In [1]:
```
#import class from scikitlearn library
from sklearn.feature_extraction.text import CountVectorizer
```

In [4]:
```
Vectorizer = CountVectorizer()
#CountVectorizer helps tokenize the documents,it converts text to vectors by assigr
```

In [5]:
```
#create 3 documents
document1 = 'This is an amazing day day'
document2 = 'This is the best day of my life'
document3 = 'How are you you feeling today'
```

In [6]:
```
#put all the documents in a list
listofdocuments = [document1,document2,document3]
```

In [7]:
```
listofdocuments
```

Out[7]:
```
['This is an amazing day day',
 'This is the best day of my life',
 'How are you you feeling today']
```

In [8]:
```
#To actually create the vectorizer, we simply need to call fit on the listofdocumer
abcd = Vectorizer.fit(listofdocuments)
```

In [9]:
```
abcd
```

Out[9]:
```
CountVectorizer()
```

In [10]:
```
# Now, we can inspect how our vectorizer vectorized the text
# This will print out a list of unique words used, and their index in the vectors
print(Vectorizer.vocabulary_)            #with BOW the value is the freq,but with v
```

```
{'this': 12, 'is': 7, 'an': 1, 'amazing': 0, 'day': 4, 'the': 11, 'best': 3, 'of':
10, 'my': 9, 'life': 8, 'how': 6, 'are': 2, 'you': 14, 'feeling': 5, 'today': 13}
```

In [11]:
```
## If we would like to actually create a matrix of features, we can do so by passir
# text into the vectorizer to get back counts
bag_of_words = Vectorizer.transform(listofdocuments)
```

In [12]:
```
#print bag of words
#the first component in the o/p is a tuple and the second component is the freq cou
#in the tuple the first component is the document number and the second is the inde
print(bag_of_words)
```

```
  (0, 0)        1
  (0, 1)        1
  (0, 4)        2
  (0, 7)        1
  (0, 12)       1
  (1, 3)        1
  (1, 4)        1
  (1, 7)        1
  (1, 8)        1
  (1, 9)        1
  (1, 10)       1
  (1, 11)       1
  (1, 12)       1
  (2, 2)        1
  (2, 5)        1
  (2, 6)        1
  (2, 13)       1
  (2, 14)       2
```

In [13]:
```
#First we import the required libraries
import pandas as pd
import string            #the class string is downloaded from nltk corpus
from nltk.corpus import stopwords
```

In [14]:
```
#Get the spam data collection dataset which is tcsv,and the sep function takes it c
#message is the text present in the dataset,while response is the label/category of
df_spamcollection = pd.read_csv(r"C:\Users\s323\Desktop\Data Science\SpamCollectio
```

In [15]:
```
df_spamcollection.head()
```

Out[15]:

	response	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 [16]:
```
#view more information about the spam data using the describe method
#freq denotes that the most common are HAM messages of count 4825,and most common m
#unique labels are spam and ham (2),and there are 403 duplicate messages,while unic
#top or the most common is a ham message,and the message is Sorry, I'll call later
df_spamcollection.describe()
```

Out[16]:

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

In [17]:
```
#View response using group by and describe method
#in ham we have a total of 4825 messages,and only 4516 unique ham messages
#in spam we have 747 total messages and 653 unique messages.
#top mess in ham(Sorry, I'll call later),and top message in spam is (Please call ou
```

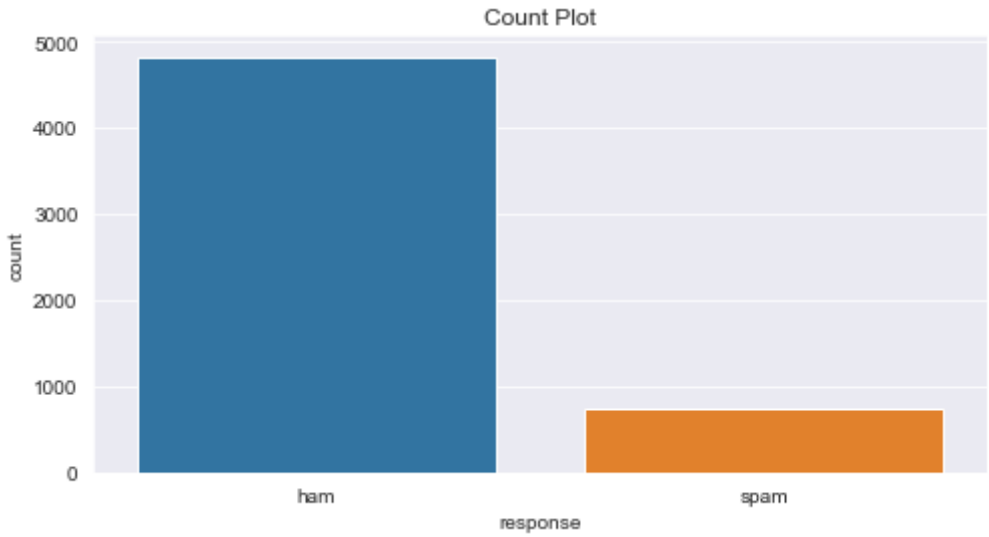
```
#The target variable is either ham or spam. There are 4825 ham messages and 747 spa
df_spamcollection.groupby('response').describe()
```

Out[17]:

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

In [18]:

```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
sns.set_style('darkgrid')
plt.figure(figsize=(8,4)) # figsize which takes a tuple as an argument that contai
sns.countplot(x='response', data=df_spamcollection)
plt.title('Count Plot')
plt.show()
```



In [20]:

```
#%matplotlib inline to enable the inline plotting, where the plots/graphs will be
```

In [21]:

```
#verify length of the messages and add it as a new column(feature).the syntax on th
#to a dataframe
df_spamcollection['length'] = df_spamcollection['message'].apply(len)
```

In [22]:

```
#view first 5 messages with length
df_spamcollection.head()
```

Out[22]:

	response	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 [23]:

```
#As we know any algorithm is good with numbers,we have to convert the text into nu
```

In [24]:
```
#importing the libraries
import string
import nltk
from nltk.corpus import stopwords
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\s323\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\stopwords.zip.
```

Out[24]:
```
True
```

In [25]:
```
#Stopwords
import nltk
from nltk.corpus import stopwords
stopwords_nltk = stopwords.words('english')
print(stopwords_nltk)
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "yo
u'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', 'we
re', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did',
'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'wh
ile', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'thr
ough', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down',
'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'he
re', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few',
'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'sam
e', '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', "had
n't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "might
n't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "should
n't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"]
```

In [26]:
```
#Print length
print(len(stopwords_nltk))
```

```
179
```

In [ ]:
```

```