DEPARTMENT OF COMPUTER SCIENCE
GOVERNMENT ARTS AND SCIENCE COLLEGE
KANGEYAM - 638108

TEAM MEMBERS NAME:

RAGUPATHI.N(2022K0325)

KARTHIK.B(2022K0315)

YUVARAJ.S(2022K0339)

SURENDHIRABABU.D(2022K0335)

OPTIMIZING SPAM FILTERING WITH MACHINE LEARNING

1.INTRODUCTION

1.1 OVERVIEW :

Machine learning models have been utilized for multiple purposes in the field of computer science from resolving a network traffic issue to detecting a malware. Emails are used regularly by many people for communication and for socialising. Security breaches that compromises customer data allows 'spammers' to spoof a compromised email address to send illegitimate (spam) emails. This is also exploited to gain unauthorized access to their device by tricking the user into clicking the spam link within the spam email, that constitutes a phishing attack. Many tools and techniques are offered by companies in order to detect spam emails in a network. Organisations have set up filtering mechanisms to detect unsolicited emails by setting up rules and configuring the firewall settings. Google is one of the top companies that offers 99.9% success in detecting such emails. There are different areas for deploying the spam filters such as on the gateway (router), on the cloud hosted applications or on the user's computer. In order to overcome the detection problem of spam emails, methods such as content-based filtering,[1] rule-based filtering or Bayesian filtering have been applied. Unlike the 'knowledge engineering' where spam detection rules are set up and are in constant need of manual updating thus consuming time and resources, Machine learning makes it easier because it learns to recognise the unsolicited emails (spam) and legitimate emails (ham) automatically and then applies those learned instructions to unknown incoming emails[2]. The proposed spam detection to resolve the issue of the spam classification problem can be further experimented by feature selection or automated parameter selection for the models. This research conducts experiments involving five different machine learning models with Particle Swarm Optimization (PSO) and Genetic Algorithm (GA). This will be compared with the base models to conclude whether the proposed models have improved the performance with parameter tuning.

1.2. PURPOSE :

2. PROBLEM DEFINITION & DESIGN THINKING

A spam filter is a program used to detect unsolicited, unwanted and virus-infected emails and prevent those messages from getting to a user's inbox. Like other types of filtering programs, a spam filter looks for specific criteria on which to base its judgments.

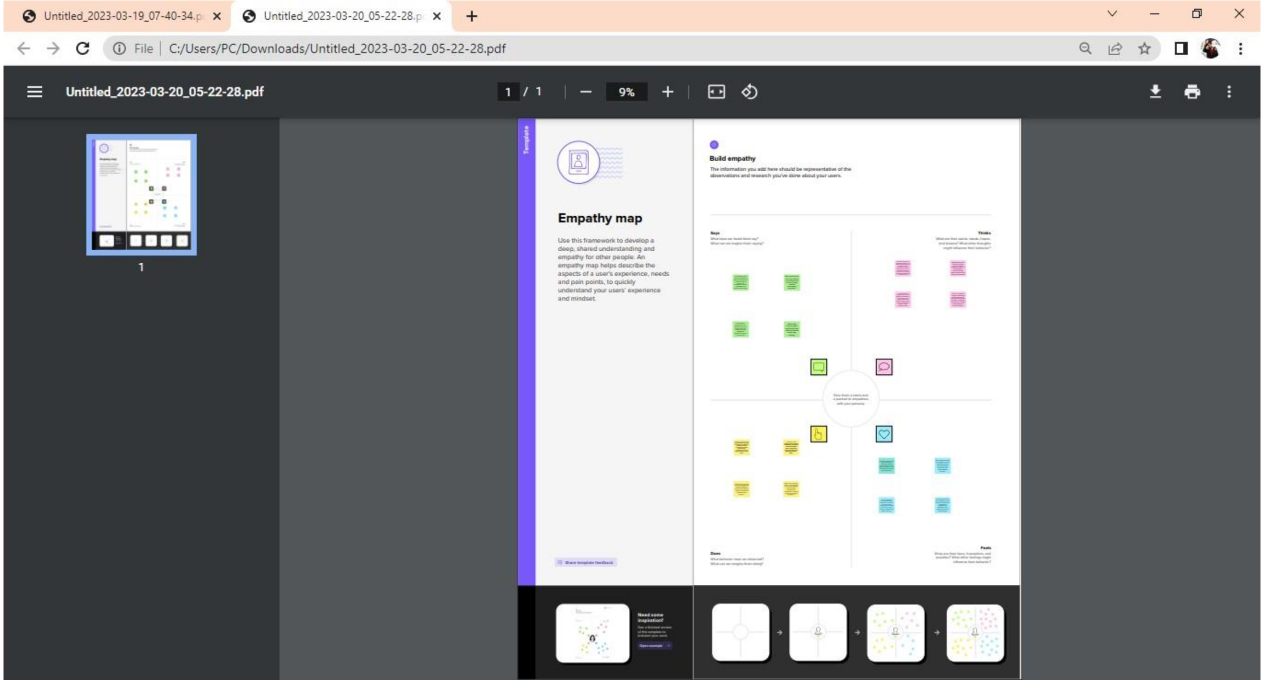
Internet service providers (ISPs), free online email services and businesses use email spam filtering tools to minimize the risk of distributing spam. For example, one of the simplest and earliest versions of spam filtering, like the one that was used by Microsoft's Hotmail, was set to watch out for particular words in the subject lines of messages. An email was excluded from the user's inbox whenever the filter recognized one of the specified words.

This method is not especially effective and often omits perfectly legitimate messages, called *false positives*, while letting actual spam messages through.

More sophisticated programs, such as Bayesian filters and other heuristic filters, identify spam messages by recognizing suspicious word patterns or word frequency. They do this by learning the user's preferences based on the emails marked as spam. The spam software then creates rules and applies them to future emails that target the user's inbox.

For example, whenever users mark emails from a specific sender as spam, the Bayesian filter recognizes the pattern and automatically moves future emails from that sender to the spam folder.

2.1. EMPATHY MAP



Untitled_2023-03-19_07-40-34.p ×

Untitled_2023-03-20_05-22-28.p ×

File | C:/Users/PC/Downloads/Untitled_2023-03-20_05-22-28.pdf

Untitled_2023-03-20_05-22-28.pdf

1 / 1 | 9%

Template

Empathy map

Use this framework to develop a deep, shared understanding and empathy for other people. An empathy map helps describe the aspects of a user's experience, needs and pain points, to quickly understand your users' experience and mindset.

Build empathy

The information you add here should be representative of the observations and research you've done about your users.

Says

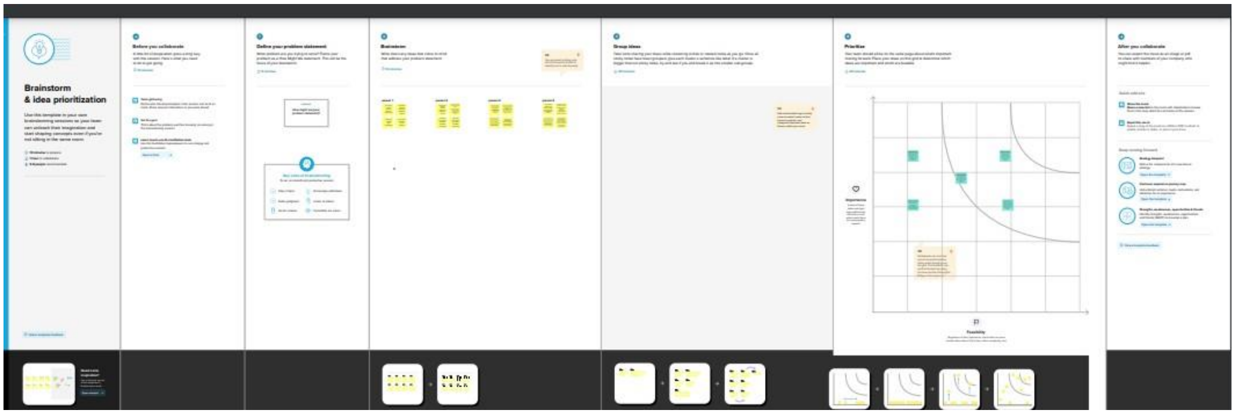
Thinks

Does

Feels

Need some inspiration?

2.2 IDEATION & BRAINSTORMINGS MAP



Brainstorm & idea prioritization

Before you collaborate

Define your problem statement

Brainstorm

Group ideas

Prioritize

After you collaborate

3. RESULT

4.ADVANTAGES& DISADVANTAGES

ADVANTAGES :

- MTA/MUA
- It is easy to install and effective in blocking a large percentage of spam
- Blacklist/ Distributed Black list
- Blocks mail from known spam sources
- Readily available pools of list
- It is effective and easy to implement.
- It reduces error rates as legitimate e-mail would not be blocked even if the ISP from which it originated, is on a real-time Block list
- The presence of a single token should not cause the e-mail to be classified as spam.
- The true strength of this technique only becomes apparent when we introduce a distribution mechanism for known spam hashes
- Blocks known spam Low rate of false positives
- The key advantage of this system is its resilience and adaptability, because personalised collaborative spam filters continually refine their contacts with whom they are connected
- In addition, possible eradication of large volumes of spam through collaborative reporting of spam
- Learns new spammer tactics automatically. Adapt to changing spam.High dimensional feature Space.

DISADVANTAGES :

The rule set needs constant updating and refinement because most spammers use obfuscation techniques.

It should be combined with other methods to filter out a larger volume of spam.May block harmless messages.Needs constant updating and maintanence. Exact rules are difficult to formulate and maintain-Spam is always changing spammer e-mail addresses

Any email sent by a stranger will simply be incorrectly classified as FP/FN List maintenance can be cumbersome

Need occasional refinement and in most cases the refinement is automated, meaning less hassle for end-users.

Spammers introduce some random variables/characters either in the body or on the subject line. So, this would create completely different outputs for each spam which has some random content.

Can be circumvented by randomization A disadvantage of this approach is that it uses SMTP to communicate between peers. SMTP is unauthenticated, and so malicious attackers could attempt to undermine the filter by spoofing reports from other peers.

Still vulnerable to random changes in spam e-mail, and there are problems with scalability of this method

Difficult for non-linear separable case Training time is high compare to NB

5. APPLICATION

A spam filtering solution cannot be 100 percent effective. However, a business email system without spam filtering is highly vulnerable, if not unusable. It is important to stop as much spam as you can, to protect your network from the many possible risks: viruses, phishing attacks, compromised web links and other malicious content

Spam filters also protect your servers from being overloaded with non-essential emails, and the worse problem of being infected with spam software that may turn them into spam servers themselves.

By preventing spam email from reaching your employees' mailboxes, spam filters give an additional layer of protection to your users, your network, and your business.

At the very least, spam email is a nuisance that will clog up your employees' inboxes and overload your servers. Spam is also dangerous — the entry point for serious attacks that could damage your computers, your computer network, your bottom line, and even your company's reputation. Yes, you need a spam filter solution as the key first line of defense.

6. CONCLUSION

The project successfully implemented models combined with bio-inspired algorithms. The spam email corpus used within the project was both numerical as well as alphabetical.

Approximately 50,000 emails were tested with the proposed models. Genetic Algorithm worked better overall for both text-based datasets and numerical-based datasets than PSO.

The PSO worked well for Multinomial Naive Bayes and Stochastic Gradient Descent, whereas GA worked well for Random Forest and Decision Tree. Naive Bayes algorithm was proved to have been the best algorithm for spam detection.

This was concluded by evaluating the results for both numerical and alphabetical based dataset.

The highest accuracy provided was 100% with GA optimization on randomized data distribution for 80:20 train and test split set on Spam Assassin dataset. In terms of F1-Score, precision and recall, Genetic Algorithm had more impact than PSO on MNB, SGD, RF and DT[8].

In the last decade, spam mail has become an increasing threat to mail communication. Thousands of undesirable email messages are generated in a bulk format by spammers. These, in turn, may often lead to a waste of time spent by users in searching and deleting the spam emails, loss of the user's network bandwidth and an unnecessary increase in the traffic volume, etc. Most spam email generally contains advertisements of products or services, which may be useless or unnecessary for the user. In this paper, a spam mail detection (SPMD) method is proposed to detect spam emails. Three algorithms are used for classification purpose like Naive Bayes, Decision Tree and Random Forest. Among these classifiers, Random Forest was shown to achieve the highest accuracy of 92.97%.

7. FUTURE SCOPE

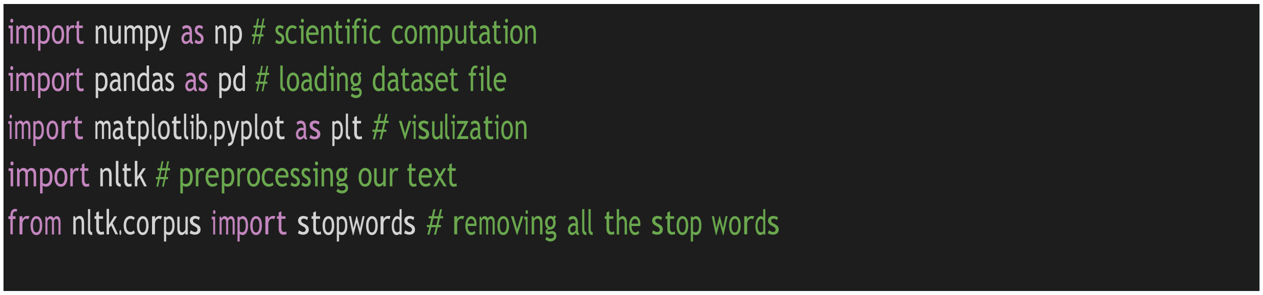
We plan to further carry out the machine learning algorithms to optimize and compare with different bio-inspired algorithms such as Firefly, Bee Colony and Ant Colony Optimization as researched in the previous sections. We could also explore the Deep learning Neural Network with PSO and GA by exploring different libraries such as Tensor Flow's DNN Classifier or similar.

We found that the Neural Network algorithm could have worked better with more dimension like providing broader range of values for learning rate, activation, solver, and alpha.

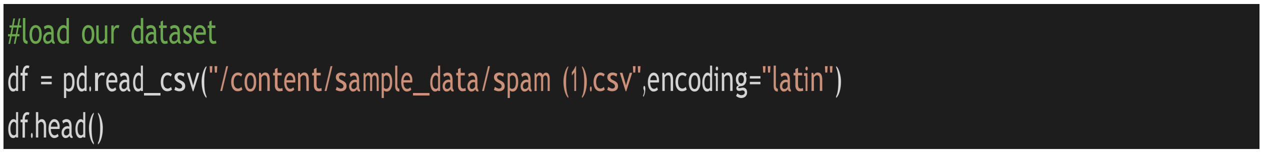
If this project is taken further, implementation for MLP could be done through Keras or Tensor Flow with GPU application. This will allow the user to input other parameters and a range of possibilities as their key values.

8. APPENDIX

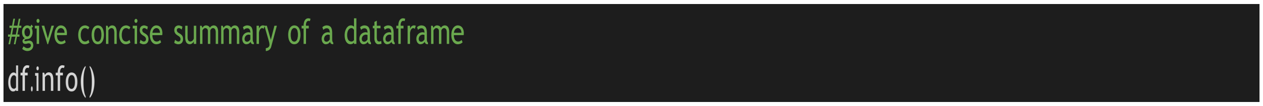
A.SOURCE CODE :



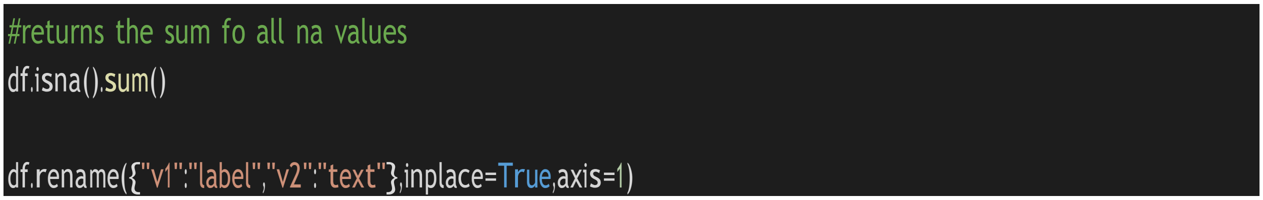
```
import numpy as np # scientific computation
import pandas as pd # loading dataset file
import matplotlib.pyplot as plt # visulization
import nltk # preprocessing our text
from nltk.corpus import stopwords # removing all the stop words
```



```
#load our dataset
df = pd.read_csv("/content/sample_data/spam (1).csv",encoding="latin")
df.head()
```



```
#give concise summary of a dataframe
df.info()
```



```
#returns the sum fo all na values
df.isna().sum()

df.rename({"v1":"label","v2":"text"},inplace=True,axis=1)
```

```
#bottom 5 rows of the dataframe
df.tail()
```

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['label'] = le.fit_transform(df['label'])
```

```
X = df
y = df
```

```
#splitting data into train and validation sets using train_test_split

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 0)
##train size 80% and test size 20%
```

```
### Given data is imbalanced one, w are balancing the data
```

```
print("Before OverSampling, counts of label '1': {}".format(sum(y_train == 1)))
print("Before OverSampling, counts of label '0': {} \n".format(sum(y_train == 0)))

from imblearn.over_sampling import SMOTE
sm = SMOTE(random_state = 2)
X_train_res, y_train_res = sm.fit_resample(X_train, y_train.ravel())

print('After OverSampling, the shape of train_X: {}'.format(X_train_res.shape))
print('After OverSampling, the shape of train_y: {} \n'.format(y_train_res.shape))
print("After OverSampling, counts of label '1': {}".format(sum(y_train_res == 1)))
print("After OverSampling, counts of label '0': {}".format(sum(y_train_res == 0)))
```

```
nltk.download("stopwords")
```

```
import nltk
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
```

```
import re
Corpus = []
length = len(df)

for i in range(0,length):
    text = re.sub("[^a-zA-Z0-9]"," ",df["text"][i])
    text = text.lower()
    text = text.split()
    pe = PorterStemmer()
    stopword = stopwords.words("english")
    text = [pe.stem(word) for word in text if not word in set(stopword)]
    text = " ".join(text)
    Corpus.append(text)
```

```
Corpus
```

```
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(max_features=35000)
x = cv.fit_transform(corpus).toarray()
```

```
impot pickle ## importing pickle used for dumping models
pickle.dump(cv,open('cv1.pk1', 'wb')) ## saving to into cv.pkl file
```

```
df.describe()
```

```
df.shape
```

```
df["label"].value_counts().plot(kind="bar",figsize=(12,6))
plt.xticks(np.arange(2), ('Non spam', 'spam'),rotation=0);
```

```
#Splitting data into train and validation sets using train_test_split
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.20, random_state = 0)
##train size 80% and test size 20%
```

```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
model.fit(x_train_res, y_train_res)
```

```
from sklearn.ensemble import RandomForestClassifier
model1 =RandomForestClassifier()
model1.fit(X_train_res, y_train_res)
```

```
from sklearn.native_bayes import multinomia1NB
model = Multinomia1NB()
```

```
#fitting the model to the training sets
model.fit(X_train_res, y_train_res)

from tensorflow.keras.models import sequential
from tensorflow.keras.layers import dense
```

```
#fitting the model to the training sets
model = Sequential()
```

```
x_train_shape
```

```
model.add(dense(units  = x_train_res.shape[1],activation="relu",kernel_initializer="random_uniform"))
```

```
model.add(dense(units=100,activation="relu",kernel_initializer="random_uniform"))
model.add(dense(units=100,activation="relu",kernel_initializer="random_uniform"))
```

```
model.add(dense(units=100,activation="sigmoid"))
```

```
model.compile(optimizer="adam",loss="binery_crossentropy",metrics=['accuracy'])
```

```
generator = model.fit(X_train_res,y_train_res,epoch=len(X_train_res)//64)
```

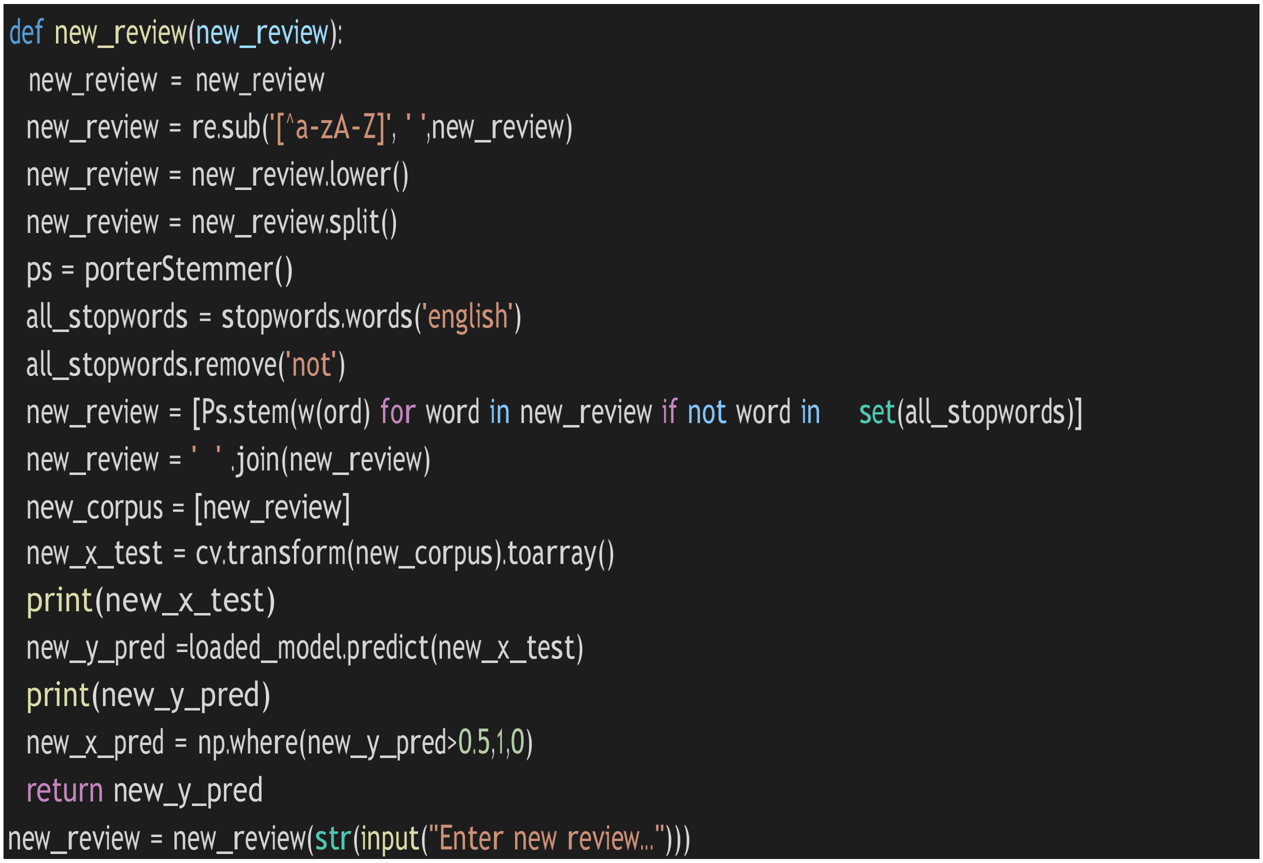
```
generator = model.fit(X_train_res,y_train_res,epoch=10,steps_per_epoch=len(X_train_res)//64)
```

```
y_pred=model.predict(x_test)
y_pred
```

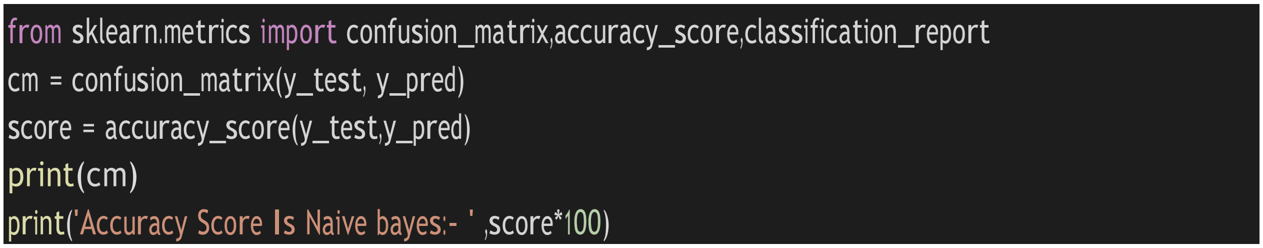
```
y_pr = np.Where(y_pred>0.5,1,0)
```

```
y_test
```

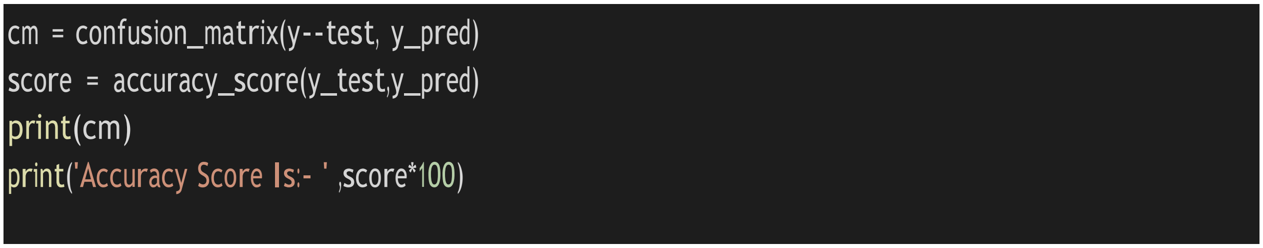
```
from  sklearn.metrics import confusion_matrix,accuracy_score
cm = confusion_matrix(y_test, y_pr)
score = accuracy_score(y_test,y_pr)
print(cm)
print('Accuracy score Is:- '  ,score*100)
```



```
def new_review(new_review):
  new_review  =  new_review
  new_review = re.sub('[^a-zA-Z]', ' ',new_review)
  new_review = new_review.lower()
  new_review = new_review.split()
  ps = porterStemmer()
  all_stopwords = stopwords.words('english')
  all_stopwords.remove('not')
  new_review = [Ps.stem(w(ord) for word in new_review if not word in     set(all_stopwords)]
  new_review = '  '.join(new_review)
  new_corpus = [new_review]
  new_x_test = cv.transform(new_corpus).toarray()
  print(new_x_test)
  new_y_pred =loaded_model.predict(new_x_test)
  print(new_y_pred)
  new_x_pred = np.where(new_y_pred>0.5,1,0)
  return new_y_pred
new_review = new_review(str(input("Enter new review...")))
```



```
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
cm = confusion_matrix(y_test, y_pred)
score = accuracy_score(y_test,y_pred)
print(cm)
print('Accuracy Score Is Naive bayes:- ' ,score*100)
```



```
cm = confusion_matrix(y--test, y_pred)
score = accuracy_score(y_test,y_pred)
print(cm)
print('Accuracy Score Is:- ' ,score*100)
```

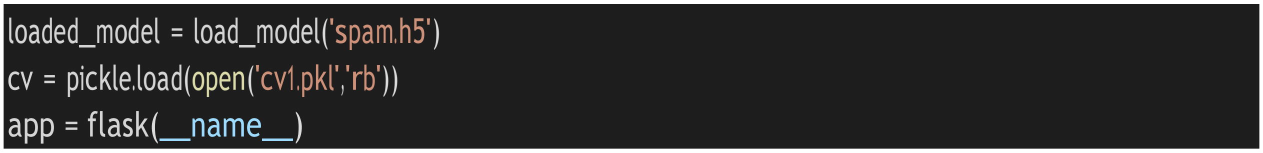
```
cm1  =  confusion_matrix(y_test,  y_pred1)
score1  =  accuracy_score(y_test,y_pred1)
print(cm1)
print('Accuracy Score Is:-  ' ,score1*100)
```

```
from sklearn.metrics import confusion_matrix,accuracy_score
cm = confusion_matrix(y_test,y_pr)
score = accuracy_score(y_test,y_pr)
print(cm)
print('Accuracy Score Is:-   ' ,score*100)
```

```
from sklearn.metrics import confusion_matrix,accuracy_score
cm = confusion_matrix(y_test, y_pr)
score = accuracy_score(y_test,y_pr)
print(cm)
print('Accuracy score Is:-   '  ,score*100)
```

```
model.save('spam.h5')
```

```
#importing essential  libraries
from flask import flask, render_template, request
import pickle
import numby as np
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem import porterStemmer
from tensorflow.keras.models import loaded_model
```



```
loaded_model = load_model('spam.h5')
cv = pickle.load(open('cv1.pkl','rb'))
app = flask(__name__)
```