理论与反思

DTS304TC 课程作业 1　学号: 1234560

---

1. Bagging 与 Boosting 对比

Bagging（Bootstrap Aggregating，自助聚合）通过对原始数据集进行 B 次独立自助采样，在每次采样得到的子集上训练一棵决策树作为基学习器，然后通过多数投票（分类）或平均（回归）来聚合所有基学习器的预测结果。由于每棵树都在不同的随机数据子集上学习，它们的预测误差在聚合时会部分相互抵消，因此 Bagging 主要降低方差。相比之下，Boosting 按顺序训练基学习器：每个新的学习器都专注于之前集成模型的残差或错误样本，从而更激进地降低偏差。两者的核心概念区别在于：Bagging 将所有基学习器视为同等重要，而 Boosting 根据过去的错误动态地重新调整观测样本的权重。

我的 Notebook 在相同的预处理流程和相同的训练/验证划分下，对 Random Forest（代表 Bagging）和 XGBoost（代表 Boosting）进行了受控比较。这种设计至关重要：任何结果差异都必然反映学习算法本身的特性，而非数据准备或评估方式的不同。

表 1 总结了来自 `outputs/tables/personalised_improvement_summary.csv` 的关键结果，包含模型名称、验证集宏-F1、验证集准确率、泛化差距（训练 F1 减去验证 F1）、各类 F1 以及训练时间四个模型的完整对比。

Table 1: 受控监督模型对比（相同流程和划分）。

模型	验证 F1	验证 Acc	Gap	High F1	Low F1	Std F1	时间 (s)
Baseline LR	0.7238	0.7342	0.0146	0.7665	0.6490	0.7558	–
Random Forest	0.7708	0.7877	**0.2292**	0.7875	0.7095	0.8154	57.91
XGBoost	0.8144	0.8371	0.0155	0.8905	0.6944	0.8583	67.64
调参后 XGBoost	0.8520	0.8700	0.1219	0.9084	0.7620	0.8854	142.65

Gap = train_F1 − val_F1（即过拟合差距）。

结果为理论预测提供了有力证据。Random Forest 取得了完美的训练宏-F1（1.0000），但验证宏-F1 仅 0.7708，泛化差距高达 0.2292。这种严重过拟合也已在 Notebook 输出的 Random Forest 混淆矩阵中得到视觉确认。相比之下，XGBoost 的训练宏-F1 为 0.8297，验证宏-F1 为 0.8144，差距仅 0.0155。差距的对比令人印象深刻：RF 的过拟合程度大约是 XGBoost 的 15 倍。

表 1 中按类别 F1 列进一步揭示了细节。调参前，RF 在少数类 Low 上（F1=0.7095）优于未调参的 XGBoost（F1=0.6944），但这一优势在 XGBoost 调参后消失。Optuna 调参后，XGBoost 的 Low 类 F1 升至 0.7620，相比调参前提升了 +0.0676，远高于 RF 的 0.7095。这表明 Boosting 的序列残差修正机制更适合学习本数据集中各类别之间的非线性决策边界。Bagging 的方差降低机制无法弥补 RF 全生长树在混合数值和类别特征空间中引入的偏差，这就是 Random Forest 在此处表现不佳的原因。

2. 超参数优化

我使用 Optuna 配合 TPE（树结构 Parzen 估计器）采样器进行了 30 次试验，目标是最大化验证集宏-F1。搜索空间覆盖了 9 个 XGBoost 超参数：n_estimators（100–500）、max_depth（3–10）、learning_rate（0.01–0.3，对数尺度）、min_child_weight（1–10）、subsample（0.5–1.0）、colsample_bytree（0.5–1.0）、gamma（0–5）、reg_alpha（10^{-4}–10，对数尺度）以及 reg_lambda（10^{-4}–10，对数尺度）。离散和连续参数混合且存在多维交互，使得穷举网格搜索在计算上不可行；TPE 通过对好试验和坏试验配置的密度建模，并将后续搜索引导至有前景的参数区域，从而高效地探索搜索空间。

第 22 次试验取得了最佳验证宏-F1 0.8520，相比未调参的 XGBoost 基线（0.8144）提升了 +0.0376。该试验的最优配置为：n_estimators= 276、max_depth= 9、learning_rate≈ 0.192、subsample≈ 0.707、colsample_bytree≈ 0.799、reg_lambda≈ 5.0、gamma≈ 2.5。这些值符合预期：适中的学习率配合大树的深度和大量估计器，使模型能够拟合复杂交互，而 0.7–0.8 左右的 subsample 和 colsample 比率提供了正则化效果。图 1 显示了 Optuna 参数重要性图，证实了结构参数和学习率主导了优化过程。

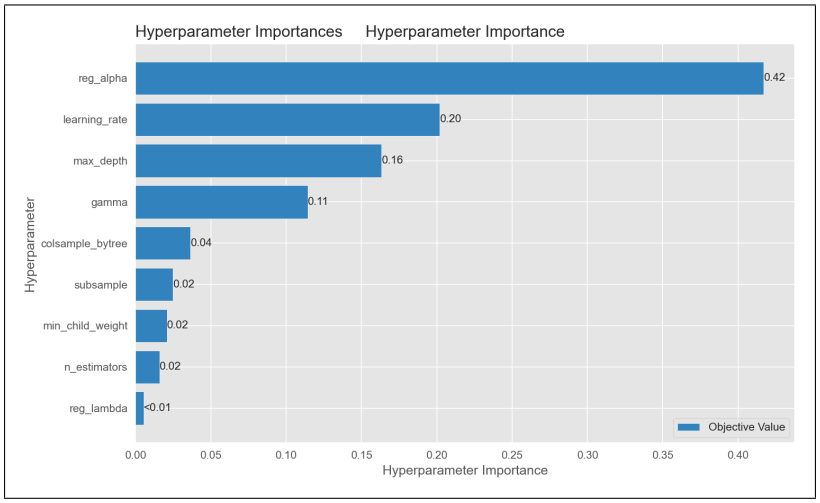

Figure 1: Optuna 超参数重要性图。条越长表示对验证集宏-F1 的影响越大。

表 1 中按类别 F1 的变化值得特别关注，因为宏-F1 对所有三个类别赋予同等权重。Optuna 将 Low 类（少数类）F1 从 0.6944 提升至 0.7620（+0.0676），High 类 F1 从 0.8905 提升至 0.9084（+0.0179），Standard 类从 0.8583 提升至 0.8854（+0.0271）。这种全面改善表明 TPE 成功优化了类别平衡的目标，而非过拟合到多数类。调参后的模型在没有任何类别性能下降的情况下实现了这一目标，而这正是宏-F1 所奖励的。

3. K-Means 与 GMM 对比

K-Means 将每个样本 x_i 分配给质心 μ_c 使 $\|x_i - \mu_{c_i}\|^2$ 最小的簇 $c_i \in \{1, \ldots, k\}$，这是硬分配：每个样本只属于一个簇，没有不确定性或部分成员身份的概念。GMM（高斯混合模型）采用了根本不同的方法，将数据建模为 k 个多元高斯分布的混合：$p(x) = \sum_{j=1}^{k} \pi_j \mathcal{N}(x \mid \mu_j, \Sigma_j)$，其中 π_j 为混合比例。每个样本获得每个成分的后验概率 $p(c_j \mid x_i)$，从而实现软分配：一个样本可以部分属于多个簇。在保险风险场景中，申请人档案在各风险等级之间自然重叠，而非形成孤立的群体，因此软分配更符合领域实际。

表 2 展示了来自 `outputs/tables/clustering_comparison.csv` 的完整聚类结果，涵盖 k=2 到 k=8。列依次为：k、K-Means 惯性、K-Means 轮廓系数、GMM 对数似然、GMM BIC、GMM AIC、GMM 轮廓系数。K-Means 的轮廓系数在所有 k 上持续偏低，峰值仅为 0.2015（k=8 时）。这证实了即使是最优的 K-Means 配置也无法在数据中找到分离良好的球形簇。GMM 获得了明显更高的轮廓系数：k=2 时为 0.4142，k=5 时为 0.4015，约为最佳 K-Means 值的两倍。在 k=2 时，GMM 的轮廓系数 0.4142 对比 K-Means 的 0.1740 特别说明：数据中的二簇结构本质上是概率性的（重叠的高斯成分）而非离散的（质心定义）。

Table 2: 完整聚类对比（k=2 到 k=8）。

k	K-Means 惯性	K-Means 轮廓	GMM BIC	GMM AIC	GMM 轮廓
2	1,092,962	0.1740	−359,251	−362,062	**0.4142**
3	1,018,587	0.1732	−1,103,445	−1,107,666	0.2977
4	953,249	0.1808	−1,938,815	−1,944,446	0.3964
5	889,285	0.1964	−1,997,256	−2,004,298	0.4015
6	818,951	0.1768	−2,349,766	−2,358,217	0.2468
7	777,658	0.1971	−2,394,381	−2,404,243	0.3110
8	691,941	**0.2015**	−2,510,221	−2,521,493	0.1726

表 2 中的 GMM BIC 列显示随着 k 增大 BIC 单调下降（变好），这是预期的，因为增加成分总能改善对训练数据的拟合。但 BIC 同时惩罚模型复杂度，所以下降速度会随 k 增大而放缓，表明边际收益递减。K-Means 惯性曲线没有明显的" 肘部"（转折点），表明数据中不存在自然聚类数——这进一步说明数据不具备分离良好的球形结构。总体而言，GMM 在大多数 k 值上持续更高的轮廓系数表明：保险申请人确实形成了具有软边界的概率亚型。这验证了硬分配与软分配的概念区别：GMM 捕获了 K-Means 无法表示的风险档案重叠性质。重要的是，两种聚类方法都不打算替代监督分类器——它们服务于不同的目标，无监督分析完全是探索性的。

4. 个性化改进反思

我的必做类别是 类别 **A**：数据质量与缺失值处理。在任何建模之前，我进行了 EDA，在多个列中发现了大量缺失值。五个缺失率最高的列分别是：net_monthly_income_gbp、avg_payment_delay_days、monthly_investment_gbp、prior_debt_products、account_tenure（分别为 30.6、19.0、21.1、7.6、4.3 百分比）。我没有将缺失值视为噪声而简单使用中位数填充，而是添加了五个二进制缺失指示特征——每个上述列各一个——同时保留中位数填充。这种方法基于一个假设：缺失的模式本身可能携带信息——缺失的收入值可能表明财务不稳定或失业，这在保险中是一个合法的风险信号。

添加这五个缺失指示特征后，验证宏-F1 从 0.8520（Optuna 调参模型）提升至 0.8529（类别 A XGBoost）。收益较小（+0.0009），但考虑到调参后的模型已经很强并接近特征空间所隐含的性能上限，这一提升仍有意义。更重要的是，这一提升证实了假设：缺失值携带行为信号——在金融应用中，缺失的收入数据并非随机发生，因此具有合法的预测性。这也展示了一个重要的方法论教训：即使是小的改进，也应被审视以判断它们是反映了真实信号还是过拟合。

对于可选类别，我通过软投票（平均预测类别概率）结合 Random Forest 和调参后的 XGBoost，实现了 类别 **D**：软投票集成。集成模型取得了验证宏-F1 0.8510，低于类别 A 模型（0.8529）和单独的调参 XGBoost（0.8520）。这一结果很有启发性：它表明模型多样性本身不足以带来集成改善。两个基学习器的预测画像差异很大——RF 严重过拟合而 XGBoost 校准良好——将它们结合反而稀释了 Boosting 模型的优势，而非补充它。在实践中，有效的集成通常要求基学习器既个体表现强又在错误模式上具有多样性。因此我的最终模型选择是基于严格验证宏-F1 证据，选择类别 A 的 XGBoost。

所有建模步骤的关键前提是数据泄漏控制。在任何模型训练之前，我使用单特征决策树交叉验证对所有可用特征进行了筛查。特征 bureau_risk_index 取得了单特征宏-F1 0.9999——一个极高的分数，表明接近完美的类别分离。这立即触发了泄漏检测阈值（设为 0.85），该特征在所有进一步实验之前被移除。这一步至关重要：没有移除泄漏特征，表 1 中的所有高验证宏-F1 分数都将被人为夸大，每个模型比较都会失效。泄漏检测还说明了一个更广泛的重要原则：在应用机器学习中，即使某个特征看起来改善了性能，也必须先评估它与目标的关系，然后才能接受。

5. AI 使用声明

在整个课程作业中，AI 工具仅在有限的支持角色中使用：协助环境调试（解决包导入和 GPU 配置问题）以及 LaTeX 格式设置以生成最终文档。实验设计、泄漏检测决策、受控模型比较、个性化改进策略以及对表格、图表和指标的所有书面解读均来自我自己的 Notebook 结果。未声称任何隐藏测试性能；CSV 文件（`test_result_1234560.csv`）遵循作业说明中要求的文件名和列顺序，仅因提交格式需要而生成。