

# COMP8060: Email Spam Detector

---

ENRON Email Data

Brian Higgins - R00239570

# 1. Introduction

Spam emails have become a persistent and increasingly common annoyance in our digital world. These unsolicited emails can clog up our inboxes with large amounts of unwanted emails. The amount of daily spam emails changes regularly with anywhere from 40% to 70% [1] of all email traffic ranging from an estimated 316.39 billion in 2020 to 122 billion in 2021 [2]. Spam email can also contain malware, phishing links and other content intended to steal our personal information and compromise our computer security [3] [4].

This project will look at using Machine Learning Algorithms such as Random Forrest, Naïve Bayes, Gradient Boosting and Kmeans to take emails from the ENRON datasets and classify them into a Ham (safe) or Spam (unsolicited). Being able to accurately classify emails can save time and increase productivity, protect against security threats, improve email performance and enhance customer experience.

# 2. Data

## 2.1 Jupyter Notebook

A jupyter notebook was used to looked at different Feature Extraction Methods and Machine Learning Models, there is a detailed description with comments on all code used and a table of contents. It should be noted some code has been commented out so the notebook will not get stuck at these points and there is limited analysis as that is contained in this report.

## 2.2 Data Source

The data used for this project was taken from a collection of public-domain emails from the Enron Corporation. These emails have already been classified into ham, see Figure 1 and spam, Figure 2. There are six datasets online, three were taken for this project to have a large enough dataset to improve on models performance as using one dataset gave very high Accuracy results for earlier models, perhaps showing overfitting (will be discussed later)

	email	label
0	Subject: christmas tree farm pictures	ham
1	Subject: vastar resources , inc .\ngary , prod...	ham
2	Subject: calpine daily gas nomination\n- calpi...	ham
3	Subject: re : issue\nfyi - see note below - al...	ham
4	Subject: meter 7268 nov allocation\nfyi .\n- ...	ham

Figure 1: Emails classified as ham

	email	label
0	Subject: dobmeos with hgh my energy level has ...	spam
1	Subject: your prescription is ready . . oxwq s...	spam
2	Subject: get that new car 8434\npeople nowthe ...	spam
3	Subject: await your response\ndeare partner ,\n...	spam
4	Subject: coca cola , mbna america , nascar par...	spam

Figure 2: Emails classified as spam

Figure 3.1 shows a breakdown of the emails dataset where we can see 12,045 ham emails and 4,496 spam emails where the ham emails make up 72.8% of the total emails and spam makes up 27.2%. We have also seen that the average length of a ham email is 1,874 characters compares to 1,310 characters in spam emails. So Spam emails are shorter on average than ham emails. When we look at Figure 3.2 we see the range of email lengths is much greater for Ham than Spam emails.

**NOTE:** When only one dataset was used, ENRON1, the opposite was the case and Spam average emails were slightly longer so this depends on the datasets.

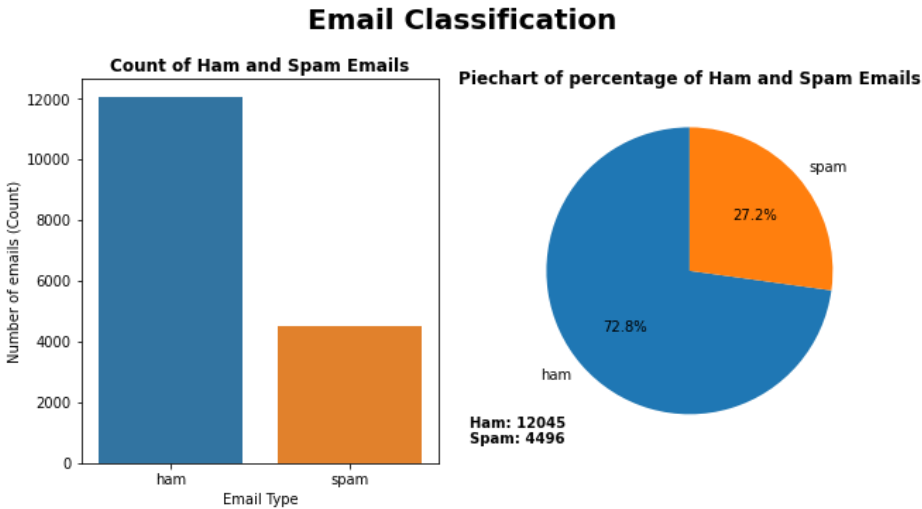


Figure 3.1: Email Classification

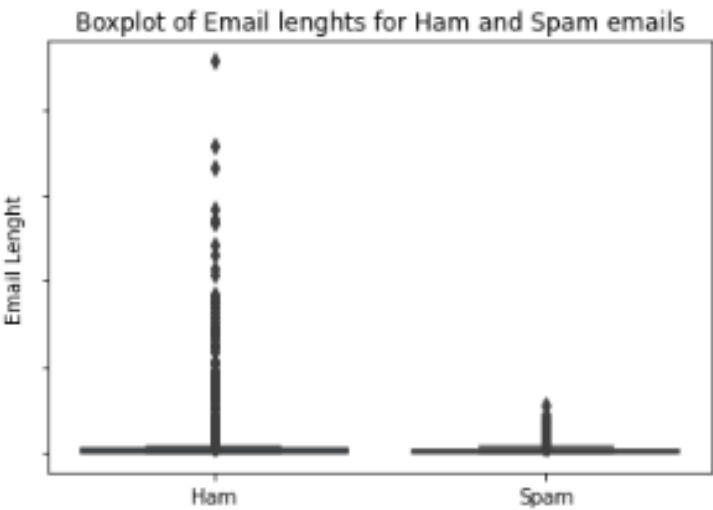


Figure 3.2: Email lengths

**Imbalanced data:**

What we can see is our data is imbalanced with a 3:1 ratio of Ham to Spam emails. This is something we will need to consider when we run our different models as this can affect the Accuracy Score (a common metric for vaulting a model's performance) as it can be misleading as a model can achieve high accuracy by predicting the majority class.

For this reason, when we are evaluating models we will also look at Precision and Recall metrics as well for more information. Precision is the fraction of correct positive predictions against all positive predictions and Recall is the fraction of all correct positive predictions against all actual positive instances. We can also use a confusion matrix to look at the prediction of actual and predicted class labels to see in the minority class is not predicted well.

Since we have seen in the introduction that the amount of spam can fluctuate, so we will keep this imbalance but in later projects, we will note this as something to change and run through our models again because if the dataset is balanced the models will not have any preference towards a specific class.

**2.1 Missing Data and Duplicate Data**

Not surprising there was no missing data, including any emails that just had an empty text, but there were 486 duplicates, these were dropped.

# 3. Pre-processing and EDA

## 3.1 Pre-processing

I had planned to do this at the beginning but I wanted to see how only a limited amount would affect the first methods of Feature Extraction. After I saw in some Word Clouds that a large amount of Parameters Symbols were showing up I then created a small function to remove these, the word “subject” (which was appearing a lot, note, I left in CC, BCC as I do not believe spam emails would have these and they did not appear in the top 20 words) and some other processing steps. It would be quite normal in a project to iterate through the different pipeline steps similar to how CRISMP-DM worked so I left this function in the section it is first used.

## 3.2 Python and Libraries

Python was used in this project with the following libraries:

# Load general libraries used	# NLP libraries, etc	# results, etc
<ul style="list-style-type: none"><li>• import pandas as pd</li><li>• import numpy as np</li><li>• import os</li><li>• import re</li><li>• import glob</li><li>• import matplotlib.pyplot as plt</li><li>• import seaborn as sns</li><li>• import spacy</li><li>• import string</li><li>• import pickle</li></ul>	<ul style="list-style-type: none"><li>• import nltk</li><li>• from nltk.tokenize import word_tokenize</li><li>• from collections import Counter</li><li>• from nltk.stem import SnowballStemmer</li><li>• from nltk.stem import PorterStemmer</li><li>• from nltk.corpus import stopwords</li><li>• from tqdm.notebook import tqdm as PROG_BAR</li><li>• from wordcloud import WordCloud</li><li>• from sklearn.feature_extraction.text import CountVectorizer</li><li>• from sklearn.feature_extraction.text import TfidfVectorizer</li><li>• from sklearn.model_selection import train_test_split</li></ul> <div># Models</div> <ul style="list-style-type: none"><li>• from sklearn.naive_bayes import MultinomialNB</li><li>• from sklearn.ensemble import RandomForestClassifier</li><li>• import sklearn.ensemble</li><li>• from sklearn.cluster import KMeans</li></ul>	<ul style="list-style-type: none"><li>• import sklearn</li><li>• from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score</li><li>• from sklearn.metrics import confusion_matrix</li><li>• from sklearn.metrics import precision_recall_curve</li><li>• from sklearn.metrics import silhouette_score, calinski_harabasz_score</li><li>• from sklearn.decomposition import PCA</li><li>• from scipy.sparse import csr_matrix</li><li>• from sklearn.manifold import TSNE</li><li>• from sklearn.model_selection import GridSearchCV</li></ul>

### 3.3 Splitting Data

We will be randomising and splitting the email data into three datasets into Train, Validation and test. Figure 4 shows a breakdown of how the three datasets make up and the percentage of ham and spam emails.

**Percentage breakdown of Spam and Ham emails for each dataset**

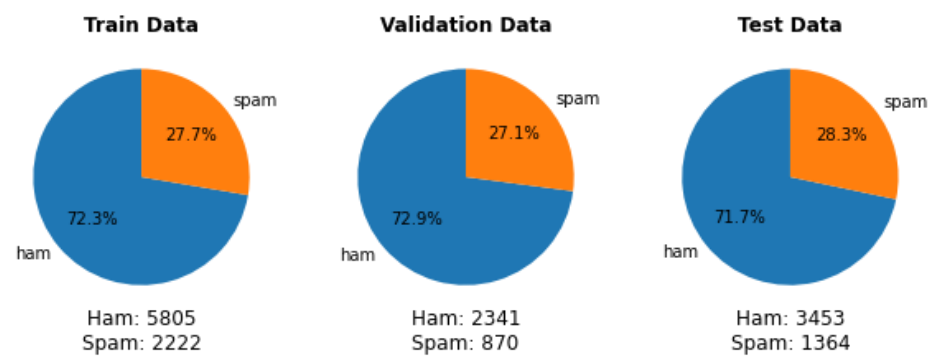


Figure 4: Train, Validation and Test dataset percentage breakdown

### 3.4 Machine Learning Models

A mixture of Machine Learning Models were used with Supervised Models; Naive Bayes, Random Forest, Gradient Boosting and Support Vector Machines and an Unsupervised Model with K-means. It's a little hard to compare these like for like but I wanted to compare how each Feature Extraction Method could change the results for each model.

#### Supervised Models:

- **Navie Bayes:** NN is a popular machine learning model for email ham and spam classification due to its effectiveness and simplicity. NN uses conditional probability to calculate the probability of an email being ham or spam based on the occurrences of specific words in an email. It was been shown to be very effective at email classifications.
- **Random Forest:** Random Forest can be a good choice for a Ham and Spam email classification that combines multiple decision trees to make accurate predictions as it works well with high-dimensional datasets. It also has a hyperparameter "class\_weight" that can be used to deal with a class imbalance which we have in our email data and we will explore more in the Tuning Section.
- **Support Vector Machine:** SVM uses a line or boundary that separates spam and ham messages. The line is chosen to maximise the margin or distance between the closest points from each class.

#### Unsupervised Model:

- **k-means:** Clustering can also be used for Email Detection. It is hard to compare an Unsupervised model against a Supervised model so this is looked at in a separate section briefly to compare how different Feature Extraction methods can affect results.

## 4. Feature Extraction

For this project we will be looking at several different Feature Extraction methods in order to explore how each one changes the text and what text features become more important, this becomes very clear with the use of Word Clouds and Bar Plots of top Ham and Spam words. We will then train and fit the different Machine Learning Models to compare.

**Feature Extraction methods used:**

- **Method 1:** Spacy Tokenization from the Labs
- **Method 2:** Basic Tokenization
- **Method 3:** Tokenization with Stop Word Removal
- **Method 4 :** Tokenization with Stop Word and Stemming
- **Method 5:** TF-IDF with Stop Words.
- Other methods can also be used such as Bag-of-words, Part of Speech (POS), Name entity recognition (NER) etc.

The following sections will follow the same steps:

- 1) Use a Feature Extraction, building on previous methods at times
- 2) Word Cloud to visualize how the words look like.
- 3) Barplot of the top 20 most popular words in Ham and Spam
- 4) Run Machine Learning Algorithms and investigate results.

## 4.1 Method 1: Spacy Tokenization

Spacy is an open-source NLP library that provides tools for tokenization, named entity recognition and other tools. It was seen in the labs and is one way to split up the emails into tokens and is the first method we used. We can see in Figure 5 a Word Cloud of this method, which shows quite several results, “ect” is mentioned more than once, “subject” which is the email heading, etc

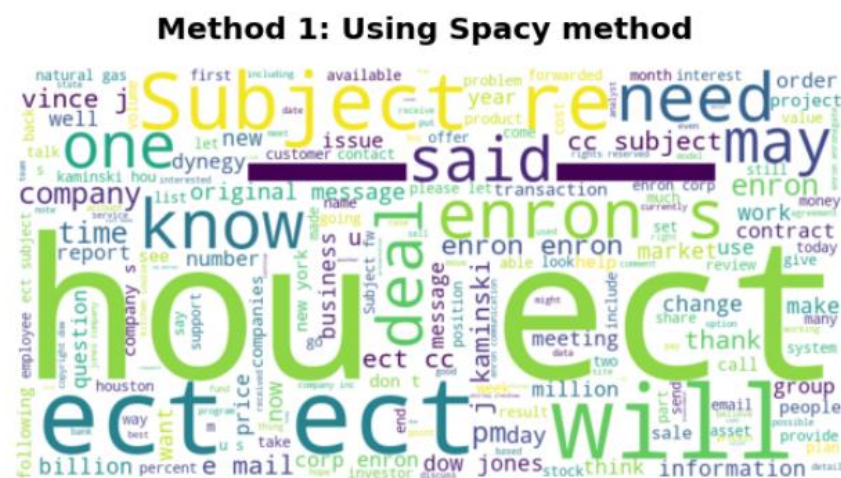
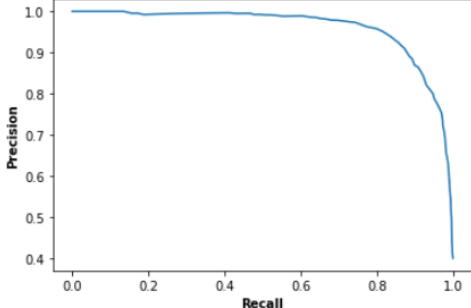
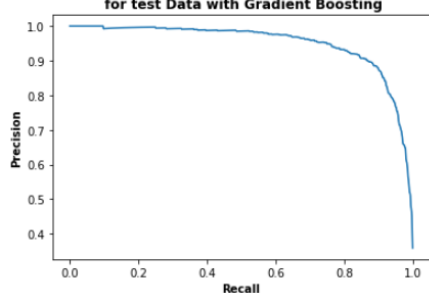
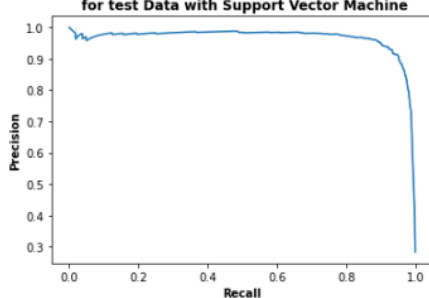


Figure 5: Word Cloud of Spacy Tokenization

Method 1: Spacy Tokenization																																																																																																								
Random Forest Model					Gradient Boosting Model					Support Vector Machine																																																																																														
Test Accuracy: 0.9333					Test Accuracy: 0.9304					Test Accuracy: 0.9591																																																																																														
Validation Data					Validation Data					Validation Data																																																																																														
<table><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr><tr><td>ham</td><td>0.93</td><td>0.98</td><td>0.96</td><td>2341</td></tr><tr><td>spam</td><td>0.94</td><td>0.81</td><td>0.87</td><td>870</td></tr><tr><td>accuracy</td><td></td><td></td><td>0.94</td><td>3211</td></tr><tr><td>macro avg</td><td>0.94</td><td>0.90</td><td>0.91</td><td>3211</td></tr><tr><td>weighted avg</td><td>0.94</td><td>0.94</td><td>0.93</td><td>3211</td></tr></table>						precision	recall	f1-score	support	ham	0.93	0.98	0.96	2341	spam	0.94	0.81	0.87	870	accuracy			0.94	3211	macro avg	0.94	0.90	0.91	3211	weighted avg	0.94	0.94	0.93	3211	<table><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr><tr><td>ham</td><td>0.94</td><td>0.97</td><td>0.95</td><td>2341</td></tr><tr><td>spam</td><td>0.92</td><td>0.82</td><td>0.87</td><td>870</td></tr><tr><td>accuracy</td><td></td><td></td><td>0.93</td><td>3211</td></tr><tr><td>macro avg</td><td>0.93</td><td>0.90</td><td>0.91</td><td>3211</td></tr><tr><td>weighted avg</td><td>0.93</td><td>0.93</td><td>0.93</td><td>3211</td></tr></table>						precision	recall	f1-score	support	ham	0.94	0.97	0.95	2341	spam	0.92	0.82	0.87	870	accuracy			0.93	3211	macro avg	0.93	0.90	0.91	3211	weighted avg	0.93	0.93	0.93	3211	<table><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr><tr><td>ham</td><td>0.98</td><td>0.97</td><td>0.97</td><td>2341</td></tr><tr><td>spam</td><td>0.92</td><td>0.94</td><td>0.93</td><td>870</td></tr><tr><td>accuracy</td><td></td><td></td><td>0.96</td><td>3211</td></tr><tr><td>macro avg</td><td>0.95</td><td>0.95</td><td>0.95</td><td>3211</td></tr><tr><td>weighted avg</td><td>0.96</td><td>0.96</td><td>0.96</td><td>3211</td></tr></table>						precision	recall	f1-score	support	ham	0.98	0.97	0.97	2341	spam	0.92	0.94	0.93	870	accuracy			0.96	3211	macro avg	0.95	0.95	0.95	3211	weighted avg	0.96	0.96	0.96	3211
	precision	recall	f1-score	support																																																																																																				
ham	0.93	0.98	0.96	2341																																																																																																				
spam	0.94	0.81	0.87	870																																																																																																				
accuracy			0.94	3211																																																																																																				
macro avg	0.94	0.90	0.91	3211																																																																																																				
weighted avg	0.94	0.94	0.93	3211																																																																																																				
	precision	recall	f1-score	support																																																																																																				
ham	0.94	0.97	0.95	2341																																																																																																				
spam	0.92	0.82	0.87	870																																																																																																				
accuracy			0.93	3211																																																																																																				
macro avg	0.93	0.90	0.91	3211																																																																																																				
weighted avg	0.93	0.93	0.93	3211																																																																																																				
	precision	recall	f1-score	support																																																																																																				
ham	0.98	0.97	0.97	2341																																																																																																				
spam	0.92	0.94	0.93	870																																																																																																				
accuracy			0.96	3211																																																																																																				
macro avg	0.95	0.95	0.95	3211																																																																																																				
weighted avg	0.96	0.96	0.96	3211																																																																																																				
Validation Data					Validation Data					Validation Data																																																																																														
<table><tr><th>Predicted</th><th>ham</th><th>spam</th><th>All</th></tr><tr><td>True</td><td></td><td></td><td></td></tr><tr><td>ham</td><td>2299</td><td>42</td><td>2341</td></tr><tr><td>spam</td><td>164</td><td>706</td><td>870</td></tr><tr><td>All</td><td>2463</td><td>748</td><td>3211</td></tr></table>					Predicted	ham	spam	All	True				ham	2299	42	2341	spam	164	706	870	All	2463	748	3211	<table><tr><th>Predicted</th><th>ham</th><th>spam</th><th>All</th></tr><tr><td>True</td><td></td><td></td><td></td></tr><tr><td>ham</td><td>2279</td><td>62</td><td>2341</td></tr><tr><td>spam</td><td>155</td><td>715</td><td>870</td></tr><tr><td>All</td><td>2434</td><td>777</td><td>3211</td></tr></table>					Predicted	ham	spam	All	True				ham	2279	62	2341	spam	155	715	870	All	2434	777	3211	<table><tr><th>Predicted</th><th>ham</th><th>spam</th><th>All</th></tr><tr><td>True</td><td></td><td></td><td></td></tr><tr><td>ham</td><td>2273</td><td>68</td><td>2341</td></tr><tr><td>spam</td><td>56</td><td>814</td><td>870</td></tr><tr><td>All</td><td>2329</td><td>882</td><td>3211</td></tr></table>					Predicted	ham	spam	All	True				ham	2273	68	2341	spam	56	814	870	All	2329	882	3211																														
Predicted	ham	spam	All																																																																																																					
True																																																																																																								
ham	2299	42	2341																																																																																																					
spam	164	706	870																																																																																																					
All	2463	748	3211																																																																																																					
Predicted	ham	spam	All																																																																																																					
True																																																																																																								
ham	2279	62	2341																																																																																																					
spam	155	715	870																																																																																																					
All	2434	777	3211																																																																																																					
Predicted	ham	spam	All																																																																																																					
True																																																																																																								
ham	2273	68	2341																																																																																																					
spam	56	814	870																																																																																																					
All	2329	882	3211																																																																																																					
Test Data					Test Data					Test Data																																																																																														
<table><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr><tr><td>ham</td><td>0.93</td><td>0.99</td><td>0.95</td><td>3453</td></tr><tr><td>spam</td><td>0.96</td><td>0.80</td><td>0.87</td><td>1364</td></tr><tr><td>accuracy</td><td></td><td></td><td>0.93</td><td>4817</td></tr><tr><td>macro avg</td><td>0.94</td><td>0.89</td><td>0.91</td><td>4817</td></tr><tr><td>weighted avg</td><td>0.93</td><td>0.93</td><td>0.93</td><td>4817</td></tr></table>						precision	recall	f1-score	support	ham	0.93	0.99	0.95	3453	spam	0.96	0.80	0.87	1364	accuracy			0.93	4817	macro avg	0.94	0.89	0.91	4817	weighted avg	0.93	0.93	0.93	4817	<table><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr><tr><td>ham</td><td>0.93</td><td>0.97</td><td>0.95</td><td>3453</td></tr><tr><td>spam</td><td>0.92</td><td>0.83</td><td>0.87</td><td>1364</td></tr><tr><td>accuracy</td><td></td><td></td><td>0.93</td><td>4817</td></tr><tr><td>macro avg</td><td>0.93</td><td>0.90</td><td>0.91</td><td>4817</td></tr><tr><td>weighted avg</td><td>0.93</td><td>0.93</td><td>0.93</td><td>4817</td></tr></table>						precision	recall	f1-score	support	ham	0.93	0.97	0.95	3453	spam	0.92	0.83	0.87	1364	accuracy			0.93	4817	macro avg	0.93	0.90	0.91	4817	weighted avg	0.93	0.93	0.93	4817	<table><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr><tr><td>ham</td><td>0.97</td><td>0.97</td><td>0.97</td><td>3453</td></tr><tr><td>spam</td><td>0.93</td><td>0.92</td><td>0.93</td><td>1364</td></tr><tr><td>accuracy</td><td></td><td></td><td>0.96</td><td>4817</td></tr><tr><td>macro avg</td><td>0.95</td><td>0.95</td><td>0.95</td><td>4817</td></tr><tr><td>weighted avg</td><td>0.96</td><td>0.96</td><td>0.96</td><td>4817</td></tr></table>						precision	recall	f1-score	support	ham	0.97	0.97	0.97	3453	spam	0.93	0.92	0.93	1364	accuracy			0.96	4817	macro avg	0.95	0.95	0.95	4817	weighted avg	0.96	0.96	0.96	4817
	precision	recall	f1-score	support																																																																																																				
ham	0.93	0.99	0.95	3453																																																																																																				
spam	0.96	0.80	0.87	1364																																																																																																				
accuracy			0.93	4817																																																																																																				
macro avg	0.94	0.89	0.91	4817																																																																																																				
weighted avg	0.93	0.93	0.93	4817																																																																																																				
	precision	recall	f1-score	support																																																																																																				
ham	0.93	0.97	0.95	3453																																																																																																				
spam	0.92	0.83	0.87	1364																																																																																																				
accuracy			0.93	4817																																																																																																				
macro avg	0.93	0.90	0.91	4817																																																																																																				
weighted avg	0.93	0.93	0.93	4817																																																																																																				
	precision	recall	f1-score	support																																																																																																				
ham	0.97	0.97	0.97	3453																																																																																																				
spam	0.93	0.92	0.93	1364																																																																																																				
accuracy			0.96	4817																																																																																																				
macro avg	0.95	0.95	0.95	4817																																																																																																				
weighted avg	0.96	0.96	0.96	4817																																																																																																				
Test Data					Test Data					Test Data																																																																																														
<table><tr><th>Predicted</th><th>ham</th><th>spam</th><th>All</th></tr><tr><td>True</td><td></td><td></td><td></td></tr><tr><td>ham</td><td>3405</td><td>48</td><td>3453</td></tr><tr><td>spam</td><td>273</td><td>1091</td><td>1364</td></tr><tr><td>All</td><td>3678</td><td>1139</td><td>4817</td></tr></table>					Predicted	ham	spam	All	True				ham	3405	48	3453	spam	273	1091	1364	All	3678	1139	4817	<table><tr><th>Predicted</th><th>ham</th><th>spam</th><th>All</th></tr><tr><td>True</td><td></td><td></td><td></td></tr><tr><td>ham</td><td>3355</td><td>98</td><td>3453</td></tr><tr><td>spam</td><td>237</td><td>1127</td><td>1364</td></tr><tr><td>All</td><td>3592</td><td>1225</td><td>4817</td></tr></table>					Predicted	ham	spam	All	True				ham	3355	98	3453	spam	237	1127	1364	All	3592	1225	4817	<table><tr><th>Predicted</th><th>ham</th><th>spam</th><th>All</th></tr><tr><td>True</td><td></td><td></td><td></td></tr><tr><td>ham</td><td>3360</td><td>93</td><td>3453</td></tr><tr><td>spam</td><td>104</td><td>1260</td><td>1364</td></tr><tr><td>All</td><td>3464</td><td>1353</td><td>4817</td></tr></table>					Predicted	ham	spam	All	True				ham	3360	93	3453	spam	104	1260	1364	All	3464	1353	4817																														
Predicted	ham	spam	All																																																																																																					
True																																																																																																								
ham	3405	48	3453																																																																																																					
spam	273	1091	1364																																																																																																					
All	3678	1139	4817																																																																																																					
Predicted	ham	spam	All																																																																																																					
True																																																																																																								
ham	3355	98	3453																																																																																																					
spam	237	1127	1364																																																																																																					
All	3592	1225	4817																																																																																																					
Predicted	ham	spam	All																																																																																																					
True																																																																																																								
ham	3360	93	3453																																																																																																					
spam	104	1260	1364																																																																																																					
All	3464	1353	4817																																																																																																					
Method 1: Precision Recall Plot with Random Forest					Method 1: Precision Recall Plot for test Data with Gradient Boosting					Method 1: Precision Recall Plot for test Data with Support Vector Machine																																																																																														
																																																																																																								

**Results:**

- Random Forest Accuracy results of 93.33%, Gradient Boost 93.05% and SVM with 95.91% so SVM have performed that best. Accuracy alone can be misleading, as it can be misleading in the case of class imbalance.
- The RF model has the lowest number of false positives but the confusion matrix of each model tells us more.
- RF has the highest number of false positives (273) compared to SVM (104) and GB (237) models. So the RF model is incorrectly classifying more ham messages than spam messages which could be a sign of overfitting on the training data. So based on the Confusion matrix alone the SVM looks like it is performing better.
- Accuracy alone may not be the best measure of a model's performance, other metrics such as precision, recall and F1 score can also be useful.
  - For the SVM the recall for spam is 0.92, whereas the recall for RF is 0.83 and for GB it's 0.83. So the Recall for SVM is highest.
  - For F1 scores, again SVM has the highest scores and outperforms RF and GB.
- The Precision Recall Plots show the SVM model is also performing the best. Although it has an initial "hiccup" it then gives the best plot of the three models as it stays closer to the top right hand corner with a more abrupt drop.
- The spacy method does a lot in one go, but the Word Cloud does not seem so insightful and seems to pick up some weird results. I will take a step back using basic Tokenization in the next method and build from that to other Feature Extraction methods to see how this improves our model performances.

4.2 Method 2: Basic Tokenization

Count Vectorization is the first basic feature extraction method we will use with Natural Language Processing to convert a collection of text documents into a matrix of token counts, this is a way to represent text data as numerical data so that the Machine Learning Algorithms can use text. CountVectorizer first breaks the text data into words or tokens and then creates a vocabulary of all the unique words. Finally, it counts the number of times each word appears in a document and creates a sparse matrix.

We will use the Train data as a comparison for each Method so that we can see how each Feature Extraction can change the number of columns and stored elements. The number of rows will always stay the same, and the Columns of tokens created and stored elements will change. We will also look at "Sparsity" which tells us what the proportion of zero values is in the matrix and also "Density" which is the opposite of Sparsity and is the proportion of non-zero elements.

Table 1 shows us our first Table with 67,336 columns created, with 973,073 non-zero elements with 0.18% of the elements in the matrix being non-zero. The majority of the entries are zero, but this is to be expected in a sparse matrix.

Method 2	Rows	Columns	Stored Elements	Density
Train Data:	8,270	67,336	973,073	0.18003%

Table 1: Count Vectorizer Table



Figure 6 shows us a Word Cloud of the frequency of the most common words, what we can see is that the most common are “the”, “and”, “of”, etc. It is not surprising these are the most common words in both Ham and Spam emails but they do not provide much information about the content of the emails. We will do further processing in later methods.

## Method 2: Word Count for Basic CountVectorizer

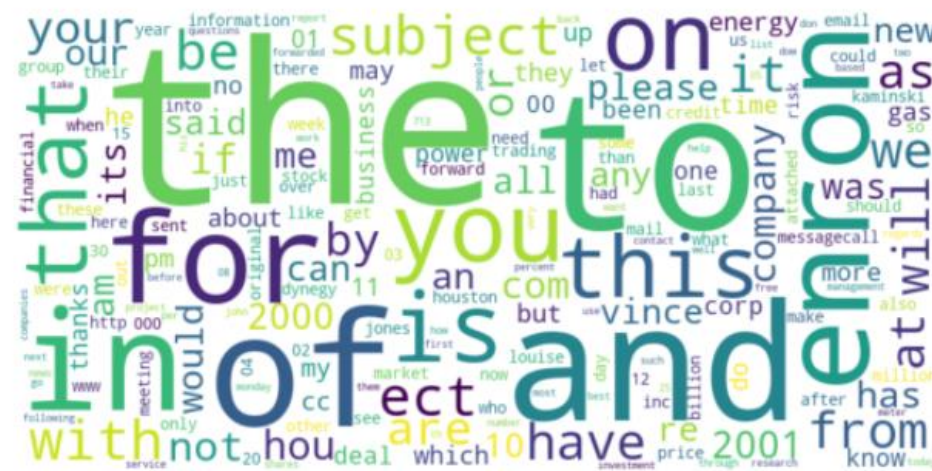


Figure 6: Count Vectorizer Word Cloud

Figure 7 shows us the top 20 words for Ham and Spam emails and looks very similar to the Word Cloud. Also, the top 20 words are mostly the same, which does not help in uniquely identifying emails as either ham or spam. These are “Stop Words” and we will remove these in the next Feature Extraction Method.

## Top Words for Ham and Spam emails with Basic Tokenization

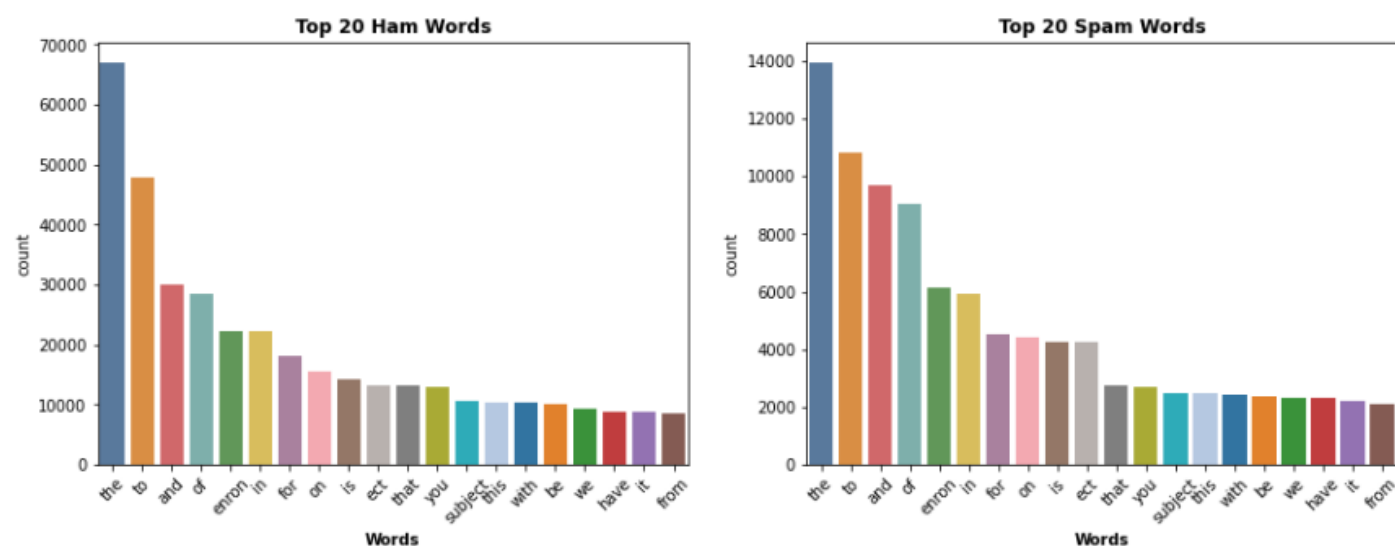
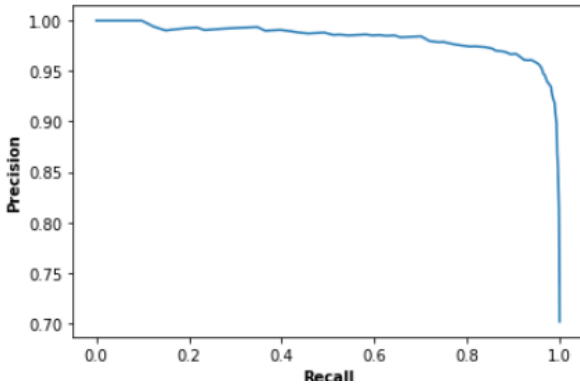
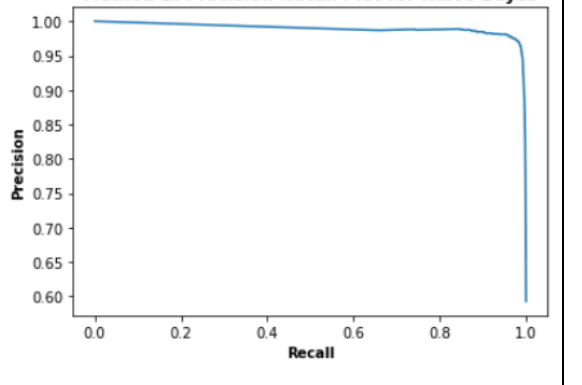
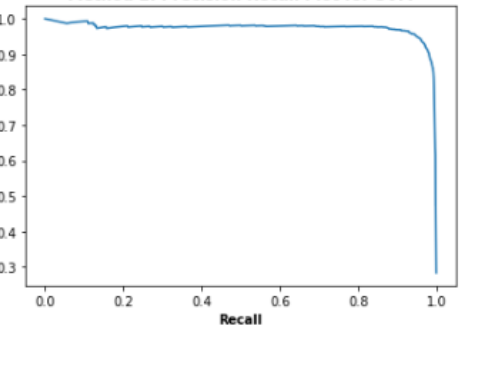


Figure 7: Count Vectorizer top 20 words Ham and Spam

Method 2: Basic Tokenization														
Random Forest Model					Naïve Bayes Model					Support Vector Machines				
Test Accuracy: 0.96					Test Accuracy: 0.9613					Test Accuracy: 0.97				
Validation Data					Validation Data					Validation Data				
	precision	recall	f1-score	support		precision	recall	f1-score	support		precision	recall	f1-score	support
ham	0.98	0.99	0.98	2341	ham	0.99	0.99	0.99	2341	ham	0.99	0.98	0.98	2341
spam	0.96	0.93	0.95	870	spam	0.97	0.98	0.98	870	spam	0.95	0.97	0.96	870
accuracy			0.97	3211	accuracy			0.99	3211	accuracy			0.98	3211
macro avg	0.97	0.96	0.96	3211	macro avg	0.98	0.99	0.99	3211	macro avg	0.97	0.97	0.97	3211
weighted avg	0.97	0.97	0.97	3211	weighted avg	0.99	0.99	0.99	3211	weighted avg	0.98	0.98	0.98	3211
Validation Data					Validation Data					Validation Data				
	Predicted	ham	spam	All		Predicted	ham	spam	All		Predicted	ham	spam	All
	True					True					True			
ham	2309	32	2341		ham	2319	22	2341		ham	2298	43	2341	
spam	58	812	870		spam	14	856	870		spam	30	840	870	
All	2367	844	3211		All	2333	878	3211		All	2328	883	3211	
Test Data					Test Data					Test Data				
	precision	recall	f1-score	support		precision	recall	f1-score	support		precision	recall	f1-score	support
ham	0.96	0.99	0.97	3453	ham	0.96	0.99	0.97	3453	ham	0.98	0.98	0.98	3453
spam	0.97	0.89	0.93	1364	spam	0.97	0.89	0.93	1364	spam	0.95	0.96	0.95	1364
accuracy			0.96	4817	accuracy			0.96	4817	accuracy			0.97	4817
macro avg	0.96	0.94	0.95	4817	macro avg	0.96	0.94	0.95	4817	macro avg	0.96	0.97	0.97	4817
weighted avg	0.96	0.96	0.96	4817	weighted avg	0.96	0.96	0.96	4817	weighted avg	0.97	0.97	0.97	4817
Test Data					Test Data					Test Data				
	Predicted	ham	spam	All		Predicted	ham	spam	All		Predicted	ham	spam	All
	True					True					True			
ham	3419	34	3453		ham	3411	42	3453		ham	3378	75	3453	
spam	39	1325	1364		spam	144	1220	1364		spam	61	1303	1364	
All	3458	1359	4817		All	3555	1262	4817		All	3439	1378	4817	
Method 2: Precision Recall Plot for Random Forest					Method 2: Precision Recall Plot for Naive Bayes					Method 2: Precision Recall Plot for SVM				
														

**Results:**

- The Accuracy results for all three models is very high. Too high. Since this is basic Tokenization its highly likely the models are overfitting. The confusion Matrices and the Precision Recall Plots are also too good considering I am only using the first of the Feature Extraction methods.

**Dealing with overfitting:**

In an attempt to deal with the likely overfitting, I tried some steps:

- 1) Increased the amount of data from 1 Enron file to 3 Enron files which increased the size of the training data. No change was seen.
- 2) As we have seen the Ham and Spam labels are imbalanced so I tested some models with a 50:50 balance of the labels. While some different results were seen, not enough to believe the issue was resolved.
- 3) Cross Validation was used in Section 5 when tuning of the models was performed. This I believe gave more accurate model results.

**NOTE: I wanted to continue to explore Feature Extraction methods so the Cross Validation was kept for the Tunning Section and so the Model results for the various Feature Extraction Methods are likely overfitting. If I was restarting the project I would have taken Cross Validation into account earlier for each for the Feature Extraction Methods I tested.**

**4.3 Method 3: Tokenization with Stop Words**

What we saw with the first method with just Count Vectorization was that we have a very large amount of Stop Words in both the Word Cloud and showing up on the two bar plots of the top 20 words of Ham and Spam. So our next method will remove these and we will compare the same results again.

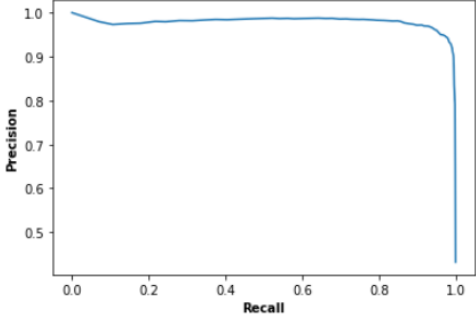
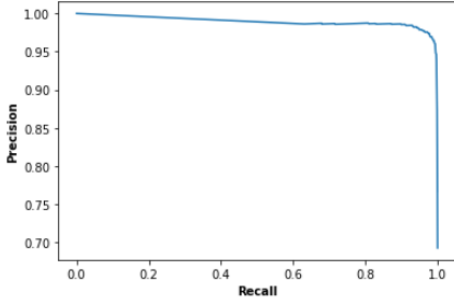
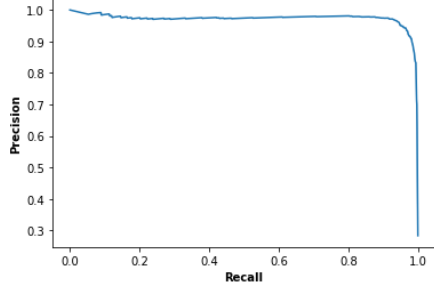
**What are Stop Words?** We will be using the “English” list of stop words that are included with the module scikit-learn and we can see some examples in Figure 8. This is a list of words that includes items like “the”, “an”, and “do”, etc as seen below.

```
frozenset({'fifteen', 'both', 'first', 'describe', 'from', 'whoever', 'off', 'become', 'due', 'nothin  
g', 'herein', 'several', 'before', 'than', 'nobody', 'always', 'thin', 'ltd', 'can', 'neither', 'tw  
o', 'others', 'do', 'anyway', 'its', 'co', 'yourself', 'own', 'whence', 'either', 'wherever', 'five',  
'our', 'he', 'inc', 'am', 'such', 'amount', 'hereafter', 'hundred', 'had', 'themselves', 'whither',  
'part', 'seemed', 'besides', 'thereafter', 'except', 'my', 'whereby', 'under', 'front', 'something',  
'upon', 'hence', 'found', 'already', 'been', 'anywhere', 'eleven', 'de', 'meanwhile', 'otherwise', 'b  
ack', 'also', 'six', 'once', 'another', 'even', 'after', 'together', 'now', 'few', 'interest', 'thei  
r', 'should', 'they', 'least', 'you', 'becomes', 'more', 'anyone', 'formerly', 'nine', 'latter', 'tha  
t', 'on', 'forty', 'him', 'have', 'her', 'therefore', 'seem', 'will', 'some', 'thus', 'onto', 'becomi  
ng', 'bill', 'herself', 'move', 'ourselves', 'whom', 'four', 'serious', 'yet', 'somehow', 'who', 'bet  
ween', 'everyone', 'bottom', 'sincere', 'cannot', 'without', 'everywhere', 'perhaps', 'please', 'fil
```

Figure 8: Print out of Stop Words from the “English” list.

Table 2 shows a look at our columns after we do a Count Vectorizer using Stop Words, what we see is the number of columns has not dropped by much, only 210 columns. However, the stored elements was drop from 970k to 770k. A significant drop and we the density values also drop to 0.14%.

Method 3:	Rows	Columns	Stored Elements	Density
Train Data:	8,270	67,027	725,852	0.13491%

Method 3: Tokenization with Stop Words														
Random Forest Model					Naïve Bayes Model					Support Vector Machines				
Test Accuracy: 0.9746					Test Accuracy: 0.97					Test Accuracy: 0.97				
Validation Data					Validation Data					Validation Data				
	precision	recall	f1-score	support		precision	recall	f1-score	support		precision	recall	f1-score	support
ham	0.99	0.98	0.98	2341	ham	0.99	0.99	0.99	2341	ham	0.99	0.98	0.98	2341
spam	0.94	0.98	0.96	870	spam	0.97	0.98	0.98	870	spam	0.95	0.97	0.96	870
accuracy			0.98	3211	accuracy			0.99	3211	accuracy			0.98	3211
macro avg	0.97	0.98	0.97	3211	macro avg	0.98	0.99	0.98	3211	macro avg	0.97	0.97	0.97	3211
weighted avg	0.98	0.98	0.98	3211	weighted avg	0.99	0.99	0.99	3211	weighted avg	0.98	0.98	0.98	3211
Validation Data					Validation Data					Validation Data				
	Predicted	ham	spam	All		Predicted	ham	spam	All		Predicted	ham	spam	All
True					True					True				
ham	2290	51	2341		ham	2317	24	2341		ham	2292	49	2341	
spam	20	850	870		spam	17	853	870		spam	28	842	870	
All	2310	901	3211		All	2334	877	3211		All	2320	891	3211	
Test Data					Test Data					Test Data				
	precision	recall	f1-score	support		precision	recall	f1-score	support		precision	recall	f1-score	support
ham	0.98	0.98	0.98	3453	ham	0.98	0.98	0.98	3453	ham	0.98	0.98	0.98	3453
spam	0.95	0.96	0.96	1364	spam	0.95	0.96	0.96	1364	spam	0.95	0.96	0.95	1364
accuracy			0.97	4817	accuracy			0.97	4817	accuracy			0.97	4817
macro avg	0.97	0.97	0.97	4817	macro avg	0.97	0.97	0.97	4817	macro avg	0.96	0.97	0.97	4817
weighted avg	0.97	0.97	0.97	4817	weighted avg	0.97	0.97	0.97	4817	weighted avg	0.97	0.97	0.97	4817
Test Data					Test Data					Test Data				
	Predicted	ham	spam	All		Predicted	ham	spam	All		Predicted	ham	spam	All
True					True					True				
ham	3384	69	3453		ham	3418	35	3453		ham	3377	76	3453	
spam	53	1311	1364		spam	36	1328	1364		spam	56	1308	1364	
All	3437	1380	4817		All	3454	1363	4817		All	3433	1384	4817	
Method 3: Precision Recall Plot Random Forest					Method 3: Precision Recall Plot Naïve Bayes					Method 3: Precision Recall Plot SVM				
														



## Results

- As we have seen the results are still overfitting but if we look past that for now as already mentioned we can look at how this method of feature extraction compares to method one.
- We can see that for all three models we get better results so the use of Stop Words does improve the accuracy of our models which makes sense as the words shown in the word cloud in Figure 9 and the top 20 words of ham and spam in figure 10 are much more interesting.
- Let's continue with more Feature Extraction methods to see how they perform.

#### 4.4 Method 4: Stemming

Stemming is the process of reducing words to its root base form, which is a stem. So "running", "ran" and "runner" are reduced to "run. Stemming can be used to improve the efficiency of text analysis by grouping together words of similar meaning which reduces the size of the vocabulary that needs to be processed which then improves the accuracy of text classification. Let's see how it compares to Method 1 and Method 2. We will also build on previous methods so will continue to use Stop Words.

Table 3 shows using Stemming the number of columns has dropped from 67,000 to 54,000, a significant decrease. The stored elements has increased and so the density has also increased.

Method 4:	Rows	Columns	Stored Elements	Density
Train Data:	8,270	54, 324	794,519	0.182205%

Table 3: Count Vectorizer with Stop Words and Stemming

Figure 11 shows the word count, since we have now used stemming it has broken up words which has left a lot of character punctuation marks in the Word Cloud. This is matched in the Top 20 words for ham and Spam which has a very large amount of punctuation symbols. This looks like a step back from our last method so we will use this point to do some pre-processing on the emails to remove some items.

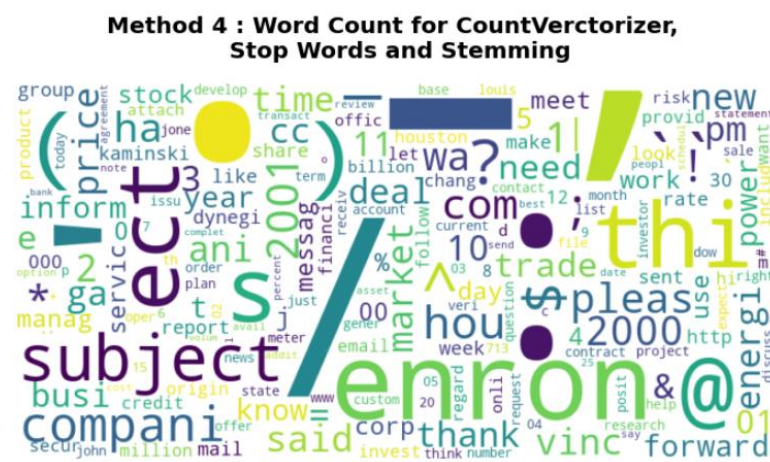


Figure 11: Count Vectorizer Word Cloud

Top Words for Ham and Spam emails with Tokenization,Stop Words and Stem Words

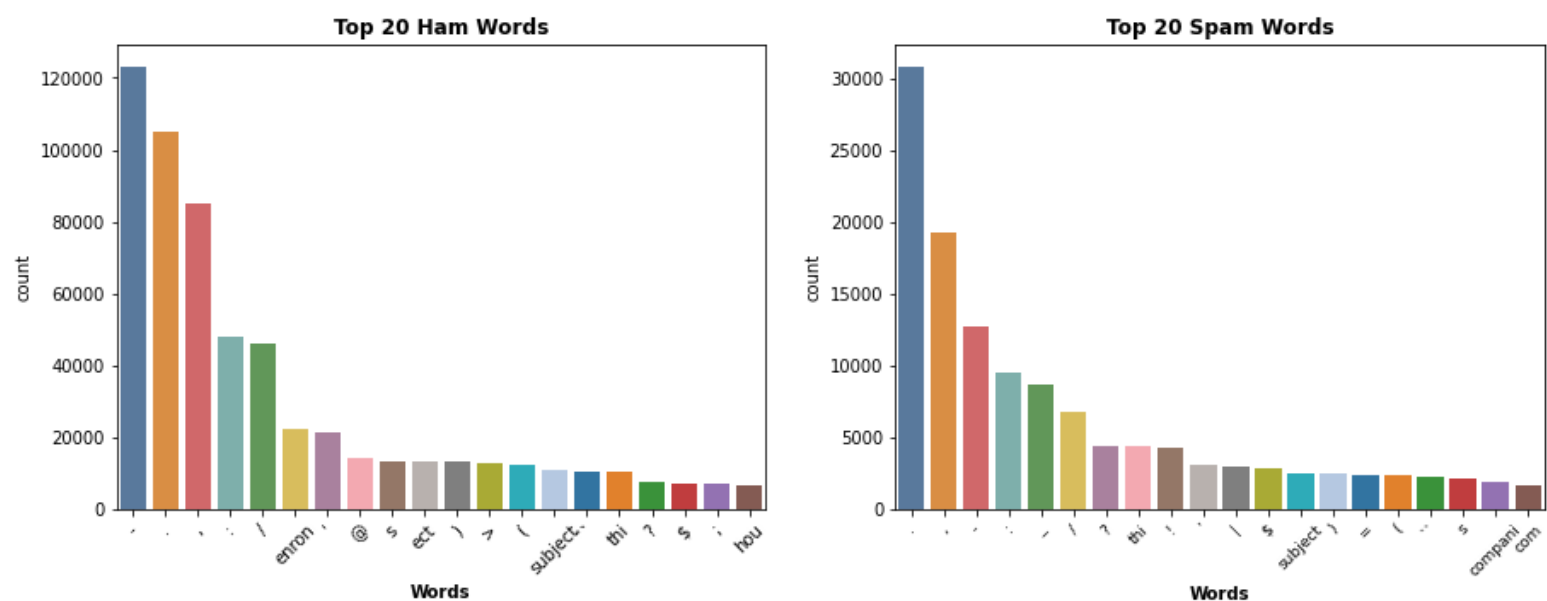


Figure 12: Count Vectorizer with Stop Words top 20 words Ham and Spam

Text Processing

We mentioned earlier we wanted to see how issues would come up we could solve during the exploration of the Feature Extraction methods and so at this point we created a function to make all characters lower case, delete the token “subject” since it is not helpful and also removed punctuation marks. We then ran Method 4 again with Stop Words and Stemming and we get table 4. We can see that the number of columns only dropped a small amount but the stored elements and density scores dropped a lot.

Method 4:	Rows	Columns	Stored Elements	Density
Version 2				
Train Data:	8,270	54,7002	731,247	0.1598%

Table 4: Count Vectorizer with Stop Words and Stemming version 2

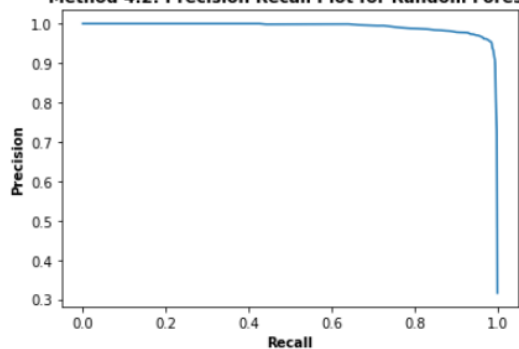
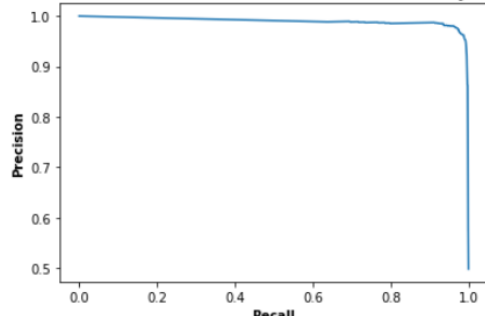
Figure 13 now shows us a new Word Cloud that looks better as it has dropped the punctuation symbols. We notice that the @ symbol is still here as we excluded it from being removed as we felt like it would still have some use when identifying emails. For example, a spam email might just have one email, whereas a ham email could have a number of ccs, bcc and other email addresses.

Similarly, Figure 14 shows the top 20 words in ham and spam emails and also shows improved identifiers that have more meaning and will help improve on our models from method 3.

### Method 4.2: Top Words for Ham and Spam emails with Tokenization, Stop Words and Stem Words Version 2





Method 4.2: Tokenization, Stop Words and Stemming																																																													
Random Forest Model	Naïve Bayes Model																																																												
Test Accuracy: 0.975	Test Accuracy: 0.9854																																																												
<div>Validation Data</div> <table><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr><tr><td>ham</td><td>0.98</td><td>0.99</td><td>0.99</td><td>2321</td></tr><tr><td>spam</td><td>0.98</td><td>0.95</td><td>0.97</td><td>890</td></tr><tr><td>accuracy</td><td></td><td></td><td>0.98</td><td>3211</td></tr><tr><td>macro avg</td><td>0.98</td><td>0.97</td><td>0.98</td><td>3211</td></tr><tr><td>weighted avg</td><td>0.98</td><td>0.98</td><td>0.98</td><td>3211</td></tr></table>		precision	recall	f1-score	support	ham	0.98	0.99	0.99	2321	spam	0.98	0.95	0.97	890	accuracy			0.98	3211	macro avg	0.98	0.97	0.98	3211	weighted avg	0.98	0.98	0.98	3211	<div>Validation Data</div> <table><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr><tr><td>ham</td><td>0.99</td><td>0.99</td><td>0.99</td><td>2321</td></tr><tr><td>spam</td><td>0.98</td><td>0.97</td><td>0.98</td><td>890</td></tr><tr><td>accuracy</td><td></td><td></td><td>0.99</td><td>3211</td></tr><tr><td>macro avg</td><td>0.98</td><td>0.98</td><td>0.98</td><td>3211</td></tr><tr><td>weighted avg</td><td>0.99</td><td>0.99</td><td>0.99</td><td>3211</td></tr></table>		precision	recall	f1-score	support	ham	0.99	0.99	0.99	2321	spam	0.98	0.97	0.98	890	accuracy			0.99	3211	macro avg	0.98	0.98	0.98	3211	weighted avg	0.99	0.99	0.99	3211
	precision	recall	f1-score	support																																																									
ham	0.98	0.99	0.99	2321																																																									
spam	0.98	0.95	0.97	890																																																									
accuracy			0.98	3211																																																									
macro avg	0.98	0.97	0.98	3211																																																									
weighted avg	0.98	0.98	0.98	3211																																																									
	precision	recall	f1-score	support																																																									
ham	0.99	0.99	0.99	2321																																																									
spam	0.98	0.97	0.98	890																																																									
accuracy			0.99	3211																																																									
macro avg	0.98	0.98	0.98	3211																																																									
weighted avg	0.99	0.99	0.99	3211																																																									
<div>Validation Data</div> <table><tr><th>Predicted</th><th>ham</th><th>spam</th><th>All</th></tr><tr><td>True</td><td></td><td></td><td></td></tr><tr><td>ham</td><td>2304</td><td>17</td><td>2321</td></tr><tr><td>spam</td><td>41</td><td>849</td><td>890</td></tr><tr><td>All</td><td>2345</td><td>866</td><td>3211</td></tr></table>	Predicted	ham	spam	All	True				ham	2304	17	2321	spam	41	849	890	All	2345	866	3211	<div>Validation Data</div> <table><tr><th>Predicted</th><th>ham</th><th>spam</th><th>All</th></tr><tr><td>True</td><td></td><td></td><td></td></tr><tr><td>ham</td><td>2304</td><td>17</td><td>2321</td></tr><tr><td>spam</td><td>27</td><td>863</td><td>890</td></tr><tr><td>All</td><td>2331</td><td>880</td><td>3211</td></tr></table>	Predicted	ham	spam	All	True				ham	2304	17	2321	spam	27	863	890	All	2331	880	3211																				
Predicted	ham	spam	All																																																										
True																																																													
ham	2304	17	2321																																																										
spam	41	849	890																																																										
All	2345	866	3211																																																										
Predicted	ham	spam	All																																																										
True																																																													
ham	2304	17	2321																																																										
spam	27	863	890																																																										
All	2331	880	3211																																																										
<div>Test Data</div> <table><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr><tr><td>ham</td><td>0.98</td><td>0.99</td><td>0.98</td><td>3460</td></tr><tr><td>spam</td><td>0.97</td><td>0.95</td><td>0.96</td><td>1356</td></tr><tr><td>accuracy</td><td></td><td></td><td>0.98</td><td>4816</td></tr><tr><td>macro avg</td><td>0.98</td><td>0.97</td><td>0.97</td><td>4816</td></tr><tr><td>weighted avg</td><td>0.98</td><td>0.98</td><td>0.98</td><td>4816</td></tr></table>		precision	recall	f1-score	support	ham	0.98	0.99	0.98	3460	spam	0.97	0.95	0.96	1356	accuracy			0.98	4816	macro avg	0.98	0.97	0.97	4816	weighted avg	0.98	0.98	0.98	4816	<div>Test Data</div> <table><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr><tr><td>ham</td><td>0.99</td><td>0.99</td><td>0.99</td><td>3460</td></tr><tr><td>spam</td><td>0.98</td><td>0.97</td><td>0.97</td><td>1356</td></tr><tr><td>accuracy</td><td></td><td></td><td>0.99</td><td>4816</td></tr><tr><td>macro avg</td><td>0.98</td><td>0.98</td><td>0.98</td><td>4816</td></tr><tr><td>weighted avg</td><td>0.99</td><td>0.99</td><td>0.99</td><td>4816</td></tr></table>		precision	recall	f1-score	support	ham	0.99	0.99	0.99	3460	spam	0.98	0.97	0.97	1356	accuracy			0.99	4816	macro avg	0.98	0.98	0.98	4816	weighted avg	0.99	0.99	0.99	4816
	precision	recall	f1-score	support																																																									
ham	0.98	0.99	0.98	3460																																																									
spam	0.97	0.95	0.96	1356																																																									
accuracy			0.98	4816																																																									
macro avg	0.98	0.97	0.97	4816																																																									
weighted avg	0.98	0.98	0.98	4816																																																									
	precision	recall	f1-score	support																																																									
ham	0.99	0.99	0.99	3460																																																									
spam	0.98	0.97	0.97	1356																																																									
accuracy			0.99	4816																																																									
macro avg	0.98	0.98	0.98	4816																																																									
weighted avg	0.99	0.99	0.99	4816																																																									
<div>Test Data</div> <table><tr><th>Predicted</th><th>ham</th><th>spam</th><th>All</th></tr><tr><td>True</td><td></td><td></td><td></td></tr><tr><td>ham</td><td>3421</td><td>39</td><td>3460</td></tr><tr><td>spam</td><td>69</td><td>1287</td><td>1356</td></tr><tr><td>All</td><td>3490</td><td>1326</td><td>4816</td></tr></table>	Predicted	ham	spam	All	True				ham	3421	39	3460	spam	69	1287	1356	All	3490	1326	4816	<div>Test Data</div> <table><tr><th>Predicted</th><th>ham</th><th>spam</th><th>All</th></tr><tr><td>True</td><td></td><td></td><td></td></tr><tr><td>ham</td><td>3427</td><td>33</td><td>3460</td></tr><tr><td>spam</td><td>37</td><td>1319</td><td>1356</td></tr><tr><td>All</td><td>3464</td><td>1352</td><td>4816</td></tr></table>	Predicted	ham	spam	All	True				ham	3427	33	3460	spam	37	1319	1356	All	3464	1352	4816																				
Predicted	ham	spam	All																																																										
True																																																													
ham	3421	39	3460																																																										
spam	69	1287	1356																																																										
All	3490	1326	4816																																																										
Predicted	ham	spam	All																																																										
True																																																													
ham	3427	33	3460																																																										
spam	37	1319	1356																																																										
All	3464	1352	4816																																																										
<div>Method 4.2: Precision Recall Plot for Random Forest</div> 	<div>Method 4.2: Precision Recall Plot for Navie Bayes</div> 																																																												

## Results

- Once again we are showing improvements In the accuracy scores for the Random Forest and Naive Bayes models.
- And once again these seem too good to be true. Since the accuracy scores and other metrics have been getting steadily better as we use each Feature Extraction method it shows that the Feature Extraction methods have a large effect on the results of the models. But they seem too good.
- We can also see from the Plots that the models are doing very well, which can be seen by the also 90 degree drop in the line plot.

#### 4.5 Method 5: Term Frequency-inverse document frequency (TF-IDF)

TF-IDF reflects the importance of a word by giving it a weight to the words that occurs more often. It uses two metrics. Term Frequency (TF): This is the number of times a word appears divided by the total number of words in a document which shows the importance of a word in a certain document. Inverse Document Frequency (IDF) looks at the total number of documents that contain the word and shows the rarity of a word in the entire corpus.

We could use TF-IDF separately but we want to build this on previous methods.

Method 5	Rows	Columns	Stored Elements	Density
Train Data:	8,270	70,161	772,986	0.1372%

Table 5: Count Vectorizer with TF-IDF

Figure 15 shows a Word Cloud for TF-IDF. Interesting we notice that we get similar results to the first method with Spacy. We can see “ect” and “hou” which we have seen in Method 1. Figure 16 shows the top 20 words for ham and spam and they do look like the most interesting we have seen in all the 5 methods.

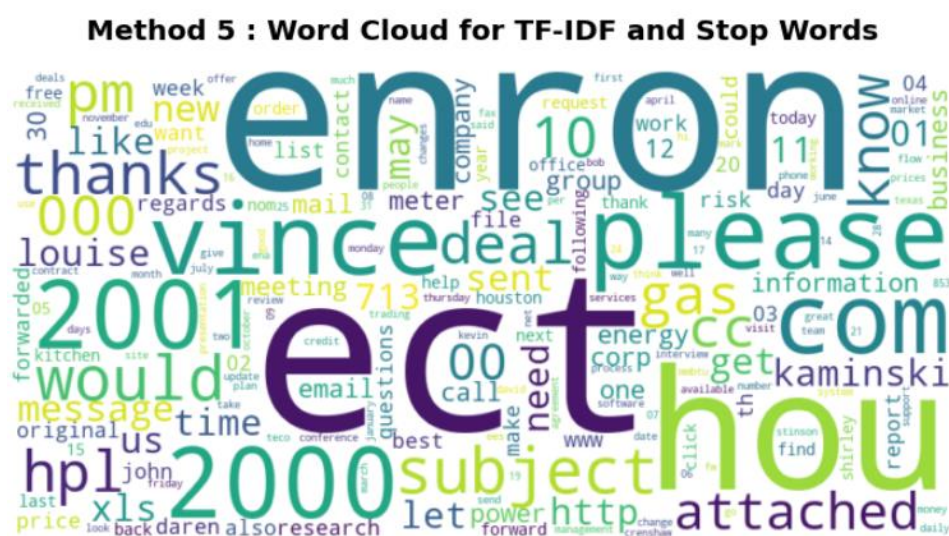


Figure 15: Count Vectorizer Word Cloud

# Top Words for Ham and Spam emails with TF-IDF and Stop Words

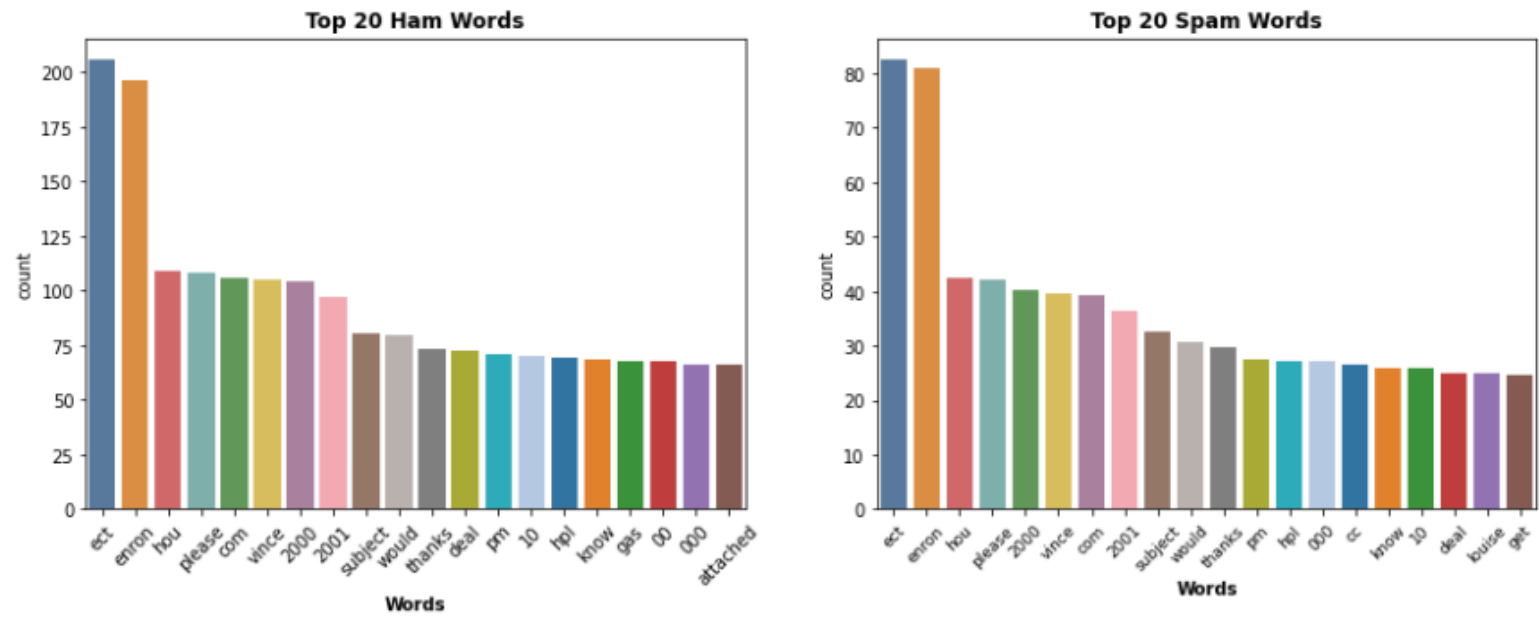
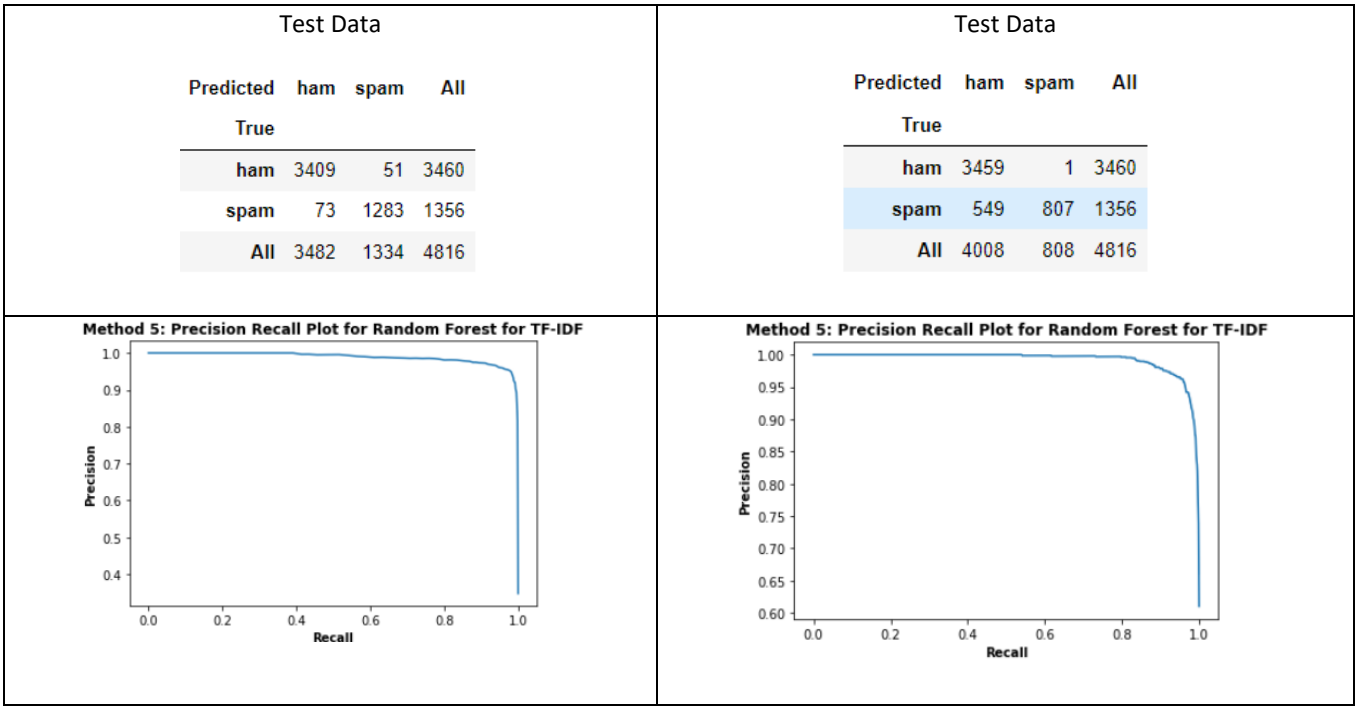


Figure 16: Count Vectorizer with Stop Words top 20 words Ham and Spam

Method 5: TF-IDF with Stop Words									
Random Forest Model					Naïve Bayes Model				
Test Accuracy: 0.9742					Test Accuracy: 0.8857				
Validation Data					Validation Data				
	precision	recall	f1-score	support		precision	recall	f1-score	support
ham	0.99	0.99	0.99	2321	ham	0.88	1.00	0.93	2321
spam	0.97	0.96	0.97	890	spam	1.00	0.64	0.78	890
accuracy			0.98	3211	accuracy			0.90	3211
macro avg	0.98	0.97	0.98	3211	macro avg	0.94	0.82	0.86	3211
weighted avg	0.98	0.98	0.98	3211	weighted avg	0.91	0.90	0.89	3211
Validation Data					Validation Data				
Predicted	ham	spam	All		Predicted	ham	spam	All	
True					True				
ham	2293	28	2321		ham	2321	0	2321	
spam	34	856	890		spam	323	567	890	
All	2327	884	3211		All	2644	567	3211	
Test Data					Test Data				
	precision	recall	f1-score	support		precision	recall	f1-score	support
ham	0.98	0.99	0.98	3460	ham	0.86	1.00	0.93	3460
spam	0.96	0.95	0.95	1356	spam	1.00	0.60	0.75	1356
accuracy			0.97	4816	accuracy			0.89	4816
macro avg	0.97	0.97	0.97	4816	macro avg	0.93	0.80	0.84	4816
weighted avg	0.97	0.97	0.97	4816	weighted avg	0.90	0.89	0.88	4816



Results

- Random Forest performs very well but this time there is a drop for the Navie Bayes model to below 89%. This is big drop.
- We will use this Navie Bayes Model as one of the ones we look to Tune since it is a good opportunity to improve on the accuracy score it has received. The model did very well at predicting all the positive cases but badly on the negative cases.

4.6 General Comments on models

Over fitting

Accuracy results were very good from the beginning and this lead the author to believe that the models were over fitting. We increased the data from one ENRON dataset to three to help but this made little difference. We also changed the ratio of Ham to Spam emails from 3:1 to 1:1 and it did not massively change the results.

Also, when we are looking at the different Feature Extraction Methods the Accuracy for the models generally improves slowly showing that each Feature Extraction method can be used to improve a models results. But they are still over performing.

We will look at Cross Fold Validation with tunning to look at fixing one of the very high Accuracy results in the HyperParameter Sections.

Accuracy results

As we saw the data is imbalanced and we did see in some models that they were biased to the majority class leading to poor performance on some minority classes. Class imbalance does not always cause overfitting however so this was not always seen. For Hyperparameters in Section 4 we will look at some tuning to deal with class imbalance with “class\_weight” in Random Forests to see if this can improve on the results.

## 5. Tuning a Model with Grid Search

Since some models have performed so well we will not use them as our models to tune. We will take one Naïve Bayes and one Random Forest and see how we can improve this using Grid Search which is a way to try out many hyperparameters and Cross Fold Validation. We want to see how powerful that tuning can be so we will take the following two models and look to improve them.

- Method 5: Navie Bayes TF-IDF Model with Accuracy results as 89%.
- Method 2: Random Forest Stop Words Model with Accuracy results as

### 5.1 Navie Bayes Hyper parameters.

Naïve Bayes are relatively simple types of models, Multinomial Naïve Bayes and Gaussian Naïve Bayes being two of the most common. We have used Multinomial NN which has a smoothing parameters (alpha) used to avoid zero probabilities when working out likelihoods.

For our Grid Search we ran various alpha values 0.01, 0.1, 1.0, 10.0 and get the best alpha of 0.01 that improves on the previous accuracy value of 89% to 98.7%. A very good improvement. More detail below.

Method 4: Navie Bayes Model	Method 4: Tuned Navie Bayes Model																																																												
Old Accuracy = 89%	New Accuracy = 98.7%																																																												
<div>Test Data</div> <table><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr><tr><td>ham</td><td>0.86</td><td>1.00</td><td>0.93</td><td>3460</td></tr><tr><td>spam</td><td>1.00</td><td>0.60</td><td>0.75</td><td>1356</td></tr><tr><td>accuracy</td><td></td><td></td><td>0.89</td><td>4816</td></tr><tr><td>macro avg</td><td>0.93</td><td>0.80</td><td>0.84</td><td>4816</td></tr><tr><td>weighted avg</td><td>0.90</td><td>0.89</td><td>0.88</td><td>4816</td></tr></table>		precision	recall	f1-score	support	ham	0.86	1.00	0.93	3460	spam	1.00	0.60	0.75	1356	accuracy			0.89	4816	macro avg	0.93	0.80	0.84	4816	weighted avg	0.90	0.89	0.88	4816	<div>Test Data</div> <table><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr><tr><td>ham</td><td>0.99</td><td>0.99</td><td>0.99</td><td>3460</td></tr><tr><td>spam</td><td>0.98</td><td>0.97</td><td>0.98</td><td>1356</td></tr><tr><td>accuracy</td><td></td><td></td><td>0.99</td><td>4816</td></tr><tr><td>macro avg</td><td>0.99</td><td>0.98</td><td>0.98</td><td>4816</td></tr><tr><td>weighted avg</td><td>0.99</td><td>0.99</td><td>0.99</td><td>4816</td></tr></table>		precision	recall	f1-score	support	ham	0.99	0.99	0.99	3460	spam	0.98	0.97	0.98	1356	accuracy			0.99	4816	macro avg	0.99	0.98	0.98	4816	weighted avg	0.99	0.99	0.99	4816
	precision	recall	f1-score	support																																																									
ham	0.86	1.00	0.93	3460																																																									
spam	1.00	0.60	0.75	1356																																																									
accuracy			0.89	4816																																																									
macro avg	0.93	0.80	0.84	4816																																																									
weighted avg	0.90	0.89	0.88	4816																																																									
	precision	recall	f1-score	support																																																									
ham	0.99	0.99	0.99	3460																																																									
spam	0.98	0.97	0.98	1356																																																									
accuracy			0.99	4816																																																									
macro avg	0.99	0.98	0.98	4816																																																									
weighted avg	0.99	0.99	0.99	4816																																																									
<div>Test Data</div> <table><tr><th>Predicted</th><th>ham</th><th>spam</th><th>All</th></tr><tr><td>True</td><td></td><td></td><td></td></tr><tr><td>ham</td><td>3459</td><td>1</td><td>3460</td></tr><tr><td>spam</td><td>549</td><td>807</td><td>1356</td></tr><tr><td>All</td><td>4008</td><td>808</td><td>4816</td></tr></table>	Predicted	ham	spam	All	True				ham	3459	1	3460	spam	549	807	1356	All	4008	808	4816	<div>Test Data</div> <table><tr><th>Predicted</th><th>ham</th><th>spam</th><th>All</th></tr><tr><td>True</td><td></td><td></td><td></td></tr><tr><td>ham</td><td>3439</td><td>21</td><td>3460</td></tr><tr><td>spam</td><td>46</td><td>1310</td><td>1356</td></tr><tr><td>All</td><td>3485</td><td>1331</td><td>4816</td></tr></table>	Predicted	ham	spam	All	True				ham	3439	21	3460	spam	46	1310	1356	All	3485	1331	4816																				
Predicted	ham	spam	All																																																										
True																																																													
ham	3459	1	3460																																																										
spam	549	807	1356																																																										
All	4008	808	4816																																																										
Predicted	ham	spam	All																																																										
True																																																													
ham	3439	21	3460																																																										
spam	46	1310	1356																																																										
All	3485	1331	4816																																																										

## 5.2 Random Forest Hyper parameters.

We will also look at using Grid Search and Cross Fold Validation with Random Forest as it has a lot more Hyperparameters that can have an effect on a models results. Some of the Hyperparameters are:

- **n\_estimators:** The number of trees in the forest.
- **max\_features:** The maximum number of features to split on.
- **max\_depth:** Max depth of each tree in the forest.
- **min\_samples\_split:** The minimum number of samples to split on a node.
- **min\_samples\_leaf:** Minimum number of samples required to be a leaf node.
- **bootstrap:** Should use bootstrap samples to build trees.
- **class\_weight:** weights assigned to each class to address class imbalance.

We created a Grid Search object with the following values for each of the hyperparameters:

- **n\_estimators:** 50, 100, 200
- **max\_features:** sqrt, log2
- **max\_depth:** 10, 20, 30
- **min\_samples\_split:** 2, 5, 10
- **min\_samples\_leaf:** 1, 2, 4

It takes a considerable time to run though the grid search. Below is a small example of the output.

```
Fitting 5 folds for each of 162 candidates, totalling 810 fits
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=50; total time= 0.6s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=50; total time= 0.6s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=50; total time= 0.6s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=50; total time= 0.6s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=50; total time= 0.6s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time= 1.2s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time= 1.2s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time= 1.2s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time= 1.2s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time= 1.2s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=200; total time= 2.5s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=200; total time= 2.5s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=200; total time= 2.5s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=200; total time= 2.5s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=200; total time= 2.5s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=1, min_samples_split=5, n_estimators=50; total time= 0.6s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=1, min_samples_split=5, n_estimators=50; total time= 0.6s
```

Figure 17: Count Vectorizer Word Cloud

We get the best hyperparameters with:

- `n_estimators`: 200
  - `max_features`: sqrt
  - `max_depth`: 30
- `min_samples_split`: 5
  - `min_samples_split`: 1

<u>Method 2: Random Forest Model</u>	<u>Method 4: Tuned Random Forest Model</u>																																																												
Old Test Accuracy = 0.9746	New Test Accuracy = 0.91																																																												
<div>Test Data</div> <table><tr><td></td><td>precision</td><td>recall</td><td>f1-score</td><td>support</td></tr><tr><td>ham</td><td>0.98</td><td>0.98</td><td>0.98</td><td>3453</td></tr><tr><td>spam</td><td>0.95</td><td>0.96</td><td>0.96</td><td>1364</td></tr><tr><td>accuracy</td><td></td><td></td><td>0.97</td><td>4817</td></tr><tr><td>macro avg</td><td>0.97</td><td>0.97</td><td>0.97</td><td>4817</td></tr><tr><td>weighted avg</td><td>0.97</td><td>0.97</td><td>0.97</td><td>4817</td></tr></table>		precision	recall	f1-score	support	ham	0.98	0.98	0.98	3453	spam	0.95	0.96	0.96	1364	accuracy			0.97	4817	macro avg	0.97	0.97	0.97	4817	weighted avg	0.97	0.97	0.97	4817	<div>Test Data</div> <table><tr><td></td><td>precision</td><td>recall</td><td>f1-score</td><td>support</td></tr><tr><td>ham</td><td>0.89</td><td>1.00</td><td>0.94</td><td>3453</td></tr><tr><td>spam</td><td>0.99</td><td>0.69</td><td>0.82</td><td>1364</td></tr><tr><td>accuracy</td><td></td><td></td><td>0.91</td><td>4817</td></tr><tr><td>macro avg</td><td>0.94</td><td>0.85</td><td>0.88</td><td>4817</td></tr><tr><td>weighted avg</td><td>0.92</td><td>0.91</td><td>0.91</td><td>4817</td></tr></table>		precision	recall	f1-score	support	ham	0.89	1.00	0.94	3453	spam	0.99	0.69	0.82	1364	accuracy			0.91	4817	macro avg	0.94	0.85	0.88	4817	weighted avg	0.92	0.91	0.91	4817
	precision	recall	f1-score	support																																																									
ham	0.98	0.98	0.98	3453																																																									
spam	0.95	0.96	0.96	1364																																																									
accuracy			0.97	4817																																																									
macro avg	0.97	0.97	0.97	4817																																																									
weighted avg	0.97	0.97	0.97	4817																																																									
	precision	recall	f1-score	support																																																									
ham	0.89	1.00	0.94	3453																																																									
spam	0.99	0.69	0.82	1364																																																									
accuracy			0.91	4817																																																									
macro avg	0.94	0.85	0.88	4817																																																									
weighted avg	0.92	0.91	0.91	4817																																																									
<div>Test Data</div> <table><tr><td>Predicted</td><td>ham</td><td>spam</td><td>All</td></tr><tr><td>True</td><td></td><td></td><td></td></tr><tr><td>ham</td><td>3384</td><td>69</td><td>3453</td></tr><tr><td>spam</td><td>53</td><td>1311</td><td>1364</td></tr><tr><td>All</td><td>3437</td><td>1380</td><td>4817</td></tr></table>	Predicted	ham	spam	All	True				ham	3384	69	3453	spam	53	1311	1364	All	3437	1380	4817	<div>Test Data</div> <table><tr><td>Predicted</td><td>ham</td><td>spam</td><td>All</td></tr><tr><td>True</td><td></td><td></td><td></td></tr><tr><td>ham</td><td>3409</td><td>51</td><td>3460</td></tr><tr><td>spam</td><td>73</td><td>1283</td><td>1356</td></tr><tr><td>All</td><td>3482</td><td>1334</td><td>4816</td></tr></table>	Predicted	ham	spam	All	True				ham	3409	51	3460	spam	73	1283	1356	All	3482	1334	4816																				
Predicted	ham	spam	All																																																										
True																																																													
ham	3384	69	3453																																																										
spam	53	1311	1364																																																										
All	3437	1380	4817																																																										
Predicted	ham	spam	All																																																										
True																																																													
ham	3409	51	3460																																																										
spam	73	1283	1356																																																										
All	3482	1334	4816																																																										

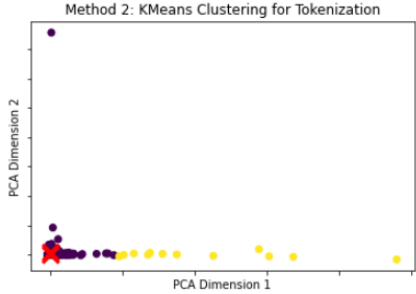
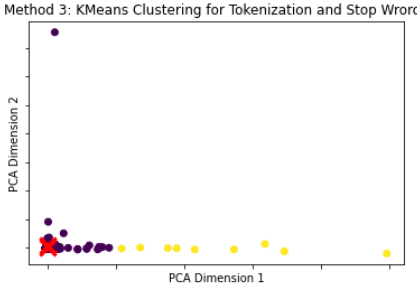
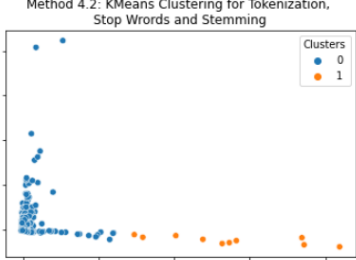
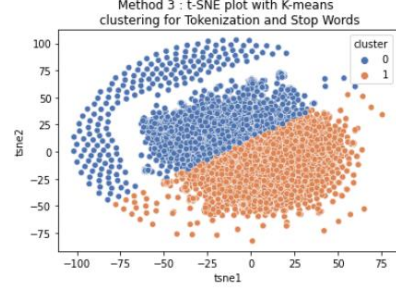
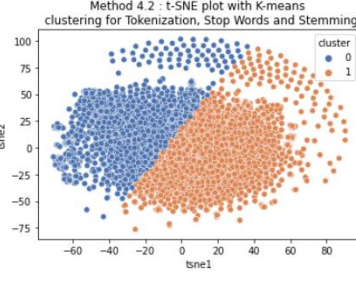
The old model is over fitting and with a accuracy score of 97% and while the Hyperparameter accuracy model score was 91% and I think this is a much more accurate value as the tuning with Grid Search uses Cross Fold Validation and this can be used to help with over fitting models.

This model was then saved to a pickle file for later use.



## 6. Unsupervised Modelling

I also looked at an Unsupervised approach with Kmeans to a smaller extent. It is hard to compare these models with the Supervised models so I have compared some of the different Feature Extraction Methods.

<u>Method 2: Basic Tokenization</u>	<u>Method 3: Tokenization with Stop Words</u>	<u>Method 4.2: Tokenization, Stop Words and Stemming</u>
Test Silhouette Score: 0.9601 Test Calinski-Harabasz: 4937	Test Silhouette Score: 0.9378 Test Calinski-Harabasz: 2382	Test Silhouette Score: 0.9515 Test Calinski-Harabasz: 3233
<b>Validation Data</b> Validation Silhouette Score: 0.9644 Validation Calinski- Harabasz: 4937	<b>Validation Data</b> Validation Silhouette Score: 0.9518 Validation Calinski- Harabasz: 2382	<b>Validation Data</b> Test Silhouette Score: 0.9601 Test Calinski- Harabasz: 3233
<b>Test Data</b> Test Silhouette Score: 0.9601 Test Calinski- Harabasz: 4937	<b>Test Data</b> Test Silhouette Score: 0.9378 Test Calinski- Harabasz: 2382	<b>Test Data</b> Test Silhouette Score: 0.9515 Test Calinski- Harabasz: 3233
		
		



Method 2 with Basic Tokenization has the highest Silhouette Score and Calinski-Harabasz Index indicating that it produces the best defined clusters. It does better when we use more complex Feature Extraction methods which is interesting.

You can see this more clearly in the T-SNE plots, Method 2 has two well grouped clusters whereas both Method 3 and Method 4.2 have clusters that are not as well defined.

## 7. Conclusion

This project explored the effectiveness of different Machine Learning Models and Feature Extraction Techniques in classifying emails from into ham and spam in order to create a Spam Detector. Several different Feature Extraction methods were applied on the email data, including basic tokenization, tokenization with stop words, tokenization with stop words and stemming, TF-IDF and Spacy with varying degrees of success using different Machine learning models such as Naive Bayes, Random Forest, Gradient Boosting and Kmeans models.

However, the project also encountered the problem of over fitting which was addressed first by attempting to triple the dataset and then by balancing the data but was finally solved by looking at Cross Fold validation. Hyperparameters were used with Grid Search to try different parameters and cross fold validation to improve a model's performance so that more realistic Accuracy results could be achieved with a final model using Random Forest with tuned parameters and an accuracy score of 91%.

## 8. References

- [1] "Spam e-mail traffic share monthly 2022," *Statista*. <https://www.statista.com/statistics/420391/spam-email-traffic-share/> (accessed Mar. 25, 2023).
- [2] "What's On the Other Side of Your Inbox - 20 SPAM Statistics for 2023," *Dataprot*. <https://dataprot.net/statistics/spam-statistics/> (accessed Mar. 25, 2023).
- [3] "Hackers threaten to publish 'confidential' MTU data unless ransom is paid, High Court told," *The Irish Times*. <https://www.irishtimes.com/ireland/education/2023/02/11/hackers-threaten-to-publish-confidential-mtu-data-unless-ransom-is-paid-high-court-told/> (accessed Mar. 17, 2023).
- [4] "How the HSE cyber attack changed the face of online crime globally," *The Irish Times*. <https://www.irishtimes.com/technology/data-security/2023/03/11/how-the-hse-cyber-attack-changed-the-face-of-online-crime-globally/> (accessed Mar. 17, 2023).