课 程 报 告

年 级 专 业：2022 级电子信息工程

学 生 姓 名：刘嘉俊

学　　号：20223004426

课 程 名 称：机器学习

任 课 老 师：兰　翔

报告成绩	
教师签名	

海南大学·信息与通信工程学院

School of Information and Communication Engineering, Hainan University

基于随机森林和 GBDT 的房价预测和性能对比

摘 要

本研究旨在基于机器学习中的随机森林（Random Forest）和梯度提升决策树（GBDT）两种集成学习算法，探讨其在房价预测任务中的性能差异。通过对 Kaggle 公开房价数据集（BostonHousing）的分析和处理，研究设计了全面的数据预处理流程，包括缺失值处理、特征相关性分析和非数值型数据转换。随后分别构建随机森林和 GBDT 模型，并通过网格搜索优化超参数以提升模型性能。

实验结果表明，随机森林在处理高维数据和噪声时表现出较强的鲁棒性，具有较快的训练速度和优异的泛化能力；而 GBDT 则在捕捉特征与目标变量间复杂非线性关系上表现出色，其在验证集上的 R^2 达到了 0.91，高于随机森林的 0.89，且 RMSE 更低，显示出更优的预测性能。本研究进一步分析了两种模型的优缺点，提出了模型选择的适用场景，并对未来研究的改进方向进行了展望

关键词：房价预测、随机森林、梯度提升决策树、机器学习、集成学习

Abstract: This study aims to compare the performance of two ensemble learning algorithms, Random Forest (RF) and Gradient Boosting Decision Tree (GBDT), in predicting house prices. Using the publicly available Kaggle housing price dataset （BostonHousing）, a comprehensive data preprocessing pipeline was designed, including handling missing values, feature correlation analysis, and converting non-numerical data. Subsequently, RF and GBDT models were constructed, and hyperparameters were optimized using grid search to enhance model performance.

The experimental results demonstrate that Random Forest exhibits strong robustness in handling high-dimensional data and noise, along with faster training speeds and excellent generalization capabilities. On the other hand, GBDT excels in capturing complex nonlinear relationships between features and target variables, achieving a higher R^2 score of 0.91 on the validation set compared to 0.89 for Random Forest, along with a lower RMSE. This highlights GBDT's superior predictive performance. The study further analyzed the advantages and disadvantages of the two models, provided guidance on model selection for practical applications, and proposed potential directions for improvement in future research.

Keywords: House price prediction, Random Forest, Gradient Boosting Decision Tree, Machine Learning, Ensemble Learning

1 引言

1.1 背景介绍

随着社会经济的不断发展和城市化进程的推进，房价预测逐渐成为房地产市场中一个重要的研究方向。准确的房价预测不仅对购房者的决策有重要意义，还能够帮助房地产开发商优化资源配置，同时为政府制定宏观经济政策提供有力支持。在数据驱动的时代，传统的房价预测方法，如基于统计学的回归分析方法，已经逐渐无法满足高维、非线性、多样化数据的需求。机器学习方法，凭借其强大的数据挖掘能力和对复杂关系建模的优势，成为房价预测领域的重要工具。

集成学习作为机器学习领域的一个重要分支，通过结合多个弱学习器的预测结果来提高模型的泛化性能，展现了显著的优越性。其中，随机森林（Random Forest）和梯度提升决策树（GBDT, Gradient Boosting Decision Tree）是集成学习中两种经典的算法，它们在分类、回归等任务中被广泛应用并表现出了较高的预测能力和稳定性。

1.2 研究目的

本研究的目标是基于随机森林和 GBDT 两种集成学习算法，对房价进行预测，并通过对两种算法性能的对比，探讨其在不同数据场景下的适用性。具体而言，本研究以 Kaggle 提供的“房价预测”数据集为实验数据，通过合理的数据预处理和特征工程构建预测模型，利用随机森林和 GBDT 分别对房价进行建模，并通过实验验证两种算法的拟合能力与泛化能力。

研究中重点解决以下几个问题：

1. 在数据集中的缺失值处理、特征选择和特征编码等预处理环节，如何有效提升模型性能？
2. 随机森林和 GBDT 两种模型在房价预测任务中的性能表现如何？是否存在显著的差异？
3. 在实际应用场景中，如何选择适合的模型以平衡预测精度和模型复杂度？

2 模型方法介绍

2.1 随机森林（Random Forest）

随机森林是一种基于决策树的集成学习方法，它通过构建多个决策树并结合其预测结果来提升模型的泛化性能。随机森林的主要思想是采用“袋装法”（Bagging）和“随机特征选择”来构造每棵决策树。具体来说：

袋装法（Bagging）：随机森林通过对训练集进行有放回的随机抽样（Bootstrap）来构造多个子训练集，确保每棵决策树的训练数据不同，从而增加模型的多样性。

随机特征选择：在构造每棵决策树的过程中，随机森林从特征集合中随机选择一部分特征作为分裂依据，进一步减少模型的过拟合风险。

预测阶段：

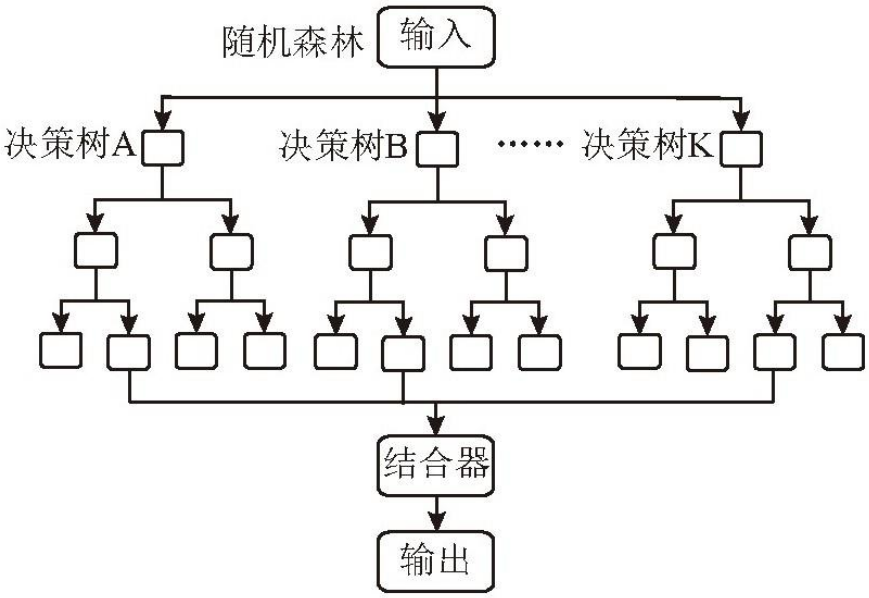
对于分类任务：通过对多个决策树的投票结果取多数来确定最终预测类别。

对于回归任务：通过对多个决策树的预测结果取平均值来得到最终预测值。

随机森林的主要优点包括：能够处理高维数据，具有较强的抗过拟合能力。在不需要特征缩放的情况下，对数据的鲁棒性较高。

可通过调整超参数(如树的数量n_estimators和最大特征数 max_features）来平衡模型的复杂度和性能。

图 1 随机森林决策树构建流程图



在本研究中，学生采用随机森林回归模型对房价进行预测，并通过网格搜索优化模型超参数。

2.2 梯度提升决策树（Gradient Boosting Decision Tree, GBDT）

GBDT 是一种基于决策树的集成学习方法，它通过“加法模型”和“梯度下降算法”逐步构建多个弱学习器（决策树），使模型的预测误差逐步减少。GBDT 的主要思想如下：

1. 加法模型：GBDT 通过将多个弱学习器的预测结果线性加权求和构成最终模型。假设已有一个初始预测值$F_0(x)$，则第 t 轮的预测模型可表示为：

$$F_t(x) = F_{t-1}(x) + \eta \cdot h_t(x)$$

其中，$h_t(x)$表示第 t 轮训练的决策树，η为学习率，用于控制步长大小。

2. 梯度下降： 在每一轮迭代中，GBDT 通过最小化目标函数的残差（即梯度信息）来训练当前决策树，使得模型逐步逼近最优解。

3. 正则化机制：

学习率（Learning Rate）：通过缩小每轮迭代的步长，平衡模型的拟合能力和泛化能力。

子采样（Subsampling）：通过随机选择部分训练样本，增加模型的鲁棒性。

GBDT 在回归任务中的主要优点包括：能够捕捉特征与目标变量之间的复杂非线性关系。对特征的重要性排序具有较强的解释能力。能通过调整超参数（如学习率、最大深度和子采样比例等）灵活控制模型复杂度。

在本研究中，学生采用 GBDT 模型对房价进行预测，并通过网格搜索优化模型超参数。

2.3 模型性能评估指标

为了评估随机森林和 GBDT 模型的性能，本研究选用以下常用指标：

1. 决定系数（R^2）： 决定系数用于衡量模型对目标变量的拟合程度，说明自变量能够通过回归模型所解释的因变量波动的占比。当决定系数接近 1 时，说明自变量对因变量的解释能力非常强，这意味着模型的预测能力就越强。其公式为：

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$$

其中，y_i为真实值，$\hat{y}_i$为预测值，$\bar{y}$为目标变量的均值。R^2 的取值范围为 [0, 1]，值越接近 1，模型拟合效果越好。

2. 均方根误差（RMSE）： RMSE 用于衡量预测值与真实值之间的偏差，它是通过均方误差开根得到的，可以从单位度量的角度来评估模型的表现。误差值越小，得到的预测值与真实值之间的差值也就越小，说明建立的

模型效果越好。其公式为：

$$RMSE = \sqrt{\frac{\sum(y_i - \hat{y}_i)^2}{n}}$$

RMSE 值越小，模型的预测误差越小。

3. 交叉验证（Cross-Validation）： 本研究采用交叉验证（K-Fold Cross-Validation)对模型进行评估,以更好地测试模型的稳定性和泛化能力。通过在验证集上计算 R^2 和 RMSE 来比较模型性能。

2.4 超参数优化

随机森林和 GBDT 模型的性能很大程度上依赖于超参数的选择。为此，本研究通过网格搜索（Grid Search）对模型进行超参数优化。

随机森林的超参数:n_estimators:随机森林中决策树的数量。max_features:用于分裂的最大特征数。max_depth：决策树的最大深度。

GBDT 的超参数：n_estimators：GBDT 中弱学习器（决策树）的数量。learning_rate：每轮更新的学习率。max_depth：单棵决策树的最大深度 subsample：每轮训练中使用的样本比例。

通过网格搜索，选择验证集上性能最优的参数组合以构建最终模型。

3 数据获取与数据预处理

3.1 数据获取

本研究使用的数据集来源于 Kaggle 公开数据集“House Prices: BostonHousing”， BostonHousing 数据集涵盖了 2006 年 1 月至 2010 年 7 月间美国波士顿市的 1460 条带有实际价格标签的房屋交易数据（train.csv）和 1459 条无实际价格标签的房屋交易数据（test.csv）。具体说明如下：

训练集（train.csv）：包含 1460 条记录，提供了房价信息（目标变量 SalePrice）以及 79 个特征，包括房屋的面积、位置、建造年份、装修情况等多种信息。

测试集（test.csv）：包含 1459 条记录，仅包含 79 个特征，未提供房价（目标变量）。用于最终预测房价并提交结果。

每条记录的特征分为以下两类：

数值型特征：如房屋面积（LotArea）、建造年份（YearBuilt）等连续变量。

非数值型特征：如街道类型（Street）、供暖类型（Heating）等分类变量。

图 2 部分特征值信息表

列名	列名含义
Id	样本编号
MSSubClass	建筑分类编号
MSZoning	区域分类（如住宅区、商业区等）
LotFrontage	与街道相邻的房屋正面宽度（以英尺为单位）
LotArea	地块面积（平方英尺）
Street	房屋的街道类型（铺装或未铺装）
HeatingQC	供暖质量与状况
CentralAir	是否有中央空调
Electrical	电气系统类型
1stFlrSF	一楼面积（平方英尺）
2ndFlrSF	二楼面积（平方英尺）
LowQualFinSF	低质量装修面积（平方英尺）
GrLivArea	地面以上的居住面积（平方英尺）
MoSold	售出月份
YrSold	售出年份
SaleType	销售类型
SaleCondition	销售条件
SalePrice	房价（目标变量，美元）

3.2 数据预处理

为了提高模型的预测性能并解决数据集存在的异常和冗余信息，对原始数据进行了以下预处理操作。

3.2.1 特征相关性分析

在建模之前，为了减少冗余特征、提升模型性能，研究了各特征与目标变量 SalePrice 之间的相关性，主要方法包括：计算每个数值型特征与 SalePrice 之间的皮尔逊相关系数。筛选相关性较高的特征（相关系数绝对值 > 0.5）进行深入分析。绘制散点图（如 SalePrice 与 LotArea 的关系图）和箱线图（如 SalePrice 与 MSZoning 的关系图），直观观察目标变量与特征之间的关系。

相关性是衡量两个变量线性关系的一种统计指标，常用的相关性系数包括皮尔逊相关系数和斯皮尔曼相关系数。本实验中，选择皮尔逊相关系数作为度量方法，其计算公式如下：

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2} \cdot \sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

r_{xy}为相关系数，x_i， y_i特征值与目标变量值，$\bar{x}$，$\bar{y}$特征均值与目标变量均值。相关系数的取值范围为 $[-1,1]$，具体含义如下：r>0：正相关，特征值越大，目标值越大；r<0：负相关，特征值越大，目标值越小；｜r｜→1：强相关；｜r｜→0：弱相关或无相关。

特征筛选：删除与目标变量相关性极低或没有意义的特征，例如一些高度冗余或不相关的数值型特征。

对分类型变量，评估其类别分布，并结合房价的均值分析其信息价值。

图 3 房价 SalePrice 统计分布表

统计项	数值
count	1460
mean	180921.2
std	79442.5
min	34900
25%	129975
50%	163000
75%	214000
max	755000

图 4 房价 SalePrice 特征相关系数热力图

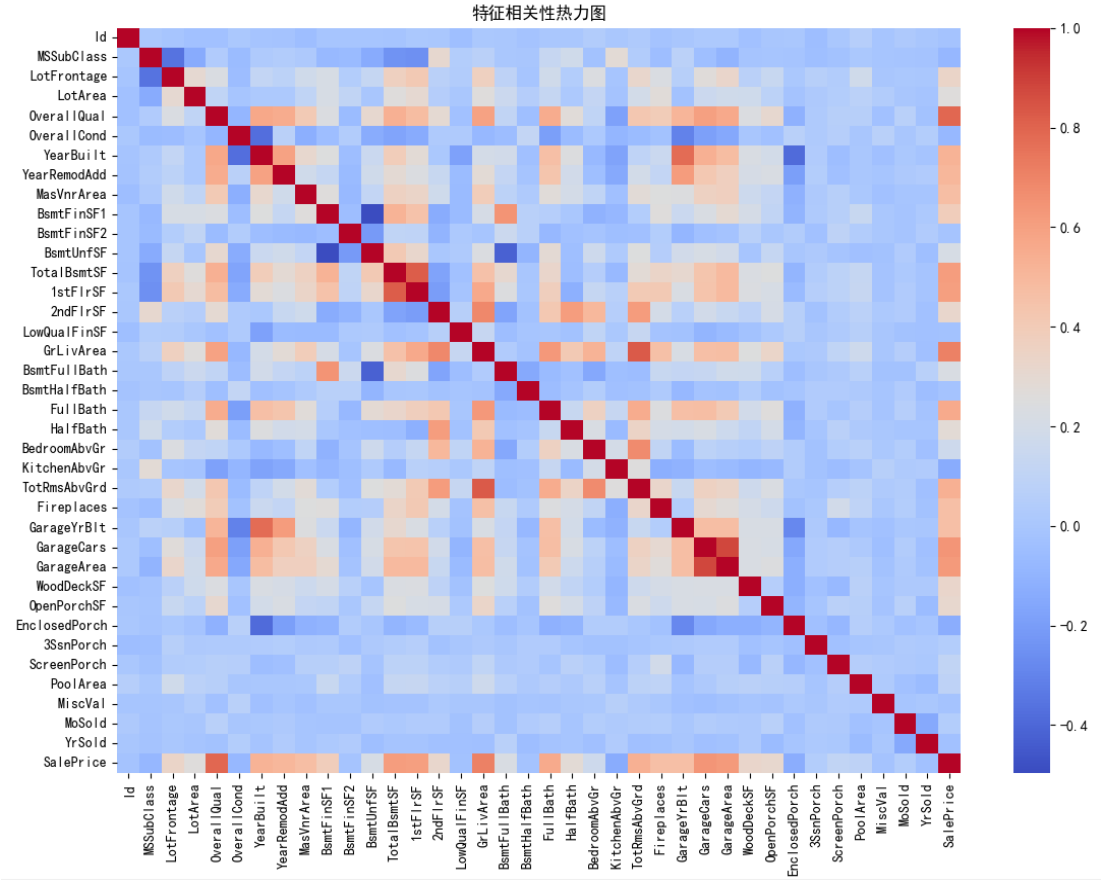


图 5 房价 SalePrice 最相关热力图（前 10 个特征）

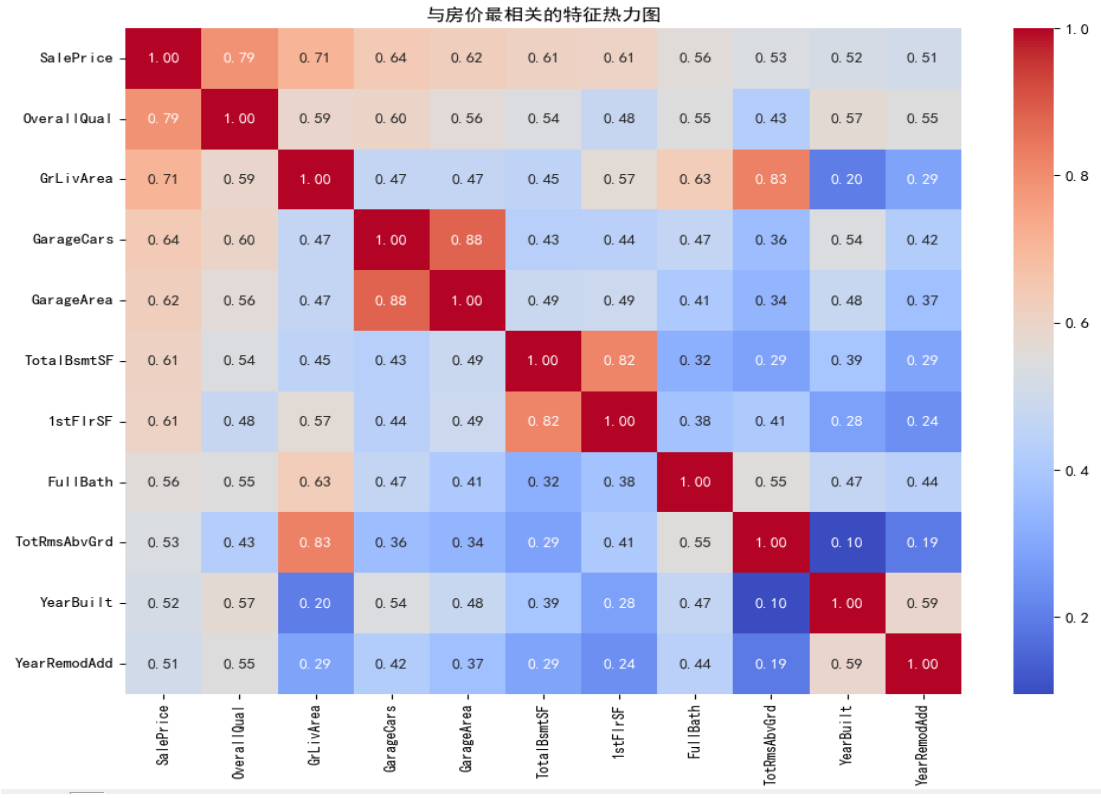


图 6 房价 SalePrice 与特征相关性表

特征名	相关性
SalePrice	1
OverallQual	0.790982
GrLivArea	0.708624
GarageCars	0.640409
GarageArea	0.623431
TotalBsmtSF	0.613581
1stFlrSF	0.605852
FullBath	0.560664
TotRmsAbvGrd	0.533723
YearBuilt	0.522897
YearRemodAdd	0.507101
MasVnrArea	0.472614
Fireplaces	0.466929
GarageYrBlt	0.466754
BsmtFinSF1	0.38642
LotFrontage	0.334771
WoodDeckSF	0.324413
2ndFlrSF	0.319334
OpenPorchSF	0.315856
HalfBath	0.284108
LotArea	0.263843
BsmtFullBath	0.227122
BsmtUnfSF	0.214479
BedroomAbvGr	0.168213
ScreenPorch	0.111447
PoolArea	0.092404
MoSold	0.046432
3SsnPorch	0.044584

图 7 房价 SalePrice 分布图

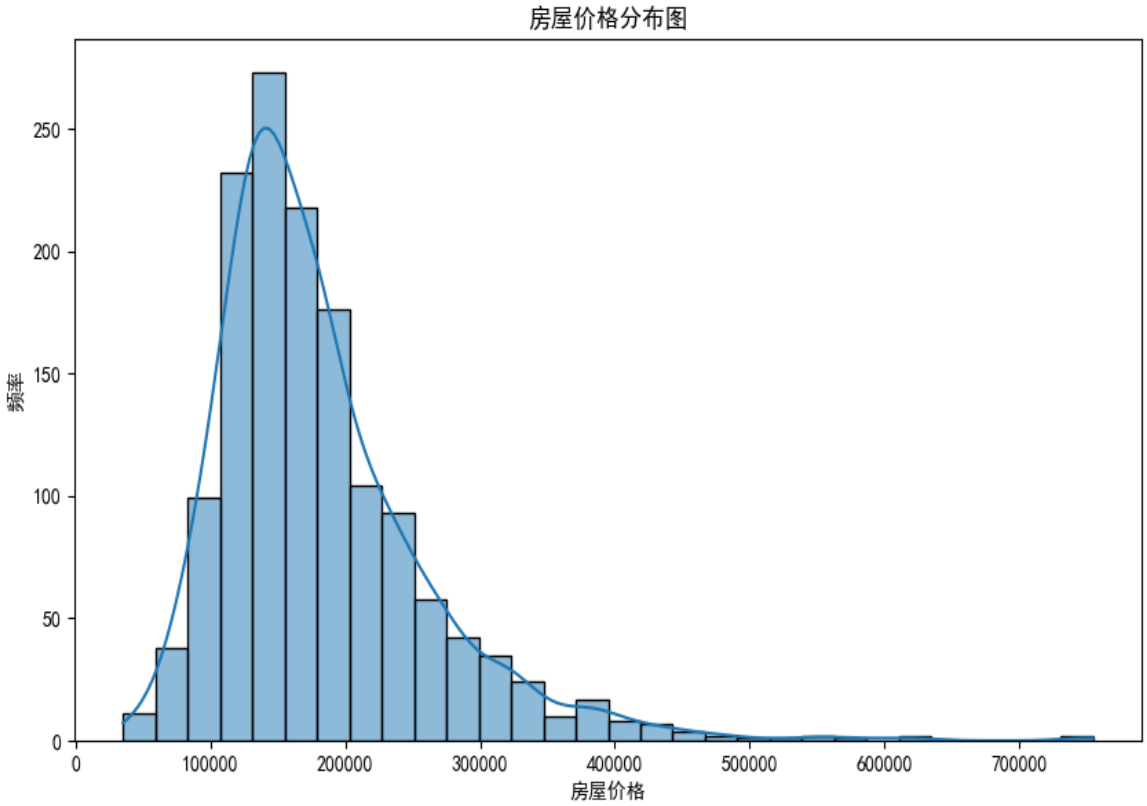


图 8 房价 SalePrice 计数统计表

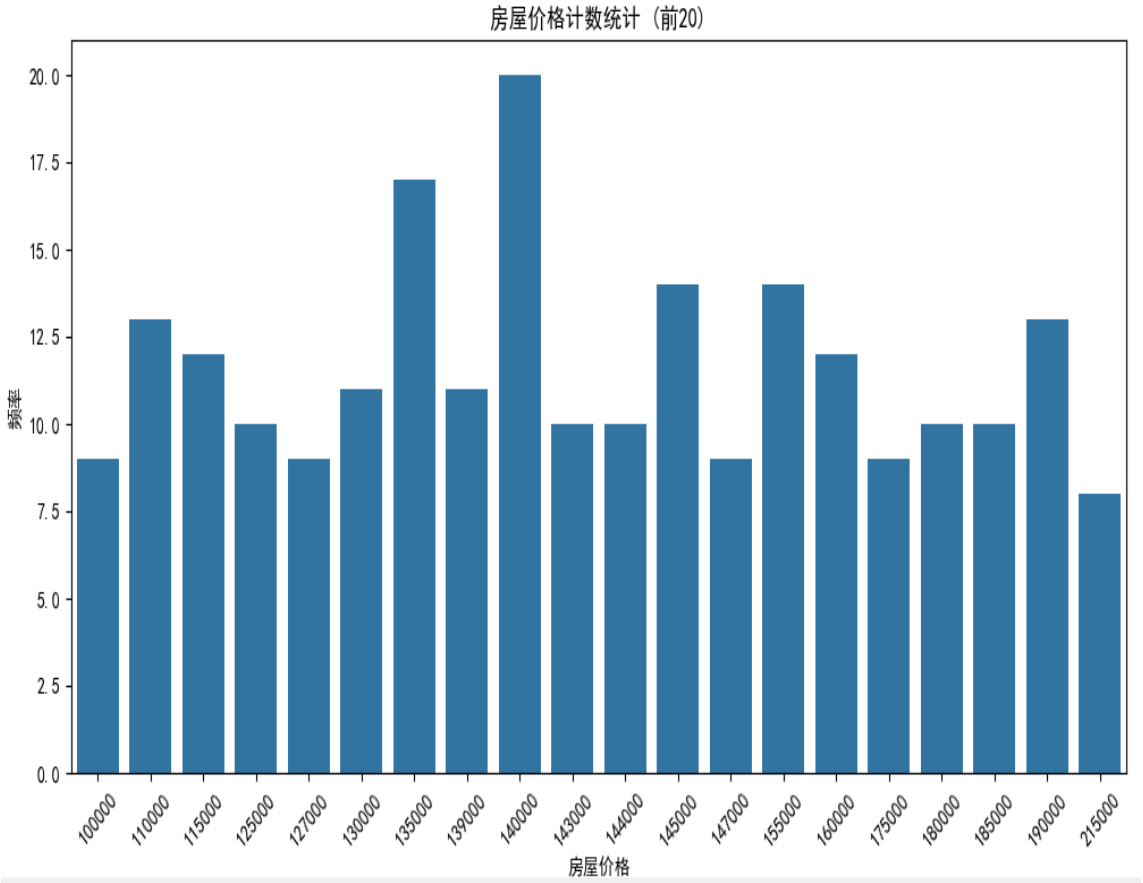


图 9 房价 SalePrice 与土地面积关系图

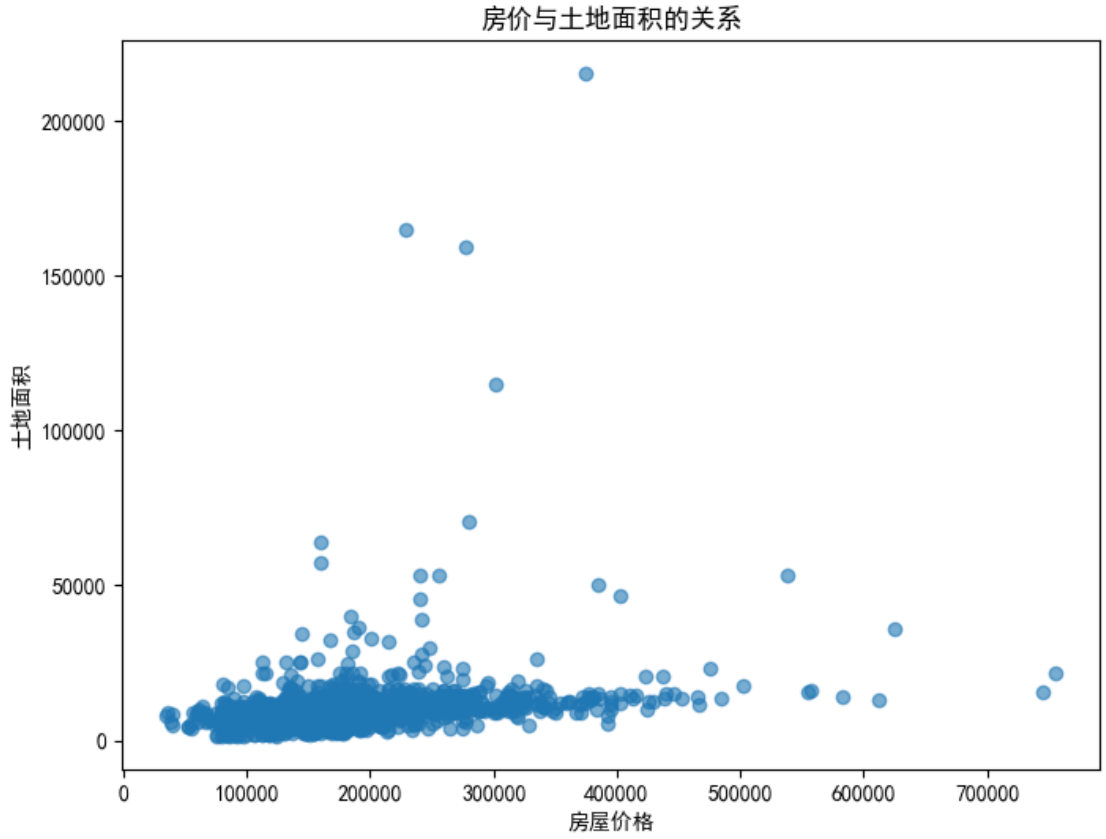


图 10 房价 SalePrice 与 CentralAir 关系图

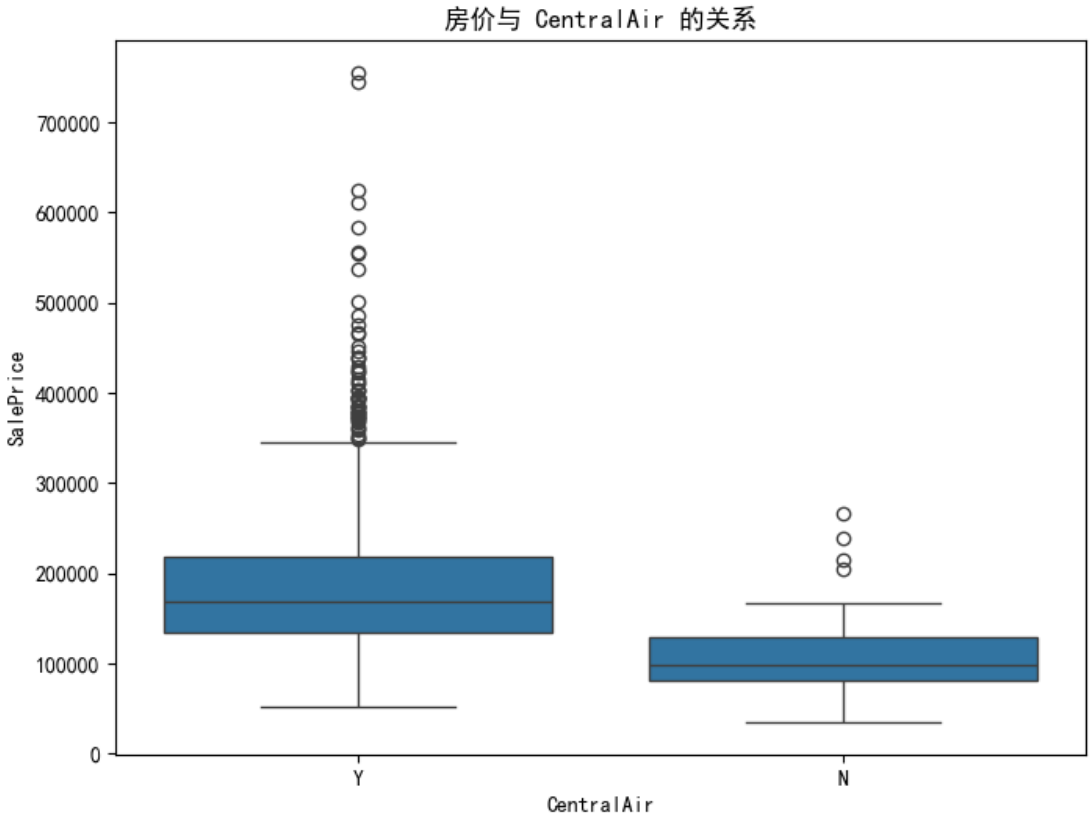
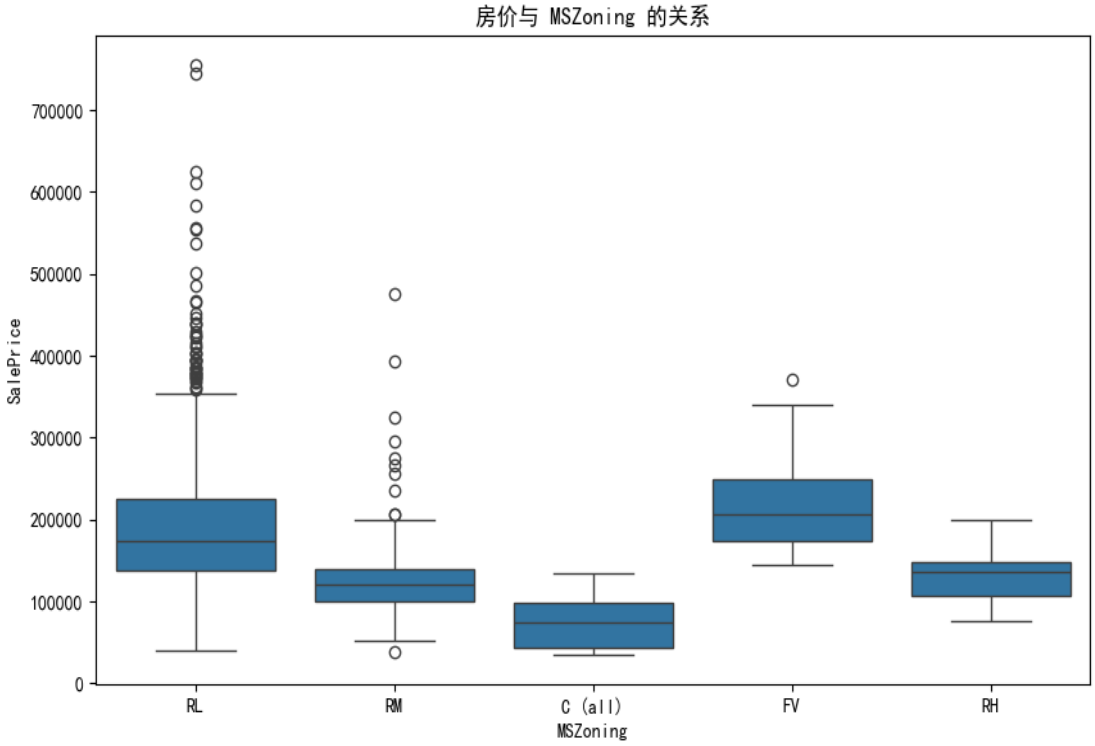


图 11 房价 SalePrice 与 MSZoning 关系图



3.2.2 缺失值处理

通过数据检查发现，训练集和测试集中的部分特征存在缺失值。针对不同类型的特征，采用不同的缺失值处理方法：

缺失值统计,统计每个特征的缺失值数量及缺失比例。对缺失值比例超过50%的特征直接删除，保留数据质量较高的特征。

图 12 部分特征缺失比例表

特征名	缺失值比例 (%)
PoolQC	99.52
MiscFeature	96.3
Alley	93.77
Fence	80.75
MasVnrType	59.73
FireplaceQu	47.26
LotFrontage	17.74
GarageType	5.55
GarageYrBlt	5.55
GarageFinish	5.55

GarageQual	5.55
GarageCond	5.55
BsmtExposure	2.6
BsmtFinType2	2.6
BsmtQual	2.53
BsmtCond	2.53
BsmtFinType1	2.53
MasVnrArea	0.55
Electrical	0.07

数值型特征的缺失值填充：对于连续变量的缺失值，用中位数填充（如LotFrontage），以减少异常值的影响。分类型特征的缺失值填充：对分类型变量的缺失值统一填充为None，表示该属性在样本中不存在，例如Alley表示的巷道类型。

图 13 特征分类处理表

删除的列及其缺失比例		**类别型变量**		**数值型变量**	
列名	**缺失比例(%)**	MSZoning	Heating	LowQualFinSF	Id
Alley	> 50	Street	HeatingQC	GrLivArea	MSSubClass
MasVnrType	> 50	LotShape	CentralAir	BsmtFullBath	LotFrontage
PoolQC	> 50	LandContour	Electrical	BsmtHalfBath	LotArea
Fence	> 50	Utilities	KitchenQual	FullBath	OverallQual
MiscFeature	> 50	LotConfig	Functional	HalfBath	OverallCond
		LandSlope	FireplaceQu	BedroomAbvGr	YearBuilt
		Neighborhood	GarageType	KitchenAbvGr	YearRemodAdd
		Condition1	GarageFinish	TotRmsAbvGrd	MasVnrArea
		Condition2	GarageQual	Fireplaces	BsmtFinSF1

		BldgType	GarageCond	GarageYrBlt	BsmtFinSF2
		HouseStyle	PavedDrive	GarageCars	BsmtUnfSF
		RoofStyle	SaleType	GarageArea	TotalBsmtSF
		RoofMatl	SaleCondition	WoodDeckSF	1stFlrSF
		Exterior1st	ExterCond	OpenPorchSF	2ndFlrSF
		Exterior2nd	Foundation	EnclosedPorch	PoolArea
		ExterQual	BsmtQual	3SsnPorch	MiscVal
		BsmtFinType1	BsmtCond	ScreenPorch	MoSold
		BsmtFinType2	BsmtExposure		YrSold

缺失值填充后的检查：检查填充后的数据集中是否仍存在缺失值，以确保数据完整性。由下表可以看出，进行缺失值填充后数据无缺失了。

图 14 特征缺失值检查表

列名	缺失值数量
Id	0
MSSubClass	0
MSZoning	0
LotFrontage	0
LotArea	0

3.2.3 非数值型数据转换

模型无法直接处理非数值型特征，因此需要将分类型变量转化为数值形式。采用的方法如下：One-Hot 编码，对所有分类型变量（如 MSZoning、CentralAir）进行 One-Hot 编码，将每个类别映射为独立的二进制特征。例如，MSZoning 中的类别值 RL、RM 会被分别转化为 MSZoning_RL 和 MSZoning_RM。

特征对齐：确保训练集和测试集的特征列完全一致。对于测试集中缺失的特征列，填充值为 0，避免预测过程中因列不一致导致的错误。

编码后检查训练集和测试集的特征数量是否一致。确保目标变量 SalePrice 只在训练集中存在，避免对测试集的预测造成干扰。

完成以上步骤，学生完成了数据的清洗与标准化，为模型训练和预测奠定了数据基础。

图 14 数值类型转换后对齐检查结果表

数据集类型	列名	列数
训练集	Id, MSSubClass, LotFrontage, LotArea, OverallQual, OverallCond, YearBuilt, YearRemodAdd, MasVnrArea, ...	246
测试集	Id, MSSubClass, LotFrontage, LotArea, OverallQual, OverallCond, YearBuilt, YearRemodAdd, MasVnrArea, ...	246

3.2.4 特征分离与数据划分

在进行数据建模之前，我们需要将数据分离为特征变量和目标变量，以便训练模型并对房价进行预测。

特征变量（X）： 特征变量包含数据集中除目标变量 SalePrice 之外的所有变量。这些变量将被用于预测目标变量。

目标变量（y）： 目标变量为数据集中的 SalePrice，表示房价。

从训练数据集中移除目标变量 SalePrice，剩余的数据作为特征变量 X 将目标变量 SalePrice 单独提取出来，作为目标变量 y。

将训练数据进一步划分为训练集和验证集，训练集（Train Set）： 用于训练机器学习模型，占总训练数据的 80%。验证集（Validation Set）： 用于评估模型性能，占总训练数据的 20%。训练集用于让模型学习特征与目标变量之间的关系。验证集用于检测模型的泛化能力，避免过拟合或欠拟合。

划分数据时需要随机打乱数据，以确保数据的分布均匀且具有代表性。我们可以使用 train_test_split 函数进行数据划分。

图 15 特征分离与数据划分表

数据集	样本数量	特征数量
训练集（X_train）	1168	245
验证集（X_val）	292	245
训练集（y_train）	1168	1
验证集（y_val）	292	1

4 模型构建与实验结果

4.1 模型构建

4.1.1 随机森林（Random Forest）

随机森林是一种基于决策树的集成学习方法。通过构建多个决策树，并在回归任务中对各树的预测结果进行平均来提升模型的预测性能。调用 Sklearn 中 ensemble 模块的 RandomForestRegressor 类构建随机森林模型对象。设置随机森林中决策树的数量，决策树的数量越大占用的相应内存越多。同时也将延长训练和预测的时间，因此在设置树的数量时也要综合考虑设备内存和性能。导入事先划分好的训练集进行模型训练，输入数据进行预测得到预测结果。调用 Sklearn 中的 cross__val__score 模块计算评估指标的值。

4.1.2 梯度提升决策树（GBDT）

梯度提升决策树（GBDT）是一种基于前向分布的集成学习算法。GBDT 通过构建多个决策树（弱学习器），并逐步拟合前一步的残差，从而优化目标函数。每一棵树的训练结果都会对整个模型的性能起到提升作用，因此模型能够有效地捕捉数据中的非线性关系。GBDT 在回归任务中通过逐步减少预测误差，具有较高的预测精度。，调用 Scikit-learn 库中 ensemble 模块的 GradientBoostingRegressor 类来构建 GBDT 模型对象。在模型构建过程中，需要设置多个关键超参数，包括决策树的数量（n_estimators）、学习率（learning_rate）、决策树的最大深度（max_depth）、子采样率（subsample）等。

完成模型训练后，输入验证集进行预测，评估模型性能。此外，调用 Scikit-learn 中的 cross_val_score 模块对模型进行交叉验证，进一步计算和评估模型的性能指标，包括决定系数（R^2）和均方根误差（RMSE）。模型的预测性能将在验证集上进行全面比较。

4.2 超参数调优

随机森林是一种基于决策树的集成学习方法，其模型性能受到多个关键超参数的影响。梯度提升决策树（GBDT）的模型性能高度依赖于超参数的设置，因此在模型训练之前对关键超参数进行调优也至关重要。

为了优化模型性能，本实验使用 GridSearchCV 网格搜索方法对模型的超参数进行调优。具体调优参数如下。

随机森林的调优参数

n_estimators（决策树的数量）：设置为 100、200、300、400、500。

max_features（每次分裂使用的最大特征数）：设置为 'sqrt'、'log2'、None。

GBDT 的调优参数

n_estimators（决策树的数量）：设置为 100、200、300、400。

learning_rate（学习率）：设置为 0.05、0.1、0.2。

max_depth（每棵树的最大深度）：设置为 3、4、5。

subsample（子采样率）：设置为 0.8、1.0。

图 16 超参数调优最佳参数表

模型类型	最佳参数	训练集 R^2(拟合率)	训练集 RMSE（均方根误差）	验证集 R^2（准确率）	验证集 RMSE（均方根误差）
随机森林 (RF)	{'max_features': None, 'n_estimators': 400}	0.98	11176.94	0.89	28779.80
梯度提升树 (GBDT)	{'learning_rate': 0.05, 'max_depth': 4, 'n_estimators': 400, 'subsample': 0.8}	N/A	N/A	0.91	26332.82

4.3 模型训练与评估

决定系数（R^2）： 衡量模型对目标变量的解释能力。值越接近 1，模型性能越好。

均方根误差（RMSE）： 衡量模型预测值与真实值之间的差异，值越小越好。

图 17 模型预测效果对比表

模型类型	最佳参数	训练集 R^2(拟合率)	训练集 RMSE（均方根误差）	验证集 R^2（准确率）	验证集 RMSE（均方根误差）
随机森林 (RF)	{'max_features': None, 'n_estimators': 400}	0.98	11176.94	0.89	28779.80
梯度提升树 (GBDT)	{'learning_rate': 0.05, 'max_depth': 4, 'n_estimators': 400, 'subsample': 0.8}	N/A	N/A	0.91	26332.82

从结果来看，GBDT 在验证集上的性能优于随机森林，在这里，GBDT 的 R^2 为 0.91，高于随机森林的 0.89，说明 GBDT 对验证集的拟合更好。GBDT 的 RMSE 为 26332.82，小于随机森林的 28779.80，说明 GBDT 预测的误差更小。

从验证集的 R^2 和 RMSE 两个指标来看，GBDT 在性能上略胜一筹。GBDT 通过调节学习率（learning_rate）和树的深度（max_depth）可以进一步控制模型的复杂度，从而在验证集上表现出更好的泛化能力。随机森林的优势在于简单易

用和对超参数不太敏感，但在这个任务中，其性能略逊于 GBDT。如果以验证集的性能作为标准，GBDT 更适合作为最终的房价预测模型。

4.4 测试集预测

在完成模型训练和验证后，将模型应用于测试集，对房价进行预测。分别保存随机森林和 GBDT 的预测结果到以下文件：

随机森林预测结果保存为 Predictions.csv。

GBDT 预测结果保存为 Predictions_GBDT.csv。

图 18 随机森林预测结果部分表 Predictions

Id	SalePrice	Id	SalePrice	Id	SalePrice	Id	SalePrice
1461	129388.9	1475	140092.8	1489	214832.6	1504	228730.9
1462	153869.5	1476	368010.8	1490	199740.1	1505	219949.8
1463	178580.1	1477	260939	1491	192675.7	1506	192360.1
1464	186362	1478	296614.4	1492	99250.83	1507	230937.2
1465	204964.7	1479	251414.7	1493	176808.3	1508	205125.8
1466	184753.2	1480	419988.9	1494	289243.7	1509	165204.7
1467	170928.6	1481	306236.3	1495	298760.3	1510	150159.6
1468	175959.4	1482	212140.6	1496	215701.5	1511	149613
1469	181174.9	1483	179686.1	1497	190218.6	1512	167273.1
1470	121085.5	1484	172481.3	1498	152603.2	1513	156251.4
1471	192512.4	1485	170294.4	1499	151847.4	1514	165538
1472	91281.75	1486	198110.3	1500	156820.5	1515	176440.2
1473	104731.1	1487	340454.8	1501	169591.4	1516	164261.1
1474	153983.1	1488	241453.4	1502	163246	1517	149555.6

图 19 GBDT 预测结果部分表 Predictions_GBDT

Id	SalePrice	Id	SalePrice	Id	SalePrice	Id	SalePrice
1461	122502.5	1475	121352	1489	201937.7	1503	300294.3
1462	160358.3	1476	355713.1	1490	194518.1	1504	240017.7
1463	182703	1477	253243.5	1491	189991.5	1505	217095.3
1464	190995	1478	286955.3	1492	87667.6	1506	189020.7
1465	201396.5	1479	236253.7	1493	190977.6	1507	240176.3
1466	175243.8	1480	466437.4	1494	277757.9	1508	196185.7
1467	182971	1481	313145	1495	291807.2	1509	163223.5
1468	165497.1	1482	205188	1496	223074.4	1510	147761.8
1469	174311.9	1483	157103.2	1497	181798.1	1511	142484.6
1470	125457	1484	153796.8	1498	155301.4	1512	165403.5
1471	199286.5	1485	168969.8	1499	153606.2	1513	155664.6
1472	91344.26	1486	190403.2	1500	136650.9	1514	171068.3
1473	95853.29	1487	378277.7	1501	173552.9	1515	189656.1
1474	153152.5	1488	231688.5	1502	147742.5	1516	161675.8

结论： GBDT 模型在验证集上的性能优于随机森林模型。GBDT 的预测结果较随机森林更具鲁棒性，适合更复杂的应用场景。

4.5 模型优劣分析

随机森林模型在高维数据处理中表现优异，通过随机采样特征和样本的方式，有效降低了过拟合风险，并具有较强的抗噪声能力和计算效率。此外，随机森林提供特征重要性输出，增强了模型的解释性。然而，随机森林在捕捉强非线性关系时表现有限，对数据分布偏斜敏感，且需要较大的存储空间和计算成本。

GBDT 模型擅长捕捉非线性特征关系，通过逐步优化残差，能更精确地拟合目标值，并在验证集上表现优于随机森林。它对连续和离散型特征兼容性强，且参数调控灵活。然而，GBDT 在高维数据中效率较低，训练速度较慢，并且对超参数和异常值较为敏感，参数调整不当可能导致模型性能下降。

图 20 随机森林模型和 GBDT 模型对比

特性	随机森林 (Random Forest)	梯度提升树 (GBDT)
非线性关系建模	一般，非线性捕捉能力有限	较强，能够捕捉复杂的非线性关系
高维数据处理能力	较强，能够有效处理高维特征数据	一般，高维特征数据可能降低训练效率
训练速度	快，并行处理训练效率较高	慢，串行处理导致训练时间较长
异常值鲁棒性	较强，通过投票或平均机制减少异常值影响	较弱，容易受到异常值的干扰
模型复杂度控制	参数较少，调优较简单	参数较多，调优较复杂
泛化能力	较好，尤其在数据样本量大时表现稳定	较好，但需要精细调参以平衡过拟合和欠拟合
存储需求	较高，需要存储所有决策树	较低，只需存储加权后的决策树

从实验结果来看，随机森林的验证集 R^2 为 0.89，RMSE 为 28779.80，适合处理高维特征数据，对异常值鲁棒性较强。GBDT 的验证集 R^2 为 0.91，RMSE 为 26332.82，更适合捕捉非线性关系且能生成更精确的预测结果。

因此，在本实验中，GBDT 更适合当前房价预测任务，但如果数据维度增加或样本量大幅提升，可以考虑使用随机森林以获得更快的训练速度和较好的泛化能力。

5 结论与展望

5.1 结论

过实验可以看出，随机森林和 GBDT 两种模型在房价预测任务中均表现出了较好的性能。随机森林模型的拟合能力较强，能够很好地处理高维数据和噪声问题，在训练集上的拟合效果优异。然而，在验证集上的预测性能稍逊于 GBDT，验证集的 R^2 为 0.89，RMSE 为 28779.80。而 GBDT 模型在验证集上的表现更为优越，其 R^2 达到了 0.91，RMSE 为 26332.82，说明 GBDT 对非线性特征关系的捕捉能力更强，更适合此类房价预测任务。总体来说，若对高维数据处理和训练效率有较高要求，可选择随机森林；若更注重模型精度和对非线性关系的刻画，则 GBDT 更为适用。

5.2 展望

尽管本研究中的模型在房价预测任务中取得了不错的结果，但仍有改进空间。首先，可以进一步优化特征工程，如引入更多与房价相关的特征，或对现有特征进行交互处理。此外，可以尝试使用模型融合（Ensemble）的方法，将随机森林与 GBDT 的预测结果结合，从而进一步提高模型的泛化能力和预测精度。此外，若数据集中包含时间序列信息（如房屋建造年份与地区经济变化），可考虑引入时间序列建模方法以提升预测性能。最后，未来可以将模型应用于更大规模的房价数据集或其他预测任务中，例如商业地产价格预测或租金预测，从而验证模型的适用性和可扩展性。

参考文献

[1]周亮锦，赵明扬.基于随机森林的深圳二手房价格分析[J].中国市场，2022(26)：68-71+133.
[2]徐颖，黄素珍.大数据时代房地产价格预测模型的研究[J].中国商论，2017(4)：134-137.
[3]李贤增.基于多元回归分析和支持向量机的房价预测[D].北京：清华大学，2019.
[4]罗博炜，洪智勇，王劲屹.多元线性回归统计模型在房价预测中的应用[J].计算机时代，2020(6)：51-54.
[5]李然，章政，缪华昌.基于神经网络模型的房价预测研究[J].中国集体经济，2022(27)：94-96.
[6]路佳佳.基于交叉验证的集成学习误差分析[J].计算机系统应用，2023，32(1)：302-309.
[7]ZOU H Y.Machine Learning-Based Analysis of Housing Price Predictors[J/OL].Academic Journal of Business & Management，2023，5(2)[2024-03-18].https://francis-press.com/papers/9179.
[8]宋阳.基于梯度提升决策树的房价预测模型[J].现代计算机,2024,30(17):81-84.
[9]崔慧莹.基于极端随机森林算法的改进与二手房价预测的研究[D].大连交通大学,2023.DOI:10.26990/d.cnki.gsltc.2023.000640.
[10]王阳春，茆梦凡，龙关旭，等.基于 GBDT 的锈蚀钢筋混凝土梁抗弯承载力预测研究[J].公路，2023,68(12):309-314.

附录

数据集获取网址：House Prices - Advanced Regression Techniques | Kaggle

Python 实验代码：

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib
from matplotlib import rcParams
from sklearn.impute import SimpleImputer
from sklearn.ensemble import RandomForestRegressor,
GradientBoostingRegressor
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import mean_squared_error, r2_score

matplotlib.use('TkAgg')  # 设置后端为 TkAgg
# 设置字体
rcParams['font.sans-serif'] = ['SimHei']  # 使用黑体
rcParams['axes.unicode_minus'] = False  # 解决负号显示问题

# 读取数据
data_train = pd.read_csv('train.csv')  # 确保路径正确，去除多余空格
data_test = pd.read_csv('test.csv')

# 查看数据基本信息
print("训练集头部数据:", data_train.head())
print("测试集头部数据:", data_test.head())
print("训练集列名:", data_train.columns)
print("训练集每列的数据格式:")
print(data_train.info())

# 数据类型
data_train_dtypes = data_train.dtypes
print("训练集数据结构为:", data_train_dtypes)

# 统计房价的基本信息
print("\n 房价描述统计信息:")
print(data_train['SalePrice'].describe())  # 输出描述性统计信息
```

```
# 检查是否有缺失值
print("\n 房价列中缺失值数量:")
print(data_train['SalePrice'].isnull().sum())  # 检查缺失值数量

# 查看因变量价格的分布情况
plt.figure(figsize=(10, 6))
sns.histplot(data_train['SalePrice'], kde=True, bins=30)  # distplot
plt.title("房屋价格分布图")
plt.xlabel("房屋价格")
plt.ylabel("频率")
plt.show()

# 按价格统计频率并绘制分布图
price_counts = data_train['SalePrice'].value_counts().reset_index()
price_counts.columns = ['SalePrice', 'Count']

plt.figure(figsize=(12, 6))
sns.barplot(x='SalePrice', y='Count', data=price_counts.head(20))  #
显示频率前 20 的价格
plt.xticks(rotation=45)
plt.title("房屋价格计数统计 (前 20)")
plt.xlabel("房屋价格")
plt.ylabel("频率")
plt.show()

# 缺失值处理逻辑
# 统计缺失值
missing_values = data_train.isnull().sum()
missing_ratio = (missing_values / len(data_train)) * 100
print("\n 缺失值统计:")
print(missing_ratio[missing_ratio > 0].sort_values(ascending=False))

# 删除缺失值比例超过 50%的列
drop_columns = missing_ratio[missing_ratio > 50].index
print("\n 以下列缺失比例超过 50%，将被删除：", drop_columns)
data_train = data_train.drop(columns=drop_columns)
data_test = data_test.drop(columns=drop_columns)

# 分类变量和数值变量分离
class_variable = [col for col in data_train.columns if
data_train[col].dtypes == 'O']
numerical_variable = [col for col in data_train.columns if
data_train[col].dtypes != 'O' and col != 'SalePrice']
```

```
print("\n类别型变量:", class_variable)
print("数值型变量:", numerical_variable)

# 填充缺失值
# 数值型变量：用中位数填充
imputer = SimpleImputer(strategy='median')
data_train[numerical_variable] =
imputer.fit_transform(data_train[numerical_variable])
data_test[numerical_variable] =
imputer.transform(data_test[numerical_variable])

# 类别型变量：填充为 "None"
data_train[class_variable] =
data_train[class_variable].fillna('None')
data_test[class_variable] = data_test[class_variable].fillna('None')

# 检查填充后的缺失值情况
print("\n处理后的数据列中缺失值数量：")
print(data_train.isnull().sum().sort_values(ascending=False).head())

# 可视化：房价与某些特征的关系
# CentralAir
plt.figure(figsize=(8, 6))
sns.boxplot(x='CentralAir', y='SalePrice', data=data_train)
plt.title("房价与 CentralAir 的关系")
plt.show()

# MSSubClass
plt.figure(figsize=(10, 6))
sns.boxplot(x='MSSubClass', y='SalePrice', data=data_train)
plt.title("房价与 MSSubClass 的关系")
plt.show()

# MSZoning
plt.figure(figsize=(10, 6))
sns.boxplot(x='MSZoning', y='SalePrice', data=data_train)
plt.title("房价与 MSZoning 的关系")
plt.show()

# SalePrice vs LotArea (散点图)
plt.figure(figsize=(8, 6))
plt.scatter(data_train['SalePrice'], data_train['LotArea'],
alpha=0.6)
plt.xlabel('房屋价格')
```

```
plt.ylabel('土地面积')
plt.title('房价与土地面积的关系')
plt.show()

# 特征相关性分析
# 仅选择数值型列用于计算相关性
numerical_data_train = data_train.select_dtypes(include=['float64',
'int64'])

# 计算特征与房价的相关性
correlation_matrix = numerical_data_train.corr()

# 选择与目标变量 SalePrice 相关性最高的特征
correlation_with_target =
correlation_matrix['SalePrice'].sort_values(ascending=False)
print("与房价相关性最高的特征：\n", correlation_with_target)

# 可视化相关性矩阵
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, cmap='coolwarm', annot=False)
plt.title("特征相关性热力图")
plt.show()

# 绘制与房价最相关的前 10 个特征
top_features = correlation_with_target.head(11).index
plt.figure(figsize=(10, 6))
sns.heatmap(numerical_data_train[top_features].corr(), annot=True,
cmap='coolwarm', fmt='.2f')
plt.title("与房价最相关的特征热力图")
plt.show()

# 数据可视化之后再进一步处理数据
# 对类别型变量进行 One-Hot 编码
data_train = pd.get_dummies(data_train, columns=class_variable,
drop_first=True)
data_test = pd.get_dummies(data_test, columns=class_variable,
drop_first=True)

# 确保训练集和测试集的列一致
data_train, data_test = data_train.align(data_test, join='left',
axis=1)
data_test = data_test.fillna(0)  # 测试集可能有部分列缺失，用 0 填充
print("训练集列名：", data_train.columns)
print("测试集列名：", data_test.columns)
```

```
print("列名是否一致: ", set(data_train.columns) ==
set(data_test.columns))
print("测试集中是否包含 'SalePrice': ", 'SalePrice' in
data_test.columns)

# 构造随机森林模型
X = data_train.drop(['SalePrice'], axis=1)  # 特征
y = data_train['SalePrice']  # 目标变量

# 数据集划分（训练集 S 和 验证集 V）
X_train, X_val, y_train, y_val = train_test_split(X, y,
test_size=0.2, random_state=42)

# 超参数调优
param_grid = {
    'n_estimators': [100, 200, 300, 400, 500],
    'max_features': ['sqrt', 'log2', None]
}

grid_search = GridSearchCV(
    estimator=RandomForestRegressor(random_state=42),
    param_grid=param_grid,
    scoring='r2',
    cv=5,
    n_jobs=-1
)

grid_search.fit(X_train, y_train)

print("最佳参数:", grid_search.best_params_)

# 使用最佳参数训练模型
rf_model = RandomForestRegressor(
    n_estimators=grid_search.best_params_['n_estimators'],
    max_features=grid_search.best_params_['max_features'],
    random_state=42
)
rf_model.fit(X_train, y_train)

# 模型评估：训练集
rf_preds_train = rf_model.predict(X_train)
rf_r2_train = r2_score(y_train, rf_preds_train)
rf_rmse_train = np.sqrt(mean_squared_error(y_train, rf_preds_train))
```

```
# 模型评估：验证集
rf_preds_val = rf_model.predict(X_val)
rf_r2_val = r2_score(y_val, rf_preds_val)
rf_rmse_val = np.sqrt(mean_squared_error(y_val, rf_preds_val))

# 打印结果
print("\n随机森林模型评估结果：")
print(f"训练集 R²（拟合率）：{rf_r2_train:.2f}")
print(f"训练集 RMSE（均方根误差）：{rf_rmse_train:.2f}")
print(f"验证集 R²（准确率）：{rf_r2_val:.2f}")
print(f"验证集 RMSE（均方根误差）：{rf_rmse_val:.2f}")

# 测试集预测
# 确保测试集和训练集的列一致（处理预测时的列名问题）
data_test = data_test.drop(columns=['SalePrice'], errors='ignore')  # 确保测试集中没有 'SalePrice'
data_test = data_test.reindex(columns=X.columns, fill_value=0)  # 调整列顺序，与训练集一致

# 对测试集进行预测
rf_preds_test = rf_model.predict(data_test)  # 预测房价

# 创建预测结果 DataFrame
test_ids = data_test['Id']  # 提取测试集的 'Id' 列
submission = pd.DataFrame({'Id': test_ids, 'SalePrice':
rf_preds_test})

# 保存预测结果到 Predictions.csv
submission.to_csv('Predictions.csv', index=False)
print("\n测试集预测结果已保存为 Predictions.csv 文件中")

# GBDT 模型
param_grid_gbdt = {
    'n_estimators': [100, 200, 300, 400],
    'learning_rate': [0.05, 0.1, 0.2],
    'max_depth': [3, 4, 5],
    'subsample': [0.8, 1.0]
}

grid_search_gbdt = GridSearchCV(
    estimator=GradientBoostingRegressor(random_state=42),
    param_grid=param_grid_gbdt,
    scoring='r2',
    cv=5,
```

```
    n_jobs=-1
)

grid_search_gbdt.fit(X_train, y_train)
print("GBDT 最佳参数:", grid_search_gbdt.best_params_)

# GBDT 模型训练
gbdt_model = GradientBoostingRegressor(
    n_estimators=grid_search_gbdt.best_params_['n_estimators'],
    learning_rate=grid_search_gbdt.best_params_['learning_rate'],
    max_depth=grid_search_gbdt.best_params_['max_depth'],
    subsample=grid_search_gbdt.best_params_['subsample'],
    random_state=42
)
gbdt_model.fit(X_train, y_train)

# GBDT 模型评估
gbdt_preds_val = gbdt_model.predict(X_val)
gbdt_r2_val = r2_score(y_val, gbdt_preds_val)
gbdt_rmse_val = np.sqrt(mean_squared_error(y_val, gbdt_preds_val))
print("\nGBDT 模型验证集性能：")
print(f"R²: {gbdt_r2_val:.2f}, RMSE: {gbdt_rmse_val:.2f}")

# 测试集预测
data_test_gbdt = data_test.drop(columns=['SalePrice'],
errors='ignore')
gbdt_preds_test = gbdt_model.predict(data_test_gbdt)
submission_gbdt = pd.DataFrame({'Id': data_test['Id'], 'SalePrice':
gbdt_preds_test})
submission_gbdt.to_csv('Predictions_GBDT.csv', index=False)
print("GBDT 测试集预测结果已保存为 Predictions_GBDT.csv")

# 随机森林和 GBDT 性能对比
print("\n 模型性能对比：")
print(f"随机森林验证集 R²: {rf_r2_val:.2f}, RMSE: {rf_rmse_val:.2f}")
print(f"GBDT 验证集 R²: {gbdt_r2_val:.2f}, RMSE: {gbdt_rmse_val:.2f}")
```