1 导入所用的包

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import warnings
from sklearn import cluster
from sklearn.cluster import KMeans
from sklearn import metrics
from sklearn import decomposition
from IPython.core.interactiveshell import InteractiveShell

warnings.filterwarnings('ignore')
InteractiveShell.ast_node_interactivity = "all"
```

executed in 2.69s, finished 08:01:58 2021-06-20

2 数据的读取与预处理（此处采用归一化为min-max标准化）

In [2]:

```
data = pd.read_csv(r"C:\Users\38061\Jupyter_Notebook\shuju
print('初始状态下的df文件***********')
data

load = data['LOAD']
load = load.values.reshape(-1,24)

hangtime = pd.date_range('2005-01-01','2010-09-30')

lietime =[]
for i in range(1,25):
    lietime.append(i)

df = pd.DataFrame(load,index=hangtime,columns=lietime)
print('构建N×24的特征矩阵的df文件***********')
df



df_norm = (df - df.min()) / (df.max() - df.min())
print('使用min-max标准化后状态下的df文件***********')
df_norm
```

executed in 410ms, finished 08:01:59 2021-06-20

初始状态下的df文件***********

	TIMESTAMP	LOAD
0	112005 1:00	125.8
1	112005 2:00	121.8
2	112005 3:00	117.0
3	112005 4:00	114.4

	TIMESTAMP	LOAD
4	112005 5:00	113.6
...	...	...
50371	9302010 20:00	176.8
50372	9302010 21:00	169.4
50373	9302010 22:00	155.8
50374	9302010 23:00	143.8
50375	1012010 0:00	130.2

构建N×24的特征矩阵的df文件***********

	1	2	3	4	5	6	7	8
2005-01-01	125.8	121.8	117.0	114.4	113.6	116.1	121.0	127.8
2005-01-02	102.2	99.6	99.2	101.2	103.9	109.0	116.9	129.0
2005-01-03	96.7	94.8	95.0	97.8	103.0	114.3	130.4	136.4
2005-01-04	84.9	81.5	79.9	80.1	83.5	93.3	111.8	114.6
2005-01-05	75.1	71.4	69.8	70.5	74.9	86.1	110.5	112.8
...	...	...	...	...	...	...	...	...
2010-09-26	146.9	137.3	130.5	127.1	124.9	124.5	125.9	134.2
2010-09-27	123.2	116.7	113.8	112.2	113.6	121.3	139.0	146.2
2010-09-28	109.7	104.9	102.7	102.1	104.7	112.7	129.9	140.3
2010-09-29	107.6	101.6	98.3	97.0	98.4	106.0	124.3	131.7
2010-09-30	115.6	112.1	110.6	111.0	107.1	118.3	130.5	142.1

2099 rows × 24 columns

使用min-max标准化后状态下的df文件***********

	1	2	3	4	5	
2005-01-01	0.349856	0.335193	0.308038	0.287527	0.265948	0.23
2005-01-02	0.236285	0.231985	0.228110	0.229759	0.224138	0.20
2005-01-03	0.209817	0.209670	0.209250	0.214880	0.220259	0.22
2005-01-04	0.153032	0.147838	0.141446	0.137418	0.136207	0.13
2005-01-05	0.105871	0.100883	0.096093	0.095405	0.099138	0.10
...	...	...	...	...	...	...
2010-09-26	0.451396	0.407252	0.368657	0.343107	0.314655	0.27
2010-09-27	0.337344	0.311483	0.293669	0.277899	0.265948	0.25
2010-09-28	0.272377	0.256625	0.243826	0.233698	0.227586	0.22
2010-09-29	0.262271	0.241283	0.224068	0.211379	0.200431	0.19
2010-09-30	0.300770	0.290098	0.279300	0.272648	0.237931	0.24

2099 rows × 24 columns

3 分别使用Silhouette系数，Calinski-Harabaz指数和Davies-Bouldin Index来评估模型

3.1 搜索使用Kmean++的情况下较优k值

In [3]:

```
scores1 = []
scores2 = []
scores3 = []
x = np.arange(2,500)
for i in x:
    model = KMeans(n_clusters=i,init='k-means++', random_s
    yhat = model.fit_predict(df_norm)
    #new1查看各个类数量
    #print('当k='+ str(i) +'的时候分类及其各个分类数量')
    #print(np.unique(yhat,return_counts=True))
    labels = model.labels_
    score1 = metrics.silhouette_score(df_norm, labels, met
    score2 = metrics.calinski_harabasz_score(df_norm, labe
    score3 = metrics.davies_bouldin_score(df_norm, labels)
    scores1.append(score1)
    scores2.append(score2)
    scores3.append(score3)
#scores1
#scores2
#scores3
```

executed in 27m 46s, finished 08:29:45 2021-06-20

3.2 可视化k值与各项指标的关系

In [4]:

```
plt.rcParams["font.sans-serif"] = ["SimHei"]
plt.figure(dpi=400, figsize=[20, 15])
plt.plot(x, scores1)
plt.title("Silhouette系数与k取值关系曲线", fontsize=30)
plt.xlabel("k的取值", fontsize=30)
plt.ylabel("Silhouette系数得分", fontsize=30)
plt.xticks(fontsize=30)
plt.yticks(fontsize=30)
plt.show()

plt.rcParams["font.sans-serif"] = ["SimHei"]
plt.figure(dpi=400, figsize=[20, 15])
plt.plot(x, scores2)
plt.title("Calinski-Harabaz指数与k取值关系曲线", fontsize=
plt.xlabel("k的取值", fontsize=30)
plt.ylabel("Calinski-Harabaz指数", fontsize=30)
plt.xticks(fontsize=30)
plt.yticks(fontsize=30)
plt.show()

plt.rcParams["font.sans-serif"] = ["SimHei"]
plt.figure(dpi=400, figsize=[20, 15])
plt.plot(x, scores3)
plt.title("Davies-Bouldin Index与k取值关系曲线", fontsize=
plt.xlabel("k的取值", fontsize=30)
plt.ylabel("Davies-Bouldin Index得分", fontsize=30)
plt.xticks(fontsize=30)
plt.yticks(fontsize=30)
plt.show()
```

executed in 6.86s, finished 08:29:52 2021-06-20

```
<Figure size 8000x6000 with 0 Axes>
```

```
[<matplotlib.lines.Line2D at 0x1c09941dc40>]
```

```
Text(0.5, 1.0, 'Silhouette系数与k取值关系曲线')
```

```
Text(0.5, 0, 'k的取值')
```

3.3 查看使用三个指标情况下分类的情况

In [5]:

```
model = KMeans(n_clusters=scores1.index(max(scores1))+3, in
yhat = model.fit_predict(df_norm)
    #new1查看各个类数量
print('当k='+ str(scores1.index(max(scores1))+3) +'的时候分
print(np.unique(yhat,return_counts=True))
labels1 = model.labels_

model = KMeans(n_clusters=scores2.index(max(scores2))+3, in
yhat = model.fit_predict(df_norm)
    #new1查看各个类数量
print('当k='+ str(scores2.index(max(scores2))+3) +'的时候分
print(np.unique(yhat,return_counts=True))
labels2 = model.labels_

model = KMeans(n_clusters=scores3.index(max(scores3))+3, in
yhat = model.fit_predict(df_norm)
    #new1查看各个类数量
print('当k='+ str(scores3.index(max(scores3))+3) +'的时候分
print(np.unique(yhat,return_counts=True))
labels3 = model.labels_
```

executed in 1.57s, finished 08:29:53 2021-06-20

```
当k=4的时候分类及其各个分类数量
(array([0, 1, 2, 3]), array([384, 900, 404, 411], dtype=
int64))
当k=3的时候分类及其各个分类数量
(array([0, 1, 2]), array([ 410,  592, 1097], dtype=int6
4))
当k=74的时候分类及其各个分类数量
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11,
12, 13, 14, 15, 16,
       17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 2
9, 30, 31, 32, 33,
       34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 4
```

```
6, 47, 48, 49, 50,
        51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 6
3, 64, 65, 66, 67,
        68, 69, 70, 71, 72, 73]), array([30, 52, 23, 24,
34, 13, 39, 40, 35, 33, 10, 52, 53, 22, 31, 26, 71,
        12, 53,  7, 26, 10, 23,  5, 14, 48, 21, 11, 35, 4
2,  4, 14, 22, 10,
        21, 21, 45, 32, 86, 38, 10, 21, 35, 48, 24, 43, 2
4, 19,  7, 24, 11,
        29, 25, 20, 17, 17, 14, 53, 26, 14, 12, 32, 24, 2
0, 29, 28, 38, 44,
        44, 56, 16,  2, 42, 43], dtype=int64))
```

3.4 可视化聚类分布

In [6]:

```
plt.rcParams["font.sans-serif"] = ["SimHei"]
plt.figure(dpi=400, figsize=[30, 15])
plt.scatter(hangtime, labels1)
plt.title("使用Silhouette系数分类关系图", fontsize=50)
plt.xlabel("时间", fontsize=50)
plt.ylabel("分类结果", fontsize=50)
plt.xticks(rotation=90)
plt.grid(True)
plt.xticks(pd.date_range('2005-01-01','2010-09-30',freq='1
plt.yticks(fontsize=50)
plt.show()

plt.rcParams["font.sans-serif"] = ["SimHei"]
plt.figure(dpi=400, figsize=[30, 15])
plt.scatter(hangtime, labels2)
plt.title("使用Calinski-Harabaz指数分类关系图", fontsize=5
plt.xlabel("时间", fontsize=50)
plt.ylabel("分类结果", fontsize=50)
plt.xticks(rotation=90)
plt.grid(True)
plt.xticks(pd.date_range('2005-01-01','2010-09-30',freq='1
plt.yticks(fontsize=50)
plt.show()

plt.rcParams["font.sans-serif"] = ["SimHei"]
plt.figure(dpi=500, figsize=[40, 30])
plt.scatter(hangtime, labels3)
plt.title("使用Davies-Bouldin Index分类关系图", fontsize=5
plt.xlabel("时间", fontsize=50)
plt.ylabel("分类结果", fontsize=50)
plt.xticks(rotation=90)
plt.grid(True)
plt.xticks(pd.date_range('2005-01-01','2010-09-30',freq='1
plt.yticks(fontsize=50)
```

```
plt.show()
```

executed in 23.8s, finished 08:30:17 2021-06-20

```
<Figure size 12000x6000 with 0 Axes>

<matplotlib.collections.PathCollection at 0x1c09a652970>

Text(0.5, 1.0, '使用Silhouette系数分类关系图')

Text(0.5, 0, '时间')

Text(0, 0.5, '分类结果')
```

4 PCA降维

In [7]:

```
pca = decomposition.PCA()
pca.fit(df_norm)
print('24维数据经过PCA计算后的数值')
print(pca.explained_variance_)
# 选择较为重要的8维数据
pca.n_components = 8
pcadf = pca.fit_transform(df_norm)
print('经过PCA降维得到的八维数据')
pcadf
```

executed in 128ms, finished 08:30:17 2021-06-20

```
PCA()
```

```
24维数据经过PCA计算后的数值
[6.48146075e-01 2.06776108e-01 1.97941109e-02 6.50353215e-
03
 3.32043729e-03 1.83025309e-03 7.49834924e-04 5.01548498e-
04
 2.98562574e-04 2.22567146e-04 1.79393699e-04 1.14667975e-
04
 6.14310726e-05 3.68989480e-05 2.99177860e-05 2.52300916e-
05
 1.45134711e-05 1.25879902e-05 1.09560955e-05 7.14689198e-
06
 6.27973783e-06 5.84517062e-06 4.26437306e-06 2.29631460e-
06]
经过PCA降维得到的八维数据
```

```
array([[-7.32432017e-01,  1.90463227e-01, -1.71867799e-0
1, ...,
        -2.90410200e-02,  2.21277815e-02,  3.73357313e-0
2],
       [-8.58362342e-01,  1.19266782e-01, -6.85035477e-0
2, ...,
        -9.40606412e-02, -2.00893728e-02, -1.22649775e-0
2],
       [-9.26121767e-01,  1.30077482e-01, -2.12741152e-0
2, ...,
```

```
        -7.39955037e-02,  2.62567856e-02,  6.18471492e-0
3],
       ...,
       [-1.90870988e-03, -1.85425297e-01, -7.19344940e-0
2, ...,
        -3.30779335e-02, -2.29654550e-02, -2.62202483e-0
2],
       [-3.46503780e-01, -9.15969921e-02,  1.47611214e-0
2, ...,
         7.31217208e-03,  8.65902538e-03, -4.27363967e-0
4],
       [ 4.34222188e-02, -7.89139633e-02, -6.43711401e-0
2, ...,
        -6.05705106e-02,  1.24326542e-02, -1.82777411e-0
2]])
```

5 搜索较优DBSCAN参数（可使用PCA降维之后的数据。。。）

In [8]:

```
res = []
epss = np.arange(0.001,1,0.005)
min_sampless = np.arange(2,100)
for eps in epss:
    for min_samples in min_sampless:
        dbscan = cluster.DBSCAN(eps = eps, min_samples = m
        # 模型拟合
        yhat = dbscan.fit(df_norm)
        # 统计各参数组合下的聚类个数（-1表示异常点）
        n_clusters = len([i for i in set(dbscan.labels_) i
        # 异常点的个数
        outliners = np.sum(np.where(dbscan.labels_ == -1,
        # 统计每个簇的样本个数
        stats = str(pd.Series([i for i in dbscan.labels_ i
        labels = dbscan.labels_
        if n_clusters>=2:
            score1 = metrics.silhouette_score(df_norm, lab
            score2 = metrics.calinski_harabasz_score(df_no
            score3 = metrics.davies_bouldin_score(df_norm,
        else:
            score1 = 0
            score2 = 0
            score3 = 0


        #score1 = metrics.silhouette_score(pcadf, labels,
        #score2 = metrics.calinski_harabasz_score(pcadf, l
        #score3 = metrics.davies_bouldin_score(pcadf, labe
        res.append({'eps':eps,
                    'min_samples':min_samples,
                    'n_clusters':n_clusters,
                    'score1':score1,
                    'score2':score2,
                    'score3':score3,
                    'outliners':outliners,
```

```
                                 'stats':stats})
# 将迭代后的结果存储到数据框中
df = pd.DataFrame(res)
#df

df = df.loc[df.n_clusters>=3, :]
df
```

executed in 50m 38s, finished 09:20:56 2021-06-20

	eps	min_samples	n_clusters	score1	score2
686	0.036	2	7	-0.514028	1.193928
784	0.041	2	10	-0.511645	1.451374
882	0.046	2	20	-0.514148	1.844854
883	0.046	3	3	-0.320677	2.069744
980	0.051	2	40	-0.596561	2.204244
...	...	...	...	...	...
5586	0.286	2	3	0.246678	16.594284
5587	0.286	3	3	0.246678	16.594284
5592	0.286	8	3	0.227715	50.138176

	eps	min_samples	n_clusters	score1	score2
5691	0.291	9	3	0.224324	51.827233
6085	0.311	11	3	0.207267	44.502291

707 rows × 8 columns

In [9]:

```
a1 = df['score1'].max()
a2 = df['score2'].max()
a3 = df['score3'].max()
print('选用Silhouette系数来选择参数')
df.loc[df.score1 == a1, :]
print('选用Calinski-Harabaz指数来选择参数')
df.loc[df.score2 == a2, :]
print('选用Davies-Bouldin Index来选择参数')
df.loc[df.score3 == a3, :]
```

executed in 40ms, finished 09:20:56 2021-06-20

选用Silhouette系数来选择参数

	eps	min_samples	n_clusters	score1	score2
4707	0.241	5	3	0.270751	57.021157

选用Calinski-Harabaz指数来选择参数

	eps	min_samples	n_clusters	score1	score2
4175	0.211	61	3	0.219816	868.141805

选用Davies-Bouldin Index来选择参数

	eps	min_samples	n_clusters	score1	score2

	eps	min_samples	n_clusters	score1	score2
2457	0.126	9	5	-0.408679	62.65093

In [12]:

```
eps = [0.241,0.211,0.126]
min_samples = [5,61,9]
name = ['Silhouette系数','Calinski-Harabaz指数','Davies-Bo
for j in range(0,3):

    dbscan = cluster.DBSCAN(eps = eps[j], min_samples = mi
    # 模型拟合
    # dbscan.fit(df_norm)
    yhat = dbscan.fit_predict(df_norm)
    #new1查看各个类数量
    print('当选用'+name[j] +'的时候分类及其各个分类数量')
    print(np.unique(yhat,return_counts=True))
    if j==0:
        labels1 = dbscan.labels_
    elif j==1:
        labels2 = dbscan.labels_
    else:
        labels3 = dbscan.labels_
```

executed in 297ms, finished 09:22:05 2021-06-20

```
当选用Silhouette系数的时候分类及其各个分类数量
(array([-1,  0,  1,  2], dtype=int64), array([  59, 2027,
9,    4], dtype=int64))
当选用Calinski-Harabaz指数的时候分类及其各个分类数量
(array([-1,  0,  1,  2], dtype=int64), array([904, 673, 36
3, 159], dtype=int64))
当选用Davies-Bouldin Index的时候分类及其各个分类数量
(array([-1,  0,  1,  2,  3,  4], dtype=int64), array([116
0,  898,   11,    8,   13,    9], dtype=int64))
```

6 可视化结果图

In [13]:

```
plt.rcParams["font.sans-serif"] = ["SimHei"]
plt.figure(dpi=400, figsize=[30, 15])
plt.rcParams['axes.unicode_minus']=False
plt.scatter(hangtime, labels1)
plt.title("使用Silhouette系数分类关系图", fontsize=50)
plt.xlabel("时间", fontsize=50)
plt.ylabel("分类结果", fontsize=50)
plt.xticks(rotation=90)
plt.grid(True)
plt.xticks(pd.date_range('2005-01-01','2010-09-30',freq='1
plt.yticks(fontsize=50)
plt.show()

plt.rcParams["font.sans-serif"] = ["SimHei"]
plt.figure(dpi=400, figsize=[30, 15])
plt.rcParams['axes.unicode_minus']=False
plt.scatter(hangtime, labels2)
plt.title("使用Calinski-Harabaz指数分类关系图", fontsize=5
plt.xlabel("时间", fontsize=50)
plt.ylabel("分类结果", fontsize=50)
plt.xticks(rotation=90)
plt.grid(True)
plt.xticks(pd.date_range('2005-01-01','2010-09-30',freq='1
plt.yticks(fontsize=50)
plt.show()

plt.rcParams["font.sans-serif"] = ["SimHei"]
plt.figure(dpi=500, figsize=[40, 30])
plt.rcParams['axes.unicode_minus']=False
plt.scatter(hangtime, labels3)
plt.title("使用Davies-Bouldin Index分类关系图", fontsize=5
plt.xlabel("时间", fontsize=50)
plt.ylabel("分类结果", fontsize=50)
plt.xticks(rotation=90)
```

```
plt.grid(True)
plt.xticks(pd.date_range('2005-01-01','2010-09-30',freq='1
plt.yticks(fontsize=50)
plt.show()
```

executed in 27.1s, finished 09:22:35 2021-06-20

```
<Figure size 12000x6000 with 0 Axes>

<matplotlib.collections.PathCollection at 0x1c09a7acac0>

Text(0.5, 1.0, '使用Silhouette系数分类关系图')

Text(0.5, 0, '时间')

Text(0, 0.5, '分类结果')
```

In [ ]:

In [ ]: