MACHINE LEARNING
BITS F464

ASSIGNMENT - 1

PART - A : Perceptron Model

The Perceptron algorithm is a type of supervised learning algorithm used for binary classification tasks(where we assigned +1 to M and -1 to D in our dataset, where the goal is to predict whether a given input belongs to one of two possible categories

The algorithm works by iteratively updating the weights of the perceptron based on the input data and the correct output labels. The weights are initially set to random values(it is set to zeros in our code), and the algorithm tries to find the best set of weights that can correctly classify the training data.

At each iteration, the algorithm processes one input example at a time and calculates the dot product between the input features and the current weights. If the dot product is greater than or equal to a threshold value(that is 0 in our code), the perceptron predicts that the input belongs to one category, otherwise it predicts that it belongs to the other category.

If the prediction is incorrect, the algorithm adjusts the weights vector so that the prediction gives the correct label. it updates the weights by adding or subtracting the input features.

The algorithm continues iterating over the training examples until it reaches a stopping criterion, such as a maximum number of iterations or when the error rate on the training data falls below a certain threshold.

Accuracy of the models are as follows -

S.No.	PM1	PM2	PM3	PM4
1	92.02127659574468 %	84.57446808510637 %	96.27659574468085 %	92.02127659574468 %
2	92.5531914893617 %	85.63829787234043 %	96.27659574468085 %	92.5531914893617 %
3	92.5531914893617 %	90.08510638297872 %	93.61702127659575 %	92.5531914893617 %
4	90.42553191489363 %	82.97872340425532 %	92.5531914893617 %	90.42553191489363 %
5	92.02127659574468 %	84.64680851320637 %	96.27659574468085 %	92.02127659574468 %
6	94.68085106382979 %	93.08510638297872 %	93.61702127659575 %	94.68085106382979 %
7	90.42553191489363 %	86.70212765957447 %	93.61702127659575 %	90.42553191489363 %
8	90.95744680851064 %	89.02127659574468 %	95.2127659574468 %	90.95744680851064 %
9	81.91489361702128 %	79.70212765957447 %	92.5531914893617 %	81.91489361702128 %
10	88.29787234042553 %	84.57446808510637 %	93.08510638297872 %	88.29787234042553 %

Average accuracy of PM1 =90.77777777777779%.

Average accuracy of PM2 =94.13829787234042%.

Average accuracy of PM3 =86.12893617021276%.

Average accuracy of PM4 =90.77777777777779%.

Precision values of the models are as follows

S.No.	PM1	PM2	PM3	PM4
1	95.83333333333334	94.47294759299292	98.4375	95.83333333333334
2	96.96969696969697	93.98348925859304	94.9367088607595	96.96969696969697
3	96.63926384927489	94.89038264957398	98.36065573770492	96.63926384927489
4	70.40816326530613	64.83593958498430	94.52054794520548	70.40816326530613

5	93.84615384615384	89.98348934093409	94.14893617021278	93.84615384615384
6	90.625	84.98580309340934	94.44444444444444	90.625
7	62.616822429906534	60.74834093494309	92.95774647887323	62.616822429906534
8	92.95774647887323	89.34983984090942	93.61702127659575	92.95774647887323
9	72.04301075268818	73.45793480340844	96.96969696969697	72.04301075268818
10	64.22018348623854	68.59735893498873	98.59154929577466	64.22018348623854

Average precision of PM1 = 83.07571579066869%

Average precision of PM2 = 81.52033059674116%

Average precision of PM3 =95.91995982362592%

Average precision of PM4 = 83.07571579066869%

Recall values of the model are as follows

S.No.	PM1	PM2	PM3	PM4
1	89.1891891891892	86.45453558498430	95.94594594594594	89.1891891891892
2	95.83333333333334	89.92348934093409	95.58823529411765	95.83333333333334
3	67.64705882352942	64.98580302625434	92.64705882352942	67.64705882352942
4	93.05555555555556	92.8949849378899	88.88888888888889	93.05555555555556
5	88.0	88.0	96.0	88.0
6	97.22222222222221	96.0	97.22222222222221	97.22222222222221
7	89.0625	89.0625	89.0625	89.0625
8	98.50746268656717	95.97373489408948	91.04477611940298	98.50746268656717
9	92.42424242424242	91.34894389983478	96.96969696969697	92.42424242424242
10	91.30434782608695	78.98389438943894	97.10144927536231	91.30434782608695

Average recall of PM1 =89.90410373163215.

Average recall of PM2 =86.29819592476489.

Average recall of PM3 =89.90410373163215.

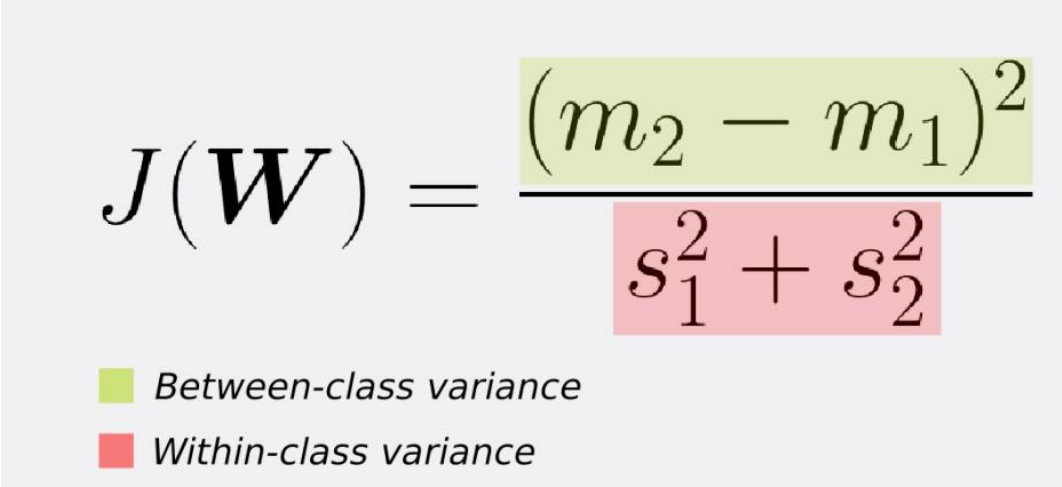
Average recall of PM4 =94.79290468376801.

Part B – Fisher's Linear Discriminant Analysis

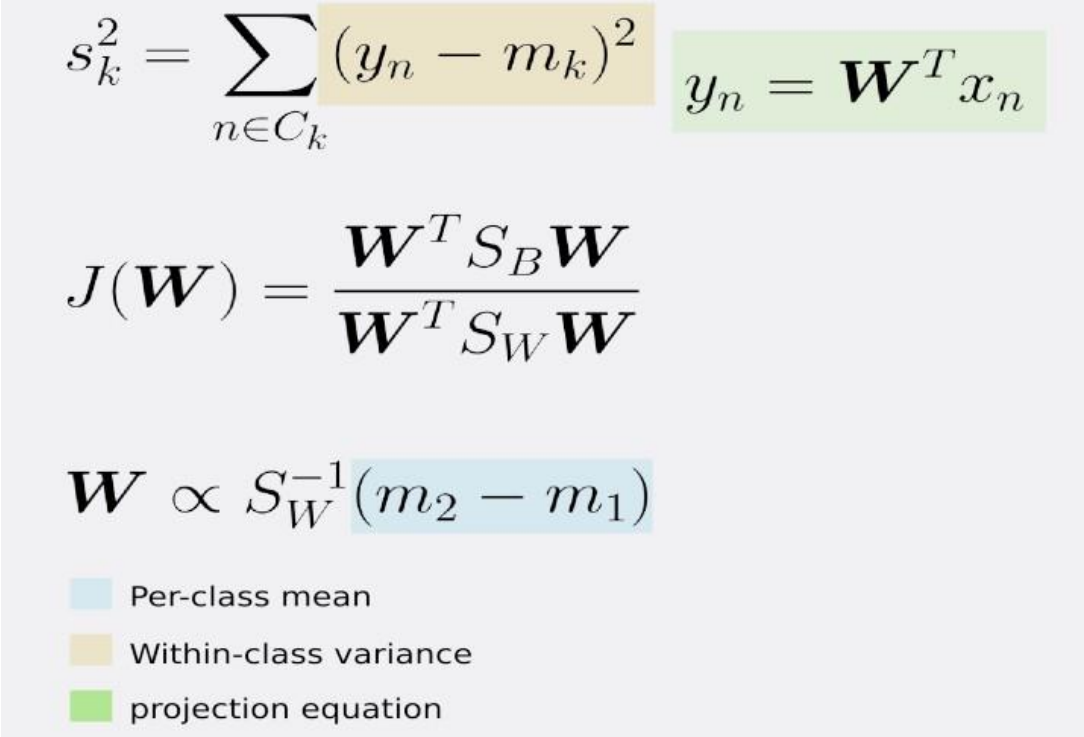
The idea proposed by Fisher is to maximize a function that will give a large separation between the projected class means, while also giving a small variance within each class, thereby minimizing the class overlap.

In other words, FLD selects a projection that maximizes the class separation. To do that, it maximizes the ratio between the between-class variance to the within-class variance.

A large between-class variance means that the projected class averages should be as far apart as possible. On the contrary, a small within-class variance has the effect of keeping the projected data points closer to one another.



W (our desired transformation) is directly proportional to the inverse of the within-class covariance matrix times the difference of the class means.



Implementation:

The weight vector is found such that the mean difference between the clusters is maximized and the variance within the clusters is minimized.

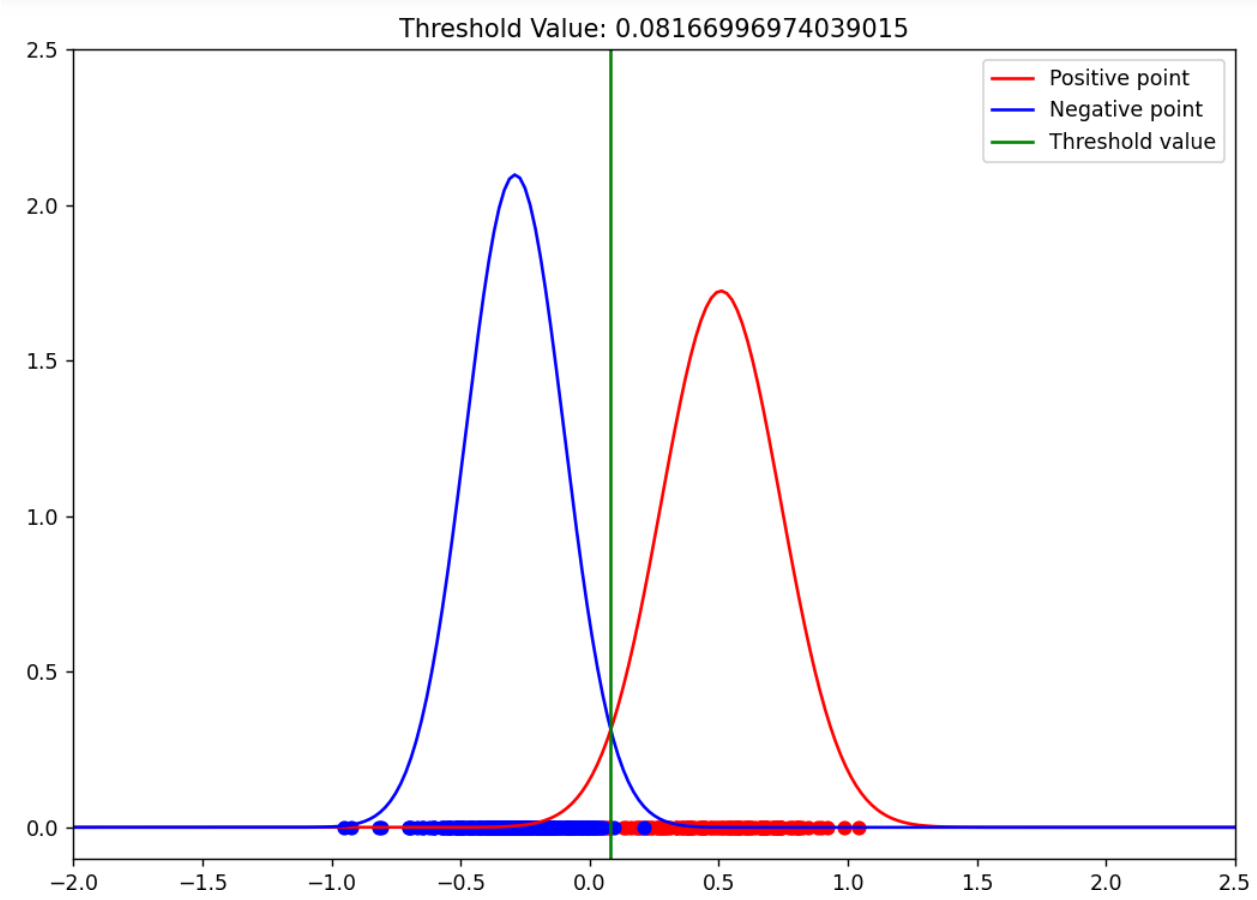
Firstly, the dataset was divided into two parts, one part for each class. Then, the covariance matrix is calculated by using the formula mentioned above. Then , by using the covariance matrix, the weight array is calculated and normalized. Then, the points are projected in 1-D and the point of intersection of their normal curves is calculated.

Finally, the training accuracy and testing accuracy was calculated. It was calculated by classifying the points and comparing with their corresponding original class to find the number of true positives, true negatives, false positives and false negatives.

Training and Testing accuracy for split of 67:33 using random seed =42 (arbitary) are 97.3753280839895 % and 96.80851063829788 %

Above model can referred as FLDM1. It's observed that even if the features are shuffled then also we get the same results which can be referred as FLDM2.

Plot of the reduced clusters and line of intersection in 1-D and corresponding Normal Distribution:



Random training and testing splits Results

S.No	Testing accuracy for FLDM1 and FLDM2	Testing Recall for FLDM1 and FLDM2	Testing Precisison for FLDM1 and FLDM2	Random seed value
1	96.80851063829788	94.73684210526315	97.2972972972973	42
2	96.80851063829788	95.52238805970148	95.52238805970148	35
3	96.80851063829788	92.85714285714286	98.48484848484848	30
4	97.11286089238845	98.52941176470588	98.52941176470588	50
5	97.3404255319149	95.71428571428572	97.10144927536231	70
6	96.27659574468085	94.8051948051948	96.05263157894737	100
7	97.3753280839895	98.46153846153847	98.46153846153847	80

8	94.68085106382979	88.40579710144928	96.82539682539682	65
9	97.87234042553192	94.5945945945946	100.0	25
10	97.9002624671916	98.57142857142858	98.57142857142858	15
Average	96.87843906478955	95.21578947368421	97.3553151072961	

PART C - LOGISTIC REGRESSION

Logistic regression is based on a probabilistic model that estimates the probability of an input belonging to a certain category. The algorithm works by first initializing the model parameters (the coefficients of the linear equation that models the relationship between the input features and the output). It then iteratively updates these parameters based on the training data until the algorithm converges to a set of parameters that can accurately predict the output for new inputs.

At each iteration, the algorithm calculates the probability of each input belonging to the positive category using a sigmoid function, which maps any real-valued number to a probability between 0 and 1. The sigmoid function is defined as follows:

sigmoid(z) = 1 / (1 + exp(-z))

where z is the dot product of the input features and the model parameters. This value is then compared to a threshold to make the final prediction.

The algorithm then calculates the error (or cost) between the predicted output and the actual output for each training example. The cost function used in logistic regression is typically the negative log-likelihood of the data, which penalizes the model for making incorrect predictions.

The algorithm then updates the model parameters using gradient descent, which involves calculating the partial derivatives of the cost function with respect to each parameter and adjusting the parameters in the direction that decreases the cost. The learning rate determines the size of the parameter update at each iteration.

The algorithm repeats this process until it converges to a set of model parameters that minimize the cost on the training data. The resulting model can then be used to predict the output for new inputs.

LR1 -

LR = 0.01 Accuracy = 55.85

LR = 0.001 Accuracy = 59.67

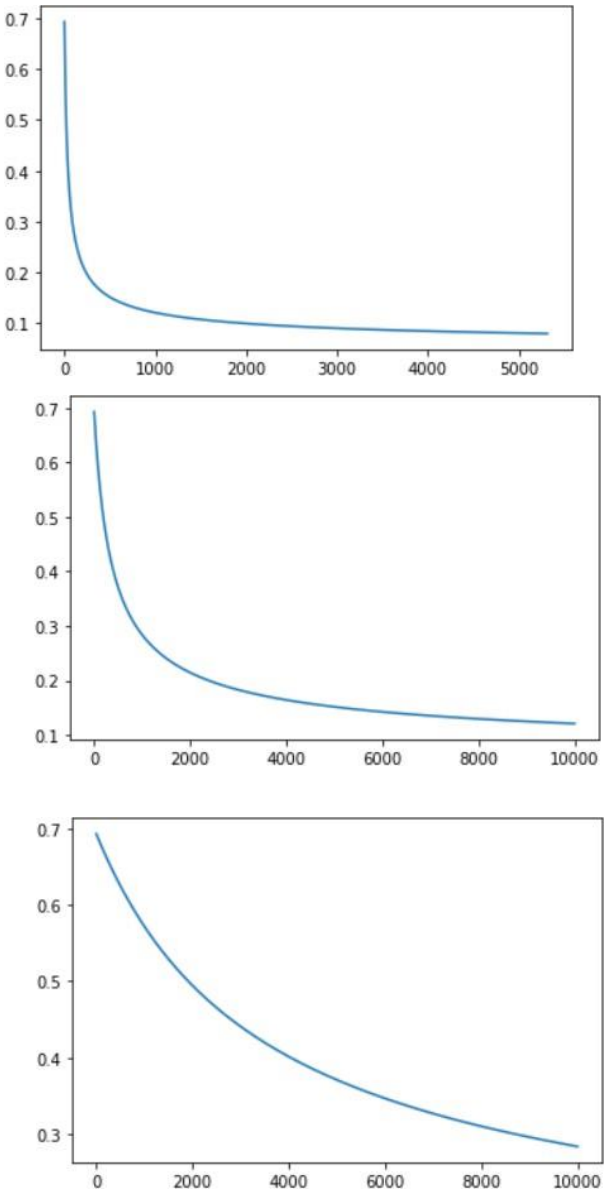
LR = 0.0001 Accuracy = 63.66

LR2 -

Batch Gradient Descent -

Threshold	**Accuracy (LR = 0.01)**	**Accuracy (LR = 0.001)**	**Accuracy (LR = 0.0001)**
0.3	96.81 %	95.74 %	84.04 %
0.4	95.74 %	96.81 %	95.21 %
0.5	96.81 %	98.94 %	97.87 %
0.6	97.87 %	97.87 %	96.81 %

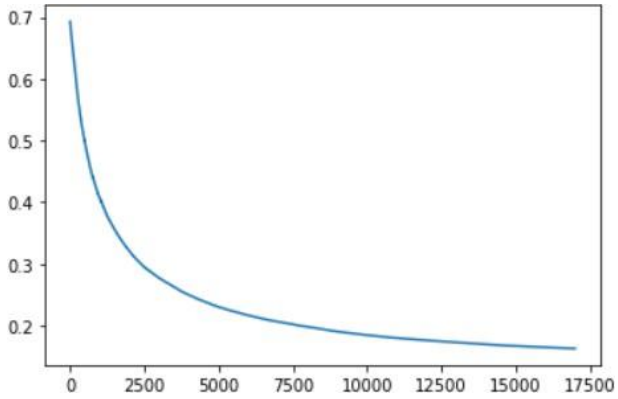
0.7	97.34 %	95.74 %	89.36 %

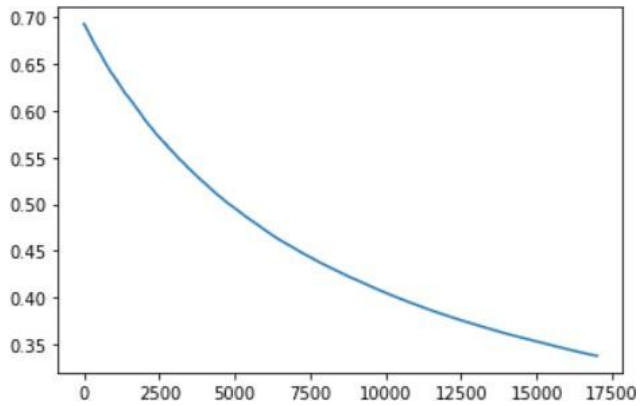


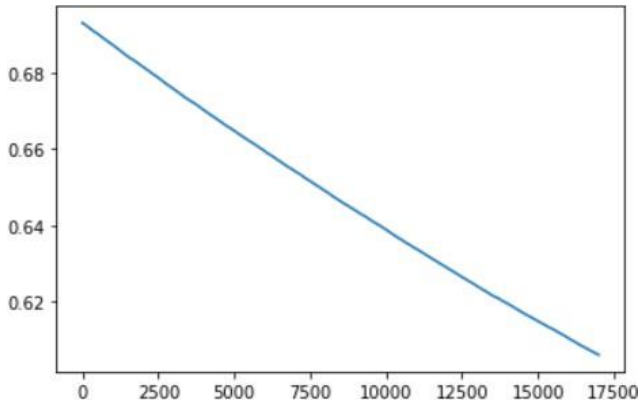
MiniBatch Gradient Descent -

Threshold	**Accuracy (LR = 0.01)**	**Accuracy (LR = 0.001)**	**Accuracy (LR = 0.0001)**
0.3	94.14	86.70	67.76
0.4	94.68	93.61	68.61

0.5	97.87	95.21	94.14
0.6	97.87	93.61	37.77
0.7	95.74	79.78	37.77

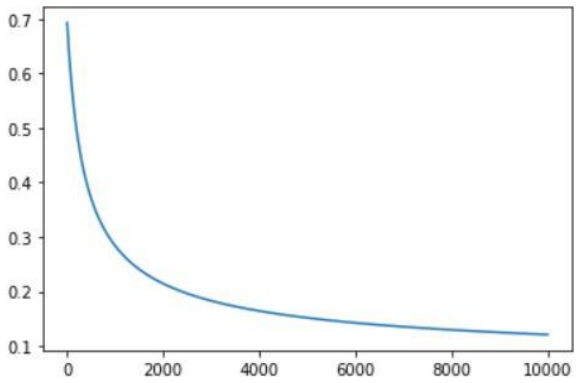


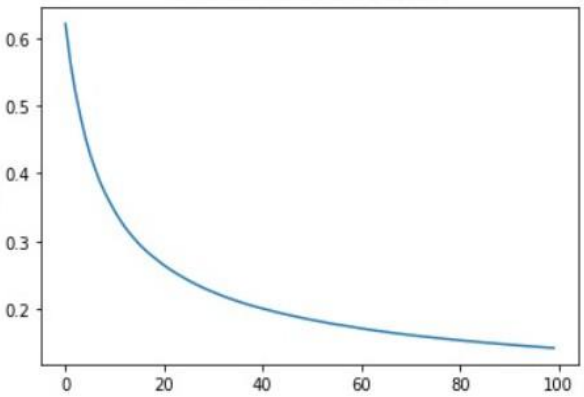


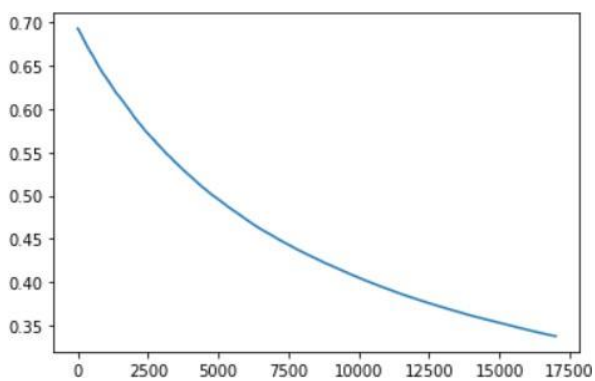


Stochastic Gradient Descent -

Threshold	**Accuracy (LR = 0.01)**	**Accuracy (LR = 0.001)**	**Accuracy (LR = 0.0001)**
0.3	22.99	63.64	76.46
0.4	22.99	70.59	86.1
0.5	22.99	73.26	90.91
0.6	22.99	77.54	93.05
0.7	23.1	82.35	95.72







PART D - COMPARATIVE STUDY

Its observed that LR2 model with batch gradient descent works best for the given data but however all the three models work equally well. In Fisher's Linear Discriminat we have obtained 96.8 % accuracy and we get the same result for FLDM1 and FLDM2 so we can say that order of the features doesnt matter in Fisher's Linear Discriminant Analysis.

In the PM1 model we split the data into 67% for the training data and the 33% testing data, and in the PM2 model we have changed the order in which the training data points are trained to the model. In the PM3 model we normalize the data and perform the perceptron model. We can observe that the accuracy of the model PM3 is maximum. In the model PM4 we have shuffled the feature columns, and then train the perceptron model. Here we observe that we get the same parameters as PM1 model because both the models are equivalent, and just the weights vector values get rearranged because the features are shuffled.

We can observe that as the number of epochs increase, we can get more accurate answer.

In LR1 we dropped features with missing values, the accuracy is 63.21 . LR1 doesn't give accurate results in absence of feature engineering tasks.

In LR2(after applying both feature engineering tasks) with Batch Gradient Descent, the average accuracy obtained is 97.81. The accuracy of Batch Gradient Descent is more than that of Stochastic and Mini Batch Gradient Descents. This is because Logistic regression can handle imbalanced datasets, where the number of samples in each class is not equal, by using class weights or adjusting the decision threshold. Perceptron and linear discriminant algorithms do not have built-in mechanisms for handling imbalanced datasets.