in this cell we imported all our classes that will be used

we also read our file csv that we will work on it

In [213]:

```
import scipy.cluster.hierarchy as sch
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import AgglomerativeClustering
Credit_Card_Customer= pd.read_csv('Credit Card Customer Data.csv')
!pip install scikit-learn-extra
from sklearn_extra.cluster import KMedoids
```

```
Requirement already satisfied: scikit-learn-extra in c:\users\hp\anaconda3\lib\site-packa
ges (0.2.0)
Requirement already satisfied: numpy>=1.13.3 in c:\users\hp\anaconda3\lib\site-packages (
from scikit-learn-extra) (1.19.2)
Requirement already satisfied: scikit-learn>=0.23.0 in c:\users\hp\anaconda3\lib\site-pac
kages (from scikit-learn-extra) (0.23.2)
Requirement already satisfied: scipy>=0.19.1 in c:\users\hp\anaconda3\lib\site-packages (
from scikit-learn-extra) (1.5.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\hp\anaconda3\lib\site-pac
kages (from scikit-learn>=0.23.0->scikit-learn-extra) (2.1.0)
Requirement already satisfied: joblib>=0.11 in c:\users\hp\anaconda3\lib\site-packages (f
rom scikit-learn>=0.23.0->scikit-learn-extra) (0.17.0)
```

in here we wanted to know the number of rows and columns in the dataset

In [214]:

```
Credit_Card_Customer.shape
```

Out[214]:

```
(660, 7)
```

we viewd our dataset

In [215]:

```
Credit_Card_Customer.head
```

Out[215]:

```
<bound method NDFrame.head of      Sl_No  Customer Key  Avg_Credit_Limit  Total_Credit_Ca
rds  \
0         1         87073            100000                   2
1         2         38414             50000                   3
2         3         17341             50000                   7
3         4         40496             30000                   5
4         5         47437            100000                   6
..      ...           ...               ...                 ...
655     656         51108             99000                  10
656     657         60732             84000                  10
657     658         53834            145000                   8
658     659         80655            172000                  10
659     660         80150            167000                   9

     Total_visits_bank  Total_visits_online  Total_calls_made
0                    1                    1                 0
```

```
1                     0                    10                     9
2                     1                     3                     4
3                     1                     1                     4
4                     0                    12                     3
..                  ...                   ...                   ...
655                   1                    10                     0
656                   1                    13                     2
657                   1                     9                     1
658                   1                    15                     0
659                   0                    12                     2

[660 rows x 7 columns]>
```

we dropped two columns

In [216]:

```
dd=Credit_Card_Customer.drop(['Sl_No','Customer Key'],axis=1)
```

we viewd our dataset after dropping the two columns

In [217]:

```
dd.head
```

Out[217]:

```
<bound method NDFrame.head of      Avg_Credit_Limit  Total_Credit_Cards  Total_visits_ban
k  \
0               100000                   2                   1
1                50000                   3                   0
2                50000                   7                   1
3                30000                   5                   1
4               100000                   6                   0
..                 ...                 ...                 ...
655              99000                  10                   1
656              84000                  10                   1
657             145000                   8                   1
658             172000                  10                   1
659             167000                   9                   0

     Total_visits_online  Total_calls_made
0                      1                 0
1                     10                 9
2                      3                 4
3                      1                 4
4                     12                 3
..                   ...               ...
655                   10                 0
656                   13                 2
657                    9                 1
658                   15                 0
659                   12                 2

[660 rows x 5 columns]>
```

in here we determined the number of columns we want to work on using the iloc function

In [218]:

```
data = dd.iloc[:, 0:2].values
```

we used hieraeachial clustering using single linkage and using

euclidean as the distance measure

In [219]:

```
cluster = AgglomerativeClustering(n_clusters=3,affinity='euclidean', linkage='single')
y_hc=cluster.fit_predict(data)
y_hc
```

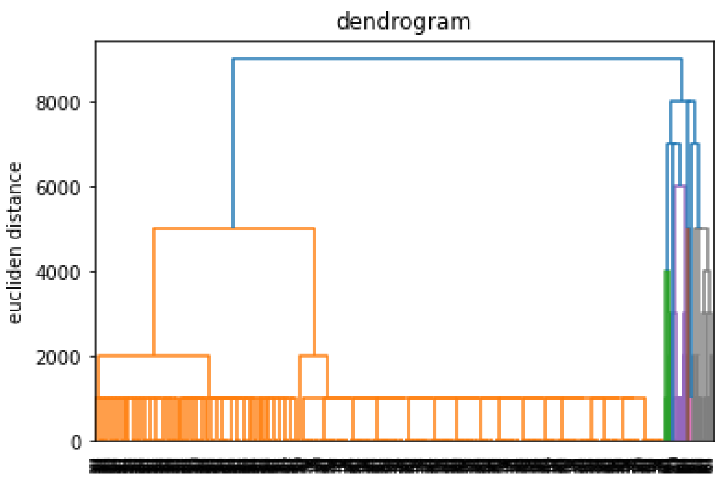
Out[219]:

```
array([2, 1, 1, 1, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 2, 0, 2,
       2, 2, 2, 0, 2, 0, 0, 0, 0, 2, 0, 0, 2, 0, 0, 0, 0, 0, 2, 2, 2, 0,
       0, 0, 2, 0, 2, 2, 2, 0, 0, 0, 2, 2, 0, 2, 0, 2, 0, 2, 2, 0, 0, 0],
      dtype=int64)
```

we visulalized our results using the dendrogram and plotting its size

In [220]:

```
dendrogram= sch.dendrogram(sch.linkage(data,'single'))
plt.title('dendrogram')
plt.xlabel('customer')
plt.ylabel('eucliden distance')
plt.show()
plt.figure(figsize=(20,20))
```



customer

Out[220]:

```
<Figure size 1440x1440 with 0 Axes>
```

```
<Figure size 1440x1440 with 0 Axes>
```

in the following cells we imported the scaled functions and we scaled our data

In [221]:

```
from sklearn.preprocessing import MinMaxScaler #Importing MinMaxScaler
scaler= MinMaxScaler() #Initialising the instance of the scaler
```

In [222]:

```
scaled_features = scaler.fit_transform(data)
```

then we imported the silhouette_score to be able to measure the validation of our data

In [223]:

```
from sklearn.metrics import silhouette_score
```

In [224]:

```
hc_score= silhouette_score(scaled_features,y_hc)
hc_score
```

Out[224]:

```
0.5255869852401159
```

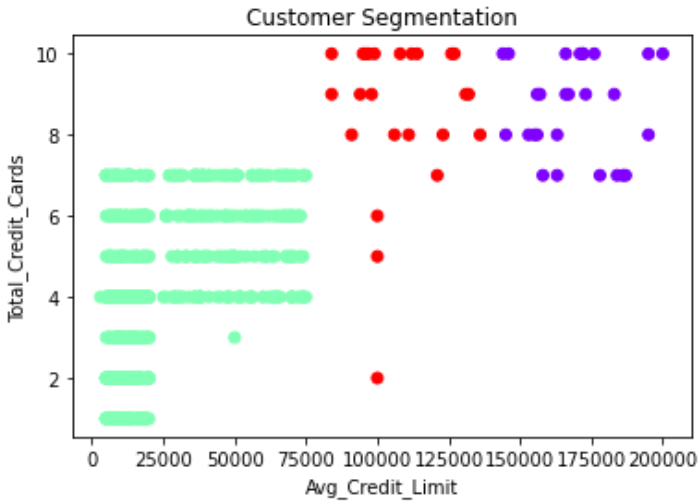
we visulalized our results using the scatter plot

In [225]:

```
plt.scatter(data[:,0], data[:,1], c=cluster.labels_, cmap='rainbow')
plt.title('Customer Segmentation')
plt.xlabel('Avg_Credit_Limit')
plt.ylabel('Total_Credit_Cards')
```

Out[225]:

```
Text(0, 0.5, 'Total_Credit_Cards')
```



we used the k-medoids clustering using the manhattan distance as the distance measure

In [226]:

```
cluster = KMedoids(n_clusters=3, metric="manhattan",init="random" ,random_state=33)
y_km=cluster.fit_predict(data)
y_km
```

Out[226]:

```
array([2, 2, 2, 1, 2, 1, 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1,
       0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1,
       1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1,
       1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0,
       0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0,
       0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0,
       0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0,
       1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0,
       1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 1, 1, 2, 2, 2, 2,
       2, 2, 1, 2, 2, 2, 1, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2,
       2, 1, 2, 2, 2, 2, 2, 2, 1, 1, 2, 1, 2, 2, 1, 2, 1, 2, 2, 2, 1, 1,
       2, 1, 2, 2, 2, 1, 1, 2, 1, 2, 1, 1, 1, 2, 2, 2, 1, 2, 2, 1, 2, 1,
       1, 2, 2, 1, 2, 2, 2, 2, 2, 1, 1, 2, 2, 2, 1, 2, 1, 1, 2, 2, 1, 2,
       1, 1, 2, 1, 1, 2, 2, 1, 1, 1, 1, 2, 1, 2, 1, 2, 1, 2, 2, 2, 1, 1,
       1, 1, 1, 1, 2, 1, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 1, 2, 1,
       2, 1, 2, 1, 1, 2, 2, 2, 1, 1, 2, 1, 2, 2, 1, 2, 1, 1, 2, 2, 2, 1,
       2, 2, 2, 2, 1, 2, 1, 2, 2, 1, 2, 2, 1, 1, 1, 2, 2, 1, 1, 2, 2, 2,
       2, 1, 1, 1, 2, 2, 2, 1, 1, 2, 1, 1, 2, 1, 2, 1, 1, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2],
      dtype=int64)
```

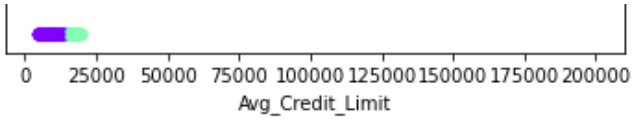
we visulalized our results using the scatter plot

In [227]:

```
plt.scatter(data[:,0], data[:,1], c=cluster.labels_, cmap='rainbow')
plt.title('Customer Segmentation')
plt.xlabel('Avg_Credit_Limit')
plt.ylabel('Total_Credit_Cards')
```

Out[227]:

```
Text(0, 0.5, 'Total_Credit_Cards')
```



we measured the silhouettee score to be able to determine the validation of the data

In [228]:

```
km_score= silhouette_score(scaled_features,y_km)
km_score
```

Out[228]:

```
0.09368221010034446
```

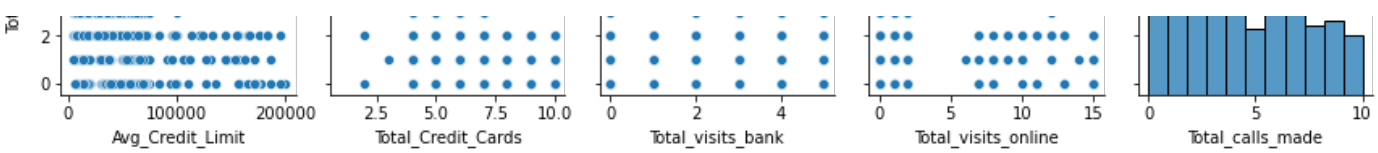
we plotted all the possible scattered between the datasets

In [229]:

```
sns.pairplot(dd)
```

Out[229]:

```
<seaborn.axisgrid.PairGrid at 0x271fd81cee0>
```


2
0
0
100000
200000
Avg_Credit_Limit
2.5
5.0
7.5
10.0
Total_Credit_Cards
0
2
4
Total_visits_bank
0
5
10
15
Total_visits_online
0
5
10
Total_calls_made

In [ ]:

In [ ]: