**房价预测的线性回归算法**

1）样本数据分析与处理

<https://www.kaggle.com/harlfoxem/housesalesprediction?select=kc_house_data.csv>下载后，对样本数据进行分析。

该实验是依据房屋的属性信息，包括房屋的卧室数量，卫生间数量，房屋的大小，房屋地下室的大小，房屋的外观，房屋的评分，房屋的修建时间，房屋的翻修时间，房屋的位置信息等，对房屋的价格进行预测。

* 分析实验要求
* 下载实验数据
* 初步分析数据

This dataset contains house sale prices for King County, which includes Seattle. It includes homes sold between May 2014 and May 2015.

数据字段:

**id** - Unique ID for each home sold

**date** - Date of the home sale

**price** - Price of each home sold

**bedrooms** - Number of bedrooms

**bathrooms** - Number of bathrooms, where .5 accounts for a room with a toilet but no shower

**sqft\_living** - Square footage of the apartments interior living space

**sqft\_lot** - Square footage of the land space

**floors** - Number of floors

**waterfront** - A dummy variable for whether the apartment was overlooking the waterfront or not

**view** - An index from 0 to 4 of how good the view of the property was

**condition** - An index from 1 to 5 on the condition of the apartment,

**grade** - An index from 1 to 13, where 1-3 falls short of building construction and design, 7 has an average level of construction and design, and 11-13 have a high quality level of construction and design.

**sqft\_above** - The square footage of the interior housing space that is above ground level

**sqft\_basement** - The square footage of the interior housing space that is below ground level

**yr\_built** - The year the house was initially built

**yr\_renovate**d - The year of the house’s last renovation

**zipcode** - What zipcode area the house is in

**lat** - Lattitude

**long** - Longitude

**sqft\_living15** - The square footage of interior housing living space for the nearest 15 neighbors

**sqft\_lot15** - The square footage of the land lots of the nearest 15 neighbors

**参考代码：**

import pandas as pd

import matplotlib.pyplot as plt

raw\_data = pd.read\_csv('kc\_house\_data.csv')

raw\_data #查看原数据

raw\_data.info() #查看数据特征类型

raw\_data.duplicated().sum() #检查是否有重复特征数据

* 划分训练集和验证集

参考代码：

X = raw\_data.drop(['id', 'date', 'price'], axis=1)

y = raw\_data['price']

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=1026)

* 对特征进行归一化

参考代码：

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

sc.fit(X\_train)

X\_train= sc.transform(X\_train)

X\_test = sc.transform(X\_test)

1. 线性回归算法建模及分析

* 要求使用python语言实现线性回归和梯度下降优化代码（不能调包）

3）超参数调优 （选做）

* 网格搜索超参数调优

示例代码：

from sklearn.model\_selection import GridSearchCV

model\_slect = GradientBoostingRegressor() #将模型替换为线性回归模型

parameters = {'loss': ['ls','lad','huber','quantile'], 'learning\_rate':[0.1, 0.2], 'min\_samples\_leaf': [1,2,3,4]}

time\_start=time.time()

model\_gs = GridSearchCV(estimator=model\_slect, param\_grid=parameters, verbose=3)

model\_gs.fit(X,y)

4）特征选择

* 用相关系数来观察特征之间以及特征和标签之间的相关性
* 用散点图观察特征与标签之间的相关性
* 用模型尝试去掉特征

5）记录并分析实验结果

**源代码：**

import pandas as pd

import matplotlib.pyplot as plt

raw\_data = pd.read\_csv(r'D:\研一\机器学习\kc\_house\_data.csv')

raw\_data #查看原数据

raw\_data.info() #查看数据特征类型

raw\_data.duplicated().sum() #检查是否有重复特征数据

X = raw\_data.drop(['id', 'date', 'price'], axis=1)

y = raw\_data['price']

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=1026)

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

sc.fit(X\_train)

X\_train= sc.transform(X\_train)

X\_test = sc.transform(X\_test)

import numpy as np

# 计算损失函数值

def J(X\_b,y,theta):

try:

return np.sum((X\_b.dot(theta) - y) \*\* 2) / 2 \* len(y)

except:

return float('inf')

# 计算梯度（针对一组theta参数的偏导数）

def dJ(X\_b,y,theta):

return X\_b.T.dot(X\_b.dot(theta) - y) / len(y)

# 以批量梯度下降法为例

def BGD(X\_b,y,initial\_theta,eta=0.01,iters=1e4,epsilon=1e-4):

theta = initial\_theta # 将参数的初始值赋给theta变量

curr\_iter = 1

while curr\_iter < iters: # 判断迭代次数

gradient = dJ(X\_b,y,theta) # 计算当前theta对应的梯度

last\_theta = theta # 先保存theta的旧值

theta = theta - eta \* gradient # 更新theta

cost\_value = J(X\_b,y,theta) # 计算更新后的theta对应的损失函数值

last\_cost\_value = J(X\_b,y,last\_theta) # 计算更新前的theta对应的损失函数值

# 如果两次损失函数的值相差的非常小，则认为损失函数已经最小了

if abs(cost\_value - last\_cost\_value) < epsilon:

break

curr\_iter += 1 # 每次迭代完毕，迭代次数加1

return theta,cost\_value # 返回最佳的theta和对应的损失函数值

# 拼接训练样本的X\_b

X\_b = np.hstack([np.ones((len(X\_train),1)),X\_train])

# 初始化theta

initial\_theta = np.random.random(size=(X\_b.shape[1]))

theta,cost\_value = BGD(X\_b,y\_train,initial\_theta,iters=1e5)

# 拼接测试样本的X\_b

X\_b\_test = np.hstack([np.ones((len(X\_test),1)),X\_test])

# 在测试集上预测

y\_predict = X\_b\_test.dot(theta)

from sklearn.metrics import mean\_squared\_error

r2 = 1 - mean\_squared\_error(y\_test,y\_predict) / np.var(y\_test)

print("在测试集上的R^2为 ",r2)

from sklearn.linear\_model import LinearRegression

linreg=LinearRegression()

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

from sklearn import metrics

y\_train\_pred = linreg.predict(X\_train)

y\_test\_pred = linreg.predict(X\_test)

train\_err = metrics.mean\_squared\_error(y\_train, y\_train\_pred)

test\_err = metrics.mean\_squared\_error(y\_test, y\_test\_pred)

print( 'Train\_RMSE分别是: {:.2f}'.format(train\_err) )

print( 'Test\_RMSE分别是: {:.2f}'.format(test\_err) )

predict\_score = linreg.score(X\_test,y\_test)

train\_score = linreg.score(X\_train,y\_train)

print('训练集上的正确率为:{:.2f} '.format(train\_score) )

print('测试集上的正确率为:{:.2f} '.format(predict\_score) )

# 输出模型测试集上的效果

%matplotlib inline

import matplotlib.pyplot as plt

fig = plt.figure(figsize=(8,5))

plt.plot(y\_test, label='real')

plt.plot(y\_test\_pred, label='predict')

plt.legend()

plt.show()

new\_X = np.array([[3,1,1110,5550,1,1,0,3,8,1110,0,1888,0,98178,47,-122,1555,8888]])

print("自己希望购买的房屋数据：",new\_X)

new\_X = sc.transform(new\_X)

print("预测希望购买的价格为：{}".format(linreg.predict(new\_X)))

from sklearn.model\_selection import GridSearchCV

model\_select=LinearRegression()

parameters={'fit\_intercept':['True','False'],'copy\_X':['True','False'],'n\_jobs':[1,2,3]}

model\_gs=GridSearchCV(estimator=model\_select,param\_grid=parameters,verbose=3)

model\_gs.fit(X\_train,y\_train)

print("最优参数：",model\_gs.best\_estimator\_.get\_params())

print("相关系数矩阵为:\n")

print(theta)

pd.plotting.scatter\_matrix(raw\_data,diagonal='kde',figsize=(30,30),color='m')

plt.show()

**运行（测试）过程及结果：**



















