

## Team No:-PNT2022TMID08666

### Assignment 4

Student Name	K.Nithyasree
Student Roll Number	19BCS023
Maximum Marks	2 Marks

#### Problem Statement :- SMS SPAM Classification

Problem Statement: 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.

#### Screenshots:-

```
[90] !pip install pyforest

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: pyforest in /usr/local/lib/python3.7/dist-packages (1.1.0)

[2] from pyforest import *

[6] import numpy as np
import pandas as pd
import pickle

[13] df=pd.read_csv('spam.csv',encoding="ISO-8859-1")
df.head(10)
```

	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
5	spam	FreeMsg Hey there darling It's been 3 week's n...	NaN	NaN	NaN

```

[13]
0s
6 ham Even my brother is not like to speak with me. ... NaN NaN NaN
7 ham As per your request 'Melle Melle (Oru Minnamin... NaN NaN NaN
8 spam WINNER!! As a valued network customer you have... NaN NaN NaN
9 spam Had your mobile 11 months or more? U R entitle... NaN NaN NaN

```

```

[14] df.shape
0s
(5572, 5)

```

```

[15] df.info()
0s
<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

```

```

[16] df.drop(columns=['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'], inplace=True)
0s

```

```

[17] df.head()
0s

```

	v1	v2
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...

```

[18] df.rename(columns={'v1': 'target', 'v2': 'text'}, inplace=True)
0s

```

```

[19] from sklearn.preprocessing import LabelEncoder
1s
encoder = LabelEncoder()

```

```

[20] df['target'] = encoder.fit_transform(df['target'])
0s

```

```

[21] df.head()
0s

```

	target	text
0	0	Go until jurong point, crazy.. Available only ...

```
✓ [21] 1      0      Ok lar... Joking wif u oni...
0s
      2      1  Free entry in 2 a wkly comp to win FA Cup fina...
      3      0  U dun say so early hor... U c already then say...
      4      0  Nah I don't think he goes to usf, he lives aro...
```

```
✓ [22] df.isnull().sum()
0s
target      0
text        0
dtype: int64
```

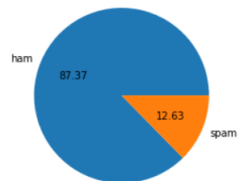
```
✓ [23] df.duplicated().sum()
0s
403
```

```
✓ [24] df=df.drop_duplicates(keep='first')
0s
df.duplicated().sum()
0
```

```
✓ [25] df.shape
0s
(5169, 2)
```

```
✓ [26] df['target'].value_counts()
0s
```

```
✓ [79] plt.pie(df['target'].value_counts(), labels=['ham', 'spam'], autopct="%0.2f")
0s
plt.show()
print("Team No:- PNT2022TMID08666")
```



Team No:- PNT2022TMID08666

```
✓ [28] import nltk
0s
%pip install nltk
nltk.download('punkt')
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: nltk in /usr/local/lib/python3.7/dist-packages (3.7)
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (from nltk) (4.64.1)
Requirement already satisfied: click in /usr/local/lib/python3.7/dist-packages (from nltk) (7.1.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.7/dist-packages (from nltk) (2022.6.2)
Requirement already satisfied: joblib in /usr/local/lib/python3.7/dist-packages (from nltk) (1.2.0)
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data] Unzipping tokenizers/punkt.zip.
```

```
✓ [29] df['num_characters'] = df['text'].apply(len)
      0s df.head()
```

	target	text	num_characters
0	0	Go until Jurong point, crazy.. Available only ...	111
1	0	Ok lar... Joking wif u oni...	29
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	155
3	0	U dun say so early hor... U c already then say...	49
4	0	Nah I don't think he goes to usf, he lives aro...	61

```
✓ [30] df['num_words'] = df['text'].apply(lambda x:len(nltk.word_tokenize(x)))
      1s df.head()
```

	target	text	num_characters	num_words
0	0	Go until Jurong point, crazy.. Available only ...	111	24
1	0	Ok lar... Joking wif u oni...	29	8
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	155	37
3	0	U dun say so early hor... U c already then say...	49	13
4	0	Nah I don't think he goes to usf, he lives aro...	61	15

```
✓ [31] df['num_sentences'] = df['text'].apply(lambda x:len(nltk.sent_tokenize(x)))
      0s df[['num_characters', 'num_words', 'num_sentences']].describe()
```

	num_characters	num_words	num_sentences
count	5169.000000	5169.000000	5169.000000
mean	78.977945	18.453279	1.947185
std	58.236293	13.324793	1.362406
min	2.000000	1.000000	1.000000
25%	36.000000	9.000000	1.000000
50%	60.000000	15.000000	1.000000
75%	117.000000	26.000000	2.000000
max	910.000000	220.000000	28.000000

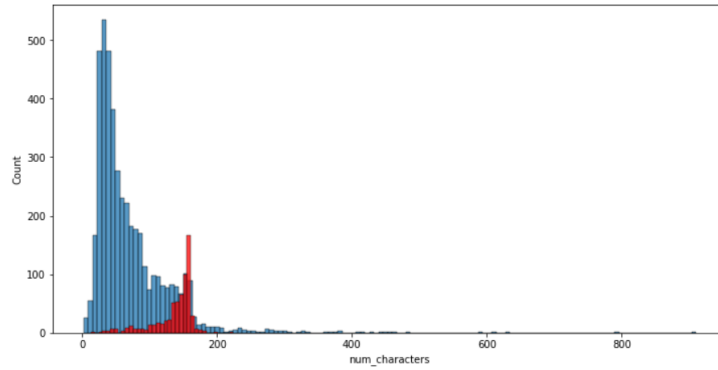
```
✓ [32] df[df['target'] == 0][['num_characters', 'num_words', 'num_sentences']].describe()
      0s
```

	num_characters	num_words	num_sentences
count	4516.000000	4516.000000	4516.000000
mean	70.459256	17.120903	1.799601
std	56.358207	13.493725	1.278465
min	2.000000	1.000000	1.000000

```
✓ [33] 75%    157.000000    32.000000    4.000000
1s      max    224.000000    46.000000    8.000000
```

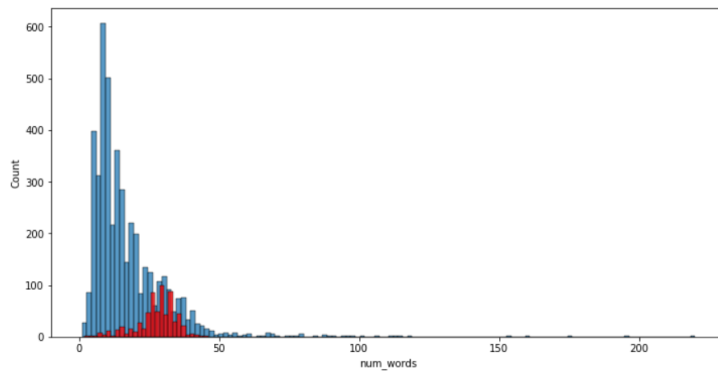
```
✓ [80] plt.figure(figsize=(12,6))
1s      sns.histplot(df[df['target'] == 0]['num_characters'])
      sns.histplot(df[df['target'] == 1]['num_characters'],color='red')
      print("Team No:- PNT2022TMID08666")
```

Team No:- PNT2022TMID08666



```
✓ [81] plt.figure(figsize=(12,6))
1s      sns.histplot(df[df['target'] == 0]['num_words'])
      sns.histplot(df[df['target'] == 1]['num_words'],color='red')
      print("Team No:- PNT2022TMID08666")
```

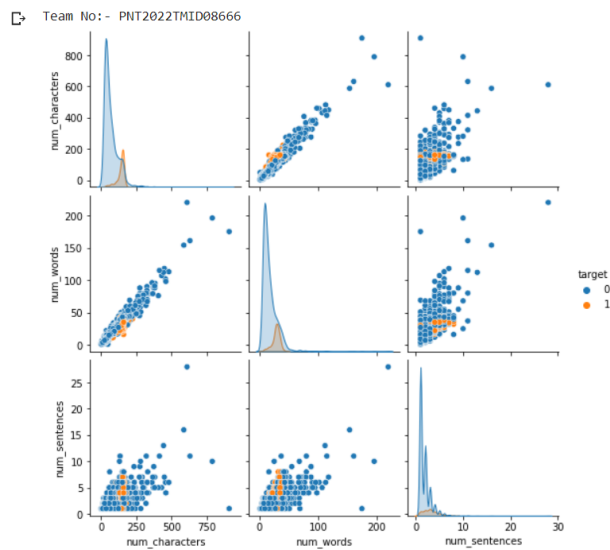
Team No:- PNT2022TMID08666



```

7s ✓ sns.pairplot(df,hue='target')
print("Team No:- PNT2022TMID08666")

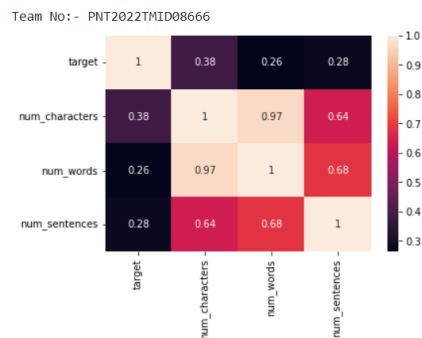
```



```

0s ✓ [83] sns.heatmap(df.corr(),annot=True)
print("Team No:- PNT2022TMID08666")

```



```

0s ✓ [38] from nltk.corpus import stopwords
import string
string.punctuation
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()

```

```

0s ✓ [41] df['text'][100]

```

```
'Okay name ur price as long as its legal! Wen can I pick them up? Y u ave x ams xx'
```

```
[46] import nltk
      nltk.download('stopwords')
```

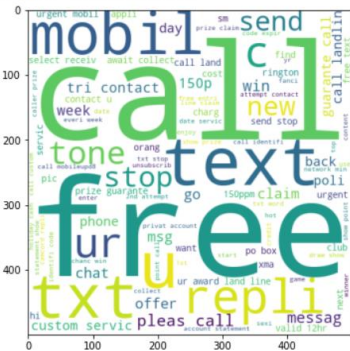
```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
True
```

```
[47] df['transformed_text'] = df['text'].apply(transform_text)
df.head()
```

	target	text	num_characters	num_words	num_sentences	transformed_text
0	0	Go until jurong point, crazy.. Available only ...	111	24	2	go jurong point crazy avail bugi n great world...
1	0	OK lar... Joking wif u oni...	29	8	2	ok lar joke wif u oni
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	155	37	2	free entri 2 wkli comp win fa cup final tkt 21...
3	0	U dun say so early hor... U c already then say...	49	13	1	u dun say earli hor u c already say
4	0	Nah I don't think he goes to usf, he lives aro...	61	15	1	nah think goe usf live around though

```
[84] from wordcloud import WordCloud
wc = WordCloud(width=500,height=500,min_font_size=10,background_color='white')
spam_wc = wc.generate(df[df['target'] == 1]['transformed_text'].str.cat(sep=" "))
plt.figure(figsize=(15,6))
plt.imshow(spam_wc)
print("Team No:- PNT2022THID08666")
```

[84] Team No:- PNT2022TMID08666



```
[85] ham_wc = wc.generate(df[df['target'] == 0]['transformed_text'].str.cat(sep=" "))
plt.figure(figsize=(15,6))
plt.imshow(ham_wc)
print("Team No:- PNT2022TMDI08666")
```

Team No:- PNT2022TMID08666



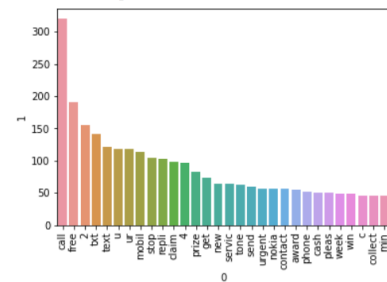
```
[50] # For Spam messages
spam_corpus = []
for msg in df[df['target'] == 1]['transformed_text'].tolist():
    for word in msg.split():
        spam_corpus.append(word)

len(spam_corpus)
```

9939

```
[86] from collections import Counter
sns.barplot(pd.DataFrame(Counter(spam_corpus).most_common(30))[0],pd.DataFrame(Counter(spam_corpus).most_common(30))[1])
plt.xticks(rotation='vertical')
plt.show()
print("Team No:- PNT2022TMID08666")
```

/usr/local/lib/python3.7/dist-packages/seaborn/\_decorators.py:43: FutureWarning: Pass the following variables as keyword args: x, y. From version (



Team No:- PNT2022TMID08666

```
[52] # For ham messages
ham_corpus = []
for msg in df[df['target'] == 0]['transformed_text'].tolist():
    for word in msg.split():
        ham_corpus.append(word)

len(ham_corpus)
```

35394

```
[53] from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer
cv = CountVectorizer()
tfidf = TfidfVectorizer(max_features=3000)
X = tfidf.fit_transform(df['transformed_text']).toarray()
X.shape
y = df['target'].values
from sklearn.model_selection import train_test_split
```

```
[54] X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=2)
from sklearn.naive_bayes import GaussianNB,MultinomialNB,BernoulliNB
from sklearn.metrics import accuracy_score,confusion_matrix,precision_score
mnb = MultinomialNB()
# MultinomialNB
mnb.fit(X_train,y_train)
y_pred1 = mnb.predict(X_test)
print(accuracy_score(y_test,y_pred1))
print(confusion_matrix(y_test,y_pred1))
print(precision_score(y_test,y_pred1))
```

0.9709864603481625

[[896 0]

[ 30 108]]

1.0



```
[55] from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.ensemble import GradientBoostingClassifier
from xgboost import XGBClassifier
```

```
[56] svc = SVC(kernel='sigmoid', gamma=1.0)
knc = KNeighborsClassifier()
mnb = MultinomialNB()
dtc = DecisionTreeClassifier(max_depth=5)
lrc = LogisticRegression(solver='liblinear', penalty='l1')
rfc = RandomForestClassifier(n_estimators=50, random_state=2)
abc = AdaBoostClassifier(n_estimators=50, random_state=2)
bc = BaggingClassifier(n_estimators=50, random_state=2)
etc = ExtraTreesClassifier(n_estimators=50, random_state=2)
gbdt = GradientBoostingClassifier(n_estimators=50, random_state=2)
xgb = XGBClassifier(n_estimators=50, random_state=2)
```

```
[57] clfs = {
    'SVC': svc,
    'KN': knc,
    'NB': mnb,
    'DT': dtc,
    'LR': lrc,
    'RF': rfc,
    'AdaBoost': abc,
    'BgC': bc,
```

```
✓ [58] train_classifier(svc,X_train,y_train,X_test,y_test)
99 (0.9758220502901354, 0.9747899159663865)
```

```
✓ 2m ▶ accuracy_scores = []
precision_scores = []

for name,clf in clfs.items():

    current_accuracy,current_precision = train_classifier(clf, X_train,y_train,X_test,y_test)

    print("For ",name)
    print("Accuracy - ",current_accuracy)
    print("Precision - ",current_precision)

    accuracy_scores.append(current_accuracy)
    precision_scores.append(current_precision)
```

```
📄 For SVC
Accuracy - 0.9758220502901354
Precision - 0.9747899159663865
For KN
Accuracy - 0.9052224371373307
Precision - 1.0
For NB
Accuracy - 0.9709864603481625
Precision - 1.0
For DT
Accuracy - 0.9332688588007737
Precision - 0.8415841584158416
For LR
Accuracy - 0.9584139264990329
Precision - 0.9702970297029703
For RF
Accuracy - 0.9748549323017408
```

```
✓ [64] Accuracy - 0.9574468085106383
2m Precision - 0.8671875
For ETC
Accuracy - 0.9748549323017408
Precision - 0.9745762711864406
For GBDT
Accuracy - 0.9477756286266924
Precision - 0.92
For xgb
Accuracy - 0.9439071566731141
Precision - 0.9347826086956522
```

```
✓ [65] performance_df = pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy':accuracy_scores,'Precision':precision_scores}).sort_values('Precision',ascending=
0s performance_df
```

	Algorithm	Accuracy	Precision
1	KN	0.905222	1.000000
2	NB	0.970986	1.000000
5	RF	0.974855	0.982759
0	SVC	0.975822	0.974790
8	ETC	0.974855	0.974576
4	LR	0.958414	0.970297
10	xgb	0.943907	0.934783
6	AdaBoost	0.960348	0.929204
9	GBDT	0.947776	0.920000
7	BgC	0.957447	0.867188
3	DT	0.933269	0.841584

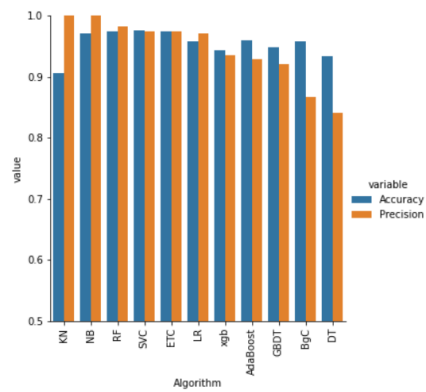
```
✓ [66] performance_df1 = pd.melt(performance_df, id_vars = "Algorithm")
0s performance_df1
```

	Algorithm	variable	value
0	KN	Accuracy	0.905222
1	NB	Accuracy	0.970986
2	RF	Accuracy	0.974855
3	SVC	Accuracy	0.975822
4	ETC	Accuracy	0.974855
5	LR	Accuracy	0.958414
6	xgb	Accuracy	0.943907
7	AdaBoost	Accuracy	0.960348
8	GBDT	Accuracy	0.947776
9	BgC	Accuracy	0.957447
10	DT	Accuracy	0.933269
11	KN	Precision	1.000000
12	NB	Precision	1.000000
13	RF	Precision	0.982759
14	SVC	Precision	0.974790
15	ETC	Precision	0.974576
16	LR	Precision	0.970297
17	xgb	Precision	0.934783

```

[89] sns.catplot(x = 'Algorithm', y='value',
             hue = 'variable',data=performance_df1, kind='bar',height=5)
plt.ylim(0.5,1.0)
plt.xticks(rotation='vertical')
plt.show()
print("Team No:- PNT2022TMID08666")

```



Team No:- PNT2022TMID08666

```

[68] temp_df = pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy_max_ft_3000':accuracy_scores,'Precision_max_ft_3000':precision_scores}).sort_values('Precision_max_ft_3000')
temp_df

```

	Algorithm	Accuracy_max_ft_3000	Precision_max_ft_3000
1	KN	0.905222	1.000000
2	NB	0.970986	1.000000
5	RF	0.974855	0.982759
0	SVC	0.975822	0.974790
8	ETC	0.974855	0.974576
4	LR	0.958414	0.970297
10	xgb	0.943907	0.934783
6	AdaBoost	0.960348	0.929204
9	GBDT	0.947776	0.920000
7	BgC	0.957447	0.867188
3	DT	0.933269	0.841584

```

[69] temp_df = pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy_scaling':accuracy_scores,'Precision_scaling':precision_scores}).sort_values('Precision_scaling')
temp_df

```

	Algorithm	Accuracy_scaling	Precision_scaling
1	KN	0.905222	1.000000
2	NB	0.970986	1.000000
5	RF	0.974855	0.982759

```
[70] new_df = performance_df.merge(temp_df,on='Algorithm')
new_df
```

	Algorithm	Accuracy	Precision	Accuracy_scaling	Precision_scaling
0	KN	0.905222	1.000000	0.905222	1.000000
1	NB	0.970986	1.000000	0.970986	1.000000
2	RF	0.974855	0.982759	0.974855	0.982759
3	SVC	0.975822	0.974790	0.975822	0.974790
4	ETC	0.974855	0.974576	0.974855	0.974576
5	LR	0.958414	0.970297	0.958414	0.970297
6	xgb	0.943907	0.934783	0.943907	0.934783
7	AdaBoost	0.960348	0.929204	0.960348	0.929204
8	GBDT	0.947776	0.920000	0.947776	0.920000
9	BgC	0.957447	0.867188	0.957447	0.867188
10	DT	0.933269	0.841584	0.933269	0.841584

```
[71] new_df_scaled = new_df.merge(temp_df,on='Algorithm')
new_df_scaled
```

	Algorithm	Accuracy	Precision	Accuracy_scaling_x	Precision_scaling_x	Accuracy_scaling_y	Precision_scaling_y
0	KN	0.905222	1.000000	0.905222	1.000000	0.905222	1.000000
1	NB	0.970986	1.000000	0.970986	1.000000	0.970986	1.000000
2	RF	0.974855	0.982759	0.974855	0.982759	0.974855	0.982759

```
[71]
```

5	LR	0.958414	0.970297	0.958414	0.970297	0.958414	0.970297
6	xgb	0.943907	0.934783	0.943907	0.934783	0.943907	0.934783
7	AdaBoost	0.960348	0.929204	0.960348	0.929204	0.960348	0.929204
8	GBDT	0.947776	0.920000	0.947776	0.920000	0.947776	0.920000
9	BgC	0.957447	0.867188	0.957447	0.867188	0.957447	0.867188
10	DT	0.933269	0.841584	0.933269	0.841584	0.933269	0.841584

```
new_df_scaled = new_df.merge(temp_df,on='Algorithm')
new_df_scaled
```

	Algorithm	Accuracy	Precision	Accuracy_scaling_x	Precision_scaling_x	Accuracy_scaling_y	Precision_scaling_y
0	KN	0.905222	1.000000	0.905222	1.000000	0.905222	1.000000
1	NB	0.970986	1.000000	0.970986	1.000000	0.970986	1.000000
2	RF	0.974855	0.982759	0.974855	0.982759	0.974855	0.982759
3	SVC	0.975822	0.974790	0.975822	0.974790	0.975822	0.974790
4	ETC	0.974855	0.974576	0.974855	0.974576	0.974855	0.974576
5	LR	0.958414	0.970297	0.958414	0.970297	0.958414	0.970297
6	xgb	0.943907	0.934783	0.943907	0.934783	0.943907	0.934783
7	AdaBoost	0.960348	0.929204	0.960348	0.929204	0.960348	0.929204
8	GBDT	0.947776	0.920000	0.947776	0.920000	0.947776	0.920000
9	BgC	0.957447	0.867188	0.957447	0.867188	0.957447	0.867188
10	DT	0.933269	0.841584	0.933269	0.841584	0.933269	0.841584

```

09 [73] # Voting Classifier
    svc = SVC(kernel='sigmoid', gamma=1.0, probability=True)
    mnb = MultinomialNB()
    etc = ExtraTreesClassifier(n_estimators=50, random_state=2)

    from sklearn.ensemble import VotingClassifier
    voting = VotingClassifier(estimators=[('svm', svc), ('nb', mnb), ('et', etc)], voting='soft')
    voting

    VotingClassifier(estimators=[('svm',
                                  SVC(gamma=1.0, kernel='sigmoid',
                                      probability=True)),
                                  ('nb', MultinomialNB()),
                                  ('et',
                                   ExtraTreesClassifier(n_estimators=50,
                                                         random_state=2))),
                      voting='soft')

56 [74] voting.fit(X_train, y_train)

    VotingClassifier(estimators=[('svm',
                                  SVC(gamma=1.0, kernel='sigmoid',
                                      probability=True)),
                                  ('nb', MultinomialNB()),
                                  ('et',
                                   ExtraTreesClassifier(n_estimators=50,
                                                         random_state=2))),
                      voting='soft')

25 [75] y_pred = voting.predict(X_test)
    print("Accuracy", accuracy_score(y_test, y_pred))
    print("Precision", precision_score(y_test, y_pred))

    Accuracy 0.9816247582205029
    Precision 0.9917355371900827

```

```

40 [76] # Applying stacking
    estimators=[('svm', svc), ('nb', mnb), ('et', etc)]
    final_estimator=RandomForestClassifier()
    from sklearn.ensemble import StackingClassifier
    clf = StackingClassifier(estimators=estimators, final_estimator=final_estimator)
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    print("Accuracy", accuracy_score(y_test, y_pred))
    print("Precision", precision_score(y_test, y_pred))

    Accuracy 0.9787234042553191
    Precision 0.9328358208955224

```

```

05 [77] import pickle
    pickle.dump(tfidf, open('vectorizer.pkl', 'wb'))
    pickle.dump(mnb, open('model.pkl', 'wb'))

```

## Code:-

```

!pip install pyforest
from pyforest import *
import numpy as np
import pandas as pd
import pickle

df=pd.read_csv('spam.csv',encoding="ISO-8859-1")
df.head(10)
df.shape
df.info()
df.drop(columns=['Unnamed: 2','Unnamed: 3','Unnamed: 4'],inplace=True)
df.head()
df.rename(columns={'v1':'target','v2':'text'},inplace=True)
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
df['target'] = encoder.fit_transform(df['target'])

```

```

df.head()
df.isnull().sum()
df.duplicated().sum()
df=df.drop_duplicates(keep='first')
df.duplicated().sum()
df.shape
df['target'].value_counts()
plt.pie(df['target'].value_counts(), labels=['ham','spam'],autopct="%0.2f")
plt.show()
print("Team No:- PNT2022TMID08666")
import nltk
%pip install nltk
nltk.download('punkt')
df['num_characters'] = df['text'].apply(len)
df.head()
df['num_words'] = df['text'].apply(lambda x:len(nltk.word_tokenize(x)))
df.head()
df['num_sentences'] = df['text'].apply(lambda x:len(nltk.sent_tokenize(x)))
df[['num_characters','num_words','num_sentences']].describe()
df[df['target'] == 0][['num_characters','num_words','num_sentences']].describe()
df[df['target'] == 1][['num_characters','num_words','num_sentences']].describe()
plt.figure(figsize=(12,6))
sns.histplot(df[df['target'] == 0]['num_characters'])
sns.histplot(df[df['target'] == 1]['num_characters'],color='red')
print("Team No:- PNT2022TMID08666")
plt.figure(figsize=(12,6))
sns.histplot(df[df['target'] == 0]['num_words'])
sns.histplot(df[df['target'] == 1]['num_words'],color='red')
print("Team No:- PNT2022TMID08666")
sns.pairplot(df,hue='target')
print("Team No:- PNT2022TMID08666")
sns.heatmap(df.corr(),annot=True)
print("Team No:- PNT2022TMID08666")
from nltk.corpus import stopwords
import string
string.punctuation
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()
df['text'][100]
import nltk
nltk.download('stopwords')
df['transformed_text'] = df['text'].apply(transform_text)
df.head()
from wordcloud import WordCloud

```

```

wc = WordCloud(width=500,height=500,min_font_size=10,background_color='white')
spam_wc = wc.generate(df[df['target'] == 1]['transformed_text'].str.cat(sep=" "))
plt.figure(figsize=(15,6))
plt.imshow(spam_wc)
print("Team No:- PNT2022TMID08666")
ham_wc = wc.generate(df[df['target'] == 0]['transformed_text'].str.cat(sep=" "))
plt.figure(figsize=(15,6))
plt.imshow(ham_wc)
print("Team No:- PNT2022TMID08666")
# For Spam messages
spam_corpus = []
for msg in df[df['target'] == 1]['transformed_text'].tolist():
    for word in msg.split():
        spam_corpus.append(word)
len(spam_corpus)
from collections import Counter
sns.barplot(pd.DataFrame(Counter(spam_corpus).most_common(30))[0],pd.DataFrame(Counter(spam_corpus).most_common(30))[1])
plt.xticks(rotation='vertical')
plt.show()
print("Team No:- PNT2022TMID08666")
# For ham messages
ham_corpus = []
for msg in df[df['target'] == 0]['transformed_text'].tolist():
    for word in msg.split():
        ham_corpus.append(word)
len(ham_corpus)
from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer
cv = CountVectorizer()
tfidf = TfidfVectorizer(max_features=3000)
X = tfidf.fit_transform(df['transformed_text']).toarray()
X.shape
y = df['target'].values
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=2)
from sklearn.naive_bayes import GaussianNB,MultinomialNB,BernoulliNB
from sklearn.metrics import accuracy_score,confusion_matrix,precision_score
mnb = MultinomialNB()
# MultinomialNB
mnb.fit(X_train,y_train)
y_pred1 = mnb.predict(X_test)
print(accuracy_score(y_test,y_pred1))
print(confusion_matrix(y_test,y_pred1))
print(precision_score(y_test,y_pred1))

```

```

from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.ensemble import GradientBoostingClassifier
from xgboost import XGBClassifier
svc = SVC(kernel='sigmoid', gamma=1.0)
knc = KNeighborsClassifier()
mnbc = MultinomialNB()
dtc = DecisionTreeClassifier(max_depth=5)
lrc = LogisticRegression(solver='liblinear', penalty='l1')
rfc = RandomForestClassifier(n_estimators=50, random_state=2)
abc = AdaBoostClassifier(n_estimators=50, random_state=2)
bc = BaggingClassifier(n_estimators=50, random_state=2)
etc = ExtraTreesClassifier(n_estimators=50, random_state=2)
gbdt = GradientBoostingClassifier(n_estimators=50, random_state=2)
xgb = XGBClassifier(n_estimators=50, random_state=2)
clfs = {
    'SVC' : svc,
    'KN' : knc,
    'NB' : mnbc,
    'DT' : dtc,
    'LR' : lrc,
    'RF' : rfc,
    'AdaBoost' : abc,
    'BgC' : bc,
    'ETC' : etc,
    'GBDT' : gbdt,
    'xgb' : xgb
}
def train_classifier(clf, X_train, y_train, X_test, y_test):
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred)
    return accuracy, precision
train_classifier(svc, X_train, y_train, X_test, y_test)
accuracy_scores = []
precision_scores = []

```



```

for name,clf in clfs.items():
current_accuracy,current_precision = train_classifier(clf, X_train,y_train,X_test,y_test)
print("For ",name)
print("Accuracy - ",current_accuracy)
print("Precision - ",current_precision)
accuracy_scores.append(current_accuracy)
precision_scores.append(current_precision)
performance_df
pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy':accuracy_scores,'Precision':precision_scores
}).sort_values('Precision',ascending=False)
performance_df
performance_df1 = pd.melt(performance_df, id_vars = "Algorithm")
performance_df1
sns.catplot(x = 'Algorithm', y='value',
            hue = 'variable',data=performance_df1, kind='bar',height=5)
plt.ylim(0.5,1.0)
plt.xticks(rotation='vertical')
plt.show()
print("Team No:- PNT2022TMID08666")
temp_df
pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy_max_ft_3000':accuracy_scores,'Precision_max_ft_3000':precision_scores}).sort_values('Precision_max_ft_3000',ascending=False)
temp_df
temp_df
pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy_scaling':accuracy_scores,'Precision_scaling':precision_scores}).sort_values('Precision_scaling',ascending=False)
temp_df
new_df = performance_df.merge(temp_df,on='Algorithm')
new_df
new_df_scaled = new_df.merge(temp_df,on='Algorithm')
new_df_scaled
new_df_scaled = new_df.merge(temp_df,on='Algorithm')
new_df_scaled
# Voting Classifier
svc = SVC(kernel='sigmoid', gamma=1.0,probability=True)
mnb = MultinomialNB()
etc = ExtraTreesClassifier(n_estimators=50, random_state=2)
from sklearn.ensemble import VotingClassifier
voting = VotingClassifier(estimators=[('svm', svc), ('nb', mnb), ('et', etc)],voting='soft')
voting
voting.fit(X_train,y_train)
y_pred = voting.predict(X_test)
print("Accuracy",accuracy_score(y_test,y_pred))
print("Precision",precision_score(y_test,y_pred))

```

```
# Applying stacking
estimators=[('svm', svc), ('nb', mnbc), ('et', etc)]
final_estimator=RandomForestClassifier()
from sklearn.ensemble import StackingClassifier
clf = StackingClassifier(estimators=estimators, final_estimator=final_estimator)
clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
print("Accuracy",accuracy_score(y_test,y_pred))
print("Precision",precision_score(y_test,y_pred))
import pickle
pickle.dump(tfidf,open('vectorizer.pkl','wb'))
pickle.dump(mnbc,open('model.pkl','wb'))
```