

PROJECT REPORT

1. INTRODUCTION

1.1 Overview

Over recent years, as the popularity of mobile phone devices has increased, Short Message Service (SMS) has grown into a multi-billion dollar industry. At the same time, reduction in the cost of messaging services has resulted in growth in unsolicited commercial advertisements (spams) being sent to mobile phones. Due to Spam SMS, Mobile service providers suffer from some sort of financial problems as well as it reduces calling time for users. Unfortunately, if the user accesses such Spam SMS they may face the problem of virus or malware. When SMS arrives at mobile it will disturb mobile user privacy and concentration. It may lead to frustration for the user. So Spam SMS is one of the major issues in the wireless communication world and it grows day by day.

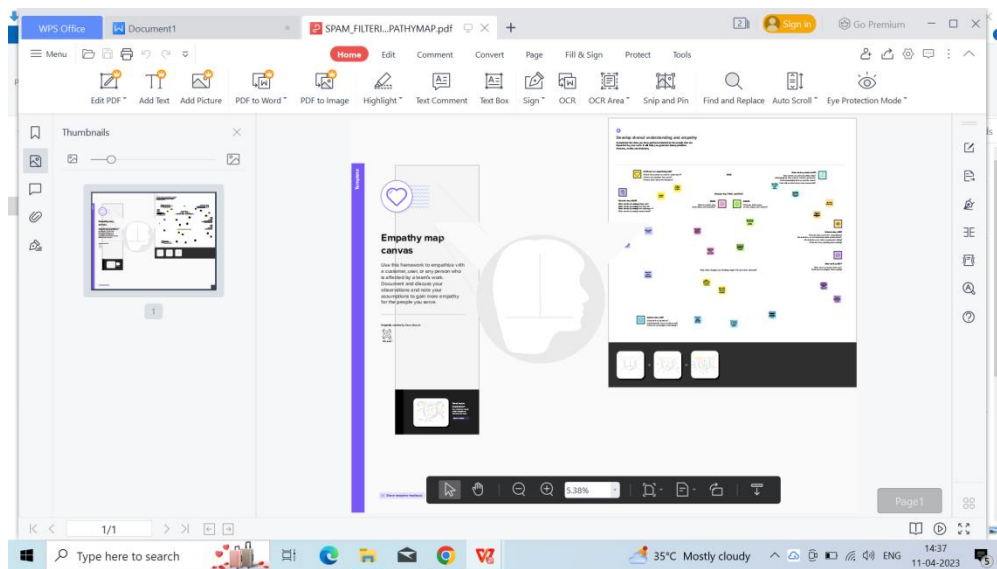
To avoid such Spam SMS people use white and black list of numbers. But this technique is not adequate to completely avoid Spam SMS. To tackle this problem it is needful to use a smarter technique which correctly identifies Spam SMS. Natural language processing technique is useful for Spam SMS identification. It analyses text content and finds patterns which are used to identify Spam and Non-Spam SMS.

1.2 Purpose

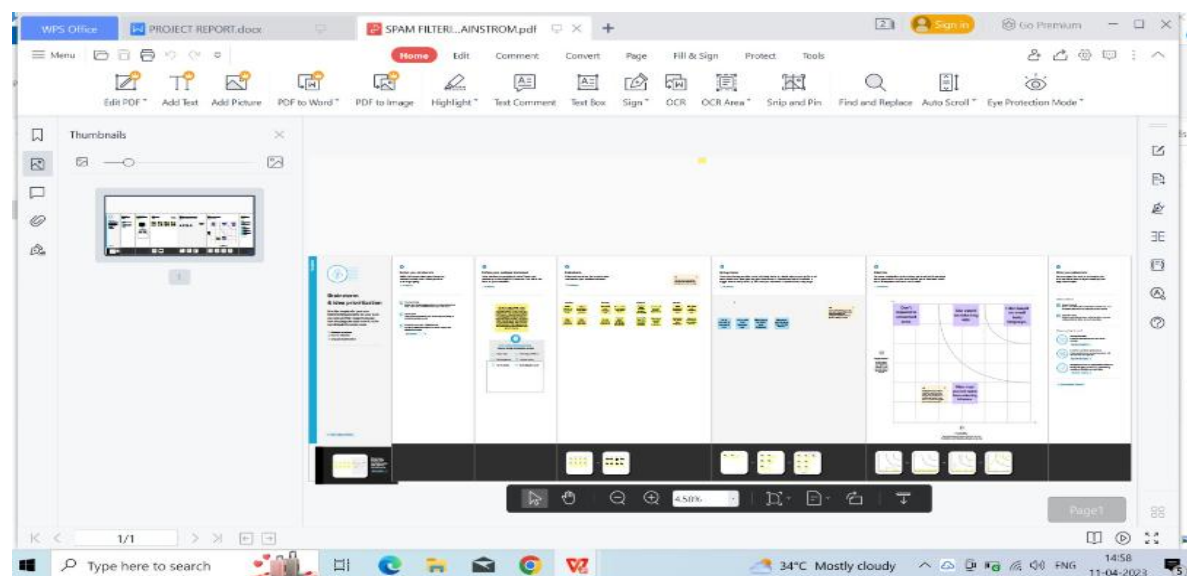
In today's society, practically everyone has a mobile phone, and they all get communications (SMS/ email) on their phone regularly. But the essential point is that majority of the messages received will be spam, with only a few being ham or necessary communications. Scammers create fraudulent text messages to deceive you into giving them your personal information, such as your password, account number, or Social Security number. If they have such information, they may be able to gain access to your email, bank, or other accounts. So we are going to develop various machine learning models using Tensor flow for SMS spam detection and also analyze the performance metrics of different models.

2. Problem Definition and Design Thinking

2.1 Empathy Map

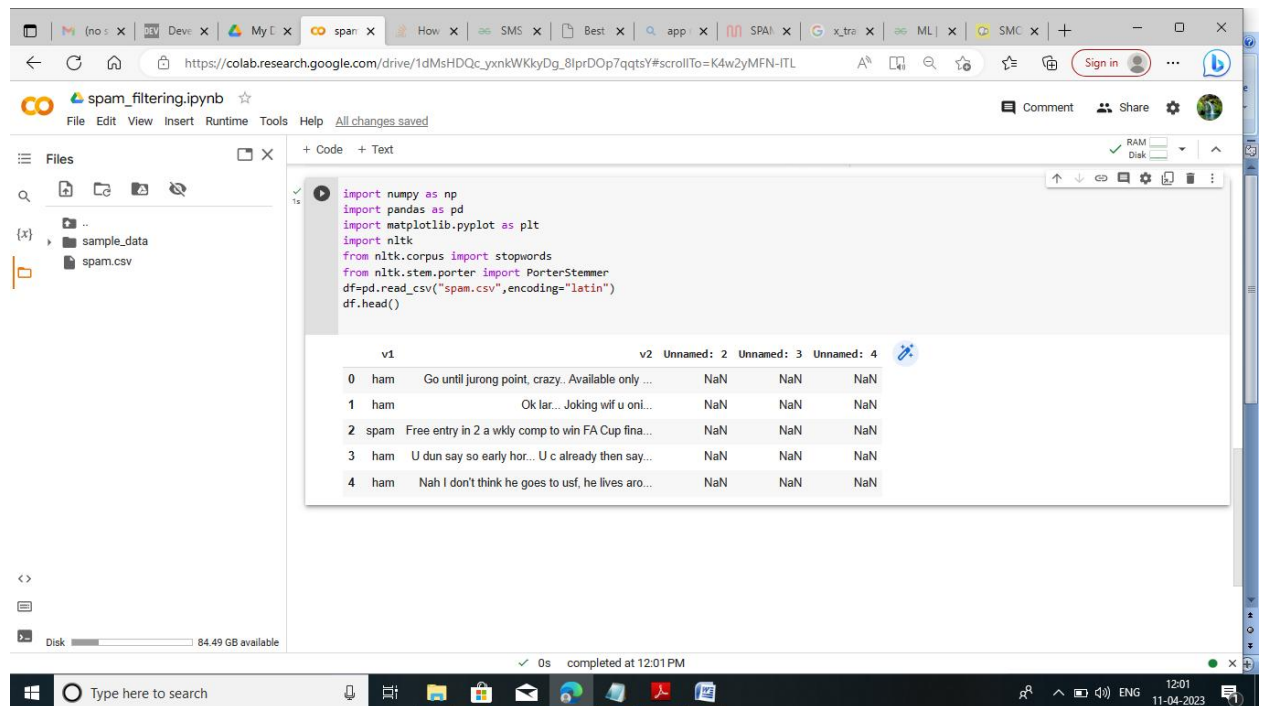


2.2 Ideation & Brainstorming Map



RESULT

Importing the libraries and Read the Dataset



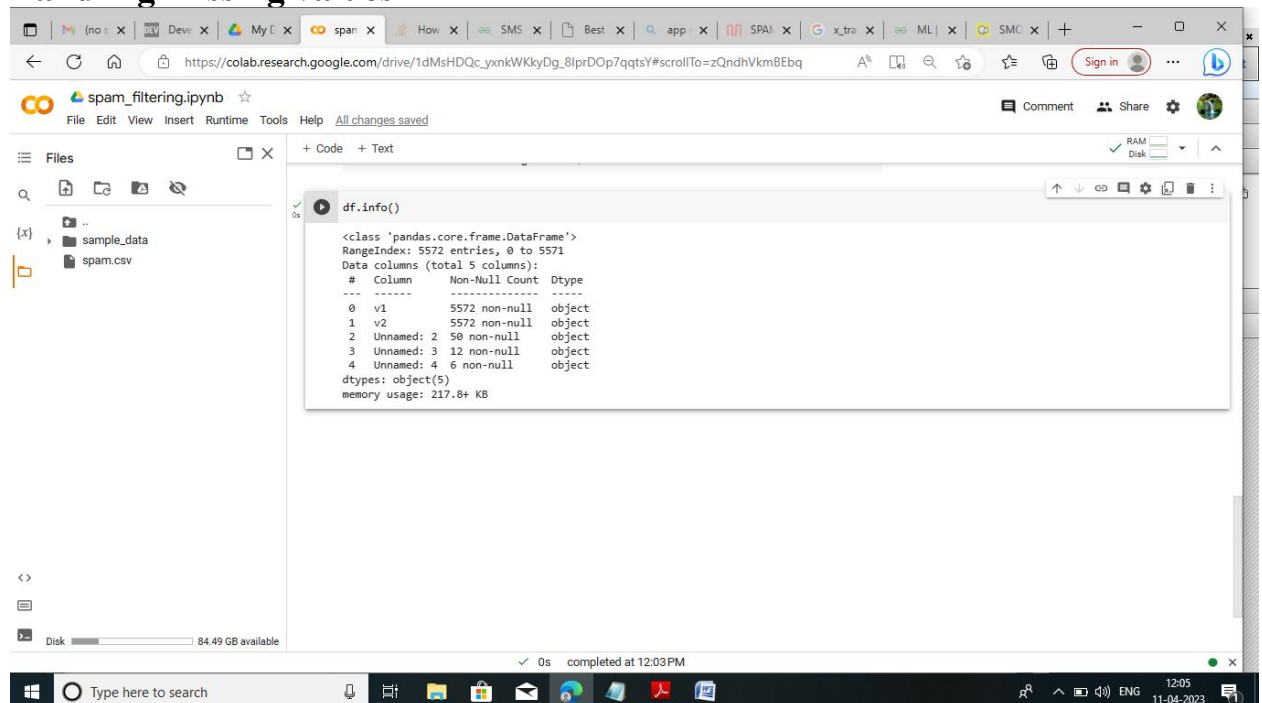
The screenshot shows a Jupyter Notebook interface with the following code in the first cell:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import nltk
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
df=pd.read_csv("spam.csv",encoding="latin")
df.head()
```

The output of the code is a preview of the first five rows of the dataset:

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN

Handling missing values



The screenshot shows a Jupyter Notebook interface with the following code in the first cell:

```
df.info()
```

The output of the code is the following information:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 5 columns):
#   Column      Non-Null Count  Dtype  
---  -
0    v1          5572 non-null   object  
1    v2          5572 non-null   object  
2    Unnamed: 2   50 non-null     object  
3    Unnamed: 3   12 non-null     object  
4    Unnamed: 4   6 non-null      object  
dtypes: object(5)
memory usage: 217.8+ KB
```

Colab interface showing the execution of `df.isna().sum()` on a dataset named `spam.csv`. The output shows the number of missing values for each column.

```
df.isna().sum()
v1      0
v2      0
Unnamed: 2    5522
Unnamed: 3    5560
Unnamed: 4    5566
dtype: int64
```

The interface also shows the file explorer on the left with `sample_data` and `spam.csv` files. The bottom status bar indicates the execution completed at 12:06 PM.

Colab interface showing the execution of `df.rename()` to rename columns `v1` to `label` and `v2` to `text`. The output shows the first and last rows of the dataset.

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

	label	text	Unnamed: 2	Unnamed: 3	Unnamed: 4
5567	spam	This is the 2nd time we have tried 2 contact u...	NaN	NaN	NaN
5568	ham	Will i_b going to esplanade fr home?	NaN	NaN	NaN
5569	ham	Pity, " was in mood for that. So...any other s...	NaN	NaN	NaN
5570	ham	The guy did some bitching but i acted like i'd...	NaN	NaN	NaN
5571	ham	Rofl. Its true to its name	NaN	NaN	NaN

The interface also shows the file explorer on the left with `sample_data` and `spam.csv` files. The bottom status bar indicates the execution completed at 12:21 PM.

Handling Categorical Values

ID	label	text
5567	spam	This is the 2nd time we have tried 2 contact u...
5568	ham	Will l_b going to esplanade fr home?
5569	ham	Pity, * was in mood for that. So...any other s...
5570	ham	The guy did some bitching but I acted like I'd...
5571	ham	Rofl. Its true to its name

```

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

x=df['text'].values
y=df['label'].values
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
print("Before oversampling,counts of label'1': {}".format(sum(x_train==1)))
print("Before oversampling,counts of label'0':{} \n".format(sum(x_train==0)))

Before oversampling,counts of label'1': 581
Before oversampling,counts of label'0':3876
  
```

```

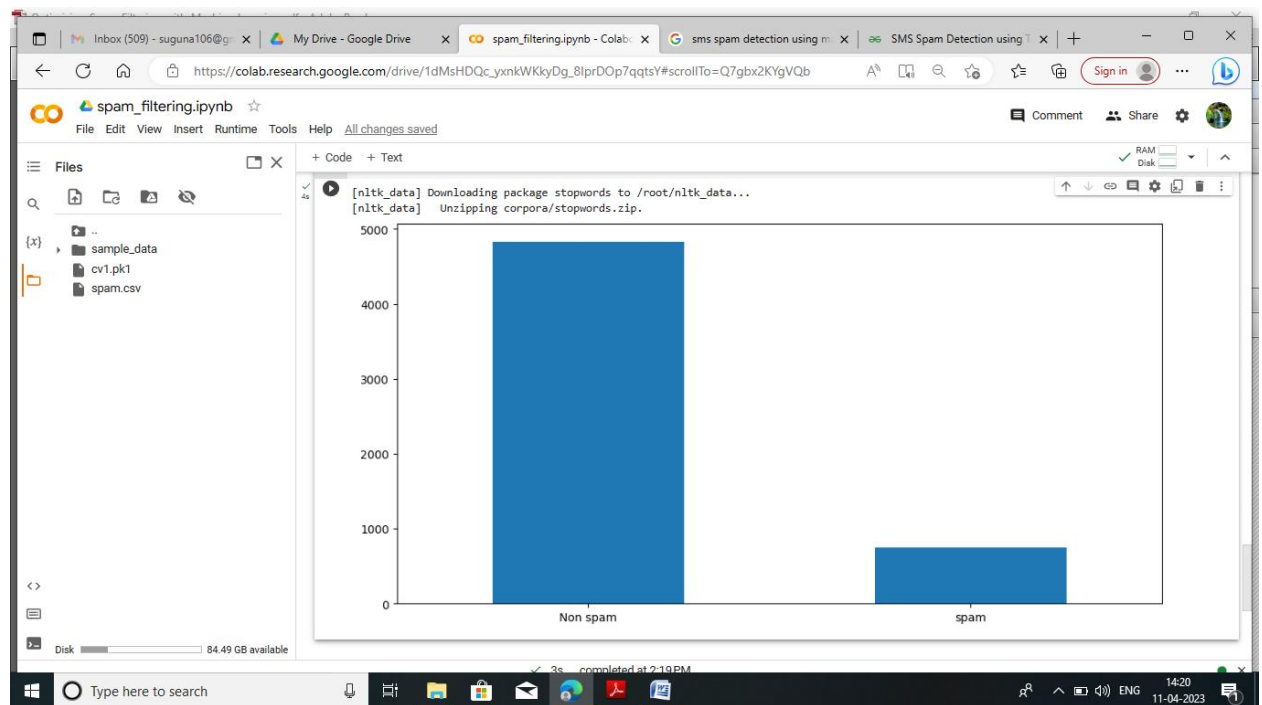
from imblearn.over_sampling import SMOTE
sm = SMOTE(random_state = 2)
x_train_res, y_train_res = sm.fit_resample(x_train,y_train.ravel())
print("After OverSampling, the shape of train_X: {}".format(y_train_res.shape))
print("After OverSampling, the shape of train_y: {} \n".format(y_train_res.shape))
print("After OverSampling, the counts of label '1': {}".format(sum(y_train_res == 1)))
print("After OverSampling, the counts of label '0': {}".format(sum(y_train_res == 0)))
  
```

Before OverSampling, counts of label '1': 581
 Before OverSampling, counts of label '0': 3876

After OverSampling, the shape of train_X: (7752, 7163)
 After OverSampling, the shape of train_y: (7752,)

After OverSampling, counts of label '1': 3876
 After OverSampling, counts of label '0': 3876

Visual analysis



Testing the model

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

35/35 [=====] - 2s 29ms/step
array([[ 1.5844109e-15],
       [ 4.4117199e-04],
       [ 1.1517070e-18],
       ...,
       [ 2.0661259e-08],
       [ 3.8018154e-17],
       [ 1.2099350e-12]], dtype=float32)

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

y_test
array([0, 0, 0, ..., 0, 0, 0])

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)

[[937  12]
 [ 16 150]]
Accuracy Score Is:- 97.48878923766816
```

Compare the model


```

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)

```

```

[[935  14]
 [ 13 153]]
Accuracy Score Is:- 97.57847533632287

```

```

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)

```

```

[[796 153]
 [ 17 149]]
Accuracy Score Is:- 84.75336322869956
[[855  94]
 [ 14 152]]
Accuracy Score Is:- 90.31390134529148

```

```

121/121 [=====] - 24s 196ms/step - loss: 0.0120 - accuracy: 0.9974
Epoch 9/10
121/121 [=====] - 23s 193ms/step - loss: 0.0103 - accuracy: 0.9977
Epoch 10/10
111/121 [=====>...] - ETA: 1s - loss: 0.0128 - accuracy: 0.9972WARNING:tensorflow:

```

```

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)

```

```

[[937  12]
 [ 16 150]]
Accuracy Score Is:- 97.48878923766816

```

```

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)

```

```

[[937  12]
 [ 16 150]]
Accuracy Score Is:- 97.48878923766816

```

ADVANTAGES AND DISADVANTAGES

ADVANTAGES

Detection of spam is important for securing message and e-mail communication

DISADVANTAGES

Spam emails can be the source of a great amount of malware like viruses, Trojans, worms, and others which are specifically designed to disrupt or damage computer systems.

The dangers of spam messages for the users are many: undesired advertisement, exposure of private information, becoming a victim of a fraud or financial scheme, being lured into malware and phishing websites, involuntary exposition to inappropriate content, etc

APPLICATION

MOBILE PHONE

CONCLUSION

From the above discussion and experimentation we are concluded that machine learning algorithm can play a vital role in identifying SPAM SMS.

FUTURE SCOPE

In the future we plan to deal with more challenging problem such as the analysis and management of report in spam filters storing. Solution for this problem is another focus of work in the future

APPENDIX

SOURCE CODE

```
from scipy.sparse import data
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import nltk
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
df=pd.read_csv("spam.csv",encoding="latin")
df.head()
df.info()
df.isna().sum()
df.rename({"v1":"label","v2":"text"},inplace=True,axis=1)
df.head()
df.tail()
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['label']=le.fit_transform(df['label'])
from sklearn.model_selection import train_test_split
x=df["label"].values
y=df["text"].values
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size
=0.20,random_state=0)
print("Before oversampling,counts of label'1': {}".format(su
m(x_train==1)))
print("Before oversampling,counts of label'0':{} \n".format(
sum(x_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.r
avel())
print('After OverSampling, the shape of train_x: {}'.format(
y_train_res.shape))
print('After OverSampling, the shape of train_y: {} \n'.form
at(y_train_res.shape))
print("After OverSampling, the counts of label '1': {}".form
at(sum(y_train_res == 1)))
print("After OverSampling, the counts of label '0': {}".form
at(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()
    stopwords = stopwords.words("english")
    text = [pe.stem(word) for word in text if not word in se
t(stopword)]
    text = " ".join(text)
    corpus.append(text)
from sklearn.feature_extraction.text import CountVectorizer
cv=CountVectorizer(max_features=35000)
x=cv.fit_transform(corpus).toarray()
import pickle
pickle.dump(cv,open('cv1.pkl','wb'))
df.describe()
df["label"].value_counts().plot(kind="bar",figsize=(12,6))
plt.xticks(np.arange(2), ('Non spam', 'spam'),rotation=0);
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_bal=sc.fit_transform(x_bal)
x_bal=pd.DataFrame(x_bal,columns=names)
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
model.fit(x_train_res, y_train_res)
from sklearn.ensemble import RandomForestClassifier
modell = RandomForestClassifier()
modell.fit(x_train_res, y_train_res)
from sklearn.naive_bayes import MultinomialNB
model = MultinomialNB()
#Fitting the model to the training sets
model.fit(x_train_res, y_train_res)
from tensorflow.keras.models import Sequential
from tensorflow.keras.models import Dense
#fitting the model to the training sets
model = Sequential()
x_train.shape
(4457,7163)

```

```

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=1,activation="sigmoid"))
model.compile(optimizer="adam",loss=binary_crossentropy,metrics=['accuracy'])
generator = model.fit(x_train_res,y_train_res,epochs=10,steps_per_epoch=len(x_train_res)//64)
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(word) 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')
from flask import Flask, render_template, request
import pickle
import numpy as np
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from tensorflow.keras.models import load_model

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

@app.route('/') # rendering the html template
def home():
    return render_template('home.html')

@app.route('/spam',methods=['POST','GET'])
def prediction(): # route which will take you to the prediction page
    return render_template('spam.html')

@app.route('/predict',methods=['POST'])
def predict():
    if request.method == 'POST':
        messagev = request.form['message']

```

```

data = message

new_review = str(data)
print(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(word) 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.predicate(new_X_test)
new_X_pred = np.where(new_y_pred>0,5,1,0)
print(new_X_pred)
if new_review[0][0]==1:
    return render_template('result.html', predication="spam")
else :
    return render_template('result.html', predication="Not a Spam")
if __name__=="__main__":
    # app.run(host='0.0.0.0', port=8000,debug=True)    # running the
    port=int(os.environ.get('PORT',5000))
    app.run(debug=False)

```