**《Python实现对数几率回归模型》**

计算机1603班

1611640305

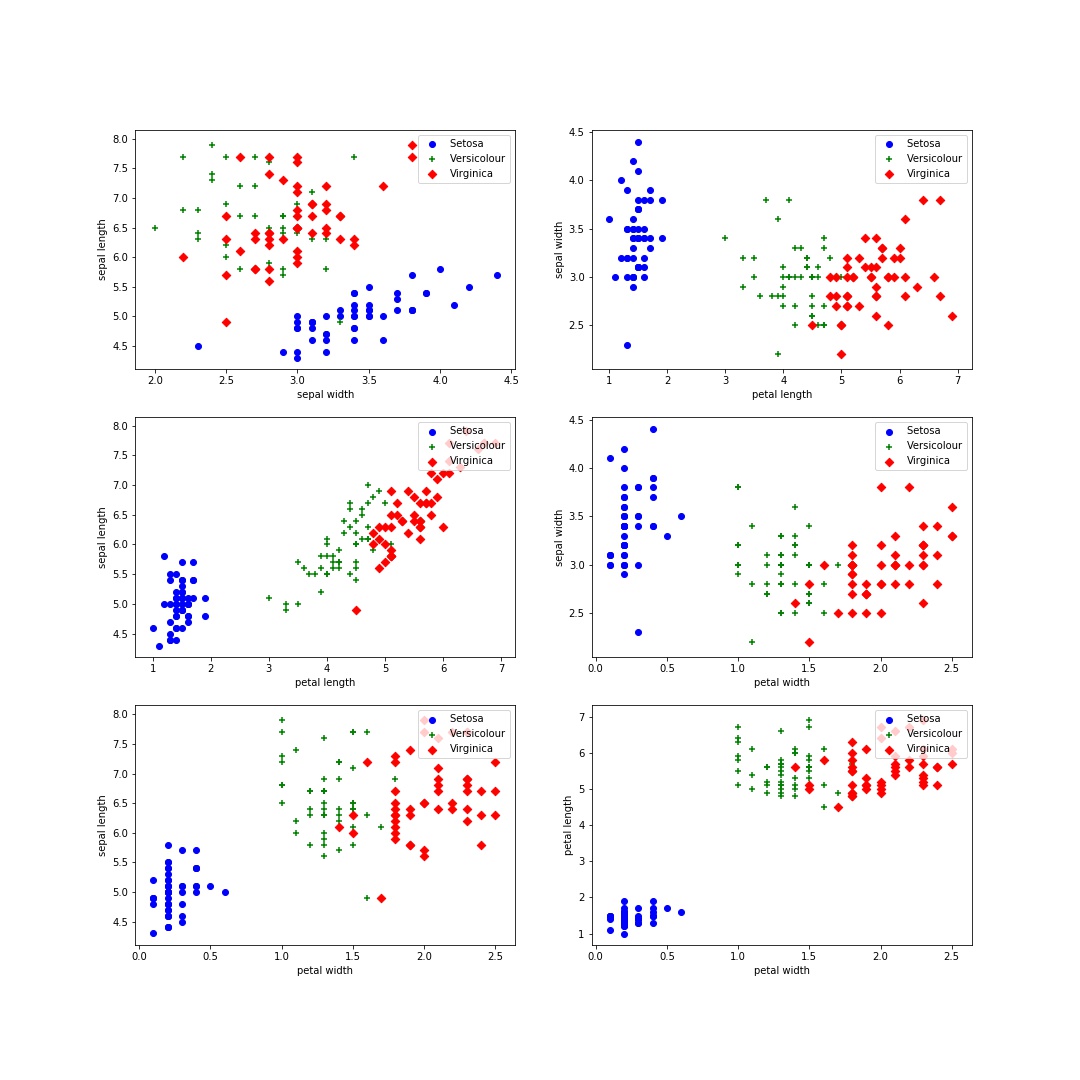
温吉祥

**课题内容：**

**1：题目：**使用Python实现对数几率回归模型.

**2：问题描述：**根据所学的对数几率回归内容，结合鸢尾花数据集，利用Python编写实现一个对数几率回归模型，并对鸢尾花数据集进行分类来测试所训练模型的效能。

**3：数据集描述：**从<http://archive.ics.uci.edu/ml/datasets/Iris上下载该数据集并保存为iris_data.csv> 文件。可看出该数据集是关于三类不同花（Setosa 、Versicolour 、Virginica）的分类，每类花有四个属性（萼片长度、萼片宽度、花瓣长度、花瓣宽度）和50个实例。下面是通过matplotlib对四个属性进行两两结合得到的散点图，对应程序(get\_data.py)并保存为iris\_picture.jpg。



通过该图发现第一张子图的Setosa 和Versicolour在属性sepal width 和sepal length中具有较好的线性区分度，所以便筛选出对应的数据集（即前一百个数据和前两个属性），以下程序即可实现：

from sklearn.datasets import load\_iris

data\_need = load\_iris() #获取原始数据

X = data\_need.data[:100,:2] #取数据的前两类和前两个属性

y = data\_need.target[:100] #取前两类数据标记值(0 or 1)

**4：编程实现：**首先将数据集分为训练集和测试集，从对数几率回归到最大对数似然，再用梯度下降法优化得到最优参数。最后用测试集测试模型的效能。

4.1：数据分类：利用sklearn包中的model\_selection方法实现，X\_train, X\_test, y\_train, y\_test = model\_selection.train\_test\_split(X\_hat, y, **test\_size=0.5**, random\_state=1).通过设置test\_size的大小即可按比例将数据集分为训练集和测试集。

4.2：对数几率回归函数：

def sigmoid(x, beta):

return 1.0 / (1 + np.math.exp(- np.dot(beta, x.T)))

4.3：最大似然估计函数：

def theta\_sub(x, y, beta):

return -y \* np.dot(beta, x.T) + np.math.log(1 + np.math.exp(np.dot(beta, x.T)))

def theta(X, y, beta):

sum = 0

m,n = np.shape(X)

for i in range(m):

sum += theta\_sub(X[i], y[i], beta)

return sum

4.4：梯度下降法：

def grad\_desc(X, y):

h = 0.05 #学习率

max\_iters= 300 # 最大迭代次数

m, n = np.shape(X) #(m,n) == (100,3)

beta = np.zeros(n) # 初始化beta值 = [0,0,0]

delta\_beta = np.ones(n)\*h #初始化梯度delta\_beta为学习率h大小

tha = 0

tha\_temp = 0

for i in range(max\_iters):

beta\_temp = beta.copy()

for j in range(n):

beta[j] += delta\_beta[j] #梯度delta\_beta

tha\_temp = theta(X, y, beta)

delta\_beta[j] = -h \* (tha\_temp - tha) / delta\_beta[j]

#课本408页（B.17）

beta[j] = beta\_temp[j]

beta += delta\_beta

tha = theta(X, y, beta) #当前参数下的损失函数值 tha

print('该模型的损失函数值为：%.2f ' %tha)

return beta

4.5：预测函数：

'''

预测函数，用测试集和获得的beta值带入sigmoid函数得到介于0-1的预测值y,

以0.5作为阈值，大于0.5输出1，反之输出0

'''

def predict(X, beta):

m, n = np.shape(X)

y = np.zeros(m)

for i in range(m):

if sigmoid(X[i], beta) > 0.5:

y[i] = 1

else:

y[i] = 0

return y

return beta

4.6：结果统计：

count = 0

ret\_target = []

for i in range(m\_test):

if y\_pret[i] == y\_test[i]:

count += 1

ret\_target += '1'

else:

ret\_target += '0'

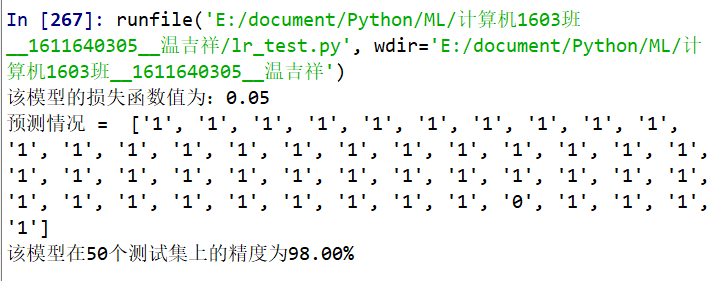
accuracy = (count/m\_test)

print('预测情况 = ',ret\_target)

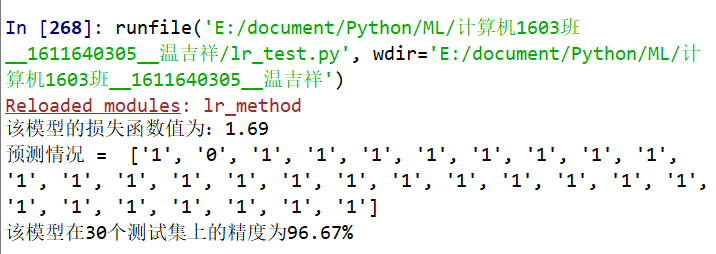
print("该模型在{0}个测试集上的精度为{1:.2%}".format(m\_test,accuracy))

4.7：实验结果：

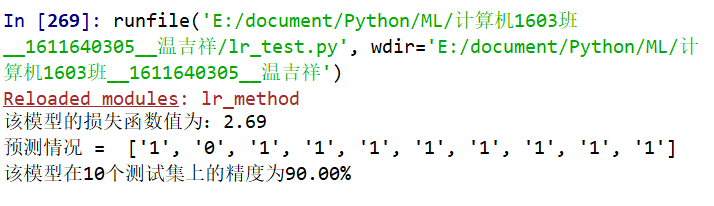
1. 50%为训练集，50%为测试集下：



1. 70%为训练集，30%为测试集下：



1. 90%为训练集，10%为测试集下：



**5：实验结果分析：**在优化训练模型过程中通过控制变量法不断改变学习率h和迭代次数max\_iters来查看在不同分类情况下的精度和损失函数大小。最后在当h = 0.05和max\_iters = 300的时候该训练模型在测试集上拥有较好的精度和损失函数值.分别为：

|  |  |  |  |
| --- | --- | --- | --- |
| 训练集个数 | 测试集个数 | 精度 | 损失函数值 |
| 50 | 50 | 98.00% | 0.05 |
| 70 | 30 | 96.67% | 1.69 |
| 90 | 10 | 90.00% | 2.69 |

**6：参考书籍：**周志华《机器学习》

【美】Alexander T.Combs Python机器学习实践指南

**参考博客：**<https://blog.csdn.net/Snoopy_Yuan/article/details/63684219>

**参考网站：**https://scipy.org/