**程序报告**

学号： 2112060 姓名：孙蕗

1. **问题重述**

（简单描述对问题的理解，从问题中抓住主干，必填）

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

K-means异常点检测模型中各个参数并没有被调整到最佳状态，效果并不好，尝试构造更多特征使模型获得更强的能力。

1. **设计思想**

（所采用的方法，有无对方法加以改进，该方法有哪些优化方向（参数调整，框架调整，或者指出方法的局限性和常见问题），伪代码，理论结果验证等… **思考题，非必填**）

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

1. 引入非线性关系和时间关系，构造新的特征，探究异常点检测。考虑异常出现的位置，肯呢个与时间点有关，肯呢个每一次产生异常点的时间段总是集中在某一个时间点，所以添加时间信息作为特征。
2. 采用 PCA (主成分分析)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。特征经过 PCA 处理后的 3 个特征就可以表达原数据 80% 以上的数据信息，用这新的 3 个特征就可以较好的表达原数据。
3. KMeans算法按照样本之间的距离大小，将样本集划分为 K 个簇。让簇内的点尽量紧密的连在一起，而让簇间的距离尽量的大。使用 KMeans 聚类方法完成检测，在 Kmeans 方法中，每个数据点具有两个属性：该点所属的簇和该点与各个簇中心点的距离。我们希望每一个点找到距离自己最近的簇，而且不同簇之间距离尽量大。KMeans 的分步优化的方法：

第一步根据当前各个簇中心，计算每个数据点与各个簇中心的距离，将该点划分为距离簇中心最近的簇类别；第二步根据重新划分的簇，更新每个簇的中心位置。直至簇中心不再变化时停止优化。

1. **代码内容**

（能体现解题思路的主要代码，有多个文件或模块可用多个"===="隔开，必填）

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

import os

import sklearn

import numpy as np

import pandas as pd

import random

from sklearn.decomposition import PCA

from copy import deepcopy

from sklearn.externals import joblib

from sklearn.preprocessing import StandardScaler

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

# 例如

# df['hours'] = df['timestamp'].dt.hour

# df['daylight'] = ((df['hours'] >= 7) & (df['hours'] <= 22)).astype(int)

df['timestamp'] = pd.to\_datetime(df['timestamp'])

# 将 df 数据按时间序列排序，方便数据展示

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['daylight'] = ((df['hours'] >= 7) & (df['hours'] <= 22)).astype(int)

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

data = df[columns]

scaler = StandardScaler()

data = scaler.fit\_transform(data)

# 通过 n\_components 指定需要降低到的维度

n\_components = 3

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

data = pca.fit\_transform(data)

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

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

return data

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

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\_[int(kmeans.labels\_[i]),:n\_features]

distance.append(np.linalg.norm(point - center))

distance = pd.Series(distance)

return distance

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

from copy import deepcopy

def get\_anomaly(data, kmeans, 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)

# data['is\_anomaly'] = None

ratio = 0.03

num\_anomaly = int(len(data) \* ratio)

new\_data = deepcopy(data)

new\_data['distance'] = get\_distance(new\_data,kmeans,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

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

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')

# 获取异常点数据信息

is\_anomaly = get\_anomaly(preprocess\_data, kmeans, ratio)

return is\_anomaly, preprocess\_data, kmeans, ratio

1. **实验结果**

（实验结果，必填）

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



1. **总结**

（自评分析（是否达到目标预期，可能改进的方向，实现过程中遇到的困难，从哪些方面可以提升性能，模型的超参数和框架搜索是否合理等），**思考题，非必填**）

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

1. 尝试调整模型参数，提高模型的性能
2. 尝试引入更多特征，使模型获得更高的能力