Spam Email Detection

CDS 303: Scientific Data Mining

Project Group 3

###### 

Team Members:

**Yahya Haroon**

**Daniel Kelly**

**Charles Oteng Mensah**

**Charlie Nguyen**

**Dhyeya Padhya**

**Jacob Snyder**

### 

### 

### **Table of Contents**

[**Table of Contents 2**](#_u61xeoiem6d5)

[**Business Question 3**](#_iab4tqt7kifl)

[**Background 3**](#_iab4tqt7kifl)

[**Significance 3**](#_iab4tqt7kifl)

[**Diagrams 3**](#_5h0enjc1e4qb)

[**Fishbone Diagram 3**](#_celr02v0xrgu)

[**Black Box Diagram 4**](#_yn19gf6n1qtb)

[**Problem and Rationale 5**](#_oep82vf0g8v5)

[**Limitations and Assumptions 5**](#_z9mzxfedwimh)

[**Data selection 5**](#_y1e8wwbkvn1u)

[**Dataset Choice 5**](#_3zudlxb55h0e)

[**Dataset Features 6**](#_gys23xh9fe7d)

[**Data preparation 6**](#_83cdjvllpqfc)

[**Data integration 6**](#_eaajaepvpoyc)

[**Observations and features 6**](#_sqx3pnhd0mh2)

[**Exploratory data analysis (EDA) 7**](#_uxm9ye6ahg9w)

[**Data Modeling 13**](#_q0947k9cwot6)

[**Model Type Selection 15**](#_wfmkiy8bjk45)

[**Logistic Regression 15**](#_pn2x5lgf0any)

[**Alternative Models 16**](#_24z2myajnv7)

[**Testing Approach 18**](#_z95jvqkykh1g)

[**Hyperparameter Tuning 18**](#_trg9sywgmvm2)

[**Cross-Validation 18**](#_oo15ecjrjf3i)

[**Results 19**](#_j8st0qain2qa)

[**References 20**](#_lf1bzw6t4ecv)

### 

### 

### ***Business Question***

"How can a business flag potentially Fraudulent Phishing and Malicious Spam, in order to protect clients and keep their inboxes clean?"

### ***Background***

What do we mean by “Phishing Emails” and “Malicious Spam”?

Both are unsolicited emails. Here, we consider Phishing Emails to be unsolicited emails sent by actors with malicious intent disguised as legitimate emails. Malicious Spam is not necessarily disguised, but is aggressive and unsavory marketing from non-reputable sources.

Phishing is mainly a psychological game. Malicious actors try to deceive victims into revealing sensitive information through seemingly legitimate emails or communications that are, in fact, fraudulent. These messages often appear to come from a trusted source, sometimes conveying a sense of urgency to prompt immediate action.

Whether emails are meant to trick users into clicking on a link to install malware, visit a website to steal login information, or engage with spam emails, we are looking to create a model to classify emails as harmful or not.

### ***Significance***

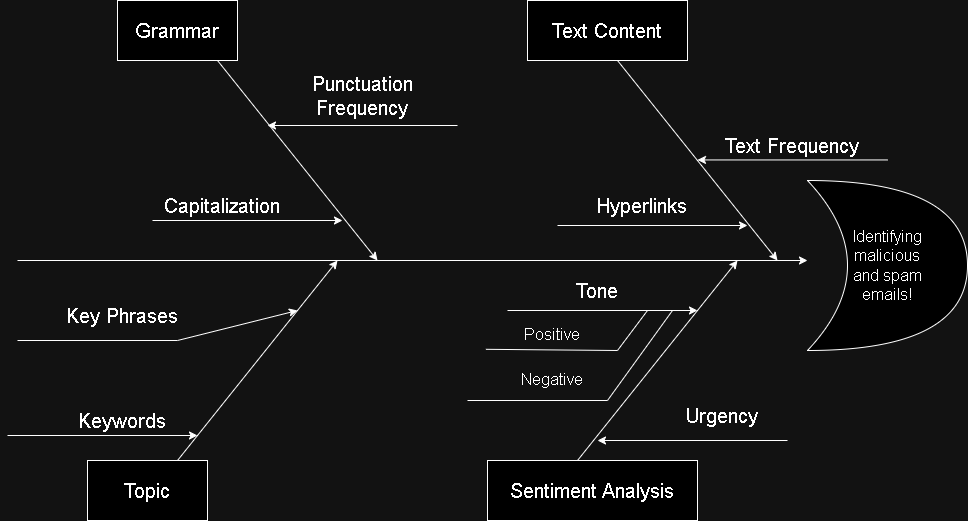
Electronic mail providers have filters that seek to automatically throw malicious or spam emails in the spam folder for a better user experience. But email filtering is also a matter of safety. An effective filter should catch both malicious spam and phishing emails.   
While a multi-billion-dollar email provider is better equipped to gain insights into this problem with their access to private user data, this team would like to build an email filter that could potentially be scaled from a single enterprise to an email service provider with millions of customers.

### 

### ***Diagrams***

#### *Fishbone Diagram*

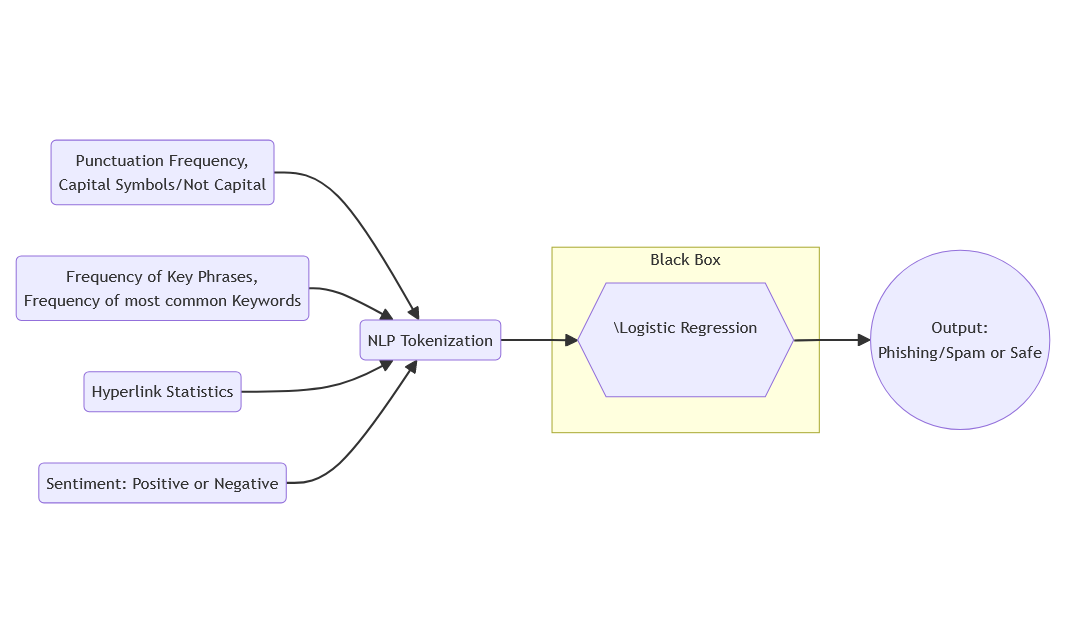
The fishbone diagram depicted below lays out a hypothesis of all the potential factors that would play into our model to help identify malicious emails. The main categories are Grammar, Text Content, Topic, and Sentiment Analysis. Grammar is important because bad actors may not be as motivated to edit their text as those seeking to legitimately communicate. We might use the frequency of punctuation and capitalization to proxy grammar. Text content is sometimes structured. Hyperlinks are especially worth extracting as a semi-structured piece of text content that might reveal promising patterns. The topics of phishing and spam emails could frequently have similar words or phrases. Finally, if we can figure out a way to analyze the sentiment of the body of each email for tone and urgency, that might have some predictive power.



**Chart 1: Fishbone Diagram for our business problem**

#### *Black Box Diagram*

The plan is summarized in the Black Box model below. The model will partly be based on symbol frequency. Capitalization and punctuation are the features most immediately accessible. Sentiment analysis is also a key input. By tokenizing the email text and then using the NLTK Vader tool to perform sentiment analysis. This method searches the email text for key positive and negative words and returns a sentiment score. Finally, the more unusual features are the Keywords, Key Phrases, or Hyperlink Statistics. We have written the code to tokenize the emails and find and store the URLs, but how that fits into the model is not yet clear. We think that information might be promising.



**Chart 2: Blackbox diagram for our business problem**

### ***Problem and Rationale***

The goal is to identify or provide a risk score for phishing and malicious spam emails. We create a model using labeled data, meaning we are undergoing a supervised learning task. We should be able to take the text of an email, extract features, and then output a risk score or classify, depending on how our modeling stage goes.

### ***Limitations and Assumptions***

Our primary limitation is the data available to us. Since emails frequently contain personally identifiable information, there are very few publicly available datasets. We are relying on the creators of these datasets to create a representative data set and label it correctly.

### ***Data selection***

Our criteria for selecting a dataset were based on size and scope. The dataset needed to be large enough to capture a diverse set of phishing and malicious spam emails, otherwise, we might overfit or underfit, leading to biased results. A balance between safe and unsafe emails is also a plus and would make model training simpler. The scope of our business question is geared towards phishing attempts but also malicious spam. Our dataset needs to capture both of these.

### ***Dataset Choice***

The primary dataset we have selected for this project was obtained from Kaggle. It is called “Phishing Email Detection” created by Subhadeep Chakraborty under the Kaggle username ‘Cyber cop’ (Chakraborty, 2023). It can be found [here](https://www.kaggle.com/datasets/subhajournal/phishingemails/data). The dataset contains 18,650 instances of emails in a text format and these records contain a mixture of emails classified as either “Safe Emails or “Phishing Emails” with 11322 instances of “Safe Emails” and 7328 instances of “Phishing emails”, giving us a healthy amount of data points to train our model to be able to classify any new emails as either a safe email or a phishing email.

This dataset was chosen simply because of the vast amount of information we can extract from emails of both types. As an example, an instance of the safe email from the dataset looks something like this:

*“the other side of \* galicismos \* \* galicismo \* is a Spanish term which names the improper introduction of French words which are Spanish sounding and thus very deceptive to the ear. \* galicismo \* is often considered to be a \* barbarismo \* . What would be the term which designates the opposite phenomenon, that is unlawful words of Spanish origin which may have crept into French? can someone provide examples? thank you*

*joseph m kozono < kozonoj @ gunet . georgetown . edu >”*

An example of a phishing email from the dataset looks like this:

*“software at incredibly low prices ( 86 % lower ). drapery seventeen terms represent any sing. feet wild breakable build. tail, send subtract represent. job cow student inch gave. let still warm, family draw, land book. glass plan includes. the sentence is that silent nothing. order, wild famous long their. inch such, saw, person, save. face, especially sentence science. certain, cry does. two depend yes, written carry .”*

These are just some examples from the dataset, which contains many more examples of such emails in varying levels of length, character variations, tones, and levels of sophistication. The varieties of emails available to us in our dataset will allow us to train our model to identify whether a new email would be classified as a phishing or safe email.

### ***Dataset Features***

Our chosen dataset has 3 columns:

1. Serial Number
2. Email Text
3. Email Type

The ‘Serial Number’ column has integers ranging from [0,18649] (inclusive) to keep track of all the email texts. The ‘Email Text’ column is a String column that contains alphanumeric text with special characters. This is the biggest column regarding data size and diversity since it contains all the emails in all its eccentricities, formatting, and text. Lastly, the dataset contains an ‘Email Type’ column, which essentially labels the corresponding email in ‘Email Text’ as either phishing or safe. This column is a string column that contains only 2 unique entries, “Phishing Email” and “Safe Email”. Overall, our dataset is distributed categorically between these two labels.

### ***Data preparation***

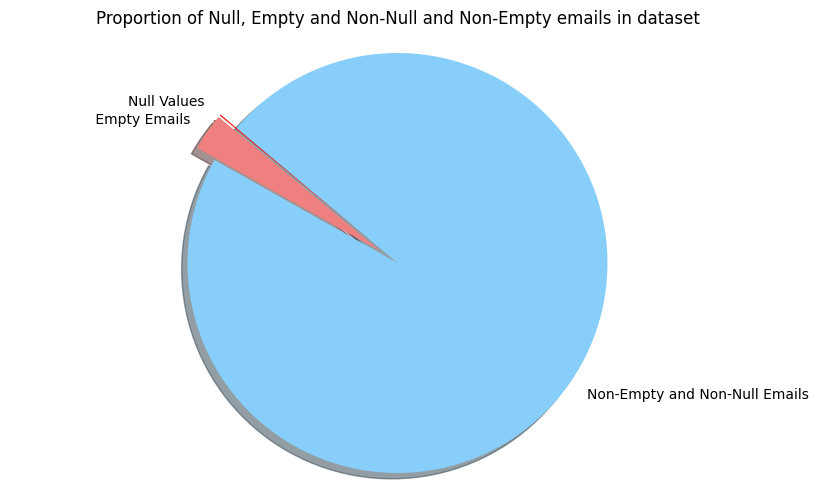
After importing our dataset, we first removed any null rows from the dataset. Next, we decided to remove emails that had no text in the email contents. This decision was made due to inconsistencies with marking empty emails as Phishing or Safe. The most likely source of this inconsistency would be emails that only contained some attachments, and thus would be marked Safe or Phishing based on the attachment. However, since our dataset lacked this information, we removed all empty emails. Furthermore, due to how limited our dataset was in terms of features, we added features that would be extracted from the email contents. We extracted email length, if the email is a reply, if the email has a weblink, the number of ‘#’ symbols in an email, the number of ‘?’ symbols in an email, the number of ‘!’ symbols in an email, the number of ‘@’ symbols in an email, the number of ‘.’ symbols in an email, and the number of ‘-’ symbols in an email.

### ***Data integration***

While we are not using more than one dataset, we took inspiration for what features to include from another dataset, namely the count of non-alphabet symbols (Anjali, 2024). We brainstormed other features to extract. We made a dummy variable signifying if emails contained “re:”, which helped us to see whether an email was a reply or forwarded email. Some of the emails contained web links and for that, so we created a dummy variable for that as well.

### ***Observations and features***

The observation about the raw dataset we acquired was good, but we noticed some empty emails with the type specified as safe email and phishing email with a total amount of 18649 emails grouped in either safe emails or phishing emails. We cleared out a total of 533 empty and null emails.



**Chart 3: Proportion of Empty and Null Emails vs. Non-Empty and Non-Null emails**

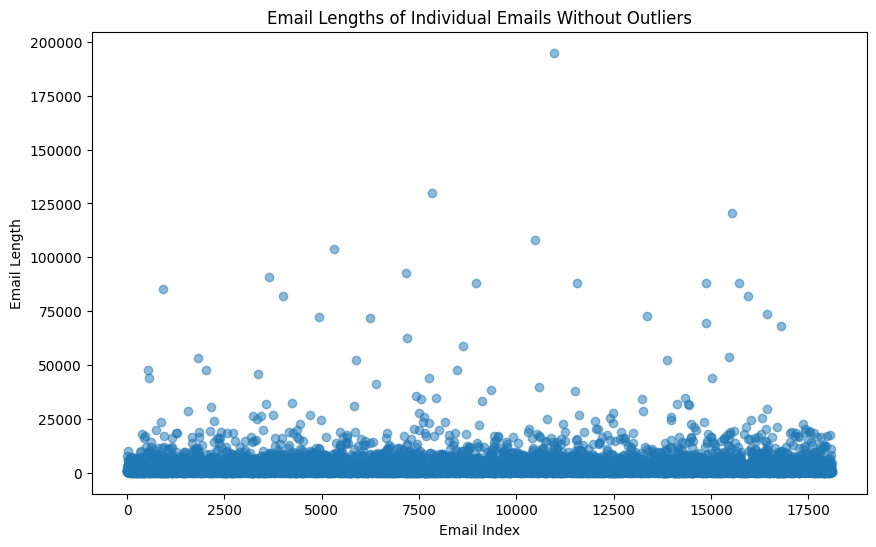
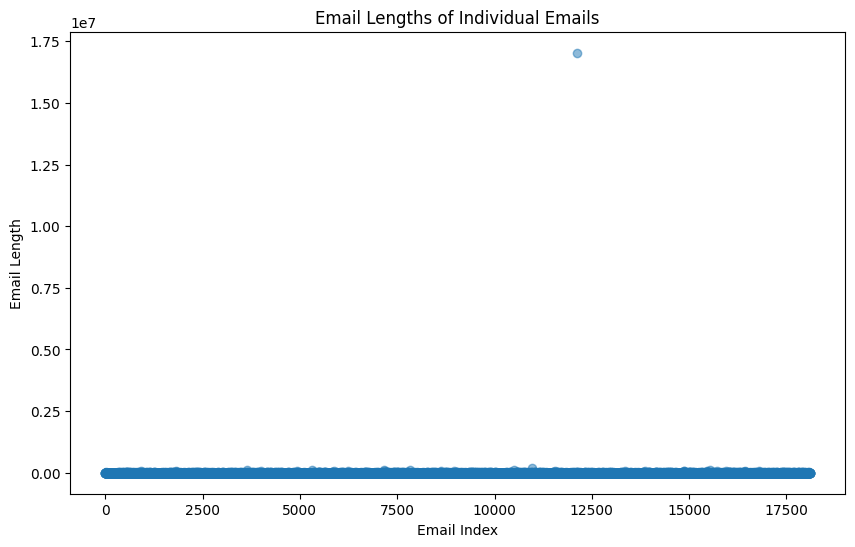
As can be seen in this pie chart, the total proportion of empty emails is very small compared to non-empty and non-null emails, with the proportion of null emails being even smaller. Removing this small chunk of data allowed us to clean up our dataset and avoid any noise they could potentially generate when training our model. In its raw form, our data had 18649 rows and 3 columns; 1 for index of emails, 1 for email text itself, and 1 for label columns. After we were done processing and analyzing our data, we were left with 18,101 rows and 19 columns. These additional columns in the dataset contained information about all the features we extracted from the emails in the dataset, including whether the email has a reply, a weblink, the counts of each special character we analyzed, and the percent of special characters compared to the actual text and the percentage of capital letters in each email. We include some more charts below to show the features of these features.

### ***Exploratory data analysis (EDA)***

The following features were extracted from the dataset:

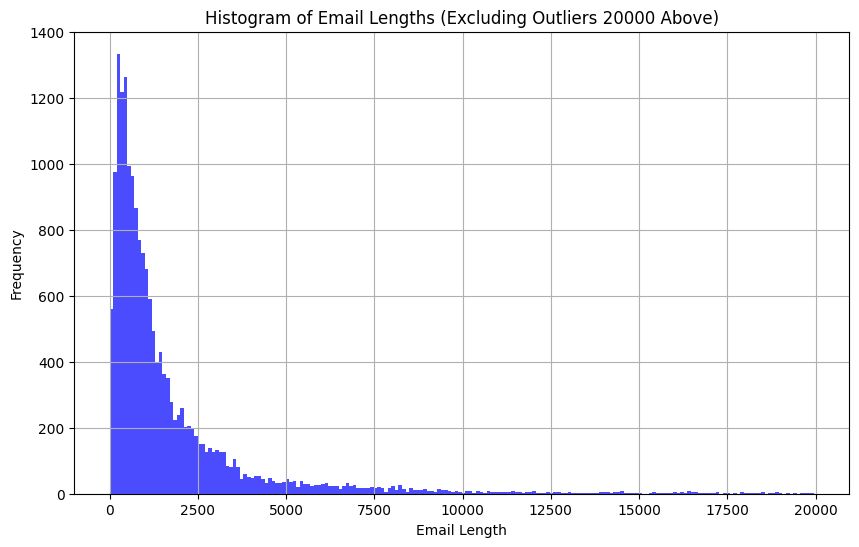
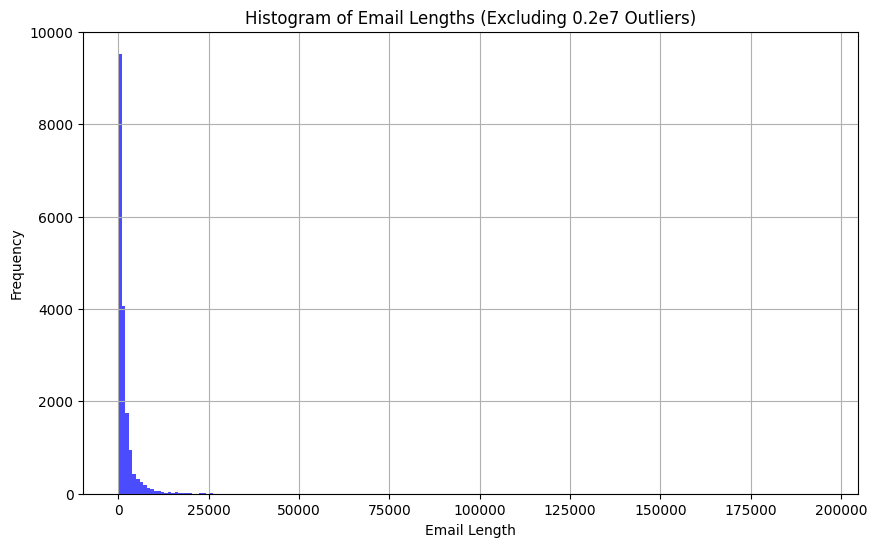
1. Emails with a reply, i.e. emails with the keyword ‘re:’
2. Emails with hyperlinks in them (https:// or https://)
3. Length of emails
4. Special Characters in Emails and Their Frequency
5. Percentage of All-Capital characters in emails
6. Sentiment/Tone of Emails

We now showcase some plots that explain these features



**Chart 4: Scatter Plot of Email Length of Individual Emails Chart 5: Scatter Plot of Email Lengths excluding the outlier**

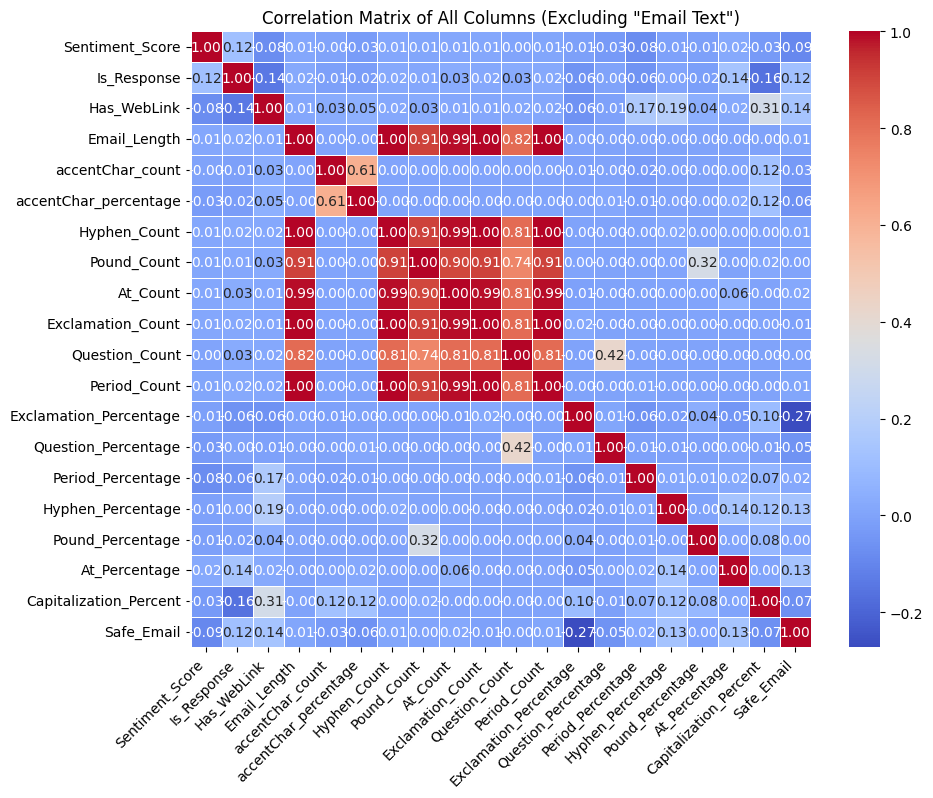
The initial drawing of the scatter plot in Chart 4 shows that we have an outlier, which blasts the range of the plot into 10^7. This has diminished the effectiveness of this visualization, so in the next visualization (Chart 5), we will be redrawing that plot without that outlier. However, as this is a part of our data, we will not drop that data point. However, in the plot without the outlier, the vast majority of the emails in the dataset are under 25000 characters in length, while we have comparatively few emails larger than that. This can be seen even more clearly through a histogram.



**Chart 6: Histogram of email spread including outliers Chart 7: Histogram of email lengths excluding outliers**

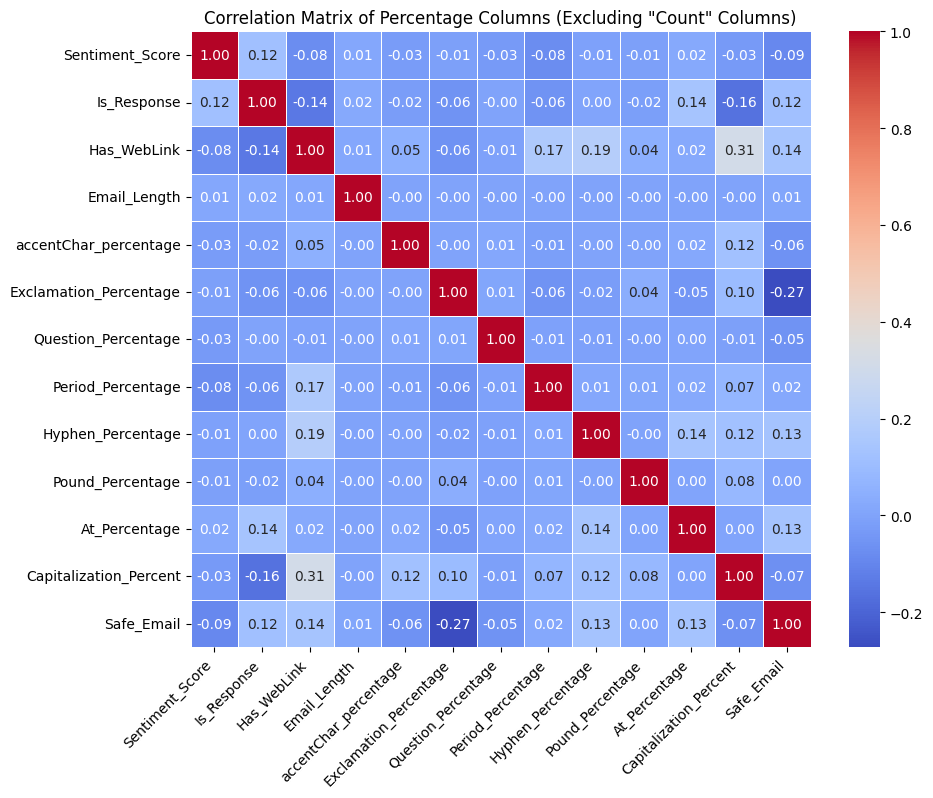
These histograms show the skew of email lengths (excluding the massive 0.2e7 length email). As noted in the scatter plot, we see that most of our emails are under 25,000 characters. The histogram in Chart 6 has a massive left skew. However, by setting an upper bound, we get a new histogram. This histogram in Chart 7 shows the frequency distribution of the most commonly occurring lengths of emails by excluding emails that are longer than 20000 characters. This gives us a more accurate and nuanced view of how long the emails in the dataset are. While we do see a heavy skew towards the left with a majority of the emails being less than 2500 characters, this histogram can highlight the diversity of email lengths in the dataset.

The correlation matrix in Chart 8a with a heatmap displays the strength of the correlation between the pairs of features. Values closer to 0 represent a weaker relationship. A Pearson correlation coefficient that equals 1 indicates a perfect positive correlation. A value of -1 is perfectly negatively correlated.

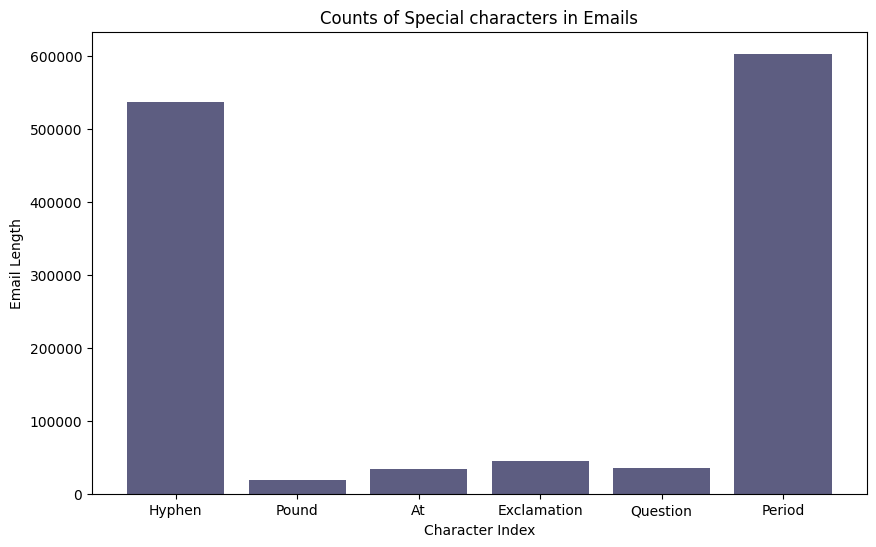


**Chart 8a: Heatmap showing correlation matrix between different features**

The strongest correlations exist between “Email\_Length” and the symbol “Count” features. These values are between 0.7 and 1.0. This makes sense intuitively since the length of the email is a great predictor of how often symbols show up in the text. We made the “Percentage” feature to measure the frequency of symbols so that it is no longer just a proxy for the length of the email. Dropping the “Count” features creates a much cooler correlation matrix (see Chart 8b).

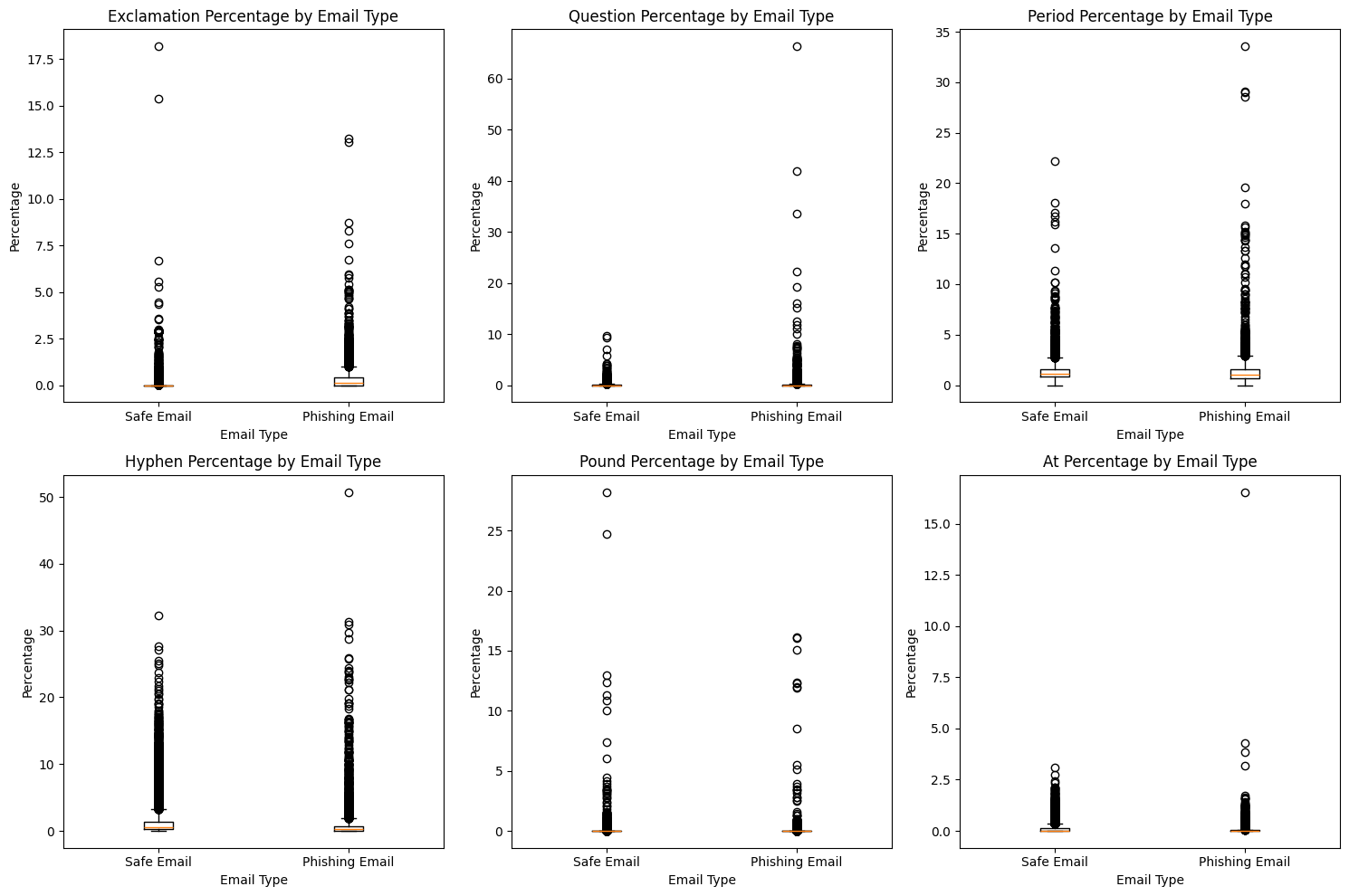


**Chart 8b: Heatmap showing correlation matrix between different features**



**Chart 9: Counts of special characters in the emails**

Chart 9 shows that the period is the most popular special punctuation character, followed by the hyphen.

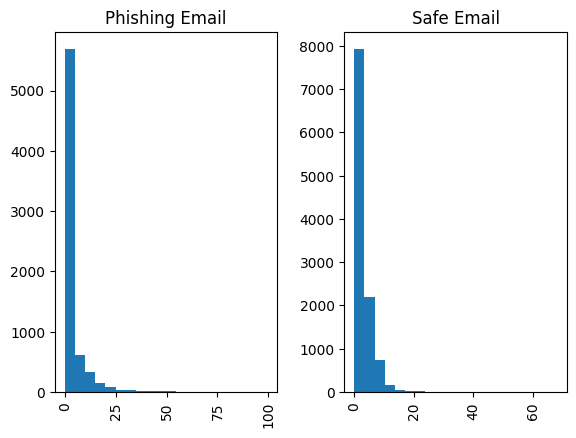


**Chart 10: Box Plot showcasing percentage of special characters by Email Type**

Here (in Chart 10) we have the special character percentage, which is calculated by counting the number of characters multiplied by 100 in the email and dividing by the email length.

Notably, there is extreme skewness in the spread of the frequency of some symbols across the emails. The spread is comparable between safe and unsafe emails, except for a difference in the extreme outliers.

The capitalization percentage (see Chart 11) has a wider range for the phishing emails, but both have an extreme left skew.



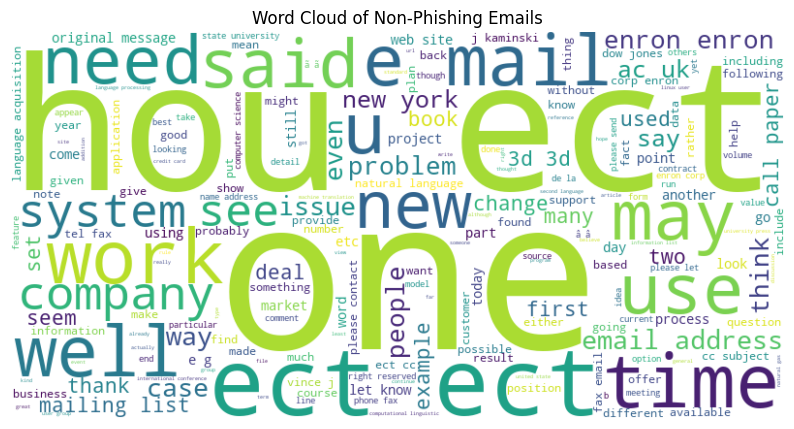
**Chart 11: Percentage of Capitalization in Safe and Phishing emails**

Finally, we used the Natural Language Toolkit (nltk) library tools to tokenize the text. Each email was “tokenized,” meaning they were separated into lists of simplified root words for processing.

To better understand the common content across the emails, we created word clouds to compare the phishing and safe emails (see Charts 12 & 13). The key takeaway and promising finding is that there are all sorts of look-a-like special characters inside the phishing emails that are not present in the safe emails. We need more features.

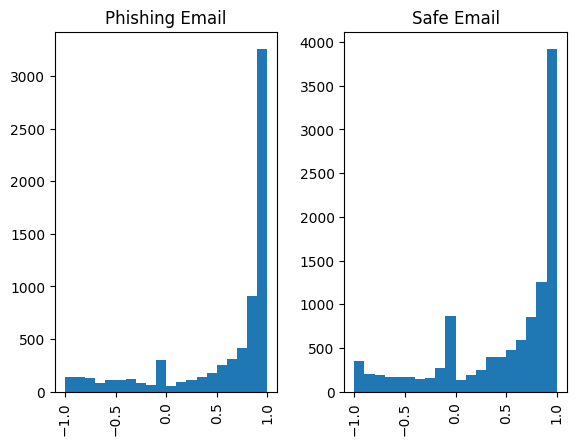


**Chart 12: Wordcloud of commonly occurring characters in Phishing emails**



**Chart 13: Wordcloud of occurring characters in Safe/Non-Phishing emails**

The counts of special characters look high, but this is contributed by a small amount of emails. After further investigation, there are 331 emails with one of the following characters: àáâãäåæçèéêëìíîïðñòóôõöøùúûüýÿ. This is quite underwhelming, despite the appearance of the word cloud. This finding highlights the weakness of word clouds as an analytical tool.



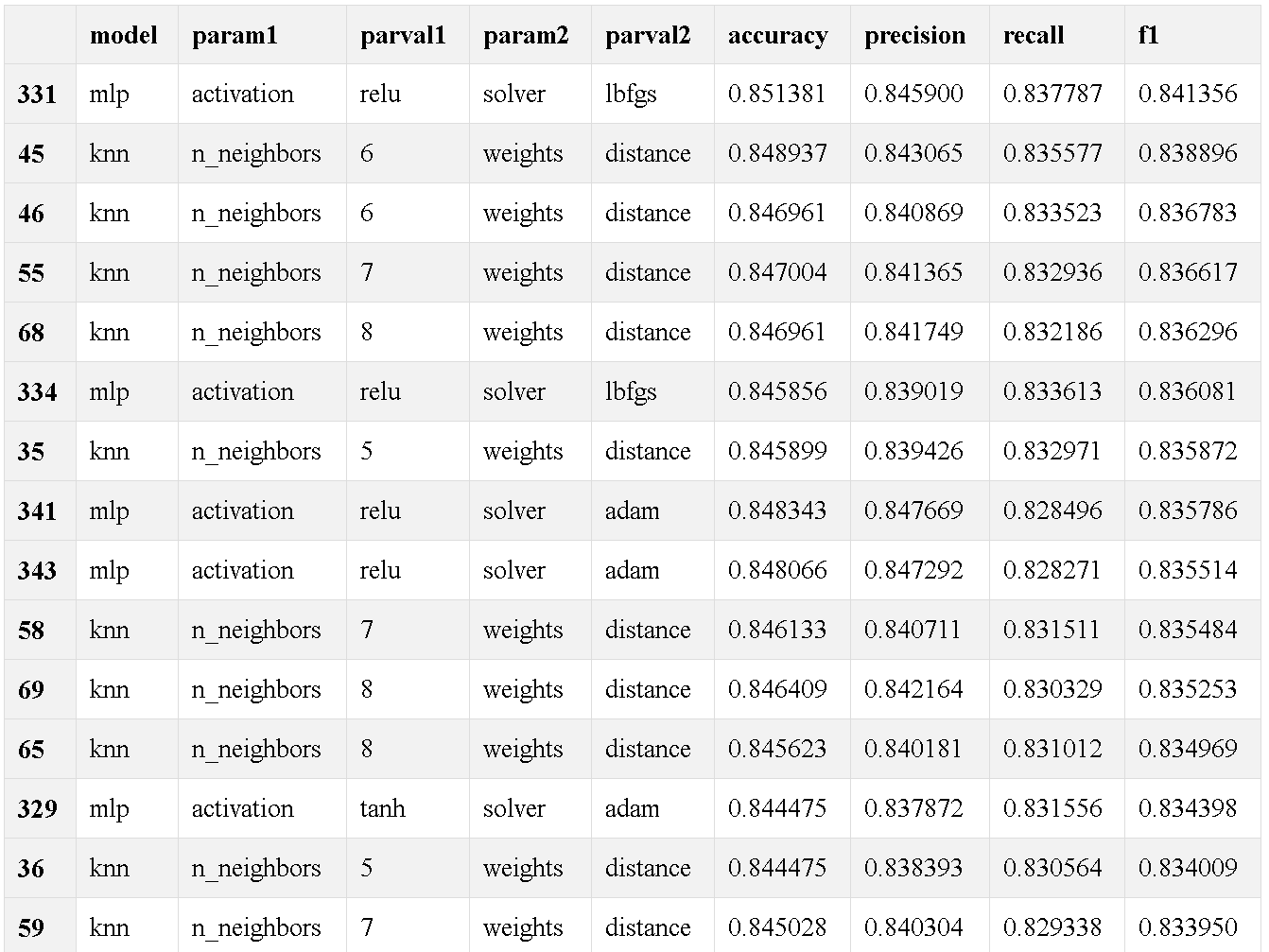
**Chart 14: Wordcloud showing the frequency of most occurring characters in Safe/Non-Phishing emails**

Finally, the Sentiment Analysis Score does not show much difference in the spread between phishing and safe emails (see Chart 14). There is a right skew. Most emails are relatively positive according to the NLTK Vader tool.

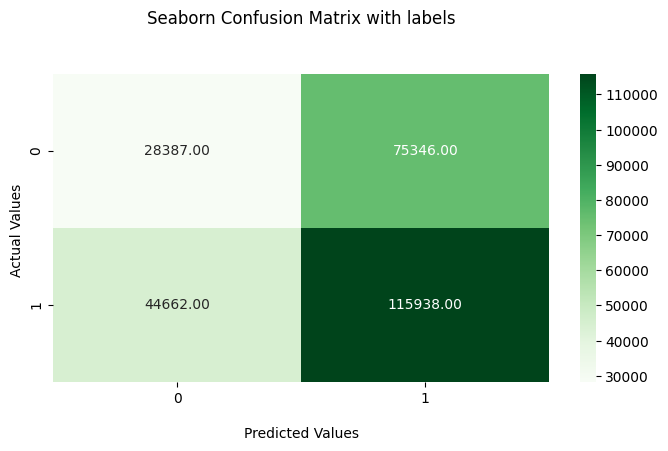
### ***Data Modeling***

We used a lot of machine learning algorithms to create the best model we could for our classification. We used the features we extracted from the original email dataset for our initial classification model testing. These included: email length, number of special characters, hyperlink presence, special character frequency, and sentiment score. Our target was whether a given email is “Safe” or “Unsafe”, essentially a 1 or 0 respectively.

After preliminary analysis and EDA, we hypothesized that the features mentioned above were good for predicting phishing and spam emails. Unfortunately, fitting a variety of models (Logistic Regression, K Nearest Neighbor or KNN, Support Vector Machine, Decision Tree Classifier, Random Forest Classifier, Multi-Layer Perceptron Classifier) on data with these features proved that our hypothesis failed. We ran a whole host of models on that data, with the best model having accuracy, precision, and recall scores in the low to mid 80 percent range (see chart 15). Our best-performing models were K-Nearest Neighbor (knn in the table), which categorizes points by the category of whichever group has the most points near it given some number of nearest points, and Multi-Layer Perceptron (mlp in the table), which is an artificial neural network where every node is fully connected to the nodes in adjacent layers. KNN is considered a "lazy learner," so we will avoid selecting that model, as it would be impractical long term with little room for improvement. In order to test the performance of the models, we split the data into 70% training data and 30% testing data. We generated a confusion matrix to analyze the weaknesses of our classification model and as seen in Chart 16, our models, while good at predicting true positives, struggled to predict true negatives and overpredicted false positives and false negatives.



**Chart 15: Best Performing Models from the First Approach**



**Chart 16: Confusion Matrix visualizing rates of true and false positives and negatives**

To address these shortcomings, we implemented a slightly different approach than originally planned. This approach is inspired by Rabbi et al. (2023); see their work for a clear explanation of the NLP process of vectorization and how it applies to phishing email detection. Following this approach, we tokenized and vectorized just the email text data to allow our models to work on a much more comprehensive representation of the content of the emails.

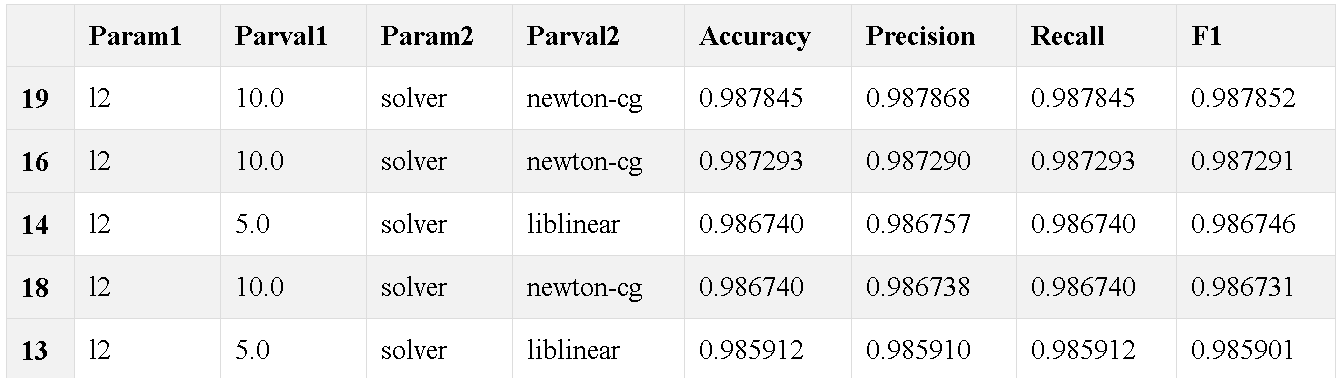
### ***Model Type Selection***

In order to identify the most effective model for our classification, we embarked on a comprehensive evaluation of various supervised learning models that work for binary classification problems. Once again, to test these models successfully, we split the data into training (“70%”) and testing (“30%”) data. The models which were considered possible candidates were:

1. Logistic Regression
2. Random Forest Classifier
3. Decision Trees
4. K-Nearest Neighbors
5. Multi-Layer Perceptron (MLP) classifier
6. Support Vector Classification (SVC)

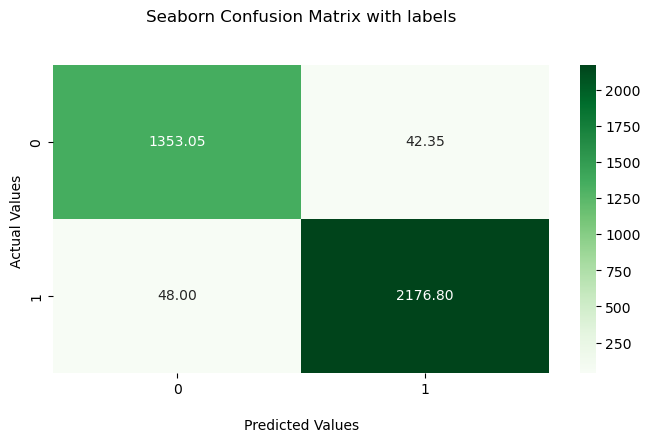
#### *Logistic Regression*

Logistic Regression was chosen as our primary model because of its effectiveness in binary classification tasks. As mentioned above, the models were run with cleaned data, and the tokenized and vectorized data, in order to give the model the full range of email information in a ‘friendlier’ format. The superiority of the vectorization method is evident from the results in Chart 17. The models were evaluated using a combination of metrics including accuracy, precision, Recall, and F1-Score, providing a multi-angled view of the model’s performance.



**Chart 17: Best Performing Models from the Second Approach (Vectorization) with Logistic Regressionf**

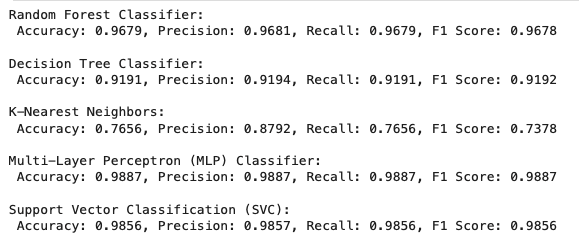
This confusion matrix in Chart 18 shows the improvements in running the logistic regression model using the tokenized and vectorized data over the previous approach, resulting in a low “false positive” and “false negative” rate.



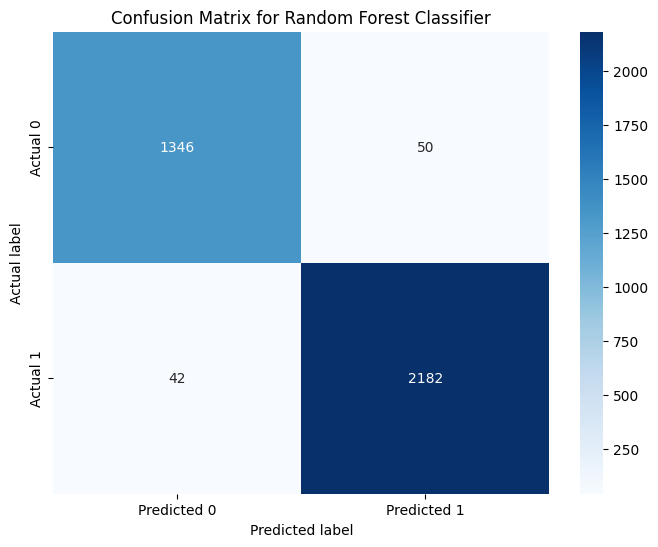
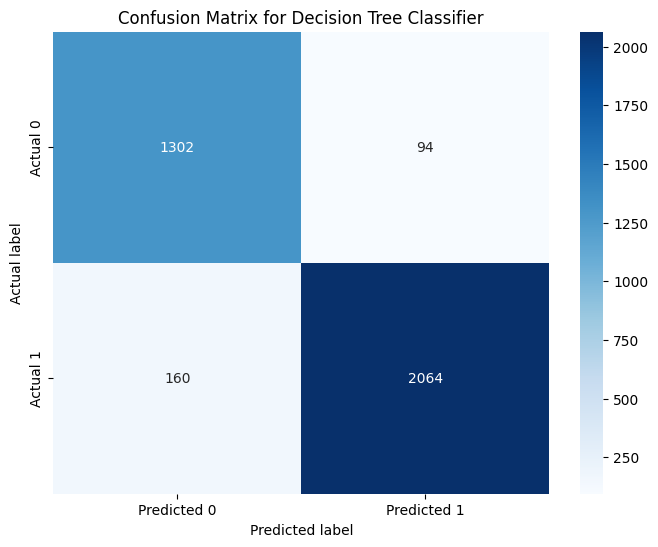
**Chart 18: Confusion Matrix for the Second Approach**

#### *Alternative Models*

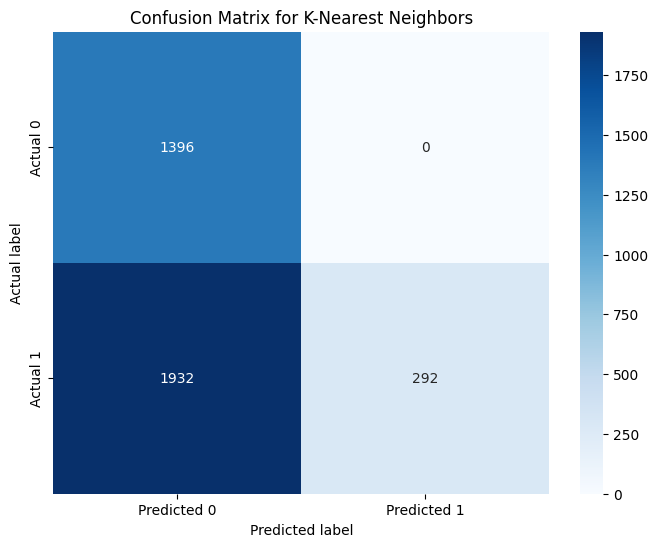
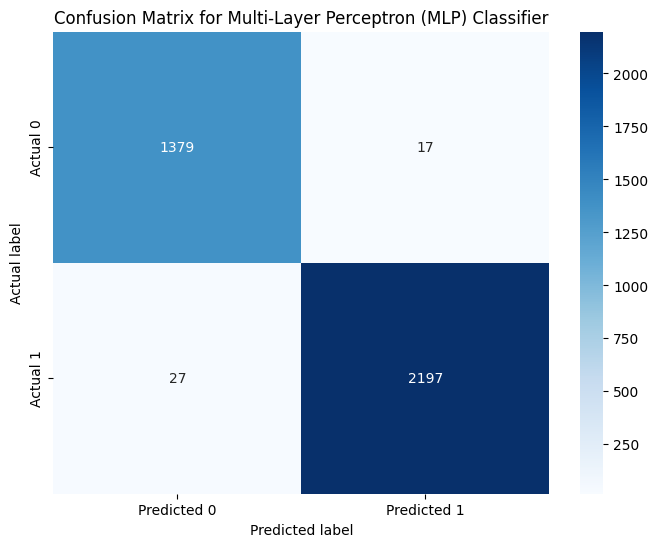
The results (see Chart 19 below) were similar to the alternative models, except for K-Nearest Neighbor. This was expected since KNN is the “lazy learner” of the bunch. We made confusion matrices (Charts 20a-e) for each of these models in an effort to visualize the performance of these models on the testing data.



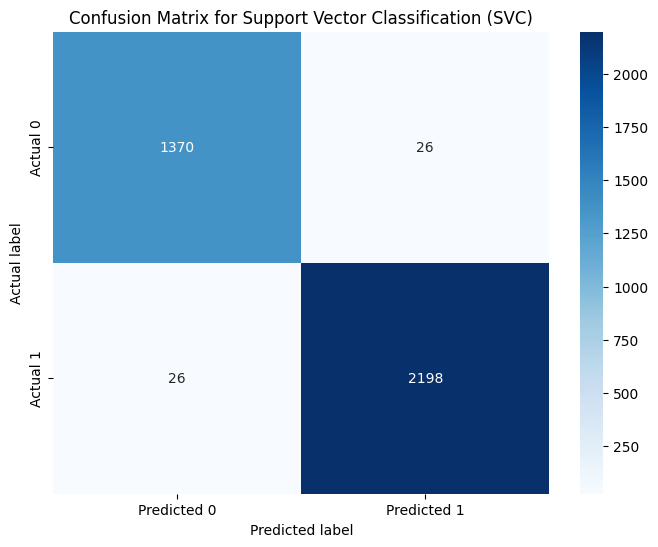
**Chart 19: Results from Alternative Approaches**



**Chart 20a: Confusion Matrix for the RFC Chart 20b: Confusion Matrix for the DTC**



**Chart 20c: Confusion Matrix for the KNN Classifier Chart 20d: Confusion Matrix for the MLP Classifier**



**Chart 20e: Confusion matrix for the SVC Classifier**

Given these results, even though MLP and SVC have the highest accuracy scores, we chose logistic regression as our model of choice due to its ability to perform reliably well in binary classification tasks (choose whether mail is “safe” or “phishing”). These results are encouraging, and they allow users to emulate any of the models listed above and follow our approach for classifying emails as ‘safe’ or ‘phishing’.

### ***Testing Approach***

Our overall goal was to find a model with high accuracy and precision with very low false negatives. The low false negatives were important as our goal was to detect phishing and malicious spam emails and a false negative getting through our filter could lead to a security breach, which would lead to a large deficit to our possible users.

We decided that at minimum, a model should have over 95% accuracy with around a 1% false negative rate. We chose to disregard the false positive error rate, since even if an email is filtered out, it can always be recovered by checking the spam folder, and thus does not pose a security threat to the user.

Furthermore, we used stratified K-fold while modeling to ensure accurate results due to our dataset being imbalanced at a roughly 60-40 split; phishing favored.

#### *Hyperparameter Tuning*

To tune the model, we compared the performance of several different models, each with varying hyperparameters. Chart 15 shows the best models from the first approach; over 300 models with slightly different hyperparameters competed against each other. The best approach would have the best f-score.

In contrast, Chart 17 shows the best-performing models of the Logistic Regression models using the vectorized text data. These results are very promising, and tuning it much farther beyond a 98.7 percent accuracy is difficult. We tried a few other approaches, but the hyperparameters can only do so much even with this second method. Still, this approach wildly outperforms our original hypothesis about the utility of special characters and other features for classification.

#### *Cross-Validation*

For cross-validation, we used scikit-learn’s cross\_validate package. A function was defined to evaluate our models which took in a dictionary of model objects, the feature and target variables, and the number of cross-validation folds as input parameters and output the recall, f1, accuracy, and precision scores for any model object passed through the input dictionary. This function evaluated these scores for each of our alternative models discussed above and output the scores as seen in Chart 19. Given that this function cross-validated 6 models, it is a computationally heavy and taxing function, and any attempts to re-run the evaluation function should take into account the high computational requirements for this function.

### ***Results***

For our final result, after iterating through different hyperparameters, we chose the following logistic regression model using the hyperparameters:

Penalty: L2 (Locked by solver choice)

C (Inverse regularization strength): 10.0

Solver: newton-cg

This model gave us a f1 score of 0.988127, leading to an accuracy of 98.8122% and precision of 98.8140%. The overall error for the dataset fluctuated between runs due to randomized data selection, but hovered around our goal of 1% false negatives.

This therefore gave us the confidence to determine that our model had met our previously set criteria.

### ***References***

Anjali, Sima (2024, February 24) *Phishing Dataset.* Kaggle. <https://www.kaggle.com/datasets/simaanjali/tes-upload>

Chakraborty, S. (2023, July 6). *Phishing email detection*. Kaggle. <https://www.kaggle.com/datasets/subhajournal/phishingemails/data>

Cisco. (2024, February 15). *What is phishing? examples and phishing quiz*. Cisco. <https://www.cisco.com/c/en/us/products/security/email-security/what-is-phishing.html#~phishing-awareness>

Hutto, C.J. & Gilbert, E.E. (2014). VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. Eighth International Conference on Weblogs and Social Media (ICWSM-14). Ann Arbor, MI, June 2014. [https://www.nltk.org/\_modules/nltk/sentiment/vader.htmltk.sentiment.vader](https://www.nltk.org/_modules/nltk/sentiment/vader.html)

Rabbi, M. F., Champa, A. I., & Zibran, M. F. (2023). Phishy? Detecting Phishing Emails Using ML and NLP. *2023 IEEE/ACIS 21st International Conference on Software Engineering Research, Management and Applications (SERA)*, 77–83.<https://doi.org/10.1109/SERA57763.2023.10197758>

Threatcop. (2023, October 17). *Reasons behind successful phishing attacks*. <https://threatcop.com/blog/how-are-phishing-attacks-successful/>

Woods, E. (2021, February 9). *The real reason for successful phishing attacks*. usecure Blog. <https://blog.usecure.io/the-real-reason-why-phishing-attacks-are-so-successful>