数据挖掘导论期末报告

3220106424 张天逸

数据集概况

Kaggle上的nus-fintech-recruitment这个项目的背景是：随着电子支付和线上消费的普及，信用卡交易量急剧增长，伴随而来的就是各种盗刷、钓鱼网站、终端劫持等欺诈手段不断翻新。对于银行、支付机构和商家来说，依靠人工审核海量交易已经不现实，同时一旦放过欺诈交易就会造成巨额损失，误拦正常交易又会影响用户体验。因此，NUS 金融科技社 “Credit Card Fraud Detection” 挑战赛，要求参赛者设计一套端到端的机器学习方案，通过对 2021 年 8 月到 2022 年 4 月的信用卡交易数据做特征工程、模型训练和评估，自动识别出欺诈交易。与其追求单一最高的准确率，本次比赛更看重同学们在数据预处理、特征构造、模型选择和评估指标（Precision — 精确率）上的思考过程与创新能力。通过这个项目，我练习从多表合并、时间与地理特征提取，到滑动窗口行为统计，再到无监督聚类、离群点检测、频繁模式挖掘，最后使用随机森林或其他分类器进行欺诈预测的全流程，深入理解机器学习在金融风控场景中的应用。

数据特征

特征名称	数据类型	物理意义	取值范围 / 取值列表
TRANSACTION_ID	int64	交易唯一编号	整数，约 59383–579320
TX_DATETIME	datetime64[ns]	交易发生的日期和时间	2021-08-01 00:00:00 至 2022-04-29 23:59:59
CUSTOMER_ID	int64	客户（持卡人）唯一编号	整数，1–1000
TERMINAL_ID	int64	刷卡终端唯一编号	整数，1–2000
TX_AMOUNT	float64	本次交易金额	实数，≈0.01–1000.00
TX_FRAUD	int64	交易是否为欺诈	{0,1}
x_customer_id	float64	客户地理位置 X 坐标	实数，≈0.00–100.00

特征名称	数据类型	物理意义	取值范围 / 取值列表
y_customer_id	float64	客户地理位置 Y 坐标	实数，≈0.00–100.00
mean_amount	float64	该客户历史交易金额的平均值	实数，≈0.00–1000.00
std_amount	float64	该客户历史交易金额的标准差	实数，≈0.00–100.00
mean_nb_tx_per_day	float64	该客户平均每日交易次数	实数，≈0.00–50.00
available_terminals	object (str)	客户半径 5 单位内可用终端 ID 列表（逗号分隔）	列表格式，包含若干 TERMINAL_ID，如 “[27,493,584,…]”
nb_terminals	int64	`available_terminals` 列表中终端的数量	整数，0–54
x_terminal_id	float64	终端地理位置 X 坐标	实数，≈0.00–100.00
y_terminal_id	float64	终端地理位置 Y 坐标	实数，≈0.00–100.00

数据分布（以是否订阅为例）

类别	样本数量	占比
正常 (0)	56 890	97.67%
欺诈 (1)	1 357	2.33%

> *注：总样本数 58 247 条，Imbalance Ratio ≈ 41.94（多数/少数）。*

部署

依赖：pandas numpy scikit-learn matplotlib scipy mlxtend

如果存在python版本冲突，建议创建虚拟环境：

cd lab4

python -m venv .venv # 或 conda create -n pcos python=3.12
source .venv/bin/activate # Windows: .venv\Scripts\activate

算法

初始化与标签分配

1. `init_centers(X, k)`

随机从数据中选取 k 个样本，作为初始质心（centroids）。

2. `assign_labels(X, centers)`

对每个样本，计算它到所有质心的欧氏距离，选择最近的那个质心，从而给出每个样本的“簇标签”。

```
dists = np.linalg.norm(X[:,None,:] - centers[None,:,:], axis=2)
labels = np.argmin(dists, axis=1)
```

标准 KMeans 算法（Lloyd’s 算法）

更新质心函数 `update_centers(X, labels, k)`

对于刚分配好的标签，重新计算每个簇的“均值位置”作为新的质心。

$$\mu_i = \frac{1}{|\{x_j : label_j = i\}|} \sum_{j:label_j=i} x_j$$

自实现kmeans算法 `kmeans(X, k, max_iters, tol)`

调用 `init_centers` 随机选质心。循环：用 `assign_labels` 给所有样本分簇；用 `update_centers` 更新质心；计算质心移动距离 `shift = ||new_centers - old_centers||`；若 `shift < tol`（收敛阈值），提前终止；收敛后：再一次 `assign_labels` 输出最终标签

缺点：全量数据、每次迭代都要遍历所有样本→对大数据量较慢。

Mini-Batch KMeans（加速kmeans）

`mini_batch_kmeans(X, k, batch_size, n_iters)`

1. 随机抽取 `batch_size` 个样本
2. 对这批样本做一次 `assign_labels`
3. 针对每个簇，计算该批次中属于它的样本均值，直接 **覆盖** 更新对应中心
   每次迭代只处理一小块数据，内存友好、速度更快，适合大规模数据。

聚类效果评估

`inertia(X, centers, labels)`

簇内平方和（SSE），即所有样本到其各自质心的平方距离之和：

$$\mathrm{SSE} = \sum_{i=1}^{n} \|x_i - \mu_{\ell_i}\|^2$$

Elbow 法则中常用来判断聚类数 k 是否合适——随着 k 增加，SSE 会下降，但拐点处就是最佳 k。

`silhouette_manual(X, labels, sample_size)`

Silhouette 系数用来衡量样本“贴合”自己簇 vs 最邻近的其他簇：

- a_i：样本到同簇内其他点的平均距离
- b_i：样本到最近邻另一簇所有点的平均距离
- 单点分数：$(b_i - a_i)/\max(a_i, b_i)$
  为了速度，支持先随机抽样 `sample_size` 条数据计算，再取平均。

降维可视化

`PCA(n_components=2)`

主成分分析，把高维特征投影到二维，保留数据方差最大的方向。

在二维平面上，用不同颜色标注聚类标签，直观展示簇间分离度。

离群点检测（IsolationForestCustom）

`IsolationForestCustom`

Isolation Forest 通过随机切分的方式“隔离”样本，路径越短越容易被隔离→越可能是异常。

1. 随机抽 `max_samples` 个点，构建一棵“随机切分树”：
    - 每个非叶节点随机选择一个特征与分割值
    - 递归下去，直到树深或样本数 ≤1
2. 重复上一步，生成 `n_estimators` 棵树 → 构成森林
3. 对新样本，计算其在每棵树上的平均路径长度 $h(x)$
4. 评分公式：

    $$s(x) = 2^{-E[h(x)]/c(n)}$$

    其中 $c(n)$ 是理想随机切分的期望路径长度
5. 按 `contamination`（异常比例）确定阈值，分出正常（1）与异常（-1）

数据预处理

数据预处理阶段，我们对原始交易、客户和终端数据做了如下几步操作：

缺失值检测并合并静态元数据

分别加载 `train.csv`、`test.csv`（将交易时间解析为 `datetime`），以及 `customer.csv`、`terminal.csv`。
检查各表的缺失值情况，确认没有丢失的关键字段。
将客户表和终端表通过 `CUSTOMER_ID` / `TERMINAL_ID` 左连接到交易主表上，补齐每笔交易对应的客户坐标、历史统计、终端坐标等静态信息。

坐标缺失填充

对于合并后可能出现的坐标列（客户和终端的 x/y 坐标），使用数据集的联合中位数进行填充，确保无空值。

计算全部可用终端数量

将 `available_terminals`（一个逗号分隔 ID 列表）拆解成实际可用终端数量 `avail_term_count`，并丢弃原始字符串列。

小时级别金额统计

从交易时间 `TX_DATETIME` 中提取小时（`hour`）、工作日/周末标记（`weekday`，`is_weekend`）、月份（`month`）等信息。
在数据集中按小时聚合计算每小时的平均交易金额和标准差（`hour_amt_mean`，`hour_amt_std`），并将这些统计值映射回数据集。

计算地理距离特征

计算每笔交易中“客户坐标”与“终端坐标”之间的直线距离 `cust_term_dist`。

金额比率特征

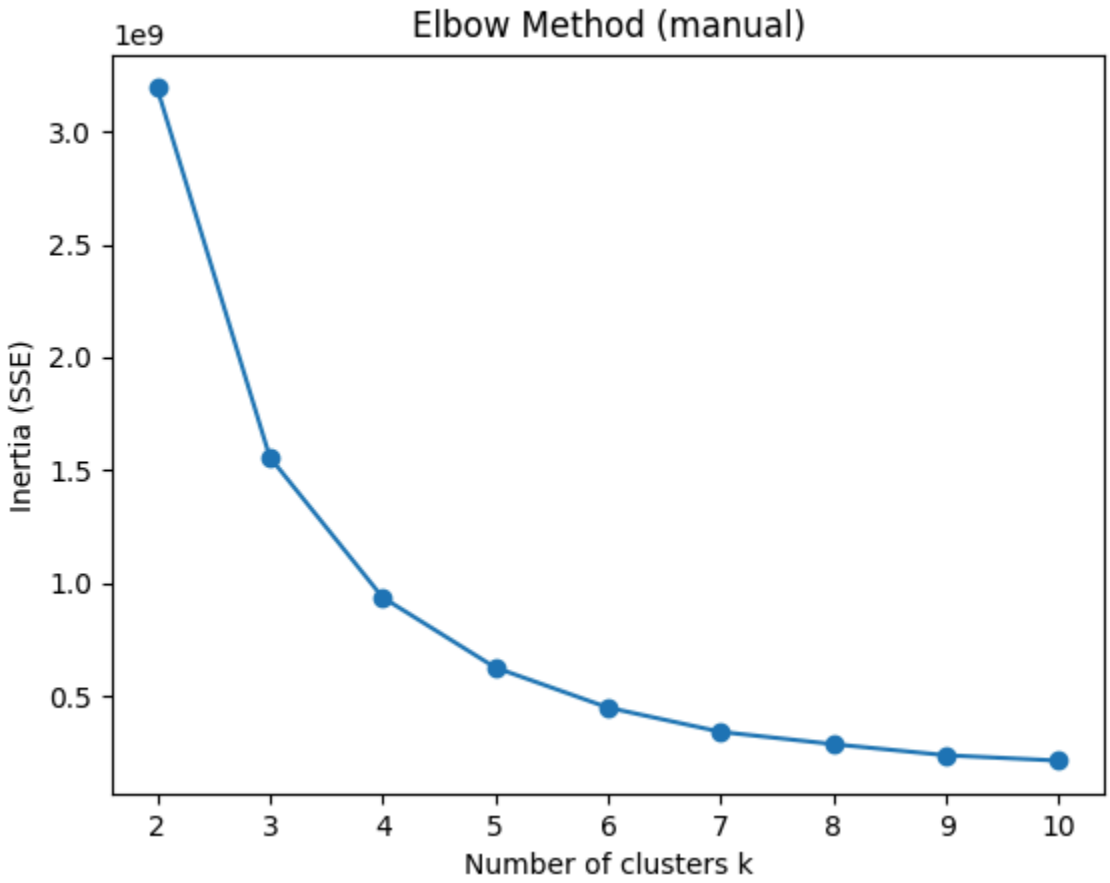
生成相对于该客户历史平均值和标准差的比率特征（`amt_to_mean`，`amt_to_std`），以及对数金额 `log_amount`。

7 天滑窗行为特征

对每个客户，按交易时间做滚动 7 天窗口，统计过去 7 天内（不含当前交易）发生的交易次数 `txn_cnt_7d` 和交易总额 `txn_amt_7d`。

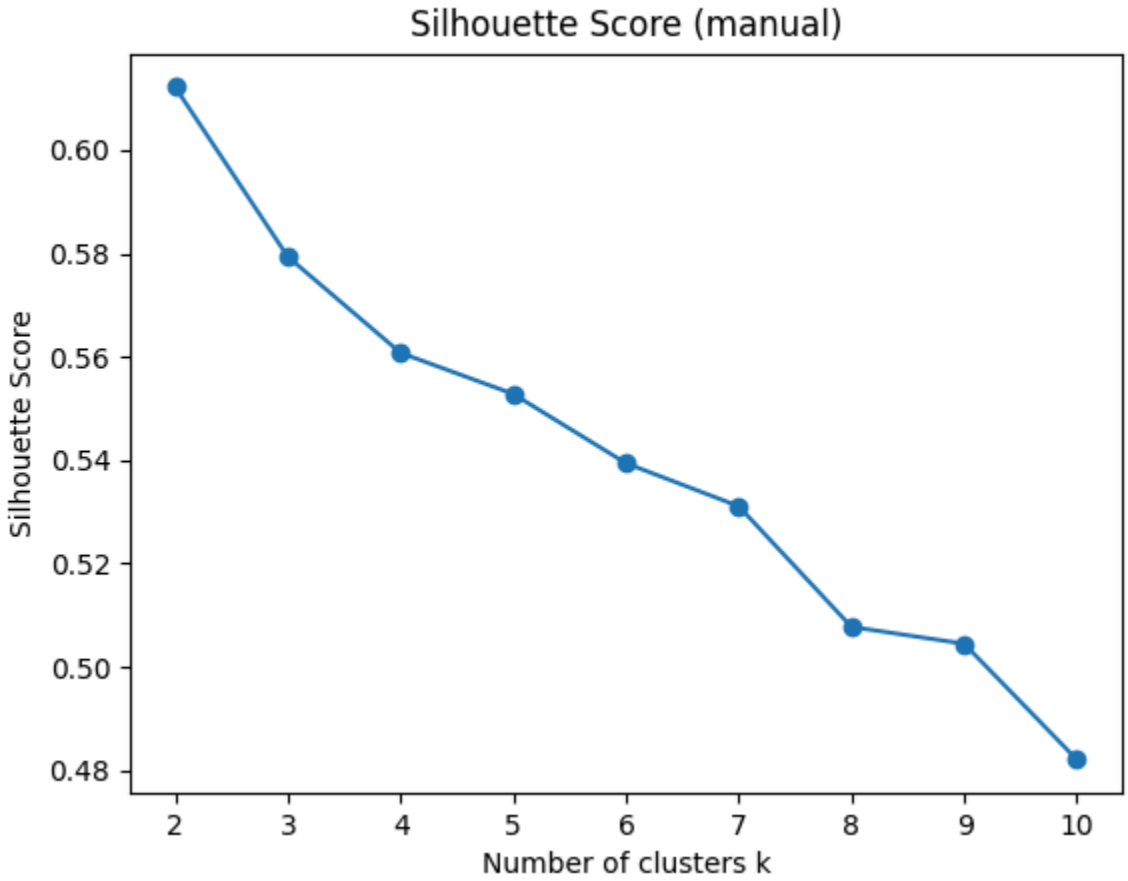
可视化选型及结果

Elbow Method (Mini-Batch KMeans)



- **选型理由**：Elbow（“肘部法则”）通过绘制簇内平方和（Inertia 或 SSE）随 k 变化的曲线，帮助我们发现“拐点”（拐点前 SSE 降得快，拐点后下降变缓），从而选出一个既能减少误差又不过度细分的 k 值。
- **结果解读**：SSE 随 k变化：从 k=2→3 出现了最大幅度的下降，从 3→4、4→5 开始趋于平缓。拐点大约出现在 k=4 左右，此后继续增大簇数带来的误差减少非常有限。

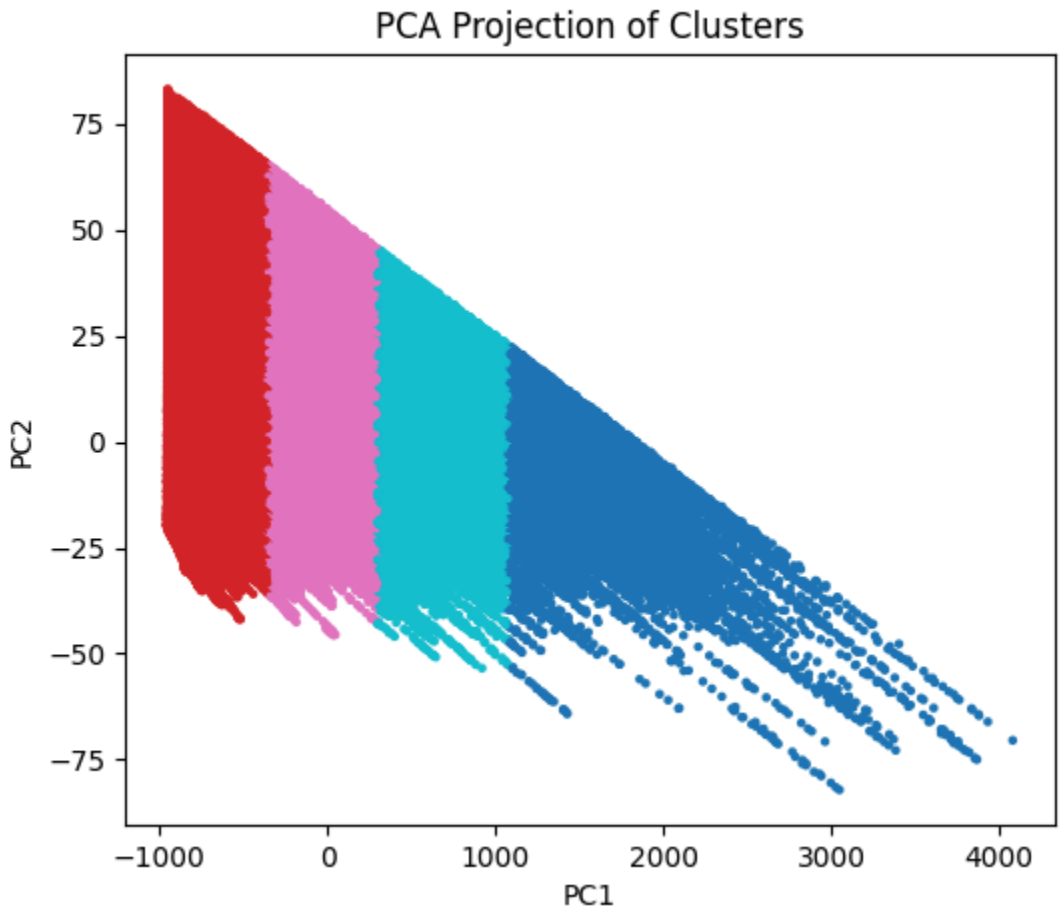
Silhouette Score (手写 Mini-Batch KMeans)



- **选型理由**：Silhouette 分数衡量样本与自身簇内点的相似度与与最近邻簇点的相似度之差，范围 $[-1, 1]$。越接近 1 越好，说明簇内紧密且簇间分离。
- **结果解读**：分数趋势：最高在 k=2 时约 0.61，随后随着 k 增加而持续下降。

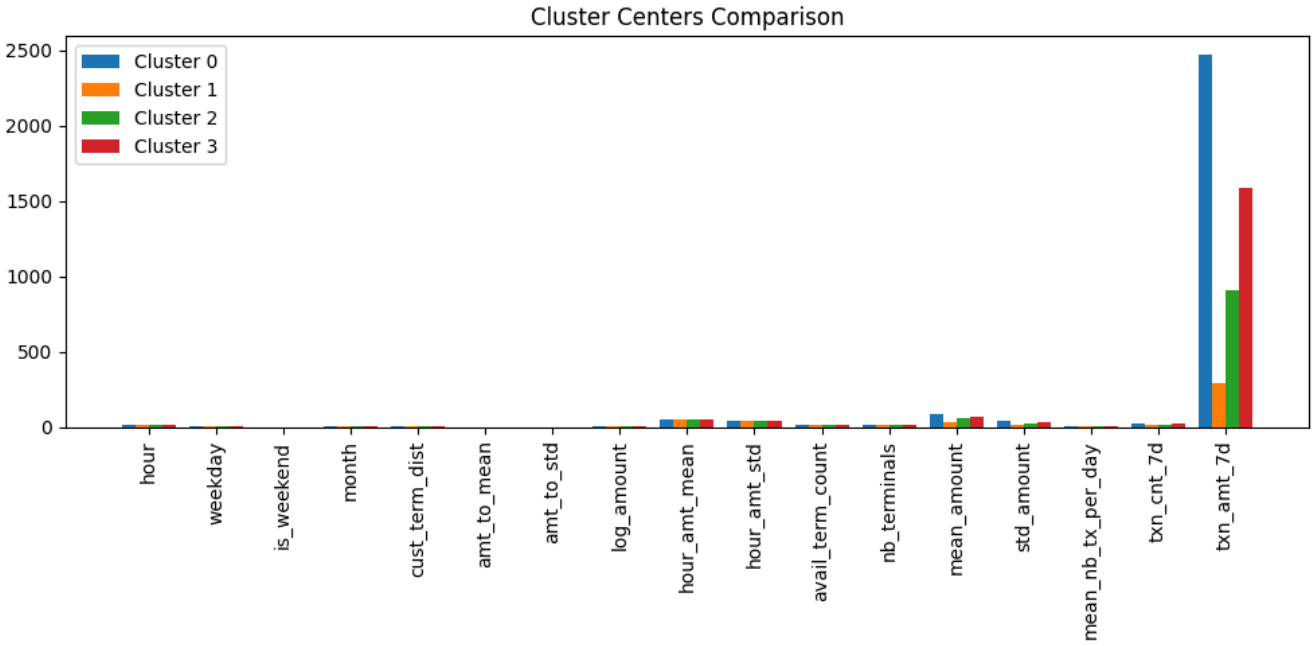
结论：从纯分离度角度看，2 个簇时各簇内部最紧凑、簇间最分离；如果追求最优的簇间可区分度，应优先考虑 k=2。

PCA 散点投影



- **选型理由**：原始特征空间是 17 维，不可视化。PCA 将高维数据投影到二维，同时尽可能保留最大方差，用不同颜色标注簇标签，肉眼观察各簇在投影平面上的分离情况。
- **结果解读**：
  聚类后的点云在第一主成分（PC1）方向上分成了 4 条“垂直带”，阶梯状依次展开。
  PC1 很大程度上对应交易金额与频次等累计特征，簇按“交易强度”由低到高排列。

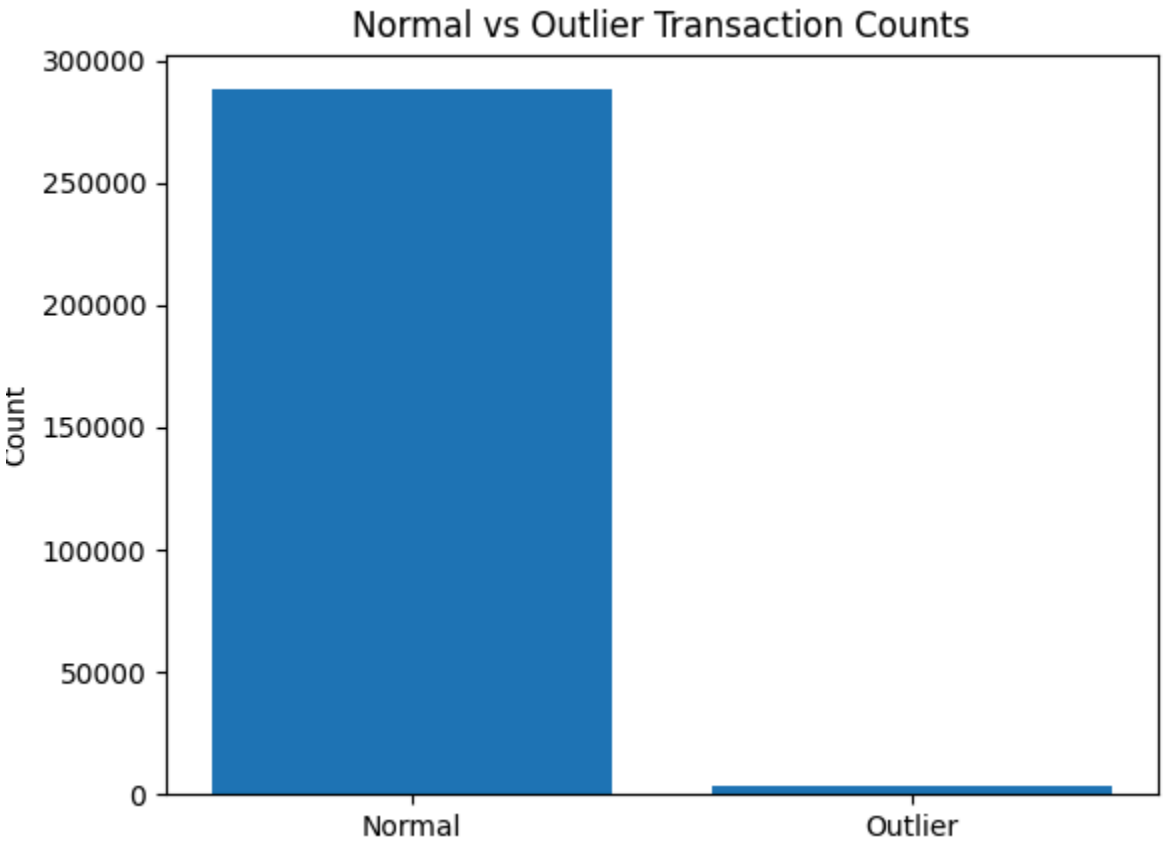
簇中心比较条形图



- **选型理由**：将各簇的中心在所有特征上的坐标并排绘制，直观对比簇之间在不同特征上的差异程度。
- **结果解读**：
  txn_amt_7d（近 7 天交易总额）对簇划分贡献最大，不同簇中心差异最明显。
  txn_cnt_7d（近 7 天交易次数）、hour_amt_mean（小时均额）、amt_to_mean（与历史均额比率）也有次级差异。
  其他特征（如 hour, weekday 等）中心值相近，说明它们对簇区分作用较弱。
  聚类主要是在用户过往一周的交易量/频次上做区分。

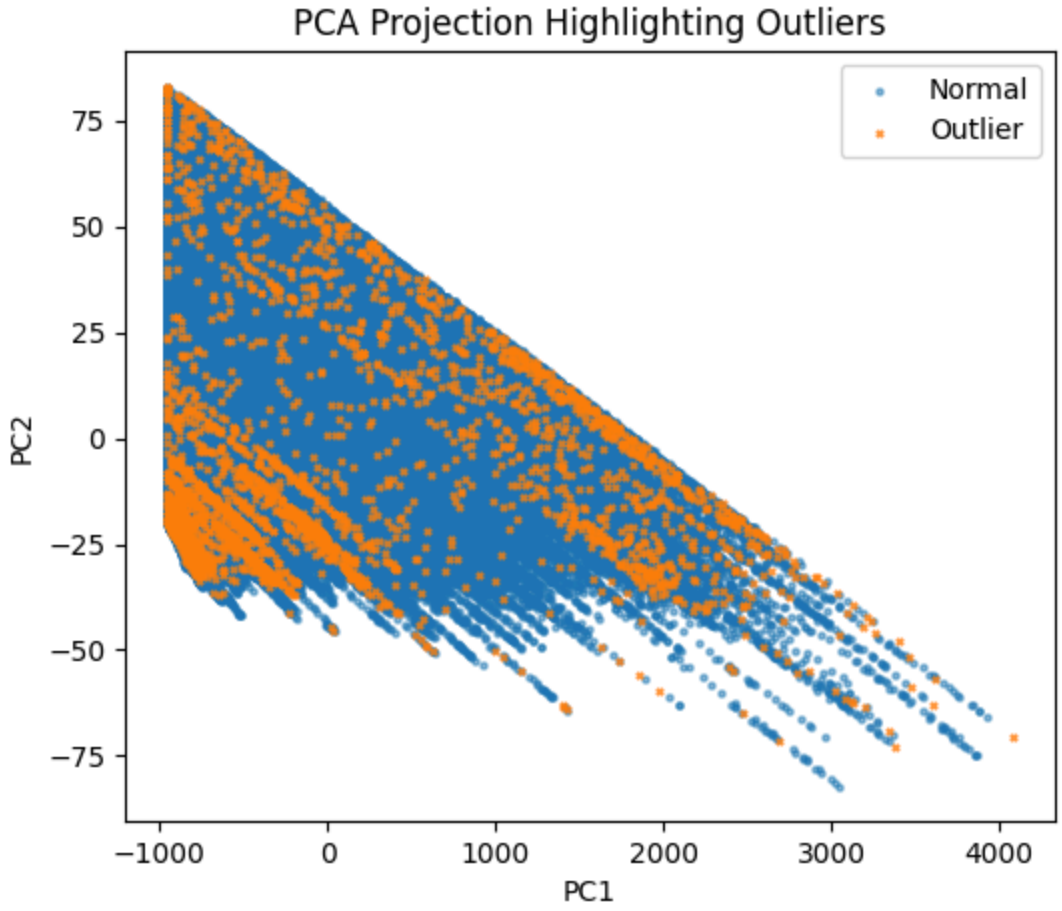
离群点检测可视化

5.1 离群点数量分布



- **选型理由**：简单条形图展示检测出的“正常” vs “异常”交易样本数，直观了解异常比例。
- **结果解读**：正常样本约 29 万条，异常（Outlier）样本约 3 千条，约占总数的 1%，与设定污染率一致。

5.2 PCA 投影高亮离群点



- **选型理由**：在同样的 PCA 平面中将异常点（用“×”标记）与正常点区分开，看异常点是否集中在某些区域。
- **结果解读**：
  离群点（橙色“×”）在 PCA 空间中散落于分布边缘，尤其多集中在点云尾部，说明它们在高维特征上更“极端”。
  IsolationForestCustom 能较好地捕捉到那些偏离主流交易模式的样本。

结果分析与讨论

聚类算法

KMeans 对初始质心敏感，可能陷入局部最优；

可尝试基于密度（DBSCAN）或层次（HAC）的聚类，发现不同形状的数据结构。

特征工程

目前特征主要聚焦金额与频次，可引入更多上下文：

终端地理集群热度、商户类别、客户信用等级等；

时序模型（LSTM/Transformer）捕捉交易行为序列。

离群检测

IsolationForest 简化版对高维深噪数据表现有限；

可对比 One-Class SVM、深度自动编码器等深度方法。

模型评估

聚类与离群本质为无监督，难以用标签直观评估；

可配合已知欺诈标签做后续统计：

分析各簇/离群点中真实欺诈率，衡量分群价值。

总结

本次实验成功将交易样本基于近 7 天的金额与频次分为多个行为簇，并在降维可视化中得到验证；

利用 IsolationForestCustom 检测到约 1% 异常交易，可用于补充传统监督模型的风险预警；

后续可融合更多特征与算法，对比不同方案的检测效果，并基于真实欺诈标签设计更精细的风控策略。