```
import numpy as np
import pandas as pd
```

```
df= pd.read_csv('/content/drive/MyDrive/emailspam/spam.csv', encoding='latin-1')
```

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

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
726	ham	Of cos can lar i'm not so ba dao ok... 1 pm lo...	NaN	NaN	NaN
4007	ham	Forgot you were working today! Wanna chat, but...	NaN	NaN	NaN
1647	ham	Evening * v good if somewhat event laden. Will...	NaN	NaN	NaN
921	ham	On ma way to school. Can you pls send me ashle...	NaN	NaN	NaN
5119	ham	Lol for real. She told my dad I have cancer	NaN	NaN	NaN

```
df.shape
```

```
(5572, 5)
```

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
```

```
#drop last 3 cols
df.drop(columns=['Unnamed: 2','Unnamed: 3','Unnamed: 4'],inplace=True)
```

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

	v1	v2
3768	ham	Was gr8 to see that message. So when r u leavi...
4308	ham	He dint tell anything. He is angry on me that ...
5405	ham	So how many days since then?
2795	ham	Tell your friends what you plan to do on Valen...
3863	ham	A pure hearted person can have a wonderful smi...

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

	target	text
3222	ham	Well that must be a pain to catch
3320	ham	Yo im right by yo work
3480	ham	Wherre's my boytoy ? :-(
3321	ham	Ok darlin i supose it was ok i just worry too ...
2738	ham	I sent you the prices and do you mean the <...

```
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...

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

	0
target	0
text	0

dtype: int64

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

```
np.int64(403)
```

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

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

```
np.int64(0)
```

```
df.shape
```

```
(5169, 2)
```

EDA

```
df.head()
```

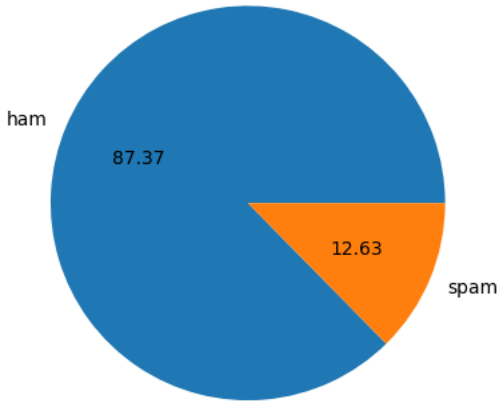
	target	text
0	0	Go until jurong point, crazy.. Available only ...
1	0	Ok lar... Joking wif u oni...
2	1	Free entry in 2 a wkly comp to win FA Cup fina...
3	0	U dun say so early hor... U c already then say...
4	0	Nah I don't think he goes to usf, he lives aro...

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

target	count
0	4516
1	653

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 /usr/local/lib/python3.12/dist-packages (3.9.1)
Requirement already satisfied: click in /usr/local/lib/python3.12/dist-packages (from nltk) (8.2.1)
Requirement already satisfied: joblib in /usr/local/lib/python3.12/dist-packages (from nltk) (1.5.1)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.12/dist-packages (from nltk) (2024.11.6)
Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dist-packages (from nltk) (4.67.1)
```

```
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

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

```
nltk.download('punkt_tab')
```

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

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

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
True
```

```
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

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

```
df.head()
```

	target	text	num_characters	num_words	num_sentences
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

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

	num_characters	num_words	num_sentences
count	5169.000000	5169.000000	5169.000000
mean	78.977945	18.455794	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

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	17.123782	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

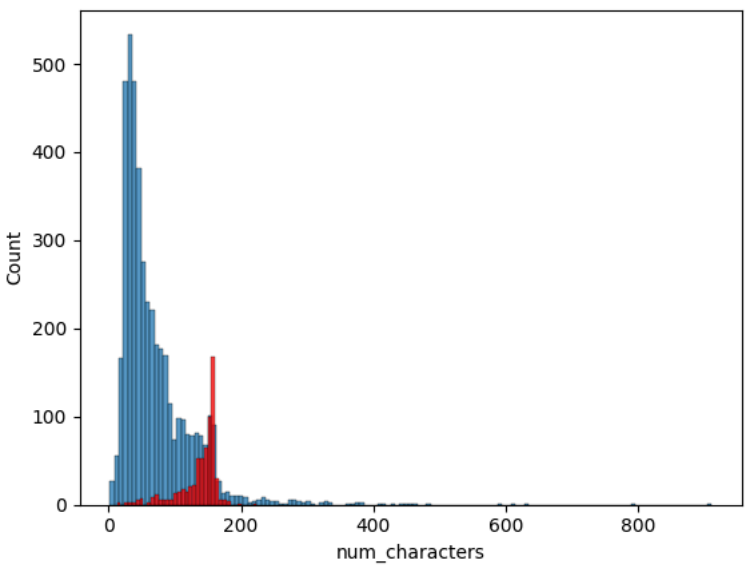
```
#spam
df[df['target']==1][['num_characters','num_words','num_sentences']].describe()
```

	num_characters	num_words	num_sentences
count	653.000000	653.000000	653.000000
mean	137.891271	27.667688	2.970904
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

```
import seaborn as sns
```

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

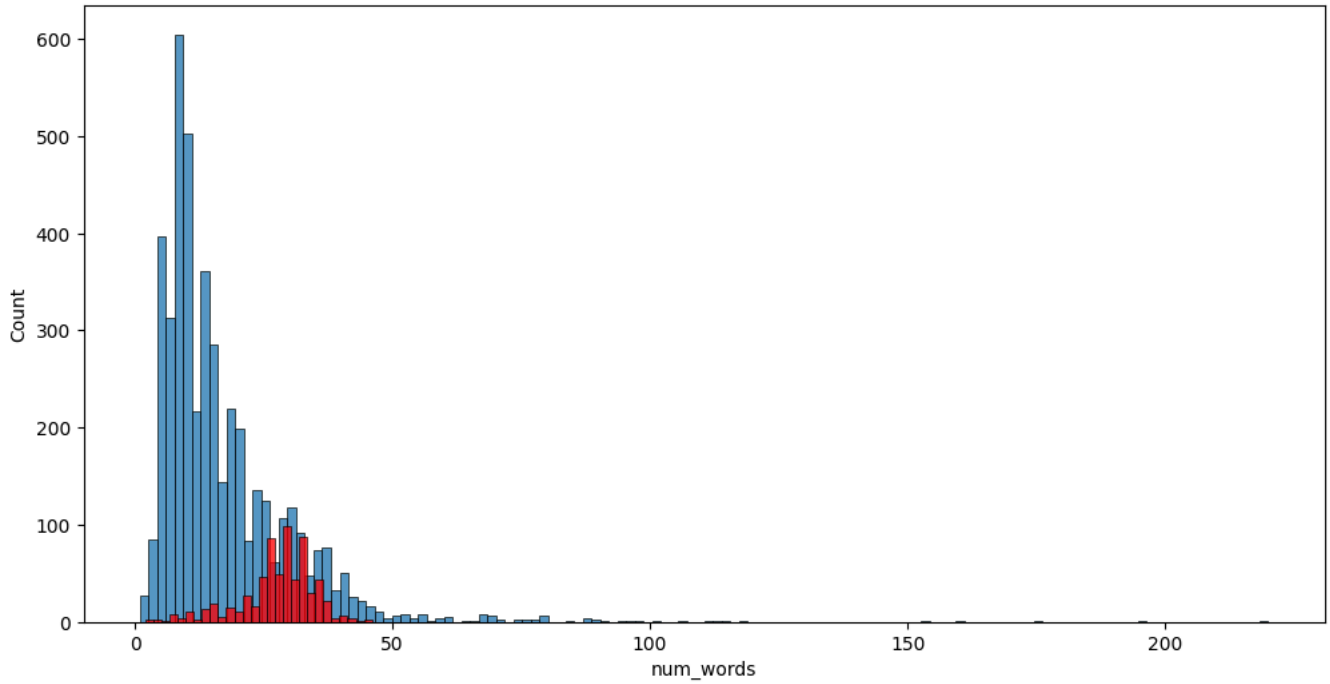
```
<Figure size 1200x600 with 0 Axes>
```




```
<Figure size 1200x600 with 0 Axes>
```

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

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

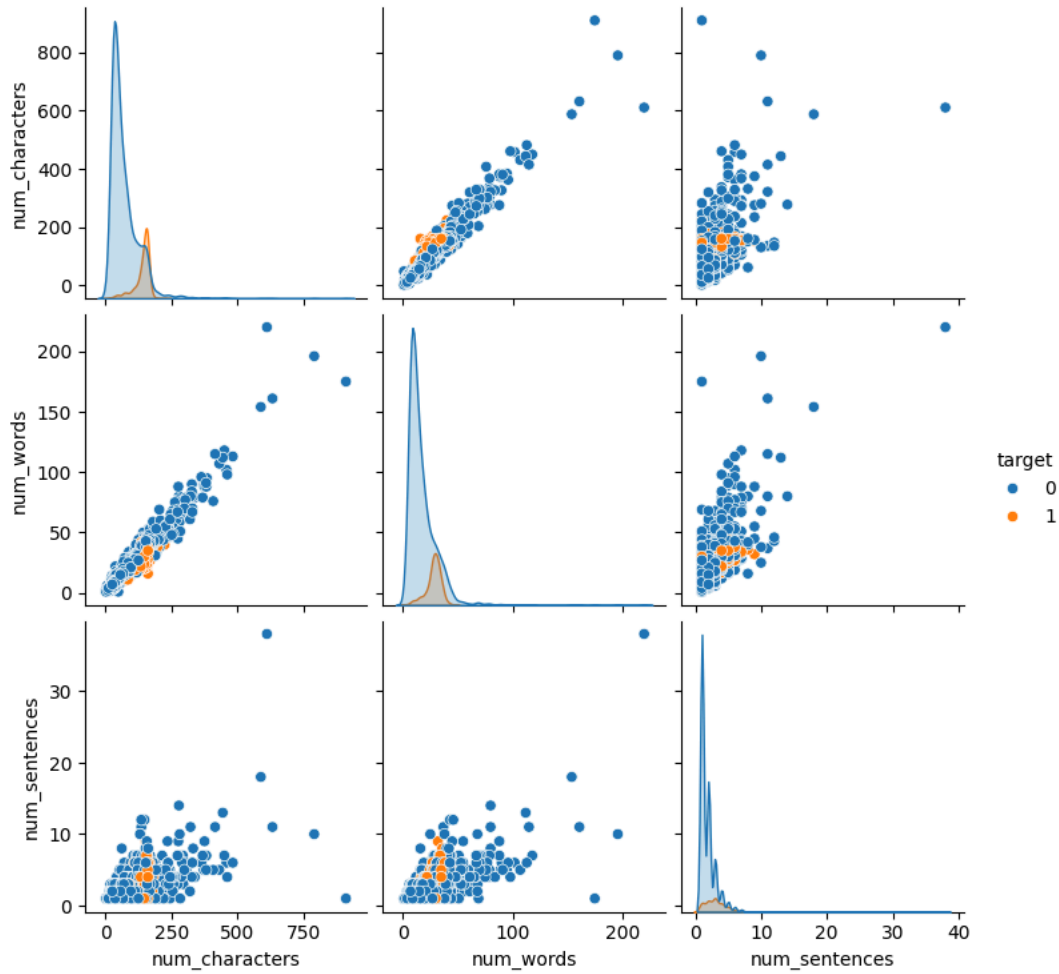


relation between no.of column,no of sentences

Double-click (or enter) to edit

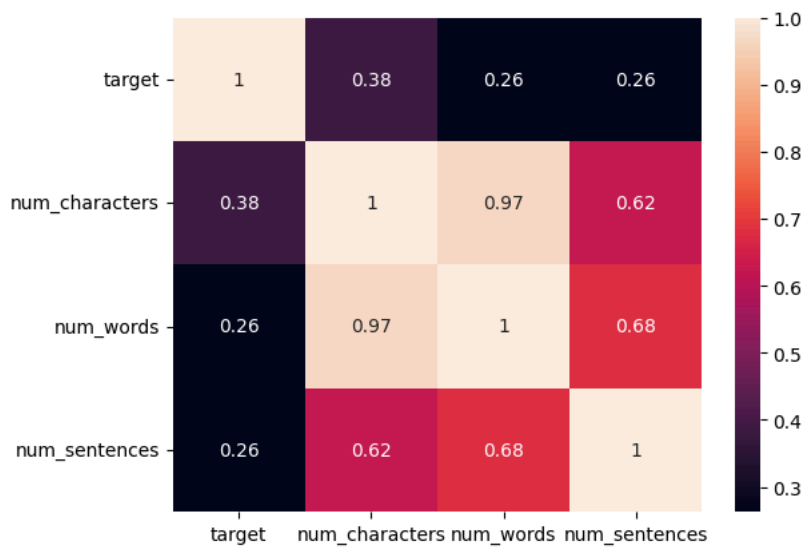
```
sns.pairplot(df,hue='target')
```

```
<seaborn.axisgrid.PairGrid at 0x7c8dc12dce00>
```




```
sns.heatmap(df[['target','num_characters','num_words','num_sentences']].corr(),annot=True)
```

```
<Axes: >
```



Data/Text preprocessing

- Lower case
- Tokenization
- Removing special character
- Removing stop words and punctuation
- Stemming

```
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)
```

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

```
 yourselves ',
 "you've"]
```

```
nltk.download('stopwords')
```

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

```
import string
string.punctuation
```

```
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
```

```
transform_text('I loved the YT lectures on Machine Learning. How about you?')
```

```
'love yt lectur machin learn'
```

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

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

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

```
'danc'
```

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

```
df.head()
```

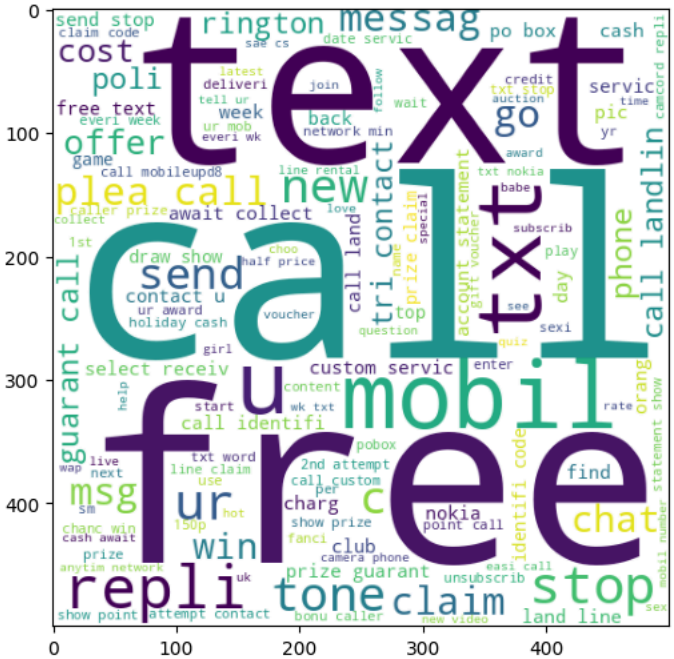
	target	text	num_characters	num_words	num_sentences	transformed_text
0	0	Go until jurong point, crazy.. Available only ...	111	24	2	go jurong point crazi avail bugi n great world...
1	0	Ok lar... Joking wif u oni...	29	8	2	ok lar joke wif u oni
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	155	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

```
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)
```

```
<matplotlib.image.AxesImage at 0x7c8dc27178f0>
```




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

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

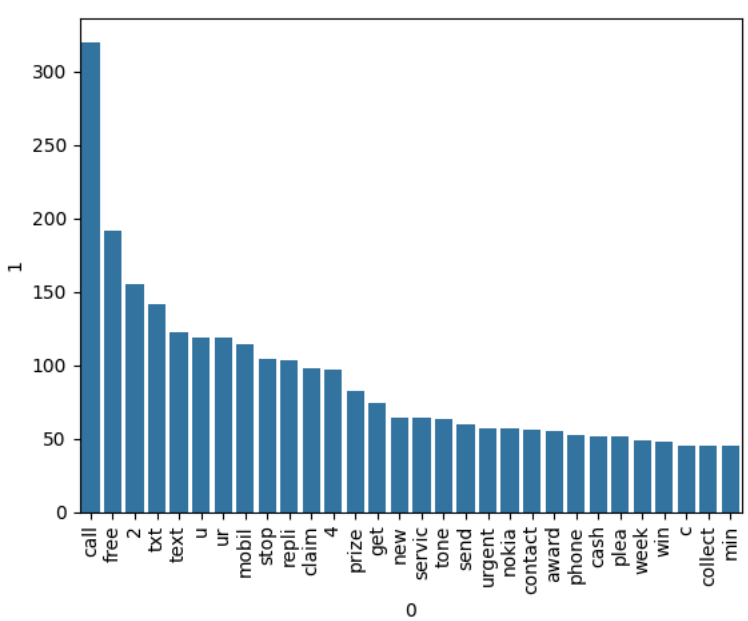
```
<matplotlib.image.AxesImage at 0x7c8dc2717f20>
```


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

```
len(spam_corpus)
```

```
9930
```

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

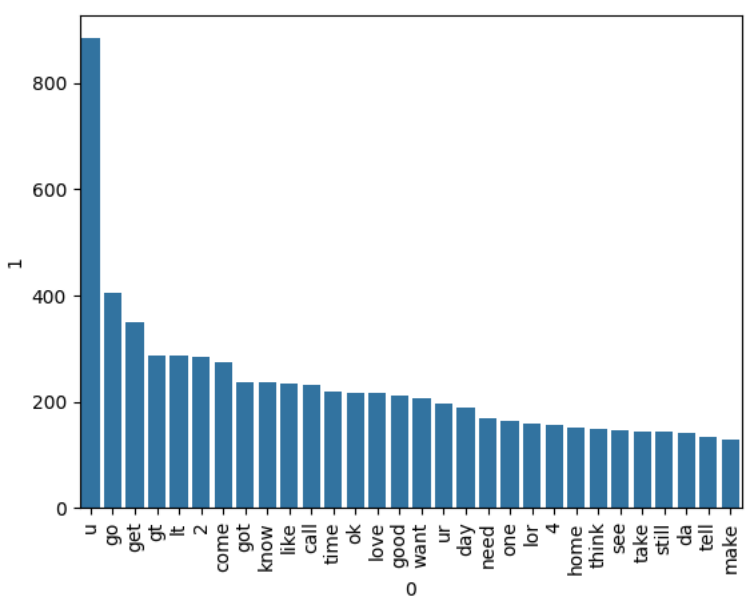



```
ham_corpus=[]
for msg in df[df['target']==0]['transformed_text'].tolist():
  for words in msg.split():
    ham_corpus.append(words)
```

```
len(ham_corpus)
```

```
35296
```

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



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()
```

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
x = scaler.fit_transform(X)
```

```
#appending the num_character col to x
#x=np.hstack(x,df['num_characters'].values.reshape(-1,1))
```

```
X.shape
```

```
(5169, 3000)
```

```
y = df['target'].values
```

```
from sklearn.model_selection import train_test_split
```

```
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
[[791 105]
 [ 25 113]]
0.518348623853211
```

```
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))
```

```
0.971953578336557
[[896   0]
 [ 29 109]]
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
```

```
# tfidf,mnb we use
```

```
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_features=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)
```

```
(0.97678916827853, 0.975)
```

```
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 - 0.97678916827853
Precision - 0.975
For KN
```

```
Accuracy - 0.9052224371373307
Precision - 1.0
For NB
Accuracy - 0.971953578336557
Precision - 1.0
For DT
Accuracy - 0.9439071566731141
Precision - 0.8278688524590164
For LR
Accuracy - 0.9555125725338491
Precision - 0.96
For RF
Accuracy - 0.9758220502901354
Precision - 0.9829059829059829
For AdaBoost
Accuracy - 0.9216634429400387
Precision - 0.8202247191011236
For BgC
Accuracy - 0.9593810444874274
Precision - 0.8692307692307693
For ETC
Accuracy - 0.97678916827853
Precision - 0.975
For GBDT
Accuracy - 0.9516441005802708
Precision - 0.9230769230769231
For xgb
Accuracy - 0.9700193423597679
Precision - 0.9495798319327731
```

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

```
performance_df
```

	Algorithm	Accuracy	Precision
1	KN	0.905222	1.000000
2	NB	0.971954	1.000000
5	RF	0.975822	0.982906
0	SVC	0.976789	0.975000
8	ETC	0.976789	0.975000
4	LR	0.955513	0.960000
10	xgb	0.970019	0.949580
9	GBDT	0.951644	0.923077
7	BgC	0.959381	0.869231
3	DT	0.943907	0.827869
6	AdaBoost	0.921663	0.820225

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

```
performance_df1
```

	Algorithm	variable	value
0	KN	Accuracy	0.905222
1	NB	Accuracy	0.971954
2	RF	Accuracy	0.975822
3	SVC	Accuracy	0.976789
4	ETC	Accuracy	0.976789
5	LR	Accuracy	0.955513
6	xgb	Accuracy	0.970019
7	GBDT	Accuracy	0.951644
8	BgC	Accuracy	0.959381
9	DT	Accuracy	0.943907
10	AdaBoost	Accuracy	0.921663

```
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()
```