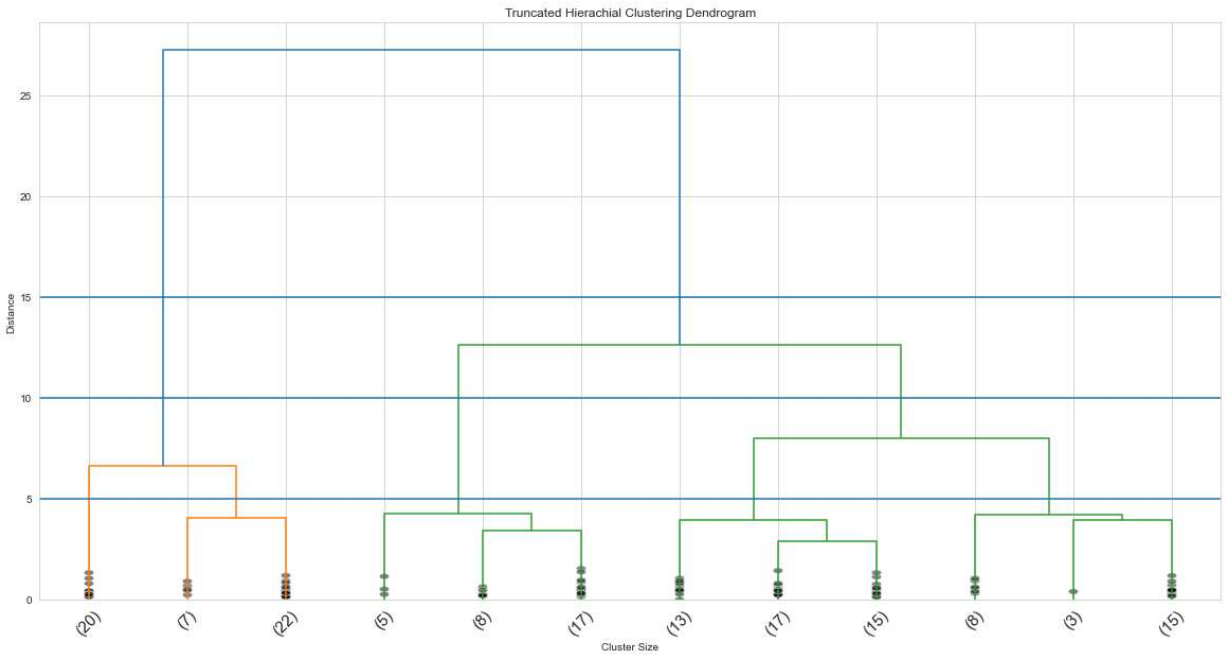
In [1]:
```
import numpy as np
import pandas as pd

import scipy
from scipy.cluster.hierarchy import dendrogram,linkage
from scipy.cluster.hierarchy import fcluster
from scipy.cluster.hierarchy import cophenet
from scipy.spatial.distance import pdist

import matplotlib.pyplot as plt
from pylab import rcParams
import seaborn as sb

import sklearn
from sklearn import datasets
from sklearn.cluster import AgglomerativeClustering
import sklearn.metrics as sm
from sklearn.preprocessing import scale
```

In [2]:
```
#Configure the output
np.set_printoptions(precision=4,suppress=True)
%matplotlib inline
rcParams["figure.figsize"] =20,10
sb.set_style("whitegrid")
```

In [3]:
```
iris = datasets.load_iris()
#scale the data
data = scale(iris.data)
target = pd.DataFrame(iris.target)
variable_names = iris.feature_names
data[0:10]
```

Out[3]:
```
array([[-0.9007,  1.019 , -1.3402, -1.3154],
       [-1.143 , -0.132 , -1.3402, -1.3154],
       [-1.3854,  0.3284, -1.3971, -1.3154],
       [-1.5065,  0.0982, -1.2834, -1.3154],
       [-1.0218,  1.2492, -1.3402, -1.3154],
       [-0.5372,  1.9398, -1.1697, -1.0522],
       [-1.5065,  0.7888, -1.3402, -1.1838],
       [-1.0218,  0.7888, -1.2834, -1.3154],
       [-1.7489, -0.3622, -1.3402, -1.3154],
       [-1.143 ,  0.0982, -1.2834, -1.4471]])
```

In [4]:
```
z = linkage(data,"ward")
#generate dendrogram
dendrogram(z,truncate_mode= "lastp", p =12, leaf_rotation=45,leaf_font_size=15, s
plt.title("Truncated Hierachial Clustering Dendrogram")
plt.xlabel("Cluster Size")
plt.ylabel("Distance")
#divide the cluster
plt.axhline(y=15)
plt.axhline(5)
plt.axhline(10)
plt.show()
```



In [5]:
```
#based on the dendrogram we have two clusetes
k =3
#build the model
HClustering = AgglomerativeClustering(n_clusters=k , affinity="euclidean",linkage
#fit the model on the dataset
HClustering.fit(data)
#accuracy of the model
sm.accuracy_score(target,HClustering.labels_)
```

Out[5]: 0.013333333333333334

In [6]:
```
#based on the dendrogram we have two clusetes
k =3
#build the model
HClustering = AgglomerativeClustering(n_clusters=k , affinity="euclidean",linkage
#fit the model on the dataset
HClustering.fit(data)
#accuracy of the model
sm.accuracy_score(target,HClustering.labels_)
```

Out[6]: 0.013333333333333334

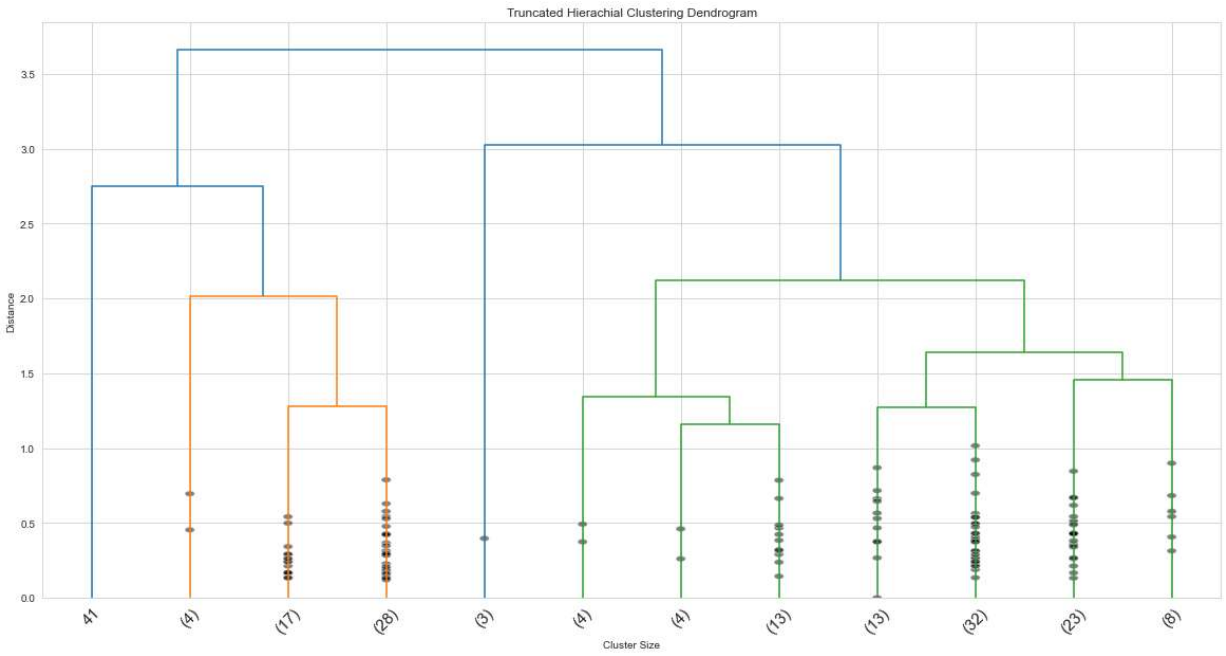
In [7]:
```
#based on the dendrogram we have two clusetes
k =3
#build the model
HClustering = AgglomerativeClustering(n_clusters=k , affinity="euclidean",linkage
#fit the model on the dataset
HClustering.fit(data)
#accuracy of the model
sm.accuracy_score(target,HClustering.labels_)
```

Out[7]: 0.6866666666666666

In [8]:
```
#based on the dendrogram we have two clusetes
k =3
#build the model
HClustering = AgglomerativeClustering(n_clusters=k , affinity="euclidean",linkage
#fit the model on the dataset
HClustering.fit(data)
#accuracy of the model
sm.accuracy_score(target,HClustering.labels_)
```

Out[8]: 0.0

In [9]:
```
z = linkage(data,"average")
#generate dendrogram
dendrogram(z,truncate_mode= "lastp", p =12, leaf_rotation=45,leaf_font_size=15, s
plt.title("Truncated Hierachial Clustering Dendrogram")
plt.xlabel("Cluster Size")
plt.ylabel("Distance")
#divide the cluster
plt.axhline(y=15)
plt.axhline(5)
plt.axhline(10)
plt.show()
```



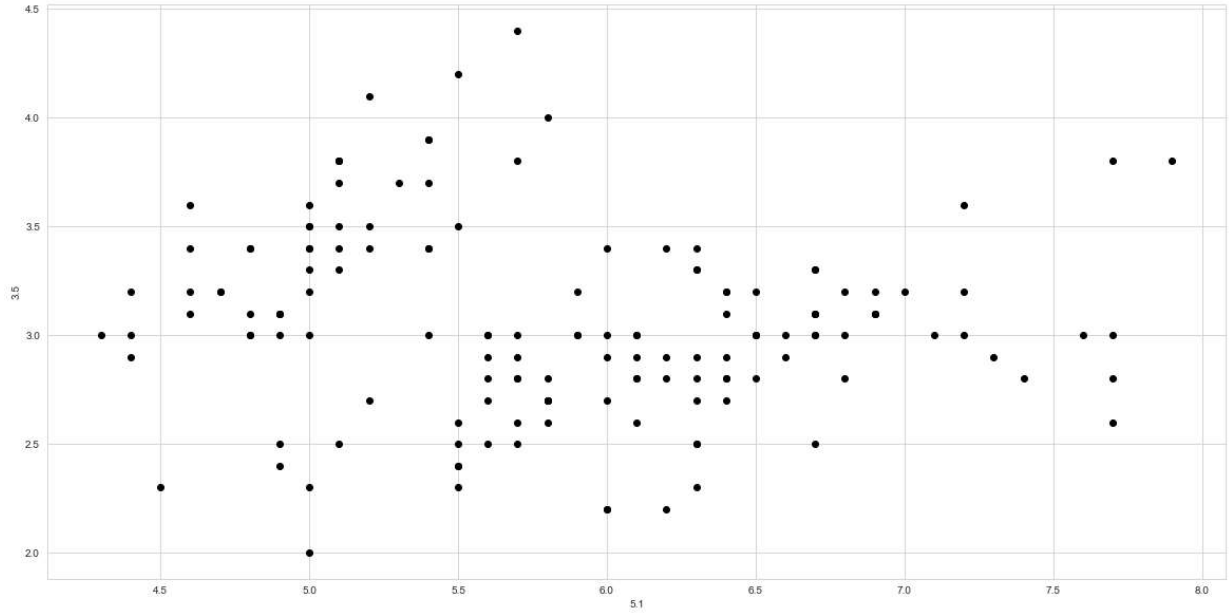
In [10]:
```
import random as rd
```

In [11]:
```
data = pd.read_csv('E:\iris.data')
data.head()
```

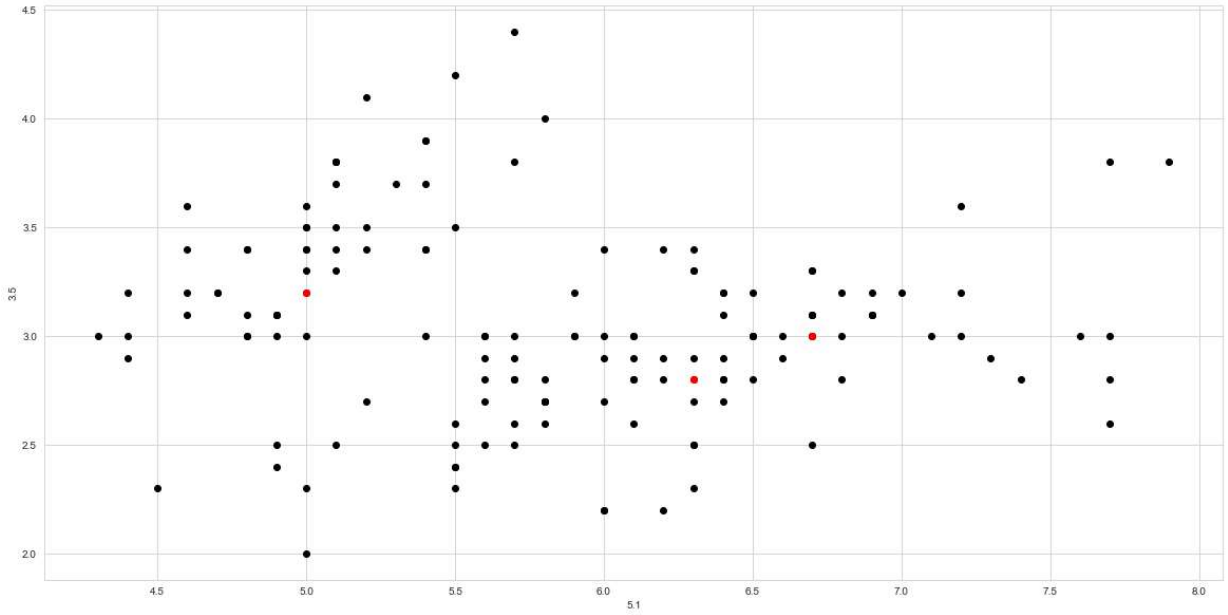
Out[11]:

	5.1	3.5	1.4	0.2	Iris-setosa
0	4.9	3.0	1.4	0.2	Iris-setosa
1	4.7	3.2	1.3	0.2	Iris-setosa
2	4.6	3.1	1.5	0.2	Iris-setosa
3	5.0	3.6	1.4	0.2	Iris-setosa
4	5.4	3.9	1.7	0.4	Iris-setosa

In [12]:
```
X = data[["5.1","3.5"]]
#Visualise data points
plt.scatter(X["5.1"],X["3.5"],c='black')
plt.xlabel('5.1')
plt.ylabel('3.5')
plt.show()
```



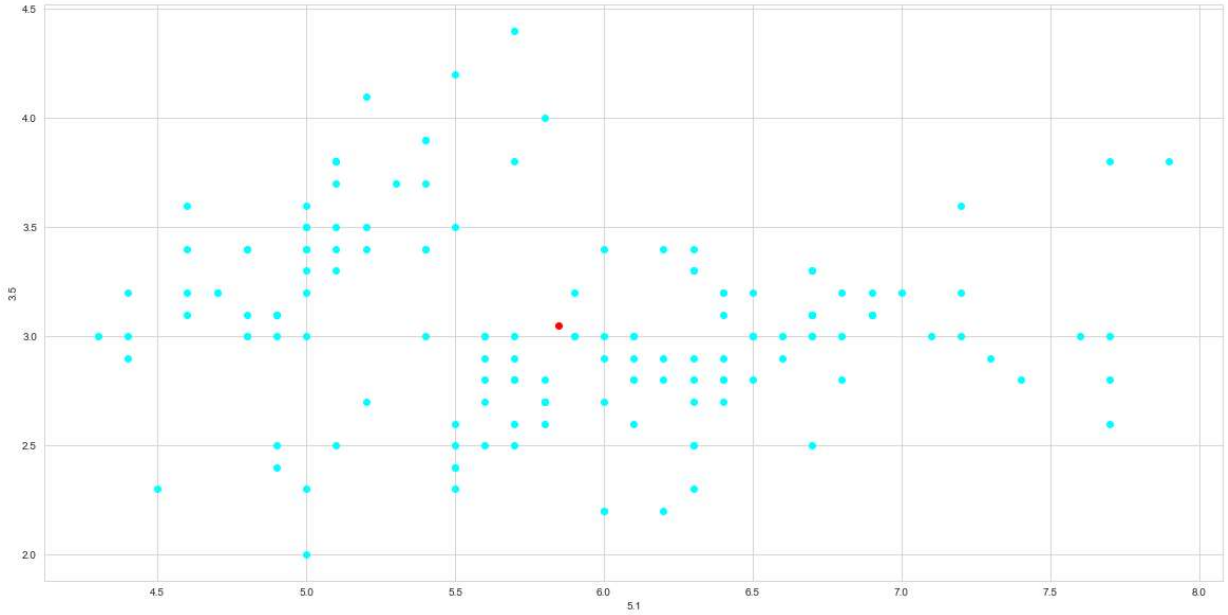
In [13]:
```
#number of clusters
K=3

# Select random observation as centroids
Centroids = (X.sample(n=K))
plt.scatter(X["5.1"],X["3.5"],c='black')
plt.scatter(Centroids["5.1"],Centroids["3.5"],c='red')
plt.xlabel('5.1')
plt.ylabel('3.5')
plt.show()
```



In [14]:
```
# Step 3 - Assign all the points to the closest cluster centroid
# Step 4 - Recompute centroids of newly formed clusters
# Step 5 - Repeat step 3 and 4

diff = 1
j=0

while(diff!=0):
    XD=X
    i=1
    for index1,row_c in Centroids.iterrows():
        ED=[]
        for index2,row_d in XD.iterrows():
            d1=(row_c["5.1"]-row_d["3.5"])**2
            d2=(row_c["3.5"]-row_d["3.5"])**2
            d=np.sqrt(d1+d2)
            ED.append(d)
        X[i]=ED
        i=i+1

    C=[]
    for index,row in X.iterrows():
        min_dist=row[1]
        pos=1
        for i in range(K):
            if row[i+1] < min_dist:
                min_dist = row[i+1]
                pos=i+1
        C.append(pos)
    X["Cluster"]=C
    Centroids_new = X.groupby(["Cluster"]).mean()[["3.5","5.1"]]
    if j == 0:
        diff=1
        j=j+1
    else:
        diff = (Centroids_new['3.5'] - Centroids['3.5']).sum() + (Centroids_new['
        print(diff.sum())
    Centroids = X.groupby(["Cluster"]).mean()[["3.5","5.1"]]
```

```
0.0

<ipython-input-14-44df8a06da73>:18: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
opy)
  X[i]=ED
```

In [15]:
```
color=['blue','green','cyan']
for k in range(K):
    data=X[X["Cluster"]==k+1]
    plt.scatter(data["5.1"],data["3.5"],c=color[k])
plt.scatter(Centroids["5.1"],Centroids["3.5"],c='red')
plt.xlabel('5.1')
plt.ylabel('3.5')
plt.show()
```



In [16]:
```
import nltk
from nltk.corpus import stopwords

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer

import pandas as pd
import string
import seaborn as sns
```

In [17]:
```
df = pd.read_csv("E:\smsspamcollection\SMSSpamCollection", names=["label","messag
df.head(2)
```

Out[17]:

	label	message
0	ham\tGo until jurong point	crazy.. Available only in bugis n great world...
1	ham\tOk lar... Joking wif u oni...	NaN

In [18]:
```
df.info()
df.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5574 entries, 0 to 5573
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   label    5574 non-null   object
 1   message  1318 non-null   object
dtypes: object(2)
memory usage: 87.2+ KB
```

Out[18]:

	label	message
count	5574	1318
unique	4969	1153
top	ham\tSorry	I'll call later
freq	52	30

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

Out[19]:

	message			
	count	unique	top	freq
label				
ham\t	4	4	im .. On the snowboarding trip. I was wonderi...	1
ham\t <#> in mca. But not conform.	0	0	NaN	NaN
ham\t <#> mins but i had to stop somewhere first.	0	0	NaN	NaN
ham\t <DECIMAL> m but its not a common car here so its better to buy from china or asia. Or if i find it less expensive. I.ll holla	0	0	NaN	NaN
ham\t and picking them up from various points	0	0	NaN	NaN
...	...	...	...	...
spam\ttodays vodafone numbers ending with 0089(my last four digits) are selected to received a £350 award. If your number matches please call 09063442151 to claim your £350 award	0	0	NaN	NaN
spam\tu r a winner U ave been specially selected 2 receive £1000 cash or a 4* holiday (flights inc) speak to a live operator 2 claim 0871277810710p/min (18)	0	0	NaN	NaN
spam\tu r subscribed 2 TEXTCOMP 250 wkly comp. 1st wk?s free question follows	1	1	subsequent wks charged@150p/msg.2 unsubscribe...	1
spam\twamma get laid?want real doggin locations sent direct to your mobile? join the UKs largest dogging network. txt dogs to 69696 now!nyt. ec2a. 3lp £1.50/msg.	0	0	NaN	NaN
spam\twe tried to contact you re your response to our offer of a new nokia fone and camcorder hit reply or call 08000930705 for delivery	0	0	NaN	NaN

4969 rows × 4 columns

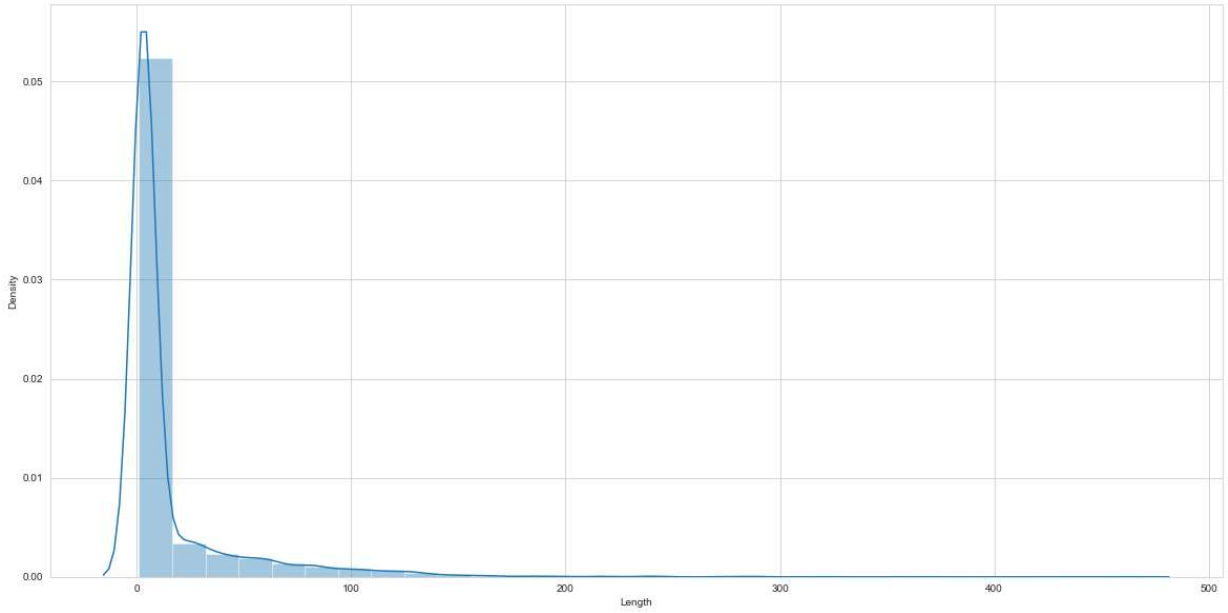
In [20]:
```
df['message'] = df['message'].apply(str)
```

In [21]:
```
df["Length"] = df["message"].apply(len)
```

In [22]:
```
sns.distplot(df["Length"], bins=30)
```

```
C:\Users\Priya\anaconda3\lib\site-packages\seaborn\distributions.py:2557: Futur
eWarning: `distplot` is a deprecated function and will be removed in a future v
ersion. Please adapt your code to use either `displot` (a figure-level function
with similar flexibility) or `histplot` (an axes-level function for histogram
s).
  warnings.warn(msg, FutureWarning)
```

Out[22]: <AxesSubplot:xlabel='Length', ylabel='Density'>



In [23]:
```
df["Length"].max()
```

Out[23]: 465

In [24]:
```
df[df["Length"]==465]["message"].iloc[0]
```

Out[24]: " hope you are having a nice day. I wanted to bring it to your notice that I ha
ve been late in paying rent for the past few months and have had to pay a $ &l
t;#> charge. I felt it would be inconsiderate of me to nag about something
you give at great cost to yourself and that's why i didnt speak up. I however a
m in a recession and wont be able to pay the charge this month hence my askin w
ell ahead of month's end. Can you please help. Thank you for everything."

In [25]:
```
df[df["Length"] == df["Length"].min()]["message"].iloc[0]
```

Out[25]: ' '

In [26]:
```
df.head(1)
```

Out[26]:

	label	message	Length
0	ham\tGo until jurong point	crazy.. Available only in bugis n great world...	89

In [28]:
```
class PreProcessText(object):
    def __init__(self):
        pass

    def __remove_punctuation(self, text):
        """
        Takes a String
        return : Return a String
        """
        message = []
        for x in text:
            if x in string.punctuation:
                pass
            else:
                message.append(x)
        message = ''.join(message)

        return message

    def __remove_stopwords(self, text):
        """
        Takes a String
        return List
        """
        words= []
        for x in text.split():

            if x.lower() in stopwords.words('english'):
                pass
            else:
                words.append(x)
        return words


    def token_words(self,text=''):
        """
        Takes String
        Return Token also called  list of words that is used to
        Train the Model
        """
        message = self.__remove_punctuation(text)
        words = self.__remove_stopwords(message)
        return words
```

In [47]:
```
import nltk
nltk.download()
```

```
showing info https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/index.xml (https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/index.xml)
```

Out[47]:
```
True
```

In [45]:
```
mess = 'Sample message! Notice: it has punctuation.'
obj = PreProcessText()
words = obj.token_words(mess)
print(words)
```

```
['Sample', 'message', 'Notice', 'punctuation']
```

In [50]:
```
bow_transformer = CountVectorizer(analyzer=obj.token_words).fit(df["message"])
```

In [51]:
```
messages_bow = bow_transformer.transform(df["message"])
```

In [52]:
```
print("Shape of sparese matrix {}".format(messages_bow.shape))
```

```
Shape of sparese matrix (5574, 2961)
```

In [53]:
```
tfidf_transformer = TfidfTransformer().fit(messages_bow)
```

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

In [55]:
```
from sklearn.naive_bayes import MultinomialNB
```

In [56]:
```
model = MultinomialNB().fit(messages_tfidf,df["label"])
```

In [57]:
```
all_predictions = model.predict(messages_tfidf)
pred = pd.DataFrame(data=all_predictions)

pred.head(6)
```

Out[57]:

	0
0	ham\tSorry
1	ham\tI cant pick the phone right now. Pls send...
2	ham\tI cant pick the phone right now. Pls send...
3	ham\tI cant pick the phone right now. Pls send...
4	ham\tSorry
5	ham\tSorry

In [58]:
```
df["label"].head(6)
```

Out[58]:
```
0                          ham\tGo until jurong point
1                  ham\tOk lar... Joking wif u oni...
2    spam\tFree entry in 2 a wkly comp to win FA Cu...
3    ham\tU dun say so early hor... U c already the...
4                ham\tNah I don't think he goes to usf
5    spam\tFreeMsg Hey there darling it's been 3 we...
Name: label, dtype: object
```

In [60]:
```
from sklearn.preprocessing import MinMaxScaler
```

In [61]:
```
from sklearn.preprocessing import StandardScaler
```

In [64]:
```
from sklearn.decomposition import PCA
```

In [65]:
```
pca=PCA(n_components=2)
```

In [75]:
```
X = df['label']
y = df['message']
```

In [76]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, rando
```

In [80]:
```
df['message'] = df['message'].astype(object)
```

In [ ]:
```

```