PM1: (avg. accuracy = 88.9%)

In PM1 we implement perceptron algorithm without normalising the data and dropping rows which possess NaN values. We have final results as shown below.

In perceptron, we find the coefficient values of a hyperplane which is our decision boundary. We do this by iterating through our data many times, and adding or subtracting a particular row based on whether our existing hyperplane (which was initialised to be 0 in the beginning) correctly or incorrec

```
PM1
AVERAGE ACCURACY 0.8887096774193548
AVERAGE PRECISION 0.9252874912501537
AVERAGE RECALL 0.8969063267086946
```

```
PM1
ACCURACY VARIANCE:  0.05156808089583569
PRECISION VARIANCE:  0.05388796638889157
RECALL VARIANCE:  0.09983908130766614
```

PM3: (avg. accuracy = 95.2%)

In PM3 we implement perceptron algorithm after normalising the data and we take mean of the respective features and replace NaNs. We have final results as shown below.

```
PM3
AVERAGE ACCURACY 0.9521276595744681
AVERAGE PRECISION 0.9556678661004643
AVERAGE RECALL 0.9674085171501403
```

```
PM3
ACCURACY VARIANCE:  0.017809222385939592
PRECISION VARIANCE:  0.007935132969455616
RECALL VARIANCE:  0.02924692172947626
```

PM4: (avg. accuracy = 95.2%)

In PM4 the only difference with respect to PM3 is that we shuffle columns of the dataset which clearly doesn't affect any of the measured parameters for perceptron. We have final results as shown below.

```
PM4
ACCURACY VARIANCE:  0.017809222385939592
PRECISION VARIANCE:  0.007935132969455616
RECALL VARIANCE:  0.02924692172947626
```

```
PM4
AVERAGE ACCURACY 0.9521276595744681
AVERAGE PRECISION 0.9556678661004643
AVERAGE RECALL 0.9674085171501403
```

FLDM1: (avg. accuracy = 96.5%)

In FDLM1 we apply fisher's linear discriminant analysis on the training data assuming gaussian distribution for both positive and negative classes in the univariate dimension. We have final results as shown below. Here we project and reduce the higher dimensional 30d vector into a univariate one by projecting it onto a vector W which maximises the difference of means and minimises the sum of variance of the two classes.

```
AVERAGE CONFUSION MATRIX [[234.    2.6]
 [ 10.7 128.7]]
AVERGAE ACCURACY: 0.9646276595744682
AVERAGE PRECISION: 0.989010989010989
AVERAGE RECALL 0.9562729873314263
```

```
ACCURACY VARIANCE:  0.011844727963168168
PRECISION VARIANCE:  0.010572777282673413
RECALL VARIANCE:  0.030184522628029408
```

FLDM2: (avg. accuracy = 96.5%)

We get the same results as in FLDM1 here as the only change we make is column shifting which doesn't change any of the things we are measuring. We have final results as shown below.

```
AVERAGE CONFUSION MATRIX [[234.    2.6]
 [ 10.7 128.7]]
AVERGAE ACCURACY: 0.9646276595744682
AVERAGE PRECISION: 0.989010989010989
AVERAGE RECALL 0.9562729873314263

ACCURACY VARIANCE:  0.011844727963168168
PRECISION VARIANCE:  0.010572777282673413
RECALL VARIANCE:  0.030184522628029408
```

LR1:

In LR1 we are required to implement logistic regression under various parameters as specified below without normalisation.

For sake of comparison here we are using the standard probability threshold of 0.5 and learning rate of 0.01, this case along with all the other cases are covered in code submitted. We are required to compare for Batch Gradient Descent, Mini-batch and Stochastic Gradient Descent.

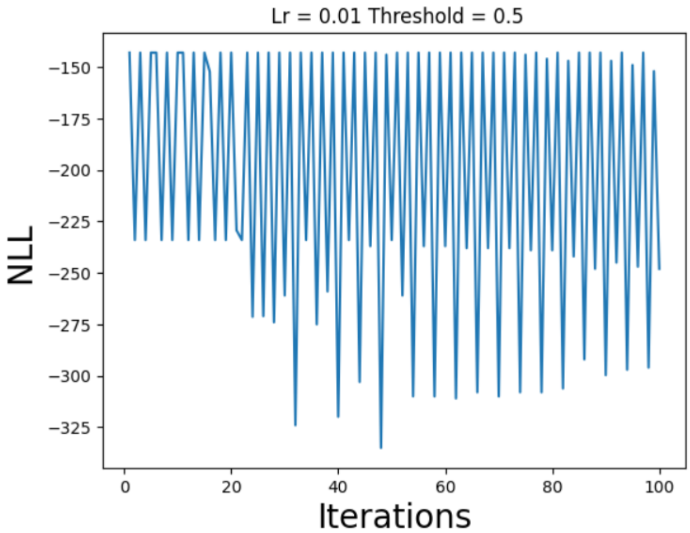
In LR, we model the distribution of our y, using the sigmoid function, which has w as a parameter. Through gradient descent, we find the optimal w that minimises our negative log likelihood function.

Batch Gradient Descent (avg. accuracy = 57.6%)

We take the average of the gradients of all the training examples and then use that mean gradient to update our parameters. So we just take one step of gradient descent per epoch. The results are as shown below.

LR1
Accuracy
Average = 0.5763440860215054 Standard Deviation = 0.17821262865050122
Precision
Average = 0.7458907294391166 Standard Deviation = 0.31128216099489203
Recall
Average = 0.5396602967368972 Standard Deviation = 0.39119858661610574

```
ITERATION : 1
```

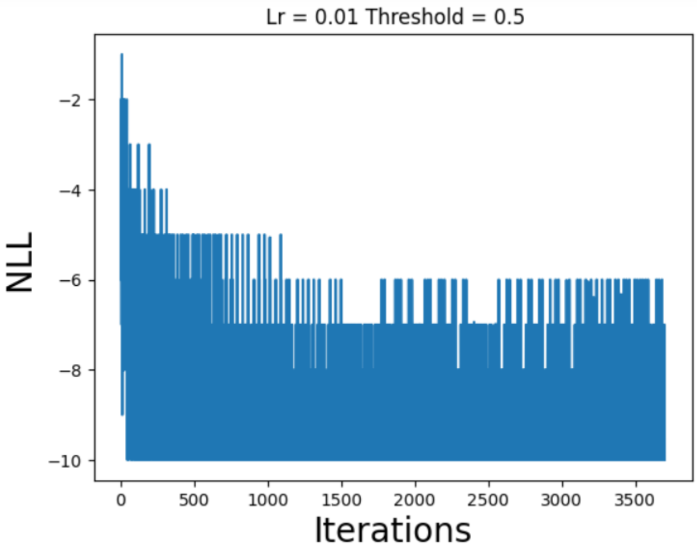



```
CONFUSION MATRIX
[[118.   0.]
 [ 58.  10.]]
ACCURACY = 0.6881720430107527
PRECISION = 1.0
RECALL = 0.14705882352941177
```

Mini-Batch Gradient Descent (avg. accuracy = 86.2%)

A mini-batch is a subset of the training data used in each iteration of the training algorithm in mini-batch gradient descent. The results are as shown below.

LR1
Accuracy
Average = 0.8623655913978494 Standard Deviation = 0.04473686895207466
Precision
Average = 0.7908524555507701 Standard Deviation = 0.12324952986857828
Recall
Average = 0.9003454932901848 Standard Deviation = 0.072502037008662



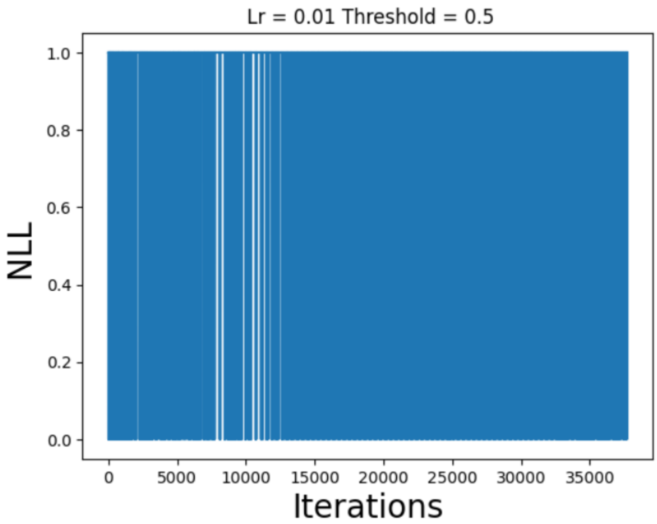
CONFUSION MATRIX

[[89. 29.]
 [2. 66.]]
ACCURACY = 0.8333333333333334
PRECISION = 0.6947368421052632
RECALL = 0.9705882352941176

Stochastic Gradient Descent (avg. accuracy = 83.4%)

Sequentially modifies the parameters of each training sample in each training sample of the dataset. The results are as shown below.

LR1
Accuracy
Average = 0.8349462365591398 Standard Deviation = 0.06460790962689322
Precision
Average = 0.7774850747464171 Standard Deviation = 0.16188651462324855
Recall
Average = 0.8794254190447062 Standard Deviation = 0.13402404380875763




```
CONFUSION MATRIX

[[112.   6.]
 [ 12.  56.]]
ACCURACY = 0.9032258064516129
PRECISION = 0.9032258064516129
RECALL = 0.8235294117647058
```

LR2:

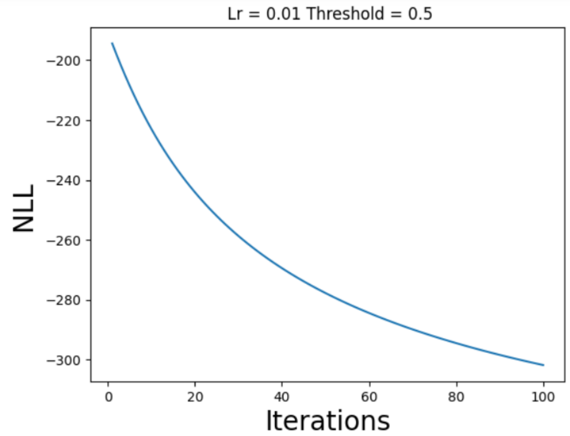
In LR2 we are required to implement logistic regression under various parameters as specified below with normalisation.

For sake of comparison here we are using the standard probability threshold of 0.5 and learning rate of 0.01, this case along with all the other cases are covered in code submitted. We are required to compare for Batch Gradient Descent, Mini-batch and Stochastic Gradient Descent.

Batch Gradient Descent (avg. accuracy = 95.9%)

We take the average of the gradients of all the training examples and then use that mean gradient to update our parameters. So we just take one step of gradient descent per epoch. The results are as shown below.

LR2
Accuracy
Average = 0.9590425531914895 Standard Deviation = 0.012372024840013833
Precision
Average = 0.9487424772768062 Standard Deviation = 0.013953432487262272
Recall
Average = 0.9386084551602998 Standard Deviation = 0.026615303607714334

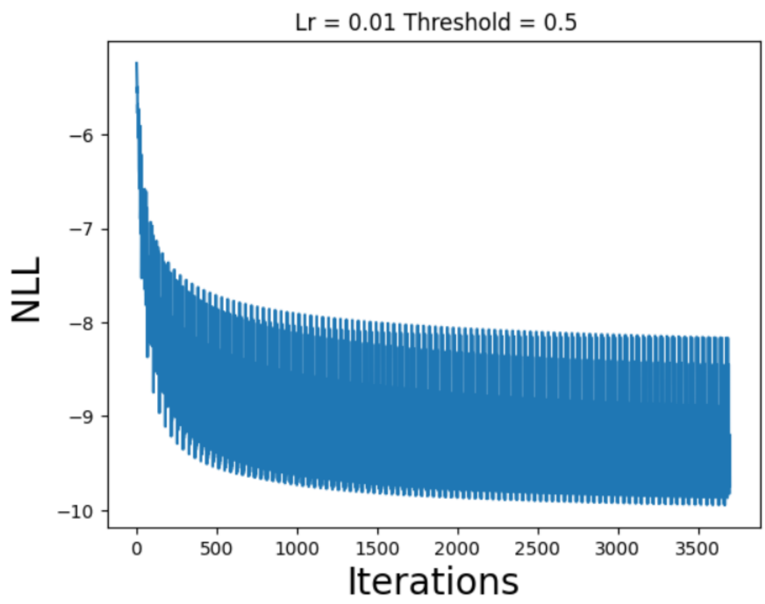



```
CONFUSION MATRIX
[[116.   4.]
 [  3.  65.]]
ACCURACY = 0.9627659574468085
PRECISION = 0.9420289855072463
RECALL = 0.9558823529411765
```

Mini-Batch Gradient Descent (avg. accuracy =96.8%)

A mini-batch is a subset of the training data used in each iteration of the training algorithm in mini-batch gradient descent. The results are as shown below.

LR2
Accuracy
Average = 0.9686170212765959 Standard Deviation = 0.008393475445776323
Precision
Average = 0.992274523539737 Standard Deviation = 0.007780490700404329
Recall
Average = 0.9201412900703569 Standard Deviation = 0.02607976198504841



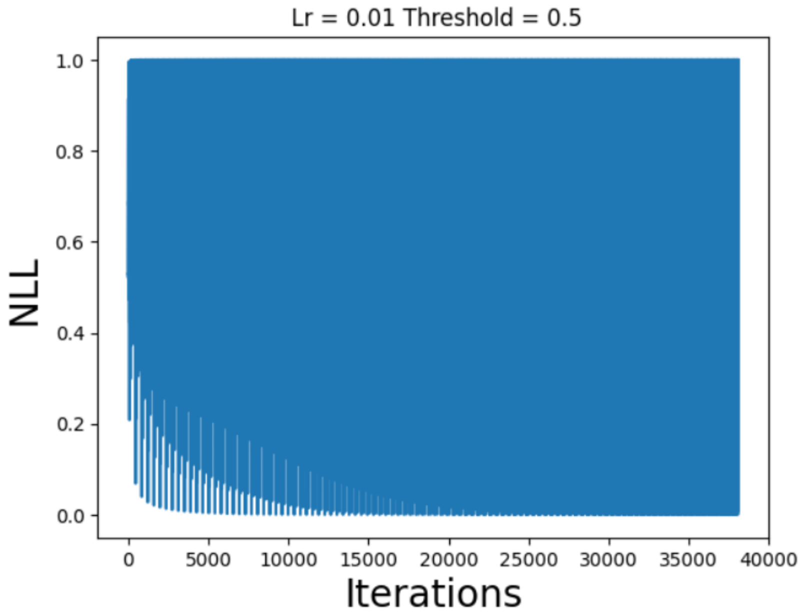

```
CONFUSION MATRIX

[[120.   0.]
 [  6.  62.]]
ACCURACY = 0.9680851063829787
PRECISION = 1.0
RECALL = 0.9117647058823529
```

Stochastic Gradient Descent (avg. accuracy = 93.9%)

Sequentially modifies the parameters of each training sample in each training sample of the dataset The results are as shown below.

LR2
Accuracy
Average = 0.9393617021276597 Standard Deviation = 0.012406280627330424
Precision
Average = 1.0 Standard Deviation = 0.0
Recall
Average = 0.8325692731556942 Standard Deviation = 0.039517369058554366




```
CONFUSION MATRIX

[[120.   0.]
 [ 11.  57.]]
ACCURACY = 0.9414893617021277
PRECISION = 1.0
RECALL = 0.8382352941176471
```

From the above observations it is clear that LR2 Mini-Batch Gradient Descent is the best performing model with the highest average accuracy at over 96.8%. This is intuitive considering both the feature engineering tasks are performed on data in this model therefore increasing accuracy and also due to the property of how Mini-Batch Gradient Descent is calculated it has a more stable convergence towards global minima since average gradient is calculated over n samples resulting in lower noise (Standard Deviation is the least out of all examples).