Project Report Template

1 **INTRODUCTION**

1.1 Overview

A brief description about your project

1.2 Purpose

The use of this project. What can be achieved using this.

2 **PROBLEM DEFINITION & DESIGN THINKING**

2.1 Empathy Map

Paste the empathy map screenshot

2.2 Ideation&BrainstormingMap

Paste the Ideation & brainstorming map screenshot

3 **RESULT**

Final findings(Output)of the project along with screenshots.

4 **ADVANTAGES&DISADVANTAGES**

List of advantages and disadvantages of the proposed solution

5 **APPLICATIONS**

The areas where this solution can be applied

6 **CONCLUSION**

Conclusion summarizing the entire work and findings.

7 **FUTURESCOPE**

Enhancements that can be made in the future.

8 **APPENDIX**

A.SourceCode

Attach the code for the solution built.

1.INTRODUCTION

1.1Overview

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.

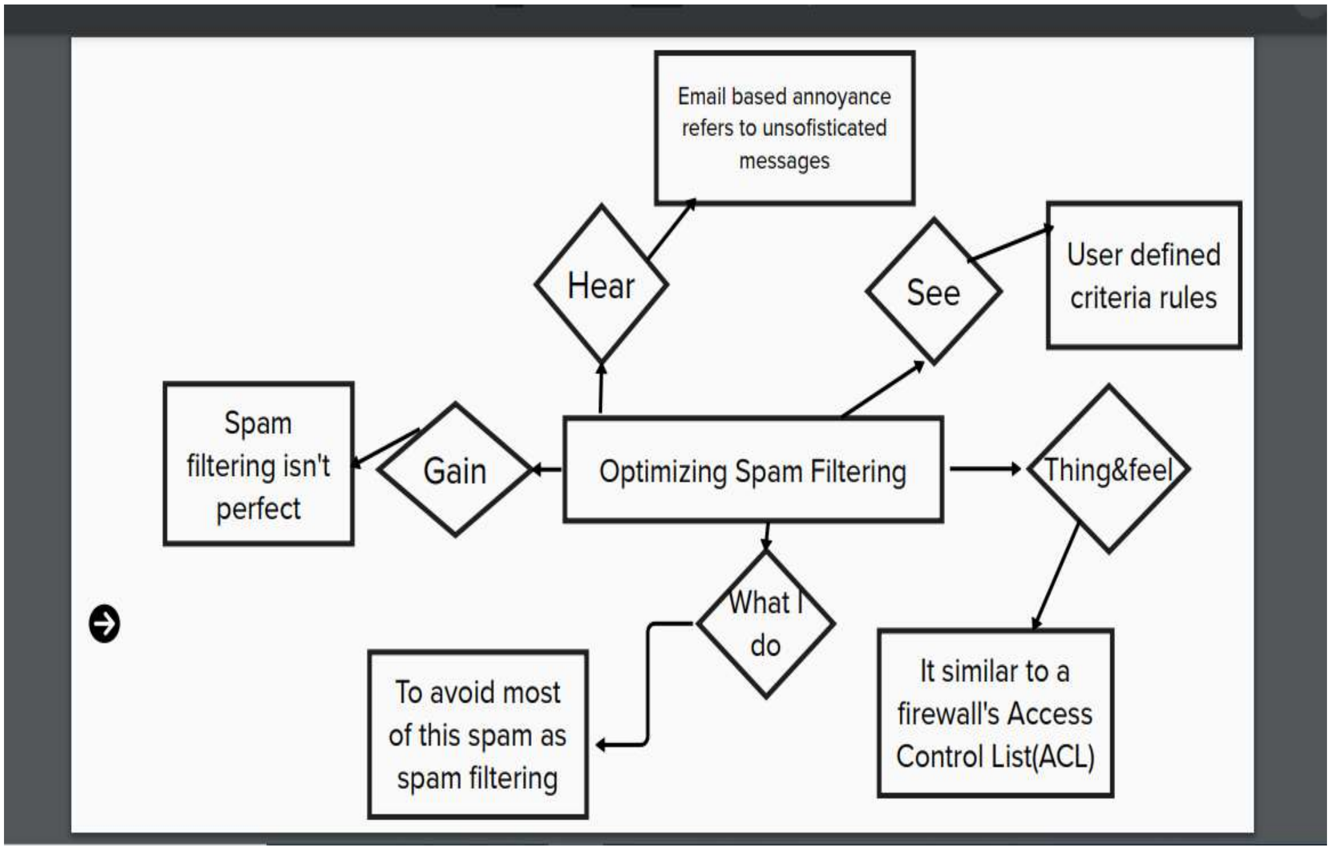
1.2 Purpose

Detecting spam alerts in emails and messages is one of the main applications that every big tech company tries to improve for its customers. Apple's official messaging app and Google's Gmail are great examples of such applications where spam detection works well to protect users from spam alerts. So, if you are looking to build a spam detection system, this article is for you. In this article, I will walk you through the task of Spam Detection with Machine Learning using Python.

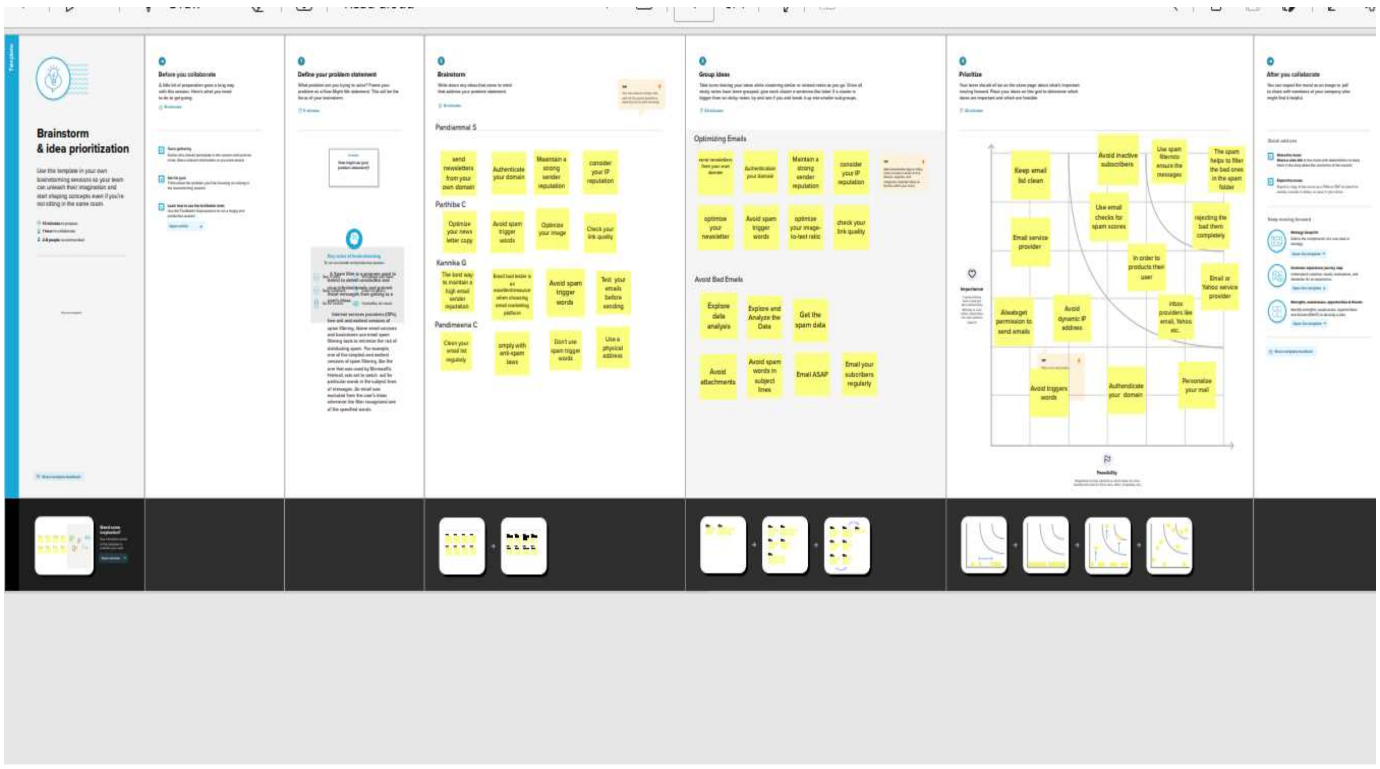
Spam detection means detecting spam messages or emails by understanding text content so that you can only receive notifications about messages or emails that are very important to you. If spam messages are found, they are automatically transferred to a spam folder and you are never notified of such alerts. This helps to improve the user experience, as many spam alerts can bother many users.

2. PROBLEM DEFINITION & DESIGN THIKING

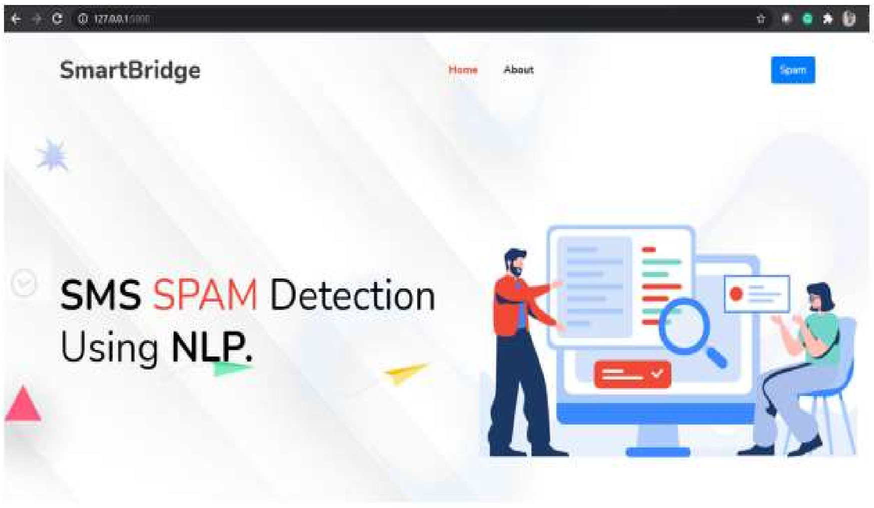
2.1 Empathy Map


Email based annoyance refers to unsofisticated messages
Hear
See
User defined criteria rules
Spam filtering isn't perfect
Gain
Optimizing Spam Filtering
Thing&feel
What I do
To avoid most of this spam as spam filtering
It similar to a firewall's Access Control List(ACL)

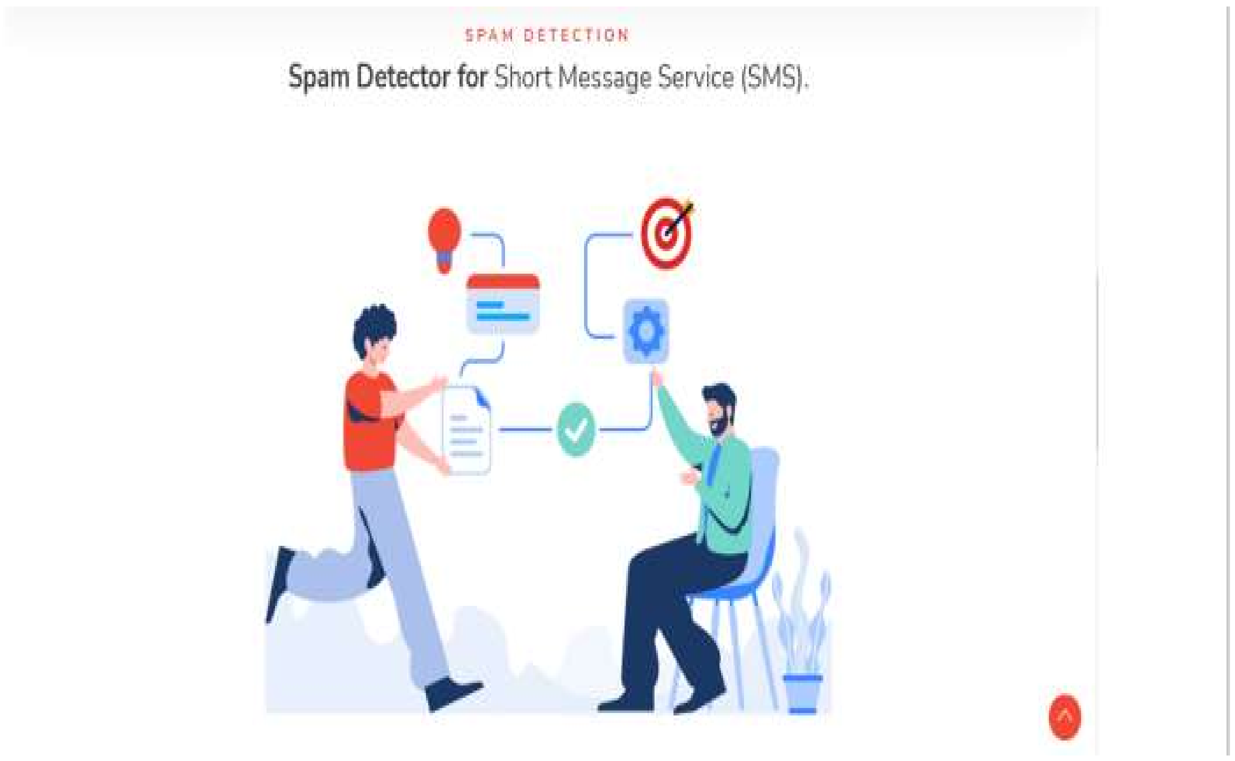
2.2 Ideation & Brainstroming Map


Brainstorm
& idea prioritization
Before you collaborate
Define your problem statement
Brainstorm
Pandiammal S
send newsletters from your own domain
Authenticate your domain
Maintain a strong sender reputation
consider your IP reputation
Parthiba C
Optimize your news letter copy
Avoid spam trigger words
Optimize your image
Check your link quality
Kannika G
The best way to maintain a high email sender reputation
Avoid spam trigger words
Test your emails before sending
Pandimeena C
Clean your email list regularly
comply with anti-spam laws
Don't use spam trigger words
Use a physical address
Group ideas
Optimizing Emails
Maintain a strong sender reputation
consider your IP reputation
optimize your newsletter
Avoid spam trigger words
optimize your image-to-text ratio
check your link quality
Avoid Bad Emails
Explore data analysis
Explore and Analyze the Data
Get the spam data
Avoid attachments
Avoid spam words in subject lines
Email ASAP
Email your subcribers regularly
Prioritize
Keep email list clean
Avoid inactive subscribers
Use spam filters to ensure the messages
The spam helps to filter the bad ones in the spam folder
Use email checks for spam scores
rejecting the bad them completely
Email service provider
In order to products their user
Email or Yahoo service provider
Alwatoget permission to send emails
Avoid dynamic IP address
inbox providers like email, Yahoo etc.
Avoid triggers words
Authendicate your domain
Personalize your mail
Importance
Feasibility
After you collaborate

3.RESULT



- Spam Detection About Page.


SPAM DETECTION
Spam Detector for Short Message Service (SMS).

4. ADVANTAGES

Blocking Spam

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.

Quarantining Spam

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.

Automatic Filter Updates

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 Malware.

Monitoring Multiple Accounts

With this feature, you can monitor and filter out spam from multiple accounts. You can filter your home email from work email, and vice versa.

Your Personal Whitelist

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.

Reporting Spam

Some anti-spam software allows you to report spam back to the company supplying the software. It helps that company to develop a new type of filters based on the analysis of the reported spam.

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.

DISADVANTAGE

There are many people who will become nervous when they receive an unwanted mail which they usually considered as spam. These people are getting a plenty of mails every month.What they do with these is they simply arrange all the unwanted messages and wanted messages and throw it to their waste bin. To send a paper we need to cut a tree and when throwing it away we are polluting our environment.

Here is the benefit of the electronic mailing system. In this one can easily keep the mail they want and can delete the unwanted. These messages will disappear the moment we delete it and there will not be a trail of their existence. By this way we are not only saving our time and money but also we are protecting our environment.

The mailbox is the place where people use to communicate with others and they try to keep it as good as possible. We can't predict when the evils like spam will come into our inbox. But sometimes we can get interesting offers and will be able to discover different areas of internet which we never heard of. In the future this can become our favorite hobbies.

5.APPLICATIONS

In the early days, spam filters used fairly basic detection methods, mainly looking for keywords and phrases that were known to be used by spammers, and filtering out emails that contained those phrases.

As spam filters evolved, so did the spammers. They caught onto the basic keyword and phrase detection, and became more sophisticated with how they constructed their emails.

At the same time, email service providers were looking for ways to tell the difference between trusted senders (those who play by the rules), and offenders. Authentication mechanisms called SPF (Sender Policy Framework), DKIM (Domain Keys Identified Mail) signing and sender scores were introduced as ways to verify the authenticity and reputation of senders, in addition to looking at the email contents.

There are now three main areas that spam filters look at:

- The contents of the email
- The legitimacy of the sender
- Engagement with the email and sender

In recent years, email providers have taken their spam filters to new levels, layering on behavioral data and machine learning to track which emails and senders result in high engagement from users, and which don’t.

The reasoning behind this is fairly straightforward. Gmail, for example, processes millions, if not billions of emails every day. Chances are if a marketing email is being sent to thousands of recipients, at least several hundred of them are using Gmail.

6.CONCLUSION

- Mailbox providers want you to use their product. The more users a mailbox provider has, the more money they can make. If users have the best, spam-free experience with Gmail, they will stick to Gmail as their primary mailbox provider.
- Spam is potentially dangerous. A big reason that it is so important for spam to be filtered is because it can contain malicious content that can spread viruses and cyber attacks. One email can very easily take down a large corporation, so businesses need the best security they can get. The mailbox provider with the best security will be the obvious choice for most companies.

7. FUTURE SCOPE

1. Provide double opt-in. Use a double opt-in form.

2. Maintain your IP reputation. Your IP address reputation is a huge factor in email deliverability.

3. Avoid trigger words or misleading subject lines.

4. Ask your subscribers to add you.

5. Send value-packed content

6. Forego attachments.

7. Follow the law.

8. Use proper text and image ratio.

8. APPENDIX

```
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

df=pd.read_csv("spam.csv",(encoding="latin")

df.head()

from google.colab import drive

drive.mount('/content/drive')

df.info()

df.isna().sum()

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

# bootom 5 rows of the dataframe

df.tail()

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

df['label'] = le.fit_transform(df['label'])
```

```
#Spliting 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%

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)))

#import SMOTE module from imblearn library

#pip install imblearn(if you don't have imblearn in your system)

from imblearn.over_sampling import SMOTE

sm=SMOTE(randam_state = 2)

X_train_res,y_train_res = sm.fit_resample(x_train,y_train.revel())



 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.dowload("stopwords")
```

```
[nltk_data] Dowloading package stopwords to

[nltk_data]    c:\Users\smart\AppData\Roaming\nltk_data...

[nltk_data]  package stopwords is already up-to-date!

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)

from sklearn.feature_extraction.text import CountVectorizer
```

```
cv = CountVectorizer(max_features=35000)

X = cv.fit_transfrom(corpus).toarray()

df.describe()

df.shape

df["label"].value_counts().plot(kind="bar",figsize=(12,6))

plt.xticks(np.arange(2),('Non spam','spam'),rotation=0);

# perfroming feature Scaling op[eration using standard scaller on x part of the dataset because]

# there different type of values in the colums

sc=StandardScaler()

x_bal=sc.fit_transform(x_bal)

x_bal = pd.DataFrame(x_bal,colums=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)



##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 RandamForestClassifier

model1 =RandomForestClassifier()

model1.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.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_initialiser="random_uniform"))

model.add(Dense(units=100,activation="relu",kernal_initializer="randam_uniform"))

model.add(Dense(units=100,activation="relu",kernal_initializer="randam_uniform"))

model.add(Dense(units=1,activation="sigmoid"))

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

```
generator = model.fit(x_train_res,y_train_res,epochs=10,steps_per_epoch=len(x_train_res)//64)

generator = model.fit(x_train_res,y_train_res,eopchs=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.matrics import confusion_matrix,accuracy_score

cm = confusion_matrix(y_test,y_pr)

score = accuracy_source(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)

corpus

from sklearn.feature_extraction.text import CountVectorizer

cv = CountVectorizer(max_features=35000)

x = cv.fit_transform(carpus).toarray()

import pickle

pickle.dump(cv,open('cv1.pk1','wb'))

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('/')

def home();

    return render_template('home.html')

@app.route('/spam,method=['POST','GET'])

def prediction():

  return render_template('spam.html')



@app.route('/predict',methods=['POST'])

def predict():

  if requst.method == 'POST':
```

```
message = regust.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.split()

ps= PorterStemmer()

all_stopwords =stopwords.words('enlish')

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_X_test = cv.transform(new_corpus).toarray()

print(new_x_test)

new_y_pred = loaded_model.predict(new_x_test)

new_x_pred = np.whrere(new_y_pred.0.5,1,0)

print(new_x_pred)

if new_review[0][0]==1:
      return render_template('result.html', preiction="spam")

    else if:

      return render_template('result.html', prediction="Not a spam")

if__name__=="__main__":

  port=int(os.environ.get('PORT',5000))

  app.run(debug=False)
```

```
Serving Flask app"app"(lazy loading)

Environment: production

use a production WSGI server instead.

Debug mode:off

Running on http://127.0.0.1:5000/(Press CTRL+C to quit)
```