聚类案例

导包

In [1]:
```
#K-means,Hierarchical,DBSCAN
#聚类包
from sklearn.cluster import KMeans  #指定K
from sklearn.cluster import AgglomerativeClustering  #指定类别的数量
from sklearn.cluster import DBSCAN  # 不需要指定类别的数量
from sklearn.cluster import MeanShift #  不需要
from sklearn.cluster import SpectralClustering
#预处理包
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
#评价,轮廓系数
from sklearn.metrics import silhouette_score
#常用工具包
import numpy as np
import pandas as pd
import warnings
warnings.filterwarnings("ignore")
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
#读入数据
data=pd.read_csv("d:/datasets/auto-mpg.csv")
```

探索性分析

In [3]:
```
data.head()
```

Out[3]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
0	18.0	8	307.0	130	3504	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165	3693	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150	3436	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150	3433	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140	3449	10.5	70	1	ford torino

In [4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   mpg           398 non-null    float64
 1   cylinders     398 non-null    int64  
 2   displacement  398 non-null    float64
 3   horsepower    398 non-null    object 
 4   weight        398 non-null    int64  
 5   acceleration  398 non-null    float64
 6   model year    398 non-null    int64  
 7   origin        398 non-null    int64  
 8   car name      398 non-null    object 
dtypes: float64(3), int64(4), object(2)
memory usage: 28.1+ KB
```

In [5]:
```
data.describe()
```

Out[5]:

	mpg	cylinders	displacement	weight	acceleration	model year	origin
count	398.000000	398.000000	398.000000	398.000000	398.000000	398.000000	398.000000
mean	23.514573	5.454774	193.425879	2970.424623	15.568090	76.010050	1.572864
std	7.815984	1.701004	104.269838	846.841774	2.757689	3.697627	0.802055
min	9.000000	3.000000	68.000000	1613.000000	8.000000	70.000000	1.000000
25%	17.500000	4.000000	104.250000	2223.750000	13.825000	73.000000	1.000000
50%	23.000000	4.000000	148.500000	2803.500000	15.500000	76.000000	1.000000
75%	29.000000	8.000000	262.000000	3608.000000	17.175000	79.000000	2.000000
max	46.600000	8.000000	455.000000	5140.000000	24.800000	82.000000	3.000000

In [6]:
```
data.sample(5)
```

Out[6]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
354	34.5	4	100.0	?	2320	15.8	81	2	renault 18i
122	24.0	4	121.0	110	2660	14.0	73	2	saab 99le
179	22.0	4	121.0	98	2945	14.5	75	2	volvo 244dl
373	24.0	4	140.0	92	2865	16.4	82	1	ford fairmont futura
68	13.0	8	350.0	155	4502	13.5	72	1	buick lesabre custom

In [7]:
```
data.horsepower.value_counts()
```

Out[7]:
```
150    22
90     20
88     19
110    18
100    17
       ..
122     1
220     1
135     1
148     1
193     1
Name: horsepower, Length: 94, dtype: int64
```

In [8]:
```
#查看object类型特征的取值情况
temp=data.horsepower.value_counts()
```

In [9]:
```
temp.index
```

Out[9]:
```
Index(['150', '90', '88', '110', '100', '75', '95', '70', '105', '67', '65',
       '97', '85', '145', '80', '140', '84', '78', '72', '92', '?', '68',
       '180', '130', '115', '71', '60', '175', '170', '86', '76', '165', '83',
       '52', '120', '215', '225', '63', '190', '74', '112', '125', '48', '96',
       '69', '139', '198', '98', '155', '81', '46', '129', '153', '62', '79',
       '87', '160', '53', '58', '137', '132', '66', '108', '107', '116', '103',
       '158', '64', '49', '200', '152', '93', '208', '61', '113', '142', '102',
       '82', '138', '54', '94', '91', '77', '230', '167', '149', '210', '89',
       '133', '122', '220', '135', '148', '193'],
      dtype='object')
```

In [10]:
```
temp[temp.index=="?"]  #查看horsepower为？的样本的数量
```

Out[10]:
```
?    6
Name: horsepower, dtype: int64
```

In [11]:
```
data_auto=data.drop(["car name","origin","model year"],axis=1)
#删除对聚类无用的特征
```

In [12]:
```
data_auto.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 6 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   mpg           398 non-null    float64
 1   cylinders     398 non-null    int64  
 2   displacement  398 non-null    float64
 3   horsepower    398 non-null    object 
 4   weight        398 non-null    int64  
 5   acceleration  398 non-null    float64
dtypes: float64(3), int64(2), object(1)
memory usage: 18.8+ KB
```

In [13]:
```
print(data_auto[data_auto["horsepower"]=="?"])
```

```
      mpg  cylinders  displacement horsepower  weight  acceleration
32   25.0          4          98.0          ?    2046          19.0
126  21.0          6         200.0          ?    2875          17.0
330  40.9          4          85.0          ?    1835          17.3
336  23.6          4         140.0          ?    2905          14.3
354  34.5          4         100.0          ?    2320          15.8
374  23.0          4         151.0          ?    3035          20.5
```

In [14]:
```
print(data_auto[data_auto["horsepower"]=="?"].index)
#查看horsepower取值为？的样本的index
```

```
Int64Index([32, 126, 330, 336, 354, 374], dtype='int64')
```

In [15]:
```
data_auto.drop(data_auto[data_auto["horsepower"]=="?"].index,inplace=True)
#删除horsepower取值为？的样本
```

In [16]:
```
data_auto.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 392 entries, 0 to 397
Data columns (total 6 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   mpg           392 non-null    float64
 1   cylinders     392 non-null    int64  
 2   displacement  392 non-null    float64
 3   horsepower    392 non-null    object 
 4   weight        392 non-null    int64  
 5   acceleration  392 non-null    float64
dtypes: float64(3), int64(2), object(1)
memory usage: 21.4+ KB
```

In [17]:
```
data_auto.describe()
```

Out[17]:

	mpg	cylinders	displacement	weight	acceleration
count	392.000000	392.000000	392.000000	392.000000	392.000000
mean	23.445918	5.471939	194.411990	2977.584184	15.541327
std	7.805007	1.705783	104.644004	849.402560	2.758864
min	9.000000	3.000000	68.000000	1613.000000	8.000000
25%	17.000000	4.000000	105.000000	2225.250000	13.775000
50%	22.750000	4.000000	151.000000	2803.500000	15.500000
75%	29.000000	8.000000	275.750000	3614.750000	17.025000
max	46.600000	8.000000	455.000000	5140.000000	24.800000

标准化

In [18]:
```
model_sc=StandardScaler()  #实例化对象StandardScaler，Z分数标准化
```

In [19]:
```
model_sc.fit(data_auto)  #训练/拟合，计算均值与方差
```

Out[19]:
```
StandardScaler()
```

In [20]:
```
model_sc.mean_  #查看均值
```

Out[20]:
```
array([  23.44591837,    5.47193878,  194.4119898 ,  104.46938776,
        2977.58418367,   15.54132653])
```

In [21]:
```
model_sc.var_  #查看标准差
```

Out[21]:
```
array([6.07627384e+01, 2.90227379e+00, 1.09224329e+04, 1.47778988e+03,
       7.19644187e+05, 7.59191457e+00])
```

In [22]:
```
data_auto_sc=model_sc.transform(data_auto)  #标准化
```

In [24]:
```
pd.DataFrame(data_auto_sc,columns=data_auto.columns).describe()  #查看标准化后的数据集
```

Out[24]:

	mpg	cylinders	displacement	horsepower	weight	acceleration
count	3.920000e+02	3.920000e+02	3.920000e+02	3.920000e+02	3.920000e+02	3.920000e+02
mean	1.450087e-16	-1.087565e-16	-7.250436e-17	-1.812609e-16	-1.812609e-17	4.350262e-16
std	1.001278e+00	1.001278e+00	1.001278e+00	1.001278e+00	1.001278e+00	1.001278e+00
min	-1.853218e+00	-1.451004e+00	-1.209563e+00	-1.520975e+00	-1.608575e+00	-2.736983e+00
25%	-8.269250e-01	-8.640136e-01	-8.555316e-01	-7.665929e-01	-8.868535e-01	-6.410551e-01
50%	-8.927701e-02	-8.640136e-01	-4.153842e-01	-2.853488e-01	-2.052109e-01	-1.499869e-02
75%	7.125143e-01	1.483947e+00	7.782764e-01	5.600800e-01	7.510927e-01	5.384714e-01
max	2.970359e+00	1.483947e+00	2.493416e+00	3.265452e+00	2.549061e+00	3.360262e+00

In [26]:
```
pd.DataFrame(data_auto_sc,columns=data_auto.columns).info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 392 entries, 0 to 391
Data columns (total 6 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   mpg           392 non-null    float64
 1   cylinders     392 non-null    float64
 2   displacement  392 non-null    float64
 3   horsepower    392 non-null    float64
 4   weight        392 non-null    float64
 5   acceleration  392 non-null    float64
dtypes: float64(6)
memory usage: 18.5 KB
```

KMeans

In [28]:
```
#聚类。实例化模型，训练模型
model_km=KMeans(n_clusters=4,random_state=10)
model_km.fit(data_auto_sc)
```

Out[28]:
```
KMeans(n_clusters=4, random_state=10)
```

In [29]:
```
model_km.labels_  #查看聚类结果，样本的标签
```

Out[29]:
```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 3, 3, 3, 2, 0, 2, 2,
       2, 2, 3, 1, 1, 1, 1, 2, 2, 2, 3, 3, 3, 3, 3, 1, 1, 1, 1, 1, 1, 1,
       3, 0, 3, 3, 2, 2, 0, 2, 0, 0, 0, 0, 2, 2, 0, 0, 2, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 2, 1, 1, 1, 1, 2, 2, 0, 0, 2, 2, 2, 2, 2, 2, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 3, 3, 3, 0, 1, 1, 1, 1, 3, 0, 0, 2,
       2, 2, 3, 2, 1, 1, 0, 2, 2, 2, 1, 2, 3, 1, 3, 3, 3, 0, 2, 0, 2, 3,
       3, 3, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 0, 0, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3,
       1, 1, 1, 1, 3, 3, 3, 3, 3, 3, 1, 2, 2, 3, 0, 2, 2, 2, 3, 2, 3, 2,
       2, 2, 2, 0, 2, 2, 2, 0, 2, 1, 1, 1, 1, 3, 3, 3, 3, 0, 0, 2, 0, 3,
       3, 3, 3, 2, 0, 2, 2, 2, 1, 0, 3, 3, 1, 1, 1, 1, 0, 2, 0, 2, 0, 1,
       3, 1, 1, 3, 3, 3, 3, 1, 1, 1, 1, 2, 2, 0, 2, 0, 2, 2, 2, 3, 2, 2,
       0, 2, 0, 0, 0, 3, 1, 1, 3, 3, 3, 2, 3, 3, 3, 3, 3, 3, 1, 1, 1, 1,
       2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 2, 3, 2, 2, 3, 3, 2, 3, 3, 1, 1, 1,
       1, 1, 1, 3, 1, 2, 2, 2, 2, 3, 3, 0, 3, 2, 2, 0, 2, 2, 2, 2, 2, 2,
       0, 2, 0, 2, 0, 0, 3, 2, 2, 0, 2, 2, 0, 2, 0, 0, 0, 0, 0, 2, 0, 2,
       2, 2, 2, 0, 2, 2, 2, 3, 2, 0, 0, 0, 0, 0, 0, 2, 2, 0, 0, 2, 2, 0,
       2, 0, 0, 0, 3, 3, 3, 3, 3, 3, 0, 0, 0, 2, 2, 0, 2, 2, 0, 0, 2, 0,
       2, 2, 0, 2, 2, 0, 3, 3, 2, 3, 2, 2, 2, 2, 0, 2, 0, 0])
```

In [30]:
```
model_km.cluster_centers_  # #查看聚类结果，类的中心
```

Out[30]:
```
array([[ 1.03528342, -0.84053395, -0.89278468, -0.95458153, -0.87730913,
         1.24461464],
       [-1.14393843,  1.47184414,  1.46979213,  1.49145953,  1.37493755,
        -1.05253303],
       [ 0.56242862, -0.85518664, -0.75866035, -0.48846684, -0.71697519,
        -0.16890336],
       [-0.47686477,  0.39093089,  0.29069931, -0.09323987,  0.31939015,
         0.35877876]])
```

In [31]:
```
pd.DataFrame(model_sc.inverse_transform(model_km.cluster_centers_),\
             columns=data_auto.columns)  #逆标准化聚类中心
```

Out[31]:

	mpg	cylinders	displacement	horsepower	weight	acceleration
0	31.516000	4.040000	101.106667	67.773333	2233.346667	18.970667
1	14.528866	7.979381	348.020619	161.804124	4143.969072	12.641237
2	27.830075	4.015038	115.124060	85.691729	2369.360902	15.075940
3	19.728736	6.137931	224.793103	100.885057	3248.528736	16.529885

In [32]:
```
auto_label=model_km.labels_
```

In [33]:
```
data_auto["label"]=auto_label #聚类结果放置到原数据集的最后一列
```

In [34]:
```
data_auto.sample(5)
```

Out[34]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	label
42	12.0	8	383.0	180	4955	11.5	1
91	13.0	8	400.0	150	4464	12.0	1
260	18.6	6	225.0	110	3620	18.7	3
289	16.9	8	350.0	155	4360	14.9	1
393	27.0	4	140.0	86	2790	15.6	2

In [35]:
```
data_auto.groupby("label").mean()  #再次根据聚类的结果，计算类别的中心。与上面的结果一致
```

Out[35]:

label	mpg	cylinders	displacement	weight	acceleration
0	31.516000	4.040000	101.106667	2233.346667	18.970667
1	14.528866	7.979381	348.020619	4143.969072	12.641237
2	27.830075	4.015038	115.124060	2369.360902	15.075940
3	19.728736	6.137931	224.793103	3248.528736	16.529885

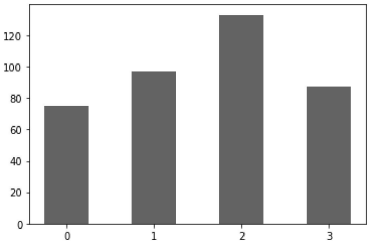
In [36]:
```
lbs=pd.Series(auto_label).value_counts()  #计算各类别的数量，保存到lbs
```

In [38]:
```
lbs
```

Out[38]:
```
2    133
1     97
3     87
0     75
dtype: int64
```

In [37]:
```
#用柱状图可视化类别的数量
plt.bar(x=lbs.index,height=lbs,width=0.5)
plt.xticks([0,1,2,3])
```

Out[37]:
```
([<matplotlib.axis.XTick at 0x2174c2f0e80>,
  <matplotlib.axis.XTick at 0x2174c2f0910>,
  <matplotlib.axis.XTick at 0x2174c1136a0>,
  <matplotlib.axis.XTick at 0x2174c663b20>],
 [Text(0, 0, ''), Text(0, 0, ''), Text(0, 0, ''), Text(0, 0, '')])
```



In [39]:
```
#用轮廓系数评价不同类别数量的聚类结果
sil=[]
print("簇数   轮廓系数")
for k in [2,3,4,6,10,20]:
    model_km=KMeans(n_clusters=k,random_state=10).fit(data_auto_sc)
    print(k,"   ",silhouette_score(data_auto_sc,model_km.labels_))
```

```
簇数   轮廓系数
2    0.5450184683536872
3    0.44234710113179243
4    0.38192260208353274
6    0.3312892026371077
10    0.25381373819967257
20    0.2626993017075033
```

In [40]:
```
#列表推导式，功能同上。
[silhouette_score(data_auto_sc,KMeans(n_clusters=k,random_state=10).fit(data_auto_sc).labels_) for k in [2,3,4,6,300,391]]
```

Out[40]:
```
[0.5450184683536872,
 0.44234710113179243,
 0.38192260208353274,
 0.3312892026371077,
 0.1369957047218599,
 0.0051020408163265 3]
```

MeanShift

In [41]:
```
model_mn=MeanShift(bandwidth=2).fit(data_auto_sc) #实例化，训练/拟合。
auto_label=model_mn.labels_   #保存结果，类别
auto_cluster=model_mn.cluster_centers_  #保存结果，聚类中心
```

In [37]:
```
auto_label
```

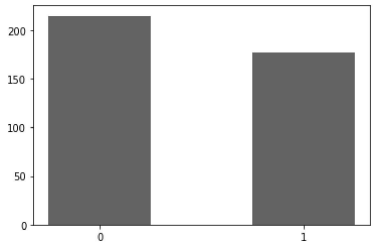
Out[37]:
```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1,
       1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0,
       0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1,
       1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0], dtype=int64)
```

In [38]:
```
pd.Series(auto_label).value_counts()
```

Out[38]:
```
0    215
1    177
dtype: int64
```

In [39]:
```
lbs=pd.Series(auto_label).value_counts()
plt.bar(x=lbs.index,height=lbs,width=0.5)
plt.xticks([0,1])
```

Out[39]:
```
([<matplotlib.axis.XTick at 0x25464708790>,
  <matplotlib.axis.XTick at 0x25464d5ea90>],
 [Text(0, 0, ''), Text(0, 0, '')])
```



In [40]:
```
model_sc.inverse_transform(auto_cluster)
```

Out[40]:
```
array([[  27.92336449,    4.21495327,  118.88551402,   82.11214953,
         2379.48598131,   16.09953271],
       [  18.00075758,    6.84090909,  259.15909091,  118.81060606,
         3505.62878788,   15.04242424]])
```

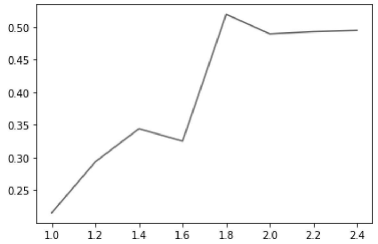
In [43]:
```
silhouette_score(data_auto_sc,auto_label)
```

Out[43]:
```
0.48929237785064933
```

In [42]:
```
bandwidth_grid=np.arange(1,2.5,0.2)
cluster_number=[]
slt_score=[]
for i in bandwidth_grid:
    model=MeanShift(bandwidth=i).fit(data_auto_sc)
    cluster_number.append(len(np.unique(model.labels_)))
    slt_score.append(silhouette_score(data_auto_sc,model.labels_))
```

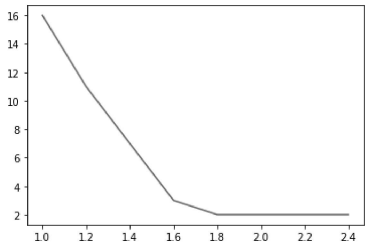
In [43]:
```
plt.plot(np.arange(1,2.5,0.2),slt_score)
```

Out[43]:
```
[<matplotlib.lines.Line2D at 0x2174c831310>]
```



In [44]:
```
plt.plot(np.arange(1,2.5,0.2),cluster_number)
```

Out[44]:
```
[<matplotlib.lines.Line2D at 0x2174cbfc880>]
```



In [46]:
```
from prettytable import PrettyTable
x = PrettyTable(["窗宽","族的个数","轮廓系数"])
#x.align["窗宽"] = "l" #以姓名字段左对齐
#x.padding_width = 1 # 填充宽度
for i,j,k in zip(bandwidth_grid,cluster_number,slt_score):
    x.add_row([round(i,4),j,round(k,4)])
print(x)
```

```
+------+----------+----------+
| 窗宽 | 族的个数 | 轮廓系数 |
+------+----------+----------+
| 1.0  |    16    |  0.2155  |
| 1.2  |    11    |  0.2936  |
| 1.4  |    7     |  0.3441  |
| 1.6  |    3     |  0.3251  |
| 1.8  |    2     |  0.5193  |
| 2.0  |    2     |  0.4893  |
| 2.2  |    2     |  0.4928  |
| 2.4  |    2     |  0.4947  |
+------+----------+----------+
```

SpectralClustering

In [47]:
```
from sklearn.cluster import SpectralClustering
```

In [48]:
```
model= SpectralClustering(n_clusters=3)
model.fit(data_auto_sc)
auto_label=model.labels_
```

In [49]:
```
auto_label
```

Out[49]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 2, 2, 1, 1, 1,
       1, 1, 2, 0, 0, 0, 0, 1, 1, 1, 2, 2, 2, 2, 2, 0, 0, 0, 0, 0, 0,
       2, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 2, 2, 2, 1, 0, 0, 0, 0, 2, 1,
       1, 1, 2, 1, 0, 0, 1, 1, 1, 1, 0, 1, 2, 0, 2, 2, 2, 2, 1, 1, 1,
       2, 2, 2, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2,
       0, 0, 0, 0, 2, 2, 2, 2, 2, 0, 0, 1, 1, 2, 1, 1, 1, 1, 2, 1, 2,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 2, 2, 2, 2, 1, 1, 1, 2,
       2, 2, 2, 1, 1, 1, 1, 1, 0, 1, 2, 2, 0, 0, 0, 1, 1, 1, 1, 1, 0,
       2, 0, 0, 2, 2, 2, 2, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1,
       1, 1, 1, 1, 1, 2, 0, 0, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 0, 0, 0,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 2, 1, 1, 2, 2, 1, 2, 2, 0, 0,
       0, 0, 0, 0, 0, 1, 1, 1, 1, 2, 0, 1, 2, 1, 1, 1, 1, 1, 2, 1, 1,
       1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 2, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1])
```

In [51]:
```
data_auto.label=auto_label
```

In [52]:
```
data_auto.groupby("label").mean()
```

Out[52]:

label	mpg	cylinders	displacement	weight	acceleration
0	14.715000	7.980000	346.370000	4126.910000	12.721000
1	29.170335	4.028708	110.595694	2324.263158	16.497129
2	19.550602	6.084337	222.385542	3237.963855	16.532530

Hierarchical

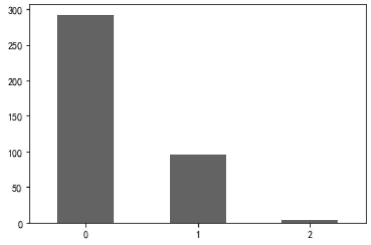
In [53]:
```
#设置字体为SimHei显示中文
plt.rcParams['font.sans-serif'] = 'SimHei'
#设置正常显示字符
plt.rcParams['axes.unicode_minus'] = False
model=AgglomerativeClustering(n_clusters=3,linkage="average").fit(data_auto_sc)
auto_label=model.labels_
```

In [54]:
```
auto_label
```

Out[54]:
```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1,
       1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0,
       0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1,
       0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       2, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
       1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0], dtype=int64)
```

In [59]:
```
lbs=pd.Series(auto_label).value_counts()
#plt.bar(x=lbs.index,height=lbs )
lbs.plot(kind="bar",rot=0)
```

Out[59]: <AxesSubplot:>

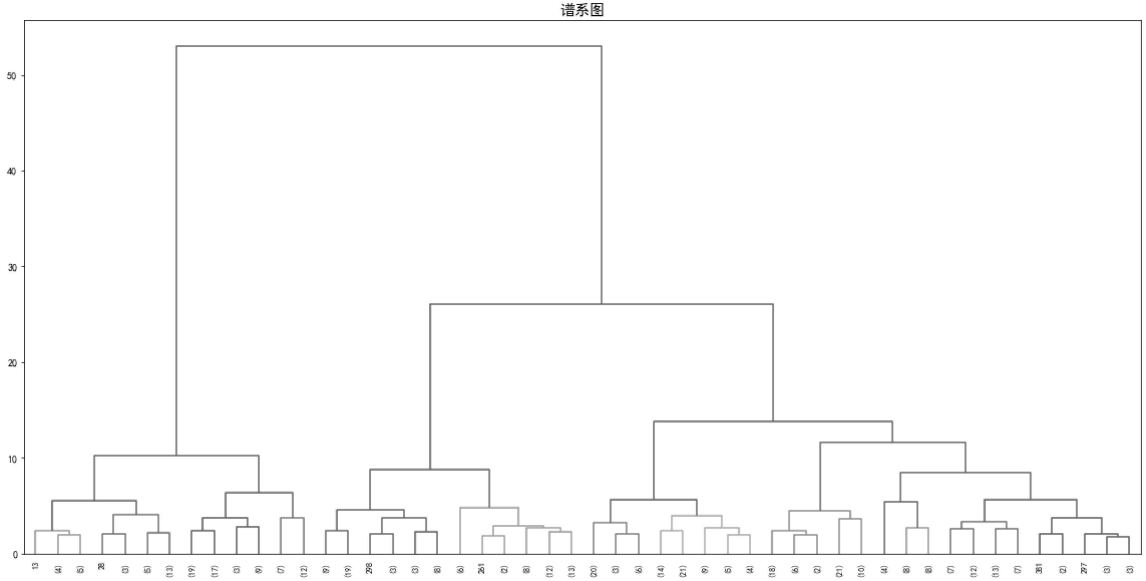


In [60]:
```
silhouette_score(data_auto_sc,auto_label)
```

Out[60]: 0.4132744493396478

In [61]:
```
# 绘制谱系图
from scipy.spatial.distance import pdist
from scipy.cluster.hierarchy import linkage, dendrogram
import matplotlib.pyplot as plt
#利用scipy中pdist,linkage,dendrogram函数绘制谱系图
#pdist函数返回距离矩阵，linkage函数返回一个ndarray对象，描述了簇合并的过程
#dendrogram函数用来绘制谱系图
row_clusters = linkage(pdist(data_auto_sc,metric='euclidean'),method='ward')
fig = plt.figure(figsize=(16,8))
#参数p和参数truncate_mode用来将谱系图截断，部分结点的子树被剪枝，横轴显示的是该结点包含的样本数
row_dendr = dendrogram(row_clusters, p=50, truncate_mode='lastp',color_threshold=5)
plt.tight_layout()
plt.title('谱系图', fontsize=15)
```

Out[61]: Text(0.5, 1.0, '谱系图')



In [62]:
```
data_auto_sc_0=pd.DataFrame(data_auto_sc,columns=data_auto.columns[:-1])
data_auto_sc_0["class"]=auto_label
```

In [65]:
```
data_auto["class"]=auto_label
```

In [66]:
```
data_auto.groupby("class").mean()
```

Out[66]:

class	mpg	cylinders	displacement	weight	acceleration	label
0	26.110959	4.660959	144.916096	2603.280822	16.396918	1.270548
1	14.495833	8.000000	349.239583	4151.250000	12.633333	0.000000
2	43.700000	4.000000	91.750000	2133.750000	22.875000	1.000000

In [ ]:

In [ ]:

In [63]:
```
data_auto_sc_0.head()
```

Out[63]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	class
0	-0.698638	1.483947	1.077290	0.664133	0.620540	-1.285258	1
1	-1.083498	1.483947	1.488732	1.574594	0.843334	-1.466724	1
2	-0.698638	1.483947	1.182542	1.184397	0.540382	-1.648189	1
3	-0.955212	1.483947	1.048584	1.184397	0.536845	-1.285258	1
4	-0.826925	1.483947	1.029447	0.924265	0.555706	-1.829655	1

In [64]:
```
model_sc.inverse_transform(data_auto_sc_0.groupby("class").mean())
```

Out[64]:
```
array([[2.61109589e+01, 4.66095890e+00, 1.44916096e+02, 8.63904110e+01,
        2.60328082e+03, 1.63969178e+01],
       [1.44958333e+01, 8.00000000e+00, 3.49239583e+02, 1.61770833e+02,
        4.15125000e+03, 1.26333333e+01],
       [4.37000000e+01, 4.00000000e+00, 9.17500000e+01, 4.90000000e+01,
        2.13375000e+03, 2.28750000e+01]])
```

In [67]:
```
model=AgglomerativeClustering(n_clusters=3,linkage="complete").fit(data_auto_sc)
auto_label=model.labels_
```

In [69]:
```
data_auto["class"]=auto_label
```

In [70]:
```
data_auto.groupby("class").mean()
```

Out[70]:

class	mpg	cylinders	displacement	weight	acceleration	label
0	24.995814	4.813953	154.788372	2665.623256	15.692093	1.372093
1	14.684000	7.980000	345.470000	4121.560000	12.702000	0.020000
2	30.497403	4.051948	108.870130	2362.961039	18.807792	1.012987

In [ ]:
```

```

DBSCAN

In [71]:
```
# 训练模型
model = DBSCAN(eps=1,min_samples=2).fit(data_auto_sc)
# 输出模型结果
auto_label = model.labels_
```

In [74]:
```
# 核心对象的索引
len(model.core_sample_indices_)
```

Out[74]:
```
383
```

In [73]:
```
# 输出核心对象
model.components_
```

Out[73]:
```
array([[-0.69863841,  1.48394702,  1.07728956,  0.66413273,  0.62054034,
        -1.285258  ],
       [-1.08349824,  1.48394702,  1.48873169,  1.57459447,  0.84333403,
        -1.46672362],
       [-0.69863841,  1.48394702,  1.1825422 ,  1.18439658,  0.54038176,
        -1.64818924],
       ...,
       [ 1.09737414, -0.86401356, -0.56847897, -0.53247413, -0.80463202,
        -1.4304305 ],
       [ 0.5842277 , -0.86401356, -0.7120053 , -0.66254009, -0.41562716,
         1.11008813],
       [ 0.96908753, -0.86401356, -0.72157372, -0.58450051, -0.30364091,
         1.40043312]])
```

In [75]:
```
# 样本的类别标签
labels = model.labels_
```

In [76]:
```
data_auto_sc
```

Out[76]:
```
array([[-0.69863841,  1.48394702,  1.07728956,  0.66413273,  0.62054034,
        -1.285258  ],
       [-1.08349824,  1.48394702,  1.48873169,  1.57459447,  0.84333403,
        -1.46672362],
       [-0.69863841,  1.48394702,  1.1825422 ,  1.18439658,  0.54038176,
        -1.64818924],
       ...,
       [ 1.09737414, -0.86401356, -0.56847897, -0.53247413, -0.80463202,
        -1.4304305 ],
       [ 0.5842277 , -0.86401356, -0.7120053 , -0.66254009, -0.41562716,
         1.11008813],
       [ 0.96908753, -0.86401356, -0.72157372, -0.58450051, -0.30364091,
         1.40043312]])
```

In [83]:
```
data_auto
```

Out[83]:

	mpg	cylinders	displacement	horsepower	weight	acceleration
0	18.0	8	307.0	130	3504	12.0
1	15.0	8	350.0	165	3693	11.5
2	18.0	8	318.0	150	3436	11.0
3	16.0	8	304.0	150	3433	12.0
4	17.0	8	302.0	140	3449	10.5
...	...	...	...	...	...	...
393	27.0	4	140.0	86	2790	15.6
394	44.0	4	97.0	52	2130	24.6
395	32.0	4	135.0	84	2295	11.6
396	28.0	4	120.0	79	2625	18.6
397	31.0	4	119.0	82	2720	19.4

392 rows × 6 columns

In [85]:
```
data_auto.drop(["label","class"],axis=1,inplace=True)
```

In [86]:
```
data_auto["class"]=labels
```

In [87]:
```
data_auto.groupby("class").mean()
```

Out[87]:

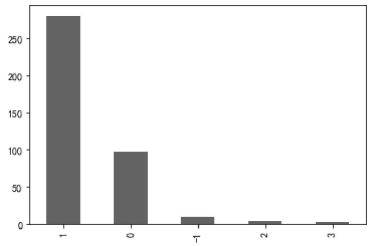
class	mpg	cylinders	displacement	weight	acceleration
-1	23.766667	6.555556	242.111111	3450.888889	18.266667
0	14.718557	8.000000	345.948454	4132.917526	12.662887
1	26.160000	4.564286	140.737500	2568.221429	16.327143
2	43.700000	4.000000	91.750000	2133.750000	22.875000
3	24.800000	8.000000	350.000000	3812.500000	18.200000

In [88]:
```
# 标签中的簇数，忽略噪声点
n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)
print('簇数: %d' % n_clusters_)
print("轮廓系数(Silhouette Coefficient): %0.4f"% silhouette_score(data_auto_sc, labels))
```

```
簇数: 4
轮廓系数(Silhouette Coefficient): 0.3084
```

In [93]:
```
pd.Series(labels).value_counts().plot(kind="bar")
```

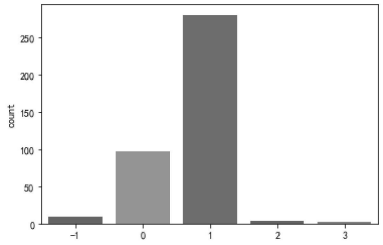
Out[93]: <AxesSubplot:>



In [95]:
```
import seaborn as sns
```

In [96]:
```
sns.countplot(labels)
```

Out[96]: <AxesSubplot:ylabel='count'>



In [ ]:
```

```