Name :- Aniket Kumar

Sr. No :- 30

Roll No :- PF45

PRN No :- 1032171203.

AML LAB ASSIGNMENT 06

Problem Statement :- Implementation and Comparison of various clustering techniques: k-means, Hierarchical.

Theory :-

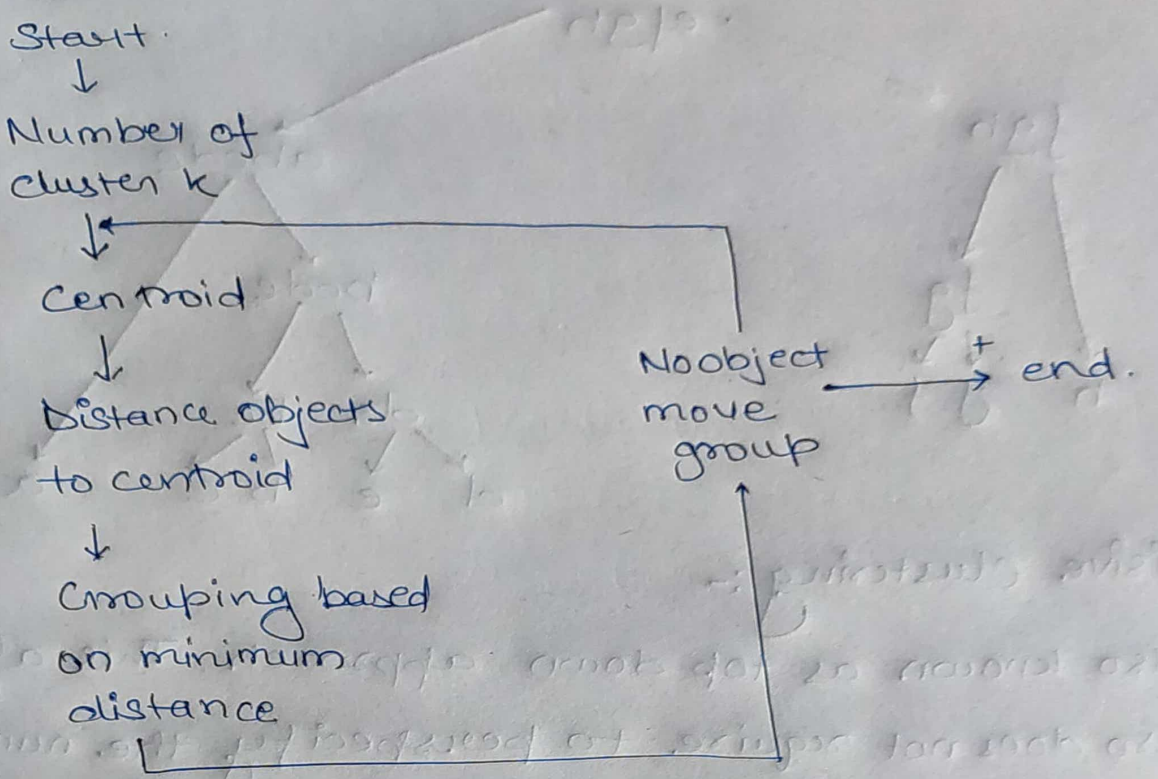
Clustering Algorithm.

Clustering is very much important as it determines the intrinsic grouping among the unlabelled data present.

K-Means clustering :-

It is the simplest supervised learning algorithm that solves clustering problem. The algorithm will categorize the items into k groups of similarity.

flow diagram.



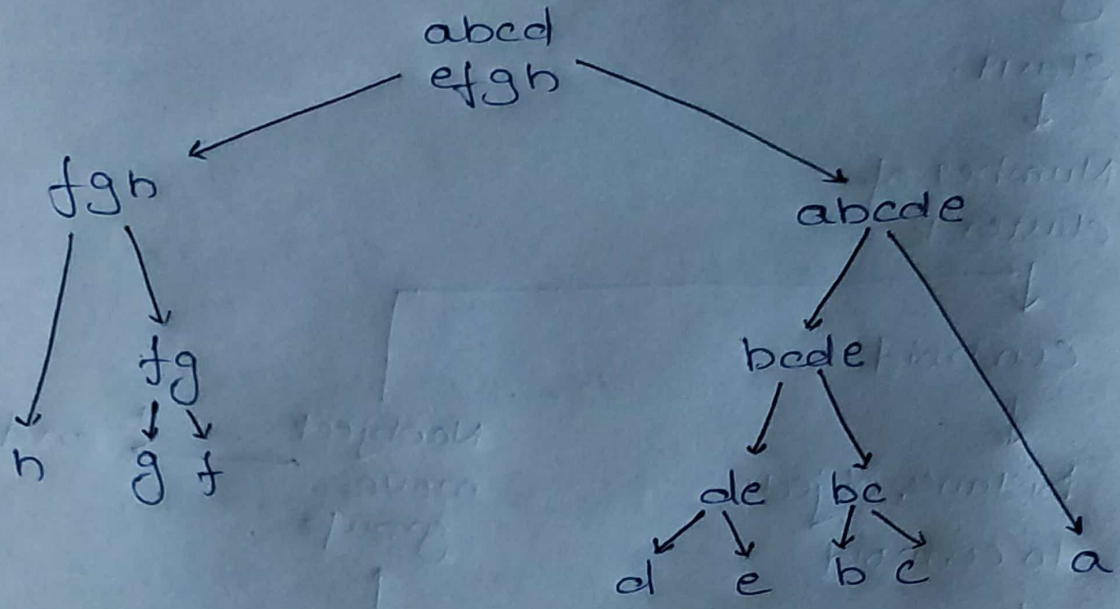
- Euclidean Distance.

$$\sqrt{(x_2-x_1)^2 + (y_2-y_1)^2}$$

Hierarchical clustering.

It is a method of cluster analysis which seek to built a hierarchy clusters. i.e tree type structure based on the hierarchy.

Agglomerative Hierarchical clustering: Also known as bottom-up approach or hierarchical agglomerative clustering.



Divisive clustering :-

Also known as top down approach. This algorithm also does not require to perspecify the number of clusters. It requires a method for ~~spl~~ting a cluster that contains the whole data and proceeds by splitting clusters recursively until individual data have been split into singleton cluster.

Conclusion :-

Hence, we implemented different clustering algorithms.

NAME : Aniket Kumar

S.no : 30

PRN : 1032171203

ROLL : PF45

Subject : AML Lab Assignment 6

In [11]:
```
#K-Means
```

In [1]:
```
# importing required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.cluster import KMeans
```

In [3]:
```
# reading the data and looking at the first five rows of the data
data=pd.read_csv(r"C:\Users\Aniket\Desktop\Aml\6\Wholesale customers data.csv")
data.head()
```

Out[3]:

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
0	2	3	12669	9656	7561	214	2674	1338
1	2	3	7057	9810	9568	1762	3293	1776
2	2	3	6353	8808	7684	2405	3516	7844
3	1	3	13265	1196	4221	6404	507	1788
4	2	3	22615	5410	7198	3915	1777	5185

In [4]:
```
# statistics of the data
data.describe()
```

Out[4]:

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
count	440.000000	440.000000	440.000000	440.000000	440.000000	440.000000	440.000000	440.000000
mean	1.322727	2.543182	12000.297727	5796.265909	7951.277273	3071.931818	2881.493182	1524.870455
std	0.468052	0.774272	12647.328865	7380.377175	9503.162829	4854.673333	4767.854448	2820.105937
min	1.000000	1.000000	3.000000	55.000000	3.000000	25.000000	3.000000	3.000000
25%	1.000000	2.000000	3127.750000	1533.000000	2153.000000	742.250000	256.750000	408.250000
50%	1.000000	3.000000	8504.000000	3627.000000	4755.500000	1526.000000	816.500000	965.500000
75%	2.000000	3.000000	16933.750000	7190.250000	10655.750000	3554.250000	3922.000000	1820.250000
max	2.000000	3.000000	112151.000000	73498.000000	92780.000000	60869.000000	40827.000000	47943.000000

In [5]:
```
# standardizing the data
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
data_scaled = scaler.fit_transform(data)

# statistics of scaled data
pd.DataFrame(data_scaled).describe()
```

Out[5]:

	0	1	2	3	4	5	6	7
count	4.400000e+02	4.400000e+02	4.400000e+02	4.400000e+02	4.400000e+02	4.400000e+02	4.400000e+02	4.400000e+02
mean	-2.452584e-16	-5.737834e-16	-2.422305e-17	-1.589638e-17	-6.030530e-17	1.135455e-17	-1.917658e-17	-8.276208e-17
std	1.001138e+00	1.001138e+00	1.001138e+00	1.001138e+00	1.001138e+00	1.001138e+00	1.001138e+00	1.001138e+00
min	-6.902971e-01	-1.995342e+00	-9.496831e-01	-7.787951e-01	-8.373344e-01	-6.283430e-01	-6.044165e-01	-5.402644e-01
25%	-6.902971e-01	-7.023369e-01	-7.023339e-01	-5.783063e-01	-6.108364e-01	-4.804306e-01	-5.511349e-01	-3.964005e-01
50%	-6.902971e-01	5.906683e-01	-2.767602e-01	-2.942580e-01	-3.366684e-01	-3.188045e-01	-4.336004e-01	-1.985766e-01
75%	1.448652e+00	5.906683e-01	3.905226e-01	1.890921e-01	2.849105e-01	9.946441e-02	2.184822e-01	1.048598e-01
max	1.448652e+00	5.906683e-01	7.927738e+00	9.183650e+00	8.936528e+00	1.191900e+01	7.967672e+00	1.647845e+01

In [6]:
```
# defining the kmeans function with initialization as k-means++
kmeans = KMeans(n_clusters=2, init='k-means++')

# fitting the k means algorithm on scaled data
kmeans.fit(data_scaled)
```

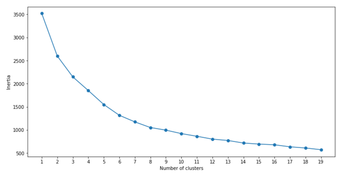
Out[6]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=2, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=None, tol=0.0001, verbose=0)
```

In [7]:
```
# inertia on the fitted data
kmeans.inertia_
```

Out[7]:
```
2599.38555935614
```

In [14]:
```
# fitting multiple k-means algorithms and storing the values in an empty list
SSE = []
for cluster in range(1,20):
    kmeans = KMeans(n_jobs = -1, n_clusters = cluster, init='k-means++')
    kmeans.fit(data_scaled)
    SSE.append(kmeans.inertia_)

# converting the results into a dataframe and plotting them
frame = pd.DataFrame({'Cluster':range(1,20), 'SSE':SSE})
plt.figure(figsize=(12,6))
plt.plot([1 for i in range(1,20)] ,SSE, marker='o')
plt.xlabel('Number of clusters')
plt.xticks(range(1,20))
plt.ylabel('Inertia')
```

Out[14]:
```
Text(0, 0.5, 'Inertia')
```



In [15]:
```
# k means using 7 clusters and k-means++ initialization
kmeans = KMeans(n_jobs = -1, n_clusters = 7, init='k-means++')
kmeans.fit(data_scaled)
pred = kmeans.predict(data_scaled)
```

In [16]:
```
frame = pd.DataFrame(data_scaled)
frame['cluster'] = pred
frame['cluster'].value_counts()
```

Out[16]:
```
4    172
2     94
3     87
1     44
0     36
6      6
5      1
Name: cluster, dtype: int64
```

In [12]:
```
#Hierarchical Clustering
```

In [17]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [18]:
```
# reading the data and looking at the first five rows of the data
data=pd.read_csv(r"C:\Users\Aniket\Desktop\Aml\6\Wholesale customers data.csv")
data.head()
```

Out[18]:

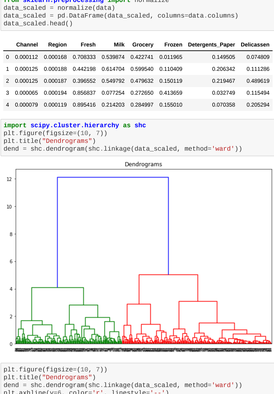
	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
0	2	3	12669	9656	7561	214	2674	1338
1	2	3	7057	9810	9568	1762	3293	1776
2	2	3	6353	8808	7684	2405	3516	7844
3	1	3	13265	1196	4221	6404	507	1788
4	2	3	22615	5410	7198	3915	1777	5185

In [19]:
```
from sklearn.preprocessing import normalize
data_scaled = normalize(data)
data_scaled = pd.DataFrame(data_scaled, columns=data.columns)
data_scaled.head()
```

Out[19]:

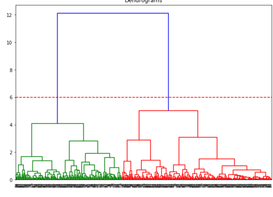
	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
0	0.000112	0.000168	0.708333	0.539874	0.422741	0.011965	0.149505	0.074809
1	0.000125	0.000188	0.442198	0.614704	0.599540	0.110409	0.206342	0.111286
2	0.000125	0.000187	0.396552	0.549792	0.479632	0.150119	0.219467	0.489619
3	0.000065	0.000194	0.856837	0.077254	0.272650	0.413659	0.032749	0.115494
4	0.000079	0.000119	0.895416	0.214203	0.284997	0.155010	0.070358	0.205294

In [20]:
```
import scipy.cluster.hierarchy as shc
plt.figure(figsize=(10, 7))
plt.title("Dendrograms")
dend = shc.dendrogram(shc.linkage(data_scaled, method='ward'))
```



In [21]:
```
plt.figure(figsize=(10, 7))
plt.title("Dendrograms")
dend = shc.dendrogram(shc.linkage(data_scaled, method='ward'))
plt.axhline(y=6, color='r', linestyle='--')
```

Out[21]:
```
<matplotlib.lines.Line2D at 0x1762d4d1b08>
```

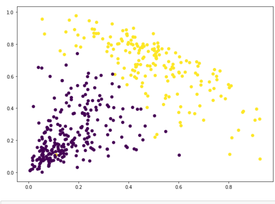


In [22]:
```
from sklearn.cluster import AgglomerativeClustering
cluster = AgglomerativeClustering(n_clusters=2, affinity='euclidean', linkage='ward')
cluster.fit_predict(data_scaled)
```

Out[22]:
```
array([1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0,
       1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1,
       0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1,
       0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1,
       0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
       0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0,
       0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0,
       1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0,
       0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1,
       1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1],
      dtype=int64)
```

In [23]:
```
plt.figure(figsize=(10, 7))
plt.scatter(data_scaled['Milk'], data_scaled['Grocery'], c=cluster.labels_)
```

Out[23]:
```
<matplotlib.collections.PathCollection at 0x1762d4a8668>
```



In [ ]:
```
```

Implementation of Kmeans, Hierarchical, DBScan and Spectral Clustering

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

In [2]:
```
data = pd.read_csv("cluster.csv")
data.head()
```

Out[2]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_me
0	842302	M	17.99	10.38	122.80	1001.0	0.118
1	842517	M	20.57	17.77	132.90	1326.0	0.084
2	84300903	M	19.69	21.25	130.00	1203.0	0.109
3	84348301	M	11.42	20.38	77.58	386.1	0.142
4	84358402	M	20.29	14.34	135.10	1297.0	0.100

5 rows × 33 columns

In [3]:
```
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt

data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 33 columns):
id                          569 non-null int64
diagnosis                   569 non-null object
```

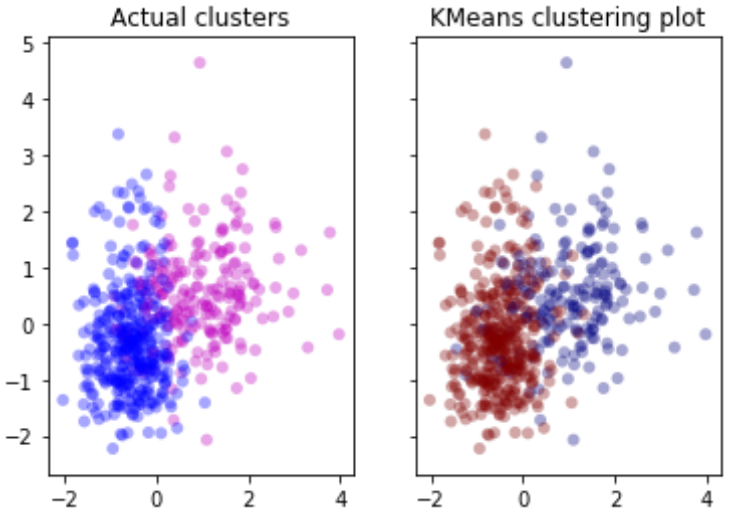
```
radius_mean                569 non-null float64
texture_mean               569 non-null float64
perimeter_mean             569 non-null float64
area_mean                  569 non-null float64
smoothness_mean            569 non-null float64
compactness_mean           569 non-null float64
concavity_mean             569 non-null float64
concave points_mean        569 non-null float64
symmetry_mean              569 non-null float64
fractal_dimension_mean     569 non-null float64
radius_se                  569 non-null float64
texture_se                 569 non-null float64
perimeter_se               569 non-null float64
area_se                    569 non-null float64
smoothness_se              569 non-null float64
compactness_se             569 non-null float64
concavity_se               569 non-null float64
concave points_se          569 non-null float64
symmetry_se                569 non-null float64
fractal_dimension_se       569 non-null float64
radius_worst               569 non-null float64
texture_worst              569 non-null float64
perimeter_worst            569 non-null float64
area_worst                 569 non-null float64
smoothness_worst           569 non-null float64
compactness_worst          569 non-null float64
concavity_worst            569 non-null float64
concave points_worst       569 non-null float64
symmetry_worst             569 non-null float64
fractal_dimension_worst    569 non-null float64
Unnamed: 32                0 non-null float64
dtypes: float64(31), int64(1), object(1)
memory usage: 146.8+ KB
```

In [4]:
```
cols_drop = ['id', 'Unnamed: 32']
data = data.drop(cols_drop, axis=1)
data['diagnonis'] = data['diagnosis'].map({'M':1,'B':0})
X = data.drop('diagnosis', axis=1).values
X = StandardScaler().fit_transform(X)
```

In [5]:
```
#1 KMeans Clustering
from sklearn.cluster import KMeans
km = KMeans(n_clusters=2, init="k-means++", n_init=8)
km_pred = km.fit_predict(X)

f, (ax1, ax2) = plt.subplots(1, 2, sharey=True)

ax1.scatter(X[:,0], X[:,1], c=data["diagnosis"], cmap="jet", edgecolor=
"None", alpha=0.35)
ax1.set_title("Actual clusters")

ax2.scatter(X[:,0], X[:,1], c=km_pred, cmap="jet", edgecolor="None", al
pha=0.35)
ax2.set_title("KMeans clustering plot")
```

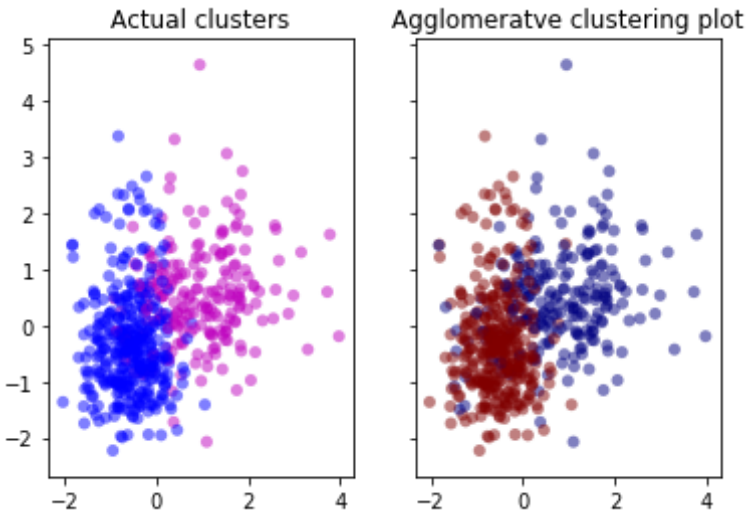
```
C:\Users\admin\Anaconda3\lib\site-packages\ipykernel_launcher.py:8: Mat
plotlibDeprecationWarning: Support for uppercase single-letter colors i
s deprecated since Matplotlib 3.1 and will be removed in 3.3; please us
e lowercase instead.
```

Out[5]:
```
Text(0.5, 1.0, 'KMeans clustering plot')
```



In [6]:
```
#2 Hierarchical Agglomerative Clustering

from sklearn.cluster import AgglomerativeClustering
ac = AgglomerativeClustering(n_clusters=2, linkage="ward")
ac_pred = ac.fit_predict(X)

# Scatter plots
f, (ax1, ax2) = plt.subplots(1, 2, sharey=True)

ax1.scatter(X[:,0], X[:,1], c=data["diagnosis"], cmap="jet", edgecolor=
"None", alpha=0.5)
ax1.set_title("Actual clusters")

ax2.scatter(X[:,0], X[:,1], c=ac_pred, cmap="jet", edgecolor="None", al
pha=0.5)
ax2.set_title("Agglomeratve clustering plot")
```

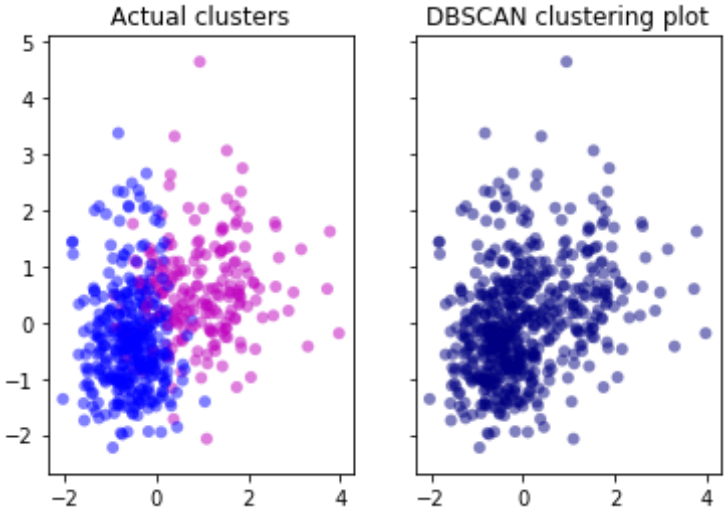
Out[6]:
```
Text(0.5, 1.0, 'Agglomeratve clustering plot')
```



In [7]:
```
#3 DBSCAN (Density-Based Clustering of Applications with Noise)

from sklearn.cluster import DBSCAN
dbs = DBSCAN(eps=0.2, min_samples=6)
```

```
dbs_pred = dbs.fit_predict(X)

# Scatter plots
f, (ax1, ax2) = plt.subplots(1, 2, sharey=True)

ax1.scatter(X[:,0], X[:,1], c=data["diagnosis"], cmap="jet", edgecolor=
"None", alpha=0.5)
ax1.set_title("Actual clusters")

ax2.scatter(X[:,0], X[:,1], c=dbs_pred, cmap="jet", edgecolor="None", a
lpha=0.5)
ax2.set_title("DBSCAN clustering plot")
```

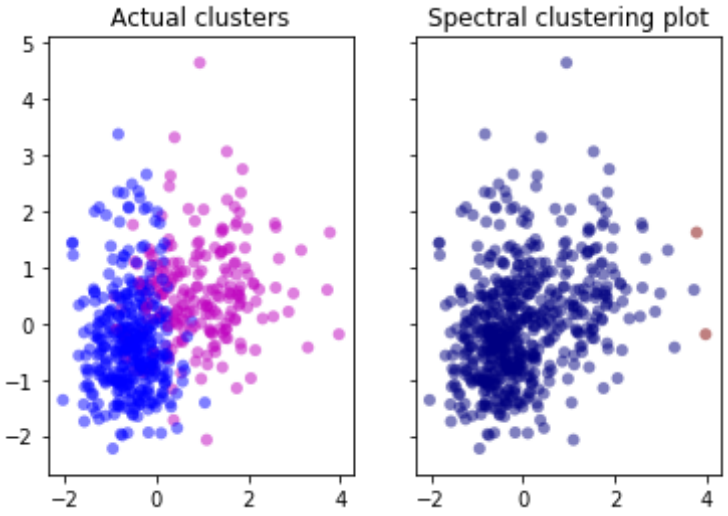
Out[7]: Text(0.5, 1.0, 'DBSCAN clustering plot')



In [8]:
```
#4 Spectral Clustering

from sklearn.cluster import SpectralClustering
sc = SpectralClustering(n_clusters=2, gamma=0.5, affinity="rbf", assign
_labels="discretize")
sc_pred = sc.fit_predict(X)

# Scatter plots
```

```
f, (ax1, ax2) = plt.subplots(1, 2, sharey=True)

ax1.scatter(X[:,0], X[:,1], c=data["diagnosis"], cmap="jet", edgecolor=
"None", alpha=0.5)
ax1.set_title("Actual clusters")

ax2.scatter(X[:,0], X[:,1], c=sc_pred, cmap="jet", edgecolor="None", al
pha=0.5)
ax2.set_title("Spectral clustering plot")
```

Out[8]: Text(0.5, 1.0, 'Spectral clustering plot')



In [ ]: