机器学习大作业——珠海二手房房价预测实验报告

一、数据爬取以及数据集构建

为了获取珠海二手房的房价信息，我们编写了一个 Python 爬虫程序，用于从链家网上爬取珠海地区二手房的数据，并将数据保存到 originaldata.csv 文件中。程序主要分为以下几个部分：

Get_url(url): 这个函数用于生成每一页的 URL 地址，其中 url 参数是链家网珠海二手房页面的基础 URL。在这个函数中，通过循环生成每一页的 URL，并将其存储在一个列表中返回。代码如下：

```
#获取每一页的url
def Get_url(url):
    all_url=[]
    for i in range(402,800):
        all_url.append(url+'pg'+str(i)+'/') #储存每一个页面的url
    return all_url
```

Get_house_url(all_url, headers): 这个函数用于获取每套房详情信息的 URL。它接收两个参数，all_url 是包含所有页面 URL 的列表，headers 则是 HTTP 请求头，用于模拟浏览器发送请求。在函数中，遍历每一页的 URL，发送 HTTP 请求，解析 HTML 内容，提取每套房的详情页 URL，并调用 Analysis_html() 函数进行详情页信息的解析。代码如下：

```
#获取每套房详情信息的url
def Get_house_url(all_url,headers):
    num=0
    #简单统计页数
    for i in all_url:
        r=requests.get(i,headers=headers)
        html=etree.HTML(r.text)
        url_ls=html.xpath("//ul[@class='sellListContent']/li/a/@href") #获取房子的url
        Analysis_html(url_ls,headers)
        time.sleep(1)
        print("第%s页爬完了"%i)
        num+=1
```

Analysis_html(url_ls, headers): 这个函数用于解析每套房详情页的 HTML 内容，提取房源的各种信息，如小区名、价格、地区、房屋基本属性等。然后调用 Save_data() 函数将这些信息保存到 CSV 文件中。代码如下：

```
#获取每套房的详情信息
def Analysis_html(url_ls,headers):
    for i in url_ls: #num记录爬取成功的索引值
        r=requests.get(i,headers=headers)
        html=etree.HTML(r.text)
        name=(html.xpath("//div[@class='communityName']/a/text()"))[0].split() #获取房名
        money = html.xpath("//span[@class='total']/text()" )# 获取价格
        area = html.xpath("//span[@class='info']/a[1]/text()")  # 获取地区
        data = html.xpath("//div[@class='content']/ul/li/text()")# 获取房子基本属性

        Save_data(name,money,area,data)
```

Save_data(name, money, area, data): 这个函数用于将爬取到的房源信息保存到 CSV 文件中。它接收房源的各种信息作为参数，并将这些信息以列表形式写入 CSV 文件。代码如下：

```
#把爬取的信息存入文件
def Save_data(name, money, area, data):
    result=[name[0]]+money+[area]+data #把详细信息合为一个列表
    with open(r'originaldata.csv','a',encoding='utf_8_sig',newline='')as f:
        wt=csv.writer(f)
        wt.writerow(result)
        print('已写入')
        f.close()
```

在 main 函数中，首先定义了要爬取的链家网珠海二手房页面的基础 URL 和 HTTP 请求头。然后调用 Get_url() 函数生成所有页面的 URL 列表，并调用 Get_house_url() 函数开始爬取数据。代码如下：

```
if __name__=='__main__':
    url='https://zh.lianjia.com/ershoufang/'
    headers={
        "Upgrade-Insecure-Requests":"1",
        "User-Agent":"Mozilla/5.0(Windows NT 10.0;Win64;x64) AppleWebKit/537.36(KHTML,like Gecko)Chrome"
                     "/72.0.3626.121 Safari/537.36"
     }
    all_url=Get_url(url)
#     with open(r'originaldata.csv', 'a', encoding='utf_8_sig', newline='') as f:
#         #首先加入表格头
#         table_label=['小区名', '价格/万', '地区', '0', '房屋户型', '1', '所在楼层', '2', '建筑面积', '3', '户型结构', '4', '套内面积', '5'
# #         table_label=['Community','Price','Region','0','House type','1','Floor','2','Area','3','House structure','4','Inside area','5','
#         wt=csv.writer(f)
#         wt.writerow(table_label)
    Get_house_url(all_url,headers)
#爬完之后记得用notepad将csv文件编码格式手动修改为utf-8，将所有特殊字符删除，将后面的无关列删除
```

originaldata.csv 文件中存储着所有爬取到的数据，格式如下图所示。

小区名	价格/万	地区	0 房屋户型	1 所在楼层	2 建筑面积	3 户型结构	4 套内面积	5 建筑类型	6 房屋朝向	7 建筑结构	8 装修情况	9 梯户比例	10 配备电梯
保利时代	170	['金湾区']	3室2厅1卫2卫	高楼层(共15层)	92.62㎡	平层	74.03㎡	板塔结合	南北	框架结构	精装	两梯四户	有
嘉珠时代广	115	['金湾区']	3室2厅1卫1卫	中楼层(共19层)	97.75㎡	平层	73.52㎡	板塔结合	南	钢混结构	精装	暂无数据	暂无数据
金地格林者	90	['金湾区']	3室2厅1卫2卫	高楼层(共28层)	90.7㎡	平层	68.6㎡	板塔结合	南北	框架结构	毛坯	两梯四户	有
龙光玖誉洋	215	['金湾区']	4室2厅1卫2卫	高楼层(共38层)	107.34㎡	平层	83.98㎡	板塔结合	东南	钢混结构	精装	两梯五户	有
金碧丽江	65	['斗门区']	3室2厅1卫2卫	高楼层(共6层)	116.68㎡	平层	102.42㎡	板楼	南北	钢混结构	其他	暂无数据	无
奥园香海彩	36	['斗门区']	1室1厅1卫1卫	高楼层(共17层)	32.37㎡	平层	22.83㎡	板塔结合	北	钢混结构	精装	三梯二十户	有
翰苑	188	['香洲区']	4室2厅1卫2卫	中楼层(共7层)	129.07㎡	平层	暂无数据	板楼	东南	钢混结构	精装	暂无数据	无
翠湖香山另	348	['高新区']	4室2厅1卫2卫	低楼层(共6层)	138.34㎡	平层	暂无数据	板楼	南	钢混结构	精装	暂无数据	暂无数据
海怡湾畔	315	['高新区']	4室2厅1卫3卫	中楼层(共5层)	205.06㎡	复式	182.18㎡	板塔结合	东南	钢结构	简装	暂无数据	暂无数据
拱运雷禽	115	['香洲区']	3室1厅1卫1卫	低楼层(共6层)	75.28㎡	平层	68.28㎡	暂无数据	东	混合结构	其他	暂无数据	无
春泽名园	93.8	['香洲区']	2室1厅1卫1卫	高楼层(共11层)	46.06㎡	平层	暂无数据	板塔结合	北	钢混结构	精装	一梯十户	有
中珠九悦	490	['香洲区']	4室2厅1卫2卫	高楼层(共26层)	142㎡	平层	暂无数据	塔楼	东南	钢混结构	精装	两梯三户	有
田德花园A	180	['香洲区']	3室2厅1卫2卫	高楼层(共7层)	115.12㎡	复式	103.48㎡	板楼	东	钢混结构	精装	暂无数据	无
石幡园	100	['香洲区']	2室2厅1卫1卫	中楼层(共7层)	73.33㎡	平层	66.83㎡	板楼	南	钢混结构	简装	暂无数据	暂无数据
海湾花园	220	['香洲区']	3室1厅1卫1卫	高楼层(共28层)	112.36㎡	平层	暂无数据	板塔结合	西南	钢混结构	其他	三梯八户	有
安居园北区	110	['香洲区']	2室2厅1卫1卫	高楼层(共7层)	63.65㎡	平层	暂无数据	塔楼	西	钢混结构	精装	暂无数据	无
华发山庄	515	['香洲区']	4室2厅1卫2卫	高楼层(共12层)	168㎡	平层	暂无数据	塔楼	东南	钢混结构	毛坯	一梯两户	有
北园新村一	107	['香洲区']	2室2厅1卫1卫	中楼层(共6层)	65㎡	平层	暂无数据	塔楼	东南	混合结构	精装	暂无数据	无
春晖花园	243	['香洲区']	3室2厅1卫1卫	低楼层(共7层)	85.5㎡	暂无数据	暂无数据	塔楼	南	钢混结构	精装	暂无数据	无
华发峰尚	85	['斗门区']	3室2厅1卫2卫	中楼层(共34层)	90.28㎡	平层	73.28㎡	板楼	南	钢混结构	毛坯	两梯六户	有

这个数据集中包含很多的定性变量，例如所在楼层、户型结构、建筑类型等等，为了将定性变量转换为数值表示，我们采用虚拟变量的方式对这些定性变量进行转换；同时这个数据集中很多变量需要删除或分割，因此我们对这个数据进行了一系列处理，处理主要分成以下几个部分：

“户型结构”的转换：“户型结构”分为平层、复式、错层三种情况，因此我们在数据集中新增两个虚拟变量“复式”和“错层”，当样本的户型结构为复式时，“复式”变量值为 1；当样本的户型结构为错层时，“错层”变量值为 1；当样本的户型结构为平层时，“复式”和“错层”变量值均为 0。

“所在楼层”的转换：“所在楼层”分为高楼层、中楼层、低楼层三种情况，因此我们在数据集中新增两个虚拟变量“高楼层”和“中楼层”，当样本的所在楼层为高楼层时，“高楼层”变量值为 1；当样本的所在楼层为中楼层时，“中楼层”变量值为 1；当样本的所在楼层为低楼层时，“高楼层”和“中楼层”变量值均为 0。

“建筑类型”的转换：“建筑类型”分为板楼、塔楼、板塔结合三种情况，因此我们在数据集中新增两个虚拟变量“板楼”和“塔楼”，当样本的建筑类型为板楼时，“板楼”变量值为 1；当样本的建筑类型为塔楼时，“塔楼”变量值为 1；当样本的建筑类型为板塔结合时，“板楼”和“塔楼”变量值均为 0。

“装修情况”的转换：“装修情况”分为毛坯、简装、精装、其它四种情况，因此我们在数据集中新增三个虚拟变量“毛坯”、“简装”和“精装”，当样本的装修情况为毛坯时，“毛坯”变量值为 1；当样本的装修情况为简装时，“简装”变量值为 1；当样本的装修情况为精装时，“精装”变量值为 1；当样本的装修情况为其它时，“毛坯”、“简装”和“精装”变量值均为 0。

“房屋户型”的转换：房屋户型包含了 4 个信息，分别是卧室数、客厅数、厨房数和卫生间数，因此我们在数据集中新增四个变量分别记录卧室数、客厅数、厨房数和卫生间数。

“配备电梯”的转换：将“有”变为 1，“无”变为 0。

“地区”的转换：我们从 58 同城网站上找到了 2024 年 6 月珠海各个区域二手房的每平方米价格均值，我们将这些价格存储到一个新增的变量“地区平均房价”中，用于代替“地区”。

随后，我们将一切包含不确定信息的样本删除，并将数据集中不需要的变量删除，将修改后的数据保存到 updateeddata.csv 文件中。

updateeddata.csv 文件中存储着所有更新后的数据，格式如下图所示。

建筑面积	套内面积	配备电梯	复式	错层	高楼层	中楼层	板楼	塔楼	毛坯	简装	精装	卧室数	客厅数	厨房数	卫生间数	地区平均房价格/万
92.62	74.03	1	0	0	1	0	0	0	0	0	1	3	2	1	2	1.2652	170
90.7	68.6	1	0	0	1	0	0	0	1	0	0	3	2	1	2	1.2652	90
107.34	83.98	1	0	0	1	0	0	0	0	0	1	4	2	1	2	1.2652	215
116.68	102.42	0	0	0	1	0	1	0	0	0	0	3	2	1	2	0.9573	65
32.37	22.83	1	0	0	1	0	0	0	0	0	1	1	1	1	1	0.9573	36
115.12	103.48	0	1	0	1	0	1	0	0	0	1	3	2	1	2	2.3838	180
90.28	73.28	1	0	0	0	1	1	0	1	0	0	3	2	1	2	0.9573	85
76.87	58.75	1	0	0	0	1	0	0	0	0	1	2	2	1	1	1.534	200
107.34	83.98	1	0	0	1	0	0	0	0	0	1	4	2	1	2	1.2652	215
32.37	22.83	1	0	0	1	0	0	0	0	0	1	1	1	1	1	0.9573	36

至此，数据集构建完毕。

二、随机森林模型构建与评估

我们在尝试了多种回归算法后，决定采用随机森林回归的算法来预测房价。首先我们将

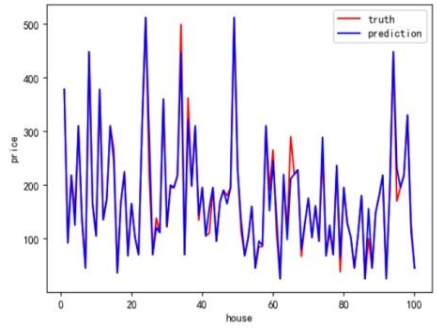
数据集划分为训练集和测试集，其中测试集占比 25%。随后，我们创建并训练了随机森林模型。为了评估模型的效果，我们计算了模型在测试集以及整个数据集上的均方误差（MSE）、绝对平均误差（MAE）以及决定系数（R^2 Score）。代码和输出结果如下图所示。

```
regressor = RandomForestRegressor(n_estimators=200, random_state=0)
regressor.fit(x_train, y_train)

# 在测试集上进行预测
y_pred = regressor.predict(x_test)
# 计算模型在测试集上的均方误差（Mean Squared Error）
mse = metrics.mean_squared_error(y_test, y_pred)
print("模型在测试集上的均方误差:", mse)
# 计算模型在测试集上的绝对平均误差（Mean Absolute Error）
mae = metrics.mean_absolute_error(y_test, y_pred)
print("模型在测试集上的绝对平均误差:", mae)
# 计算模型在测试集上的决定系数（R^2 Score）
r2 = metrics.r2_score(y_test, y_pred)
print("模型在测试集上的决定系数:", r2)

# 在整个数据集上进行预测
y_allpred = regressor.predict(x)
# 计算模型在整个数据集上的均方误差（Mean Squared Error）
mse = metrics.mean_squared_error(y, y_allpred)
print("模型在整个数据集上的均方误差:", mse)
# 计算模型在整个数据集上的绝对平均误差（Mean Absolute Error）
mae = metrics.mean_absolute_error(y, y_allpred)
print("模型在整个数据集上的绝对平均误差:", mae)
# 计算模型在整个数据集上的决定系数（R^2 Score）
r2 = metrics.r2_score(y, y_allpred)
print("模型在整个数据集上的决定系数:", r2)
```

```
模型在测试集上的均方误差：478.4930418369557
模型在测试集上的绝对平均误差：7.575672710367273
模型在测试集上的决定系数：0.9619026098834969
模型在整个数据集上的均方误差：192.32339887511935
模型在整个数据集上的绝对平均误差：4.218579945917655
模型在整个数据集上的决定系数：0.9846554784607627
```

为了更直观的显示模型的预测结果与实际结果的差距，我们根据测试集的前 100 个样

本的真实价格和模型预测价格绘制了折线图，如下图所示。



三、随机森林模型优化

为了提高模型的性能，我们选择对模型的超参数进行优化。首先，将数据集划分为训练集和测试集，其中测试集占比 25%。然后，我们选择了对 n_estimators（决策树个数）、max_depth（决策树最大深度）、min_samples_split（最小样本分离数）这三个超参数进行优化（原本还有 max_features、min_samples_leaf 这两个参数，我们跑了多次程序后发现这两个参数取默认值时效果就是最好的）。

首先，我们对随机森林模型超参数各自的范围加以确定，其中 n_estimators_range 的取值范围是[50, 2000]这个范围内的 50 的倍数；max_depth 的取值范围是[10, 300]这个范围内的 10 的倍数；min_samples_split 的取值范围是[2, 5, 10]。代码如下：

```
from pprint import pprint

n_estimators_range=[int(x) for x in np.linspace(start=50,stop=2000,num=40)]
max_depth_range=[int(x) for x in np.linspace(10,300,num=30)]
# max_depth_range.append(None)
min_samples_split_range=[2,5,10]

random_forest_hp_range={'n_estimators':n_estimators_range,#决策树个数
                        'max_depth':max_depth_range,#决策树最大深度
                        'min_samples_split':min_samples_split_range,#最小分离样本数
                        }
```

然后，我们利用 RandomizedSearchCV 的功能，采用超参数随机匹配择优的方式，获取其所得到的最优超参数匹配组合 best_hp_now。代码如下：

```
random_forest_model_test_base=RandomForestRegressor()
random_forest_model_test_random=RandomizedSearchCV(estimator=random_forest_model_test_base,
                                                   param_distributions=random_forest_hp_range,
                                                   n_iter=100,#随机搭配的超参数组合次数
                                                   n_jobs=-1,
                                                   cv=3,#交叉验证的折数
                                                   verbose=1,
                                                   random_state=42
                                                   )
random_forest_model_test_random.fit(x_train,y_train)

best_hp_now=random_forest_model_test_random.best_params_
```

接着，由于超参数随机匹配择优得到的结果可能并不是全局最优的，而只是一个大概的范围，我们需要在 best_hp_now 的临近范围内选取几个数值，并通过 GridSearchCV 对每一种匹配都遍历，从而选出比较好的超参数匹配组合。（不直接进行遍历匹配择优是因为这样非常耗费时间）代码如下：

```
random_forest_hp_range_2={'n_estimators':[1330,1340,1350,1360,1370],
                          'max_depth':[200,210,220,230,240],
                          'min_samples_split':[2,3]
                          }
random_forest_model_test_2_base=RandomForestRegressor()
random_forest_model_test_2_random=GridSearchCV(estimator=random_forest_model_test_2_base,
                                               param_grid=random_forest_hp_range_2,
                                               cv=3,
                                               verbose=1,
                                               n_jobs=-1)
random_forest_model_test_2_random.fit(x_train,y_train)

best_hp_now_2=random_forest_model_test_2_random.best_params_
```

最后，获得了新的超参数匹配组合后，我们根据这个匹配组合创建并训练了新的随机森林模型。为了评估模型的效果，我们计算了模型在测试集以及整个数据集上的均方误差（MSE）、绝对平均误差（MAE）以及决定系数（R^2 Score）。代码和输出结果如下图所示。

```
random_forest_model_test_2_random = RandomForestRegressor(max_depth=220, min_samples_split=2, n_estimators=1350)
random_forest_model_test_2_random.fit(x_train,y_train)

# 在测试集上进行预测
y_pred = random_forest_model_test_2_random.predict(x_test)
# 计算模型在测试集上的均方误差（Mean Squared Error）
mse = metrics.mean_squared_error(y_test, y_pred)
print("模型在测试集上的均方误差:", mse)
# 计算模型在测试集上的绝对平均误差（Mean Absolute Error）
mae = metrics.mean_absolute_error(y_test, y_pred)
print("模型在测试集上的绝对平均误差:", mae)
# 计算模型在测试集上的决定系数（R^2 Score）
r2 = metrics.r2_score(y_test, y_pred)
print("模型在测试集上的决定系数:", r2)

# 在整个数据集上进行预测
y_allpred = random_forest_model_test_2_random.predict(x)
# 计算模型在整个数据集上的均方误差（Mean Squared Error）
mse = metrics.mean_squared_error(y, y_allpred)
print("模型在整个数据集上的均方误差:", mse)
# 计算模型在整个数据集上的绝对平均误差（Mean Absolute Error）
mae = metrics.mean_absolute_error(y, y_allpred)
print("模型在整个数据集上的绝对平均误差:", mae)
# 计算模型在整个数据集上的决定系数（R^2 Score）
r2 = metrics.r2_score(y, y_allpred)
print("模型在整个数据集上的决定系数:", r2)

index = list(range(1, 101))
plt.xlabel('house')
plt.ylabel('price')
plt.plot(index, y_test[0:100], color='r',label='truth')
plt.plot(index, y_pred[0:100], color='b',label='prediction')
plt.legend()
plt.show()
```

```
模型在测试集上的均方误差: 448.3774144325129
模型在测试集上的绝对平均误差: 7.442564337000879
模型在测试集上的决定系数: 0.9643004019212361
模型在整个数据集上的均方误差: 183.21236192265303
模型在整个数据集上的绝对平均误差: 4.132638649815203
模型在整个数据集上的决定系数: 0.9853824025042208
```

可以看出，模型的性能相较于之前有了一定的进步。

为了更直观的显示模型的预测结果与实际结果的差距，我们根据测试集的前 100 个样本的真实价格和模型预测价格绘制了折线图，如下图所示。