ARIGNAR ANNA GOVERNMENT ARTS COLLEGE

VILLUPURAM – 605 602.

DEPARTMENT OF COMPUTER SCIENCE

MACHINE LEARNING WITH PYTHON

Project Title : Optimizing Spam Filtering with Machine Learning

Team Id :NM 2023TMID16946

Team Leader : RAJESH.E

Team member:ARAVINTH.A

Team member:RAGHUL.S

Team member:GOKULAKRISHNAN

1. INTRODUCTION

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.

1.1 Overview:

A spam filter is a program or service that is designed to prevent unwanted, unsolicited and potentially harmful messages from reaching an email inbox, or other communication platform, such as messaging apps or social media.

The primary function of a spam filter is to distinguish between legitimate messages and unwanted ones, such as unsolicited advertising, phishing scams, malware, and other types of junk mail. Spam filters use a variety of techniques to identify spam messages, including content analysis, sender reputation, domain name analysis, and user feedback.

1.2 Purpose:

The purpose of a spam filter is to prevent unwanted, unsolicited, and potentially harmful messages from reaching a user's inbox, or other communication platform. Spam filters help to protect users from a variety of threats, including phishing scams, malware, identity theft, and other types of online fraud.

Spam filters use a variety of techniques to identify and block spam messages, including content filtering, sender reputation analysis, and machine learning algorithms. By analyzing the content, structure, and context of incoming messages, spam filters can determine the likelihood that a message is spam, and automatically move it to a separate folder, quarantine it, or delete it altogether.

Overall, the purpose of a spam filter is to help users to stay safe online, while also improving the efficiency and effectiveness of their email and messaging workflows.

1.5. 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

To accomplish this, we have to complete all the activities listed below,

- 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

- Performance Testing & Hyperparameter Tuning
  - Testing model with multiple evaluation metrics
  - Comparing model accuracy before & after applying hyperparameter tuning
- Model Deployment
  - Save the best model
  - Integrate with Web Framework
- Project Demonstration & Documentation
  - Record explanation Video for project end to end solution
  - Project Documentation-Step by step project development procedure.

Learning

Optimizing spam filtering is crucial to improving email productivity and reducing the risk of falling victim to phishing or other fraudulent emails.With the help of machine learning, we can develop smarter and more effective spam filters that can accurately identify and filter out unwanted emails, while ensuring that important messages are delivered to the inbox.

Says

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

"I'm so tired of receiving all this spam!"

"I wish I could just filter out all the junk and focus on important messages."

"It's frustrating when important emails get lost in my cluttered inbox."

Thinks

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

"There has to be a better way to manage my inbox."

"I don't want to waste my time sorting through spam emails."

"I worry that important emails might get marked as spam and I'll miss them."

optimizing spam filtering with machine learning

Does

What behavior have we observed?
What can we imagine them doing?

They may regularly delete or mark spam emails as such.

They may use filters or rules to automatically sort incoming emails into different folders.

They may be willing to invest time and effort into setting up and optimizing spam filters to improve their email experience.

Feels

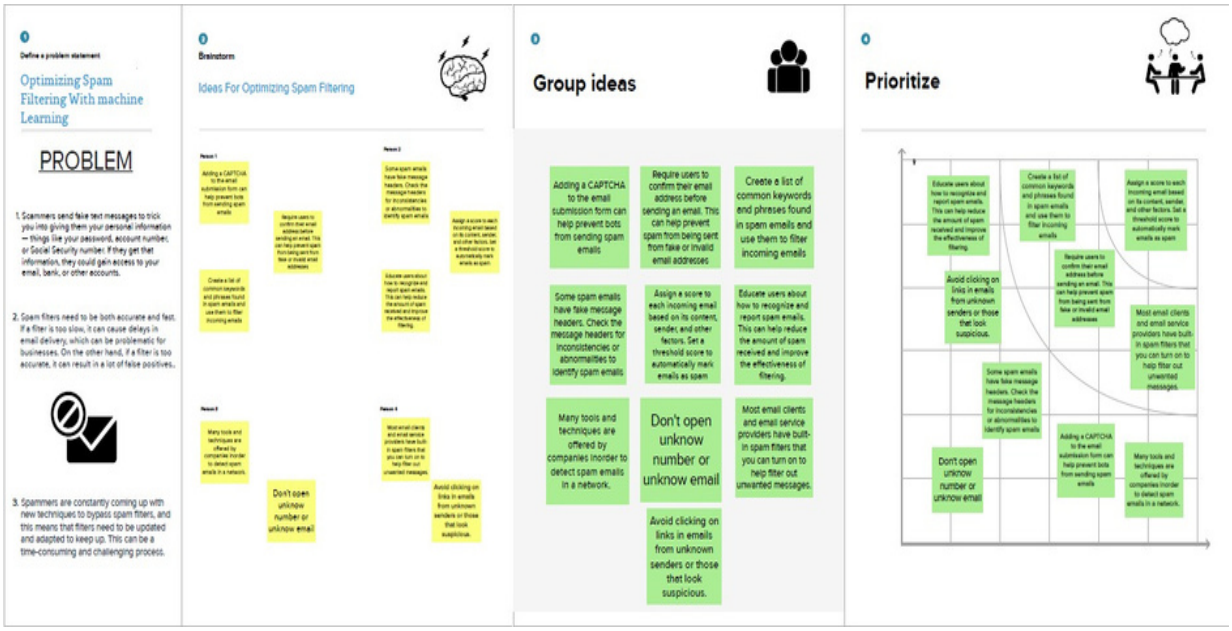
What are their fears, frustrations, and anxieties? What other feelings might influence their behavior?

They may feel anxious about missing important emails if they are mistakenly marked as spam.

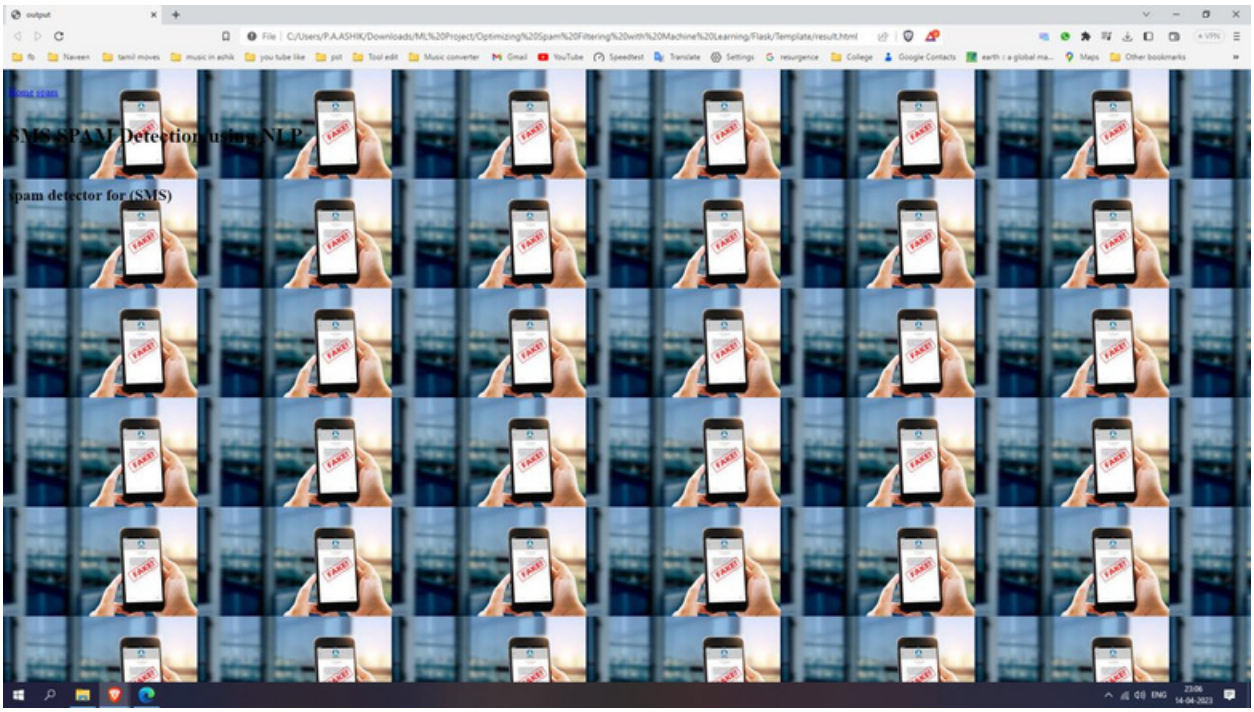
They may feel frustrated or annoyed by the presence of spam emails in their inbox.

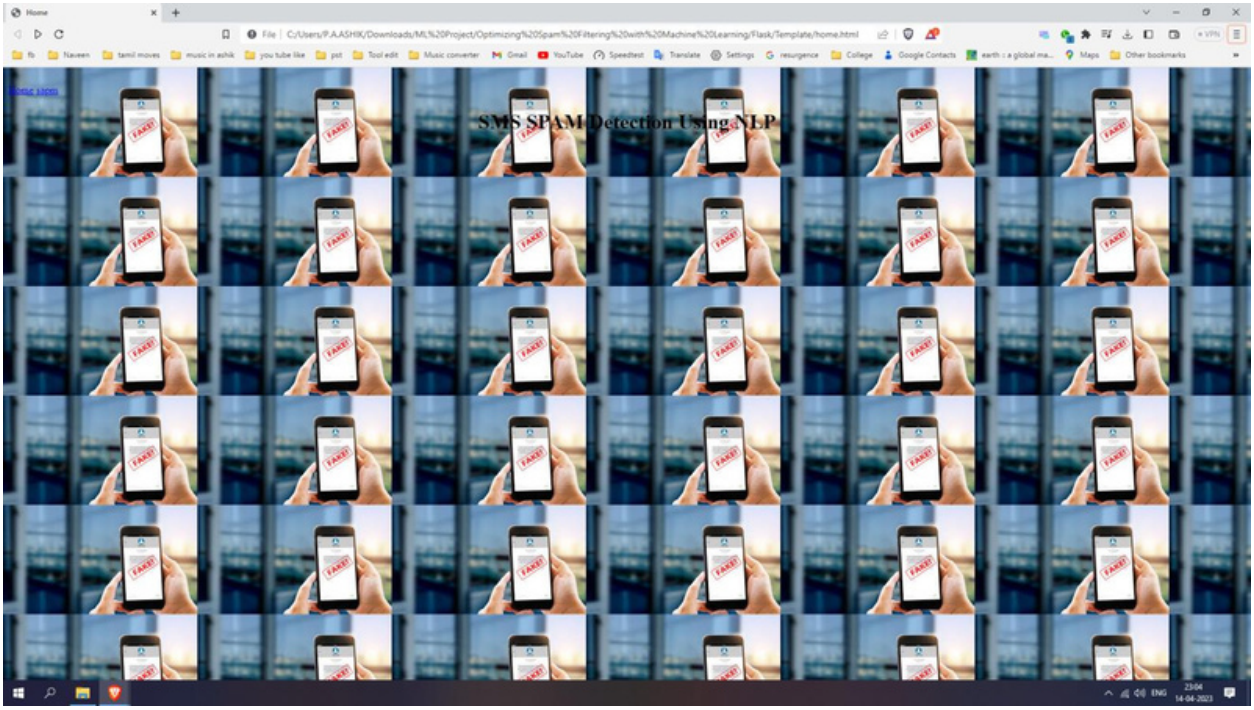
The user may feel overwhelmed by the sheer volume of emails they receive each day.

2.2 Ideation & Brainstorming Map:



3. RESULT


output
Home page
SMS SPAM Detection using NLP
spam detector for (SMS)


Home
Home page
SMS SPAM Detection Using NLP

4. ADVANTAGES AND DISADVANTAGES

ADVANTAGES :

1. Protection from spam: Spam filters help to protect users from unwanted, unsolicited, and potentially harmful messages that could contain malware, phishing scams, or other types of online fraud.
2. Improved productivity: By automatically filtering out spam messages, users can save time and focus on more important tasks, which can improve productivity and reduce stress.
3. Enhanced privacy: Spam filters can help to protect user privacy by blocking unwanted messages that could contain personal or sensitive information.
4. Customization: Spam filters can be customized to suit individual preferences, allowing users to set their own thresholds for spam filtering, and specify which messages should be blocked or allowed.

DISADVANTAGES :

1. False positives: Spam filters can sometimes incorrectly classify legitimate messages as spam, which can result in important emails being missed or deleted.

2. False negatives: Similarly, spam filters can also fail to identify some spam messages, which can result in unwanted messages reaching the user's inbox.

3. Over-reliance: Some users may become over-reliant on spam filters, and fail to exercise appropriate caution when receiving and opening emails from unknown senders.

4. Resource-intensive: Spam filters can be resource-intensive, particularly for organizations that receive large volumes of email traffic, which can impact network performance and increase costs.

Overall, the advantages of spam filters in protecting users from online threats and improving productivity typically outweigh the disadvantages, but users should be aware of their limitations and exercise appropriate caution when using email or messaging platforms.

5. APPLICATIONS

Spam filters are commonly used in email and messaging applications to automatically filter and categorize incoming messages based on their content and other features. Here are some applications of spam filters:

1. Email clients: Email clients, such as Microsoft Outlook and Gmail, use spam filters to automatically move incoming messages to the Spam folder or Junk Mail folder, where they can be reviewed or deleted by the user.

2. Messaging apps: Messaging apps, such as WhatsApp and Telegram, use spam filters to block unsolicited messages and prevent spam from reaching the user's inbox.

3. Social media: Social media platforms, such as Facebook and Twitter, use spam filters to automatically detect and remove spammy content, such as fake news, phishing scams, and other types of online fraud.

4. Web applications: Spam filters can be integrated into web applications, such as online forums and e-commerce platforms, to block spam messages and prevent spammers from posting unwanted content.

5. Network security: Spam filters can be used in network security applications, such as firewalls and intrusion detection systems, to

prevent spam and other types of online threats from entering the network.

Overall, spam filters are an essential tool for protecting users from online threats and improving the efficiency of communication platforms. By automatically filtering out unwanted messages, spam filters can help users to stay safe and productive online.

6. CONCLUSION

In conclusion, spam filters are a crucial tool for protecting users from unwanted, unsolicited, and potentially harmful messages in email and messaging applications. They help to prevent users from falling victim to phishing scams, malware, and other types of online fraud, while also improving productivity by reducing the amount of time users spend
sorting through their inbox.

Spam filters use a range of techniques, including content filtering, sender reputation analysis, and machine learning algorithms, to analyze incoming messages and determine whether they are likely to be spam. While spam filters can sometimes result in false positives or false negatives, the benefits of using spam filters typically outweigh the disadvantages, and users can customize their filters to suit their individual preferences.
Overall, spam filters are an essential tool for staying safe and productive

online, and their continued development and improvement will be critical in the fight against online threats and cybercrime.

7. FUTURE SCOPE

The future scope of spam filters is promising, with ongoing research and development in the field of machine learning, artificial intelligence, and natural language processing. Here are some potential future advancements in spam filtering:

1. Improved accuracy: Machine learning algorithms are becoming increasingly sophisticated in their ability to detect spam messages, and are likely to become even more accurate in the future. This will reduce the number of false positives and false negatives, and improve the overall effectiveness of spam filters.

2. Better understanding of context: Advances in natural language processing will enable spam filters to better understand the context of messages, and distinguish between legitimate messages and spam messages that may contain similar language. This will improve the accuracy of spam detection and reduce the likelihood of legitimate messages being incorrectly identified as spam.

Overall, the future of spam filters is bright, with ongoing research and development in the field of machine learning, natural language processing, and artificial intelligence. As online threats continue to evolve and become more sophisticated, spam filters will play an increasingly important role in protecting users from harm and ensuring the security and efficiency of communication platforms.

8. APPENDIX

A. Source Code:

Milestone 2: Data Collection & Preparation

Importing the libraries:

```
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
```

Read the Dataset:

```
#Load our dataset
df = pd.read_csv("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

Handling missing values:

```
#Give concise summary of a DataFrame
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
```

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

```
v1              0
v2              0
Unnamed: 2    5522
Unnamed: 3    5560
Unnamed: 4    5566
dtype: int64
```

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

```
# bottom 5 rows of the dataframe
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 İ_ 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

Handling Categorical Values:

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

Handling Imbalance Data:

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

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 0)

##train size 80% and test size 20%
```

Given data is imbalanced one, we are balancing the data

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

# 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(random_state = 2)
X_train_res, y_train_res = sm.fit_resample(X_train, y_train.ravel())

print('After OverSampling, the shape of train_X: {}'.format(X_train_res.shape))
print('After OverSampling, the shape of train_y: {} \n'.format(y_train_res.shape))

print("After OverSampling, counts of label '1': {}".format(sum(y_train_res == 1)))
print("After OverSampling, counts of label '0': {}".format(sum(y_train_res == 0)))
```

```
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
```

Exploratory Data Analysis:

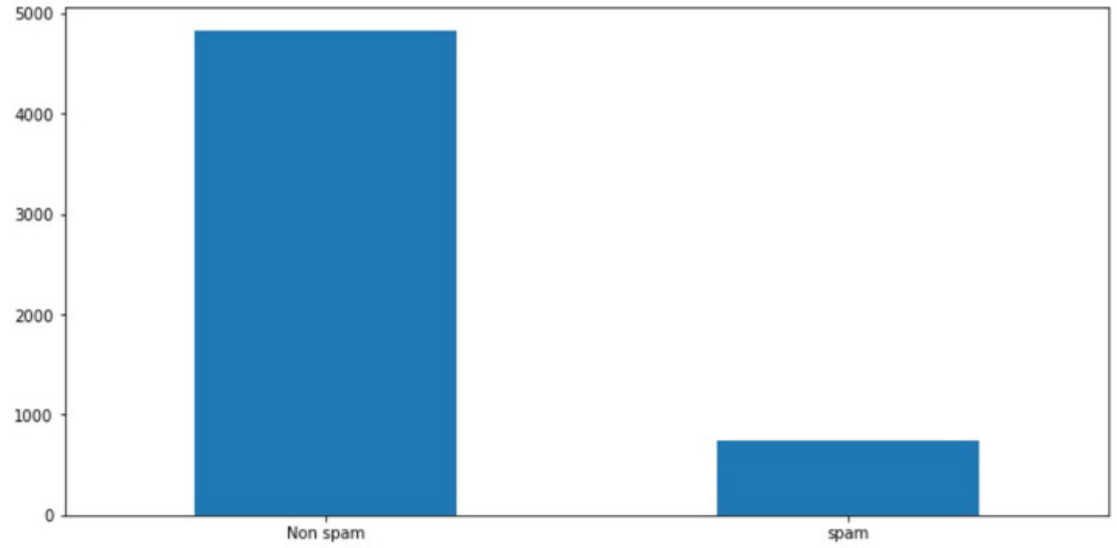
```
df.describe()
```

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
count	5572	5572	50	12	6
unique	2	5169	43	10	5
top	ham	Sorry, I'll call later	bt not his girlfrnd... G o o d n i g h t . . .@"	MK17 92H. 450Ppw 16"	GNT:-)"
freq	4825	30	3	2	2

```
df.shape
```

```
(5572, 5)
```

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



Scalling the Data:

```
# perfroming feature Scaling op[eration using standard scaller on X part of the dataset becau
# there different type of values in the columns
sc=StandardScaler()
x_bal=sc.fit_transform(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)

##train size 80% and test size 20%
```

Model Building:

1.1 Decision Tree Model:

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

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

1.2 Random Forest Model:

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

```
▾ RandomForestClassifier
RandomForestClassifier()
```

1.3 KNN model:

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

```
]: #Fitting the model to the training sets
   model.fit(X_train_res, y_train_res)
```

```
]: ▾ MultinomialNB
   MultinomialNB()
```

1.4 Xgboost model:

1.5 ANN model:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers 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)
```

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

```
Epoch 1/10
121/121 [==============================] - 26s 203ms/step - loss: 0.1404 - accuracy: 0.9565
Epoch 2/10
121/121 [==============================] - 24s 199ms/step - loss: 0.0220 - accuracy: 0.9950
Epoch 3/10
121/121 [==============================] - 23s 194ms/step - loss: 0.0155 - accuracy: 0.9965
Epoch 4/10
121/121 [==============================] - 23s 194ms/step - loss: 0.0163 - accuracy: 0.9962
Epoch 5/10
121/121 [==============================] - 24s 195ms/step - loss: 0.0158 - accuracy: 0.9965
Epoch 6/10
121/121 [==============================] - 23s 194ms/step - loss: 0.0132 - accuracy: 0.9974
Epoch 7/10
121/121 [==============================] - 23s 194ms/step - loss: 0.0130 - accuracy: 0.9972
Epoch 8/10
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:Your input ran out of da
ta; interrupting training. Make sure that your dataset or generator can generate at least `steps_per_epoch * epochs` batches (i
n this case, 1210 batches). You may need to use the repeat() function when building your dataset.
121/121 [==============================] - 23s 193ms/step - loss: 0.0128 - accuracy: 0.9972
```

Testing the model:

[45]:
```
y_pred=model.predict(X_test)
y_pred
```

```
35/35 [==============================] - 2s 29ms/step
```

[45]:
```
array([[1.5844109e-15],
       [4.4117199e-04],
       [1.1517070e-18],
       ...,
       [2.0661259e-08],
       [3.8018154e-17],
       [1.2099350e-12]], dtype=float32)
```

[46]:
```
y_pr = np.where(y_pred>0.5,1,0)
```

[49]:
```
y_test
```

[49]:
```
array([0, 0, 0, ..., 0, 0, 0])
```

[50]:
```
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
```

```
: 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...")))
```

```
Enter new review...hello fg hou hkl
[[0 0 0 ... 0 0 0]]
1/1 [==============================] - 0s 45ms/step
[[0.9784973]]
```

Perfomance Testing & Hyperparameter

Tuning: 1.1: 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:tens
```

```
]: 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
```

Model Deployment:

Saving our model

By comparing the all the model , we can come to a conclusion that ANN is the best model

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

Building Html pages:

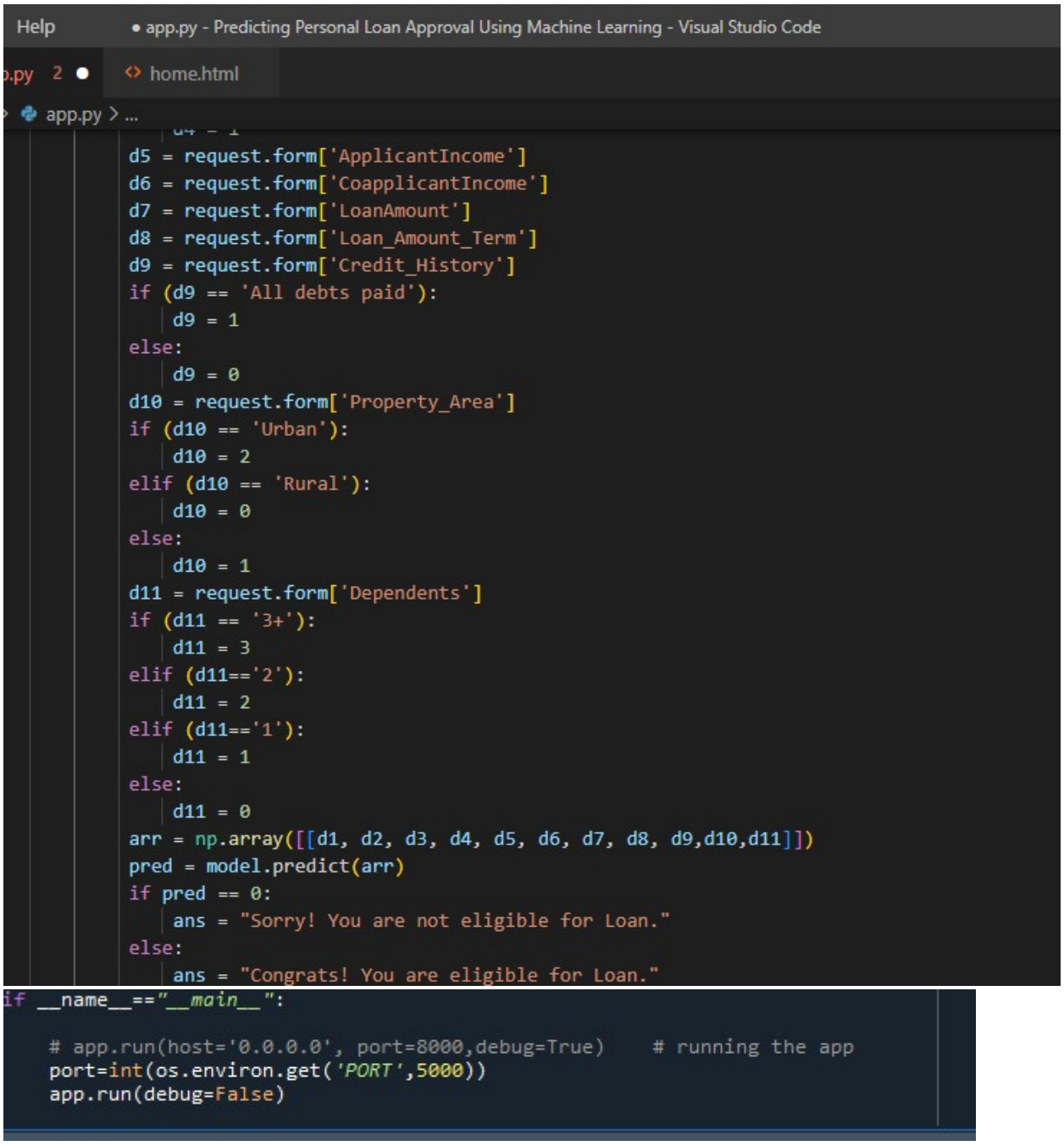
Build Python code:

```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
```

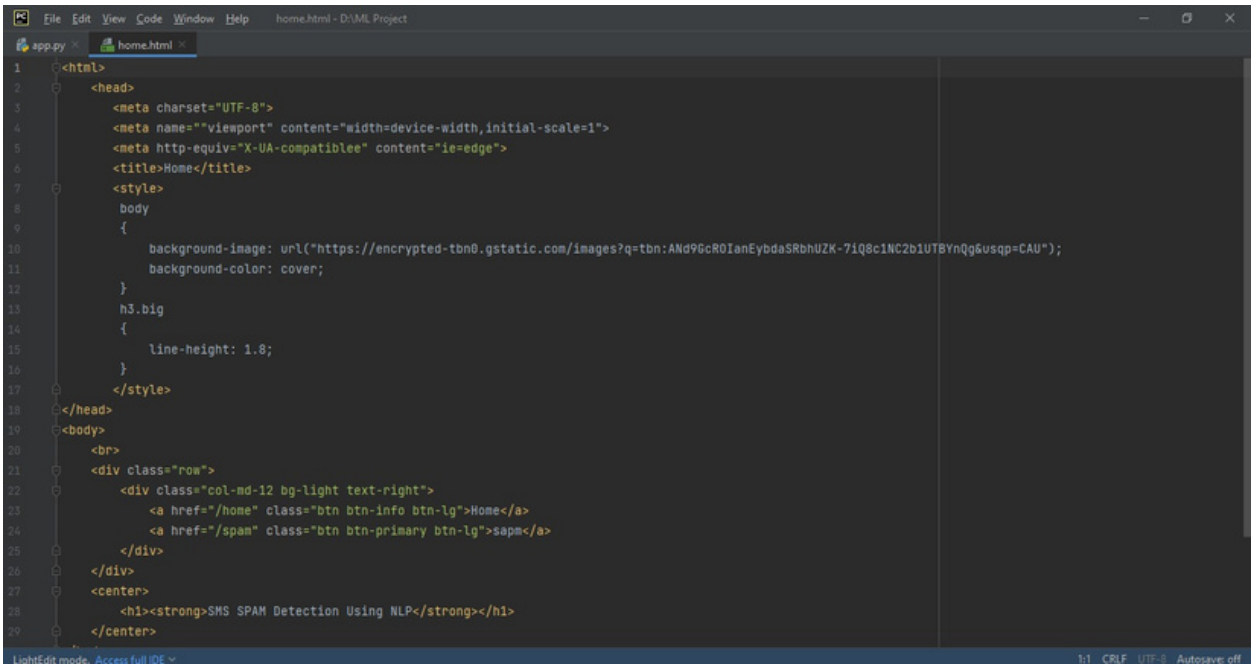
```python
@app.route('/Spam',methods=['POST','GET'])
def prediction(): # route which will take you to the prediction page
    return render_template('spam.html')

@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")
```


Help
• app.py - Predicting Personal Loan Approval Using Machine Learning - Visual Studio Code
.py 2 ●
home.html
app.py > ...
d5 = request.form['ApplicantIncome']
d6 = request.form['CoapplicantIncome']
d7 = request.form['LoanAmount']
d8 = request.form['Loan_Amount_Term']
d9 = request.form['Credit_History']
if (d9 == 'All debts paid'):
d9 = 1
else:
d9 = 0
d10 = request.form['Property_Area']
if (d10 == 'Urban'):
d10 = 2
elif (d10 == 'Rural'):
d10 = 0
else:
d10 = 1
d11 = request.form['Dependents']
if (d11 == '3+'):
d11 = 3
elif (d11=='2'):
d11 = 2
elif (d11=='1'):
d11 = 1
else:
d11 = 0
arr = np.array([[d1, d2, d3, d4, d5, d6, d7, d8, d9,d10,d11]])
pred = model.predict(arr)
if pred == 0:
ans = "Sorry! You are not eligible for Loan."
else:
ans = "Congrats! You are eligible for Loan."
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)

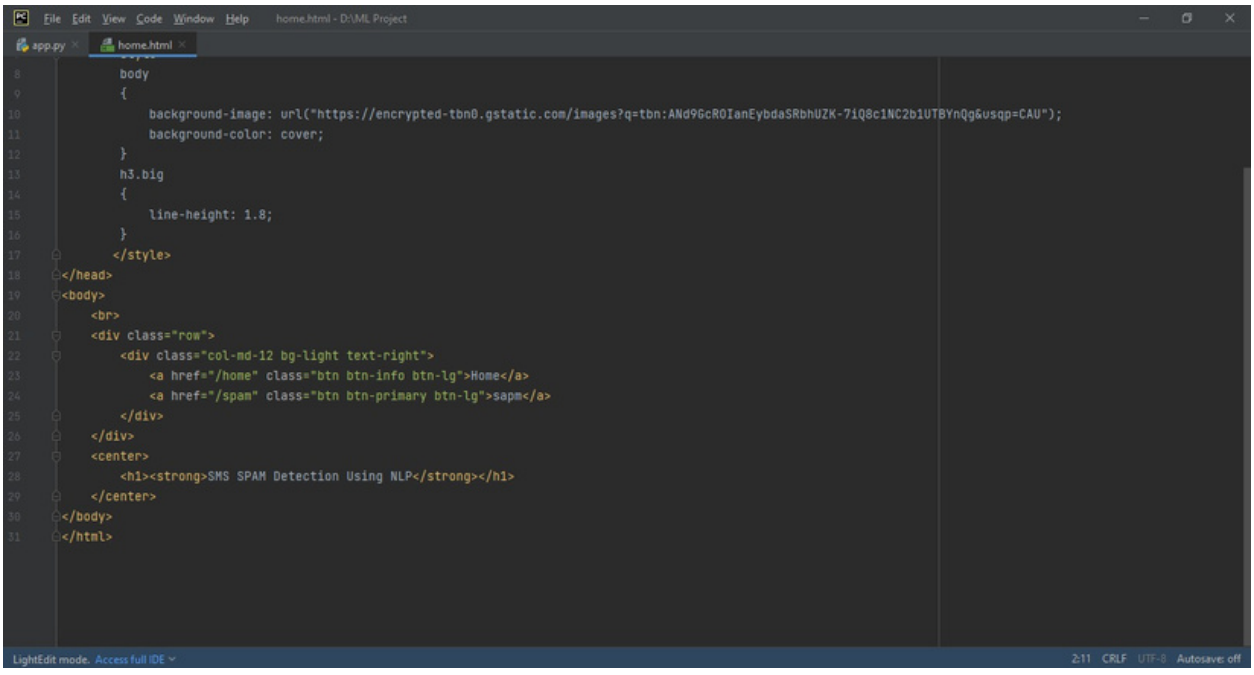
home.html:


File Edit View Code Window Help home.html - D:\ML Project
app.py
home.html
<html>
<head>
<meta charset="UTF-8">
<meta name=""viewport" content="width=device-width,initial-scale=1">
<meta http-equiv="X-UA-compatiblee" content="ie=edge">
<title>Home</title>
<style>
body
{
background-image: url("https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcROIanEybdaSRbhUZK-7iQ8c1NC2b1UTBYnQg&usqp=CAU");
background-color: cover;
}
h3.big
{
line-height: 1.8;
}
</style>
</head>
<body>

Home
sapm

<h1>SMS SPAM Detection Using NLP</h1>

LightEdit mode. Access full IDE
1:1 CRLF UTF-8 Autosave: off

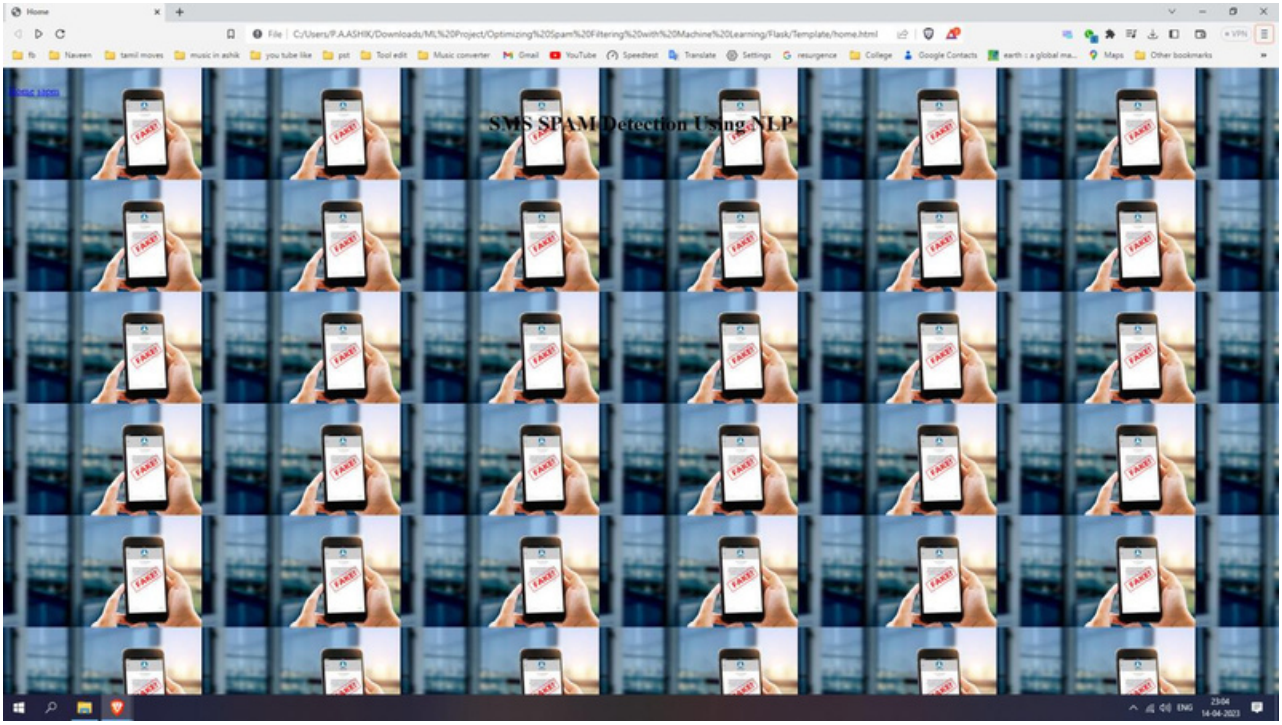

File Edit View Code Window Help home.html - D:\ML Project
app.py
home.html
body
{
background-image: url("https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcROIanEybdaSRbhUZK-7iQ8c1NC2b1UTBYnQg&usqp=CAU");
background-color: cover;
}
h3.big
{
line-height: 1.8;
}
</style>
</head>
<body>

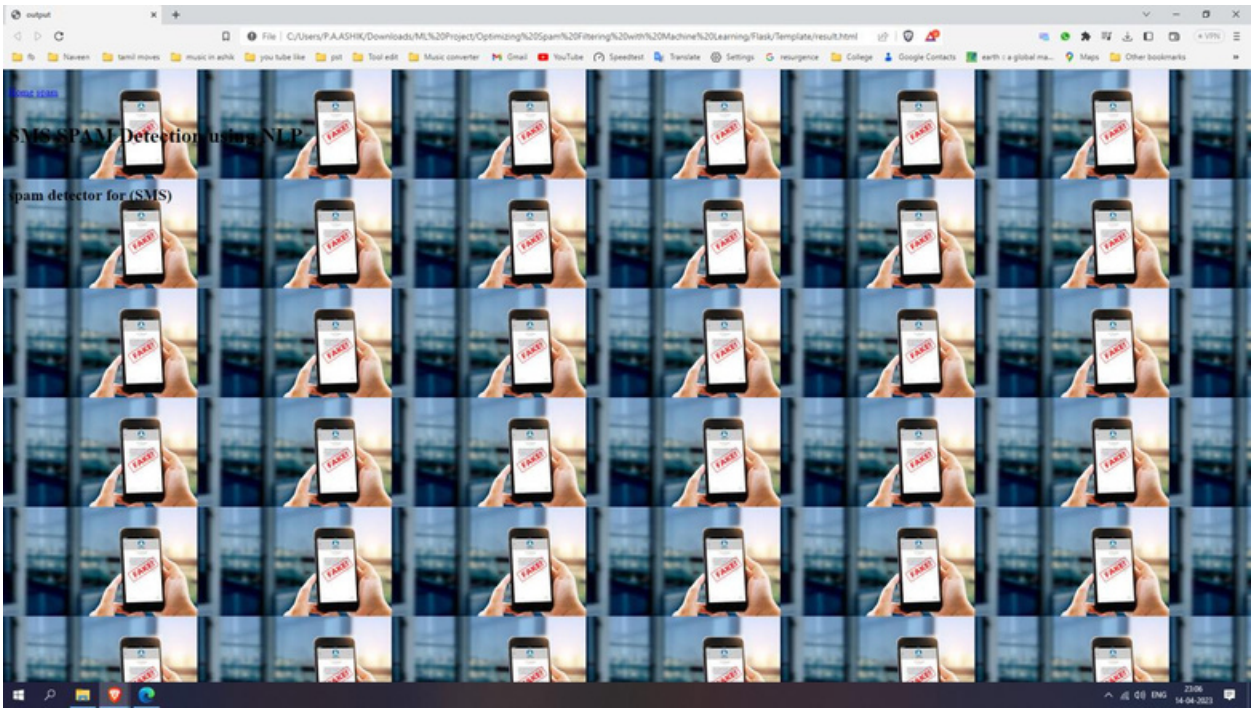
Home
sapm

<h1>SMS SPAM Detection Using NLP</h1>

</body>
</html>
LightEdit mode. Access full IDE
2:11 CRLF UTF-8 Autosave: off

Output:




output
File | C:/Users/P.AASHIK/Downloads/ML%20Project/Optimizing%20Spam%20Filtering%20with%20Machine%20Learning/Flask/Template/result.html
Home page
SMS SPAM Detection using NLP
spam detector for (SMS)