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

```
df = pd.read_csv('/content/drive/MyDrive/Bharat Intern Internship/SMS spam detection/encoded-sp
```

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

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
5318	ham	Good morning, my Love ... I go to sleep now an...	NaN	NaN	NaN
711	ham	It just seems like weird timing that the night...	NaN	NaN	NaN
		Ok I din get ur			

```
df.shape
```

```
(5572, 5)
```

```
# 1. Data cleaning
# 2. EDA
# 3. Text Preprocessing
# 4. Model building
# 5. Evaluation
# 6. Improvement
# 7. Website
# 8. Deploy
```

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
```

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

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

	v1	v2
1464	ham	Wat makes u thk i'll fall down. But actually i...
288	ham	My life Means a lot to me, Not because I love ...
4840	ham	Hmmm.... Mayb can try e shoppin area one, but ...
3973	ham	Roger that. We‰Û÷re probably going to rem in a...
3375	ham	Good afternon, my love. How are today? I hope ...

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

	target	text
1236	ham	How much are we getting?
4493	ham	Man this bus is so so so slow. I think you're ...
2087	ham	\alright babe
2501	ham	No da..today also i forgot..
575	spam	You have won ?1,000 cash or a ?2,000 prize! To...

```
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()
```

```
target    0
text      0
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, 2)
```

2.EDA

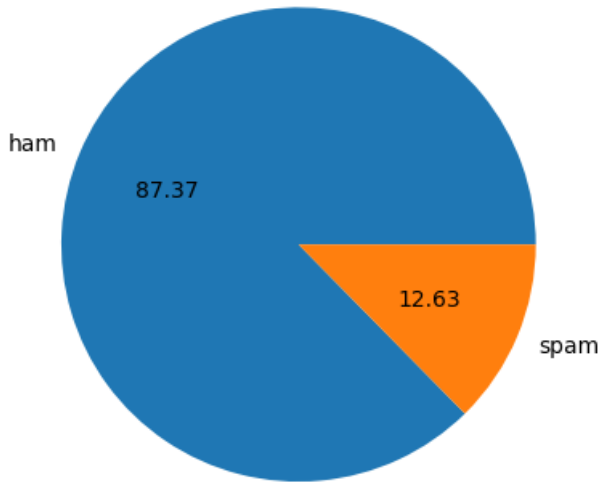
```
df.head()
```

	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()
```

```
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
```

```
!pip install nltk
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (3.8.1)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk) (8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from nltk) (1.3.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-packages (from nltk) (2023
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from nltk) (4.66.1)
```

```
1 import nltk
```

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

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

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

```
1  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

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

```
1 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

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

```
1  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

```
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

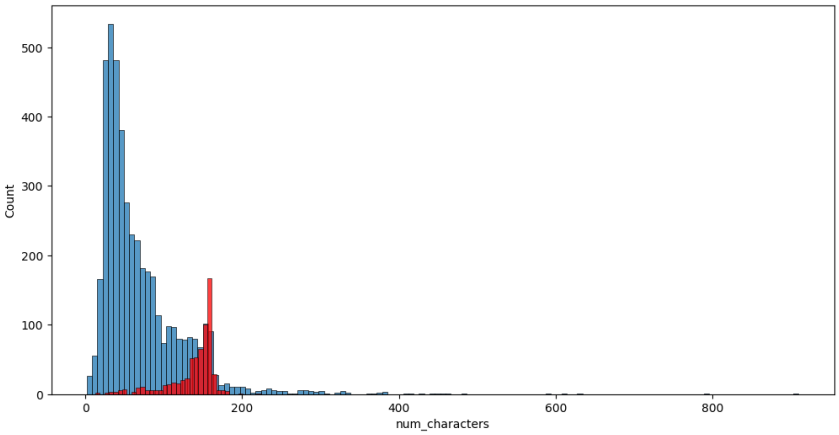
```
#spam
df[df['target'] == 1][['num_characters','num_words','num_sentences']].describe()
```

	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
```

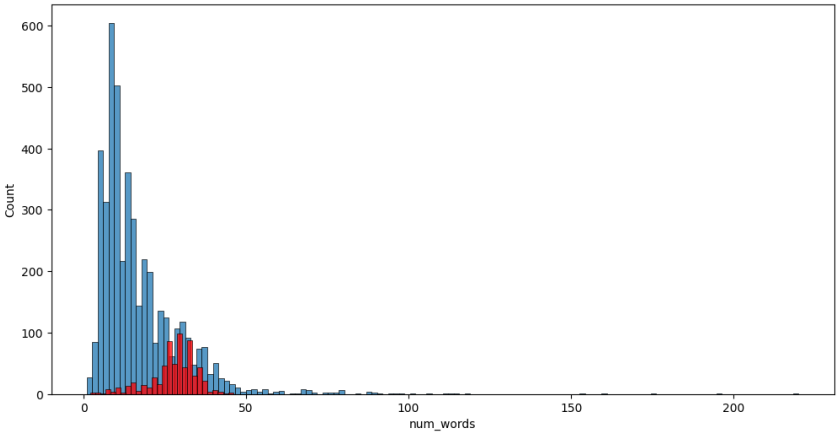
```
plt.figure(figsize=(12,6))
sns.histplot(df[df['target'] == 0]['num_characters'])
sns.histplot(df[df['target'] == 1]['num_characters'],color='red')
```

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



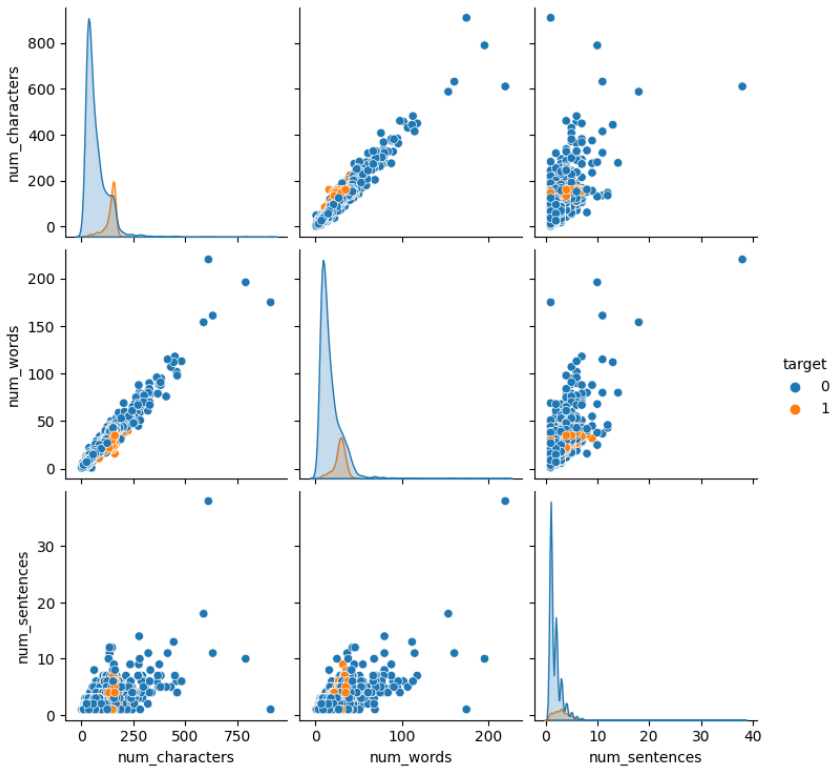

```
1  plt.figure(figsize=(12,6))
2  sns.histplot(df[df['target'] == 0]['num_words'])
3  sns.histplot(df[df['target'] == 1]['num_words'],color='red')
```

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



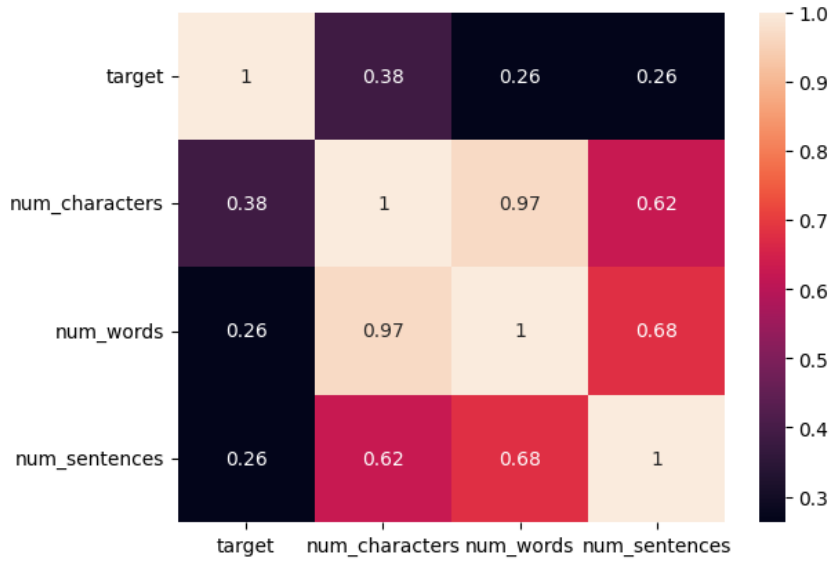

```
1 sns.pairplot(df,hue='target')
```

```
<seaborn.axisgrid.PairGrid at 0x780ed062bbb0>
```




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

```
<ipython-input-81-8df7bcac526d>:1: FutureWarning: The default value c
  sns.heatmap(df.corr(),annot=True)
<Axes: >
```



3.Data Preprocessing

---

1. Lower case
2. Tokenization
3. Removing special characters
4. Removing stop words and punctuation
5. Stemming

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

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
'love'
```

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

```
df.head()
```

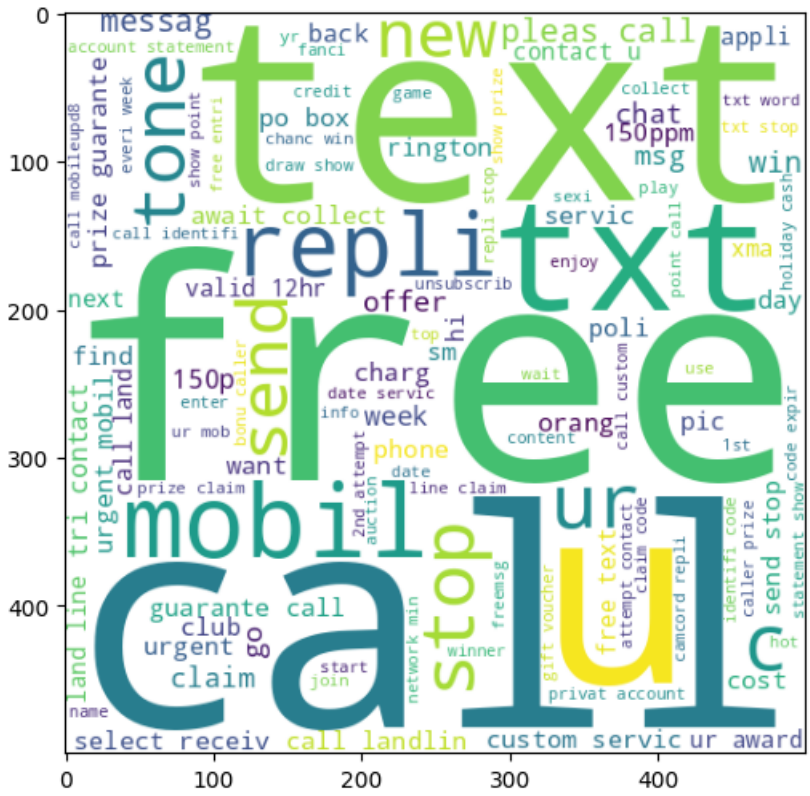
	target	text	num_characters	num_words	num_sentences	transfo
0	0	Go until jurong point, crazy.. Available only ... Ok lar...	111	24	2	go crazi g

```
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)
```

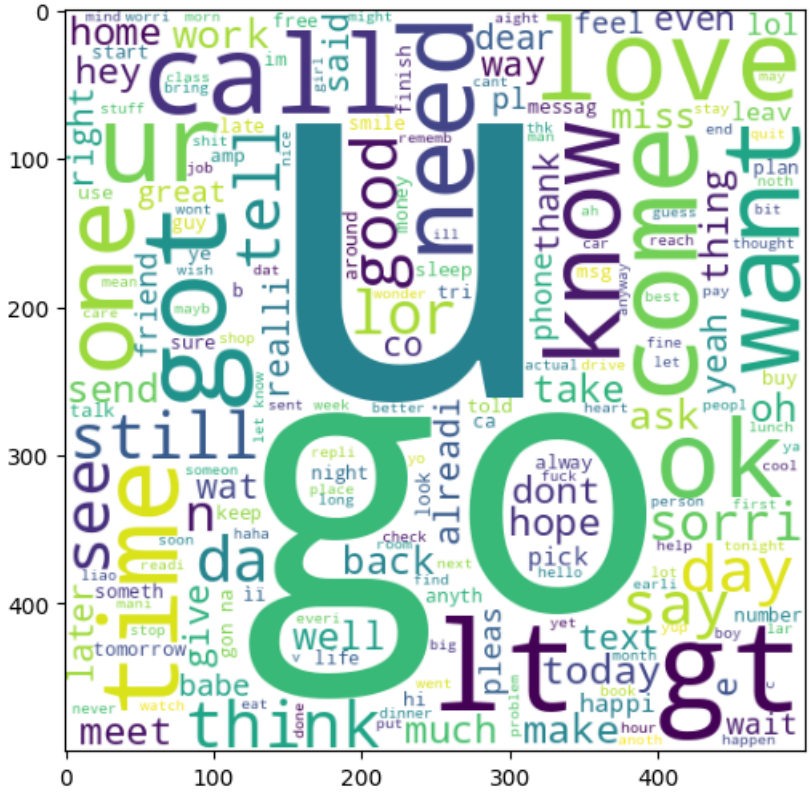
```
<matplotlib.image.AxesImage at 0x780ecfd6b6d0>
```




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

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

```
<matplotlib.image.AxesImage at 0x780ecf1ee350>
```




```
1  df.head()
```

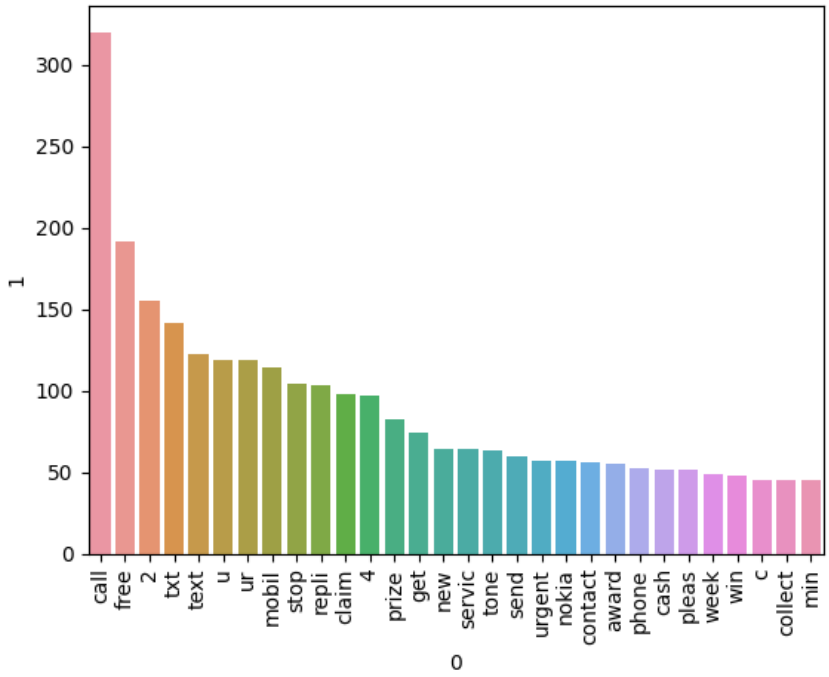
	target	text	num_characters	num_words	num_sentences	transfo
0	0	Go until jurong point, crazy.. Available only ... Ok lar...	111	24	2	go crazi g

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

```
1 len(spam_corpus)
```

```
9939
```

```
1 from collections import Counter
2 import matplotlib.pyplot as plt
3 sns.barplot(x=pd.DataFrame(Counter(spam_corpus).most_common(30))[0],y=pd.DataFrame(Counter(spam
4 plt.xticks(rotation='vertical')
5 plt.show()
```

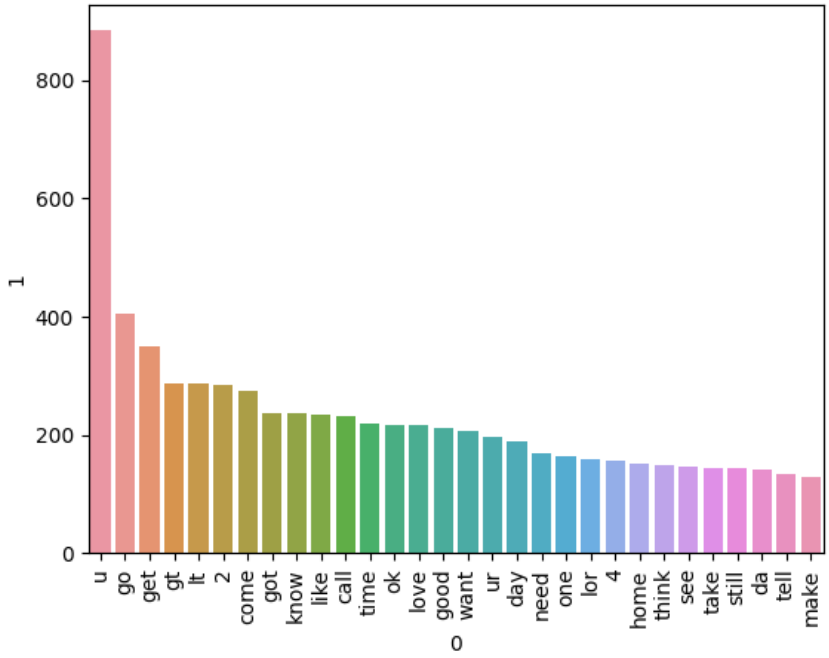



```
1 ham_corpus = []
2 for msg in df[df['target'] == 0]['transformed_text'].tolist():
3     for word in msg.split():
4         ham_corpus.append(word)
```

```
1 len(ham_corpus)
```

```
35404
```

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




```
# Text Vectorization
# using Bag of Words
df.head()
```

	target	text	num_characters	num_words	num_sentences	transfo
0	0	Go until jurong point, crazy.. Available only ... Ok lar...	111	24	2	go crazi g

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()
```

```
#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)))
```

```
1 X.shape
```

```
(5169, 3000)
```

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

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

```
1 X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=2)
```

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

```
1 gnb = GaussianNB()
2 mnb = MultinomialNB()
3 bnb = BernoulliNB()
```

```
1 gnb.fit(X_train,y_train)
2 y_pred1 = gnb.predict(X_test)
3 print(accuracy_score(y_test,y_pred1))
4 print(confusion_matrix(y_test,y_pred1))
5 print(precision_score(y_test,y_pred1))
```

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

```
1 mnb.fit(X_train,y_train)
2 y_pred2 = mnb.predict(X_test)
3 print(accuracy_score(y_test,y_pred2))
4 print(confusion_matrix(y_test,y_pred2))
5 print(precision_score(y_test,y_pred2))
```

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

```
1 bnb.fit(X_train,y_train)
2 y_pred3 = bnb.predict(X_test)
3 print(accuracy_score(y_test,y_pred3))
4 print(confusion_matrix(y_test,y_pred3))
5 print(precision_score(y_test,y_pred3))
```

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

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

```
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)
```

```
(0.9758220502901354, 0.9747899159663865)
```

```
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 -  0.9303675048355899
Precision -  0.8173076923076923
For  LR
Accuracy -  0.9584139264990329
Precision -  0.9702970297029703
For  RF
Accuracy -  0.9758220502901354
Precision -  0.9829059829059829
For  AdaBoost
Accuracy -  0.960348162475822
Precision -  0.9292035398230089
For  BgC
Accuracy -  0.9584139264990329
Precision -  0.8682170542635659
For  ETC
Accuracy -  0.9748549323017408
Precision -  0.9745762711864406
For  GBDT
Accuracy -  0.9468085106382979
Precision -  0.9191919191919192
For  xgb
Accuracy -  0.9671179883945842
Precision -  0.9262295081967213
```

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

	Algorithm	Accuracy	Precision
1	KN	0.905222	1.000000
2	NB	0.970986	1.000000
5	RF	0.975822	0.982906
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	0.967118	0.926230
9	GBDT	0.946809	0.919192
7	BgC	0.958414	0.868217
3	DT	0.930368	0.817308

```
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	0.975822
3	SVC	Accuracy	0.975822
4	ETC	Accuracy	0.974855
5	LR	Accuracy	0.958414
6	AdaBoost	Accuracy	0.960348
7	xgb	Accuracy	0.967118
8	GBDT	Accuracy	0.946809
9	BgC	Accuracy	0.958414
10	DT	Accuracy	0.930368
11	KN	Precision	1.000000
12	NB	Precision	1.000000
13	RF	Precision	0.982906
14	SVC	Precision	0.974790
15	ETC	Precision	0.974576
16	LR	Precision	0.970297
17	AdaBoost	Precision	0.929204
18	xgb	Precision	0.926230
19	GBDT	Precision	0.919192
20	BgC	Precision	0.868217
21	DT	Precision	0.817308

```
sns.catplot(x = 'Algorithm', y='value',
                hue = 'variable',data=performance_df1, kind='bar',height=5)
```

```
3   plt.ylim(0.5,1.0)
4   plt.xticks(rotation='vertical')
5   plt.show()
```