ASSIGNMENT 1 NNFL (BITS F312)

NAME : Kushaal Tummala

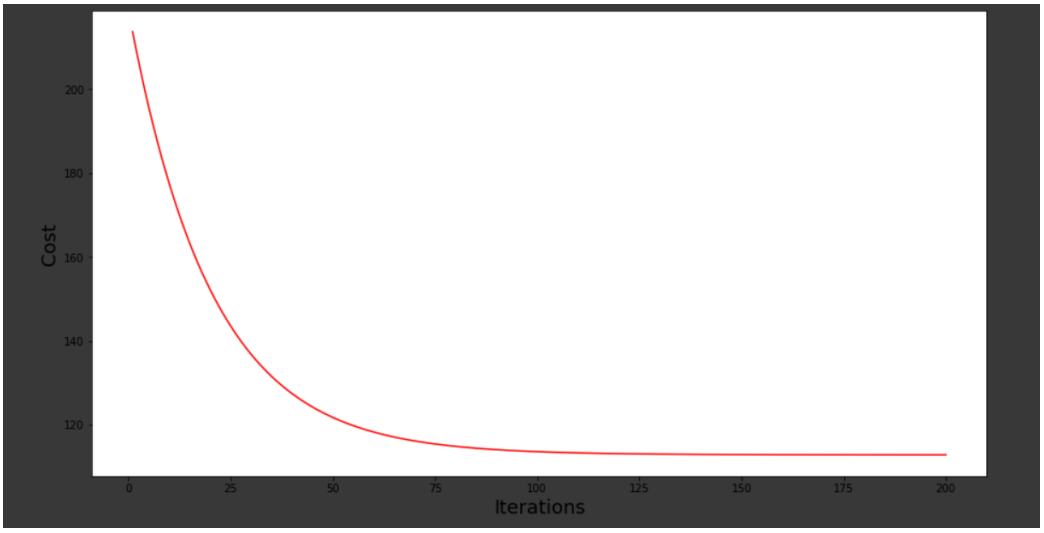
ID No. : 2018AAPS0422H

Github Link (All Questions): https://github.com/SanePai/Neural_Networks-Assignment-1/

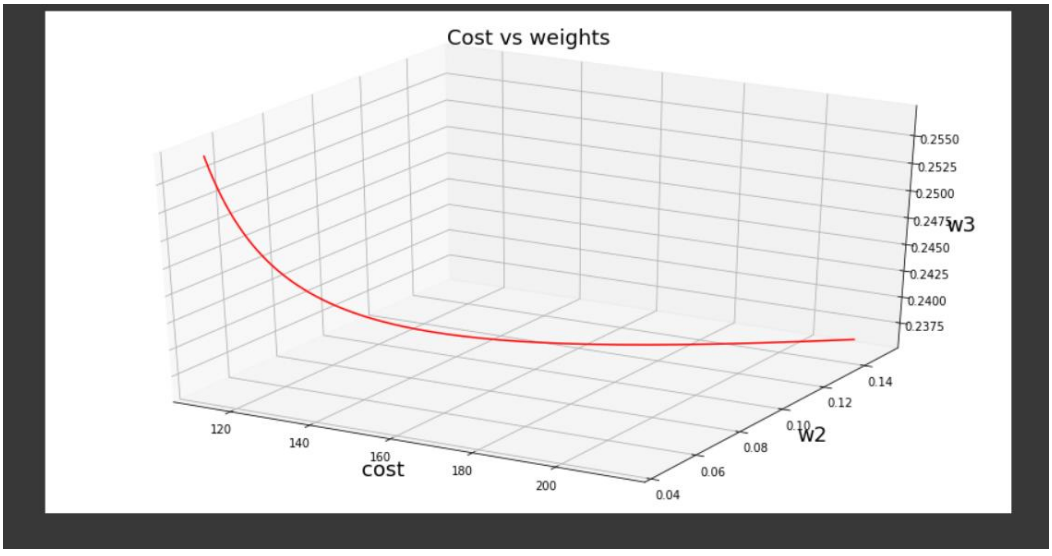
Q1) Linear Regression

(https://colab.research.google.com/github/SanePai/Neural_Networks-Assignment-1/blob/master/Question_1.ipynb)

a. Cost vs Number of Iterations



b. Cost vs Weights (J vs w1, w2 3D graph)



c. MSE for test data

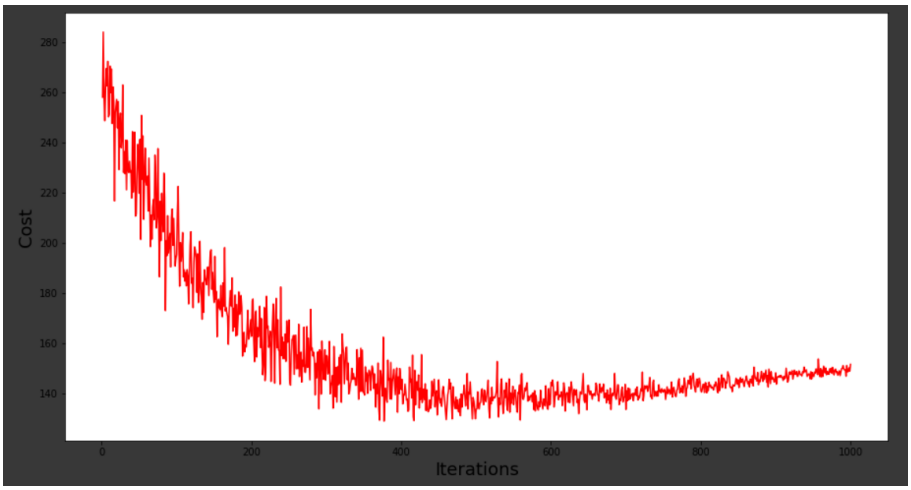
```
Final weights:  [0.00631462 0.04495034 0.25724827]
Testing error(MSE):  [2.65093505]
```

Q2) Linear regression using Mini Batch and Stochastic Gradient Descent

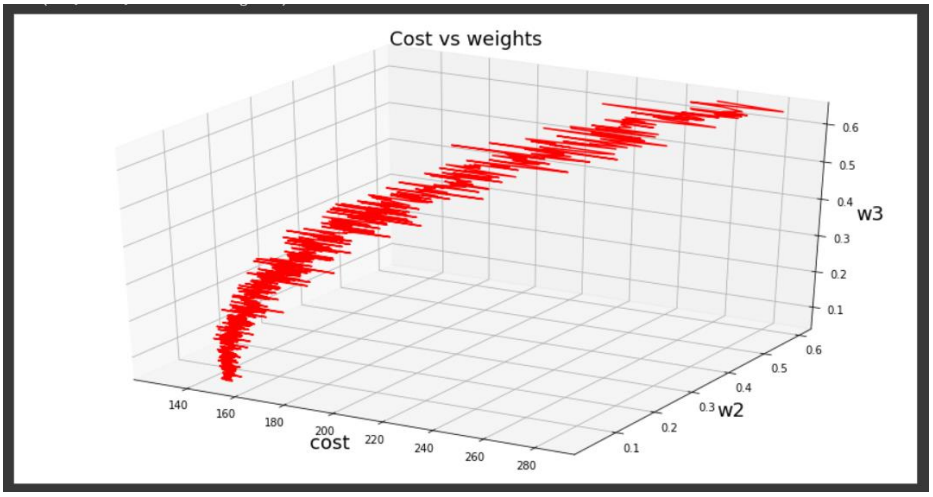
(https://colab.research.google.com/github/SanePai/Neural_Networks-Assignment-1/blob/master/Question_2.ipynb)

MINI BATCH

a. Cost vs Number of Iterations



b. Cost vs Weights (J vs w1, w2 3D graph)

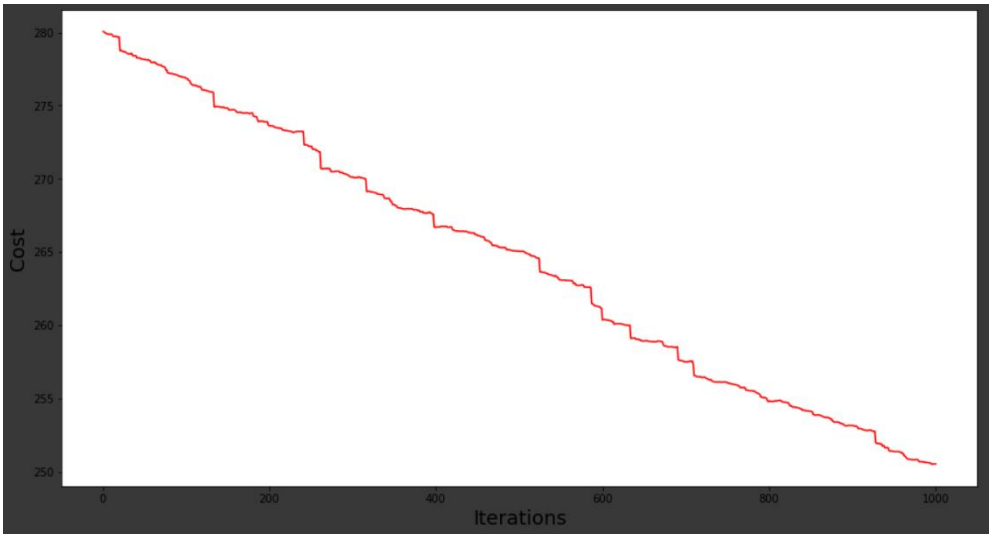


c. MSE for test data

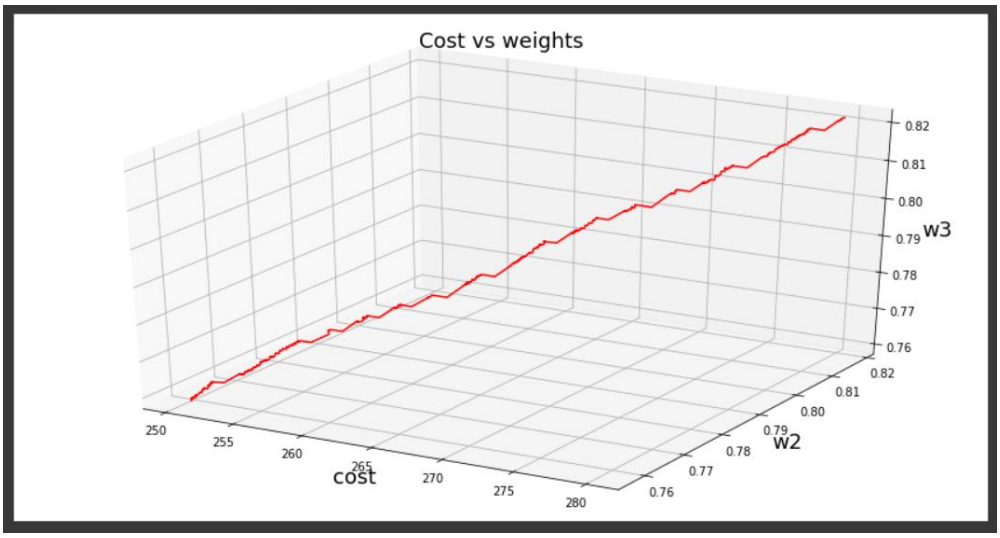
```
MSE(Test data):  [3.26230942]
Final Weights: [-0.46791549  0.0356337   0.05009992]
```

STOCHASTIC

a. Cost vs Number of Iterations



b. Cost vs Weights (J vs w1, w2 3D graph)



c. MSE for test data

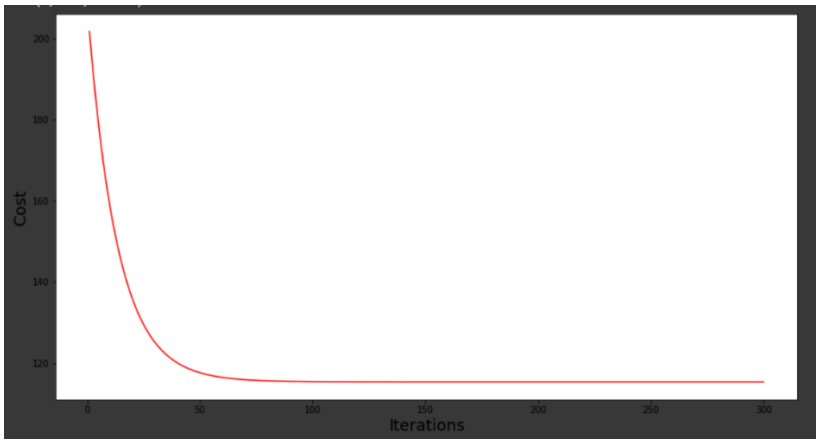
```
MSE(Test data):  [6.20609018]
Final Weights: [0.41544224 0.75774997 0.75843902]
```

Q3) Ridge Regression using batch, mini batch and stochastic gradient descent.

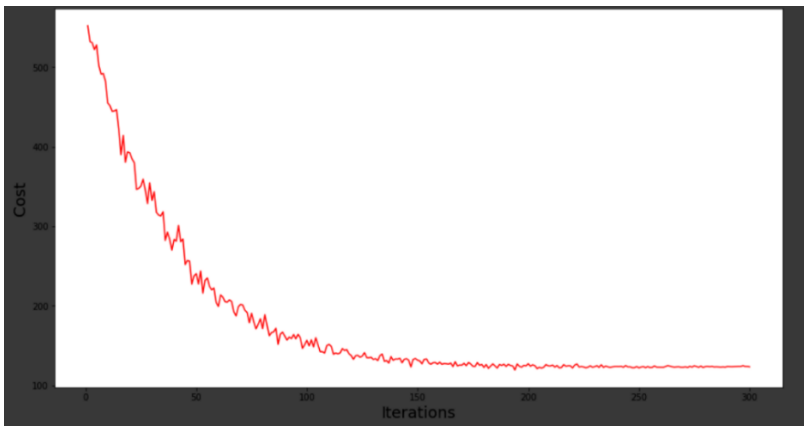
(https://colab.research.google.com/github/SanePai/Neural_Networks-Assignment-1/blob/master/Question_3.ipynb)

a. Cost vs Number of Iterations

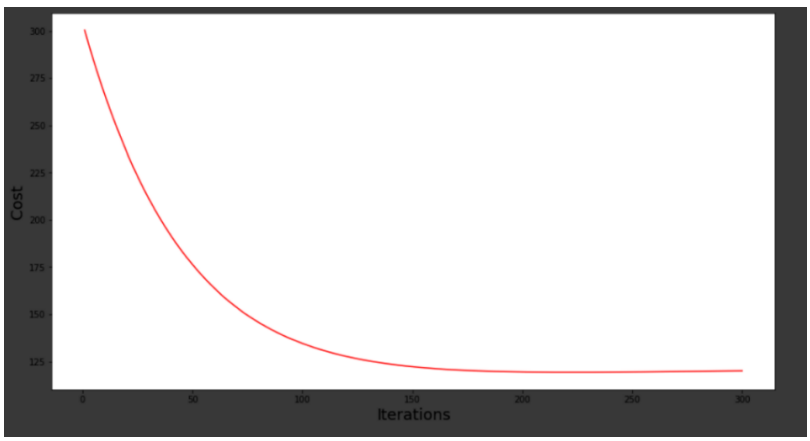
BATCH



MINI BATCH

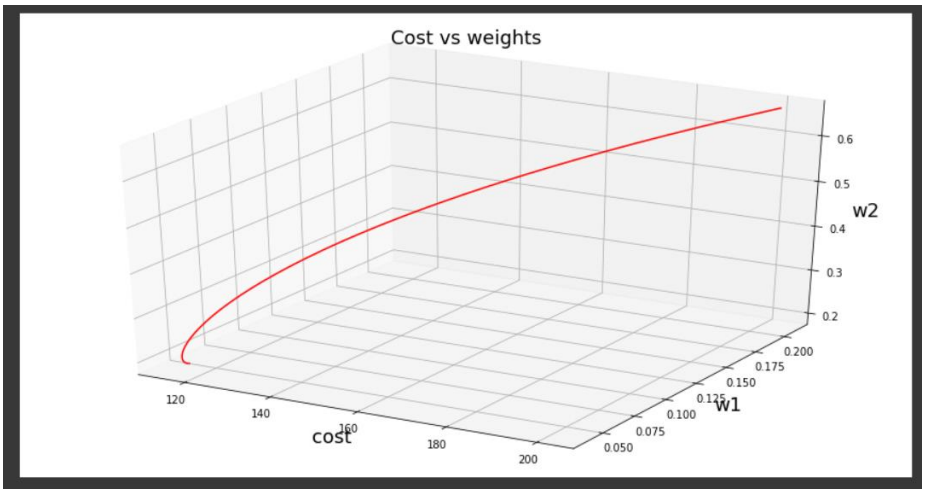


STOCHASTIC

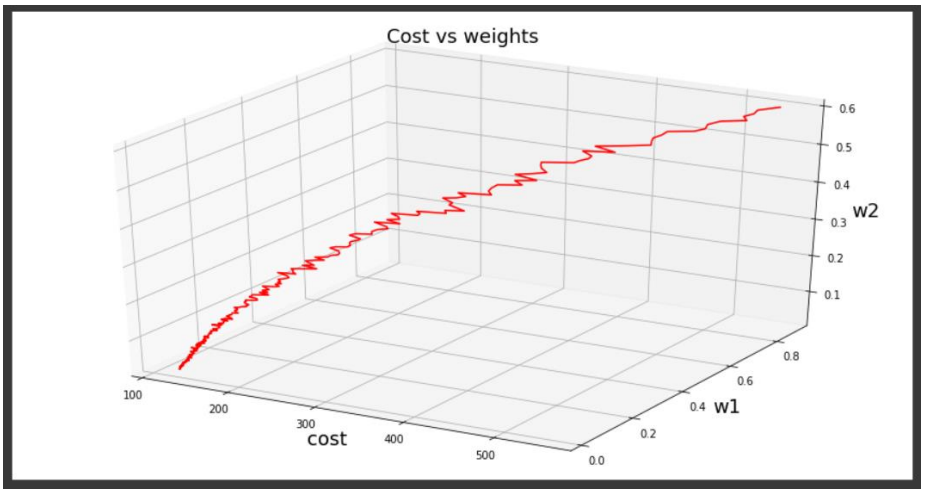


b. Cost vs Weights (J vs w1, w2 3D graph)

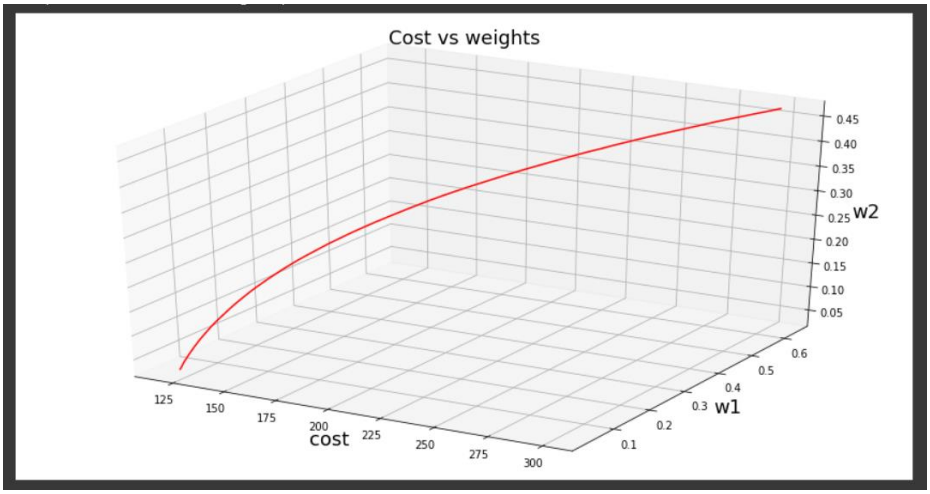
BATCH



MINI BATCH



STOCHASTIC



c. MSE for test data

BATCH

```
MSE(Test data):  [2.63305431]
Final Weights: [1.15655888e-05 4.50791791e-02 1.83340286e-01]
```

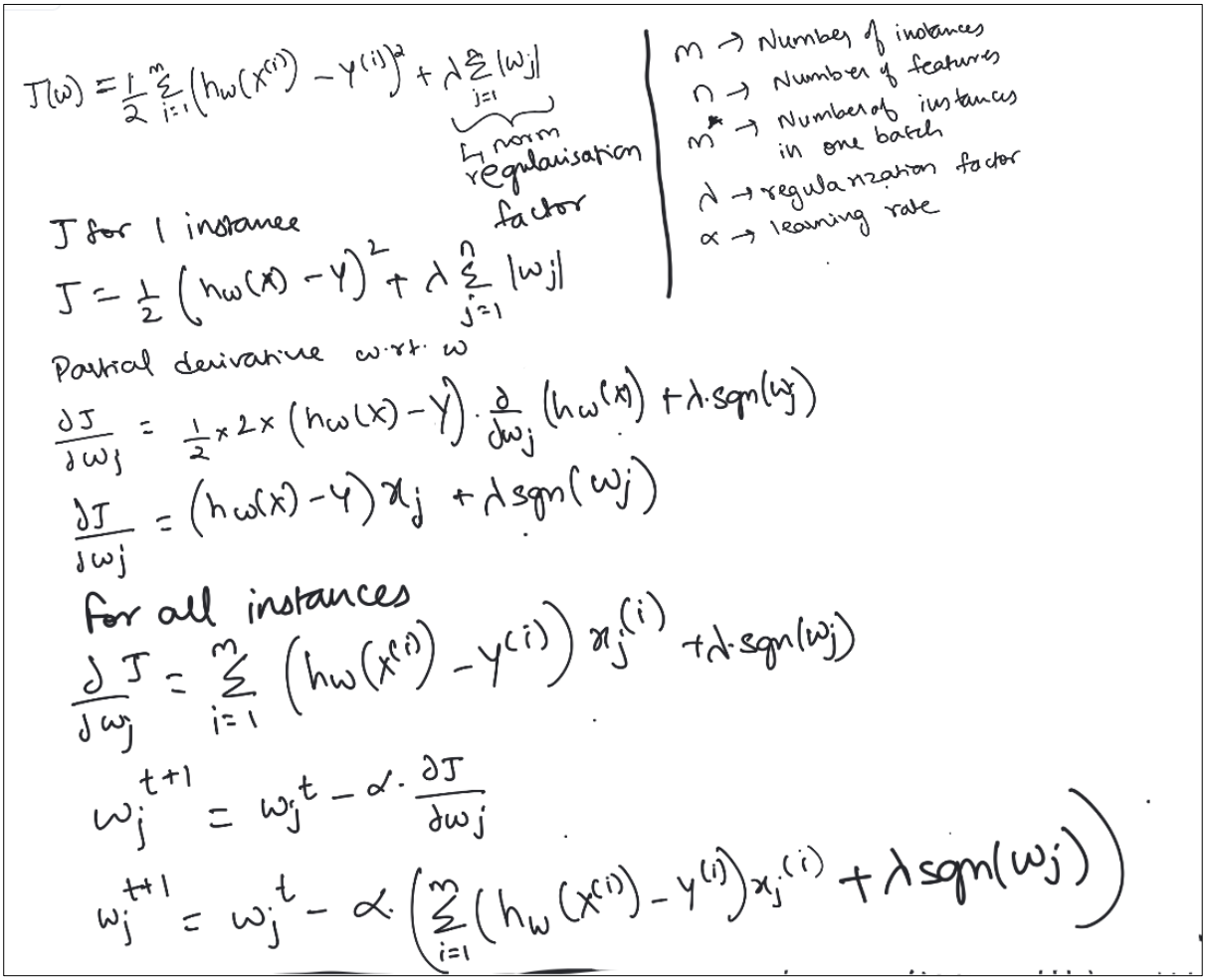
MINI BATCH

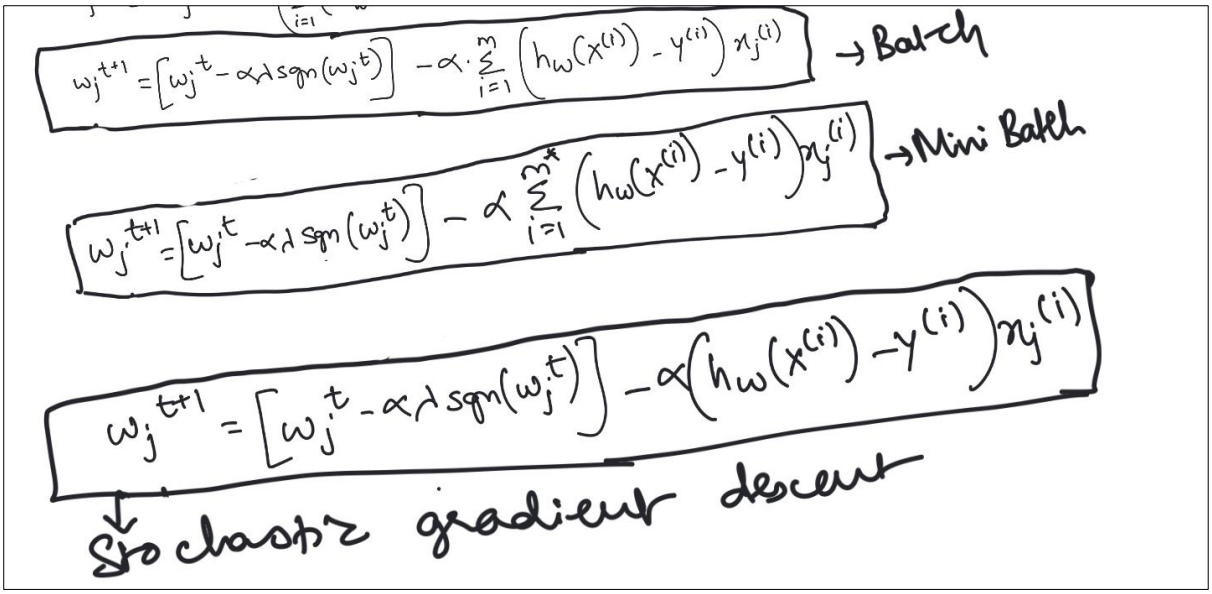
```
MSE(Test data):  [2.70903254]
Final Weights: [-0.07751761  0.02262486  0.01575964]
```

STOCHASTIC

```
MSE(Test data):  [2.68996936]
Final Weights: [0.0264554  0.03040138 0.02428556]
```

Q4.1) LAR Proof





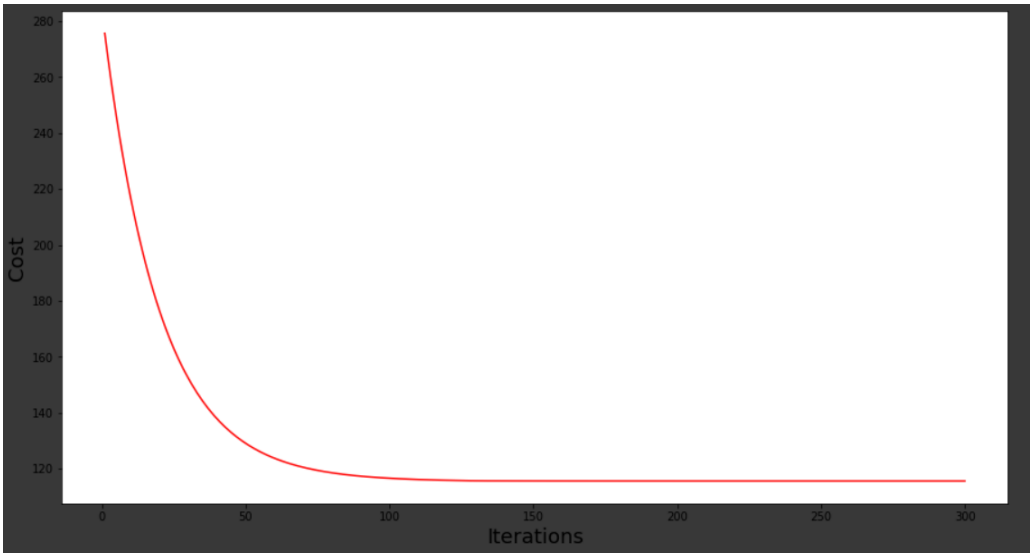
Note: As the weight updates for mini batch and stochastic gradient descents are similar to batch gradient descent with minor modifications (summation over batch size instead of all instances for mini batch and remove the summation term altogether for stochastic gradient descent) they were not derived separately.

Q4.2) Least Angle Regression Implementation (LAR)

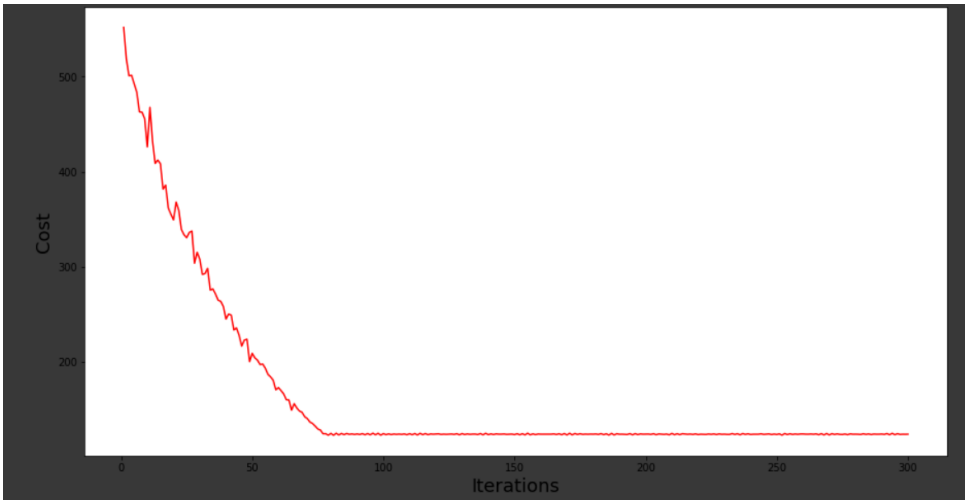
(https://colab.research.google.com/github/SanePai/Neural_Networks-Assignment-1/blob/master/Question_4(2).ipynb)

a. Cost vs Number of Iterations

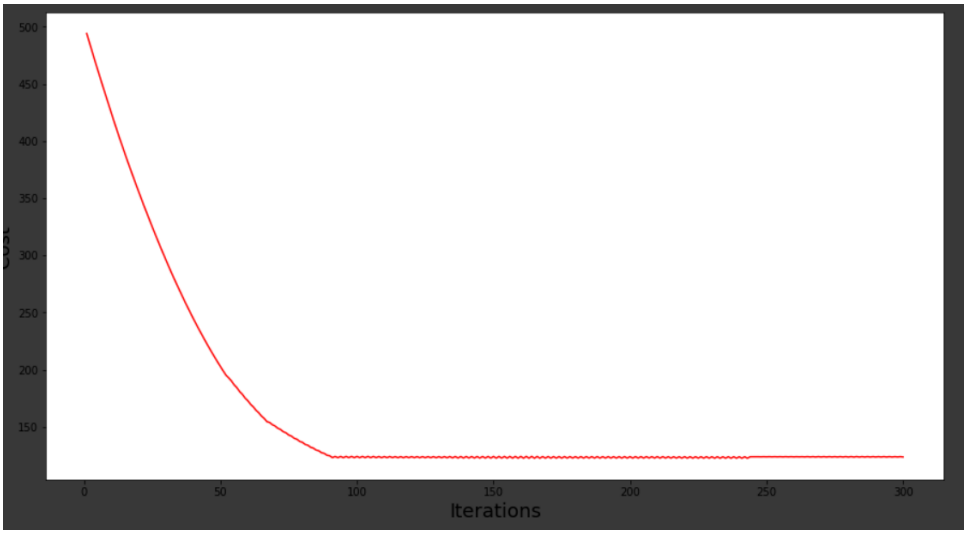
BATCH



MINI BATCH

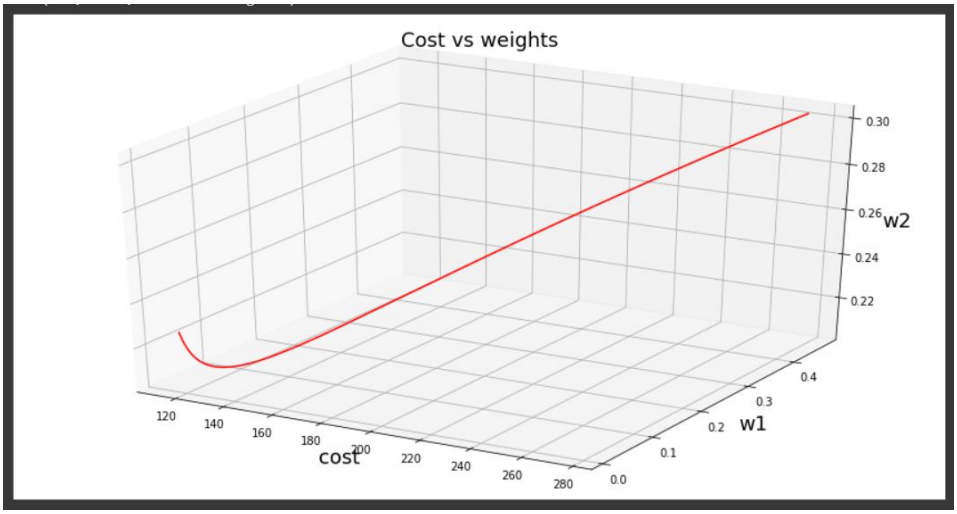


STOCHASTIC

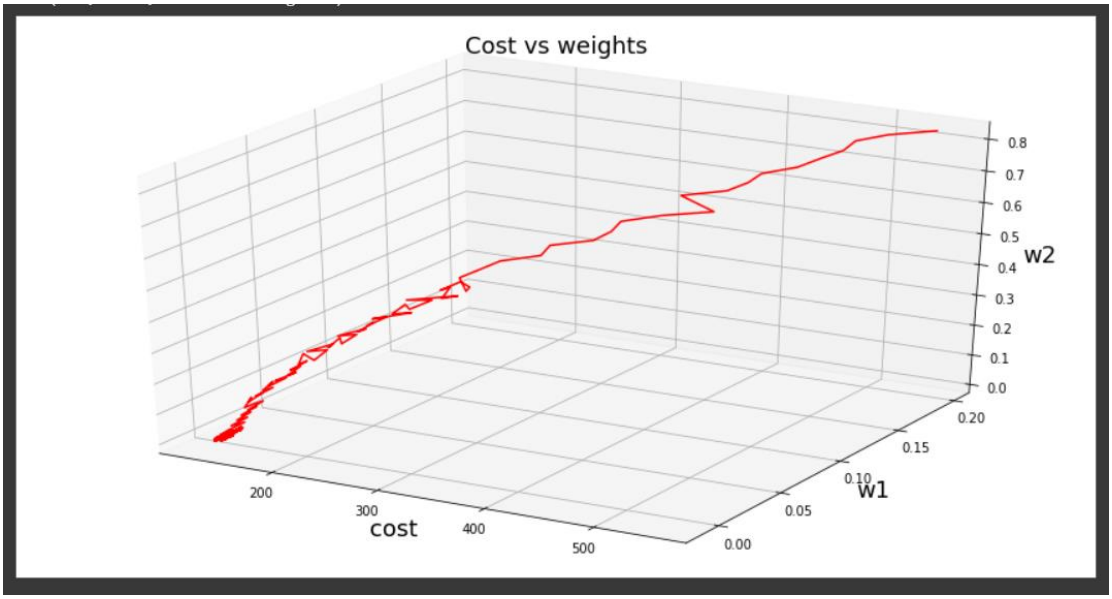


b. Cost vs Weights (J vs w1, w2 3D graph)

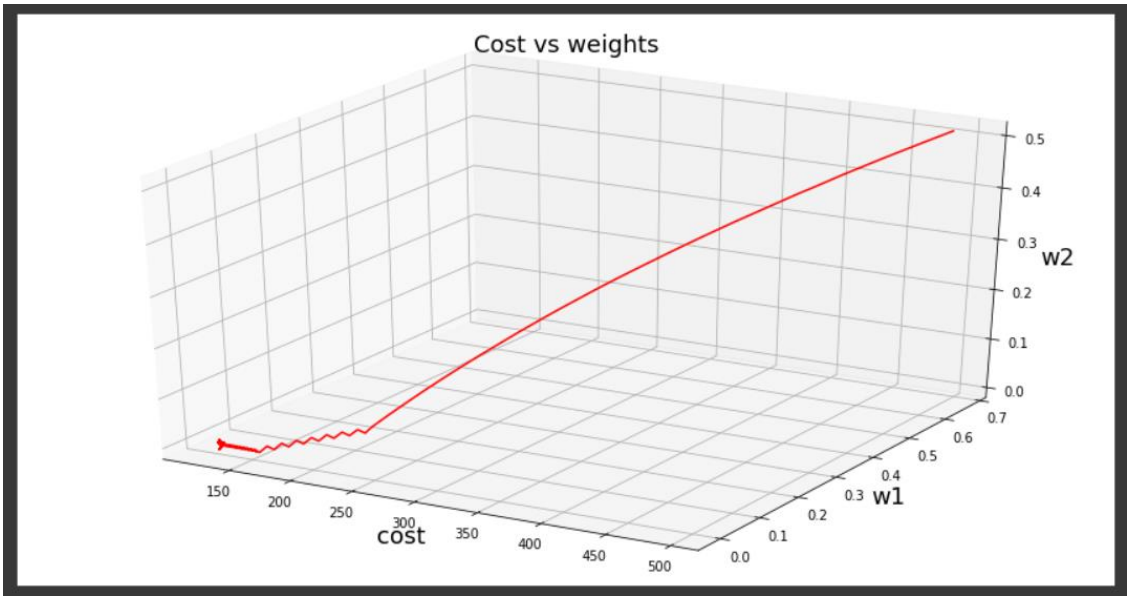
BATCH



MINI BATCH



STOCHASTIC



c. MSE for test data

BATCH

```
MSE(Test data):  [2.62162528]
Final Weights: [0.00050295 0.0117209  0.22589832]
```

MINI BATCH

```
MSE(Test data):  [2.70725955]
Final Weights: [ 0.00159903  0.00962267 -0.00327742]
```

STOCHASTIC

```
MSE(Test data):  [2.71123855]
Final Weights: [-0.00686121 -0.00241285 -0.0091682 ]
```

Q5) Vectorized Linear Regression, Ridge Regression and Least Angle Regression

(https://colab.research.google.com/github/SanePai/Neural_Networks-Assignment-1/blob/master/Question_5.ipynb)

Weights:

The weights vary as the data is shuffled but they are of the same order.

Linear Regression

```
Final Weights: [[1.12440908e+01]
 [6.04224645e-03]
 [4.31499476e-03]]
```

Ridge Regression

```
Final Weights: [[ 6.66414312e+00]
 [-1.83411783e-03]
 [ 7.65626894e-03]]
```

Least Angle Regression

```
Final Weights: [[1.09001239e+01]
 [5.43834701e-03]
 [4.56592818e-03]]
```

MSE:

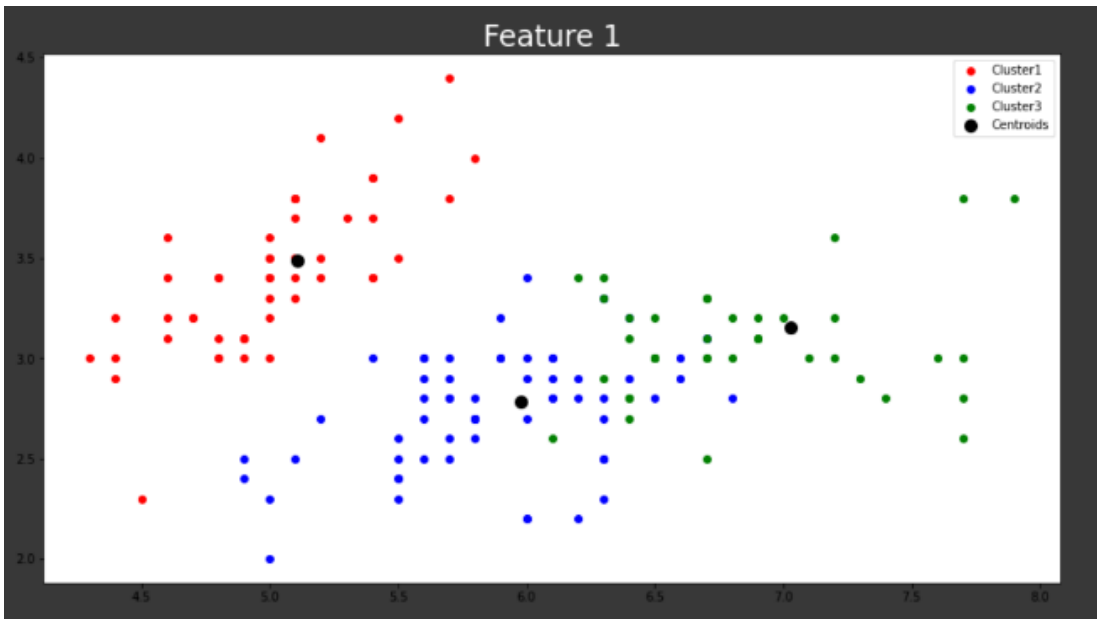
The errors are as expected (Error is lower when regularized)

Linear Regression	Ridge Regression	LAR
17.054135979307006	14.845764080029259	16.860227574901252

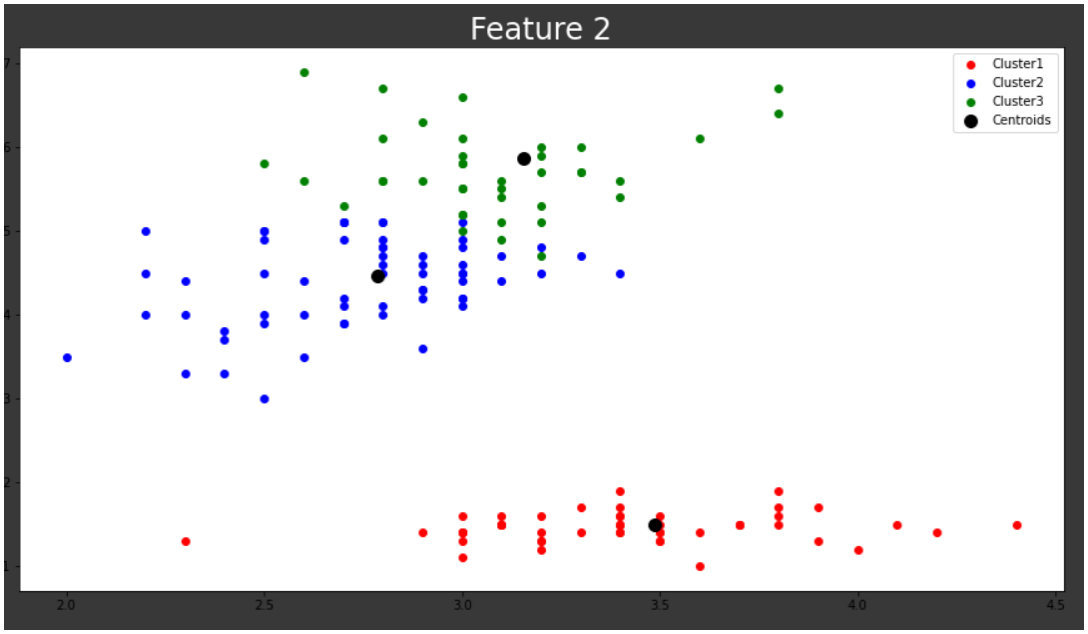
Q6) K means

(https://colab.research.google.com/github/SanePai/Neural_Networks-Assignment-1/blob/master/Question_6.ipynb)

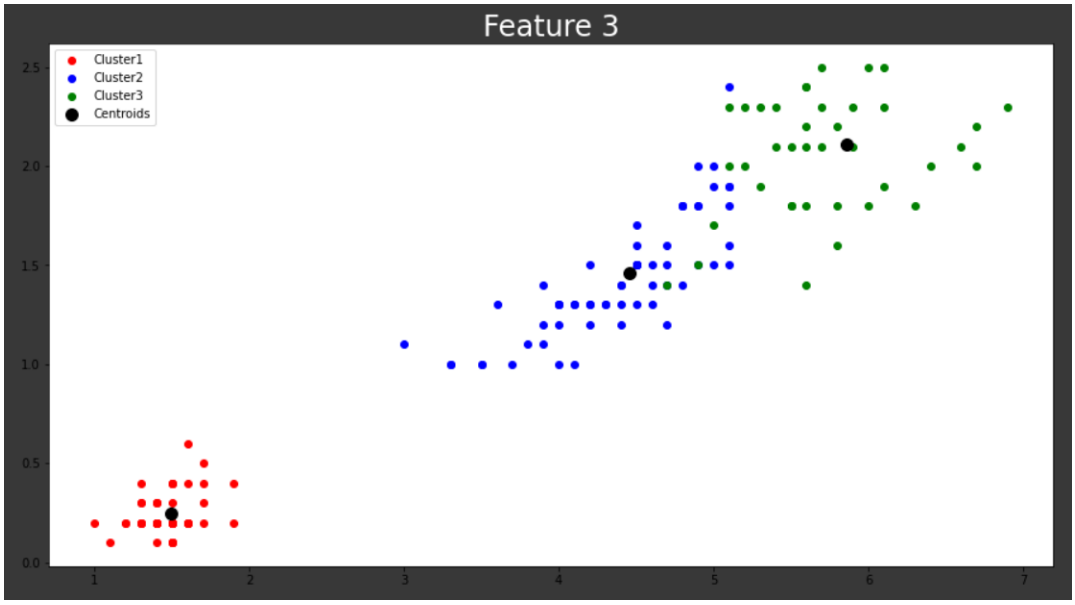
1 vs 2



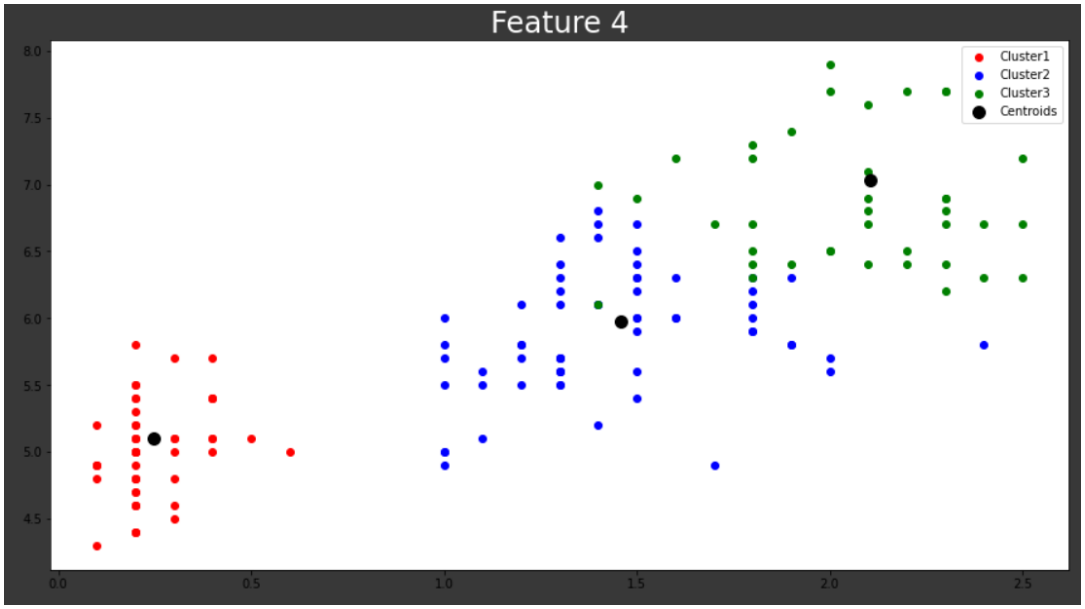
2 vs 3



3 vs 4


Feature 3
Cluster1
Cluster2
Cluster3
Centroids
2.5
2.0
1.5
1.0
0.5
0.0
1
2
3
4
5
6
7

4 vs 1


Feature 4
Cluster1
Cluster2
Cluster3
Centroids
8.0
7.5
7.0
6.5
6.0
5.5
5.0
4.5
0.0
0.5
1.0
1.5
2.0
2.5

Q7) Logistic Regression

(https://colab.research.google.com/github/SanePai/Neural_Networks-Assignment-1/blob/master/Question_7.ipynb)

```
Sensitivity : 1.0
Specificity : 1.0
Accuracy    : 100.0 percent
```

Q8) ONE VS ALL

(https://colab.research.google.com/github/SanePai/Neural_Networks-Assignment-1/blob/master/Question_8_1.ipynb)

```
Overall Accuracy : 0.9666666666666667
Accuracy of class 1 : 1.0
Accuracy of class 2 : 0.8947368421052632
Accuracy of class 3 : 1.0
```

ONE VS ONE

(https://colab.research.google.com/github/SanePai/Neural_Networks-Assignment-1/blob/master/Question_8_2.ipynb)

```
Overall Accuracy : 0.9833333333333333
Accuracy of class 1 : 1.0
Accuracy of class 2 : 0.9411764705882353
Accuracy of class 3 : 1.0
```

Q9) K-fold Cross Validation (K=5)

(https://colab.research.google.com/github/SanePai/Neural_Networks-Assignment-1/blob/master/Question_9.ipynb)

```
Fold 1 Results :
Predicted  1.0  2.0  3.0
Actual
1.0          6    0    0
2.0          0   13    0
3.0          0    0   11
Final Results :
Overall Accuracy : 1.0
Accuracy of class 1 : 1.0
Accuracy of class 2 : 1.0
Accuracy of class 3 : 1.0
***************
```

```
Fold 2 Results :
Predicted  1.0  2.0  3.0
Actual
1.0         13    0    0
2.0          0    8    0
3.0          0    0    9
Final Results :
Overall Accuracy : 1.0
Accuracy of class 1 : 1.0
Accuracy of class 2 : 1.0
Accuracy of class 3 : 1.0
***************
```

```
Fold 3 Results :
Predicted  1.0  2.0  3.0
Actual
1.0         10    0    0
2.0          0   10    2
3.0          0    0    8
Final Results :
Overall Accuracy : 0.9333333333333333
Accuracy of class 1 : 1.0
Accuracy of class 2 : 0.8333333333333334
Accuracy of class 3 : 1.0
***************
```

```
Fold 4 Results :
Predicted  1.0  2.0  3.0
Actual
1.0          9    0    0
2.0          0   12    0
3.0          0    1    8
Final Results :
Overall Accuracy : 0.9666666666666667
Accuracy of class 1 : 1.0
Accuracy of class 2 : 1.0
Accuracy of class 3 : 0.8888888888888888
***************
```

```
Fold 5 Results :
Predicted  1.0  2.0  3.0
Actual
1.0         12    0    0
2.0          0    4    1
3.0          0    0   13
Final Results :
Overall Accuracy : 0.9666666666666667
Accuracy of class 1 : 1.0
Accuracy of class 2 : 0.8
Accuracy of class 3 : 1.0
***************
```

Q10) Classification using LRT

(https://colab.research.google.com/github/SanePai/Neural_Networks-Assignment-1/blob/master/Question_10.ipynb)

```
TP:  17
FP:  0
TN:  23
FN:  0
accuracy:  1.0
sensitivity:  1.0
specificity:  1.0
```

Q11) Classification using MAP

(https://colab.research.google.com/github/SanePai/Neural_Networks-Assignment-1/blob/master/Question_11.ipynb)

Confusion Matrix

Predicted	1	2	3
Actual			
1.0	10	0	0
2.0	0	20	1
3.0	0	0	14

Accuracies

```
Overall Accuracy : 0.9777777777777777
Accuracy of class 1 : 1.0
Accuracy of class 2 : 0.9523809523809523
Accuracy of class 3 : 1.0
```

Q12) Classification using Maximum Likelihood

(https://colab.research.google.com/github/SanePai/Neural_Networks-Assignment-1/blob/master/Question_12.ipynb)

Confusion Matrix

Predicted	1	2	3
Actual			
1.0	13	0	0
2.0	0	14	1
3.0	0	0	17

Accuracies

```
Overall Accuracy : 0.9777777777777777
Accuracy of class 1 : 1.0
Accuracy of class 2 : 0.9333333333333333
Accuracy of class 3 : 1.0
```

Q13) Although we have been taught the concepts and the logic to implement these algorithms, I feel like I have learned a lot more by writing the code itself. Seeing the concepts in action and troubleshooting any errors helped me understand the concepts better.