Assignment 2 – CLL788

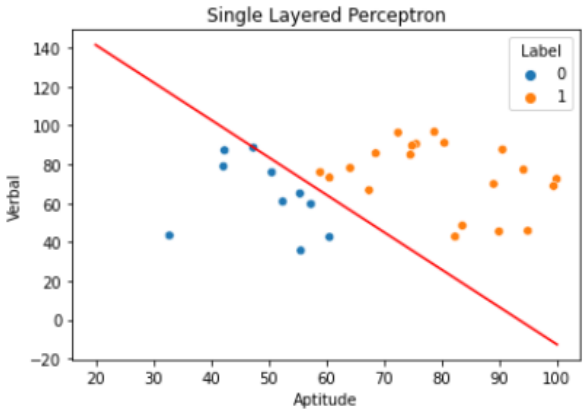
Ansh Lodhi, 2019CH70161

Filename: assignment2.ipynb

A)

```
def perceptron(X,Y,alpha):
    y = Y
    x = np.hstack((np.ones(Y.shape),X))
    w = np.ones((x.shape[1],1)).reshape(3,1)

    for i in range(50):
        for xx, yy in zip(x, y):
            if np.dot(xx, w) >0:
                pp = 1
            else:
                pp=-1
            if(pp==yy):
                continue
            w+= alpha*xx.reshape(3,1)*(yy)
    return w

def pred(X, w):
    x = np.hstack((np.ones((X.shape[0],1)),X))
    act_input = np.dot(x,w)
    prediction = np.where(act_input>t, 1,0)
    return prediction

weights = perceptron(x,y_,0.1)
```

```
------------------------------(A) Single Layer Perceptron----------------------------------
Weights Obtained:
[[-4.        ]
 [ 1.54170043]
 [ 0.79984075]]
learning rate:  0.1
-------------------------------------------------------------------------------------------
```



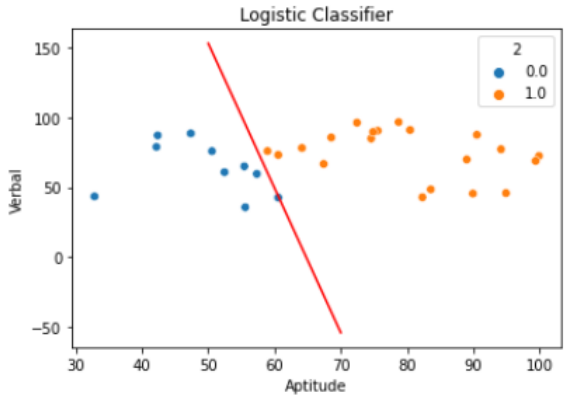
B) Tried Building the model, however I was unable to get desired result. Below is the work I did.

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

def sigmoid_grad(x):
    # val = np.zeros(x.shape)
    val = sigmoid(x)*(1-sigmoid(x))
    return val

def predict(w1, w2 ,X):
    # y = Y.reshape((X.shape[0],1))
    x = np.hstack((np.ones((X.shape[0],1)),X))
    hidden = x.dot(w1.T)
    activated1 =sigmoid(hidden)
    activated1 = np.concatenate([np.ones((activated1.shape[0], 1)), activated1], axis=1)
    final = activated1.dot(w2.T)
    activated2 = sigmoid(final)
    return x, activated1, activated2

def cost(W1, W2, X, Y):
    w1 = W1
    w2 = W2
    J = 0

    grad_w1 = np.zeros(w1.shape)
    grad_w2 = np.zeros(w2.shape)
    m = Y.size
    a1, a2, a3 = predict(w1, w2, X)
    J = (-1 / m) * np.sum((np.log(a3) * Y) + np.log(1 - a3) * (1 - Y))

    d3 = a3 - Y.reshape((70,1))
    d2 = d3.dot(w2)[:, 1:]*sigmoid_grad(a1.dot(w1.T))
    D1 = d2.T.dot(a1)
    D2 = d3.T.dot(a2)
    #print(sigmoid_grad(a1.dot(w1.T)))
    grad_w1 = (1/m)*D1
    grad_w2 = (1/m)*D2

    return J, grad_w1, grad_w2

w1 = np.random.random((3,3))
w2 = np.random.random((1,4))

for i in range(10000):
    k, wc1, wc2 = cost(w1,w2,x,y)
    w1 = w1 - 0.01*wc1
    w2 = w2 - 0.01*wc2
pred = np.where(predict(w1, w2, x_test)[2]>0.5,1,0)
```

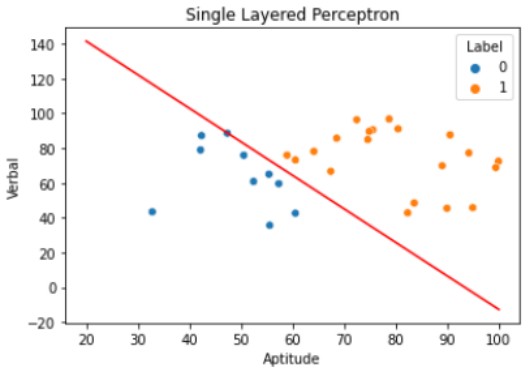
```
------------------------------(A) Multiple Layer Perceptron------------------
Weights Obtained:
w1: [[0.22127512 0.13082527 0.25982897]
 [0.24390041 0.22502522 0.55432843]
 [0.24002284 0.66896143 0.81222318]]
w2: [[-0.26859059 -0.51289511  0.42994086  0.40870324]]
-----------------------------------------------------------------------------
```

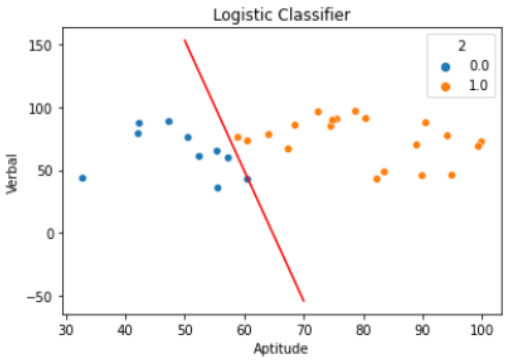
C)

```
------------------------------------------------------------
Logistic Regression
Number of Iterations:  189561
Time Taken:  24.834946632385254
The Values of theta for Logistic regression [[-4.71972859]
 [ 0.06900916]
 [ 0.00665459]]
The Final cost is:  0.480046705755547
------------------------------------------------------------
```



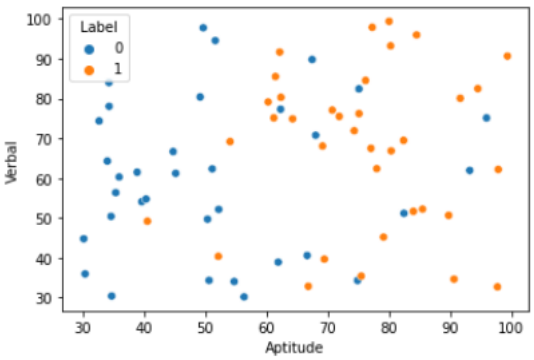
D)





Although the classification result was almost same in both single layer perceptron and logistic classifier, we can see that separation line in logistic classifier is much more steeper.

This would have led to difference in classification if data set was larger.



If we compare both the cases with the original given dataset, me see that logistic classifier would have performed better.