TWEETS OF ELON MUSK AND THE PRICE OF DOGECOIN

Research Project

Florian Ringel
florian.ringel@uni-potsdam.de
816209

Akhyar Ahmed
ahmed2@uni-potsdam.de
804864

Jurate Vaistaraite
vaistaraite@uni-potsdam.de
805328

ABSTRACT

Cryptocurrencies are one of the latest and most discussed topics in the financial world. Every time a new one entered the market, influencers like Elon Musk started talking about them. With our paper, we investigated the connection between the price development of Dogecoin and the sentiment of tweets from Elon Musk that contain Dogecoin-related content. Since the data basis of a single influencer was not enough to work reliably with, we also included Dogecoin tweets from every other user. The data, scraped from Twitter's API, was cleaned, and partially annotated. More than 45,000 tweets were then analyzed using different sentiment analysis models. Our results show a decent connection between Musk's tweets and Dogecoin price but not between the latter and the group of every other Twitter user.

Keywords

Sentiment analysis, Twitter, Elon Musk, Cryptocurrency, Dogecoin.

INTRODUCTION

With the start of 2022, the trend of social media proliferating continues for another year. According to Datareportal (Kemp, 2020)**,** about 4,62 billion people were active on social media platforms worldwide till January this year. This is an increase of 424 million users from the year before. One more famous social media platform is the US American microblogging and social networking service Twitter. Producing more than an estimated 500 million tweets per day the platform is perfect for collecting information and especially opinions on thousands of topics. Multiple researchers have already done this by working on papers in different scientific areas, notably on prediction since such studies normally require large amounts of data.

Predicting financial movement is no exception here. Moreover, papers on how to predict stock prices utilizing Twitter have become common in the last few years, for example, Mittal and Goel 2012 and Oliveira et al. 2017. The premise is almost always to use the observed mood of a large group of people (sentiment) on financial products and derive certain speculations and possible

outcomes of stock market prices. However, these bigger groups are mostly also influenced by smaller parties, or even individual persons often influence these bigger groups. That, in turn, is a topic with little research, especially regarding how and why relations occur and not solely on what predictive power they have.

Our financial product of choice is Dogecoin. Released in 2013 this crypto coin was meant to be a not-so-serious currency by having the "Doge", a Shiba Inu dog, as its mascot. Nevertheless, it became an internet meme and reached a lot of social media engagement.

The individual and its influence that we want to study in this paper is tech billionaire and business magnate Elon Musk. As among the richest persons in the world, Mr. Musk is an influential personality. Regarding social media presence, he gathered over 100 million followers on Twitter, where he tweets nearly daily, reaching thousands of people simultaneously. Since he also posts a lot about financial topics, especially modern cryptocurrencies, Elon Musk, and his followership prove to be an excellent data basis for sentiment analysis. Over the last years, one of his favorite crypto coins became Dogecoin.

Our paper aims to study the relations between social media influencer Elon Musk and the cryptocurrency "Dogecoin." Therefore, we are addressing the following three research questions:

1. How do the tweets of Elon Musk about Dogecoin affect the tweet rate of every other Twitter user on the same topic?
2. Is there a correlation between the sentiment of Elon Musk's Dogecoin tweets and the stock market closing price of Dogecoin?
3. What is the behavior of Dogecoin's closing price concerning the sentiment of all Twitter users?

These questions will be answered using statistical measures and causation analysis on bulk data sets collected from Twitter. We chose the already mentioned social media platform because it is a reliable source for easy accessing blog-like posts. It also encourages users to research their data by providing a beginner-friendly API setup.

The paper is structured into three primary components. First, we present recent work and contributions on similar topics and background knowledge for a deeper understanding of our study. In the main part, we illustrate our approach for answering the research questions starting from data accumulation over classification till performing the sentiment analysis. Finally, the last section consists of the results presentation and a discussion of the findings. We also provide an outlook on what can be improved for future studies.

THEORETICAL BACKGROUND / RELATED LITERATURE

In this part, we will discuss our background knowledge of this analysis. First, people analyzed the stock market in many directions, establishing relationships between the stock market and macroeconomic variables. For example, Vlastakis and Markellos 2012 showed a positive correlation between volatility and trading volume. Then in behavioral finance, we will see how social media headlines impact the stock market. Later we justify our chosen technologies for this analysis.

Literature History

People analyzed the demand and supply of information in the early days. For example, Vlastakis and Markellos 2012 showed that the volatility and the number of searches is positively correlated. That means the need for information has significance not only on a single stock but also on the overall stock market. Takeda and Wakao 2014 also showed that stock return positively correlates

with search intensity. Moreover, Mao et al. 2011 introduced a new term called the Tweet Volumes of Financial Search Term (TV-FST). By calculating the TV-FST, the authors found a correlation between the TV-FST and the stock market. More precisely, Twitter is one of the best stock market predictors of all other social sites available in the market.

Alongside the supply of information demand, headlines start seeking people's attention. In the analysis of Howlett et al. 2007, they assemble a connection between news headlines and stock price returns. Most of the time, the news headlines on the announcement day influence the stock price return. Birz and Lott Jr 2011 did a similar analysis but differently. The author focuses on four headline subjects: GDP, unemployment rate, retail sales, and durable goods. Later, classify them as positive, negative, neutral, and mixed, respectively. As a result, only GDP and unemployment news impact the stock market return. Daniel et al. 2002 have shown that the stock return has a higher negative drift when bad news comes out rather than good news. With the change in technology and the help of advanced evaluation methods, researchers changed their research areas into demand/sales of a product, brand value, stock market performance, marketing, Etc. For example, Fehle et al. 2005 took samples from 1969 to 2001 and found that whoever put their adverts into the Super Bowl increased their stock price (The final game of the National Football League is called the Super Bowl). Kim and Meschke 2011 found a similar pattern for CEO interviews on CNBC (CNBC is a TV channel).

Afterward, an experiment by Oliveira et al. 2017 showed that in terms of mood states analysis, Twitter is an excellent platform for predicting people's mood states. The writer claimed it by surveying many people and asking them about their moods on some specific Twitter posts. Later, cross-matched them with the predictions and found the relevance of their claim. Similarly, several researchers (Sprenger et al. 2014; Ranco et al. 2015; Zhang et al. 2011; Smailovic et al. 2012; Moa et al. 2011) showed a correlation between sentiment emanated from Twitter and stock returns. Among them, Zhang et al. 2011 did sentiment analysis only with fear and hope, Moa et al. 2011 worked with bullish tweet sentiment posted by the investors, and other mood states (positive, negative, neutral, and mixed) were defined by Sprenger et al. 2014; Ranco et al. 2015; Smailovic et al. 2012. On the other hand, Bing et al. 2014 claimed that the type of industry plays a vital role in predicting stock prices from Twitter.

In the modern world, fast trading has become more popular. Especially crypto trading took all the attention from the stock markets. Behrendt and Schmidt 2018 analyzed Twitter from an intraday viewpoint. In conclusion, the authors found that Twitter sentiment does not impact intraday stock prices.

In our research, we mainly focused on three things, Elon Musk's tweets on Dogecoin, other people's tweets on Dogecoin, and the closing price of Dogecoin. However, researchers recently have worked on predicting cryptocurrency prices (Bitcoin, Dogecoin, and Ethereum are the top cryptocurrencies) from Twitter sentiment and time-series data of the coins. The below table shows the current work on predicting cryptocurrency prices.

Research Title	Methods	Crypto	Data Source	Discussion
"Predicting the Price of Bitcoin Using Machine Learning"- McNally et al. 2018	LSTM, RNN	Bitcoin	Bitnex, Coinbase, itBit, and Bitsamp	RNN showed better performance in predicting the price of Bitcoin. RMSE: 5.45

"Predicting Price of Cryptocurrency - A Deep Learning Approach" - Samiksha et al. 2020	RNN uses LSTM regression	Bitcoin	Kaggle	Considered attributes are Open, High, Low, Close, Volume, Currency, and Weighted Bitcoin price. RMSE: 3.38
"Bitcoin Price Prediction and Analysis Using Deep Learning Models" - Muniye et al. 2020	Gated Recurrent Unit (GRU)	Bitcoin	Kaggle	Considered attributes are Open, High, Low, Close, Volume, Currency, and Weighted Bitcoin price. RMSE: 0.069
"Prediction of Dogecoin price using deep learning and social media trends" - Agarwal et al. 2021	GRU, LSTM, RNN	Dogecoin	Kaggle, and Twitter API	LSTM using filtered historical market data (open, close, and volume) showed the best performance. RMSE: 0.017
"What Drives Cryptocurrency Prices? An Investigation of Google Trends and Telegram Sentiment" - Smuts et al. 2019	LSTM	Bitcoin, Ethereum	Google Trends, Market data aggregator, Telegram	Short-term trends and telegram used as a feature to detect the price movements of Cryptocurrencies. LSTM showed 63% and 56% accuracy on BTC and ETH, respectively.

Table 1. Comparison Table for Current Research

From the above discussion, we can say that there is no concrete work on our research attributes. Nevertheless, two research studies were done by Dablander 2021 and Ante 2021. Dablander 2021 is an unpublished work, and Ante 2021 published it on Blockchain Research Lab (BRL). Dablander 2021 took a few tweets from Elon Musk on Dogecoin and performed a causal analysis of the price of Dogecoin. The author concluded that each time after the Elon Musk tweets, there were 25-45% of price increases for Dogecoin. The drawbacks are that the author worked with very few examples and did not show any definite correlation between them. On the other side, Ante 2021 worked with Dogecoin price and Bitcoin price and tried to find movements after Elon Musk's tweets. Here the author took six events where Elon Musk posted something on Dogecoin or Bitcoin. Then the Cumulative Abnormal Return (CAR) and Cumulative Abnormal log trading Volume (CAV) were calculated. Later performing t-statistics over CAR and CAV, the author found significance for 1%, 5%, and 10% of CAR and CAV values. This research work also has

some limitations. They worked with very few examples and selected events randomly, so there is a high chance that these events' selection is biased.

Twitter API/Stock API

IvyPanda. 2022 analyzed different types of API for data collection. Researchers always need such an API to collect datasets. Irvin et al. 2020 did concrete research between a paid web scraper (Hoaxy) and Twitter API. The authors created three test criteria: text, user, and social credibility, then checked the credibility of the collected data. In most test cases, paid web scrapper showed similar data quality to the Twitter API's data, but web scrapper software was 50 times faster than Twitter API. However, researchers always overlooked it as the web scrapper is very expensive.
Collecting time-series data is a widespread practice now. Researchers usually use some free APIs from the market. Some of them are "Yahoo Finance," "Alpha Vintage API," "Crypto compare API," Etc. In the Technical University of Berlin's library blog, Golas 2022 showed how to fetch the financial data using Alpha Vintage API. However, we use Crypto compare API in our analysis.

METHODOLOGY

Methodology describes the strategy, rules, and procedures of analysis. In this section, we will discuss the whole process of our analysis. Figure 1 shows our workflow for this analysis. We split our workflow into different subsections. Each section describes our process, technology, and the reason behind using each method. At the end of this section, people will get an in-depth idea of our process.

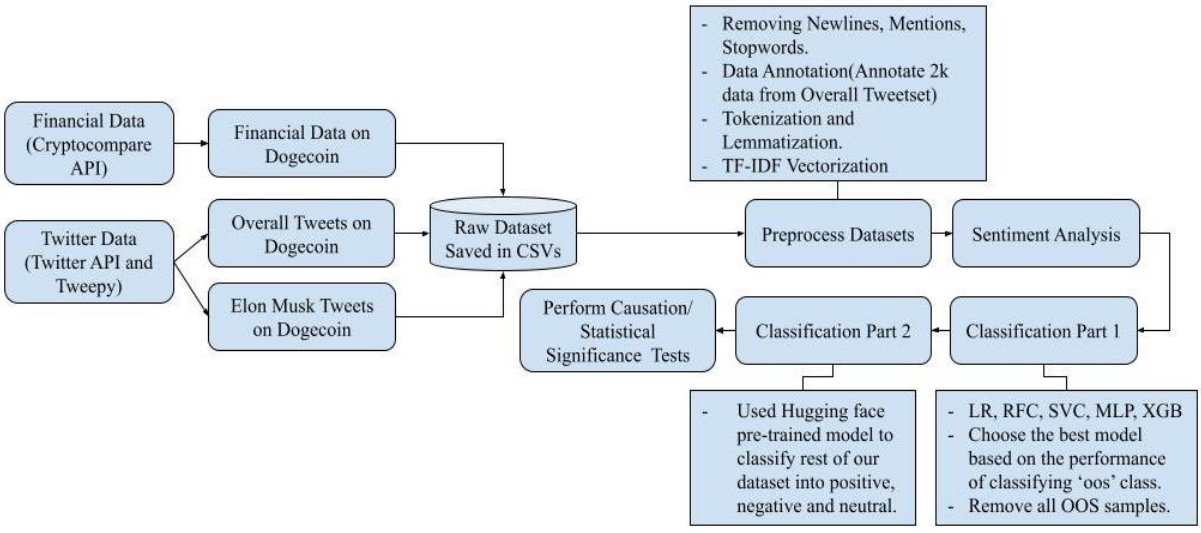


Figure 1. Proposed Architecture

Sentiment Analysis

Twitter Data Extraction

For getting the data from the Twitter API, we used Tweepy. Tweepy is an open-source package that makes extracting Twitter data easier by providing a better interface for accessing Twitter API.

Firstly, we needed 4 Twitter authentication credentials: consumer key, consumer secret, access token, and access secret used by the `OAuthHandler` class to authenticate the user. Then we created an "API" object used later to invoke Twitter API methods. To extract the tweets, we used a Pagination feature required when we wanted to get more than 100 tweets. The code for that,

```
tweets = tw.Paginator(client.search_all_tweets, query='#dogecoin -is:retweet lang:en', tweet_fields=['context_annotations', 'created_at', "text", "author_id", "source", "entities"], start_time=start_time, end_time=end_time, max_results=100).flatten(limit=6000)
```

The arguments in the Paginator ask to extract the tweets in English that were retweeted and contain the mention of "#dogecoin". We selected 6 features - context_annotations, created_at, text, author_id, source, entities. The timeframe is one day before Elon Musk tweeted about Dogecoin and one day after it. We retrieve 100 tweets per page, in total maximum of 6000 tweets. This data is converted into a data frame object and saved in a CSV file.

For storage, we created a small logging system. Given the preferred location and directory name, it creates a new directory (if it does not already exist). Then, each batch of the extracted data creates a new CSV file and writes the data there. The logger contains all the logging information for a single run.

Anyone can run our Twitter scrapper by executing the below command from the root directory of our repository (Twitter_sentiment_and_dogecoin_price).

```
python -m data_analysis.scraper --scraper tweepy
```

Table 2 shows the list of arguments given to the function.

Argument name	Meaning
`--dataset_path`	the location of the dataset
`--scraper`	the scraper name e, g., twint, tweepy
`--scraper_limit`	defines scraping limit. e, g., 10000, 20000
`--log_folder`	the location of the output folder
`--experiment_name`	the unique name for a single run
`--asset_path`	the location of the Twitter API configuration
`--logging_steps`	the number of cycles to run before the output is logged
`--subsample`	to subsample the dataset. e, g., true/false
`--device`	to set the GPU numbers
`--local_rank`	local rank should always be -1 (used for distributed training)

Table 2. Command-line Arguments

Financial Data Extraction

We extracted Dogecoin financial data using Crypto compare API. It collects historical price data of the most popular cryptocurrencies. There are multiple extraction periods to choose from – extraction minutely, hourly, and daily. During the chosen period of the time, it returns the highest, the lowest, the opening and the closing prices, and the volume. For this project, we extracted the hourly Dogecoin price from 2018 to July of 2022 and looked mainly at the closing price of the cryptocurrency. The data shows that more significant price deviations of Dogecoin started at the beginning of 2021, with the peak in June 2021. After reaching the peak, it shows a fluctuating downward trend.

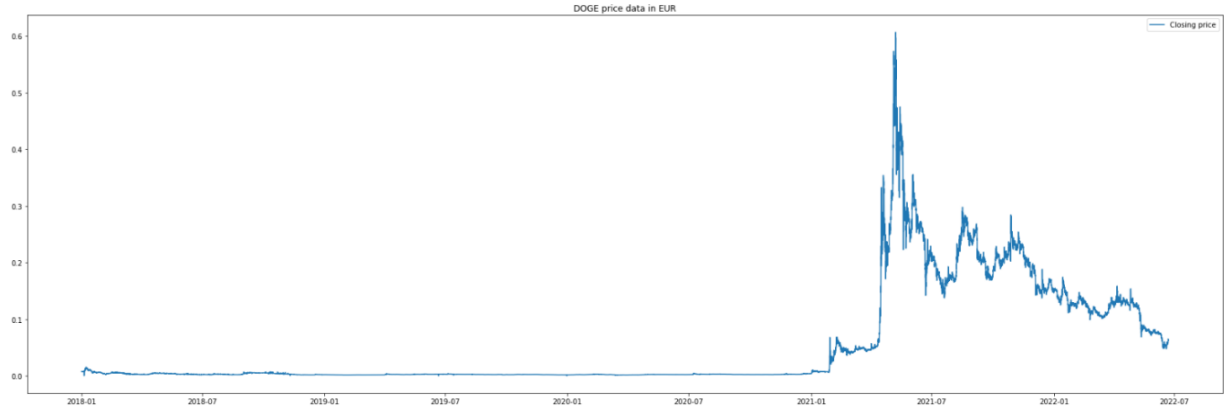


Figure 2. Dogecoin Price Comparison

Data Preprocessing

The Twitter dataset (mainly the unstructured textual data - tweets) needed much data pre-processing. Some of the tweets had to be removed, while the rest were modified to contain only the most essential information and then vectorized.

Removing OOS Tweets Based on Keywords, the first pre-processing step was filtering out-of-scope (OOS) tweets. OOS tweets are mostly cryptocurrency price updates, promotions, giveaways, or news articles without extra context. Many times, these tweets are written by bots. Some tweets contain the same text, while others have variations. We removed a big part of these by deleting the tweets containing duplicate text. Then we counted the number of tweets from each user id and deleted the ones with the most significant count. Most of the time, users with the highest number of tweets are bots, but to avoid deleting very active real users, we manually checked some tweets of each user before removing them. Account name data would have also helped filter out OOS. Unfortunately, Twitter API did not provide it.

Bots and spamming users often change accounts, so detecting them only with the user id is impossible. To counteract that, we also filtered out OOS tweets based on keywords. For example, tweets about cryptocurrency prices often contained phrases: price “price”, “pump alert”, “market cap”. Promotional tweets tend to ask people to check out, try, buy, join, download, register, follow or click on something. The keywords were adjusted to include promotions but excluded genuine tweets. For example, instead of taking the word “join” as a keyword, we took “join here”, “join my”, and “join telegram”. Promotional messages also tend to contain links to external websites and the word ‘link’ itself. Some also discuss complimentary items, rewards, referral links, giveaways, and airdrops.

This approach to detecting OOS tweets allowed us to remove the obvious promotion messages. However, the undetected ones had too many variations and similarities to normal tweets, so it was impossible to filter them out using keywords. To detect the rest of the OOS tweets, we employed Machine Learning algorithms (detailed description in Classification part 1).

Data Annotation, after filtering out some OOS tweets, we manually divided 2000 tweets into four classes: positive, neutral, negative, and OOS. Each category has a similar number of annotated tweets, except for the negative category, as negative sentiment occurs less frequently in the dataset.

Class	Label	Tweet example
Positive	1	"We will not be stopped. Dogecoin WILL reach 1 dollar. #DOGE #dogecoin #dogecointoadollar #DogecoinToTheMoon #dogecoinarmy #kpop"
Neutral	0	"@davidgokhshtein Even though there's no limit to how much #Dogecoin there can be, is anyone still mining it?"
Negative	-1	"Really starting to lose faith in #dogecoin ... general direction is down"
OOS	2	"The Best Game Ever!!! Just won 0.0274 $BTC #slots Come play! $DOGE #dogecoin #doge https://t.co/Hu1GAuVPz0"

Table 3. Tweet Example from Each Class

Some issues made annotation more difficult. Firstly, additional context is often needed to understand the message fully. That context, for example, can be another tweet (the original tweet that the message is replying to), a news article, or a Reddit post. It can also be some action happening worldwide (Elon Musk buying many bitcoins, a financial institution announcing stronger cryptocurrency market regulations). Secondly, tweets contain a lot of specific financial and cryptocurrency terms and expressions from the internet slang language. What is more, some tweets are sarcastic and long, making it even more challenging to understand the true intent behind the message. Often these are the tweets containing a negative sentiment.
Sentiment also depends on how the reader interprets the message. It means that different people could assign different sentiments to the same tweet, especially if the polarity of the message is not very strong. To minimize errors and get the most accurate annotations, three people annotated Elon Musk's tweets using the most common sentiment.

Convert Emojis to Text, Emojis and emoticons are a problem for the modeling process, so we must handle them. They can either be removed or converted to text for later usage. In sentiment analysis, it is usually helpful to convert them to words as they can provide useful insights into the sentiment of the text. There were almost no emoticons in the extracted tweets, so the few existing ones were deleted by removing punctuation in earlier preprocessing steps. For converting emojis from icon to text (emoji name), we used the emot library. Given a string or a list of strings, this library returns a dictionary with emojis (or emoticons) and their meaning. Based on the emo_unicode.py file, which contains a big dictionary of emoji Unicode characters and their

representation in text. In this project, the dataset without emoji information gave us better results, so we removed all of them.

Simplifying the Tweet Information, the raw Twitter data is unstructured and chaotic. It often contains spelling mistakes, informal language, new words, inconsistent formatting, and unnecessary information. To make it simpler, we firstly removed some elements that were redundant in the tweets. That were newlines, links to websites, and mentions of the other Twitter accounts. All the tweets were normalized (converted to lowercase letters). Using the regular expression, we removed the numbers and punctuation.
We often used NLTK (Natural Language Toolkit) for the multiple preprocessing steps. It is a suite of open-source programs and libraries for textual data processing. It can help in classifying, stemming, part-of-speech tagging, tokenizing, or parsing textual data. We used NLTK for tokenization – splitting each tweet by word into smaller units – tokens. We tokenize the words with the word_tokenize function from the tokenizing package. Using another NLTK package – corpus, we downloaded the list of pre-defined stopwords in the English language (such as "the", "of", "between"). Stopwords are the most common words that often do not carry any information, so we removed them from our data. The next preprocessing step was lemmatization. The lemmatization process looks at the morphological analysis to reduce the number of tokens in the dataset. It is done by grouping together the inflected words and returning the lemma (the proper base, dictionary form) for each word. For this, we used the `WordNetLemmatizer()` method from `nltk.stem` package. It looks up the lemmas in WordNet, a large English lexical database of words semantically grouped into synsets. So, all the tokens were reduced to their lemma form. Finally, we joined the tokens that were left after the preprocessing back to the strings (tweets).

TF-IDF Vectorization, the final step of our preprocessing process was text vectorization - transforming the data from text to numbers. For that, we used the TF-IDF statistic measure. It is calculated using the product of term frequency (1) and inverse document frequency (2). In TF formula (1), *t* stands for a term and *d* for a document. In IDF formula (2), *N* stands for the total number of documents, *df* for the total number of documents containing the term.

1. $TF(t,d) = \frac{number\ of\ times\ t\ appears\ in\ d}{total\ number\ of\ terms\ in\ d}$
2. $IDF(t) = \log\frac{N}{df} + 1$
3. $TFIDF(t,d) = TF(t,d) * IDF(t)$

We did vectorization using `TfidfVectorizer()` from `sklearn.feature_extraction` module. The given parameters were `stop_words = 'english'` (removed English language stop words) and `max_features = 3500` (only 3500, the most frequent features across the corpus, were considered). Then the `fit_transform()` method learned the vocabulary, IDF, and returned the document-term matrix.

Classification (Part 1)

While preprocessing our datasets, we found a huge portion containing promotional data. We classify them as OOS. We also tried to detect them using keywords, but we found it hard to detect all these tweets. So, we decided to classify them by building machine learning models. For this purpose, we went through all the preprocessing steps for our datasets.
Classification is a supervised learning approach in machine learning. In supervised machine learning, machines need true labels for each sample of inputs. So that machines can train

themselves with these true labels for future predictions. Kotsiantis et al. 2006 also said, "The goal of supervised learning is to build a concise model of the distribution of class labels in terms of predictor features. The resulting classifier is then used to assign class labels to the testing instances where the values of the predictor features are known, but the value of the class label is unknown.". For the true labels, we annotated our dataset. In the data annotation part, we described the whole process of data labeling.

Many machine learning classifiers are available, but there is no specific model for a specific problem. However, finding the best model for a specific problem is a process- Kotsiantis et al. 2006 (see Appendix 1 for visual explanation). Furthermore, in Classification (Part 1), we wanted to build a machine learning model to classify OOS tweets from our datasets. For our problem, we decided to proceed with five machine learning classifiers. These are Random Forest Classifier (RFC), Multi-Layer Perceptron Classifier (MLP), Support Vector Classifier (SVC), Logistic Regression (LR), and Extreme Gradient Boosting (XGBoost).

Random Forest Classifier (RFC)

In supervised machine learning, Random Forest is a well-known algorithm. We can use it for both classification and regression. Random Forest Classifier usually creates decision trees for each sample and takes the majority vote for the final class. For regression, it calculates the mean value for the final output. It is very convenient to use because it trains very fast. It predicts very well with a large dataset. Even though it can handle missing data well. **Figure 3** shows how Random Forest Classifier works. Python (Programming Language) has a built-in sklearn package that can call a method to perform this algorithm for us. We can use it by calling `sklearn.ensemble.RandomForestClassifier()` in Python. Important hyperparameters are `n_estimators` (denotes the number of trees an algorithm will build) and `max_depth` (denotes the maximum depth of the tree). The Hyperparameter Tuning section describes all the numbers and parameters we used to tune our model.

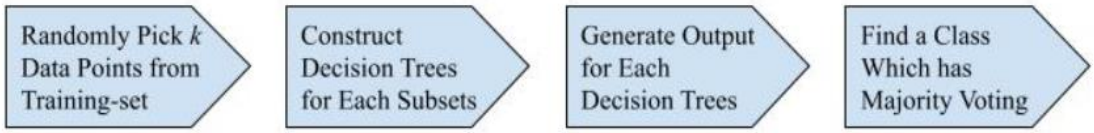


Figure 3. Steps of Random Forest Classifier

Logistic Regression (LR)

If we want to calculate the probability of all class labels, we will need Logistic Regression. Logistic Regression gives us a list of probabilities between 0 to 1 for each class. It is a sigmoid of the linear layer. Let us assume z is the output of a linear layer. If we put it into a *sigmoid(z)*, we can get a probability between 0 and 1. Formula (4) shows the mathematical term of a sigmoid function.

4. $y' = \frac{1}{1+e^{-z}}$

$Here,$
$y' = the\ output\ of\ Logistic\ Regression$
$z = linear\ layer\ (z = b\ + wb)$
$w\ is\ the\ learned\ weight, b\ is\ the\ bias, and\ x\ is\ the\ feature\ vector$
(See Appendix 2 for more information about the sigmoid function).
There are two types of Logistic Regression available. Binary Logistic Regression is for classifying two class labels (e.g., 0/1, True/False, Spam/Not Spam, Etc.). Multi-class Logistic Regression is

for classifying more than two class labels (e.g., -1/0/1/2 and so on). We will use multi-class logistic regression as we have four class labels (positive, negative, neutral, and oos). By calling `sklearn.linear_model.LogisticRegression()` in Python, we can easily train this model.

Multi-layer Perceptron (MLP)

A common neural network algorithm is Multi-layer Perceptron. The input layer, Hidden layer, and Output layer are the three main layers of this network. Each layer relates to every other layer. We can compare it with human brains. The way neurons share information with other neurons, each layer of the Multi-layer Perceptron network also shares information with other layers. An ideal network has three layers. The first one is the Input Layer, the leftmost layer of the network that takes features as input. The second one is the Hidden Layer. Hidden Layers node always be one more than the Input Layer. Moreover, an extra node of the Hidden Layer is called a Bias node. The Output Layer contains the network's output (See Appendix 3 for visual information). Furthermore, an activation function is required while data transform from one network to another. The Multi-layer Perceptron uses three types of activation functions on the base of the problem statement. Appendix 4 shows the curve of all three activation functions. The key hyperparameters to tune are the `hidden_layer_sizes`, `activation`, and `learning_rate`. Python also has a ready method for this algorithm. To train this model in Python, we need to call `sklearn.neural_network.MLPClassifier()`.

Support Vector Classifier (SVC)

Support Vector Classifier is a classification method from Support Vector Machines(SVM). We can do both classification and regression using SVM. SVM works well when the input feature is more than two. Because SVM works with hyperplanes. For two input features, SVM creates some linear hyperplanes and try to fit them into the data points. It only keeps that single hyperplane which can linearly separate most data points. For three input features, it creates 2D hyperplanes and tries to separate all the data points. SVM has a cool feature which is called the SVMKernel/ Kernel trick. It can transfer a low-dimensional input space into a high-dimensional space (see Appendix 5 for examples of linear and non-linear data points). That means SVM works with both linear and non-linear classification. Moreover, a Support Vector Classifier is used for classification problems. By using Python, we can easily call SVC e, g., `sklearn.svm.SVC()`. "`C`," "`kernel`," and "`decision_function_shape`" are the key hyperparameters for SVC.

Extreme Gradient Boosting (XGBoost)

Gradient Boosting is an ensemble learning algorithm. It creates a decision tree and tries to optimize the prediction losses. It is also called Gradient Tree Boosting or Stochastic Gradient Boosting. The idea of Extreme Gradient Boosting also comes from Gradient Boosting. Chan and Guestrin 2016 first implemented it in 2016. They made Gradient Boosting more efficient by caching access patterns, data compression, and sharding for building a decision tree. In Gradient Boosting, it selects the best node by calculating the Mean Square Error (MSE), but in XGBoost, it selects the best tree node by calculating similarity scores and gain value.

5. $similarity = \frac{(\sum_{i=1}^{n} Residual_i)^2}{\sum_{i=1}^{n}[Previous\ Probability_i * (1 - Previous\ Probability_i)] + \gamma}$

Here,

$$Residual = actual\ value - predicted\ value$$

$$Previous\ Probability = probability\ calculated\ in\ the\ previous\ step. Initially\ it\ is\ 0.05$$

$$\gamma = regularizer\ parameter$$

6. $gain = Left\ Leaf_{similarity} + Right\ Leaf_{similarity} - Root_{similarity}$

As Chan and Guestrin 2016 mentioned, XGBoost is a state-of-the-art model, especially in classification. We also found the same in our classification analysis. The result section shows our results for all the algorithms mentioned above. We can use it easily in Python. However, before that, we must install Python's `xgboost` package. Later from this package, we can call the classifier by using `xgboost.XGBClassifier()`.

Hyperparameter Tuning and Cross-validation

Every model has its hyperparameters. We can increase our model's performance by putting different combinations of values into these hyperparameters. This process of finding the best hyperparameter values is called the Hyperparameter tuning of a model. On the other hand, we also wanted to calculate the performance of a model on unseen data. Calculating the performance of a model on unseen data statistics shows us a way called Cross-validation, and the k-fold Cross-validation is a pervasive way to measure the performance of a model on unseen data. **Figure 4** shows the process of k-fold Cross-validation.

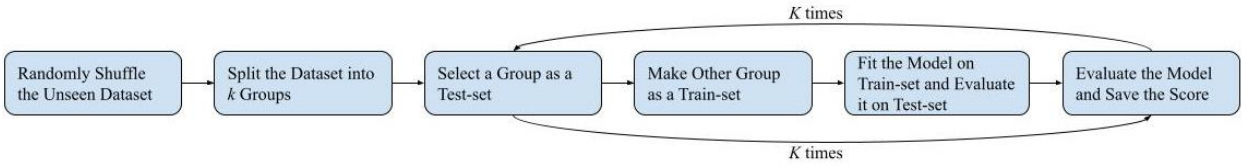


Figure 4. *k*-fold Cross-validation Process

Using *k*-fold Cross-validation, we can tune our hyperparameters. GridSearchCV made this process straightforward. It is a method to create a grid with the set value of hyperparameters. It then calculates the average estimated value, which comes from the Cross-validation process for each combination of hyperparameters. We use 5-fold Cross-validation in our classification process for all the machine learning algorithms. Appendix 6 shows which hyperparameter we tuned, and Appendix 7 shows the best-estimated model for each algorithm.

Classification (Part 2)

After classifying the OOS class using XGBoost from our dataset we stepped forward to the second part of our analysis. The models we built could not perform well with our dataset's other classes (positive, negative, and neutral). So, we decided to classify these remaining classes using other available technologies. We tried TextBlob, but it does not make sense for our analysis. Because TextBlob uses a pre-trained Naive Bayes algorithm behind it, we tried such models in our classification (part 1). Moreover, our dataset is from the Twitter domain, specifically the finance domain. Now, we need a pre-trained model which is trained on more Twitter data. After some research, we found that the Hugging Face has some pre-trained models, which are pre-trained with a huge corpus of Twitter data. We took a subsample from our annotated dataset and tests several pre-trained models from Hugging Face,

- `transformers.pipeline(model="sentiment-analysis")`

- `transformers.pipeline(model="finite automata/bertweet-base-sentiment-analysis")`
- `transformers.pipeline(model="cardiffnlp/Twitter-roberta-base-sentiment")`
- `transformers.pipeline(model="cardiffnlp/Twitter-xlm-roberta-base-sentiment")`

Among them "cardiffnlp/Twitter-roberta-base-sentiment" (Barbieri et al. 2020) performed well on our dataset. This model uses the BERT (Devlin et al. 2019) model. For comparison purposes to see which pre-processed dataset performs well. We ran the same analysis three times on three states of datasets (raw, partially pre-processed, and fully pre-processed data). The sentiment analysis result section shows the results of this model. Later we classify our whole datasets into positive, negative, and neutral.

Evaluation Measures

In machine learning, evaluation is a key component in model selection. The metric to evaluate a machine learning algorithm is called Evaluation Measures. It allows us to compare the performance of different machine learning algorithms and optimize an algorithm. Evaluation can be done in several ways. Generally, people rely on the accuracy score, f1-score, AUC curve, error rate, etc. as an evaluation metric of models. Some analyses required only visual representation, so their evaluation metric is finding any pattern in the visuals. Moreover, some analyses also depend on the correlation scores of two variables. In such a case, the correlation score is the evaluation metric.

A general approach would be to see the Confusion matrix. The confusion matrix gives us a good idea about the quality of a trained model. Table 4 shows a sample confusion matrix for two class labels.

	Negative	Positive
Negative	True Negative (TN)	False Positive (FP)
Positive	False Negative (FN)	True Positive (TP)

Table 4. Confusion Matrix

A confusion matrix provides four measurement scales. True Positive denotes the number of positive instances classified by a model. False Positive shows the number of predicted positive instances which are originally negative. False Negative also shows us the number of predicted negative instances which are originally positive. True Negative is the number of predicted negative instances which are originally negative instances. The number of class labels denotes the row and column number for a confusion matrix. If a confusion matrix has more than two class labels, the correctly classified class number is always placed diagonally. We can also denote a threshold for a model to predict future data. However, this is not our concern for this analysis. From the confusion matrix, we can calculate the f1-score for each class label and the overall model. F1-score combines precision and recall. as the harmonic mean. Precision is the fraction of correctly predicted positive instances to the overall positive instances, and recall is the ratio of correctly predicted positive observations to the total true positive observation. High precision relates to the low false positive rate, and high recall relates to the low false negative rate. All the numbers are always between 0 to 1.

7. $Precision = \frac{TP}{TP+FP}$
8. $Recall = \frac{TP}{TP+FN}$
9. $f1\ score = 2 \times \frac{Precision \times Recall}{Precision+Recall}$

We consider precision, recall, and f1-score as evaluation metrics of the models in our analysis. Of course, we took weighted scores for all the metrics as we have some imbalance class labels for the negative class. In the Classification (part 1), we optimize our model based on the f1-score of a Model. For part 2, we also used weighted precision, recall, and f1-score as our evaluation metrics to choose the best pre-trained model for our analysis.
To validate the relations of our research questions, we used the Pearson Correlation Coefficient test. The probable values are between -1 to +1. Negative correlation scores mean both variables go in opposite directions. Positive correlation scores mean both variables change in the same direction. Exact zero or near zero means both variables do not correlate with each other. The Result section also contains the result for all these correlation coefficient tests.

RESULTS

Sentiment Analysis Results

Subsequently, the results (Table 5) of our model selection process for removing the "out-of-scope" tweets (sentiment value "2") and the confusion matrix for each candidate (Figure 5):

Class Label	Model	Precision	Recall	F1-Score
-1	Random Forest Classifier (RFC)	0.75	0.08	0.15
	Multi-layer Perceptron (MLP)	0.00	0.00	0.00
	Logistic Regression (LR)	0.75	0.08	0.15
	Support Vector Classifier (SVC)	0.72	0.35	0.47
	Extreme Gradient Boosting (XGBoost)	0.52	0.35	0.42
0	RFC	0.55	0.27	0.36
	MLP	0.56	0.18	0.28
	LR	0.55	0.27	0.36
	SVC	0.50	0.40	0.45
	XGBoost	0.56	0.40	0.45
1	RFC	0.44	0.84	0.58
	MLP	0.39	0.95	0.55
	LR	0.44	0.84	0.58

	SVC	0.51	0.80	0.62
	XGBoost	0.73	0.78	0.75
2	RFC	0.73	0.73	0.72
	MLP	0.80	0.53	0.64
	LR	0.73	0.73	0.72
	SVC	0.76	0.62	0.68
	XGBoost	**0.73**	**0.78**	**0.75**

Table 5. Classification Results for "overall" Data (4 Classes)

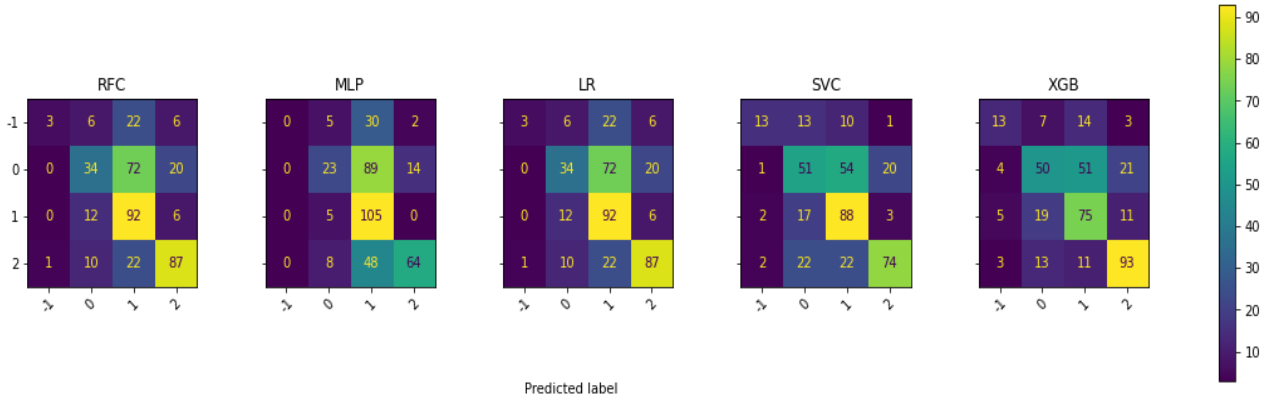


Figure 5. Confusion Matrices of The Model Selection for Removing OOS Data

Table 6 and figure 6 show the outcome of the pre-trained model (Twitter Roberta Base Sentiment) from the platform "Huggingface", which we needed to classify the remaining positive, negative, and neutral tweets. The pre-trained model was used on three variants of the remaining Twitter data set. First, we let it work with the raw version of the tweets, where no cleaning or pre-processing had been performed. The second iteration classified the cleaned tweets, meaning newlines, mentions, and links were removed. Lastly, we let the model work with the preprocessed tweets, so

in addition to cleaning them, stopwords were deleted, and tokenization and lemmatization were applied:

Class Label	Model	Precision	Recall	F1-Score
-1	Twitter-roberter-base-sentiment with raw_tweet	0.63	0.54	0.58
	Twitter-roberter-base-sentiment with cleaned_tweet	**0.62**	**0.55**	**0.58**
	Twitter-roberter-base-sentiment with preprocessed_tweet	0.67	0.39	0.49
0	raw_tweet	0.56	0.54	0.55
	cleaned_tweet	**0.57**	**0.56**	**0.57**
	preprocessed_tweet	0.45	0.77	0.57
1	raw_tweet	0.64	0.69	0.67
	cleaned_tweet	**0.65**	**0.69**	**0.67**
	preprocessed_tweet	0.62	0.32	0.42

Table 6. Final Classification Results for Variations of "overall" Data (3 Classes)

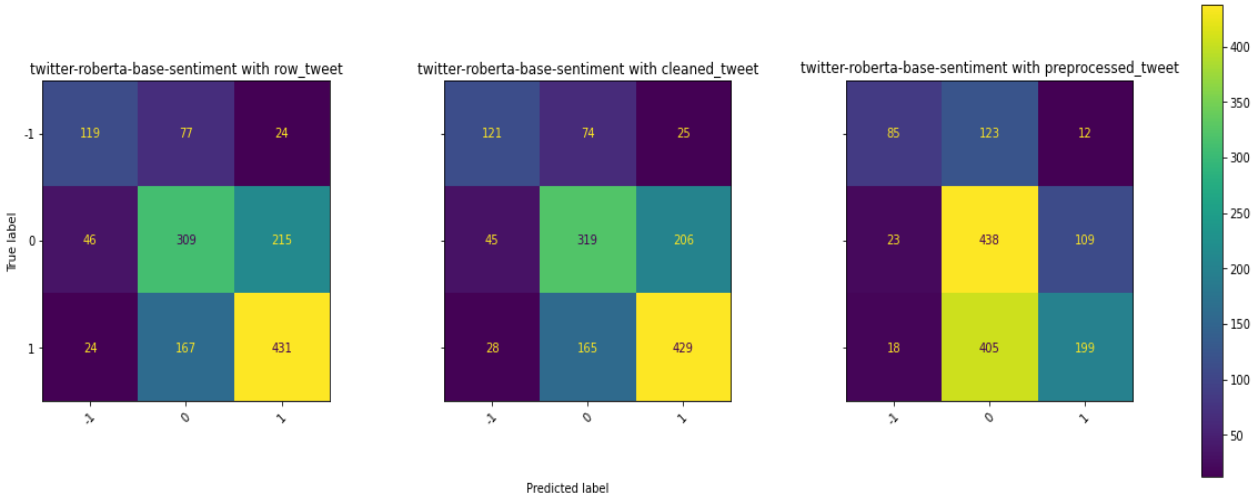


Figure 6. Confusion Matrices for Classification of Variations of "overall"

Dataset characteristics

The baseline of all the findings in this project consists of the three datasets that we scraped, formed, and analyzed.

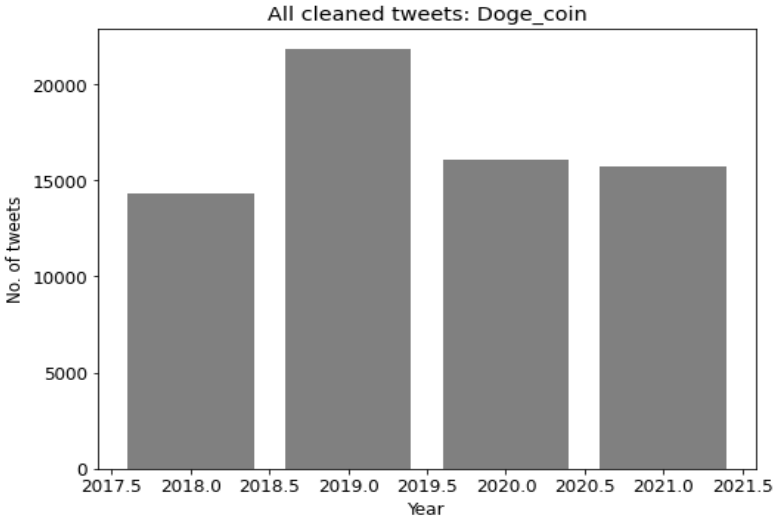


Figure 7. Number of Cleaned Dogecoin Tweets from Every Twitter User

Starting with the set that contains every tweet from the year 2021 that is about dogecoin. After the first basic cleanup of duplicates and corrupted data, we were left with 73,390 tweets. The final step for this data set to be ready for analysis was to remove all the "out of scope", or short "oos", tweets. Nearly 40% of the tweets were sorted out in this process so that in the end the overall tweet set on dogecoin counted 45,760 tweets in total to work with. If we would've used the cleaned data instead of the pre-processed one, we would be left with way less data per year (figure 7).

States	Number of tweets
Before removing OOS	73,390
After removing OOS	45,760

Table 7. Tweet Count Before and After OOS Removal

The second important dataset is the one with tweets from Elon Musk on the topic of dogecoin. When scraping and cleaning them finished, it quickly became clear that they underly a very

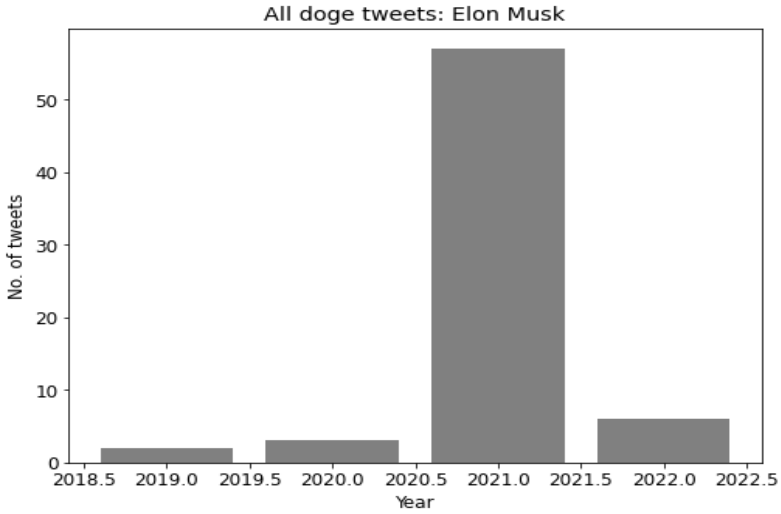


Figure 8. Number of Elon Musk's Tweets from 2018-2022

interesting and one-sided distribution. As visible in the following figure (Figure 8), Elon Musk talked about dogecoin on Twitter almost only in 2021. Every other year from 2018, when dogecoin became more popular, till now only contains a fraction of Musk's dogecoin tweets. With 83% the year 2021 accounts for most of them which in turn makes it obvious why we focused on this year regarding the project.

At last, we worked with a financial dataset holding all the information about the different OHLC (open, high, low, close) prices of the cryptocurrency represented in table 8. Since our code allowed

time	high	low	open	volumefrom	volumeto	close
2022-05-28	0.07749	0.07533	0.07574	18486554.82	1411239.05	0.07622
2022-05-29	0.07752	0.07448	0.07622	13691223.47	1043169.81	0.07703
2022-05-30	0.08210	0.07654	0.07703	38331471.94	3049260.57	0.08163
2022-05-31	0.08191	0.07767	0.08163	39689187.20	3181568.98	0.07995
2022-06-01	0.08334	0.07933	0.07995	14745785.39	1196144.56	0.07970

Table 8. OHLC Prices of Dogecoin per Day

us to choose the time interval of the financial data freely, we were able to represent prices month-, day-, hour- or even minute-wise. In the interval from 01.01.2019 to mid-2022 for example we have been able to access between 42 (monthly) to up to 1,837,440 (minutely) data points. Redrawn

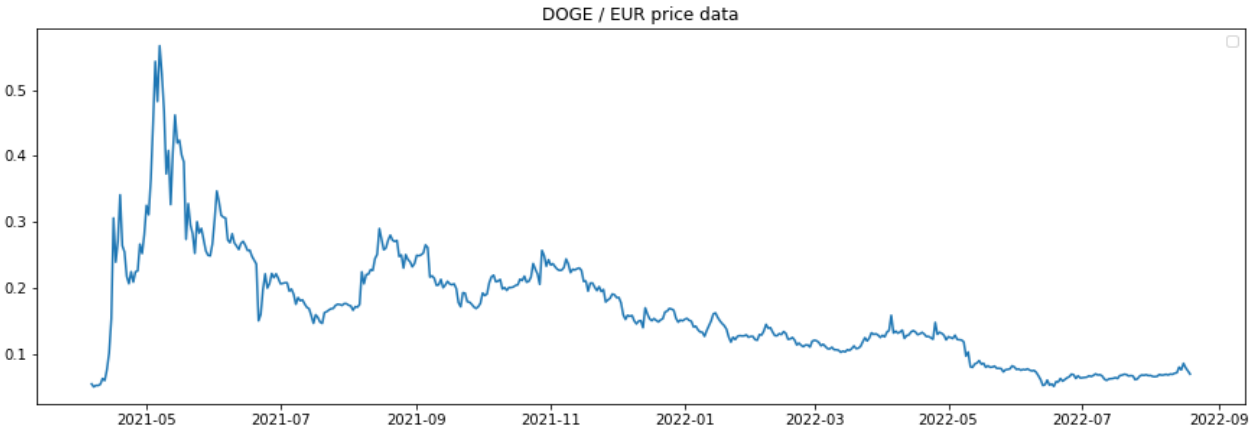


Figure 9. Daily Dogecoin Price (in €) for 16 Months

as a graph (Figure 9) these values portray the volatility of dogecoin on different scales and can be seen again in the correlation analysis of the project.

Correlation analysis

Figure 10 shows the development of the rate of Dogecoin tweets from Elon Musk and "overall" in 2021 per month. For better comparison, we scaled the values to the same magnitude, with 1.0 as

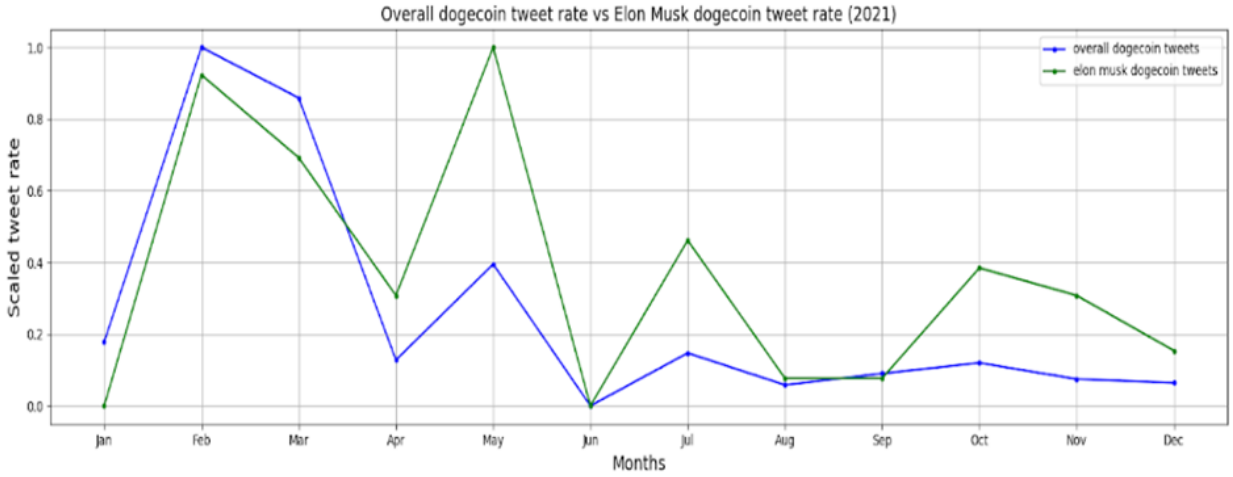


Figure 10. Tweet Rate: Overall vs Elon Musk (Monthly, 2021)

the maximum per group. As can be seen, the parties started low in January, but within February, "overall" hit their peak already, and Elon Musk's rate with his second highest value. This indicates their first connection, and this trend has continued over the year. Every time Elon Musk's tweet rate de- or increases, the same happens to the "overalls" rate, with remarkable examples in April, May, and June, even if it does not reach the same speeds. In the second half of 2021, however, the correlation faded. Even if Mr. Musk tweeted a lot about the cryptocurrency of choice, the "overall" tweet rate failed to increase. The impact Elon Musk had on the Dogecoin crowd became

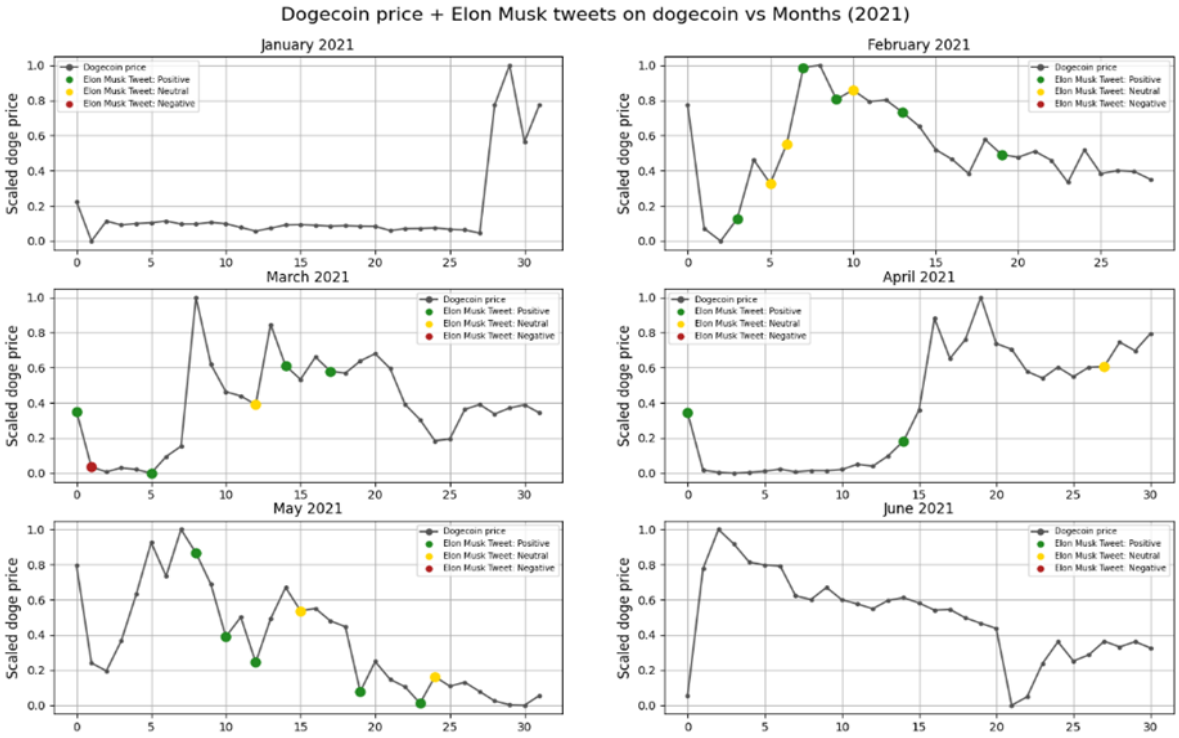


Figure 11. Dogecoin Price and Sentiment of Elon Musk's Tweets (Monthly, 1. half of 2021)

increasingly damped. To measure the effect of Elon Musk on the "overall" tweet rate, we used the "Pearson correlation." The results stated that the correlation between the number of overall tweets and the ones of Elon Musk is roughly **0.777** out of 1. It is what can be anticipated when looking at figure 10.

With the following two figures, 11 and 12, we focused on how the tweets of Elon Musk and their sentiment affected the closing price of Dogecoin. The premise was to check if the price moves

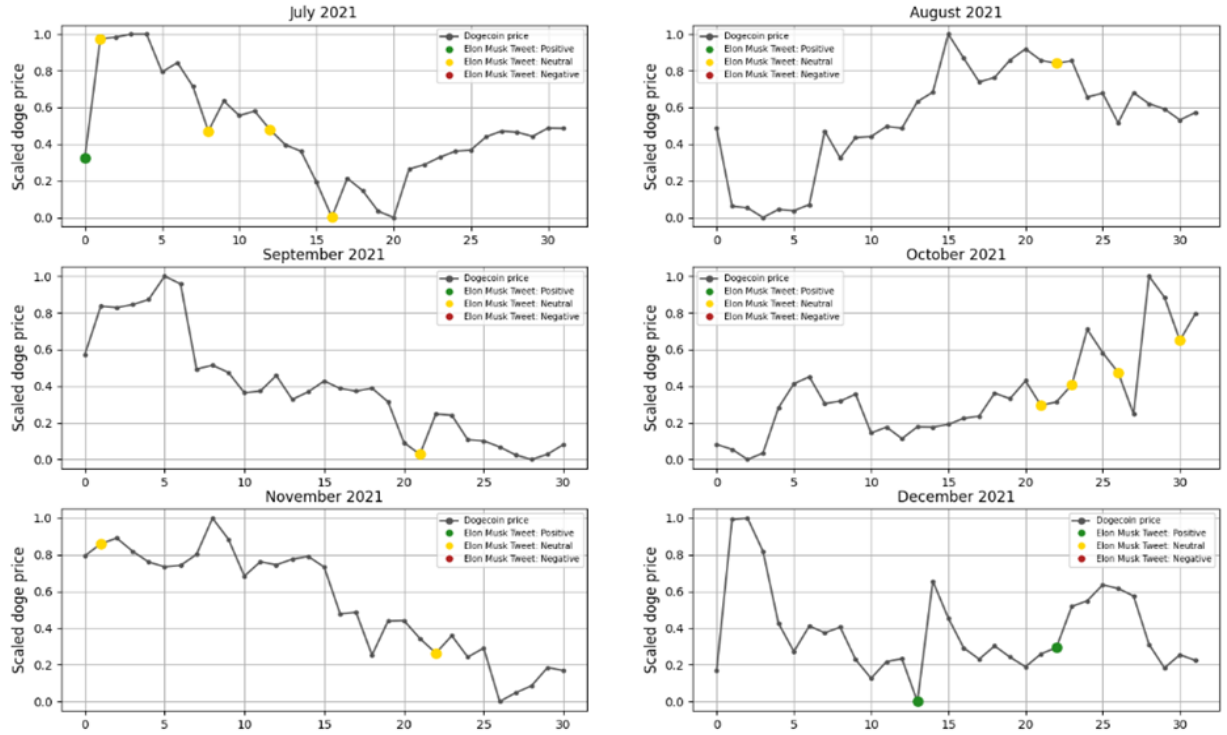


Figure 12. Dogecoin Price and Sentiment of Elon Musk's Tweets (Monthly, 2. half of 2021)

after a tweet that can be positive, neutral, or negative and watch whether it goes up, down, or stays the same. This test has been done every month of 2021 with daily financial data points. The colored dots represent the time when Elon Musk tweeted about Dogecoin, with the colors representing the sentiment from the latest tweet of the day. Since Mr. Musk did not post consistently, the representation per month varies. Especially after a positive tweet, the Dogecoin price tends to increase, in some cases, like after May 5th or April 14th, by a lot. What can also be observed is that the price sometimes rises even when a neutral classified tweet is posted. Reasons for this behavior might lie in misclassifying a usually positive tweet. The "overall" users might have positively interpreted the text, while the algorithm struggled with the context and sorted it in the wrong class. Similar assumptions can be made in the year's second half (figure 12), but with less confidence because we worked with fewer data points on Elon Musk's tweets here. As seen previously, the closing price of Dogecoin moves upwards after a neutral or positive tweet for at least a day, for example, on July 16th, September 21st, and especially on December 13th. Somehow there are also some apparent counterexamples for the theory that only tweets influence market prices. The rise and fall in the first two weeks of March cannot be based on Elon Musk. The same observations can be made at the same time in September, October, and December.

At last, we evaluated how the average Dogecoin price behaved per month compared to the "overall" sentiment in 2021. Again, we split the sentiment into its three classes represented and worked with the total number of tweets per class (figure 13). Both sentiment and Dogecoin price

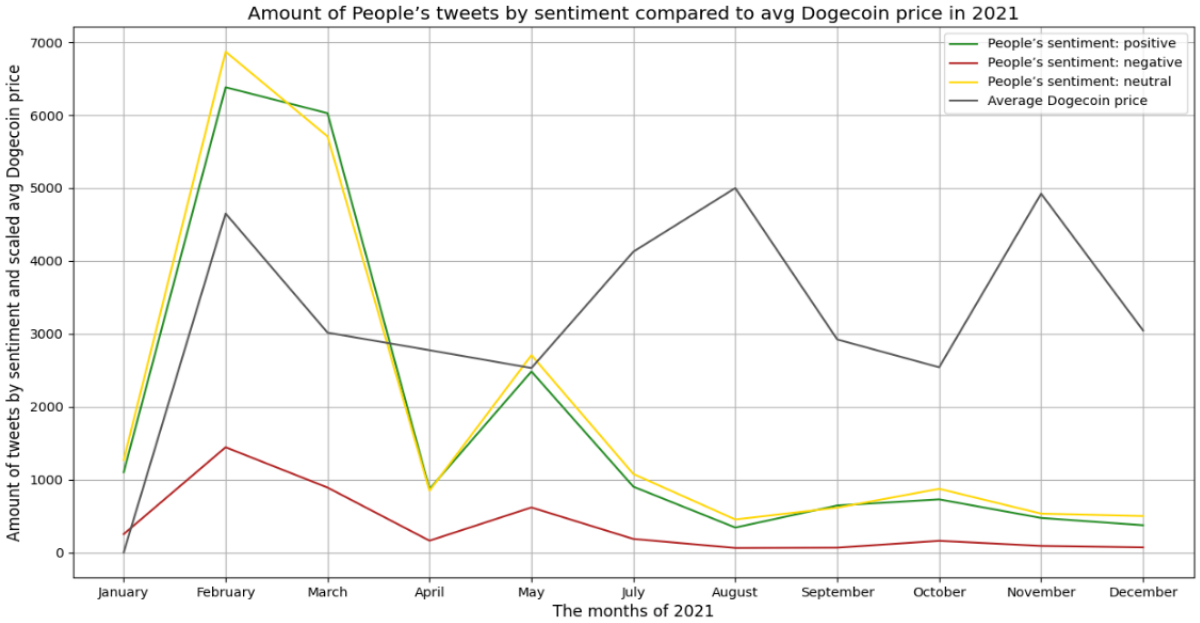


Figure 13. Overall Sentiment Vs. Monthly Dogecoin Price (2021)

were scaled to fit in the same magnitude. While it initially appeared that people influenced the cryptocurrency positively in February, it becomes clear as the year progresses that this is not the case. Even a heavy dip in positive and neutral sentiment in April did not affect much. To test if a connection is present between the "overall" sentiment and Dogecoin price, we calculated the Pearson correlation on daily bases. However, the data for the number of tweets per every month

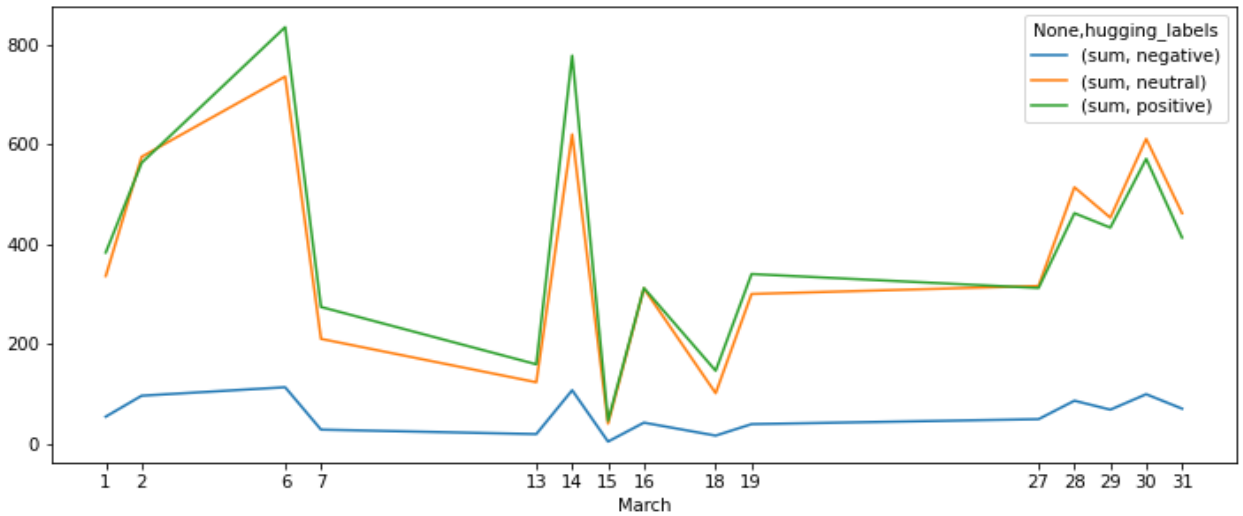


Figure 14. Dates in March 2021 with Available "overall" Tweet Data

was not distributed evenly over all its days (figure 14), primarily due to inconsistent activity on the topic and the deletion of many "oos" tweets. This left us with only those cases where we had

collected data for several consecutive days, like March 27th - 31st. Table 9 shows the calculations result for February, March, and May:

Time period	Pearson Correlation
February 25th - 28th	-0,800
March 27th - 31st	-0,300
May 24th - 26th	-0,500

Table 9. Pearson Correlation for Consecutive Data Points per Month

Surprisingly, the math shows a negative correlation for all three, with February being the strongest. We are not fully yet sure how much we can trust this result, but we are certain that the calculations are correct and explained by the lack of more consistent and consecutive data.

DISCUSSION

The topic of researching relations between big social media influencers and the development of cryptocurrencies is still in an early stage. While most other paper concentrate on using sentiment analysis to predict the future financial movement, we focused more on the "how" behind the scenes. An integration into other existing research as well checking for similarities or contradictions tends to be hard. Instead, we use common sense and reasoning to validate our findings. Therefore, we still rely on previous work nonetheless like Sprenger et al. 2014; Ranco et al. 2015; Zhang et al. 2011; Smailovic et al. 2012; Moa et al. 2011 because they showed at least some correlation between sentiment that emanated from Twitter and stock returns. But utilizing Elon Musk and Dogecoin was a novelty, and our work offers a new perspective. Yet we are aware that we can't answer all question to full disclosure like why the Pearson Correlation is negative for "overalls" sentiment and Dogecoin price even if the graph promises otherwise. Here we come across the limitations of what we can do now. Since we cannot compare to other papers we can only rely on our own calculations, which of course leaves room for mistakes, although we took the math very seriously. As for the future we would like to enhance our doing with the knowledge we obtained during this study. For example, there is improvement in hand annotating more tweets in preparation for the sentiment analysis. This would allow classification to become better and the following calculations to be even more precise

CONCLUSION

At the end of our study, we can say that we found out results which most definitely have not been captured yet. Elon Musk has an influence on how much people tweet about certain topics, but it also looks like that those people tend get saturated after some time. Nothing about this seems to only apply for Musk, since feels natural that one might lose interest on something after getting repetitively information from the same source. The engagement also decreases. Furthermore, we discovered punctual influence of Mr. Musk's tweet on the Dogecoin price but with only low endurance, which means after a short amount of time the share price "normalizes". This is also understandable because influencers like him aren't the only impact on financial movement. A whole lot of other factors play a role, too. Same goes for the rest of Twitter users on stock price development.

There are obviously relations between Elon Musk and Dogecoin prices, but their strength and implications may vary widely under different premises.

ACKNOWLEDGEMENTS
This template has been prepared based on AMCIS Template, which was in turn adapted from a workshop document created by Ping Zhang. For AMCIS template please follow this link:

REFERENCES

1. Agarwal, B. et al., 2021. Prediction of dogecoin price using deep learning and social media trends. *EAI Endorsed Transactions on Industrial Networks and Intelligent Systems,* 8(29)(e2).
2. Ante, L., 2021. *How Elon Musk's Twitter activity moves cryptocurrency markets,* Hamburg: BRL Working Paper Series No. 16.
3. Awoke, T., Rout, M., Mohanty, L. & Satapathy, S. C., 2021. *Bitcoin Price Prediction and Analysis Using Deep Learning Models.* Singapore: Springer Singapore.
4. Barbieri, F., Camacho-Collados, J., Neves, L. & Espinosa-Anke, L., 2020. *TweetEval: Unified Benchmark and Comparative Evaluation for Tweet Classification.* s.l., arXiv.
5. Behrendt, S. & Schmidt, A., 2018. The Twitter myth revisited: Intraday investor sentiment, Twitter activity and individual-level stock return volatility. *Journal of Banking & Finance,* Volume 96, pp. 355-367.
6. Birz, G. & Lott, J. R., 2011. The effect of macroeconomic news on stock returns: New evidence from newspaper coverage. *Journal of Banking & Finance,* 35(11), pp. 2791-2800.
7. Dablander, F., 2021. *Causal effect of Elon Musk tweets on Dogecoin price,* s.l.: s.n.
8. Daniel, K., Hirshleifer, D. & Subrahmanyam, A., 2002. Investor Psychology and Security Market Under- and Overreactions. *The Journal of Finance,* 53(6), pp. 1839-1885.
9. Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K., 2019. *{BERT}: Pre-training of Deep Bidirectional Transformers for Language Understanding.* Minneapolis, Minnesota, Association for Computational Linguistics.
10. Dongo, I. et al., 2020. *Web Scraping versus Twitter API: A Comparison for a Credibility Analysis.* New York, 22nd International Conference on Information Integration and Web-based Applications & Services (iiWAS '20).
11. Fehle, F., Tsyplakov, S. & Zdorovtsov, V., 2005. Can Companies Influence Investor Behaviour through Advertising? Super Bowl Commercials and Stock Returns. *European Financial Management,* 11(5), pp. 625-647.
12. Golas, U., 2022. *Open access to financial market data in the cloud via the Alpha Vantage API,* Berlin: Technical University of Berlin Library Blog.
13. Greenspan, H., Shen, D. & Zhou, K., 2017. An Introduction to Neural Networks and Deep Learning. In: *Deep Learning for Medical Image Analysis.* s.l.:s.n., pp. 3-24.
14. IvyPanda., 2022. *witter: Data Collection With API,* s.l.: s.n.

15. Kemp, S., 2022. *DIGITAL 2022: GLOBAL OVERVIEW REPORT,* s.l.: DATAREPORTAL.

16. Kotsiantis, S. B., Zaharakis, I. D. & Pintelas, P. E., 2006. Machine learning: a review of classification and combining techniques. *Artificial Intelligence Review,* 26(3), pp. 159-190.

17. Mao, H., Counts, S. & Bollen, J., 2011. *Predicting Financial Markets: Comparing Survey, News, Twitter and Search Engine Data.* s.l.:Cornell Univerity.

18. Marne, S., Churi, S., Correia, D. & Gomes, J., 2020. Predicting Price of Cryptocurrency - A Deep Learning Approach. *INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) NTASU,* 9(3).

19. Mcnally, S., Roche, J. & Caton, S., 2018. Predicting the Price of Bitcoin Using Machine Learning. *26th Euromicro International Conference on Parallel, Distributed and Network-based Processing(PDP),* pp. 339-343.

20. Meschke, J. F., 2002. *CEO Interviews on CNBC,* s.l.: Arizona State University.

21. Mittal, A. & Goel, A., 2011. Stock Prediction Using Twitter Sentiment Analysis.

22. Oliveira, N., Cortez, P. & Areal, N., 2017. The impact of microblogging data for stock market prediction: Using Twitter to predict returns, volatility, trading volume and survey sentiment indices. *Expert Systems with Applications,* Volume 73, pp. 125-144.

23. Pagolu, V. S., Challa, K. N. R., Panda, G. & Majhi, . B., 2016. *Sentiment Analysis of Twitter Data for Predicting Stock Market Movements.* s.l., International conference on Signal Processing, Communication, Power and Embedded System (SCOPES).

24. Ranco, G. et al., 2015. *The Effects of Twitter Sentiment on Stock Price Returns,* s.l.: PLOS ONE.

25. Smailovic, J., Grcar, M. & Žnidaršič, M., 2012. *Sentiment analysis on tweets in a financial domain,* s.l.: SEMANTIC SCHOLAR.

26. Smuts, N., 2019. What Drives Cryptocurrency Prices? An Investigation of Google Trends and Telegram Sentiment. *SIGMETRICS Perform. Eval. Rev.,* 46(3), pp. 131-134.

27. Sprenger, T. O. et al., 2014. Tweets and Trades: The Information Content of Stock Microblogs. *European Financial Management,* 20(5), pp. 926-957.

28. Takahashi, S., Takahashi, M., Takahashi, H. & Tsuda, K., 2007. *Analysis of the Relation Between Stock Price Returns and Headline News Using Text Categorization.* Berlin, Heidelberg: Springer.

29. Takeda, F. & Wakao, T., 2014. Google search intensity and its relationship with returns and trading volume of Japanese stocks. *Pacific-Basin Finance Journal,* pp. 1-18.

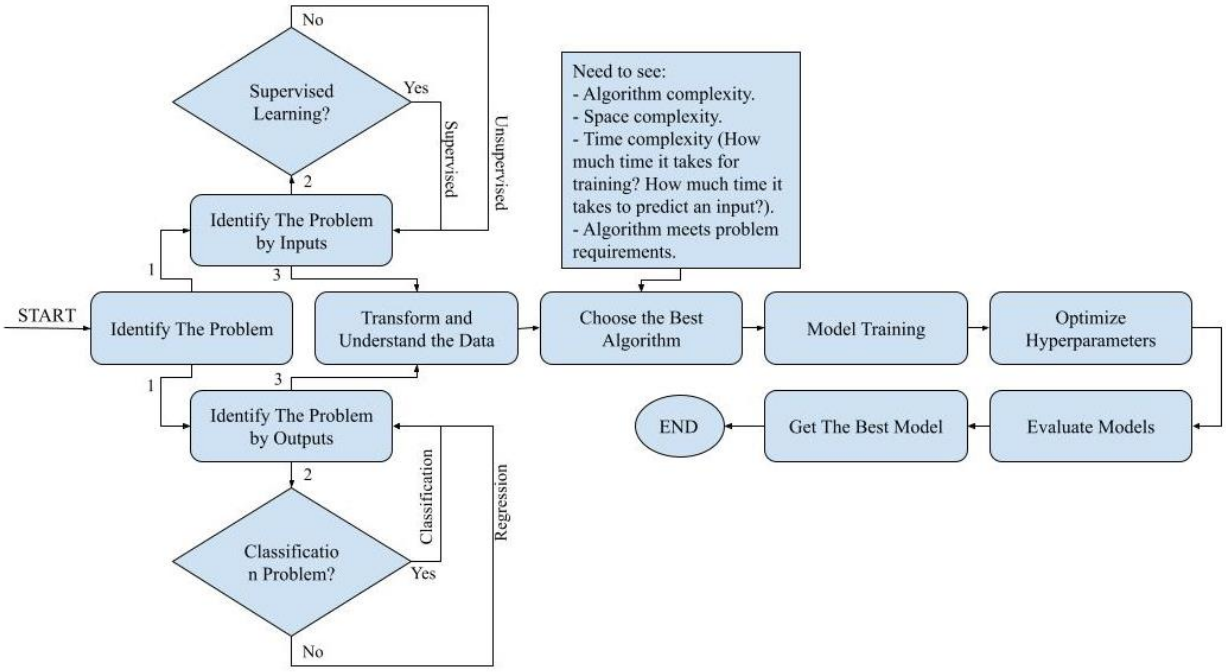
30. Tianqi Chen, C. G., 2016. *XGBoost: A Scalable Tree Boosting System,* s.l.: Cornell University.

31. Vlastakis, N. & Markellos, R. N., 2012. Information demand and stock market volatility. *Journal of Banking & Finance,* 36(6), pp. 1808-1821.

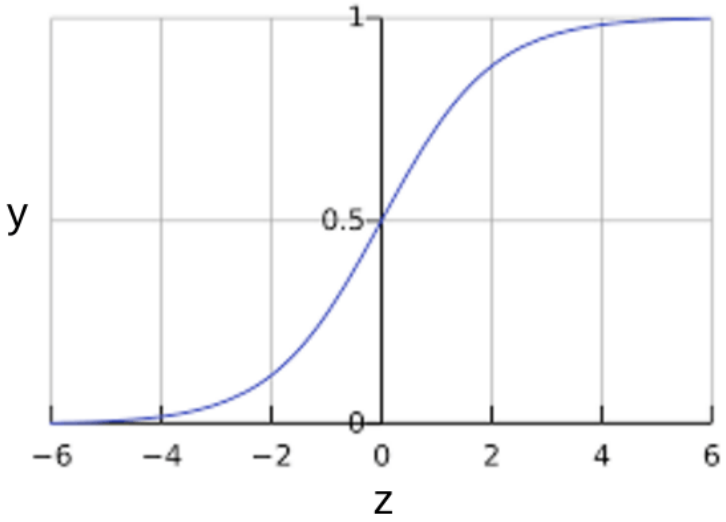
32. Zhang, X., Fuehres, H. & Gloor, P. A., 2011. Predicting Stock Market Indicators Through Twitter "I hope it is not as bad as I fear". *Procedia - Social and Behavioral Sciences,* Volume 26, pp. 55-62.

APPENDIX

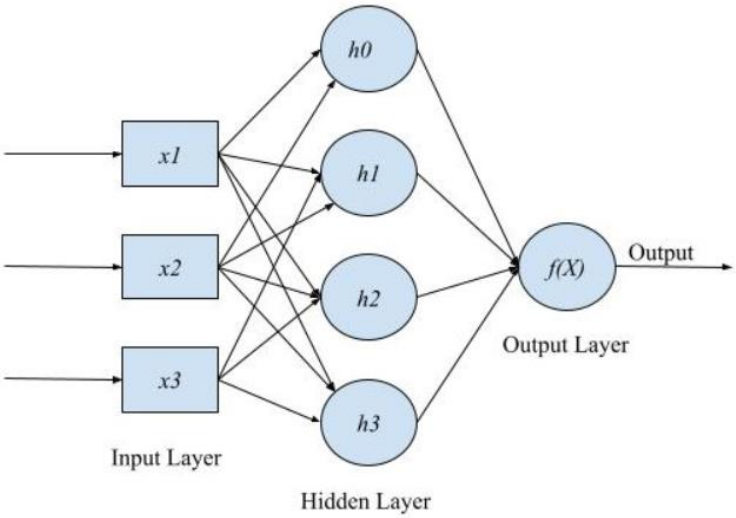
Appendix 1: Process to Find the Best Model



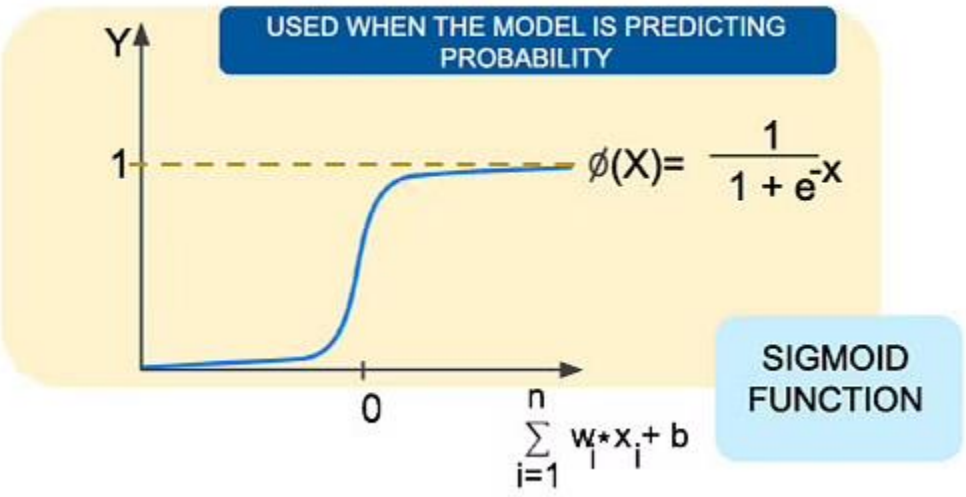
Appendix 2: Logistic Regression (Sigmoid Function)

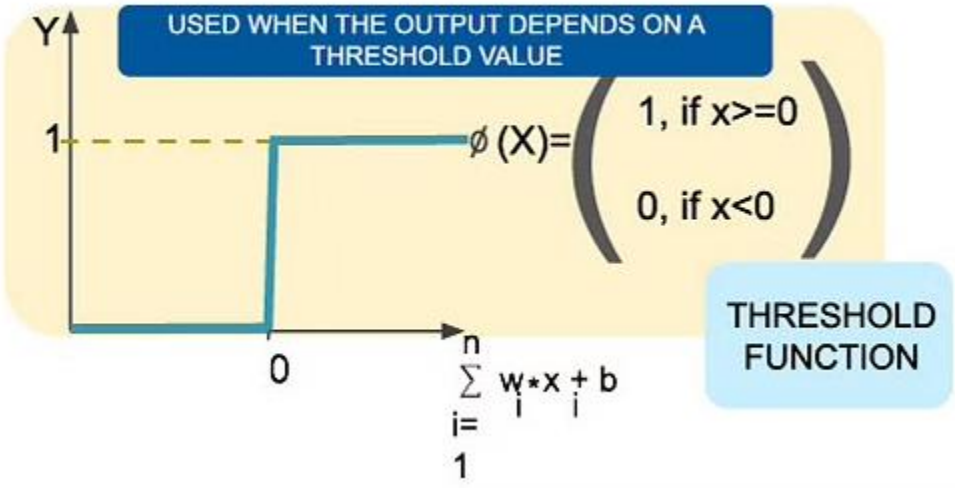


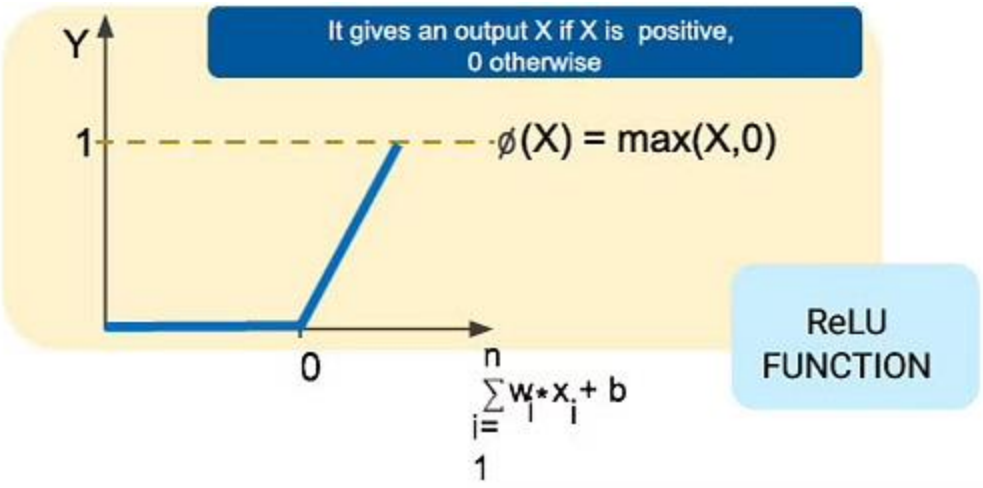
Appendix 3: A Multi-layer Perceptron Network



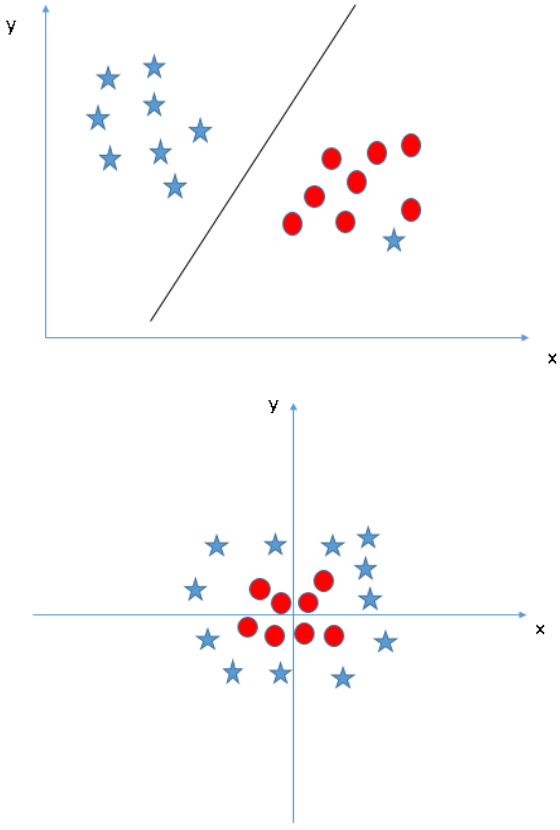
Appendix 4: Activation Functions of Multi-layer Perceptron







Appendix 5: Example of Linear and Non-linear Datapoints Respectively



Appendix 6: Used Hyperparameters and Set Values for Each Model

Model	Hyperparameters	Values
Logistic Regression (LR)	*C*, Regularizer.	"C": [0.1, 1, 10, 100]

Random Forest Classifier (RFC)	*n_estimators*, Number of Trees in the Forest. *max_depth*, The maximum depth of the tree.	"n_estimators": [5, 50, 250, 500], "max_depth": [2, 4, 8, 16, 32, 64, None]
Multi-layer Perceptron (MLP)	*hidden_layer_sizes*, Number of neurons in each hidden layer. *activation*, Activation function for the hidden layer. *learning_rate*, learning rate for weight updates.	"hidden_layer_sizes": [(10,), (50,), (100,)], "activation": ["relu", "tanh", "logistic"], "learning_rate": ["constant", "invscaling", "adaptive"]
Support Vector Classifier (SVC)	*C*, Regularizer. *kernel*, Denote the kernel of the algorithm. *decision_function_shape*, whether to return one-vs-rest(ovr) decision function shape or one-vs-one(ovo) decision function shape.	"C": [0.1, 1.0, 10], "kernel": ["linear","rbf","poly", "sigmoid"], "decision_function_shape": ["ovo", "ovr"]
Extreme Gradient Boosting (XGBoost)	*n_estimators*, Number of Trees in the Forest. *max_depth*, The maximum depth of the tree. *learning_rate*, learning rate for weight updates.	"n_estimators": [50, 100, 200], "max_depth": [1, 3, 5], "learning_rate": [0.001, 0.01, 0.1, 1.0]

Appendix 7: Best Estimated Model for Each Algorithm

Model	Best Estimated Model
Logistic Regression (LR)	Best Parameters: {'C': 10}
Random Forest Classifier (RFC)	Best Parameters: {'max_depth': 64, 'n_estimators': 50}
Multi-layer Perceptron (MLP)	Best Parameters: {'activation': 'relu', 'hidden_layer_sizes': (100,), 'learning_rate': 'constant'}
Support Vector Classifier (SVC)	Best Parameters: {'C': 1.0, 'decision_function_shape': 'ovo', 'kernel': 'linear'}
Extreme Gradient Boosting (XGBoost)	Best Parameters: {'learning_rate': 1.0, 'max_depth': 1, 'n_estimators': 100}