In [1]:

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

In [2]:

```
df = pd.read_csv('spam.csv',encoding='ISO-8859-1')
```

In [3]:

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

Out[3]:

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
3043	ham	Let me know how it changes in the next 6hrs. I...	NaN	NaN	NaN
5557	ham	No. I meant the calculation is the same. That ...	NaN	NaN	NaN
988	ham	Yun ah.the ubi one say if Ì_ wan call by tomor...	NaN	NaN	NaN
4521	ham	DO U WANT 2 MEET UP 2MORRO	NaN	NaN	NaN
4064	ham	Dont kick coco when he's down	NaN	NaN	NaN

In [4]:

```
df.shape
```

Out[4]:

```
(5572, 5)
```

In [5]:

```
# 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]:

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

In [8]:

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

Out[8]:

	v1	v2
1689	ham	Bring tat cd don forget
4232	ham	My love ... I hope your not doing anything dra...
3361	ham	No messages on her phone. I'm holding it now
1487	ham	I told your number to gautham..
2472	spam	Final Chance! Claim ur å£150 worth of discount...

In [9]:

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

Out[9]:

	target	text
2555	spam	FreeMSG You have been awarded a FREE mini DIGI...
816	ham	He has lots of used ones babe, but the model d...
4063	ham	Prof: you have passed in all the papers in thi...
3328	ham	Sac will score big hundred.he is set batsman:-)
2469	ham	* Am on my way

In [10]:

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

In [11]:

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

In [12]:

```python
df.head()
```

Out[12]:

	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 [13]:

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

Out[13]:

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

In [14]:
```
# check for duplicate values
df.duplicated().sum()
```

Out[14]:
```
403
```

In [15]:
```
# remove duplicates
df = df.drop_duplicates(keep='first')
```

In [16]:
```
df.duplicated().sum()
```

Out[16]:
```
0
```

In [17]:
```
df.shape
```

Out[17]:
```
(5169, 2)
```

2.EDA

In [18]:
```
df.head()
```

Out[18]:

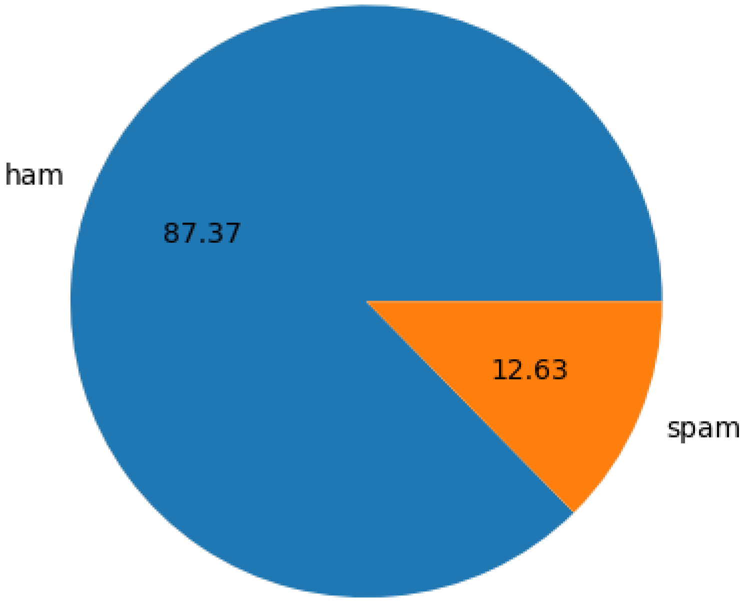
	target	text
0	0	Go until jurong point, crazy.. Available only ...
1	0	Ok lar... Joking wif u oni...
2	1	Free entry in 2 a wkly comp to win FA Cup fina...
3	0	U dun say so early hor... U c already then say...
4	0	Nah I don't think he goes to usf, he lives aro...

In [19]:
```
df['target'].value_counts()
```

Out[19]:
```
target
0    4516
1     653
Name: count, dtype: int64
```

In [20]:

```
import matplotlib.pyplot as plt
plt.pie(df['target'].value_counts(), labels=['ham','spam'],autopct="%0.
plt.show()
```



In [21]:

```
# Data is imbalanced
```

In [22]:

```
import nltk
```

In [23]:

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

In [24]:

```
df.head()
```

Out[24]:

	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 [25]:

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

In [26]:
```python
df.head()
```

Out[26]:

	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 [27]:
```python
df['num_sentences'] = df['text'].apply(lambda x:len(nltk.sent_tokenize(
```

In [28]:
```python
df.head()
```

Out[28]:

	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 [29]:
```python
df[['num_characters','num_words','num_sentences']].describe()
```

Out[29]:

	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

In [30]:

```
# ham
df[df['target'] == 0][['num_characters','num_words','num_sentences']].d
```

Out[30]:

	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

In [31]:

```
#spam
df[df['target'] == 1][['num_characters','num_words','num_sentences']].d
```

Out[31]:

	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

In [32]:

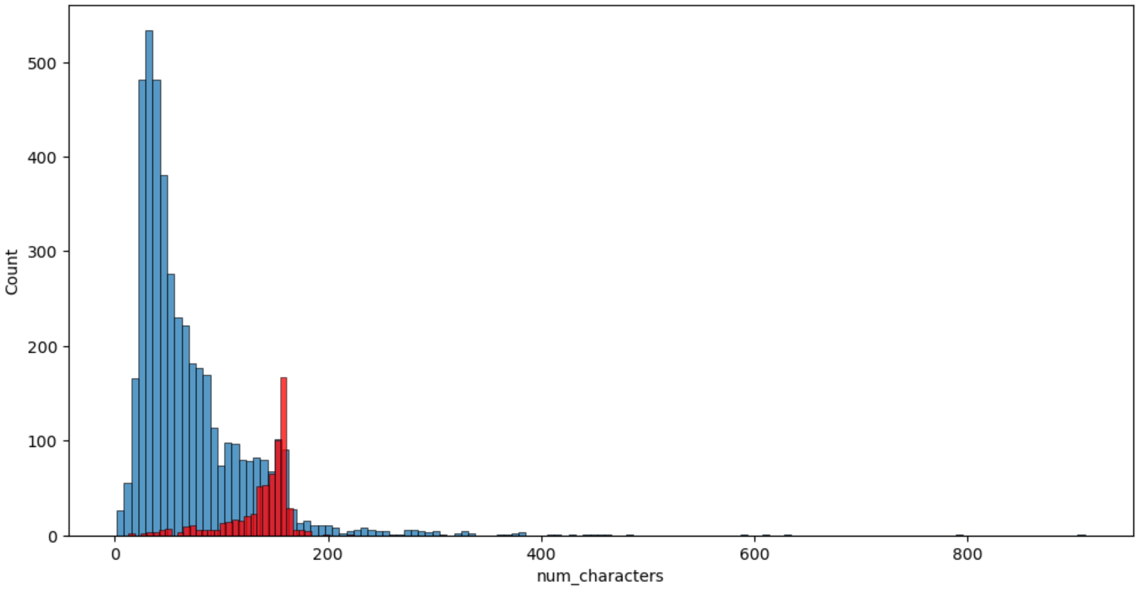
```
import seaborn as sns
```

In [33]:

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

Out[33]:

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

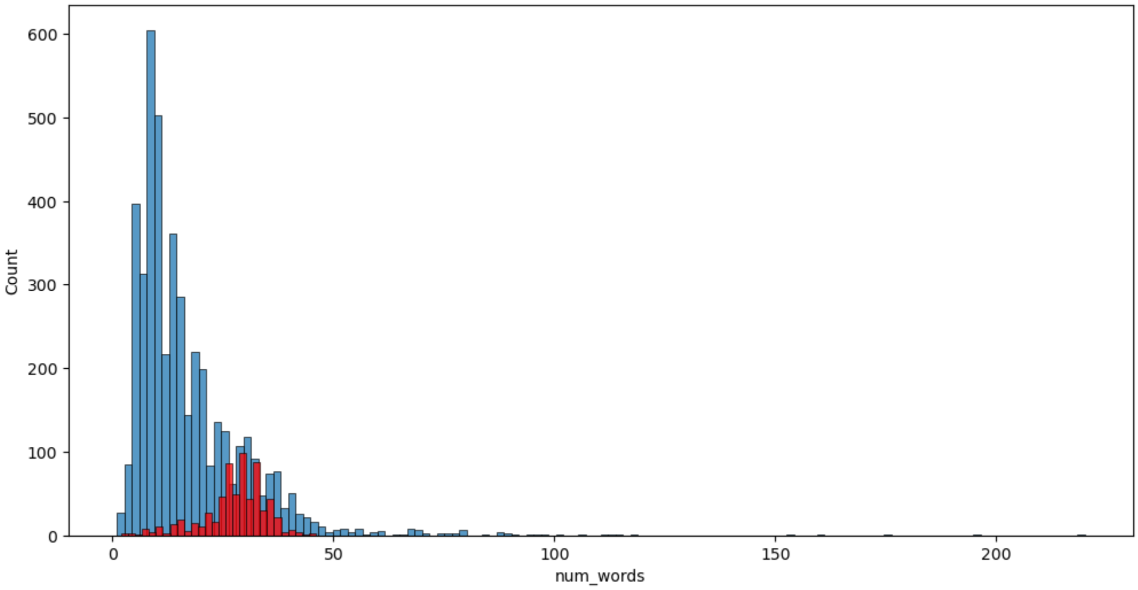


In [34]:

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

Out[34]:

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



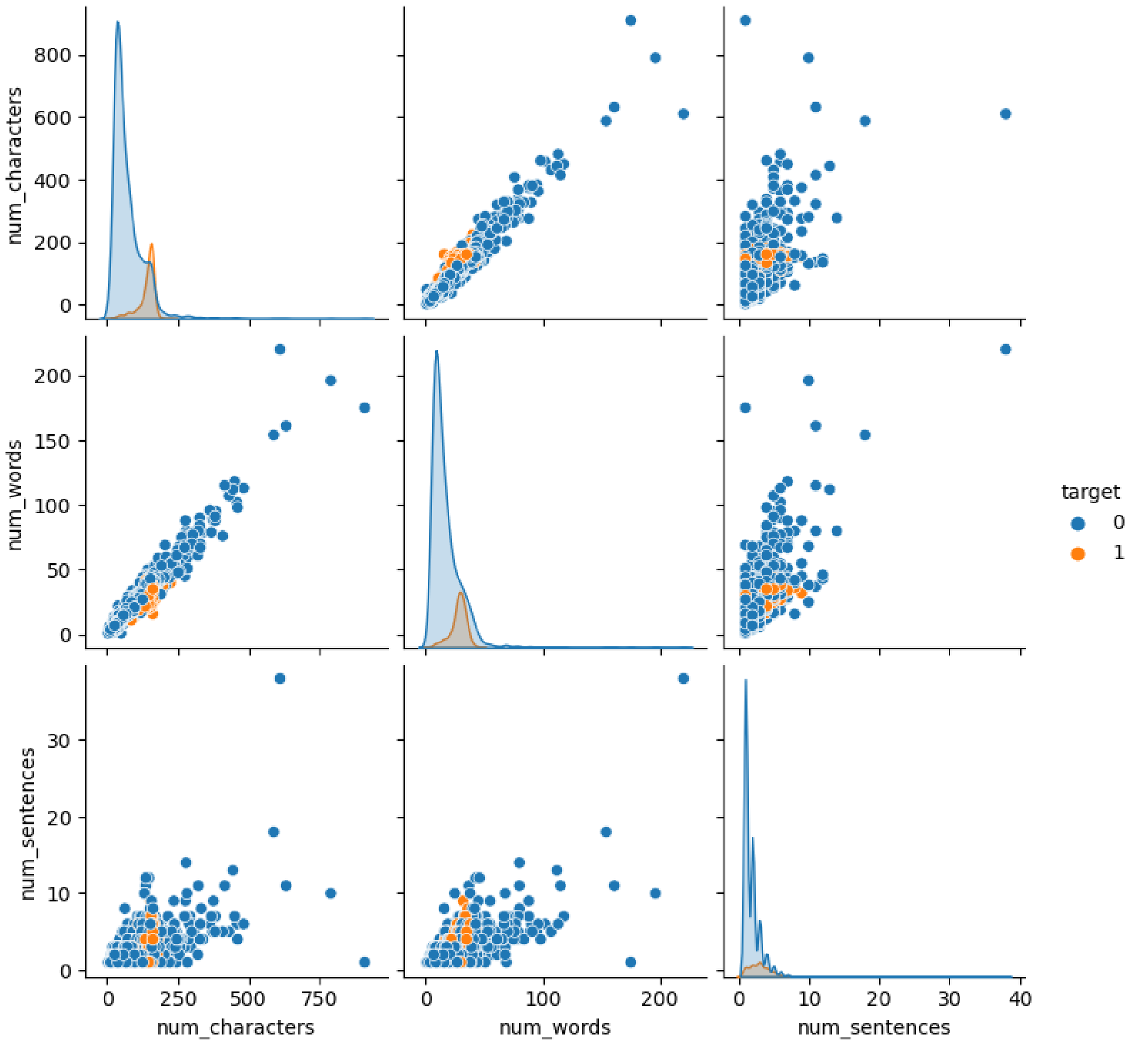
In [35]:

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

```
C:\Users\91861\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWa
rning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

Out[35]:

```
<seaborn.axisgrid.PairGrid at 0x23a5713ec90>
```



3. Data Preprocessing

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

In [36]:

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

In [37]:

```
from nltk.corpus import stopwords

```

In [38]:
```
import string
```

In [39]:
```
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()
```

In [40]:
```
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.punc
            y.append(i)

    text = y[:]
    y.clear()

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


    return " ".join(y)


```

In [41]:
```
transform_text("I'm gonna be home soon and i don't want to talk about t
```

Out[41]:
```
'gon na home soon want talk stuff anymor tonight k cri enough today'
```

In [42]:
```
df['text'][10]
```

Out[42]:
```
"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 [43]:
```
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()
ps.stem('loving')
```

Out[43]:
```
'love'
```

In [44]:
```
df['transformed_text'] = df['text'].apply(transform_text)
```

In [45]:

```
df.head()
```

Out[45]:

	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 [46]:

```
# !pip install wordcloud
```

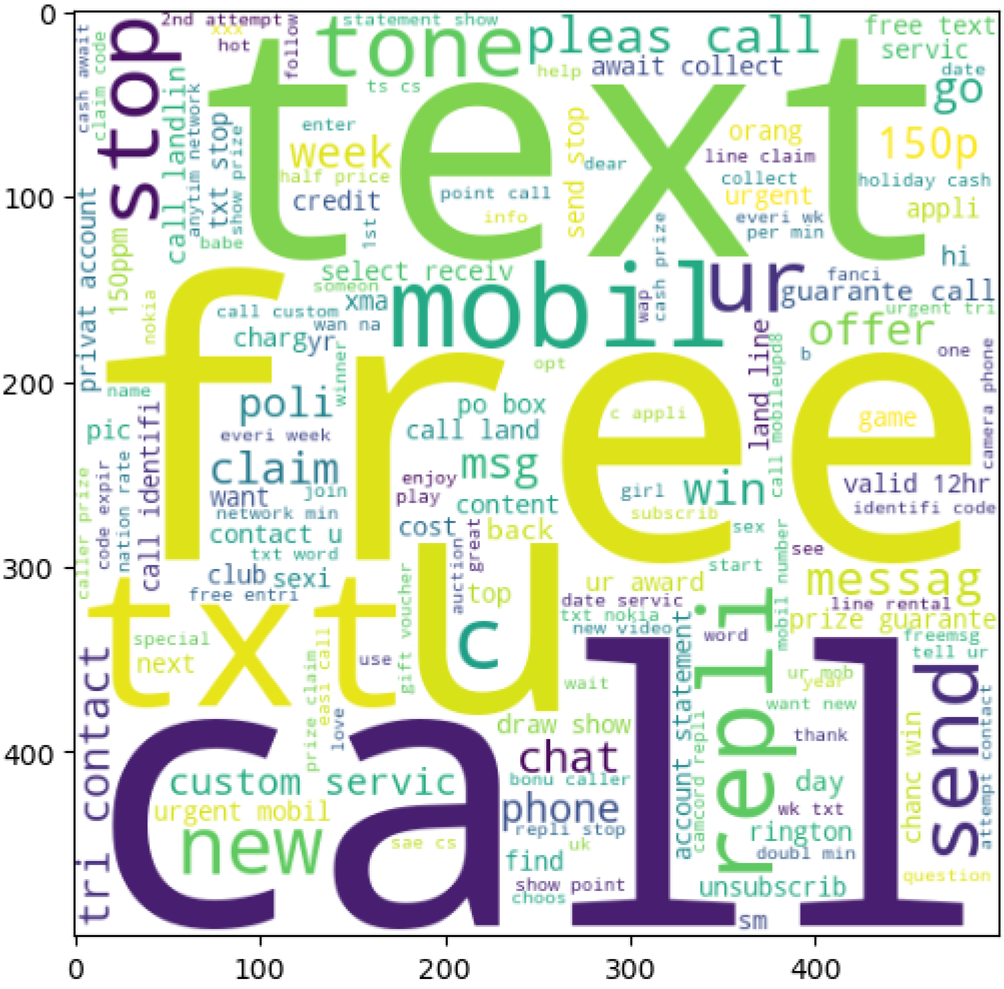
In [47]:

```
from wordcloud import WordCloud
wc = WordCloud(width=500,height=500,min_font_size=10,background_color='
```

In [48]:

```
spam_wc = wc.generate(df[df['target'] == 1]['transformed_text'].str.cat
In [49]: plt.figure(figsize=(15,6))
         plt.imshow(spam_wc)
```

```
Out[49]: <matplotlib.image.AxesImage at 0x23a5945c150>
```



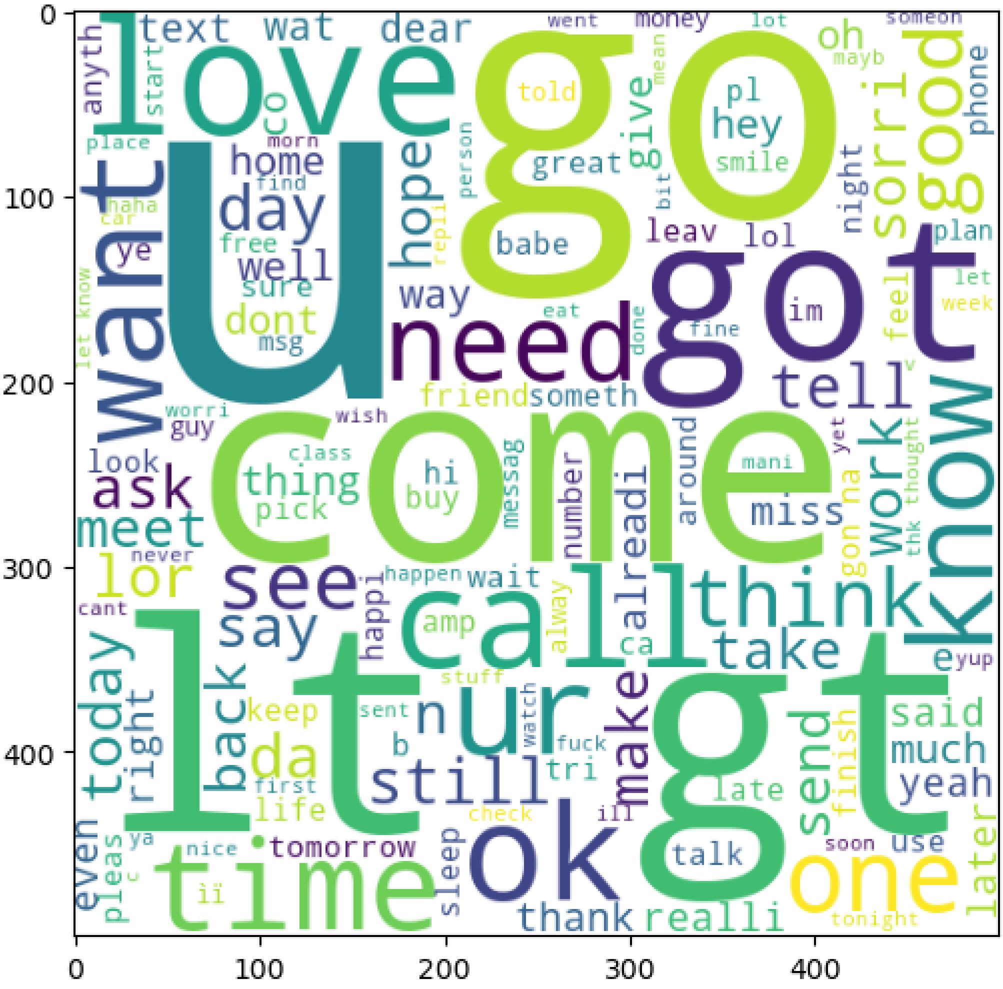
```
In [50]: ham_wc = wc.generate(df[df['target'] == 0]['transformed_text'].str.cat(
```

In [51]:

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

Out[51]:

```
<matplotlib.image.AxesImage at 0x23a595a2790>
```



In [52]:

```
df.head()
```

Out[52]:

	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 [53]:

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

```

In [54]:

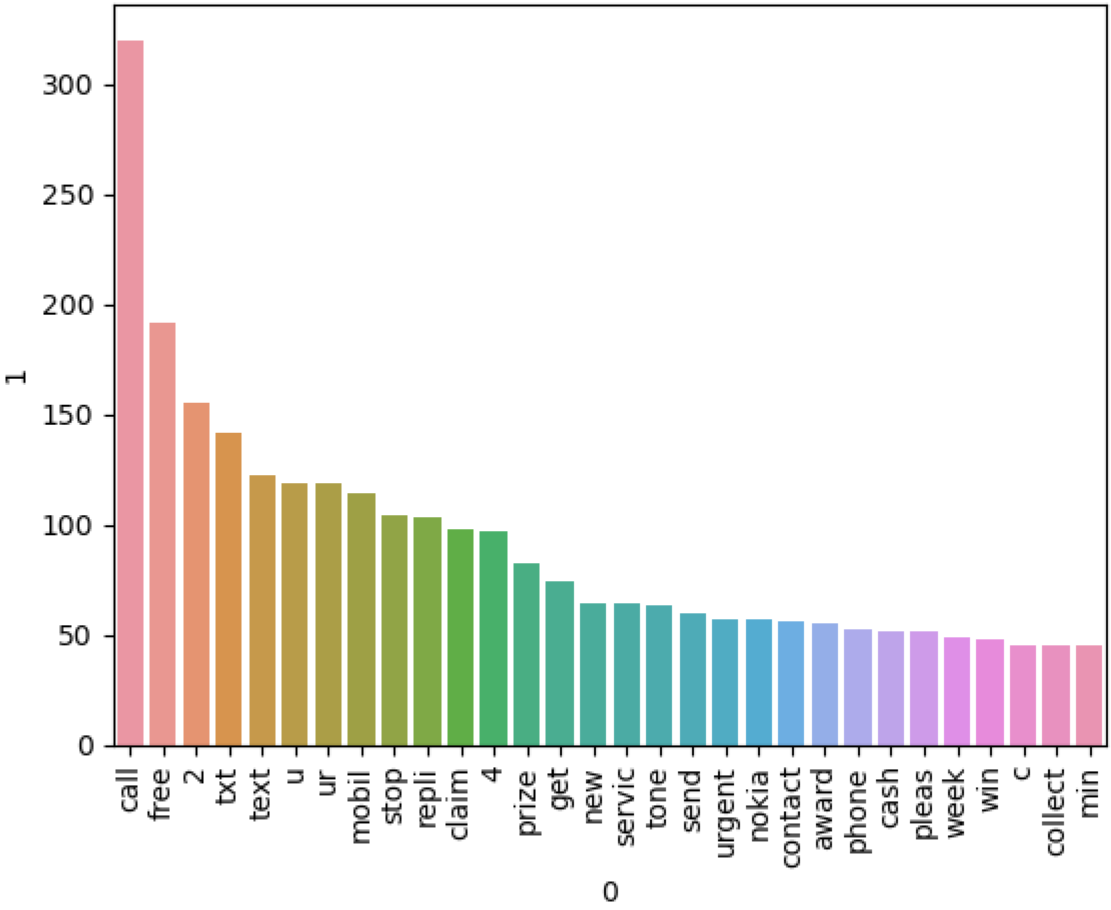
```
len(spam_corpus)
```

Out[54]:

```
9939
```

In [55]:

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



In [56]:

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

In [57]:

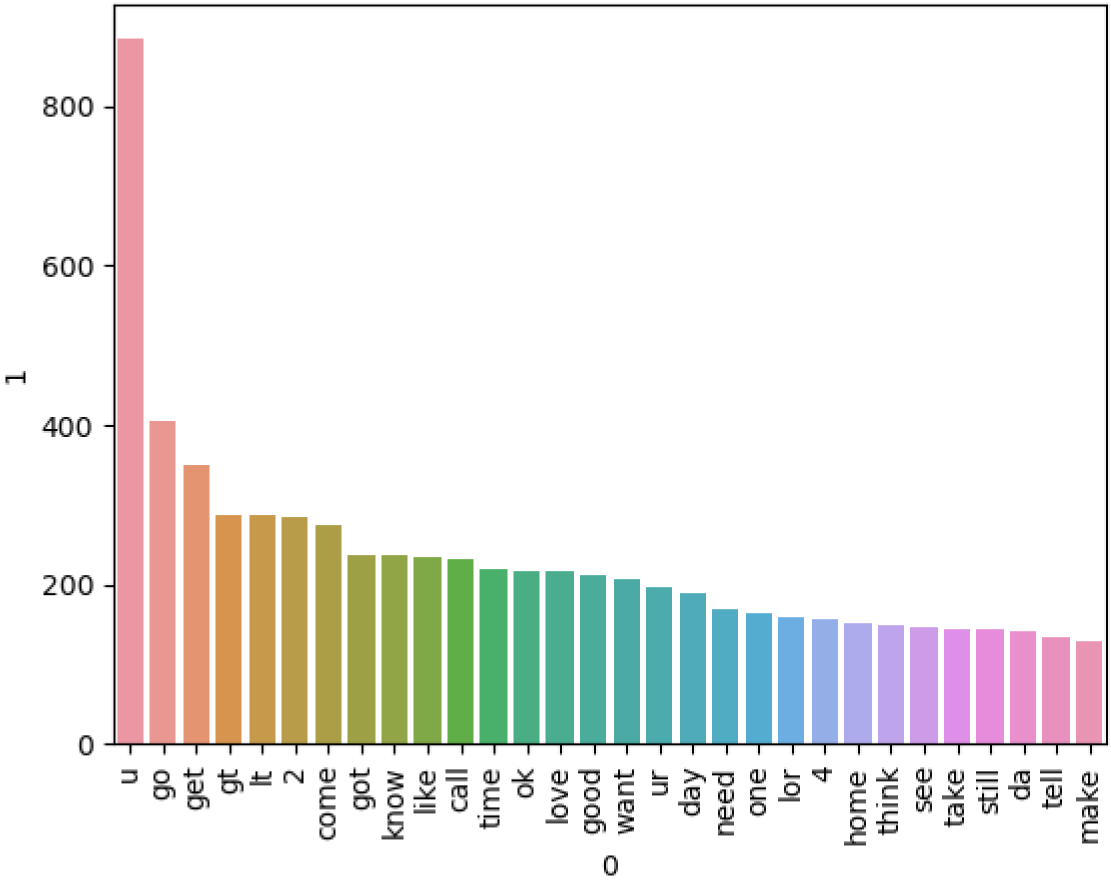
```
len(ham_corpus)
```

Out[57]:

```
35404
```

In [58]:

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



In [59]:

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

Out[59]:

	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 [60]:

```
from sklearn.feature_extraction.text import CountVectorizer,TfidfVector
cv = CountVectorizer()
tfidf = TfidfVectorizer(max_features=3000)
```

In [61]:

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

In [62]:

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

In [63]:

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

In [64]:

```
X.shape
```

Out[64]:

```
(5169, 3000)
```

In [65]:

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

In [66]:

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

In [67]:

```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,rand
```

In [68]:

```
from sklearn.naive_bayes import GaussianNB,MultinomialNB,BernoulliNB
from sklearn.metrics import accuracy_score,confusion_matrix,precision_s
```

In [69]:

```
gnb = GaussianNB()
mnb = MultinomialNB()
bnb = BernoulliNB()
```

In [70]:

```
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
```

In [71]:

```
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
```

In [72]:

```
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 [73]:

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

In [74]:

```
# !pip install xgboost
```

In [75]:

```
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 [76]:

```
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 [77]:

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

In [78]:

```
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 [79]:

```
train_classifier(svc,X_train,y_train,X_test,y_test)
```

Out[79]:

```
(0.9758220502901354, 0.9747899159663865)
```

In [80]:

```python
accuracy_scores = []
precision_scores = []

for name,clf in clfs.items():

    current_accuracy,current_precision = train_classifier(clf, X_train,

    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 -  0.9332688588007737
Precision -  0.8415841584158416
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 -  0.9333333333333333
```

In [81]:

```python
performance_df = pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy':accur
```

In [82]:

```
performance_df
```

Out[82]:

	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
10	xgb	0.967118	0.933333
6	AdaBoost	0.960348	0.929204
9	GBDT	0.946809	0.919192
7	BgC	0.958414	0.868217
3	DT	0.933269	0.841584

In [83]:

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

In [84]:

```
performance_df1
```

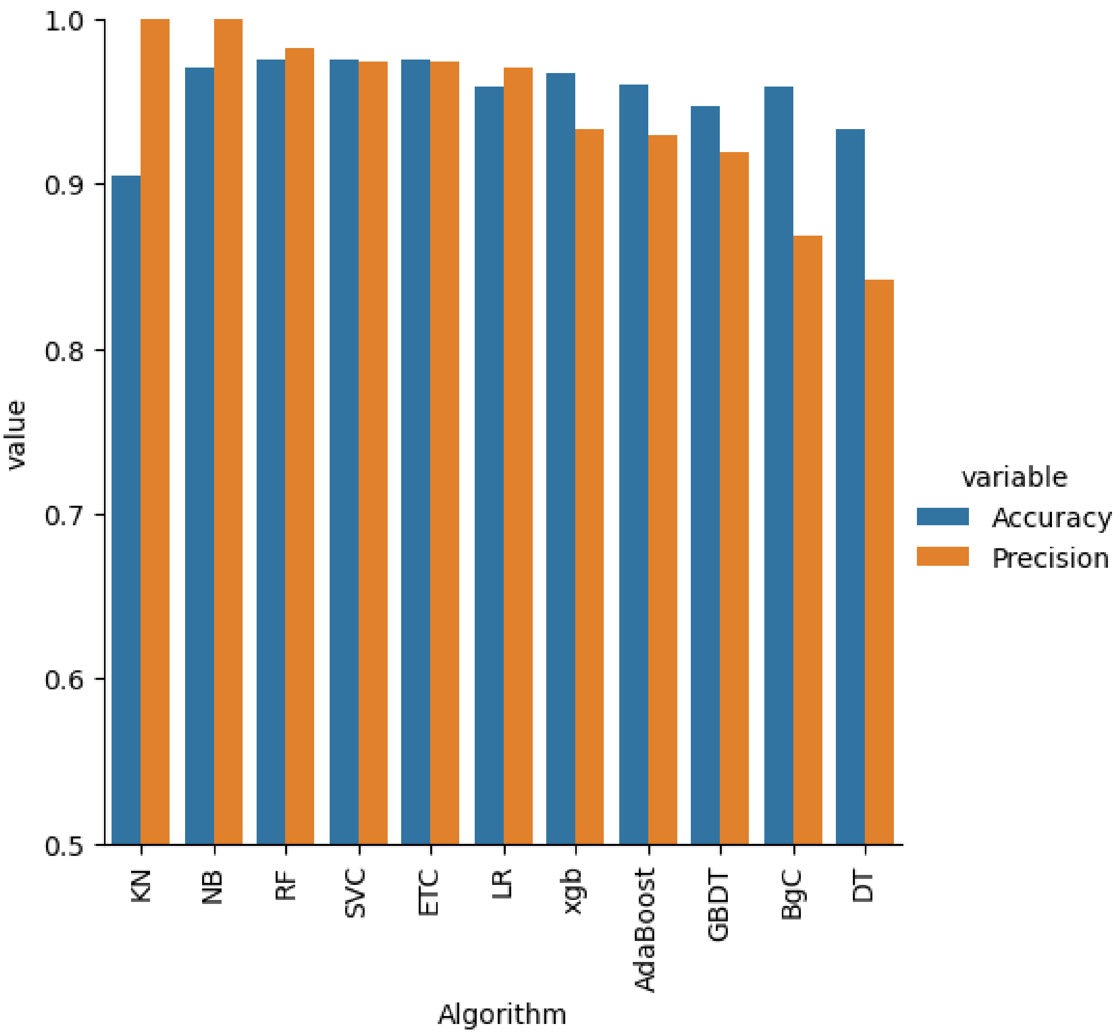
Out[84]:

	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	xgb	Accuracy	0.967118
7	AdaBoost	Accuracy	0.960348
8	GBDT	Accuracy	0.946809
9	BgC	Accuracy	0.958414
10	DT	Accuracy	0.933269
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	xgb	Precision	0.933333
18	AdaBoost	Precision	0.929204
19	GBDT	Precision	0.919192
20	BgC	Precision	0.868217
21	DT	Precision	0.841584

In [85]:

```
sns.catplot(x = 'Algorithm', y='value',
               hue = 'variable',data=performance_df1, kind='bar',height
plt.ylim(0.5,1.0)
plt.xticks(rotation='vertical')
plt.show()
```

```
C:\Users\91861\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWa
rning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```



In [86]:

```
# model improve
# 1. Change the max_features parameter of TfIdf
```

In [87]:

```
temp_df = pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy_max_ft_3000':
```

In [88]:

```
temp_df = pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy_scaling':accu
```

In [89]:

```
new_df = performance_df.merge(temp_df,on='Algorithm')
```

In [90]:

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

In [91]:

```
temp_df = pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy_num_chars':ac
```

In [92]:

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

Out[92]:

	Algorithm	Accuracy	Precision	Accuracy_scaling_x	Precision_scaling_x	Accuracy_scalin
0	KN	0.905222	1.000000	0.905222	1.000000	0.905
1	NB	0.970986	1.000000	0.970986	1.000000	0.970
2	RF	0.975822	0.982906	0.975822	0.982906	0.975
3	SVC	0.975822	0.974790	0.975822	0.974790	0.975
4	ETC	0.974855	0.974576	0.974855	0.974576	0.974
5	LR	0.958414	0.970297	0.958414	0.970297	0.958
6	xgb	0.967118	0.933333	0.967118	0.933333	0.967
7	AdaBoost	0.960348	0.929204	0.960348	0.929204	0.960
8	GBDT	0.946809	0.919192	0.946809	0.919192	0.946
9	BgC	0.958414	0.868217	0.958414	0.868217	0.958
10	DT	0.933269	0.841584	0.933269	0.841584	0.933

In [93]:

```
# 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
```

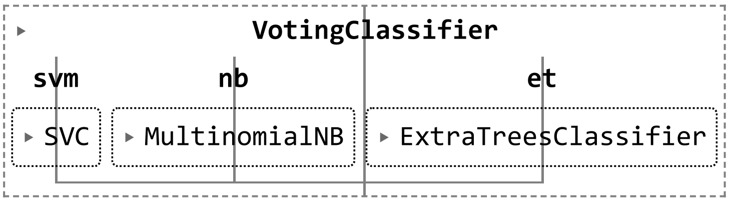
In [94]:

```
voting = VotingClassifier(estimators=[('svm', svc), ('nb', mnb), ('et',
```

In [95]:

```
voting.fit(X_train,y_train)
```

Out[95]:



In [96]:

```
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 [97]:

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

In [98]:

```
from sklearn.ensemble import StackingClassifier
```

In [99]:

```
clf = StackingClassifier(estimators=estimators, final_estimator=final_e
```

In [100]:

```
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 0.9393939393939394
```

In [101]:

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

In [ ]:

```

```