To Spamalot or not to Spamalot?

Joshua Wolfe
College of Engineering & Computer Science
Syracuse University
Syracuse, New York 13244, USA
jbwolfe@syr.edu

Anthony Redamonti
College of Engineering & Computer Science
Syracuse University
Syracuse, New York 13244, USA
aeredamo@syr.edu

Dana Dippery
College of Engineering & Computer Science
Syracuse University
Syracuse, New York 13244, USA
ddippery@syr.edu

Hal Baird
College of Engineering & Computer Science
Syracuse University
Syracuse, New York 13244, USA
hdbaird@syr.edu

Abstract - As cellular telephones have become ubiquitous, hackers have adapted new techniques to phish users through SMS messaging. Hackers have recently increased their focus on attacking unsuspecting cellular telephone users through SMS messaging, as evidenced by several high-profile breaches in which SMS messaging was used as the primary attack vector. Team Spamalot proposes to counter this emerging threat by using open-source tools to create an SMS spam classifier capable of identifying and removing spam messages. As described in much more detail below, Team Spamalot has proven that such an effective SMS spam filter can be built using only open-source tools. Moreover, the final section of this paper discusses potential avenues for improvement of the spam filter.

Index Terms – Computer Science, Machine Learning, Artificial Intelligence, Natural Language Processing, Word Lemmatizing, Word Stemming.

I. INTRODUCTION

As cellular telephones have become ubiquitous and necessary for many people's work, family, and social obligations in society today, hackers have adapted new techniques to phish users through SMS messaging. Evidence of this phenomenon is the tenfold increase in phishing scams from April 2021 to the April 2022. [1] High profile breaches, such as Twilio, have originated from SMS vectors. [20] Moreover, because many users of cellular telephones remain unaware that SMS text messaging is an active attack vector, hackers have recently conducted successful phishing campaigns to breach high-profile targets. [2]

In this paper, Team Spamalot will summarize its results to create an SMS spam classifier. We used publicly available SMS spam/ham data from the University of California Irvine Machine Learning Repository and other sources (www.kaggle.com) which was pre-labelled. [3] We demonstrate that an effective SMS spam filter can be built from open-source tools known by most data science students. We conclude with future improvements and follow-up enhancements for our spam classifier.

II. TERMINOLOGY

This section details the key terms, beyond the scope of classroom lecture, required to understand how our classifier was constructed.

Computer Science

Computer Science is a branch of science that deals with the theory of computation or the design of computers. [4]

Artificial Intelligence

Artificial Intelligence (hereinafter "AI") is a subset of the field of Computer Science that has been applied to an ever-increasing number of applications in society today. In general, AI enables computers to solve problems ordinarily handled by biological systems. Natural Language Processing (hereinafter "NLP") and Machine Learning (hereinafter "ML") are overlapping subsets of AI. [6]

Machine Learning

Machine Learning enables systems to automatically learn and improve from experience without being explicitly programmed. Machine Learning can be applied to solving AI problems and to improving the quality of NLP by automating processes and delivering accurate responses. [6]

Natural Language Processing

Natural Language Processing (NLP) enables machines to read and understand human language and can be accomplished in conjunction with Machine Learning whose goal is to extract insights from human language, in either written or spoken form. With NLP, machines can make sense of written or spoken text and perform tasks including speech recognition, sentiment analysis, and automatic text summarization. [6] NLP helps organizations to analyze data, discover insights, automate time-consuming processes and/or gain competitive advantages. Programs that interpret a human's voice commands, such as Amazon's Alexa are examples of NLP. [5] Other examples, including this paper's spam classifier, include:

Automated language translators

Translating languages is more complex than making a word-to-word replacement method. Since each language has grammar rules, the challenge of translating languages is to do so without changing meaning and style. Since computers do not understand grammar, a process through which sentences can be deconstructed and reconstructed in the target language in a way that makes sense is necessary. [6]

Grammar checkers

Automatic grammar checking, the task of detecting and correcting grammatical errors and spelling mistakes in text depending on context, is another major part of NLP. Automatic Grammar Checking will alert a human being to a possible error by underlining the word in red. [6]

Chatbots

Chatbots are programs used to provide automated answers to common customer queries. Chatbots use pattern recognition systems to provide heuristic responses, which enables them to have conversations with humans. Initially, chatbots were used to answer basic questions to alleviate heavy volume call centers and offer quick customer support services. But AI-powered chatbots are increasingly able to properly respond to more complex requests making conversational experiences increasingly human-like. [6]

Speech recognition and Automated social media content moderators

Speech recognition is a machine's ability to identify and interpret phrases and words from spoken language and convert them into a machine-readable format. In speech recognition, NLP enables computers to simulate human interaction, and Machine Learning to respond in a way that mimics human responses. [6]

Sentiment Analysis

Sentiment analysis uses NLP and Machine Learning to interpret and analyze emotions in subjective data like news articles and tweets. Positive, negative, and neutral opinions can be identified to determine a customer's sentiment toward the subject matter of processed language. Example uses of sentiment analysis include measuring public opinion, brand reputation, and customer experiences. [6]

Question/answer systems

Question-answer systems are intelligent systems that are used to provide answers to customer queries. Other than chatbots, question-answer systems have a huge array of knowledge and good language understanding rather than canned answers. They can answer questions like “When was Abraham Lincoln assassinated?” or “How do I get to the airport?” and can be created to deal with textual data, audio, images, and videos. [6]

Market intelligence

Market Intelligence is the gathering of valuable insights surrounding trends, consumers, products, and competitors to extract actionable information that can be used for strategic decision-making. Market Intelligence can analyze topics, sentiment, keywords, and intent in unstructured data and is less time consuming than traditional desk research. [6]

Automatic text classification

Automatic text classification is another fundamental solution of NLP. It is the process of assigning tags to text according to its content and semantics which allows for rapid, easy retrieval of information in the search phase. This NLP application can differentiate spam from non-spam based on its content. [6]

Word Stemming

In the context of NLP, word stemming is the process of reducing words in a document down to their stems, or a word equal (or an approximation to) their morphological root. [7] The morphological root would the underived/non-prefixed/non-postfixed/neutral tense/etc. base version of the word. [6] For example, the following phrase: “I went to many stores and bought milky way candy bars.” Would get stemmed to: “I go to many store and buy milk way candy bar.”

This process simplifies the task of learning several words for *go*; e.g. go, goes, going, went, etc. When word stemming is down in the pre-processing step, the NLP learning system now only needs to understand the word “go.” [7] Obviously, there are significant savings when all words are stemmed accordingly. [8]

However, stemming does lead to some loss in context. In the above example, the past tense is lost and the plurality of the candy bars. Depending on the task, these contextual facts may or may not be important. [8]

Word Lemmatizing

Another issue with word stemming is confusion over word context and part of speech. Consider the following example, “Jack saw a tree.”

Most English readers would agree that Jack’s visually located a tree. But a word stemmer might get confused interrupt this Jack cut a tree with a blade instrument known as a saw. Who’s correct?

Lemmatization better derives context for words in a phrase and then stems accordingly. A key step in the lemmatizing process is to determine a word’s part of speech: noun, verb, adjective, etc. By determining a word’s part of speech, it is more likely the word will be properly stemmed. [8] For example, “Your bandaging is loose.”

By lemmatizing first, we determine that bandaging is a noun and would not need to be further stemmed, as opposed to the verb form, which would be stemmed to bandage.
Actual lemmatizers reduce words beyond their Webster dictionary stem word forms. This is due in part because it would be costly and impractical for lemmatizers to maintain all known words in all human languages. Hence, word stem approximations are often made for efficiency purposes. [8]

Word stemmers and lemmatizers are usually packaged together in a single pre-processing step. The following are the most commonly used: 1) Lovins Stemmer, 2) Port Stemmer and Paice Stemmer. [8]

Stop Words

Stop words are words removed from text in an NLP pre-processing step because they do not contribute significantly to extracting insights. Typical examples of these words are the article words: a, an, the

For some domains, the stop words may be specific. For example, search engines may include stop words [the, is, at, which, on] which confuses searches for topics (popular bands) like "The Who" and "The The". Search engines may also put "want" in their stop list simply because it is over-used and contributes no meaning in search queries. [9]

By removing these words, NLP processing resources can be saved for processing the words that will likely contribute insights.

Term Frequency-Inverse Document Frequency Vectorization

A commonly used analysis technique to distinguish how relevant a key word(s) is in a document is Term Frequency-Inverse Document Frequency (TF-IDF) vectorization. TF-IDF analyzes how unique a word is to document relative to a corpus of related documents. [10]

The TF-IDF statistic can be understood in two stages. Consider a word that appears in document with a given a frequency. If this frequency is considered high, relative to other words, then it may be a uniquely relevant term to this document, e.g., the frequency of "cardiac" in a medical article on heart disease. This would yield a high TF statistic. [10]

The frequency of a commonly used English word, such "and" may be high, but it would also yield a high TF statistic in the same article. However, the word "and" would also yield high TF statistics in many other articles, unrelated to heart disease, and so it would be reduced by its "IDF" statistic. Thus, the word "cardiac" would stand out with the higher TF-IDF score and be analysed as a relevant term for the medical article. [10]

This helps to show that order of pre-processing steps may be important, i.e., pre-processing stop words to reduce the burden of TF-IDF processing. Such optimizations are beyond the scope of this project.

III. Exploratory Data Analysis

Data Preparation

Before performing Exploratory Data Analysis (EDA), data quality issues in the dataset were addressed first. This involved using various data pre-processing techniques in order to prepare the dataset for Machine Learning (ML) algorithms used in later steps. Firstly, the non-numeric data types were encoded to numeric values. A "LabelEncoder" in the SciKit-Learn library was used to transform the label value that indicated if a message was Spam or not to either a 1 or 0, respectively. Next, SMS messages could not be handled as simple. Instead, Natural Language Processing (NLP) techniques were used in later steps to prepare the data. After data type issues were resolved, data quality issues were addressed. The dataset contained a small margin of duplicate SMS messages which were removed resulting in the dataset shrinking less than 5%.

Data Splitting

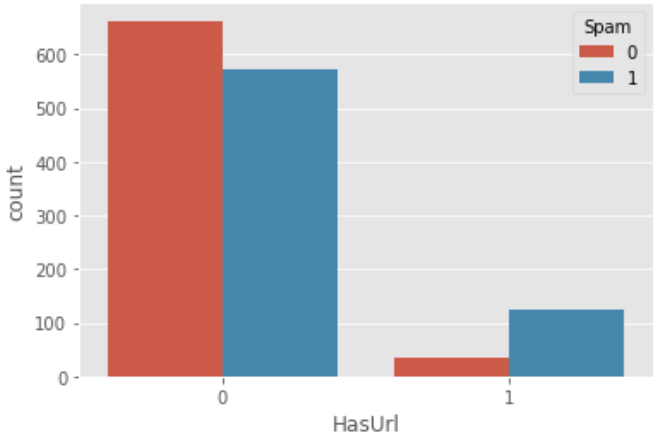
After the dataset was prepared for EDA, it was reviewed for distribution of spam to ham. Ideally, this should be evenly distrusted across the dataset, but instead the dataset was 88.6% ham and 11.4% spam. To fix this, a subset of the dataset was used for EDA that was equally distrusted between ham and spam.

Data Exploration

Spam, whether over SMS or any other means of communication, typically has an intent associated with it. This could be in the form of advertisements for marketing purposes or pretext for malicious scams. This intent, though, usually has one key goal which is to coax the receiver to perform some

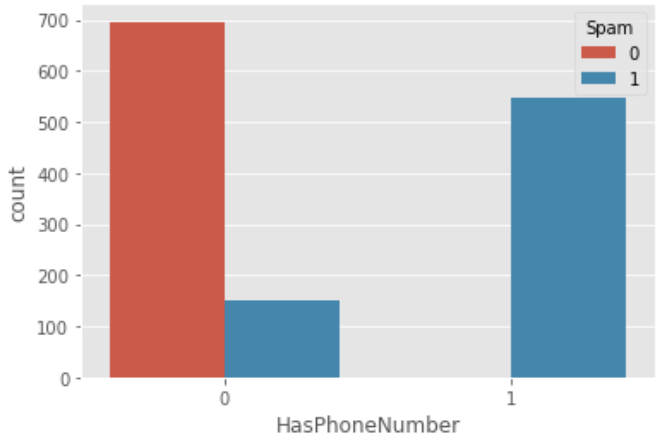
action, such as visit a website, call a number, text a phrase to a short-code number, etc.

Intuition lead our research to hypothesise that website addresses would be the most common method of coaxing receivers to perform an action. After further review, however, our assumption was incorrect. URLs only appeared in 11.2% of the dataset and of that subset, only 78% was spam which means only 8% of the entire dataset was spam with URLs.

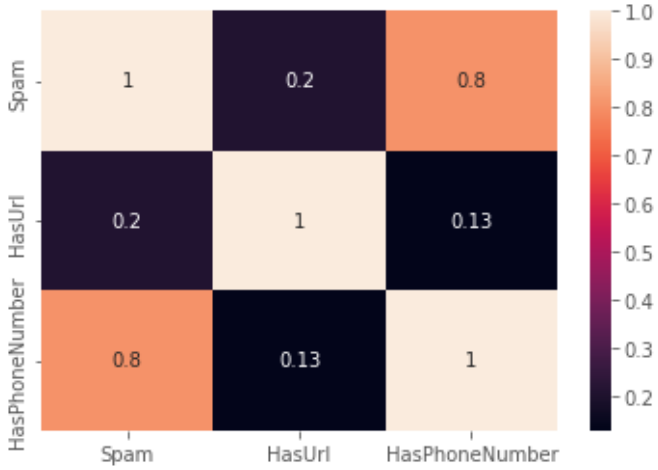


Further investigation into the dataset revealed a previously unknown method of coaxing spam receivers into performing some action. Telephone numbers and 5 digit shortcodes were commonly found accompanied with advertisements and scams. To better understand the distribution of this method, similar grouping techniques previously used for URLs was performed using pandas and regular expressions (RegEx). The RegEx pattern used for mining this feature from SMS messages matched various formats of telephone numbers, including thoses with and without area code, along with 5 digit shortcodes which is typically used in text for a variety of services. All of these forms of telephone numbers matched matched only North American Number Plan (NANP) telephone numbers.

Telephone numbers appeared in 39.3% of the dataset and of that subset 100% was spam. The remaining 60.7% of the dataset that did not contain telephone numbers was made up of 17.6% spam. This meant that telephone numbers could be used as a high confidence key indicator to determine whether or not a SMS message was spam.



The heat-map of these findings correlated to each other and the spam label show that using URL as a key indicator of spam doesn't yield high confidence, but using telephone numbers does.



Natural Language Processing

Using various NLP techniques, the dataset can be further explored for key words that highly correlate to spam messages. Before generating these findings, the messages must first be processed using methods such as tokenization, removing stop words, word stemming, and lemmatization. NLP feature engineering can be conducted after this to find word counts, frequencies, bag of words, TF-IDF, and word embedding.

To begin NLP feature engineering, a vocabulary was established form the set cleaned set of tokens. This represents all the words that appear in SMS messages and their frequency as they appear in the dataset. Once all the words are counted, extracting a meaningful spam vocabulary is computed simply by taking the most frequent words that appear in spam. The frequency alone cannot reveal much insights apart from common words. To enhance these findings, a bag of words is establishes which represents a combination of the frequency scores of with the vocabulary. Scoring SMS messages by the frequency of words ignores uncommon words that may actually contain more relevant meaning. A more sophisticated approach that builds on this is using the Term Frequency over Inverse Document Frequency, or TF-IDF. This score takes the frequency of a work in a message and divides it by the inverse frequency of that word as it appears in all the messages. Using this method of scoring the vocabulary, the most common words in spam SMS messages can be derived. The top 5 words in spam were, “free”, “text”, “ur”, “mobile”, and “claim”.

In order to measure the similarity between 2 SMS messages, the distance in TF-IDF document vectors was measured. This would measure the distance between n dimensional vectors using cosine distance, which calculates the dot product of the two vectors and divides them by the product of their lengths. This will range from 1.0 (exact) to -1.0 (exact opposite) no matter what vectors are used.

Word similarity can be measured in a similar manner by using a concept called word embedding. This representation is a vector of words learned from either a statistical or deep learning model, of which we used Word2Vec from gensim. Message lists of words were fed into the model and which it learns a vector representation of these words. This can then be used to measure the similarity between words in the entire dataset, which creates a model that is called a Continuous Bag of Words (CBOW).

Pairs of words or bigrams can be used in the previous methods to create better context and understanding of SMS messages. We created n-Gram bag of words vector by extending the previous model to use bigrams instead of single words. This showed that the most common phrases in spam SMS message all centred around a pretext of monetary value, such as “won prize”, “await collection”, “chance win”, “prize guaranteed”, etc.

Using these models, we were able to establish a text classification workflow that consisted of various steps. First, data was broken down into tokens. Then the tokens are pre-processed and cleaned by removing stop words, punctuation, stemming, and lemmatization. Next feature engineering is performed, where SMS messages were broken down into Bag of Words, TF-IDF vectors, and Word Embeddings. From there, models were built and trained on the feature engineering results which were then evaluated by examining their performance.

IV. Model Training and Testing

Model Construction

The goal is to create a model that inputs an SMS message and outputs a binary prediction that corresponds to the likelihood that the message is spam. If the prediction is positive, then the message would be predicted as spam and either quarantined from the user’s SMS inbox or delivered with a warning, so the user is on alert.

Data Set Description

	A	B
1	ham	Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine
2	ham	Ok lar... Joking wif u oni...
3	spam	Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 8712
4	ham	U dun say so early hor... U c already then say...
5	ham	Nah I don't think he goes to usf, he lives around here though
6	spam	FreeMsg Hey there darling it's been 3 week's now and no word back! I'd like some f
7	ham	Even my brother is not like to speak with me. They treat me like aids patent.
8	ham	As per your request 'Melle Melle (Oru Minnaminunginte Nurungu Vettam)' has been
9	spam	WINNER!! As a valued network customer you have been selected to receivea £900
10	spam	Had your mobile 11 months or more? U R entitled to Update to the latest colour m

The training data consists of two attributes, the message, and the labelled output of “spam” or “ham.” The output of “ham” refers to a non-spam message, a message which should be delivered to the user without warning.

The dataset is derived from several sources, shown below.

Source	Link	Count
UCI	https://archive.ics.uci.edu/ml/machine-learning-databases/00228/	5574
Kaggle1 “spamraw.csv”	https://www.kaggle.com/datasets/shravan3273/sms-spam	5156
Kaggle2 “sms_spam.csv”	https://www.kaggle.com/datasets/hdza1991/sms-spam	5160
Kaggle3 “smsspamcollection.csv”	https://www.kaggle.com/datasets/shrutipandit707/smsspamcollection	5168
Kaggle4 “SMSCollection.csv”	https://www.kaggle.com/datasets/arunasivapragasam/spam-or-ham	5169

While the above sources have many entries per source, very few of them are unique.

Source	Unique Entries
UCI	5158
Kaggle1 “spamraw.csv”	645
Kaggle2 “sms_spam.csv”	80
Kaggle3 “smsspamcollection.csv”	36
Kaggle4 “SMSCollection.csv”	1

The total number of unique entries is 5,920.

Number of HAM Entries	5255
Number of SPAM Entries	665
Total Number of Unique Entries	5920

All messages in the data set are in the English language. The country/region of origin is not specified.

Data Pre-processing

The following steps have been taken to pre-process the SMS messages prior to model training and testing:

- All words in the messages have been stemmed and lemmatized using the PortStemmer package from the NLTK Python library
- All English “stop” words have been removed
- All messages have been TF-IDF vectorized using the TfidfVectorizer from the Scikit-Learn Python Library

Model Scoring

The models are scored, based on the model test data set, in two dimensions, based their accuracy with the respect to predicting spam messages.

First, the models will be selected based on the highest precision score:

> Precision = (# true positives) / ((# true positives) + (# false positives)

The precision score was chosen because it maximizes models that identify spam messages accurately without misidentifying non-spam messages. Models are rewarded that correctly identify spam messages. A system that fails to deliver non-spam messages (i.e., ham messages classified as spam) to users would be considered flawed and unusable.

If precision scores are equal between model candidates, then recall, with respect to spam prediction, will be used to select the best performing models. The recall score describes the model that best identifies the maximum number spam messages in the data set. A combination of the spam precision and recall score, F1, allows the optimal selection criteria for the secondary level of model selection.

V. MODEL PERFORMANCE & RESULTS

As previously mentioned, all messages were TF-IDF vectorized. The vectorizer assigns weight to each word, known as the IDF weight. The IDF weight was an integral feature used in model training and testing as it is used to find words that appear with a high frequency in SPAM messages but not in HAM messages. The words with the lowest IDF weights are below and appear the most in SPAM and not in HAM.

Spam Word	IDF Weight
free	2.48
txt	2.53
ur	2.89
claim	2.91
www	2.94

The following models were trained using the data set described above. The data set was split into a 70% training set and 30% testing set.

Model Type	Sci-kit Learn Library Used
Naïve Bayes (Gaussian)	naive_bayes.GaussianNB
Naïve Bayes (Bernoulli)	naive_bayes.BernoulliNB
Random Forest	ensemble. RandomForestClassifier
K-Means Nearest Neighbor	neighbors. KNeighborsClassifier
Gradient Boosting	ensemble. GradientBoostingClassifier

These models exhibited the following performance characteristics, specific to the spam classification scores:

Model Type	Precision	Recall	F1
Naïve Bayes (Gaussian)	0.92	0.89	0.91
Gradient Boosting	0.95	0.77	0.85
K Nearest Neighbor	1.00	0.31	0.47
Naïve Bayes (Bernoulli)	0.99	0.82	0.90
Random Forest	1.00	0.82	0.90

Based on these results and the scoring criteria, the Random Forest based model demonstrates the top performance for identifying spam messages with respect to precision and recall. However, the Naïve Bayes (Bernoulli) model's performance is very close in performance, and further testing may reveal these models perform identically.

VI. IMPROVEMENTS & NEXT STEPS

While this model shows significant promise towards classifying spam SMS messages, there are several limitations that our team noted:

- The data set is small: the number of labeled spam messages is significantly overwhelmed by the number of non-spam messages. Perhaps the spam messages in this data are not diverse or respective of other attack techniques?
- This data is specific to English language spam.
- Data attributes are limited. These public data sets only include textual messages and spam/ham labels. Other message attributes may be insightful, such as timestamps, origination, URL
  meta data, is this a known contact, etc.
- The model output is limited to binary predictions. A future enhancement would be to predict spam based on a probabilistic scale rather than 0 or 1. Such a prediction could be fed into other larger machine learning algorithms that consider other factors.

Bibliography

[1] K. Orred, “2022 SPAM text statistics: Are spam texts on the rise?,” *TEAOrangeWhiteStacked-OL-01-1*, 2022. [Online]. Available: https://www.text-em-all.com/blog/spam-text-statistics. [Accessed: 10-Dec-2022]

[2] 2022 11:33 pm U. T. C. Dan Goodin - Aug 9 and OllieJones Ars Centurion et Subscriptor jump to post, “Phishers who breached Twilio and targeted cloudflare could easily get you, too,” *Ars Technica*, 09-Aug-2022. [Online]. Available: https://arstechnica.com/information-technology/2022/08/phishers-breach-twilio-and-target-cloudflare-using-workers-home-numbers. [Accessed: 10-Dec-2022]

[3] *UCI Machine Learning Repository: Data Sets*. [Online]. Available: https://archive.ics.uci.edu /ml/datasets.php. [Accessed: 10-Dec-2022]

[4] “Computer science definition & meaning,” *Merriam-Webster*. [Online]. Available: https://www.merriam-webster.com/dictionary /computer%20science. [Accessed: 10-Dec-2022]

[5] R.-A. S. Alvarez, “Nombres en -eús y nombres en -us, -u en micénico: Contribución al Estudio del Origen del Sufijo -eús,” *Amazon*, n.d.. [Online]. Available: https://developer.amazon.com/en-US/alexa. [Accessed: 10-Dec-2022]

[6] X. Bolaños, “Natural language processing (NLP) for Machine Learning,” *Encora*, 18-Jul-2022. [Online]. Available: https://www.encora.com /insights/natural-language-processing-and-machine-learning. [Accessed: 10-Dec-2022]

[7] *https://en.wikipedia.org/wiki/Root _(linguistics) root*, 2022. [Online]. Available: https://en.wikipedia.org/wiki/Root_(linguistics) root. [Accessed: 2022]

[8] *Stemming and lemmatization*. [Online] Available: https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html. [Accessed: 10-Dec-2022]

[9] “Stop word,” *Wikipedia*, 09-Nov-2022. [Online]. Available: https://en.wikipedia.org /wiki/Stop_word. [Accessed: 10-Dec-2022]

[10] “TF-IDF,” *Wikipedia*, 12-Sep-2012. [Online]. Available: https://en.wikipedia.org/wiki/tf-idf. [Accessed: 10-Dec-2022]

[11] “Welcome to Python.org,” *Python.org*, 2022. [Online]. Available: https://www.python.org/. [Accessed: 10-Dec-2022]

[12] “Project jupyter,” *Project Jupyter*, 2022. [Online]. Available: https://jupyter.org/. [Accessed: 10-Dec-2022]

[13] *NumPy*, 2022. [Online]. Available: https://numpy.org/. [Accessed: 10-Dec-2022]

[14] “Pandas,” *pandas*, 2022. [Online]. Available: https://pandas.pydata.org/. [Accessed: 10-Dec-2022]

[15] “Visualization with python,” *Matplotlib*, 2022. [Online]. Available: https://matplotlib.org/. [Accessed: 10-Dec-2022]

[16] “Matplotlib application interfaces (apis)#,” *Matplotlib Application Interfaces (APIs) - Matplotlib 3.6.2 documentation*, 2022. [Online]. Available: https://matplotlib.org /stable/users/explain /api_interfaces.html#api-interfaces. [Accessed: 10-Dec-2022]

[17] “Statistical Data Visualization#,” *seaborn*, 2022. [Online]. Available: https://seaborn. pydata.org/. [Accessed: 10-Dec-2022]

[18] “Learn: Machine learning in python - scikit-learn 0.16.1 documentation,” *scikit*, 2022. [Online]. Available: https://scikit-learn.org/. [Accessed: 10-Dec-2022]

[19] *NLTK*, 2022. [Online]. Available: https://www.nltk.org/. [Accessed: 10-Dec-2022]

[20] https://arstechnica.com/information-technology/2022/08/phishers-breach-twilio-and-target-cloudflare-using-workers-home-numbers/