Email spam detection

In [ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import nltk
```

In [ ]:
```
nltk.download('punkt')
```

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

Out[ ]: True

In [ ]:
```
df=pd.read_csv('spam.csv', encoding='latin-1')
```

In [ ]:
```
df.head()
```

Out[ ]:

	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

In [ ]:
```
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
```

In [ ]:
```
df.shape
```

Out[ ]:
```
(5572, 5)
```

Data cleaning

In [ ]:
```
df.drop(df.iloc[:,2:],axis=1,inplace=True)
```

In [ ]:
```
df.head()
```

Out[ ]:

	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...

In [ ]:
```
df['v1'].value_counts()
```

Out[ ]:
```
v1
ham     4825
spam     747
Name: count, dtype: int64
```

In [ ]:
```
df['v1']=pd.get_dummies(df['v1'],drop_first=True,dtype='int')
```

In [ ]:
```
df.head()
```

Out[ ]:

	v1	v2
0	0	Go until jurong point, crazy.. Available only ...
1	0	Ok lar... Joking wif u oni...
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...

In [ ]:
```
df.isnull().sum()
```

Out[ ]:
```
v1    0
v2    0
dtype: int64
```

In [ ]:
```
df.duplicated().sum()
```

Out[ ]:
```
403
```

In [ ]:
```
df.drop_duplicates(keep='first',inplace=True)
```

In [ ]:
```
df.duplicated().sum()
```

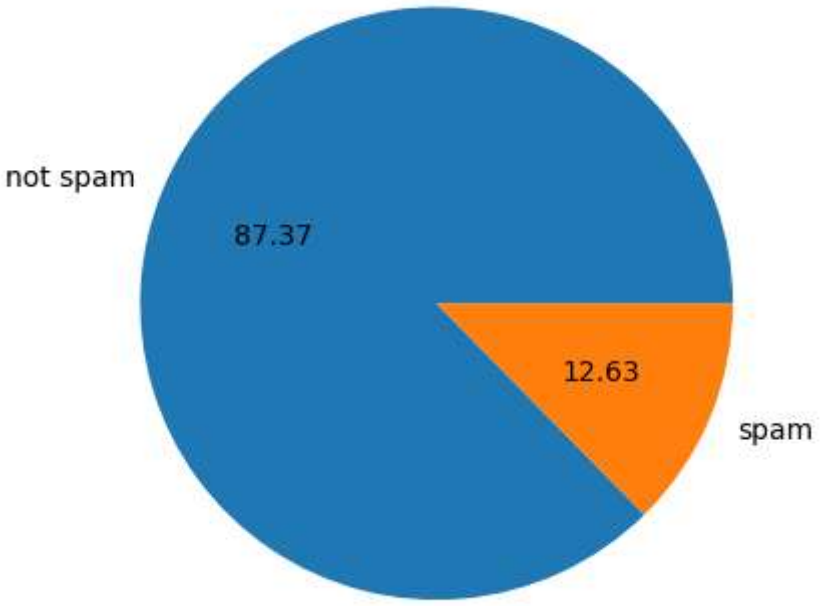
Out[ ]:
```
0
```

In [ ]:
```
df.shape
```

Out[ ]:
```
(5169, 2)
```

EDA

In [ ]:
```
plt.pie(df['v1'].value_counts(),labels=['not spam','spam'],autopct='%.2f')
plt.show()
```



from above it depicts that data is unbalanced

In [ ]:
```
df['num of characters']=df['v2'].apply(len)
```

In [ ]:
```
df['no of words']=df['v2'].apply(lambda x: len(nltk.word_tokenize(x)))
```

In [ ]:
```
df['num of sentenses']=df['v2'].apply(lambda x: len(nltk.sent_tokenize(x)))
```

In [ ]:
```
df.head()
```

Out[ ]:

	v1	v2	num of characters	no of words	num of sentenses
0	0	Go until jurong point, crazy.. Available only ...	111	24	2
1	0	Ok lar... Joking wif u oni...	29	8	2
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	155	37	2
3	0	U dun say so early hor... U c already then say...	49	13	1
4	0	Nah I don't think he goes to usf, he lives aro...	61	15	1

In [ ]:
```
df.iloc[:,2:].describe()
```

Out[ ]:

	num of characters	no of words	num of sentenses
count	5169.000000	5169.000000	5169.000000
mean	78.977945	18.455794	1.965564
std	58.236293	13.324758	1.448541
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	38.000000

In [ ]:
```
#spam
df[df['v1']==1 ][['num of characters'   ,'no of words'  ,'num of sentenses']].describe()
```

Out[ ]:

	num of characters	no of words	num of sentenses
count	653.000000	653.000000	653.000000
mean	137.891271	27.667688	2.970904
std	30.137753	7.008418	1.488425
min	13.000000	2.000000	1.000000
25%	132.000000	25.000000	2.000000
50%	149.000000	29.000000	3.000000
75%	157.000000	32.000000	4.000000
max	224.000000	46.000000	9.000000

for spam msgs on avg num of words=27 ,num of charcters=138

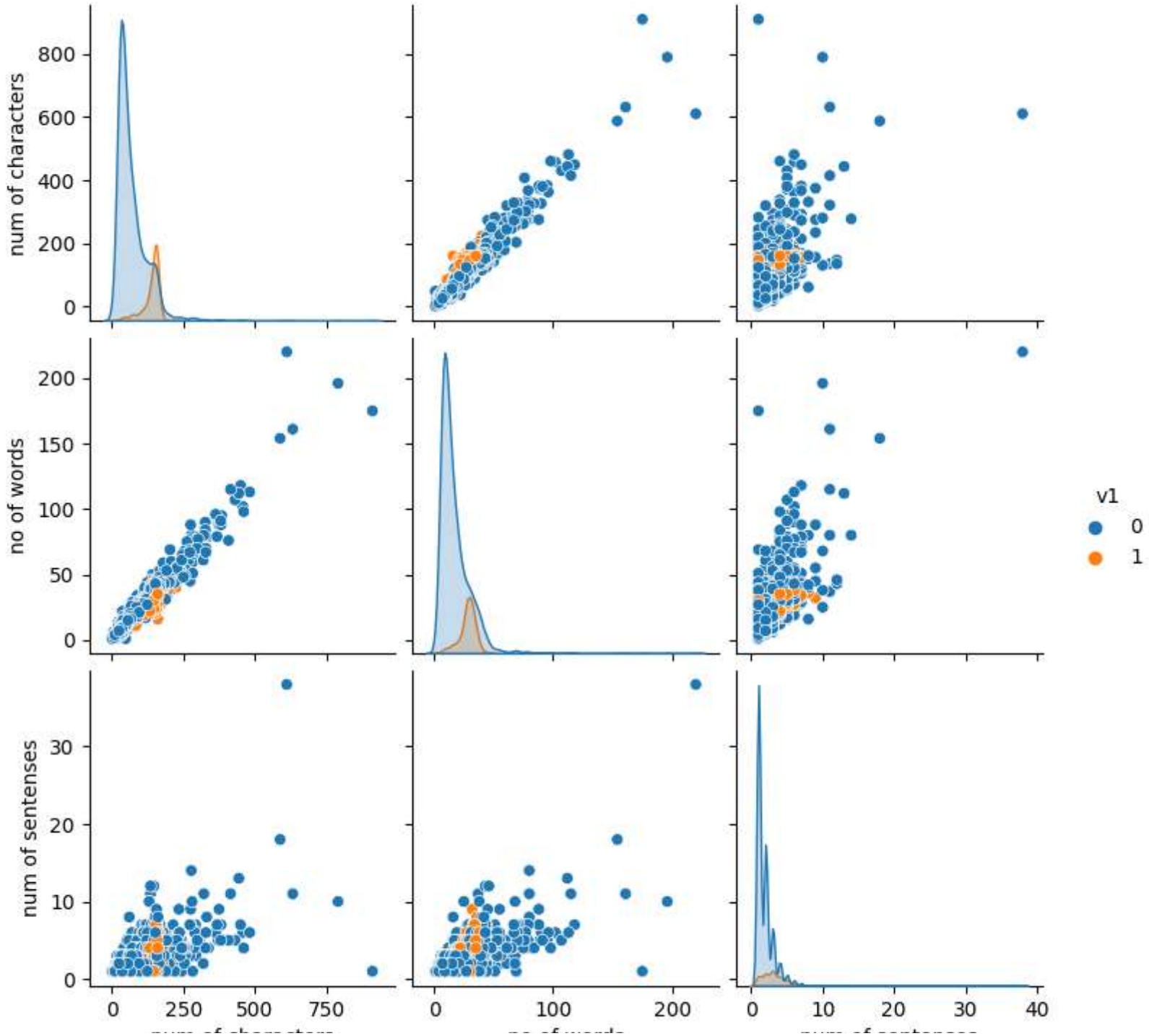
In [ ]:
```
# not spam
df[df['v1']==0 ][['num of characters'   ,'no of words'  ,'num of sentenses']].describe()
```

Out[ ]:

	num of characters	no of words	num of sentenses
count	4516.000000	4516.000000	4516.000000
mean	70.459256	17.123782	1.820195
std	56.358207	13.493970	1.383657
min	2.000000	1.000000	1.000000
25%	34.000000	8.000000	1.000000
50%	52.000000	13.000000	1.000000
75%	90.000000	22.000000	2.000000
max	910.000000	220.000000	38.000000

In [ ]:
```
sns.pairplot(df,hue='v1')
```

Out[ ]: <seaborn.axisgrid.PairGrid at 0x201e66c6f90>


num of characters
no of words
num of sentenses
v1
0
1
0
250
500
750
0
100
200
0
10
20
30
40

num of characters no of words num of sentenses

Data preprocessing for case converting token generation removing special charaters removing stop words stemming

In [ ]:
```
nltk.download('stopwords')
```

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

Out[ ]: True

In [ ]:
```
from nltk.stem.porter import PorterStemmer
from nltk.corpus import stopwords
import string
ps=PorterStemmer()
```

In [ ]:
```
def preprocess(text):

    #case convertion
    text=text.lower()
    #token generation
    text=nltk.word_tokenize(text)

    #removing special characters
    x=[]
    for i in text:
        if i.isalnum():
            x.append(i)

    #removing stop words
    text=x[:]
    y=[]
    for i in text:
        if i not in stopwords.words('english') and i not in string.punctuation:
            y.append(i)

    #stemming
    text=y[:]
    y.clear()
    for i in text:
```

```
        y.append(ps.stem(i))

    return " ".join(y)
```

In [ ]:
```
df['preprocessed_text']=df['v2'].apply(preprocess)
```

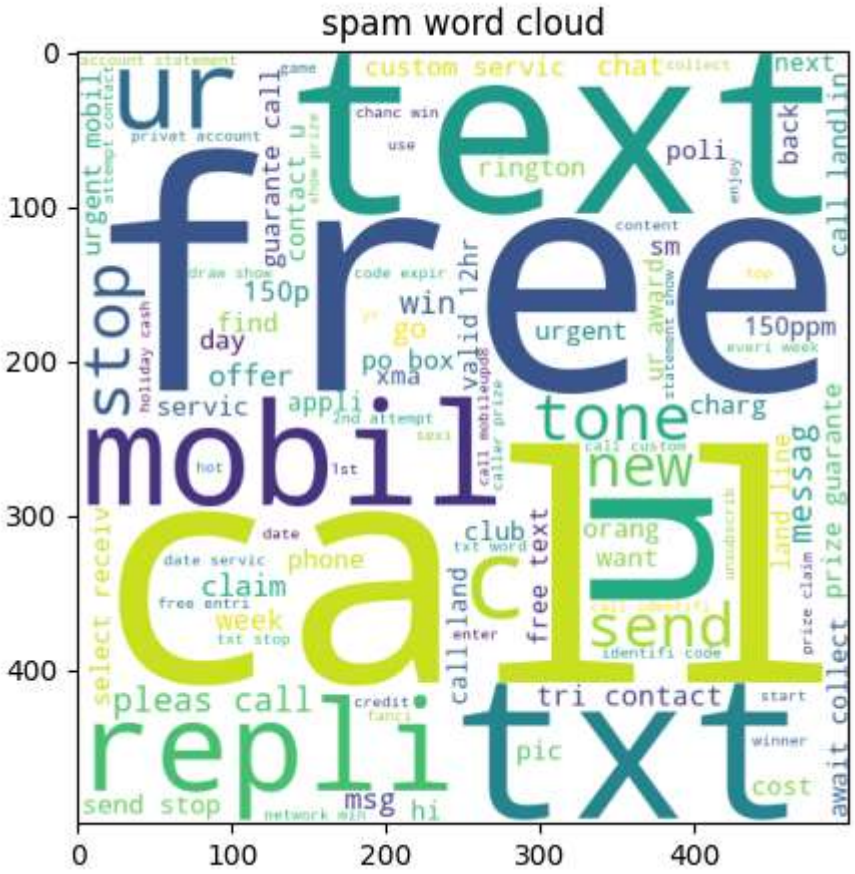
In [ ]:
```
df.head()
```

Out[ ]:

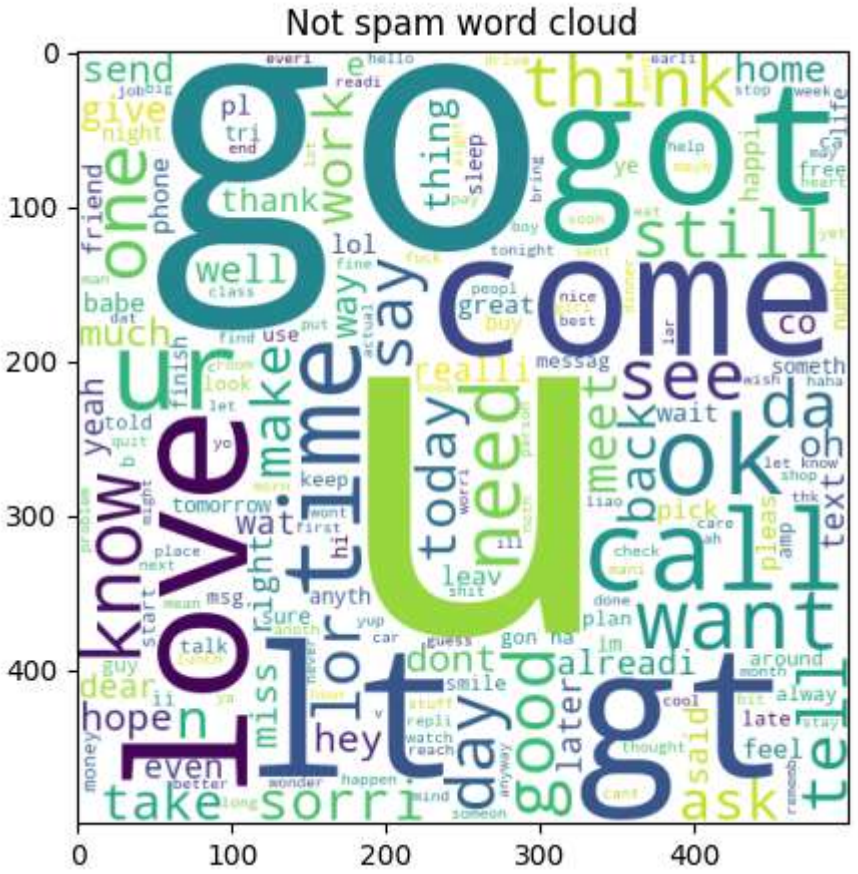
	v1	v2	num of characters	no of words	num of sentenses	preprocessed_text
0	0	Go until jurong point, crazy.. Available only ...	111	24	2	go jurong point crazi avail bugi n great world...
1	0	Ok lar... Joking wif u oni...	29	8	2	ok lar joke wif u oni
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	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 alreadi 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

In [ ]:
```
from wordcloud import WordCloud
wc=WordCloud(width=500,height=500,min_font_size=10,background_color='white')
```

In [ ]:
```
spamw=wc.generate(df[df['v1']==1]['preprocessed_text'].str.cat(sep=' '))
plt.figure(figsize=(10,5))
plt.imshow(spamw)
plt.title('spam word cloud')
plt.show()
```



In [ ]:
```
spamwn=wc.generate(df[df['v1']==0]['preprocessed_text'].str.cat(sep=' '))
plt.figure(figsize=(10,5))
plt.imshow(spamwn)
plt.title('Not spam word cloud')
plt.show()
```



vectorizing

In [ ]:
```
from sklearn.feature_extraction.text import CountVectorizer
cv=CountVectorizer()
```

In [ ]:
```
x=cv.fit_transform(df['preprocessed_text']).toarray()
```

In [ ]:
```
x.shape
```

Out[ ]:
```
(5169, 6708)
```

In [ ]:
```
y=df['v1'].values
```

Model Building

In [ ]:
```
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)
```

In [ ]:
```
from lazypredict.Supervised import LazyClassifier
lazy=LazyClassifier()
```

In [ ]:
```
lazy.fit(x_train,x_test,y_train,y_test)
```

```
100%|██████████| 29/29 [07:29<00:00, 15.51s/it]
```

Out[ ]: (

Model	Accuracy	Balanced Accuracy	ROC AUC	F1 Score	Time Taken
PassiveAggressiveClassifier	0.97	0.92	0.92	0.97	1.98
Perceptron	0.95	0.92	0.92	0.95	1.68
LinearSVC	0.97	0.91	0.91	0.97	28.57
LGBMClassifier	0.97	0.89	0.89	0.97	2.46
XGBClassifier	0.97	0.89	0.89	0.97	10.99
ExtraTreesClassifier	0.97	0.87	0.87	0.97	47.24
BernoulliNB	0.97	0.87	0.87	0.96	1.43
AdaBoostClassifier	0.96	0.87	0.87	0.96	27.47
NearestCentroid	0.96	0.86	0.86	0.96	1.38
DecisionTreeClassifier	0.95	0.86	0.86	0.95	11.90
ExtraTreeClassifier	0.95	0.86	0.86	0.95	1.66
LogisticRegression	0.96	0.85	0.85	0.96	
RandomForestClassifier	0.96	0.85	0.85	0.96	
BaggingClassifier	0.95	0.84	0.84	0.95	
GaussianNB	0.86	0.83	0.83	0.87	
RidgeClassifierCV	0.95	0.82	0.82	0.94	
RidgeClassifier	0.94	0.80	0.80	0.93	
LinearDiscriminantAnalysis	0.94	0.80	0.80	0.93	
SVC	0.93	0.74	0.74	0.92	
QuadraticDiscriminantAnalysis	0.55	0.74	0.74	0.61	
CalibratedClassifierCV	0.92	0.71	0.71	0.91	
SGDClassifier	0.92	0.69	0.69	0.90	
LabelPropagation	0.88	0.54	0.54	0.83	
LabelSpreading	0.88	0.54	0.54	0.83	
KNeighborsClassifier	0.88	0.52	0.52	0.82	
DummyClassifier	0.87	0.50	0.50	0.81	

```
LogisticRegression                 2.43
RandomForestClassifier            18.40
BaggingClassifier                 42.19
GaussianNB                         1.80
RidgeClassifierCV                 13.81
RidgeClassifier                    5.06
LinearDiscriminantAnalysis        49.21
SVC                               40.25
QuadraticDiscriminantAnalysis     27.20
CalibratedClassifierCV            92.49
SGDClassifier                      1.72
LabelPropagation                   5.85
LabelSpreading                     5.91
KNeighborsClassifier               2.11
DummyClassifier                    1.09  ,
                           Accuracy  Balanced Accuracy  ROC AUC  F1 Score  \
Model
PassiveAggressiveClassifier      0.97               0.92     0.92      0.97
Perceptron                       0.95               0.92     0.92      0.95
LinearSVC                        0.97               0.91     0.91      0.97
LGBMClassifier                   0.97               0.89     0.89      0.97
XGBClassifier                    0.97               0.89     0.89      0.97
ExtraTreesClassifier             0.97               0.87     0.87      0.97
BernoulliNB                      0.97               0.87     0.87      0.96
AdaBoostClassifier               0.96               0.87     0.87      0.96
NearestCentroid                  0.96               0.86     0.86      0.96
DecisionTreeClassifier           0.95               0.86     0.86      0.95
ExtraTreeClassifier              0.95               0.86     0.86      0.95
LogisticRegression               0.96               0.85     0.85      0.96
RandomForestClassifier           0.96               0.85     0.85      0.96
BaggingClassifier                0.95               0.84     0.84      0.95
GaussianNB                       0.86               0.83     0.83      0.87
RidgeClassifierCV                0.95               0.82     0.82      0.94
RidgeClassifier                  0.94               0.80     0.80      0.93
LinearDiscriminantAnalysis       0.94               0.80     0.80      0.93
SVC                              0.93               0.74     0.74      0.92
QuadraticDiscriminantAnalysis    0.55               0.74     0.74      0.61
CalibratedClassifierCV           0.92               0.71     0.71      0.91
SGDClassifier                    0.92               0.69     0.69      0.90
LabelPropagation                 0.88               0.54     0.54      0.83
LabelSpreading                   0.88               0.54     0.54      0.83
KNeighborsClassifier             0.88               0.52     0.52      0.82
```

```
DummyClassifier                    0.87               0.50      0.50      0.81

                                Time Taken
Model
PassiveAggressiveClassifier           1.98
Perceptron                            1.68
LinearSVC                            28.57
LGBMClassifier                        2.46
XGBClassifier                        10.99
ExtraTreesClassifier                 47.24
BernoulliNB                           1.43
AdaBoostClassifier                   27.47
NearestCentroid                       1.38
DecisionTreeClassifier               11.90
ExtraTreeClassifier                   1.66
LogisticRegression                    2.43
RandomForestClassifier               18.40
BaggingClassifier                    42.19
GaussianNB                            1.80
RidgeClassifierCV                    13.81
RidgeClassifier                       5.06
LinearDiscriminantAnalysis           49.21
SVC                                  40.25
QuadraticDiscriminantAnalysis        27.20
CalibratedClassifierCV               92.49
SGDClassifier                         1.72
LabelPropagation                      5.85
LabelSpreading                        5.91
KNeighborsClassifier                  2.11
DummyClassifier                       1.09  )
```

In [ ]:
```
#LinearSVM
from sklearn.svm import LinearSVC
from sklearn.metrics import accuracy_score,precision_score
lsvm=LinearSVC()
```

In [ ]:
```
lsvm.fit(x_train,y_train)
predsvm=lsvm.predict(x_test)
accsvm=accuracy_score(y_test,predsvm)
precsvm=precision_score(y_test,predsvm)
print(f'LinearSVC \n accuracy={accsvm*100}\tprecision={precsvm}')
```

```
LinearSVC
 accuracy=97.77562862669245     precision=0.9911504424778761
```

In [ ]:
```
from sklearn.naive_bayes import MultinomialNB
mnb=MultinomialNB()
```

In [ ]:
```
mnb.fit(x_train,y_train)
predmnb=mnb.predict(x_test)
accmnb=accuracy_score(y_test,predmnb)
precmnb=precision_score(y_test,predmnb)
print(f'MultinomialNB\n accuracy={accmnb*100}\tprecision={precmnb}')
```

```
MultinomialNB
 accuracy=97.38878143133462     precision=0.8962962962962963
```

In [ ]:
```
from sklearn.ensemble import HistGradientBoostingClassifier
hgbc=HistGradientBoostingClassifier()
```

In [ ]:
```
hgbc.fit(x_train,y_train)
predhgbc=hgbc.predict(x_test)
acchgbc=accuracy_score(y_test,predhgbc)
prehgbc=precision_score(y_test,predhgbc)
print(f'HistGradientBoostingClassifier\n accuracy={acchgbc*100}\tprecision={prehgbc}')
```

```
HistGradientBoostingClassifier
 accuracy=96.80851063829788     precision=0.954954954954955
```

In [ ]:
```
from sklearn.linear_model import PassiveAggressiveClassifier
pas=PassiveAggressiveClassifier()
```

In [ ]:
```
pas.fit(x_train, y_train)
predpas=pas.predict(x_test)
accpas=accuracy_score(y_test,predpas)
prepas=precision_score(y_test,predpas)
print(f'PassiveAggressiveClassifier\n accuracy={accpas*100}\tprecision={prepas}')
```

```
PassiveAggressiveClassifier
 accuracy=97.87234042553192     precision=0.9827586206896551
```

In [ ]:
```
a=[accsvm,accpas,accmnb,acchgbc]
p=[precsvm,prepas,precmnb,prehgbc]
rep={'accuracy':a,'precision':p}
```

```
ind=['svm','passiveagressive','multinomialNB','histgradientboosting']
re=pd.DataFrame(rep,index=ind)
re
```

Out[ ]:

	accuracy	precision
svm	0.98	0.99
passiveagressive	0.98	0.98
multinomialNB	0.97	0.90
histgradientboosting	0.97	0.95

Overall LinearSVM classifier got highest performance accuracy=97.77562862669245 precision=0.9911504424778761

In [ ]:

```
import pickle
pickle.dump(lsvm,open('spam_detection_LSVM.pkl','wb'))
pickle.dump(pas,open('spam_detection_passiveAgressiveclassifier.pkl','wb'))
pickle.dump(cv,open('Vectorizer.pkl','wb'))
```