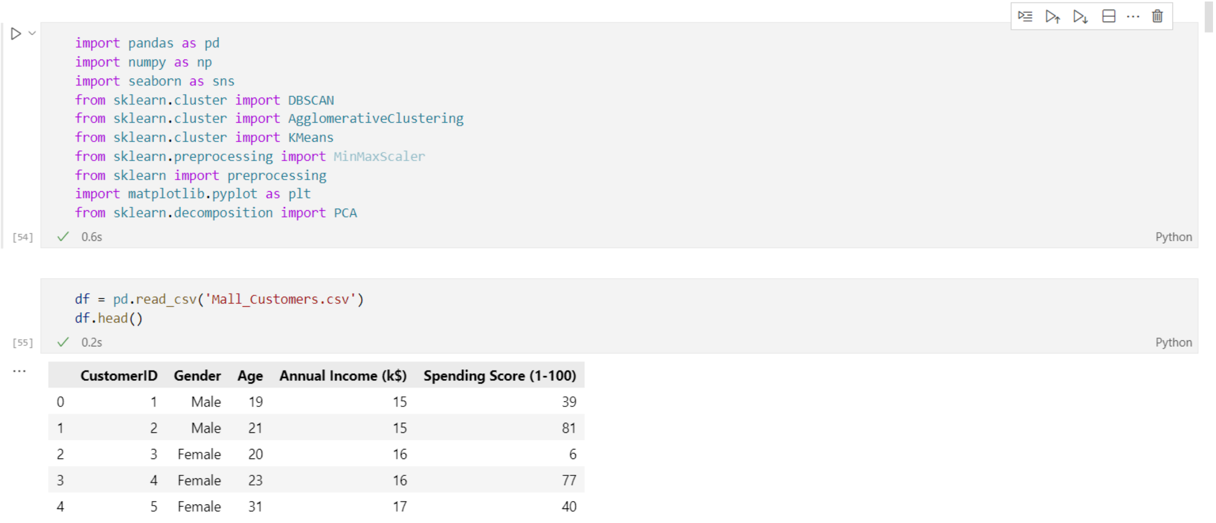
lab 6ii.ipynb



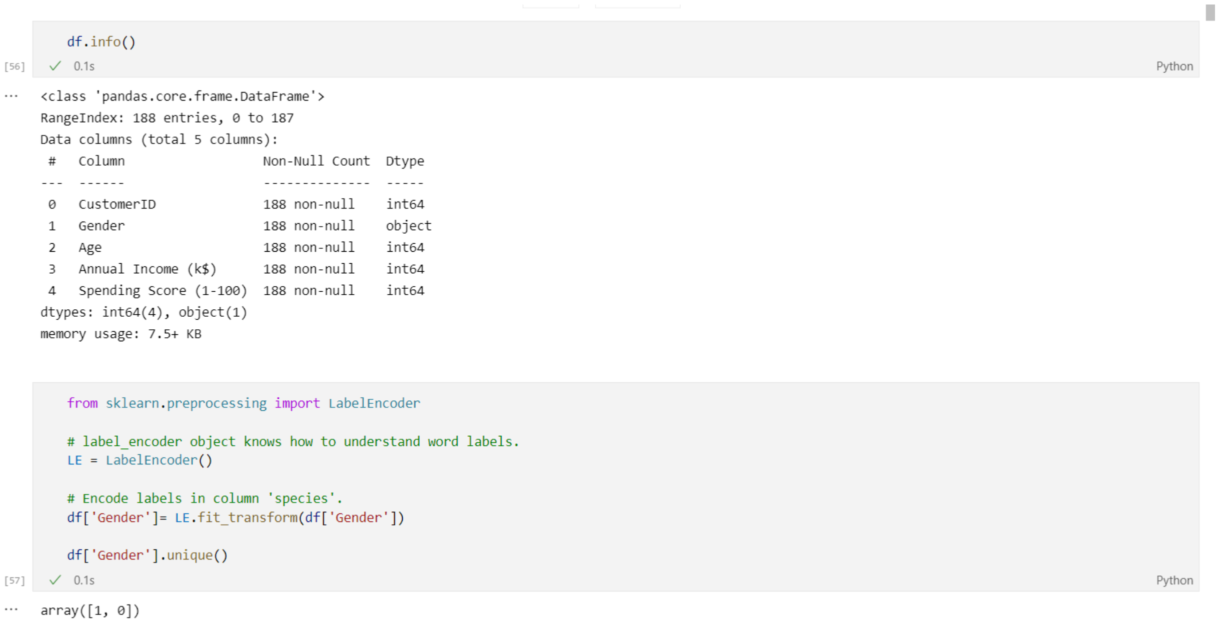
```python
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.cluster import DBSCAN
from sklearn.cluster import AgglomerativeClustering
from sklearn.cluster import KMeans
from sklearn.preprocessing import MinMaxScaler
from sklearn import preprocessing
import matplotlib.pyplot as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

[54] ✓ 0.6s Python

```python
df = pd.read_csv('Mall_Customers.csv')
df.head()
```

[55] ✓ 0.2s Python

	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
4	5	Female	31	17	40



```python
df.info()
```

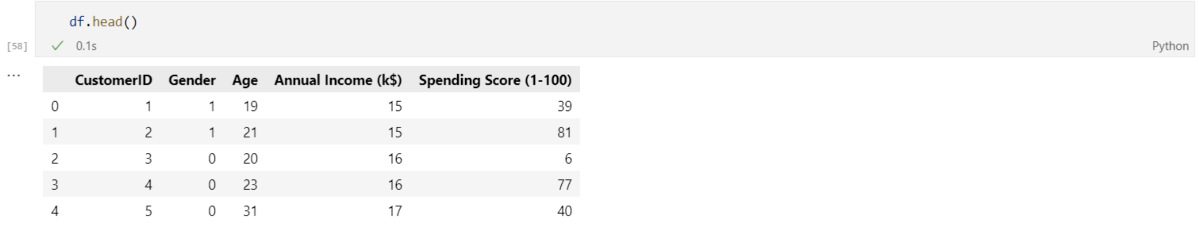
[56] ✓ 0.1s Python

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 188 entries, 0 to 187
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype 
---  ------                  --------------  ----- 
 0   CustomerID              188 non-null    int64 
 1   Gender                  188 non-null    object
 2   Age                     188 non-null    int64 
 3   Annual Income (k$)      188 non-null    int64 
 4   Spending Score (1-100)  188 non-null    int64 
dtypes: int64(4), object(1)
memory usage: 7.5+ KB
```

```python
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()
```

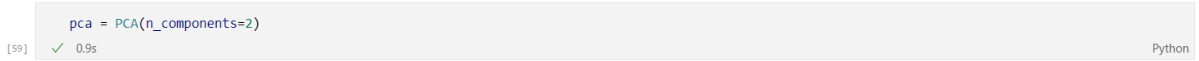
[57] ✓ 0.1s Python

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

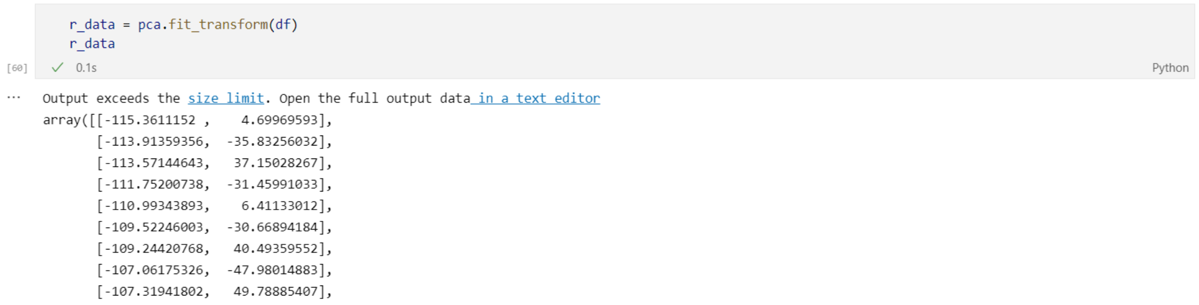


```python
df.head()
```

	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
4	5	0	31	17	40

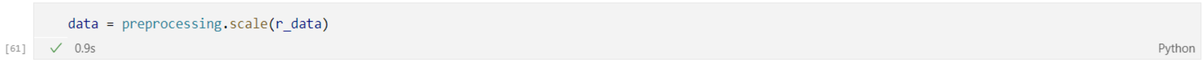


```python
pca = PCA(n_components=2)
```

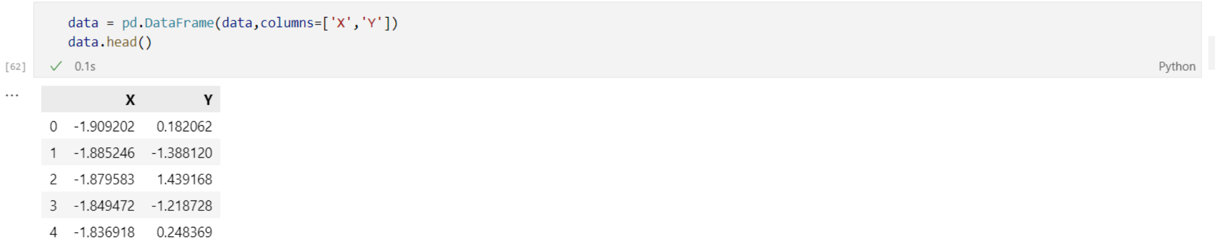


```python
r_data = pca.fit_transform(df)
r_data
```

```
Output exceeds the size limit. Open the full output data in a text editor
array([[-115.3611152 ,    4.69969593],
       [-113.91359356,  -35.83256032],
       [-113.57144643,   37.15028267],
       [-111.75200738,  -31.45991033],
       [-110.99343893,    6.41133012],
       [-109.52246003,  -30.66894184],
       [-109.24420768,   40.49359552],
       [-107.06175326,  -47.98014883],
       [-107.31941802,   49.78885407],
```

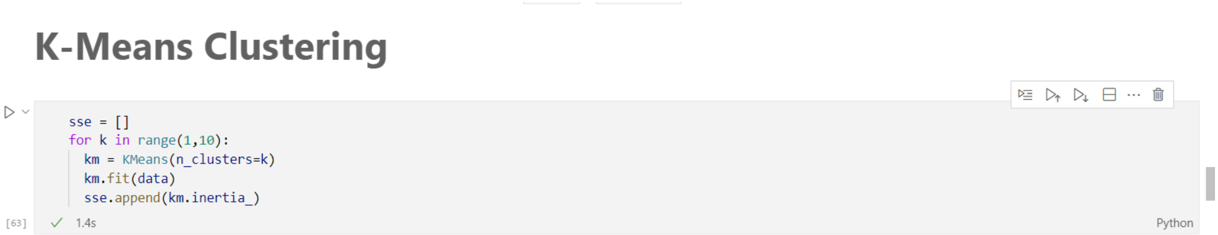


```python
data = preprocessing.scale(r_data)
```



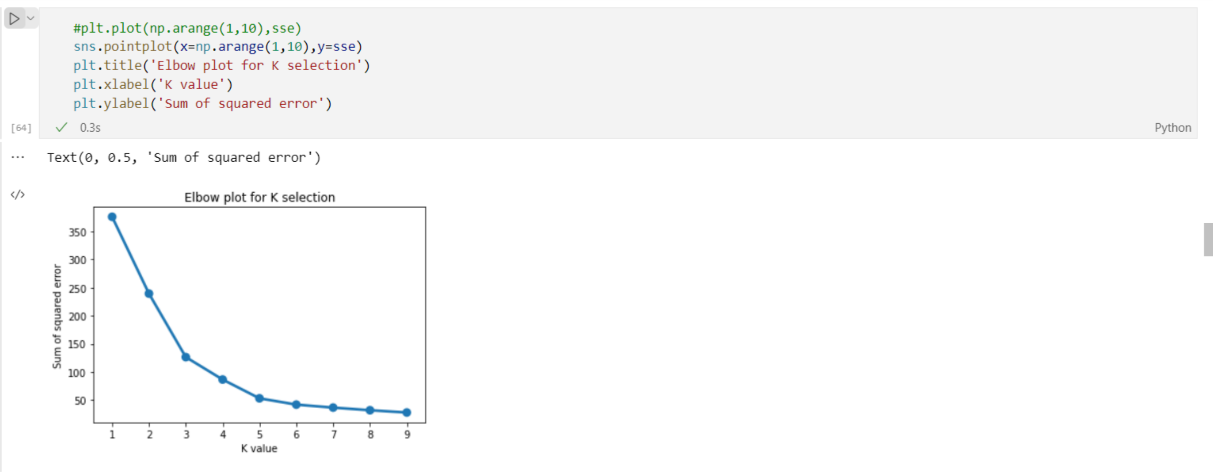
```python
data = pd.DataFrame(data,columns=['X','Y'])
data.head()
```

	X	Y
0	-1.909202	0.182062
1	-1.885246	-1.388120
2	-1.879583	1.439168
3	-1.849472	-1.218728
4	-1.836918	0.248369



K-Means Clustering

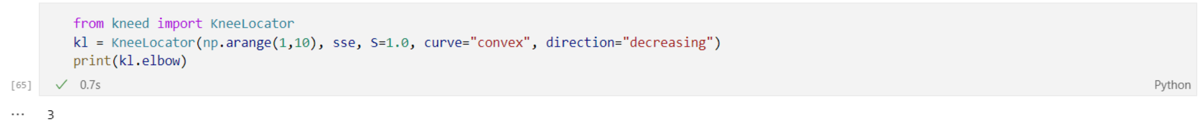
```python
sse = []
for k in range(1,10):
    km = KMeans(n_clusters=k)
    km.fit(data)
    sse.append(km.inertia_)
```



```
#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')
```

[64] ✓ 0.3s Python

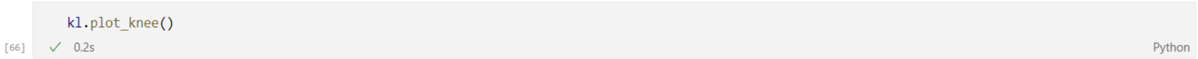
```
Text(0, 0.5, 'Sum of squared error')
```



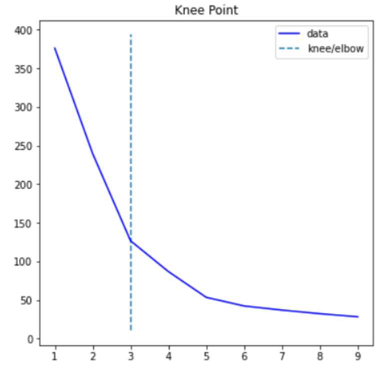
```
from kneed import KneeLocator
kl = KneeLocator(np.arange(1,10), sse, S=1.0, curve="convex", direction="decreasing")
print(kl.elbow)
```

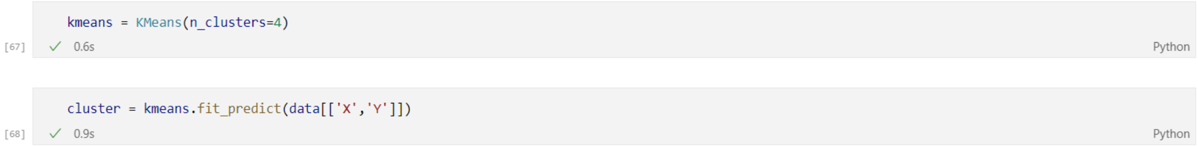
[65] ✓ 0.7s Python

```
3
```



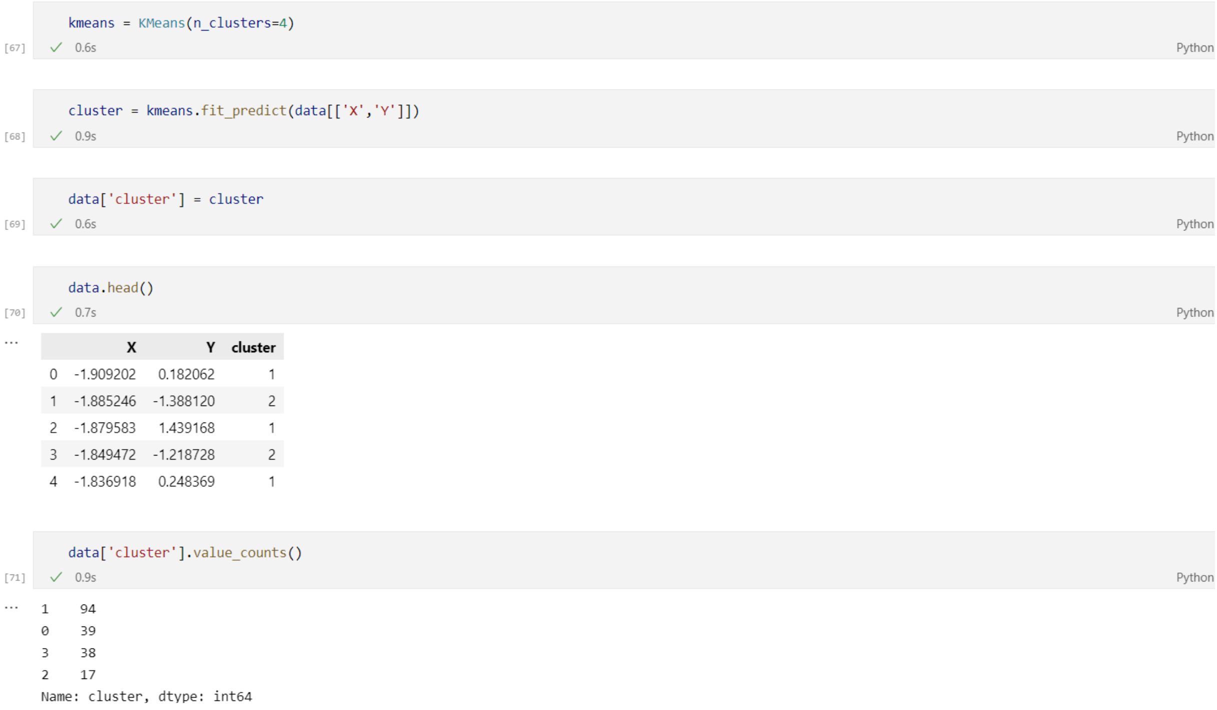
```
[66]  kl.plot_knee()
      ✓ 0.2s                                                    Python
```





```
[67]  kmeans = KMeans(n_clusters=4)
      ✓ 0.6s                                                    Python
```

```
[68]  cluster = kmeans.fit_predict(data[['X','Y']])
      ✓ 0.9s                                                    Python
```



```
[67]  kmeans = KMeans(n_clusters=4)
      ✓ 0.6s                                                    Python
```

```
[68]  cluster = kmeans.fit_predict(data[['X','Y']])
      ✓ 0.9s                                                    Python
```

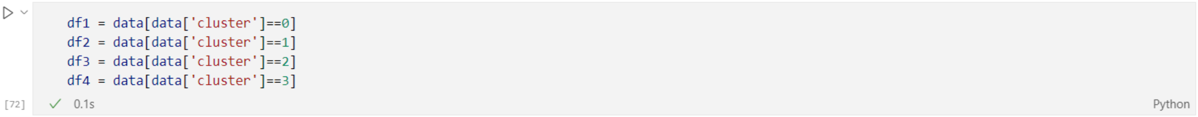
```
[69]  data['cluster'] = cluster
      ✓ 0.6s                                                    Python
```

```
[70]  data.head()
      ✓ 0.7s                                                    Python
```

	X	Y	cluster
0	-1.909202	0.182062	1
1	-1.885246	-1.388120	2
2	-1.879583	1.439168	1
3	-1.849472	-1.218728	2
4	-1.836918	0.248369	1

```
[71]  data['cluster'].value_counts()
      ✓ 0.9s                                                    Python
```

```
1    94
0    39
3    38
2    17
Name: cluster, dtype: int64
```



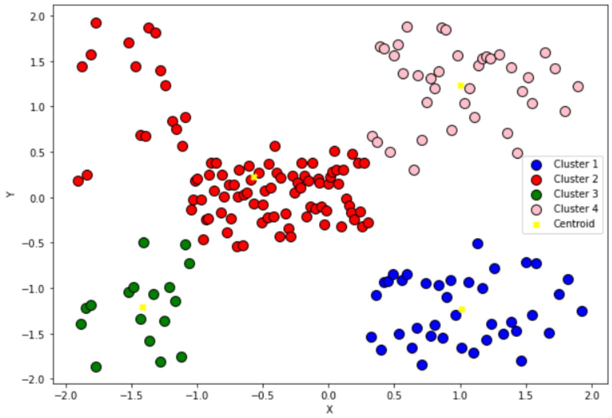
```
df1 = data[data['cluster']==0]
df2 = data[data['cluster']==1]
df3 = data[data['cluster']==2]
df4 = data[data['cluster']==3]
```

```
[72]  ✓ 0.1s                                                    Python
```

```python
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()
```

[73] ✓ 0.3s Python

```
<matplotlib.legend.Legend at 0x1bd5556d670>
```



DBSCAN Clustering

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

[74] ✓ 0.6s Python

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

[75] ✓ 0.6s Python

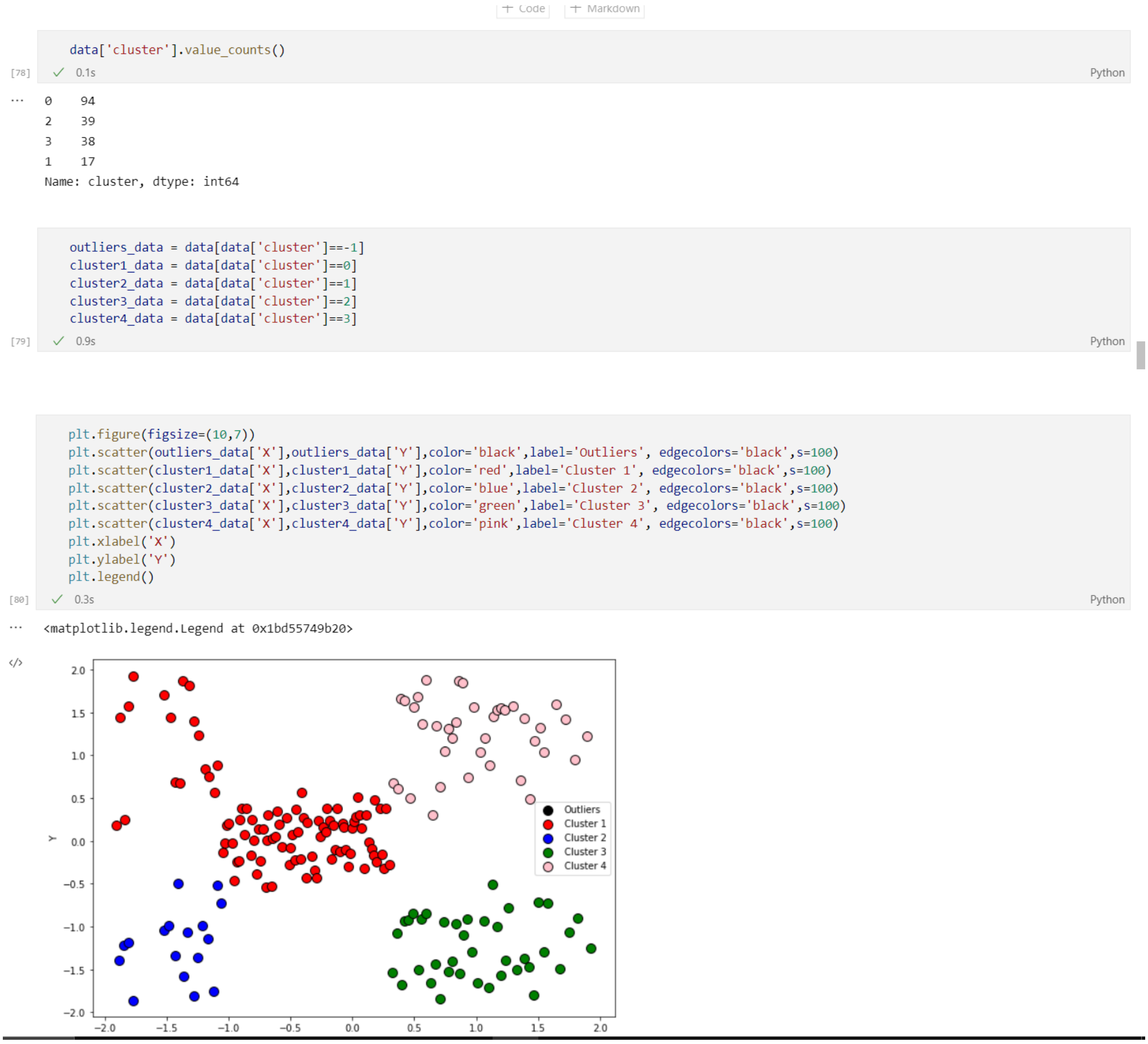
```python
data['cluster']=pr
```

[76] ✓ 0.6s Python

```python
data.head()
```

[77] ✓ 0.6s Python

	X	Y	cluster
0	-1.909202	0.182062	0
1	-1.885246	-1.388120	1
2	-1.879583	1.439168	0
3	-1.849472	-1.218728	1
4	-1.836918	0.248369	0



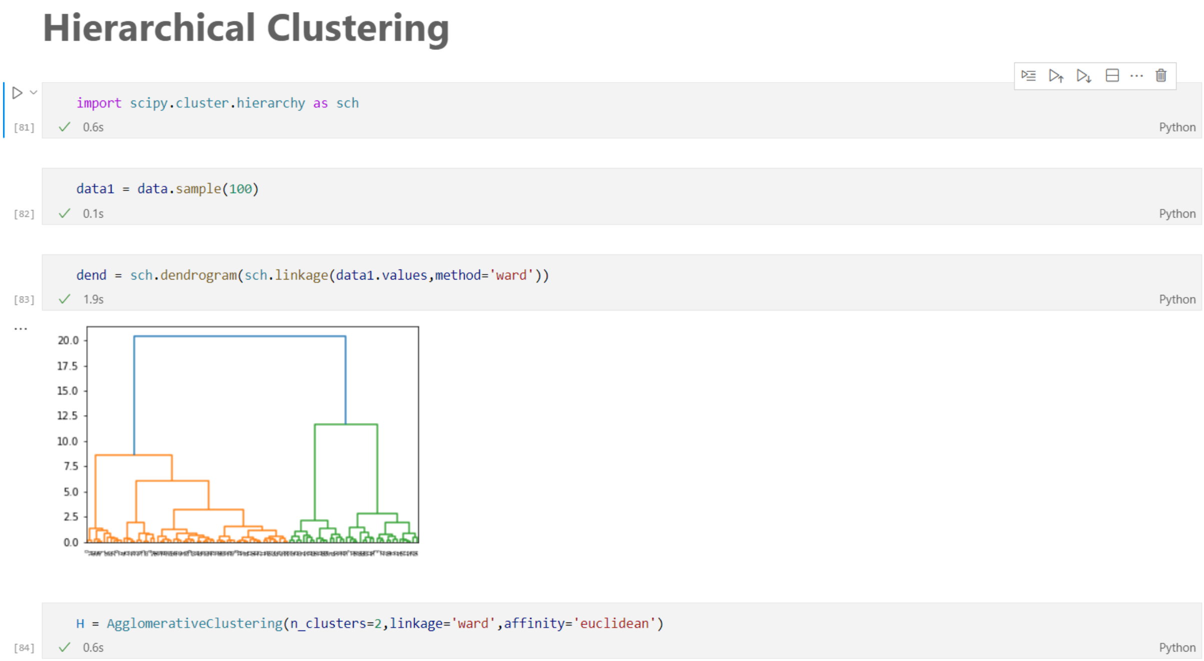
```
data['cluster'].value_counts()
```

```
0    94
2    39
3    38
1    17
Name: cluster, dtype: int64
```

```
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 0x1bd55749b20>
```



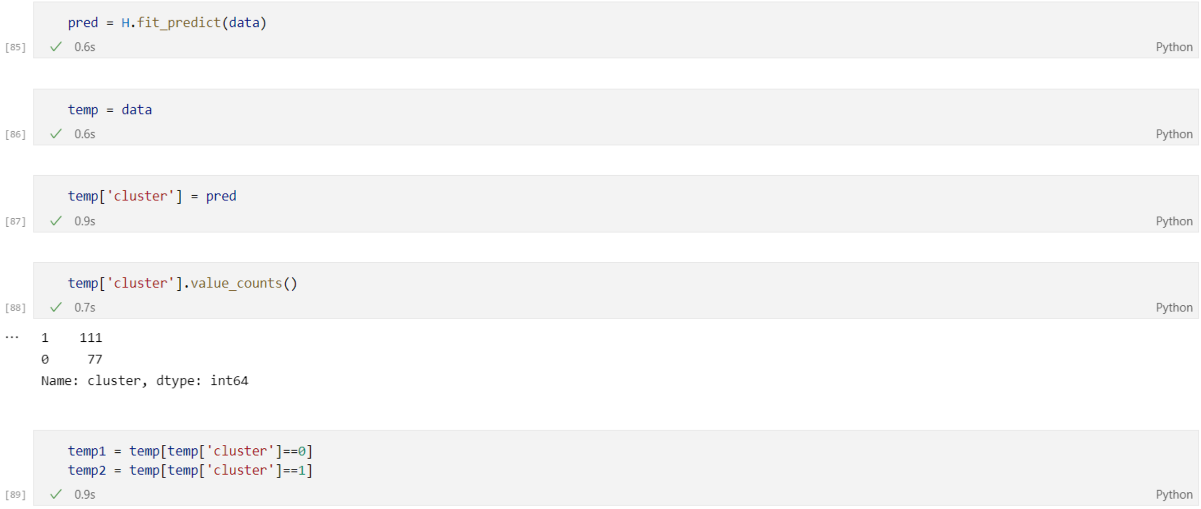
Hierarchical Clustering

```
import scipy.cluster.hierarchy as sch
```
[81] ✓ 0.6s Python

```
data1 = data.sample(100)
```
[82] ✓ 0.1s Python

```
dend = sch.dendrogram(sch.linkage(data1.values,method='ward'))
```
[83] ✓ 1.9s Python

```
H = AgglomerativeClustering(n_clusters=2,linkage='ward',affinity='euclidean')
```
[84] ✓ 0.6s Python



```
pred = H.fit_predict(data)
```
[85] ✓ 0.6s Python

```
temp = data
```
[86] ✓ 0.6s Python

```
temp['cluster'] = pred
```
[87] ✓ 0.9s Python

```
temp['cluster'].value_counts()
```
[88] ✓ 0.7s Python

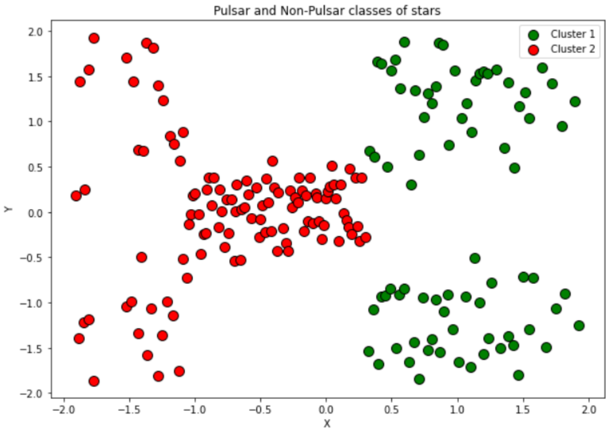
```
1    111
0     77
Name: cluster, dtype: int64
```

```
temp1 = temp[temp['cluster']==0]
temp2 = temp[temp['cluster']==1]
```
[89] ✓ 0.9s Python

```
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()
```

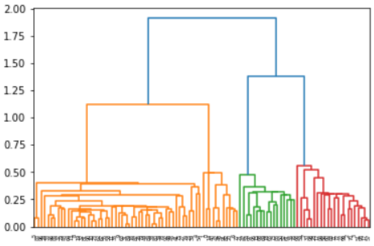
[90] ✓ 0.3s Python

```
<matplotlib.legend.Legend at 0x1bd55a4e7c0>
```



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

[91] ✓ 2.1s Python



```
H = AgglomerativeClustering(n_clusters=2,linkage='single',affinity='euclidean')
```

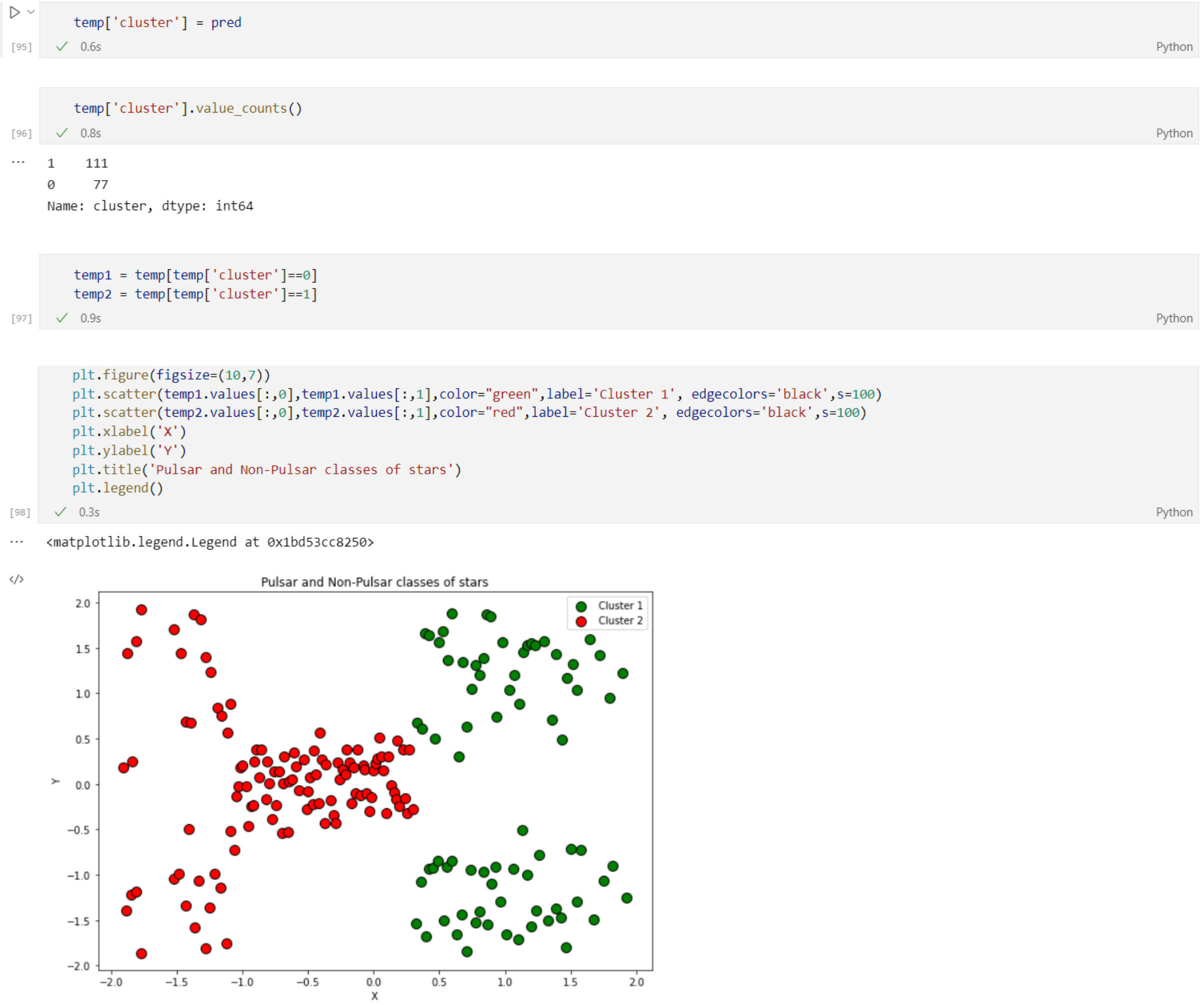
[92] ✓ 0.6s Python

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

[93] ✓ 0.8s Python

```
temp = data
```

[94] ✓ 0.6s Python



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

[95] ✓ 0.6s Python

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

[96] ✓ 0.8s Python

```
1    111
0     77
Name: cluster, dtype: int64
```

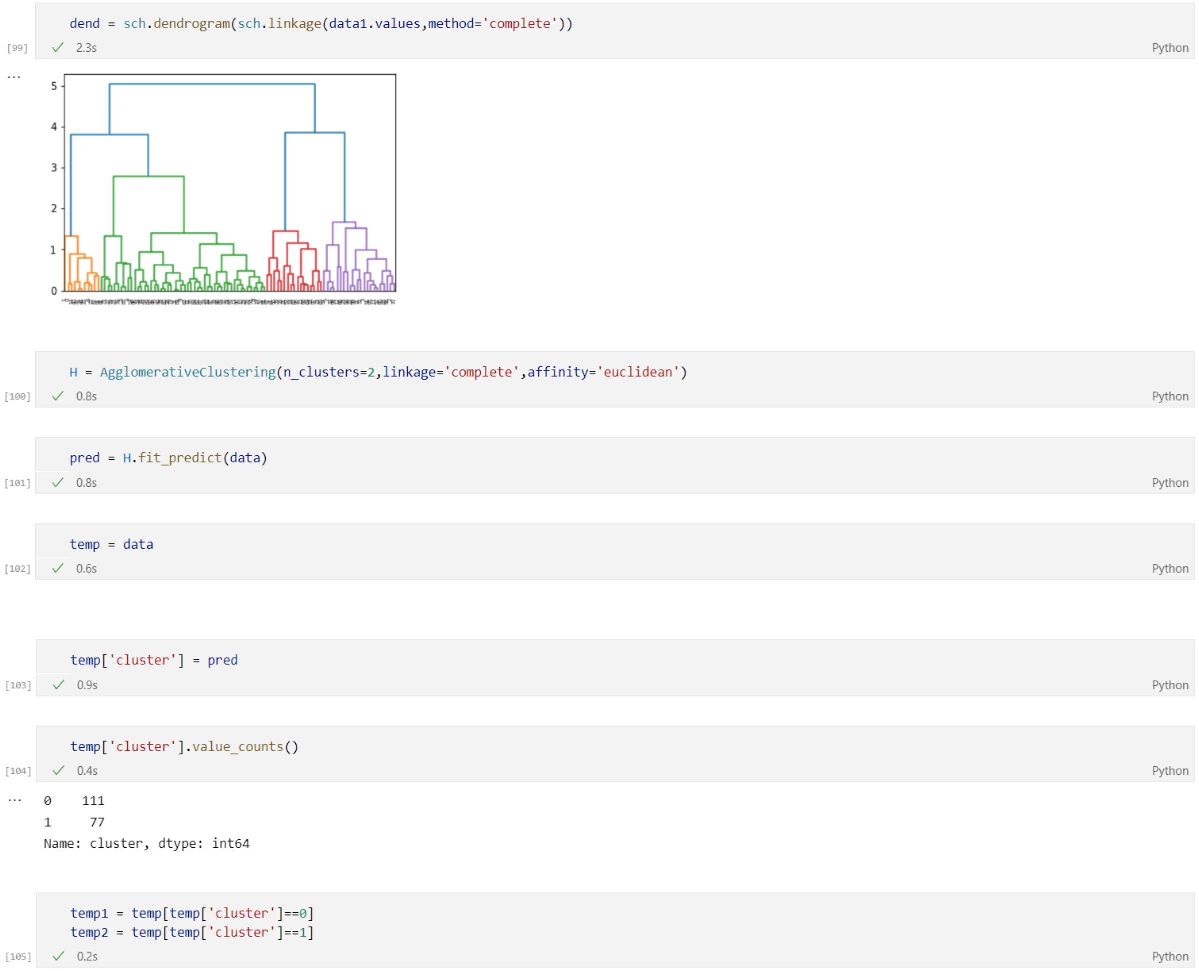
```
temp1 = temp[temp['cluster']==0]
temp2 = temp[temp['cluster']==1]
```

[97] ✓ 0.9s Python

```
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()
```

[98] ✓ 0.3s Python

```
<matplotlib.legend.Legend at 0x1bd53cc8250>
```



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

[99] ✓ 2.3s Python

```
H = AgglomerativeClustering(n_clusters=2,linkage='complete',affinity='euclidean')
```

[100] ✓ 0.8s Python

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

[101] ✓ 0.8s Python

```
temp = data
```

[102] ✓ 0.6s Python

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

[103] ✓ 0.9s Python

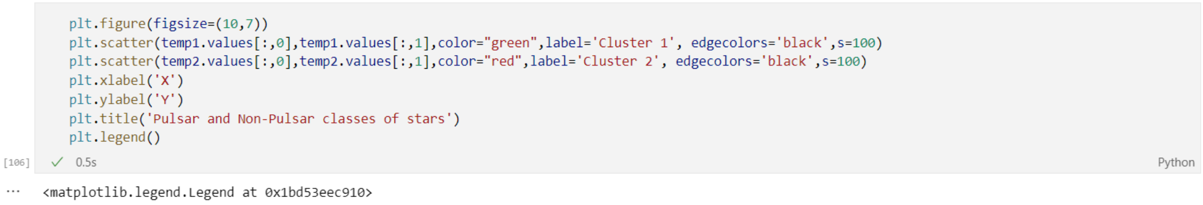
```
temp['cluster'].value_counts()
```

[104] ✓ 0.4s Python

```
0    111
1     77
Name: cluster, dtype: int64
```

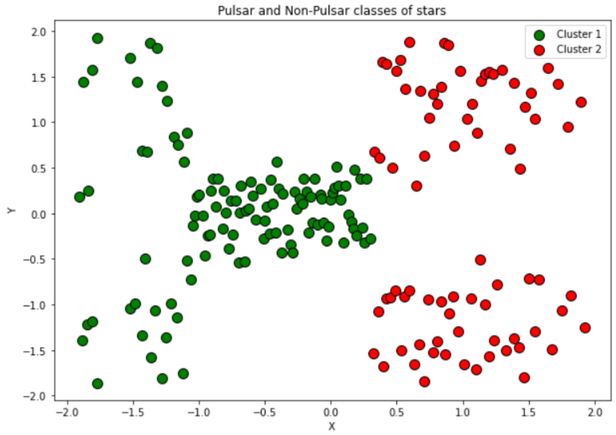
```
temp1 = temp[temp['cluster']==0]
temp2 = temp[temp['cluster']==1]
```

[105] ✓ 0.2s Python



```
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 0x1bd53eec910>
```



Pulsar and Non-Pulsar classes of stars
● Cluster 1
● Cluster 2
Y
X