OPTIMIZING SPAM FILTERING WITH MACHINE LEARNING

TEAM MEMBERS' NAME : K.MUNIYAMMAL

D.DURGADEVI

V.JESSIVIOLET

K.NISHA

CLASS : III-BSC COMPUTER SCIENCE

TABLE OF INDEX

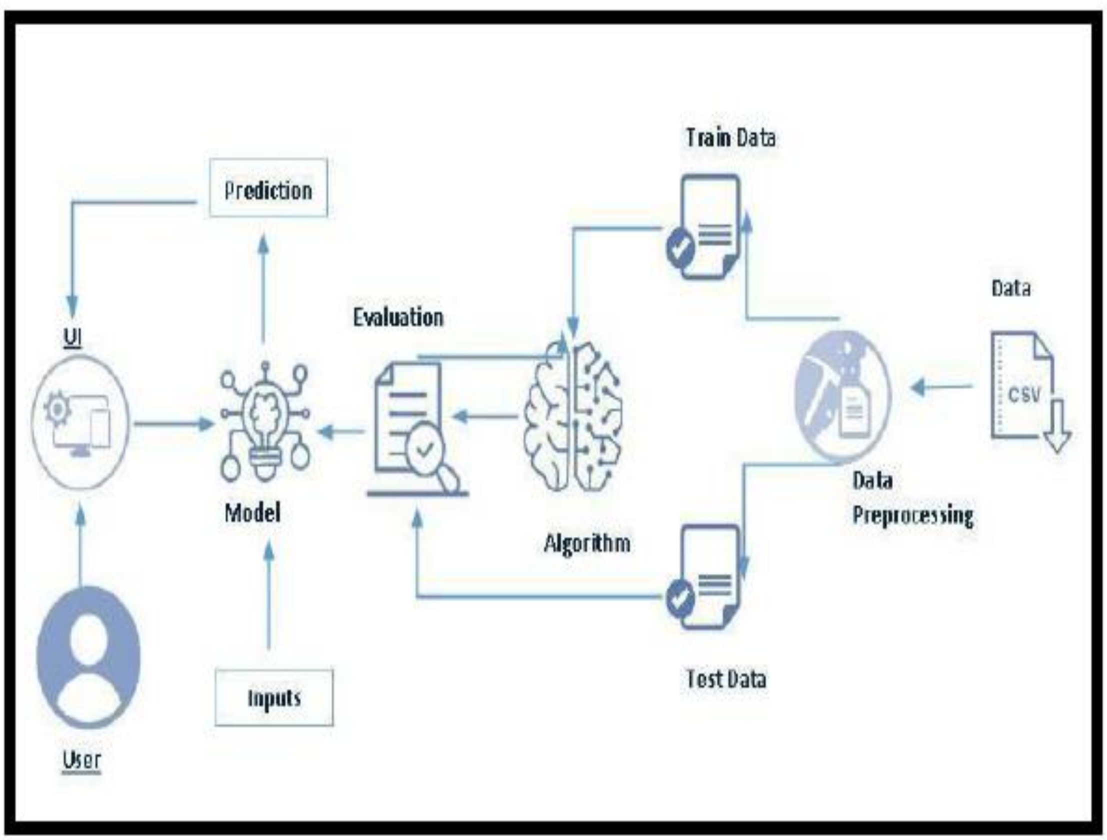
1.INTRODUCTION

OPTIMIZING SPAM FILTERING WITH MACHINE LEARNING :

Over recent years, as the popularity of mobile phone devices has increased, Short Message Service (SMS) has grown into a multi-billion dollar industry. At the same time, reduction in the cost of messaging services has resulted in growth in unsolicited commercial advertisements (spams) being sent to mobile phones. Due to Spam SMS, Mobile service providers suffer from some sort of financial problems as well as it reduces calling time for users. Unfortunately, if the user accesses such Spam SMS they may face the problem of virus or malware. When SMS arrives at mobile it will disturb mobile user privacy and concentration. It may lead to frustration for the user. So Spam SMS is one of the major issues in the wireless communication world and it grows day by day.

To avoid such Spam SMS people use white and black list of numbers. But this technique is not adequate to completely avoid Spam SMS. To tackle this problem it is needful to use a smarter technique which correctly identifies Spam SMS. Natural language processing technique is useful for Spam SMS identification. It analyses text content and finds patterns which are used to identify Spam and Non-Spam SMS.

Technical Architecture:

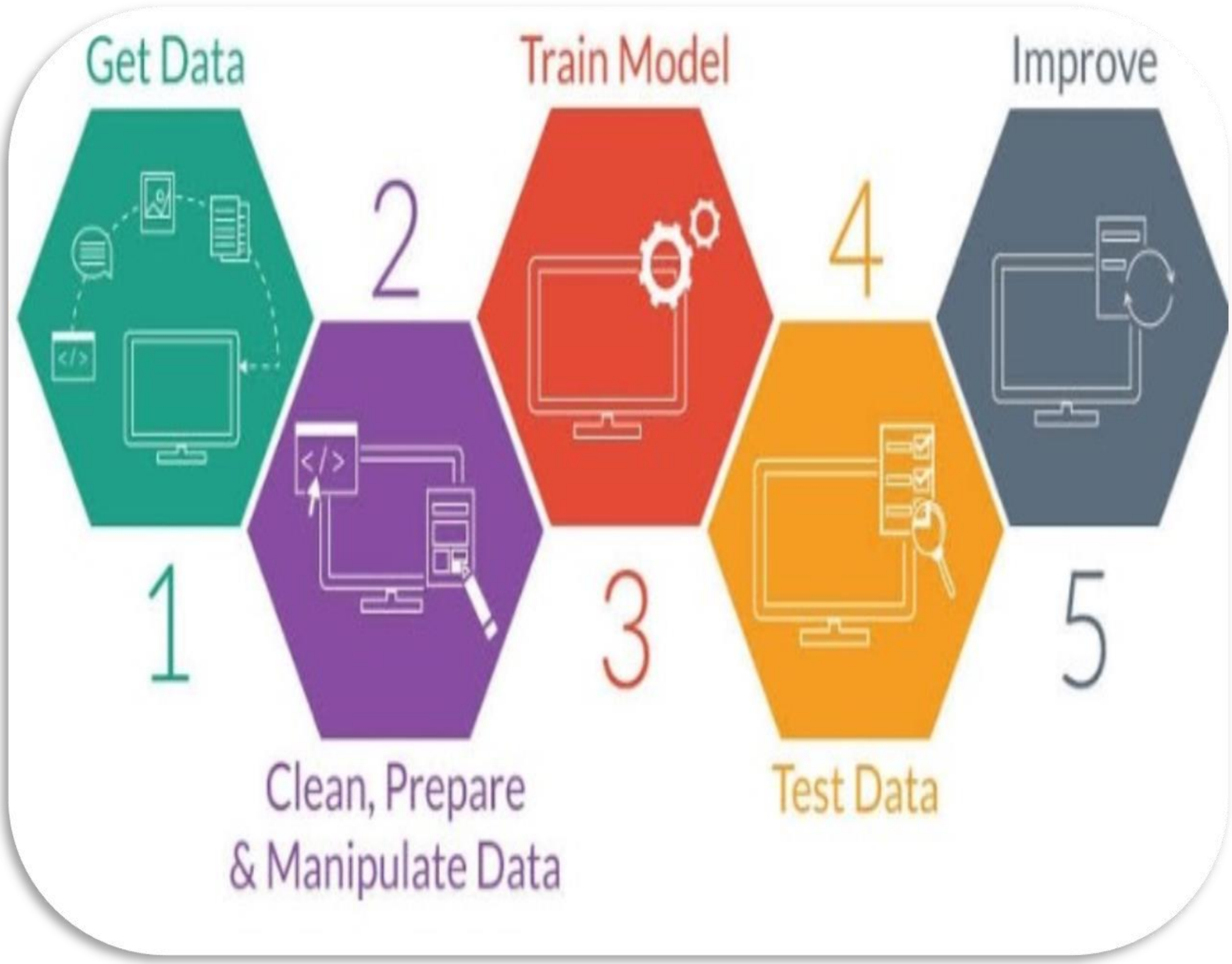


1.1.OVERVIEW

- 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



1.2.PURPOSE

Thus consuming time and resources, Machine learning makes it easier because it learns to recognize the unsolicited spam and legitimate ham automatically and then applies those learned instructions to unknown incoming spams.

2.PROBLEM DEFINITION & DESIGN THINKING

Over recent years, as the popularity of mobile phone devices has increased, Short Message Service (SMS) has grown into a multi-billion dollar industry. At the same time, reduction in the cost of messaging services has resulted in growth in unsolicited commercial advertisements (spams) being sent to mobile phones. Due to Spam SMS, Mobile service providers suffer from some sort of financial problems as well as it reduces calling time for users. Unfortunately, if the user accesses such Spam SMS they may face the problem of virus or malware. When SMS arrives at mobile it will disturb mobile user privacy and concentration. It may lead to frustration for the user. So Spam SMS is one of the major issues in the wireless communication world and it grows day by day.

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 is the initial step in building a machine learning model which aims to understand the need for it in the organisation. The machine learning development process can be resource intensive, so clear objectives should be agreed and set at the start. Clearly define the problem that a model needs to solve and what success looks like. A deployed model will bring much more value if it’s fully aligned with the objectives of the organisation. Before the project begins, there are key elements that need to be explored and planned.

2.1 EMPATHY MAP

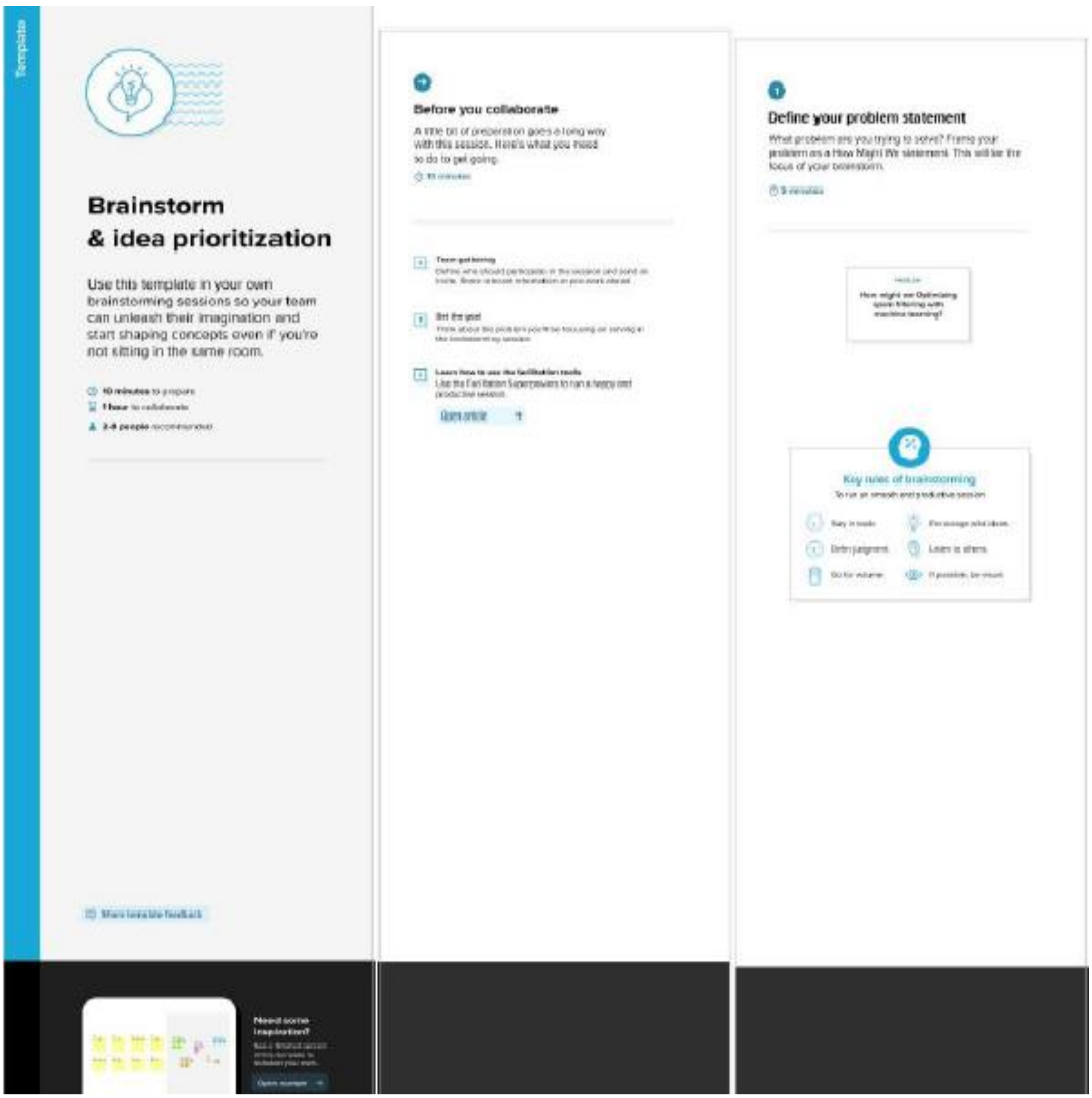
In the ideation phase we have empathized as our client Optimizing spam filtering with machine learning and we have acquired the details which are represented in the Empathy Map given below.

WHO are we empathizing with?
Who is the person we want to understand?
What is the situation they are in?
What is their role in the situation?
GOAL
Convenience
Quality
Compare the cost
What do they need to DO?
What do they need to do differently?
What job(s) do they want or need to get done?
What decision(s) do they need to make?
How will we know they were successful?
I'll give it a try
I need more flexible software
What do they HEAR?
What are they hearing others say?
What are they hearing from friends?
What are they hearing from colleagues?
What are they hearing second-hand?
It's too difficult
It has many process
What do they THINK and FEEL?
PAINS
What are their fears, frustrations, and anxieties?
GAINS
What are their wants, needs, hopes, and dreams?
Too expensive
Unsure whom to trust
Easy to identify the unwanted emails
Easy to access
I want to see the spam mail in individual menu
What do they SEE?
What do they see in the marketplace?
What do they see in their immediate environment?
What do they see others saying and doing?
What are they watching and reading?
What other thoughts and feelings might influence their behavior?
Not smart enough
Impatient
What do they SAY?
What have we heard them say?
What can we imagine them saying?
I love the new features
Buy the Proper equipments
What do they DO?
What do they do today?
What behavior have we observed?
What can we imagine them doing?
I don't spent a lot of money
I have the software at affordable prices.

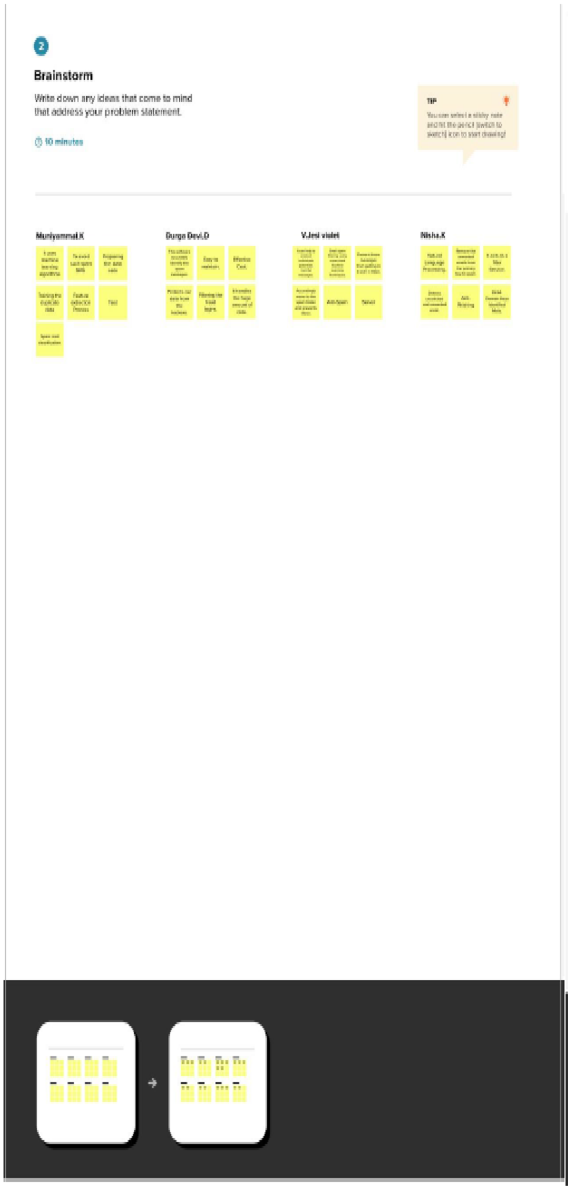
2.2 IDEATION & BRAINSTORMING MAP

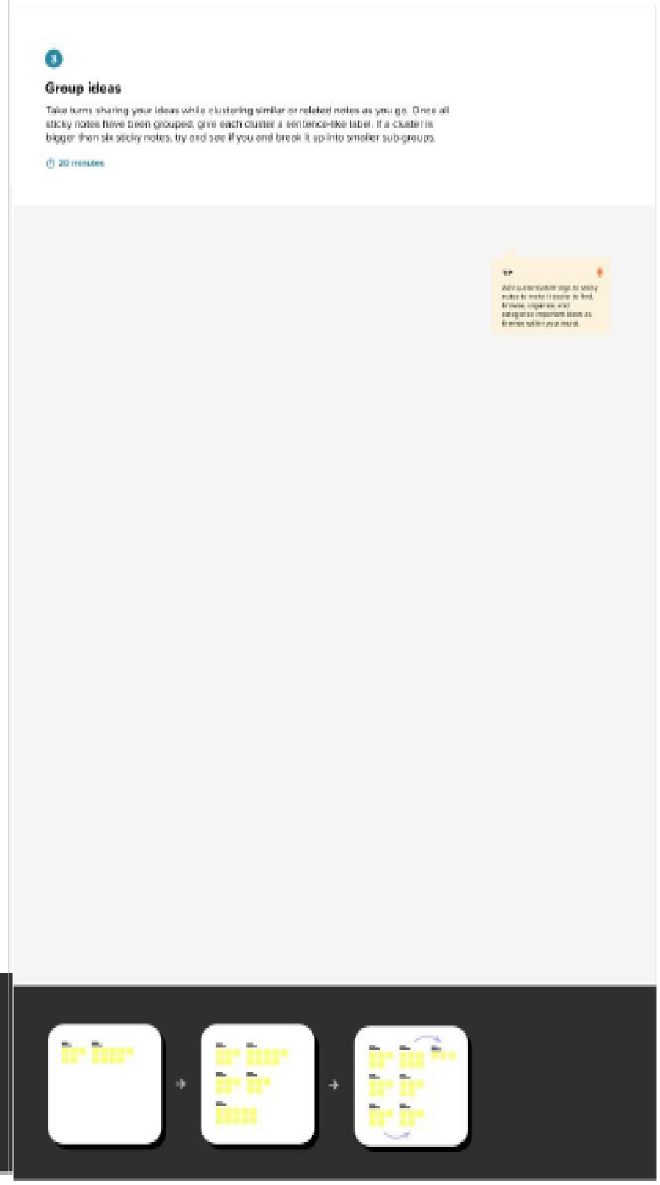
Under this activity our team members have gathered and discussed various idea to solve our project problem. Each member contributed 6 to 10 ideas after gathering all ideas we have assessed the impact and feasibility of each point. Finally, we have assign the priority for each point based on the impact value.

STEP 1:Team Gathering, collaboration and Select the Problem

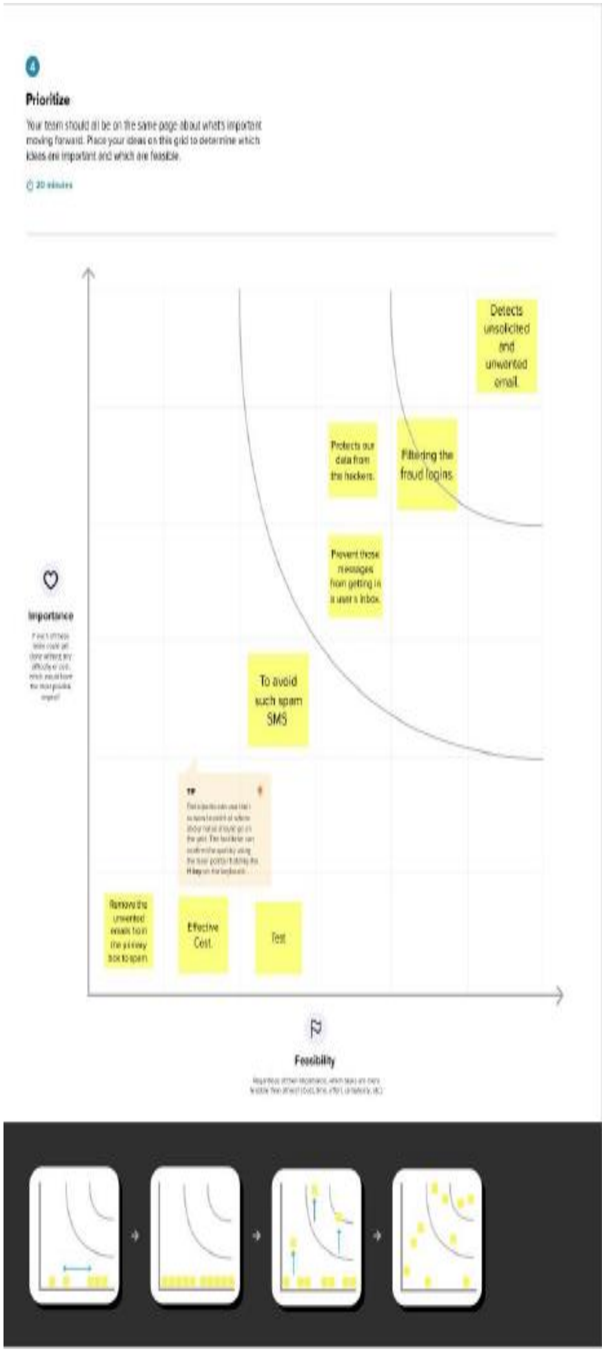


STEP-2: Brainstorm, Idea Listing and Grouping





STEP-3: Idea Prioritization



3.RESULT

Read the datasets

	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

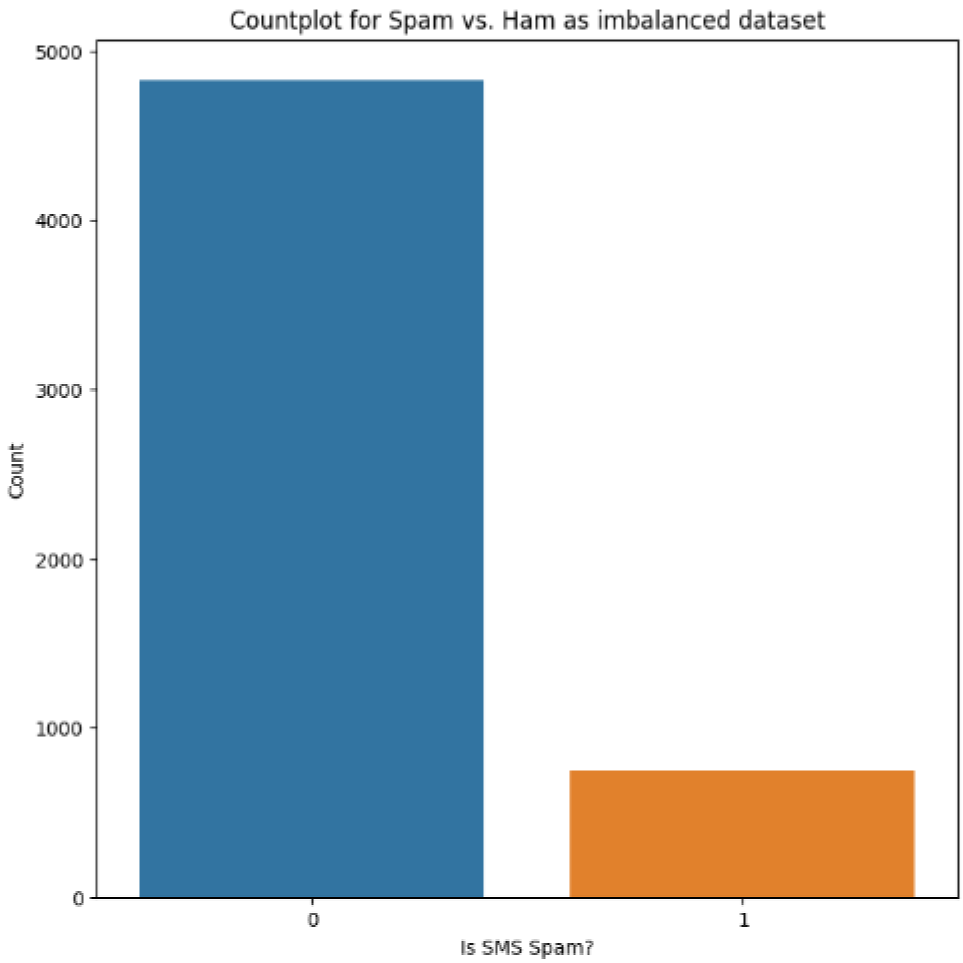
Handling missing values

```
<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
```

```
label                     0
msg                       0
word_count                0
contains_currency_symbol  0
dtype: int64
```

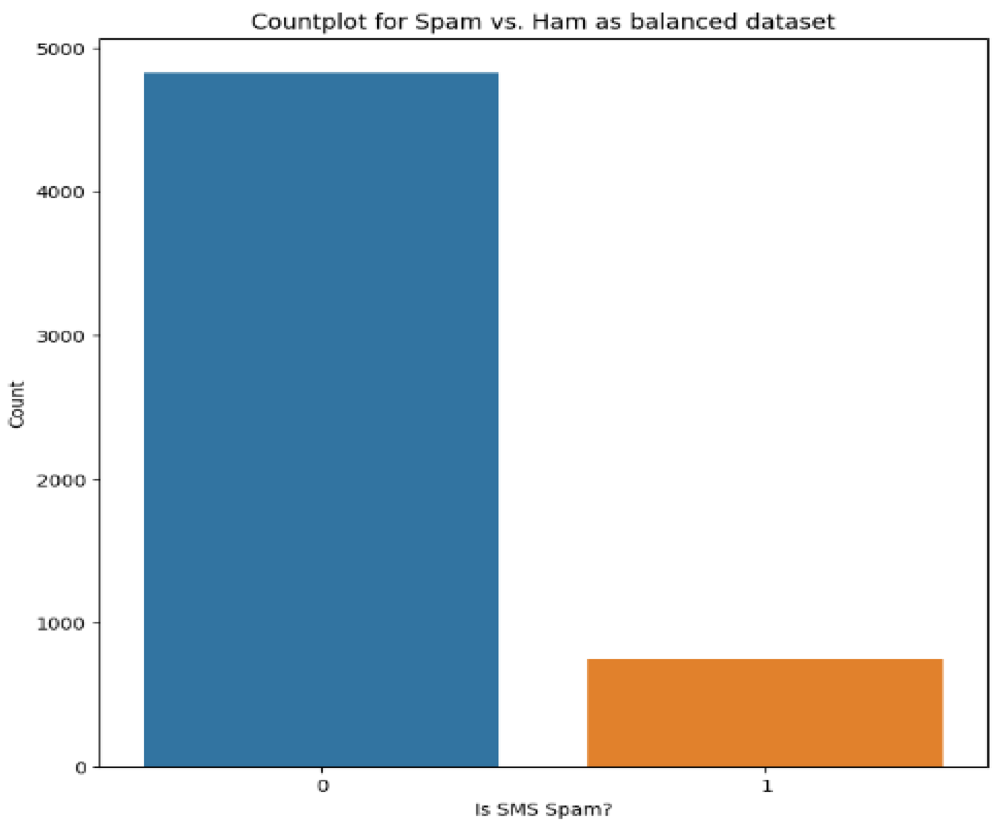
	label	msg
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

	label	msg
5567	spam	This is the 2nd time we have tried 2 contact u...
5568	ham	Will Ì_ b going to esplanade fr home?
5569	ham	Pity, * was in mood for that. So...any other s...
5570	ham	The guy did some bitching but I acted like i'd...
5571	ham	Rofl. Its true to its name

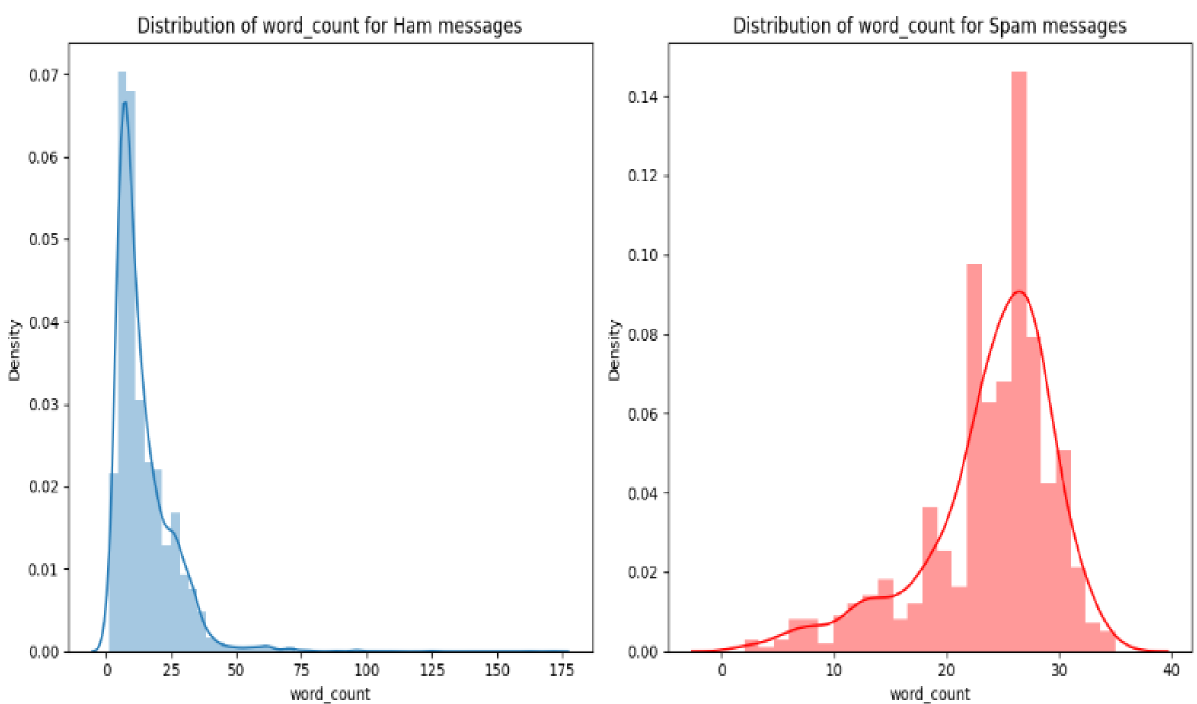


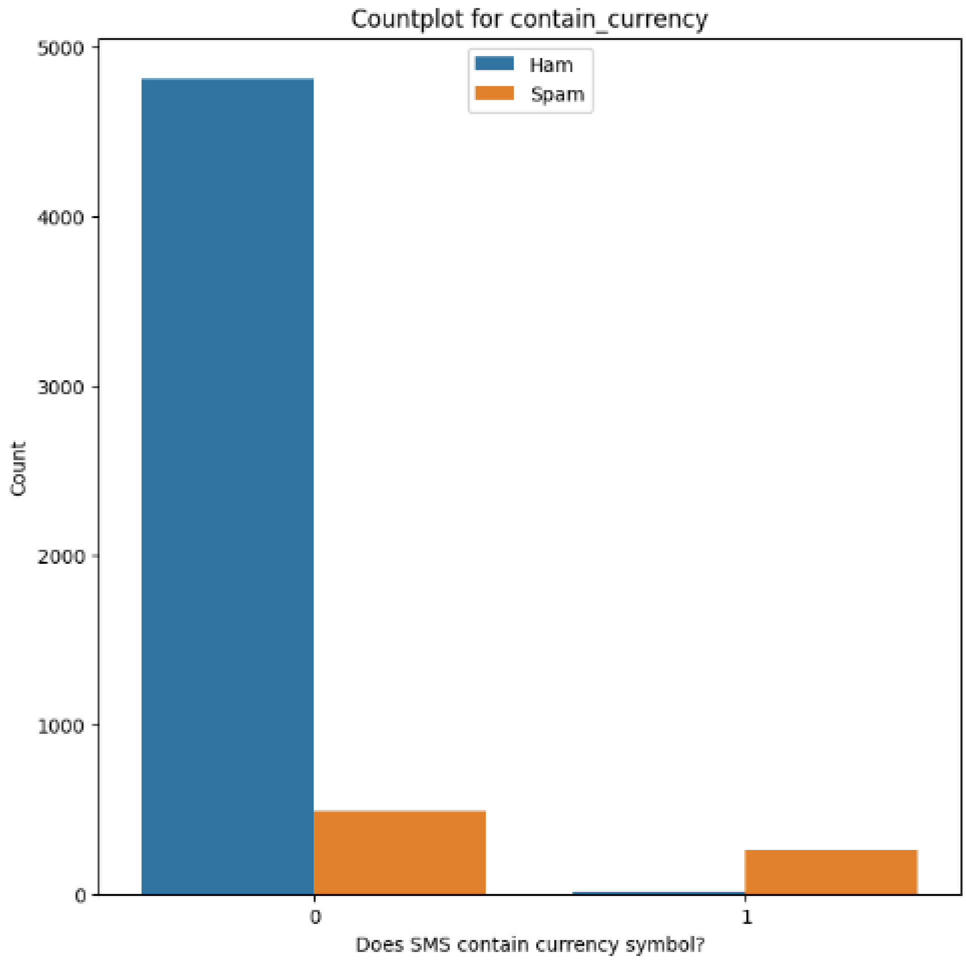

```
Number of Spam records: 747
Number of Ham records: 4825
```

(9307, 5)

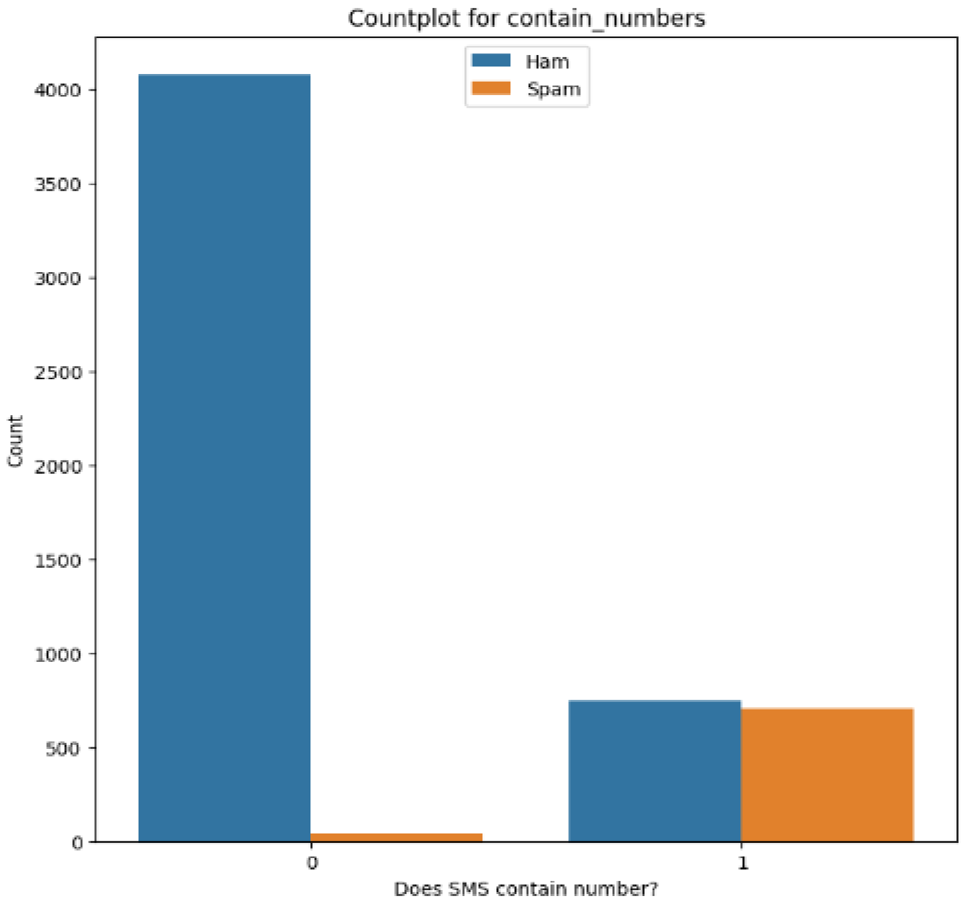

Countplot for Spam vs. Ham as balanced dataset
5000
4000
3000
2000
1000
0
Count
0
1
Is SMS Spam?

```
<ipython-input-44-6776e7c342cf>: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-44-6776e7c342cf>: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')
```





Countplot for contain_currency
Ham
Spam
Count
5000
4000
3000
2000
1000
0
0
1
Does SMS contain currency symbol?



Clear the text data

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
```

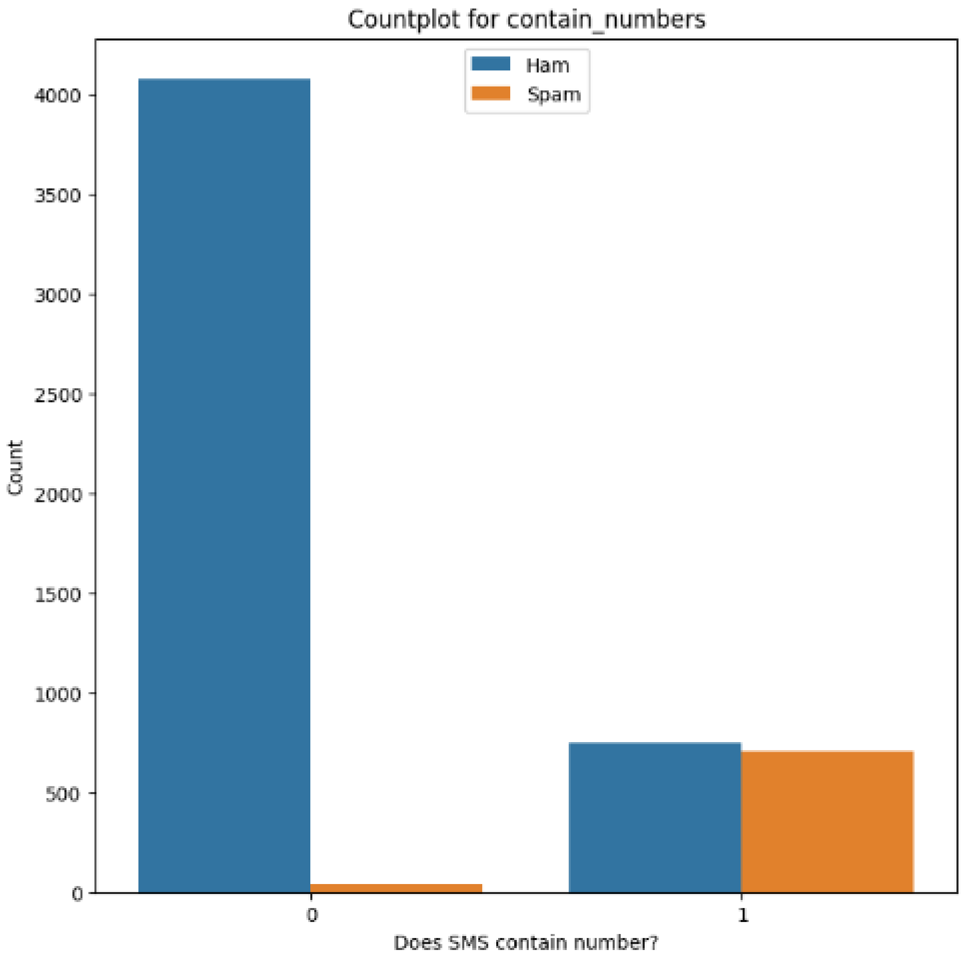
Exploratory Data Analysis

Descriptive statistical

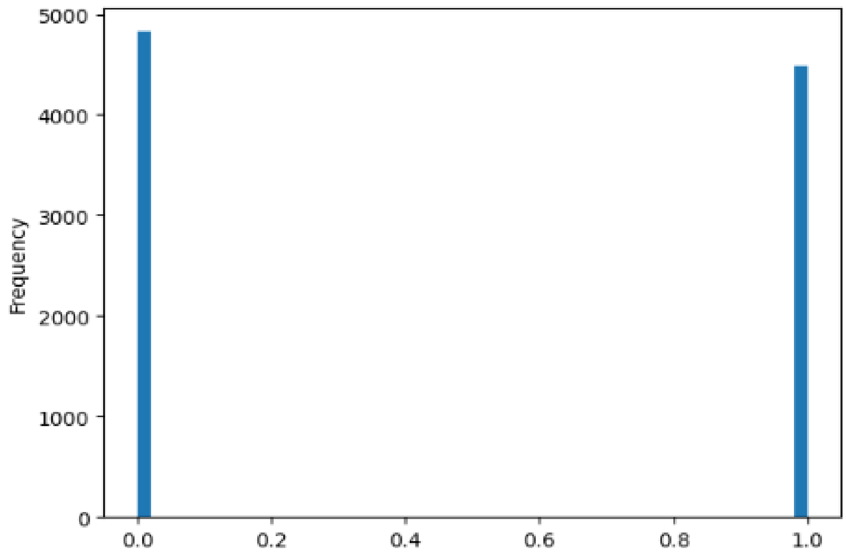
	label	msg
count	5572	5572
unique	2	5169
top	ham	Sorry, I'll call later
freq	4825	30

```
(5572, 2)
```

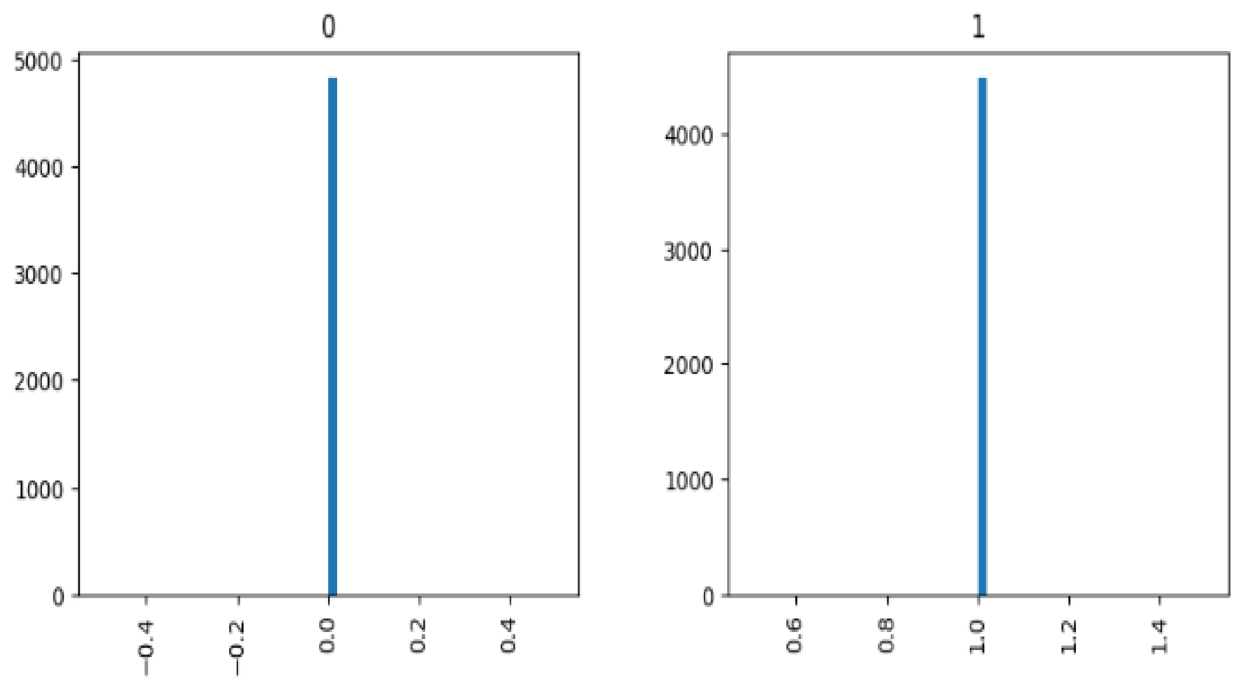
Visual Analysis



```
<Axes: ylabel='Frequency'>
```




```
array([<Axes: title={'center': '0'}>, <Axes: title={'center': '1'}>],
      dtype=object)
```



MODEL BUILDING

Naïve Bayes model

```
--- Average F1-Score for MNB model: 0.898 ---
Standard Deviation: 0.014
```

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

           0       0.97      1.00      0.98       965
           1       0.97      0.81      0.88       150

    accuracy                           0.97      1115
   macro avg       0.97      0.90      0.93      1115
weighted avg       0.97      0.97      0.97      1115
```

Decision Tree Model

```
--- Average F1-Score for Decision Tree model: 0.865 ---
Standard Deviation: 0.032
```

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

           0       0.97      0.98      0.98       965
           1       0.88      0.81      0.85       150

    accuracy                           0.96      1115
   macro avg       0.93      0.90      0.91      1115
weighted avg       0.96      0.96      0.96      1115
```

Random Forest Model

```
--- Average F1-Score for Random Forest model: 0.912 ---
Standard Deviation: 0.016
```

```
--- Classification report for Random Forest model ---
              precision    recall  f1-score   support

           0       0.98      1.00      0.99       965
           1       0.97      0.85      0.91       150

    accuracy                           0.98      1115
   macro avg       0.97      0.92      0.95      1115
weighted avg       0.98      0.98      0.98      1115
```

Integrate with web FrameWork

Building HTML pages

INDEX.html

```
<!DOCTYPE html>
<html>
<head>
<title>SMS spam detection</title>
</head>
<style>
body
{
background-image:url("sms1.JPG");
background-repeat:no repeat;
}
h1
{
color:navy;
text-decoration:underline;
}
h2
{
color:black;
margin-left:40px;
}
h3
{
color:red;
margin-left:60px;
}
</style>

<h1>
<center>
<b>
<i>
```

```
<font size=15>
 SMS spam detection
</font>
</i>
</b>
</center>
</h1>
<div style="background-color:bisque">
<hr>
<hr></div>
<h2> SMS spam detection</h2>
<h4>

<form action="/getdata" method="post">
 <table>
  <tr>
   <td>msg</td>: <input type="text" name="enter the SMS"
required='required'/><br>
  </tr>
  <tr>

<td><br><br>          &
nbsp    
    <button type="submit" class="btn btn-primary btn-block btn-large"> Predict
</button>
  </td>
  </tr>
 </table>
     
     
</form>
</h4>
<h3>
<b>
{{ prediction_text }}
</b>
```

```
</h3>
</html>
```

Building Python Code

```python
import flask
from flask import Flask, render_template, request
import pickle
import numpy as np
import sklearn
import warnings

warnings.filterwarnings('ignore')

app = Flask(__name__)


model = pickle.load(open('rf.pkl', 'rb'))


@app.route('/')
def home():
    return render_template('index.html')


@app.route('/getdata', methods=['POST'])
def pred():
    MSG = request.form['msg']
```

```
    print(msg)
    inp_features = [[msg]]
    print(inp_features )
    prediction = model.predict(inp_features)
    print(type(prediction))
    t = prediction[0]
    print(t)
    if t > 0.5:
        prediction_text = 'SMS is SPAM'
    else:
        prediction_text = 'SMS is not SPAM'
    print(prediction_text)
    return render_template('prediction.html',prediction_results=prediction_text)

if __name__ == "__main__":
    app.run()
```

Run the Web Application

4.ADVANTAGES & DISADVANTAGES

Advantages:

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

Disadvantages:

- Thousands of spam may raach Inboxes before a spammer's spam address, IP or domain is blacklisted.
- Spam filtering is machine-based on there is a room for mistakes called "false positivies".Bayesian filters may be fooled by spammers,e.g, in a case of usig large blocked of legitimate text.

5.APPLICATIONS

A spam filter is a program used to detect unsolicited, unwanted and virus-infected spam and prevent those messages from to a user's inbox.

Example:

Congratulations, you've won

Verify or update your account

Assist a family member

Mine or claim a cryptocurrency

You have received a scholarship fund

Package delivery

Two-factor authentication

Coronavirus messages.

6.CONCLUSION

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.

We have develop a machine learning model using python programming language and the reports are shown above.

7.FUTURE SCOPE

- It provides sensitivity to the client adapts well to the future spam techniques.
- It considers a complete message instead of single words with respect to its organization.
- It increases Security and Control.
- It reduces IT Administration costs.
- It also reduce network Resources Costs.

8.APPENDIX

SOURCE CODE:

Importing the libraries:

```
import numpy as np

import pandas as pd


import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline
```

Read the Dataset:

```
# load our dataset

df=pd.read_csv("/content/spam.csv",encoding="latin-1")


df.head()
```

Handling missing values:

```
df.info()

# Returns the sum of all na values

df.isna().sum()

 df=df[v1,v2]

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


df.head()

df.tail()
```

Mapping values for label:

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

Handling Imbalance Data:

Countplot for Spam vs.Ham as Imbalanced Data:

```
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')
```

Handling imbalanced dataset using Oversampling:

```
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]))


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
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
df['word_count'] = df['msg'].apply(lambda x: len(x.split()))
df.head()

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()
```

Spam messages word_count fail in the range of 15-30 words, where as majority of the Ham messages fail in the range of below 25 words.

```
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()



plt.figure(figsize=(8,8))

g = sns.countplot(x='contains_currency_symbol', data=df, hue='label')

p = plt.title('Countplot for contain_currency')

p = plt.xlabel('Does SMS contain currency symbol?')

p = plt.ylabel('Count')

p = plt.legend(labels=['Ham', 'Spam'], loc=9)
```

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()

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)
```

Cleaning the text data:

```
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


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)
```

```
# 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']
```

Exploratory Data Analysis

Descriptive statistical

```
df.describe(include='O')
df.shape
```

Visual analysis

```
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)


import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

df['label'].plot(bins=50, kind='hist')


df.hist(column='label', by='label', bins=50,figsize=(10,4))
```

Scaling the Data

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

Model Building

Naïve Bayes model

```
# 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)))


# 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))



# 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 ---')
```

Decision Tree Model

```
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)))
```

```
# 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))


# 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 ---')
```

Rendom Forest Model

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


# Classification report for Random Forest model
```

```
rf = RandomForestClassifier(n_estimators=20)

rf.fit(X_train, y_train)

y_pred = rf.predict(X_test)


print('--- Classification report for Random Forest model ---')

print(classification_report(y_test, y_pred))


# 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 ---')
```