实验内容	基于 Kmeans 聚类算法与线性回归模型对主要城市分组并探究城市环境与城市发展的关系

小组成员

姓名	班级	学号
苏现亮	数 211-2	202163501205

实验目的

1,掌握 k-means 聚类算法，理解并掌握基本原理
2,收集城市的相关数据，掌握搜集数据的能力
3,学会分析聚类的结果，并评价聚类结果的合理性
4,学会使用统计学（线性回归）分析城市环境与城市发展的关系

实验原理、步骤、算法

一,实验步骤

1,问题一的实验步骤

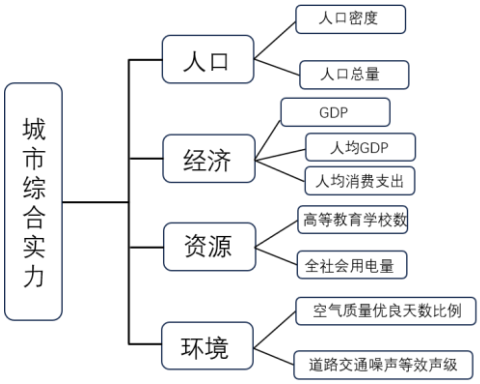
Step1.在中国统计年鉴的资源环境页面，找到主要城市的相关数据，将找到的所有数据复制于 2022.xlsx 与 2012. xlsx 中。

Step2.将两个 excel 文件放入文件‘问题一.py’同一个文件夹下,分别读取两个表格的数据，调用 Kmeans 算法，设置聚类数为 4 组，迭代 100 次，得到两组数据的分类结果。

Step3.手动将分类结果一一对应相应的城市，得到分类情况，用表格表示出来。

2.问题二的实验步骤

Step1.结合城市的人口或经济数据，为了便于分析聚类结果，我们要使寻找的数据能体现城市的综合实力，建立数据指标体系。在统计年鉴中寻找相应的数据，将数据存入‘数据集 2.xlsx‘。



Step2.将这个 excel 文件放入文件‘问题二.py’同一个文件夹下,读取表格的数据，调

用 Kmeans 函数算法，设置聚为 4 组，迭代 1000 次，得到分类结果。

Step3.为分析聚类结果的合理性，我们从三个方面进行分析，分别是内部聚合度，间部分离度和可解释性。内部聚合度主要考虑轮廓系数与簇内平均距离；间部分离度主要考虑 Davies-Bouldin 指数与簇间平均距离;可解释性考虑聚类结果与城市分级的相似程度。(注意：此处所说的城市分级是指，主流媒体对中国各大城市的等级划分)。

Step4.将聚类结果与具体分析结果写入实验结果部分。

3. 问题三的实验步骤

Step1.使用城市 GDP 数据反映城市的发展，使用城市的空气质量优良天数比例与道路交通噪声等效声级数据反映城市环境。

Step2.令城市 GDP 数据作为因变量，城市的空气质量优良天数比例与道路交通噪声等效声级数据作自变量。为了减小特殊地理环境与政策优惠城市对回归模型的干扰，我们利用 IsolationForest 模型，剔除显著离群的数据，并构建线性回归模型，并检验线性回归模型，该步骤代码为‘问题三 1.py’。

Step3.分别构建城市 GDP 与城市的空气质量优良天数比例与道路交通噪声的线性回归模型，分别分析它们之间的关系，该步骤代码分别为‘问题三 2.py’ 与‘问题三 3.py’。

二,实验原理

1. K-means 算法是一种常用的基于距离的聚类算法，其实验原理如下：

[1] 随机选择 k 个初始聚类中心：在样本集合中随机选择 k 个点作为初始的聚类中心。

[2] 根据距离度量将每个样本点分配到最近的聚类中心：对于每个样本点，计算其与 k 个聚类中心的距离，将其分配到距离最近的聚类中心所代表的类别。

[3] 重新计算每个聚类的中心：对于每个聚类，重新计算它的中心（即该聚类所有样本点的均值)。

[4] 重复步骤 2 和 3，直到聚类中心不再发生变化或者达到预设的迭代次数。

[5] 输出最终的聚类结果：输出每个样本点所属的聚类类别

2. 轮廓系数

对于一个样本集合，它的轮廓系数是所有样本轮廓系数的平均值。轮廓系数的取值范围是[-1,1]，同类别样本距离越相近，不同类别样本距离越远，分数越高。

针对某个样本的轮廓系数 s 为：

$$s = \frac{b-a}{Max(a,b)}$$

其中,a 表示某个样本与其所在簇内其他样本的平均距离，b 表示某个样本与其他簇样本的平均距离。

聚类总的轮廓系数 SC 为:

$$SC = \frac{\sum_{i=1}^{N} s_i}{N}$$

3. 簇内平均距离

具体来说，对于某个簇 C_i，其簇内平均距离定义为:

$$D_i = \frac{1}{n_i(n_i-1)} \sum_{j=1}^{n_i} \sum_{k=1,k\neq j}^{n_i} d(x_j, x_k)$$

其中，n_i 表示簇 C_i 中的数据点数量，x_i 和 x_k 分别表示簇 C_i 中的第 j 个和第 k 个数据点，

$d(x_j, x_k)$ 表示这两个点的距离。

4. Davies-Bouldin 指数

对于簇 C_i，计算该簇内所有样本点之间的平均距离，表示为 s_i，表示为平均内部距离。

对于簇 C_i，计算其与其他所有簇中心点的平均距离 S_i，表示为平均外部距离。

对于簇 C_i，计算其 Davies-Bouldin 指数 DB_i，如下式所示

$$DB_i = \frac{s_i + S_i}{s_i}$$

5. 簇间距离

对于第 i 个簇 C_i，n_i 表示簇 C_i 中的样本点的数量。

簇 C_i 和簇 C_k 的簇间距离可以表示为:

$$d_{ik} = \frac{1}{n_i n_k} \sum_{x\in C_i} \sum_{x\in C_k} \|x-y\|$$

其中，$\|x-y\|$ 表示样本点 x 和 y 之间的欧氏距离。

6. 线性回归

线性回归是一种用于建立特征（自变量）和目标值（因变量）之间线性关系的统计学方法。其原理可以总结如下:

[1]**线性关系假设:** 线性回归假设自变量 x 和因变量 y 之间存在线性关系，即可以用线性函数来描述它们之间的关系。

[2]**线性模型：**线性回归模型通过以下线性方程来表示自变量和因变量之间的关系：

$$y = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + \varepsilon$$

其中，$x_1, x_2, \cdots x_n$是自变量的特征，$w_0, w_1, \cdots, w_n$是模型的参数，y是因变量的值，ε是误差项。

[3]**最小二乘法：**线性回归通常使用最小二乘法来估计模型的参数。最小二乘法的目标是使得模型预测值与实际观测值之间的残差平方和最小化。通过最小化残差平方和，可以得到最优的参数估计值，从而获得最拟合的线性模型。

[4]**参数估计：**通过最小化残差平方和，可以得到参数的估计值，这些估计值使得模型对观测数据拟合最好。

[5]**模型评价：**一旦获得了线性回归模型的参数估计值，就可以使用各种评估指标（如均方误差、R 平方等）来评价模型的拟合程度和预测能力。

[6]**预测：**最终，线性回归模型可以用于预测新的自变量对应的因变量取值。将新的自变量特征代入线性方程中，即可得到对应的预测值。

三、实验算法

1.Kmeans 聚类算法：

[1]初始化 K 个中心$m_1, m_2, \cdots m_K$，确定距离函数；

[2]计算每个点x_i到每个中心点m_k的距离d_{ik}，记为$D \in \square^{N \times K}$；

[3]统计属于类k的样本点：

$$C_k = \left\{ i : \arg\min_j d_{ij} = k \right\}, k = 1, 2, \cdots K;$$

[4]更新聚类中心m_k'：

$$m_k' = \frac{1}{|C_k|} \sum_{i \in C_k} x_i, k = 1, 2, \cdots, K;$$

[5]倘若满足停机条件，如m, m'的距离小于一个阈值ε或者达到最大迭代次数便退出，否则回到步骤 2。

输出：类别中心$m_1, m_2, \cdots m_K$以及各样本所属类别C_k。

2.最小二乘法算法：

本文构建线性回归模型的算法是最小二乘法：

[1]**构建线性模型：**线性回归模型通过以下线性方程来表示自变量和因变量之间的关系：

$$y = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + \varepsilon$$

其中，$x_1, x_2, \cdots x_n$是自变量的特征，$w_0, w_1, \cdots, w_n$是模型的参数，y是因变量的值，ε是误差项。

[2]**定义损失函数：**定义损失函数，通常选择残差平方和作为损失函数，即

$$L=\sum(y_i-(w_0+w_1x_{i1}+w_2x_{i2}+\cdots+w_nx_{in}))^2$$

[3]**最小化损失函数：**对损失函数求偏导数，令偏导数等于零，求解模型参数 w_i 的最优值

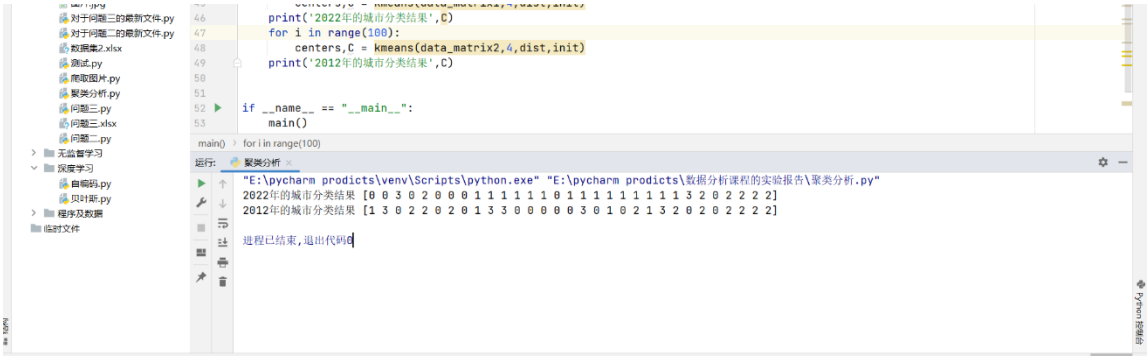
[4]**参数估计：**根据上述公式，计算出最优的参数估计值 w_i 。

[5]**评估模型：**使用各种评估指标（如均方误差、R 平方等）来评估模型的拟合程度和预测能力。

[6]**应用模型：**利用估计得到的参数，将新的自变量特征代入线性方程中，进行预测和分析。

实验结果

一,问题一



2022 年城市分组			
北京	上海	呼和浩特	石家庄
天津	南京	拉萨	昆 明
太 原	杭州	兰州	
沈阳	合肥	西宁	
长春	福州	银 川	
哈尔滨	南 昌	乌鲁木齐	
济南	郑州		
西安	武 汉		
	长沙		
	广州		
	南宁		
	海口		
	重庆		
	成都		
	贵阳		

2012 年城市分组			
石家庄	北京	太 原	天津
沈阳	上海	呼和浩特	南京
哈尔滨	广州	长春	杭州
合肥	重庆	海口	武汉
福州		贵阳	成都
南昌		拉萨	
济南		兰州	
郑州		西宁	
长沙		银川	
南宁		乌鲁木齐	
昆明			
西安			

二,问题二

1、聚类结果:

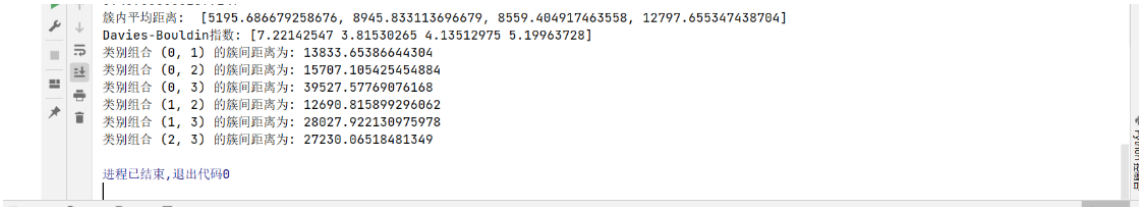
0	1	2	3
石家庄	天津	沈阳	北京
太 原	合肥	南京	上海
呼和浩特	福州	杭州	广州
长春	郑州	济南	
哈尔滨	武汉	长沙	
南 昌	重庆	昆 明	
南宁	成都		
海口	西安		
贵阳			
拉萨			
兰州			
西宁			
银 川			
乌鲁木齐			

2、各城市轮廓系数

城市	轮廓系数	城市	轮廓系数
北京	0.61453	武 汉	-0.07719
天津	0.328109	长沙	0.47531
石家庄	0.5677	广州	0.167873
太 原	0.598812	南宁	0.567317
呼和浩特	0.713167	海口	0.683556
沈阳	0.070616	重庆	0.305894
长春	0.438883	成都	0.364796

哈尔滨	0.670769	贵阳	0.609869
上海	0.632446	昆 明	0.264214
南京	0.268826	拉萨	0.568829
杭州	0.344218	西安	-0.08125
合肥	0.194846	兰州	0.708805
福州	0.266314	西宁	0.676079
南昌	0.54948	银 川	0.717861
济南	0.155593	乌鲁木齐	0.459039
郑州	0.221253		
整体轮廓系数: 0.4208567123730163			

3、其他结果数据



4、对聚类结果的分析

通过以下三个方面分析聚类结果的合理性，具体分析如下：

- 内部聚合度

  由于轮廓系数的取值范围在 [-1, 1] 之间，越接近 1 表示聚类效果越好，越接近 -1 表示聚类效果越差。若某个数据点的轮廓系数为负数，说明该点被错误地分配到了不应该属于的簇中。通过我们的计算结果，整体的轮廓系数约为 0.4208，相对接近 1，说明整体聚类效果较好；其中，西安与武汉的轮廓系数均为负数，说明这两座城市不属于所属的簇。对于 31 座城市，仅两座城市被错误分配，聚类的正确率为 93.5%，聚类的正确率较高。

  由于簇内平均距离越小，表示簇内数据点之间的相似性和紧密度越高，聚类效果越好。通过计算可以分析出，第 0 类簇的数据具有最高的相似性和紧密度，因此第 0 类簇的聚类效果最佳；反之，第 3 类簇的数据相似性和紧密度最低，所以第 3 类簇的聚类效果最差。

- 间部分离度

  由于簇的 Davies-Bouldin 指数是用来衡量这个簇内部紧密度和与其他簇之间的分离度之间的平衡关系的。指数越小，说明该簇内部数据点越紧密且该簇与其他簇之间的分离度越大，表示该聚类结果中这个簇的质量越好。通过计算可以分析出，第 1 类簇内部数据点紧密且该簇与其他簇之间的分离度大，第 1 类簇的质量最好；第 0 类簇的内部数据点的紧密程度与该簇与其他簇之间的分离度不平衡，由于之前对第 0 类簇的簇内平均距离的分析，推测第 0 类簇与其他簇之间的分离度最大。

  由于簇间平均距离是指不同簇之间数据点的平均距离。在评价聚类结果时，簇间平均距离可以帮助我们衡量不同簇之间的分离度，即不同簇之间的数据点距离越远，表示这些簇更加分离，聚类结果的质量会更好。通过分析，我们发现

簇组合（0，3），（1，3），（2，3）的簇间平均距离较大，说明第 3 类簇与其他簇的分离度较大；

- 可解释性

本文所找的数据，均是能体现城市综合实力的数据，所以聚类后相同簇内城市的综合实力应该相似，不同簇内城市的综合实力差别大。我们结合网址 2022 年中国 336 座城市综合实力排行榜（完整榜）_(baidu.com)其中对于城市综合实力的分级，比较聚类簇中城市的分级数量情况，分析所聚类的科学性与准确性。

由以上网址整理出所选城市的分级，如下所示

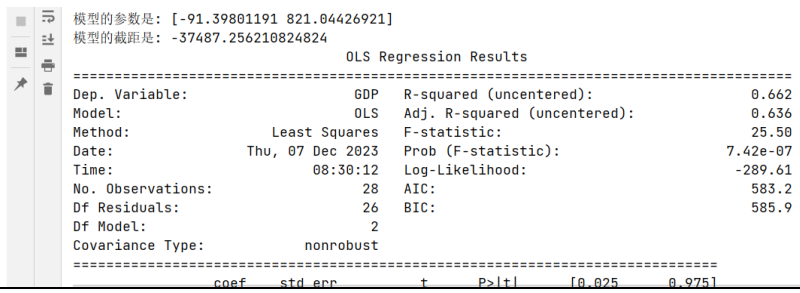
一线城市	新一线城市	二线城市	三线及以下城市
上海	武汉	合肥	乌鲁木齐
北京	南京	福州	海口
广州	成都	济南	银川
	重庆	哈尔滨	呼和浩特
	天津	石家庄	西宁
	杭州	长春	拉萨
	沈阳	贵阳	
	西安	太原	
	长沙	兰州	
	郑州	南昌	
		南宁	
		昆明	

统计聚类后各簇内不同城市的数目，如下图所示

簇类别	一线城市	新一线城市	二线城市	三线及以下城市	总数
0 类	0	0	8	6	14
1 类	0	6	2	0	8
2 类	0	4	2	0	6
3 类	3	0	0	0	3
总数	3	10	12	6	

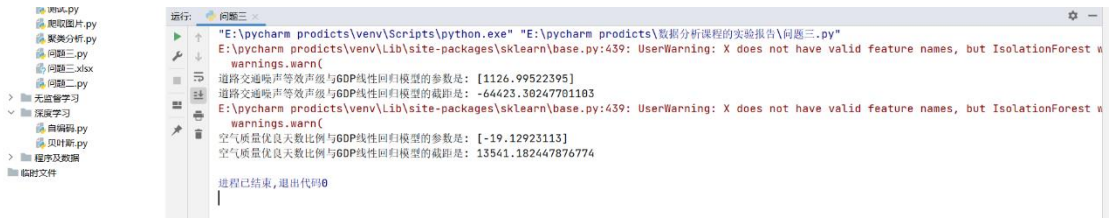
通过分析以上统计数据，相同等级的城市被分到同一簇的比例较高，其中一线城市与三线及以下城市均分别被分入相同簇内；新一线城市与二线城市由于数量较多，未分入相同簇内，但同级城市分入某簇的聚集程度较高。例如，67%的二线城市被分入 0 类簇，60%的新一线城市被分入 1 类簇中。这说明，聚类结果能够表现城市的综合实力的差异，具有可解释性以及科学性。

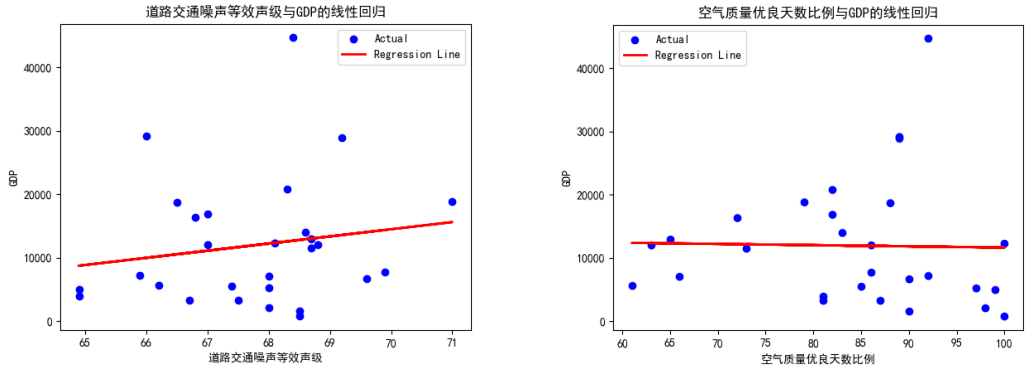
三,问题三



```
模型的参数是: [-91.39801191 821.04426921]
模型的截距是: -37487.256210824824
                            OLS Regression Results
==============================================================================
Dep. Variable:                    GDP   R-squared (uncentered):                   0.662
Model:                            OLS   Adj. R-squared (uncentered):              0.636
Method:                 Least Squares   F-statistic:                              25.50
Date:                Thu, 07 Dec 2023   Prob (F-statistic):                    7.42e-07
Time:                        08:30:12   Log-Likelihood:                         -289.61
No. Observations:                  28   AIC:                                      583.2
Df Residuals:                      26   BIC:                                      585.9
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
```

分析：该线性回归模型的 R-squared 为 0.662，说明回归方程对观测数据的拟合度较好，拟合效果较好。F 统计量的值为 25.50，用来检验所有自变量的系数是否同时显著不为零。在这里的显著性水平为 7.42e-07，非常接近于零，这表示我们有充分的理由拒绝“所有自变量系数均为零”的原假设，即回归方程整体是显著的。





分析：分别作道路交通噪声等效声级，空气质量优良天数比例与 GDP 的回归模型，并将图像直观展示出来。显然，城市道路交通噪声与城市 GDP 是正相关的，城市空气优良天数比例与城市 GDP 是负相关的。由两个模型的回归参数可知，道路交通噪声相比于空气质量优良天数比例，对城市 GDP 的影响更显著。结合这两组数据，道路交通噪声等效声级与环境优劣程度是负相关的，空气质量优良天数比例与环境优劣程度是正相关的。所以，环境越好，GDP 就越低，城市发展就越差。综上，城市环境与城市发展是负相关的。

由于数据量较大，本文所用数据集见下链接：

(以下链接中的文件有两个，原始数据集包含城市标签，不能直接在代码中使用，代码数据集可以在代码中直接使用。)

链接：https://pan.baidu.com/s/185gcsdXpY28xNGCwlOUyGg

提取码：8l74

附录：程序代码

问题一

```
# File:问题一.py
from   typing import Callable,Tuple
import numpy as np
import pandas as pd

# 读取 Excel 文件
df1 = pd.read_excel('2022.xlsx')
df2= pd.read_excel('2012.xlsx')

# 将数据转换为矩阵
data_matrix1 = df1.to_numpy()
data_matrix2 = df2.to_numpy()

def kmeans(
    X:np.ndarray,K:int,
    dis_fn:Callable,init_fn:Callable,
    eps:float=1e-4,max_iters:int=100,
    **kwargs
)->Tuple[np.ndarray]:

    centers=init_fn(X,K,dis_fn=dis_fn,**kwargs)
    eps=np.mean(np.linalg.norm(X,axis=-1))*eps
    for _ in range (max_iters):
        D=dis_fn(X,centers)
        C=np.argmin(D,axis=-1)
        new_center=np.zeros_like(centers)
        for k in range(K):
            new_center[k]=np.mean(X[C==k],axis=0)
        if np.all(np.linalg.norm(centers-new_center,axis=-1)<eps):
            break
        centers=new_center
    C=np.argmin(dis_fn(X,centers),axis=-1)
    return   centers,C

def dist(x,y):
    x=x[:,None,:]
    return np.linalg.norm(x-y,axis=-1)

def init(X,K,**kwargs):
    X=np.copy(X)
    np.random.shuffle(X)
```

```
    return X[:K]

def main():
    for i in range(100):
        centers,C = kmeans(data_matrix1,4,dist,init)
    print('2022 年的城市分类结果',C)
    for i in range(100):
        centers,C = kmeans(data_matrix2,4,dist,init)
    print('2012 年的城市分类结果',C)

if __name__ == "__main__":
    main()
```

问题二

```
# File:问题二.py

from   typing import Callable,Tuple
import numpy as np
import pandas as pd
from sklearn.metrics import silhouette_samples, silhouette_score
from sklearn.metrics import pairwise_distances
from scipy.spatial.distance import cdist

# 读取 Excel 文件

df = pd.read_excel('数据集 2.xlsx')

# 将数据转换为矩阵
data_matrix = df.to_numpy()

def kmeans(
    X:np.ndarray,K:int,
    dis_fn:Callable,init_fn:Callable,
    eps:float=1e-4,max_iters:int=100,
    **kwargs
)->Tuple[np.ndarray]:

    centers=init_fn(X,K,dis_fn=dis_fn,**kwargs)
    eps=np.mean(np.linalg.norm(X,axis=-1))*eps
    for _ in range (max_iters):
        D=dis_fn(X,centers)
        C=np.argmin(D,axis=-1)
        new_center=np.zeros_like(centers)
```

```
            for k in range(K):
                new_center[k]=np.mean(X[C==k],axis=0)
            if np.all(np.linalg.norm(centers-new_center,axis=-1)<eps):
                break
            centers=new_center
        C=np.argmin(dis_fn(X,centers),axis=-1)
        return   centers,C
#
X=data_matrix
#
def dist(x,y):
    x=x[:,None,:]
    return np.linalg.norm(x-y,axis=-1)

def init(X,K,**kwargs):
    X=np.copy(X)
    np.random.shuffle(X)
    return X[:K]


def Calculate_profile_coefficient(data,labels):
    # 计算轮廓系数
    silhouette_avg = silhouette_score(data, labels)
    print("整体轮廓系数:", silhouette_avg)

    # 计算每个样本点的轮廓系数
    sample_silhouette_values = silhouette_samples(data, labels)
    for i in range(len(data)):
        # print("样本点", i+1, "的轮廓系数:", sample_silhouette_values[i])
        print( sample_silhouette_values[i])
def avg_intra_cluster_distance(data, labels):
    """
    计算簇内平均距离
    :param data: 数据集，每行表示一个样本点
    :param labels: 样本点所属的簇标签
    :return: 簇内平均距离列表，长度为簇的个数
    """
    k = len(set(labels))  # 簇的个数
    distances = cdist(data, data)  # 计算所有样本点之间的距离矩阵
    avg_distances = []
    for i in range(k):
        indices = np.where(labels == i)[0]  # 获取属于第 i 个簇的样本点的索引
        if len(indices) > 1:
            dists = distances[indices][:, indices]  # 获取这些样本点之间的距离
```

```
            avg_dist = np.mean(dists[np.triu_indices(len(dists), k=1)])  # 取这些距离的平均值
            avg_distances.append(avg_dist)
        else:
            avg_distances.append(0)  # 如果簇中只有一个样本点，平均距离为 0\
    print("簇内平均距离: ",avg_distances)


def davies_bouldin_score(X, labels):
    # 计算 Davies-Bouldin 指数
    num_clusters = len(np.unique(labels))
    cluster_centers = []
    for i in range(num_clusters):
        cluster_centers.append(np.mean(X[labels == i], axis=0))

    distances = pairwise_distances(cluster_centers, metric='euclidean')

    scores = np.zeros(num_clusters)
    for i in range(num_clusters):
        intra_cluster_distances = pairwise_distances(X[labels == i], [cluster_centers[i]],
metric='euclidean')
        avg_intra_cluster_distance = np.mean(intra_cluster_distances)

        inter_cluster_distances = distances[i]
        inter_cluster_distances = np.delete(inter_cluster_distances, i)
        avg_inter_cluster_distance = np.mean(inter_cluster_distances)

        scores[i] = (avg_intra_cluster_distance + avg_inter_cluster_distance) /
avg_intra_cluster_distance

    return scores


def calculate_average_distance_between_clusters(data, labels):
    # 计算簇间平均距离
    num_clusters = len(set(labels))  # 簇的数量
    distances = []  # 存储不同簇之间的距离

    for i in range(num_clusters):
        for j in range(i + 1, num_clusters):
            cluster_i_indices = np.where(labels == i)[0]  # 簇 i 的样本索引
            cluster_j_indices = np.where(labels == j)[0]  # 簇 j 的样本索引

            cluster_i_samples = data[cluster_i_indices]  # 簇 i 的样本集合
            cluster_j_samples = data[cluster_j_indices]  # 簇 j 的样本集合
```

```
                distance_ij = np.mean(
                    np.linalg.norm(cluster_i_samples[:, np.newaxis] - cluster_j_samples, axis=2))  # 计
算簇 i 和簇 j 之间的平均距离
                distances.append(distance_ij)
                print("类别组合", (i, j), "的簇间距离为:", distance_ij)


def main():
    # 主函数
    for i in range(1000):
        centers, C = kmeans(X, 4, dist, init)
    print(C)
    Calculate_profile_coefficient(X, C)  # 计算轮廓系数
    avg_intra_cluster_distance(X, C)  # 计算簇内距离
    Davies_Bouldin = davies_bouldin_score(X, C)
    print("Davies-Bouldin 指数:", Davies_Bouldin )
    calculate_average_distance_between_clusters(X, C)


if __name__ == "__main__":
    main()
```

问题三

```
# File:问题三 1.py
from sklearn.ensemble import IsolationForest
import pandas as pd
from sklearn.linear_model import LinearRegression
import statsmodels.api as sm

data =pd.read_excel('问题三.xlsx')

# 初始化 Isolation Forest 模型
clf = IsolationForest(contamination=0.1)  # contamination 参数表示预期离群点的比例

# 拟合模型并预测离群点
clf.fit(data)
outliers = clf.predict(data)

# 根据预测结果筛选非离群点
filtered_data = data[outliers == 1]

df = filtered_data
```

```
X = df[['空气质量优良天数比例', '道路交通噪声等效声级']]
Y = df['GDP']
model = LinearRegression()
model.fit(X, Y)


print('模型的参数是:',model.coef_)
print('模型的截距是:',model.intercept_)

#用于模型检验
X2 = sm.add_constant(X)
est = sm.OLS(Y,X).fit()
est.summary()
print(est.summary())

# File:问题三 2.py

from sklearn.ensemble import IsolationForest
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
plt.rcParams['font.family'] = 'SimHei'
data =pd.read_excel('问题三.xlsx')

# 初始化 Isolation Forest 模型
clf = IsolationForest(contamination=0.1)  # contamination 参数表示预期离群点的比例

# 拟合模型并预测离群点
clf.fit(data)
outliers = clf.predict(data)

# 根据预测结果筛选非离群点
filtered_data = data[outliers == 1]

df = filtered_data

X = df[['道路交通噪声等效声级']]
Y = df['GDP']
model = LinearRegression()
model.fit(X, Y)

print('道路交通噪声等效声级与 GDP 线性回归模型的参数是:',model.coef_)
print('道路交通噪声等效声级与 GDP 线性回归模型的截是:',model.intercept_)
```

```
# 绘制散点图
plt.scatter(X, Y, color='blue', label='Actual')

# 绘制回归线
plt.plot(X, model.predict(X), color='red', linewidth=2, label='Regression Line')

# 添加标题和标签
plt.title('道路交通噪声等效声级与 GDP 的线性回归')
plt.xlabel('道路交通噪声等效声级')
plt.ylabel('GDP')

# 显示图例
plt.legend()

# 显示图形
plt.show()


# File:问题三 3.py
from sklearn.ensemble import IsolationForest
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
plt.rcParams['font.family'] = 'SimHei'
data =pd.read_excel('问题三.xlsx')

# 初始化 Isolation Forest 模型
clf = IsolationForest(contamination=0.1)  # contamination 参数表示预期离群点的比例

# 拟合模型并预测离群点
clf.fit(data)
outliers = clf.predict(data)

# 根据预测结果筛选非离群点
filtered_data = data[outliers == 1]

df = filtered_data

X = df[['空气质量优良天数比例']]
Y = df['GDP']
model = LinearRegression()
model.fit(X, Y)
```

```
print('空气质量优良天数比例与 GDP 线性回归模型的参数是:',model.coef_)
print('空气质量优良天数比例与 GDP 线性回归模型的截是:',model.intercept_)

# 绘制散点图
plt.scatter(X, Y, color='blue', label='Actual')

# 绘制回归线
plt.plot(X, model.predict(X), color='red', linewidth=2, label='Regression Line')

# 添加标题和标签
plt.title('空气质量优良天数比例与 GDP 的线性回归')
plt.xlabel('空气质量优良天数比例')
plt.ylabel('GDP')

# 显示图例
plt.legend()

# 显示图形
plt.show()
```