CS60050: MACHINE LEARNING

Project 1: Spam Filter using Naive Bayes

Group Number: **27**

Members:

Akshay Ramesh Bhivagade (22CL60R16)

Anubhav Dhar (20CS30004)

Rahul Arvind Mool (22CS60R72)

1 Introduction

The aim of this project is to create a **Email Spam Filter using Naive Bayesian algorithm**. We start with a dataset of over 5000 samples in the file `email_spam_dataset`. In this report, we denote by `SPAM` the class of spam mails and by `HAM` the class of mails which are not spam.

We first analyse the various aspects of this data set including the top ten words in both `SPAM` and `HAM`. We then analyse the relation between word lengths and the classes.

Following the Analysis, we start by building a *Naive Bayes Classifier* on 10 independent and random 80-20 splits of the given dataset. We choose the best out of these 10 splits (the one with the best accuracy) and train the classifier with other variations of the data (with & without Laplace Correction and with & without Text Cleaning).

We conclude the creating a classification report of both the classifier and analysing the impact of the performance of the models.

2 Observations

We start by reporting some observations of the text in the dataset.

2.1 Top Ten words of each class

To select the top ten words of each class, we performed text cleaning, considered words which are purely alphabetic (*i.e.* no numeric/special characters appear). The results were:

SPAM	HAM
subject	ect
http	hou
company	enron
com	subject
u	deal
price	gas
e	meter
www	com
font	hpl
statement	please

2.2 Various Plots of the dataset

Following text cleaning, we analyse class distribution and the relationship between **word length** and the classes (`SPAM` and `HAM`). We first observe that the number of `HAM` mails are much more than the number of `SPAM` mails. This is expected as relevant mails appear more in frequency than spams in real life. Further we observe that spam has a range of different word lengths; which is also expected as relevant mails tend to be precise and to the point avoiding unnecessary word decorations.

Following is the *Class Distribution* of the dataset when plotted in the form of a bar graph (using `matplotlib` library):

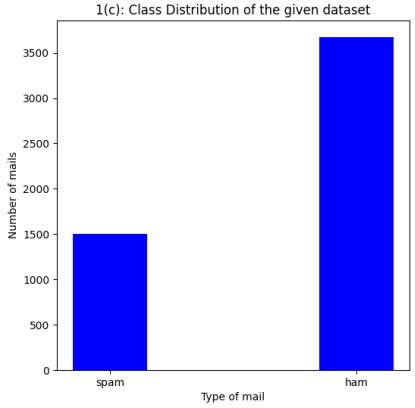

Figure 1: Class Distribution of the given dataset

Following is the *Box plot* of the word lengths for each of the classes `SPAM` and `HAM` of the dataset when plotted in the form of a bar graph (using `matplotlib` library):

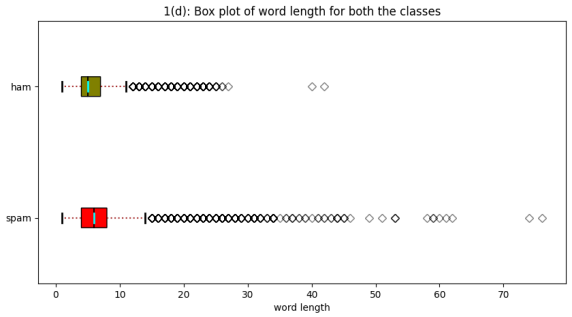

Figure 2: Box Plot of word lengths for each class

Finally, we plot the *Kernel Density Estimation (KDE)* plot of the word lengths for each of the classes (using `seaborn` and `matplotlib` libraries):

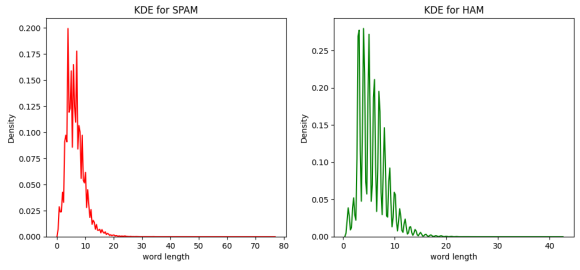

Figure 3: KDE Plot of word length for each class

2.3 Classification Report

Now lets talk about outputs. We trained the data for top n features *i.e.* (top n words which comes most frequently in our corpus for `SPAM` and `HAM`). For different values of n, we found that it was not giving any significant result. After trying multiple values of n, we found that the value of $n = 425$ has the best effect. So for all the problems given in the question we are using 425 as the size of features in our training set.

- For **2.a** we trained the data in $80 - 20$ split we found the average accuracy by running the code 10 time for different randomised splits of $80 - 20$ percentage out of which we also calculated the best accuracy from all the 10 randomised splits.

```
Without laplace correction average of 10 different data set is  0.8871497584541064
Without laplace correction maximum accuracy of 10 different data set is  0.9169082125603865
```

Figure 4: Maximum and average accuracy for 10 different splits

- For **2.b** we used Laplace correction with value of k as 1 and found out that there was little to no difference when we use Laplace correction for small amount of corpus. However, when we increase the size of the corpus there is a minor change in the accuracy. Since Laplace correction is used to handle zero value we believe that it won't provide a significant change in the accuracy.

```
With laplace correction and cleaning 0.8879227053140096
```

Figure 5: Result with text cleaning and laplace

- For **2.c** we trained data with and without Laplace correction with out cleaning the data

```
With laplace correction and no clearning:  0.8946859903381642
Without laplace correction and no cleaning:  0.8946859903381642
```

Figure 6: Results when we don't perform text cleaning

- For **3.a** and **3.b** We created classification report for Laplace correction and without Laplace correction and also with test cleaning and without test cleaning

```
Classification Report for text cleaning without laplace correction
        precession     recall   f1score    support
for 0:-       0.936     0.915     0.925     0.709
for 1:-       0.787     0.835     0.81     0.291
accuracy is  0.8927536231884058

Classification Report for text cleaning with laplace correction
        precession     recall   f1score    support
for 0:-       0.929     0.914     0.922     0.709
for 1:-       0.787     0.82     0.803     0.291
accuracy is  0.8879227053140096

Classification Report for non text cleaning without laplace correction
        precession     recall   f1score    support
for 0:-       0.932     0.921     0.926     0.709
for 1:-       0.804     0.829     0.816     0.291
accuracy is  0.8946859903381642

Classification Report for non text cleaning with laplace correction
        precession     recall   f1score    support
for 0:-       0.929     0.923     0.926     0.709
for 1:-       0.811     0.824     0.817     0.291
accuracy is  0.8946859903381642
```

Figure 7: Classification Report for different parameters

From the output we are getting we can say that text cleaning is not giving that much improvement in our model. Because the number of features we are using is 425 which means the top 425 words occurring in the corpus. The stop-words we are using will be that subset of these 425 words and the remaining words will be the features which we are using to determine whether a mail is a spam or not. So both the models(with and without text cleaning) have features which are determine whether a mail is spam or not.

3 Documentation: Code for the Naive Bayes Algorithm

We created a function which randomly splits the data in the dataset. We have made a function named `createdata(data, y, boolean = True)` function which will return a Dictionary. The Dictionary has a word and the count of that word in our corpus. The word can not be a stop word if we set the value of the boolean parameter to True otherwise words can be stop words as well. `createdata(data, y, boolean=True)` generates features for a specific specific `y` only say spam or ham. The `attribute(Dict,size)` function return the top n words of `SPAM` and `HAM` create a set of those words and return those words. This will be the features in our corpus. The `nparray(spreadsheet,features)` function is used to create a list `x` and a list `y` form corpus and the features. list `x` and list `y` will be used for training as well as testing.

Now lets talk about our Naive Bayes classifier from scratch. Here we are creating a fit function which will return a Dictionary. The dictionary contains the total count of corpus data. total count of an output `y` say spam or not spam is also included inside the dictionary. Inside the Dictionary there are two dictionary for spam and ham labeled as 0 and 1. Inside these sub dictionary for a particular features we are storing feature index then possible outcome for that features and then we save the count of that.

This will be our fit function we will use this function to predict for our testing data.

The `prob(dict,x,currentclass,laplace)` function is used to find $\mathbb{P}(X|y)$, the probability of X given Y *i.e* `currentclass`. Here, $X = [X_1, X_2, \ldots, X_n]$. So due to our assumption of conditional independence , we have

$$\mathbb{P}(X|Y) = \mathbb{P}(X_1|Y) \cdot \mathbb{P}(X_2|Y) \cdot \cdots \cdot \mathbb{P}(X_n|Y)$$

and,

$$\mathbb{P}(Y|X) \cdot \mathbb{P}(X) = \mathbb{P}(X|Y) \cdot \mathbb{P}(Y)$$

Since for a single test data, we take decisions based on which is more among $\mathbb{P}(\texttt{SPAM}|X)$ and $\mathbb{P}(\texttt{HAM}|X)$ where X does not change, we can instead check the values of $\mathbb{P}(\texttt{SPAM}|X) \cdot \mathbb{P}(X)$ and $\mathbb{P}(\texttt{HAM}|X) \cdot \mathbb{P}(X)$ instead.

The function `predictsinglerow(dict,x,laplace)` will check the probability of X for different Y i.e `SPAM` and `HAM`. It will return the Y which has the highest probability.

The function `predictions(dict,x,laplace)` will be used to predict for the whole `X_test` data to genereate `Y_pred`.

We created the function `Returnaccuracy(true, pred)`, `precision(true, pred)`, `recall(true, pred)`, `support(true, pred)`, `f1score(true, pred)` for our classification report and we have used the standard formula to create these functions.

The function `accuracyontenrandomsplit(spreadsheet,count,laplace=0)` returns the maximum accuracy out of the 10 randomly split dataset.

Here in many places you will see `laplace` if the value is set to 0 it means that we are not using `laplace` correction if the value is 1 or greater than 1 it means that we are using `laplace` correction.