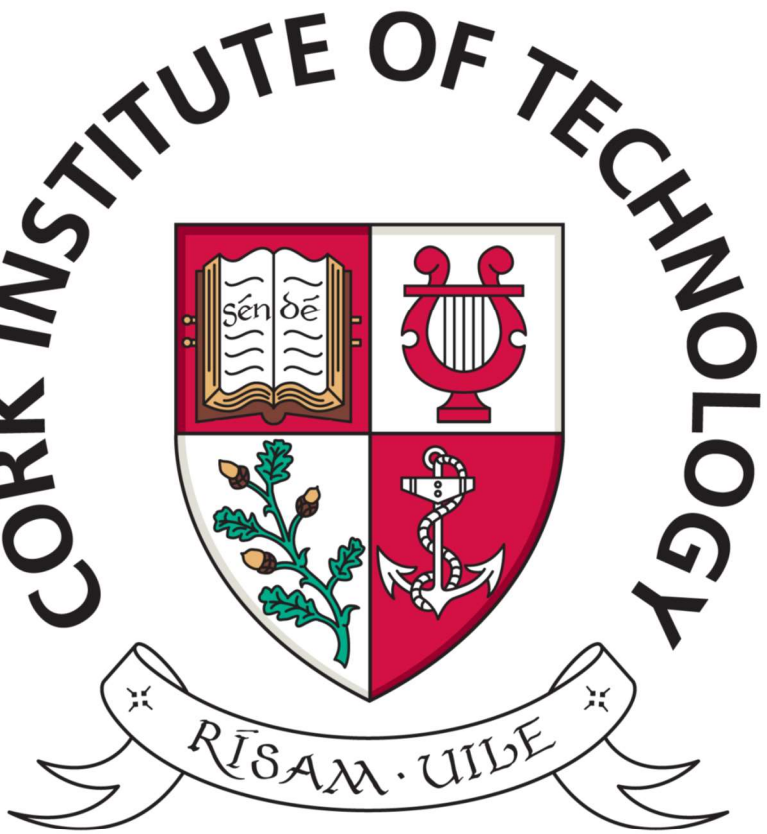


Name: Sumeet Shivgand
Student Id: R00182850
Subject: Applied Machine Learning

Assignment 1

Table of Contents

1. Introduction:

Email has become one of the most important forms of communication. In 2014, there are estimated to be 4.1 billion email accounts worldwide, and about 196 billion emails are sent each day worldwide. Spam is one of the major threats posed to email users. In 2013, 69.6% of all email flows were spam. Links in spam emails may lead to users to websites with malware or phishing schemes, which can access and disrupt the receiver's computer system. These sites can also gather sensitive information. Additionally, spam costs businesses around $2000 per employee per year due to decreased productivity. Therefore, an effective spam filtering technology is a significant contribution to the sustainability of the cyberspace and to our society.

There are currently different approaches to spam detection. These approaches include blacklisting, detecting bulk emails, scanning message headings, greylisting, and content-based filtering:

- Blacklisting is a technique that identifies IP addresses that send large amounts of spam. These IP addresses are added to a Domain Name System-Based Blackhole List and future email from IP addresses on the list are rejected. However, spammers are circumventing these lists by using larger numbers of IP addresses.

- Detecting bulk emails is another way to filter spam. This method uses the number of recipients to determine if an email is spam or not. However, many legitimate emails can have high traffic volumes.

- Scanning message headings is a reliable way to detect spam. Programs written by spammers generate headings of emails. Sometimes, these headings have errors that cause them to not fit standard heading regulations. When these headings have errors, it is a sign that the email is probably spam. However, spammers are learning from their errors and making these mistakes less often.

- Greylisting is a method that involves rejecting the email and sending an error message back to the sender. Spam programs will ignore this and not resend the email, while humans are more likely to resend the email. However, this process is annoying to humans and is not an ideal solution.

Current spam techniques could be paired with content-based spam filtering methods to increase effectiveness. Content-based methods analyze the content of the email to determine if the email is spam. The goal of this assignment is to create a supervised classification pipeline to classify emails as spam or non-spam from the training data and determine their effectiveness as content-based spam filters. We are going to use 'Enron' email dataset for analysis.

In this assignment, Enron email dataset is going to be used. The dataset nothing but the collection of public domain emails from the Enron Corporation. The main aim of this assignment is to create a supervised classification pipeline to classify emails as spam or non-spam from the training data.

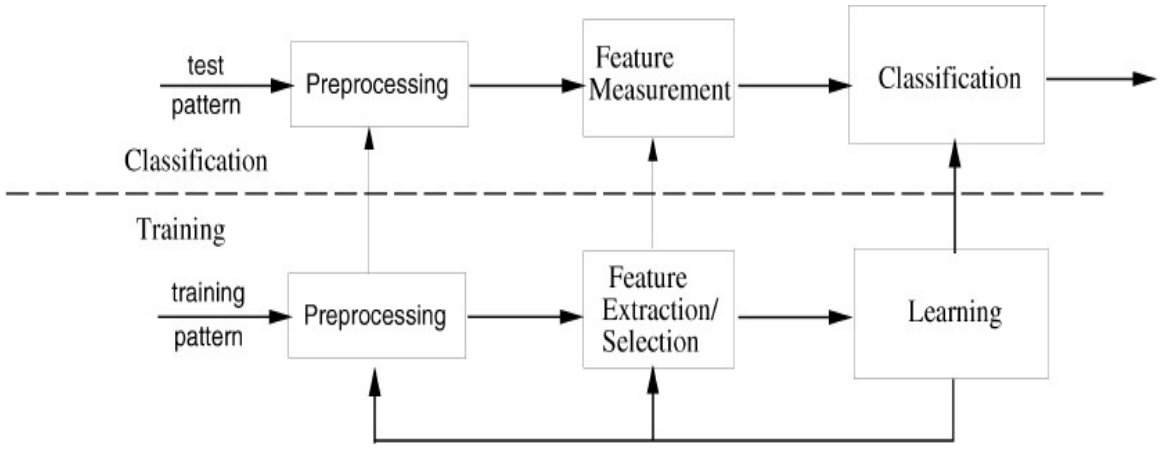
2. Machine Learning Basics

When we choose to approach spam filtering from a machine learning perspective, we view the problem as a classification problem. That is, we aim to classify an email as spam or not spam (ham) depending on its feature values.

Machine learning systems can be trained to classify emails. As shown in Figure 1, a machine learning system operates in two modes: training and testing.

Training
During training, the machine learning system is given labeled data from a training data set. In our project, the labeled training data are a large set of emails that are labeled spam or ham. During the training process, the classifier (the part of the machine learning system that predicts labels of future emails) learns from the training data by determining the connections between the features of an email and its label.

Testing (Classification)
During testing, the machine learning system is given unlabeled data. In our case, these data are emails without the spam/ham label. Depending on the features of an email, the classifier predicts whether the email is spam or ham. This classification is compared to the true value of spam/ham to measure performance.



A model of the machine learning process. Image Credit: "Statistical Pattern Recognition".

3. Methods

Figure 1 shows the different steps in the machine learning process. First, in testing and training, the data must be preprocessed so that they able to be used by the classifier. Then, the classifier operates in training or testing mode to either learn from the preprocessed data or classify future data.

3.1 Preprocessing

Our classifiers require a feature matrix that consists of the count of each word in each email. As will we discuss, this feature matrix can grow very large, very quickly. Thus, preprocessing the data involves two main steps: creating the feature matrix and reducing the dimensionality of the feature matrix.

3.1.1. Cleaning Process

Now, we understand how our dataset look like, lets convert our labels to binary variables, 0 represent 'ham'(non-spam) and 1 represent 'spam' for ease of computation. Scikit-learn only deals with numerical values and hence if we were to leave our label values as strings, Scikit-learn would do the conversion internally (more specifically, the string labels will cast to unknown float values). Our model would still be able to make predictions if we left our labels as strings, but we could have issues later when calculating performance metrics, for example when calculating our precision and recall scores. Hence, to avoid unexpected problems later, it is good practice to have our categorical values be fed into our model as integers. The given dataset contains duplicates value, so we drop the duplicates value using 'drop_duplicate'. Below is how our dataset look like: It contains 5172 rows and 2 columns.

	Emails	target
0	Subject: nom chnage - 7 / 7 / 2000 - eastrans\...	0
1	Subject: re : teco gas processing company\n- -...	0
2	Subject: it ' s cheating , but it works !\ncan...	1
3	Subject: welcome to woodworkingtips . com !\n*...	0
4	Subject: enron / hpl actuals for july 26 , 200...	0

Since, there was duplicate values in dataset. So, after removing the duplicates observation we get the following output. Earlier, there was 5172 observations and after removing duplicates we get 4994 observations. Also, there was no missing values like NAN, NaN, na.

```
(5172, 2)
Index(['Emails', 'target'], dtype='object')
(4994, 2)

Emails    0
target    0
dtype: int64
```

3.1.2. Training and Test splits

Split the dataset into a training and testing set by using the train_test_split method in sklearn. Split the data using the following variables:

- X_train is our training data for the 'email_message' column.
- y_train is our training data for the 'label/target' column
- X_test is our testing data for the ' email _message' column.
- y_test is our testing data for the 'label/target' column.

Here, we split the dataset into 70:30 i.e. 70% of training set and 30 % test set. We shuffle the data set so that test set is not bias in any way. Also, duplicates and empty emails had handled properly. Later, we store the resulting training and test set into files such as 'train_set' and 'test_set' as csv file. Next, we count the total numbers of spam and non-spam in each set.

Train set:
From following output, we see that, there are 2476 'ham (non-spam)' emails and 1019 'spam' emails.

col_0	count
target	
0	2476
1	1019

Test set:
From following output, we see that, there are 1055 'ham (non-spam)' emails and 444 'spam' emails.

col_0	count
target	
0	1055
1	444

3.1.3 Creating the Feature Matrix (Feature Extraction)

Figure 2 shows an example of an email in our dataset taken from Enron corporation. This corpus is described in more detail in Section 3.3.

```
Subject: dobmeos with hgh my energy level has gone up ! stukm
introducing
doctor - formulated
hgh
human growth hormone - also called hgh
is referred to in medical science as the master hormone . it is very plentiful
when we are young , but near the age of twenty - one our bodies begin to produce
less of it . by the time we are forty nearly everyone is deficient in hgh ,
and at eighty our production has normally diminished at least 90 - 95 % .
advantages of hgh :
- increased muscle strength
- loss in body fat
- increased bone density
- lower blood pressure
- quickens wound healing
- reduces cellulite
- improved vision
- wrinkle disappearance
- increased skin thickness texture
- increased energy levels
- improved sleep and emotional stability
- improved memory and mental alertness
- increased sexual potency
- resistance to common illness
- strengthened heart muscle
- controlled cholesterol
- controlled mood swings
- new hair growth and color restore
read
more at this website
unsubscribe
```

An example of an email in the Enron dataset.

There are a couple steps involved in creating the feature matrix, as shown in Figure 3. After preliminary preprocessing (removing HTML tags and headers from the email in the dataset), we take the following steps:

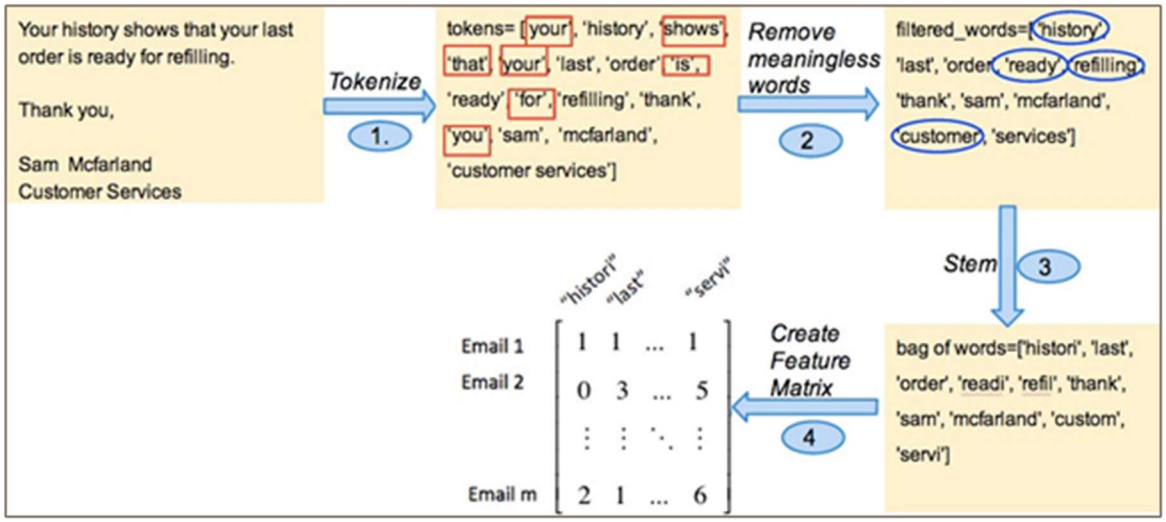
1. **Tokenize** - We create "tokens" from each word in the email by removing punctuation.

2. **Remove meaningless words** – Meaningless words, known as stop-words, do not provide meaningful information to the classifier, but they increase dimensionality of feature matrix. In Figure 3, the red boxes outline the stop-words, which should be removed. In addition to many stop-words, we removed words over 12 characters and words less than three characters.

3. **Stem** - Similar words are converted to their "stem" in order to form a better feature matrix. This allows words with similar meanings to be treated the same. For example, history, histories, historic will be considered same word in the feature matrix. Each stem is placed into our "bag of words", which is just a list of every stem used in the dataset. In Figure 3, the tokens in blue circle are converted to their stems.

4. **Create feature matrix** - After creating the "bag of words" from all the stems, we create a feature matrix. The feature matrix is created such that the entry in row *i* and column *j* is the number of times that token *j* occurs in email *i*.

These steps are completed using a Python NLTK (Natural Language Toolkit).



Here, the documents in the data frame or dataset are numbered in the rows, and each word is a column name, with the corresponding value being the frequency of that word in the document. Later, we must break this down and see how we can do this conversion using a small set of documents. To handle this, we will be using sklearns 'count vectorizer' method which does the following:

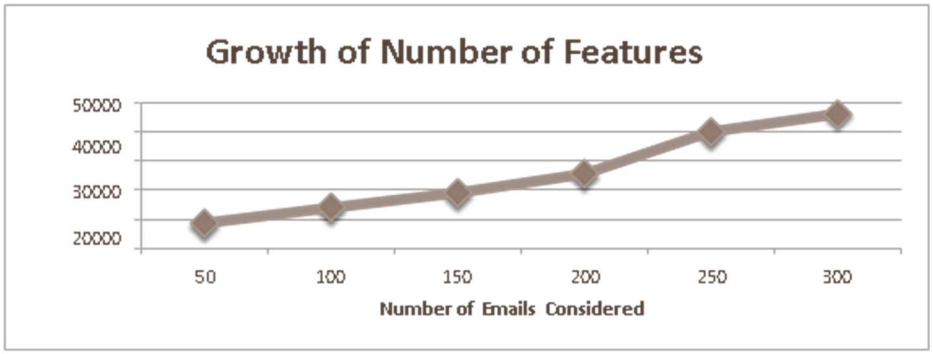
- It tokenizes the string (separates the string into individual words) and gives an integer ID to each token.
- It counts the occurrence of each of those tokens.

```
0    [subject, nom, chnage, 7, 7, 2000, eastrans, i...
1    [subject, teco, gas, processing, company, forw...
2    [subject, cheating, works, guess, old, woman, ...
3    [subject, welcome, woodworkingtips, com, pleas...
4    [subject, enron, hpl, actuals, july, 26, 2000,...
Name: Emails, dtype: object
```

3.1.4. Reducing the Dimensionality (Bag-of-Words)

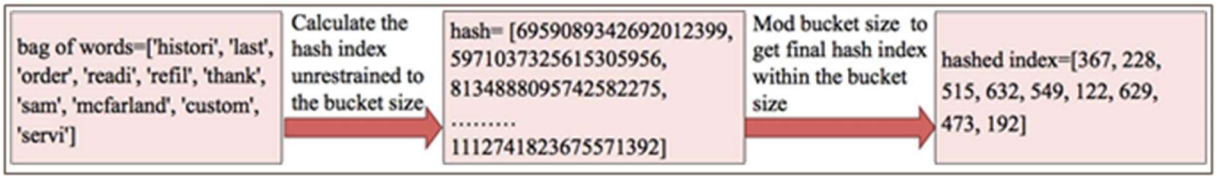
The bag of word method creates a highly dimensional feature matrix and this matrix grows quickly with respect to the number of emails considered. A highly dimensional feature matrix greatly slows the runtime of our algorithms.

When using 50 emails, there are roughly 8,700 features, but this number quickly grows to over 45,000 features when considering 300 emails (as shown in Figure 4). Thus, it is necessary to reduce the dimensionality of the feature matrix.



The growth of number of features with respect to the number of emails considered.

To reduce the dimensionality, we implemented a hash table to group features together. Each stem in the bag of words comes with a built-in hash index in Python. We can then decide how many hash buckets (or features) we would like to have. We take the built-in hash index mod the bucket size to find the new hashed index. This process is shown in Figure 5.



Now, we can convert a collection of documents to a feature matrix, with each document being a row and each word(token) being the column, and the corresponding (row, column) values being the frequency of occurrence of each word or token in that document. We will be using 'sklearn.feature_extraction.text' package to implement Bag-of-Word.

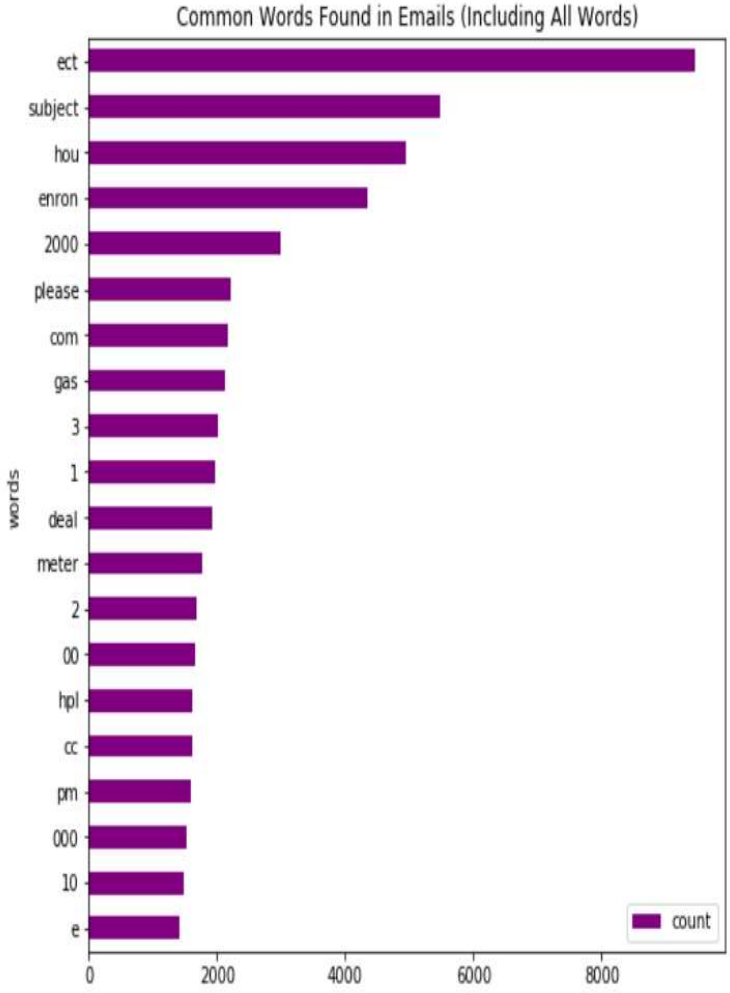
```
50463
CountVectorizer(analyzer='word', binary=False, decode_error='strict',
                dtype=<class 'numpy.int64'>, encoding='utf-8',
                input=<function process_text at 0x000002192CA50E18>,
                lowercase=True, max_df=1.0, max_features=None, min_df=1,
                ngram_range=(1, 1), preprocessor=None, stop_words=None,
                strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
                tokenizer=None, vocabulary=None)
(4994, 50463)
```

After this preprocessing is finished, the classifiers can use the feature matrix in its training and testing modes.

4. Exploratory Data Analysis

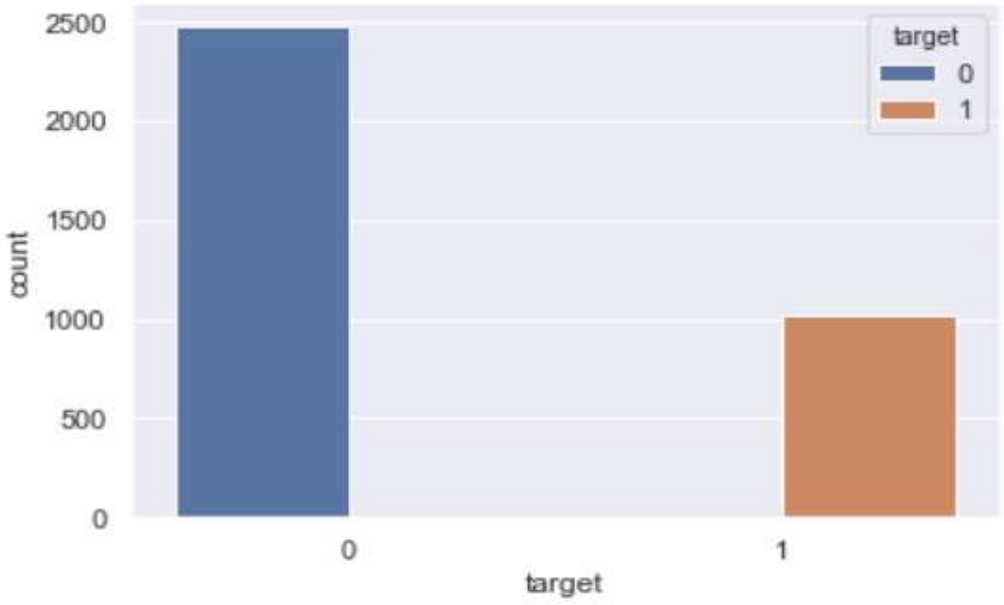
We had found the top 20 most frequently used words in spam and non-spam and we will be using a bar plot to show their relative frequencies. From output, we see that, ‘ect’ word is the most frequently word used in spam and non-spam emails.

```
<class 'pandas.core.series.Series'>
[('ect', 9449), ('subject', 5489), ('hou', 4956), ('enron', 4356), ('2000', 2988), ('please', 2206), ('com', 2179), ('gas', 213
2), ('3', 2003), ('1', 1958), ('deal', 1927), ('meter', 1768), ('2', 1676), ('00', 1647), ('hpl', 1620), ('cc', 1606), ('pm', 1
586), ('000', 1530), ('10', 1475), ('e', 1411)]
```



Below bar graph shows that, non-spam (0) emails have the maximum distribution of email lengths compared to spam emails.

```
<matplotlib.axes._subplots.AxesSubplot at 0x2c9ea73d748>
```



5. Supervised Classification

In this, we are going to train a supervised classification model on our features i.e. bag-of-words/ TF-IDF to calculate validation accuracy.
Sklearn has several Naive Bayes implementations that we can use and so we do not have to do the math from scratch. We will be using sklearn's '**sklearn.naive_bayes**' method to make predictions on our dataset. Specifically, we will be using the **Multinomial Naive Bayes** implementation. This classifier is suitable for classification with discrete features (such as in our case, word counts for text classification). It takes in integer word counts as its input.

```
MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
```

Now, we calculate the accuracy score and confusion matrix to check how many emails are misclassified.

```
Accuracy Score:
 0.8579052701801201
```

	Predicted 0	Predicted 1
Actual 0	1056	0
Actual 1	213	230

From above confusion matrix, we see that, model misclassify 231 spam messages as non-spam emails whereas we don't misclassify any non-spam message. There are two types of error i.e. Type I and Type II. Type I error generated by model is zero (0) and Type II error generated by model is 231.

6. Model Selection

As mentioned before, a machine learning system can be trained to classify emails as spam or ham. To classify emails, the machine learning system must use some criteria to make its decision. The different algorithms that we describe below are different ways of deciding how to make the spam or ham classification.

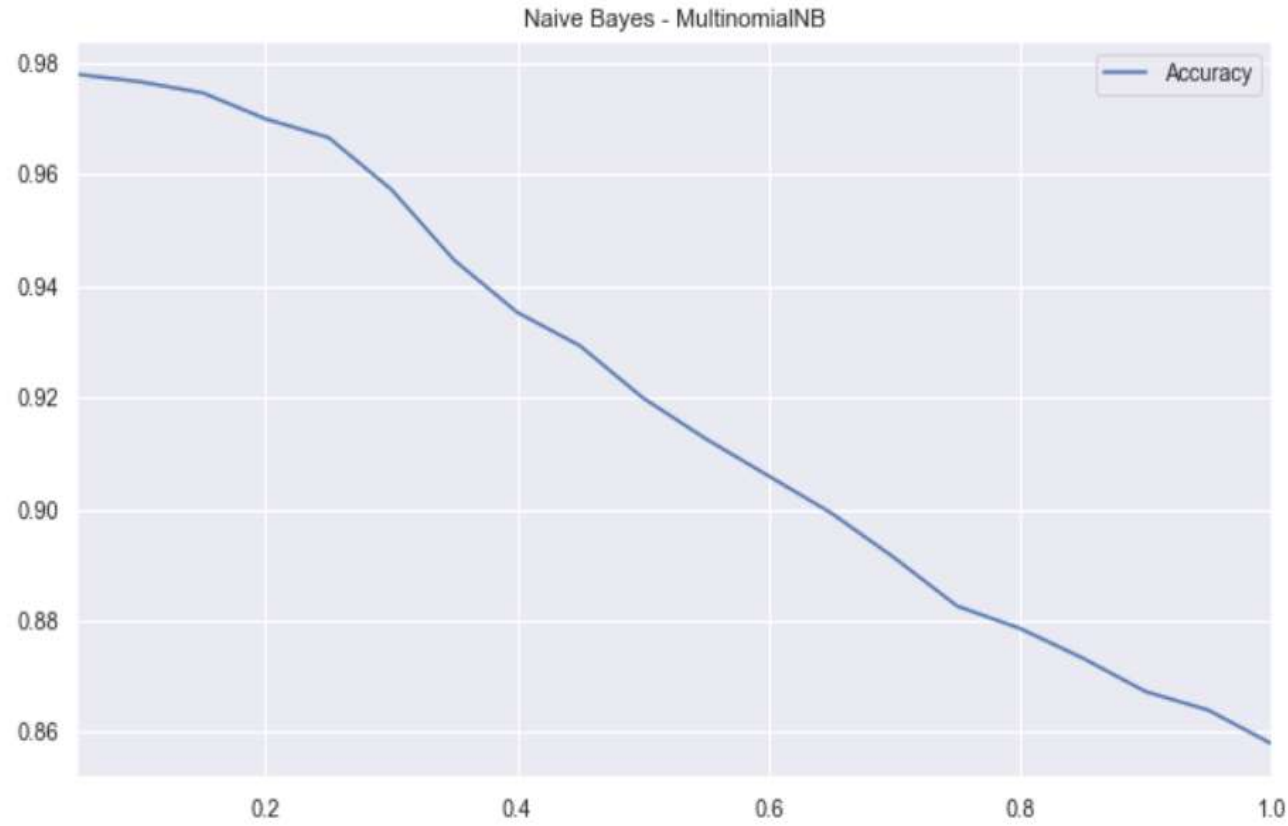
In this section, we are going to compare two different classifiers or models using hyperparameter tuning i.e. 'Naive Bayes Multinomial' and 'Decision Tree'. Then use a validation set to compare the accuracy of these two models. Later, comparing the models by plotting and use the use the most effective model for evaluation.

6.1 Hyper tuning

6.1.1 Naive Bayes Multinomial

The multinomial Naive Bayes classifier is suitable for classification with discrete features (e.g., word counts for text classification). The multinomial distribution normally requires integer feature counts. However, in practice, fractional counts such as tf-idf may also work.

	Accuracy
0.05	0.977985



Multinomial Model Plot

6.1.2 Decision Tree

Decision tree learning is a method commonly used in data mining. The decision tree algorithm aims to build a tree structure according to a set of rules from the training dataset, which can be used to classify unlabeled data. In our implementation, we used the well-known iterative Dichotomiser 3 (ID3) algorithm invented by Ross Quinlan to generate the decision tree.

	Accuracy
5	0.941294

Decision Tree Model Plot

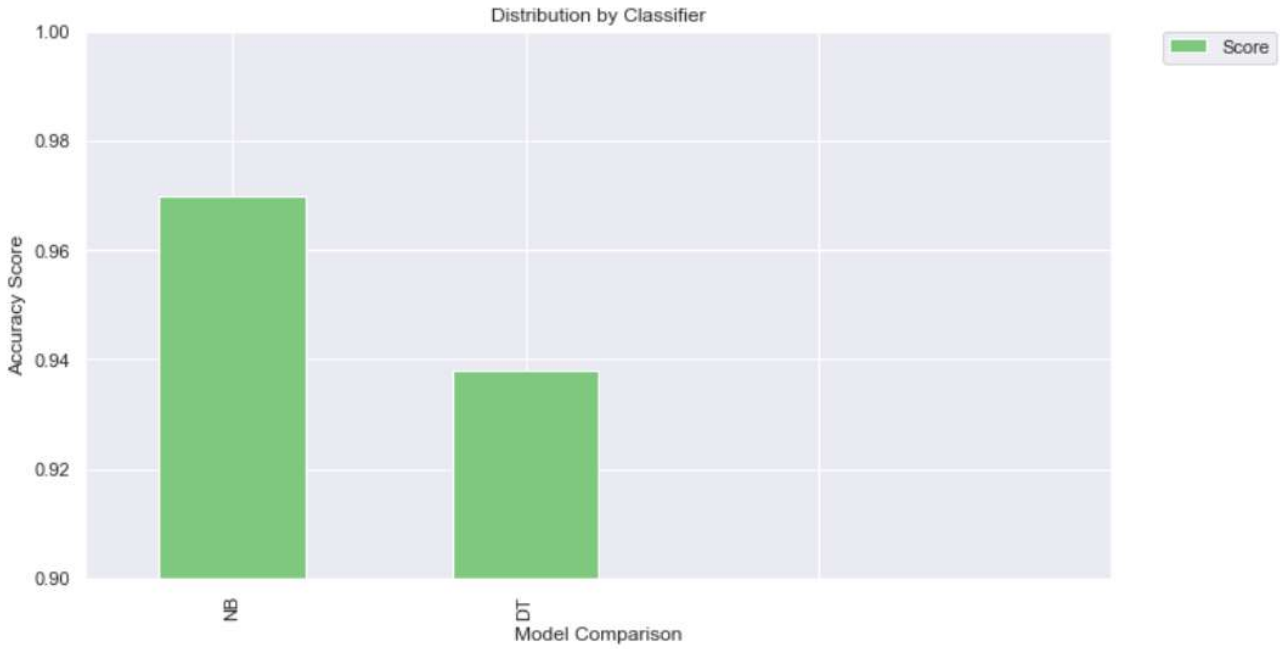
Here, we compare the accuracy score for two models i.e. 'Multinomial' and 'Decision Tree'. For 'Multinomial model', we get the highest accuracy score 98.27% at alpha value '0.05' and '0.10'. For 'Decision Tree model' we get the highest accuracy score 94.32% at min_split '4'.

6.2 Model Comparison

Now, we are going to use bar plot for model comparison based on accuracy score.

	Score
NB	0.969980
DT	0.937959

```
<matplotlib.legend.Legend at 0x2280ad58f98>
```



From above bar plot, we can clearly say that, 'Multinomial' has the highest accuracy score compared 'Decision Tree'. So, we are going to retain 'Multinomial' model for evaluation as it gives the highest accuracy score.

7. Model Evaluation

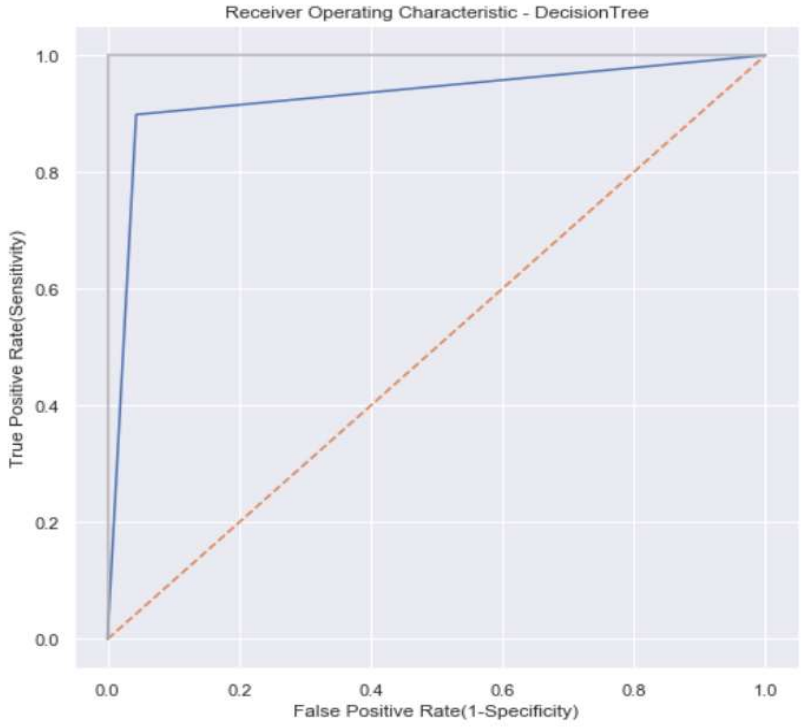
Now, we have made predictions on our test set, our next goal is to do model evaluation to see how well our model is performing. There are various mechanisms for doing so, but we will be using 'Classification Report 'as performance metrics.

	precision	recall	f1-score	support
0	0.83	1.00	0.91	1056
1	1.00	0.52	0.68	443
accuracy			0.86	1499
macro avg	0.92	0.76	0.80	1499
weighted avg	0.88	0.86	0.84	1499

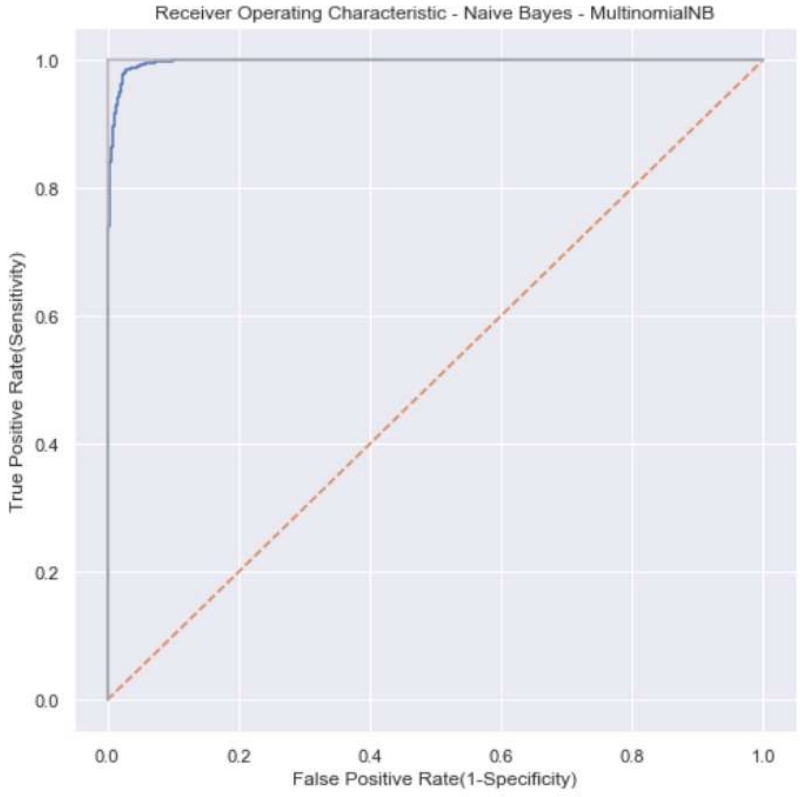
As we seen in Figure 13, the Multinomial algorithm performed the best with the following performance metrics:

- Accuracy: - 86%
- Precision: - 92%
- Recall: - 76%
- F1-Score: - 80%
- Error rate: - 14%

```
Roc_auc_score for DecisionTree:  0.9274296292496066
```




```
Roc_auc_score for Naive Bayes(MultinimialNB):  0.9966118578562145
```



ROC Curve:

ROC Curve is nothing but '**Receiver Operating Characteristic curve'.** It is a plot of the true positive rate against the false positive rate for the different possible cutpoints of a diagnostic test.
A ROC curve demonstrates several things:

- It shows the tradeoff between sensitivity and specificity (any increase in sensitivity will be accompanied by a decrease in specificity).
- The closer the curve follows the left-hand border and then the top border of the ROC space, the more accurate the test.
- The closer the curve comes to the 45-degree diagonal of the ROC space, the less accurate the test.

Here, we have plot ROC curve for 'Multinomial' and 'Decision Tree' model. We see that, the curve for 'Decision Tree' is little bit closer to the 45-degree diagonal as compared to 'Multinomial' model. It means that the accuracy for 'Multinomial' is higher than 'Decision Tree' model.

8. Conclusion:

We found that the Naive Bayes algorithm outperformed the other classifiers. This simple algorithm achieved great performance and was easy to implement. However, we believe that the performance of this algorithm could still be improved. One of the major advantages is that its model training and prediction times are very fast for data it can handle. All in all, Naive Bayes' really is a gem of an algorithm.

Another key finding was that we discovered that the precision of this algorithms was very high, while the recall was lower. This suggests that our algorithms are very liberal in labeling an email as spam. To combat this, perhaps mapping the features to a higher dimension, as is done in Support Vector Machine algorithms, would be a solution to this problem.

After analysis, we believe that a machine learning approach to spam filtering is a viable and effective method to supplement current spam detection techniques.