# 4.2

import numpy as np

import pandas as pd

from scipy.stats import shapiro

from statsmodels.multivariate.manova import MANOVA

# 数据

data = {

'耐用性1': [194, 208, 233, 241, 265, 269, 239, 189, 224, 243, 243, 226],

'耐用性2': [192, 188, 217, 222, 252, 283, 127, 105, 123, 123, 117, 125],

'耐用性3': [141, 165, 171, 201, 207, 191, 90, 85, 79, 110, 100, 75],

'品牌': ['甲']\*6 + ['乙']\*6

}

# 创建 DataFrame

df = pd.DataFrame(data)

# 检查数据是否有重复行

df = df.drop\_duplicates()

# 多元正态性检验

for col in ['耐用性1', '耐用性2', '耐用性3']:

stat, p = shapiro(df[col])

print(f'{col} 正态性检验 p值: {p}')

# MANOVA 分析

maov = MANOVA.from\_formula('耐用性1 + 耐用性2 + 耐用性3 ~ 品牌', data=df)

print(maov.mv\_test())

# 计算协方差矩阵

# 分别计算品牌甲和品牌乙的协方差矩阵

cov\_matrix\_A = np.cov(df[df['品牌'] == '甲'][['耐用性1', '耐用性2', '耐用性3']], rowvar=False)

cov\_matrix\_B = np.cov(df[df['品牌'] == '乙'][['耐用性1', '耐用性2', '耐用性3']], rowvar=False)

print("甲协方差矩阵：")

print(cov\_matrix\_A)

print("乙协方差矩阵：")

print(cov\_matrix\_B)

# 计算组合协方差矩阵

s\_p = (5 \* cov\_matrix\_A + 5 \* cov\_matrix\_B) / 10

print("组合协方差矩阵：")

print(s\_p)

# 计算逆矩阵

s\_p\_inv = np.linalg.inv(s\_p)

print("组合协方差矩阵的逆矩阵：")

print(s\_p\_inv)

# 计算结果

x\_y = np.array([[7.667], [105.67], [89.5]])

result = 3 \* np.dot(np.dot(x\_y.T, s\_p\_inv), x\_y)

print("结果：")

print(result)

# 计算 fx1, fx2, fx3

T2x1 = 3 \* 7.667 \* 0.0094 \* 7.667

print(f"T2x1: {T2x1}")

T2x2 = 3 \* 105.67 \* 0.005420 \* 105.67

print(f"T2x2: {T2x2}")

T2x3 = 3 \* 89.5 \* 0.00687788 \* 89.5

print(f"T2x3: {T2x3}")

## 结果：

耐用性1 正态性检验 p值: 0.5954723913661737

耐用性2 正态性检验 p值: 0.08487800487321405

耐用性3 正态性检验 p值: 0.10326631936280978

Multivariate linear model

=============================================================

-------------------------------------------------------------

Intercept Value Num DF Den DF F Value Pr > F

-------------------------------------------------------------

Wilks' lambda 0.0120 3.0000 8.0000 219.1355 0.0000

Pillai's trace 0.9880 3.0000 8.0000 219.1355 0.0000

Hotelling-Lawley trace 82.1758 3.0000 8.0000 219.1355 0.0000

Roy's greatest root 82.1758 3.0000 8.0000 219.1355 0.0000

-------------------------------------------------------------

-------------------------------------------------------------

品牌 Value Num DF Den DF F Value Pr > F

-------------------------------------------------------------

Wilks' lambda 0.0266 3.0000 8.0000 97.4171 0.0000

Pillai's trace 0.9734 3.0000 8.0000 97.4171 0.0000

Hotelling-Lawley trace 36.5314 3.0000 8.0000 97.4171 0.0000

Roy's greatest root 36.5314 3.0000 8.0000 97.4171 0.0000

=============================================================

甲协方差矩阵：

[[ 901.2 1026.4 666.4 ]

[1026.4 1324.26666667 644.13333333]

[ 666.4 644.13333333 623.06666667]]

乙协方差矩阵：

[[421.86666667 128. 143.66666667]

[128. 65.2 -0.6 ]

[143.66666667 -0.6 174.16666667]]

组合协方差矩阵：

[[661.53333333 577.2 405.03333333]

[577.2 694.73333333 321.76666667]

[405.03333333 321.76666667 398.61666667]]

组合协方差矩阵的逆矩阵：

[[ 0.00943305 -0.00542679 -0.00520435]

[-0.00542679 0.00542085 0.00113839]

[-0.00520435 0.00113839 0.00687788]]

结果：

[[365.32486146]]

T2x1: 1.6576774697999999

T2x2: 181.561581114

T2x3: 165.28061481

# 4.10

import numpy as np

import pandas as pd

# 数据

data = {

'左耳长度': [59, 60, 58, 59, 50, 59, 62, 63, 68, 63, 66, 56, 62, 66, 65, 61, 60, 60, 58, 58,

70, 69, 65, 62, 59, 55, 60, 58, 65, 67, 60, 53, 66, 60, 59, 58, 60, 54, 62, 59,

63, 56, 62, 59, 62, 50, 63, 61, 55, 63, 65, 64, 65, 67, 55, 56, 65, 62, 55, 58],

'右耳长度': [59, 65, 62, 59, 48, 65, 62, 62, 72, 66, 63, 56, 64, 68, 66, 60, 64, 57, 60, 59,

69, 68, 65, 60, 56, 58, 58, 64, 67, 62, 57, 55, 65, 53, 58, 54, 56, 59, 66, 61,

63, 57, 62, 58, 58, 57, 63, 62, 59, 63, 70, 64, 65, 67, 55, 56, 67, 65, 61, 58],

'组别': [1]\*20 + [2]\*20 + [3]\*20

}

df = pd.DataFrame(data)

# 计算每组的均值向量

group\_means = df.groupby('组别').mean()

overall\_mean = df[['左耳长度', '右耳长度']].mean().values

# 计算类内散布矩阵 S\_W

S\_W = np.zeros((2, 2))

for group in df['组别'].unique():

group\_data = df[df['组别'] == group][['左耳长度', '右耳长度']].values

group\_mean = group\_means.loc[group].values

S\_W += np.dot((group\_data - group\_mean).T, (group\_data - group\_mean))

# 计算类间散布矩阵 S\_B

S\_B = np.zeros((2, 2))

for group in df['组别'].unique():

n\_group = len(df[df['组别'] == group])

group\_mean = group\_means.loc[group].values

S\_B += n\_group \* np.dot((group\_mean - overall\_mean).reshape(2, 1), (group\_mean - overall\_mean).reshape(1, 2))

# 计算特征值和特征向量

eig\_vals, eig\_vecs = np.linalg.eig(np.linalg.inv(S\_W).dot(S\_B))

# 选择最大的特征值对应的特征向量

eig\_pairs = [(np.abs(eig\_vals[i]), eig\_vecs[:, i]) for i in range(len(eig\_vals))]

eig\_pairs = sorted(eig\_pairs, key=lambda k: k[0], reverse=True)

# 打印前两个特征值及其对应的特征向量

print("前两个特征值及其对应的特征向量：")

for i in range(2):

print(f"特征值 {i+1}: {eig\_pairs[i][0]}")

print(f"特征向量 {i+1}: {eig\_pairs[i][1]}")

# 计算 Wilks' Lambda 统计量

wilks\_lambda = np.prod(1 / (1 + eig\_vals))

print(f"Wilks' Lambda: {wilks\_lambda}")

前两个特征值及其对应的特征向量：

特征值 1: 0.0793873447540309

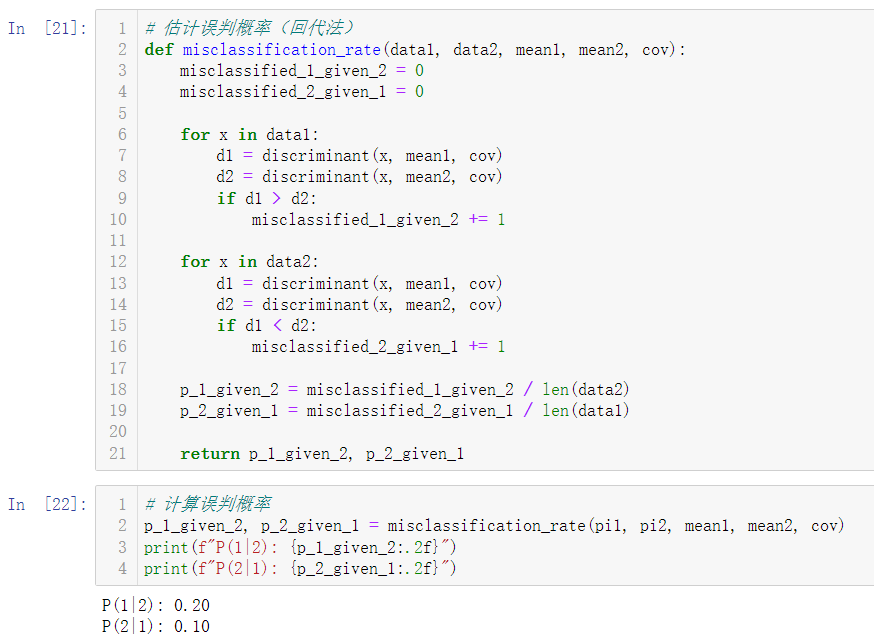
特征向量 1: [ 0.705717 -0.70849384]

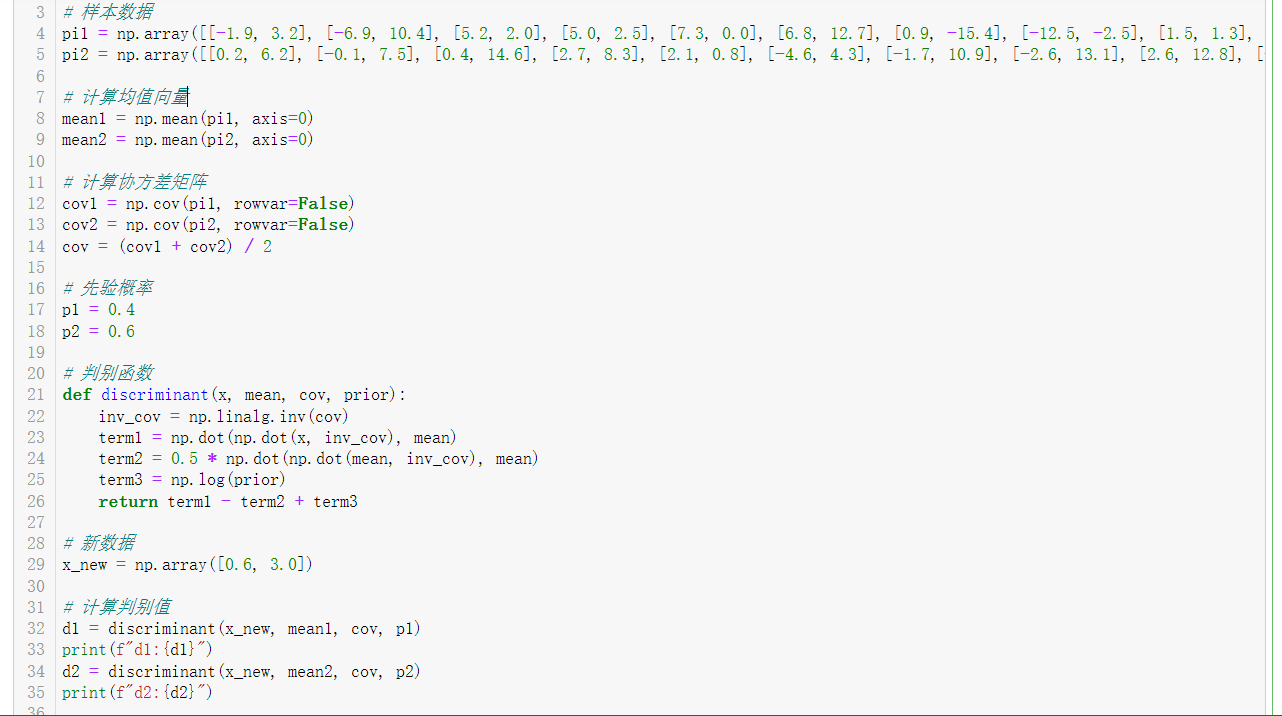
特征值 2: 0.0011498225485081712

特征向量 2: [-0.89053745 -0.45490993]

Wilks' Lambda: 0.9253874458316688

# 5.5

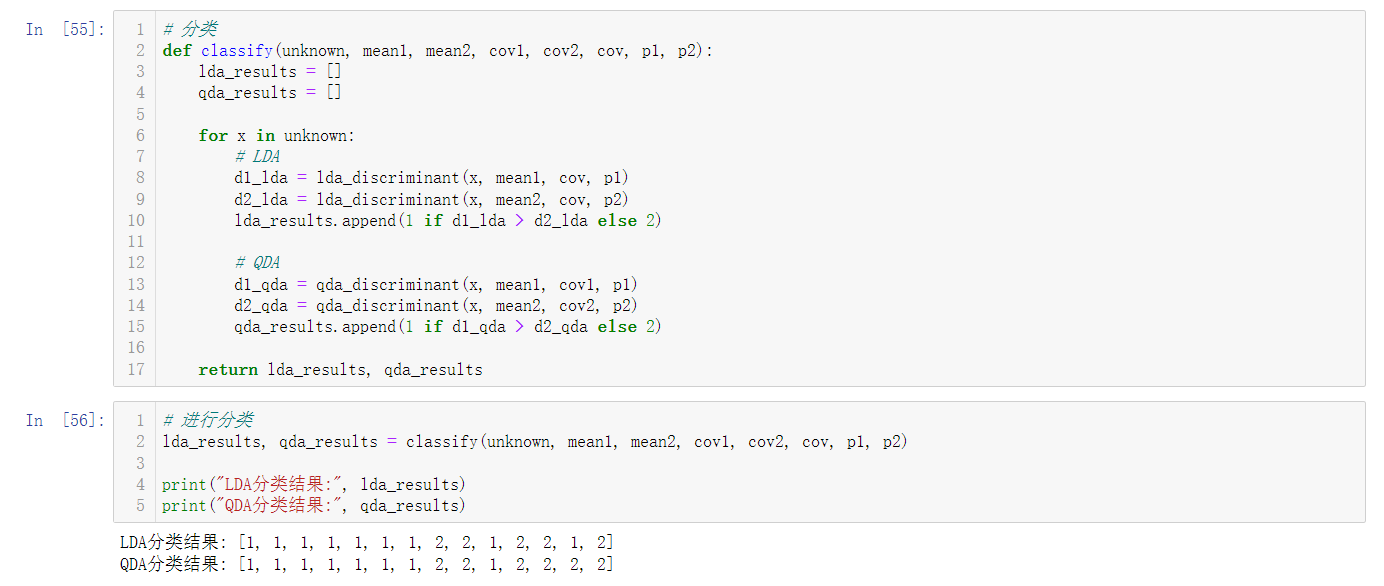


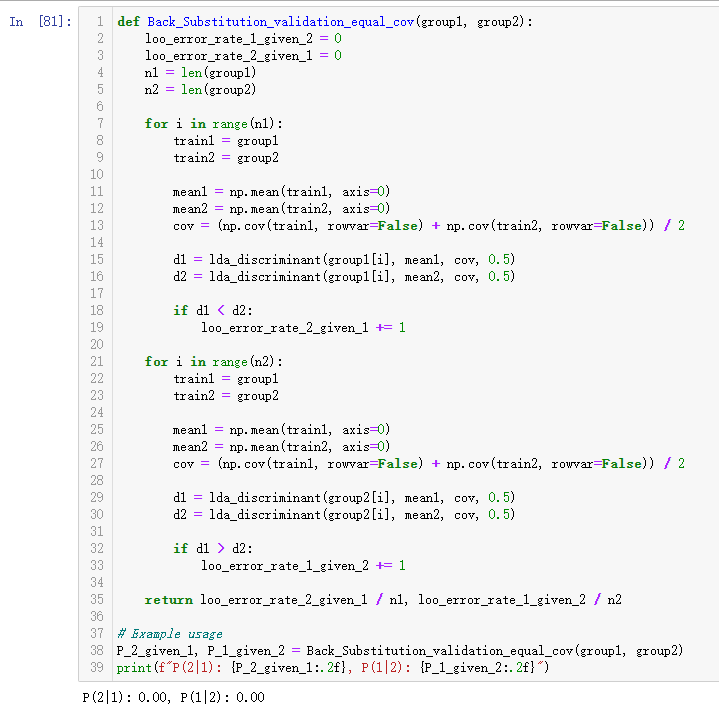




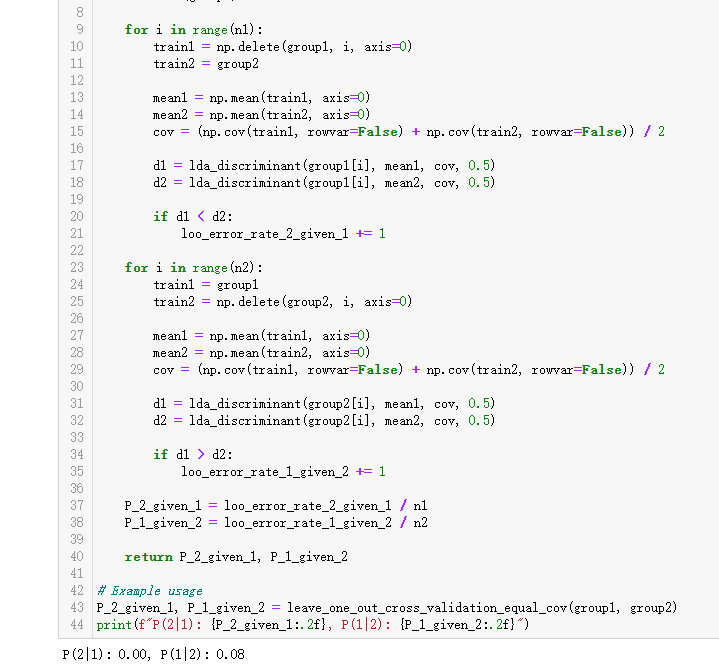
# 5.6

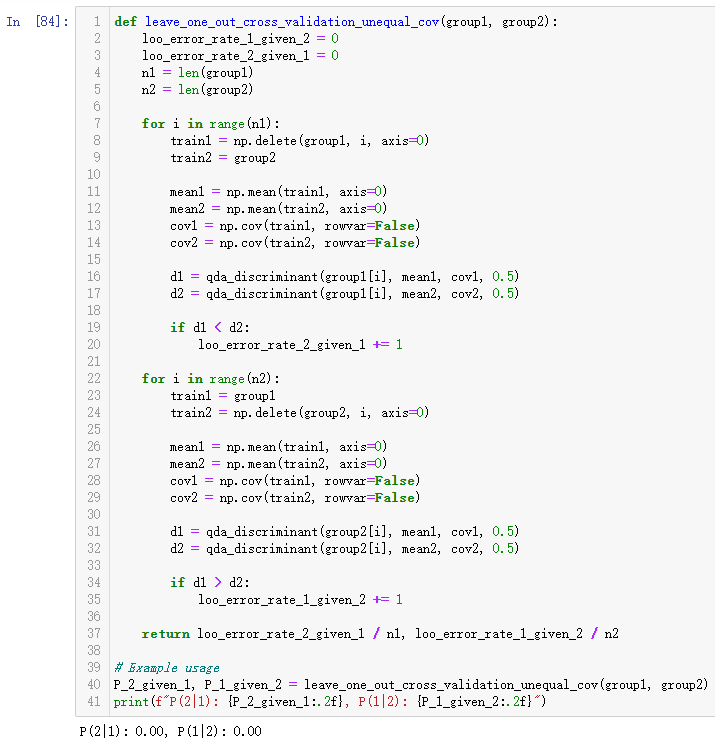


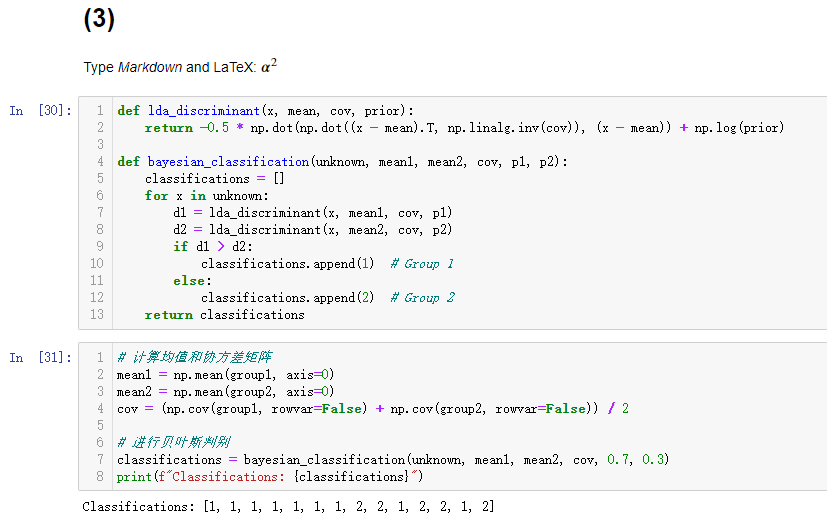












# 5.8

## MATLAB求解：

% 创建数据矩阵

data = [

110, 2, 2, 180, 1.5, 10.5, 10, 70, 1;

110, 6, 2, 290, 2, 17, 1, 105, 1;

110, 1, 1, 180, 0, 12, 13, 55, 1;

110, 1, 1, 180, 0, 12, 13, 65, 1;

110, 1, 1, 280, 0, 15, 9, 45, 1;

110, 3, 1, 250, 1.5, 11.5, 10, 90, 1;

110, 2, 1, 260, 0, 21, 3, 40, 1;

110, 2, 1, 180, 0, 12, 12, 55, 1;

100, 2, 1, 220, 2, 15, 6, 90, 1;

130, 3, 2, 170, 1.5, 13.5, 10, 120, 1;

100, 3, 2, 140, 2.5, 10.5, 8, 140, 1;

110, 2, 1, 200, 0, 21, 3, 35, 1;

140, 3, 1, 190, 4, 15, 14, 230, 1;

100, 3, 1, 200, 3, 16, 3, 110, 1;

110, 1, 1, 140, 0, 13, 12, 25, 1;

100, 3, 1, 200, 3, 17, 3, 110, 1;

110, 2, 1, 200, 1, 16, 8, 60, 1;

70, 4, 1, 260, 9, 7, 5, 320, 2;

110, 2, 0, 125, 1, 11, 14, 30, 2;

100, 2, 0, 290, 1, 21, 2, 35, 2;

110, 1, 0, 90, 1, 13, 12, 20, 2;

110, 3, 3, 140, 4, 10, 7, 160, 2;

110, 2, 0, 220, 1, 21, 3, 30, 2;

110, 2, 1, 125, 1, 11, 13, 30, 2;

110, 1, 0, 200, 1, 14, 11, 25, 2;

100, 3, 0, 0, 3, 14, 7, 100, 2;

120, 3, 0, 240, 5, 14, 12, 190, 2;

110, 2, 1, 170, 1, 17, 6, 60, 2;

160, 3, 2, 150, 3, 17, 13, 160, 2;

120, 2, 1, 190, 0, 15, 9, 40, 2;

140, 3, 2, 220, 3, 21, 7, 130, 2;

90, 3, 0, 170, 3, 18, 2, 90, 2;

100, 3, 0, 320, 1, 20, 3, 45, 2;

120, 3, 1, 210, 5, 14, 12, 240, 2;

110, 2, 0, 290, 0, 22, 3, 35, 2;

110, 2, 1, 70, 1, 9, 15, 40, 2;

110, 6, 0, 230, 1, 16, 3, 55, 2;

120, 1, 2, 220, 0, 12, 12, 35, 3;

120, 1, 2, 220, 1, 12, 11, 45, 3;

100, 4, 2, 150, 2, 12, 6, 95, 3;

50, 1, 0, 0, 0, 13, 0, 15, 3;

50, 2, 0, 0, 1, 10, 0, 50, 3;

100, 5, 2, 0, 2.7, 1, 1, 110, 3

];

% 分离特征和标签

X = data(:, 1:8);

y = data(:, 9);

% 计算组内均值

mu1 = mean(X(y == 1, :));

mu2 = mean(X(y == 2, :));

mu3 = mean(X(y == 3, :));

% 计算总体均值

mu = mean(X);

% 计算组内散布矩阵

S1 = cov(X(y == 1, :));

S2 = cov(X(y == 2, :));

S3 = cov(X(y == 3, :));

Sw = S1 + S2 + S3;

% 计算组间散布矩阵

Sb = (mu1 - mu)' \* (mu1 - mu) + (mu2 - mu)' \* (mu2 - mu) + (mu3 - mu)' \* (mu3 - mu);

% 计算费希尔判别函数

[V, D] = eig(Sb, Sw);

[~, idx] = sort(diag(D), 'descend');

W = V(:, idx(1:2)); % 选择前两个特征向量

% 归一化特征向量使其满足 t'Sw^(-1)t = 1

for i = 1:size(W, 2)

W(:, i) = W(:, i) / sqrt(W(:, i)' \* inv(Sw) \* W(:, i));

end

% 投影数据到新空间

X\_lda = X \* W;

% 显示结果

disp('归一化后的费希尔判别函数特征向量:');

disp(W);

## 结果：

归一化后的费希尔判别函数特征向量:

-0.0163 -0.0084

-0.1623 0.0555

0.6108 0.0247

0.0006 0.0028

-0.4779 -0.7732

-0.0746 0.0148

-0.0624 0.0148

0.0086 0.0205

## Python：

