Natural Language Processing using NLTK on University California Irvine (UCI) dataset : Spam Detection Filter

Import nltk

In [1]:
```
import nltk
```

In [2]:
```
nltk.download_shell()
```

```
NLTK Downloader
---------------------------------------------------------------------------
    d) Download   l) List    u) Update   c) Config   h) Help   q) Quit
---------------------------------------------------------------------------
Downloader> stopwords
Command 'stopwords' unrecognized

---------------------------------------------------------------------------
    d) Download   l) List    u) Update   c) Config   h) Help   q) Quit
---------------------------------------------------------------------------
Downloader> d

Download which package (l=list; x=cancel)?
  Identifier> stopwords
    Downloading package stopwords to
        C:\Users\User.DESKTOP-3HHGVTH\AppData\Roaming\nltk_data...
      Package stopwords is already up-to-date!
---------------------------------------------------------------------------
    d) Download   l) List    u) Update   c) Config   h) Help   q) Quit
---------------------------------------------------------------------------
Downloader> q
```

Strip the dataset

In [3]:
```
messages = [line.rstrip() for line in open('smsspamcollection/SMSSpamCollection') ]
```

In [4]:
```
messages[50]
```

Out[4]:
```
'ham\tWhat you thinked about me. First time you saw me in class.'
```

In [5]:
```
messages[0]
```

Out[5]:
```
'ham\tGo until jurong point, crazy.. Available only in bugis n great world la e buffe
t... Cine there got amore wat...'
```

In [6]:
```
print(len(messages))
```

```
5574
```

In [7]:
```
for msg_no, message in enumerate(messages[:10]):
    print(msg_no, message)
```

```
0 ham   Go until jurong point, crazy.. Available only in bugis n great world la e buffe
t... Cine there got amore wat...
```

```
1 ham   Ok lar... Joking wif u oni...
2 spam  Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 8
7121 to receive entry question(std txt rate)T&C's apply 08452810075over18's
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 around here though
5 spam  FreeMsg Hey there darling it's been 3 week's now and no word back! I'd like some
fun you up for it still? Tb ok! XxX std chgs to send, Â£1.50 to rcv
6 ham   Even my brother is not like to speak with me. They treat me like aids patent.
7 ham   As per your request 'Melle Melle (Oru Minnaminunginte Nurungu Vettam)' has been
set as your callertune for all Callers. Press *9 to copy your friends Callertune
8 spam  WINNER!! As a valued network customer you have been selected to receivea Â£900 p
rize reward! To claim call 09061701461. Claim code KL341. Valid 12 hours only.
9 spam  Had your mobile 11 months or more? U R entitled to Update to the latest colour m
obiles with camera for Free! Call The Mobile Update Co FREE on 08002986030
```

Load dataset into a pandas DataFrame

In [8]:
```
import pandas as pd
```

In [9]:
```
df = pd.read_csv('smsspamcollection/SMSSpamCollection', sep='\t', names = ['Label', 'Me
```

In [10]:
```
df.head()
```

Out[10]:

	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 [11]:
```
df.describe()
```

Out[11]:

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

In [12]:
```
df['Label'].describe()
```

Out[12]:
```
count     5572
unique       2
top        ham
freq      4825
Name: Label, dtype: object
```

Do Feature Engineering

In [13]:
```
df.groupby('Label').describe()
```

Out[13]:

	Message

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

In [14]:
```
df['length'] = df['Message'].apply(len)
```

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

Out[15]:

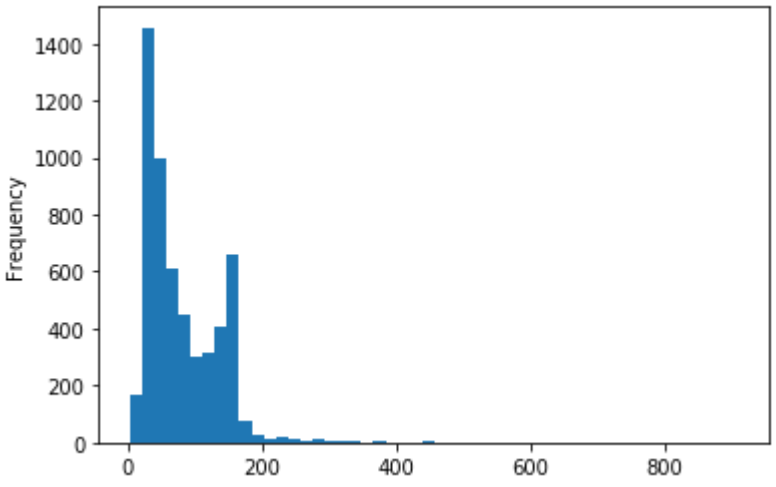
	Label	Message	length
0	ham	Go until jurong point, crazy.. Available only ...	111
1	ham	Ok lar... Joking wif u oni...	29
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	155
3	ham	U dun say so early hor... U c already then say...	49
4	ham	Nah I don't think he goes to usf, he lives aro...	61

Visualize message length to get an idea of spam vs normal

In [16]:
```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

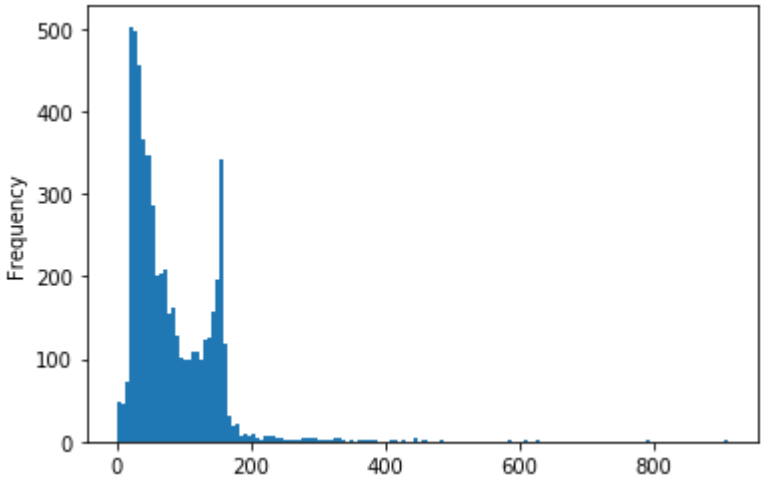
In [17]:
```
df['length'].plot.hist(bins=50)
```

Out[17]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1b4b1161588>
```



In [18]:
```
df['length'].plot.hist(bins=150)
```

Out[18]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1b4b9985ec8>
```



In [19]:
```
df['Message'].describe()
```

Out[19]:
```
count                    5572
unique                   5169
top       Sorry, I'll call later
freq                       30
Name: Message, dtype: object
```

In [20]:
```
df['length'].describe()
```

Out[20]:
```
count    5572.000000
mean       80.489950
std        59.942907
min         2.000000
25%        36.000000
50%        62.000000
75%       122.000000
max       910.000000
Name: length, dtype: float64
```

In [21]:
```
df[df['length']==910]
```

Out[21]:

	Label	Message	length
1085	ham	For me the love should start with attraction.i...	910

In [22]:
```
df[df['length']==2]
```

Out[22]:

	Label	Message	length
1925	ham	Ok	2
3051	ham	Ok	2
4498	ham	Ok	2
5357	ham	Ok	2

In [23]:
```
df[df['length']==36]
```

Out[23]:

	Label	Message	length
83	ham	You will be in the place of that man	36

	Label	Message	length
258	ham	Where are you lover ? I need you ...	36
379	ham	Keep my payasam there if rinu brings	36
403	ham	The hair cream has not been shipped.	36
438	ham	How long does applebees fucking take	36
...	...	...	...
5320	ham	But we havent got da topic yet rite?	36
5389	ham	Ok.ok ok..then..whats ur todays plan	36
5411	ham	I ask if u meeting da ge tmr nite...	36
5502	ham	Apo all other are mokka players only	36
5568	ham	Will ü b going to esplanade fr home?	36

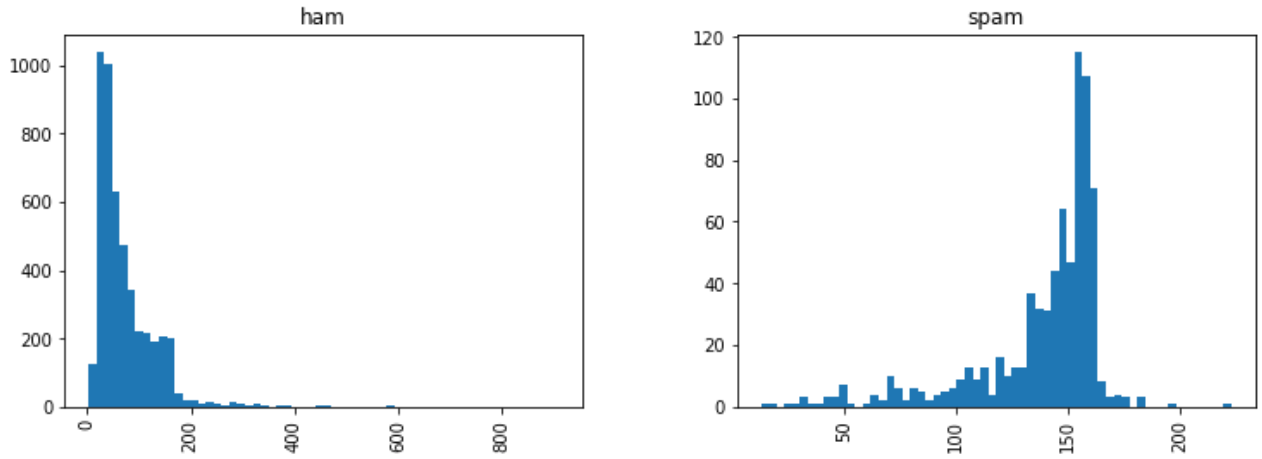
68 rows × 3 columns

In [24]:
```
df[df['length']==910]['Message'].iloc[0]
```

Out[24]:
```
"For me the love should start with attraction.i should feel that I need her every time a
round me.she should be the first thing which comes in my thoughts.I would start the day
and end it with her.she should be there every time I dream.love will be then when my eve
ry breath has her name.my life should happen around her.my life will be named to her.I w
ould cry for her.will give all my happiness and take all her sorrows.I will be ready to
fight with anyone for her.I will be in love when I will 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 l
ady on the whole planet.I will always be singing praises for her.love will be when I sta
rt up making chicken curry and end up makiing sambar.life will be the most beautiful the
n.will get every morning and thank god for the day because she is with me.I would like t
o say a lot..will tell later.."
```

In [25]:
```
df.hist(column='length', by='Label', bins=60, figsize=(12,4))
```

Out[25]:
```
array([<matplotlib.axes._subplots.AxesSubplot object at 0x000001B4B9BBA748>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x000001B4B9BD0E88>],
      dtype=object)
```



Converting Raw messages into Vectors for further processing

In [26]:
```
import string
```

Remove Punctuation

In [27]:
```
string.punctuation
```

Out[27]:
```
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
```

In [28]:
```
mess = 'This is a sample message, with! a lot of : punctuation marks.'
```

In [29]:
```
nopunc = [p for p in mess if p not in string.punctuation]
```

In [30]:
```
nopunc
```

Out[30]:
```
['T',
 'h',
 'i',
 's',
 ' ',
 'i',
 's',
 ' ',
 'a',
 ' ',
 's',
 'a',
 'm',
 'p',
 'l',
 'e',
 ' ',
 'n',
 'e',
 's',
 's',
 'a',
 'g',
 'e',
 ' ',
 'w',
 'i',
 't',
 'h',
 ' ',
 'a',
 ' ',
 'l',
 'o',
 't',
 ' ',
 'o',
 'f',
 ' ',
 ' ',
 'p',
 'u',
 'n',
 'c',
 't',
 'u',
 'a',
 't',
 'i',
```

```
 'o',
 'n',
 ' ',
 'm',
 'a',
 'r',
 'k',
 's']
```

Import Stopwords from Nltk corpus

In [31]:
```
from nltk.corpus import stopwords
```

In [32]:
```
stopwords.words('english')
```

Out[32]:
```
['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"]
```

Join the letters in the list to form a sentence without punctuation

In [33]:
```
nopunc = ''.join(nopunc)
```

In [34]:
```
type(nopunc)
```

Out[34]:
```
str
```

In [35]:
```
nopunc.split()
```

Out[35]:
```
['This',
 'is',
 'a',
 'sample',
 'nessage',
 'with',
 'a',
 'lot',
 'of',
 'punctuation',
 'marks']
```

Remove stopwords

In [36]:
```
clean_mess = [word for word in nopunc.split() if word.lower() not in stopwords.words('e
```

In [37]:
```
clean_mess
```

Out[37]:
```
['sample', 'nessage', 'lot', 'punctuation', 'marks']
```

In [55]:
```
def text_process(text):
    '''
    1. Remove punctuation
    2. Remove stop words
    3. Return clean data
    '''
    nopunc = [word for word in text if word not in string.punctuation]
    nopunc = ''.join(nopunc)

    return [word for word in nopunc.split() if word.lower() not in stopwords.words('eng
```

Tokenize text

In [56]:
```
new_msg = 'This is a simple text. to check for! various puncs. And stopwords!.'
```

In [57]:
```
text_process(new_msg)
```

Out[57]:
```
['simple', 'text', 'check', 'various', 'puncs', 'stopwords']
```

In [58]:
```
df.head()
```

Out[58]:

	Label	Message	length
0	ham	Go until jurong point, crazy.. Available only ...	111
1	ham	Ok lar... Joking wif u oni...	29

	Label	Message	length
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

Apply the created method for text tokenization

In [60]:
```
df['Message'].head(5).apply(text_process)
```

Out[60]:
```
0    [Go, jurong, point, crazy, Available, bugis, n...
1                        [Ok, lar, Joking, wif, u, oni]
2    [Free, entry, 2, wkly, comp, win, FA, Cup, fin...
3        [U, dun, say, early, hor, U, c, already, say]
4    [Nah, dont, think, goes, usf, lives, around, t...
Name: Message, dtype: object
```

Due to multiple shorthand texts, stemming wont be much fruitful. So next we convert our list of words to Vectors

Perform following steps using the Bag of Words Model: 1.Term Frequency: Count how many times a word occurs in each message 2.Inverse Document Frequency: Weigh the counts so that frequent tokens get lower weight 3.L2 Norm: Normalize the vectors to unit length, to abstract from the original text length

Using Scikit-Learn's Countvectorizer model to convert a collection of text documents to a matrix of text counts

In [61]:
```
#Sparse matrix: Matrix in which most of the elements are zero
```

In [62]:
```
from sklearn.feature_extraction.text import CountVectorizer
```

In [63]:
```
bag_of_words_transformer = CountVectorizer(analyzer=text_process)
```

In [64]:
```
bag_of_words_transformer.fit(df['Message'])
```

Out[64]:
```
CountVectorizer(analyzer=<function text_process at 0x000001B4BD7C6B88>,
                binary=False, decode_error='strict',
                dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',
                lowercase=True, max_df=1.0, max_features=None, min_df=1,
                ngram_range=(1, 1), preprocessor=None, stop_words=None,
                strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
                tokenizer=None, vocabulary=None)
```

In [65]:
```
print(len(bag_of_words_transformer.vocabulary_))
```

```
11425
```

In [66]:
```
sample = df['Message'][3]
```

In [67]:
```
sample
```

Out[67]:
```
'U dun say so early hor... U c already then say...'
```

In [69]:
```
bowt_sample = bag_of_words_transformer.transform([sample])
```

In [74]:
```
print(bowt_sample, "\n\n", bowt_sample.shape)
```

```
  (0, 4068)     2
  (0, 4629)     1
  (0, 5261)     1
  (0, 6204)     1
  (0, 6222)     1
  (0, 7186)     1
  (0, 9554)     2

 (1, 11425)
```

In [77]:
```
bag_of_words_transformer.get_feature_names()[4068]
```

Out[77]:
```
'U'
```

In [78]:
```
bag_of_words_transformer.get_feature_names()[9554]
```

Out[78]:
```
'say'
```

Transform entire dataframe of messages into Sparse Matrix

In [79]:
```
messages_bag_of_words = bag_of_words_transformer.transform(df['Message'])
```

In [80]:
```
print('Shape of Sparse Matrix: ', messages_bag_of_words.shape)
```

```
Shape of Sparse Matrix:  (5572, 11425)
```

Check amount of non-zero occurences

In [81]:
```
messages_bag_of_words.nnz
```

Out[81]:
```
50548
```

Convert Word Count into TermFrequency Inverse Document Frequency[tfidf]

In [82]:
```
from sklearn.feature_extraction.text import TfidfTransformer
```

In [83]:
```
tfidf_transformer = TfidfTransformer()
```

In [84]:
```
tfidf_transformer.fit(messages_bag_of_words)
```

Out[84]:
```
TfidfTransformer(norm='l2', smooth_idf=True, sublinear_tf=False, use_idf=True)
```

In [85]:
```
sample_tfidf = tfidf_transformer.transform(bowt_sample)
```

In [86]:
```
print(sample_tfidf)
```

```
  (0, 9554)     0.5385626262927564
  (0, 7186)     0.4389365653379857
  (0, 6222)     0.3187216892949149
  (0, 6204)     0.29953799723697416
  (0, 5261)     0.29729957405868723
  (0, 4629)     0.26619801906087187
  (0, 4068)     0.40832589933384067
```

In [87]:
```
tfidf_transformer.idf_[bag_of_words_transformer.vocabulary_['university']]
```

Out[87]:
```
8.527076498901426
```

Convert entire Bag of Words Corpus into a TFIDF Corpus

In [90]:
```
messages_tfidf = tfidf_transformer.transform(messages_bag_of_words)
```

Using Naive-Bayes classification algorithm

In [91]:
```
from sklearn.naive_bayes import MultinomialNB
```

In [92]:
```
spam_detection_model = MultinomialNB()
```

In [93]:
```
spam_detection_model.fit(messages_tfidf,df['Label'])
```

Out[93]:
```
MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
```

Split Dataset into Train Test split

In [94]:
```
from sklearn.model_selection import train_test_split
```

In [95]:
```
msg_train,msg_test,label_train,label_test = train_test_split(df['Message'],df['Label'],
```

Use Scikit-Learn Pipeline

In [96]:
```
from sklearn.pipeline import Pipeline
```

In [97]:
```
pipeline = Pipeline(
[
    ('bagofwords',CountVectorizer(analyzer=text_process)),
    ('tfidf',TfidfTransformer()),
    ('classifier',MultinomialNB())
])
```

In [98]:
```
pipeline.fit(msg_train,label_train)
```

Out[98]:
```
Pipeline(memory=None,
         steps=[('bagofwords',
                 CountVectorizer(analyzer=<function text_process at 0x000001B4BD7C6B88>,
                                 binary=False, decode_error='strict',
                                 dtype=<class 'numpy.int64'>, encoding='utf-8',
                                 input='content', lowercase=True, max_df=1.0,
                                 max_features=None, min_df=1,
                                 ngram_range=(1, 1), preprocessor=None,
                                 stop_words=None, strip_accents=None,
                                 token_pattern='(?u)\\b\\w\\w+\\b',
                                 tokenizer=None, vocabulary=None)),
                ('tfidf',
                 TfidfTransformer(norm='l2', smooth_idf=True,
                                  sublinear_tf=False, use_idf=True)),
                ('classifier',
                 MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True))],
         verbose=False)
```

In [99]:
```
predictions = pipeline.predict(msg_test)
```

Model Evaluation and Reporting Metrics

In [100…
```
from sklearn.metrics import classification_report, confusion_matrix
```

In [101…
```
print(classification_report(label_test,predictions))
print('\n')
print(confusion_matrix(label_test,predictions))
```

```
              precision    recall  f1-score   support

         ham       0.96      1.00      0.98      1453
        spam       1.00      0.70      0.82       219

    accuracy                           0.96      1672
   macro avg       0.98      0.85      0.90      1672
weighted avg       0.96      0.96      0.96      1672



[[1453    0]
 [  66  153]]
```

Comparing Naives Bayes classification to Random Forest Classifier

In [102…
```
from sklearn.ensemble import RandomForestClassifier
```

In [103…
```
pipeline = Pipeline(
[
    ('bagofwords',CountVectorizer(analyzer=text_process)),
    ('tfidf',TfidfTransformer()),
    ('classifier',RandomForestClassifier())
])
```

In [104…
```
pipeline.fit(msg_train,label_train)
```

```
e:\users\user.desktop-3hhgvth\anaconda3\envs\nlpenv\lib\site-packages\sklearn\ensemble\forest.py:245: FutureWarning: The default value of n_estimators will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

Out[104…
```
Pipeline(memory=None,
         steps=[('bagofwords',
                 CountVectorizer(analyzer=<function text_process at 0x000001B4BD7C6B88>,
                                 binary=False, decode_error='strict',
                                 dtype=<class 'numpy.int64'>, encoding='utf-8',
                                 input='content', lowercase=True, max_df=1.0,
                                 max_features=None, min_df=1,
                                 ngram_range=(1, 1), preprocessor=None,
                                 stop_words=None, strip_accents=None,
                                 token_pattern='(?u)\\...
                 RandomForestClassifier(bootstrap=True, class_weight=None,
                                        criterion='gini', max_depth=None,
                                        max_features='auto',
                                        max_leaf_nodes=None,
                                        min_impurity_decrease=0.0,
                                        min_impurity_split=None,
                                        min_samples_leaf=1, min_samples_split=2,
                                        min_weight_fraction_leaf=0.0,
                                        n_estimators=10, n_jobs=None,
                                        oob_score=False, random_state=None,
                                        verbose=0, warm_start=False))],
         verbose=False)
```

In [105…
```
predictions = pipeline.predict(msg_test)
```

Model Evaluation and reporting for Random Forest Algorithm

In [106…
```
print(classification_report(label_test,predictions))
print('\n')
print(confusion_matrix(label_test,predictions))
```

```
              precision    recall  f1-score   support

         ham       0.96      1.00      0.98      1453
        spam       1.00      0.75      0.86       219

    accuracy                           0.97      1672
   macro avg       0.98      0.88      0.92      1672
weighted avg       0.97      0.97      0.97      1672



[[1453    0]
 [  54  165]]
```

In [ ]: