BITS F464

MACHINE LEARNING

ASSIGNMENT - 1

Visweswar Sirish Parupudi
2020AAPS0330H

Kedarnath Senegavaram
2020A7PS0245H

Konkala Rithvik
2020A8PS0517H

Feature Engineering Tasks

According to the problem statement, for some of the data points, one or more feature values were missing in the data set. We were asked to fill in these values so that the data tuple can be made use of in building the model as well as testing the model.

We initially performed some preprocessing by dropping the “id” column and changing the labels M and B for malignant and benign to 1 and 0.

In order to do any feature engineering at all, firstly we needed to know where the null values were present. We found this out using internal methods provided by pandas.

We then created a new .csv file where all the samples with null values were dropped. This .csv file is the one being used when none of the feature engineering tasks were required to be carried out.

For feature engineering task 1, instead of dropping null values, we imputed the missing values with the average of the existing values of the corresponding feature.

For feature engineering task 2, as required, we normalized the data by using the following formula:

$$X' = (X - \mu) / \sigma$$

where μ represents the mean of feature value, and σ represents the standard deviation of feature values).

Both feature engineering tasks were accomplished by internal pandas methods.

In each individual model implementation, we have divided the required dataset into 10 different train-test splits using an array called ‘randlist’ which contains the 10 seeds for our random splits.

Part A - Perceptron Learning Algorithm

LEARNING TASK 1:

PM1: We build a Perceptron model for classification and check whether the data is linearly separable or not. We can initially see from our Feature Engineering file that there are outliers in our dataset which may cause issues. Moving ahead, we build a Perceptron class on the unnormalized data for this task. We initialize a random list of integers for our train test split with a seed as a random state and this same dataset split will be used for all the upcoming models. The "Perceptron" class has the fit method which is used to train the data and the predict method is used to test the data. We use the Perceptron Model based on the following functions:

$$E_{\mathrm{P}}(\mathbf{w}) = -\sum_{n\in\mathcal{M}} \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}_n t_n$$

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta\nabla E_{\mathrm{P}}(\mathbf{w}) = \mathbf{w}^{(\tau)} + \eta\boldsymbol{\phi}_n t_n \qquad f(a) = \begin{cases} +1, & a \geqslant 0 \\ -1, & a < 0. \end{cases}$$

We finally compute accuracy, precision and recall, for all 10 train test splits, and then take their average and variance as shown below.

Mean values and variance of accuracy, precision and recall for all 10 data splits:

```
Mean Accuracy = 92.15053763440861 and Variance of Accuracy = 0.8209041507688714
Mean Precision = 0.9015402967554657 and Variance of Precision = 0.0034535361606881096
Mean Recall = 0.8972625133538885 and Variance of Recall = 0.0058208507192448345
```

Performance Metrics (Accuracy, Precision, Recall) for each individual data split:

```
Metrics for test-train split 0
91.93548387096774 0.8271604938271605 0.9852941176470589


Metrics for test-train split 1
93.01075268817203 0.9411764705882353 0.8767123287671232


Metrics for test-train split 2
91.93548387096774 0.9833333333333333 0.8082191780821918


Metrics for test-train split 3
90.86021505376344 0.9636363636363636 0.7794117647058824


Metrics for test-train split 4
92.47311827956989 0.8857142857142857 0.9117647058823529


Metrics for test-train split 5
93.01075268817203 0.8674698795180723 0.972972972972973


Metrics for test-train split 6
93.01075268817203 0.9787234042553191 0.7931034482758621


Metrics for test-train split 7
93.01075268817203 0.8666666666666667 0.9558823529411765


Metrics for test-train split 8
91.93548387096774 0.8904109589041096 0.9027777777777778


Metrics for test-train split 9
90.32258064516128 0.8111111111111111 0.9864864864864865
```

PM2: This is an extension of PM1 in which we shuffle the order of training examples. We see a change in accuracy metrics in this case. The reason for this change is due to the training order being different from before, leading to the formation of a slightly different hyperplane. The accuracy metrics are as shown below.

Mean values of performance metrics over 10 data splits:

```
Mean Accuracy = 90.86486486486487 and Variance of Accuracy = 22.582907231555915
Mean Precision = 0.9021753647581289 and Variance of Precision = 0.014531865556976952
Mean Recall = 0.8917423209472963 and Variance of Recall = 0.004951544095975183
```

Metrics for individual data splits(accuracy, precision, recall)

```
Metrics for test-train split 0
91.8918918918919 0.8452380952380952 0.9726027397260274


Metrics for test-train split 1
95.13513513513514 1.0 0.8732394366197183


Metrics for test-train split 2
92.43243243243244 1.0 0.8313253012048193


Metrics for test-train split 3
89.1891891891892 0.7590361445783133 1.0


Metrics for test-train split 4
77.29729729729729 0.6146788990825688 1.0


Metrics for test-train split 5
91.35135135135135 0.9827586206896551 0.7916666666666666


Metrics for test-train split 6
93.51351351351352 0.96875 0.8611111111111112


Metrics for test-train split 7
92.97297297297298 0.927536231884058 0.8888888888888888


Metrics for test-train split 8
92.97297297297298 0.9384615384615385 0.8714285714285714


Metrics for test-train split 9
91.8918918918919 0.9852941176470589 0.8271604938271605
```

LEARNING TASK 2:

PM3: In this model we apply the perceptron algorithm over the normalized data, which was done in feature engineering task 2. We now see that the accuracy has increased as expected for normalized data. The metrics are as shown below.

```
Mean Accuracy = 96.01063829787235 and Variance of Accuracy = 0.6931869624264365
Mean Precision = 0.9459606970260197 and Variance of Precision = 0.0005751920004782246
Mean Recall = 0.9473656711652139 and Variance of Recall = 0.00029830547292258647
```

```
Metrics for test-train split 0
96.27659574468085 0.9583333333333334 0.9452054794520548


Metrics for test-train split 1
96.27659574468085 0.9333333333333333 0.9722222222222222


Metrics for test-train split 2
96.80851063829788 0.9726027397260274 0.9466666666666667


Metrics for test-train split 3
95.74468085106383 0.9558823529411765 0.9285714285714286


Metrics for test-train split 4
95.2127659574468 0.927536231884058 0.9411764705882353


Metrics for test-train split 5
95.74468085106383 0.9333333333333333 0.958904109589041


Metrics for test-train split 6
94.68085106382979 0.8923076923076924 0.9508196721311475


Metrics for test-train split 7
95.74468085106383 0.9444444444444444 0.9444444444444444


Metrics for test-train split 8
95.74468085106383 0.9692307692307692 0.9130434782608695


Metrics for test-train split 9
97.87234042553192 0.9726027397260274 0.9726027397260274
```

LEARNING TASK 3:

PM4: We now shuffle the features of the normalized data, but there is no change in any metrics. This is because the change in the order of the feature vector would not have an effect on the predictions. Metrics are as displayed below.

```
Mean Accuracy = 96.01063829787235 and Variance of Accuracy = 0.6931869624264365
Mean Precision = 0.9459606970260197 and Variance of Precision = 0.0005751920004782246
Mean Recall = 0.9473656711652139 and Variance of Recall = 0.00029830547292258647
```

```
Metrics for test-train split 0
96.27659574468085 0.9583333333333334 0.9452054794520548

Metrics for test-train split 1
96.27659574468085 0.9333333333333333 0.9722222222222222

Metrics for test-train split 2
96.80851063829788 0.9726027397260274 0.9466666666666667

Metrics for test-train split 3
95.74468085106383 0.9558823529411765 0.9285714285714286

Metrics for test-train split 4
95.2127659574468 0.927536231884058 0.9411764705882353

Metrics for test-train split 5
95.74468085106383 0.9333333333333333 0.958904109589041

Metrics for test-train split 6
94.68085106382979 0.8923076923076924 0.9508196721311475

Metrics for test-train split 7
95.74468085106383 0.9444444444444444 0.9444444444444444

Metrics for test-train split 8
95.74468085106383 0.9692307692307692 0.9130434782608695

Metrics for test-train split 9
97.87234042553192 0.9726027397260274 0.9726027397260274
```

Part B - Fisher's Linear Discriminant Analysis

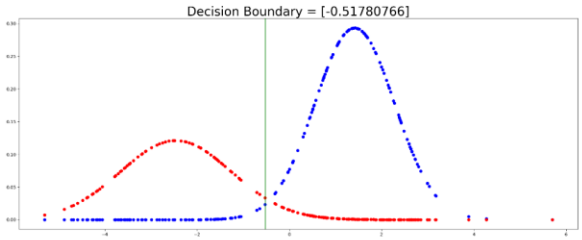
LEARNING TASK 1:

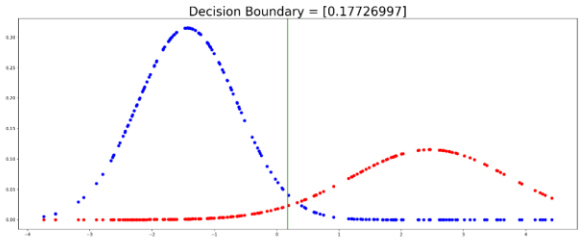
FLDM1: We build a Fisher's Linear discriminant model to bring down the 32 dimensional features to 1 dimension by projecting it onto the hyperplane. We have used scikit learn only for FLDA projection to 1 dimension as instructed. We then find the decision boundary using the generative approach. We calculate the posterior probability using prior and likelihood assuming gaussian distribution for the data as instructed. We then calculate posterior probability using the below formula, for both classes and for a given sample, and compare the probabilities to evaluate which class it belongs to.

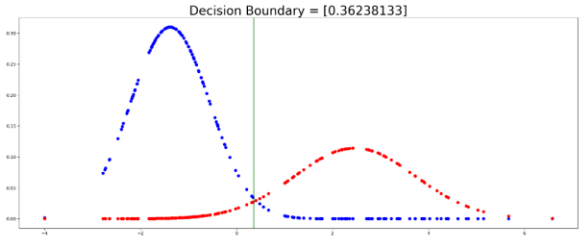
$$p(\theta|x) = \frac{p(x|\theta)}{p(x)}p(\theta)$$

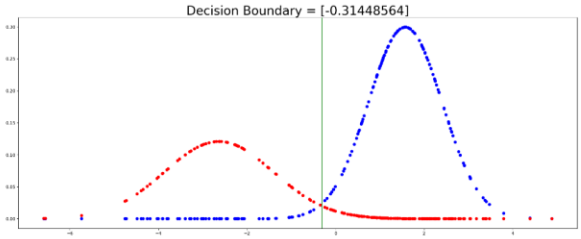
Now we can find the decision boundary, at the point where both the posterior probabilities of both the classes intersect. We do this for 10 train-test splits and plot the graphs as shown. The metrics of accuracy, precision and recall are also given below.

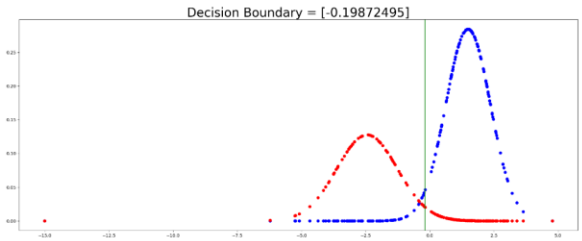
Training Features and their respective labels, along with the decision boundary for 10 data splits. Label 0 in blue and Label 1 in red

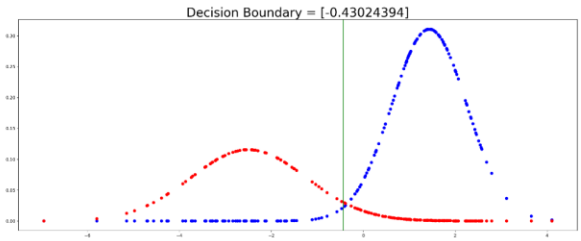


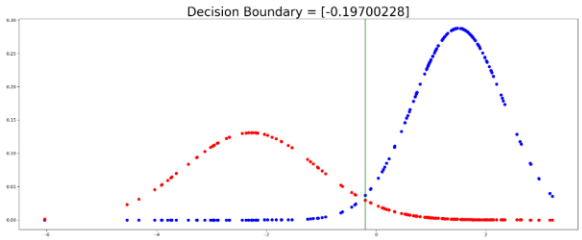


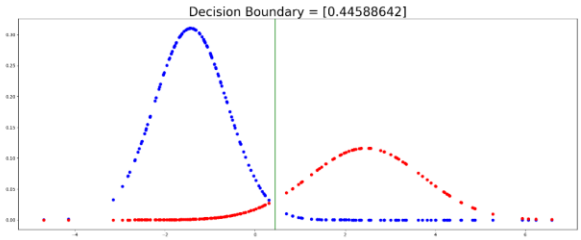


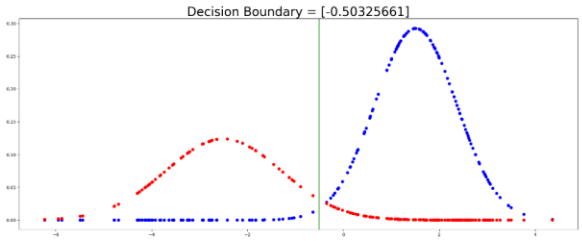


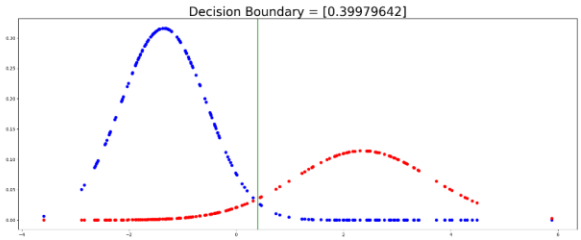












Mean values for metrics over 10 splits:

```
Mean Accuracy = 96.11702127659575 and Variance of Accuracy = 0.6252829334540574
Mean Precision = 0.9747566774335809 and Variance of Precision = 0.0002851207985647865
Mean Recall = 0.9206697558925411 and Variance of Recall = 0.00036039918091811064
```

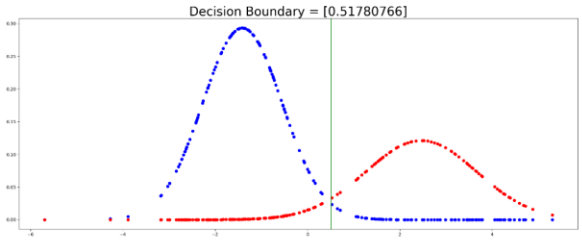
Individual values for each split:

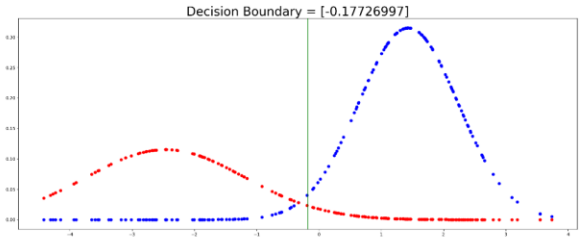
```
Metrics for test-train split 0
97.3404255319149 0.9857142857142858 0.9452054794520548


Metrics for test-train split 1
95.74468085106383 0.9444444444444444 0.9444444444444444


Metrics for test-train split 2
95.74468085106383 0.971830985915493 0.92


Metrics for test-train split 3
95.2127659574468 0.9692307692307692 0.9


Metrics for test-train split 4
96.27659574468085 0.9692307692307692 0.9264705882352942


Metrics for test-train split 5
95.2127659574468 0.9571428571428572 0.9178082191780822


Metrics for test-train split 6
96.80851063829788 0.9661016949152542 0.9344262295081968


Metrics for test-train split 7
96.27659574468085 1.0 0.9027777777777778


Metrics for test-train split 8
95.2127659574468 0.9838709677419355 0.8840579710144928


Metrics for test-train split 9
97.3404255319149 1.0 0.9315068493150684
```

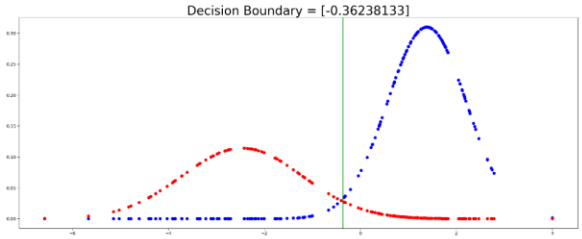
LEARNING TASK 1:

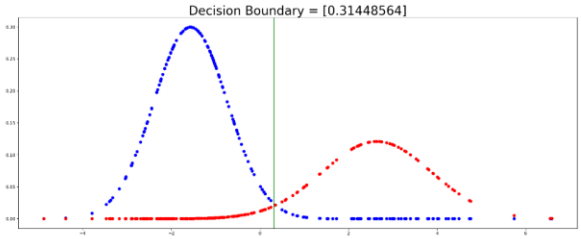
FLDM2: Here we shuffle the order of feature vectors before projecting it onto 1-Dimension and as we can see it does not affect the accuracy metrics of the model as shown below. This is because regardless of a shuffle in the order, the same 32 features are being brought down to 1 feature by the scikit learn LDA model.

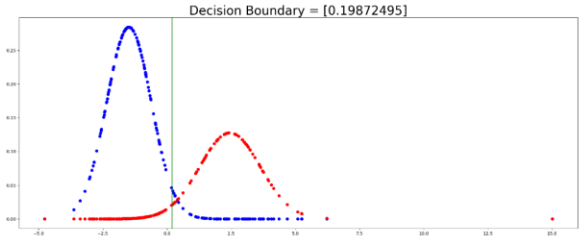
Training Features v Labels, with y=0 in blue and y=1 in red:

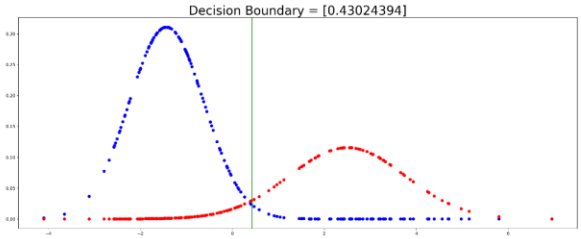


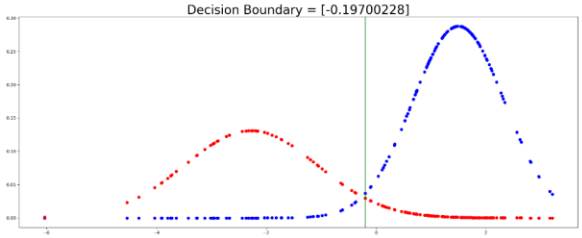


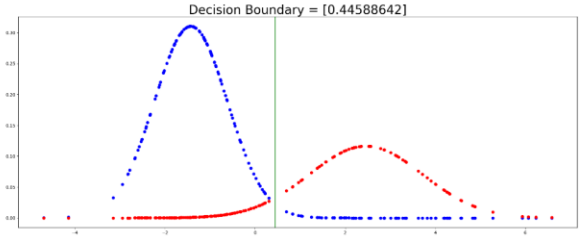


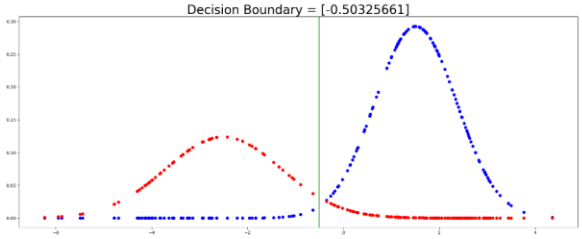


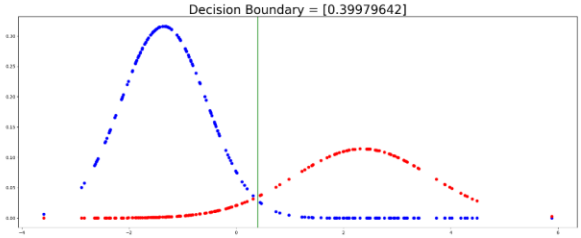












Mean and variance of performance metrics over 10 splits:

```
Mean Accuracy = 96.11702127659575 and Variance of Accuracy = 0.6252829334540574
Mean Precision = 0.9747566774335809 and Variance of Precision = 0.0002851207985647865
Mean Recall = 0.9206697558925411 and Variance of Recall = 0.00036039918091811064
```

Performance metrics for each split:

```
Metrics for test-train split 0
97.3404255319149 0.9857142857142858 0.9452054794520548


Metrics for test-train split 1
95.74468085106383 0.9444444444444444 0.9444444444444444


Metrics for test-train split 2
95.74468085106383 0.971830985915493 0.92


Metrics for test-train split 3
95.2127659574468 0.9692307692307692 0.9


Metrics for test-train split 4
96.27659574468085 0.9692307692307692 0.9264705882352942


Metrics for test-train split 5
95.2127659574468 0.9571428571428572 0.9178082191780822


Metrics for test-train split 6
96.80851063829788 0.9661016949152542 0.9344262295081968


Metrics for test-train split 7
96.27659574468085 1.0 0.9027777777777778


Metrics for test-train split 8
95.2127659574468 0.9838709677419355 0.8840579710144928


Metrics for test-train split 9
97.3404255319149 1.0 0.9315068493150684
```

Part C - Logistic Regression

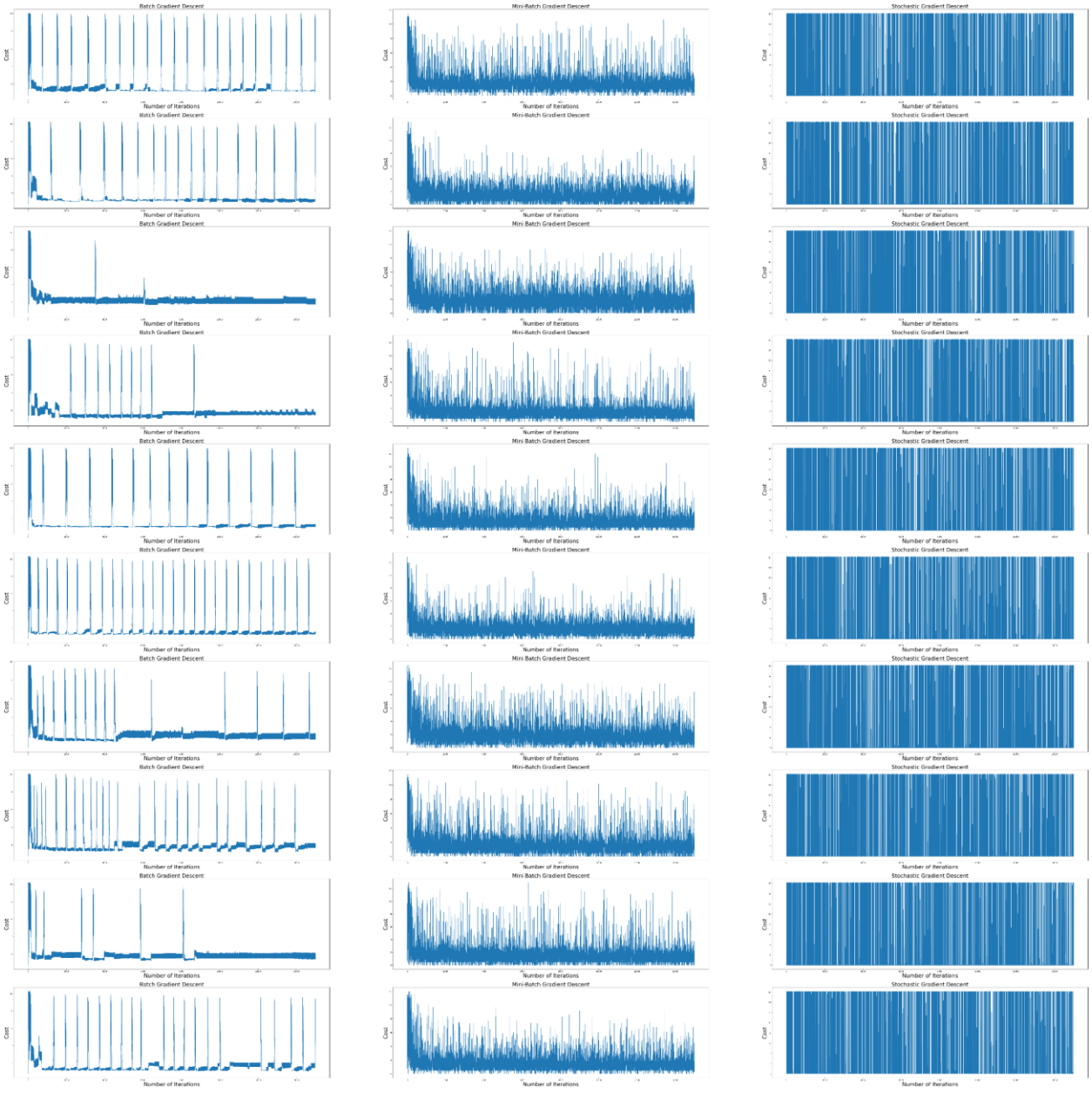
LEARNING TASK 1:

LR1: We implement a Logistic Regression Model for classification using only dropped values and no feature engineering. We implement Batch Gradient Descent, Mini Batch Gradient Descent, and Stochastic Gradient Descent for threshold values of 0.3,0.4,0.5,0.6 and 0.7 with learning rates 0.01,0.001,0.0001 respectively. We also plot the cost function vs iterations graphs for all 3 models for each of the 10 train-test splits as shown below. This is done using the Logistic Regression class which has methods for BGD,MBGD and SGD. The cost function kept dropping to NaN as the data was not normalized and can also be due to the value of the learning rate, thus we have padded the cost function to prevent NaN values.

$$J(\theta) = -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

$$\theta_j := \theta_j - \alpha \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

Cost function vs number of iterations for all 3 models (BGD, MBGD, SGD) for each of the 10 train-test splits:



The graphs don't decrease continuously because the data is not normalized, and some problems are being caused by the bias term being added to ensure we don’t get any NaN values for log-likelihood.

Average performance metrics for threshold value 0.5

```
Average accuracy of Batch Gradient Descent model with threshold value 0.5 = 92.1146953405018
Average accuracy of Mini-Batch Gradient Descent model with threshold value 0.5 = 91.57706093189965
Average accuracy of Stochastic Gradient Descent model with threshold value 0.5 = 85.12544802867383


Average precision of Batch Gradient Descent model with threshold value 0.5 = 0.9228777501037775
Average precision of Mini-Batch Gradient Descent model with threshold value 0.5 = 0.9505498495694574
Average precision of Stochastic Gradient Descent model with threshold value 0.5 = 0.750043873729267

Average recall of Batch Gradient Descent model with threshold value 0.5 = 0.8627450980392156
Average recall of Mini-Batch Gradient Descent model with threshold value 0.5 = 0.8310502283105023
Average recall of Stochastic Gradient Descent model with threshold value 0.5 = 0.9908675799086758
```

Variance in metrics:

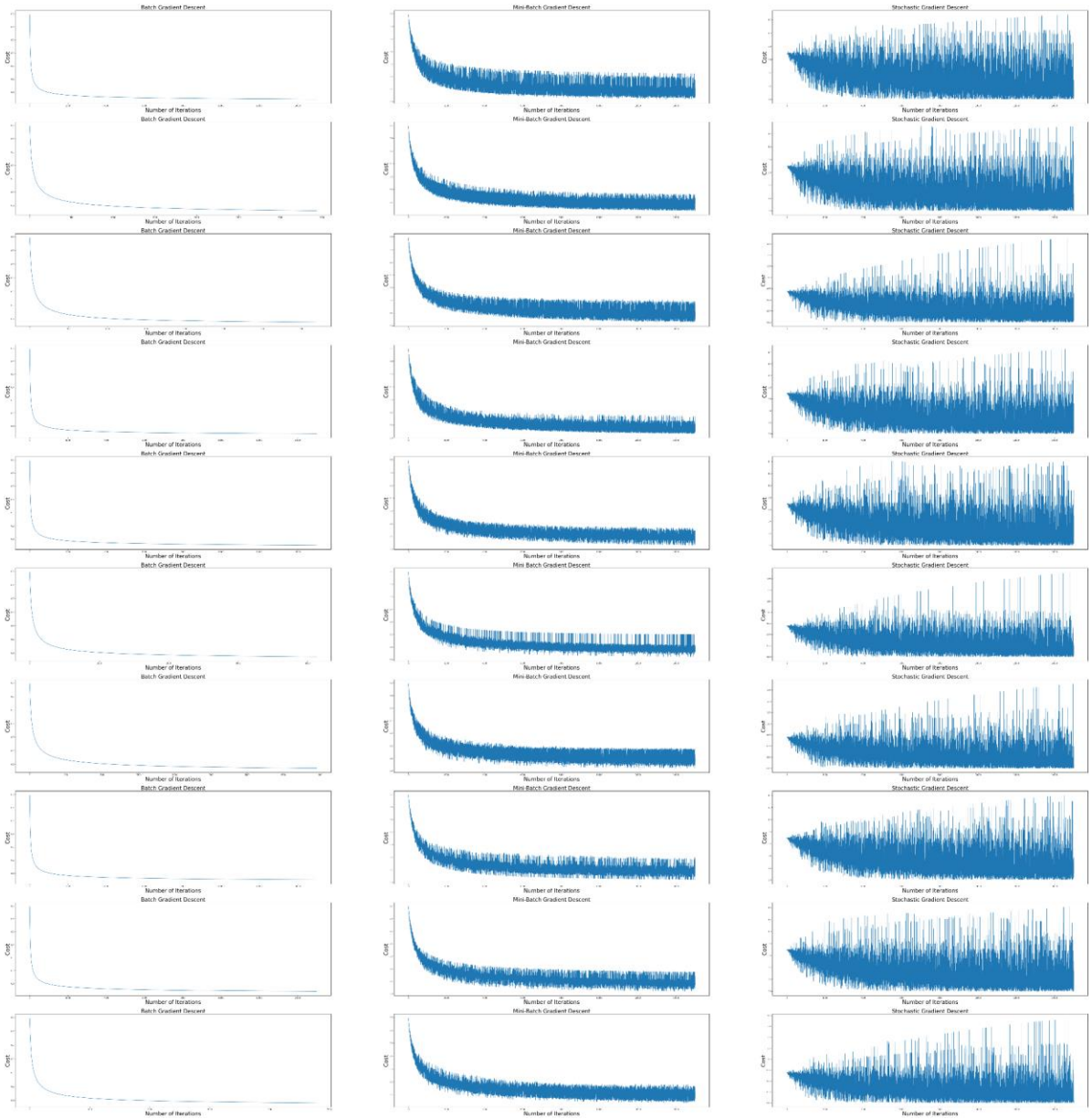
```
Variance in accuracy of Batch Gradient Descent model with threshold value 0.5 = 1.2204365308770433
Variance in accuracy of Mini-Batch Gradient Descent model with threshold value 0.5 = 4.881746123508201
Variance in accuracy of Stochastic Gradient Descent model with threshold value 0.5 = 86.77946069552024


Variance in precision of Batch Gradient Descent model with threshold value 0.5 = 0.0030330842694847988
Variance in precision of Mini-Batch Gradient Descent model with threshold value 0.5 = 0.0007667629842759895
Variance in precision of Stochastic Gradient Descent model with threshold value 0.5 = 0.01417175344852413


Variance in recall of Batch Gradient Descent model with threshold value 0.5 = 0.002499038831218763
Variance in recall of Mini-Batch Gradient Descent model with threshold value 0.5 = 0.005546172932174056
Variance in recall of Stochastic Gradient Descent model with threshold value 0.5 = 4.170054836221129e-05
```

LEARNING TASK 2:

LR2: We implement the same models as LR1, namely Batch Gradient Descent, Mini-Batch Gradient Descent and Stochastic Gradient Descent but with the normalized dataset. We observe that the performance metrics as well as the cost vs iterations curves are both significantly better. This is because of the test data fitting better with the log-likelihood term.

We also observe that the threshold value of 0.5 gives best results among the other thresholds and we thus use that in our comparative study ahead.

Cost vs n_iters curves for all 3 models over 10 splits:



Average performance metrics over 10 splits:

```
Average accuracy of Batch Gradient Descent model with threshold value 0.5 = 97.3404255319149
Average accuracy of Mini-Batch Gradient Descent model with threshold value 0.5 = 96.9858156028369
Average accuracy of Stochastic Gradient Descent model with threshold value 0.5 = 97.16312056737588


Average precision of Batch Gradient Descent model with threshold value 0.5 = 0.9811535881958418
Average precision of Mini-Batch Gradient Descent model with threshold value 0.5 = 0.9550352799055543
Average precision of Stochastic Gradient Descent model with threshold value 0.5 = 0.9604093567251462


Average recall of Batch Gradient Descent model with threshold value 0.5 = 0.9497716894977168
Average recall of Mini-Batch Gradient Descent model with threshold value 0.5 = 0.9675925925925926
Average recall of Stochastic Gradient Descent model with threshold value 0.5 = 0.9688888888888889
```

Variance in performance metrics over 10 splits:

```
Variance in accuracy of Batch Gradient Descent model with threshold value 0.5 = 0.18862230270107105
Variance in accuracy of Mini-Batch Gradient Descent model with threshold value 0.5 = 0.8173633117046429
Variance in accuracy of Stochastic Gradient Descent model with threshold value 0.5 = 1.949097127911071


Variance in precision of Batch Gradient Descent model with threshold value 0.5 = 4.34622040312521e-05
Variance in precision of Mini-Batch Gradient Descent model with threshold value 0.5 = 0.0005774357046195344
Variance in precision of Stochastic Gradient Descent model with threshold value 0.5 = 0.0003432098765432103


Variance in recall of Batch Gradient Descent model with threshold value 0.5 = 4.1700548362210624e-05
Variance in recall of Mini-Batch Gradient Descent model with threshold value 0.5 = 4.286694101508886e-05
Variance in recall of Stochastic Gradient Descent model with threshold value 0.5 = 0.0002765432098765438
```

Part D - Comparative Study

LEARNING TASK 1:

We perform a comparative study on all the models so far, using their average metrics of 10 train-test splits and evaluate their performance as shown below. We have rounded each metric to 3 decimals. Very small values have been rounded to 0. As mentioned above in the Logistic Regression section (pg16, LR2) we are considering threshold value 0.5 for Logistic Regression models LR1 and LR2 as they provided the best average metrics.

An LR2 implementation of batch gradient descent, with threshold 0.5 proves to be a very comprehensive model, with the highest average accuracy and average precision and the lowest variance in both categories.

An LR2 implementation of stochastic gradient descent with the threshold set to 0..5 gives the highest average recall.

A broad trend is that non-normalized feature values harm the model to quite an extent.

Model		Average Accuracy	Variance Accuracy	Average Precision	Variance precision	Average Recall	Variance Recall
PM1		0.9205	0.0082	0.9015	0.0034	0.8972	0.0058
PM2		0.9086	0.2258	0.9021	0.0145	0.8917	0.0049
PM3		0.9601	0.0069	0.9459	0.0005	0.9473	0.0003
PM4		0.9601	0.0069	0.9459	0.0005	0.9473	0.0003
FLDM1		0.9611	0.0062	0.9747	0.0003	0.9206	0.0003
FLDM2		0.9611	0.0062	0.9747	0.0003	0.9206	0.0003
LR1, Threshold = 0.5	BGD	0.9211	0.0122	0.9228	0.003	0.8627	0.0025
	MBGD	0.9158	0.0488	0.9505	0.0007	0.831	0.0055
	SGD	0.8515	0.8677	0.75	0.1417	0.9908	0
LR2, Threshold = 0.5	BGD	**0.9734**	0.0019	**0.9811**	0	0.9497	0
	MBGD	0.9698	0.0081	0.955	0.0005	0.9676	0
	SGD	0.9716	0.0194	0.9604	0.0003	**0.9689**	0.0002

x----------x