

PROFESSIONAL TRAINING REPORT

at

SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY

Submitted in partial fulfillment of the requirements for the award of

Bachelor of Engineering degree in

Information Technology

By

T.DIVYA BARATHI

37120024



SCHOOL OF COMPUTING

SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

DECEMBER 2020



**SATHYABAMA INSTITUTE OF SCIENCE AND
TECHNOLOGY**

(Established under Section 3 of UGC Act,1956)



BONAFIDECERTIFICATE

This is to certify that this Project Report is the bonafide work of **T.DIVYA BARATHI (Reg.37120024)** and who carried out the project entitled “**E-MAIL SPAM DETECTION**” under our supervision from July 2020 to December 2020.

Internal Guide

DR.Y.BEVISH JINILA M.E., Ph.D.,

Head of the Department

DR.R.SUBHASHINI M.E.,Ph.D.,

Submitted for Viva voice Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I **T.DIVYA BARATHI (Reg No:37120024)** hereby declare that the Project Report entitled **“E-MAIL SPAM DETECTION”** done by me under the guidance of **Dr. Y.BEVISH JINILA**, M.E., Ph.D., is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Information Technology

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to Board of Management of **SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph.D, Dean, School of Computing** , **Dr.SUBHASHINI M.E., Ph.D.** , Head of the Department of Information and Technology for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. Bevish Jinila M.E., Ph.D.,**for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the Department of Information Technology who were helpful in many ways for the completion of the project.

TRAINING CERTIFICATES:



ABSTRACT

E-mail has become one of the most important forms of communication. By the report of 2020 . Spam is one of the major threats posed to email users. In 2020, 53.95% of all email flows were spam. In this project , A spam detector is built using two Machine Learning methods and evaluate them with test data using different performances metrics and then categorize the e-mails into “spam” and “ham” (non spam) using data sets that are compiled from various sources on the web (5000 spam emails and 6000 non- spam emails). It can be achieved by Data Processing and Data Training models. The Training models involved in this are Naives Baye’s and Random Forest Methods. Following are the metrics used to evaluate the performance of Machine Learning Techniques – Accuracy, Precision, Re-call, F-Score and AUC. The two results are compared with each other Random forest method gives the best result in give very accurate value.

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	VI
	LIST OF FIGURES	IX
	LIST OF ABBREVIATION	X
1	INTRODUCTION	
	1.1 OVERVIEW	11
	1.2 OUTLINE OF THE PROJECT	12
	1.3 PROMBLEM STATEMENT	12
	1.4 OBJECTIVES	13
2	AIM AND SCOPE	
	2.1 AIM	14
	2.2 SCOPE	14
	2.3 LITERATURE REVIEW	14
	2.4 PROPOSED SYSTEM	15
3	ALGORITHMS AND METHODS	
	3.1 OVERVIEW	16
	3.2 ARCHITECTURE DIAGRAM	16
	3.3 ALGORITHM	17
	3.4 PROJECT DESCRIPTION	19
	3.5 METRICS	20
4	SYSTEM IMPLEMENTATION	
	4.1 SYSTEM REQUIREMENTS	21
	4.2 ROLE OF PYTHON	21

	4.3 COMPARATIVE GRAPHS	22
	4.4 RESULTS	22
5	CONCLUSION AND FUTURE WORK	
	5.1 CONCLUSION	25
	5.2 FUTURE WORK	25
	REFERENCES	26
	APPENDICES	26

LIST OF FIGURES

S.NO	DIAGRAM NAME	PAGE NO
3.1	System Architecture	16
4.1	COMPARISON GRAPH	22
4.2	TRAINED DATA SET	24
4.3	NAVIES BAYE METRICS	24
4.4	RANDOM FOREST METHOD	25

LIST OF ABBREVIATIONS

S.NO	ABBEVIATIONS	EXPANSIONS
1	TTS	Text To Speech
2	ASR	Automate Speech Recognition
3	TF-IDF	Term Frequency Inverse Document Frequency
4	TP	True Positive
5	TN	True Negative
6	FP	False Positive
7	FN	False Negative
8	AUC	Area Under ROC
9	ROC	Receiver Operating Characteristic Curve

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW:

Email communications are indispensable nowadays, but the threat of unsolicited junk emails growing drastically. Many surveys by the website views that 40 percent of emails are considered to be spams. Recently the statics collected by the message labs shows that spam rate is over 70 percent and becomes very high.

In 2020, there are estimated to be 4.1 billion email accounts worldwide, and about 196 billion emails are sent each day worldwide. Spam is one of the major threats posed to email users. In 2019, 69.6% of all email flows were spam. Therefore, an effective spam filtering technology is a significant contribution to the sustainability of the cyberspace and to our society .

To add more, with increasing security measures in network services, remote exploitation is getting harder. As a result, attackers concentrate on more reliable attack vectors like email: victims are infected using either malicious attachments or links leading to malicious websites. Therefore efficient filtering and blocking methods for spam messages are needed.

1.2 OUTLINE OF THE PROJECT:

This project is based on building a spam detector using 2 machine learning algorithms. It is evaluated with the Training data set using diff performance metrics. The data set that we are going to use is from a shuffled sample of email sub and bodies containing both spam and ham emails in different proportions

1.3 PROMBLEM STATEMENT:

- **Problems Related to Costs-** Spam imposes costs on all Internet users. These costs have been increasing with the growth of the number of spam messages infiltrating the Internet daily. It is difficult to calculate the total costs of spam at the global level, though estimates suggest the costs are high.
- **Problems Related to Privacy:** The main privacy problem is that it causes significant unwanted intrusions. In addition, the collection of e-mail addresses is frequently made without the users' knowledge, much less with a specification of the purpose and consent. These problems are exacerbated when spam is sent indiscriminately .
- **Problems Related to Spam Content :** The content of spam messages may create a problem due to fraud and deception. Fraudulent or deceptive spam can exist in a number of forms. Spammers disguise the origin of their messages because they know their messages are being blocked or filtered and they aim to entice individuals to open their messages. A common trick that spammers use is to forge the headers of messages

1.4 OBJECTIVES:

Spam filtering is aimed at reducing to the barest minimum the volume of unsolicited emails. The main objective of identification of spam emails are

- ✚ To study different techniques for each step of the Naïve Bayes classifier.
- ✚ To focus on improving classification accuracy of the model instead on maximizing the values.
- ✚ To use current knowledge of the local classifiers for calculating the increment done for each token, instead of increasing the value by one.
- ✚ To test and validate the model by using different spam courses and comparing the result with the existing model.

CHAPTER 2

AIM AND SCOPE

2.1 AIM:

The main objective of this project is to build a spam detector using two machine learning algorithms.

2.2 SCOPE:

Adapting these methods provides sensitivity to the client and adapts well to the future spam techniques. It considers the complete message instead of single words with respect to the organization. It increases the security and control of the users sent mails and received mails. It reduces the IT administration costs. It also reduces Network Resources Cost.

2.3 LITERATURE REVIEW:

Spam filters have evolved quickly — and somewhat separately with different histories, objectives, evaluation methods, and measures of success. Practitioners have been concerned primarily with keeping their heads above water, delivering spam filters as quickly as possible to combat an ever-increasing tide of spam. Academics have, in large part, studied the problem as an application of the techniques and methods of information retrieval, machine learning and computer systems. Commercial product development and product testing involve yet another set of interests, methods, and measures of success.

These groups have had limited interaction; as a consequence, it is exceedingly difficult to deduce from the literature or other sources the relative performance and promise of current and proposed spam filter methods.

The literature, including the so-called gray literature (dissertations, technical reports, popular press articles, commercial reports, web publications, software documentation and cited unpublished works) was searched for articles describing a spam filter or spam filtering method and an evaluation of its

effectiveness were omitted from this review. Results derived using incorrect methodology, or results that are insufficiently noteworthy because their results are represented better elsewhere, were similarly omitted. Several hundred articles were considered for this review; perhaps one-third of them met our selection criteria. Certain aspects of spam filtering are well represented in the literature, while others are hardly represented or not represented at all. This review reflects this uneven coverage, reporting some aspects in detail while leaving others as largely uncharted territory.

2.4 PROPOSED SYSTEM:

In this project we are describing the method that is used to perform spam email classification. The first step is to select the data set file and apply the feature extraction technique for the extracted feature. For which we are using the word-count algorithm. The next step is to form the set of data that is extracted using the characteristic extraction technique. For the formation of the data we can calculate the probability of spam and not spam words in the document. The next step is to test the data with the help of Naïve Bayesian Classifier and Random Forest method is used because it is flexible, easy to use machine learning algorithm that produces, even without hyper-parameter tuning it gives a great result most of the time. It is also one of the most used algorithms, because of its simplicity and diversity. Both these methods are involved in spam and non-spam mail and it is used to make prediction whose value is greater. If spam words are larger than words that are not spam in an email, the mail is unwanted emails. In the next step we are calculating the words that are misclassified by the classifier and we calculate the accuracy of the classifier and also calculate the classifier error rate by calculating the fraction of the word that is misclassified and the total number of words in the document.

CHAPTER 3.

ALGORITHMS AND METHODS

3.1 OVERVIEW:

In this project we a spam detector Is built to reduce the un-wanted messages to be stored in the inbox. Spam –mails are not only annoying but also dangerous to consumers. Spam e-mail are messages randomly sent to multiple addresses by all sorts of groups, but mostly lazy advertisers and criminals who wish to lead you to phishing sites.

3.2 ARCHITECTURE DIAGRAM:

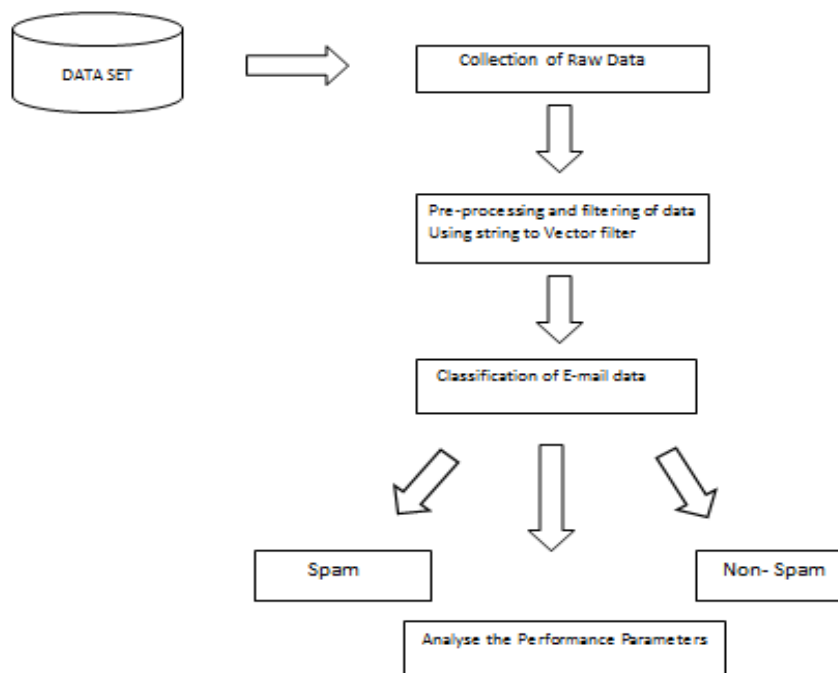


FIG 3.1 System Architecture

3.3 ALGORITHMS:

❖ Naive Bayes

Naive Bayes is the easiest classification algorithm (fast to build, regularly used for spam detection). It is a popular (baseline) method

For text categorization, the problem of judging documents as belonging to one category or the other (such as spam or legitimate, sports or politics, etc.) with word frequencies as the features.

The fundamental Naive Bayes assumption is that each feature makes an:

- Independent
- equal

contribution to the outcome.

Bayes 'Theorem

$$P(A|B) = \frac{P(B|A) P(A)}{P(B)}$$

Naive assumption

If any two events A and B are independent, then,

$$P(A,B) = P(A)P(B)$$

So,

$$y = \operatorname{argmax}_y P(y) \prod_{i=1}^n P(x_i|y)$$

❖ **RANDOM FOREST:**

- Random forest is like bootstrapping algorithm with Decision tree (CART) model.
- Random forest tries to build multiple CART model with different sample and different initial variables. For instance, it will take a random sample of 100 observation and 5 randomly chosen initial variables to build a CART model.
- It will repeat the process (say) 10 times and then make a final prediction on each observation.
- Final prediction is a function of each prediction. This final prediction can simply be the mean of each prediction.
- Random forest gives much more accurate predictions when compared to simple CART/CHAID or regression models in many scenarios. These cases generally have high number of predictive variables and huge sample size.
- This is because it captures the variance of several input variables at the same time and enables high number of observations to participate in the prediction.
- The fundamental concept behind random forest is a simple but powerful one the wisdom of crowds. In data science speak, the reason that the random forest model works so well is because a large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models.

3.3 PROJECT DESCRIPTION:

- ❖ Download spam and ham emails through Google's takeout service as an mbox file.
- ❖ Read the mbox files into lists using the 'mailbox' package. Each element in the list contained an individual email.
- ❖ Unpacked each email and concatenated their subject and body. We decided to include the email subject as well in our analysis because it is also a great indicator whether an email is a spam or ham.
- ❖ Converted the lists to dataframes, joined the spam and ham dataframes and shuffled the resultant dataframe.
- ❖ Split the dataframe into train and test dataframes. The test data was 33% of the original dataset.
- ❖ Split the mail text into lemmas and applied TF-IDF transformation using Count Vectorizer followed by TF-IDF transformer.
- ❖ Trained four models using the training data:
 - © Naive Bayes
 - © Random Forest
- ❖ Using the trained models, predicted the email label for test dataset. Calculated five metrics to gauge performance of the models:
 - © Accuracy
 - © Precision
 - © Recall
 - © F-score
 - © AUC

3.4 METRICS USED:

1. Precision

Precision refers to the closeness of two or more measurements to each other. In Machine Learning, precision is the fraction of relevant instances among the retrieved instances. $\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$ (Where TP = True Positive, TN = True Negative, FP = False Positive, FN = False Negative).

2. Accuracy

Accuracy refers to the closeness of a measured value to a standard or known value. $\text{Accuracy} = (\text{TP} + \text{TN}) / \text{ALL}$

3. Recall

Recall is how many of the true positives were recalled (found), i.e. how many of the correct hits were also found. $\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$

4. F-Score

F-scores are a statistical method for determining accuracy accounting for both precision and recall. It is essentially the harmonic mean of precision and recall.

5. AUC

AUC is the area under the ROC curve. The closer the AUC value is to 1, the better the model.

CHAPTER 4

SYSTEM IMPLEMENTATION

4.1 SYSTEM REQUIREMENTS:

HARDWARE REQUIREMENTS:

- System : Pentium IV 2.4 GHz
- Hard Disk : 40 GB
- Floppy Disk : 1.44 Mb
- Monitor : 15 VGA Colour
- Mouse : Dell
- RAM : 512 Mb

SOFTWARE REQUIREMENTS:

- Operating System : Windows XP/7
- Coding Language : Python
- IDE : Google Co-Lab and Goolge Drive.

4.2 ROLE OF PYTHON:

Python is a general-purpose coding language—which means that, unlike HTML, CSS, and JavaScript, it can be used for other types of programming and software development besides web development. That includes back end development, software development, data science and writing system scripts among other things. Python offers concise and readable code. While complex algorithms and versatile workflows stand behind machine learning and AI, Python's simplicity allows developers to write reliable systems. Python code is understandable by humans, which makes it easier to build models for machine learnin

4.3 COMPARATIVE GRAPHS:

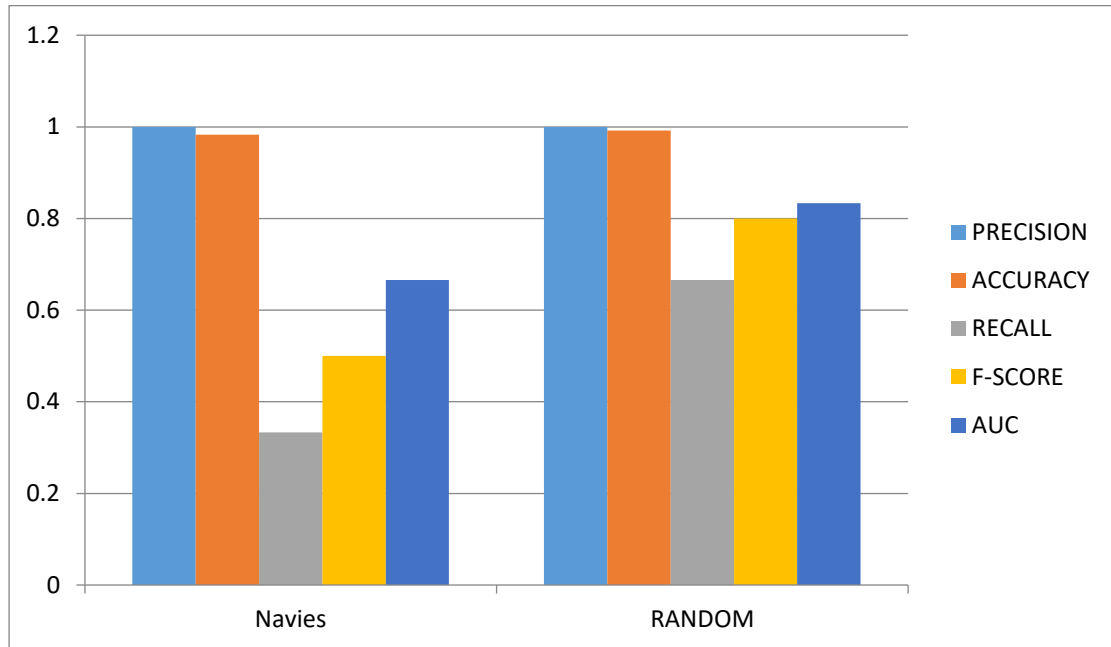


FIG 4.1 Comparison graph

Comparing the graphs Random Forest Method is more efficient and gives more accurate values than Naïve Bayes Classification. .

4.4 RESULT AND DISSCUSION:

	message	label	length
0	Achieve more with great connections. None	spam	41
1	Equip yourself for life in the new normal None	spam	46
2	Listen to ghazals by Ghulam Ali None	spam	36
3	Hi Aakash ,Your Knowastro.com Daily Overview H...	spam	75
4	=?utf-8?B?8J+TjfCfk40gSGkgLSBJbnRyb2R1Y2luZyBD...	spam	4959
	message	label	length
0	I have just fixed a bug in errorHandling.c tha...	ham	237
1	Errata for documentation b"\nThere is an errat...	ham	203
2	Re: problems with Self b'\nRe T.W. Leung\'s di...	ham	525
3	Errata for documentation b"\n(Apologies to tho...	ham	265
4	startup.self b'\nA simpler default startup.sel...	ham	220

Fig – 4.2 (Trained Data Set)

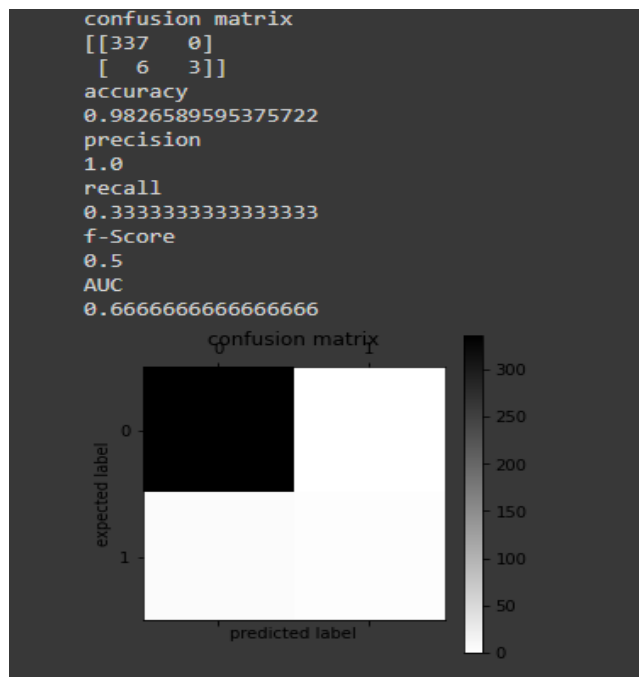


FIG –4.3 (NAÏVE BAYES)

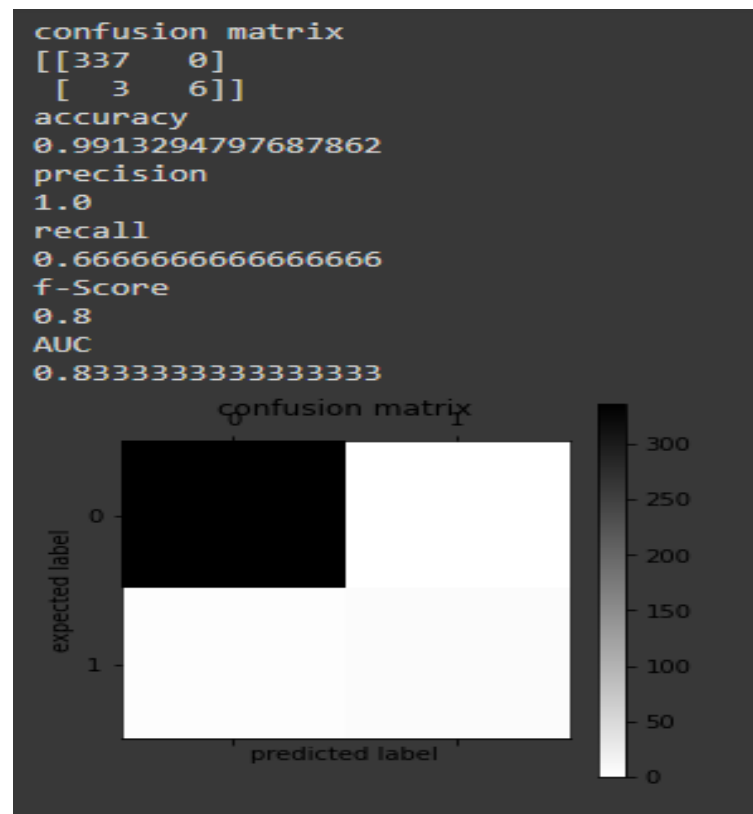


FIG -4.4 (RANDOM FOREST)

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 CONCLUSION:

Naive Bayes and Random Forest both work pretty well. While Naive Bayes algorithm has a high accuracy and a good precision. As far as the F-score is concerned, Random Forest has a good score as a result of good precision and recall both. Overall, we think that both Naive Bayes and Random Forest will be very good for spam detection. Of course, we can employ boosting or stacking ensemble methods to combine two or more of these models into a really good spam detector.

The main difference between Naive Bayes(NB) and Random Forest (RF) are their model size. Naive Bayes model size is low and quite constant with respect to the data. ... On the other hand, Random Forest model size is very large and if not carefully built, it results to over fitting. Thus our model of combining both thereby using their advantage to the best use provides a great spam filtering model

5.2 FUTURE WORK:

This work proposes a model for improving detection of malicious spam in email. Our model will employ a novel dataset for the process of feature selection, and then validate the set of selected features using three classifiers known in spam detection. Feature selection is expected to improve training time and accuracy for the classifiers

REFERENCES:

- [1] Rushdi, S. and Robet, M, "Classification spam emails using text and readability features", *IEEE 13th International Conference on Data Mining*, 2013.
- [2] Androutsopoulos, I., Paliouras, G., and Michelakis, "E. Learning to filter unsolicited commercial e-mail", *Technical report NCSR Demokritos*, 2011.
- [3] Rathi, M. and Pareek, V. "Spam Mail Detection through Data Mining A Comparative Performance Analysis", *I.J. Modern Education and Computer Science*, 2013, 12, 31-39.
- [4] Patil, T. and Sherekar, S. "Performance Analysis of Naïve Bayes and Classification Algorithm for Data Classification", *International Journal Of Computer Science And Applications*, 2013.
- [5] Kumar, S., Gao, X., Welch, I. and Mansoori, M., "A Machine Learning Based Web Spam Filtering Approach", *IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)*, Crans-Montana, 2016, pp. 973-980.

APPENDIX:

```
%matplotlib inline

#import metrics libraries

from sklearn.metrics import recall_score

from sklearn.metrics import confusion_matrix

from sklearn.metrics import f1_score

from sklearn.metrics import roc_auc_score

from sklearn.metrics import precision_score


import matplotlib.pyplot as plt

import sklearn
```

```

from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer

from sklearn.naive_bayes import MultinomialNB

from sklearn.svm import SVC, LinearSVC

from sklearn.metrics import classification_report, f1_score, accuracy_score,
confusion_matrix

from sklearn.pipeline import Pipeline

from sklearn.model_selection import GridSearchCV, learning_curve,
StratifiedKFold, cross_val_score, train_test_split

from sklearn.tree import DecisionTreeClassifier


from textblob import TextBlob

import pandas

import numpy as np

import csv

import mailbox

import nltk

nltk.download('punkt')

nltk.download('wordnet')

!pip install -U -q PyDrive # Pydrive helps in mounting dataset to colab


from pydrive.auth import GoogleAuth

from pydrive.drive import GoogleDrive

from google.colab import auth

from oauth2client.client import GoogleCredentials #oauth2 support given

```

```

auth.authenticate_user()

gauth = GoogleAuth()


gauth.credentials = GoogleCredentials.get_application_default() # prompt to
sign in

drive = GoogleDrive(gauth)

#function to get email text from email body

def getmailtext(message): #getting plain text 'email body'

    body = None

    #check if mbox email message has multiple parts

    if message.is_multipart():

        for part in message.walk():

            if part.is_multipart():

                for subpart in part.walk():

                    if subpart.get_content_type() == 'text/plain':

                        body = subpart.get_payload(decode=True)

            elif part.get_content_type() == 'text/plain':

                body = part.get_payload(decode=True)

    #if message only has a single part

    elif message.get_content_type() == 'text/plain':

        body = message.get_payload(decode=True)

    #return mail text which concatenates both mail subject and body

    mailtext=str(message['subject'])+" "+str(body)

    return mailtext


file_list = drive.ListFile({'q': "'1XXGiY6NH7DQyMTecW3JOB97sLCKnZA-a' in
parents and trashed=false"}).GetList() # folder_id posted here

```

```

for file1 in file_list:

    print('%s -> %s' % (file1['title'], file1['id'])) # List of files in the folder

    f_id=""

    drive_file = drive.CreateFile({'id': f_id})

    drive_file = drive.CreateFile({'id': '1NRCWLjIpsBrwOArPqfCSq_xnN5rYd5NN'})
    #file_id to mount

    drive_file.GetContentFile('Spam.mbox') #getfile

    #read spam mbox email file

    mbox = mailbox.mbox('Spam.mbox')


    mlist_spam = []

    #create list which contains mail text for each spam email message

    for message in mbox:

        mlist_spam.append(getmailtext(message))

        #break

    f_id=""

    drive_file = drive.CreateFile({'id': f_id})

    drive_file = drive.CreateFile({'id': '1wOOB70_pqjZm_sfqWXmUr422V3as-
    mdr'}) #file_id to mount

    drive_file.GetContentFile('ham.mbox') #getfile

    #read ham mbox email file

    mbox_ham = mailbox.mbox('ham.mbox')


    mlist_ham = []

    count=0

    #create list which contains mail text for each ham email message

```

```

for message in mbox_ham:

    mlist_ham.append(getmailtext(message))

    if count>1000:

        break

    count+=1

#create 2 dataframes for ham spam mails which contain the following info-
#Mail text, mail length, mail is ham/spam label

import pandas as pd

spam_df = pd.DataFrame(mlist_spam, columns=["message"])

spam_df["label"] = "spam"

spam_df['length'] = spam_df['message'].map(lambda text: len(text))

print(spam_df.head())

ham_df = pd.DataFrame(mlist_ham, columns=["message"])

ham_df["label"] = "ham"

ham_df['length'] = ham_df['message'].map(lambda text: len(text))

print(ham_df.head())

#merge and shuffle dataframes for ham/spam mails

mail_df = pd.concat([spam_df,ham_df])

from sklearn.utils import shuffle

mail_df = shuffle(mail_df)

len(mail_df)

```

```
mail_train, mail_test, y_train, y_test =
train_test_split(mail_df['message'],mail_df['label'],test_size=0.33,
random_state=42)
```

#for each word in the email text, get the base form of the word and return the list of base words

```
def split_into_lemmas(message):
```

```
    message = message.lower()
```

```
    words = TextBlob(message).words
```

```
    # for each word, take its "base form" = lemma
```

```
    return [word.lemma for word in words]
```

#function to apply the count vectorizer(BOW) and TF-IDF transforms to a set of input features

```
def features_transform(mail):
```

```
    #get the bag of words for the mail text
```

```
    bow_transformer =
CountVectorizer(analyzer=split_into_lemmas).fit(mail_train)
```

```
    #print(len(bow_transformer.vocabulary_))
```

```
    messages_bow = bow_transformer.transform(mail)
```

```
    #print sparsity value
```

```
    print('sparse matrix shape:', messages_bow.shape)
```

```
    print('number of non-zeros:', messages_bow.nnz)
```

```
    print('sparsity:  %.2f%%' % (100.0 * messages_bow.nnz /
(messages_bow.shape[0] * messages_bow.shape[1])))
```

```
    #apply the TF-IDF transform to the output of BOW
```

```
    tfidf_transformer = TfidfTransformer().fit(messages_bow)
```

```
    messages_tfidf = tfidf_transformer.transform(messages_bow)
```

```
    #print(messages_tfidf.shape)
```

```
    #return result of transforms
```

```

return messages_tfidf

#transform training set features into a set of useful features to build models

train_features=features_transform(mail_train)

#function which takes in y test value and y predicted value and prints the
associated model performance metrics

def model_assessment(y_test,predicted_class):

    print('confusion matrix')

    print(confusion_matrix(y_test,predicted_class))

    print('accuracy')

    print(accuracy_score(y_test,predicted_class))

    print('precision')

    print(precision_score(y_test,predicted_class,pos_label='spam'))

    print('recall')

    print(recall_score(y_test,predicted_class,pos_label='spam'))

    print('f-Score')

    print(f1_score(y_test,predicted_class,pos_label='spam'))

    print('AUC')

print(roc_auc_score(np.where(y_test=='spam',1,0),np.where(predicted_class=
=='spam',1,0)))

plt.matshow(confusion_matrix(y_test,                                predicted_class),
cmap=plt.cm.binary, interpolation='nearest')

plt.title('confusion matrix')

plt.colorbar()

plt.ylabel('expected label')

plt.xlabel('predicted label')

```

CODES FOR PERFORMANCE METRICS:

NAIVES BAYES:

```
#create and fit NB model

modelNB=MultinomialNB()

modelNB.fit(train_features,y_train)

#transform test features to test the model performance

test_features=features_transform(mail_test)

#NB predictions

predicted_class_NB=modelNB.predict(test_features)

#assess NB

model_assessment(y_test,predicted_class_NB)
```

RANDOM FOREST METHOD:

```
from sklearn.ensemble import RandomForestClassifier

#create and fit model

model_rf=RandomForestClassifier(n_estimators=20,criterion='entropy')

model_rf.fit(train_features,y_train)

#run model on test and print metrics

predicted_class_rf=model_rf.predict(test_features)

model_assessment(y_test,predicted_class_rf).
```