Bitcoin Price Prediction: A Comparison of MLP, XGBoost, and LSTM Models

Eric Schmid

November 19, 2024

Abstract

This paper explores and compares three machine learning models—Multi-Layer Perceptron (MLP), XGBoost, and Long Short-Term Memory (LSTM)—for predicting Bitcoin price movements using technical indicators such as Simple Moving Average (SMA), Exponential Moving Average (EMA), and Relative Strength Index (RSI). We report the performance of these models using two key metrics: Area Under the Curve (AUC) and test accuracy. XGBoost outperformed the other models with an AUC of 0.5600 and accuracy of 62.26%, while MLP and LSTM showed competitive results. The study highlights the computational efficiency of XGBoost and the significant resource requirements of LSTM.

GitHub Repository: https://github.com/ericschmid-uchicago/depaul-final-project

1 Introduction

Bitcoin, the first cryptocurrency, is known for its volatility, creating both risks and opportunities in the financial markets. Predicting price movements is crucial for algorithmic trading. This study investigates the application of three machine learning models for predicting Bitcoin prices: Multi-Layer Perceptron (MLP), XGBoost, and Long Short-Term Memory (LSTM). Using technical indicators like the Simple Moving Average (SMA), Exponential Moving Average (EMA), and Relative Strength Index (RSI), we aim to forecast price movements.

1.1 Dataset

The dataset used in this project consists of high-frequency historical data for the Bitcoin (BTC/USDT) trading pair, collected from the Binance.US exchange. The data spans from January 1, 2021, to December 31, 2021, at a 1-minute interval. Each data point includes the following features: opening price, high price, low price, closing price, volume, and additional trading-related metrics such as the number of trades and quote asset volume. For feature engineering, we applied several technical indicators, including Simple Moving Average (SMA), Exponential Moving Average (EMA), Relative Strength Index (RSI), and Bollinger Bands. The target variable is a binary classification, predicting whether the return (log difference between consecutive closing prices) for the next minute will be positive or negative. After computing the technical indicators and applying the necessary preprocessing steps, such as removing missing values, the dataset was split into an 80% training set and a 20% test set to evaluate model performance.

1.2 Technical Indicators

Technical indicators are essential tools in financial analysis. The SMA is the average of the closing prices over a specific period, smoothing out short-term fluctuations:

$$SMA_t = \frac{1}{n} \sum_{i=0}^{n-1} Price_{t-i}$$

The EMA gives more weight to recent prices, providing a faster reaction to price changes:

$$EMA_t = \alpha \times \text{Price}_t + (1 - \alpha) \times EMA_{t-1}$$

where $\alpha = \frac{2}{n+1}$ is the smoothing factor. The Relative Strength Index (RSI) is another technical indicator used to measure the magnitude of price changes, calculated as:

$$RSI = 100 - \frac{100}{1 + \frac{\text{Average Gain}}{\text{Average Loss}}}$$

The RSI measures the magnitude of recent price changes to assess whether an asset is overbought or oversold.

2 Mathematical Models

We implemented three machine learning models—MLP, XGBoost, and LSTM—to evaluate their performance in predicting the Bitcoin price movement.

2.1 Multi-Layer Perceptron (MLP)

MLP is a feedforward neural network where information flows from input to output through one or more hidden layers. The function of each neuron is:

$$y = \sigma\left(\sum_{i=1}^{n} w_i x_i + b\right)$$

where w_i are the weights, x_i are the inputs, b is the bias, and σ is the activation function (LeakyReLU in this case). The MLP uses backpropagation to adjust the weights and minimize the loss function. For our experiments, we used 100 epochs and applied TimeSeriesSplit for validation.

2.2 XGBoost

XGBoost is an ensemble learning technique based on gradient boosting. It builds multiple weak learners (decision trees) sequentially, where each subsequent tree corrects the errors of its predecessor. The objective function minimizes the loss function:

$$obj(\theta) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{j=1}^{T} \Omega(f_j)$$

where $l(y_i, \hat{y}_i)$ is the loss function, and $\Omega(f_j)$ is the regularization term to penalize the loss function.

New trees are added iteratively to the model in XGBoost using an additive strategy, where each new tree is trained to correct the residual error of the previous ones:

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i)$$

Each new tree f_t is added based on minimizing a second-order Taylor expansion of the error, leading to the following objective function for each new tree:

$$obj^{(t)} = \sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t)$$

where g_i and h_i are the gradient and Hessian of the loss function with respect to the prediction.

Source: XGBoost Documentation.

2.3 Long Short-Term Memory (LSTM)

LSTMs are a type of recurrent neural network (RNN) that can capture sequential dependencies and handle long-term dependencies. Each LSTM cell contains several gates that control the flow of information: input, forget, and output gates. The core computations at each time step t are:

1. Input Gate:

$$i_t = \sigma(W_{ii}x_t + W_{hi}h_{t-1} + b_{ii} + b_{hi})$$

This gate controls how much of the new input should influence the cell state.

2. Forget Gate:

$$f_t = \sigma(W_{if}x_t + W_{hf}h_{t-1} + b_{if} + b_{hf})$$

This gate determines how much of the previous memory should be forgotten.

3. Output Gate:

$$o_t = \sigma(W_{io}x_t + W_{ho}h_{t-1} + b_{io} + b_{ho})$$

This controls the amount of information to be passed to the next hidden state.

4. Cell State Update:

$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W_{ig}x_t + W_{hg}h_{t-1} + b_{ig} + b_{hg})$$

5. Hidden State Update:

$$h_t = o_t \odot \tanh(c_t)$$

The hidden state h_t is used in the subsequent time step.

Source: PyTorch LSTM Documentation.

3 Literature Review

The application of machine learning models in financial prediction has evolved significantly in recent years. Traditional technical indicators like SMA, EMA, and RSI have been widely used for trend detection (Murphy, 1999). However, with advancements in machine learning, these indicators are now integrated into more sophisticated models to improve predictive accuracy.

Recent studies, such as Parente et al. (2024), have explored the use of deep learning models for cryptocurrency trading. Their findings suggest that neural networks can effectively capture non-linear and volatile patterns in Bitcoin prices. Similarly, XGBoost, which has gained popularity due to its computational efficiency and performance, has been shown to outperform traditional machine learning models on tabular financial data (Chen Guestrin, 2016). LSTM models, introduced by Gers et al. (2000), have proven useful in handling time-series data, though they require significant computational resources to train effectively.

4 Results

The performance of the models was evaluated using two key metrics: Area Under the Curve (AUC) and Test Accuracy. Below are the results from the final intervals of TimeSeriesSplit:

- MLP Model: AUC: 0.5226, Test Accuracy: 61.99%.
- XGBoost Model: AUC: 0.5600, Test Accuracy: 62.26%.
- LSTM Model: AUC: 0.5504, Test Accuracy: 61.59% (only 10 epochs due to computational limitations).

The AUC metric shows that XGBoost outperforms the other models in terms of its ability to discriminate between positive and negative price movements. While LSTM was computationally expensive, its results were competitive. The MLP model was close to XGBoost, though slightly behind in performance.

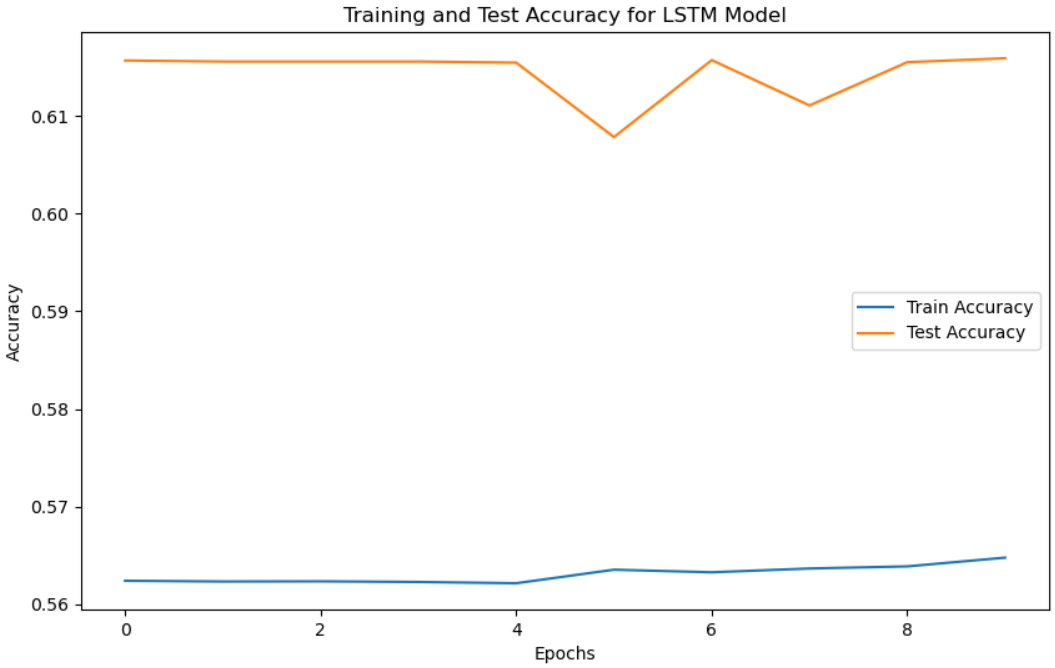

Figure 1: Training and Test Accuracy for LSTM Model over 10 Epochs

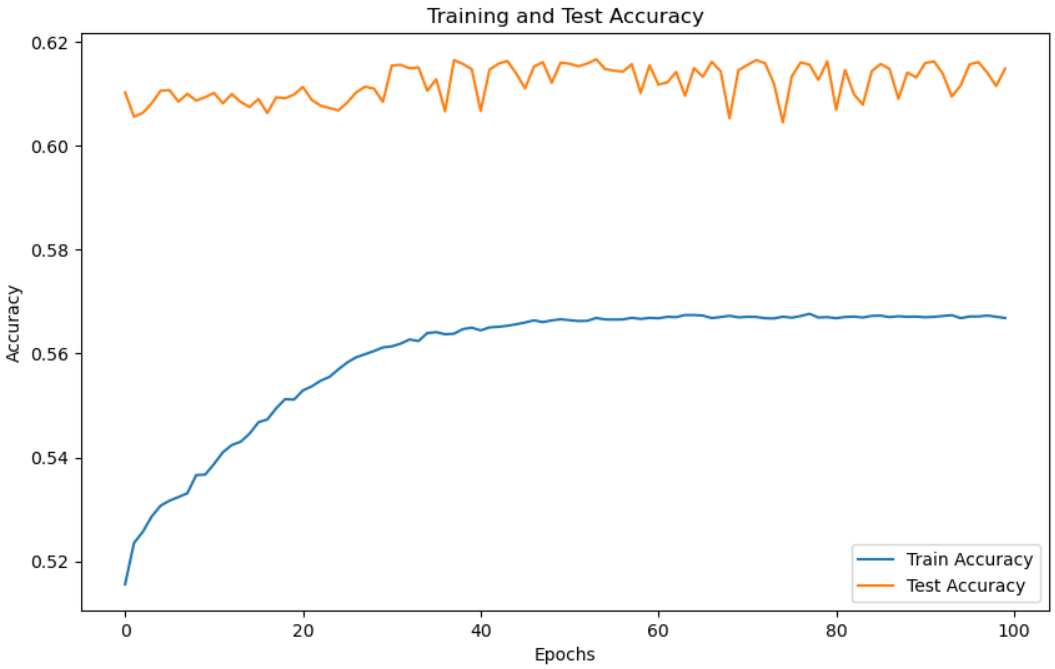

Figure 2: Confirmation of Training and Test Accuracy for MLP Model over 100 Epochs after initial TimeSplitSeries test run

5 Conclusion

This project compared MLP, XGBoost, and LSTM models for predicting Bitcoin price movements. XGBoost performed the best overall, with an AUC of 0.5600 and test accuracy of 62.26%, demonstrating its superiority in handling tabular data. Although LSTM models can capture sequential dependencies, they are computationally expensive and require more resources for training. MLP performed adequately, though it lagged slightly behind the other models. In future work, we aim to improve the LSTM model by increasing the number of epochs and using more powerful computational resources.

6 References

- Chen, T., & Guestrin, C. (2016). *XGBoost: A scalable tree boosting system.* Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.
- Gers, F. A., Schmidhuber, J., & Cummins, F. (2000). *Learning to forget: Continual prediction with LSTM.* Neural Computation.
- Murphy, J. (1999). *Technical Analysis of the Financial Markets.* New York Institute of Finance.
- Parente, M., et al. (2024). *A profitable trading algorithm for cryptocurrencies using a Neural Network model.* Expert Systems with Applications.
- XGBoost Documentation from GeeksForGeeks.
- PyTorch LSTM Documentation.