Optimizing Spam Filtering with Machine Learning

The project submitted to Smart Internz

By
P.NAVANEETHAKRISHNAN (2022K2012)
P.PRASANTH (2022K2013)
K.PUGAZHENTHI (2022K2014)
M.RAJKUMAR (2022K2016)

Department of Computer Science
GOVERNMENT ARTS AND SCIENCE COLLEGE
MODAKURICHI
(Affiliated to Bharathiar University)
ERODE - 638104

INDEX

NO.	TITLE	PAGE NO
1.	INTRODUCTION	3
2.	PROBLEM DEFINITION & DESIGN THINKING	4
3.	RESULT	8
4.	ADVANTAGES & DISADVANTAGES	9
5.	APPLICATIONS	10
6.	CONCLUSION	11
7.	FUTURE SCOPE	11
8.	APPENDIX	12

ABSTRACT:

Spam filtering is the process of identifying and removing unsolicited or unwanted emails from a user's inbox. The goal of spam filtering is to accurately classify emails as either spam or legitimate, while minimizing the number of false positives and false negatives. Several approaches have been developed for spam filtering, including rule-based filtering, content-based filtering, and machine learning-based filtering. Machine learning-based approaches have become increasingly popular in recent years, as they can automatically learn from large amounts of data and adapt to new types of spam. Commonly used techniques in machine learning-based spam filtering include naive Bayes, support vector machines, and artificial neural networks. Despite the success of these approaches, spam filtering remains a challenging problem due to the constantly evolving nature of spam and the need for continuous adaptation and improvement of filtering algorithms.

1.INTRODUCTION:

- A spam filter is a program used to detect unsolicited, unwanted and virus-infected emails and prevent those messages from getting to a user's inbox.
- different types of the spam filtering is available, ex: content filters, permission filters, community filters, etc.

1.1 Overview

- 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 arrive sat 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.

- This interface allows you to configure the spam filter settings (powered by Apache Spam Assassin™) for your account. Spam filters identify and sort or delete unsolicited

email, commonly known as spam. You can also use this interface to configure your whitelist and blacklist settings. For more information, read Apache Spam Assassin's overview documentation.

1.2 Purpose

- A spam filter is an email service feature that filters and quarantines spam emails from a user's inbox. It ensures a clean and spam-free mailbox for the user that saves productive energy and provides a high level of work efficiency. A spam filter has become a necessity in recent times as cyber adversaries rigorously work to find and exploit vulnerabilities. Sending spam and phishing emails is one of the most preferred and lucrative ways which they can use to lure users into falling into their trap.

- For security reasons, there is an urge for an effective spam filter system. The spam filter is a reliable security control against adversaries. The job of a spam filter is to look for particular words in the email or website content based on precise filtering analysis and consequently excludes them from the user's inbox. If we check the various published reports.

2. PROBLEM DEFINITION & DESIGN THINKING

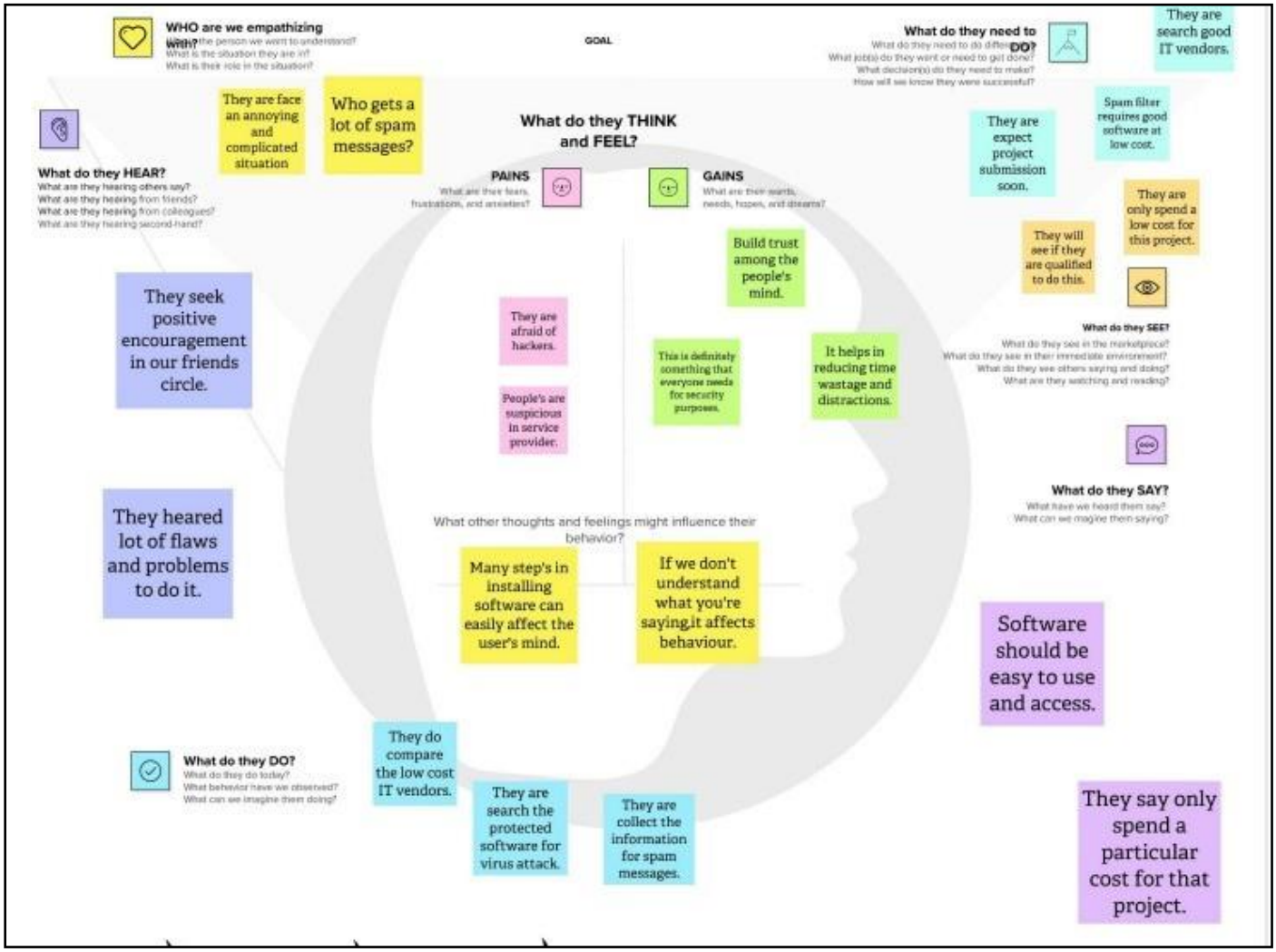
- Social Impact:- it can help protect individuals from unwanted and potentially harmful messages. Spam messages can include phishing attempts, scams, and fraud, which can have serious financial and personal consequences for recipients. By accurately identifying and flagging spam messages, the system can help prevent these types of attacks and protect individuals from falling victim to them.

- Business Model/Impact:- it can help protect their customers and improve their reputation. Spam messages can harm a business's reputation and lead to customer complaints and lost business. By accurately identifying and flagging spam messages, the system can help protect businesses and improve their customer's trust.

2.1 Empathy Map

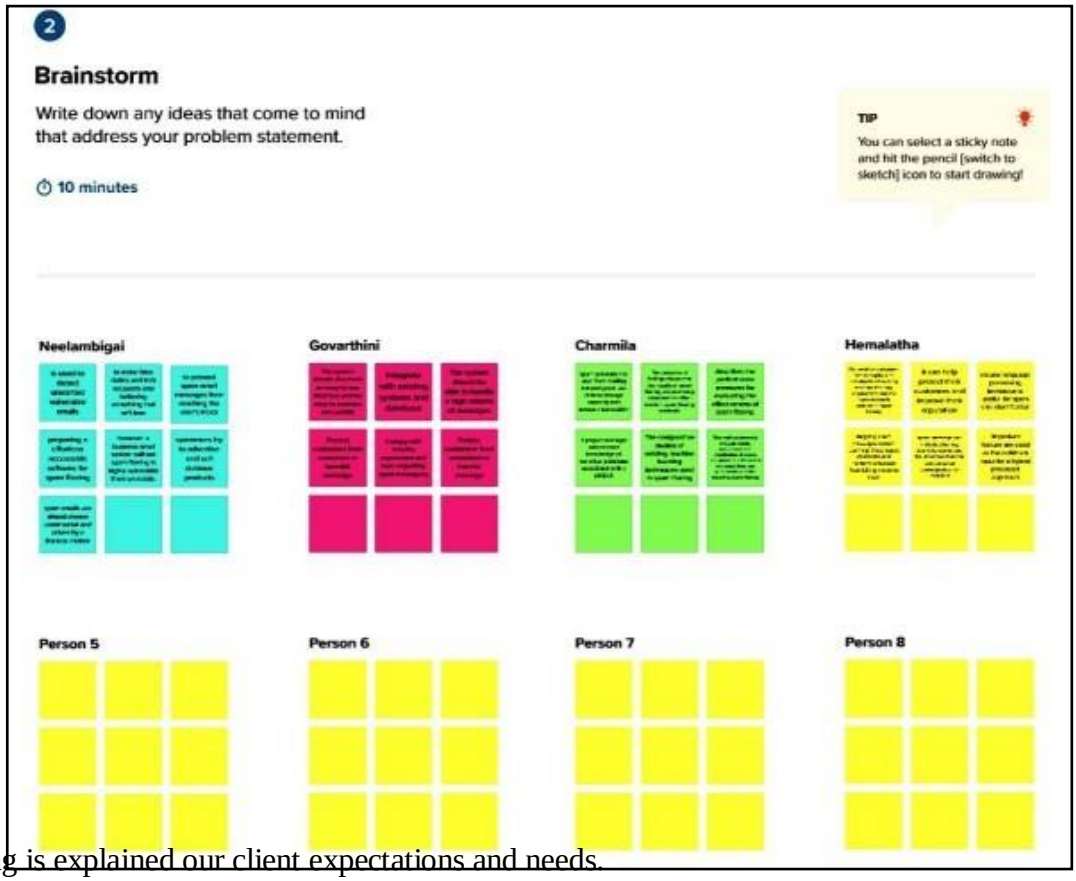
- An empathy map is a template that organizes a user's behaviors and feelings to create a sense of empathy between the user and your team. The empathy map represents a principal user and helps teams better understand their motivations, concerns, and user experience.

- In the following empathy map is explained about our project spam filtering troubles and plan for a clarification.

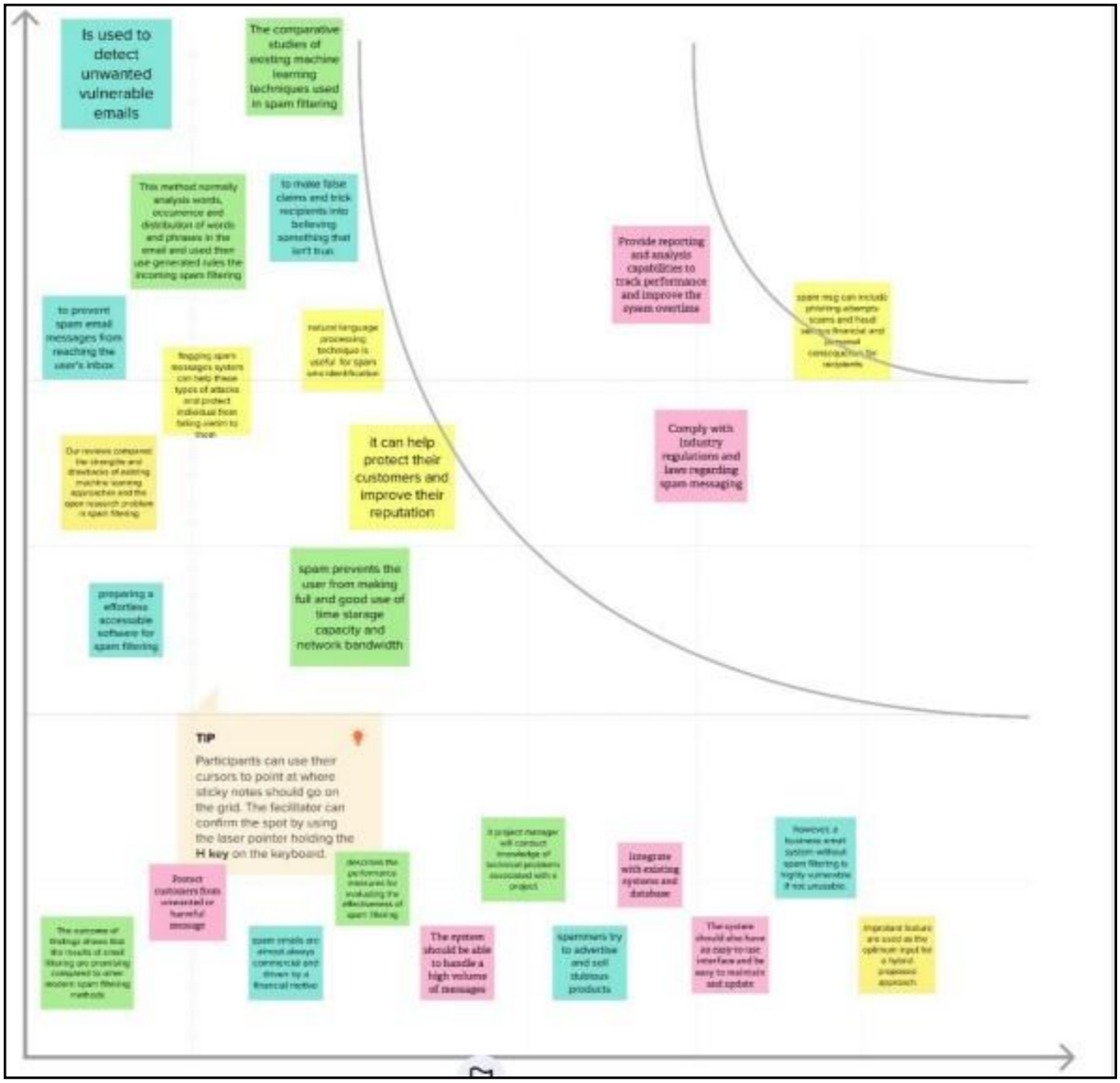


2.2 Ideation & Brainstorming Map

Brainstorming is a group problem-solving method that involves the spontaneous contribution of creative ideas and solutions. This technique requires intensive, freewheeling discussion in which every member of the group is encouraged to think aloud and suggest as many ideas as possible based on their diverse knowledge.

The 

following brainstorming is explained our client expectations and needs.

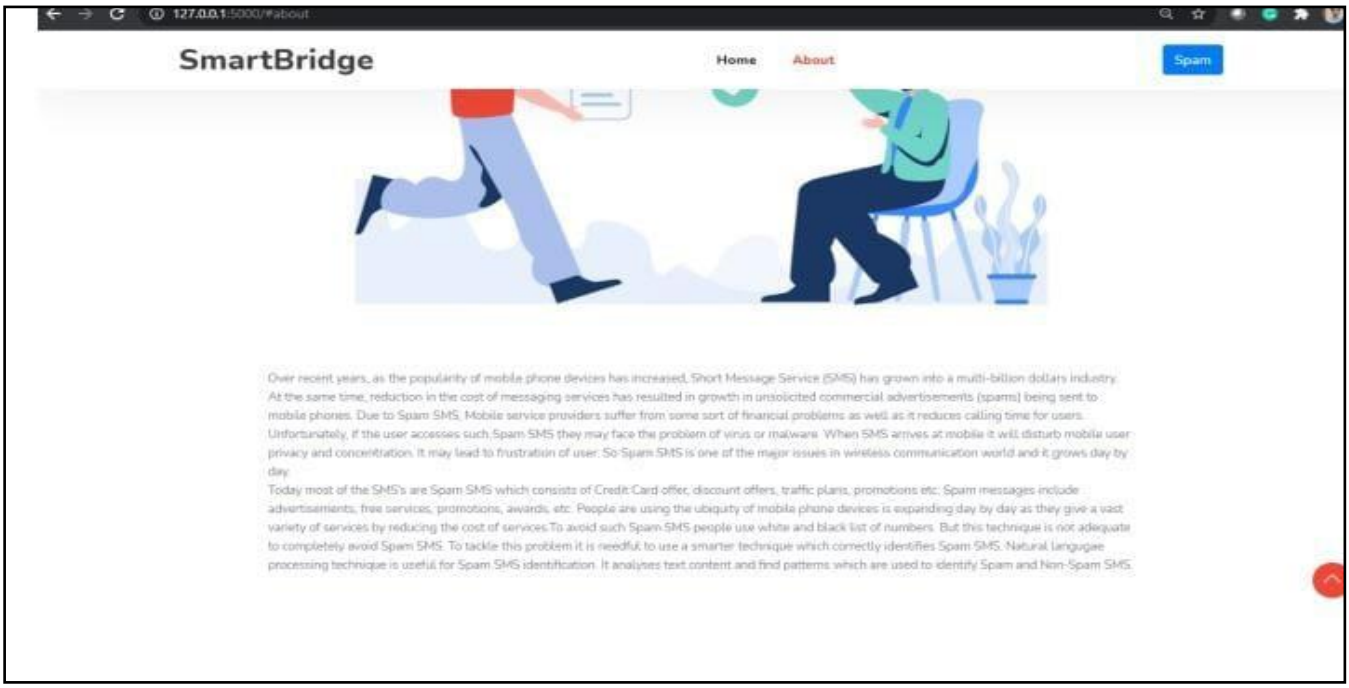


The above brainstorming is explained our client expectations and needs.

3.RESULT:

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

The following image is the spam detection about page.



In the following image is the spam detection page.

4.ADVANTAGES & DISADVANTAGES:

ADVANTAGES OF SPAM FILTERING:

- Saves time: By filtering out unwanted messages, spam filtering saves time by reducing the amount of time spent sorting through irrelevant messages.

- Increases productivity: With less time spent on spam emails, individuals can focus on important tasks and be more productive.

- Protects against malware: Spam emails often contain malware or viruses that can damage computer systems. Spam filtering can help protect against such threats by preventing these emails from reaching the inbox.

- Reduces the risk of phishing attacks: Phishing emails are designed to trick individuals into giving away sensitive information. Spam filtering can help reduce the risk of such attacks by filtering out suspicious messages.

- Improves security: Spam filtering helps to maintain the security of computer systems by preventing unwanted and potentially harmful messages from entering the inbox.

- Enhances user experience: By reducing the amount of spam in the inbox, spam filtering can improve the user experience and make email communication more efficient and effective.

DISADVANTAGES OF SPAM FILTERING:

- False positives: Over-optimizing a spam filter can lead to legitimate emails being marked as spam and sent to the user's junk folder. This can result in important messages being missed or delayed, causing frustration and potentially damaging business relationships.

- Resource-intensive: Optimizing a spam filter often requires a significant amount of computational resources, which can slow down email processing and increase the time it takes for messages to reach their intended recipients.

- Adapting to new spam tactics: Spam tactics are constantly evolving, and it can be difficult for a spam filter to keep up with the latest techniques used by spammers. Over -optimization may lead to a false sense of security, as the filter may not be able to detect new forms of spam that it has not been trained on.

- Privacy concerns: Some spam filters rely on analyzing the content of emails to determine whether they are spam or not. This raises privacy concerns, as it may involve scanning the contents of emails that contain sensitive or confidential

information.

- Cost: Optimizing a spam filter can be costly, as it may require purchasing specialized software or hiring experts in machine learning and natural language processing. Small businesses or individuals may not have the resources to invest in this level of optimization.

5.APPLICATIONS:

- Spam filtering is a common problem in email communication and involves the automatic identification and removal of unwanted, unsolicited email messages, commonly referred to as spam. Optimization techniques can be applied to spam filtering to improve the accuracy and efficiency of the filtering process. Here are some ways optimization can be applied in spam filtering:

- Feature selection: Spam filtering involves identifying certain features of an email that distinguish it from legitimate messages. Optimization techniques can be applied to select the most relevant features that can be used to classify emails as spam or not.

- Algorithm selection: Various machine learning algorithms can be used for spam filtering. Optimization techniques can be applied to select the most suitable algorithm for a particular dataset.

- Parameter tuning: The performance of machine learning algorithms depends on various parameters that need to be tuned for optimal performance. Optimization techniques such as grid search and random search can be used to find the best values of these parameters.

- Ensemble methods: Ensemble methods involve combining multiple machine learning models to improve the accuracy of spam filtering. Optimization techniques can be used to determine the optimal combination of models to use in an ensemble.

- Active learning: Active learning involves selecting the most informative samples for labeling to improve the accuracy of a machine learning model. Optimization techniques can be applied to select the most informative samples for labeling, thus reducing the number of samples required for training.

6.CONCLUSION:

- Spam filtering is the process of identifying and filtering out unsolicited or unwanted

messages, typically emails, from a user's inbox. Spam filtering is an essential tool for protecting users from unwanted messages, such as phishing scams, malware, and unwanted advertisements.

- There are different techniques used for spam filtering, including content-based filtering, blacklisting, and whitelisting. Content-based filtering involves analyzing the content of the message, including the subject line, body, and attachments, to determine if it's spam. Blacklisting involves blocking messages from known spam senders, while whitelisting involves allowing messages from known trusted senders.

- In conclusion, spam filtering is an important tool for protecting users from unwanted and potentially harmful messages. It's important to use a combination of different techniques to ensure the highest level of protection.

7.FUTURE SCOPE

- The future scope of optimizing spam filtering is quite promising as there are always new techniques and methods being developed to improve the accuracy and efficiency of spam filtering. Some potential areas for improvement include:

- Machine Learning: With the advancements in machine learning techniques such as deep learning, neural networks, and natural language processing, there is a great potential to improve the accuracy of spam filtering. This is because machine learning algorithms can learn from vast amounts of data and can adapt to new types of spam messages.

- Artificial Intelligence: AI-powered spam filters can use advanced algorithms to analyze user behavior and preferences to identify spam messages. AI-powered spam filters can also learn from user feedback and adjust the filters accordingly.

- Collaborative filtering: Collaborative filtering is a technique that uses the preferences and behavior of a group of users to identify spam messages. This technique has the potential to improve the accuracy of spam filtering as it takes into account the preferences and behavior of a large group of users.

- Behavioral analysis: Behavioral analysis is a technique that analyzes the behavior of users to identify spam messages. This technique can help in identifying phishing emails that look legitimate but are actually designed to steal personal information.

- Big data analysis: Big data analysis can be used to analyze large amounts of data to identify patterns and trends in spam messages. This can help in developing more effective spam filtering algorithms.

- Overall, optimizing spam filtering is an ongoing process, and with the advancements in technology, there is great potential for improving the accuracy and efficiency of spam filtering in the future.

8.APPENDIX:

SOURCE CODE:

Milestone 1: Data Collection & Preparation

ML depends heavily on data. It is the most crucial aspect that makes algorithm training possible. So this section allows you to download the required dataset.

Activity 1: Collect the dataset

There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc. In this project we have used .csv data. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset. Link: https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset As the dataset is downloaded. Let us read and understand the data properly with the help of some visualisation techniques and some analysing techniques.

Note: There are a number of techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques.

Activity 1.1: Importing the libraries

Import the necessary libraries as shown in the image. (optional) Here we have used visualisation style as fivethirtyeight

```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Activity 1.2: Read the Dataset

Our dataset format might be in .csv, excel files, .txt, .json, etc. We can read the dataset with the help of pandas. In pandas we have a function called read_csv() to read the dataset. As a parameter we have to give the directory of the csv file.

Loading the dataset

```
df = pd.read_csv('/content/spam.csv', encoding="ISO-8859-1")
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

Activity 2: Data Preparation

As we have understood how the data is, let's pre-process the collected data. The download data set is not suitable for training the machine learning model as it might have so much randomness so we need to clean the dataset properly in order to fetch good results. This activity includes the following steps.

- Handling missing values
- Handling categorical data
- Handling Imbalance Data

Note: These are the general steps of pre-processing the data before using it for machine learning. Depending on the condition of your dataset, you may or may not have to go through all these steps.

Activity 2.1: Handling missing values

Let's find the shape of our dataset first. To find the shape of our data, the df.shape method is used. To find the data type, df.info() function is used.

```
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
```

For checking the null values, df.isnull() function is used. To sum those null values we use .sum() function. From the below image we found that there are no null values present in our dataset. So we can skip handling the missing values step.

```
[48] df.isna().sum()

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

From the above code of analysis, we can infer that columns such as V1 and v2 are not having missing columns,unnamed columns are not required for analysis

Renaming the columns according the requirement mapping values for label.

```
df = df[['v1','v2']]
df.rename(columns = {'v1':'label','v2':'msg'}, inplace = True)

<ipython-input-6-c1c62c3b687b>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-vie
  df.rename(columns = {'v1':'label','v2':'msg'}, inplace = True)
```

```
df.head()
```

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

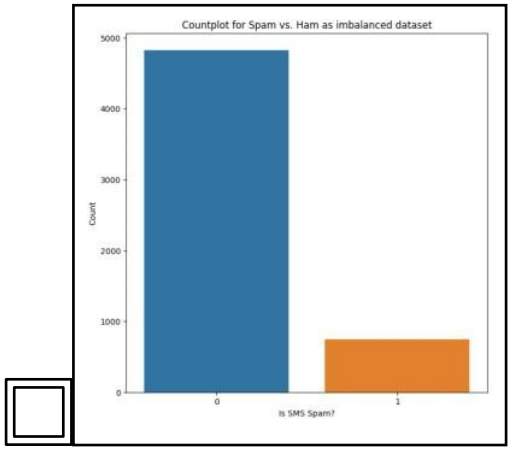
Mapping values for label

```
[11] df['label'] = df['label'].map({'ham': 0, 'spam': 1})
```

A countpolt for a spam and ham as imbalanced dataset.

Countplot for Spam vs. Ham as imbalanced dataset

```
[12] plt.figure(figsize=(8,8))
     g = sns.countplot(x='label', data=df)
     p = plt.title('Countplot for Spam vs. Ham as imbalanced dataset')
     p = plt.xlabel('Is SMS Spam?')
     p = plt.ylabel('Count')
```



Activity 2.3 Handling Imbalance Data

Data Balancing is one of the most important step, which need to be performed for classification models, because when we train our model on imbalanced data set ,we will get biassed results, which means our model is able to predict only one class element for balancing the data.

We are using oversampling method.

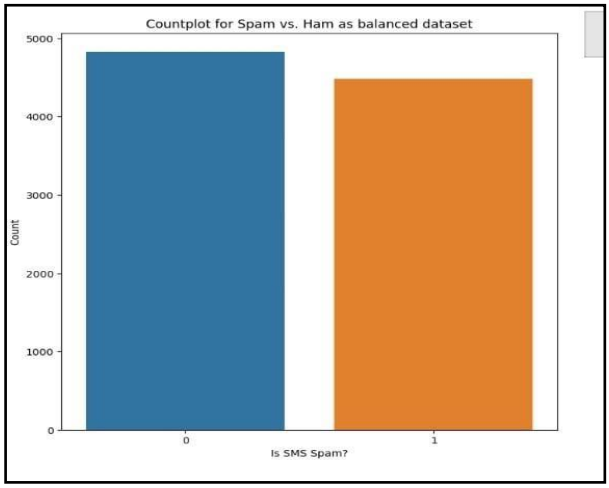
```
only_spam = df[df['label']==1]
print('Number of Spam records: {}'.format(only_spam.shape[0]))
print('Number of Ham records: {}'.format(df.shape[0]-only_spam.shape[0]))
```

```
Number of Spam records: 747
Number of Ham records: 4825
```

```
[61] count = int((df.shape[0]-only_spam.shape[0])/only_spam.shape[0])
     for i in range(0, count-1):
       df = pd.concat([df, only_spam])

     df.shape
```

```
(9307, 2)
```

```
plt.figure(figsize=(8,8))
g = sns.countplot(x='label', data=df)
p = plt.title('Countplot for Spam vs. Ham as balanced dataset')
p = plt.xlabel('Is SMS Spam?')
p = plt.ylabel('Count')
```



Creating new feature:

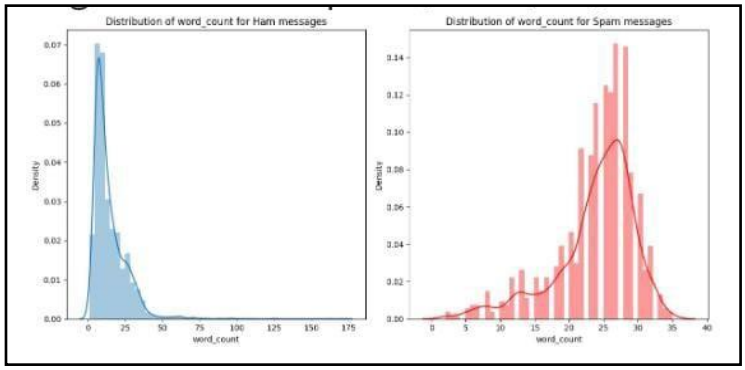
Word count is a commonly used feature in spam filtering using machine learning. The idea behind using word count is that certain words or phrases are more likely to be used in spam messages than legitimate ones. By counting the frequency of these words in each message, we can create a feature vector that can be used as input to a machine learning algorithm.

```
# Creating new feature word_count
df['word_count'] = df['msg'].apply(lambda x: len(x.split()))
df.head()
```

	label	msg	word_count
0	0	Go until jurong point, crazy.. Available only ...	20
1	0	Ok lar... Joking wif u oni...	6
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	28
3	0	U dun say so early hor... U c already then say...	11
4	0	Nah I don't think he goes to usf, he lives aro...	13

```
plt.figure(figsize=(12, 6))

# 1-row, 2-column, go to the first subplot
plt.subplot(1, 2, 1)
g = sns.distplot(a=df[df['label']==0].word_count)
p = plt.title('Distribution of word_count for Ham messages')

# 1-row, 2-column, go to the second subplot
plt.subplot(1, 2, 2)
g = sns.distplot(a=df[df['label']==1].word_count, color='red')
p = plt.title('Distribution of word_count for Spam messages')

plt.tight_layout()
plt.show()
```

```
<ipython-input-84-6776e7c342cf>:5: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  g = sns.distplot(a=df[df['label']==0].word_count)
<ipython-input-84-6776e7c342cf>:10: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  g = sns.distplot(a=df[df['label']==1].word_count, color='red')
```



Activity 2.3 Cleaning the text data

Data Cleaning

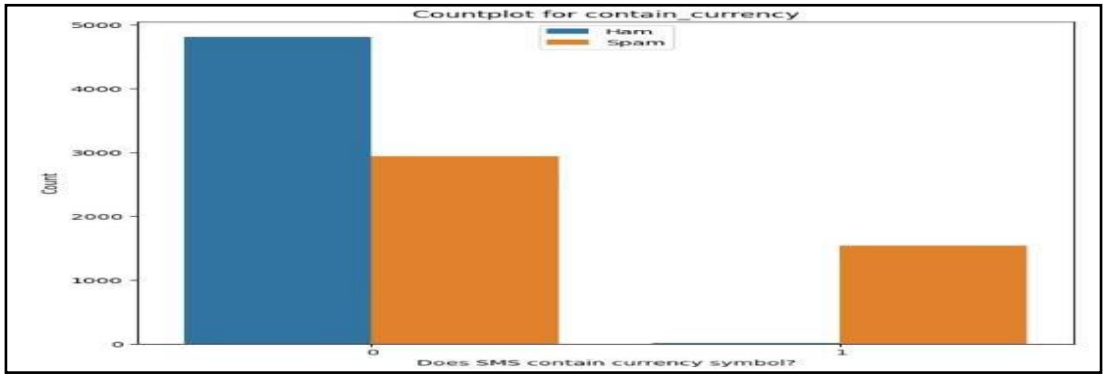
* Removing special character and numbers using regular expression
* Converting the entire sms into lower case
* Tokenizing the sms by words
* Removing the stop words
* Lemmatizing the words
* Joining the lemmatized words
* Building a corpus of messages

Spam messages word_count fall in the range of 15-
30 words, whereas majority of the Ham messages fall in the range of below 25 words.

```
plt.figure(fig
g = sns.countp
p = plt.title(
p = plt.xlabel
p = plt.ylabel
p = plt.legend
```

```
def currency(x):
  currency_symbols = ['€', '$', '¥', '£', '₹']
  for i in currency_symbols:
    if i in x:
      return 1
  return 0

df['contains_currency_symbol'] = df['msg'].apply(currency)
df.head()
```

	label	msg	word_count	contains_currency_symbol
0	0	Go until jurong point, crazy.. Available only ...	20	0
1	0	Ok lar... Joking wif u oni...	6	0
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	28	0
3	0	U dun say so early hor... U c already then say...	11	0
4	0	Nah I don't think he goes to usf, he lives aro...	13	0



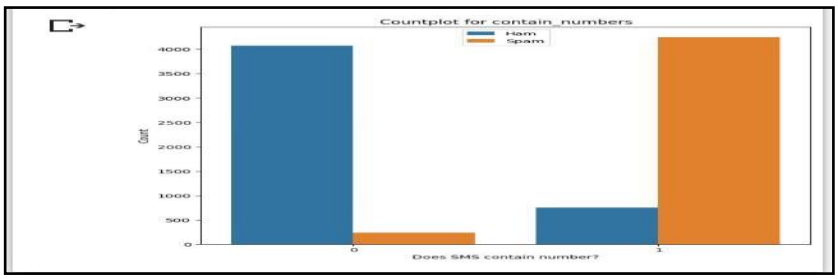
Almost 1/3 of Spam messages contain currency symbols, and currency symbols are rarely used in Ham messages.

```
def numbers(x):
  for i in x:
    if ord(i)>=48 and ord(i)<=57:
      return 1
  return 0

df['contains_number'] = df['msg'].apply(numbers)
df.head()
```

	label	msg	word_count	contains_currency_symbol	contains_number
0	0	Go until jurong point, crazy.. Available only ...	20	0	0
1	0	Ok lar... Joking wif u oni...	6	0	0
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	28	0	1
3	0	U dun say so early hor... U c already then say...	11	0	0
4	0	Nah I don't think he goes to usf, he lives aro...	13	0	0

Using countplot for contain numbers

```
plt.figure(figsize=(8,8))
g = sns.countplot(x='contains_number', data=df, hue='label')
p = plt.title('Countplot for contain_numbers')
p = plt.xlabel('Does SMS contain number?')
p = plt.ylabel('Count')
p = plt.legend(labels=['Ham', 'Spam'], loc=9)
```



Most of the Spam messages contain numbers, and majority of the Ham messages don't co ntain
numbers.

```
import nltk
import re
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('omw-1.4')
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
```

```
corpus = []
wnl = WordNetLemmatizer()

for sms_string in list(df.msg):

  # Cleaning special character from the sms
  message = re.sub(pattern='[^a-zA-Z]', repl=' ', string=sms_string)

  # Converting the entire sms into lower case
  message = message.lower()

  # Tokenizing the sms by words
  words = message.split()

  # Removing the stop words
  filtered_words = [word for word in words if word not in set(stopwords.words('english'))]

  # Lemmatizing the words
  lemmatized_words = [wnl.lemmatize(word) for word in filtered_words]

  # Joining the lemmatized words
  message = ' '.join(lemmatized_words)

  # Building a corpus of messages
  corpus.append(message)
```

In creating a new methods and features for cleaning the special charecters from the sms, converting the entires sms into lower case , tokenizing the sms by words lemmatized words, building a corpus of messages.

```
# Creating the Bag of Words model
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf = TfidfVectorizer(max_features=500)
vectors = tfidf.fit_transform(corpus).toarray()
feature_names = tfidf.get_feature_names_out()

# Extracting independent and dependent variables from the dataset
X = pd.DataFrame(vectors, columns=feature_names)
y = df['label']
```

```
X
```

	ac	access	account	address	admirer	age	already	also	always	amp	...	xxx	ya	yeah	year	yes	yesterday	yet	yo	yr	yup
0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.409254	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9302	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
9303	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.325761	0.0	0.0	0.0	0.0	0.0
9304	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.287064	0.0	0.0	0.0	0.0	0.0
9305	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
9306	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0

9307 rows × 500 columns

Above the picture explain the creating the bag of words model,extracting independent and dependent variables from dataset

Milestone 2: Exploratory Data Analysis

Activity 1: Descriptive statistical

Descriptive analysis is to study the basic features of data with the statistical process. Here pandas has a worthy function called describe. With this describe function we can understand the unique, top and frequent values of categorical features. And we can findmean, std, min, max and

percentile values of continuous features

```
df.describe()
```

	label	word_count	contains_currency_symbol	contains_number
count	9307.000000	9307.000000	9307.000000	9307.000000
mean	0.481573	18.848179	0.167616	0.536800
std	0.499687	10.351809	0.373545	0.498671
min	0.000000	1.000000	0.000000	0.000000
25%	0.000000	10.000000	0.000000	0.000000
50%	0.000000	20.000000	0.000000	1.000000
75%	1.000000	26.000000	0.000000	1.000000
max	1.000000	171.000000	1.000000	1.000000

Model Building & Evaluation

```
[87] df.shape
     (9307, 5)
```

Activity 2: Visual analysis

Visual analysis is the process of using visual representations, such as charts, plots, and graphs, to explore and understand data. It is a way to quickly identify patterns, trends, and outliers in the data, which can help to gain insights and make informed decisions.

```
from sklearn.model_selection import cross_val_score
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.model_selection import train_test_split

[74] #split the dataset
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Milestone 3: Model Building

Activity 1: Training the model in multiple algorithms

Now our data is cleaned and it’s time to build the model. We can train our data on different algorithms. For this project we are applying four classification algorithms. The best model is saved based on its performance.

Activity 1.1: Decision tree model

A function named decisionTree is created and train and test data are passed as the parameters. Inside the function, DecisionTreeClassifier algorithm is initialised and training data is passed to the model with the .fit() function. Test data is predicted with .predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is

done. Fitting

Fitting the decision tree model

Activity 1.2: Random forest model

```
# Classification report for Decision Tree model
dt = DecisionTreeClassifier()
dt.fit(X_train, y_train)
y_pred = dt.predict(X_test)

print('--- Classification report for Decision Tree model ---')
print(classification_report(y_test, y_pred))
```

```
--- Classification report for Decision Tree model ---
              precision    recall  f1-score   support

           0       1.00      0.97      0.98       958
           1       0.97      1.00      0.98       904

    accuracy                           0.98      1862
   macro avg       0.98      0.98      0.98      1862
weighted avg       0.98      0.98      0.98      1862
```

A function named randomForest is created and train and test data are passed as the parameters. Inside the function, RandomForestClassifier algorithm is initialised and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done.

```
# Fitting Random Forest to the Training set
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()
cv = cross_val_score(dt, X, y, scoring='f1', cv=10)
print('--- Average F1-Score for Decision Tree model: {} ---'.format(round(cv.mean(), 3)))
print('Standard Deviation: {}'.format(round(cv.std(), 3)))
```

```
--- Average F1-Score for Decision Tree model: 0.98 ---
Standard Deviation: 0.005
```

Activity 1.3: Naïve Bayes model

A function named MultinomialNB is created and train and test data are passed as the parameters. Inside the function, MultinomialNB algorithm is initialised and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable. For evaluating the model, confusion matrix and classification report is done.

```
# Fitting Naive Bayes to the Training set
from sklearn.naive_bayes import MultinomialNB
mnb = MultinomialNB()
cv = cross_val_score(mnb, X, y, scoring='f1', cv=10)
print('--- Average F1-Score for MNB model: {} ---'.format(round(cv.mean(), 3)))
print('Standard Deviation: {}'.format(round(cv.std(), 3)))
```

```
--- Average F1-Score for MNB model: 0.944 ---
Standard Deviation: 0.004
```

```
# Classification report for MNB model
mnb = MultinomialNB()
mnb.fit(X_train, y_train)
y_pred = mnb.predict(X_test)

print('--- Classification report for MNB model ---')
print(classification_report(y_test, y_pred))
```

```
--- Classification report for MNB model ---
              precision    recall  f1-score   support

           0       0.94      0.95      0.94       958
           1       0.94      0.94      0.94       904

    accuracy                           0.94      1862
   macro avg       0.94      0.94      0.94      1862
weighted avg       0.94      0.94      0.94      1862
```

Milestone 4: Performance Testing & Hyperparameter Tuning

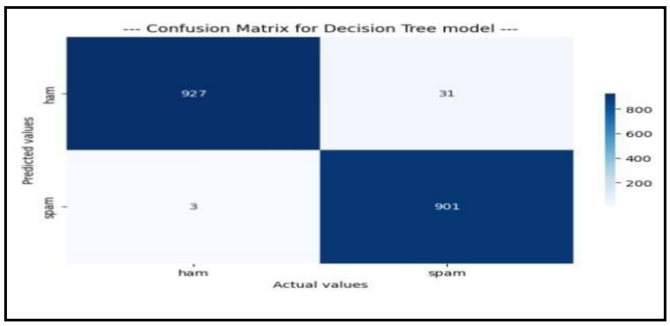
Activity 1: Testing model with multiple evaluation metrics

Multiple evaluation metrics means evaluating the model's performance on a test set using different performance measures. This can provide a more comprehensive understanding of the model's strengths and weaknesses. We are using evaluation metrics for classification tasks including accuracy, precision, recall, support and F1-score.

For comparing the above four models, the compareModel function is defined.

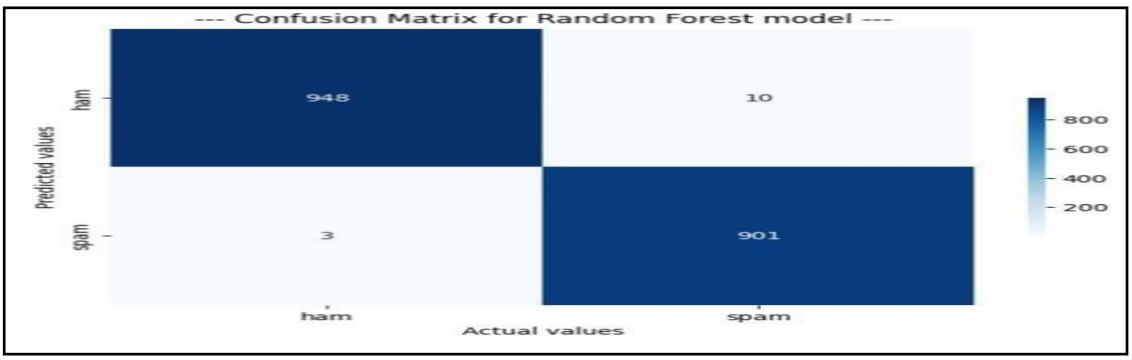
Confusion matrix of decision tree model

```
# Confusion matrix of Decision Tree model
cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(8,5))
axis_labels = ['ham', 'spam']
g = sns.heatmap(data=cm, annot=True, cmap="Blues", xticklabels=axis_labels, yticklabels=axis_labels, fmt='g', cbar_kws={"shrink": 0.5})
p = plt.xlabel('Actual values')
p = plt.ylabel('Predicted values')
p = plt.title('--- Confusion Matrix for Decision Tree model ---')
```



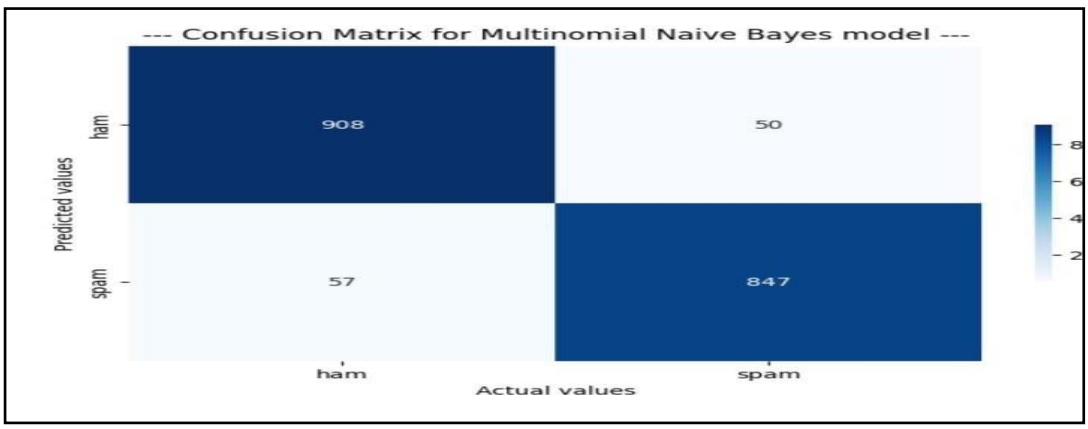
Confusion matrix of random forest model

```
# Confusion matrix of Random Forest model
cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(8,5))
axis_labels = ['ham', 'spam']
g = sns.heatmap(data=cm, annot=True, cmap="Blues", xticklabels=axis_labels, yticklabels=axis_labels, fmt='g', cbar_kws={"shrink": 0.5})
p = plt.xlabel('Actual values')
p = plt.ylabel('Predicted values')
p = plt.title('--- Confusion Matrix for Random Forest model ---')
```



Confusion matrix of multinomial naive bayes model

```
# Confusion matrix of MNB model
cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(8,5))
axis_labels = ['ham', 'spam']
g = sns.heatmap(data=cm, annot=True, cmap="Blues", xticklabels=axis_labels, yticklabels=axis_labels, fmt='g', cbar_kws={"shrink": 0.5})
p = plt.xlabel('Actual values')
p = plt.ylabel('Predicted values')
p = plt.title('--- Confusion Matrix for Multinomial Naive Bayes model ---')
```

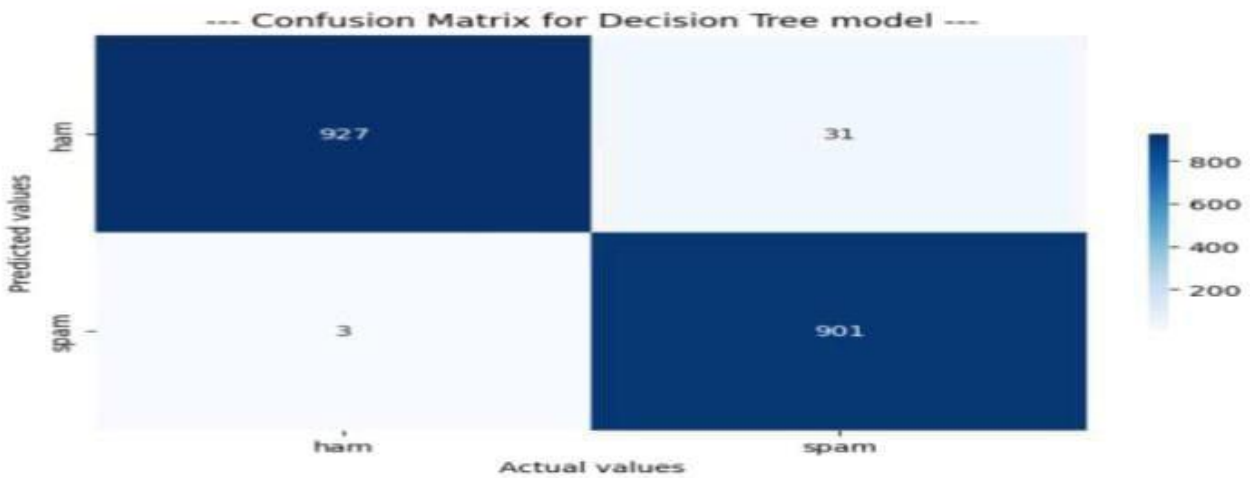


MILESTONE 5: MODEL DEPLOYMENT

Activity 1: Save the best model

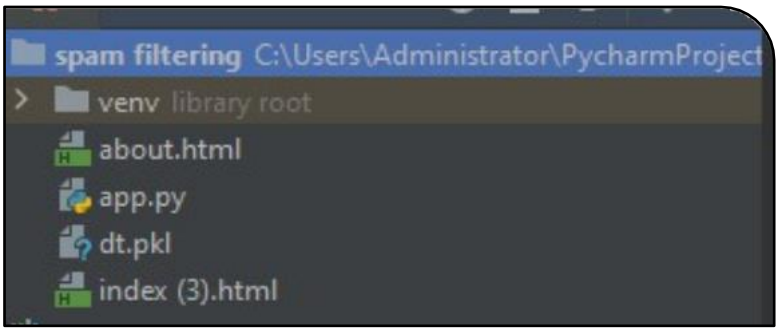
Saving the best model after comparing its performance using different evaluation metrics means selecting the model with the highest performance and saving its weights and configuration. This can be useful in avoiding the need to retrain the model every time it is needed and also to be able to use it in the future.

```
# Confusion matrix of Decision Tree model
cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(8,5))
axis_labels = ['ham', 'spam']
g = sns.heatmap(data=cm, annot=True, cmap="Blues", xticklabels=axis_labe
p = plt.xlabel('Actual values')
p = plt.ylabel('Predicted values')
p = plt.title('--- Confusion Matrix for Decision Tree model ---')
```




```
pickle.dump(rf,open("modelrf.pk1","wb"))
```

Activity 2:Integrate with web frame work



- Build the html pages.
- Run the web application.