## Validation report

Team 3 俞项天 梁泓铭 江孟书 胡熙媛

我们将对手组的代码分为三个部分，数据预处理、拟合模型、模型的进一步设计，由三位同学分别进行一部分的阅读和评议，最后由胡熙媛同学进行补充。

俞项天（数据预处理部分）：

Advantages：

* 使用随机采样填补缺失值，完善了数据的统计学意义。
* 按板块分组，对于分类变量使用众数填充，对数值变量使用中位数填充。非常细致。
* 对每个变量单独分区，使用函数编译，结构清晰。

Something to improve：

* 在环线的处理上，将不同城市的相同环线标记为同一数字，再转化为独热变量。且不论标记的比较粗，例如将 '三至四环' '三环外''外环外' '四环外''四至五环''五至六环''六环外'都标记为3。然后放入模型中，此时模型中有包含另一个城市独热变量。此时由于我们的模型是线性模型，其无法自行学习交互项。如此一来，被标记为3的项将会平均所有城市的被标记为此类的环线特征，这必然会导致相当大的误差。
* 中文数字到阿拉伯数字的映射可以直接调包，无需输入字典。
* 在对测试集的处理部分，test\_data=test并未创建test的副本，而是引用标签，引发了链式索引，导致输出了巨量Warnings。严重影响代码阅读体验。

梁泓铭（拟合模型部分）：

代码中的R方、MAE、MSE和图标基本上可以复现。

Advantages:

1. 绘制了残差-预测值和真实值-预测值之间的散点图，可以直观看出模型在的预测表现，并得出了“极大高估高价值的房产价格”的结论，对后续模型优化有很大意义；

Something to improve：

1. 可能的data leakage: 在划分测试集和训练集之前，进行随机采样和中位数填补缺失值，用到了测试集的信息；

2. 经纬度不经过处理直接加入了线性模型，而经纬度与房价之间是否存在线性关系值得怀疑；虽然未经过尝试，直觉上认为距离市中心的距离作为变量是更好的选择；

3. 对价格进行了对数处理后MAE和MSE的尺度也发生了变化，将MAE，MSE还原为原来的尺度会更直观；

4. 线性回归没有进行期中考核要求的交叉验证

江孟书：（模型的进一步的设计部分）：

✅ 一、Advantage

1. 数据预处理模块化清晰

使用了 Pipeline 和 ColumnTransformer 组织整个建模流程：

preprocessor = ColumnTransformer(...)

model\_pro = Pipeline(steps=[('preprocessor', preprocessor), ('regressor', LinearRegression())])

好处是：

所有特征处理逻辑都被封装在一个模块中，方便复用；

能直接和 GridSearchCV、cross\_val\_score 搭配使用，避免数据泄露；

保证测试集不会“偷看到”训练集的变换参数（如标准差、均值等）。

2. 引入非线性项和交互项

针对一些核心特征引入了 PolynomialFeatures(degree=2 or 4)：

('poly', PolynomialFeatures(degree=2, include\_bias=False))

它的作用是：

自动生成这些变量的一次、二次项，以及两两之间的乘积（交互项）；

可以表达例如“建筑面积大+户型紧凑”的组合效应；

尤其适合模拟高价房的“复杂构成”，这正是观察到的模型“高估”问题的突破口。

3. 目标变量使用对数变换

对 Y 使用了：

log\_y = np.log1p(Y)

意义：

抑制了房价数据中的极端高值；

缩小了方差，提高了模型稳定性；

符合经济数据常见的“相对变化”视角。

在预测输出时还做了逆变换 np.expm1，这说明数据处理得非常细致和完整，符合回归建模的标准流程。

4. 异常值处理逻辑明确

自定义了一个函数 outlier()：

condition1 = df["套内面积"] > df["建筑面积"]

condition2 = (df["套内面积"] / df["建筑面积"]) < 0.1

优点：

不是盲目地用 IQR 或 Z-score，而是根据业务常识（套内面积不可能大于建筑面积）来做清洗；

加入了“房屋总数 > 10000”、“建筑面积 > 2000”的极端情况过滤，贴近实际。

这个设计筛除掉了对模型干扰较大的异常房源，提升了模型的稳健性。

5. 可视化残差图进行模型诊断

通过残差图可以看出：

plt.scatter(np.expm1(filtered\_data[mask\_abs]), filtered\_residuals[mask\_abs], alpha=0.5)

作用：

检查模型是否存在系统性偏误（比如所有预测都偏高、偏低）；

看是否有“扇形图案”（异方差性）；

看高价区域是否预测误差更大。

❌ 二、Disadvantage

1. 多项式维度过高 → 可能引起过拟合

PolynomialFeatures(degree=4) 对 6 个变量扩展后会生成：

特征数=(6+44)−1=125−1=124个新特征\text{特征数} = \binom{6+4}{4} - 1 = 125 - 1 = 124 个新特征

在样本量不够大（比如几千行）的情况下，这样的扩展非常容易：

导致冗余维度；

使得线性回归解变得不稳定；

误差在测试集上变大（过拟合现象）。

2. 没有使用正则化（Ridge/Lasso）

当前模型：

model\_pro = Pipeline([... , ('regressor', LinearRegression())])

问题：

LinearRegression() 本身不具备任何特征选择能力；

在高维场景（尤其用了多项式特征）中，容易出现“系数过大”、“符号反复”的现象；

模型可能在训练集上表现很好，在测试集却崩塌。

3. 未将特征筛选结果用于建模

计算了：

pearson\_scores = {col: pearsonr(X[col], Y)[0] for col in numeric\_features}

f\_scores = f\_classif(X[categorical\_features], Y)

但后续建模时这些结果并未用于“筛选掉低贡献特征”；

这意味着可能纳入了一些对预测几乎无用，甚至引入噪声的变量。

4. 测试集处理代码冗余

多次出现：

x\_out=test\_data1.drop(columns=[...])

重复定义特征删减列，容易出错；

若列名稍有不同或后续改动，极易遗漏。

可以封装为函数：

def preprocess\_test\_data(df):

drop\_cols = ['小区名称', '房屋年限', ...]

return df.drop(columns=drop\_cols)

5. 缺少评估指标输出

只画了残差图，但没展示：

from sklearn.metrics import mean\_squared\_error, r2\_score, mean\_absolute\_error

这样无法准确比较不同模型（如 degree=2 vs 4 的多项式）哪一个更好。

建议补上：

print("RMSE:", mean\_squared\_error(y\_test, y\_pred1, squared=False))

print("MAE:", mean\_absolute\_error(y\_test, y\_pred1))

print("R²:", r2\_score(y\_test, y\_pred1))

6. 异常值阈值选择偏随意

使用 threshold\_abs = 1e8 过滤残差过大点，没有解释依据；

如果残差本来就有几千的量级，1e8 就完全不起作用。

建议改进：

residual\_std = residuals.std()

threshold = 3 \* residual\_std # 使用三倍标准差

mask\_abs = np.abs(residuals) < threshold

更有统计学根据，更适合通用模型筛选。

胡熙媛（补充）：

Advantages：

1.使用四分位间距(IQR)方法检测异常值，统计学上可靠，不受极端值影响；通用性强，适用于各种数据分布；容易解释，便于向非技术人员说明异常值标准

2.将预测结果保存为CSV文件，同时对异常值统计进行汇总：

结果可复现，便于后续分析；流程自动化，减少人工操作；模型可比较，便于评估不同模型的效果

3.设计了通用函数calculate\_and\_save,接收模型名称和缩放后的预测结果,处理逆变换、异常值检测、结果保存等全流程.

4.在主循环中灵活应用于多个模型,支持已训练模型和新训练模型两种情况

Disadvantages:

1.异常值检测方法过于简单化,仅使用IQR方法可能不够全面.房价数据通常呈现右偏分布，简单的IQR可能误判高价房为异常。可考虑使用更先进的方法如DBSCAN、Isolation Forest等

2.简单使用round()函数处理浮点数：

python'lower\_bound': round(lower\_bound, 2),

'upper\_bound': round(upper\_bound, 2),

未处理潜在的浮点数精度问题（如IEEE-754标准中的表示误差）

缺少非数值异常处理（如NaN、Inf值）

应使用numpy的专用函数或设置安全阈值处理边界情况

3.仅识别异常值但缺乏深入分析：

pythonis\_outlier = (predictions < lower\_bound) | (predictions > upper\_bound)

outlier\_count = sum(is\_outlier)

没有提供具体异常值的特征剖析

缺少异常值方向性分析（过高预测vs过低预测）

应添加聚类或特征重要性分析来解释异常值成因