## 机器学习与数据挖掘平时作业一

**学号：16337072 姓名：郭旺振 专业：计算机科学与技术**

**2.1** 数据集包含1000一个样本，其中500个正例、500个负例子，将其划分为包含70%样本的训练集和30%样本的测试集用于留出法评估，试估算共有多少种划分方式。

答：共有划分方式：****种。

**2.2**数据集包含1000个样本其中正例、反例各一半，假定学习算法所产生的模型是将新样本预测为训练样本数较多的类别（训练样本数相同时进行随机猜测），试给出用10折交叉验证法和留一法分别对错误率进行评估所得的结果。

答：**10折交叉验证法：**将样本划分为10个大小相似的互斥子集，并保持原来的数据分布的一致性。每次用9个子集进行训练，余下的一个用来预测。由于训练集的数据分布与原来的保持一致，即正例包含50%、负例包含50%，所以训练得到的学习器将会进行随机猜测，且测试集的正例和负例各一半，所以该学习器的错误率和正确率均为50%。

**留一法：**每次用一个来进行测试，剩下的用来训练。则训练集的正例和负例数目将会不一样。假如正例多于负例，则测试集是一个负例，而学习器算法的预测将为正例，此时预测错误；若负例多于正例，则测试集是一个正例，而学习器算法预测的将是负例，此时预测错误。即无论何种情况，学习器的错误率为100%、正确率为0.

**2.4**试述真正例率(TPR)、假正例率(FPR)与查准率(P)、查全率(R)之间的联系。

答：**真正例率：**表示预测为正例且是实际为正例的样本占所有实际正例样本的比率，即；**假正例率：**表示预测为正例但实际为负例的样本占所有实际为负例样本的比例，即;**查准率：**表示预测为正例且实际为正例的样本占所有预测未正例的样本的比例，即;**查全率：**表示预测为正例且实际为正例的样本占所有实际为正例样本的比例，即。从表达式上看，查全率和真正例率本质上是一样，是同一个式子的不同含义。而查准率和假正例率没有直接的数值关系。

**2.8** Min-max规范化和z-score规范化是两种常用的规范化方法，令x和分别表示变量在规范化前后的取值，相应的，令表示规范化后的最小值和最大值，，则min-max规范化、z-score规范化分别如式子(2.43)和(2.44)所示，试分析二者的优缺点。



答：**min-max规范化**实际上是将数据进行了长度上的压缩，能够将数据映射到对应的区间上，但同时也改变了数据之间的关系，没有很好的反映原来数据应有的特征。此外，每当有新的元素进来，只有该元素大于最大值或者小于最小值时才要重新计算全部元素。但若存在一个极大（小）的元素，会导致其他元素变得非常小（大）。

**z-score规范化**实际上是对数据进行了归一化处理，保持了原来数据之间的关系又能够达到压缩数据长度的目的。该标准化对个别极端元素不敏感，且把所有元素分布在0的周围，一般情况下元素越多，0周围区间会分布大部分的元素，每当有新的元素进来，都要重新计算方差和均值。

**3.3** 编程实现对数率回归，并给出西瓜数据集上的结果。

代码实现：

import numpy as np

def sigmod(z):

return 1/(1+np.exp(-z))

def firstDerivative():

total = np.zeros(3)

for i in range(dataX\_.shape[0]):

temp = np.dot(beta\_,dataX\_[i])

total += dataX\_[i]\*(-dataY[i]+sigmod(temp))

return total

# def secondDerivative():

# total = 0

# for i in range(dataX\_.shape[0]):

# p1 = sigmod(np.dot(beta\_,dataX\_[i]))

# total += np.dot(dataX\_[i],dataX\_[i])\*p1\*(1-p1)

# return total

def train(beta\_,dataX\_,dataY):

traintimes = 500

h=0.01

for i in range(traintimes):

beta\_ = beta\_ - h\*firstDerivative()

print("The weight and bias: ")

print(beta\_)

def test(beta\_,dataX\_,dataY):

correct\_num = 0

for i in range(dataX\_.shape[0]):

pi = sigmod(np.dot(beta\_,dataX\_[i]))

if pi>0.5 and dataY[i]==1 or pi<0.5 and dataY[i]==0:

correct\_num +=1

print("correctrate in trainset:"+str(correct\_num/dataX\_.shape[0]))

def getploty(beta\_,x):

y = -beta\_[0]\*x/beta\_[1]-beta\_[2]/beta\_[1]

return y

import matplotlib.pyplot as plt

plt.rcParams['font.sans-serif']=['SimHei']

plt.rcParams['axes.unicode\_minus'] = False

if \_\_name\_\_=='\_\_main\_\_':

dataset=[0.697, 0.460, 1,

0.774, 0.376, 1,

0.634, 0.264, 1,

0.608, 0.318, 1,

0.556, 0.215, 1,

0.403, 0.237, 1,

0.481, 0.149, 1,

0.437, 0.211, 1,

0.666, 0.091, 0,

0.243, 0.267, 0,

0.245, 0.057, 0,

0.343, 0.099, 0,

0.639, 0.161, 0,

0.657, 0.198, 0,

0.360, 0.370, 0,

0.593, 0.042, 0,

0.719, 0.103, 0, ]

dataX\_=np.array(dataset).reshape((17,3))

dataY=np.array([i for i in dataX\_[:,2:3]]).reshape(17)

beta\_=np.array([ 3.94723283, -9.4821469, 1.74809055])

for i in range(8,17):

dataX\_[i][2] = 1.0

train(beta\_,dataX\_,dataY)

test(beta\_,dataX\_,dataY)

##plot

plt.scatter(dataX\_[:8,:1].reshape(8),dataX\_[:8,1:2].reshape(8),marker='+',color='black',s=20,label='好瓜')

plt.scatter(dataX\_[8:,:1].reshape(9),dataX\_[8:,1:2].reshape(9),marker='x',color='red',s=20,label='坏瓜')

x=dataX\_[:,:1].reshape(17)

y=getploty(beta\_,x)

plt.plot(x,y,color='blue',label='分割线')

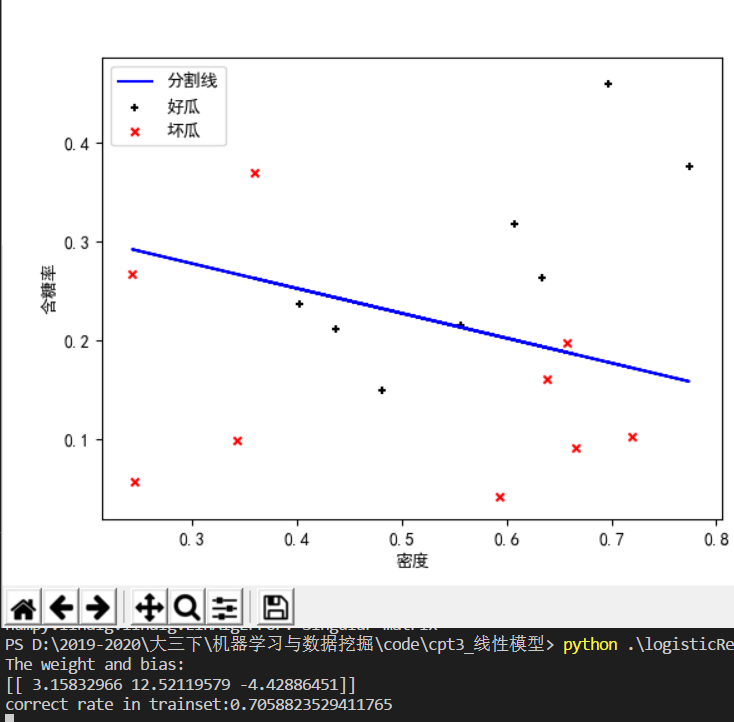
plt.legend(loc=2)

plt.xlabel('密度')

plt.ylabel('含糖率')

plt.show()

实验结果：



对数几率回归对西瓜数据集的训练所能达到的精度为0.706

**3.4** 选择两个UCI数据集，比较10折交叉验证法和留一法所估计的对率回归的错误率。

数据集描述：

data\_banknote.txt:

Attribute Information:

1. variance of Wavelet Transformed image (continuous)   
2. skewness of Wavelet Transformed image (continuous)   
3. curtosis of Wavelet Transformed image (continuous)   
4. entropy of image (continuous)   
5. class (integer)

Haberman\_data.txt:

Number of Attributes: 4 (including the class attribute)

Attribute Information:

1. Age of patient at time of operation (numerical)

2. Patient's year of operation (year - 1900, numerical)

3. Number of positive axillary nodes detected (numerical)

4. Survival status (class attribute)

1 = the patient survived 5 years or longer

2 = the patient died within 5 year

代码实现：

import numpy as np

class logitRegressioin:

def \_\_init\_\_(self):

self.X,self.Y = None,None

self.beta = None

self.filename = None

self.rate = 0.01

self.times = 250

def sigmod(self,z):

return 1/(1+np.exp(-z))

def gradDscent(self,X,Y):

total = np.zeros(X.shape[1])

for i in range(X.shape[0]):

p1 = self.sigmod(np.dot(X[i],self.beta))

total += X[i]\*(p1-Y[i])

return total

def train(self,X,Y):

for \_ in range(self.times):

self.beta -= self.rate\*self.gradDscent(X,Y)

def test(self,X,Y):

acc = 0

for i in range(X.shape[0]):

pi = self.sigmod(np.dot(self.beta,X[i]))

if abs(Y[i]-pi)<0.5:

acc+=1

return acc/X.shape[0]

def compare(self):

##10折交叉验证

step = self.X.shape[0]//10

acc = 0

for cnt in range(10):

testX = self.X[cnt\*step:(cnt+1)\*step]

testY = self.Y[cnt\*step:(cnt+1)\*step]

trainX = np.vstack((self.X[:cnt\*step],self.X[(cnt+1)\*step:]))

trainY = np.hstack((self.Y[:cnt\*step],self.Y[(cnt+1)\*step:]))

self.train(trainX,trainY)

acc += self.test(testX,testY)

print('the results of the ten-fold cross validation in '+self.filename+':')

print('accuracy: '+str(acc/10))

print('beta: ')

print(self.beta)

##留一法

self.beta = np.random.normal(0,1,self.X.shape[1])

acc=0

for cnt in range(self.X.shape[0]):

testX = self.X[cnt:cnt+1]

testY = self.Y[cnt:cnt+1]

trainX = np.vstack((self.X[:cnt],self.X[cnt+1:]))

trainY = np.hstack((self.Y[:cnt],self.Y[cnt+1:]))

self.train(trainX,trainY)

acc += self.test(testX,testY)

acc /= self.X.shape[0]

print('the result of the Leave-One-Out in '+self.filename+':')

print('accuracy: '+str(acc))

print('beta:')

print(self.beta)

def initial(self,filename):

self.filename = filename

X = np.loadtxt(filename,delimiter=',')

Y = np.array([X[i][-1] for i in range(X.shape[0])])

Y = Y-1 if filename=='haberman\_data.txt' else Y

X[:,-1:] = np.ones((X.shape[0],1))

self.X,self.Y = X,Y

self.normalization()

self.beta = np.random.normal(0,1,X.shape[1])

def normalization(self):

for col in range(self.X.shape[1]-1):

mu = np.mean(self.X[:,col:col+1])

sigma = np.sqrt(np.var(self.X[:,col:col+1]))

self.X[:,col:col+1] = (self.X[:,col:col+1]-mu)/sigma

if \_\_name\_\_ == '\_\_main\_\_':

log\_instance = logitRegressioin()

print('-----------------------------------------------------------------------')

log\_instance.initial('haberman\_data.txt')

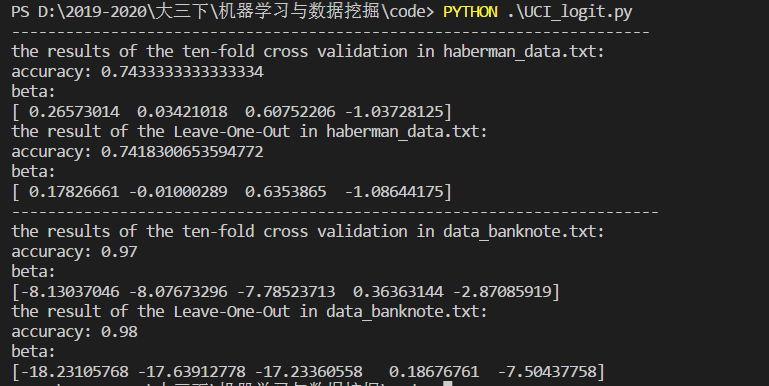
log\_instance.compare()

print('------------------------------------------------------------------------')

log\_instance.initial('data\_banknote.txt')

log\_instance.compare()

实验结果：



**3.5** 编程实现线性判别分析。并给出西瓜数据集上的结果。

代码实现：

import numpy as np

import matplotlib.pyplot as plt

plt.rcParams['font.sans-serif']=['SimHei']

plt.rcParams['axes.unicode\_minus'] = False

dataset=[0.697, 0.460, 1,

0.774, 0.376, 1,

0.634, 0.264, 1,

0.608, 0.318, 1,

0.556, 0.215, 1,

0.403, 0.237, 1,

0.481, 0.149, 1,

0.437, 0.211, 1,

0.666, 0.091, 0,

0.243, 0.267, 0,

0.245, 0.057, 0,

0.343, 0.099, 0,

0.639, 0.161, 0,

0.657, 0.198, 0,

0.360, 0.370, 0,

0.593, 0.042, 0,

0.719, 0.103, 0, ]

dataX=np.array(dataset).reshape((17,3))

dataY=np.array([i for i in dataX[:,2:3]]).reshape(17)

dataX=dataX[:,:2]

mean1,mean0 = np.mean(dataX[:8],axis=0),np.mean(dataX[8:],axis=0)

print('mean0, mean1:')

print(mean0,mean1)

mean1.reshape(1,2)

mean0.reshape(1,2)

# cov1=np.cov(np.array([dataX[:8,:1].reshape(8),dataX[:8,1:].reshape(8)]))

# cov0=np.cov(np.array([dataX[8:,:1].reshape(9),dataX[8:,1:].reshape(9)]))

# print(cov0+cov1)

# w = (np.mat(cov0+cov1)).I\*(mean0-mean1).reshape(2,1)

def sigma(x,mean):

total = np.zeros((2,2))

for i in range(x.shape[0]):

temp = (x[i]-mean).reshape(2,1)

total += temp\*temp.T

return total

def getSw(dataX,mean0,mean1):

Sw = sigma(dataX[:8],mean1)+sigma(dataX[8:],mean0)

print('Sw')

print(Sw)

return np.mat(Sw)

def getw(mean0,mean1,Sw):

return Sw.I\*(mean0-mean1).reshape(2,1)

def getplotxy(w\_):

x = np.linspace(0,0.8,30)

w\_=w\_.tolist()

y = -w\_[0][0]\*x/w\_[1][0]

return x,y

if \_\_name\_\_=='\_\_main\_\_':

Sw = getSw(dataX,mean0,mean1)

w\_ = getw(mean0,mean1,Sw)

print('w:')

print(w\_)

##plot

plt.scatter(dataX[:8,:1].reshape(8),dataX[:8,1:].reshape(8),marker='+',color='black',label='好瓜')

plt.scatter(dataX[8:,:1].reshape(9),dataX[8:,1:].reshape(9),marker='x',color='red',label='坏瓜')

x,y = getplotxy(w\_)

plt.plot(x,y,color='orange',label='投影线')

plt.legend(loc=2)

plt.xlabel('密度')

plt.ylabel('含糖率')

plt.show()

实验结果：

