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

In [2]:
```
df = pd.read_csv(r'C:\Users\Arindham Krishna\OneDrive\Desktop\Projects for Portfolio\Spam Detect
```

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)
```

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
```

We can see that there are null values in col, unnamed:2 3 and 4

We shall drop these columns as they bring no help in our analysis.

In [6]:
```
#dropping columns.
df.drop(columns=['Unnamed: 2','Unnamed: 3','Unnamed: 4'],inplace=True)
```

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

Out[7]:

	v1	v2
28	ham	I'm back & we're packing the car now, I'll...
809	ham	Ugh I don't wanna get out of bed. It's so warm.
3914	ham	Ard 530 lor. I ok then message _ lor.
4603	ham	THANX 4 PUTTIN DA FONE DOWN ON ME!!
1954	ham	Good night. Am going to sleep.

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

Out[8]:

	target	text
3035	ham	;-) ok. I feel like john lennon.
391	ham	Hey so this sat are we going for the intro pil...
2959	ham	Sir send to group mail check it.
687	ham	Dear,Me at cherthala.in case u r coming cochin...
4127	ham	I dont thnk its a wrong calling between us

Now, as we can see that we have our target and text column.

We will do the label encoding for our target column, that means the spam and ham values will be converted to (0,1 or 1,0) format.

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...

You can see here that, ham values are converted to 0 and spam values are converted to 1.

ham: 0 and spam: 1

In [12]:
```
#let's check for 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')
df.duplicated().sum()
```

Out[14]:
```
0
```

In [15]:
```
df.shape
```

Out[15]:
```
(5169, 2)
```

EDA

In [16]:
```
df.head()
```

Out[16]:

	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...

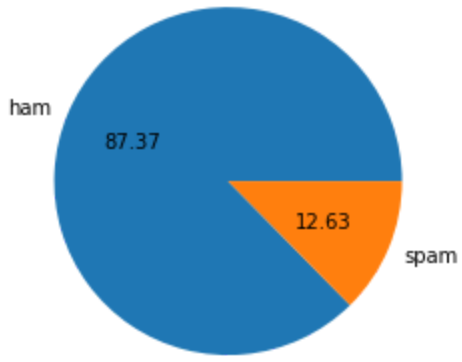
In [17]:
```
df['target'].value_counts()
```

Out[17]:
```
0    4516
1     653
Name: target, dtype: int64
```

We can see that, there are 4516 hams and only 653 spams.

The data is imbalanced.

In [18]:
```
import matplotlib.pyplot as plt
plt.pie(df['target'].value_counts(), labels=['ham','spam'],autopct="%0.2f")
plt.show()
```



In [19]:
```
import nltk
!pip install nltk
nltk.download('punkt')
```

```
Requirement already satisfied: nltk in c:\users\arindham krishna\anaconda3\lib\site-packages (3.
6.2)
Requirement already satisfied: joblib in c:\users\arindham krishna\anaconda3\lib\site-packages
(from nltk) (1.0.1)
Requirement already satisfied: regex in c:\users\arindham krishna\anaconda3\lib\site-packages (f
rom nltk) (2021.7.6)
Requirement already satisfied: tqdm in c:\users\arindham krishna\anaconda3\lib\site-packages (fr
om nltk) (4.61.2)
Requirement already satisfied: click in c:\users\arindham krishna\anaconda3\lib\site-packages (f
rom nltk) (8.0.1)
Requirement already satisfied: colorama in c:\users\arindham krishna\anaconda3\lib\site-packages
(from click->nltk) (0.4.4)
[nltk_data] Downloading package punkt to C:\Users\Arindham
[nltk_data]     Krishna\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

Out[19]:
```
True
```

In [20]:
```
df['num_characters'] = df['text'].apply(len)
```

In [21]:
```
df.head()
```

Out[21]:

	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 [22]:
```
# num of words
df['num_words'] = df['text'].apply(lambda x:len(nltk.word_tokenize(x)))
```

In [23]:
```
df.head()
```

Out[23]:

	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 [24]:
```
df['num_sentences'] = df['text'].apply(lambda x:len(nltk.sent_tokenize(x)))
df.head()
```

Out[24]:

	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 [25]:
```
df[['num_characters','num_words','num_sentences']].describe()
```

Out[25]:

	num_characters	num_words	num_sentences
count	5169.000000	5169.000000	5169.000000
mean	78.923776	18.456375	1.962275
std	58.174846	13.323322	1.433892
min	2.000000	1.000000	1.000000
25%	36.000000	9.000000	1.000000
50%	60.000000	15.000000	1.000000
75%	117.000000	26.000000	2.000000
max	910.000000	220.000000	38.000000

In [26]:
```
# ham
df[df['target'] == 0][['num_characters','num_words','num_sentences']].describe()
```

Out[26]:

	num_characters	num_words	num_sentences
count	4516.000000	4516.000000	4516.000000
mean	70.456820	17.123339	1.815545
std	56.356802	13.491315	1.364098
min	2.000000	1.000000	1.000000
25%	34.000000	8.000000	1.000000
50%	52.000000	13.000000	1.000000
75%	90.000000	22.000000	2.000000
max	910.000000	220.000000	38.000000

In [27]:
```
#spam
df[df['target'] == 1][['num_characters','num_words','num_sentences']].describe()
```

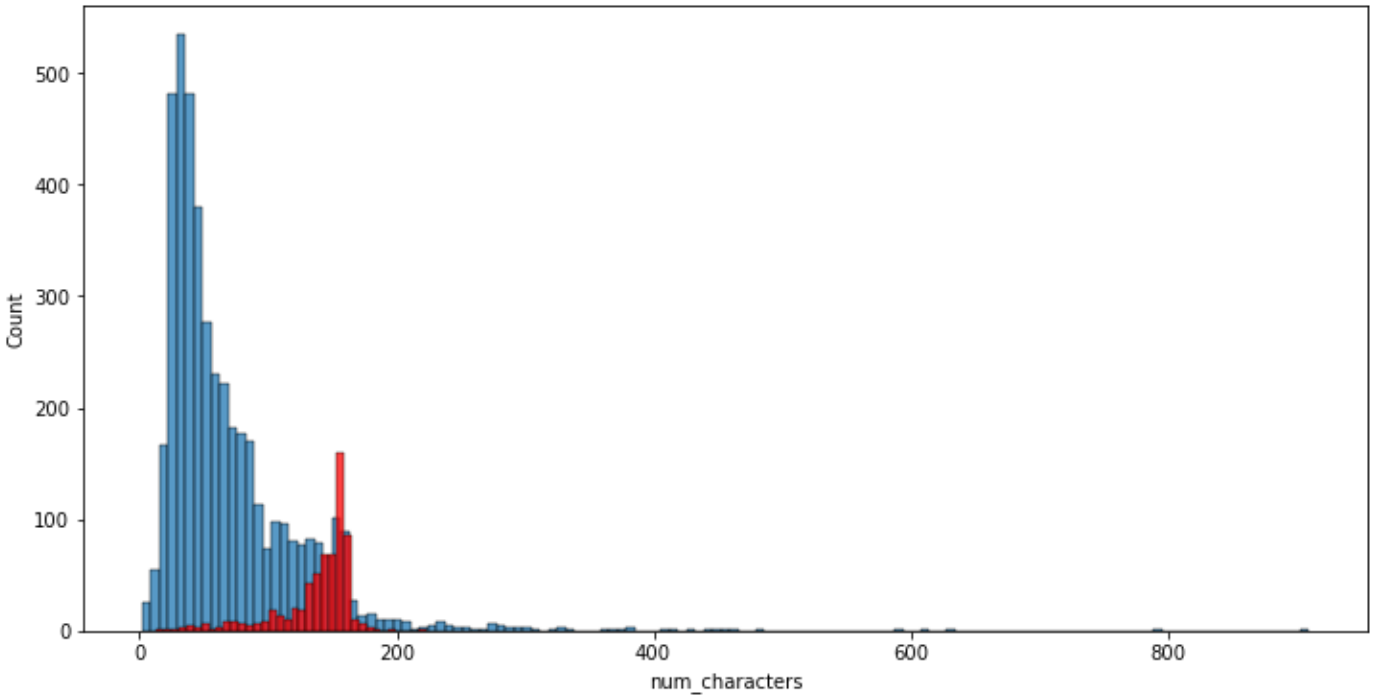
Out[27]:

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

In [28]:
```
import seaborn as sns
```

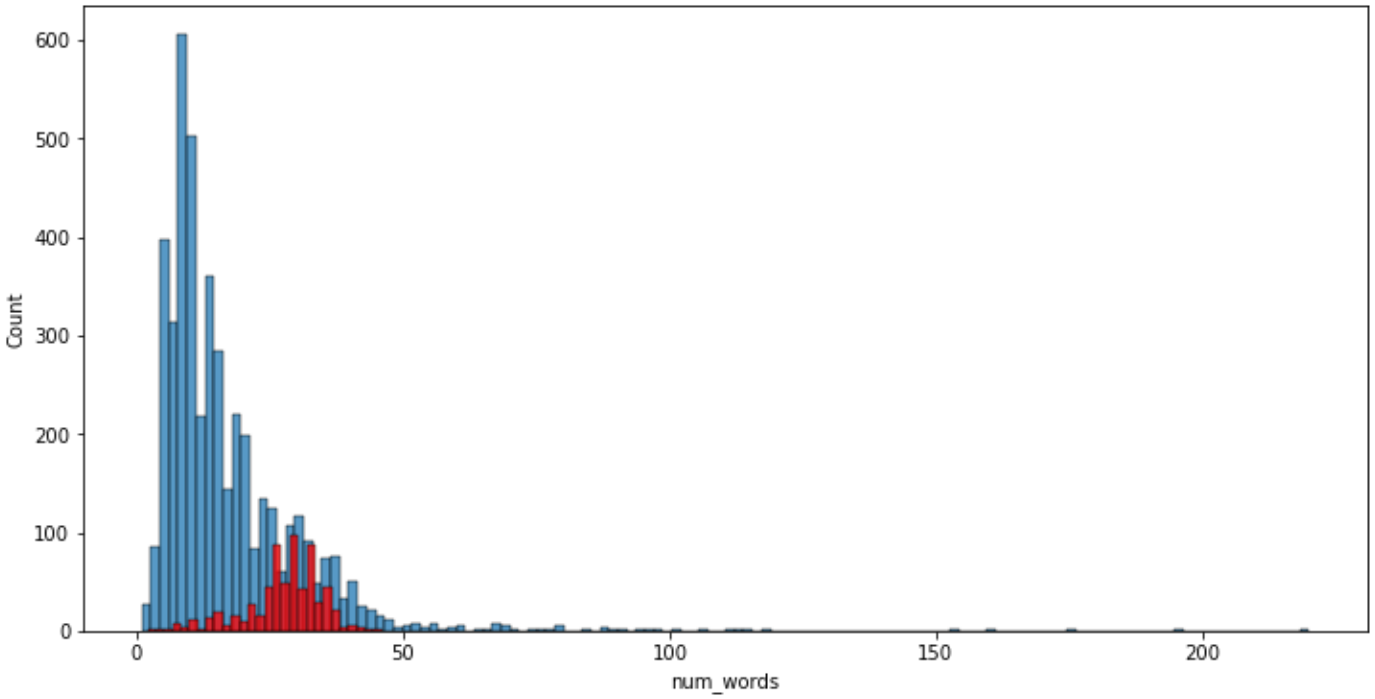
In [29]:
```
plt.figure(figsize=(12,6))
sns.histplot(df[df['target'] == 0]['num_characters'])
sns.histplot(df[df['target'] == 1]['num_characters'],color='red')
```

Out[29]: <AxesSubplot:xlabel='num_characters', ylabel='Count'>



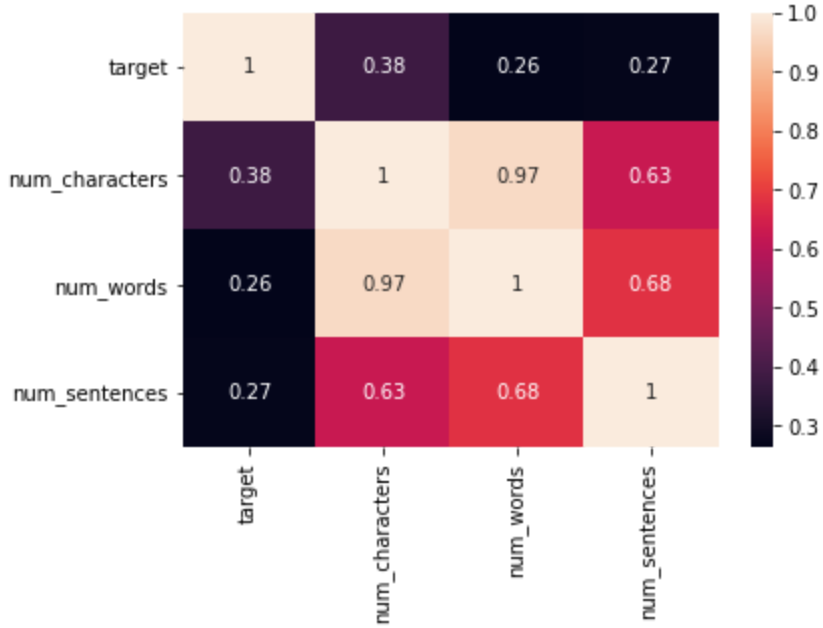
In [30]:
```
plt.figure(figsize=(12,6))
sns.histplot(df[df['target'] == 0]['num_words'])
sns.histplot(df[df['target'] == 1]['num_words'],color='red')
```

Out[30]: <AxesSubplot:xlabel='num_words', ylabel='Count'>



In [31]:
```
sns.heatmap(df.corr(),annot=True)
```

Out[31]: <AxesSubplot:>



Data Pre Processing

Lower case

Tokenization

Removing special characters

Removing stop words and punctuation

Stemming

In [32]:
```
import nltk
```

In [33]:
```
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()
```

In [34]:
```
from nltk.corpus import stopwords
stopwords.words('english')
```

Out[34]:
```
['i',
 'me',
 'my',
 'myself',
 'we',
 'our',
 'ours',
 'ourselves',
 'you',
 "you're",
 "you've",
 "you'll",
 "you'd",
 'your',
 'yours',
 'yourself',
 'yourselves',
 'he',
 'him',
 'his',
 'himself',
 'she',
 "she's",
 'her',
 'hers',
 'herself',
 'it',
 "it's",
 'its',
 'itself',
 'they',
 'them',
 'their',
 'theirs',
 'themselves',
 'what',
 'which',
 'who',
 'whom',
 'this',
 'that',
 "that'll",
 'these',
 'those',
 'am',
 'is',
 'are',
 'was',
 'were',
 'be',
 'been',
 'being',
 'have',
 'has',
 'had',
 'having',
 'do',
 'does',
 'did',
 'doing',
 'a',
```

```
'an',
'the',
'and',
'but',
'if',
'or',
'because',
'as',
'until',
'while',
'of',
'at',
'by',
'for',
'with',
'about',
'against',
'between',
'into',
'through',
'during',
'before',
'after',
'above',
'below',
'to',
'from',
'up',
'down',
'in',
'out',
'on',
'off',
'over',
'under',
'again',
'further',
'then',
'once',
'here',
'there',
'when',
'where',
'why',
'how',
'all',
'any',
'both',
'each',
'few',
'more',
'most',
'other',
'some',
'such',
'no',
'nor',
'not',
'only',
'own',
'same',
'so',
```

```
 'than',
 'too',
 'very',
 's',
 't',
 'can',
 'will',
 'just',
 'don',
 "don't",
 'should',
 "should've",
 'now',
 'd',
 'll',
 'm',
 'o',
 're',
 've',
 'y',
 'ain',
 'aren',
 "aren't",
 'couldn',
 "couldn't",
 'didn',
 "didn't",
 'doesn',
 "doesn't",
 'hadn',
 "hadn't",
 'hasn',
 "hasn't",
 'haven',
 "haven't",
 'isn',
 "isn't",
 'ma',
 'mightn',
 "mightn't",
 'mustn',
 "mustn't",
 'needn',
 "needn't",
 'shan',
 "shan't",
 'shouldn',
 "shouldn't",
 'wasn',
 "wasn't",
 'weren',
 "weren't",
 'won',
 "won't",
 'wouldn',
 "wouldn't"]
```

In [35]:
```
import string
string.punctuation
```

Out[35]:
```
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
```

In [36]:
```
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 [37]:
```
df['transformed_text'] = df['text'].apply(transform_text)
```

In [38]:
```
df.head()
```

Out[38]:

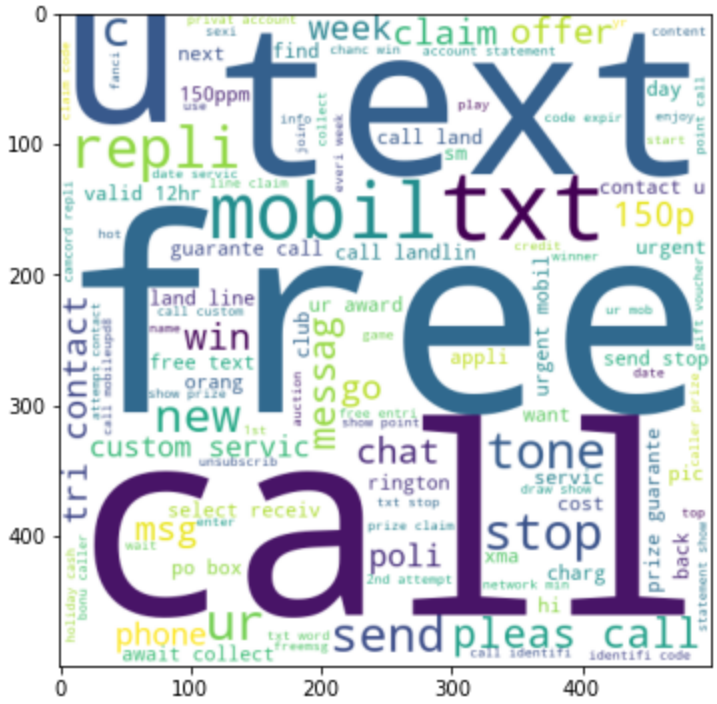
	target	text	num_characters	num_words	num_sentences	transformed_text
0	0	Go until jurong point, crazy.. Available only ...	111	24	2	go jurong point crazi avail bugi n great world...
1	0	Ok lar... Joking wif u oni...	29	8	2	ok lar joke wif u oni
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	155	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 [39]:
```
from wordcloud import WordCloud
wc = WordCloud(width=500,height=500,min_font_size=10,background_color='white')
```

In [40]:
```
spam_wc = wc.generate(df[df['target'] == 1]['transformed_text'].str.cat(sep=" "))
```

In [41]:
```
plt.figure(figsize=(15,6))
plt.imshow(spam_wc)
```

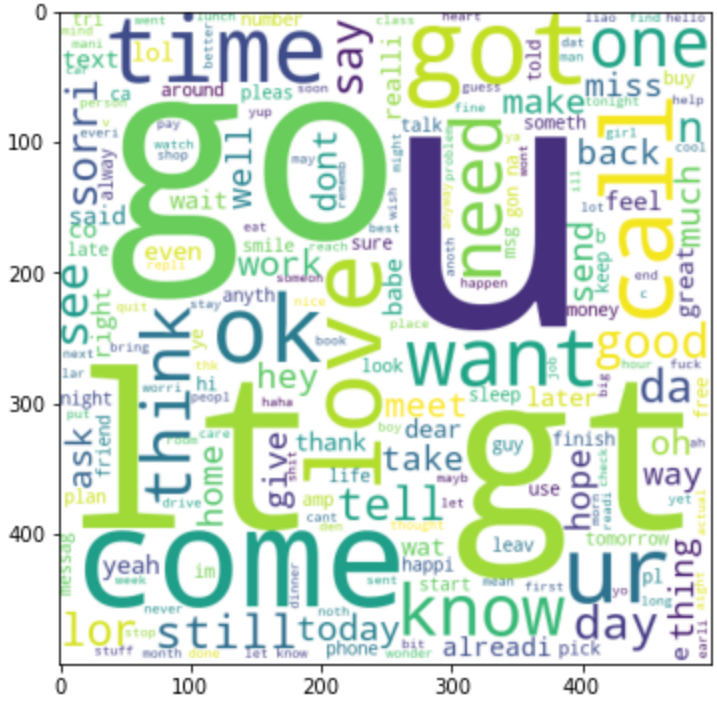
Out[41]:
```
<matplotlib.image.AxesImage at 0x27bb38007c0>
```



In [42]:
```
ham_wc = wc.generate(df[df['target'] == 0]['transformed_text'].str.cat(sep=" "))
```

In [43]:
```
plt.figure(figsize=(15,6))
plt.imshow(ham_wc)
```

Out[43]: <matplotlib.image.AxesImage at 0x27bb3338370>



In [44]:
```
df.head()
```

Out[44]:

	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 [45]:
```
spam_corpus = []
for msg in df[df['target'] == 1]['transformed_text'].tolist():
    for word in msg.split():
        spam_corpus.append(word)
```

In [46]:
```
len(spam_corpus)
```

Out[46]:
```
9941
```

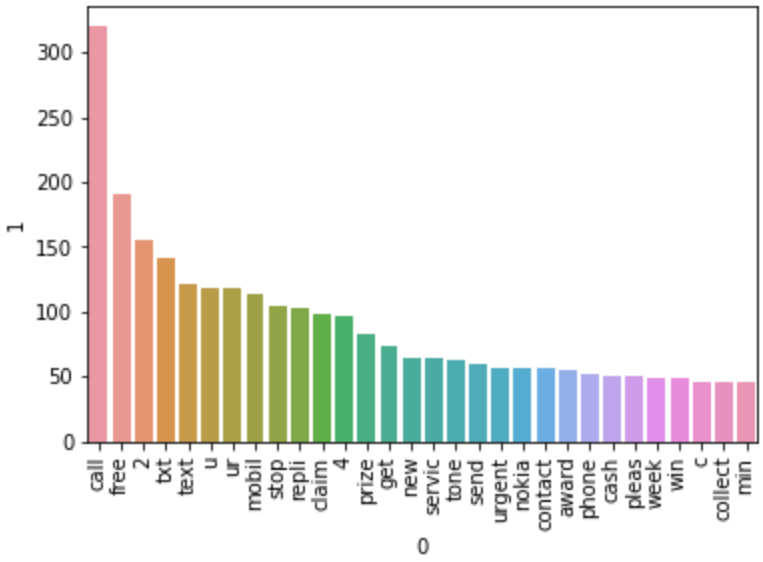
In [47]:
```
from collections import Counter
pd.DataFrame(Counter(spam_corpus).most_common(10))
```

Out[47]:

	0	1
0	call	320
1	free	191
2	2	155
3	txt	141
4	text	122
5	u	119
6	ur	119
7	mobil	114
8	stop	104
9	repli	103

In [48]:
```
from collections import Counter
sns.barplot(pd.DataFrame(Counter(spam_corpus).most_common(30))[0],pd.DataFrame(Counter(spam_corpu
plt.xticks(rotation='vertical')
plt.show()
```

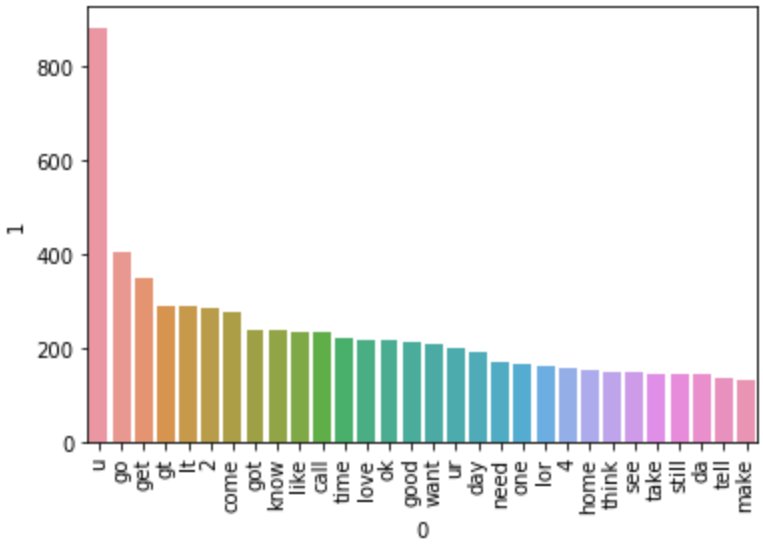
```
C:\Users\Arindham Krishna\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:
Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional
argument will be `data`, and passing other arguments without an explicit keyword will result in
an error or misinterpretation.
  warnings.warn(
```



In [49]:
```
ham_corpus = []
for msg in df[df['target'] == 0]['transformed_text'].tolist():
    for word in msg.split():
        ham_corpus.append(word)
```

In [50]:
```
from collections import Counter
sns.barplot(pd.DataFrame(Counter(ham_corpus).most_common(30))[0],pd.DataFrame(Counter(ham_corpus
plt.xticks(rotation='vertical')
plt.show()
```

```
C:\Users\Arindham Krishna\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:
Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional
argument will be `data`, and passing other arguments without an explicit keyword will result in
an error or misinterpretation.
  warnings.warn(
```



Model Building

In [51]:
```
from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer
cv = CountVectorizer()
tfidf = TfidfVectorizer(max_features=3000)
```

In [52]:
```
X = tfidf.fit_transform(df['transformed_text']).toarray()
```

In [53]:
```
X.shape
```

Out[53]: (5169, 3000)

In [54]:
```
y = df['target'].values
```

In [55]:
```
from sklearn.model_selection import train_test_split
```

In [56]:
```
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,recall_score
```

In [58]:
```
gnb = GaussianNB()
mnb = MultinomialNB()
bnb = BernoulliNB()
```

In [71]:
```
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))
print(recall_score(y_test,y_pred1))
```

```
0.8704061895551257
[[788 108]
 [ 26 112]]
0.509090909090909
0.8115942028985508
```

In [72]:
```
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))
print(recall_score(y_test,y_pred2))
```

```
0.971953578336557
[[896   0]
 [ 29 109]]
1.0
0.7898550724637681
```

In [73]:
```
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))
print(recall_score(y_test,y_pred3))
```

```
0.9835589941972921
[[895   1]
 [ 16 122]]
0.991869918699187
0.8840579710144928
```

In [ ]:
```

```