COL774-ML ASSIGNMENT-1
RISHI SHAH
2019CS10394

Q1)

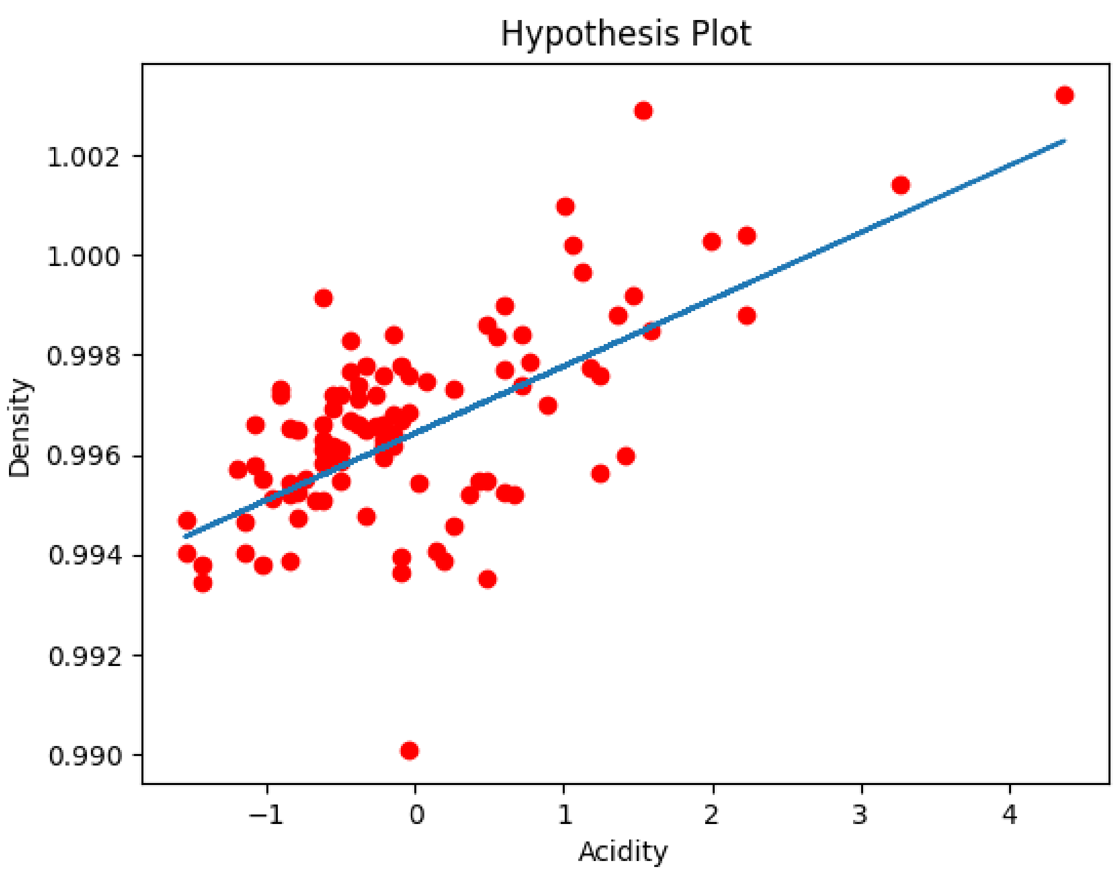
a) Learning Rate chosen is :- 0.025
   Reason for keeping this learning rate is that convergence was fast and it did not overshoot the minima. According to the hypothesis $y = \theta_0 + \theta_1 x$, then

   Final Parameters are: $\theta_0 = 0.996$ and $\theta_1 = 0.0013$.

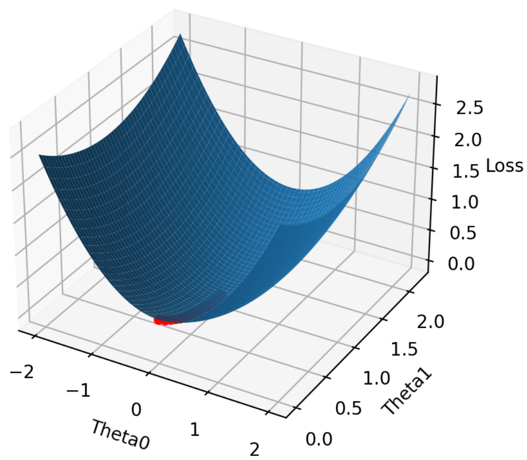
   Final Loss is : 1.21×10^{-6}

   Stopping Criteria :- previous_loss - current_loss $< 10^{-9}$

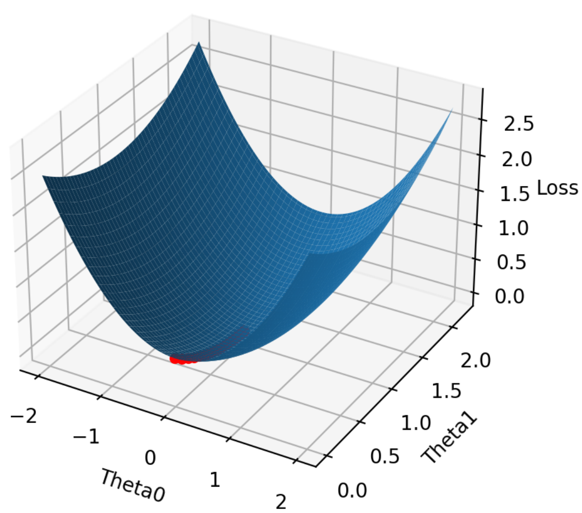
b) Plot of Data with the Hypothesis Function :-



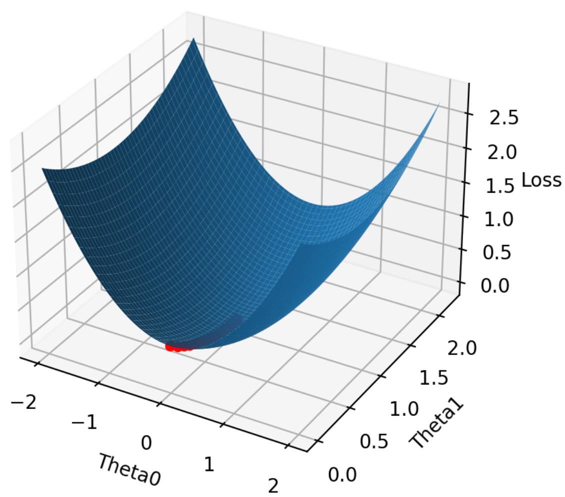
c) Mesh :-
   i) Learning Rate = 0.001



ii) Learning Rate = 0.025

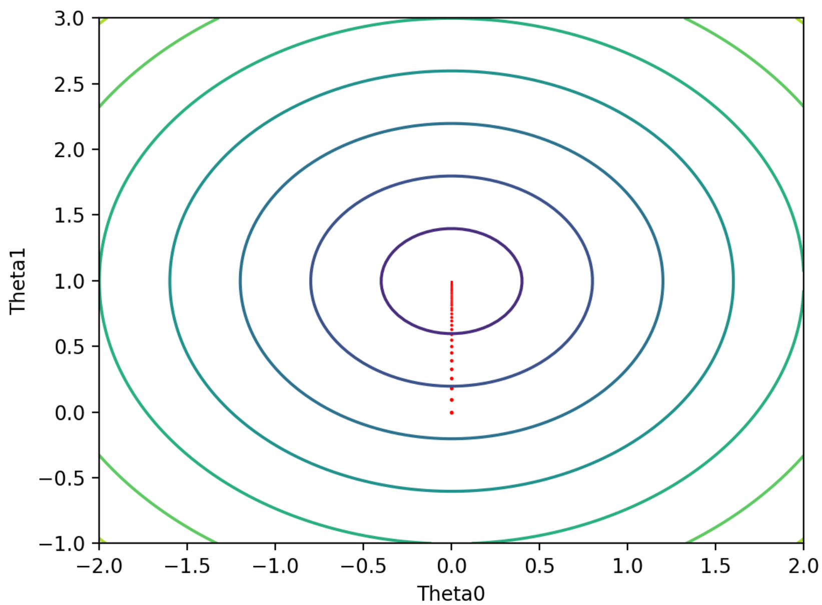


iii) Learning Rate = 0.1

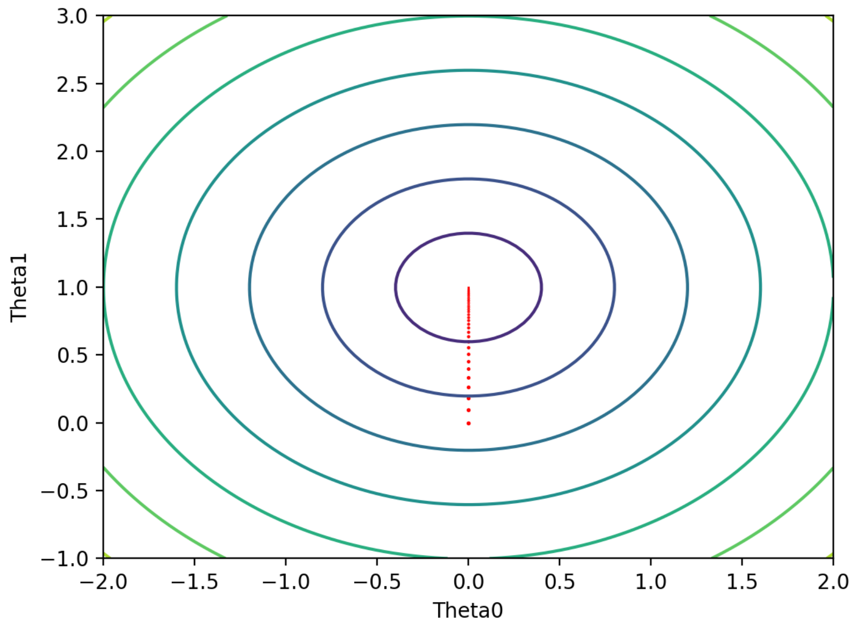


d) Contours :-

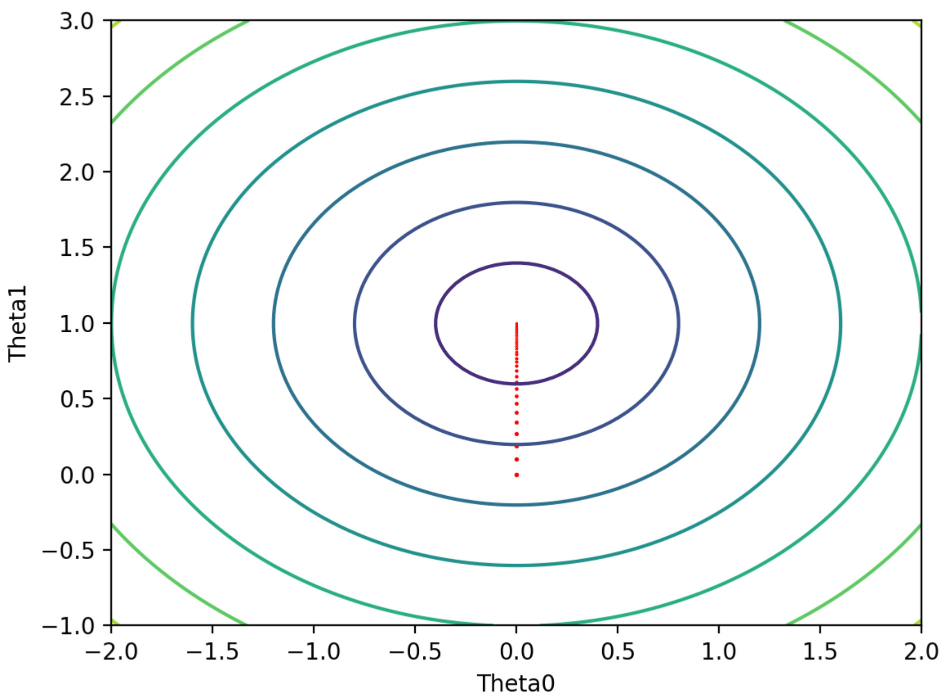
i) Learning Rate = 0.001



ii) Learning Rate $= 0.025$



iii) Learning Rate $= 0.1$



Observation :- Contours are nearly the same, but when I plot them dynamically, the initial increase of parameters is large for a large learning rate. And as the learning rate increases, the number of iterations to reach minima decreases.

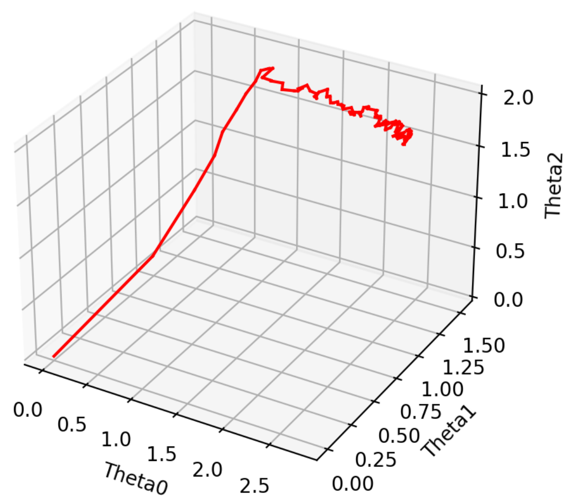
Q2)

a) Sampling done using random.sample function of numpy library.
b) Convergence criteria :-
   Avg of previous ‘k’ loss function - Avg of next ‘k’ loss function < Delta
   The value of ‘k’ is set such that as batch size decreases, ‘k’ should increase as the loss variation becomes noisy. I have kept the Delta as 10^{-4}.
   Value of ‘k’ :-
   i) r = 1 => 1000
   ii) r = 100 => 100
   iii) r = 10000 => 100
   iv) r = 1000000 => 1

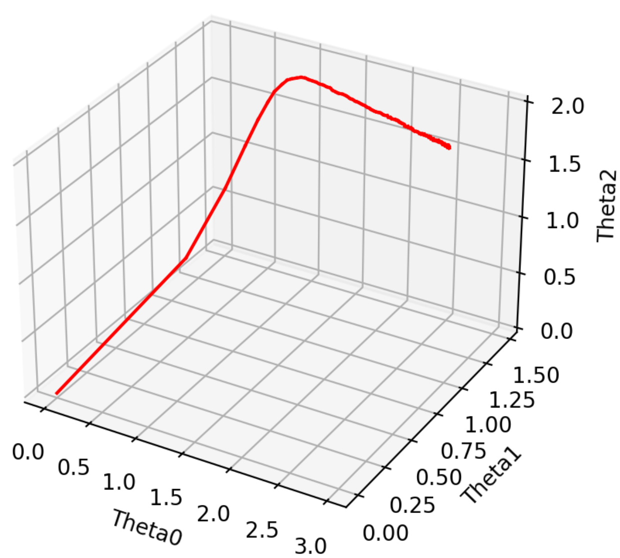
c) The value of parameters for batch size(1,100,10000) came close to the actual parameters([3,1,2]). The training and test set loss are close to 1, which is due to noise of the data(std=2). To get more accurate results, the value of delta has to be kept lower when batch size is large. When I kept the delta around 10^{-9} for k=1000000, the results were very good but we have to trade off training time.
   Parameter Values :-
   i) r = 1 => [3.04 1.02 1.98]
   ii) r = 100 => [2.977 1.004 1.989]
   iii) r = 10000 => [2.89 1.02 1.99]
   iv) r = 1000000 => [1.86 1.24 1.91]
   Speed of convergence was fastest when r=1000, and slowest when r=1. The reason is because updation in r=1 is after 1000 iterations, which makes the convergence slower.
   The total iterations done were -
   i) r = 1 => 19000
   ii) r = 100 => 17000
   iii) r = 10000 => 11900
   iv) r = 1000000 => 3300
   As batch size increases, number of iterations decreases.
   Error on the given dataset was -
   i) r = 1 => 1.018
   ii) r = 100 => 0.990
   iii) r = 10000 => 1.015
   iv) r = 1000000 => 4.70
   Test error on the original Hypothesis :- 0.9829

d) The graph of parameters -

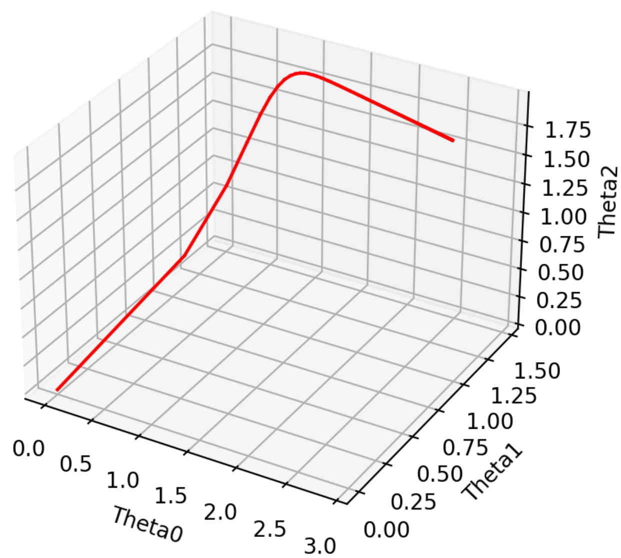
i) r = 1 =>



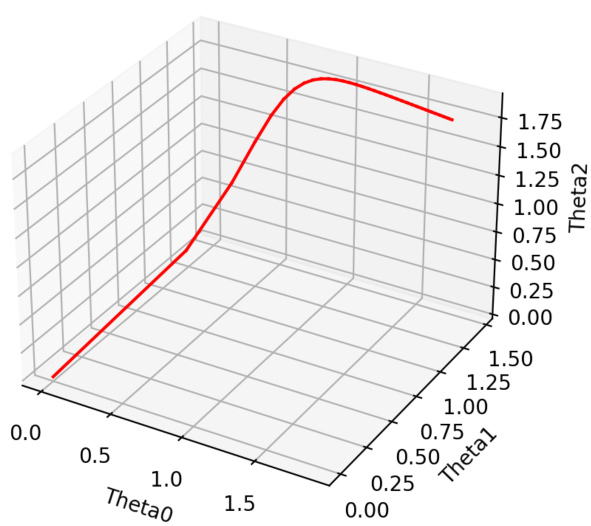
ii) r = 100 =>



iii) r = 10000 =>



iv) r = 1000000 =>



For smaller batch sizes, the graph is more noisy. This is because at each iteration, the algorithm optimises based on a small part of the dataset and does not move towards the actual minima. But the average does move towards minima.

Q3)

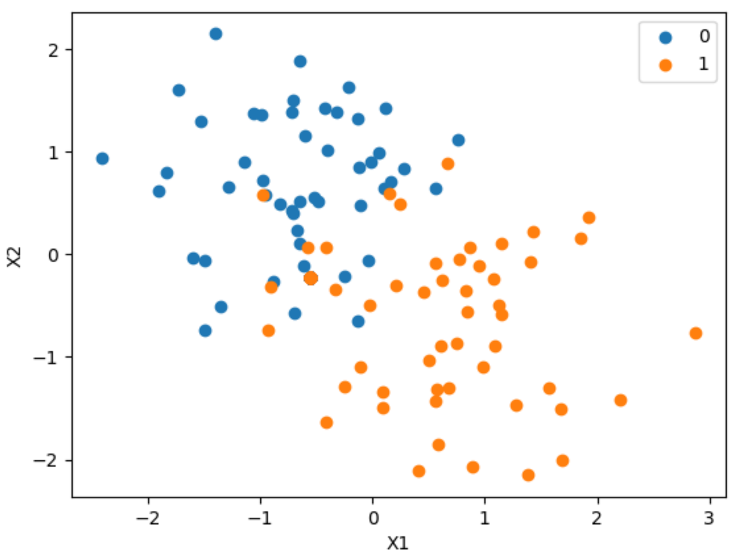
a) According to the hypothesis $y = \theta_0 + \theta_1 x_1 + \theta_2 x_2$, then

Final Parameters are: $\theta_0 = 0.401$ and $\theta_1 = 2.588$ and $\theta_2 =- 2.7255$.

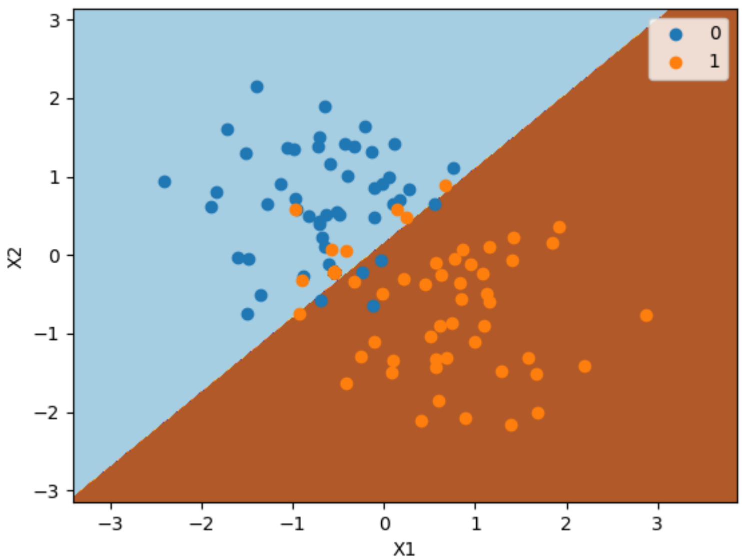
Condition for convergence :- previous_loss - current_loss $< 10^{-9}$

b) The region above the line shows $h_\theta(x) > 0.5$ and region below the line shows $h_\theta(x) < 0.5$.

Plot of Training Data :-



Plot with The Decision Boundary :-



Q4)

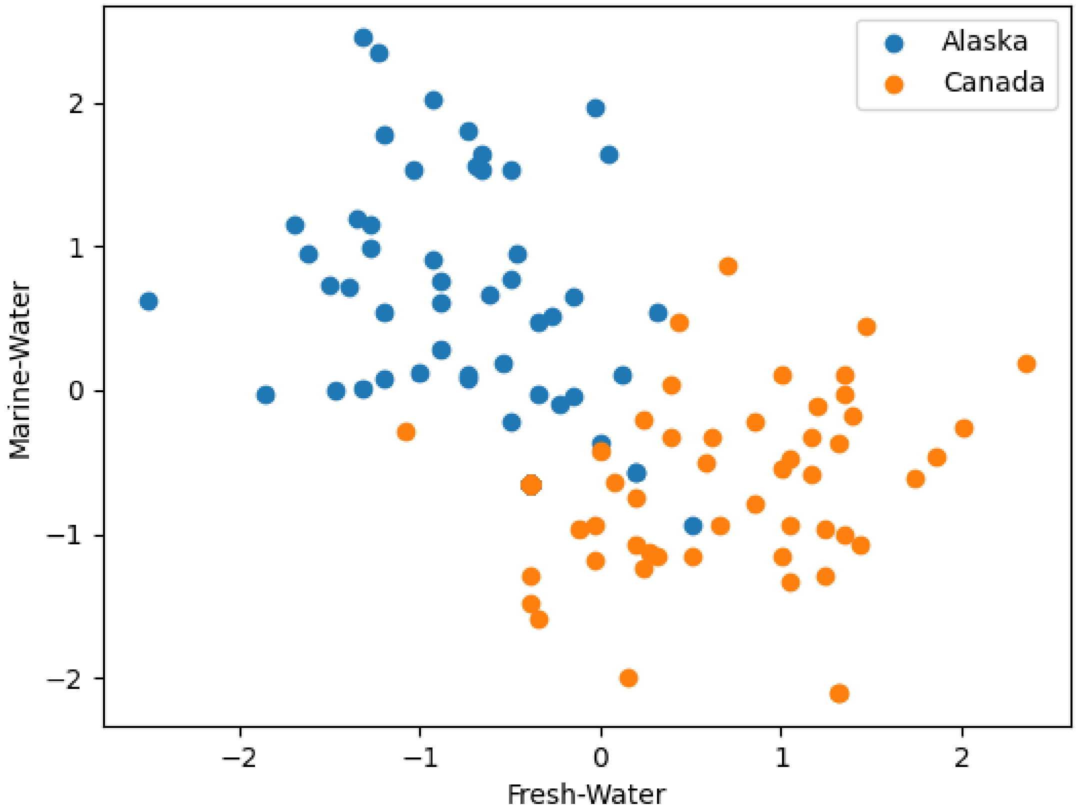
a) $\varphi = 0.5$

μ_0 = [-0.755 0.685]

μ_1 = [0.755 -0.685]

Σ = [[0.429 -0.224][-0.224 0.530]]

b) Training Data Plot :-

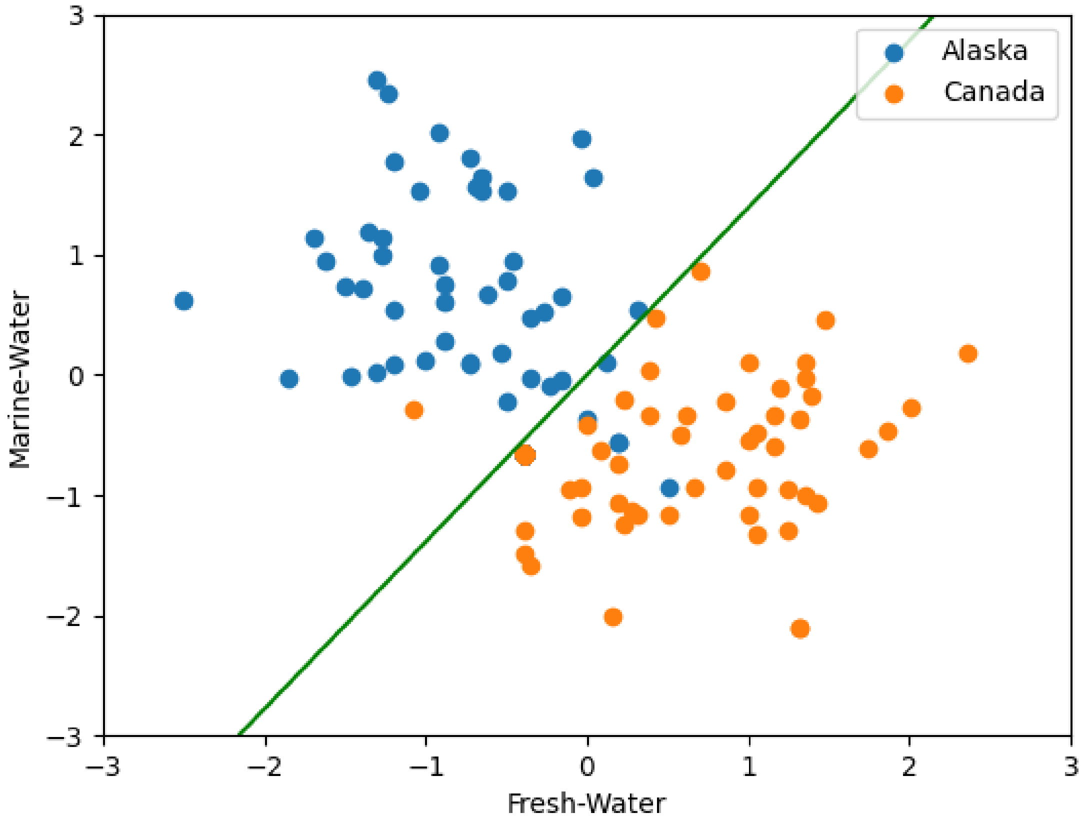


c) Equation of the decision boundary is :-

$$-(\mu_1-\mu_0)^T\Sigma^{-1}x + (1/2)(\mu_1^T\Sigma^{-1}\mu_1 - \mu_0^T\Sigma^{-1}\mu_0) + C = 0,$$

$Here,\ C = log((1-\varphi)/\varphi)$.

Plot with Decision Boundary :-



d) Σ_0 = [[0.381 -0.154][-0.154 0.647]]

Σ_1 = [[0.477 0.109][0.109 0.413]]

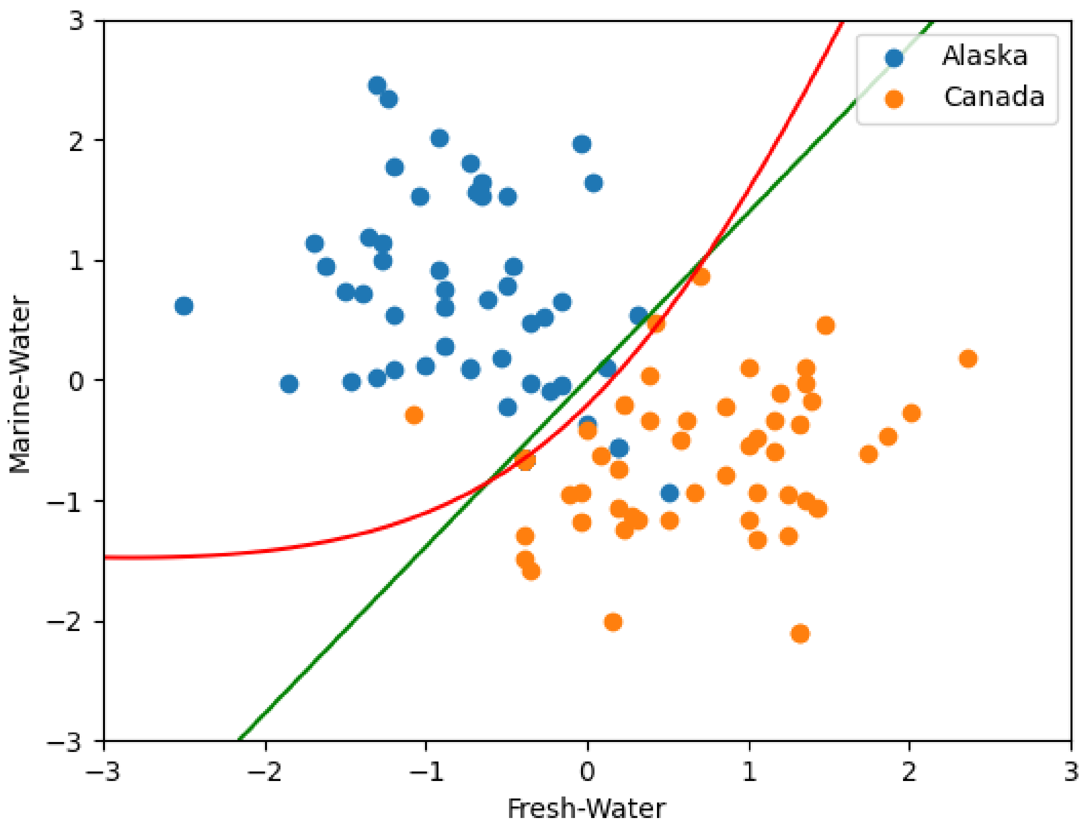
e) Equation of the decision boundary is :-

$(1/2)x^T(\Sigma_1^{-1} - \Sigma_0^{-1})x - (\mu_1\Sigma_1^{-1} - \mu_0\Sigma_0^{-1})^T x + (1/2)(\mu_1^T\Sigma_1^{-1}\mu_1 - \mu_0^T\Sigma_0^{-1}\mu_0) + C = 0$

,

$Here,\ C = log((1 - \varphi)/\varphi) + (1/2)log(\Sigma_1/\Sigma_0)$.

Plot with Decision Boundary :-



f) Both the linear and quadratic plots fit the data quite well. Except few, most training data points are classified correctly by both the plots. The only difference is that quadratic is a curve towards Alaska, and assigns more region to Canada. Whereas the linear plot is just a straight line.

Note : Resource used for plotting graph -

a) https://hackernoon.com/how-to-plot-a-decision-boundary-for-machine-learning-algorithms-in-python-3o1n3w07
b) https://www.geeksforgeeks.org/contour-plot-using-matplotlib-python/
c) Few other pages from geeksforgeeks