Final Project Report

INFO-I513 Usable AI

Sumanth Gopalkrishna and Sharanbasav Sumbad

DATA COLLECTION:

The Data for this Project was Borrowed from the Authors of the below cited Paper

Verma, R. M., Zeng, V., & Faridi, H. (2019). Data Quality for Security Challenges: Case Studies of Phishing, Malware and Intrusion Detection Datasets. *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2605–2607. Presented at the London, United Kingdom. doi:10.1145/3319535.3363267

This is a proprietary data can be collected again by the contacting the author. The author manually collected every email that are phishing and legit and stored them as text file (.txt extension) with and without the email headers. The data can be verified though either by contacting the author or by visiting the citation of the paper given above.

DATA MANAGEMNET:

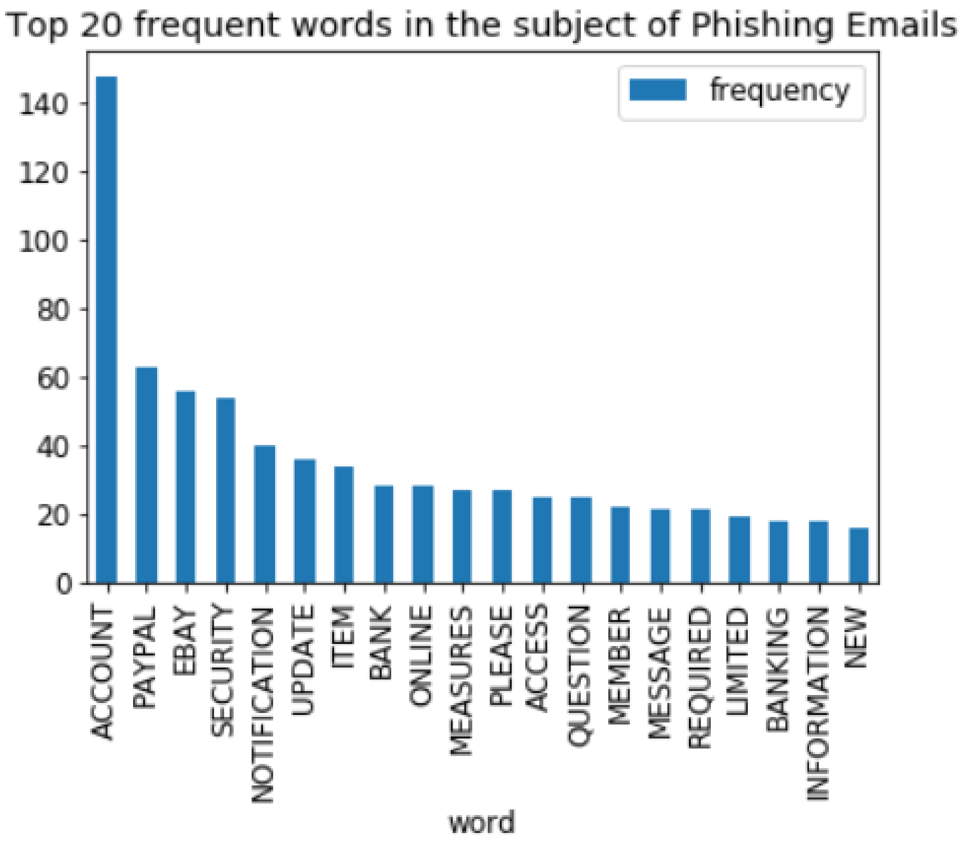
The Initial data was that we obtained was as text files. There were about 500 text files which were labeled as legit and around 4000 text files which were labeled as Phishing. Each individual text files had Email heads which will be used in pour analysis
In an e-mail, the body (content text) is always preceded by header lines that identify routing information of the message, including the sender, recipient, date, and subject. Some headers are mandatory, such as the FROM, TO and DATE headers. Others are optional, but very commonly used, such as SUBJECT and CC. Other headers include the sending time stamps and the receiving time stamps of all mail transfer agents that have received and sent the message. In other words, any time a message is transferred from one user to another (i.e., when it is sent or forwarded), the message is date/time stamped by a mail transfer agent (MTA) – a computer program or software agent that facilitates the transfer of email message from one computer to another. This date/time stamp, like FROM, TO, and SUBJECT, becomes one of the many headers that precede the body of an email. For initial analysis we run a python script to read through all the text files and read the contents of the text files to a data frame and label the each and individual text file as 0 for Legit and 1 for Phish. Then we also processed the data frame and to extract the headers of the text file using the python email parser library and extract the required hears for now that we initially plan to use. From all the headers in present in an email we choose to work with only the FROM, DATE, TO, SUBJECT and BODY as these are the main headers and can be found across all the emails.

The next steps after the data were prepared for preprocessing, we had to clean the data, we did not have any missing columns or records as the ingestion was done manually. Since we are dealing with text data, we had use NLP for cleaning and text analysis.

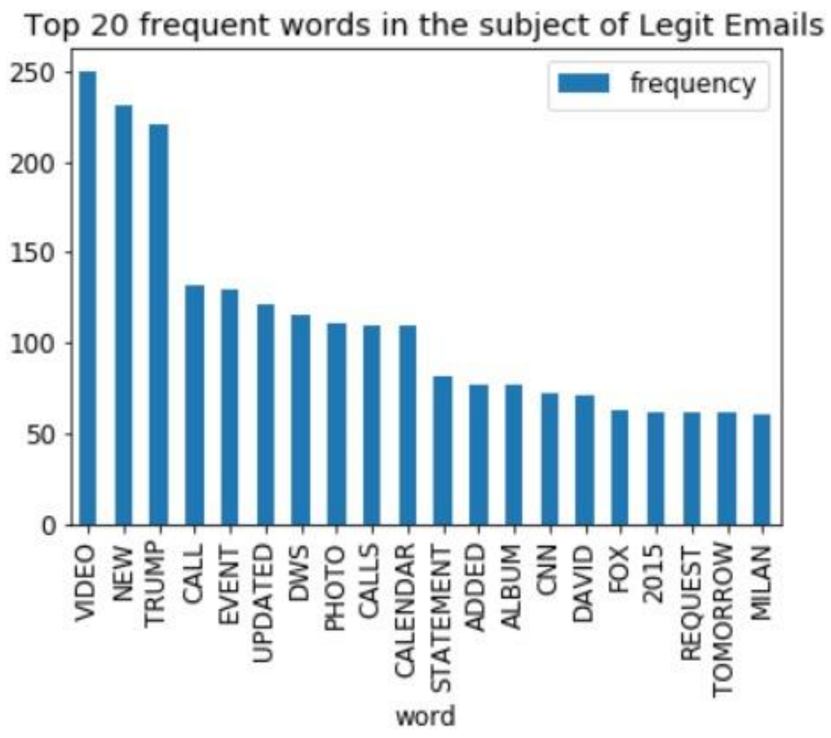
Data cleaning steps:

1. Removal of stop words
2. Removal of punctuation marks.
3. Removal of special characters
4. Tokenization
5. Capitalization of words

DATA ANALYSIS:

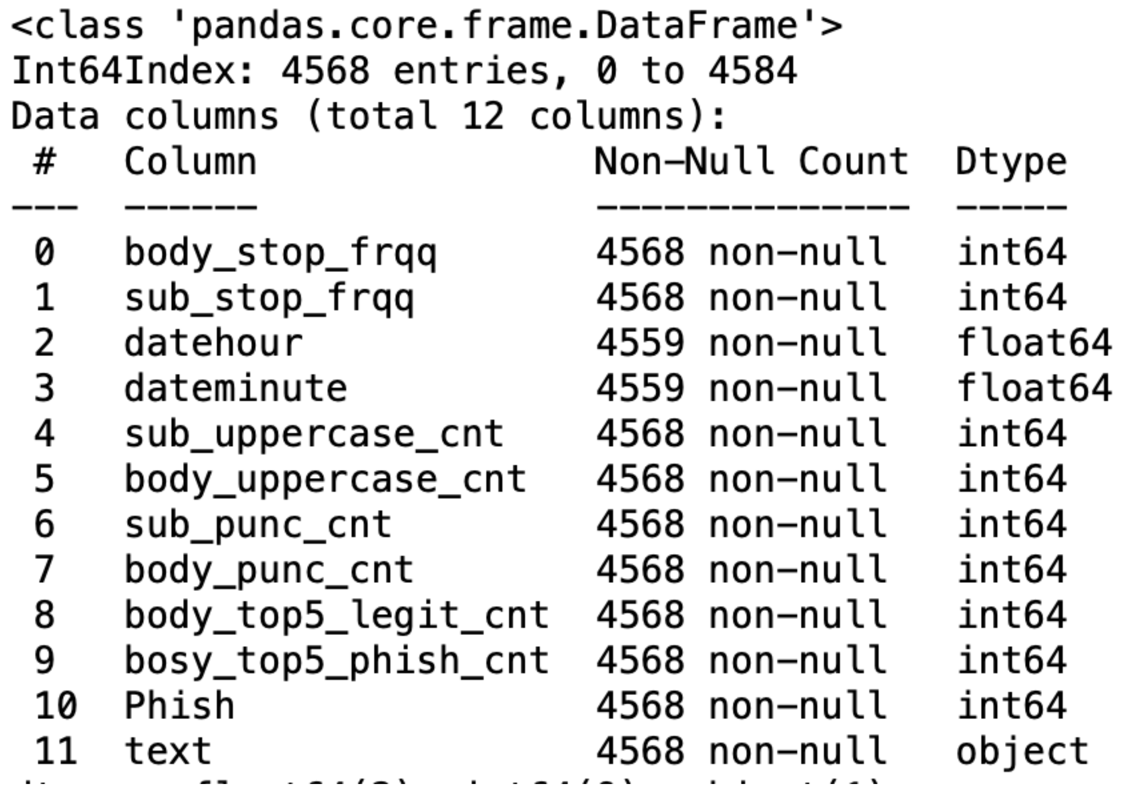


The above Bar chart shows the top 20 frequent words that appear in the Subject of the email which is not legit. From our analysis from the corpus, we have we see that word 'Account' that appears most frequent (148 times) that are phishing. That makes a lot of sense as most phishing emails tend to get users bank account details or other financial account details that need to target. The next most frequent word is the word 'PayPal' (63 times), as we have Identified PayPal is a software platform that deals with banks and customers with transaction related queries.



We also look at the words that appear in the Legit set of email's subjects, the words such as 'Video', 'New', 'Trump', 'Call' appear most frequently which I suppose are random to any corpus. As now we cannot derive any relation between the words as we did for phishing emails but through further analysis maybe we can pinpoint the characteristics. But surely, we can say the word 'video' appeared more in the legit documents the most, this may be contributed to the factored that the data is imbalanced but also, according to that logic the word 'video' should have appeared far a greater number of times.

From the above analysis we choose to go ahead with count vectorization method and add in some more features such as count of punctuation, count of capital letters and so on. Since the count vectorization would result in high dimensional data, we used the words frequency and designed a feature to just keep the top 1000 words and use their frequency to as features. Apart from this since we have a datetime column we also used the hour of the day and minutes to add into our features.



```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4568 entries, 0 to 4584
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   body_stop_frqq        4568 non-null   int64  
 1   sub_stop_frqq         4568 non-null   int64  
 2   datehour              4559 non-null   float64
 3   dateminute            4559 non-null   float64
 4   sub_uppercase_cnt     4568 non-null   int64  
 5   body_uppercase_cnt    4568 non-null   int64  
 6   sub_punc_cnt          4568 non-null   int64  
 7   body_punc_cnt         4568 non-null   int64  
 8   body_top5_legit_cnt   4568 non-null   int64  
 9   bosy_top5_phish_cnt   4568 non-null   int64  
 10  Phish                 4568 non-null   int64  
 11  text                  4568 non-null   object 
```

The above figure shows the all the features that we were able to extract from our text preprocessing.

body_stop_frqq: Frequency of stop words in body of a particular email

sub_stop_frqq: Frequency of stop words in subject of a particular email

datehour: Hour of the day of a particular email it was sent

dateminute: minute of the Hour of a particular email it was sent

sub_uppercase_cnt: Frequency of Capital letters in Subject of a particular email

body_uppercase_cnt: Frequency of Capital letters in body of a particular email

sub_punc_cnt: Frequency of punctuation in the subject of a particular email

body_punc_cnt: Frequency of punctuation in the body of a particular email

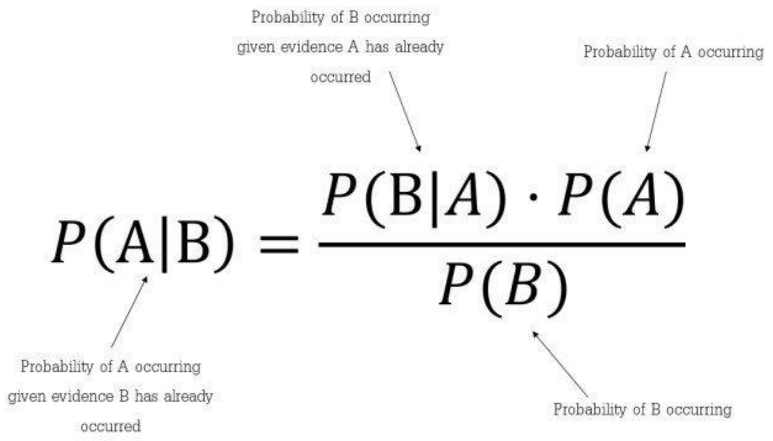
body_top5_legit_cnt: Frequency of top 5 words in legit emails in a particular email

bosy_top5_phish_cnt: Frequency of top 5 words in phishing emails in a particular email

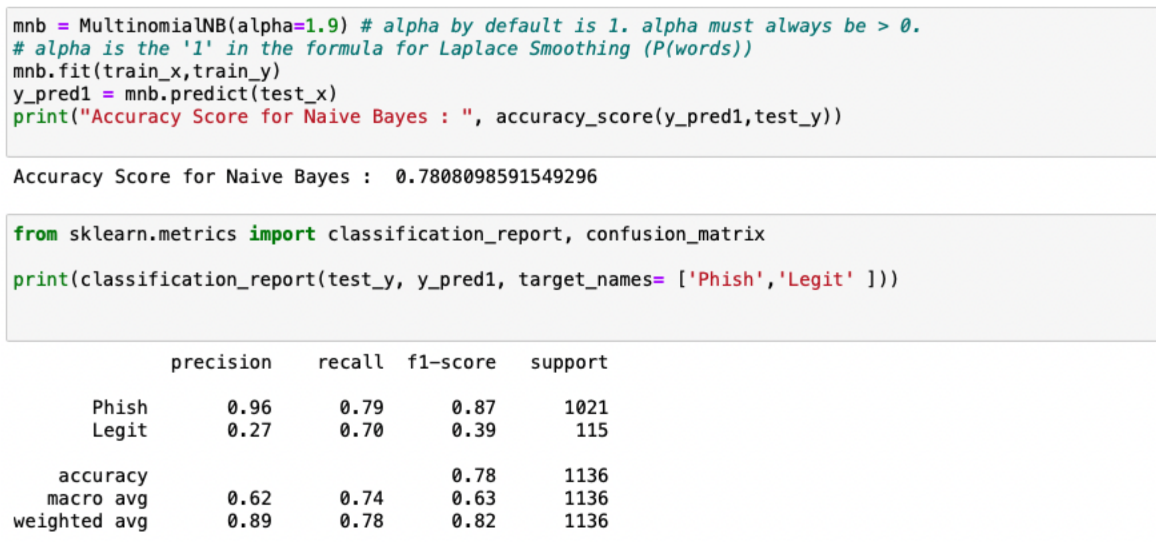
As we analyzed we found out most of the phishing emails have an unfamiliar tone or greeting. When reading im properly used words. Grammar and spelling errors. Inconsistencies in email addresses, links and domain name, Threats or a sense of urgency, unusual requests. These observations are supported by analysis of top 50words, as we saw above, the phishing emails have words related to banking terms, like Payment, Account and so on. Based on this intuition we choose to extract the count of top 5 words in phishing emails and legit and use them as features in our code.
As for data sub-setting we do that while training our model and test the trained model on the test by using the Sklearn. Also as mentioned before while using the count vectorization we don't necessarily use all the words in the text data, while using count vectorization, we remove stop words, punctuation, all numeric characters, empty strings, and all other non-English words. and use only top 1000 words and their frequency for classification. This is done to avoid the curse of dimensionality into our ML algorithm.

DATA ARGUMENT:

In this section we talk about the ML algorithm that we will be using for the classification, for first instinct approach we went ahead with Multinomial Naïve Bayes Algorithm. There are thousands of software's or tools for the analysis of numerical data but there are very few for texts. Multinomial Naive Bayes is one of the most popular supervised learning classifications that is used for the analysis of the categorical text data.
Multinomial Naive Bayes algorithm is a probabilistic learning method that is mostly used in Natural Language Processing (NLP). The algorithm is based on the Bayes theorem and predicts the tag of a text such as a piece of email or newspaper article. It calculates the probability of each tag for a given sample and then gives the tag with the highest probability as output.Naive Bayes classifier is a collection of many algorithms where all the algorithms share one common principle, and that is each feature being classified is not related to any other feature. The presence or absence of a feature does not affect the presence or absence of the other feature.
Naive Bayes is a powerful algorithm that is used for text data analysis and with problems with multiple classes. To understand Naive Bayes theorem's working, it is important to understand the Bayes theorem concept first as it is based on the latter. Bayes theorem, formulated by Thomas Bayes, calculates the probability of an event occurring based on the prior knowledge of conditions related to an event. It is based on the following formula:
P(A|B) = P(A) * P(B|A)/P(B) Where we are calculating the probability of class A when predictor B is already provided.


Probability of B occurring given evidence A has already occurred
Probability of A occurring
$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$
Probability of A occurring given evidence B has already occurred
Probability of B occurring

The Multinomial Naive Bayes can be accepted as the probabilistic approach to classifying documents in the case of acknowledging the frequency of a specified word in a text document. The term “bag of words” is widely used as the selected document to be processed under the context of Naive Bayes while depicting the document itself as a bag and each vocabulary in the texture as the items in the bag by permitting multiple occurrences. To be able to properly classify, the existence of the word in the given text shall be known beforehand. This classifier achieves well on discrete types as the number of words found in a document. An example of the usage area of this approach can be the prediction of the matching category of the document with the help of the occurrences of the words allocated in the document. The output of this algorithm produces a vector composed of integer frequency values of the word set.



```
mnb = MultinomialNB(alpha=1.9) # alpha by default is 1. alpha must always be > 0.
# alpha is the '1' in the formula for Laplace Smoothing (P(words))
mnb.fit(train_x,train_y)
y_pred1 = mnb.predict(test_x)
print("Accuracy Score for Naive Bayes : ", accuracy_score(y_pred1,test_y))
```

```
Accuracy Score for Naive Bayes :  0.7808098591549296
```

```
from sklearn.metrics import classification_report, confusion_matrix

print(classification_report(test_y, y_pred1, target_names= ['Phish','Legit' ]))
```

```
              precision    recall  f1-score   support

       Phish       0.96      0.79      0.87      1021
       Legit       0.27      0.70      0.39       115

    accuracy                           0.78      1136
   macro avg       0.62      0.74      0.63      1136
weighted avg       0.89      0.78      0.82      1136
```

Above code snippet is the result of the base model of Multinomial Naïve Byes approach, The results shown at this stage are good enough to continue this approach. This approach can be improved by

- Removing stop words: common words that don’t add value. For example, such as, able to, either, else, ever, etc.
- Lemmatizing words: Grouping different inflections of the same word. For example, draft, drafted, drafts, drafting, etc.
- N-grams: The n-gram is the probability of the appearance of a word or a sequence of words of ‘n length’ within a text.
- TF-IDF: Short for term frequency-inverse document frequency, TF-IDF is a metric that quantifies how important a word is to a document in a document set. It is very powerful when used to score words, i.e. it increases proportionally to the number of times a specific word appears in a document, but is offset by the number of documents that contain said word.

We will be trying out other approaches also and then evaluate in that how well the modal is performing. Other Approaches can be as follows:

Support Vector Machines

Support Vector Machines (SVM) is a classification algorithm that performs at its best when handling a limited amount of data. It determines the best result between vectors that belong to a given group or category, as well as the vectors that don’t belong to the group.
For example, let’s say you have two tags: expensive and cheap and the data has two features: x and y. For each pair of coordinates (x, y) there should be a result for each one to determine which is expensive and which one is cheap. To do this, SVM creates a divisor line between the data points, known as the decision boundary, and classifies everything that falls on one side as expensive and everything that falls on the other side as cheap.

The decision boundary divides a space into two subspaces, one for vectors that belong to a group and another for vectors that don't belong to that group. Here, vectors represent training text, and a group represents the tag you use to tag your texts. A perk of using SVM is that it doesn't require a lot of training data to produce accurate results, although it does require more computational resources than Naive Bayes to yield more accurate results.

Deep Learning

Deep Learning is comprised of algorithms and techniques that are designed to mimic the human brain. With text classification, there are two main deep learning models that are widely used: Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN).

CNN is a type of neural network that consists of an input layer, an output layer, and multiple hidden layers that are made of convolutional layers. Convolutional layers are the major building blocks used in CNN, while a convolution is the linear operation that automatically applies filters to inputs – resulting in activations. These complex layers are key ingredients in a convolutional neural network as they assign importance to various inputs and differentiate one from the other.

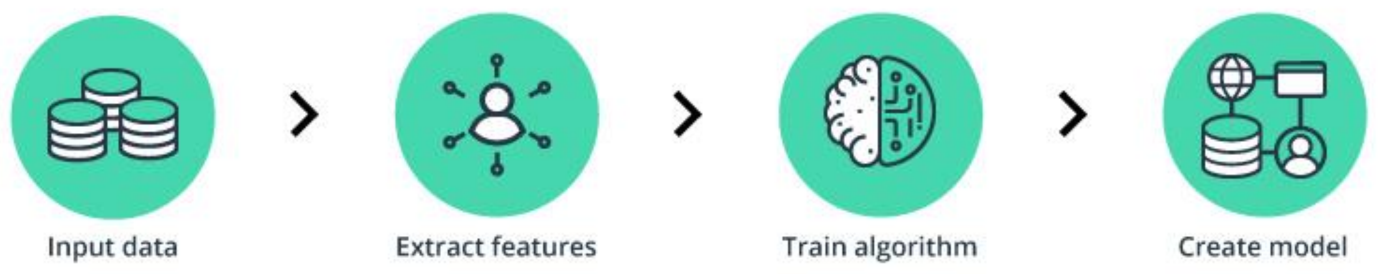
Within the context of text classification, CNN represents a feature that is applied to words or n-grams to extract high-level features are specialized neural-based networks that process sequential information. For each input sequence, the RNN performs calculations that are conditioned to the previous computer outputs. The key advantage of using RNN is the ability to memorize the results of previous computations and use it for current ones. It's important to remember that deep learning algorithms require millions of tagged examples, as they work best when fed more data.

Metrics and Evaluation

One of the most common types of performance evaluators of a text classifier is cross-validation. In this method, training datasets are randomly divided into same-length data sets. Then, for each same-length set, the text classifier is trained with the remaining sets to test predictions. This helps classifiers make predictions for each corresponding set, which they compare with human-tagged data to avoid false positives or false negatives.

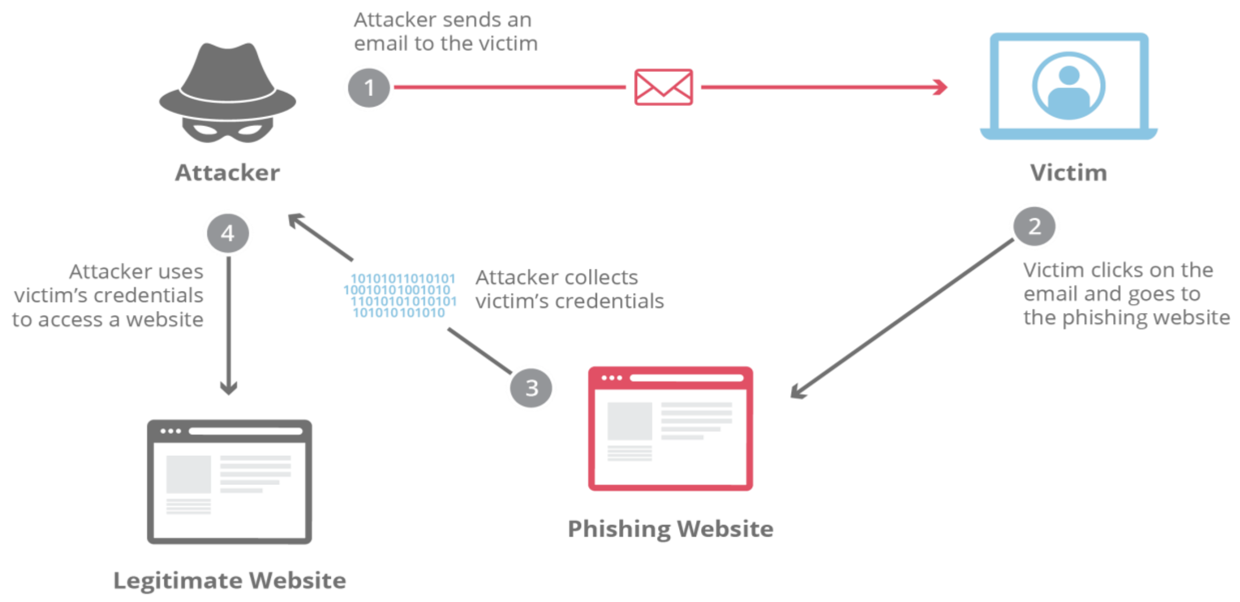
These results lead to valuable metrics that demonstrate how effective a classifier is:

- Accuracy: percentage of texts that were predicted with the correct tag.
- Precision: percentage of texts the classifier got right out of the total number of examples it predicted for a specific tag.
- Recall: percentage of examples the classifier predicted for a specific tag out of the total number of examples it should have predicted for that tag.
- F1 Score: harmonic mean between precision and recall.



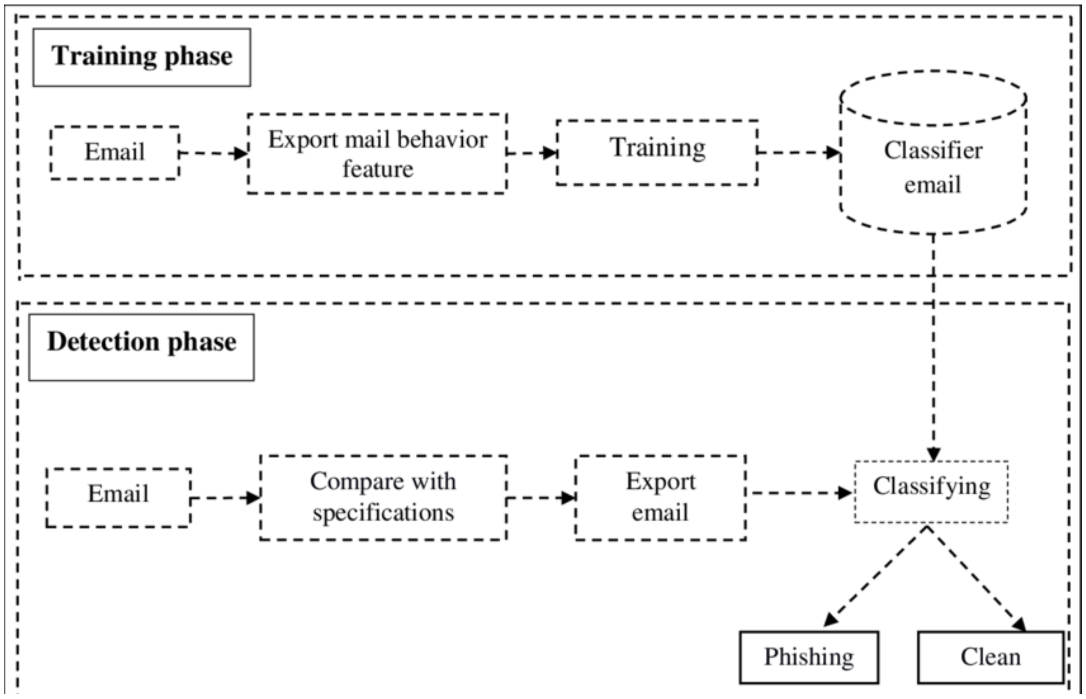
STAKEHOLDER ANALYSIS:

Phishing email attacks are social engineering email attacks in which victims are directed to websites that are impersonating reputable sites via email. Phishing websites are usually well-hidden. Phishing email victims mistakenly believe these sites are affiliated with reputable firms such as Amazon or Google and are thus duped into logging in and giving important information. Every day, an estimated 269 billion emails are sent (Danny,2020), with around one in every 2,000 of them being phishing emails, totaling 135 million phishing assaults every day. Though the precise cost is impossible to calculate, the FBI estimates that phishing assaults cost US firms roughly $5 billion each year.
According to the Anti-Phishing Working Group (APWG)2, the payment industry is the most targeted (39.4%), webmail is the second most targeted (18.7%), and financial institutions are the third most targeted (14.2%). (APWG, 2021). Because phishing assaults affect millions of internet users (both individuals and businesses), and the APWG reports a steady growth in unique phishing sites, it's becoming increasingly necessary to find measures to protect ourselves. One of the most difficult challenges is defending against phishing assaults (Thakur and Kaur, 2016). Even though numerous spam email filters have been produced, few phishing email filters have been developed ML filters have produced the greatest results so far of the phishing email filters that have been created Phishing e-mails are a security concern that has caused considerable financial losses for corporations as well as annoyance to internet consumers. As attackers have become more knowledgeable and competent, phishing advertising and pornographic e-mails have become more difficult to detect. Attackers keep track of users and tailor their attacks to their interests and popular subjects, which may be gleaned through community news and diaries. The focus of this study is on misleading Phishing attacks and its variations, such as ads and pornographic e-mails. We propose the Phishing Alerting System (PHAS) as a framework for correctly classifying e-mails as Phishing, advertising, or pornographic. PHAS can detect and notify users to all forms of fraudulent e-mails, assisting them in making decisions.
Because of the importance of protecting online users from becoming victims of online fraud, divulging confidential information to an attacker, and other effective uses of phishing as an attacker's tool, phishing detection tools play a critical role in ensuring that users have a secure online experience. Unfortunately, many existing phishigdetection tools, particularly those that rely on an existing blacklist, have fl aws such as low detection accuracy and a high rate of false alarms, which are frequently caused by either a dela y in blacklist update becauseof the human verification process involved in classification, or by human error in cl assification, which canresult in incorrect classification of the classes.Because of the unpredictable nature of atta ck behaviors and the everchanging URL phish patterns, the referencemodel must be updated on a regular basis. As a result, an efficient approach for regulating retraining is required for the machine learning algorithm to activ ely adapt to changes in phishing trends.


Attacker sends an email to the victim
1
Attacker
Victim
2
Victim clicks on the email and goes to the phishing website
3
Attacker collects victim's credentials
4
Attacker uses victim's credentials to access a website
Phishing Website
Legitimate Website

CONTEXT:

In this Code Pattern, we can create an app that categorizes emails, designating them as "Phishing" or "Legit" if they don't appear to be suspicious. We'll train a model with email instances from an email dataset using IBM using the classifier we built the custom model can be developed fast and simply in the Web UI, then uploaded to the NodeJS app using the Cloud Nodejs SDK and launched directly from the browser. This will offer email gateway as well as direct API connection with email software. Every client in the company has access to internal, external, and historical email communication thanks to this connection. These communications are used by ML pipeline to discover communication trends inside the organization and between personnel. The API design will provide real-time phishing and impersonation security, with minimal impact on email or network speed.



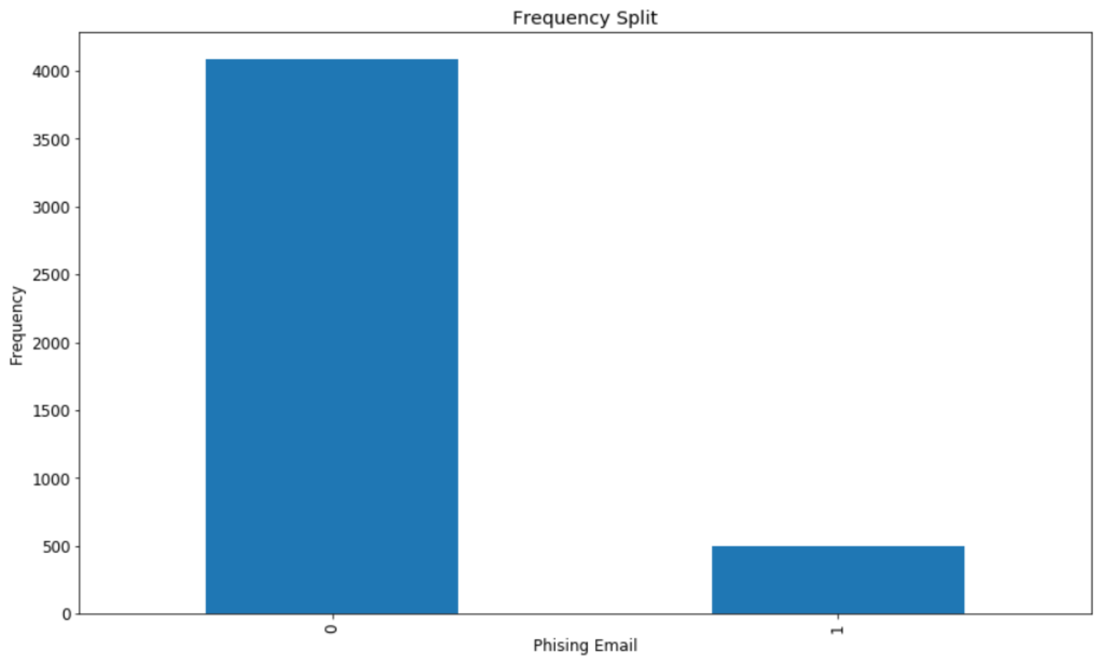
Since Emails are being shared with an app client a security remains that needs to be addressed which can be done with high level of encryption. Is if a user is also sharing personal emails for the benefit for the application performance there is also a risk of privacy. So, the requirements that come about due to these reasons we need to develop a flow that can handle these issues without any concern.

Individuals, businesses, educational institutions, and others are vulnerable to phishing because the bad guys might obtain access to financial information, personal data, confidential corporate information, health information, student data, and more.

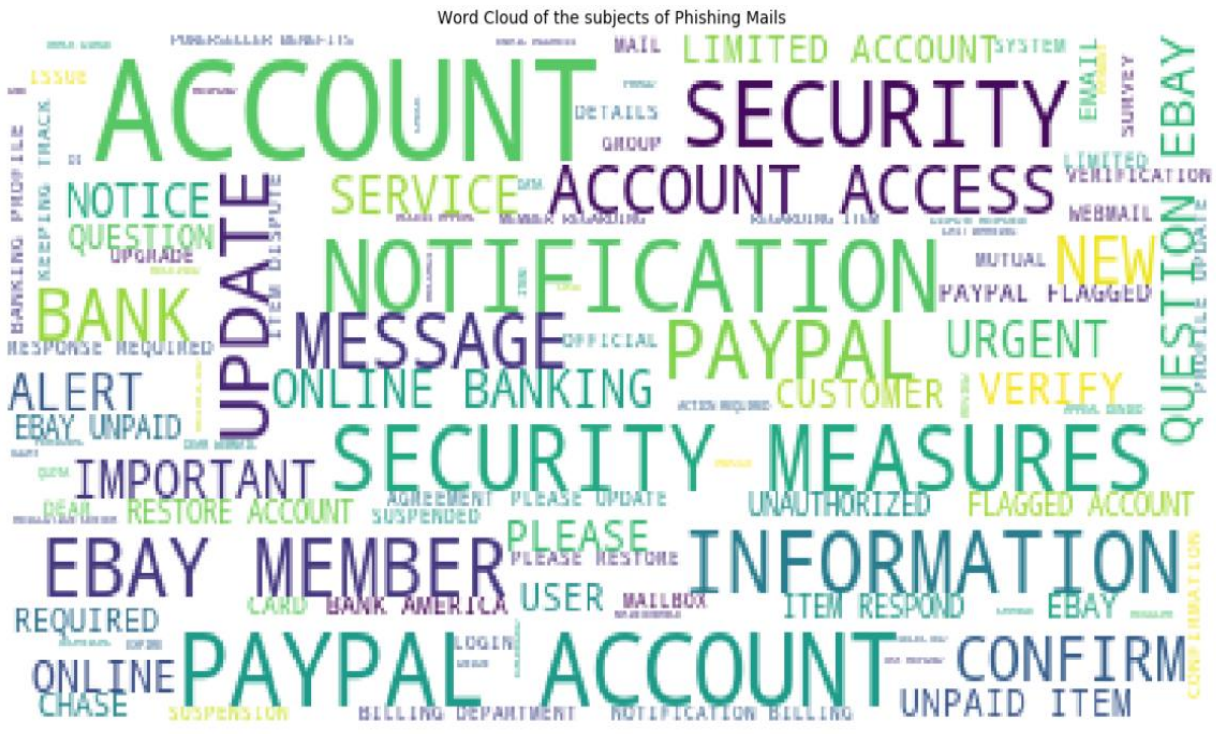
Firms and educational institutions may be blacklisted by internet or financial services companies in extreme situations of phishing attacks, preventing the entities and their workers from communicating with the outside world and paying for products and services.

Phishing may also take up important time from employees, such as those in IT and HR departments, by diverting their focus away from their typical productive work to repair the harm created by phishing.

VISULIZATIONS:

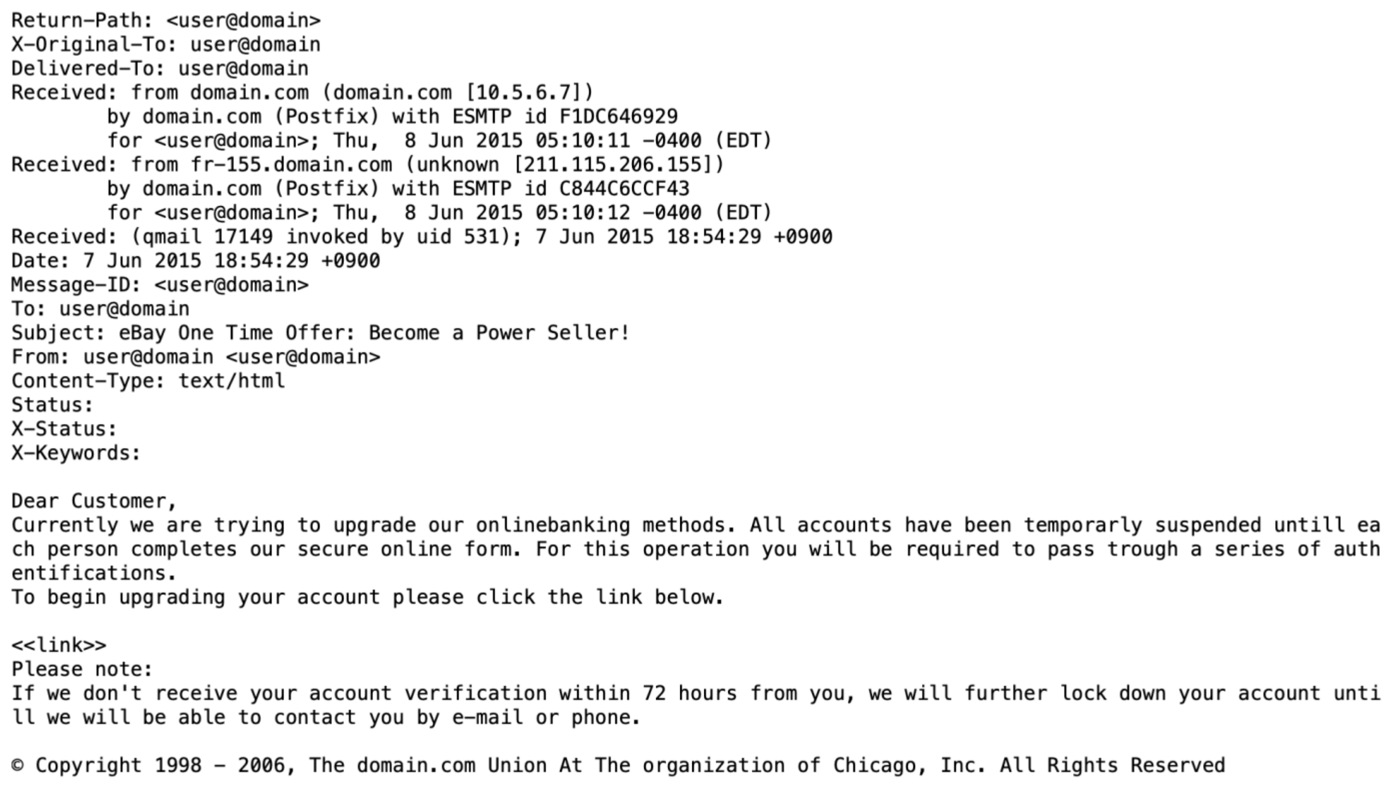


This is the data distribution of the corpus as legit and phish. ‘0’ is labelled for legit and ‘1’ is labeled for phish in the data frame. As initially pointed, we have around 4000 legit emails and 500 phish emails to train our classifier.



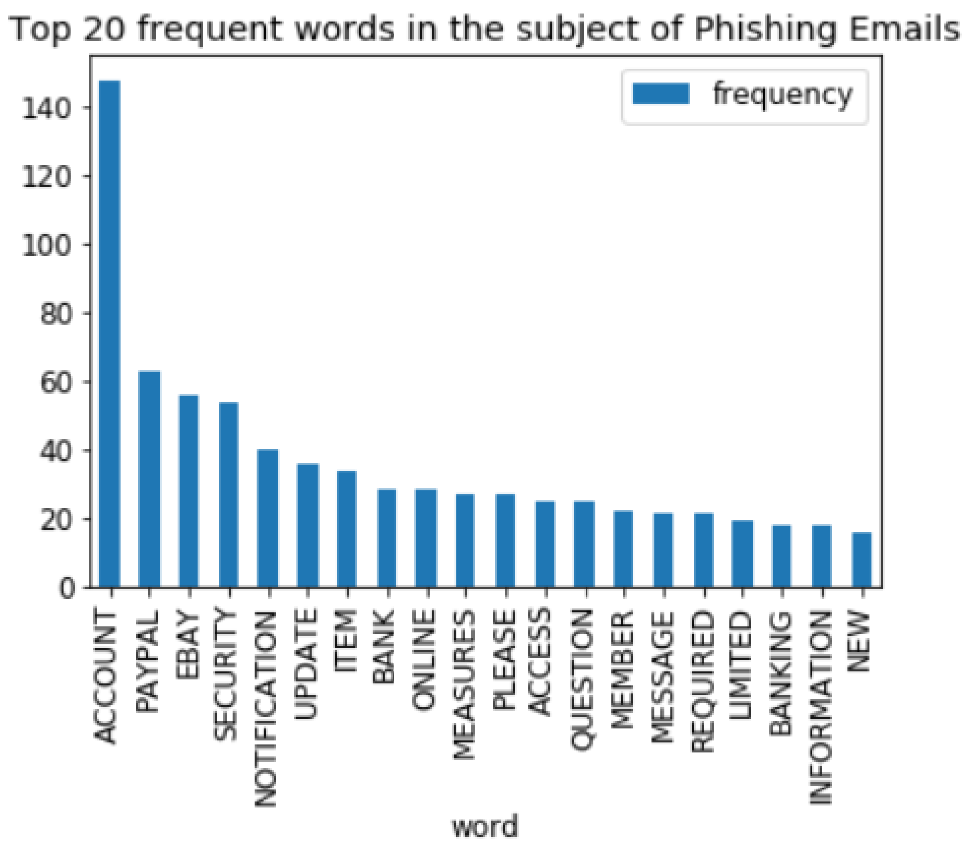
A word cloud visualization of the subject of the phishing email document. After just a glance at the cloud we can say that most of them are banking terms. So, we can conclude that after initial analysis that we can suspect that the emails with Banking terms can serve as indicator for a phishing email.

From our Corpus This is what a Phishing email looks like:



```
Return-Path: <user@domain>
X-Original-To: user@domain
Delivered-To: user@domain
Received: from domain.com (domain.com [10.5.6.7])
        by domain.com (Postfix) with ESMTP id F1DC646929
        for <user@domain>; Thu,  8 Jun 2015 05:10:11 -0400 (EDT)
Received: from fr-155.domain.com (unknown [211.115.206.155])
        by domain.com (Postfix) with ESMTP id C844C6CCF43
        for <user@domain>; Thu,  8 Jun 2015 05:10:12 -0400 (EDT)
Received: (qmail 17149 invoked by uid 531); 7 Jun 2015 18:54:29 +0900
Date: 7 Jun 2015 18:54:29 +0900
Message-ID: <user@domain>
To: user@domain
Subject: eBay One Time Offer: Become a Power Seller!
From: user@domain <user@domain>
Content-Type: text/html
Status:
X-Status:
X-Keywords:

Dear Customer,
Currently we are trying to upgrade our onlinebanking methods. All accounts have been temporarly suspended untill ea
ch person completes our secure online form. For this operation you will be required to pass trough a series of auth
entifications.
To begin upgrading your account please click the link below.

<<link>>
Please note:
If we don't receive your account verification within 72 hours from you, we will further lock down your account unti
ll we will be able to contact you by e-mail or phone.

© Copyright 1998 - 2006, The domain.com Union At The organization of Chicago, Inc. All Rights Reserved
```

From the above emails there a lot of meta-data info and hidden features that can help us in detection of an phishing email. Such as most Frequent words in phishing emails:



ETHICS:

There are several anti-phishing programs professing to safeguard you, just as there are many hackers eager to lure you into their nasty phishing schemes. Though there appear to be a plethora of options, they all fit into one of a few groups based on the strategy they take.

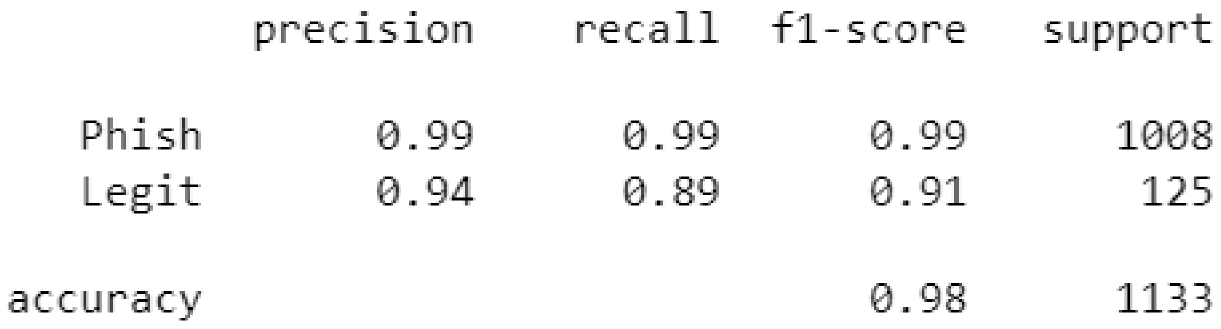
"Google and Microsoft have my back, they're industry titans," one would believe when it comes to phishing protection. While they are unquestionably industry leaders, they may not necessarily give the best security. Although G-Suite and Office 365 have recently improved their anti-phishing defenses, they're still not foolproof (for example, Microsoft's Advanced Threat Protection (ATP) requires you to define (and maintain) complicated rules for catching different types of phish, and both providers largely ignore file sharing, messaging, and other outlets for malicious phishing attacks).Spam filters are ultimately responsible for most of the provider's email security, and with roughly 15 billion spam emails sent every day, spam filters are working overtime. It's inevitable that some phish will get through. More focused on spam than phish. Require customers to manage cumbersome security rules.

Companies who want to go beyond the protection provided by their email provider might use a gateway-based technology. These businesses used to be appliance-based, but they've subsequently "cloud-washed" themselves, which doesn't make them cloud-native. Email must be sent through a message transfer agent to their servers to use their anti-phishing services. A reroute not only delays the delivery of your emails, but most MTAs only check incoming mail. Outgoing emails and employee-to-employee communications are left unscanned, placing companies and people with whom they communicate at risk. Hackers frequently acquire employee credentials to initiate phishing scams. The phishing email will appear to be a valid outbound email or employee-to-employee communication using the stolen credentials. Require us to route traffic through their cloud, which can cause misconfiguration issues, lead to privacy and security concerns, and can have performance impact.

Natural language processing (NLP) is being used to identify phishing in a new wave of phishing prevention. NLP stands for "communication profiling," and it works on the premise that quirks in a person's writing may be utilized to detect phishing emails. Proponents of NLP think that by studying word and phrase choices over time, "profiles" may be created, and that anything that deviates from the profile would be deemed a phishing fraud.
While this anti-phishing software is still in its early phases of development, one possible difficulty for it may be that individuals write differently depending on who they're writing to; an email to an executive is likely to seem quite different than an email to a close colleague. Unproven technology that may require long “ramp time” before becoming effective and can lead to a high false-positive rate.

Results:

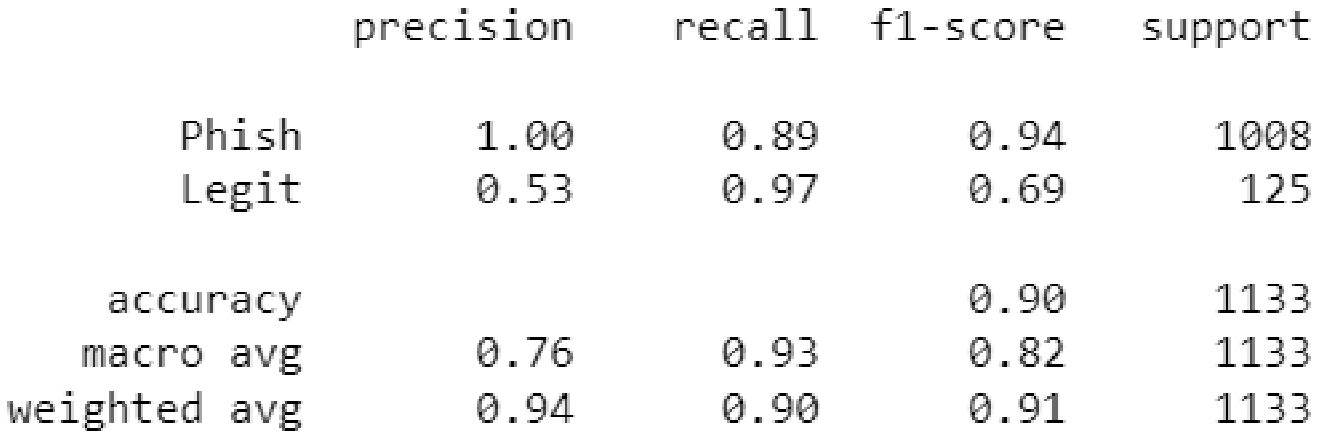
Logistic Regression model:



```
              precision    recall  f1-score   support

       Phish       0.99      0.99      0.99      1008
       Legit       0.94      0.89      0.91       125

    accuracy                           0.98      1133
```

Naive bayes model (Baseline):

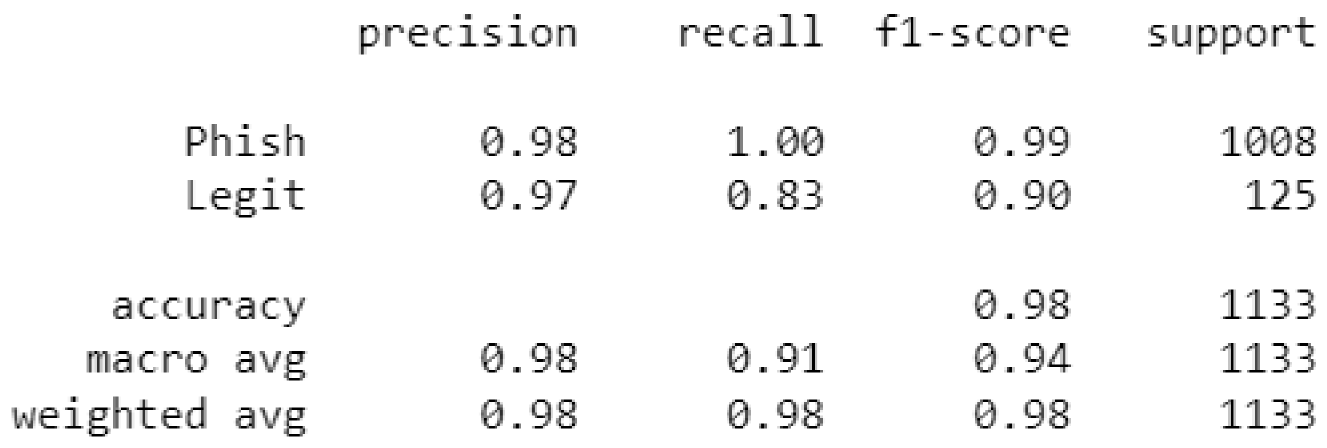
Precision = 0.54 **Recall** = 0.97



```
              precision    recall  f1-score   support

       Phish       1.00      0.89      0.94      1008
       Legit       0.53      0.97      0.69       125

    accuracy                           0.90      1133
   macro avg       0.76      0.93      0.82      1133
weighted avg       0.94      0.90      0.91      1133
```

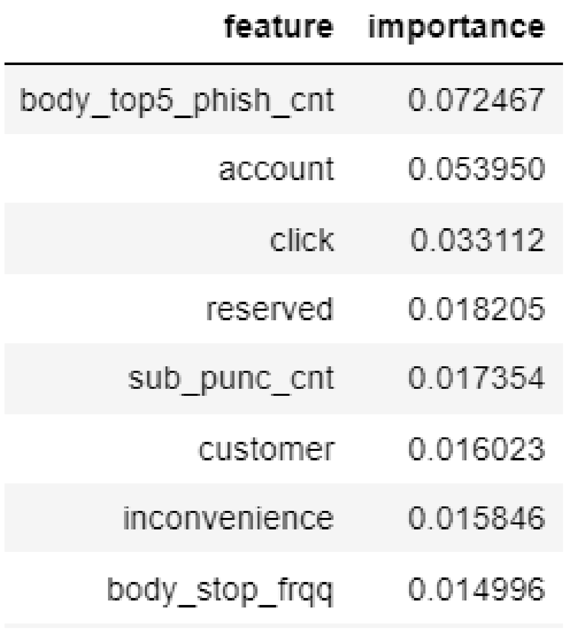
Random Forest model:

Precision = 0.98 **Recall** = 0.84

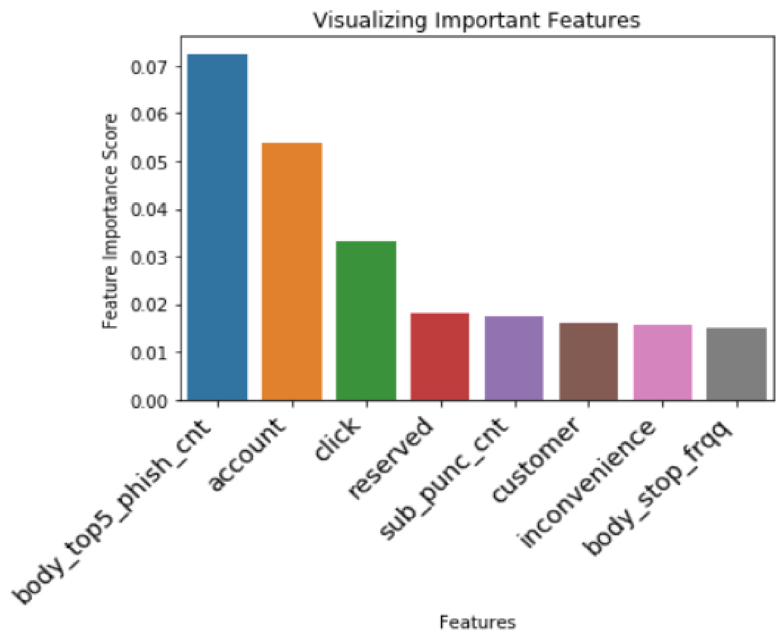


```
              precision    recall  f1-score   support

       Phish       0.98      1.00      0.99      1008
       Legit       0.97      0.83      0.90       125

    accuracy                           0.98      1133
   macro avg       0.98      0.91      0.94      1133
weighted avg       0.98      0.98      0.98      1133
```

Feature importance:



feature	importance
body_top5_phish_cnt	0.072467
account	0.053950
click	0.033112
reserved	0.018205
sub_punc_cnt	0.017354
customer	0.016023
inconvenience	0.015846
body_stop_frqq	0.014996



Future Works:

- Use TF-IDF and check model performance
- Perform sentiment analysis on the text data
- Tree based boosting models like XgBoost and Neural Network