Untitled1

June 20, 2021

[1]:
```
#importing the required modules
import numpy as np
import pandas as pd
#import pandas_profiling as pp
from matplotlib import pyplot as plt
%matplotlib inline
import streamlit as st
```

[24]:
```
df = pd.read_csv("/Users/kirankunwar/Desktop/BFR_input.csv")
df.head()
df1 = df.head(500)
```

[25]:
```
df1
```

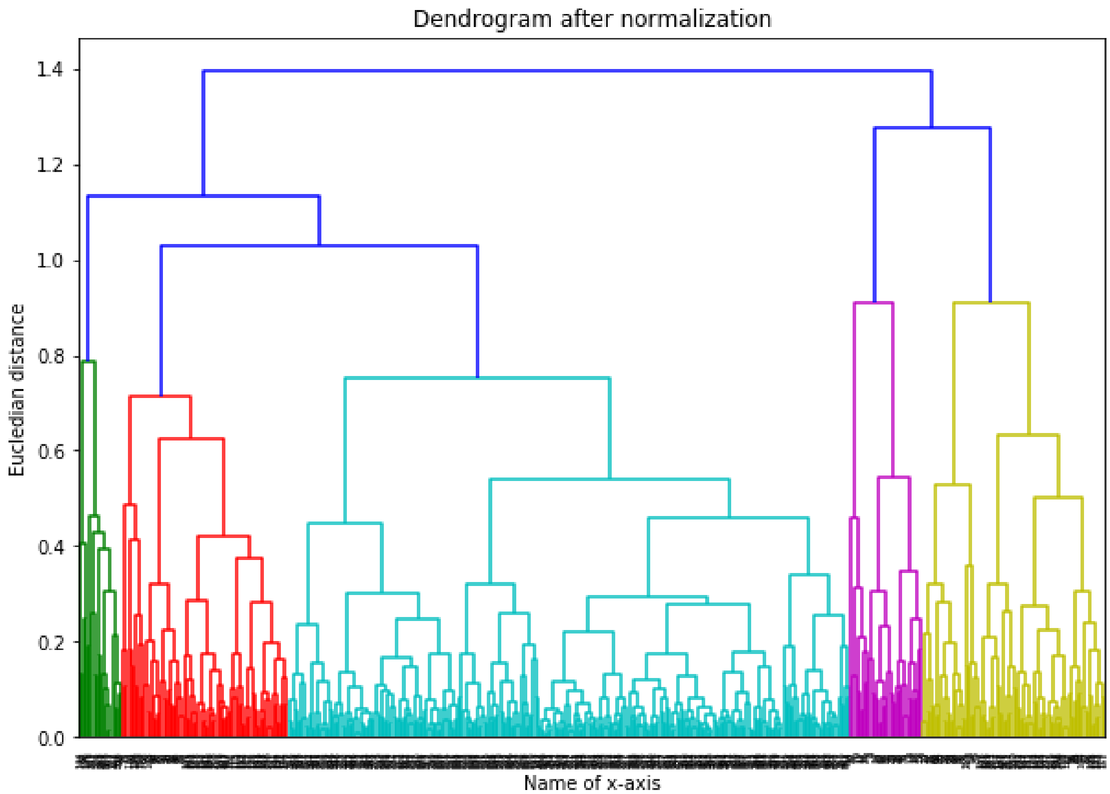
[25]:
```
       0    140.5625  55.68378214  -0.234571412  -0.699648398  3.199832776  \
0      1  102.507812    58.882430      0.465318     -0.515088     1.677258
1      2  103.015625    39.341649      0.323328      1.051164     3.121237
2      3  136.750000    57.178449     -0.068415     -0.636238     3.642977
3      4   88.726562    40.672225      0.600866      1.123492     1.178930
4      5   93.570312    46.698114      0.531905      0.416721     1.636288
..   …           …            …             …             …            …
495  496   85.671875    46.698687      1.084528      1.100297   109.575251
496  497  114.835938    51.926146     -0.009627     -0.528806     1.227425
497  498  127.992188    50.727596     -0.065354     -0.393038     1.173913
498  499   85.523438    36.694040      0.733149      2.476574     1.957358
499  500  131.367188    60.134468     -0.128237     -0.775476     2.463211

     19.11042633  7.975531794  74.24222492  0.1
0      14.860146    10.576487   127.393580    0
1      21.744669     7.735822    63.171909    0
2      20.959280     6.896499    53.593661    0
3      11.468720    14.269573   252.567306    0
4      14.545074    10.621748   131.394004    0
..           …            …           …   …
495    82.243735    -0.090485    -1.401877    0
496    12.031249    14.087309   234.473686    0
497    14.394636    14.087723   213.922448    0
```

```
498    18.373467    10.403407   114.392908    0
499    16.309439    10.016563   119.224947    0

[500 rows x 10 columns]
```

[26]:
```
#normalising the data so that all the variables are scaled same. Now the model␣
↪is not biased towards the variable with higher values
from sklearn import preprocessing
data_normalized = preprocessing.normalize(df1)
data_normalized = pd.DataFrame(data_normalized, columns=df1.columns)
data_normalized.head()
```

[26]:
```
          0  140.5625  55.68378214  -0.234571412  -0.699648398  3.199832776  \
0  0.005722  0.586563     0.336933      0.002663     -0.002947     0.009597
1  0.015477  0.797199     0.304450      0.002502      0.008135     0.024154
2  0.018842  0.858862     0.359111     -0.000430     -0.003996     0.022880
3  0.014737  0.326893     0.149848      0.002214      0.004139     0.004343
4  0.029590  0.553748     0.276359      0.003148      0.002466     0.009684

   19.11042633  7.975531794  74.24222492  0.1
0     0.085032     0.060520     0.728963  0.0
1     0.168274     0.059865     0.488864  0.0
2     0.131635     0.043314     0.336596  0.0
3     0.042254     0.052573     0.930526  0.0
4     0.086078     0.062859     0.777588  0.0
```

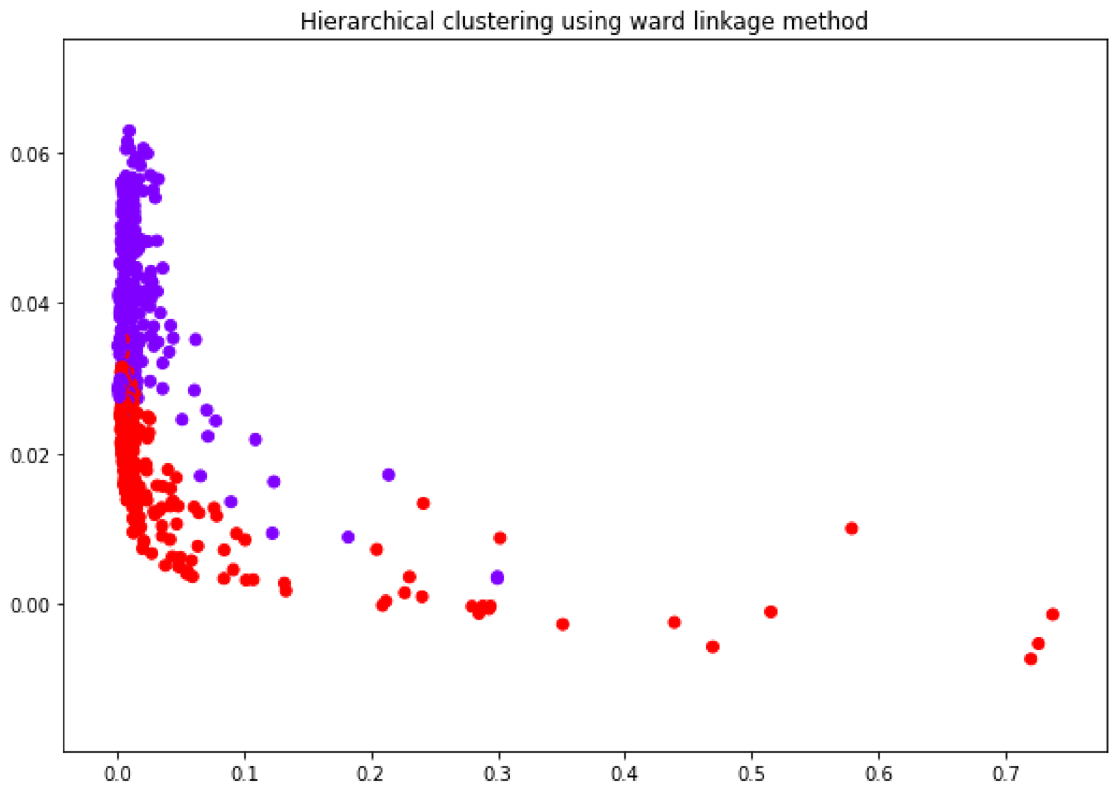
[38]:
```
from scipy.cluster import hierarchy
from sklearn.cluster import AgglomerativeClustering
from sklearn import metrics
fig3 = plt.figure(figsize=(10, 7))
plt.title("Dendrogram after normalization")
plt.xlabel("Name of x-axis")
plt.ylabel("Eucledian distance")
dendrogram = hierarchy.dendrogram(hierarchy.linkage(data_normalized,␣
↪method='complete'))
st.write(fig3)
```



[41]:
```
#performing hiererchical clustering using scikitlearn package
from sklearn.cluster import AgglomerativeClustering
from sklearn import metrics

#defining the number of clusters that we wanted. 2 is the optimal number of␣
↪cluster in our case
#clustering is performed using ward linkage method
k = 2
cluster = AgglomerativeClustering(n_clusters=k, affinity='euclidean',␣
↪linkage='ward')
cluster_predict = cluster.fit_predict(data_normalized)
#print ("Cluser prediction: ",cluster_predict)
#print ("\nTraget variable: ",y )
#print ("\nCluster labels: ",cluster.labels_)
fig4 = plt.figure(figsize = (10,7))
ax = plt.scatter(data_normalized.iloc[:,5], data_normalized.iloc[:,7], c =␣
↪cluster_predict, cmap='rainbow')

# accuracy of clustering is calculated.
# cluster evaluation is done using external validation index called Rand index
```

```
#print("\nAccuracy of ward linkage: ",metrics.
 ↪adjusted_rand_score(cluster_predict,y))
plt.title("Hierarchical clustering using ward linkage method")
#fig4.update()
```

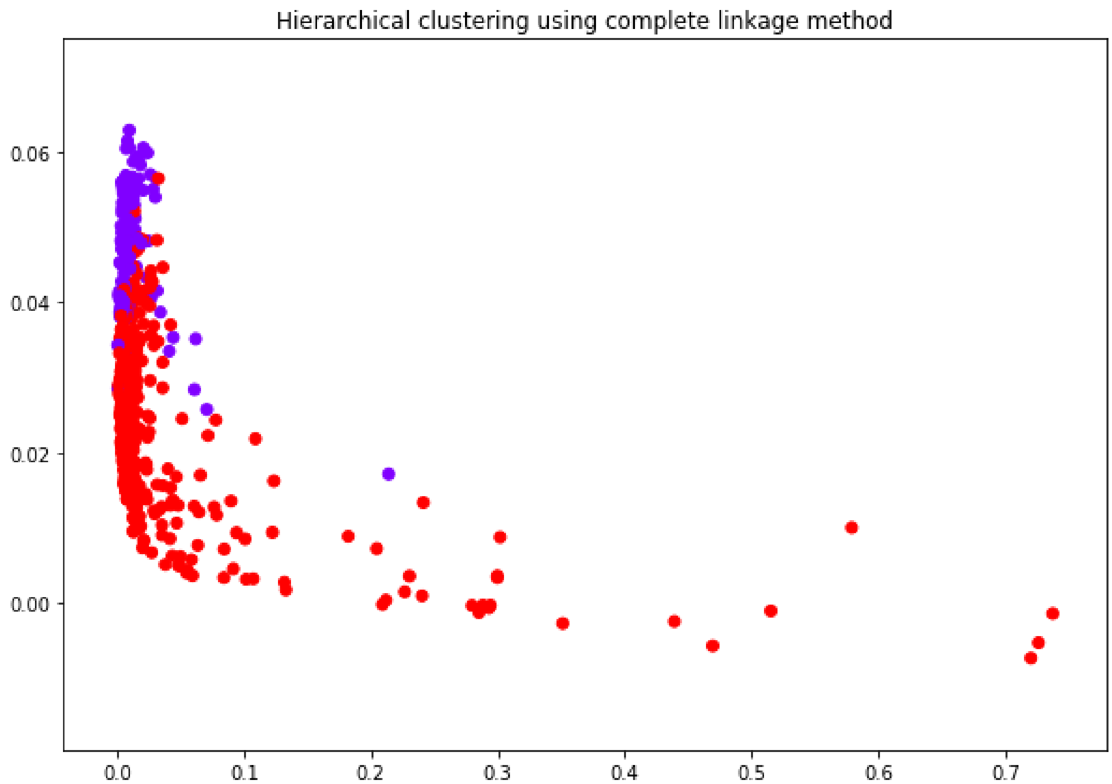
[41]: Text(0.5, 1.0, 'Hierarchical clustering using ward linkage method')



[42]:
```
cluster = AgglomerativeClustering(n_clusters=k, affinity='euclidean',␣
 ↪linkage='complete')
cluster_predict = cluster.fit_predict(data_normalized)
#print ("Cluser prediction: ",cluster_predict)
#print ("\nTraget variable: ",y )
#print ("\nCluster labels: ",cluster.labels_)
fig4 = plt.figure(figsize = (10,7))
ax = plt.scatter(data_normalized.iloc[:,5], data_normalized.iloc[:,7], c =␣
 ↪cluster_predict, cmap='rainbow')

# accuracy of clustering is calculated.
# cluster evaluation is done using external validation index called Rand index
#print("\nAccuracy of ward linkage: ",metrics.
 ↪adjusted_rand_score(cluster_predict,y))
```

```
plt.title("Hierarchical clustering using complete linkage method")
```

[42]: Text(0.5, 1.0, 'Hierarchical clustering using complete linkage method')



[ ]: