In [435…
```
!pip install scikit-learn
```

```
Requirement already satisfied: scikit-learn in c:\users\dell\appdata\local\pr
ograms\python\python311\lib\site-packages (1.3.2)
Requirement already satisfied: numpy<2.0,>=1.17.3 in c:\users\dell\appdata\lo
cal\programs\python\python311\lib\site-packages (from scikit-learn) (1.24.3)
Requirement already satisfied: scipy>=1.5.0 in c:\users\dell\appdata\local\pr
ograms\python\python311\lib\site-packages (from scikit-learn) (1.11.2)
Requirement already satisfied: joblib>=1.1.1 in c:\users\dell\appdata\local\p
rograms\python\python311\lib\site-packages (from scikit-learn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\dell\appdata
\local\programs\python\python311\lib\site-packages (from scikit-learn) (3.2.
0)
```

In [436…
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
```

In [437…
```
df = pd.read_csv('spam.csv')
```

In [438…
```
df.head()
```

Out[438]:

	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 [439…
```
df.shape
```

Out[439]: (5572, 5)

In [440…
```
#Task
#1.Data cleaning
#2.EDA
#3.Text Preprocessing
#4.Model building
#5.Evaluation
#6.Improvement
#7.Website
#8.Deploy
```

1.Data Cleaning

In [441…
```
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 [442…
```
#dropping columns which are not in use
df.drop(columns=['Unnamed: 2','Unnamed: 3','Unnamed: 4'],inplace=True)
```

In [443…
```
df.head()
```

Out[443]:

	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 [444…
```
#renaming the column name
df.rename(columns={'v1':'target','v2':'message'},inplace=True)
```

In [445…
```
df.head()
```

Out[445]:

	target	message
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 [446…
```
#Label Encoding is a technique that is used to convert categorical columns i
#so that they can be fitted by machine learning models which only take numer
from sklearn.preprocessing import LabelEncoder
encoder=LabelEncoder()
```

In [447…
```
encoder.fit_transform(df['target'])
```

Out[447]:
```
array([0, 0, 1, ..., 0, 0, 0])
```

In [448…
```
df['target']=encoder.fit_transform(df['target'])
```

In [449…
```
df.head()
```

Out[449]:

	target	message
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 [450…
```
#missing values
df.isnull().sum()
```

Out[450]:
```
target     0
message    0
dtype: int64
```

In [451…
```
df.duplicated().sum()
```

Out[451]:
```
403
```

In [452…
```
df=df.drop_duplicates(keep='first')
```

In [453…
```
df.duplicated().sum() #after deleting duplicate values
```

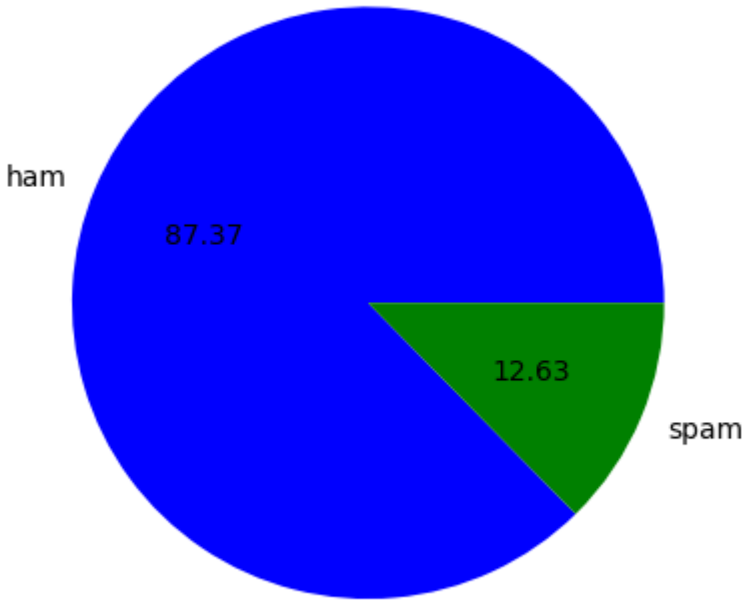
Out[453]:
```
0
```

2.EDA

In [454…
```
df['target'].value_counts()
```

Out[454]:
```
target
0    4516
1     653
Name: count, dtype: int64
```

In [455…
```
plt.pie(df['target'].value_counts(),labels=['ham','spam'],colors=['blue','gr
```



In [456…
```
# here we can see that spam messages are more in number than ham
```

In [457…
```
pip install nltk
```

```
Requirement already satisfied: nltk in c:\users\dell\appdata\local\programs\p
ython\python311\lib\site-packages (3.8.1)
Requirement already satisfied: click in c:\users\dell\appdata\local\programs
\python\python311\lib\site-packages (from nltk) (8.1.7)
Requirement already satisfied: joblib in c:\users\dell\appdata\local\programs
\python\python311\lib\site-packages (from nltk) (1.3.2)
Requirement already satisfied: regex>=2021.8.3 in c:\users\dell\appdata\local
\programs\python\python311\lib\site-packages (from nltk) (2023.10.3)
Requirement already satisfied: tqdm in c:\users\dell\appdata\local\programs\p
ython\python311\lib\site-packages (from nltk) (4.66.1)
Requirement already satisfied: colorama in c:\users\dell\appdata\local\progra
ms\python\python311\lib\site-packages (from click->nltk) (0.4.6)
Note: you may need to restart the kernel to use updated packages.
```

In [458…
```
import nltk
```

In [459…
```
nltk.download('punkt')
```

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

Out[459]: True

In [460…
```
df['num_characters']=df['message'].apply(len)
```

In [461…
```
df.head()
```

Out[461]:

	target	message	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 [462…
```
df.head()
```

Out[462]:

	target	message	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 [463…
```
df['num_sentences']=df['message'].apply(lambda x:len(nltk.word_tokenize(x)))
```

In [464…
```
df.head()
```

Out[464]:

	target	message	num_characters	num_sentences
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 [465…
```
df['num_words']=df['message'].apply(lambda x:len(nltk.sent_tokenize(x)))
```

In [466…
```
df.head()
```

Out[466]:

	target	message	num_characters	num_sentences	num_words
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 [467…
```
df[['num_characters','num_words','num_sentences']].describe()
```

Out[467]:

	num_characters	num_words	num_sentences
count	5169.000000	5169.000000	5169.000000
mean	78.977945	1.965564	18.455794
std	58.236293	1.448541	13.324758
min	2.000000	1.000000	1.000000
25%	36.000000	1.000000	9.000000
50%	60.000000	1.000000	15.000000
75%	117.000000	2.000000	26.000000
max	910.000000	38.000000	220.000000

In [468…
```
df[df['target']==0][['num_characters','num_words','num_sentences']].describe
```

Out[468]:

	num_characters	num_words	num_sentences
count	4516.000000	4516.000000	4516.000000
mean	70.459256	1.820195	17.123782
std	56.358207	1.383657	13.493970
min	2.000000	1.000000	1.000000
25%	34.000000	1.000000	8.000000
50%	52.000000	1.000000	13.000000
75%	90.000000	2.000000	22.000000
max	910.000000	38.000000	220.000000

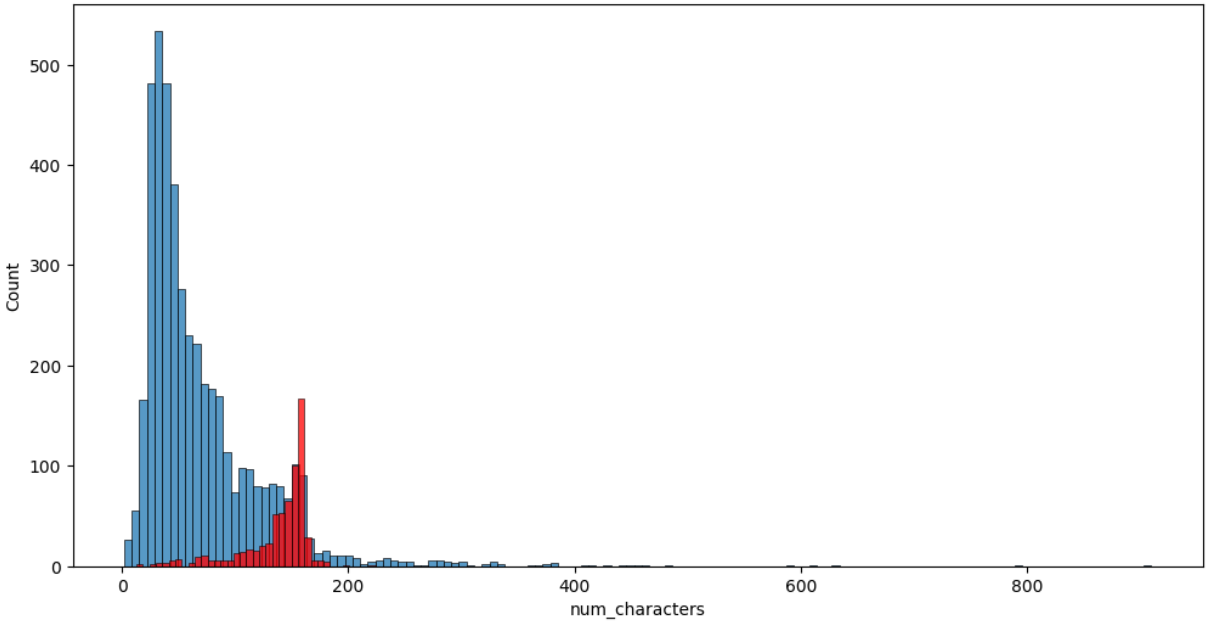
In [469…
```
df[df['target']==1][['num_characters','num_words','num_sentences']].describe
```

Out[469]:

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

In [470…
```
plt.figure(figsize=(12,6))
sns.histplot(df[df['target']==0]['num_characters'])
sns.histplot(df[df['target']==1]['num_characters'],color='red')
```

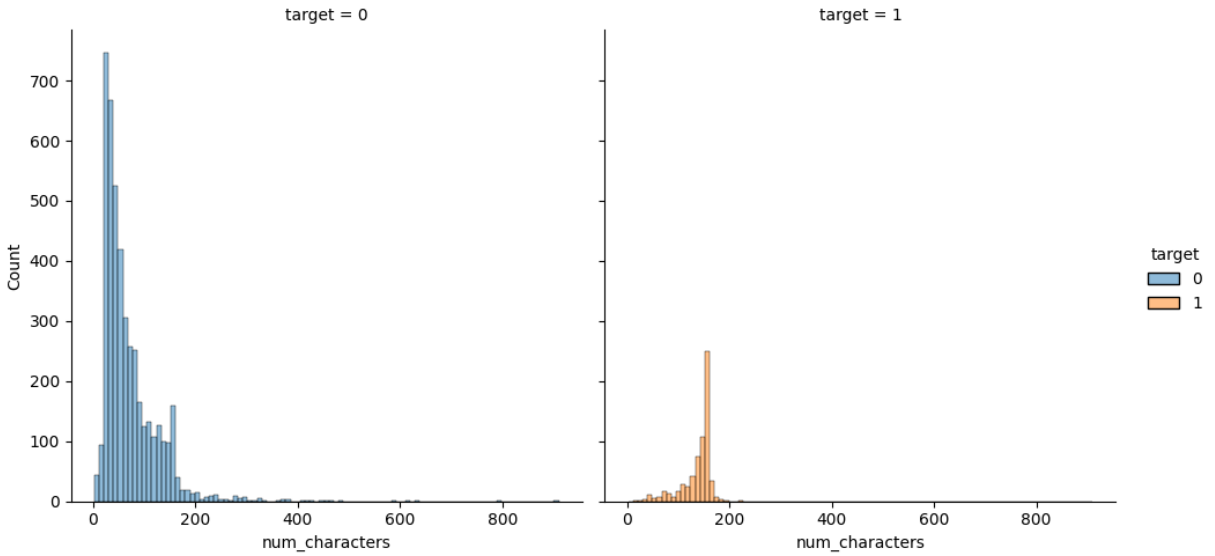
Out[470]: <Axes: xlabel='num_characters', ylabel='Count'>



In [471…
```
#here through histogram plot we can see that num of characters in spam messa
```

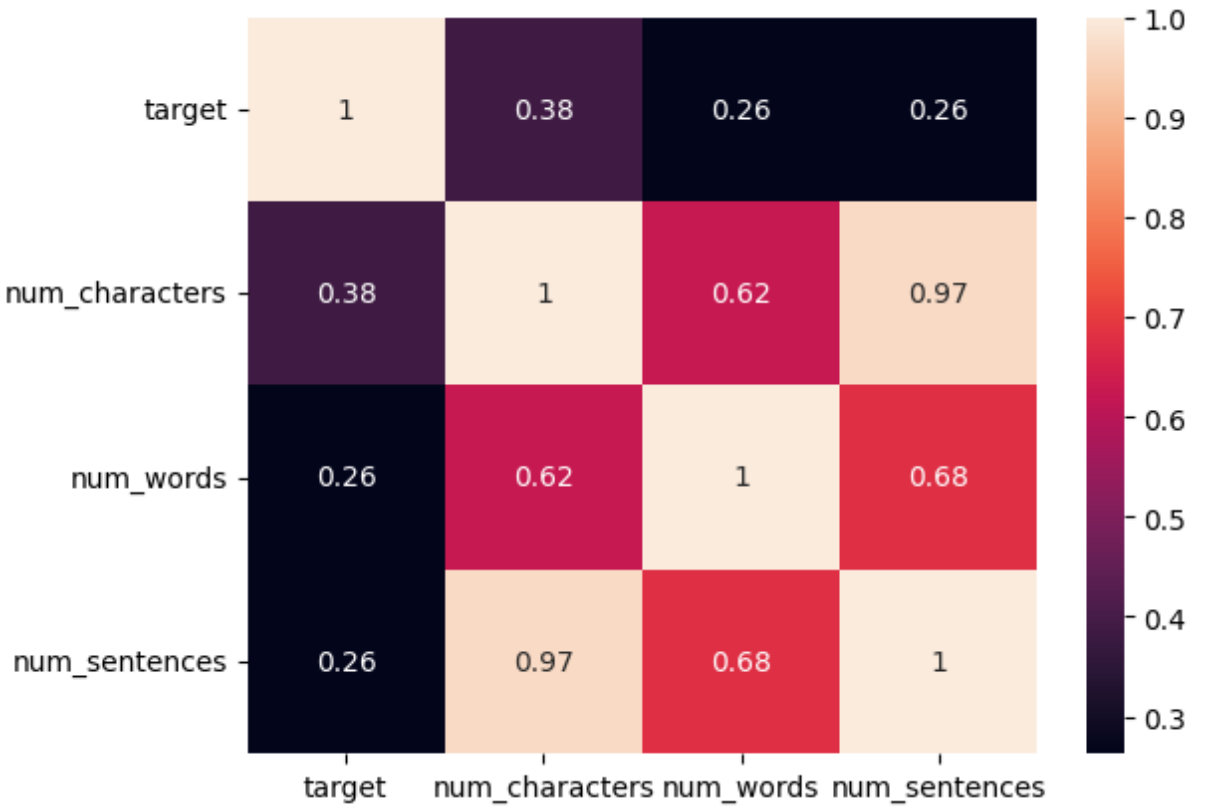
In [472…
```
sns.displot(data=df, x="num_characters", hue="target", col="target")
```

Out[472]: <seaborn.axisgrid.FacetGrid at 0x1748f7ab110>



In [473…
```
sns.heatmap(df[['target','num_characters','num_words','num_sentences']].corr
```

Out[473]: <Axes: >



In [474…
```
df.head()
```

Out[474]:

	target	message	num_characters	num_sentences	num_words
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

3.Data Preprocessing

In [475…

```
#lower case
#Tokenization
#Removing special characters
#Removing stop words and punctuation
#stemming
```

In [476…

```
import string
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()
```

In [477…

```
def transform_text(message):
    message = message.lower()
    message = nltk.word_tokenize(message)
    y = []
    for i in message:
        if i.isalnum():
            y.append(i)
    message = y[:]
    y.clear()
    for i in message:
        if i not in stopwords.words('english') and i not in string.punctuati
            y.append(i)
    message = y[:]
    y.clear()
    for i in message:
        y.append(ps.stem(i))


    return " ".join(y)
```

In [478…

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

Out[478]:

	target	message	num_characters	num_sentences	num_words	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 [479…

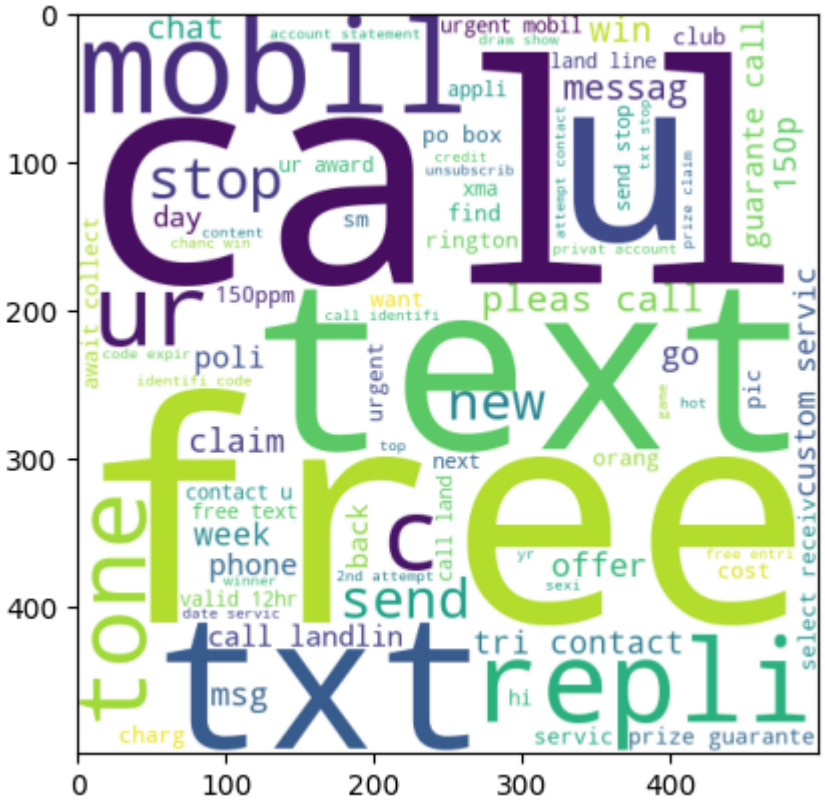
```
from wordcloud import WordCloud
wc = WordCloud(width=500,height=500,min_font_size=10,background_color='white
```

In [480…

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

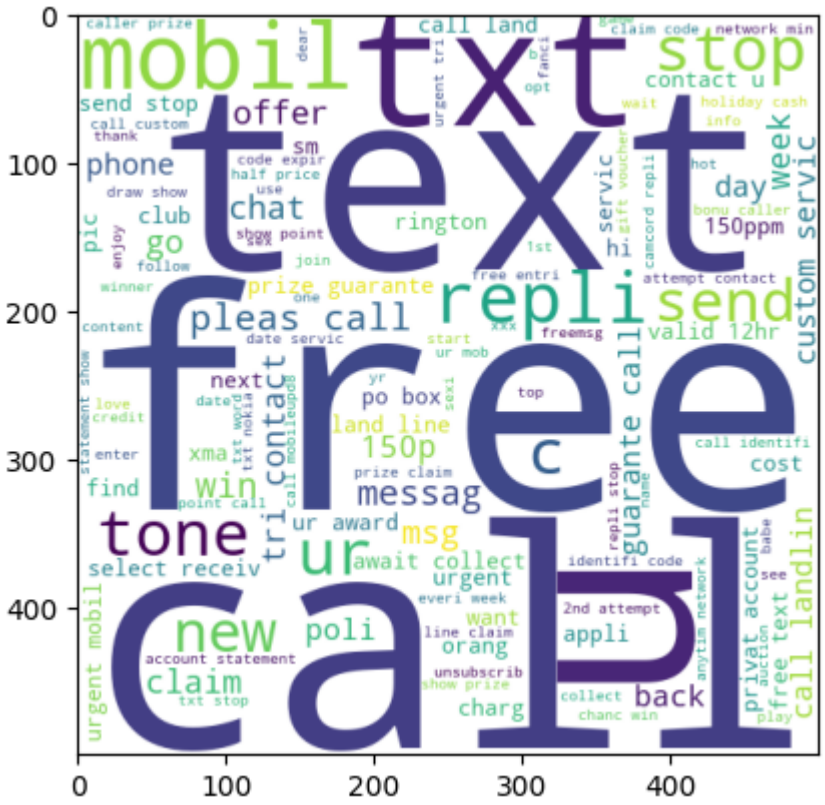
Out[480]:

```
<matplotlib.image.AxesImage at 0x1748f7cbd50>
```



In [481…
```
spam_wc = wc.generate(df[df['target'] == 1]['transformed_text'].str.cat(sep=
plt.imshow(spam_wc)
```

Out[481]: <matplotlib.image.AxesImage at 0x174b2cf38d0>



Loading [MathJax]/extensions/Safe.js

In [482…
```
spam_corpus = []
for msg in df[df['target'] == 0]['transformed_text'].tolist():
    for word in msg.split():
        spam_corpus.append(word)
```

In [483…
```
ham_corpus = []
for msg in df[df['target'] == 1]['transformed_text'].tolist():
    for word in msg.split():
        ham_corpus.append(word)
```

In [484…
```
len(spam_corpus)
```

Out[484]: 35404

In [485…
```
len(ham_corpus)
```

Out[485]: 9939

In [486…
```
df.head()
```

Out[486]:

	target	message	num_characters	num_sentences	num_words	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

Model Building

In [487…
```
from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer
cv = CountVectorizer()
tfidf = TfidfVectorizer(max_features=3000)
```

In [488…
```
X = tfidf.fit_transform(df['transformed_text']).toarray()
```

In [489…
```
X.shape
```

Out[489]: (5169, 3000)

In [490…
```
Y = df['target'].values
```

In [491…
```
from sklearn.model_selection import train_test_split
```

In [492…
```
x_train,x_test,y_train,y_test = train_test_split(X,Y,test_size=0.2,random_st
```

In [493…
```
from sklearn.naive_bayes import GaussianNB,MultinomialNB,BernoulliNB
from sklearn.metrics import accuracy_score,confusion_matrix,precision_score
```

In [494…
```
gnb = GaussianNB()
mnb = MultinomialNB()
bnb = BernoulliNB()
```

In [495…
```
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 [496…
```
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 [497…
```
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 [498…
```
#tfidf-->MNB
```

In [499…
```
!pip install xgboost
```

```
Requirement already satisfied: xgboost in c:\users\dell\appdata\local\program
s\python\python311\lib\site-packages (2.0.2)
Requirement already satisfied: numpy in c:\users\dell\appdata\local\programs
\python\python311\lib\site-packages (from xgboost) (1.24.3)
Requirement already satisfied: scipy in c:\users\dell\appdata\local\programs
\python\python311\lib\site-packages (from xgboost) (1.11.2)
```

In [500…
```
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 [501…
```
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)
etc =  ExtraTreesClassifier(n_estimators=50,random_state=2)
bc = BaggingClassifier(n_estimators=50,random_state=2)
gbdt = GradientBoostingClassifier(n_estimators=50,random_state=2)
xgb =  XGBClassifier(n_estimators=50,random_state=2)
```

In [502…
```
clfs  = {
    'SVC' : svc,
    'KN' : knc,
    'NB' : mnb,
    'DT' : dtc,
    'LR' : lrc,
    'RF' : rfc,
    'AdaBoost' : abc,
    'BgC' : bc,
    'ETC' : etc,
    'GBDT' : gbdt,
    'xbg': xgb
```

```
}
```

In [503…
```
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 [504…
```
train_classifier(svc,x_train,y_train,x_test,y_test)
```

Out[504]:
```
(0.9758220502901354, 0.9747899159663865)
```

In [505…
```
accuracy_scores = []
precision_scores = []
for name,clf in clfs.items():
    current_accuracy,current_precision = train_classifier(clf,x_train,y_trai
    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 -  0.9294003868471954
Precision -  0.8282828282828283
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 xbg
Accuracy -  0.9671179883945842
Precision -  0.9262295081967213
```

In [506…
```
performance_df = pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy':accuracy_s
```

In [507…
```
performance_df
```

Out[507]:

	Algorithm	Accuracy	Precision
0	SVC	0.975822	0.974790
1	KN	0.905222	1.000000
2	NB	0.970986	1.000000
3	DT	0.929400	0.828283
4	LR	0.958414	0.970297
5	RF	0.975822	0.982906
6	AdaBoost	0.960348	0.929204
7	BgC	0.958414	0.868217
8	ETC	0.974855	0.974576
9	GBDT	0.946809	0.919192
10	xbg	0.967118	0.926230

In [508…
```
performance_df1 = pd.melt(performance_df,id_vars = 'Algorithm')
```

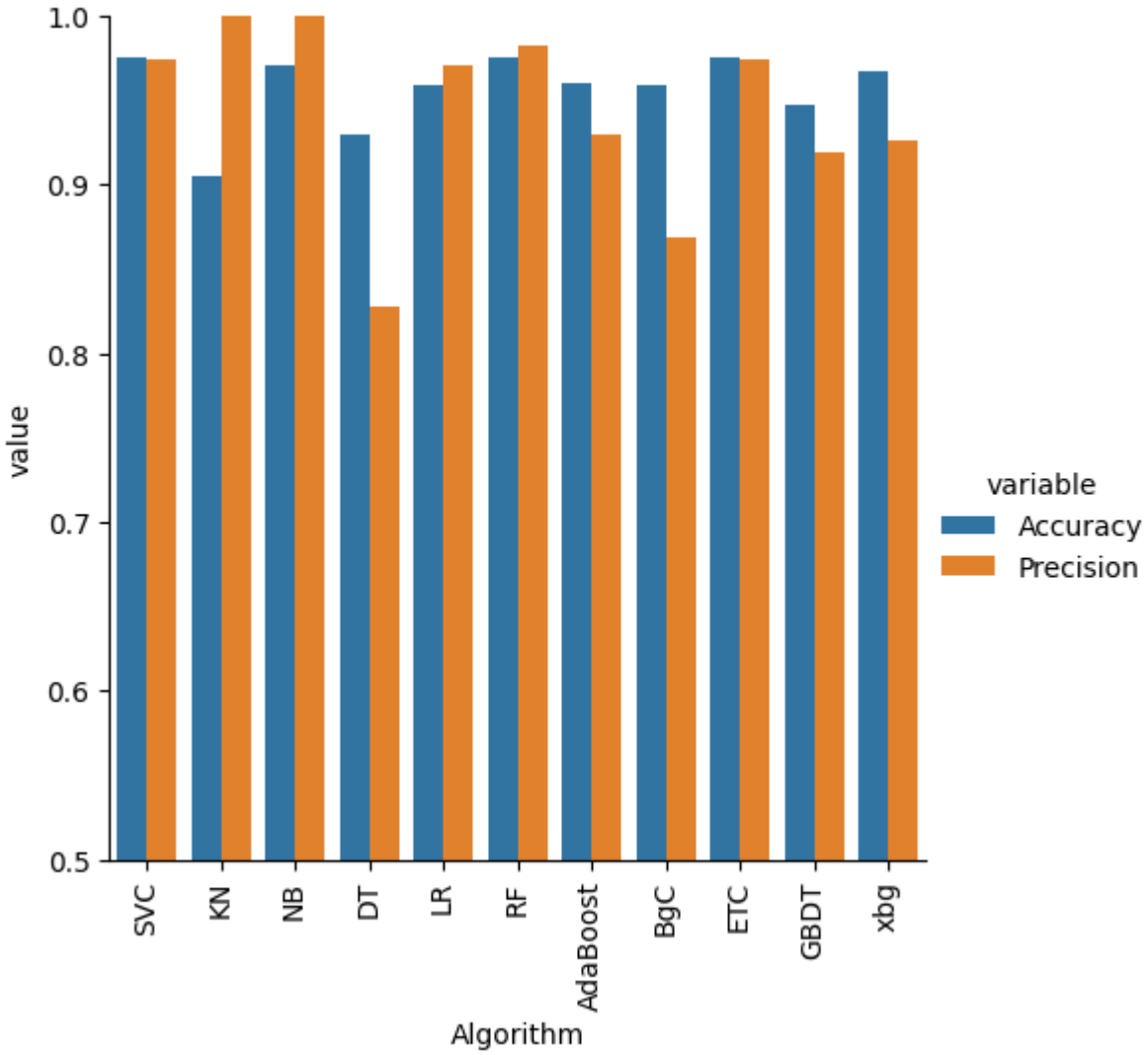
In [509…
```
performance_df1
```

Out[509]:

	Algorithm	variable	value
0	SVC	Accuracy	0.975822
1	KN	Accuracy	0.905222
2	NB	Accuracy	0.970986
3	DT	Accuracy	0.929400
4	LR	Accuracy	0.958414
5	RF	Accuracy	0.975822
6	AdaBoost	Accuracy	0.960348
7	BgC	Accuracy	0.958414
8	ETC	Accuracy	0.974855
9	GBDT	Accuracy	0.946809
10	xbg	Accuracy	0.967118
11	SVC	Precision	0.974790
12	KN	Precision	1.000000
13	NB	Precision	1.000000
14	DT	Precision	0.828283
15	LR	Precision	0.970297
16	RF	Precision	0.982906
17	AdaBoost	Precision	0.929204
18	BgC	Precision	0.868217
19	ETC	Precision	0.974576
20	GBDT	Precision	0.919192
21	xbg	Precision	0.926230

In [510…

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



In [511…
```
#model improve
```

In [512…
```
temp_df = pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy_max_ft_3000':accur
```

In [513…
```
new_df = performance_df.merge(temp_df,on='Algorithm')
new_df_scaled = new_df.merge(temp_df,on='Algorithm')
```

In [514…
```
temp_df = pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy_num_chars':accurac
```

In [515…
```
new_df_scaled.merge(temp_df,on='Algorithm')
```

Out[515]:

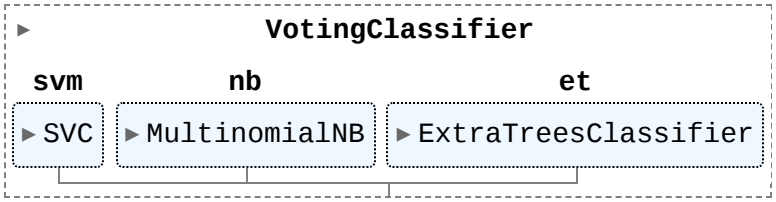
	Algorithm	Accuracy	Precision	Accuracy_max_ft_3000_x	Precision_max_ft_3000_x
0	SVC	0.975822	0.974790	0.975822	0.974790
1	KN	0.905222	1.000000	0.905222	1.000000
2	NB	0.970986	1.000000	0.970986	1.000000
3	DT	0.929400	0.828283	0.929400	0.828283
4	LR	0.958414	0.970297	0.958414	0.970297
5	RF	0.975822	0.982906	0.975822	0.982906
6	AdaBoost	0.960348	0.929204	0.960348	0.929204
7	BgC	0.958414	0.868217	0.958414	0.868217
8	ETC	0.974855	0.974576	0.974855	0.974576
9	GBDT	0.946809	0.919192	0.946809	0.919192
10	xbg	0.967118	0.926230	0.967118	0.926230

In [516…
```
#Voting classifier
```

In [517…
```
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 [518…
```
voting = VotingClassifier(estimators=[('svm',svc),('nb',mnb),('et',etc)],vot
voting.fit(x_train,y_train)
```

Out[518]:



In [519…
```
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 [520…
```
#Applying stacking
```

In [521…
```
estimators=[('svm',svc),('nb',mnb),('et',etc)]
final_estimator=RandomForestClassifier()
```

In [522…
```
from sklearn.ensemble import StackingClassifier
```

In [523…
```
clf = StackingClassifier(estimators=estimators,final_estimator=final_estimat
```

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

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

In [ ]:
```

```