PROJECT REPORT CONTENT

|  |  |  |
| --- | --- | --- |
| **CHAPTER** | **TITLE** | **PAGE.NO.** |
| 1 | INTRODUCTION   * 1. OVERVIEW   2. PURPOSE | 2 |
| 2 | PROBLEM DEFINITION & DESIGN THINKING   * 1. EMPATHY MAP   2. IDEATION & BRAINSTOMING MAP | 3  4 |
| 3 | RESULT | 5 |
| 4 | ADVANTAGES & DISADVANTAGES | 9 |
| 5 | APPLICATIONS | 10 |
| 6 | CONCLUSION | 11 |
| 7 | FUTURE SCOP | 12 |
| 8 | APPENDIX  8.1 SOURCE CODE | 13 |

**OPTIMIZING SPAM FILTERING WITN MACHINE LEARNING**

**1.INDRODUCTION**

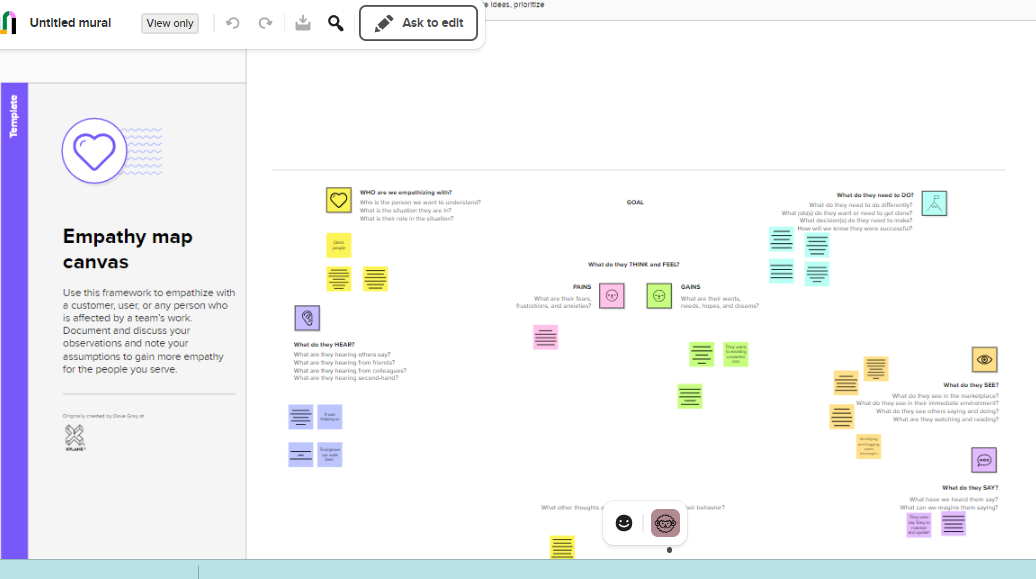
**OverView:**

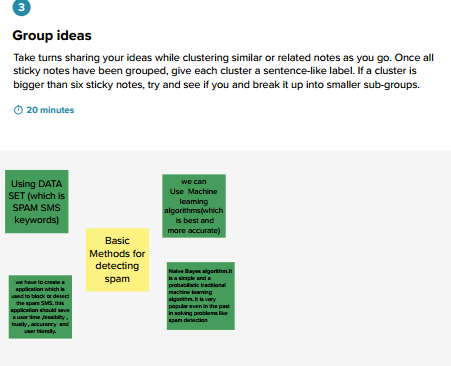
The problem of spam messages, particularly SMS spam, has been an ongoing issue for both individuals and organizations. To combat this problem, machine learning algorithms have been utilized to optimize spam filtering. In this work, we discuss the effectiveness of machine learning techniques in identifying spam messages in SMS data. We review data preprocessing techniques, such as decision trees, naive Bayes, and support vector machines, and evaluate their performance in terms of accuracy, precision, and recall. We also discuss the importance of feature selection and various approaches in improving the performance of the spam filter. Finally, we provide recommendations on how to implement an optimized spam filtering system using machine learning. Overall, this work aims to provide insights into the use of machine learning for spam filtering and how it can be leveraged to improve the efficiency and accuracy of spam detection in SMS messages using URL.

**Purpose :**

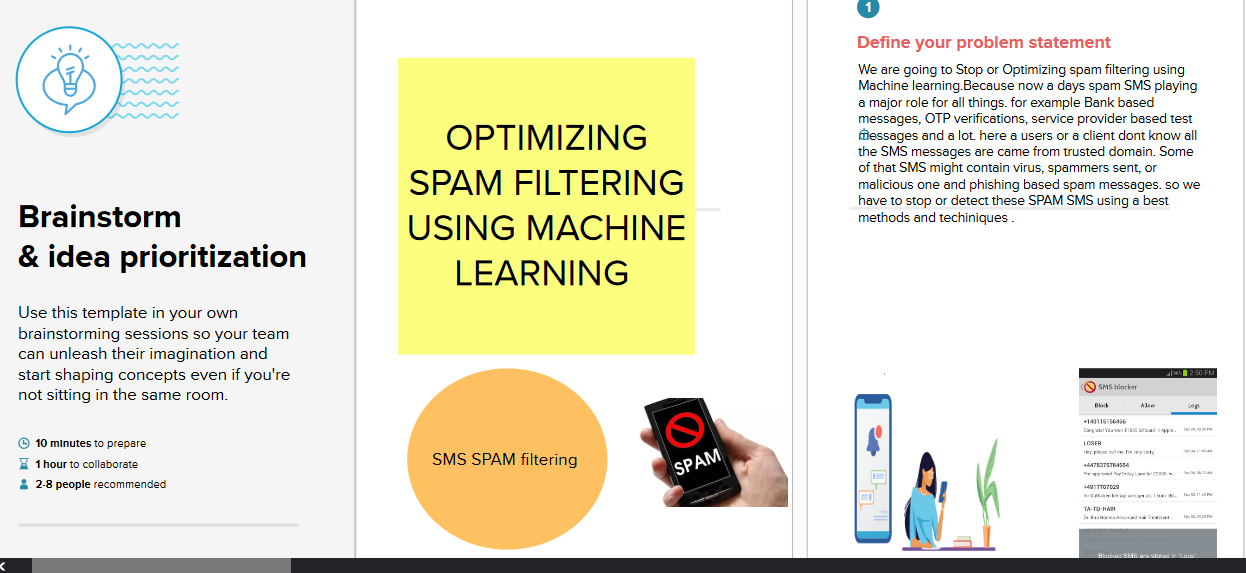
* Optimizing spam filtering, using our Project or This model enables the people to avoid viewing the unwanted SMS messages, malicious links , frauds and the filtered messages will be a vital one.By filtering the Spam SMS, Essential SMS and Detail alone are viewed which we promote the service provider business aspect using android application.

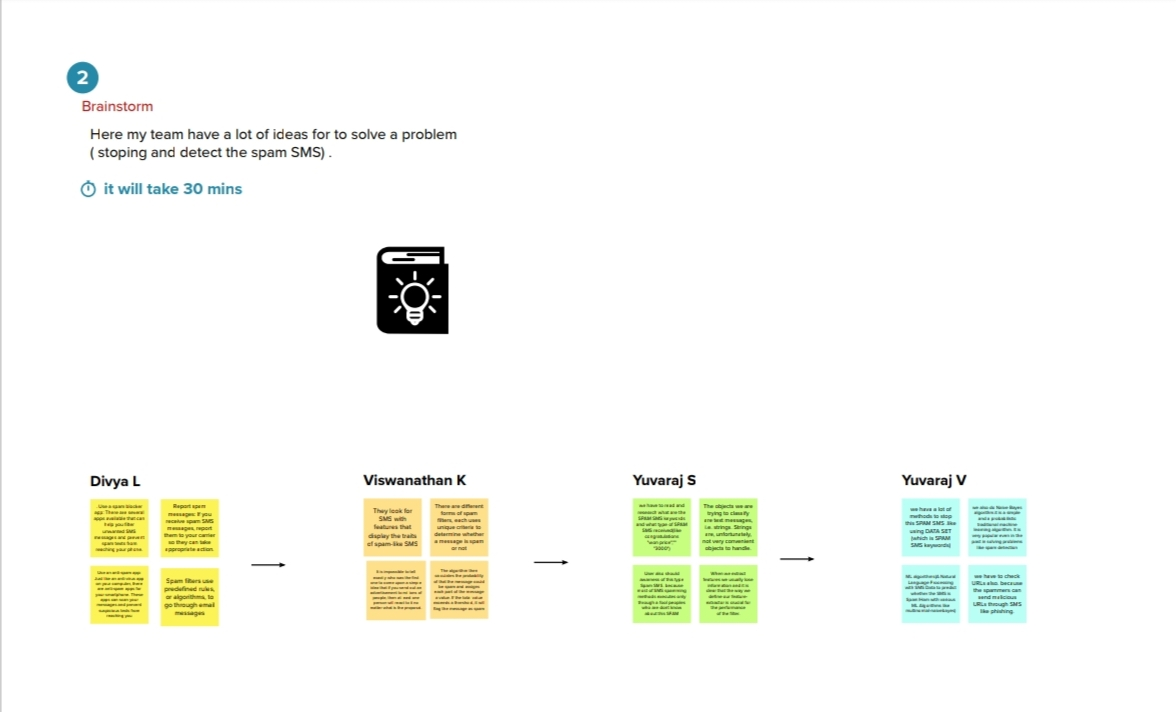
**Empathy Map:**

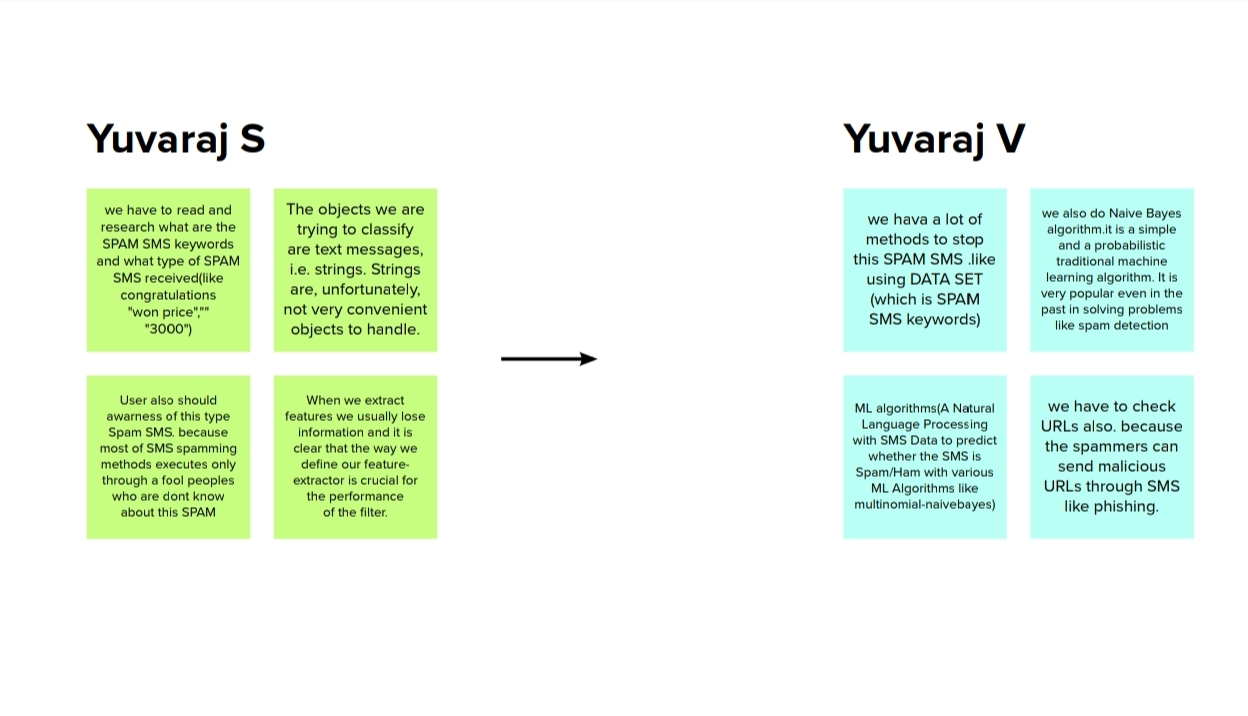


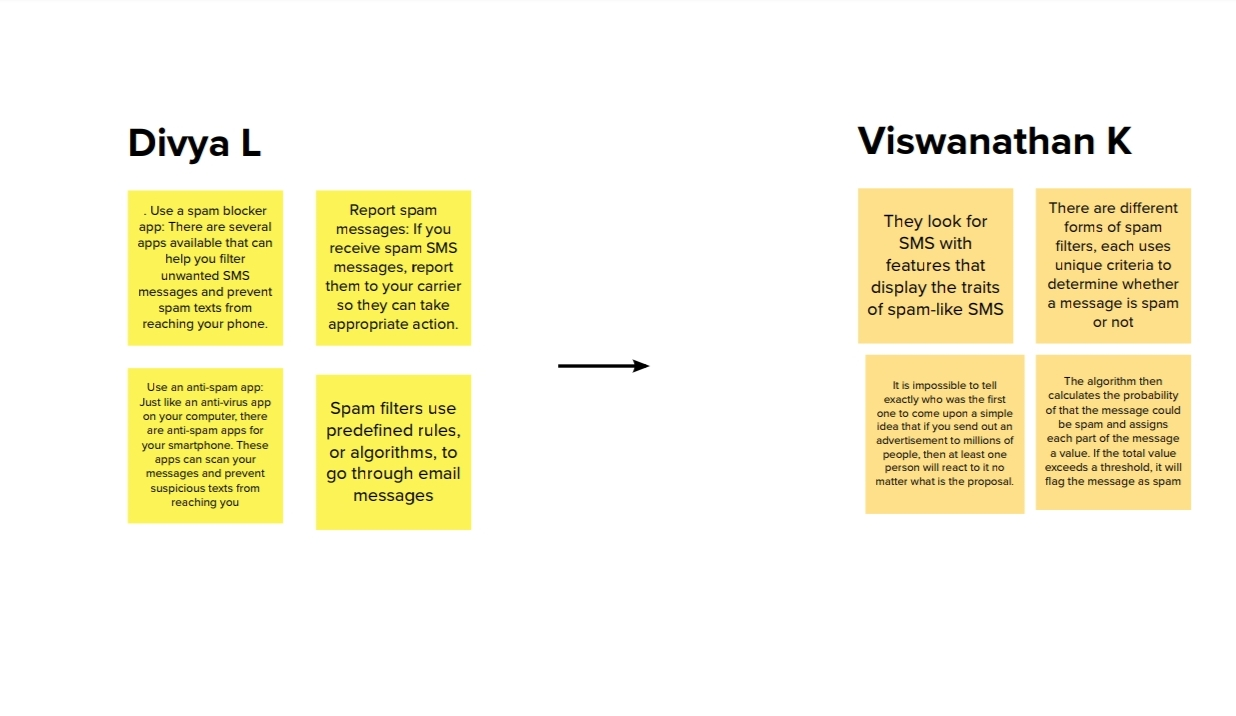


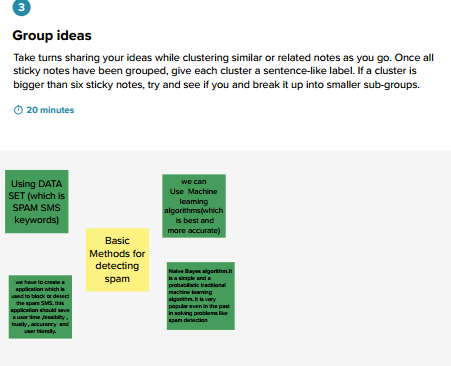
**Ideation and Brainstorming Map:**

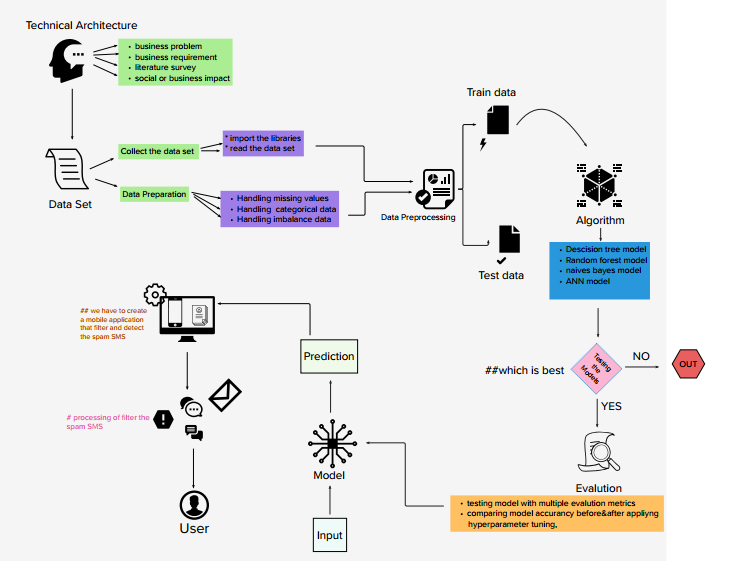


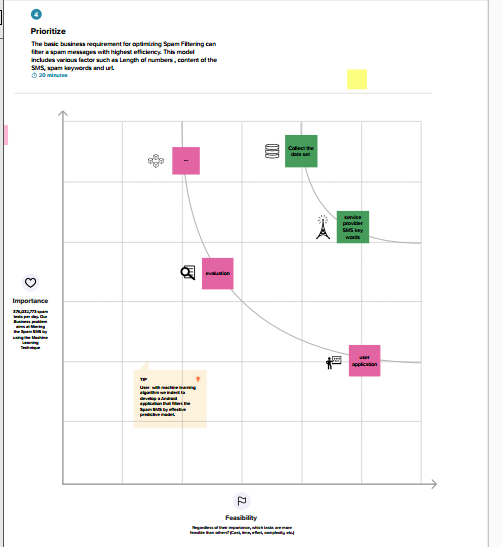






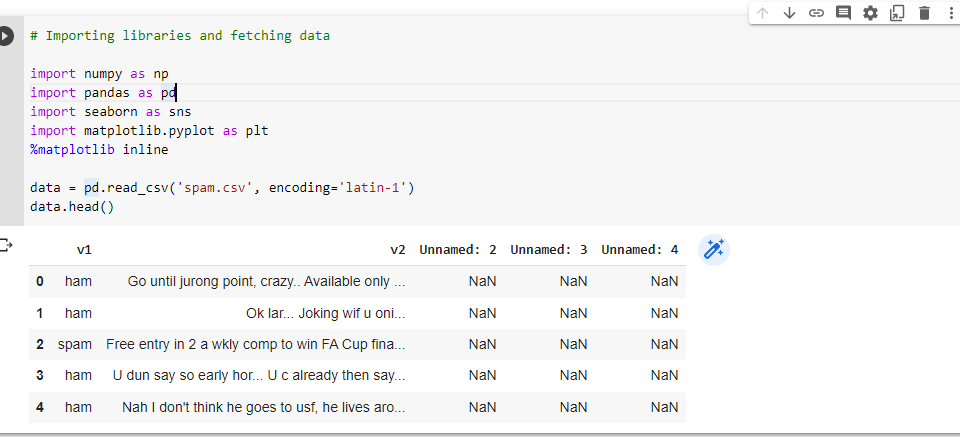




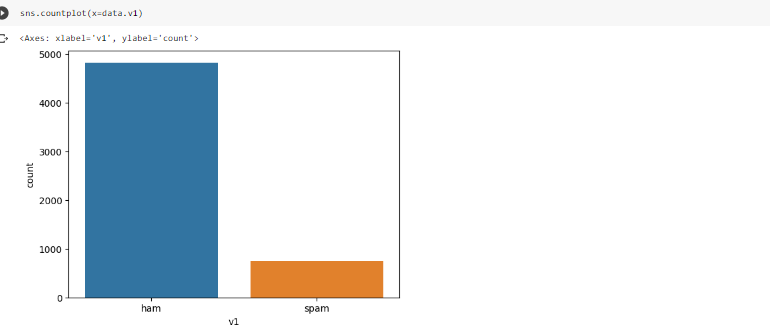


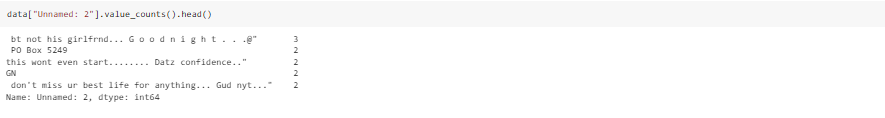
**3) RESULT:**

Output1 : Importing the libraries



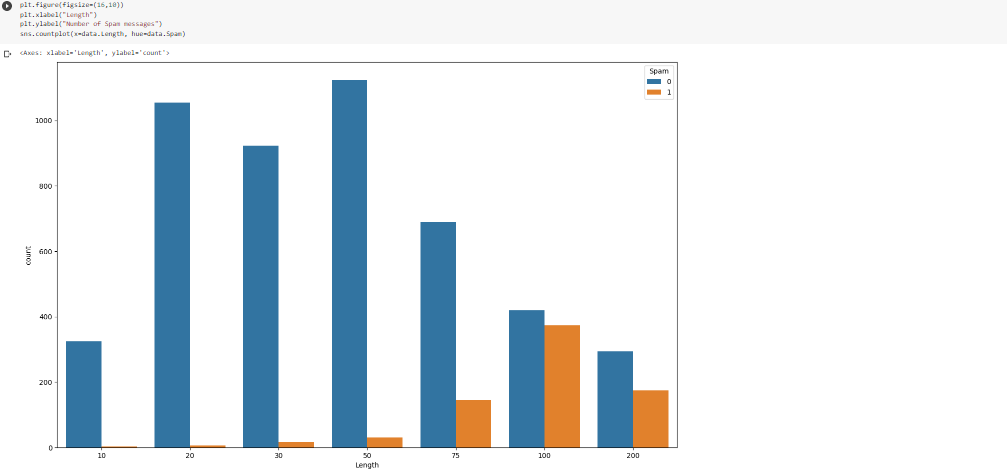
Output2 : Data analysis

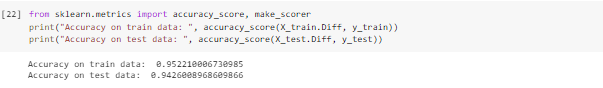


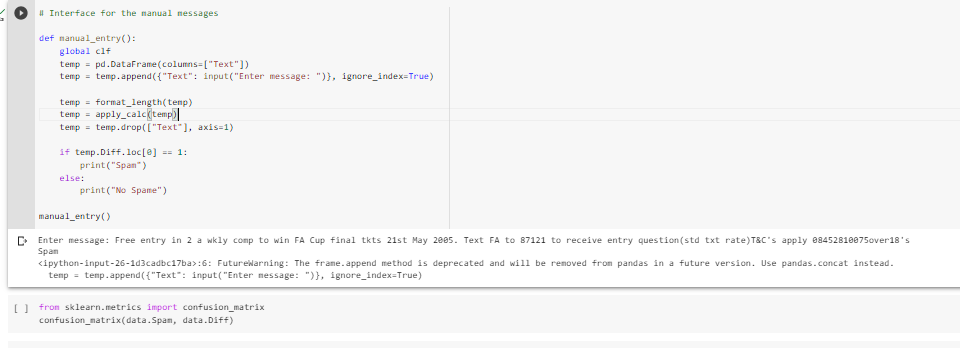


Output3:Create\_model

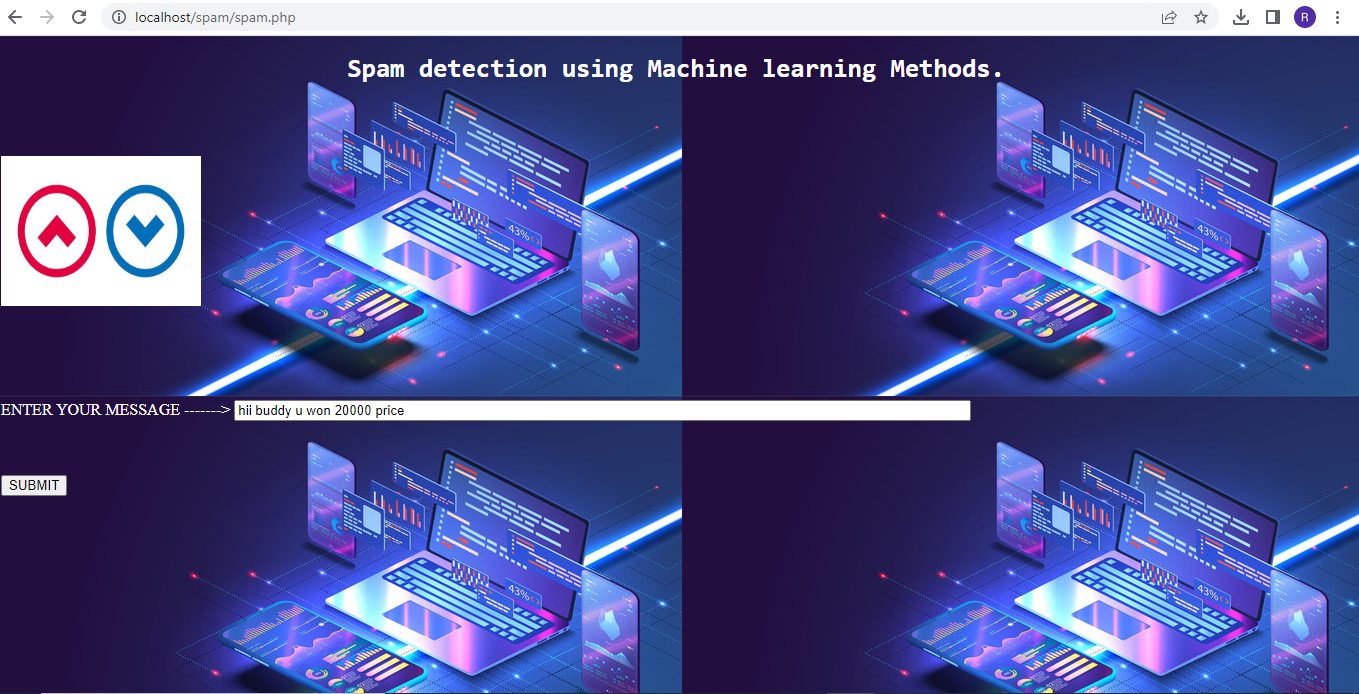




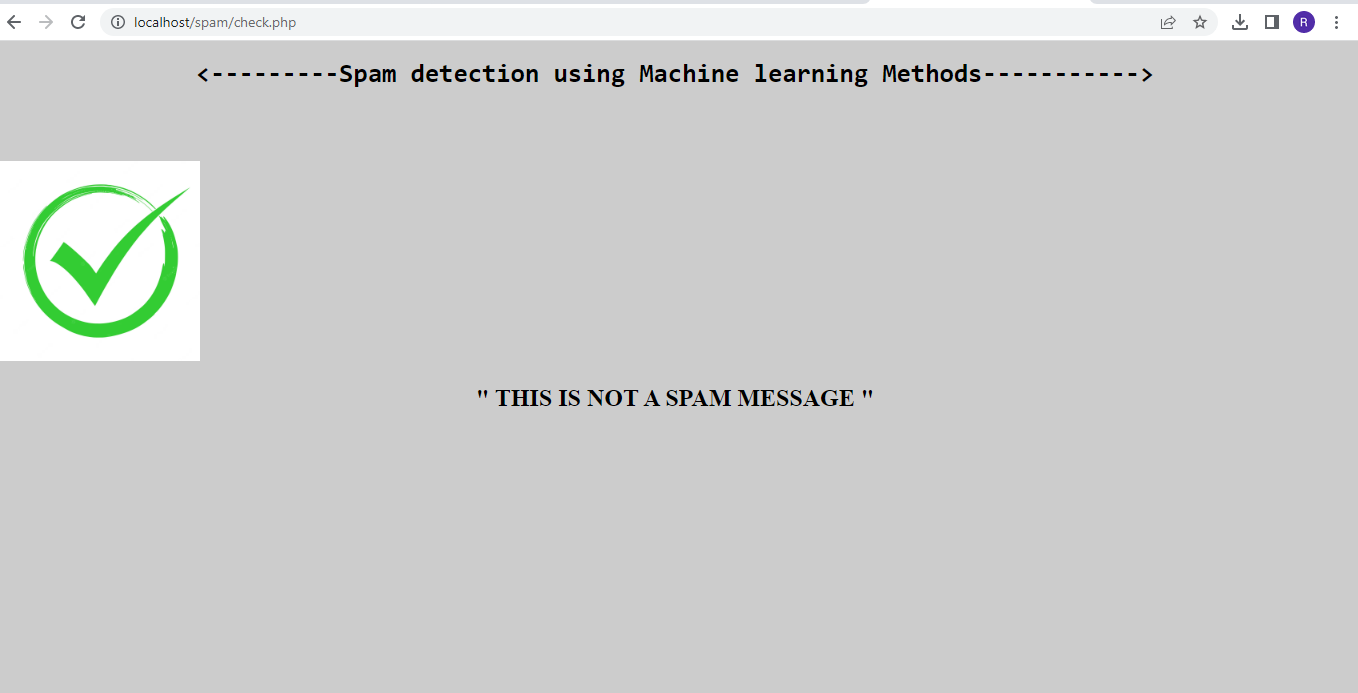




Output 4: Front end for the model



Output 5: final output



**4) ADVANTAGES AND DISADVANTAGE**

**Advantage :**

* The ideal anti-spam software(in our project) is the one that ensures you your privacy and blocks 99% percent of the SMS you do not want to receive.
* It can be customized to your needs, and only the approved SMS come into your inbox.
* This protects the user from any potential cyber threat and facilitates smooth communications and workflow

**Disadvantage :**

* But it is never going to be possible. Spammers are always inventing new techniques to trick the filters, and the developers of anti-spam software try not to overlook it.
* Thousands of spam SMS may reach Inboxes before a spammer's SMS s address, IP or domain is blacklisted. Spam filtering is machine-based so there is a room for mistakes called “false positives.
* Spam SMS can be the source of a great amount of malware like viruses, Trojans, worms, and others which are specifically designed to disrupt or damage computer systems.

**5) APPLICATIONS**

Recent updates about Spam SMS is Spammers tries to intrude in mobile Computing device. And SMS support for mobile devices had become vulnerable, as attacker tries to intrude to the system by sending unwanted link, with which on clicking those link the attacker can again remote access over the mobile computing device.

So , In our project( this application model )will be used or model enable the people to avoid viewing the unwanted SMS messages and all the filtered messages stands important . so we can use this model as a application in client androids and devices.

We can also use this model at Business side( eg : service provider ) , A lot of spam SMS came to our inbox which is look like similarly came from Service providers but that’s not originally came from service providers. (for example: VT-ViCARE , VT- V1CARE ) .By filtering the Spam SMS, Essential SMS and Detail alone are viewed which we promote the service provider business aspect using android application.

**6) CONCLUSION :**

In this end-to-end project we have learned how to approach a problem statement, and gather useful conclusions from the data using Data preprocessing, Data Visualisation which will help you build a good Machine Learning Model.In order to solve this classification problem we used the Naive Bayes Algorithm and in particular, the Multinomial Naive Bayes algorithm as it was having the highest precision score (i.e least False Positives)

**7) FUTURE SCOPE**

Advanced Feature Engineering: Currently, spam filters rely on a variety of features, such as keywords, sender reputation, and header information, to identify spam emails. However, with advancements in natural language processing (NLP) and deep learning, there is an opportunity to develop more advanced features that can capture the nuanced characteristics of spam emails, such as text sentiment, writing style, and contextual clues. Incorporating these features into machine learning models can potentially improve the accuracy and efficiency of spam detection.Hybrid Models: Combining multiple machine learning algorithms, such as supervised and unsupervised approaches, can lead to more robust spam filters. Hybrid models can leverage the strengths of different algorithms, for example, using supervised learning for labeled data and unsupervised learning for detecting new and unknown types of spam. This can lead to higher accuracy rates and improved generalization to handle new and evolving spamming techniques.Explainable AI: As machine learning models become more complex, it is important to ensure that their decision-making process is transparent and interpretable. Explainable AI techniques can help in understanding the reasoning behind the predictions made by spam filters. This can be valuable in building trust and accountability, as users can better understand why certain emails are flagged as spam and provide feedback for model improvement.

**8) APENDIX**

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

%matplotlib inline

data = pd.read\_csv('spam.csv', encoding='latin-1')

data.head()

sns.countplot(x=data.v1)

data.info()

data["Unnamed: 2"].value\_counts().head()

import nltk

nltk.download('stopwords')

import string

from nltk.corpus import stopwords

from nltk.stem import SnowballStemmer

stemmer = SnowballStemmer("english")

def simplify\_data(data):

data = pd.read\_csv('spam.csv', encoding='latin-1')

data["Spam"] = data.v1.map({'ham':0, 'spam':1})

data["Text"] = data.v2.str.lower()

data.Text = data.Text.str.replace(r'[.,\\&;!:-?(|)#@$^%\*0-9/\'\"+={|}~`\_[|]]\*', '')

data = data.drop(["v1", "v2", "Unnamed: 2", "Unnamed: 3", "Unnamed: 4"], axis=1)

return data

def remove\_stopwords(message):

# Remove stop words from the text

stop\_words = set(stopwords.words('english'))

message = message.translate(str.maketrans('', '', string.punctuation))

text = [word for word in message.split() if word not in stop\_words and len(word) > 2]

return " ".join(text)

def text\_length(text):

return len(text)

def format\_length(data):

data["Length"] = data.Text.apply(text\_length)

data.Length = pd.cut(data.Length, [-1, 10, 20, 30, 50, 75, 100, 999], labels=[10,20,30,50,75,100,200])

return data

def apply\_transformations(data):

data = simplify\_data(data)

data.Text = data.Text.apply(remove\_stopwords)

data = format\_length(data)

return data

data = apply\_transformations(data)

data.head()

#data visual

plt.figure(figsize=(16,10))

plt.xlabel("Length")

plt.ylabel("Number of Spam messages")

sns.countplot(x=data.Length, hue=data.Spam)

spam\_words = []

ham\_words = []

def getSpam(text):

global spam\_words, spam\_messages

messages = text.split()

words = [x for x in messages]

spam\_words += words

def getHam(text):

global ham\_words, ham\_messages

messages = text.split()

words = [x for x in messages]

ham\_words += words

# Separate spam and ham messages

spam\_messages = data[data["Spam"] == 1]["Text"]

ham\_messages = data[data["Spam"] == 0]["Text"]

# Store common words in Spam/Ham

spam\_messages.apply(getSpam)

ham\_messages.apply(getHam)

def countSpam(text):

count = 0

for x in text.split():

if x in spam\_words:

count += spam\_words.count(x)

return count

def countHam(text):

count = 0

for x in text.split():

if x in ham\_words:

count += ham\_words.count(x)

return count

def getCounts(data):

SpamCount = data.Text.apply(countSpam)

HamCount = data.Text.apply(countHam)

data["Diff"] = SpamCount - HamCount

return data

def categorize(diff):

if diff <= 0:

return 0

else:

return 1

def apply\_calc(data):

data = getCounts(data)

data.Diff = data.Diff.apply(categorize)

return data

data = apply\_calc(data)

data.head()

spam\_words.count("free")

ham\_words.count("free")

from sklearn.model\_selection import train\_test\_split, GridSearchCV

X = data.drop(["Spam"], axis=1)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, data.Spam, test\_size=0.2, random\_state=23)

from sklearn.metrics import accuracy\_score, make\_scorer

print("Accuracy on train data: ", accuracy\_score(X\_train.Diff, y\_train))

print("Accuracy on test data: ", accuracy\_score(X\_test.Diff, y\_test))

from sklearn.ensemble import RandomForestClassifier

from sklearn.svm import SVC

from sklearn.naive\_bayes import GaussianNB

X\_train = X\_train[["Length", "Diff"]]

X\_test = X\_test[["Length", "Diff"]]

# RandomForestModel

# Trying different parameters and selecting the best one's to run

clf = RandomForestClassifier()

parameters = {'n\_estimators': [4, 6, 9],

'max\_features': ['log2', 'sqrt','auto'],

'criterion': ['entropy', 'gini'],

'max\_depth': [2, 3, 5, 10],

'min\_samples\_split': [2, 3, 5],

'min\_samples\_leaf': [1,5,8]

}

acc\_scorer = make\_scorer(accuracy\_score)

grid\_obj = GridSearchCV(clf, parameters, scoring=acc\_scorer, cv=3)

grid\_obj = grid\_obj.fit(X\_train, y\_train)

clf = grid\_obj.best\_estimator\_

clf.fit(X\_train, y\_train)

# Predicting the reuslts and calculating the accuracy

preds = clf.predict(X\_test)

clf\_acc = nb\_acc = accuracy\_score(y\_test, preds)

print("Accuracy with RandomForestClassifier: ", accuracy\_score(y\_test, preds))

# SVC model

svc\_clf = SVC(gamma='scale')

svc\_clf.fit(X\_train,y\_train)

svc\_preds = svc\_clf.predict(X\_test)

svc\_acc = accuracy\_score(y\_test, svc\_preds)

print("Accuracy with SVC: ", accuracy\_score(y\_test, svc\_preds))

nb = GaussianNB()

nb.fit(X\_train, y\_train)

nb\_preds = nb.predict(X\_test)

nb\_acc = accuracy\_score(y\_test, nb\_preds)

print("Accuracy with NaiveBayesian: ", accuracy\_score(y\_test, nb\_preds))

sns.countplot(x=X\_test.Length, hue=y\_test)

# Interface for the manual messages

def manual\_entry():

global clf

temp = pd.DataFrame(columns=["Text"])

temp = temp.append({"Text": input("Enter message: ")}, ignore\_index=True)

temp = format\_length(temp)

temp = apply\_calc(temp)

temp = temp.drop(["Text"], axis=1)

if temp.Diff.loc[0] == 1:

print("Spam")

else:

print("No Spame")

manual\_entry()