ML - Assignment 2

-Harkishan Singh(2017233)

Question 1:

a) For this part, I have implemented the Regression class:

- ➔ First I have changed the data file to .csv file and then I have changed the first column of data (since it is character type) in the following manner :

  M → -1, F → 1 and I → 0.
- ➔ For fit function : LinearRegression from Sklearn is used.
- ➔ For predict function : It is implemented using output of the fit function (which is the model). fit() gives us 8 coefficients (W) and 1 intercept (c). Equation used for predicting y is: **y = X.W + c** (X is the instance).

b) Following are the results asked by the question. These results are stored in "Q2_b.csv":

Q1_b

Fold Number	Train Error	Validation Error	Sklearn Train Error	Sklearn Validation Error
0	3.8382412497410083	10.088946266779553	3.8382412497410083	10.088946266779553
1	5.406303714549687	3.1521805786831023	5.406303714549687	3.1521805786831023
2	4.711081942571674	5.9629761841117945	4.711081942571674	5.9629761841117945
3	5.191065445123249	3.8434382309444324	5.191065445123249	3.8434382309444324
4	5.115016338415645	4.134503154592122	5.115016338415645	4.134503154592122

It can be seen from the above table that Sklearn MSE is exactly the same as MSE function built by me and so are the mean of these errors.
Here is the mean of these errors:

```
Mean of train MSE =  4.852341738080253
Mean of train MSE of Sklearn =  4.852341738080253
Mean of test/validation MSE =  5.436408883022201
Mean of test/validation MSE of Sklearn =  5.436408883022201
```

c) For this part, I have made the function “normal_equation_train” in file “Q1_c.py” which is basically the train function and it returns the coefficient of linear regression. Its results are exactly (stored in file “Q3_c.csv”) the same as the results I received in part (b).
Equation used for training (normal equation):

$$\textbf{Coefficients} = (X^T.X)^{-1}.(X^T.y)$$

X → Train data, y → labels

Here are the screenshots:

Q1_c

Fold Number	Train Error	Validation Error	Sklearn Train Error	Sklearn Validation Error
0	3.8382412497410083	10.088946266781068	3.8382412497410083	10.088946266781068
1	5.406303714549687	3.1521805786837174	5.406303714549687	3.1521805786837174
2	4.711081942571674	5.962976184111094	4.711081942571674	5.962976184111094
3	5.191065445123249	3.8434382309456283	5.191065445123249	3.8434382309456283
4	5.115016338415645	4.1345031545932605	5.115016338415645	4.1345031545932605

Mean of these MSEs:

```
Mean of train MSE =  4.852341738080253
Mean of train MSE of Sklearn =  4.852341738080253
Mean of test/validation MSE =  5.436408883022954
Mean of test/validation MSE of Sklearn =  5.436408883022954
```

Next page ⇒

d) LinearRegression from sklearn is used completely for this part (see file “Q1_d.py”) and below are the results:

Q1_d

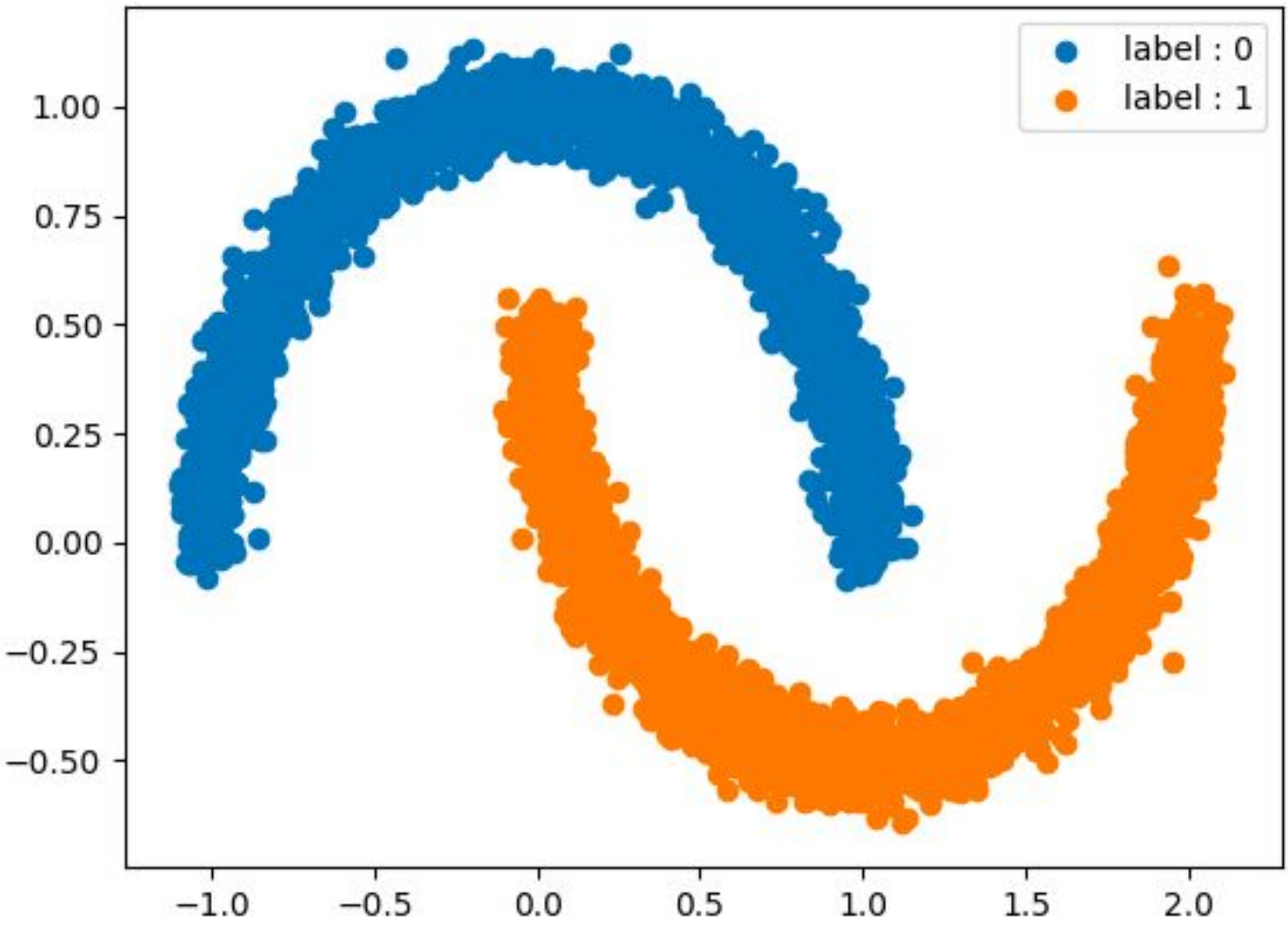
Fold Number	Train Error	Validation Error	Sklearn Train Error	Sklearn Validation Error
0	3.8382412497410088	10.088946266779551	3.8382412497410088	10.088946266779551
1	5.406303714549687	3.1521805786831023	5.406303714549687	3.1521805786831023
2	4.711081942571674	5.9629761841117945	4.711081942571674	5.9629761841117945
3	5.191065445123249	3.8434382309444324	5.191065445123249	3.8434382309444324
4	5.115016338415645	4.134503154592122	5.115016338415645	4.134503154592122

Following are the mean of these MSEs:

```
Mean of train MSE =  4.852341738080253
Mean of train MSE of Sklearn =  4.852341738080253
Mean of test/validation MSE =  5.4364088830222
Mean of test/validation MSE of Sklearn =  5.4364088830222
```

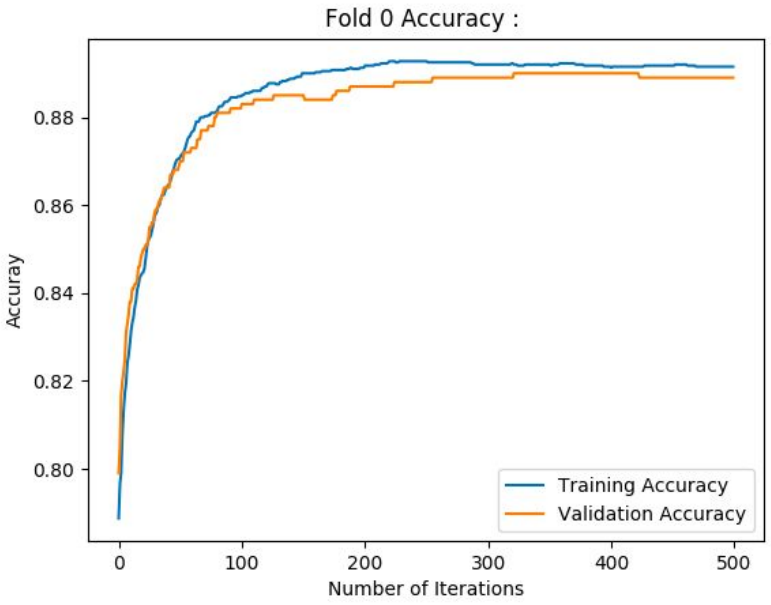
Question 2:

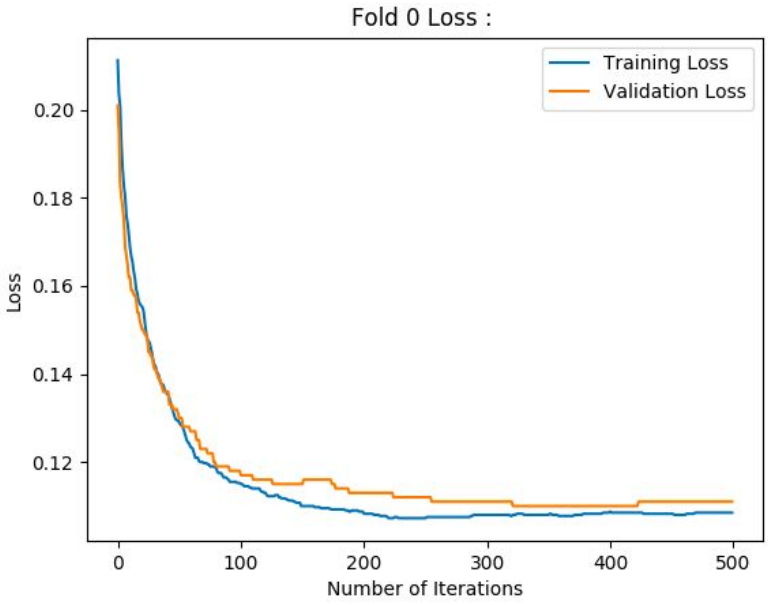
a) Following is a scatter graph of ‘dataset_1.mat’ :



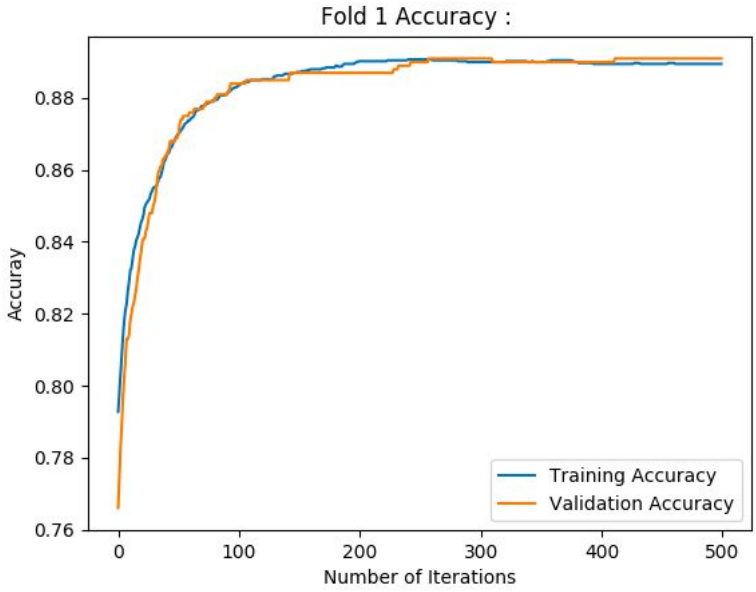
c) For this part, I have coded the LogRegression class and for each fold (5 folds), I have plotted 2 graphs for Accuracy and Loss, wherein each graph has 2 curves : for training and for validation. Following are the graphs:

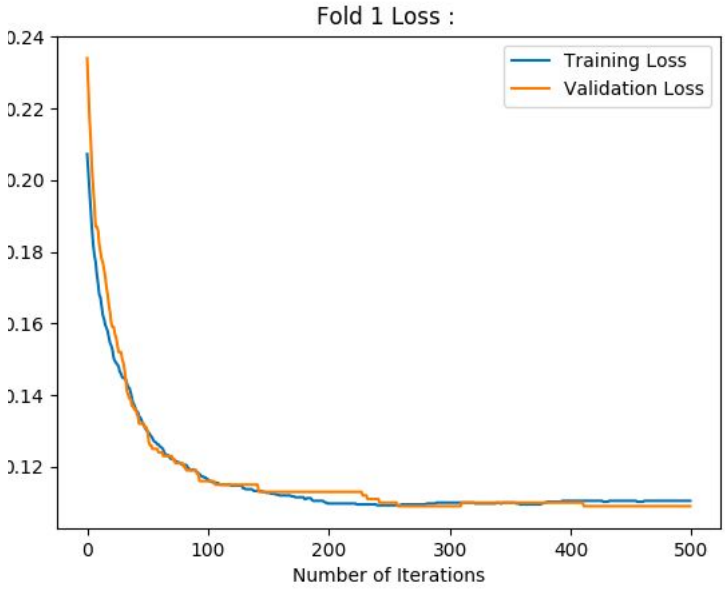
Fold 0:



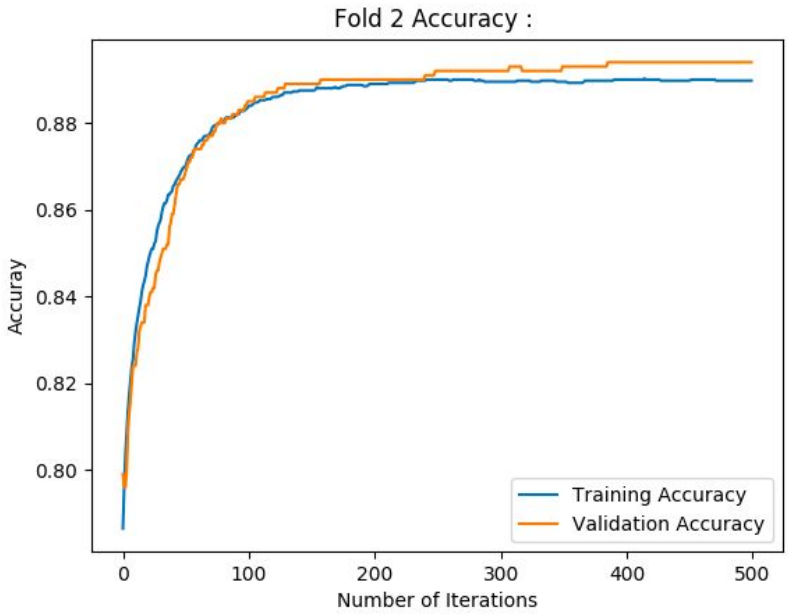


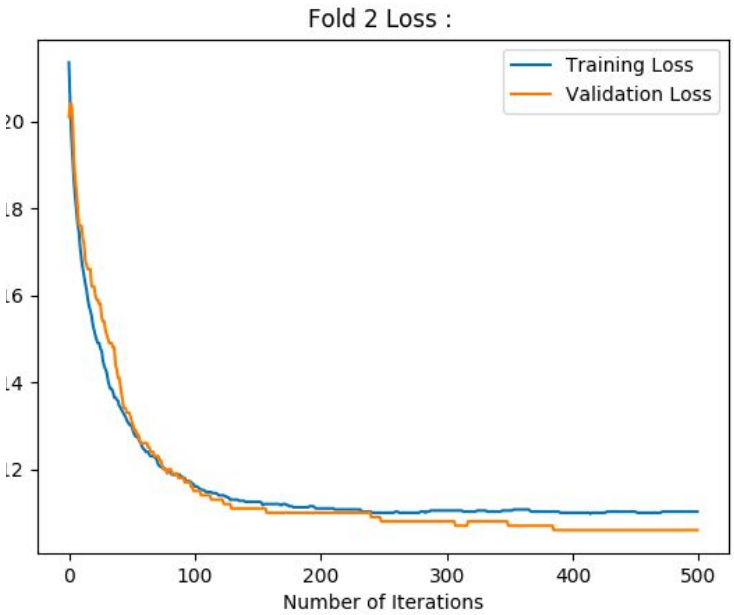
Fold 1:



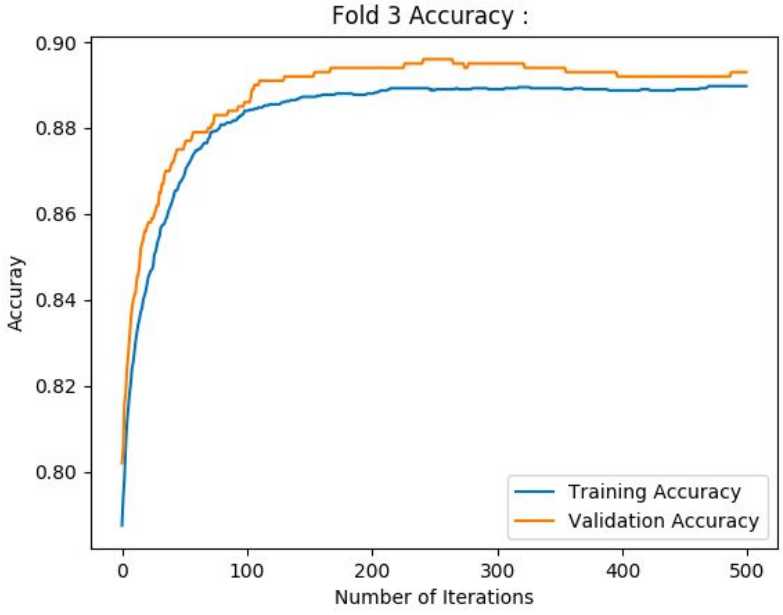


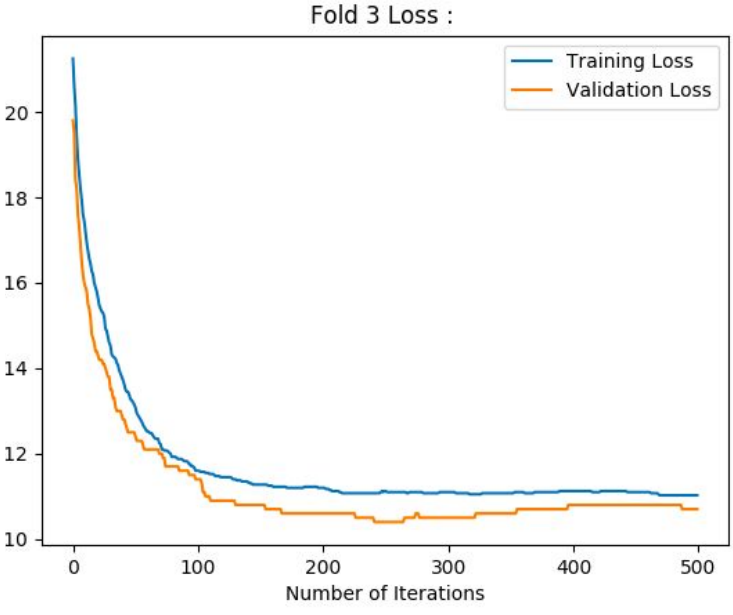
Fold 2:


Fold 2 Accuracy :
Accuray
0.88
0.86
0.84
0.82
0.80
0
100
200
300
400
500
Number of Iterations
Training Accuracy
Validation Accuracy

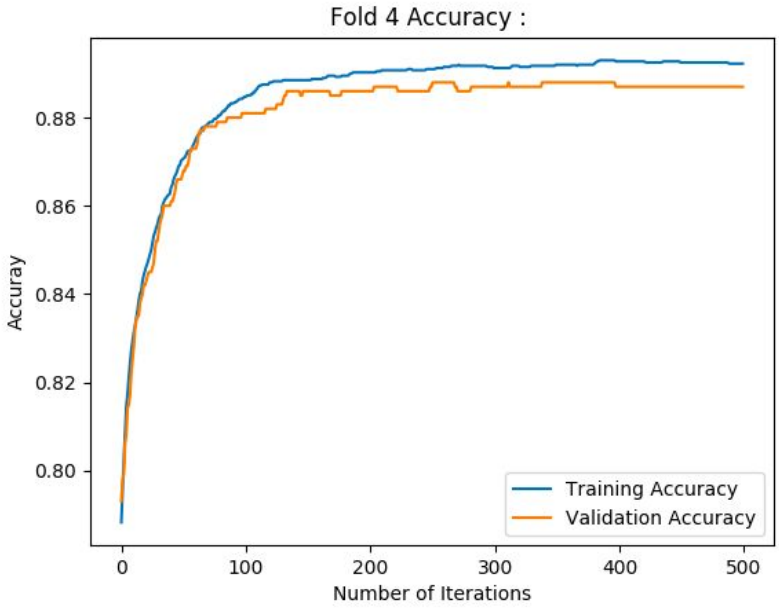

Fold 2 Loss :
Training Loss
Validation Loss
0
100
200
300
400
500
Number of Iterations

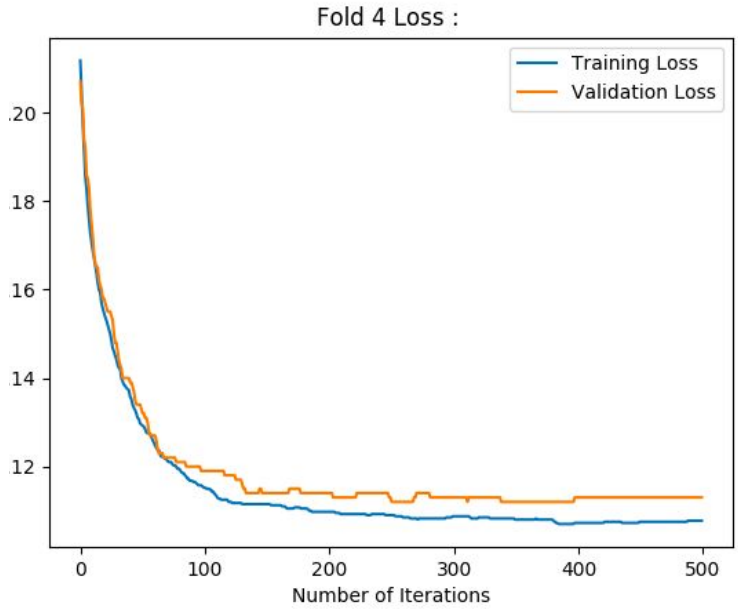
Fold 3:


Fold 3 Accuracy :
Accuray
0.90
0.88
0.86
0.84
0.82
0.80
0
100
200
300
400
500
Number of Iterations
Training Accuracy
Validation Accuracy


Fold 3 Loss :
Training Loss
Validation Loss
20
18
16
14
12
10
0
100
200
300
400
500
Number of Iterations

Fold 4:





Following is the accuracy and loss table similar to one shown in Question 1 (see "Q2_c.csv"):

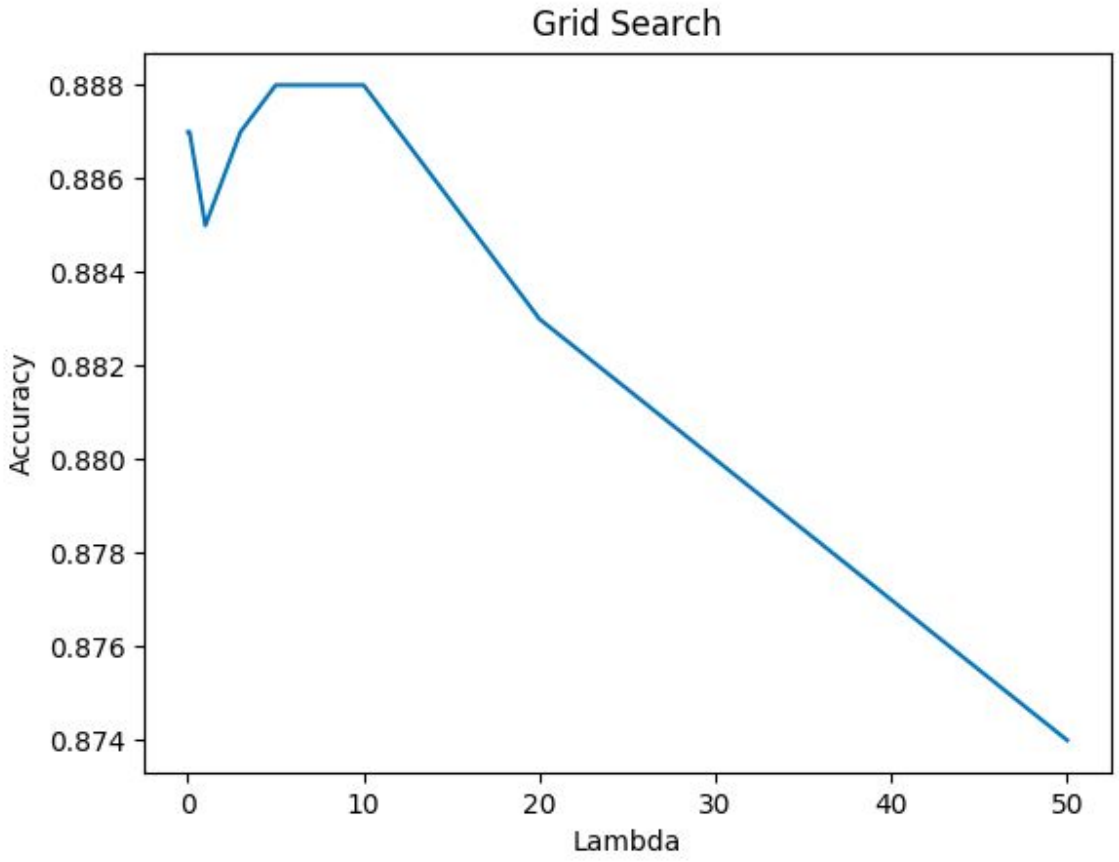
Q2_c

Fold Number	Train Accuracy	Validation Accuracy	Train Loss	Validation Loss
0	0.8915	0.889	0.1085	0.111
1	0.8895	0.891	0.1105	0.109
2	0.88975	0.894	0.11025	0.106
3	0.88975	0.893	0.11025	0.107
4	0.89225	0.887	0.10775	0.113

Following are the mean values of above data:

```
Mean of train accuracy =  0.89055
Mean of train loss =  0.10945
Mean of test/validation accuracy =  0.8908000000000001
Mean of test/validation loss =  0.1092
```

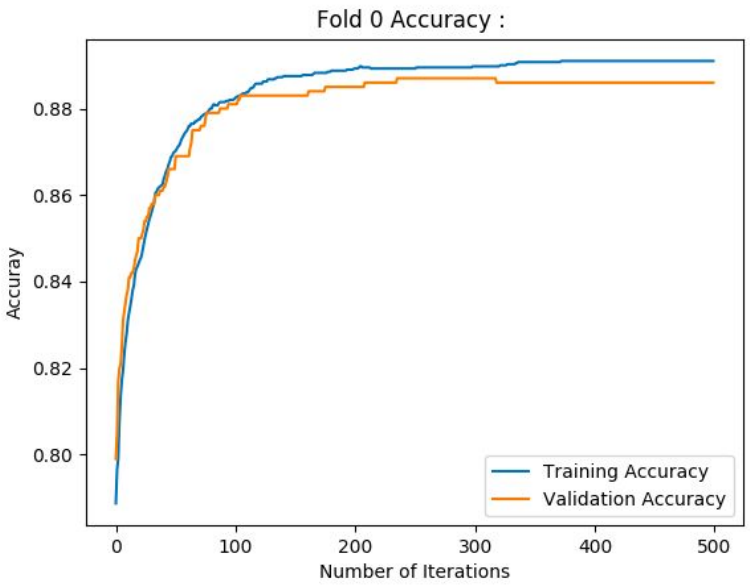
d) For this part, first I have performed grid search on the values of lambda. The values of lambda used for grid search are : [0.01,0.05,.1,1,2,3,5,10,20,50]. The optimal value I have got for lambda is : 5. Following is the graph for lambda vs accuracy (on fold 5):

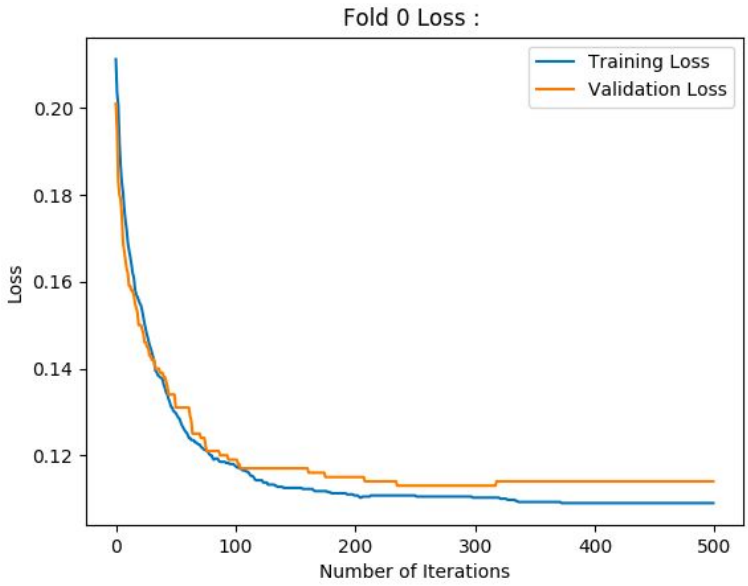


From the above graph, it can be clearly seen that accuracy is highest at lambda = 5 in l2 regularization.
So, after choosing lambda = 5 and applying Logistic Regression, following are the results:

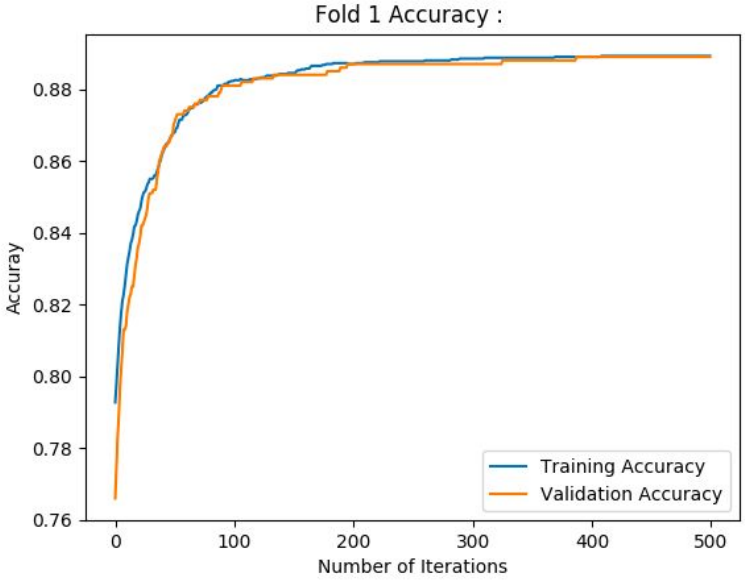
Next page ⇒

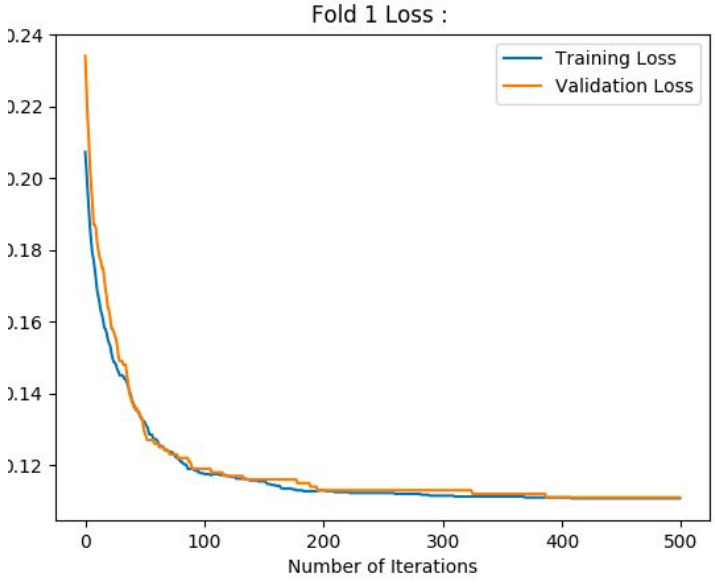
Fold 0:


Fold 0 Accuracy :
Accuray
Number of Iterations
Training Accuracy
Validation Accuracy

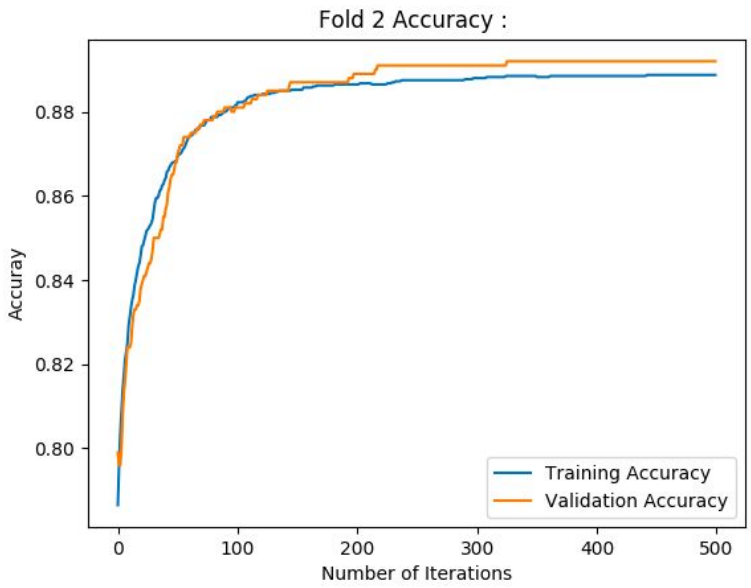

Fold 0 Loss :
Loss
Number of Iterations
Training Loss
Validation Loss

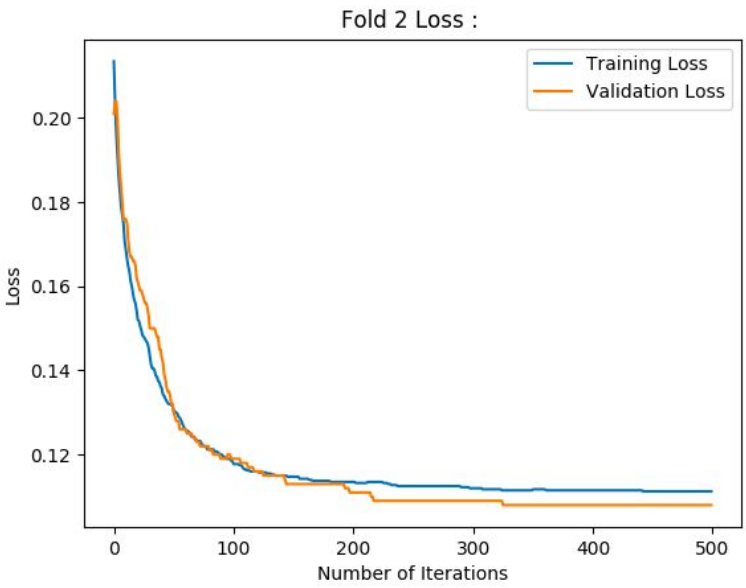
Fold 1:


Fold 1 Accuracy :
Accuray
Number of Iterations
Training Accuracy
Validation Accuracy

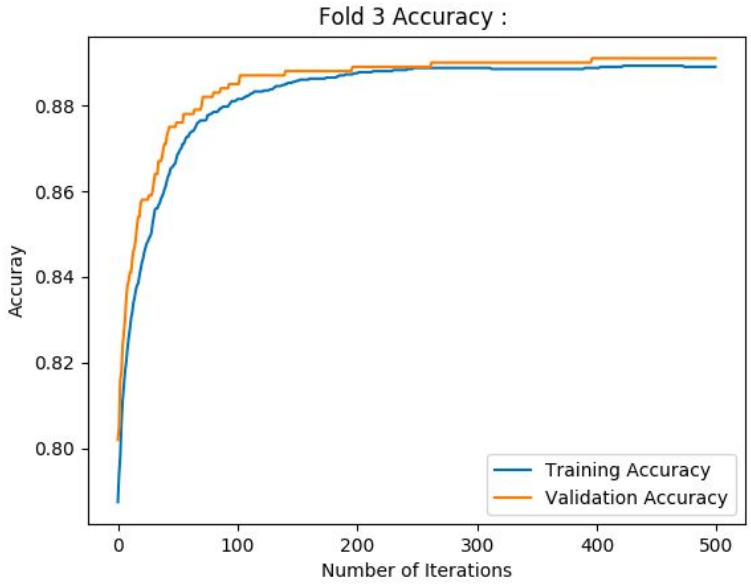

Fold 1 Loss :
Number of Iterations
Training Loss
Validation Loss

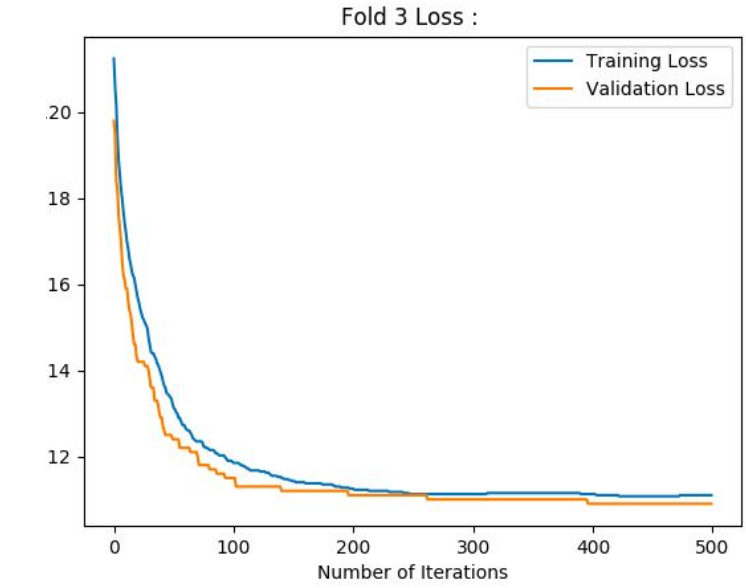
Fold 2:


Fold 2 Accuracy :
Accuray
Number of Iterations
Training Accuracy
Validation Accuracy

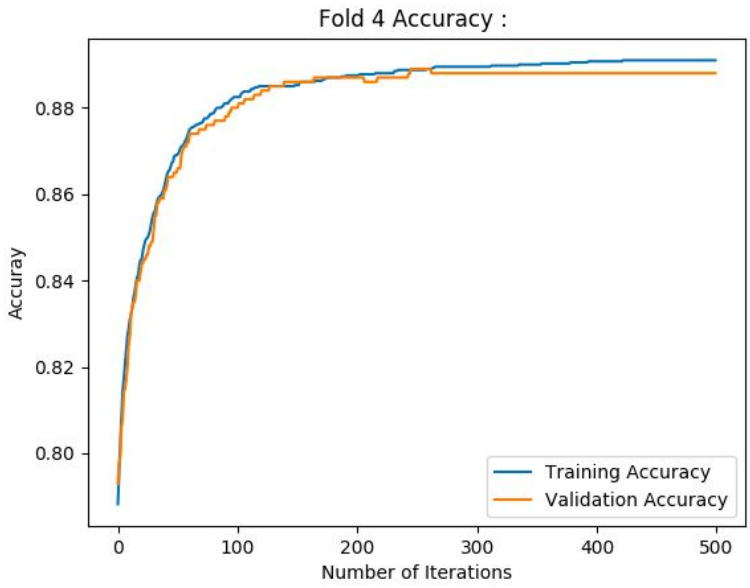

Fold 2 Loss :
Loss
Number of Iterations
Training Loss
Validation Loss

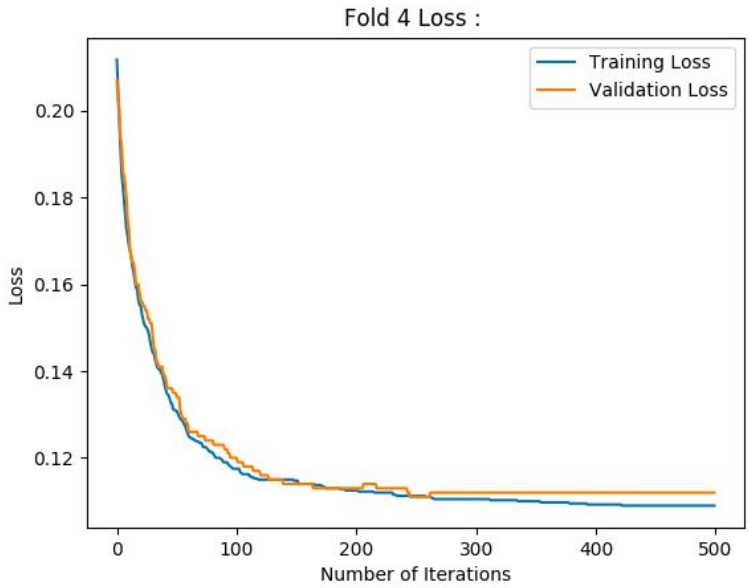
Fold 3:


Fold 3 Accuracy :
Accuray
Number of Iterations
Training Accuracy
Validation Accuracy


Fold 3 Loss :
Number of Iterations
Training Loss
Validation Loss

Fold 4:





Following is the accuracy and loss table similar to one shown in d) part (see "Q2_d.csv"):

Q2_d

Fold Number	Train Accuracy	Validation Accuracy	Train Loss	Validation Loss
0	0.891	0.886	0.109	0.114
1	0.88925	0.889	0.11075	0.111
2	0.88875	0.892	0.11125	0.108
3	0.889	0.891	0.111	0.109
4	0.891	0.888	0.109	0.112

Mean of these accuracy and losses:

```
Mean of train accuracy =  0.8897999999999999
Mean of train loss =  0.1102
Mean of test/validation accuracy =  0.8892
Mean of test/validation loss =  0.11080000000000001
```

e) For this part, I have simply used the LogisticRegression class from the sklearn module. Following are the results :

Q2_e

Fold Number	Train Accuracy	Validation Accuracy
0	0.89225	0.89
1	0.88975	0.891
2	0.88975	0.893
3	0.88875	0.893
4	0.8925	0.888

Mean of training and validation accuracy :

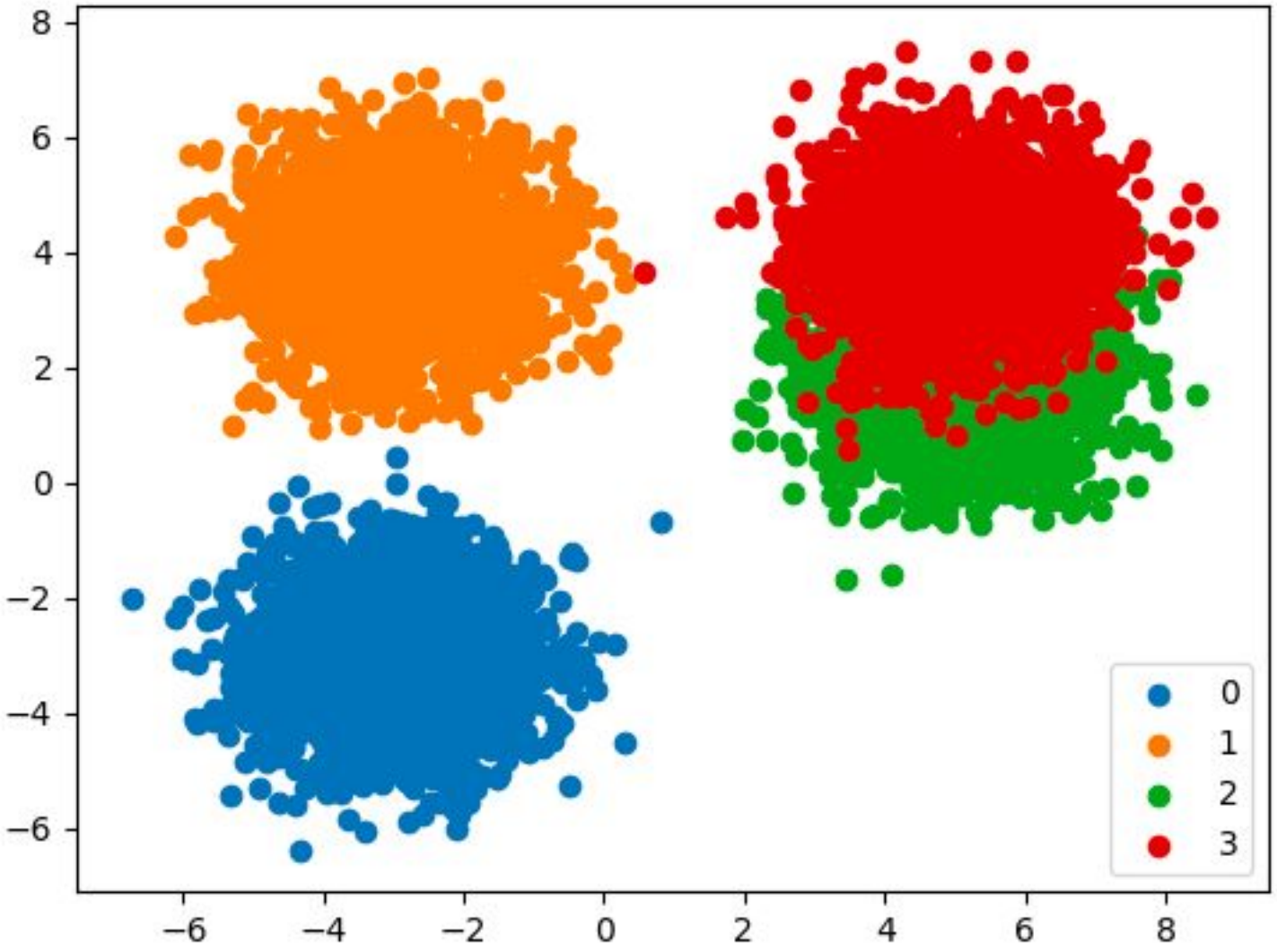
```
Mean of train accuracy =  0.8906000000000001
Mean of test/validation accuracy =  0.891
```

Observation/Notes :

- ➔ Accuracy in all the 3 cases (part (c), (d) and (e)) is almost the same with negligible difference.
- ➔ When applying LogRegression with l2 regularization, accuray has fallen by negligible amount.
- ➔ The LogisticRegression from sklearn give almost equal accuracy on the data as by the methods implemented in LogRegression.

Question 3 :

a) Following is a scatter graph of ‘dataset_2.mat’ :



b) See file “Q3_b.py”. This is the OVO implementation: Accuracy of distinction between whole dataset and 0 and 3 is 40%.

c) See file “Q3_c.py”. This is the OVR implementation

d) For this part, I have used the sklearn model : ‘LogisticRegression’. On training the model on train data, accuracy is coming out to be = 0.9255 = 92.55%. It is very good accuracy.