Optimizing Spam Filtering With Machine Learning

1 INTRODUCTION

1.1 Overview

In Machine Learning, A spam filter is a program used to detect unsolicited, unwanted and virus-infected emails and prevent those messages from a getting to a user's inbox. Like other types of filtering programs, spam filter looks for specific criteria on which to base its judgments.

Internet service providers (ISPs), free online email services and businesses use email spam filtering tools to minimize the risk of distributing spam.

For example, one of the simplest and earliest versions of spam filtering, like the one that was used by Microsoft's Hotmail, was set to watch out for particular words in the subject lines of messages. An email was excluded from the user's inbox whenever the filter recognized one of the specified words.

This method is not especially effective and often omits perfectly legitimate messages, called false positives, while letting actual spam messages through.

More sophisticated programs, such as Bayesian filters and other heuristic filters, identify spam messages by recognizing suspicious word patterns or word frequency. They do this by learning the user's preferences based on the emails marked as spam. The spam software then creates rules and applies them to future emails that target the user's inbox.

For example, whenever users mark emails from a specific sender as spam, the Bayesian filter recognizes the pattern and automatically moves future emails from that sender to the spam folder.

ISPs apply spam filters to both inbound and outbound emails. However, small to medium enterprises usually focus on inbound filters to protect their network. There are also many different spam filtering solutions available. They can be hosted in the cloud, hosted on servers or integrated into email software, such as Microsoft Outlook.

In machine learning, spam filtering protocols use instance-based or memory-based learning methods to identify and classify incoming spam emails based on their resemblance to stored training examples of spam emails.

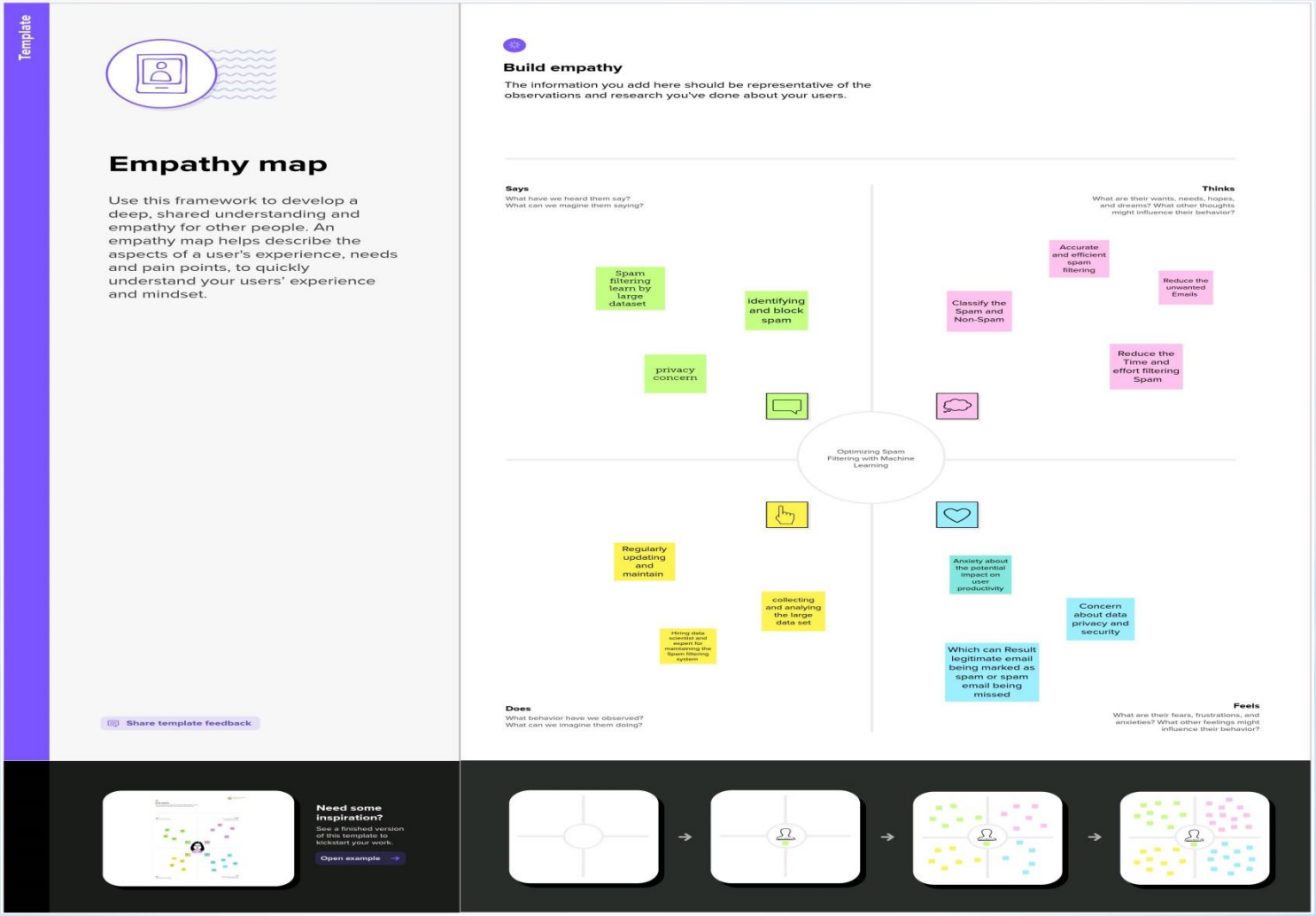
The upsurge in the volume of unwanted emails called spam has created an intense need for the development of more dependable and robust antispam filters. Machine learning methods of recent are being used to successfully detect and filter spam emails. We present a systematic review of some of the popular machine learning based email spam filtering approaches.

1.2 Purpose

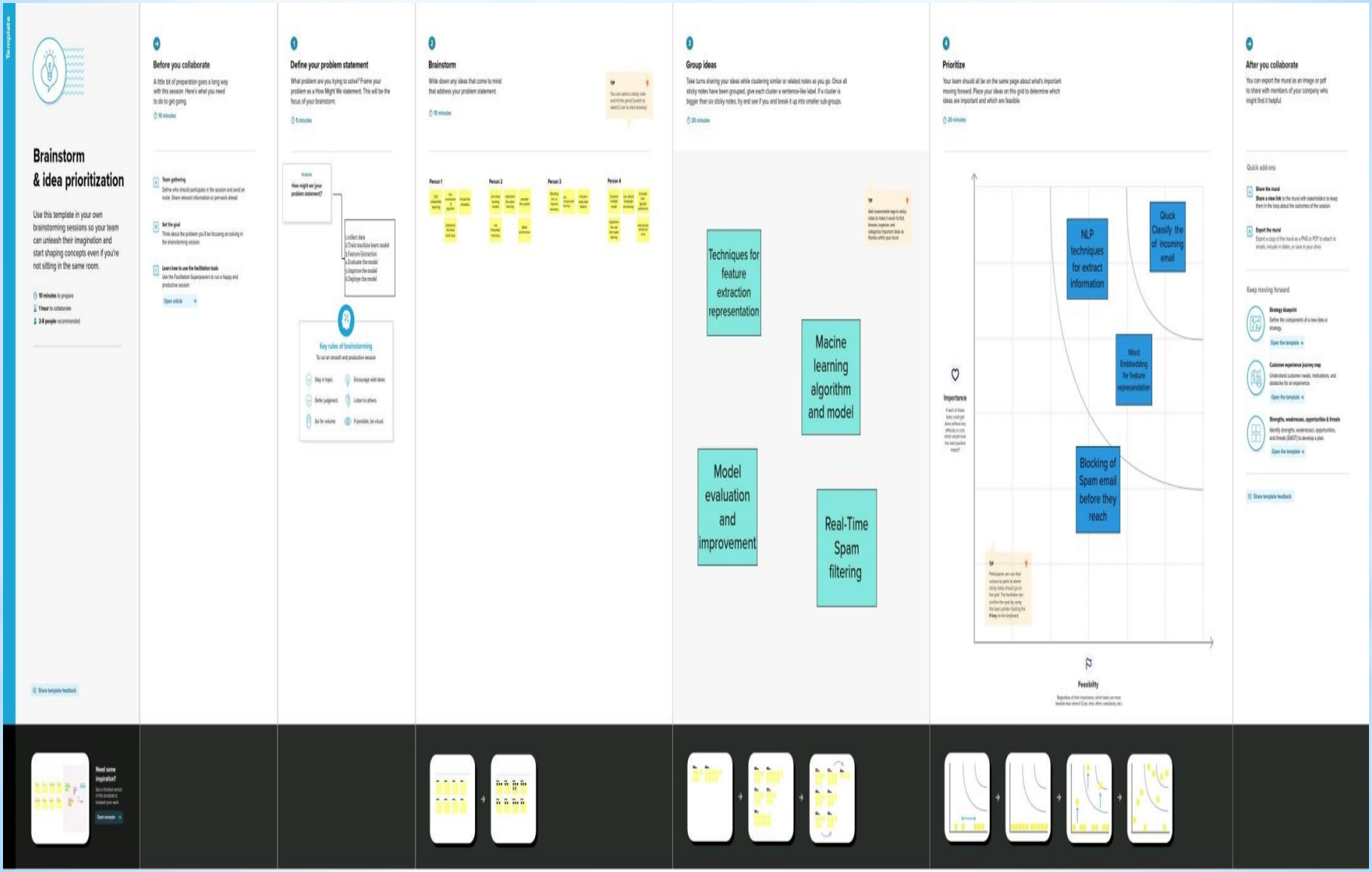
An email spam filter is a necessity for every individual and organization operating emailing activities on a regular base. An average person roughly receives 100-120 emails a day, out of which an average of 80% of emails are spam. At its very root, keeping your communications flow smooth requires a reliable email spam filter.

2 PROBLEM DEFINITION & DESIGN THINKING

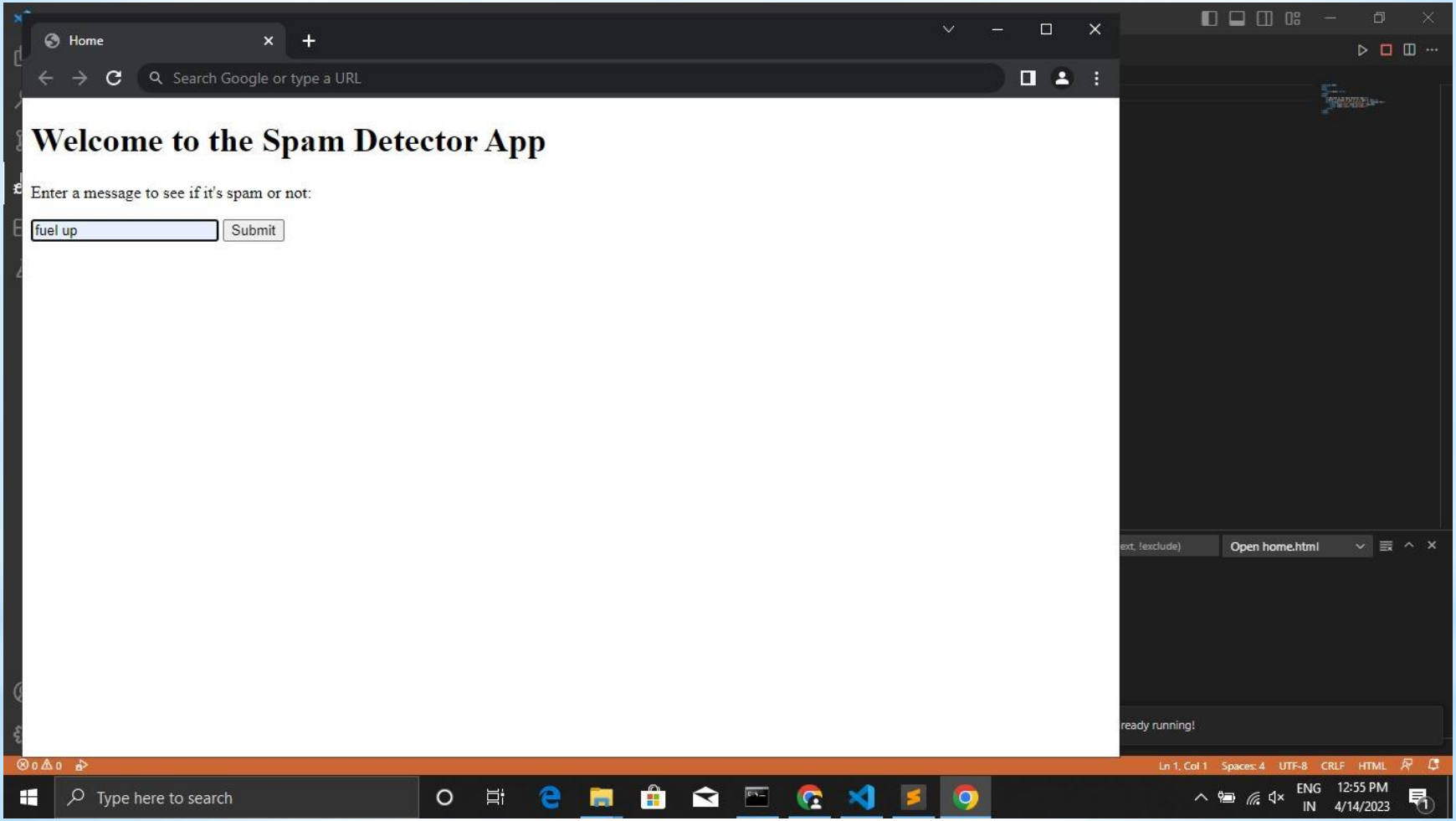
2.1 Empathy Map

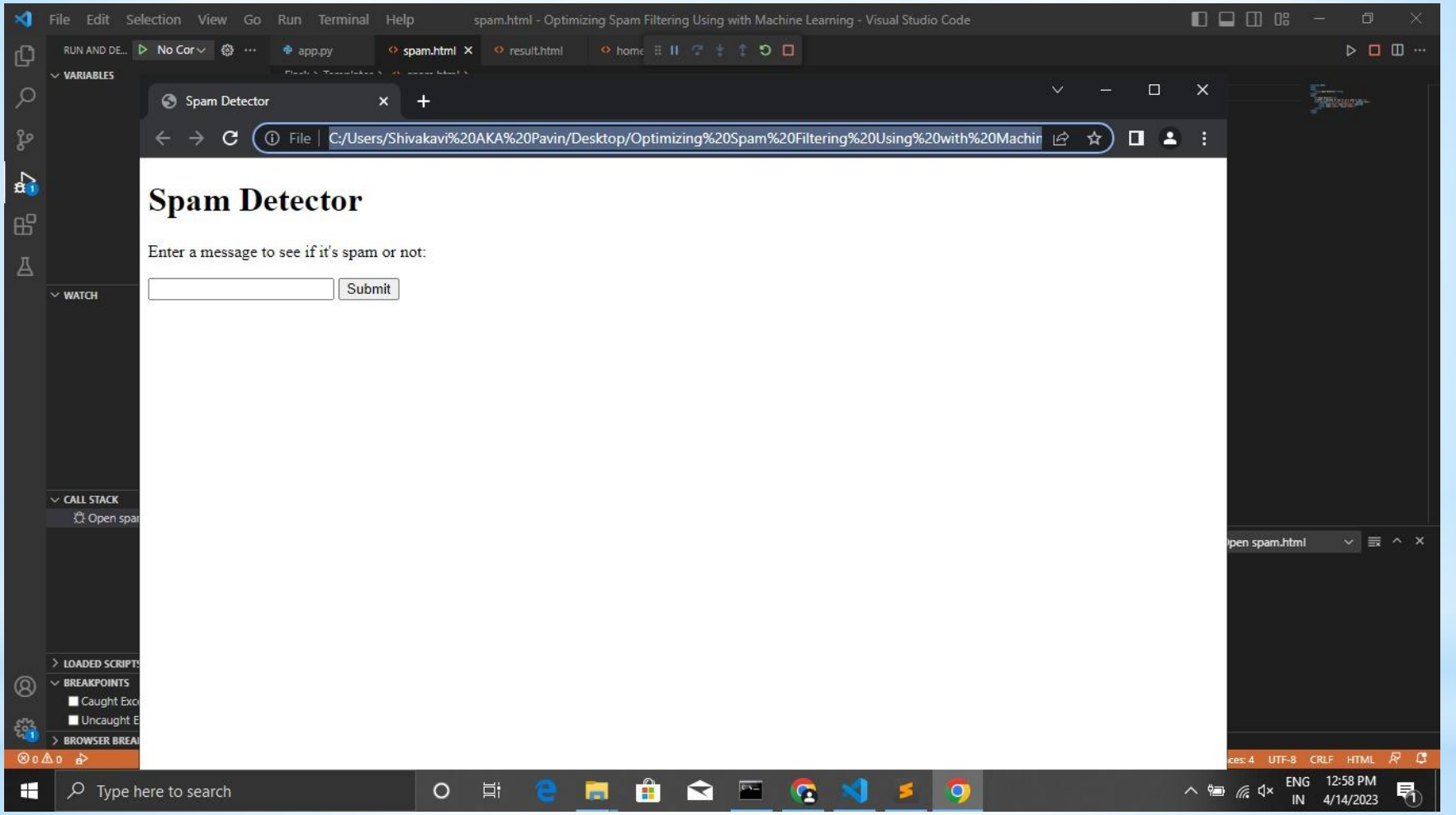


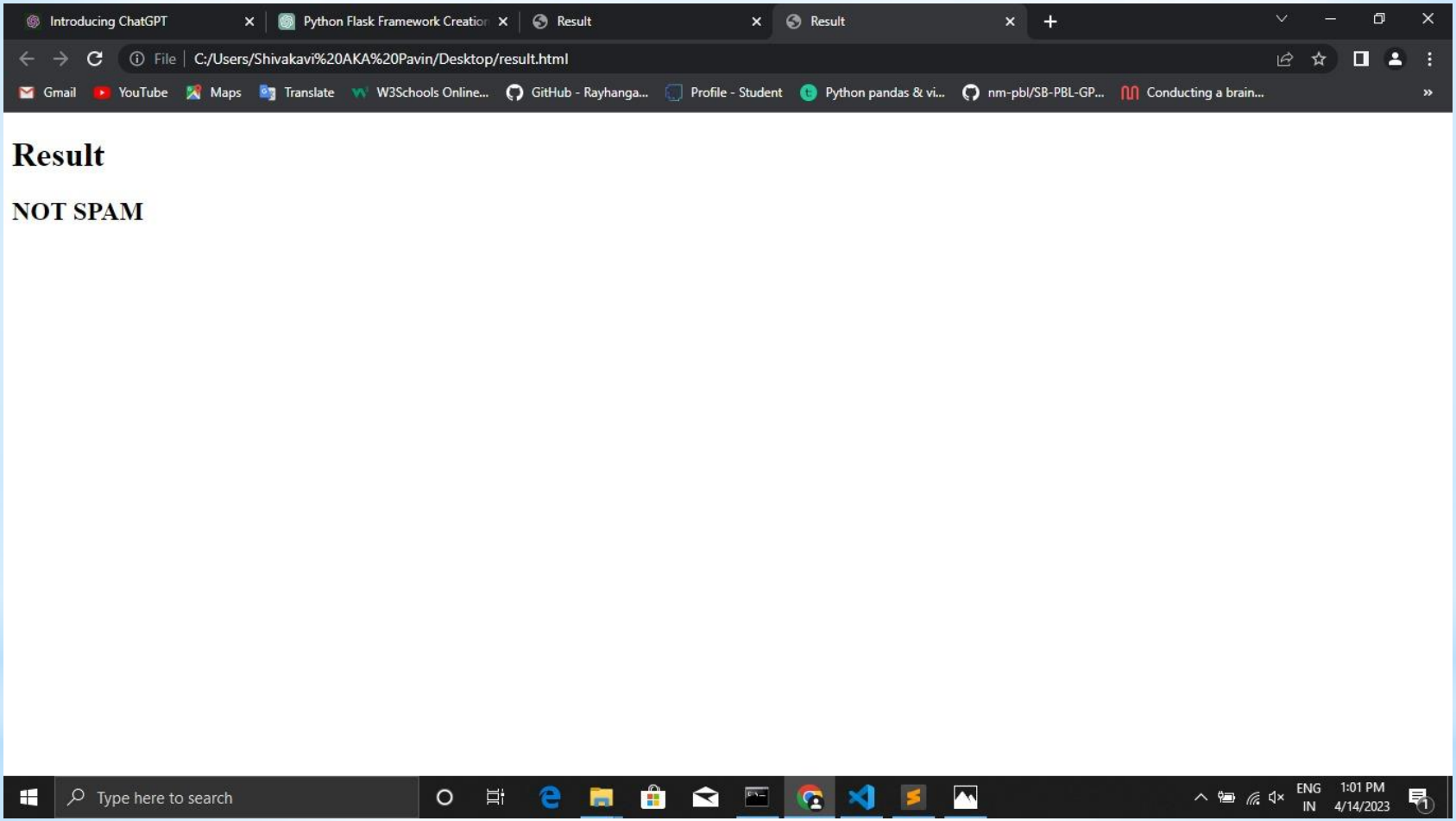
2.2 Ideation & Brainstorming Map



3 RESULT


Home
Search Google or type a URL
Welcome to the Spam Detector App
Enter a message to see if it's spam or not:
fuel up
Submit
Open home.html
ready running!
Ln 1, Col 1
Spaces: 4
UTF-8
CRLF
HTML
Type here to search
ENG
IN
12:55 PM
4/14/2023


File Edit Selection View Go Run Terminal Help
spam.html - Optimizing Spam Filtering Using with Machine Learning - Visual Studio Code
app.py
spam.html
result.html
VARIABLES
WATCH
CALL STACK
LOADED SCRIPTS
BREAKPOINTS
Spam Detector
File | C:/Users/Shivakavi%20AKA%20Pavin/Desktop/Optimizing%20Spam%20Filtering%20Using%20with%20Machin
Spam Detector
Enter a message to see if it's spam or not:
Submit
Type here to search
ENG
IN
12:58 PM
4/14/2023


Introducing ChatGPT
Python Flask Framework Creation
Result
Result
File | C:/Users/Shivakavi%20AKA%20Pavin/Desktop/result.html
Gmail
YouTube
Maps
Translate
W3Schools Online...
GitHub - Rayhanga...
Profile - Student
Python pandas & vi...
nm-pbl/SB-PBL-GP...
Conducting a brain...
Result
NOT SPAM
Type here to search
ENG
IN
1:01 PM
4/14/2023

4 ADVANTAGES & DISADVANTAGES

Advantages

Security :

Out of all the emails received by an individual throughout the day, the possibility of a phishing attack or cyber threat is never zero. With the benefits of email spam filters, the security risk can be reduced since the user gets in hand the emails that have gone through various spam checks. Moreover, these email spam filters throw out malware, malicious, and virus-infested emails and protect user security.

Time-Saving :

Let us go back to the emailing stats we discussed at the beginning of this section. Having to filter out the 20% important emails out of the average 80% clutter does seem time-consuming.

This can be of greater concern if these stats are put into an organization's emailing communications. By filtering out the important emails and sending to the spam box the junk emails, an email spam filter saves time for the user and keeps the business communications going by streamlining the user inbox.

Increased Productivity:

Along the lines of the time-saving benefit of email spam filters, these tools facilitate increased productivity of the user by keeping away unwanted emails. As mentioned in the types of email spam filters, in certain cases the users can set up standards for email spam division. By keeping away the emails that might distract or waste the time of the employees, these email spam filters can keep the inbox of employees clean and facilitate increased productivity.

Disadvantage

The biggest disadvantage of using an email filter is that you may end up with messages being identified as being spam through a mistake of the algorithm that is used. According to Steven Scott Bayesian specialist, even with the very best spam filters on the market you can still end up with messages being improperly labeled.

While missing out on important emails is a nuisance, we need to think about the fact that you can also miss the same emails if you receive a lot of spam. How can you see that message from the boss if there are hundreds of emails sent every single day? You can be highly attentive and still miss out on some emails.

5 APPLICATIONS

Spam to a private email can cause havoc throughout the system. Nowadays, it has created many problems in business life, such as occupying network bandwidth and the space in users' mailboxes. Research has been conducted in this area to resolve this issue and spam detection systems (SDS) have been developed to monitor spammers and filter email activities by identifying patterns in email messages, thus improving the tool to detect spam.

Both the knowledge filtering and the guideline filtering strategies are used to detect spam. Both have advantages and disadvantages, but neither is effective against all threats . The guideline detection method works well for identifying recognised communications but not spam. In comparison,

the knowledge detection strategy is effective at finding new messages, but it has a low detection rate and a high percentage of false positives. As such, our study introduces a new method. Most investigations into spam detection in the literature have focused on the knowledge detection strategy since it seemed more promising.

Spam filters are applied to both inbound email (email entering the network) and outbound email (email leaving the network). ISPs use both methods to protect their customers. SMBs typically focus on inbound filters.

There are many spam filtering solutions available. They can be hosted in the "cloud," on computer servers, or integrated into email software such as Microsoft Outlook.

6 CONCLUSION

To review the results of the hypothesis it can be said, that the design of a Meta spam filter make sense as well as has its ground. Although the notion deals with existing spam filters as well as e-mail corpus, the over describe methodology can as well be applied for extra filters also. Studies of Bayesian networks have provided a fine base for the creation of a Meta spam filter.

7 FUTURE SCOPE

This work proposes a model for improving recognition of cruel spam in email. Our model resolve employ a novel dataset intended for the process of feature choice, and then validate the set of chosen features using three classifiers identified in spam detection: Support Vector Machine, Naïve Bayes, and Multilayer Perception. Feature selection is projected to recover training time as well as accuracy for the classifiers.

8 APPENDIX

A. Source Code

2 Data Collection & Preparation

2.1 Collect The Dataset

2.2 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
```

2.3 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

2.4 Data Preparation

2.5 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
```

Rename The Column

```
: 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 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

2.6 Handling Categorical Values

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

2.7 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
```

2.8 Cleaning The Text Data

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

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\smart\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
True
```

```
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)
```

```
[18]: corpus
```

```
[18]: ['go jurong point crazi avail bugi n great world la e buffet cine got amor wat',
 'ok lar joke wif u oni',
 'free entri 2 wkli comp win fa cup final tkt 21st may 2005 text fa 87121 receiv entri question std txt rate c appli 08452810
075over18',
 'u dun say earli hor u c alreadi say',
 'nah think goe usf live around though',
 'freemsg hey darl 3 week word back like fun still tb ok xxx std chg send 1 50 rcv',
 'even brother like speak treat like aid patent',
 'per request mell mell oru minnaminungint nurungu vettam set callertun caller press 9 copi friend callertun',
 'winner valu network custom select receivea 900 prize reward claim call 09061701461 claim code kl341 valid 12 hour',
 'mobil 11 month u r entitl updat latest colour mobil camera free call mobil updat co free 08002986030',
 'gonna home soon want talk stuff anymor tonight k cri enough today',
 'six chanc win cash 100 20 000 pound txt csh11 send 87575 cost 150p day 6day 16 tsandc appli repli hl 4 info',
 'urgent 1 week free membership 100 000 prize jackpot txt word claim 81010 c www dbuk net lccltd pobox 4403ldnw1a7rw18',
 'search right word thank breather promis wont take help grant fulfil promis wonder bless time',
 'date sunday',
 'xxxmobilemovieclub use credit click wap link next txt messag click http wap xxxmobilemovieclub com n qjkgighjjgcbl',
 'oh k watch',
 'eh u rememb 2 spell name ye v naughti make v wet',
```

```
[19]: 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.pkl', 'wb')) ## saving to into cv.pkl file
```

3 Exploratory Data Analysis

3.1 Descriptive Statistical

```
df.describe()
```

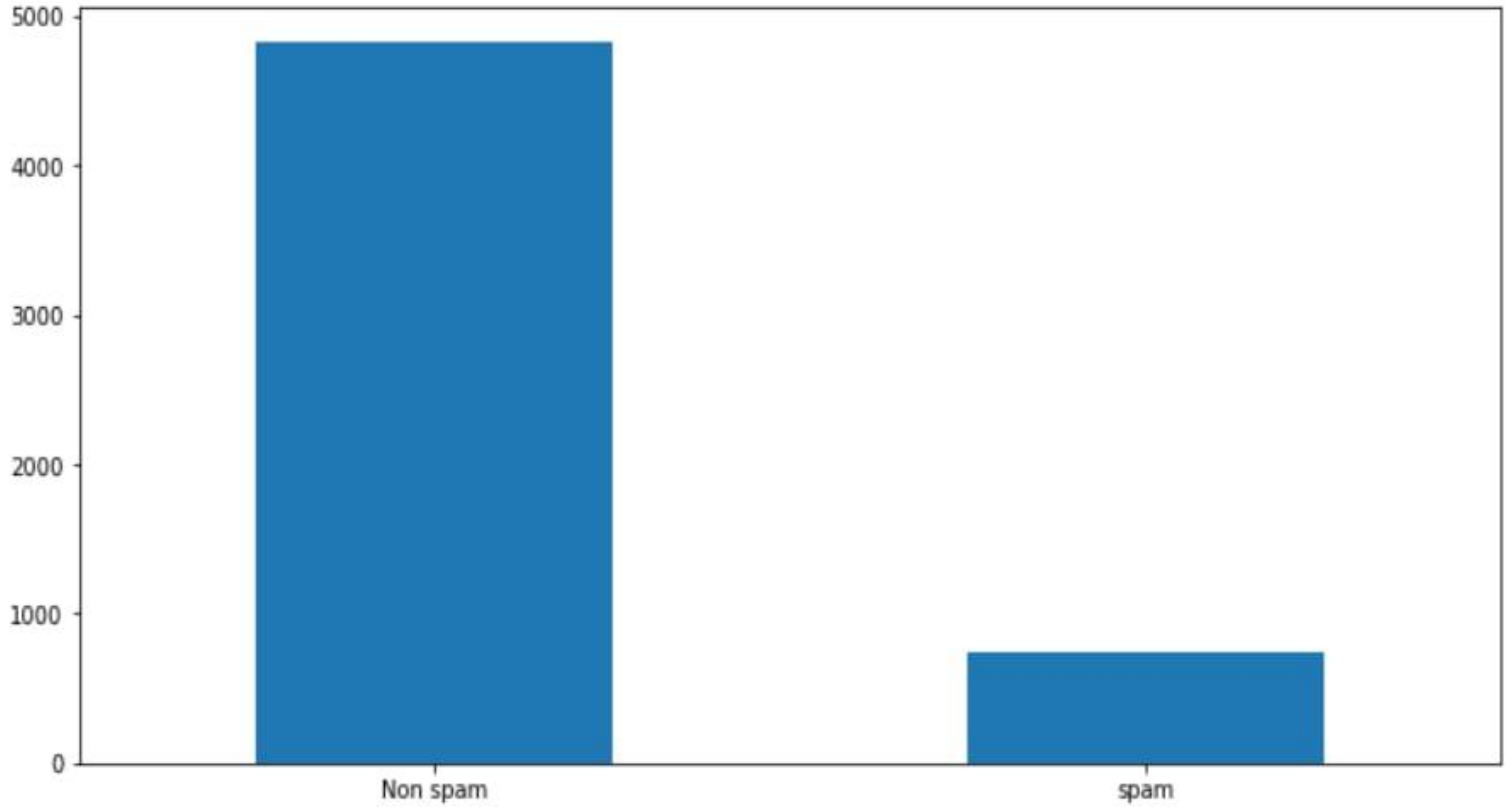
	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
count	5572	5572	50	12	6
unique	2	5169	43	10	5
top	ham	Sorry, I'll call later	bt not his girlfrnd... G o o d n i g h t . . .@"	MK17 92H. 450Ppw 16"	GNT:-)"
freq	4825	30	3	2	2

```
df.shape
```

```
(5572, 5)
```

3.2 Visual Analysis

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



4 Model Building

4.1 Training The Model In Multiple Algorithms

```
#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%
```

Splitting data into train and test

4.2 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
```

5 Performance Testing & Hyperparameter Tuning

5.1 Testing Model With Multiple Evaluation Metrics

5.2 Compare The Model

5.3 Comparing Model Accuracy Before & After Applying Hyperparameter Tuning

```
]: 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
```

6 Model Deployment

6.1 Save The Best Model

6.2 Integrate With Web Framework

6.3 Building Html Pages

6.4 Build Python Code

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

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

```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")
```

```
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)
```

6.5 Run The Web Application