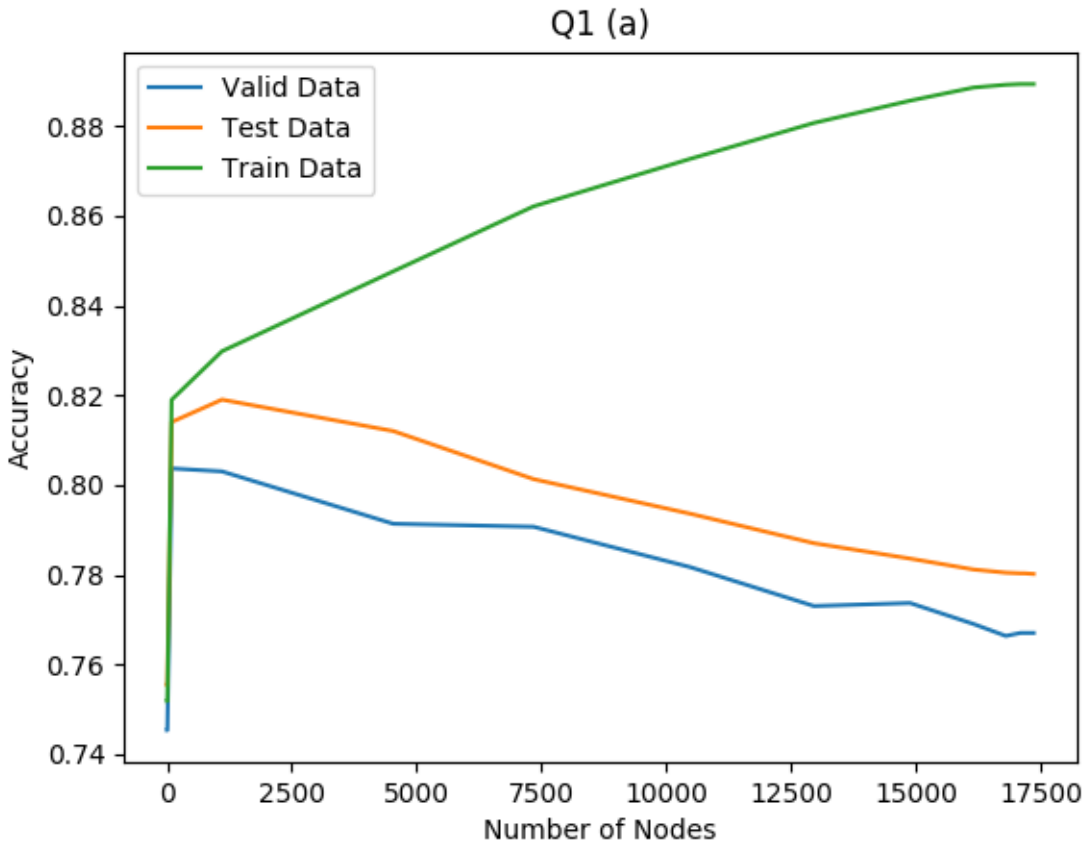
COL 774: Assignment 3

Question1

Part(a): Constructing Decision Tree

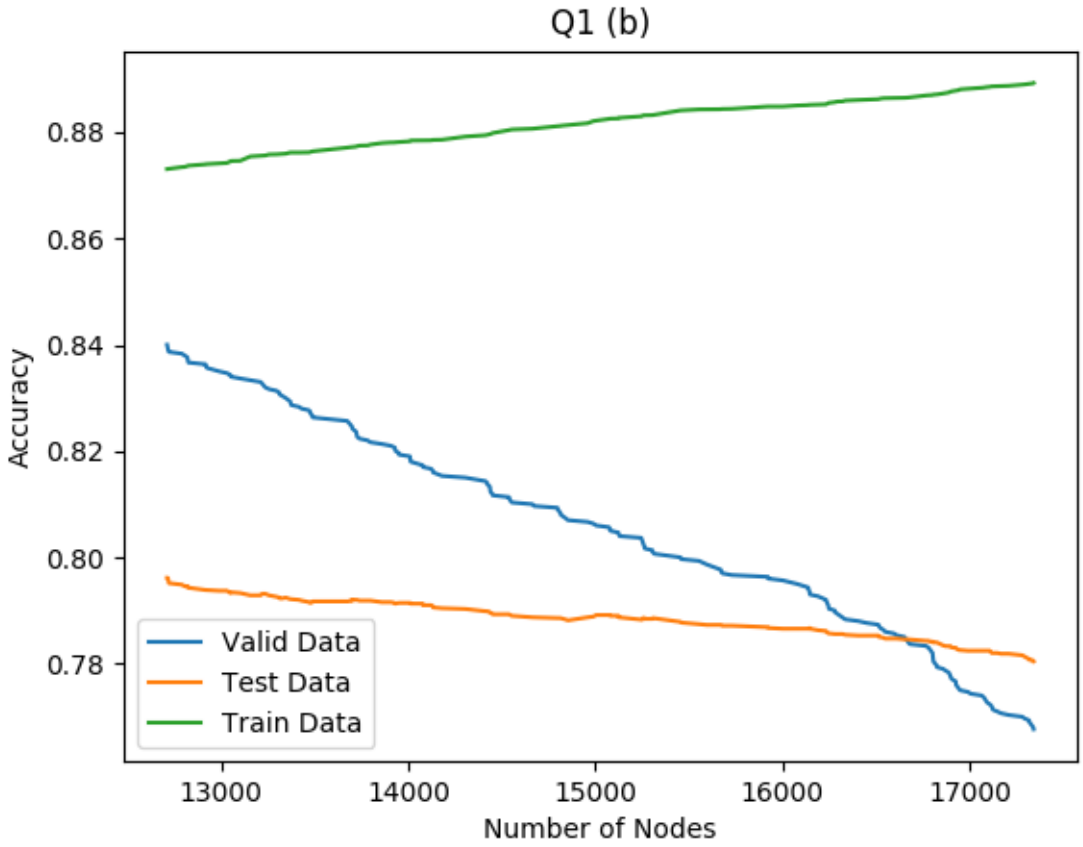
As can be seen from graphs, the test and validation accuracies increase till a certain point (different for all three). After that point the validation accuracy as well as test accuracy start decreasing while training accuracy still increases, indicating model starts over-fitting.

Before pruning:
validation accuracy=76.7%
test accuracy=78.01428571428572%
train accuracy=88.93333333333334%



Part(b): post-pruning based on a validation set

After pruning:
validation accuracy=84.0%
test accuracy=79.614%
train accuracy=87.307%

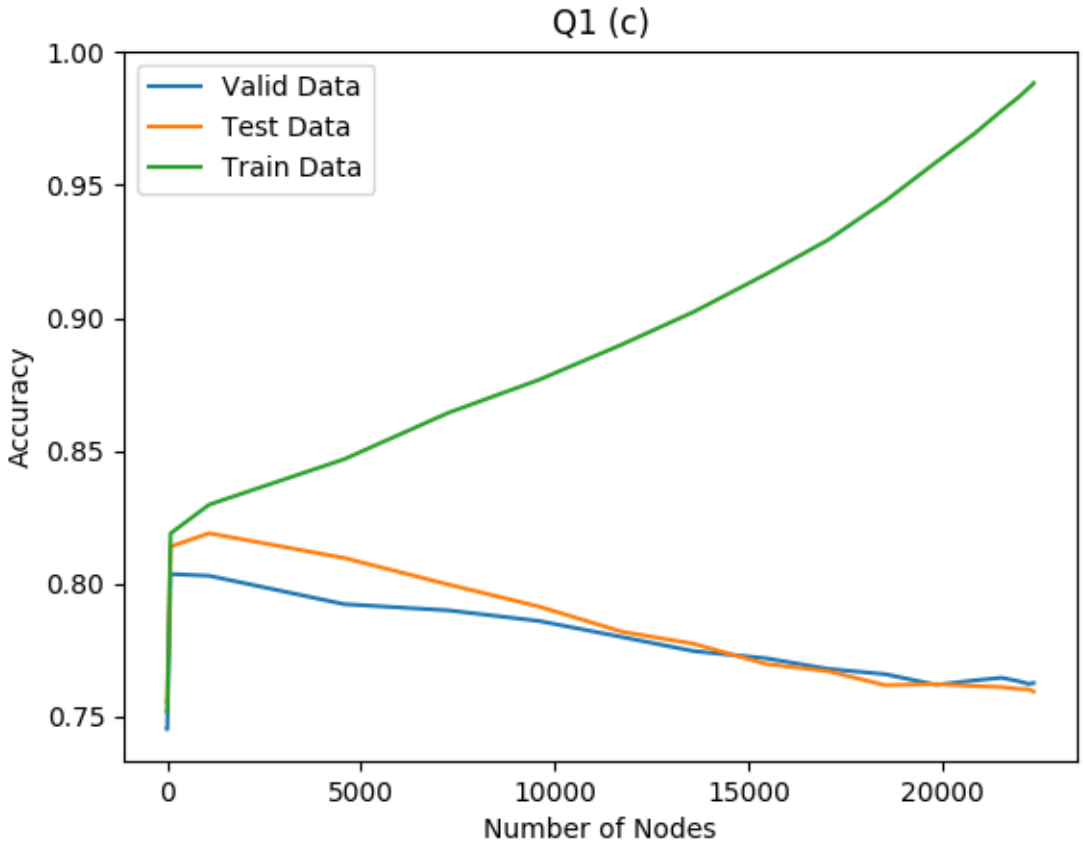


Pruning is done by removing all the nodes s.t. it leads to increase in validation accuracy.
Validation accuracy increases as the number of nodes decrease, test accuracy also increases a bit while training accuracy decreases continuously.
This trend can be explained as the number of nodes are more, the tree fits more and more perfectly on the data.
As we decrease the nodes, the validation accuracy increases (because it was the factor being considered while removing nodes).

Part(c)

Made decision tree without preprocessing:

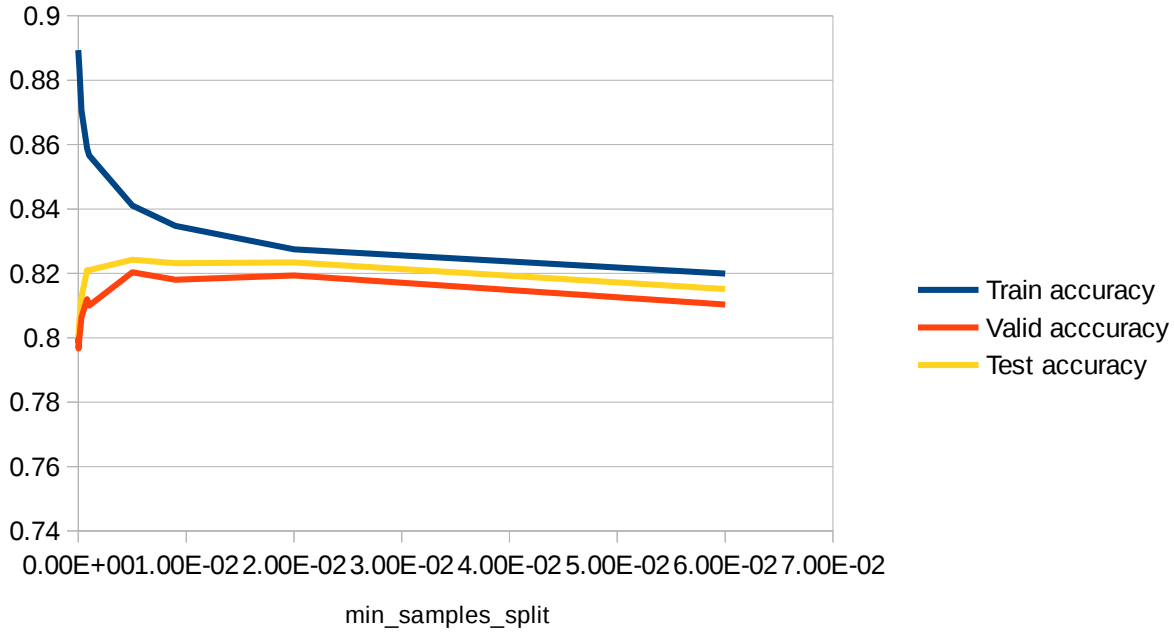
validation accuracy=76.266%
test accuracy=75.943%
train accuracy=98.829%



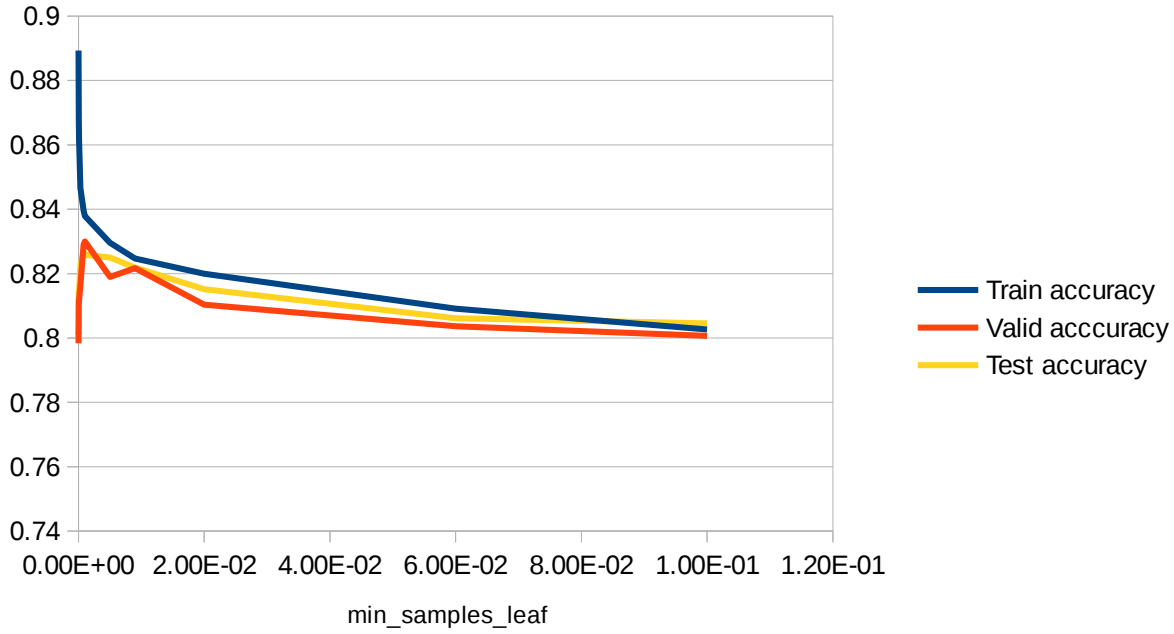
As we can see from the plot and the results that no-preprocessing model gives better accuracies on training data and similar to those as above parts for testing and validation data. This can be attributed to the fact that now the model has more options available to it to find out the best attribute to divide on.
The training accuracy reaches 98.8% with this model.
The small values of validation and test accuracy represent high over fitting in the model.

Part(d): SciKit-Learn Decision Tree

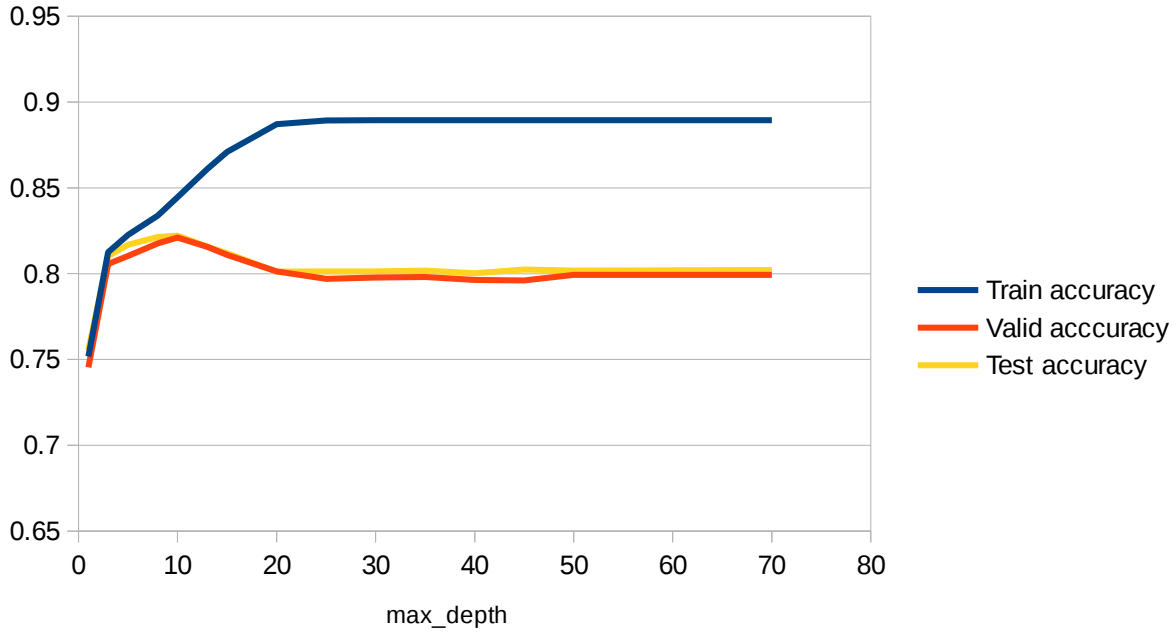
1) Varying min_samples_split



2) Varying min_samples_leaf



3)



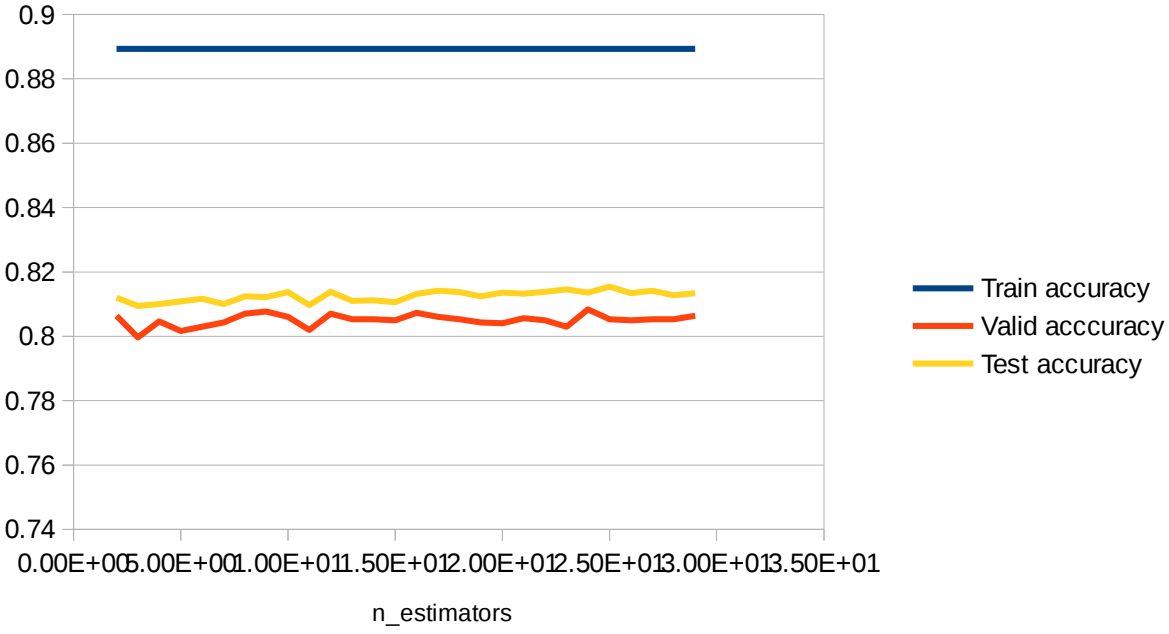
Best parameters: (min_samples_split=1e-10, min_samples_leaf=0.001, max_depth=15)

Train Accuracy= 83.7888%
Validition Accuracy= 82.9999%
Test Accuracy= 82.58%

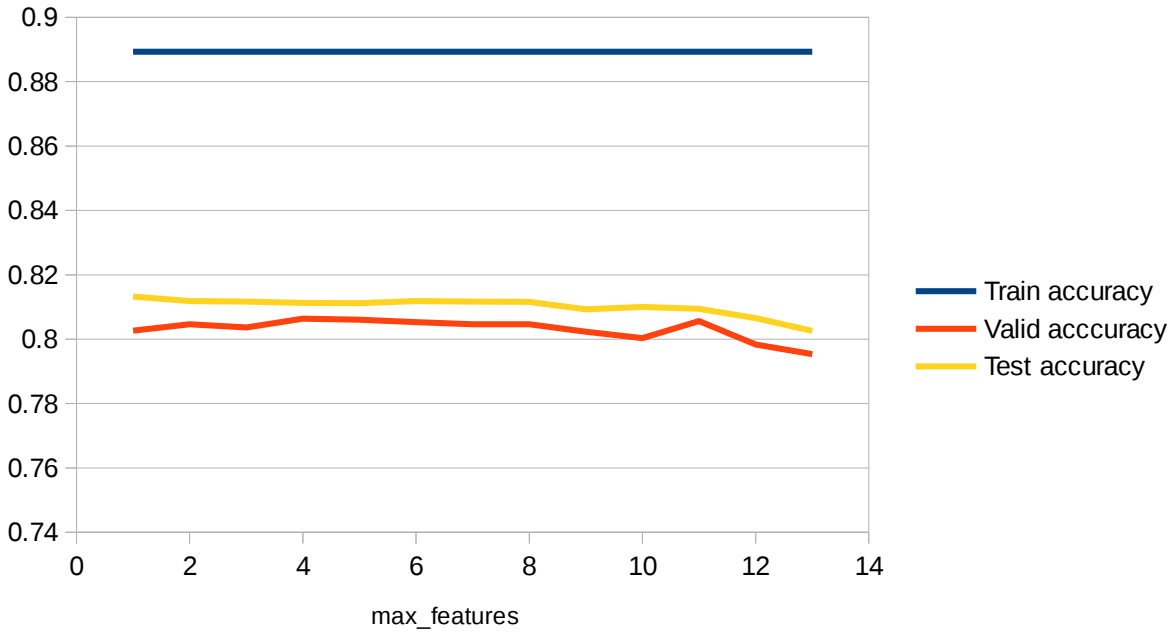
Test accuracy is more than part(b). But Train and validation accuracies are less than part(b). This shows there is over-fitting in part(b).

Part(e)

1) Varying n_estimators



2) Varying max_features



Best parameters: (max_features=11, n_estimators=12, max_depth=True)

Train Accuracy= 0.8852592592592593%
Validation Accuracy= 0.8143333333333335%
Test Accuracy= 0.81357142857142861%

Test accuracy is more than part(b) and part(c). But Train and validation accuracies are less than part(b). This shows there is over-fitting in part(b) and part(c).

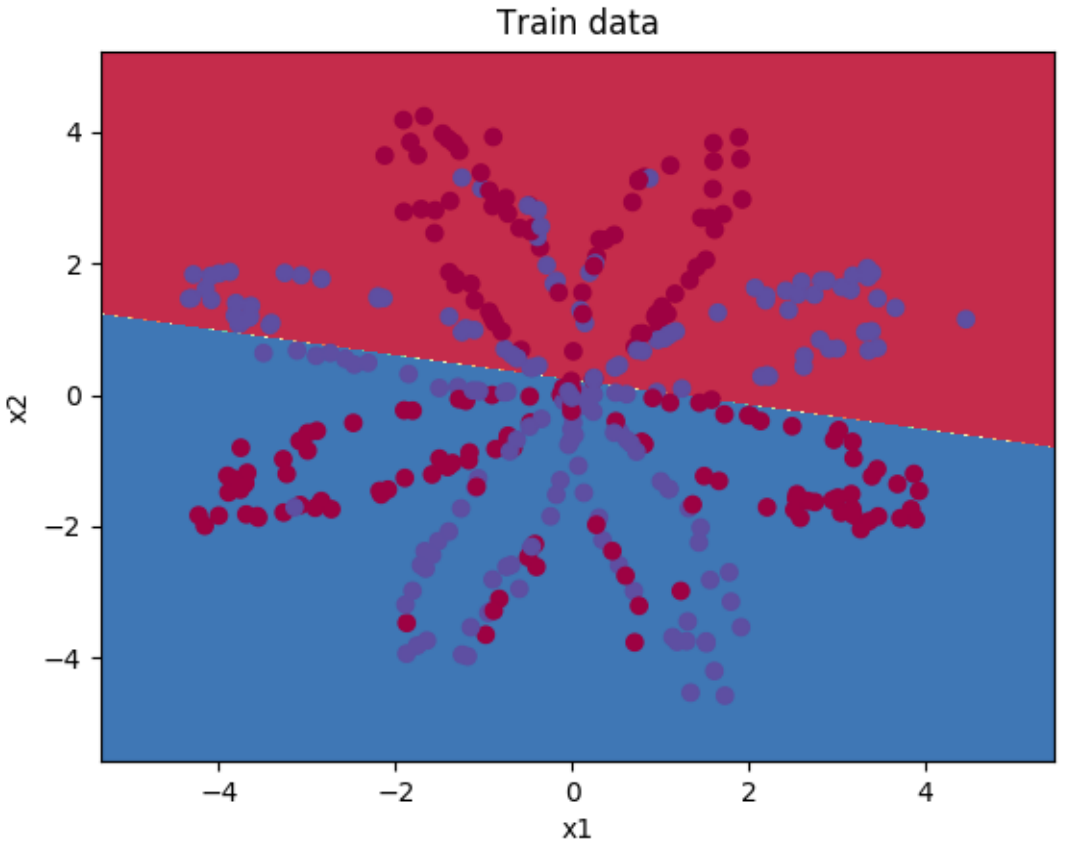
Question2

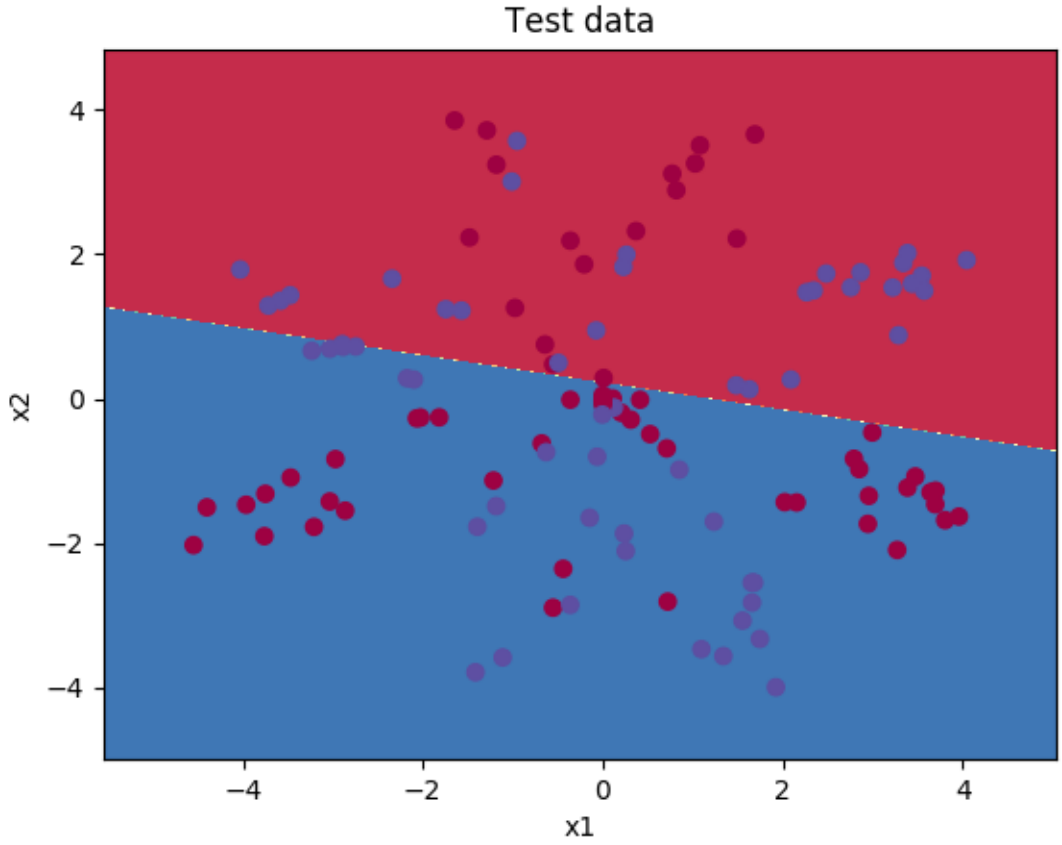
Part(a)
Implemented the generic Neural Network using sigmoid as activation function.

Part(b)

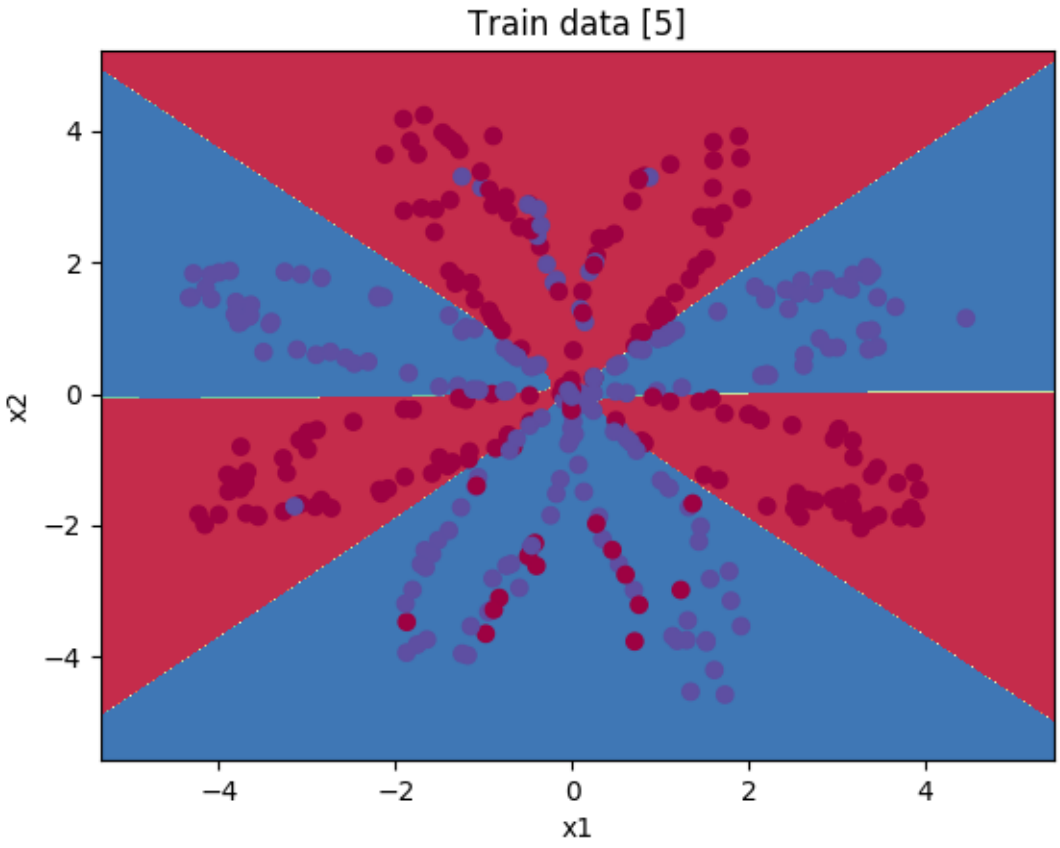
(1)SciKit-learn Logistic regression learner

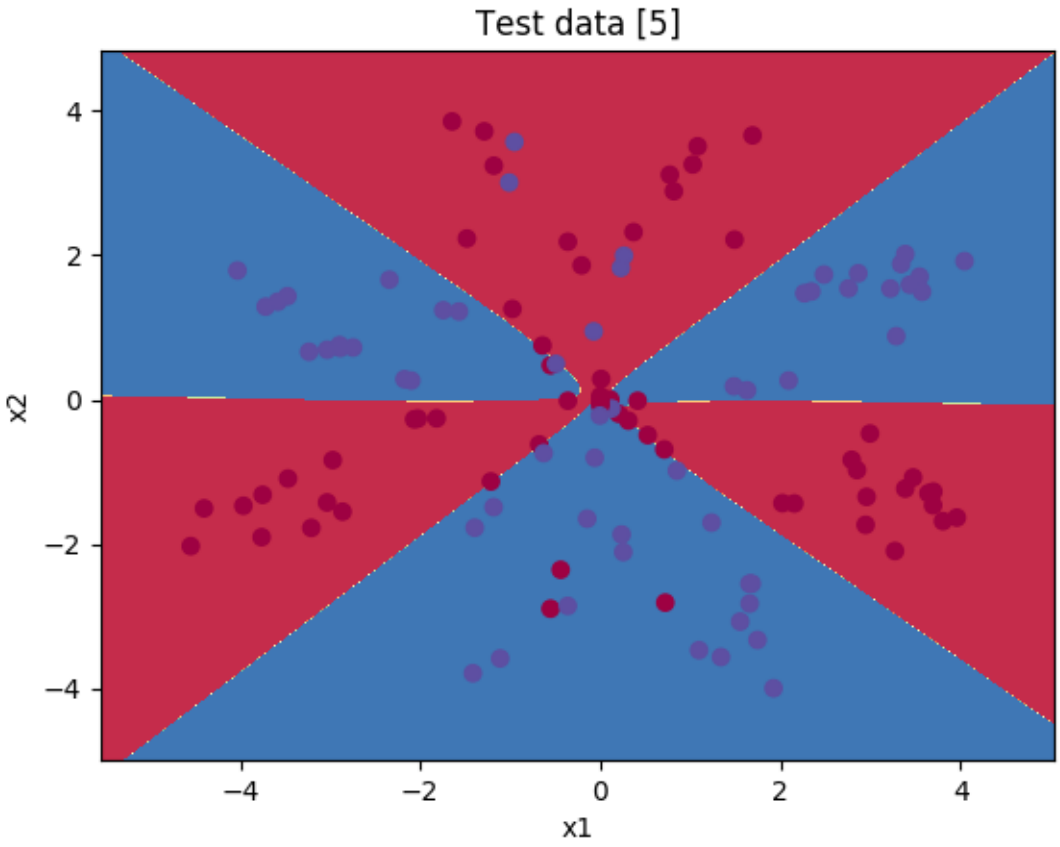
Train data accuracy=45.789%
Test data accuracy=38.333%


Train data
4
2
0
−2
−4
x2
−4
−2
0
2
4
x1


Test data
4
2
0
−2
−4
x2
−4
−2
0
2
4
x1

(2) Single hidden layer with 5 units
eta=0.08
epsilon=0.00001
Accuracy on train data=89.4736%
Accuracy on test data=87.5%

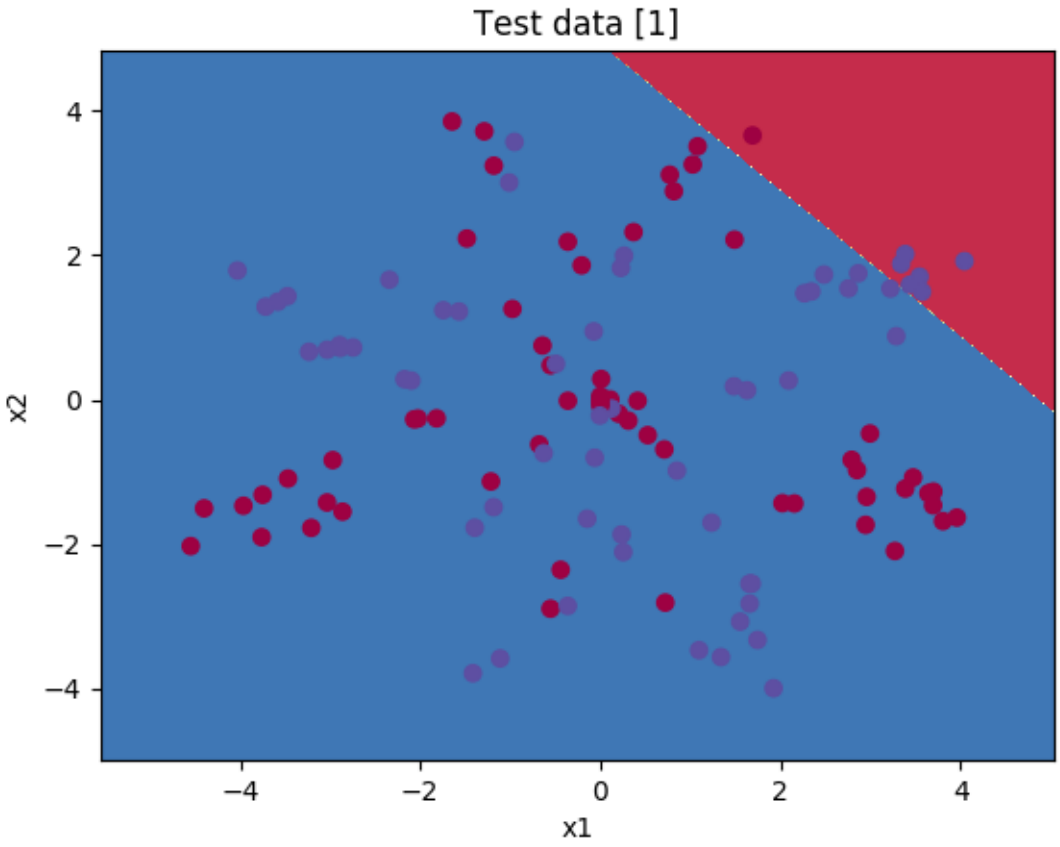




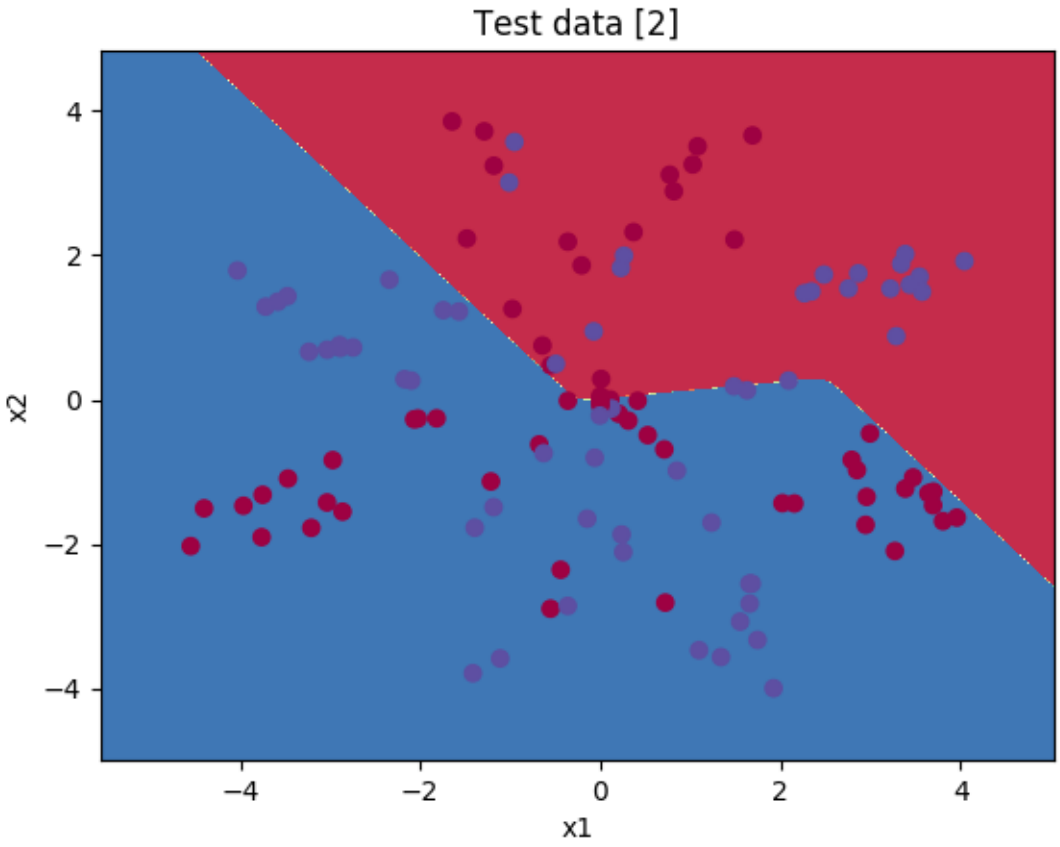
(3) Single hidden layer architecture

Varied the number of units in the hidden layer as (1,2,3,10,20,30,40).

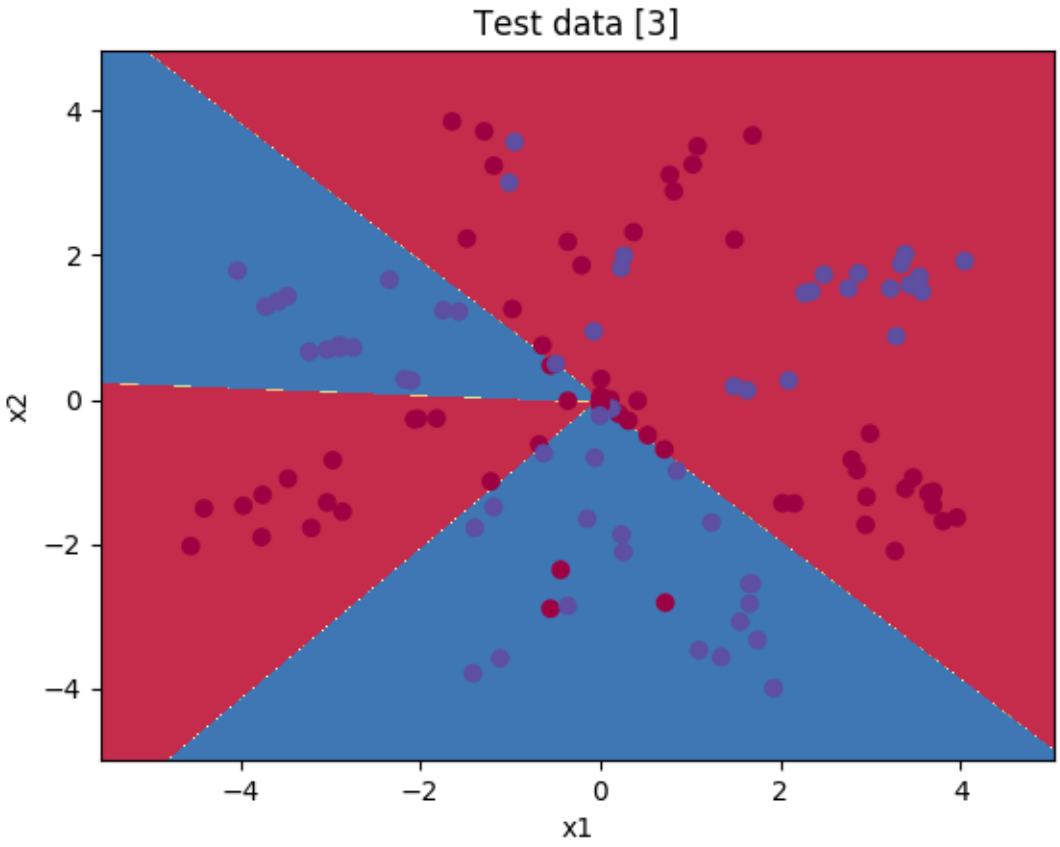
number of units in the hidden layer = 1
Accuracy on train data=50.0%
Accuracy on test data=44.166666666666664%
eta=0.24
epsilon=0.00001



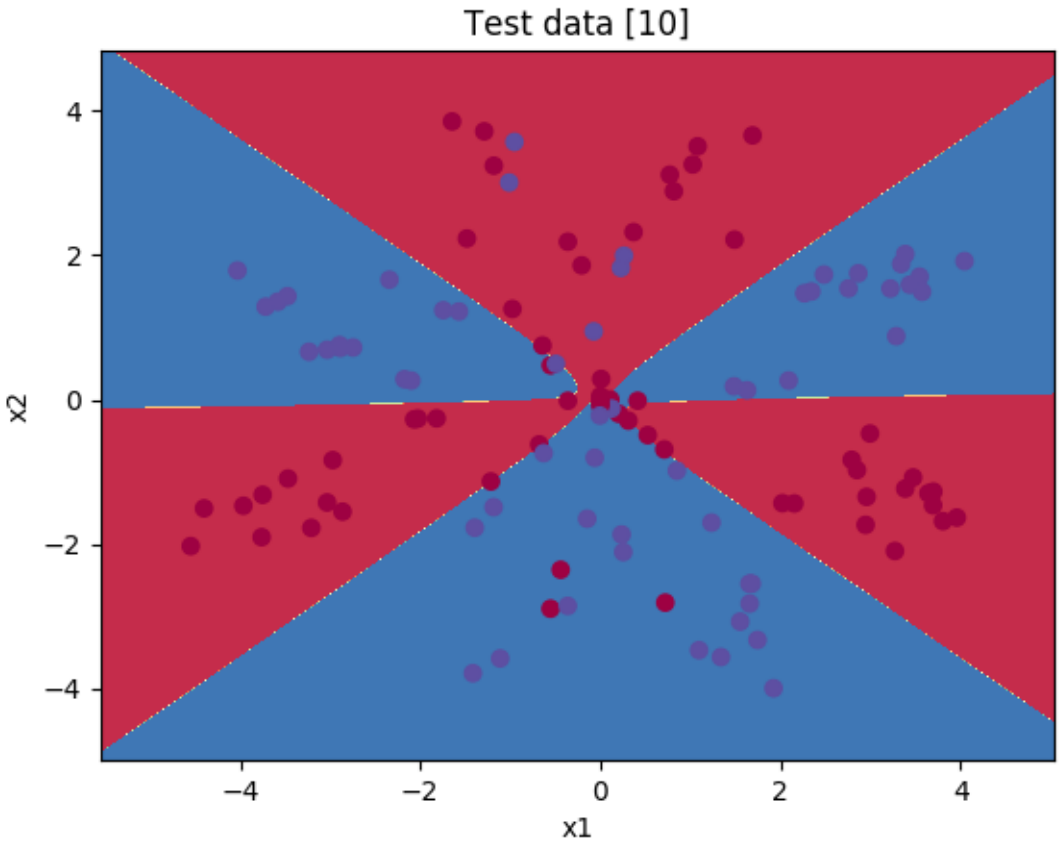
number of units in the hidden layer = 2
Accuracy on train data=57.89473684210527%
Accuracy on test data=53.333333333333336%
eta=0.2



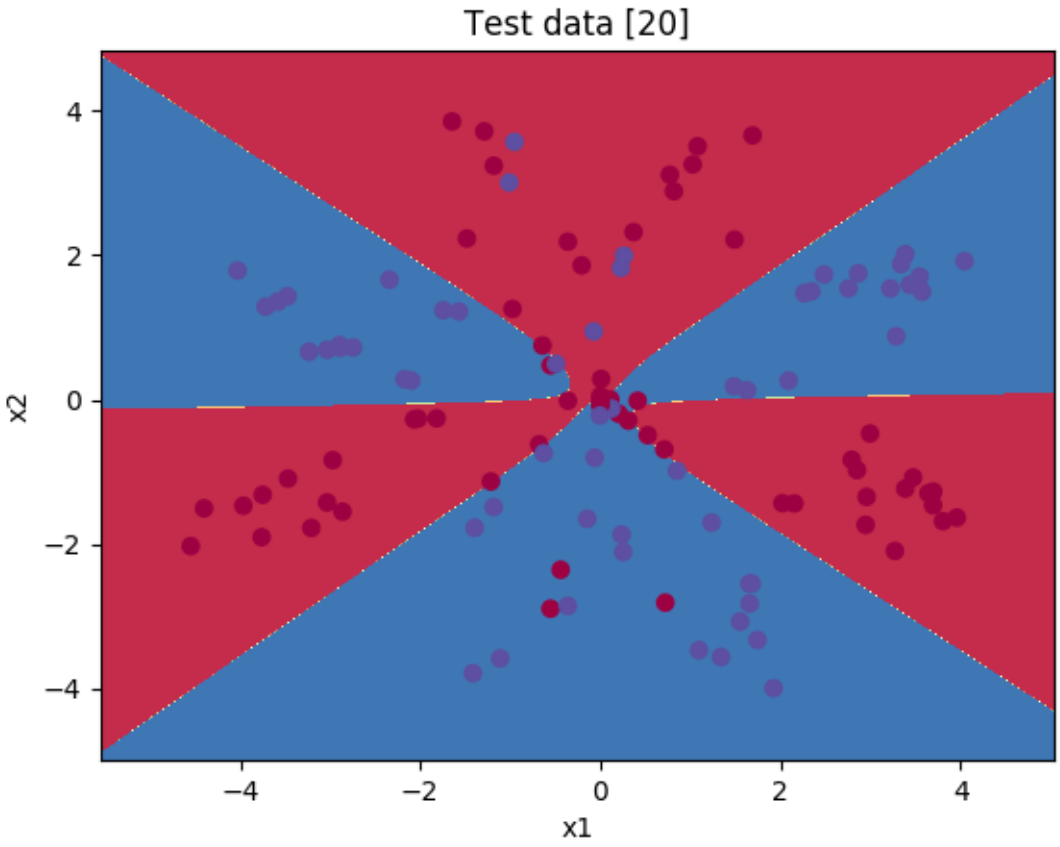
number of units in the hidden layer = 3
Accuracy on train data=75.0%
Accuracy on test data=76.66666666666667%
eta=0.17



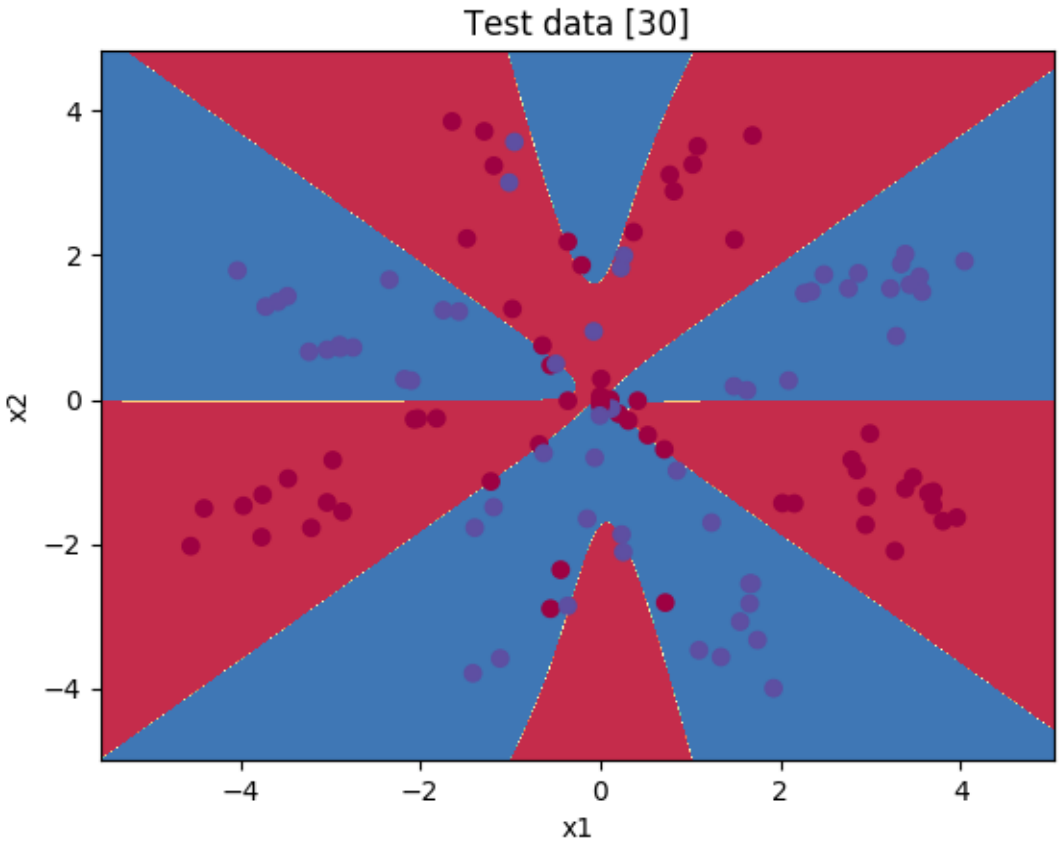
number of units in the hidden layer = 10
Accuracy on train data=89.21052631578948%
Accuracy on test data=87.5%
eta=0.03



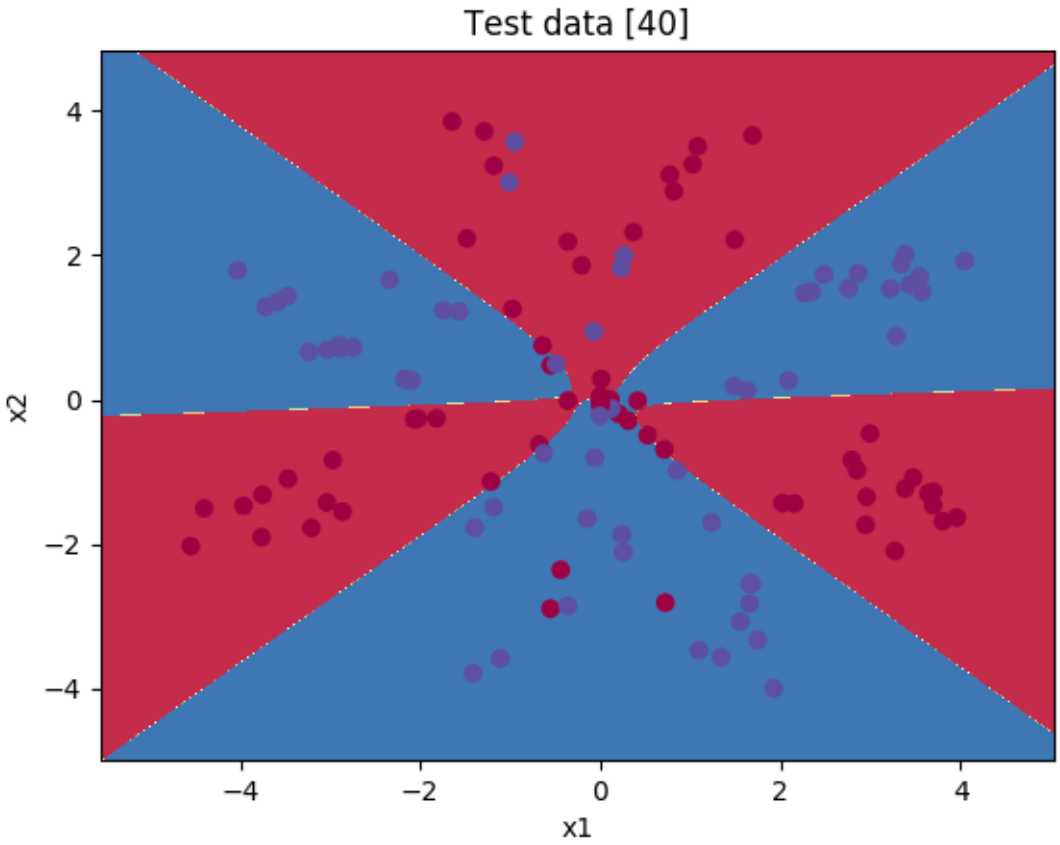
number of units in the hidden layer = 20
Accuracy on train data=89.21052631578948%
Accuracy on test data=86.66666666666667%
eta=0.01



number of units in the hidden layer = 30
Accuracy on train data=90.26315789473685%
Accuracy on test data=84.16666666666667%
eta=0.011

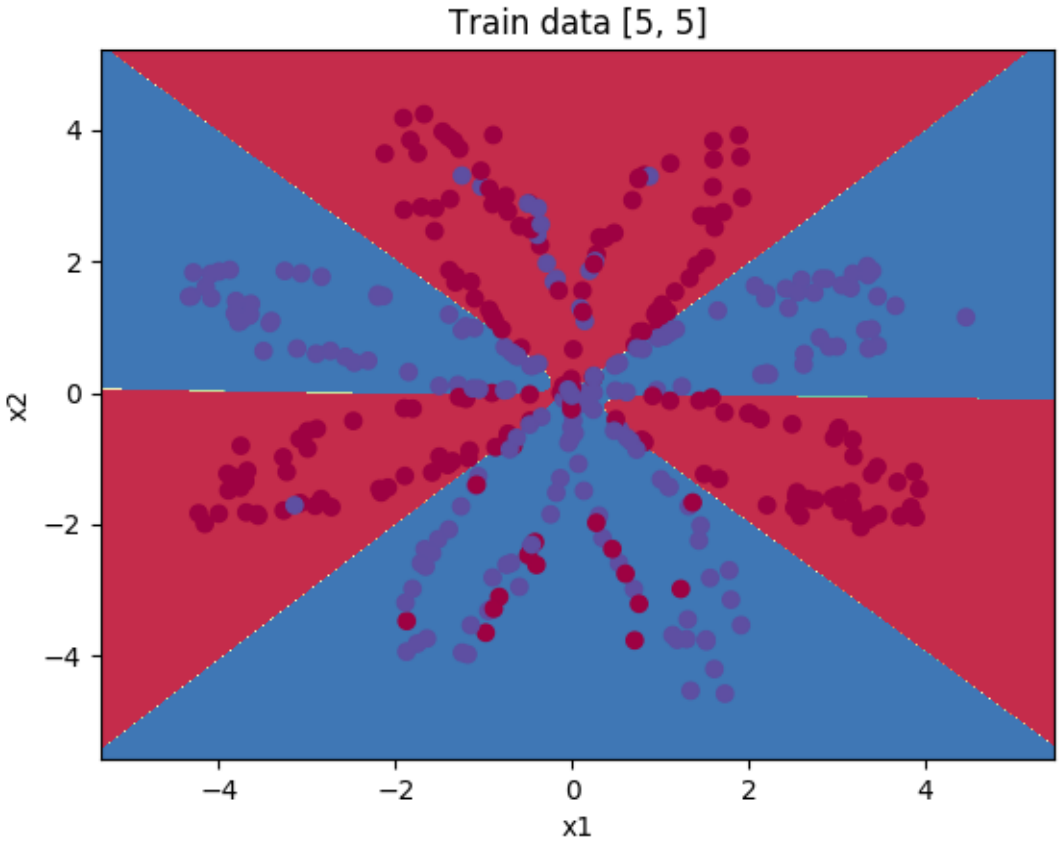


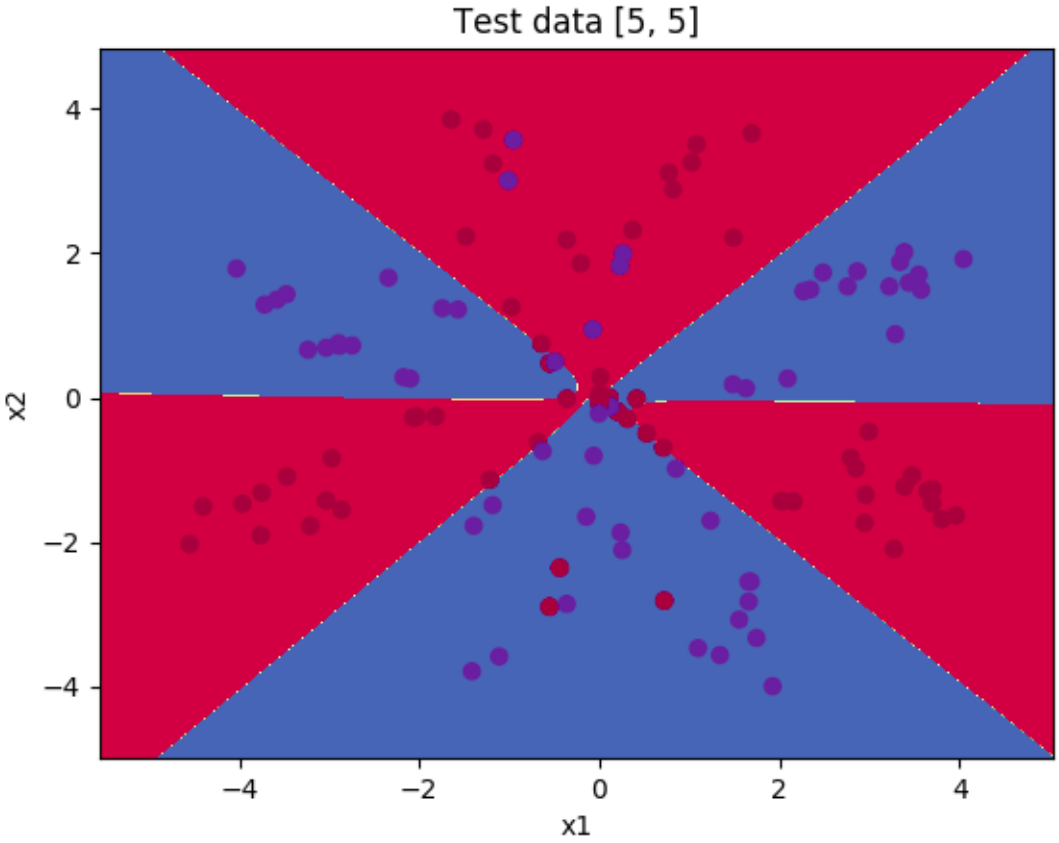
number of units in the hidden layer = 40
Accuracy on train data=89.21052631578948%
Accuracy on test data=85.83333333333333%
eta=0.008



As can be seen from the results and the graphs, less hidden layer units result in under-fitting while more hidden layer units lead to over-fitting. So there is a optimal no. of hidden layer units which is around 3 in this case.
As the number of hidden units are increased, the model fits the training data more and more closely and hence leading to over-fitting after some optimal no. of hidden units.

(4)
Accuracy on train data=89.7368%
Accuracy on test data=85.8333%
eta=0.03


Train data [5, 5]
x2
x1
4
2
0
−2
−4
−4
−2
0
2
4


Test data [5, 5]
x2
x1
4
2
0
−2
−4
−4
−2
0
2
4

If compared with NN models with single layer, the test accuracies are lower in NN with 2 hidden layer. This indicates that the model has over-fit to the data.
This indicates that as number of hidden layer increases, NN can learn more and more complicated boundaries.

Part(c): Working with MNIST

(1)

Used **LIBSVM** to train a linear classifier:
Train data Accuracy = 100%
Test data Accuracy = 98.4722%

Neural network with single hidden layer and one perceptron:
Accuracy on train data=97.8%
Accuracy on test data=97.5833%

(2) single hidden layer with 100 units

Accuracy on train data=98.47%
Accuracy on test data=97.86%

Stopping criteria:
if change in the value of Error function(for all the examples) is less than 0.01 then stop.

(3) Using **ReLU** as the activtion instead of the sigmoid function

Accuracy on train data=98.53%
Accuracy on test data=97.64%

Their is no improvement in the accuracy values (they are almost same), but the model converges faster with ReLU activation as compared to sigmoid activation.