In [1]:

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

In [2]:

```
df = pd.read_csv('C:/Users/Megha Sharma/Desktop/MEGHA/MSC DATA SCIENCE/INTERNSHI
```

In [3]:

```
df.head()
```

Out[3]:

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN

In [4]:

```
df.shape
```

Out[4]:

```
(5572, 5)
```

1. Data Cleaning

In [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
```

In [6]:

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

In [7]:

```
df.head()
```

Out[7]:

	v1	v2
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

In [8]:

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

Out[8]:

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

In [9]:

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

In [10]:

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

In [11]:

```
df.head()
```

Out[11]:

	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 [12]:

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

Out[12]:

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

In [13]:

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

Out[13]:

```
403
```

In [14]:

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

In [15]:

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

Out[15]:

```
0
```

In [16]:

```
df.shape
```

Out[16]:

```
(5169, 2)
```

2.EDA

In [17]:

```
df.head()
```

Out[17]:

	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 [18]:

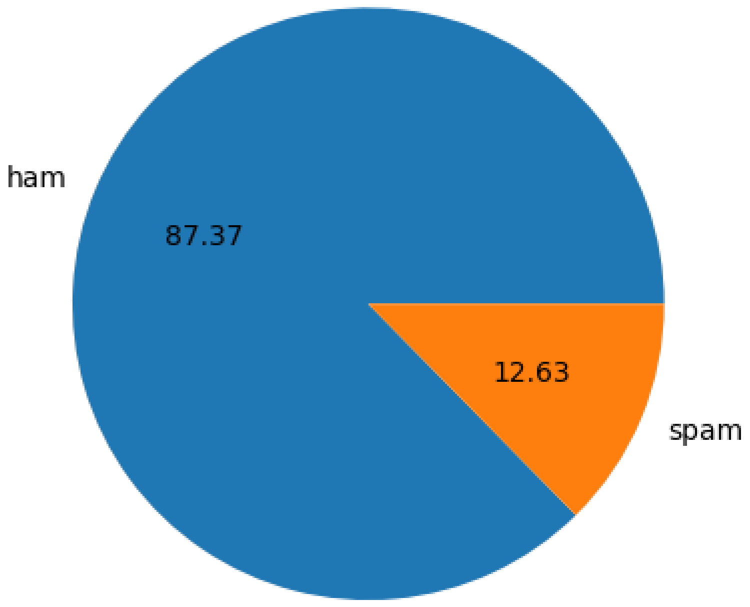
```
df['target'].value_counts()
```

Out[18]:

```
target
0    4516
1     653
Name: count, dtype: int64
```

In [19]:

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



In [20]:

```python
import nltk
```

In [21]:

```python
# num of characters
df['num_characters'] = df['text'].apply(len)
```

In [22]:

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

Out[22]:

	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 [23]:

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

In [24]:

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

Out[24]:

	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 [25]:

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

In [26]:

```
df.head()
```

Out[26]:

	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 [27]:

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

Out[27]:

	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 [28]:

```
df[df['target'] == 0][['num_characters','num_words','num_sentences']].describe()
```

Out[28]:

	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 [29]:

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

Out[29]:

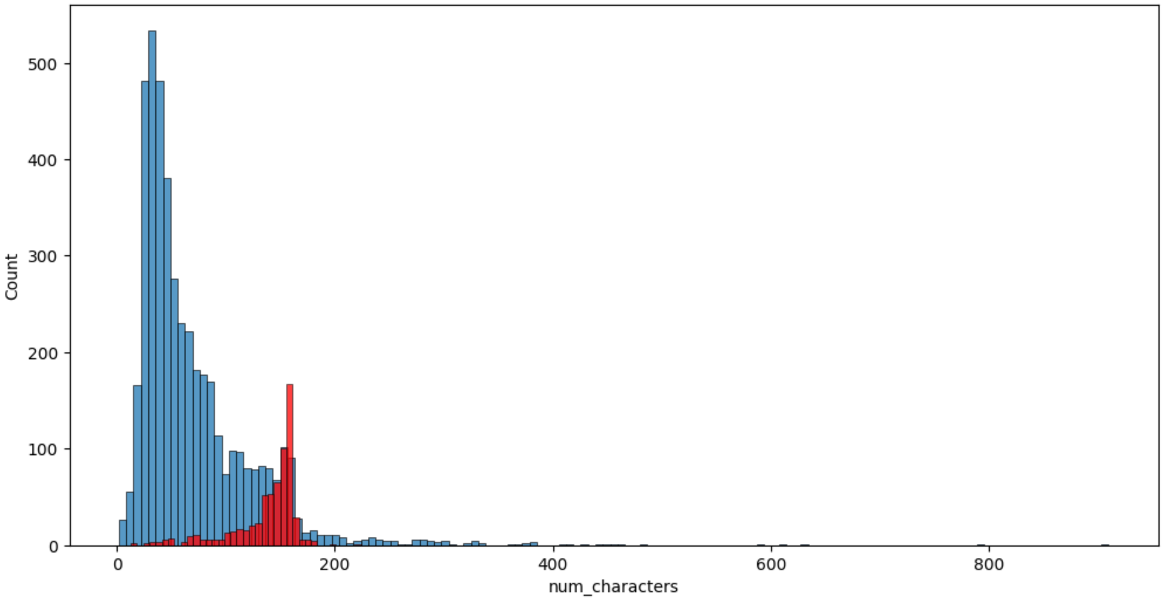
	num_characters	num_words	num_sentences
count	653.000000	653.000000	653.000000
mean	137.891271	27.667688	2.970904
std	30.137753	7.008418	1.488425
min	13.000000	2.000000	1.000000
25%	132.000000	25.000000	2.000000
50%	149.000000	29.000000	3.000000
75%	157.000000	32.000000	4.000000
max	224.000000	46.000000	9.000000

In [30]:

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

Out[30]:

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

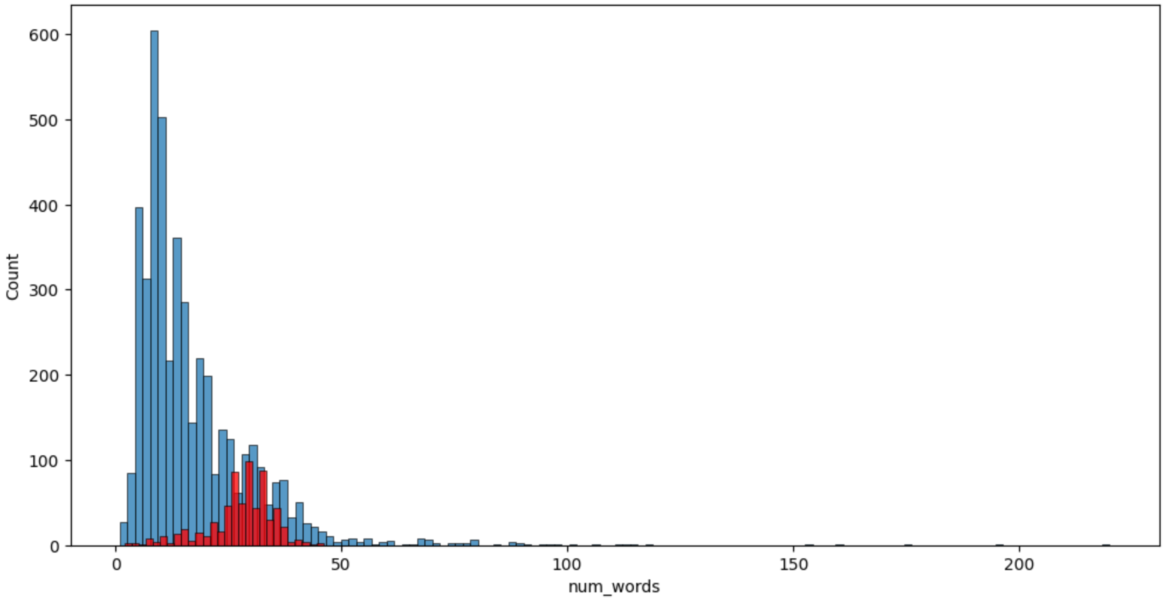


In [31]:

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

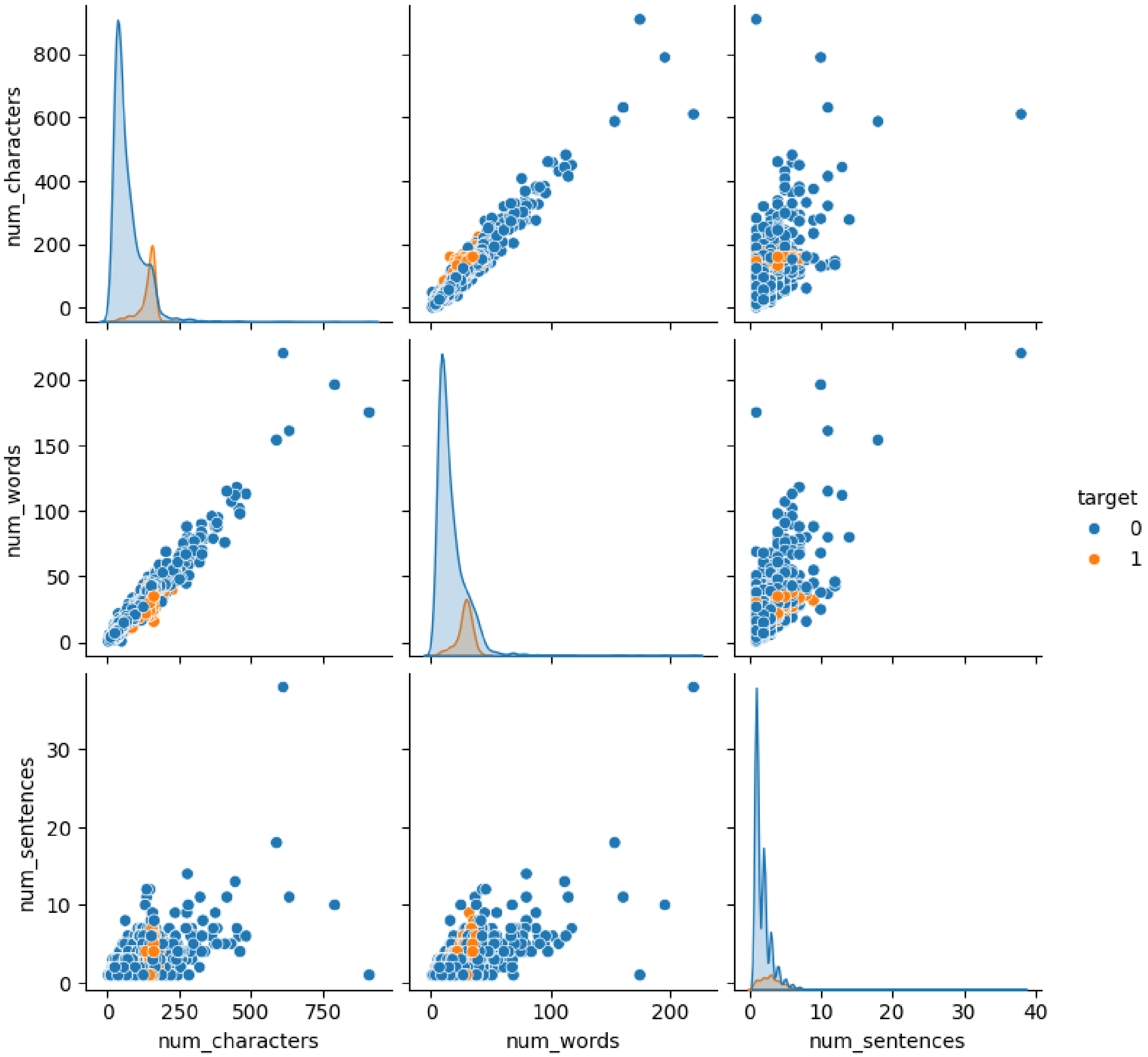
Out[31]:

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



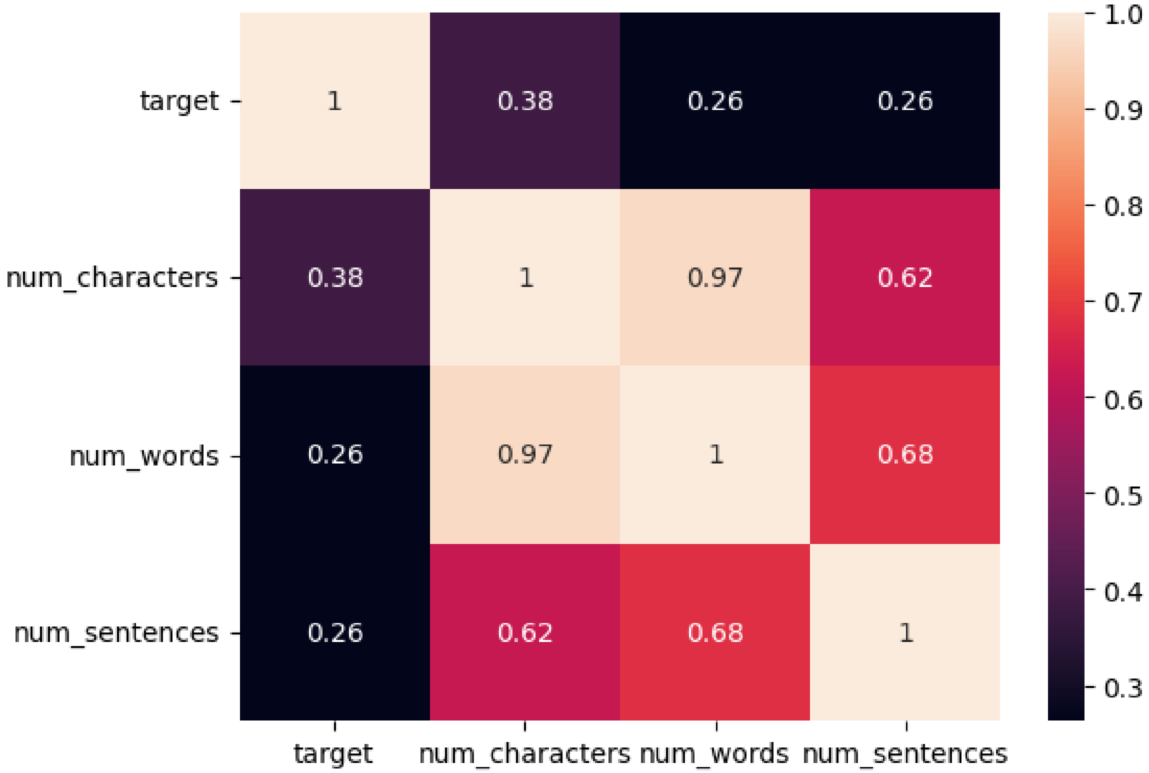
```
In [32]: sns.pairplot(df,hue='target')
```

```
Out[32]: <seaborn.axisgrid.PairGrid at 0x263a12347f0>
```



```
In [33]: numeric_values = df.select_dtypes( include = ['number'])
         sns.heatmap(numeric_values.corr(),annot=True)
```

```
Out[33]: <Axes: >
```



3. Data Preprocessing

Lower case

Tokenization

Removing special characters

Removing stop words and punctuation

Stemming

In [34]:

```
import nltk
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
import string
```

In [35]:

```
def transform_text(text):
    ps = PorterStemmer()
    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)
```

In [36]:
```
transformed_text = transform_text("I'm gonna be home soon and i don't want to ta
print(transformed_text)
```

```
gon na home soon want talk stuff anymor tonight k cri enough today
```

In [37]:
```
df['text'][10]
```

Out[37]:
```
"I'm gonna be home soon and i don't want to talk about this stuff anymore tonig
ht, k? I've cried enough today."
```

In [38]:
```
ps = PorterStemmer()
ps.stem('loving')
```

Out[38]:
```
'love'
```

In [39]:
```
df['transformed_text'] = df['text'].apply(transform_text)
```

In [40]:
```
df.head()
```

Out[40]:

	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 [41]:

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

In [42]:

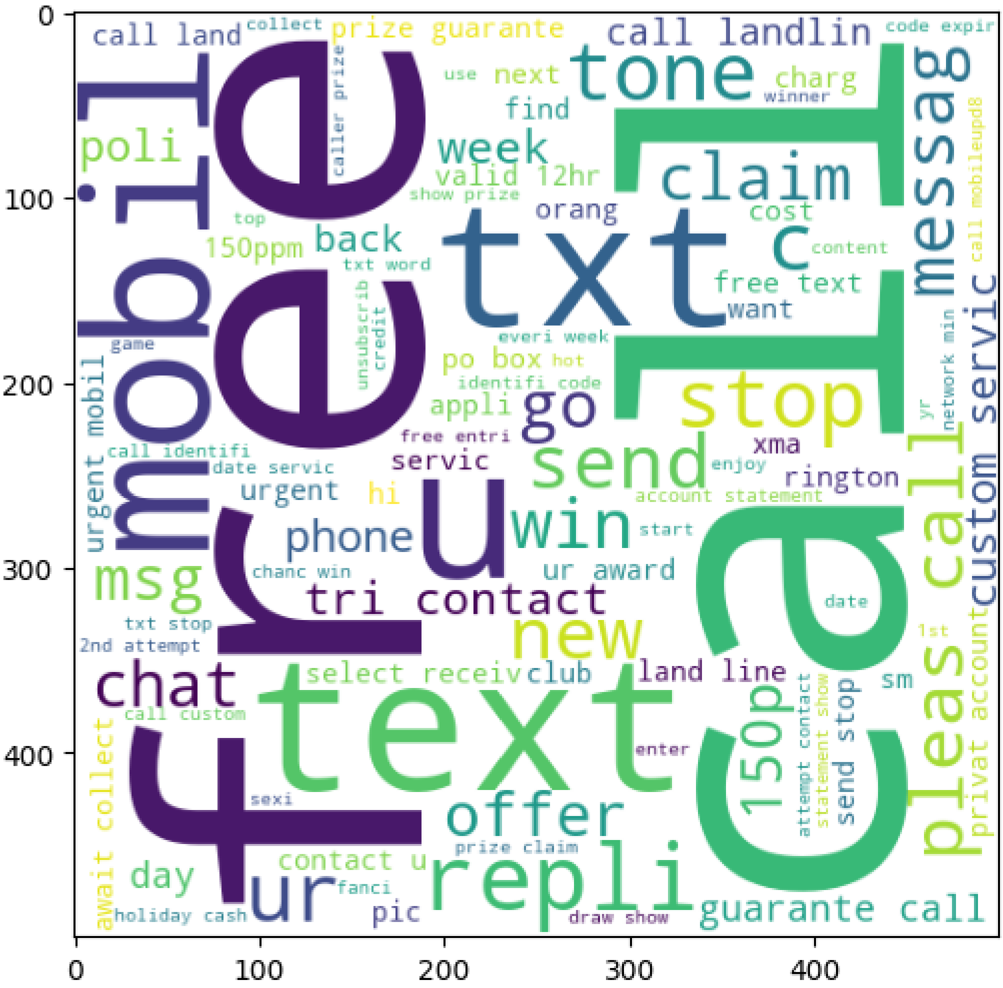
```
spam_wc = wc.generate(df[df['target'] == 1]['transformed_text'].str.cat(sep=" "))
```

In [43]:

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

Out[43]:

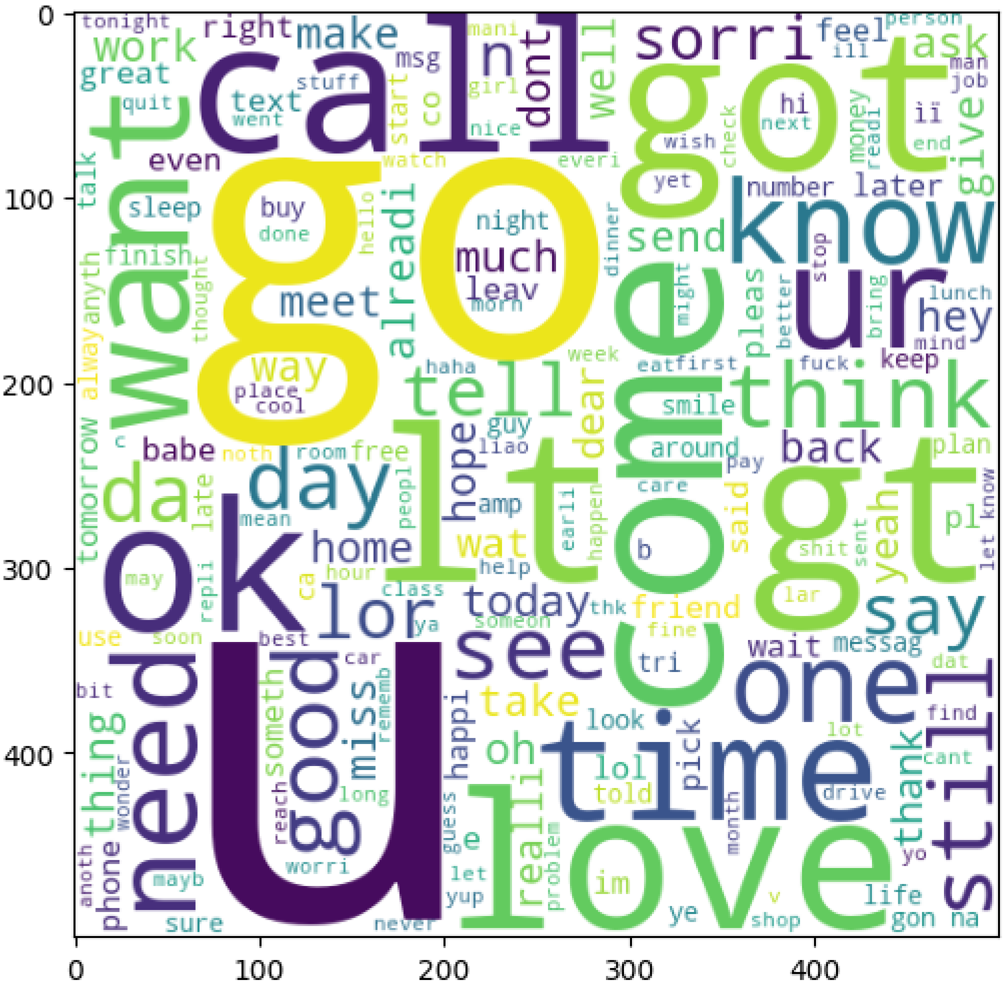
```
<matplotlib.image.AxesImage at 0x263a4469be0>
```



```
In [44]: ham_wc = wc.generate(df[df['target'] == 0]['transformed_text'].str.cat(sep=" "))
```

```
In [45]: plt.figure(figsize=(15,6))
         plt.imshow(ham_wc)
```

```
Out[45]: <matplotlib.image.AxesImage at 0x263a441d7f0>
```


0
100
200
300
400
0
100
200
300
400
u
go
call
got
want
ok
come
lt
gt
love
time
know
ur
need
day
good
think
tell
one
see
still
da
lor
say
sorri
work
great
right
make
text
send
much
meet
night
leav
alreadi
way
back
home
today
take
wait
oh
lol
friend
hope
dear
later
number
hey
give
ask
feel
well
dont
babe
free
smile
tomorrow
thing
phone
miss
happi
pick
realli
thank
yeah
life
shop
gon na

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

Out[46]:

	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 [47]:

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

In [48]:

```
len(spam_corpus)
```

Out[48]:

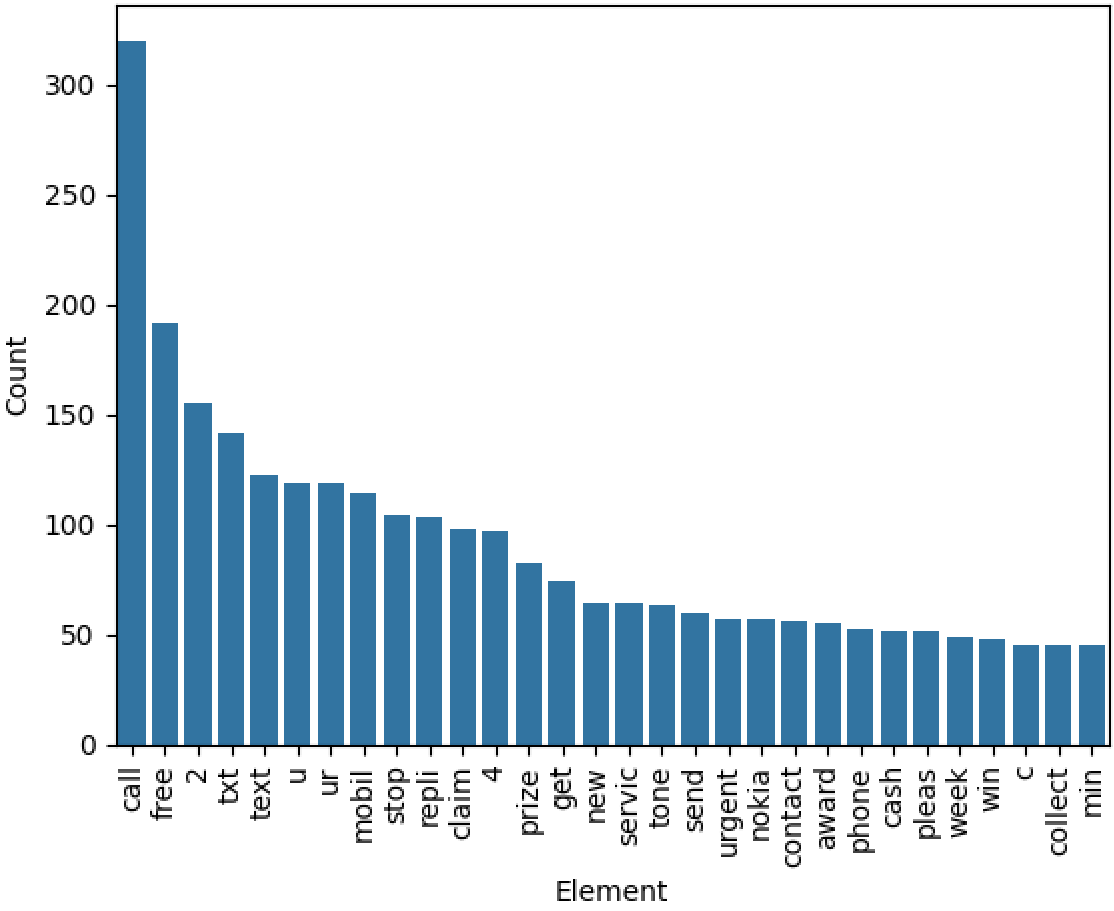
```
9939
```

In [49]:

```
from collections import Counter
counter = Counter(spam_corpus)
common_elements = counter.most_common(30)
df_common_elements = pd.DataFrame(common_elements, columns=['Element', 'Count'])
```

In [50]:

```
from collections import Counter
sns.barplot(x='Element', y='Count', data=df_common_elements)
plt.xticks(rotation='vertical')
plt.show()
```



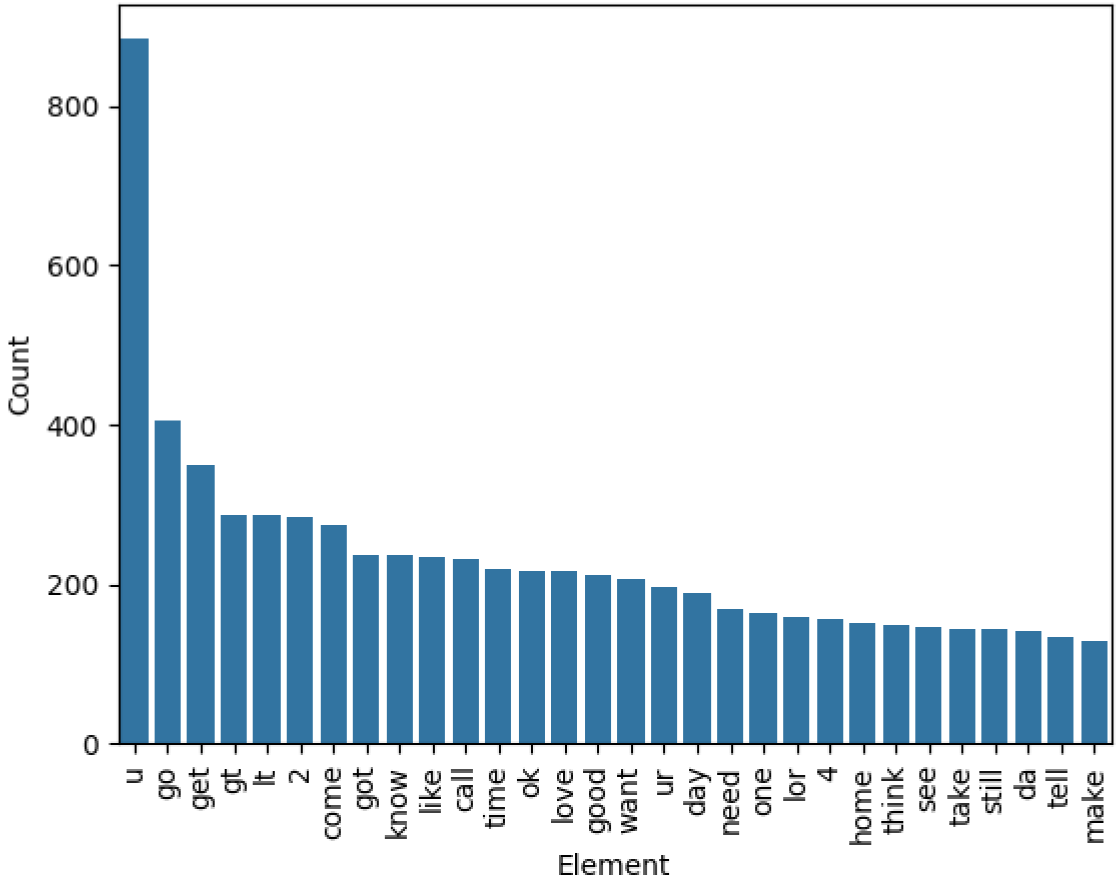
In [51]:
```
ham_corpus = []
for msg in df[df['target'] == 0]['transformed_text'].tolist():
    for word in msg.split():
        ham_corpus.append(word)
```

In [52]:
```
len(ham_corpus)
```

Out[52]:
```
35404
```

In [53]:
```
counter = Counter(ham_corpus)
common_elements = counter.most_common(30)
df_common_elements = pd.DataFrame(common_elements, columns=['Element', 'Count'])
```

In [54]:
```
sns.barplot(x='Element', y='Count', data=df_common_elements)
plt.xticks(rotation='vertical')
plt.show()
```


Count
800
600
400
200
0
u
go
get
gt
lt
2
come
got
know
like
call
time
ok
love
good
want
ur
day
need
one
lor
4
home
think
see
take
still
da
tell
make
Element

In [55]:
```
df.head()
```

Out[55]:

	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 [56]:

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

In [57]:

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

In [58]:

```
X.shape
```

Out[58]:

```
(5169, 3000)
```

In [59]:

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

In [60]:

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

In [61]:

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

In [62]:

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

In [63]:

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

In [64]:

```
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 [65]:

```
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 [66]:

```
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 [67]:

```
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 [68]:

```
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 [69]:
```
clfs = {
    'SVC' : svc,
    'KN' : knc,
    'NB': mnb,
    'DT': dtc,
    'LR': lrc,
    'RF': rfc,
    'AdaBoost': abc,
    'BgC': bc,
    'ETC': etc,
    'GBDT':gbdt,
    'xgb':xgb
}
```

In [70]:
```
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 [71]:
```
train_classifier(svc,X_train,y_train,X_test,y_test)
```

Out[71]:
```
(0.9758220502901354, 0.9747899159663865)
```

In [72]:
```
accuracy_scores = []
precision_scores = []

for name,clf in clfs.items():

    current_accuracy,current_precision = train_classifier(clf, X_train,y_train,X

    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 -  0.9323017408123792
Precision -  0.8333333333333334
For  LR
Accuracy -  0.9584139264990329
Precision -  0.9702970297029703
For  RF
Accuracy -  0.9758220502901354
Precision -  0.9829059829059829
```

```
C:\Users\Megha Sharma\AppData\Roaming\Python\Python39\site-packages\sklearn\ensem
ble\_weight_boosting.py:519: FutureWarning: The SAMME.R algorithm (the default) i
s deprecated and will be removed in 1.6. Use the SAMME algorithm to circumvent th
is warning.
  warnings.warn(
```

```
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
```

In [73]:

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

Out[73]:

	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	0.932302	0.833333

In [74]:

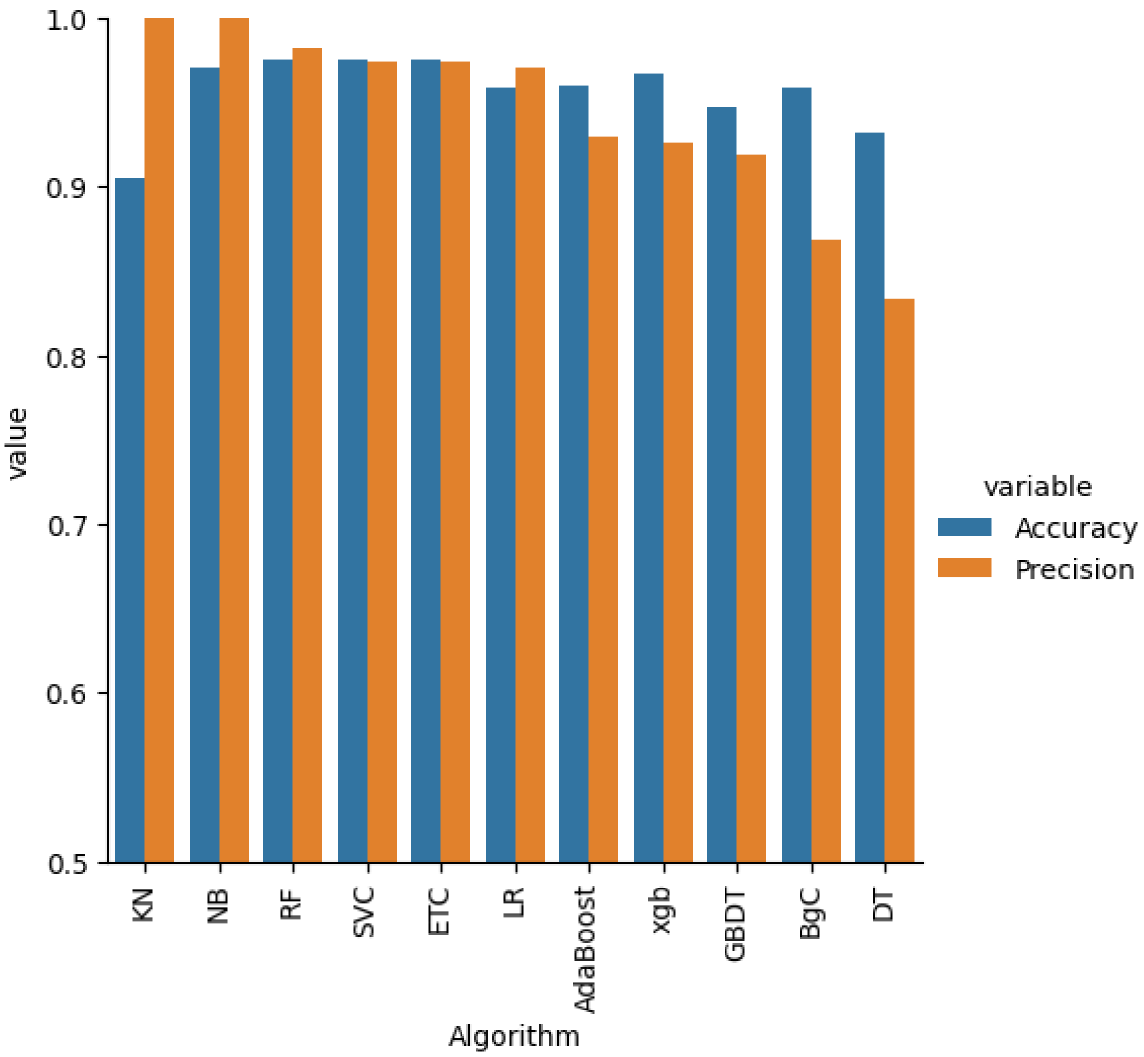
```
performance_df1 = pd.melt(performance_df, id_vars = "Algorithm")
performance_df1
```

Out[74]:

	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	0.932302
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	0.833333

In [75]:

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

```
In [77]: temp_df = pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy_scaling':accuracy_scor
```

```
In [78]: new_df = performance_df.merge(temp_df,on='Algorithm')
```

```
In [79]: new_df_scaled = new_df.merge(temp_df,on='Algorithm')
```

```
In [80]: temp_df = pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy_num_chars':accuracy_sc
```

```
In [81]: new_df_scaled.merge(temp_df,on='Algorithm')
```

Out[81]:

	Algorithm	Accuracy	Precision	Accuracy_scaling_x	Precision_scaling_x	Accuracy_sc
0	KN	0.905222	1.000000	0.905222	1.000000	0.
1	NB	0.970986	1.000000	0.970986	1.000000	0.
2	RF	0.975822	0.982906	0.975822	0.982906	0.
3	SVC	0.975822	0.974790	0.975822	0.974790	0.
4	ETC	0.974855	0.974576	0.974855	0.974576	0.
5	LR	0.958414	0.970297	0.958414	0.970297	0.
6	AdaBoost	0.960348	0.929204	0.960348	0.929204	0.
7	xgb	0.967118	0.926230	0.967118	0.926230	0.
8	GBDT	0.946809	0.919192	0.946809	0.919192	0.
9	BgC	0.958414	0.868217	0.958414	0.868217	0.
10	DT	0.932302	0.833333	0.932302	0.833333	0.

In [82]:

```
# 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
```

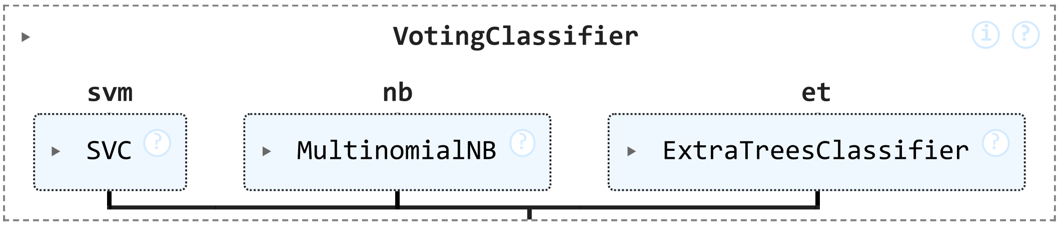
In [83]:

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

In [84]:

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

Out[84]:



In [85]:

```
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 [86]:

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

In [87]:

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

In [88]:

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

In [89]:

```
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 0.9806576402321083
Precision 0.9538461538461539
```

In [ ]:

In [ ]: