深 圳 大 学 实 验 报 告

课程名称：	机器学习
实验项目名称 ：	Machine Learning Task 3
学院 ：	电子与信息工程学院
专业 ：	电子信息工程
指导教师：	欧阳乐
报告人：	余韦藩　学号：2020285102
班级：	文华班
实验时间：	2022.5.23 ——2022.5.29
实验报告提交时间：	2022.5.29

教务处制

Aim of Experiment:

(1) Understand the principle of cluster analysis and learn how to apply the cluster analysis model.

(2) Observe and compare the performance of cluster analysis model training on different datasets which varies from specific data distribution characteristics.

(3) Compare the performance of different cluster analysis model such as K-means clustering and DBSCAN clustering training on the same datasets.

(4) Learn to set the hyperparameter for the specific cluster analysis model.

Experiment Content:

(1) Load the five datasets: noisy_circles.txt, noisy_moons.txt, blobs.txt, aniso.txt and no_structure.txt

(2) Plot scatter diagram to visualize data set.

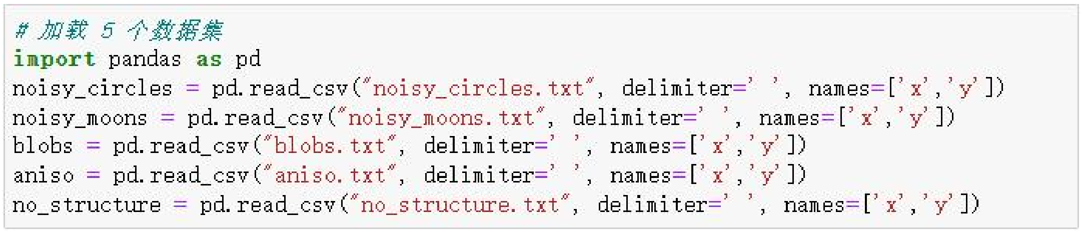
(3) Use k-means clustering for all data, and plot a figure to present the clustering results.

(4) Use DBSCAN (adjust EPS for different data) for all data, and plot a figure to present the clustering results.

Experiment Process:

A. Load the five datasets: noisy_circles.txt, noisy_moons.txt, blobs.txt, aniso.txt and no_structure.txt

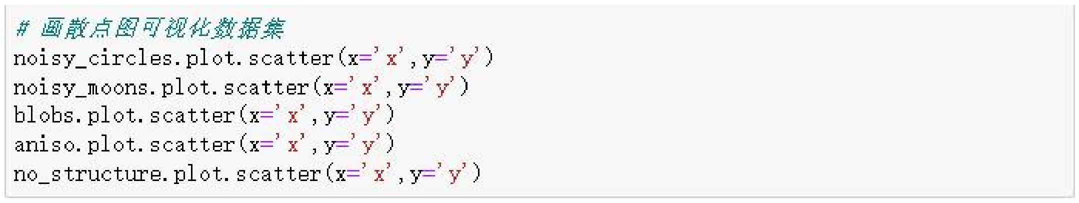
We use pandas.read_csv() function to load the five datasets and assign them to noisy_circles, noisy_moons, blobs, aniso and no_structure respective. The codes are shown in the following figure.



```
# 加载 5 个数据集
import pandas as pd
noisy_circles = pd.read_csv("noisy_circles.txt", delimiter=' ', names=['x','y'])
noisy_moons = pd.read_csv("noisy_moons.txt", delimiter=' ', names=['x','y'])
blobs = pd.read_csv("blobs.txt", delimiter=' ', names=['x','y'])
aniso = pd.read_csv("aniso.txt", delimiter=' ', names=['x','y'])
no_structure = pd.read_csv("no_structure.txt", delimiter=' ', names=['x','y'])
```

B. Plot scatter diagram to visualize data set.

We use plot.scatter() function to visualize the distribution of each datasets. The codes are shown in the following figure.



```
# 画散点图可视化数据集
noisy_circles.plot.scatter(x='x',y='y')
noisy_moons.plot.scatter(x='x',y='y')
blobs.plot.scatter(x='x',y='y')
aniso.plot.scatter(x='x',y='y')
no_structure.plot.scatter(x='x',y='y')
```

C. Use k-means clustering for all data, and plot a figure to present the clustering results.

For noisy_circles, we first use sklearn.cluster.KMeans and set n_cluster=2, max_iter=50 and random_state=100 to cluster the noisy circles datasets and assign it to k_means_noisy_circles. Then we extract the predicted label for each point and combine with the original noisy circles datasets. Finally, we use plot.scatter() function to show the predicted result for each point. Different class is recorded by different color.

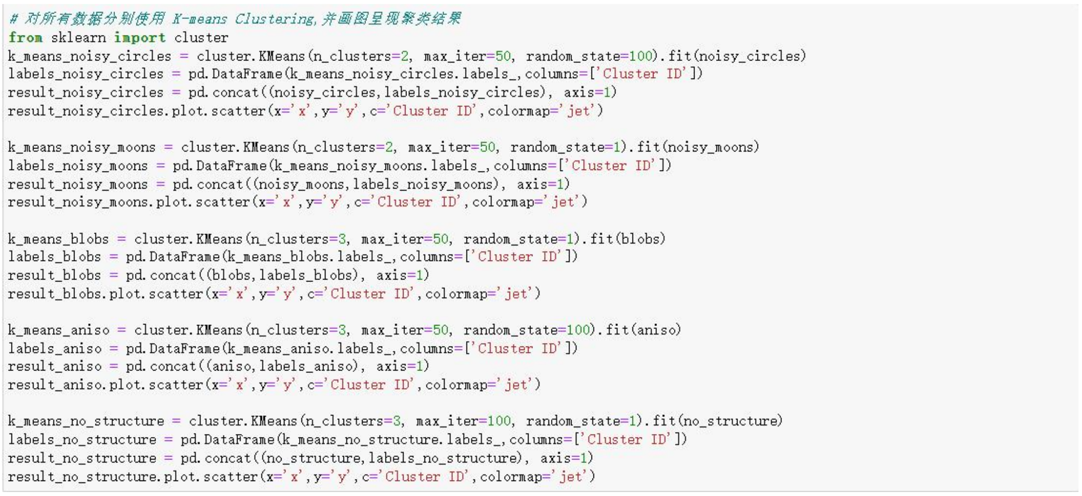
For noisy_moon, we first use sklearn.cluster.KMeans and set n_cluster=2, max_iter=50 and random_state=1 to cluster the noisy moon datasets and assign it to k_means_noisy_moon. Then we extract the predicted label for each point and combine with the original noisy moon datasets. Finally, we use plot.scatter() function to show the predicted result for each point. Different class is recorded by different color.

For blobs, we first use sklearn.cluster.KMeans and set n_cluster=3, max_iter=50 and random_state=1 to cluster the blobs datasets and assign it to k_means_blobs. Then we extract the predicted label for each point and combine with the original blobs datasets. Finally, we use plot.scatter() function to show the predicted result for each point. Different class is recorded by different color.

For aniso, we first use sklearn.cluster.KMeans and set n_cluster=3, max_iter=50 and random_state=1 to cluster the aniso datasets and assign it to k_means_aniso. Then we extract the predicted label for each point and combine with the original aniso datasets. Finally, we use plot.scatter() function to show the predicted result for each point. Different class is recorded by different color.

For no_structure, we first use sklearn.cluster.KMeans and set n_cluster=3, max_iter=100 and random_state=1 to cluster the no_structure datasets and assign it to k_means_no_structure. Then we extract the predicted label for each point and combine with the original no_structure datasets. Finally, we use plot.scatter() function to show the predicted result for each point. Different class is recorded by different color.

All the codes are shown in the following figure.



```python
# 对所有数据分别使用 K-means Clustering,并画图呈现聚类结果
from sklearn import cluster
k_means_noisy_circles = cluster.KMeans(n_clusters=2, max_iter=50, random_state=100).fit(noisy_circles)
labels_noisy_circles = pd.DataFrame(k_means_noisy_circles.labels_,columns=['Cluster ID'])
result_noisy_circles = pd.concat((noisy_circles,labels_noisy_circles), axis=1)
result_noisy_circles.plot.scatter(x='x',y='y',c='Cluster ID',colormap='jet')

k_means_noisy_moons = cluster.KMeans(n_clusters=2, max_iter=50, random_state=1).fit(noisy_moons)
labels_noisy_moons = pd.DataFrame(k_means_noisy_moons.labels_,columns=['Cluster ID'])
result_noisy_moons = pd.concat((noisy_moons,labels_noisy_moons), axis=1)
result_noisy_moons.plot.scatter(x='x',y='y',c='Cluster ID',colormap='jet')

k_means_blobs = cluster.KMeans(n_clusters=3, max_iter=50, random_state=1).fit(blobs)
labels_blobs = pd.DataFrame(k_means_blobs.labels_,columns=['Cluster ID'])
result_blobs = pd.concat((blobs,labels_blobs), axis=1)
result_blobs.plot.scatter(x='x',y='y',c='Cluster ID',colormap='jet')

k_means_aniso = cluster.KMeans(n_clusters=3, max_iter=50, random_state=100).fit(aniso)
labels_aniso = pd.DataFrame(k_means_aniso.labels_,columns=['Cluster ID'])
result_aniso = pd.concat((aniso,labels_aniso), axis=1)
result_aniso.plot.scatter(x='x',y='y',c='Cluster ID',colormap='jet')

k_means_no_structure = cluster.KMeans(n_clusters=3, max_iter=100, random_state=1).fit(no_structure)
labels_no_structure = pd.DataFrame(k_means_no_structure.labels_,columns=['Cluster ID'])
result_no_structure = pd.concat((no_structure,labels_no_structure), axis=1)
result_no_structure.plot.scatter(x='x',y='y',c='Cluster ID',colormap='jet')
```

D. Use DBSCAN (adjust EPS for different data) for all data, and plot a figure to present the clustering results.

For noisy_circles, we first use DBSCAN function and set parameter eps=0.2 and min_sample=3 to cluster the noisy circles datasets and assign it to db_noisy_circles. Then we extract the predicted label for each point and combine with the original noisy circles datasets. Finally, we use plot.scatter() function to show the predicted result for each point. Different class is recorded by different color.

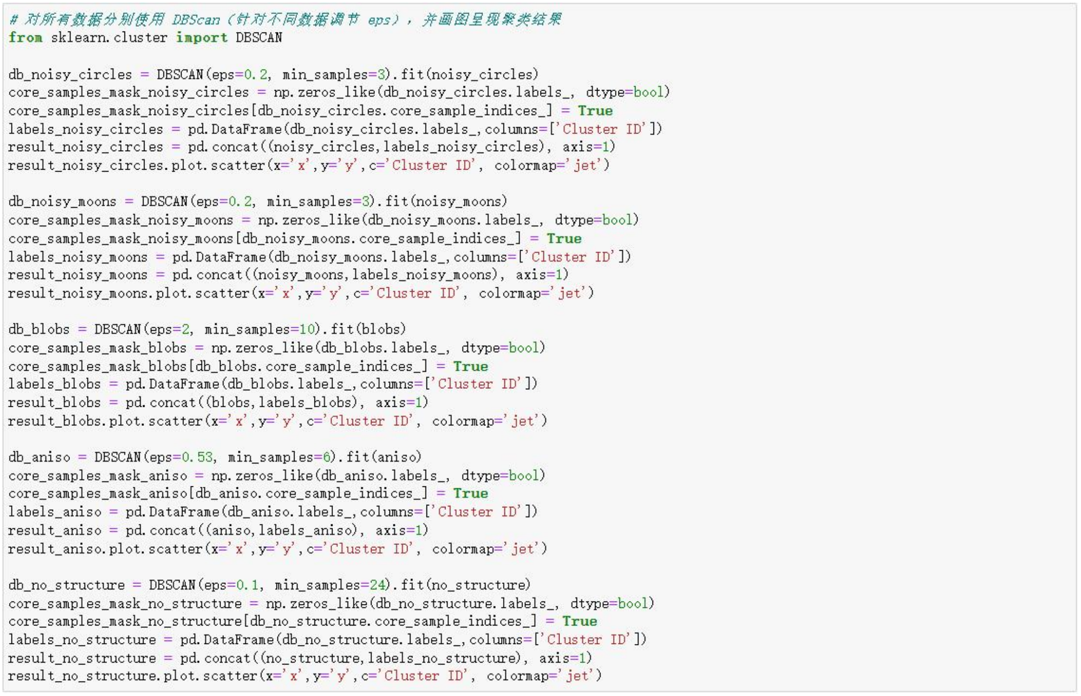
For noisy_moon, we first use DBSCAN function and set parameter eps=0.2 and min_sample=3 to cluster the noisy moon datasets and assign it to db_noisy_moon. Then we extract the predicted label for each point and combine with the original noisy moon datasets. Finally, we use plot.scatter() function to show the predicted result for each point. Different class is recorded by different color.

For blobs, we first use DBSCAN function and set parameter eps=2 and min_sample=10 to cluster the blobs datasets and assign it to db_blobs. Then we extract the predicted label for each point and combine with the original blobs datasets. Finally, we use plot.scatter() function to show the predicted result for each point. Different class is recorded by different color.

For aniso, we first use DBSCAN function and set parameter eps=0.53 and min_sample=6 to cluster the aniso datasets and assign it to db_aniso. Then we extract the predicted label for each point and combine with the original aniso datasets. Finally, we use plot.scatter() function to show the predicted result for each point. Different class is recorded by different color.

For no_structure, we first use DBSCAN function and set parameter eps=0.1 and min_sample=24 to cluster the no structure datasets and assign it to db_no_structure. Then we extract the predicted label for each point and combine with the original no_structure datasets. Finally, we use plot.scatter() function to show the predicted result for each point. Different class is recorded by different color.

All the codes are shown in the following figure.



```python
# 对所有数据分别使用 DBScan（针对不同数据调节 eps），并画图呈现聚类结果
from sklearn.cluster import DBSCAN

db_noisy_circles = DBSCAN(eps=0.2, min_samples=3).fit(noisy_circles)
core_samples_mask_noisy_circles = np.zeros_like(db_noisy_circles.labels_, dtype=bool)
core_samples_mask_noisy_circles[db_noisy_circles.core_sample_indices_] = True
labels_noisy_circles = pd.DataFrame(db_noisy_circles.labels_,columns=['Cluster ID'])
result_noisy_circles = pd.concat((noisy_circles,labels_noisy_circles), axis=1)
result_noisy_circles.plot.scatter(x='x',y='y',c='Cluster ID', colormap='jet')

db_noisy_moons = DBSCAN(eps=0.2, min_samples=3).fit(noisy_moons)
core_samples_mask_noisy_moons = np.zeros_like(db_noisy_moons.labels_, dtype=bool)
core_samples_mask_noisy_moons[db_noisy_moons.core_sample_indices_] = True
labels_noisy_moons = pd.DataFrame(db_noisy_moons.labels_,columns=['Cluster ID'])
result_noisy_moons = pd.concat((noisy_moons,labels_noisy_moons), axis=1)
result_noisy_moons.plot.scatter(x='x',y='y',c='Cluster ID', colormap='jet')

db_blobs = DBSCAN(eps=2, min_samples=10).fit(blobs)
core_samples_mask_blobs = np.zeros_like(db_blobs.labels_, dtype=bool)
core_samples_mask_blobs[db_blobs.core_sample_indices_] = True
labels_blobs = pd.DataFrame(db_blobs.labels_,columns=['Cluster ID'])
result_blobs = pd.concat((blobs,labels_blobs), axis=1)
result_blobs.plot.scatter(x='x',y='y',c='Cluster ID', colormap='jet')

db_aniso = DBSCAN(eps=0.53, min_samples=6).fit(aniso)
core_samples_mask_aniso = np.zeros_like(db_aniso.labels_, dtype=bool)
core_samples_mask_aniso[db_aniso.core_sample_indices_] = True
labels_aniso = pd.DataFrame(db_aniso.labels_,columns=['Cluster ID'])
result_aniso = pd.concat((aniso,labels_aniso), axis=1)
result_aniso.plot.scatter(x='x',y='y',c='Cluster ID', colormap='jet')

db_no_structure = DBSCAN(eps=0.1, min_samples=24).fit(no_structure)
core_samples_mask_no_structure = np.zeros_like(db_no_structure.labels_, dtype=bool)
core_samples_mask_no_structure[db_no_structure.core_sample_indices_] = True
labels_no_structure = pd.DataFrame(db_no_structure.labels_,columns=['Cluster ID'])
result_no_structure = pd.concat((no_structure,labels_no_structure), axis=1)
result_no_structure.plot.scatter(x='x',y='y',c='Cluster ID', colormap='jet')
```

Experimental Results and Analysis:

A. Load the five datasets: noisy_circles.txt, noisy_moons.txt, blobs.txt, aniso.txt and no_structure.txt

All the datasets are successfully loaded and assign to the corresponding variable.

B. Plot scatter diagram to visualize data set.

The following five figures show the distribution of five datasets. From the figure 1, we see that noisy circles data presents circular distribution and two cluster data form the two different circle which indicates that one circle may represents one class. From the figure 2, we see that noisy moon data presents curves distribution and two cluster data form the two different curves whose shape like a moon. That may indicates that the one curves may represents one class. From the figure 3, we see that blobs data presents group distribution and three cluster data form the three different group which indicates that one group may represents one class. From the figure 4, we see that aniso data presents linear distribution and three cluster data form three different linear which indicates that one linear may represents one class. From the figure 5, we see that no structure data present random distribution which indicates that the data is not related.

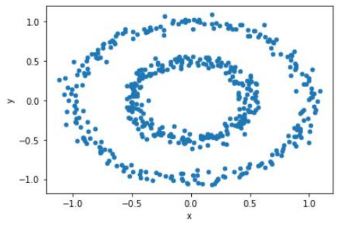


Figure 1: The distribution of noisy circle

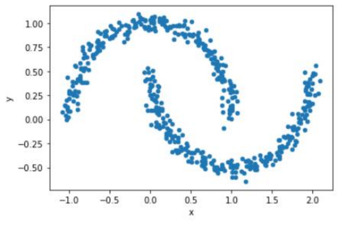


Figure 2: The distribution of noisy moon

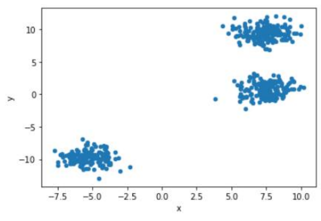


Figure 3: The distribution of blobs

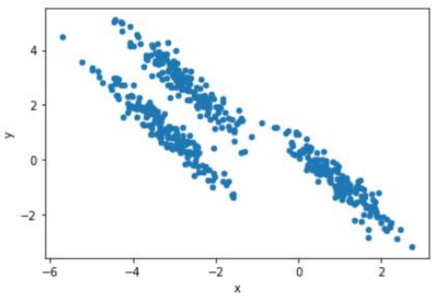


Figure 4: The distribution of aniso

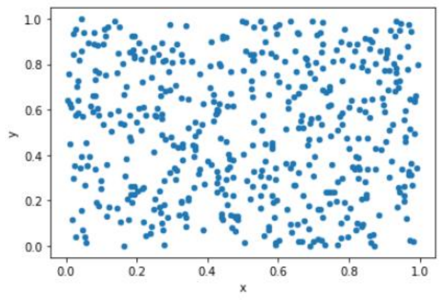


Figure 5: The distribution of no structure

C. Use k-means clustering for all data, and plot a figure to present the clustering results.

The clustering result using k-means clustering in shown in the following five figure. From the figure 6, we observe that there are two different color presented in the result indicating that k-means clustering method clusters two class for the noisy circle data. However, we see that when y>0, the data belongs to blue class; when y<0, the data belongs to red color. This kind of clustering may not subject to what we analyze before. What's more, no matter what random_state we set, the result also show the same clustering outcome.

From the figure 7, we observe that there are two different color presented in the result indicating that k-means clustering method clusters two class for the noisy moon data. However, we see that a part of curve is clustered to blue class while the rest past of curve is clustered to red class. This kind of clustering may not subject to what we analyze before. What’s more, no matter what random_state we set, the result also show the same clustering outcome.

From the figure 8, we observe that there are three different color presented in the result indicating that k-means clustering method clusters three class for the blobs data. One group is clustered to one class. This kind of clustering is subject to what we analyze before.

From the figure 9, we observe that there are three different color presented in the result indicating that k-means clustering method clusters three class for the aniso data. However, we see that a part of one linear is clustered to green color while the rest part of the linear is clustered to brown color. This kind of clustering is subject to what we analyze before. What’s more, no matter what random_state we set, the result also show the same clustering outcome.

From the figure 10, we observe that there are three different color presented in the result indicating that k-means clustering method clusters three classes for the no structure data. The three concentrated parts are divided into three class. Intuitively, this kind of clustering may be reasonable.

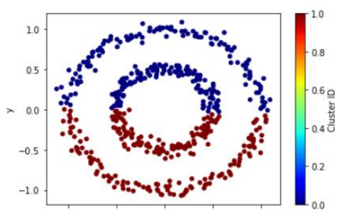


Figure 6: K-means clustering for noisy circle

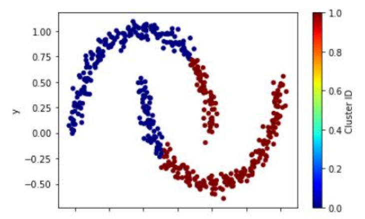


Figure7: K-means clustering for noisy moon

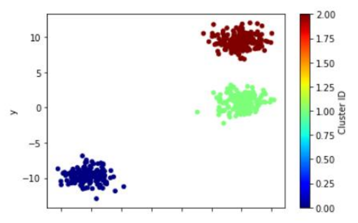


Figure 8: K-means clustering for blobs

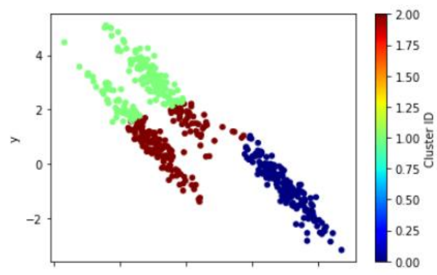


Figure 9: K-means clustering for aniso

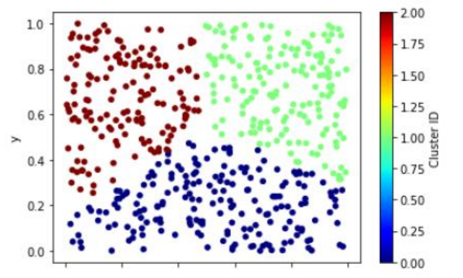


Figure 10: K-means clustering for no structure

D. Use DBSCAN (adjust EPS for different data) for all data, and plot a figure to present the clustering results.

All the clustering result for corresponding data are shown in the following five figure. From the figure 11, we see that there are two different color presented in the result indicating that DBCAN clustering method clusters two class for the noisy circle data. And one circle is clustered to one class which is subject to the former analysis.

From the figure 12, we see that there are two different color presented in the result indicating that DBCAN clustering method clusters two class for the noisy moon data. And one curve is clustered to one class which is subject to the former analysis.

From the figure 13, we observe that there are three different color presented in the result indicating that DBCAN clustering method clusters three class for the blobs data. And one group is clustered to one class which is subject to the before analysis.

From the figure 14, we see that there are three different color presented in the result indicating that DBCAN clustering method clusters three class for the aniso data. And one linear is clustered to one class which is subject to the before analysis.

From the figure 15, we observe that there are three different color presented in the result indicating that DBCAN clustering method clusters three class for the aniso data. The three concentrated parts are divided into three class. Intuitively, this kind of clustering may be reasonable.

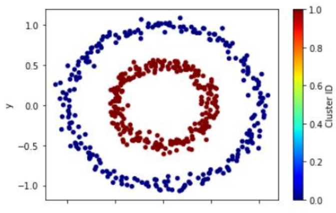


Figure 11: DBSCAN for noisy circle

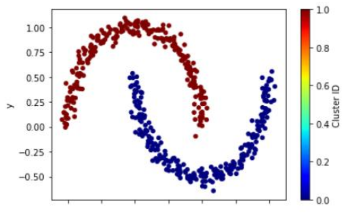


Figure12: DBSCAN for noisy moon

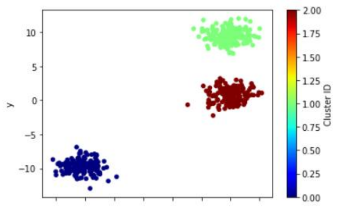


Figure 13: DBSCAN for blobs

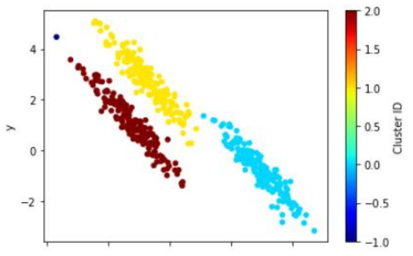


Figure 14: DBSCAN for noisy circle

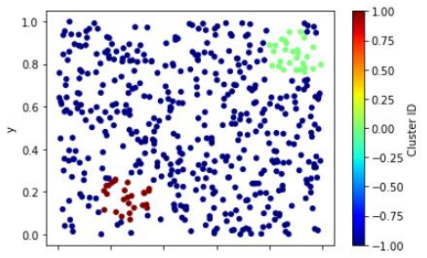


Figure15: DBSCAN for noisy moon

Now we Compare the performance of two cluster method for the specific data.

From the figure 16, we see that the DBSCAN cluster method is better than k-means cluster method since one circle more likely represents one class.

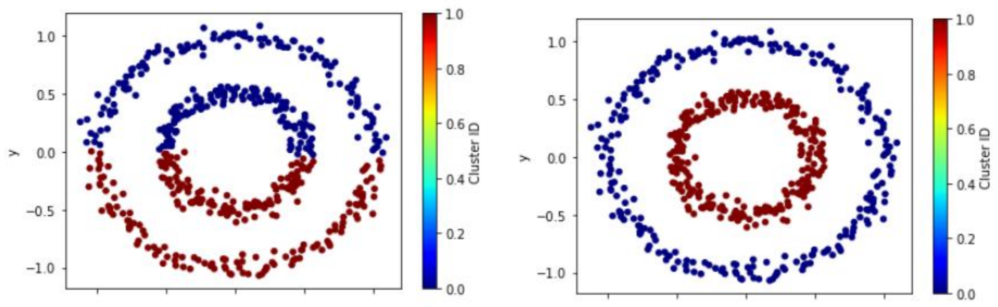


Figure 16: Comparison of two cluster method for noisy circle

From the figure 17, we see that the DBSCAN cluster method is better than k-means cluster method since one curve more likely represents one class. Combined the analysis of figure 16, we can draw a conclusion that if the distribution of data presents non-linear separation characteristic, DBSCAN cluster method may be suitable to correctly classify.

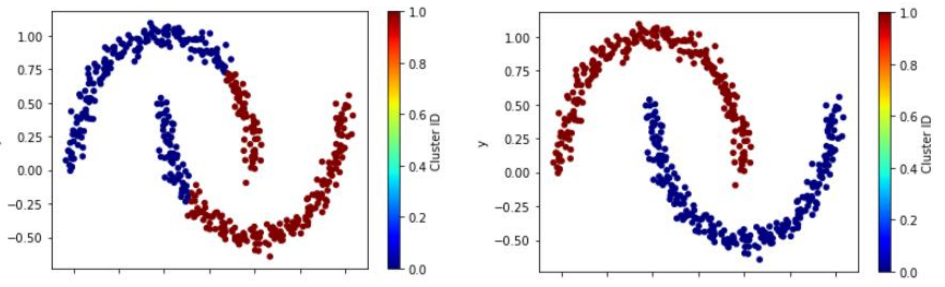


Figure 17: Comparison of two cluster method for noisy moon

From the figure 18, we see that the performance of k-means method and DBSCAN method is same. So we can draw a conclusion that if the distribution of data presents linear separation characteristic, k-means method and DBSCAN cluster method are all suitable to correctly classify.

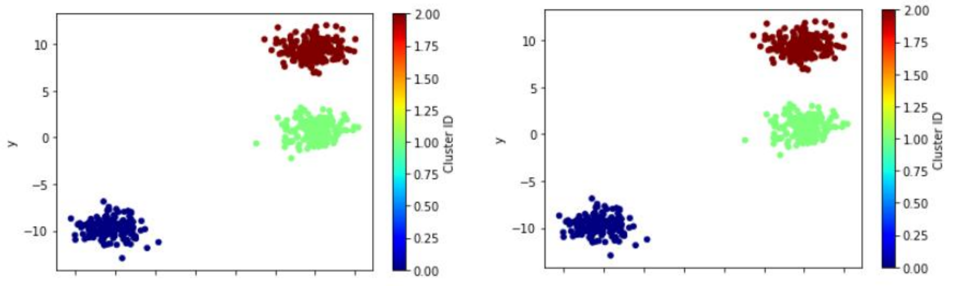


Figure 18: Comparison of two cluster method for blobs

From the figure 19, we see that the DBSCAN cluster method is better than k-means cluster method since one linear more likely represents one class. Combined the analysis of figure 18, we can revise the conclusion that if the distribution of data presents linear separation characteristic and two potential class data not distribute closely, k-means method and DBSCAN cluster method are all suitable to correctly classify. Otherwise, DBSCAN cluster method may be suitable to correctly classify.

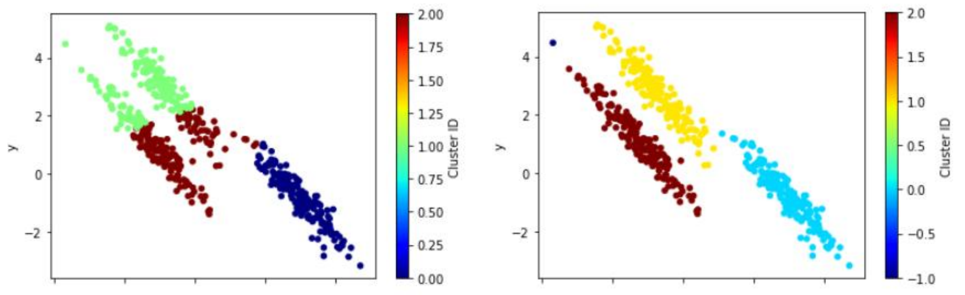


Figure 19: Comparison of two cluster method for aniso

From the figure 20, we see that k-mean cluster method and DBSCAN cluster method accidentally cluster the data into three class. In the experiment, for k-means method, we can set the n_clusters parameter but for DBSCAN method we only set the eps and min_sample to achieve three-class clustering. Since no structure data distributes randomly, there are no concept of class for this data. Consequently, if the method clusters three central class for the data, we view that the method perform well. In conclusion, we draw a conclusion that we can adjust the parameter of k-means clustering and DBSCAN method to achieve 'correctly' classify for the random distributed data.

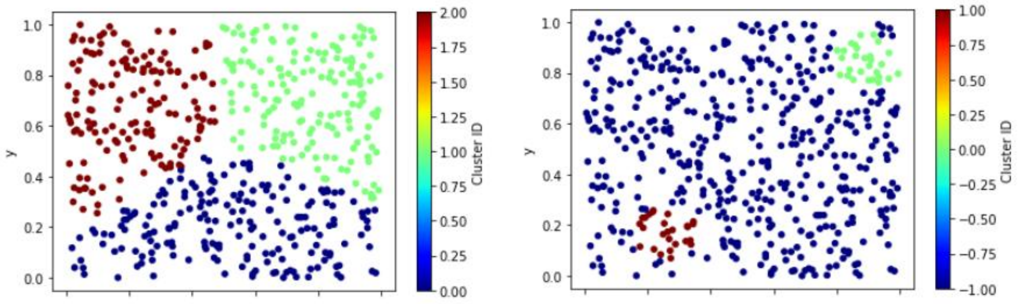


Figure 20: Comparison of two cluster method for no structure

Experimental conclusion:

(1) If the distribution of data presents non-linear separation characteristic, DBSCAN cluster method may be suitable to correctly classify.

(2) If the distribution of data presents linear separation characteristic and two potential class data not distribute closely, k-means method and DBSCAN cluster method are all suitable to correctly classify. Otherwise, DBSCAN cluster method may be suitable to correctly classify.

(3) We can adjust the parameter of k-means clustering and DBSCAN method to achieve 'correctly' classify for the random distributed data.

指导教师批阅意见：   成绩评定：   指导教师签字： 年　　月　　日
备注：

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。

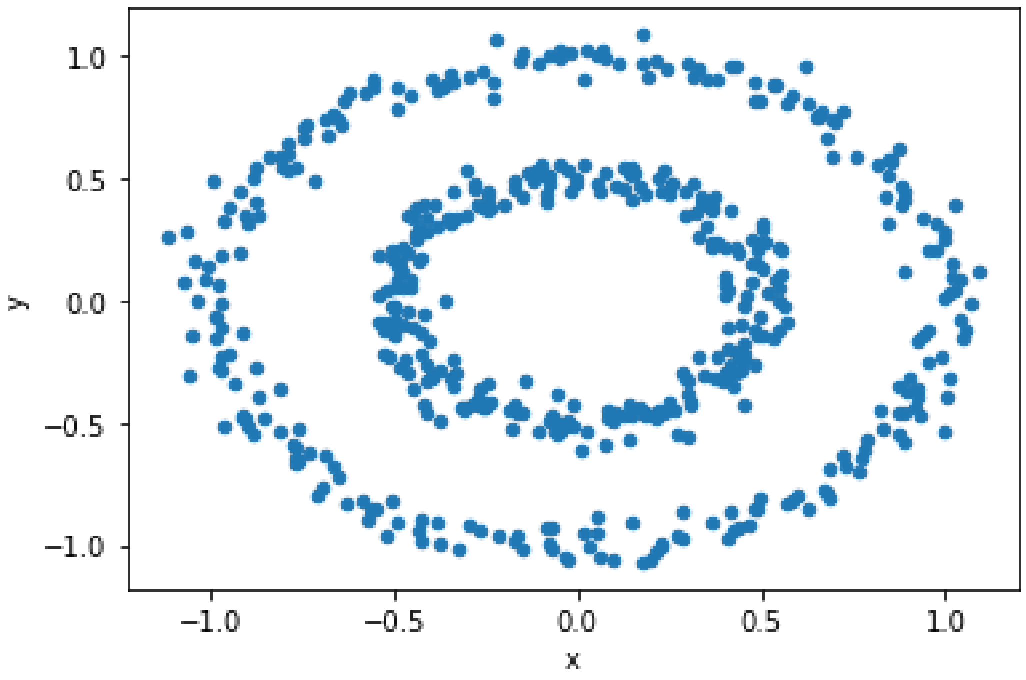
Clustering experiment

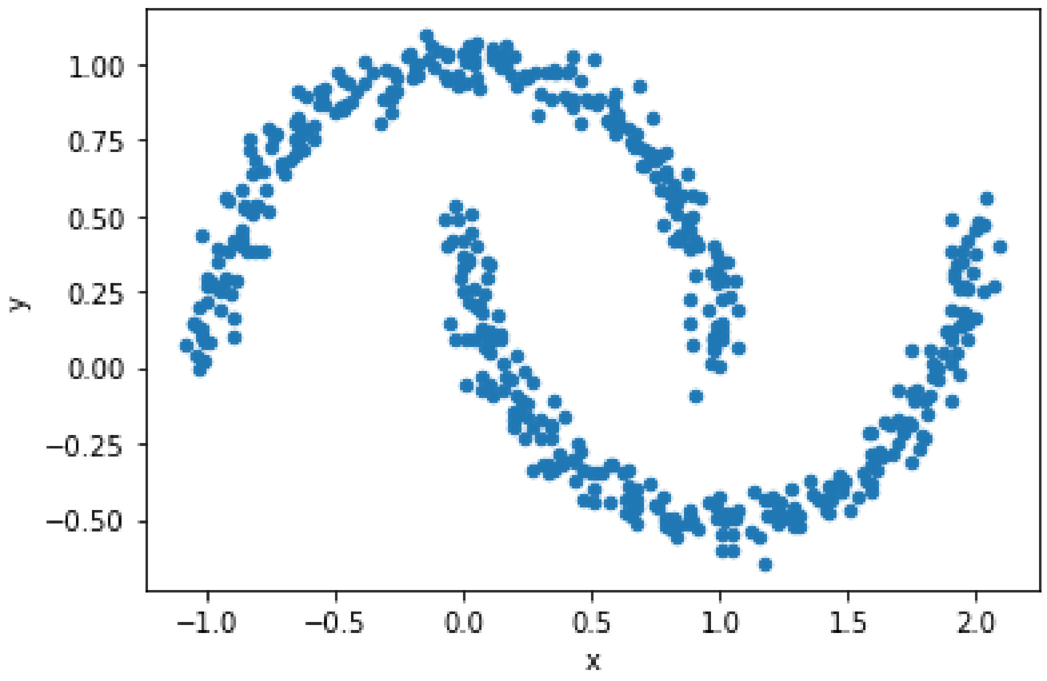
May 27, 2022

[2]:
```
#   5
import pandas as pd
noisy_circles = pd.read_csv("noisy_circles.txt", delimiter=' ', names=['x','y'])
noisy_moons = pd.read_csv("noisy_moons.txt", delimiter=' ', names=['x','y'])
blobs = pd.read_csv("blobs.txt", delimiter=' ', names=['x','y'])
aniso = pd.read_csv("aniso.txt", delimiter=' ', names=['x','y'])
no_structure = pd.read_csv("no_structure.txt", delimiter=' ', names=['x','y'])
```

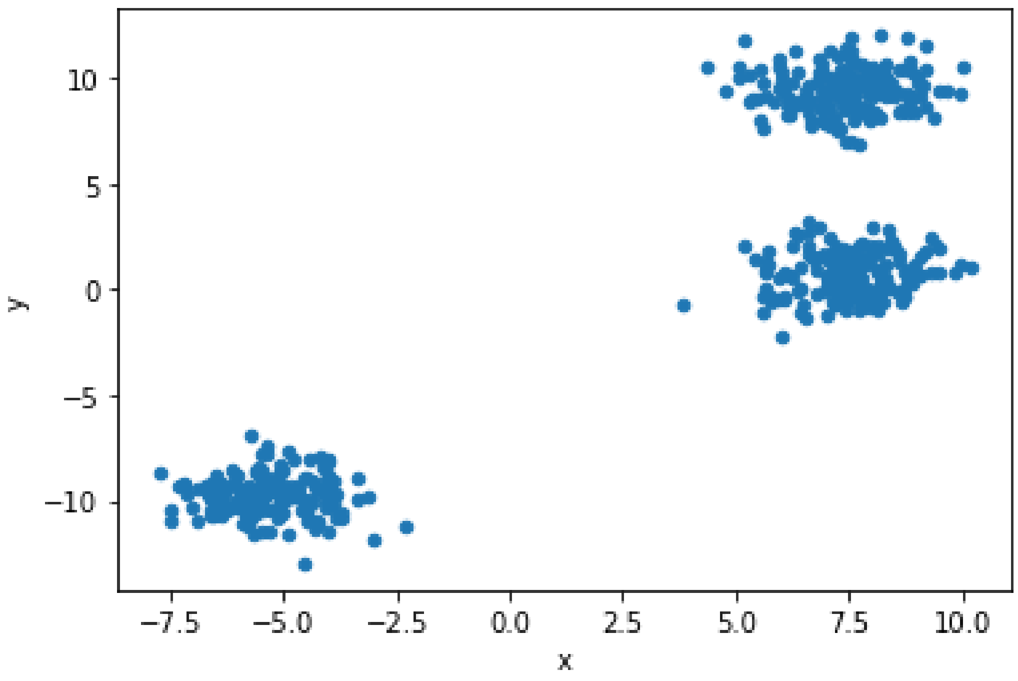
[12]:
```
#
noisy_circles.plot.scatter(x='x',y='y')
noisy_moons.plot.scatter(x='x',y='y')
blobs.plot.scatter(x='x',y='y')
aniso.plot.scatter(x='x',y='y')
no_structure.plot.scatter(x='x',y='y')
```

[12]:
```
<AxesSubplot:xlabel='x', ylabel='y'>
```

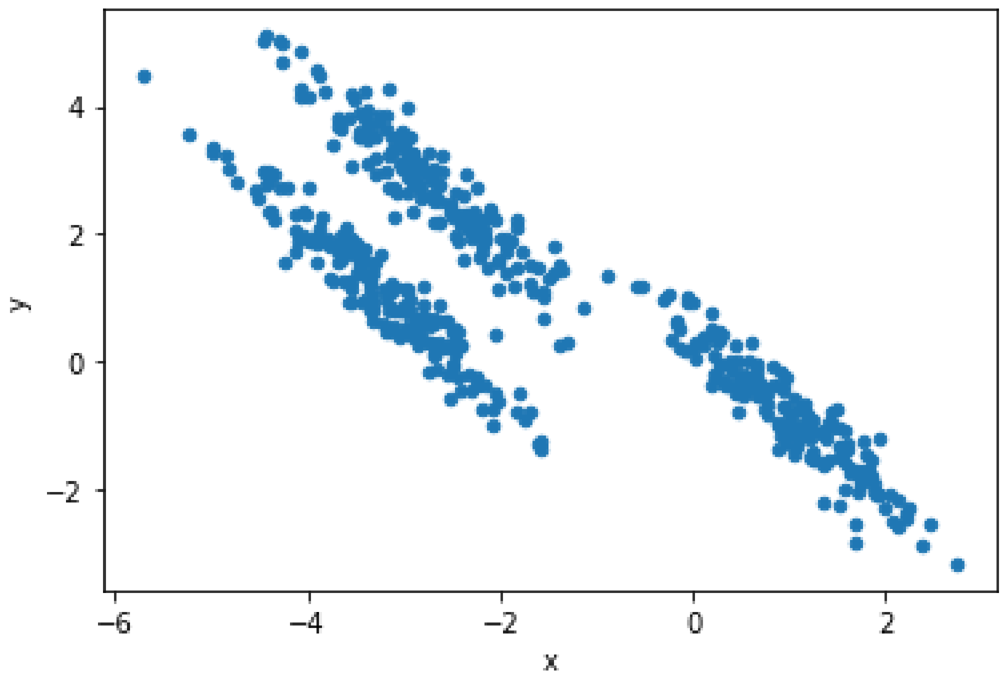


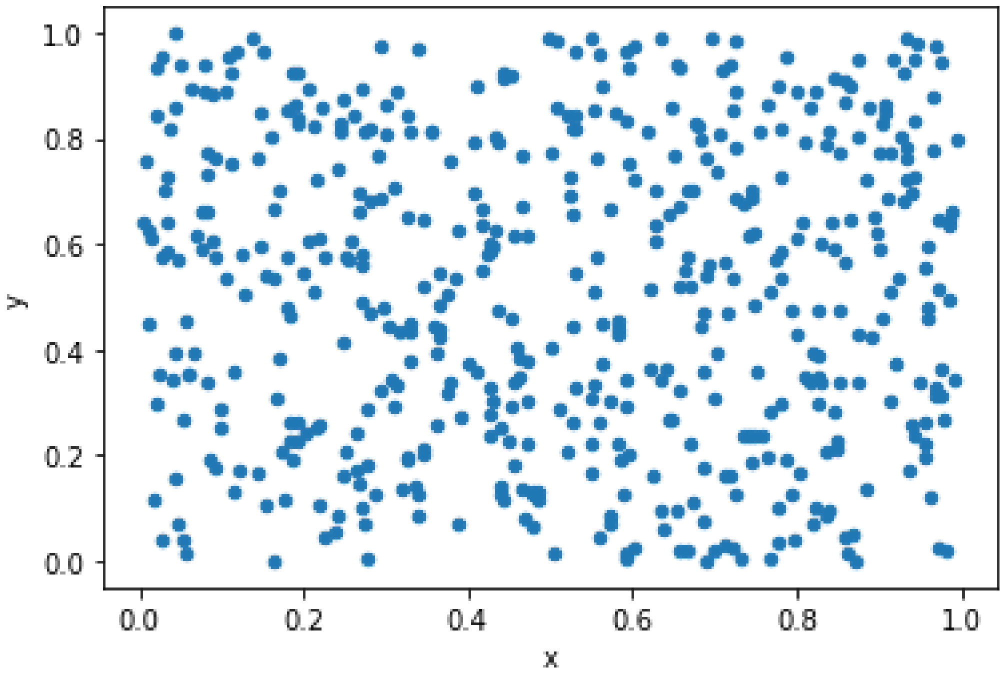


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
y
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
x



10
5
0
−5
−10
y
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
x

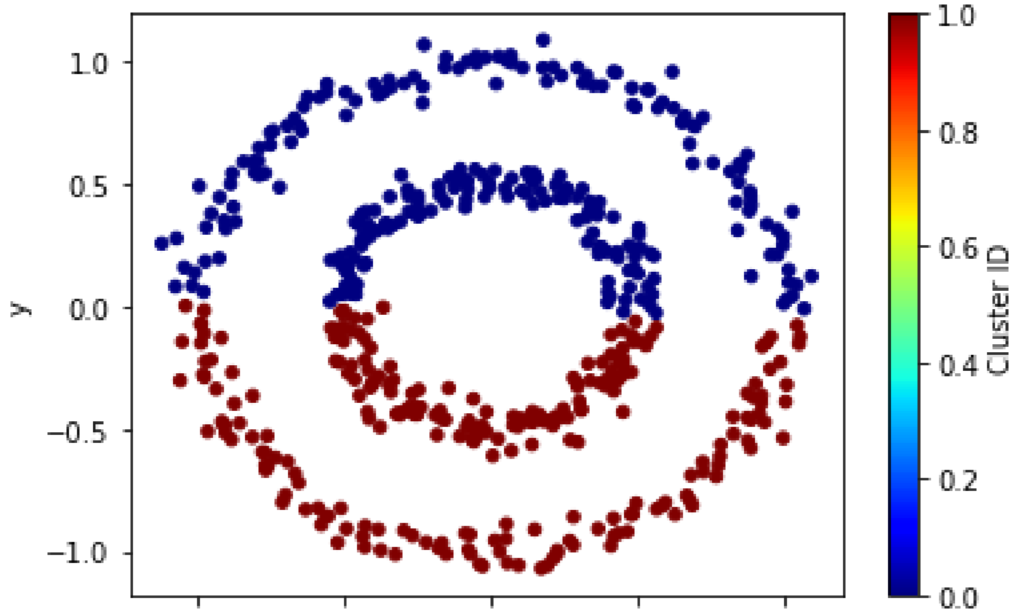


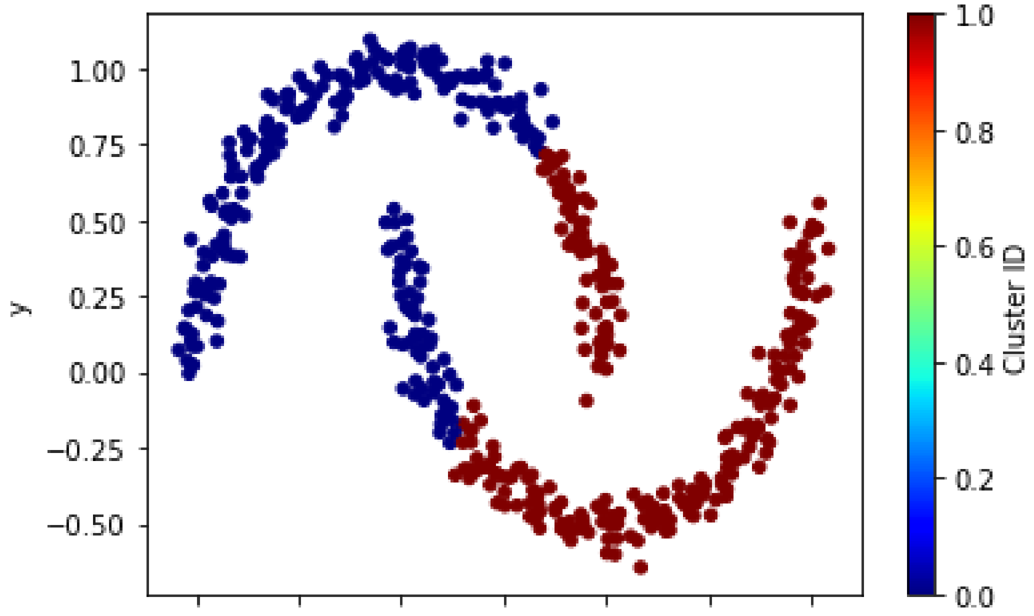


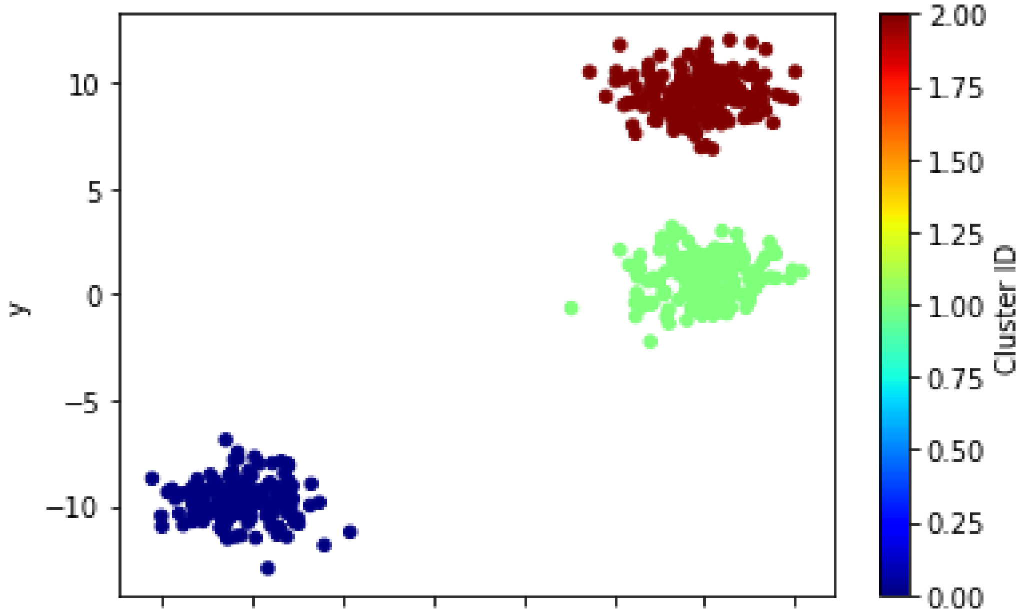
[68]:
```
#       K-means Clustering,
from sklearn import cluster
```

```
k_means_noisy_circles = cluster.KMeans(n_clusters=2, max_iter=50,␣
 ↪random_state=100).fit(noisy_circles)
labels_noisy_circles = pd.DataFrame(k_means_noisy_circles.
 ↪labels_,columns=['Cluster ID'])
result_noisy_circles = pd.concat((noisy_circles,labels_noisy_circles), axis=1)
result_noisy_circles.plot.scatter(x='x',y='y',c='Cluster ID',colormap='jet')

k_means_noisy_moons = cluster.KMeans(n_clusters=2, max_iter=50, random_state=1).
 ↪fit(noisy_moons)
labels_noisy_moons = pd.DataFrame(k_means_noisy_moons.labels_,columns=['Cluster␣
 ↪ID'])
result_noisy_moons = pd.concat((noisy_moons,labels_noisy_moons), axis=1)
result_noisy_moons.plot.scatter(x='x',y='y',c='Cluster ID',colormap='jet')

k_means_blobs = cluster.KMeans(n_clusters=3, max_iter=50, random_state=1).
 ↪fit(blobs)
labels_blobs = pd.DataFrame(k_means_blobs.labels_,columns=['Cluster ID'])
result_blobs = pd.concat((blobs,labels_blobs), axis=1)
result_blobs.plot.scatter(x='x',y='y',c='Cluster ID',colormap='jet')

k_means_aniso = cluster.KMeans(n_clusters=3, max_iter=50, random_state=100).
 ↪fit(aniso)
labels_aniso = pd.DataFrame(k_means_aniso.labels_,columns=['Cluster ID'])
result_aniso = pd.concat((aniso,labels_aniso), axis=1)
result_aniso.plot.scatter(x='x',y='y',c='Cluster ID',colormap='jet')

k_means_no_structure = cluster.KMeans(n_clusters=3, max_iter=100,␣
 ↪random_state=1).fit(no_structure)
labels_no_structure = pd.DataFrame(k_means_no_structure.
 ↪labels_,columns=['Cluster ID'])
result_no_structure = pd.concat((no_structure,labels_no_structure), axis=1)
result_no_structure.plot.scatter(x='x',y='y',c='Cluster ID',colormap='jet')
```

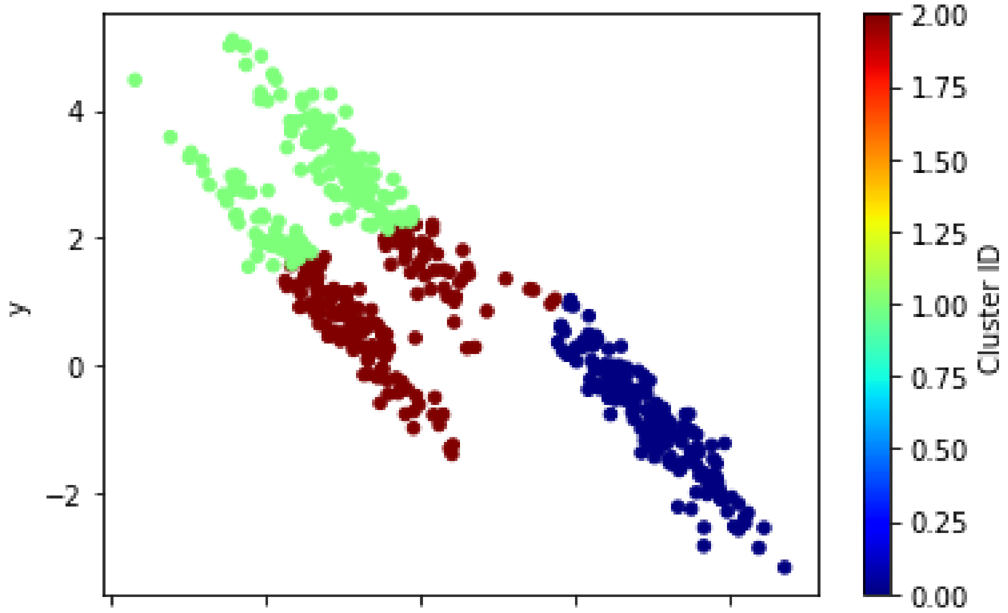
```
[68]: <AxesSubplot:xlabel='x', ylabel='y'>
```


1.0
0.5
0.0
−0.5
−1.0
y
1.0
0.8
0.6
0.4
0.2
0.0
Cluster ID

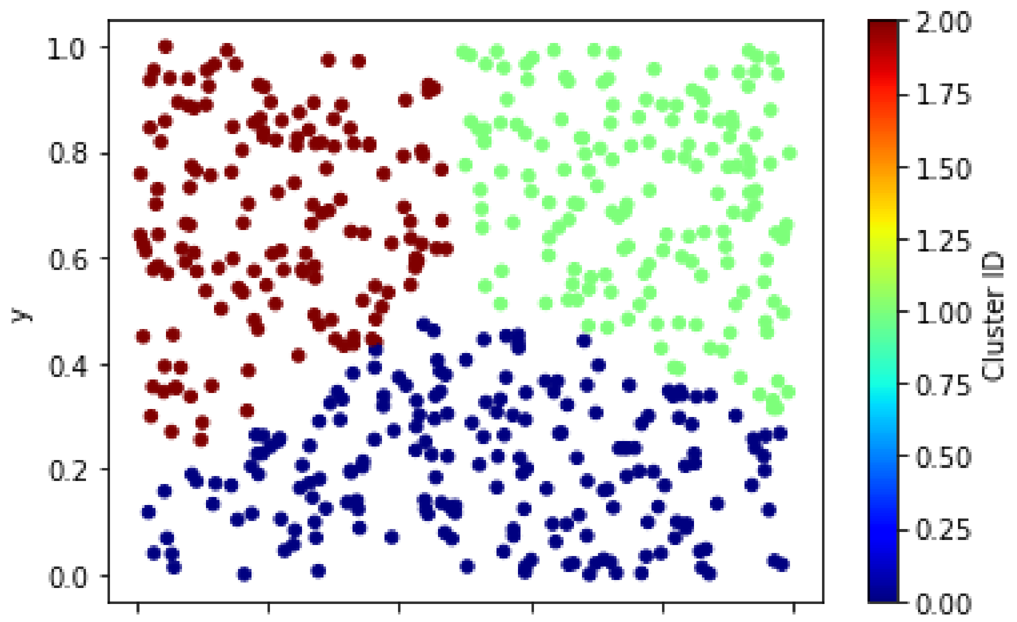

1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
y
1.0
0.8
0.6
0.4
0.2
0.0
Cluster ID



10
5
0
−5
−10
y
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Cluster ID



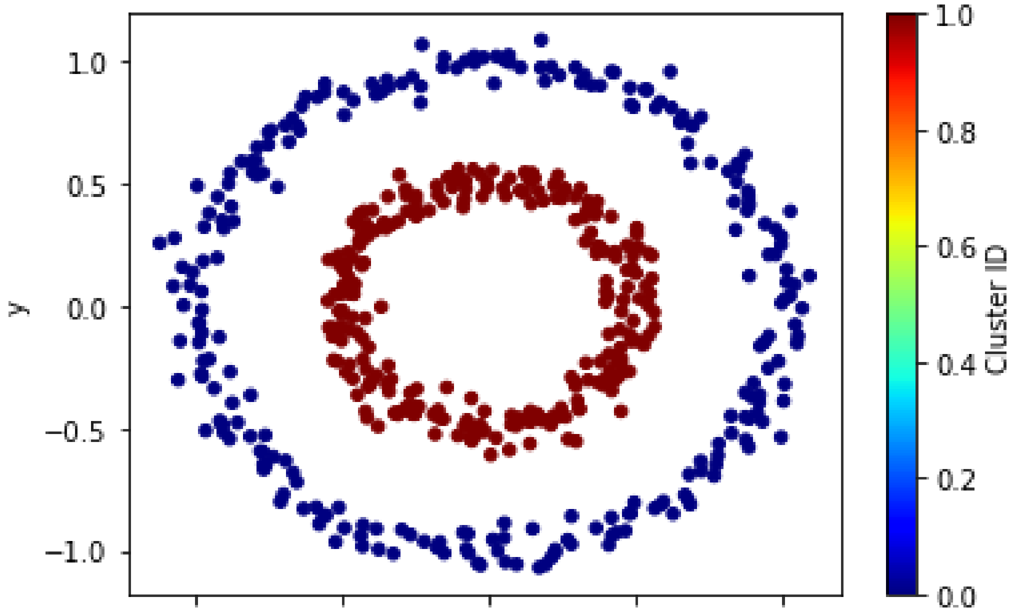
4
2
0
−2
y
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Cluster ID

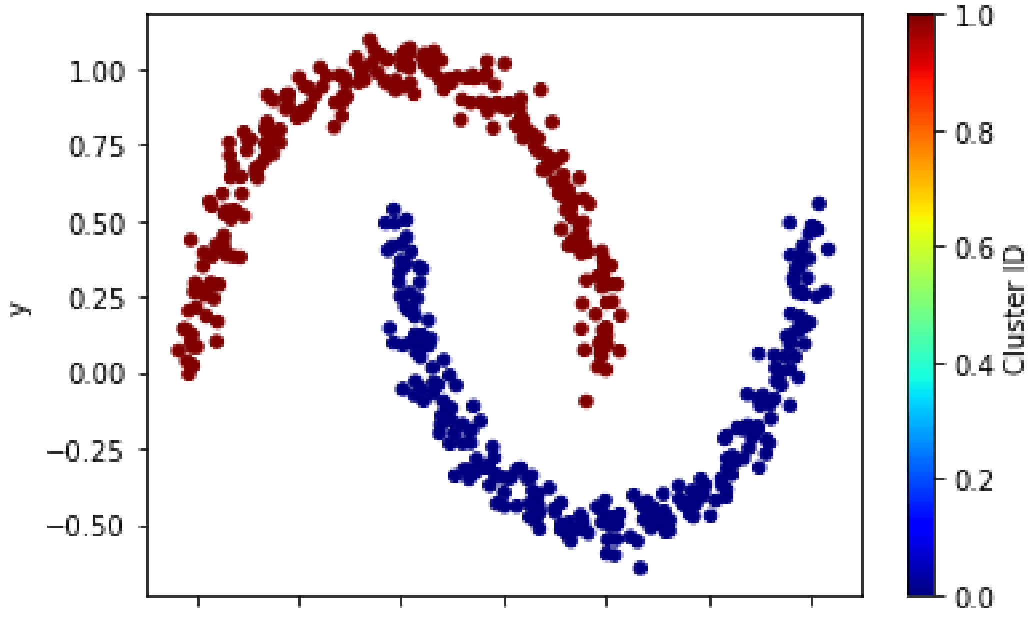


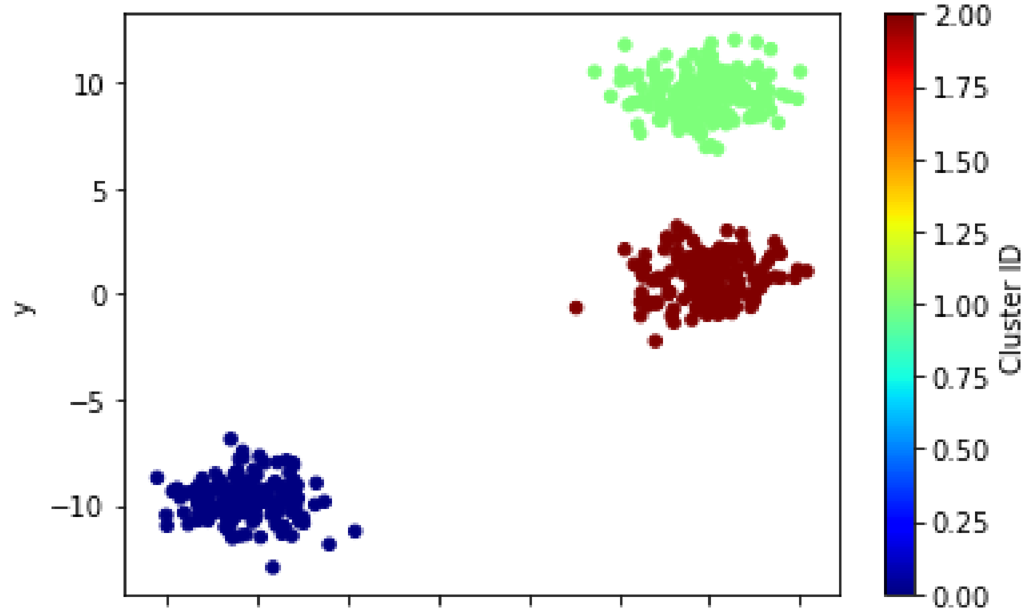
```
[4]: #       DBScan       eps
from sklearn.cluster import DBSCAN
import numpy as np

db_noisy_circles = DBSCAN(eps=0.2, min_samples=3).fit(noisy_circles)
core_samples_mask_noisy_circles = np.zeros_like(db_noisy_circles.labels_,␣
 ↪dtype=bool)
core_samples_mask_noisy_circles[db_noisy_circles.core_sample_indices_] = True
labels_noisy_circles = pd.DataFrame(db_noisy_circles.labels_,columns=['Cluster␣
 ↪ID'])
result_noisy_circles = pd.concat((noisy_circles,labels_noisy_circles), axis=1)
result_noisy_circles.plot.scatter(x='x',y='y',c='Cluster ID', colormap='jet')

db_noisy_moons = DBSCAN(eps=0.2, min_samples=3).fit(noisy_moons)
core_samples_mask_noisy_moons = np.zeros_like(db_noisy_moons.labels_,␣
 ↪dtype=bool)
core_samples_mask_noisy_moons[db_noisy_moons.core_sample_indices_] = True
labels_noisy_moons = pd.DataFrame(db_noisy_moons.labels_,columns=['Cluster ID'])
result_noisy_moons = pd.concat((noisy_moons,labels_noisy_moons), axis=1)
result_noisy_moons.plot.scatter(x='x',y='y',c='Cluster ID', colormap='jet')

db_blobs = DBSCAN(eps=2, min_samples=10).fit(blobs)
core_samples_mask_blobs = np.zeros_like(db_blobs.labels_, dtype=bool)
core_samples_mask_blobs[db_blobs.core_sample_indices_] = True
labels_blobs = pd.DataFrame(db_blobs.labels_,columns=['Cluster ID'])
result_blobs = pd.concat((blobs,labels_blobs), axis=1)
```

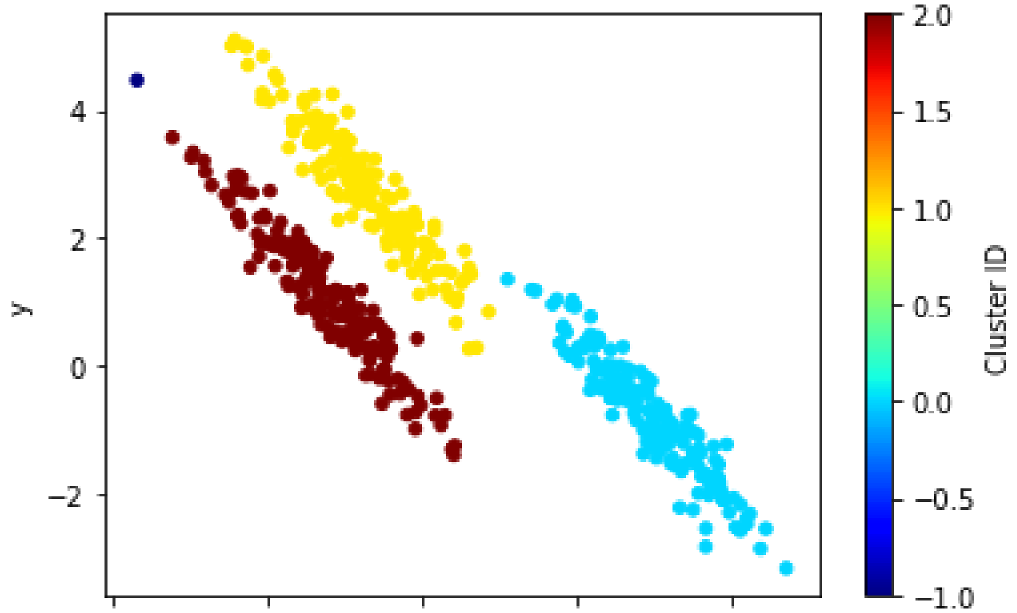
```
result_blobs.plot.scatter(x='x',y='y',c='Cluster ID', colormap='jet')

db_aniso = DBSCAN(eps=0.53, min_samples=6).fit(aniso)
core_samples_mask_aniso = np.zeros_like(db_aniso.labels_, dtype=bool)
core_samples_mask_aniso[db_aniso.core_sample_indices_] = True
labels_aniso = pd.DataFrame(db_aniso.labels_,columns=['Cluster ID'])
result_aniso = pd.concat((aniso,labels_aniso), axis=1)
result_aniso.plot.scatter(x='x',y='y',c='Cluster ID', colormap='jet')

db_no_structure = DBSCAN(eps=0.1, min_samples=24).fit(no_structure)
core_samples_mask_no_structure = np.zeros_like(db_no_structure.labels_,␣
 ↪dtype=bool)
core_samples_mask_no_structure[db_no_structure.core_sample_indices_] = True
labels_no_structure = pd.DataFrame(db_no_structure.labels_,columns=['Cluster␣
 ↪ID'])
result_no_structure = pd.concat((no_structure,labels_no_structure), axis=1)
result_no_structure.plot.scatter(x='x',y='y',c='Cluster ID', colormap='jet')
```

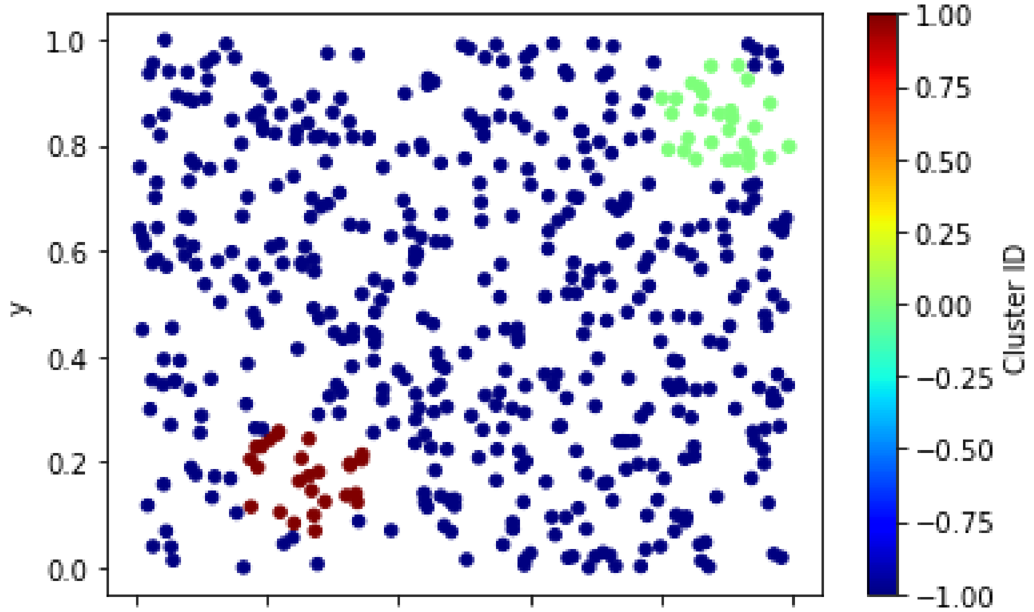
[4]:
```
<AxesSubplot:xlabel='x', ylabel='y'>
```




1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
y
1.0
0.8
0.6
0.4
0.2
0.0
Cluster ID


10
5
0
−5
−10
y
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Cluster ID


y
4
2
0
−2
Cluster ID
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0


y
1.0
0.8
0.6
0.4
0.2
0.0
Cluster ID
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00