PERCEPTRON MODEL

We ran all the Perceptron Models with 10000 epochs. Each model was run 10 times with the database shuffled each time. The data was split into 66% testing data and 33% training data.

```
PM1==============>

Accuracy on train set: 0.937007874015748
Precision: 0.8591549295774648
Recall: 0.9104477611940298
Accuracy: 0.9148936170212766

PM2==============>

Accuracy on train set: 0.8267716535433071
Precision: 0.6767676767676768
Recall: 1.0
Accuracy: 0.8297872340425532

PM3==============>

Accuracy on train set: 1.0
Precision: 0.9285714285714286
Recall: 0.9701492537313433
Accuracy: 0.9627659574468085

PM4==============>

Accuracy on train set: 0.937007874015748
Precision: 0.8591549295774648
Recall: 0.9104477611940298
Accuracy: 0.9148936170212766
```

As we can see, PM1 and PM4 give exactly the same results, because changing the order of the features doesn't make any difference in the predicted output. The swapping is compensated by swapping the values of the weights. Change in the ordering of dimensions will just flip the hyperplane accordingly.

There is a difference in PM1 and PM2, because in PM2 we have shuffled the ordering of the tuples, so the order in which we are feeding the data to the model has changed, which means when the model starts processing the same tuple, the weights might not be the same, so the predicted value for that tuple might be different. That's why the two models will be different. It is not guaranteed which one will be better though.

PM3 is much better than the rest of the models, because the data is normalized, i.e, the values have been changed to a common scale.

As we can see, when the data is normalized, the accuracy of the model on the training data is 100%, and on the testing data also it's around 96%, which means that the data is linearly separable.

FISHER'S LINEAR DISCRIMINANT MODEL

FLDM1 => Fisher's Linear Discriminant Analysis
FLDM2 => Same as FLDM1 but the order of features are shuffled

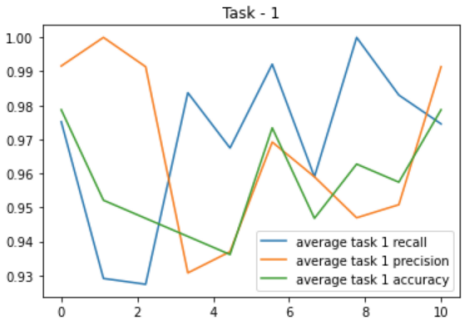
We ran both FLDM1 and FLDM2 10 times, shuffling the order of tuples each time and taking the average of the scores shown below.

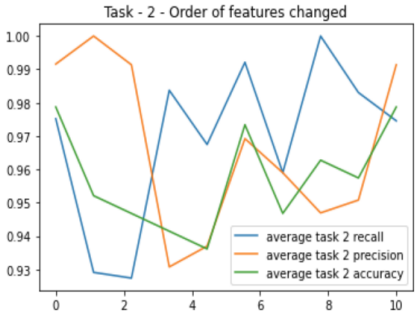
FLDM1:

Average recall : 0.96917488
Average precision: 0.96681689
Average accuracy: 0.95744681

FLDM2:

Average recall : 0.96917488
Average precision: 0.96681689
Average accuracy: 0.95744681





Looking at the scores we were able to see that both FLDM1 and FLDM2 are equivalent Models.

It is what we would expect as the only difference is that the order of features are changed , this means that The order of coefficients may be different in the best plane but they will still be the same coefficients.

LOGISTIC REGRESSION MODEL

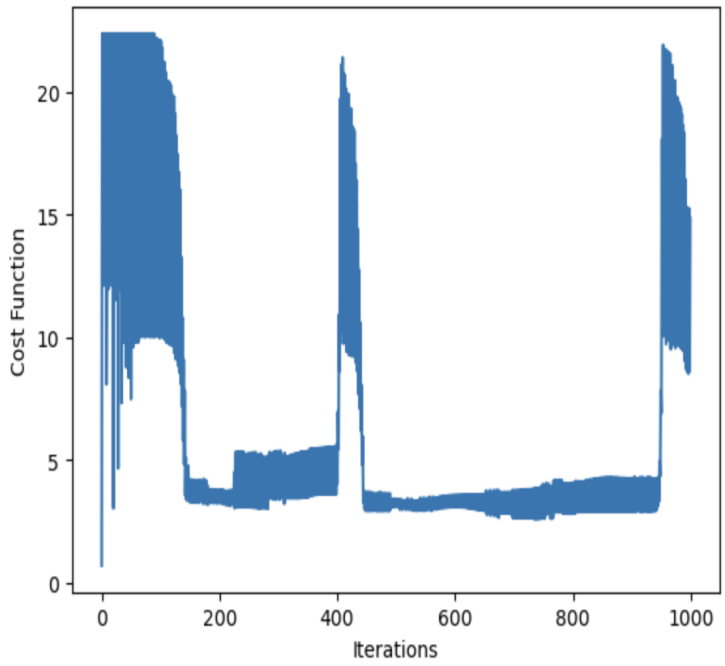
As we have total of 90 graphs it will be cumbersome to go through all the graphs, so we have shown graph for 1 threshold value 0.3 and considered the learning rate as 0.01. We have considered the value of epochs as 1000.

a)threshold=0.3

i)Batch

Unnormalized

```
0
0.3
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:16: RuntimeWarning: overflow encountered in exp
```

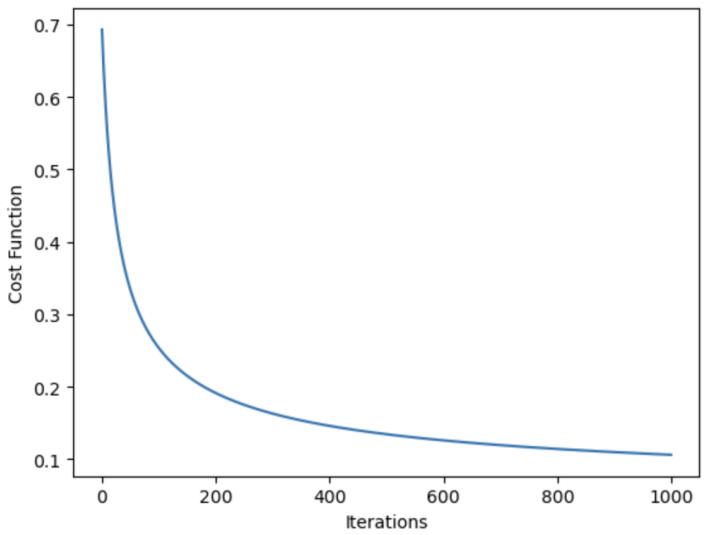



```
Accuracy,Precision,Recall =  0.7180851063829787 1.0 0.32051282051282054

Mean accuracy for 10 iterations =  0.8936170212765957
Mean precision for 10 iterations =  0.9033898884676793
Mean recall for 10 iterations =  0.8182692777335332
```

Nomalized

```
0.3
```

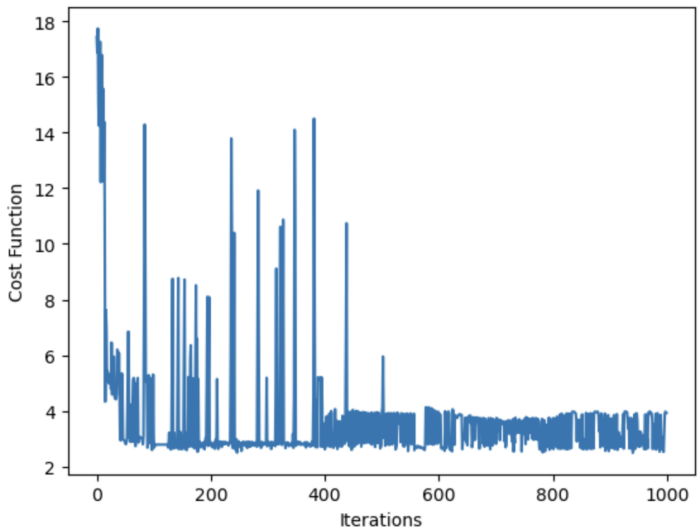



```
Accuracy,Precision,Recall =  0.973404255319149 0.9493670886075949 0.9868421052631579

Mean accuracy for 10 iterations =  0.9542553191489361
Mean precision for 10 iterations =  0.9100238517440896
Mean recall for 10 iterations =  0.9777321296319302
```

ii) Mini batch
Unnormalized

```
1
0.3
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:16: RuntimeWarning: overflow encountered in exp
```

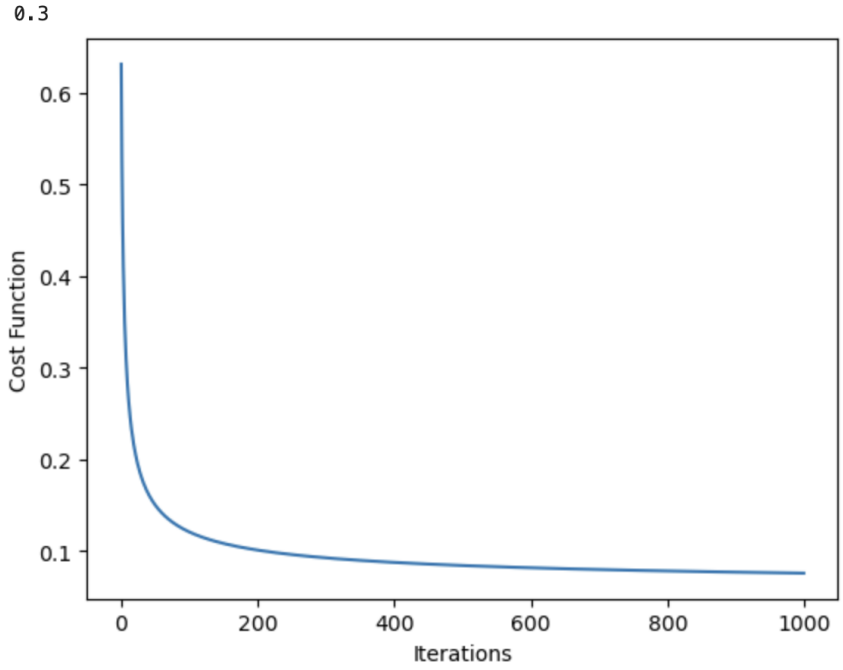



```
Accuracy,Precision,Recall =  0.9148936170212766 0.9104477611940298 0.8591549295774648

Mean accuracy for 10 iterations =  0.8468085106382979
Mean precision for 10 iterations =  0.8509866693833057
Mean recall for 10 iterations =  0.7782804056293541
```

Normalized

```
0.3
```



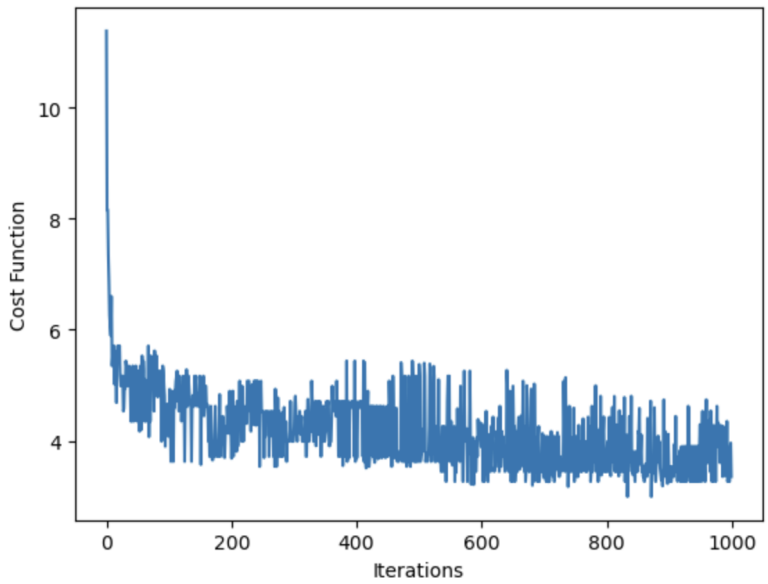

```
Accuracy,Precision,Recall =  0.9893617021276596 0.972972972972973 1.0

Mean accuracy for 10 iterations =  0.9691489361702127
Mean precision for 10 iterations =  0.9425218412090641
Mean recall for 10 iterations =  0.9782235017593527
```

iii)Stochastic

Unnormalized

```
2
0.3
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:16: RuntimeWarning: overflow encountered in exp
```

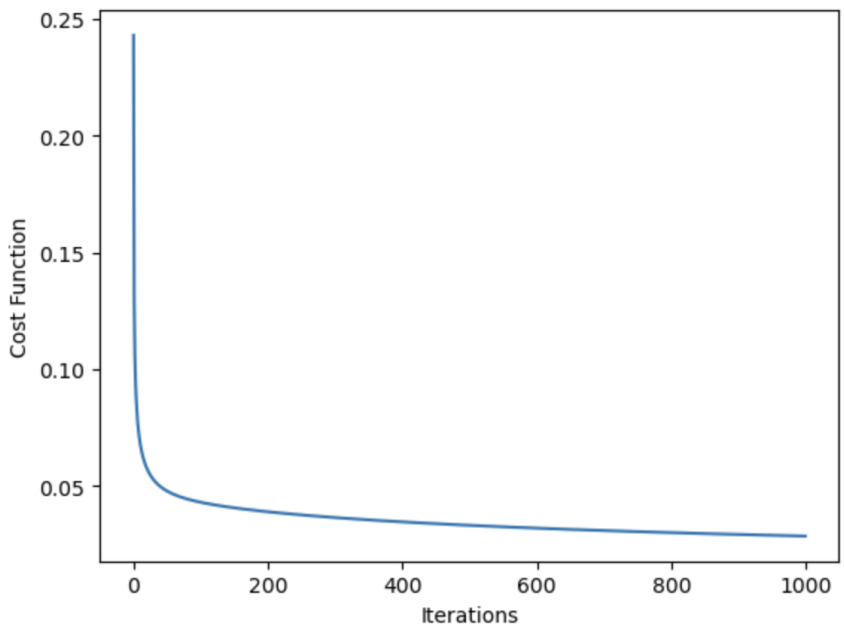



```
Accuracy,Precision,Recall =  0.9148936170212766 0.881578947368421 0.9054054054054054

Mean accuracy for 10 iterations =  0.8989361702127658
Mean precision for 10 iterations =  0.9303052627435381
Mean recall for 10 iterations =  0.805917026595219
```

Normalized

```
0.3
```




```
Accuracy,Precision,Recall =  0.973404255319149 0.987012987012987 0.95

Mean accuracy for 10 iterations =  0.9590425531914895
Mean precision for 10 iterations =  0.935867020512943
Mean recall for 10 iterations =  0.9626609265544289
```

PART D :

Model	Precision	Recall	Accuracy
PM1	0.86	0.91	0.91
PM2	0.67	1	0.82
PM3	0.93	0.97	0.96
PM4	0.86	0.91	0.92
FLDM1	0.97	0.97	0.96
FLDM2	0.97	0.97	0.96
LR1	0.87	0.85	0.88
LR2	0.99	0.9	0.96

As we can see, FLDM works best for the given dataset. Both the FLDMs are the same, so we can say that either FLDM1 or FLDM2 is giving the best results for the particular dataset.