# PROJECT 2: MULTILAYER PERCEPTRON AND DECISION TREE INDUCTION

Srinath Kanna Dhandapani(sd2689)

Shivankar Ojha(sxo4955)

## PROBLEM DEFINITION:

What we try and achieve through this project is to basically compare two algorithms: the multilayer perceptron training algorithm and a modification of the decision tree induction algorithm, to get two classifiers trained on the same training data. We then use a testing data set to compare the performances of the two classifiers and conduct experiments to gain further insight into features of the classifiers made from these algorithms, such as the sum of squared errors, the number of weight updates in the MLP, the number of nodes generated for the decision tree and so on.

## MODELS:

Our aim, as mentioned earlier is to build a multilayer perceptron and a decision tree based on the training data set. The multilayer perceptron consists of an input, a hidden and an output layer with 2, 5 and 4 nodes respectively. The 2 input nodes take as input the two attributes, which are propagated forward to the hidden layer, which in turn get their values based on the weights and send them forward. So, the output layer has 4 values, which together act as the output vector, and classifies a record to the class which has the highest corresponding value.

The decision tree has nodes implemented as a data structure, and the attributes are chosen to be split on their midpoint values rather than the attribute values themselves. We use a binary search tree to traverse through our tree structure and classify data after the tree has been trained.

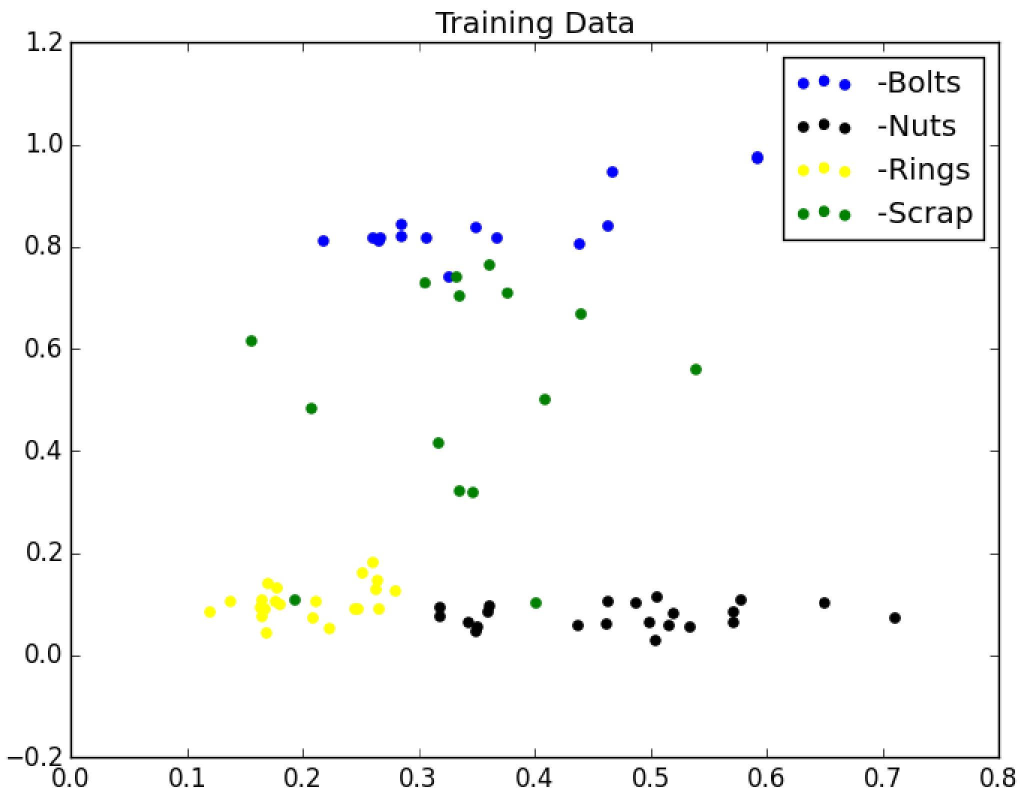
## ALGORITHMS:

The major similarity between these two algorithms is that they come up with classifiers at the end of the algorithm execution, classifying each record into a particular class, which in our data's case, for 4 possible classes of nut, bolts, rings and scrap.

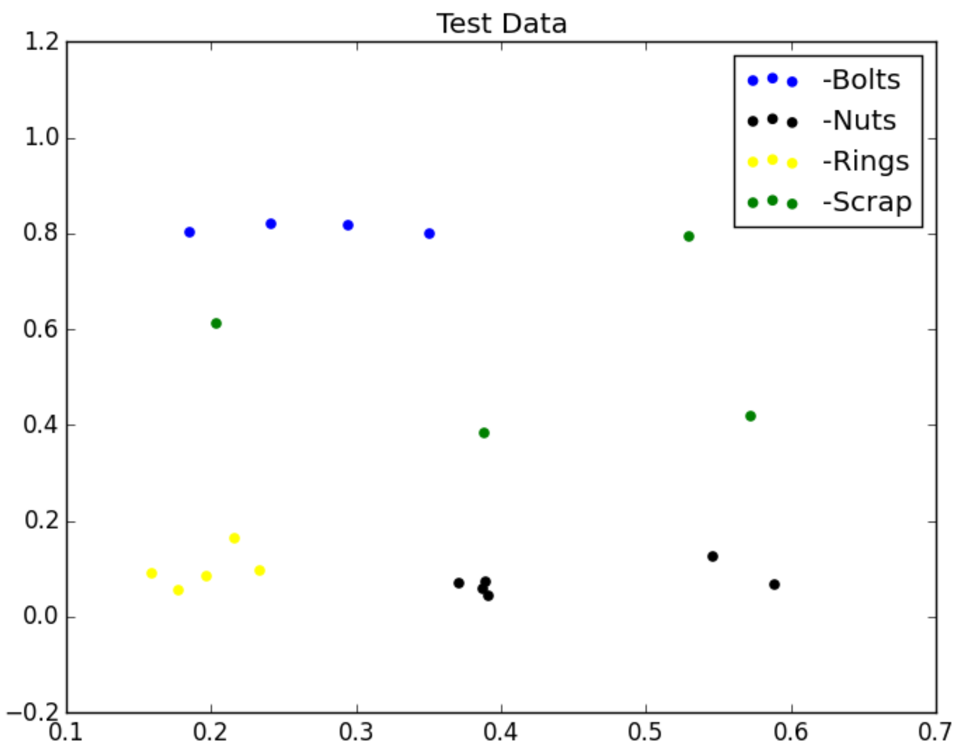
So, before we started to implement the algorithms, our initial perception was that the multilayer perceptron neural network's performance would be much more efficient than the decision tree. Our reason for thinking so was that the multilayer perceptron has features like back propagation which enable the errors of each layer to be sent to the previous layer and update the weights accordingly. It is therefore a much more efficient learning algorithm on paper at least. The decision tree on the other hand, has a greater chance of suffering from over fitting. What it will do is make the splits in the nodes in such a way that they are best suited to the training data set and thus may not be as efficient on the testing data set. Even though the chi squared pruning does reduce the over fitting, it would still not be able to match the excellent learning structure of the multilayer perceptrons which have back propagation as a powerful weapon.

## DATA:

Shown below are the plots for the training and the test data respectively:



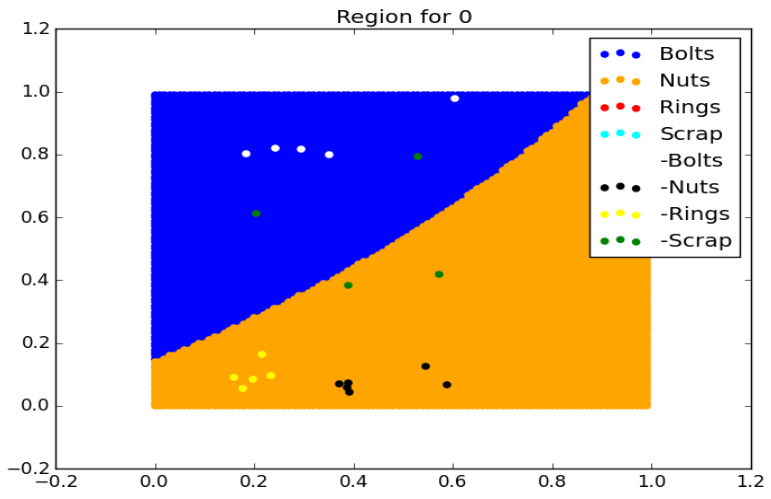
There are 15, 22, 22 and 15 records of each of the 4 class of nut, bolts, rings and scraps respectively in the 74 records of the training data set.  In the first look, we can clearly see the presence of 4 well defined clusters of each class present in the data distribution. So it is possible to train a classifier, whether it is a MLP or a decision tree to classify the data with a more than a reasonable accuracy.

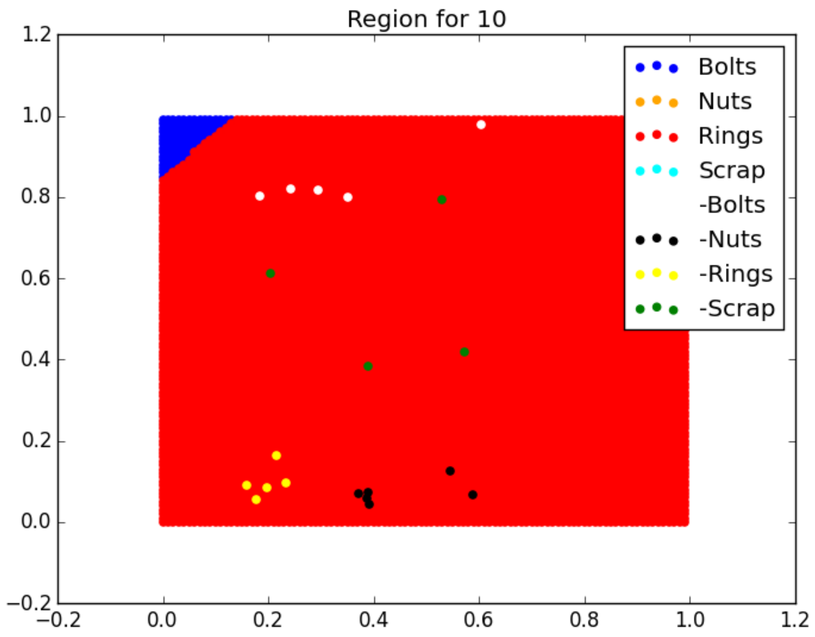


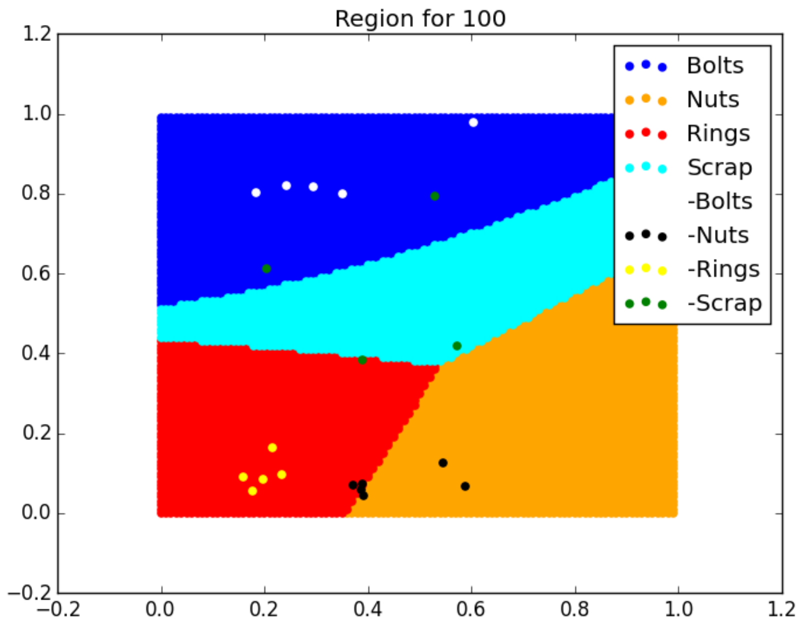
Similarly in the test data set, we can again see four well defined clusters. Thus, the MLP after a number of iterations should be able to classify the data set accurately. The decision tree may or may not be able to classify it completely correctly since it may be a overfit decision tree.

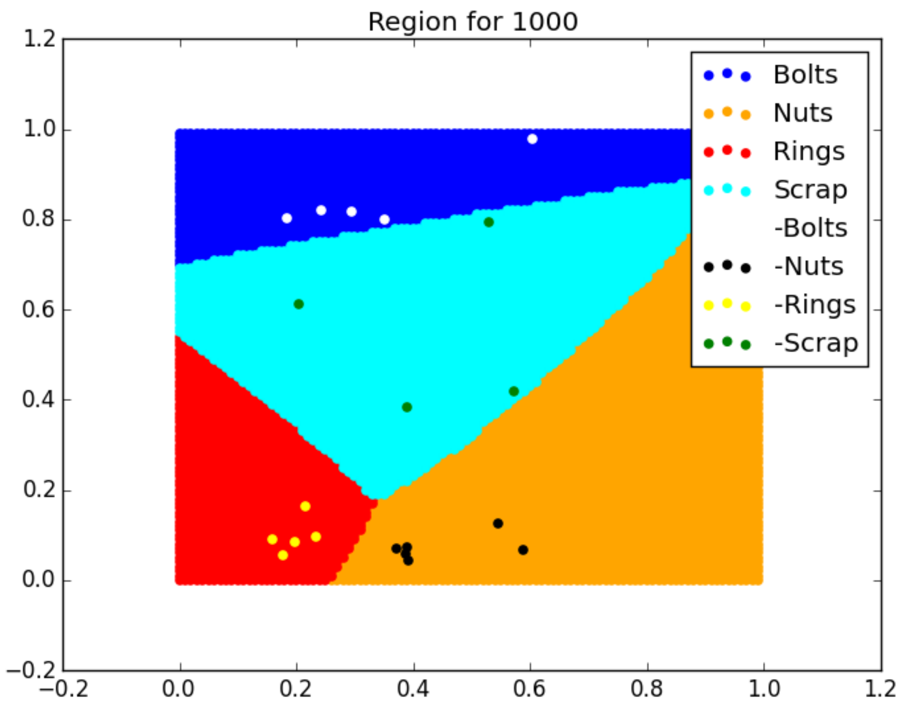
## **RESULTS:**

Shown below are the graphs depicting the regional classification for the epochs:

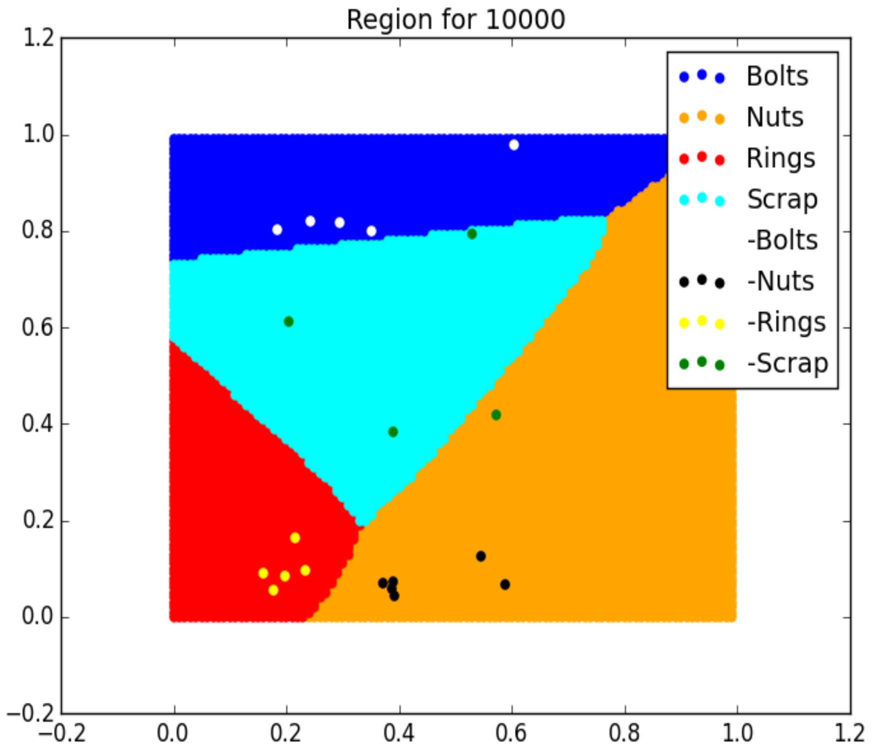



Region for 10
Bolts
Nuts
Rings
Scrap
-Bolts
-Nuts
-Rings
-Scrap
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2


Region for 100
Bolts
Nuts
Rings
Scrap
-Bolts
-Nuts
-Rings
-Scrap
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2

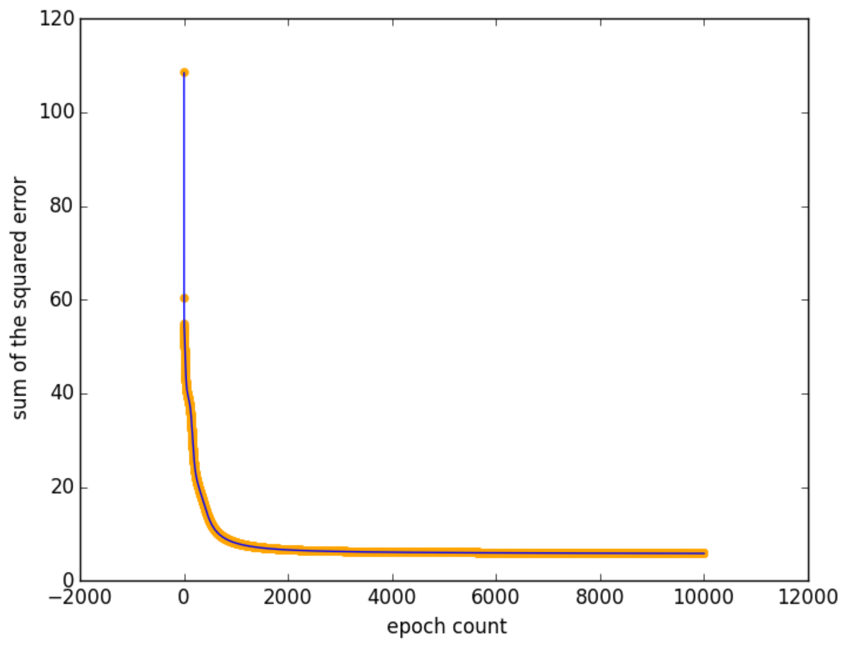


Region for 1000
Bolts
Nuts
Rings
Scrap
-Bolts
-Nuts
-Rings
-Scrap



Region for 10000
Bolts
Nuts
Rings
Scrap
-Bolts
-Nuts
-Rings
-Scrap

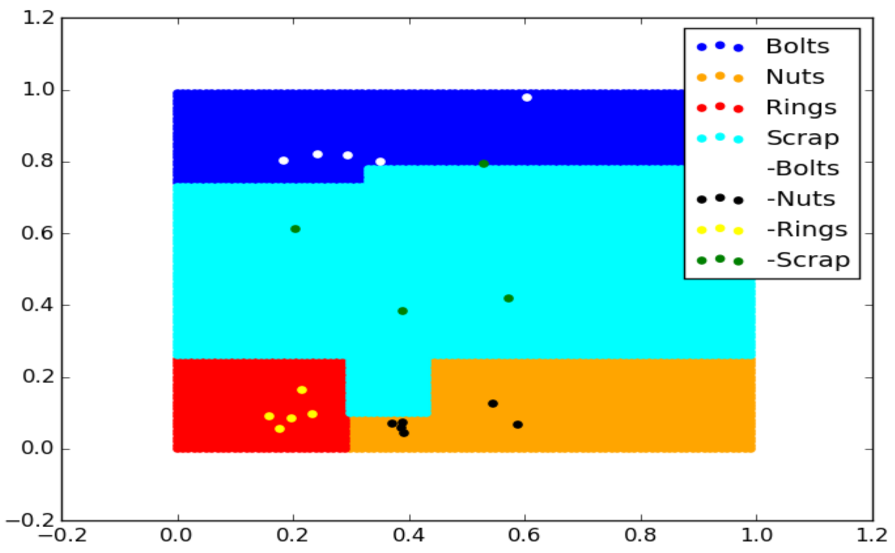
Shown below is the curve for the SSD v/s the epoch count for the multi layer perceptron. We see that after a certain point it becomes constant or changes negligibly. This point is the inflection point.



Shown below is the table showing the recognition rate and profit for each number of saved epochs for the MLP. As expected, the recognition rate keeps on increasing proportional to the increase in the number of epochs.

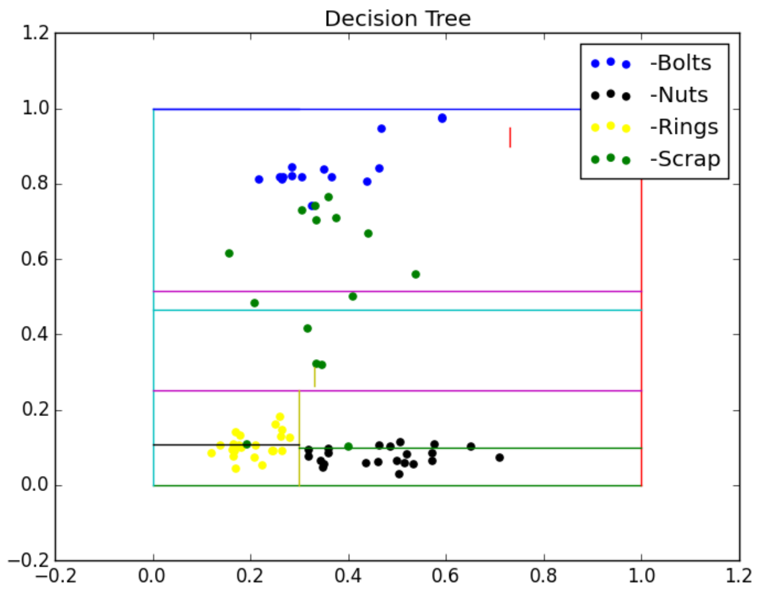
| ***Epoch*** | ***Recognition rate*** | ***Profit*** |
|---:|---:|---:|
| 0 | 25% | 15 |
| 10 | 25% | -80 |
| 100 | 60% | 63 |
| 1000 | 100% | 203 |
| 10000 | 100% | 203 |

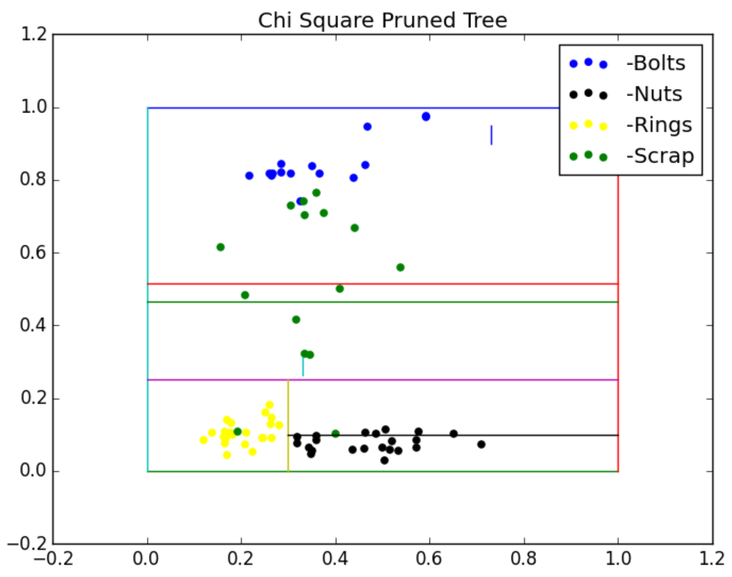
For the ***decision trees***, the classification regions for each of the decision trees are as follows:



**The plots for both the Pruned and Unpruned trees are the same as above for our implementation.**

Shown below are the splits in the feature space for the Unpruned and Pruned decision tree.





Shown below is the recognition rate and the profit for the pruned and unpruned decision tree.

| ***Tree*** | ***Recognition rate*** | ***Profit*** |
|---|---|---|
| Unpruned Tree | 95% | 199 |
| Pruned Tree | 95% | 199 |

## DISCUSSION:

As mentioned earlier, before implementing the solution to this problem, our view was that the multilayer perceptron is bound to outperform the decision trees in terms of classification accuracy. It has a much more sophisticated structure which facilitates feedback, whether it is positive or negative. If its output is higher than the expected value, then the feedback from the output layer is negative to compensate for it and vice versa.
As we increased the number of epochs of our MLP, we witnessed a steady decrease in the SSD as was expected, till it reached the inflection point, from where onwards the change was negligible. Another fascinating thing to note about the MLP was that it managed to classify the

training and testing data completely correctly, further reiterating the advantage of the feedback architecture.

As the number of epoch increased, so did the accuracy in classification till we finally managed to get 100% accuracy.

Surprisingly, the decision tree did not perform too badly for the classification problem either, getting a very high accuracy of 95% for both pruned and unpruned trees.
The MLP did outperform the decision tree though, simply because it got 100% accuracy a majority of the times in both the training and testing data samples. The accuracy changed for various runs since we were using random initial weights at each run. In contrast, there is no change in the accuracy of the decision tree at all.
If we judge the classifiers based on the profit values, again there is hardly any difference, the MLP giving us a maximum possible profit of 203, while it was 199 for a decision tree, as shown in the tables above.

If we try and notice the class boundaries in the MLP boundary diagram and the feature space boundaries for the decision trees, we notice that the boundaries are almost identical. This was possible only because of the fact that this data was clustered in a good way for classification. If the dataset had been oblique or slanted, the decision tree would not have been able to learn and classify the data points as affectively as it has done in this case.

## **BONUS:**

Our code for the MLP was a generalized one, since we had the number of input, output and hidden layers nodes stored in variables. So, in order to run our code for different number of nodes in the hidden layer, all we had to do was change the number of nodes in the hidden layer variable. We observed that the performance of the MLP was worsened by an increase in the number of nodes from 5 to 15, the accuracy fell drastically down to 50%, even after 10,000 epochs. And of course, the time increased alongside with it.

For the Chi squared pruning, we tried the pruning for both 1% and 5% significance levels. We have commented out the 1% significance level list in our code, since we used 5% only. But the result of our pruning process was similar in both of these cases, as the number of nodes decreased from 19 to 15, without any change in accuracy. Also, the behavior of both the pruned and unpruned trees was identical.