第十一章 数据探查建模实验

通过数据处理，挖掘其背后隐藏的信息是数据处理的根本目的。根据数据处理的任务，大体分类两类，一类是以探查数据的特征，寻找数据符合某种规律为任务的数据处理称为数据探查建模；一种是根据已知样本数据特征，构建数据模型，实现对未知样本数据的预测称为数据预测建模。也就是获得一批数据后，希望对这批数据寻找到某种规律性，就属于探查建模，已知一批数据的规律性，希望对于未知数据进行预测，这就是预测建模。它们也称为无模型训练和有模型训练。本章实验主要讨论探查建模实验，包括聚类分析和关联分析等，编写过程中参考了文献[20,22]和文献[23]，也参考了网上的一些短文材料。

第一节 聚类分析

获得一批数据，对这批数据进行简化描述的最好手段就是聚类，也就是根据数据的指标特征，分析哪些数据具有相似的特征，从而归为一类，这样就将纷繁复杂的数据通过简单的归类进行简化，同时也把握了其本质特征。比如刚入学的大学新生，学校希望进行归类，比如根据英语成绩、数学成绩和兴趣爱好等进行归类，也就是将有相同特长和兴趣爱好的学生进行归类，然后开展有针对性的教学，这就是聚类分析。

根据聚类是否指定聚类数目，可以将聚类分为两种，一种是层次聚类，一种是划分聚类。层次聚类就是最初每一个样本都是一类，然后根据给定标准进行两两合并，一直到所有的样本都聚为一类为止；而划分聚类是根据给定聚类标准，将样本归类为给定数目。例如有各种鱼、禽和肉类食品，如果根据食品中的一些营养标准，就可以这些食品聚为几个类别。

一、聚类的步骤

对于一批数据进行聚类，显然需要讨论聚类的标准，没有标准就没有办法进行归类，因此在聚类的时候需要对数据进行一系列的处理。

1. 选择合适的变量

我们对于一批样本，它可能有许多指标取值，比如上面谈到的鱼和肉等食品，它们所含的营养指标：氨基酸，维生素，蛋白质，糖类，铁和钠等含量。每一个样本都在这些指标上取值，如果我们对这些样本进行聚类，目的是关注营养成分，可能我们只能选择氨基酸和蛋白质以及糖类等作为聚类指标，舍弃其它非营养成分的含量。

2. 数据标准化

由于不同指标，量纲不同，有的指标可能取值范围为 0 到 1 之间，有的取数百万大小，这样一起聚类，可能取值较大的指标淹没了取值较小的指标，造成聚类效果不佳，因此可以统一使用标准化等手段，将所有指标取值都标准化到 0 和 1 之间；

3. 异常点的处理

某些指标可能由于误差或者其它因素，取值异常，这就可能对聚类效果造成干扰。

4. 计算样本之间的距离

两个样本是否可以归为一类，总要有一个标准，这个标准就是衡量两个样本之间的距离大小，距离越大越不能归为一类，因此这个距离的选取十分关键和重要，目前有很多距离计算方法，比如欧氏距离，非对称二元距离等等，Python 语言有一个 scipy.spatial.distance.dist()函数，提供很多计算指标距离的方法

5. 聚类算法的确定

是选择层次聚类还是选择划分聚类，不同的方法有不同的适应范围，比如层次聚类适合小样本的聚类。

6. 确定类的数目

最终聚类成几类，需要确定类的数目，如果给定一个标准，也就是指标距离足够大，那么所有的样本都可能归为一类。因此给定距离或者给定聚类数目，都可以将给定样本进行归类。

7. 可视化结果

将聚类结果做可视化输出，比如层次聚类就可以使用树状图显示出来

8. 聚类结果的解读

对于聚类结果，可以观察哪些样本聚为一类，比如根据一些营养指标，将某些鱼类和禽类聚为一类，那么我们可以解读出，这几种食品具有相同的营养价值。

9. 验证结果

验证结果讨论的是聚类的稳定性问题，也就是给一批同样性质的样本，它们是否也会输出相近似的聚类结果。如果不能，就说明聚类模型不可靠，需要改进。

二、层次聚类分析

这种聚类方法是通过迭代步骤实现的。刚开始每一个样本都可以看为一类，然后计算两两样本之间的距离，距离最小的聚为一类，这样得到新的样本分类，然后再每两个类计算距离，将距离小的再聚为一类，这样持续下去直到所有的样本都聚为一类为止。这里面要注意的是如果两个样本已经聚为一类了，那么就必须立刻作为一个整体参与和其它样本或者样本类计算距离。

这里面就有一个问题，那就是计算两个样本之间距离很简单，比如使用欧氏距离就可以了，但是计算两个类之间的距离怎么办了呢？很显然也要通过计算一个类的样本和另外一个类的样本之间的距离来表示。这时候就有很多方法，比如单联动计算方法就是两个类中样本的最小距离；全联动法就是一个类中样本和另外一个类中样本的最大距离；平均联动就是两个类所有点之间的平均距离；质心法就是两个类质心的距离；Ward 法：就是两个类之间所有变量的方差的平方和。

层次聚类方法在 Python 语言中可以使用 scipy.cluster.hierarchy. linkage()函数来实现，这个函数的格式是 linkage(d, method=)，其中 d 是通过 dist()函数生成的距离矩阵，method 可以取值为 single，complete，average，centroid 和 ward 等。对应的就是上面的单联动方法和全联动方法等等。本教材提供一个数据集为 nutrient，是化验各种食品的热量、蛋白质、脂肪、含钙量和含铁量得到的数据样本，现在要对这些样本进行聚类，具体实现代码如下，柱状聚类图如图 14.1 所示。

```
import readDataFromWugangTOP as wt
url = "https://wugang69.top/dataset/nutrient.csv"
nutrient = wt.read_csv_with_headers(url)

import matplotlib.pyplot as plt
from scipy.cluster.hierarchy import linkage, dendrogram
from scipy.spatial.distance import pdist

# 1 提取食物名称（第一行）
labels = nutrient.iloc[:, 0].str.lower().values

# 2 提取数值数据（去掉第一列）
nutrient_numeric = nutrient.iloc[:, 1:]

from sklearn.preprocessing import StandardScaler
# 标准化
scaler = StandardScaler()
nutrient_scaled = scaler.fit_transform(nutrient_numeric)

# 3 计算距离矩阵（欧氏距离）
dist_matrix = pdist(nutrient_scaled, metric='euclidean')

# 4 层次聚类
Z = linkage(dist_matrix, method='complete')  # 'ward','complete','average','single'

# 5 绘制树状图
plt.figure(figsize=(12, 6))
dendrogram(Z, labels=labels, leaf_rotation=90, leaf_font_size=10)
plt.title("Hierarchical Clustering Dendrogram (nutrient)")
plt.xlabel("Foods")
plt.ylabel("Distance")
plt.tight_layout()
plt.show()
```

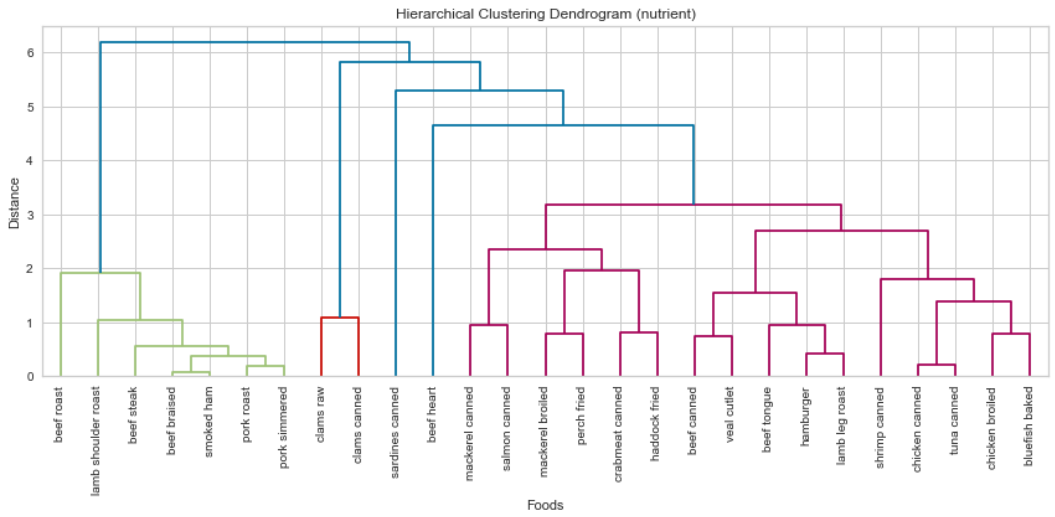

图 14.1 食品数据的平均联动聚类

图 14.1 从下面向上看，我们看所有食品中，距离最小的两个样本是 beef braised 和 smoked ham，它们当然合并在一起，接着计算发现 pork roast 和 pork simmered 距离最小，因此合并为一类，这两类和其它所有样本再两两计算距离，最小的在合并为一类，依次进行下去，就会得到如图 14.1 所示的树状图。注意计算两个类之间的距离使用的平均联动方法。分析图 14.1，我们可以观察哪些食品营养价值是较为相似的，哪些存在很大不同，进而给客户提供购买意见。

如果希望根据层次聚类将数据分为指定类别，可以使用下面的语句

```
clusters = fcluster(Z, t=3, criterion='maxclust')
```

上面 clusters 就是每一个样本对应的类号，Z 就是层次聚类返回的结果，t=3 表示指定分为三类，criterion="maxclust"表示强制切为 3 类，也就是强制切为 3 个簇。

三、划分聚类分析

划分聚类分析主要分为两种，一种是 K-均值聚类，一种是基于中心点的划分聚类。

1. K 均值聚类

(1) K 均值聚类具体步骤

随机选择 K 个样本点作为中心点；将每一个样本点归类到距离它最近的中心点；重新计算得到的 K 个类的中心点；将每一个样本点重新归类到距离它最近的中心点；重复上面的步骤，直到样本不再被重新分配或者达到最大迭代次数。

(2) K 均值聚类的优缺点

K 均值可以处理比层次聚类更大的数据集，缺点是要求变量是连续变化的，同时受异常值影响大，不太适用于非凸数据集。

(3) Python 语言的实现

由于 K 均值聚类需要指定聚类数目，指定不合适的聚类数目会使得聚类效果很差，因此 Python 语言的包 yellowbrick.cluster 提供了 KElbowVisualizer ()用来探查数据究竟分为几类较为合适。KElbowVisualizer ()的调用形式为 KElbowVisualizer(KMeans(), k=(2,12), metric='silhouette')，其中 k=(2,12)表示类别在 2 到 12 中选取。该函数提供用来衡量数据集聚为 n 类的得分。下面以 wine 数据集来实验 K 均值聚类的代码，其中特别注意的是 wine 数据集中第一列不是 wine 的指标数据，而是需要去除。另外特别注意要进行标准化，因为计算欧氏距离等如果不进行标准化会造成数据指标大的掩盖数据指标小的信息。标准化后一般均值为 0，方差为 1。

```
import readDataFromWugangTOP as wt
url = "https://wugang69.top/dataset/wine.csv"
wine = wt.read_csv_with_headers(url)

# 如果没有安装，需要在 console 中运行 pip install yellowbrick
from yellowbrick.cluster import KElbowVisualizer
```

```
from sklearn.cluster import KMeans

wineData = wine.iloc[:,1:] # 将索引号和样本类型去掉，注意 dataframe 数据类型从 0 编号
# 用 KMeans + Elbow 方法找最佳簇数
from sklearn.preprocessing import StandardScaler
# 标准化
scaler = StandardScaler() # 标准化，均值为 0，方差为 1。
wine_scaled = scaler.fit_transform(wineData)

# 用 KMeans + Elbow 方法找最佳簇数
model = KMeans(random_state=42)
visualizer = KElbowVisualizer(model, k=(2,12), metric='silhouette')

visualizer.fit(wine_scaled)  # 注意这里用的是标准化后的数据
visualizer.show()
```

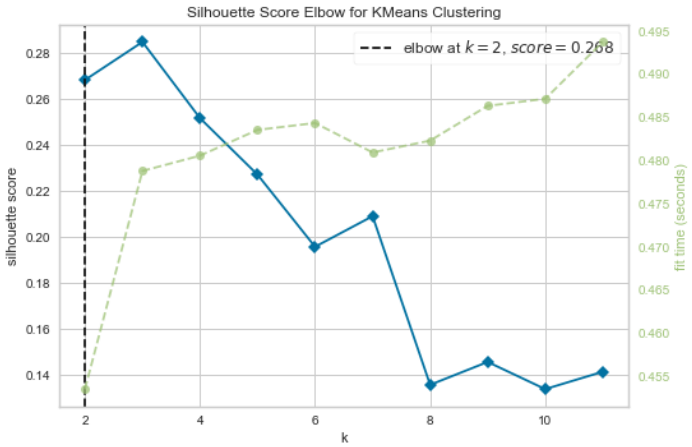

图 14.2 酒类数据集推荐的聚类个数

图 14.2 是生成的聚类数目和具体得分图，通过该图观察，可以发现聚为 3 类得分最高，聚为 2 类得分次之，因此在实际运算的时候，事实上，该 wine 数据集的确是分为三类的。下面是使用 Python 语言提供的 sklearn.cluster.KMeans()聚类函数，选择聚为 3 类针对 wine 数据集进行聚类的代码。

```
# 下面是具体 K 均值聚类
from sklearn.metrics import accuracy_score, confusion_matrix
from scipy.stats import mode
import numpy as np
import pandas as pd
# 提取分类号
y_true = wine["Type"].values
n_clusters = 3


# 3. KMeans 聚类
kmeans = KMeans(n_clusters=3, random_state=42, n_init=10)
y_pred = kmeans.fit_predict(wine_scaled)

# ==========特别注意，下面是类号对齐，已经在 wugangFunction 里面写成了函数==========
labels_map = {}
for i in range(n_clusters):
    mask = (y_pred == i)
    labels_map[i] = mode(y_true[mask], keepdims=True).mode[0]

y_pred_mapped = np.array([labels_map[c] for c in y_pred])

# ------------------------------
# 4. 输出准确率
acc = accuracy_score(y_true, y_pred_mapped)
print("KMeans 聚类准确率: {:.2f}%".format(acc*100))

# ------------------------------
# 5. 输出聚类中心点
print("\n 聚类中心点 (Centroids):")
centers =pd.DataFrame(kmeans.cluster_centers_, columns=wine.columns[1:]) # 功能是加上指标名称
print(centers)

# ------------------------------
# 6. 混淆矩阵 & 错误分类统计
```

```
cm = confusion_matrix(y_true, y_pred_mapped, labels=[1,2,3])
print("\n 混淆矩阵 (行:真实类别, 列:预测类别):\n", cm)

print("\n 每一类的错误分类情况：")
for i, true_class in enumerate([1,2,3]):
    total = cm[i,:].sum()
    correct = cm[i,i]
    errors = total - correct
    if errors > 0:
        print(f"真实类别 {true_class}: 错 {errors} 个")
        for j, pred_class in enumerate([1,2,3]):
            if j != i and cm[i,j] > 0:
                print(f"   → 被分到 类别 {pred_class}: {cm[i,j]} 个")
```

K 均值聚类的结果显示分类准确率为 96.63%，因此分类的准确率较高，混淆矩阵显示大多数数据正确的分类。具体运行结果如下。

```
KMeans 聚类准确率: 96.63%

聚类中心点 (Centroids):
    Alcohol     Malic       Ash  ...       Hue  Dilution   Proline
0  0.164907  0.871547  0.186898  ... -1.164789 -1.292412 -0.407088
1  0.835232 -0.303810  0.364706  ...  0.473984  0.779247  1.125185
2 -0.926072 -0.394042 -0.494517  ...  0.461804  0.270764 -0.753846

混淆矩阵 (行:真实类别, 列:预测类别):
 [[59  0  0]
 [ 3 65  3]
 [ 0  0 48]]

每一类的错误分类情况：
真实类别 2: 错 6 个
   → 被分到 类别 1: 3 个
   → 被分到 类别 3: 3 个
```

2. 基于中心点的划分聚类(PAM)

K 均值聚类是基于样本均值聚类，也就是根据距离样本中心点的距离来聚类，很显然如果有一个非常大的异常点，就会使得中心点严重偏离大多数样本点，这就使得 K 均值聚类变得不稳定了，因此提出基于中心点的划分聚类。

PAM 与 K-means 相比，PAM 可以使用任意度量来计算两个样本的距离，因此可以容纳混合数据，不限于连续变量，另外就是 PAM 算法和 K-means 算法根本区别是 PAM 选择一个最具有代表性的样本来代替 K-means 的中心点，因此 PAM 稳健性更强。很显然选择代表性的点需要不停的迭代计算，具体算法的思想就是不再使用 K-means 的中心点计算，而是任意选一个点作为中心点，然后计算所有点和该点距离的和，再选另外一个点作为中心点，再计算这种距离和，比较大小，距离和小的点作为代表点，重复计算下去，直到收敛或者达到指定次数为止。通过这种算法，可以观察到 PAM 计算量很大，收敛速度较慢。读者可以借助人工智能编写代码实现对 wine 的聚类。

3. 划分聚类的可视化

划分聚类对于变量数目超过 3 个的数据集，很难可视化表示出聚类的情况，但是如果使用主成分分析方法，将最大的两个主成分指标作为平面坐标系的两个坐标轴，倒是可以可视化聚类的效果，可以使用 Python 语言实现该功能，具体代码如下：

```
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
from matplotlib.patches import Ellipse

# ------------------------------
# 4. PCA 降维到 2D
pca = PCA(n_components=2)
X_pca = pca.fit_transform(wineData)
centers_pca = pca.transform(kmeans.cluster_centers_)

# ------------------------------
# 5. 可视化 + 椭圆
plt.figure(figsize=(8,6))
colors = ['red', 'green', 'blue']
```

```
for i in range(1, n_clusters+1):
    mask = (y_pred_mapped == i)
    cluster_points = X_pca[mask]
    plt.scatter(cluster_points[:,0], cluster_points[:,1],
                label=f'Cluster {i}', alpha=0.6, s=50, c=colors[i-1])

    # 计算协方差矩阵
    cov = np.cov(cluster_points, rowvar=False)
    # 特征值分解得到椭圆轴长度和方向
    vals, vecs = np.linalg.eigh(cov)
    order = vals.argsort()[::-1]
    vals = vals[order]
    vecs = vecs[:, order]

    angle = np.degrees(np.arctan2(*vecs[:,0][::-1]))
    width, height = 2 * np.sqrt(vals)  # 1 std 对应的半轴
    ellipse = Ellipse(xy=cluster_points.mean(axis=0), width=width*2, height=height*2,
                      angle=angle, edgecolor=colors[i-1], fc='None', lw=2, linestyle='--')
    plt.gca().add_patch(ellipse)

# 绘制聚类中心
plt.scatter(centers_pca[:,0], centers_pca[:,1],
            marker='X', s=200, c='black', label='Centroids')

plt.title('KMeans Clustering of Wine Dataset (PCA 2D) with Ellipses')
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
plt.legend()
plt.grid(True)
plt.show()
```

如图 14.3 所示，注意其中的椭圆是人为绘制的，是包含同一类样本的最小椭圆。对于 K-means 聚类的结果，

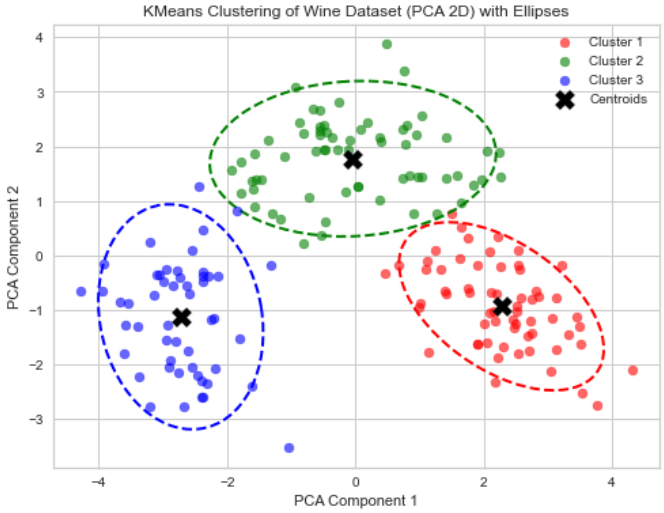

图 14.3 酒类数据集 PAM 聚类的两个主成分可视化图形

第二节 关联分析

关联分析就是计算事务中一些项同时出现的频率。这在现实生活中很常见，比如图书馆借书，每个学生一次借书就是一个事务，一次借书事务中一共借了几本书，就是几项。关联分析就是在一段时间内，分析哪些书是一起借阅的，了解这些信息就可以为图书馆里人员排列书架等提供依据，因此关联分析也是一种从数据集中探查建模的数据处理方法，具体就是探查出事务数据集中的关联规则，寻找哪些项目是有联系的。关联分析寻找事务集中的强关联规则通常使用 Apriori 算法，其中需要两个指标，一个是支持度，一个是置信度，比如对于关联规则 X->Y，支持度表示同时包含 X 和 Y 的事务占总事务的比例，置信度指的是同时包含 X 和 Y 的事务在所有包含 X 事务的占比。通过支持度和置信度来确定关联规则。

关联分析只能针对类别型的数据探查关联规则，连续型数据需要离散化，Python 语言中使用 mlxtend.frequent_patterns.apriori()进行关联分析，使用该模块下的 association_rules 给出关联规则。使用该函数的时候，需要在 spyder 的 console 窗口中运行 pip install mlxtend。下面举例一般事务集如何进行关联分析。

假设有事务数据为

```
transactions = [
    ["a","b","c"],
    ["a","b"],
    ["a","b","d"],
    ["c","e"],
    ["a","b","d","e"]
]
```

1.导入必要库，如果没有安装 mlxtend 则需要首先安装

```
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori, association_rules
```

2.将事务数据转换为 One-hot 编码

apriori 函数要求输入 0/1 的 DataFrame，表示每个事务中是否包含某个项目。

```
te = TransactionEncoder()
te_ary = te.fit(transactions).transform(transactions)
df = pd.DataFrame(te_ary, columns=te.columns_)
print("One-hot 编码后的数据:")
print(df)
```

输出示例：

```
       a      b      c      d      e
0   True   True   True  False  False
1   True   True  False  False  False
2   True   True  False   True  False
3  False  False   True  False   True
4   True   True  False   True   True
```

3.挖掘频繁项集

使用 apriori 挖掘频繁项集，设置最小支持度 min_support。

```
min_support = 0.4  # 例如最小支持度为 0.4
frequent_itemsets = apriori(df, min_support=min_support, use_colnames=True)
print("\n 频繁项集:")
print(frequent_itemsets)
```

4.生成关联规则

使用 association_rules 函数生成关联规则，可以设置最小置信度 min_threshold。

```
min_confidence = 0.7  # 例如最小置信度为 0.7
rules=association_rules(frequent_itemsets,metric="confidence", min_threshold=min_confidence)
print("\n 关联规则:") # 显示规则及其指标
print(rules[['antecedents','consequents','support','confidence','lift']])
```

通过以上几个步骤，即可以获得关联规则，具体代码就是上面各个步骤的组合，运行的结果如下：

```
频繁项集:
   support   itemsets
0      0.8        (a)
1      0.8        (b)
2      0.4        (c)
3      0.4        (d)
4      0.4        (e)
5      0.8     (a, b)
6      0.4     (a, d)
7      0.4     (d, b)
8      0.4  (a, d, b)

关联规则:
  antecedents consequents  support  confidence  lift
0         (a)         (b)      0.8         1.0  1.25
```

1	(b)	(a)	0.8	1.0	1.25
2	(d)	(a)	0.4	1.0	1.25
3	(d)	(b)	0.4	1.0	1.25
4	(a, d)	(b)	0.4	1.0	1.25
5	(d, b)	(a)	0.4	1.0	1.25
6	(d)	(a, b)	0.4	1.0	1.25

可以发现 a 和 b 互相推导，(a,d)可以推导出 b，它们置信度分别是 0.8 和 0.4。如果上面是销售购物票的话，很显然商品 a 和 b 应该进行交叉销售，或者定制一个套餐。

第三节 综合实验十一

一、实验目的

1. 了解数据探查建模的基本思想
2. 掌握聚类分析中 K-means 和 PAM 聚类的 Python 语言实现，理解相关函数中的各种参数
3. 掌握关联分析的 Python 语言实现，能够具体应用

二、实验平台和软件

1. 互联网平台
2. 计算机系统
3. spyde 环境和 Python 语言编译系统

三、实验内容和步骤

1. 层次聚类分析

(1) 获取 nutrient 数据集
(2) 阅读 nutrient 数据集的特点
(3) 对该数据集进行标准化并计算样本距离
(4) 对计算的距离矩阵进行层次聚类
(5) 绘制层次聚类图
(6)分析该聚类结果的含义

2. K 均值聚类

(1) 获取 wine 数据集
(2) 了解该数据集的特点
(3) 标准化该数据集
(4) 使用 KelbowVisualizer()函数探查该数据集聚几类最为合适
(5) 使用 KMeans()函数进行聚类
(6) 将聚类结果和实际分类比较，生成混淆矩阵，分析结果
(7) 了解 K-means 聚类的效果

3. 关联分析

(1) 安装 mlxtend 包
(2) 构造一个项目集
(3) 使用 apriori()函数对该数据集进行关联分析
(4) 使用 association_rules()函数了解关联规则
(5) 分析和解释一些规则的含义

四、实验报告 （总分 8 分）

1. 下载 iris 数据集，分析说明该数据集各指标含义，探查聚类的合理数目，然后使用 K-means 进行聚类，并分析聚类的有效性，最后可视化聚类结果。(2 分)

2. 使用层次聚类方法对 iris 进行聚类，绘制层次聚类图，并和 K-means 聚类的分类结果进行比较，分析各自特点。 (3 分)

3. 下面是某个超市购物记录，使用关联分析探查一些关联规则，并给出支持度和置信度。(3 分)

交易时间	购买商品
14：25	i1，i2，i4
15：07	i1，i2，i3
16：33	i2，i3
17：05	i1，i3
18：40	i1，i2，i3，i5
18：55	i2，i3
19：26	i1，i2，i5
19：52	i2，i4
20：03	i1，i2，i3
20：16	i1，i3