CS5691: Pattern Recognition and Machine Learning

Assignment 3- Spam Classifier

Name: Anukul Dewangan Roll No: CS22M016

SPAM or HAM?

In this assignment, you will build a spam classifier from scratch. No training data will be provided. You are free to use whatever training data that is publicly available/does not have any copyright restrictions (You can build your own training data as well if you think that is useful). You are free to extract features as you think will be appropriate for this problem. The final code you submit should have a function/procedure that when invoked will be able to automatically read a set of emails from a folder titled test in the current directory.
Each _le in this folder will be a test email and will be named 'email#.txt' ('email1.txt', 'email2.txt', etc). For each of these emails, the classifier should predict +1 (spam) or 0 (non-Spam). You are free to use whichever algorithm learned in the course to build a classifier(or even use more than one). The algorithms (except SVM) need to be coded from scratch. Your report should clearly detail information relating to the data set chosen, the features extracted and the exact algorithm/procedure used for training including hyperparameter tuning/kernel selection if any. The performance of the algorithm will be based on the accuracy of the test set.

Solution: I have used Naïve Bayes Algorithm for solving the Spam Classification Problem since a Naïve Bayes text Classifier is based on the Bayes theorem which helps us compute the conditional probabilities of occurrence of two events based on the probabilities of occurrence of each element. Therefore, those probabilities are extremely useful

Data set used for training and testing Naïve Bayes Algorithm for spam classification: 'spam.csv'.

Percentage of ham/spam emails in this dataset:

```
ham     0.865937
spam    0.134063
Name: Label, dtype: float64
```

We can see that 86.57 percent of our emails are ham and 13.4 percent of emails are spam.

Spam Classifier Accuracy using naïve bayes algorithm:

```
Accuracy on test Data: 97.4789237668162
Number of Incorrect Prediction: 28
number of correct Prediction: 1072
```

My Approach:

Step 1: Data pre-processing/ Data cleaning: Remove punctuations and convert every email into lowercase.

Step 2: Now, We are going to split out the data set into a training set and a testing set. I have used 80% dataset for training and 20% for testing.

Step 3: Creating a dictionary for training emails, testing emails, spam emails, and non-spam emails that consist of only unique words. We transform each email in the message column into a list by splitting the string at the space character and appending the list unique words.

Step 4: calculate prior Probablities of p(spam =1) and p(spam =0)

P(spam =1) = number of emails flagged 'spam' / total number of emails
P(ham = 0) = number of emails flagged 'ham' / total number of emails.

Step 4: Take the input string, first do data cleaning, and create a dictionary that contains words that are present in the input string and our training dictionary, ignore words that are not present in our training dictionary.

Example: Input string "lucky win lucky joy kick "

Calculate p(lucky/label=1) , p(win/label=1),p(joy/label=1),p(kick/label=1)
P(luck/label=0), p(win/label =0), p(joy/label = 0), p(kick/label =0)

P(label=1) = p(label =1) *p(lucky/label=1) * p(win/label=1)*p(joy/label=1)*p(kick/label=1)
P(label =0) = p(label =0) *p(lucky/label=0) * p(win/label=0)*p(joy/label=0)*p(kick/label=0)

If(P(label=1) > p(label = 0):
return 'This email is a Spam'
else:
return 'This email is not a spam mail
'.

Below is the result of some emails :