K-Means 异常检测程序报告

学号：2111460　　姓名：张洋

一、问题重述

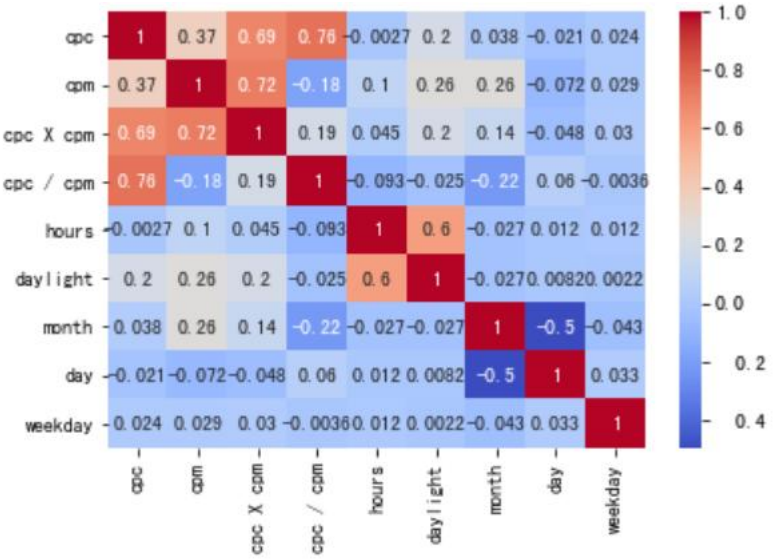
(1)了解 KMeans、PCA 算法，了解算法的基本原理

(2)运用 KMeans 算法完成异常点检测，调整模型的参数使模型调整到最佳状态，尝试构造更多特征使模型获得更强的能力。完成 preprocess_data、get_distance、get_anomaly 函数的相关改进，调整超参数，完成训练。

二、设计思想

1、选取特征

首先我选取了一些额外的特征绘制出热力图：



经过测试，我选择了四个特征进行分析：cpc, cpm, hours, daylight。虽然 hours 特征的信息包括了 daylight 特征的信息，但是能得到比单独使用更好的结果。猜测是由于 hours 特征和 daylight 特征均与成本有关，并且 daylight 特征与成本更相关，所以需要使用这两个特征。

以上各个特征之间并不是相互独立的，存在一定程度上的相关性；故采用 PCA 来寻找数据集的低维度表达。PCA 通过对数据特征的变换，寻找特征空间中，数据分布方差最大的方向，称为特征方向或主成分方向，选择其中特征值较大的几个特征方向，将数据点投影到这些方向上，完成数据降维。

2、标准化

使用 StandardScaler 进行标准化，去除量纲的影响。

因为多个特征分布的区间并不相同，因此在寻找异常点之前进行数据标准化，使用 sklearn 中的标准化函数，数据标准化后分布在（-1，1）之间。

标准化公式：

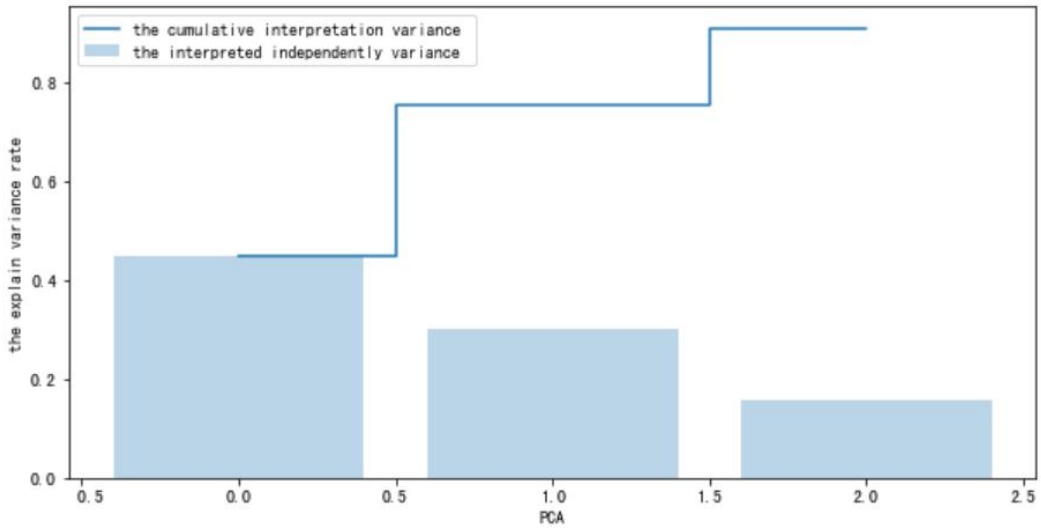
$$x^* = \frac{x - \mu}{\sigma}$$

其中 μ 为所有样本数据的均值，σ 为所有样本数据的标准差。

3、主成分分析

使用 PCA 进行主成分分析，根据方差贡献率超过 90%的标准，选择将数据降维成 3 维。得出每个保留特征占所有特征的方差百分比

array([0.45072317, 0.30293587, 0.15629988])



4、聚类

通过 KMeans 进行聚类，根据轮廓系数指标，选择将数据聚类成 5 类。

calinski_harabasz_score：通过计算簇中各点与簇中心的距离平方和来度量簇内的紧密度，通过计算各簇中心点与数据集中心点距离平方和来度量数据集的分离度，由分离度与紧密度的比值得到。即该指标越大代表着簇自身越紧密，簇与簇之间越分散，即聚类结果越好。

silhouette_score(轮廓系数)结合了凝聚度和分离度，轮廓系数取值为 [-1, 1]，其值越大越好，且当值为负时，表明 ai < bi，样本被分配到错误的簇中，聚类结果不可接受。对于接近 0 的结果，则表明聚类结果有重叠的情况。

```
聚类数目:2  calinski_harabasz_score:981.89      silhouette_score:0.56
聚类数目:3  calinski_harabasz_score:1090.07     silhouette_score:0.6
聚类数目:4  calinski_harabasz_score:1205.04     silhouette_score:0.56
聚类数目:5  calinski_harabasz_score:1557.87     silhouette_score:0.53
聚类数目:6  calinski_harabasz_score:1590.42     silhouette_score:0.48
聚类数目:7  calinski_harabasz_score:1708.36     silhouette_score:0.5
聚类数目:8  calinski_harabasz_score:1682.88     silhouette_score:0.51
聚类数目:9  calinski_harabasz_score:1660.73     silhouette_score:0.47
```

根据如上分析加上多次尝试，选取聚类数目为 5。

5、异常检测

根据指定的 ratio，选择距离各自聚类中心距离最大的 num_anomaly 个数据作为异常值，根据阈值距离大小判断每个点是否是异常值。

三、代码内容

1、模型生成与保存

该函数生成了 StandardScaler 、KMeans、PCA 三个模型的实例并保存。根据之后的计算方差解释率和最佳聚类数的代码，主成分数为 3，聚类数为 5，其他参数如下：

```
scaler = StandardScaler()
pca = PCA(n_components=3) #PCA
kmeans = KMeans(n_clusters=5,init='k-means++',n_init=50,max_iter=800)

joblib.dump(kmeans, './results/model.pkl')
joblib.dump(scaler, './results/scaler.pkl')
joblib.dump(pca, './results/pca.pkl')
```

2、数据预处理

在数据预处理中，首先生成了 hours, daylight, cpcXcpm, cpc/cpm,weekday,day,month 等特征，然后选取了 cpc, cpm, hours, daylight 四个特征进行了标准化和主成分分析，用于后续的异常检测。除此之外，还给出了计算累计贡献率和以及最佳聚类数的代码。

```
import os
import sklearn
import numpy as np
import pandas as pd
from copy import deepcopy
from sklearn.cluster import KMeans
from sklearn.externals import joblib


def preprocess_data(df):
    """
    数据处理及特征工程等
    :param df: 读取原始 csv 数据，有 timestamp、cpc、cpm 共 3 列特征
    :return: 处理后的数据，返回 pca 降维后的特征
    """
    # 请使用 joblib 函数加载自己训练的 scaler、pca 模型，方便在测试时系统对数据进行相同的变换
    # ====================数据预处理、构造特征等=======================

    df['timestamp'] = pd.to_datetime(df['timestamp'])
    df = df.sort_values(by='timestamp').reset_index(drop=True)
    df['cpc X cpm'] = df['cpm'] * df['cpc']
    df['cpc / cpm'] = df['cpc'] / df['cpm']
    df['hours'] = df['timestamp'].dt.hour
    df['month'] = df['timestamp'].dt.month
    df['day'] = df['timestamp'].dt.day
    df['weekday'] = df['timestamp'].dt.weekday
    df['daylight'] = ((df['hours'] >= 7) & (df['hours'] <=
```

```
22)).astype(int)

    # ========================  模型加载  ============================

    columns = ['cpc','cpm','hours','daylight']
    data = df[columns]
    scaler = joblib.load('./results/scaler.pkl')
    pca = joblib.load('./results/pca.pkl')
    data = scaler.fit_transform(data)
    data = pd.DataFrame(data, columns=columns)
    data = pca.fit_transform(data)
    data = pd.DataFrame(data,columns=['Dimension' + str(i+1) for i in
range(len(data[0]))])

    return data
```

累计贡献率：

```
var_explain = pca.explained_variance_ratio_
# 梯形累计和，axis=0，按照行累加。axis=1，按照列累加。axis 不给定具体值，就
把数组当成一个一维数组
cum_var_explian = np.cumsum(var_explain)

plt.figure(figsize=(10, 5))
plt.bar(range(len(var_explain)), var_explain, alpha=0.3,
align='center', label='the interpreted independently variance ')
plt.step(range(len(cum_var_explian)), cum_var_explian, where='mid',
label='the cumulative interpretation variance')
plt.ylabel('the explain variance rate')
plt.xlabel('PCA')
plt.legend(loc='best')
plt.show()
```

计算最佳聚类数：

```
# 寻找最佳聚类数目
from sklearn.cluster import KMeans
from sklearn.metrics import calinski_harabasz_score,silhouette_score

'''
n_clusters 指定了需要聚类的个数，这个超参数需要自己调整，会影响聚类的效果
init 指明初始聚类中心点的初始化方式，kmeans++是一种初始化方式，还可以选择为
random
n_init 指定计算次数，算法并不会运行一遍后就返回结果，而是运行多次后返回最好的
一次结果，n_init 即指明运行的次数
```

```
max_iter 指定单次运行中最大的迭代次数，超过当前迭代次数即停止运行
'''
score1_list = []
score2_list = []
for i in range(2,10):
    kmeans =
KMeans(n_clusters=i,init='k-means++',n_init=10,max_iter=100)
    kmeans.fit(data)
    score1 = round(calinski_harabasz_score(data,kmeans.labels_), 2)
    score2 = round(silhouette_score(data,kmeans.labels_), 2)
    score1_list.append(score1)
    score2_list.append(score2)
    print('聚类数目:%s  calinski_harabasz_score:%-10s
silhouette_score:%-10s'%(i,score1,score2))
```

3、计算距离

该函数计算了每个点到对应聚类中心的距离，并存储在一个列表中返回。

```
def get_distance(data, kmeans, n_features):
    """
    计算样本点与聚类中心的距离
    :param data: preprocess_data 函数返回值，即 pca 降维后的数据
    :param kmeans: 通过 joblib 加载的模型对象，或者训练好的 kmeans 模型
    :param n_features: 计算距离需要的特征的数量
    :return:每个点距离自己簇中心的距离，Series 类型
    """
    # ====================计算样本点与聚类中心的距离
========================
    distance = []
    for i in range(0,len(data)):
        point = np.array(data.iloc[i,:n_features])
        center = kmeans.cluster_centers_[kmeans.labels_[i],:n_features]
        distance.append(np.linalg.norm(point - center))
    distance = pd.Series(distance)
    return distance
```

4、异常检测

该函数根据输入的 ratio 计算异常数目 num_anomaly，然后得到每个点到对应聚类中心的距离，排序后选取距离最大的 num_anomaly 个数据作为异常数据返回。

```
def get_anomaly(data, kmean, ratio):
    """
    检验出样本中的异常点，并标记为 True 和 False，True 表示是异常点

    :param data: preprocess_data 函数返回值,即 pca 降维后的数据,DataFrame 类
型
    :param kmean: 通过 joblib 加载的模型对象，或者训练好的 kmeans 模型
```

```
    :param ratio: 异常数据占全部数据的百分比,在 0 - 1 之间，float 类型
    :return: data 添加 is_anomaly 列，该列数据是根据阈值距离大小判断每个点是否是异常值，元素值为 False 和 True
    """
    # ====================检验出样本中的异常点=======================
    num_anomaly = int(len(data) * ratio)
    new_data = deepcopy(data)
    new_data['distance'] =
get_distance(new_data,kmean,n_features=len(new_data.columns))
    threshould =
new_data['distance'].sort_values(ascending=False).reset_index(drop=True
)[num_anomaly]
    new_data['is_anomaly'] = new_data['distance'].apply(lambda x: x >
threshould)
    normal = new_data[new_data['is_anomaly'] == 0]
    anormal = new_data[new_data['is_anomaly'] == 1]

    return new_data
```

对检测结果可视化代码：

```
fig = plt.figure(1, figsize=(6, 6))
ax = Axes3D(fig)
ax.scatter(anormal.iloc[:, 0], anormal.iloc[:, 1], anormal.iloc[:, 2],
c='red', edgecolors='k')
ax.scatter(normal.iloc[:, 0], normal.iloc[:, 1], normal.iloc[:, 2],
c='blue', edgecolors='k');

a = df.loc[new_data['is_anomaly'] == 1, ['timestamp', 'cpc']]
plt.figure(figsize=(20,6))
plt.plot(df['timestamp'], df['cpc'], color='blue')
# 聚类后 cpc 的异常点
plt.scatter(a['timestamp'],a['cpc'], color='red')
plt.show()

a = df.loc[new_data['is_anomaly'] == 1, ['timestamp', 'cpm']]
plt.figure(figsize=(20,6))
plt.plot(df['timestamp'], df['cpm'], color='blue')
# 聚类后 cpm 的异常点
plt.scatter(a['timestamp'],a['cpm'], color='red')
plt.show()
```

5、预测

该函数首先将异常比例设为 0.03，然后加载 kmeans 模型并 fit，最后得到异常值并返回。

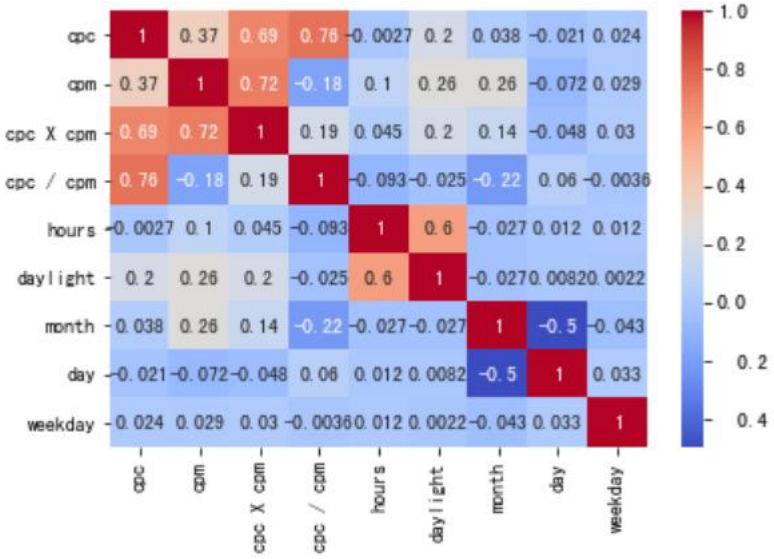
```
def predict(preprocess_data):
```

```
    """
    该函数将被用于测试，请不要修改函数的输入输出，并按照自己的模型返回相关的数据。
    在函数内部加载 kmeans 模型并使用 get_anomaly 得到每个样本点异常值的判断
    :param preprocess_data: preprocess_data 函数的返回值，一般是 DataFrame 类型
    :return:is_anomaly:get_anomaly 函数的返回值，各个属性应该为（Dimesion1,Dimension2,.....数量取决于具体的 pca），distance,is_anomaly，请确保这些列存在
            preprocess_data:  即直接返回输入的数据
            kmeans: 通过 joblib 加载的对象
            ratio:  异常点的比例，ratio <= 0.03    返回非异常点得分将受到惩罚！
    """
    # 异常值所占比率
    ratio = 0.03
    # 加载模型
    kmeans = joblib.load('./results/model.pkl')
    kmeans.fit(preprocess_data)
    # 获取异常点数据信息
    is_anomaly = get_anomaly(preprocess_data, kmeans, ratio)

    return is_anomaly, preprocess_data, kmeans, ratio
```

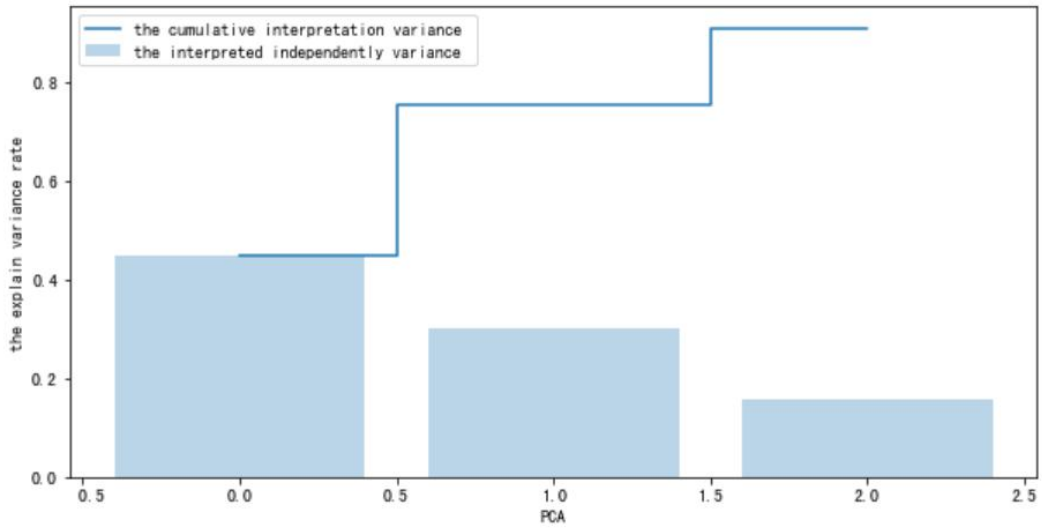
四、实验结果

1、相关性分析



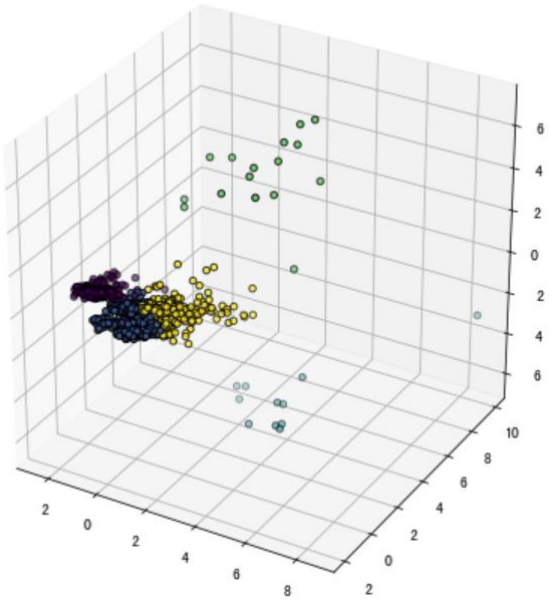
由上图可知，daylight 与 cpc 和 cpm 具有弱相关关系。虽然 hours 与 cpc、cpm 的皮尔逊相关系数较小，但是由于皮尔逊相关系数只能刻画线性相关关系，这不能代表 hours 与 cpc、cpm 的相关性弱。相比较于 hours 和 daylight，其他特征与 cpc、cpm 关系更弱，因此选择 cpc、cpm、hours、daylight 四个特征。

2. 累计贡献率



由上图可知，将特征降维为 3 个主成分可以解释原始数据 91%的差异，效果较好。

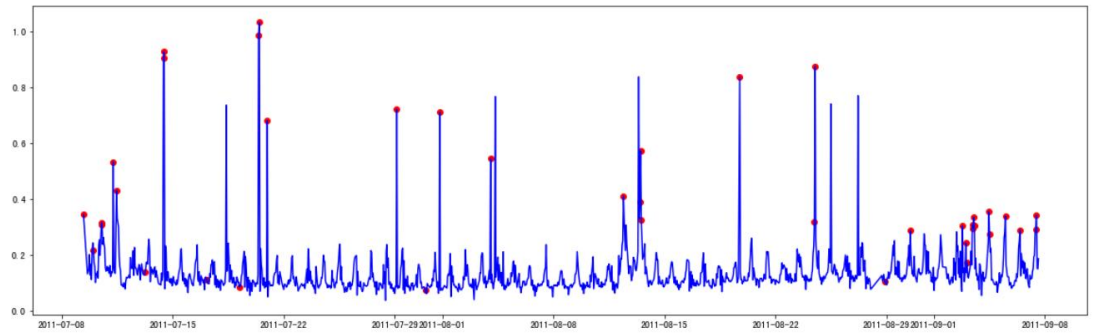
3、检测结果可视化（三维）

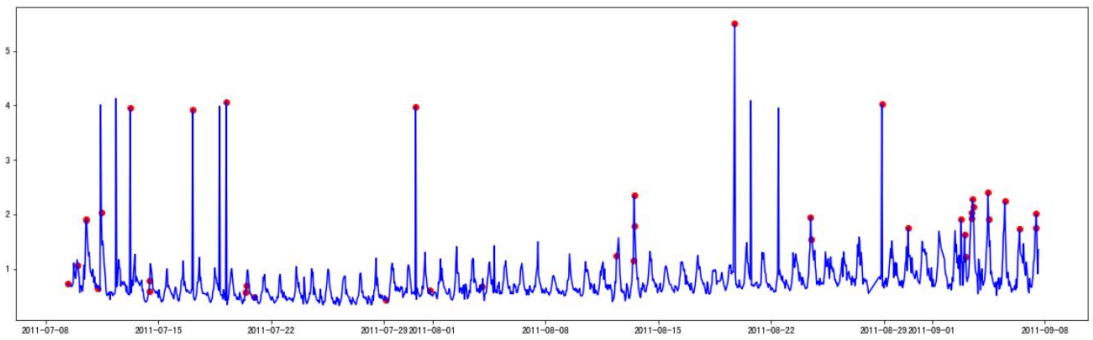


由图可知，本实验应该有五个聚类中心，本模型较好地完成了分类。

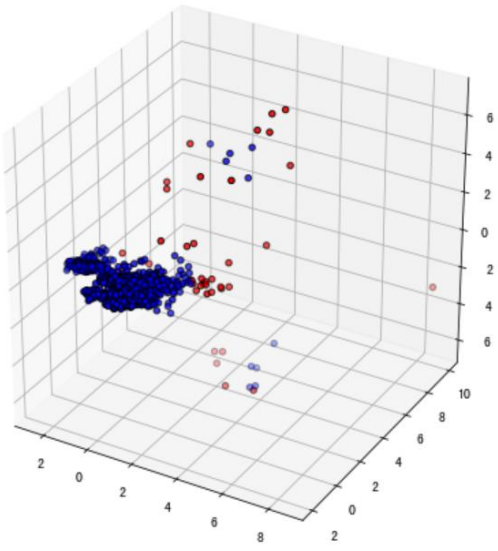
4、检测结果可视化（二维）

下图将异常检测的结果在 cpc、cpm 随时间变化的折线图中标出，以进一步说明检测结果的准确性。





我们可以看到，本模型将那些远离平均值的数据均标记为了异常，再次说明模型的优异性。值得一提的是，在两张图中也有很多红点标记在了明显不是异常值的位置，这并不是模型的判断失误，而是因为该数据的另一个属性出现了异常。具体来说，如果某个数据只有 cpc 是异常的，那么在 cpm 的图上就表现出将正常值标记为了异常的假象，然而实际上该数据确实是异常的。所以我们需要将两个图联合起来判断检测的准确性。



最终检测出四个异常值：

测试点	状态	时长	结果
测试结果	✓	0s	通过测试，检测出异常点4个，异常点总数为5个

五、总结

本实验主要是对广告投放数据进行特征选择、标准化、主成分分析、聚类和异常检测。实验达到了预期的目标，并且得到了较好的聚类效果和异常检测效果。但是在实验中也遇到了一些困难，如选择合适的聚类数目、设置合理的异常检测参数等，提醒我们在实验过程中需要注意细节问题。

可能的改进方向：特征工程方面，可以增加更多的特征，对不同的特征组合进行尝试，可能会得到更好的聚类效果；异常检测方面，可以尝试其他的异常检测算法，如 LOF、HBOS 等，对比不同算法的表现；模型评估方面，可以使用更多的指标来评估聚类和异常检测的效果，比如 ARI、NMI 等指标；数据处理方面，可以进行数据清洗和数据采样，消除异常值和不平衡样本对结果的影响。

如何提升性能：数据预处理方面，对数据进行预处理可以提高模型的鲁棒性和泛化性；超参数搜索方面，可以使用网格搜索、贝叶斯优化等方法寻找最优的超参数组合；模型选择方面，可以使用更先进的模型，如深度学习模型、集成学习模型等。

总之，通过此次实验我学会了主成分分析和聚类的方法，对特征处理和模型参数的选择更加熟练了。