.53 | 68.43 | 0 | $7.79 \times 10^8$ | 0.00 |
| 25% | 2014-05-21 18:00:00 | 254.29 | 260.33 | 248.84 | 254.32 | $1.46 \times 10^7$ | $3.60 \times 10^9$ | 0.00 |
| 50% | 2015-06-14 12:00:00 | 438.60 | 447.56 | 430.57 | 438.86 | $3.28 \times 10^7$ | $6.39 \times 10^9$ | 1.00 |
| 75% | 2016-07-07 06:00:00 | 662.44 | 674.53 | 646.74 | 663.40 | $7.77 \times 10^7$ | $9.90 \times 10^9$ | 1.00 |
| Max | 2017-07-31 00:00:00 | 2953.22 | 2999.91 | 2840.53 | 2958.11 | $2.57 \times 10^9$ | $4.84 \times 10^{10}$ | 1.00 |
| Std | — | 523.14 | 542.99 | 505.88 | 525.90 | $3.03 \times 10^8$ | $8.71 \times 10^9$ | 0.50 |

TABLE II
BTC DATA STATISTICS

| **Statistic** | **Date** | **Open** | **High** | **Low** | **Close** | **Adj Close** | **Volume** | **Movement** |
|---|---|---|---|---|---|---|---|---|
| Count | 2747 | 2747 | 2747 | 2747 | 2747 | 2747 | 2747 | 2747 |
| Mean | 2018-06-21 00:00:00 | 11668.60 | 11981.03 | 11325.60 | 11682.89 | 11682.89 | $1.48 \times 10^{10}$ | 0.54 |
| Min | 2014-09-17 00:00:00 | 176.90 | 211.73 | 171.51 | 178.10 | 178.10 | $5.91 \times 10^{6}$ | 0.00 |
| 25% | 2016-08-03 12:00:00 | 609.12 | 611.89 | 606.31 | 609.23 | 609.23 | $8.16 \times 10^{7}$ | 0.00 |
| 50% | 2018-06-21 00:00:00 | 6371.85 | 6500.87 | 6285.63 | 6376.71 | 6376.71 | $5.23 \times 10^{9}$ | 1.00 |
| 75% | 2020-05-07 12:00:00 | 10728.27 | 10992.47 | 10412.89 | 10755.40 | 10755.40 | $2.50 \times 10^{10}$ | 1.00 |
| Max | 2022-03-25 00:00:00 | 67549.73 | 68789.63 | 66382.06 | 67566.83 | 67566.83 | $3.51 \times 10^{11}$ | 1.00 |
| Std | — | 16323.68 | 16759.57 | 15825.58 | 16330.19 | 16330.19 | $1.99 \times 10^{10}$ | 0.50 |

## Visualization Techniques:

- **Histograms:** Histograms were generated to visualize the distribution of price variables. These plots revealed the skewness and variability in the data.

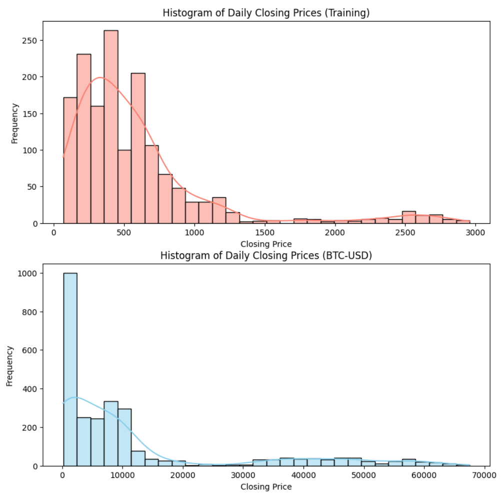

Fig. 1. Histogram from data shown

- **Box Plots:** Box plots were employed to identify outliers in the *Close* price and other numeric features. This step was crucial to understand the range of the data and detect any anomalies that might negatively affect training the models.

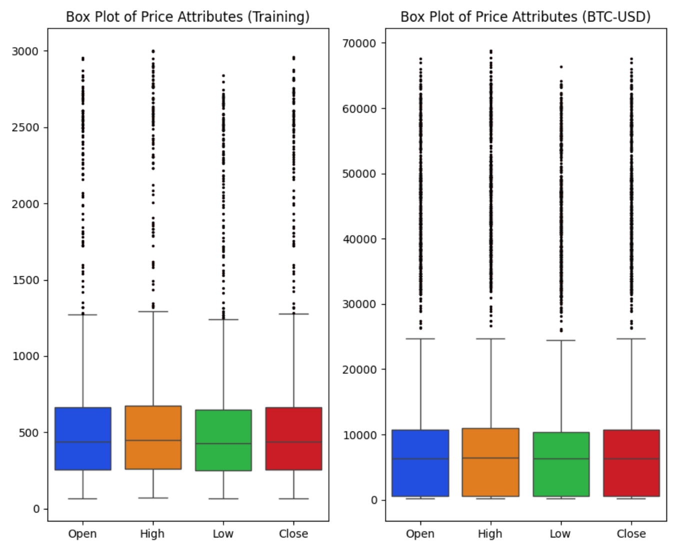

Fig. 2. Boxplot of Columns

- **Scatter Plots:** Scatter plots, particularly those comparing *Open* and *Close* prices, were used to investigate the relationships between different price measures. This helped in checking for any potential linear/non-linear trends.

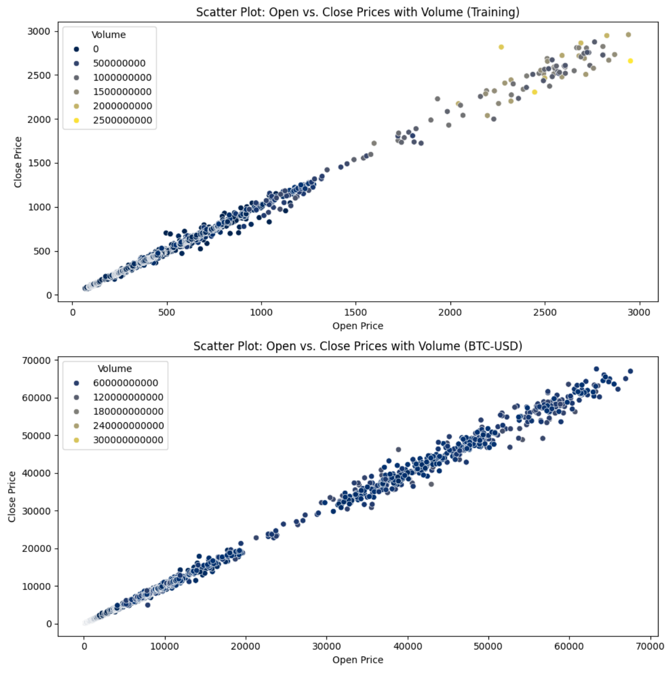

Fig. 3. Scatter of Open vs. Close

- **Correlation Heatmaps:** Heatmaps were created to check the strength of the relationships of the variables. The correlation analysis found strong correlations among price related variables.

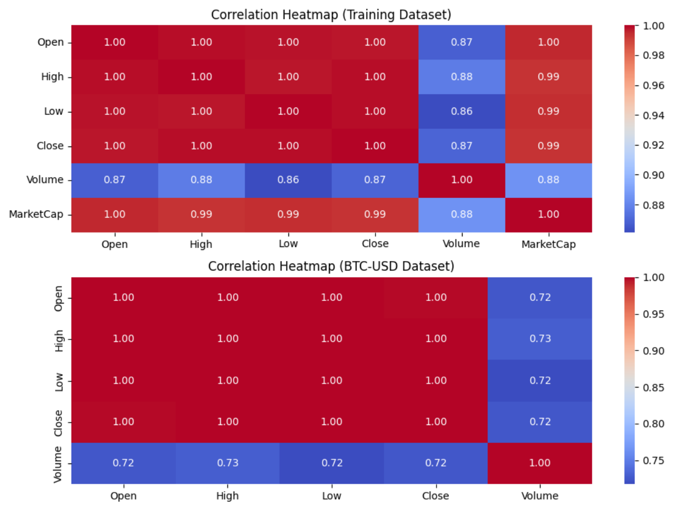

Fig. 4. Correlation map of columns

## Target Variable Distribution:

The distribution of the target variable, *Movement*, was examined using count plots. This step confirmed that the binary target was well-defined.

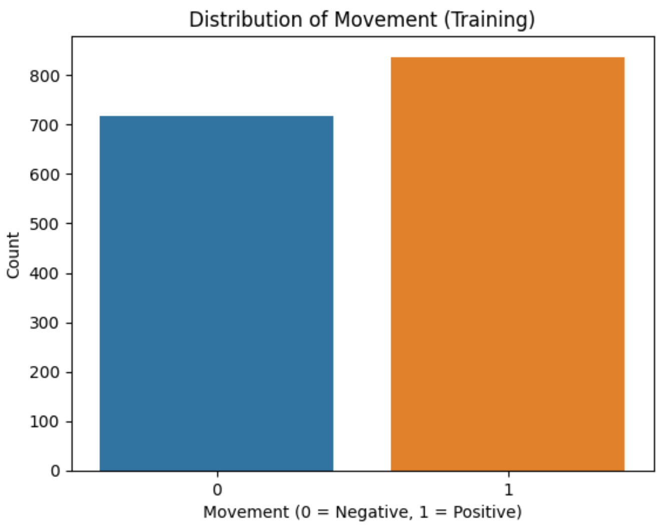

Fig. 5. Count of Movement Variable

The EDA phase provided valuable insights into the dataset, including its central tendencies, variability, and any outliers. This formed the foundation for further feature engineering, especially in extracting time related features and designing an ordered split for time series analysis.

## *B. Feature Engineering*

To capture the temporal dependencies in time series data, additional features were created from the dataset. This can help provide the models with more explicit information about the time-based patterns in Bitcoin prices. The feature engineering process included:

- **Date Feature Extraction:** New features such as `DayOfWeek` and `Month` were derived from the Date column. These features help capture weekly and monthly trends that may affect Bitcoin price movements.
- **Lag Features:** A lag feature, `Lag`, was calculated to represent the closing price of the previous day. This feature provided the model with closer context, which is important for predictions.
- **Rolling Window Features:** A 3-day rolling mean, `RollingMean`, was calculated to provide more context, by recording short-term fluctuations in the closing price. This feature helps in capturing changes over a bigger window of time than lag.

After engineering these features, rows containing missing values (resulting from the lag and rolling window calculations) were removed, and the dataset was sorted in chronological order again to preserve the time structure.

## *C. Experimental Design*

Given the time series nature of the data, an ordered (chronological) split was performed, where the first 80% of the data (representing earlier dates) were used for training and the last 20% (later dates) were reserved for testing. This more closely simulates a real-world scenario and preserves the time order of the data.

Three classification models were evaluated on this ordered split, each chosen for its distinct approach to handling the data:

- **Logistic Regression:** Logistic Regression is a parametric linear model that uses a sigmoid function to estimate the probability of an upward price movement (Movement = 1). It models the relationship between the input features and the log-odds of the target variable being 1. Because it is linear, its performance is highly influenced by the quality/relevance of the features. The addition of time features (such as `DayOfWeek`, `Month`, `Lag`, and `RollingMean`) provides meaningful context about seasonality.
- **Decision Tree:** Decision Trees are non-linear models that separate the data by recursively applying splits on the features. This model captures the non-linear relationships among variables. The increased dimensions from adding the time based features may complicate the splitting. In this experiment, the Decision Tree had moderate performance with basic features, but its performance dropped after including time features—suggesting that more work is necessary to fully use these new variables.
- **k-Nearest Neighbors (k-NN):** k-NN is a learning method that classifies a new observation based on the majority class among its k nearest neighbors, which is determined by a distance metric. The k-NN model uses distance calculations, so it will be sensitive to the scale of features; hence, feature scaling using standardization is important. The addition of time features increases dimensionality, and if these features are not relevant enough to the distance metric, they can interfere with more relevant features. In the current experiment, k-NN showed reduced performance after adding time features, which may be because of this increased dimensionality or the k value.

All models were evaluated using standard performance metrics including accuracy, precision, recall, and F1-score. A comparison of the models' performance before and after the incorporation of time features highlights the varying impacts on each model. Logistic Regression showed an improvement, whereas the Decision Tree and k-NN models had worse performance.

## III. Results

The experimental results are summarized in Table III.

TABLE III
PERFORMANCE COMPARISON BEFORE AND AFTER DATE FEATURE ENGINEERING

| **Model** | **Accuracy** | **Precision** | **Recall** | **F1-Score** | **Comments** |
|---|---|---|---|---|---|
| **Before Date Features** | | | | | |
| Logistic Regression | 54% | 0.54 | 1.00 | 0.70 | Low recall for class 0. |
| Decision Tree | 62.5% | 0.66 | 0.62 | 0.51 | Better performance and predictions. |
| k-NN | 57% | 0.59 | 0.65 | 0.46 | Middle performing model. |
| **After Date Features** | | | | | |
| Logistic Regression | 66% | 0.65 | 0.96 | 0.78 | Improved performance but class 0 recall is still low. |
| Decision Tree | 52% | 0.69 | 0.40 | 0.51 | Accuracy is worse. |
| k-NN | 50% | 0.69 | 0.34 | 0.46 | Model has become worse. |

The experimental results reveal a clear difference in how different classification models respond to the addition of time based features. Logistic Regression had an improvement, with its accuracy increasing from 54% to 66% and an F1-score increasing from 0.70 to 0.78. This suggests that the addition of date-based features (e.g., DayOfWeek, Month) and other features (e.g., Lag, RollingMean) provided the linear model with better context to separate the classes. However, there is a persistently low recall for class 0, indicating that the model can identify upwards movements of the data (class 1) but not class 0.

In contrast, the Decision Tree and k-NN models did not show improvements with the new features. The accuracy of the Decision Tree dropped from 62.5% to 52%, and its F1-score remained at 0.51. Similarly, k-NN's performance declined, with accuracy decreasing from 57% to 50% and no change in the F1-score. These results imply that the additional temporal features may have increased the complexity of the data in a way that the models can't effectively solve without further tweaking or feature selection. The increased complexity may have reduced the power of more relevant features.

## IV. CONCLUSION

Incorporating date-based features and lag features helped improve the performance of Logistic Regression, demonstrating the importance of time based features in financial data. However, the other models did not show a similar improvement, suggesting that further optimizations are necessary. Overall, this study emphasizes the importance of more specific feature engineering in enhancing model performance for time series data.