```
import pandas as pd
import numpy as np

import warnings
warnings.filterwarnings('ignore')

encodings=['utf-8','latin1','ISO-8859-1','cp1252']
for encoding in encodings:
    try:
        df=pd.read_csv('spam.csv',encoding=encoding)
        print(f"File successfully read with encoding:{encoding}")
        break
    except UnicodeDecodeError:
        print(f"Failed to read with encoding:{encoding}")
        continue

if 'df' in locals():
    print("CSV file has been successfully loaded.")
else:
    print("All encoding attempts failed . unable to read the csv
file.")
```

```
Failed to read with encoding:utf-8
File successfully read with encoding:latin1
CSV file has been successfully loaded.
```

```
df.sample(4)
```

```
        v1                                                  v2 Unnamed:
2  \
4014  spam  You will be receiving this week's Triple Echo ...
NaN
1012   ham     I dunno they close oredi not... ÌÏ v ma fan...
NaN
3143   ham      Haha I heard that, text me when you're around
NaN
4122   ham  Cool, want me to go to kappa or should I meet ...
NaN

     Unnamed: 3 Unnamed: 4
4014        NaN        NaN
1012        NaN        NaN
3143        NaN        NaN
4122        NaN        NaN
```

```
df.shape
```

```
(5572, 5)
```

1.Data Cleaning

```
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
```

```
df.isnull().sum()
```

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

```
# lot of values in column {Unnamed: 2 ,Unnamed: 3,  Unnamed: 4 } are
NULL
# so we drop these columns
df.drop(columns=['Unnamed: 2' ,'Unnamed: 3',  'Unnamed:
4'],inplace=True)
```

```
df.sample(5)
```

```
        v1                                                 v2
1675   ham  Painful words- \I thought being Happy was the ...
1521   ham            Are you angry with me. What happen dear
1738   ham                 K go and sleep well. Take rest:-).
4795  spam  URGENT This is our 2nd attempt to contact U. Y...
5132   ham  it's still not working. And this time i also t...
```

```
# renaming the columns
df.rename(columns={'v1':'target','v2':'text'},inplace=True)
df.head()
```

```
  target                                               text
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...
```

```python
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()

df['target']=le.fit_transform(df['target'])

df.head()
```

```
   target                                               text
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...
```

```python
#missing value
df.isnull().sum()
```

```
target    0
text      0
dtype: int64
```

```python
df.duplicated().sum()
```

```
403
```

```python
#remove duplicates
df=df.drop_duplicates(keep='first')

df.duplicated().sum()
```

```
0
```

```python
df.shape
```

```
(5169, 2)
```

EDA

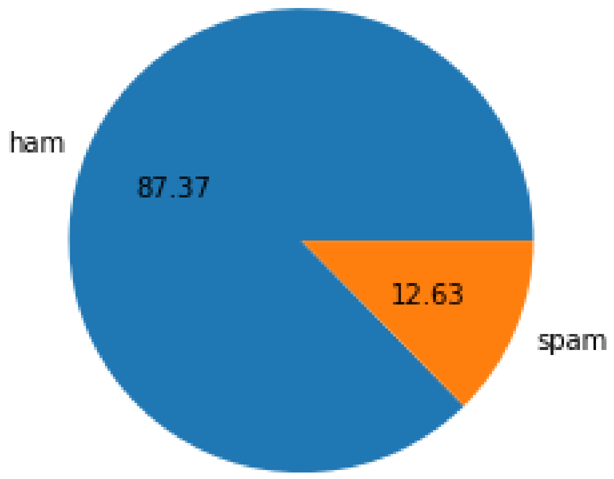
```python
df.head()
```

```
   target                                               text
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...
```

```
df['target'].value_counts()
```

```
0    4516
1     653
Name: target, dtype: int64
```

```
import matplotlib.pyplot as plt
plt.pie(df['target'].value_counts(),labels=['ham','spam'],autopct="%0.
2f")
plt.show()
```



```
import nltk
```

```
!pip install nltk
```

```
Requirement already satisfied: nltk in c:\users\dhiraj kumar\
anaconda3\lib\site-packages (3.7)
Requirement already satisfied: tqdm in c:\users\dhiraj kumar\
anaconda3\lib\site-packages (from nltk) (4.64.0)
Requirement already satisfied: joblib in c:\users\dhiraj kumar\
anaconda3\lib\site-packages (from nltk) (1.1.0)
Requirement already satisfied: regex>=2021.8.3 in c:\users\dhiraj
kumar\anaconda3\lib\site-packages (from nltk) (2022.3.15)
Requirement already satisfied: click in c:\users\dhiraj kumar\
anaconda3\lib\site-packages (from nltk) (8.0.4)
Requirement already satisfied: colorama in c:\users\dhiraj kumar\
anaconda3\lib\site-packages (from click->nltk) (0.4.4)
```

```
nltk.download('punkt')
```

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

True
```

```python
# now we create a new column of number of char
df['num_char']=df['text'].apply(len)

df.head()
```

```
   target                                               text  num_char
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
```

```python
# new column for number of words
df['num_words']=df['text'].apply(lambda x:len(nltk.word_tokenize(x)))

df.sample(5)
```

```
      target                                               text
num_char  \
3021       0                             How dare you change my ring
27
2057       0                Nothing, i got msg frm tht unknown no..
39
5183       0                    Fuuuuck I need to stop sleepin, sup
35
4481       0  What do u reckon as need 2 arrange transport i...
69
94         0  Havent planning to buy later. I check already ...
107

      num_words
3021          6
2057         10
5183          8
4481         17
94           23
```

```python
df['text'][0]
```

```
'Go until jurong point, crazy.. Available only in bugis n great world
la e buffet... Cine there got amore wat...'
```

```python
# new column for number of sentence
df['num_sent']=df['text'].apply(lambda x:len(nltk.sent_tokenize(x)))

df.head()
```

```
   target                                               text  num_char  \
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   


   num_words  num_sent
0         24         2
1          8         2
2         37         2
3         13         1
4         15         1
```

```
df[['num_char','num_words','num_sent']].describe()
```

```
          num_char    num_words     num_sent
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
```

```
# targetting ham
df[df['target']==0][['num_char','num_words','num_sent']].describe()
```

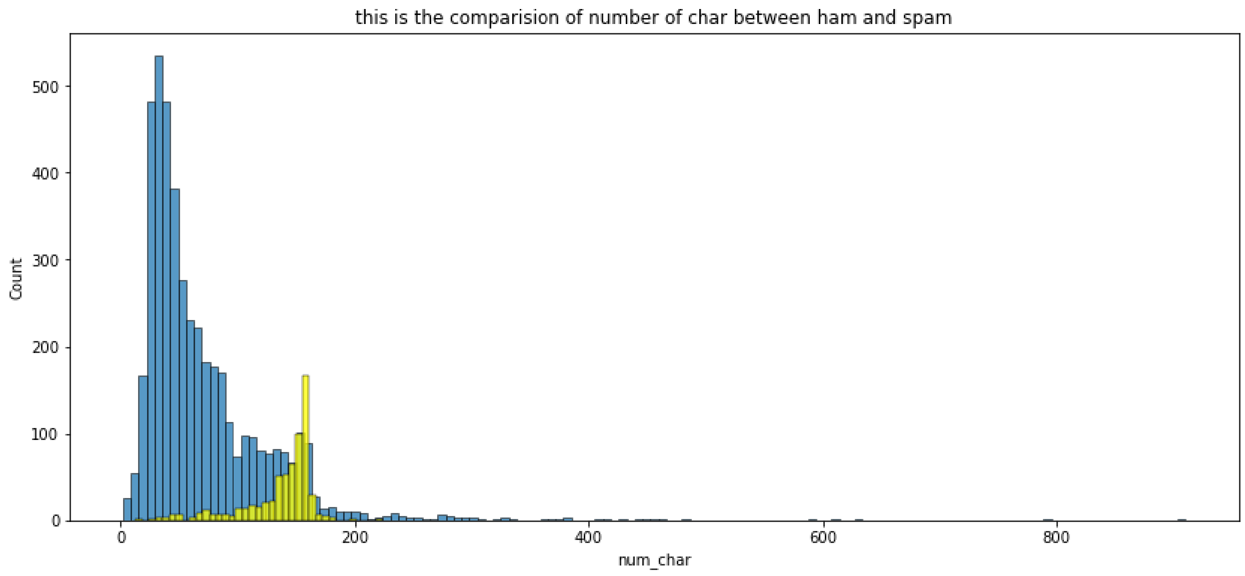
```
          num_char    num_words     num_sent
count  4516.000000  4516.000000  4516.000000
mean     70.459256    17.120903     1.799601
std      56.358207    13.493725     1.278465
min       2.000000     1.000000     1.000000
25%      34.000000     8.000000     1.000000
50%      52.000000    13.000000     1.000000
75%      90.000000    22.000000     2.000000
max     910.000000   220.000000    28.000000
```

```
# targetting spam
df[df['target']==1][['num_char','num_words','num_sent']].describe()
```

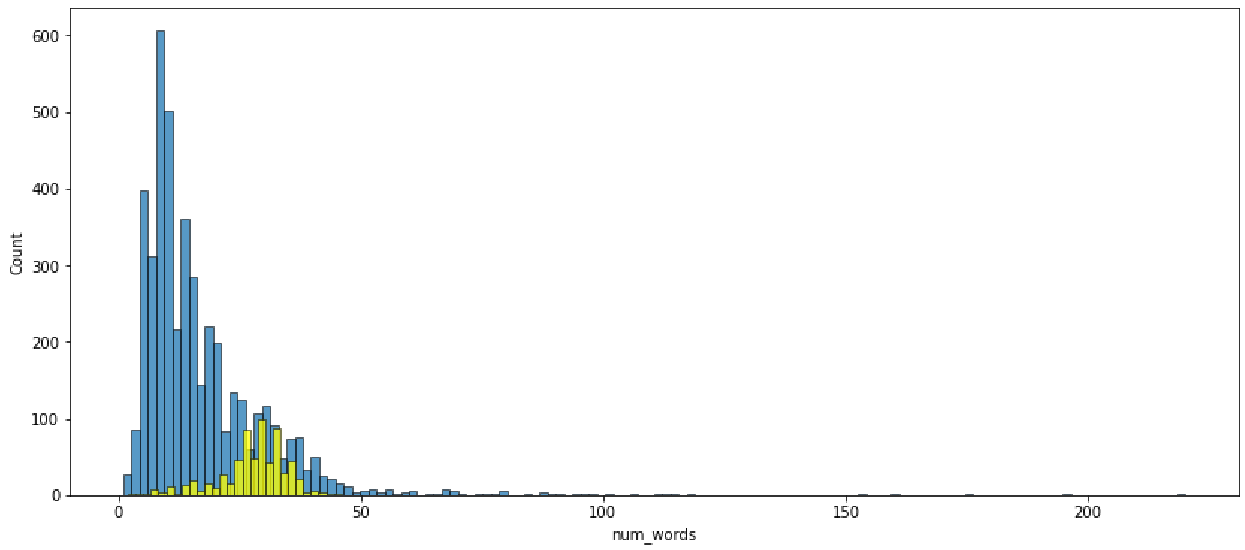
```
         num_char   num_words    num_sent
count  653.000000  653.000000  653.000000
mean   137.891271   27.667688    2.967841
```

```
std     30.137753    7.008418    1.483201
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    8.000000
```

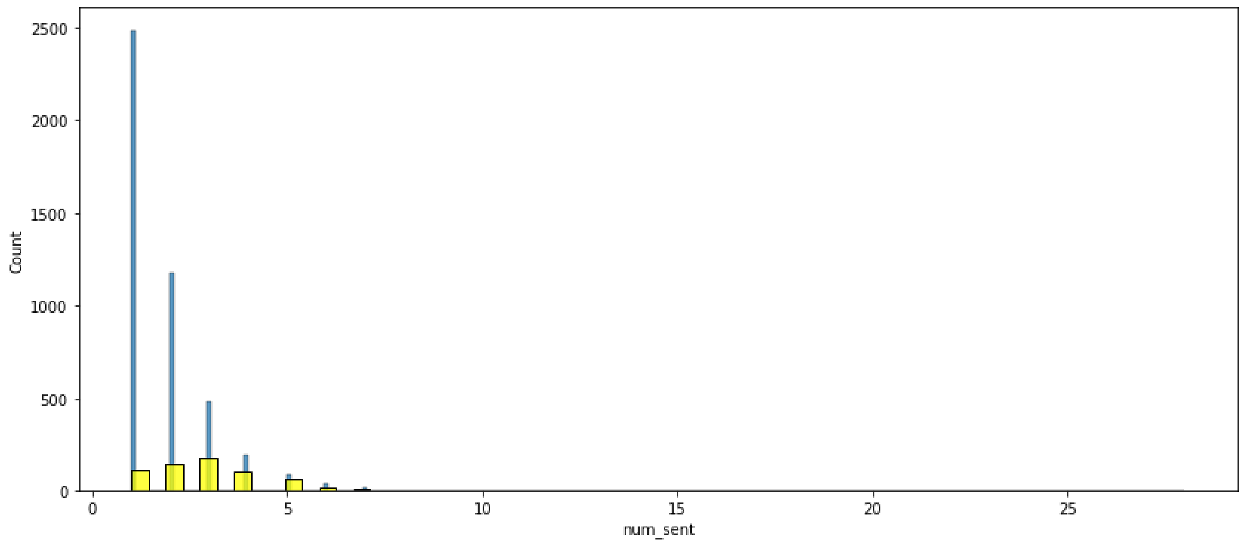
```
import seaborn as sns

plt.figure(figsize=(14,6))
# plt.subplot(121)
sns.histplot(df[df['target']==0]['num_char'])
# plt.subplot(122)
sns.histplot(df[df['target']==1]['num_char'],color='yellow')
plt.title("this is the comparision of number of char between ham and 
spam")
plt.show()
```



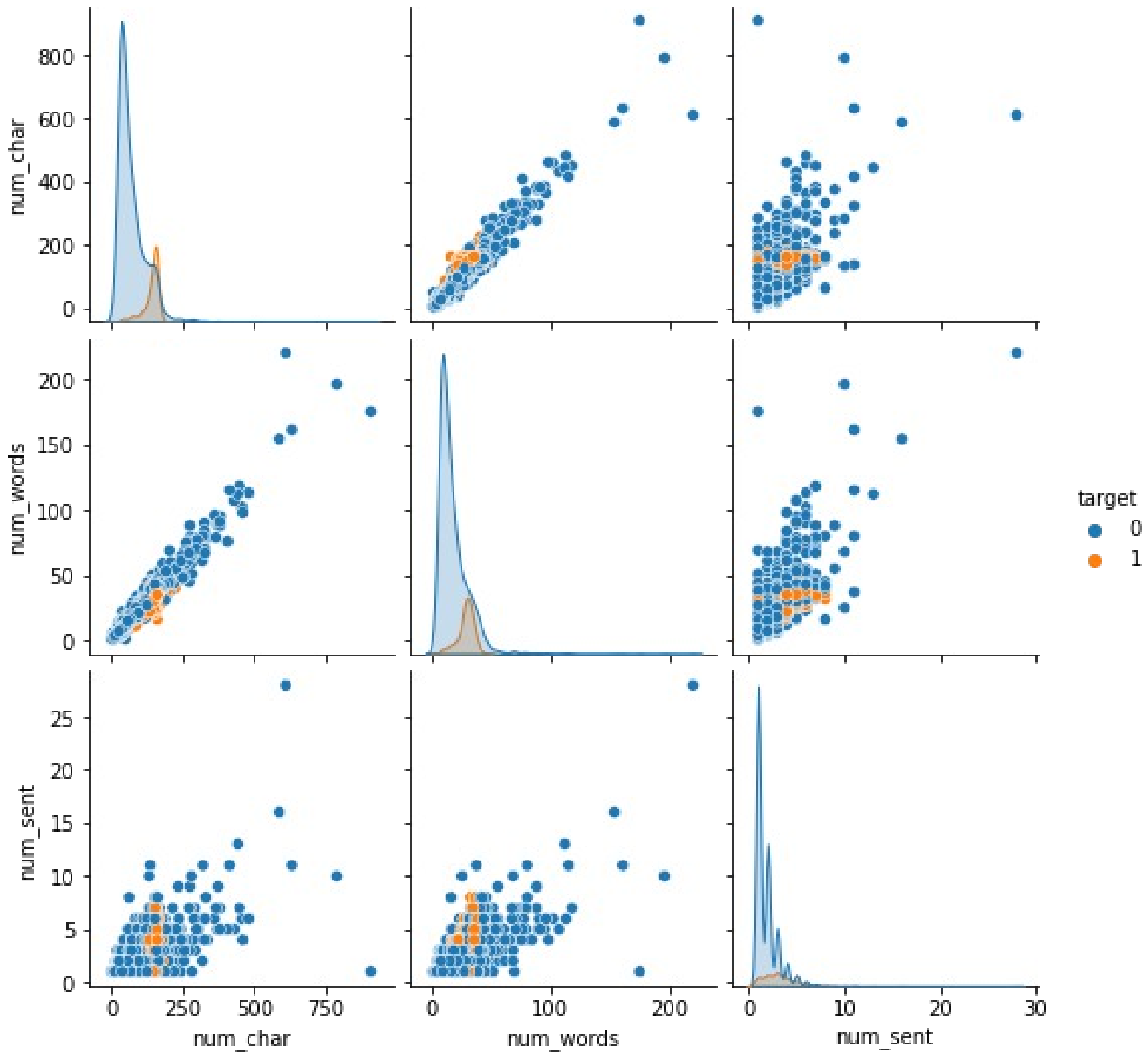
```
plt.figure(figsize=(14,6))
sns.histplot(df[df['target']==0]['num_words'])
sns.histplot(df[df['target']==1]['num_words'],color='yellow')
plt.show()
```



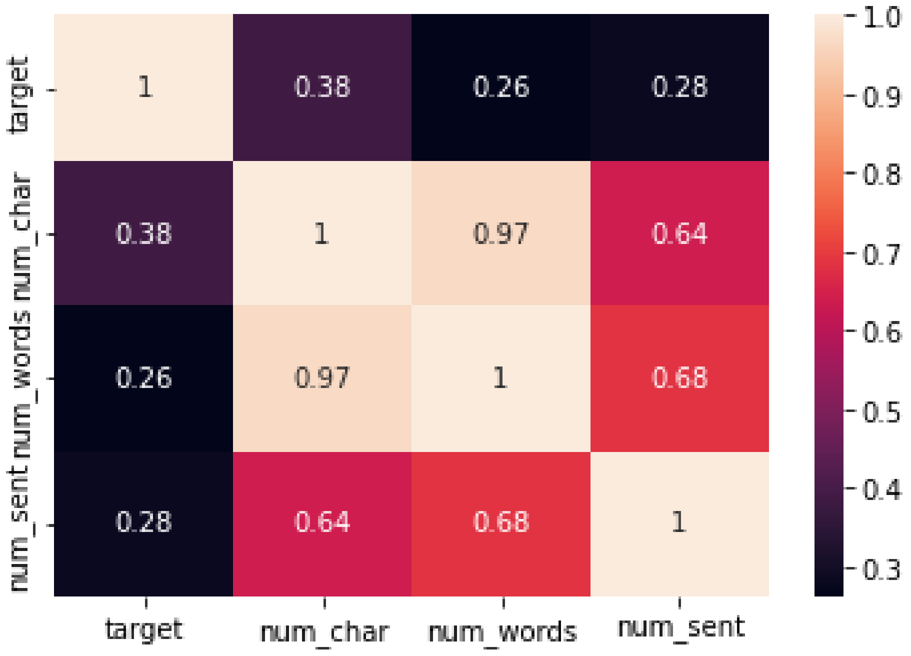
```
plt.figure(figsize=(14,6))
sns.histplot(df[df['target']==0]['num_sent'])
sns.histplot(df[df['target']==1]['num_sent'],color='yellow')
plt.show()
```



```
sns.pairplot(df,hue='target')

<seaborn.axisgrid.PairGrid at 0x2965dfb9730>
```

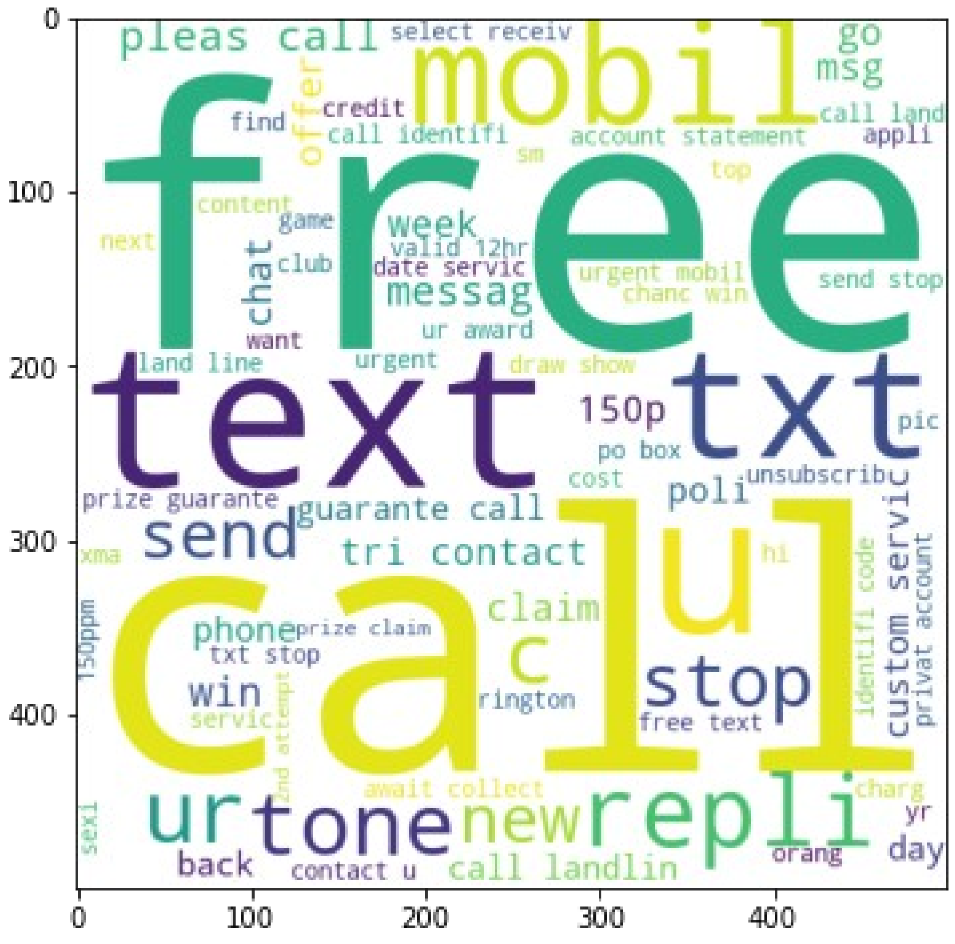


```
sns.heatmap(df.corr(),annot=True)

<AxesSubplot:>
```

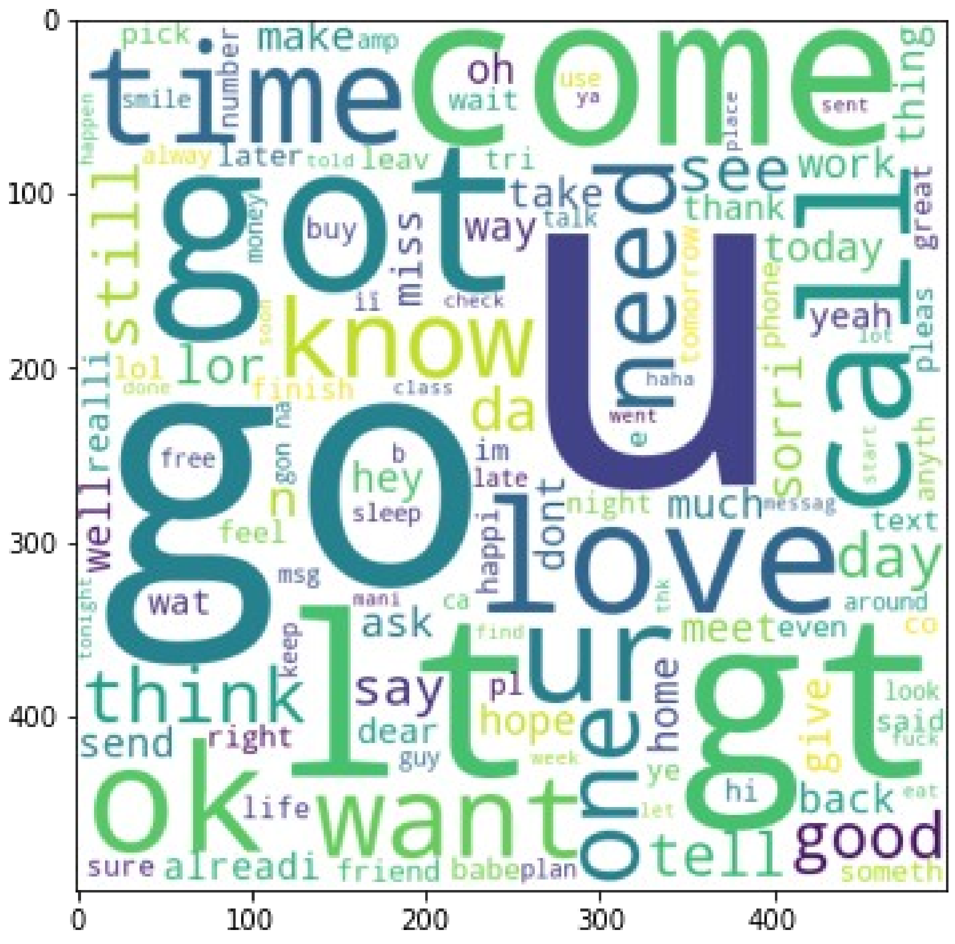


```
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
import string

nltk.download('stopwords')
ps=PorterStemmer()

def transform_text(text):
    text=text.lower()
    text=nltk.word_tokenize(text)


    y=[]
    for i in text:
        if i.isalnum():
            y.append(i)

    text=y[:]
    y.clear()

    for i in text:
        if i not in stopwords.words('english') and i not in
string.punctuation:
            y.append(i)

    text=y[:]
    y.clear()

    for i in text:
        y.append(ps.stem(i))


    return " ".join(y)

transformed_text=transform_text("Go until jurong point, crazy..
Available only in bugis n great world la e buffet... Cine there got
amore wat...")
print(transformed_text)

go jurong point crazi avail bugi n great world la e buffet cine got
amor wat

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

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

df.head()

   target                                               text  num_char
\
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


   num_words  num_sent
transformed_text
0         24         2  go jurong point crazi avail bugi n great
world...
1          8         2                              ok lar joke wif u
oni
2         37         2  free entri 2 wkli comp win fa cup final tkt
21...
3         13         1                 u dun say earli hor u c alreadi
say
4         15         1                nah think goe usf live around
though

from wordcloud import WordCloud
wc=WordCloud(width=500,height=500,min_font_size=10,background_color='w
hite')
```

```
spam_wc=wc.generate(df[df['target']==1]
['transformed_text'].str.cat(sep=" "))

plt.figure(figsize=(15,6))
plt.imshow(spam_wc)

<matplotlib.image.AxesImage at 0x2965f3c98e0>
```

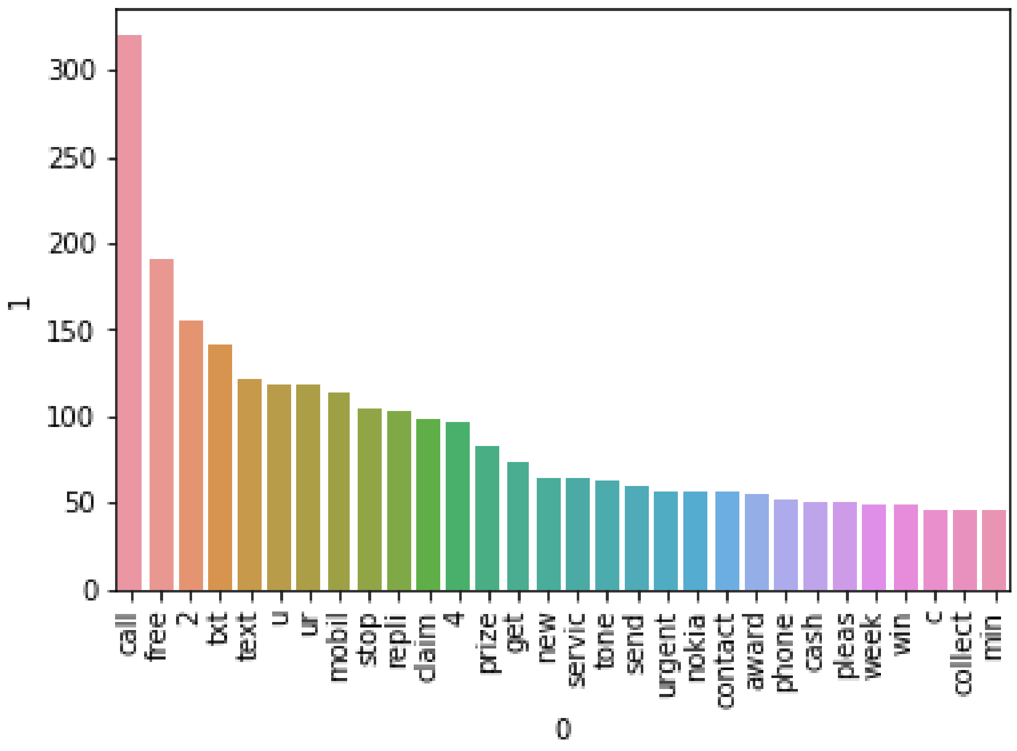


```
ham_wc=wc.generate(df[df['target']==0]
['transformed_text'].str.cat(sep=" "))

plt.figure(figsize=(15,6))
plt.imshow(ham_wc)

<matplotlib.image.AxesImage at 0x296601c5fd0>
```



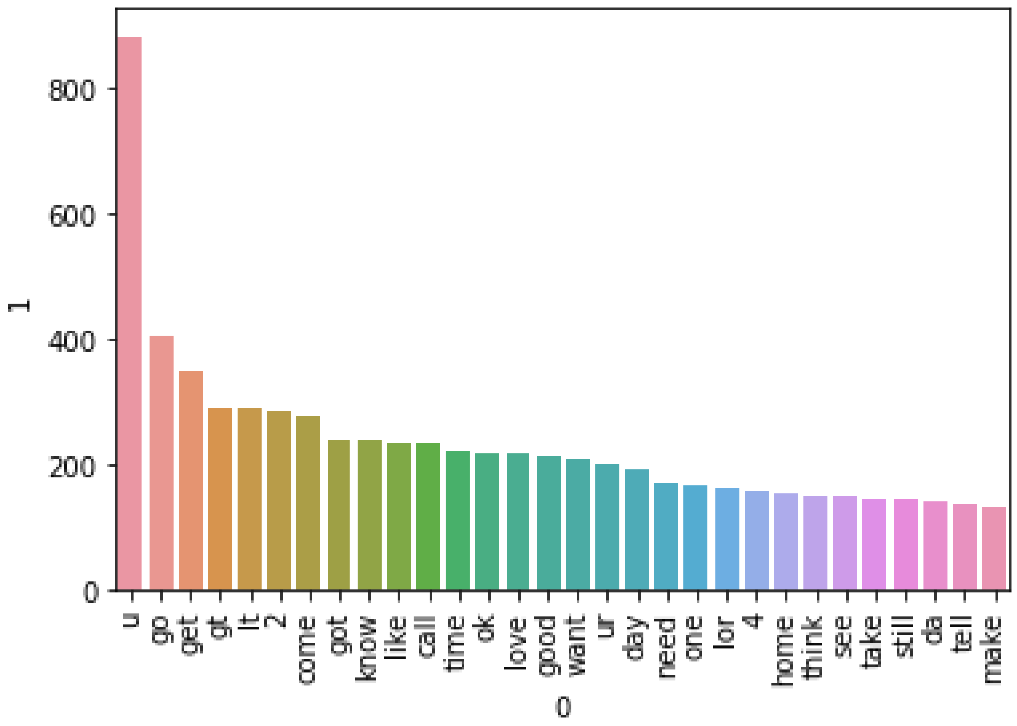
```
df.head()

   target                                               text  num_char  \
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


   num_words  num_sent
transformed_text
0         24         2  go jurong point crazi avail bugi n great
world...
1          8         2                              ok lar joke wif u
oni
```

```
2          37          2  free entri 2 wkli comp win fa cup final tkt
21...
3          13          1                 u dun say earli hor u c alreadi
say
4          15          1                nah think goe usf live around
though

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

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()
```



```
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

from collections import Counter
sns.barplot(pd.DataFrame(Counter(ham_corpus).most_common(30))
[0],pd.DataFrame(Counter(ham_corpus).most_common(30))[1])
plt.xticks(rotation='vertical')
plt.show()
```



```
# text vectorisation using bag of words
df.head()

   target                                               text  num_char
\
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


   num_words  num_sent
```

```
transformed_text
0         24         2  go jurong point crazi avail bugi n great
world...
1          8         2                            ok lar joke wif u
oni
2         37         2  free entri 2 wkli comp win fa cup final tkt
21...
3         13         1                 u dun say earli hor u c alreadi
say
4         15         1                nah think goe usf live around
though
```

building The model

```
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

(5169, 3000)

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,rando
m_state=2)

from sklearn.naive_bayes import GaussianNB,MultinomialNB,BernoulliNB
from sklearn.metrics import accuracy_score,
confusion_matrix,precision_score

gnb=GaussianNB()
mnb=MultinomialNB()
bnb=BernoulliNB()

gnb.fit(x_train,y_train)
y_pred1=gnb.predict(x_test)
print(accuracy_score(y_test,y_pred1))
print(confusion_matrix(y_test,y_pred1))
print(precision_score(y_test,y_pred1))

0.8694390715667312
[[788 108]
 [ 27 111]]
0.5068493150684932
```

```
def accuracy(algo):
    trf=algo()
    trf.fit(x_train,y_train)
    y_pred1=trf.predict(x_test)
    print(accuracy_score(y_test,y_pred1))
    print(confusion_matrix(y_test,y_pred1))
    print(precision_score(y_test,y_pred1))
```

```
accuracy(GaussianNB)
```

```
0.8694390715667312
[[788 108]
 [ 27 111]]
0.5068493150684932
```

```
accuracy(MultinomialNB)
```

```
0.9709864603481625
[[896   0]
 [ 30 108]]
1.0
```

```
accuracy(BernoulliNB)
```

```
0.9835589941972921
[[895   1]
 [ 16 122]]
0.991869918699187
```

```
!pip install xgboost
```

```
Requirement already satisfied: xgboost in c:\users\dhiraj kumar\
anaconda3\lib\site-packages (2.1.1)
Requirement already satisfied: numpy in c:\users\dhiraj kumar\
anaconda3\lib\site-packages (from xgboost) (1.22.4)
Requirement already satisfied: scipy in c:\users\dhiraj kumar\
anaconda3\lib\site-packages (from xgboost) (1.7.3)
```

```
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
```

```python
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)

clfs = {
'SVC': svc,
'KN': knc,
'NB': mnb,
'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)
```

```
(0.9758220502901354, 0.9747899159663865)
```

```python
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 -  0.9294003868471954
Precision -  0.8282828282828283
For  LR
Accuracy -  0.9584139264990329
Precision -  0.9702970297029703
For  RF
Accuracy -  0.9748549323017408
Precision -  0.9827586206896551
For  AdaBoost
Accuracy -  0.960348162475822
Precision -  0.9292035398230089
For  BgC
Accuracy -  0.9574468085106383
Precision -  0.8671875
For  ETC
Accuracy -  0.9748549323017408
Precision -  0.9745762711864406
For  GBDT
Accuracy -  0.9477756286266924
Precision -  0.92
For  xgb
Accuracy -  0.9661508704061895
Precision -  0.9256198347107438
```

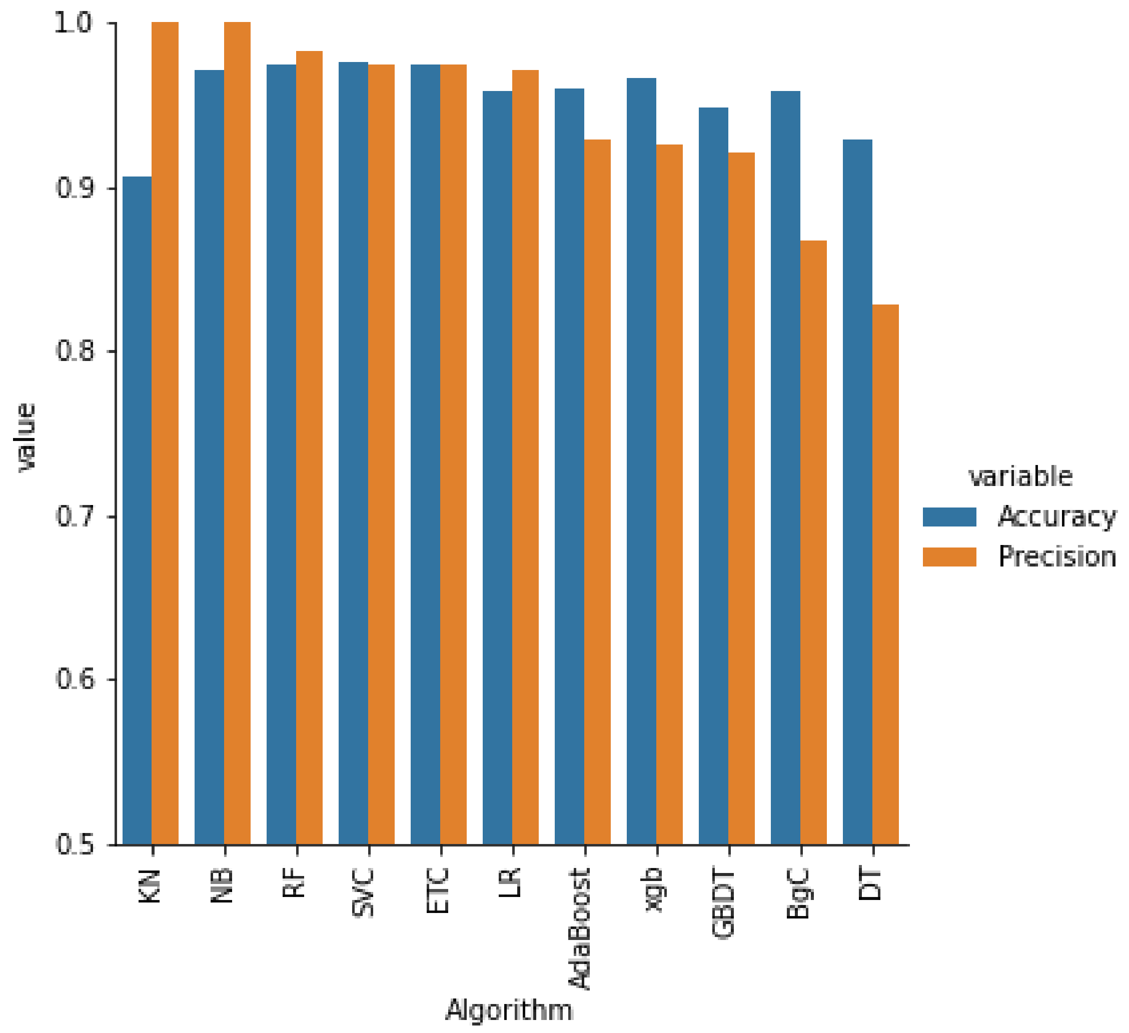
```
performance_df=pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy':accura
cy_scores,'Precision':precision_scores}).sort_values('Precision',ascen
ding=False)

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
6   AdaBoost  0.960348   0.929204
10       xgb  0.966151   0.925620
9       GBDT  0.947776   0.920000
7        BgC  0.957447   0.867188
3         DT  0.929400   0.828283
```

```
performance_df1= pd.melt(performance_df,id_vars="Algorithm")

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   AdaBoost   Accuracy  0.960348
7        xgb   Accuracy  0.966151
8       GBDT   Accuracy  0.947776
9        BgC   Accuracy  0.957447
10        DT   Accuracy  0.929400
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  AdaBoost  Precision  0.929204
18       xgb  Precision  0.925620
19      GBDT  Precision  0.920000
20       BgC  Precision  0.867188
21        DT  Precision  0.828283
```

```
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()
```



```
#model improve
# 1. change the max_features parameter of IfIdf

temp_df=pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy_max_ft_3000':a
ccuracy_scores,'precision_max_ft_3000':precision_scores}).sort_values(
'precision_max_ft_3000',ascending=False)

new_df=performance_df.merge(temp_df,on='Algorithm')

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

temp_df=pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy_num_chars':acc
uracy_scores,'precision_num_chars':precision_scores}).sort_values('pre
cision_num_chars',ascending=False)

# 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.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')

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

# Applying stocking
estimators=[('svm',svc),('nb',mnb),('et',etc)]
final_estimator=RandomForestClassifier()

from sklearn.ensemble import StackingClassifier

clf=StackingClassifier(estimators=estimators,final_estimator=final_est
imator)

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 0.9806576402321083
Precision 0.946969696969697

import pickle
pickle.dump(tfidf,open('vectorizer.pkl','wb'))
pickle.dump(mnb,open('model.pkl','wb'))

import pickle
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB

# sample test data and corresponding labels (replaced with your actual
data)
```

```
x_train=["Sample text 1","Sample text 2","Sample text 3"]
y_train=[0,1,0]  # Examples labels (0 for negative, 1 for positive)

#create ans train tf-IDF vectorizer
tfidf =TfidfVectorizer(lowercase=True,stop_words='english')
x_train_tfidf=tfidf.fit_transform(x_train)

#create and train the naive bayes classifier
mnb=MultinomialNB()
mnb.fit(x_train_tfidf,y_train)

# savee the trained TF-IDF vectorizer and naive bayes model to files
with open('vectorizer.pkl','wb') as vectorizer_file:
    pickle.dump(tfidf,vectorizer_file)
with open('model.pkl','wb')as model_file:
    pickle.dump(mnb,model_file)
```

```python
import streamlit as st
import pickle
from sklearn.feature_extraction.text import TfidfVectorizer

#lets load the saved vectorizer and naive model
tfidf=pickle.load(open('vectorizer.pkl','rb'))
model=pickle.load(open('model.pkl','rb'))




#transform text function for text preprocessing
import nltk
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
import string
nltk.download('punkt')
nltk.download('stopwords')
ps =PorterStemmer()

def transform_text(text):
      text= text.lower() #Converting to lowercase
      text= nltk.word_tokenize(text) #Tokenize Removing special characters and retaining
alphanumeric words
      text=[word for word in text if word.isalnum()]
    #   Removing stopwords and punctuation
      text= [word for word in text if word not in stopwords.words('english') and word not in
string.punctuation]
      #Applying stemning
      text = [ps.stem(word) for word in text]

      return " ".join(text)

#saving streamlit code

st.title("Email spam Classifier")
input_sms=st.text_area("Enter message")


if st.button('predict'):
    #preprocess
    tranformed_sms=transform_text(input_sms)
    #vectorize
    vector_input=tfidf.transform([tranformed_sms])
    #predict
    result=model.predict(vector_input)[0]
    #display
    if result ==1:
        st.header("spam")
    else:
        st.header("Not Spam")
```