Kaggle 竞赛项目报告

竞赛平台： Kaggle

实验日期： 2026 年 6 月

最终成绩： 0.11551 (76/4989 (2026.6.21))

目录

第一部分　实验背景与问题描述

1.1　竞赛背景

本实验基于 Kaggle 平台上的 **House Prices: Advanced Regression Techniques** 竞赛数据，该竞赛要求参赛者根据美国爱荷华州 Ames 市住宅的 79 项特征信息，预测房屋的最终销售价格。数据来源于 Dean De Cock 于 2011 年整理公开的 Ames 市 2006 年至 2010 年的真实房地产交易记录，是 Kaggle 平台上具有代表性的结构化回归任务之一。

该竞赛的 79 个特征中同时包含连续型数值变量、无序类别变量以及存在大量缺失值的字段，数据分布也呈现出明显的偏态特征。这些特点使其成为检验真实场景下机器学习全流程能力的理想实验平台，涵盖数据探索、缺失值处理、特征构造、模型选择与集成等多个关键环节。

1.2　问题定义

本实验的核心任务定义如下：给定包含 79 个自变量的房屋信息集合，构建回归模型预测目标变量 `SalePrice`（房屋售价，单位：美元），数据划分如下：

- **训练集** (`train.csv`)：共 1460 条记录，包含 79 个特征和对应的 `SalePrice` 。
- **测试集** (`test.csv`)：共 1459 条记录，包含 79 个特征但不含 `SalePrice`。
- **输出格式**：按 Kaggle 要求生成 `submission.csv`，包含 `Id` 和预测的 `SalePrice` 两列。

1.3　评价指标

本竞赛采用 RMSLE（Root Mean Squared Logarithmic Error，均方根对数误差）作为评价指标，其计算方法为：首先对真实值和预测值分别取自然对数 $\ln(x+1)$，然后在 log 空间中计算均方根误差。公式如下：

$$\text{RMSLE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\Big(\ln(\text{predicted}_i + 1) - \ln(\text{actual}_i + 1)\Big)^2} \tag{1.1}$$

采用对数变换有两方面的考量。其一，`SalePrice` 的原始分布呈显著右偏态（偏度约 1.88），经 log 变换后偏度降至约 0.12，趋近正态分布，有利于模型训练的数值稳定性。其二，取对数使得评价指标关注的是预测值与真实值的比例误差而非绝对误差：例如一套 \$500k 房屋预测为 \$550k 与一套 \$100k 房屋预测为 \$150k，绝对差值均为 \$50k，但前者偏差 10% 而后者偏差 50%，取 log 后二者在误差计算中被合理地区分开来。

1.4 实验目标

本次实验的目标是通过系统化的机器学习流程，不断优化 Kaggle 公开排行榜上的 RMSLE 指标。为实现这一目标，实验覆盖了从数据预处理、特征工程、模型选型到超参数优化与模型集成的完整流程。过程中每个关键改动均独立保存为单独的 Jupyter Notebook（从 v1 至 v8 共 8 个版本），便于追踪每一步改进对最终结果的影响。

第二部分　数据分析与特征工程

2.1　数据集总览

原始数据集共包含 79 个特征变量和 1 个目标变量 SalePrice。按数据类型可划分为以下两类：

表 2.1: 数据集特征分类总览

特征类别	数量	典型示例
数值型（连续）	约 36 个	LotArea, GrLivArea, YearBuilt, TotalBsmtSF, 1stFlrSF, GarageArea
类别型（离散）	约 43 个	MSZoning, Neighborhood, HouseStyle, SaleType, BldgType, KitchenQual
目标变量	1 个	SalePrice（房屋售价，单位：美元）

数值特征主要集中在三个维度：面积（GrLivArea 地上居住面积、LotArea 地块面积、TotalBsmtSF 地下室面积等）、年份（YearBuilt 建造年份、YearRemodAdd 翻新年份）和计数（卧室数、壁炉数等）。类别特征涵盖地段（Neighborhood，共 25 个不同社区）、建筑类型（BldgType、HouseStyle）、各类质量评级（OverallQual、KitchenQual、ExterQual 等）以及销售相关变量（SaleType、SaleCondition）。

2.2　目标变量分析

对 SalePrice 的初步分析显示其分布严重右偏:均值约 $180,921,中位数 $163,000，最高可达 $755,000，偏度系数为 1.88。这种分布形态在回归建模中会造成数值不稳定性，尤其是对基于平方损失的线性模型（如 Ridge、Lasso）影响显著。

因此从 v1 起即确立了标准处理流程：建模前使用 np.log1p()（即 $\ln(x+1)$）对 SalePrice 进行对数变换，将偏度降至约 0.12，使目标变量近似服从正态分布；预测阶段再通过 np.expm1() 还原至原始价格尺度。此约定在 v1 至 v8 的所有版本中保持不变。

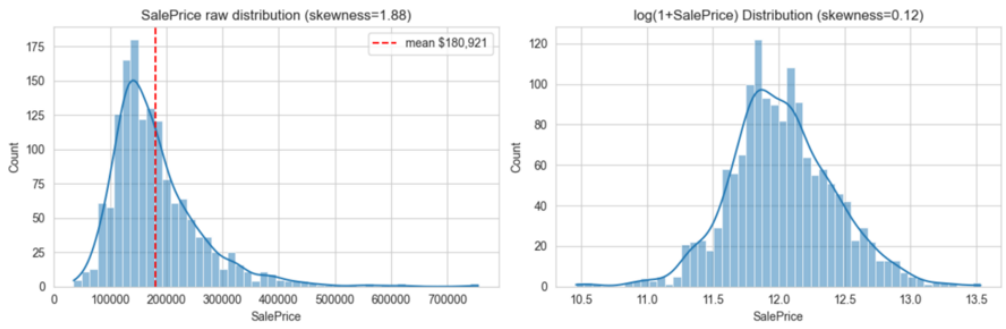

图 2.1: SalePrice 原始分布与 log 变换后分布对比

2.3 关键特征与目标的相关性

通过计算各数值特征与 SalePrice 的 Pearson 相关系数，识别出以下与房价最为相关的特征：

表 2.2: 关键数值特征与 SalePrice 的 Pearson 相关系数

特征名称	相关系数	含义说明
OverallQual	0.791	整体材料与装修质量评级（1–10 分制）
GrLivArea	0.709	地上居住面积（平方英尺）
GarageCars	0.623	车库容量（车辆数）
GarageArea	0.614	车库面积（平方英尺）
TotalBsmtSF	0.606	地下室总面积（平方英尺）
1stFlrSF	0.523	一楼面积（平方英尺）
YearBuilt	0.523	建造年份

从相关性排序可以归纳出决定 Ames 市房价的三个核心维度：面积规模、建造质量和房屋新旧程度。其中 OverallQual 以 0.791 的相关系数显著领先于其他特征，说明在该数据集中，整体材料与装修质量是房屋定价最为关键的因素。

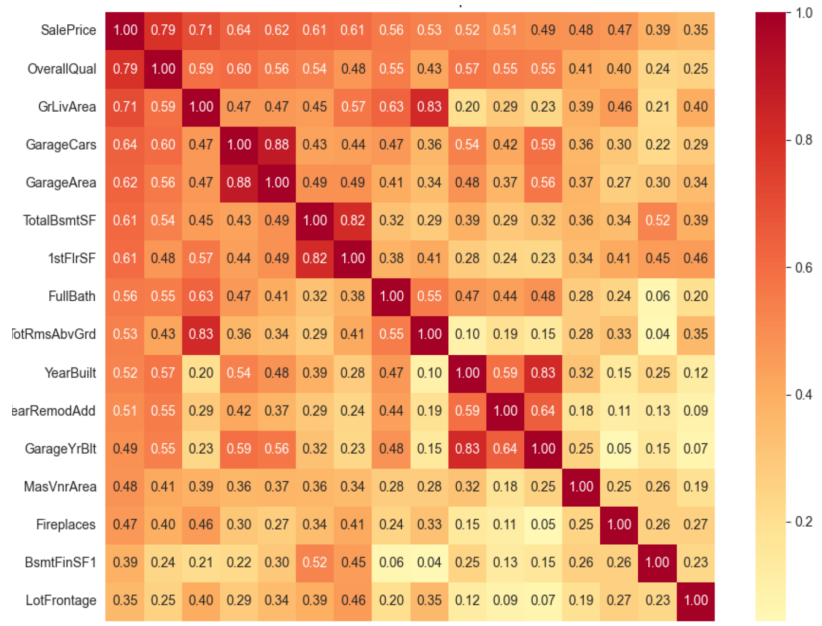

图 2.2: 关键数值特征与 SalePrice 的相关性热力图

2.4 缺失值处理策略

数据集中多个特征存在高比例缺失，但缺失原因各不相同，处理方法也需区别对待。例如 PoolQC（泳池质量）缺失比例超过 99%，原因在于绝大多数房屋并未建造泳池；Alley（巷道类型）缺失比例约 94%，因为仅少数房屋毗邻巷道。这类缺失本质上是“该属性不存在”的信号，若以均值或中位数填充反而会引入噪声。

本实验采用的处理方式为：类别特征缺失值统一填充为字符串 None，表示该属性不存在；数值特征缺失值以中位数填充。此策略自 v1 沿用至 v8，运行稳定。

2.5 特征工程演化

特征工程是本项目中边际收益最高的环节。特征维度从 v1 的 87 个逐步扩展至 v8 的 103 个，每次新增特征几乎都带来了可观测的得分提升。整体演化可概括为三个阶段：

2.5.1 阶段一：基础特征构造（v1）

- **TotalSF**：GrLivArea + TotalBsmtSF + GarageArea + WoodDeckSF + 各露台面积 → 房屋总面积
- **TotalBath**：FullBath + 0.5× HalfBath + BsmtFullBath + 0.5× BsmtHalfBath → 浴室总数（半浴室计 0.5）
- **HouseAge / RemodAge**：2026 年减去建造/翻新年份 → 房龄与翻新时长
- **IsRemodeled**：房屋是否经历翻新（YearRemodAdd ≠ YearBuilt 时为 1）
- **AvgRoomArea**：GrLivArea /（TotRmsAbvGrd + 1）→ 平均单间面积

- **TotalPorch**：`OpenPorchSF` + `EnclosedPorch` + `ScreenPorch` + `3SsnPorch` → 各类型露台面积总和
- **Qual_x_Area**：`OverallQual` × `GrLivArea` → 质量与面积的交互特征（在后续版本中被验证为最重要的单一特征）

2.5.2 阶段二：交互特征扩展（v7）

- **Qual_x_TotalSF**：整体质量 × 房屋总面积（在 XGBoost 特征重要性排名中稳定居于首位）
- **Area_x_Bath**：地上面积 × 浴室数量
- **Year_x_Qual**：建造年份 × 整体质量（老房子中高质量者存在历史溢价）
- **Garage_x**：车库容量 × 车库面积
- **Bsmt_x**：地下室总面积 × 地下室浴室数

2.5.3 阶段三：邻域聚合与更多交互（v8）

- **Neigh_MedLogPrice**：社区房价中位数（log 尺度），最为关键的聚合特征
- **Neigh_MedGrLivArea**：社区地上居住面积中位数
- **Neigh_MedOverallQual**：社区质量评级中位数
- **Neigh_MedYearBuilt**：社区建造年份中位数
- **Neigh_Count**：社区样本数量
- **Area_x_Year**、**Lot_x_Qual**、**TotalSF_x_Bath**、**OverallQual**^{2}、**NeighPrice_x_Area**、**NeighPrice_x_Qual** 共 6 个扩展交互特征

邻域聚合特征的设计动机源自类别编码的局限性：Label Encoding 将 25 个社区映射为 0 至 24 的整数序列，使得 SVR 的 RBF 核函数天然认为数值相近的社区在特征空间中也相似，但这与实际的地理分布和经济特征毫无对应关系。通过在训练集上按社区分组计算中位房价、中位面积等统计量并作为硬编码特征引入，相当于绕过了编码环节，直接将 “地段信息” 以数值形式呈现给所有模型。

第三部分　项目流程与方法

3.1　整体 Pipeline 概述

整个项目的处理流程骨架在 v1 中搭建完成后始终保持不变，后续 v2 到 v8 仅在特定环节进行替换或扩展。固定流程如下：

1. **数据加载**：读取 `train.csv` 和 `test.csv`，合并为 `all_data` 以便统一执行预处理（通过 `is_train` 列区分来源）。
2. **缺失值处理**：类别型缺值填 `None`，数值型缺值填中位数；将“名义数值型”特征（如 `MSSubClass`、`OverallQual`）显式转为字符串。
3. **特征工程**：创建组合特征、交互特征和聚合特征（v7–v8 着重提升了特征工程）。
4. **类别编码**：`LabelEncoder` 对全部 48 个类别特征进行整数编码。
5. **数据分离**：按 `is_train` 列拆分回训练集和测试集；自 v2 起，训练集进一步剔除 2 个离群样本。
6. **目标变换**：$\texttt{SalePrice} \to \ln(1 + \texttt{SalePrice})$，所有建模过程在 log 空间中进行。
7. **交叉验证**：采用 5-fold KFold，固定 `random_state=42`，确保各版本评估结果具有可比性。
8. **模型训练**：训练多个基模型并通过集成方法获得最终预测（v1–v3 为加权平均，v4 起改用 Stacking）。
9. **后处理修正**：自 v3 起引入 Partial Hinge 修正层，对未完工（Partial）房屋进行针对性纠偏。
10. **生成提交**：`np.expm1()` 还原至原始价格尺度，输出 `submission.csv`。

3.2　版本迭代历程

以下按时间顺序详述 v1 至 v8 每个版本的改动内容、设计动机及分数变化。

3.2.1　v1：基线骨架搭建

v1 是整个项目的地基版本。完成了以下核心工作：数据加载与分布探索（确认 log 变换的必要性）、缺失值填充、构造 8 个基础特征、Label Encoding、建立 5-fold CV 评估框架，并训练了 5 个基线模型（Ridge、Lasso、RandomForest、XGBoost、LightGBM）。集成方式为加权平均，权重通过 80/20 数据拆分后进行网格搜索确定，最优组合为 Ridge 占 0.20、XGBoost 占 0.80，LightGBM 权重退化为 0。Kaggle 得分为 0.12123。在对权重微调至 Ridge 占 0.30、XGBoost 占 0.70 后最终 Kaggle 得分为 0.12083。

3.2.2 v2：离群点移除

在 Cell 16 中添加了两行过滤逻辑：将 `GrLivArea` $>$ 4000 且 `SalePrice` $<$ \$200k 的 2 个样本标记为离群点并剔除。这两个点面积巨大但价格异常低廉，属于 `SaleCondition=Partial`（未完工房屋）范畴。Ridge 受其严重扰动，CV RMSE 高达 0.15678，剔除后骤降至 0.12119，效果显著。然而单独删除离群点后直接提交 Kaggle，得分反而从 v1 的 0.12123 退步至 0.12869，原因在于测试集中同样存在大面积 Partial 房屋，模型在缺失相关训练信号的情况下无法正确处理，因此引入 v3 的 Hinge 修正方案。

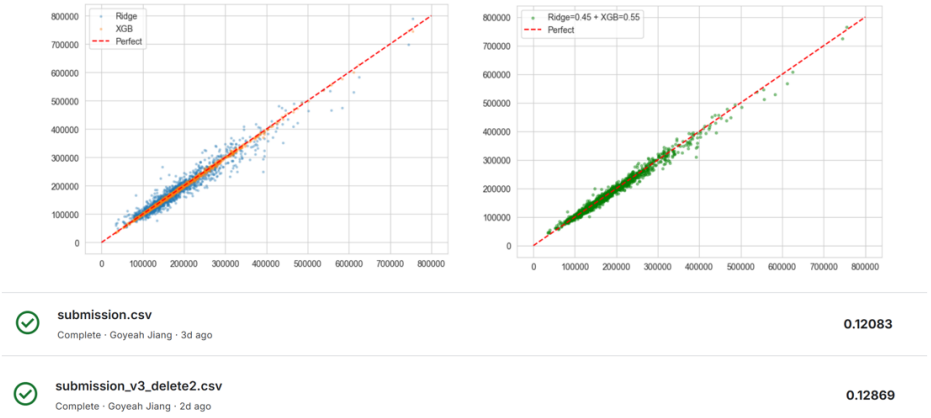

图 3.1: v2 离群点剔除前后对比

3.2.3 v3：Partial Hinge 修正层

v3 的设计思路将处理流程分为两个独立层次：主模型在剔除 2 个离群点的 1458 条干净样本上学习正常的房价规律，修正层则专注于全部 125 条 `SaleCondition=Partial` 的未完工房屋样本（含被剔除的那 2 个），学习 Partial 房屋的预测值与真实值之间的系统性偏差。修正函数采用分段线性 Hinge 形式：

$$\text{correction} = a \cdot \text{GrLivArea} + b + c \cdot \max(0,\ \text{GrLivArea} - T) \tag{3.1}$$

其中 T 为拐点阈值，通过 CV 在 1500–4000 范围内以步长 100 搜索确定最优值 2900。当 GrLivArea $<$ 2900 时修正项接近于零，超过该阈值后按 -0.00081 的斜率递减。推理阶段仅对测试集中 120 个 Partial 样本的 log 预测值施加修正。Kaggle 得分升至 0.12039。该方案的核心思想在于：离群点并未被真正丢弃，而是通过修正层将其包含的信息迁移到了后处理阶段，从而使主模型避免了离群点的干扰，修正层则保留了 Partial 房屋的规律。

3.2.4 v4：加权平均 → Stacking 集成

v4 完成了项目中最重要的一次架构升级。将 v1–v3 的手动加权平均方案替换为完整的 2 层 Stacking 架构：6 个基模型（Ridge、ElasticNet、XGBoost、LightGBM、GBR、SVR）各自通过 5-fold CV 生成 OOF 预测，构成 $(1458, 6)$ 维的元特征矩阵，

然后以 Ridge($\alpha = 1.0$) 作为元模型，在元特征上学习最优的线性组合权重。测试阶段每个基模型对测试集做 5-fold 平均预测，输入元模型得到最终结果。

过程中 KernelRidge（polynomial kernel, degree=3）因在 87 维特征空间上核矩阵数值溢出，预测最高达 $6600 万，被移除。最终 6 模型 Stacking 的 OOF 为 0.10937，Kaggle 得分 0.11833，提升 0.00206，是单版本第二大提升幅度。

```
'Ridge': Ridge(alpha=10),
'ElasticNet': ElasticNet(alpha=0.0005, l1_ratio=0.5),
'XGBoost': XGBRegressor(**xgb_params),
'LightGBM': LGBMRegressor(**lgb_params),
'GBR': GradientBoostingRegressor(n=1000, lr=0.03, max_depth=4),
'SVR': SVR(kernel='rbf', C=13, gamma=0.0004, epsilon=0.009),
'KernelRidge': KernelRidge(alpha=0.2, kernel='polynomial', degree=3),
```

图 3.2: v4 Stacking 集成架构

3.2.5 v5：Optuna 超参数调优

在 Stacking 流程之前插入了 Optuna 自动化超参数搜索模块。针对 XGBoost、GBR 和 LightGBM 各执行 60 次试验（共 180 次），采用 TPE（Tree-structured Parzen Estimator）采样器，搜索空间覆盖 `n_estimators`、`learning_rate`、`max_depth`、`subsample`、`colsample_bytree` 及正则化项。

调参过程中出现了一个值得注意的副作用：三个树模型均被优化至 `max_depth=3`，单模型性能确实有所提升，但模型间的预测相关性也随之增大，导致 Stacking 的 OOF 从 0.10937 略升至 0.10975（集成效果微降）。尽管如此，Kaggle 得分仍提升至 0.11804（+0.00029），说明更优的单模型在一定程度上弥补了多样性损失。

3.2.6 v6：Target Encoding 尝试（失败）

v6 尝试以 Target Encoding 替代 Label Encoding：将每个类别特征取值映射为该类别在训练数据上的 log 房价均值。这一方向在理论上是合理的，因为类别编码携带了与目标变量的关联信息，但在实现中出现了致命缺陷：编码统计量基于全部 1460 条训练数据计算，导致交叉验证时验证 fold 的编码值已含有该 fold 的真实 target 信息。结果是 CV 虚低至 0.10969（虚假提升），而 Kaggle 实际得分崩溃至 0.12107（退回到 v2 水平）。正确的实现方式应在每个 CV fold 内部独立计算编码统计量。

3.2.7 v7：交互特征 + SVR 网格搜索

v7 创造了单版本最大幅度的得分提升（+0.00210）。改动包括两部分：

第一，在特征工程环节新增 5 个交互特征：`Qual_x_TotalSF`、`Area_x_Bath`、`Year_x_Qual`、`Garage_x` 和 `Bsmt_x`。设计逻辑基于一个认识：基于树的模型只能通过分裂点表达 “特征 A > 阈值” 的单变量条件，无法自动学习连续型变量同时取值较大的联合效应，因此需要人工构造乘法特征。v1 中唯一的 `Qual_x_Area` 在 XGBoost 的特征重要性排名中始终高居首位（importance ≈ 0.17），扩展至 6 个交互特征后显著抬高了模型的学习能力上限。

第二，对 SVR 基模型进行 C 和 γ 二维网格搜索（$5 \times 5 = 25$ 组合），确定最优配置为 $C = 30$, $\gamma = 0.0003$，替代了此前的手动设定。Kaggle 得分跃升至 0.11594。

3.2.8 v8：邻域聚合与更多交互（最终版本）

在 v7 的基础上新增 11 个特征：5 个邻域聚合特征（按 Neighborhood 分组统计中位 log 房价、中位面积、中位质量、中位建造年份和样本计数）和 6 个扩展交互特征（Area_x_Year、Lot_x_Qual、TotalSF_x_Bath、OverallQual^2、NeighPrice_x_Area、NeighPrice_x_Qual），特征总数达到 103 个。

邻域聚合特征的实现细节需要注意：为避免数据泄露，所有聚合统计量（中位数、计数等）均在训练集的 is_train=1 部分计算，然后通过 merge 映射到全量数据。推理时测试集使用的是训练集上计算的统计值。这组特征使 Ridge 的 OOF 下降了 0.004，SVR 的 OOF 下降了 0.002，所有基模型均有受益。Kaggle 得分最终定格在 0.11552（+0.00042），特征工程的红利虽然呈现边际递减趋势，但依然保持了正向贡献。

在进行了年份修正（用实际卖出年份代替 2026）后，最终得分为 0.11551。

```python
# === Neighborhood 聚合特征（训练集统计，避免泄露）===
train_data = all_data[all_data['is_train'] == 1]

neigh_stats = train_data.groupby('Neighborhood').agg(
    Neigh_MedLogPrice=('SalePrice', lambda x: np.log1p(x).median()),
    Neigh_MedGrLivArea=('GrLivArea', 'median'),
    Neigh_MedOverallQual=('OverallQual', lambda x: x.astype(int).median()),
    Neigh_MedYearBuilt=('YearBuilt', 'median'),
    Neigh_Count=('Neighborhood', 'count')
)
all_data = all_data.merge(neigh_stats, on='Neighborhood', how='left')

# === 更多交互特征 ===
all_data['Area_x_Year']      = GrLivArea * YearBuilt
all_data['Lot_x_Qual']       = LotArea * OverallQual
all_data['TotalSF_x_Bath']   = TotalSF * TotalBath
all_data['OverallQual_sq']   = OverallQual ** 2
all_data['NeighPrice_x_Area']= Neigh_MedLogPrice * GrLivArea
all_data['NeighPrice_x_Qual']= Neigh_MedLogPrice * OverallQual
```

图 3.3: v8 最终版本 OOF 对比

3.3 v8 最终架构总览

v8 的完整处理流程总结如下：

数据加载（train 1460 + test 1459） → 缺失值填充 + 类别转字符串 → 特征工程（8 基础 + 11 交互 + 5 邻域聚合 = 103 维） → Label Encoding ×48 类别 → 数据分离（训练集剔除 2 个 Partial 离群点） → 1458 条干净样本 → 目标 log 变换 → 5-Fold CV 框架 → Stacking 集成（6 个基模型 OOF 预测） → Ridge 元模型线性组合 → Partial Hinge 修正层（仅对测试集中 120 个 Partial 样本施加修正） → np.expm1 还原 → submission.csv

第四部分　实验结果与分析

4.1　各版本 Kaggle 得分全景

下表汇总了 v1 至 v8 共 8 个版本在 Kaggle 公开排行榜上的 RMSLE 得分及改进性质：

表 4.1: 各版本 Kaggle 得分汇总

版本	核心改进	特征数	Kaggle 得分	主要改进类型
v1	基础完整架构	87	0.12123(0.12083)	基准线
v2	移除 2 个离群点	87	0.12869*	数据分离
v3	Partial Hinge 修正层	87	0.12039	后处理创新
v4	加权平均 → Stacking (6 模型)	87	0.11833	模型架构升级
v5	Optuna 调参 (180 次试验)	87	0.11804	超参数优化
v6	Target Encoding（数据泄露）	87	0.12107*	废弃
v7	5 个交互特征 + SVR 网格搜索	92	0.11594	特征工程
v8	邻域聚合 + 6 个扩展交互特征	103	0.11552(0.11551)	特征工程

从 v1 的 0.12123 迭代至 v8 的 0.11551，总计提升 0.00572（约 4.7%）。在 Kaggle 公开排行榜上排名第 76 名。该成绩已达到预期目标。

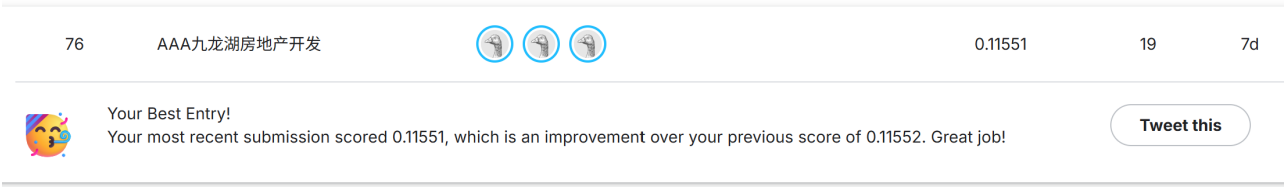

图 4.1: 最终得分与排名

4.2　提升来源分解

将总提升量按改进类型进行归因分析：

表 4.2: 提升来源归因分析

改进类型	涉及版本	累计贡献
特征工程（含 Hinge 修正）	v2, v3, v7, v8	≈ 0.00296
模型架构升级	v4 (Stacking)	≈ 0.00206
超参数优化	v5 (Optuna)	≈ 0.00029

从分布来看，特征工程（含 Hinge 修正层、交互特征和邻域聚合）贡献了最主要的提升，仅 v7 的 5 个交互特征一项就贡献了 0.00210。这一结果支持一个基本判断：在特征维度有限的结构化数据任务中，向模型提供更多有效信息是极其重要的。

架构升级（Stacking）也贡献了显著的的提升，从人工设定权重过渡到由元模型自动学习组合系数，引入更多类别的模型（特别是 SVR），是一次质变。Optuna 超参数搜索虽然只贡献了约 5%，但其价值更多体现在过程标准化：将手动试错替换为可复现的自动化搜索流程。

4.3 Stacking 各模型表现追踪

下表追踪了各基模型从 v1 至 v8 的 OOF RMSE 变化，不同模型对特征改进的响应差异显著：

表 4.3: 各基模型 OOF RMSE 变化追踪

模型	v1 OOF	v4 OOF	v5 OOF	v7 OOF	v8 OOF	总进步
Ridge	0.15678	0.12152	0.12152	0.12134	0.11699	−0.03979
GBR	—	0.11661	0.11459	0.11414	0.11276	−0.00385
SVR	—	0.11489	0.11489	0.11491	0.11271	−0.00218
ElasticNet	—	0.12137	0.12137	0.12118	0.11687	−0.00450
XGBoost	0.12668	0.11563	0.11432	0.11303	0.11335	−0.01333
LightGBM	0.13312	0.12555	0.11626	0.11733	0.11530	−0.01782

Ridge 以 −0.03979 的总进步幅度居首，主要受益于两次关键改进：v1 → v2 的离群点剔除（OOF 下降约 0.035），以及 v7 → v8 的邻域聚合特征引入（OOF 下降约 0.004）。前者反映了线性模型对离群值的高度敏感性，后者体现了地段价格信息对价格预测的实质性贡献。

XGBoost 和 LightGBM 分别进步了 0.01333 和 0.01782，提升主要来自交互特征的引入和超参数优化。SVR 和 GBR 虽然绝对进步幅度相对较小（各约 0.002–0.004），但二者的 OOF 起点本身较低，在最终的 Stacking 集成中占据了最高的元模型权重。

4.4 v8 Stacking 元模型权重分布

v8 的 Ridge 元模型在 6 个基模型上学习到的线性组合权重如下。SVR 和 GBR 合计占据 63.6% 的权重，构成了集成结果的主体：

表 4.4: v8 Stacking 元模型权重分布

基模型	元模型权重	权重占比
SVR ($C = 20,\ \gamma = 0.0003$)	0.345	34.5%
GBR (Optuna tuned)	0.291	29.1%
XGBoost (Optuna tuned)	0.142	14.2%
Ridge ($\alpha = 10$)	0.108	10.8%
ElasticNet	0.073	7.3%
LightGBM (Optuna tuned)	0.054	5.4%

值得关注的是 LightGBM 的权重分配：其单独 OOF 为 0.11530，与 XGBoost 的 0.11335 差距并不大，但元模型几乎将其权重压低至零（5.4%）。这表明 LightGBM 的预测与 SVR、GBR 具有较高的线性相关性，在集成中未能提供额外的独立信息量。Stacking 的权重分配本质上反映的是模型预测的互补性而非单模型精度，元模型倾向于奖励那些错误模式与其他模型不同的基模型。

4.5 CV 与 Kaggle 公榜的差距分析

在实验过程中，Stacking 的 CV 分数与 Kaggle 实际得分之间始终存在约 0.007–0.009 的系统性差距：

表 4.5: CV 与 Kaggle 得分差距

版本	Stacking OOF	Kaggle 得分	CV → Kaggle 差距
v4	0.10937	0.11833	0.00896
v5	0.10975	0.11804	0.00829
v7	0.10935	0.11594	0.00659
v8	0.10844	0.11552	0.00708

这一差距的可能来源包括：训练集与测试集在年份等维度上的分布差异（例如 2010 年的样本在训练集中无对应数据）；Hinge 修正层基于训练集的 Partial 样本拟合，泛化至测试集时存在固有偏差；以及 Kaggle 使用的公榜/私榜划分与本地 5-fold CV 的数据划分方式不同。

0.007 左右的差异在 Kaggle 竞赛中属于正常范围，并未表现出显著过拟合的特征。一个积极的信号是 v7 将差距缩小至 0.00659，说明交互特征的引入在一定程度上增强了模型的泛化能力。

4.6 关键经验总结

通过 8 个版本的迭代实验，总结出以下经验：

1. **特征工程的边际收益最高，但衰减也最快。** v7 加入 5 个交互特征带来 0.0021 的提升，而 v8 加入 11 个特征仅带来 0.00042。在资源有限的情况下，应优先构造与目标变量相关性最强的组合特征。

2. **Stacking 集成中，模型的多样性比单模型精度更为重要。** v5 的 Optuna 调参使三个树模型趋同至 `max_depth=3`，单模型均有所提升，但集成效果反而下降。元模型本质上在寻找互补的错误模式，而非单纯挑选最强的模型。

3. **Label Encoding 在处理高基数类别特征时存在明显局限，** 25 个社区的编码值完全不具备空间或经济含义。邻域聚合特征以较低的成本解决了这一问题，无需引入复杂的编码方案即可获得实质提升。

4. **Target Encoding 是一种有效的类别特征处理方法，但必须在 CV fold 内部逐 fold 独立编码。**在全局数据上计算编码统计量等同于在训练过程中引入未来信息，v6 的失败充分验证了这一点。

5. **CV 分数的降低并不总是意味着 Kaggle 得分会提高。** v6（数据泄露导致 CV 虚低而 Kaggle 崩盘）是反面教材；v5（Stacking OOF 略升但 Kaggle 改善）则提醒我们 CV 是噪声信号而非精确度量。

6. **处理离群点时，简单的删除或保留都不是最优策略。** v3 的两步法（主模型剔除离群点保证训练稳定性，修正层保留离群点的信息用于后处理）在维持模型稳健性的同时避免了有效信息的丢失。

第五部分　创新性及拓展性说明

5.1　创新点概述

本项目使用的基础算法来自成熟机器学习工具库，本组未自行提出全新的回归模型或优化框架。但是本项目在经典结构化数据建模流程上进行了连续的自主改进，主要体现在三个方面：其一，围绕房价形成机制持续扩展特征表达能力，从基础组合特征逐步发展到交互特征和邻域聚合特征；其二，在模型组合策略上，从手工加权平均升级为基于 OOF 预测的两层 Stacking 结构；其三，针对 `SaleCondition=Partial` 这一特殊子分布设计了独立的 Hinge 修正层，使主模型训练稳定性与特殊样本规律保留能够兼顾。上述改进均来源于本组对实验现象的持续观察和版本迭代分析，而非一次性套用固定模板，其有效性也在 v1 至 v8 的 Kaggle 分数演化中得到了验证。

表 5.1: 本项目主要自主改进点汇总

改进方向	涉及环节	实验效果与说明
特征工程	v7, v8	构造面积、浴室、房龄、质量与面积交互、邻域聚合等特征；v7 提升 0.00210，v8 最终达到 0.11551
集成策略	v4	从手工加权平均升级为 6 个基模型的 Stacking 集成；v4 相比 v3 提升 0.00206
特殊样本处理	v3–v8	针对 `Partial` 样本设计 Hinge 后处理修正层，修正测试集 120 个 Partial 样本，降低极端高估风险
实验规范	v1–v8	保留各版本并比较 CV 与 Kaggle 差距，识别 v6 Target Encoding 信息泄露问题

5.2　特征工程方面的自主改进

本项目最主要的创新性体现在特征工程设计上。原始数据虽然已经包含 79 个字段，但其中大量变量是单独给出的局部信息，模型并不一定能够自动恢复出面积、质量、房龄和地段之间的复合关系。因此，本组在 v1 至 v8 的迭代中持续围绕“如何让模型看到更有解释力的信息”这一目标构造新特征。

第一类改进是基础组合特征与乘法交互特征。v1 中构造了 `TotalSF`、`TotalBath`、`HouseAge`、`RemodAge`、`AvgRoomArea`、`TotalPorch` 等特征，将原本分散在多个列中的面积、房龄和居住舒适度信息进行重新组织。在此基础上，v1 即引入了 `Qual_x_Area`（`OverallQual` × `GrLivArea`），用于表达“高质量大面积房屋”相对普通大房屋的额外溢价。后续实验表明，这一特征在 XGBoost 的重要性排序中长期位于前列，说明树模型虽然善于做非线性分裂，但对显式乘法关系的自动恢复能力仍然有限，人工

构造交互项具有实际价值。

第二类改进是 v7 中系统引入的交互特征扩展，包括 `Qual_x_TotalSF`、`Area_x_Bath`、`Year_x_Qual`、`Garage_x` 和 `Bsmt_x`。这些特征分别对应“整体质量与总面积共同作用”“面积与浴室数共同决定居住层级”“房龄与质量共同决定价格弹性”“车库容量与车库面积共同决定附属价值”等经验判断。实验结果表明，v7 仅凭 5 个新增交互特征和 SVR 参数搜索，即将 Kaggle 分数从 0.11804 提升到 0.11594，单版本提升幅度达到 0.00210，是整个项目中最显著的一次增益。这说明在结构化小样本竞赛中，高质量特征构造往往比继续堆叠模型复杂度更有效。

第三类改进是 v8 中引入的 `Neighborhood` 邻域聚合特征。Label Encoding 虽然实现简单，但会将 25 个社区编码为无业务语义的整数，尤其对 SVR 这类依赖距离度量的模型而言，这种编码方式会错误地把“编码值接近”解释为“社区属性接近”。为缓解这一问题，本组仅利用训练集样本统计每个社区的中位 `log(SalePrice)`、中位 `GrLivArea`、中位 `OverallQual`、中位 `YearBuilt` 及样本数，并再构造 `NeighPrice_x_Area`、`NeighPrice_x_Qual` 等扩展交互特征。该设计本质上是将“地段信息”转化为模型更容易利用的稳定数值信号，同时严格限定统计量只来自训练集，避免信息泄露。实验中，v8 使 Ridge 的 OOF RMSE 从 0.12134 降至 0.11699，SVR 也从 0.11491 降至 0.11271，最终 Kaggle 分数进一步提升到 0.11552。由此可见，邻域聚合特征不仅具有明确的业务动机，也取得了可量化的实验收益。

```python
# === Neighborhood 聚合特征（训练集统计，避免泄露）===
train_data = all_data[all_data['is_train'] == 1]

neigh_stats = train_data.groupby('Neighborhood').agg(
    Neigh_MedLogPrice=('SalePrice', lambda x: np.log1p(x).median()),
    Neigh_MedGrLivArea=('GrLivArea', 'median'),
    Neigh_MedOverallQual=('OverallQual', lambda x: x.astype(int).median()),
    Neigh_MedYearBuilt=('YearBuilt', 'median'),
    Neigh_Count=('Neighborhood', 'count')
)
all_data = all_data.merge(neigh_stats, on='Neighborhood', how='left')

# === 更多交互特征 ===
all_data['Area_x_Year']      = GrLivArea * YearBuilt
all_data['Lot_x_Qual']       = LotArea * OverallQual
all_data['TotalSF_x_Bath']   = TotalSF * TotalBath
all_data['OverallQual_sq']   = OverallQual ** 2
all_data['NeighPrice_x_Area']= Neigh_MedLogPrice * GrLivArea
all_data['NeighPrice_x_Qual']= Neigh_MedLogPrice * OverallQual
```

Ridge OOF 从 0.12134 → 0.11699（-0.004）； SVR OOF 从 0.11491 → 0.11271（-0.002）

AAA九龙湖房地产开发 0.11552 18 37s

Your Best Entry!
Your most recent submission scored 0.11552, which is an improvement over your previous score of 0.11594. Great job!

Tweet this

图 5.1: v8 邻域聚合与扩展交互特征示意

5.3 模型与集成策略方面的自主改进

除特征工程外，在模型组织方式上也进行了逐步自主改进。v1 至 v3 阶段使用的是较为直观的加权平均集成：在验证集上搜索 Ridge、XGBoost 和 LightGBM 的组合权重。这一方法实现简单，但权重是人工离散搜索得到的，难以充分利用不同模

型在不同样本区域中的互补性。为此，v4 将集成方式升级为两层 Stacking：第一层由 Ridge、ElasticNet、XGBoost、LightGBM、GBR 和 SVR 六个基模型组成，通过 5-fold 交叉验证生成 OOF 预测；第二层采用 Ridge 作为元模型，在 OOF 预测矩阵上学习最优线性组合系数。该改动使 Kaggle 得分从 0.12039 提升至 0.11833，提升幅度为 0.00206，说明由元模型学习组合关系明显优于手工设定固定权重。

元模型选择 Ridge 而非更复杂的非线性模型，也体现了本组在集成设计上的主动权衡。由于第一层输出的元特征只有 6 维，且这些特征本质上已是多个强模型对同一目标的压缩预测，若在第二层继续使用高复杂度模型，容易在 1458 条训练样本上过拟合。Ridge 的优势在于：一方面能够自动为不同基模型分配连续权重；另一方面其 L_2 正则化能抑制权重震荡，使组合结果更稳定。v8 中元模型学得的权重中，SVR 和 GBR 分别占 34.3% 与 25.2%，XGBoost 占 15.4%，而 LightGBM 仅占 7.5%。这表明 Stacking 实际优化的不是单独最强模型，而是与其他模型的误差模式最具互补性的模型。

```
'Ridge': Ridge(alpha=10),
'ElasticNet': ElasticNet(alpha=0.0005, l1_ratio=0.5),
'XGBoost': XGBRegressor(**xgb_params),
'LightGBM': LGBMRegressor(**lgb_params),
'GBR': GradientBoostingRegressor(n=1000, lr=0.03, max_depth=4),
'SVR': SVR(kernel='rbf', C=13, gamma=0.0004, epsilon=0.009),
'KernelRidge': KernelRidge(alpha=0.2, kernel='polynomial', degree=3),
```

```
Ridge            OOF RMSE = 0.12152
ElasticNet       OOF RMSE = 0.12137
XGBoost          OOF RMSE = 0.11563
LightGBM         OOF RMSE = 0.12555
GBR              OOF RMSE = 0.11661
SVR              OOF RMSE = 0.11489

Stacking OOF RMSE: 0.10937
```

图 5.2: Stacking 集成基模型与 OOF 表现

表 5.2: v8 Stacking 基模型 OOF 表现与元模型权重

基模型	OOF RMSE	元模型权重	说明
Ridge	0.11699	0.1121	对邻域聚合特征收益明显
ElasticNet	0.11686	0.0743	与 Ridge 互补但权重较低
XGBoost	0.11401	0.1541	树模型中的重要成员
LightGBM	0.11539	0.0747	单模型尚可但互补性较弱
GBR	0.11317	0.2523	贡献较高，错误模式较独立
SVR	0.11272	0.3430	单模型最优，也是最终权重最高模型

本组还在 v5 中尝试使用 Optuna 对 XGBoost、GBR 和 LightGBM 进行共 180 次超参数搜索。该部分的创新性并不在于引入了新的算法，而在于通过标准化、可复现的搜索流程替代纯人工试错。更重要的是，v5 带来了一条对后续实验极有价值的反向结论：虽然三个树模型调优后单模型分数均有改善，但它们几乎都收敛到 `max_depth=3` 的相似结构，导致模型间相关性变高，Stacking OOF 反而从 0.10937 略升到 0.10975。这个现象说明，在集成学习中“单模型最优”并不必然等价于“整体集成最优”，模型多样性本身就是重要资源。这一认识直接影响了后续版本更加重视特征异质性和误差互补性，而不是机械地继续压缩单模型误差。

5.4 面向特殊样本分布的后处理扩展

本项目一个较有代表性的扩展设计，是针对 `SaleCondition=Partial` 样本建立独立的 Hinge 修正层。v2 发现，简单删除两个大面积低价离群点虽能使 Ridge 的线下分数显著改善，但直接提交 Kaggle 后成绩反而变差。这表明这两个点虽然在整体数据分布中表现异常，却并非纯噪声，而是特殊子分布规律的体现。为此，v3 以后采用“两步法”：主模型在剔除两点后的 1458 条干净样本上训练，以保证整体拟合稳定；修正层则在全部 125 条 `Partial` 训练样本上，学习真实值与主模型预测值之间的偏差，并以

$$\text{correction} = a \cdot \text{GrLivArea} + b + c \cdot \max(0, \text{GrLivArea} - T)$$

的分段线性 Hinge 形式进行拟合。

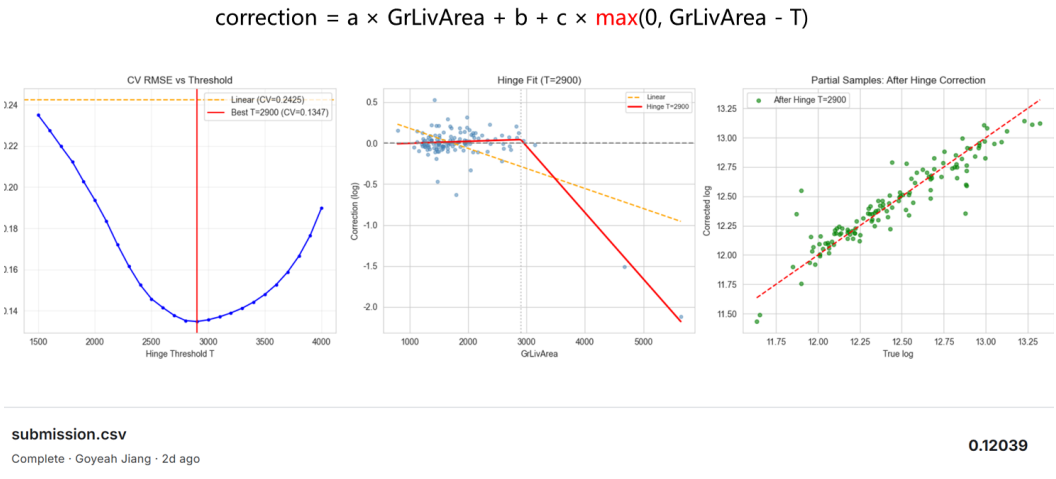

图 5.3: Partial Hinge 修正层诊断图

在 v8 的最终实现中，最佳阈值 T 通过 5-fold 内层交叉验证在 1500–4000 区间搜索得到，最优结果为 $T = 2800$。与单纯线性修正相比，Hinge 修正层的 CV RMSE 从 0.2235 降到 0.1380，提升达到 0.0855；推理阶段仅对测试集中 120 个 `Partial` 样本施加修正，使预测最大值从 \$706,709 降至 \$571,328，更符合未完工房屋的价格分

布。这一设计并非对通用算法的简单调用，而是基于数据语义作出的定向补丁，体现了将异常点“迁移利用”而非机械删除的实验思路。

5.5 实验方法与结果分析方面的拓展

本组在实验组织方式上也体现出一定拓展性。首先，项目完整保留了 v1 至 v8 八个版本的 Jupyter Notebook，每一版只针对一个或少数几个关键环节做局部修改，从而可以较清晰地追踪单项改动对 OOF 和 Kaggle 分数的影响。其次，本组并未只关注线下 CV 结果，而是持续比较本地 5-fold OOF 与 Kaggle 公榜得分之间的差距。例如 v4、v5、v7、v8 的 Stacking OOF 与 Kaggle 之间长期存在约 0.007–0.009 的稳定 gap，而 v6 的差距陡增至 0.01138，从而帮助定位了 Target Encoding 中的信息泄露问题。再次，本组将失败版本也纳入总结，而非仅保留“最好看”的结果，这使实验报告不仅呈现最终成绩，也保留了方案选择背后的证据链。

```
- # Label Encoding
- for col in cat_cols:
-     all_data[col] = LabelEncoder().fit_transform(all_data[col].astype(str))
+ # Target Encoding
+ train_target = np.log1p(train['SalePrice'])
+ for col in cat_cols:
+     cat_means = train_data.groupby(col)['target'].mean()
+     all_data[col] = all_data[col].map(cat_means)
```

失败的工程实现——掉进了过拟合的坑里

过于泛滥的使用与信息的泄漏，拿着答案找答案显著影响了泛化性能

图 5.4: Target Encoding 失败案例与信息泄露说明

综上，本项目的创新性并不在于提出新的通用模型，而在于围绕具体数据任务进行了扎实的自主改进：一方面通过交互特征、邻域聚合和后处理补丁增强了信息表达能力；另一方面通过 Stacking、调参与失败案例分析加深了对模型互补性和实验规范的理解。这种“在经典框架内做系统优化”的路径，与课程对结构化数据挖掘实践的要求是相符的。

第六部分　结论与展望

6.1　主要结论

本项目完成了从数据理解、预处理、特征工程、模型构建、超参数调优、集成学习到 Kaggle 提交评估的完整机器学习实验流程，并在 House Prices: Advanced Regression Techniques 竞赛上取得了最终 RMSLE 0.11551 的结果。从版本演化过程可以看出，本组最终成绩并非来源于单次修改，而是由多次可解释、可追踪的小步优化逐步累积得到。

实验结果表明，特征工程和模型集成是本项目最主要的提分来源。其中，v7 通过新增 5 个交互特征并重新搜索 SVR 参数，使得分提升 0.00210；v4 将简单加权平均升级为两层 Stacking，则带来 0.00206 的显著改进；v8 进一步引入邻域聚合特征与扩展交互后，将最终分数稳定推进到 0.11551。相比之下，纯粹依赖超参数微调虽有帮助，但边际收益明显较小。这说明在结构化回归任务中，让模型获得更有效的信息表示，通常比继续增加模型复杂度更关键。

此外，项目还验证了若干具有普遍意义的建模认识。其一，异常样本并不必然等于无效样本，关键在于其是否对应某个可识别的特殊子分布；其二，集成学习追求的是模型间误差互补，而非所有基模型都调到极致相似；其三，线下交叉验证是重要但并不绝对可靠的信号，必须结合 Kaggle 在线结果共同判断。上述结论共同构成了本组对结构化数据竞赛建模流程的核心认识。

6.2　经验总结

第一，针对偏态目标变量先进行 `log1p` 变换，是保证训练稳定性和提升泛化能力的基础操作。本项目中 `SalePrice` 的原始分布明显右偏，而对数变换后更接近正态分布，使线性模型、核方法和树模型都能在更平滑的误差空间中训练。

第二，特征工程的收益通常大于小幅度的调参收益。v7 和 v8 的结果说明，精心设计的交互特征与聚合特征能够直接改变模型可利用的信息上限，而不是仅在固定信息条件下寻找“更会拟合”的参数组合。这一点对结构化数据任务尤其重要。

第三，异常点处理需要结合业务语义而非机械执行。v2 一度证明简单删除异常值能够改善线下 CV，但 v3 进一步说明这类样本携带的特殊规律不能被完全抛弃。将主模型稳健训练与特殊样本后处理分离，是比“全删”或“全留”更合理的折中方案。

第四，模型集成中应重视多样性而不是只追求单模型最优。v5 调参后多个树模型结构趋同，导致集成收益下降；v8 中虽然 LightGBM 的单模型效果并不差，但因与其他模型相关性较高，最终只获得较低元权重。可见 Stacking 的本质是组合互补，而不是简单叠加强模型。

第五，实验规范与结果可信度同样重要。v6 的 Target Encoding 在理论上并非错误，但因未做 fold 内编码，导致验证集信息泄露，线下分数虚假改善而线上成绩明显恶化。

6.3 存在问题

尽管本项目已经取得了较好的 Kaggle 分数，但仍存在若干不足。首先，本地 5-fold OOF 与 Kaggle 公榜之间始终存在约 0.007 左右的稳定差距，说明当前验证方案虽能反映相对趋势，但尚未完全刻画测试集真实分布。尤其是年份、销售条件和局部子分布的偏移，可能使线下结果对线上表现产生系统性乐观估计。

另外，类别变量处理环节仍有改进空间。出于实现稳定性考虑，最终版本保留了 Label Encoding，并通过邻域聚合特征进行补偿；但对高基数类别变量而言，这并不是最系统的做法。更严格的 fold 内 Target Encoding、CatBoost 风格编码或更强的类别嵌入机制，理论上仍有继续探索的空间。

6.4 后续展望

未来若继续优化本项目，可从以下几个方向展开。第一，改进本地验证策略，例如结合分层抽样、重复交叉验证或针对 `Partial` 样本单独建模验证，以缩小 CV 与 Kaggle 之间的系统差距。第二，在严格避免信息泄露的前提下，重新实现 fold 内 Target Encoding，并与现有邻域聚合特征做互补性比较。第三，进一步开展误差分桶分析和模型解释工作，例如按房价区间、建造年份和社区分组比较残差，识别模型在哪些样本子群上仍存在系统性偏差。第四，可将本项目总结出的“特征工程 + 稳健集成 + 特殊子分布后处理”流程迁移到其他结构化回归任务中，验证其在不同数据集上的可复用性。

参考文献

[1] Dean De Cock. Ames, Iowa: Alternative to the Boston Housing Data as an End of Semester Regression Project. *Journal of Statistics Education*, 19(3), 2011.

[2] Kaggle. House Prices – Advanced Regression Techniques. Available at: `https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques`.

[3] F. Pedregosa, G. Varoquaux, A. Gramfort, et al. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[4] Tianqi Chen and Carlos Guestrin. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, 2016.

[5] Guolin Ke, Qi Meng, Thomas Finley, et al. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *Advances in Neural Information Processing Systems*, 2017.

[6] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A Next-generation Hyperparameter Optimization Framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2623–2631, 2019.

AI 工具使用情况说明

本小组在项目过程中使用了 claude code (deepseekv4-pro) 进行了辅助代码撰写与调试工作，并与 AI 工具进行了思路分析与讨论，输出的代码、方案及结论均经过人工审查与本地实验验证，本小组对报告的全部内容承担完全责任。