FOUNDATIONS OF INTELLIGENT SYSTEMS

PROJECT – 2

Aravindh Kuppusamy,

Deepak Sharma,

Karan Jariwala

---

1) ALGORITHMS:

- **Similarities:**

  Both Decision tree and Multilayer perceptron (MLP) algorithms solves optimization problems.

  MLP uses gradient descent algorithm which is a variant of hill climbing. In MLP, we start with assigning random weights to each connection between the neurons and then we measure the fitness of the model by checking sum of squared error. It tries to optimize (minimize) the sum of squared error using the derivative of the SSE (Sum of Squared Error) with respect to weights of the network. Based on the results of derivative by performing backpropagation, MLP updates its weights for generating approximately matching function between the provided set of inputs (attributes) and outputs (labels) by changing the weights.

  On the other hand, decision tree tries various values for deciding a set of thresholds for data attributes which can be used in a set of sequential conditional logical operations of decision tree. Here we have used information gain as fitness function for deciding the threshold values and sequences of conditional logical operations.

- **Differences:**

  MLP with more than one layer can generate complex non-linear matching function whereas decision boundary can only synthesize linear hypothesis. As in our experiment we have found that decision tree is not capable of generating non-linear boundaries while the trained MLP can generate the nonlinear boundaries.

  Decision tree uses attributes based on their importance and pruning can throw away attributes if it found they are not useful. On the other side, MLP uses all the attributes of the given dataset irrespective of their usefulness.

For neural network, training and testing both can be computationally intensive. However, testing of decision tree is computationally less intensive.

- **Expectation:**

  After checking the distribution of the data, we found that class scrap is not linearly separable hence the decision tree may not be able to perform well while synthesizing decision boundary between scrap class and other classes. If sufficient train data is available then a two layer MLP can construct a true decision boundary between scrap class and other classes. So we expect MLP to perform better given a good number of epochs to update the weights. We also expect decision tree algorithm to correctly classify classes other than scrap class.

2) DATA:

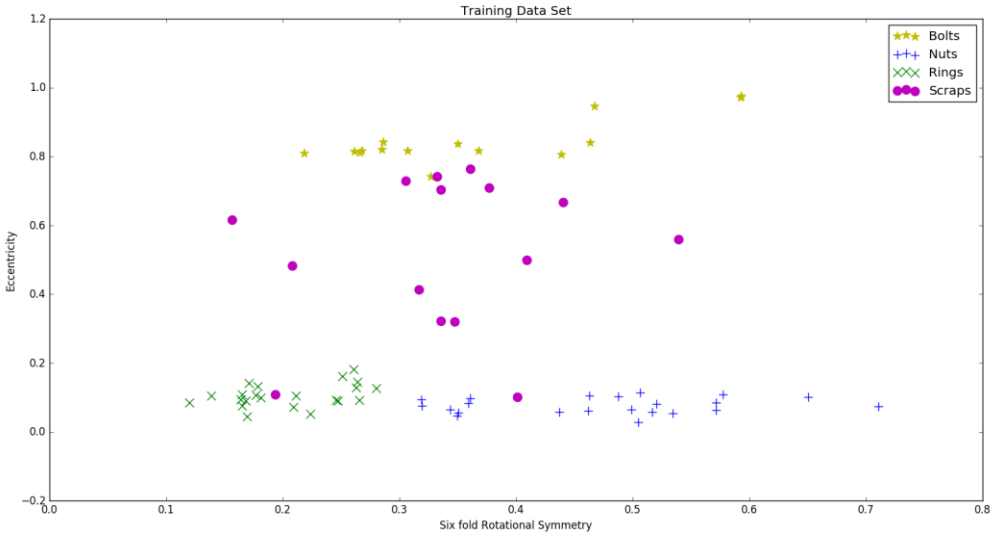

Figure [1]: Classification using Decision Tree before pruning

Data consists of four classes (1. Bolts, 2. Nuts, 3. Rings, 4. Scraps). The four classes are distributed in the following manner.

1. Bolts have high eccentricity and are mostly concentrated within small region (~0.8 < x > ~0.9) over all other classes. But their rotational symmetry spreads over a large number of values (~0.2 < x > 0.6).

2. Nuts have very less eccentricity and are mostly concentrated within small region (~0 < x > ~0.1) compared to nuts and scraps but similar to that of rings. But their rotational symmetry spreads over a large number of values (~0.3 < x > 0.7) and is higher than rings.

3. Rings have both very less eccentricity (~0 < x > ~0.2) and very less rotational symmetry (~0.1 < x > ~0.3) and are mostly concentrated within small region compared to other classes.

4. Scraps have eccentricity and rotational symmetry spread over in between all other classes. Their eccentricity is more than nuts and rings.

3) RESULTS

(A) Multilayer Perceptron

(i) Test Samples and Classification Region

(a) epoch: 0

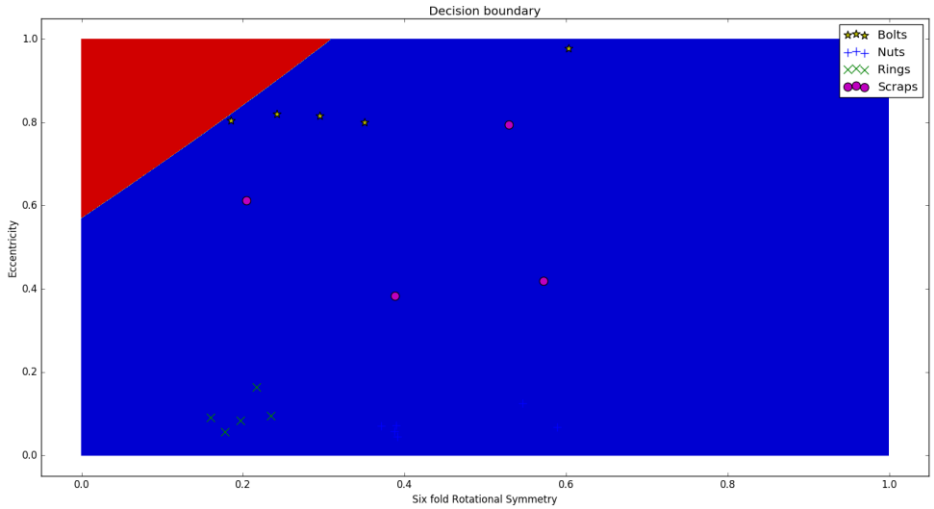

Figure [2]: Decision Boundary for 0 epochs

(b) epoch: 10

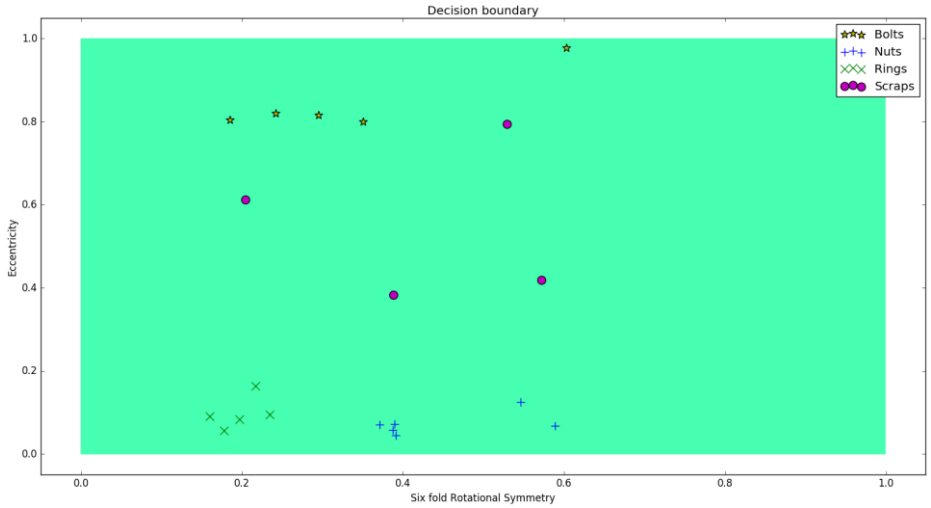

Figure [3]: Decision Boundary for 10 epochs

(c) epoch: 100

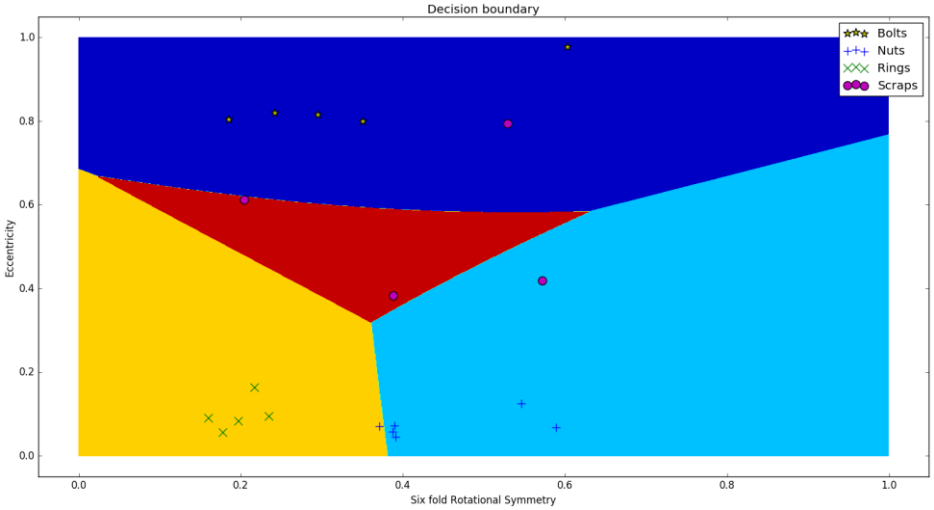

Figure [4]: Decision Boundary for 100 epochs

(d) epoch: 1000

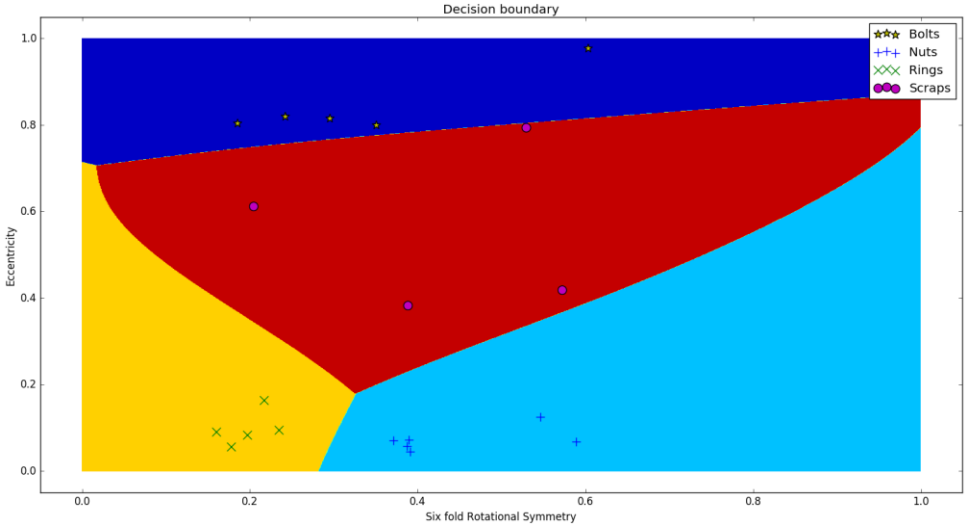

Figure [5]: Decision Boundary for 1000 epochs

(e) epoch: 10,000

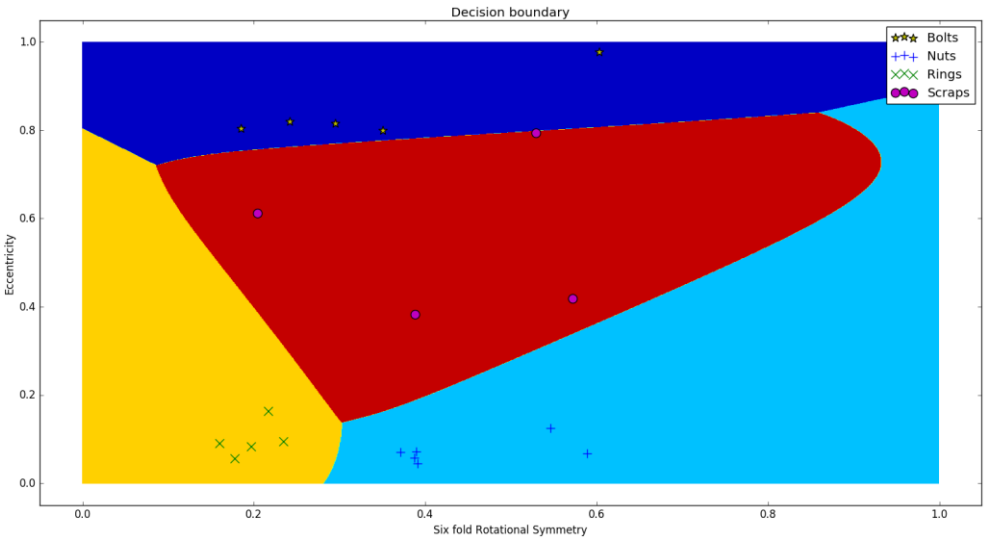

Figure [6]: Decision Boundary for 10000 epochs

(ii) The learning curve image (SSD vs. Epoch) for the trained MLP from 0 to 10,000 Epochs

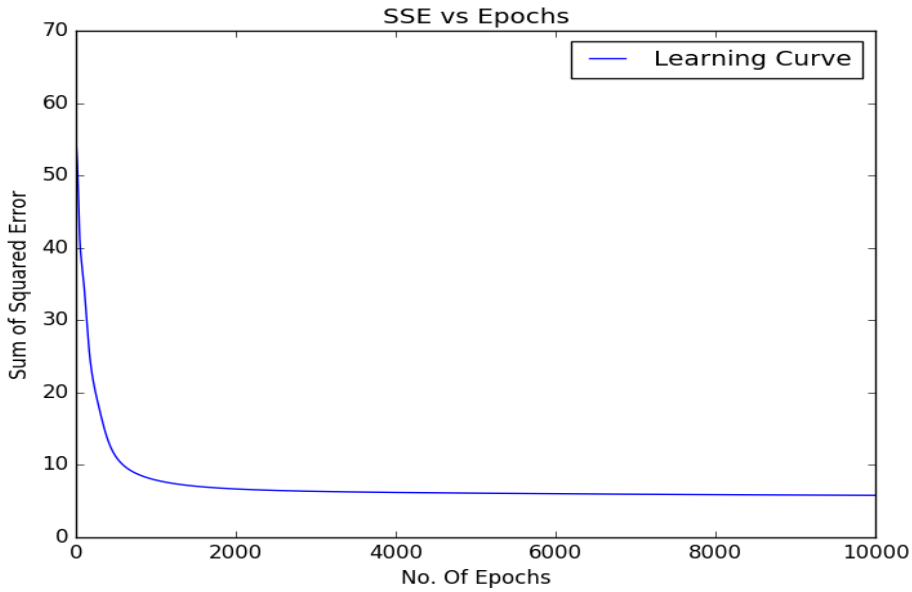

Figure [7]: SSE vs Epochs for the trained MLP from 0 to 10,000 epochs

(iii) A table showing the recognition rate and profit for each number of saved epochs for the MLP

Epochs	0	10	100	1000	10000
Recognition Rate (%)	30	25	85	100	100
profit	-8	-80	173	203	203

Table [1]: Recognition rate and profit for each epochs

B) DECISION TREE:

(i) Plots showing the test samples and classification regions produced by each of your decision tree before and after pruning.

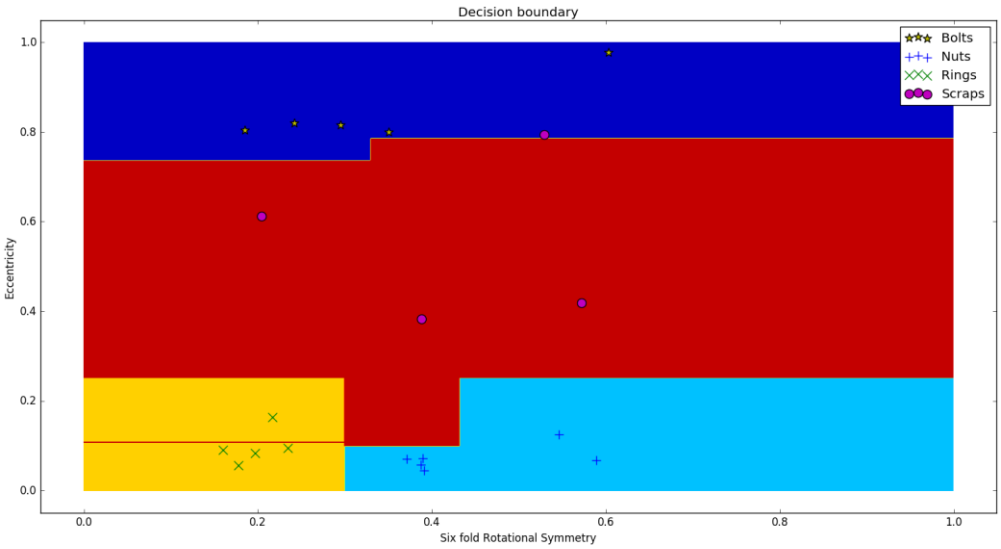

Figure [8]: Decision Boundary without pruning

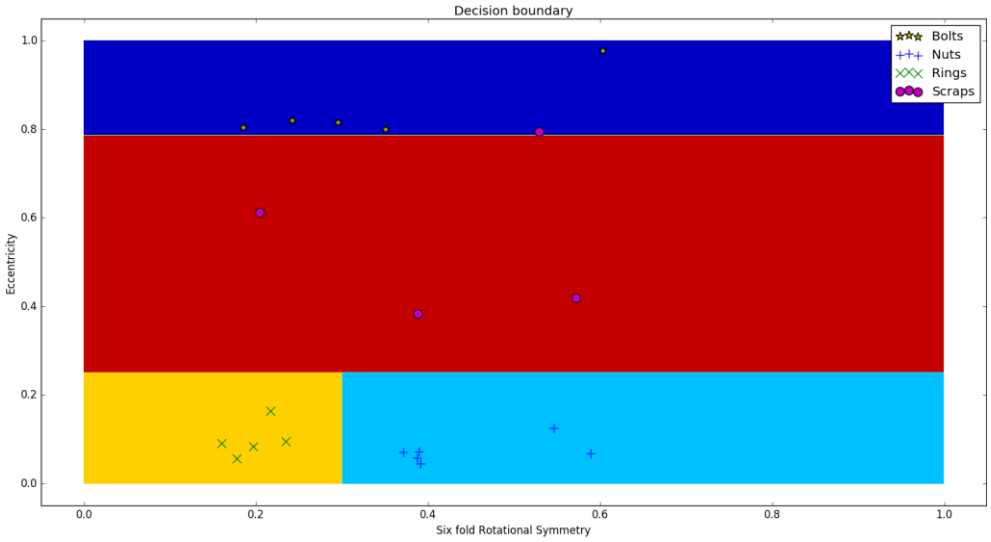

Figure [9]: Decision Boundary after pruning

(ii) Plots showing how feature space is split your decision tree before and after pruning.

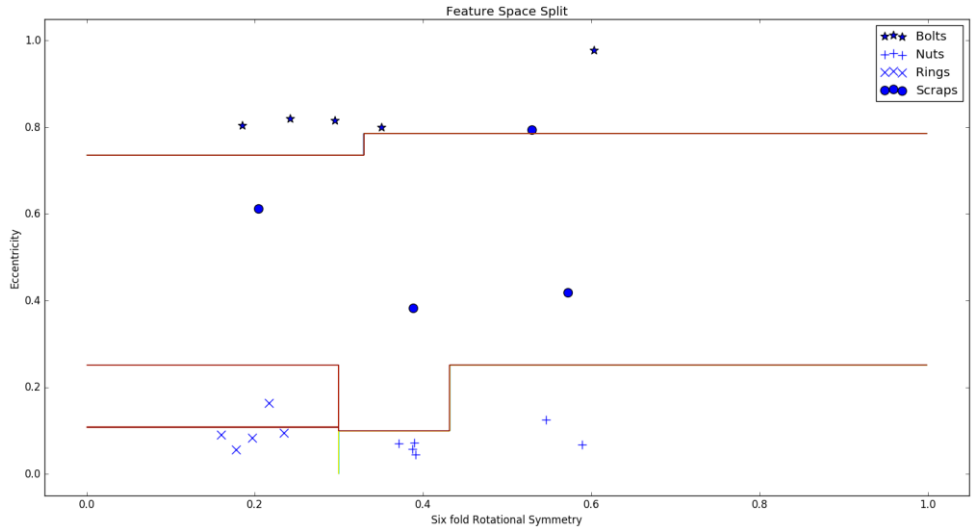

Figure [10]: Feature space split before pruning

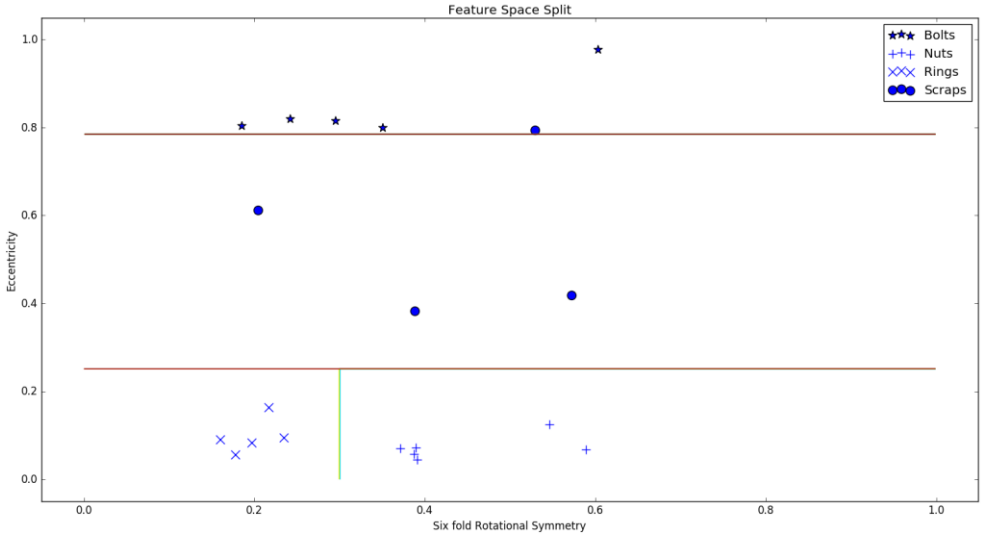

Figure [11]: Feature space split after pruning

(iii) A table providing the recognition rate and profit obtained before and after pruning, along with the tree metrics produced by trainDT.py

Category	Without prune	With prune
Recognition Rate (%)	95	95
Profit	199	199

Table [2]: Recognition rate and profit

4) DISCUSSION

(i) Which of the different classifiers performed best in terms of 1) accuracy and 2) profit? Did this meet your expectations?

As we can see from the above tables of Multi-layer perceptron and Decision tree, after 1000 epochs, the accuracy and profit of Multi-layer perceptron is better than the decision tree.

This meets our expectation for the given distribution of test data. But this data is almost linearly separable. If the provided distribution of test data is more tangled between the classes, then we don't expect the decision tree to give the same profit and accuracy and much lesser than what is provided by MLP.

(ii) How do the hypotheses (i.e. class boundaries) and performance metrics differ between the different version of the MLP and decision trees, and why?

With 0 epochs of training the multilayer perceptron was working with just an initial set of randomly assigned weights. As shown in figure [2], the decision boundaries do not make any sense.

With 10 epochs, the network does not understand the underlying distribution of data and it was predicting all the samples to be of Rings (Class-3) as shown in the below table.

Pre/Act *	Class 1	Class 2	Class 3	Class 4
Class 1 *	0	0	0	0
Class 2 *	0	0	0	0
Class 3 *	5	6	5	4
Class 4 *	0	0	0	0

Table [3]: Table Confusion Matrix Generated using MLP trained with 100 Epochs.

With 100 epochs, the neural network reduced the Sum of Squared error significantly from 10 epochs (62 to 28) as shown in figure [7]. The backpropagation process assigned a set to weights to the network which could approximate the function which represents the underlying distribution of the data. Here data points from Class Nuts and Class Scraps were misclassified in test dataset.

With 1000 epochs, the neural network almost approximated the matching function and it has shown high accuracy. Figure [7] shows that SSE reduces to value less than 10. The Decision boundary curve in figure [5] shows that neural network was able to build nonlinear hypothesis since the decision boundaries are high degree polynomial. A two-layer neural network have capacity to approximate non-linear hypothesis.

With 10000 epochs, the neural network didn't reduce the SSE much significantly after 1000 epochs as we can see in figure [7]. But as we can see in figure [6], the decision boundaries were made of high degree polynomials. These high degrees polynomial might result in overfitting. But with the given distribution of test dataset overfitting effect does not appear.

In Decision Tree, we are getting the same recognition rate and profit with and without pruning of the tree for the provided dataset. But it might not always be the same case.

Without pruning, the decision boundary as shown in figure [8], also considers the outliers and makes the boundaries which are not necessarily be true. On pruning the decision boundary as shown in figure [9], doesn't consider the outliers and makes good boundaries.

As we discussed earlier, the decision tree isn't able to synthesize high degree polynomial relationship between attributes and labels.

Bonus:

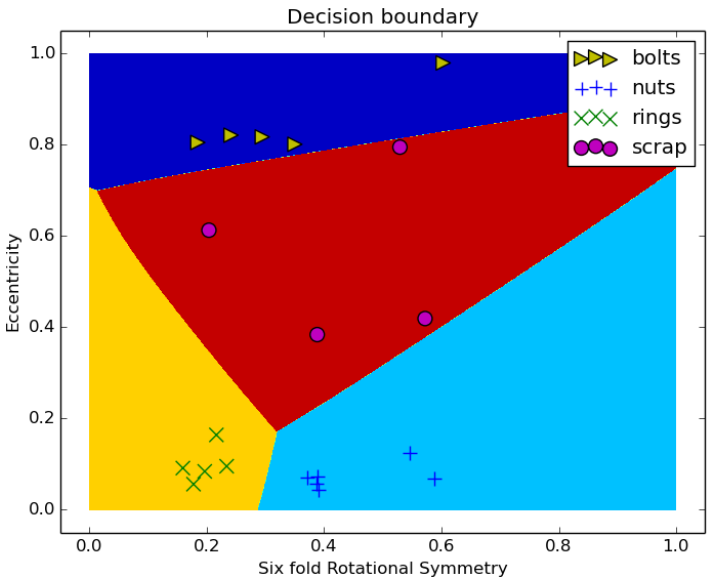

Figure [12]: Decision boundary with 3 neurons in hidden layer after 1000 epochs

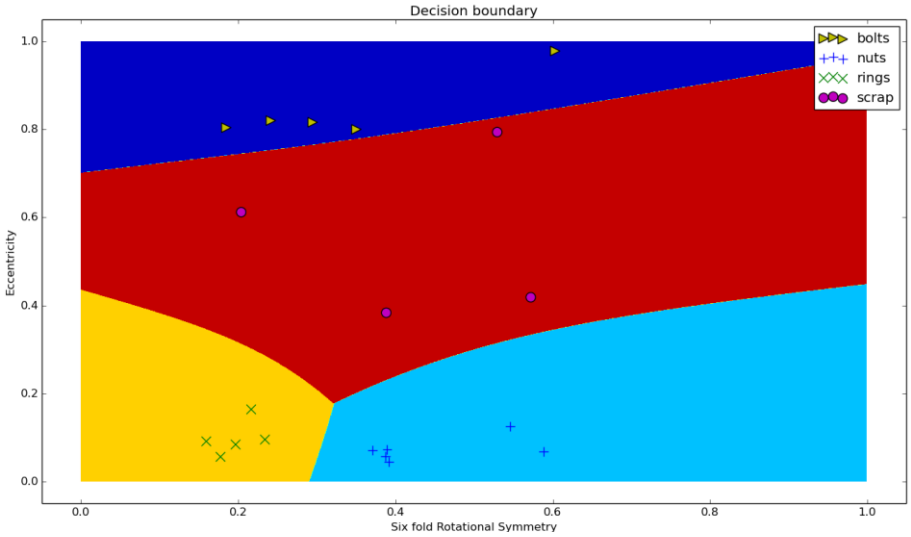

Figure [13]: Decision boundary with 10 neurons in hidden layer after 1000 epochs

Multi-Layer Perceptron:

With 3 neurons in hidden layer, the capacity of the network was lesser than the network with 5 neurons. Hence after 100 epochs, a network with 5 neurons and more in hidden layer produced profit of 501, but a network with only 3 neurons in hidden layer generated profit of only 459. However, the final accuracy of all MLPs after 1000 epochs were same.

The decision boundaries generated using trained neural network with 10 neurons in hidden layer represents a high non-linear function. However, the decision boundaries is almost linear when generated using 3 neurons in hidden layer. Because the capacity of network is higher for the network with 10 neurons in hidden layer and it can map high degree polynomial matching function than the network with 3 neurons in the hidden layer. In figure [12] and [13], we can see that the decision boundaries generated with 5 or more neurons represent a more complex function than the decision boundaries generated with 3 neurons.

Decision Tree:

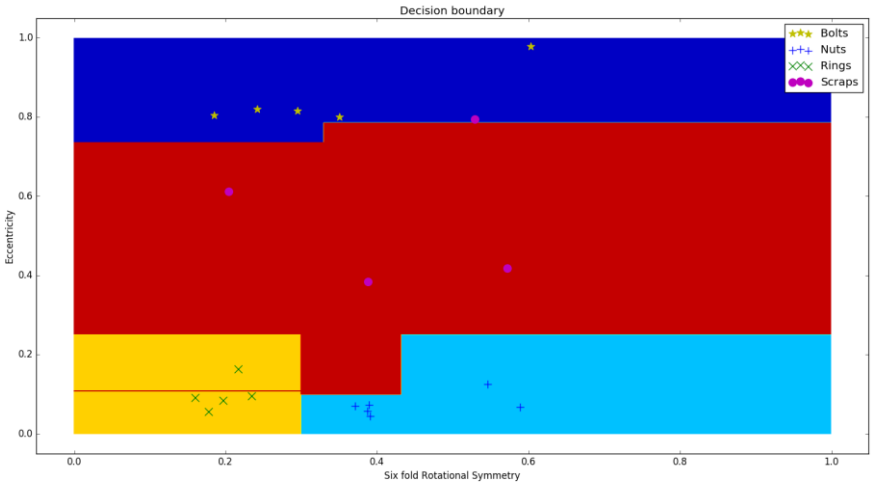

Figure [14]: Decision boundary with significance value = 0.75 without pruning

By changing the significance value to “0.1” and “0.75” for the Chi-Square test, we noticed that the profit and the recognition rate, remains the same as the experiments done with significance values “0.01” and “0.05”, as you can see from the decision boundary figure [14] which is same as [8] and [9].