程序报告

一、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）

==

KMeans 算法部分：

```
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
        self.labels_ = []
        self.cluster_centers_=[]


    def randCenters(self, data):
        # number of points & dimensions
        n, dimensions = data.shape
        # the matrix storing the centers
        centers = np.zeros((self.n_clusters, dimensions))
        # the index of centers
        #    indexes    =    [int(random.uniform(0,    n-1))    for    i    in
range(self.n_clusters)]
        indexes = [random.randint(0, n-1) for i in range(self.n_clusters)]
        while len(set(indexes)) != len(indexes):
            # avoid duplicate indexes
            indexes    =    [int(random.uniform(0,    n))    for    i    in
range(self.n_clusters)]
```

```
        for i in range(self.n_clusters):
            centers[i, :] = data.iloc[indexes[i], :]
        return centers

    def run(self, data):
        # number of points & dimensions
        n, dimensions = data.shape
        # [index, error]
        result = np.mat(np.zeros((n, 2)))
        result = result - 1
        changed = True
        iterations = self.max_iter

        # 1. create initial centers
        centers = self.randCenters(data)

        # 4. terminate iteration if centers do not change or it reaches the
limitation
        while changed and iterations:
            changed = False
            iterations -= 1
            # 2. cluster
            for i in range(n):
                # find the closest centroid
                point = np.array(data.iloc[i, :])
                distances = [np.linalg.norm(point - centers[j, :]) for j in
range(self.n_clusters)]
                ind = distances.index(min(distances))
                if result[i, 0] != ind:
                    changed = True
                    result[i, :] = ind, distances[ind]**2
            # 3. update centers
            for j in range(self.n_clusters):
                # points in j-th cluster
                points = np.array(data)[np.nonzero(result[:,0].A == j)[0]]
                centers[j, :] = np.mean(points, axis=0)
        return centers, result

    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
        """

########################################################################
############
        #### 请勿修改该函数的输入输出 ####

########################################################################
############
        # #

        # #

########################################################################
############
        ############# 在生成 main 文件时, 请勾选该模块 #############

########################################################################
############

        minsum = np.inf
        for i in range(self.n_init):
            centers, result = self.run(x)
            # 求和保留最小的
            target = np.sum(result, axis=0)[0, 1]
            if target < minsum:
                minsum = target
                best_centers = centers
                best_labels = result[:, 0].astype(int)
        self.cluster_centers_ = best_centers
        self.labels_ = best_labels
        return self
```

main.py 中的其余代码：

```
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['timestamp'] = pd.to_datetime(df['timestamp'])
    df = df.sort_values(by='timestamp').reset_index(drop=True)
    df['hours'] = df['timestamp'].dt.hour
    df['daylight'] = ((df['hours'] >= 7) & (df['hours'] <= 22)).astype(int)


    # ========================  模型加载  ===========================
    # 请确认需要用到的列名，e.g.:columns = ['cpc','cpm']
    columns = ['cpc', 'cpm', 'hours', 'daylight']
    data = df[columns]
    # 例如
    scaler = joblib.load('./results/scaler.pkl')
    pca = joblib.load('./results/pca.pkl')
    data = scaler.fit_transform(data)
    data = pd.DataFrame(data, columns=columns)
    # dimensionality reduction
    n_components = pca.n_components_
    data = pca.fit_transform(data)
    data = pd.DataFrame(data,columns=['Dimension' + str(i+1) for i in
range(n_components)])
    data = deepcopy(data)

    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])
        kmeans.labels_[i]
        center = kmeans.cluster_centers_[kmeans.labels_[i],:n_features]
        distance.append(np.linalg.norm(point - center))
    distance = pd.Series(distance)

    return distance
```

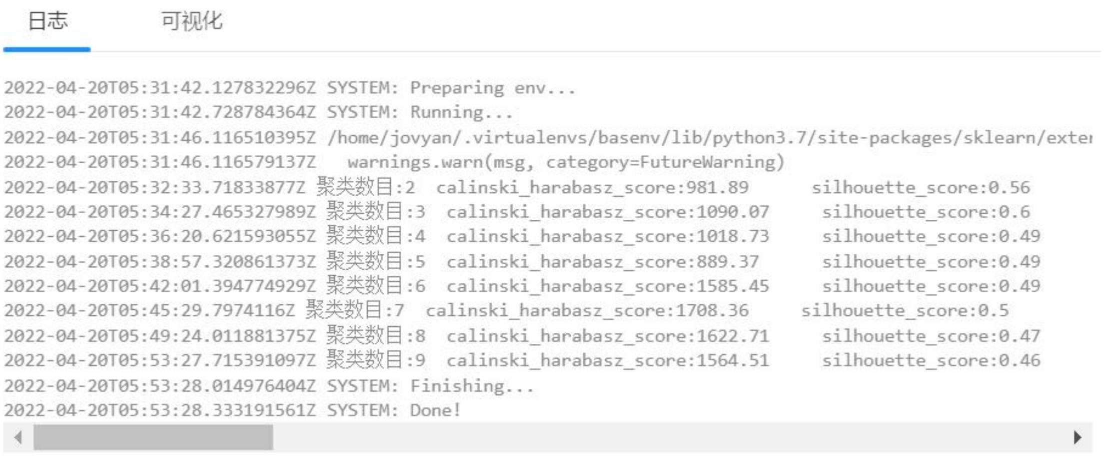
```
def get_anomaly(data, kmean, ratio):
    """
    检验出样本中的异常点，并标记为 True 和 False，True 表示是异常点

    :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 = deepcopy(data)
    data['distance']                                                 =
get_distance(data,kmean,n_features=len(data.columns))
    threshould                                                       =
data['distance'].sort_values(ascending=False).reset_index(drop=True)[nu
m_anomaly]
    data['is_anomaly'] = data['distance'].apply(lambda x: x > threshould)
    return 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/km.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
```

二、实验结果

（实验结果，必填）

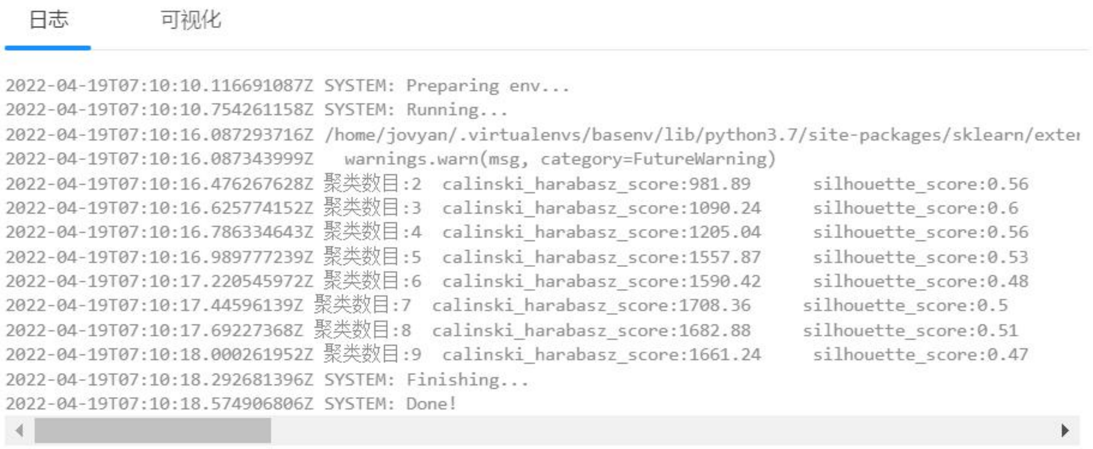
==

以下是自己实现的 KMeans 聚类的结果：



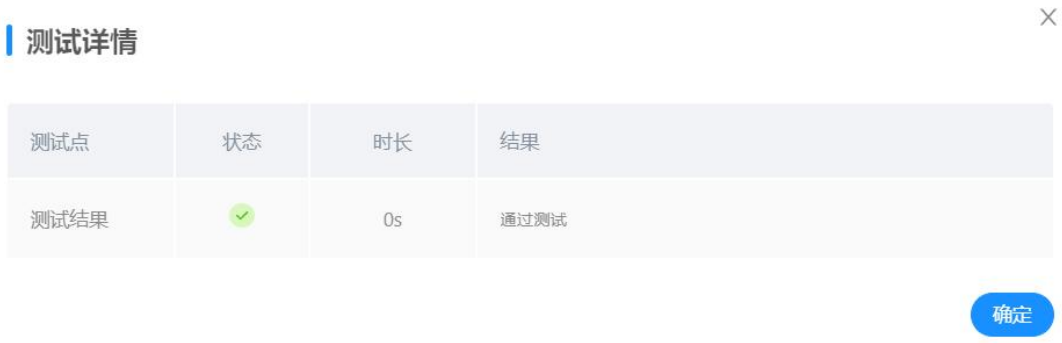
```
2022-04-20T05:31:42.127832296Z SYSTEM: Preparing env...
2022-04-20T05:31:42.728784364Z SYSTEM: Running...
2022-04-20T05:31:46.116510395Z /home/jovyan/.virtualenvs/basenv/lib/python3.7/site-packages/sklearn/exte
2022-04-20T05:31:46.116579137Z   warnings.warn(msg, category=FutureWarning)
2022-04-20T05:32:33.71833877Z 聚类数目:2  calinski_harabasz_score:981.89      silhouette_score:0.56
2022-04-20T05:34:27.465327989Z 聚类数目:3  calinski_harabasz_score:1090.07     silhouette_score:0.6
2022-04-20T05:36:20.621593055Z 聚类数目:4  calinski_harabasz_score:1018.73     silhouette_score:0.49
2022-04-20T05:38:57.320861373Z 聚类数目:5  calinski_harabasz_score:889.37      silhouette_score:0.49
2022-04-20T05:42:01.394774929Z 聚类数目:6  calinski_harabasz_score:1585.45     silhouette_score:0.49
2022-04-20T05:45:29.7974116Z 聚类数目:7  calinski_harabasz_score:1708.36     silhouette_score:0.5
2022-04-20T05:49:24.011881375Z 聚类数目:8  calinski_harabasz_score:1622.71     silhouette_score:0.47
2022-04-20T05:53:27.715391097Z 聚类数目:9  calinski_harabasz_score:1564.51     silhouette_score:0.46
2022-04-20T05:53:28.014976404Z SYSTEM: Finishing...
2022-04-20T05:53:28.333191561Z SYSTEM: Done!
```

以下是 sklearn 中的 KMeans 聚类结果：



```
2022-04-19T07:10:10.116691087Z SYSTEM: Preparing env...
2022-04-19T07:10:10.754261158Z SYSTEM: Running...
2022-04-19T07:10:16.087293716Z /home/jovyan/.virtualenvs/basenv/lib/python3.7/site-packages/sklearn/exte
2022-04-19T07:10:16.087343999Z   warnings.warn(msg, category=FutureWarning)
2022-04-19T07:10:16.476267628Z 聚类数目:2  calinski_harabasz_score:981.89      silhouette_score:0.56
2022-04-19T07:10:16.625774152Z 聚类数目:3  calinski_harabasz_score:1090.24     silhouette_score:0.6
2022-04-19T07:10:16.786334643Z 聚类数目:4  calinski_harabasz_score:1205.04     silhouette_score:0.56
2022-04-19T07:10:16.989777239Z 聚类数目:5  calinski_harabasz_score:1557.87     silhouette_score:0.53
2022-04-19T07:10:17.220545972Z 聚类数目:6  calinski_harabasz_score:1590.42     silhouette_score:0.48
2022-04-19T07:10:17.44596139Z 聚类数目:7  calinski_harabasz_score:1708.36     silhouette_score:0.5
2022-04-19T07:10:17.69227368Z 聚类数目:8  calinski_harabasz_score:1682.88     silhouette_score:0.51
2022-04-19T07:10:18.000261952Z 聚类数目:9  calinski_harabasz_score:1661.24     silhouette_score:0.47
2022-04-19T07:10:18.292681396Z SYSTEM: Finishing...
2022-04-19T07:10:18.574906806Z SYSTEM: Done!
```

可以看到大多结果较为一致，但是自己实现的 KMeans 运行速度明显比调用包的慢。最后通过了测验：



测试详情

测试点	状态	时长	结果
测试结果	✓	0s	通过测试