中山大学计算机学院

人工智能

本科生实验报告

课程名称：Artificial Intelligence

学号	23336179	姓名	马福泉

一、 实验题目

感知机算法

1. 房价预测任务

   data.csv 数据集包含五个属性，共 10000 条数据，其中 longitude 和 latitude 表示房子经纬度，housing_age 表示房子年龄，homeowner_income 表示房主的收入（单位：十万元）， house_price 表示房子的价格。请根据数据集 data.csv 中四个属性 longitude、latitude、housing_age、homeowner_income，利用感知机算法预测房价 house_price，并画出数据可视化图、loss 曲线图。

2. 提示：最后提交的代码只需包含性能最好的实现方法和参数设置. 只需提交一个代码文件, 请不要提交其他文件.本次作业可以使用 numpy 库、matplotlib 库以及 python 标准库.数据集可以在 Github 上下载。

二、 实验内容

1. 算法原理

（1） 数据加载与预处理

① 数据加载 - load_data 函数：使用 Python 的 csv 模块读取文件，跳过表头，将每行数据转换为浮点数，最后返回 NumPy 数组

② 异常值处理 - detect_and_remove_outliers 函数

IQR 方法：基于箱线图原理，计算每个特征的四分位数范围，将超出(Q1-1.5IQR)和(Q3+1.5IQR)的样本视为异常值

Z-score 方法：计算每个特征值的 Z 分数，将绝对 Z 分数大于阈值的样本视为异常值

返回：清洗后的特征数据、目标变量及标识异常值的布尔掩码

③ 数据标准化 - normalize_data 函数

对每个特征计算均值和标准差，然后使用公式(X-mean)/std 进行标准化

（2） 地理特征转换

transform_geo_features 函数

① 实现原理：使用 K-means 聚类将地理位置分为 5 个区域

② 计算每个样本点到各区域中心的距离作为新特征

③ 将得到的 5 个距离特征与原有的房龄、收入特征合并

这段 Python 代码实现了一个用于回归任务（预测房价）的多层感知机 (MLP) 模型，并包含了详细的数据预处理步骤。

（3） MLP 模型构建:

定义了一个 `MLP` 类。

① 初始化:根据指定的网络层级大小 (`layer_sizes`) 初始化权重 (使用 He 初始化) 和偏置。

② 前向传播 (`forward`):实现了混合激活函数策略：根据层的位置选择不同的激活函数（例如，第一隐藏层用 tanh，第二隐藏层用 ReLU，输出层用 linear）。

计算每一层的输出。

③ 后向传播 (`backward`):计算输出层误差。

根据链式法则和激活函数的导数，逐层反向计算梯度。

④ 使用梯度下降法更新权重和偏置。

（4） 模型训练与评估:

定义 MLP 的网络结构（例如，输入层-20 节点隐层-10 节点隐层-输出层）。

实例化 `MLP` 模型，设置学习率和迭代次数。

使用训练集训练模型。

使用测试集评估模型性能，计算标准化 MSE、原始尺度 MSE 和 R^2 (决定系数)。

（5） 可视化部分

数据清理可视化：对比清理前后各特征与房价的关系

地理聚类可视化：展示 K-means 聚类的区域划分

损失曲线可视化：展示模型训练过程中损失的变化

MSE 比较可视化：比较不同参数组合的模型性能

预测结果可视化：展示真实值与预测值的对比，以及特征与房价的关系

特征重要性可视化：展示各特征对预测结果的影响程度

2. 关键代码展示

(1) 数据读取与处理

```python
# 读取数据函数
def load_data(filename):
    data = []
    with open(filename, 'r', encoding='utf-8') as f:
        reader = csv.reader(f)
        next(reader)  # 跳过表头
        for row in reader:
            data.append([float(x) for x in row])  #浮点数
    return np.array(data)  # 转换为NumPy数组并返回

```

```python
# 异常值检测与处理
def detect_and_remove_outliers(X, y, method='iqr', threshold=1.5):
    # 将特征和目标合并为一个数组进行异常值检测，确保同时移除X和y中对应的异常点
    data = np.hstack((X, y.reshape(-1, 1)))
    n_samples, n_features = data.shape
    outliers_mask = np.zeros(n_samples, dtype=bool)  # 初始化异常值掩码为全False
    if method == 'iqr':
        # 使用IQR方法(箱线图法)检测异常值:异常值定义为小于Q1-threshold*IQR或大于Q3+threshold*IQR的值
        for i in range(n_features):
            q1 = np.percentile(data[:, i], 25)
            q3 = np.percentile(data[:, i], 75)
            iqr = q3 - q1
            lower_bound = q1 - threshold * iqr
            upper_bound = q3 + threshold * iqr
            column_outliers = (data[:, i] < lower_bound) | (data[:, i] > upper_bound)
            outliers_mask = outliers_mask | column_outliers
    elif method == 'zscore':
        # 使用Z-score方法检测异常值: Z-score = (x - μ)/σ，异常值定义为|Z-score| > threshold的值
        for i in range(n_features):
            z_scores = np.abs(stats.zscore(data[:, i]))
            column_outliers = z_scores > threshold
            outliers_mask = outliers_mask | column_outliers
    # 返回清洗后的数据和异常值掩码
    return X[~outliers_mask], y[~outliers_mask], outliers_mask

# 数据标准化
def normalize_data(X):
    mean = np.mean(X, axis=0)  # 计算每列(每个特征)的均值
    std = np.std(X, axis=0)    # 计算每列的标准差
    return (X - mean) / std, mean, std  # 返回标准化后的数据及均值、标准差(用于后续逆变换)

```

(2) 经纬度联合处理

```python
def transform_geo_features(X, y=None, cluster_centers=None, n_clusters=5):
    # 提取地理坐标(经度、纬度)
    geo_coords = X[:, :2].copy()

    # 1. 使用K-means进行区域聚类
    if cluster_centers is None:
        # 训练阶段：拟合模型
        kmeans = KMeans(n_clusters=n_clusters, random_state=42)
        kmeans.fit(geo_coords)
        cluster_centers = kmeans.cluster_centers_
    else:
        # 预测阶段：使用已有中心点
        kmeans = KMeans(n_clusters=len(cluster_centers), random_state=42)
        kmeans.cluster_centers_ = cluster_centers

    # 2. 计算到各聚类中心的距离作为特征
    dist_to_clusters = np.zeros((len(X), n_clusters))
    for i in range(n_clusters):
        center = cluster_centers[i]
        # 计算欧几里得距离
        dist_to_clusters[:, i] = np.sqrt(np.sum((geo_coords - center) ** 2, axis=1))

    # 提取房龄和收入特征
    other_features = X[:, 2:].copy()

    # 将距离特征与房龄、收入特征合并
    X_transformed = np.hstack((other_features, dist_to_clusters))

    return X_transformed, cluster_centers
```

(3) 应用激活函数

```python
def forward(self, X):
        activations = [X]  # 第一个元素是网络的输入
        layer_inputs = []  # 存储每层的输入(激活函数前的值)

        # 前向传播
        for i in range(self.num_layers - 1):
            # 计算当前层的输入
            layer_input = np.dot(activations[-1], self.weights[i]) + self.biases[i]
            layer_inputs.append(layer_input)

            # 应用激活函数
            if i == 0:  # 第一隐藏层使用tanh激活函数
                activation = self.tanh(layer_input)
            elif i == self.num_layers - 2:  # 输出层使用线性激活函数
                activation = self.linear(layer_input)
            elif i == 1:  # 第二隐藏层使用ReLU激活函数
                activation = self.relu(layer_input)
            elif i == 2 and self.num_layers > 4:  # 第三隐藏层使用Leaky ReLU
                activation = self.leaky_relu(layer_input)
            elif i == 3 and self.num_layers > 5:  # 第四隐藏层(新增)使用tanh激活函数
                activation = self.tanh(layer_input)
            else:  # 其他隐藏层使用ReLU激活函数
                activation = self.relu(layer_input)
            activations.append(activation)

        return activations, layer_inputs
```

3. 创新点&优化

（1） 数据预处理（代码见上文关键代码展示）

① 多种异常值处理方法的整合：实现了两种常见的异常值检测算法（IQR 和 Z-score）。

② 特征与目标变量的联合异常值处理：任何维度出现异常都会使整个样本被标记，确保数据质量

③ 处理前后对比可视化：直接通过图形对比展示清理前后的数据分布变化：用不同颜色突出显示被识别的异常点

④ 精细的数据标准化设计

（2） 经纬度联合考量（代码见上文关键代码展示）

用 K-means 聚类将地理位置分为 5 个区域

计算每个样本点到各区域中心的距离作为新特征

将得到的 5 个距离特征与原有的房龄、收入特征合并

（3） 不同参数组合的考量与选择

尝试不同的学习率和迭代次数组合

训练多个模型并评估其性能

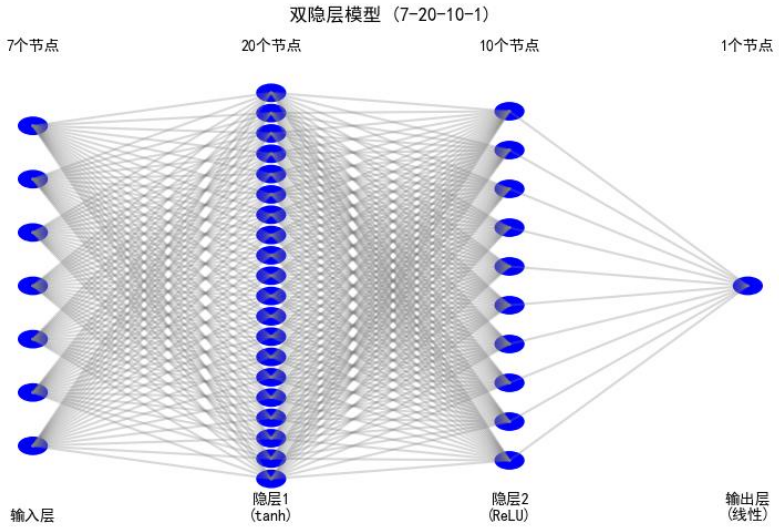
可视化不同参数下的损失曲线

模型评估

（4） 混合使用多种激活函数策略

- 第一隐藏层：使用 tanh 激活函数，输出范围为 [-1, 1]，在 0 附近具有较强梯度
- 中间隐藏层：使用 ReLU 激活函数，解决深层网络的梯度消失问题
- 输出层：不另外使用激活函数，使用线性激活，适合回归任务的任意范围输出。

结合了不同激活函数的优势，充分发挥了 tanh 在输入处理和 ReLU 在中间层特征提取方面的各自优点。



（5） 完整的可视化过程

数据清理可视化：对比清理前后各特征与房价的关系

地理聚类可视化：展示 K-means 聚类的区域划分

损失曲线可视化：展示模型训练过程中损失的变化

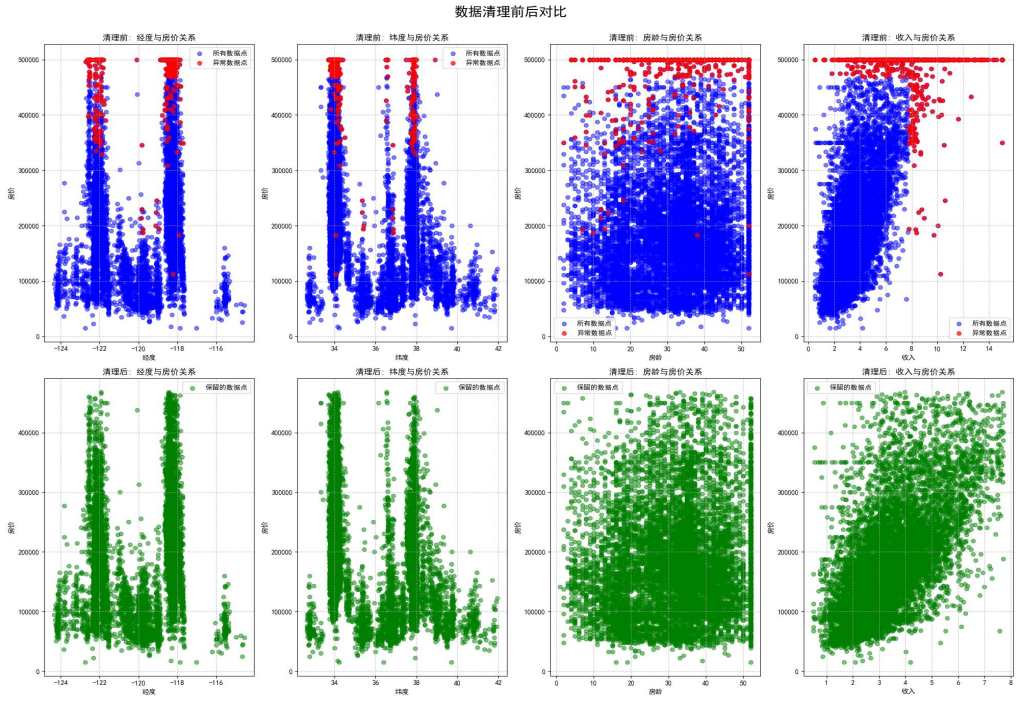
预测结果可视化：展示真实值与预测值的对比，以及特征与房价的关系

特征重要性可视化：展示各特征对预测结果的影响程度

三、 实验结果及分析

1. 实验结果展示示例

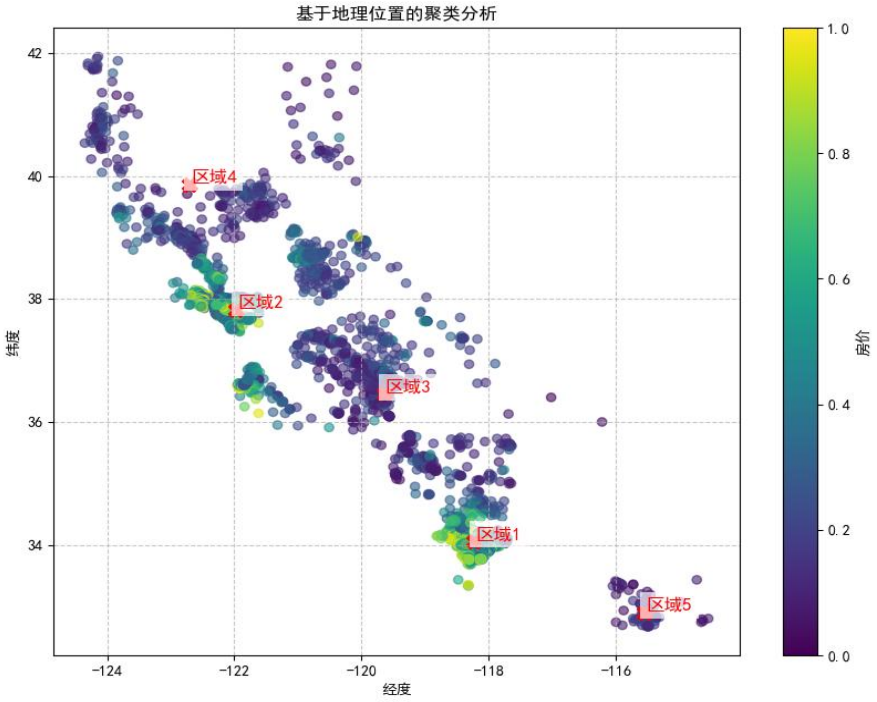
（1） 数据预处理结果（data_cleaning_comparison.png）




```
检测并处理异常值...
原始数据样本数: 10000
处理后数据样本数: 9320
被移除的异常样本数: 680 (6.80%)
```

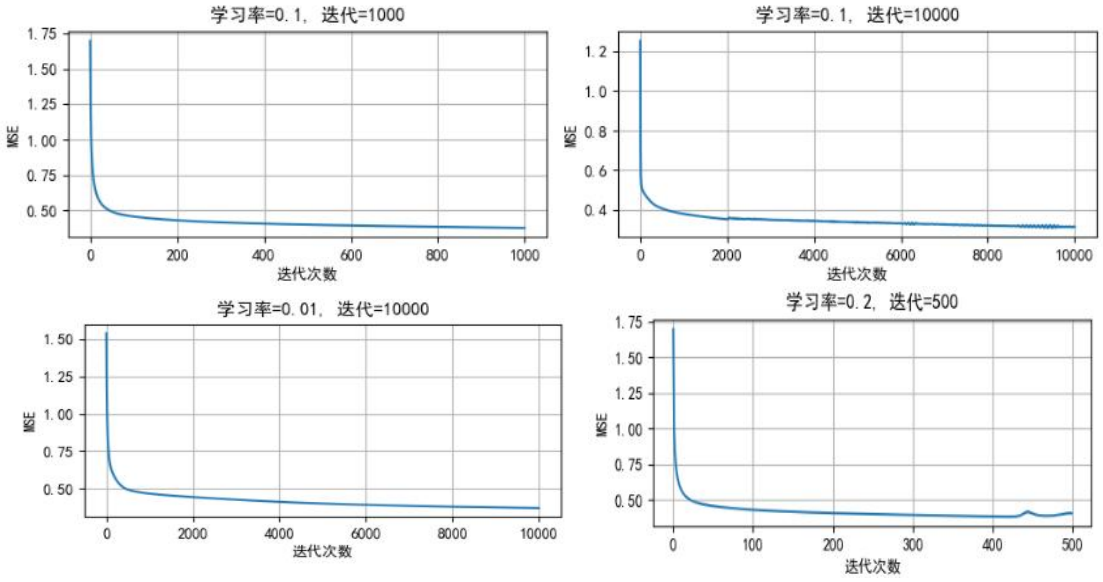
如上图，采用 IQR 方法去除红色数据。

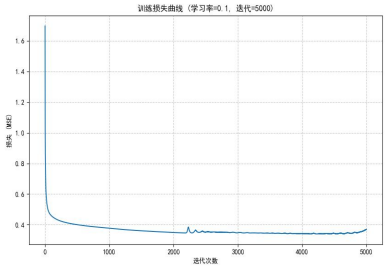
（2） 经纬度处理结果（geo_clustering.png）



如图所示，采用 Kmeans 算法，得到五个区域。

（3） 不同参数组合的比较选择（过程代码已舍去）





选择 lr=0.1,ep=5000 的参数组合,损失值在 1000 次迭代后稳定在 0.4 附近，表明模型已完成主要学习过程,稳定性后期小幅波动（±0.005）属于正常随机梯度下降现象，未出现剧烈震荡.

① 学习率（lr）=0.1：

当学习率为 0.2 时（左列三图），误差曲线初期下降较快，但后期波动明显且收敛平缓，说明学习率过大可能导致模型在最优解附近震荡，难以稳定收敛。

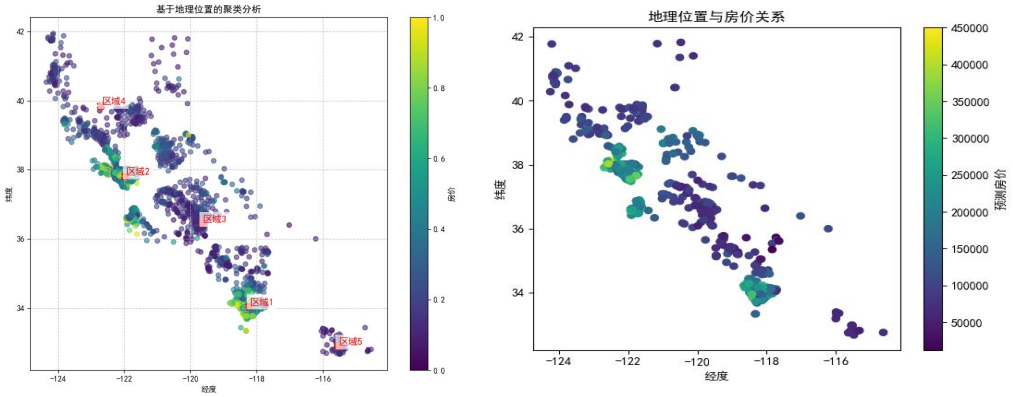
而学习率 0.1 时（右列三图），误差曲线下降更平滑，尤其在迭代 10000 次后趋于稳定，表明较小的学习率能更精准地逼近最优解，避免过冲或震荡。

学习率 0.1 既不会因过大（如 0.2）而跳过最优解，也不会因过小（如 0.01）而需要极高迭代次数，在效率和精度间取得了平衡。

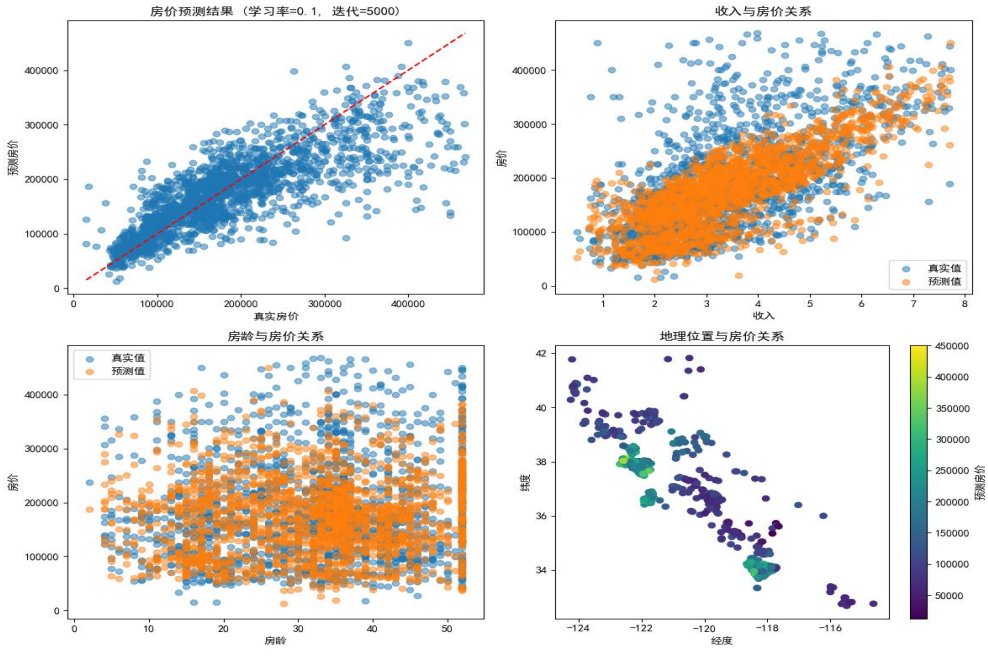
② 迭代次数=5000 的合理性

在 lr=0.1 时，迭代 5000 次的误差曲线（中下图）已接近最低点，继续增加到 10000 次（右下图）误差仅小幅下降但计算成本翻倍，而且出现数据不稳定现象。相比之下，5000 次是保证充分收敛的临界点。

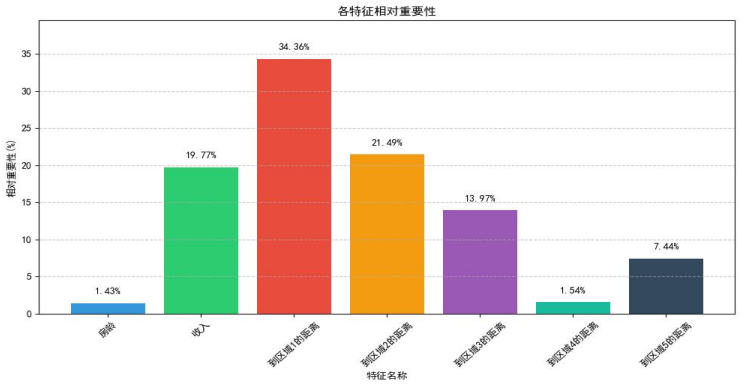
（4） 最佳参数的预测结果



（左图为真是数据，右图为预测数据）



```
MLP模型评估结果:
学习率: 0.1, 迭代: 5000, 标准化MSE: 0.388274
R²(决定系数): 0.6217
```



各变量维度预测结果与真实数据对比如上图，预测结果与实际数据较为吻合，以及可视化的各个特征重要值。

2. 评测指标展示及分析

(1) 模型性能评估：标准化 MSE=0.388274 ，R^2 =0.6217（决定系数）

解释力 ：模型能够解释目标变量（房价）62.17%的方差，剩余约 38%的波动未被捕捉。

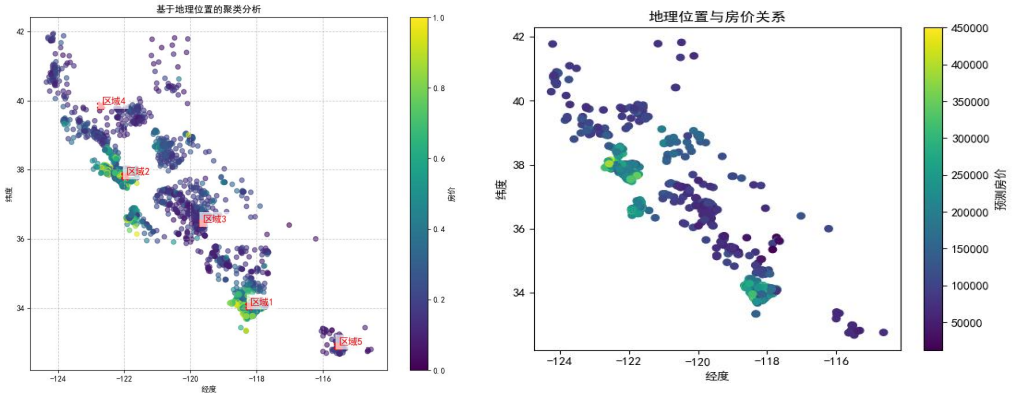
在房价预测问题中，R^2 通常期望达到 0.5~0.8（中等以上），考虑到房价还有其他因素影响，且采用的只是简单的线性预测模型，当前结果合理但仍有提升空间。则当前模型有一定预测价值，但需进一步优化。

(2) 参数合理性分析

学习率：0.1 是中等偏大的学习率，适合快速收敛。

迭代次数：5000 次迭代可能已使模型接近收敛，但需检查训练曲线：训练损失和验证损失均平稳，说明迭代足够；

(3) 变量重要性分析



① 到区域 1 的距离（34.36%）：绝对重要性最高（0.9628），可能反映核心区位价值（如市中心、商业区）。

② 收入（19.77%）：重要性次之（0.5538），符合经济常识（收入高的人群购房能力更强）。

③ 到区域 2，3 的距离（合计 35.46%）：可能与区域 1 形成互补（如次

级商圈或居住区），需分析这些区域的实际功能。

④ 到区域 5 的距离（7.44%）：贡献较低但不可忽略，可能对应郊区或特定设施（如公园）。

⑤ 房龄（1.43%）和到区域 4 的距离（1.54%）：影响微弱，但需验证是否因数据分布或模型缺陷导致低估（如房龄与房价实际为非线性关系）。

（4）后续改进建议

① 升级模型 ：尝试非线性模型（如 XGBoost、随机森林），自动捕捉变量间复杂关系。

② 数据增强

补充特征 ：加入房屋面积、卧室数量、周边设施（地铁、学校）等关键因子。

数据变换 ：对“距离”类变量尝试分箱或对数变换，解决非线性问题。

四、 参考资料

（1）部分 debug，优化建议参考了 ai 大模型的建议。

（2）阅读参考了以下文档：

课程实验指导文档

多层感知机（MLP）简介-CSDN 博客

MLP 神经网络：多层感知机实现波士顿房价预测_mlp 波士顿-CSDN 博客

【人工智能】保姆级波士顿房价预测-CSDN 博客

python 机器学习 波士顿房价预测 详细教程 数据集+源码+结果图+远程部署_波士顿房价数据集-CSDN 博客

使用 Python 和 Numpy 构建神经网络模型——波士顿房价预测案例 - 知乎