一、课题综述

1.1. 课题说明

2252445 王泓烨，线性回归模型实现和分析，模型对比与报告撰写。
2252085 朱亚琨，多项式回归模型实现和分析，数据预处理。
2252699 王鉴戈，XGBoost 模型实现和分析，模型对比与报告撰写。
2250409 罗尹泽，Light GBM 模型实现和分析，数据预处理。
（注：排序与贡献无关，贡献度均分）

1.2. 课题目标

本课题的目标是通过特征工程和数据预处理，对目标数据集进行可视化分析并处理，并尝试构建多种机器学习模型，包括线性回归、多项式回归、XGBoost、LightBGM 等模型，来预测目标变量。研究过程中将分析输入变量间的**关联性**，高相关性变量对只保留一个；筛选与目标变量相关性高的变量，实现**数据降维**；对线性回归、多项式回归和 XGBoost 模型进行**手写实现**；对搭建的模型进行超参数调优，并开展多种**对比实验**，分析和讨论不同学习率、训练方法、正则化、多项式回归最高系数、树的深度等对模型的影响。通过这次课题，我们不仅掌握了回归分析的基本知识，还进一步培养和数据分析和模型构建的能力。

1.3. 课题数据集

本课题的数据集为来自 kaggle 的 House Prices - Advanced Regression Techniques（House Prices - Advanced Regression Techniques | Kaggle）本数据集提供了爱荷华州 Ames 市房屋的详细信息，包括房屋的面积、卧室数量、建筑质量和地理位置等特征。房屋面积涵盖地上和地下室的总面积、车库面积等，反映了房屋的大小和可用空间。卧室数量影响房屋居住功能，而建筑质量则通过评分衡量房屋的施工标准和材料耐久性，房屋的地理位置包括所在街区和与交通设施的距离等因素会显著影响房屋的市场价值。

二、实验报告设计

2.1. 数据准备

1、 从指定路径加载数据，并将加载的数据存储在一个 DataFrame 对象 df_train 中
2、 查看 df_train 的简要摘要，包括每列的非空值数量、数据类型以及内存使用情况
3、 显示 df_train 的前几行数据，快速查看数据的样本
4、 查看 df_train 的维度（行数和列数）
5、 查看 df_train 的缺失值

2.2. 数据预处理

描述性统计分析：对目标变量 SalePrice 进行描述性统计分析，包括计算均值、标准差、最小值、最大值等。使用 sns.distplot 绘制 SalePrice 的分布图（图 1）。计算 SalePrice 的偏度和峰度。

绘制散点图：分别绘制 GrLivArea（地上居住面积）和 TotalBsmtSF（地下室总面积）与 SalePrice 的散点图（图 2、图 3），以观察它们之间的关系。

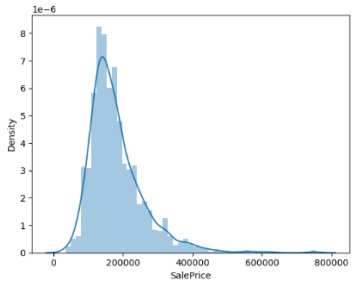


图 1

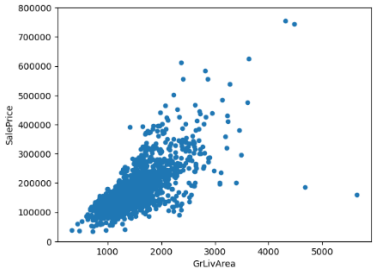


图 2

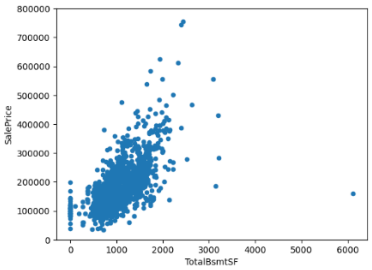


图 3

绘制箱线图：分别绘制 OverallQual（总体质量）和 YearBuilt（建造年份）与 SalePrice 的箱线图（图 4、图 5），以观察它们之间的关系。

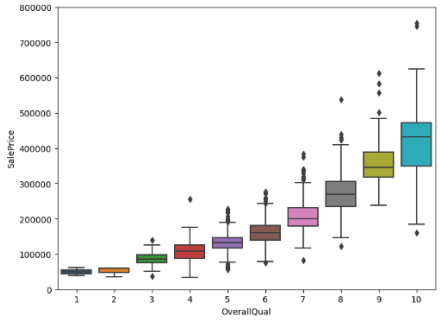


图 4

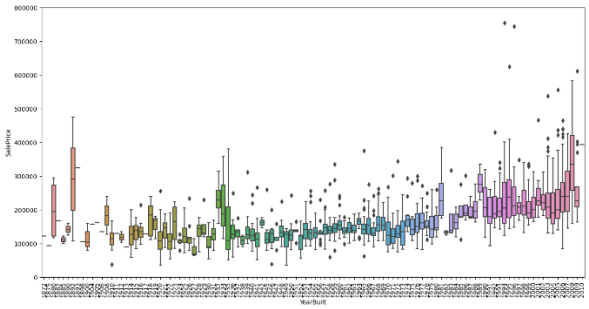


图 5

计算并绘制相关性矩阵的热图：选择数值型列并计算相关性矩阵。使用 seaborn 的 heatmap 函数绘制相关性矩阵的热图（图 6）。

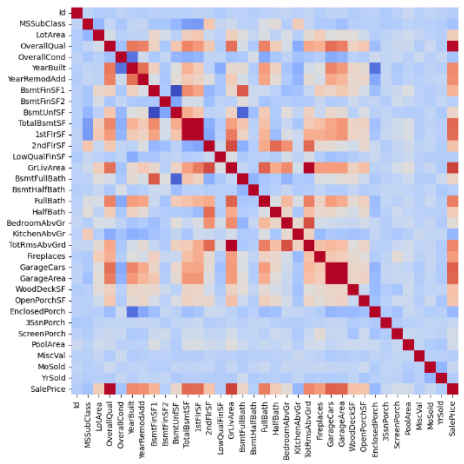


图 6

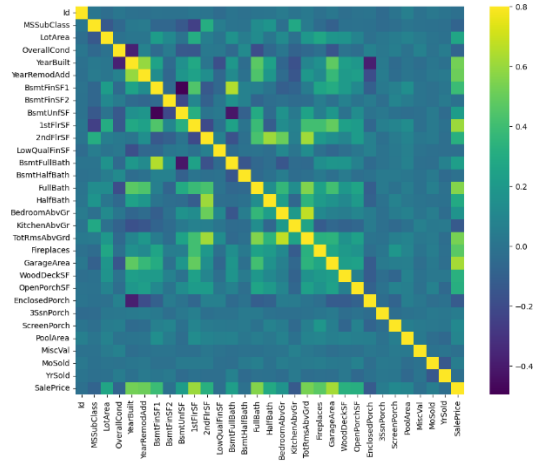


图 7

处理高相关性变量：找出相关性高于 0.7 的变量对。删除高度相关的变量，只保留一个，以减少多重共线性。绘制去除高相关性的变量后的相关性热图（图 7）。

处理缺失值：找出含有缺失值的列，并计算每列缺失值的比例。删除缺失值较多的列。对于缺失值较少的列，使用均值填充数值型列，使用众数填充非数值型列。检查处

理后的数据是否还有缺失值。将处理后的数据保存为 CSV 文件，并打印输出文件路径和最大缺失值剩余数量。

筛选与目标变量相关性最高的变量：读取处理后的数据集。计算数值型列的相关性矩阵。筛选与 SalePrice 相关性最高的 12 个变量（包括 SalePrice 本身）。提取这些变量的数据并保存为新的 CSV 文件。

在上述过程中，涉及到了一些数据降维的方法。

1、计算相关性矩阵：选择数值型列并计算相关性矩阵，以识别与目标变量 SalePrice 相关性较高的特征。

2、处理高相关性变量：找出相关性高于 0.7 的变量对，并删除这些高相关性的变量中的一个，以减少多重共线性。这一步是数据降维的一种方法，通过减少特征之间的冗余，有助于提高模型的泛化能力。

2.3. 模型搭建

2.3.1 线性回归模型

线性回归模型通过拟合一条直线来表示自变量（特征）与因变量（目标）之间的关系，常用于预测和数据分析。

在机器学习领域，线性回归模型记为：

$$\mathrm{y} = \sum_{i=1}^{n} w_i x_i + b = w^T x + b$$

其中，y 是目标变量，w 为模型参数，x 是特征输入，b 是偏置量。线性回归模型通过最小化损失函数使模型拟合。这里采用的损失函数为

$$\mathrm{L(w)} = \frac{1}{2}\sum_{i=1}^{n}\left[y^{(i)} - \sum_{i=1}^{n} w_i x_i^{(j)} - b\right]^2$$

在本次实验中，采用手写梯度下降进行迭代搜索，得出线性回归模型。

除此之外，还手写最小二乘法直接求解最优参数，得出线性回归模型。其公式如下：

$$\theta = (X^T X)^{-1} X^T y$$

具体模型搭建代码在 code/ LinearRegression.ipynb 中。

2.3.2 多项式回归模型

多项式回归模型是线性回归的一种扩展，通过引入多项式特征来拟合非线性关系。多项式回归模型的表达式为：

$$\mathrm{y} = w_0 + w_1 x + w_2 x^2 + \cdots + w_n x^n$$

其中，y 是目标变量，w 是模型参数，x 是特征输入，b 是偏置量。通过引入高次项，模型能够捕捉到更复杂的数据模式。

多项式回归中采用的损失函数与线性回归中相同。搭建的类同样手写最小二乘法和梯度下降法进行模型训练。具体搭建代码在 Polynomial_regression.ipynb 中。

2.3.3 XGBoost 模型

XGBoost 是一种基于梯度提升（Gradient Boosting）的集成学习方法，通过组合多个弱学习器（通常是决策树）来提升预测性能。每棵新树都是在前一棵树的基础上，尝试减少模型的残差。

实现 XGBoost 模型的流程如下：

1、初始化：使用均值等初始化预测值。

2、计算残差：计算当前模型的预测值与真实值之间的差异（残差）。

3、训练新模型：使用残差作为新的目标，训练新的模型来拟合这些残差。

4、更新预测：将新模型的预测值加权后添加到当前预测值中。

5、重复：重复以上步骤，直到满足停止条件（如达到最大迭代次数或误差小于阈值）。

6、预测：最终预测值等于所有模型的预测值的加权平均值。

7、评估：使用评价指标（如 RMSE、R^2 等）评估模型性能。

具体搭建代码在 XGBoost.ipynb 中，分别使用 xgboost 库和手写进行实现。

2.3.4 LightBGM 模型

LightGBM 是一种基于决策树的梯度提升框架，在传统梯度提升树（GBDT）算法的基础上，做了一系列优化，使得模型在处理高维特征和大规模数据时表现优异。

实现的 LightBGM 模型流程如下：

1. 模型参数设置

指定模型的超参数，包括 boosting_type、objective、metric、num_leaves、learning_rate、max_depth 等。

2. 模型训练

调用 lgb.train 方法进行模型训练。指定训练参数、数据集和迭代轮数。在训练过程中，LightGBM 会根据验证集评估性能，并根据需要更新模型。

3. 预测与评估

使用训练好的模型对测试集进行预测，并计算预测结果的均方根误差（RMSE）和 R^2 值。具体搭建代码在 light-gbm.ipynb 中。

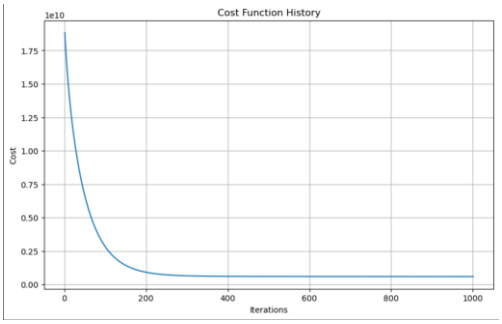
2.4. 模型训练测试与结果

2.4.1 模型准确度的指标

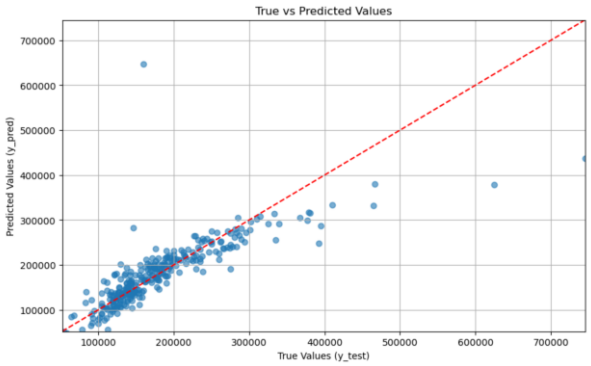
明确衡量模型准确度的指标：MAE、MSE、RMSE 和 R^2 Score

2.4.2 线性回归

建立一个有正则化、有偏置项的线性回归模型，采用梯度下降法进行训练，其中学习率为 0.01，迭代次数为 1000，正则化值为 0.1。训练过程中的 cost 变化曲线如下：



可以看到模型在上述超参数设定下，平滑下降并在迭代 300 次左右逐渐收敛到一个较小的值。说明模型表现良好，过拟合或欠拟合情况不明显，找到了合适的参数。用训练好的线性回归模型进行预测，预测结果如下：



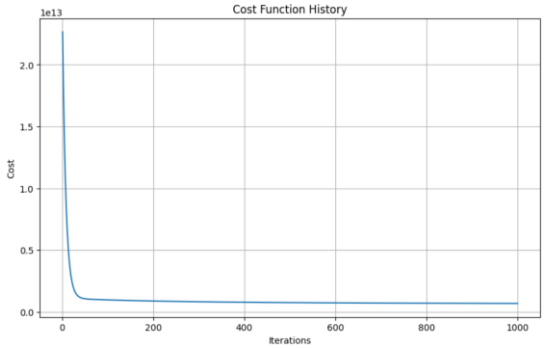
得到 r2 score 为 0.6644474168297945。

线性回归假设自变量与因变量之间存在线性关系。调用 statsmodels 中的库函数得到的 r2 取值为 0.6340292035508076

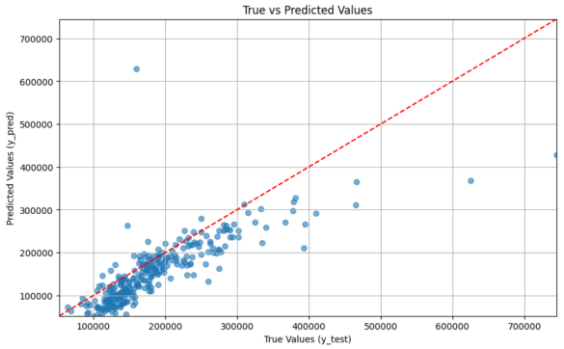
说明预测结果的 r2 值不够接近 1，可能是由于自变量和因变量之间的真实关系是非线性的，线性回归可能无法有效建模。

2.4.3 多项式回归

建立一个无正则化、有偏置项的多项式回归模型，采用最小二乘法进行训练。训练过程中的 cost 变化曲线如下：



用训练好的多项式模型进行预测，预测结果如下：

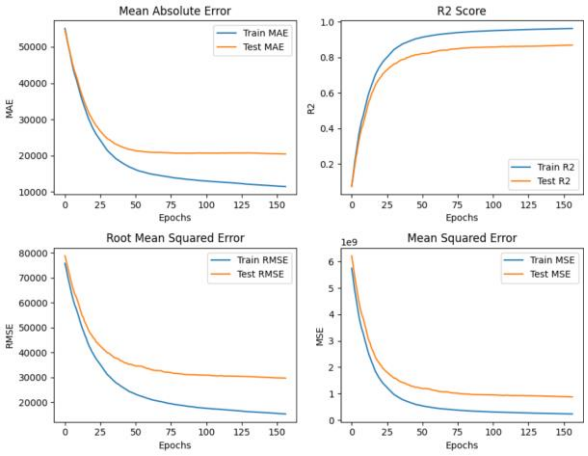


整体来看，虽然有些点云位置有些偏离红线，但大部分点位于红线的周围，说明模型的预测值大致接近真实值，表现出较好的预测效果，r2 score 为 0.8295855029281748

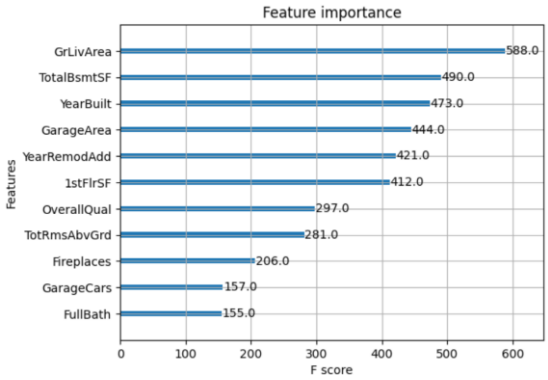
2.4.4 XGBoost

调用 xgboost 库实现：

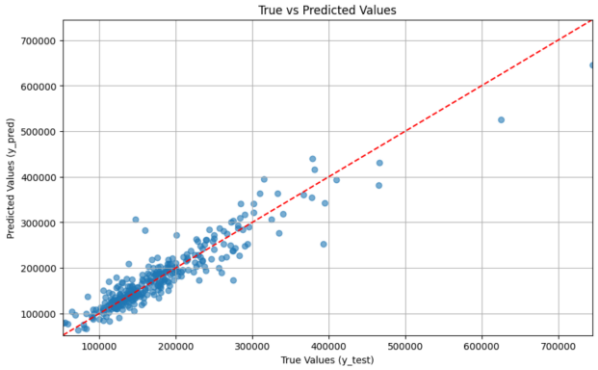
超参数设置如下：损失函数使用均方根误差 ，树的最大深度设置为 5，学习率设置为 0.05，每棵树训练时的特征采样比例 0.3，每棵树训练时的子采样比例 0.7，L1 正则化系数 0.1，L2 正则化系数 0.1。训练过程如下：



可以看到模型在测试集上的表现趋于稳定后，在 157 轮时自动停止了训练，避免了过拟合。为了进一步展示模型训练的情况，同时输出了训练好的模型中参与决策的各特征的重要性占比：



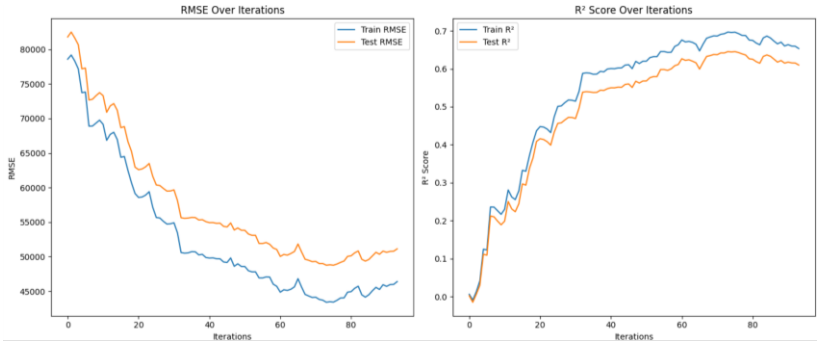
使用训练好的模型进行预测，得到的结果如下：



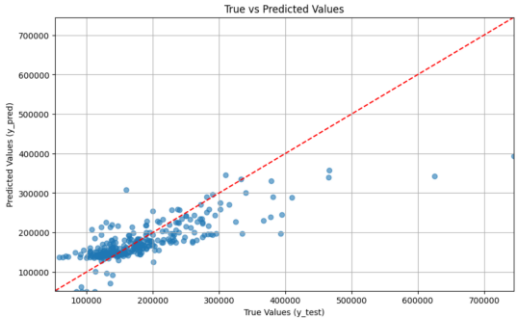
可以看的大部分点紧密沿红色虚线分布，说明模型的预测值大致接近真实值，表现出很好的预测效果。模型最优的各个评级指标为：MAE: 20465.836593000855， MSE 879820549.6560767， RMSE: 29661.769159240597， R2: 0.8687798059859982

手写实现：

训练过程：手写实现中未设置正则化项，其余超参数设置基本相同。可以非常明显的看到，由于训练过程中未使用正则化，模型的训练上会出现较为明显的波动。最终在模型表现趋于稳定后，模型自动停止防止过拟合。



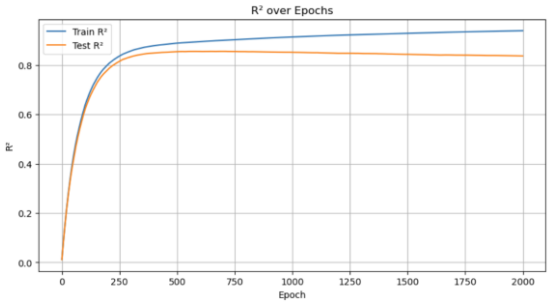
使用训练好的模型进行预测，得到的结果如下：



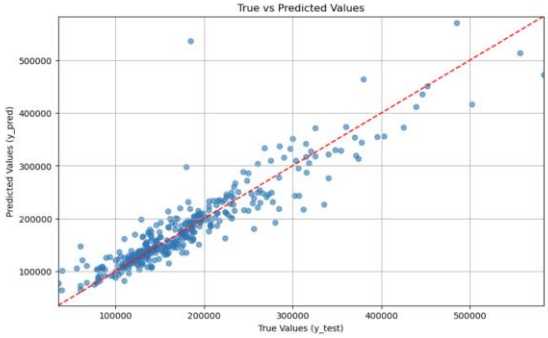
可以看到点云稍稍稍离红线，与 xgboost 库实现的效果相比有些差距，但总体仍然沿红线分布，说明模型仍可以较好的实现该任务。模型最优的 RMSE 为 48772.85571951464，R^2 为 0.6452168838625452，同样和官方库有一定的差距，原因可能在于树的构建、特征处理、超参数的选择等方面。

2.4.5 LightBGM

建立 LightBGM 模型，将 max_depth 设置为 15，learning_rate 为 0.008，num_leaves 设置为 8，lambda_l1（L1 正则化）设置为 1，训练过程如下：



得到的预测结果如下：



整体来看，大部分点位于红色虚线的周围，说明模型的预测值大致接近真实值，表现出

较好的预测效果。模型最优的 r2 score 为 0.8107697304769449

2.5. 分析和优化（要包含对两类模型的结果的比较讨论）

为了得到更好的预测结果，我们需要对模型进行调优，进行一系列对比实验，来选择合适的超参数等。

2.5.1 线性回归

实验 1 训练方法

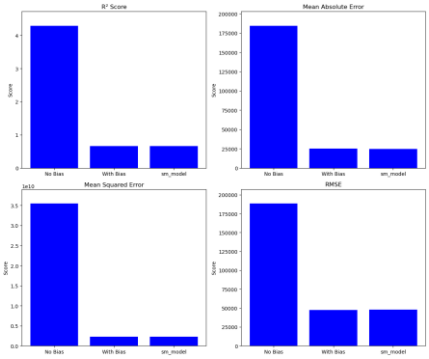
前面的模型训练采用的是梯度下降法，这里用最小二乘法来训练模型，得到的预测结果为：

r2 score: 0.6597331155741537

因此，梯度下降法训练处的模型预测结果更好

实验 2 偏置项

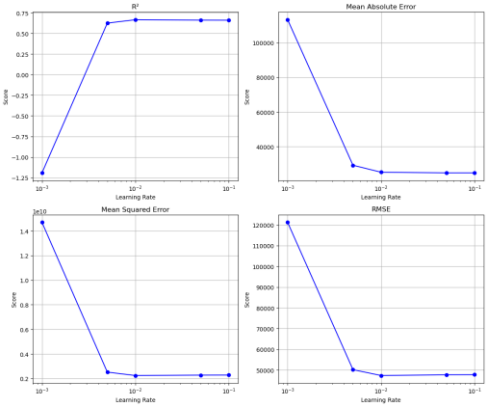
在有正则化的情况下建立线性回归模型，采用梯度下降法进行训练，比较有偏置项、无偏置项和调用 statsmodels 中的库函数时 MAE、MSE、RMSE、R2Score 的值。



可以看到有偏置项时，模型的准确性更高。

实验 3 学习率

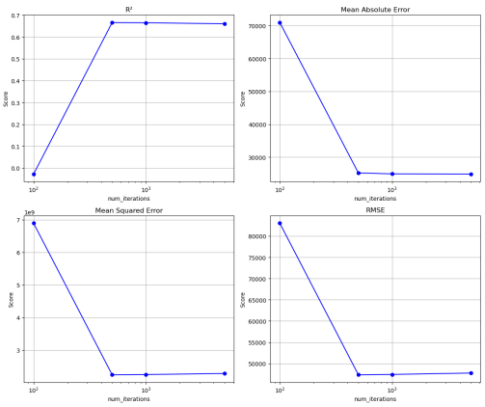
在其余条件相同的情况下，比较学习率为 0.001,0.005, 0.01,0.05, 0.1 时 MAE、MSE、RMSE、R2Score 值的变化。



可以看到 R² 随着学习率从 0.001 到 0.01 急剧上升，之后趋于平稳，在 0.01 附近达到了最佳值。MAE、MSE、RMSE 随着学习率的增加从一个较高的数值迅速下降，最终趋于稳定。

实验 4 迭代次数

在其余条件相同的情况下，比较迭代次数为 100, 500, 1000, 5000 时 MAE、MSE、RMSE、R2Score 值的变化。



R² 随着迭代次数的增加，模型的 R² 值急剧上升，在迭代次数达到 1000 后趋于平稳。MAE、MSE、RMSE 随着迭代次数增加，误差迅速下降，最终趋于稳定。

通过实验对模型进行调优，得到更拟合的线性回归模型

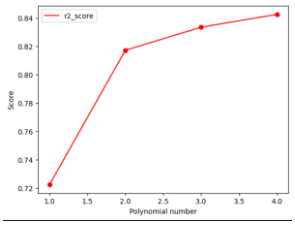
实验 5 正则化参数

与多项式回归中的正则化参数实验相似，在 2.5.2 中具体描述。

2.5.2 多项式回归

实验 1 多项式特征系数

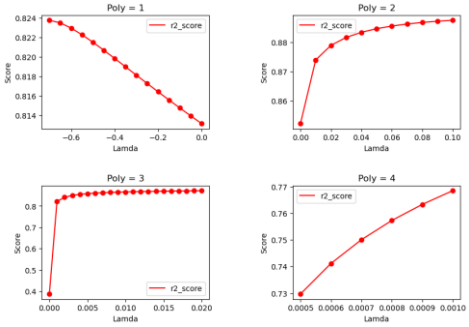
在其余情况相同的条件下，比较多项式维度为 1、2、3、4 时，模型 R2 Score 的变化如下：

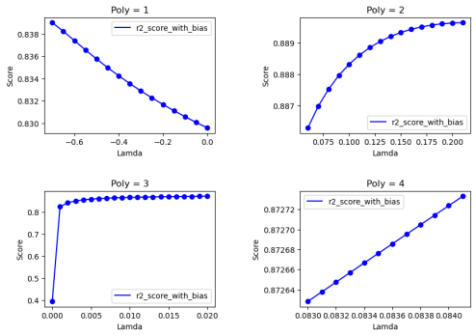


随着多项式阶数增加，模型的拟合效果逐步提升，但提升幅度逐渐减小。这符合一般规律，即模型复杂度的增加通常会带来性能提升，但超过一定复杂度后，可能会面临过拟合的问题。

实验 2 正则化参数

在其余条件相同的情况下，比较正则化参数在[-0.7, -0.65, -0.6, ..., -0.05, 0],[0, 0.01, 0.02, ..., 0.1],[0, 0.001, 0.002, ..., 0.02],[0.0005, 0.0006, 0.0007, 0.0008, 0.0009, 0.001]四个数值集合内，模型有偏置项和无偏置项时，R2 Score 的变化如下：





其中蓝色和红色曲线分别代表包含和不包含偏置项的两种情况。从整体趋势来看，随着多项式度数的增加，模型对正则化的响应发生显著变化：线性模型（Poly=1）显示正则化会降低性能，二次模型（Poly=2）则通过适度正则化获得性能提升，而高阶多项式（尤其是三次和四次）对正则化表现出更高的敏感度，往往只需要很小的正则化参数就能产生显著效果。

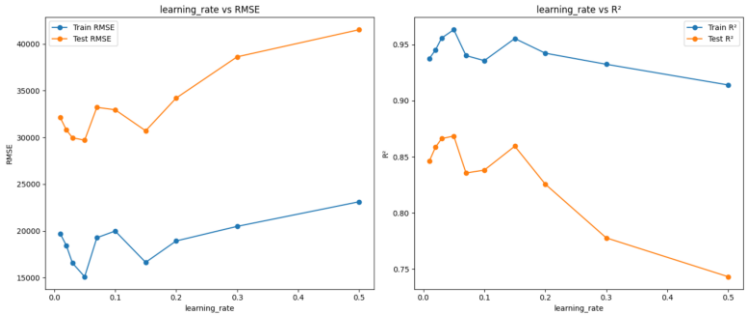
实验 3/4/5 学习率、迭代次数、偏置项

与线性回归中对这三个变量的实验差别不大，在这里不再赘述。具体内容在 Polynomial_regression.ipynb 中。

2.5.3 XGBoost

实验 1 学习率

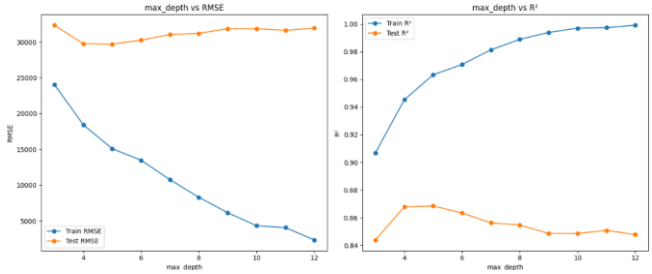
在其余情况相同的情况下，比较学习率在[0.01, 0.02, 0.03, 0.05, 0.07, 0.1, 0.15, 0.20, 0.30, 0.50]中变化时，模型在测试集上的 RMSE 和 R^2 变化如下：



在一定迭代次数情况下，学习率过低，会导致模型训练时间长，模型效果未达到最优情况；学习率过高，则会导致模型过拟合。在该问题场景下，学习率在 0.1 左右是较为合适的选择。

实验 2 树的最大深度（max_depth）

在其余情况相同的情况下，比较树的最大深度（max_depth）在从 3 到 12 的整数中变化时，模型在测试集上的 RMSE 和 R2 变化如下：

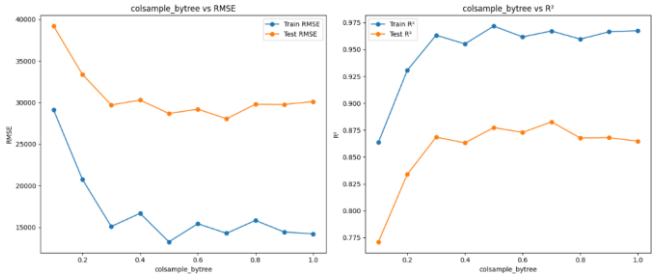


随着树的深度的加深，训练集上的损失越来越小，R2 越来越接近 1，但是测试集上的

损失则先减后增，说明当深度较大时，模型过拟合。在该问题场景下，树的深度选择 4-6 层较为合适

实验 3 每棵树训练时的特征采样比例(colsample_bytree)

在其余情况相同的情况下，比较每棵树训练时的特征采样比例(colsample_bytree) 从 0.1 到 1 的 10 个值中变化时，模型在测试集上的 RMSE 和 R^2 变化如下：

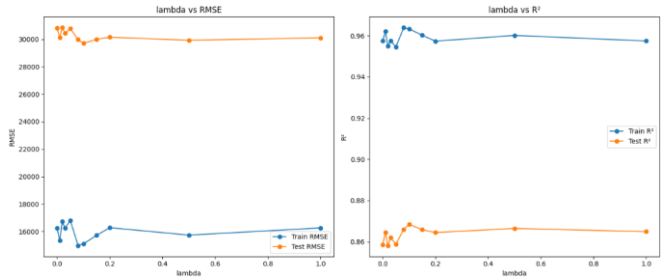


当特征子集增加到一定大小时，模型在训练集和测试集上的表现均趋于稳定；但随着特征子集的增加，模型训练时间显著增长。在该问题场景下，特征子集大小选择接近 0.3(>0.3)时，模型效果最佳。

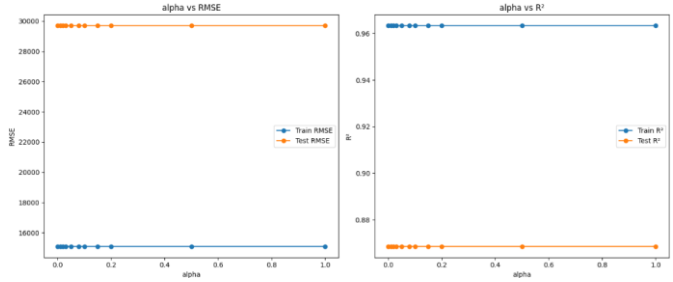
实验 4 正则化参数

在其余情况相同的情况下，比较 L1 正则化参数和 L2 正则化参数分别在[0, 0.01, 0.02, 0.03, 0.05, 0.08, 0.1, 0.15, 0.2, 0.5, 1]中变化时，模型在测试集上的 RMSE 和 R^2 变化如下：

L1 正则化参数：



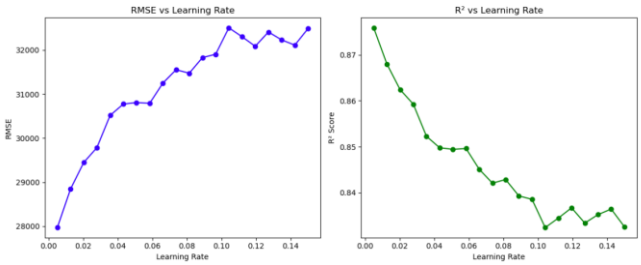
L2 正则化参数：



L1 正则化参数 Lambda 对模型的影响不大,适当大小的正则化系数对模型表现有一定的提示；而 L2 正则化参数 alpha 则对模型几乎没有影响。在该问题情境下，选择 0.05 左右的 L1 正则化系数有助于模型表现的提升

2.5.4 LightBGM

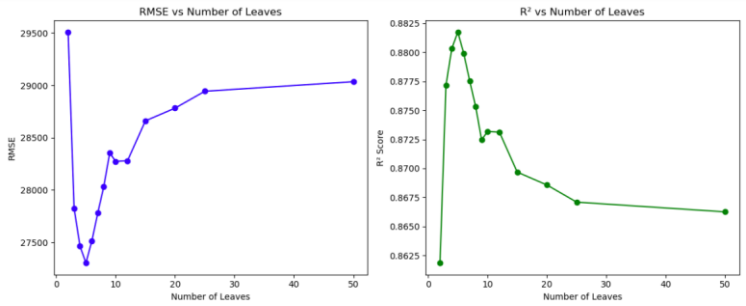
实验 1 学习率



随着学习率从 0 增大到 0.14，RMSE 逐渐增加，R^2 逐渐减小。在低学习率时，RMSE 处于较低水平，R^2 值接近 0.87，而随着学习率增大，RMSE 明显增高，这意味着模型的误差越来越大，性能越来越差。

实验 2 num_leaves

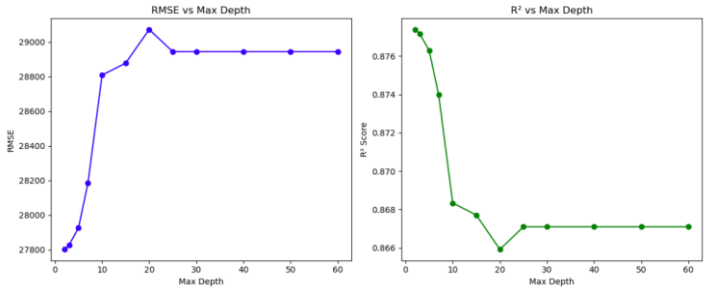
num_leaves 越大，模型越复杂，能够捕捉更多的数据细节。影响：较小的 num_leaves 可能导致模型欠拟合，而较大的值可能导致模型过拟合，增加计算成本。通常要在模型复杂度和性能之间找到平衡。



当叶子节点数量从 0 增加到 10 左右时，RMSE 明显下降，模型预测误差减小，R^2 在叶子节点数量增加到 10 左右时达到峰值，这表明模型的性能在逐渐提升；然而当叶子节点数量继续增大时，RMSE 反而开始增加，R^2 开始下降，表明模型的预测误差增加，性能下降。

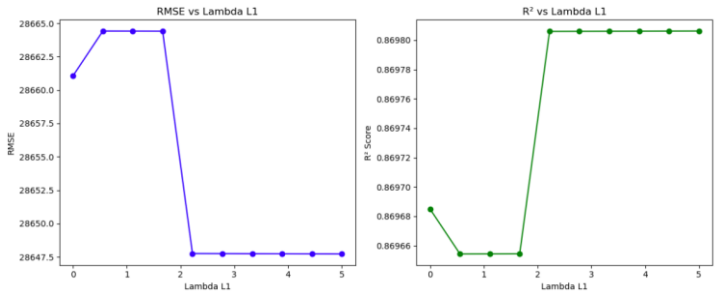
实验 3 max_depth

Max_depth 限制每棵树的最大深度，从而控制模型的复杂度。较浅的树可能无法捕捉到复杂的特征关系，导致欠拟合；较深的树会使模型更复杂，容易过拟合并增加计算成本。设置合适的 max_depth 可以有效地控制模型的拟合能力。



在 max_depth 较小（如 0 到 10）时，RMSE 逐渐降低，R^2 得到提升，达到最高值（约为 0.876）,表明模型性能提升，误差减小。在 max_depth 达到 20 之后，RMSE 显著增加，R^2 迅速下降并趋于稳定,表示模型开始变得不稳定，误差增加；随后，RMSE 保持相对稳定，表明更大的树深度未能进一步降低误差。

实验 4 L1 正则化值



当 Lambda L1 从 0 增加到 2 时，RMSE 有一个非常明显的下降，R^2 快速上升并保持稳定。这意味着模型的预测误差大幅减少，性能提升。当 Lambda L1 超过 2 后，RMSE 保持稳定，不再显著下降，这表明在超过某个阈值后，L1 正则化强度对模型性能不再有显著影响。

根据上述实验结果进行调参，我们将 max_depth 设置为 20，learning_rate 保持为 0.008，num_leaves 设置为 7，lambda_l1（L1 正则化）设置为 1 。

最终得到的 R2 达到了 0.884，比最初的实验结果更为精确。

3. 总结

这次实验围绕回归任务展开，目标是利用特征工程和数据预处理，对数据集进行分析和预测。实验共构建了四种回归模型，包括线性回归、多项式回归、XGBoost 和 LightGBM，以对比它们在预测任务中的表现。

1. **数据预处理**：对数据进行了描述性统计分析、缺失值处理以及降维操作，以提升模型的泛化能力。
2. **模型实现与调优**：
   - **线性回归**：通过梯度下降和最小二乘法手动实现，并调整学习率、迭代次数等参数来优化。
   - **多项式回归**：引入多项式特征来捕捉非线性关系，通过调节多项式次数和正则化参数提升效果。
   - **XGBoost**：使用库和手动实现方式，调节学习率、树深度、特征采样比例等参数，得到了良好的预测结果。
   - **LightGBM**：通过叶子数、学习率等参数的调整，提高了大数据集下的模型效率。
3. **实验结果与优化**：通过多种评价指标（MAE、MSE、RMSE、R^2）评估模型性能。XGBoost 和 LightGBM 在处理大规模数据方面表现出色，且在调参后实现了较高的预测准确性。

此次实验不仅展示了不同回归模型的实现和调优方法，也为团队成员们在数据分析和建模方面提供了宝贵经验。