Classification (Using Discriminative models) Report

Reetwik Das

July 18, 2021

Contents

List of Tables

List of Figures

1 Dataset 1a

1.1 Perceptron for every pair of class

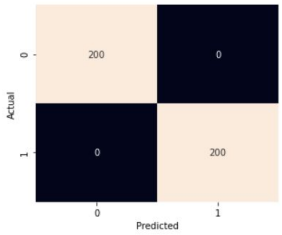
Hyperparameter $\eta = \{0.1, 0.5, 1\}$

Since the classes are linearly well separated we got the same accuracies for each value of η.

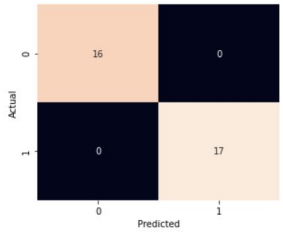
classes	Training Dataset	Validation Dataset	Test Dataset
0 vs 1	100	100	100
0 vs 2	100	100	100
0 vs 3	100	100	100
1 vs 2	100	100	100
1 vs 3	100	100	100
2 vs 3	100	100	100

Table 1.1.1: Classification accuracy using perceptron on Dataset 1a

As we can see from the table above the model with the best performance on the test data is with $\eta = 1$ with an accuracy of 100% for each pair of class .

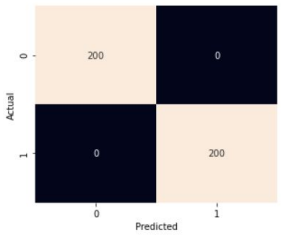


(a) Training data

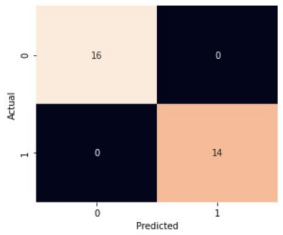


(b) Test data

Figure 1.1.1: Confusion matrix for the classification between class 0 vs class 1 of dataset 1a using Perceptron

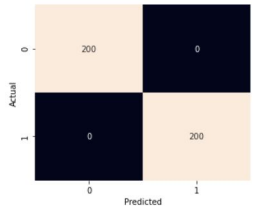


(a) Training data

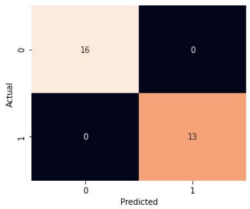


(b) Test data

Figure 1.1.2: Confusion matrix for the classification between class 0 vs class 2 of dataset 1a using Perceptron

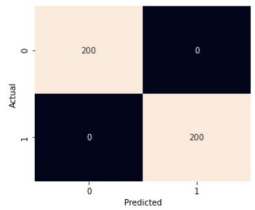


(a) Training data

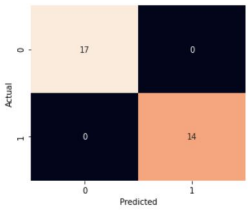


(b) Test data

Figure 1.1.3: Confusion matrix for the classification between class 0 vs class 3 of dataset 1a using Perceptron

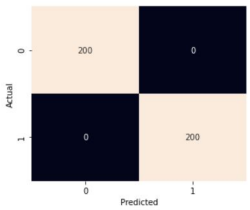


(a) Training data

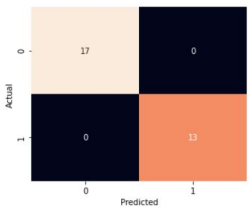


(b) Test data

Figure 1.1.4: Confusion matrix for the classification between class 1 vs class 2 of dataset 1a using Perceptron

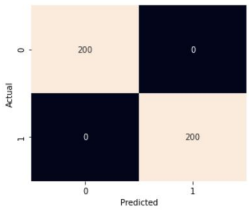


(a) Training data

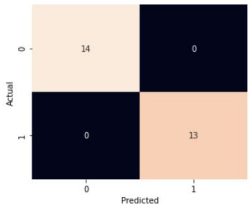


(b) Test data

Figure 1.1.5: Confusion matrix for the classification between class 1 vs class 3 of dataset 1a using Perceptron

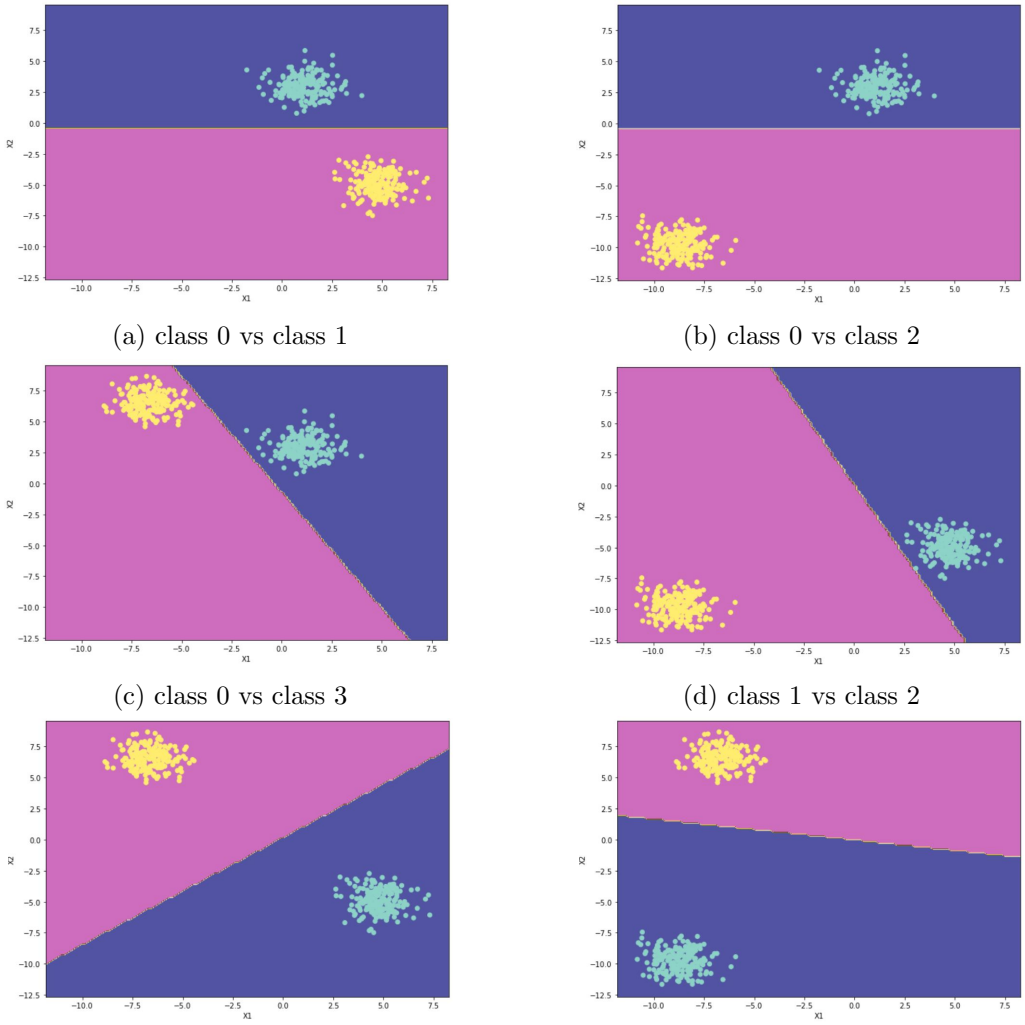


(a) Training data



(b) Test data

Figure 1.1.6: Confusion matrix for the classification between class 2 vs class 3 of dataset 1a using Perceptron



(a) class 0 vs class 1

(b) class 0 vs class 2

(c) class 0 vs class 3

(d) class 1 vs class 2

(e) class 1 vs class 3

(f) class 2 vs class 3

Figure 1.1.7: Decision region plots between each pair of classes of dataset 1a using Perceptron

We observe that since the classes are well separated we are clearly able to classify them using perceptron with very high accuracy.

1.2 MLFFNN with single hidden layer for all classes

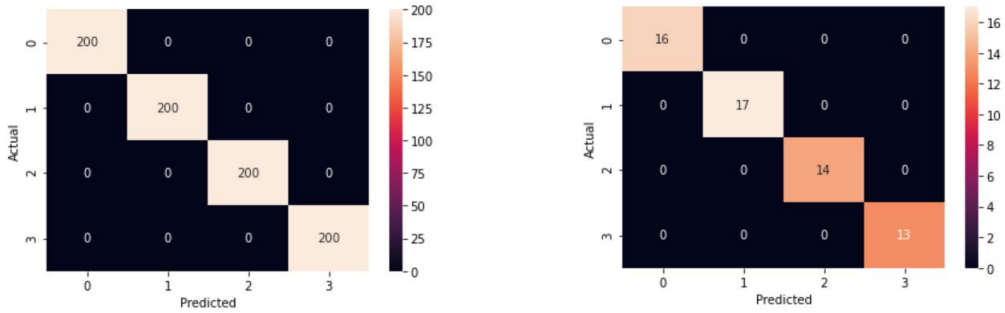
Hyperparameter no. of nodes in the hidden layer = $\{3, 4, 5\}$

Since the classes are linearly well separated we got the same accuracies for each value of number of nodes. We used a learning rate $\eta = 0.001$, $\alpha = 1$.

# of nodes	Training Dataset	Validation Dataset	Test Dataset
3	100	100	100
4	100	100	100
5	100	100	100

Table 1.2.1: Classification accuracy using MLFFNN on Dataset 1a

As we can see from the table above the best performance that we got from using MLFFNN was 100% with 3 nodes in the hidden layer (best accuracy for the least number of nodes used).



(a) Training data (b) Test data

Figure 1.2.1: Confusion matrix for the classification of dataset 1a using mlffnn with 3 nodes in hidden layer

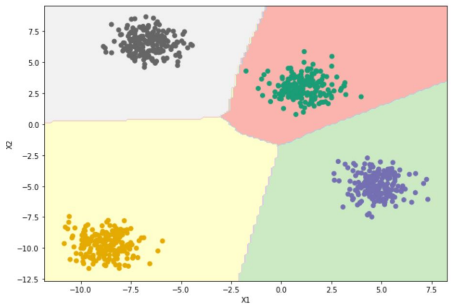


Figure 1.2.2: Decision surfaces for plots of dataset 1a using mlffnn with 3 nodes in hidden layer

1.3 Linear SVM classifier for every pair of class

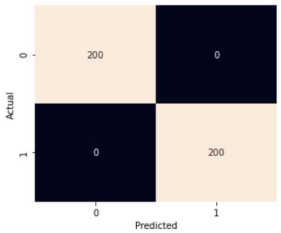
Hyperparameter $C = \{0.1, 1\}$

Since the classes are linearly well separated we got the same accuracies for each value of C.

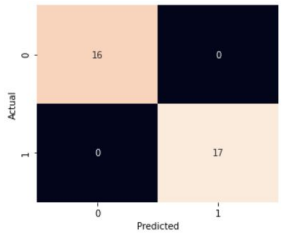
classes	Training Dataset	Validation Dataset	Test Dataset
0 vs 1	100	100	100
0 vs 2	100	100	100
0 vs 3	100	100	100
1 vs 2	100	100	100
1 vs 3	100	100	100
2 vs 3	100	100	100

Table 1.3.1: Classification accuracy using linear SVM on Dataset 1a

As we can see from the table above the model with the best performance on the test data is with $\eta = 1$ with an accuracy of 100% for all pairs of classes .

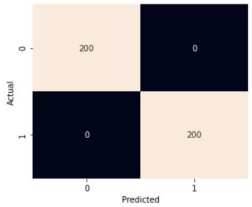


(a) Training data

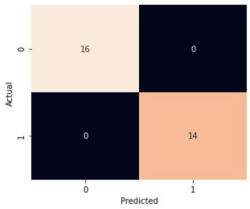


(b) Test data

Figure 1.3.1: Confusion matrix for the classification between class 0 vs class 1 of dataset 1a using linear SVM

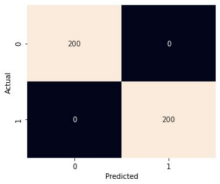


(a) Training data

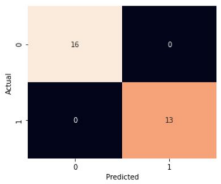


(b) Test data

Figure 1.3.2: Confusion matrix for the classification between class 0 vs class 2 of dataset 1a using linear SVM

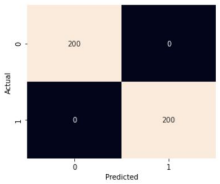


(a) Training data

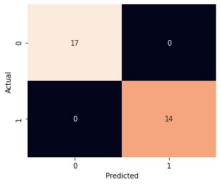


(b) Test data

Figure 1.3.3: Confusion matrix for the classification between class 0 vs class 3 of dataset 1a using linear SVM

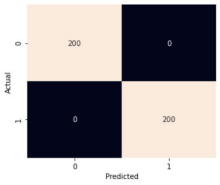


(a) Training data

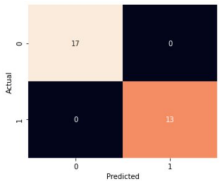


(b) Test data

Figure 1.3.4: Confusion matrix for the classification between class 1 vs class 2 of dataset 1a using linear SVM

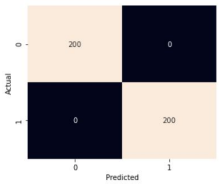


(a) Training data

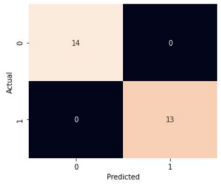


(b) Test data

Figure 1.3.5: Confusion matrix for the classification between class 1 vs class 3 of dataset 1a using linear SVM

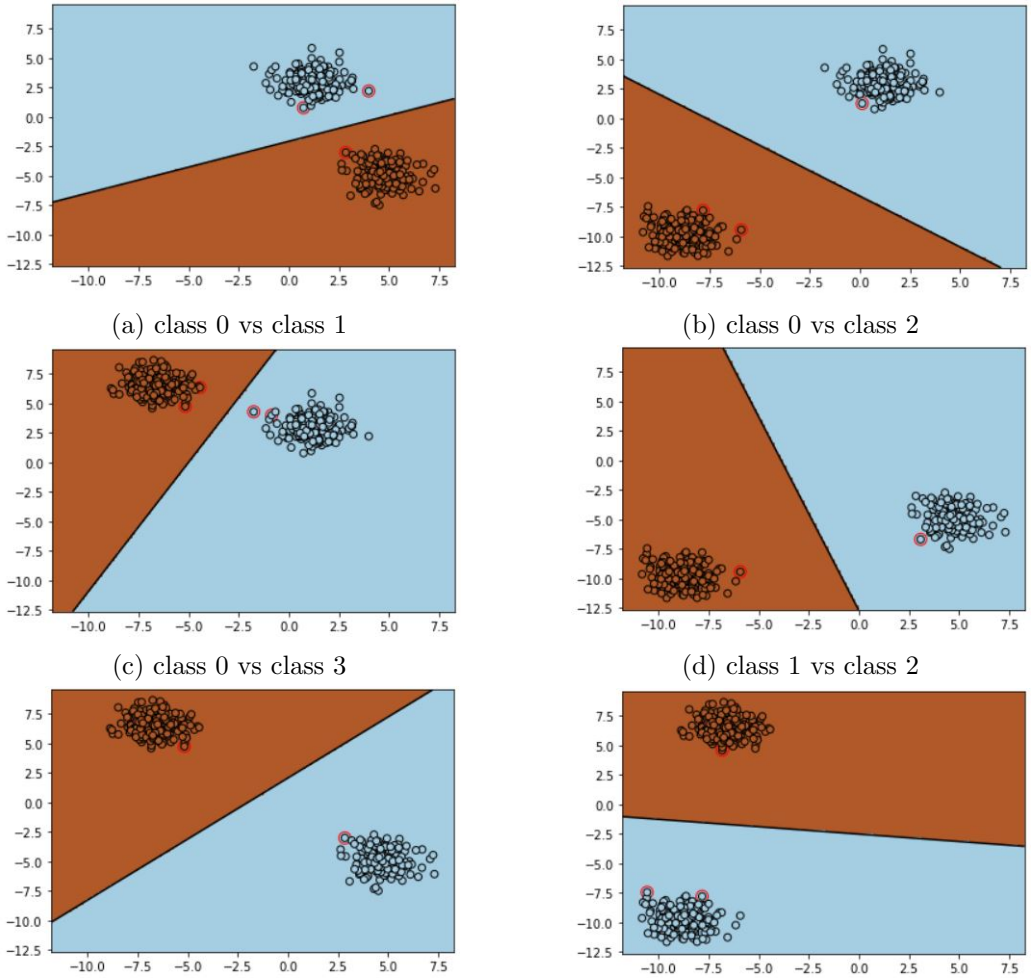


(a) Training data



(b) Test data

Figure 1.3.6: Confusion matrix for the classification between class 2 vs class 3 of dataset 1a using linear SVM



(a) class 0 vs class 1

(b) class 0 vs class 2

(c) class 0 vs class 3

(d) class 1 vs class 2

(e) class 1 vs class 3

(f) class 2 vs class 3

Figure 1.3.7: Decision region plots between each pair of classes of dataset 1a using linear SVM

Since the classes are well separated we get pretty good classification using linear SVMs and the decision surface is quite away from the data points as well.

2 Dataset 1b

2.1 MLFFNN with 2 hidden layers

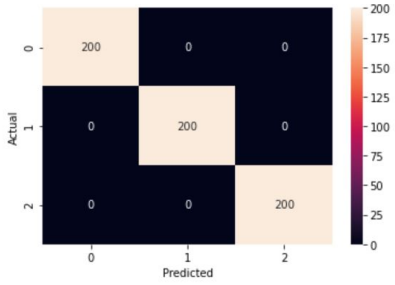
Hyperparameters : number of nodes in the hidden layers = $\{(4,2),(5,4),(6,4),(7,6)\}$

We used a learning rate $\eta = 0.001$, $\alpha = 1$.

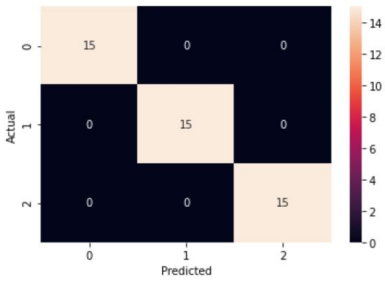
#nodes(layer 1)	#nodes(layer 2)	Training Dataset	Validation Dataset	Test Dataset
4	2	36.56	33.33	24.44
5	4	93.5	93.33	88.88
6	4	98.83	97.77	100
7	6	100	100	100

Table 2.1.1: Classification accuracy of MLFFNN with 2 hidden layer on Dataset 1b

As we can see from the table above the model had the best classification accuracy using MLFFNN was 100% when the number of nodes in the hidden layers were $7, 6$ respectively.



(a) Training data



(b) Test data

Figure 2.1.1: Confusion matrix for datatset 1b using MLFFNN with 7 nodes in Hidden layer 1 and 6 nodes in Hidden layer 2

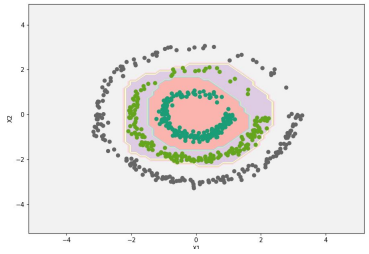
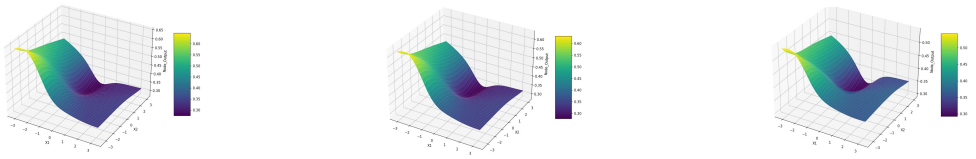


Figure 2.1.2: Decision surfaces for plots of dataset 1b using mlffnn

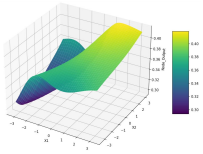
2.1.1 Output layer outputs



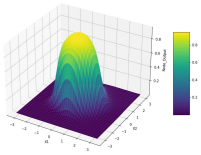
(a) node 1, epoch 1

(b) node 1, epoch 5

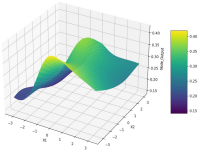
(c) node 1, epoch 20



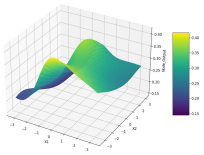
(d) node 1, epoch 100



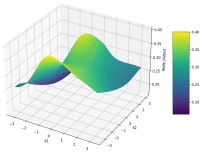
(e) converged node 1



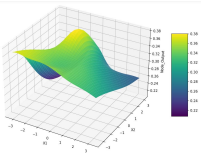
(f) node 2, epoch 1



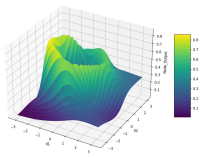
(g) node 2, epoch 5



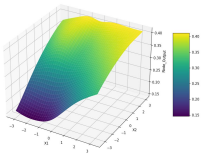
(h) node 2, epoch 20



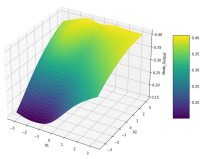
(i) node 2, epoch 100



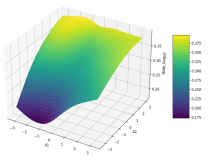
(j) converged node 2



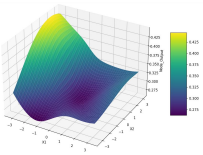
(k) node 3, epoch 1



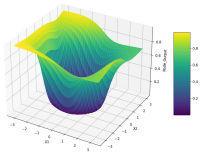
(l) node 3, epoch 5



(m) node 3, epoch 20



(n) node 3, epoch 100

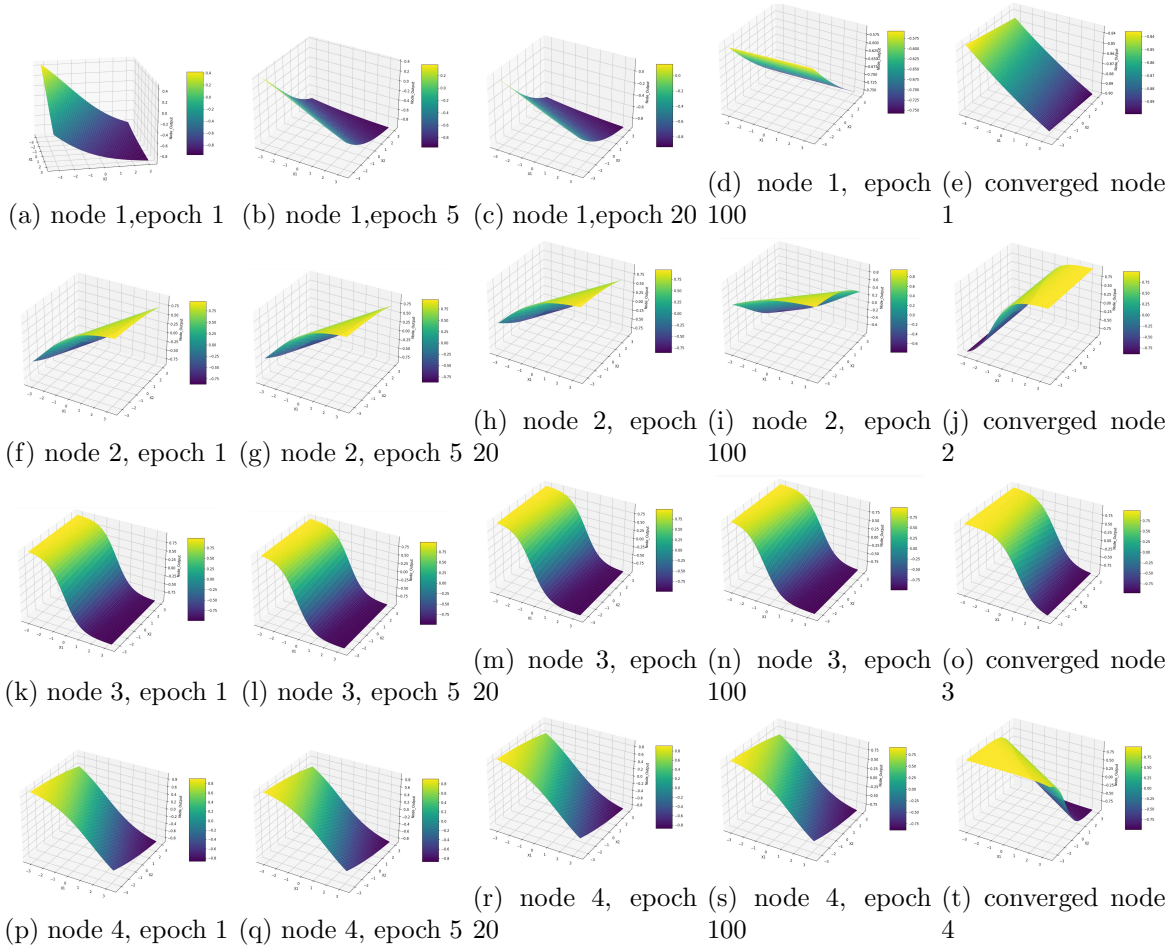


(o) converged node 3

Figure 2.1.3: Outputs of the output nodes for datatset 1b using MLFFNN with 7 nodes in Hidden layer 1 and 6 nodes in Hidden layer 2

We can see the output of each of the node in the output layer through different number of Epochs completed during the classification done using MLFFNN with 2 hidden layers having $7, 6$ nodes respectively.

2.1.2 Hidden Layer 1 outputs



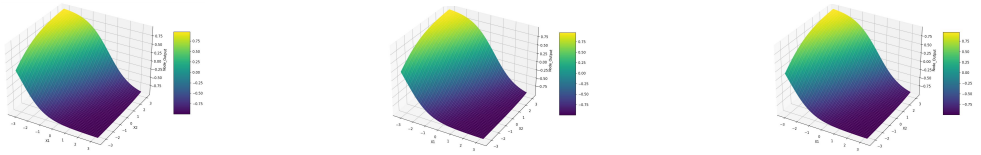
(a) node 1,epoch 1 (b) node 1,epoch 5 (c) node 1,epoch 20 (d) node 1, epoch 100 (e) converged node 1

(f) node 2, epoch 1 (g) node 2, epoch 5 (h) node 2, epoch 20 (i) node 2, epoch 100 (j) converged node 2

(k) node 3, epoch 1 (l) node 3, epoch 5 (m) node 3, epoch 20 (n) node 3, epoch 100 (o) converged node 3

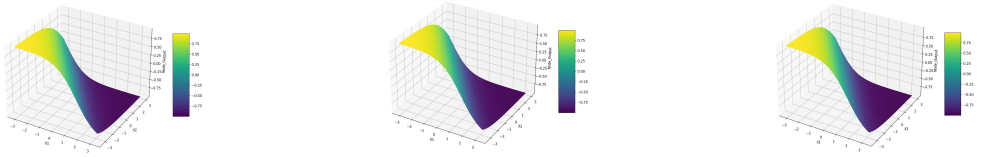
(p) node 4, epoch 1 (q) node 4, epoch 5 (r) node 4, epoch 20 (s) node 4, epoch 100 (t) converged node 4

Figure 2.1.4: Outputs of the hidden layer 1 nodes (1-4) for datatset 1b using MLFFNN with 7 nodes in Hidden layer 1 and 6 nodes in Hidden layer 2



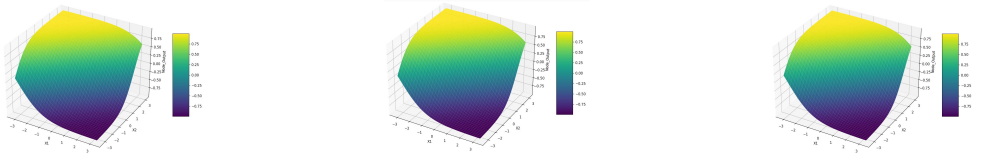
(a) node 5, epoch 1 (b) node 5, epoch 5 (c) node 5, epoch 20

(d) node 5, epoch 100 (e) converged node 5



(f) node 6, epoch 1 (g) node 6, epoch 5 (h) node 6, epoch 20

(i) node 6, epoch 100 (j) converged node 6

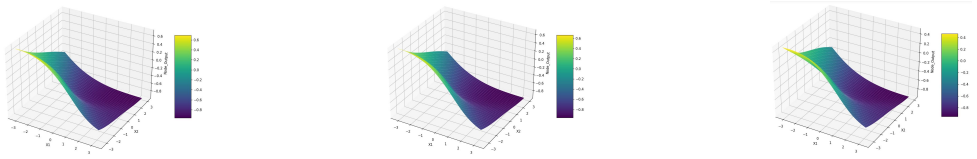


(k) node 7, epoch 1 (l) node 7, epoch 5 (m) node 7, epoch 20

(n) node 7,epoch 100 (o) converged node 7

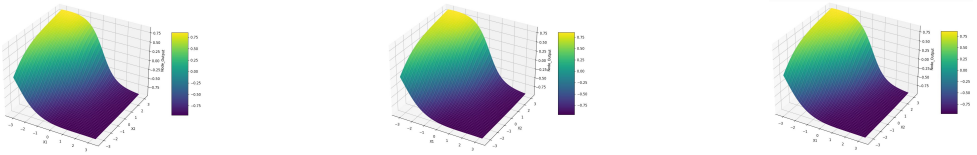
Figure 2.1.5: Outputs of the hidden layer 1 nodes(5-7) for datatset 1b using MLFFNN with 7 nodes in Hidden layer 1 and 6 nodes in Hidden layer 2

2.1.3 Hidden layer 2 outputs



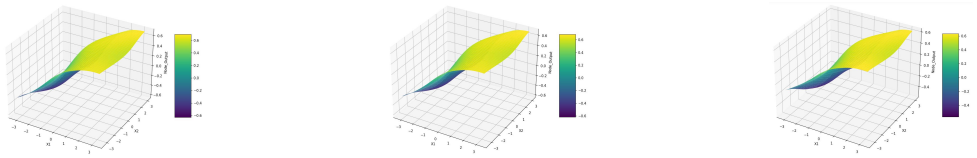
(a) node 1, epoch 1 (b) node 1, epoch 5 (c) node 1, epoch 20

(d) node 1, epoch 100 (e) converged node 1



(f) node 2, epoch 1 (g) node 2, epoch 5 (h) node 2, epoch 20

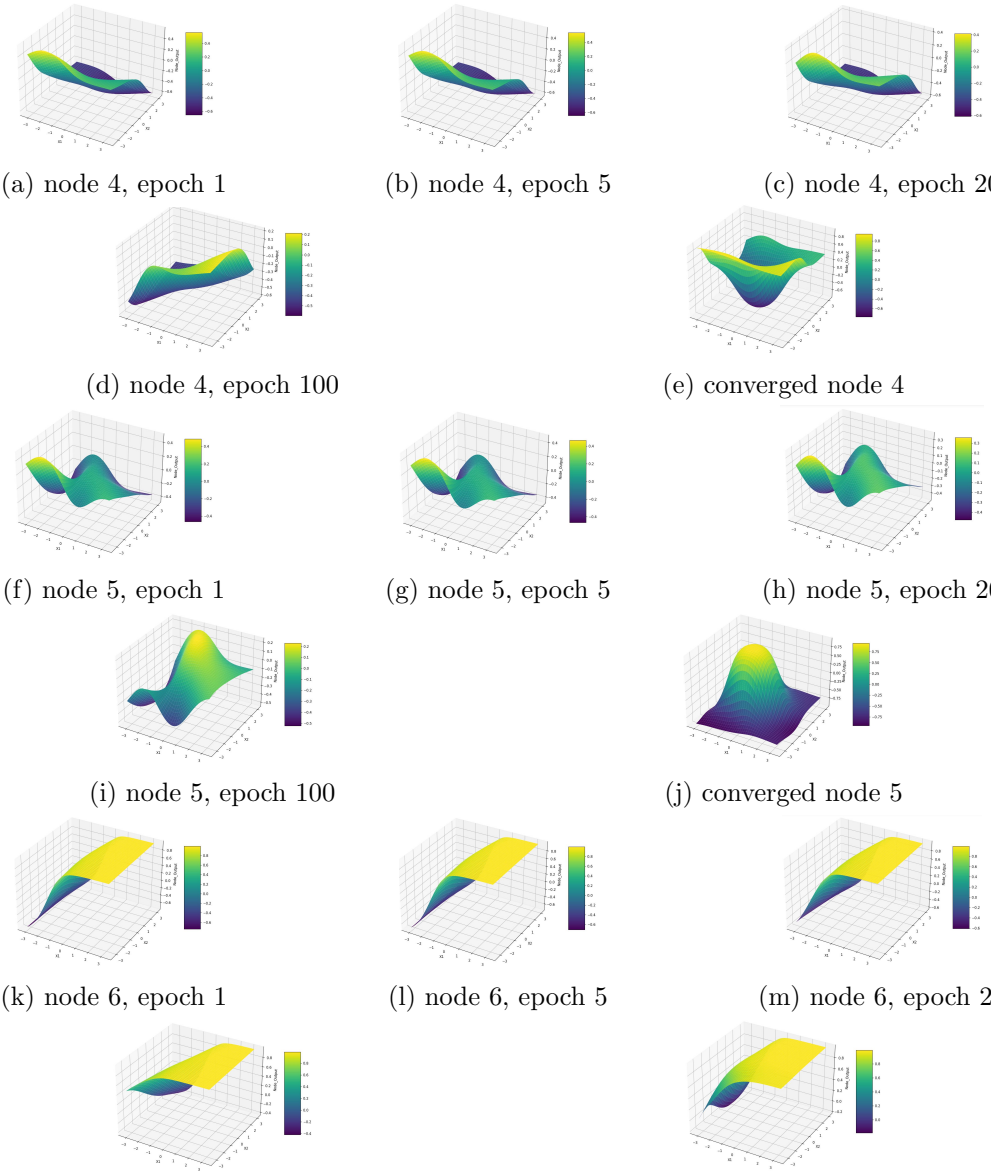
(i) node 2, epoch 100 (j) converged node 2



(k) node 3, epoch 1 (l) node 3, epoch 5 (m) node 3, epoch 20

(n) node 3,epoch 100 (o) converged node 3

Figure 2.1.6: Outputs of the hidden layer 2 nodes(1-3) for datatset 1b using MLFFNN with 7 nodes in Hidden layer 1 and 6 nodes in Hidden layer 2



(a) node 4, epoch 1 (b) node 4, epoch 5 (c) node 4, epoch 20

(d) node 4, epoch 100 (e) converged node 4

(f) node 5, epoch 1 (g) node 5, epoch 5 (h) node 5, epoch 20

(i) node 5, epoch 100 (j) converged node 5

(k) node 6, epoch 1 (l) node 6, epoch 5 (m) node 6, epoch 20

(n) node 6,epoch 100 (o) converged node 6

Figure 2.1.7: Outputs of the hidden layer 2 nodes(4-6) for datatset 1b using MLFFNN with 7 nodes in Hidden layer 1 and 6 nodes in Hidden layer 2

2.2 Non-linear SVM using 1 vs rest, Polynomial kernel

We used $C = 1$ and $degree = 4$ for classification using polynomial kernel in SVM. We also used the parameter to balance the number of training datapoints from both the classes.

classes	Training Dataset	Validation Dataset	Test Dataset
class 0 vs rest	100	100	100
class 1 vs rest	66.67	66.67	66.67
class 2 vs rest	100	100	100

Table 2.2.1: Classification accuracy of polynomial SVM classifier on Dataset 1b

As we can see from the table above the model with the best performance on the test data has an accuracy of 100%, 66.67%, 100% for class 0 vs rest, class 1 vs rest and class 2 vs rest respectively.

(a) Training data (b) Test data

Figure 2.2.1: Confusion matrix for class 0 vs rest in datatset 1b using polynomial SVM

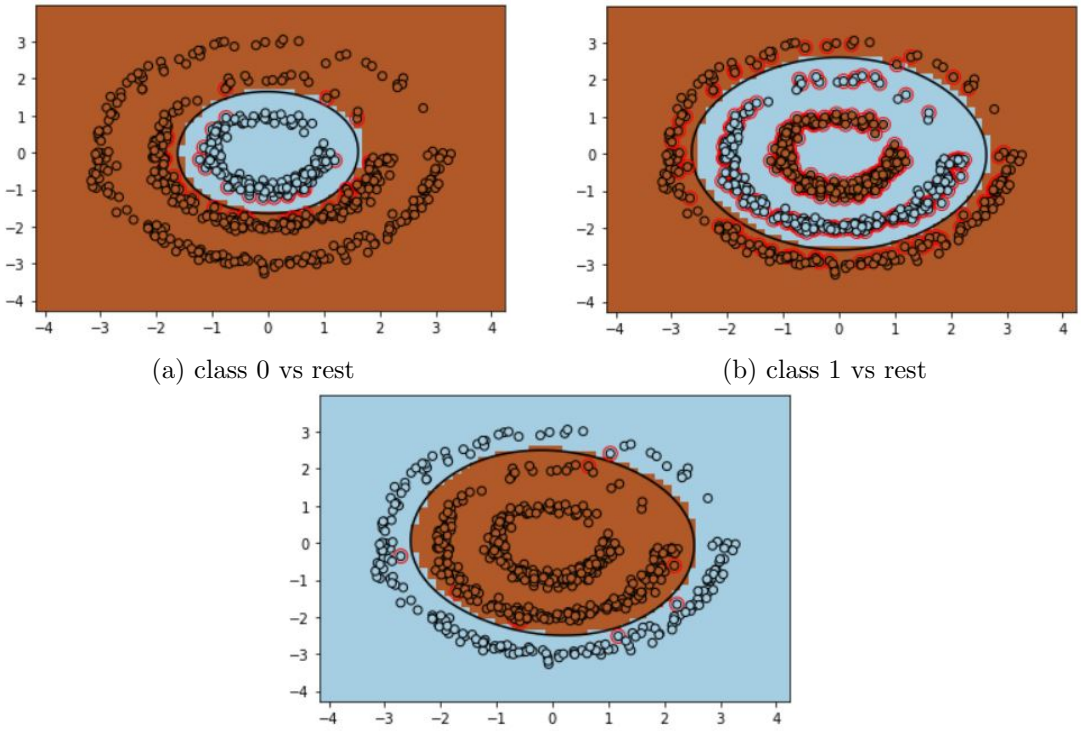
(a) Training data (b) Test data

Figure 2.2.2: Confusion matrix for class 1 vs rest in datatset 1b using polynomial SVM

Our classification accuracy for class 1 vs rest is not high when we used polynomial kernel in classification using SVM for any values of the hyperparameters, because class 1 is surrounded by points that don't belong to it from both inside and outside. It is quite difficult to come up with a polynomial which is able to linearly separate class 1 from the other two classes.

(a) Training data (b) Test data

Figure 2.2.3: Confusion matrix for class 2 vs rest in datatset 1b using polynomial SVM



(a) class 0 vs rest (b) class 1 vs rest

(c) class 2 vs rest

Figure 2.2.4: Decision region plots for 1 class vs frest in datatset 1b using polynomial SVM

As explained before we can observe that in the decision region plots that class 0, class 2 are easily separated from the other classes but class 1 still has some points which are on the wrong side of the separating curve.

2.3 Non-linear SVM using 1 vs rest, Gaussian kernel

Hyperparameters : $C = \{0.1, 1\}$
We used for classification using rbf kernel in SVM. We also used the parameter to balance the number of training datapoints from both the classes.
We got similar accuracy for both the values of C.

classes	**Training Dataset**	**Validation Dataset**	**Test Dataset**
class 0 vs rest	100	100	100
class 1 vs rest	99.83	100	100
class 2 vs rest	100	100	100

Table 2.3.1: Classification accuracy of Gaussian SVM classifier on Dataset 1b

As we can see from the table above the model with the best performance on the test data for $C = 1$ with an accuracy of 100% for all the classes.

(a) Training data (b) Test data

Figure 2.3.1: Confusion matrix for class 0 vs rest in datatset 1b using Gaussian SVM

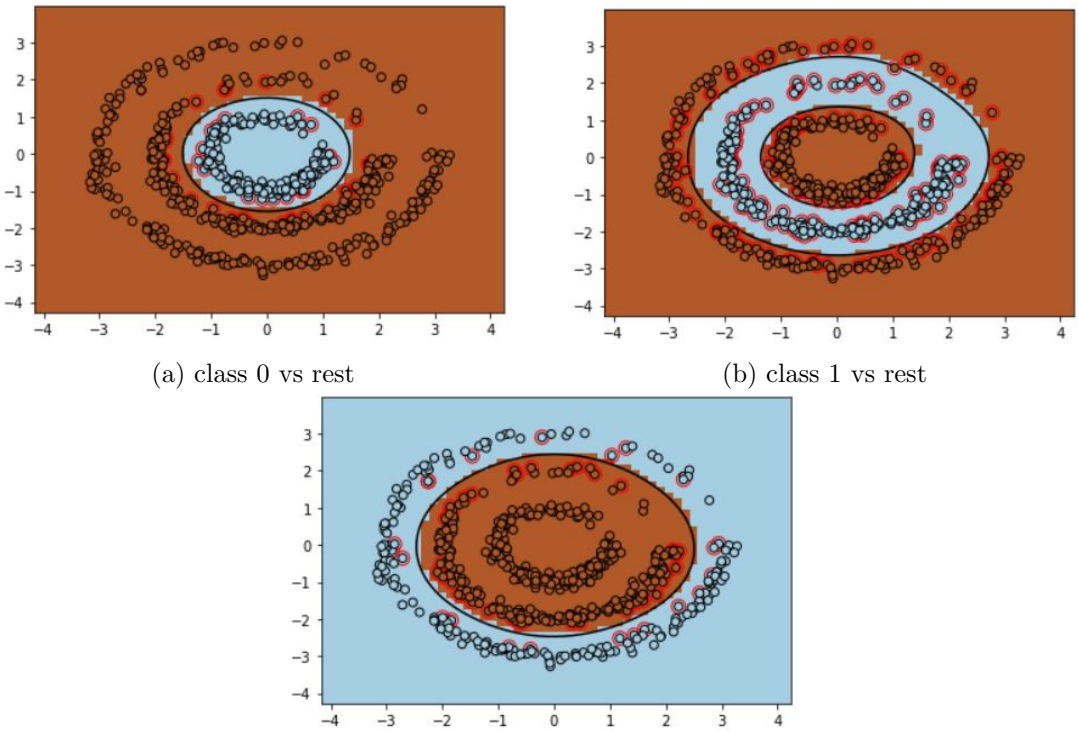
(a) Training data (b) Test data

Figure 2.3.2: Confusion matrix for class 1 vs rest in datatset 1b using Gaussian SVM

Unlike the polynomial kernel, Gaussian kernel is able to separate class 1 from the other 2 classes with pretty good accuracy.

(a) Training data (b) Test data

Figure 2.3.3: Confusion matrix for class 2 vs rest in datatset 1b using Gaussian SVM



(a) class 0 vs rest (b) class 1 vs rest

(c) class 2 vs rest

Figure 2.3.4: Decision region plots for 1 class vs frest in datatset 1b using Gaussian SVM

3 Dataset 2

The classes we had to distinguish between were:

1. Forest
2. Highway
3. Inside city
4. Mountain
5. Street

3.1 MLFFNN with 2 hidden layers

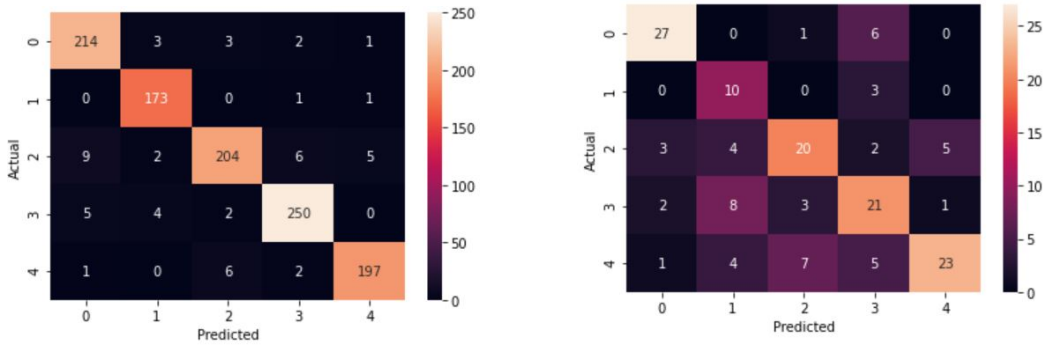
Hyperparameters : #nodes in hidden layers 1,2 = $\{(39, 20), (40, 40), (20, 10), (30, 30), (50, 40), (70, 60)\}$

We used a learning rate $\eta = 0.001$, $\alpha = 1$.

#nodes(layer 1)	#nodes(layer 2)	Training	Validation	Test
39	20	94.5	58.59	58.97
40	40	94.04	64.33	62.82
20	10	86.35	59.87	62.17
30	30	91.12	66.24	62.17
50	40	92.5	63.7	61.9
70	60	95.14	63.7	64.74

Table 3.1.1: Classification accuracy of MLFFNN with 2 hidden layers on Dataset 2

The best classification accuracy of 64.74% was observed when we used 70, 60 nodes in the hidden layers 1,2 respectively.



(a) Training data

(b) Test data

Figure 3.1.1: Confusion matrix for datatset 2 using MLFFNN with 70 nodes in Hidden layer 1 and 60 nodes in Hidden layer 2

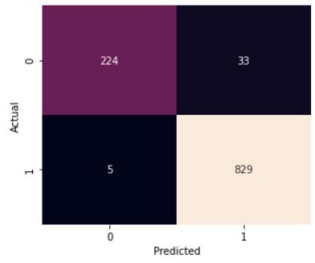
3.2 Gaussian kernel based SVM using 1 vs rest

We used $C = 1$ for classification using rbf kernel in SVM. We also used the parameter to balance the number of training datapoints from both the classes.

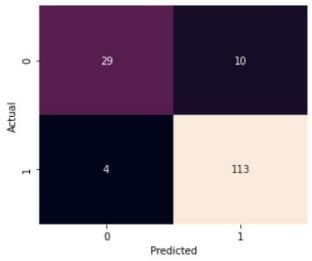
classes	Training Dataset	Validation Dataset	Test Dataset
class 0 vs rest	96.5169	93.6305	91.0256
class 1 vs rest	99.5417	90.4458	87.8205
class 2 vs rest	91.2007	80.8917	84.6153
class 3 vs rest	94.8670	82.8025	80.1282
class 4 vs rest	95.6003	82.8025	77.5641

Table 3.2.1: Classification accuracy of Gaussian SVM classifier on Dataset 2

As we can see from the table above the model with the best performance on the test data with an accuracy of $91.02\%, 87.82\%, 84.61\%, 80.12\%, 77.56\%$ for the classes 0,1,2,3,4 respectively.

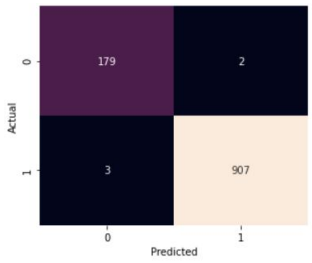


(a) Training data

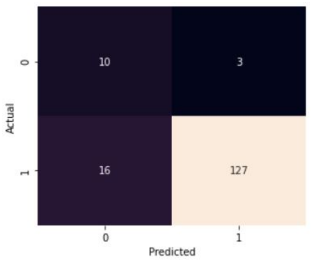


(b) Test data

Figure 3.2.1: Confusion matrix for class 0 vs rest in datatset 2 using Gaussian SVM

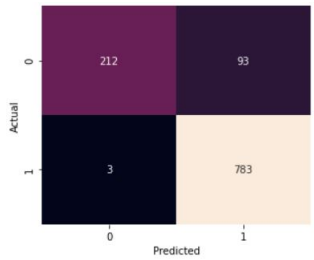


(a) Training data

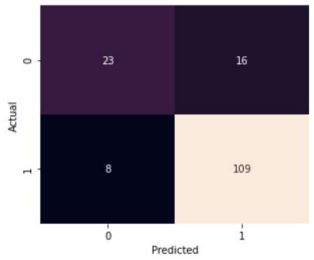


(b) Test data

Figure 3.2.2: Confusion matrix for class 1 vs rest in datatset 2 using Gaussian SVM

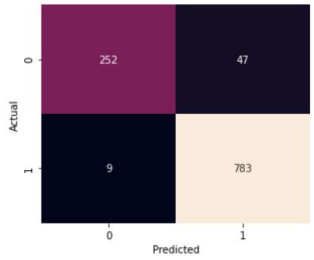


(a) Training data

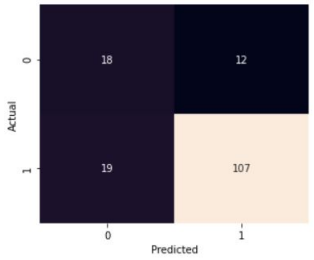


(b) Test data

Figure 3.2.3: Confusion matrix for class 2 vs rest in datatset 2 using Gaussian SVM

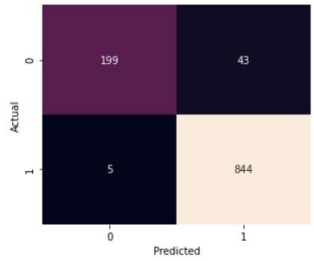


(a) Training data

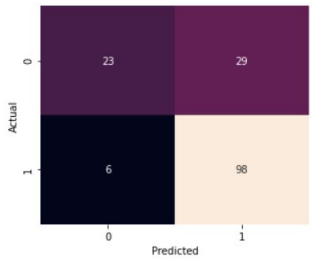


(b) Test data

Figure 3.2.4: Confusion matrix for class 3 vs rest in datatset 2 using Gaussian SVM



(a) Training data



(b) Test data

Figure 3.2.5: Confusion matrix for class 4 vs rest in datatset 2 using Gaussian SVM