ML Assignment2

Anjali MT20082

5-fold splits with one fold used as a validation set and remaining four folds as the training sets at a particular instance, thus we have 5 models. This is followed for all questions and their subparts.

Answer 1a

Approach : fit used from sklearn, predict implemented from scratch as asked in question.

pre-processing: .data file read as a csv file, label encoding done for gender column.

assumptions : The dataset consists of nine columns, and the last column represents the target variable. The remaining columns denote the features.

The required answer Implemented in the code.

Answer 1b

Train and test MSE and also mean MSE.

	Model Number	Train MSE	Test MSE
0	1.0	4.978041	4.640657
1	2.0	4.702434	6.330742
2	3.0	4.911267	4.907774
3	4.0	4.965461	4.693391
4	5.0	4.916009	4.890618

```
Mean train MSE :  4.894642310267241
Mean test MSE :  5.092636177857865
```

Comparison with sklearn mean_squared_error function:

	Model Number	Train MSE SKL	Test MSE SKL
0	1.0	4.978041	4.640657
1	2.0	4.702434	6.330742
2	3.0	4.911267	4.907774
3	4.0	4.965461	4.693391
4	5.0	4.916009	4.890618

```
Mean train MSE SKL :  4.89464231026724
Mean test MSE SKL:  5.092636177857864
```

I got the same results.

Answer 1c
Using normal equations, predictions made:

The results are same as I got in part b as shown below

	Model Number	Train MSE NE	Test MSE NE
0	1.0	4.978041	4.640657
1	2.0	4.702434	6.330742
2	3.0	4.911267	4.907774
3	4.0	4.965461	4.693391
4	5.0	4.916009	4.890618

```
Training MSE 4.89464231026724
Test MSE 5.0926361778582745
```

Answer 1d

using the `LinearRegression' from the sklearn to make the predictions:

	Model Number	Train MSE SKL	Test MSE SKL
0	1.0	4.978041	4.640657
1	2.0	4.702434	6.330742
2	3.0	4.911267	4.907774
3	4.0	4.965461	4.693391
4	5.0	4.916009	4.890618

```
Mean mse skl train :  4.89464231026724
Mean mse skl test :  5.092636177857864
```

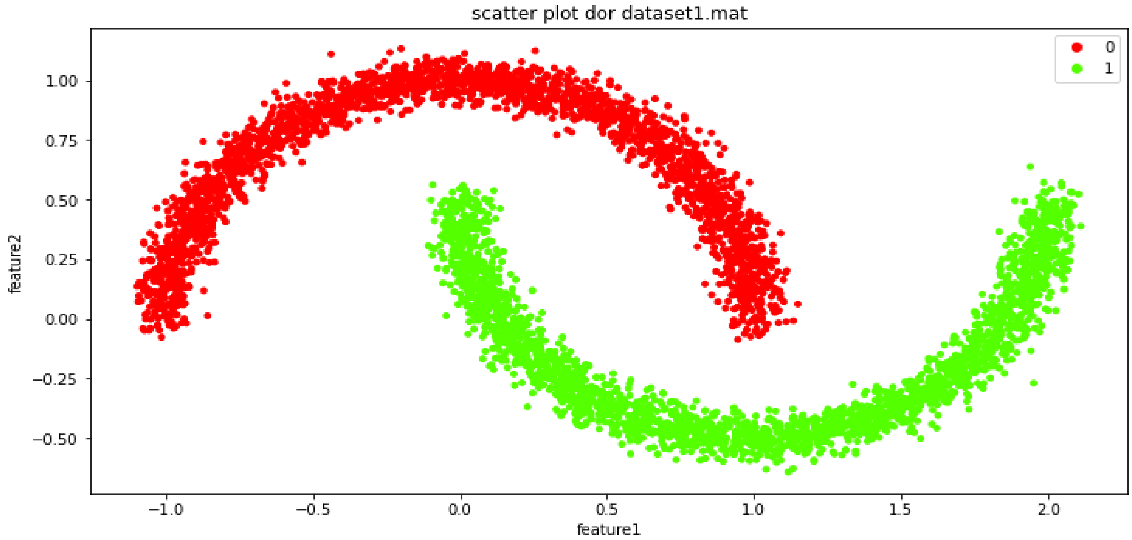
I got the same results for MSE in part b, part c, part d.

ML Assignment2

Anjali MT20082

Answer 2a
Scatter plot for dataset1.mat :

x axis- feature 1
y-axis – feature 2



Answer 2b

Approach : fit , predict implemented from scratch for log regression

pre-processing: no such preprocessing required.

assumptions : The dataset consists of three columns, and the labels column represents the target variable. The remaining columns denote the features.

Functions implemented inside the LogRegression class –
sigmoid(), fit(), predict() and probab_predict(),

The parameters `reg`, `reg_constant` have been added in order to accommodate regularization and the parameters `class_of_interest`, `multiclass` have been added in order to accommodate multiclass in fit function of the logistic regression class.

The required answer Implemented in the code.

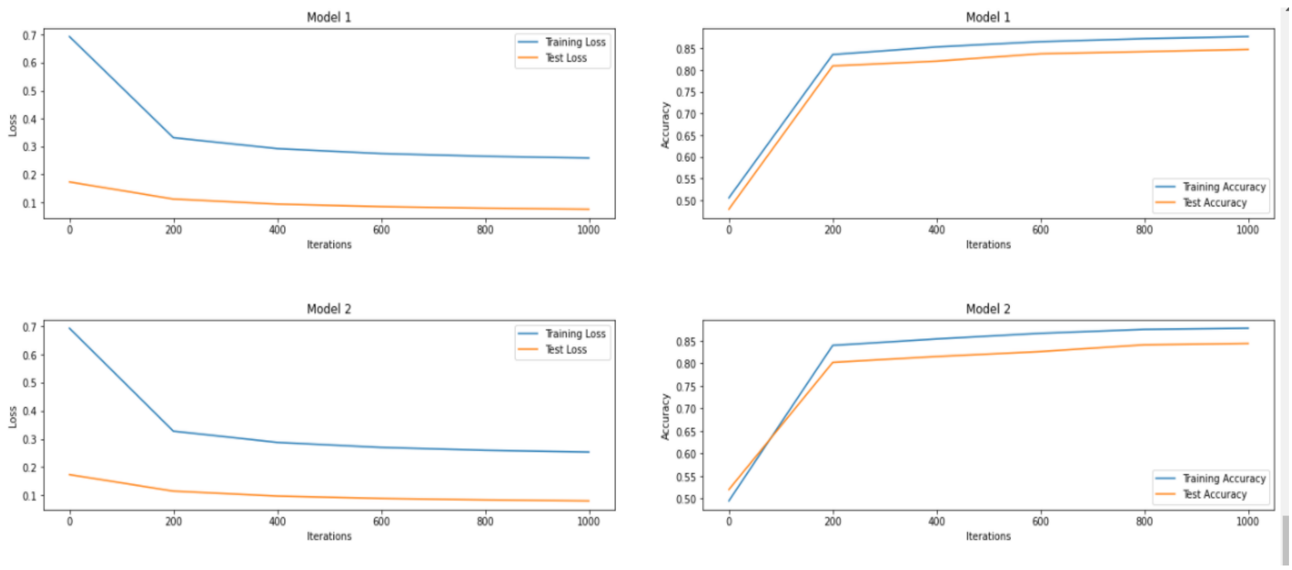
Answer 2c

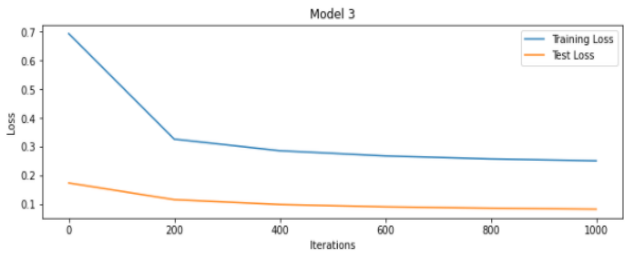
Using `LogRegression' class, performance is reported over 5-folds in terms of accuracy and loss. Also, the training curves are plotted with each fold as the validation set. For each fold there are two plots, one for accuracy, and other for loss. Each plot contains two curves, one representing training statistics, and other validation statistics.

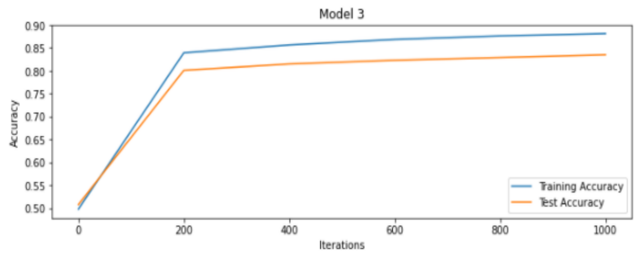
	Model Number	Iterations	Train Accuracy	Test Accuracy	Train Loss	Test Loss
0	1	0	0.50525	0.479	0.693127	0.173282
1	2	0	0.49500	0.520	0.693127	0.173282
2	3	0	0.49800	0.508	0.693127	0.173282
3	4	0	0.49825	0.507	0.693127	0.173282
4	5	0	0.50350	0.486	0.693127	0.173282
5	1	200	0.83625	0.810	0.331629	0.112233
6	2	200	0.83975	0.802	0.327616	0.114629
7	3	200	0.83925	0.801	0.325820	0.115106
8	4	200	0.83575	0.825	0.329809	0.113590
9	5	200	0.83825	0.812	0.330196	0.112064
10	1	400	0.85375	0.821	0.292479	0.093987
11	2	400	0.85400	0.815	0.287749	0.097050
12	3	400	0.85650	0.815	0.285449	0.098254
13	4	400	0.85275	0.832	0.290220	0.095456
14	5	400	0.85650	0.825	0.290106	0.094460
15	1	600	0.86600	0.838	0.274860	0.085085

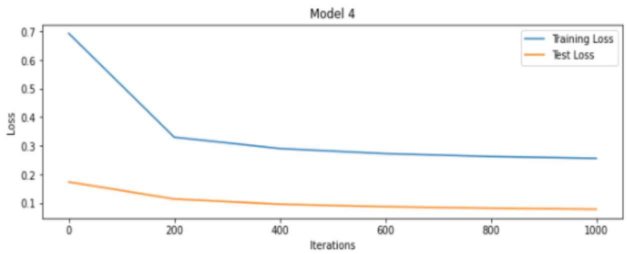
16	2	600	0.86650	0.826	0.269796	0.088656
17	3	600	0.86850	0.823	0.267253	0.090260
18	4	600	0.86550	0.842	0.272324	0.086759
19	5	600	0.86775	0.830	0.272026	0.086112
20	1	800	0.87300	0.843	0.264936	0.079645
21	2	800	0.87525	0.841	0.259652	0.083574
22	3	800	0.87625	0.829	0.256955	0.085433
23	4	800	0.87300	0.846	0.262180	0.081494
24	5	800	0.87475	0.837	0.261810	0.081093
25	1	1000	0.87775	0.848	0.258808	0.075869
26	2	1000	0.87775	0.844	0.253359	0.080058
27	3	1000	0.88125	0.835	0.250553	0.082096
28	4	1000	0.87875	0.853	0.255871	0.077860
29	5	1000	0.87950	0.841	0.255475	0.077641

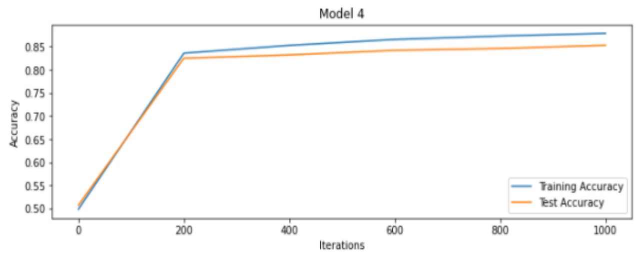
Plots depicting the same data as the above table

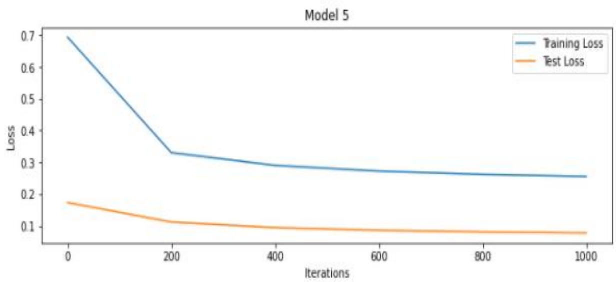


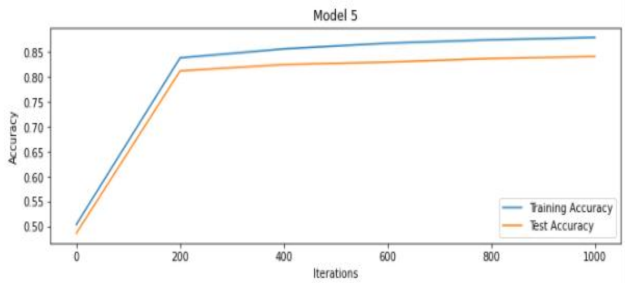












Loss and accuracy have been calculated over 1000 iterations. These values have been recorded after every 200 iterations to observe the change in loss and accuracy in the model. This has been followed in the all the subparts of this question and the next question too.

Answer 2d

Grid search on lambda (regularisation constant):

Anjali MT20082

	Model Number	Lambda	Iterations	Train Accuracy	Test Accuracy	Train Loss	Test Loss
0	1	0.1	1000	0.88050	0.838	0.253125	0.080789
1	2	0.1	1000	0.87600	0.851	0.262142	0.073009
2	3	0.1	1000	0.87875	0.850	0.253692	0.079697
3	4	0.1	1000	0.88025	0.845	0.253671	0.079795
4	5	0.1	1000	0.88025	0.839	0.252787	0.080384
5	1	0.2	1000	0.88050	0.838	0.253428	0.080898
6	2	0.2	1000	0.87600	0.851	0.262436	0.073131
7	3	0.2	1000	0.87875	0.850	0.253996	0.079810
8	4	0.2	1000	0.88025	0.845	0.253974	0.079908
9	5	0.2	1000	0.88025	0.839	0.253091	0.080495
10	1	0.3	1000	0.88050	0.838	0.253731	0.081007
11	2	0.3	1000	0.87600	0.851	0.262729	0.073252
12	3	0.3	1000	0.87875	0.850	0.254299	0.079922
13	4	0.3	1000	0.88025	0.845	0.254276	0.080022
14	5	0.3	1000	0.88025	0.839	0.253395	0.080604
15	1	1.0	1000	0.88025	0.837	0.255823	0.081760
16	2	1.0	1000	0.87600	0.851	0.264758	0.074092
17	3	1.0	1000	0.87850	0.850	0.256396	0.080700
18	4	1.0	1000	0.87950	0.845	0.256368	0.080805
19	5	1.0	1000	0.88025	0.839	0.255495	0.081364
20	1	5.0	1000	0.87925	0.837	0.267069	0.085796
21	2	5.0	1000	0.87500	0.851	0.275669	0.078585
22	3	5.0	1000	0.87800	0.849	0.267668	0.084865
23	4	5.0	1000	0.87775	0.845	0.267614	0.085002
24	5	5.0	1000	0.87875	0.839	0.266781	0.085433
25	1	10.0	1000	0.87800	0.836	0.279635	0.090278
26	2	10.0	1000	0.87400	0.850	0.287881	0.083558
27	3	10.0	1000	0.87650	0.849	0.280262	0.089485
28	4	10.0	1000	0.87700	0.845	0.280181	0.089654
29	5	10.0	1000	0.87825	0.839	0.279398	0.089945

From the above table at lambda = 0.1, we get optimal values of accuracy and loss.
With lambta – 0.1, table and plot same as part c

ML Assignment2

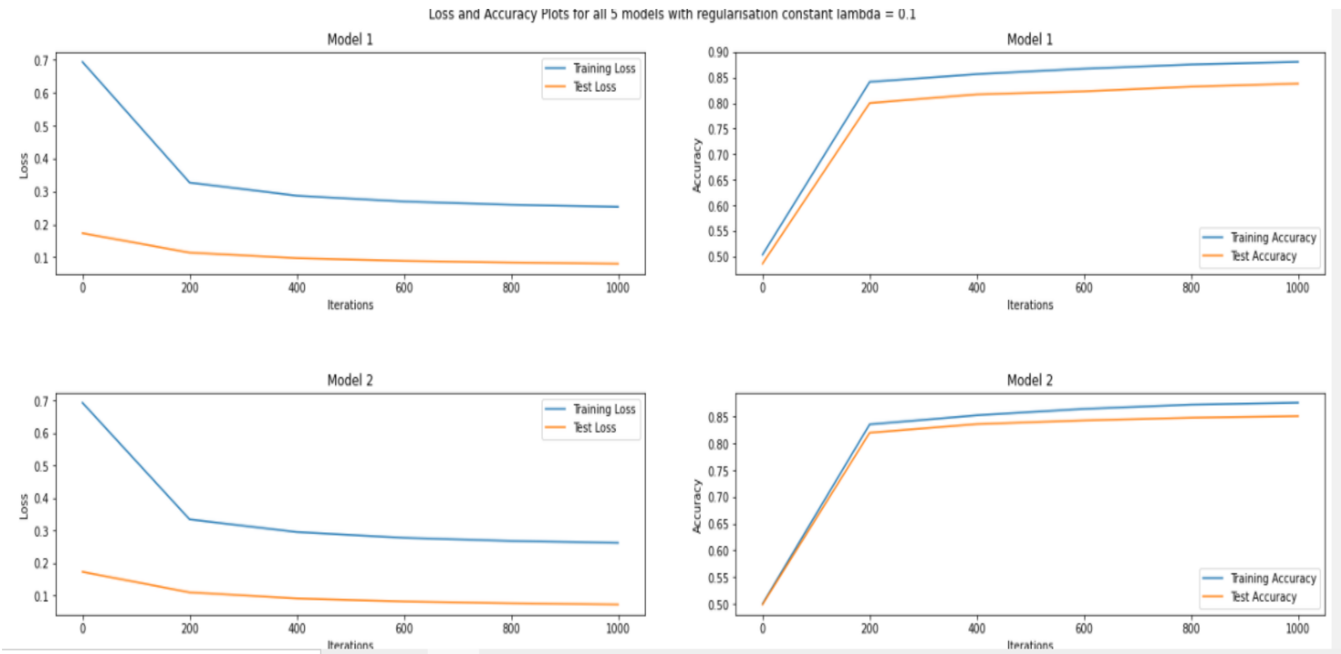
Anjali MT20082

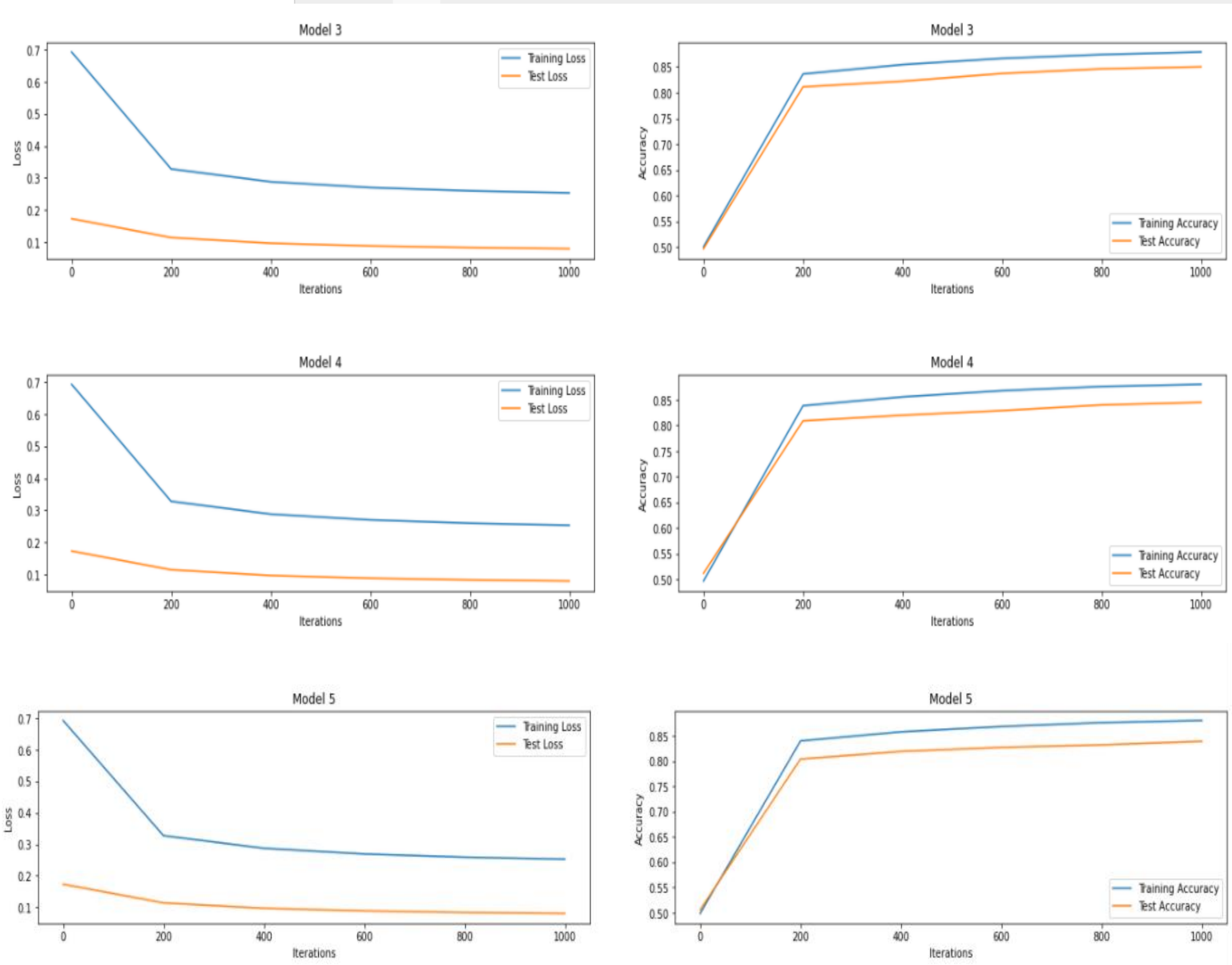
	Model Number	Iterations	Train Accuracy	Test Accuracy	Train Loss	Test Loss
0	1	0	0.50350	0.486	0.693127	0.173282
1	2	0	0.50025	0.499	0.693127	0.173282
2	3	0	0.50075	0.497	0.693127	0.173282
3	4	0	0.49700	0.512	0.693127	0.173282
4	5	0	0.49850	0.506	0.693127	0.173282
5	1	200	0.84150	0.800	0.326878	0.114187
6	2	200	0.83550	0.820	0.334546	0.110168
7	3	200	0.83600	0.811	0.328269	0.114369
8	4	200	0.83850	0.809	0.328185	0.114825
9	5	200	0.83975	0.804	0.327571	0.114120
10	1	400	0.85650	0.817	0.287071	0.097286
11	2	400	0.85250	0.836	0.295499	0.091349
12	3	400	0.85425	0.822	0.288368	0.096720
13	4	400	0.85575	0.820	0.288332	0.096968
14	5	400	0.85775	0.819	0.287426	0.096904
15	1	600	0.86725	0.823	0.269310	0.089202
16	2	600	0.86450	0.843	0.277981	0.082286
17	3	600	0.86625	0.837	0.270307	0.088280
18	4	600	0.86775	0.829	0.270279	0.088437
19	5	600	0.86825	0.827	0.269338	0.088716
20	1	800	0.87550	0.832	0.259312	0.084255
21	2	800	0.87250	0.848	0.268167	0.076797
22	3	800	0.87375	0.846	0.260064	0.083193
23	4	800	0.87600	0.840	0.260040	0.083309
24	5	800	0.87600	0.832	0.259122	0.083782
25	1	1000	0.88050	0.838	0.253125	0.080789
26	2	1000	0.87600	0.851	0.262142	0.073009
27	3	1000	0.87875	0.850	0.253692	0.079697
28	4	1000	0.88025	0.845	0.253671	0.079795
29	5	1000	0.88025	0.839	0.252787	0.080384

Plots:

ML Assignment2

Anjali MT20082





Answer 2e

ML Assignment2

Anjali MT20082

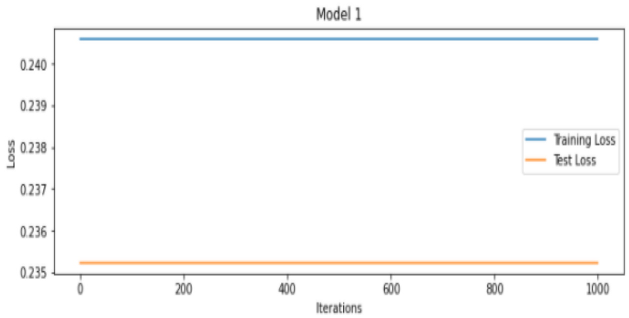
Approach : Using the logistic regression from the sklearn to obtain the performance over the 5-folds. All assumptions are same as above.

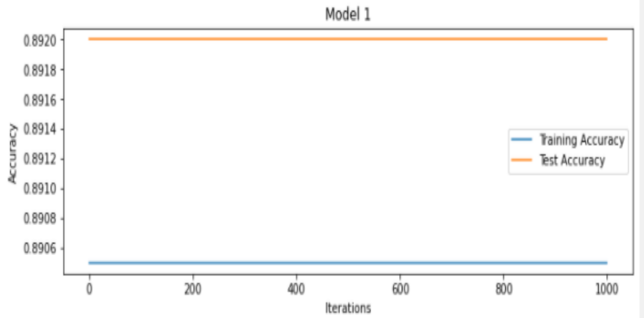
	Model Number	Iterations	Train Accuracy	Test Accuracy	Train Loss	Test Loss
0	1	0	0.89050	0.892	0.240592	0.235231
1	2	0	0.88925	0.895	0.242969	0.225560
2	3	0	0.88950	0.896	0.241289	0.232748
3	4	0	0.89100	0.887	0.236338	0.252142
4	5	0	0.89200	0.885	0.235885	0.253107
5	1	200	0.89050	0.892	0.240592	0.235231
6	2	200	0.88925	0.895	0.242969	0.225560
7	3	200	0.88950	0.896	0.241289	0.232748
8	4	200	0.89100	0.887	0.236338	0.252142
9	5	200	0.89200	0.885	0.235885	0.253107
10	1	400	0.89050	0.892	0.240592	0.235231
11	2	400	0.88925	0.895	0.242969	0.225560
12	3	400	0.88950	0.896	0.241289	0.232748
13	4	400	0.89100	0.887	0.236338	0.252142
14	5	400	0.89200	0.885	0.235885	0.253107
15	1	600	0.89050	0.892	0.240592	0.235231
16	2	600	0.88925	0.895	0.242969	0.225560
17	3	600	0.88950	0.896	0.241289	0.232748
18	4	600	0.89100	0.887	0.236338	0.252142
19	5	600	0.89200	0.885	0.235885	0.253107
20	1	800	0.89050	0.892	0.240592	0.235231
21	2	800	0.88925	0.895	0.242969	0.225560
22	3	800	0.88950	0.896	0.241289	0.232748
23	4	800	0.89100	0.887	0.236338	0.252142
24	5	800	0.89200	0.885	0.235885	0.253107
25	1	1000	0.89050	0.892	0.240592	0.235231
26	2	1000	0.88925	0.895	0.242969	0.225560
27	3	1000	0.88950	0.896	0.241289	0.232748
28	4	1000	0.89100	0.887	0.236338	0.252142
29	5	1000	0.89200	0.885	0.235885	0.253107

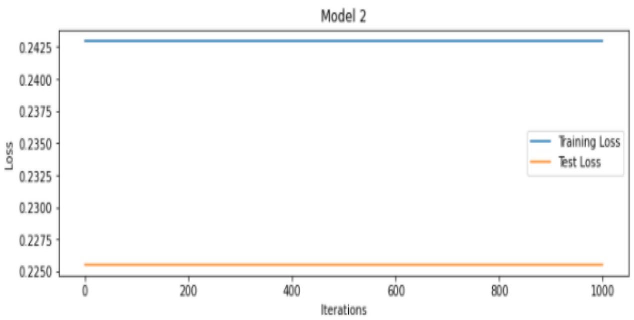
Plots:

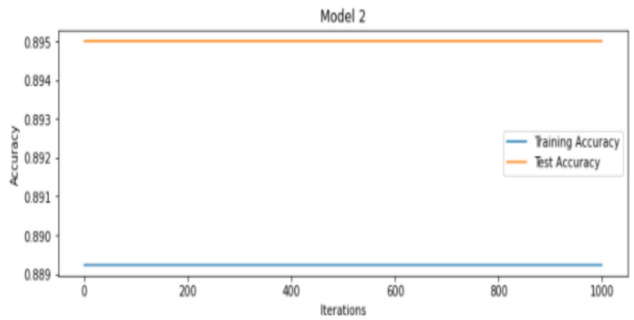
ML Assignment2

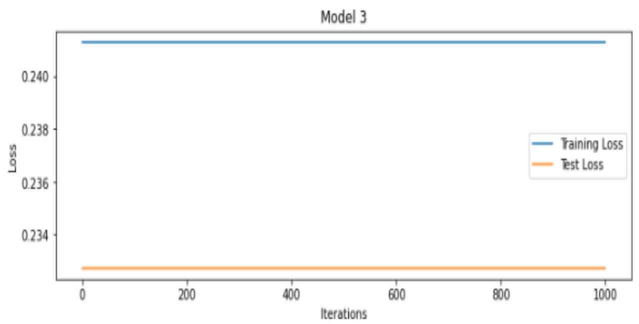
Anjali MT20082

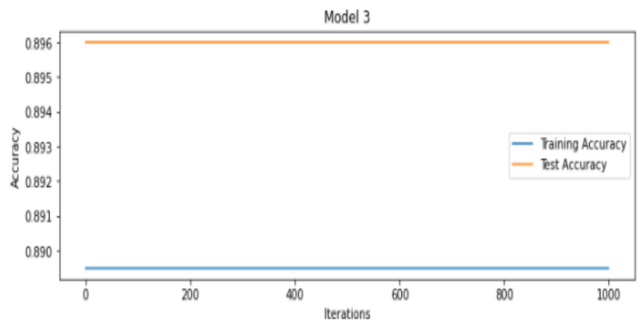


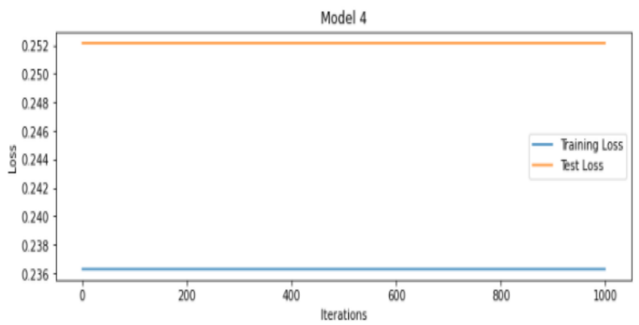


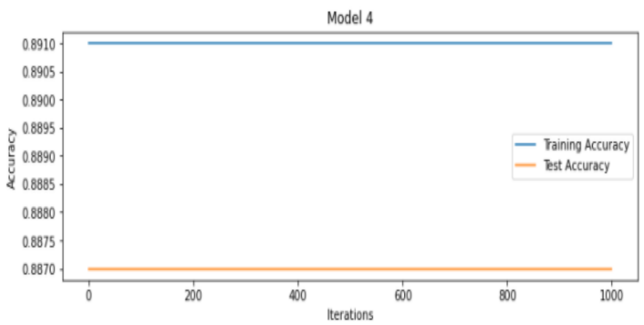


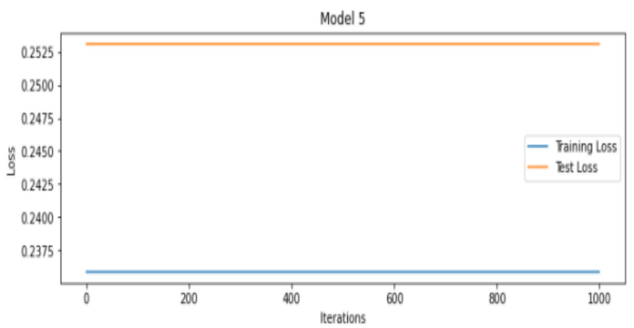


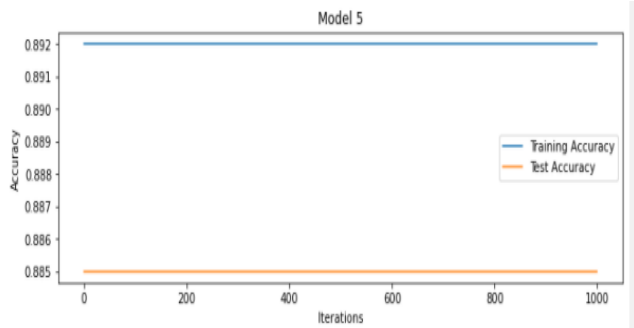












With Regularisation, table:

Iterations

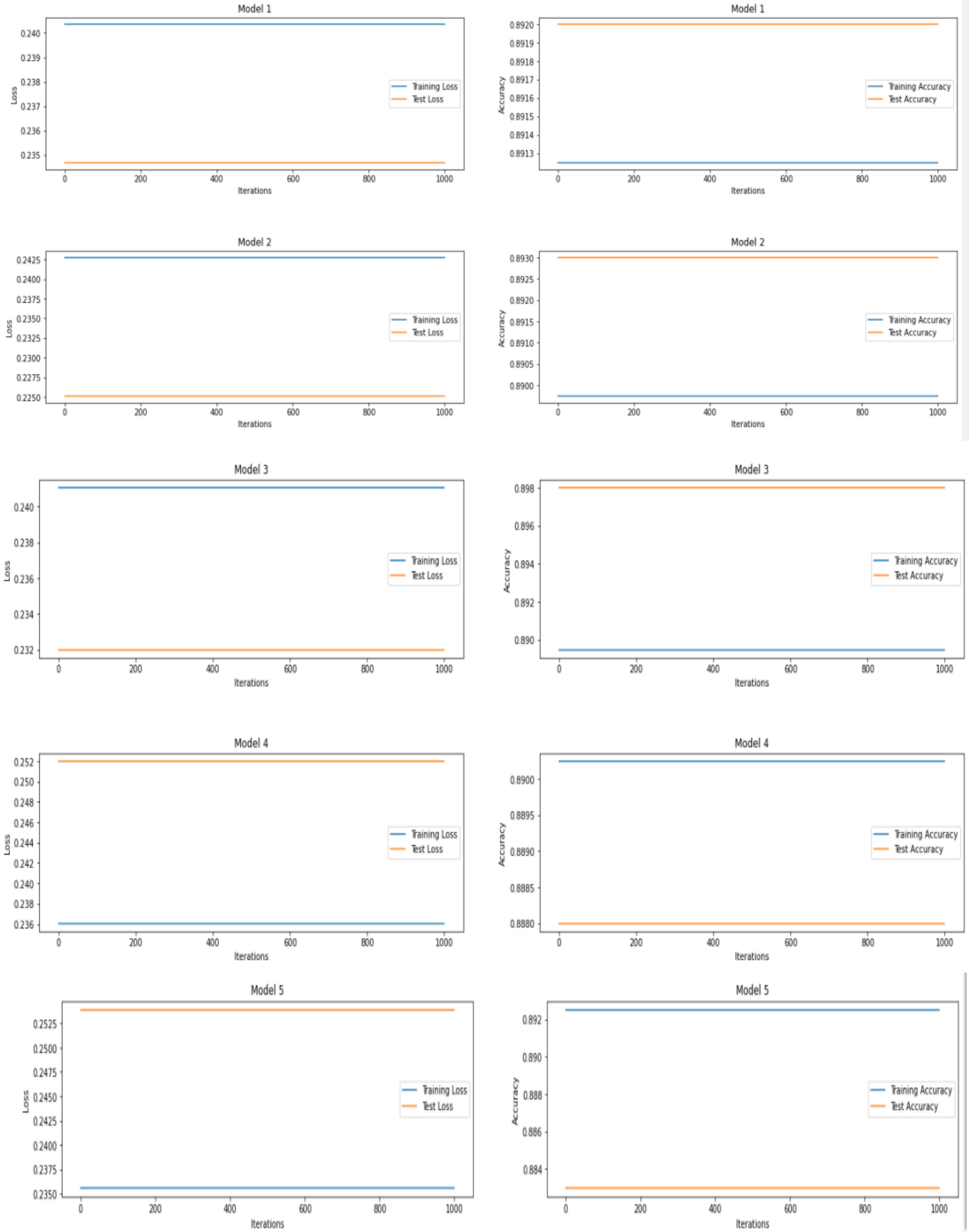
	Model Number	Iterations	Train Accuracy	Test Accuracy	Train Loss	Test Loss
0	1	0	0.89125	0.892	0.240359	0.234698
1	2	0	0.88975	0.893	0.242737	0.225164
2	3	0	0.88950	0.898	0.241062	0.232011
3	4	0	0.89025	0.888	0.236096	0.251995
4	5	0	0.89250	0.883	0.235623	0.253870
5	1	200	0.89125	0.892	0.240359	0.234698
6	2	200	0.88975	0.893	0.242737	0.225164
7	3	200	0.88950	0.898	0.241062	0.232011
8	4	200	0.89025	0.888	0.236096	0.251995
9	5	200	0.89250	0.883	0.235623	0.253870
10	1	400	0.89125	0.892	0.240359	0.234698
11	2	400	0.88975	0.893	0.242737	0.225164
12	3	400	0.88950	0.898	0.241062	0.232011
13	4	400	0.89025	0.888	0.236096	0.251995
14	5	400	0.89250	0.883	0.235623	0.253870
15	1	600	0.89125	0.892	0.240359	0.234698
16	2	600	0.88975	0.893	0.242737	0.225164
17	3	600	0.88950	0.898	0.241062	0.232011
18	4	600	0.89025	0.888	0.236096	0.251995
19	5	600	0.89250	0.883	0.235623	0.253870
20	1	800	0.89125	0.892	0.240359	0.234698
21	2	800	0.88975	0.893	0.242737	0.225164
22	3	800	0.88950	0.898	0.241062	0.232011
23	4	800	0.89025	0.888	0.236096	0.251995
24	5	800	0.89250	0.883	0.235623	0.253870
25	1	1000	0.89125	0.892	0.240359	0.234698
26	2	1000	0.88975	0.893	0.242737	0.225164
27	3	1000	0.88950	0.898	0.241062	0.232011
28	4	1000	0.89025	0.888	0.236096	0.251995
29	5	1000	0.89250	0.883	0.235623	0.253870

Plots with regularization:

ML Assignment2

Anjali MT20082

Loss and Accuracy Plots for all 5 models with regularisation constant lambda = 0.1



Anjali MT20082

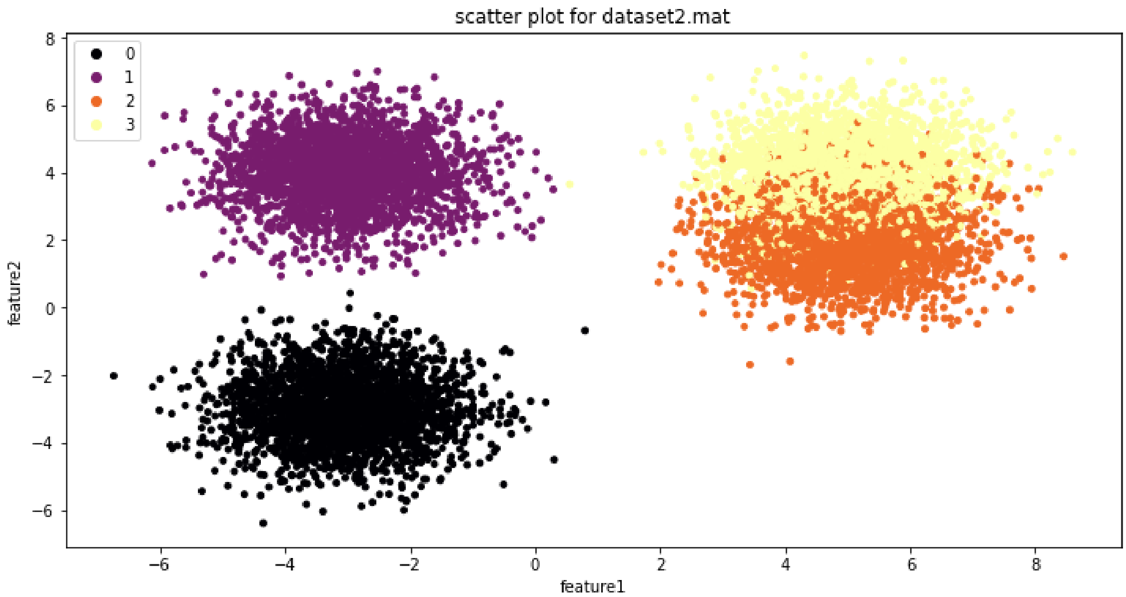
There has been a difference in the results of sklearn maybe because the internal implementation in sklearn is such that the accuracy and loss do not depend on the number of iterations and give the best results without any iterations.

Answer 3a

scatter plot for dataset2.mat

x-axis - feature1

y-axis – feature 2



Answer 3b

One vs one has been performed by forming nC2 models for a model and then taking the voting of the y_predict labels to form the final y_predict. And hence calculate accuracy.

Not getting the desired result. Will look into it further.

```
fold :  1 :  0.2245
fold :  2 :  0.2515
fold :  3 :  0.2445
fold :  4 :  0.2575
fold :  5 :  0.272
```

Answer 3c The performance table below shows the accuracy for one vs rest performed by scratch. One vs rest is performed by making 4 models for one model and then taking the maximum probability from the 4 y_predict labels to form the new y_predict and hence calculated accuracy.

```
overall test accuracy :

Fold 1 :  0.9105
Fold 2 :  0.9145
Fold 3 :  0.9095
Fold 4 :  0.9175
Fold 5 :  0.905
overall train accuracy :

Fold 1 :  0.913375
Fold 2 :  0.9095
Fold 3 :  0.91125
Fold 4 :  0.909125
Fold 5 :  0.912875
```

Classwise accuracy and loss

	Model Number	Class	Train Accuracy	Test Accuracy	Train Loss	Test Loss
0	1	0	0.999625	0.9995	0.003807	0.002360
1	2	0	0.999625	1.0000	0.003975	0.002183
2	3	0	0.999750	0.9995	0.003864	0.002289
3	4	0	0.999625	1.0000	0.003933	0.002178
4	5	0	0.999875	0.9990	0.003563	0.002639
5	1	1	0.996250	0.9900	0.030594	0.019402
6	2	1	0.995125	0.9945	0.032431	0.017434
7	3	1	0.995250	0.9940	0.031661	0.018738
8	4	1	0.996000	0.9935	0.031876	0.017961
9	5	1	0.995875	0.9910	0.030582	0.019617
10	1	2	0.875250	0.8740	0.269001	0.071942
11	2	2	0.874625	0.8770	0.268460	0.072705
12	3	2	0.875750	0.8730	0.267053	0.073475
13	4	2	0.873625	0.8780	0.270038	0.070962

14	5	2	0.876625	0.8755	0.268893	0.071873
15	1	3	0.910375	0.8890	0.246342	0.079246
16	2	3	0.909875	0.8885	0.248838	0.076982
17	3	3	0.910750	0.8730	0.243258	0.082628
18	4	3	0.906750	0.8935	0.247432	0.078584
19	5	3	0.910375	0.8845	0.243390	0.082823

Answer 3d Using sklearn the above parts performed.

The results match with the one vs rest function implemented from scratch.

```
OVR FOLD Accuracy 1 :  0.9195
OVO FOLD Accuracy 1 :  0.916
OVR FOLD Accuracy 2 :  0.926
OVO FOLD Accuracy 2 :  0.926
OVR FOLD Accuracy 3 :  0.925
OVO FOLD Accuracy 3 :  0.9235
OVR FOLD Accuracy 4 :  0.9275
OVO FOLD Accuracy 4 :  0.928
OVR FOLD Accuracy 5 :  0.9205
OVO FOLD Accuracy 5 :  0.928
```

Classwise accuracy :

```
OVR FOLD Accuracy 1 :  0.919
Classwise accuracy for OVR  1
 class 0 :  1.0

 class 1 :  1.0

 class 2 :  0.8634453781512605

 class 3 :  0.8116504854368932

OVO FOLD Accuracy 1 :  0.917
Classwise accuracy for OVO  1
 class 0 :  1.0

 class 1 :  1.0

 class 2 :  0.8613445378151261

 class 3 :  0.8058252427184466
```

```
OVR FOLD Accuracy 2 :  0.9265
Classwise accuracy for OVR  2
 class 0 :  1.0

 class 1 :  1.0

 class 2 :  0.849802371541502

 class 3 :  0.8565656565656565

OVO FOLD Accuracy 2 :  0.93
Classwise accuracy for OVO  2
 class 0 :  1.0

 class 1 :  1.0

 class 2 :  0.8596837944664032

 class 3 :  0.8606060606060606
```

ML Assignment2

Anjali MT20082

```
OVR FOLD Accuracy 3 :  0.9155
Classwise accuracy for OVR  3
 class 0 :  1.0

 class 1 :  1.0

 class 2 :  0.8151750972762646

 class 3 :  0.8442105263157895

OVO FOLD Accuracy 3 :  0.917
Classwise accuracy for OVO  3
 class 0 :  1.0

 class 1 :  1.0

 class 2 :  0.8229571984435797

 class 3 :  0.8421052631578947
```

```
OVR FOLD Accuracy 4 :  0.9275
Classwise accuracy for OVR  4
 class 0 :  1.0

 class 1 :  1.0

 class 2 :  0.8377281947261663

 class 3 :  0.875717017208413

OVO FOLD Accuracy 4 :  0.929
Classwise accuracy for OVO  4
 class 0 :  1.0

 class 1 :  1.0

 class 2 :  0.8417849898580122

 class 3 :  0.8776290630975143
```

```
OVR FOLD Accuracy 5 :  0.9305
Classwise accuracy for OVR  5
 class 0 :  0.9980392156862745

 class 1 :  1.0

 class 2 :  0.8414872798434442

 class 3 :  0.8841463414634146

OVO FOLD Accuracy 5 :  0.9285
Classwise accuracy for OVO  5
 class 0 :  0.9980392156862745

 class 1 :  1.0

 class 2 :  0.8414872798434442

 class 3 :  0.8760162601626016
```