**程序报告**

学号： 2113662 姓名：张丛

1. **问题重述**

====================================================================

**实验背景：**

异常值检测（outlier detection ）是一种数据挖掘过程，用于发现数据集中的异常值并确定异常值的详细信息。  
 当前数据容量大、数据类型多样、获取数据速度快；但是数据也比较复杂，数据的质量有待商榷；而数据容量大意味着手动标记异常值成本高、效率低下；因此能够自动检测异常值至关重要。  
 自动异常检测具有广泛的应用，例如信用卡欺诈检测、系统健康监测、故障检测以及传感器网络中的事件检测系统等。

### **实验要求：**

### (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

1. **设计思想**

====================================================================

Means是一种聚类算法，它可以将数据分成不同的簇，通过最小化每个簇的样本与簇中心之间的距离，使得同一簇内的数据点相似度最高，不同簇之间的相似度最小。

而异常检测的目的是发现那些与其他数据点不同的数据点。K-Means可以用于异常检测，实现思想是将那些被分到离群簇中的数据看作是异常。

K-Means算法步骤：

1. 随机选取K个簇心。

2. 对于每个数据点，计算它们与每个簇心的距离，并将其指派给距离最近的簇。

3. 对于每个簇，重新计算簇心位置，即将该簇内的所有数据点对应轴上取平均值。

4. 重复步骤2和3，直到簇心位置不再改变或达到预设的最大轮次。

5. 得到每个数据点所属的簇，形成K个簇。

6. 计算每个簇内数据点与簇心之间的距离的平均值，作为该簇的代价函数值。

K-Means检测异常的实验步骤：

1. 收集数据集，即给定的cpm、cpc数据

2. 对数据集进行预处理。

3. 特征提取：从数据中提取可以反映数据特征的指标或特征，如数据类型、大小、密度和变异性等。实验中，还进行了特征构造和PCA算法。

4. 选择K值：在K-Means算法中，需要指定聚类的数量K。

5. 运行K-Means：使用选择的K值运行K-Means聚类算法。

6. 检测异常：将数据点标记为异常值，如果它们属于非常小的簇或者被指派到离群簇。

7. 绘制结果：将检测到的异常值绘制在图表上，以便更好地展示和理解检测结果。

1. **代码内容**

**三个主要代码：**

====================================================================

import os

import sklearn

import numpy as np

import pandas as pd

import random

from copy import deepcopy

import joblib

from sklearn.decomposition import PCA

from sklearn.preprocessing import StandardScaler

def preprocess\_data(df):

columns = ['cpc', 'cpm', 'cpc X cpm', 'cpc / cpm']

data = df[columns]

scaler = StandardScaler()

data = scaler.fit\_transform(data)

data = pd.DataFrame(data, columns=columns)

#通过 n\_components 指定需要降低到的维度

n\_components = 1

pca = PCA(n\_components=n\_components)

data = pca.fit\_transform(data)

data = pd.DataFrame(data,columns=['Dimension' + str(i+1) for i in range(n\_components)])

return data

====================================================================

def get\_distance(data, kmeans, n\_features):

# ====================

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

====================================================================

def get\_anomaly(data, kmean, ratio):

# ====================

检验出样本中的异常点

========================

# 求阈值大小

ratio = 0.03

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]

print('阈值距离：'+str(threshould))

# 根据阈值距离大小判断每个点是否是异常值

new\_data['is\_anomaly'] = new\_data['distance'].apply(lambda x: x > threshould)

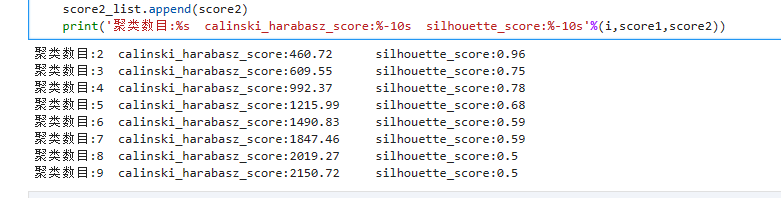
data['distance'] = new\_data['distance']

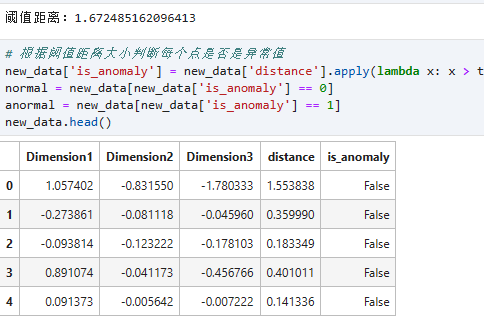
data['is\_anomaly'] = new\_data['is\_anomaly']

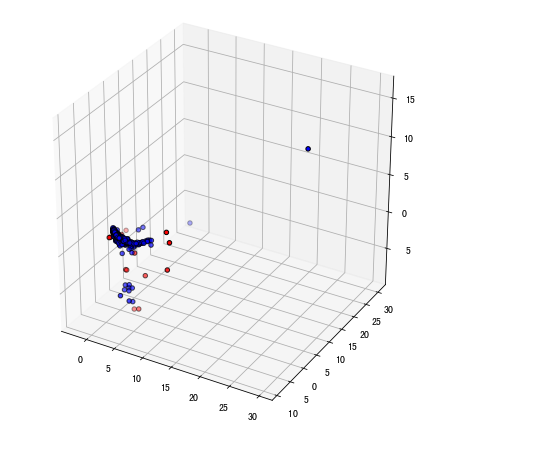
return data

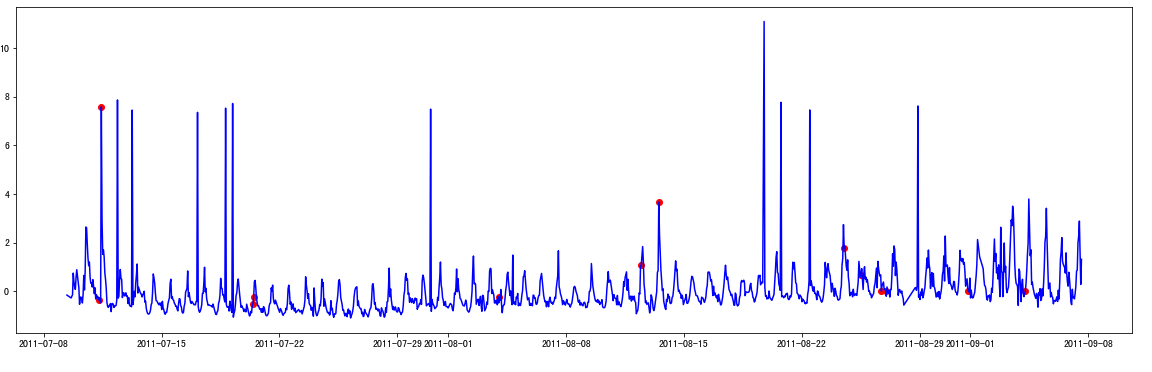
1. **实验结果**

====================================================================











1. **总结**

====================================================================

此次实验学习了K-Means算法及对于异常检测的应用。

优化了实验提供的模型算法，使得异常检测的准确率提高。

异常检测实验的优化方向：

1. 聚类算法：K-Means算法对于噪声点或者非球形簇的处理效果不佳。可以尝试使用基于密度的聚类算法。

2. 优化簇数量选择：K-Means算法需要事先确定簇的数量K，但这个值的选择对于异常检测的结果产生影响较大。可以使用基于信息准则的方法，如Gap统计量和平均轮廓系数来优化K值的选择。

3. 引入权重机制：在实际应用中，不同的数据点在异常检测中具有不同的重要性。可以引入权重机制来考虑每个数据点的重要性。

4. 模型评价指标：在选择簇的数量、判断是否是异常点和比较不同算法效果时，需要使用模型评价指标来衡量模型性能。可以使用混淆矩阵、精确度、召回率、F1分数和ROC曲线等评价指标来对K-Means算法进行评价和优化。