武汉大学计算机学院

本科生课程设计报告

人工智能引论之机器学习预测房价模型设计与实现

专 业 名 称 ：CS

课 程 名 称 ：人工智能引论

指 导 教 师 一：AY

学 生 学 号 ：

学 生 姓 名 ：JaeHua

二〇二三年十一月

郑 重 声 明

本人呈交的设计报告，是在指导老师的指导下，独立进行实验工作所取得的成果，所有数据、图片资料真实可靠。尽我所知，除文中已经注明引用的内容外，本设计报告不包含他人享有著作权的内容。对本设计报告做出贡献的其他个人和集体，均已在文中以明确的方式标明。本设计报告的知识产权归属于培养单位。

本人签名：JaeHua 日期：2023 年 11 月 23 日星期四

摘要

实验目的:研究机器学习算法在房价预测中的应用,设计和实现一个能够高准确度预测房屋价格的模型。
实验设计与内容:
(1)收集房屋特征和对应价格作为训练和测试数据集
(2)预处理数据,处理缺失值和异常值
(3)提取并选择影响房价的主要特征
(4)比较和选择 DecisionTreeRegressor, LinearRegression, GradientBoosting, SVR 算法
(5)训练和测试不同算法模型,选择误差最小模型
实验结论:
通过实验, GradientBoosting 模型在测试集上预测准确度最高,叫较解决了房价预测任务。

关键词：机器学习；线性回归；决策树回归;k 折验证法；支持向量机

目录

1.实验背景与目的

1.1 实验背景

房地产行业是人们日常生活中一个重要且与金钱息息相关的领域。随着经济的持续发展,房屋销售价格也在不断变化。如何更精准地预测房屋价格对房产中介公司和购房者都很重要。

然而,房价受各种因素影响,如房屋面积、户型、装修情况等结构属性,以及所在小区、交通便利程度等环境属性,很难根据个别因素准确预测价格。传统的人工预测方法难以考虑所有影响因素的复杂关系。

与此同时,随着大数据和机器学习技术的发展,通过大量历史交易数据练模型来实现自动化预测已成为可能。国外一些公司已经成功地采用机器习算法进行房价预测。

本实验考虑的主要房屋特征包括:RM - 每房屋平均房间数(rooms per house);LSTAT - 居民地位降低比例(percentage of lower status of the population);PTRATIO - 学生与教师比例(pupil-teacher ratio by town)这三个变量能较好反映房屋质量和所在小区环境质量。另外,实验中的目标变量是:MEDV - 中值房屋价格(median value of owner-occupied homes)

1.2 实验目的

本实验主要研究 RM、LSTAT、PTRATIO 这三个特征变量与房屋价格 MEDV 之间的关系,探索这三个变量在何种程度上可以影响和预测房屋价格,为后续模型建立奠定基础。

同时，比较和选择 DecisionTreeRegressor, LinearRegression, GradientBoosting, SVM 等机器学习算法在房价预测任务中的表现;使用不同算法训练出各种预测模型,通过测试集评估不同模型的预测准确度,选择误差最小的模型来实现房价的自动预测。

总体来说,本实验的目的是设计和实现一个能够高精度预测房屋价格的机器

学习模型。通过算法的选择、模型训练与评估,寻找到在给定数据集下效果最好的预测算法,以实现房价自动化预测的目的。

2.算法介绍

2.1LinearRegression 算法

Linear Regression（线性回归）是一种经典的回归分析算法，用于建立自变量（输入特征）与因变量（输出目标）之间的线性关系模型。它假设自变量和因变量之间存在线性关系，并试图找到最佳拟合的直线（或超平面），以最小化实际观测值与模型预测值之间的残差平方和。

在简单线性回归中，只有一个自变量和一个因变量。模型表示为：
y = b0 + b1*x
其中，y 是因变量，x 是自变量，b0 和 b1 是回归系数（截距和斜率），代表了直线的位置和斜率。

在多元线性回归中，有多个自变量和一个因变量。模型表示为：
y = b0 + b1*x1 + b2*x2 + ... + bn*xn
其中，y 是因变量，x1, x2, ..., xn 是多个自变量，b0, b1, b2, ..., bn 是回归系数。

线性回归的关键任务是通过最小化残差平方和来估计回归系数。常用的方法是最小二乘法，它通过最小化实际观测值与模型预测值之间的差异来确定最佳拟合直线。线性回归在实际应用中广泛使用，特别是在预测和趋势分析方面。它可以用于连续的数值预测问题，并提供了对变量之间关系的解释和理解。然而，线性回归也有一些限制，例如对非线性关系的建模能力较弱。

DecisionTreeRegressor 算法

DecisionTreeRegressor（决策树回归）是一种基于决策树的回归分析算法，用于建立自变量（输入特征）与因变量（输出目标）之间的非线性关系模型。与线性回归不同，决策树回归可以捕捉到自变量和因变量之间的非线性关系和交互作用。

决策树是一种树状的模型结构，由节点（Node）和边（Edge）组成。每个节点表示一个特征或属性，边表示该特征的取值。决策树通过一系列的判断条件来将输入样本逐步分配到不同的叶节点（Leaf），每个叶节点代表一个输出值（在回归问题中通常是一个连续值）。

2.3 GradientBoostingRegressor 算法

GradientBoostingRegressor（梯度提升回归）是一种集成学习方法，通过将多个弱回归模型（通常是决策树）进行加权组合来建立一个更强大的回归模型。它是一种基于提升算法的回归模型。

梯度提升回归的基本思想是迭代地训练一系列的弱回归模型，每个模型都试图修正前一个模型的预测误差。在每次迭代中，新的模型被拟合到前一个模型的残差上，然后将前一个模型的预测值与新模型的预测值相加，得到更新后的预测值。通过迭代的方式不断减少残差，最终得到一个累加的回归模型。

2.4SVM 算法

支持向量机（Support Vector Machine，SVM）是一种常用的监督学习算法，主要用于分类和回归问题。其基本思想是通过在特征空间中找到一个最优的超平面，将不同类别的样本分开。SVM 的关键思想是将非线性问题转化为高维特征空间中的线性问题。通过使用核函数，可以将原始特征映射到一个更高维度的特征空间，使得原本线性不可分的问题在新的特征空间中线性可分。

3.实验设计

1. 搭建环境

本实验我采用的是 Windows 操作系统下的 jupyter 来进行编写代码，采用 python3.11.0 版本

2. 导入必需库

在实验开始时,需要导入所有需要使用的库、包或模块。这包括数值计算库 NumPy,数据处理库 Pandas,模型构建和评估需要的机器学习库 Scikit-learn 等。正确导入必需的库是开展实验的前提。

3. 读取数据并处理

(1)读取数据。使用 Pandas 读取保存数据的文件,例如本实验的 csv 文件。

(2)数据探索。对读入的数据进行观察,查看数据类型、特征分布、缺失值情况等,找出数据问题。

(3)数据预处理。对特征进行编码、标准化处理,去除出局值和空值。此外,把数据随机分为训练和测试数据集。

(4)特征工程。对原始特征进行转换,增加有效特征。

有些部分由于本实验数据很干净没有用到。以上步骤目的是为后续建模工作做好准备。

4. 构建不同模型

使用 Scikit-learn 等工具构建不同类型的回归模型,本实验构造线性回归、决策树回归、支持向量机等。并进行模型训练。

5. 模型评估

在测试数据集上评估不同模型的表现,主要指标为预测精度等。

6. 交叉验证
使用交叉验证的方式进一步评估模型稳定性,避免过拟合。

4.实验内容及代码分析

4.1 导入所需要的库

首先导入本实验必须要的库。本实验导入的主要库有:
1.numpy:用于数值计算,如生成随机数,线性代数运算等
2.pandas:用于读取和处理数据,如读取 csv 文件,数据清洗,探索等
3.seaborn 和 matplotlib:用于数据可视化,绘制各种图表观察数据
4.sklearn:核心机器学习库
5.linear_model:实现线性模型如 LinearRegression
6.preprocessing:数据标准化,标签编码等预处理
7.model_selection:模型选择与评估,如数据划分,交叉验证
8.metrics:评估模型性能,如均方误差
9.tree:决策树回归 DecisionTreeRegressor
10.ensemble:梯度提升算法 GradientBoostingRegressor
11.svm:支持向量机回归 SVR
12.scipy:统计和机器学习的科学计算包,提供箱形变换,正态分布等函数
13.warnings:控制警告信息输出
主要是为后续的数据探索,特征工程,模型训练,评估和选择提供依赖

4.2 数据探索

在如下读取.csv 文件的数据之后，我们进行对数据的初步探索。

```
#读取数据
df = pd.read_csv("housing.csv")
#拷贝一份数据，以免原有数据被破坏
df_copy = df.copy()
```

首先观察数据后五行，看看是什么状况，有没有正确的读入，以及大致确定数据类型。

```
#查看数据
#查看后五行
df.tail()
```

	RM	LSTAT	PTRATIO	MEDV
484	6.59	9.67	21.00	470400.00
485	6.12	9.08	21.00	432600.00
486	6.98	5.64	21.00	501900.00
487	6.79	6.48	21.00	462000.00
488	6.03	7.88	21.00	249900.00

图表 1 数据后五行

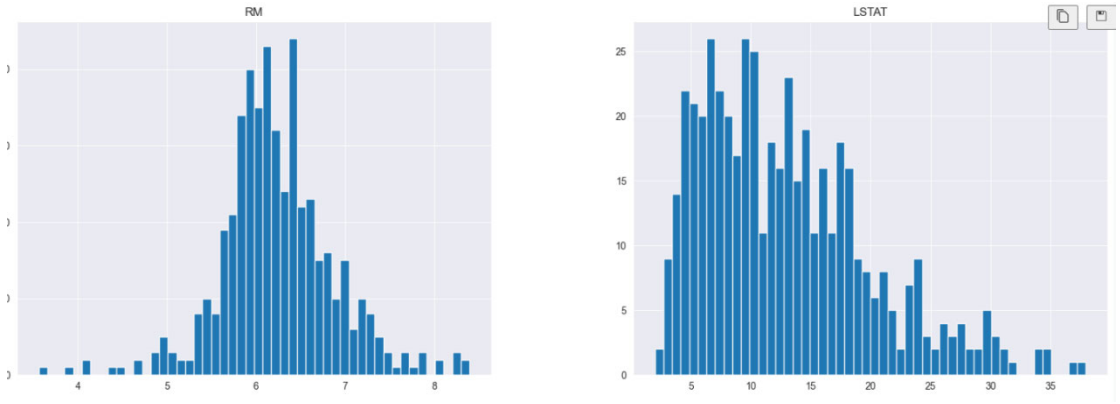
进行简单查看后，我们通过查看数据集的属性描述，看看是否有异常类型的数据，或者有空缺的数据。

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 489 entries, 0 to 488
Data columns (total 4 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   RM       489 non-null    float64
 1   LSTAT    489 non-null    float64
 2   PTRATIO  489 non-null    float64
 3   MEDV     489 non-null    float64
dtypes: float64(4)
memory usage: 15.4 KB
```

图表 2 数据集属性描述

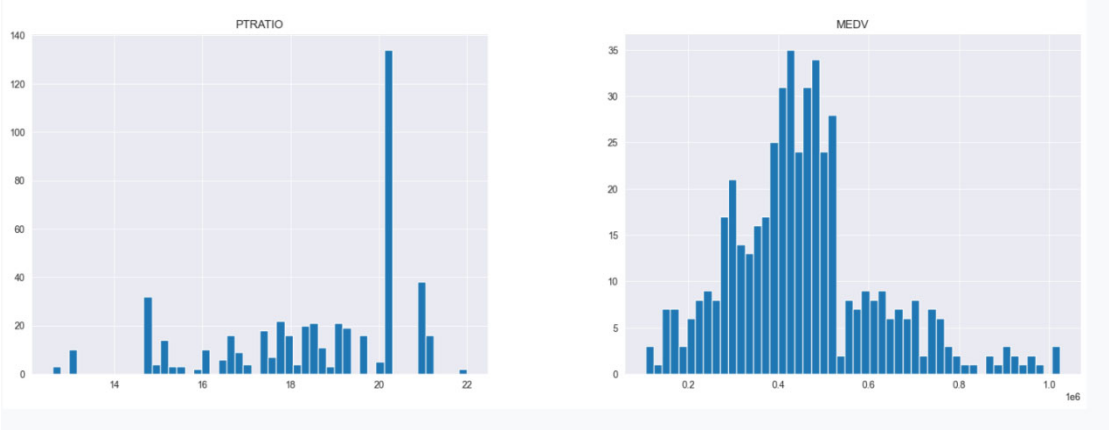
通过查看后发现数据集很干净，不用做另外的处理。进一步我们来看看各属性的分布图案，这个对于我们来建立模型有很直观的帮助。

```
#查看各属性的各自的分布图案
df.hist(bins=50, figsize=(20,15))#bins 代表直方图分区数量，figsize 代表图像的宽高
```



图表 3RM&&LSTAT 的分布

由图看出 RM 的分布近似于正态分布，而 LSTAT 的分布集中于小于 20，大于 20 的相对而言分布很少。



图表 4PTRATION&&MEDV **的分布**

由图可以看出 PTRATIO 的分布相对而言比较均匀，其中有一个类似于异常值的情况，远高于其他的值，MEDV 也就是目标变量的分布近似于正态分布，是比较好的数据分布情况。

4.3 数据集的划分

数据集的划分我采用了留出法，适用于小数据的划分方法。

```
#采用留出法划分数据集
train_data, test_data = train_test_split(df,train_size =
0.8,test_size=0.2, random_state=42)
```

首先运用留出法的步骤是确定划分比例：首先需要确定训练集和测试集所占的比例。常见的做法是将数据集按照一定比例划分，例如将数据集划分为训练集和测试集的比例为 7:3 或 8:2。

随机划分：根据确定的比例，将原始数据集随机划分为训练集和测试集。确保划分过程中的随机性，以避免引入偏差。

训练集与测试集的用途：训练集用于模型的训练和参数调整，而测试集则用于评估训练得到的模型的性能。测试集在模型训练过程中并不参与，只在最后进行模型评估时使用。

上面代码中的 train_test_split 函数是一个用于划分数据集的方法，它接受多个参数。其中，df 是要划分的原始数据集，train_size 表示训练集所占的比例，test_size 表示测试集所占的比例，random_state 用于设置随机数种子，以确保每次运行代码时得到相同的划分结果。通过调用 train_test_split(df, train_size=0.8, test_size=0.2, random_state=42)，将原始数据集 df 按照 80%的比例划分为训练集，20%的比例划分为测试集。

划分完成后，返回的结果是两个数据集对象，分别赋值给 train_data 和 test_data。train_data 是划分后的训练集，test_data 是划分后的测试集。

4.4 分析训练集特征间关系

```
#房价分布，s-蓝色-PTRATION，c-颜色-价格-（蓝-红）
train_data.plot(kind = "scatter", x = "RM", y = "LSTAT", alpha = 0.4,
             s = train_data["PTRATIO"]*3, label = "PTRATIO", figsize =
(10,7),
             c = "MEDV", cmap = plt.get_cmap("jet"), colorbar = True,
            )
plt.legend()
```

这样一 RM 位横坐标，LSTAT 为纵坐标，PTRATIO 作为点的大小，颜色的深浅代表的不同 MEDV 的分布。

图表 5 不同特征值与目标变量的分布

4.5 相关系数计算与不同变量之间分布关系

通过对相关系数的计算，初步得出不同变量与目标变量 MEDV 的相关关系。

```
#计算每对属性的相关系数
corr_matrix = train_data.corr()
#每个属性与房价中位数的相关系数
corr_matrix["MEDV"].sort_values(ascending = False)
```

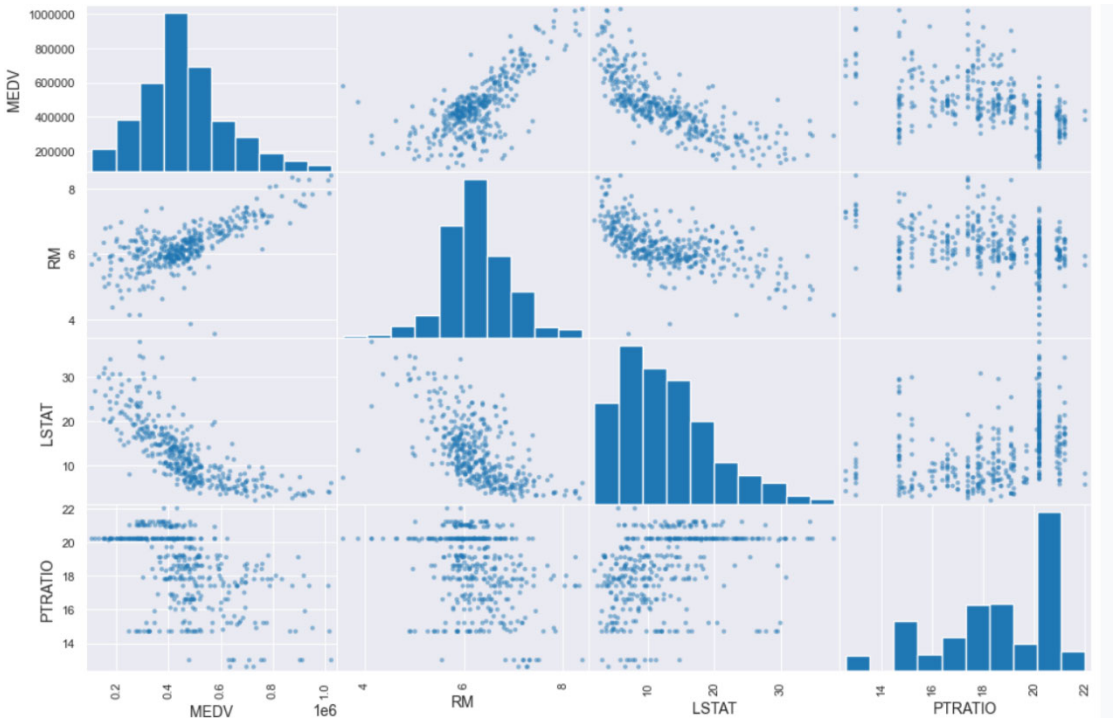
```
MEDV       1.00
RM         0.71
PTRATIO   -0.53
LSTAT     -0.76
Name: MEDV, dtype: float64
```

图表 6 相关系数

由表得知 RM 与 MEDV 有较强的正相关，而 LSTAT 与 MEDV 有较强的负相关，PTRATIO 与 MEDV 的负相关不是那么强。

进一步，我们查看每队变量的相关分布情况。

```
#查看变量因变量两俩之间的关系
attributes = ["MEDV", "RM", "LSTAT", "PTRATIO"]
scatter_matrix(train_data[attributes], figsize = (12,8))
```



图表 7 变量之间分布情况

由表可以看出从图中看出 RM、LSTAT 与 MEDV 相关性比较强，与之前的计算相近。

4.6 LinearRegression 模型建立

通过调用 LinearRegression 库进行线性回归建模。

```
# 模型实例化
le = LinearRegression()
# 拟合过程，梯度下降法
le.fit(housing, housing_labels)
# 得到回归系数
```

```
coef1 = le.coef_  # 3 个回归系数
```

通过这个处理得到回归系数为[87322.20361861 -10620.63731522 -19324.4102965]

4.7 对测试集进行预测

在通过线性回归确定系数之后，我们丢之前随机划分出的测试集进行预测，并计算预测得分情况。

```
#对测试集进行测试
predict1 = le.predict(housing_test)
```

注意，predict 方法根据模型学习到的参数和规则，对输入数据进行预测，并返回预测结果。预测结果的具体形式取决于模型的类型和任务类型。在调用 predict 方法之前，必须首先对模型进行训练，以便模型能够学习到输入数据的模式和规律。训练过程通常包括输入数据的特征提取、模型参数的优化等步骤。只有在模型经过训练后，才能调用 predict 方法进行预测

4.8 预测结果得分计算

根据预测的结果，我们可以得到预测的评分结果。我们用 RMSE 来评判。
预测得分 RMSE（Root Mean Squared Error，均方根误差）是用于评估回归模型预测性能的常用指标之一。它衡量了预测值与真实值之间的平均差异的平方。RMSE 的计算方法是将模型对每个样本的预测值与对应的真实值之差的平方进行求和，然后再除以样本数量，得到平均值。数学公式表示为：MSE = (1/n) * Σ(y_pred - y_true)^2。RMSE 在其基础上取平方根。其中，MSE 是均方误差，n 是样本数量，y_pred 是模型的预测值，y_true 是对应的真实值。
RMSE 衡量了预测值与真实值之间的平均差异。因此，RMSE 越小表示模型的预测结果与真实值的拟合程度越好。

```
#得分显示
#MSE （Mean Squared Error）是指预测值与实际值的平方差之和，公式为： MSE = Σ(y_hat - y)^2 / n， 其中 n 是样本总数。
# 得分
print("Score: ", le.score(housing_test, housing_test_labels))
print("RSME: ", np.sqrt(mean_squared_error(housing_test_labels, predict1)))
```

5.实验结果分析

5.1 得分情况

上述得分代码的计算结果是 Score： 0.6910934003098509RSME： 82395.54332162568。给定的线性回归模型的得分为 0.6910934003098509。得分通常是指模型在测试数据上的拟合程度，取值范围为 0 到 1。得分越接近 1，表示模型对数据的拟合程度越好。在这种情况下，模型的得分为 0.6910934003098509，说明模型对数据的拟合程度一般，还有改进的空间。对于 RMSE 看着很大，其实这个是未归一化后的结果，归一化后 RMSE 为 0.4988。在归一化后的情况下，RMSE 值为 0.4988 表示模型的预测误差相对较小，即模型的预测结果与真实值之间的差异较小。

5.2 K 折验证法

```
#K 折验证法
#评分函数
def display_scores(scores):
    print("Scores:")
    print(pd.Series(scores).mean())
#线性模型的评分：
lin_scores = cross_val_score(Le,X, y,scoring =
"neg_mean_squared_error", cv = 10)
lin_rmse_scores = np.sqrt(-lin_scores)
print("LinearRegression")
display_scores(lin_rmse_scores)
```

K 折验证法后得分更加低了，仅仅为 0.5589

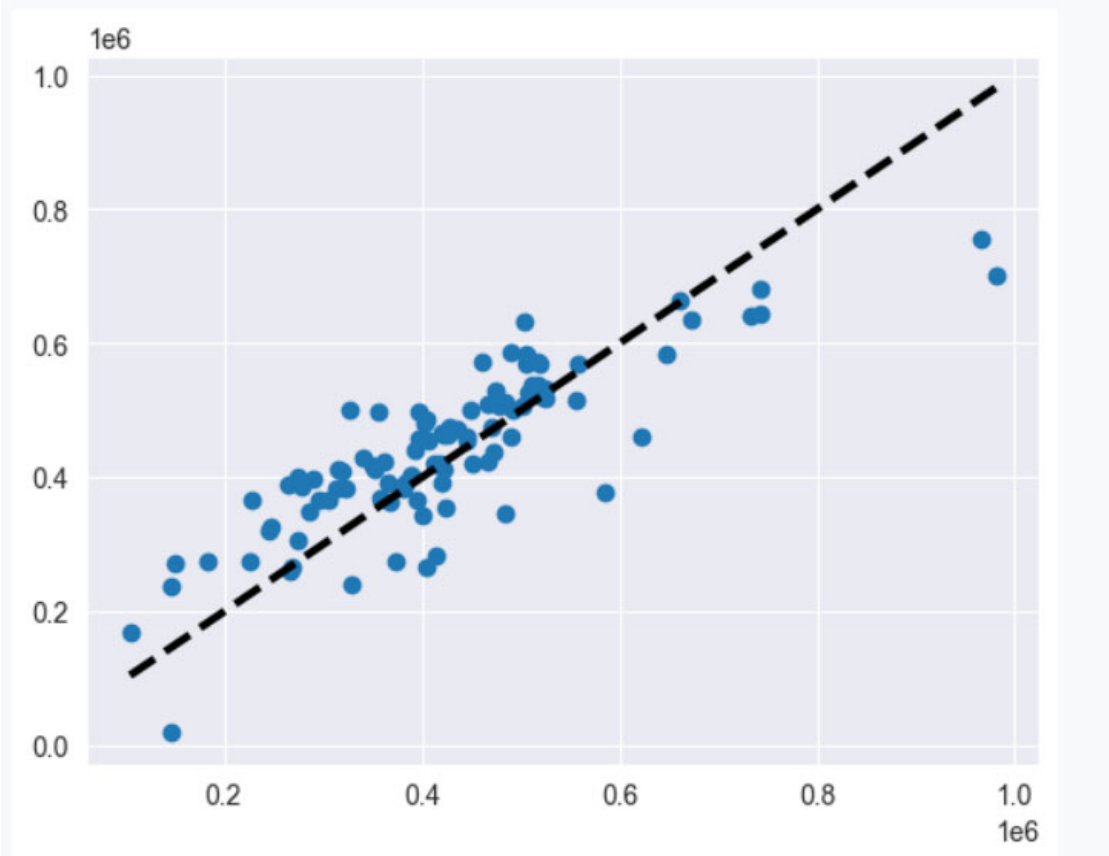
5.3 预测结果对比图

```
#预测结果对比折线图
train_test_comp.plot(figsize=(18,10))
plt.show()
#下图蓝色的是预测值，黄色的是实际值
```

图表 8 预测结果对比折线图

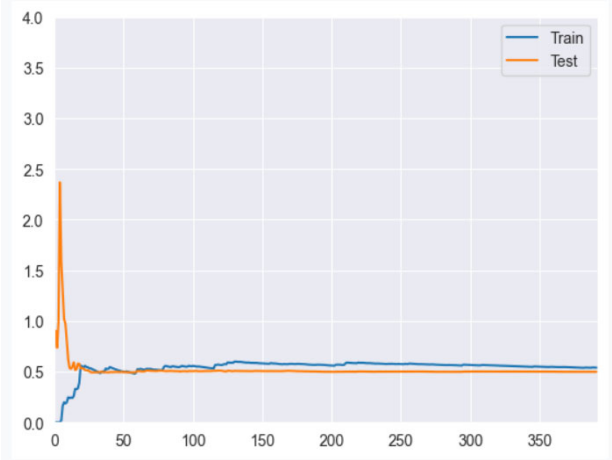
阶段性结论发现模型预测的结果大约 70%是测试房价大于实际房价，但从图上看来整体还可以接受，预测结果良好。

5.4 预测与真实值散点图



在测试集上面的整体评价,将真实值和预测值的散点分布图画在坐标轴上
横坐标 housing_test_labels，纵坐标 predict1.
根据上图可以发现在 0.2*10^6 ~ 0.6*10^6 的范围内预测较为准确超过 0.6*10^6 之后，预测值要比较小

5.5 学习曲线

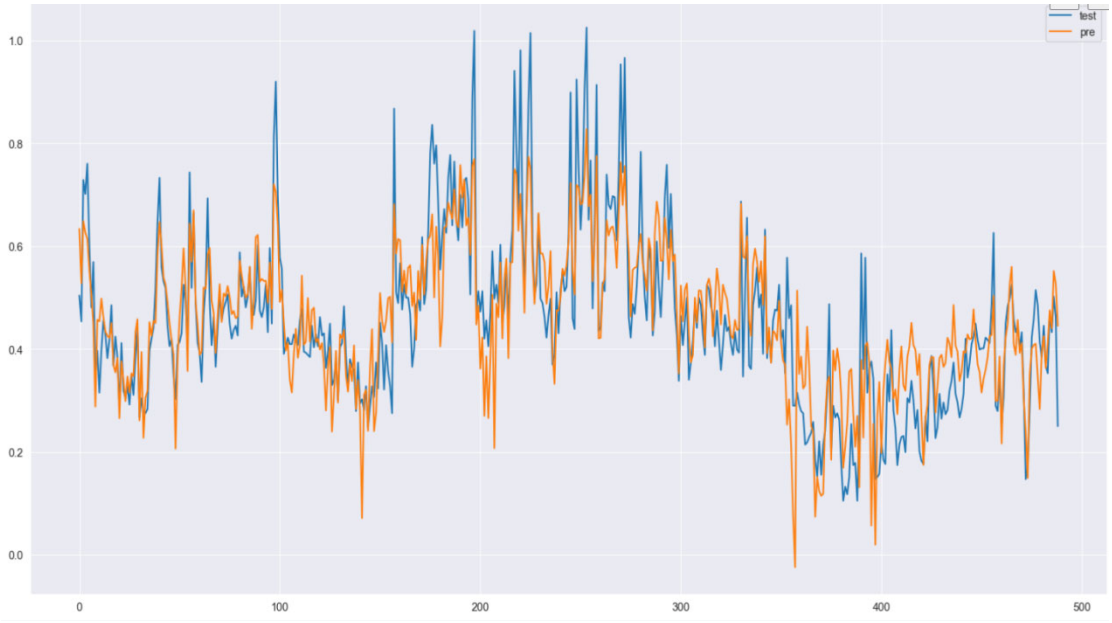


图表 9LinearRegression **学习曲线**

由图中学习曲线看来拟合效果还是可以接受的，后面随着训练次数增加误差不是那么大。

5.6 整体结果分析与问题

为了实验结果的稳健性，我又在整体数据集上采用同样的方法的方法进行了建模，也得到了整体的评价结果。



图表 10 **整体对比图**

综合上面的得分情况，以及更加直观的预测值与真实值的对比折线图，以及进一步的散

点图，采用 LinearRegression 算法的预测模型较为可观，但是还是不是那么的精确。

模型可观性：综合得分情况、预测值与真实值的对比折线图以及散点图，为采用 Linear Regression 算法的预测模型在整体上表现较为可观。这意味着模型能够捕捉到数据中的一些趋势和模式，对目标变量的变化进行一定程度的解释。模型的得分和对比图表显示出模型在一定程度上能够预测目标变量的变化。

预测精确性：然而，模型仍然不够精确。这可能意味着模型存在一些局限性或不足之处。可能的原因包括特征选择不充分、模型假设不满足等。这些因素都可能导致模型在预测过程中产生较大的误差。

6.模型优化

6.1 数据归一化

数据归一化使用 sklearn 中的 StandardScaler 可以将数值特征归一化到 [-1, 1] 范围内。归一化的目的是为了消除特征之间的差异，使模型收敛更快，提高训练效率。要注意的是，当输入数据分布范围很大的时候可以使用归一化，以防止过大或过小的值影响到模型的学习。把所有的数值特征都缩放到相同的尺度上；平移数据，使其均值为 0；拉伸数据使其变化范围变大。进行了上述的操作，为后面的优化做铺垫。

6.2 决策树回归

我们可以尝试采用决策树模型来进行可能的优化。通过调用 DecisionTreeRegressor 库进行调用建模。

```
#决策树回归
tr = DecisionTreeRegressor(max_depth=2)
#进行拟合
tr.fit(X_train, y_train)
# 预测值
tr_pre = tr.predict(X_test)

# 模型评分
print('Score:{:.4f}'.format(tr.score(X_test, y_test)))
# RMSE(标准误差)
print('RMSE:{:.4f}'.format(np.sqrt(mean_squared_error(y_test,tr_pre))))
```

```
Score:0.6817
RMSE:0.5063
```

图表 11 决策树回归得分

但是整体得分情况还不如原始的，决策树的得分仅有 0.68，所以这个优化是不合适的。

6.3 梯度提升法

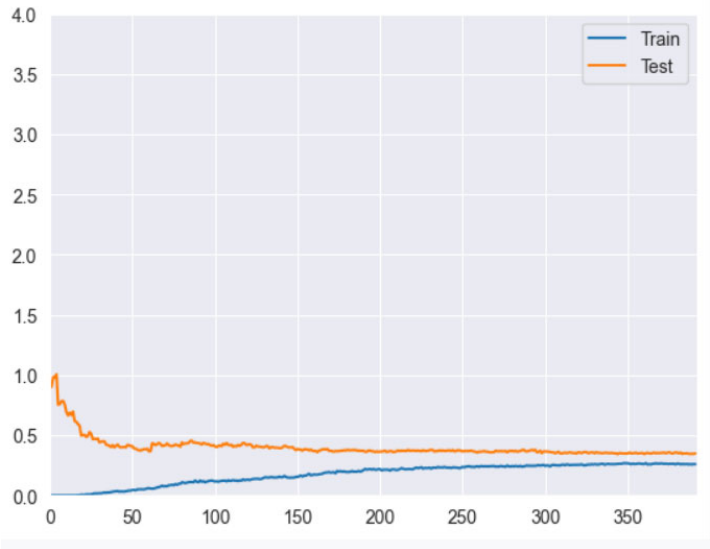
梯度提升法 GradientBoosting,对每个预测进行弱学习器的加权组合，形成更强的模型，由多个弱分类器组成。它通过梯度下降法迭代构建子模型来拟合残差，最后将各个子模型的结果进行叠加

```
gb = ensemble.GradientBoostingRegressor()

gb.fit(X_train, y_train)
gb_pre=gb.predict(X_test)

# 模型评分
print('Score:{:.4f}'.format(gb.score(X_test, y_test)))
# RMSE(标准误差)
print('RMSE:{:.4f}'.format(np.sqrt(mean_squared_error(y_test,gb_pre))))
```

这个模型的得分结果是 0.8476，RMSE 为 0.3504。效果要远远好于 LinearRegression 模型。



图表 12GradientBoosting 学习曲线

这个预测的训练结果是肉眼可见的好。事实证明梯度提升法是一个不错的优化思路。

6.4 支持向量机

```
linear_svr = SVR(kernel="linear")
linear_svr.fit(X_train, y_train)
linear_svr_pre = linear_svr.predict(X_test)
```

```
# 模型评分
print('Score:{:.4f}'.format(linear_svr.score(X_test, y_test)))
# RMSE(标准误差)
print('RMSE:{:.4f}'.format(np.sqrt(mean_squared_error(y_test,linear_svr
_pre))))
```

支持向量机的得分是 0.6903，RMSE 是 0.4995.这个与开始的 LinearRegression 方法的差别不大，优化效果不明显。

6.5 优化总结

在数据归一化的操作之下，显然整体看来梯度提升法的优化效果是最好的，它能最准确的预测房价，也是本实验得出的最优的预测模型。

7.实验心得

在研究机器学习算法在房价预测中的应用，并设计和实现一个能够高准确度预测房屋价格的模型的实验过程中，以下是在实验中我的一些实验心得：

数据质量至关重要：确保所使用的房价数据质量高、完整且准确。数据的准确性直接影响模型的预测性能。在数据准备阶段，进行数据清洗和处理，包括处理缺失值、异常值和重复值等，以确保数据的质量和一致性。

特征工程的重要性：特征工程是提取、转换和选择特征的过程。在房价预测中，选择合适的特征对模型的性能至关重要。需要进行特征分析，考虑哪些特征与房价具有相关性，并进行特征转换、组合或生成新的特征，以提高模型的表现。

算法选择与比较：探索和比较不同的机器学习算法对于房价预测的性能。常见的算法包括线性回归、决策树、随机森林、支持向量机等。尝试不同的算法，并评估它们的准确度、稳定性和泛化能力，选择最适合的算法作为基准模型。

模型评估与调优：使用适当的评估指标来评估模型的性能，例如均方根误差（RMSE）、平均绝对误差（MAE）。

总之,这个实验积累的算法经验和思考方式,将促进我今后算法学习的深入,并更好地应用算法解决实际问题。我会继续努力,进一步提高自己的算法设计与研究能力。