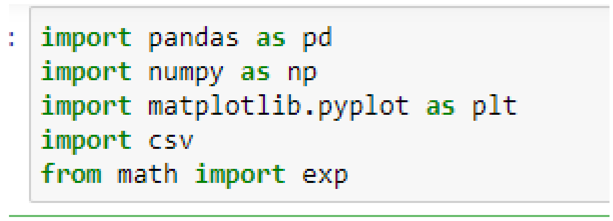
Applied Machine Learning Assignment 3

Name: Aneeq Asghar
Roll number: 17L-4525
Section: EE-7A

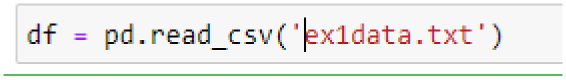
Exercise 1:

Imports



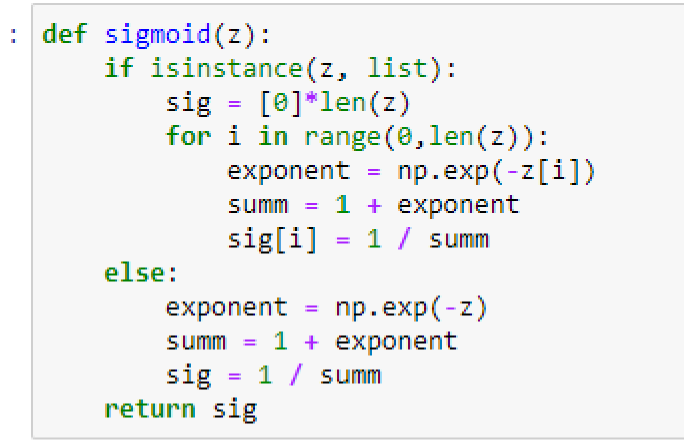
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import csv
from math import exp
```

Reading text file



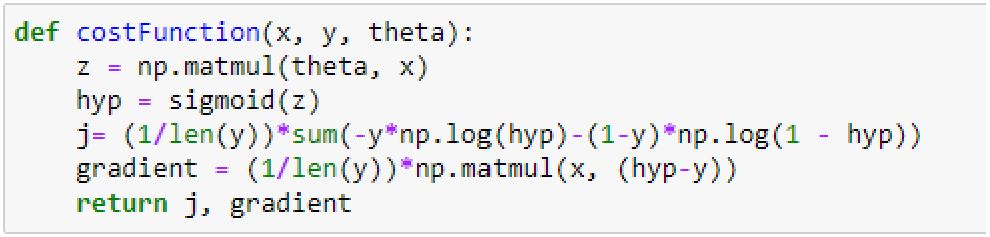
```python
df = pd.read_csv('ex1data.txt')
```

Sigmoid Function:



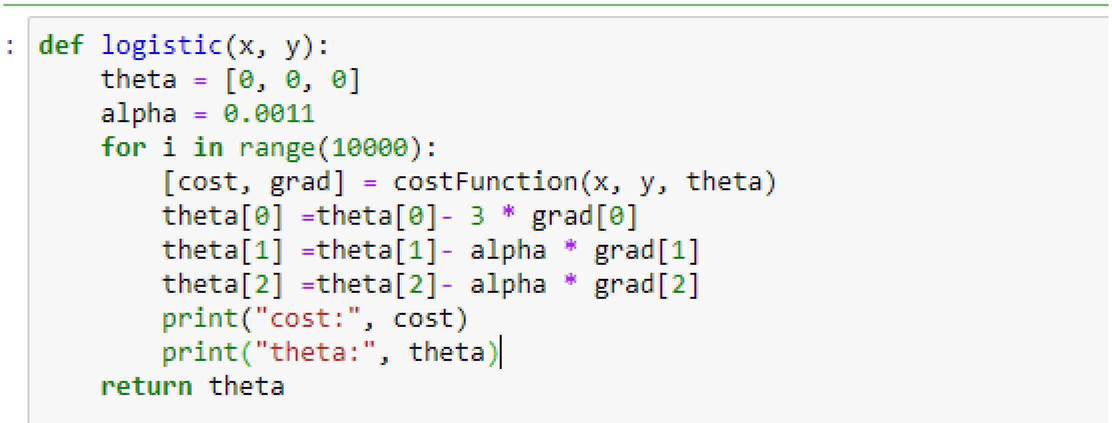
```python
def sigmoid(z):
    if isinstance(z, list):
        sig = [0]*len(z)
        for i in range(0,len(z)):
            exponent = np.exp(-z[i])
            summ = 1 + exponent
            sig[i] = 1 / summ
    else:
        exponent = np.exp(-z)
        summ = 1 + exponent
        sig = 1 / summ
    return sig
```

Cost function:



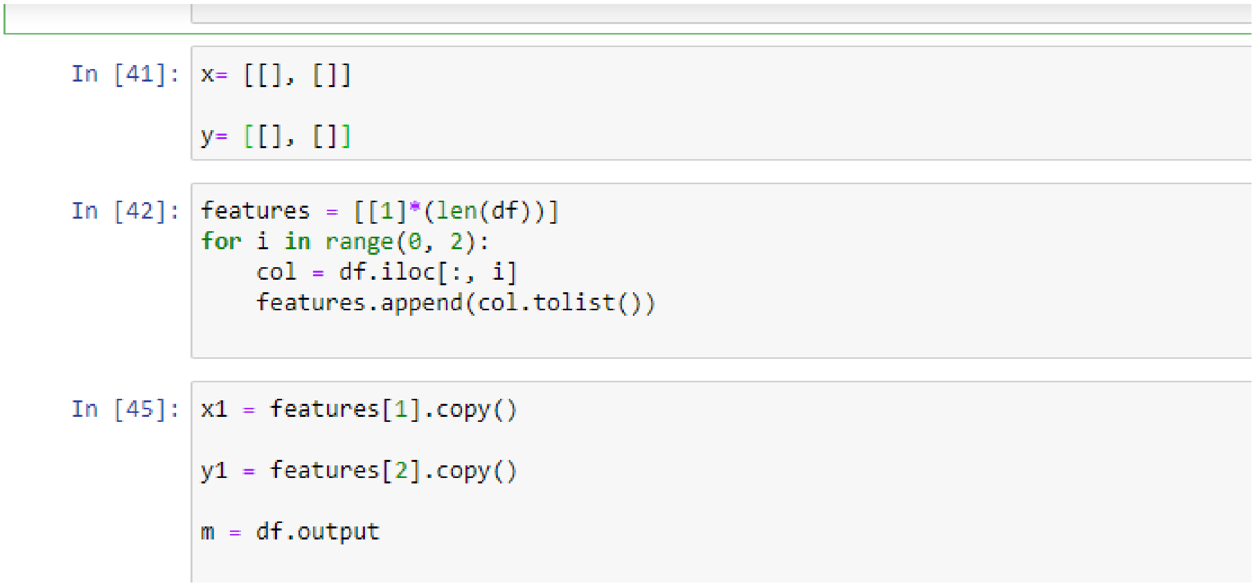
```python
def costFunction(x, y, theta):
    z = np.matmul(theta, x)
    hyp = sigmoid(z)
    j= (1/len(y))*sum(-y*np.log(hyp)-(1-y)*np.log(1 - hyp))
    gradient = (1/len(y))*np.matmul(x, (hyp-y))
    return j, gradient
```

Logistic function:

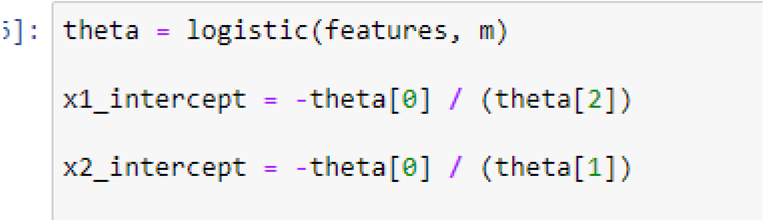


```
: def logistic(x, y):
      theta = [0, 0, 0]
      alpha = 0.0011
      for i in range(10000):
          [cost, grad] = costFunction(x, y, theta)
          theta[0] =theta[0]- 3 * grad[0]
          theta[1] =theta[1]- alpha * grad[1]
          theta[2] =theta[2]- alpha * grad[2]
          print("cost:", cost)
          print("theta:", theta)
      return theta
```

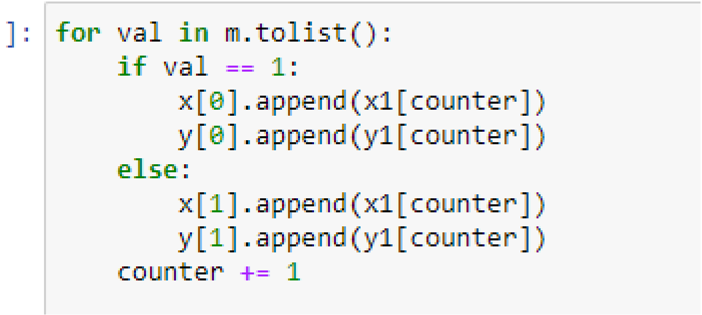
Main:



```
In [41]: x= [[], []]

         y= [[], []]

In [42]: features = [[1]*(len(df))]
         for i in range(0, 2):
             col = df.iloc[:, i]
             features.append(col.tolist())

In [45]: x1 = features[1].copy()

         y1 = features[2].copy()

         m = df.output
```

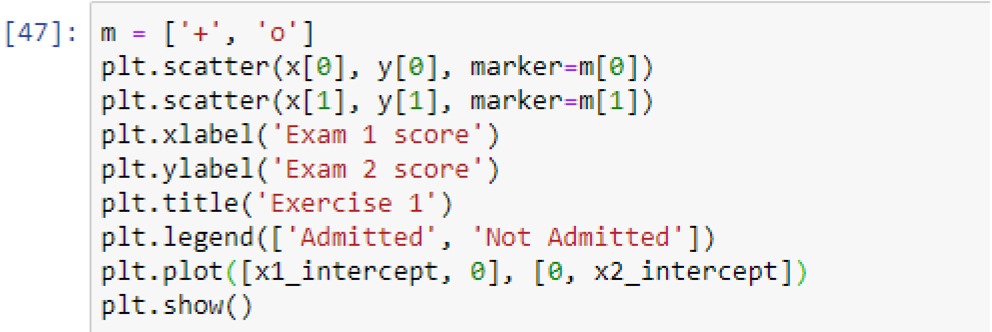


```
5]: theta = logistic(features, m)

    x1_intercept = -theta[0] / (theta[2])

    x2_intercept = -theta[0] / (theta[1])
```



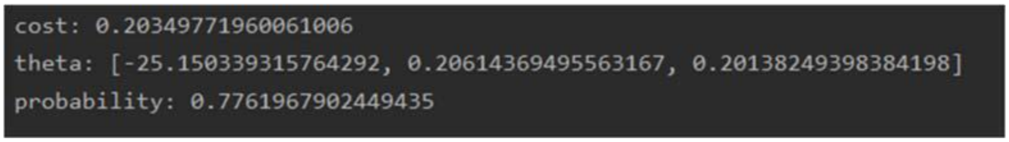
```
]: for val in m.tolist():
       if val == 1:
           x[0].append(x1[counter])
           y[0].append(y1[counter])
       else:
           x[1].append(x1[counter])
           y[1].append(y1[counter])
       counter += 1
```

Plotting:

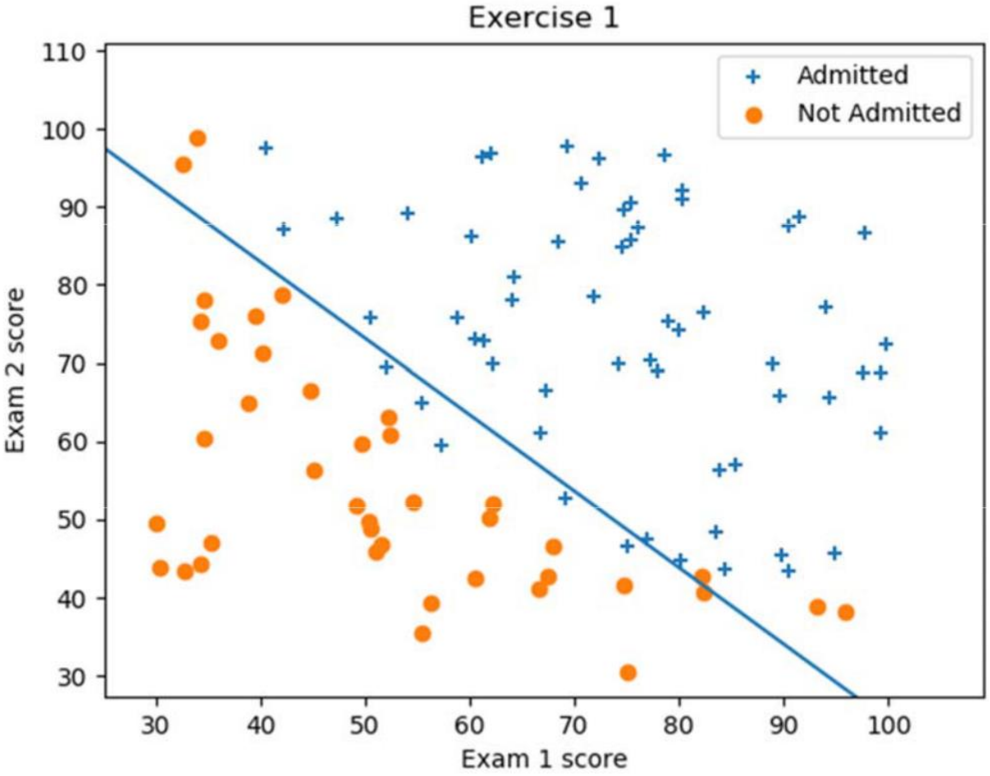


```
[47]: m = ['+', 'o']
      plt.scatter(x[0], y[0], marker=m[0])
      plt.scatter(x[1], y[1], marker=m[1])
      plt.xlabel('Exam 1 score')
      plt.ylabel('Exam 2 score')
      plt.title('Exercise 1')
      plt.legend(['Admitted', 'Not Admitted'])
      plt.plot([x1_intercept, 0], [0, x2_intercept])
      plt.show()
```

Output:

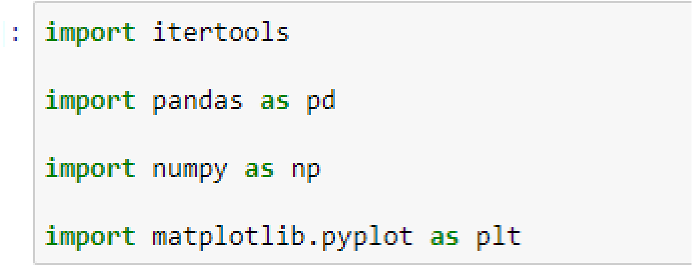


```
cost: 0.20349771960061006
theta: [-25.150339315764292, 0.20614369495563167, 0.20138249398384198]
probability: 0.7761967902449435
```

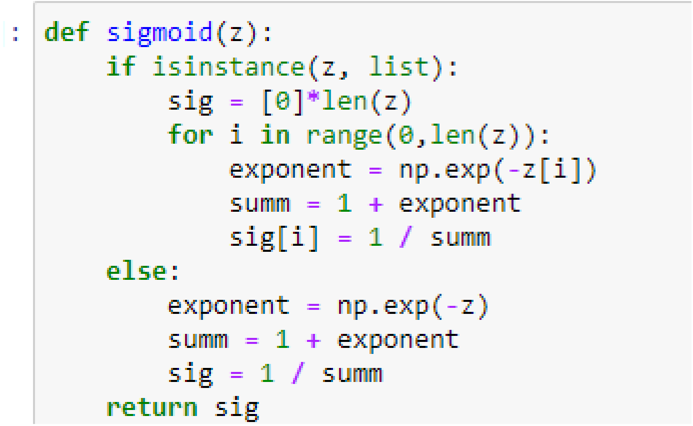

Exercise 1
Admitted
Not Admitted
Exam 2 score
110
100
90
80
70
60
50
40
30
Exam 1 score
30
40
50
60
70
80
90
100

EXERCISE 2:

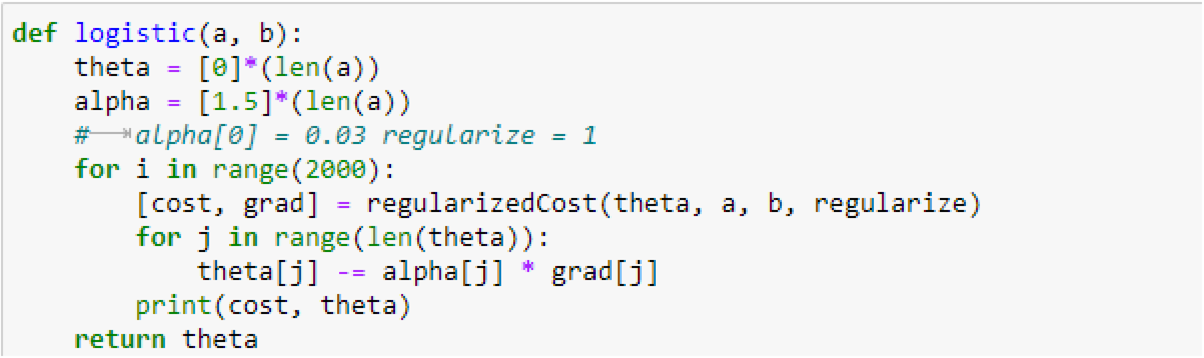
IMPORTS



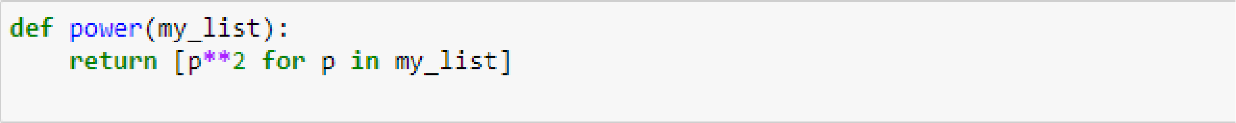
```python
import itertools

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt
```



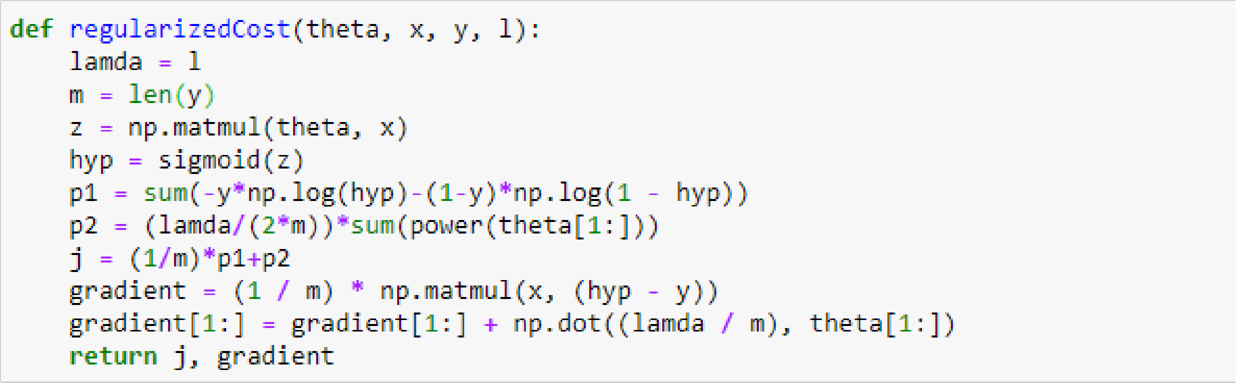
```python
def sigmoid(z):
    if isinstance(z, list):
        sig = [0]*len(z)
        for i in range(0,len(z)):
            exponent = np.exp(-z[i])
            summ = 1 + exponent
            sig[i] = 1 / summ
    else:
        exponent = np.exp(-z)
        summ = 1 + exponent
        sig = 1 / summ
    return sig
```



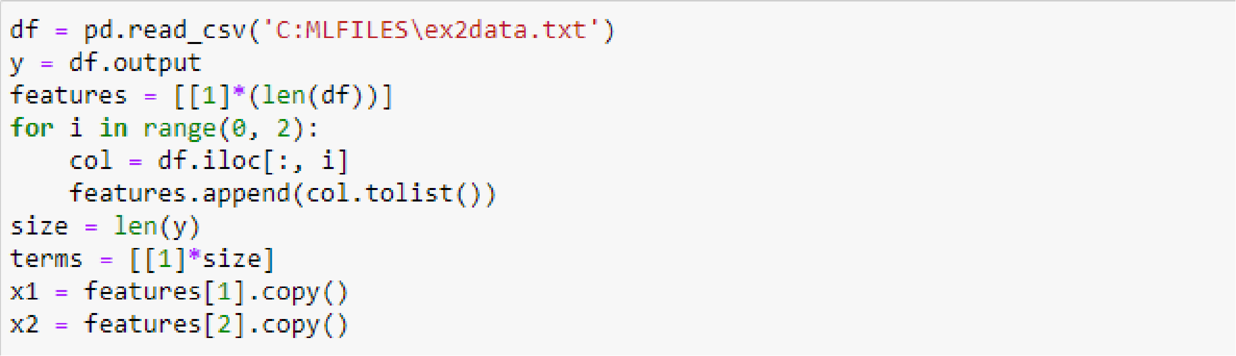
```python
def logistic(a, b):
    theta = [0]*(len(a))
    alpha = [1.5]*(len(a))
    #→alpha[0] = 0.03 regularize = 1
    for i in range(2000):
        [cost, grad] = regularizedCost(theta, a, b, regularize)
        for j in range(len(theta)):
            theta[j] -= alpha[j] * grad[j]
        print(cost, theta)
    return theta
```



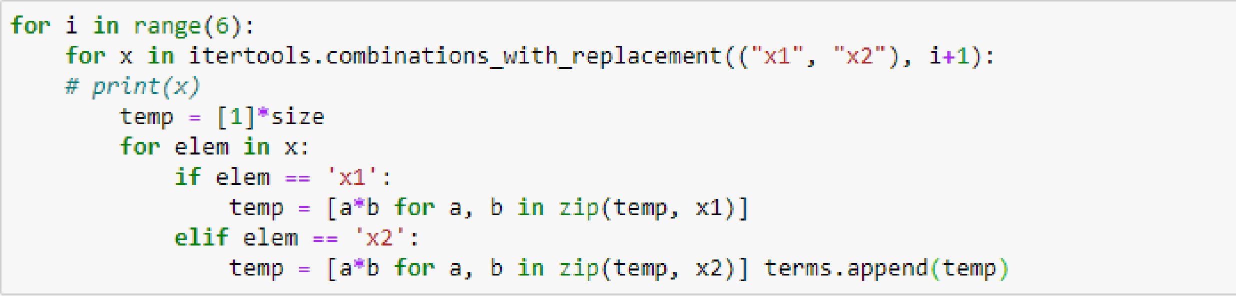
```
def power(my_list):
    return [p**2 for p in my_list]
```



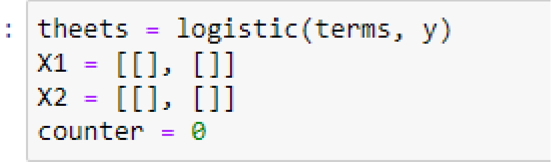
```
def regularizedCost(theta, x, y, l):
    lamda = l
    m = len(y)
    z = np.matmul(theta, x)
    hyp = sigmoid(z)
    p1 = sum(-y*np.log(hyp)-(1-y)*np.log(1 - hyp))
    p2 = (lamda/(2*m))*sum(power(theta[1:]))
    j = (1/m)*p1+p2
    gradient = (1 / m) * np.matmul(x, (hyp - y))
    gradient[1:] = gradient[1:] + np.dot((lamda / m), theta[1:])
    return j, gradient
```



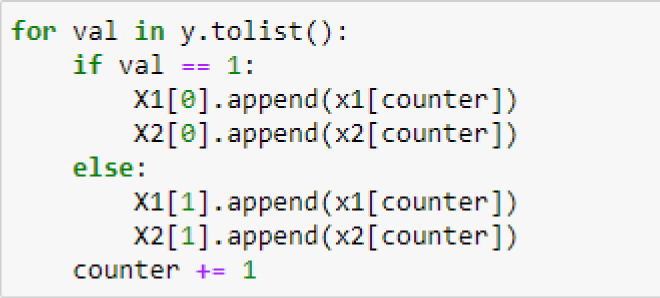
```
df = pd.read_csv('C:MLFILES\ex2data.txt')
y = df.output
features = [[1]*(len(df))]
for i in range(0, 2):
    col = df.iloc[:, i]
    features.append(col.tolist())
size = len(y)
terms = [[1]*size]
x1 = features[1].copy()
x2 = features[2].copy()
```



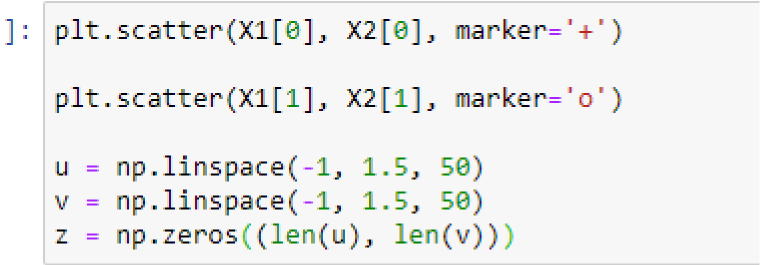
```
for i in range(6):
    for x in itertools.combinations_with_replacement(("x1", "x2"), i+1):
    # print(x)
        temp = [1]*size
        for elem in x:
            if elem == 'x1':
                temp = [a*b for a, b in zip(temp, x1)]
            elif elem == 'x2':
                temp = [a*b for a, b in zip(temp, x2)] terms.append(temp)
```



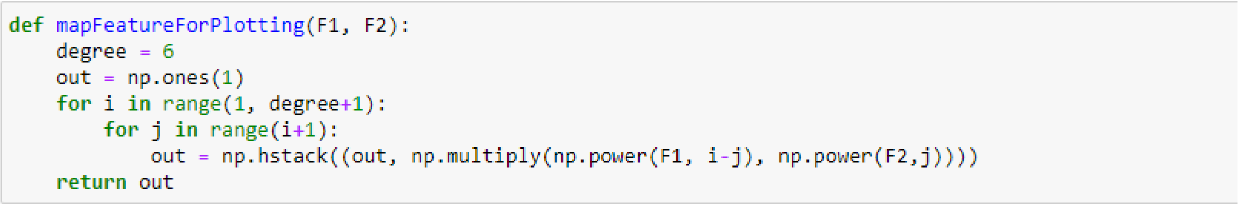
```
: theets = logistic(terms, y)
  X1 = [[], []]
  X2 = [[], []]
  counter = 0
```



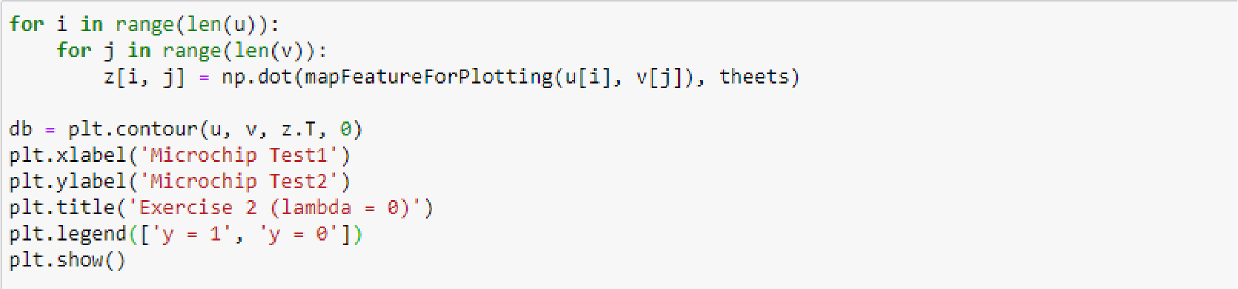
```
for val in y.tolist():
    if val == 1:
        X1[0].append(x1[counter])
        X2[0].append(x2[counter])
    else:
        X1[1].append(x1[counter])
        X2[1].append(x2[counter])
    counter += 1
```



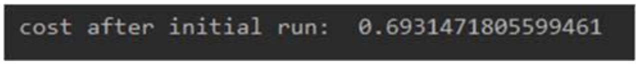
```
plt.scatter(X1[0], X2[0], marker='+')

plt.scatter(X1[1], X2[1], marker='o')

u = np.linspace(-1, 1.5, 50)
v = np.linspace(-1, 1.5, 50)
z = np.zeros((len(u), len(v)))
```



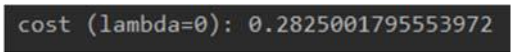
```
def mapFeatureForPlotting(F1, F2):
    degree = 6
    out = np.ones(1)
    for i in range(1, degree+1):
        for j in range(i+1):
            out = np.hstack((out, np.multiply(np.power(F1, i-j), np.power(F2,j))))
    return out
```



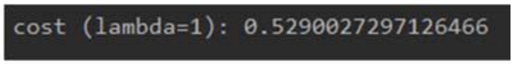
```
for i in range(len(u)):
    for j in range(len(v)):
        z[i, j] = np.dot(mapFeatureForPlotting(u[i], v[j]), theets)

db = plt.contour(u, v, z.T, 0)
plt.xlabel('Microchip Test1')
plt.ylabel('Microchip Test2')
plt.title('Exercise 2 (lambda = 0)')
plt.legend(['y = 1', 'y = 0'])
plt.show()
```



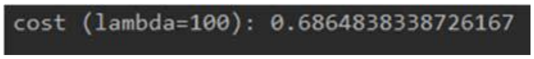
```
cost after initial run:  0.6931471805599461
```



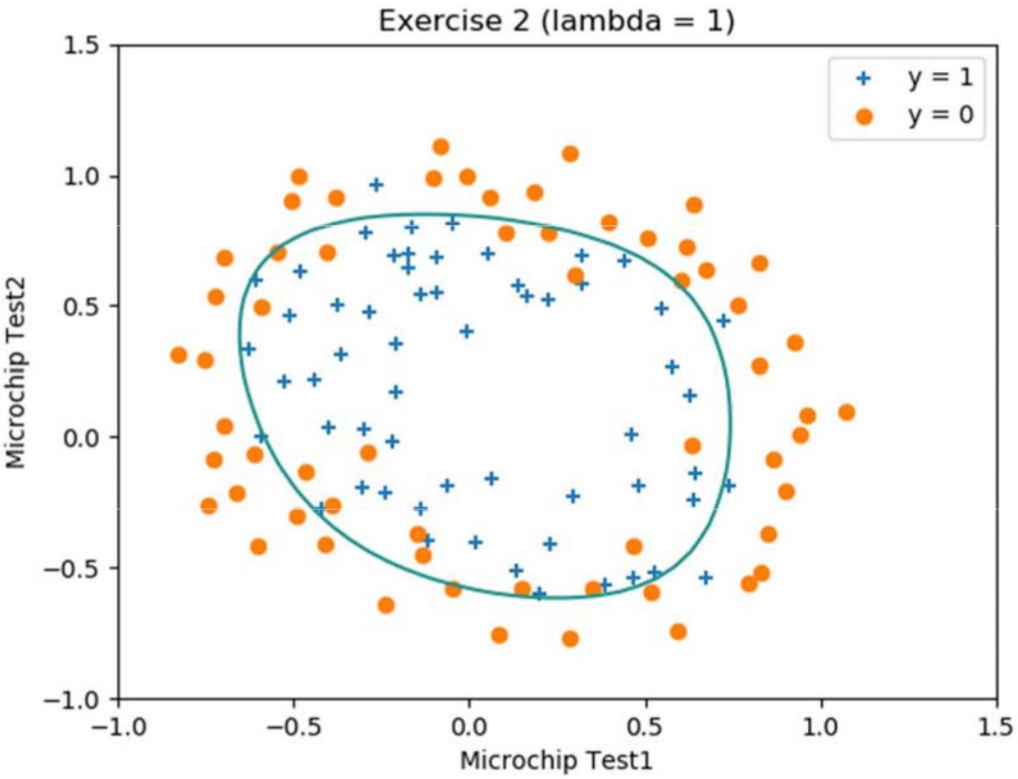
```
cost (lambda=0): 0.2825001795553972
```

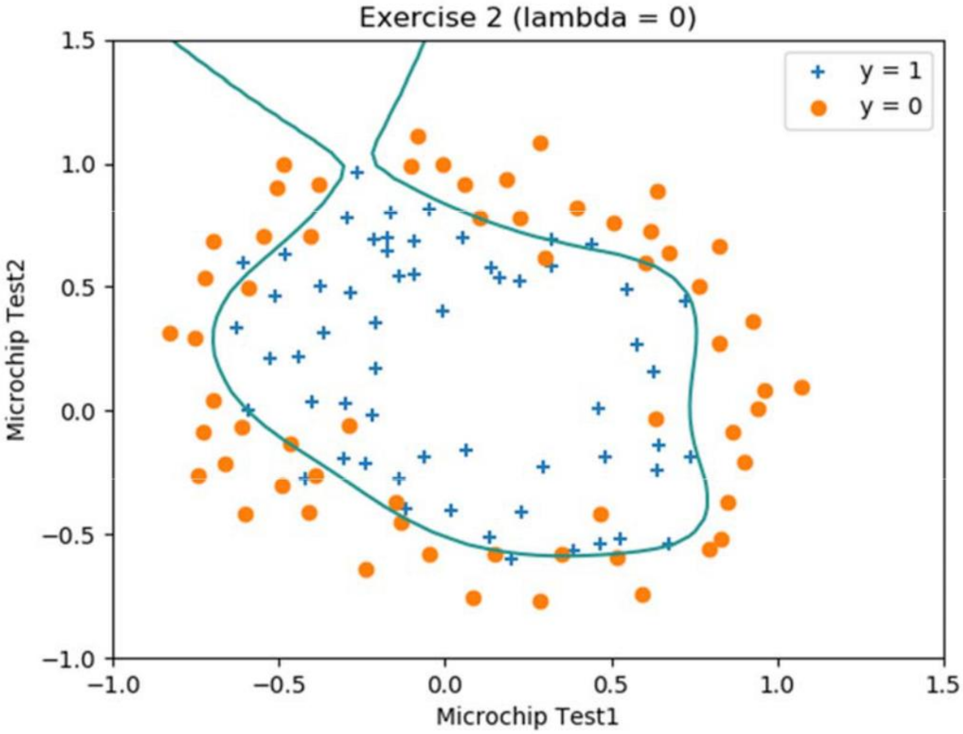


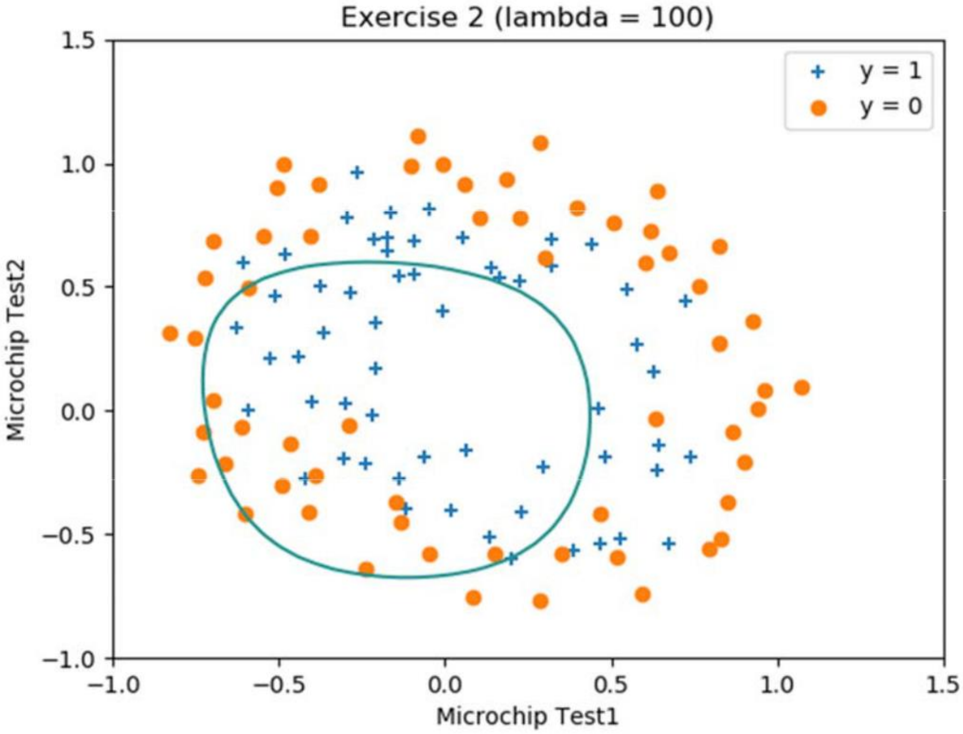
```
cost (lambda=1): 0.5290027297126466
```



```
cost (lambda=100): 0.6864838338726167
```




Exercise 2 (lambda = 0)
y = 1
y = 0
Microchip Test2
Microchip Test1
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0
1.5


Exercise 2 (lambda = 100)
y = 1
y = 0
Microchip Test2
Microchip Test1
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0
1.5