# 实验二：无监督学习入门（KMeans 聚类 + PCA 可视化，2 学时）

## 学习目标

* 使用无监督学习方法：KMeans 聚类；并用 PCA 将高维数据可视化为 2D 散点图。
* 学会在不改代码的前提下，通过命令行参数尝试不同的聚类数 k，并理解指标 inertia 与 silhouette 的直观意义。
* 会保存结果图与 CSV 指标，便于在实验报告中引用。 （本章课时：**2 学时**）

## 先修知识与软件环境

* 系统：Windows 10 / cmd
* Python：3.12（已配置清华源）
* 依赖：numpy、pandas、scikit-learn、matplotlib、seaborn

## 实验准备

1. 激活虚拟环境（如已创建可跳过）：

\.venv\Scripts\activate.bat

1. 确认依赖（若首次运行本课程，可执行）：

pip install numpy pandas scikit-learn matplotlib seaborn

1. 创建输出目录：

mkdir outputs

## 核心知识要点

* KMeans：把数据分成 k 个簇，让“同簇更相似、异簇更不同”；k 是人为设定的参数。
* inertia（惯性）：样本到簇中心的平方距离之和，越小代表簇内更紧凑；随 k 增大往往单调下降。
* silhouette（轮廓系数）：衡量样本与本簇的紧密度与同其他簇的分离度，范围约 [-1,1]，越大越好。
* PCA：把高维数据“投影”为 2D/3D 以便画图，方便直观观察聚类结果。

## 实验任务清单

1. 加载内置的 Iris 数据集（无须联网）。
2. 标准化特征（StandardScaler）。
3. 扫描 k=2..8，记录每个 k 的 inertia 与 silhouette。
4. 选择 silhouette 最大的 k 为“推荐值”，并用该 k 进行聚类与 2D 可视化。
5. 保存指标曲线与 PCA 聚类散点图，以及 CSV 指标汇总。

## 操作步骤

执行策略：只需运行给定命令/代码即可完成。代码已包含详细中文注释。

1. 新建脚本 exp2\_unsupervised.py，将以下代码完整复制粘贴保存：

"""  
脚本名称：exp2\_unsupervised.py  
用途：使用 KMeans 聚类 + PCA 可视化，演示无监督学习的基本流程。  
如何运行（Windows 10 + cmd）：  
 1) 激活虚拟环境： .\.venv\Scripts\activate.bat  
 2) 执行命令（Iris，k=2..8）：  
 python exp2\_unsupervised.py --dataset iris --k-range 2 8 --seed 42 --max-iter 300 --out-dir outputs  
  
输出：  
 - 指标文件：outputs\k\_scan\_metrics.csv （每个 k 的 inertia 与 silhouette）  
 - 曲线图： outputs\k\_elbow\_inertia.png （inertia-肘部图）  
 outputs\k\_silhouette.png （silhouette-轮廓系数）  
 - 聚类图： outputs\pca\_clusters.png （PCA 2D 聚类散点）  
 - 可选对照：outputs\pca\_true\_labels.png （PCA 2D 真实标签散点，仅 iris/digits 可用）  
"""  
  
import argparse  
import os  
from typing import Tuple  
  
import numpy as np  
import pandas as pd  
import seaborn as sns  
import matplotlib.pyplot as plt  
  
from sklearn.datasets import load\_iris, load\_digits  
from sklearn.preprocessing import StandardScaler  
from sklearn.decomposition import PCA  
from sklearn.cluster import KMeans  
from sklearn.metrics import silhouette\_score, adjusted\_rand\_score, normalized\_mutual\_info\_score  
  
  
def load\_dataset(name: str) -> Tuple[np.ndarray, np.ndarray]:  
 """加载内置数据集。  
 返回：X（特征），y（真实标签，若无则返回占位 -1）。  
 默认使用 iris（简单直观）。  
 """  
 name = name.lower()  
 if name == "iris":  
 ds = load\_iris()  
 return ds.data, ds.target  
 elif name == "digits":  
 ds = load\_digits()  
 return ds.data, ds.target  
 else:  
 raise ValueError(f"Unsupported dataset: {name}")  
  
  
def main():  
 # 解析命令行参数：不需要学生改代码，只改参数即可尝试不同 k 与数据集  
 parser = argparse.ArgumentParser(  
 description=(  
 "Exp2: Unsupervised Learning "  
 "with KMeans + PCA"  
 )  
 )  
 parser.add\_argument(  
 "--dataset",  
 default="iris",  
 choices=["iris", "digits"],  
 help="数据集选择（默认 iris）",  
 )  
 parser.add\_argument(  
 "--k-range",  
 nargs=2,  
 type=int,  
 default=[2, 8],  
 help="k 的起止（含），如 2 8",  
 )  
 parser.add\_argument(  
 "--seed",  
 type=int,  
 default=42,  
 help="随机种子",  
 )  
 parser.add\_argument(  
 "--max-iter",  
 type=int,  
 default=300,  
 help="KMeans 最大迭代次数",  
 )  
 parser.add\_argument(  
 "--out-dir",  
 default="outputs",  
 help="输出目录",  
 )  
 args = parser.parse\_args()  
  
 os.makedirs(args.out\_dir, exist\_ok=True)  
 sns.set(style="whitegrid", font="sans-serif")  
  
 # 1) 加载数据  
 print("[1/6] 加载数据集（内置，无需联网）...")  
 X, y\_true = load\_dataset(args.dataset)  
 print(  
 f" 数据集: {args.dataset} 样本数: {X.shape[0]} "  
 f"特征数: {X.shape[1]}"  
 )  
  
 # 2) 标准化特征（让不同量纲的特征处于可比尺度）  
 print("[2/6] 标准化特征（StandardScaler）...")  
 scaler = StandardScaler()  
 X\_std = scaler.fit\_transform(X)  
  
 # 3) 扫描不同的 k，记录 inertia 与 silhouette  
 print(  
 "[3/6] 扫描 k，在每个 k 上拟合 KMeans 并计算指标..."  
 )  
 k\_start, k\_end = int(args.k\_range[0]), int(args.k\_range[1])  
 all\_rows = []  
 for k in range(k\_start, k\_end + 1):  
 # 说明：n\_init=10 表示尝试多次随机初始化以获得更稳定的结果  
 km = KMeans(  
 n\_clusters=k,  
 n\_init=10,  
 max\_iter=args.max\_iter,  
 random\_state=args.seed,  
 )  
 labels = km.fit\_predict(X\_std)  
 inertia = float(km.inertia\_)  
 # silhouette 需要 k>=2 且每簇至少有一个点，这里数据集满足，仍做保护  
 sil = float(silhouette\_score(X\_std, labels)) if k >= 2 else float("nan")  
 row = {"k": k, "inertia": inertia, "silhouette": sil}  
 # 若有真实标签（iris/digits），在“推荐 k”时再计算 ARI/NMI，这里先不算  
 all\_rows.append(row)  
  
 df = pd.DataFrame(all\_rows)  
 csv\_path = os.path.join(args.out\_dir, "k\_scan\_metrics.csv")  
 df.to\_csv(csv\_path, index=False, encoding="utf-8-sig")  
 print(" 已保存:", csv\_path)  
  
 # 4) 可视化 k 扫描的两张曲线：肘部图（inertia）与 silhouette  
 print("[4/6] 绘制 k-曲线（inertia 与 silhouette）...")  
 plt.figure(figsize=(6, 4))  
 sns.lineplot(  
 x="k",  
 y="inertia",  
 data=df,  
 marker="o",  
 )  
 plt.xlabel("k (clusters)")  
 plt.ylabel("inertia (越低越好)")  
 plt.title("KMeans Elbow Curve (Inertia)")  
 plt.tight\_layout()  
 elbow\_path = os.path.join(args.out\_dir, "k\_elbow\_inertia.png")  
 plt.savefig(elbow\_path, dpi=150)  
 plt.close()  
  
 plt.figure(figsize=(6, 4))  
 sns.lineplot(  
 x="k",  
 y="silhouette",  
 data=df,  
 marker="o",  
 )  
 plt.xlabel("k (clusters)")  
 plt.ylabel("silhouette (越高越好)")  
 plt.title("KMeans Silhouette Score")  
 plt.tight\_layout()  
 sil\_path = os.path.join(args.out\_dir, "k\_silhouette.png")  
 plt.savefig(sil\_path, dpi=150)  
 plt.close()  
  
 # 5) 选择 silhouette 最大的 k 作为推荐值，并进行 PCA 2D 可视化  
 print("[5/6] 选择 silhouette 最大的 k 作为推荐值，并进行 PCA 可视化...")  
 # 若所有 silhouette 为 NaN（极少数情况），退化为选 inertia 最小  
 if df["silhouette"].notna().any():  
 best\_row = df.sort\_values(by="silhouette", ascending=False).iloc[0]  
 else:  
 best\_row = df.sort\_values(by="inertia", ascending=True).iloc[0]  
 best\_k = int(best\_row["k"])  
 print(f" 推荐 k = {best\_k}")  
  
 km\_best = KMeans(  
 n\_clusters=best\_k,  
 n\_init=10,  
 max\_iter=args.max\_iter,  
 random\_state=args.seed,  
 )  
 labels\_best = km\_best.fit\_predict(X\_std)  
  
 # PCA 到 2D 便于画图  
 pca = PCA(n\_components=2, random\_state=args.seed)  
 X\_2d = pca.fit\_transform(X\_std)  
  
 plt.figure(figsize=(6, 5))  
 sns.scatterplot(  
 x=X\_2d[:, 0],  
 y=X\_2d[:, 1],  
 hue=labels\_best,  
 palette="tab10",  
 s=24,  
 alpha=0.85,  
 legend=True,  
 )  
 plt.xlabel("PC1")  
 plt.ylabel("PC2")  
 plt.title(f"PCA Scatter (KMeans k={best\_k})")  
 plt.tight\_layout()  
 pca\_cluster\_path = os.path.join(args.out\_dir, "pca\_clusters.png")  
 plt.savefig(pca\_cluster\_path, dpi=150)  
 plt.close()  
  
 # 可选：若数据集有真实标签，画一张“真实标签”的对照图（便于直观比较）  
 if (args.dataset in ["iris", "digits"]) and (y\_true is not None):  
 plt.figure(figsize=(6, 5))  
 sns.scatterplot(  
 x=X\_2d[:, 0],  
 y=X\_2d[:, 1],  
 hue=y\_true,  
 palette="tab10",  
 s=24,  
 alpha=0.85,  
 legend=True,  
 )  
 plt.xlabel("PC1")  
 plt.ylabel("PC2")  
 plt.title("PCA Scatter (True Labels)")  
 plt.tight\_layout()  
 pca\_true\_path = os.path.join(args.out\_dir, "pca\_true\_labels.png")  
 plt.savefig(pca\_true\_path, dpi=150)  
 plt.close()  
 else:  
 pca\_true\_path = None  
  
 # 6) 若有真实标签，报告 ARI/NMI（仅作为参考，不影响聚类本身的“无监督”性质）  
 summary\_lines = []  
 if (args.dataset in ["iris", "digits"]) and (y\_true is not None):  
 ari = adjusted\_rand\_score(y\_true, labels\_best)  
 nmi = normalized\_mutual\_info\_score(y\_true, labels\_best)  
 summary\_lines.append(f"ARI = {ari:.3f}")  
 summary\_lines.append(f"NMI = {nmi:.3f}")  
  
 # 保存一份简要总结，便于写报告  
 summary\_lines = [  
 f"dataset = {args.dataset}",  
 f"k\_recommended = {best\_k}",  
 f"elbow\_png = {os.path.basename(elbow\_path)}",  
 f"silhouette\_png = {os.path.basename(sil\_path)}",  
 f"pca\_clusters\_png = {os.path.basename(pca\_cluster\_path)}",  
 f"pca\_true\_labels\_png = {os.path.basename(pca\_true\_path) if pca\_true\_path else 'N/A'}",  
 ] + summary\_lines  
  
 summary\_path = os.path.join(args.out\_dir, "summary.txt")  
 with open(summary\_path, "w", encoding="utf-8") as f:  
 f.write("\n".join(summary\_lines))  
  
 print("输出完成：")  
 print(" - 指标CSV:", csv\_path)  
 print(" - 肘部图:", elbow\_path)  
 print(" - 轮廓系数图:", sil\_path)  
 print(" - 聚类散点:", pca\_cluster\_path)  
 if pca\_true\_path:  
 print(" - 真实标签散点:", pca\_true\_path)  
 print(" - 总结:", summary\_path)  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

1. 运行：

python exp2\_unsupervised.py --dataset iris --k-range 2 8 --seed 42 --max-iter 300 --out-dir outputs

可选：在 digits 数据集上再运行一次（维度更高，PCA 可视化更有对比）：

python exp2\_unsupervised.py --dataset digits --k-range 2 12 --seed 42 --max-iter 300 --out-dir outputs

1. 查看输出：

* 指标 CSV：outputs/k\_scan\_metrics.csv
* 曲线图：outputs/k\_elbow\_inertia.png、outputs/k\_silhouette.png
* 聚类图：outputs/pca\_clusters.png（以及 outputs/pca\_true\_labels.png 如有）
* 摘要：outputs/summary.txt

## 预期输出

* 随着 k 增大，inertia 通常下降（肘部法寻找“弯折点”）。
* silhouette 在某个 k 附近达到较高值，代表簇更紧凑、相互更分离。
* PCA 2D 散点图中，不同颜色代表不同簇，能直观看到“分组”。

## 思考与讨论

1. 如果把 k 设得很大（例如接近样本数），inertia 会怎样变化？silhouette 又可能出现什么情况？
2. 看 PCA 散点图时，你能否用生活中的“分组”例子来帮助描述这些簇（例如按兴趣、口味分组）？
3. 为什么标准化能帮助 KMeans 更好地工作？如果不标准化会发生什么？

## 常见错误与排错

* ValueError: Number of labels is 1：silhouette 需要至少 2 个簇；确保 k≥2。
* 曲线或散点图未生成：检查 outputs 目录是否存在，或重新运行脚本（脚本会自动创建目录）。
* 运行很慢：本实验数据集较小，如发生卡顿可降低 k 上限或减少 max-iter。

## 提交要求与评分标准

* 提交内容：PDF 实验报告 + outputs 目录（含 CSV、曲线图、PCA 图、摘要）。
* 命名规范：学号\_姓名\_实验二
* 评分 Rubric：准确性 40% / 完整性 30% / 表达 20% / 规范 10%