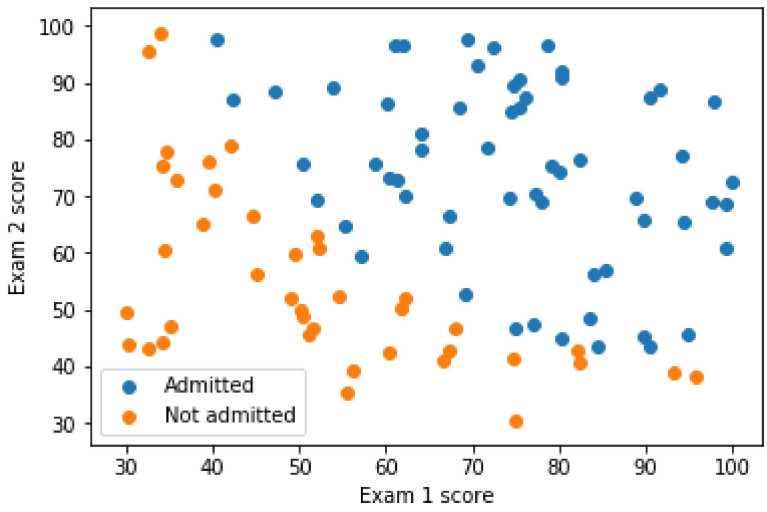
In [16]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import scipy.optimize as opt
```

In [17]:
```
data = pd.read_csv('examscore_train.txt', header = None)
X = data.iloc[:,:-1]
y = data.iloc[:,2]
data.head(10)
```

Out[17]:

	0	1	2
0	34.623660	78.024693	0
1	30.286711	43.894998	0
2	35.847409	72.902198	0
3	60.182599	86.308552	1
4	79.032736	75.344376	1
5	45.083277	56.316372	0
6	61.106665	96.511426	1
7	75.024746	46.554014	1
8	76.098787	87.420570	1
9	84.432820	43.533393	1

In [18]:
```
mask = y == 1
adm = plt.scatter(X[mask][0].values, X[mask][1].values)
not_adm = plt.scatter(X[~mask][0].values, X[~mask][1].values)
plt.xlabel('Exam 1 score')
plt.ylabel('Exam 2 score')
plt.legend((adm, not_adm), ('Admitted', 'Not admitted'))
plt.show()
```



In [19]:
```
def sigmoid(x):
    return 1/(1+np.exp(-x))
```

In [20]:
```
def costFunction(theta, X, y):
    J = (-1/m) * np.sum(np.multiply(y, np.log(sigmoid(X @ theta)))+ np.multipl
y((1-y), np.log(1 - sigmoid(X @ theta))))
    return J
```

In [21]:
```
def gradient(theta, X, y):
    return ((1/m) * X.T@ (sigmoid(X @ theta) - y))
```

In [22]:
```
(m, n) = X.shape
X = np.hstack((np.ones((m,1)), X))
y = y[:, np.newaxis]
theta = np.zeros((n+1,1))
J = costFunction(theta, X, y)
print(J)
```

```
0.6931471805599453
```

In [23]:
```
temp=opt.fmin_tnc(func=costFunction,
                  x0=theta.flatten(),fprime=gradient,
                  args=(X,y.flatten()))
theta_optimized=temp[0]
print(theta_optimized)
```

```
[-25.16131862   0.20623159   0.20147149]
```

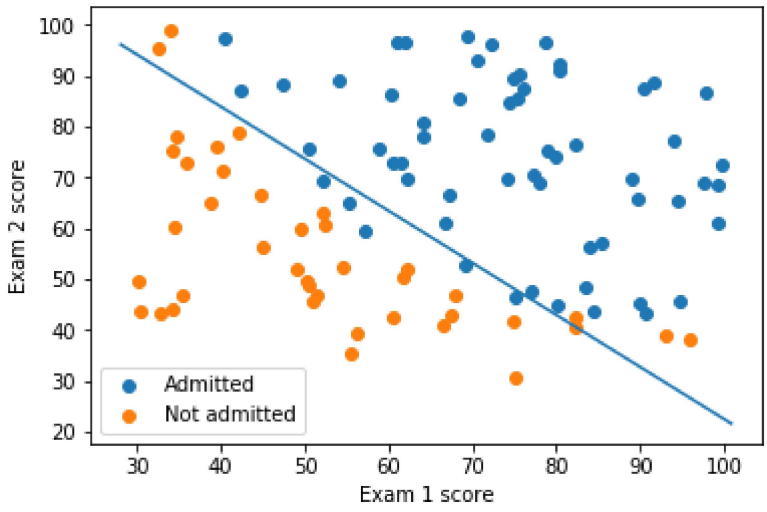
In [24]:
```
J=costFunction(theta_optimized[:,np.newaxis],X,y)
print(J)
```

```
0.20349770158947464
```

In [25]:
```
plot_x = [np.min(X[:,1]-2), np.max(X[:,2]+2)]
```

In [26]:
```
plot_y = -1/theta_optimized[2]*(theta_optimized[0]
          + np.dot(theta_optimized[1],plot_x))
```

In [27]:
```
mask = y.flatten() == 1
adm = plt.scatter(X[mask][:,1], X[mask][:,2])
not_adm = plt.scatter(X[~mask][:,1], X[~mask][:,2])
decision_boun = plt.plot(plot_x, plot_y)
plt.xlabel('Exam 1 score')
plt.ylabel('Exam 2 score')
plt.legend((adm, not_adm), ('Admitted', 'Not admitted'))
plt.show()
```



In [ ]: