Bitcoin & Ethereum Price Prediction

I. INTRODUCTION:

My motivation for undergoing this project was having recently began investing in cryptocurrency, I was wondering what was the best current way to predict future prices. After much research, I determined that using a Long Short-term Memory (LSTM) to look at historical market data was the best and most accurate way to go about this. I believe that having a machine to learn the historical data and give me more insight into what the market future could look like would help improve my trading skills.

I plan to investigate how to use machine learning to build a model that is specialized in predicting the future of cryptocurrency prices. I plan on determining which parameters and hyperparameters lead to the best accuracy, precision, and recall of the model on the test data.

My approach was to use data gathered from a coinbase.com API call and preprocess this data to feed it as input to my model which would output future price predictions. I ran into many challenges with the demands the Sklearn LSTM model forces on the dimensions and type of data. This led to me spending more time formatting the data properly than I would have liked.

The results of this experiment was a surprisingly more visually accurate model than I was expecting for all parameter variations. These results are documented thoroughly and plotted for an easier visual display of the model prediction. As expected, the more epochs taken, the greater the model accuracy as well as lower loss. There was also a slight correlation between a larger batch size and a more accurate model.

II. DATA MINING TASK:

This task involved mining data of the history of Bitcoin and Ethereum prices to feed an LSTM model that will learn the past patterns and attempt to apply this knowledge to learn future patterns. This knowledge could be useful when determining the proper time to buy and sell Bitcoin and Ethereum.

Input Data

The input data from my project was the price history of Bitcoin and Ethereum as taken from a CoinBase.com API call. This data was then transferred into CSV files to be accessed easier and reduce program execution time. The size of the input data is 98,990 data entries taken 5 minutes apart beginning 01/01/2021 and ending 10/01/2021. Each entry contains (time, low, high, open, close, volume) and these are grouped into blocks of 24 which represents 2 hours of data points in each block. This data is what a trader would use and look at through charts in order to make their own predictions of the future trends of cryptocurrency prices. This data is further separated into training and testing data with 80% being train data and 20% test data. Model uses market close data for price prediction with the other entry points as feature values.

Output Data

The output data from the model in my project is price predictions corresponding to future time points. This output data will be analyzed and compared to the actual price data to determine accuracy and loss of the model. This data would give the user an accurate prediction of the future trends in the prices of cryptocurrency which the users can use to determine when to buy and sell.

Questions to Investigate

- Can an LSTM accurately predict cryptocurrency prices?
- How far out can the model accurately make predictions?
- Does the model improve accuracy after more epochs?

Key Challenges

I ran into the most challenges with preprocessing the data and transforming the data into the correct format for the LSTM. After the data was formatted correctly and the base LSTM algorithms was operational, I was able to easily adjust the parameters and hyperparameters to look at various combinations and determine the best accuracy and loss values.

III. TECHNICAL APPROACH:

My algorithmic approach to solving the data mining task that I have described above can be detailed below. All the steps that I describe below are repeated for both Bitcoin and Ethereum. First, I used a coinbase.com API call to gather data about the history of the past year of Bitcoin and Ethereum markets. This information was brought in as data frames and saved as csv files to make the data more accessible to the script. The next step was to preprocess this data and prepare it to be used with the TensorFlow Keras LSTM model. This involved first compressing the output data into a smaller range of values (between 0 and 1). After the data is compressed and reshaped into a scaled form it is broken down into 2-hour bites. This is done by grouping 24 data points together into a single sequence and storing the entire dataset as a list of sequences. This allowed me to then process the data sequences to break the data into training and testing data. The first 80% of data points were used for training and the last 20% used for testing. The closing price is used as the output data and is what the model attempts to predict.

Next, the LSTM model is built using a bidirectional LSTM layer followed by a Dense layer and a linear activation function layer. I chose this model for a few reasons. When I was reading documentation about the Bidirectional LSTM models for time series tracking, I became convinced that they had the best performance. I knew that I wanted to finish with a linear activation function because the predictions would be made on a linear scale. I also read from Keras documentation that adding a dense layer in the LSTM model is a great way to condense the data right before a prediction is outputted. This is done to make the data more readable to users as well as prepare the data for the activation layer.

Finally, the model was compiled using the mean squared average loss function and the Adamax optimizer algorithm. The model was then trained on the appropriate training data and tested on the appropriate testing data using the Keras model.fit function. This data had already been preprocessed above in my script and was ready for use with the LSTM model. The inputs to this fitting function included training data, number of epochs and batch size. Shuffle was also necessary to include and set to False so that the order of the data can remain unchanged and the model can accurately use date/time as a learnable feature. The parameters and hyper parameters are adjusted, and the results are displayed accordingly in the results section below.

I addressed the challenges mentioned above by using google to constantly look up how to solve the errors that I would encounter. This led to me using Sklearn preprocessing to scale the data down between 0 and 1. I also learned about adding the dense layer to the 2nd to last layer of the LSTM model through these efforts. I learned that it isn't necessary to have a dense layer between a bidirectional LSTM layer and an activation layer, although it can lead to a smoother transition and less of a loss of information.

Hyperparameter tuning was performed in order to optimize the LSTM model to predict cryptocurrency prices. The parameters that were adjusted were the number of epochs which was tested at 5 and 10 (I would have liked to test higher numbers of epochs; however, the execution

time became too great past 10 epochs). I also tested the model using 3 different batch sizes of 32, 64, and 128.

IV. EVALUATION METHODOLOGY

The Dataset that I used to study this task was relatively clean and had no major issues with missing features or noisy data. The only challenge with the dataset was working to achieve the proper formatting to become input for the LSTM model. The source of the dataset was from Coinbase.com market history data for Bitcoin and Ethereum for the year of 2021 from January 1st until October 1st.

The metrics that I used to evaluate the output of this data mining task was visual inspection of graphs and loss value. I used visual inspection of plots to determine the answer to questions I investigated above such as how far out the model can make accurate predictions. Loss value is a good metric to evaluate an LSTM model because it assigns a value to the amount of information lost during each epoch of the training and testing sessions. This value is a good way to determine whether your model is learning the appropriate features and weights to make accurate predictions. By visually inspecting the plots produced that plot test results and predicted results side by side, it is easy to determine which hyperparameter combination leads to the best results.

V. RESULTS AND DISCUSSIONS

When it came to determining the results of my model, I used a few functions to help determine the model accuracy and loss values. I will further discuss the results below in this section after I finish describing how I analyzed the results. After the model is evaluated for accuracy and loss, I then wrote a function that would plot the model predictions to a graph. This graph would contain the model prediction prices as well as the actual cryptocurrency prices in order to give the user a great visual comparison of how accurate the model is predicting prices. These graphs and results are all included further below in this section. This includes charts showing the model predictions compared to the actual prices for both Bitcoin and Ethereum.

One of the things that worked very well was using matplotlib.pyplot to graph the results in an easily digestible visually appealing manner. This allowed me to easily see which changes les to the best results and update the model accordingly. I also found it very useful to use the TensorFlow, Keras, and Scikit learn libraries for data manipulation and model design. Although it took a while to learn how to properly use these libraries, without them this project would have been much harder.

One of the things that I encountered as a problem and eventually discovered would not work was trying to use accuracy as a good summary of the model. This is due to the price predictions being even cents off from the actual price would be marked incorrect. This led to me visually comparing graphs in order to better determine how accurate my model would perform rather than looking at an accuracy metric. I also ran into difficulties with plotting the data side by side to compare predictions vs real prices. My solution to this was to produce 2 graphs which I could screenshot and include in this report below where I discuss the visual differences and what this means.

The actual prediction charts were modeled with 24 predictions per data point which allows the model to naturally predict a high and low value while at the same time predicting the overall trend with the market prices.

BATCH SIZE 32 RESULTS

Ethereum Model Training Loss values

Epoch #1: 2.265e-4, Epoch #2: 2.187e-4, Epoch #3: 1.084e-4, Epoch #4: 1.0453-4,

Epoch #5: 1.065e-4, Epoch #6: 1.097e-4, Epoch #7: 1.126e-4, Epoch #8: 1.146e-4,

Epoch #9: 1.153e-4, Epoch #10: 1.569e-4

```
Epoch 1/10
2475/2475 [==============================] - 37s 13ms/step - loss: 2.2654e-04
Epoch 2/10
2475/2475 [==============================] - 30s 12ms/step - loss: 2.1876e-04
Epoch 3/10
2475/2475 [==============================] - 29s 12ms/step - loss: 1.0843e-04
Epoch 4/10
2475/2475 [==============================] - 30s 12ms/step - loss: 1.0449e-04
Epoch 5/10
2475/2475 [==============================] - 34s 14ms/step - loss: 1.0650e-04
Epoch 6/10
2475/2475 [==============================] - 33s 13ms/step - loss: 1.0971e-04
Epoch 7/10
2475/2475 [==============================] - 30s 12ms/step - loss: 1.1257e-04
Epoch 8/10
2475/2475 [==============================] - 30s 12ms/step - loss: 1.1465e-04
Epoch 9/10
2475/2475 [==============================] - 31s 12ms/step - loss: 1.1536e-04
Epoch 10/10
2475/2475 [==============================] - 32s 13ms/step - loss: 1.1569e-04
```

Bitcoin Model Training Loss values

Epoch #1: 6.059e-4, Epoch #2: 2.163e-4, Epoch #3: 2.156e-4, Epoch #4: 2.121e-4,

Epoch #5: 2.074e-4, Epoch #6: 2.065e-4, Epoch #7: 2.056e-4, Epoch #8: 2.027e-4,

Epoch #9: 1.962e-4, Epoch #10: 1.885e-4

```
Epoch 1/10
2475/2475 [==============================] - 38s 13ms/step - loss: 6.0598e-04
Epoch 2/10
2475/2475 [==============================] - 35s 14ms/step - loss: 2.1630e-04
Epoch 3/10
2475/2475 [==============================] - 33s 13ms/step - loss: 2.1557e-04
Epoch 4/10
2475/2475 [==============================] - 32s 13ms/step - loss: 2.1212e-04
Epoch 5/10
2475/2475 [==============================] - 32s 13ms/step - loss: 2.0742e-04
Epoch 6/10
2475/2475 [==============================] - 32s 13ms/step - loss: 2.0651e-04
Epoch 7/10
2475/2475 [==============================] - 34s 14ms/step - loss: 2.0564e-04
Epoch 8/10
2475/2475 [==============================] - 32s 13ms/step - loss: 2.0268e-04
Epoch 9/10
2475/2475 [==============================] - 32s 13ms/step - loss: 1.9619e-04
Epoch 10/10
2475/2475 [==============================] - 35s 14ms/step - loss: 1.8853e-04
```

Ethereum Model Testing Loss value: 0.01279

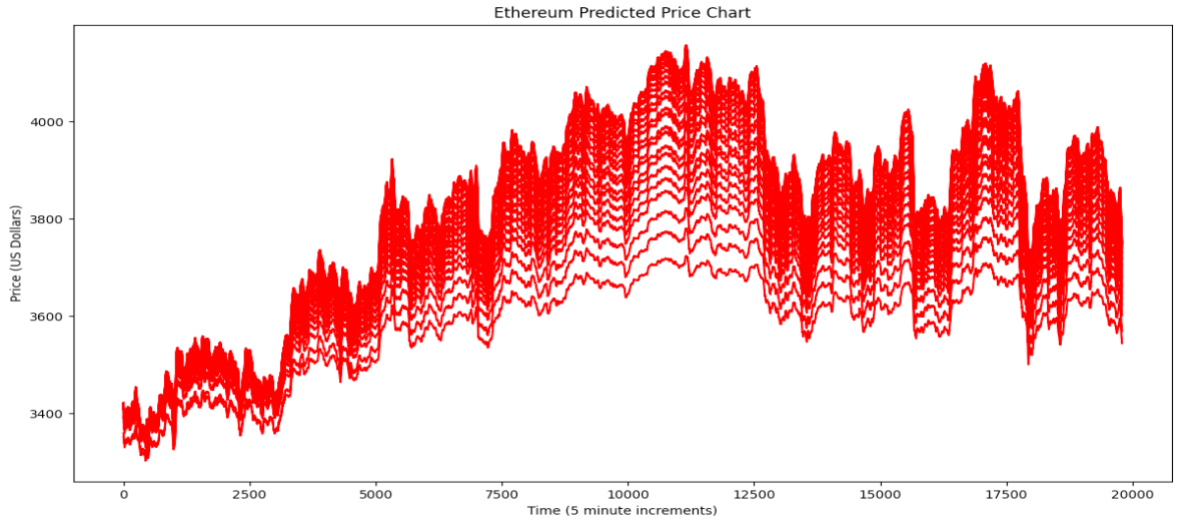
Bitcoin Model Testing Loss value: 0.15923

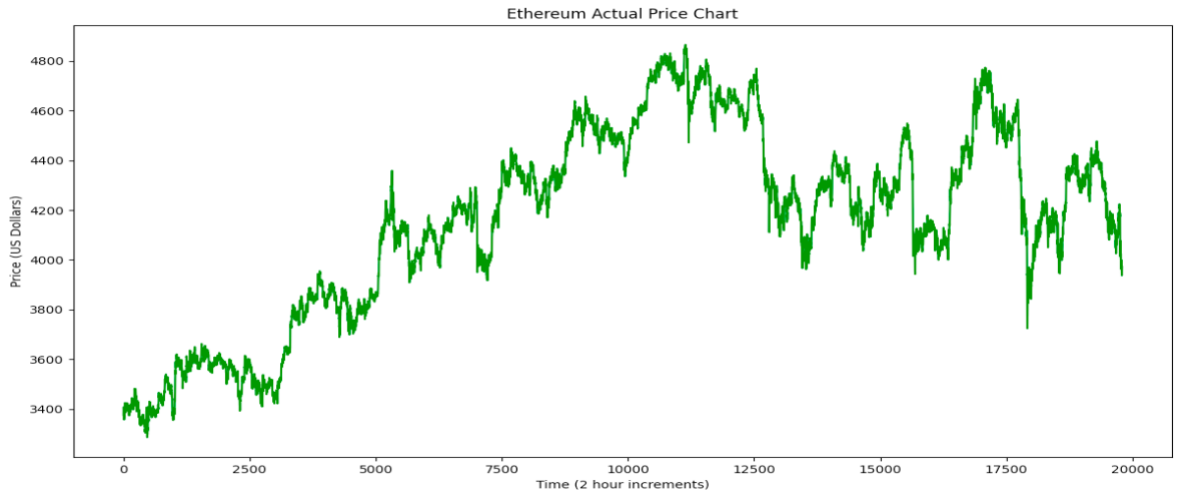
```
Ethereum Results:

619/619 [==============================] - 3s 4ms/step - loss: 0.0128
0.012786277569830418
Bitcoin results:

619/619 [==============================] - 3s 4ms/step - loss: 0.0159
0.015923094004392624
```

Ethereum Model Predictions vs Actual price charts

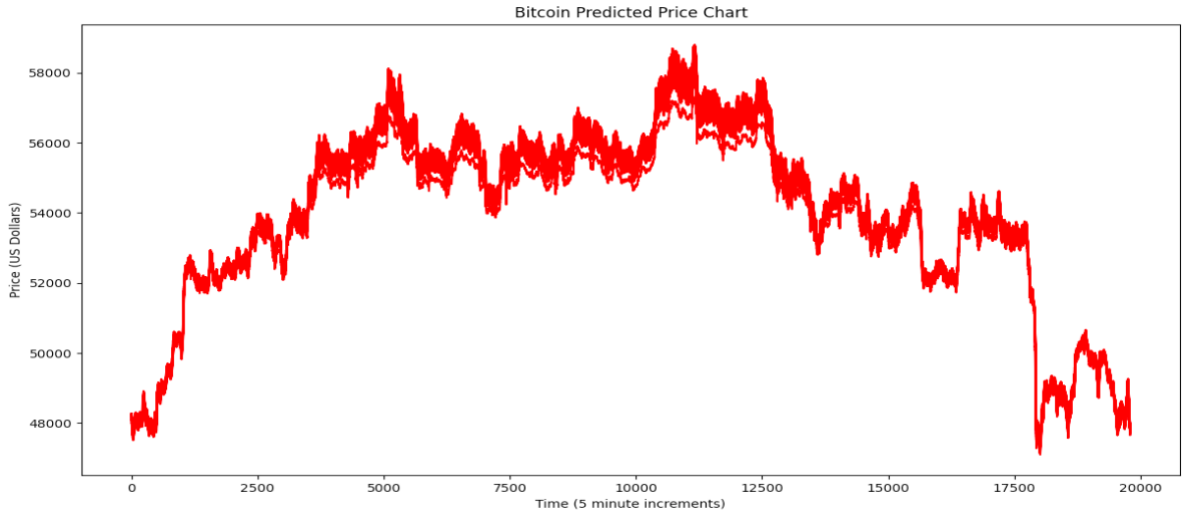


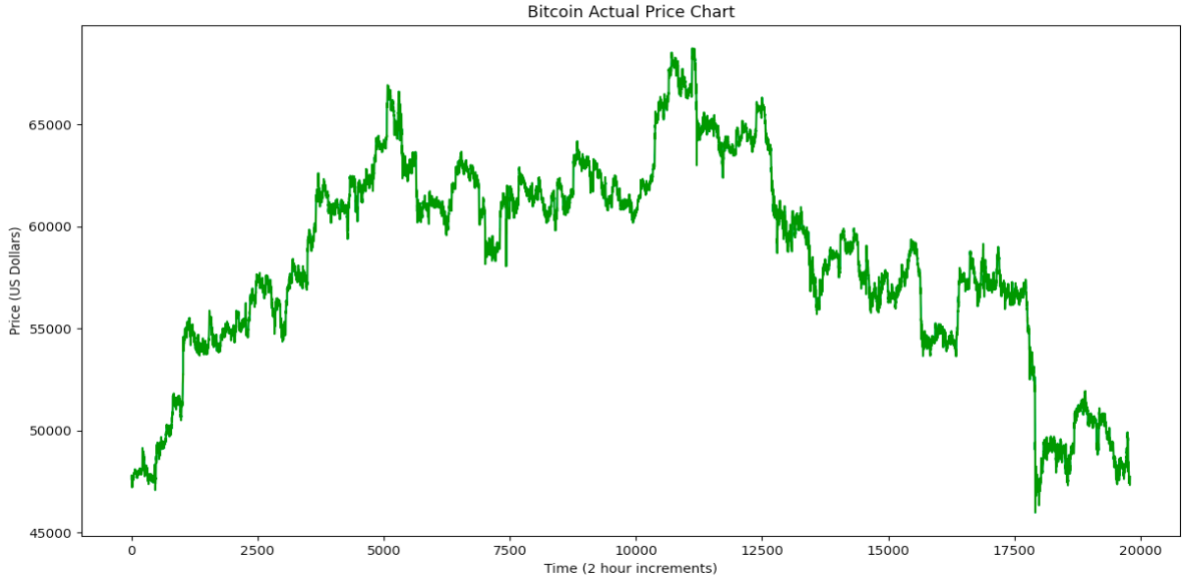


As clearly seen, the Ethereum model did a very good job at predicting the price movement or future patterns of Ethereum prices. Since the model made 24 prediction lines each slightly higher or lower than the other, the lowest of the lines was correct in determining the bottom of the valleys when the prices dipped very low. The model highest line prediction was the appropriate line in determining the height of the peaks when the price goes up very high. I believe this will be very helpful in predicting the movement of Ethereum prices. I can look to the

middle of the prediction lines during the normal smaller peaks and valleys, however, when a large peak or valley is predicted, I can look at the highest or lowest prediction lines.

Bitcoin Model Prediction vs Actual price charts





The Bitcoin model is similarly accurate to the Ethereum model in predicting the trend of future prices. The bitcoin model made much more concise predictions relative to the Ethereum model. It was more confident in the price predictions rather than providing more of a range or prices when determining the trend. Although the trend is predicted well, the accuracy in the price prediction on the peaks is not as accurate as the Ethereum model. The model predicts lower than the actual peak value which can be interpreted as either good or bad.

BATCH SIZE 64 RESULTS

Ethereum model training loss values:

```
Epoch 1/10
1238/1238 [==============================] - 31s 18ms/step - loss: 4.7766e-04
Epoch 2/10
1238/1238 [==============================] - 17s 14ms/step - loss: 3.1897e-04
Epoch 3/10
1238/1238 [==============================] - 18s 15ms/step - loss: 1.2486e-04
Epoch 4/10
1238/1238 [==============================] - 19s 15ms/step - loss: 1.0209e-04
Epoch 5/10
1238/1238 [==============================] - 21s 17ms/step - loss: 9.7533e-05
Epoch 6/10
1238/1238 [==============================] - 21s 17ms/step - loss: 9.5480e-05
Epoch 7/10
1238/1238 [==============================] - 18s 15ms/step - loss: 9.4448e-05
Epoch 8/10
1238/1238 [==============================] - 20s 16ms/step - loss: 9.3873e-05
Epoch 9/10
1238/1238 [==============================] - 20s 16ms/step - loss: 9.3342e-05
Epoch 10/10
1238/1238 [==============================] - 19s 15ms/step - loss: 9.2740e-05
```

Bitcoin model training loss values:

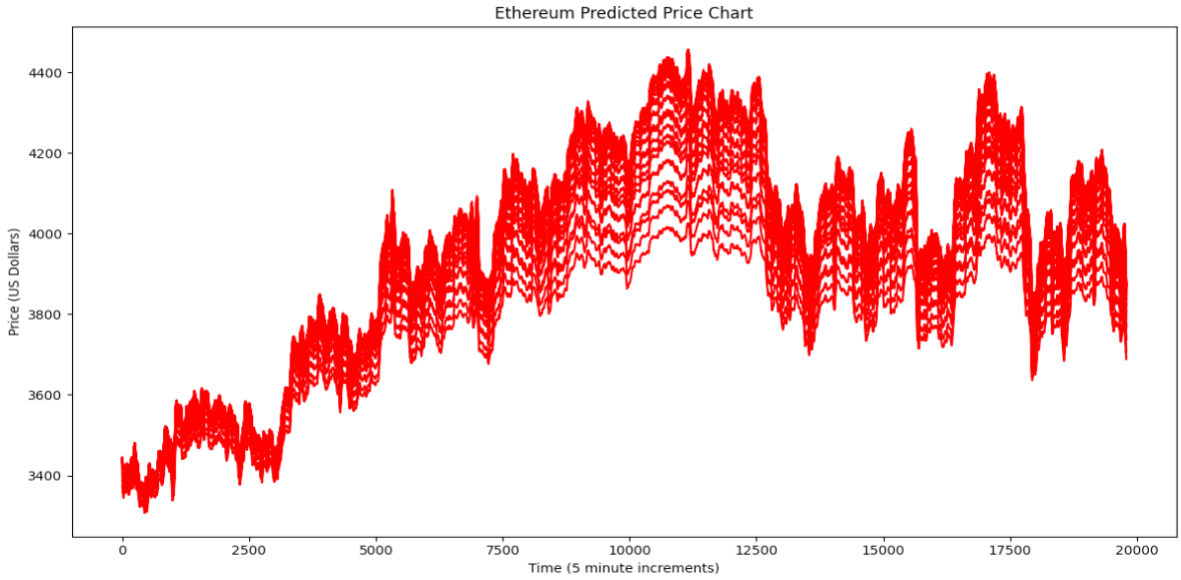
```
Epoch 1/10
1238/1238 [==============================] - 25s 15ms/step - loss: 5.9093e-04
Epoch 2/10
1238/1238 [==============================] - 18s 14ms/step - loss: 2.4417e-04
Epoch 3/10
1238/1238 [==============================] - 18s 15ms/step - loss: 1.6301e-04
Epoch 4/10
1238/1238 [==============================] - 21s 17ms/step - loss: 1.6589e-04
Epoch 5/10
1238/1238 [==============================] - 20s 16ms/step - loss: 1.7006e-04
Epoch 6/10
1238/1238 [==============================] - 20s 16ms/step - loss: 1.7748e-04
Epoch 7/10
1238/1238 [==============================] - 19s 16ms/step - loss: 1.8484e-04
Epoch 8/10
1238/1238 [==============================] - 17s 14ms/step - loss: 1.8868e-04
Epoch 9/10
1238/1238 [==============================] - 17s 14ms/step - loss: 1.9331e-04
Epoch 10/10
1238/1238 [==============================] - 17s 14ms/step - loss: 1.9732e-04
```

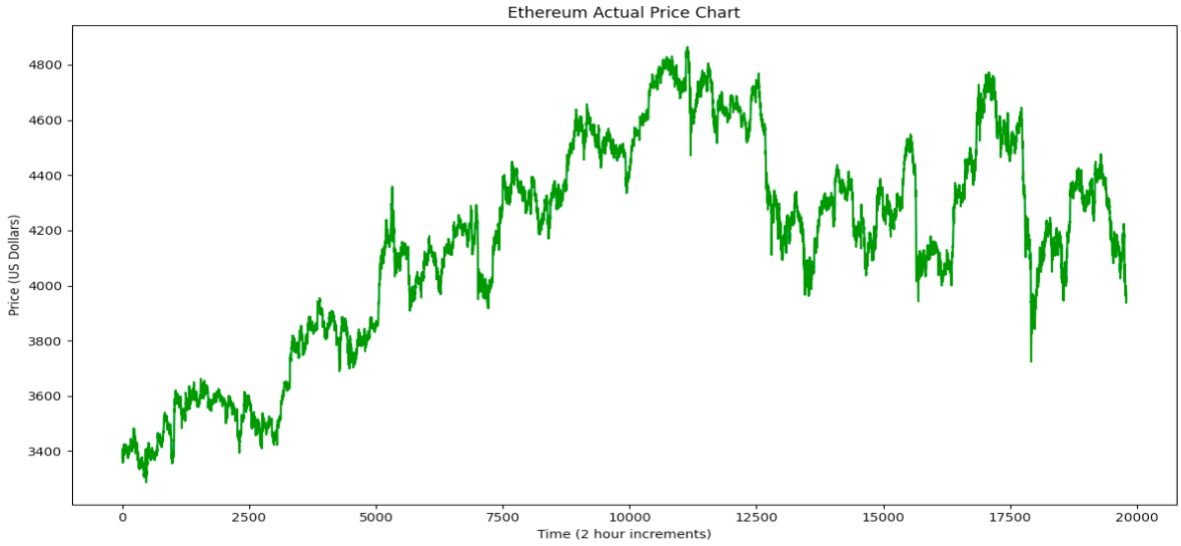
```
Ethereum Results:

619/619 [==============================] - 4s 4ms/step - loss: 0.0049
0.0048973229713737965
Bitcoin results:

619/619 [==============================] - 4s 4ms/step - loss: 0.0073
0.007291084621101618
bnels17@Brendens-MacBook-Air coinbase %
```

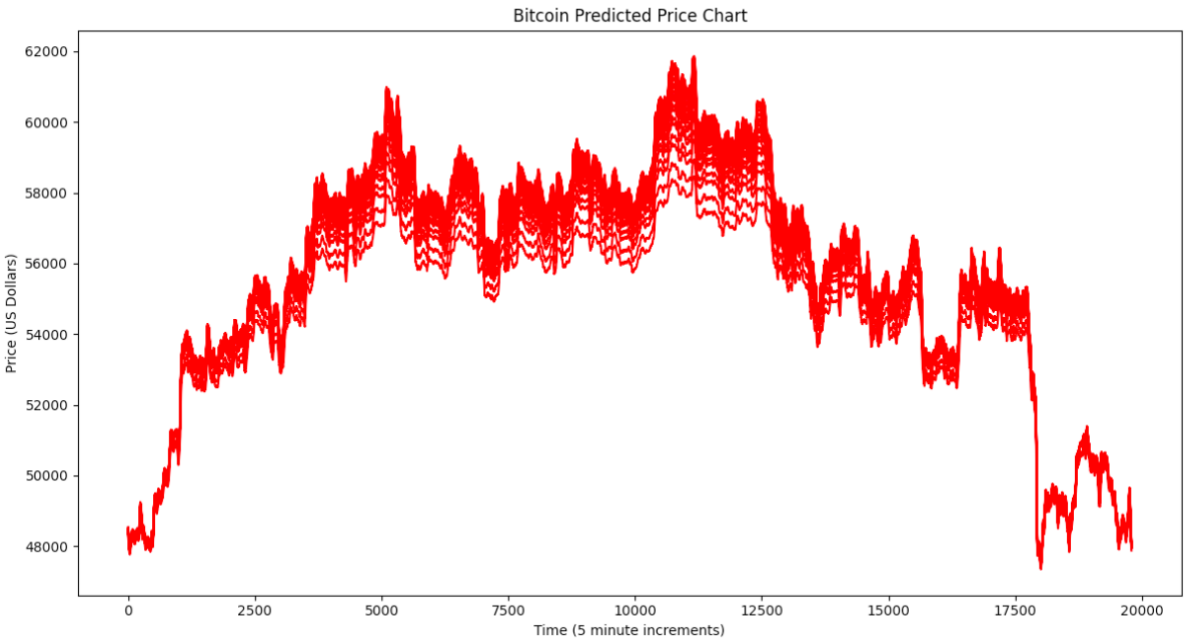
Ethereum Model Predictions vs Actual price charts

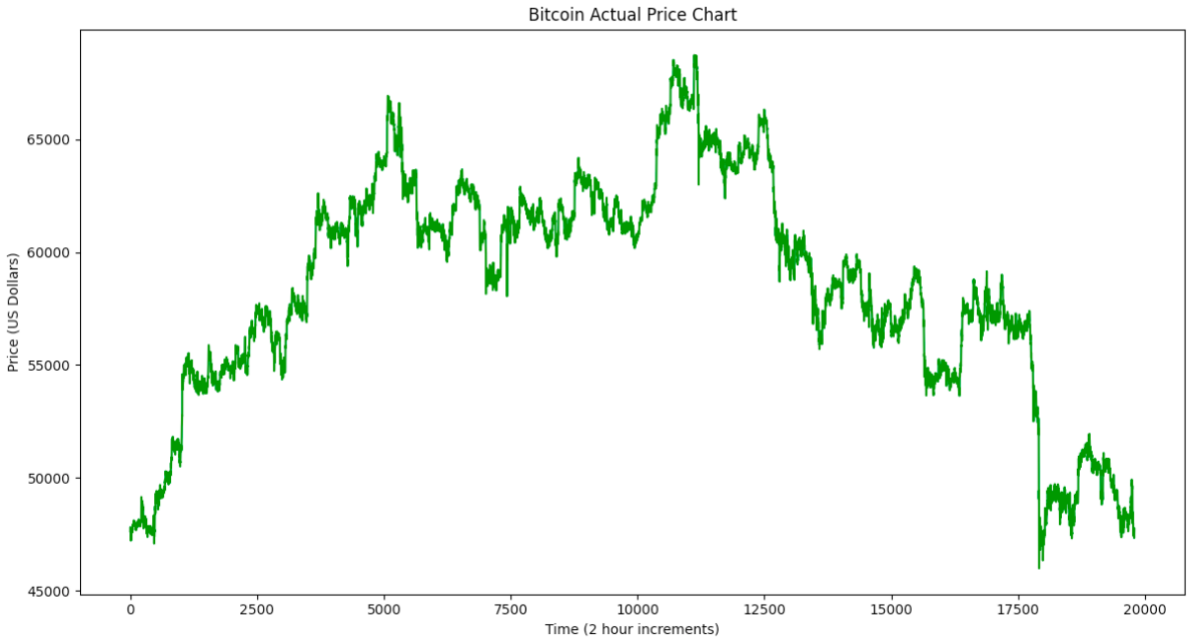




This model was again very accurate on the test data and predicted the price trend of Ethereum very well. The same as above, this model has a range of predictions with a similar spread.

Bitcoin Model Prediction vs Actual price charts





This model was slightly more accurate in predicting the price and trends in the Bitcoin market than the 32 batch size. This model was also better at predicting the high peaks and low valleys than the previous model.

BATCH SIZE 128 RESULTS

Ethereum model training loss values:

```
Epoch 1/10
619/619 [==============================] - 23s 29ms/step - loss: 6.4749e-04
Epoch 2/10
619/619 [==============================] - 21s 34ms/step - loss: 4.7379e-04
Epoch 3/10
619/619 [==============================] - 14s 22ms/step - loss: 1.4205e-04
Epoch 4/10
619/619 [==============================] - 13s 20ms/step - loss: 8.5683e-05
Epoch 5/10
619/619 [==============================] - 15s 23ms/step - loss: 7.6065e-05
Epoch 6/10
619/619 [==============================] - 11s 18ms/step - loss: 7.2046e-05
Epoch 7/10
619/619 [==============================] - 11s 18ms/step - loss: 6.8795e-05
Epoch 8/10
619/619 [==============================] - 14s 23ms/step - loss: 6.6264e-05
Epoch 9/10
619/619 [==============================] - 16s 26ms/step - loss: 6.4291e-05
Epoch 10/10
619/619 [==============================] - 16s 26ms/step - loss: 6.2743e-05
```

Bitcoin model training loss values:

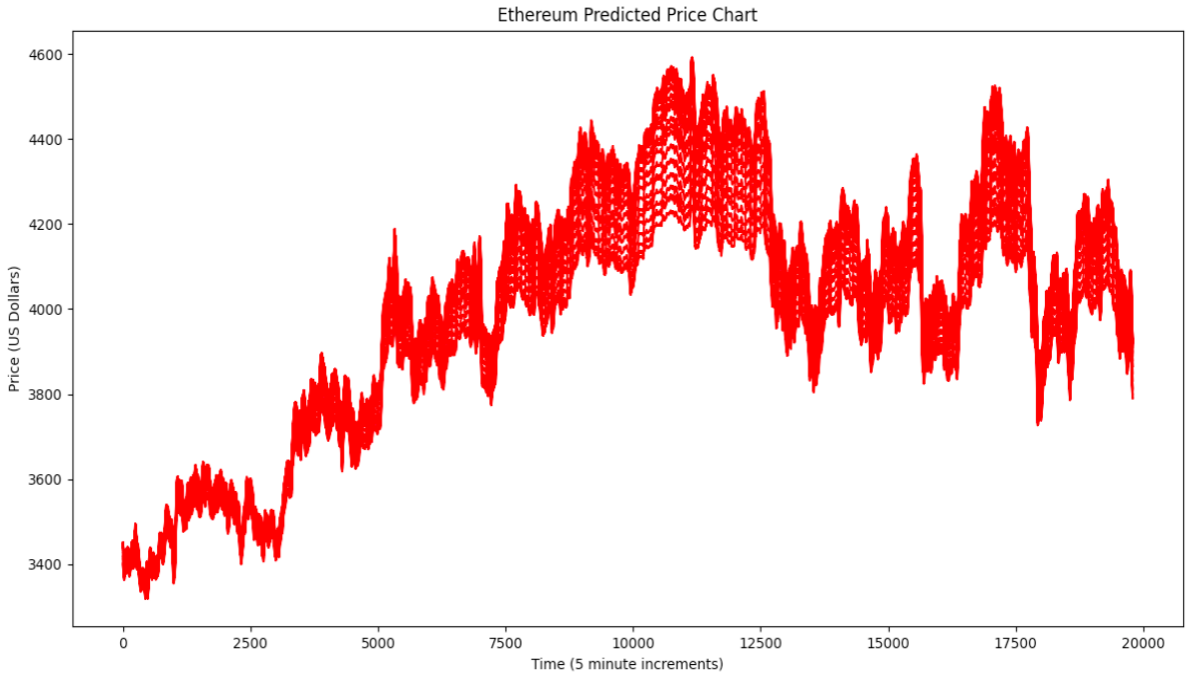
```
Epoch 1/10
619/619 [==============================] - 22s 26ms/step - loss: 0.0013
Epoch 2/10
619/619 [==============================] - 18s 29ms/step - loss: 2.0442e-04
Epoch 3/10
619/619 [==============================] - 14s 23ms/step - loss: 1.4454e-04
Epoch 4/10
619/619 [==============================] - 16s 26ms/step - loss: 1.4311e-04
Epoch 5/10
619/619 [==============================] - 13s 21ms/step - loss: 1.4101e-04
Epoch 6/10
619/619 [==============================] - 18s 29ms/step - loss: 1.3973e-04
Epoch 7/10
619/619 [==============================] - 14s 22ms/step - loss: 1.3919e-04
Epoch 8/10
619/619 [==============================] - 14s 23ms/step - loss: 1.3912e-04
Epoch 9/10
619/619 [==============================] - 13s 20ms/step - loss: 1.3922e-04
Epoch 10/10
619/619 [==============================] - 12s 20ms/step - loss: 1.3927e-04
```

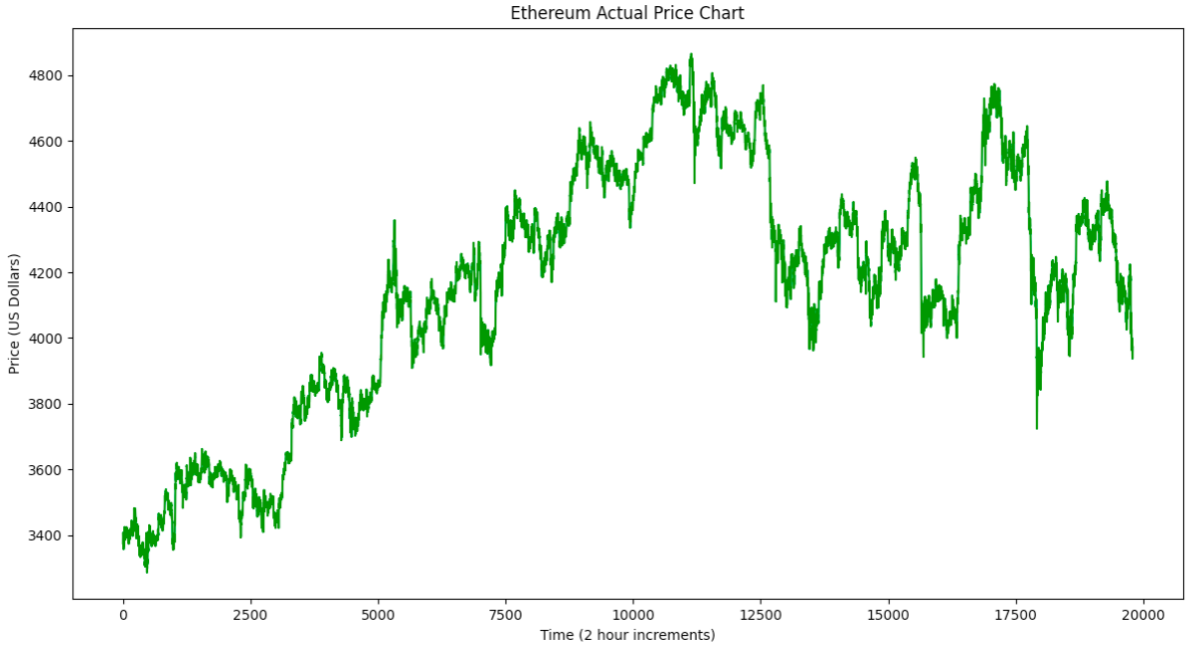
```
Ethereum Results:

619/619 [==============================] - 3s 4ms/step - loss: 0.0024
0.0023888680152595043
Bitcoin results:

619/619 [==============================] - 3s 4ms/step - loss: 0.0030
0.002979262499138713
```

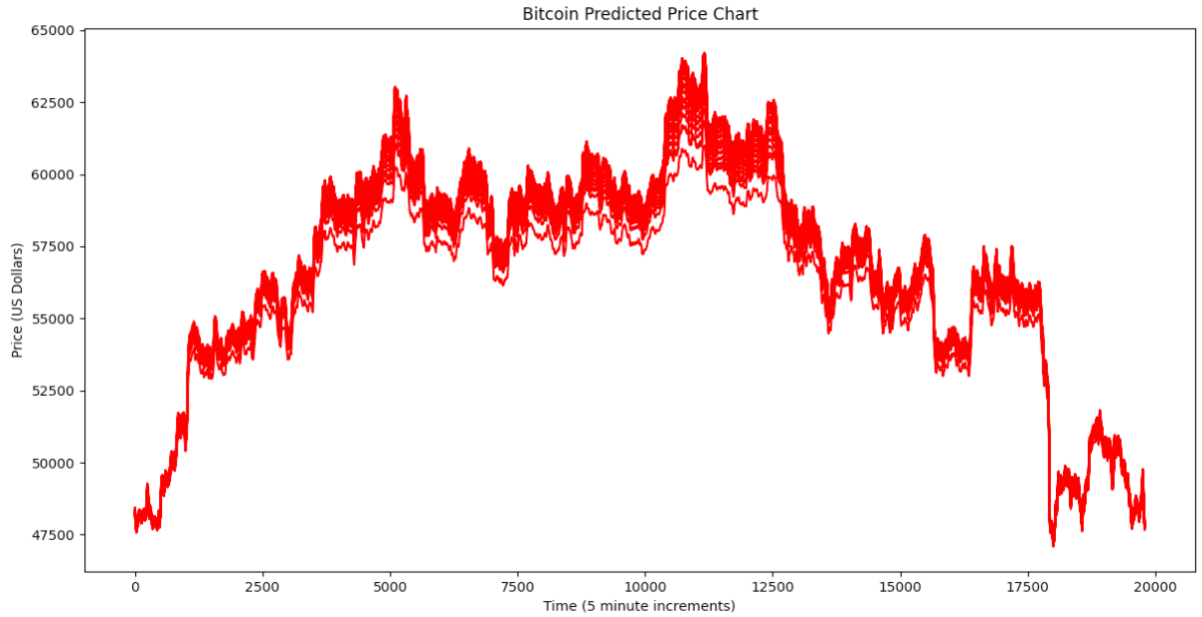
Ethereum Model Predictions vs Actual price charts

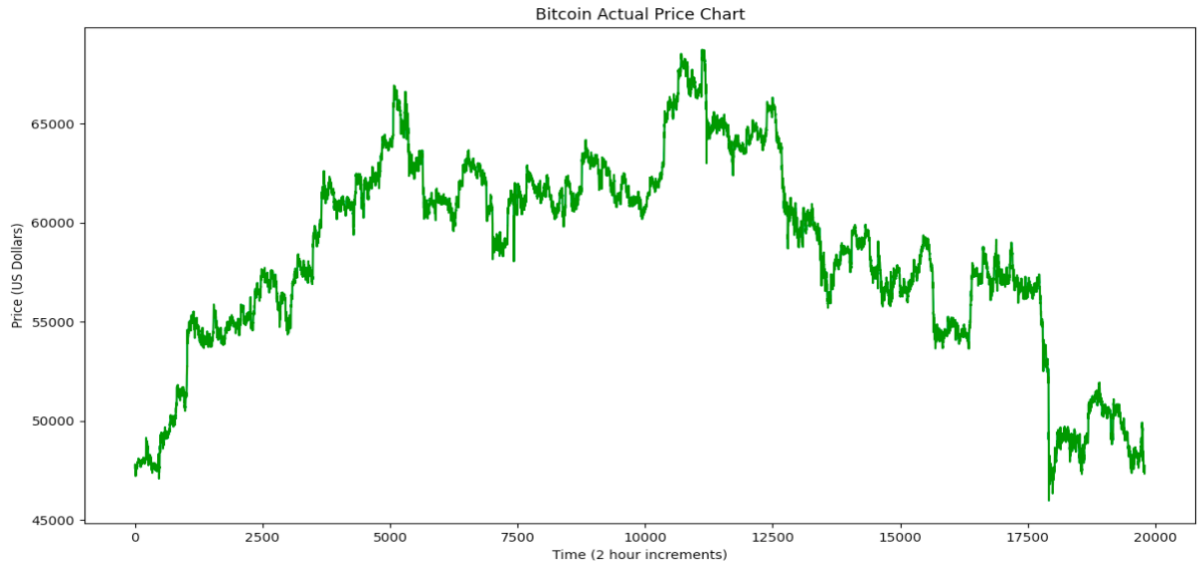




This final Ethereum model with the 128-batch size was the most accurate model in predicting the correct prices of Ethereum. This model performed like the previous 2 in terms of future pattern prediction.

Bitcoin Model Prediction vs Actual price charts





This final Bitcoin model with a 128-batch size, like the Ethereum model counterpart was very accurate in predicting the future prices of bitcoin. This model also performed similarly to the previous models in terms of future pattern recognition.

In all, I believe these Bitcoin and Ethereum LSTM models performed exceptionally well at predicting the future price trend regardless of batch size. Batch size, however, was very helpful in predicting more accurately what the actual prices would be on this trend. For example, the first two Bitcoin models were consistently underestimating how high the high peaks should go. The final example with a 128-batch size was much more accurate at predicting the right prices for these peaks as well as predicting the trend and when the peaks are likely to occur.

VI. LESSONS LEARNED

From doing this project I learned a lot about many different python packages and libraries. This was the first time that I worked so in-depth with the python Keras library and I got more practice working with Sklearn to preprocess data. I learned that not only is it possible to predict future market prices of cryptocurrency, but that by using a LSTM model, these results can be very accurate.

In hindsight, there are definitely a few decisions I should have made to improve this project. I spent a great deal of time setting up the proper environment to run this script on my computer by installing all the packages 1 by 1 and running everything form the same script. If I could go back and redo this, I would have written this project file inside of google colab which already has all these libraries built in and ready for use without any downloading which would have saved me some time during the initial stages of the coding portion of this project.

I also would have chosen to use google colab for another and more important reason which is the time that was required to run my program. This program sometimes took 30 minutes to run the longer epochs on my machine locally where on colab I'm sure this time could have been reduced due to the better CPU power. This, in turn, could have allowed me to test this model even more by performing test runs with higher numbers of epochs.

VII. ACKNOWLEDGEMENTS

1. https://www.tensorflow.org/api_docs/python/tf/keras/layers/LSTM
- (Documentation and examples of Keras LSTM layers)

2. https://www.tensorflow.org/api_docs/python/tf/keras/layers/Bidirectional
- (Documentation and examples of Keras Bidirectional layers)

3. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html
- (Documentation and examples of Sklearn MinMaxScaler)

4. https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/
- (Good examples of LSTMS, including helpful example of a LSTM time series Bidirectional model)

5. https://keras.io/api/layers/core_layers/dense/
- (Documentation and examples of Keras dense layers)

6. https://www.tensorflow.org/guide/keras/train_and_evaluate
- (Documentation and examples of the evaluate function for determining accuracy and loss of the test data)

7. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html
- (Good examples and documentation about MinMaxScaler function in Sklearn)