

**机器学习**



**姓 名 王明瑞**

**学 号 2018211807**

**班 级 2018211406**

**学 院 现代邮政学院(自动化学院)**

**专 业 自动化**

**内 容 第三章作业**

**课 程 机器学习**

**队 友 赵浩然 2018211812**

**2021年 3月31日**

**目录**

[**1、数据集介绍** 1](#_Toc69074938)

[**2、Logistics回归模型** 3](#_Toc69074939)

[**2.1、Logistics回归模型介绍** 3](#_Toc69074940)

[**2.2、Logistics回归算法原理** 3](#_Toc69074941)

[**2.3、Logistics回归算法核心代码解释** 4](#_Toc69074942)

[**2.4、Logistics回归结果** 6](#_Toc69074943)

[**3、对率回归模型的检验** 8](#_Toc69074944)

[**3.1、乳腺癌“breast\_cancer”数据集测试** 9](#_Toc69074945)

[**3.2、糖尿病“Diabetes”数据集测试** 10](#_Toc69074946)

[**4、线性判别分析** 10](#_Toc69074947)

[**5、运行结果** 12](#_Toc69074948)

[**5.1、第一题** 12](#_Toc69074949)

[**5.2、第二题** 13](#_Toc69074950)

[**5.3、第三题** 14](#_Toc69074951)

[**6、代码** 14](#_Toc69074952)

[**6.3、题3.3** 14](#_Toc69074953)

[**6.3、题3.4** 17](#_Toc69074954)

[**6.3、题3.5** 23](#_Toc69074955)

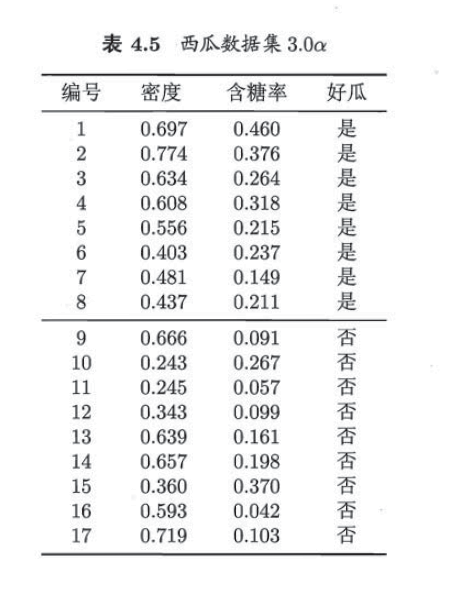
**第三章 线性模型**

**1、数据集介绍**

本次实验使用到三个数据集，分别是西瓜数据集3.0 ，UCI分类数据集中的糖尿病数据集“Diabetes.xls”和乳腺癌数据集“breast\_cancer.csv”。

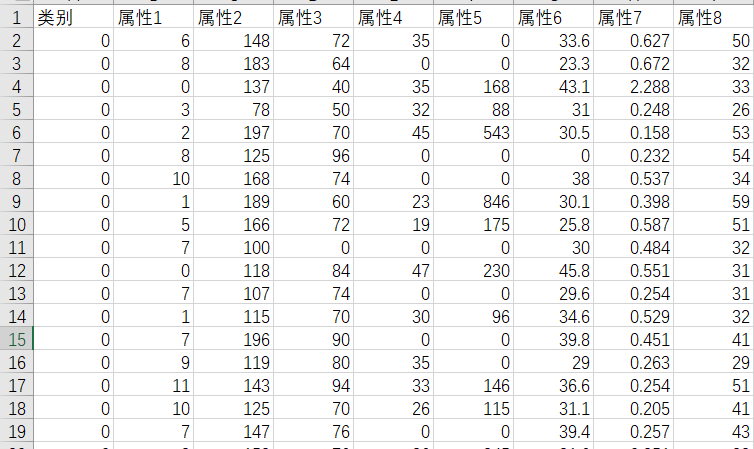
西瓜数据集3.0包含17条信息，每条信息对应西瓜的2种属性，给出了该西瓜是否为好瓜，“是”表示该西瓜是好瓜，“否”表该西瓜不是好瓜。西瓜数据集3.0的具体内容如下图所示。

表1 西瓜数据集3.0



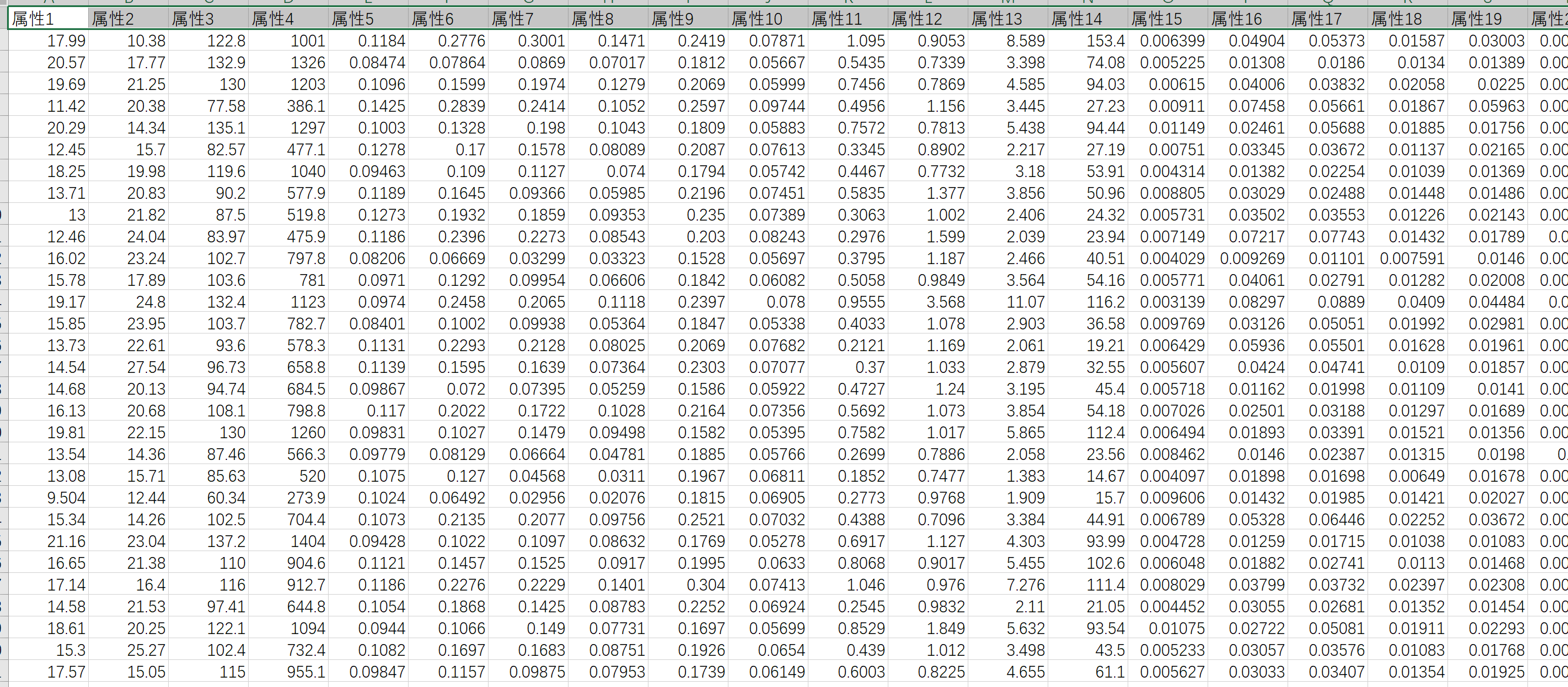
糖尿病“Diabetes”数据集共包含768条信息，每条信息对应一位可能患有糖尿病的患者的8个属性，并给出了该患者是否患有糖尿病的结果，“1”表示该患者确实患有糖尿病，“0”表该患者不患有糖尿病。“Diabetes”数据集的具体内容如下图所示。

表2 糖尿病“Diabetes”数据集



乳腺癌“breast\_cancer”数据集共包含568条信息，每条信息对应一位可能患有乳腺癌的患者的30个属性，并给出了该患者是否患有乳腺癌的结果，“1”表示该患者确实患有乳腺癌，“0”表该患者不患有乳腺癌。乳腺癌“breast\_cancer”数据集的具体内容如下所示：

表3 乳腺癌“breast\_cancer”数据集



**2、Logistics回归模型**

**2.1、Logistics回归模型介绍**

Logistic Regression是经典的分类模型，常用于二分类。Logistics回归可以认为是一个被Sigmoid函数（logistic方程）所归一化后的线性回归模型。Logistic 回归的本质是：假设数据服从这个分布，然后使用极大似然估计做参数的估计。

**2.2、Logistics回归算法原理**

Logistic函数（或称为Sigmoid函数），函数形式为：

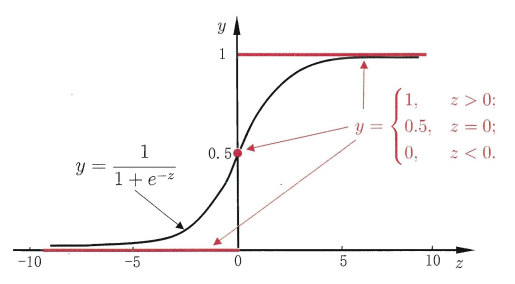


图2.2.1 Logistic函数

对于线性边界的情况，边界形式如下：

其中，训练数据为向量，最佳参数，构造预测函数为：

基于最大似然估计推导得到代价函数J：

构造整体代价函数Cost为：

使用梯度下降法求解Cost的最优解：

梯度下降法算法流程：

1. 初始化W
2. 更新W：
3. 迭代达到一定的次数或一定阈值

**2.3、Logistics回归算法核心代码解释**

1. 定义Sigmoid函数：

1. def sigmoid(x, W):
2. return 1.0 / (1.0 + np.exp(-x \* W))

2. 梯度下降法求解logistics回归权重W

1. *# logistics回归，返回W权重*
2. def logistic\_regression(train\_X, labels, alpha=0.01, max\_iter=1001):
3. X = np.mat(train\_X)
4. Y = np.mat(labels).T *# 转置为列向量*
5. m, n = np.shape(X)
6. *# 随机初始化W*
7. W = np.mat(np.random.randn(n, 1))
8. *# 更新W*
9. for i in range(max\_iter):
10. H = sigmoid(X, W)
11. dW = X.T \* (H - Y)  *# dW:(3,1）根据梯度下降算法，需要先求得dCost/dW，此处用dW代替*
12. W -= alpha \* dW  *# 梯度下降 W = W - alpha\*dCost/dW*
13. return W

在梯度下降法求解logistics回归权重W的函数中，我们将训练集train\_X和真实分类情况labels进行了矩阵化处理，得到X，Y矩阵。注意此时X矩阵为X=[X, 1]，即为X多添加了一列全1列向量，方便计算，如下所示。

W更新部分的代码参照梯度下降法推导的公式编写，，推广到矩阵形式即为

最后函数返回W即为logistics回归模型的W参数。

3. 结果预测函数

1. *# 结果输出函数*
2. def predict(X, W):
3. m = len(X)
4. pred = np.zeros(m)
5. for i in range(m):
6. if sigmoid(X[i], W) > 0.5:      *# 使用sigmoid判断，大于0.5label为1，否则为0*
7. pred[i] = 1
8. return pred

将训练好的模型参数W和测试集X输入到函数，根据sigmoid函数进行划分，大于0.5为1，否则为0，可以输出预测结果。

4. 数据可视化处理

1. *# 数据可视化*
2. def show\_diagram(train\_X, labels, W):
3. w1 = W[0, 0]
4. w2 = W[1, 0]
5. b = W[2, 0]
6. plot\_x1 = np.arange(0, 1, 0.01)
7. plot\_x2 = -w1 / w2 \* plot\_x1 - b / w2
8. plt.plot(plot\_x1, plot\_x2, c='r', label='decision boundary')
9. plt.title('watermelon\_3a')
10. plt.xlabel('density')
11. plt.ylabel('ratio\_sugar')
12. plt.scatter(train\_X[labels == 0, 0].A, train\_X[labels == 0, 1].A, marker='^', color='r', s=80, label='bad')
13. plt.scatter(train\_X[labels == 1, 0].A, train\_X[labels == 1, 1].A, marker='o', color='g', s=80, label='good')
14. plt.legend(loc='upper right')
15. plt.show()

根据数据集绘制散点图，并根据分类模型的结果绘制决策边界，，上式中x1对应于横坐标，x2对用于纵坐标，决策边界z=0，因此得到直线方程是：

**2.4、Logistics回归结果**

输出可视化图形：

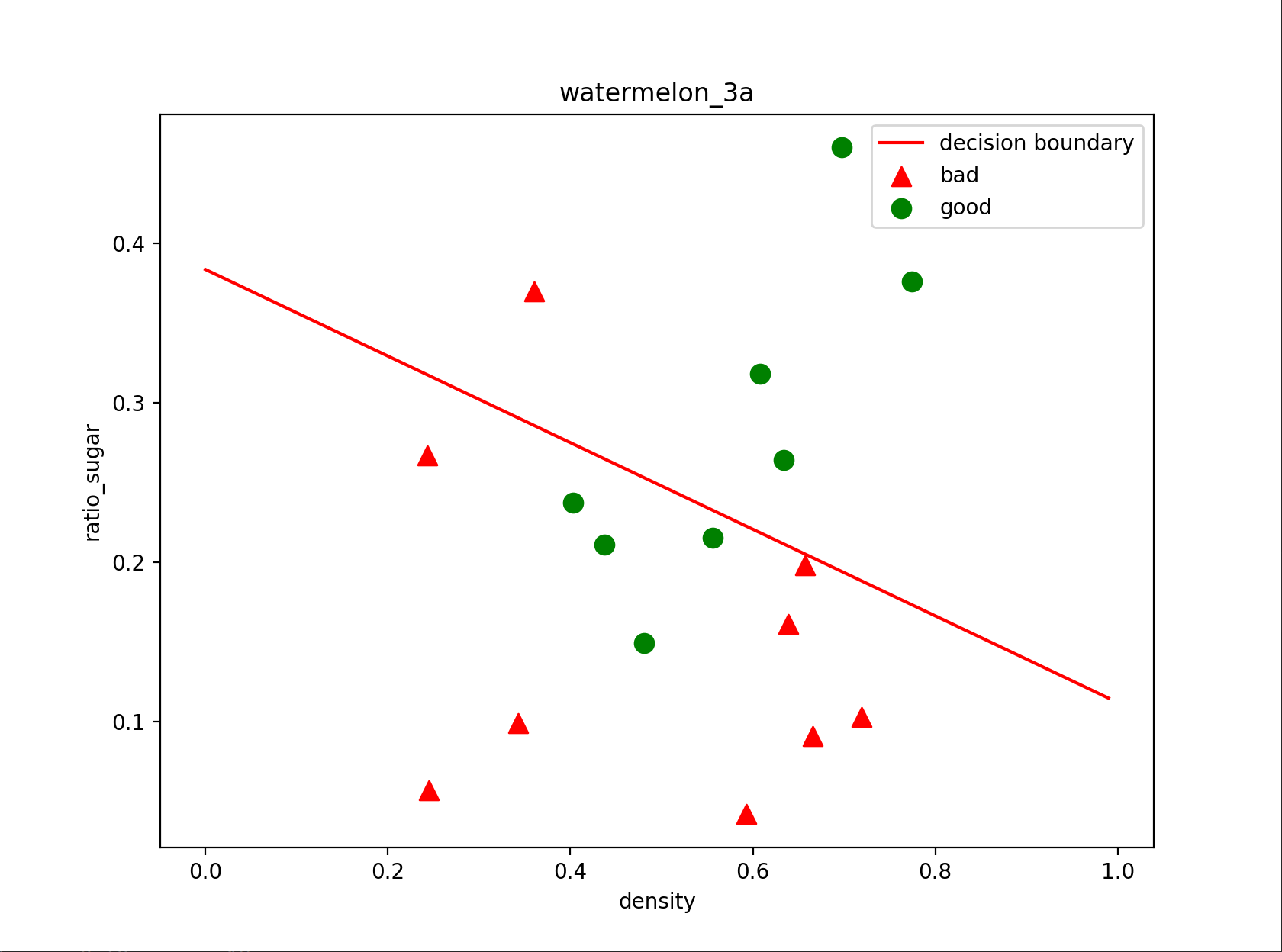


图2.4.1 logistics回归结果

可以看到，绿色代表good，红色代表bad，经过决策边界划分后，区域被分为两部分，代表二分类结果。

输出结果部分：

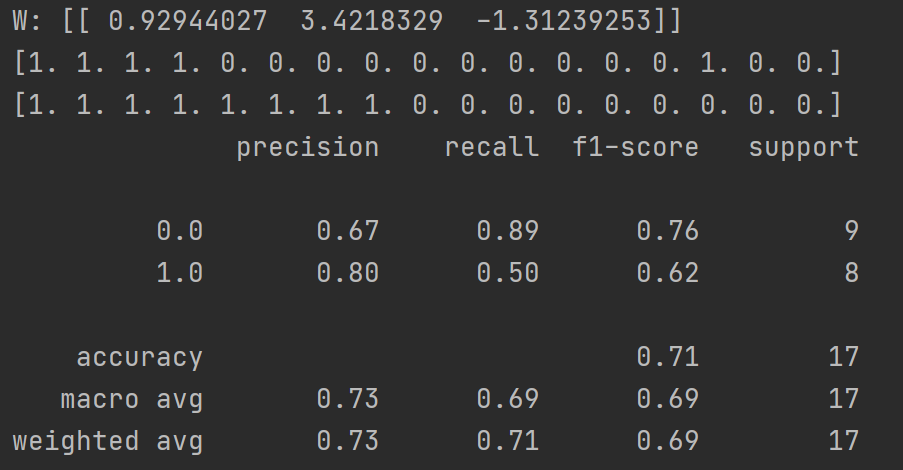


图2.4.2 logistics回归结果数据

logistics回归最终的输出结果，前两行分别为预测结果和真实结果，下面为二分类的性能评估，准确率在71%，梯度下降模型做的还不够完美，如果使用随机梯度下降的话，结果可能会更好一些。

**3、对率回归模型的检验**

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。

10折交叉验证法函数：

1. def K\_fold(train\_X, labels, splits=10):
2. order\_id = []
3. total\_acc = 0
4. sfolder = StratifiedKFold(n\_splits=splits, shuffle=True)  *# 十折交叉验证划分数据集*
5. for num, (train, test) in enumerate(sfolder.split(train\_X, labels)):
6. x\_train = train\_X[train, :]
7. y\_train = labels[train]
8. x\_test = train\_X[test, :]
9. y\_test = labels[test]
10. order\_id.extend(test)
11. *# 开始进行logistics回归分类训练*
12. W = logistics.logistic\_regression(x\_train, y\_train, alpha=0.01, max\_iter=1001)  *# 计算权重W*
13. y\_pred = logistics.predict(x\_test, W)  *# 根据训练好的模型进行预测并输出预测值*
14. acc = accuracy\_score(y\_test, y\_pred)
15. total\_acc += acc
16. print('第', num + 1, '折验证的错误率', 1 - acc)
17. print("十折交叉验证的平均错误率为：", 1 - total\_acc / splits)

这部分算法在上次作业中已经详细解释过，在本部分不再赘述

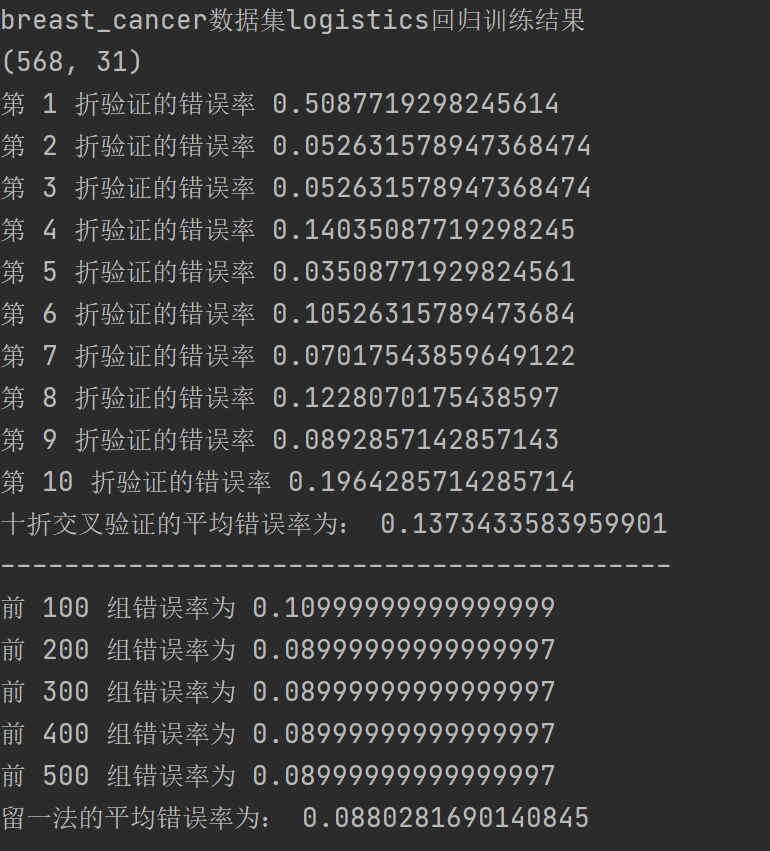
留一法函数：

1. def leave\_one(train\_X, labels):
2. loo = LeaveOneOut()
3. total\_acc = 0
4. loo.get\_n\_splits(train\_X)
5. num = 0
6. for train\_index, test\_index in loo.split(train\_X, labels):
7. x\_train, x\_test = train\_X[train\_index], train\_X[test\_index]
8. y\_train, y\_test = labels[train\_index], labels[test\_index]
9. *# 开始进行logistics回归分类训练*
10. W = logistics.logistic\_regression(x\_train, y\_train, alpha=0.01, max\_iter=1001)  *# 计算权重W*
11. y\_pred = logistics.predict(x\_test, W)   *# 根据训练好的模型进行预测并输出预测值*
12. acc = accuracy\_score(y\_test, y\_pred)
13. total\_acc += acc
14. num += 1
15. if num % 100 == 0:
16. print("前", num, "组错误率为", 1 - (total\_acc / num))
17. print("留一法的平均错误率为：", 1 - (total\_acc / num))

如果数据集D的大小为N,那么用N-1条数据进行训练，用剩下的一条数据作为验证。用一条数据作为验证的坏处就是可能和相差很大，所以在留一交叉验证里，每次从D中取一组作为验证集，直到所有样本都作过验证集，共计算N次，最后对验证误差求平均，这种方法称之为留一法交叉验证。

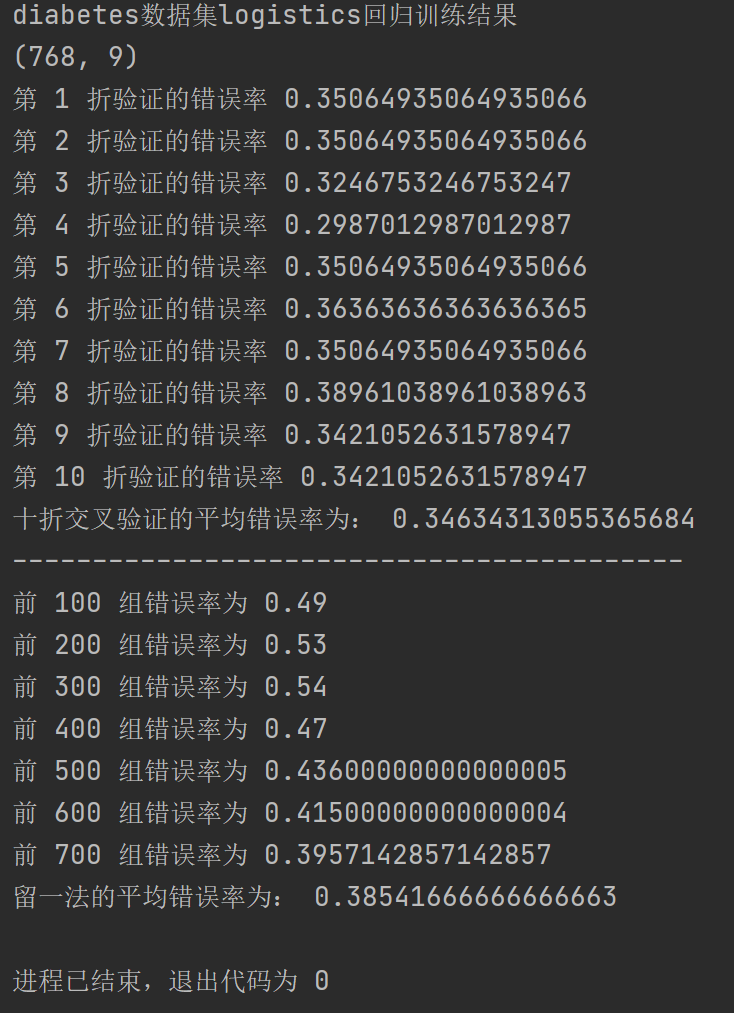
**3.1、乳腺癌“breast\_cancer”数据集测试**

带入logistics回归模型后得到的10折交叉验证法和留一法所估计的对率回归的错误率结果输出如下：



**3.2、糖尿病“Diabetes”数据集测试**

带入logistics回归模型后得到的10折交叉验证法和留一法所估计的对率回归的错误率结果输出如下：



**4、线性判别分析**

线性判别分析是一种经典的线性学习方法，亦称“Fisher判别分析”。LDA的思想较为朴素：给定训练样例集，设法将阳历投影到一条直线上，使得同类样例的投影点尽可能接近、异类样例的投影点尽可能远离；在队新样本进行分类时，将其投影到同样的这条直线上，再根据投影点的位置来确定新样本的类别。

给定数据集，令分别表示第类示例的集合、均值向量、协方差矩阵。若将数据投影到直线上，则两类样本的中心在直线上的投影分别为和；若将所有样本点都投影到直线上，则两类样本的协方差分别为和。由于直线式一维空间，因此和均为试数。

故最大化目标

定义“类内散度矩阵”

以及“类间散度矩阵”

结合“拉格朗日乘子法”，可得

在代码实现的过程中

1. mean1 = np.array([np.mean(x1[:, 0]), np.mean(x1[:, 1])])
2. mean2 = np.array([np.mean(x2[:, 0]), np.mean(x2[:, 1])])
3. Sw = np.zeros((2, 2))
4. **for** i **in** range(x1.shape[0]):
5. Sw = calculate(Sw, x1[i, :] - mean1)
6. **for** i **in** range(x2.shape[0]):
7. Sw = calculate(Sw, x2[i, :] - mean2)
8. w = np.linalg.inv(Sw) @ (mean1 - mean2).transpose()

变量mean1和mean2则表示两类样例的均值向量，Sw则表示计算出的“类间散度矩阵，进而就计算出矩阵系数向量w。由于我们仅关心样例点投影到直线后的投影点直接的距离关系，所以只计算出一条过原点的直线，直线沿垂直方向移动并不影响投影结果。

**5、运行结果**

**5.1、第一题**

输出可视化图形：

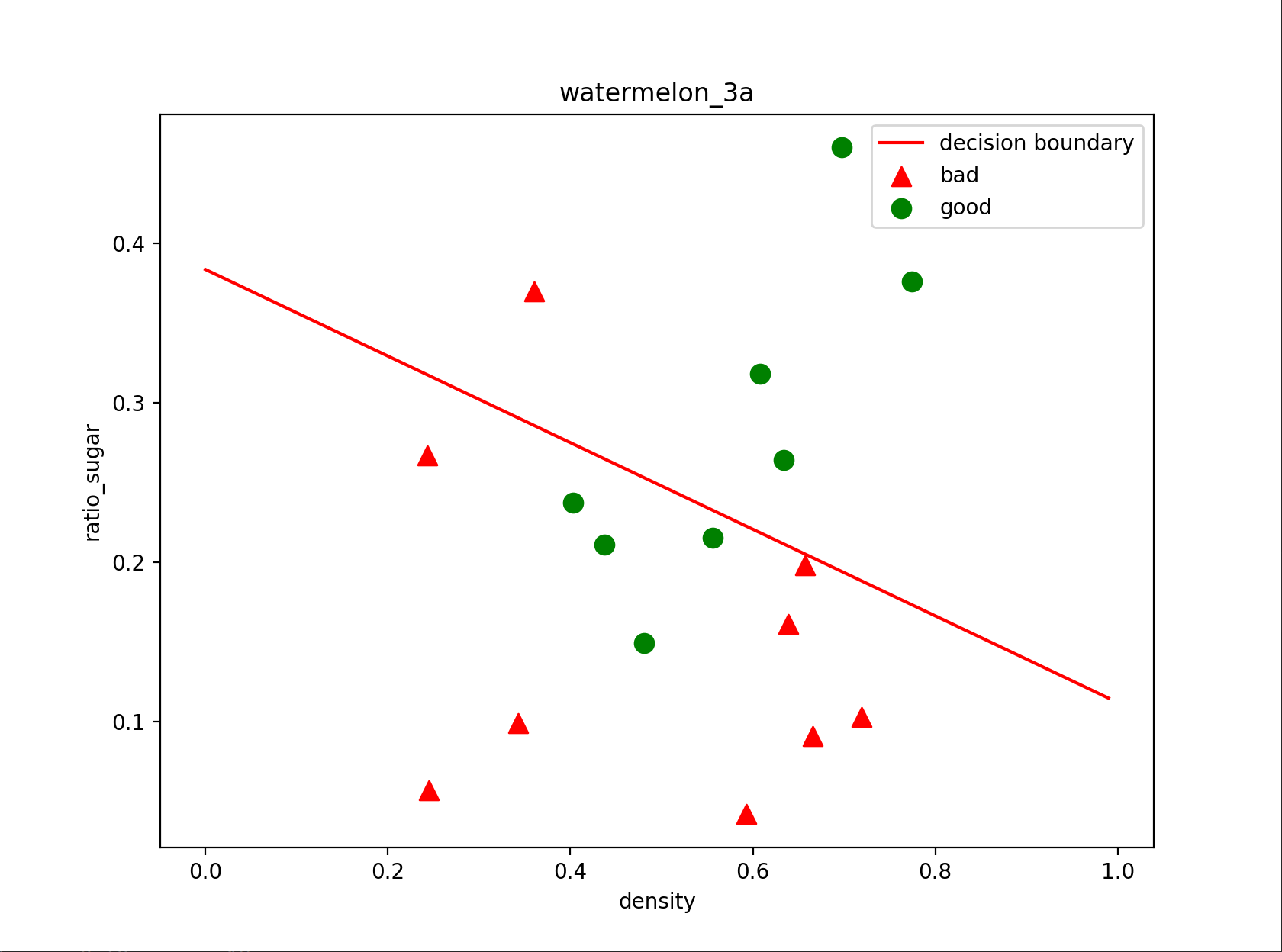


图2.4.1 logistics回归结果

可以看到，绿色代表good，红色代表bad，经过决策边界划分后，区域被分为两部分，代表二分类结果。

输出结果部分：

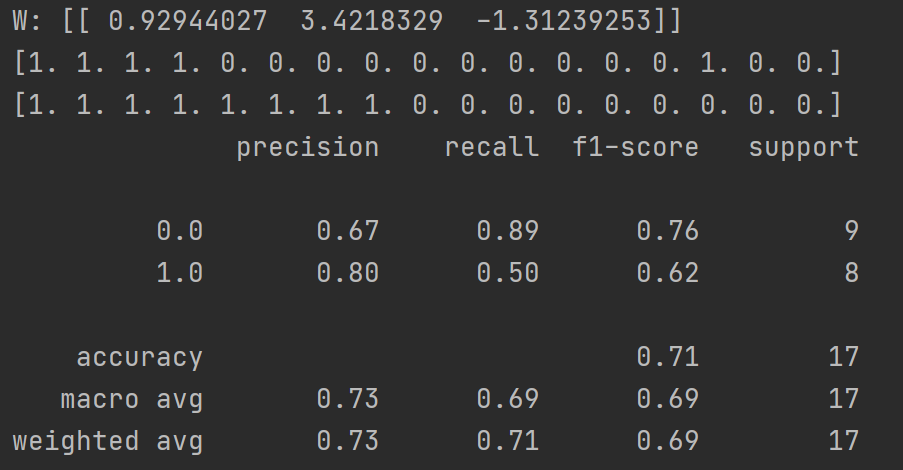
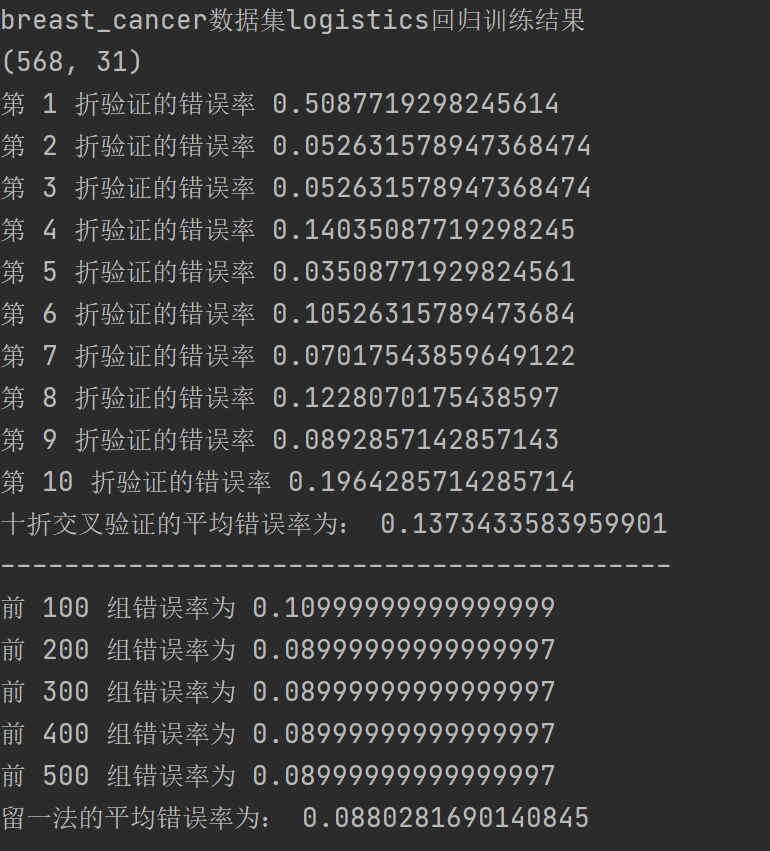


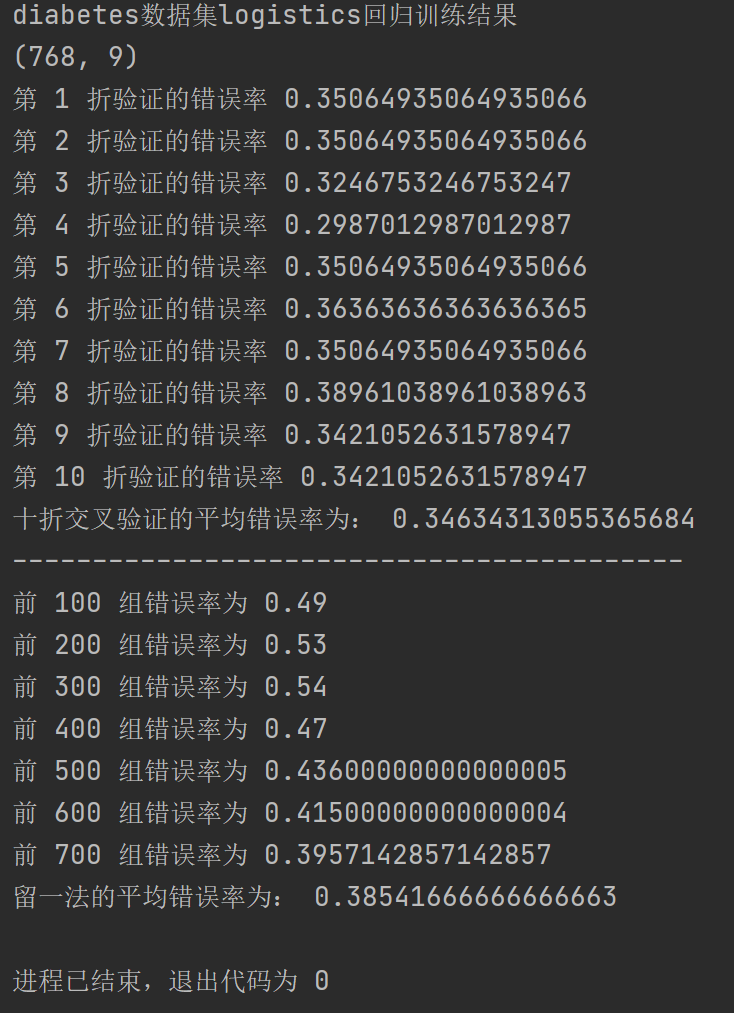
图2.4.2 logistics回归结果数据

**5.2、第二题**

将UCI乳腺癌数据集带入logistics回归模型后得到的10折交叉验证法和留一法所估计的对率回归的错误率结果输出如下

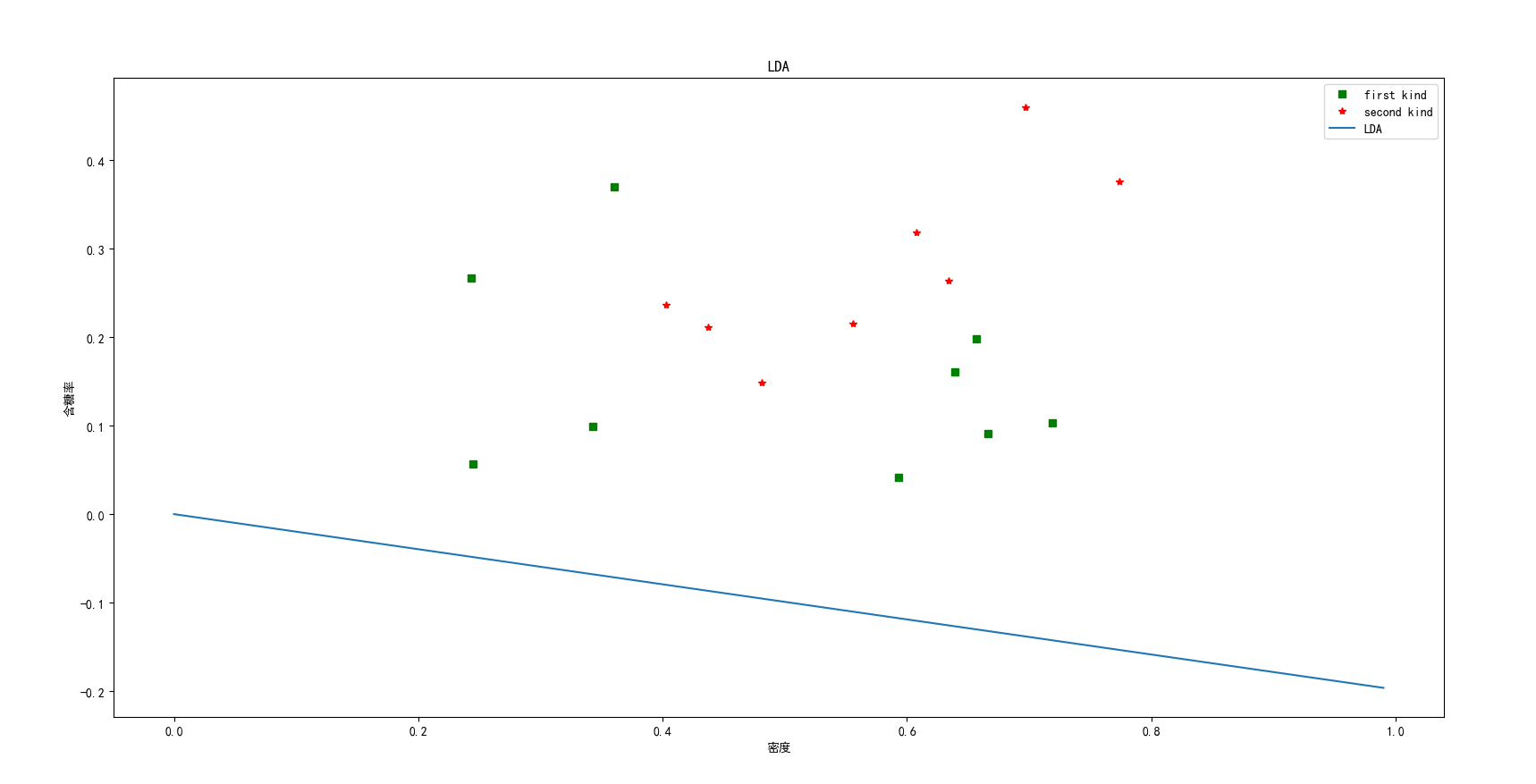


将UCI糖尿病数据集带入logistics回归模型后得到的10折交叉验证法和留一法所估计的对率回归的错误率结果输出如下



**5.3、第三题**

线性判别分析结果为



线性判别分析的直线为



即为

由线性判别分析结果图可以看出，该数据集使用“线性判别分析”进行降维的结果并不理想。

**6、代码**

**6.3、题3.3**

# -\*- coding: utf-8 -\*

# @File: logistics.py

import numpy as np

import matplotlib.pyplot as plt

from sklearn import metrics

# 读入文件

def load\_data(filename):

dataset = np.loadtxt(filename, encoding='utf-8', skiprows=1)

temp = np.mat(dataset[:, 1:3])

train\_X = np.c\_[temp, np.ones(len(temp))] # 增加一列1，用于计算β

labels = dataset[:, -1]

return train\_X, labels

def sigmoid(x, W):

return 1.0 / (1.0 + np.exp(-x \* W))

# logistics回归，返回W权重

def logistic\_regression(train\_X, labels, alpha=0.01, max\_iter=1001):

X = np.mat(train\_X)

Y = np.mat(labels).T

m, n = np.shape(X)

# 随机初始化W

W = np.mat(np.random.randn(n, 1))

w\_save = []

# 更新W

for i in range(max\_iter):

H = sigmoid(X, W)

dW = X.T \* (H - Y) # dW:(3,1）根据梯度下降算法，需要先求得dCost/dW，此处用dW代替

W -= alpha \* dW # 梯度下降 W = W - alpha\*dCost/dW

return W

# 数据可视化

def show\_diagram(train\_X, labels, W):

w1 = W[0, 0]

w2 = W[1, 0]

b = W[2, 0]

plot\_x1 = np.arange(0, 1, 0.01)

plot\_x2 = -w1 / w2 \* plot\_x1 - b / w2

plt.plot(plot\_x1, plot\_x2, c='r', label='decision boundary')

plt.title('watermelon\_3a')

plt.xlabel('density')

plt.ylabel('ratio\_sugar')

plt.scatter(train\_X[labels == 0, 0].A, train\_X[labels == 0, 1].A, marker='^', color='r', s=80, label='bad')

plt.scatter(train\_X[labels == 1, 0].A, train\_X[labels == 1, 1].A, marker='o', color='g', s=80, label='good')

plt.legend(loc='upper right')

plt.show()

# 结果输出函数

def predict(X, W):

m = len(X)

pred = np.zeros(m)

for i in range(m):

if sigmoid(X[i], W) > 0.5: # 使用sigmoid判断，大于0.5label为1，否则为0

pred[i] = 1

return pred

def main():

# 加载数据集

train\_X, labels = load\_data('watermelon\_3a.txt')

# 对数据进行logistics回归分类

W = logistic\_regression(train\_X, labels)

print('W:', W.T)

# 将数据集带入模型预测

y\_pred = predict(train\_X, W)

print(y\_pred)

print(labels)

# 输出二分类的性能评估

print(metrics.classification\_report(labels, y\_pred))

# 可视化

show\_diagram(train\_X, labels, W)

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

main()

**6.3、题3.4**

# -\*- coding: utf-8 -\*

# @File: 3.4.1.py

import logistics # 导入已经习题3.3写好的logistics文件，直接调用

import numpy as np

import pandas as pd

import warnings

from sklearn.metrics import accuracy\_score

from sklearn.model\_selection import StratifiedKFold

from sklearn.model\_selection import LeaveOneOut

warnings.filterwarnings('ignore')

# 读入文件

def load\_data(filename):

df = pd.read\_csv(filename)

dataset = df.values

print(np.shape(dataset))

temp = np.mat(dataset[:, 0:30])

train\_X = np.c\_[temp, np.ones(len(temp))] # 增加一列1，用于计算β

labels = dataset[:, -1]

return train\_X, labels

def leave\_one(train\_X, labels):

loo = LeaveOneOut()

total\_acc = 0

loo.get\_n\_splits(train\_X)

num = 0

for train\_index, test\_index in loo.split(train\_X, labels):

x\_train, x\_test = train\_X[train\_index], train\_X[test\_index]

y\_train, y\_test = labels[train\_index], labels[test\_index]

# 开始进行logistics回归分类训练

W = logistics.logistic\_regression(x\_train, y\_train, alpha=0.01, max\_iter=1001) # 计算权重W

y\_pred = logistics.predict(x\_test, W) # 根据训练好的模型进行预测并输出预测值

acc = accuracy\_score(y\_test, y\_pred)

total\_acc += acc

num += 1

if num % 100 == 0:

print("前", num, "组错误率为", 1 - (total\_acc / num))

print("留一法的平均错误率为：", 1 - (total\_acc / num))

def K\_fold(train\_X, labels, splits=10):

order\_id = []

total\_acc = 0

sfolder = StratifiedKFold(n\_splits=splits, shuffle=True) # 十折交叉验证划分数据集

for num, (train, test) in enumerate(sfolder.split(train\_X, labels)):

x\_train = train\_X[train, :]

y\_train = labels[train]

x\_test = train\_X[test, :]

y\_test = labels[test]

order\_id.extend(test)

# 开始进行logistics回归分类训练

W = logistics.logistic\_regression(x\_train, y\_train, alpha=0.01, max\_iter=1001) # 计算权重W

y\_pred = logistics.predict(x\_test, W) # 根据训练好的模型进行预测并输出预测值

acc = accuracy\_score(y\_test, y\_pred)

total\_acc += acc

print('第', num + 1, '折验证的错误率', 1 - acc)

print("十折交叉验证的平均错误率为：", 1 - (total\_acc / splits))

def main():

x, y = load\_data('breast\_cancer.csv') # 读取文件

K\_fold(x, y)

print("------------------------------------------")

leave\_one(x, y)

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

print("breast\_cancer数据集logistics回归训练结果")

main()

# -\*- coding: utf-8 -\*

# @File: 3.4.2.py

import logistics

import numpy as np

import pandas as pd

import warnings

from sklearn.metrics import accuracy\_score

from sklearn.model\_selection import StratifiedKFold

from sklearn.model\_selection import LeaveOneOut

warnings.filterwarnings('ignore')

# 读入文件

def load\_data(filename):

df = pd.read\_excel(filename)

dataset = df.values

print(np.shape(dataset))

temp = np.mat(dataset[:, 1:])

train\_X = np.c\_[temp, np.ones(len(temp))] # 增加一列1，用于计算β

labels = dataset[:, 0]

return train\_X, labels

def leave\_one(train\_X, labels):

loo = LeaveOneOut()

total\_acc = 0

loo.get\_n\_splits(train\_X)

num = 0

for train\_index, test\_index in loo.split(train\_X, labels):

x\_train, x\_test = train\_X[train\_index], train\_X[test\_index]

y\_train, y\_test = labels[train\_index], labels[test\_index]

# 开始进行logistics回归分类训练

W = logistics.logistic\_regression(x\_train, y\_train, alpha=0.01, max\_iter=1000) # 计算权重W

y\_pred = logistics.predict(x\_test, W) # 根据训练好的模型进行预测并输出预测值

acc = accuracy\_score(y\_test, y\_pred)

total\_acc += acc

num += 1

if num % 100 == 0:

print("前", num, "组错误率为", 1 - (total\_acc / num))

print("留一法的平均错误率为：", 1 - (total\_acc / num))

def K\_fold(train\_X, labels, splits=10):

order\_id = []

total\_acc = 0

sfolder = StratifiedKFold(n\_splits=splits, shuffle=True) # 十折交叉验证划分数据集

for num, (train, test) in enumerate(sfolder.split(train\_X, labels)):

x\_train = train\_X[train, :]

y\_train = labels[train]

x\_test = train\_X[test, :]

y\_test = labels[test]

order\_id.extend(test)

# 开始进行logistics回归分类训练

W = logistics.logistic\_regression(x\_train, y\_train, alpha=0.01, max\_iter=1001) # 计算权重W

y\_pred = logistics.predict(x\_test, W) # 根据训练好的模型进行预测并输出预测值

acc = accuracy\_score(y\_test, y\_pred)

total\_acc += acc

print('第', num + 1, '折验证的错误率', 1 - acc)

print("十折交叉验证的平均错误率为：", 1 - (total\_acc / splits))

def main():

x, y = load\_data('Diabetes.xls') # 读取文件

K\_fold(x, y)

print("------------------------------------------")

leave\_one(x, y)

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

print("diabetes数据集logistics回归训练结果")

main()

**6.3、题3.5**

import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
  
  
def calculate(sw, x):  
 sw[0, 0] += x[0] \* x[0]  
 sw[0, 1] += x[0] \* x[1]  
 sw[1, 0] += x[1] \* x[0]  
 sw[1, 1] += x[1] \* x[1]  
 return sw  
  
  
def LDA\_w(df):  
 df\_labels = df.iloc[:, -1]  
 labels = list(set(df\_labels.values))  
 index\_1 = []  
 index\_2 = []  
 for i in range(df.shape[0]):  
 if df.iloc[i, -1] == labels[0]:  
 index\_1.append(i)  
 else:  
 index\_2.append(i)  
 df1 = df.iloc[index\_1, :]  
 df2 = df.iloc[index\_2, :]  
 x1 = df1.values[:, 1:3]  
 x2 = df2.values[:, 1:3]  
 mean1 = np.array([np.mean(x1[:, 0]), np.mean(x1[:, 1])])  
 mean2 = np.array([np.mean(x2[:, 0]), np.mean(x2[:, 1])])  
 Sw = np.zeros((2, 2))  
 for i in range(x1.shape[0]):  
 Sw = calculate(Sw, x1[i, :] - mean1)  
 for i in range(x2.shape[0]):  
 Sw = calculate(Sw, x2[i, :] - mean2)  
 w = np.linalg.inv(Sw) @ (mean1 - mean2).transpose()  
 return w  
  
  
def LDA\_plot(df, w):  
 df\_labels = df.iloc[:, -1]  
 labels = list(set(df\_labels.values))  
 x1 = []  
 y1 = []  
 x2 = []  
 y2 = []  
 for i in range(df.shape[0]):  
 if df.iloc[i, -1] == labels[0]:  
 x1.append(df.iloc[i, 1])  
 y1.append(df.iloc[i, 2])  
 else:  
 x2.append(df.iloc[i, 1])  
 y2.append(df.iloc[i, 2])  
 plt.plot(x1, y1, 'gs', label="first kind")  
 plt.plot(x2, y2, 'r\*', label="second kind")  
 x = np.arange(0, 1, 0.01)  
 y = np.array((-w[0] \* x) / w[1])  
 plt.plot(x, y, label="LDA")  
 plt.rcParams['font.sans-serif'] = ['SimHei']  
 plt.rcParams['axes.unicode\_minus'] = False  
 plt.xlabel(df.columns[1])  
 plt.ylabel(df.columns[2])  
 plt.title('LDA')  
 plt.legend(loc='best')  
 plt.show()  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 io = 'D://西瓜数据集3.0a.xlsx'  
 dataframe = pd.read\_excel(io)  
 w = LDA\_w(dataframe)  
 LDA\_plot(dataframe, w)  
 print('LDA曲线为：', w[0], '\*x ', w[1], '\*y = 0')