In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
df = pd.read_csv('spam.csv')
```

In [4]:
```
df.sample(5)
```

Out[4]:

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
2464	ham	They will pick up and drop in car.so no problem..	NaN	NaN	NaN
1248	ham	HI HUN! IM NOT COMIN 2NITE-TELL EVERY1 IM SORR...	NaN	NaN	NaN
1413	spam	Dear U've been invited to XCHAT. This is our f...	NaN	NaN	NaN
2995	ham	They released vday shirts and when u put it on...	NaN	NaN	NaN
4458	spam	Welcome to UK-mobile-date this msg is FREE giv...	NaN	NaN	NaN

In [5]:
```
df.shape
```

Out[5]:
```
(5572, 5)
```

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

1. Data Cleaning

In [6]:
```
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
```

In [7]:
```
# drop last 3 cols
df.drop(columns=['Unnamed: 2','Unnamed: 3','Unnamed: 4'],inplace=True)
```

In [8]:
```
df.sample(5)
```

Out[8]:

	v1	v2
1947	ham	The battery is for mr adewale my uncle. Aka Egbon
2712	ham	Hey you still want to go for yogasana? Coz if ...
4428	ham	Hey they r not watching movie tonight so i'll ...
3944	ham	I will be gentle princess! We will make sweet ...
49	ham	U don't know how stubborn I am. I didn't even ...

In [9]:
```
# renaming the cols
df.rename(columns={'v1':'target','v2':'text'},inplace=True)
df.sample(5)
```

Out[9]:

	target	text
1418	ham	Lmao. Take a pic and send it to me.
2338	ham	Alright, see you in a bit
88	ham	I'm really not up to it still tonight babe
3735	ham	Hows the street where the end of library walk is?
3859	ham	Yep. I do like the pink furniture tho.

In [10]:
```
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
```

In [12]:
```
df['target'] = encoder.fit_transform(df['target'])
```

In [13]:
```
df.head()
```

Out[13]:

	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...

In [14]:
```
# missing values
df.isnull().sum()
```

Out[14]:
```
target    0
text      0
dtype: int64
```

In [15]:
```
# check for duplicate values
df.duplicated().sum()
```

Out[15]:
```
403
```

In [17]:
```
# remove duplicates
df = df.drop_duplicates(keep='first')
```

In [18]:
```
df.duplicated().sum()
```

Out[18]:
```
0
```

In [19]:
```
df.shape
```

Out[19]:
```
(5169, 2)
```

2.EDA

In [29]:
```
df.head()
```

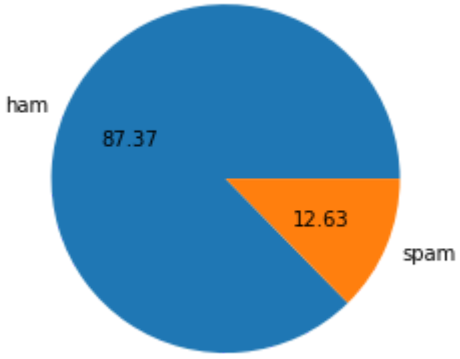
Out[29]:

	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...

In [31]:
```
df['target'].value_counts()
```

Out[31]:
```
0    4516
1     653
Name: target, dtype: int64
```

In [33]:
```
import matplotlib.pyplot as plt
plt.pie(df['target'].value_counts(), labels=['ham','spam'],autopct="%0.2f")
plt.show()
```



In [34]:
```
# Data is imbalanced
```

In [35]:
```
import nltk
```

In [ ]:
```
!pip install nltk
```

In [37]:
```
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\91842\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping tokenizers\punkt.zip.
```

Out[37]:
```
True
```

In [45]:
```
df['num_characters'] = df['text'].apply(len)
```

In [46]:
```
df.head()
```

Out[46]:

	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

In [50]:
```
# num of words
df['num_words'] = df['text'].apply(lambda x:len(nltk.word_tokenize(x)))
```

In [51]:
```
df.head()
```

Out[51]:

	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

In [53]:
```
df['num_sentences'] = df['text'].apply(lambda x:len(nltk.sent_tokenize(x)))
```

In [54]:
```
df.head()
```

Out[54]:

	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

In [55]:
```
df[['num_characters','num_words','num_sentences']].describe()
```

Out[55]:

	num_characters	num_words	num_sentences
count	5169.000000	5169.000000	5169.000000
mean	78.923776	18.456375	1.962275
std	58.174846	13.323322	1.433892
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

In [58]:
```
# ham
df[df['target'] == 0][['num_characters','num_words','num_sentences']].descr
```

Out[58]:

	num_characters	num_words	num_sentences
count	4516.000000	4516.000000	4516.000000
mean	70.456820	17.123339	1.815545
std	56.356802	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

In [59]:
```
#spam
df[df['target'] == 1][['num_characters','num_words','num_sentences']].descr
```

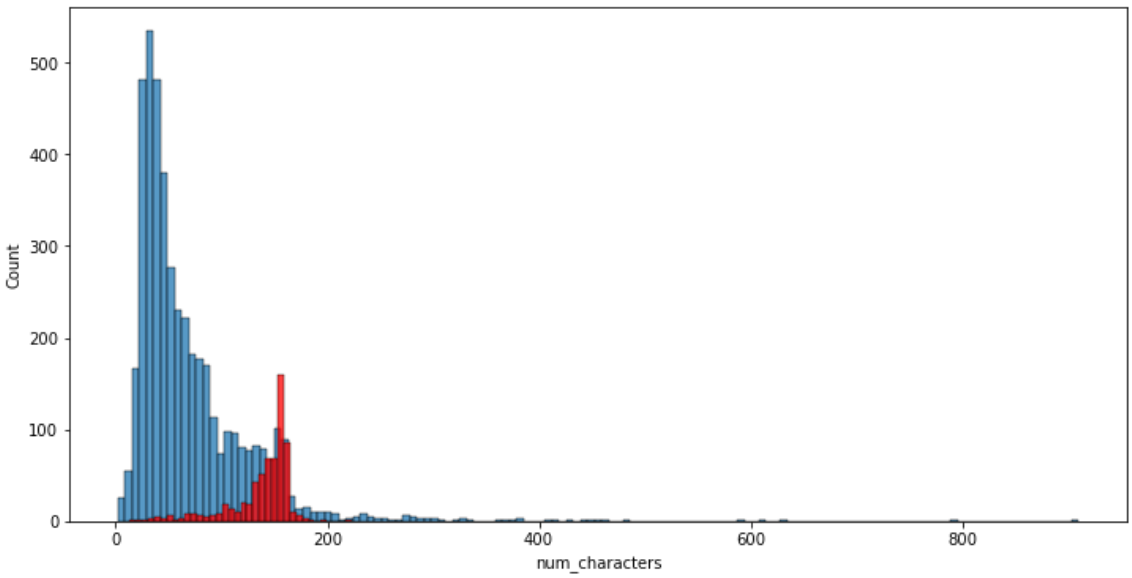
Out[59]:

	num_characters	num_words	num_sentences
count	653.000000	653.000000	653.000000
mean	137.479326	27.675345	2.977029
std	30.014336	7.011513	1.493676
min	13.000000	2.000000	1.000000
25%	131.000000	25.000000	2.000000
50%	148.000000	29.000000	3.000000
75%	157.000000	32.000000	4.000000
max	223.000000	46.000000	9.000000

In [78]:
```
import seaborn as sns
```

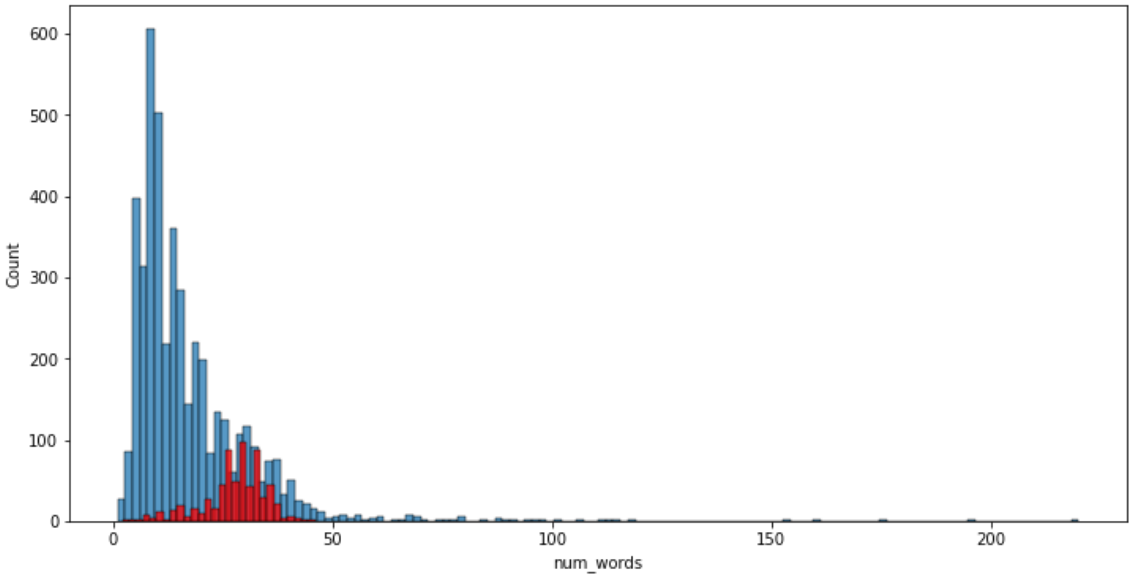
In [84]:
```
plt.figure(figsize=(12,6))
sns.histplot(df[df['target'] == 0]['num_characters'])
sns.histplot(df[df['target'] == 1]['num_characters'],color='red')
```

Out[84]:
```
<AxesSubplot:xlabel='num_characters', ylabel='Count'>
```



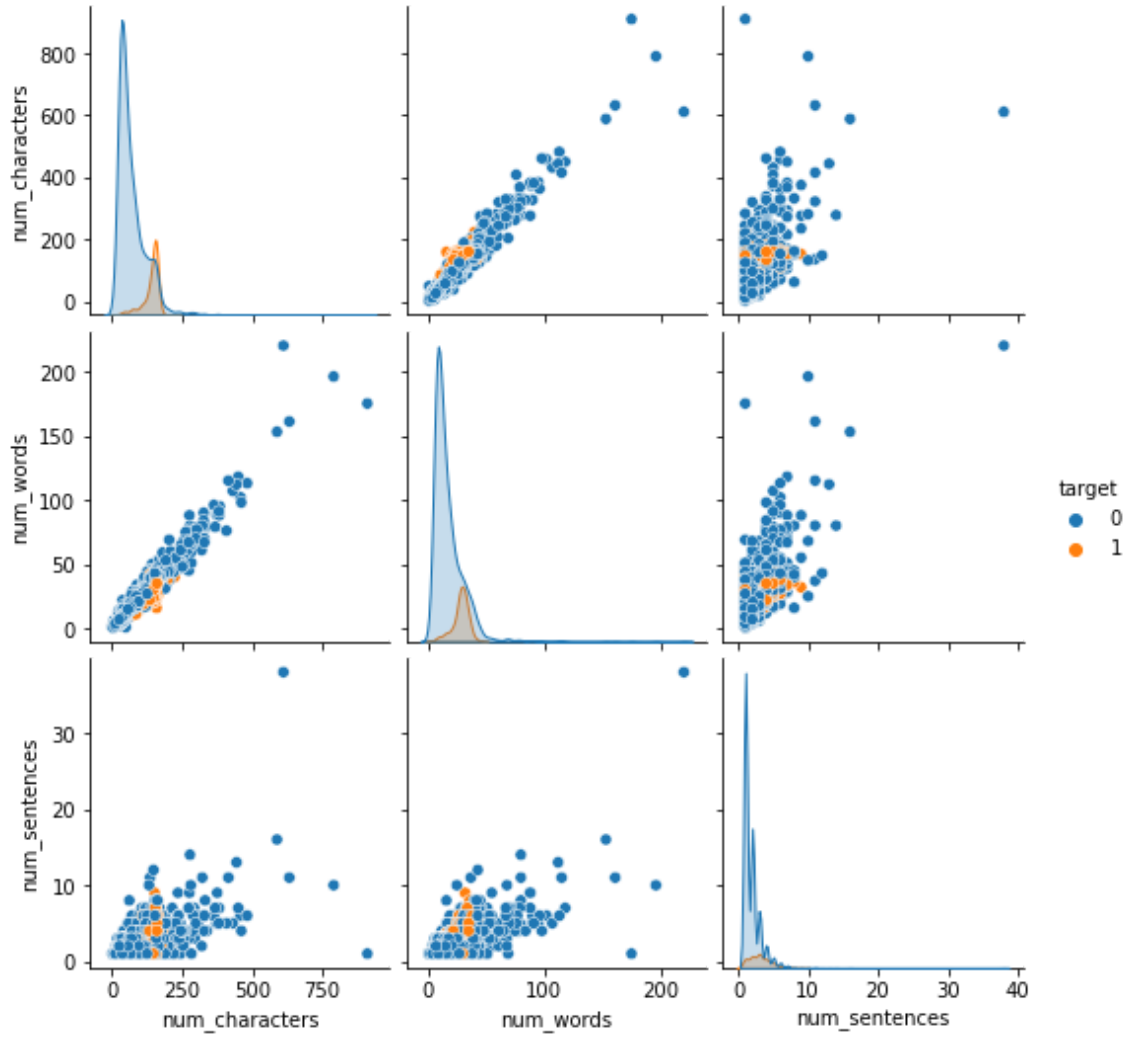
In [85]:
```
plt.figure(figsize=(12,6))
sns.histplot(df[df['target'] == 0]['num_words'])
sns.histplot(df[df['target'] == 1]['num_words'],color='red')
```

Out[85]: <AxesSubplot:xlabel='num_words', ylabel='Count'>



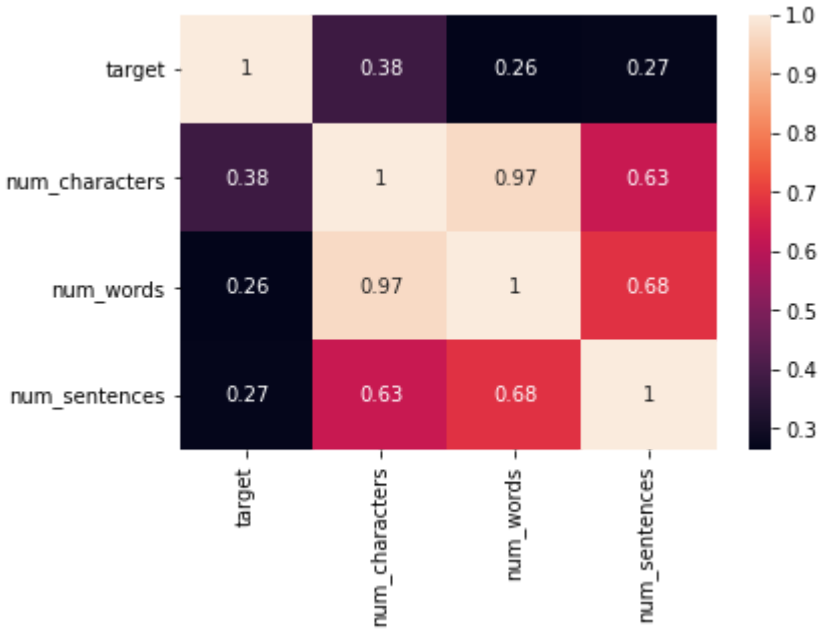
In [86]:
```
sns.pairplot(df,hue='target')
```

Out[86]: <seaborn.axisgrid.PairGrid at 0x16f88c4a4f0>



In [89]:
```
sns.heatmap(df.corr(),annot=True)
```

Out[89]: <AxesSubplot:>



3. Data Preprocessing

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

In [187]:
```
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.punctuat
            y.append(i)

    text = y[:]
    y.clear()

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


    return " ".join(y)
```

In [192]:
```
transform_text("I'm gonna be home soon and i don't want to talk about this
```

Out[192]:
```
'gon na home soon want talk stuff anymor tonight k cri enough today'
```

In [191]:
```
df['text'][10]
```

Out[191]:
```
"I'm gonna be home soon and i don't want to talk about this stuff anymore
tonight, k? I've cried enough today."
```

In [186]:
```
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()
ps.stem('loving')
```

Out[186]:
```
'love'
```

In [194]:
```
df['transformed_text'] = df['text'].apply(transform_text)
```

In [195]:
```
df.head()
```

Out[195]:

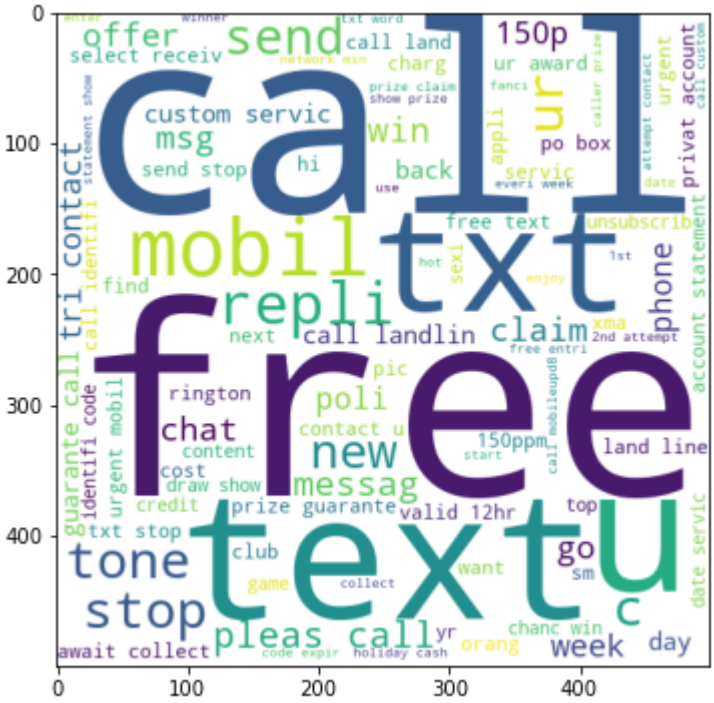
	target	text	num_characters	num_words	num_sentences	transformed_text
0	0	Go until jurong point, crazy.. Available only ...	111	24	2	go jurong point crazi avail bugi n great world...
1	0	Ok lar... Joking wif u oni...	29	8	2	ok lar joke wif u oni
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	155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

In [232]:
```
from wordcloud import WordCloud
wc = WordCloud(width=500,height=500,min_font_size=10,background_color='whit
```

In [233]:
```
spam_wc = wc.generate(df[df['target'] == 1]['transformed_text'].str.cat(sep
```

In [236]:
```
plt.figure(figsize=(15,6))
plt.imshow(spam_wc)
```

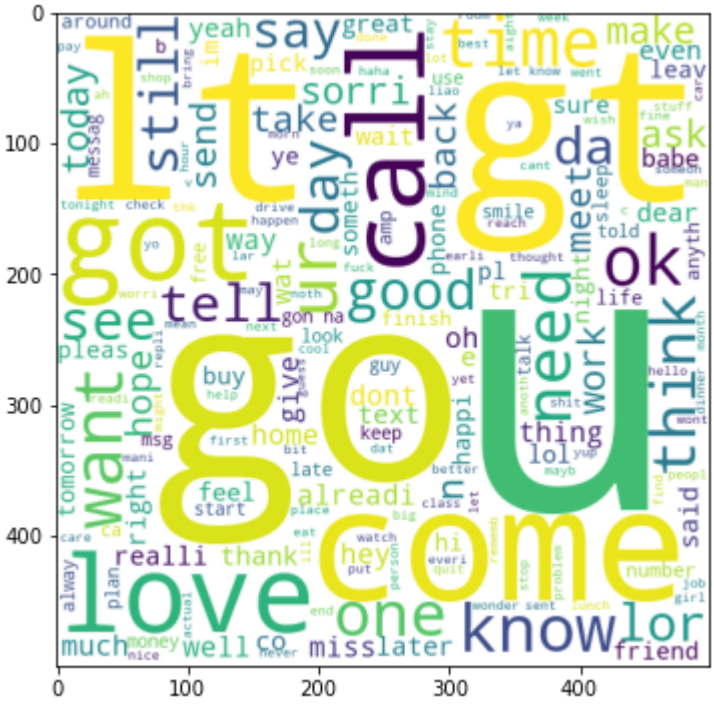
Out[236]:
```
<matplotlib.image.AxesImage at 0x16f87ea8cd0>
```



In [237]:
```
ham_wc = wc.generate(df[df['target'] == 0]['transformed_text'].str.cat(sep=
```

In [238]:
```
plt.figure(figsize=(15,6))
plt.imshow(ham_wc)
```

Out[238]:
```
<matplotlib.image.AxesImage at 0x16f87f6c280>
```



In [267]:
```
df.head()
```

Out[267]:

	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

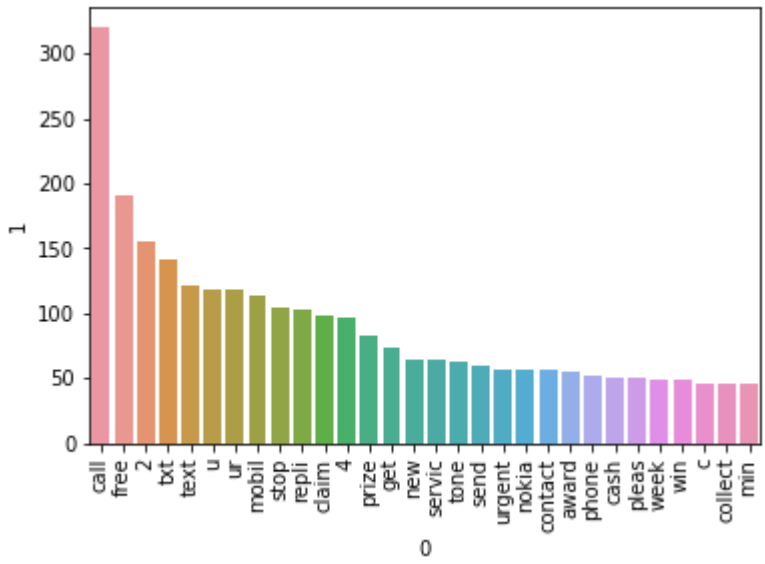
In [272]:
```
spam_corpus = []
for msg in df[df['target'] == 1]['transformed_text'].tolist():
    for word in msg.split():
        spam_corpus.append(word)
```

In [274]:
```
len(spam_corpus)
```

Out[274]: 9941

In [280]:
```
from collections import Counter
sns.barplot(pd.DataFrame(Counter(spam_corpus).most_common(30))[0],pd.DataFr
plt.xticks(rotation='vertical')
plt.show()
```

```
C:\Users\91842\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Futu
reWarning: Pass the following variables as keyword args: x, y. From versio
n 0.12, the only valid positional argument will be `data`, and passing oth
er arguments without an explicit keyword will result in an error or misint
erpretation.
  warnings.warn(
```



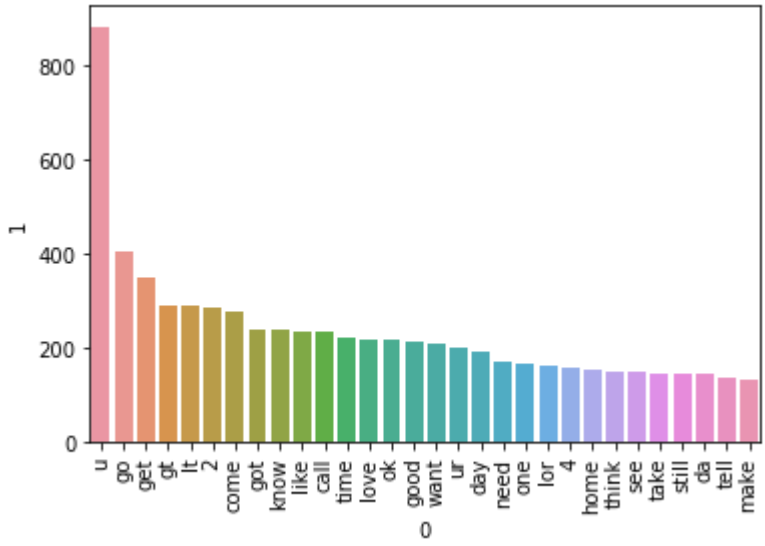
In [281]:
```
ham_corpus = []
for msg in df[df['target'] == 0]['transformed_text'].tolist():
    for word in msg.split():
        ham_corpus.append(word)
```

In [282]:
```
len(ham_corpus)
```

Out[282]: 35303

In [284]:
```
from collections import Counter
sns.barplot(pd.DataFrame(Counter(ham_corpus).most_common(30))[0],pd.DataFra
plt.xticks(rotation='vertical')
plt.show()
```

```
C:\Users\91842\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Futu
reWarning: Pass the following variables as keyword args: x, y. From versio
n 0.12, the only valid positional argument will be `data`, and passing oth
er arguments without an explicit keyword will result in an error or misint
erpretation.
  warnings.warn(
```



In [285]:
```
# Text Vectorization
# using Bag of Words
df.head()
```

Out[285]:

	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

4. Model Building

In [522]:
```
from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer
cv = CountVectorizer()
tfidf = TfidfVectorizer(max_features=3000)
```

In [523]:
```
X = tfidf.fit_transform(df['transformed_text']).toarray()
```

In [470]:
```
#from sklearn.preprocessing import MinMaxScaler
#scaler = MinMaxScaler()
#X = scaler.fit_transform(X)
```

In [483]:
```
# appending the num_character col to X
#X = np.hstack((X,df['num_characters'].values.reshape(-1,1)))
```

In [524]:
```
X.shape
```

Out[524]:
```
(5169, 3000)
```

In [525]:
```
y = df['target'].values
```

In [526]:
```
from sklearn.model_selection import train_test_split
```

In [527]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_s
```

In [528]:
```
from sklearn.naive_bayes import GaussianNB,MultinomialNB,BernoulliNB
from sklearn.metrics import accuracy_score,confusion_matrix,precision_score
```

In [489]:
```
gnb = GaussianNB()
mnb = MultinomialNB()
bnb = BernoulliNB()
```

In [490]:
```
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))
```

```
0.8916827852998066
[[808  88]
 [ 24 114]]
0.5643564356435643
```

In [529]:
```
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
```

In [492]:
```
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
```

In [493]:
```
# tfidf --> MNB
```

In [494]:
```
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
```

In [495]:
```
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)
```

In [496]:
```
clfs = {
    'SVC' : svc,
    'KN' : knc,
    'NB': mnb,
    'DT': dtc,
    'LR': lrc,
    'RF': rfc,
    'AdaBoost': abc,
    'BgC': bc,
    'ETC': etc,
    'GBDT':gbdt,
    'xgb':xgb
}
```

In [497]:
```
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
```

In [348]:
```
train_classifier(svc,X_train,y_train,X_test,y_test)
```

Out[348]:
```
(0.9729206963249516, 0.9741379310344828)
```

In [498]:
```
accuracy_scores = []
precision_scores = []

for name,clf in clfs.items():

    current_accuracy,current_precision = train_classifier(clf, X_train,y_tr

    print("For ",name)
    print("Accuracy - ",current_accuracy)
    print("Precision - ",current_precision)

    accuracy_scores.append(current_accuracy)
    precision_scores.append(current_precision)
```

```
C:\Users\91842\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1245: UndefinedMetricWarning: Precision is ill-defined and being set
to 0.0 due to no predicted samples. Use `zero_division` parameter to contr
ol this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```
For  SVC
Accuracy -  0.8665377176015474
Precision -  0.0
For  KN
Accuracy -  0.9284332688588007
Precision -  0.7711864406779662
For  NB
Accuracy -  0.9400386847195358
Precision -  1.0
For  DT
Accuracy -  0.9439071566731141
Precision -  0.8773584905660378
For  LR
Accuracy -  0.9613152804642167
Precision -  0.9711538461538461
For  RF
Accuracy -  0.9748549323017408
Precision -  0.9827586206896551
For  AdaBoost
Accuracy -  0.971953578336557
Precision -  0.9504132231404959
For  BgC
Accuracy -  0.9680851063829787
Precision -  0.9133858267716536
For  ETC
Accuracy -  0.97678916827853
Precision -  0.975
For  GBDT
Accuracy -  0.9487427466150871
Precision -  0.9292929292929293
```

```
C:\Users\91842\anaconda3\lib\site-packages\xgboost\sklearn.py:1146: UserWa
rning: 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 objec
t; 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)
```

```
[14:16:02] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release
_1.4.0/src/learner.cc:1095: Starting in XGBoost 1.3.0, the default evaluat
ion metric used with the objective 'binary:logistic' was changed from 'err
or' to 'logloss'. Explicitly set eval_metric if you'd like to restore the
old behavior.
For  xgb
Accuracy -  0.9700193423597679
Precision -  0.9421487603305785
```

In [386]:
```
performance_df = pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy':accuracy_
```

In [387]:
```
performance_df
```

Out[387]:

	Algorithm	Accuracy	Precision
1	KN	0.900387	1.000000
2	NB	0.959381	1.000000
8	ETC	0.977756	0.991453
5	RF	0.970019	0.990826
0	SVC	0.972921	0.974138
6	AdaBoost	0.962282	0.954128
10	xgb	0.971954	0.950413
4	LR	0.951644	0.940000
9	GBDT	0.951644	0.931373
7	BgC	0.957447	0.861538
3	DT	0.935203	0.838095

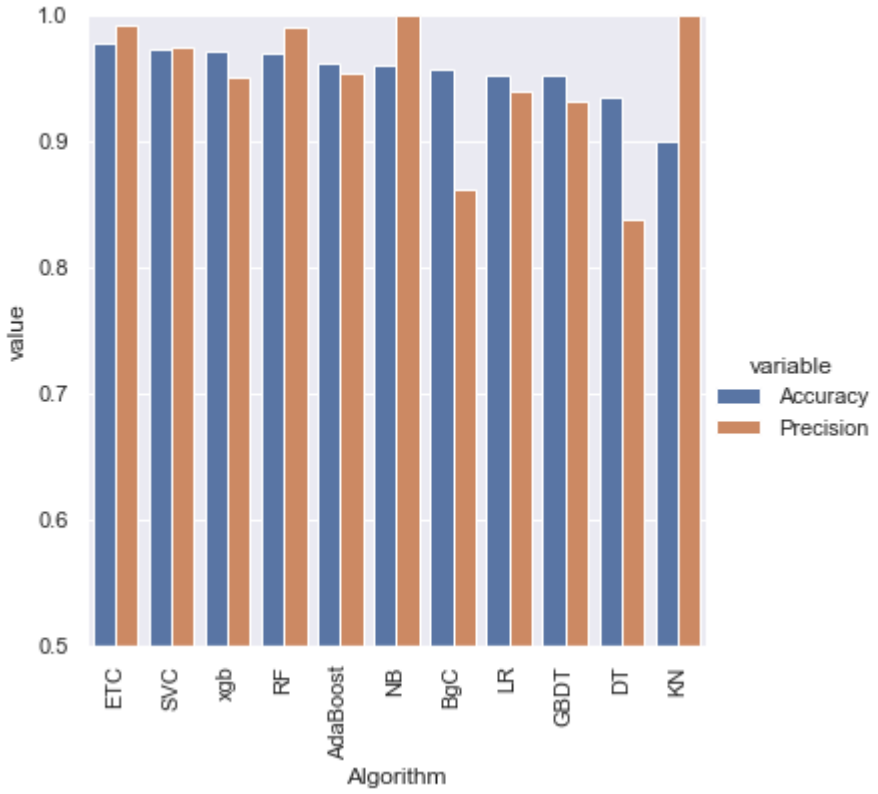
In [364]:
```
performance_df1 = pd.melt(performance_df, id_vars = "Algorithm")
```

In [365]:
```
performance_df1
```

Out[365]:

	Algorithm	variable	value
0	ETC	Accuracy	0.977756
1	SVC	Accuracy	0.972921
2	xgb	Accuracy	0.971954
3	RF	Accuracy	0.970019
4	AdaBoost	Accuracy	0.962282
5	NB	Accuracy	0.959381
6	BgC	Accuracy	0.957447
7	LR	Accuracy	0.951644
8	GBDT	Accuracy	0.951644
9	DT	Accuracy	0.935203
10	KN	Accuracy	0.900387
11	ETC	Precision	0.991453
12	SVC	Precision	0.974138
13	xgb	Precision	0.950413
14	RF	Precision	0.990826
15	AdaBoost	Precision	0.954128
16	NB	Precision	1.000000
17	BgC	Precision	0.861538
18	LR	Precision	0.940000
19	GBDT	Precision	0.931373
20	DT	Precision	0.838095
21	KN	Precision	1.000000

In [385]:
```
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()
```



In [ ]:
```
# model improve
# 1. Change the max_features parameter of TfIdf
```

In [428]:
```
temp_df = pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy_max_ft_3000':accu
```

In [454]:
```
temp_df = pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy_scaling':accuracy
```

In [452]:
```
new_df = performance_df.merge(temp_df,on='Algorithm')
```

In [456]:
```
new_df_scaled = new_df.merge(temp_df,on='Algorithm')
```

In [499]:
```
temp_df = pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy_num_chars':accura
```

In [501]:
```
new_df_scaled.merge(temp_df,on='Algorithm')
```

Out[501]:

	Algorithm	Accuracy	Precision	Accuracy_max_ft_3000	Precision_max_ft_3000	Accuracy_
0	KN	0.900387	1.000000	0.905222	1.000000	0
1	NB	0.959381	1.000000	0.971954	1.000000	0
2	ETC	0.977756	0.991453	0.979691	0.975610	0
3	RF	0.970019	0.990826	0.975822	0.982906	0
4	SVC	0.972921	0.974138	0.974855	0.974576	0
5	AdaBoost	0.962282	0.954128	0.961315	0.945455	0
6	xgb	0.971954	0.950413	0.968085	0.933884	0
7	LR	0.951644	0.940000	0.956480	0.969697	0
8	GBDT	0.951644	0.931373	0.946809	0.927835	0
9	BgC	0.957447	0.861538	0.959381	0.869231	0
10	DT	0.935203	0.838095	0.931335	0.831683	0

In [514]:
```
# 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
```

In [515]:
```
voting = VotingClassifier(estimators=[('svm', svc), ('nb', mnb), ('et', etc
```

In [516]:
```
voting.fit(X_train,y_train)
```

Out[516]:
```
VotingClassifier(estimators=[('svm',
                              SVC(gamma=1.0, kernel='sigmoid',
                                  probability=True)),
                             ('nb', MultinomialNB()),
                             ('et',
                              ExtraTreesClassifier(n_estimators=50,
                                                   random_state=2))],
                 voting='soft')
```

In [517]:
```
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
```

In [518]:
```
# Applying stacking
estimators=[('svm', svc), ('nb', mnb), ('et', etc)]
final_estimator=RandomForestClassifier()
```

In [519]:
```
from sklearn.ensemble import StackingClassifier
```

In [520]:
```
clf = StackingClassifier(estimators=estimators, final_estimator=final_estim
```

In [521]:
```
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 0.9787234042553191
Precision 0.9328358208955224
```

In [530]:
```
import pickle
pickle.dump(tfidf,open('vectorizer.pkl','wb'))
pickle.dump(mnb,open('model.pkl','wb'))
```

In [ ]:
```

```