KMeans异常检测

学号：3200102312；姓名：胡若凡

一、问题重述

本实验的数据集为，2011.7-2011.9内各个时间段内的广告曝光率，主要使用了cpm和cpc两种数据指标。本实验的任务为，通过KMeans算法，进行无监督学习，尽可能精确的统计出指标异常的数据点。

二、KMeans算法设计思想

KMeans算法的具体设计思路如下：

聚类：是一种无监督学习，就是在给定的数据集中根据特征的相似度或者距离，将数据集中的对象归并到若干个类中

KMeans聚类：K指代算法需要将样本集合划分称为K个子集，每个样本只属于一个类，是一种硬聚类的算法。对于特征间的距离，算法采用欧几里得距离，并用所有点到其所在的类的中心的距离的平方和作为损失函数，并可以得到算法的目标其实就是最优化损失函数：

$$C^* = \arg\min W(C) = \arg\min \sum_{l=1}^{k} \sum_{C(i)=l} ||x_i - \bar{x}_l||^2$$

实现步骤：该算法的具体实现步骤思路如下:

1. 首先要随机选 K 个样本作为初始的 K 个类的中心点，然后进行如下步骤
2. 对于每个样本点，计算其与 K 个类的中心点的距离，然后在选择最近的中心点所在的类作为该样本点的类
3. 上一步循环完成之后，在每个类中找到使得到该类中的所有样本点距离最小的中心点， 使得该类中距离，也就是该类的样本中所有点的均值
4. 然后回到第 2 步用新的 K 个中心点计算出新的分类，重复上述步骤直到划分结果不再改 变，就得到了 K 均值聚类的最终结果

三、KMeans算法代码内容

对于算法设计内容，这里结合mo平台的KMeans类来进行说明

该类包括两部分：

init部分：

用于对聚类个数，计算次数，迭代次数进行指定，若无直接调用指定，则使用默认参数

fit部分

用于实现KMeans的算法，具体思路如下：

- 数据格式转换，将要处理的数据转换为处理numpy数组
- 随机选择中心点
- 将数据集x中的每个点分配到距离其最近的质心所在的簇
- 重新计算所有簇的质心坐标

- 把使得簇内均方误差最小的结果，并将该结果的簇中心点坐标和数据点所属的簇标签保存到best_centers和best_labels中。

```
class KMeans():
    """
    Parameters
    ----------
    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
    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
    """
    def __init__(
                self,
                n_clusters=8,
                n_init=10,
                max_iter=300
                ):

        self.n_clusters = n_clusters
        self.max_iter = max_iter
        self.n_init = n_init


    def fit(self, x):
        """
        用fit方法对数据进行聚类
        :param x: 输入数据
        :best_centers: 簇中心点坐标 数据类型: ndarray
        :best_labels: 聚类标签 数据类型: ndarray
        :return: self
        """
        #x 若为 Pandas 数据框类型，先将其转换为 numpy 数组类型再进行聚类操作
        if isinstance(x, pd.DataFrame):
            x = x.values

        best_mse = np.inf
        for i in range(self.n_init):
            #获取中心点
            center_indices = np.random.choice(len(x), self.n_clusters,
replace=False)
            centers = x[center_indices]
            for j in range(self.max_iter):
                #将数据集x中的每个点分配到距离其最近的质心所在的簇
                dists = np.linalg.norm(x[:, np.newaxis, :] - centers, axis=-1)
                labels = np.argmin(dists, axis=-1)
                #重新计算所有簇的质心坐标；
                new_centers = np.array([x[labels == k].mean(axis=0) for k in
range(self.n_clusters)])
                if np.allclose(centers, new_centers):
                    break
                else:
                    centers = new_centers
            #使簇内均方误差最小的结果，并将该结果的簇中心点坐标和数据点所属的簇标签保存到
best_centers和best_labels中。
            mse = ((x - centers[labels])**2).sum()
            if mse < best_mse:
```

```
                best_mse = mse
                best_centers = centers
                best_labels = labels

        self.cluster_centers_ = best_centers
        self.labels_ = best_labels
        return self
```

四、KMeans算法实验结果

Calinski-Harabasz指数是一种聚类结果的评估指标，用于评估聚类效果的好坏，通常取值越大越好；轮廓系数也是另一种评估聚类效果的指标，通常取值范围在[-1, 1]之间，越接近1则聚类效果越好。

使用参数：

n_clusters = 9

n_init = 20

max_iter = 300

```
聚类数目:9  calinski_harabasz_score:1192.12     silhouette_score:0.51
```

五、总结

对于KMeans的优化，我了解到了还有以下几个改进的措施：

- **层次聚类**：KMeans 算法基于欧几里得距离度量进行聚类，对于某些不符合欧几里得距离假设的数据并不适用。层次聚类是一种非参数聚类方法，它利用样本之间的相似性和距离等信息动态地构建聚类树，能够在无需指定聚类数目的情况下快速得出聚类结果。
- **MiniBatch KMeans**：本次的数据集较小，对于大型数据集，KMeans 算法的计算复杂度很高。MiniBatch KMeans 利用随机梯度下降，将数据划分成若干批次（batch），每次只使用一批数据来更新聚类中心，从而加速算法的收敛速度
- **KMeans++ 初始化**：在原始 KMeans 算法中，初始化聚类中心是随机选取的，可能会收敛到子优解。而 KMeans++ 算法通过一定的随机概率选择更好的初始点，提升了算法的稳定性和精度，能够有效地防止陷入局部最小值。

对于KMeans和决策树算法做分类时的相同的：

- 都需要类间节点的关系：KMeans使用的是一个类间点的欧几里得距离，而决策树则是看的节点之前的纯度关系

- 都需要多次划分：KMeans 算法通过迭代更新聚类中心来最小化样本点到聚类中心的距离，决策树算法通过不断分裂来得到最优的特征划分。