Bitcoin Price Forecasting: LSTM Time-Series Modeling

1. Executive Summary

This project addresses time-series forecasting on financial markets. I implemented an LSTM model trained on time series data of Bitcoin price from 2019 to current day. The model successfully captured time trends with healthy training behavior but demonstrated inherent limitations when new spikes in price were observed.

2. Approach & Methodology

Data

To access real bitcoin price data, I used the yfinance library to import bitcoin price from the beginning of 2019 to the most current price available.

Data Processing

For timeseries-forecasting, I used a 30-day window, where the model uses past 30 days of “high” prices of bitcoin to predict the next day’s price. Data was scaled using MinMaxScaler between 0 and 1. Date from January 2019 to December 2022 was used for training, and data from January 2023 to present was used for testing.

Models

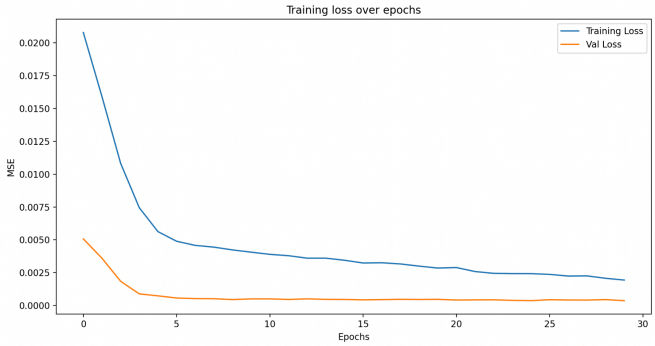
A sequential LSTM architecture with 2 layers of LSTM layers of 50 nodes with dropout layers with dropout rate 0.2 was followed by 1 dense layer of 25 nodes with ReLU activation, and finally 1 node layer for the output. The model was compiled with Adam optimizer with learning rate = 1e-4 and loss=”mean_squared_error”.

Evaluation

To evaluate the performance of the model, I created a plot comparing the actual price and the predicted price. In addition, the RMSE, root mean squared error, was computed as a measuring statistic.

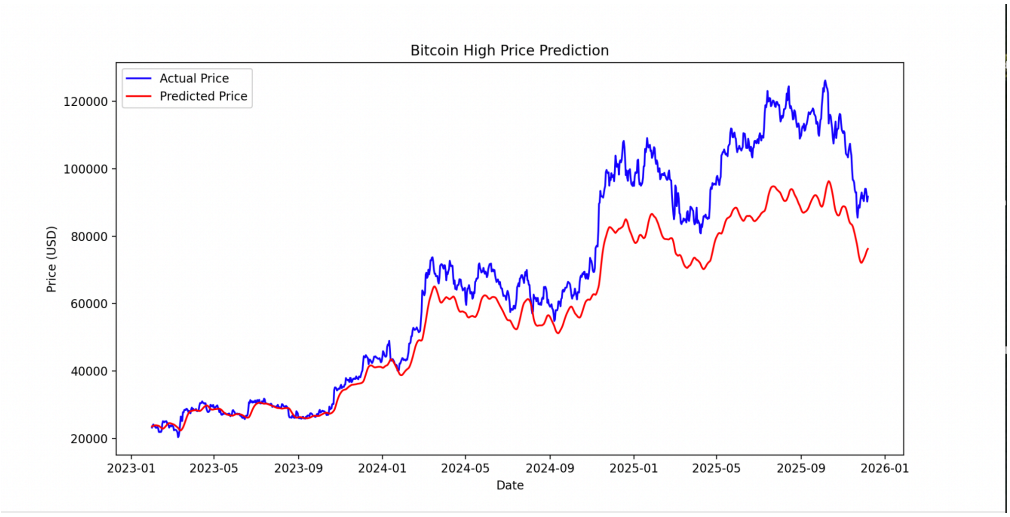
3. Result & Analysis

Figure 1: LSTM training loss (mean squared error) over epochs



The LSTM model displayed healthy training behavior reflected through the training loss and validation loss over epochs. The figure above shows a sharp drop in training loss in the first few epochs and a steady decrease for the remaining epochs, which correlates with the drop in validation loss that stays very low.

Figure 2: LSTM Bitcoin High Price actual vs predicted Bitcoin price (2023-2025)



This figure shows the values of actual price and predicted price by the model. This shows that the model was successful in finding the patterns of price change throughout the time. The model generated accurate predicted prices especially during the early periods. However, when there is a sudden stark increase in price, the model seems to output a lower predicted price than actual, underestimating the change in price afterwards.
Finally, the output returned RMSE of 10353.39.

Discussion

The results of this model illustrate both the advantages and disadvantages of the LSTM model. The model evidently learns the patterns correctly and predicts accurate predictions. However, figure 3 highlights a critical weakness. The model struggles when it encounters patterns not seen from the training data, such as sudden spikes and new all-time highs observed in the test period. This is a classic example of bias-variance trade-off, where the model is overfit to the

training data, having low bias, resulting in high variance when generalizing to unseen test data. This may contribute to the RMSE that seems to be high. However, considering the bitcoin prices in the test period range from $20,000 to $100,000, an error of $10,353 is roughly 10% average error, which is contextually reasonable.

A method to improve the result may be increasing the window size, perhaps to 60 days. This will allow the model to look back 60 days instead of 30, which means the model will be given more context on long-term trends. Also, these results reinforce the importance of feature scaling. Unscaled data can cause activation functions like sigmoid or tanh to saturate with high magnitude data, causing vanishing gradients. The optimizer will focus on features with large magnitudes, destabilizing learning. Another important factor of using LSTM is the temporal structure of data. Since the LSTM architecture is specialized for time series data where the gates and cells of the model learn and store patterns over extended sequences, preserving temporal structure is key to train successfully and get accurate results.

This LSTM neural network and the machine learning approach to tabular churn models shows clear distinctions. First of all, there is a clear difference between two examples in terms of data preparation. For the churn model, the goal is to find relationships between individual features and a target outcome. On the other hand, time-series forecasting analyzes a single variable's history to predict its future state, assuming all necessary trends are implied within the pattern over time. Thus, the preparation is different accordingly. For churn models, we convert all heterogenous features to unified form as input for models. In time-series, we create sliding windows of time and the corresponding variable. Therefore, the two architectures are suitable for different situations. The churn model is useful when we have many features and want to predict outcome based on features, while time-series excel for capturing a single variable's pattern over time.

The metrics used to evaluate performance also differ by task. For the churn classification model, I used AUC. AUC, or Area under the curve, is used to assess the accuracy of a binary classifier. The value reflects the model's ability to distinguish between two classes by looking at the trade-off between true positive and false positive rate. AUC ranges from 0.5 to 1, where 0.5 means the model is guessing by random chance, and 1 being perfectly distinguishing every case. The AUC serves as a good measure of a classifier's performance. The RMSE, root mean squared error, was used for time-series forecasting. RMSE is used to assess regression, measuring the average difference between actual and predicted value by the model. Key advantage is that it has the same units as the target variable, allowing easy interpretation of the model's performance.

4. Outlook / Future Enhancements

This model showed some decrease in performance for later periods due to unseen trends from training data such as spike in prices. A solution to this can be adding more inputs such as S&P 500 index or sentiment data to add more context and connection to macroeconomic changes rather than relying solely on past price change trends of bitcoin. Also, differencing the data to predict the change in price rather than the absolute price can also yield better results.