COL774: ASSIGNMENT 1

Name: Udayin Biswas
Entry Number: 2014CS50296

QUESTION-1:LINEAR REGRESSION

(A)Batch gradient descent method for optimizing J(θ)

1. Initialised the parameters as θ=0

2. Learning rate, n= 0.001

   Stopping criterion: | J(θ)[t+1] - J(θ)[t] | <= epsilon

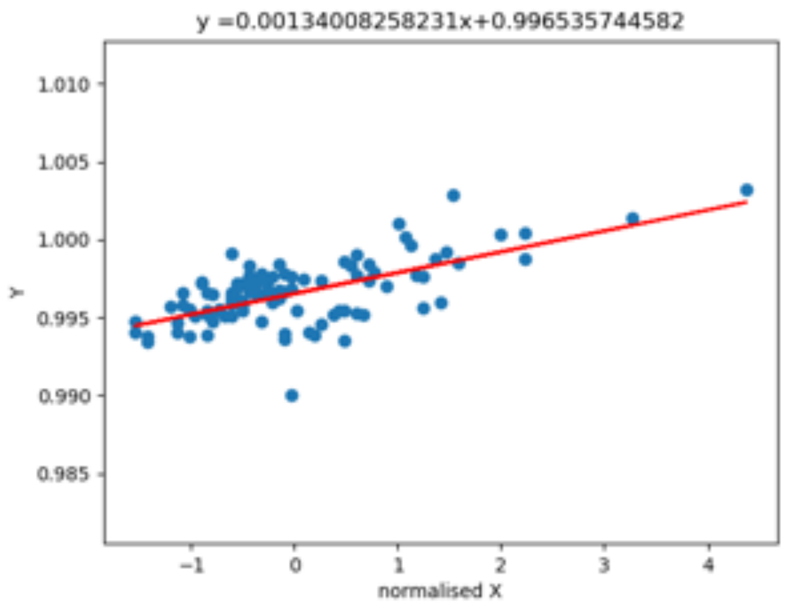
3. epsilon =0.0000001

$python linregAB.py

4. θ value

   θ0= 0.996535744582, θ1= 0.00134008258231

(B)Plot of data and hypothesis function



(E) OBSERVATIONS FOR DIFFERENT n

```
$python linregD.py <n>
```

1.n=0.001
For the epsilon=0.0000001, it takes 78 iterations for gradient descent method to satisfy convergence condition. Contour shrinks with each iteration.
2.n=0.005
For the epsilon=0.0000001, it takes 14 iterations for gradient descent method to satisfy convergence condition. Contour shrinks with each iteration.
3.n=0.009
For the epsilon=0.0000001, it takes 5 iterations for gradient descent method to satisfy convergence condition. Contour shrinks with each iteration.
4.n=0.013
For the epsilon=0.0000001, it takes 9 iterations for gradient descent method to satisfy convergence condition. Contour shrinks with each iteration.
5.n=0.017
For the epsilon=0.0000001, it takes 25 iterations for gradient descent method to satisfy convergence condition. Contour shrinks with each iteration.
6.n=0.021 and n=0.025
For the epsilon=0.0000001, it doesn't seem to converge and the contours infact grow in size with time instead of shrinking. The gradient descent method didn't seem to converge and got stuck satisfying the convergence condition.

So we can observe that for convergence condition, we should not choose a very large n since then it may not converge and infact move further away from the minimum point. Also, it is wise to choose a n which minimises the time to reach the convergence point, i.e. takes least number of iterations to give an accurate estimate of value of θ.
In this case, n=0.009 seems to be the ideal choice.

QUESTION-2: LOCALLY WEIGHTED LINEAR REGRESSION

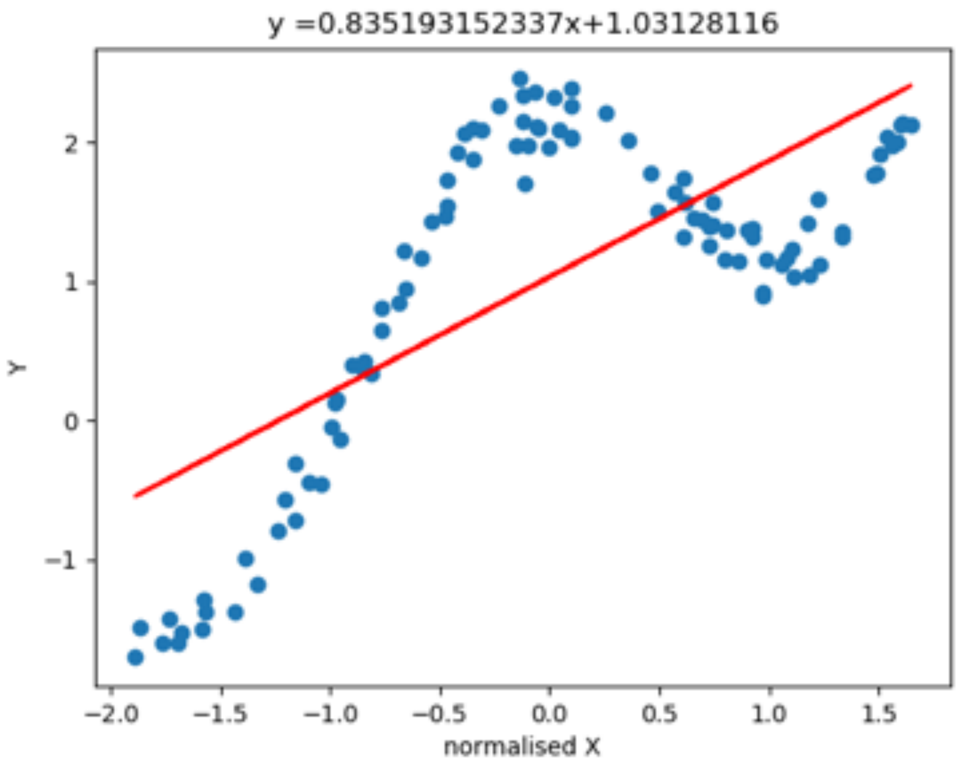
(A) UNWEIGHTED LINEAR REGRESSION USING NORMAL EQUATIONS

Unweighted linear regression was implemented using the following normal equation:

$\theta = (X^T X)^{-1} X^T Y$

```
$ python unweighted.py
```

We obtain the following plot of the data and the straight line derived:



θ = [1.03128116 , 0.83519315237] ~ [1.03,0.84]

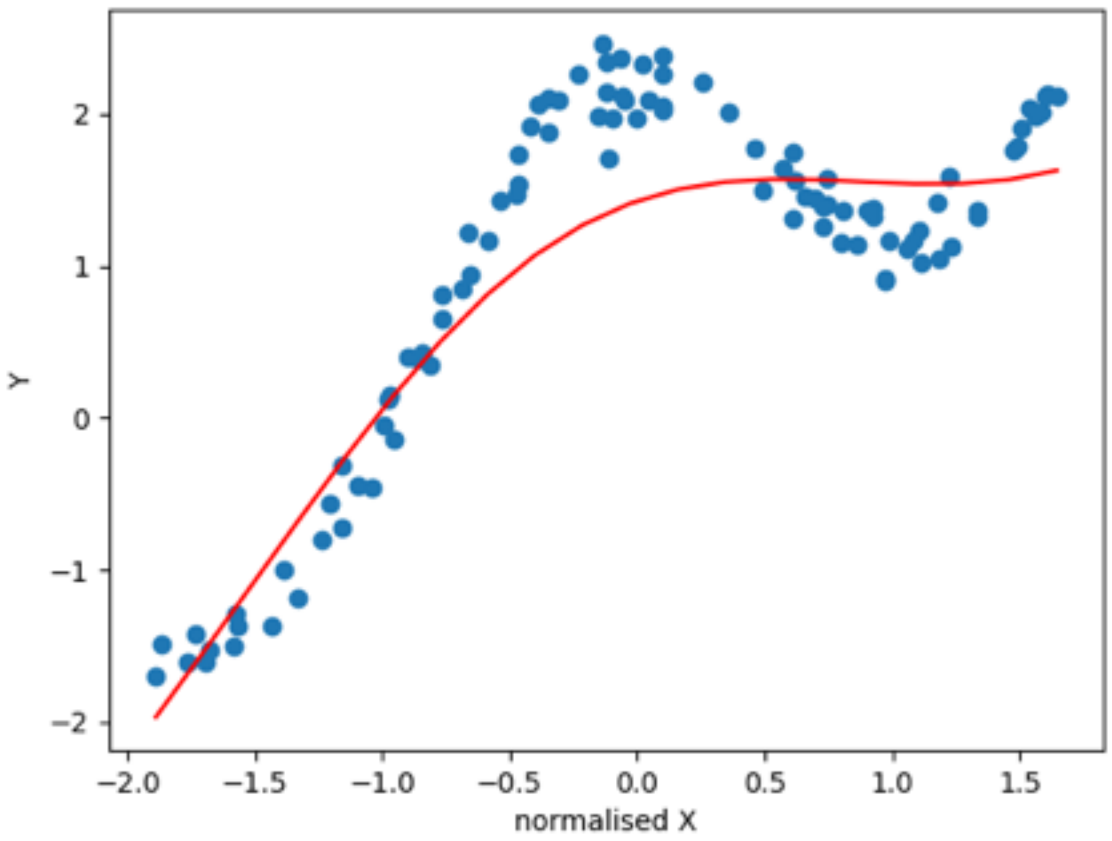
So θ_0 = 1.03, θ_1=0.84

(B) WEIGHTED LINEAR REGRESSION USING NORMAL EQUATIONS

Weighted linear regression was implemented using the following normal equation to derive θ :-

$\theta = (X^T W X)^{-1} X^T W Y$

$python locallyweighted.py 0.8

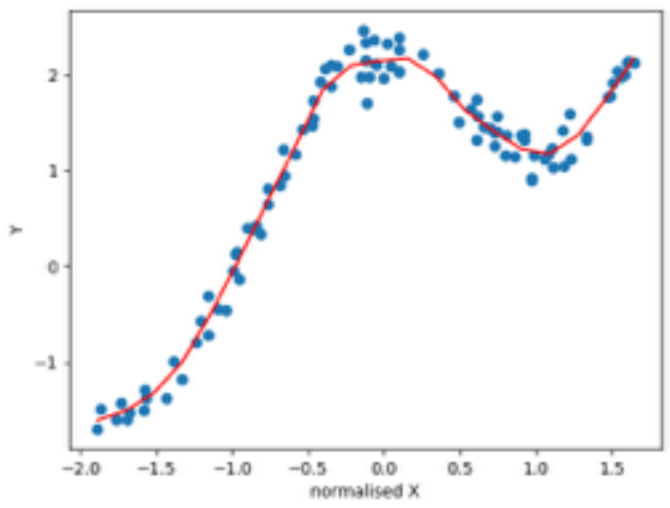


(1)Setting bandwidth parameter: **τ = 0.8**.
(2) Choosing query points by:
np.linspace(np.amin(normalisedX),np.amax(normalisedX),num=20)

This helps us choose 20 query points on the normalised X axis between the max value of normalised x of training data and the min value. We calculate the weighted components for all normalised training data , about one point at a time from these query points. This gives us a θ value corresponding to each query point- so we get a line corresponding to each θ value. Computing hθ(query point) will give its y value. Obtain these y values of each query point and join them to get the curve.

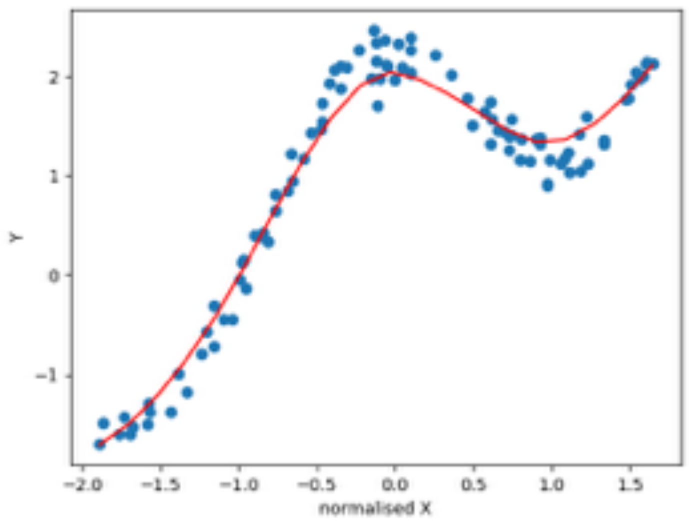
(C) REPEATING (B) WITH DIFFERENT τ

1.τ = 0.1



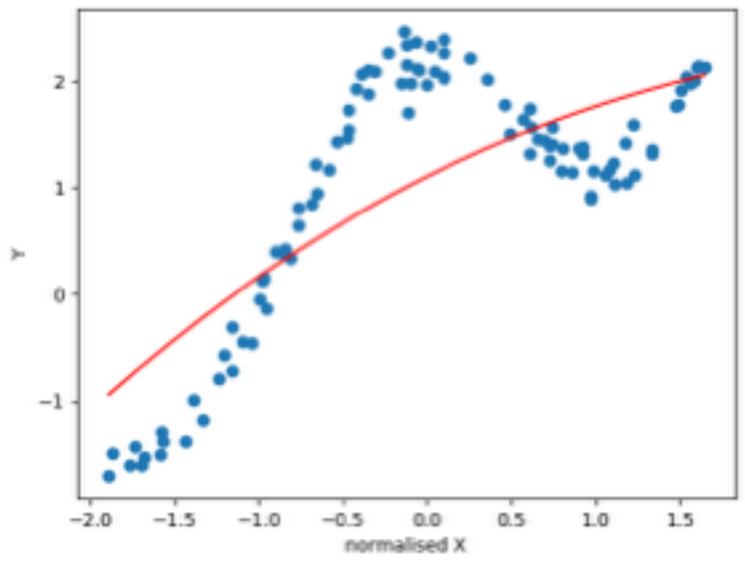
Y
2
1
0
−1
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
normalised X

1.τ = 0.3

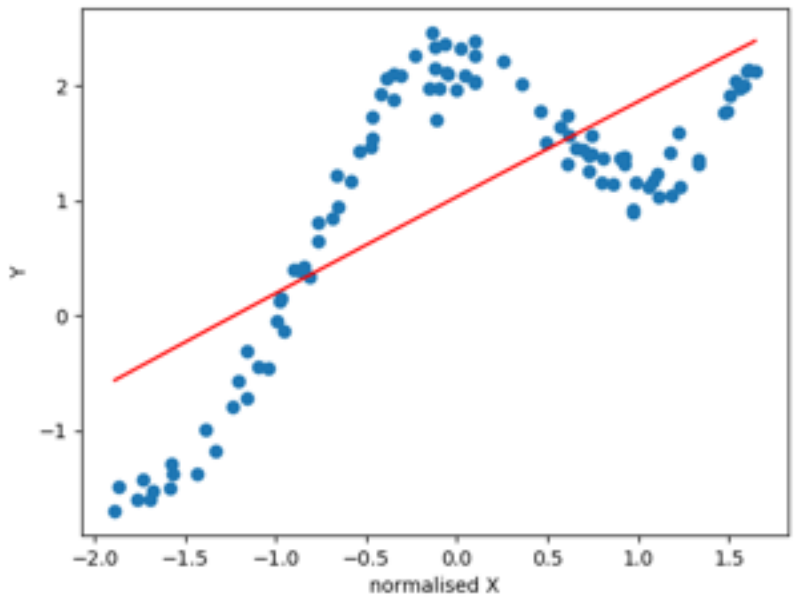


Y
2
1
0
−1
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
normalised X

<u>1.τ = 2</u>



<u>1.τ = 10</u>



I think when τ = 0.3 the curve best represents the training data. This is because it doesn't look underfitted as well as overfitted, and it more or less is an optimal representation of the data.
When the τ is large(2 or 10)cd ../, it approaches unweighted linear regression and hence we get a straight line. From the example, we see

that the line doesn't represent the data points scattered accurately and hence **underfitted.**
When the τ is small(0.1), it means it will be **overfitted** and will take noise of the points into too much consideration. This is not ideal when we want to represent the training data by a curve for future predictions, instead we would like a more generalised model.

QUESTION-3: LOGISTIC REGRESSION

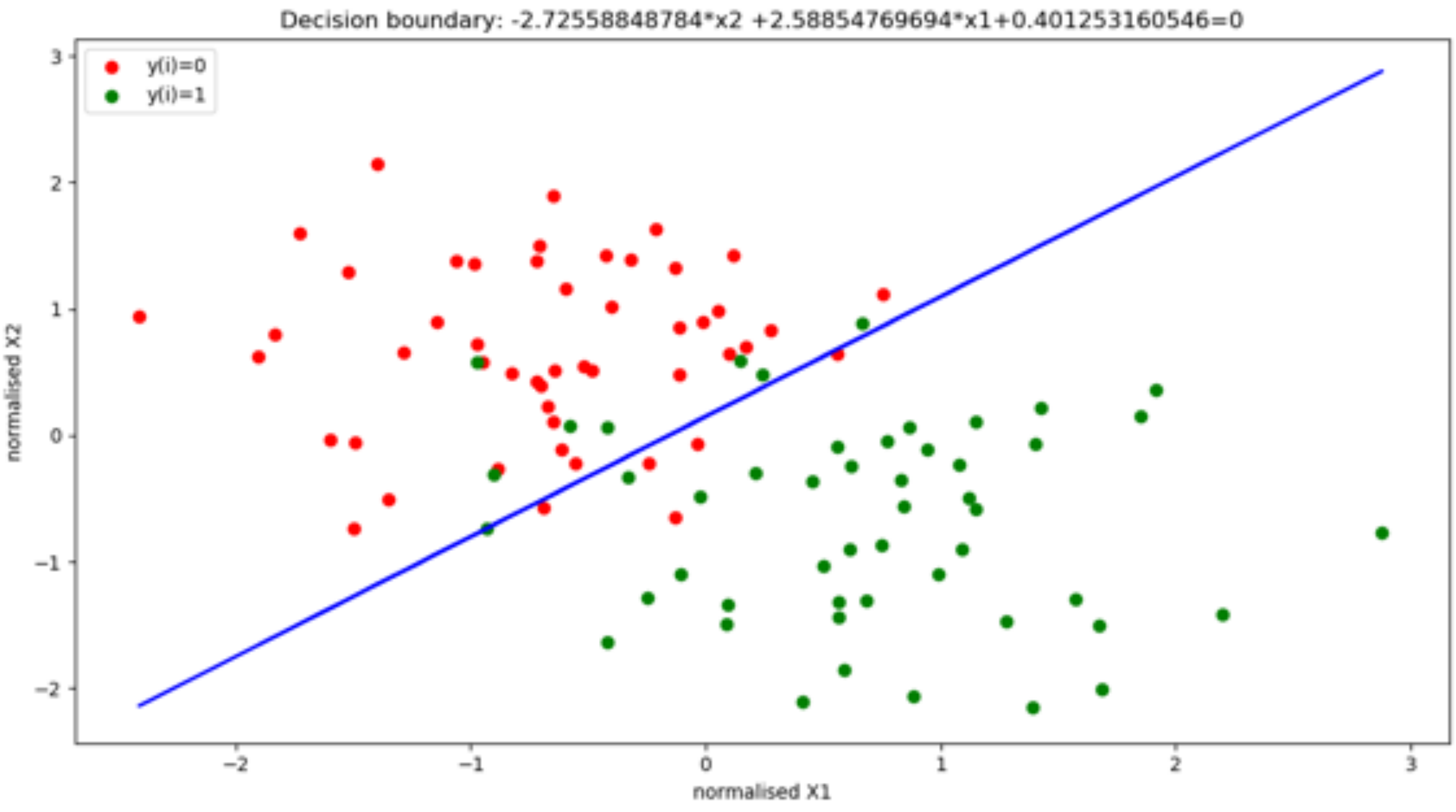
(a) Implementing logistic regression by Newton method

```
$python logistic.py
```

1.Initialize Newton's method with $\theta = \vec{0}$ (the vector of all zeros)
2.Applying θ(t+1)= θ(t)- H.g where H is the Hessian matrix and g is the gradient at θ(t) of h θ(x)/g(z).
3. Values obtained:
   θ2= -2.72558848784
   θ1= 2.58854769694
   θ0= 0.401253160546

(a) Plotting data points and the decision boundary

Decision boundary: -2.72558848784*x2+ 2.58854769694*x1+0.401253160546



QUESTION-4:GAUSSIAN DISCRIMINANT ANALYSIS

(a) Implementing GDA when both classes have same covariance matrix

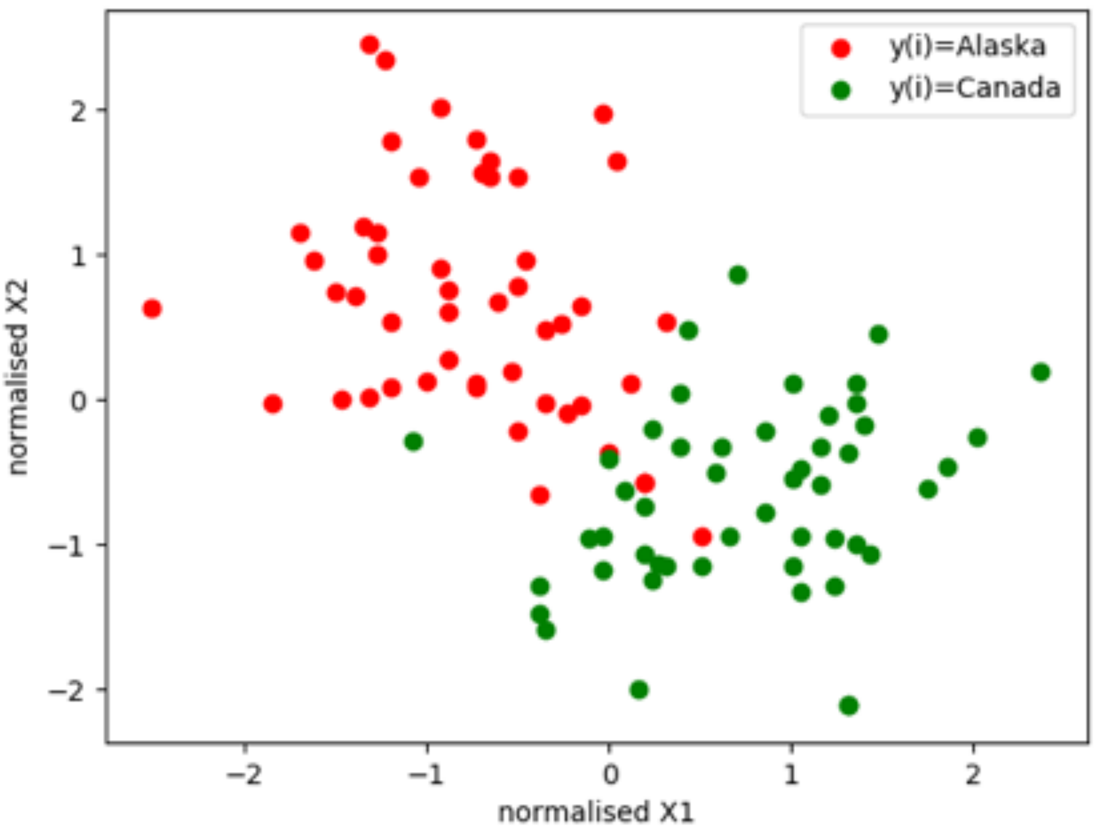
$$\mu_0 = \frac{\sum_{i=1}^{m} 1\{y^{(i)}=0\}\, x^{(i)}}{\sum_{i=1}^{m} 1\{y^{(i)}=0\}}$$

$$\mu_1 = \frac{\sum_{i=1}^{m} 1\{y^{(i)}=1\}\, x^{(i)}}{\sum_{i=1}^{m} 1\{y^{(i)}=1\}}$$

$$\Sigma = \frac{1}{m} \sum_{i=1}^{m} (x^{(i)} - \mu_{y^{(i)}})(x^{(i)} - \mu_{y^{(i)}})^T \quad (\Sigma_0 = \Sigma_1)$$

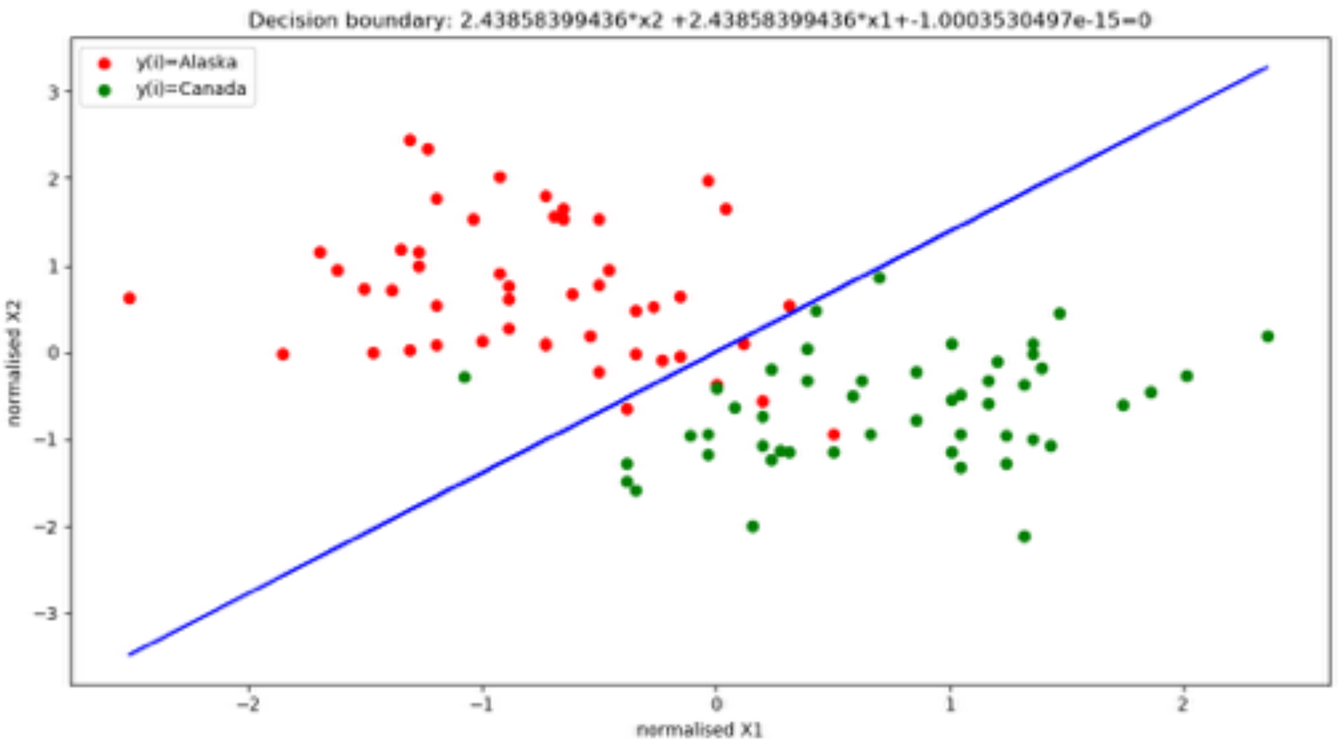
1. $\Sigma_0 = \Sigma_1 = \Sigma$
2. μ0 = [-0.75529433 0.68509431] (for Alaska)
3. μ1 = [0.75529433 -0.68509431] (for Canada)
4. Covariance= Σ=[[0.42953048 -0.02247228]
   [-0.02247228 0.53064579]]

<u>(b) PLOT OF TRAINING DATA</u>



(C) LINEAR SEPARATOR PLOT

$$\log\left(\frac{\phi}{1-\phi}\right) - \frac{1}{2}(\mu_1+\mu_0)^T \Sigma^{-1}(\mu_1-\mu_0) + x^T \Sigma^{-1}(\mu_1-\mu_0) = 0$$



(D) Implementing GDA when both classes have different covariance matrix

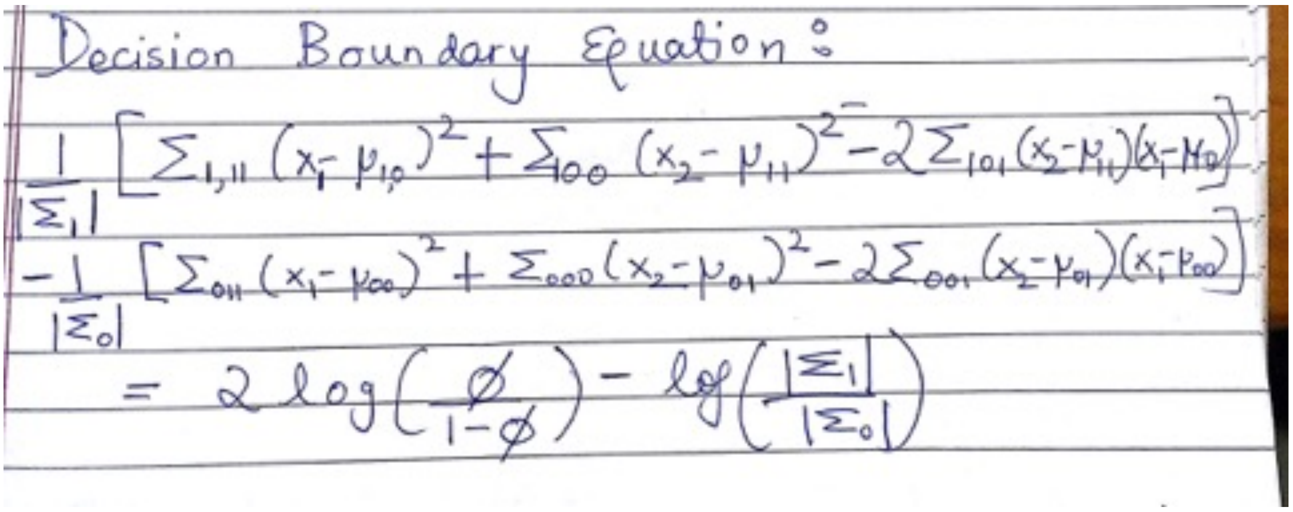
$$\Sigma_0 = \frac{\sum_{i=1}^{m} 1\{y^{(i)}=0\}\,[x^{(i)}-\mu_0][x^{(i)}-\mu_0]^T}{\sum_{i=1}^{m} 1\{y^{(i)}=0\}}$$

$$\Sigma_1 = \frac{\sum_{i=1}^{m} 1\{y^{(i)}=1\}\,[x^{(i)}-\mu_1][x^{(i)}-\mu_1]^T}{\sum_{i=1}^{m} 1\{y^{(i)}=1\}}$$

1.Σ_0 ! = Σ_1
2. μ0 = [-0.75529433 0.68509431] (for Alaska)

3. µ1 = [0.75529433 -0.68509431] (for Canada)
4.Covariance, Σ_0= [[0.38158978 -0.15486516]
[-0.15486516 0.64773717]]
5.Covariance, Σ_1=[[0.47747117 0.1099206]
[0.1099206   0.41355441]]

(E) QUADRATIC DECISION BOUNDARY PLOT

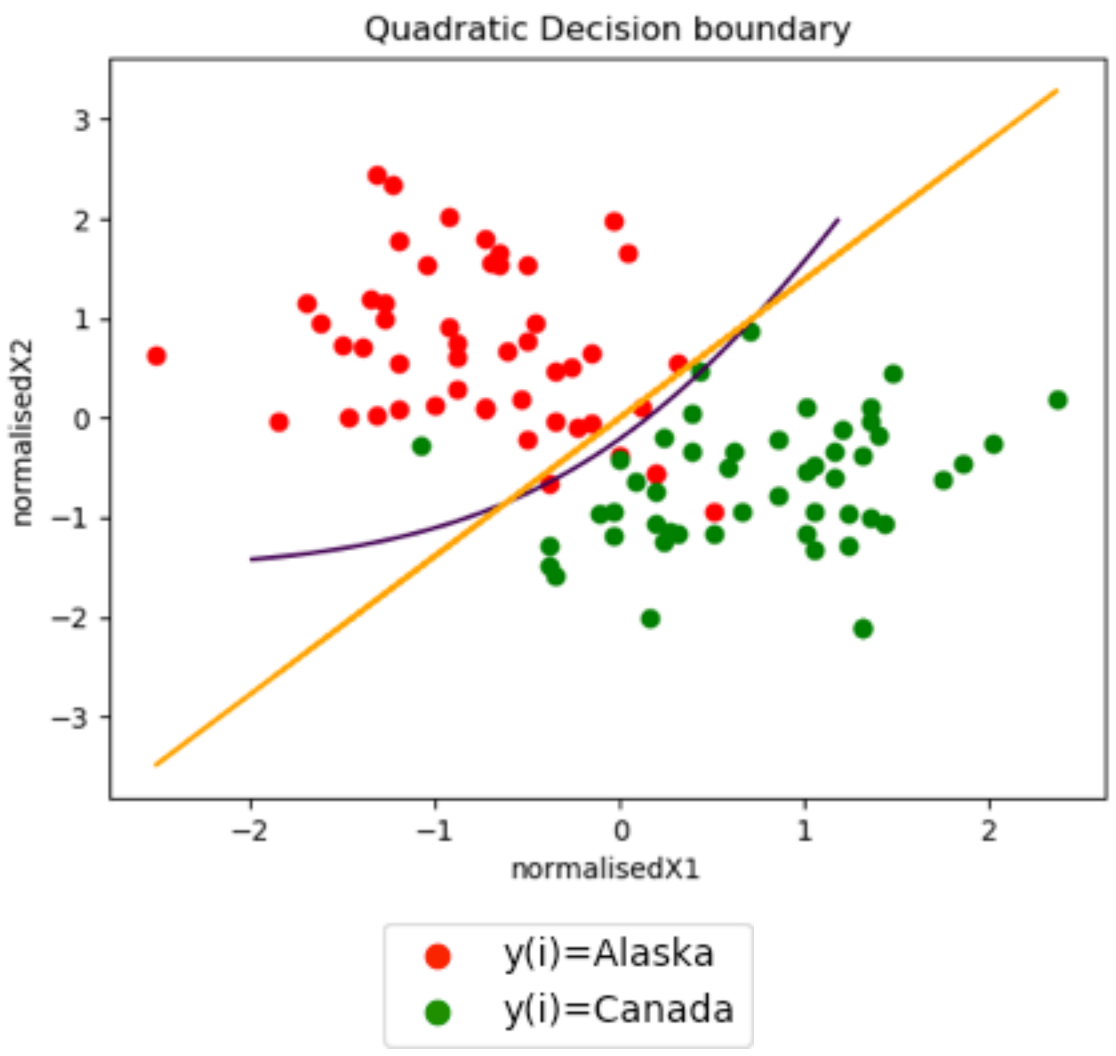


Decision Boundary Equation :

$$\frac{1}{|\Sigma_1|}\left[\Sigma_{1,11}(x_1-\mu_{1,0})^2+\Sigma_{100}(x_2-\mu_{11})^2-2\Sigma_{101}(x_2-\mu_{11})(x_1-\mu_{10})\right]$$
$$-\frac{1}{|\Sigma_0|}\left[\Sigma_{011}(x_1-\mu_{00})^2+\Sigma_{000}(x_2-\mu_{01})^2-2\Sigma_{001}(x_2-\mu_{01})(x_1-\mu_{00})\right]$$
$$=2\log\left(\frac{\phi}{1-\phi}\right)-\log\left(\frac{|\Sigma_1|}{|\Sigma_0|}\right)$$

On substituting the means and covariances, we get:

(((0.413554410787*(x1-0.755294327991)^2)+(0.477471172592*(x2--0.685094305549)^2)-(0.219841198664*(x2--0.685094305549)*(x1-0.755294327991)))/0.185377771292)-(((0.647737174222*(x1--0.755294327991)^2)+(0.381589783617*(x2-0.685094305549)^2)-(-0.309730316573*(x2-0.685094305549)*(x1--0.755294327991)))/0.223186670901)=0.18561276054



(F) ANALYSIS

After analysing the linear and quadratic boundaries, we can say that both of them can help us distinguish between the two types of data present in the training data,i.e. belong to Alaska or Canada. But quadratic help us distinguish between the two types of data present in the training data,i.e. belong to Alaska or Canada. Linear boundary is not inclined to any one of the classes i.e. Alaska or Canada since both the classes have the same covariance matrix and hence the same variability.
But we observe that the quadratic boundary curve is more inclined toward Alaska class than Canada class and bends towards Alaska class. This is because Alaska class has a greater variability in its data than Canada class as can be observed from the covariance matrices.