Yash Murthy and Daniel Valverde

August 17, 2022

COMP 4449 - Data Science Capstone

Term Project Write Up

**Research Question**

How well do Long Short Term Memory neural networks predict the price of Ethereum and how well can we use these predictions to create a trading strategy?

**Introduction: Purpose and Significance**

The purpose of this project is to use Long Short Term Memory, a recurrent neural network to predict the price of Ethereum, and using those predictions to create trade signals and backtest how the model and strategy performed. The ability to predict price movement is a problem that if solved can be very lucrative. We looked to expand on some of the data science techniques learned over the entirety of our program in order to predict the price of Ethereum.

**Description of the Dataset**

In order to get up to date data, we connected to the coingecko.com API. This API allows us to pull almost any available cryptocurrency and check the price of that cryptocurrency on a given day, in either daily or hourly intervals. The nice thing about this API is that it allows you to pull the historic price for each cryptocurrency from the beginning of its availability on the exchanges. We chose to pull 5 years worth of data (2017 to Present) and utilized only the daily price of Ethereum throughout that time period.

**Methods**

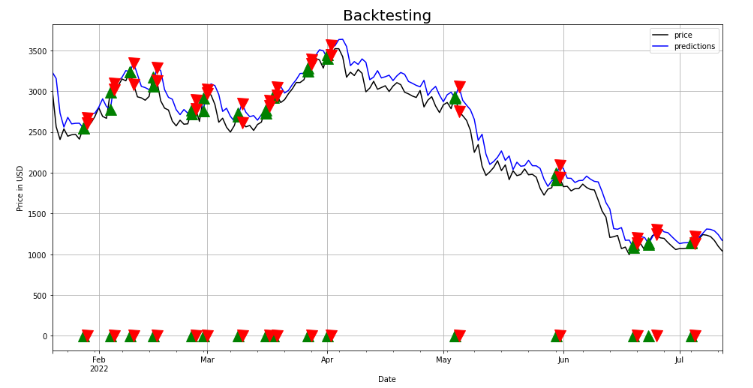
***LSTM:***

LSTM, or long-short term memory, is a type of recurrent neural network, RNN. It is quite frequently used in natural language processing and time series forecasting, such as stock market predictions. LSTM uses multiple sets of gates and cells to store and forget data points based on a probabilistic model. An important facet that separates it from other RNNs is that it uses both long and short term memory. The main architecture of a LSTM model involves multiple memory blocks called cells and areas of memory manipulation called gates. The gates are held within the cells. There are three main types of gates, forget gates, input gates and output gates. The forget gates are used to keep or discard valuable information for the model. Inputs are fed into the date and multiplied with weight matrices and then a bias value is added. These are then passed through a specific activation function which provides the binary output 1 to keep, and 0 to discard. The input gates take information using two input values, similar to the forget gate. A vector is created with those values and then multiplied to retain useful information. The output gate has a similar vector multiplication process as the input gate. This gate is used to extract the useful information from the cell to pass along to the next cell in the series. Once a prediction is created, it is reintroduced into the model and used to predict the next value within the sequence. Well known applications of LSTM models include language modeling, machine translation, image captioning, handwriting generation, and question answering chatbots.1

***Backtesting & Crossover:***

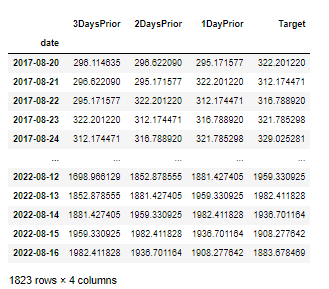
Backtesting is a method for seeing how well a strategy or model would have performed using historical data. We performed backtesting in order to simulate when our trading strategy would have performed trades, given the specified trading signals. The strategy we implemented was a crossover strategy between the predicted price and the actual price of ethereum on each day.

Our Crossover strategy performs a buy whenever the predicted price crosses the price line at the instant that the predicted price is less than the actual price. It performs a sell whenever the predicted price crosses back over at the instant the predicted price is greater than the actual price. Looking at the image below the green triangles indicate that a buy was performed and the red triangles indicate that a sell was performed.



**Data Preprocessing**

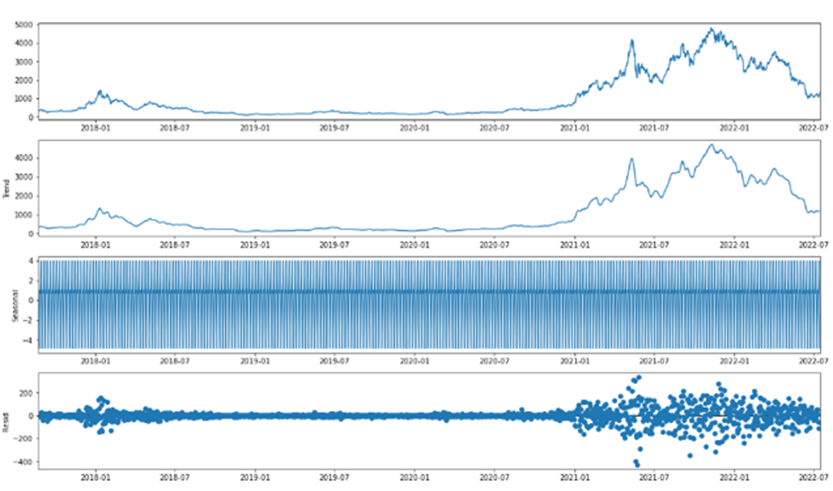
We transformed the data into a sliding window and we tried out different ranges for the sliding window. We utilized a 3-day and 60-day sliding window. Below you can see the 3-day sliding window that we implemented. The target column is the price on the date of its respective row. We are using the prior day's prices as the features for the LSTM model.



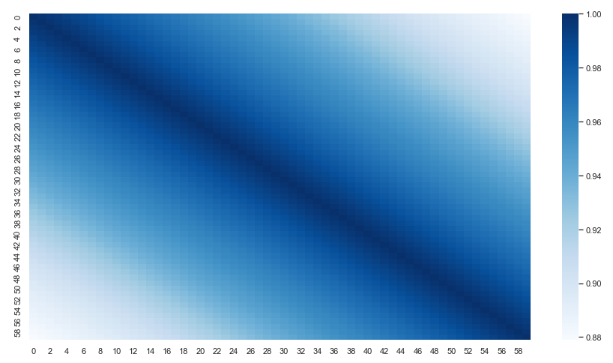
Prior to running the LSTM model we reshaped the data into a matrix and scaled the features. In the 3-day sliding window LSTM model, we did not scale the data, but in the 60-day sliding window LSTM model we performed a Min-Max Scaler. We reshaped and scaled the data to improve the training of the model. We reshaped the X-train to be a matrix of shape(length\_of\_training\_set, number\_of\_features, 1). The 1 indicates we are performing a Univariate forecasting model. We also reshaped the X-test to be a matrix of similar format.

**Exploratory Analysis**

The first thing that we did was check for trends and seasonality. Over the 5-year period, there were no observed seasonality or trends. In order to analyze this, we utilized the statsmodel library’s seasonal\_decompose function on the target variable, which is the price of Ethereum over time.



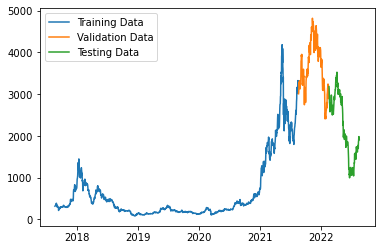
Next, we checked the correlation of the target variable versus the sliding window of prices. There was very high correlation amongst the features and the price, but the further back we went from the target price the less correlated it became. Variable 60 is the target variable in the correlation heatmap below.



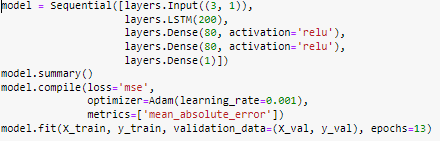
**Data Analysis, Visualizations & Modeling**

***3-day window LSTM model:***

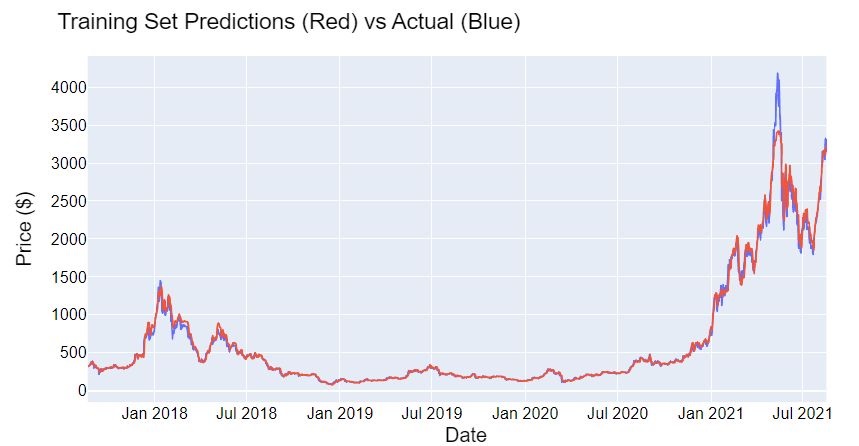
The first model we explored was the 3-day sliding window LSTM model.2 A function was created to set the range of the window along with the time period. A 3-day window was chosen as a short period window, while a 60-day window was chosen as the long period window. The complete dataset was split into an 80-10-10 Train-Validate-Test (seen below). The 3-day window analysis utilizes a validate set while the 60-day window does not.

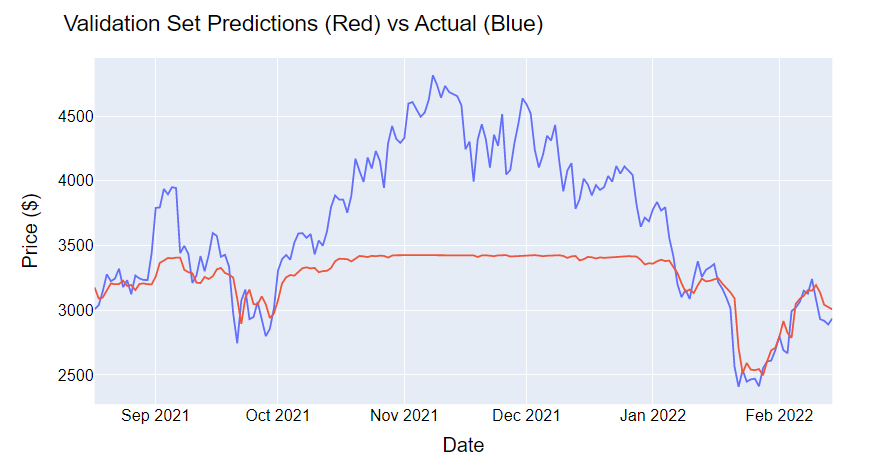


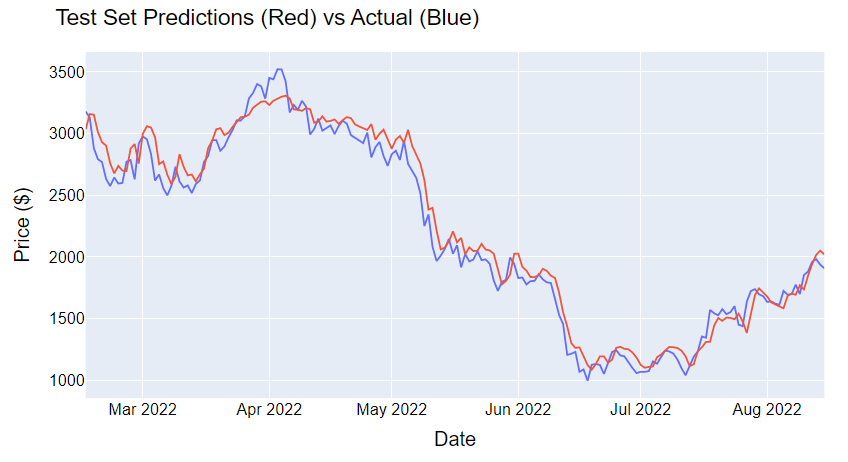
After the split, we moved onto model creation. We used the tensorflow.keras package and utilized a sequential model. We compiled the model using mean square error as the loss function and a mean absolute error metric. This tells us the average of how far off our prediction values are from the actual values, which suits stock prediction models well. After running through various epochs, we concluded that 13 epochs was the value that minimized the MSE and the mean absolute error.



We then fit the model onto the data and the actual vs predicted values for each set is shown below:



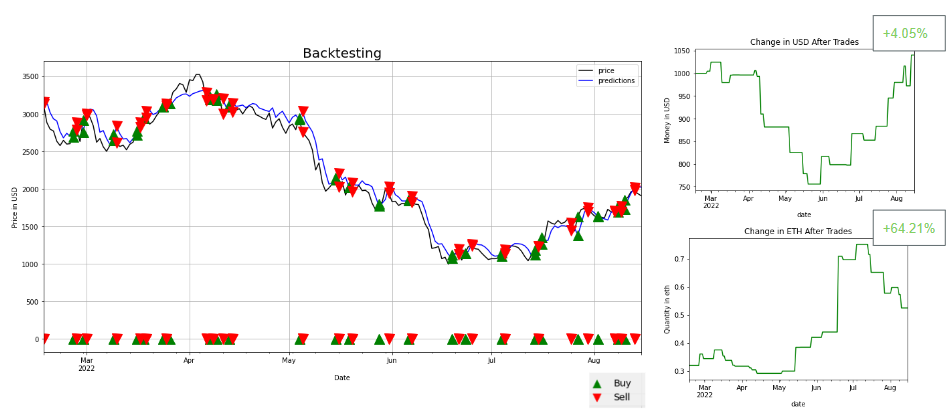




For the 3-day sliding window portion, we also looked to implement a recursive prediction method. Recursive prediction is using previous prediction data to predict stock price, compared to using actual pricing. We chose to do a 7-day window, which aimed to predict the most recent week of ETH price. We can see in the graph below that the predictions were a bit higher than the actual observed prices for that week. The recursive predictions are seen in pink on the right hand side of the graph, while the observed prices are seen in brown.

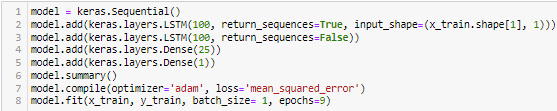


As for performance, the model did a great job of predicting the training and test sets. The validation set was not as accurate and the model struggled to compute the large volatility change between October 2021 and January 2022. We factored in RMSE, R-Squared, and the trade effectiveness (change in ETH and $ amount over time). The RMSEs of the train-validation- test set were 7.14, 383.65, and 52.23, respectively. The trade effectiveness was calculated using the backtesting and crossover strategy. The effectiveness was a 4.05% gain in capital and a 64.21% gain in ETH. The 3-day sliding window model performed second only to the 60-day sliding window based on this metric. A table comparing the model evaluations can be found in the Model Performance Summary Table section at the end of the paper.

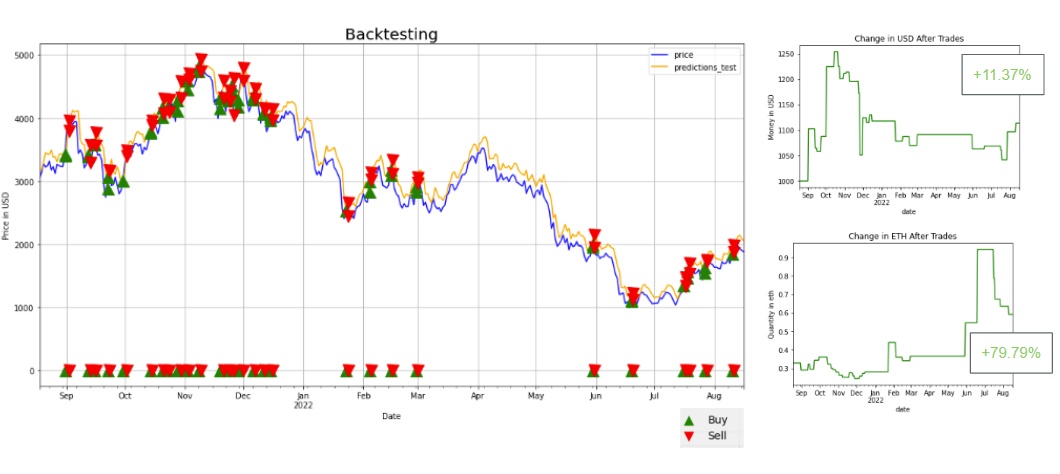


***60-day window LSTM model:***

For the LSTM model utilizing a 60-day sliding window, we performed an 80-20 train test split. We chose twenty percent as the test set because we had 5 years of total data and twenty percent gave us exactly one year for the test set. The neural network used can be seen below.3



Raising the epochs from 3 to 9 improved the performance a bit more. Lastly we increased each prediction by seven percent, in order for it to better fit the crossover strategy. After doing all of this the 60-day sliding window LSTM model performed the best out of all the models we tried. Factoring in RMSE, R Squared, and the effectiveness of the trades to increase money and ethereum over time, this model was a clear winner. Using our backtesting and crossover strategy of predicted price and price, over the course of a year we would have gained 79.79% more ethereum and increased our capital by 11.37% with this model. The RMSE of the training set was 6.79, while the RMSE of the test set was 38.77. The values of each of the models can be compared below in the Model Performance Summary Table.



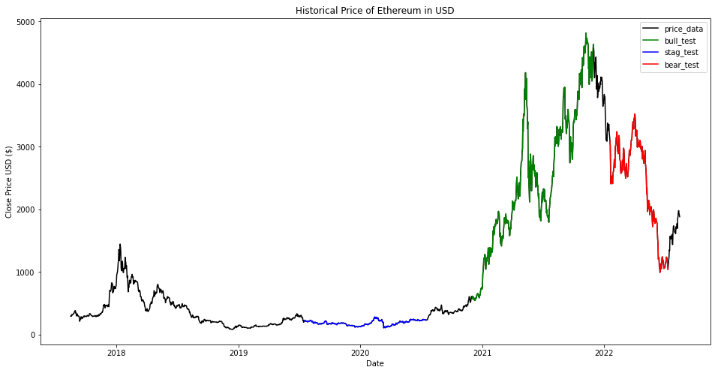
***Linear Regression Model:***

The Linear Regression model was the best performing model from the midterm project when I performed predictions on this same dataset, using both XGBoost and Linear Regression. The model performed decently when it came to R-Squared and RMSE on both the training and the test set, but when we back tested this model using a crossover strategy during this iteration of the project, the backtesting strategy performed quite poorly. The linear regression model was constantly crossing over so it was constantly performing trades.The details of which can be found in the Model Performance Summary Table at the end of this paper.

***Bull, Stagnant, and Bear LSTM Exploratory Model Performance:***

After having found a model that performed well, we were curious to see how the LSTM model would perform if the test set was specified to be either Bull, a Bear, or a Stagnant period. Bull means that the price of Ethereum is steadily increasing over a year. We defined a bear period as Ethereum decreasing over the period of 8 months because that was the largest portion of decline in our data. Our stagnant period was when the change in the price of Ethereum did not fluctuate too far from the mean during that period, which was 1 year.

We utilized the 60-day Sliding Window LSTM as the features and the same LSTM network as the 60-day Sliding Window LSTM model from before. The results of each of the models can be found in the Model Performance Summary Table below.



The least successful model was the LSTM model during the stagnant period. it definitely seemed to overfit during this time period because of how low the RMSE\_train was compared to the RMSE\_test. When you look at the R-Squared values of train versus test, the R-Squared values are substantially better for the training set. This model only performed two trades because there was only one crossover.

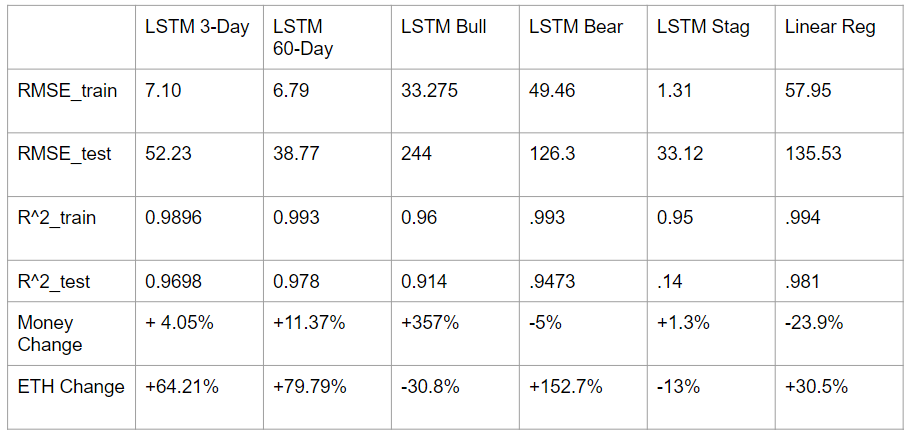
The LSTM Bull model did not perform as well as we would have hoped for the Bull period. The RMSE scores were much higher for both the training and test set, but the R Squared values were still pretty good. The change in money increased by 357% but we also lost 30% of the Ethereum we would have started with. Although the increase in money was substantial it is because over the period of the test set the price of Ethereum increased by over +660%.

The performance of the model during the Bear period was better than expected. Although we lost 5% of our money over this period of time. The price of Ethereum dropped by -66% over this period of time so losing 5% is not that bad. The amount of Ethereum we gained was about +152%. The RMSE and R-squared scores were not as good as our LSTM 60-day model but it did improve from the Bull model.

**Conclusion & Future Considerations**

We believe the LSTM model does not deal with data outside of its range very well if the data was not trained on increasing volatile data. This was deduced from the evidence obtained from the 3-day sliding window model. The poor predictions for the period of high volatility may have also been due to the lack of scaling the 3-day sliding window data. On the contrary, we were able to create models that performed very well during periods of price decline. It would be interesting to try and find a model and trading strategy that ideally performs well in all types of periods (bear, bull, and stagnant). Some future considerations include attempting implementation of other neural network methods, such as transformers, for predicting the price of Ethereum. We would also look into implementing a recursive prediction function to predict prices of future dates. Additionally, we could use more features other than sliding windows, as our input data, such as the historic sentiment, bid, and ask pricing data. Another future consideration would be to utilize other trade signals other than a crossover strategy between prediction and price. We look forward to implementing these strategies with a cryptocurrency exchange in order to make live trades in the future.

**Model Performance Summary Table**



**Works Cited**

1. Chugh, Aakarsha. “Deep Learning: Introduction to Long Short Term Memory.” *GeeksforGeeks*, 29 Sept. 2021, <https://www.geeksforgeeks.org/deep-learning-introduction-to-long-short-term-memory/>.
2. Hogg, Greg. “Microsoft Stock Forecasting with LSTMs.” *Google Colab*, Google, 26 Mar. 2022, https://colab.research.google.com/drive/1Bk4zPQwAfzoSHZokKUefKL1s6lqmam6S?usp=sharing#scrollTo=khQN7tf1BJCB.
3. Teo, Bee Guan. “Stock Prices Prediction Using Long Short-Term Memory (LSTM) Model in Python.” *Medium*, The Handbook of Coding in Finance, 26 Oct. 2021, https://medium.com/the-handbook-of-coding-in-finance/stock-prices-prediction-using-long-short-term-memory-lstm-model-in-python-734dd1ed6827#:~:text=Save-,Stock%20Prices%20Prediction%20Using%20Long%20Short,Memory%20(LSTM)%20Model%20in%20Python&text=Long%20Short%2DTerm%20Memory%20(LSTM)%20is%20one%20type%20of,useful%20in%20predicting%20stock%20prices.