Email Spam Classification Using Machine Learning

Harshitha Pothula
&
Ssrk Kasyap

Contents

- Introduction
- Technologies
- Libraries
- Machine Learning
- Data Set
- Problem Definition
- Algorithms
- Conclusion

INTRODUCTION

In today's globalized world, email is a primary source of communication. This communication can vary from personal, business, corporate to government. With the rapid increase in email usage, there has also been increase in the SPAM emails. SPAM emails, also known as junk email involves nearly identical messages sent to numerous recipients by email. Apart from being annoying, spam emails can also pose a security threat to computer system. It is estimated that spam cost businesses on the order of $100 billion in 2007. In this project, we use text mining to perform automatic spam filtering to use emails effectively. We try to identify patterns using Data-mining classification algorithms to enable us classify the emails as HAM or SPAM.

TECHNOLOGIES

Technologies Used:

- **Python**: Python is an interpreted, object-oriented, high-level programming language with dynamic semantics developed by Guido van Rossum

Libraries:

1. Numpy
2. Pandas
3. Sklearn
4. Matplotlib

Libraries

- **NumPy:-** NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. Moreover, NumPy forms the foundation of the Machine Learning stack.
- **Pandas:-** Pandas is one of the tools in Machine Learning which is used for data cleaning and analysis. It has features which are used for exploring, cleaning, transforming and visualizing from data.
- **Sklearn:-** Sklearn is intended to support research and teaching in NLP or closely related areas, including empirical linguistics, cognitive science, artificial intelligence, information retrieval and machine learning.
- **Matplotlib:-** Matplotlib is a low-level library of Python which is used for data visualization. It is easy to use and elulates MATLAB like graphs and visualization. This library is built on the top of NumPy arrays and consist of several plots like line chart, bar chart, histogram, etc.

Machine Learning

Arthur Samuel, an early American leader in the field of computer gaming and artificial intelligence, coined the term "Machine Learning" in 1959 while at IBM. He defined machine learning as "the field of study that gives computers the ability to learn without being explicitly programmed ".

- Machine learning is programming computers to optimize a performance criterion using example data or past experience.
- The field of study known as machine learning is concerned with the question of how to construct computer programs that automatically improve with experience.

Dataset

A machine learning dataset is a collection of data that is used to train the model. A dataset acts as an example to teach the machine learning algorithm how to make predictions. dataset as "a collection of data that is treated as a single unit by a computer". This means that a dataset contains a lot of separate pieces of data but can be used to train an algorithm with the goal of finding predictable patterns inside the whole dataset.

How to train the data?

- Al training data will vary depending on whether you're using supervised or unsupervised learning. Unsupervised learning uses unlabeled data. Models are tasked with finding patterns (or similarities and deviations) in the data to make inferences and reach conclusion.
- With supervised learning, on the other hand, humans must tag, label, or annotate the data to their criteria, in order to train the model to reach the desired conclusion (output)Labeled data is shown in the examples above, where the desired outputs are predetermined.

Problem Definition

- Short Message (SMS) and email has grown into a multi-billion dollars commercial industry.
- SMS spam is still not as common as email spam.
- SMS Spam is showing growth, and in 2012 in parts of Asia up to 30% of text messages was spam.

Algorithms

Different algorithms that can be used for Email Spam Detection are:

1. Deep Learning
2. Naive Bayes
3. Support Vector Machine
4. K-Nearest Neighbour
5. Random Forest
6. Multinomial naive

Conclusion

Spam is a major problem in today's world. Spam messages are the most unwanted messages the end user clients receive in our daily lives. Spam emails are available nothing but an ad for any company, any kind of virus etc. It will be too much. It is easy for hackers to access our system using these spam emails.

THANK YOU

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
```

```
df = pd.read_csv("/content/emails.csv")
df.head(2)
```

	Email No.	the	to	ect	and	for	of	a	you	hou	...	connevey	jay	valued	lay	infrastructure	military	allowing	ff	dry	Predict
0	Email 1	0	0	1	0	0	0	2	0	0	...	0	0	0	0	0	0	0	0	0	

```
df.isnull().sum()
```

```
Email No.     0
the           0
to            0
ect           0
and           0
             ..
military      0
allowing      0
ff            0
dry           0
Prediction    0
Length: 3002, dtype: int64
```

```
df.describe()
```

	the	to	ect	and	for	of	a	you	hou	
count	5172.000000	5172.000000	5172.000000	5172.000000	5172.000000	5172.000000	5172.000000	5172.000000	5172.000000	517
mean	6.640565	6.188128	5.143852	3.075599	3.124710	2.627030	55.517401	2.466551	2.024362	1
std	11.745009	9.534576	14.101142	6.045970	4.680522	6.229845	87.574172	4.314444	6.967878	1
min	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	1.000000	1.000000	0.000000	1.000000	0.000000	12.000000	0.000000	0.000000	
50%	3.000000	3.000000	1.000000	1.000000	2.000000	1.000000	28.000000	1.000000	0.000000	
75%	8.000000	7.000000	4.000000	3.000000	4.000000	2.000000	62.250000	3.000000	1.000000	1
max	210.000000	132.000000	344.000000	89.000000	47.000000	77.000000	1898.000000	70.000000	167.000000	22

8 rows × 3001 columns

Creating the NB Model

```
X = df.iloc[:,1:3001]
X
```

	the	to	ect	and	for	of	a	you	hou	in	...	enhancements	connevey	jay	valued	lay	infrastructure	military
0	0	0	1	0	0	0	2	0	0	0	...	0	0	0	0	0	0	0
1	8	13	24	6	6	2	102	1	27	18	...	0	0	0	0	0	0	0
2	0	0	1	0	0	0	8	0	0	4	...	0	0	0	0	0	0	0
3	0	5	22	0	5	1	51	2	10	1	...	0	0	0	0	0	0	0
4	7	6	17	1	5	2	57	0	9	3	...	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
5171	22	24	5	1	6	5	148	8	2	23	...	0	0	0	0	0	0	0

```
Y = df.iloc[:,-1].values
Y
```

```
array([0, 0, 0, ..., 1, 1, 0])
```

```
train_x,test_x,train_y,test_y = train_test_split(X,Y,test_size = 0.25)
```

Naive Bayes

```
mnb = MultinomialNB(alpha=1.9)         # alpha by default is 1. alpha must always be > 0.
# alpha is the '1' in the formula for Laplace Smoothing (P(words))
mnb.fit(train_x,train_y)
y_pred1 = mnb.predict(test_x)
print("Accuracy Score for Naive Bayes : ", accuracy_score(y_pred1,test_y))
```

```
Accuracy Score for Naive Bayes :  0.9365815931941222
```

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.

Let us see the model performance.

```
svc = SVC(C=1.0,kernel='rbf',gamma='auto')
svc.fit(train_x,train_y)
y_pred2 = svc.predict(test_x)
print("Accuracy Score for SVC : ", accuracy_score(y_pred2,test_y))
```

```
Accuracy Score for SVC :  0.888631090487239
```

As expected, SVM's performance is slightly poorer than Multinomia Naive Bayes

Random Forests (Bagging)

Ensemble methods turn any feeble model into a highly powerful one. Let us see if ensemble model can perform better than Naive Bayes

```
rfc = RandomForestClassifier(n_estimators=100,criterion='gini')
# n_estimators = No. of trees in the forest
# criterion = basis of making the decision tree split, either on gini impurity('gini'), or on infromation gain('entropy')
rfc.fit(train_x,train_y)
y_pred3 = rfc.predict(test_x)
print("Accuracy Score of Random Forest Classifier : ", accuracy_score(y_pred3,test_y))
```

```
Accuracy Score of Random Forest Classifier :  0.9682907965970611
```

Ending notes:

This was a purely comparative study to check the workability of the dataset that I created, and to check how conventional models perform on my dataset. In my next kernel, I will show the code behind the extraction of this dataset from the raw text files.

✓ 2s completed at 9:59 PM

Email Spam Classification Report

Introduction

We've all been the recipient of spam emails before. Spam mail, or junk mail, is a type of email that is sent to a massive number of users at one time, frequently containing cryptic messages, scams, or most dangerously, phishing content.

While spam emails are sometimes sent manually by a human, most often, they are sent using a bot. Most popular email platforms, like Gmail and Microsoft Outlook, automatically filter spam emails by screening for recognizable phrases and patterns. A few common spam emails include fake advertisements, chain emails, and impersonation attempts. While these built-in spam detectors are usually pretty effective, sometimes, a particularly well-disguised spam email may fall through the cracks, landing in your inbox instead of your spam folder.

Clicking on a spam email can be dangerous, exposing your computer and personal information to different types of malware. Therefore, it's important to implement additional safety measures to protect your device, especially when it handles sensitive information like user data.

In this , we'll use Python to build an email spam detector. Then, we'll use machine learning to train our spam detector to recognize and classify emails into spam and non-spam.

Prerequisites

First, we'll import the necessary dependencies. NumPy is a Python library used for working with arrays.It also has functions for working in the domain of linear algebra, fourier transform, and matrices.NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.

Pandas is a Python library used for working with data sets.It has functions for analyzing, cleaning, exploring, and manipulating data.Pandas allows us to analyze big data and make conclusions based on statistical theories.Pandas can clean messy data sets, and make them readable and relevant.

Scikit-learn, also called Sklearn, is a robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling, including classification, regression, clustering, and dimensionality reduction via a consistent interface.

Run the command below to import the necessary dependencies:

```
import numpy as np # linear algebra

import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

from sklearn.model_selection import train_test_split

from sklearn.naive_bayes import MultinomialNB

from sklearn.svm import SVC

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score
```

Getting started

To get started, first, run the code below:

```
df = pd.read_csv("/content/emails.csv")
```

In the code above, we created an “emails.csv” file, which we’ll turn into a data frame and save to our folder spam. A data frame is a structure that aligns data in a tabular fashion in rows and columns.

Python `train_test_split()`

We'll use a train-test split method to train our email spam detector to recognize and categorize spam emails. The train-test split is a technique for evaluating the performance of a machine learning algorithm. We can use it for either classification or regression of any supervised learning algorithm.

The procedure involves taking a dataset and dividing it into two separate datasets. The first dataset is used to fit the model and is referred to as the training dataset. For the second dataset, the test dataset, we provide the input element to the model. Finally, we make predictions, comparing them against the actual output.

- Train dataset: used to fit the machine learning model
- Test dataset: used to evaluate the fit of the machine learning model

In practice, we'd fit the model on available data with known inputs and outputs. Then, we'd make predictions based on new examples for which we don't have the expected output or target values.

To split the data into our two datasets, we'll use scikit-learn's `train_test_split()` method.

Let's say we have 100 records in the loaded dataset. If we specify the test dataset is 30 percent, we'll split 70 records for training and use the remaining 30 records for testing.

Run the command below:

```
train_x,test_x,train_y,test_y = train_test_split(X,Y,test_size = 0.25)
```

The function z_train, z_test,y_train, y_test = train_test_split(z,y,test_size = 0.25) divides columns z and y into z_train for training inputs, y_train for training labels, z_test for testing inputs, and y_test for testing labels.

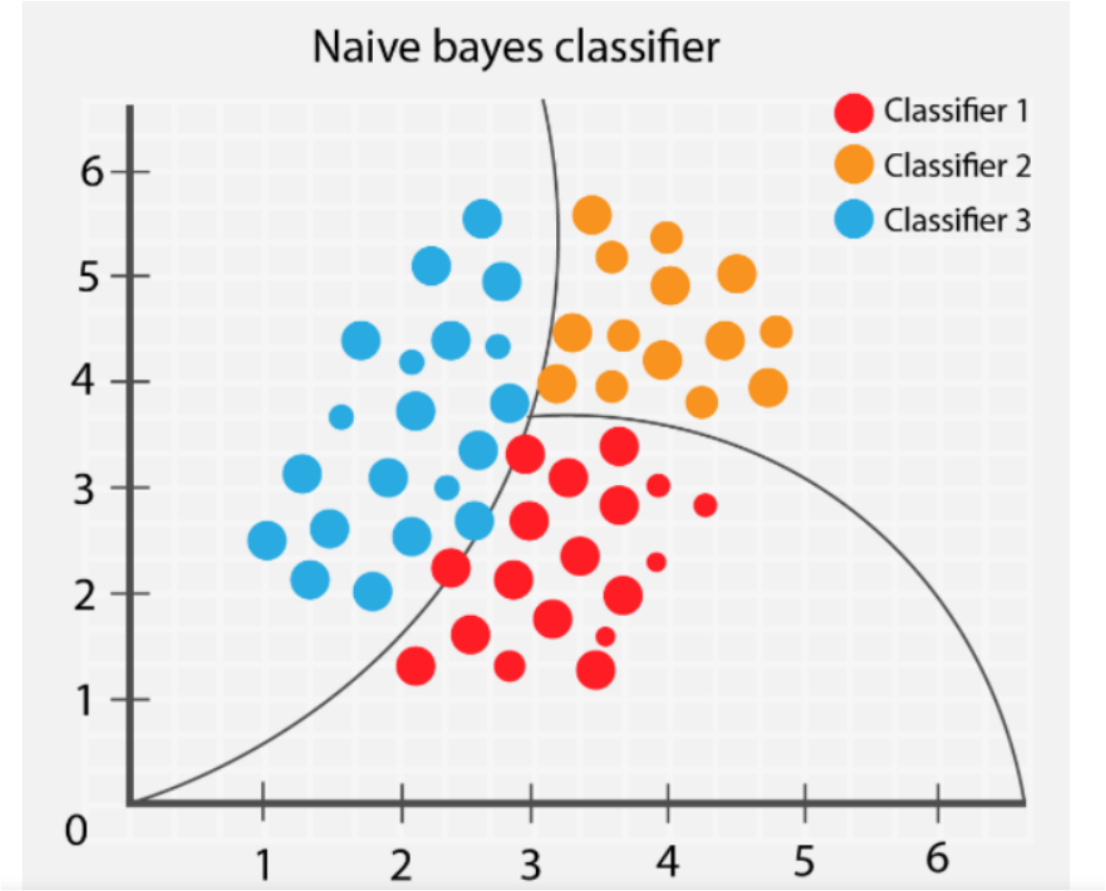
test_size=0.25 sets the testing set to 25 percent of z and y.

Building the model

Naive Bayes:

The Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems.It is mainly used in *text classification* that includes a high-dimensional training dataset.The Naïve Bayes Classifier is one of the simple and most effective

Classification algorithms which helps in building the fast machine learning models that can make quick predictions.It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.

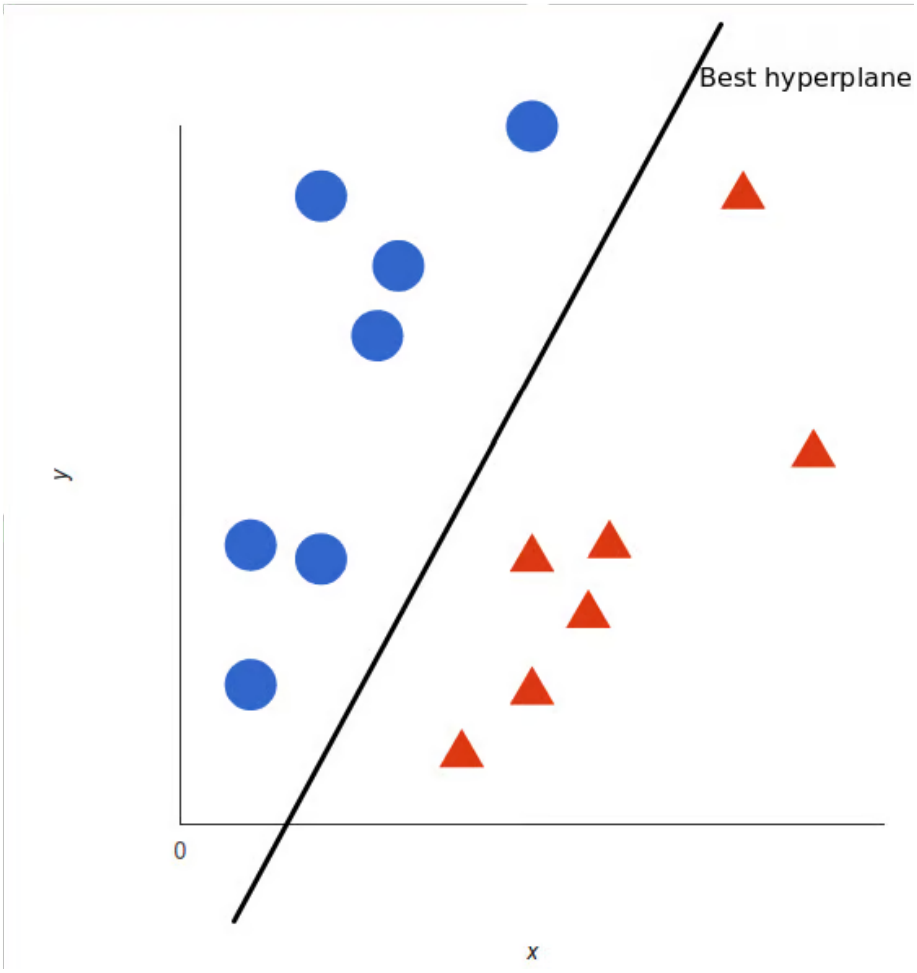


Let’s create an Naive Bayes model with the code below:

```
mnb = MultinomialNB(alpha=1.9)              # alpha by default is 1. alpha must always be > 0.

# alpha is the '1' in the formula for Laplace Smoothing (P(words))

mnb.fit(train_x,train_y)

y_pred1 = mnb.predict(test_x)

print("Accuracy Score for Naive Bayes : ", accuracy_score(y_pred1,test_y))

Accuracy Score for Naive Bayes : 0.9365815931941222
```

Support Vector Machine:

SVM, the support vector machine algorithm, is a linear model for classification and regression. The idea of SVM is simple, the algorithm creates a line, or a hyperplane, which separates the data into classes. SVM can solve both linear and non-linear problems:



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.

Let’s create an SVM model with the code below:

```
svc = SVC(C=1.0,kernel='rbf',gamma='auto')
svc.fit(train_x,train_y)
y_pred2 = svc.predict(test_x)
print("Accuracy Score for SVC : ", accuracy_score(y_pred2,test_y))
```

```
Accuracy Score for SVC : 0.888631090487239
```

model = svm.SVC() assigns svm.SVC() to the model. In the model.fit(features,y_train) function, model.fit trains the model with features and y_train. Then, it checks the prediction against the y_train label and adjusts its parameters until it reaches the highest possible accuracy.

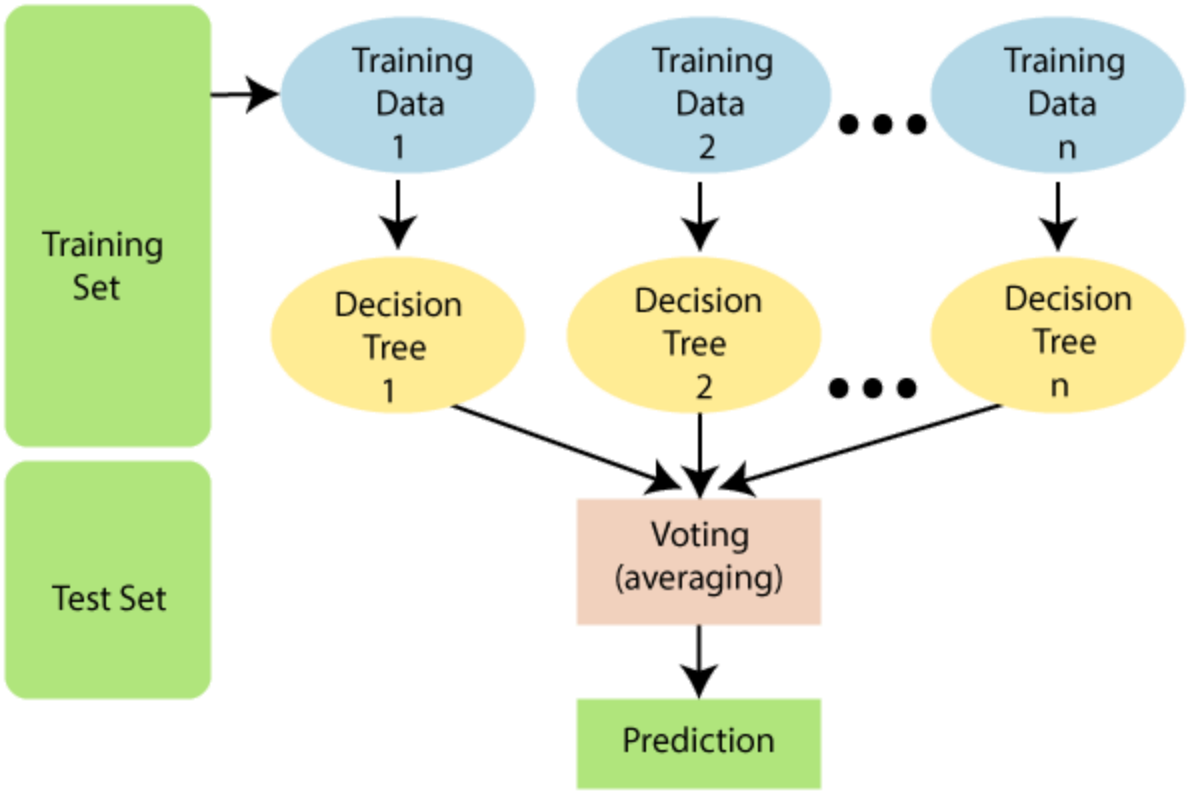
As expected, SVM’s performance is poorer than Naive bayes.

Random Forest:

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of *combining multiple classifiers to solve a complex problem and to improve the performance of the model.*

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.

The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.



Below are some points that explain why we should use the Random Forest algorithm:

- It takes less training time as compared to other algorithms.

- It predicts output with high accuracy, even for the large dataset it runs efficiently.
- It can also maintain accuracy when a large proportion of data is missing.

Let's create an Random Forest model with the code below:

```
rfc = RandomForestClassifier(n_estimators=100,criterion='gini')
# n_estimators = No. of trees in the forest
# criterion = basis of making the decision tree split, either on gini impurity('gini'), or on information gain('entropy')
rfc.fit(train_x,train_y)
y_pred3 = rfc.predict(test_x)
print("Accuracy Score of Random Forest Classifier : ", accuracy_score(y_pred3,test_y))
```

```
Accuracy Score of Random Forest Classifier : 0.9682907965970611
```

Random Forest model is also called as bagging method.
These ensemble methods turn any feeble model into a highly powerful one.
So Random Forest method has the highest accuracy than other methods.

Conclusion:

This system, in addition to lessening the workload, also fixes any false data about the users that they may have. It is a benefit for the users' who's important time and data is preserved, for the Affected users or

authority whose data is immensely important, whose data will be secured from misuse.

We are able to classify email as spam or non spam. With a huge number of emails if people are using the system it will be difficult to handle all the possible mails as our project deals with only a limited amount.

The website is for end users, it is user friendly. Because of the end user it uses without any other help and without any conflicts. The website goal is “Email Spam or Non Spam " using machine learning, related to its use for free and maintenance (coding, updates, uploading data, datasets, etc) cost is less. Many goals was successfully completed and achieved by us.