Predicting Bitcoin Prices Using Machine Learning

Hrachya Baghdasaryan

January 17, 2025

Abstract

Bitcoin (BTC) is a highly volatile financial instrument. Accurate prediction of its price movements is essential for traders and investors. This report explores BTC-USD data through machine learning techniques, analyzing historical trends and correlations. The primary goal is to derive insights that could inform predictive models, focusing on statistical relationships and potential forecasting methodologies.

Introduction

Cryptocurrency trading has emerged as a pivotal domain in finance. However, its high volatility poses significant challenges for prediction. This study aims to analyze historical BTC-USD data[1], identify key patterns, and leverage machine learning techniques for improved predictive accuracy. The methodology includes exploratory data analysis, statistical evaluation, and consideration of modeling approaches.

1. Data Collection and Preprocessing

1.1 Overview of BTC-USD Data

Data for this study comprises historical price and volume metrics for Bitcoin(BTC) traded in USD. The dataset spans several years and includes features such as open, high, low, close prices, and trading volume.

Price	Adj Close	Close	High	Low
Ticker	BTC-USD	BTC-USD	BTC-USD	BTC-USD
count	2862.000000	2862.000000	2862.000000	2862.000000
mean	25550.535741	25550.535741	26095.477765	24919.800709
std	22156.334796	22156.334796	22608.725653	21631.648787
min	937.520020	937.520020	975.760986	903.713013
25%	7679.916992	7679.916992	7888.550171	7502.744873
50%	18544.335938	18544.335938	18928.085938	17850.670898
75%	40368.338867	40368.338867	41447.698242	39352.166992
max	106140.601562	106140.601562	108268.445312	105291.734375

Price	Open	Volume
Ticker	BTC-USD	BTC-USD
count	2862.000000	2.862000e+03
mean	25520.218833	2.437248e+10
std	22127.623231	2.006050e+10
min	936.539978	1.341270e+08
25%	7677.806519	9.378169e+09
50%	18452.333984	2.130695e+10
75%	40232.836914	3.371901e+10
max	106147.296875	3.509679e+11

A detailed statistical summary of key metrics

1.2 Feature Engineering

To enhance the predictive power of the dataset, several new features were engineered:

1.Moving Averages(MA)[2]:

- 7-day MA:
- 14-day MA: is the average closing price over the past 14 days.
- 30-day MA: uses a 30-day rolling window.

$$\mathrm{MA}_7 = \frac{1}{7}\sum_{i=t-6}^{t} \mathrm{Close}_i \quad \mathrm{MA}_{14} = \frac{1}{14}\sum_{i=t-13}^{t} \mathrm{Close}_i \quad \mathrm{MA}_{30} = \frac{1}{30}\sum_{i=t-29}^{t} \mathrm{Close}_i$$

2.Volatility Features[3]:

- Volatility over 7, 14, and 30 days was computed as the standard deviation pf the daily returns in the respective periods:

$$\text{Volatility}_n = \sqrt{\frac{1}{n}\sum_{i=t-n+1}^{t}(\text{Return}_i - \overline{\text{Return}})^2} \quad \text{Return}_t = \frac{\text{Close}_t - \text{Close}_{t-1}}{\text{Close}_{t-1}} \times 100$$

3.Lagged Features[4]:

- Previous values of the closing price were included as lagged features to capture temporal dependencies:

$$\text{Close\_Lag\_n} = \text{Close}_{t-n}$$

4.Daily Returns

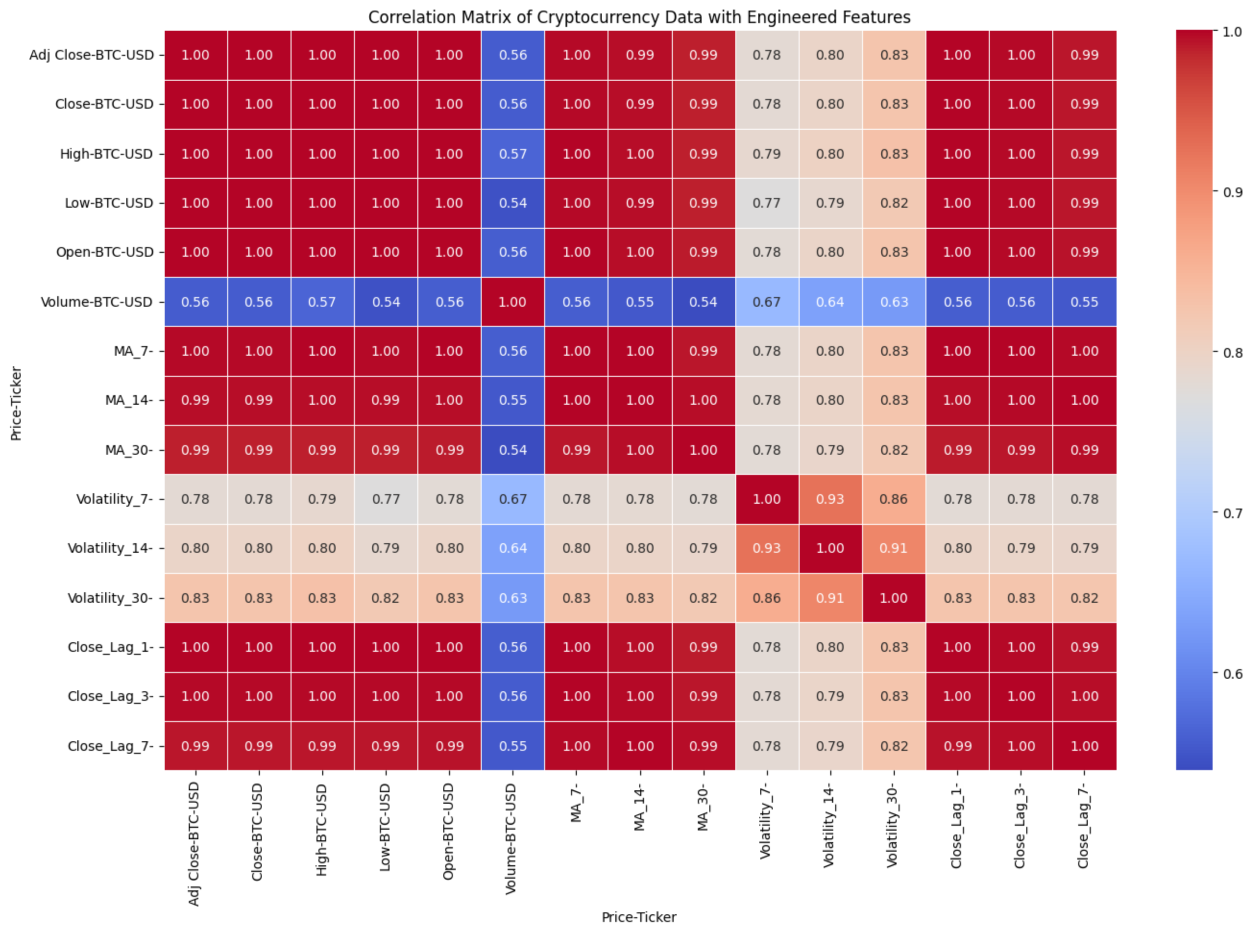
- Calculated as the percentage change in closing price:

5.Normalized Features:

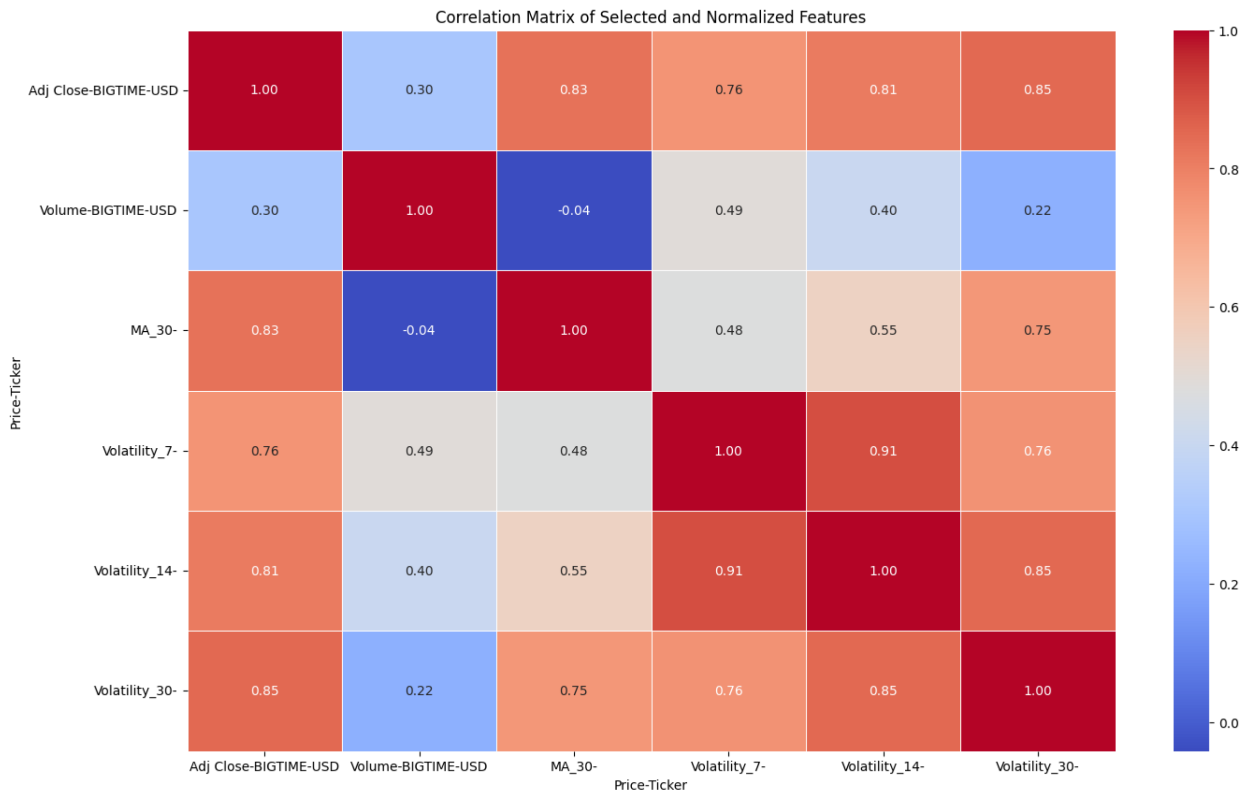
- To prepare the data for machine learning models, all features were normalized to the range using min-max scaler:

1.3 Correlation Analysis

Below is a heatmap displaying correlations between all features, including engineered ones:



For normalized and selected features:



2. Machine Learning Techniques

2.1 Target and Selected Features

The target feature for this study is the Adjusted Close price(Adj Close), representing the value to be predicted.

The features used for training were selected based on their correlation with the target and their predictive importance. These include:

- Moving Average(MA_30)
- Volatility metrics(Volatility_7, Volatility_14, Volatility_30)
- Trading volume(Volume-BTC-USD)

2.2 Temporal Sequence Splitting

To prepare the data for the reservoir network, the dataset was divided into temporal sequances:

- Sequence Length: 30 days(each sequence contains 30 consecutive time steps).
- Sliding Window: A sliding window approach was used to generate overlapping sequences for training and testing.

2.3 Reservoir Computing Network

Training Process

- The input sequences were fed into the reservoir[5], generating a high-dimensional state vector
- The state vector captures the dynamic behavior of the input features over time
- Ridge regression was used to train the output weights, mapping the reservoir states to the target feature.

2.4 Model Evaluation

The model was evaluated using the following metrics:

- MSE
- R^2 score

2.5 Hyperparameters Optimization using Grid Search

To optimize the performance of the reservoir computing model, a grid search was conducted over key hyperparameters:

1.Hyperparameters Tuned:

- Reservoir size:[300, 500, 700, 1000]
- Leak rate:[0.1, 0.2, 0.3, 0.5]
- Ridge regularization: [1e-4, 1e-3, 1e-2]
- Density:[0.05, 0.1, 0.2]

2.Procedure:

- For each combination of hyperparameters, the reservoir and ridge regression models were initialized and trained on the training set.
- The validation set was used to compute the MSE
- The configuration yielding the lowest validation MSE was recorded as the best.

3.Chalanges:

- Due the limited computational resources, the grid search was not exhaustive, and not all configurations could be tested.

3. Results and Discussion

3.1 Model Performance

Table comparing the performance metrics of the reservoir computing model:

Metric	Training Set	Validation Set	Testing Set
MSE	0.0106	0.0353	0.16687
R^2	0.604	0.6040	-5.9984

4. Conclusion

The reservoir computing model applied in this study has shown limited predictive performance so far.

5. Future work

1. Perform a more comprehensive grid search with sufficient computational resources.
2. Explore alternative architecture or hybrid models to improve performance.
3. Incorporate external factors to enhance predictive capabilities.

Despite the current limitations, this approach lays the groundwork for further exploration and optimization of reservoir computing models for financial time series prediction.

References

1. Historical BTC-USD Dataset(https://finance.yahoo.com/qu)
2. Moving Averages
3. Volatility Features
4. Lagged Features
5. Reservoir Neural Network