In [1]:
```
import string
import nltk
import pandas as pd
import matplotlib. pyplot as plt
from nltk.corpus import stopwords
from sklearn import metrics
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
```

In [2]:
```
messages = pd.read_csv('spam.csv',encoding = 'latin-1')
messages.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]:
```
messages.tail()
```

Out[3]:

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
5567	spam	This is the 2nd time we have tried 2 contact u...	NaN	NaN	NaN
5568	ham	Will Ì_ b going to esplanade fr home?	NaN	NaN	NaN
5569	ham	Pity, * was in mood for that. So...any other s...	NaN	NaN	NaN
5570	ham	The guy did some bitching but I acted like i'd...	NaN	NaN	NaN
5571	ham	Rofl. Its true to its name	NaN	NaN	NaN

In [4]:
```
messages = messages.drop(labels = ["Unnamed: 2", "Unnamed: 3", "Unnamed: 4"], axis
messages.columns = ["label", "message"]
```

In [5]:
```
messages.head()
```

Out[5]:

	label	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 [6]:
```
messages.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   label    5572 non-null   object
 1   message  5572 non-null   object
dtypes: object(2)
memory usage: 87.2+ KB
```

In [7]:
```
messages.describe()
```

Out[7]:

	label	message
count	5572	5572
unique	2	5169
top	ham	Sorry, I'll call later
freq	4825	30

In [8]:
```
messages.groupby('label').describe().T
```

Out[8]:

	label	ham	spam
message	**count**	4825	747
	unique	4516	653
	top	Sorry, I'll call later	Please call our customer service representativ...
	freq	30	4

In [9]:
```
messages['length'] = messages['message'].apply(len)
messages.head()
```

Out[9]:

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

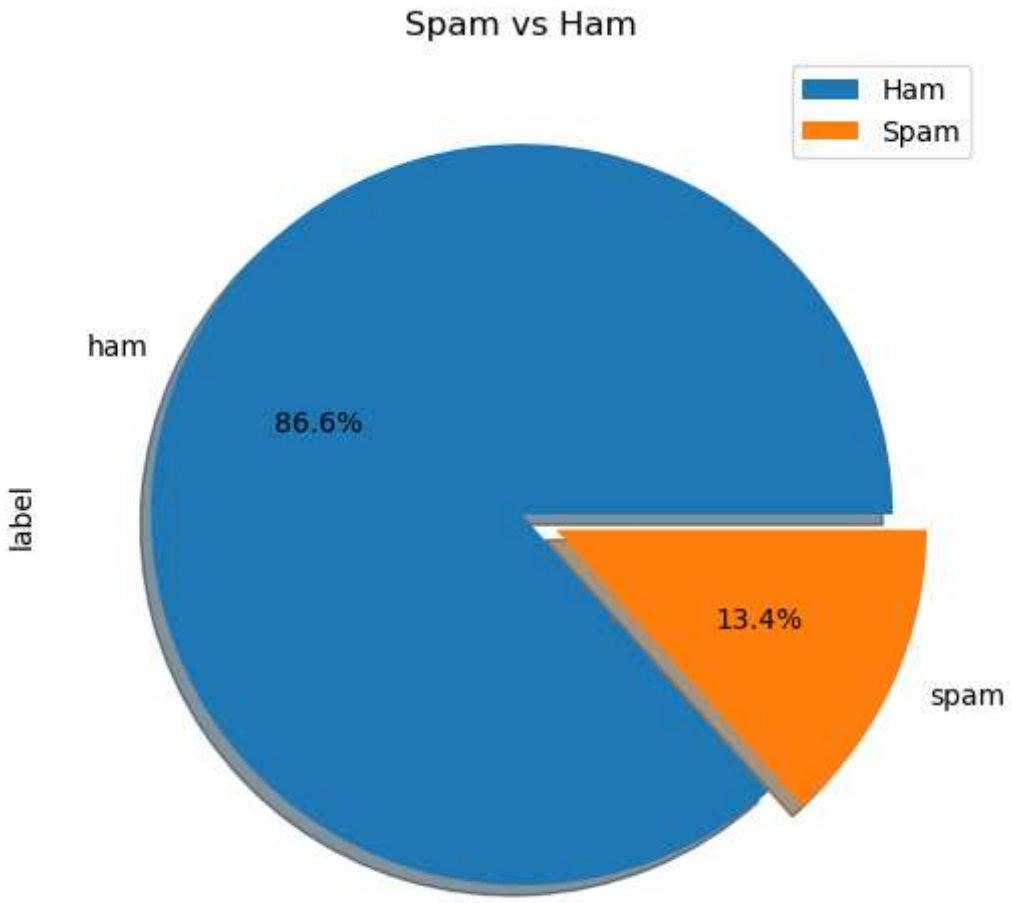
In [10]:
```
messages['message'].value_counts().rename_axis(['message']).reset_index(name='coun
```

Out[10]:

	message	counts
0	Sorry, I'll call later	30
1	I cant pick the phone right now. Pls send a me...	12
2	Ok...	10
3	7 wonders in My WORLD 7th You 6th Ur style 5th...	4
4	Say this slowly.? GOD,I LOVE YOU & I NEED ...	4

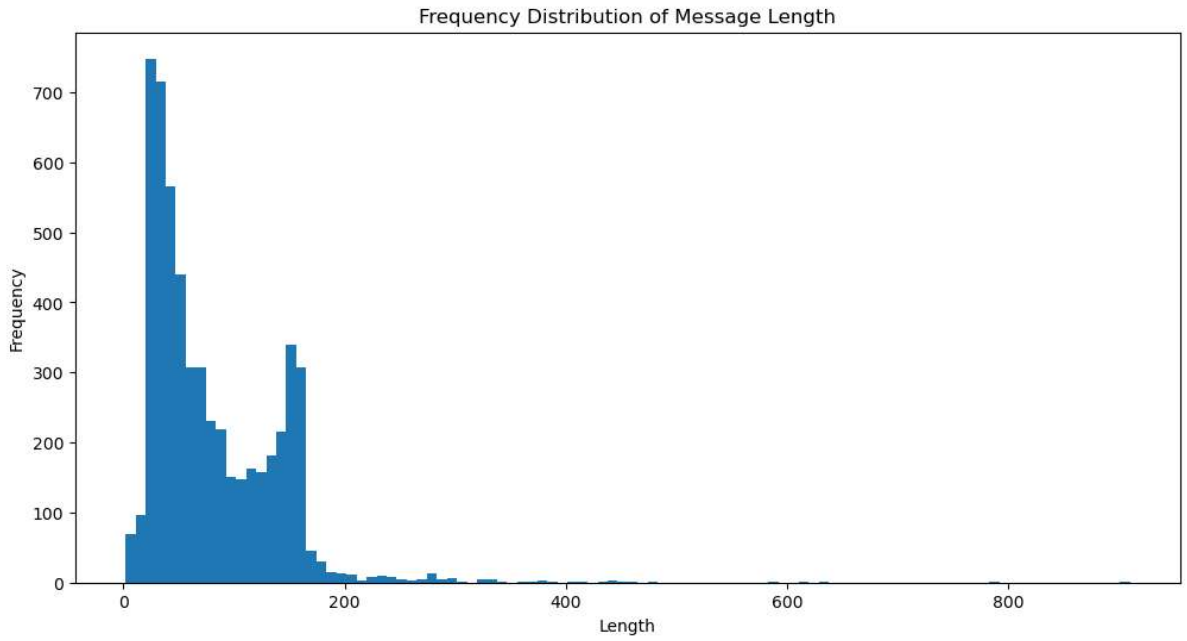
In [11]:
```
#Let's visualize
```

In [12]:
```
messages["label"].value_counts().plot(kind = 'pie',explode=[0, 0.1],figsize=(6, 6)
plt.title("Spam vs Ham")
plt.legend(["Ham", "Spam"])
plt.show()
```



In [13]:
```
plt.figure(figsize=(12,6))
messages['length'].plot(bins=100, kind='hist')
plt.title("Frequency Distribution of Message Length")
plt.xlabel("Length")
plt.ylabel("Frequency")
```

Out[13]:
```
Text(0, 0.5, 'Frequency')
```



In [14]:
```
messages['length'].describe()
```

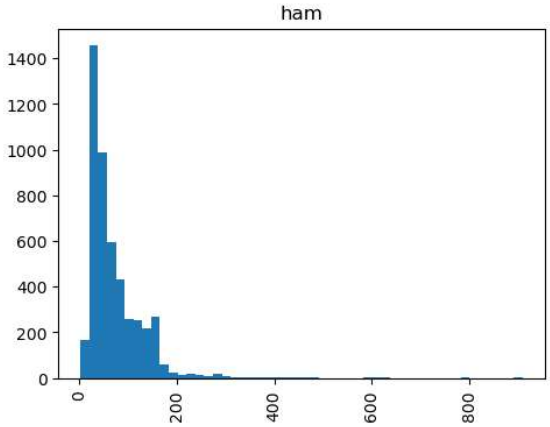
Out[14]:
```
count    5572.000000
mean       80.118808
std        59.690841
min         2.000000
25%        36.000000
50%        61.000000
75%       121.000000
max       910.000000
Name: length, dtype: float64
```

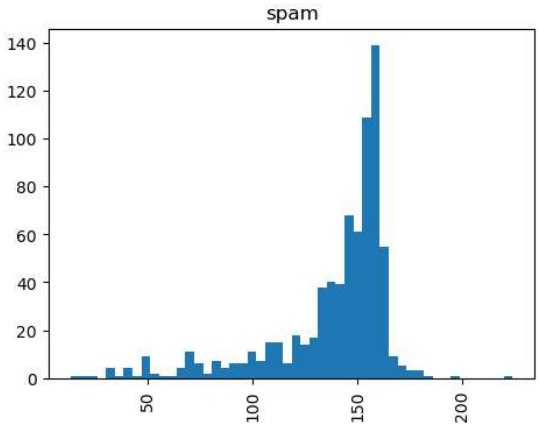
In [15]:
```
messages[messages['length'] == 910]['message'].iloc[0]
```

Out[15]:
```
"For me the love should start with attraction.i should feel that I need her every
time around me.she should be the first thing which comes in my thoughts.I would st
art the day and end it with her.she should be there every time I dream.love will b
e then when my every breath has her name.my life should happen around her.my life
will be named to her.I would cry for her.will give all my happiness and take all h
er sorrows.I will be ready to fight with anyone for her.I will be in love when I w
ill be doing the craziest things for her.love will be when I don't have to proove
anyone that my girl is the most beautiful lady on the whole planet.I will always b
e singing praises for her.love will be when I start up making chicken curry and en
d up makiing sambar.life will be the most beautiful then.will get every morning an
d thank god for the day because she is with me.I would like to say a lot..will tel
l later.."
```

In [16]:
```
messages.hist(column='length', by='label', bins=50,figsize=(12,4))
```

Out[16]:
```
array([<AxesSubplot:title={'center':'ham'}>,
       <AxesSubplot:title={'center':'spam'}>], dtype=object)
```





In [29]:
```
#Text Pre-processing
!pip install wordcloud command
```

```
Collecting wordcloud
  Downloading wordcloud-1.8.2.2-cp39-cp39-win_amd64.whl (153 kB)
     ------------------------------------ 153.1/153.1 kB 1.1 MB/s eta 0:00:00
Collecting command
  Downloading Command-0.1.0.tar.gz (5.3 kB)
  Preparing metadata (setup.py): started
  Preparing metadata (setup.py): finished with status 'done'
Requirement already satisfied: pillow in c:\users\om prakash bidhar\anaconda3\lib
\site-packages (from wordcloud) (9.2.0)
Requirement already satisfied: numpy>=1.6.1 in c:\users\om prakash bidhar\anaconda
3\lib\site-packages (from wordcloud) (1.21.5)
Requirement already satisfied: matplotlib in c:\users\om prakash bidhar\anaconda3
\lib\site-packages (from wordcloud) (3.5.2)
Requirement already satisfied: setuptools in c:\users\om prakash bidhar\anaconda3
\lib\site-packages (from command) (63.4.1)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\om prakash bidhar\ana
conda3\lib\site-packages (from matplotlib->wordcloud) (1.4.2)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\om prakash bidhar\anac
onda3\lib\site-packages (from matplotlib->wordcloud) (3.0.9)
Requirement already satisfied: cycler>=0.10 in c:\users\om prakash bidhar\anaconda
3\lib\site-packages (from matplotlib->wordcloud) (0.11.0)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\om prakash bidhar
\anaconda3\lib\site-packages (from matplotlib->wordcloud) (2.8.2)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\om prakash bidhar\ana
conda3\lib\site-packages (from matplotlib->wordcloud) (4.25.0)
Requirement already satisfied: packaging>=20.0 in c:\users\om prakash bidhar\anaco
nda3\lib\site-packages (from matplotlib->wordcloud) (21.3)
Requirement already satisfied: six>=1.5 in c:\users\om prakash bidhar\anaconda3\li
b\site-packages (from python-dateutil>=2.7->matplotlib->wordcloud) (1.16.0)
Building wheels for collected packages: command
  Building wheel for command (setup.py): started
  Building wheel for command (setup.py): finished with status 'done'
  Created wheel for command: filename=Command-0.1.0-py3-none-any.whl size=6410 sha
256=b2d8dc4a112ffbef50867105c6e41db53adff2f5fea7391badd9e4800093bd87
  Stored in directory: c:\users\om prakash bidhar\appdata\local\pip\cache\wheels\7
7\42\83\f4c689dc8dcbbc64185e89c5ad9afc0f4803528f26e529a148
Successfully built command
Installing collected packages: command, wordcloud
Successfully installed command-0.1.0 wordcloud-1.8.2.2
```

In [30]:
```
import seaborn as sns
from wordcloud import WordCloud
def text_preprocess(mess):
    """
    Takes in a string of text, then performs the following:
    1. Remove all punctuation
    2. Remove all stopwords
```

```
    3. Returns a list of the cleaned text
    """
    # Check characters to see if they are in punctuation
    nopunc = [char for char in mess if char not in string.punctuation]

    # Join the characters again to form the string.
    nopunc = ''.join(nopunc)
    nopunc = nopunc.lower()

    # Now just remove any stopwords and non alphabets
    nostop=[word for word in nopunc.split() if word.lower() not in stopwords.words

    return nostop
```

In [19]:
```
spam_messages = messages[messages["label"] == "spam"]["message"]
ham_messages = messages[messages["label"] == "ham"]["message"]
print("No of spam messages : ",len(spam_messages))
print("No of ham messages : ",len(ham_messages))
```

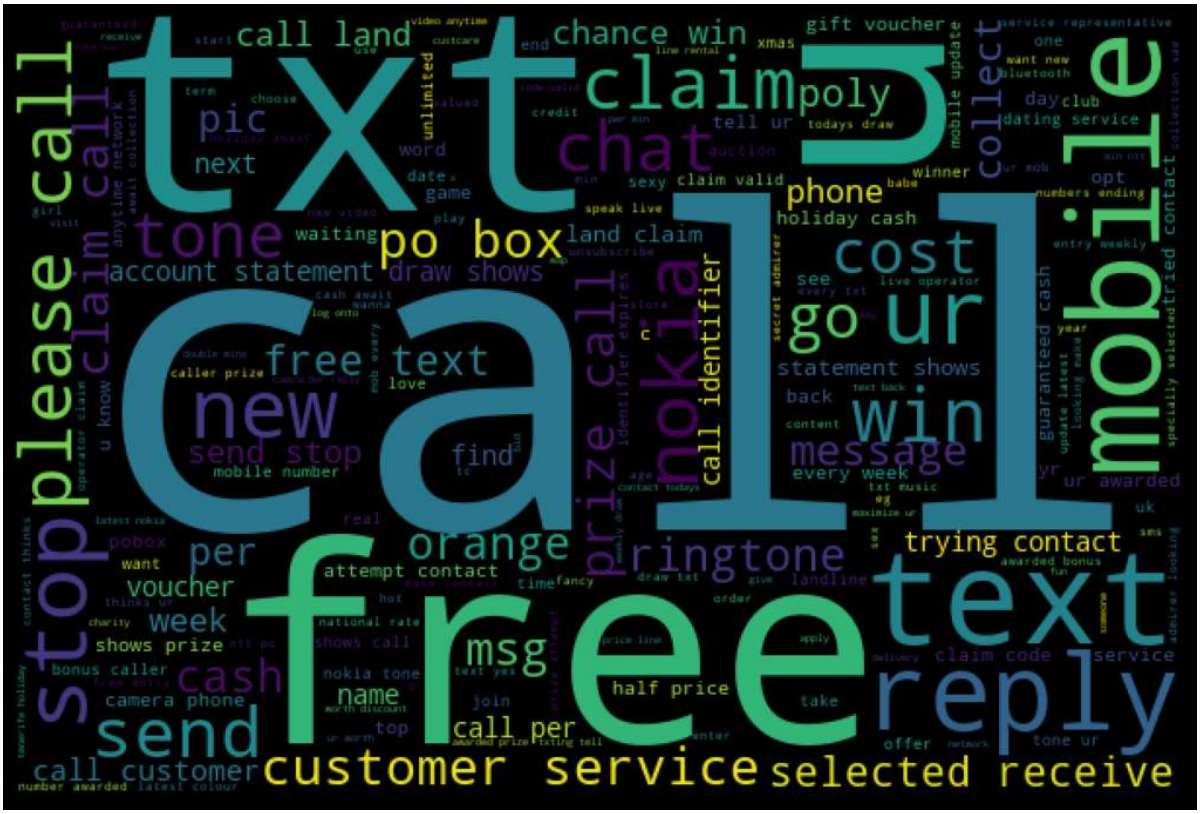
```
No of spam messages :  747
No of ham messages :  4825
```

In [20]:
```
spam_words = text_preprocess(spam_messages)
```

In [21]:
```
spam_words[:10]
```

Out[21]:
```
['free', 'entry', 'wkly', 'comp', 'win', 'fa', 'cup', 'final', 'tkts', 'may']
```

In [32]:
```
spam_wordcloud = WordCloud(width=600, height=400).generate(' '.join(spam_words))
plt.figure( figsize=(10,8), facecolor='k')
plt.imshow(spam_wordcloud)
plt.axis("off")
plt.tight_layout(pad=0)
plt.show()
```



In [24]:
```
print("Top 10 Spam words are :\n")
print(pd.Series(spam_words).value_counts().head(10))
```

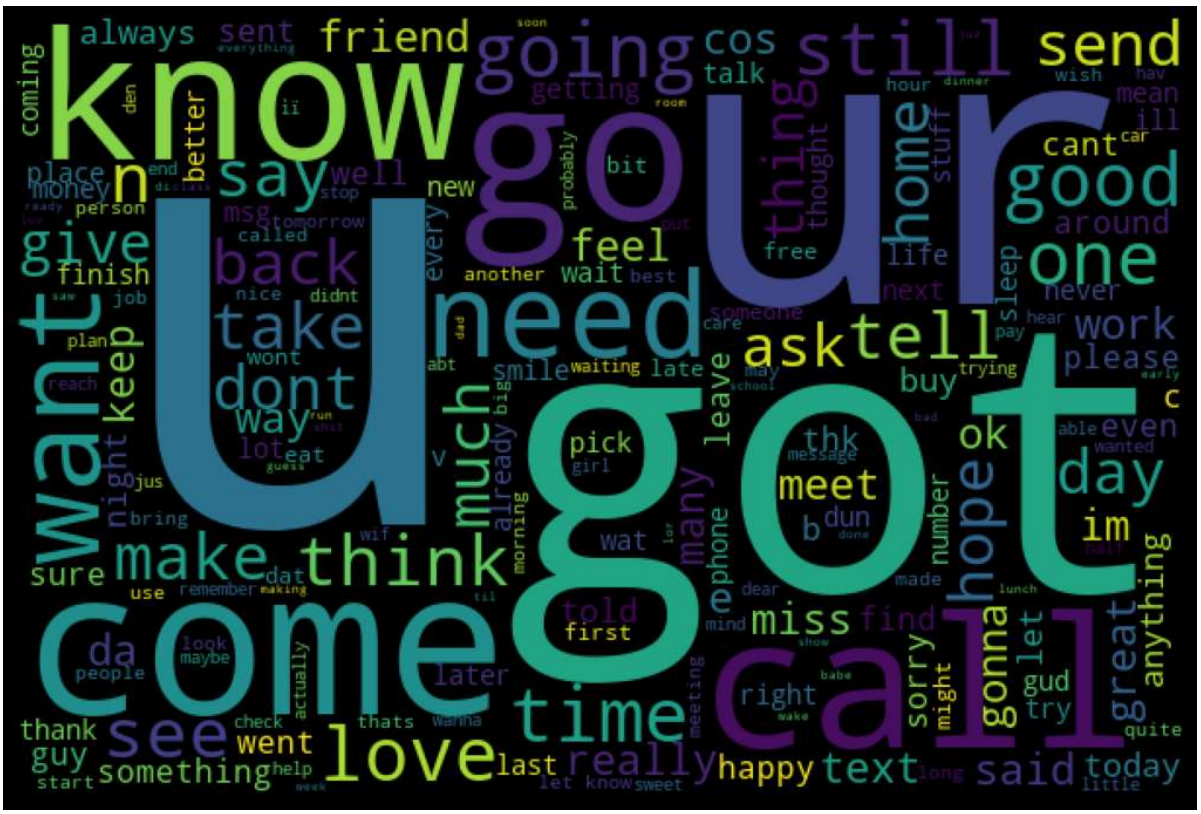
```
Top 10 Spam words are :

call      331
free      146
txt       136
ur        129
mobile    108
text      107
claim     105
u          98
reply      98
stop       78
dtype: int64
```

In [25]:
```
ham_words = text_preprocess(ham_messages)
```

In [26]:
```
ham_words[:10]
```

Out[26]:
```
['go',
 'jurong',
 'available',
 'bugis',
 'n',
 'great',
 'world',
 'la',
 'e',
 'cine']
```

In [33]:
```
ham_wordcloud = WordCloud(width=600, height=400).generate(' '.join(ham_words))
plt.figure( figsize=(10,8), facecolor='k')
plt.imshow(ham_wordcloud)
plt.axis("off")
plt.tight_layout(pad=0)
plt.show()
```



In [34]:
```
print("Top 10 Ham words are :\n")
print(pd.Series(ham_words).value_counts().head(10))
```

```
Top 10 Ham words are :

u        820
get      287
ur       235
go       231
got      216
like     215
know     202
come     201
call     200
going    151
dtype: int64
```

In [35]:
```
#data transformation
```

In [36]:
```
messages.head()
```

Out[36]:

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

In [37]:
```
messages["message"] = messages["message"].apply(text_preprocess)
```

In [38]:
```
messages["message"] = messages["message"].agg(lambda x: ' '.join(map(str, x)))
```

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

Out[40]:

	label	message	length
0	ham	go jurong point crazy available bugis n great ...	111
1	ham	ok lar joking wif u oni	29
2	spam	free entry wkly comp win fa cup final tkts may...	155
3	ham	u dun say early hor u c already say	49
4	ham	nah dont think goes usf lives around though	61

In [41]:
```
messages["message"][7]
```

Out[41]:
```
'per request melle melle oru minnaminunginte nurungu vettam set callertune callers press copy friends callertune'
```

In [43]:
```
#Creating the Bag of Words
```

In [44]:
```
vectorizer = CountVectorizer()
bow_transformer = vectorizer.fit(messages['message'])

print("20 Bag of Words (BOW) Features: \n")
print(vectorizer.get_feature_names()[20:40])

print("\nTotal number of vocab words : ",len(vectorizer.vocabulary_))
```

```
20 Bag of Words (BOW) Features:

['absence', 'absolutely', 'abstract', 'abt', 'abta', 'aburo', 'abuse', 'abusers',
'ac', 'academic', 'acc', 'accent', 'accenture', 'accept', 'access', 'accessible',
'accidant', 'accident', 'accidentally', 'accommodation']

Total number of vocab words :  8084
```

```
C:\Users\OM PRAKASH BIDHAR\anaconda3\lib\site-packages\sklearn\utils\deprecation.p
y:87: FutureWarning: Function get_feature_names is deprecated; get_feature_names i
s deprecated in 1.0 and will be removed in 1.2. Please use get_feature_names_out i
nstead.
  warnings.warn(msg, category=FutureWarning)
```

In [45]:
```
message4 = messages['message'][3]
print(message4)
```

```
u dun say early hor u c already say
```

In [46]:
```
bow4 = bow_transformer.transform([message4])
print(bow4)
print(bow4.shape)
```

```
  (0, 207)      1
  (0, 1981)     1
  (0, 2000)     1
  (0, 3141)     1
  (0, 5945)     2
(1, 8084)
```

In [47]:
```
print(bow_transformer.get_feature_names()[5945])
```

```
say
```

In [48]:
```
messages_bow = bow_transformer.transform(messages['message'])
```

In [49]:
```
print('Shape of Sparse Matrix: ', messages_bow.shape)
print('Amount of Non-Zero occurences: ', messages_bow.nnz)
```

```
Shape of Sparse Matrix:  (5572, 8084)
Amount of Non-Zero occurences:  44211
```

In [50]:
```
from sklearn.feature_extraction.text import TfidfTransformer

tfidf_transformer = TfidfTransformer().fit(messages_bow)
```

In [51]:
```
tfidf4 = tfidf_transformer.transform(bow4)
print(tfidf4)
```

```
  (0, 5945)     0.6206136295983186
  (0, 3141)     0.5139528069861297
  (0, 2000)     0.37127907512470754
  (0, 1981)     0.3420160440299522
  (0, 207)      0.3096257562744466
```

In [52]:
```
print(bow_transformer.get_feature_names()[5945])
print(bow_transformer.get_feature_names()[3141])
```

```
say
hor
```

In [53]:
```
print(tfidf_transformer.idf_[bow_transformer.vocabulary_['say']])
```

```
5.14835197309133
```

In [54]:
```
messages_tfidf = tfidf_transformer.transform(messages_bow)
print(messages_tfidf.shape)
```

```
(5572, 8084)
```

In [55]:
```
messages["message"][:10]
```

Out[55]:
```
0    go jurong point crazy available bugis n great ...
1                              ok lar joking wif u oni
2    free entry wkly comp win fa cup final tkts may...
3                u dun say early hor u c already say
4          nah dont think goes usf lives around though
5    freemsg hey darling weeks word back id like fu...
6       even brother like speak treat like aids patent
7    per request melle melle oru minnaminunginte nu...
8    winner valued network customer selected receiv...
9    mobile months u r entitled update latest colou...
Name: message, dtype: object
```

In [56]:
```
from sklearn.feature_extraction.text import TfidfVectorizer

vec = TfidfVectorizer(encoding = "latin-1", strip_accents = "unicode", stop_words =
features = vec.fit_transform(messages["message"])
print(features.shape)

print(len(vec.vocabulary_))
```

```
(5572, 7927)
7927
```

In [57]:
```
#train test split
```

In [58]:
```
msg_train, msg_test, label_train, label_test = \
train_test_split(messages_tfidf, messages['label'], test_size=0.2)
```

In [59]:
```
print("train dataset features size : ",msg_train.shape)
print("train dataset label size", label_train.shape)

print("\n")

print("test dataset features size", msg_test.shape)
print("test dataset lable size", label_test.shape)
```

```
train dataset features size :  (4457, 8084)
train dataset label size (4457,)


test dataset features size (1115, 8084)
test dataset lable size (1115,)
```

In [60]:
```
#Building Naive Bayes classifier Model
```

In [61]:
```
from sklearn.naive_bayes import MultinomialNB

clf = MultinomialNB()
spam_detect_model = clf.fit(msg_train, label_train)
```

In [62]:
```
predict_train = spam_detect_model.predict(msg_train)
```

In [63]:
```
print("Classification Report \n",metrics.classification_report(label_train, predic
print("\n")
print("Confusion Matrix \n",metrics.confusion_matrix(label_train, predict_train))
```

```
print("\n")
print("Accuracy of Train dataset : {0:0.3f}".format(metrics.accuracy_score(label_t
```

```
Classification Report
              precision    recall  f1-score   support

         ham       0.97      1.00      0.98      3852
        spam       1.00      0.80      0.89       605

    accuracy                           0.97      4457
   macro avg       0.99      0.90      0.94      4457
weighted avg       0.97      0.97      0.97      4457



Confusion Matrix
 [[3852    0]
 [ 119  486]]


Accuracy of Train dataset : 0.973
```

In [64]:
```
print('predicted:', spam_detect_model.predict(tfidf4)[0])
print('expected:', messages['label'][3])
```

```
predicted: ham
expected: ham
```

In [65]:
```
#model evaluation
```

In [66]:
```
label_predictions = spam_detect_model.predict(msg_test)
print(label_predictions)
```

```
['ham' 'ham' 'ham' ... 'ham' 'ham' 'ham']
```

In [67]:
```
print(metrics.classification_report(label_test, label_predictions))
print(metrics.confusion_matrix(label_test, label_predictions))
```

```
              precision    recall  f1-score   support

         ham       0.96      1.00      0.98       973
        spam       1.00      0.74      0.85       142

    accuracy                           0.97      1115
   macro avg       0.98      0.87      0.92      1115
weighted avg       0.97      0.97      0.96      1115

[[973   0]
 [ 37 105]]
```

In [68]:
```
print("Accuracy of the model : {0:0.3f}".format(metrics.accuracy_score(label_test,
```

```
Accuracy of the model : 0.967
```

In [ ]: