第九章

1.实践案例一

##导入调用所需的库

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

from sklearn.preprocessing import StandardScaler##np.random.seed(42)是用于设置NumPy中随机数生成器的种子函数的调用。通过设置随机种子，可以确保每次运行程序时生成的随机数序列是相同的。这对于调试代码或确保实验结果的可重复性时非常有用的。

np.random.seed(42)

##生成模拟数据：客户年龄、年收入、消费分数，并取整

age = np.concatenate([np.random.normal(25, 5, 100).astype(int),

np.random.normal(40, 5, 100).astype(int),

np.random.normal(60, 5, 100).astype(int)])

income = np.concatenate([np.random.normal(30, 8, 100).astype(int),

np.random.normal(60, 8, 100).astype(int),

np.random.normal(90, 8, 100).astype(int)])

score = np.concatenate([np.random.normal(40, 6, 100).astype(int),

np.random.normal(60, 6, 100).astype(int),

np.random.normal(20, 6, 100).astype(int)])

##创建DataFrame

data = pd.DataFrame({'Age': age, 'Annual Income (k$)': income, 'Spending Score (1-100)': score})

##显示完整生成的数据

print("生成的完整数据：")

print(data)##在进行聚类之前，先对数据进行标准化，使各变量的范围相同，以确保所有特征对模型的影响是等同的。

scaler = StandardScaler()

scaled\_data = scaler.fit\_transform(data)

##将标准化数据转换为DataFrame，方便查看

scaled\_data\_df = pd.DataFrame(scaled\_data, columns=['Age', 'Annual Income (k$)', 'Spending Score (1-100)'])

##输出标准化后的完整数据

print("标准化后的完整数据：")

print(scaled\_data\_df)##使用K-means算法对标准化后的数据进行聚类，首先我们需要确定合适的聚类数量。可通过肘部法则来实现。

##肘部法则通过绘制不同K值的总误差平方和（SSE，Sum of Squared Errors）曲线，帮助识别最佳的聚类数量。SSE是所有点到其所属簇中心距离的平方和，他是评估聚类效果的一个重要指标。通常情况下，在K值增加时，SSE会下降，当下降幅度减小时，即图中出现“肘部”位置时，对应的K值就是合适的聚类数量。

##使用肘部法则

sse = []

k\_range = range(1, 11)

for k in k\_range:

kmeans = KMeans(n\_clusters=k, random\_state=42)

kmeans.fit(scaled\_data)

sse.append(kmeans.inertia\_)

##绘制肘部法则图

plt.figure(figsize=(8, 6))

plt.plot(k\_range, sse, marker='o')

plt.xlabel('Number of clusters (K)')

plt.ylabel('Sum of Squared Errors (SSE)')

plt.title('Elbow Method for Optimal K')

plt.show()

##确定最佳K值并进行聚类

##从肘部图上选择了最佳的K值为3，接下来进行K-均值聚类：

##设置最佳K值

optimal\_k = 3 ##这里假设最佳K值为3

kmeans = KMeans(n\_clusters=optimal\_k, random\_state=42)

data['Cluster'] = kmeans.fit\_predict(scaled\_data)

##输出聚类后的完整数据

print("聚类后的完整数据：")

print(data)##可视化聚类结果

##使用散点图展示每个聚类的分布状况

plt.figure(figsize=(8, 6))

plt.scatter(data['Age'], data['Annual Income (k$)'], c=data['Cluster'], cmap='viridis', s=50)

plt.xlabel('Age')

plt.ylabel('Annual Income (k$)')

plt.title('K-means Clustering Results')

plt.colorbar(label='Cluster')

plt.show()

2.实践案例二

##导入调用所需的库

import pandas as pd

import numpy as np

from sklearn.cluster import KMeans

from sklearn.metrics import silhouette\_score

import matplotlib.pyplot as plt

from sklearn.datasets import load\_iris##加载鸢尾花数据集

data = load\_iris()

iris\_data = pd.DataFrame(data.data, columns=data.feature\_names)

##显示完整数据

print("原始数据：")

print(iris\_data.head())

##轮廓系数法确定最佳K值

silhouette\_scores = []

k\_values = range(2, 11)

for k in k\_values:

kmeans = KMeans(n\_clusters=k, random\_state=42)

cluster\_labels = kmeans.fit\_predict(iris\_data.iloc[:, :-1])

silhouette\_avg = silhouette\_score(iris\_data.iloc[:, :-1], cluster\_labels)

silhouette\_scores.append(silhouette\_avg)

##绘制轮廓系数随K值变化的折线图

plt.figure(figsize=(8, 5))

plt.plot(k\_values, silhouette\_scores, marker='o', linestyle='-', color='b')

plt.title('Silhouette Scores for Different K Values')

plt.xlabel('Number of Clusters (K)')

plt.ylabel('Silhouette Score')

plt.grid()

plt.show()

##输出最佳K值

best\_k = k\_values[np.argmax(silhouette\_scores)]

print(f"最佳的K值为：{best\_k}")

##基于最佳K值进行K-均值聚类

kmeans = KMeans(n\_clusters=best\_k, random\_state=42)

iris\_data['Cluster'] = kmeans.fit\_predict(iris\_data.iloc[:, :-1])

##显示聚类结果

print("\n聚类结果：")

print(iris\_data.head())##可视化聚类结果（以第1和第2个特征为例）

plt.figure(figsize=(8, 6))

for cluster in range(best\_k):

cluster\_data = iris\_data[iris\_data['Cluster'] == cluster]

plt.scatter(cluster\_data.iloc[:, 0], cluster\_data.iloc[:, 1], label=f'Cluster {cluster}')

3.实验一

##导入调用所需的库

import pandas as pd

import numpy as np

from sklearn.cluster import KMeans

from sklearn.metrics import silhouette\_score

import matplotlib.pyplot as plt

from sklearn.datasets import load\_iris##加载鸢尾花数据集

data = load\_iris()

iris\_data = pd.DataFrame(data.data, columns=data.feature\_names)

##显示完整数据

print("原始数据：")

print(iris\_data.head())

##轮廓系数法确定最佳K值

silhouette\_scores = []

k\_values = range(2, 11)

for k in k\_values:

kmeans = KMeans(n\_clusters=k, random\_state=42)

cluster\_labels = kmeans.fit\_predict(iris\_data.iloc[:, :-1])

silhouette\_avg = silhouette\_score(iris\_data.iloc[:, :-1], cluster\_labels)

silhouette\_scores.append(silhouette\_avg)

##绘制轮廓系数随K值变化的折线图

plt.figure(figsize=(8, 5))

plt.plot(k\_values, silhouette\_scores, marker='o', linestyle='-', color='b')

plt.title('Silhouette Scores for Different K Values')

plt.xlabel('Number of Clusters (K)')

plt.ylabel('Silhouette Score')

plt.grid()

plt.show()

##输出最佳K值

best\_k = k\_values[np.argmax(silhouette\_scores)]

print(f"最佳的K值为：{best\_k}")

##基于最佳K值进行K-均值聚类

kmeans = KMeans(n\_clusters=best\_k, random\_state=42)

iris\_data['Cluster'] = kmeans.fit\_predict(iris\_data.iloc[:, :-1])

##显示聚类结果

print("\n聚类结果：")

print(iris\_data.head())##可视化聚类结果（以第1和第2个特征为例）

plt.figure(figsize=(8, 6))

for cluster in range(best\_k):

cluster\_data = iris\_data[iris\_data['Cluster'] == cluster]

plt.scatter(cluster\_data.iloc[:, 0], cluster\_data.iloc[:, 1], label=f'Cluster {cluster}')##导入必要的库

import pandas as pd

import numpy as np

from sklearn.datasets import load\_wine

from sklearn.cluster import KMeans

import matplotlib.pyplot as plt

from sklearn.decomposition import PCA##加载葡萄酒数据集

wine = load\_wine()

wine\_data = pd.DataFrame(wine.data, columns=wine.feature\_names)

##查看数据集信息

print("数据集前5行：")

print(wine\_data.head())##使用肘部法则确定最佳K值

inertia\_values = [] ##存储不同K值下的惯性

k\_values = range(1, 11) ##设置K值范围

for k in k\_values:

kmeans = KMeans(n\_clusters=k, random\_state=42)

kmeans.fit(wine\_data)

inertia\_values.append(kmeans.inertia\_)

##绘制肘部法则图

plt.figure(figsize=(8, 5))

plt.plot(k\_values, inertia\_values, marker='o', linestyle='-', color='b')

plt.title('Elbow Method for Determining Optimal K')

plt.xlabel('Number of Clusters (K)')

plt.ylabel('Inertia')

plt.grid()

plt.show()##基于PCA降维到2D

pca = PCA(n\_components=2)

wine\_data\_2d = pca.fit\_transform(wine\_data)

##基于最佳K值进行K-means聚类（本次实验选取最佳K值为3，可通过图形手动选择最佳K值）

best\_k = 3 ##请根据肘部法则图手动调整

kmeans = KMeans(n\_clusters=best\_k, random\_state=42)

labels = kmeans.fit\_predict(wine\_data\_2d)##可视化聚类结果

plt.figure(figsize=(8, 6))

for cluster in range(best\_k):

cluster\_data = wine\_data\_2d[labels == cluster]

plt.scatter(cluster\_data[:, 0], cluster\_data[:, 1], label=f'Cluster {cluster}')

4.实验二

##导入调用所需的库

import pandas as pd

import numpy as np

from sklearn.cluster import KMeans

from sklearn.metrics import silhouette\_score

import matplotlib.pyplot as plt

from sklearn.datasets import load\_iris##加载鸢尾花数据集

data = load\_iris()

iris\_data = pd.DataFrame(data.data, columns=data.feature\_names)

##显示完整数据

print("原始数据：")

print(iris\_data.head())

##轮廓系数法确定最佳K值

silhouette\_scores = []

k\_values = range(2, 11)

for k in k\_values:

kmeans = KMeans(n\_clusters=k, random\_state=42)

cluster\_labels = kmeans.fit\_predict(iris\_data.iloc[:, :-1])

silhouette\_avg = silhouette\_score(iris\_data.iloc[:, :-1], cluster\_labels)

silhouette\_scores.append(silhouette\_avg)

##绘制轮廓系数随K值变化的折线图

plt.figure(figsize=(8, 5))

plt.plot(k\_values, silhouette\_scores, marker='o', linestyle='-', color='b')

plt.title('Silhouette Scores for Different K Values')

plt.xlabel('Number of Clusters (K)')

plt.ylabel('Silhouette Score')

plt.grid()

plt.show()

##输出最佳K值

best\_k = k\_values[np.argmax(silhouette\_scores)]

print(f"最佳的K值为：{best\_k}")

##基于最佳K值进行K-均值聚类

kmeans = KMeans(n\_clusters=best\_k, random\_state=42)

iris\_data['Cluster'] = kmeans.fit\_predict(iris\_data.iloc[:, :-1])

##显示聚类结果

print("\n聚类结果：")

print(iris\_data.head())##可视化聚类结果（以第1和第2个特征为例）

plt.figure(figsize=(8, 6))

for cluster in range(best\_k):

cluster\_data = iris\_data[iris\_data['Cluster'] == cluster]

plt.scatter(cluster\_data.iloc[:, 0], cluster\_data.iloc[:, 1], label=f'Cluster {cluster}')##导入必要的库

import pandas as pd

import numpy as np

from sklearn.datasets import load\_wine

from sklearn.cluster import KMeans

import matplotlib.pyplot as plt

from sklearn.decomposition import PCA##加载葡萄酒数据集

wine = load\_wine()

wine\_data = pd.DataFrame(wine.data, columns=wine.feature\_names)

##查看数据集信息

print("数据集前5行：")

print(wine\_data.head())##使用肘部法则确定最佳K值

inertia\_values = [] ##存储不同K值下的惯性

k\_values = range(1, 11) ##设置K值范围

for k in k\_values:

kmeans = KMeans(n\_clusters=k, random\_state=42)

kmeans.fit(wine\_data)

inertia\_values.append(kmeans.inertia\_)

##绘制肘部法则图

plt.figure(figsize=(8, 5))

plt.plot(k\_values, inertia\_values, marker='o', linestyle='-', color='b')

plt.title('Elbow Method for Determining Optimal K')

plt.xlabel('Number of Clusters (K)')

plt.ylabel('Inertia')

plt.grid()

plt.show()##基于PCA降维到2D

pca = PCA(n\_components=2)

wine\_data\_2d = pca.fit\_transform(wine\_data)

##基于最佳K值进行K-means聚类（本次实验选取最佳K值为3，可通过图形手动选择最佳K值）

best\_k = 3 ##请根据肘部法则图手动调整

kmeans = KMeans(n\_clusters=best\_k, random\_state=42)

labels = kmeans.fit\_predict(wine\_data\_2d)##可视化聚类结果

plt.figure(figsize=(8, 6))

for cluster in range(best\_k):

cluster\_data = wine\_data\_2d[labels == cluster]

plt.scatter(cluster\_data[:, 0], cluster\_data[:, 1], label=f'Cluster {cluster}')##导入调用所需的库

import pandas as pd

from sklearn.cluster import KMeans

from sklearn.preprocessing import StandardScaler

import matplotlib.pyplot as plt##读取 Excel 文件

file\_path = r"D:\yzgj.xlsx" ##文件路径

data = pd.read\_excel(file\_path)

##查看数据

print("全部数据预览：")

print(data)##假设选择数据的某几列进行聚类

columns\_for\_clustering = ['Per capita GDP','urban（%）','Aging（%）']

cluster\_data = data[columns\_for\_clustering].dropna()

##在进行聚类之前，先对数据进行标准化，使各变量的范围相同，以确保所有特征对模型的影响是等同的。

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

scaled\_data = scaler.fit\_transform(cluster\_data)

##将标准化数据转换为DataFrame，方便查看

scaled\_data\_df = pd.DataFrame(scaled\_data, columns=['Per capita GDP','urban（%）','Aging（%）'])

##输出标准化后的完整数据

print("标准化后的完整数据：")

print(scaled\_data\_df)

##使用K-means算法对标准化后的数据进行聚类，首先我们需要确定合适的聚类数量。

##使用肘部法则，图中出现“肘部”位置时，对应的K值就是合适的聚类数量。

inertia = []

k\_values = range(1, 11)

for k in k\_values:

kmeans = KMeans(n\_clusters=k, random\_state=42)

kmeans.fit(scaled\_data)

inertia.append(kmeans.inertia\_)

##绘制肘部法曲线

plt.figure(figsize=(8, 5))

plt.plot(k\_values, inertia, marker='o')

plt.title('Elbow Method for Optimal K')

plt.xlabel('Number of Clusters (K)')

plt.ylabel('Inertia')

plt.show()

##从肘部图上选择了最佳的K值为3，接下来进行K-均值聚类：

optimal\_k = 3

kmeans = KMeans(n\_clusters=optimal\_k, random\_state=42)

cluster\_labels = kmeans.fit\_predict(scaled\_data)

##将聚类结果添加到原始数据

data['Cluster'] = cluster\_labels

print("聚类后的完整数据：")

print(data)

##可视化聚类结果

##使用散点图展示每个聚类的分布状况

plt.figure(figsize=(10, 6))

colors = ['red', 'blue', 'green']

for cluster in range(optimal\_k):

cluster\_points = data[data['Cluster'] == cluster]

plt.scatter(

cluster\_points['Per capita GDP'],

cluster\_points['Aging（%）'],

label=f'Cluster {cluster}',

color=colors[cluster]

)

plt.title('K-Means Clustering Visualization')

plt.xlabel('Per capita GDP')

plt.ylabel('Aging（%）')

plt.legend()

plt.grid()

plt.show()