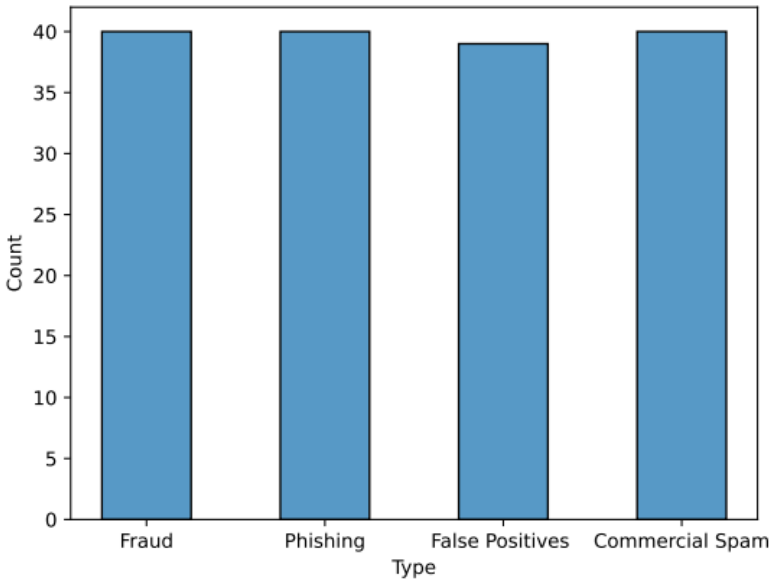
Introduction

The proliferation of online communication has led to an exponential increase in the volume of emails being exchanged. Unfortunately, this has also given rise to an increase in the number of fraudulent or spam emails that are sent with the intent to deceive individuals, especially those who are elderly or less tech-savvy. As a result, it is important to develop automated systems that can quickly and accurately distinguish between legitimate and illegitimate emails to prevent financial loss or damage to reputation. Naive Bayes Classifier is a simple but powerful algorithm that has been widely used for text classification, making it an ideal candidate for determining if an email is spam or not based on keywords. The report aims to provide insights into the effectiveness of the classifier and explore ways to optimize its performance.

Github: https://github.com/Jamessanfor/Project5
Presentation: Presentation

Dataset

For this project, we used the "Phishing Email Data by Type" dataset from Kaggle and applied the Naive Bayes Classifier to determine whether an email was a scam or not. The dataset included email subject lines, content, and labels such as fraud, phishing, spam, or legitimate emails. We consolidated the fraud and phishing categories to focus solely on identifying scam emails. To do this, we created a list of keywords based on the email content that we believed would indicate if an email was a scam. Using this approach, we were able to classify emails as legitimate or scams and demonstrated the effectiveness of the Naive Bayes Classifier for this purpose.



Analysis technique

For this project, we employed Multinomial Naive Bayes as it was appropriate for counting the frequency of keywords in emails. Gaussian Naive Bayes was not preferred as we were only interested in the presence or absence of keywords. To achieve this, we developed the function 'sus_words_search' that tallied the keyword occurrences in an email and subsequently used this information to train our algorithm. The below code was used to determine our results:

```
def sus_words_search(keywords, email):
    count=0;
    email= email.lower()
    for ele in keywords:
        if ele in email:
            count=count+1
    return {'trigger_words' : count}
featuresets = [(sus_words_search(keywords, ele1), ele2) for (ele1, ele2)
in emails_text ]
train_set, test_set = train_test_split(featuresets)
classifier = nltk.NaiveBayesClassifier.train(train_set)
```

Results

We used the train_test_split() function to test our data, training on 75% and testing on 25%. The precision, recall, and F1-score were used to evaluate the model's performance in identifying 'Valid' and 'Scam' instances. The precision for 'Valid' ranged from 0.65 to 0.825, with a mean of 0.7525, indicating the model predicted 75.25% of 'Valid' instances correctly. Recall for 'Valid' was 1, meaning all actual 'Valid' instances were correctly identified. The F1-score for 'Valid' ranged from 0.788 to 0.90, with an average of 0.8576. No 'Scam' instances were predicted. While we believe no alternative approaches would have produced different results, the model performed well in identifying 'Valid' instances.

We also noticed that trigger-free emails are twice as likely to be valid, while emails with two or more trigger words are three times more likely to be spam. Emails with one trigger word are equally likely to be spam or valid.

Based on the results, we think this model can help those who are more susceptible to email scams by identifying valid emails with greater accuracy and reducing the chance of scam emails ending up in their inbox. While the model may not catch all scam emails, with additional time, it could be improved to more effectively distinguish between scam and valid emails, further increasing its usefulness.