Twitter Data-Driven Security Analysis: Spam tweets & malicious URL identification

Marcos Pallikaras

School of Computing Science

Sir Alwyn Williams Building

University of Glasgow

G12 8RZ

A dissertation presented in part fulfillment of the requirements of the Degree of Master of Science at the University of Glasgow

December 10th, 2021

Abstract

Social engineering is an effective enabler to extract valuable information from inconspicuous social media users. Consequently, communication-based platforms, such as Twitter, need to ensure their application is prepared to identify such threats. Furthermore, the somewhat-overlooked technically inept users should be informed about the possible security threats that may arise throughout their use of the application and their likelihood. The aim of this project is to develop an application that attempts to identify and increase awareness of possible security attacks on Twitter. The project at a minimum will involve data science and development of a reusable system. The application will attempt to use machine learning techniques to classify tweets from the United Kingdom into spam or not-spam and produce statistics that can support communicating the information to low-level users in aspiration of improving their ability to identify and avoid them, therefore mitigating the problem.

Education Use Consent

I hereby give my permission for this project to be shown to other University of Glasgow students and to be distributed in an electronic form.

Name: Marcos Pallikaras Signature: [illegible]

Acknowledgements

I would like to express my gratitude towards my supervisor, Dr Joseph Maguire, for his aid and guidance throughout the course of the creation of this project.

Contents

Chapter 1 Introduction

1.1 Motivation

Many users have fallen victims to scam as the use of technology integrates its way into our daily lives at an exponentially increasing rate, with the attempts at attacks increasing in a similar way. Social media platforms, such as Twitter, open a door for social engineering as an enabler for these attacks to occur, with the main targets being technically inept users that are most likely unaware of the possible media and channels of attack. According to Twitter, they identified and challenged a constantly increasing amount of potentially spammy accounts with 9.9 million accounts per week in May 2018, a tremendous increase from 6.4 million per week in December 2017, and 3.2 million per week in September 2017. [1]

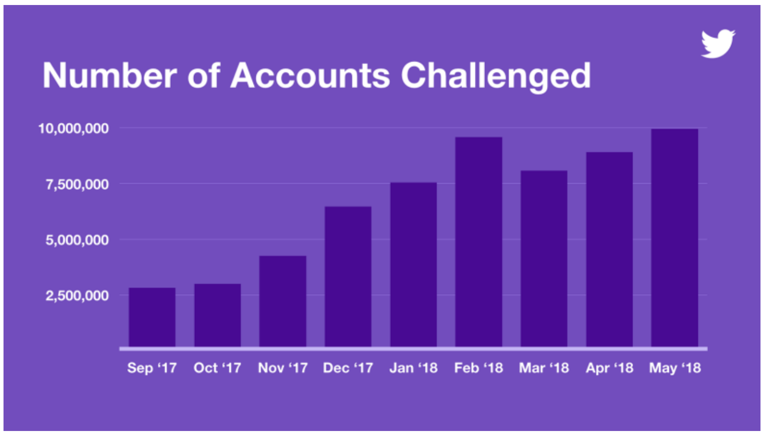


Figure 1: Number of potentially spammy or automated Twitter accounts challenged between September 2017 and May 2018. **[1]**

This increase in the attempts of attacks could correlate with a high rate of success for the attackers, which endangers the integrity of social media platforms even more, as well as the safety of their users, especially the individuals with low or no technical knowledge on how to stay secure whilst using them.

1.2 Aim and Objectives

In an attempt to improve the awareness of low-level users, this project aims to provide statistics for some of the potential threats posed to them:

- Spam tweets
- Malicious URLs posted in tweets.

This is done by utilizing a dataset of labelled tweets as spam/not-spam and its metadata as the training data fed to various machine learning models in order to train an accurate classifier. The classifier should be able to predict if a never-seen-before tweet is considered spam or not spam. Furthermore, the maliciousness of URLs will be verified by a variety of website/domain scanning engines & datasets by utilizing VirusTotal's services.

1.3 Outline

The subsequent sections of the dissertation follow through background analysis with statistics gathered directly from Twitter, as well as previous pieces of relevant work attempted. Hence follows a requirements analysis to identify what this project needs to bring to the table, preparatory to the design section on how this project can be put together and its technical aspects. Finally comes the implementation, as well as the testing and evaluation strategies followed throughout the project's lifecycle, leading up to the conclusion and mentions of possible additions and improvements to the project as future work.

Chapter 2 Background Survey

2.1 Problem Analysis

According to Twitter, spam is a form of platform manipulation, which refers to “the unauthorized use of Twitter to mislead others and/or disrupt their experience by engaging in bulk, aggressive, or deceptive activity.” [2] Spam can include, but is not limited to, unsolicited or repeated actions, malicious automation, and the creation of fake accounts to deploy them. [3] Some of the more noticeable categories of spam on social media platforms like Twitter are:

- “Commercially-motivated spam, that typically aims to drive traffic or attention from a conversation on Twitter to accounts, websites, products, services, or initiatives.” [4]
- “Coordinated activity, that attempts to artificially influence conversations through the use of multiple accounts, fake accounts, automation and/or scripting” [4]

Despite the 140-character limit Twitter imposes on tweets giving limited flexibility to spammers, they make use of URL shorteners that completely change the visibility of the URL’s true domain, which obfuscates it from the user. This can also prevent Twitter from effectively filtering them with their current application of blacklists. Furthermore, an estimated 80% of shortened URLs lead to spam-related content. [5]

Compared to the last reporting period, Twitter reports a 6% increase in global anti-spam challenges, and a 14% decrease in global spam reports. [2]

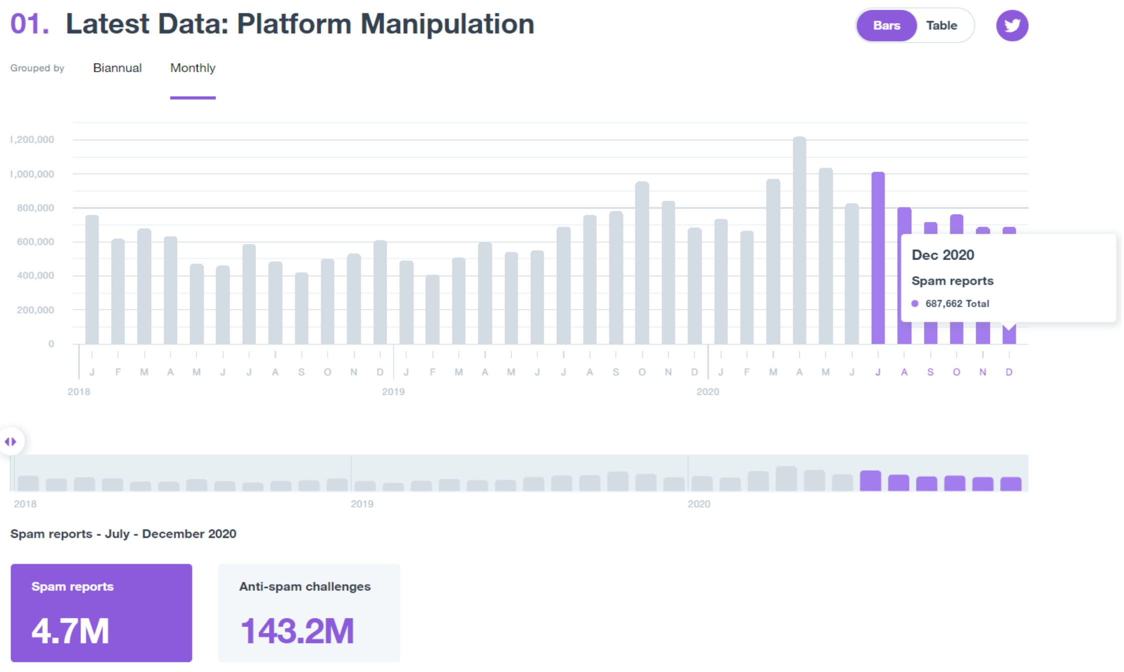


Figure 2: Bar chart of number of spam reports between Jan 2018 and Dec 2020 posted by Twitter. **[2]**

2.2 Previous Work

2.2.1 Seven Months with the Devils: A Long-Term Study of Content Polluters on Twitter [6]

Lee et al. present a system for automatic detection and profiling of spam Twitter users. Through the use of a honeypot-derived technique, they lured and harvested 36,000 candidate spam users, *or content polluters as they are referred to as in the paper*, over a seven-month period and the deployment of 60 honeypots. Upon validation to check if the harvested accounts were considered as official violators by Twitter's terms of service, only 23% of those accounts were eventually suspended. Lee et al. further examined and analysed the content polluter accounts to extract potential features that correlated which could then be used during the automatic classification phase. Features such as the number of followings, followers, tweets posted, the length of screen name and description in user profile were deemed as useful.

For the classification, 30 different machine learning algorithms from the Weka machine learning toolkit were used with their default hyperparameters with varying results. With performances ranging from 89% up to 98.42% accuracy score, and with their best classifier being Random Forest, they provide strong evidence of the existence of strong significance of the features used for the robust classification of content polluter accounts.

Limitations: Although Lee et al. make use of many models for classification in the experiment, they only use the default model hyperparameters for all models, leaving possibilities for a higher score by the optimization of model hyperparameters.

2.2.2 Birds of prey: identifying lexical irregularities in spam on Twitter [7]

Robinson et al. present a lexical content analysis of malicious spam on Twitter in an attempt to identify associated textual regularities such as terms and symbols, that could aid in their identification and classification via the use of machine learning.

A dataset of 14,414 tweets of equal spam/non-spam tweet distribution was used to extract various features that can be used during the classification, such as the amount of emojis, hashtags (#), mentions (@), numbers (e.g., 10) and URLs. These were used to train a Random Forest classifier with a final classification accuracy greater than 90% after the optimization of all its hyperparameters.

Additionally, by processing each tweet's text, four frequency-mapped corpora were created for each class (spam/non-spam); a words corpus, an emoji corpus, a numbers corpus, and a stop-words corpus. To convey the differences between the two classes, they produced visualizations from each corpus against each other to observe a clear distinction between them.

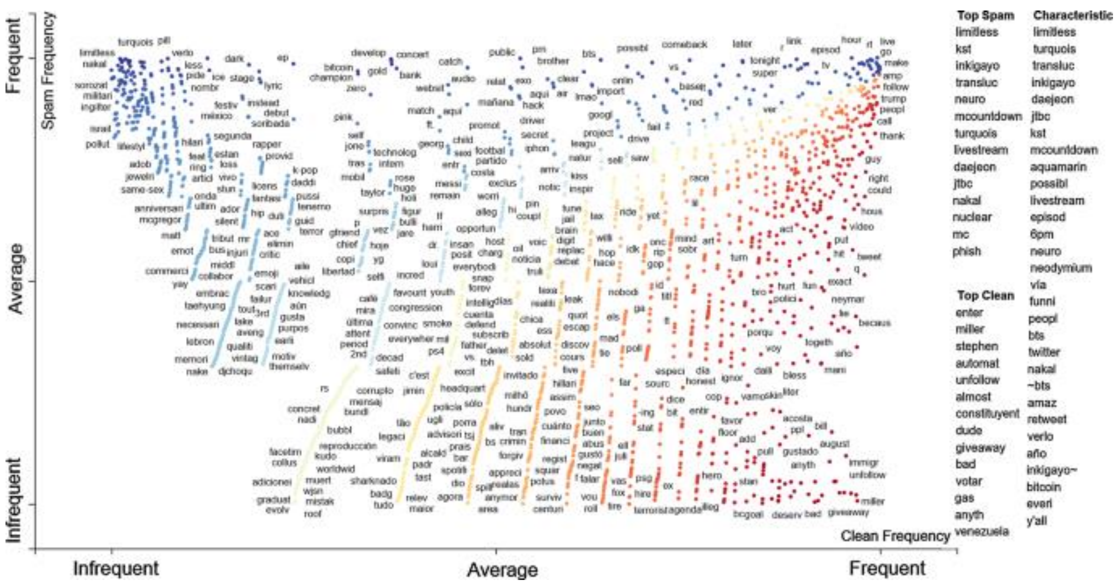


Figure 3: Frequency of words found in tweets classified as spam versus frequency of words found in tweets classified as non-spam (clean). **[7]**

Limitations: Although Robinson et al. perform an in-depth analysis and extraction of valuable metadata used as features in the classification as well as optimization of the model's hyperparameters to achieve a high accuracy score, they only make use of one model, leaving possibilities for a higher scoring model.

Chapter 3 Requirement Analysis

The primary objective of this project is to provide a method of classifying tweets as spam or not spam, with a similar or better accuracy than similar projects, by utilizing a machine learning approach.

The extraction and utilization of metadata is an aspect of utmost importance for this project, as it will be the determining factor for the improvement of classification. Lee et al. in 2.2.1 extract and make use of the number of followings, the number of followers, the number of tweets posted, the length of screen name and description in user profile for each user and associate it with each of their tweets [6]. Robinson et al. in 2.2.2 delve deeper into the tweets' textual data and extract numerical features such as the count of hashtags, mentions, numbers and URLs. In an attempt to improve the accuracy of classification, a combination of both papers' features proposals will be used. This will provide both numerical features, as well as statistics from lexical analysis.

Furthermore, the time difference between the user account's creation and the tweet's time of posting will be used as an extra feature. The logic behind this feature is that spam users tend to have younger accounts relative to the time tweet posting, in other words, a smaller time difference. This could be due to the suspension of their previous account, or just the spontaneity of creating an account solely for posting spam. Additionally, further preprocessing of the tweets' text by using feature extraction methods such as TF-IDF will provide more features to be used in the classification.

Moreover, statistics of the maliciousness of URLs collected from tweets as payload will be attempted as a secondary objective to provide an expanded statistical insight.

3.1 Functional Requirements

- **Machine Learning approach:**
  The classification of tweets as spam/non-spam should be through the use of a machine learning approach and the utilization of appropriate models produced.
- **Training Dataset:**
  A dataset of tweets and their metadata, labelled as spam/not spam to be used as the training data for the classifier models.
- **Tweet Collection:**
  A way for collecting recently posted tweets and their metadata to be classified.
- **Textual Data Preprocessing:**
  A way to preprocess textual data from the tweets and extract any valuable and useful metadata.
- **Tweet Classification:**
  A way to classify tweets into spam/not spam via the use of the trained classifier models.
- **URL Maliciousness Classification:**
  A way to classify URL maliciousness.

3.2 Non-functional Requirements

- **Visual & Interactive Results:**
  The system should provide visual and interactive results where possible to improve conveying the results to the user.
- **Scalability:**
  The system should be scalable to handle the classification of at least 100,000 tweets within a few minutes.
  The system should be able to save trained models for rapid deployment for tweet classification.
- **Universality:**
  The system should be able to classify any tweet provided in its original format.

3.3 Features (MoSCoW prioritization)

- **Must have:**
  - A range of basic classifiers that can classify tweets into spam/not spam using their textual data.
- **Should have:**
  - A tweet collector to gather new tweets to be classified.
  - A basic storage method for collected data.
  - Textual data preprocessing for metadata extraction.
  - An advanced model that can classify tweets with numerical features extracted from the tweets' metadata.
  - A way to save models after training so that they don't have to be trained every time for classification.
  - A way to determine if URLs embedded in tweets are malicious.
- **Could have:**
  - Visualizations to aid presentation of results.
  - Categorical features from metadata used in advanced model classification.
  - Real-time tweet classification.
  - Use different models for numerical and categorical features and combine with pipelines.
- **Won't have:**
  - Feature Selection to determine which features are improving the classifiers' performance and discard the rest.
  - Database for storing and loading collected tweets and metadata more efficiently.
  - Graphical User Interface.

Chapter 4 Design

4.1 Technology Choices

The programming language of choice for this project is Python 3.9.9. Its versatility and availability of libraries for the project's requirements make it the perfect choice. Below are the main libraries used and their purpose:

4.1.1 Data Collection (Twitter Streaming API & Tweepy) [8, 9]

There are various ways of collecting tweets, but Twitter provides its own official Streaming API for free which allows for an educational use upgrade to Elevated status. This allows for the collection of 2 million tweets per month, with a rate limit of 50 requests per 15 minutes.

Tweepy 3.10.0 provides useful methods that ease the implementation and integration of the Twitter API, such as the ability to automatically keep within the limits of the rate quota when collecting tweets.

4.1.2 Machine Learning (Scikit-learn)

The 5 most common models used in textual classification are as follows:

Logistic Regression [10]

Logistic Regression is a probabilistic classifier that makes use of a function to plot a boundary that separates the classes and attempts to minimize the loss by making adjustments to the loss function.

K-Nearest Neighbors [11]

K-Nearest Neighbors is a fast non-probabilistic classifier which takes its input in vector form and classifies it, as the name suggests, depending on a specified amount (k) of its nearest neighbors in the vector space.

Support Vector Machine [12]

SVM is a binary classifier which sets a decision boundary in the vector space of its input data and minimizes the margin, which is the perpendicular distance from the decision boundary to the closest points from each class.

Multinomial Naïve Bayes [13]

Multinomial Naïve Bayes is statistical algorithm which provides a probabilistic classifier that classifies each instance by the frequency of its appearance, hence provides the probability of each tweet being in each class.

Decision Tree [14]

A Decision Tree classifier is, as the name suggests, a tree-like model of decisions with the leaves being the classes and the branches being conjunctions of features leading to each class.

4.1.2.1 TF-IDF Vectorizer [15]

TF-IDF (Term Frequency – Inverse Document Frequency) allows the quantification of relevance of each term/word in textual data, the tweet's text in this case. Essentially, it is a term weighting scheme that for text similarity. It is deemed as the most popular method today due to its ability to adjust to words appearing very frequently compared to others. It can improve the classification as the term weighting can help identify lexical regularities within the tweets' text.

4.1.2.2 Evaluation [16]

To evaluate the models, SKLearn provides evaluation libraries that can be easily integrated to be used with the trained models and provide useful classification metrics and insights to the models' performance.

Confusion Matrix:

A confusion matrix summarizes the results of a classifier's predictions against the actual labels in a table format using the following metrics:

- True Positives: *Spam tweets classified correctly as spam.*
- True Negatives: *Non-spam tweets classified correctly as non-spam.*
- False Positives: *Non-spam tweets classified incorrectly as spam.*
- False Negatives: *Spam tweets classified incorrectly as non-spam.*

Accuracy:

The Accuracy score is a ratio of all the correctly labelled tweets (*TP* + *TN*) to the total amount of tweets.

Precision:

The Precision score is the ratio of True Positives to the total amount of predicted Positives (*TP* + *FP*). This is the ratio of correctly classified spam tweets to the total amount of classified spam tweets.

Recall:

The Recall score is the ratio of True Positives to the total amount of Positives (*TP + FN).* This is the ratio of correctly classified spam tweets to the total amount of spam tweets.

Macro F1:

The F1 score is the harmonic mean of Precision and Recall, between 0-1, with a higher score being a better score. The macro-averaged F1 score is used when the training data classes are balanced, which means that they have the same number of examples for each label; Spam/Not Spam.

All metrics are between 0-1, with a higher score meaning that the classifier performs better.

The Macro F1 metric will be the deciding evaluation metric to compare the models' performance, as unlike accuracy, it penalizes extreme values and gives a better measure of the incorrectly classified cases.

4.1.2.3 GridSearchCV [17]

Each model has a set of parameters that determine how it functions. A different set of parameters might be optimal for each situation. To find the optimal hyperparameters for each model in this situation, GridSearchCV is used.

GridSearchCV "performs an exhaustive search over specified parameter values" [17] as the hyperparameters and makes use of the validation data to be used as never-seen-before testing data to evaluate the performance of each version of the model trained. It returns the optimal hyperparameters that maximize a specified evaluation metric score. In this case, the macro-averaged F1 score was specified as the deciding score metric.

4.1.3 Natural Language Processing (SpaCy's Tokenizer) [18]

To extract any potentially useful features from the tweet's textual data through the utilization of natural language processing, SpaCy's NLP library will be used in combination with a custom tokenizer.

SpaCy's NLP utilizes a language model that can be used to break down text into linguistic annotations by performing tokenization and provides easy access to them. SpaCy provides a various selection of language models in different languages, but the model selected for this project is **'en_core_web_sm' 3.0.0**, which is a small English language model with a trained pipeline optimized for CPU usage, as the project will be run on a personal computer.

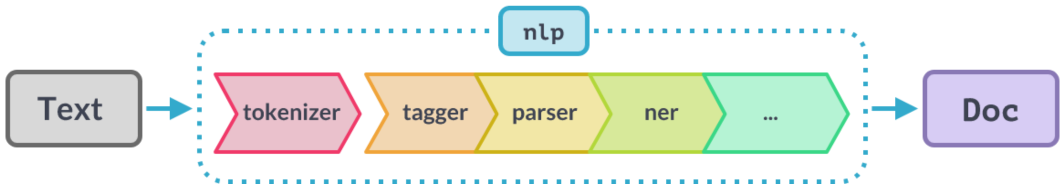


Figure 4: SpaCy's Tokenizer Pipeline. **[18]**

SpaCy also allows the creation and integration of a custom tokenizer method which will be useful as the intention is to also extract specific metadata, such as emojis, stop words, numerical tokens, mentions, hashtags and URLs. Furthermore, SpaCy provides a lemmatization option, which allows for the reduction of a word to its root word. This can group words with the same root into one, and in combination with TF-IDF it will produce a more accurate vectorization.

4.1.4 URL Threat Classification (VirusTotal API) [19]

VirusTotal is a free service that collaborates with members of the antivirus industry to provide malicious content analysis for files and URLs. In this project, its free API is used for the identification of malicious URLs. Each URL receives votes from 83 different contributors that determine if its malicious or not. The contributors are website/domain scanning engines & datasets. Although free, the API comes with a rate limit of 4 requests per minute and 500 requests per day.

4.1.5 Results Visualization (Plotly Pie Charts) [20]

Plotly provides a variety of interactive data visualization libraries that can be used to visualize the project's statistical results. The project mainly focuses on the prevalence of these threats; therefore, a pie chart was deemed the best type of visualization for both the classification of spam tweets and malicious URLs.

4.1.6 Version Control

A Git repository[1] was set up on GitHub for version control as well as way of mitigation of potential and unexpected loss of data.

4.2 Social Honeypot Dataset [6, 21]

To train the classifiers, a sizeable, dependable, and as-recent-as-possible dataset is required to ensure as accurate classifications as possible. At a minimum, the dataset must contain tweets with their textual data labelled as "Spam" or "Not Spam".

The only available dataset found that fit most of the aforementioned requirements was the dataset used in 2.2.1 Seven Months with the Devils: A Long-Term Study of Content Polluters on Twitter. As previously mentioned, Lee et al. refer to spam users as "content polluters" and non-spam users as "legitimate users". The social honeypot dataset contains data collected from December 30th, 2009, to August 2nd, 2010, on Twitter, and is divided into 6 different subsets as text files, out of which 4 are useful and will be used for the project:

1. **content_polluters.txt:**
   Contains 22,223 spam users' profile information: *(User ID, Account Creation Time, Time of collection, Number of Followings, Number of Followers, Number of Tweets, Length of Screen Name, Length of Description in User Profile).*
2. **content_polluters_tweets.txt:**
   Contains 2,353,473 spam tweets and their information: *(User ID, Tweet ID, Tweet, Time of Posting).*
3. **legitimate_users.txt:**
   Contains 19,276 non-spam users' profile information: *(User ID, Account Creation Time, Time of collection, Number of Followings, Number of Followers, Number of Tweets, Length of Screen Name, Length of Description in User Profile).*
4. **legitimate_users_tweets.txt:**
   Contains 3,259,693 non-spam tweets and their information: *(User ID, Tweet ID, Tweet, Time of Posting).*

It is important to note that the dataset contains data that is almost 12 years old at the time of writing. This can have a negative impact at the classification of more recent tweets due to possible changes in the way people communicate/write on social media (e.g., new words, alternate meaning of words etc.). Unfortunately, due to lack of availability of more recent datasets, this is deemed to be the best dataset to use at the time of implementation.

[1] https://github.com/Ghosthunter22/Twitter-Data-Driven-Security-Threat-Analysis

Chapter 5 Implementation

5.1 Data Preprocessing

The dataset used requires some preprocessing for its use in this project. To avoid having to process the dataset every time the code is run, **Data Preprocessor.ipynb** is responsible for preprocessing the dataset to its final structure and save it externally so that it can be loaded straight into memory when needed without the need of preprocessing it every time.

As mentioned in **section 4.2**, the dataset comes in different parts. First, the spam tweets and non-spam tweets are read into memory and parsed as Pandas DataFrames to ease data manipulation. A Boolean (True, False) label is added to each tweet to represent Spam/Not Spam respectively. There were some cases of "empty" tweets (tweets that have no textual data) occurring in the dataset that cannot be used; hence the dataset is cleaned up by removing any empty tweets.

Next up, the spam user and non-spam user data is loaded in a similar manner. The user information is then appended onto their respective tweets by merging onto User ID. Unfortunately, when merging the tweets' data with the user data, only 7,872 non-spam tweets had data available for their respective users, hence the non-spam tweets class had been narrowed down to 7,872 tweets. When training the models, if the spam tweets class remained at 2,353,469 tweets, there would exist class imbalance, which could hinder the model's performance and generalizability. Using 7,872 tweets from each class was deemed as the best solution, as it's still a sizeable and adequate training dataset, coming to a total of 15,744 tweets from both classes.

The time difference (time delta) between the time of the tweet being posted and the respective user's account being created, as mentioned in **Requirement Analysis**, is then calculated for each tweet and appended to the DataFrames to give them their final form.

The preprocessed datasets are then finally saved as .csv files 'content_polluters_tweets_pp.csv' and 'legitimate_users_tweets_pp.csv' under the 'Data' folder. The structure of the data is as follows:

[Index, User ID, Tweet ID, Tweet, Created At, Spam, User Created At, Number of Followings, Number of Followers, Number of Tweets, Length of Screen Name, Length of Description in User Profile, Time Delta(Days)]

5.2 Tweet Collector

Tweet Collector.ipynb is responsible for the collection of new tweets that will be used for classification later on. By utilizing Twitter's API and Tweepy, a query is made every 15 seconds with the constraint of country being 'United Kingdom'. Retweets are also filtered out. The tweets received from the query are then checked against the already collected tweets for duplicates, and if the language they are written in is English. The tweets are received and kept in their original data form as JSON objects. To retain all of their information and metadata, the tweets are cast to a String and are stored in a text file ('collected_tweet_data.txt') separated by new line characters ('`\n`'). The 15 second delay between requests is intended so that the same tweets won't be collected before new ones are posted and using up the quota more efficiently. Tweepy ensures that collection rates are kept within the rate limit set by Twitter by setting `wait_on_rate_limit=True` when setting up the API.

5.3 Machine Learning Models

Model Training.ipynb is responsible for the training of the basic and advanced machine learning models to be used for the classification.

Initially, the 5 most common models used in textual classification, mentioned in **section 4.1.2**, have been used to create the basic models. The same procedure was carried for each basic model. A general data-flow diagram for the basic classifiers can be found below:

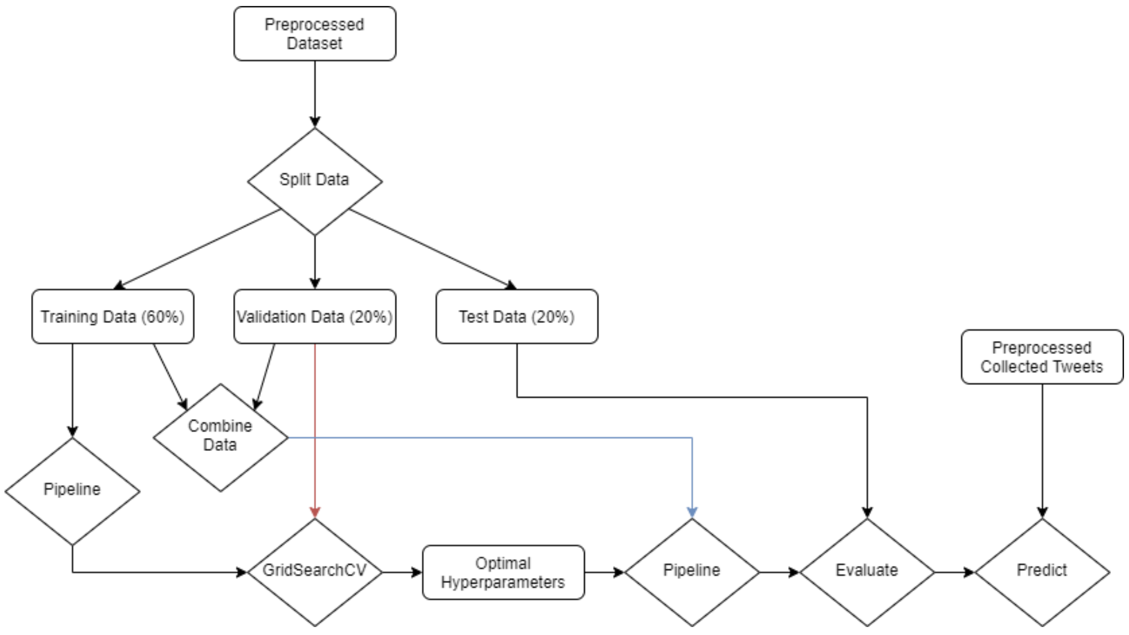


Figure 5: Data-flow diagram for classifiers.

5.3.1 Splitting the data

The preprocessed dataset's textual data was split up into training (60%), validation (20%), and test (20%) subsets using Scikit-learn's train_test_split library. The training data was used to train the model with its default parameters to create the base version of the model. The validation data was used for optimizing each model's hyperparameters and finally the test data was used to evaluate its performance.

5.3.2 Basic Pipeline

A pipeline from Scikit-learn was created for each model. Each pipeline consisted of a TF-IDF vectorizer [15] followed by the model itself. The text from each tweet is fed into the pipeline, which first applies TF-IDF vectorization to the text and produces a vectorized form, consequently fed to the model.

5.3.3 Tuning Model Hyperparameters with GridSearchCV

After training the basic models with their default parameters, Scikit-learn's GridSearchCV was used with the validation data to determine the optimal hyperparameters for the model. Scoring was set to macro-averaged F1 score, so that the optimal hyperparameters would maximize that metric.

Once retrieving the optimal hyperparameters for the model, the model was retrained by combining the training and validation subsets as a bigger training subset. The model was finally evaluated with the test data used as never-seen-before data.

5.4 Advanced Model

After training, optimizing, and evaluating all the basic models, the best performing model was chosen to produce a more advanced model.

5.4.1 Numerical features from tweets' metadata

Some of the numerical features from the tweets' metadata are already stored in the dataset, as mentioned in **section 4.2**. In order to obtain the tweets' numerical features from their metadata, they go through a textual preprocessing method which includes a custom tokenizer utilizing SpaCy's NLP tokenizer which tokenizes the text, extracts emojis, URLs, hashtags (#), mentions (@), stop-words, and numbers (e.g., 10) into separate lists, and lemmatizes the remaining tokens.

Finally, the counts of numbers, emojis and stop-words are appended to the numerical features attained from the dataset and the time delta, with the final set of numerical features being:

["Number of Followers", "Number of Tweets", "Length of Screen Name", "Length of Description in User Profile", "Time Delta (Days)", "Numerical", "Emoji count", "Stop-word count"]

These features are combined with the TF-IDF vectorized tokenized version of the text in a Column Transformer and then fed to a Support Vector Machine classifier with its optimized hyperparameters to create the advanced model.

5.5 Collected Tweets Classification

5.5.1 Preprocessing & Metadata Extraction

The collected tweets are loaded into memory from 'collected_tweet_data.txt', where they are stored in their originally retrieved JSON format. In order to feed these tweets into the classifiers for prediction, they first need to be preprocessed to be in the correct format, and to extract the required metadata used as features in the classification.

Firstly, the required data is extracted and parsed from its JSON format into its respective format (e.g., tweet's text into String), which are once again then stored in a Pandas DataFrame for further operations. Each row is a tweet with its extracted metadata. Finally, each tweet's textual metadata is extracted as features in the same way as mentioned in **section 5.4.1**.

5.5.2 Predictions & Result Visualization

In order to classify the tweets with a trained classifier, it must be loaded into memory from its saved format. The advanced model is loaded as `adv_model`. For classifications, the `predict_tweets_adv()` method is created, which takes in the model and tweets to be classified as its two parameters. It handles taking the required data from the tweets in the DataFrame and feeds it into the advanced model and retrieves its predictions in the form of an array of True/False values, which stands for Spam and Not Spam respectively. The method also handles creating an interactive Plotly Pie Chart to visualize the results.

5.6 URL Threat Classification

During the preprocessing of the collected tweets in **section 5.5.1**, any URLs in the tweets' payload is extracted and appended to the DataFrame where they are stored. By utilizing VirusTotal's API (**section 4.1.4**), the system runs through all the collected URLs, creates and sends a POST analysis request to VirusTotal's servers, returning a request ID, which is then used in a report retrieval GET request to retrieve the results.

Results come in JSON form, which is accessed to retrieve the number of contributors stating if the URL is 'Malicious' or 'Harmless'. If the URL receives no votes for being malicious, it is classified as 'Harmless'. If the URL receives between 1 and 10 (inclusive) votes for being malicious, it is classified as 'Potentially Malicious'. If the URL receives more than 10 votes for being malicious, then it is classified as 'Malicious'. Finally, if the URL receives no votes, it is classified as 'Unknown'.

Due to the API's strict quota restrictions, it wouldn't be possible for all the collected URLs to be analysed by the time of submission. To address this, URLs whose domains were considered common and safe, namely Twitter, Instagram, and YouTube, were skipped and classified as 'Harmless' as a heuristic. This allowed for the rest of the collected URLs to be analysed in time.

5.7 Testing

Testing consisted of input and output data validation, as well as pre-train tests in methods used by all the models during the model training phase in the form of white box testing, throughout the implementation. By utilizing Python's try blocks, assertions were made on the data's format (e.g., shape, size, length), as well as type, for both input and output variables. This ensured that input and output data was as expected.

Chapter 6 Evaluation & Results

6.1 Evaluation

To train each model, the data was split into **training**, **validation**, and **test** subsets, in **60%**, **20%**, **20%** ratios respectively. After training each basic model with the **training** data and optimizing its hyperparameters with the **validation** data, the **test** data was used as never-seen-before data to be predicted by the model. The true labels of the classification are known; therefore, the model's performance could be evaluated using the evaluation metrics discussed earlier.

The macro-averaged F1 metric was used to determine the best performing basic model so that its classifier would be used in the advanced model. Furthermore, a confusion matrix was created to visualize the performance of the classifier.

6.2 Results

6.2.1 Basic Models

The Logistic Regression and Support Vector Machine models were the best performing models with a macro F1 score of **0.906** and **0.913** respectively. Then followed Multinomial Naïve Bayes with a respectable score of **0.893**.

K-Nearest Neighbors and Decision Tree did not seem to be suitable for the task, as they had scores of **0.727** and **0.797** respectively. The detailed scores, confusion matrices and optimal hyperparameters for each model can be found in **Appendix B** .

6.2.2 Advanced Model

The **Support Vector Classifier** was deemed the best with a macro-averaged F1 score of **0.913**, hence it was used to create the advanced model. After training and testing the advanced model, its macro-averaged F1 score rose to **0.954**. This was by far the best performing model to be used for the classification of the collected tweets.

After processing the collected tweets, the data was fed into the model to be predicted. From a total of **100,000** tweets, **12,465** were classified as **spam**, and **87,535** as **not spam**. The results are visualized in a pie chart below:

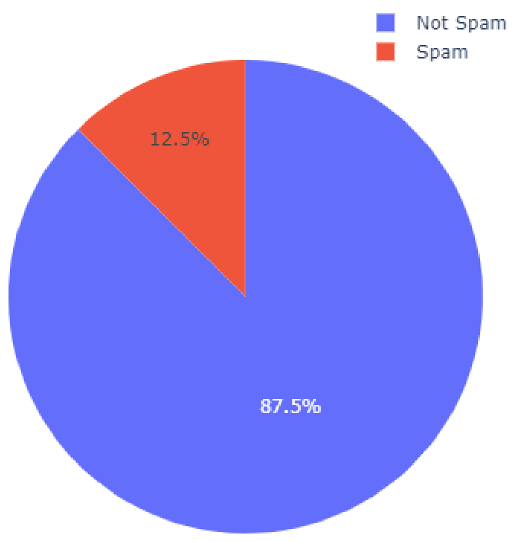


Figure 6: Advanced Model collected tweets classification results.

6.2.3 URL Classification

By utilizing VirusTotal's API, requests had been sent for all the URLs embedded within the tweets collected. From a total of **3,085** URLs requested for analysis, **151** were classified as **harmless**, **46** as **potentially malicious**. Unfortunately, no classifications were received for the rest, which is a very large proportion, marked as '**Unknown**'. The results are visualized in a pie chart below:

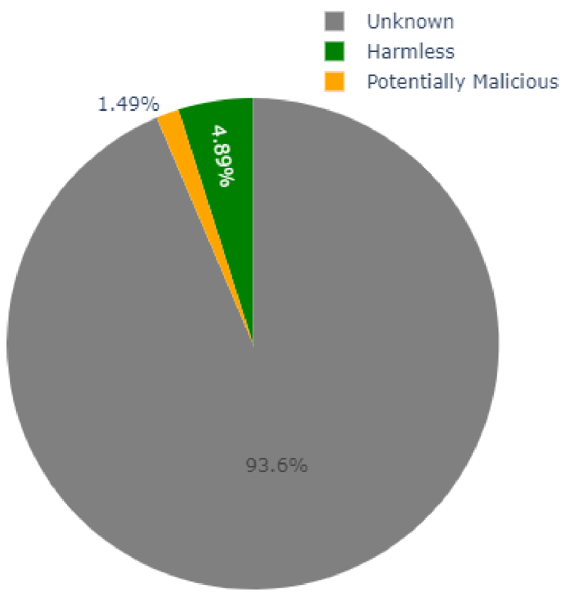


Figure 7: URL Classification results.

Chapter 7 Conclusion

7.1 Reflection & Contribution

Conclusively, all of the project's basic features have been implemented including some additional features such as the visualization of results. Unfortunately, the rest of the features have not been attempted or implemented due to lack of time but can be achieved in the future.

However, there are some shortcomings, as previously mentioned in **section 4.2**, the dataset used to train the classifiers is quite aged due to lack of availability of more recent datasets, which will have a negative effect on the classifiers' accuracy when classifying recent data. Furthermore, the URL Classification was not as effective as originally desired, as VirusTotal contributors did not provide data for most of the URLs sent for classification. This could be due to a short URL lifetime at the time of collection and testing, as the contributors might not have had enough time to detect and classify them (e.g., news articles). This could not be mitigated, as there wasn't enough time to collect URLs and classify them at a later stage to allow time for the contributors to classify them.

Nonetheless, during the evaluation, similar accuracy scores were achieved with previous work, and the results can provide a relatively accurate statistical insight. A percentage of around 12.5% of tweets posted being spam and only about 5% of the URLs posted being successfully and decisively classified as 'Harmless', reinforces the fact that technically inept users should exercise extreme caution when coming across unknown material not just on social media platforms, but in their entire online experience in their daily lives.

7.2 Future Work

There are various ways to improve on, both in terms of the performance of classification, as well as the addition of extra features and technical improvements, such as:

- The use of a more recent labelled dataset to train models, in order to improve the accuracy of classification.
- The inclusion of a wider variety of models and the optimization of their hyperparameters to potentially get a better performing model.
- The use of more recent state-of-the-art BERT model.
- The use of Feature Selection to determine which features are improving the classifiers' performance and which don't.
- The implementation of real-time tweet collection and classification.

Technical Improvements:

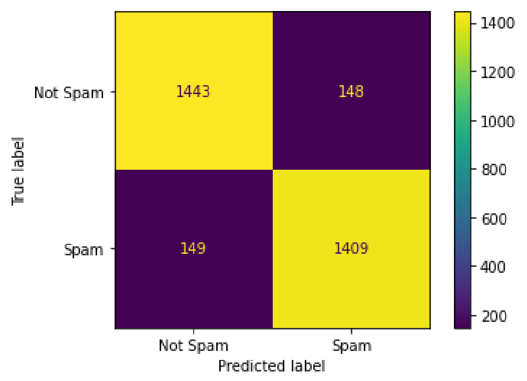
- The integration of a database for more efficient storing and loading of data.
- The integration of a Graphical User Interface to improve user experience.

Appendix A Test Cases & Results

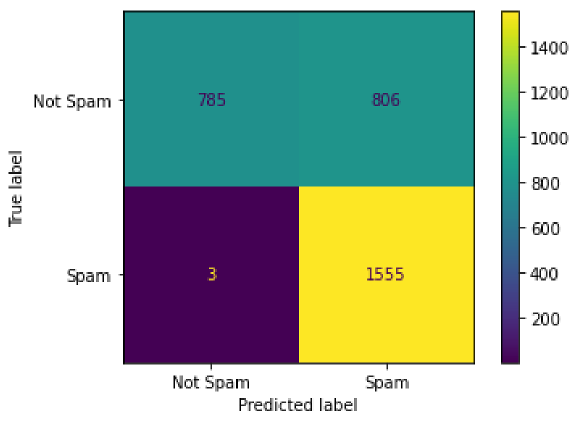
Model Training Data input	
Content Polluters' tweets must be a Pandas DataFrame.	PASS
Legitimate Users' tweets must be a Pandas DataFrame.	PASS
Content Polluters' tweets and Legitimate Users' tweets DataFrames must be of identical shapes.	PASS
Content Polluters' tweets dataset must not contain empty (np.nan) tweets.	PASS
Legitimate Users' tweets dataset must not contain empty (np.nan) tweets.	PASS
Model Training Data Splitting	
Input must be a Pandas DataFrame.	PASS
Split data must be of identical length to input dataset.	PASS
Split labels must be of identical length to input dataset.	PASS
Training data and labels must be of equal lengths.	PASS
Test data and labels must be of equal lengths.	PASS
Model Training Evaluation Summary	
Description must be a string.	PASS
Predictions must be a Pandas Series.	PASS
True labels must be a Pandas Series.	PASS
Model Training Grid Search	
Pipeline must be a SKLearn Pipeline.	PASS
Training data must be a length 2 tuple of Pandas Series.	PASS
Parameters must be a Pandas Dictionary.	PASS

Appendix B Model Evaluation Results

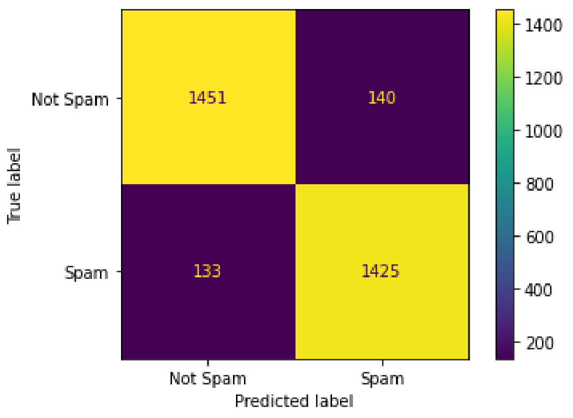
Logistic Regression Basic Model Evaluation:

Confusion Matrix	Accuracy	0.906
	Precision	0.906
	Recall	0.906
	Macro F1	0.906
	Optimal Hyperparameters:	
	TF-IDF sublinear_tf	True
	LR C	100

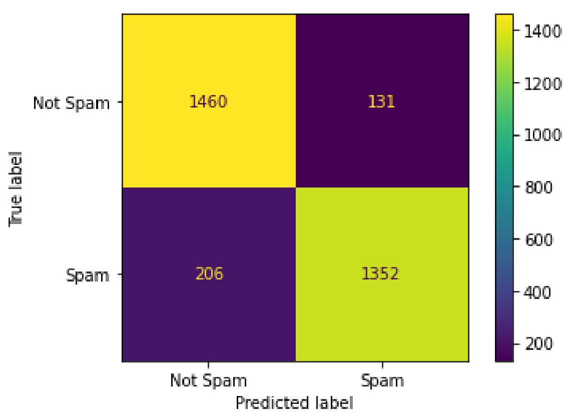
K-Nearest Neighbors Basic Model Evaluation:

Confusion Matrix	Accuracy	0.743
	Precision	0.746
	Recall	0.827
	Macro F1	0.727
	Optimal Hyperparameters:	
	TF-IDF sublinear_tf	True
	KNN n_neighbors	5
	KNN weights	'distance'

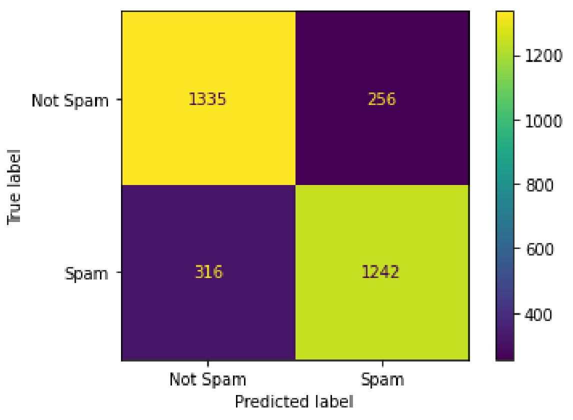
Support Vector Machine Basic Model Evaluation:

Confusion Matrix	Accuracy	0.913
	Precision	0.913
	Recall	0.913
	Macro F1	0.913
	Optimal Hyperparameters:	
	TF-IDF sublinear_tf	True
	SVM C	10
	SVM kernel	'rbf'

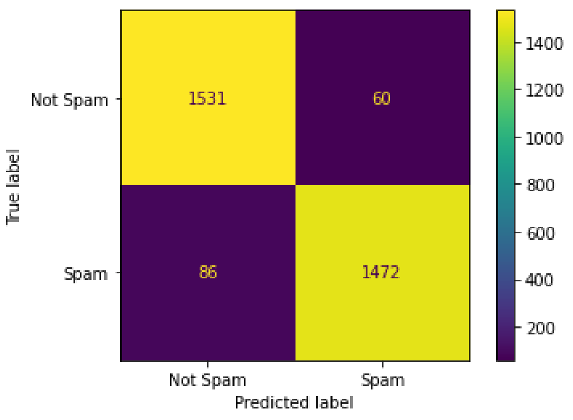
Multinomial Naïve Bayes Basic Model Evaluation:

Confusion Matrix		
	**Accuracy**	0.893
	Precision	0.893
	Recall	0.894
	Macro F1	0.893
	Optimal Hyperparameters:	
	TF-IDF sublinear_tf	True
	MNB fit_prior	True
	MNB alpha	0.1

Decision Tree Basic Model Evaluation:

Confusion Matrix		
	**Accuracy**	0.797
	Precision	0.797
	Recall	0.798
	Macro F1	0.797
	Optimal Hyperparameters:	
	TF-IDF sublinear_tf	False
	DT criterion	‘gini’
	DT max_features	‘auto’

Advanced Model Evaluation:

Confusion Matrix		
	**Accuracy**	0.954
	Precision	0.954
	Recall	0.954
	Macro F1	0.954
	Optimal Hyperparameters:	
	TF-IDF sublinear_tf	True
	SVM C	10
	SVM kernel	‘rbf’

References

[1] Roth, Yoel; Harvey, Del;, "How Twitter is fighting spam and malicious automation," Twitter, 26 June 2018. [Online]. Available: https://blog.twitter.com/en_us/topics/company/2018/how-twitter-is-fighting-spam-and-malicious-automation. [Accessed 15 October 2021].

[2] Twitter, "Platform Manipulation," Twitter, December 2020. [Online]. Available: https://transparency.twitter.com/en/reports/platform-manipulation.html#2020-jul-dec. [Accessed 15 October 2021].

[3] Hinesley, Kara;, "A reminder about spammy behaviour and platform manipulation on Twitter," Twitter, 11 February 2019. [Online]. Available: https://blog.twitter.com/en_au/topics/company/2019/Spamm-ybehaviour-and-platform-manipulation-on-Twitter. [Accessed 15 October 2021].

[4] "Platform manipulation and spam policy," Twitter, September 2020. [Online]. Available: https://help.twitter.com/en/rules-and-policies/platform-manipulation. [Accessed 15 October 2021].

[5] D. Wang, S. B. Navathe, L. Liu, D. Irani, A. Tamersoy and C. Pu, "Click traffic analysis of short URL spam on Twitter," 23 October 2013. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6679991. [Accessed 15 October 2021].

[6] K. Lee, B. Eoff and J. Caverlee, "Seven Months with the Devils: A Long-Term Study of Content Polluters on Twitter," 2011. [Online]. Available: https://www.researchgate.net/publication/221297999_Seven_Months_with_the_Devils_A_Long-Term_Study_of_Content_Polluters_on_Twitter. [Accessed 15 October 2021].

[7] K. Robinson and V. Mago, "Birds of prey: identifying lexical irregularities in spam on Twitter," 11 December 2018. [Online]. Available: https://link.springer.com/article/10.1007/s11276-018-01900-9. [Accessed 16 October 2021].

[8] J. Roesslein, "Tweepy Twitter API," [Online]. Available: https://docs.tweepy.org/en/stable/api.html. [Accessed 17 October 2021].

[9] "About the Twitter API," Twitter, [Online]. Available: https://developer.twitter.com/en/docs/twitter-api/getting-started/about-twitter-api. [Accessed 17 October 2021].

[10] "sklearn.linear_model.LogisticRegression," scikit-learn, [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html. [Accessed 16 October 2021].

[11] "sklearn.neighbors.KNeighborsClassifier," scikit-learn, [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html. [Accessed 16 October 2021].

[12] "sklearn.svm.SVC," scikit-learn, [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html. [Accessed 16 October 2021].

[13] "sklearn.naive_bayes.MultinomialNB," scikit-learn, [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html#sklearn.naive_bayes.MultinomialNB. [Accessed 16 October 2021].

[14] "sklearn.tree.DecisionTreeClassifier," scikit-learn, [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html. [Accessed 16 October 2021].

[15] "sklearn.feature_extraction.text.TfidfVectorizer," scikit-learn, [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html. [Accessed 16 October 2021].

[16] "Metrics and scoring: quantifying the quality of predictions," scikit-learn, [Online]. Available: https://scikit-learn.org/stable/modules/model_evaluation.html. [Accessed 20 October 2021].

[17] "sklearn.model_selection.GridSearchCV," scikit-learn, [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html. [Accessed 16 October 2021].

[18] "SpaCy Tokenizer," Explosion, [Online]. Available: https://spacy.io/api/tokenizer. [Accessed 17 October 2021].

[19] "VirusTotal API," VirusTotal, [Online]. Available: https://support.virustotal.com/hc/en-us/articles/115002100149-API. [Accessed 19 October 2021].

[20] "Plotly Pie Charts," plotly Graphing Libraries, [Online]. Available: https://plotly.com/python/pie-charts/. [Accessed 21 October 2021].

[21] "infolab," Texas A&M University, 2011. [Online]. Available: http://infolab.tamu.edu/data/. [Accessed 16 October 2021].

[22] "VirusTotal Contributors," VirusTotal, [Online]. Available: https://support.virustotal.com/hc/en-us/articles/115002146809-Contributors. [Accessed 17 October 2021].