OPTIMIZING SPAM FILTERING

1 INTRODUCTION

1.1 Overview

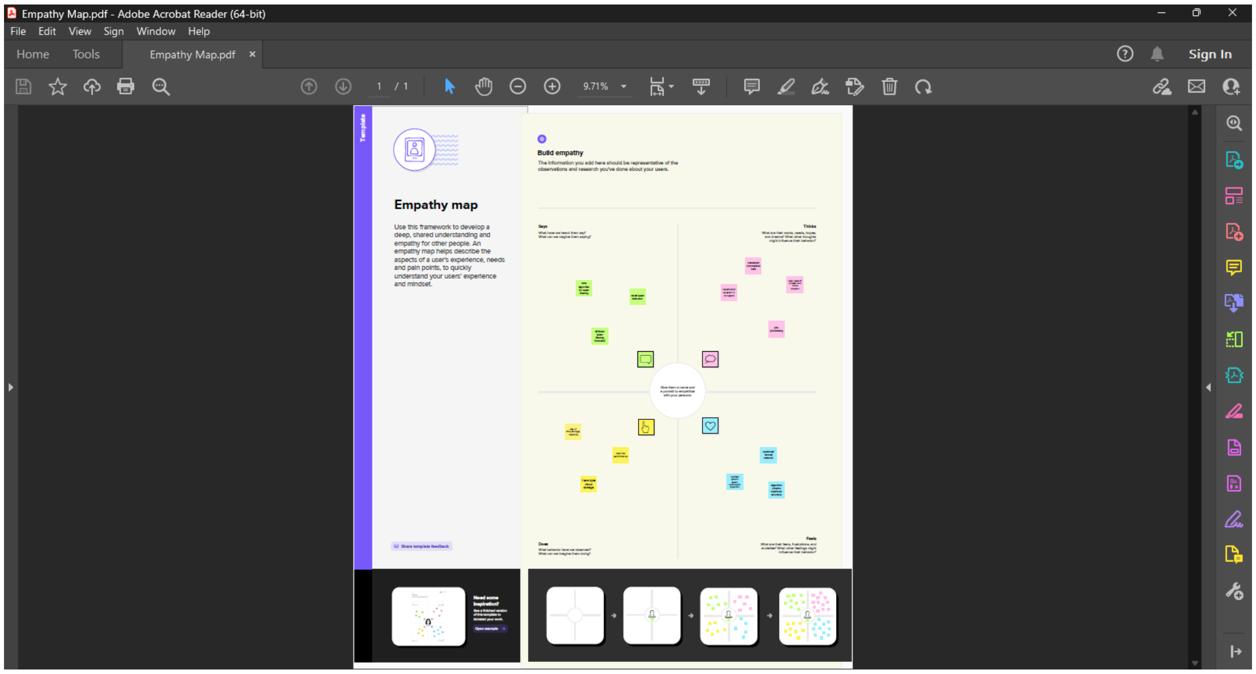
Over recent years, as the popularity of mobile phone devices has increased, Short Message Service (SMS) has grown into a multi-billion dollar industry. At the same time, reduction in the cost of messaging services has resulted in growth in unsolicited commercial advertisements (spams) being sent to mobile 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

Social Impact:- it can help protect individuals from unwanted and potentially harmful messages. Spam messages can include phishing attempts, scams, and fraud, which can have serious financial and personal consequences for recipients. By accurately identifying and flagging spam messages, the system can help prevent these types of attacks and protect individuals from falling victim to them. Business Model/Impact:- it can help protect their customers and improve their reputation. Spam messages can harm a business's reputation and lead to customer complaints and lost business. By accurately identifying and flagging spam messages, the system can help protect businesses and improve their customer's trus

2 Problem Definition&Design Thinking

2.1 Empathy Map


Empathy Map.pdf - Adobe Acrobat Reader (64-bit)
File Edit View Sign Window Help
Home Tools Empathy Map.pdf
Sign In
9.71%
Template

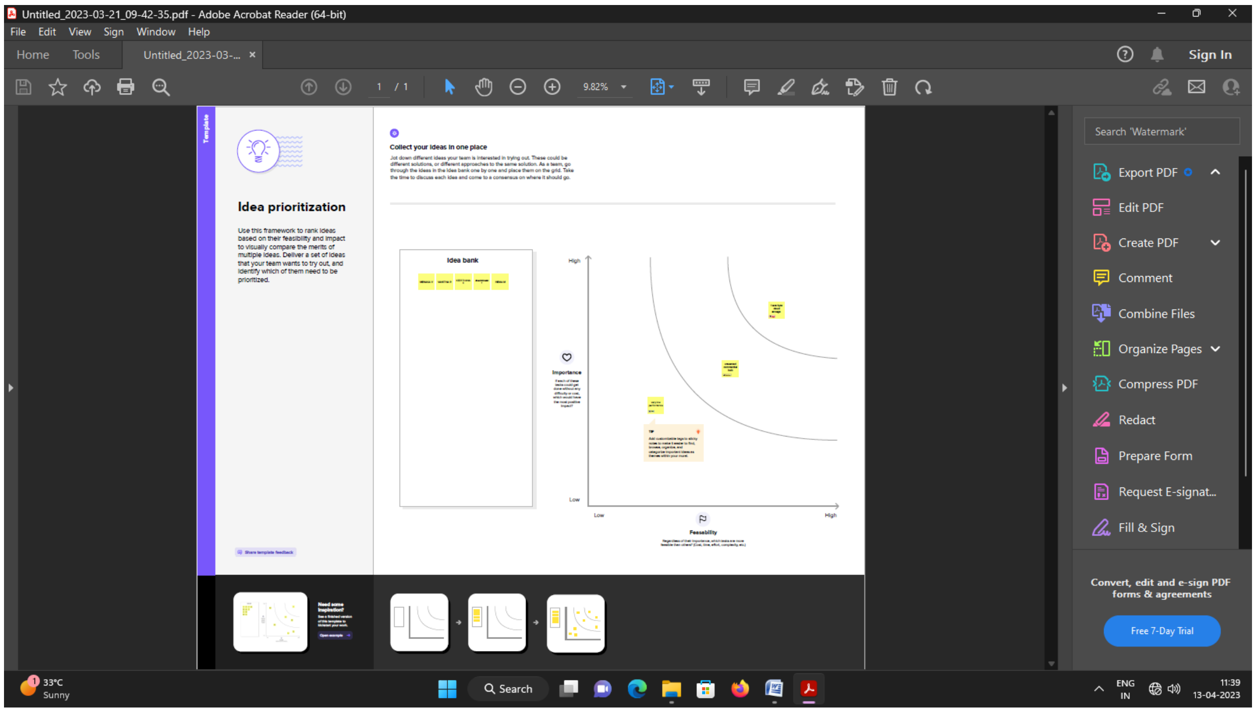
Empathy map

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.

2.2 Ideation & Brainstorming map screenshot



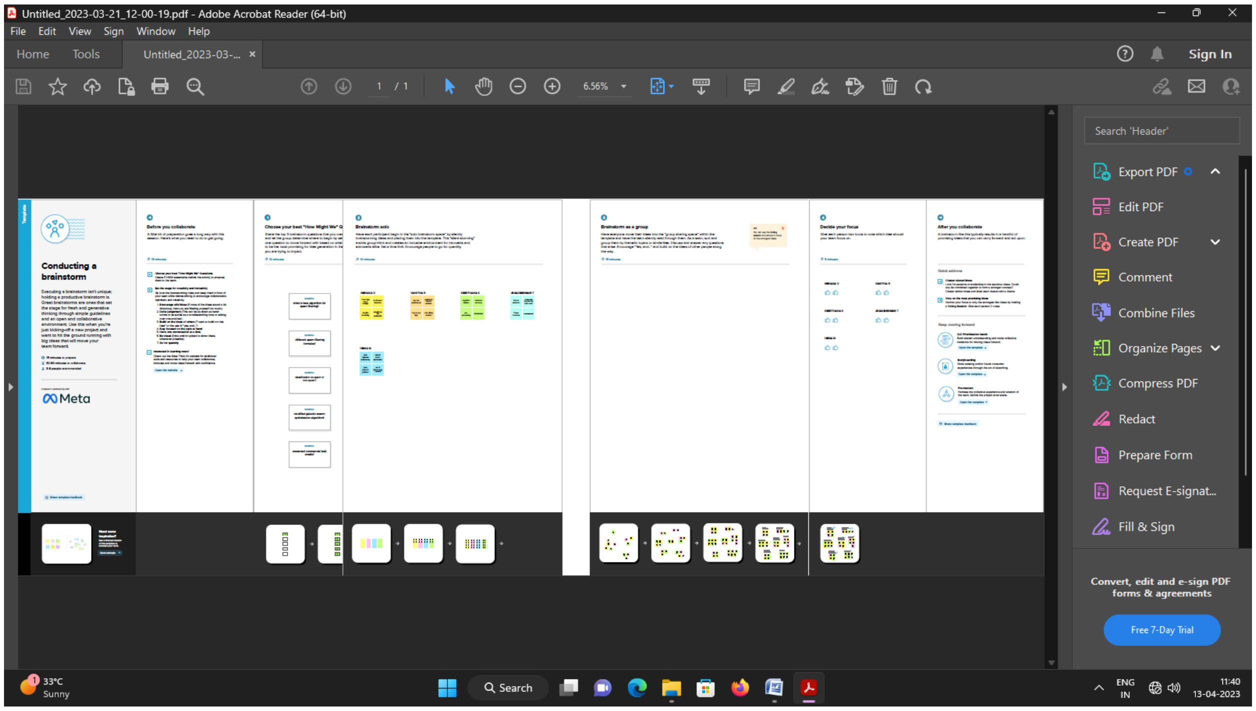
Idea prioritization

Use this framework to rank ideas based on their feasibility and impact to visually compare the merits of multiple ideas. Deliver a set of ideas that your team wants to try out, and identify which of them need to be prioritized.

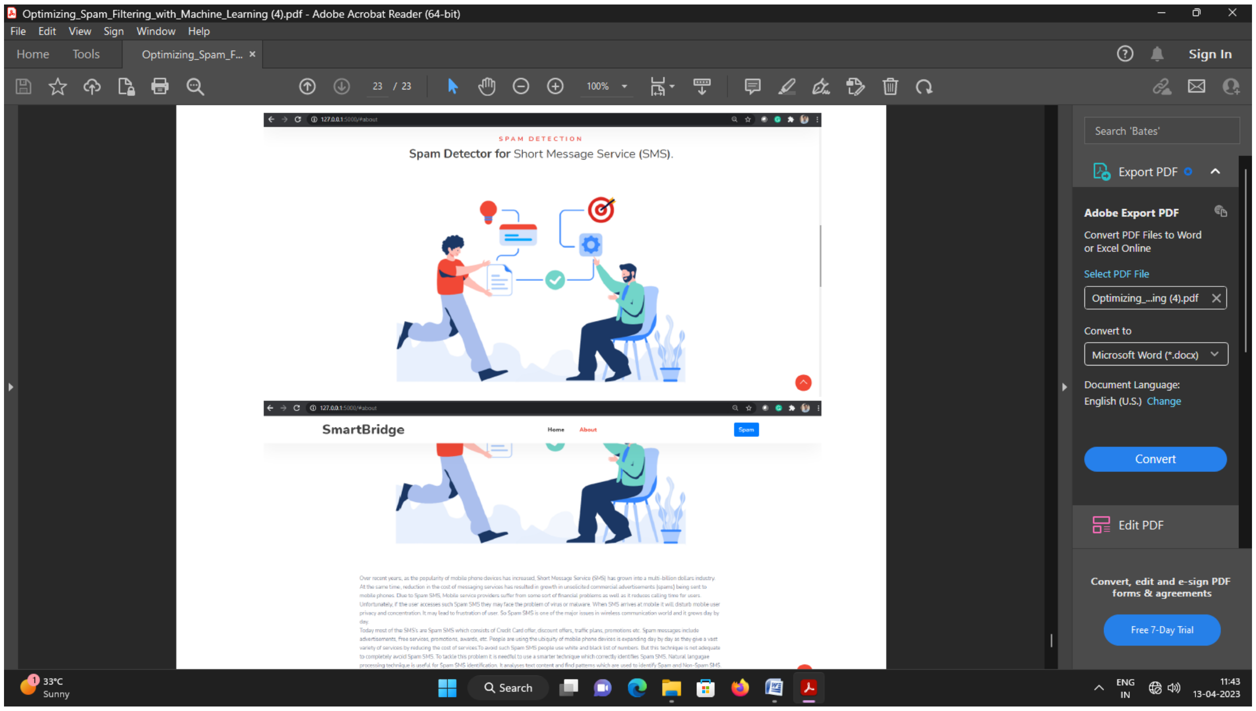
Collect your ideas in one place

Jot down different ideas your team is interested in trying out. These could be different solutions, or different approaches to the same solution. As a team, go through the ideas in the idea bank one by one and place them on the grid. Take the time to discuss each idea and come to a consensus on where it should go.

Convert, edit and e-sign PDF forms & agreements



3 Result


Optimizing_Spam_Filtering_with_Machine_Learning (4).pdf - Adobe Acrobat Reader (64-bit)
File Edit View Sign Window Help
Home Tools Optimizing_Spam_F...
Sign In
23 / 23
100%
127.0.0.1:5000/about
SPAM DETECTION
Spam Detector for Short Message Service (SMS).
SmartBridge
Home About
Spam
Search 'Bates'
Export PDF
Adobe Export PDF
Convert PDF Files to Word or Excel Online
Select PDF File
Optimizing_...ing (4).pdf
Convert to
Microsoft Word (*.docx)
Document Language:
English (U.S.) Change
Convert
Edit PDF
Convert, edit and e-sign PDF forms & agreements
Free 7-Day Trial
33°C Sunny
Search
ENG IN
11:43 13-04-2023

Over recent years, as the popularity of mobile phone devices has increased, Short Message Service (SMS) has grown into a multi-billion dollars 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 of user. So Spam SMS is one of the major issues in wireless communication world and it grows day by day.

Today most of the SMS's are Spam SMS which consists of Credit Card offer, discount offers, traffic plans, promotions etc. Spam messages include advertisements, free services, promotions, awards, etc. People are using the ubiquity of mobile phone devices is expanding day by day as they give a vast variety of services by reducing the cost of services.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 langugae processing technique is useful for Spam SMS identification. It analyses text content and find patterns which are used to identify Spam and Non-Spam SMS.

4 Advantages & Disadvantagess

- Some **anti-spam** solutions not only block specific email addresses but also search for subject lines and text in the email messages. You can customize it to block incoming emails based on senders, and even if your email address is not in the recipient field.
- Anti-spam filters automatically quarantine the spam emails, ensuring your inbox is spam free. Quarantined emails are kept for a fixed number of days and then discarded. During that period, you can check and recover any legitimate email that may have been quarantined.
- Most of the antivirus software comes with automatic filter update feature for timely detection of new types of Malware threats. Automatic updates not only helps the anti-spam software to stay up-to-date, but it also helps secure your system from new kinds of Malwar
- Some anti-spam software allows you to maintain a 'friendly' list of people whose emails you wish to accept. These emails will never be mistaken for spam as against the blacklist of spammers. You can also update the list in the future
- Emails have become a viral way of advertising, and it is time that you start filtering your emails, to avoid spam. Most of the anti-spam solutions are signature based that use their signature file (blacklist) to detect and respond to the new type of Malware.
- In signature based anti-spam software, new and unknown types of Malware goes undetected since there is a time gap between the time these new type of Malware threats are released and the time anti-spam software vendors have identified them and updated their signature file.
- This is where Containment technology comes into play.Containment technology works by keeping the threats or harmful files under control or within certain limits. The harmful files are processed in a controlled

operating system environment, thus limiting the resources and the spread of infection.

- *Comodo Dome Antispam* is the only enterprise anti-spam solution that has containment technology built-in. It uses advanced spam filters, and content analysis engines to identify and prevent unsolicited ema emails from entering your network If you are in search of a *good anti-spam solution*, look no further get
- Anti-spam refers to the use of any software, hardware or process to block spam from entering a system. The anti-spam software uses a set of protocols to determine unsolicited and unwanted messages and prevent those messages from getting to a user’s inbox.
- Most of the Anti-spam solutions that are available today can be customized as per your needs, allowing only the approved emails into your inbox. Such software always presumes that all the incoming emails are spam, and only allow those, from the people you know, to come in. There are free anti-spam software as well as paid anti-spam software.

5 Applications

- ✓ This work is devoted to the problem of optimising scores for anti-spam filters, which is essential for the accuracy of any filter based anti-spam system, and is also one of the biggest challenges in this research area. In particular, this optimisation problem is considered from two different points of view: single and multiobjective problem formulations. Some of existing approaches within both formulations are surveyed, and their advantages and disadvantages are discussed. Two most popular evolutionary multiobjective algorithms and one single objective algorithm are adapted to optimisation of the anti-spam filters’ scores and compared on publicly available datasets widely used for benchmarking purposes. This comparison is discussed, and the recommendations for the developers and users of optimising anti-spam filters are provided.

Highlights

- ❖ Analysis of current spam-filter software operation.
- ❖ Review of spam filter optimization schemes.
- ❖ Multiobjective evolutionary techniques for optimizing anti-spam filters.

- ✓ The rest of the paper is structured as follows: Section 2 introduces the target problem and surveys the techniques used for optimising scores of anti-spam filters. Section 3 describes the experimental protocol and the experimental results are provided in section 4. Finally, the conclusions and future work are drawn in section 5.

Optimisation of anti-spam filters

- ✓ Recently, the open source SpamAssassin filtering system gained popularity among SMEs users and became a reference in the anti-spam filtering domain. Its popularity is not onlydue to its public availability to research and development (becoming a disadvantage being available to spammers), but also because of its performance. SpamAssassin introduced to the anti-spam filtering domain two major features (Pérez-Díaz et al., 2012b): (*i*) the possibility of modelling the filter operation

Experimental protocol

- ✓ For setting up a protocol for testing efficiency of the selected optimisation algorithms, two major choices were made *(i)* of dataset for testing the algorithms, and *(ii)* of comparison schemes used for efficiency evaluation of each of the algorithms. For anti-spam filter optimisation using EAs it means selecting a corpus of messages to be used for classification (in two classes: spam or legitimate), and choosing among large amount of EAs performance measures.

Experimental results

- ✓ For testing the selected algorithms, Grindstone4SPAM, NSGA-II and SPEA2, according to both comparison schemes, we adopted the SpamAssassin default configuration. In particular, the threshold is set to 5 and scores range are selected in the interval [-5; 5]. SpamAssassin public mail corpus 2005 (The Apache SpamAssassin Group, 2005) is the dataset used in all experiments. NSGA-II (Deb et al., 2002) and SPEA2 (Zitzler et al., 2002) default configurations are taken as reference configurations

Conclusions and future work

- ✓ In this work, optimisation of the anti-spam filtering system was analysed from single and multiobjective points of view. A detailed literature survey has shown potential of the evolutionary approaches when applied to this domain and lead to the selection of three evolutionary algorithms for performance evaluation comparison. Two most widely used multiobjective evolutionary algorithms, NSGA-II and SPEA2, were compared with a single objective evolutionary algorithm, Grindstone4SPAM

- ✓ Spammers found and developed a wide variety of forms to distribute illegal and fraudulent advertisements. Due to the continuous changing of techniques used to distribute spam, anti-spam filters become obsolete in a short time period and need to be updated on a regular base. This situation created preconditions for the development and wide spreading of professional anti-spam filtering services. Aiming at customer satisfaction and accuracy of emails classification, the modern anti-spam filtering systems are desired to have the following properties

- ✓ Hundreds of enterprises develop and commercialise anti-spam filtering services. Most of these services are based on signup (gathering information about username, pay methods, mail transfer agent server and target domain) and *change mail exchange* (MX) register (Mockapetris,1987) of the target domain (AgentSPAM, 2012). Providing continuous updating of filtering services at affordable cost makes these services very attractive

CONCLUSION

Keywords

- Computer science
- Computer security
- Computer privacy
- Analysis of algorithms
- Machine learning
- Spam filtering
- Deep learning
- Neural networks
- Support vector machines

- Naïve Bayes

- ✓ The upsurge in the volume of unwanted emails called spam has created an intense need for the development of more dependable and robust antispam filters. Machine learning methods of recent are being used to successfully detect and filter spam emails. We present a systematic review of some of the popular machine learning based email spam filtering approaches. Our review covers survey of the important concepts, attempts, efficiency, and the research trend in spam filtering.

FUTURE

f you send marketing emails, you probably expect that your subscribers will at least see your email in their inbox. But if you're reading this article, chances are you're beginning to suspect your subscribers aren't even seeing your' emails. If you're beginning to wonder "*why is my domain email going to spam?,*" you're in the right place. In this post, we're going to take a deep dive into how spam filters work and more importantly, how to how to avoid spam filters.

Gateway spam filters are installed on servers onsite, whereas cloud-based filters run on 3rd party servers. Functionally, they serve a similar purpose, protecting a company's network by securing its digital borders.

Even though Gmail and Microsoft provide a high level of spam and malware protection on their own, cloud-based and gateway spam filters allow network administrators to have an extra level of control and insight into traffic both in and out of their network.

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
from nltk.stem.porter import PorterStemmer # stemming of words
#Load our dataset
df = pd.read_csv("spam.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()
nltk.download("stopwords")
import nltk
from nltk.corpus import stopwords
from nltk.stem import porter
import re
corpus = []
length = len(df)
from nltk.stem import PorterStemmer
df=df.apply(lambda x: x.astype(str).str.lower().str.replace(’ ’,’_’)
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()
import pickle ## importing pickle used for dumping models
pickle.dump(cv, open(’cv1.pk1’, ’wb’)) ## saving to into cv.pkl file
df.shape
df["label"].value_counts().plot(kind="bar",figsize=(12,6))
plt.xticks(np.arange(2), (’Non spam’,’spam’),rotation=0);
#preforming feature scaling op[eration using standard scaller on X part of the dataset
because]
```

```
# there different type of values in the columns
sc=StandardScaler()
x_bal=sc.fit_transforms(x_bal)
x_bal = pd.DataFrame(x_bal,columns=names)
#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)
from nltk.classify.decisiontree import DecisionTreeClassifier
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.naive_bayes import MultinomialNB
model = MultinomialNB()
from nltk.tag import sequential
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=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)
y_pred=model.predict(X_test)
y_pred
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)
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_testr,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)
print5('Accuracy Score Is:-',score*100)
cml=confusion_matrix(y_test,y_pred)
scorel=accuracy_score(y_test, y_pr)
print(cm)
print('Accuracy Score Is:- ' , score*100)
from sklearn.metrics import comfution_matrix,accurasy_score
cm=cofution_matrix(y_test,y_pr)
score = accuracy_score(y_test,y_pr)
print(cm)
print('Accuracy ScoreIs:- ' ,score*100)
from sklearn.metrics import confusion_matrix,accuracy_score
cm=cofution_matrix(y_test,y_pr)
score = accuracy_score(y_test,y_pr)
print(cm)
print('Accuracy ScoreIs:- ' ,score*100)
#Importing essential libraries
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
#Load the Multinomial Naive Baves model and CountVector
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":
message= 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.predict(new_X_test)
print(new_y_pred)
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' , prediction="Spam")
else :
return render_template('result.html', prdiction="Not a Spam")
if__name__=="__main__":
#app.run(host-='0.0.0.0', port=8000, debug=True)
port=int(os.environ.get('PORT',5000))
app.run(debug=False)
from sklearn.preprocessing import LabelEncoder
le LabelEncoder()
df['label'] = le.fit_transform(df['label'])
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 6.20, randon_state= 0)
print("Before Oversampling, counts of label '1': [],format(sum(y_train = 1)))
print("Before Oversampling, counts of label "o: (n".format(ly train == 0)))
```

```
from Inblearn.over_sampling import SHOTE
print("after oversampling, the shape of train x.format(x train res.shape))
print("After Oversampling, counts of label "1" [".format(my train res 1)))
print("After Oversampling, counts of label ".format(unty_train_res == 0)))
```