Regresión logística

Código

Regresión logística

```
import matplotlib.pyplot as plt
import numpy as np
from pandas.io.parsers import read_csv
import scipy.optimize as opt

data = read_csv('p2/ex2data1.csv', header=None).values.astype(float)
x = data[:,:2]
y = data[:,2]
size = len(y)
positive = plt.scatter(x[np.where(y == 1), 0], x[np.where(y == 1), 1], marker='+',
c='k')
negative = plt.scatter(x[np.where(y == 0), 0], x[np.where(y == 0), 1], marker='o',
c='b')
plt.xlabel("Exam 1 score")
plt.ylabel("Exam 2 score")
plt.legend((positive, negative),('Admitted', 'Not admitted'))
plt.show()

X = np.ones((size,3))
X[:,1:] = x

theta = np.zeros((3, 1))

def sigmoid(x):
    return 1.0 / (1.0 + np.exp(-x))

def cost_function(theta, x, y, m):
    J = (-np.log(sigmoid(x.dot(theta)).T).dot(y) - np.log(1 -
sigmoid(x.dot(theta)).T).dot(1 - y))/m
    return J

def gradient_function(theta, x, y, m):
    h = sigmoid(x.dot(theta)).reshape(-1, 1)
    y = y.reshape(m, 1)
    gradient = x.T.dot(h - y)/m
    return gradient

print("Initial cost = " + str(cost_function(theta, X, y, size)))
print("Initial gradient = " + str(gradient_function(theta, X, y, size)))

result = opt.fmin_tnc(func=cost_function , x0=theta , fprime=gradient_function ,
args=(X, y, size))
theta_opt = result[0]
print("Optimal cost = " + str(cost_function(theta_opt, X, y, size)))
theta_opt = theta_opt.reshape((3,1))

linspace = np.linspace(30, 100, 1000)
boundary = -(theta_opt[0] + theta_opt[1]*linspace)/theta_opt[2]
positive = plt.scatter(x[np.where(y == 1), 0], x[np.where(y == 1), 1], marker='+',
c='k')
```

```
negative = plt.scatter(x[np.where(y == 0), 0], x[np.where(y == 0), 1], marker='o', c='b')
plt.plot(linspace, boundary)
plt.xlabel("Exam 1 score")
plt.ylabel("Exam 2 score")
plt.legend((positive, negative),('Admitted', 'Not admitted'))
plt.show()

def accuracy(theta, x, y, m):
    predictions = sigmoid(x.dot(theta))
    predictions_corrected = [1 if pred >= 0.5 else 0 for pred in predictions]
    number = np.sum(predictions_corrected == y)
    return (float(number)/m)*100

print("Accuracy = " + str(accuracy(theta_opt, X, y, size)) +"%")
```

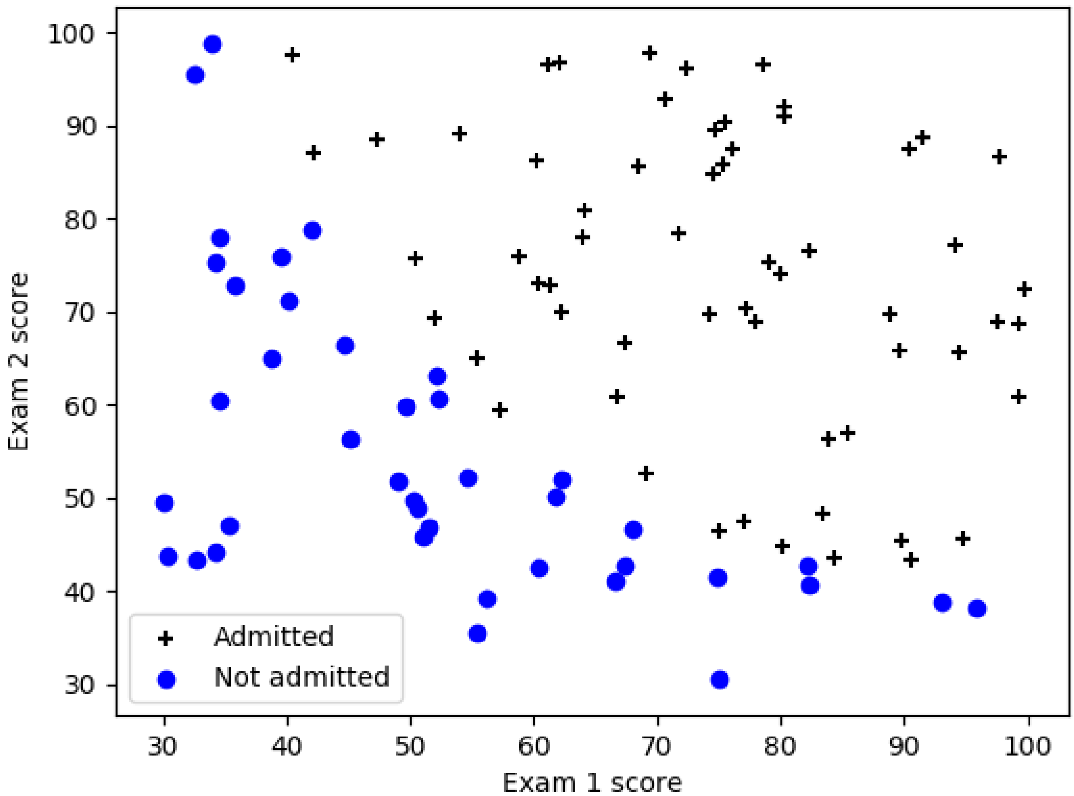
Regresión logística regularizada

```
import matplotlib.pyplot as plt
import numpy as np
from pandas.io.parsers import read_csv
import scipy.optimize as opt
from sklearn.preprocessing import PolynomialFeatures

data = read_csv('p2/ex2data2.csv', header=None).values.astype(float)
x = data[:,:2]
y = data[:,2]
size = len(y)
positive = plt.scatter(x[np.where(y == 1), 0], x[np.where(y == 1), 1], marker='+', c='k')
negative = plt.scatter(x[np.where(y == 0), 0], x[np.where(y == 0), 1], marker='o', c='b')
plt.xlabel("Microchip test 1")
plt.ylabel("Microchip test 2")
plt.legend((positive, negative),('Passed', 'Not passed'))
plt.show()

polynomial = PolynomialFeatures(6)
x_poly = polynomial.fit_transform(x)

theta = np.zeros((28, 1))
lambda_reg = 1

def sigmoid(x):
    return 1.0 / (1.0 + np.exp(-x))

def cost_function(theta, x, y, m, lambda_reg):
    J = (-np.log(sigmoid(x.dot(theta)).T).dot(y) - np.log(1 - sigmoid(x.dot(theta)).T).dot(1 - y))/m + lambda_reg*np.sum(np.square(theta))/(2*m)
    return J

def gradient_function(theta, x, y, m, lambda_reg):
    h = sigmoid(x.dot(theta)).reshape(-1, 1)
    y = y.reshape(m, 1)
    gradient = np.zeros((theta.shape[0], 1))
    gradient = x.T.dot(h - y)/m
    theta = theta.reshape((theta.shape[0], 1))
    gradient[1:] = gradient[1:] + (lambda_reg/m)*theta[1:]
    return gradient
```

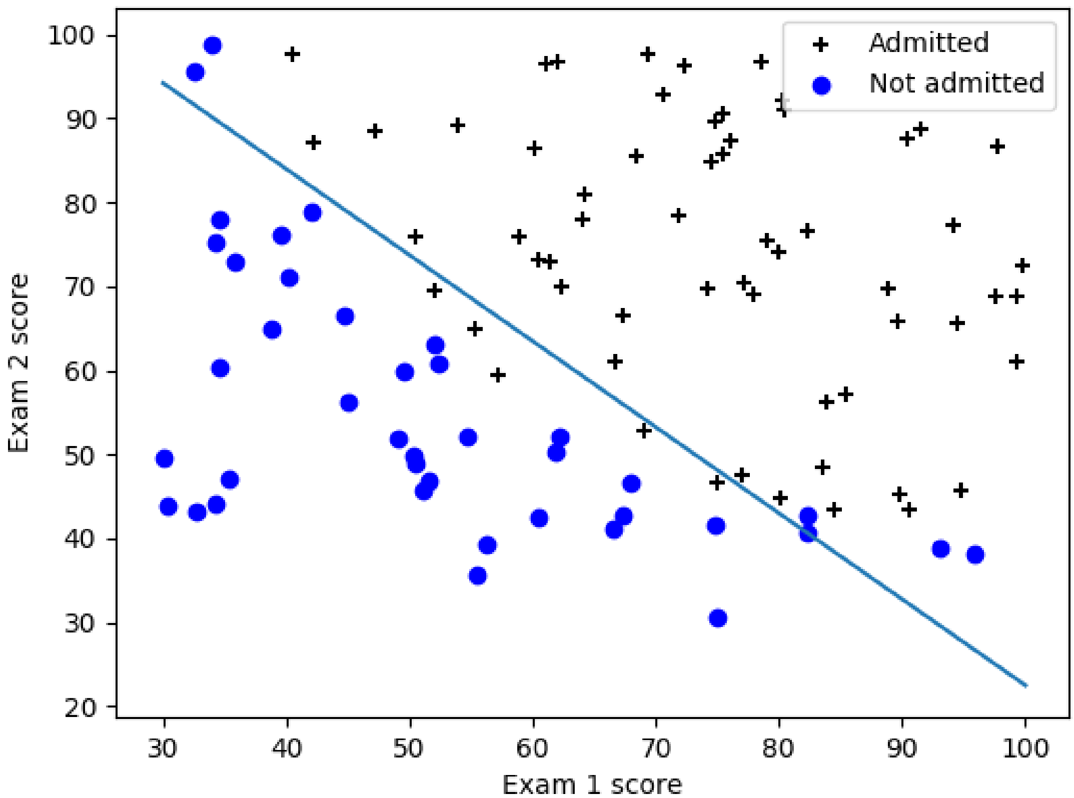
```
print("Initial cost = " + str(cost_function(theta, x_poly, y, size, lambda_reg)))

result = opt.fmin_tnc(func=cost_function , x0=theta , fprime=gradient_function , 
args=(x_poly, y, size, lambda_reg))
theta_opt = result[0]

lin1 = np.linspace(-0.75, 1.00, 50)
lin2 = np.linspace(-0.75, 1.00, 50)
z = np.zeros((len(lin1), len(lin2)))

def plotting_preprocessing(lin1, lin2, theta_opt):
    for i in range(len(lin1)):
        for j in range(len(lin2)):
            z[i,j] = np.dot(polynomial.fit_transform(np.column_stack((lin1[i], 
lin2[j]))), theta_opt)
    return z

def accuracy(theta, x, y, m):
    predictions = sigmoid(x.dot(theta))
    predictions_corrected = [1 if pred >= 0.5 else 0 for pred in predictions]
    number = np.sum(predictions_corrected == y)
    return (float(number)/m)*100

positive = plt.scatter(x[np.where(y == 1), 0], x[np.where(y == 1), 1], marker='+', 
c='k')
negative = plt.scatter(x[np.where(y == 0), 0], x[np.where(y == 0), 1], marker='o', 
c='b')
plt.contour(lin1,lin2,plotting_preprocessing(lin1, lin2, theta_opt).T,0)
plt.xlabel("Microchip test 1")
plt.ylabel("Microchip test 2")
plt.legend((positive, negative),('Passed', 'Not passed'))
plt.title("Lambda = 1")
plt.show()

for lambda_reg in np.arange(0, 10, 0.5):
    theta = np.zeros((28, 1))
    result = opt.fmin_tnc(func=cost_function , x0=theta , 
fprime=gradient_function , args=(x_poly, y, size, lambda_reg))
    theta_opt = result[0]

    lin1 = np.linspace(-0.75, 1.00, 50)
    lin2 = np.linspace(-0.75, 1.00, 50)
    z = np.zeros((len(lin1), len(lin2)))

    positive = plt.scatter(x[np.where(y == 1), 0], x[np.where(y == 1), 1], 
marker='+', c='k')
    negative = plt.scatter(x[np.where(y == 0), 0], x[np.where(y == 0), 1], 
marker='o', c='b')
    plt.contour(lin1,lin2,plotting_preprocessing(lin1, lin2, theta_opt).T,0)
    plt.xlabel("Microchip test 1")
    plt.ylabel("Microchip test 2")
    plt.legend((positive, negative),('Passed', 'Not passed'))
    plt.title("Lambda = " + str(lambda_reg))
    plt.show()

print("Accuracy = " + str(accuracy(theta_opt, x_poly, y, size)) + "%")
```

Resultados

Regresión logística

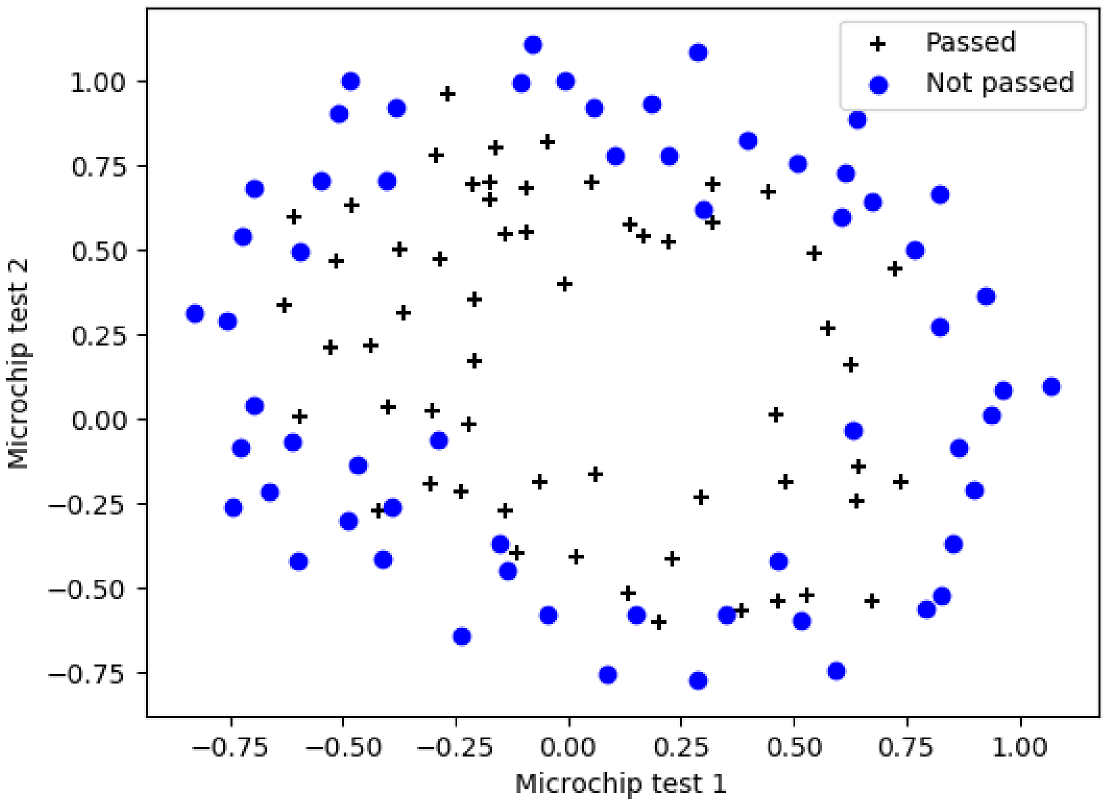



```
Initial cost = [0.69314718]
Initial gradient = [[-0.1]
 [-12.00921659]
 [-11.26284221]]
Optimal cost = 0.20349770158947517
Accuracy = 89.0%
```

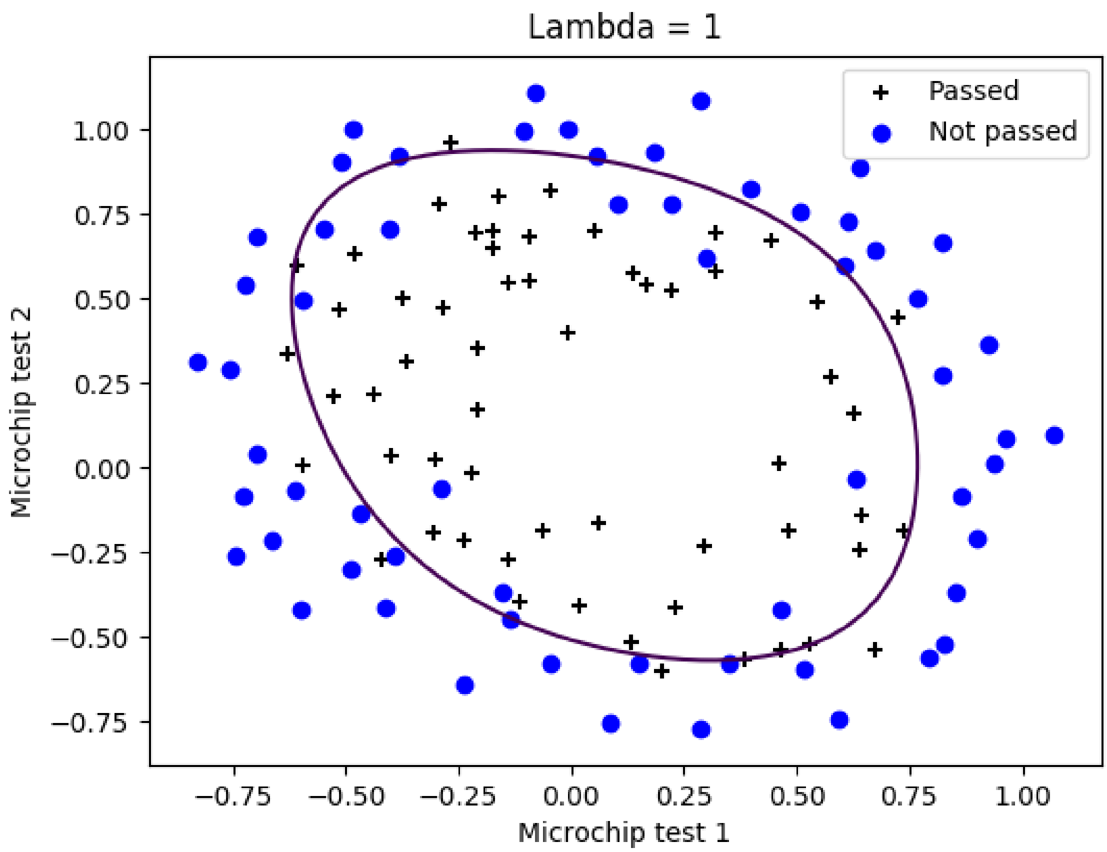


Admitted
Not admitted
Exam 2 score
Exam 1 score
100
90
80
70
60
50
40
30
20
30
40
50
60
70
80
90
100

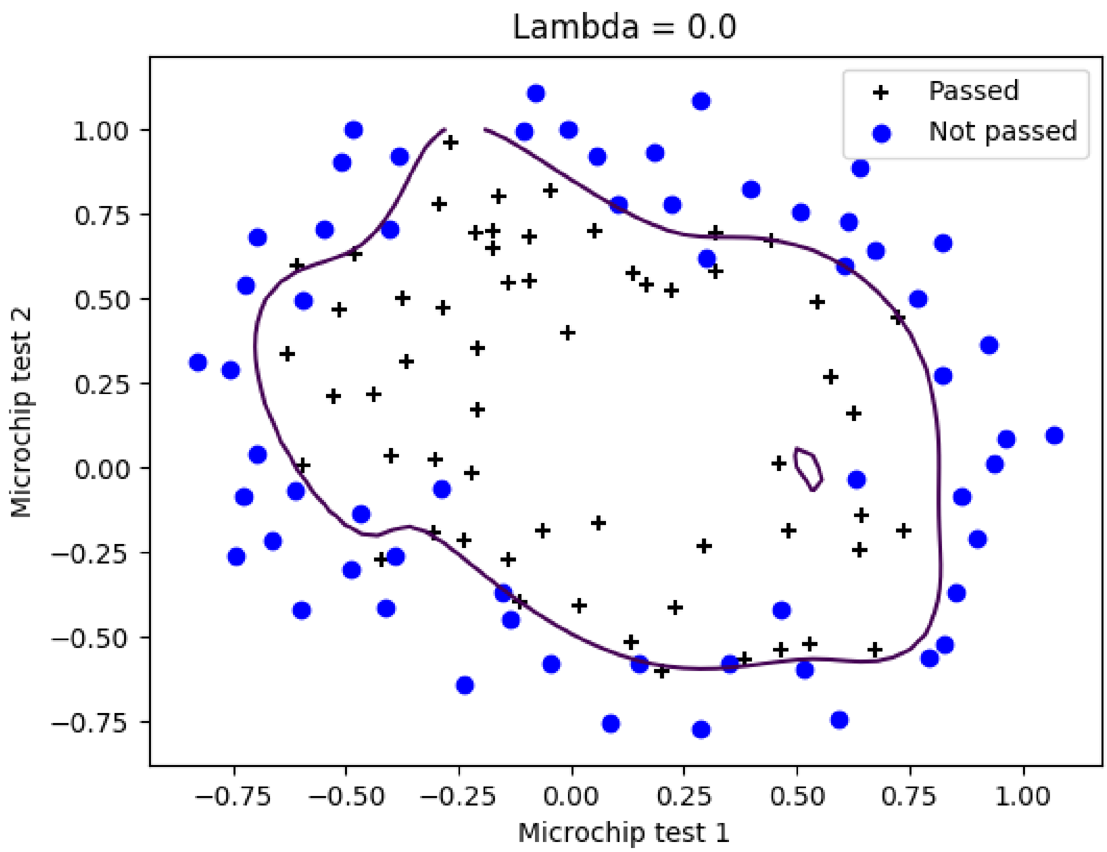
Regresión logística regularizada



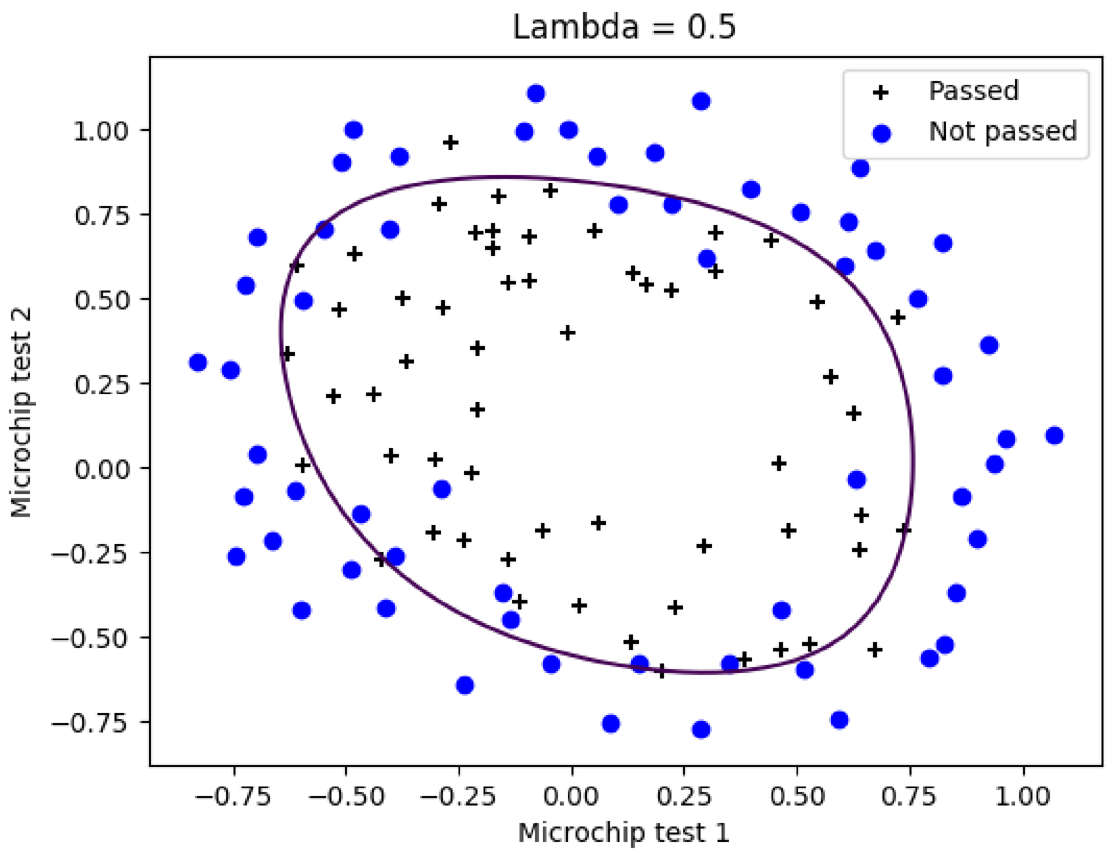

```
Initial cost = [0.69314718]
```



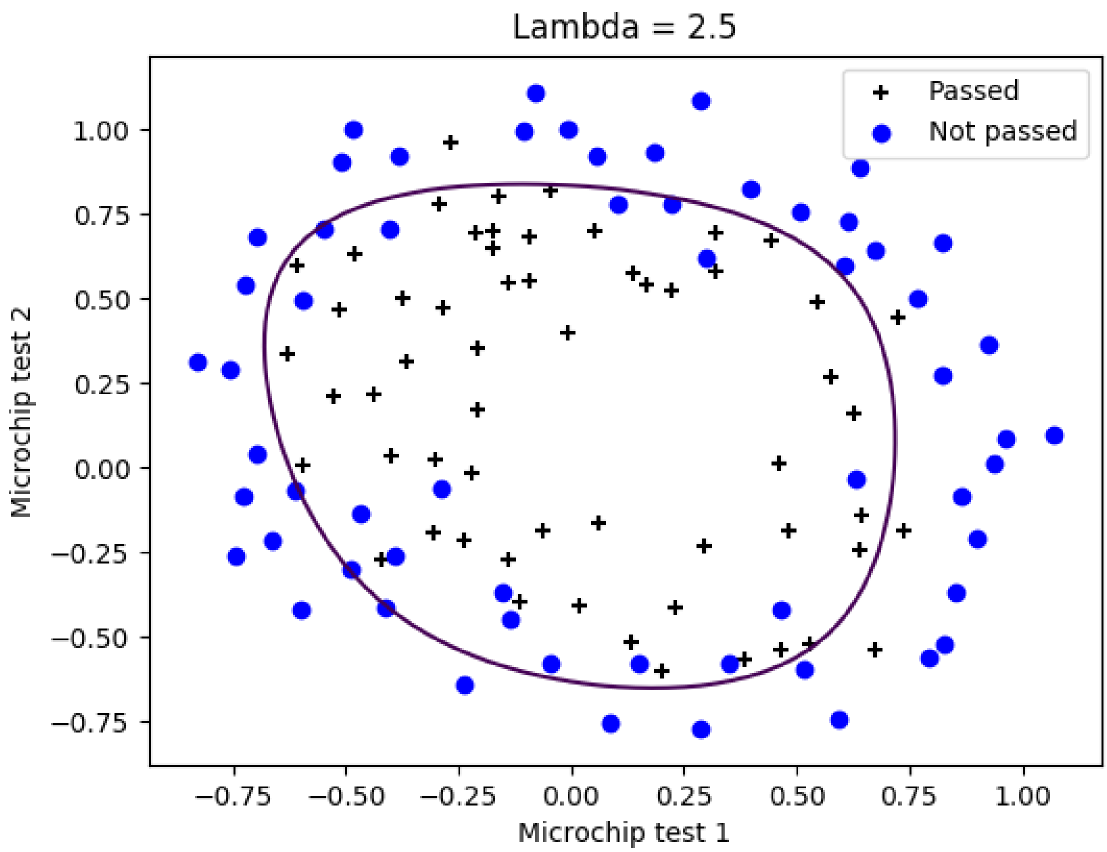

```
Accuracy = 83.0508474576%
```



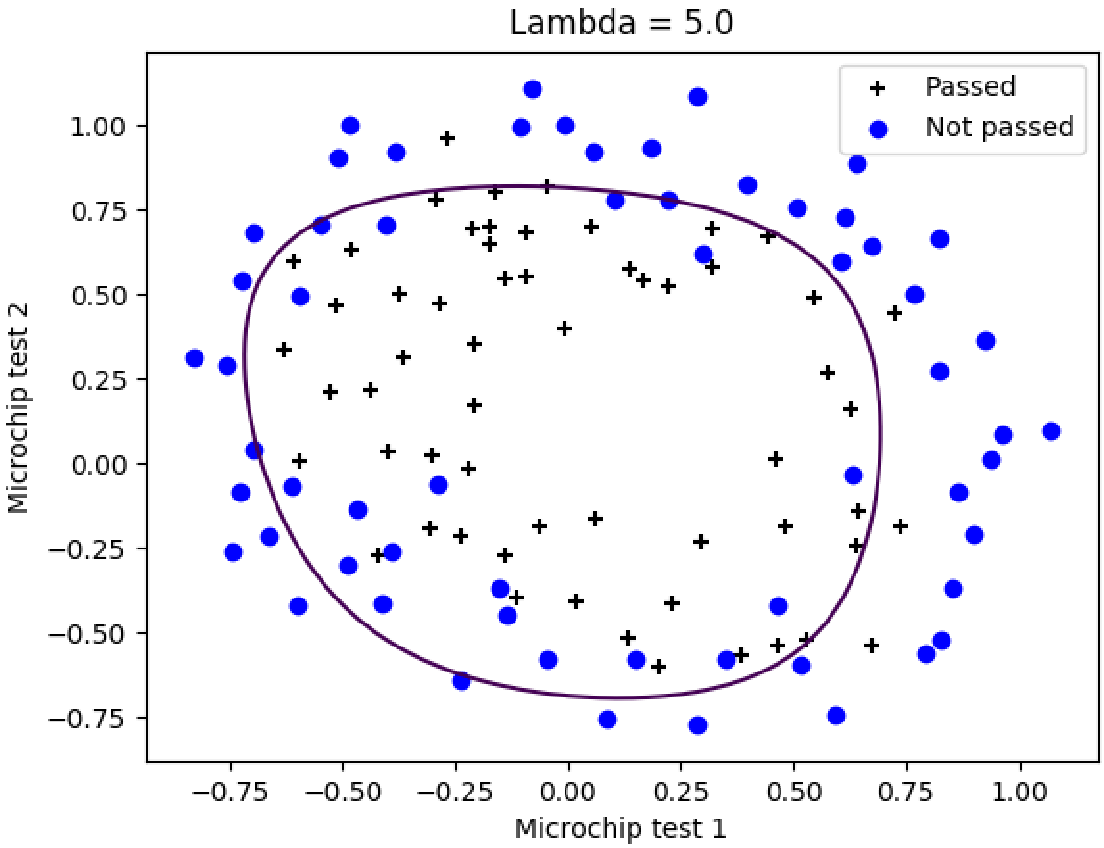

```
Accuracy = 88.1355932203%
```




```
Accuracy = 82.2033898305%
```



Accuracy = 82.2033898305%




```
Accuracy = 81.3559322034%
```

Comentarios

Cuando lambda se aumenta, para pequeños valores de lambda el valor de accuracy es mejor, pero para los valores de lambda más grandes el valor de accuracy es peor. Para este conjunto de datos el valor optimal de lambda es entre lambda = 1 y lambda = 2. Para lambda = 0 el valor de accuracy es mejor debido a overfitting.