实验报告

姓名：曹昕城 学号：2111446 专业：密码科学与技术

一、问题描述及数据集分析：波士顿房价数据集统计的是20世纪70年代中期波士顿郊区房价的中位数，统计了城镇人均犯罪率、不动产税等共计13个指标，统计出房价，试图能找到那些指标与房价的关系。数据集中一共有506个样本，每个样本包含13个特征信息和实际房价，波士顿房价预测问题目标是给定某地区的特征信息，预测该地区房价，是典型的回归问题（房价是一个连续值）。波士顿房价数据集中主要的指标名称及其含义如下。



机器学习库scikit-learn中自带了波士顿房价数据集，可直接加载。房价预测可采用[线性](https://so.csdn.net/so/search?q=%E7%BA%BF%E6%80%A7&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/Dream_Gao1989/article/details/_blank)回归算法。

二、线性回归算法

线性回归是利用数理统计中回归分析，来确定两种或两种以上变量间相互依赖的定量关系的一种统计分析方法，通过属性的线性组合进行预测的线性模型，其目的是找到一条直线或者一个平面或者更高维的超平面，使得预测值与真实值之间的误差最小化。

线性回归分析中，如果只包括一个自变量和一个因变量，且二者的关系可用一条直线近似表示，这种回归分析称为一元线性回归分析。如果包括两个或两个以上的自变量，且因变量和自变量之间是线性关系，则称为多元线性回归分析。线性回归算法的公式如下，xi表示样本，可以是多维的，w为系数矩阵，b为偏置。目标是学习w和b两个参数，使得计算结果和真实结果之间的误差最小。

三、直接调用scikit-learn中的相关函数，实现波士顿房价简单预测的代码如下，主要步骤包括：

1.加载数据

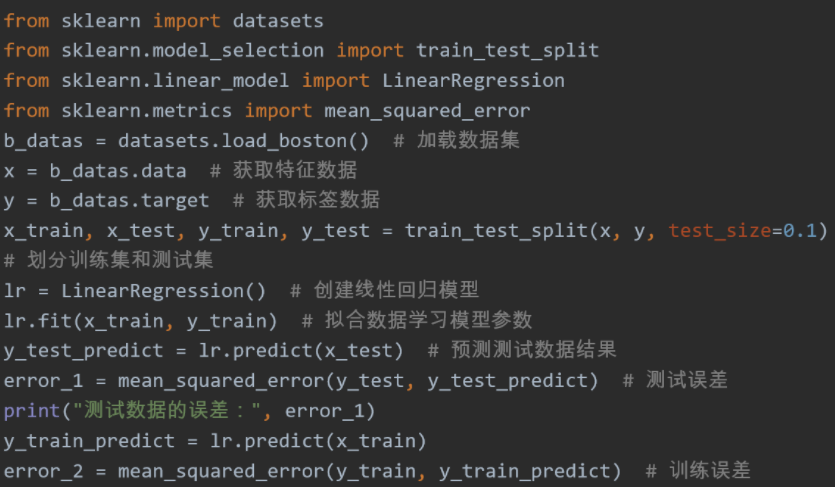
2.划分训练集和测试集

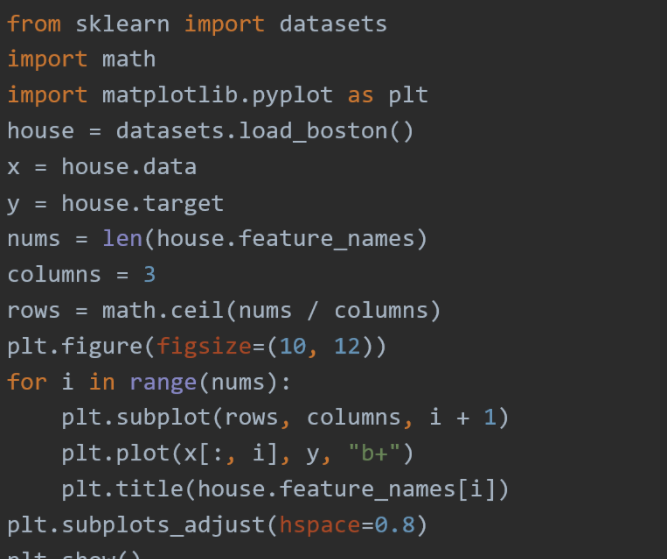
3.创建线性回归模型

4.拟合训练数据

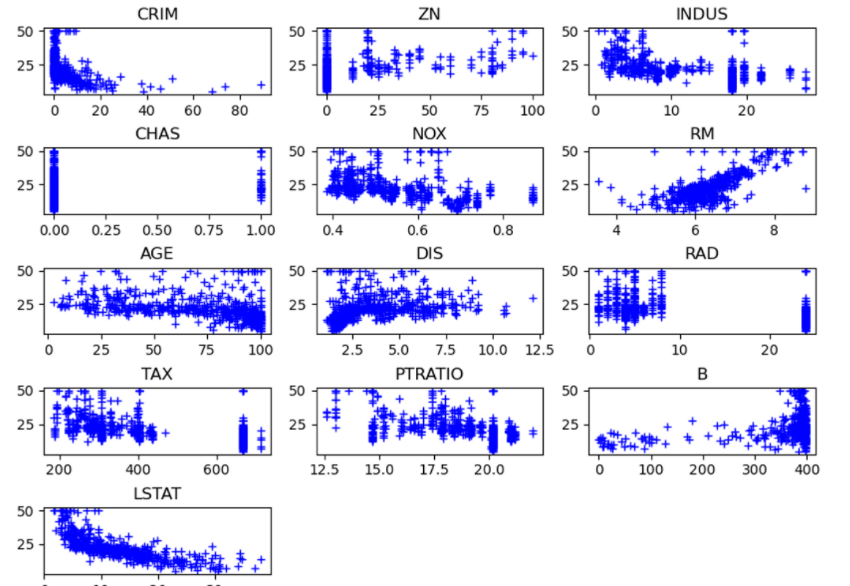
5.得到预测结果

6.计算相应的评测指标





各个指标与房价之间的关系可视化效果如下



通过可视化图表分析，可以看到不同特征的取值范围有较大差异，有些取值范围从0到1，有些从1到100，还有0到400等等，为了消除不同量纲的影响，可对数据进行标准化处理。一般采用将数据减去数据组的平均数再除以标准差。z = (x - u) / s （u表示均值，s表示标准差）

四、此外，通过图表效果可知，部分指标与房价之间存在明显的线性关系，而有些指标与房价之间的关系则不那么明显。可调用机器学习库中相关方法获取最相关的特征，关键代码如下：

import numpy as np

import pandas as pd

from pandas import Series, DataFrame

import matplotlib.pyplot as plt

import sklearn.datasets as datasets

from sklearn.model\_selection import train\_test\_split # train\_test\_split是交叉验证中常用的函数，功能是从样本中随机的按比例选取train data和test data。

from sklearn.metrics import r2\_score #导入评判标准，计算预测值和真实值的拟合程度

# 机器算法模型

boston = datasets.load\_boston()

data\_name = boston['feature\_names']

data\_1 = boston['data'][:5]

train = boston.data # 样本

# print(train.shape[0]) #输出为506

target = boston.target # 标签

# print(target.shape[0]) #输出为506

# 切割数据样本集合测试集

X\_train, x\_test, y\_train, y\_true = train\_test\_split(train, target, test\_size=0.2)

#参数：所要划分的样本特征集；所要划分的样本结果；

# 20%测试集；80%训练集

#波士顿房价预测----Lasso

from sklearn.linear\_model import Lasso # 线性回归算法Lasso回归，可用作特征筛选

# 模型训练

lasso = Lasso() #实例化lasso模型

lasso.fit(X\_train, y\_train) # 模型训练

# 预测数据

y\_pre\_lasso = lasso.predict(x\_test)

#R^2 score，即决定系数，反映因变量的全部变异能通过回归关系被自变量解释的比例。计算公式：R^2=1-\frac{SSE}{SST}

lasso\_score = r2\_score(y\_true, y\_pre\_lasso)

print('w = ', lasso.coef\_) # w值

print('b = ', lasso.intercept\_) # b值

print("训练集得分：{:.2f}".format(lasso.score(X\_train, y\_train)))

print("测试集得分：{:.2f}".format(lasso.score(x\_test, y\_true)))

# 绘图

# Lasso

plt.plot(y\_true, label='true')

plt.plot(y\_pre\_lasso,'r:', label='lasso')

plt.legend() #使用plt.legend()使上述plt.plot()代码产生效果

plt.show()

# plt.savefig('F:/新桌面/pred\_GT.jpg')

#波士顿房价预测----Ridge Regression

#1.模型导入

#注意：其他包的导入如Lasso回归

from sklearn.linear\_model import Ridge # 线性回归算法Ridge回归，岭回归

ridge = Ridge(alpha=1,max\_iter=1000) # 模型实例化

#alpha:正则化力度，必须是一个正浮点数。正则化提升了问题的条件，减少了估计器的方差。

#max\_iter:共轭梯度求解器的最大迭代次数。

ridge.fit(X\_train, y\_train) # 模型训练

# 模型预测

y\_pre\_ridge = ridge.predict(x\_test)

# print('预测结果：', y\_pre\_ridge)

print('w = ', ridge.coef\_) # w值

print('b = ', ridge.intercept\_) # b值

print("训练集得分：{:.2f}".format(ridge.score(X\_train, y\_train)))

print("测试集得分：{:.2f}".format(ridge.score(x\_test, y\_true)))

ridge\_score = r2\_score(y\_true, y\_pre\_ridge) #决定系数，反映因变量的全部变异能通过回归关系被自变量解释的比例

#绘图

plt.plot(y\_true, label='true')

plt.plot(y\_pre\_ridge,'r:', label='ridge')

plt.legend()

plt.show()

# plt.savefig(''F:/新桌面/回归曲线.jpg')

# 波士顿房价预测----Multi LinearRegression

#1.模型导入

#注意：其他包的导入如Lasso回归

from sklearn.linear\_model import LinearRegression # 多元线性回归算法

linear= LinearRegression()

linear.fit(X\_train, y\_train)

# 预测数据

y\_pre\_linear = linear.predict(x\_test)

print('预测结果：', y\_pre\_linear)

print('w = ', linear.coef\_) # w值

print('b = ', linear.intercept\_) # b值

print("训练集得分：{:.2f}".format(linear.score(X\_train, y\_train)))

print("测试集得分：{:.2f}".format(linear.score(x\_test, y\_true)))

linear\_score = r2\_score(y\_true, y\_pre\_linear)

# 绘图

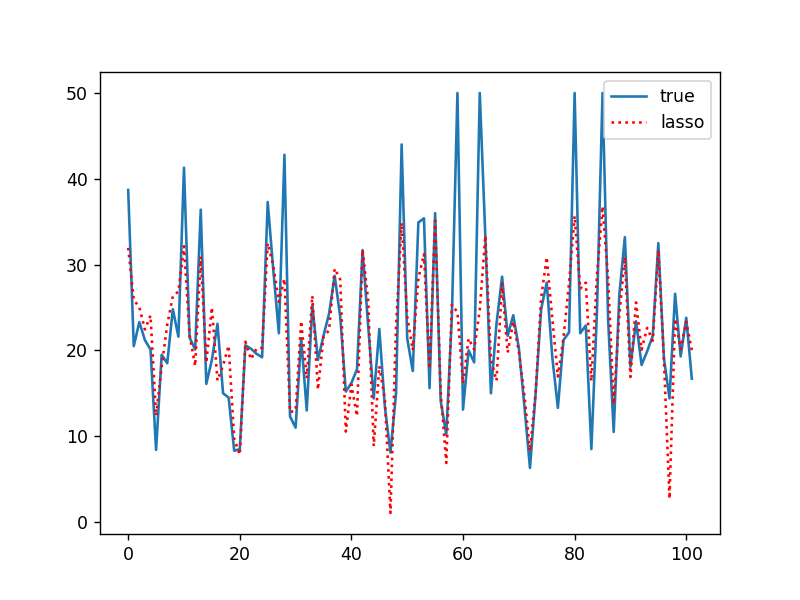
plt.plot(y\_true, label='true')

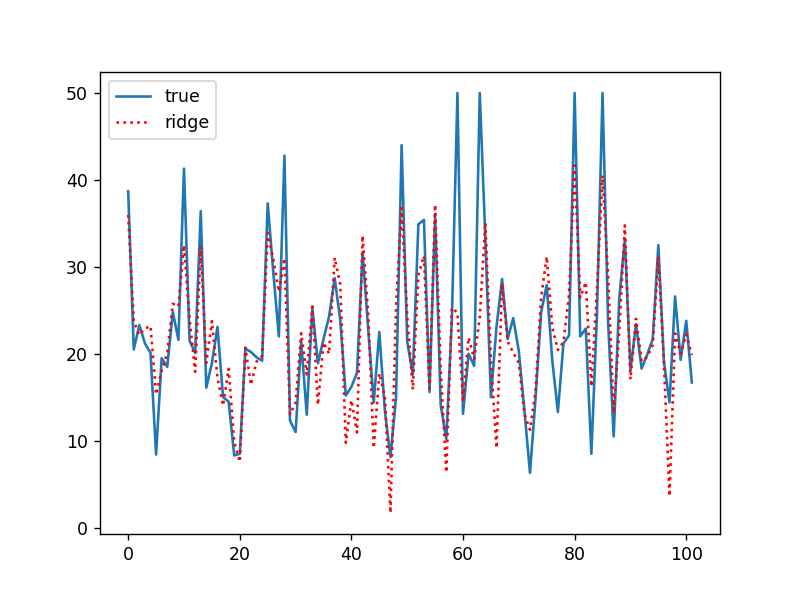
plt.plot(y\_pre\_linear,'r:', label='linear', )

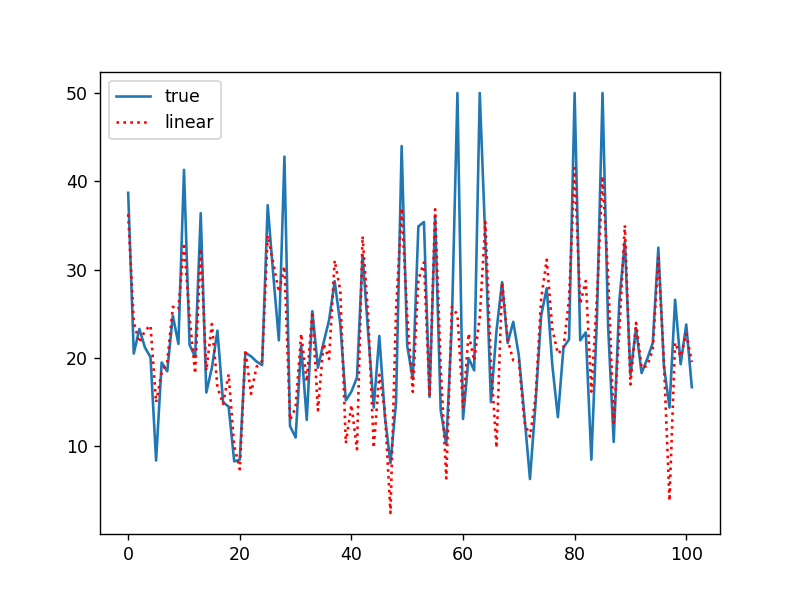
plt.legend()

plt.show()

# plt.savefig(''F:/新桌面/第三个曲线.jpg')







五、结束语：利用sklearn工具包，利用多种算法以及图表方式进行整合。面对这样一个从未接触的东西，我大多数都是来自于csdn中庞大的代码进行筛选再加上少部分自己的理解，让我切实了解了python内置工具包的强大。