**第8-9章 线性回归&逻辑回归**

1. 教材P158二、1

import numpy as np  
from sklearn.feature\_extraction import DictVectorizer  
from sklearn import linear\_model  
import random  
import matplotlib.pyplot as plt  
import numpy as np  
from sklearn.linear\_model import LinearRegression  
  
  
# 定义多特征值数据 ：  
x = np.array([[8,28,9,51],[2,30,16,45],[12,36,18,43],[15,43,21,52],  
 [11,45,15,55],[21,38,19,60],[20,39,22,58],[19,36,21,55],  
 [3,22,21,54],[8,29,36,56],[18,30,41,58],[20,26,42,59],  
 [9,21,32,52]])  
y = np.array([80.3,78.5,82.1,77.6,76.9,73.2,70.5,76.2,72.1,76.3,78.2,77.1,75.2])  
  
  
reg = linear\_model.LinearRegression()  
reg.fit(x,y)  
  
# coef\_ attribute 代表的是这个 线性回归预测模型中对于 最终拟合曲线的权重。  
print('value weight is : (w)学习参数' , reg.coef\_)  
print('学习参数 b 截断数据 ： ',reg.intercept\_)  
# 由于学习参数w 权重中的  
w = reg.coef\_  
b = reg.intercept\_  
  
  
reg\_main = linear\_model.LinearRegression()  
reg\_main.fit(x[:,0:1],y)  
  
wmain = reg\_main.coef\_  
bmain = reg\_main.intercept\_  
print(wmain)  
print(bmain)  
fig=plt.figure()  
ax1=fig.add\_subplot(1,1,1)  
# ax2=fig.add\_subplot(1,4,2)  
# ax3=fig.add\_subplot(1,4,3)  
# ax4=fig.add\_subplot(1,4,4)  
  
ax1.scatter(x[:,0:1],y,marker='o')  
# ax2.scatter(x[:,1:2],y,marker='o')  
# ax3.scatter(x[:,2:3],y,marker='o')  
# ax4.scatter(x[:,3:4],y,marker='o')  
tmp = x[:,0:1] \* w[0] + x[:,1:2] \* w[1] + x[:,2:3] \* w[2] + x[:,3:4] \* w[3] + bmain  
ax1.scatter(tmp,y,c='red')  
  
# tmp = np.array(tmp)  
pre =reg\_main.predict(tmp)  
ax1.plot(tmp,pre,c = 'green')  
  
plt.show()

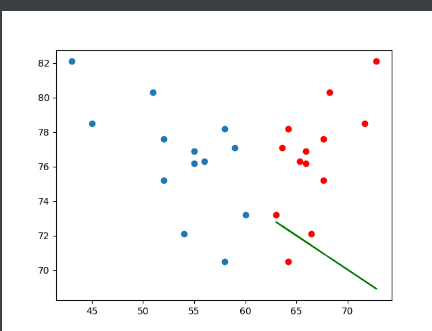
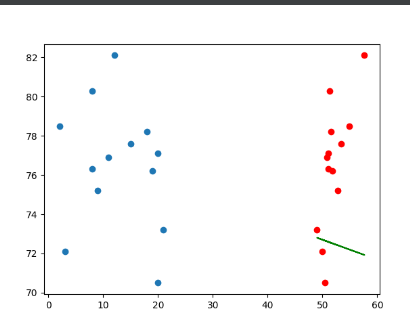
**实验结果： 由此可以看出将其中的值作为核心特征值。**

**value weight is : (w)学习参数 [ 0.10893698 0.04716845 0.09558218 -0.57748723]**

**学习参数 b 截断数据 ： 102.25639011559056**

**以下图片（左图），蓝点代表的是根据第一个特征值画出的与y值的相关三点图，右侧红点则代表四个特征值迭代计算后用学习参数为w和b 将所有的四种特征值整合成为一种特征值和y的相关值，绿色线条则是最后的线性的值的预测拟合曲线。由于将四维平面拟合成为二维线段所以看起来极其不相关。**

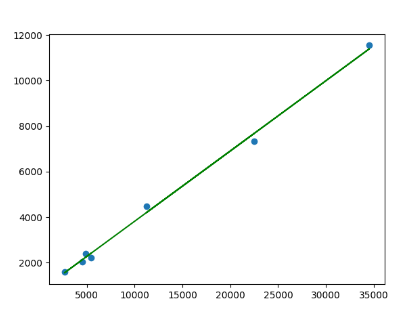
**右图则是将主要特征值改为第四特征值进行拟合，相关代码在图片之后。**



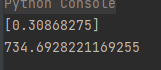
eg\_main = linear\_model.LinearRegression()  
reg\_main.fit(x[:,3:4],y)  
  
wmain = reg\_main.coef\_  
bmain = reg\_main.intercept\_  
print(wmain)  
print(bmain)  
fig=plt.figure()  
ax1=fig.add\_subplot(1,1,1)  
# ax2=fig.add\_subplot(1,4,2)  
# ax3=fig.add\_subplot(1,4,3)  
# ax4=fig.add\_subplot(1,4,4)  
  
ax1.scatter(x[:,3:4],y,marker='o')  
# ax2.scatter(x[:,1:2],y,marker='o')  
# ax3.scatter(x[:,2:3],y,marker='o')  
# ax4.scatter(x[:,3:4],y,marker='o')  
tmp = x[:,0:1] \* w[0] + x[:,1:2] \* w[1] + x[:,2:3] \* w[2] + x[:,3:4] \* w[3] + bmain  
tmp = x[:,3:4] \* w[3] + bmain  
ax1.scatter(tmp,y,c='red')

1. 教材P158二、2

import numpy as np  
from sklearn.feature\_extraction import DictVectorizer  
from sklearn import linear\_model  
import random  
import matplotlib.pyplot as plt  
import numpy as np  
from sklearn.linear\_model import LinearRegression  
x = np.array([22460,11226,34547,4851,5444,2662,4549])  
x = x.reshape(-1,1)  
y = np.array([7326,4490,11546,2396,2208,1608,2035])  
reg = linear\_model.LinearRegression()  
reg.fit(x,y)  
pre = reg.predict(x)  
fig = plt.figure()  
ax1 = fig.add\_subplot(1,1,1)  
ax1.scatter(x,y,marker='o')  
ax1.plot(x,pre,c = 'green')  
plt.show()



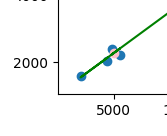
（2）方程参数 ： w = 0.30868275 b = 734.6928221169255. 曲线如上图绿色直线



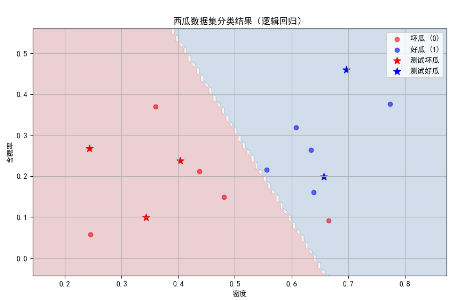
（3）若某地区的人均GDP 为5000元，预测其人均消费水平。

prex = np.array([[5000],])  
prey = reg.predict(prex)  
ax1.scatter(prex,prey,color='pink')





1. 教材P174二、1
2. import numpy as np  
   import matplotlib.pyplot as plt  
   from sklearn.linear\_model import LogisticRegression  
   from sklearn.model\_selection import train\_test\_split  
     
   # 西瓜数据集（密度, 含糖率, 标签：0=坏瓜, 1=好瓜）  
   X = np.array([  
    [0.697, 0.460], [0.774, 0.376], [0.634, 0.264], [0.608, 0.318], [0.556, 0.215],  
    [0.403, 0.237], [0.481, 0.149], [0.437, 0.211], [0.666, 0.091], [0.243, 0.267],  
    [0.245, 0.057], [0.343, 0.099], [0.639, 0.161], [0.657, 0.198], [0.360, 0.370]  
   ])  
   y = np.array([1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0])  
     
   # 划分训练集和测试集  
   X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)  
     
   # 创建模型并训练  
   model = LogisticRegression()  
   model.fit(X\_train, y\_train)  
     
   # 评估模型  
   print("训练集准确率:", model.score(X\_train, y\_train))  
   print("测试集准确率:", model.score(X\_test, y\_test))  
     
   # 设置画布  
   plt.figure(figsize=(10, 6))  
     
   # 绘制训练数据点  
   plt.scatter(X\_train[y\_train == 0, 0], X\_train[y\_train == 0, 1], color='red', label="坏瓜 (0)", alpha=0.6)  
   plt.scatter(X\_train[y\_train == 1, 0], X\_train[y\_train == 1, 1], color='blue', label="好瓜 (1)", alpha=0.6)  
     
   # 绘制测试数据点（用星号标记）  
   plt.scatter(X\_test[y\_test == 0, 0], X\_test[y\_test == 0, 1], color='red', marker='\*', s=100, label="测试坏瓜")  
   plt.scatter(X\_test[y\_test == 1, 0], X\_test[y\_test == 1, 1], color='blue', marker='\*', s=100, label="测试好瓜")  
     
   # 绘制决策边界  
   x\_min, x\_max = X[:, 0].min() - 0.1, X[:, 0].max() + 0.1  
   y\_min, y\_max = X[:, 1].min() - 0.1, X[:, 1].max() + 0.1  
   xx, yy = np.meshgrid(np.linspace(x\_min, x\_max, 100), np.linspace(y\_min, y\_max, 100))  
   Z = model.predict(np.c\_[xx.ravel(), yy.ravel()]).reshape(xx.shape)  
   plt.contourf(xx, yy, Z, alpha=0.2, cmap=plt.cm.RdBu)  
     
   # 添加标签和标题  
   plt.xlabel("密度")  
   plt.ylabel("含糖率")  
   plt.title("西瓜数据集分类结果（逻辑回归）")  
   plt.legend()  
   plt.grid(True)  
   plt.show()



1-3题的要求：

1）针对当前数据、模型，**详叙数据预处理、模型参数调整过程及效果比对**；

2）代码要列出，重点代码加注释说明，特别是自己调试过程中的自我理解；

3）运行结果要截图，结果要文字说明；

4）注意排版