Mid Term Presentation

Group 4：
沈伯衡 2022200859
傅恒恒 2023200240
李雨佳 2023201772

1. Pipeline 架构设计

- **自动化数据清洗 + 模块化特征提取**
- 多个 Transformer 类让模型可重用、可调参

RawData→Cleaning→Encoding→Scaling→Feature Selection→Linear Models

```
# ---------- 各处理模块 ----------
class LeakRemover(BaseEstimator, TransformerMixin):…
class BuildYearAverager(BaseEstimator, TransformerMixin):…
class HouseLayoutExtractor(BaseEstimator, TransformerMixin):…
class UnitCleaner(BaseEstimator, TransformerMixin):…
class LadderRatioExtractor(BaseEstimator, TransformerMixin):…
class ElevatorFlagger(BaseEstimator, TransformerMixin):…
class DynamicWinsorizer(BaseEstimator, TransformerMixin):…
class DynamicMissingDropper(BaseEstimator, TransformerMixin):…
class ImputerTransformer(BaseEstimator, TransformerMixin):…
class FloorExtractor(BaseEstimator, TransformerMixin):…
class RingEncoder(BaseEstimator, TransformerMixin):…
class CategoricalEncoder(BaseEstimator, TransformerMixin):…
class SelectiveStandardizer(BaseEstimator, TransformerMixin):…
class ColumnPruner(BaseEstimator, TransformerMixin):…
class FinalImputer(BaseEstimator, TransformerMixin):…
```

2. 特征工程处理

- **结构化非结构化结合 + 智能数值化**
- 将文本型特征（如“建筑年代”、“户型”等）进行特殊化处理，并形成“特征组合变量”
- log1p处理价格，纠正价格右偏，分布趋于对称

```
class HouseLayoutExtractor(BaseEstimator, TransformerMixin):
    """提取房屋户型为室/厅/厨/卫"""
    Trae: 解释代码 | 注释代码 | ×
    def __init__(self, col='房屋户型'): self.col = col
    Trae: 解释代码 | 注释代码 | ×
    def fit(self, X, y=None): return self
    Trae: 解释代码 | 注释代码 | ×
    def transform(self, X):
        X = X.copy()
        Trae: 解释代码 | 注释代码 | ×
        def extract_layout(s):
            if pd.isna(s) or str(s).strip()=='':
                return (0,0,0,0)
            s = str(s)
            if '房间' in s:
                rooms = re.findall(r'(\d+)房间', s)
                baths = re.findall(r'(\d+)卫', s)
                return (int(rooms[0]) if rooms else 0, 0, 0, int(baths[0]) if baths else 0)
            rooms = re.findall(r'(\d+)室', s)
            halls = re.findall(r'(\d+)厅', s)
            kitchens = re.findall(r'(\d+)厨', s)
            baths = re.findall(r'(\d+)卫', s)
            return (int(rooms[0]) if rooms else 0,
                    int(halls[0]) if halls else 0,
                    int(kitchens[0]) if kitchens else 0,
                    int(baths[0]) if baths else 0)
        layout_df = pd.DataFrame(X[self.col].apply(extract_layout).tolist(),
                                 columns=['户型_室数','户型_厅数','户型_厨数','户型_卫数'], index=X.index)
        return pd.concat([X.drop(columns=[self.col], errors='ignore'), layout_df], axis=1)
```

```
USE_LOG_TARGET = True  # 对价格取对数使分布更稳定

if USE_LOG_TARGET:
    y_all = np.log1p(y_price)
else:
    y_all = y_price
```

3. 模型验证与调优

- **使用OLS / RidgeCV / LassoCV / ElasticNetCV**

```python
# 2. 房屋朝向评分
if '房屋朝向' in df.columns:# 检查'房屋朝向'列是否存在
    print("  创建朝向评分...")
    orientation_scores = {
        '南': 10, '东南': 9, '西南': 8, '东': 7, '西': 6,
        '东北': 5, '西北': 4, '北': 3, '南北': 9, '东西': 6,
        '东南北': 7, '西南北': 6, '未知': 5
    }
    orientation = df['房屋朝向'].fillna('未知').astype(str)# 缺失值填充为'未知'
    X['朝向评分'] = orientation.map(orientation_scores).fillna(5)# 缺失值填充为5
    X['是否南向'] = orientation.str.contains('南').fillna(0).astype(int)# 缺失值填充为0

# 3. 建筑年代和房龄特征
if '建筑年代' in df.columns:# 检查'建筑年代'列是否存在
    print("  处理建筑年代...")
    year_extracted = df['建筑年代'].astype(str).str.extract(r'(\d{4})').astype(float)# 提取4位数字作为年份
    current_year = 2023
    X['房龄'] = current_year - year_extracted[0]# 计算房龄
    X['房龄'] = X['房龄'].clip(0, 100)# 限制房龄在0到100之间

# 4. 地理坐标特征
if 'lon' in X.columns and 'lat' in X.columns:
    print("  创建地理坐标特征...")
    if is_training:
        center_lon, center_lat = X['lon'].median(), X['lat'].median()# 计算中位数作为市中心坐标
        new_encoders['center_lon'] = center_lon
        new_encoders['center_lat'] = center_lat
    else:
        center_lon, center_lat = encoders['center_lon'], encoders['center_lat']# 使用训练集计算的中位数作为市中心坐标

    X['距市中心距离'] = np.sqrt((X['lon']-center_lon)**2 + (X['lat']-center_lat)**2)# 计算距离
```

特征工程中的部分处理：

1.利用字典定义映射关系，将房屋朝向这一分类特征转换为数值评分

2.从“建筑年代”中提取年份数字：使用正则表达式提取4位数字，计算房龄（当前年份−建筑年份）。对房龄进行裁剪，避免极端值（如负值或过大值）的影响

3.量化地理坐标，计算每个点位到市中心（定义为坐标中位数）的欧氏距离，创建地理坐标特征

处理流程
1．数据验证 → 2．缺失处理 → 3．特征转换 → 4．异常控制 → 5．参数保存

特征处理（半结构化→结构化）

- 统一度量衡：如租期（包含“月/年”单位）等
- One-hot处理：租赁方式、建筑结构、供暖方式等
- 文本数据取字符串长度：房屋优势、核心卖点等
- 调用大语言模型：客户反馈特征【成本较高】

```python
@retry(stop=stop_after_attempt(3), wait=wait_exponential(multiplier=1))
async def async_get_intro(comment, semaphore):
    async with semaphore:  # 控制并发量
        response = await aclient.chat.completions.create(
            model="deepseek-chat",
            messages=[
                {"role": "system", "content": "你是一名精明的房产从业者，熟知评判一处房
                {"role": "user", "content": f"假设将房屋硬件条件良好、配套设施齐全、周
            ],
            stream=False
        )
        return response.choices[0].message.content
```

数据清洗

- 缺失值处理：占比>20%，将剔除该特征；否则中位数填充
- 异常值处理：按3倍z-score规则进行训练集异常值剔除；
  中位数填充，如“总楼层=0”

模型拟合与预测

- 采用OLS、Lasso、Ridge、Elastic-Net、MLP
- 遍历所有交互项与平方项，通过单变量F检验进行特征选择

Metrics	In sample	out of sample	Cross-validation	Kaggle Score
OLS	764748.0128	779055.6001	766108.3478	10.99
LASSO	793809.8988	806826.7524	794901.8004	27.88
Ridge	765197.8763	779670.7695	766579.7736	23.3
ElasticNet	793790.3865	806823.5351	794916.9892	28.19
MLP	111708.8594	/	/	57.13