Bitcoin Price Prediction with an LSTM

Raunak Sood and Noah Baird

I. Abstract

This paper shows that an LSTM is able to accurately predict Bitcoin prices from limit order book data from Binance. We achieve a mean absolute error of 9.98 when predicting price and 0.00015 when predicting price change. We also utilize an Auto ARIMA model as a baseline which reaches a mean absolute error of 1101.95 for price and 0.00016 for price change. Thus, we find that the LSTM greatly outperforms the baseline model for price prediction and marginally outperforms it for price percent change prediction. Although there is very limited research, particularly on crypto LOB data, some studies on general LOB data suggest that implementing an inception model and a CNN along with the LSTM could improve the models performance.

II. Introduction

Today, a majority of market trading is done through limit order books (LOB). These provide a transparent and efficient mechanism to match buyers and sellers. This system allows both buyers and sellers to submit their orders with specified price levels. Viewers can assess resting limit orders on the exchange to make informed trading decisions. Because the LOB consists of many levels of prices and volumes of orders, modeling these prices is a complex, multidimensional problem that requires a novel approach. In this paper both an autoregressive integrated moving average (ARIMA) as well as a long short term memory (LSTM) model are developed. These models aim to capture patterns in noisy LOB data to make accurate predictions about future prices and price percent changes of bitcoin. With these we aim to assess the feasibility of deploying an LSTM model as a short term future price predictor for bitcoin.

III. Methodology

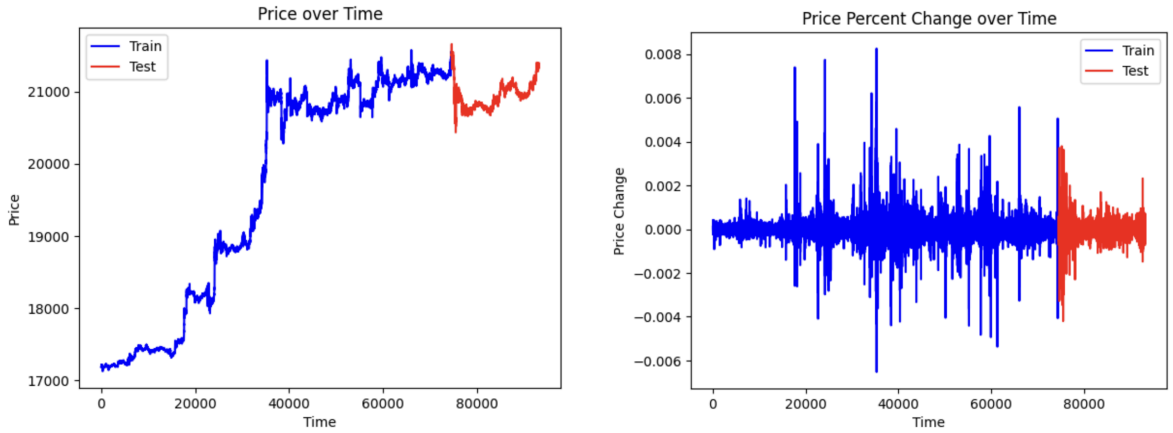
A. Data

The data used in this study is taken from Kaggle and titled, “Bitcoin Limit Order Book (LOB) Data.” According to the description, data is taken from Binance Bitcoin perpetual data. It encompasses 12 consecutive days of data, from January 9th, 2023 to January 20th, 2023. The “data points are sampled at a frequency of 250 milliseconds and contain a total of 3,730,870 rows and 42 columns” (Raz 2023). The data includes columns of index, time, and timestamp, followed by 20 columns of bid prices and volumes, and 20 columns of ask prices and volumes. For both bids and asks, we have 10 levels of volumes and prices.

B. Pre-Processing

The first step we took was reading in the data and selecting every 40 rows or 10 seconds. This both reduced the noise and represented the latency seen in crypto exchanges. Latency refers to the amount of time between an order being placed and it being executed. Next, we create a ‘price’ column. This is calculated by taking the average of the second and twenty second columns. This is because the second column is the highest bid and the twenty second column is the lowest ask. Thus it is intuitive that the price purchased should be the middle of these prices or very close to it. Next we create a column for ‘next_price’ which is the price of the row beneath. The reason for doing this is because we want to use the current price as a feature when predicting the next price. We also take ‘price_change_pct’ which is the price percentage change between the price in one row and the price in one above. We also do ‘next_pct’ for this. Predicting the price percent change is quite important as well as traders often use percent change as exit and entry points. Next, we drop the first three columns as we don’t need index, time, and timestamp

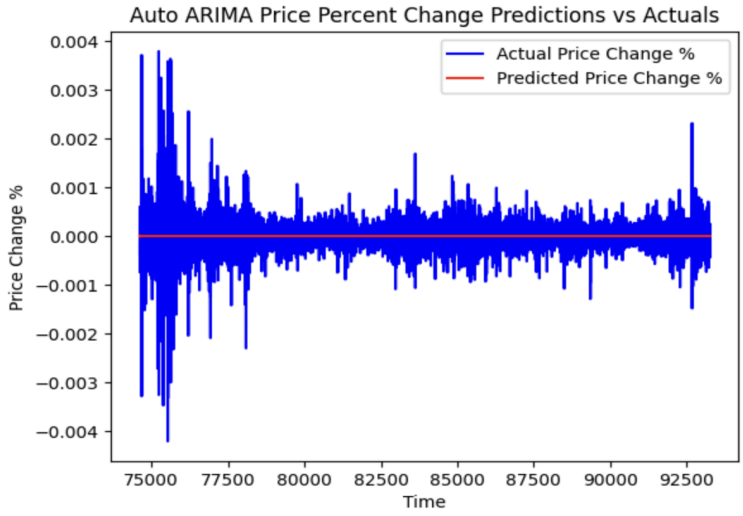
as features. Finally we split the data into training, validation, and test sets. Below are plots of price over time and price percent change over time.



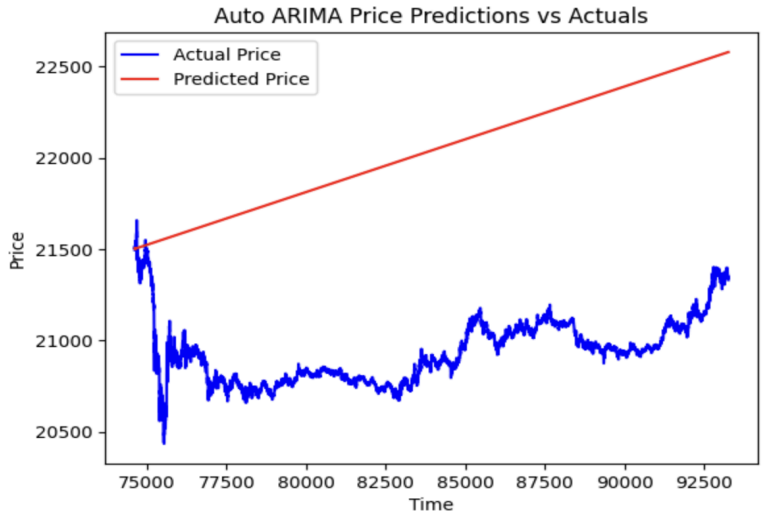
We see that both plots look very different. Price over time has a pretty steep upward trend with a small steep drop right as our test set begins. On the other hand, price percent change is very stationary with values often at or very close to 0, due to the high frequency nature of the data.

C. ARIMA

We begin with constructing an ARIMA model, a common statistical model used for time series analysis. This will serve as our baseline to see how our LSTM model compares to a more traditional approach. ARIMA models combine autoregressive, integrated, and moving average components. The autoregressive component looks at the correlation with past values. This is represented by the parameter ‘p’ which is how many lagged observations are included in the model. The integrated component, represented by the parameter ‘d’, is the order of differencing. Finally, the moving average component is represented by the parameter ‘q’ and represents the lagged forecast errors that the model will include. We choose to use auto ARIMA to automate the process of picking our best parameters (p, d, q) using BIC and AIC, goodness of fit metrics. For price percent change, we reach a model of 0,0,3. Below is the graph of the predictions it reached.



We see that the model came up with a straight line at 0 for the predictions. Although this looks simple, it actually performs very well with a mean absolute error of 0.00016. As said previously, this is because price percent change is often 0 or very close to it due to its high frequency nature. When running auto ARIMA for price, we reach a model of 2,1,2. Below is the graph of the predictions reached.



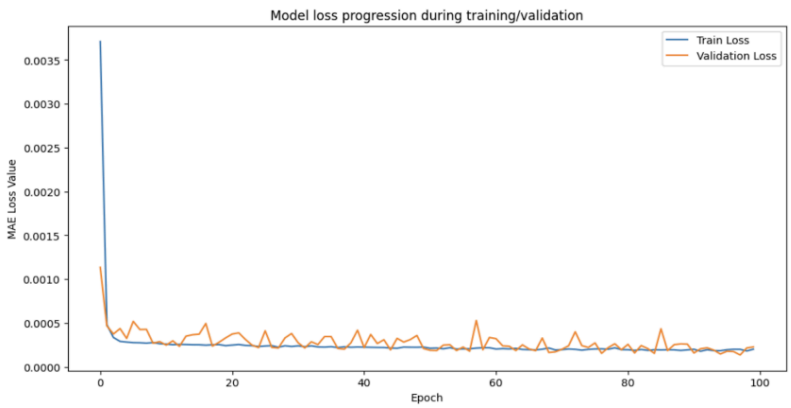
As seen, the line predicted doesn't seem to perform well, sloping upwards, quite far off the actuals. This is likely because the training set, which the ARIMA trained on, had a steep upward slope, while the validation set had an unexpected drop downwards. It reaches a mean absolute error of 1101.5.

D. LSTM

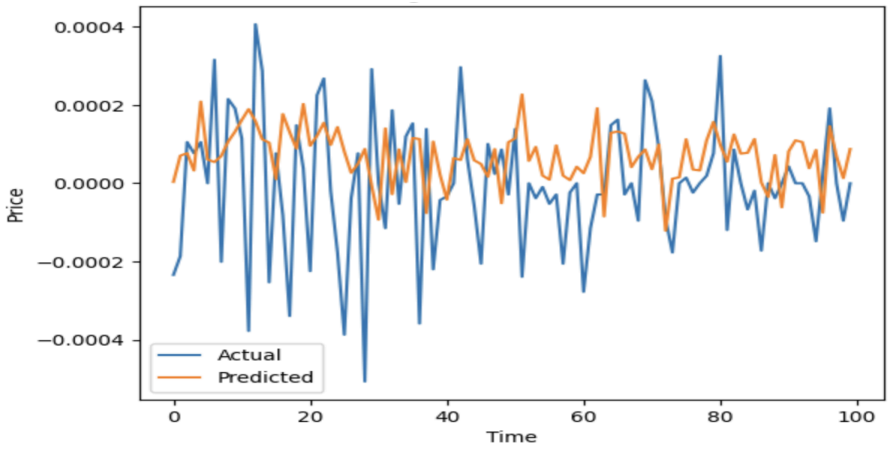
After creating the ARIMA model, we developed two LSTM models to generate more accurate predictions. Recurrent neural networks (RNN's) such as an LSTM have the capability to predict time series data by taking into account both previous outputs and the current input. However, basic RNN's struggle to retain information over long periods of time, also causing them to suffer from vanishing and exploding gradients where the model isn't able to learn effectively. However, an LSTM is able to mitigate this problem by maintaining long term dependencies with its cell state. The cell state holds information as a vector that can be changed through the forget and input gates. The forget gate takes in the previous cells hidden state or output and combines this with the input from the current data to decide information that will be forgotten from the cell state. The input gate first determines which entries in the cell state to update using the previous hidden state and the current input, and then it determines how much of the proposed update we should include in the cell state. The cell state is updated with this new information. The output gate then determines the next hidden state based on the updated cell state. This process is repeated until all of the data for training has been passed through.

For both LSTM models, the data went through min max scaling to regularize it between 0 and 1. The first model created using this architecture was for predicting the next percent change. This used a LSTM layer with 25 nodes, a dense hidden layer of 16 nodes, and a final dense output layer of 1 node. This was compiled with the Adam optimizer, a learning rate of 0.00005,

and mean absolute error loss function. Regularization techniques such as dropout were tested, although they didn't help. We plotted the validation against training to check for overfitting and find the right amount of epochs to use. Below we see that although loss stays around the same, it gradually decreases as it gets to 100, so we choose to set epochs to 100.

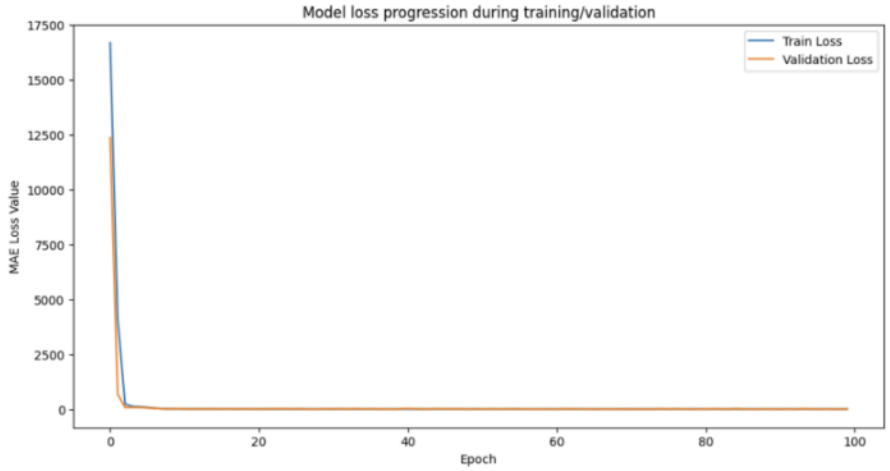


The plot below shows the predictions over the first 100 data points of the test data.

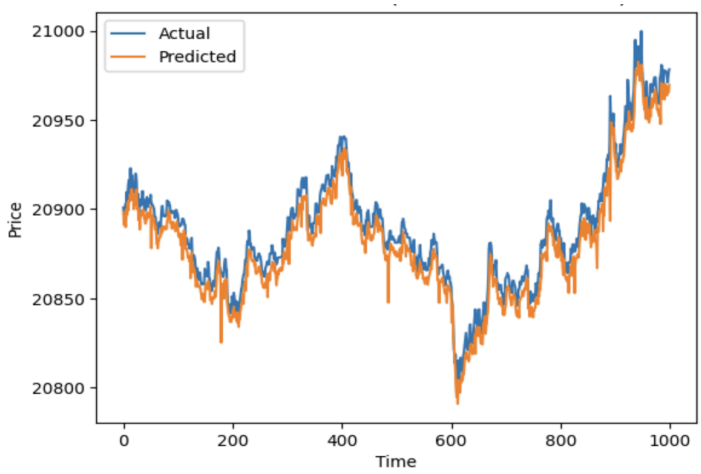


In comparison to the ARIMA model, the LSTM was able to get a mean absolute error of 0.00015 and produce predictions for the percent price change that were not just zero. We reached our best predictions when creating a shallower network that generalized on unseen data better. However, this model still wasn't able to perfectly calculate the up and down spikes seen in this data.

The second model was used for predicting price change. This used an LSTM layer of 50 nodes, a dense layer of 128 nodes with the rectified linear unit (relu) activation, and a final output layer of 1 node. It was compiled using the adam optimizer with a learning rate of 0.0001 and mean absolute error loss function. Once again, we find that the validation loss, although relatively stable, was lowest at 100 epochs.



Thus, it was fit for 100 epochs. The predictions for this model's first 1000 test observations are shown below.



This model was able to produce accurate predictions over the course of the test set, getting a mean absolute error of 9.98. The recurrent nature of this model, and the feature set including past

prices helped it perform very well. This model outperformed our baseline ARIMA and followed bitcoin's trend very well making accurate predictions for price.

IV. Conclusion

To conclude, we find that the LSTM performs significantly better than the ARIMA for price prediction, and marginally better for price percent change predictions. This is likely due to the LSTM's ability to hold long term dependencies with its cell state, and using features like bid prices and volumes, ask prices and volumes, and the previous price or price percent change. It is important to note that the LSTM struggled when trying to outperform the ARIMA by a large margin for price percent change. Due to the volatility of the data, the model couldn't predict the up and down spikes with great accuracy. Overall this study was quite exciting as it appears to be one of the few applications of deep learning to crypto trading from limit order books. It was also able to reach high accuracy for both models, and with further improvements, could be deployed for real time trading. Research on price predictions with limit order book data suggests that using an inception model to wrap convolution and pooling layers along with LSTM would help improve performance. It is suggested that convolution layers would help understand spatial information of the data and pooling layers would help extract the most important features. Regardless, this study sees the successful implementation and use of neural networks to predict prices and price percent changes of bitcoin, given limit order book data from Binance.

References

"Bitcoin." *Wikipedia*, Wikimedia Foundation, 6 Dec. 2023, en.wikipedia.org/wiki/Bitcoin.

Chollet, Francois. *Deep Learning with Python*. 2nd ed.

Journal of La Deeplob: Deep Convolutional Neural Networks for Limit ..., arxiv.org/pdf/1808.03668.pdf. Accessed 13 Dec. 2023.

Kenton, Will. "What Is a Limit Order Book? Definition and Data." *Investopedia*, Investopedia, www.investopedia.com/terms/l/limitorderbook.asp#:~:text=A%20limit%20order%20book%20is,a%20specific%20price%20or%20better. Accessed 12 Dec. 2023.

Siavash_raz. "Bitcoin Limit Order Book (LOB) Data." *Kaggle*, 5 Aug. 2023, www.kaggle.com/datasets/siavashraz/bitcoin-perpetualbtcusdtp-limit-order-book-data.