```
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.cluster import AgglomerativeClustering
from sklearn.preprocessing import MinMaxScaler
from sklearn import preprocessing
from sklearn.cluster import DBSCAN
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA


df=pd.read_csv("/content/sample_data/Mall_Customers (1).csv")
df.head()
```

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   CustomerID              200 non-null    int64
 1   Gender                  200 non-null    object
 2   Age                     200 non-null    int64
 3   Annual Income (k$)      200 non-null    int64
 4   Spending Score (1-100)  200 non-null    int64
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
```

```
from sklearn.preprocessing import LabelEncoder
# label_encoder object knows how to understand word labels.
LE = LabelEncoder()

# Encode labels in column 'species' .
df['Gender']= LE.fit_transform(df['Gender'])
df['Gender'].unique()
```

```
array([1, 0])
```

```
df.head()
```

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	1	19	15	39
1	2	1	21	15	81
2	3	0	20	16	6
3	4	0	23	16	77

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
pca=PCA(n_components=2)


r_data=pca.fit_transform(df)
r_data
```

```
[ 5.12330746e+01, -2.67131449e+01],
[ 5.25996074e+01,  3.94587757e+01],
[ 5.29732462e+01, -3.36582168e+01],
[ 5.43827783e+01,  3.63534576e+01],
[ 5.48127449e+01, -3.09773619e+01],
[ 5.61340396e+01,  2.48956272e+01],
[ 5.65943147e+01, -3.55696404e+01],
[ 5.81109966e+01,  4.00641245e+01],
[ 5.83792556e+01, -4.78819487e+01],
[ 5.98459161e+01,  2.86593788e+01],
[ 6.02226657e+01, -4.72030119e+01],
[ 6.16384130e+01,  2.37830791e+01],
[ 6.25536841e+01, -1.91847043e+01],
[ 6.39448044e+01,  3.37269265e+01],
[ 6.52131146e+01, -3.99079146e+01],
[ 6.66416701e+01,  4.29738590e+01],
[ 6.86293685e+01, -2.66332837e+01],
[ 6.99504744e+01,  2.42662809e+01],
[ 7.08701774e+01, -3.06526839e+01],
[ 7.23352368e+01,  4.44016833e+01],
[ 7.31854458e+01, -2.24786110e+01],
[ 7.43621754e+01,  1.34695199e+01],
[ 7.48949576e+01, -3.70339221e+01],
[ 7.62914212e+01,  2.60567015e+01],
[ 7.67242385e+01, -3.90448847e+01],
[ 7.81836229e+01,  4.07531776e+01],
[ 7.88838456e+01, -3.98176982e+01],
[ 8.04122551e+01,  3.62755256e+01],
[ 8.06882590e+01, -3.92611145e+01],
[ 8.21428206e+01,  2.04086918e+01],
[ 8.45418779e+01, -4.05129379e+01],
[ 8.61020080e+01,  3.89593736e+01],
[ 8.82484722e+01, -1.79810676e+01],
[ 8.95505745e+01,  3.57157728e+01],
[ 9.03167375e+01, -3.66123814e+01],
[ 9.18153789e+01,  3.83334023e+01],
[ 9.27394358e+01, -1.21234745e+01],
[ 9.41038274e+01,  4.68433740e+01],
[ 9.52051912e+01, -2.97338063e+01],
[ 9.65642394e+01,  1.90540495e+01],
[ 9.78767205e+01, -3.35854954e+01],
[ 9.92721234e+01,  3.37362796e+01],
[ 9.97835718e+01, -2.61483832e+01],
[ 1.01014768e+02,  1.90735920e+01],
[ 1.05596395e+02, -4.06055221e+01],
[ 1.07032676e+02,  3.90183096e+01],
[ 1.10249446e+02, -3.60991880e+01],
[ 1.11652574e+02,  2.79646451e+01],
[ 1.14615358e+02, -2.40178247e+01],
[ 1.15911505e+02,  2.37299674e+01],
[ 1.20939935e+02, -3.08598887e+01],
[ 1.22297753e+02,  3.28530691e+01]])
```

```
data=preprocessing.scale(r_data)
```

```
data=pd.DataFrame(data,columns=['X','Y'])
data.head()
```

	X	Y
0	-1.730708	-0.208398
1	-1.712020	1.328786
2	-1.698942	-1.439447
3	-1.677289	1.162589

Next steps: Generate code with data | View recommended plots | New interactive sheet

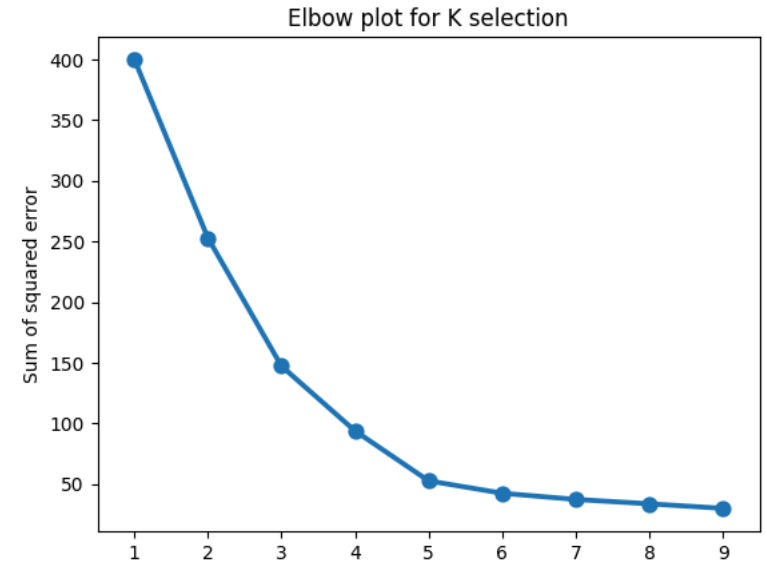
K-Means Clustering

```
sse = []
for k in range(1,10):
  km = KMeans(n_clusters=k)
  km.fit(data)
  sse.append(km.inertia_)
```

```
#plt.plot(np.arange(1,10),sse)
sns.pointplot(x=np.arange(1, 10),y=sse)
plt.title('Elbow plot for K selection')
```

```
plt.xlabel('K value')
plt.ylabel('Sum of squared error')
```

```
Text(0, 0.5, 'Sum of squared error')
```



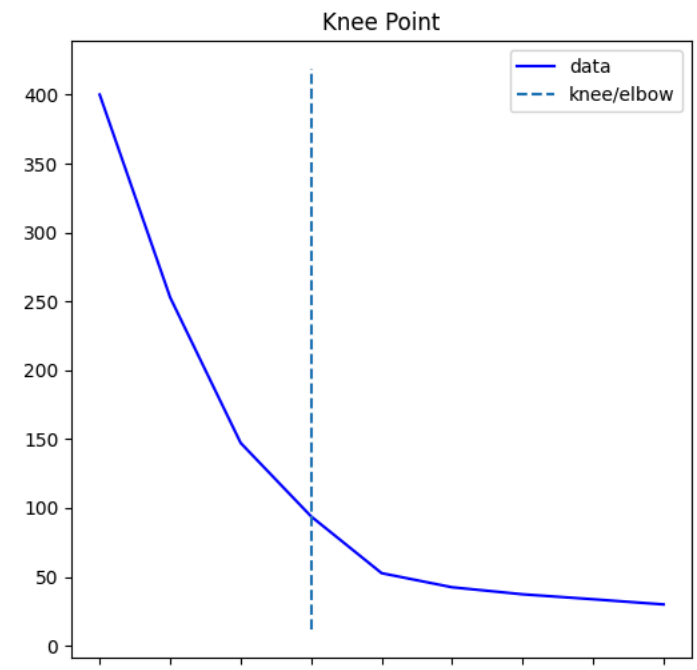

```
!pip install kneed
```

```
Requirement already satisfied: kneed in /usr/local/lib/python3.10/dist-packages (0.8.5)
Requirement already satisfied: numpy>=1.14.2 in /usr/local/lib/python3.10/dist-packages (from kneed) (1.26.4)
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from kneed) (1.13.1)
```

```
from kneed import KneeLocator
kl=KneeLocator(np.arange(1,10), sse, S=1.0, curve="convex", direction="decreasing")
print(kl.elbow)
```

```
4
```

```
kl.plot_knee()
```




```
kmeans=KMeans(n_clusters=4)
```

```
cluster=kmeans.fit_predict(data[['X','Y']])
```

```
kmeans=KMeans(n_clusters=4)
```

```
cluster=kmeans.fit_predict(data[['X','Y']])

data['cluster']=cluster

data.head()
```

	X	Y	cluster
0	-1.730708	-0.208398	1
1	-1.712020	1.328786	1
2	-1.698942	-1.439447	1
3	-1.677289	1.162589	1

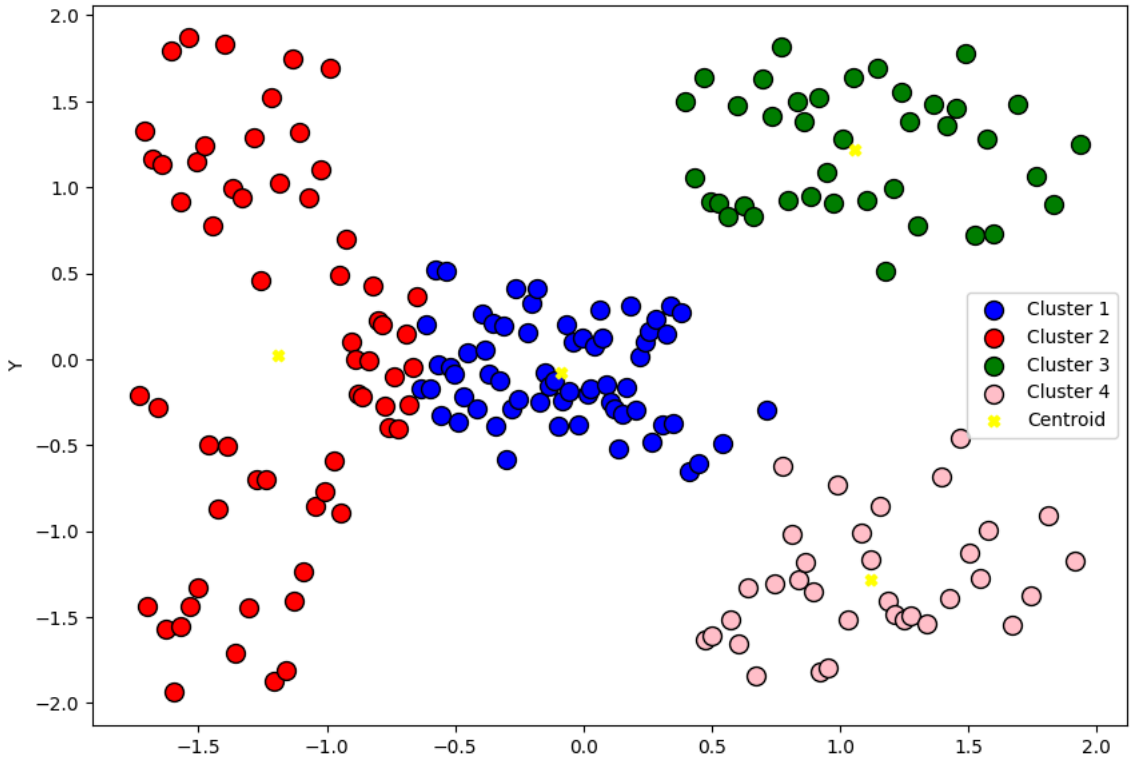
Next steps: Generate code with data | View recommended plots | New interactive sheet

```
data['cluster'].value_counts()
```

cluster	count
0	65
1	62
2	39
3	34

```
df1= data[data['cluster']==0]
df2= data[data['cluster']==1]
df3= data[data['cluster']==2]
df4= data[data['cluster']==3]


plt.figure(figsize=(10,7))
plt.scatter(df1.values[:,0],df1.values[:,1],color="blue",label= 'Cluster 1', edgecolors="black",s=100)
plt.scatter(df2.values[:,0],df2.values[:,1],color="red",label= 'Cluster 2', edgecolors="black",s=100)
plt.scatter(df3.values[:,0],df3.values[:,1],color="green",label= 'Cluster 3', edgecolors="black",s=100)
plt.scatter(df4.values[:,0],df4.values[:,1],color="pink",label= 'Cluster 4', edgecolors="black",s=100)
plt.xlabel("X")
plt.ylabel("Y")
plt.scatter(kmeans.cluster_centers_[:,0],kmeans.cluster_centers_[:,1],marker='X',color='yellow',label='Centroid')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f12041a2e90>
```



DBSCAN Clustering

```
db=DBSCAN(eps=1.0,metric='euclidean')
```

```
pr=db.fit_predict(data)
```

```
data['cluster']=pr
```

```
data.head()
```

	X	Y	cluster
0	-1.730708	-0.208398	0
1	-1.712020	1.328786	0
2	-1.698942	-1.439447	0
3	-1.677289	1.162589	0

Next steps: Generate code with data | View recommended plots | New interactive sheet

```
data['cluster'].value_counts()
```

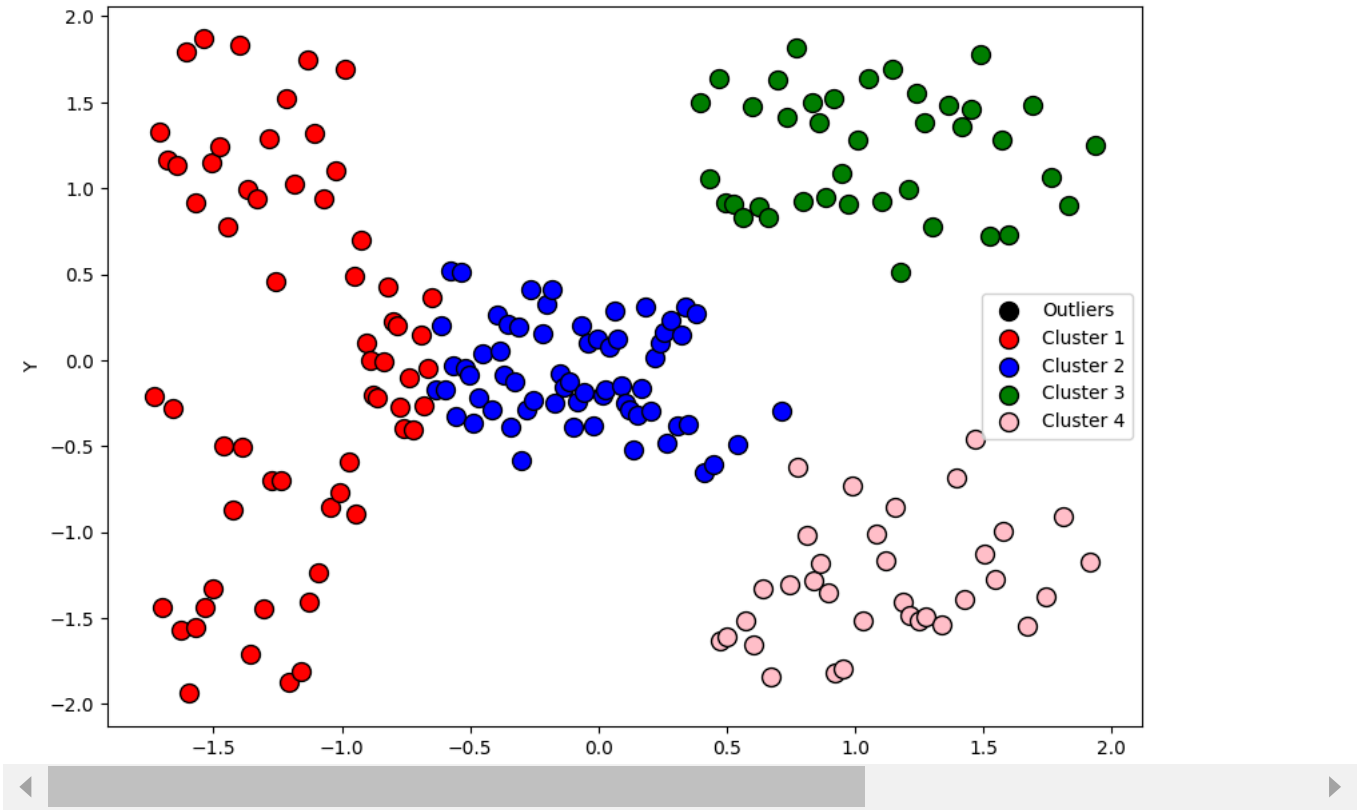
cluster	count
1	65
0	62
2	39
3	34

```
outliers_data=data[data['cluster']==-1]
cluster1_data=data[data['cluster']==0]
cluster2_data=data[data['cluster']==1]
cluster3_data=data[data['cluster']==2]
```

```
cluster4_data=data[data['cluster']==3]
```

```
plt.figure(figsize=(10,7))
plt.scatter(outliers_data['X'],outliers_data['Y'],color="black",label= 'Outliers', edgecolors="black",s=100)
plt.scatter(cluster1_data['X'],cluster1_data['Y'],color="red",label= 'Cluster 1', edgecolors="black",s=100)
plt.scatter(cluster2_data['X'],cluster2_data['Y'],color="blue",label= 'Cluster 2', edgecolors="black",s=100)
plt.scatter(cluster3_data['X'],cluster3_data['Y'],color="green",label= 'Cluster 3', edgecolors="black",s=100)
plt.scatter(cluster4_data['X'],cluster4_data['Y'],color="pink",label= 'Cluster 4', edgecolors="black",s=100)
plt.xlabel("X")
plt.ylabel("Y")
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f1204a16410>
```

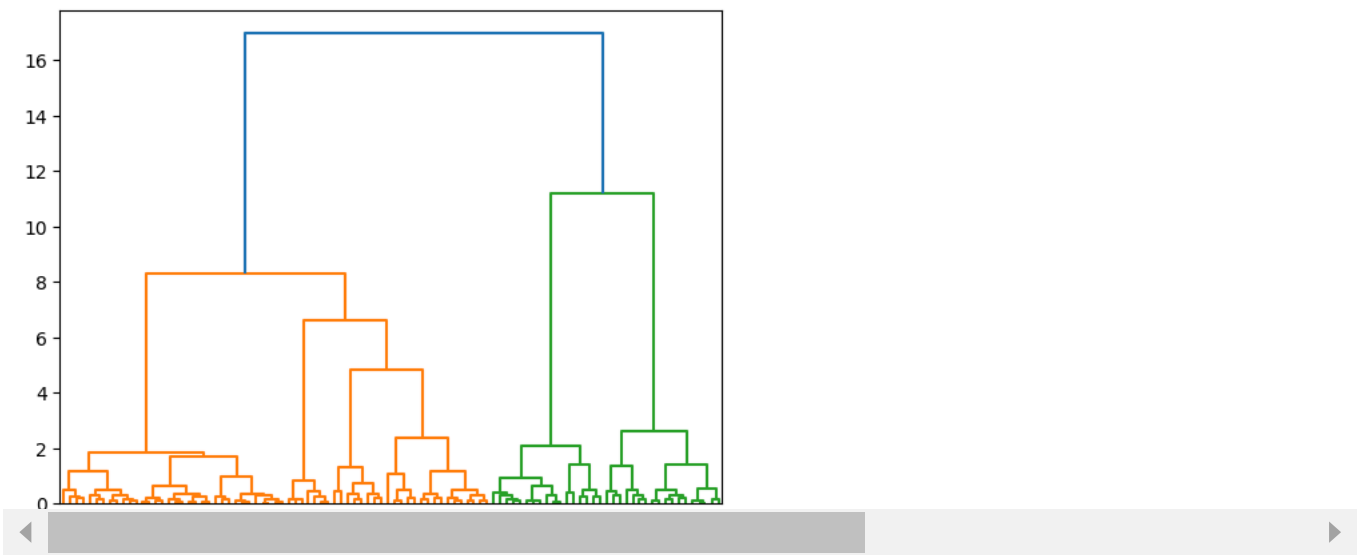


Heirarchical Clustering

```
import scipy.cluster.hierarchy as sch
```

```
data1=data.sample(100)
```

```
dend=sch.dendrogram(sch.linkage(data1.values,method='ward'))
```

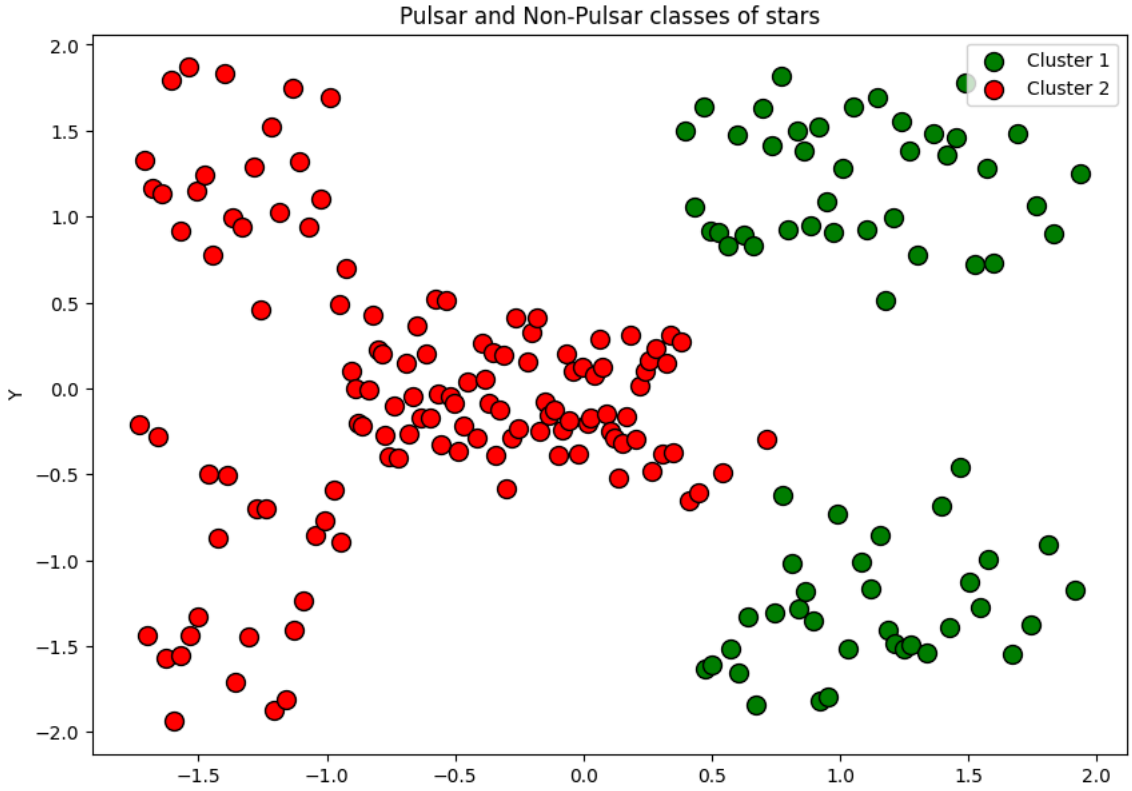


```
H=AgglomerativeClustering(n_clusters=2,linkage='ward')

pred=H.fit_predict(data)

temp=data

temp['cluster']=pred

temp['cluster'].value_counts()
```

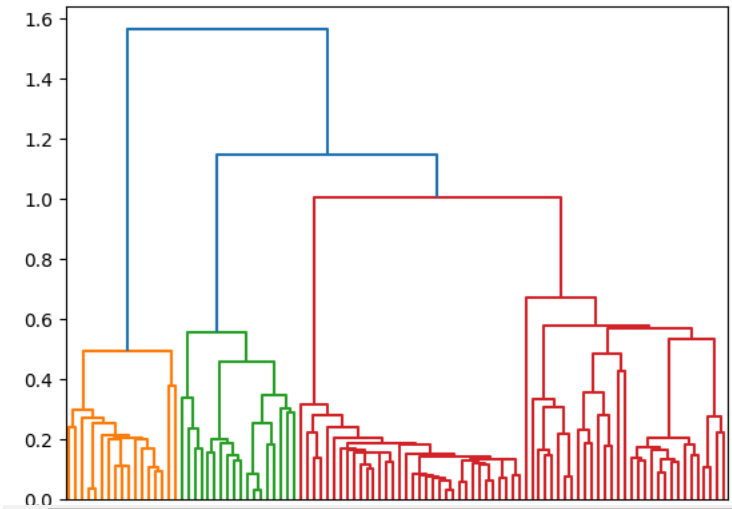
cluster	count
1	127
0	73

```
temp1=temp[temp['cluster']==0]
temp2=temp[temp['cluster']==1]


plt.figure(figsize=(10,7))
plt.scatter(temp1.values[:,0], temp1.values[:,1],color="green",label="Cluster 1",edgecolors="black",s=100)
plt.scatter(temp2.values[:,0], temp2.values[:,1],color="red",label="Cluster 2",edgecolors="black",s=100)
plt.xlabel("X")
plt.ylabel("Y")
plt.title('Pulsar and Non-Pulsar classes of stars')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f1204cbc430>
```




```
dend=sch.dendrogram(sch.linkage(data1.values,method='single'))
```

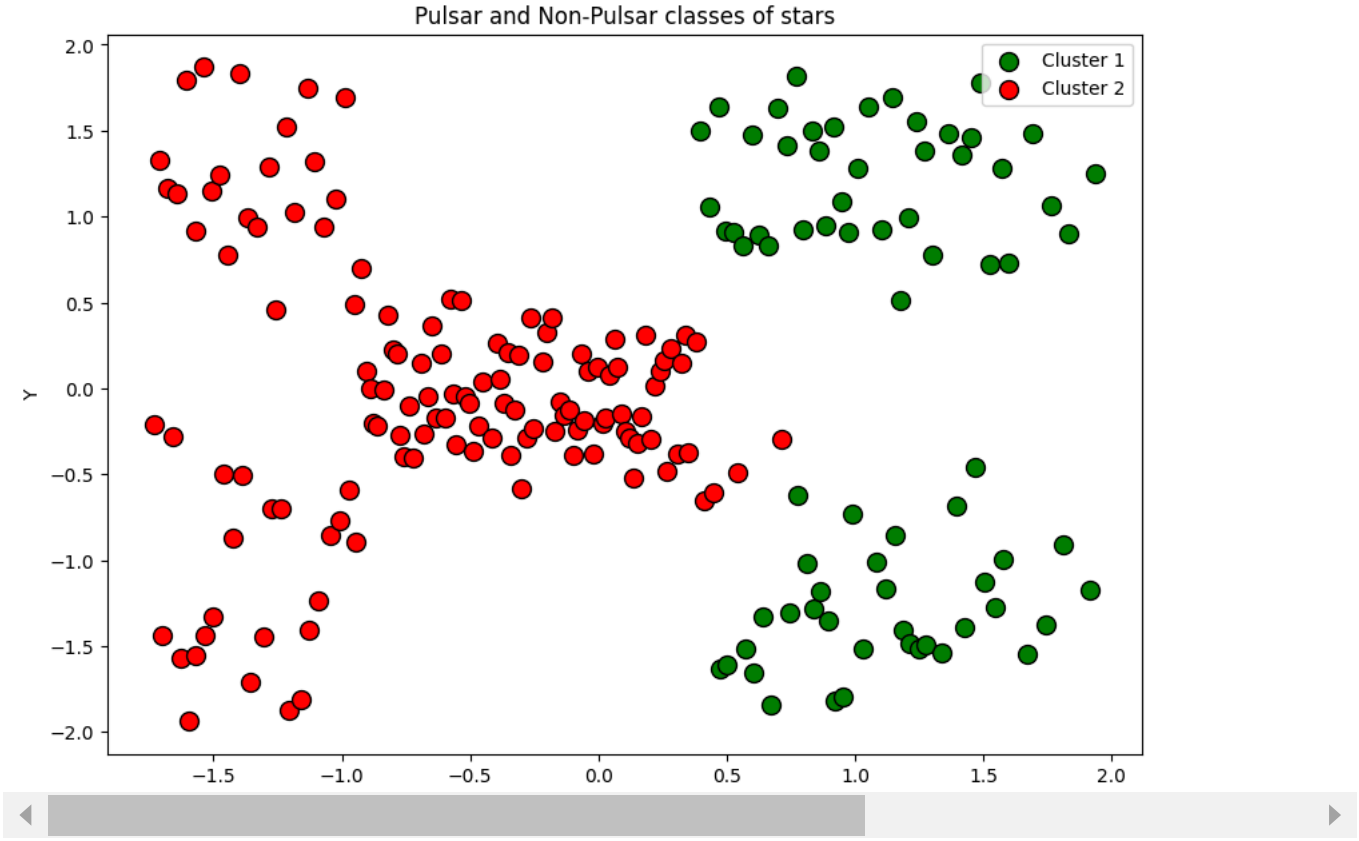



```
H=AgglomerativeClustering(n_clusters=2,linkage="single")

pred=H.fit_predict(data)

temp=data

temp['cluster']=pred

temp['cluster'].value_counts()
```

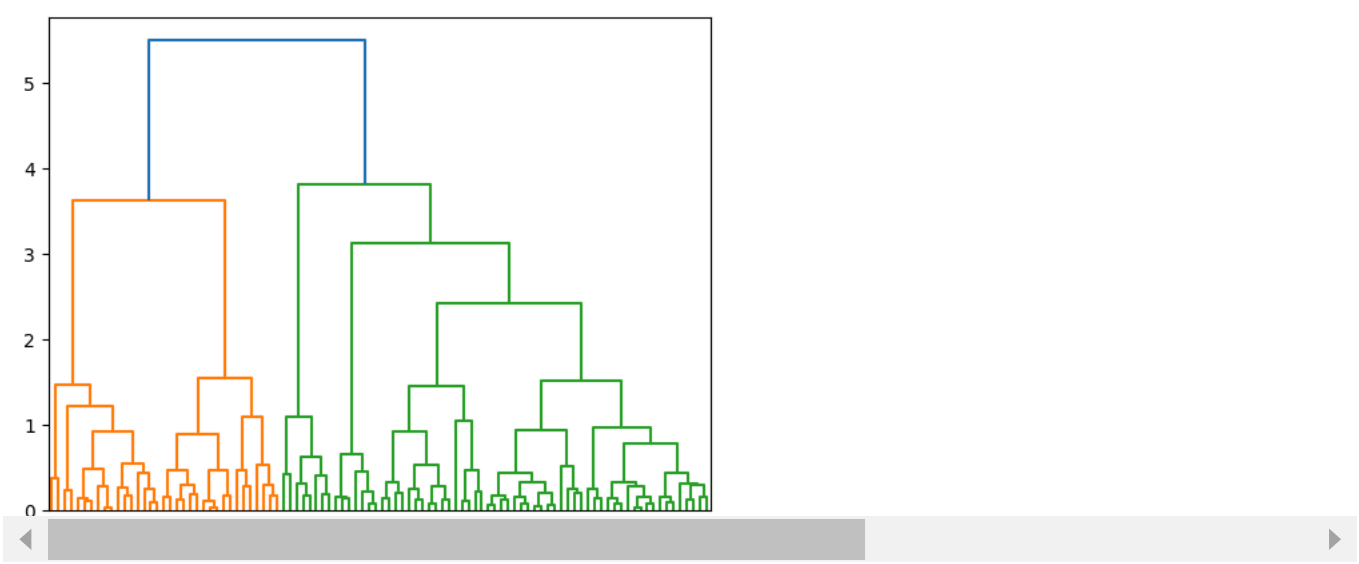
cluster	count
1	127
0	73

```
temp1=temp[temp['cluster']==0]
temp2=temp[temp['cluster']==1]


plt.figure(figsize=(10,7))
plt.scatter(temp1.values[:,0], temp1.values[:,1],color="green",label="Cluster 1",edgecolors="black",s=100)
plt.scatter(temp2.values[:,0], temp2.values[:,1],color="red",label="Cluster 2",edgecolors="black",s=100)
plt.xlabel("X")
plt.ylabel("Y")
plt.title('Pulsar and Non-Pulsar classes of stars')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f12046dab60>
```




```
dend=sch.dendrogram(sch.linkage(data1.values,method='complete'))
```




```
H=AgglomerativeClustering(n_clusters=2,linkage="complete")
```

```
pred=H.fit_predict(data)
```

```
temp=data
```

```
temp['cluster']=pred
```

```
temp["cluster"].value_counts()
```

cluster	count
0	127
1	73

dtype: int64

```
temp1=temp[temp['cluster']==0]
temp2=temp[temp['cluster']==1]
```