NAÏVE BAYES CLASSIFIER

1. Brief Description of Model and Implementation

Naive Bayes classifiers are a popular statistical technique for e-mail filtering. They typically use bag of words features to identify spam e-mail, an approach commonly used in text classification.

Naive Bayes classifiers work by correlating the use of tokens (typically words, or sometimes other things), with spam and non-spam e-mails and then using Bayes' theorem to calculate a probability that an email is or is not spam.

The Naive Bayes algorithm relies on Bayes Rule. This algorithm will classify each object by looking at all of it's features individually. Bayes Rule below shows us how to calculate the posterior probability for just one feature. The posterior probability of the object is calculated for each feature and then these probabilities are multiplied together to get a final probability. This probability is calculated for the other class as well. Which ever class has the greater probability that ultimately determines what class the object is in.

The object is an email and the features are the unique words in the email. Thus, there is a posterior probability calculated for each unique word in the emails. Plugging this into Bayes Rule, our formula will look something like this:

$$P(SPAM/ Word) = [P(Word/SPAM) \times P(SPAM)] / P(Word)$$

$$P(Word) = P(Word/Spam) * P(Spam) + P(Word/ Not Spam) * P(Not Spam)$$

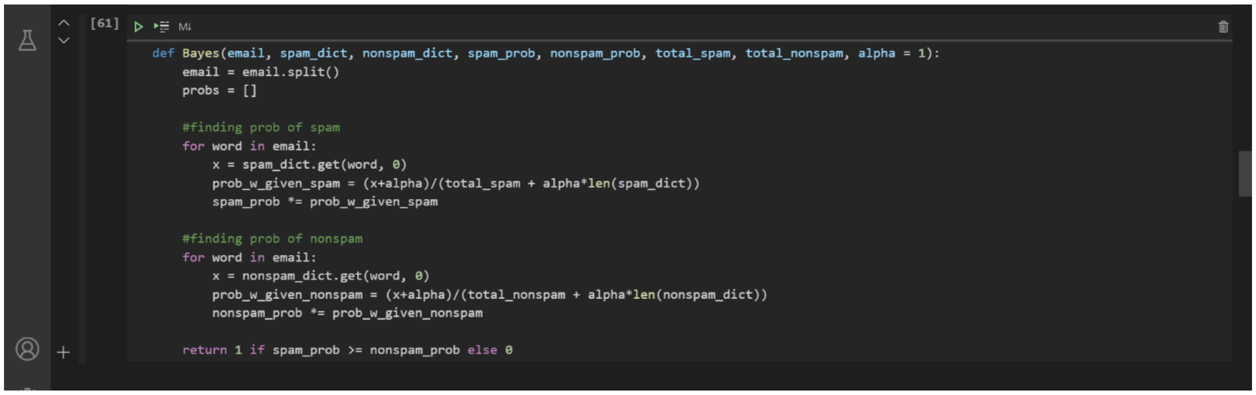
Implementation:

1. Reading and cleaning the dataset – Used an array of common stop words to remove all those words which do not contribute to the sentiment of the email.

2. Creating features and class array.

3. Using the spam and non spam mail lists classified using the class specified in the text file to create the unique spam and non spam word lists and also find their total counts in order to use them to find the probabilities.

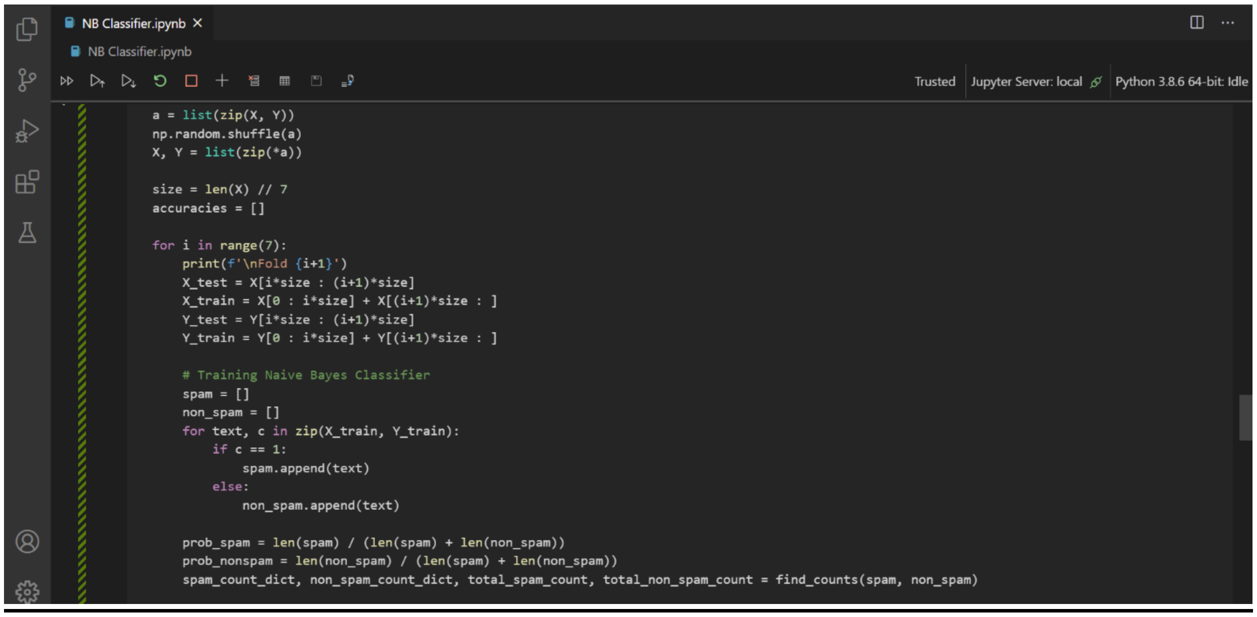
4. Calculating the probability of each email using it's words in order to classify it as a spam or non-spam email. We also apply Laplace smoothing to remove the issue created in the Zero Frequency case.



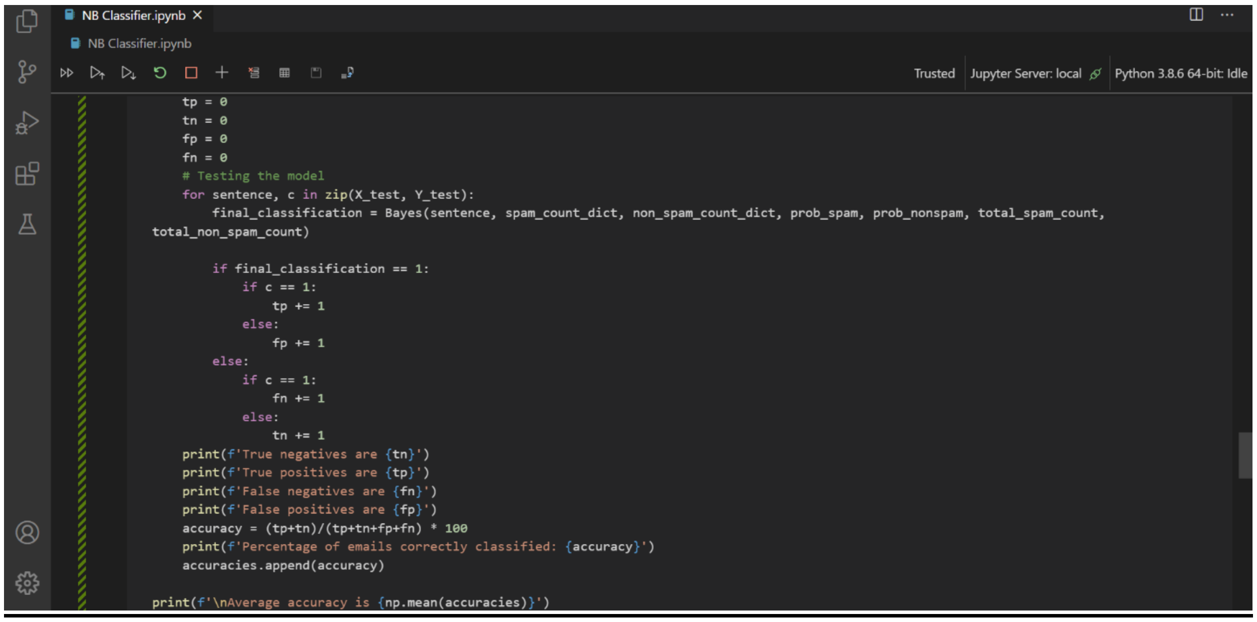
```
def Bayes(email, spam_dict, nonspam_dict, spam_prob, nonspam_prob, total_spam, total_nonspam, alpha = 1):
    email = email.split()
    probs = []

    #finding prob of spam
    for word in email:
        x = spam_dict.get(word, 0)
        prob_w_given_spam = (x+alpha)/(total_spam + alpha*len(spam_dict))
        spam_prob *= prob_w_given_spam

    #finding prob of nonspam
    for word in email:
        x = nonspam_dict.get(word, 0)
        prob_w_given_nonspam = (x+alpha)/(total_nonspam + alpha*len(nonspam_dict))
        nonspam_prob *= prob_w_given_nonspam

    return 1 if spam_prob >= nonspam_prob else 0
```

5. Firstly, shuffling the given dataset and then applying 7-fold cross validation. In each fold, we divide the shuffled data into training and testing data and then train the model using the training data.
   Finally, we classify the testing data and match with the initial class to find the number of true positives, true negatives, false positives and false negatives and finally use these values to find the accuracy in the current fold and then use these 7 accuracy values obtained in each fold to find out the overall average accuracy.

   The code snippet implementing it is :



```
a = list(zip(X, Y))
np.random.shuffle(a)
X, Y = list(zip(*a))

size = len(X) // 7
accuracies = []

for i in range(7):
    print(f'\nFold {i+1}')
    X_test = X[i*size : (i+1)*size]
    X_train = X[0 : i*size] + X[(i+1)*size : ]
    Y_test = Y[i*size : (i+1)*size]
    Y_train = Y[0 : i*size] + Y[(i+1)*size : ]

    # Training Naive Bayes Classifier
    spam = []
    non_spam = []
    for text, c in zip(X_train, Y_train):
        if c == 1:
            spam.append(text)
        else:
            non_spam.append(text)

    prob_spam = len(spam) / (len(spam) + len(non_spam))
    prob_nonspam = len(non_spam) / (len(spam) + len(non_spam))
    spam_count_dict, non_spam_count_dict, total_spam_count, total_non_spam_count = find_counts(spam, non_spam)
```



```
    tp = 0
    tn = 0
    fp = 0
    fn = 0
    # Testing the model
    for sentence, c in zip(X_test, Y_test):
        final_classification = Bayes(sentence, spam_count_dict, non_spam_count_dict, prob_spam, prob_nonspam, total_spam_count,
total_non_spam_count)

        if final_classification == 1:
            if c == 1:
                tp += 1
            else:
                fp += 1
        else:
            if c == 1:
                fn += 1
            else:
                tn += 1
    print(f'True negatives are {tn}')
    print(f'True positives are {tp}')
    print(f'False negatives are {fn}')
    print(f'False positives are {fp}')
    accuracy = (tp+tn)/(tp+tn+fp+fn) * 100
    print(f'Percentage of emails correctly classified: {accuracy}')
    accuracies.append(accuracy)

print(f'\nAverage accuracy is {np.mean(accuracies)}')
```

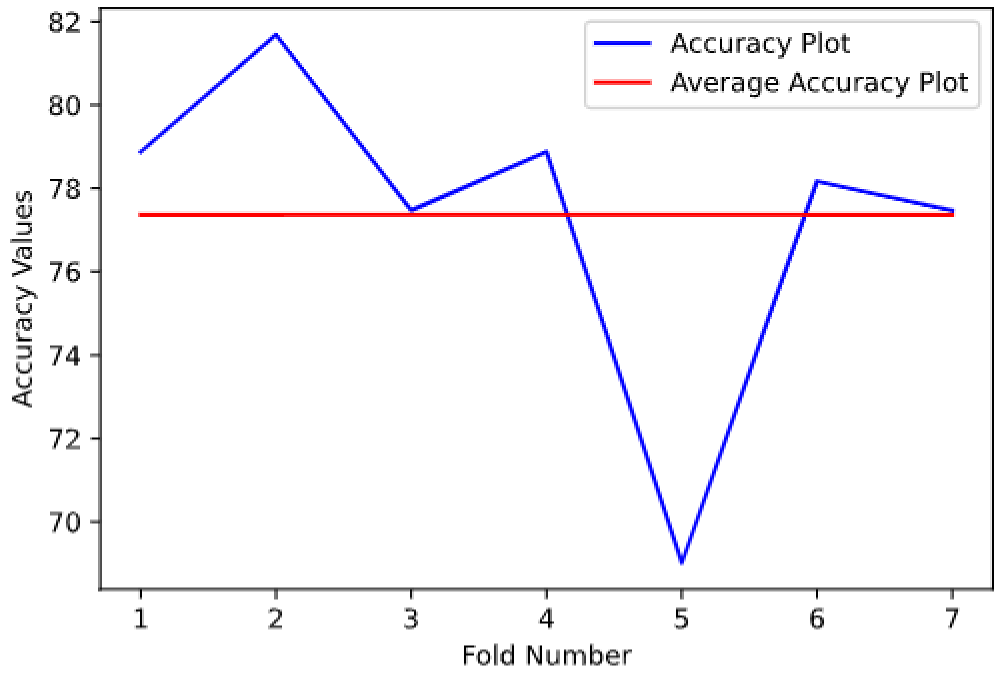
2. Accuracy of your model over each fold and the overall average accuracy.

Results for the different folds after shuffling the dataset:

```
Fold 1
True negatives are 50
True positives are 62
False negatives are 6
False positives are 24
Percentage of emails correctly classified: 78.87323943661971

Fold 2
True negatives are 47
True positives are 69
False negatives are 4
False positives are 22
Percentage of emails correctly classified: 81.69014084507043

Fold 3
True negatives are 42
True positives are 68
False negatives are 8
False positives are 24
Percentage of emails correctly classified: 77.46478873239437

Fold 4
True negatives are 47
True positives are 65
False negatives are 6
False positives are 24
Percentage of emails correctly classified: 78.87323943661971

Fold 5
True negatives are 42
True positives are 56
False negatives are 11
False positives are 33
Percentage of emails correctly classified: 69.01408450704226
```

```
Fold 6
True negatives are 49
True positives are 62
False negatives are 9
False positives are 22
Percentage of emails correctly classified: 78.16901408450704

Fold 7
True negatives are 50
True positives are 60
False negatives are 12
False positives are 20
Percentage of emails correctly classified: 77.46478873239437

Average accuracy is 77.364185110664
```

Plot for :



3. Major limitations of the Naive Bayes classifier

1. The main limitation of Naive Bayes is the **assumption of independent predictor features**. Naive Bayes implicitly assumes that all the attributes are mutually independent. In real life, it's almost impossible that we get a set of predictors that are completely independent or one another.

2. If a categorical variable has a category in the test dataset, which was not observed in training dataset, then the model will assign a 0 (zero) probability and will be unable to make a prediction. This is often known as **Zero Frequency**. To solve this, we can use a smoothing technique. We have used Laplace smoothing in order to counter this "Zero Frequency" situation.