Project Title - SMS-Spam-Detection

Supervisor - Dr. K. Sri Phani Krishna

Members - Vikas Dhayal (521255) ,Mayank Saini (521160) ,Kamlesh Godara (521147)

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

In [2]:
```
df = pd.read_csv('spam.csv' ,encoding='ISO-8859-1')
df.head()
```

Out[2]:

	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 [3]:
```
df.sample(5)
```

Out[3]:

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
3498	ham	I hope you arnt pissed off but id would really...	NaN	NaN	NaN
1250	ham	Ummmmmaah Many many happy returns of d day my ...	NaN	NaN	NaN
2200	ham	Haha... can... But i'm having dinner with my c...	NaN	NaN	NaN
5053	spam	Double Mins & Double Txt & 1/2 price Linerenta...	NaN	NaN	NaN
654	ham	Did u got that persons story	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]:
```
# drop last 3 cols
cols_to_drop = ['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4']
for col in cols_to_drop:
    if col in df.columns:
        df.drop(columns=col, inplace=True)
```

In [7]:
```
df.sample(5)
```

Out[7]:

	v1	v2
1033	ham	OH MR SHEFFIELD! You wanna play THAT game, oka...
4277	ham	I've reached home finally...
3277	ham	What happened in interview?
5201	spam	YOU VE WON! Your 4* Costa Del Sol Holiday or å...
4649	ham	Finally it has happened..! Aftr decades..! BEE...

In [8]:
```
# renaming the cols
df.rename(columns={'v1':'target','v2':'text'},inplace=True)
df.sample(5)
```

Out[8]:

	target	text
4822	ham	:-) :-)
818	ham	Then why you not responding
3267	ham	Which is why i never wanted to tell you any of...
683	spam	Hi I'm sue. I am 20 years old and work as a la...
1860	ham	It could work, we'll reach a consensus at the ...

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

Out[12]:
```
target    0
text      0
dtype: int64
```

In [13]:
```
# check for duplicate values
df.duplicated().sum()
```

Out[13]:
```
403
```

In [14]:
```
# remove duplicates
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()
```

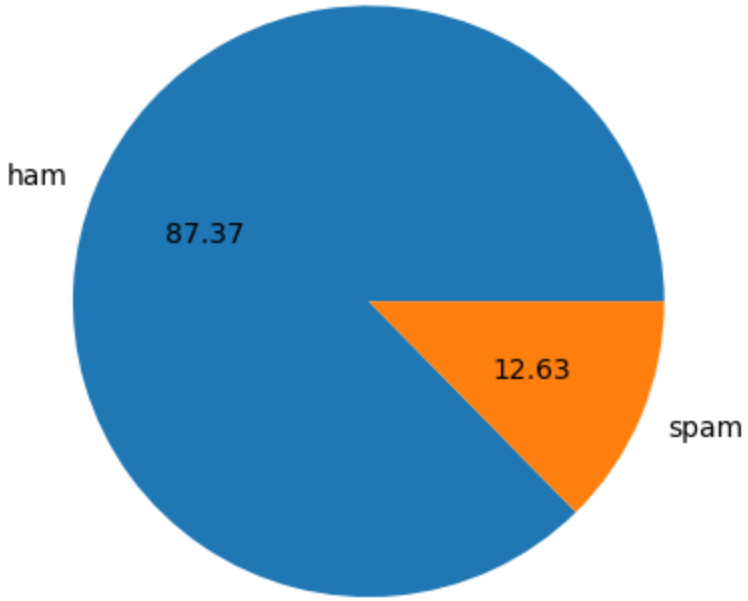
Out[17]:

	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]:
```
# Data is imbalanced
```

In [21]:
```
import nltk
```

In [22]:
```
!pip install nltk
```

```
Requirement already satisfied: nltk in c:\users\acer\anaconda\lib\site-packages (3.8.
1)
Requirement already satisfied: click in c:\users\acer\anaconda\lib\site-packages (from
nltk) (8.0.4)
Requirement already satisfied: joblib in c:\users\acer\anaconda\lib\site-packages (fro
m nltk) (1.2.0)
Requirement already satisfied: regex>=2021.8.3 in c:\users\acer\anaconda\lib\site-pack
ages (from nltk) (2022.7.9)
Requirement already satisfied: tqdm in c:\users\acer\anaconda\lib\site-packages (from
nltk) (4.65.0)
Requirement already satisfied: colorama in c:\users\acer\anaconda\lib\site-packages (f
rom click->nltk) (0.4.6)
```

In [23]:
```
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\ACER\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

Out[23]:
```
True
```

In [24]:
```
df['num_characters'] = df['text'].apply(len)
```

In [25]:
```
df.head()
```

Out[25]:

	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 [26]:
```
# num of words
df['num_words'] = df['text'].apply(lambda x:len(nltk.word_tokenize(x)))
```

In [27]:
```
df.head()
```

Out[27]:

	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 [28]:
```
df['num_sentences'] = df['text'].apply(lambda x:len(nltk.sent_tokenize(x)))
```

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

Out[29]:

	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 [30]:
```
df[['num_characters','num_words','num_sentences']].describe()
```

Out[30]:

	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 [31]:
```
# ham
df[df['target'] == 0][['num_characters','num_words','num_sentences']].describe()
```

Out[31]:

	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 [32]:
```
#spam
df[df['target'] == 1][['num_characters','num_words','num_sentences']].describe()
```

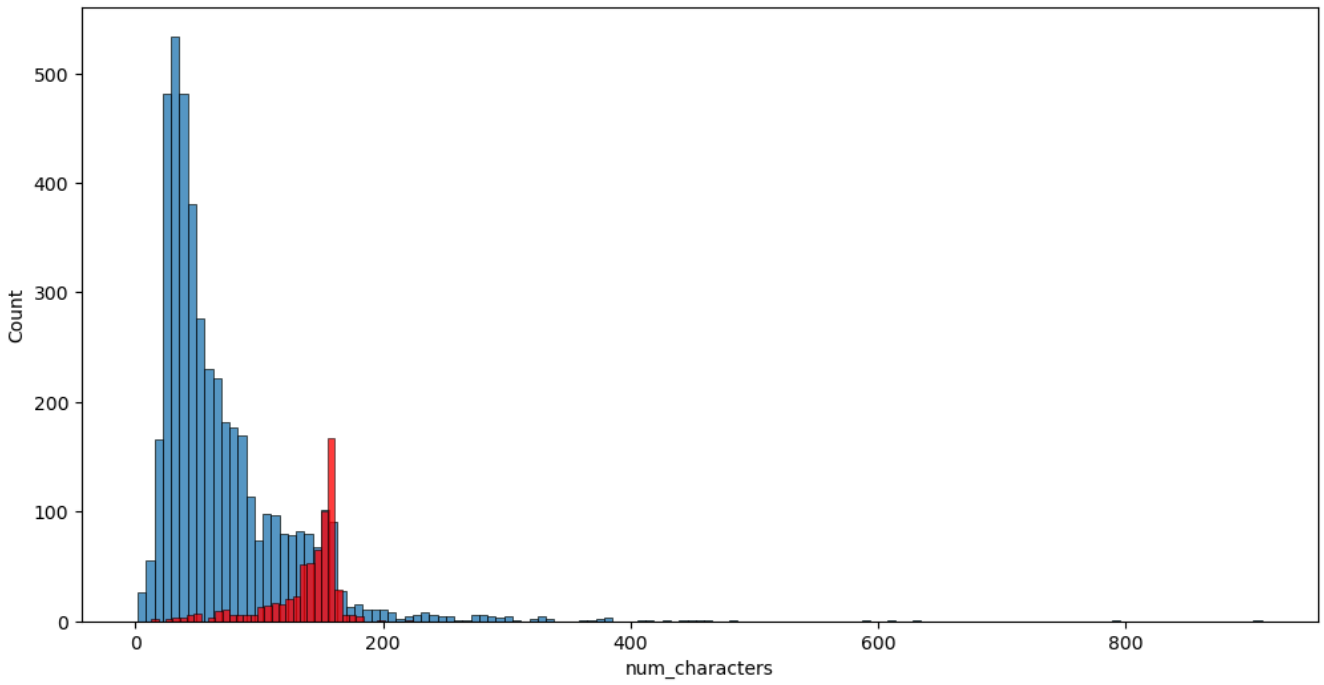
Out[32]:

	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 [33]:
```
import seaborn as sns
```

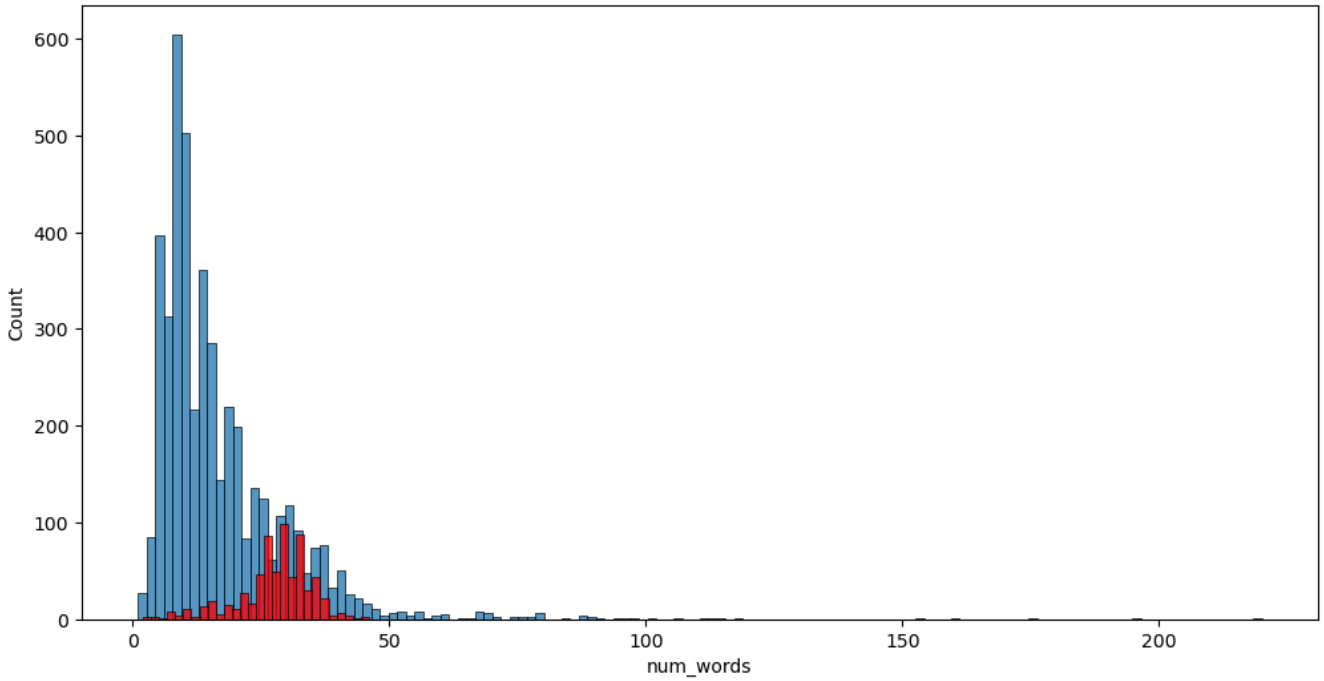
In [34]:
```
plt.figure(figsize=(12,6))
sns.histplot(df[df['target'] == 0]['num_characters'])
sns.histplot(df[df['target'] == 1]['num_characters'],color='red')
```

Out[34]: <Axes: xlabel='num_characters', ylabel='Count'>



In [35]:
```
plt.figure(figsize=(12,6))
sns.histplot(df[df['target'] == 0]['num_words'])
sns.histplot(df[df['target'] == 1]['num_words'],color='red')
```

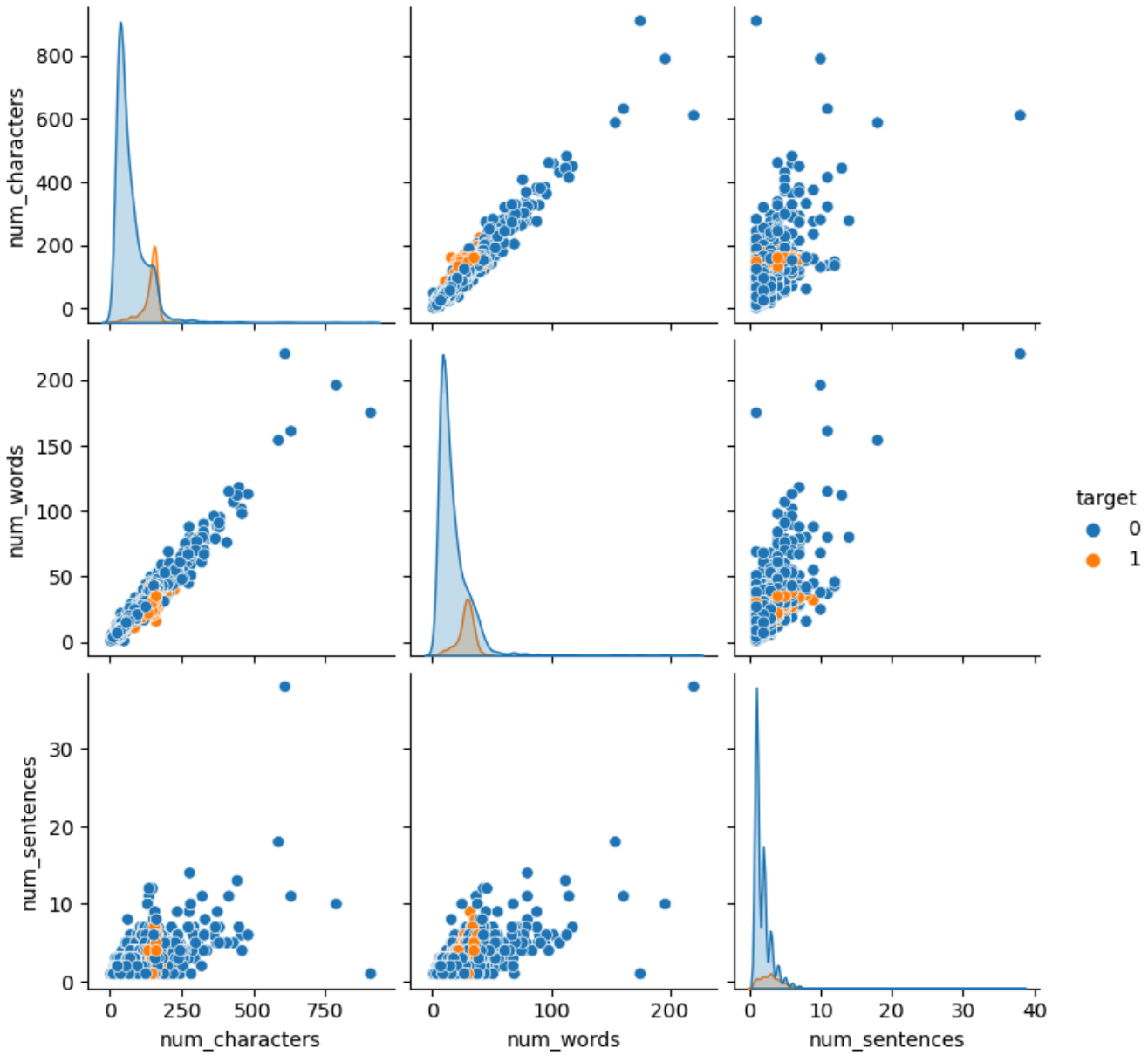
Out[35]: <Axes: xlabel='num_words', ylabel='Count'>



In [36]:
```
sns.pairplot(df,hue='target')
```

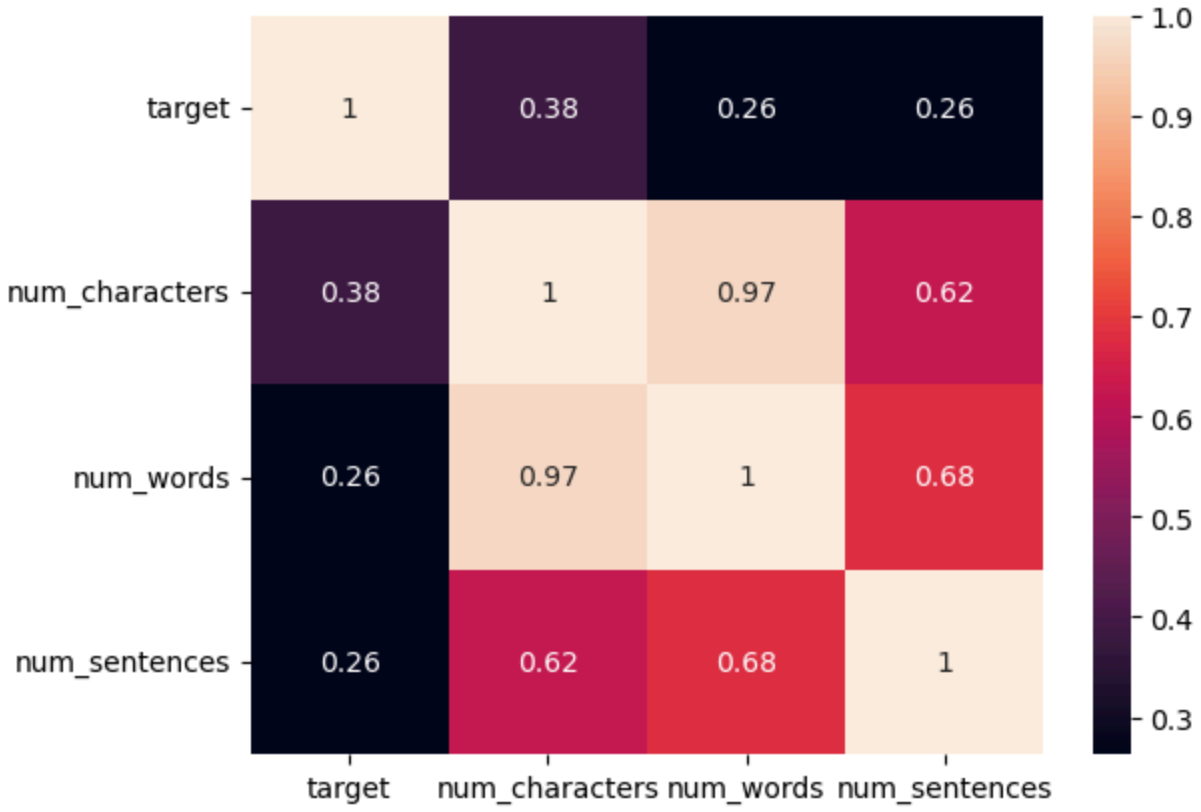
```
C:\Users\ACER\anaconda\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The fig
ure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

Out[36]:
```
<seaborn.axisgrid.PairGrid at 0x18511a8c8d0>
```



In [37]:
```
df_numeric = df.select_dtypes(include=[np.number])
sns.heatmap(df_numeric.corr(), annot=True)
```

Out[37]: <Axes: >



3. Data Preprocessing

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

In [38]:
```
import nltk
from nltk.corpus import stopwords
import string
from nltk.stem import PorterStemmer

ps = PorterStemmer()

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)
```

In [39]:
```
df['text'][10]
```

Out[39]:
```
"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 [40]:
```
transform_text("I'm gonna be home soon and i don't want to talk about this stuff anymore
```

Out[40]:
```
'gon na home soon want talk stuff anymor tonight k cri enough today'
```

In [41]:
```
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()
ps.stem('loving')
```

Out[41]:
```
'love'
```

In [42]:
```
df['transformed_text'] = df['text'].apply(transform_text)
```

In [43]:
```
df.head()
```

Out[43]:

	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 [44]:
```
!pip install wordcloud
```

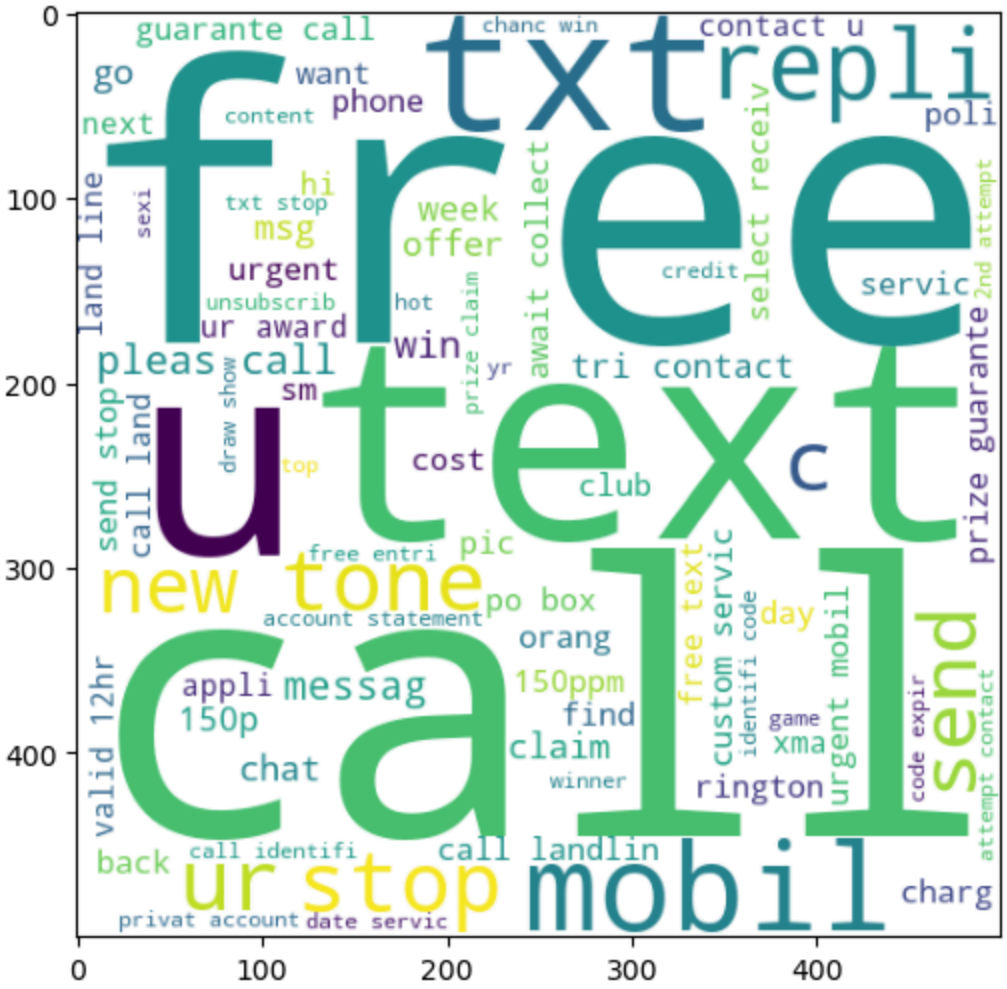
```
Requirement already satisfied: wordcloud in c:\users\acer\anaconda\lib\site-packages
(1.9.3)
Requirement already satisfied: numpy>=1.6.1 in c:\users\acer\anaconda\lib\site-package
s (from wordcloud) (1.24.3)
Requirement already satisfied: pillow in c:\users\acer\anaconda\lib\site-packages (fro
m wordcloud) (10.2.0)
Requirement already satisfied: matplotlib in c:\users\acer\anaconda\lib\site-packages
(from wordcloud) (3.7.2)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\acer\anaconda\lib\site-pac
kages (from matplotlib->wordcloud) (1.0.5)
Requirement already satisfied: cycler>=0.10 in c:\users\acer\anaconda\lib\site-package
s (from matplotlib->wordcloud) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\acer\anaconda\lib\site-pa
ckages (from matplotlib->wordcloud) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\acer\anaconda\lib\site-pa
ckages (from matplotlib->wordcloud) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\acer\anaconda\lib\site-pack
ages (from matplotlib->wordcloud) (23.1)
Requirement already satisfied: pyparsing<3.1,>=2.3.1 in c:\users\acer\anaconda\lib\sit
e-packages (from matplotlib->wordcloud) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\acer\anaconda\lib\site
-packages (from matplotlib->wordcloud) (2.8.2)
Requirement already satisfied: six>=1.5 in c:\users\acer\anaconda\lib\site-packages (f
rom python-dateutil>=2.7->matplotlib->wordcloud) (1.16.0)
```

In [45]:
```
from wordcloud import WordCloud
wc = WordCloud(width=500,height=500,min_font_size=10,background_color='white')
```

In [46]:
```
spam_wc = wc.generate(df[df['target'] == 1]['transformed_text'].str.cat(sep=" "))
```

In [47]:
```
plt.figure(figsize=(15,6))
plt.imshow(spam_wc)
```

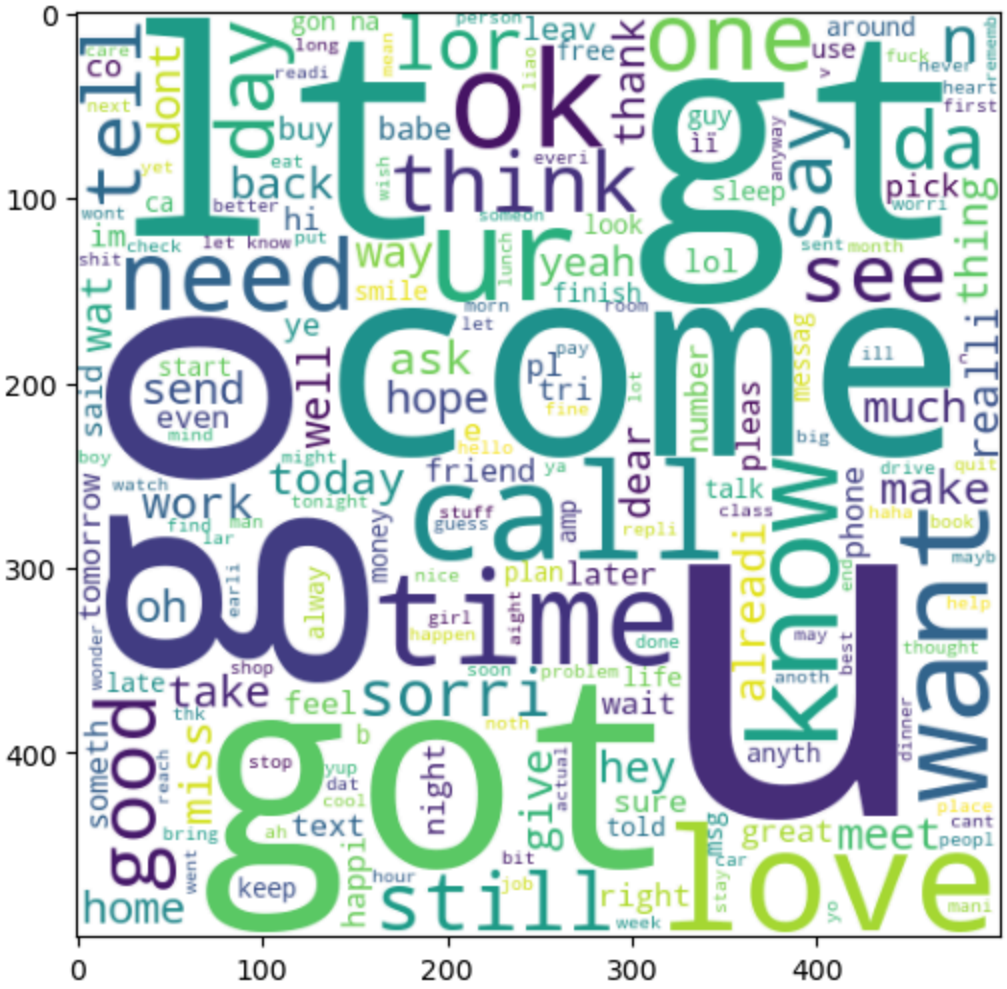
Out[47]: <matplotlib.figure.AxesImage at 0x185148ec150>



In [48]:
```
ham_wc = wc.generate(df[df['target'] == 0]['transformed_text'].str.cat(sep=" "))
```

In [49]:
```
plt.figure(figsize=(15,6))
plt.imshow(ham_wc)
```

Out[49]:
```
<matplotlib.image.AxesImage at 0x18514cfbe50>
```



In [50]:
```
df.head()
```

Out[50]:

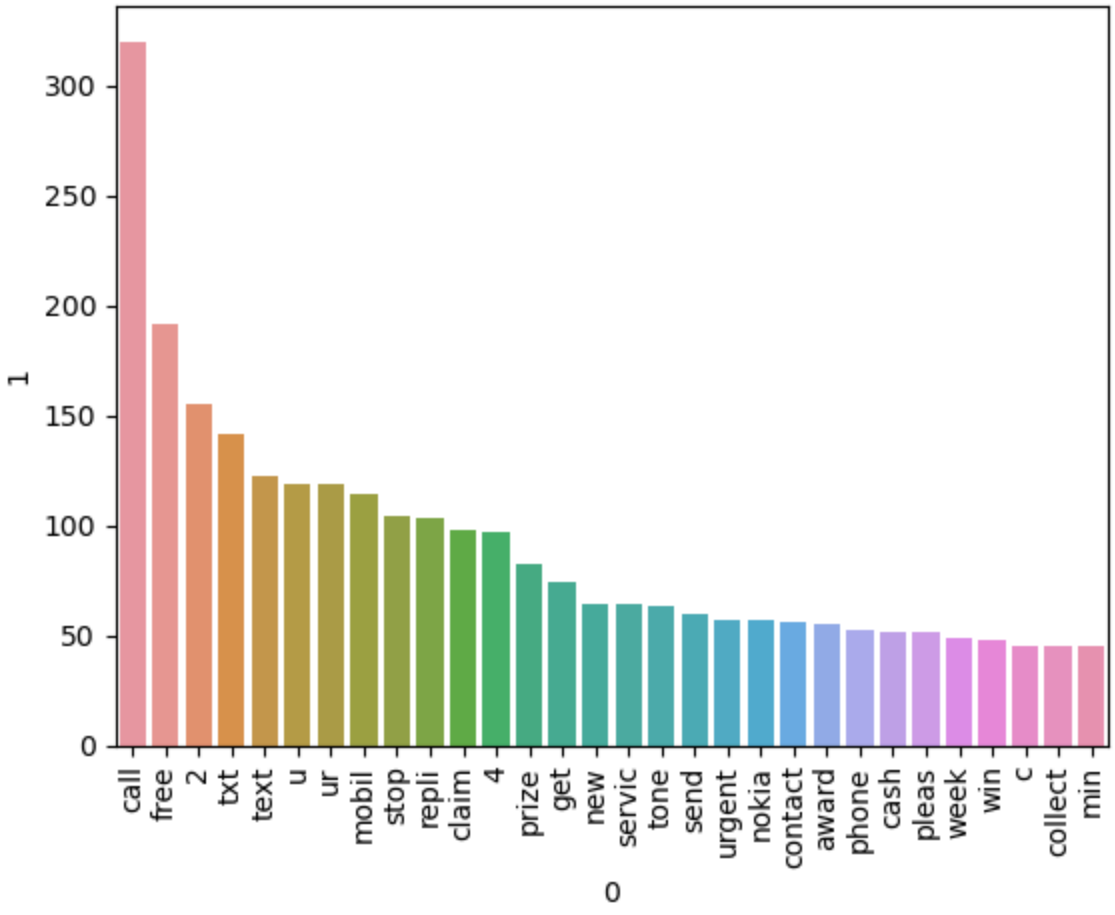
	target	text	num_characters	num_words	num_sentences	transformed_text
0	0	Go until jurong point, crazy.. Available only ...	111	24	2	go jurong point crazi avail bugi n great world...
1	0	Ok lar... Joking wif u oni...	29	8	2	ok lar joke wif u oni
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	155	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 [51]:
```
spam_corpus = []
for msg in df[df['target'] == 1]['transformed_text'].tolist():
    for word in msg.split():
        spam_corpus.append(word)
```

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

Out[52]: 9939

In [53]:
```
import seaborn as sns
from collections import Counter
import pandas as pd

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

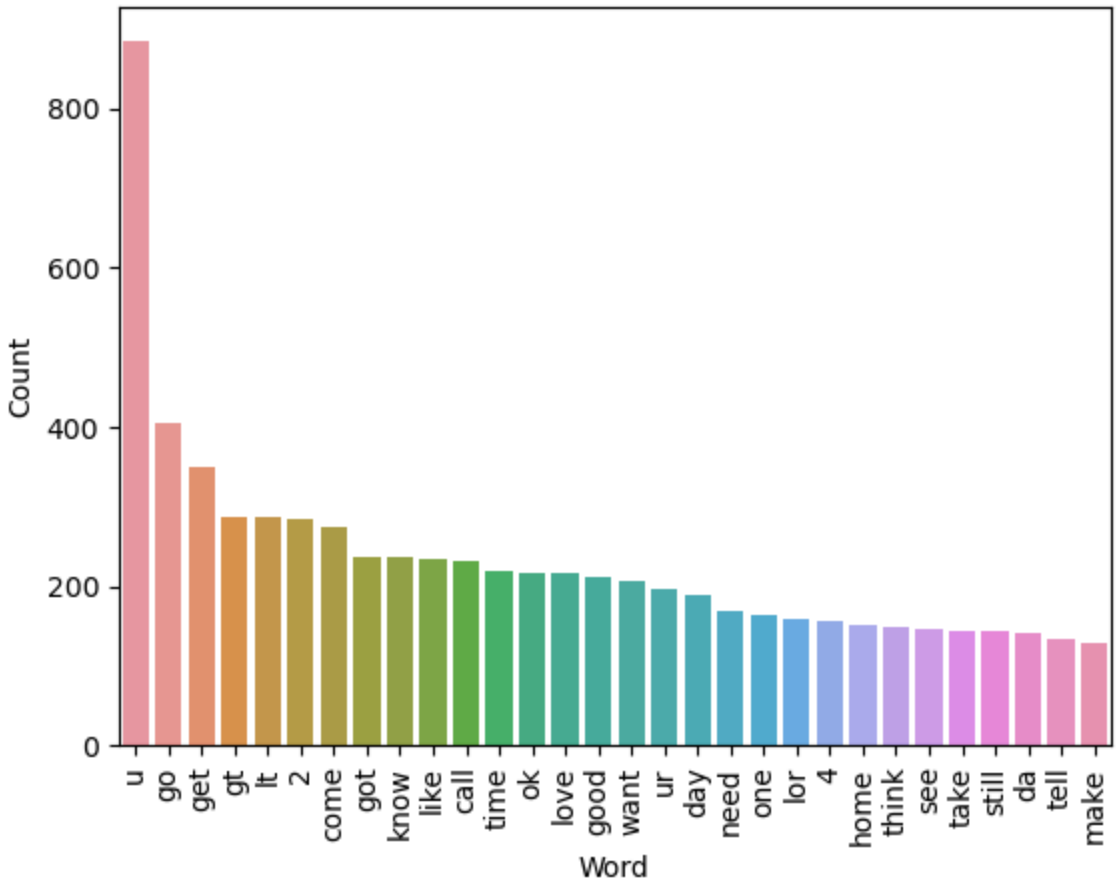


In [54]:
```
ham_corpus = []
for msg in df[df['target'] == 0]['transformed_text'].tolist():
    for word in msg.split():
        ham_corpus.append(word)
```

In [55]:
```
len(ham_corpus)
```

Out[55]: 35404

In [56]:
```
import pandas as pd
import seaborn as sns
from collections import Counter

# Assuming ham_corpus is a list of words
data = pd.DataFrame(Counter(ham_corpus).most_common(30), columns=['Word', 'Count'])

sns.barplot(x='Word', y='Count', data=data)
plt.xticks(rotation='vertical')
plt.show()
```



In [57]:
```
# Text Vectorization
# using Bag of Words
df.head()
```

Out[57]:

	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 [58]:
```
from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer
cv = CountVectorizer()
tfidf = TfidfVectorizer(max_features=3000)
```

In [59]:
```
X = tfidf.fit_transform(df['transformed_text']).toarray()
```

In [60]:
```
# from sklearn.preprocessing import MinMaxScaler
# scaler = MinMaxScaler()
# X = scaler.fit_transform(X)
```

In [61]:
```
# appending the num_character col to X
#X = np.hstack((X,df['num_characters'].values.reshape(-1,1)))
```

In [62]:
```
X.shape
```

Out[62]:
```
(5169, 3000)
```

In [63]:
```
y = df['target'].values
```

In [64]:
```
from sklearn.model_selection import train_test_split
```

In [65]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=2)
```

In [66]:
```
from sklearn.naive_bayes import GaussianNB,MultinomialNB,BernoulliNB
from sklearn.metrics import accuracy_score,confusion_matrix,precision_score
```

In [67]:
```
gnb = GaussianNB()
mnb = MultinomialNB()
bnb = BernoulliNB()
```

In [68]:
```
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 [69]:
```
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 [70]:
```
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 [71]:
```
# tfidf ---> mnb
```

In [72]:
```
!pip install xgboost
```

```
Requirement already satisfied: xgboost in c:\users\acer\anaconda\lib\site-packages (2.
0.3)
Requirement already satisfied: numpy in c:\users\acer\anaconda\lib\site-packages (from
xgboost) (1.24.3)
Requirement already satisfied: scipy in c:\users\acer\anaconda\lib\site-packages (from
xgboost) (1.11.1)
```

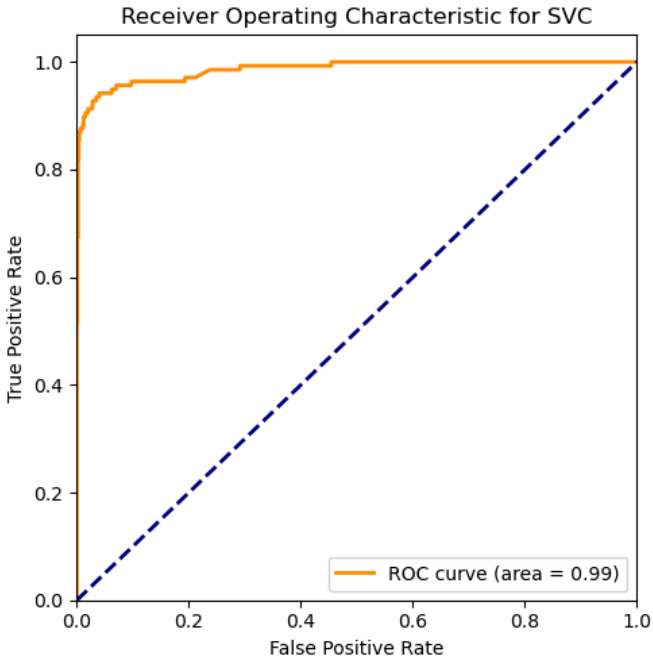
In [73]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.ensemble import GradientBoostingClassifier
from xgboost import XGBClassifier
```

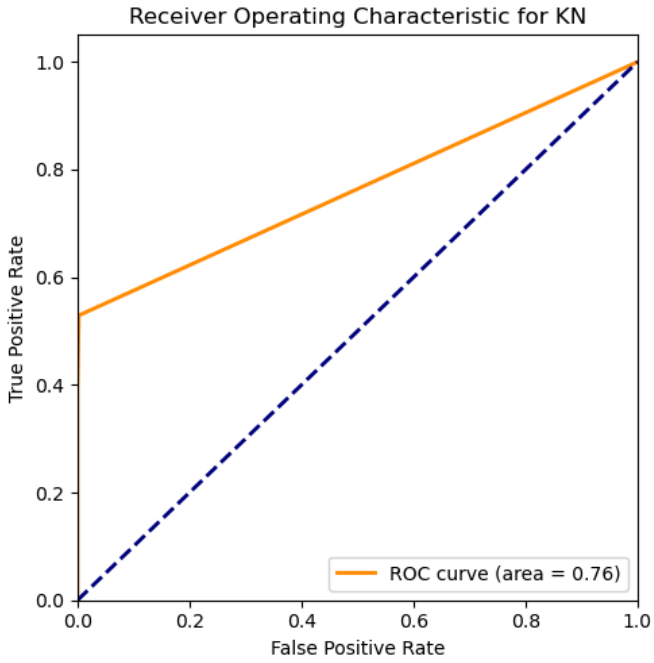
In [74]:
```
svc = SVC(kernel='sigmoid', gamma=1.0, probability=True)
knc = KNeighborsClassifier()
mnb = MultinomialNB()
dtc = DecisionTreeClassifier(max_depth=5)
lrc = LogisticRegression(solver='liblinear', penalty='l1')
rfc = RandomForestClassifier(n_estimators=50, random_state=2)
abc = AdaBoostClassifier(n_estimators=50, 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)
```

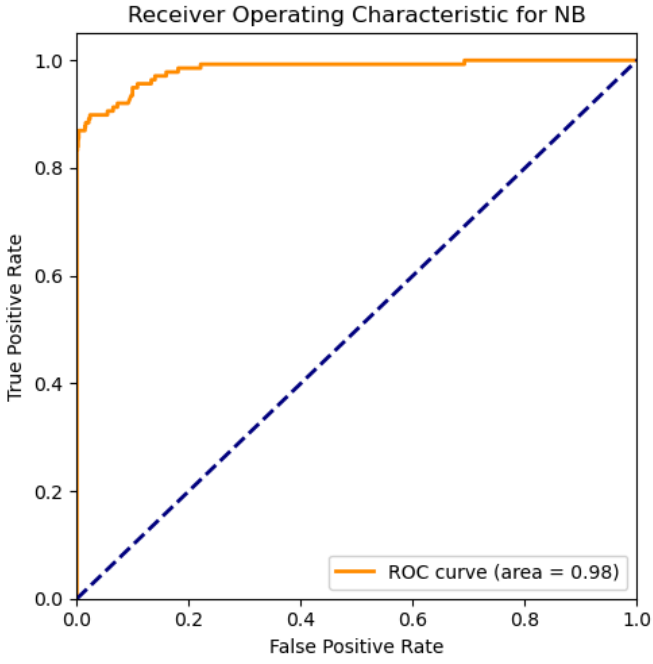
In [75]:
```
clfs = {
    'SVC' : svc,
    'KN' : knc,
    'NB': mnb,
    'DT': dtc,
    'LR': lrc,
    'RF': rfc,
    'AdaBoost': abc,
#     'BgC': bc,
    'ETC': etc,
    'GBDT':gbdt,
    'xgb':xgb
}
```

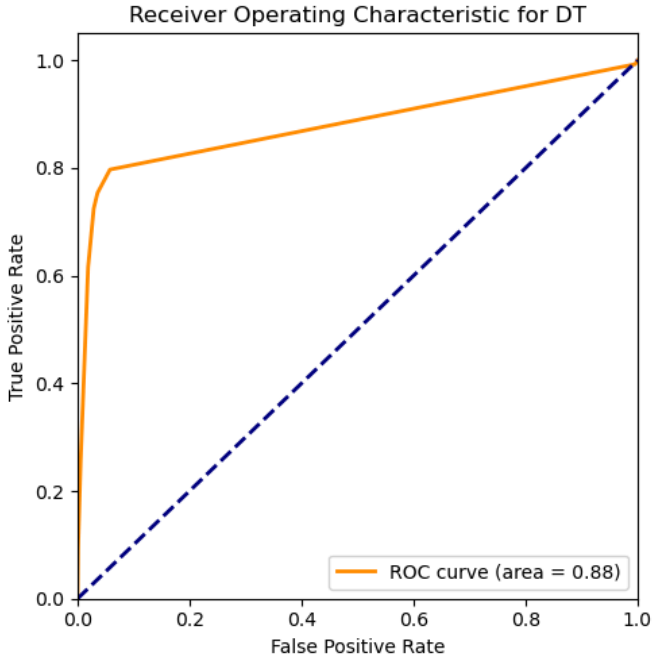
In [76]:
```
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt
import math

def train_classifier(clf, X_train, y_train, X_test, y_test):
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    y_pred_proba = clf.predict_proba(X_test)[:, 1]
    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred)
    fpr, tpr, _ = roc_curve(y_test, y_pred_proba)
    roc_auc = auc(fpr, tpr)

    return accuracy, precision, fpr, tpr, roc_auc

accuracy_scores = []
precision_scores = []
roc_curves = []

# Calculate the number of subplots for the grid
n = len(clfs)
ncols = 2
nrows = math.ceil(n / ncols)

fig, axs = plt.subplots(nrows, ncols, figsize=(10, 5*nrows))

for idx, (name, clf) in enumerate(clfs.items()):
    current_accuracy, current_precision, fpr, tpr, roc_auc = train_classifier(clf, X_tra
    print("For ", name)
    print("Accuracy - ", current_accuracy)
    print("Precision - ", current_precision)
    print("ROC AUC - ", roc_auc)
    accuracy_scores.append(current_accuracy)
    precision_scores.append(current_precision)
    roc_curves.append((fpr, tpr, roc_auc))

    # Plot ROC curve
    row = idx // ncols
    col = idx % ncols
    ax = axs[row, col]
    lw = 2
    ax.plot(fpr, tpr, color='darkorange', lw=lw, label='ROC curve (area = %0.2f)' % roc_
    ax.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
    ax.set_xlim([0.0, 1.0])
    ax.set_ylim([0.0, 1.05])
    ax.set_xlabel('False Positive Rate')
    ax.set_ylabel('True Positive Rate')
    ax.set_title('Receiver Operating Characteristic for ' + name)
    ax.legend(loc="lower right")

plt.tight_layout()
plt.show()
```

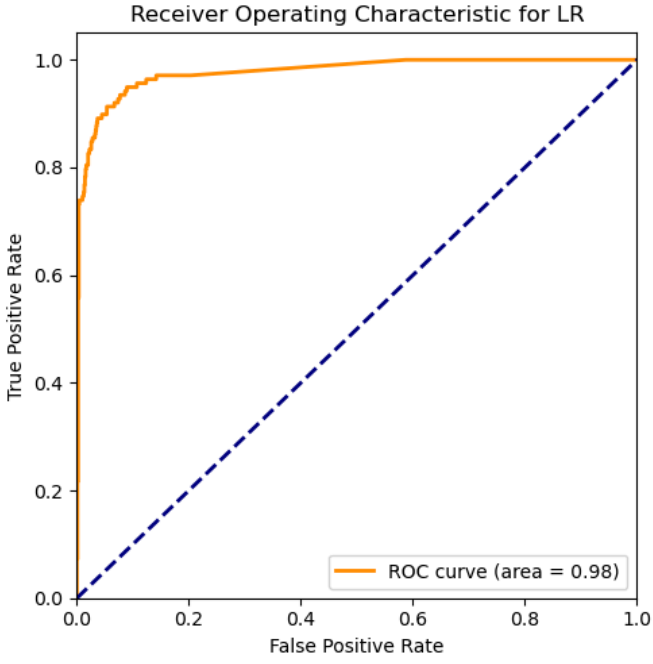
```
For  SVC
Accuracy -  0.9758220502901354
Precision -  0.9747899159663865
ROC AUC -  0.9860733695652174
For  KN
Accuracy -  0.9052224371373307
Precision -  1.0
ROC AUC -  0.7638134057971016
For  NB
Accuracy -  0.9709864603481625
Precision -  1.0
ROC AUC -  0.9833317158385094
For  DT
Accuracy -  0.9323017408123792
Precision -  0.8333333333333334
ROC AUC -  0.879092261904762
For  LR
Accuracy -  0.9584139264990329
Precision -  0.9702970297029703
ROC AUC -  0.9770234860248448
For  RF
Accuracy -  0.9758220502901354
Precision -  0.9829059829059829
ROC AUC -  0.9818436206004142
For  AdaBoost
Accuracy -  0.960348162475822
Precision -  0.9292035398230089
ROC AUC -  0.9668170936853001
For  ETC
Accuracy -  0.9748549323017408
Precision -  0.9745762711864406
ROC AUC -  0.9841121570910973
For  GBDT
Accuracy -  0.9468085106382979
Precision -  0.9191919191919192
ROC AUC -  0.9628461438923395
For  xgb
Accuracy -  0.9671179883945842
Precision -  0.9262295081967213
ROC AUC -  0.9841323757763975
```

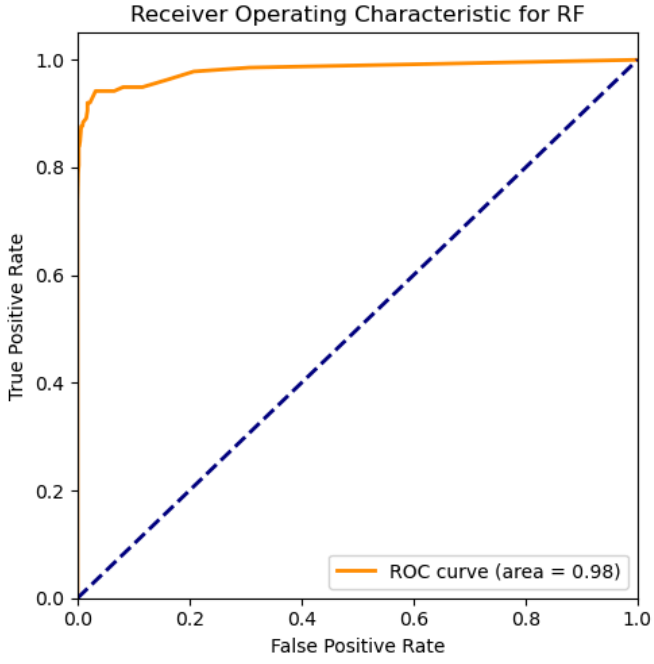






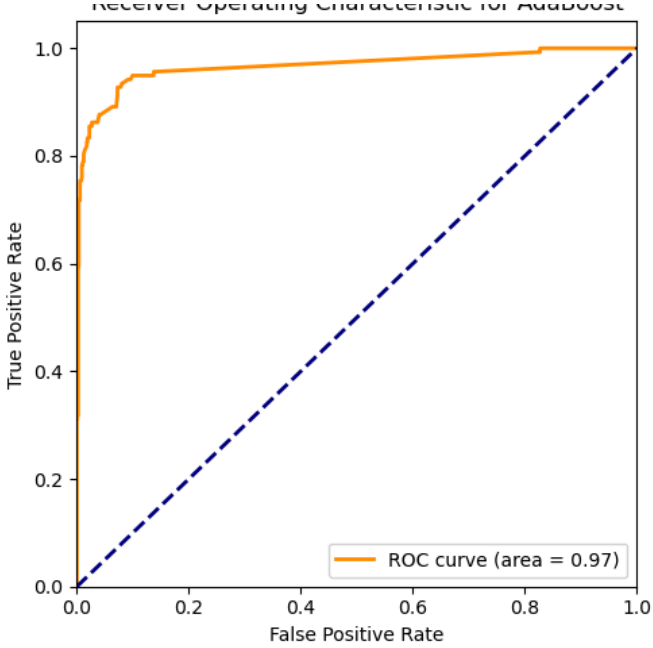


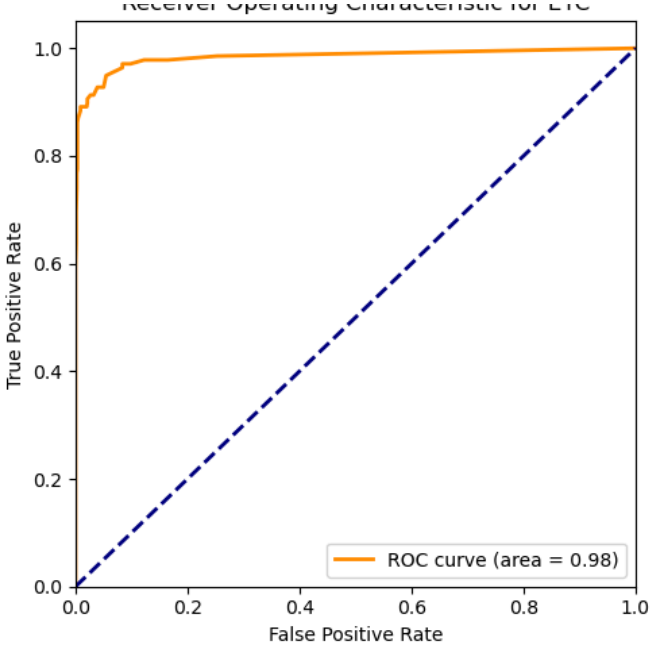


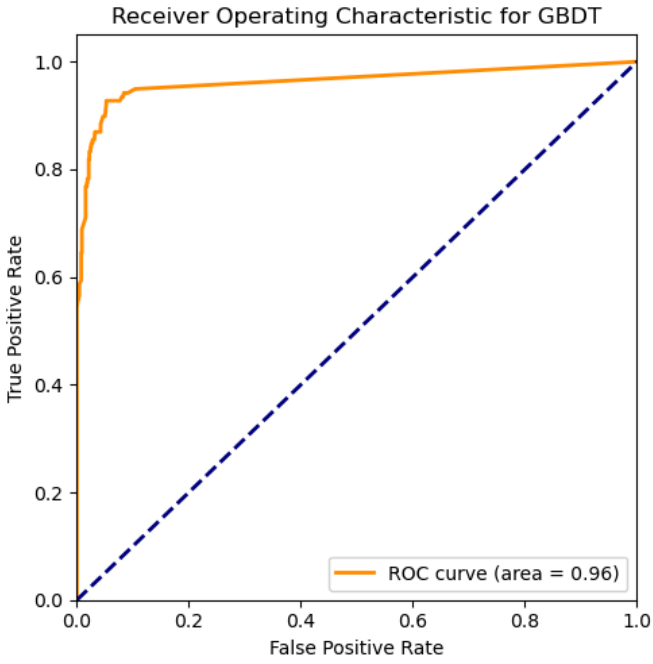


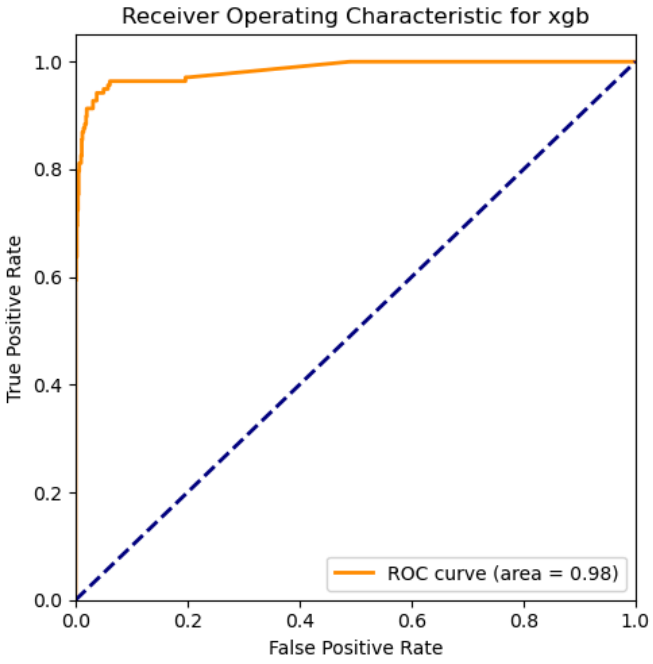
Receiver Operating Characteristic for AdaBoost

Receiver Operating Characteristic for ETC









In [77]:
```
performance_df = pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy':accuracy_scores,'Preci
```

In [78]:
```
performance_df
```

Out[78]:

	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
7	ETC	0.974855	0.974576
4	LR	0.958414	0.970297
6	AdaBoost	0.960348	0.929204
9	xgb	0.967118	0.926230
8	GBDT	0.946809	0.919192
3	DT	0.932302	0.833333

In [79]:
```
performance_df1 = pd.melt(performance_df, id_vars = "Algorithm")
```

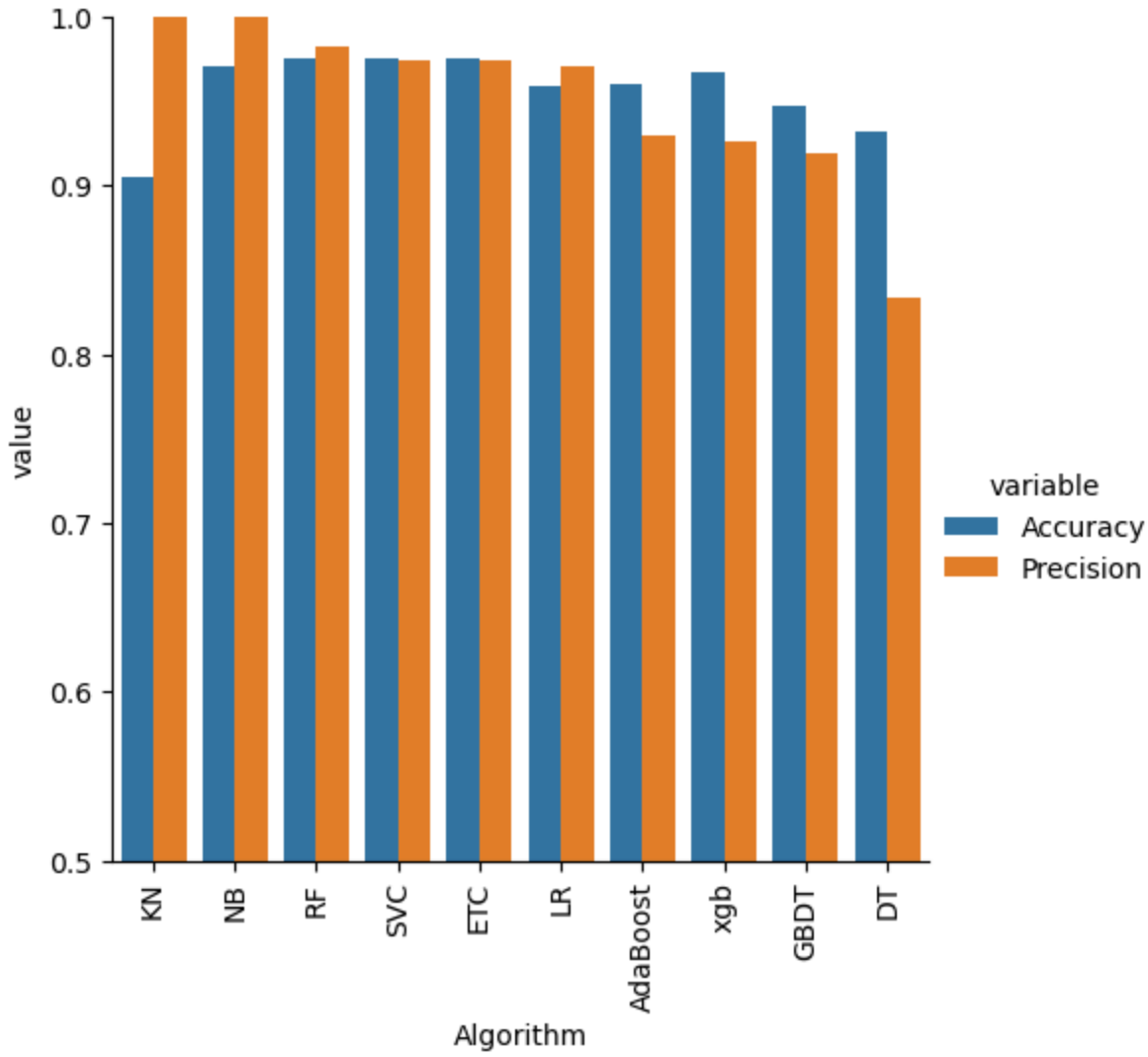
In [80]:
```
performance_df1
```

Out[80]:

	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	DT	Accuracy	0.932302
10	KN	Precision	1.000000
11	NB	Precision	1.000000
12	RF	Precision	0.982906
13	SVC	Precision	0.974790
14	ETC	Precision	0.974576
15	LR	Precision	0.970297
16	AdaBoost	Precision	0.929204
17	xgb	Precision	0.926230
18	GBDT	Precision	0.919192
19	DT	Precision	0.833333

In [81]:
```
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()
```

```
C:\Users\ACER\anaconda\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The fig
ure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```



In [82]:
```
# model improve
# 1. Change the max_features parameter of TfIdf
# 2. num_chars
# 3. Voting Classifier
# 4. Applying stacking
```

In [83]:
```
temp_df = pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy_max_ft_3000':accuracy_scores,'
```

In [84]:
```
temp_df = pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy_scaling':accuracy_scores,'Prec
```

In [85]:
```
new_df = performance_df.merge(temp_df,on='Algorithm')
```

In [86]:
```
new_df_scaled = new_df.merge(temp_df,on='Algorithm')
```

In [87]:
```
temp_df = pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy_num_chars':accuracy_scores,'Pr
```

In [88]:
```
new_df_scaled.merge(temp_df,on='Algorithm')
```

Out[88]:

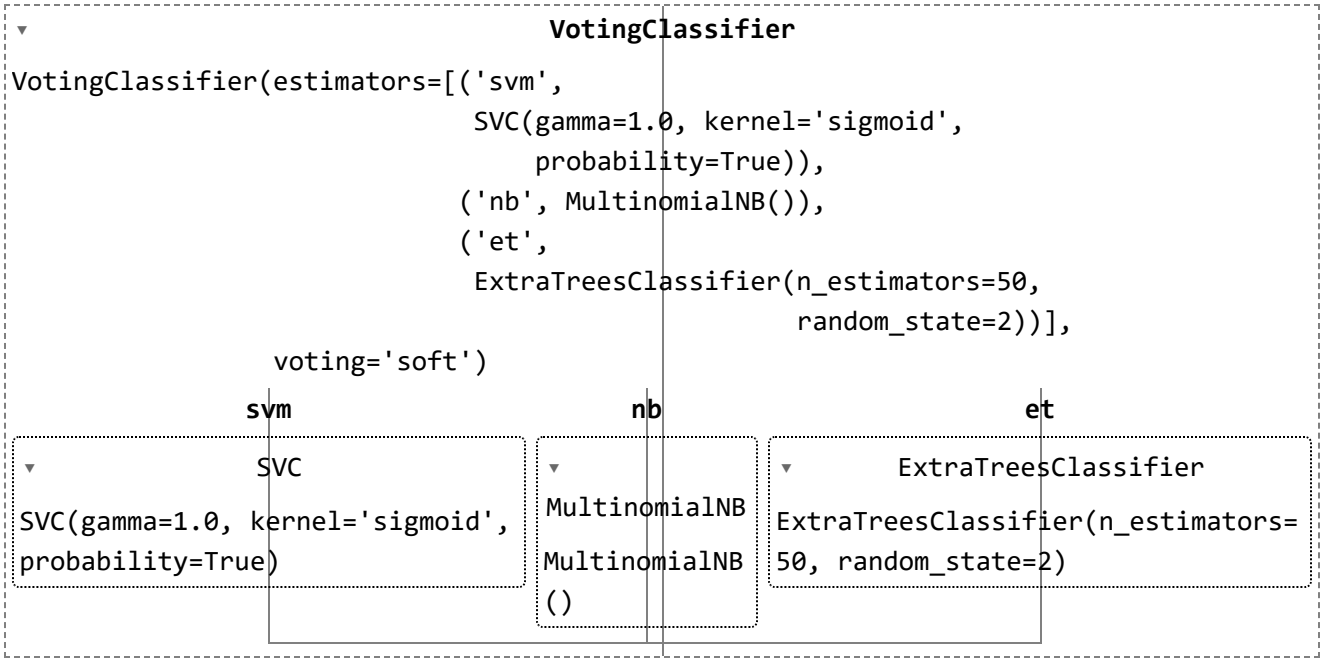
	Algorithm	Accuracy	Precision	Accuracy_scaling_x	Precision_scaling_x	Accuracy_scaling_y	Precision_sca
0	KN	0.905222	1.000000	0.905222	1.000000	0.905222	1.0
1	NB	0.970986	1.000000	0.970986	1.000000	0.970986	1.0
2	RF	0.975822	0.982906	0.975822	0.982906	0.975822	0.9
3	SVC	0.975822	0.974790	0.975822	0.974790	0.975822	0.9
4	ETC	0.974855	0.974576	0.974855	0.974576	0.974855	0.9
5	LR	0.958414	0.970297	0.958414	0.970297	0.958414	0.9
6	AdaBoost	0.960348	0.929204	0.960348	0.929204	0.960348	0.9
7	xgb	0.967118	0.926230	0.967118	0.926230	0.967118	0.9
8	GBDT	0.946809	0.919192	0.946809	0.919192	0.946809	0.9
9	DT	0.932302	0.833333	0.932302	0.833333	0.932302	0.8

In [89]:
```
# 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 [90]:
```
voting = VotingClassifier(estimators=[('svm', svc), ('nb', mnb), ('et', etc)],voting='so
```

In [91]:
```
voting.fit(X_train,y_train)
```

Out[91]:



In [92]:
```
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 [93]:
```
# Applying stacking
estimators=[('svm', svc), ('nb', mnb), ('et', etc)]
final_estimator=RandomForestClassifier()
```

In [94]:
```
from sklearn.ensemble import StackingClassifier
```

In [95]:
```
clf = StackingClassifier(estimators=estimators, final_estimator=final_estimator)
```

In [96]:
```
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
```