第五小组模型核验与建议

1 分工情况

- **邵远平**：在团队作业中进行统筹和任务分配；主要负责分析和检查第六小组代码中的数据处理部分，结合整个代码对该部分进行评价；针对数据的处理方式、代码的写作形式和特征变量的选择提出建议。
- **李馨怡**：第 1、2、3 条建议。在成功复现第六组模型运行结果的基础上，从代码规范性、结构简洁性与运行效率三个方面对代码进行了细致分析，提出了三条建议。
- **方国圳**：主要负责分析第六小组代码中的模型训练部分，对样本划分和数据泄露问题进行了分析，提出了第 5 条和第 11 条建议，对数据预处理的均值填充和独热编码处理提出了改进建议。
- **滕明阳**：第 8、10、11、12 条建议，并补充了第 2、9 条建议。主要负责代码数据处理部分的复现、检查与修改，主要针对补全缺失值更加精确的方式，以及最大化利用数据信息的方法，更加简洁明了的代码写作风格提出了建议，还负责了最终 LaTeX 文档的撰写。

2 代码运行情况

经过李馨怡的检验，代码的运行结果与第六组的结果一致。

经过邵远平和滕明阳对数据处理部分的检查，发现在数据整合过程中，提取了 *detail* 数据集中的关键特征，并将其与所计算的每平米房价 *rent* 数据整合到了 *train* 与 *test* 数据集中；处理缺失值时，对于虚拟变量则赋为零，对于数值型变量则取全样本平均值或同城市平均值。在信息数据阶段，分别用建筑面积估算了套内面积；用独热编码生成了区域虚拟变量；在文字表述中搜索关键词以生成新虚拟变量；通过房屋结构计算户型得分；又生成了总楼层与相对高度的交叉项；利用函数为其他存在交互作用的变量生成交叉项。总之，在数据代码的数据处理有效地提取了部分重要的数据和文字，并进行了数据清洗，最终保证了后续回归的正常实现。

在样本划分部分，方国圳同学认为第 6 组的代码在进行样本划分部分大体上是正确的。代码使用 sklearn 包中的 KFold 函数进行样本的 6 折划分，在划分过程中，函数中的 shuffle 参数被设置为 True，说明代码在划分样本时对数据的原始顺序进行了打乱，使划分后得到的各个样本子集都能随机地包含不同的样本，提高了模型评估的准确性。如果没有进行打乱，那么数据的原始排列可能就代表了不同的样本分类。这样对样本进行划分就可能会导致一个样本子集里只包含一种类别的样本，影响模型的训练效果。

3 建议

1. 以下代码的变量名大小写有一点问题（将 training 改为 Training ，testing 改为 Testing），修改后代码仍能正常运行并运行结果与原小组的运行结果没有差异。

```
# 合并到Training
training_merged = pd.merge(
training,
detail_unique,
left_on='小区名称',
right_on='小区名称',
how='left'
)

# 合并到Testing
testing_merged = pd.merge(
testing,
detail_unique,
left_on='小区名称',
right_on='小区名称',
how='left'
)
```

2. 存在大量结构相同但逻辑仅参数不同的代码，代码太长，从简洁性考虑可以在这些地方使用循环（例如在生成交乘项时，源代码对于每个需要生成的交乘项变量都单独生成，可以将这类代码统一封装成一个批量交乘项构造函数，这样代码会更加简洁。）而且在生成数值型变量的非线性项时，要考虑到被解释变量的右偏分布，应该纳入对数项。

3. Linear/Lasso/Ridge 均有交叉验证和预测保存逻辑，且 Lasso 和 Ridgey 运算时间太长（Lasso 运行有 40 分钟），可以提取共用部分，生成一个能够共用的函数，提高执行效率。

4. 在数据处理过程中，使用整个样本的平均值填补空缺，这就导致这一均值的信息由后续交叉验证过程中的训练组和测试组的信息提供，这就产生的数据泄露的问题。另外，模型在对样本进行 6 折划分前对原始数据进行了一定的预处理，其中包含对一些数值变量的缺失值进行均值填充，这样做也有一定的数据泄露风险。因为均值填充使用的是整个数据集的平均值，所以这个填充值所在的样本子集就接受了额外的其他样本子集的信息，形成了数据泄露。这就使得对模型真实效果的评估可能偏高。因此可以试着先对样本进行划分，再在不同的样本子集里对数据的缺失值进行均值填充。

5. 数据没用对可能存在的极端值和异常值进行处理，因为这些偏离正常分布范围的数据点可能会严重影响统计分析结果的准确性和模型的性能表现。因此，应使用箱线图、散点图、直方图等直观地展示数据分布，帮助发现数据的异常点，并利用均值 ± 3 倍标准差、四分位距 (IQR) 法则对数据进行处理。在填充异常值的时候，也应当限制填充的范围，以防止异常值和极端值的出现。

6. 当前的代码在数据处理过程中使用直接处理和自定义函数混合的方式，此时不仅较为混乱，而且还会产生数据泄露的问题。因此，建议全部使用自定义函数的方式进行数据处理，并在数据分成训练组和测试组后再应用数据处理自定义函数分别进行处理，就可以使代码更加简洁有条理。

7. 产生的特征和交叉项可能存在较高的共线性，建议对特征进行筛选后再进行拟合。

8. 补全数据时，可以用精度更高的方法，如线性拟合，根据同一小区（而不是同一城市）的平均值或中位数拟合，等等。

9. 模型对样本数据集中如“区域”、“板块”等变量进行了独热编码处理，但都是在划分样本子集前完成的。可以在样本划分之后在各个子集上利用 sklearn 包中的 OneHotEncoder 类进行更灵活的独热编码操作，使用 handle_unknown='ignore' 参数可以忽略测试集中出现的新类别，避免报错。然后在训练集上使用 fit_transform 方法进行拟合和转换。这样可以更好防止数据泄露问题。当然，也可以采用寻找邻居这种更加细致的方法处理测试集中存在新类别的问题。

10. 以下代码的逻辑比较奇怪，为何采用 0.95 作为系数用建筑面积估计套内面积？如果考虑两个变量比值的平均数，应为 0.80 左右。另外，对于并未缺失套内面积数据的观测点，为何也要再次“估计”套内面积而覆盖原数据？建议保留所有原始数据，重新估算套内面积与建筑面积之间的比例。

```
mask_train = Traindata['套内面积'].isna()
Traindata.loc[mask_train, '套内面积'] = 0.95 * Traindata['建筑面积']
Traindata.loc[~mask_train, '套内面积'] = 0.2*Traindata['建筑面积'] + 0.8*Traindata['套内面积']
```

11. 以下代码的逻辑同样奇怪，如此计算户型得分似乎没有依据。建议可以直接记下每套房子的室厅厨卫数量，作为数值型变量纳入特征矩阵即可。

```
#—— 后续处理 ——
# 按比例计算户型得分（可调整权重）
df['户型得分'] = df['室']*1 + df['厅']*0.5 + df['厨']*0.3 + df['卫']*0.3
```

12. 事实上，代码中对于测试集与训练集的操作几乎全部都是平行的，可以设计 data_processing() 函数直接同时进行两组数据的整合与清洗；也可以同时设计 get_feature_matrix() 函数直接得到训练集与测试集的特征矩阵。需要注意的是，有时必须对测试集数据进行调整以保证变量与训练集一一对应，那么就可以设定一个参数 is_test 确定函数到底处理测试集还是训练集。