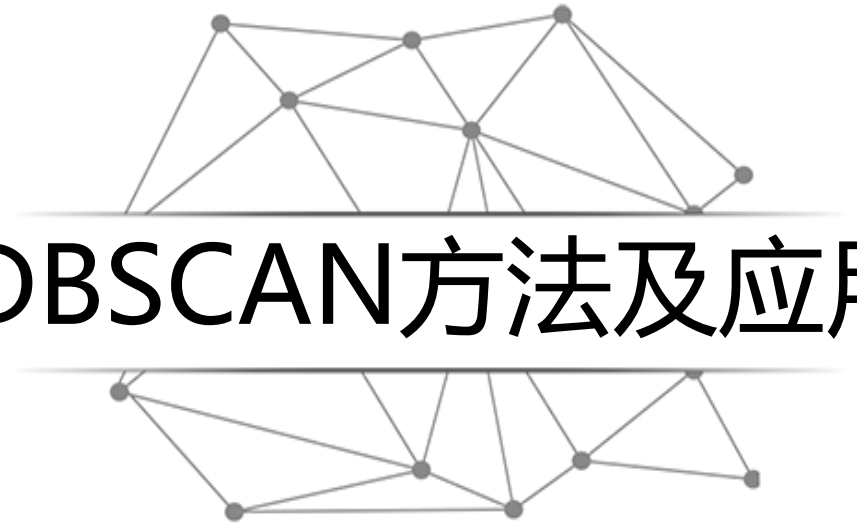
无监督学习-聚类

ML06

礼欣

www.python123.org



DBSCAN方法及应用

DBSCAN密度聚类

DBSCAN算法是一种基于密度的聚类算法：

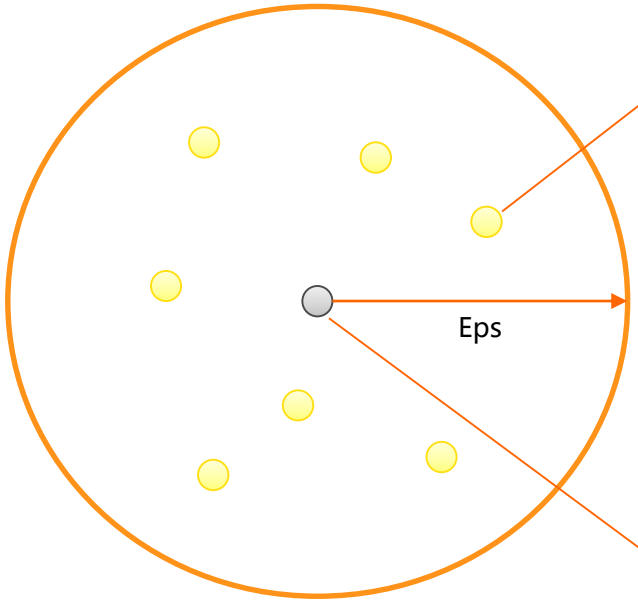
- 聚类的时候不需要预先指定簇的个数
- 最终的簇的个数不定

DBSCAN密度聚类

DBSCAN算法将数据点分为三类：

- 核心点：在半径Eps内含有超过MinPts数目的点
- 边界点：在半径Eps内点的数量小于MinPts，但是落在核心点的邻域内
- 噪音点：既不是核心点也不是边界点的点

DBSCAN密度聚类



边界点：若其邻域内点不超过MinPts个

核心点：邻域内点的个数超过MinPts

DBSCAN密度聚类

DBSCAN算法流程：

1.将所有点标记为核心点、边界点或噪声点；

2.删除噪声点；

3.为距离在Eps之内的所有核心点之间赋予一条边；

4.每组连通的核心点形成一个簇；

5.将每个边界点指派到一个与之关联的核心点的簇中（哪一个核心点的半径范围之内）。

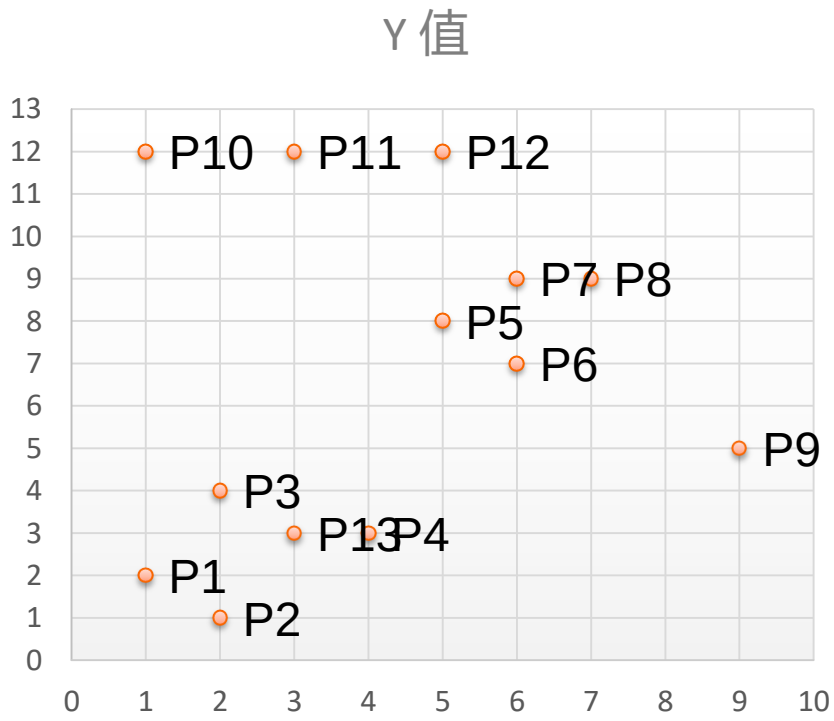
DBSCAN密度聚类

举例：有如下13个样本点，使用DBSCAN进行聚类

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13
X	1	2	2	4	5	6	6	7	9	1	3	5	3
Y	2	1	4	3	8	7	9	9	5	12	12	12	3

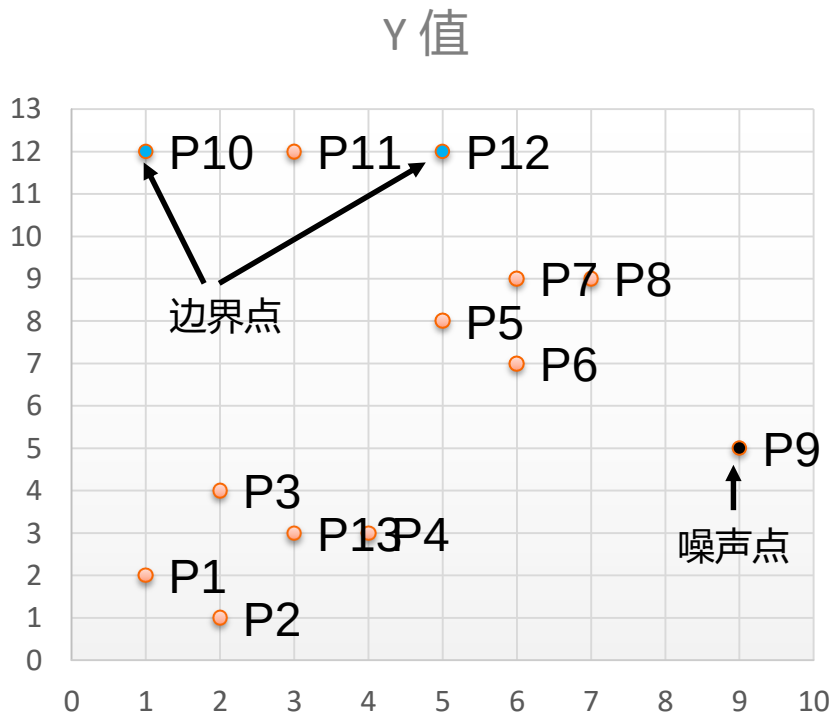
DBSCAN密度聚类

取Eps=3，MinPts=3，依据DBSACN对所有点进行聚类（曼哈顿距离）。



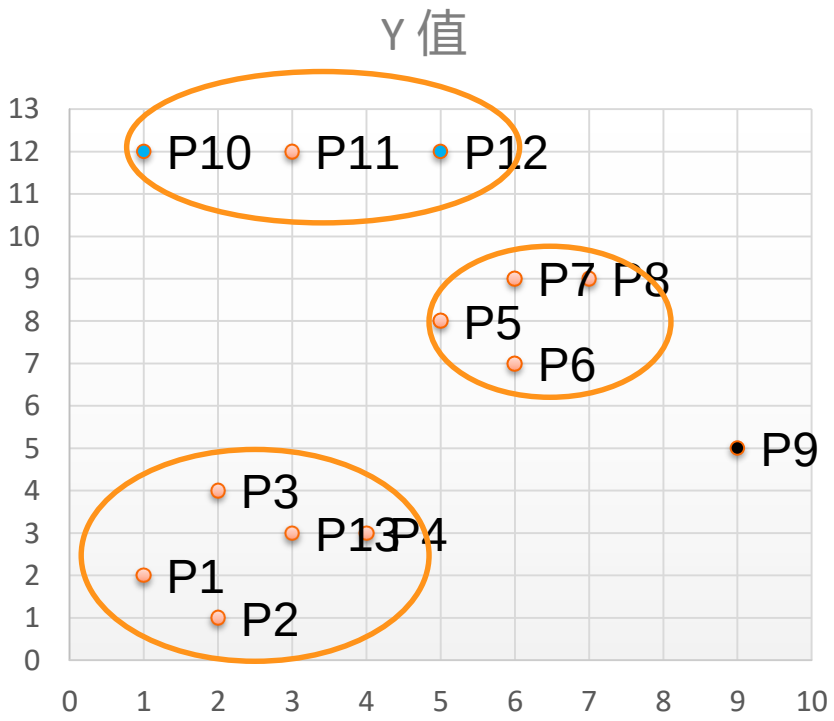
DBSCAN密度聚类

- 对每个点计算其邻域Eps=3内的点的集合。
- 集合内点的个数超过MinPts=3的点为核心点
- 查看剩余点是否在核心点的邻域内，若在，则为边界点，否则为噪声点。



DBSCAN密度聚类

将距离不超过Eps=3的点相互连接，构成一个簇，核心点邻域内的点也会被加入到这个簇中。则右侧形成3个簇。



DBSCAN的应用实例

数据介绍：

现有大学校园网的日志数据，290条大学生的校园网使用情况数据，数据包括用户ID，设备的MAC地址，IP地址，开始上网时间，停止上网时间，上网时长，校园网套餐等。利用已有数据，分析学生上网的模式。

实验目的：

通过DBSCAN聚类，分析学生上网时间和上网时长的模式。

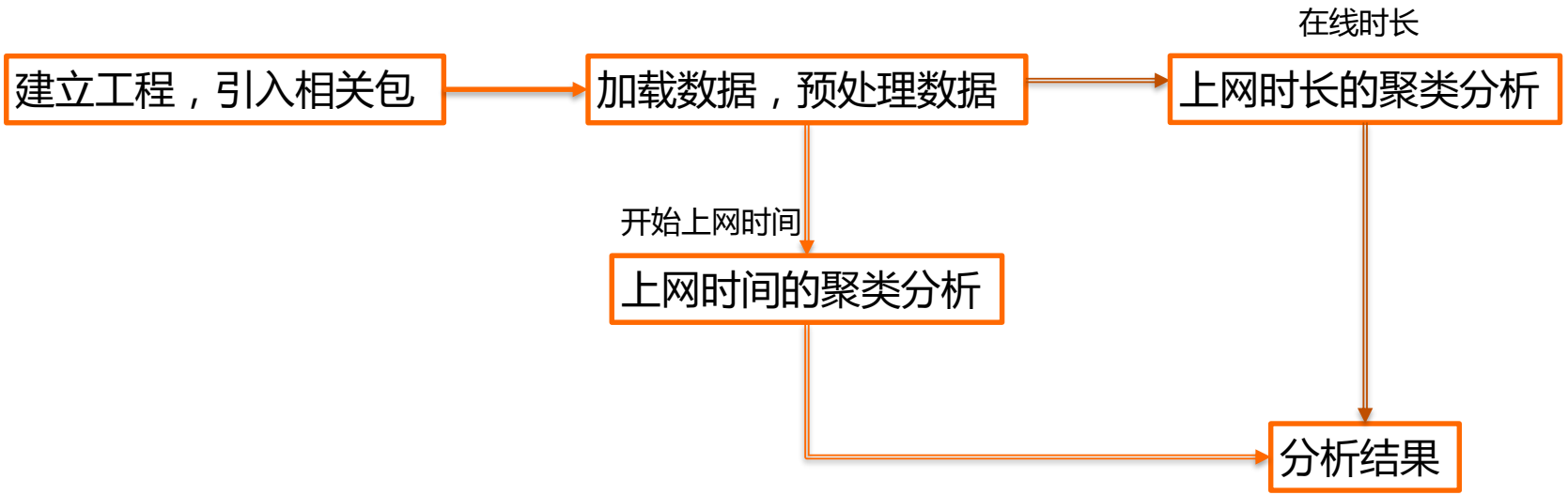
技术路线：`sklearn.cluster.DBSCAN`

数据实例：

学生上网日志（单条数据格式）	
记录编号	2c929293466b97a6014754607e457d68
学生编号	U201215025
MAC地址	A417314EEA7B
IP地址	10.12.49.26
开始上网时间	2014-07-20 22:44:18.540000000
停止上网时间	2014-07-20 23:10:16.540000000
上网时长	1558

实验过程：

- 使用算法： DBSCAN聚类算法
- 实现过程：



1. 建立工程，导入sklearn相关包

```
import numpy as np
from sklearn.cluster import DBSCAN
```

DBSCAN主要参数：

- eps: 两个样本被看作邻居节点的最大距离
- min_samples: 簇的样本数
- metric：距离计算方式

例：sklearn.cluster.DBSCAN(eps=0.5, min_samples=5, metric='euclidean')

详细：http://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html#sklearn.cluster.DBSCAN

2. 读入数据并进行处理

```
import numpy as np
import sklearn.cluster as skc
from sklearn import metrics
import matplotlib.pyplot as plt


mac2id=dict()
onlinetimes=[]
f=open('TestData.txt')
for line in f:
    mac=line.split(',')[2]
    onlinetime=int(line.split(',')[6])
    starttime=int(line.split(',')[4].split(' ')[1].split(':')[0])
    if mac not in mac2id:
        mac2id[mac]=len(onlinetimes)
        onlinetimes.append((starttime,onlinetime))
    else:
        onlinetimes[mac2id[mac]]=[(starttime,onlinetime)]
real_X=np.array(onlinetimes).reshape((-1,2))
```

读取每条数据中的mac地址，
开始上网时间，上网时长

mac2id是一个字典：
key是mac地址
value是对应mac地址的上网时长以及开始上网时间

3-1. **上网时间**聚类，创建DBSCAN算法实例，并进行训练，获得标签：

```
X=real_X[:,0:1]

db=skc.DBSCAN(eps=0.01,min_samples=20).fit(X)
labels = db.labels_
```

调用DBSCAN方法进行训练，labels为每个数据的簇标签

```
print('Labels:')
print(labels)
raito=len(labels[labels[:] == -1]) / len(labels)
print('Noise raito:',format(raito, '.2%'))
```

打印数据被记上的标签，计算标签为-1，即噪声数据的比例。

```
n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)

print('Estimated number of clusters: %d' % n_clusters_)
print("Silhouette Coefficient: %0.3f"% metrics.silhouette_score(X, labels))
```

计算簇的个数并打印，评价聚类效果

```
for i in range(n_clusters_):
    print('Cluster ',i,':')
    print(list(X[labels == i].flatten()))
```

打印各簇标号以及各簇内数据

4. 输出标签，查看结果

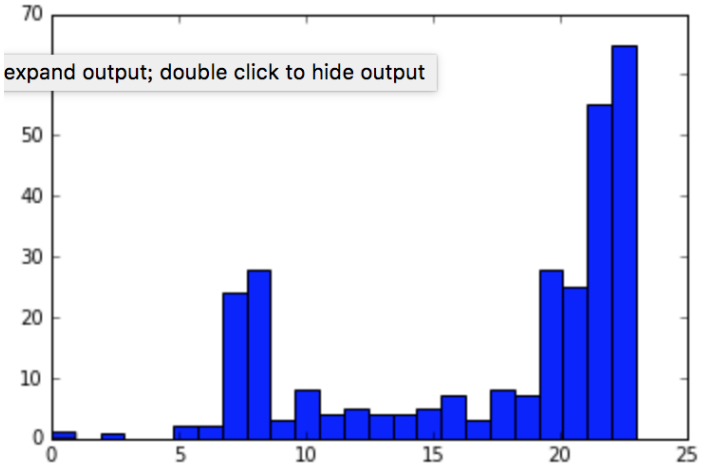
```
Labels:
[ 0 -1  0  1 -1  1  0  1  2 -1  1  0  1  1  3 -1 -1  3 -1  1  1 -1  1  3  4
 -1  1  1  2  0  2  2 -1  0  1  0  0  0  1  3 -1  0  1  1  0  0  2 -1  1  3
  1 -1  3 -1  3  0  1  1  2  3  3 -1 -1 -1  0  1  2  1 -1  3  1  1  2  3  0
  1 -1  2  0  0  3  2  0  1 -1  1  3 -1  4  2 -1 -1  0 -1  3 -1  0  2  1 -1
 -1  2  1  1  2  0  2  1  1  3  3  0  1  2  0  1  0 -1  1  1  3 -1  2  1  3
  1  1  1  2 -1  5 -1  1  3 -1  0  1  0  0  1 -1 -1 -1  2  2  0  1  1  3  0
  0  0  1  4  4 -1 -1 -1 -1  4 -1  4  4 -1  4 -1  1  2  2  3  0  1  0 -1  1
  0  0  1 -1 -1  0  2  1  0  2 -1  1  1 -1 -1  0  1  1 -1  3  1  1 -1  1  1
  0  0 -1  0 -1  0  0  2 -1  1 -1  1  0 -1  2  1  3  1  1 -1  1  0  0 -1  0
  0  3  2  0  0  5 -1  3  2 -1  5  4  4  4 -1  5  5 -1  4  0  4  4  4  5  4
  4  5  5  0  5  4 -1  4  5  5  5  1  5  5  0  5  4  4 -1  4  4  5  4  0  5
  4 -1  0  5  5  5 -1  4  5  5  5  5  4  4]
Noise raito: 22.15%
Estimated number of clusters: 6
Silhouette Coefficient: 0.710
```

每个数据被划分的簇的分类

噪声数据的比例
簇的个数
聚类效果评价指标

```
Cluster  0 :
[22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22,
22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22]
Cluster  1 :
[23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23,
23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 2
3, 23, 23, 23, 23, 23, 23]
Cluster  2 :
[20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20]
Cluster  3 :
[21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21]
Cluster  4 :
[8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8]
Cluster  5 :
[7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7]
```

5.画直方图，分析实验结果

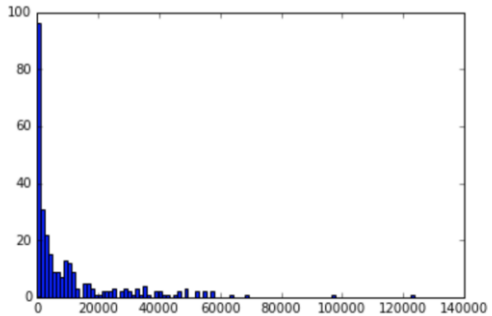



```
import matplotlib.pyplot as plt
plt.hist(X,24)
```

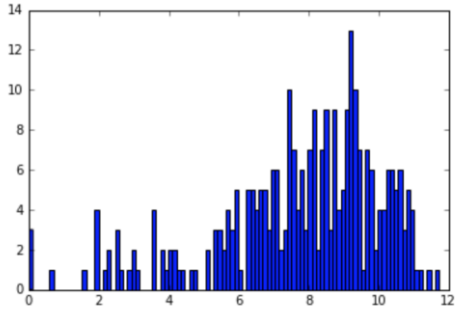
- 转换直方图分析
- 观察：上网时间大多聚集在22：00和23：00

6. 数据分布 vs 聚类

技巧：对数变换



原始数据分布



变换后的数据分布

3-2. **上网时长**聚类，创建DBSCAN算法实例，并进行训练，获得标签：

```
X=np.log(1+real_X[:,1:])
db=skc.DBSCAN(eps=0.14,min_samples=10).fit(X)
labels = db.labels_
```

调用DBSCAN方法进行训练，labels为每个数据的簇标签

```
print('Labels:')
print(labels)
raito=len(labels[labels[:] == -1]) / len(labels)
print('Noise raito:',format(raito, '.2%'))
```

打印数据被记上的标签，计算标签为-1，即噪声数据的比例。

```
# Number of clusters in labels, ignoring noise if present.
n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)

print('Estimated number of clusters: %d' % n_clusters_)
print("Silhouette Coefficient: %0.3f"% metrics.silhouette_score(X, labels))
```

```
for i in range(n_clusters_):
    print('Cluster ',i,':')
    count=len(X[labels == i])
    mean=np.mean(real_X[labels == i][:,1])
    std=np.std(real_X[labels == i][:,1])
    print('\t number of sample: ',count)
    print('\t mean of sample  : ',format(mean,'.1f'))
    print('\t std of sample   : ',format(std,'.1f'))
```

统计每一个簇内的样本个数，均值，标准差

4-2. 输出标签，查看结果

```
Labels:
[ 0  1  0  4  1  2  0  2  0  3 -1  0 -1 -1  0  3  1  0  3  2  2  1  2  0  1
  1 -1 -1  0  0  0  0  1  0 -1  0  0  0  2  0  1  0 -1 -1  0  0  0  3  2  0
 -1  1  0  1  0  0 -1  2  0  0  0  1  3  3  0  2  0 -1  3  0  0  2  0  0  0
  2  1 -1  0  0  0  0  0  0  1 -1  0  3  1  0  1  1  0  1  0  1  0  0 -1  1
  1  0  0  2  0  0  0  2  2  0  0  0 -1  0  0  4  0  1  2 -1  0  1  0  2  0
 -1 -1 -1  0  1  1  3 -1  0  1  0  2  0  0  2  1  1  0  0  0  0  4 -1  0  0
  0  0  2  0  0  0  0 -1  2  0  0  0  0  4  0  0 -1  0  2  0  0 -1  0  1  4
  0  0 -1  1  1  0  0  2  0  0  3 -1 -1 -1  1  0  0  2  1  0 -1 -1  3  2  2
  0  0  3  0  1  0  0  0  3  2  0 -1  0  1 -1 -1  0  2  2  1  4  0  0  1  0
  2  0  0  0  0  1  1  0  0  1  0  4 -1 -1  0  0  0 -1 -1  1 -1  4 -1  0  2
  2 -1  2  1  2 -1  0 -1  0  2  2  1 -1  0  1  2 -1 -1  1 -1  2 -1 -1  1  4
  2  3  1  0  4  0  0  4  2  4  0  0  2 -1]
Noise raito: 16.96%
Estimated number of clusters: 5
Silhouette Coefficient: 0.227
```

```
Cluster  0 :
         number of sample:  128
         mean of sample   :  5864.3
         std of sample    :  3498.1
Cluster  1 :
         number of sample:  46
         mean of sample   :  36835.1
         std of sample    :  11314.1
Cluster  2 :
         number of sample:  40
         mean of sample   :  843.2
         std of sample    :  242.9
Cluster  3 :
         number of sample:  14
         mean of sample   :  16581.6
         std of sample    :  1186.7
Cluster  4 :
         number of sample:  12
         mean of sample   :  338.4
         std of sample    :  31.9
```

Label表示样本的类别，-1表示DBSCAN划分为噪声。

- 按照上网时长DBSCAN聚了5类，右图所示，显示了每个聚类的样本数量、聚类的均值、标准差。
- 时长聚类效果不如时间的聚类效果明显。