OPTIMIZING SPAM FILTERING WITH MACHINE LEARNING

1.1 OVERVIEW:

Spam Detector is used to detect unwanted, malicious and virus infected texts and helps to separate them from the non spam texts. It uses a binary type of classification containing the labels such as ‘ham’ (non spam) and spam. Application of this can be seen in Electronic Mail (E-MAIL) where it segregates the spam emails in order to prevent them from getting into the user’s inbox.

About the Project:

In this Machine Learning Spam Filtering project, we will develop a Spam Detector app using Support Vector Machine (SVM) technique for classification and Natural Language Processing. We will detect whether the piece of input text is “ham”(nonspam) or “spam”. We will split our dataset into training and testing and then train our classifier with SVM classifier.

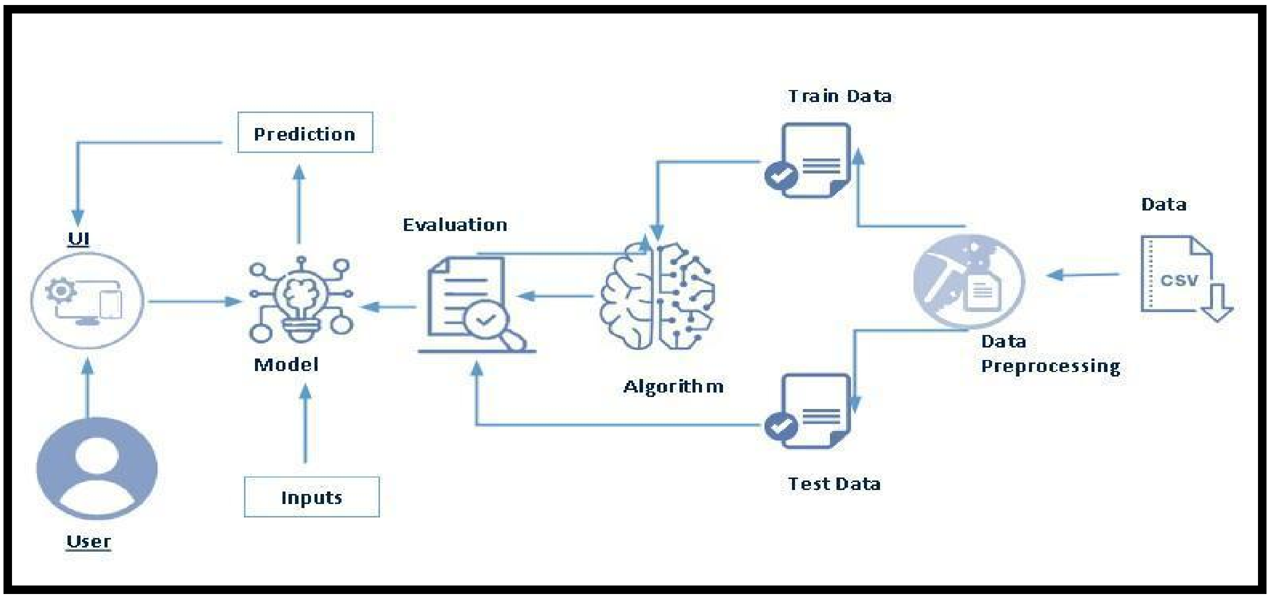
A PROJECT DESCRIPTION:

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.

Technical Architecture:



Project Flow:

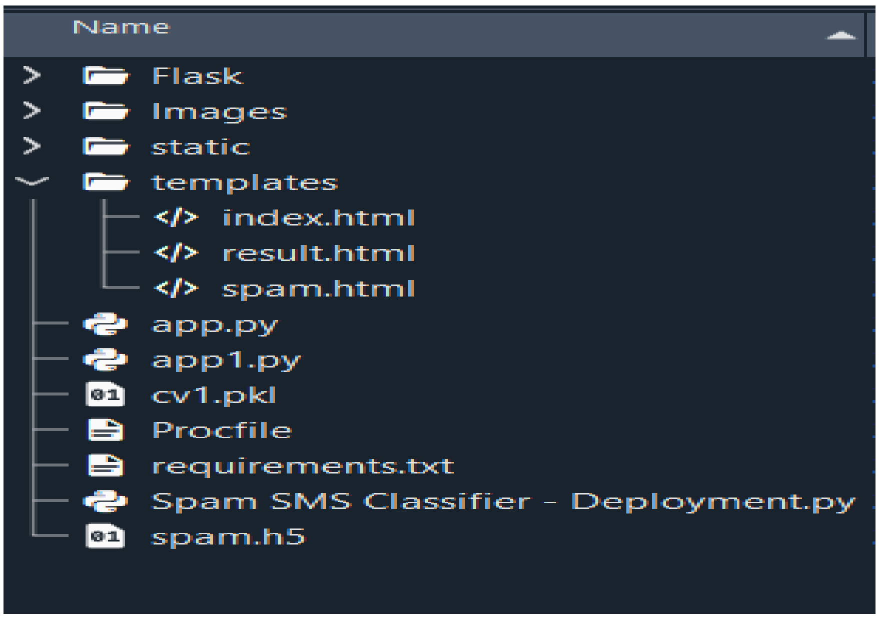
User interacts with the UI to enter the input.

Entered input is analysed by the model which is integrated.

Once model analyses the input the prediction is showcased on the UI

Project Structure:

Create the Project folder which contains files as shown below



```
Name
> Flask
> Images
> static
v templates
    index.html
    result.html
    spam.html
  app.py
  app1.py
  cv1.pkl
  Procfile
  requirements.txt
  Spam SMS Classifier - Deployment.py
  spam.h5
```

Build a flask application which needs HTML pages stored in the templates folder and a python script app.py for scripting.

Spam.h5 is our saved model. Further we will use this model for flask integration.

1.2 PURPOSE:

An email spam filter is a tool used in email hosting software that churns out unsolicited, unwanted, and virus-infested emails and keeps such emails off of the user's inbox. This **protects the user from any potential cyber threat and facilitates smooth communications and workflow**.

The Use Of This Project What Can Be Achieved

As a software developer, email is one of the very important tool for communication. To have effective communication, spam filtering is one of the important feature.

Outline:

The main goal of these two parts of article is to show how you could design a spam filtering system from scratch.

Outlines of this article are summarized as below:

EDA (Exploratory data analysis)

Data Pre-processing

Feature Extraction

Scoring & Metrics

Exploratory Data Analysis (EDA):

Exploratory Data Analysis is a very important process of data science. It helps the data scientist to understand the data at hand and relates it with the business context.

The open source tools that I will be using in visualizing and analyzing my data is **Word Cloud.**

Word Cloud is a data visualization tool used for representing text data. The **size** of the texts in the image represent the **frequency or importance** of the words in the training data.

Steps to take in this section:

Get the email data

Explore and analyze the data

Visualize the training data with Word Cloud & Bar Chart

Get the spam data:

Data is the essential ingredients before we can develop any meaningful algorithm. Knowing where to get your data can be a very handy tool especially when you are just a beginner.

Below are a few of the famous repositories where you can easily get thousand kind of data set for **free :**

UC Irvine Machine Learning Repository

Kaggle datasets

AWS datasets

For this email spamming data set, it is distributed by Spam Assassin, you can click this link to go to the data set. There are a few categories of the data, you can read the *readme.html* to get more background information on the data.

In short, there is two types of data present in this repository, which is **ham** (non-spam) and **spam** data. Furthermore, in the ham data, there are easy and hard, which mean there is some non-spam data that has a very high similarity with spam data. This might pose a difficulty for our system to make a decision.

If you are using Linux or Mac, simply do this in the terminal, *wget* is simply a command that help you to download file given an url:

https://spamassassin.apache.org/old/publiccorpus/20030228_easy_ham.tar.bz2wget
https://spamassassin.apache.org/old/publiccorpus/20030228_easy_ham_2.tar.bz2wget
https://spamassassin.apache.org/old/publiccorpus/20030228_spam.tar.bz2wget
https://spamassassin.apache.org/old/publiccorpus/20050311_spam_2.tar.bz2wget
https://spamassassin.apache.org/old/publiccorpus/20030228_hard_ham.tar.bz2

Explore and Analyze the Data

The email message content and have a basic understanding of the data

Ham

It like a normal email reply to another person, which is not difficult to classified as a ham:

This is a bit of a messy solution but might be useful -

If you have an internal zip drive (not sure about external) and Bios supports using a zip as floppy drive, use a bootable zip disk with all the relevant dos utils.

Hard Ham (Ham email that is trickier)

Hard Ham is indeed more difficult to differentiate from the spam data, as they contain some key words such as *limited time order*, *Special "Back to School" Offer*, this make it very suspicious !

Spam

One of the spam training data does look like one of those spam advertisement email in our junk folder:

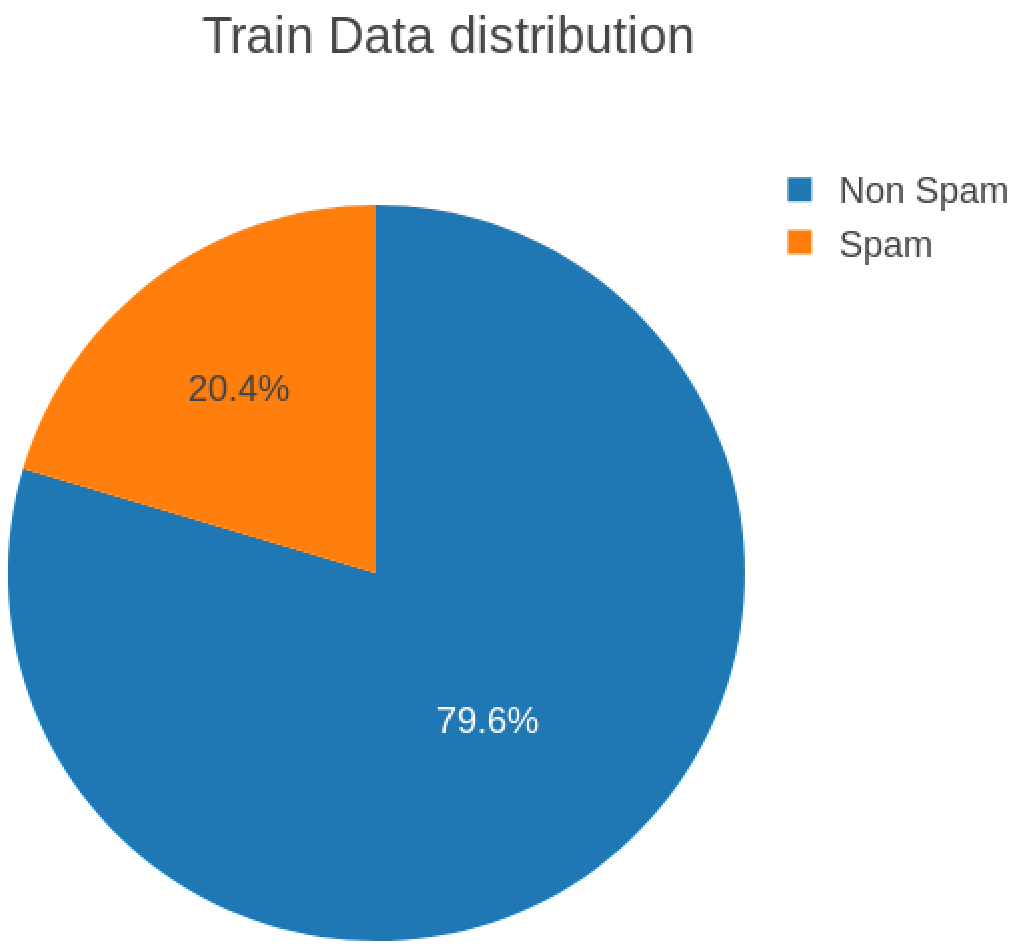
Visualization

Train Test Split:

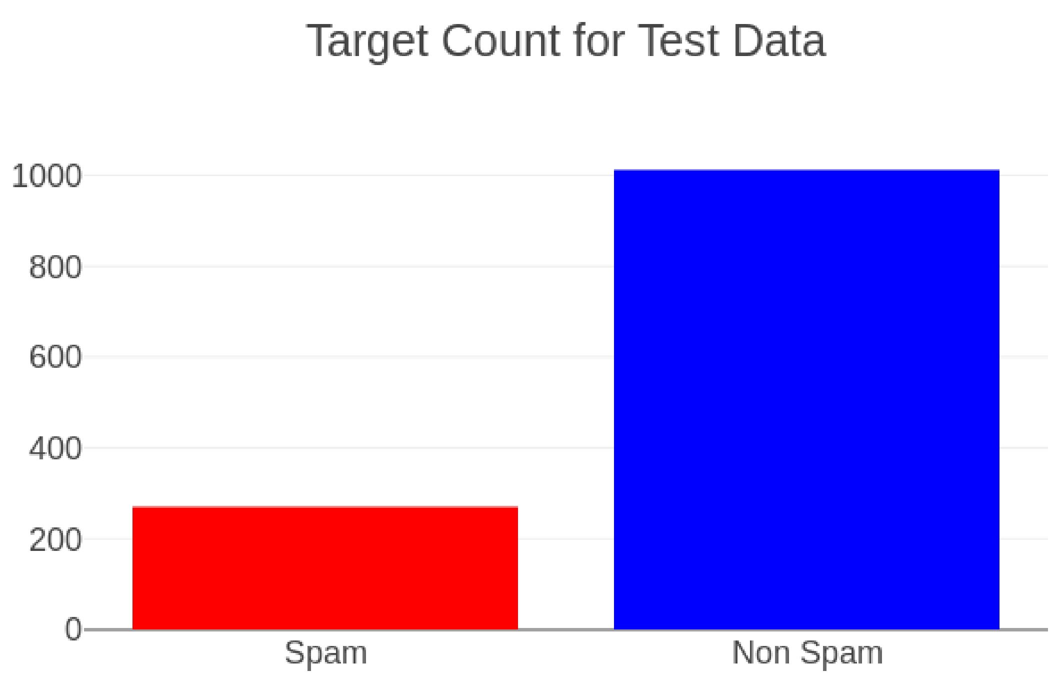
It is important to split your data set to **training set** and **test set**, so that you can evaluate the performance of your model using the test set before deploying it in a production environment.

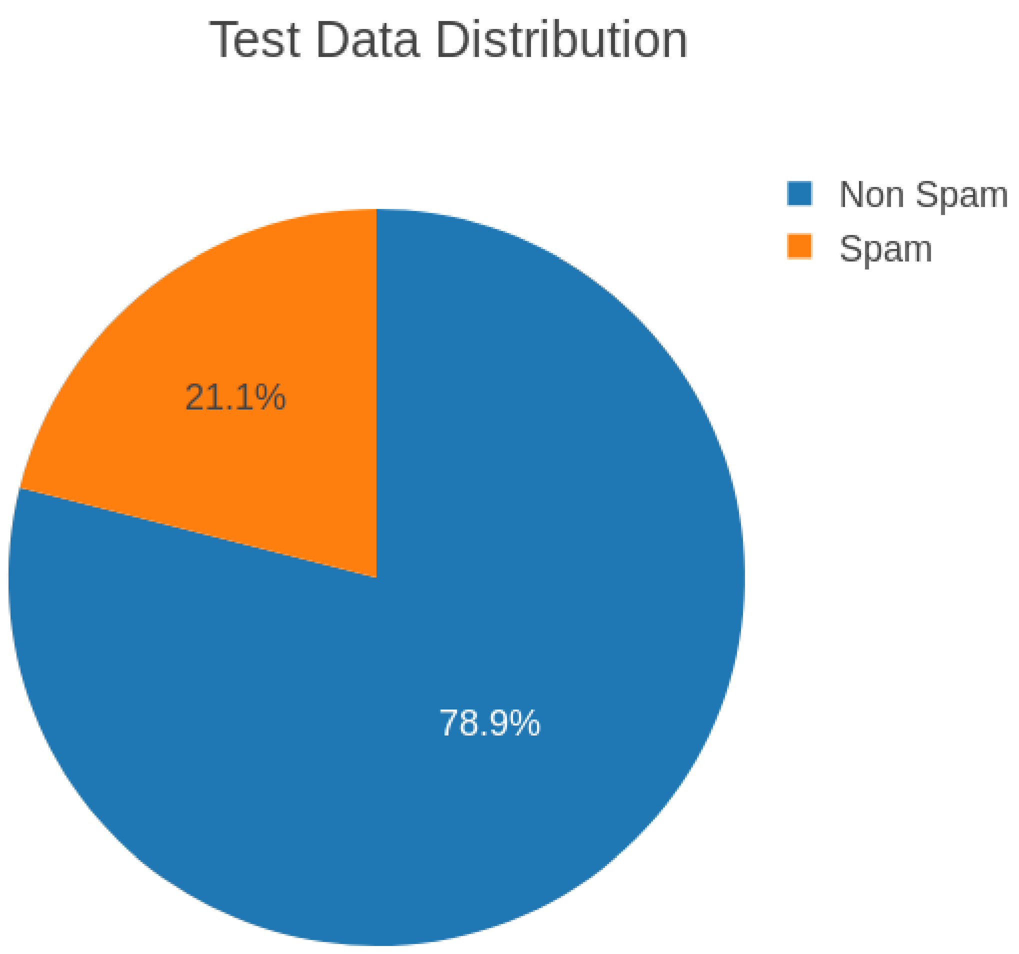
One important thing to note when doing the train test split is to make sure the distribution of the data between the training set and testing set are similar.

What it means in this context is that the percentage of spam email in the training set and test set should be similar



Train Data Distribution


Target Count for Test Data
1000
800
600
400
200
0
Spam
Non Spam



Test Data Distribution:

The distribution between train data and test data are quite similar which is around 20–21%, so we are good to go and start to process our data !

Data Preprocessing:

Text Cleaning

Text Cleaning is a very important step in machine learning because your data may contains a lot of noise and unwanted character such as punctuation, white space, numbers, hyperlink and etc.

Some standard procedures that people generally use are:

convert all letters to lower/upper case

removing numbers

removing punctuation

removing white spaces

removing hyperlink

removing stop words such as *a, about, above, down, doing* and the list goes on…

Word Stemming

Word lemmatization

The two techniques that might seem foreign to most people are **word stemming** and **word lemmatization**. Both these techniques are trying to reduce the words to its most basic form, but doing this with different approaches.

Word stemming — Stemming algorithms work by removing the end or the beginning of the words, using a list of common prefixes and suffixes that can be found in that language.

NLTK library has provided the implementation of these two algorithms, so we can use it out of the box from the library!

Import the library and start designing some functions to help us understand the basic working of these two algorithms.

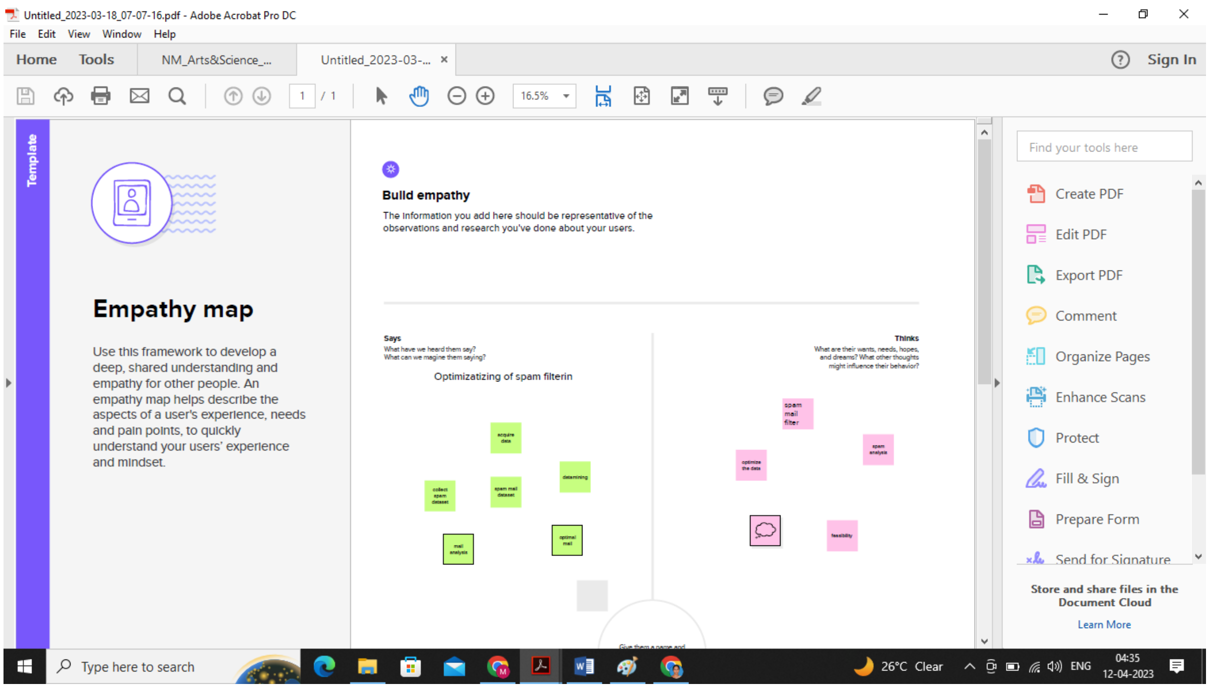
# Just import them

nltk.stem import Porter Stemmer
from nltk.stem import WordNetLemmatizer

2. PROBLEM DEFINITION AND DESIGN THINKING

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.

2.1 EMPATHY MAP



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

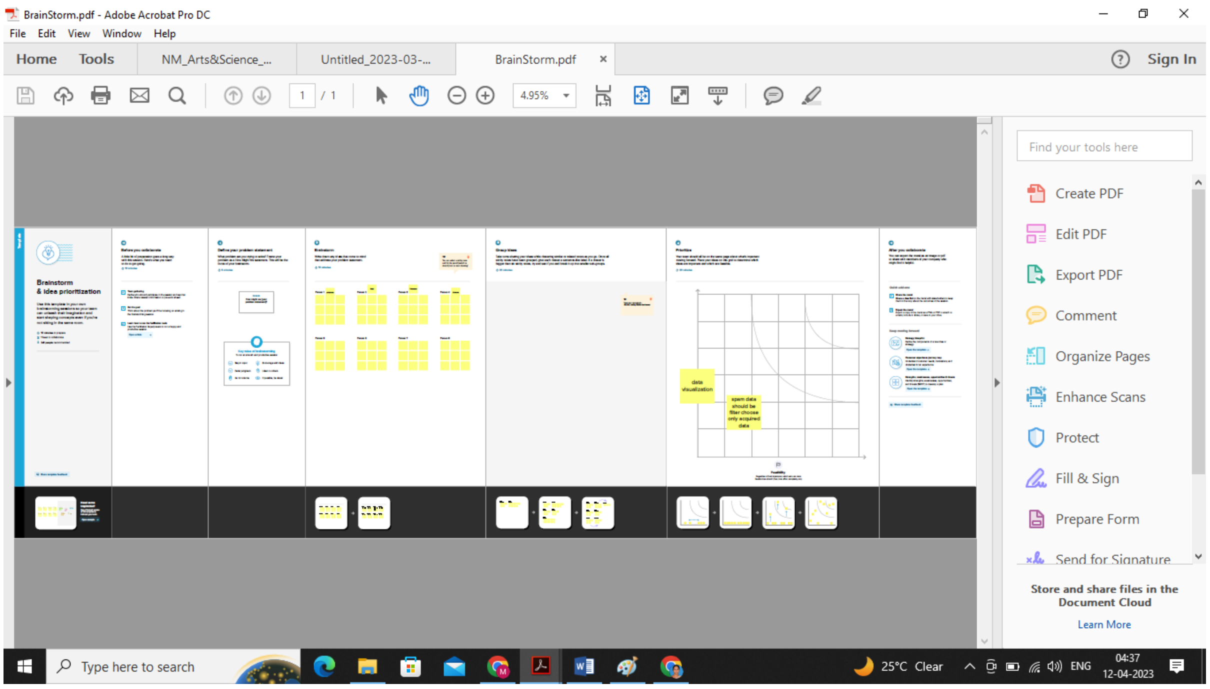
What have we heard them say? What can we magine them saying?

Optimizatizing of spam filterin

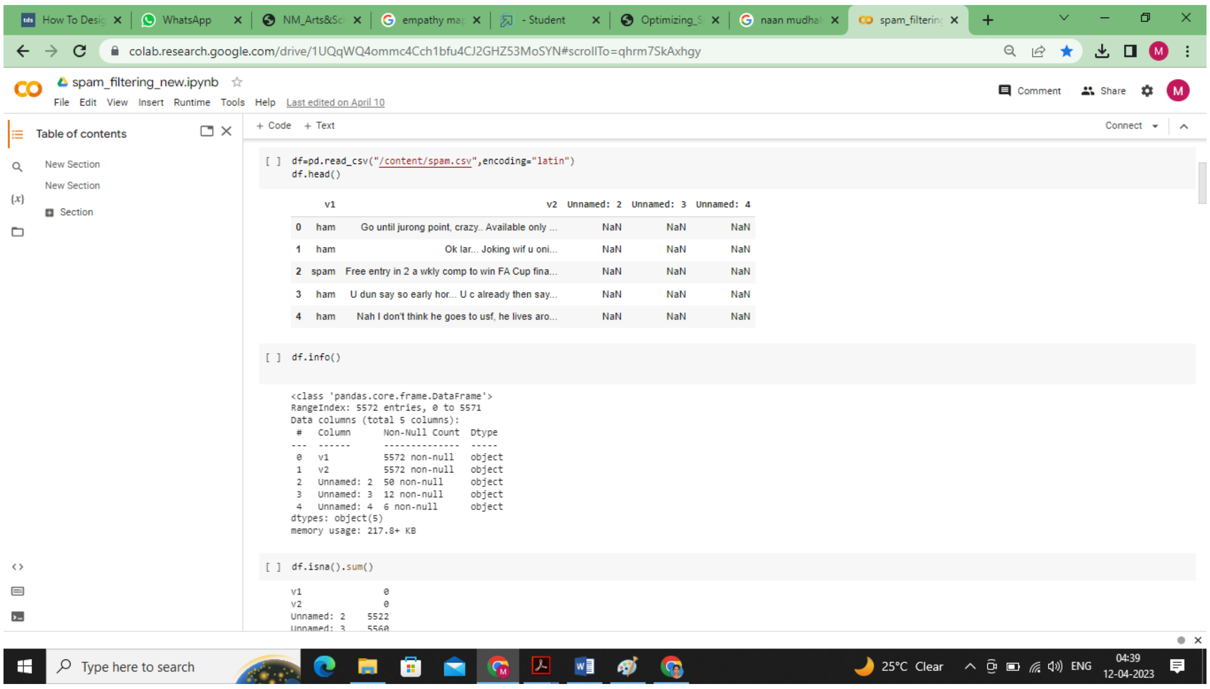
Thinks

What are their wants, needs, hopes, and dreams? What other thoughts might influence their behavior?

2.2 Ideation and Brainstorming Map:


BrainStorm.pdf - Adobe Acrobat Pro DC
File Edit View Window Help
Home Tools NM_Arts&Science_... Untitled_2023-03-... BrainStorm.pdf
Sign In
4.95%
Brainstorm & idea prioritization
data visualization
spam data should be filter choice only acquired data
Find your tools here
Create PDF
Edit PDF
Export PDF
Comment
Organize Pages
Enhance Scans
Protect
Fill & Sign
Prepare Form
Send for Signature
Store and share files in the Document Cloud
Learn More
Type here to search
25°C Clear
ENG
04:37
12-04-2023

3.RESULT:



```
df=pd.read_csv("/content/spam.csv",encoding="latin")
df.head()
```

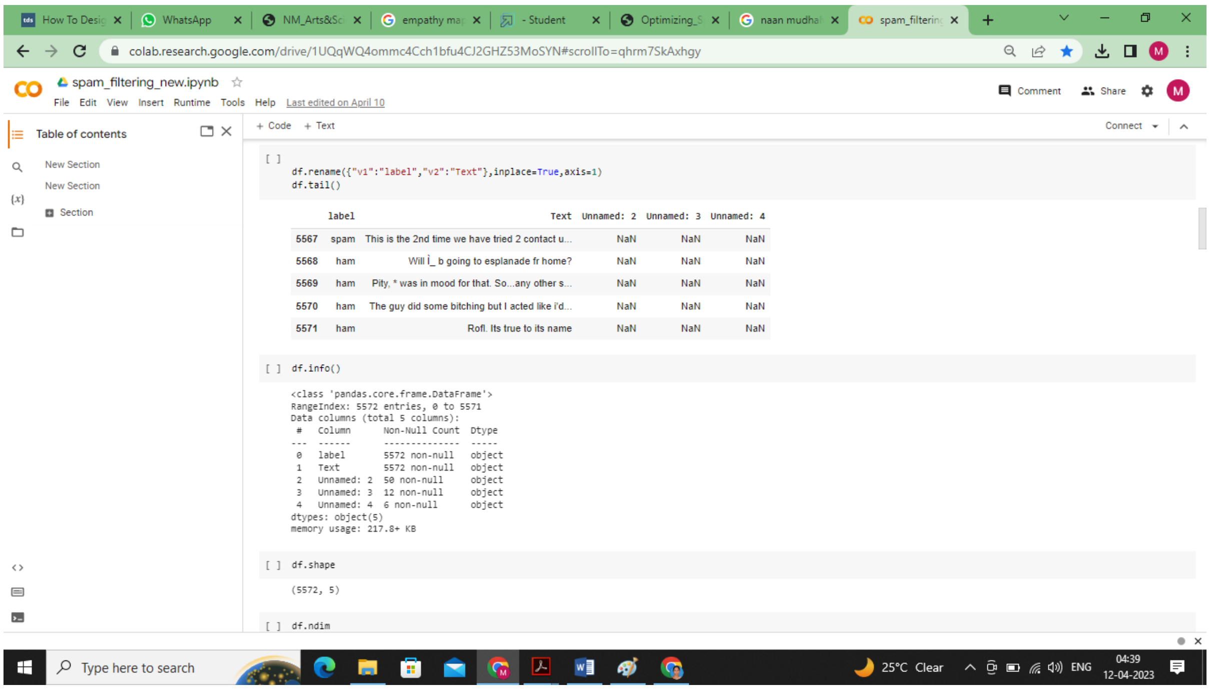
	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN
4	ham	Nah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

```
df.info()
```

```
<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
```

```
df.isna().sum()
```

```
v1               0
v2               0
Unnamed: 2    5522
Unnamed: 3    5560
```



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

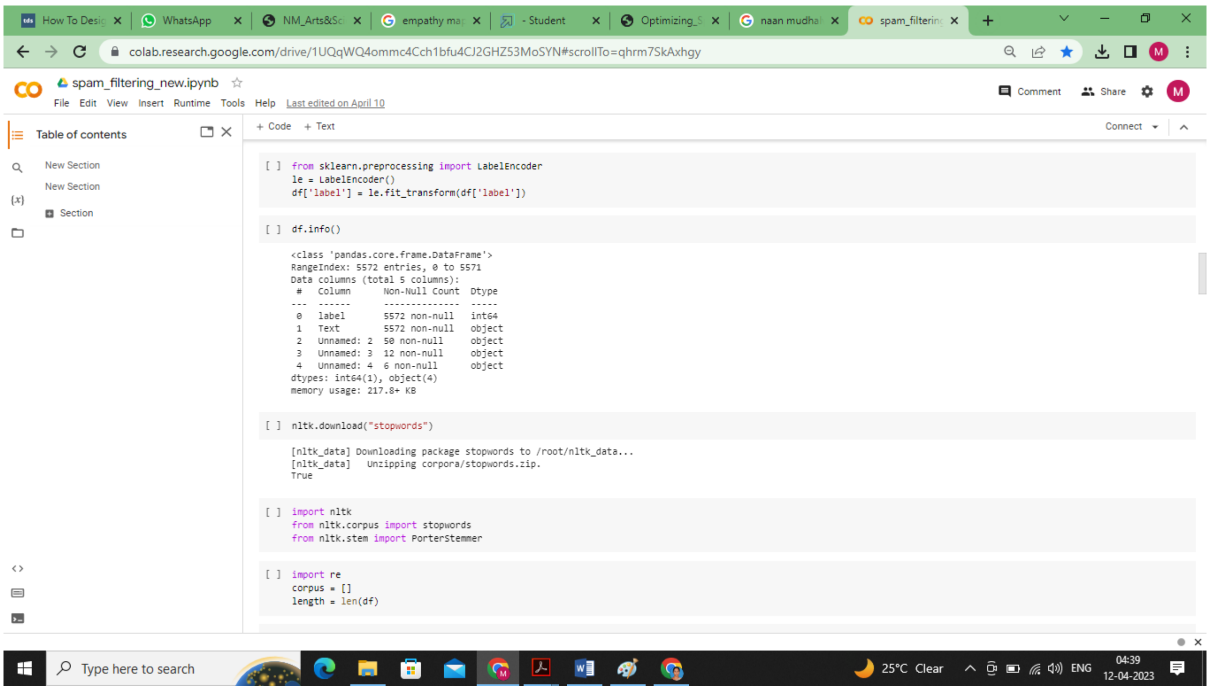
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   label       5572 non-null   object
 1   Text        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
```

```
df.shape
```

```
(5572, 5)
```

```
df.ndim
```



colab.research.google.com/drive/1UQqWQ4ommc4Cch1bfu4CJ2GHZ53MoSYN#scrollTo=qhrm7SkAxhgy

spam_filtering_new.ipynb

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

```
[ ] df.info()
```

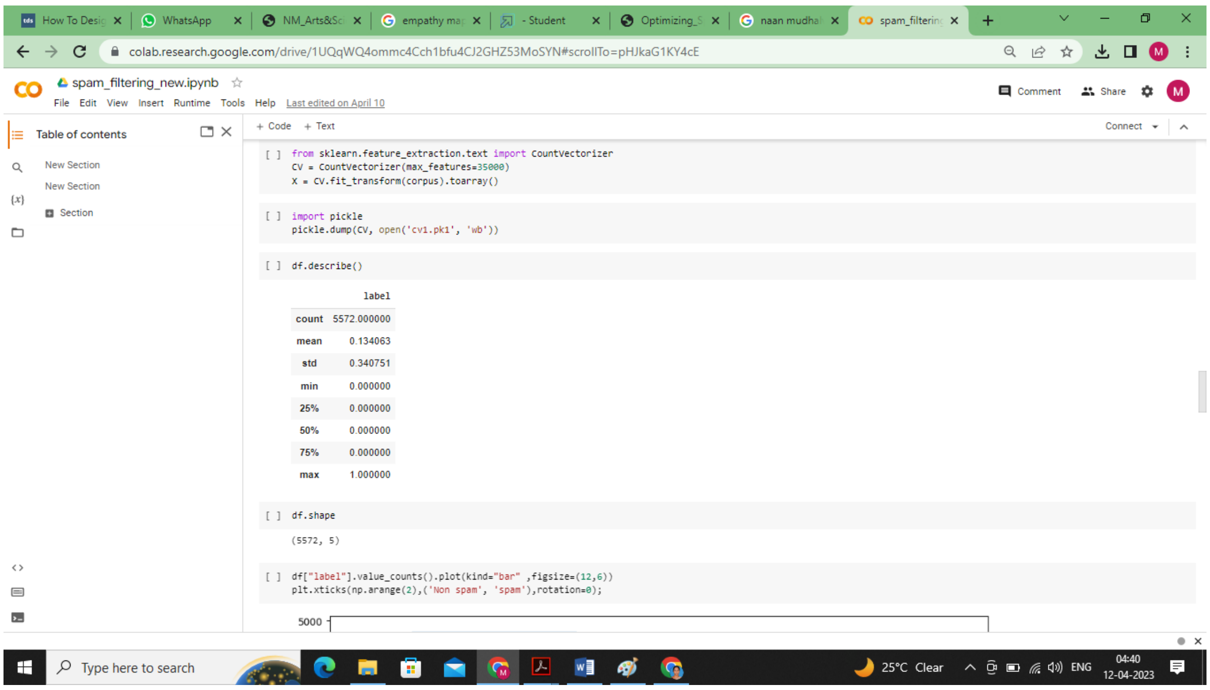
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   label       5572 non-null   int64
 1   Text        5572 non-null   object
 2   Unnamed: 2  50 non-null     object
 3   Unnamed: 3  12 non-null     object
 4   Unnamed: 4  6 non-null      object
dtypes: int64(1), object(4)
memory usage: 217.8+ KB
```

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

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
True
```

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

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



colab.research.google.com/drive/1UQqWQ4ommc4Cch1bfu4CJ2GHZ53MoSYN#scrollTo=pHJkaG1KY4cE

spam_filtering_new.ipynb

```
[ ] 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.pk1', 'wb'))
```

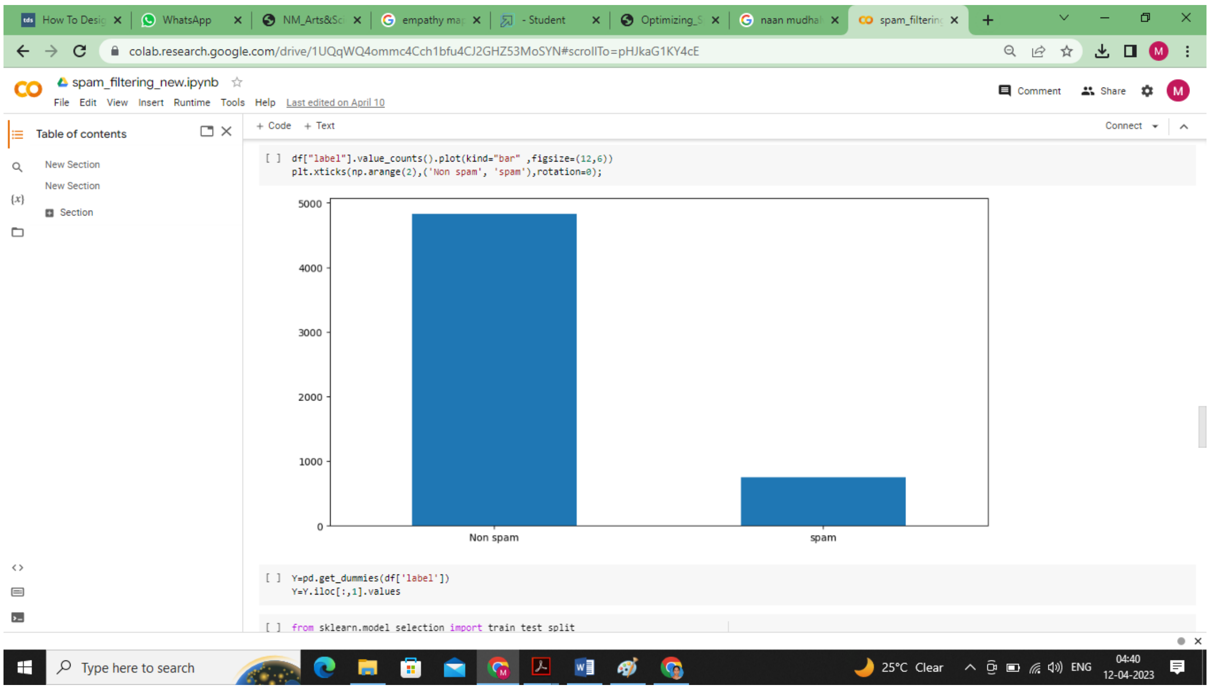
```
[ ] df.describe()
```

	label
count	5572.000000
mean	0.134063
std	0.340751
min	0.000000
25%	0.000000
50%	0.000000
75%	0.000000
max	1.000000

```
[ ] df.shape
```

```
(5572, 5)
```

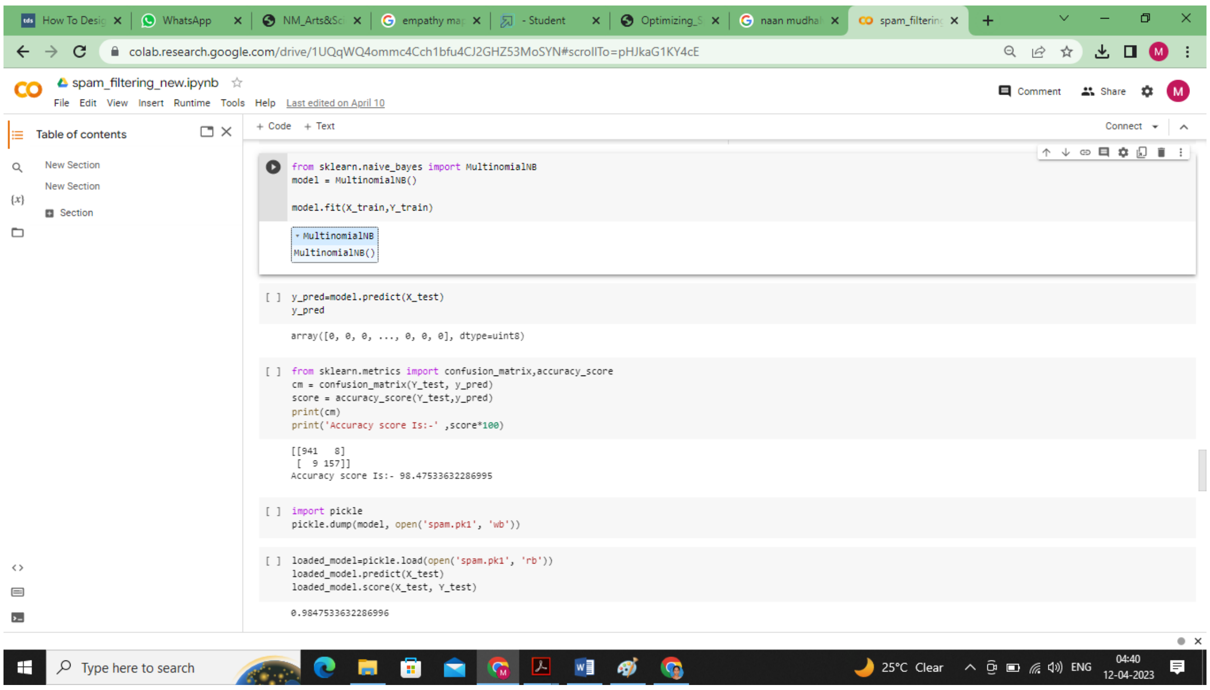
```
[ ] df["label"].value_counts().plot(kind="bar" ,figsize=(12,6))
    plt.xticks(np.arange(2),('Non spam', 'spam'),rotation=0);
```



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

```
Y=pd.get_dummies(df['label'])
Y=Y.iloc[:,1].values
```

```
from sklearn.model_selection import train_test_split
```



```
from sklearn.naive_bayes import MultinomialNB
model = MultinomialNB()

model.fit(X_train,Y_train)
```

```
▾ MultinomialNB
MultinomialNB()
```

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

```
array([0, 0, 0, ..., 0, 0, 0], dtype=uint8)
```

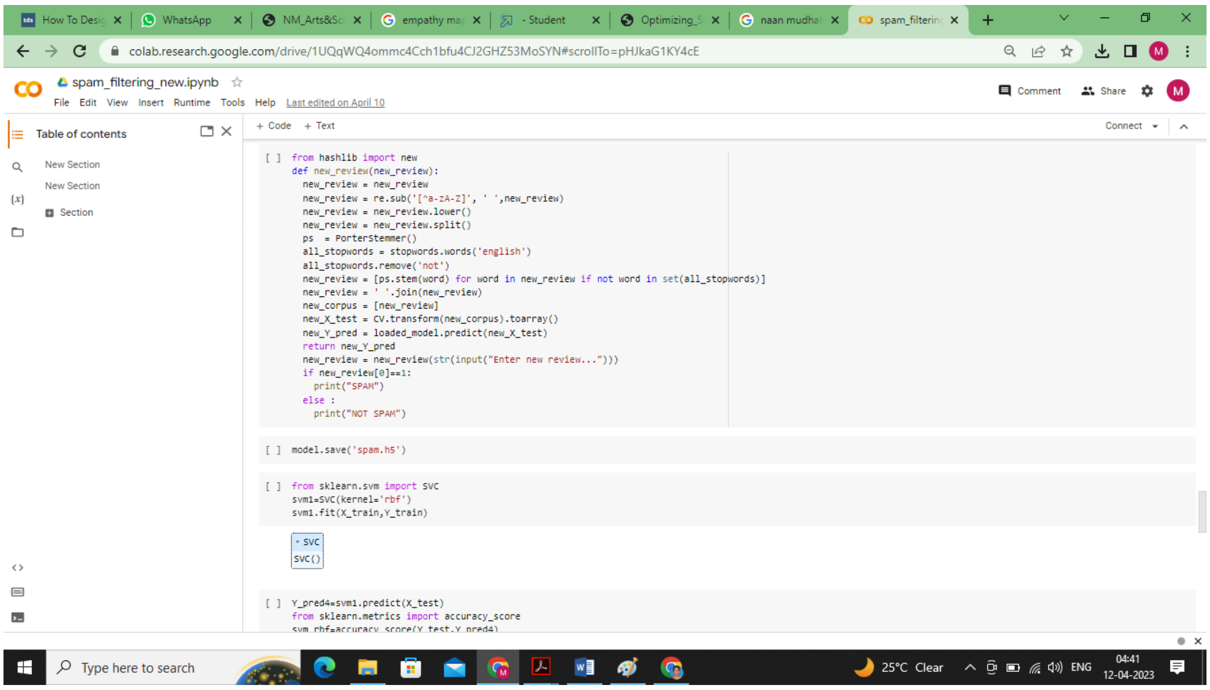
```
from sklearn.metrics import confusion_matrix,accuracy_score
cm = confusion_matrix(Y_test, y_pred)
score = accuracy_score(Y_test,y_pred)
print(cm)
print('Accuracy score Is:-' ,score*100)
```

```
[[941   8]
 [  9 157]]
Accuracy score Is:- 98.47533632286995
```

```
import pickle
pickle.dump(model, open('spam.pk1', 'wb'))
```

```
loaded_model=pickle.load(open('spam.pk1', 'rb'))
loaded_model.predict(X_test)
loaded_model.score(X_test, Y_test)
```

```
0.9847533632286996
```



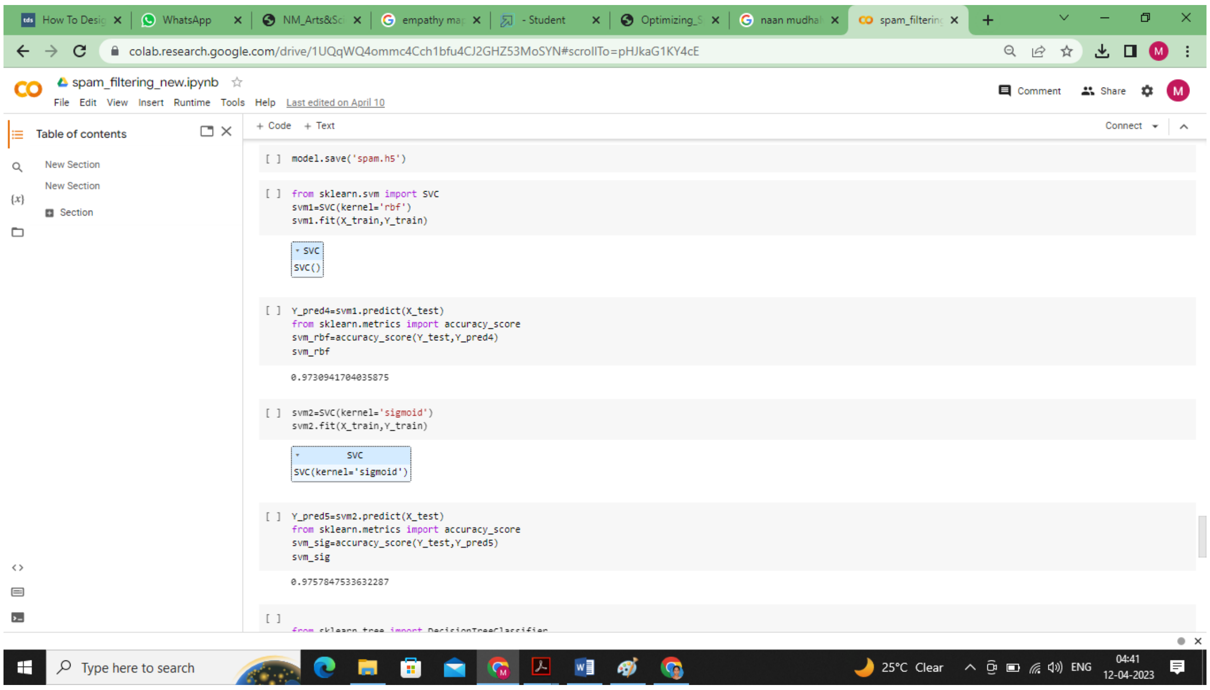
```
from hashlib import new
def new_review(new_review):
  new_review = new_review
  new_review = re.sub('[^a-zA-Z]', ' ',new_review)
  new_review = new_review.lower()
  new_review = new_review.split()
  ps  = PorterStemmer()
  all_stopwords = stopwords.words('english')
  all_stopwords.remove('not')
  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()
  new_Y_pred = loaded_model.predict(new_X_test)
  return new_Y_pred
  new_review = new_review(str(input("Enter new review...")))
  if new_review[0]==1:
    print("SPAM")
  else :
    print("NOT SPAM")
```

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

```
from sklearn.svm import SVC
svm1=SVC(kernel='rbf')
svm1.fit(X_train,Y_train)
```

```
▾ SVC
SVC()
```

```
Y_pred4=svm1.predict(X_test)
from sklearn.metrics import accuracy_score
```



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

```
from sklearn.svm import SVC
svm1=SVC(kernel='rbf')
svm1.fit(X_train,Y_train)
```

```
▾ SVC
SVC()
```

```
Y_pred4=svm1.predict(X_test)
from sklearn.metrics import accuracy_score
svm_rbf=accuracy_score(Y_test,Y_pred4)
svm_rbf
```

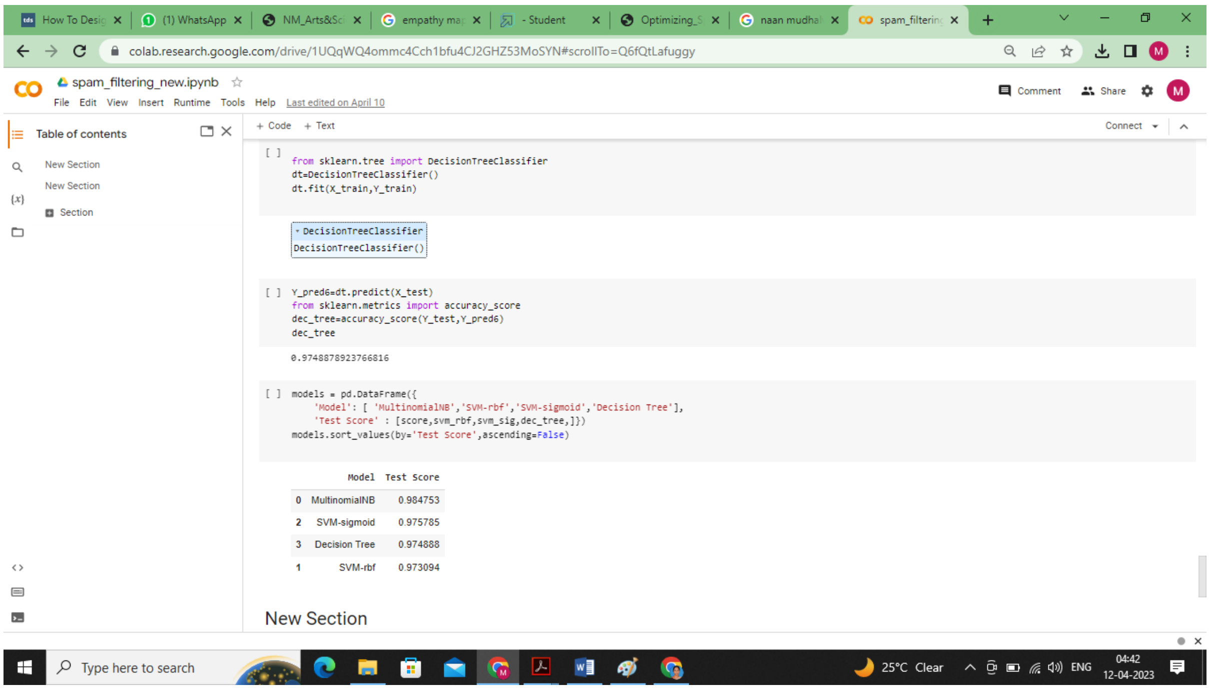
```
0.9730941704035875
```

```
svm2=SVC(kernel='sigmoid')
svm2.fit(X_train,Y_train)
```

```
▾          SVC
SVC(kernel='sigmoid')
```

```
Y_pred5=svm2.predict(X_test)
from sklearn.metrics import accuracy_score
svm_sig=accuracy_score(Y_test,Y_pred5)
svm_sig
```

```
0.9757847533632287
```



```
from sklearn.tree import DecisionTreeClassifier
dt=DecisionTreeClassifier()
dt.fit(X_train,Y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier()
```

```
Y_pred6=dt.predict(X_test)
from sklearn.metrics import accuracy_score
dec_tree=accuracy_score(Y_test,Y_pred6)
dec_tree
```

```
0.9748878923766816
```

```
models = pd.DataFrame({
    'Model': [ 'MultinomialNB','SVM-rbf','SVM-sigmoid','Decision Tree'],
    'Test Score' : [score,svm_rbf,svm_sig,dec_tree,]})
models.sort_values(by='Test Score',ascending=False)
```

```
            Model  Test Score
0   MultinomialNB    0.984753
2     SVM-sigmoid    0.975785
3   Decision Tree    0.974888
1         SVM-rbf    0.973094
```

New Section

4.ADVANTAGE AND DISADVANTAGE:

List Of Advantage and Disadvantage:

Advantage:

An email spam filter is a tool used in email hosting software that churns out unsolicited, unwanted, and virus-infested emails and keeps such emails off of the user's inbox. This protects the user from any potential cyber threat and facilitates smooth communications and workflow.

Disadvantage:

It consumes Internet resources. A deluge of spam will clog mail servers, making all email slow and burdening the ISP.

It reduces the effectiveness of legitimate advertising.

It raises costs for everyone who uses the Internet.

It exposes children to inappropriate material.

It wastes people's time. This costs the world economy billions of dollars per year in lost productivity.

It threatens the very utility of email as a form of communication.

5.APPLICATION

- Define Problem / Problem Understanding
- Specify the business problem
- Business requirements
- Literature Survey
- Social or Business Impact.
- Data Collection & Preparation
- Collect the dataset
- Data Preparation
- Exploratory Data Analysis
- Descriptive statistical
- Visual Analysis
- Model Building
- Training the model in multiple algorithms
- Testing the model

6.CONCLUSION

- Explore and understand your data
- Visualize the data at hand to gain a better intuition — Word cloud, N-gram Bar Chart
- Text Cleaning — Word Stemmer and Word Lemmatization
- Feature Extraction — Count Vectorizer, Tfidf Vectorizer, Word Embedding
- Algorithm — Naive Bayes
- Scoring & Metrics — Accuracy, Precision, Recall

7.FUTURE SCOPE

Generally, a **90% spam catch rate** (90 out of 100 spam messages are correctly identified as spam) and a false positive rate of less than 1% (less than 1 legitimate message out of a hundred incorrectly identified as spam) is considered good.

Help prevent similar attacks and respond to changing behaviour.

Time Saving

Cyber Security analyse Pattern

Fraud Detection.

8.APPENDIX

Source Code:

spam_filtering_new.ipynb

```
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
from google.colab import files
import io
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split

df=files.upload()

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

df.info()

df.isna().sum()

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

df.info()

df.shape

df.ndim

from sklearn.preprocessing import LabelEncoder
```

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

df.info()

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()

import pickle
pickle.dump(CV, open('cv1.pk1', 'wb'))
```

```
df.describe()

df.shape

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

Y=pd.get_dummies(df['label'])
Y=Y.iloc[:,1].values

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 sklearn.naive_bayes import MultinomialNB
model = MultinomialNB()

model.fit(X_train,Y_train)

y_pred=model.predict(X_test)
y_pred

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

import pickle
pickle.dump(model, open('spam.pk1', 'wb'))

loaded_model=pickle.load(open('spam.pk1', 'rb'))
loaded_model.predict(X_test)
loaded_model.score(X_test, Y_test)

from hashlib import new
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()
  new_Y_pred = loaded_model.predict(new_X_test)
  return new_Y_pred
  new_review = new_review(str(input("Enter new review...")))
  if new_review[0]==1:
    print("SPAM")
  else :
    print("NOT SPAM")

model.save('spam.h5')

from sklearn.svm import SVC
svm1=SVC(kernel='rbf')
svm1.fit(X_train,Y_train)




Y_pred4=svm1.predict(X_test)
```

```
from sklearn.metrics import accuracy_score
svm_rbf=accuracy_score(Y_test,Y_pred4)
svm_rbf

svm2=SVC(kernel='sigmoid')
svm2.fit(X_train,Y_train)

Y_pred5=svm2.predict(X_test)
from sklearn.metrics import accuracy_score
svm_sig=accuracy_score(Y_test,Y_pred5)
svm_sig

from sklearn.tree import DecisionTreeClassifier
dt=DecisionTreeClassifier()
dt.fit(X_train,Y_train)

Y_pred6=dt.predict(X_test)
from sklearn.metrics import accuracy_score
dec_tree=accuracy_score(Y_test,Y_pred6)
dec_tree

models = pd.DataFrame({
    'Model': [ 'MultinomialNB','SVM-rbf','SVM-sigmoid','Decision Tree'],
    'Test Score' : [score,svm_rbf,svm_sig,dec_tree,]})
models.sort_values(by='Test Score',ascending=False)
```

Integrate with Web Framework

In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.

This section has the following tasks

- Building HTML Pages
- Building server side script
- Run the web application

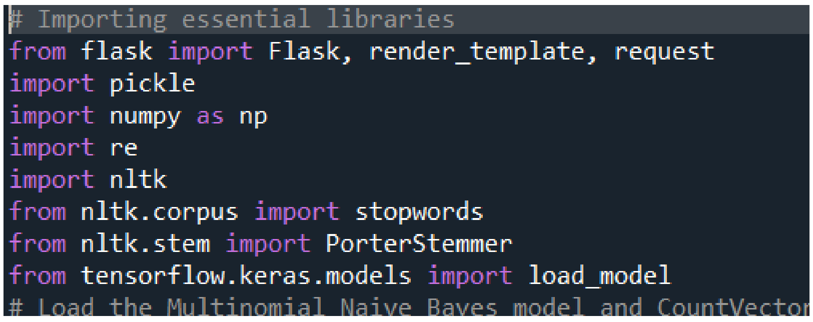
Building Html Pages:

For this project create two HTML files namely

- index.html
- spam.html
- result.html

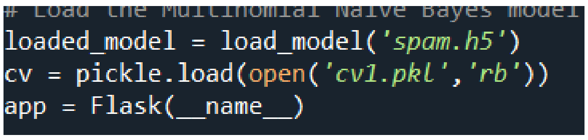
Python code:

Import the libraries



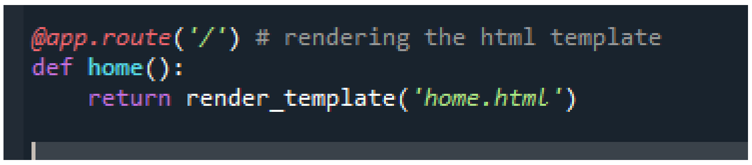
```python
# Importing essential libraries
from flask import Flask, render_template, request
import pickle
import numpy as np
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from tensorflow.keras.models import load_model
# Load the Multinomial Naive Bayes model and CountVector
```

Load the saved model. Importing the flask module in the project is mandatory. An object of Flask class is our WSGI application. Flask constructor takes the name of the current module (_name_) as argument.



```python
# Load the Multinomial Naive Bayes model
loaded_model = load_model('spam.h5')
cv = pickle.load(open('cv1.pkl','rb'))
app = Flask(__name__)
```

Render HTML page:

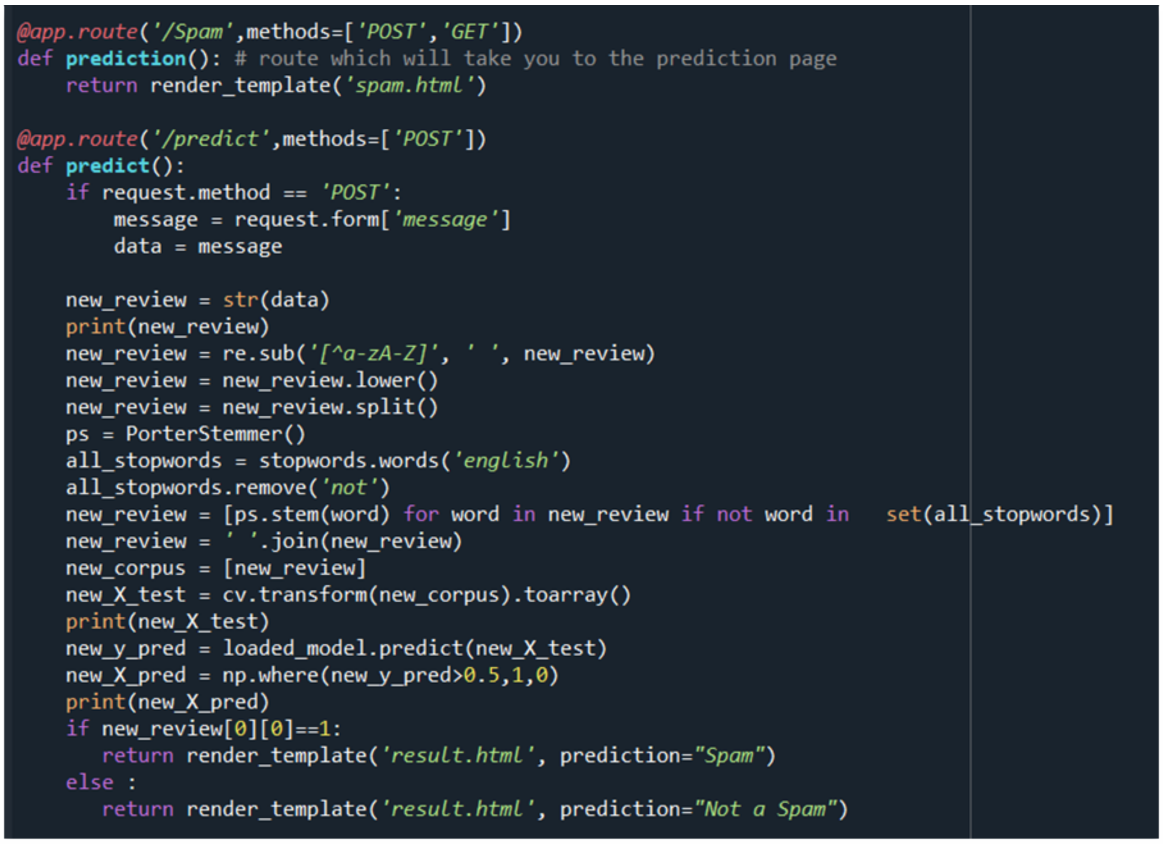


```python
@app.route('/') # rendering the html template
def home():
    return render_template('home.html')
```

Here we will be using a declared constructor to route to the HTML page which we have created earlier.

In the above example, '/' URL is bound with the home.html function. Hence, when the home page of the web server is opened in the browser, the html page will be rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method.

Retrieves the value from UI:



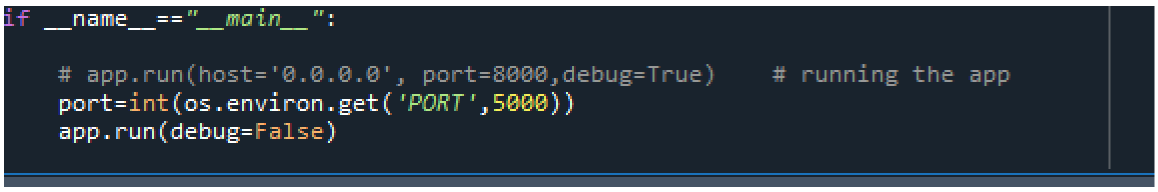
```python
@app.route('/Spam',methods=['POST','GET'])
def prediction(): # route which will take you to the prediction page
    return render_template('spam.html')

@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)
    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', prediction="Not a Spam")
```

Here we are routing our app to predict() function. This function retrieves all the values from the HTML page using Post request. That is stored in an array. This array is passed to the model.predict() function. This function returns the prediction. And this prediction value will be rendered to the text that we have mentioned in the submit.html page earlier.

Main Function:

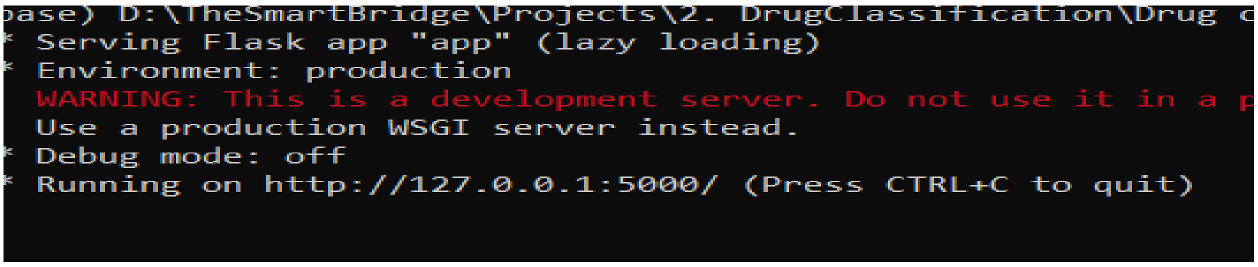


```python
if __name__=="__main__":

    # app.run(host='0.0.0.0', port=8000,debug=True)    # running the app
    port=int(os.environ.get('PORT',5000))
    app.run(debug=False)
```

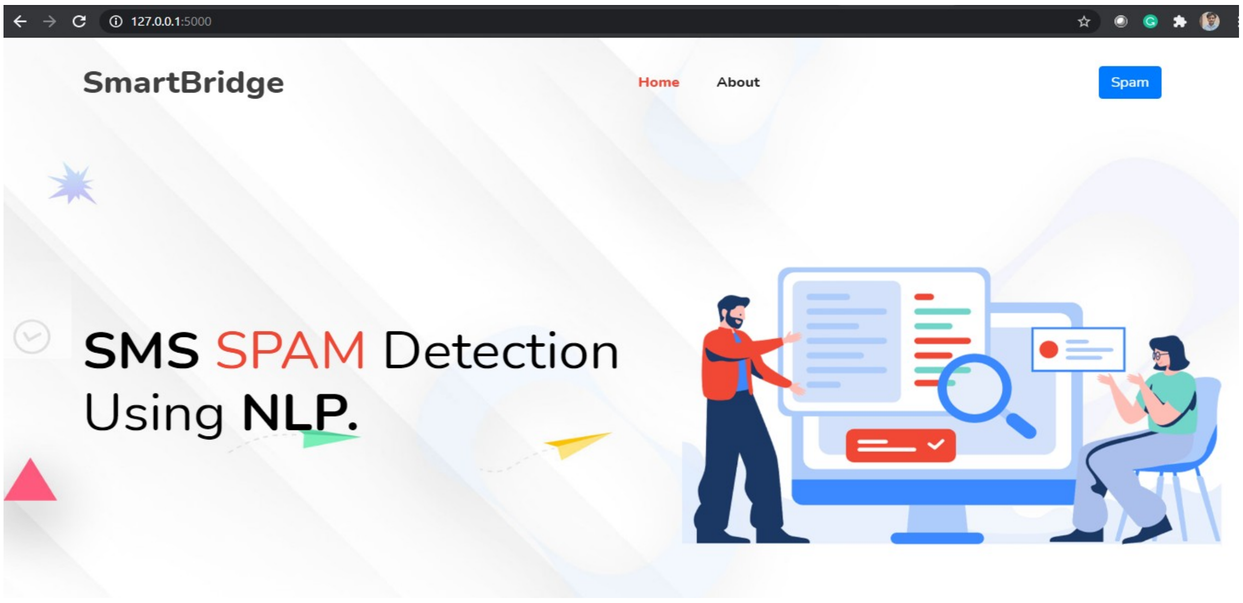
Run the web application

- Open anaconda prompt from the start menu
- Navigate to the folder where your python script is.
- Now type “python app.py” command
- Navigate to the localhost where you can view your web page.
- Click on the predict button from the top left corner, enter the inputs, click on the submit button, and see the result/prediction on the web.

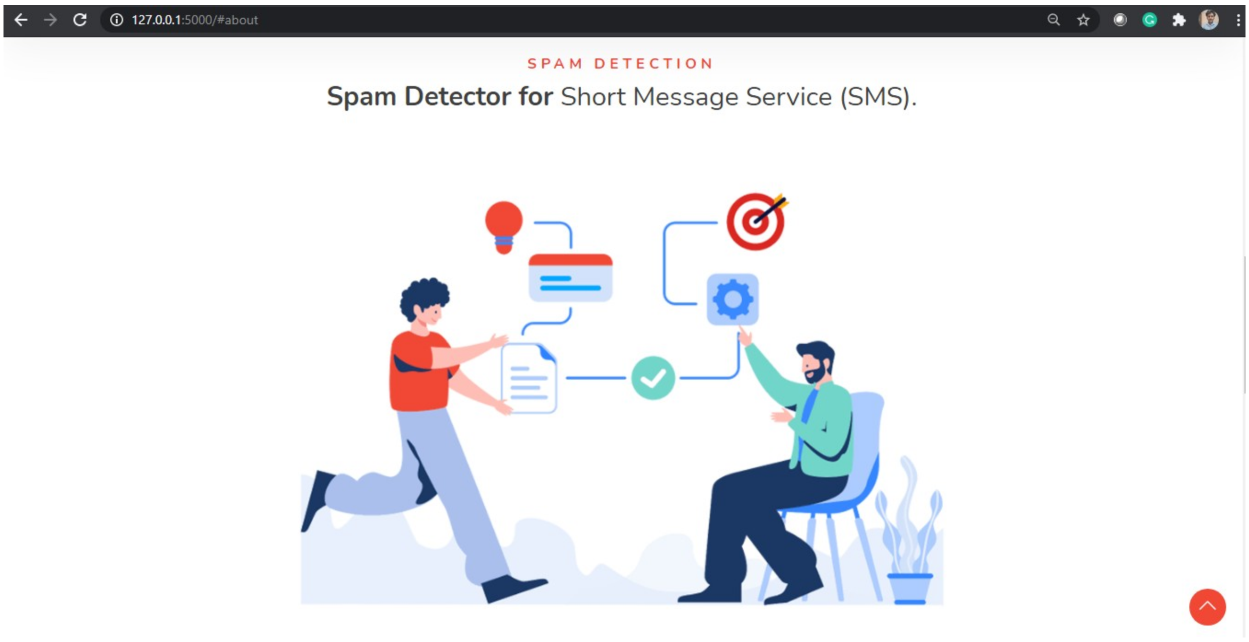


```
base) D:\TheSmartBridge\Projects\2. DrugClassification\Drug c
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a p
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

- This is the main page of Spam Detection , where you can know about the project and also from this page users can click onto the spam button and they will redirect onto the spam/ prediction page for providing the inputs.

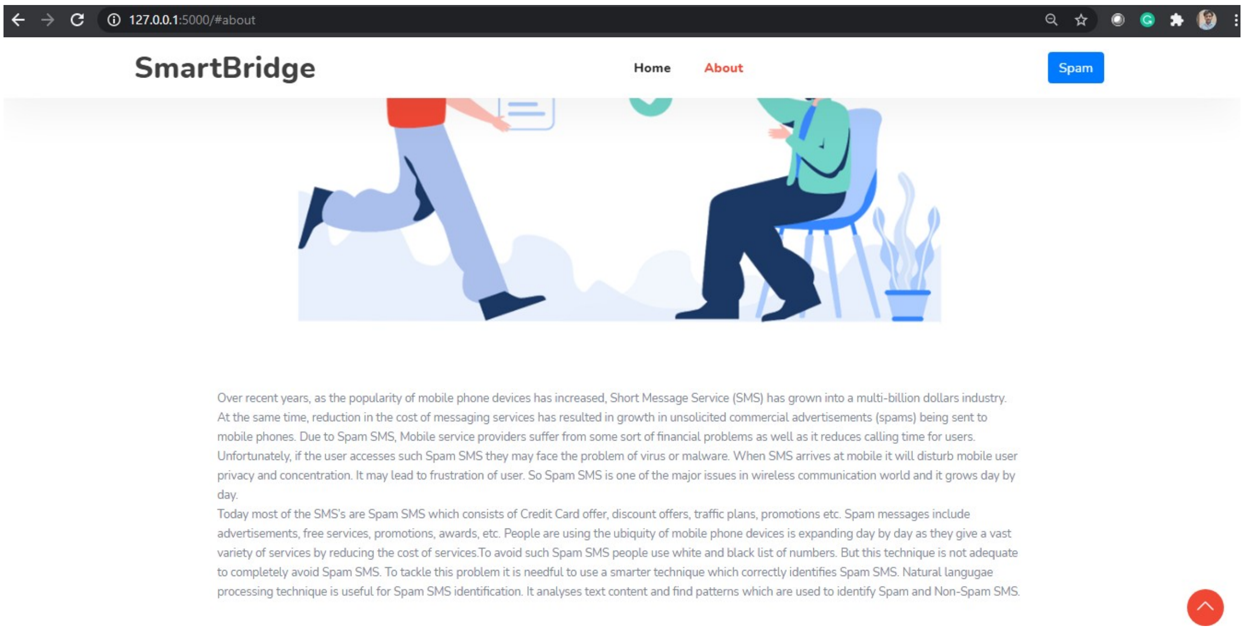


- Spam Detection About Page.



SPAM DETECTION

Spam Detector for Short Message Service (SMS).



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.