```
import numpy as np
import pandas as pd

df = pd.read_csv('spam.csv',encoding= "latin-1")
```

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

```
       v1                                                 v2 Unnamed:
2  \
612   ham                             I have many dependents
NaN
4350  ham  Night has ended for another day, morning has c...
NaN
4456  ham  Aight should I just plan to come up later toni...
NaN
4486  ham  Miss call miss call khelate kintu opponenter m...
NaN
789   ham                       Gud mrng dear hav a nice day
NaN

     Unnamed: 3 Unnamed: 4
612         NaN        NaN
4350        NaN        NaN
4456        NaN        NaN
4486        NaN        NaN
789         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.sample(5)
```

```
       v1                                                 v2 Unnamed:
2  \
5466  spam  http//tms. widelive.com/index. wml?id=820554ad...
NaN
1212   ham  Yo, the game almost over? Want to go to walmar...
NaN
2958   ham  Buzzzz! *grins* Did I buzz your ass? Buzz your...
NaN
7      ham  As per your request 'Melle Melle (Oru Minnamin...
NaN
4591   ham  Right it wasnt you who phoned it was someone w...
NaN

     Unnamed: 3 Unnamed: 4
5466        NaN        NaN
1212        NaN        NaN
2958        NaN        NaN
7           NaN        NaN
4591        NaN        NaN
```

```
# renaming the cols
df.rename(columns={'v1':'target','v2':'text'},inplace=True)
df.sample(5)
```

```
     target                                               text
Unnamed: 2  \
1394    ham               R we still meeting 4 dinner tonight?
NaN
1906    ham  And stop being an old man. You get to build sn...
NaN
4016    ham  Eek that's a lot of time especially since Amer...
NaN
922     ham  It shall be fine. I have avalarr now. Will hol...
NaN
5244    ham  thanks for the temales it was wonderful. Thank...
NaN

     Unnamed: 3 Unnamed: 4
1394        NaN        NaN
1906        NaN        NaN
4016        NaN        NaN
922         NaN        NaN
5244        NaN        NaN
```

```
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()

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

df.head()
```

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

  Unnamed: 3 Unnamed: 4
0        NaN        NaN
1        NaN        NaN
2        NaN        NaN
3        NaN        NaN
4        NaN        NaN
```

```
# missing values
df.isnull().sum()
```

```
target           0
text             0
Unnamed: 2    5522
Unnamed: 3    5560
Unnamed: 4    5566
dtype: int64
```

```
# check for duplicate values
df.duplicated().sum()
```

```
403
```

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

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

```
0
```

```
df.shape
```

```
(5169, 5)
```

2.EDA

```
df.head()
```

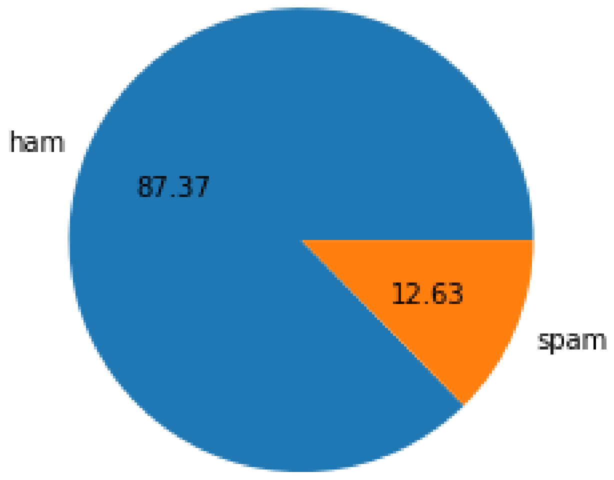
```
   target                                               text Unnamed:
2  \
0       0  Go until jurong point, crazy.. Available only ...
NaN
1       0                      Ok lar... Joking wif u oni...
NaN
2       1  Free entry in 2 a wkly comp to win FA Cup fina...
NaN
3       0  U dun say so early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

  Unnamed: 3 Unnamed: 4
0        NaN        NaN
1        NaN        NaN
2        NaN        NaN
3        NaN        NaN
4        NaN        NaN
```

```
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()
```



```
# Data is imbalanced
```

```
import nltk
```

```
!pip install nltk
```

```
Requirement already satisfied: nltk in c:\users\dell\anaconda3\lib\
site-packages (3.6.5)
Requirement already satisfied: click in c:\users\dell\anaconda3\lib\
site-packages (from nltk) (8.0.3)
Requirement already satisfied: joblib in c:\users\dell\anaconda3\lib\
site-packages (from nltk) (1.1.0)
Requirement already satisfied: regex>=2021.8.3 in c:\users\dell\
anaconda3\lib\site-packages (from nltk) (2021.8.3)
Requirement already satisfied: tqdm in c:\users\dell\anaconda3\lib\
site-packages (from nltk) (4.62.3)
Requirement already satisfied: colorama in c:\users\dell\anaconda3\
lib\site-packages (from click->nltk) (0.4.4)
```

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

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

True
```

```
df['num_characters'] = df['text'].apply(len)
```

```
df.head()
```

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

  Unnamed: 3 Unnamed: 4  num_characters
0        NaN        NaN             111
1        NaN        NaN              29
2        NaN        NaN             155
3        NaN        NaN              49
4        NaN        NaN              61
```

```
# num of words
df['num_words'] = df['text'].apply(lambda
x:len(nltk.word_tokenize(x)))
```

```
df.head()
```

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

  Unnamed: 3 Unnamed: 4  num_characters  num_words
0        NaN        NaN             111         24
1        NaN        NaN              29          8
2        NaN        NaN             155         37
3        NaN        NaN              49         13
4        NaN        NaN              61         15
```

```
df['num_sentences'] = df['text'].apply(lambda
x:len(nltk.sent_tokenize(x)))

df.head()
```

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

  Unnamed: 3 Unnamed: 4  num_characters  num_words  num_sentences
0        NaN        NaN             111         24              2
1        NaN        NaN              29          8              2
2        NaN        NaN             155         37              2
3        NaN        NaN              49         13              1
4        NaN        NaN              61         15              1
```

```
df[['num_characters','num_words','num_sentences']].describe()
```

```
       num_characters    num_words  num_sentences
count     5169.000000  5169.000000    5169.000000
mean        78.977945    18.455407       1.961308
std         58.236293    13.322448       1.432583
```

```
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
```

```
# ham
df[df['target'] == 0]
[['num_characters','num_words','num_sentences']].describe()
```

```
       num_characters    num_words  num_sentences
count     4516.000000  4516.000000    4516.000000
mean        70.459256    17.123339       1.815545
std         56.358207    13.491315       1.364098
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
```

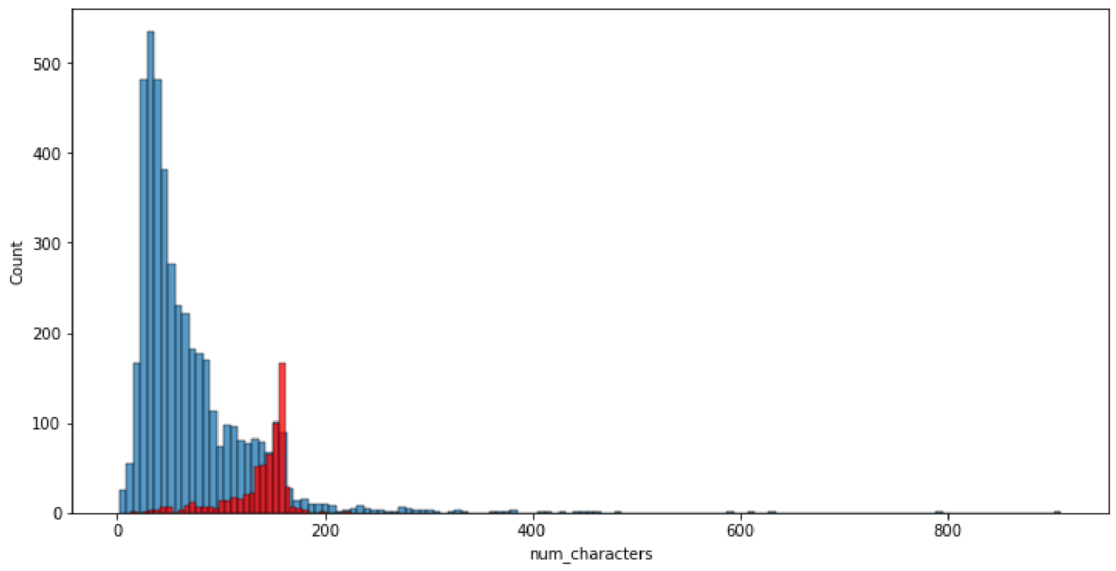
```
#spam
df[df['target'] == 1]
[['num_characters','num_words','num_sentences']].describe()
```

```
       num_characters   num_words  num_sentences
count      653.000000  653.000000     653.000000
mean       137.891271   27.667688       2.969372
std         30.137753    7.008418       1.488910
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
```

```
import seaborn as sns
```

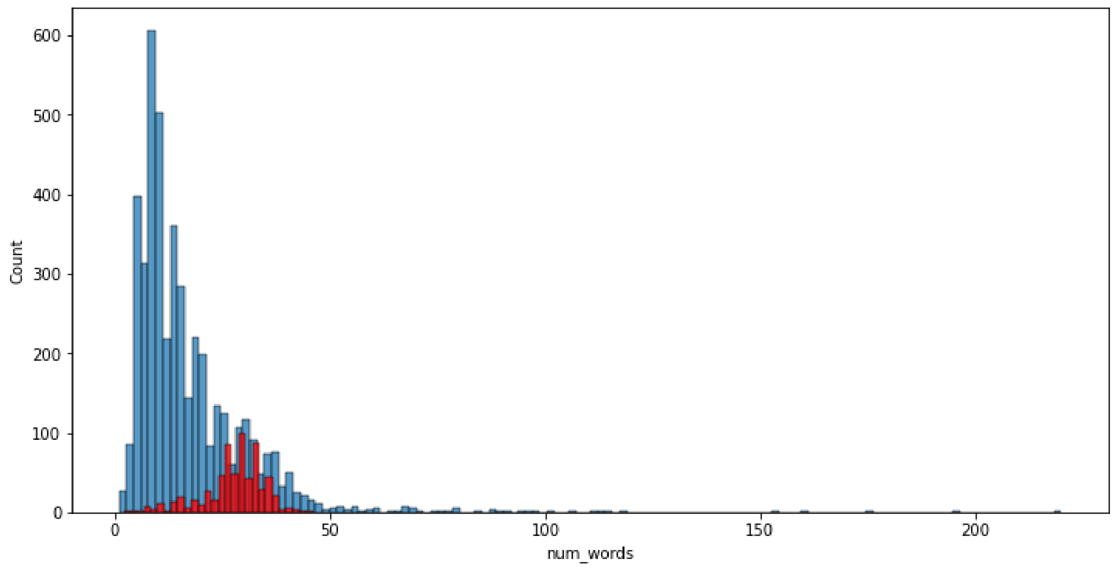
```
plt.figure(figsize=(12,6))
sns.histplot(df[df['target'] == 0]['num_characters'])
sns.histplot(df[df['target'] == 1]['num_characters'],color='red')
```

```
<AxesSubplot:xlabel='num_characters', ylabel='Count'>
```



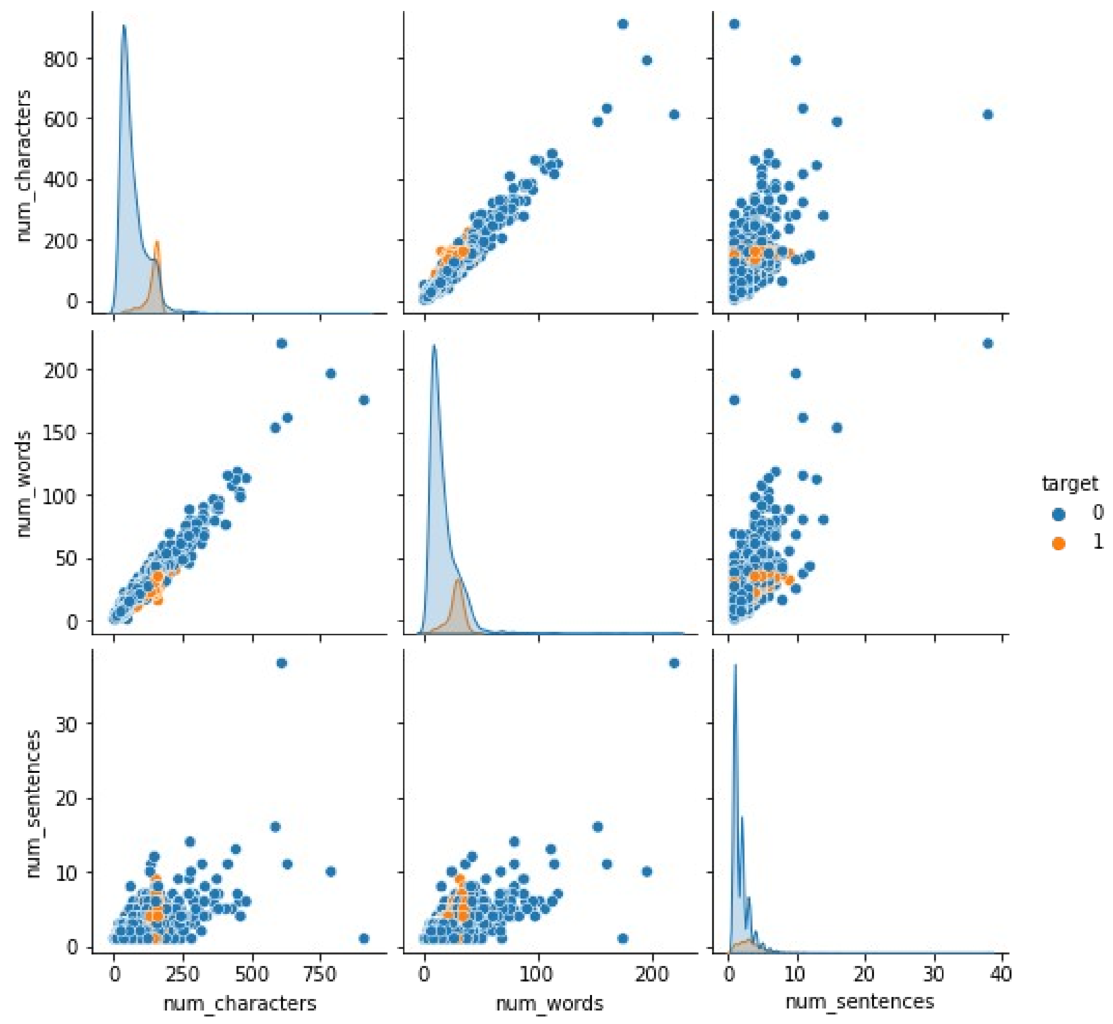
```
plt.figure(figsize=(12,6))
sns.histplot(df[df['target'] == 0]['num_words'])
sns.histplot(df[df['target'] == 1]['num_words'],color='red')
```

```
<AxesSubplot:xlabel='num_words', ylabel='Count'>
```



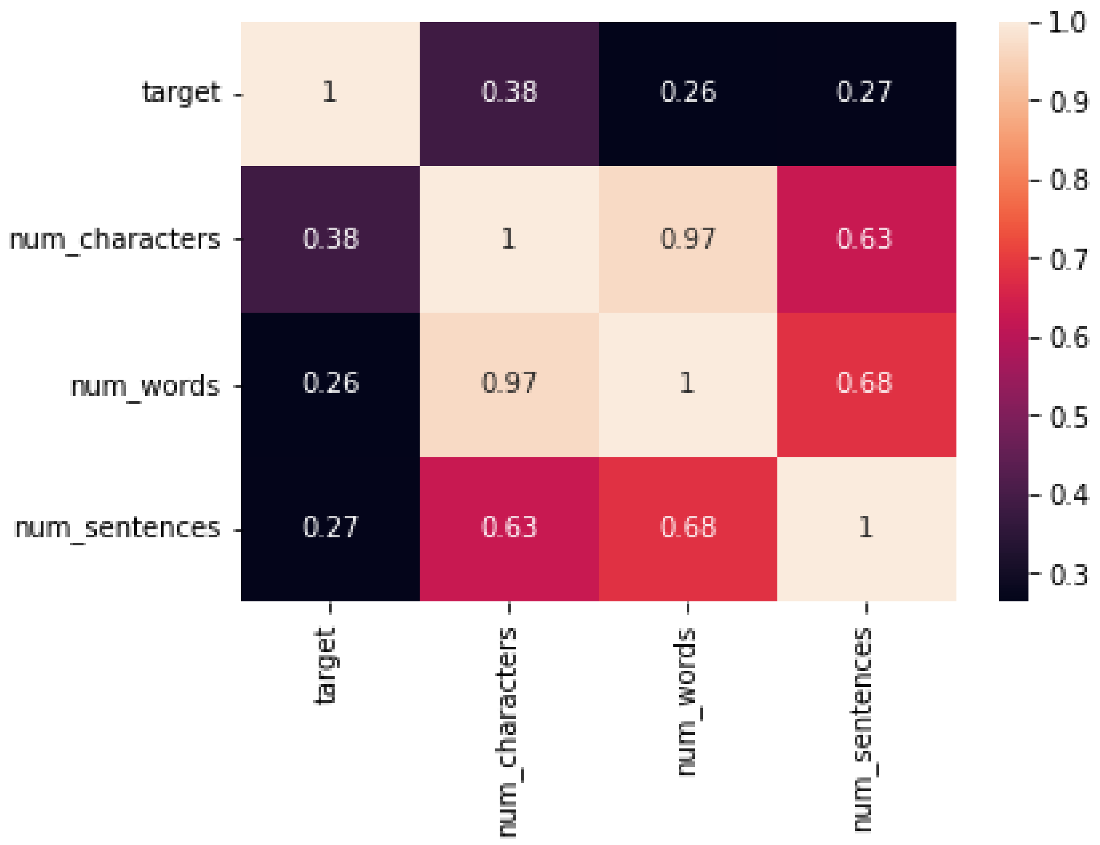
```
sns.pairplot(df,hue='target')
```

```
<seaborn.axisgrid.PairGrid at 0x1f7d96d0e80>
```



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

```
<AxesSubplot:>
```



3. Data Preprocessing

```
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords

import string
string.punctuation

from nltk.stem.porter import PorterStemmer
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)
```

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

```
transform_text("I'm gonna be home soon and i don't want to talk about 
this stuff anymore tonight, k? I've cried enough today.")
```

```
'gon na home soon want talk stuff anymore tonight k cried enough today 
gon na home soon want talk stuff anymor tonight k cri enough today'
```

```
df['text'][10]
```

```
"I'm gonna be home soon and i don't want to talk about this stuff 
anymore tonight, k? I've cried enough today."
```

```
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()
ps.stem('loving')
```

```
'love'
```

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

```
df.head()
```

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

```
  Unnamed: 3 Unnamed: 4  num_characters  num_words  num_sentences  \
0        NaN        NaN             111         24              2
1        NaN        NaN              29          8              2
2        NaN        NaN             155         37              2
3        NaN        NaN              49         13              1
4        NaN        NaN              61         15              1

                                    transformed_text
0  go jurong point crazy available bugis n great ...
1      ok lar joking wif u oni ok lar joke wif u oni
2  free entry 2 wkly comp win fa cup final tkts 2...
3  u dun say early hor u c already say u dun say ...
4  nah think goes usf lives around though nah thi...
```

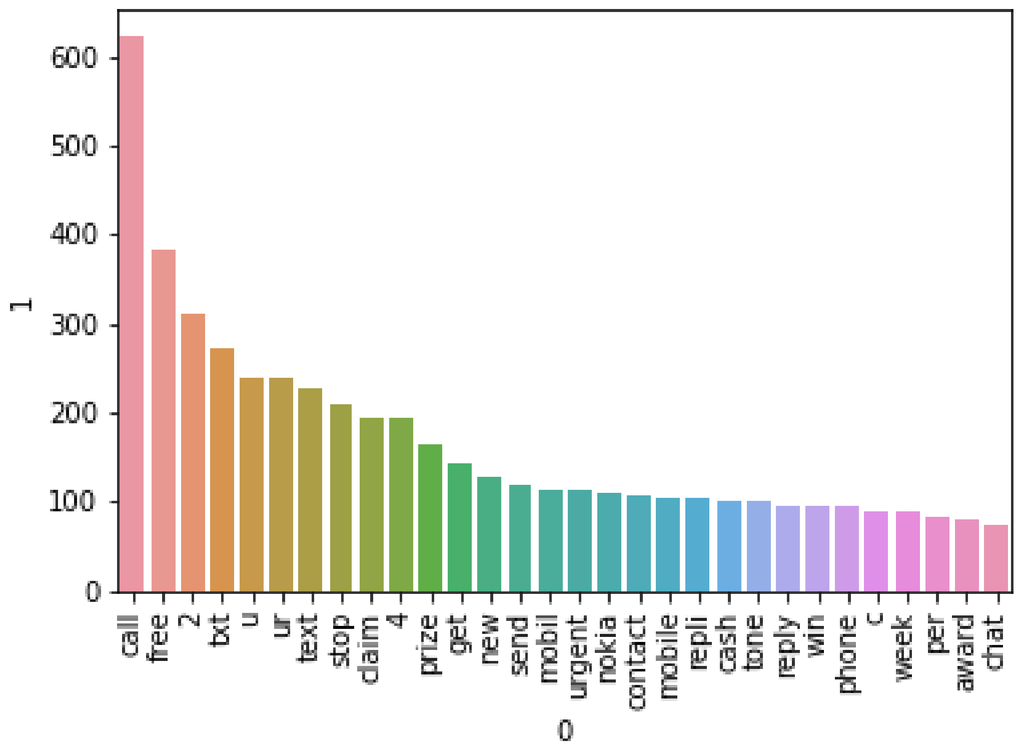
```
spam_corpus = []
for msg in df[df['target'] == 1]['transformed_text'].tolist():
    for word in msg.split():
        spam_corpus.append(word)
```

```
len(spam_corpus)
```

```
19878
```

```
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()
```

```
C:\Users\Dell\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y.
From version 0.12, the only valid positional argument will be `data`,
and passing other arguments without an explicit keyword will result in
an error or misinterpretation.
  warnings.warn(
```



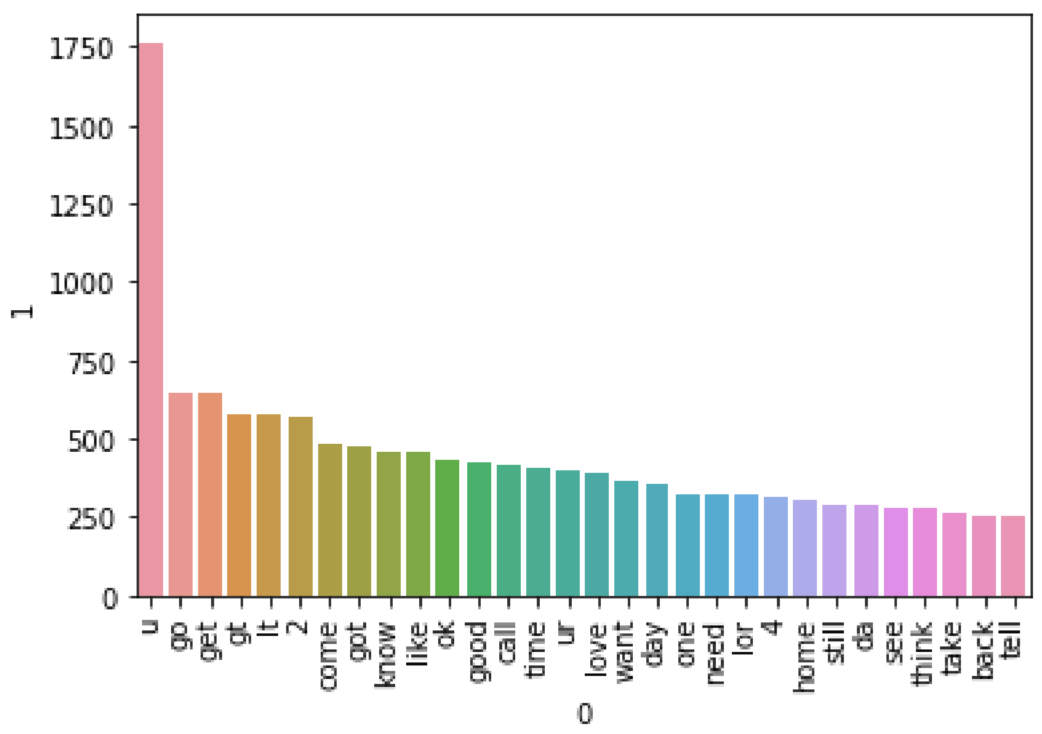
```
ham_corpus = []
for msg in df[df['target'] == 0]['transformed_text'].tolist():
    for word in msg.split():
        ham_corpus.append(word)
```

```
len(ham_corpus)
```

```
70804
```

```
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()
```

```
C:\Users\Dell\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y.
From version 0.12, the only valid positional argument will be `data`,
and passing other arguments without an explicit keyword will result in
an error or misinterpretation.
  warnings.warn(
```



```
df.head()

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

  Unnamed: 3 Unnamed: 4  num_characters  num_words  num_sentences  \
0        NaN        NaN             111         24              2
1        NaN        NaN              29          8              2
2        NaN        NaN             155         37              2
3        NaN        NaN              49         13              1
4        NaN        NaN              61         15              1

                                    transformed_text
0  go jurong point crazy available bugis n great ...
1      ok lar joking wif u oni ok lar joke wif u oni
2  free entry 2 wkly comp win fa cup final tkts 2...
3  u dun say early hor u c already say u dun say ...
4  nah think goes usf lives around though nah thi...
```

4. Model Building

```
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,random_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))

0.874274661508704
[[787 109]
 [ 21 117]]
0.5176991150442478

mnb.fit(X_train,y_train)
y_pred2 = mnb.predict(X_test)
print(accuracy_score(y_test,y_pred2))
print(confusion_matrix(y_test,y_pred2))
print(precision_score(y_test,y_pred2))

0.9748549323017408
[[896   0]
 [ 26 112]]
1.0

bnb.fit(X_train,y_train)
y_pred3 = bnb.predict(X_test)
print(accuracy_score(y_test,y_pred3))
print(confusion_matrix(y_test,y_pred3))
print(precision_score(y_test,y_pred3))
```

```
0.9825918762088974
[[895   1]
 [ 17 121]]
0.9918032786885246
```

```
# tfidf --> MNB
```

```
pip install xgboost
```

```
Collecting xgboost
  Downloading xgboost-1.5.2-py3-none-win_amd64.whl (106.6 MB)
Requirement already satisfied: numpy in c:\users\dell\anaconda3\lib\
site-packages (from xgboost) (1.20.3)
Requirement already satisfied: scipy in c:\users\dell\anaconda3\lib\
site-packages (from xgboost) (1.7.1)
Installing collected packages: xgboost
Successfully installed xgboost-1.5.2
Note: you may need to restart the kernel to use updated packages.
```

```
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()
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)

(0.9748549323017408, 0.9745762711864406)

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 -  0.9748549323017408
Precision -  0.9745762711864406
For  KN
Accuracy -  0.9061895551257253
Precision -  1.0
For  NB
Accuracy -  0.9748549323017408
Precision -  1.0
For  DT
Accuracy -  0.937137330754352
Precision -  0.8543689320388349
For  LR
Accuracy -  0.9545454545454546
Precision -  0.9595959595959596
For  RF
```

```
Accuracy -  0.97678916827853
Precision -  0.9830508474576272
For  AdaBoost
Accuracy -  0.9680851063829787
Precision -  0.9487179487179487
For  BgC
Accuracy -  0.9613152804642167
Precision -  0.8951612903225806
For  ETC
Accuracy -  0.9748549323017408
Precision -  0.9590163934426229
For  GBDT
Accuracy -  0.9477756286266924
Precision -  0.9468085106382979

C:\Users\Dell\anaconda3\lib\site-packages\xgboost\sklearn.py:1224: 
UserWarning: The use of label encoder in XGBClassifier is deprecated 
and will be removed in a future release. To remove this warning, do 
the following: 1) Pass option use_label_encoder=False when 
constructing XGBClassifier object; and 2) Encode your labels (y) as 
integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[16:40:33] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0, 
the default evaluation metric used with the objective 
'binary:logistic' was changed from 'error' to 'logloss'. Explicitly 
set eval_metric if you'd like to restore the old behavior.
For  xgb
Accuracy -  0.9661508704061895
Precision -  0.9557522123893806
```

```
performance_df = 
pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy':accuracy_scores,'Prec
ision':precision_scores}).sort_values('Precision',ascending=False)

performance_df
```

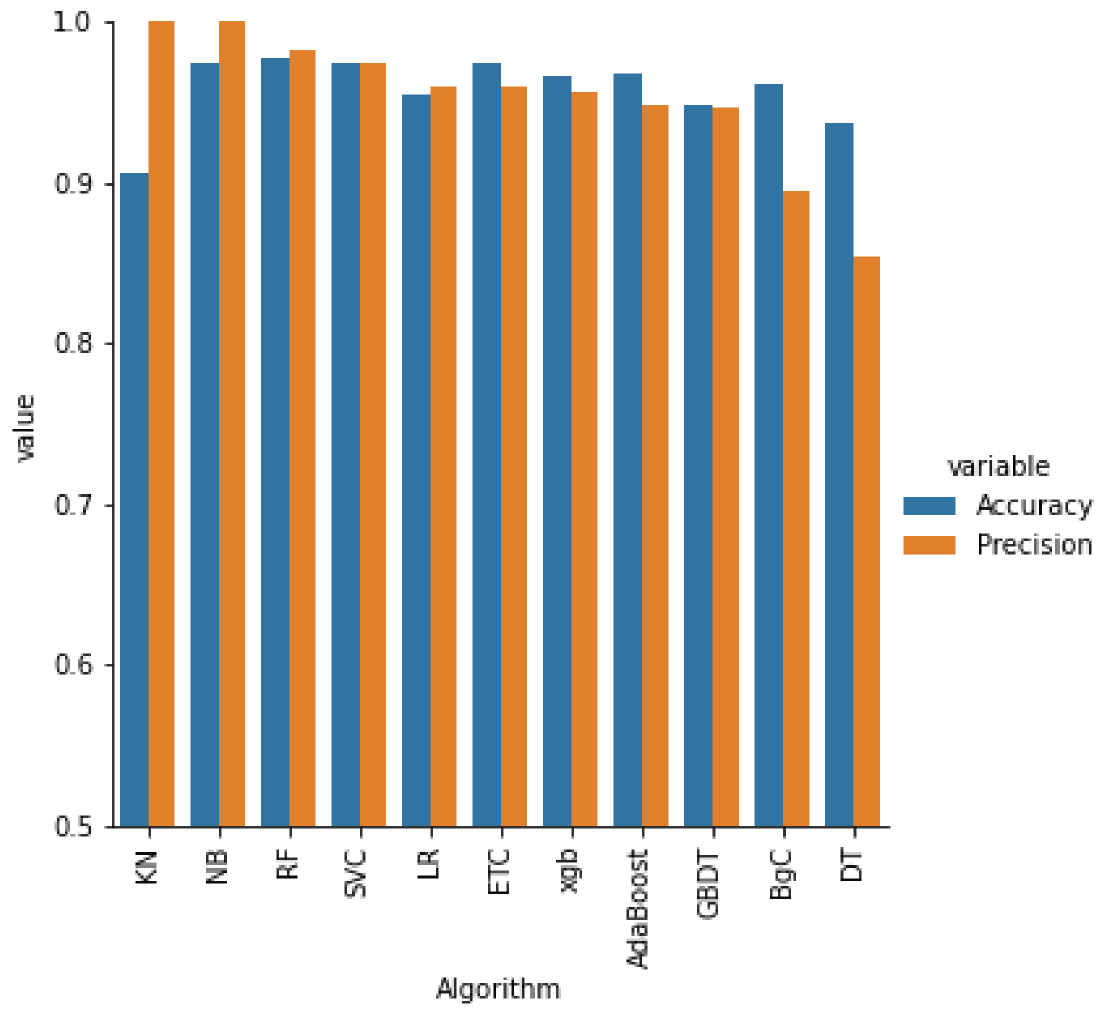
```
   Algorithm  Accuracy  Precision
1         KN  0.906190   1.000000
2         NB  0.974855   1.000000
5         RF  0.976789   0.983051
0        SVC  0.974855   0.974576
4         LR  0.954545   0.959596
8        ETC  0.974855   0.959016
10       xgb  0.966151   0.955752
6   AdaBoost  0.968085   0.948718
9       GBDT  0.947776   0.946809
7        BgC  0.961315   0.895161
3         DT  0.937137   0.854369
```

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

```
performance_df1
```

```
   Algorithm   variable     value
0         KN   Accuracy  0.906190
1         NB   Accuracy  0.974855
2         RF   Accuracy  0.976789
3        SVC   Accuracy  0.974855
4         LR   Accuracy  0.954545
5        ETC   Accuracy  0.974855
6        xgb   Accuracy  0.966151
7   AdaBoost   Accuracy  0.968085
8       GBDT   Accuracy  0.947776
9        BgC   Accuracy  0.961315
10        DT   Accuracy  0.937137
11        KN  Precision  1.000000
12        NB  Precision  1.000000
13        RF  Precision  0.983051
14       SVC  Precision  0.974576
15        LR  Precision  0.959596
16       ETC  Precision  0.959016
17       xgb  Precision  0.955752
18  AdaBoost  Precision  0.948718
19      GBDT  Precision  0.946809
20       BgC  Precision  0.895161
21        DT  Precision  0.854369
```

```
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 TfIdf

temp_df =
pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy_max_ft_3000':accuracy_
scores,'Precision_max_ft_3000':precision_scores}).sort_values('Precisi
on_max_ft_3000',ascending=False)

temp_df =
pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy_scaling':accuracy_scor
es,'Precision_scaling':precision_scores}).sort_values('Precision_scali
ng',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':accuracy_sc
ores,'Precision_num_chars':precision_scores}).sort_values('Precision_n
um_chars',ascending=False)

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

   Algorithm  Accuracy  Precision  Accuracy_scaling_x
Precision_scaling_x  \
0         KN  0.906190   1.000000            0.906190
1.000000
1         NB  0.974855   1.000000            0.974855
1.000000
2         RF  0.976789   0.983051            0.976789
0.983051
3        SVC  0.974855   0.974576            0.974855
0.974576
4         LR  0.954545   0.959596            0.954545
0.959596
5        ETC  0.974855   0.959016            0.974855
0.959016
6        xgb  0.966151   0.955752            0.966151
0.955752
7   AdaBoost  0.968085   0.948718            0.968085
0.948718
8       GBDT  0.947776   0.946809            0.947776
0.946809
9        BgC  0.961315   0.895161            0.961315
0.895161
10        DT  0.937137   0.854369            0.937137
0.854369

    Accuracy_scaling_y  Precision_scaling_y  Accuracy_num_chars  \
0             0.906190             1.000000            0.906190
1             0.974855             1.000000            0.974855
2             0.976789             0.983051            0.976789
3             0.974855             0.974576            0.974855
4             0.954545             0.959596            0.954545
5             0.974855             0.959016            0.974855
6             0.966151             0.955752            0.966151
7             0.968085             0.948718            0.968085
8             0.947776             0.946809            0.947776
9             0.961315             0.895161            0.961315
10            0.937137             0.854369            0.937137

    Precision_num_chars
0              1.000000
1              1.000000
2              0.983051
3              0.974576
4              0.959596
5              0.959016
```

```
6              0.955752
7              0.948718
8              0.946809
9              0.895161
10             0.854369
```

```
# 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 0.9796905222437138
Precision 0.9834710743801653
```

```
# 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))

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