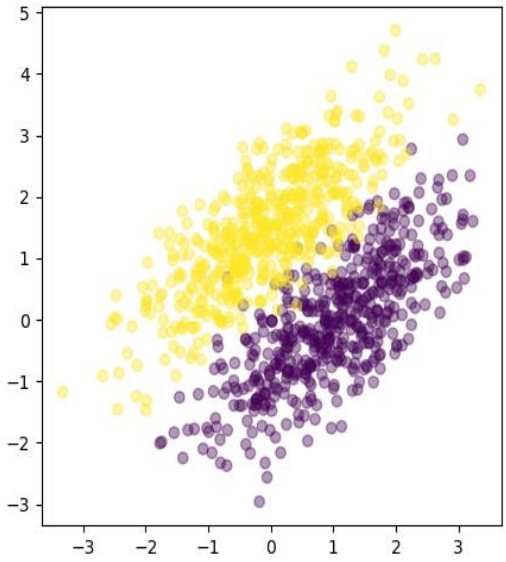
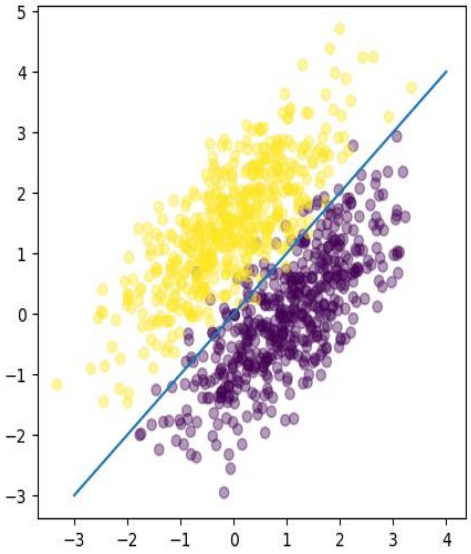
Data Mining CSE-5334-001
Assignment 3- Report

Name: Swati Sriram Mani
UTA Id: 1001648136

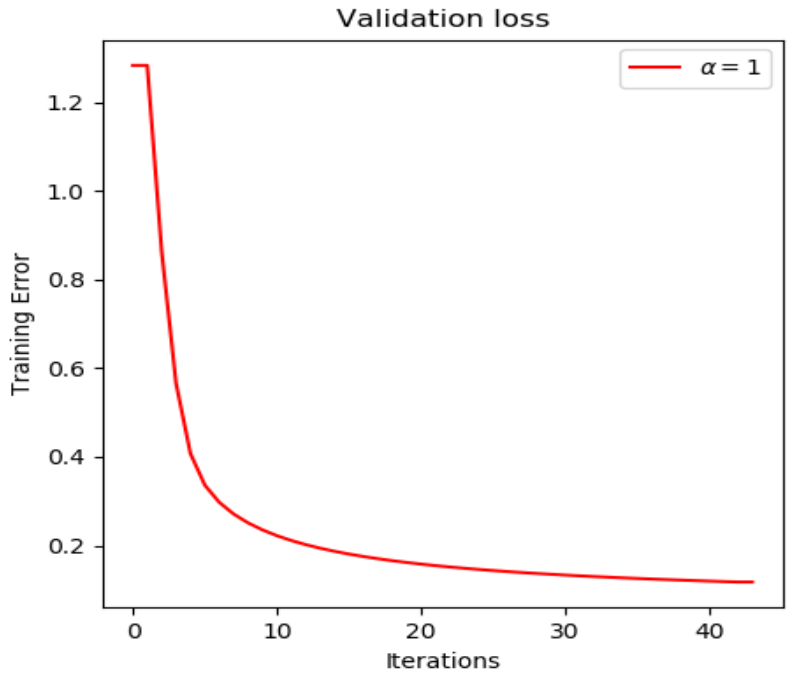
Question 1(Batch Training)

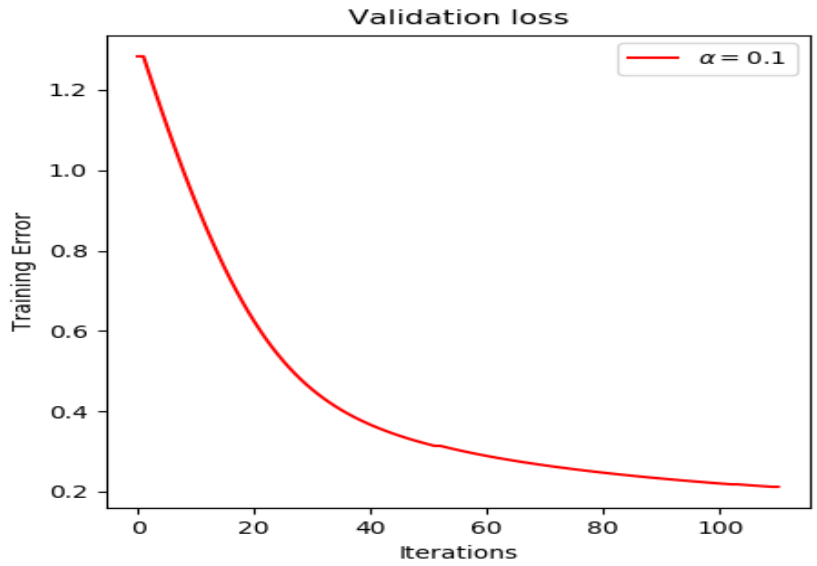
a)**ScatterPlot for the testing data and decision boundary:**

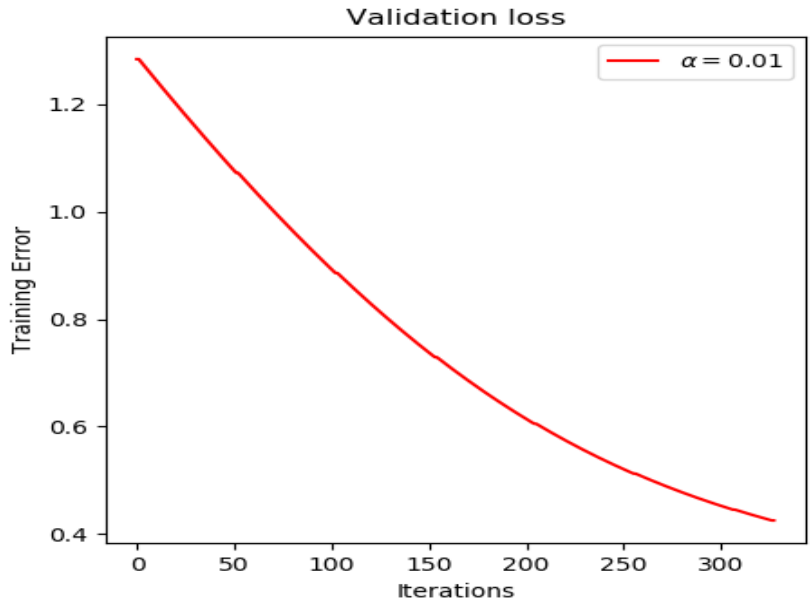


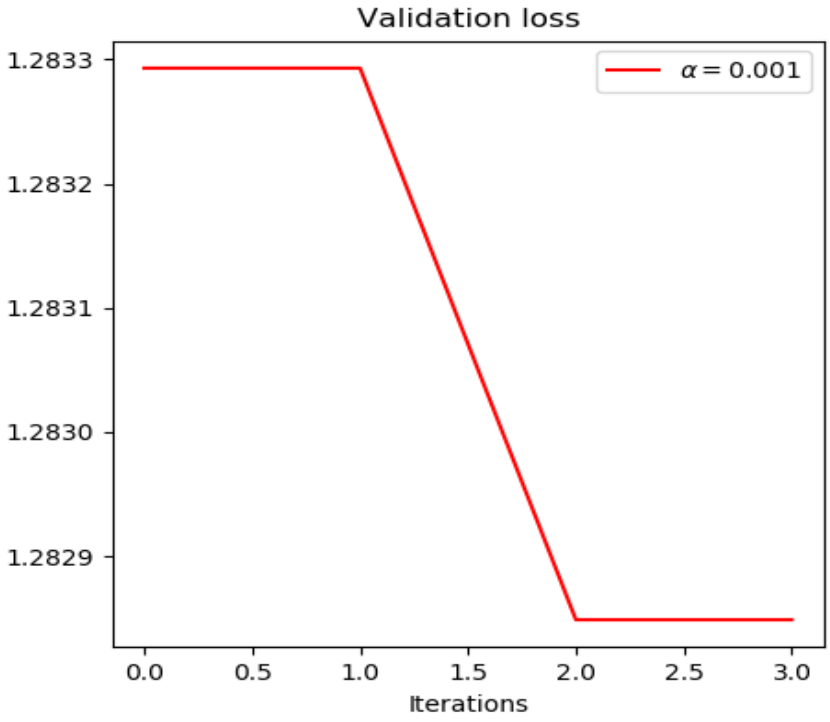


b)**Figure of change of training error(cross entropy) with respect to iteration**

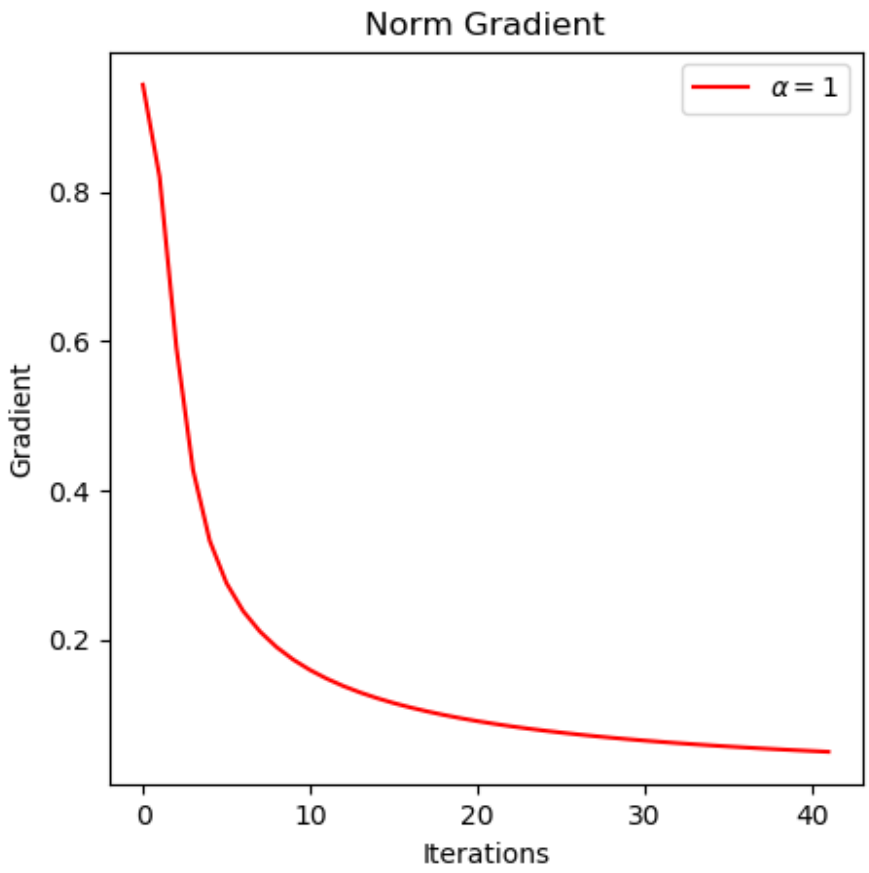


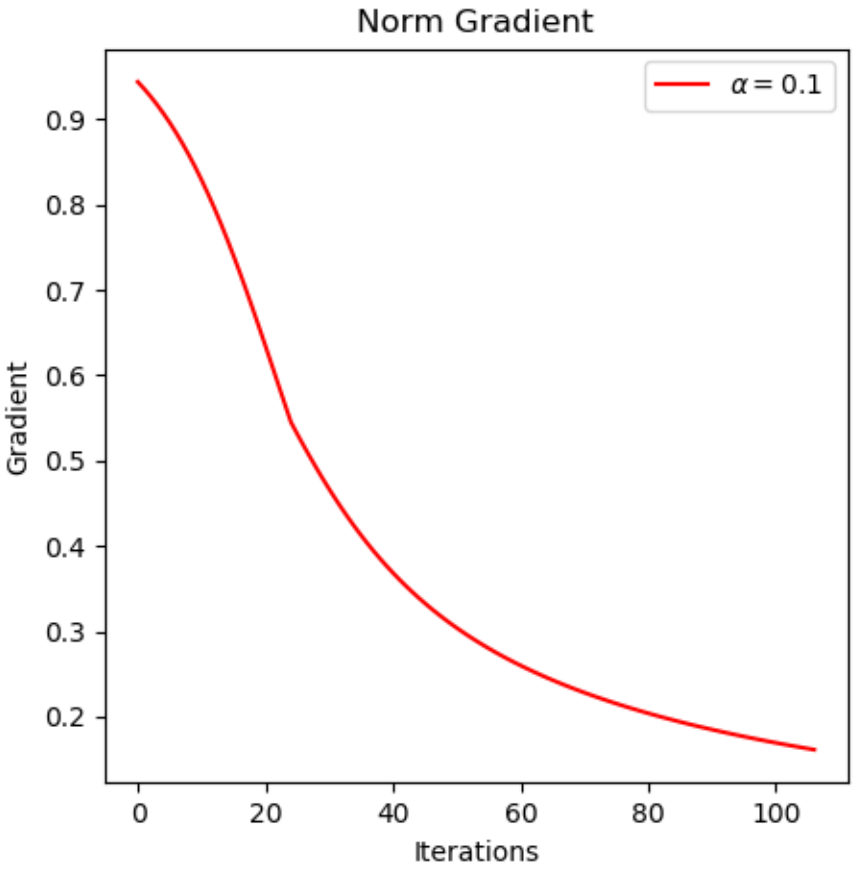


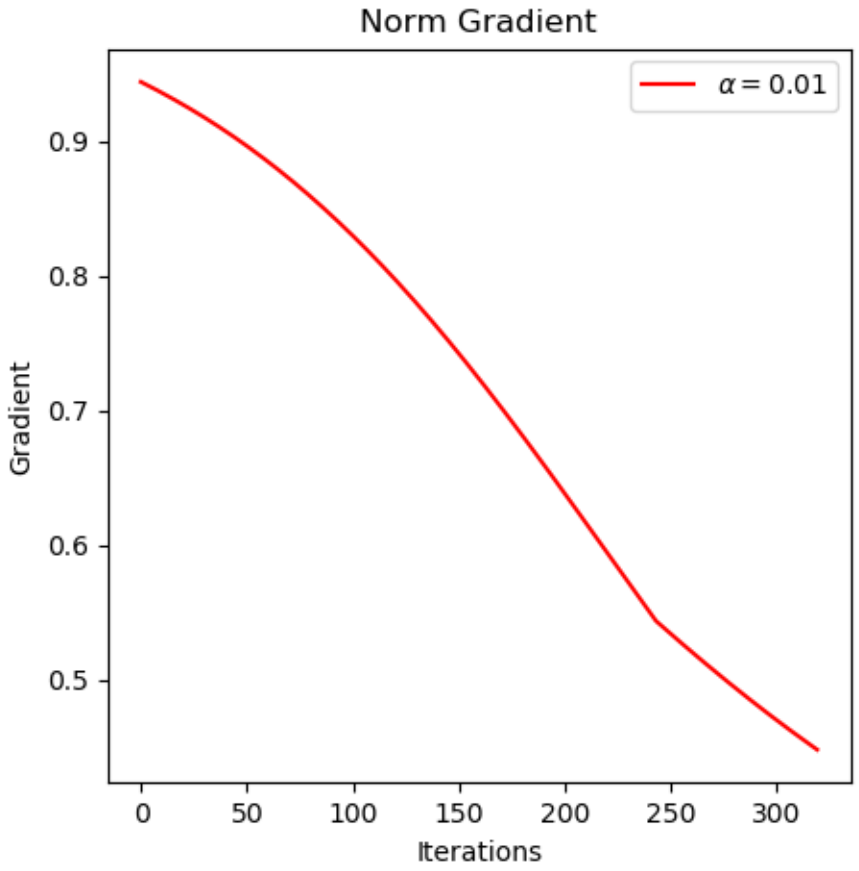

Validation loss
α = 0.01
Training Error
1.2
1.0
0.8
0.6
0.4
0
50
100
150
200
250
300
Iterations

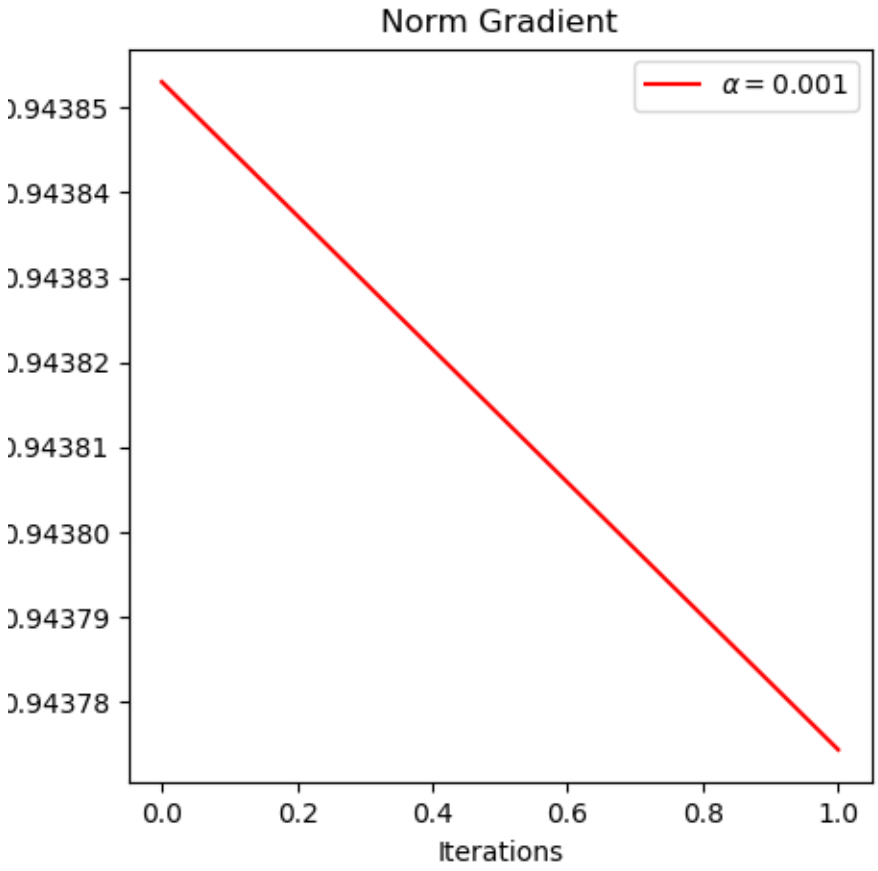

Validation loss
α = 0.001
1.2833
1.2832
1.2831
1.2830
1.2829
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Iterations

c)**Figure of changes of norm of gradient with respect to iteration:**




Norm Gradient
$\alpha = 0.1$
Gradient
Iterations
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
20
40
60
80
100


Norm Gradient
$\alpha = 0.01$
Gradient
0.9
0.8
0.7
0.6
0.5
0
50
100
150
200
250
300
Iterations


Norm Gradient
$\alpha = 0.001$
0.94385
0.94384
0.94383
0.94382
0.94381
0.94380
0.94379
0.94378
0.0
0.2
0.4
0.6
0.8
1.0
Iterations

d)**Output:**

Learning rate (Alpha): 1

Total Epochs: 100000

Cross - Entropy loss at epoch 1: 1.2832928981145257

Cross - Entropy loss at epoch 42: 0.11766635814457527

Loss optimized is less than threshold!

total no. of iterations run: 42

Final Weights: [-0.5748971 -2.52126031 2.6016152]

Accuracy: 96.6

AUC: 0.801874

Learning rate (Alpha): 0.1

Total Epochs: 100000

Cross - Entropy loss at epoch 1: 1.2832928981145257

Cross - Entropy loss at epoch 51: 0.3138616055813652

Cross - Entropy loss at epoch 101: 0.2180976107386285

Cross - Entropy loss at epoch 107: 0.21192710223337474

Loss optimized is less than threshold!

total no. of iterations run: 107

Final Weights: [-0.15408701 -1.4146219   1.44444363]

Accuracy: 96.6

AUC: 0.794238

Learning rate (Alpha): 0.01

Total Epochs: 100000

Cross - Entropy loss at epoch 1: 1.2832928981145257

Cross - Entropy loss at epoch 51: 1.0718796558395716

Cross - Entropy loss at epoch 101: 0.8855175344250131

Cross - Entropy loss at epoch 151: 0.729175545860614

Cross - Entropy loss at epoch 201: 0.6054200037023261

Cross - Entropy loss at epoch 251: 0.5127132947132632

Cross - Entropy loss at epoch 301: 0.4457429835327352

Cross - Entropy loss at epoch 320: 0.42562535373315247

Loss optimized is less than threshold!

total no. of iterations run: 320

Final Weights: [0.27052635 -0.44511361 0.92844834]

Accuracy: 79.5

AUC: 0.45849

Learning rate (Alpha): 0.001

Total Epochs: 100000

Cross - Entropy loss at epoch 1: 1.2832928981145257

Cross - Entropy loss at epoch 2: 1.2828489076520277

Loss optimized is less than threshold!

total no. of iterations run: 2

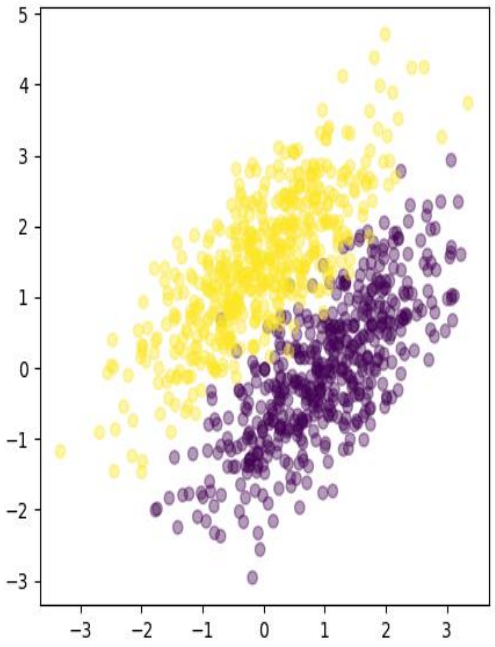
Final Weights: [0.99939482 0.99881688 0.99990068]

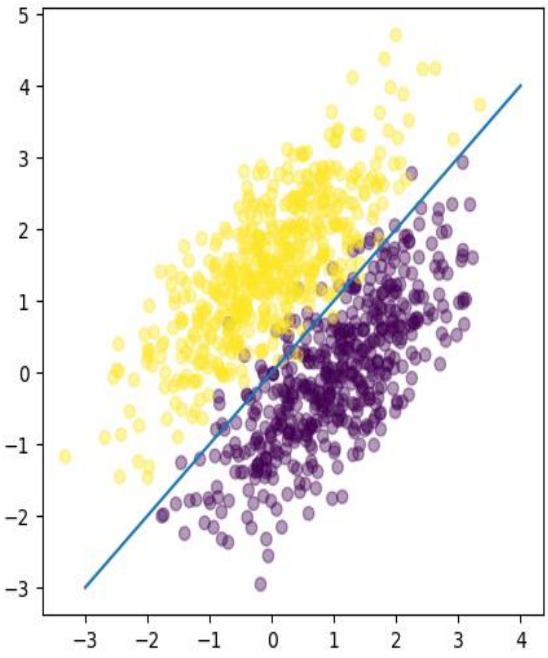
Accuracy: 49.9

AUC: 0.0

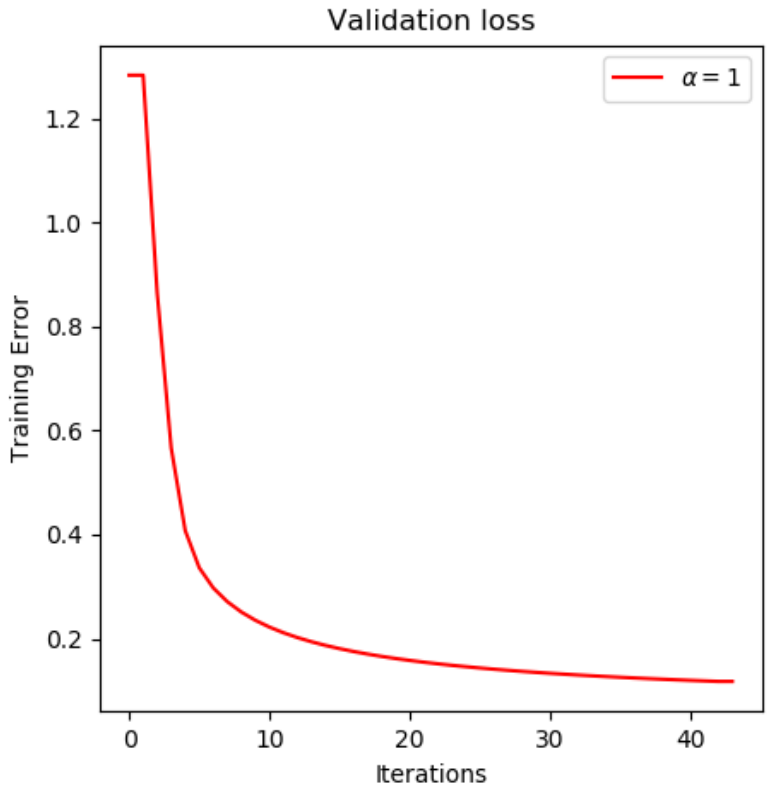
Online Training:

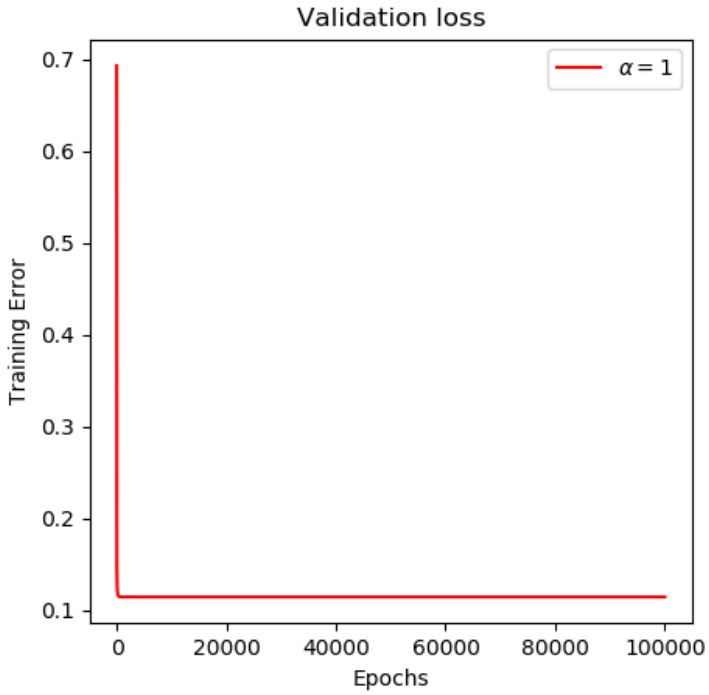
a) **ScatterPlot for the testing data and decision boundary:**

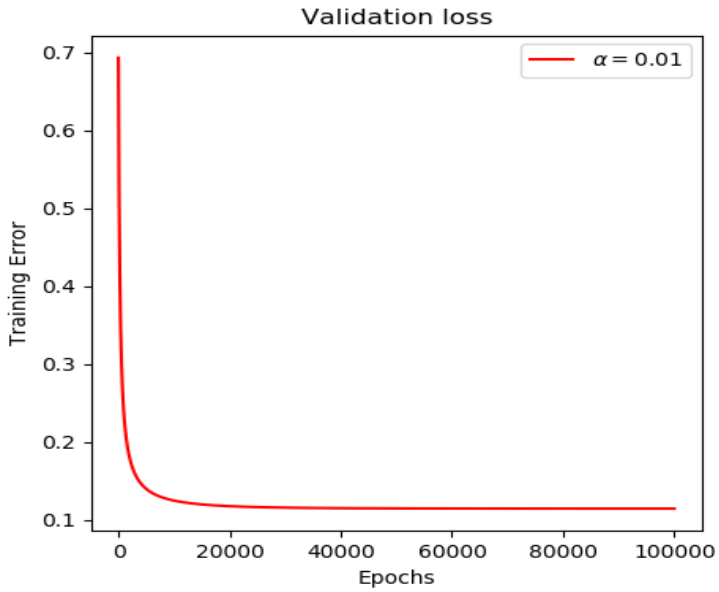


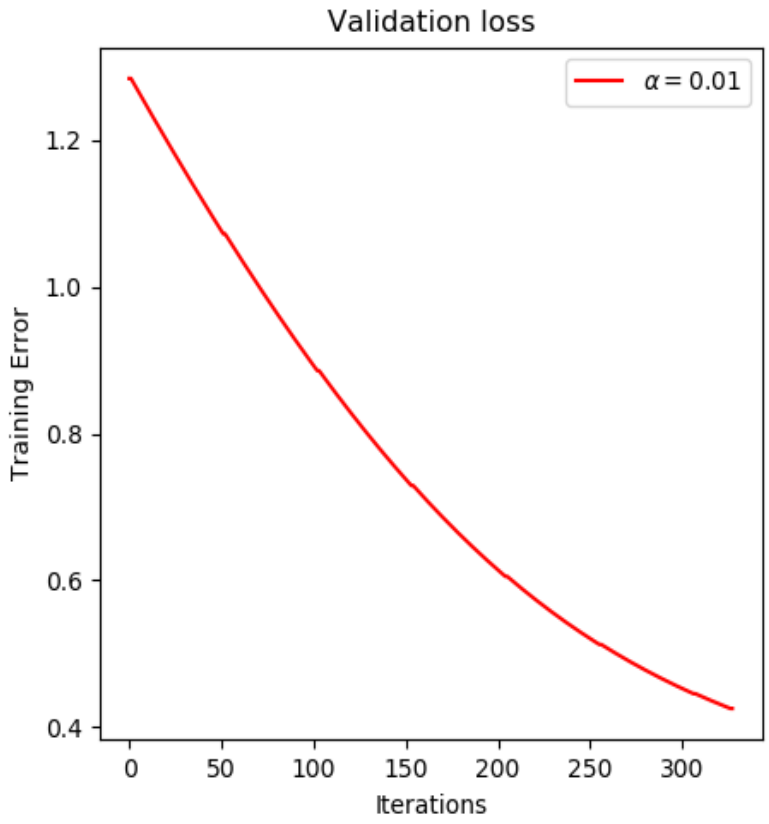


b) **Figure of change of training error(cross entropy) with respect to iteration**

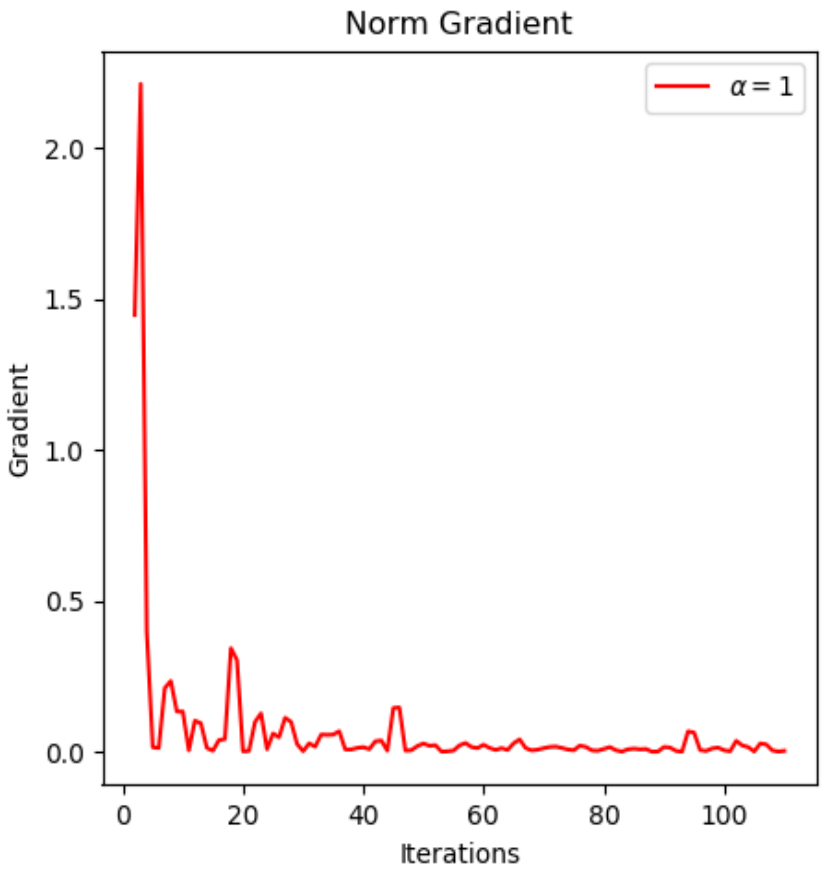


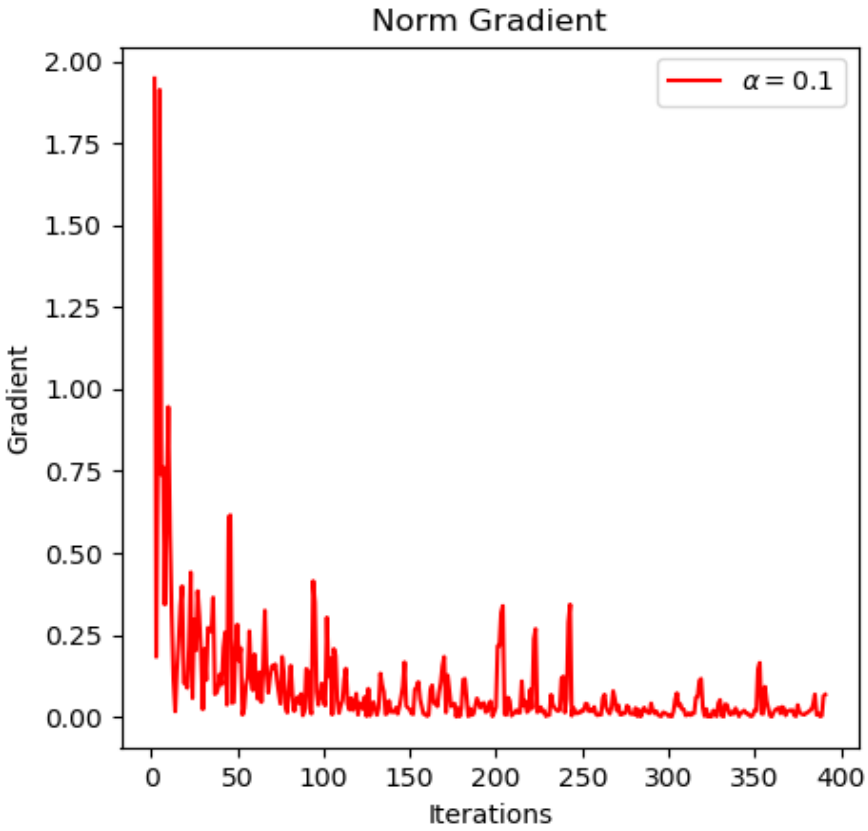


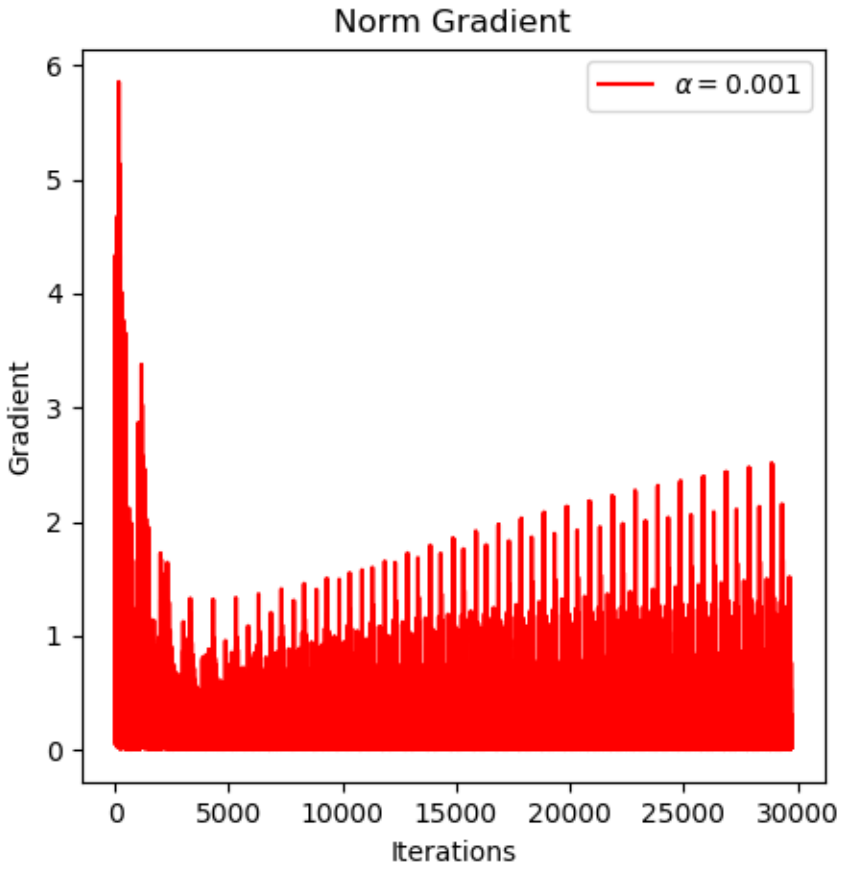

Validation loss
$\alpha = 0.01$
Training Error
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
20000
40000
60000
80000
100000
Epochs

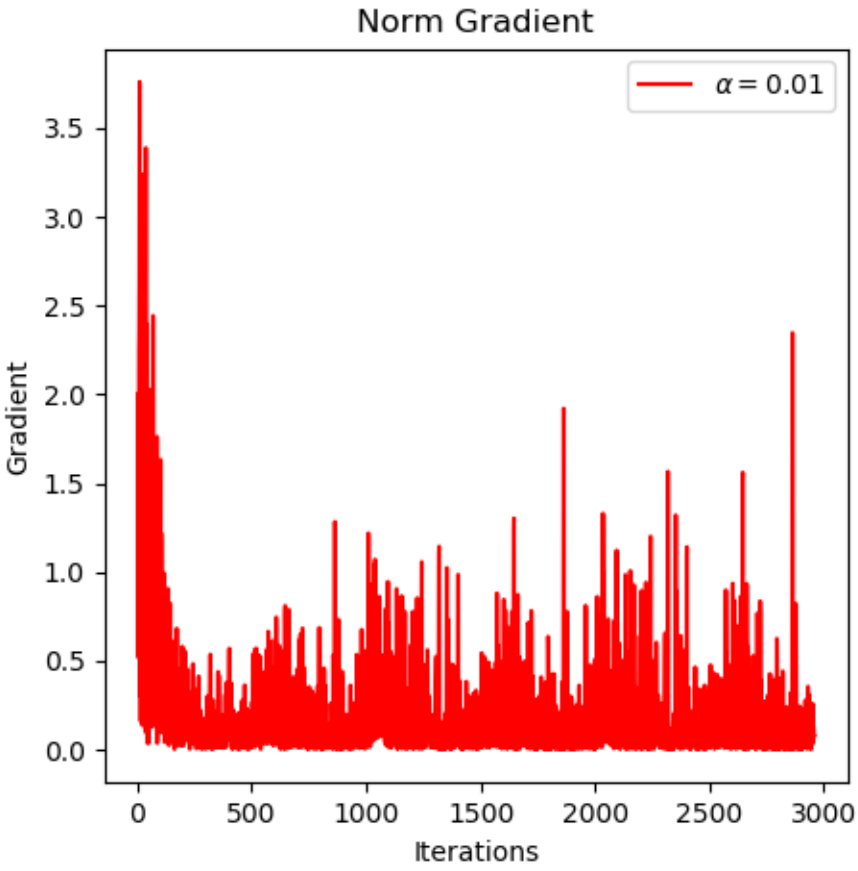

Validation loss
$\alpha = 0.01$
1.2
1.0
0.8
0.6
0.4
Training Error
0
50
100
150
200
250
300
Iterations

c) **Figure of changes of norm of gradient with respect to iteration:**




Norm Gradient
$\alpha = 0.1$
Gradient
Iterations
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
50
100
150
200
250
300
350
400


Norm Gradient
$\alpha = 0.001$
Gradient
Iterations
0
1
2
3
4
5
6
0
5000
10000
15000
20000
25000
30000


Norm Gradient
$\alpha = 0.01$
Gradient
Iterations
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
0
500
1000
1500
2000
2500
3000

d)**Output**:

Learning rate (Alpha): 1

Total Epochs: 100000

total no. of iterations run: 110

Final Weights: [-3.84137918 -1.5351489   1.11438724]

Accuracy: 50.8

==========

Learning rate (Alpha): 0.1

Total Epochs: 100000

Epoch - 1

total no. of iterations run: 391

Final Weights: [-2.64115187 -1.60196429 0.97809784]

Accuracy: 56.39999999999999

==========

Learning rate (Alpha): 0.01

Total Epochs: 100000

Epoch - 30

Cross - Entropy loss at iteration 29001: 0.33060201984910764

Cross - Entropy loss at iteration 29101: 0.4047227167826834

Cross - Entropy loss at iteration 29201: 0.3241272832732063

Cross - Entropy loss at iteration 29301: 0.04323337374540192

Cross - Entropy loss at iteration 29401: 1.2694378569636633

Cross - Entropy loss at iteration 29501: 0.018348222459828648

Cross - Entropy loss at iteration 29601: 0.8398761427923586

Cross - Entropy loss at iteration 29701: 0.23629739378497885

Loss optimized is less than threshold!

total no. of iterations run: 29721

Final Weights: [-0.4642949 -2.16475545 2.25174115]

Accuracy: 97.2

Observation:

When learning rate is 1,the program doesn't seem to be learning well.

For learning rates less then 1,example:0.001 the program is learning well and has an accuracy of 97.2%.

Question 2:

1)Code explanation:

I have included an input_shape as the first parameter because on running I got an error saying that Sequential function needs the input_shape.

=>*Import tensorflow library as tf*->

it is used for using the tensorflow library

=>*mnist=tf.keras.datasets.mnist*->

a)keras is a high-level nn API running on tensorflow

b)we are accessing the mnist dataset of handwritten digits which has a training set of 60,000 examples, and a test set of 10,000 examples

```
=>(x_train, y_train),(x_test, y_test) = mnist.load_data()
  x_train, x_test = x_train / 255.0, x_test / 255.0
```

loading the dataset and separating them into 2 groups train and test also separating images and labels

x_train and x_test contain RGB codes(from 0 to 255) while y_train and y_test contain labels from 0 to 9

=>*x_train, x_test = x_train / 255.0, x_test / 255.0*

We normalise the data by dividing the RGB codes by 255

```
=> model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(28,28)),
    tf.keras.layers.Dense(512, activation=tf.nn.relu),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(10,activation=tf.nn.softmax)])
```

A sequential model is being built which is a linear stack of layers and has many parameters
The first layer ->Flatten, transforms the format of images from a 2-d array to a 1-d array of 28*28 = 784 pixels
A Dense Connected NN layer is built which implements
output = activation(dot(input,kernel) + bias)
The activation function is used for getting the output in this we have used relu other option is sigmoid which are in-built functions
The first parameter of Dense is the units is positive integer which tells the dimensionality of the output space
To avoid overfitting Dropout class used the no. tells the fraction of the input to be dropped Softmax computes
tf.exp(logits) /tf.reduce_sum(tf.exp(logits), axis)

```
=>model.compile(optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy'])
    #the epochs is no. of iterations it will go through the input
    #groups layers into object with training features and inference      features
    model.fit(x_train, y_train, epochs=5)
   #evaluating model based on accuracy
    model.evaluate(x_test, y_test)
```

After the model above is constructed the learning process is configured by using compile
Optimizer explains the training procedure in this we have used Adam we can also use Gradient
Descent also
Loss is the function to minimise during optimisation like mean square error, cross entropy

Metrics used to monitor training

2)

Accuracy	5	10	128	512
Accuracy	85.96	85.96	97.85	97.87

As the number of hidden nodes increases, the accuracy increases as can be seen from the above results which we got on changing the number of hidden layers in the code provided.

tf.keras.layers.Dense(n,activation=tf.nn.relu) where n=5,10,128,512

3)