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/shrut/Downloads/spam.csv',encoding='ISO-8859-1')
```

In [3]:
```
df.head(6)
```

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
5	spam	FreeMsg Hey there darling it's been 3 week's n...	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
```

In [6]:
```
#Droping last 3 columns
df.drop(columns=['Unnamed: 2','Unnamed: 3','Unnamed: 4'],axis=1,inplace=True)
```

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

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]:
```
#renaming columns
df.rename(columns={'v1':'Target','v2':'Text'},inplace=True)
```

In [9]:
```
df.sample(5)
```

Out[9]:

	Target	Text
3736	ham	Plz note: if anyone calling from a mobile Co. ...
3786	ham	WHORE YOU ARE UNBELIEVABLE.
3639	ham	He's really into skateboarding now despite the...
1918	ham	Is fujitsu s series lifebook good?
639	ham	I had askd u a question some hours before. Its...

In [10]:
```
from sklearn.preprocessing import LabelEncoder
encoder= LabelEncoder()
```

In [11]:
```
df['Target']=encoder.fit_transform(df['Target'])
```

In [12]:
```
df.head(5)
```

Out[12]:

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

In [13]:
```
#checking missing values
df.isnull().sum()
```

Out[13]:
```
Target    0
Text      0
dtype: int64
```

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

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

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

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

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

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

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

Exploratory Data Analysis

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

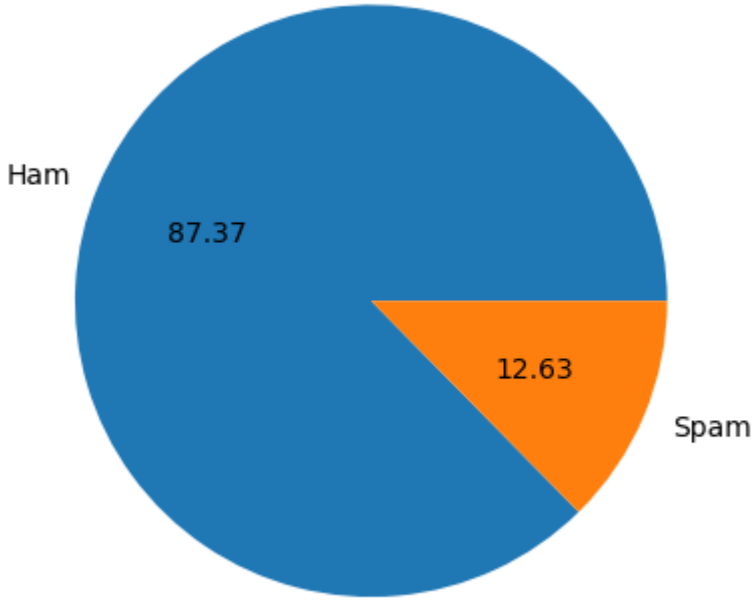
Out[18]:

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

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

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

In [20]:
```
plt.pie(df['Target'].value_counts(),labels=['Ham','Spam'],autopct="%0.2f")
plt.show()
```



From the above piechart, we get to know that our data is imbalanced

In [21]:
```
import nltk #natural language tool kit
```

In [22]:
```
nltk.download('punkt')
```

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

Out[22]:
```
True
```

In [23]:
```
df['num_characters']=df['Text'].apply(len)
```

In [24]:
```
df.head(4)
```

Out[24]:

	Target	Text	num_characters
0	0	Go until jurong point, crazy.. Available only ...	111
1	0	Ok lar... Joking wif u oni...	29
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	155
3	0	U dun say so early hor... U c already then say...	49

In [25]:
```
#fetching the number of words
df['num_words']=df['Text'].apply(lambda x:len(nltk.word_tokenize(x))) #passing
```

In [26]:
```
df.head(5)
```

Out[26]:

	Target	Text	num_characters	num_words
0	0	Go until jurong point, crazy.. Available only ...	111	24
1	0	Ok lar... Joking wif u oni...	29	8
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	155	37
3	0	U dun say so early hor... U c already then say...	49	13
4	0	Nah I don't think he goes to usf, he lives aro...	61	15

In [27]:
```
df['num_sentences']=df['Text'].apply(lambda x:len(nltk.sent_tokenize(x)))
```

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

Out[28]:

	Target	Text	num_characters	num_words	num_sentences
0	0	Go until jurong point, crazy.. Available only ...	111	24	2
1	0	Ok lar... Joking wif u oni...	29	8	2
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	155	37	2
3	0	U dun say so early hor... U c already then say...	49	13	1
4	0	Nah I don't think he goes to usf, he lives aro...	61	15	1

In [29]:
```
#Ham
df[df['Target']==0][['num_characters','num_words','num_sentences']].describe()
```

Out[29]:

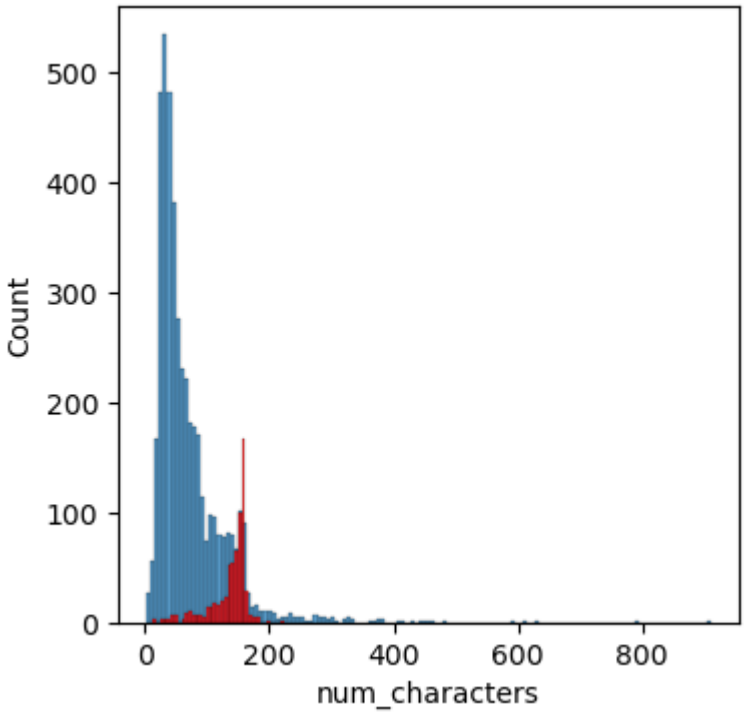
	num_characters	num_words	num_sentences
count	4516.000000	4516.000000	4516.000000
mean	70.459256	17.123782	1.820195
std	56.358207	13.493970	1.383657
min	2.000000	1.000000	1.000000
25%	34.000000	8.000000	1.000000
50%	52.000000	13.000000	1.000000
75%	90.000000	22.000000	2.000000
max	910.000000	220.000000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

In [30]:
```
#Spam
df[df['Target']==1][['num_characters','num_words','num_sentences']].describe()
```

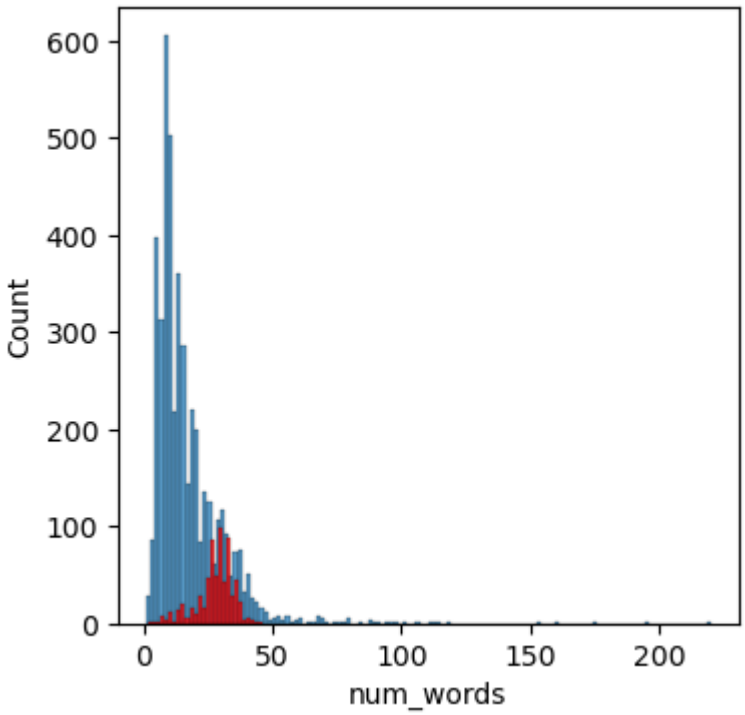
Out[30]:

	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 [31]:
```
plt.figure(figsize=(4,4))
sns.histplot(df[df['Target']==0]['num_characters'])
sns.histplot(df[df['Target']==1]['num_characters'],color='red')
plt.show()
```

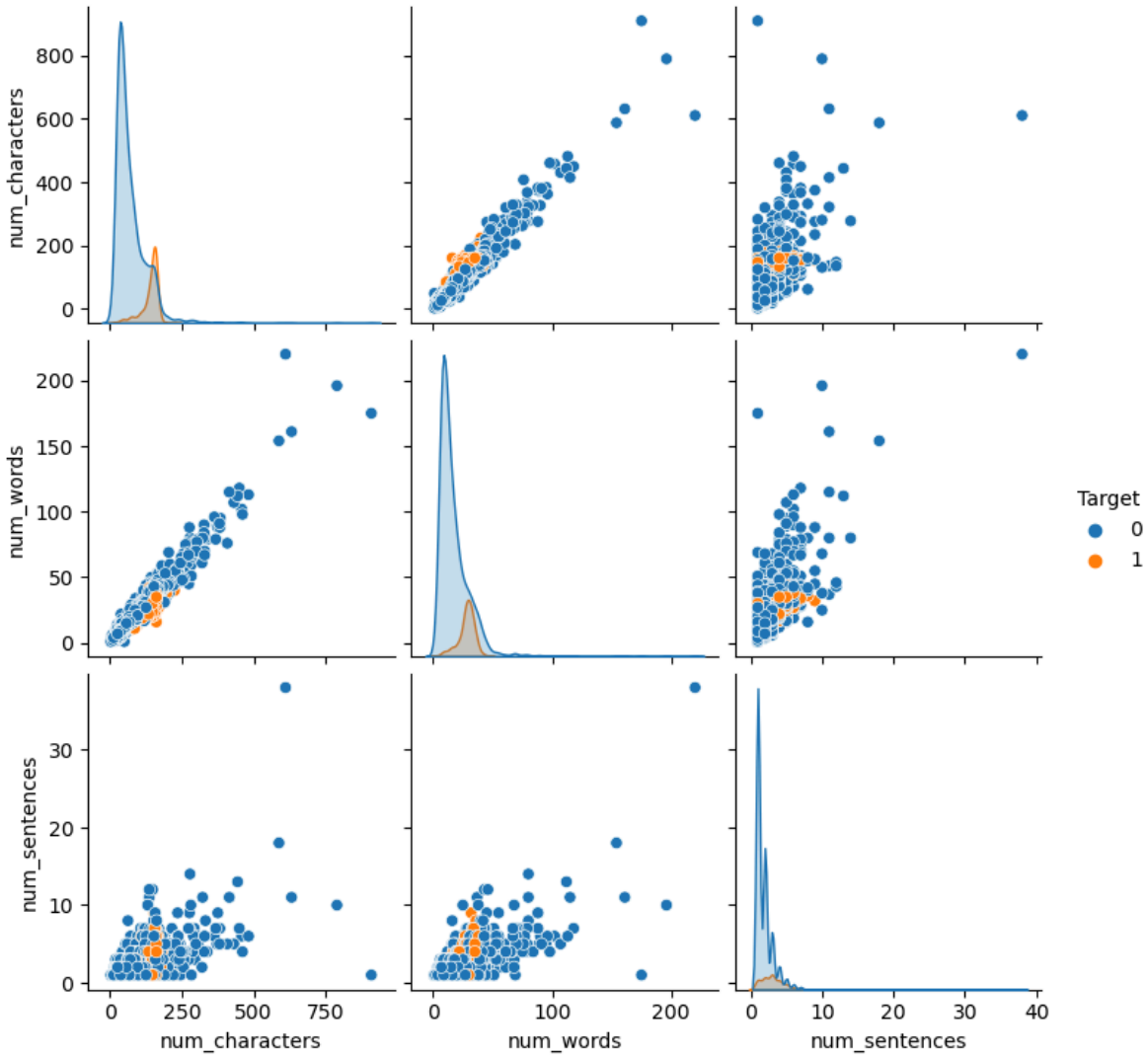


In [32]:
```
plt.figure(figsize=(4,4))
sns.histplot(df[df['Target']==0]['num_words'])
sns.histplot(df[df['Target']==1]['num_words'],color='red')
plt.show()
```



In [33]:
```
sns.pairplot(df,hue='Target') #relationship between them
```

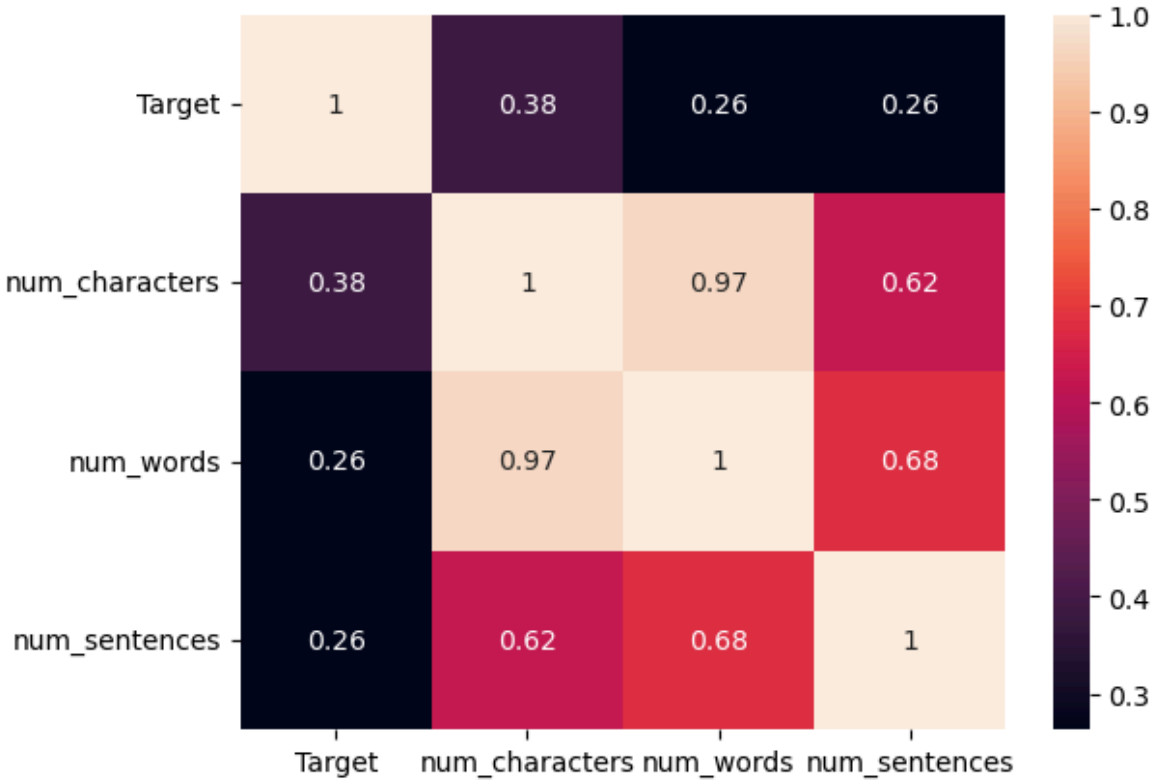
Out[33]: <seaborn.axisgrid.PairGrid at 0x11d09c129d0>



In [34]:
```
sns.heatmap(df.corr(),annot=True)
```

```
C:\Users\shrut\AppData\Local\Temp\ipykernel_21024\4277794465.py:1: FutureWar
ning: The default value of numeric_only in DataFrame.corr is deprecated. In
a future version, it will default to False. Select only valid columns or spe
cify the value of numeric_only to silence this warning.
  sns.heatmap(df.corr(),annot=True)
```

Out[34]: <Axes: >



Data Preprocessing

In [35]:
```
import nltk
from nltk.corpus import stopwords
import string #for punctuation
from nltk.stem.porter import PorterStemmer
ps=PorterStemmer()
nltk.download('stopwords')

def transform_text(text):
    text= text.lower()
    text= nltk.word_tokenize(text)

    y=[] #removing special characters
    for i in text:
        if i.isalnum():
            y.append(i)

    text=y[:] #coping entire list
    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)
```

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

In [36]:
```
transform_text('I loved the youtube video on machine learning!!.How about you?
```

Out[36]:
```
'love  youtub  video  machin  learn'
```

In [37]:
```
df['transform_text']=df['Text'].apply(transform_text)
```

In [38]:
```
df.head(4)
```

Out[38]:

	Target	Text	num_characters	num_words	num_sentences	transform_text
0	0	Go until jurong point, crazy.. Available only ...	111	24	2	go jurong point crazi avail bugi n grea...
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 fina...
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

In [39]:
```
!pip install wordcloud
```

```
Requirement already satisfied: wordcloud in c:\users\shrut\anaconda3\lib\sit
e-packages (1.9.4)
Requirement already satisfied: numpy>=1.6.1 in c:\users\shrut\anaconda3\lib
\site-packages (from wordcloud) (1.24.3)
Requirement already satisfied: pillow in c:\users\shrut\anaconda3\lib\site-p
ackages (from wordcloud) (9.4.0)
Requirement already satisfied: matplotlib in c:\users\shrut\anaconda3\lib\si
te-packages (from wordcloud) (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\shrut\anaconda3
\lib\site-packages (from matplotlib->wordcloud) (1.0.5)
Requirement already satisfied: cycler>=0.10 in c:\users\shrut\anaconda3\lib
\site-packages (from matplotlib->wordcloud) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\shrut\anaconda3
\lib\site-packages (from matplotlib->wordcloud) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\shrut\anaconda3
\lib\site-packages (from matplotlib->wordcloud) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\shrut\anaconda3\l
ib\site-packages (from matplotlib->wordcloud) (23.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\shrut\anaconda3
\lib\site-packages (from matplotlib->wordcloud) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\shrut\anacon
da3\lib\site-packages (from matplotlib->wordcloud) (2.8.2)
Requirement already satisfied: six>=1.5 in c:\users\shrut\anaconda3\lib\site
-packages (from python-dateutil>=2.7->matplotlib->wordcloud) (1.16.0)
```

In [40]:
```
from wordcloud import WordCloud
spam_text = df[df['Target']==1]['transform_text'].str.cat(sep=" ")

if spam_text.strip():
    wc = WordCloud(width=300, height=300, min_font_size=10, background_color='
    plt.figure(figsize=(8, 8))
    plt.imshow(wc, interpolation='bilinear')
    plt.axis('off')
    plt.show()
else:
    print("No spam text available for word cloud.")
```

In [41]:
```
ham_text = df[df['Target']==0]['transform_text'].str.cat(sep=" ")

if ham_text.strip():
    wc = WordCloud(width=300, height=300, min_font_size=10, background_color='
    plt.figure(figsize=(8, 8))
    plt.imshow(wc, interpolation='bilinear')
    plt.axis('off')
    plt.show()
else:
    print("No spam text available for word cloud.")
```

In [42]:
```
spam_corpus=[]
for msg in df[df['Target']==1]['transform_text'].tolist():
    for word in msg.split():
        spam_corpus.append(word)
```

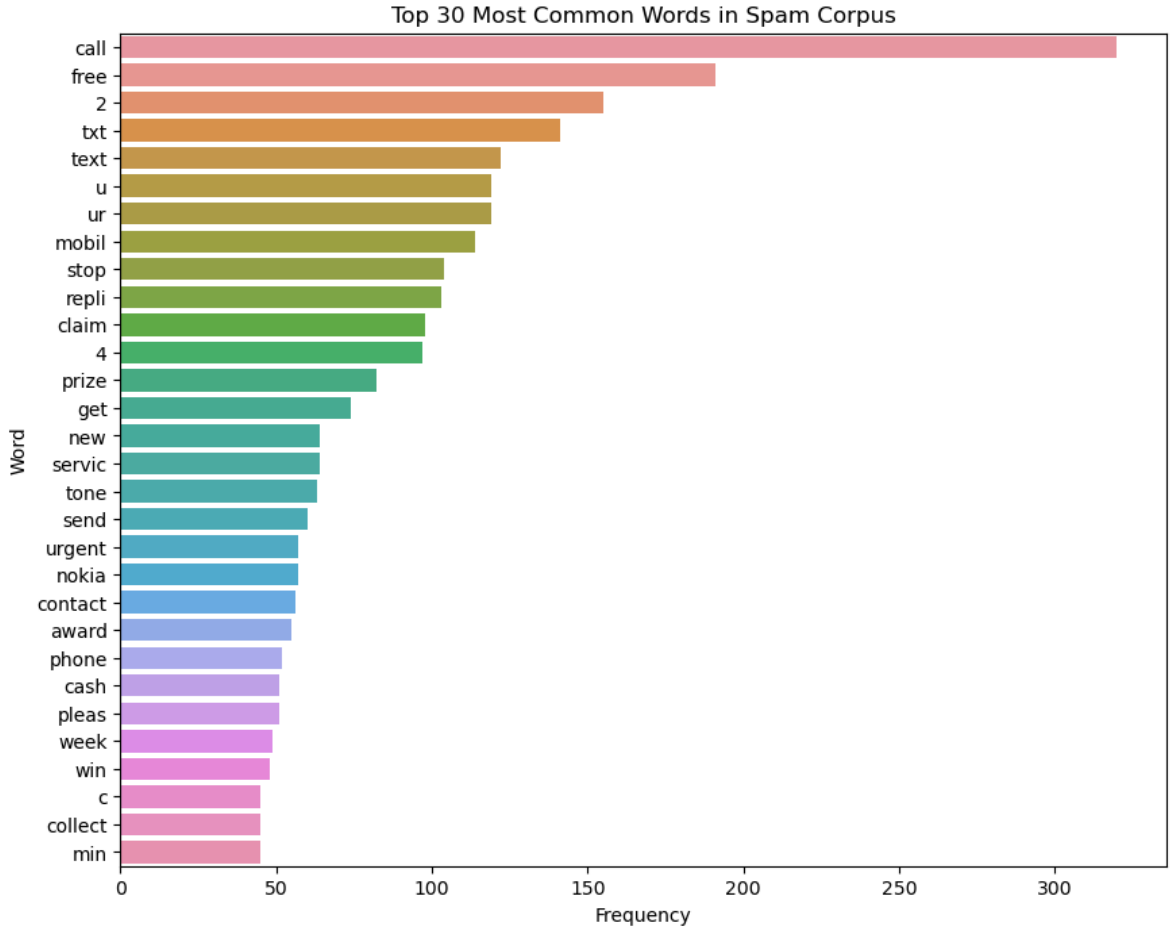
In [43]:
```
spam_corpus
```

Out[43]:
```
['free',
 'entri',
 '2',
 'wkli',
 'comp',
 'win',
 'fa',
 'cup',
 'final',
 'tkt',
 '21st',
 'may',
 'text',
 'fa',
 '87121',
 'receiv',
 'entri',
 'question',
 'std',
```

In [44]:
```
len(spam_corpus)
```

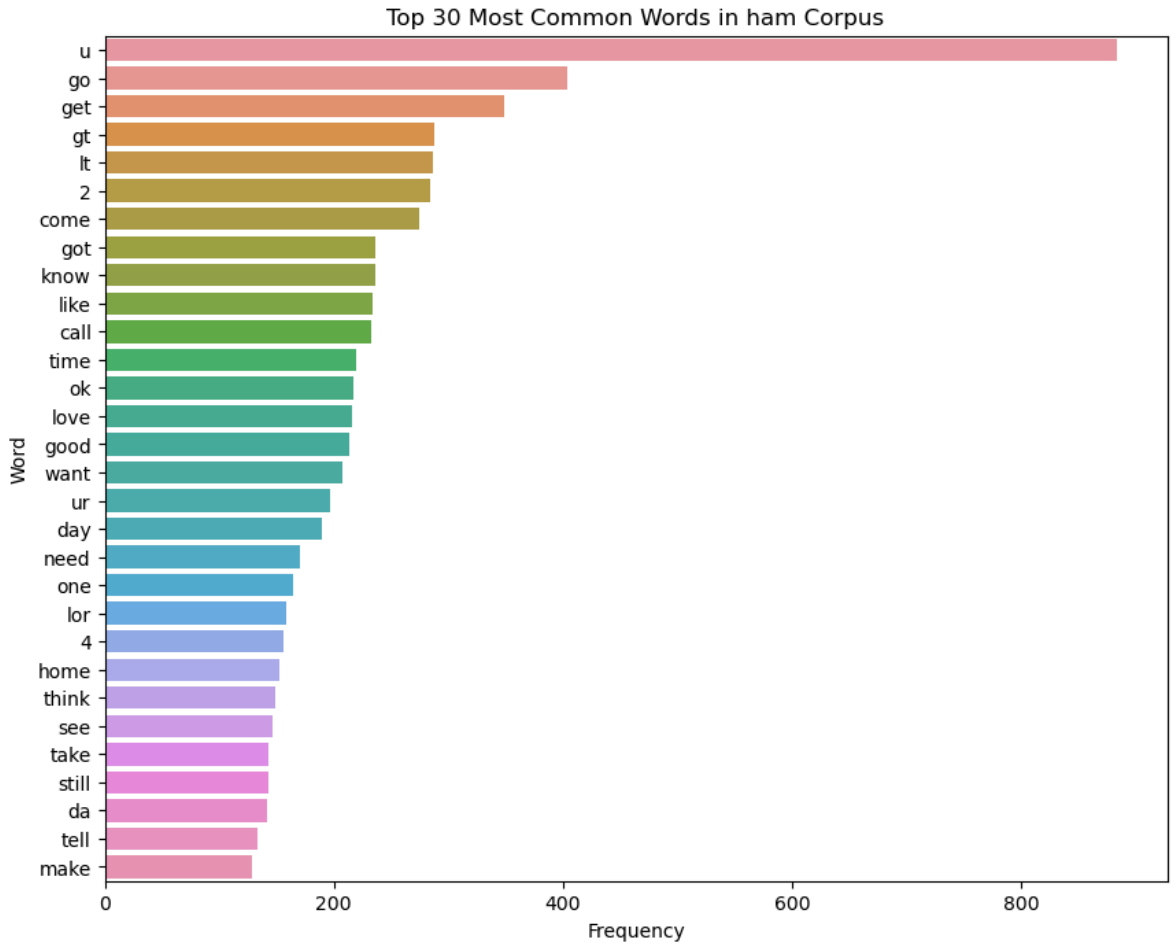
Out[44]:
```
9939
```

In [45]:
```
from collections import Counter

counter_df = pd.DataFrame(Counter(spam_corpus).most_common(30), columns=['Word

plt.figure(figsize=(10,8))
sns.barplot(x='Frequency', y='Word', data=counter_df)
plt.xlabel('Frequency')
plt.ylabel('Word')
plt.title('Top 30 Most Common Words in Spam Corpus')
plt.show()
```



In [46]:
```
ham_corpus=[]
for msg in df[df['Target']==0]['transform_text'].tolist():
    for word in msg.split():
        ham_corpus.append(word)
```

In [47]:
```
count_df = pd.DataFrame(Counter(ham_corpus).most_common(30), columns=['Word',

plt.figure(figsize=(10,8))
sns.barplot(x='Frequency', y='Word', data=count_df)
plt.xlabel('Frequency')
plt.ylabel('Word')
plt.title('Top 30 Most Common Words in ham Corpus')
plt.show()
```



Model Builiding

In [48]:
```
#will use naive bayes as it performs better in textual data
#it accepts numerical values but in our case we have transform_text in the for
#so,need to convert in numerical or vectorization is done
```

In [49]:
```
from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer
cv = CountVectorizer()
tfidf=TfidfVectorizer(max_features=3000)
```

In [50]:
```
x = tfidf.fit_transform(df['transform_text']).toarray()
```

In [51]:
```
x.shape
```

Out[51]:
```
(5169, 3000)
```

In [63]:
```
x
```

Out[63]:
```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
```

In [52]:
```
y = df['Target'].values
```

In [53]:
```
y
```

Out[53]:
```
array([0, 0, 1, ..., 0, 0, 0])
```

In [54]:
```
from sklearn.model_selection import train_test_split
```

In [55]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_stat
```

In [56]:
```
from sklearn.naive_bayes import GaussianNB,MultinomialNB,BernoulliNB
from sklearn.metrics import accuracy_score,precision_score,confusion_matrix
```

In [57]:
```
gnb=GaussianNB()
mnb=MultinomialNB()
bnb=BernoulliNB()
```

In [58]:
```
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 [59]:
```
mnb.fit(x_train,y_train)
y_pred1 = mnb.predict(x_test)
print(accuracy_score(y_test,y_pred1))
print(confusion_matrix(y_test,y_pred1))
print(precision_score(y_test,y_pred1))
```

```
0.9709864603481625
[[896   0]
 [ 30 108]]
1.0
```

In [60]:
```
bnb.fit(x_train,y_train)
y_pred1 = bnb.predict(x_test)
print(accuracy_score(y_test,y_pred1))
print(confusion_matrix(y_test,y_pred1))
print(precision_score(y_test,y_pred1))
```

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

In [61]:
```
# tfidf --> mnb
```

In [62]:
```
import pickle
pickle.dump(tfidf,open('vectorizer.pkl','wb'))
pickle.dump(mnb,open('model.pkl','wb'))
```