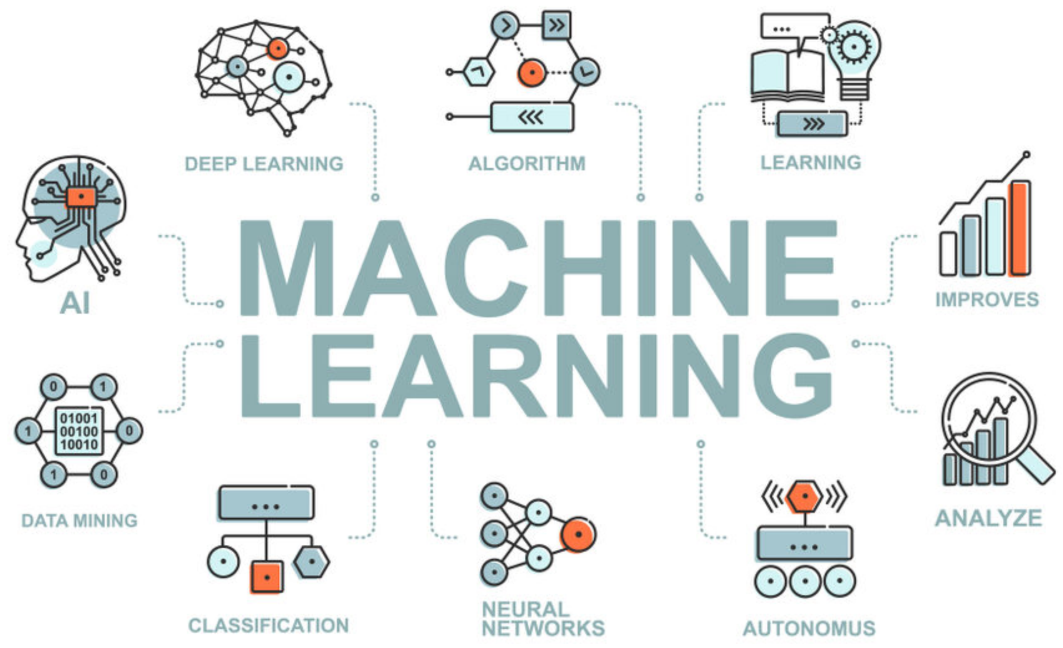
BITS F464

Assignment 1

MACHINE LEARNING



Team Members

- Divyateja Pasupuleti 2021A7PS0075H
- Adarsh Das 2021A7PS1511H
- Kumarasamy 2021A7PS0096H

Introduction

Implemented Perceptron’s Algorithm, Fisher's Linear Discriminant Analysis, and Logistic Regression from scratch without using ML libraries such as *scikit-learn* or *seaborn*. Made models which predicted if the person's cancer was benign or malignant based on the dataset given.

We have two types of datasets, the normal dataset with just those tuples dropped, which have nan values and the feature-engineered dataset. In the feature-engineered dataset, we were asked to perform two steps; one is to replace all nan(s) with means of the respective column and also perform normalization.

We use accuracy, precision, recall, and f1 scores as metrics

1. Accuracy: Represents the number of correctly classified data instances over the total number of instances
2. Precision talks about how many positives are correct
3. Recall talks about how many true positives exist and how many should be there
4. F1 Score considers both precision and recall and takes the harmonic mean of the same

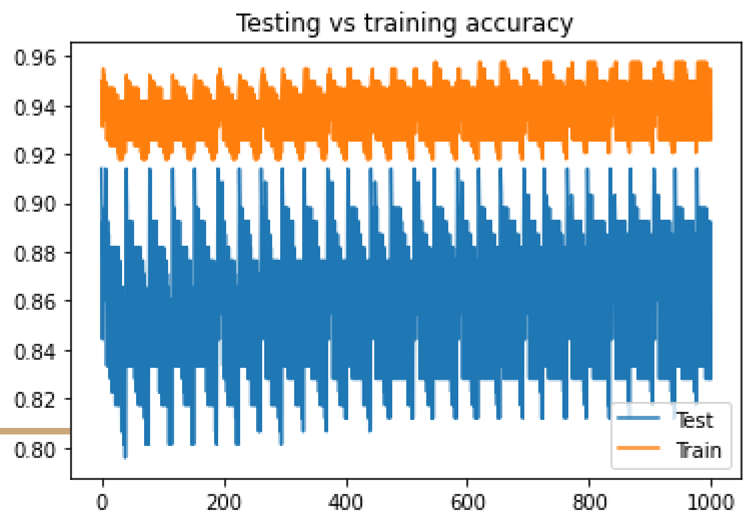
Perceptron

Perceptron is a model which works best on data that is linearly separable. In this project, we analyzed four variations of the perceptron algorithm.

1. PM1: Rows with null values were dropped
2. PM2: Rows with null values are dropped and shuffled
3. PM3: All columns are normalized and null values are replaced with mean
4. PM4: Columns are shuffled, and null values dropped

PM1

As a method to save time, the function was seeded with a set of initial weights. In short, we calculate the weights that would give the maximum accuracy and then feed it as the initial weights for the next run. As the dataset was the same, this helped us to divide the training into



smaller time frames instead of one big one-hour training.

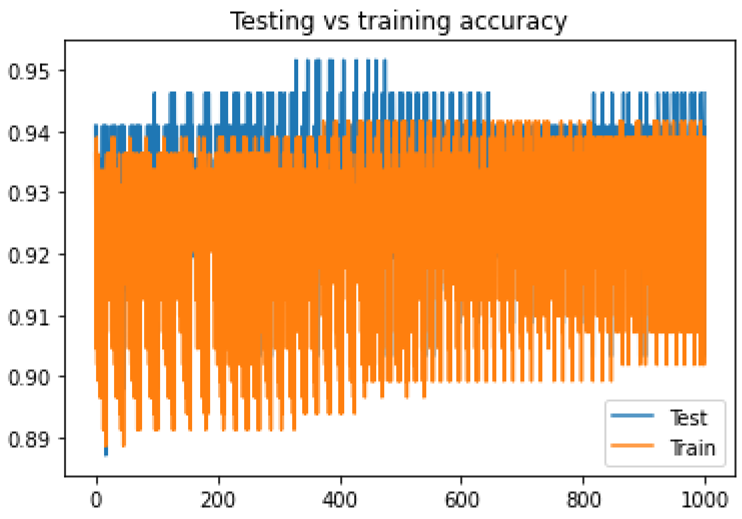
The following data contains the statistics for the weights, which provided us with the maximum accuracy.

Epochs	1000
Learning Rate	0.3
Accuracy (Avg)	90.32 %
Precision (Avg)	93.10 %
Recall (Avg)	81.81 %
F1 (Avg)	87.64 %

Due to the periodic nature of the accuracy, we deduced that the data is linearly inseparable, and it would be impossible to reach the weights which provide complete separation.

PM2

This model also had the same dataset with a null value dropped, just like PM1, with the difference being the rows were shuffled with initial weights initialized.



Here also we took the weights with the maximum accuracy, and due to no evidence of the data being linearly separable, we simply took the max.

The reason is the seemingly random test and train accuracy to the epochs.

Epochs	1000
Learning Rate	0.3

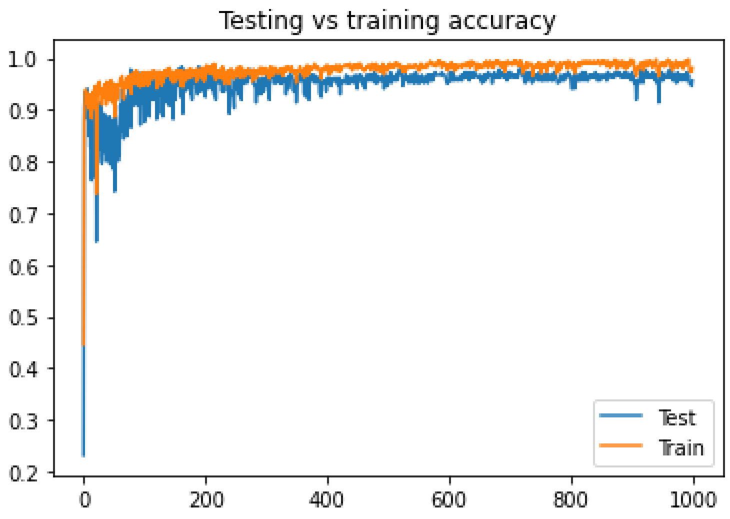
Accuracy (Avg)	88.70 %
Precision (Avg)	90.90 %
Recall (Avg)	91.66 %
F1 (Avg)	91.28 %

PM3

In this model, all columns were normalized and empty cells filled with the mean of that column.

On plotting the accuracy of testing and training data, we can see a clear indication of reaching a maximum.

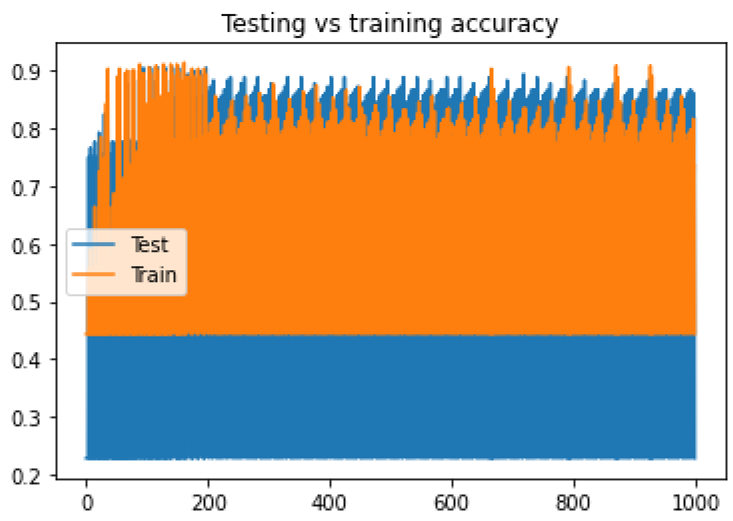
On the personal testing, we reached training accuracies of 100%, making this the only model to give 100% accuracy. This proves that the normalized data is linearly separable.



Epochs	1000
Learning Rate	0.3
Accuracy (Avg)	97.31 %
Precision (Avg)	99.99 %
Recall (Avg)	96.50 %
F1 (Avg)	98.22 %

PM4

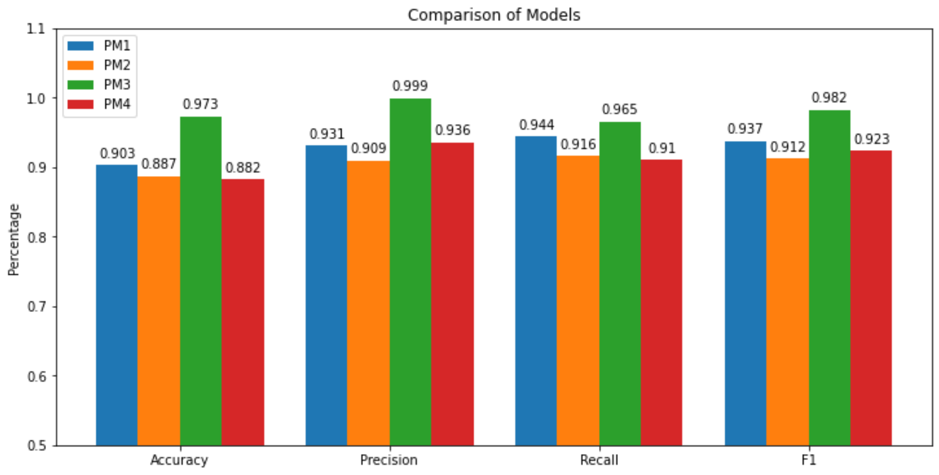
PM4 also takes the simple dataset with dropped null values, furthermore while training the columns are shuffled.



Epochs	1000
Learning Rate	0.3
Accuracy (Avg)	88.29 %
Precision (Avg)	93.61 %
Recall (Avg)	91.03 %
F1 (Avg)	92.30 %

Analysis

Finally, we compare all the models by plotting their accuracy, precision, recall, and F1 as a bar graph.



We can observe PM3 outperforms all other models in all metrics. Hence PM3 is the best model among Perceptron models.

Fisher's Linear Discriminant Analysis

In FLDM, we have four types of models

- FLDM1: Using un-normalized data and non-shuffled columns
- FLDM1: Using normalized data and non-shuffled columns
- FLDM2: Using un-normalized data and shuffled columns
- FLDM2: Using normalized data and shuffled columns

We are following a class-based coding practice, where each model runs on the dataset. In this preliminary analysis, we talk about running all four abovementioned models only once.

We split the dataset into two parts. Our training contained 209 benign and 168 malignant samples.

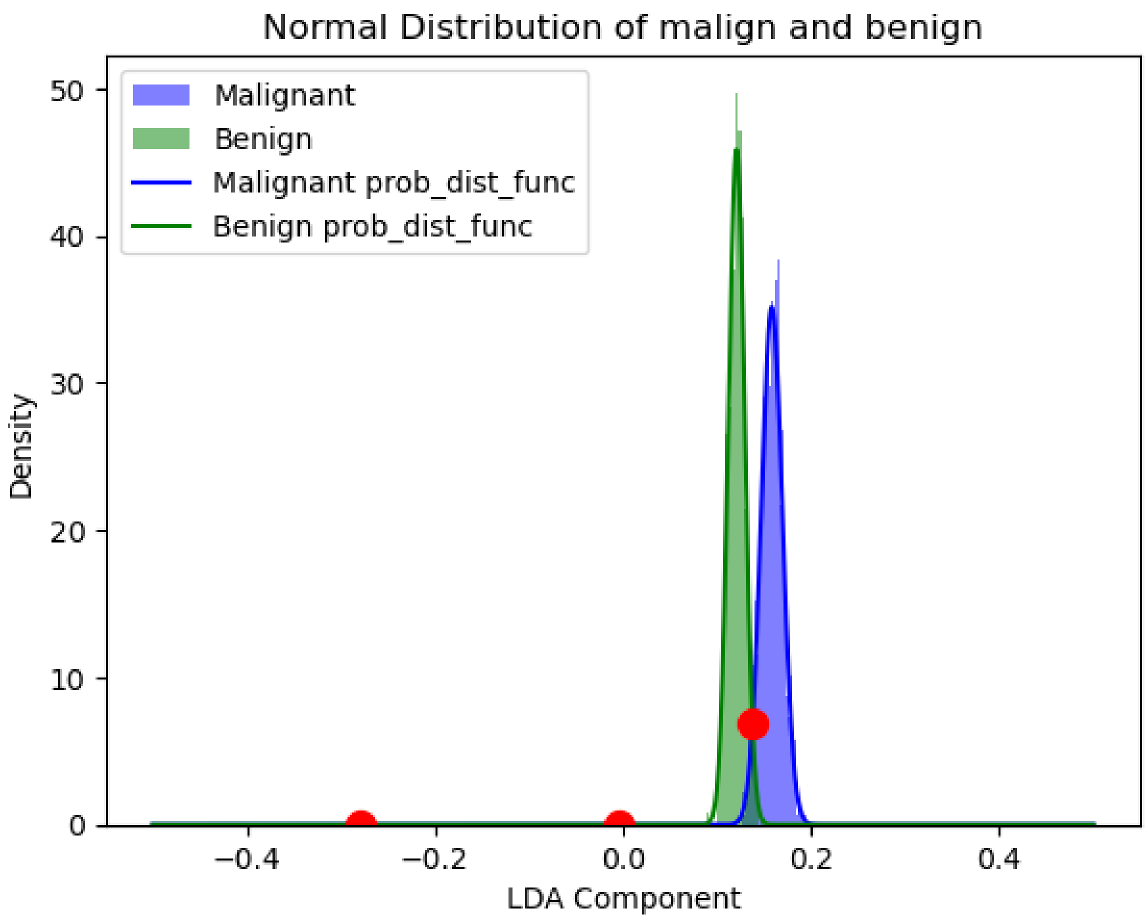
FLDM1

After training it on the data, we got a training accuracy of **97.61273209549072%** and a testing accuracy of **97.31182795698925%.**

The following is the data for the same:

Accuracy	97.31182795698925%
Precision	99.28571428571429%

F1	98.23321554770317%
Recall	97.2027972027972%



Here, the red point represents the boundary between both the normal distributions made from the mean and standard deviation of the given training data.

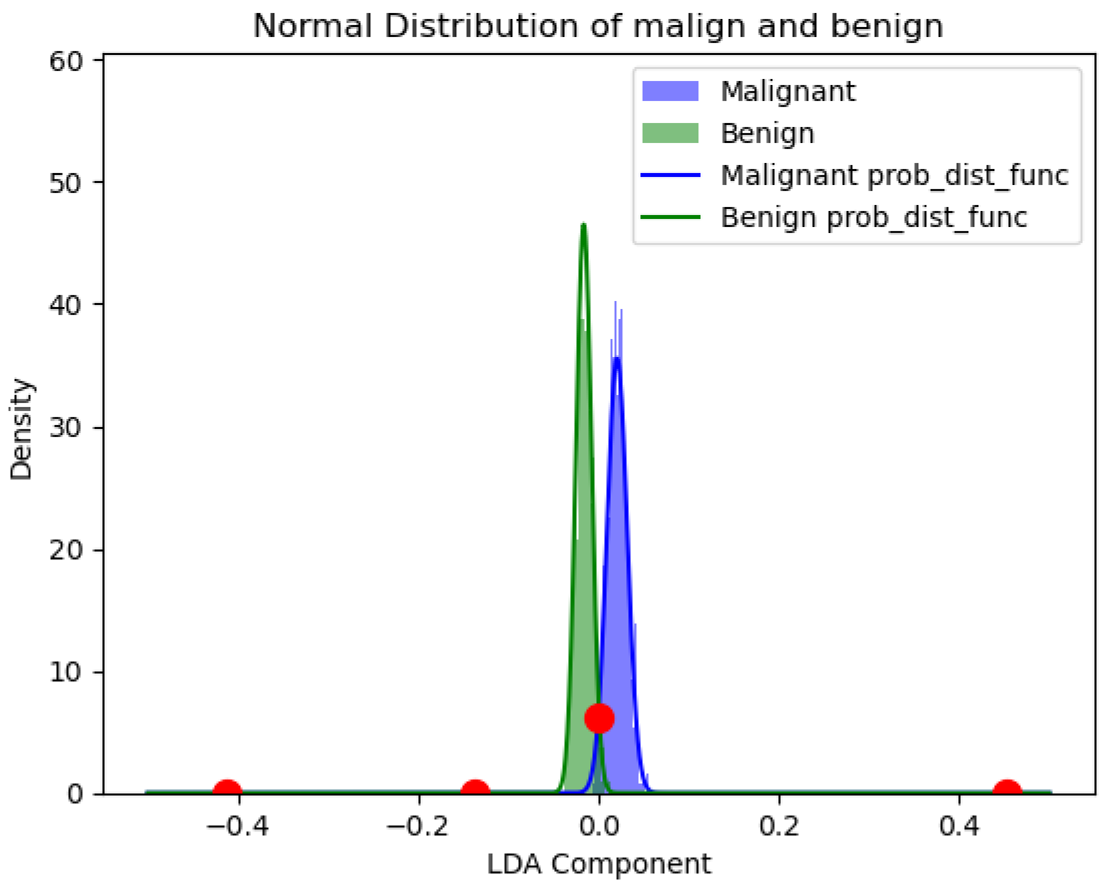
FLDM1 with normalized dataset

After training it on the data, we got a training accuracy of **97.63779527559056%** and a testing accuracy of **96.80851063829787%.**

The following is the data for the same:

Accuracy	96.80851063829787%
Precision	99.29078014184397%

F1	97.90209790209791%
Recall	96.55172413793104%



Here, the red point represents the boundary between both the normal distributions made from the mean and standard deviation of the given training data.

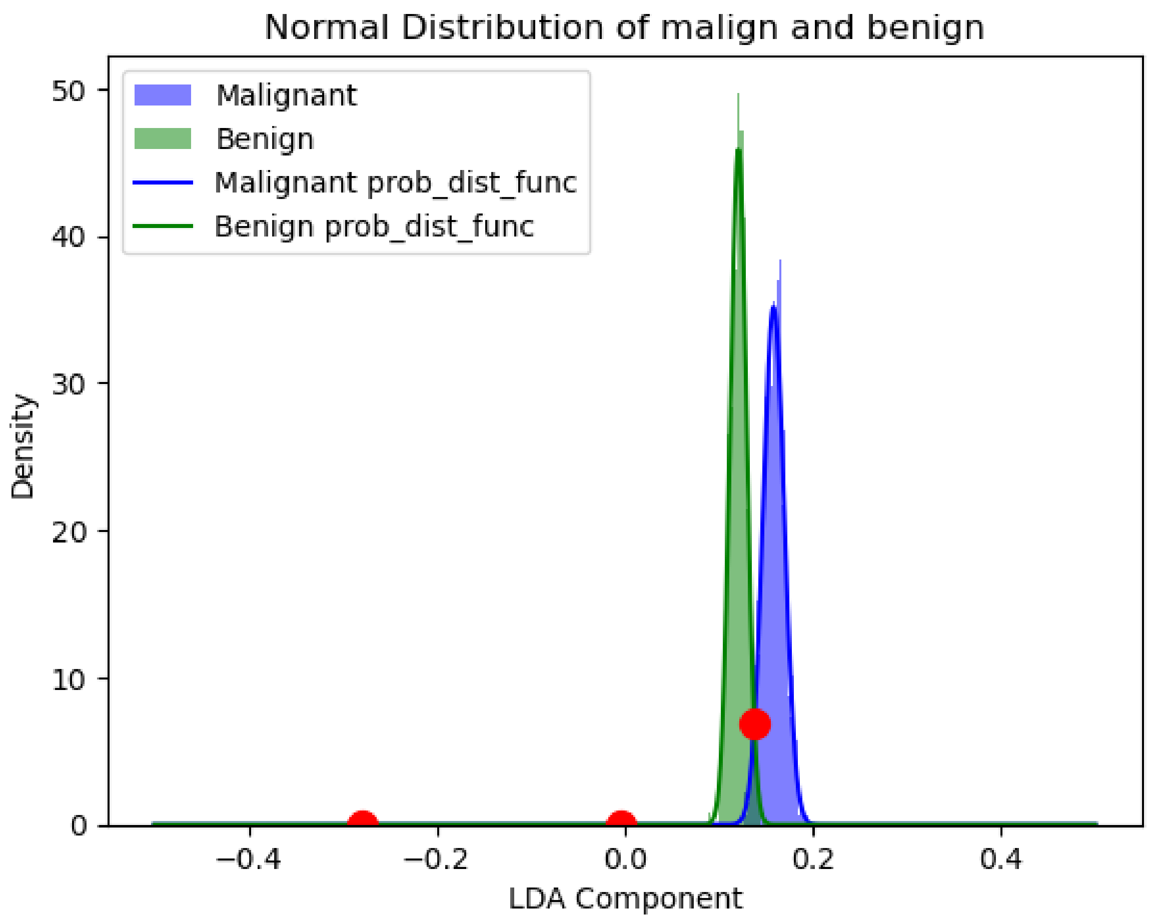
FLDM2

After training it on the data, we got a training accuracy of **97.61273209549072%** and a testing accuracy of **97.31182795698925%.**

The following is the data for the same:

Accuracy	97.31182795698925%
Precision	99.28571428571429%
F1	98.23321554770317%

Recall	97.2027972027972%



Here, the red point represents the boundary between both the normal distributions made from the mean and standard deviation of the given training data.

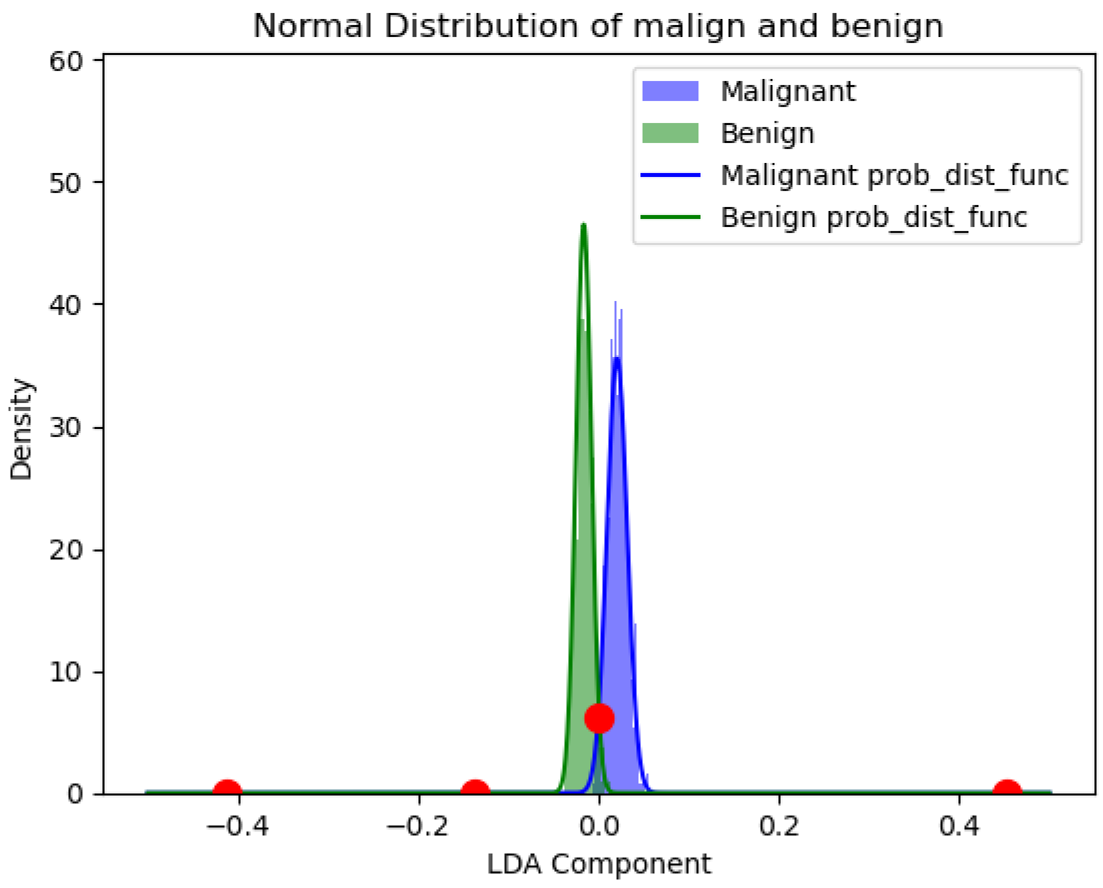
FLDM2 with normalized dataset

After training it on the data, we got a training accuracy of **97.63779527559056%** and a testing accuracy of **96.80851063829787%.**

The following is the data for the same:

Accuracy	96.80851063829787%
Precision	99.29078014184397%
F1	97.90209790209791%

Recall	96.55172413793104%



Here, the red point represents the boundary between both the normal distributions made from the mean and standard deviation of the given training data.

Analysis

- We observe that FLDM1, FLDM2, FLDM1(normalized), and FLDM2 (normalized) have the same accuracies, precisions, f1 scores, and recalls. This is because the order of columns or the features does not matter since, in the end, we are converting all 32 parts into 1-dimensional data and comparing them
- Due to this, the weights will get adjusted automatically and give the same result.
- We have assumed gaussian distribution for the models to plot the intersection points as given in the problem description.

Logistic Regression

We use the sigmoid function to simulate the probabilistic model. We decide whether a sample is benign or malignant based on whether the probability obtained in the sigmoid is above the threshold or below it.

We run the model on three gradient descent types, three learning rates, and five thresholds. Therefore, we get 45 possible permutations. These permutations are run on ten shuffled datasets to understand better the accuracy, precision, recall, and F1 score.

We have two types of models:-

- LR1: Using un-normalized data
- LR2: Using normalized data.

We are also plotting the graphs of Cost VS Iterations for all the permutations mentioned above.

LR1

The following table describes the training accuracy, accuracy, precision, f1, recall for logistic regression without normalization

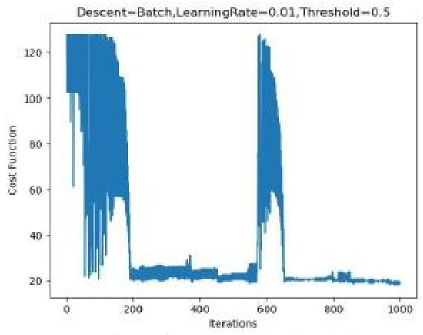
NOTE: The following data is only for 50 epochs

Batch Type	**Learning Rate**	**Threshold**	**Training Accuracy**	**Accuracy**	**Precision**	**F1**	**Recall**
Batch	0.01	0.5	90.981432 %	92.473118 %	93.877551 %	95.172414 %	96.503497 %
Batch	0.01	0.3	91.511936 %	90.322581 %	95.620438 %	93.571429 %	91.608392 %
Batch	0.01	0.4	91.246684 %	86.559140 %	96.093750 %	90.774908 %	86.013986 %
Batch	0.01	0.6	91.246684	93.548387	95.172414	95.833333	96.503497

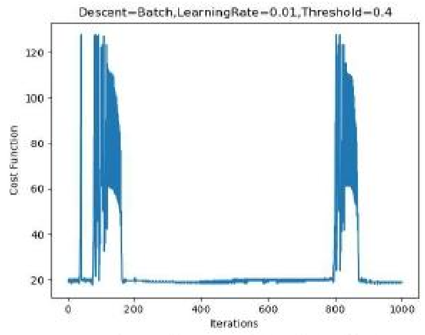
			%	%	%	%	%
Batch	0.01	0.7	91.246684%	88.709677%	95.522388%	92.418773%	89.510490%
Batch	0.001	0.5	91.511936%	90.322581%	95.620438%	93.571429%	91.608392%
Batch	0.001	0.3	91.511936%	90.860215%	95.652174%	93.950178%	92.307692%
Batch	0.001	0.4	91.777188%	90.860215%	95.652174%	93.950178%	92.307692%
Batch	0.001	0.6	92.042440%	91.935484%	95.714286%	94.699647%	93.706294%
Batch	0.001	0.7	92.042440%	92.473118%	95.744681%	95.070423%	94.405594%
Batch	0.0001	0.5	92.042440%	92.473118%	95.744681%	95.070423%	94.405594%
Batch	0.0001	0.3	92.042440%	92.473118%	95.744681%	95.070423%	94.405594%
Batch	0.0001	0.4	92.307692%	92.473118%	95.744681%	95.070423%	94.405594%
Batch	0.0001	0.6	92.042440%	92.473118%	95.744681%	95.070423%	94.405594%
Batch	0.0001	0.7	92.042440%	92.473118%	95.744681%	95.070423%	94.405594%
Mini-Batch	0.01	0.5	82.493369%	68.817204%	98.850575%	74.782609%	60.139860%
Mini-Batch	0.01	0.3	84.615385%	73.655914%	97.959184%	79.668050%	67.132867%
Mini-Batch	0.01	0.4	84.615385%	72.580645%	97.916667%	78.661088%	65.734266%
Mini-Batch	0.01	0.6	86.206897%	76.344086%	97.142857%	82.258065%	71.328671%

Mini-Batch	0.01	0.7	86.472149 %	75.268817 %	97.087379 %	81.300813 %	69.930070 %
Mini-Batch	0.001	0.5	90.981432 %	88.172043 %	96.183206 %	91.970803 %	88.111888 %
Mini-Batch	0.001	0.3	90.981432 %	88.172043 %	96.183206 %	91.970803 %	88.111888 %
Mini-Batch	0.001	0.4	91.246684 %	88.709677 %	96.212121 %	92.363636 %	88.811189 %
Mini-Batch	0.001	0.6	92.042440 %	88.172043 %	95.488722 %	92.028986 %	88.811189 %
Mini-Batch	0.001	0.7	92.042440 %	88.172043 %	95.488722 %	92.028986 %	88.811189 %
Mini-Batch	0.0001	0.5	92.042440 %	88.172043 %	96.183206 %	91.970803 %	88.111888 %
Mini-Batch	0.0001	0.3	92.042440 %	88.172043 %	96.183206 %	91.970803 %	88.111888 %
Mini-Batch	0.0001	0.4	92.042440 %	88.172043 %	96.183206 %	91.970803 %	88.111888 %
Mini-Batch	0.0001	0.6	92.042440 %	88.172043 %	96.183206 %	91.970803 %	88.111888 %
Mini-Batch	0.0001	0.7	92.042440 %	88.172043 %	96.183206 %	91.970803 %	88.111888 %
Stochastic	0.01	0.5	83.819629 %	82.956989 %	86.059904 %	82.036438 %	82.681899 %
Stochastic	0.01	0.3	84.562334 %	83.763441 %	83.152434 %	83.757389 %	88.560067 %
Stochastic	0.01	0.4	86.233422 %	85.161290 %	84.939365 %	84.695109 %	87.694739 %
Stochastic	0.01	0.6	87.082228 %	85.430108 %	86.112964 %	85.103543 %	87.333717 %
Stoch	0.01	0.7	88.381963	86.827957	86.450126	86.442314	88.791391

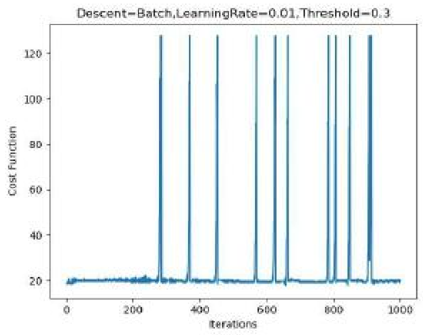
astic			%	%	%	%	%
Stochastic	0.001	0.5	91.140584%	90.591398%	89.444443%	89.973399%	90.724649%
Stochastic	0.001	0.3	90.928382%	90.430108%	89.213008%	89.769339%	90.569712%
Stochastic	0.001	0.4	91.087533%	90.645161%	89.788611%	90.007348%	90.405564%
Stochastic	0.001	0.6	91.273210%	90.591398%	90.183779%	89.845608%	89.694275%
Stochastic	0.001	0.7	91.352785%	90.860215%	90.432698%	90.148551%	90.056610%
Stochastic	0.0001	0.5	91.909814%	91.344086%	90.819672%	90.706287%	90.643474%
Stochastic	0.0001	0.3	91.909814%	91.344086%	90.810567%	90.705072%	90.643474%
Stochastic	0.0001	0.4	91.909814%	91.344086%	90.857814%	90.725272%	90.643474%
Stochastic	0.0001	0.6	91.856764%	91.344086%	90.772426%	90.686086%	90.643474%
Stochastic	0.0001	0.7	91.936340%	91.344086%	90.857814%	90.725272%	90.643474%



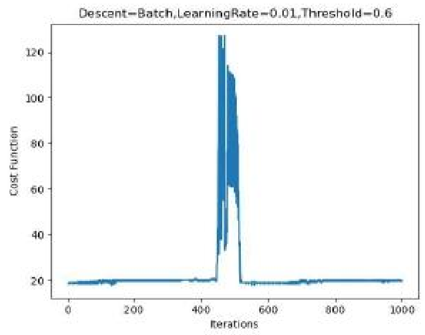
Descent: Batch Learning Rate: 0.01 Threshold: 0.5



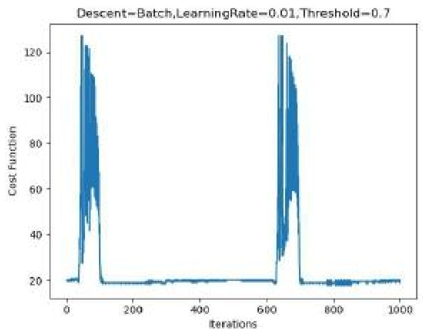
Descent: Batch Learning Rate: 0.01 Threshold: 0.4



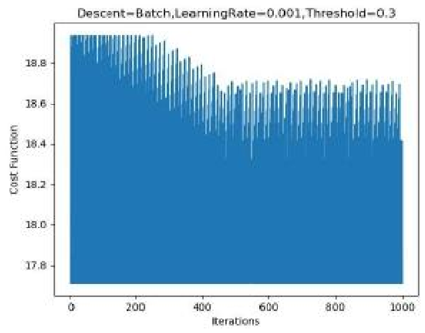
Descent: Batch Learning Rate: 0.01 Threshold: 0.3



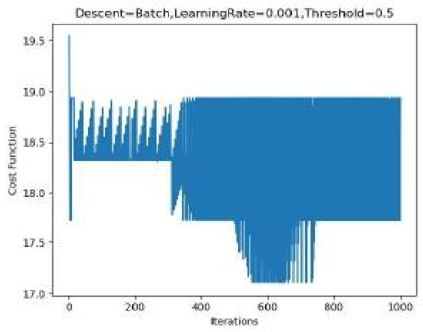
Descent: Batch Learning Rate: 0.01 Threshold: 0.6



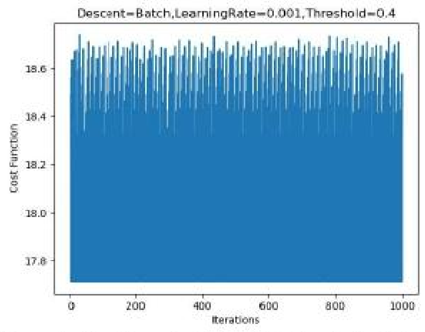
Descent: Batch Learning Rate: 0.01 Threshold: 0.7



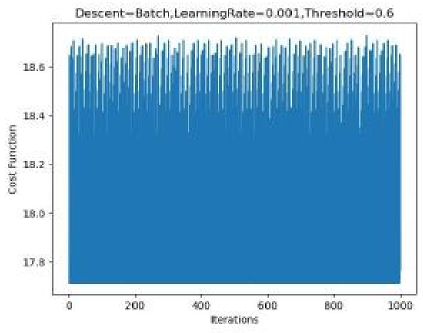
Descent: Batch Learning Rate: 0.001 Threshold: 0.3



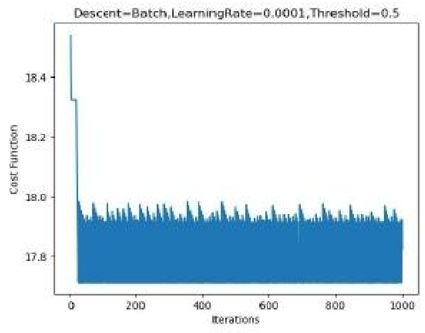
Descent: Batch Learning Rate: 0.001 Threshold: 0.5



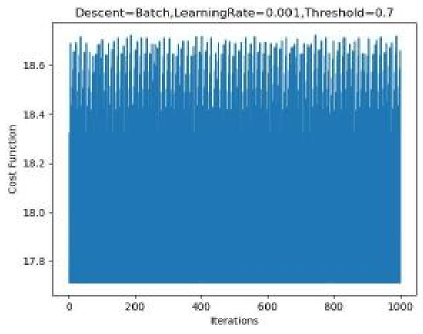
Descent: Batch Learning Rate: 0.001 Threshold: 0.4



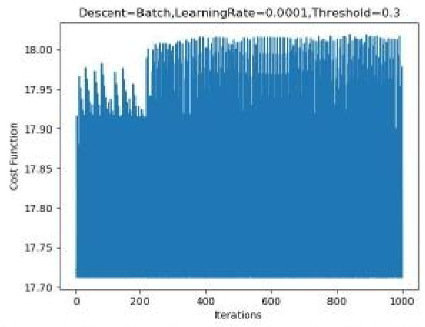
```
Descent: Batch Learning Rate: 0.001 Threshold: 0.6
```



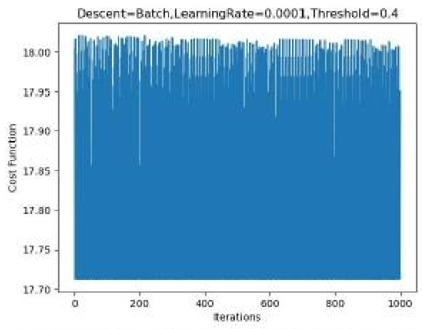
```
Descent: Batch Learning Rate: 0.0001 Threshold: 0.5
```



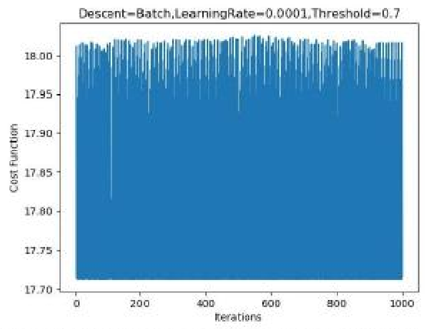
```
Descent: Batch Learning Rate: 0.001 Threshold: 0.7
```



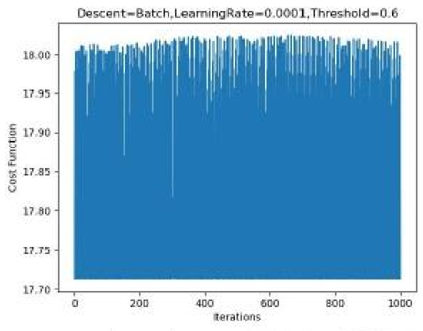
```
Descent: Batch Learning Rate: 0.0001 Threshold: 0.3
```



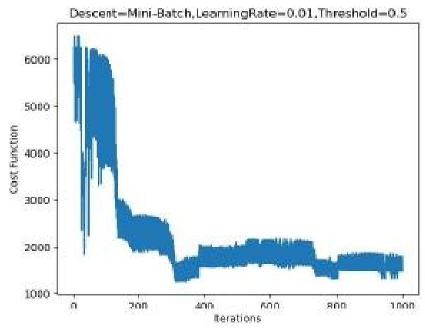
```
Descent: Batch Learning Rate: 0.0001 Threshold: 0.4
```



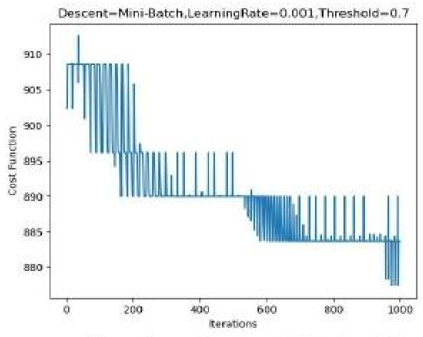
```
Descent: Batch Learning Rate: 0.0001 Threshold: 0.7
```



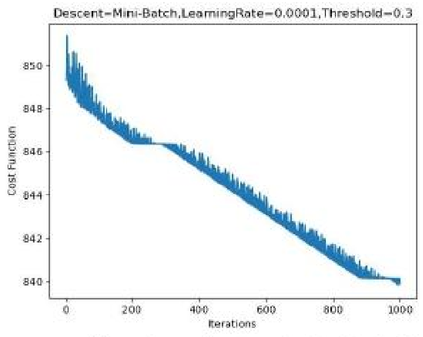
```
Descent: Batch Learning Rate: 0.0001 Threshold: 0.6
```



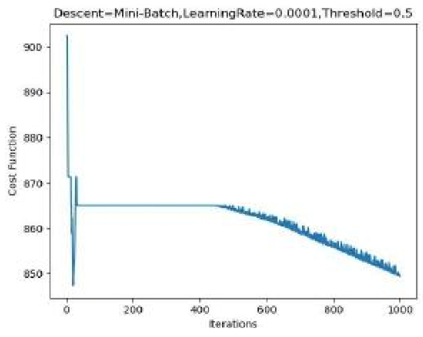
```
Descent: Mini-Batch Learning Rate: 0.01 Threshold:
0.5
```



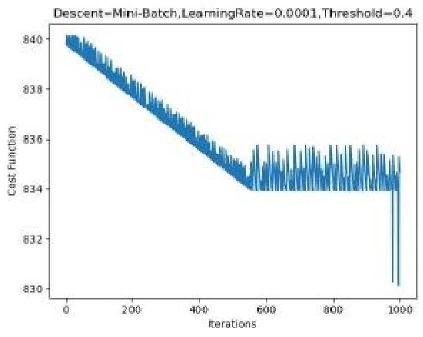
```
Descent: Mini-Batch Learning Rate: 0.001 Threshold:
0.7
```



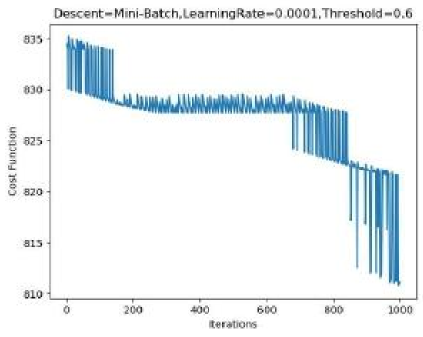
```
Descent: Mini-Batch Learning Rate: 0.0001 Threshold:
0.3
```



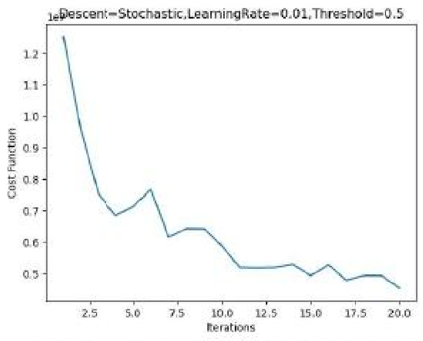
```
Descent: Mini-Batch Learning Rate: 0.0001 Threshold:
0.5
```



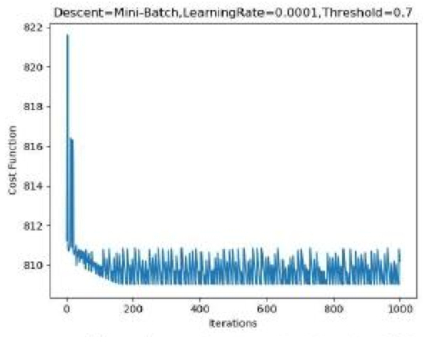
```
Descent: Mini-Batch Learning Rate: 0.0001 Threshold:
0.4
```



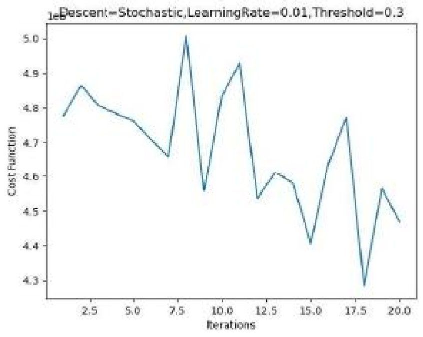
```
Descent: Mini-Batch Learning Rate: 0.0001 Threshold:
0.6
```



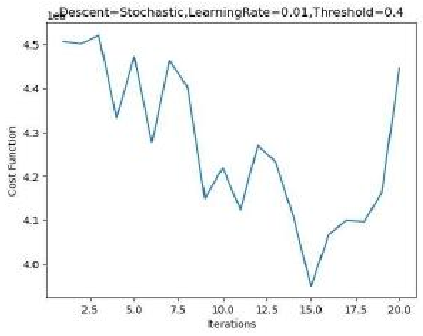
```
Descent: Stochastic Learning Rate: 0.01 Threshold:
0.5
```



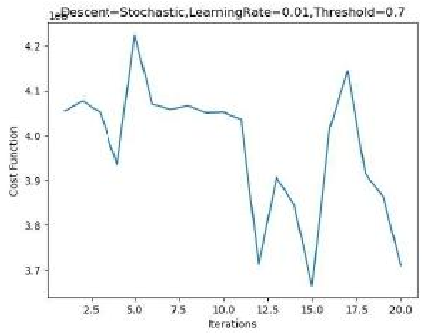
```
Descent: Mini-Batch Learning Rate: 0.0001 Threshold:
0.7
```



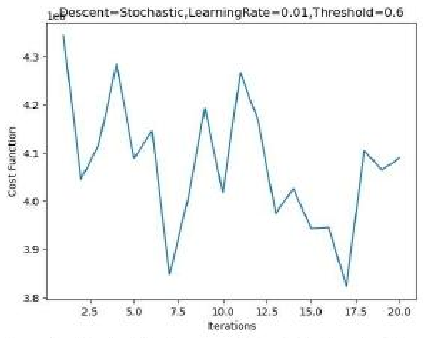
```
Descent: Stochastic Learning Rate: 0.01 Threshold:
0.3
```



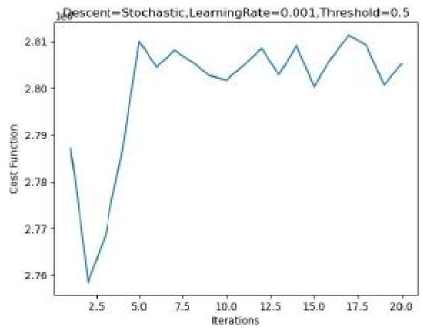
Descent: Stochastic Learning Rate: 0.01 Threshold:
0.4



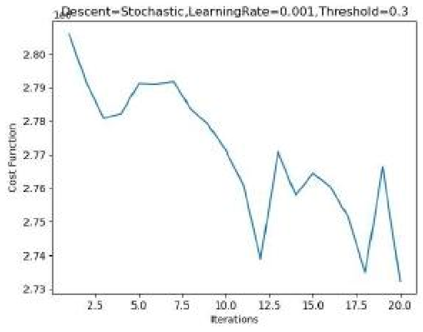
Descent: Stochastic Learning Rate: 0.01 Threshold:
0.7



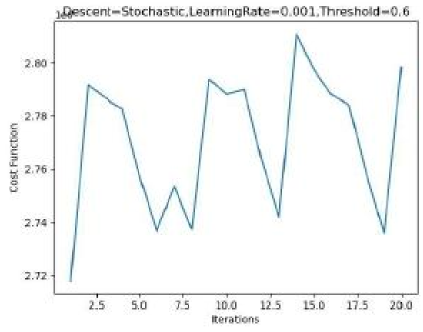
Descent: Stochastic Learning Rate: 0.01 Threshold:
0.6



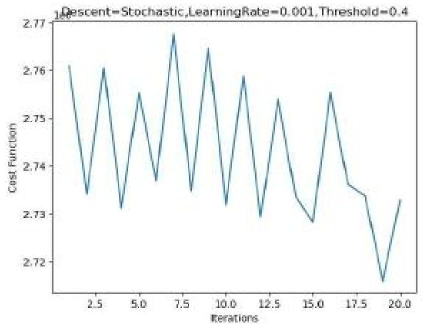
Descent: Stochastic Learning Rate: 0.001 Threshold:
0.5



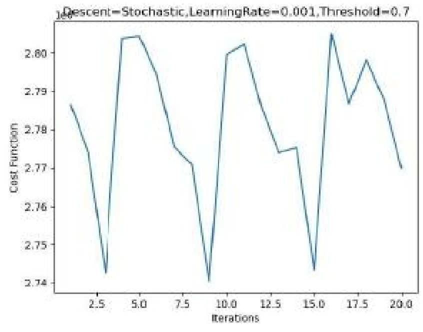
Descent: Stochastic Learning Rate: 0.001 Threshold:
0.3



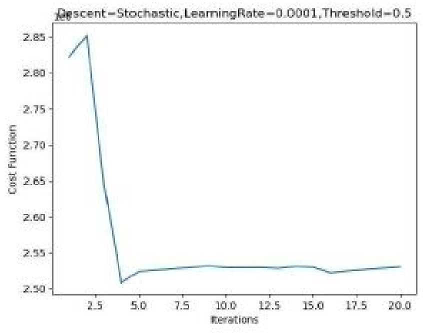
Descent: Stochastic Learning Rate: 0.001 Threshold:
0.6



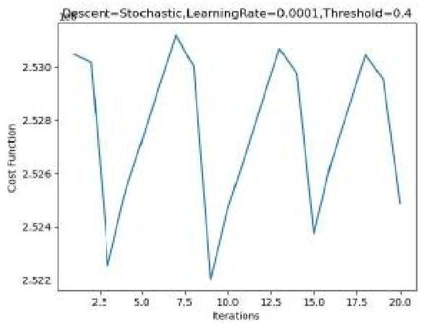
Descent: Stochastic Learning Rate: 0.001 Threshold:
0.4



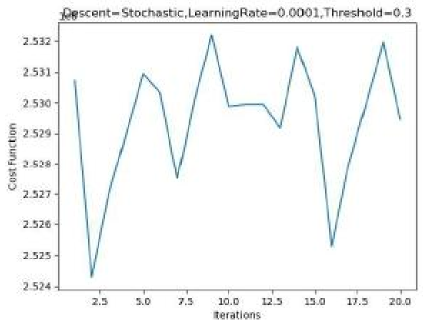
Descent: Stochastic Learning Rate: 0.001 Threshold:
0.7



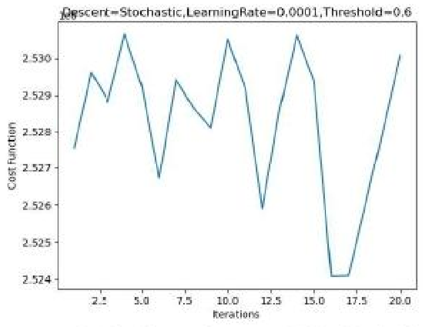
```
Descent: Stochastic Learning Rate: 0.0001 Threshold:
0.5
```



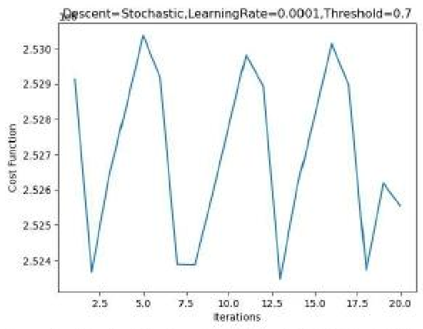
```
Descent: Stochastic Learning Rate: 0.0001 Threshold:
0.4
```



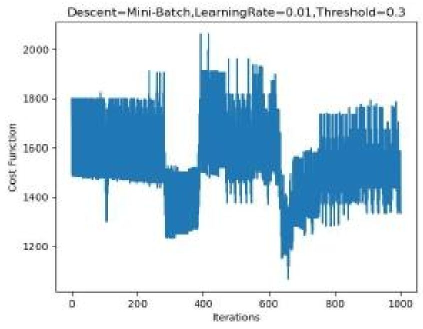
```
Descent: Stochastic Learning Rate: 0.0001 Threshold:
0.3
```



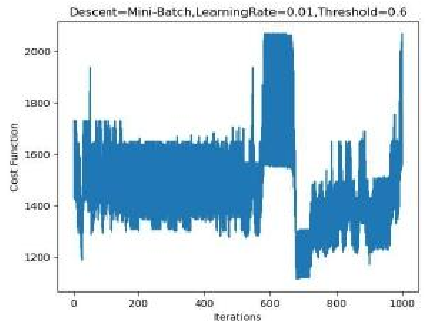
```
Descent: Stochastic Learning Rate: 0.0001 Threshold:
0.6
```



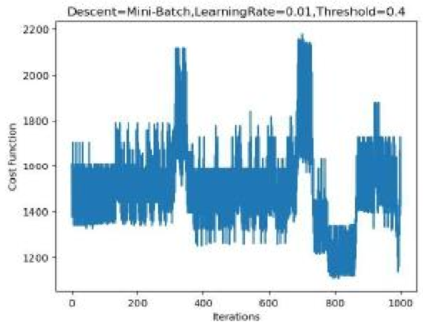
```
Descent: Stochastic Learning Rate: 0.0001 Threshold:
0.7
```



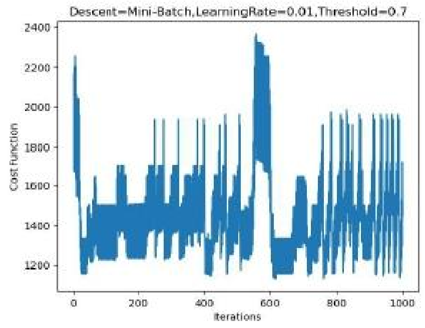
```
Descent: Mini-Batch Learning Rate: 0.01 Threshold:
0.3
```



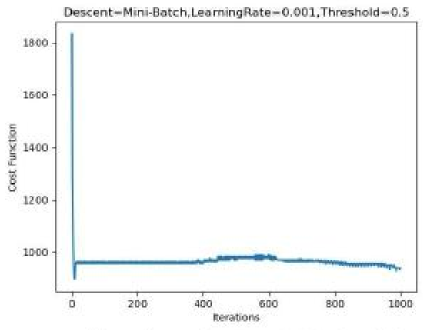
```
Descent: Mini-Batch Learning Rate: 0.01 Threshold:
0.6
```



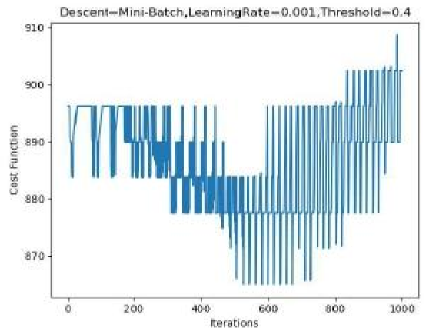
```
Descent: Mini-Batch Learning Rate: 0.01 Threshold:
0.4
```



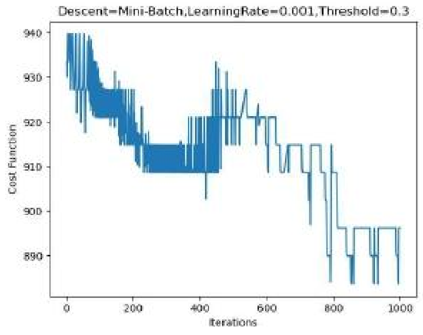
```
Descent: Mini-Batch Learning Rate: 0.01 Threshold:
0.7
```



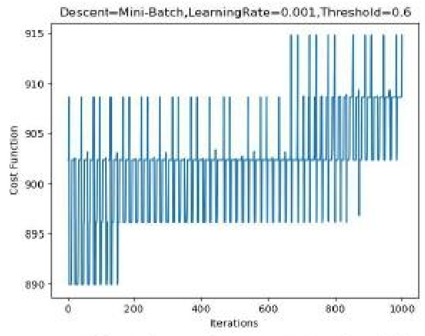
```
Descent: Mini-Batch Learning Rate: 0.001 Threshold:
0.5
```



```
Descent: Mini-Batch Learning Rate: 0.001 Threshold:
0.4
```



```
Descent: Mini-Batch Learning Rate: 0.001 Threshold:
0.3
```



```
Descent: Mini-Batch Learning Rate: 0.001 Threshold:
0.6
```

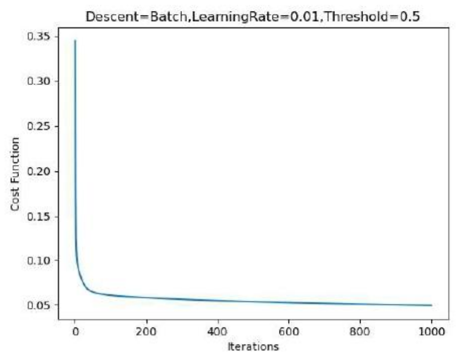
LR2

Batch Type	Learning Rate	Threshold	Training Accuracy	Accuracy	Precision	F1	Recall
Batch	0.01	0.5	98.608924%	97.659574%	97.784676%	97.545594%	97.330866%
Batch	0.01	0.3	98.372703%	97.446809%	96.302100%	97.255979%	98.243733%
Batch	0.01	0.4	98.792651%	97.712766%	97.147909%	97.482025%	97.858269%
Batch	0.01	0.6	98.661417%	97.287234%	98.552443%	97.047715%	95.655056%
Batch	0.01	0.7	98.372703%	96.914894%	98.671509%	96.571317%	94.613126%
Batch	0.001	0.5	98.897638%	97.500000%	97.593613%	97.142012%	96.744813%
Batch	0.001	0.3	98.582677%	97.234043%	95.768640%	97.017843%	98.338917%
Batch	0.001	0.4	98.871391%	97.393617%	96.667854%	97.162110%	97.695023%
Batch	0.001	0.6	98.661417%	97.127660%	98.207414%	96.880882%	95.650744%
Batch	0.001	0.7	98.398950%	96.968085%	98.674533%	96.659552%	94.774417%
Batch	0.0001	0.5	98.950131%	97.393617%	97.435362%	97.026130%	96.657857%
Batch	0.0001	0.3	98.556430%	97.287234%	95.929842%	97.096479%	98.338917%
Batch	0.0001	0.4	98.871391%	97.287234%	96.584144%	97.043300%	97.543085%
Batch	0.0001	0.6	98.661417%	97.07446	98.04074	96.80714	95.65074

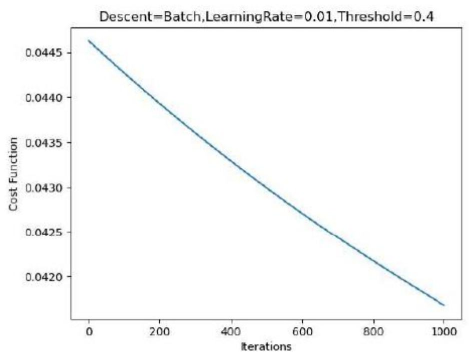
				8%	7%	1%	4%
Batch	0.0001	0.7	98.398950%	96.968085%	98.674533%	96.659552%	94.774417%
Mini-Batch	0.01	0.5	98.713911%	97.659574%	97.710971%	97.542836%	97.404544%
Mini-Batch	0.01	0.3	98.372703%	97.606383%	96.292711%	97.447144%	98.649461%
Mini-Batch	0.01	0.4	98.713911%	97.393617%	96.819750%	97.192216%	97.601712%
Mini-Batch	0.01	0.6	98.740157%	97.180851%	98.128565%	96.895975%	95.740514%
Mini-Batch	0.01	0.7	98.398950%	96.755319%	98.498358%	96.445784%	94.542518%
Mini-Batch	0.001	0.5	98.897638%	97.340426%	97.431234%	97.128823%	96.880716%
Mini-Batch	0.001	0.3	98.530184%	97.127660%	95.904414%	96.958891%	98.082359%
Mini-Batch	0.001	0.4	98.845144%	97.180851%	96.576129%	96.954212%	97.376168%
Mini-Batch	0.001	0.6	98.740157%	97.127660%	98.042398%	96.853456%	95.740514%
Mini-Batch	0.001	0.7	98.398950%	96.755319%	98.498358%	96.445784%	94.542518%
Mini-Batch	0.0001	0.5	98.897638%	97.234043%	97.267299%	96.972573%	96.731462%
Mini-Batch	0.0001	0.3	98.530184%	97.127660%	95.904414%	96.958891%	98.082359%
Mini-Batch	0.0001	0.4	98.845144%	97.127660%	96.499938%	96.914714%	97.376168%
Mini-Batch	0.0001	0.6	98.740157%	97.127660%	98.042398%	96.853456%	95.740514%

Mini-Batch	0.0001	0.7	98.398950%	96.755319%	98.498358%	96.445784%	94.542518%
Stochastic	0.01	0.5	98.188976%	98.031915%	98.045485%	97.975690%	97.911427%
Stochastic	0.01	0.3	97.585302%	97.606383%	96.056764%	97.500764%	98.998392%
Stochastic	0.01	0.4	98.372703%	98.138298%	97.497207%	98.005429%	98.522151%
Stochastic	0.01	0.6	98.372703%	97.234043%	98.370614%	96.981028%	95.661966%
Stochastic	0.01	0.7	97.952756%	96.702128%	99.169266%	96.450521%	93.947195%
Stochastic	0.001	0.5	98.713911%	97.978723%	98.034686%	97.835692%	97.661102%
Stochastic	0.001	0.3	98.005249%	97.553191%	96.111579%	97.461794%	98.861011%
Stochastic	0.001	0.4	98.582677%	98.031915%	97.341239%	97.850547%	98.372898%
Stochastic	0.001	0.6	98.530184%	97.287234%	98.372139%	97.025455%	95.748922%
Stochastic	0.001	0.7	98.031496%	96.808511%	99.172513%	96.542151%	94.121108%
Stochastic	0.0001	0.5	98.713911%	97.978723%	98.034686%	97.835692%	97.661102%
Stochastic	0.0001	0.3	98.005249%	97.500000%	96.025034%	97.384702%	98.798714%
Stochastic	0.0001	0.4	98.582677%	98.031915%	97.341239%	97.850547%	98.372898%
Stochastic	0.0001	0.6	98.556430%	97.287234%	98.372139%	97.025455%	95.748922%
Stochas	0.0001	0.7	98.057743%	96.80851	99.17251	96.54215	94.12110

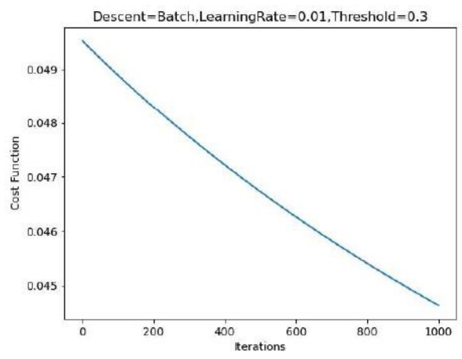
tic				1%	3%	1%	8%



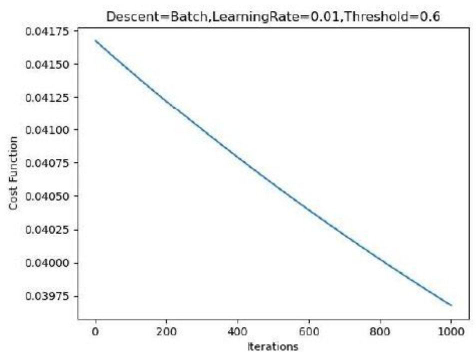
```
Descent: Batch Learning Rate: 0.01 Threshold: 0.5
```



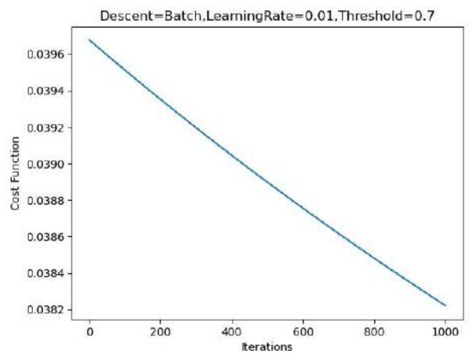
```
Descent: Batch Learning Rate: 0.01 Threshold: 0.4
```



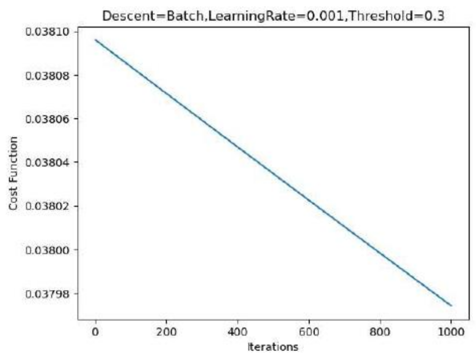
```
Descent: Batch Learning Rate: 0.01 Threshold: 0.3
```



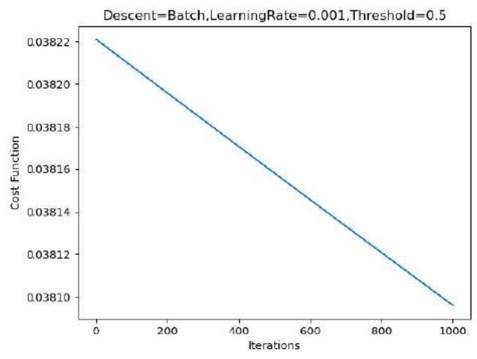
```
Descent: Batch Learning Rate: 0.01 Threshold: 0.6
```



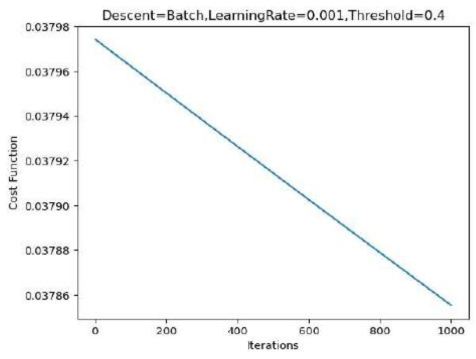
```
Descent: Batch Learning Rate: 0.01 Threshold: 0.7
```



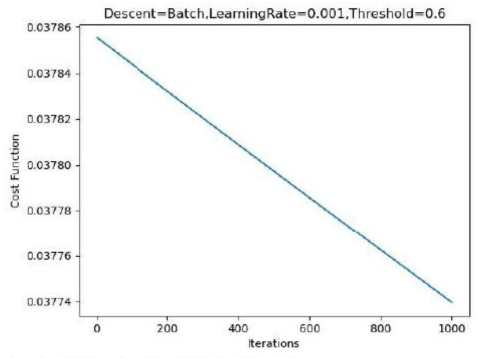
```
Descent: Batch Learning Rate: 0.001 Threshold: 0.3
```



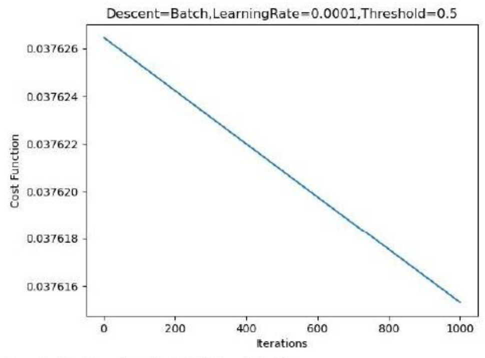
```
Descent: Batch Learning Rate: 0.001 Threshold: 0.5
```



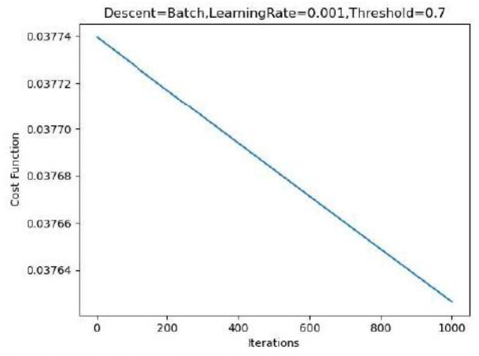
```
Descent: Batch Learning Rate: 0.001 Threshold: 0.4
```



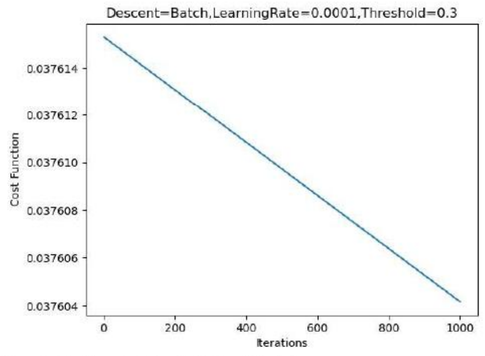
Descent: Batch Learning Rate: 0.001 Threshold: 0.6



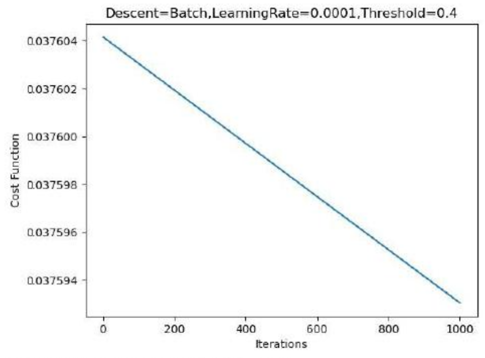
Descent: Batch Learning Rate: 0.0001 Threshold: 0.5



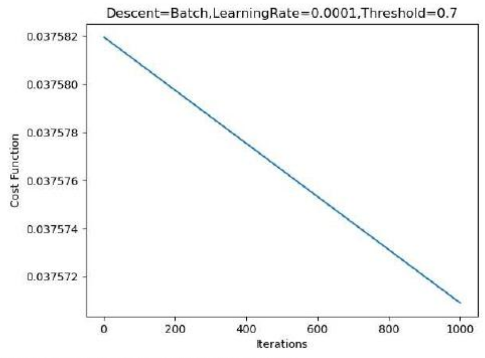
Descent: Batch Learning Rate: 0.001 Threshold: 0.7



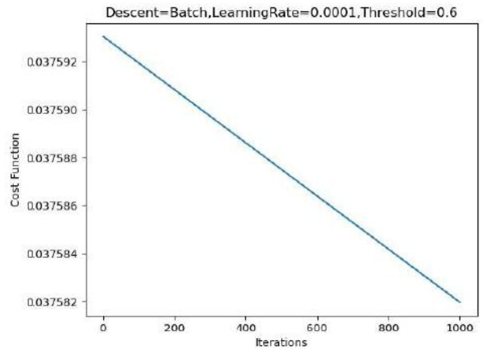
Descent: Batch Learning Rate: 0.0001 Threshold: 0.3



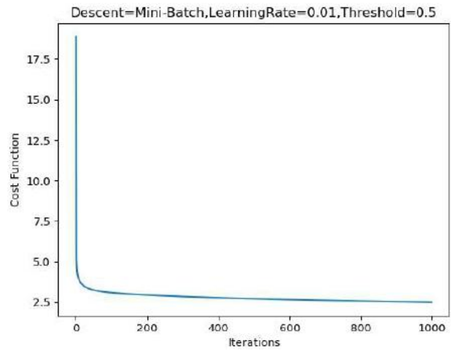
Descent: Batch Learning Rate: 0.0001 Threshold: 0.4



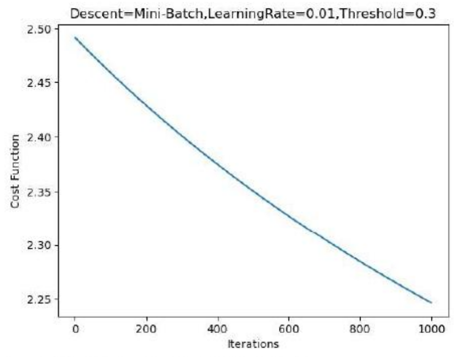
Descent: Batch Learning Rate: 0.0001 Threshold: 0.7



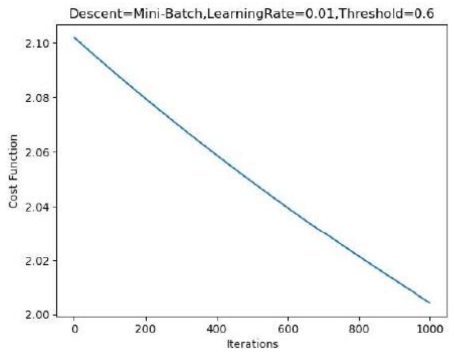
Descent: Batch Learning Rate: 0.0001 Threshold: 0.6



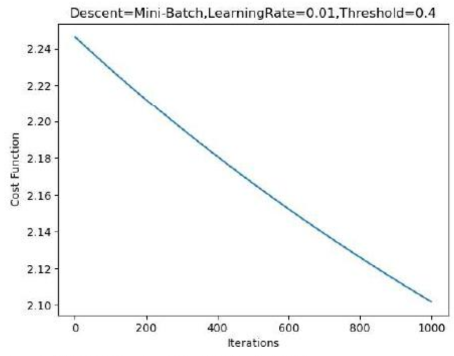
Descent: Mini-Batch Learning Rate: 0.01 Threshold: 0.5



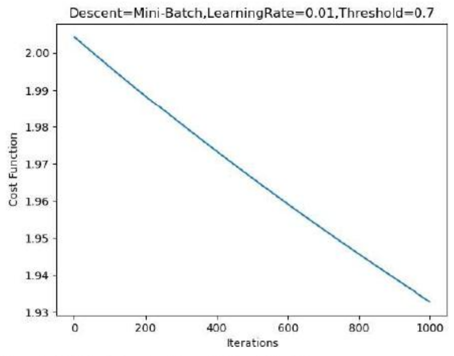
Descent: Mini-Batch Learning Rate: 0.01 Threshold: 0.3



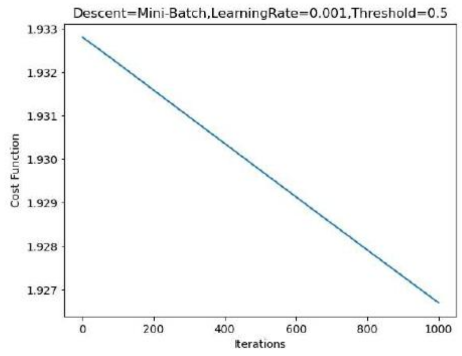
Descent: Mini-Batch Learning Rate: 0.01 Threshold: 0.6



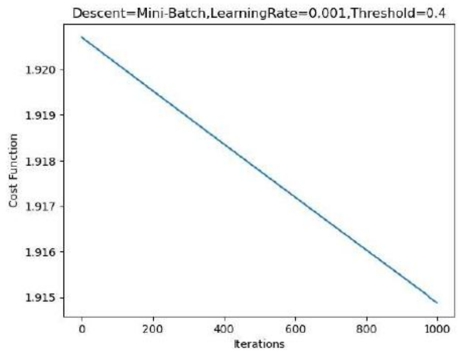
Descent: Mini-Batch Learning Rate: 0.01 Threshold: 0.4



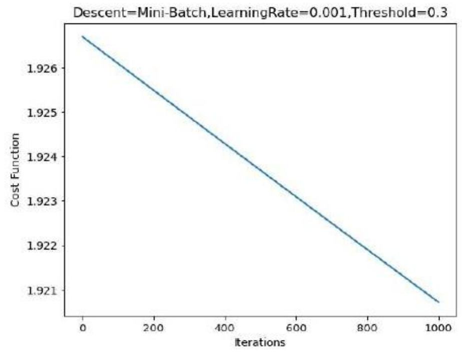
Descent: Mini-Batch Learning Rate: 0.01 Threshold: 0.7



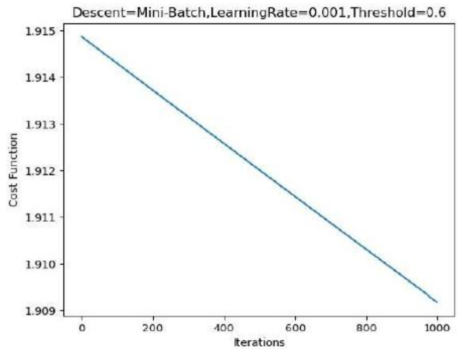
Descent: Mini-Batch Learning Rate: 0.001 Threshold: 0.5



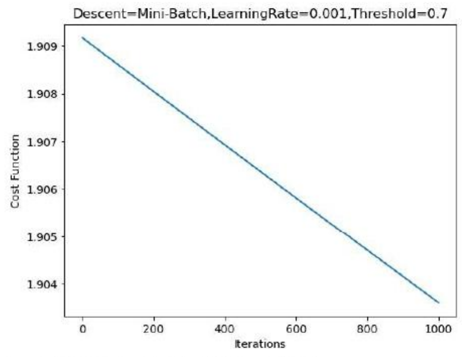
Descent: Mini-Batch Learning Rate: 0.001 Threshold: 0.4



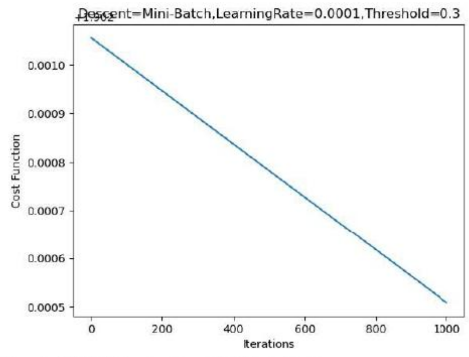
Descent: Mini-Batch Learning Rate: 0.001 Threshold: 0.3



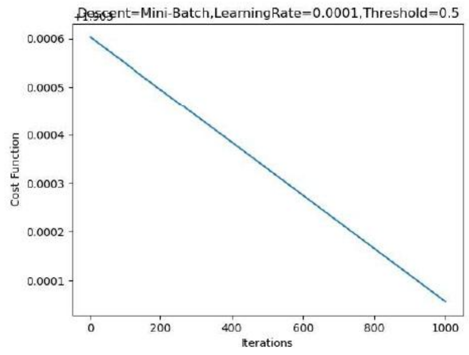
Descent: Mini-Batch Learning Rate: 0.001 Threshold: 0.6



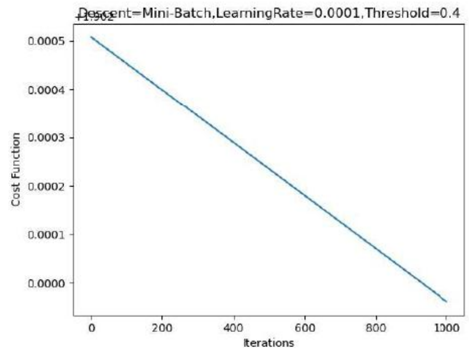
Descent: Mini-Batch Learning Rate: 0.001 Threshold: 0.7



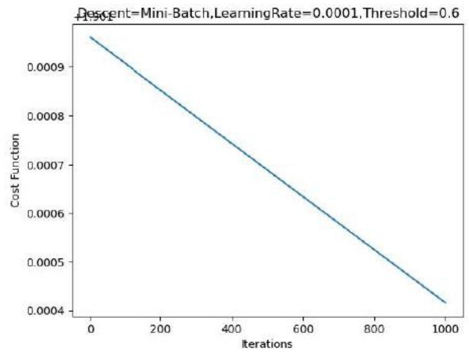
Descent: Mini-Batch Learning Rate: 0.0001 Threshold: 0.3



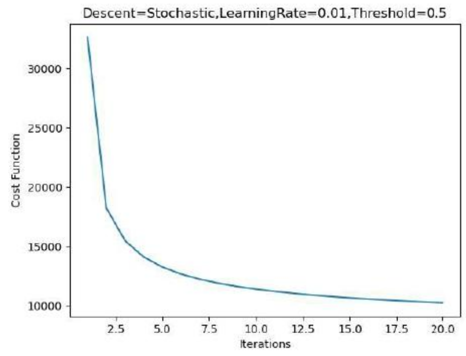
Descent: Mini-Batch Learning Rate: 0.0001 Threshold: 0.5



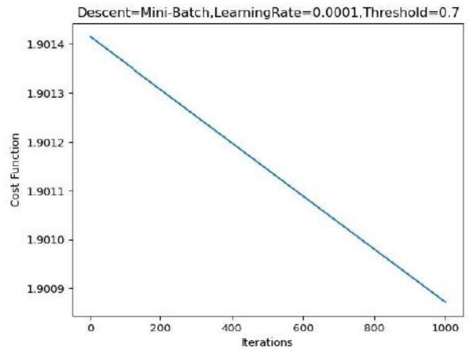
Descent: Mini-Batch Learning Rate: 0.0001 Threshold: 0.4



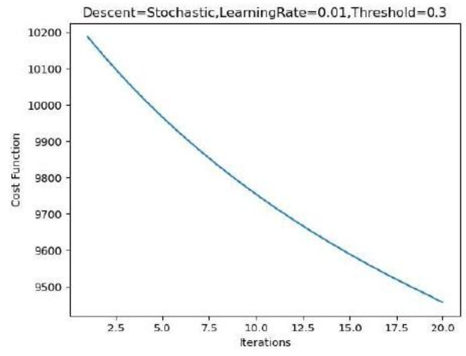
Descent: Mini-Batch Learning Rate: 0.0001 Threshold: 0.6



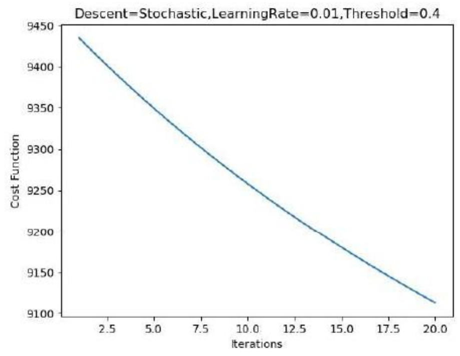
Descent: Stochastic Learning Rate: 0.01 Threshold: 0.5



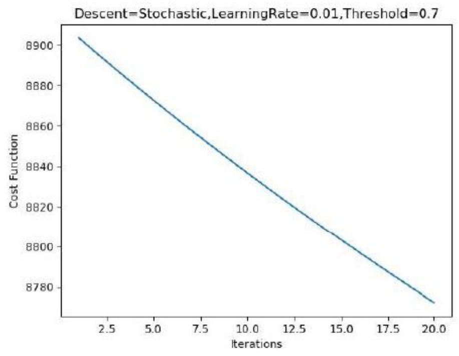
Descent: Mini-Batch Learning Rate: 0.0001 Threshold: 0.7



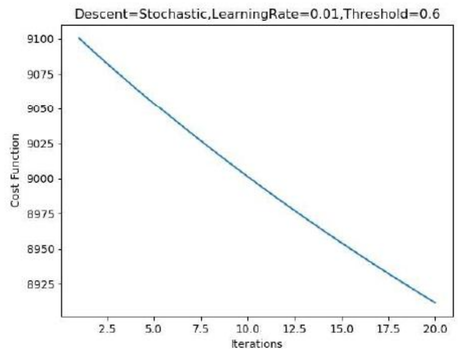
Descent: Stochastic Learning Rate: 0.01 Threshold: 0.3



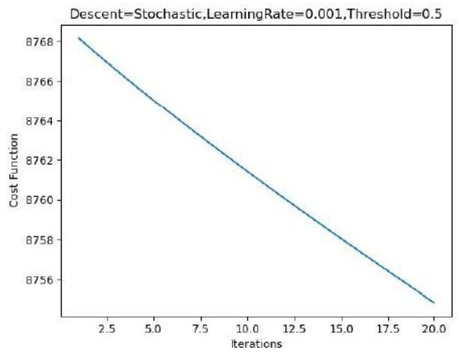
```
Descent: Stochastic Learning Rate: 0.01 Threshold: 0.4
```



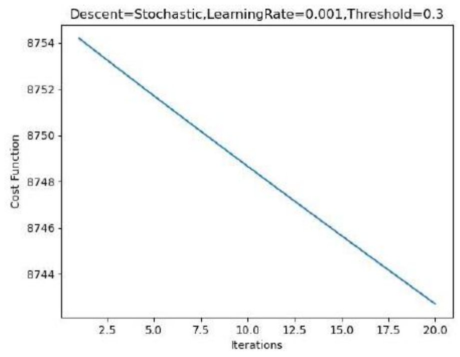
```
Descent: Stochastic Learning Rate: 0.01 Threshold: 0.7
```



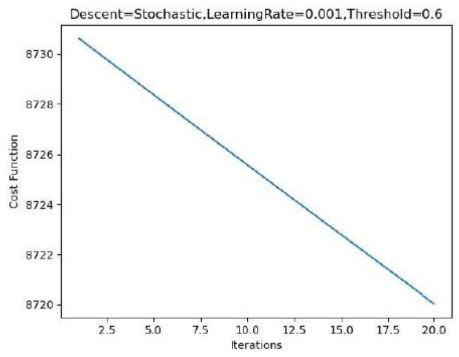
```
Descent: Stochastic Learning Rate: 0.01 Threshold: 0.6
```



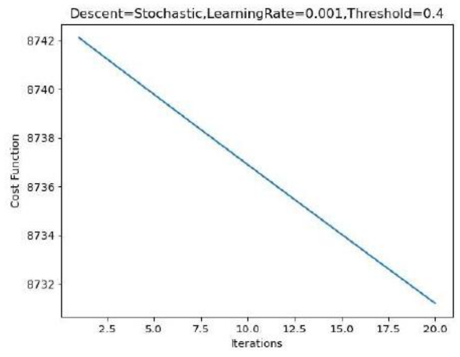
```
Descent: Stochastic Learning Rate: 0.001 Threshold: 0.5
```



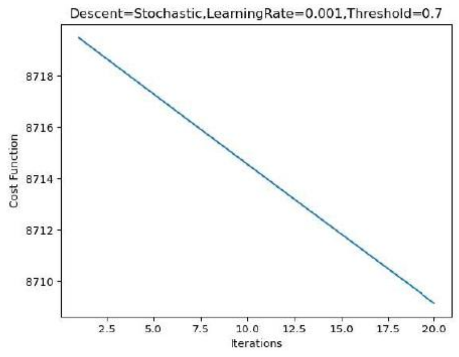
```
Descent: Stochastic Learning Rate: 0.001 Threshold: 0.3
```



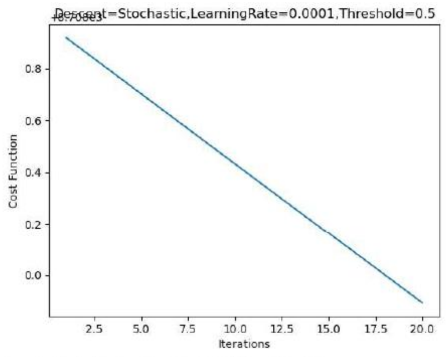
```
Descent: Stochastic Learning Rate: 0.001 Threshold: 0.6
```



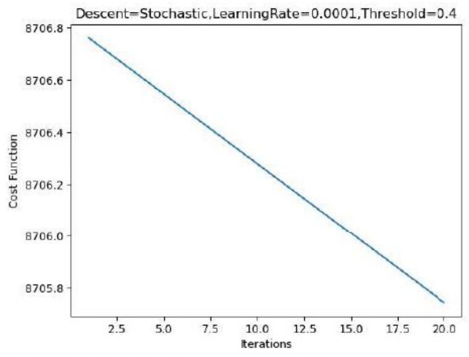
```
Descent: Stochastic Learning Rate: 0.001 Threshold: 0.4
```



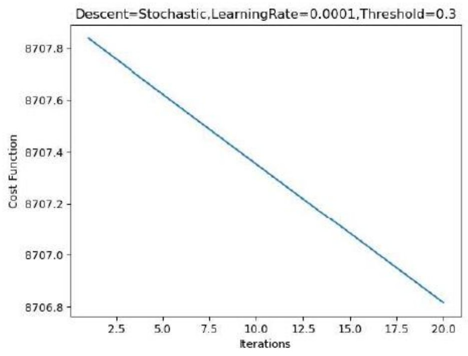
```
Descent: Stochastic Learning Rate: 0.001 Threshold: 0.7
```



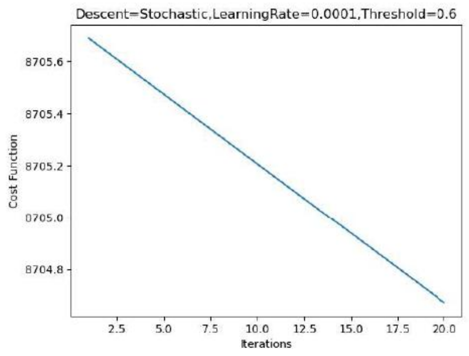
```
Descent: Stochastic Learning Rate: 0.0001 Threshold: 0.5
```



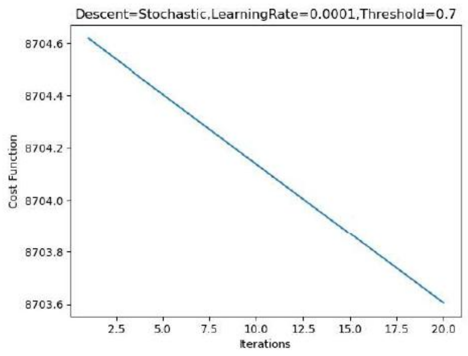
```
Descent: Stochastic Learning Rate: 0.0001 Threshold: 0.4
```



```
Descent: Stochastic Learning Rate: 0.0001 Threshold: 0.3
```



```
Descent: Stochastic Learning Rate: 0.0001 Threshold: 0.6
```



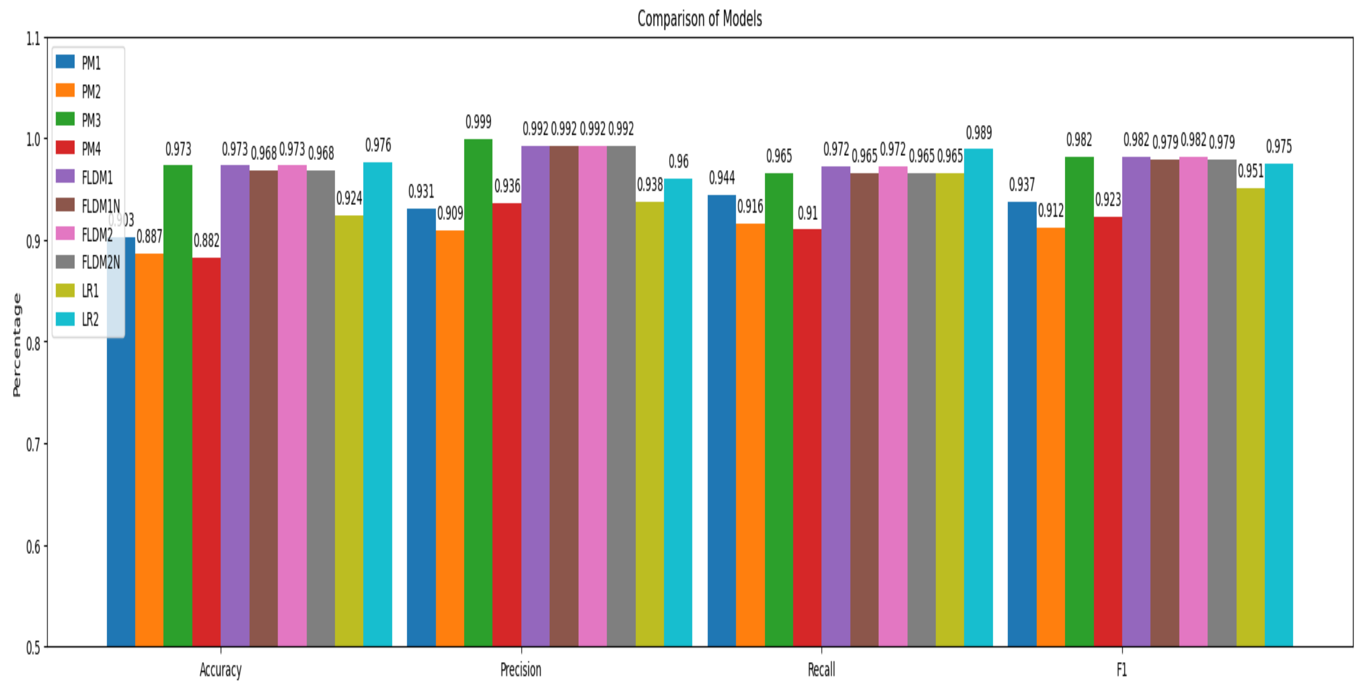
```
Descent: Stochastic Learning Rate: 0.0001 Threshold: 0.7
```

Analysis

- We observe that the variance of the various evaluation metrics is lesser in LR2 than in LR1 because the dataset used in LR2 is normalized. In an un-normalized dataset some features may have a larger range than others thus resulting in a suboptimal decision boundary, thus the larger features may dominate the training process leading to biased and inaccurate coefficients.
- Normalization is the process of scaling all input features to a common scale. This prevents any feature from dominating and results in a more accurate decision boundary.
- If we observe the threshold values also, <u>accuracy increases with threshold upto 0.5 and then decreases as threshold increases</u>
- Testing accuracy is <u>inversely proportional to learning rate</u> as lower learning rate prevents the minima from being overshoot
- But lower learning rate, increases time in training the model
- Batch is better than mini batch better stochastic and this is because in batch we alter w only after running through all the input data while in mini batch we take small batches and in stochastic we update for every tuple which leads to randomized results.
- Time taken to reach convergence follows the following order Stochastic < Mini-Batch < Batch.

Comparative Study

- We observe that FLDM runs the fastest, and this is due to the fact that it is independent of the number of iterations that take place in the training of the model
- Perceptron depends a lot on the dataset, and this is because perceptron needs linear
- We observe that LR2 gives a very high accuracy even at low epochs but takes a lot of time for the model to terminate



- We observe that FLDM gives consistent results but perceptron also gives us very high accuracy since this was a linearly separable model