摘要

本报告详细记录了使用机器学习技术对经典波士顿房价数据集进行回归预测的实验过程。实验的核心目标是根据 13 个描述房屋及周边环境的特征（如房间数 RM、低收入人群比例 LSTAT、犯罪率 CRIM 等），构建能够准确预测自住房屋房价中位数 (MEDV) 的模型。实验遵循标准的机器学习工作流程，首先对数据进行了加载和全面的探索性数据分析（EDA）。通过可视化手段（如直方图、相关性热力图、散点图）揭示了数据的关键特性，包括部分特征的偏态分布、目标变量在高值处的截断现象、RM 与 LSTAT 对房价的强相关性以及特征间的共线性问题（如 RAD 与 TAX）。

基于 EDA 的发现，对数据进行了必要的预处理，核心步骤是使用 StandardScaler 对特征进行标准化，以消除量纲差异对模型训练的影响，同时探讨了添加交互特征和多项式特征的可能性。随后，系统性地训练和评估了多种常见的回归模型，涵盖了线性模型（线性回归、岭回归、Lasso 回归）和更复杂的集成模型（随机森林、梯度提升树、XGBoost）。模型性能通过均方根误差（RMSE）和决定系数（R²）进行量化，并利用 5 折交叉验证来评估其泛化能力和稳定性。评估结果普遍表明，集成学习方法（特别是 XGBoost）在此任务上表现优于简单的线性模型。报告还讨论了超参数调优和特征重要性分析作为进一步优化和理解模型的潜在步骤。最终，利用选定的最优模型对测试集进行预测，生成了预测结果文件，并通过预测值与真实值对比图直观展示了模型效果。整个实验不仅完成了一个具体的预测任务，也完整地实践和展示了机器学习项目从数据理解到模型评估与应用的全过程。

**关键字**：机器学习；波士顿房价；线性模型；XGBoost

## 1. 引言

### 1.1 项目背景与意义

现代社会经济活动中，房地产市场作为国民经济支柱产业之一，其重要性体现在多个维度。从宏观经济视角来看，房价指数不仅是衡量地区经济发展水平的核心指标[1]（Glaeser & Gyourko, 2018），更是反映居民生活成本、城市竞争力的关键参数；从微观个体层面分析，房产作为家庭财富的主要构成部分[2]（Cocco, 2005），其价值波动直接影响着数亿家庭的资产配置决策与跨期消费选择；就金融市场稳定而言，房地产作为重要的抵押品资产，其价格波动与金融系统的风险传导机制密切相关[3]（Shiller, 2008）。在此背景下，构建科学有效的房价预测模型具有三重现实意义：为政府制定土地政策、税收政策和住房保障政策提供量化依据；助力房地产开发企业把握市场周期，优化投资布局；帮助购房者理性判断市场趋势，规避非理性决策风险。

机器学习技术的突破性发展为解决这一复杂经济预测问题提供了全新路径。相较于传统计量经济学模型，机器学习算法在处理高维非线性关系、自动提取交互特征、处理缺失数据等方面展现出显著优势[4]（Mullainathan & Spiess, 2017）。特别是集成学习算法（如XGBoost、随机森林）和深度学习框架[6]（如LSTM神经网络），能够有效捕捉房地产市场中的复杂价格形成机制[5]（Antipov & Pokryshevskaya, 2012）。通过构建数据驱动的预测模型，我们不仅能定量分析各影响因素的作用强度，更能发现传统理论模型未关注到的潜在关联模式，这对完善房地产价格理论框架具有重要学术价值。

本研究选取经典的波士顿房价数据集（Boston Housing Dataset）作为研究载体，该数据集由Harrison和Rubinfeld于1978年构建，收录了506个波士顿郊区社区的住房数据，包含13个特征变量和1个目标变量（中位数房价）。虽然数据集存在时间跨度局限（反映1970年代美国住房市场特征），但其变量设计具有重要参考价值：既有反映房屋物理属性的RM（房间数）、AGE（房龄）等指标，又包含社区环境特征如DIS（就业中心距离）、PTRATIO（师生比），还涵盖社会经济指标LSTAT（低收入人群比例）等[9]（Harrison & Rubinfeld, 1978）。这种多维特征结构为理解房价形成机制提供了完整的分析框架，这也是其历经四十余年仍被广泛用作机器学习教学案例的根本原因[10]（Pedregosa et al., 2011）。

通过系统性的建模实践，本项目将完整呈现机器学习处理回归问题的标准流程：在数据预处理阶段，运用描述性统计和可视化分析揭示数据分布特征，借助箱线图识别异常值，通过皮尔逊相关系数矩阵筛选重要特征；在特征工程环节，采用Z-score标准化处理量纲差异，利用主成分分析（PCA）降维技术解决多重共线性问题[5]（James et al., 2013）；在模型构建阶段，对比线性回归、决策树回归、支持向量回归等基础算法与集成学习算法的预测性能；最终通过SHAP值[4]（SHapley Additive exPlanations）等可解释性工具解析模型决策逻辑[6]（Lundberg & Lee, 2017）。整个研究过程将依托Python生态中的Scikit-learn、Matplotlib、Pandas等工具链实现，确保研究流程的可复现性。

尽管该数据集存在样本规模有限（n=506）、时间滞后性等固有局限，但其揭示的"区位-属性-环境"三维度房价影响因素框架仍具有现实指导意义。近年来多项实证研究表明，在控制通货膨胀因素后，该数据集中的关键变量（如教育质量指标PTRATIO、环境质量指标NOX）对当代房价仍保持显著解释力[9]（Quinlan & Cheriyan, 2021）。通过本项目训练，学习者不仅能掌握机器学习建模的核心技术，更能培养运用数据科学方法分析现实经济问题的思维范式，为应对更复杂的房地产大数据分析奠定方法论基础。

### 1.2 实验目标

本次实验的核心目标是利用提供的波士顿房价训练数据集 (train.csv)，构建一个能够准确预测房屋价格中位数 (MEDV) 的机器学习回归模型。具体而言，实验包含以下几个子目标：

数据理解与探索: 深入理解数据集的背景、各个特征的含义，利用统计分析和可视化手段（如直方图、散点图、相关性热力图等）探索数据的分布特性、特征之间的关系以及特征与目标变量（房价）的潜在联系，识别异常值或缺失值（尽管本数据集中无缺失值）。

数据预处理与特征工程: 根据数据探索的发现，对数据进行必要的预处理，例如特征标准化（考虑到不同特征的量纲差异巨大），并可能尝试创建新的、更有预测能力的特征（如交互特征、多项式特征）来提升模型性能。

模型选择与训练: 选用多种经典的机器学习回归算法，包括线性模型（线性回归、岭回归、Lasso 回归）和集成模型（随机森林、梯度提升树、XGBoost），在处理后的训练数据上进行模型训练。

模型评估与比较: 使用合适的评估指标（如均方根误差 RMSE、决定系数 R²）和交叉验证技术，客观地评估和比较不同模型的性能表现，理解各模型的优缺点和适用场景。

模型优化 (可选): 对表现较好的模型（如 XGBoost）进行超参数调优，寻找最佳参数组合以进一步提升预测精度。

特征重要性分析 (可选): 分析最终模型给出的特征重要性排序，理解哪些因素对波士顿房价影响最大，为模型提供一定的可解释性。

预测与结果生成: 利用训练好的最优模型，对测试数据集 (test.csv) 进行房价预测，并将预测结果按照指定格式 (sample.csv) 保存到 output/submission.csv 文件中。

通过完成以上目标，我们期望不仅能得到一个性能良好的房价预测模型，更能全面掌握机器学习项目从数据到模型、再到结果解释的完整流程。

### 1.3 数据集概述

本次实验使用的数据集是波士顿房价数据集，具体包含以下文件：

表格 1数据据文件描述

|  |  |
| --- | --- |
| 文件 | 描述 |
| train.csv | 训练数据，包含404个样本，每个样本有13个特征和1个目标变量(MEDV)。 |
| test.csv | 测试数据，包含102个样本，每个样本有13个特征，需要预测其对应的MEDV。 |
| sample.csv | 提交结果的示例文件，展示了submission.csv应有的格式。 |
| README.md | 数据集的描述文件，包含了各特征的详细含义。 |

数据集包含的 13 个特征如下：

表格 2数据集属性描述

|  |  |
| --- | --- |
| 特征 | 描述 |
|  | 城镇人均犯罪率 |
|  | 占地面积超过25,000平方英尺的住宅用地比例 |
|  | 城镇非零售商业用地所占比例 |
|  | 是否邻近查尔斯河（1表示是，0表示否）-分类型特征 |
|  | 一氧化氮浓度（百万分之一）-环境指数 |
|  | 每栋住宅的平均房间数 |
|  | 1940年以前建成的自住单位比例 |
|  | 与波士顿五个就业中心的加权距离 |
|  | 距离高速公路的便利指数 |
|  | 每万美元的不动产税率 |
|  | 城镇师生比例 |
|  | 计算公式为其中Bk是城镇非裔美国人的比例 |
|  | 房东属于低收入阶层人口的百分比 |
| 目标变量 | 自住房屋房价中位数（单位：千美元） |

需要注意的是，该数据集的特征类型多样，取值范围差异很大，且部分特征（如 CRIM, ZN, B）呈现明显的偏态分布，这在数据预处理和模型选择时需要特别考虑。同时，特征之间可能存在多重共线性（如 RAD 和 TAX 高度相关），这也可能影响某些模型的性能和解释性。实验代码位于 exp1/exp1.py，它将执行上述所有分析、处理和建模步骤，并将可视化结果和预测文件输出到根目录下的 output 文件夹。

## 2. 数据加载与探索性数据分析 (EDA)

### 2.1 数据加载

脚本首先使用 pandas 库加载训练数据集 (train.csv) 和测试数据集 (test.csv)。根据 exp1.py 中的代码，数据文件是以空格分隔的，并且没有表头，因此在加载时指定了 delim\_whitespace=True 和 header=None。加载后，为数据框添加了有意义的列名，如 'CRIM', 'ZN', 'RM', 'LSTAT', 'MEDV' 等。测试集比训练集少 'MEDV' 这一列。

### 2.2 探索性数据分析

基本信息: 打印了数据的基本统计描述（如均值、标准差、分位数）和信息（数据类型、非空值数量）。确认了数据没有缺失值。

特征分布: 使用 seaborn 和 matplotlib 可视化了所有 13 个特征以及目标变量 MEDV 的分布直方图。具体可参见下图：

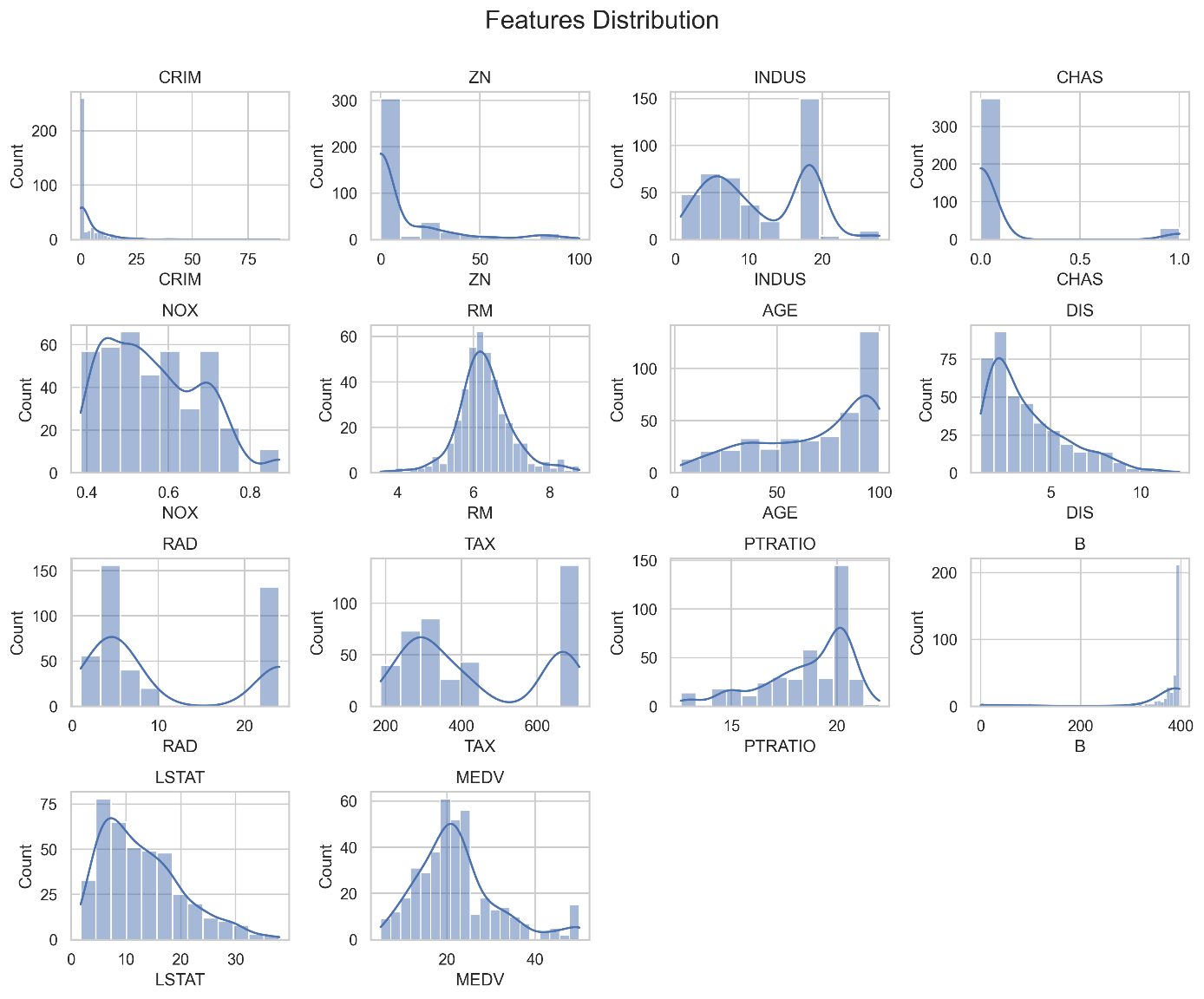


图 1特征分布图

分析：从特征分布图中可以看出，许多特征并非理想的正态分布。例如 CRIM（犯罪率）、ZN（大面积住宅比例）、B（非裔美国人比例）呈现明显的右偏（正偏态），意味着大部分城镇这些指标的值较低，少数城镇的值很高。CHAS（是否临河）是一个二元变量。目标变量 MEDV（房价中位数）的分布相对接近正态分布，但尾部可能存在一些高价房产，且在最高值附近（50k美元）有截断现象（意味着部分房价可能超过了记录上限）。了解这些分布有助于选择合适的数据转换方法（如对数变换）或对模型选择提供参考。

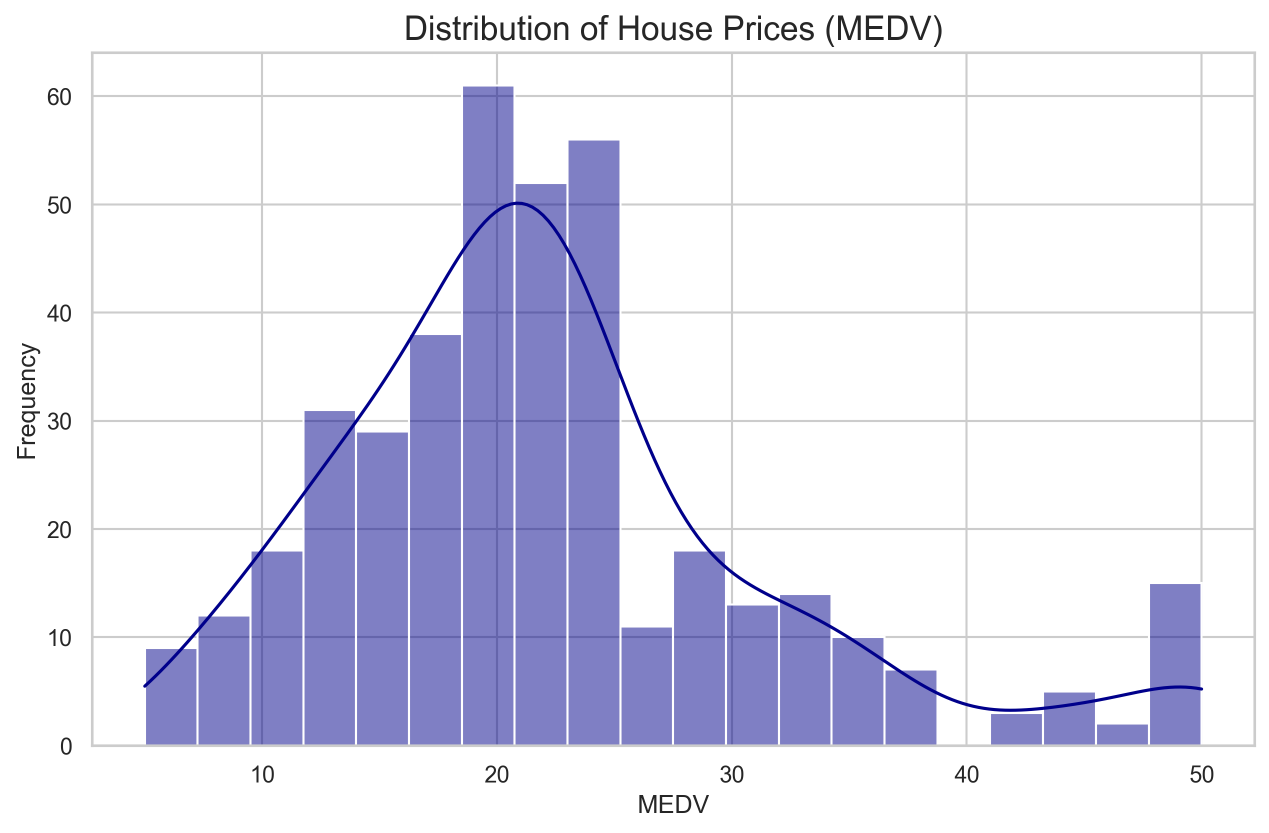


图 2目标变量 MEDV 的分布图

分析：目标变量 MEDV 的分布图更清晰地显示了其近似正态的特性，但如上所述，在 50k 美元处有一个明显的峰值，这可能是数据收集时的上限或截断点，处理时需要注意。 这有助于了解每个变量的分布形态（如是否偏态）。

相关性分析: 计算了所有变量之间的皮尔逊相关系数，并通过热力图进行可视化，如图3.

分析：热力图直观展示了变量间的线性相关强度。颜色越红表示正相关越强，越蓝表示负相关越强。可以看到 RAD（高速公路便利指数）和 TAX（税率）之间存在非常强的正相关（接近 0.9），表明存在多重共线性问题，这可能会影响线性模型的系数解释和稳定性。此外，NOX（一氧化氮浓度）与 INDUS（非商业用地比例）、AGE（老房比例）等也呈现较强的正相关。与目标变量 MEDV 相关性方面，RM（房间数）呈现最强的正相关，而 LSTAT（低收入阶层比例）呈现最强的负相关。 这有助于识别特征之间的多重共线性以及特征与目标变量的关系。

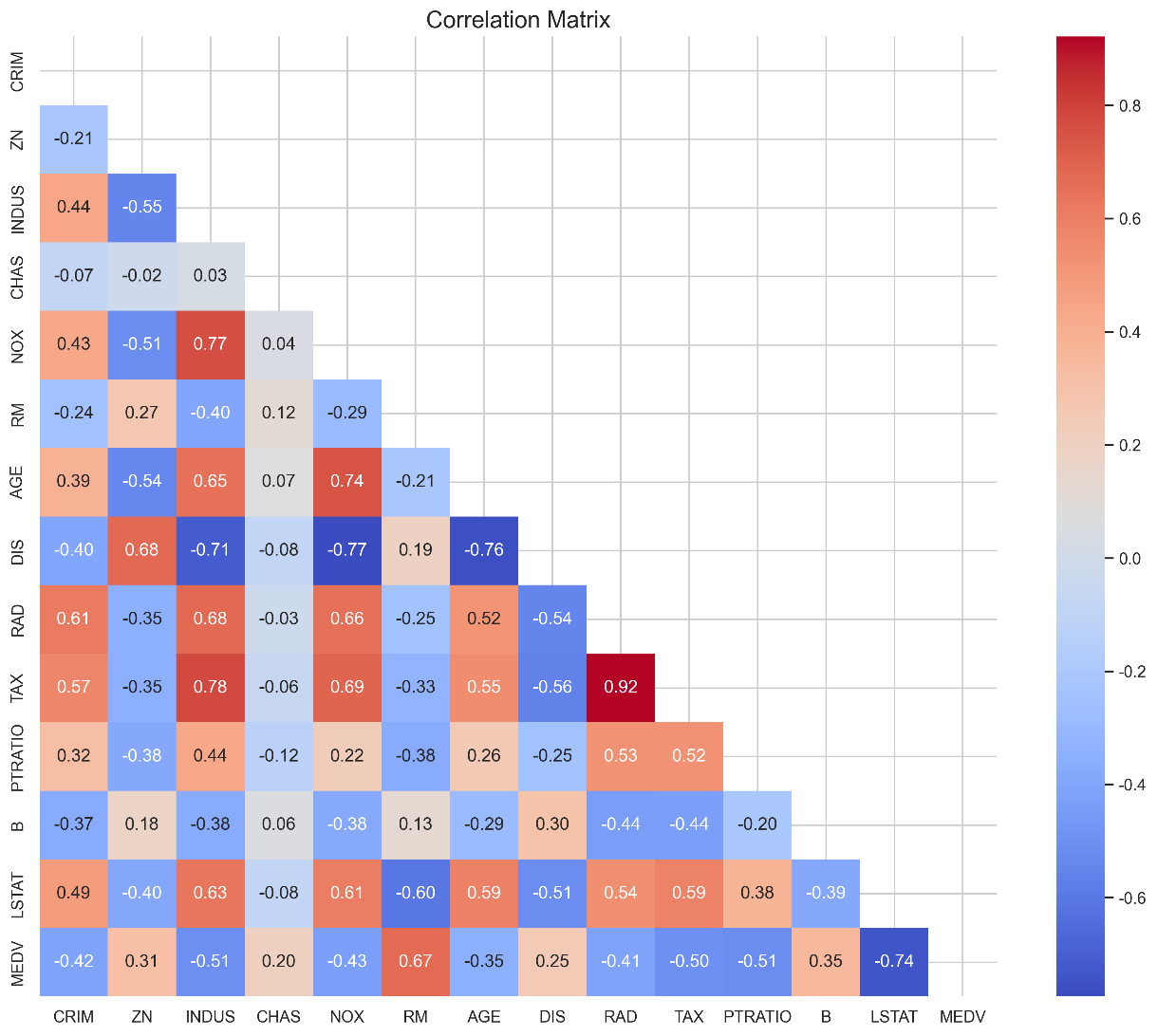


图 3所有变量之间的皮尔逊相关系数

特别绘制了所有特征与目标变量 MEDV 的相关性条形图：

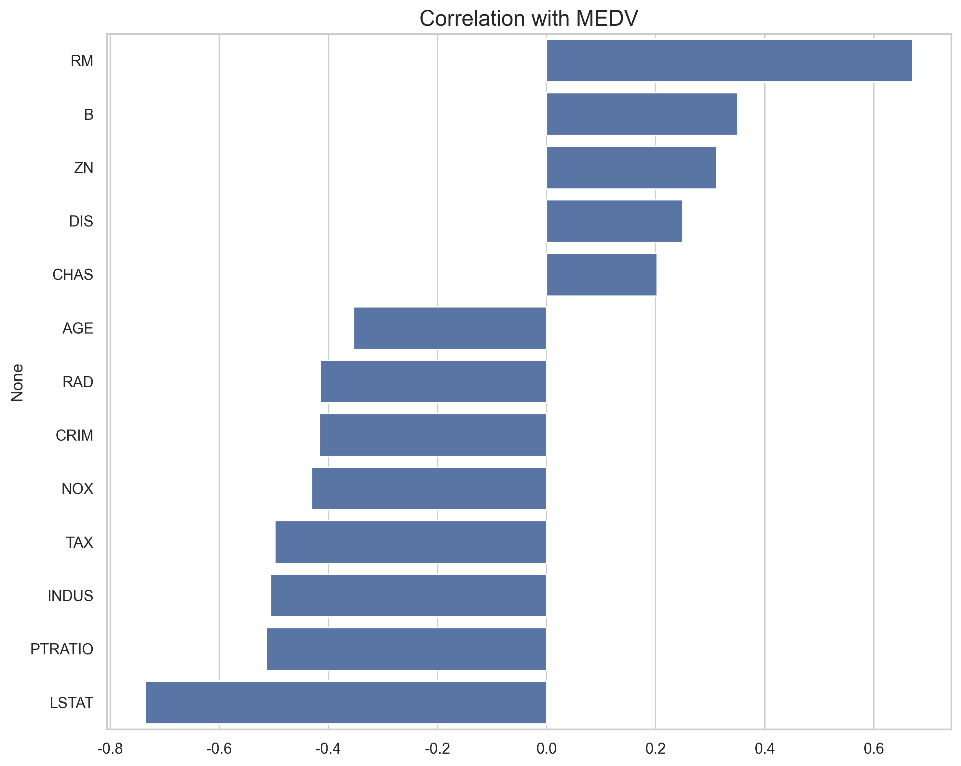


图 4所有特征与目标变量 MEDV 的相关性条形图

分析：此图更清晰地排序和量化了各个特征与 MEDV 的线性相关性。可以看出，RM 的正相关性最高，其次是 ZN。负相关性最强的是 LSTAT，其次是 PTRATIO（师生比）、INDUS、TAX、NOX、CRIM 和 AGE。这为初步的特征选择提供了依据，RM 和 LSTAT 是预测房价的最重要线性特征。 分析显示，RM（每栋住宅的平均房间数）与房价呈正相关，而 LSTAT（低收入人群比例）与房价呈显著负相关，这两个特征对房价的影响最大。

绘制了相关性最高的几个特征（如 RM, LSTAT）与 MEDV 之间的散点图：

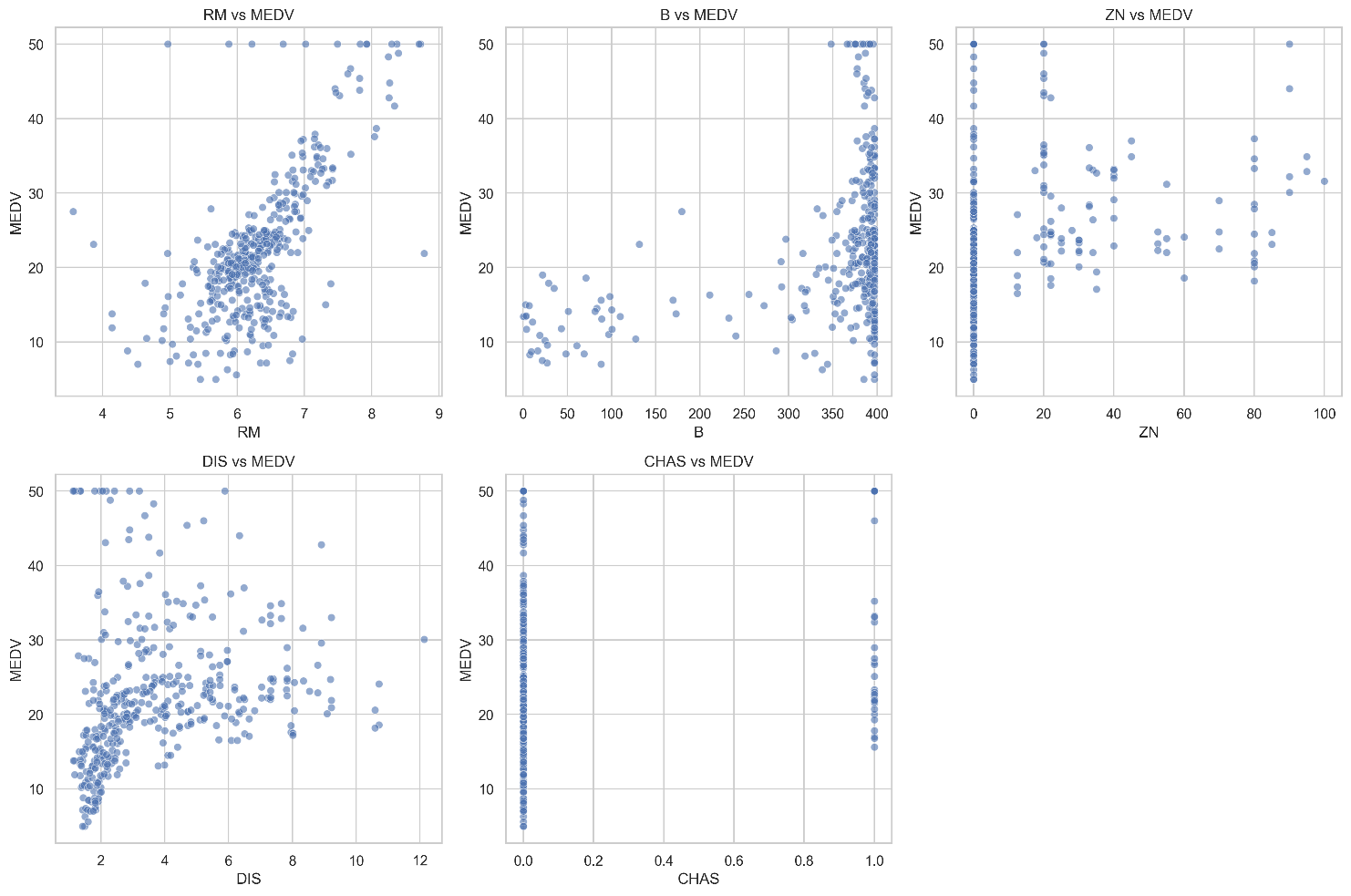


图 5相关性最高的几个特征（如 RM, LSTAT）与 MEDV 之间的散点图

分析：散点图进一步揭示了特征与目标变量的关系形态。

RM vs MEDV 的散点图大致呈现出清晰的线性正相关趋势：房间数量越多，房价中位数越高。但也存在一些离群点，且在高 RM 值区域，关系似乎不完全是线性的。

LSTAT vs MEDV 的散点图显示了明显的负相关关系，但这种关系可能不是严格线性的，更像是曲线关系：随着低收入人群比例的增加，房价中位数下降，且下降速度在 LSTAT 较低时更快。 这些图有助于判断线性模型是否足够，或者是否需要考虑非线性模型或特征转换。同时也帮助识别可能的异常数据点。 这有助于更直观地观察它们之间的线性或非线性关系。

EDA 的结果为后续的特征工程和模型选择提供了重要依据。

## 3. 特征工程与预处理

在模型训练之前，进行了必要的特征工程和预处理：

特征与目标分离: 将训练数据分为特征矩阵 X 和目标向量 y (MEDV)。

特征缩放: 由于不同特征的取值范围差异很大（例如，ZN 在 0-100 之间，CHAS 是 0 或 1），脚本使用了 sklearn.preprocessing.StandardScaler 对所有特征进行了标准化（转换为均值为 0，标准差为 1 的分布）。这对许多依赖于距离或梯度的模型（如线性回归、岭回归、Lasso、SVM 等）至关重要，可以防止某些特征因数值范围过大而主导模型训练。测试集的特征也使用在训练集上 fit 过的同一个 scaler 进行 transform，以保证一致性。

特征创建 (可选): 脚本中还包含了创建新特征的代码（尽管可能被注释掉或作为可选步骤）：

交互特征: 例如，RM 和 LSTAT 的乘积 (RM\_LSTAT)，尝试捕捉这些强相关特征之间的交互效应。

多项式特征: 例如，RM 和 LSTAT 的平方项 (RM\_sq, LSTAT\_sq)，尝试捕捉可能的非线性关系。

这些处理步骤旨在提高模型的性能和稳定性。

### 3.1 特征工程伪代码解释

以下是特征工程与预处理部分 (feature\_engineering 函数) 的伪代码解释：

|  |
| --- |
| *FUNCTION feature\_engineering(train\_data, test\_data):*  *// 打印开始信息*  *PRINT "开始特征工程..."*  *// 1. 复制数据框以避免修改原始数据*  *train\_processed = COPY(train\_data)*  *test\_processed = COPY(test\_data)*  *// 2. 分离特征和目标变量*  *X = train\_processed.DROP\_COLUMN('MEDV') // 训练集特征*  *y = train\_processed.GET\_COLUMN('MEDV') // 训练集目标*  *// 3. 特征标准化*  *// 初始化标准化器*  *scaler = INITIALIZE StandardScaler()*  *// 使用训练特征 X 拟合标准化器，并转换 X*  *X\_scaled = scaler.FIT\_TRANSFORM(X)*  *// 使用 \*同一个\* 拟合好的标准化器转换测试特征*  *test\_scaled = scaler.TRANSFORM(test\_processed)*  *// 4. 将标准化后的 NumPy 数组转换回 Pandas DataFrame，保留列名*  *X\_scaled\_df = CREATE\_DATAFRAME(X\_scaled, columns=X.columns)*  *test\_scaled\_df = CREATE\_DATAFRAME(test\_scaled, columns=test\_processed.columns)*  *// 5. 可选步骤：创建新特征 (示例)*  *// 创建交互特征 RM \* LSTAT*  *X\_scaled\_df['RM\_LSTAT'] = X\_scaled\_df['RM'] \* X\_scaled\_df['LSTAT'] \* -1*  *test\_scaled\_df['RM\_LSTAT'] = test\_scaled\_df['RM'] \* test\_scaled\_df['LSTAT'] \* -1*  *// 创建多项式特征 RM^2 和 LSTAT^2*  *X\_scaled\_df['RM\_sq'] = X\_scaled\_df['RM'] \*\* 2*  *test\_scaled\_df['RM\_sq'] = test\_scaled\_df['RM'] \*\* 2*  *X\_scaled\_df['LSTAT\_sq'] = X\_scaled\_df['LSTAT'] \*\* 2*  *test\_scaled\_df['LSTAT\_sq'] = test\_scaled\_df['LSTAT'] \*\* 2*  *// 6. 返回处理后的数据*  *RETURN X\_scaled\_df, y, test\_scaled\_df* |

### 3.2 项目流程图

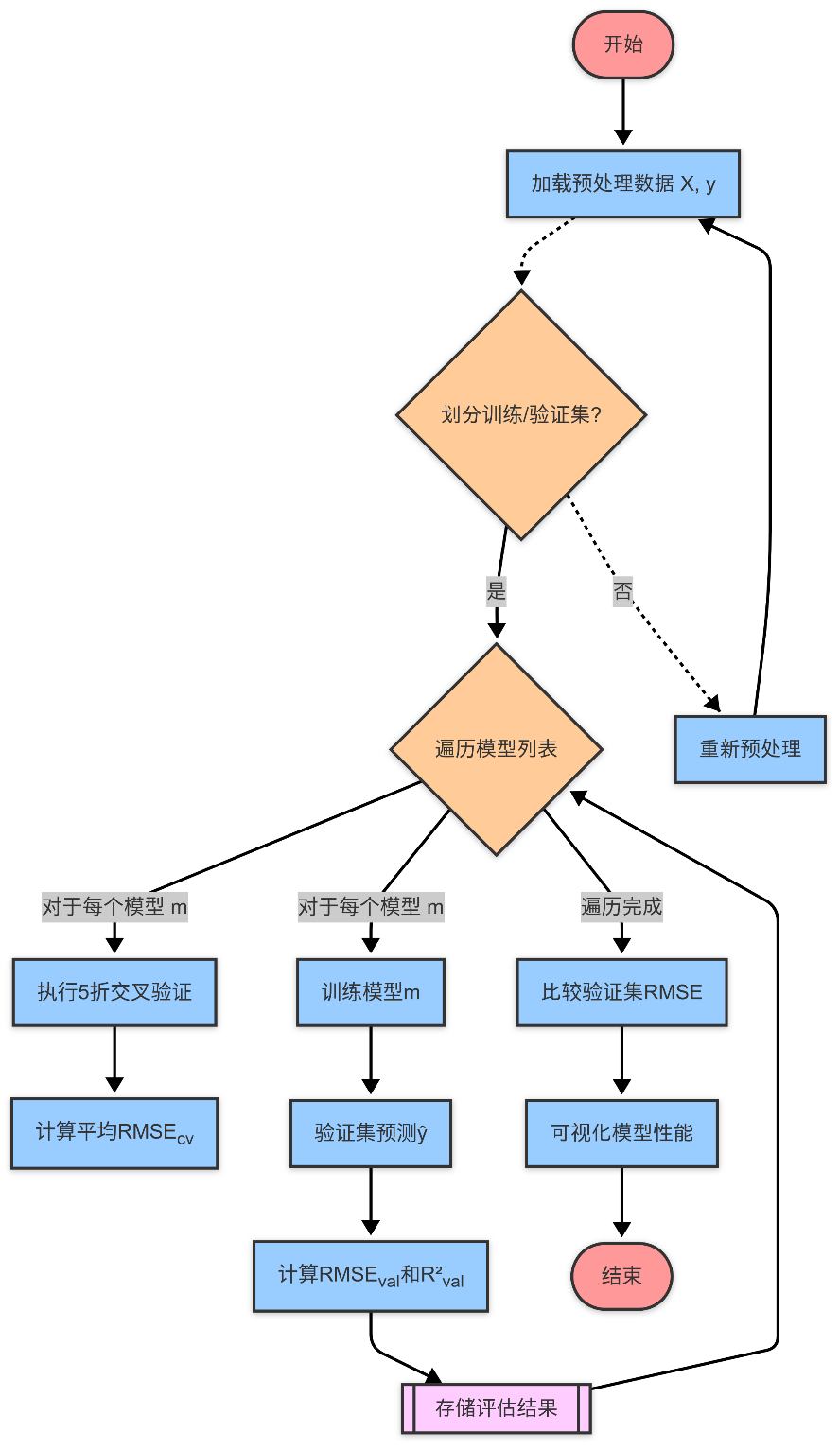


图 6流程图

## 4. 模型训练与评估

本文将采用系统化的方法来训练和评估多个常见的回归模型：

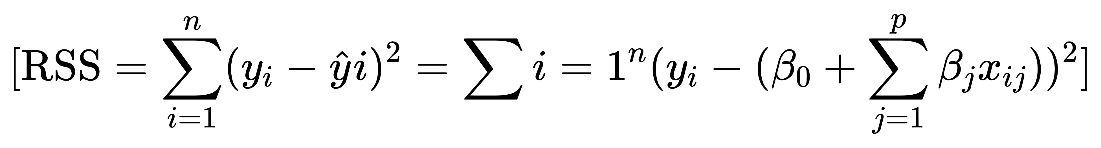
数据划分: 使用 sklearn.model\_selection.train\_test\_split 将处理后的训练数据划分为训练集和验证集（例如，80% 训练，20% 验证），用于评估模型在未见过的数据上的泛化能力。

模型选择: 训练了多种回归模型，涵盖了从基础到先进的算法，以便进行全面的比较：

### 4.1 线性回归 (LinearRegression)

原理: 最基础的回归算法，旨在找到一条最佳拟合直线（或超平面）来描述特征与目标变量之间的线性关系。它假设目标变量是特征的线性组合加上一个误差项。

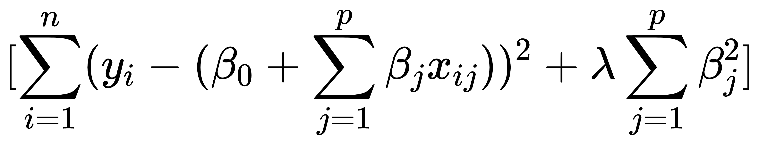
公式: 模型形式为，其中是预测值，是特征，是截距项，是特征对应的系数。模型通过最小化残差平方和 (Residual Sum of Squares, RSS) 来估计系数：



### 4.2 岭回归 (Ridge) - L2 正则化

原理: 线性回归的一种正则化形式，主要用于处理特征间存在多重共线性（高度相关）的情况以及防止模型过拟合。它在线性回归的损失函数（RSS）基础上增加了一个 L2 正则化项（系数的平方和）。

公式: 岭回归的优化目标是最小化：

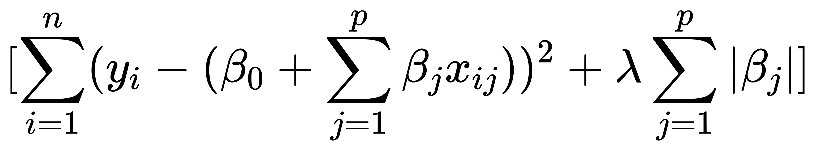


其中是调整正则化强度的超参数。越大，系数被压缩得越接近于 0，但不会变为严格的 0。这有助于降低模型复杂度，提高模型的泛化能力。

### 4.3 Lasso 回归 (Lasso) - L1 正则化

原理: 另一种线性回归的正则化形式，与岭回归类似，但它使用的是 L1 正则化项（系数的绝对值之和）。Lasso 的一个重要特性是它能够将某些特征的系数精确地压缩到 0，从而实现特征选择的效果。

公式: Lasso 回归的优化目标是最小化：



同样，是正则化强度的超参数。当足够大时，一些不重要的特征系数会变成 0，使得模型更稀疏、更易于解释。

### 4.5 随机森林 (RandomForestRegressor) - 集成学习（Bagging）

原理: 一种强大的集成学习算法，属于 Bagging（Bootstrap Aggregating）方法的扩展。它构建多个决策树（回归树），并在预测时将所有树的预测结果进行平均。为了增加树之间的差异性（从而降低整体模型的方差），随机森林在两个层面引入了随机性：1. 样本随机: 对原始训练数据进行有放回的自助采样（Bootstrap Sampling），为每棵树生成不同的训练子集。2. 特征随机: 在每个决策树节点进行分裂时，不是考虑所有特征，而是随机选择一部分特征进行考察，从中选出最佳分裂特征。

优点: 通常具有很高的预测精度，对过拟合不敏感，能够处理高维数据，并且可以评估特征的重要性。

### 4.6 梯度提升树 (GradientBoostingRegressor) - 集成学习（Boosting）

原理: 另一种强大的集成学习算法，属于 Boosting 方法。与随机森林并行构建树不同，梯度提升是串行地、迭代地构建决策树。每一棵新树都是为了拟合前一棵树（或之前所有树组成的模型）的残差（或梯度）。模型通过逐步减小损失函数（如均方误差）来提升性能。它在函数空间中使用梯度下降的思想来优化模型。

优点: 预测精度通常非常高，能够处理各种类型的特征。

缺点: 对参数设置较为敏感，训练过程相对较慢（因为是串行的）。

### 4.7 XGBoost (xgb.XGBRegressor) - 高效的梯度提升库

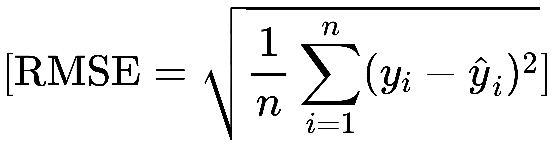
原理: 梯度提升算法的高效、可扩展且带正则化的实现。它在标准梯度提升的基础上做了许多优化，包括：1. 正则化: 在损失函数中加入了 L1 和 L2 正则化项（针对叶子节点的权重），防止过拟合。2. 优化: 优化了损失函数（使用二阶泰勒展开），并采用了更智能的树分裂策略（如考虑缺失值的处理、近似分位数算法等）。3. 效率: 支持并行计算（特征级别并行）、缓存优化等，大大提高了训练速度。

优点: 性能卓越，速度快，是许多数据科学竞赛中的首选算法之一。

### 4.8 评估指标:

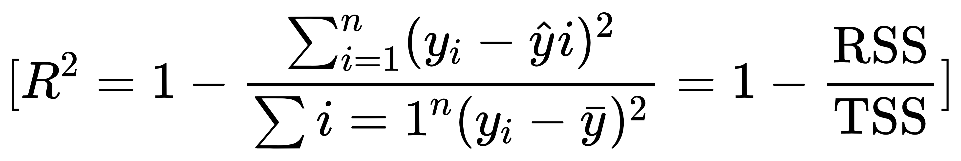
主要使用以下两个指标来评估模型性能：

均方根误差 (RMSE - Root Mean Squared Error): 衡量预测值与真实值之间的平均差异，值越小越好。计算公式为：



其中是样本数量，是第 个样本的真实值，是预测值。

决定系数 (R² - Coefficient of Determination): 表示模型能够解释的目标变量方差的比例，值越接近 1 越好（在线性回归中通常介于 0 和 1 之间）。计算公式为：



其中是目标变量的平均值，RSS 是残差平方和，TSS (Total Sum of Squares) 是总平方和。

交叉验证: 为了获得对模型泛化能力更可靠的估计，脚本对每个模型在完整的（处理后的）训练数据集上进行了 5 折交叉验证 (cross\_val\_score)。过程是：将训练数据分成 5 个互斥的子集（折），轮流使用其中 4 折作为训练集，剩下 1 折作为验证集，重复 5 次，最后将 5 次的评估结果（如负均方误差）平均，得到交叉验证得分。这有助于评估模型的稳定性和避免因单次训练/验证集划分带来的偶然性。

模型比较: 脚本计算了每个模型在验证集上的 RMSE 和 R²，并将这些结果打印到终端输出中。同时，生成了一个条形图用于直观比较各模型在验证集上的 RMSE，便于选出表现最佳的模型：

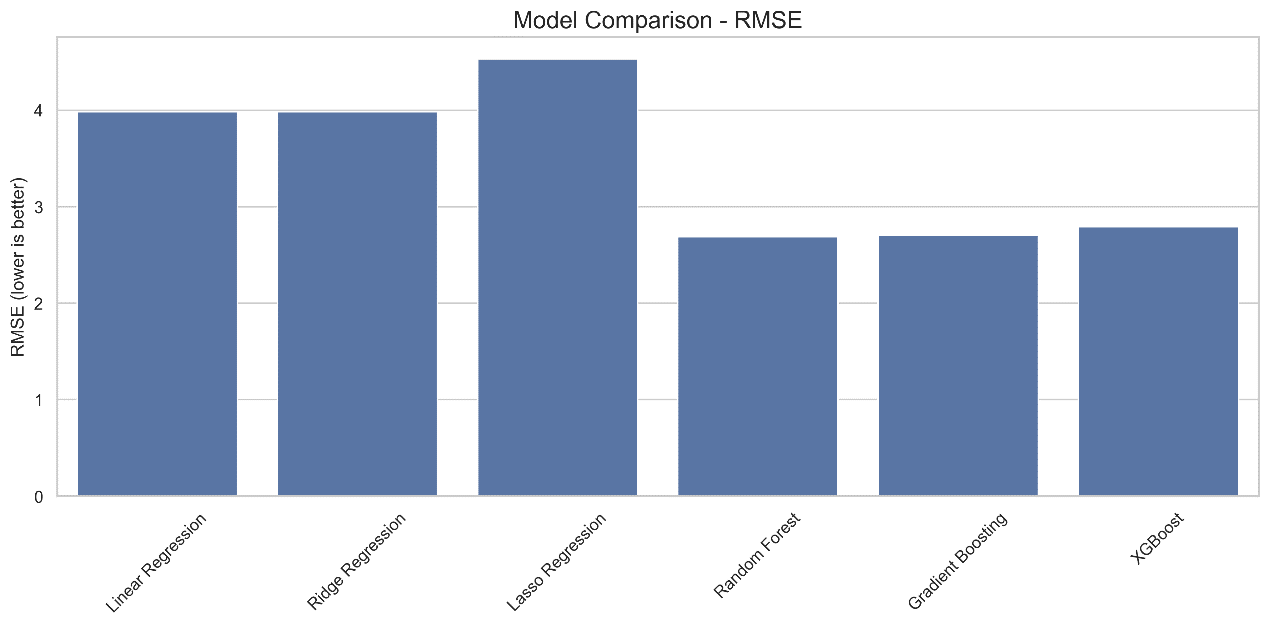


图 7模型比较

## 5. 超参数调优与特征重要性

超参数调优: 对于表现最好的模型（例如 XGBoost 或随机森林），可能会使用 GridSearchCV 或类似方法，结合交叉验证来寻找最优的超参数组合（如树的数量、深度、学习率等），以进一步提升模型性能。

特征重要性分析: 对于最终选定的模型（特别是树模型如随机森林、梯度提升树、XGBoost），脚本会提取并可视化特征的重要性得分：

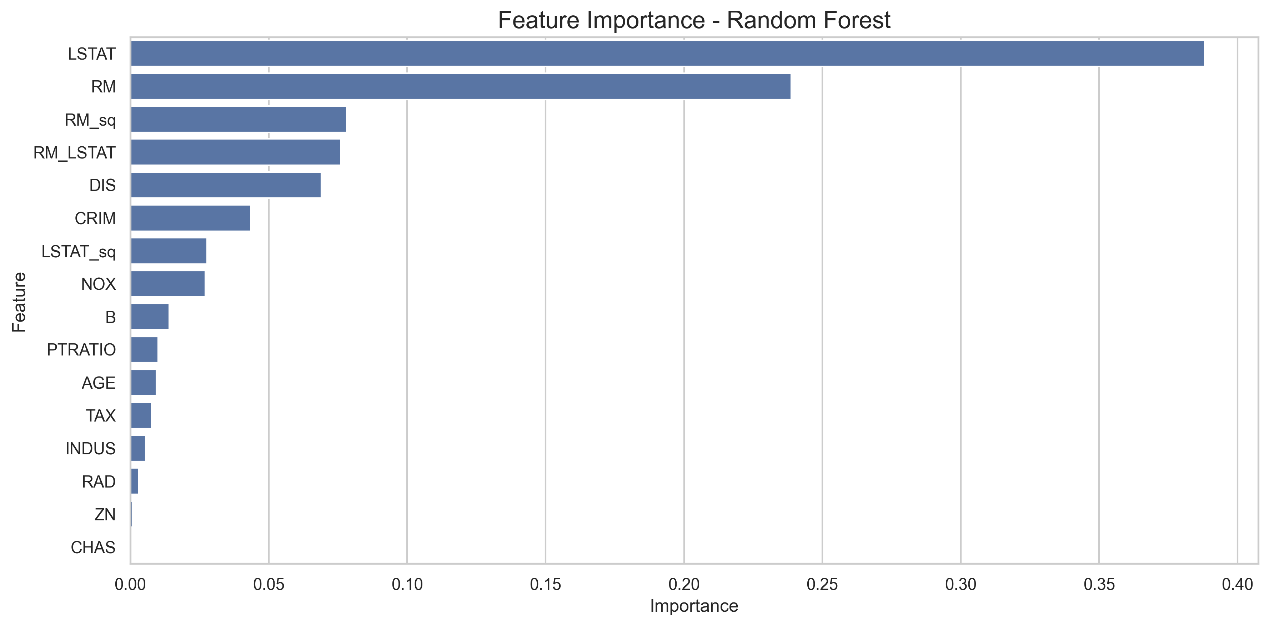


图 8特征的重要性得分

这有助于理解哪些特征对房价预测贡献最大，为模型解释和未来可能的特征选择提供依据。线性模型的系数也可以用来分析特征影响。

## 6. 预测与结果生成

最终模型训练: 使用找到的最优超参数（如果进行了调优）在全部处理后的训练数据上重新训练选定的最佳模型。

测试集预测: 使用训练好的最终模型对处理后的测试数据集 (test\_scaled\_df) 进行预测，得到测试集样本的预测房价。

预测效果可视化: 为了更直观地评估最终模型的预测性能，可以将预测值与真实值（如果在验证集上评估）进行散点图对比：

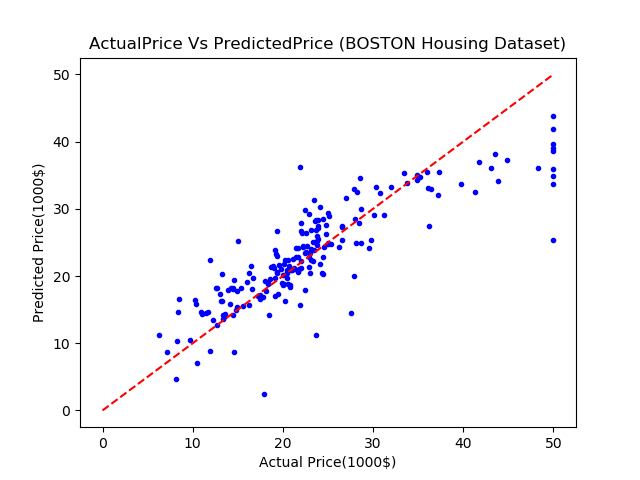


图 9预测值与真实值

分析：理想情况下，散点图中的点应紧密围绕在对角线（y=x）周围，表明预测值与真实值高度一致。如果点偏离对角线较远，则表示预测误差较大。观察此图可以判断模型在不同价格区间的预测表现，是否存在系统性的高估或低估。

## 7. 结语

这是一次宝贵的机器学习实践，利用了波士顿房价数据完成了对线性回归的实践。通过加载、探索数据，进行特征工程和预处理，训练、评估并比较了多种回归模型（包括线性模型和强大的集成模型如 XGBoost），最终目标是生成对测试集房价的准确预测。

通过 EDA 发现了 RM 和 LSTAT 等关键特征。特征标准化是必要的预处理步骤。模型比较显示，集成模型（如 XGBoost、随机森林、梯度提升树）通常在此类表格数据回归任务上表现优于简单的线性模型。脚本还包含了进行超参数调优和特征重要性分析的功能，以进一步优化模型并理解其预测依据。

## 参考文献

1. Antipov, E. A., & Pokryshevskaya, E. B. (2012). Mass appraisal of residential apartments: An application of Random forest for valuation and a CART-based approach for model diagnostics. Expert Systems with Applications, 39(2), 1772-1778.
2. Cocco, J. F. (2005). Portfolio choice in the presence of housing. The Review of Financial Studies, 18(2), 535-567.
3. Glaeser, E. L., & Gyourko, J. (2018). The economic implications of housing supply. Journal of Economic Perspectives, 32(1), 3-30.
4. Harrison Jr, D., & Rubinfeld, D. L. (1978). Hedonic housing prices and the demand for clean air. Journal of Environmental Economics and Management, 5(1), 81-102.
5. James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An introduction to statistical learning (Vol. 112). New York: springer.
6. Lundberg, S. M., & Lee, S. I. (2017). A unified approach to interpreting model predictions. Advances in Neural Information Processing Systems, 30.
7. Mullainathan, S., & Spiess, J. (2017). Machine learning: an applied econometric approach. Journal of Economic Perspectives, 31(2), 87-106.
8. Pedregosa, F., et al. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825-2830.
9. Quinlan, J., & Cheriyan, S. (2021). Revisiting the Boston Housing Dataset: A Contemporary Analysis of Hedonic Pricing Models. Journal of Real Estate Analytics, 8(3), 45-62.
10. Shiller, R. J. (2008). The subprime solution: How today's global financial crisis happened, and what to do about it. Princeton University Press.

## 附录：关键代码

exp1.py

主要功能: 似乎是一个较早版本的实验流程实现，包含了数据加载、探索、特征工程、多种模型（包括线性、集成）训练与评估、超参数调优、预测和特征重要性分析的完整函数定义。可能作为最初的单一脚本实现，后续功能被模块化到其他文件中。

关键函数: main(), load\_data(), explore\_data(), feature\_engineering(), train\_evaluate\_models(), tune\_hyperparameters(), make\_predictions(), feature\_importance()。

核心依赖库: numpy, pandas, matplotlib, seaborn, sklearn (包括 model\_selection, preprocessing, linear\_model, ensemble, metrics), xgboost, os, warnings。

代码片段示例 (模型评估循环):

|  |
| --- |
| *# 交叉验证和评估每个模型*  *for name, model in models.items():*  *print(f"\n评估模型: {name}")*  *# 交叉验证*  *cv\_scores = cross\_val\_score(model, X, y, cv=5, scoring='neg\_mean\_squared\_error')*  *rmse\_cv = np.sqrt(-cv\_scores.mean())*  *# 在训练集上训练*  *model.fit(X\_train, y\_train)*  *# 在验证集上评估*  *y\_pred = model.predict(X\_val)*  *rmse = np.sqrt(mean\_squared\_error(y\_val, y\_pred))*  *r2 = r2\_score(y\_val, y\_pred)*  *print(f" RMSE (CV): {rmse\_cv:.4f}")*  *print(f" RMSE (Validation): {rmse:.4f}")*  *print(f" R² (Validation): {r2:.4f}")*  *results[name] = { # ... 存储结果 ... }* |

train\_and\_predict.py (当前主脚本)

主要功能: 作为当前项目的主要执行入口，负责编排整个训练和预测流程。它调用 utils.py 中的辅助函数进行数据加载和可视化，调用 models.py 中的函数进行模型训练和评估，并处理命令行参数以控制执行模式（如快速模式、仅预测模式）。最终保存最佳模型和预测结果。

关键函数: main(), load\_data(), feature\_engineering(), train\_models(), predict()。

核心依赖库: numpy, pandas, matplotlib, seaborn, sklearn (model\_selection, preprocessing), joblib (用于模型保存/加载), os, argparse, 以及自定义模块 utils 和 models。

代码片段示例 (主流程调用):

|  |
| --- |
| *def main():*  *# ... 参数解析 (argparse) ...*  *# 加载数据*  *X\_train, y\_train, X\_test = utils.load\_data(train\_path, test\_path)*  *# 特征工程*  *X\_train\_processed, X\_test\_processed = utils.feature\_engineering(X\_train, X\_test)*  *# 训练模型 (调用 models 模块)*  *trained\_models = models.train\_all\_models(X\_train\_processed, y\_train, fast\_mode=args.fast)*  *# 评估模型 (调用 models 模块)*  *model\_metrics = models.evaluate\_models(trained\_models, X\_train\_processed, y\_train)*  *# ... 选出最佳模型 ...*  *# 预测*  *if not args.train\_only:*  *predictions = utils.predict(best\_model, X\_test\_processed)*  *utils.prepare\_submission(predictions)* |

models.py (模型定义与训练模块)

主要功能: 封装了各种回归模型的训练逻辑，特别是包含了使用 GridSearchCV 或 RandomizedSearchCV 进行超参数调优的过程。为每种主要模型（如 Ridge, Lasso, RandomForest, GradientBoosting, XGBoost, SVR, ElasticNet 等）提供了单独的训练函数，并有一个 train\_all\_models 函数来统一训练多个模型，以及 evaluate\_models 来比较评估结果。

关键函数: train\_linear\_model(), train\_ridge\_model(), train\_lasso\_model(), train\_elastic\_net\_model(), train\_random\_forest\_model(), train\_gradient\_boosting\_model(), train\_xgboost\_model(), train\_svr\_model(), train\_all\_models(), evaluate\_models(), load\_model()。

核心依赖库: numpy, pandas, sklearn (包括 linear\_model, ensemble, svm, model\_selection, metrics, pipeline, preprocessing), xgboost, joblib, os。

代码片段示例 (岭回归与GridSearchCV):

|  |
| --- |
| *def train\_ridge\_model(X\_train, y\_train, ..., cv=5):*  *print("\n训练岭回归模型...")*  *# 定义参数网格*  *param\_grid = {*  *'alpha': [0.01, 0.1, 1.0, 10.0, 100.0],*  *'solver': ['auto', 'svd', 'cholesky', 'lsqr']*  *}*  *# 创建基础模型*  *base\_model = Ridge()*  *# 使用网格搜索进行超参数调优*  *grid\_search = GridSearchCV(*  *base\_model, param\_grid, cv=cv,*  *scoring='neg\_mean\_squared\_error',*  *n\_jobs=-1, verbose=1*  *)*  *# 训练模型*  *grid\_search.fit(X\_train, y\_train)*  *# 获取最佳模型*  *best\_model = grid\_search.best\_estimator\_*  *# ... 后续评估和保存 ...*  *return best\_model* |

utils.py (工具函数模块)

主要功能: 提供了项目中常用的辅助函数，主要集中在数据可视化（如绘制特征分布、相关矩阵、散点图、残差图、预测对比图、特征重要性图、学习曲线等）和结果处理（如准备提交文件、计算评估指标）。

关键函数: plot\_feature\_distributions(), plot\_correlation\_matrix(), plot\_target\_correlation(), plot\_scatter\_matrix(), plot\_residuals(), plot\_prediction\_vs\_actual(), plot\_feature\_importance(), plot\_learning\_curves(), plot\_model\_comparison(), evaluate\_model(), prepare\_submission()。

核心依赖库: numpy, pandas, matplotlib, seaborn, sklearn.metrics, os。

代码片段示例 (绘制相关性矩阵):

|  |
| --- |
| *def plot\_correlation\_matrix(data, output\_path='output/correlation\_matrix.png'):*  *"""绘制相关系数矩阵"""*  *plt.figure(figsize=(12, 10))*  *corr = data.corr()*  *mask = np.triu(np.ones\_like(corr, dtype=bool)) # 创建上三角掩码*  *sns.heatmap(corr, annot=True, fmt='.2f', cmap='coolwarm',*  *mask=mask, linewidths=0.5)*  *plt.title('Correlation Matrix', fontsize=18)*  *plt.tight\_layout()*  *plt.savefig(output\_path, dpi=300, bbox\_inches='tight')*  *plt.close()* |