CA-3

Machine Learning Foundation

Report: Email Spam Classification Dataset CSV

Name: Saurabh Tripathi

Section: KM032

Regn.No: 11807532

Roll Nu: B70

Project Repo: https://github.com/beingsaurabh/spam-email

To: Asst. Prof. Rachna Kohar

Lovely Professional University

PUNJAB INDIA

Code Link:

https://colab.research.google.com/drive/1oq8ibS-X97rZIsBfK6mshimqbT25CULw?usp=sharing

ABSTRACT:

Introduction

This is a csv file containing related information of 5172 randomly picked email files and their respective labels for spam or not-spam classification.

About the Dataset

The csv file contains 5172 rows, each row for each email. There are 3002 columns. The first column indicates Email name. The name has been set with numbers and not recipients' name to protect privacy. The last column has the labels for prediction : 1 for spam, 0 for not spam. The remaining 3000 columns are the 3000 most common words in all the emails, after excluding the non-alphabetical characters/words. For each row, the count of each word(column) in that email(row) is stored in the respective cells. Thus, information regarding all 5172 emails are stored in a compact data frame rather than as separate text files.

To complete this assignment, I have taken tree classification algorithm:

1. Naïve Bayes Classifier

2. Support Vector Classifier (SVC)

3. Random Forest Classifier

Creating the NB Model

In this project we are classifying mails typed in by the user as either 'Spam' or 'Not Spam'. Our original dataset was a folder of 5172 text files containing the emails.

Now let us understand why we have separated the words from the mails. This is because, this is a text-classification problem. When a spam classifier looks at a mail, it searches for potential words that it has seen in the previous spam emails. If it finds a majority of those words, then it labels it as 'Spam'. **Why did I say majority ? -->**

CASE 1 : suppose let's take a word 'Greetings'. Say, it is present in both 'Spam' and 'Not Spam' mails.

CASE 2 : Let's consider a word 'lottery'. Say, it is present in only 'Spam' mails.

CASE 3 : Let's consider a word 'cheap'. Say, it is present only in spam.

If now we get a test email, and it contains all the three words mentioned above, there's high probability that it is a 'Spam' mail.

The most effective algorithm for text-classification problems is the Naive Bayes algorithm, that works on the classic Bayes' theorem. This theorem works on every individual word in the test data to make predictions(the conditional probability with higher probability is the predicted result).

Say, our test email(S)is, *"You have won a lottery"*

HOW NAIVE BAYES WORKS ON THIS DATA -->

P(S) = P('You') *P('have')* P('won') *P('a')* P('lottery') __ 1

Therefore, P(S|Spam) =
P('You'|Spam) *P('have'|Spam)* P('won'|Spam) *P('a'|Spam)* P('lottery'|Spam) __ 2

Same calculation for P(S|Not_Spam) __ 3

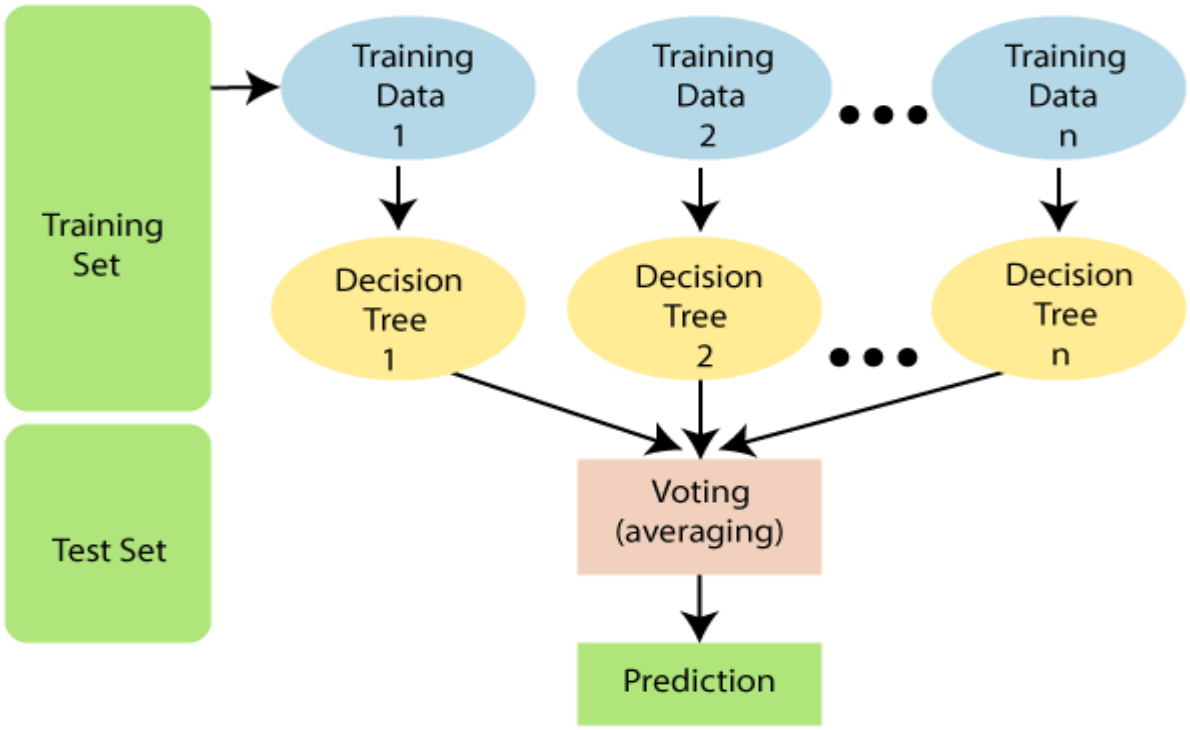
If 2 > 3, then 'Spam' Else, 'Not_Spam'.

WHAT IF THE PROBABILITY IS ZERO ? Here comes the concept of Laplace Smoothing, where P(words) = (word_count + 1)/(total_no_of_words + no_of_unique_words)

Here, we'll work on the existing Multinomial Naive Bayes classifier (under scikit-learn). To further understand how well Naive Bayes works for text-classification, we'll use another standard classifier, SVC, to see how the two models perform.

Random Forest Classifier: Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

Following Diagram shows how random Forest Classifier works..

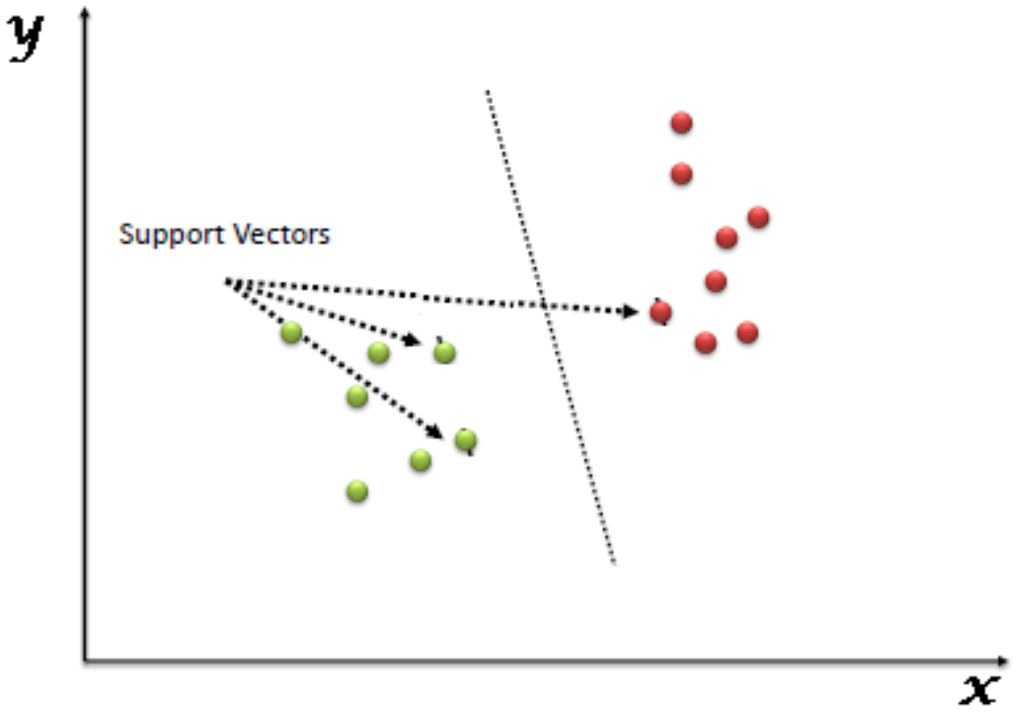


Support Vector Classifier (SVC):

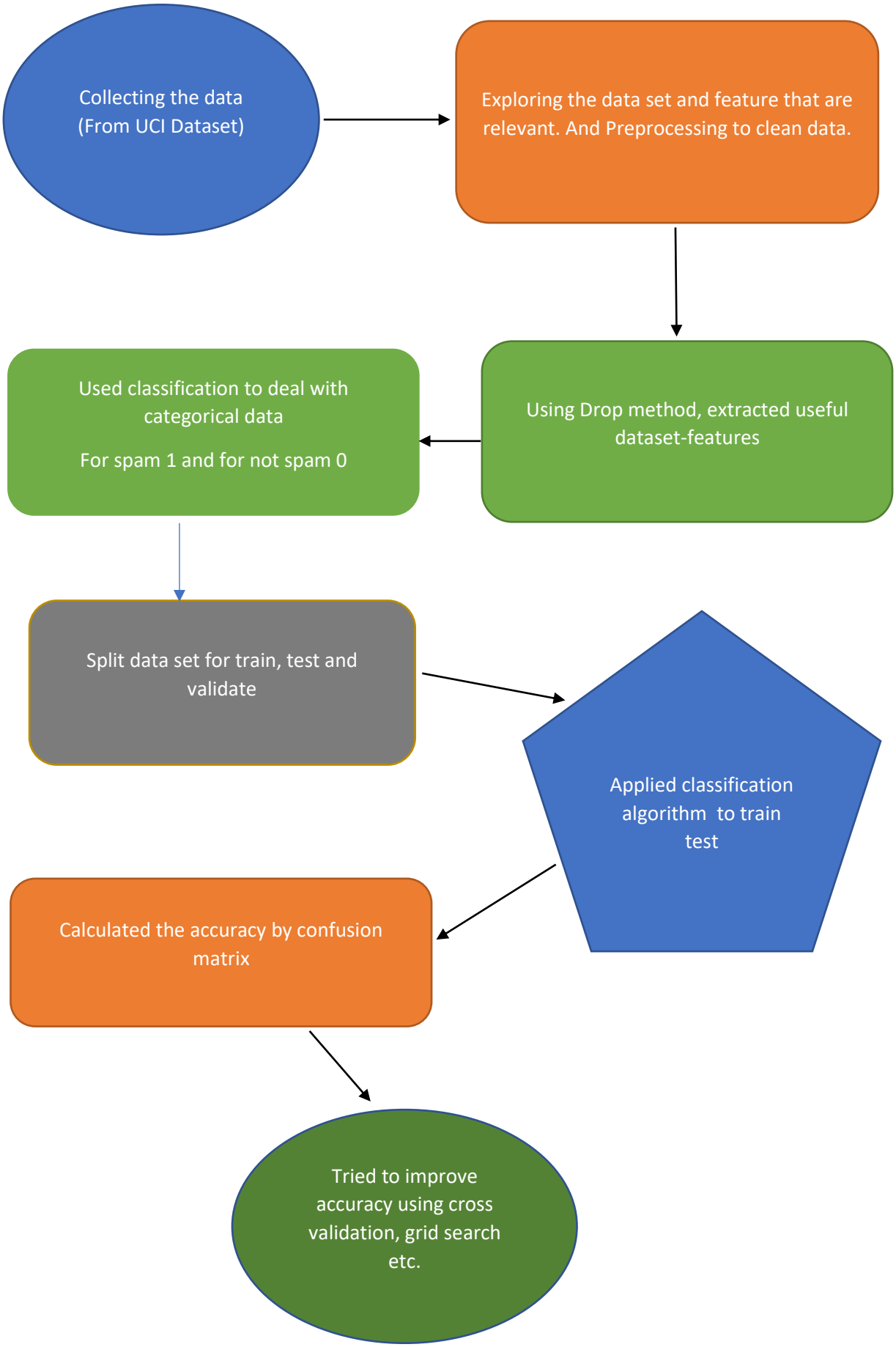
Support Vector Machines

Support Vector Machine is the most sought-after algorithm for classic classification problems. SVMs work on the algorithm of Maximal Margin, i.e., to find the maximum margin or threshold between the support vectors of the two classes (in binary classification). The most effective Support vector machines are the soft maximal margin classifier, that allows one misclassification, i.e., the model starts with low bias(slightly poor performance) to ensure low variance later.

“Support Vector Machine” (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well.


y
Support Vectors
x

Methodology:



Result and Discussion:

Naïve Bayes gives the accuracy of 94%

```
Accuracy Score for Naive Bayes :  0.9466357308584686
Accuracy % for Naive Bayes     :  94.66357308584686


Confusion Matrix
[[885  14]
 [ 55 339]]
              precision    recall  f1-score   support

           0       0.94      0.98      0.96       899
           1       0.96      0.86      0.91       394

    accuracy                           0.95      1293
   macro avg       0.95      0.92      0.94      1293
weighted avg       0.95      0.95      0.95      1293
```

SVC gives the accuracy of 90%

```
Accuracy % for SVC :  90.71925754060325
```

```
Confusion Matrix
[[905  85]
 [ 35 268]]
              precision    recall  f1-score   support

           0       0.96      0.91      0.94       990
           1       0.76      0.88      0.82       303

    accuracy                           0.91      1293
   macro avg       0.86      0.90      0.88      1293
weighted avg       0.92      0.91      0.91      1293
```

Random Forest Classifier gives the accuracy of 97%

```
Accuracy % of Random Forest Classifier :  97.4477958236659
```

```
Confusion Matrix
[[925  18]
 [ 15 335]]
              precision    recall  f1-score   support

           0       0.98      0.98      0.98       943
           1       0.95      0.96      0.95       350

    accuracy                           0.97      1293
   macro avg       0.97      0.97      0.97      1293
weighted avg       0.97      0.97      0.97      1293
```

----------Concluded----------