问题3

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

import plotly
import plotly.express as px
import plotly.graph_objects as go
import plotly.figure_factory as ff
plotly.offline.init_notebook_mode()

import scipy
import scipy.cluster.hierarchy as sch

from sklearn.metrics import *
from sklearn.ensemble import IsolationForest

import chart_studio
import chart_studio.plotly as py
from chart_studio.plotly import plot, iplot
chart_studio.tools.set_credentials_file(username='zhiliao0824', api_key='qrGJtJRojwpsVJ3nosn0')

import matplotlib.pyplot as plt
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

from IPython.display import HTML
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = 'all'
InteractiveShell.ast_node_interactivity = 'last'

import pylatex
import latexify
```

In [2]:
```
writer_improve = pd.ExcelWriter('模型改进数据.xlsx')
```

处理数据-得到改进后的数据

漏水量分布

In [3]:
```
def leaking_distribution(x):
    if x <= 25:
        return x
    elif x <= 30:
        return x - 25 + 0.7 * x
    else:
        return x - 25 + 0.5 * x
```

In [4]:
```
InteractiveShell.ast_node_interactivity = 'all'

DMA_leaking = pd.read_excel('./按照日期处理后的数据.xlsx', sheet_name='DMA1和DMA2的漏水量', index_col=0)
DMA_leaking.head()
DMA_leaking = DMA_leaking.applymap(leaking_distribution)
DMA_leaking.head()

DMA_leaking.to_excel(writer_improve, 'DMA1和DMA2的漏水量')
```

用水量数据

In [5]:
```
columns_name = ["当地时间(北京时间)", "DMA1", "DMA2"]
path = 'B1题附件.xls'
data = pd.read_excel(path)
data = pd.DataFrame(data.values, columns=columns_name)
data_time = data.set_index("当地时间(北京时间)")
data_time.head()
```

Out[5]:

当地时间(北京时间)	DMA1	DMA2
2014-04-15 00:00:00	48.89	36.98
2014-04-15 00:15:00	37.78	29.88
2014-04-15 00:30:00	34.44	29.04
2014-04-15 00:45:00	33.33	29.96
2014-04-15 01:00:00	33.33	30.96

In [6]:
```
InteractiveShell.ast_node_interactivity = 'all'

data_time1 = pd.DataFrame(data_time.iloc[:, 0])
```

```
data_time2 = pd.DataFrame(data_time.iloc[:, 1])
# data_time1.head()
# data_time2.head()

data_time_YHD = data_time.copy()
data_time_YHD.index = data_time_YHD.index.strftime("%Y-%m-%d")
# data_time_YHD.head()

DMA_user = data_time_YHD - DMA_leaking
DMA_user.index = list(data_time.index)
DMA_user1 = pd.DataFrame(DMA_user.iloc[:, 0])
DMA_user2 = pd.DataFrame(DMA_user.iloc[:, 1])
DMA_user1[DMA_user1 < 0] = 0
DMA_user2[DMA_user2 < 0] = 0
DMA_user1.head()
DMA_user2.head()
```

Out[6]:

	DMA1
2014-04-15 00:00:00	25.56
2014-04-15 00:15:00	14.45
2014-04-15 00:30:00	11.11
2014-04-15 00:45:00	10
2014-04-15 01:00:00	10

Out[6]:

	DMA2
2014-04-15 00:00:00	11.775
2014-04-15 00:15:00	4.675
2014-04-15 00:30:00	3.835
2014-04-15 00:45:00	4.755
2014-04-15 01:00:00	5.755

In [7]:

```
# 区域1用户用水量
tmp = pd.DataFrame()
tmp_index = None
for i in range(24):
    for j in range(0, 60, 15):
        tmp_data =  DMA_user1.between_time(f"{i}:{j}:00", f"{i}:{j}:00").iloc[:, 0]
        if i == 0 and j == 0:
```

```
            tmp_index = list(tmp_data.index)
        try:
            tmp_data.index = list(tmp_index)
        except ValueError as e:
            tmp_data.index = list(tmp_index)[:-1]
        tmp[f"{i}:{j}:00"] = tmp_data
tmp[:-1].to_excel(writer_improve, 'DMA1的用户用水量')
tmp[:-1].head()
```

Out[7]:

	0:0:00	0:15:00	0:30:00	0:45:00	1:0:00	1:15:00	1:30:00	1:45:00	2:0:00	2:15:00	...	21:30:00	21:45:00	22:0:00	22:15:00	22:30:00	22:45:00	23:0:00
2014-04-15	25.56	14.45	11.11	10	10	8.89	8.89	10	8.89	11.11	...	37.78	37.78	36.67	41.11	44.45	45.56	43.34
2014-04-16	26.22	17.33	16.22	15.11	12.89	11.78	9.56	9.56	5.11	5.11	...	35.11	36.22	39.56	42.89	40.67	40.67	36.22
2014-04-17	27.78	20	15.56	15.56	15.56	21.11	20	15.56	15.56	10	...	44.45	44.45	46.67	45.56	46.67	43.34	43.34
2014-04-18	24.777	22.557	15.887	14.777	12.557	12.557	12.557	6.997	6.997	4.777	...	43.667	44.777	45.887	44.777	42.557	43.667	48.107
2014-04-19	31.665	28.335	25.005	18.335	18.335	16.115	10.555	8.335	10.555	9.445	...	36.115	38.335	40.555	46.115	50.555	51.665	52.775

5 rows × 96 columns

In [8]:
```
# 区域2用户用水量
tmp = pd.DataFrame()
tmp_index = None
for i in range(24):
    for j in range(0, 60, 15):
        tmp_data =  DMA_user2.between_time(f"{i}:{j}:00", f"{i}:{j}:00").iloc[:, 0]
        if i == 0 and j == 0:
            tmp_index = list(tmp_data.index)
        try:
            tmp_data.index = list(tmp_index)
        except ValueError as e:
            tmp_data.index = list(tmp_index)[:-1]
        tmp[f"{i}:{j}:00"] = tmp_data
tmp[:-1].to_excel(writer_improve, 'DMA2的用户用水量')
tmp[:-1].head()
```

Out[8]:

	0:0:00	0:15:00	0:30:00	0:45:00	1:0:00	1:15:00	1:30:00	1:45:00	2:0:00	2:15:00	...	21:30:00	21:45:00	22:0:00	22:15:00	22:30:00	22:45:00	23:0:00
2014-04-15	11.775	4.675	3.835	4.755	5.755	6.835	7.775	8.615	9.305	9.065	...	6.555	5.985	8.315	8.325	7.975	8.805	9.725
2014-04-16	9.99	5.71	6.37	7.92	9.04	10.39	11.12	12.15	8.74	8.31	...	9.35	9.7	9.35	8.82	9.01	9.33	7.91
2014-04-17	12.63	11.14	7.96	10.23	11.23	12.11	12.8	11.73	8.93	9.73	...	8.59	10.59	10.11	9.22	9.45	10.25	10.5
2014-04-18	13.015	11.975	8.135	9.705	10.165	11.425	12.845	9.555	9.115	9.725	...	11.805	11.825	11.275	10.365	10.635	10.935	11.415
2014-04-19	11.205	10.145	10.875	9.545	11.185	12.225	9.405	8.445	9.315	9.945	...	11.535	11.785	10.855	9.925	9.945	10.055	10.945

5 rows × 96 columns

In [9]:
```
writer_improve.save()
```

孤立森林

在 DMA1 的 2 类，DMA2 的 4 类中找到数量最多的那 2 类，分别做孤立森林找异常值

DMA1 的 2 类都要做孤立森林，DMA2 的 4 个类中的 2 类做孤立森林

In [10]:
```
writer_im_q3 = pd.ExcelWriter('问题3数据.xlsx')  # 对比后面的改进，取名为“问题3-模型-孤立森林数据”更好一点
```

In [11]:
```
path = './模型改进数据.xlsx'
sheet = 'DMA1的用户用水量'
DMA1_data = pd.read_excel(path, sheet_name=sheet, index_col=0)
DMA1_data.index = DMA1_data.index.strftime("%m-%d")
DMA1_data.head()
```

Out[11]:

	0:0:00	0:15:00	0:30:00	0:45:00	1:0:00	1:15:00	1:30:00	1:45:00	2:0:00	2:15:00	...	21:30:00	21:45:00	22:0:00	22:15:00	22:30:00	22:45:00	23:0:00	23
04-15	25.560	14.450	11.110	10.000	10.000	8.890	8.890	10.000	8.890	11.110	...	37.780	37.780	36.670	41.110	44.450	45.560	43.340	
04-16	26.220	17.330	16.220	15.110	12.890	11.780	9.560	9.560	5.110	5.110	...	35.110	36.220	39.560	42.890	40.670	40.670	36.220	
04-17	27.780	20.000	15.560	15.560	15.560	21.110	20.000	15.560	15.560	10.000	...	44.450	44.450	46.670	45.560	46.670	43.340	43.340	
04-18	24.777	22.557	15.887	14.777	12.557	12.557	12.557	6.997	6.997	4.777	...	43.667	44.777	45.887	44.777	42.557	43.667	48.107	
04-19	31.665	28.335	25.005	18.335	18.335	16.115	10.555	8.335	10.555	9.445	...	36.115	38.335	40.555	46.115	50.555	51.665	52.775	

5 rows × 96 columns

In [12]:
```
DMA1_class1_April = [20, 27, 21, 23, 15, 24, 22, 26, 16, 17, 25, 29, 19, 28]
DMA1_class1_May = [1]
DMA1_class1 = [f'04-{i}' for i in DMA1_class1_April] + [f'05-{i}' for i in DMA1_class1_May]

DMA1_data_class1_mask = DMA1_data.apply(lambda x: x.name in DMA1_class1, axis=1)
DMA1_data_class1 = DMA1_data[DMA1_data_class1_mask]
DMA1_data_class1.to_excel(writer_im_q3, 'DMA1-class1-all')

rng = np.random.RandomState(24)
n, m = DMA1_data_class1.shape

xtrain = DMA1_data_class1
clf = IsolationForest(n_estimators=n, random_state=rng)
clf.fit(xtrain)
ypred = clf.predict(xtrain)
# print(ypred)
# print(ypred < 0)
DMA1_data_class1[ypred > 0].to_excel(writer_im_q3, 'DMA1-class1-normal')
DMA1_data_class1[ypred < 0].to_excel(writer_im_q3, 'DMA1-class1-abnormal')
```

Out[12]:
```
IsolationForest(n_estimators=14,
                random_state=RandomState(MT19937) at 0x2AC5D7A8990)
```

In [13]:
```
DMA1_class2 = ['05-25', '05-27']
DMA1_class3 = ['05-21', '06-06']
DMA1_class4_ = DMA1_class1 + DMA1_class2 + DMA1_class3 + ['05-28']
```

```
DMA1_data_class4_mask = DMA1_data.apply(lambda x: x.name not in DMA1_class4_ , axis=1)
DMA1_data_class4 = DMA1_data[DMA1_data_class4_mask]
DMA1_data_class4.to_excel(writer_im_q3, 'DMA1-class4-all')

rng = np.random.RandomState(24)
n, m = DMA1_data_class4.shape

xtrain = DMA1_data_class4
clf = IsolationForest(n_estimators=n, random_state=rng)
clf.fit(xtrain)
ypred = clf.predict(xtrain)
# print(ypred)
# print(ypred < 0)
DMA1_data_class4[ypred > 0].to_excel(writer_im_q3, 'DMA1-class4-normal')
DMA1_data_class4[ypred < 0].to_excel(writer_im_q3, 'DMA1-class4-abnormal')
```

Out[13]:
```
IsolationForest(n_estimators=39,
                random_state=RandomState(MT19937) at 0x2AC5D7A8780)
```

In [ ]:

In [14]:
```
path = './模型改进数据.xlsx'
sheet = 'DMA2的用户用水量'
DMA2_data = pd.read_excel(path, sheet_name=sheet, index_col=0)
DMA2_data.index = DMA2_data.index.strftime("%m-%d")
DMA2_data.head()
```

Out[14]:

	0:0:00	0:15:00	0:30:00	0:45:00	1:0:00	1:15:00	1:30:00	1:45:00	2:0:00	2:15:00	...	21:30:00	21:45:00	22:0:00	22:15:00	22:30:00	22:45:00	23:0:00	23
04-15	11.775	4.675	3.835	4.755	5.755	6.835	7.775	8.615	9.305	9.065	...	6.555	5.985	8.315	8.325	7.975	8.805	9.725	
04-16	9.990	5.710	6.370	7.920	9.040	10.390	11.120	12.150	8.740	8.310	...	9.350	9.700	9.350	8.820	9.010	9.330	7.910	
04-17	12.630	11.140	7.960	10.230	11.230	12.110	12.800	11.730	8.930	9.730	...	8.590	10.590	10.110	9.220	9.450	10.250	10.500	
04-18	13.015	11.975	8.135	9.705	10.165	11.425	12.845	9.555	9.115	9.725	...	11.805	11.825	11.275	10.365	10.635	10.935	11.415	
04-19	11.205	10.145	10.875	9.545	11.185	12.225	9.405	8.445	9.315	9.945	...	11.535	11.785	10.855	9.925	9.945	10.055	10.945	

5 rows × 96 columns

In [15]:
```
DMA2_class1_April = [18, 22, 27, 17, 19, 23, 24, 28, 20, 21, 25, 26, 29, 15, 16]
DMA2_class1 = [f'04-{i}' for i in DMA2_class1_April]

DMA2_data_class1_mask = DMA2_data.apply(lambda x: x.name in DMA2_class1, axis=1)
DMA2_data_class1 = DMA2_data[DMA1_data_class1_mask]
DMA2_data_class1.to_excel(writer_im_q3, 'DMA2-class1-all')

rng = np.random.RandomState(24)
n, m = DMA2_data_class1.shape

xtrain = DMA2_data_class1
clf = IsolationForest(n_estimators=n, random_state=rng)
clf.fit(xtrain)
ypred = clf.predict(xtrain)
# print(ypred)
# print(ypred < 0)
DMA2_data_class1[ypred > 0].to_excel(writer_im_q3, 'DMA2-class1-normal')
DMA2_data_class1[ypred < 0].to_excel(writer_im_q3, 'DMA2-class1-abnormal')
```

Out[15]:
```
IsolationForest(n_estimators=14,
                random_state=RandomState(MT19937) at 0x2AC5D7A8BA0)
```

In [16]:
```
# DMA2_class2  # auto get
DMA2_class2_ = DMA2_class1 + ['05-28']

DMA2_data_class2_mask = DMA2_data.apply(lambda x: x.name not in DMA2_class2_ , axis=1)
DMA2_data_class2 = DMA2_data[DMA2_data_class2_mask]
DMA2_data_class2.to_excel(writer_im_q3, 'DMA2-class2-all')

rng = np.random.RandomState(24)
n, m = DMA2_data_class2.shape
xtrain = DMA2_data_class2
clf = IsolationForest(n_estimators=n, random_state=rng)
clf.fit(xtrain)
ypred = clf.predict(xtrain)
# print(ypred)
# print(ypred < 0)
DMA2_data_class2[ypred > 0].to_excel(writer_im_q3, 'DMA2-class2-normal')
DMA2_data_class2[ypred < 0].to_excel(writer_im_q3, 'DMA2-class2-abnormal')
```

Out[16]:
```
IsolationForest(n_estimators=42,
                random_state=RandomState(MT19937) at 0x2AC5D7A8A98)
```

In [17]:
```
writer_im_q3.save()
```

In [ ]: