The Effect of Elon Musk's Tweets on the Crypto Market

Bitcoin, Dogecoin, Ethereum

Dharam Madnani, Idilsu Guney

dmadnani, iguney1

CSCI 1470 Deep Learning Final Project
Github Link: https://github.com/idilsuguney/finalproject

INTRODUCTION

Elon Musk's Twitter activity leads to more than just people's reactions; his tweets have caused cryptocurrency prices to skyrocket, plateau or significantly decrease. As a group, we have always been interested in cryptocurrencies and the opportunity to look at a data-driven approach of how an influential figure such as Elon Musk can affect such prices is extremely interesting, moreso seeing the relative effectiveness of his tweets on cryptocurrencies across a time period. We chose to only look at Elon Musk's tweet impact on Bitcoin, Dogecoin and Ethereum.

This kind of problem uses a sentiment analysis model using the techniques we learned from Deep Learning (CSCI 1470) in order to categorize Musk's tweets with sentiments and a Granger-Causality test to determine whether Musk has an overall effect on cryptocurrency prices using the techniques from Data Science (CSCI 195A).

DATA

There were 5 datasets we used:

1. **A Kaggle dataset of tweets from Twitter sentiment analysis**
    - This dataset contained 41,156 tweets scraped from Twitter. Since this dataset was previously used for sentiment analysis by other projects, we expected it to work well with our sentiment analysis model.
    - We removed all unnecessary columns and only kept the original tweet and the label, which is the sentiment. The tweet column required filtering out a lot of unnecessary characters and punctuation, including dots, commas, dashes, etc. We used regular expressions to remove patterns from the tweets, which was extremely useful for removing hashtags, mentions, and hyperlinks.
    - The labels consisted of 5 possible categories: extremely negative, negative, neutral, positive, extremely positive. We converted each of these categories to numbers, so that 0 would represent extremely negative, 1 would represent negative, 2 would represent neutral, 3 would represent positive, and 4 would represent extremely positive.
    - We used the following dataset:
        i. https://www.kaggle.com/cerolacia/covid-19-tweet-classification/data

2. **A Kaggle dataset of Elon Musks' tweets from 2011-2021**
    - This dataset contained 11,715 tweets scraped from Elon Musk's Twitter account between the years of 2011 and 2021.
    - Similarly, this dataset required significant preprocessing. We removed all

unnecessary columns and only kept the tweet ID, tweet date and the original string. We then had to cast each column to their respective appropriate data types (tweet ID as int, tweet date as a datetime object, tweet string as a string). For the tweet column, we filtered out unnecessary characters and punctuation, as well as mentions, hashtags, and hyperlinks using regular expressions.

- We filtered out this data to only obtain tweets that have references to cryptocurrencies. After implementing simple preprocessing and converting all tweet inputs to lowercase, we checked whether each tweet has references to "bitcoin", "dogecoin" , and "ethereum". We also included other words that may refer to these cryptocurrencies, which included "btc", "doge", "eth", "ether", "bitcoins", "dogecoins", and "ethers".
- Additionally, we chose to only look at the years of 2019 and 2020 as these years contained the most number of tweets that included references to cryptocurrencies.
- We used the following dataset:
    i. https://www.kaggle.com/ayhmrba/elon-musk-tweets-2010-2021?select=2011.csv

3. **Yahoo Finance Time-series dataset.**
    - No significant preprocessing was needed.
    - We used the adjusted close price for all 3 cryptocurrencies for the period between January 1st, 2019 and January 1st, 2021. We added two columns: "Returns" and "Tweet Shock" which will be explained later.
        i. Bitcoin time-series
            1. https://finance.yahoo.com/quote/BTC-USD/history
        ii. Dogecoin time-series
            1. https://finance.yahoo.com/quote/DOGE-USD/history/
        iii. Ethereum time-series
            1. https://finance.yahoo.com/quote/ETH-USD/history/

METHODOLOGY

Building Our Own Sentiment Analysis Model

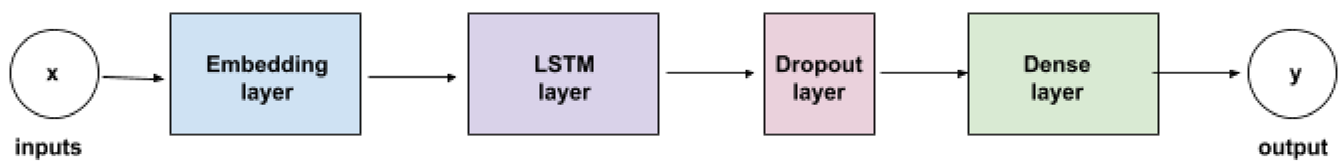
We built our own sentiment analysis model, which was trained on the larger dataset of tweets obtained from Kaggle (first dataset in the Data section of this report). After preprocessing the tweets to remove any punctuation and characters and re-labeling sentiments as (0, 1, 2, 3, 4) to represent ("extremely negative", "negative", "neutral", "positive", "extremely positive") , respectively, we categorized the tweets as our inputs variable and the sentiments as our labels variable and converted them into arrays.

We then took 80% of our inputs (tweets) and labels (sentiments) and defined them as our train inputs and train labels. The remaining 20% were defined as test inputs and test labels.

As a next step, we obtained all the unique words in our inputs and tokenized them. We then put all the unique words in a dictionary and assigned each unique word to a unique number. Our output consisted of a large dictionary, where each key's value was an array of numbers representing each tweet, such that each number in the array represented a word in the tweet. We also made sure to drop the keys where the length of the tweet was zero and added padding to the array of numbers, since sentences were in different lengths. We added the digit “0” as the padding and made sure 0 was not mapped to any word when converting words to numbers.

At the end, all the arrays were 64 digits in length as the longest sentence in the dataset was 64 words. Finally, we created a 2-D array for both of our training and test inputs which were of shape (32926, 64) and (8231, 64), respectively. This meant our training input contained 32,926 tweets that were all of 64 digits in length including padding and our test input contained 8,231 tweets that were all of 64 digits in length including padding.

After this step, we were done with the preprocessing of data and worked on creating our own sentiment analysis model. Our model architecture is as follows:



For our model, we used several different hyperparameters: vocab_size, LSTM_size, embedding_size, batch_size, learning_rate, as well as an Adam optimizer. The vocab size was equal to the number of unique words in our input, which was 54,261. We passed the vocab size as a parameter to the model. We used 150 as our embedding size and passed the size of vocabs and embedding into our embedding layer. Similarly, for our LSTM size, we used 150 and passed it into our LSTM layer. The size of the dense layer was 5, since we had 5 possible categories for the labels. We used the “softmax” activation for our dense layer, since we needed the output layer of our neural network model to predict a multinomial probability distribution. Hence, the softmax activation was very useful in scaling our logits into probabilities. We set the learning rate to be equal to 0.01 and passed it into our Adam optimizer. Lastly, we used 100 as our batch size.

After defining our hyperparameters, we implemented our model following the architecture above. In the call function, we passed our inputs through the layers. We wanted to use a Dropout Layer with a dropout rate of 0.3 to prevent overfitting of our model. In the loss function, we used Sparse Categorical Entropy from the Keras losses library. We preferred using this loss function rather than the Categorical Entropy loss, as our labels were mutually exclusive. In the train function, we shuffled our train inputs and labels, implemented batching of the data and calculated the loss by adding each loss to a loss list array. We then called tf.reduce_mean on the loss. In the accuracy function, we called np.argmax on the results returned from the train function, which gave us the index of the maximum probability. Since we defined our labels to be equal to (0, 1, 2, 3, 4), the output of the argmax function gave us the label prediction of our model. We compared whether our logit equaled the label for each logit and calculated our percentage accuracy accordingly.

In the main function, we trained our model for a maximum of 10 epochs. We calculated the loss and accuracy per epoch and also visualized losses for each epoch and accuracies for each epoch using the Matplotlib pyplot library.

Testing Our Own Sentiment Analysis Model on Musk's Tweets

Once our sentiment analysis model was built and trained correctly, we then fed our set of Elon Musk's tweets into the model which gave us the sentiment of each tweet.

We repeated the same preprocessing steps for the dataset of Musk's tweets. However, after removing any punctuation and characters using regular expressions, we also filtered out data to only have tweets that refer to either bitcoin, dogecoin, and ethereum. We implemented this by creating a dictionary, where tweet ids are keys and the content of the tweet are values. We then searched for references to "bitcoin", "dogecoin", "ethereum" as well as to "btc", "doge", "eth", "ether", "bitcoins", "dogecoins", and "ethers" by looping through each of the values in the dictionary. We specifically wanted to focus on the time period between January 1st, 2019 to January 1st, 2021, since Musk had the most impact on the cryptocurrency market between these years. Between this time period, Musk had a total of 19 tweets that referred to either bitcoin, dogecoin, or ethereum.

Then, similar to the procedure we followed with the larger tweets dataset, we obtained all the unique words in Musk's tweets and tokenized them. We put all the unique words in a dictionary and assigned each unique word to a unique number. Our output consisted of a large dictionary, where each key's value was an array of numbers representing each tweet, such that each number in the array represented a word in the tweet. We also made sure to drop the keys where the length of the tweet was zero and added padding to the array of numbers, since sentences were in different lengths. We added the digit "0" as the padding and made sure 0 was not mapped to any word when converting words to numbers.

At the end, all the arrays were 25 digits in length as the longest tweet Musk had in the dataset was 25 words. Finally, we created a 2-D array for our inputs which was of shape (19, 25). This meant our input contained 19 tweets that were all of 25 digits in length including padding. We used the previous training inputs to train the model and this 2-D array as the testing inputs.

Since we didn't have any labels for Musk's tweets, we couldn't calculate the accuracy of our model. Yet, since our model had around 75% accuracy rate for the testing input generated from the larger Twitter dataset, we were confident that we would get reasonable results after feeding Elon Musk's tweets into the model.

Similarly, in the main function, we trained our model for a maximum of 10 epochs and calculated the loss and accuracy per epoch. Additionally, we added a visualize_results function to our model, which we called in the main function. This function created a dataframe that included the tweet id, the date the tweet was created, the content of the tweet, the sentiment in the integer form (0, 1, 2, 3, 4), and the sentiment in the string from ("extremely negative", "negative", "neutral", "positive", "extremely positive") of each tweet. We also exported the results documented in this dataframe to an Excel file, which made it easier for us to create visualizations.

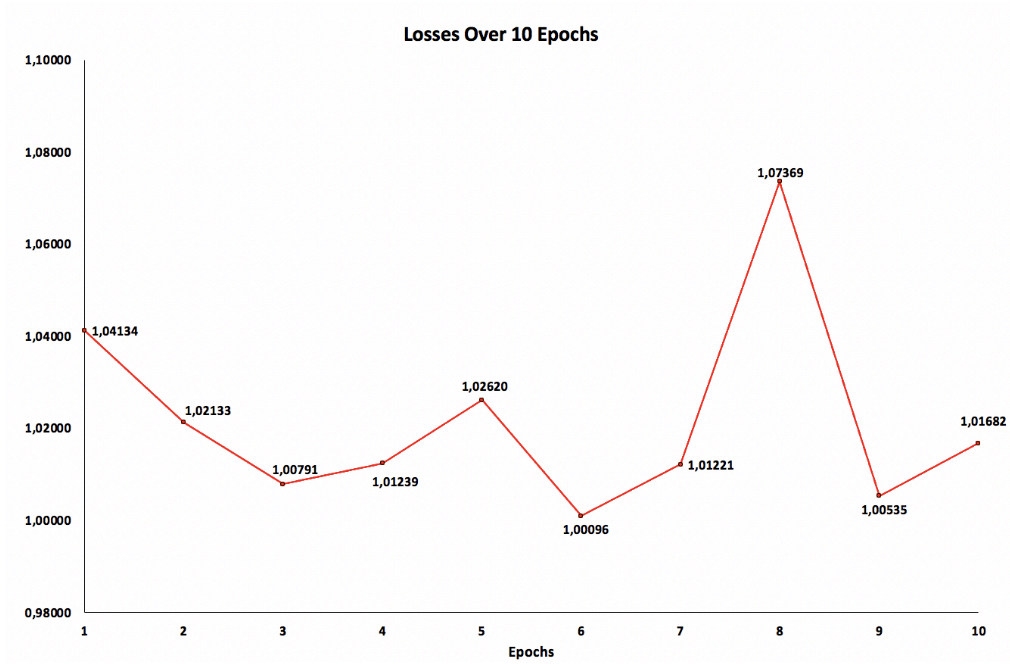
Data Analysis on the Effect of Musk's Tweets on Cryptocurrency Prices

We plotted each of these tweets and corresponding sentiments on a timeseries plot for each cryptocurrency to examine the effect they had on cryptocurrency price and volatility. We added a "Returns" column which was calculated by ((Day's Adjusted Closing - Previous Day's Adjusted Closing) / Previous Day's Adjusted Closing) which was used to determine volatility. We used a statistical technique called the Granger causality test which was used to determine whether there is a causal relationship between Musk's tweets and the increase in cryptocurrency prices, with Bitcoin, Dogecoin and Ethereum serving as a proxy for the entire market. For the Granger causality test, we had to create a set of dummy variables based on the sentiment of Elon Musk's tweets. This was added in as "Tweet Shock" with 0 indicating there was no tweet and 1 indicating there was a tweet. This was cross-referenced with the volatility and price charts, and a value of 1 was added for 20 days prior to the tweet as well as after.

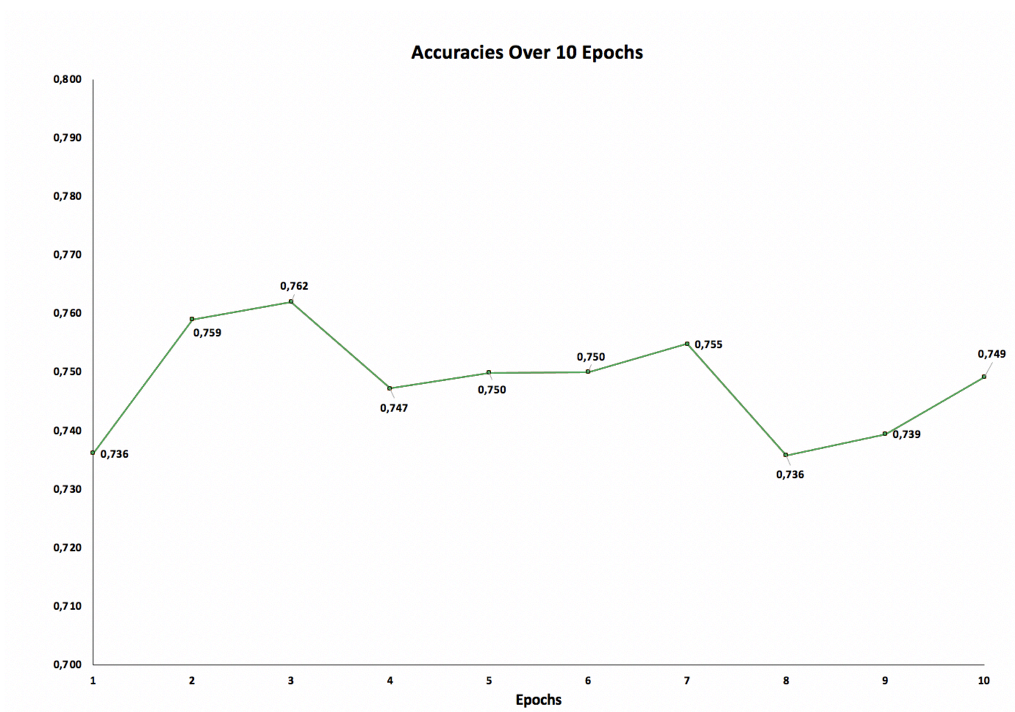
RESULTS

Results From Our Sentiment Analysis Model Trained on the Large Twitter Dataset

After training our model for 10 epochs, we got a loss value of around 1.0168. The graph below shows our loss values for all 10 epochs. An important trend to notice is that our loss doesn't go down as the number of epochs increases. We think this might be because our model is overfitting the data. Even though we included a dropout layer with a dropout rate of 0.3 to prevent overfitting, in order to decrease loss through epochs we could have decreased the learning rate or increased the dropout rate.



After training our model for 10 epochs, we got an accuracy value of around 0.749, which corresponds to around 74.9%. The graph below shows our accuracy values for all 10 epochs. Similar to the trend mentioned above, our accuracy value doesn't increase as the number of epochs increases due to the problem of overfitting.

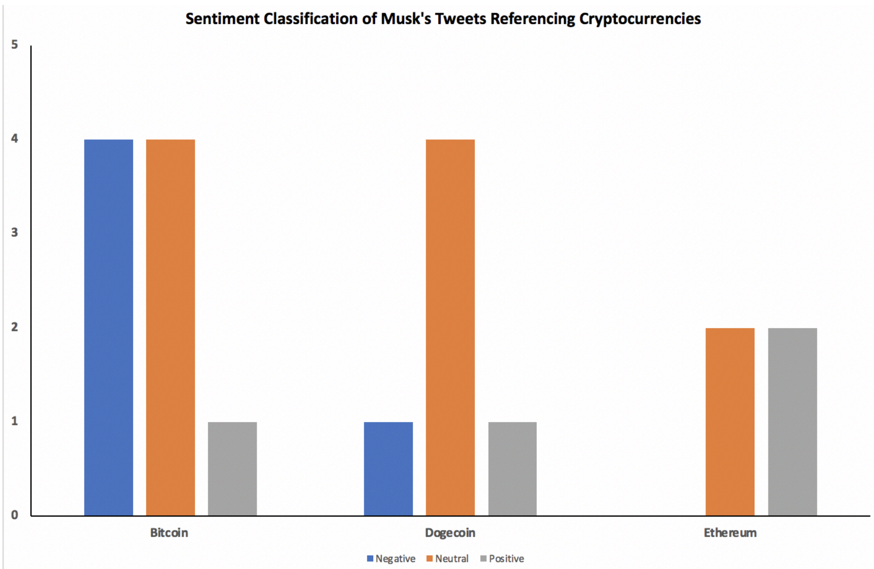


Results From Our Sentiment Analysis Model Trained on Elon Musk's Tweets

After running our sentiment analysis model on the preprocessed Elon Musk tweets data for 10 epochs, we got the following results, which belongs to the dataframe output of the 10th epoch. It's interesting to note that our model didn't classify a tweet as extremely negative or extremely positive even though we had these sentiments as possible labels. Due to the limited number of tweets, it's possible to see that our model wasn't very accurate. However, we think this was mostly due to Musk's use of slang and sarcasm, which we will further discuss in the challenges section. Below shows the model output of each of Elon Musk's tweets and their respective sentiment score and interpretation.

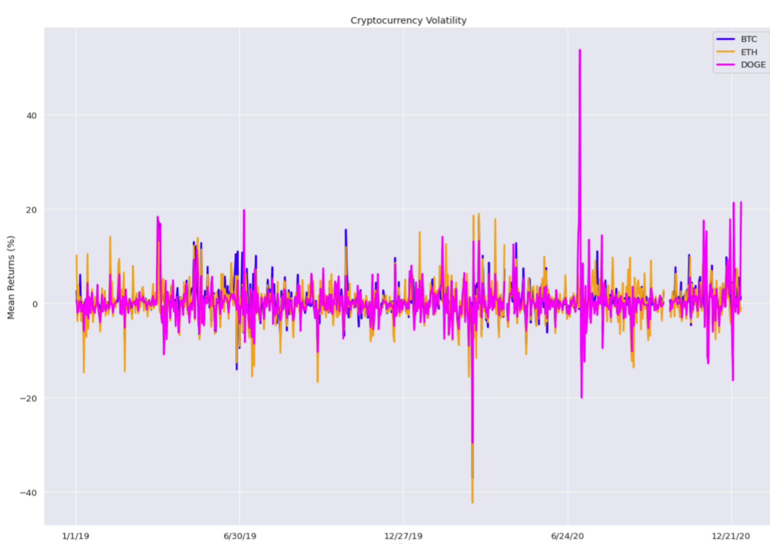
Date	Tweet	Sentiment	Interpretation	Crypto
2019-02-21 10:44:32	that said i still only own btc which a friend sent me several years ago don't have any crypto holdings	1	Negative	BTC
2019-02-21 10:50:48	whoever owns the early btc deserves a nobel prize in delayed gratification	2	Neutral	BTC
2020-01-10 06:53:10	bitcoin is *not* my safe word	2	Neutral	BTC
2020-05-01 22:44:56	how much for some anime bitcoin	1	Negative	BTC
2020-05-15 22:03:01	pretty much although massive currency issuance by govt central banks is making bitcoin internet 👻 money look solid by comparison	3	Positive	BTC
2020-05-15 22:51:44	i still only own bitcoins btw	1	Negative	BTC
2020-11-16 22:02:51	🎶 toss a bitcoin to ur witcher 🎶	2	Neutral	BTC
2020-12-20 08:21:25	bitcoin is my safe word	2	Neutral	BTC
2020-12-20 09:24:37	bitcoin is almost as bs as fiat money	1	Negative	BTC
2019-04-02 09:24:39	dogecoin might be my fav cryptocurrency it's pretty cool	2	Neutral	DOGE
2019-04-02 20:16:58	dogecoin rulz	2	Neutral	DOGE
2019-04-02 20:38:38	dogecoin value may vary	3	Positive	DOGE
2020-04-25 13:29:52	dogecoin mode	2	Neutral	DOGE
2020-07-18 00:53:43	excuse me i only sell doge	1	Negative	DOGE
2020-12-20 09:30:04	one word doge	2	Neutral	DOGE
2019-04-30 01:15:53	ethereum	2	Neutral	ETH
2019-04-30 01:44:32	stop giving away free eth 🤣🤣	2	Neutral	ETH
2019-04-30 07:24:40	what should be developed on ethereum	3	Positive	ETH
2020-07-02 16:42:52	i'm not building anything on ethereum not for or against it just don't use it or own any	3	Positive	ETH

As we can see from the graph below, Musk's tweets regarding Bitcoin were mostly categorized negative or neutral, those regarding dogecoin were mostly categorized neutral, and those regarding ethereum were mostly categorized neutral or positive. This graph is based on the results of our 10th epoch, where the y-axis represents the number of tweets.



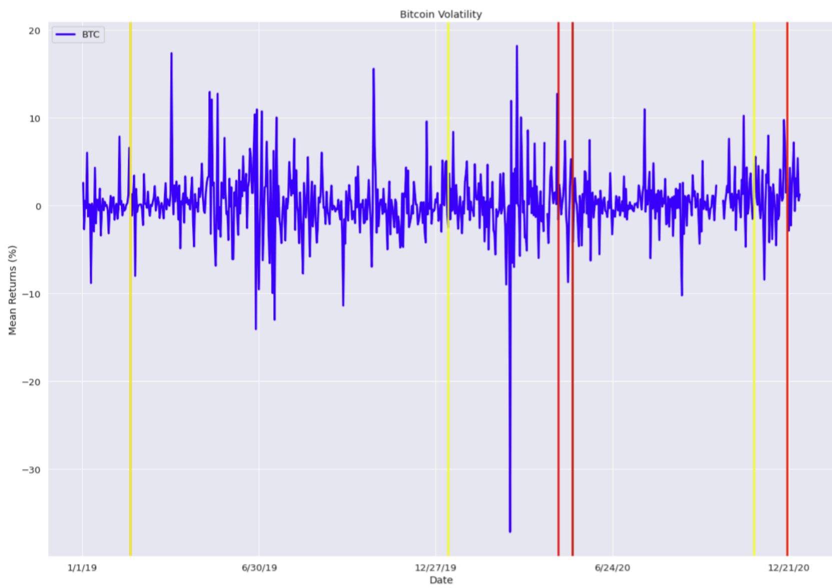
Results from Data Analysis

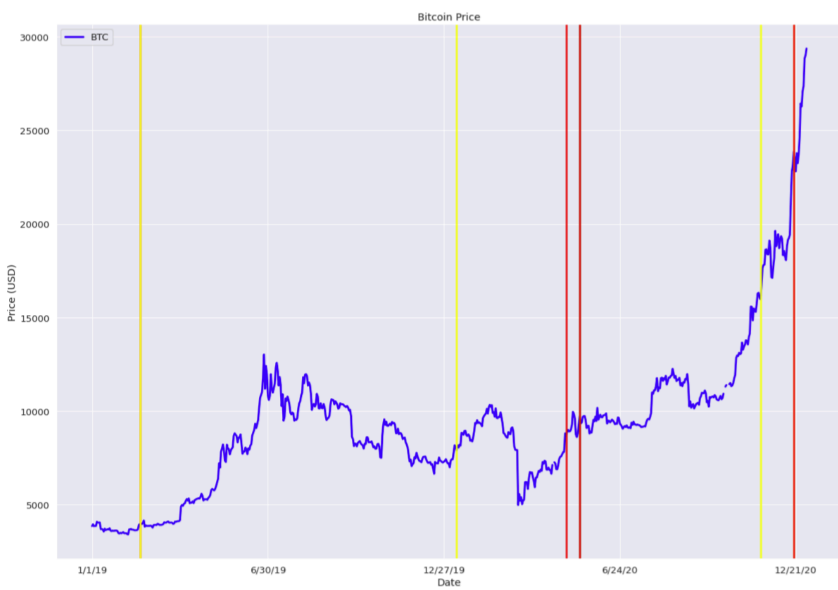
The graph below shows a timeseries of the volatility of Bitcoin, Dogecoin and Ethereum over the 2 year period.



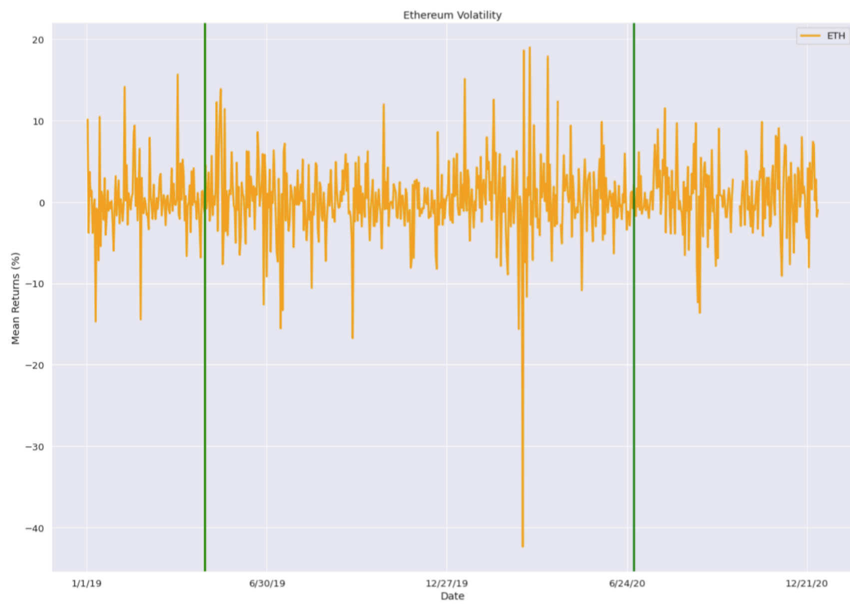
For the following individual volatility and price graphs for Bitcoin, Dogecoin and Ethereum, all 19 tweets have been plotted accordingly by a vertical line. For tweets that have a negative interpretation, the vertical line is red. For tweets that have a neutral interpretation, the vertical line is yellow. For tweets that have a positive interpretation, the vertical line is green. The influence of the sample tweets on each crypto's volatility and price acted as a dummy variable; this process was repeated across each cryptocurrency.

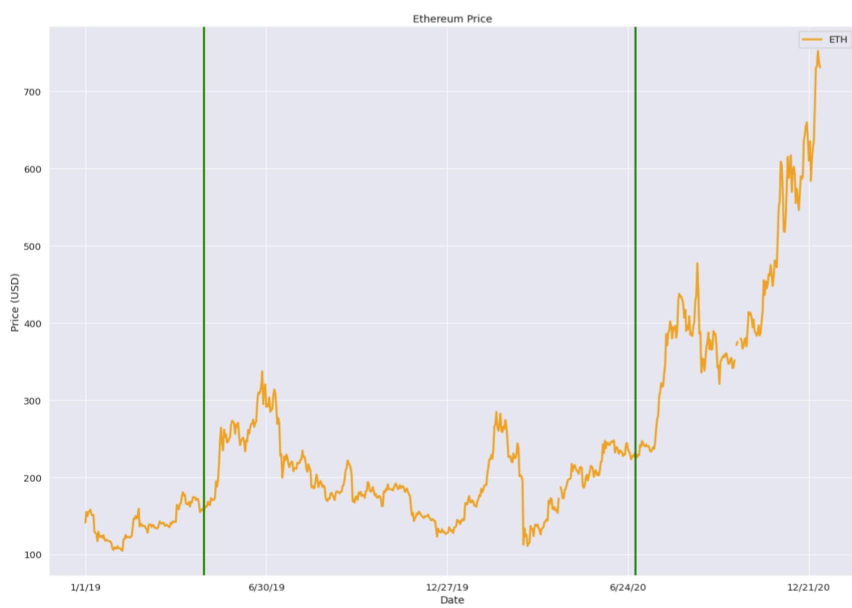
Below are Bitcoin's volatility and price graphs.



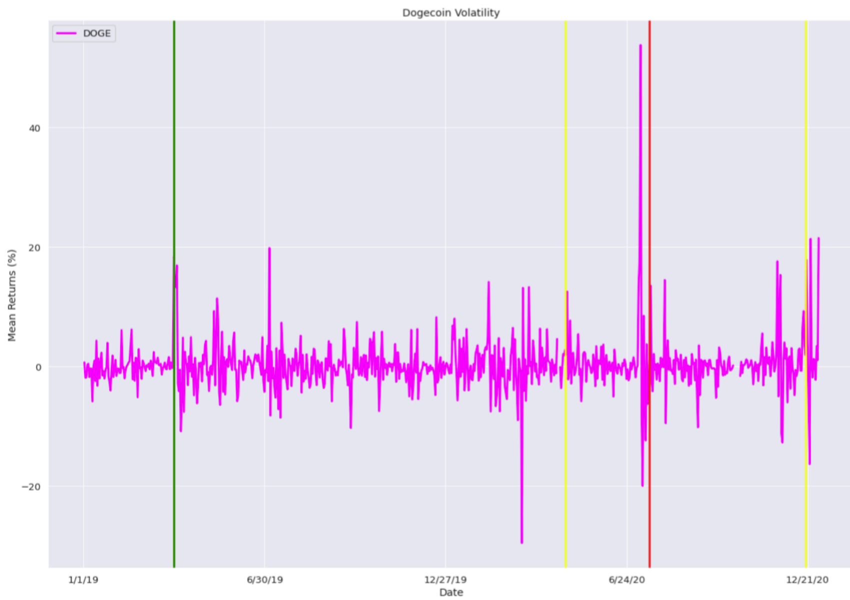


Below are Ethereum's volatility and price graphs.





Below are Dogecoin's volatility and price graphs.



We used a hypothesis test for the Granger causality test with the following hypotheses:

- H0: Elon Musk's tweets do not cause a change in the price of the tested cryptocurrency
- H1: Elon Musk's tweets cause a change in the price of the tested cryptocurrency

Cryptocurrency	Granger Causality Final Lag Number	Lowest P-Value	Highest P-Value
Bitcoin	3	0.387 (SSR based chi2 test)	0.394 (SSR based F test and Parameter F test)
Dogecoin	1	0.0065 (SSR based chi2 test)	0.0068 (SSR based F test and Parameter F test)
Ethereum	4	0.0989 (SSR based chi2 test)	0.1041 (SSR based F test and Parameter F test)

The Granger causality test findings for Bitcoin and its corresponding dummy variable show insignificant findings with p-values ranging from 0.387-0.394, all of which are greater than 0.05 which means the p-values are insignificant. There is therefore not enough evidence to reject the null hypothesis, showing that the dummy variable used to capture Elon Musk's tweet effect does not directly affect Bitcoin's price.

The Granger causality test findings for Dogecoin and its corresponding dummy variable show significant findings with p-values ranging from 0.0065-0.0068, all of which are less than 0.05 meaning that the test is statistically significant. There is therefore enough evidence to reject the null hypothesis, meaning that Elon Musk's tweets do indeed cause a change in the price of Dogecoin.

The Granger causality test findings for Ethereum and its corresponding dummy variables show insignificant findings with p-values ranging from 0.0989-0.1041, all of which are greater than 0.05 which means the p-values are insignificant. There is therefore not enough evidence to reject the null hypothesis, showing that the dummy variable used to capture Elon Musk's tweet effect does not directly affect Ethereum's price.

CHALLENGES

Throughout this project, one of the main challenges that we faced was increasing the accuracy rate of our sentiment analysis model. Even though we were able to obtain a fair accuracy rate of around 74.9%, our accuracy didn't improve after training for 10 epochs, which was caused by the overfitting of our model. While changing the hyperparameters of our model (including the learning rate and the dropout rate) did not help to solve this problem, we think our model's accuracy rate could be improved by modifying the architecture.

Additionally, our main challenge was to accurately classify Musk's tweets with sentiments. This was caused by Musk's frequent use of sarcasm and slang in his tweets that made it difficult for our model to accurately classify tweets' sentiments. Even though we received a good accuracy rate for our model when we ran it on the larger dataset of tweets, our model didn't perform equally well with Musk's tweets. This also affected the results of our data analysis and led to some inaccurate results, such as Musk's tweets having a non-significant effect on the trading prices of bitcoin after we ran the Granger Causality test.

REFLECTION

- Overall, we are happy with how the project turned out. Relative to our base goals, we kept extending the timeline for when our model was to be completed by due to its complexity and issues to have it produce a high accuracy.
- While our model worked as expected for the larger dataset of tweets (for which we received >70% accuracy), our model performed poorly for Elon Musk's dataset of tweets due to his use of slang and sarcasm. Nevertheless, we think we were successful in creating a sentiment analysis model as it's not possible to train a model that would completely eliminate and correctly understand sarcasm in text.
- Our approach changed overtime as we were initially going to train a basic classifier to look at the direct effect of a tweet on a cryptocurrency's price. However, we felt this would be extremely similar to past projects, and given that this project is a capstone, we wanted to take our project in a new direction. This was a big pivot, as it meant rethinking our entire project and knowledge from class and online research showed us that conducting sentiment analysis on tweets and analyzing the effect of sentiments on the trading prices would be perfect for our project. If we had to do the project over again, we would modify the model's architecture as well as use a larger date range of tweets for the analysis and examine more cryptocurrencies. Additionally, the project workload across 2 people was quite tough, so in the future having a third or fourth member for this project would have significantly helped.
- If we had more time, we could further improve our accuracy for the model. Some of the sentiment classifications of Musk's tweets which were clearly not-neutral were classified as neutral, and vice versa with neutral tweets being classified as positive or negative. Improving the model would therefore improve its accuracy and reduce the likelihood of an incorrect classification.
- One of the biggest learning lessons from this project is that the cryptocurrency market is much more volatile than the stock market, which makes it a lot harder to use data-driven approaches and predict what is going to happen next. Working with messy data and learning how to split strings and clean unwanted characters

was another big learning lesson from this project. Most of all, the entire process of searching the data, building the model and running statistical analysis after and seeing tangible results with the work produced was extremely gratifying.