```
!pip install nltk

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
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: click in /usr/local/lib/python3.7/dist-
packages (from nltk) (7.1.2)
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-
packages (from nltk) (4.64.1)
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)
```

```
df = pd.read_csv('/content/spam.csv', encoding='latin-1')
df.sample(5)
```

```
        v1                                                 v2 Unnamed:
2  \
1667   ham  So now my dad is gonna call after he gets out ...
NaN
3556   ham        I had it already..sabarish asked me to go..
NaN
2729  spam  Urgent! Please call 09066612661 from your land...
NaN
3866   ham         Alright we're hooked up, where you guys at
NaN
2760   ham         I dont thnk its a wrong calling between us
NaN

     Unnamed: 3 Unnamed: 4
1667        NaN        NaN
3556        NaN        NaN
2729        NaN        NaN
3866        NaN        NaN
2760        NaN        NaN
```

```
df.shape
```

```
(5572, 5)
```

```
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.drop(columns=['Unnamed: 2','Unnamed: 3','Unnamed: 4'],inplace=True)
df.sample(5)
```

```
        v1                                                 v2
4286   ham  I wud never mind if u dont miss me or if u don...
2103   ham  Its a site to simulate the test. It just gives...
4726  spam  Had your mobile 10 mths? Update to the latest ...
2350   ham                You will be in the place of that man
2733   ham                          Do Ì_ noe if ben is going?
```

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

```
     target                                               text
5439    ham       Hey i've booked the 2 lessons on sun liao...
3303    ham  IM GONNAMISSU SO MUCH!!I WOULD SAY IL SEND U A...
1946    ham  Hey we can go jazz power yoga hip hop kb and y...
2536    ham                            You do what all you like
435     ham  The message sent is askin for  &lt;#&gt; dolla...
```

```
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
df['target'] = encoder.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...
```

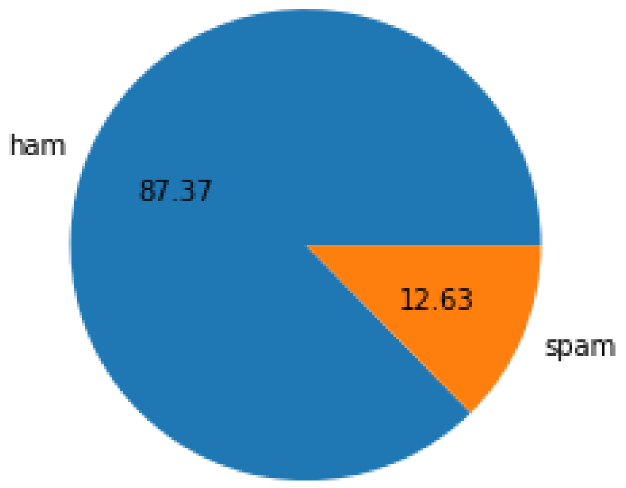
```
df.isnull().sum()
```

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

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

```
403
```

```
df = df.drop_duplicates(keep='first')
df.duplicated().sum()

0

df.shape

(5169, 2)

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

plt.pie(df['target'].value_counts(),
labels=['ham','spam'],autopct="%0.2f")
plt.show()
```



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

[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
```

```
True

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

df['num_words'] = df['text'].apply(lambda
x:len(nltk.word_tokenize(x)))
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

   num_words
0         24
1          8
2         37
3         13
4         15

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

   num_words  num_sentences
0         24              2
1          8              2
2         37              2
3         13              1
4         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    18.453279       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
```

```
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    17.120903       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
```

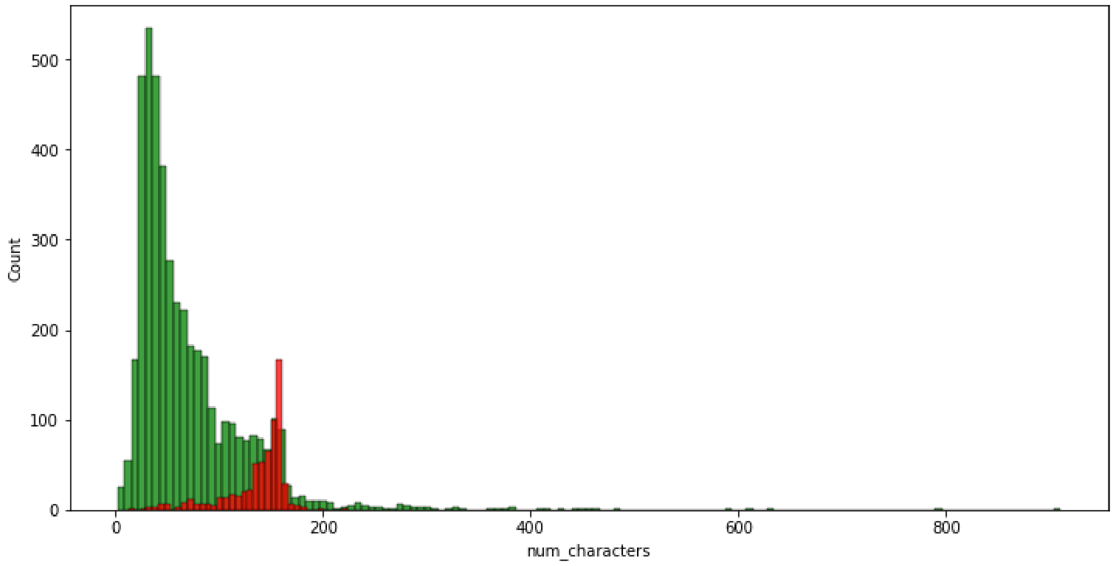
```
df[df['target'] == 1]
[['num_characters','num_words','num_sentences']].describe()
```

```
       num_characters   num_words  num_sentences
count      653.000000  653.000000     653.000000
mean       137.891271   27.667688       2.967841
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
```

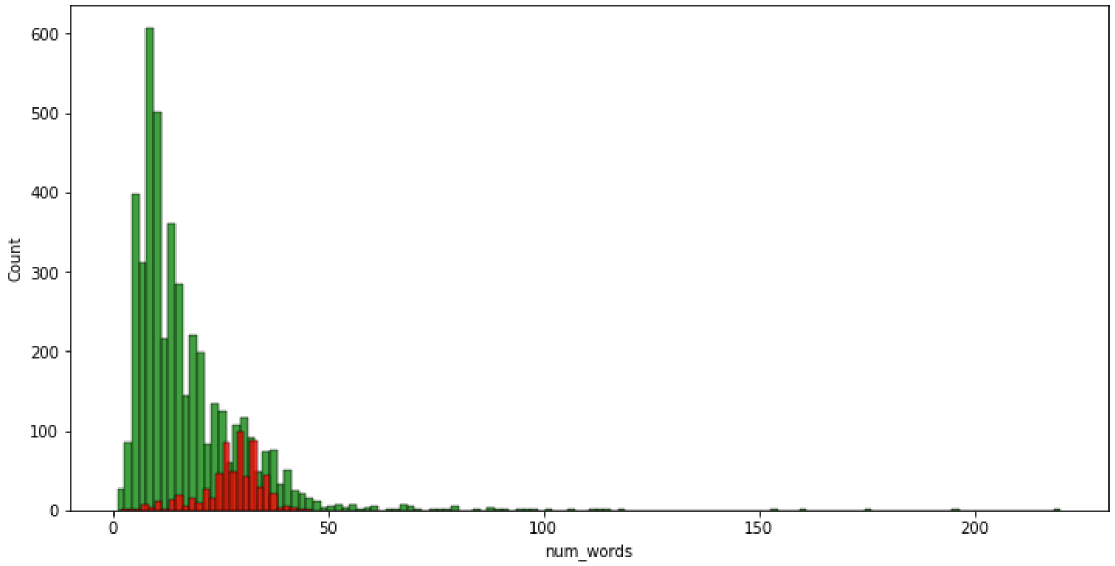
```
plt.figure(figsize=(12,6))
sns.histplot(df[df['target'] == 0]['num_characters'], color='green')
sns.histplot(df[df['target'] == 1]['num_characters'], color='red')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe65436d790>
```



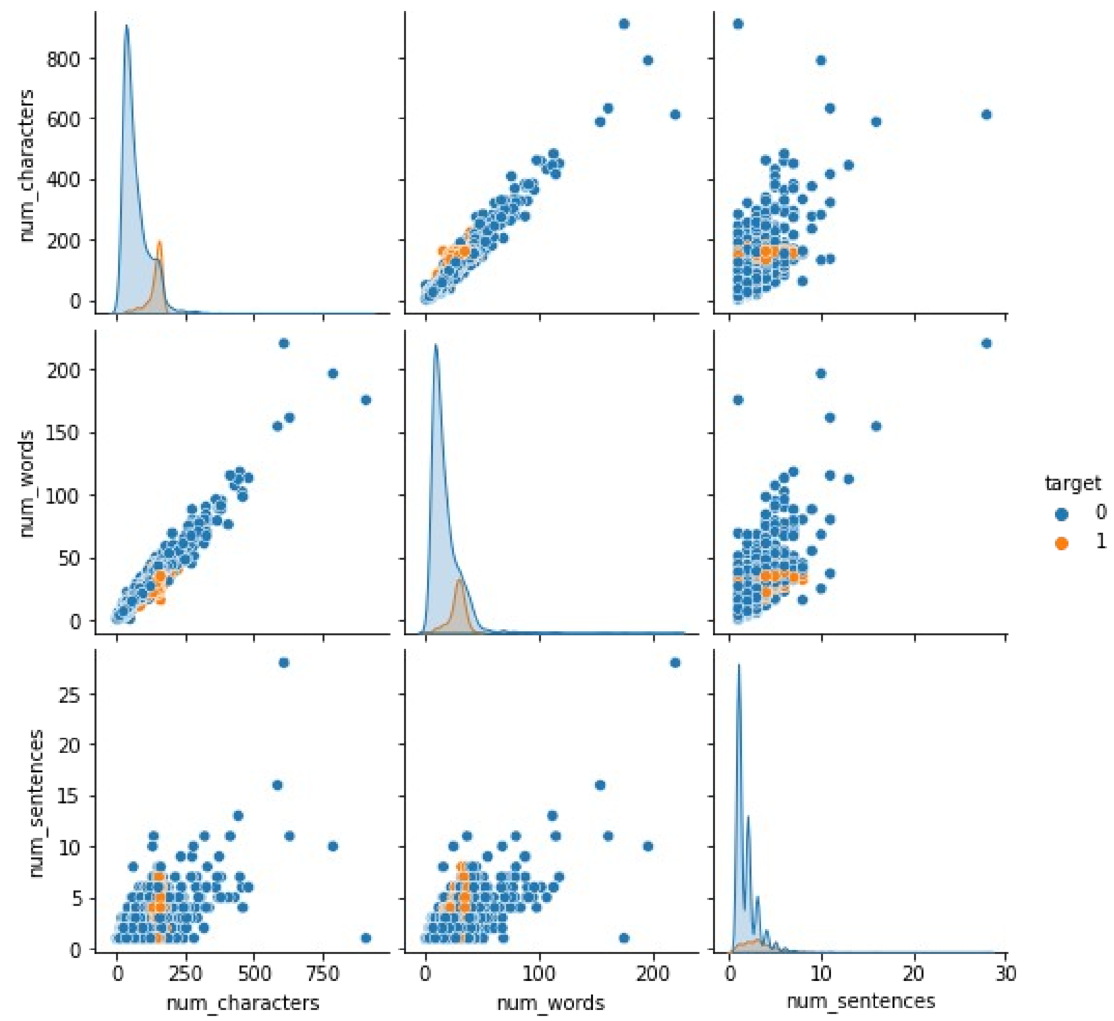
```
plt.figure(figsize=(12,6))
sns.histplot(df[df['target'] == 0]['num_words'], color='green')
sns.histplot(df[df['target'] == 1]['num_words'],color='red')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe6540c56d0>
```



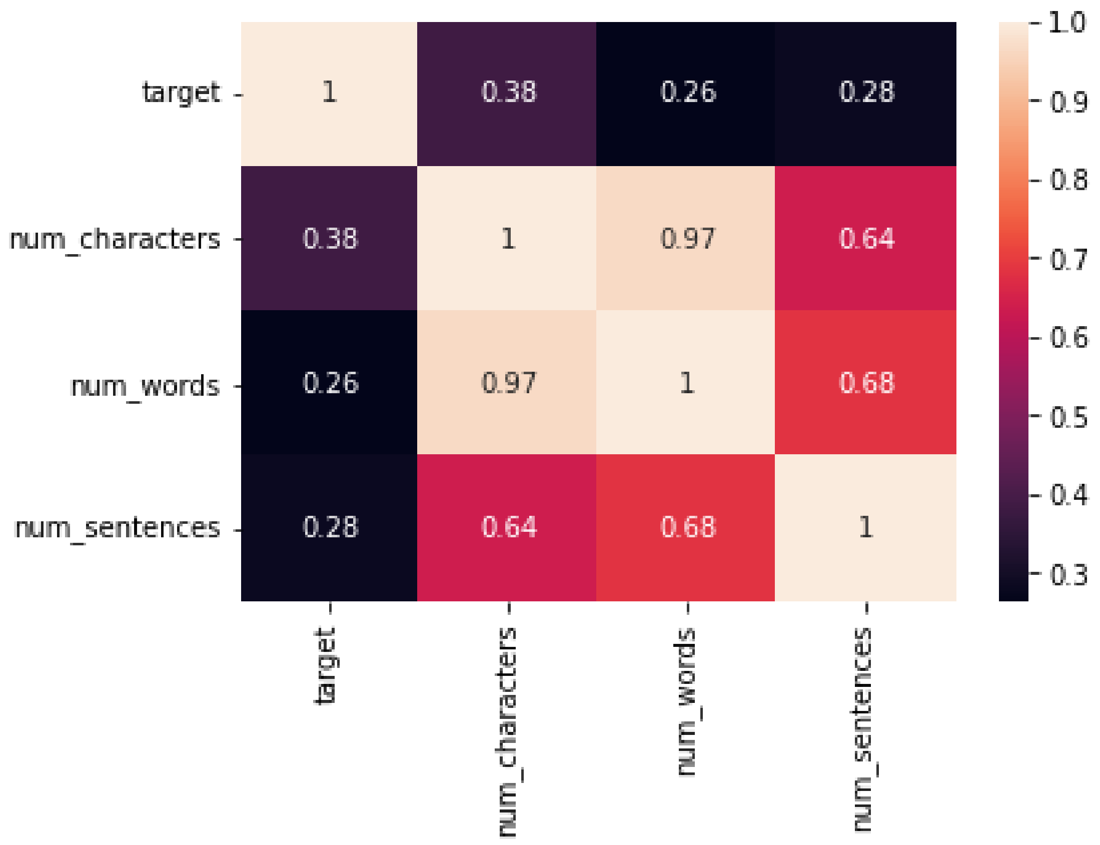
```
sns.pairplot(df, hue='target')
```

```
<seaborn.axisgrid.PairGrid at 0x7fe653e9a250>
```



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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe653fd6410>
```



```
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()
print(ps.stem('played'))
print(ps.stem('playing'))
```

```
play
play
```

```
import string
string.punctuation
```

```
{"type":"string"}
```

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

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

```
['i',
 'me',
 'my',
 'myself',
 'we',
 'our',
 'ours',
 'ourselves',
 'you',
 "you're",
 "you've",
 "you'll",
 "you'd",
 'your',
 'yours',
 'yourself',
 'yourselves',
 'he',
 'him',
 'his',
 'himself',
 'she',
 "she's",
 'her',
 'hers',
 'herself',
 'it',
 "it's",
 'its',
 'itself',
 'they',
 'them',
 'their',
 'theirs',
 'themselves',
 'what',
 'which',
 'who',
 'whom',
 'this',
 'that',
 "that'll",
 'these',
 'those',
 'am',
 'is',
 'are',
 'was',
 'were',
 'be',
 'been',
 'being',
 'have',
 'has',
 'had',
 'having',
     'do',
     'does',
     'did',
     'doing',
     'a',
     'an',
     'the',
     'and',
     'but',
     'if',
     'or',
     'because',
     'as',
     'until',
     'while',
     'of',
     'at',
     'by',
     'for',
     'with',
     'about',
     'against',
     'between',
     'into',
     'through',
     'during',
     'before',
     'after',
     'above',
     'below',
     'to',
     'from',
     'up',
     'down',
     'in',
     'out',
     'on',
     'off',
     'over',
     'under',
     'again',
     'further',
     'then',
     'once',
     'here',
     'there',
     'when',
     'where',
     'why',
     'how',
  'all',
  'any',
  'both',
  'each',
  'few',
  'more',
  'most',
  'other',
  'some',
  'such',
  'no',
  'nor',
  'not',
  'only',
  'own',
  'same',
  'so',
  'than',
  'too',
  'very',
  's',
  't',
  'can',
  'will',
  'just',
  'don',
  "don't",
  'should',
  "should've",
  'now',
  'd',
  'll',
  'm',
  'o',
  're',
  've',
  'y',
  'ain',
  'aren',
  "aren't",
  'couldn',
  "couldn't",
  'didn',
  "didn't",
  'doesn',
  "doesn't",
  'hadn',
  "hadn't",
  'hasn',
  "hasn't",
  'haven',
  "haven't",
  'isn',
  "isn't",
  'ma',
  'mightn',
  "mightn't",
  'mustn',
  "mustn't",
  'needn',
  "needn't",
  'shan',
  "shan't",
  'shouldn',
  "shouldn't",
  'wasn',
  "wasn't",
  'weren',
  "weren't",
  'won',
  "won't",
  'wouldn',
  "wouldn't"]
```

```
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)
```

```
df['text'][23]
```

{"type":"string"}

```
transform_text(df['text'][23])
```

{"type":"string"}

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

   num_words  num_sentences                                  
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='whit
e')
```

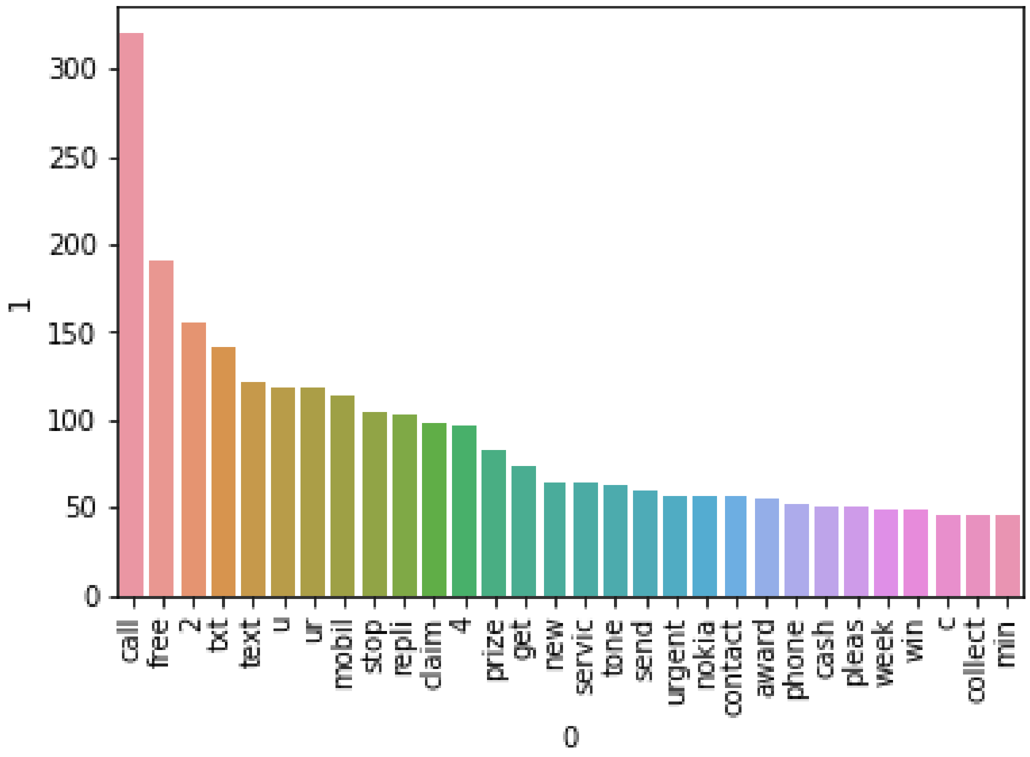
```
spam_corpus = []
for msg in df[df['target'] == 1]['transformed_text'].tolist():
    for word in msg.split():
        spam_corpus.append(word)
len(spam_corpus)
```

```
9939
```

```
from collections import Counter
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()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
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.
  FutureWarning
```



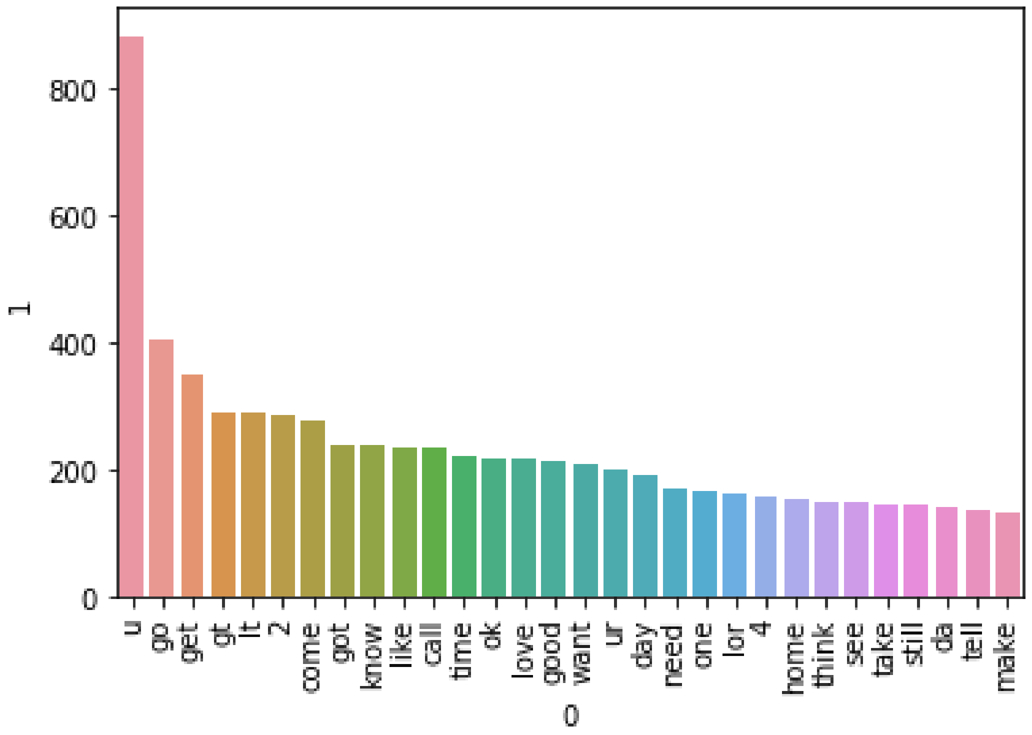
```
ham_corpus = []
for msg in df[df['target'] == 0]['transformed_text'].tolist():
    for word in msg.split():
        ham_corpus.append(word)
len(ham_corpus)
```

```
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()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
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.
  FutureWarning
```



```
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer(max_features=3000) # 3000 is giving good
accuracy and precision

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))
```

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

```
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))
```

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

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

```
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)
```

```python
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

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 -  0.9284332688588007
Precision -  0.82
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 -  0.9439071566731141
Precision -  0.9347826086956522
```

```
df =
pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy':accuracy_scores,'Prec
ision':precision_scores}).sort_values('Precision',ascending=False)
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  0.928433   0.820000
```