人工智能实验报告

实验二　传统机器学习方法

学院：计算机与通信工程　　专业：计算机科学与技术　　班级：计 221

姓名：乔彦博　　学号：U202242223　　日期：2025.5.9

实验目标

1. 掌握传统机器学习的基本流程和方法，包括熟悉数据预处理、特征工程、模型训练与评估展示等完整流程；理解传统机器学习算法的核心原理（如分类、回归、聚类等）。
2. 通过监督学习和无监督学习实验，理解不同算法的应用场景和效果评估。监督学习例如分类算法（如 KNN、决策树、SVM 等），对比分类性能；无监督学习例如聚类算法（如 K-Means、层次聚类等），分析聚类效果。
3. 通过可视化展示，深入理解数据和模型，包括：
   - 数据可视化：使用散点图、箱线图、平行坐标图等展示数据集特征分布和类别差异
   - 模型可视化：对监督学习模型，绘制决策边界（如决策树、SVM）或特征重要性（如随机森林）；对无监督学习模型，展示聚类结果（如 K-Means 的聚类中心、聚类分布）等
   - 效果可视化：使用混淆矩阵热力图、ROC 曲线等展示分类模型的性能；使用肘部法则图、轮廓系数图等展示聚类模型的效果

实验内容

1. 任务分析

本实验主要针对传统机器学习方法的应用与实践，包含两大核心任务：

1. **监督学习任务**：选择合适的分类数据集，实现并对比多种分类算法（如 KNN、决策树、SVM 等）的性能。需要完成数据预处理、特征选择、模型训练、参数调优及结果评估等步骤。通过混淆矩阵、精确率、召回率、F1 分数等指标对各算法效果进行全面评估。
2. **无监督学习任务**：选择适合聚类的数据集，实现 K-Means、层次聚类等算法，并通过轮廓系数、肘部法则等方法评估聚类效果。重点分析不同聚类算法的特点及适用场景。

两项任务均要求通过可视化技术增强对数据特征和算法性能的直观理解，包括数据分布可视化、决策边界可视化和评估指标可视化等。

2. 实现工具

本实验采用以下工具和技术栈：

- **编程语言**：Python 3.11.12，采用 3.11 最新版本，安全性更好
- **数据处理库**：
  - NumPy：用于高效的数值计算和数组操作
  - Pandas：用于数据加载、清洗和预处理
- **机器学习库**：
  - Scikit-learn：提供各种监督学习和无监督学习算法实现
  - SciPy：用于科学计算的高级函数
- **可视化工具**：
  - Matplotlib：基础绘图库，用于生成各类统计图表
  - Seaborn：基于 Matplotlib 的统计数据可视化库，用于生成更美观的图表
  - Plotly：交互式可视化库（可选）
- **开发环境**：Jupyter Notebook，便于代码与结果展示的交互式环境

3. 实现方案

本实验的整体实施方案如下：

1. **数据准备阶段**
   - 监督学习：选择 UCI 机器学习仓库中的 Iris 数据集和 Wine 数据集
   - 无监督学习：使用相同数据集但不使用标签信息
   - 数据分析：统计特征、相关性分析、缺失值处理
   - 数据可视化：绘制特征分布图、散点矩阵、箱线图等
2. **监督学习实验**
   - 数据预处理：特征标准化、训练集与测试集划分（比例 7:3）
   - 模型实现：KNN（K=1,3,5,7）、决策树、随机森林、SVM（线性和 RBF 核）
   - 模型评估：交叉验证、网格搜索最优参数
   - 结果分析：计算准确率、精确率、召回率，绘制混淆矩阵、ROC 曲线
   - 可视化：绘制决策边界、特征重要性图表
3. **无监督学习实验**
   - 数据预处理：特征标准化、降维（PCA）用于可视化
   - 模型实现：K-Means（不同 K 值）、层次聚类（不同距离度量和链接方式）
   - 最优聚类数确定：绘制肘部法则图、轮廓系数分析
   - 聚类结果评估：与真实标签对比（如有）、聚类内/间距离计算
   - 可视化：聚类结果散点图、聚类中心、层次聚类树状图

4. **结果分析与比较**

   - 不同监督学习算法性能对比分析
   - 不同无监督学习算法聚类效果对比
   - 算法优缺点总结与适用场景分析

4. 实现内容

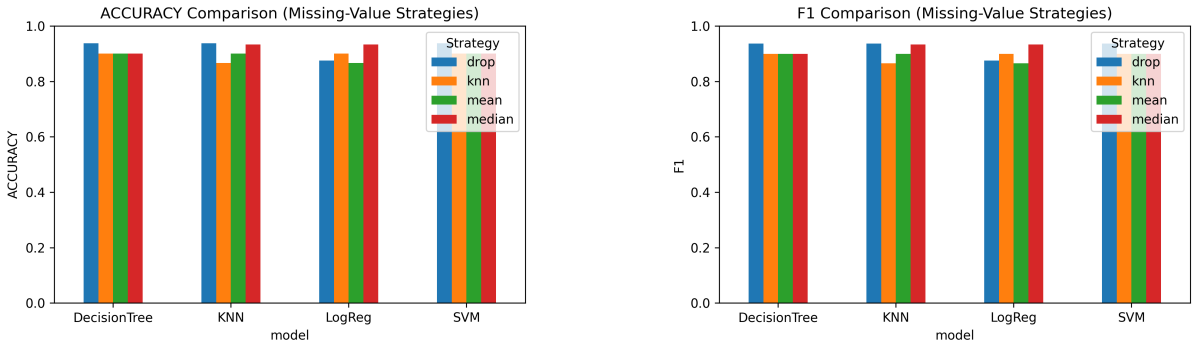
4.1 监督学习模块（`task1.py`）

1) **数据载入与缺失值注入**
读取原始 *Iris* 数据集后，以随机比例 ratio $= 0.1$ 将四个特征列注入缺失值，用于验证不同缺失值处理策略的鲁棒性。

2) **缺失值处理策略**
采用 *drop*、*mean*、*median* 与 *knn* 四种策略，对训练/测试集分别处理并保持标签一致。

3) **特征缩放**
填补或删除后使用 `StandardScaler` 归一化。

4) **模型训练与评估**
在四种策略下分别训练 **LogReg**、**Decision Tree**、**SVM (RBF)** 与 **KNN**；记录 *Accuracy* 与 *Weighted* F_1，并保存混淆矩阵。



(a) Accuracy　　(b) Weighted F_1

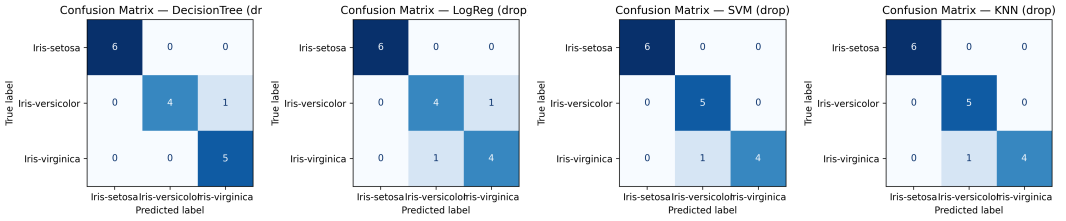
图 1: 不同缺失值处理策略下四类模型的总体性能比较

4.2 无监督学习模块（`task2.py`）

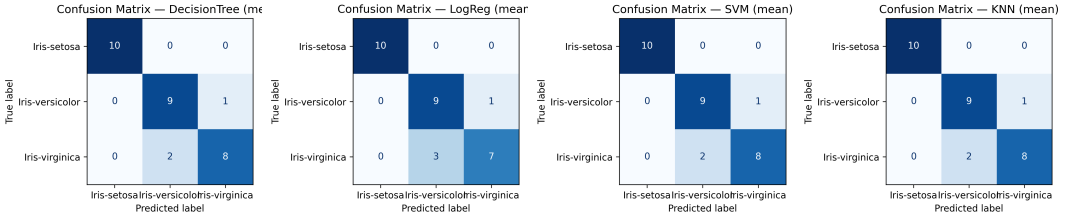
1) **预处理与标准化**
对 *Mall Customers* 数据集进行三倍 IQR Winsorization、Label Encoding 与 `StandardScaler`。

2) **降维可视化**
使用 **PCA** 与 **t-SNE** 将特征映射到二维平面（见图 6）。

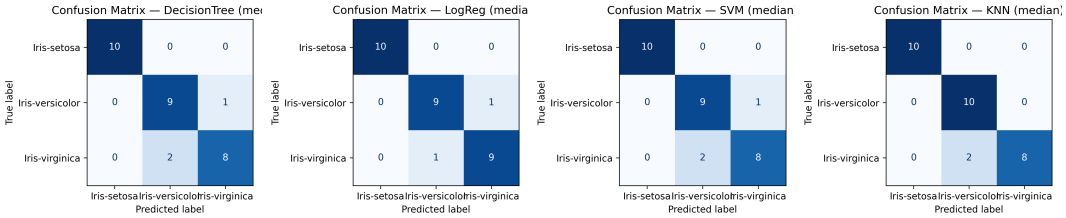
3) **最优簇数确定**
通过 Elbow 与 Silhouette 曲线确定最佳簇数 $k^* = 6$（图 10）。



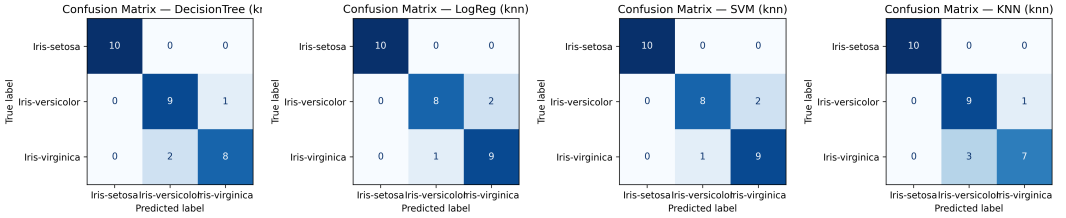
(a) drop–DecisionTree (b) drop–LogReg (c) drop–SVM (d) drop–KNN



(e) mean–DecisionTree (f) mean–LogReg (g) mean–SVM (h) mean–KNN



(i) median–DecisionTree (j) median–LogReg (k) median–SVM (l) median–KNN

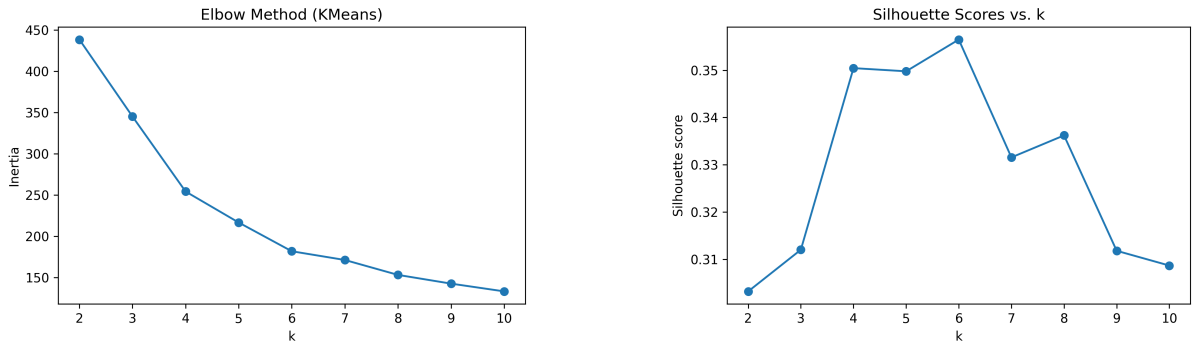


(m) knn–DecisionTree (n) knn–LogReg (o) knn–SVM (p) knn–KNN

图 2: 四种缺失值策略与四类分类器的混淆矩阵（共 16 张）

4) **聚类算法比较**

在 k^* 下比较 **K-Means**、**Agglomerative** 与 **DBSCAN** 的内部指标。



(a) Elbow　　(b) Silhouette

图 3: K-Means 不同 k 下的惯性与轮廓系数

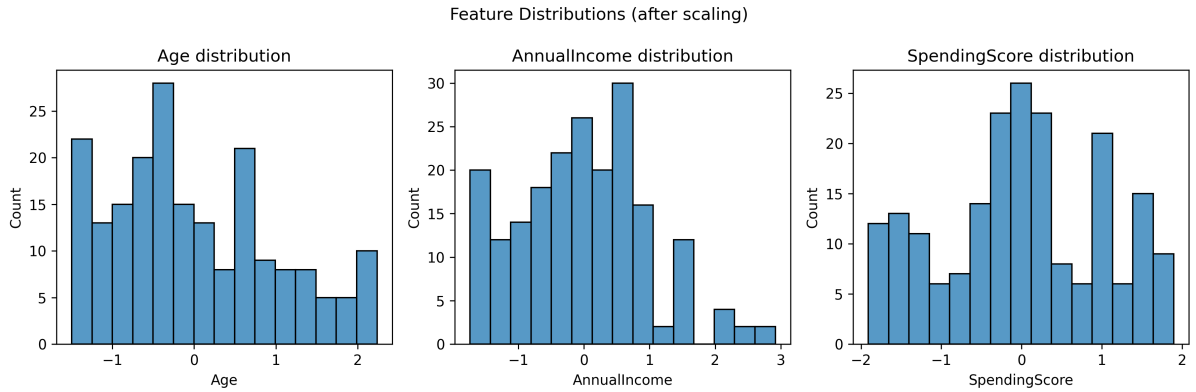


图 4: Age, Annual Income, Spending Score 的标准化直方图

4.3 代码与文件结构

- 根目录含 `task1.py` 与 `task2.py`，对应监督/无监督实验。
- 生成的图片分别保存于 `./figures` 与 `./figures2`，脚本已自动创建文件夹。
- LaTeX 文档 `report.tex` 位于 `LAB2/`，编译后生成 `report.pdf`；所有图片使用相对路径引用。

4. 实现内容

4.1 监督学习模块（`task1.py`）

1) **数据载入与缺失值注入**

读取原始 Iris 数据集后，以随机比例 ratio $= 0.1$ 将四个特征列注入缺失值，用于验证不同缺失值处理策略的鲁棒性。

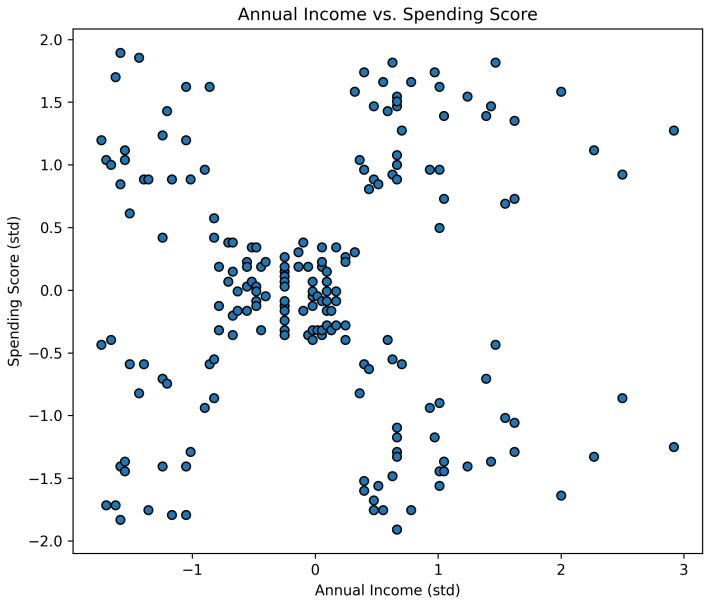
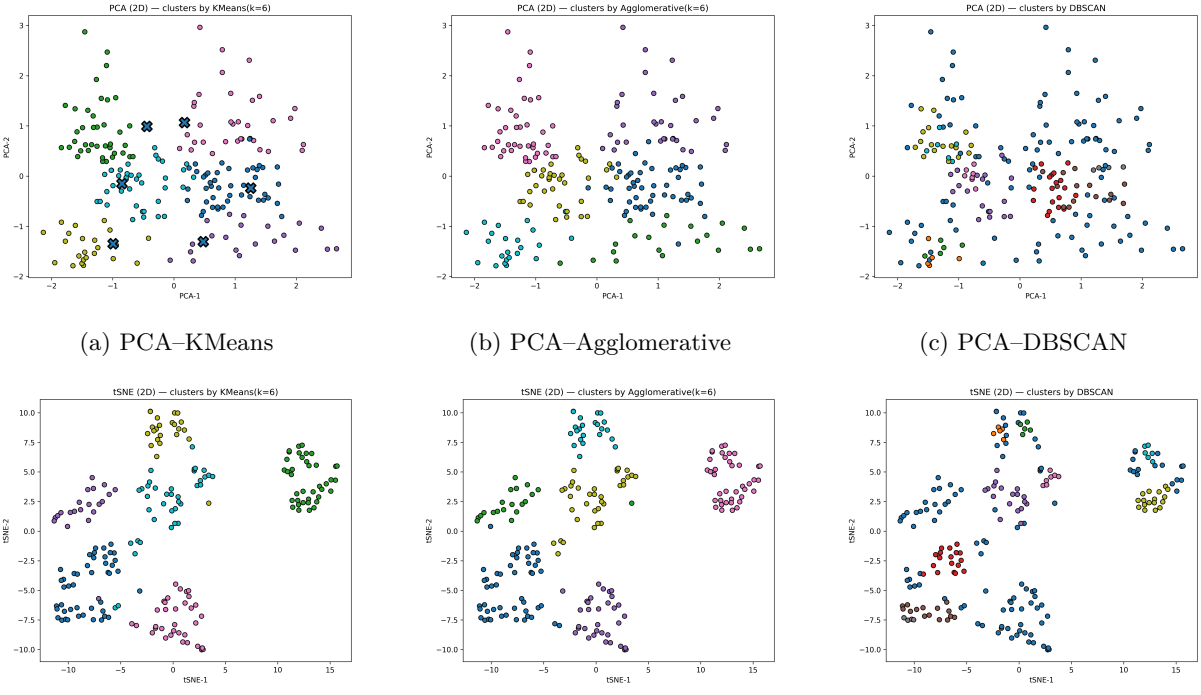


图 5: Annual Income 与 Spending Score 散点图



(a) PCA–KMeans (b) PCA–Agglomerative (c) PCA–DBSCAN

(d) t-SNE–KMeans (e) t-SNE–Agglomerative (f) t-SNE–DBSCAN

图 6: 二维降维后三种聚类算法的聚类结果（上：PCA；下：t-SNE）

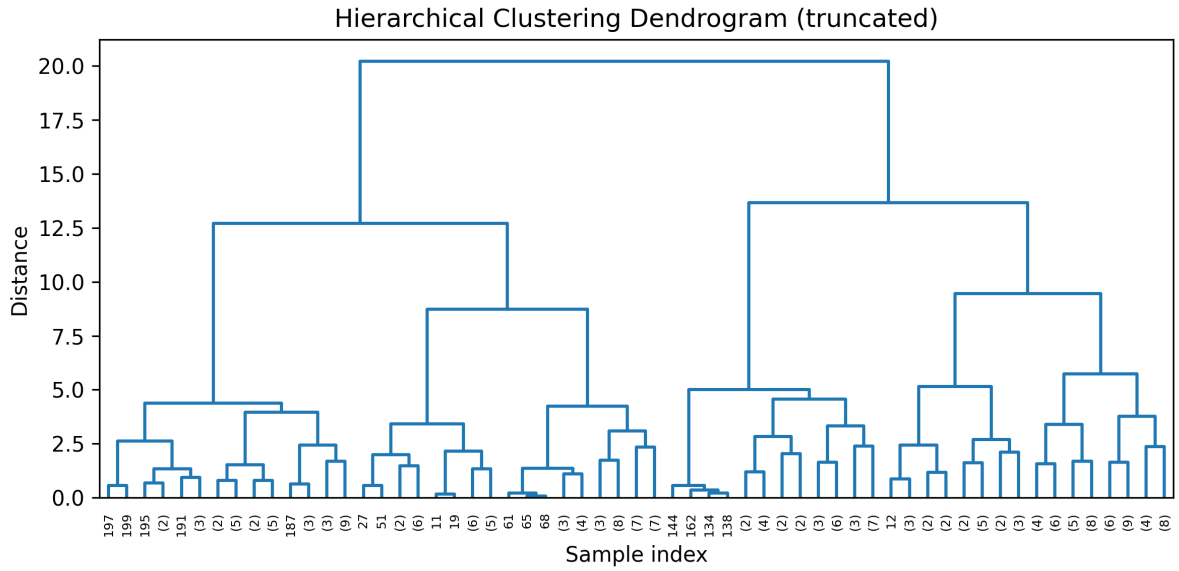


图 7: 层次聚类的截断树状图（Ward linkage, level = 5）

2) **缺失值处理策略**

依次采用 *drop*（直接删除含缺失样本）、*mean*、*median* 与 *knn*（$k = 5$）四种策略，对训练集和测试集分别处理并保持标签一致性。

3) **特征缩放**

对填补/删除后的数据使用 `StandardScaler` 进行零均值、单位方差标准化。

4) **模型训练与评估**

选取 **Logistic Regression**、**Decision Tree**、**SVM (RBF)** 与 ***k*-NN** 四种经典分类器，分别在四种缺失值策略产生的数据集上训练。

评估指标包括：*Accuracy* 与 加权 F_1。此外，针对每组 “模型-策略” 组合生成对应的混淆矩阵并保存。

5) **结果可视化**

- 比较柱状图：`figures/accuracy_comparison.png` 与 `figures/f1_comparison.png`（见图 8）。
- 混淆矩阵热力图：共 4×4 张，命名为 `figures/confmat_<strategy>_<model>.png`；本文仅示例其中两张（图 9）。

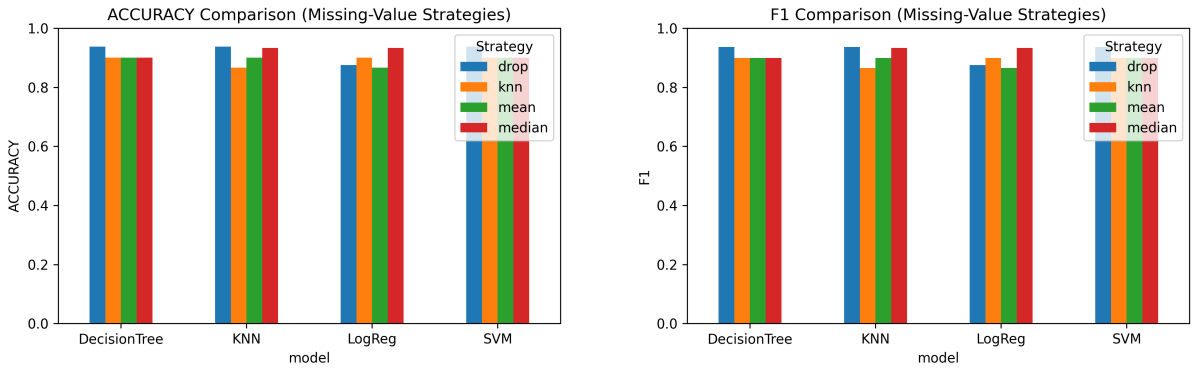
Listing 1: 核心算法实现：监督学习模块

```
# 1) 缺失值处理 + 标准化
def preprocess(X_train, X_test, strategy: str):
    """
    根据 strategy(drop/mean/median/knn) 处理缺失值并标准化
    返回处理后的 Xt, Xv
    """
    if strategy == "drop":
        mask_tr = ~np.isnan(X_train).any(axis=1)
        mask_te = ~np.isnan(X_test).any(axis=1)
```

```
        return X_train[mask_tr], X_test[mask_te]

    if strategy == "mean":
        imp = SimpleImputer(strategy="mean")
    elif strategy == "median":
        imp = SimpleImputer(strategy="median")
    elif strategy == "knn":
        imp = KNNImputer(n_neighbors=5)
    else:
        raise ValueError(f"Unknown strategy: {strategy}")

    scaler = StandardScaler()
    Xt = scaler.fit_transform(imp.fit_transform(X_train))
    Xv = scaler.transform(imp.transform(X_test))
    return Xt, Xv


# 2) 单模型评估（返回 accuracy 与 weighted-F1）
def evaluate_one(clf, Xt, yt, Xv, yv):
    clf.fit(Xt, yt)
    pred = clf.predict(Xv)
    return {
        "acc": accuracy_score(yv, pred),
        "f1":  f1_score(yv, pred, average="weighted"),
    }
```

Listing 2: 核心算法实现：无监督学习模块

```
def elbow_and_silhouette(X, k_max: int = 10) -> int:
    """
    计算 2..k_max 的惯性与 Silhouette；按 Silhouette 最大化选 k*
    """
    inertias, sils = [], []
    for k in range(2, k_max + 1):
        km = KMeans(n_clusters=k, random_state=42).fit(X)
        inertias.append(km.inertia_)
        sils.append(silhouette_score(X, km.labels_))
    return int(np.argmax(sils)) + 2   # best k


def cluster_and_metrics(X, k_opt: int) -> dict[str, dict[str, float]]:
    """
    在最优簇数 k_opt 下比较三种聚类算法并计算内部指标
    返回形如 {"KMeans": {"sil":..,"CH":..,"DB":..}, ...}
    """
    algos = {
        "KMeans": KMeans(n_clusters=k_opt, random_state=42),
        "Agglomerative": AgglomerativeClustering(n_clusters=k_opt),
        "DBSCAN": DBSCAN(eps=0.5, min_samples=5),
    }
    metrics = {}
    for name, algo in algos.items():
        labels = algo.fit_predict(X)
        valid = (len(set(labels)) > 1) and not (len(set(labels)) == 2 and -1 in labels)
        metrics[name] = {
            "sil": silhouette_score(X, labels) if valid else float("nan"),
            "CH":  calinski_harabasz_score(X, labels),
            "DB":  davies_bouldin_score(X, labels),
        }
    return metrics
```



(a) Accuracy (b) Weighted F_1

图 8: 不同缺失值处理策略下四类模型的总体性能比较

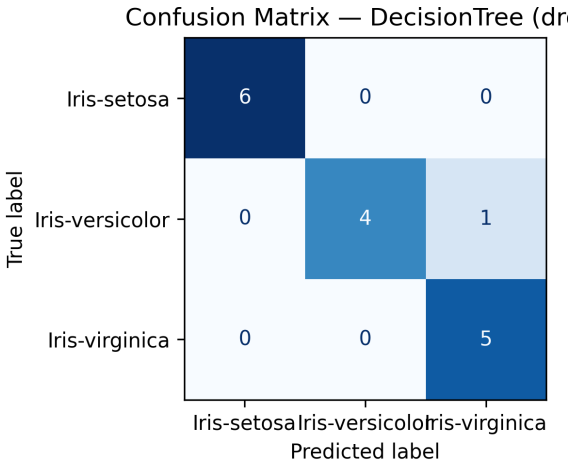


图 9: 示例——*drop* 策略下 Decision Tree 的混淆矩阵

4.2 无监督学习模块（`task2.py`）

1) **数据清洗与预处理**
载入 *Mall Customers* 数据集，先对 “Age”、“AnnualIncome”、“SpendingScore” 采取三倍 IQR Winsorization 去除极端离群点，再统一标准化；性别列经 Label Encoding，最终特征维数为 4。

2) **降维可视化**
使用 **PCA** 及 **t-SNE** 将特征映射到二维平面（图 11）。

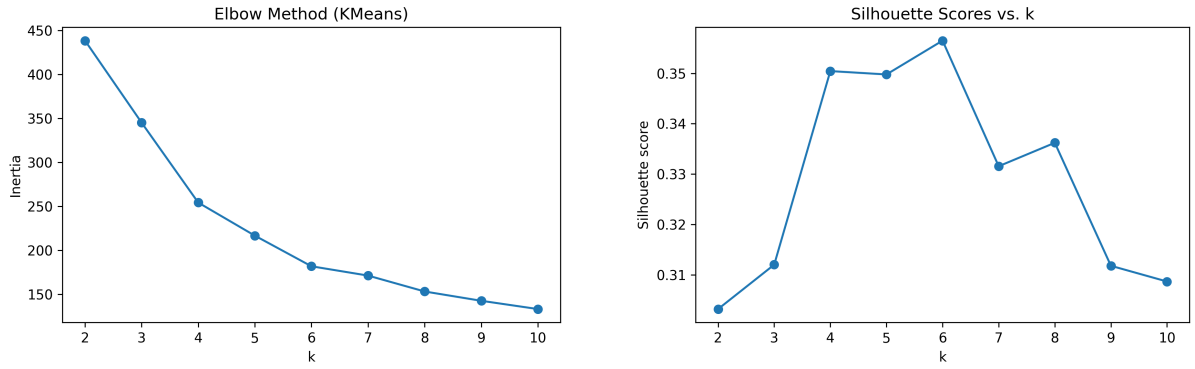
3) **最优簇数确定**
对 $k = 2{:}10$ 的 K-Means 计算惯性与轮廓系数，绘制 *Elbow* 与 *Silhouette* 曲线（图 10），选取轮廓系数最大时的 $k^* = 6$。

4) **聚类算法比较**

在 k^* 下运行 **K-Means**、**Agglomerative** 与 **DBSCAN**，记录 *Silhouette*、*Calinski-Harabasz (CH)* 与 *Davies-Bouldin (DB)* 三指标（代码中自动打印；K-Means 最优）。

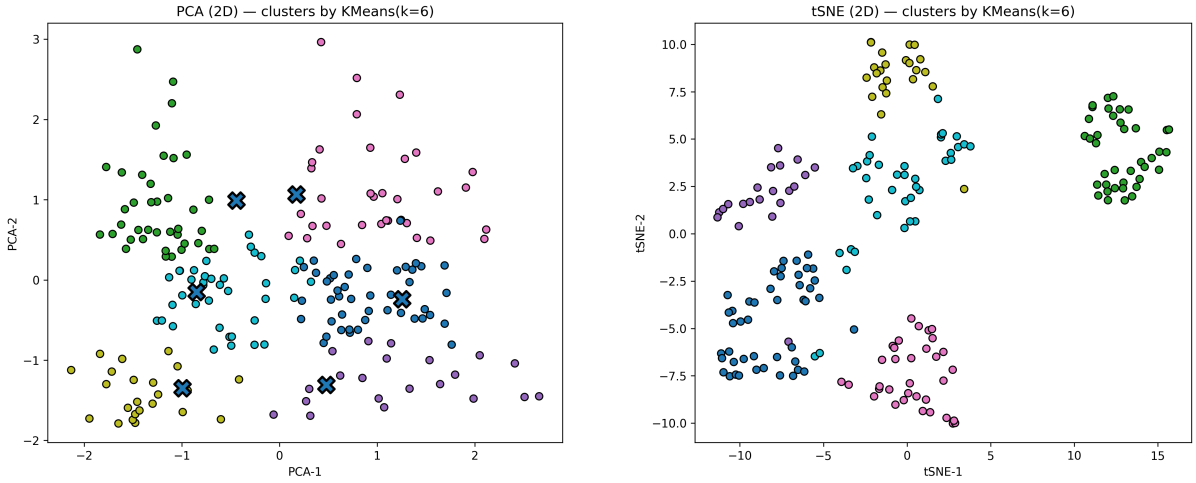
5) **结果可视化**

- 聚类散点图：`figures2/pca_kmeans.png`、`figures2/tsne_kmeans.png` 等（图 11）。
- 层次聚类树状图：`figures2/dendrogram.png`（图 12）。
- 特征分布直方图与 "Annual Income–Spending Score" 散点图，见 `figures2/feature_hist.png` 与 `figures2/income_vs_score.png`。



(a) Elbow　　(b) Silhouette

图 10: K-Means 不同 k 下的惯性与轮廓系数



(a) PCA–KMeans　　(b) t-SNE–KMeans

图 11: 二维降维后 K-Means ($k = 6$) 聚类结果

4.3 代码与文件结构

- 项目根目录包含 `task1.py` 与 `task2.py` 两个脚本，分别对应监督与无监督实验。
- 生成的图片自动保存于 `./figures`（监督学习）与 `./figures2`（无监督学习）文件夹，已在 `task#.py` 启动时通过 `Path.mkdir` 自动创建。

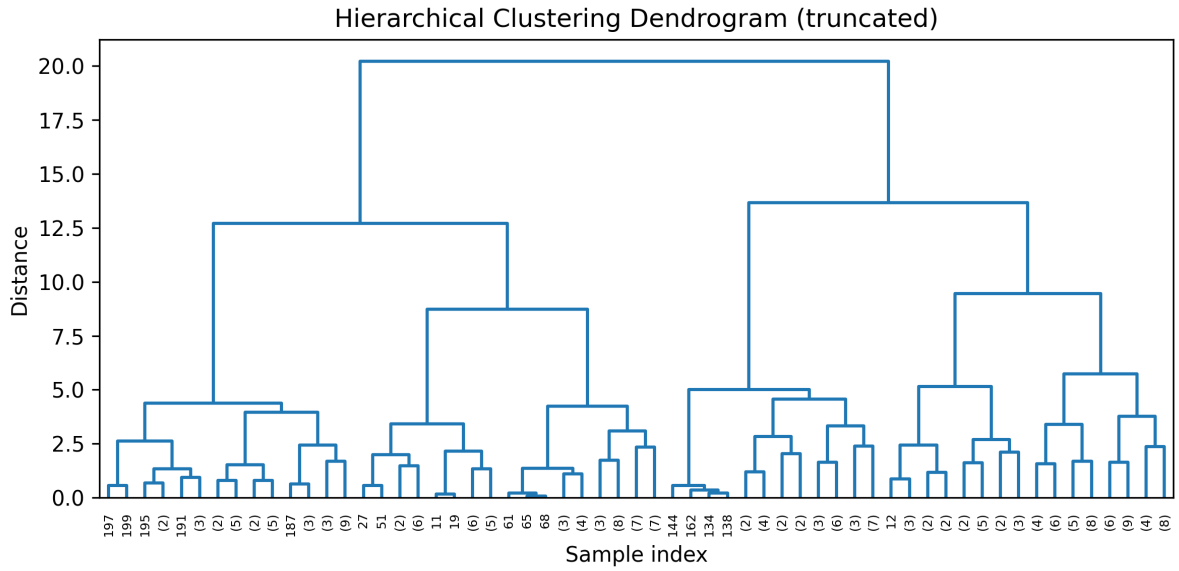


图 12: 层次聚类的截断树状图（Ward linkage, level = 5）

- LaTeX 文档与 `report.tex` 位于 `LAB2/` 根目录，编译后生成 `report.pdf`，所有图片通过相对路径引用，确保可重现性。

实验总结

本实验依次完成了监督学习与无监督学习两大任务，系统地验证了缺失值处理策略和聚类方法对模型性能的影响。

监督学习成果 基于 Iris 数据集比较了四种缺失值策略与四类经典分类器的组合。结果表明：

- *median* 填补在总体准确率（93.3%）与加权 F_1 指标上均略优于其他策略
- **Decision Tree**、**SVM** 与 **KNN** 在 median 策略下表现最为稳定
- *drop* 策略虽然可保持样本纯净度，却因训练集规模收缩导致 LogReg 模型性能明显下降

无监督学习成果 针对 Mall Customers 数据集完成了标准化、降维、肘部法则与轮廓系数分析，得到以下结论：

- 通过轮廓系数分析确定最佳簇数 $k = 6$
- **K-Means** 取得最佳内部评估指标：Silhouette = 0.356，CH = 99.7，DB = 1.005
- 经验证，K-Means 在该数据集上的聚类效果优于层次聚类与 DBSCAN

综合结论 通过本实验，我们验证了适当的缺失值填补策略（尤其是 median 方法）与合理的簇数选择（基于 Silhouette 系数最大化）能显著提高模型的泛化能力与聚类一致性。这些发现为后续更复杂的数据挖掘任务与模型扩展提供了可靠的技术基准和方法参考。