Foundations of Intelligent Systems - Project 2

Akshay Karki
Krishna Tippur Gururaj

Algorithms:

- Similarities

The multilayer perceptron (MLP) and the decision tree (DT) algorithms are used to achieve statistical classification, i.e., they are classifiers.

The MLP starts with a set of randomly assigned network weights for the neuron connections and with each feed-forward propagation (progress of an input through the network till the output), the efficiency of the model's accuracy is computed using the concept of sum of squared errors (SSE). Based on the errors seen at the output layer, the network then undergoes the process of back-propagation, during which all network weights are modified using gradient descent algorithm. This one iteration is supposed to make the network "learn" how to output a certain value given a set of input values.

The DT algorithm tries various values for deciding thresholds for the input attributes for performing data splits sequentially to finally arrive at a predicted class for a set of given input attributes. In this implementation, we have used information gain to determine the best possible attribute to use for splitting data at a certain point.

- Differences

One major difference between the two algorithms is that MLP is able to generate decision boundaries for non-linear data whereas DT is unable to do so; DT is only able to handle linear data.
The other difference is that decision trees are capable of stopping the splitting of data after a certain point (prune the tree) to make computation more efficient whereas an MLP has no such feature; it considers all attributes every time. Classifying data using a trained MLP is computationally more intensive than doing so with a trained DT.

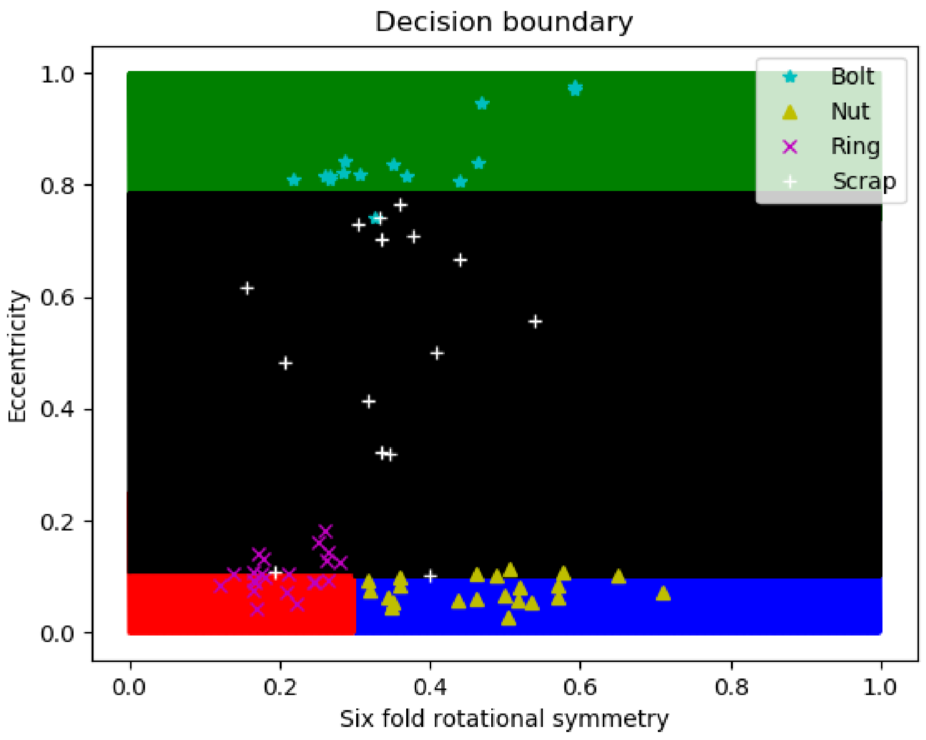
- Expectation

Looking at the distribution of the given training data, we have seen that the data points signifying Scrap are not linearly separable unlike the other three classes' data points.

Considering the number of epochs that we plan on run the MLP for (10,000 epochs) on a considerably smaller dataset (74), we expect the MLP to be able to classify all the data points perfectly whereas the DT should be able to classify the Bolts, Nuts, and Rings classes perfectly but would struggle with classifying the Scrap data points.

Data:

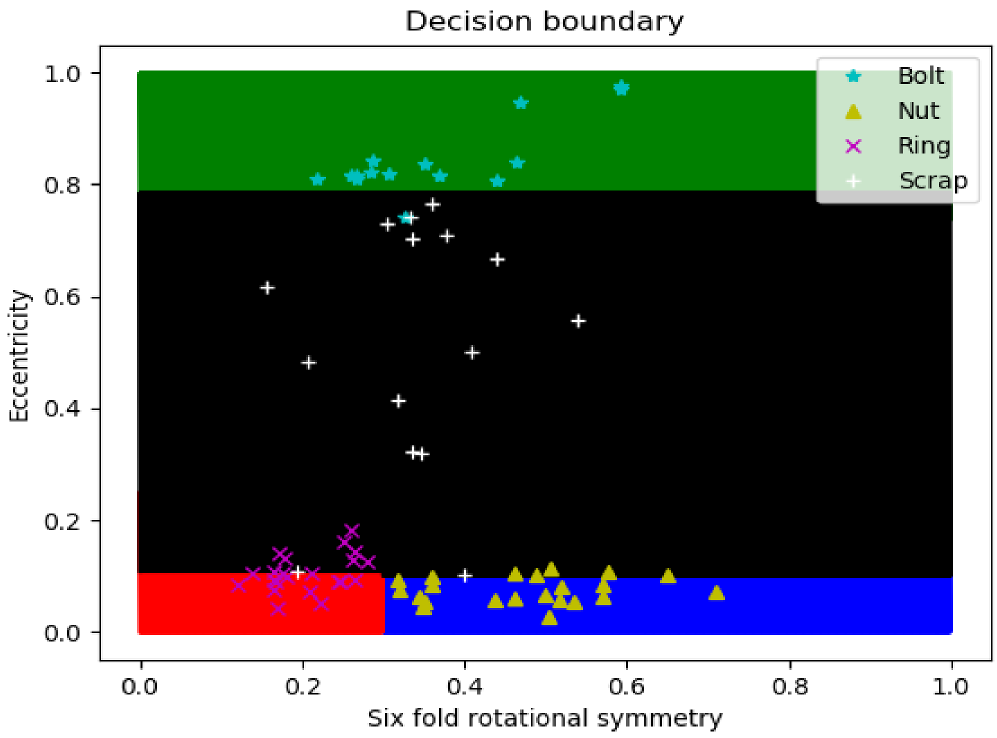
Plot of training data before pruning



The provided training data consists of four class labels, Bolt (class 1), Nut (class 2), Ring (class 3), and Scrap (class 4). We observed the following distribution in the data:

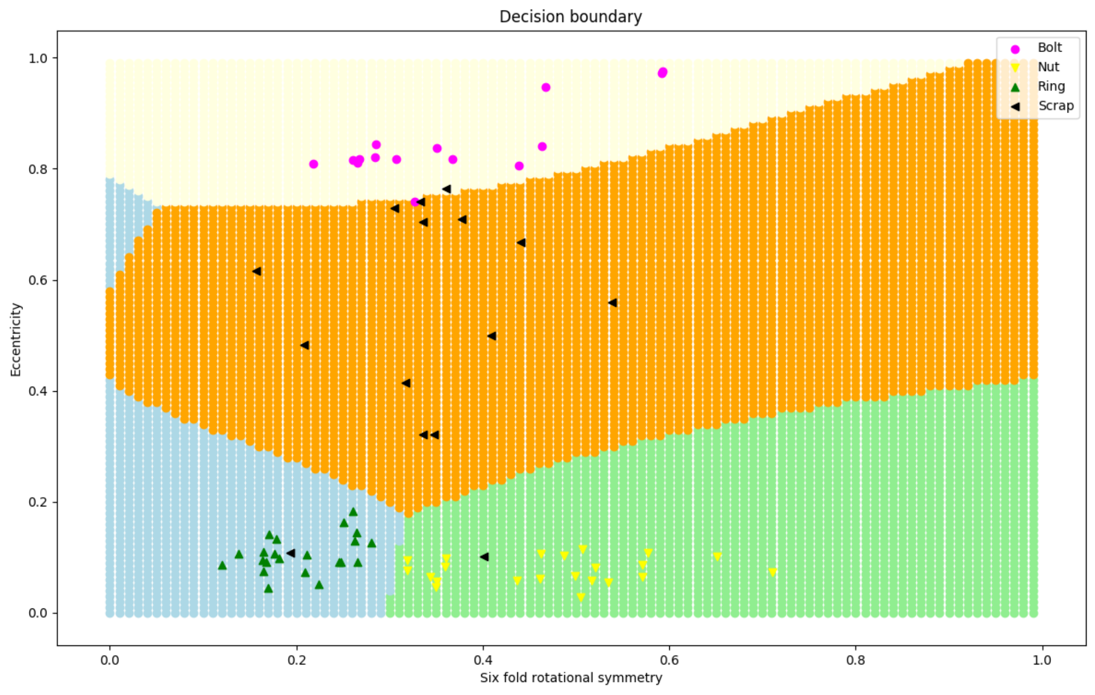
1. Bolts all have high eccentricities concentrated within a small range (~0.8 - 1.0) whereas their six-fold rotational symmetry values are more spread out (~0.2 - 0.6).
2. Nuts have extremely low eccentricities which are also concentrated within a small range (~0.0 - 0.1) while their six-fold rotational symmetry values are also within a relatively broader range (~0.3- 0.7).
3. Rings have a narrow range of spread of their eccentricities (~0.05 - 0.2) as well as their rotational symmetries (~0.1 - 0.3). They are concentrated in a very specific part of the plot.
4. Scrap values are spread out both in terms of their eccentricities (~0.1 - 0.8) as well as their rotational symmetries (~0.2 - 0.6).

Plot of training data after pruning



The provided training data consists of four class labels, Bolt (class 1), Nut (class 2), Ring (class 3), and Scrap (class 4). We observe that the decision plot is the same as before pruning. This is because we are generating the same tree after pruning. So, we expect the results to remain the same after pruning as they were before pruning.

Plot of training data



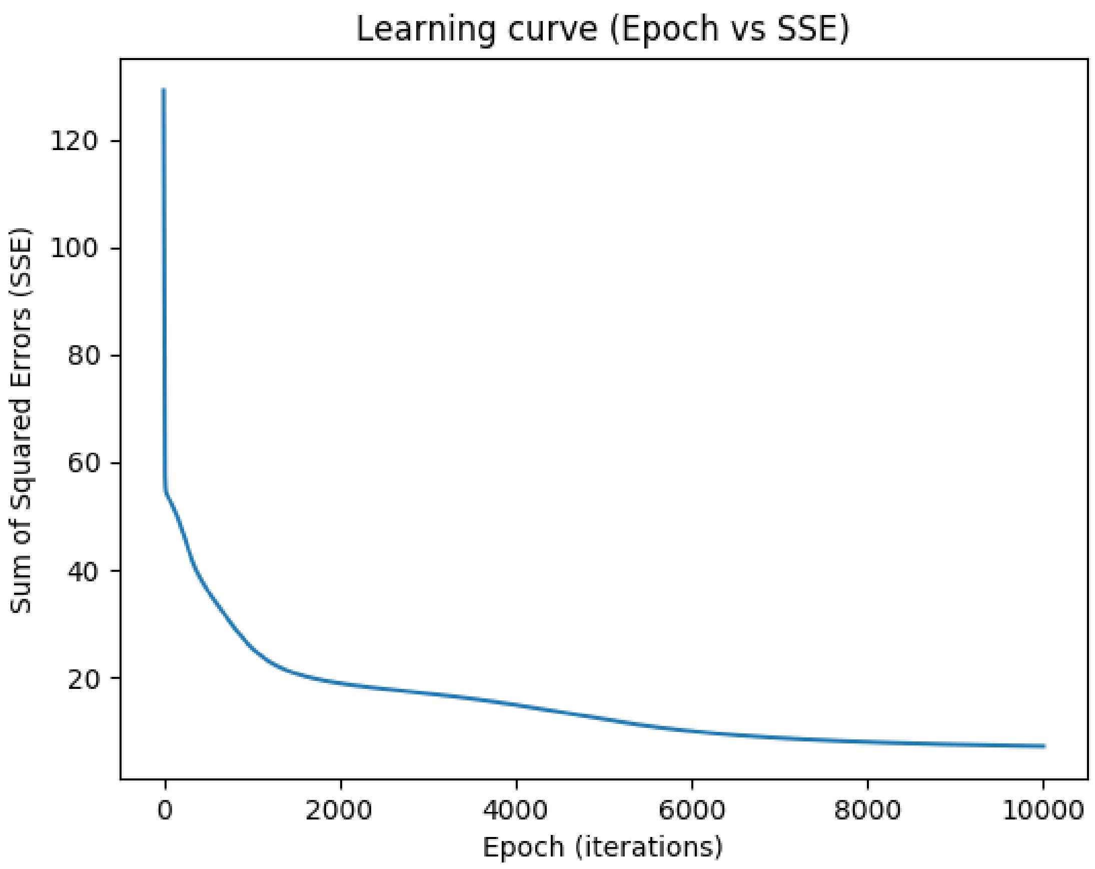
The provided training data consists of four class labels, Bolt (class 1), Nut (class 2), Ring (class 3), and Scrap (class 4). We observed the following distribution in the data:

1. Bolts all have high eccentricities concentrated within a small range (~0.8 - 1.0) whereas their six-fold rotational symmetry values are more spread out (~0.2 - 0.6).
2. Nuts have extremely low eccentricities which are also concentrated within a small range (~0 - 0.2) while their six-fold rotational symmetry values are also within a relatively broader range (~0.3- 0.7).
3. Rings have a narrow range of spread of their eccentricities (~0 - 0.2) as well as their rotational symmetries (~0.1 - 0.3). They are concentrated in a very specific part of the plot.
4. Scrap values are spread out both in terms of their eccentricities (~0.1 - 0.8) as well as their rotational symmetries (~0.2 - 0.6).

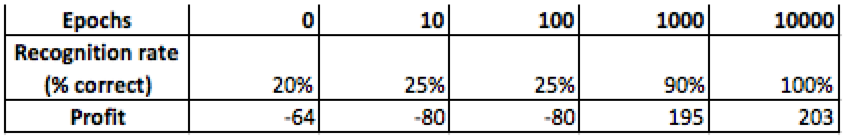
Results:

- <u>MLP</u>

  <u>The learning curve (SSE vs epoch) for epoch ranging from 0 to 10,000</u>:



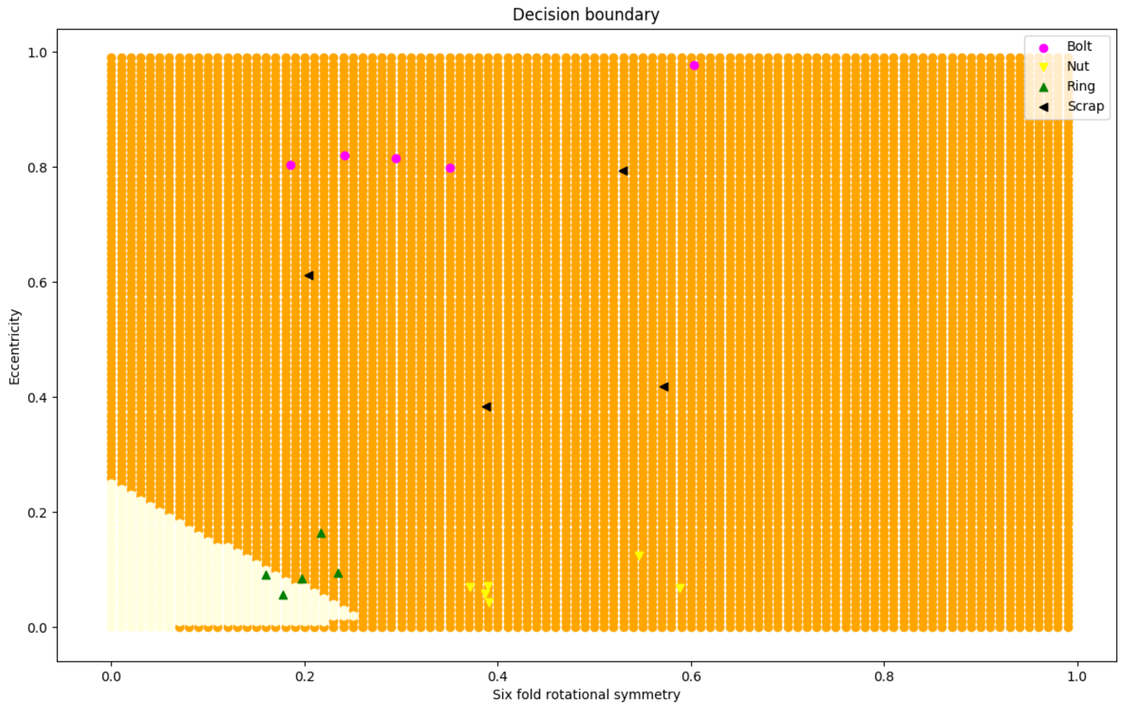
<u>The accuracy and the profit earned by using the model for different epochs</u>:



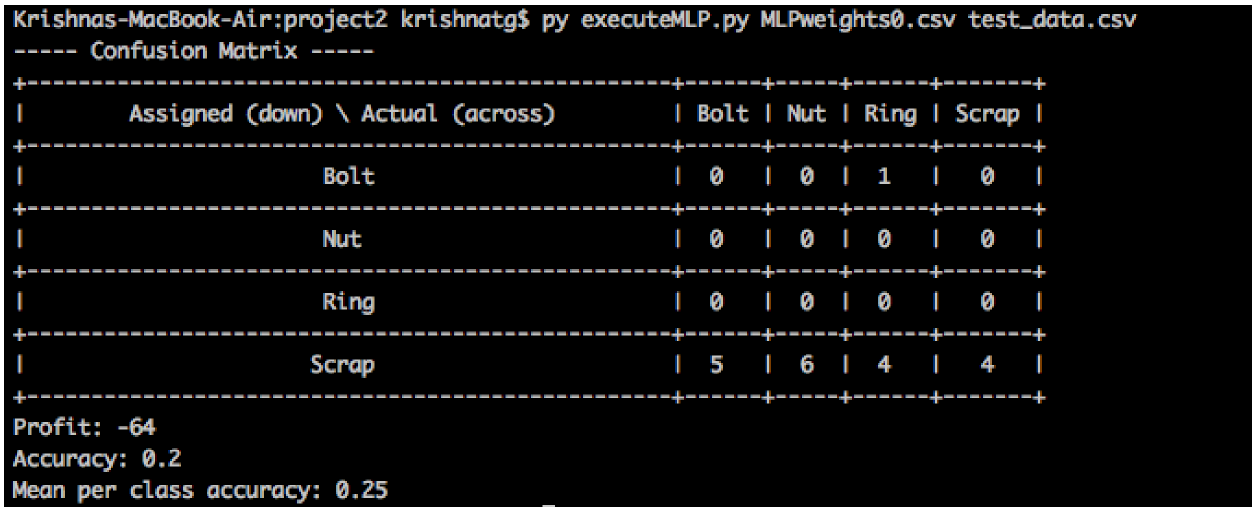
Epochs	0	10	100	1000	10000
Recognition rate (% correct)	20%	25%	25%	90%	100%
Profit	-64	-80	-80	195	203

<u>The decision boundaries and the test data plotted on them for different epochs</u>:

i) Epoch 0:

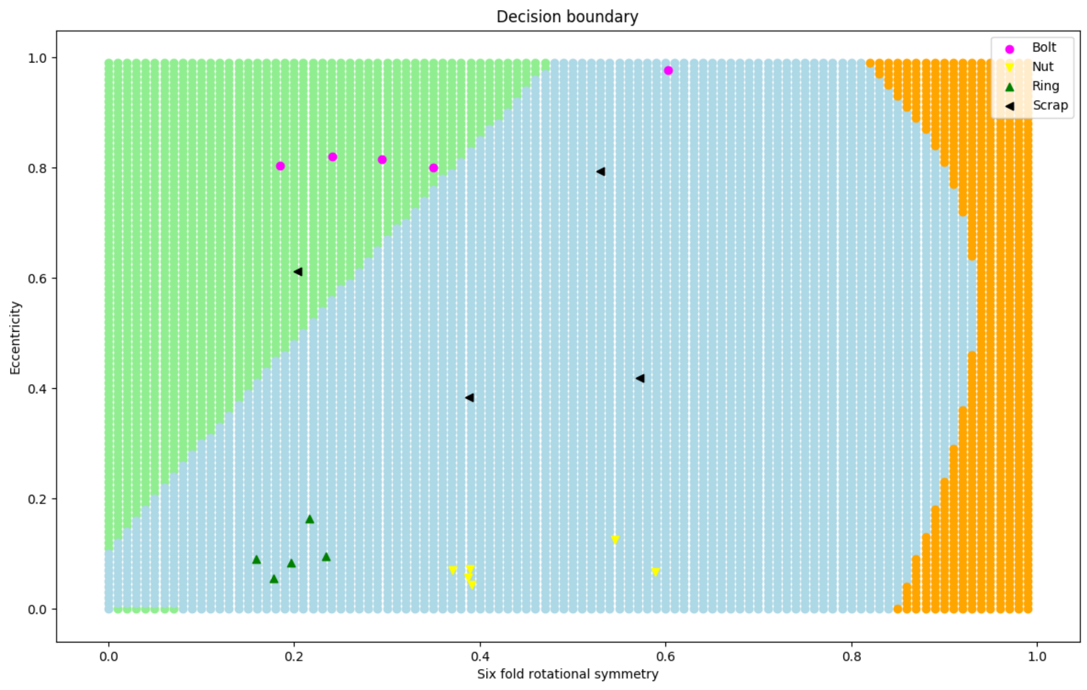


Confusion matrix:

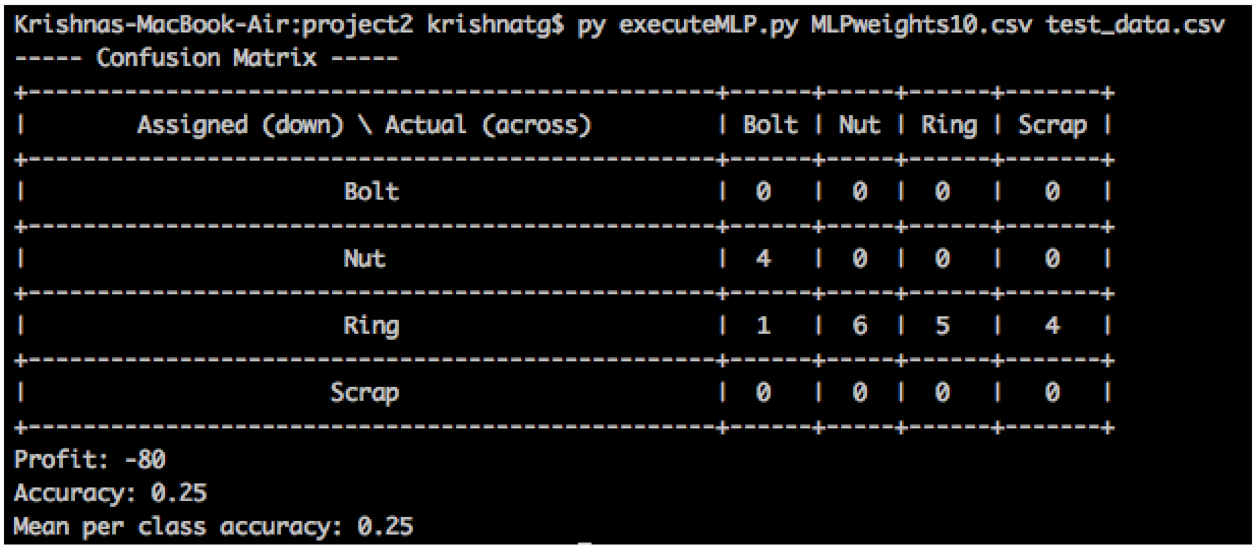


```
Krishnas-MacBook-Air:project2 krishnatg$ py executeMLP.py MLPweights0.csv test_data.csv
----- Confusion Matrix -----
+----------------------------------------------+------+-----+------+-------+
|        Assigned (down) \ Actual (across)     | Bolt | Nut | Ring | Scrap |
+----------------------------------------------+------+-----+------+-------+
|                     Bolt                     |  0   |  0  |  1   |   0   |
+----------------------------------------------+------+-----+------+-------+
|                     Nut                      |  0   |  0  |  0   |   0   |
+----------------------------------------------+------+-----+------+-------+
|                     Ring                     |  0   |  0  |  0   |   0   |
+----------------------------------------------+------+-----+------+-------+
|                    Scrap                     |  5   |  6  |  4   |   4   |
+----------------------------------------------+------+-----+------+-------+
Profit: -64
Accuracy: 0.2
Mean per class accuracy: 0.25
```

ii) Epoch 10:

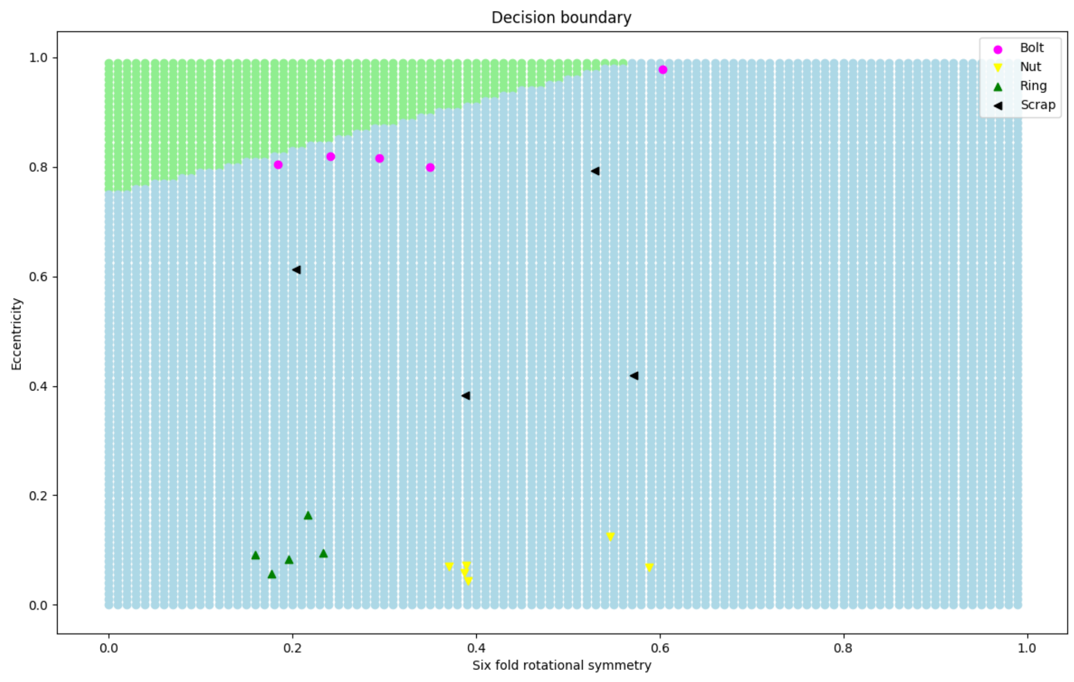


Confusion matrix:

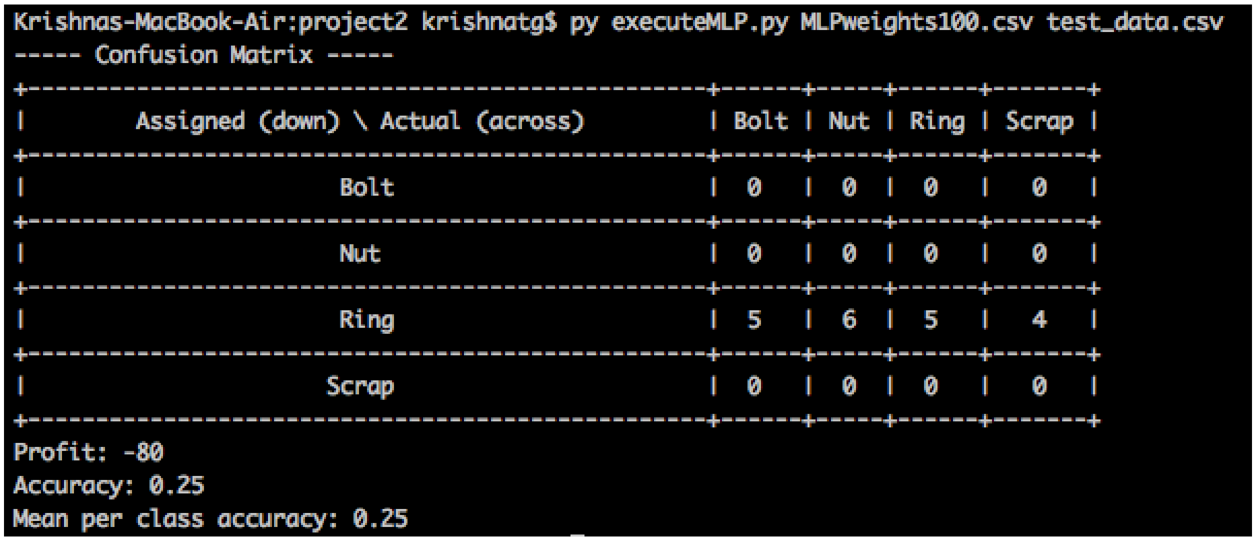


```
Krishnas-MacBook-Air:project2 krishnatg$ py executeMLP.py MLPweights10.csv test_data.csv
----- Confusion Matrix -----
+-----------------------------------------------------+------+-----+------+-------+
|          Assigned (down) \ Actual (across)          | Bolt | Nut | Ring | Scrap |
+-----------------------------------------------------+------+-----+------+-------+
|                         Bolt                        |  0   |  0  |  0   |   0   |
+-----------------------------------------------------+------+-----+------+-------+
|                         Nut                         |  4   |  0  |  0   |   0   |
+-----------------------------------------------------+------+-----+------+-------+
|                         Ring                        |  1   |  6  |  5   |   4   |
+-----------------------------------------------------+------+-----+------+-------+
|                        Scrap                        |  0   |  0  |  0   |   0   |
+-----------------------------------------------------+------+-----+------+-------+
Profit: -80
Accuracy: 0.25
Mean per class accuracy: 0.25
```

iii) Epoch 100:

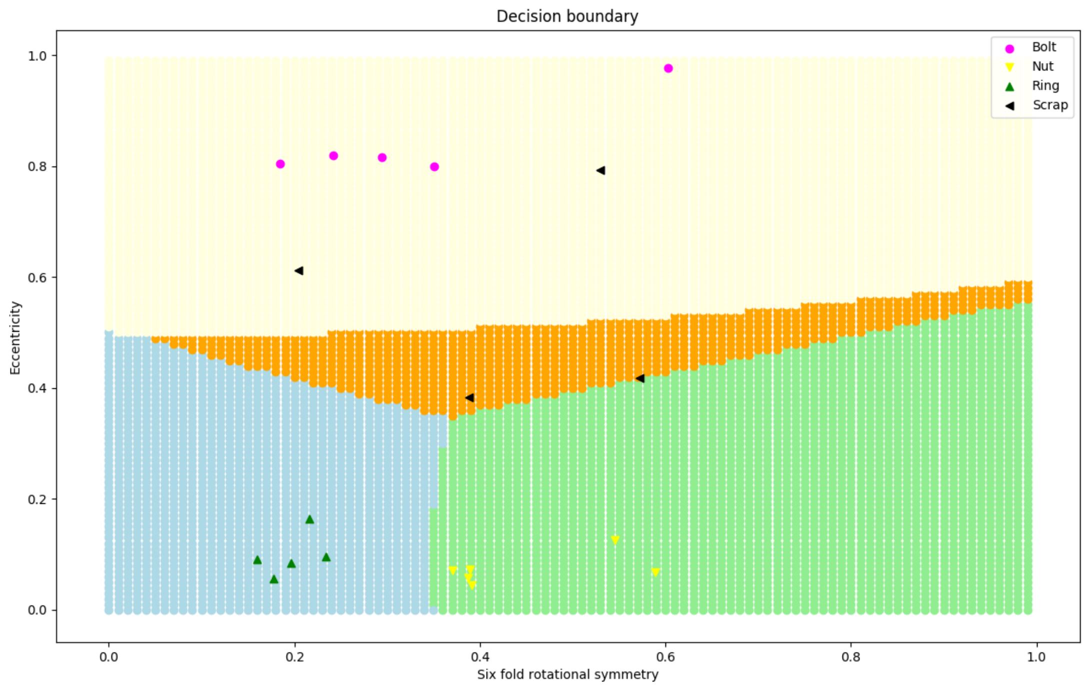


Confusion matrix:

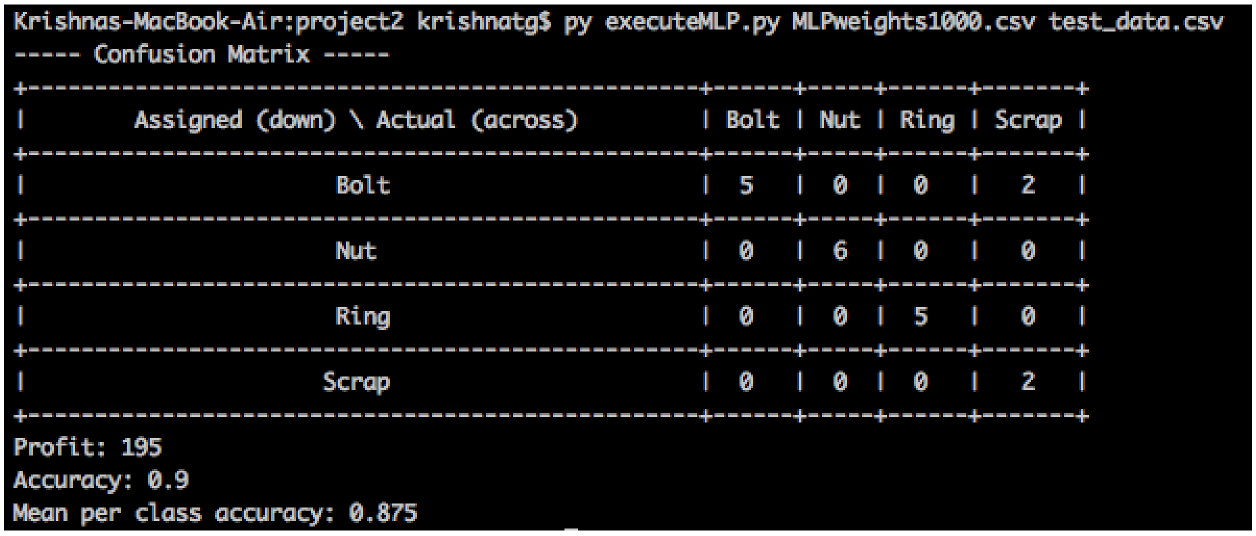


```
Krishnas-MacBook-Air:project2 krishnatg$ py executeMLP.py MLPweights100.csv test_data.csv
----- Confusion Matrix -----
+------------------------------------------------+------+-----+------+-------+
|        Assigned (down) \ Actual (across)        | Bolt | Nut | Ring | Scrap |
+------------------------------------------------+------+-----+------+-------+
|                      Bolt                      |  0   |  0  |  0   |   0   |
+------------------------------------------------+------+-----+------+-------+
|                      Nut                       |  0   |  0  |  0   |   0   |
+------------------------------------------------+------+-----+------+-------+
|                      Ring                      |  5   |  6  |  5   |   4   |
+------------------------------------------------+------+-----+------+-------+
|                     Scrap                      |  0   |  0  |  0   |   0   |
+------------------------------------------------+------+-----+------+-------+
Profit: -80
Accuracy: 0.25
Mean per class accuracy: 0.25
```

iv) Epoch 1000:

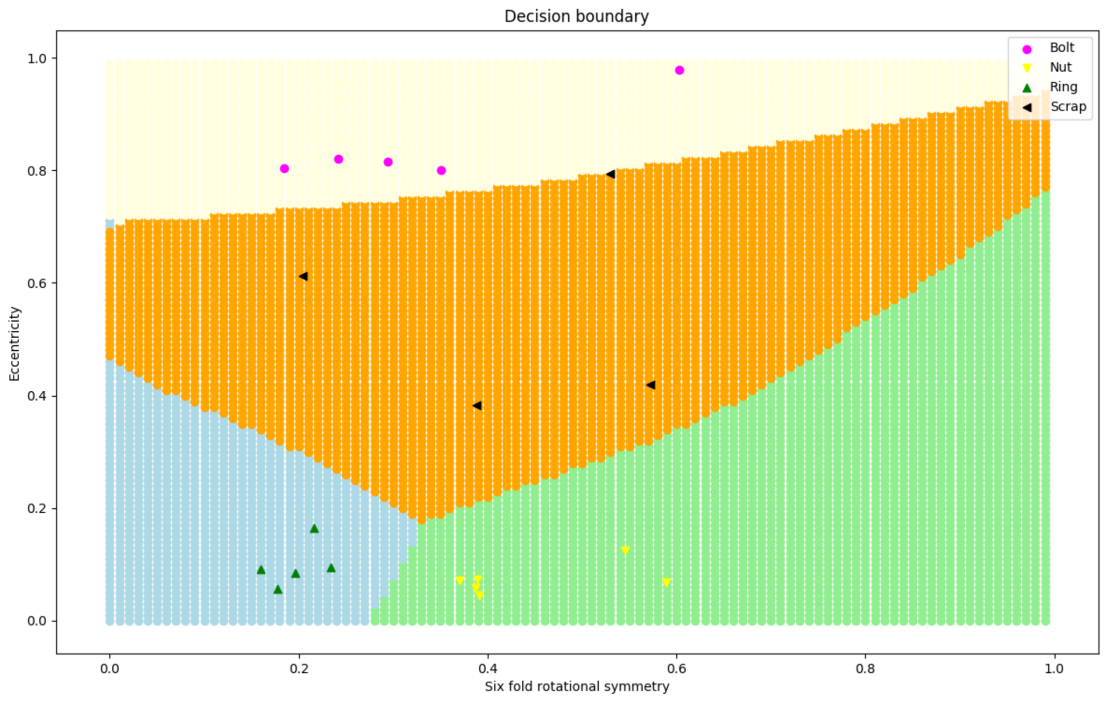


Confusion matrix:

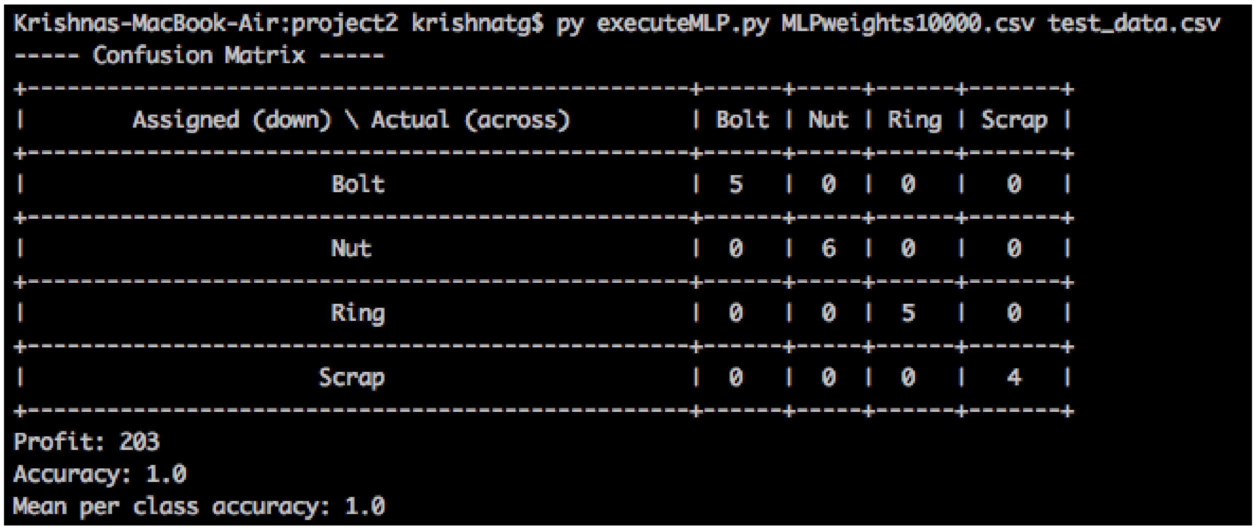


```
Krishnas-MacBook-Air:project2 krishnatg$ py executeMLP.py MLPweights1000.csv test_data.csv
----- Confusion Matrix -----
+---------------------------------------------+------+-----+------+-------+
|       Assigned (down) \ Actual (across)     | Bolt | Nut | Ring | Scrap |
+---------------------------------------------+------+-----+------+-------+
|                    Bolt                     |  5   |  0  |  0   |   2   |
+---------------------------------------------+------+-----+------+-------+
|                    Nut                      |  0   |  6  |  0   |   0   |
+---------------------------------------------+------+-----+------+-------+
|                    Ring                     |  0   |  0  |  5   |   0   |
+---------------------------------------------+------+-----+------+-------+
|                   Scrap                     |  0   |  0  |  0   |   2   |
+---------------------------------------------+------+-----+------+-------+
Profit: 195
Accuracy: 0.9
Mean per class accuracy: 0.875
```

v) Epoch 10000:



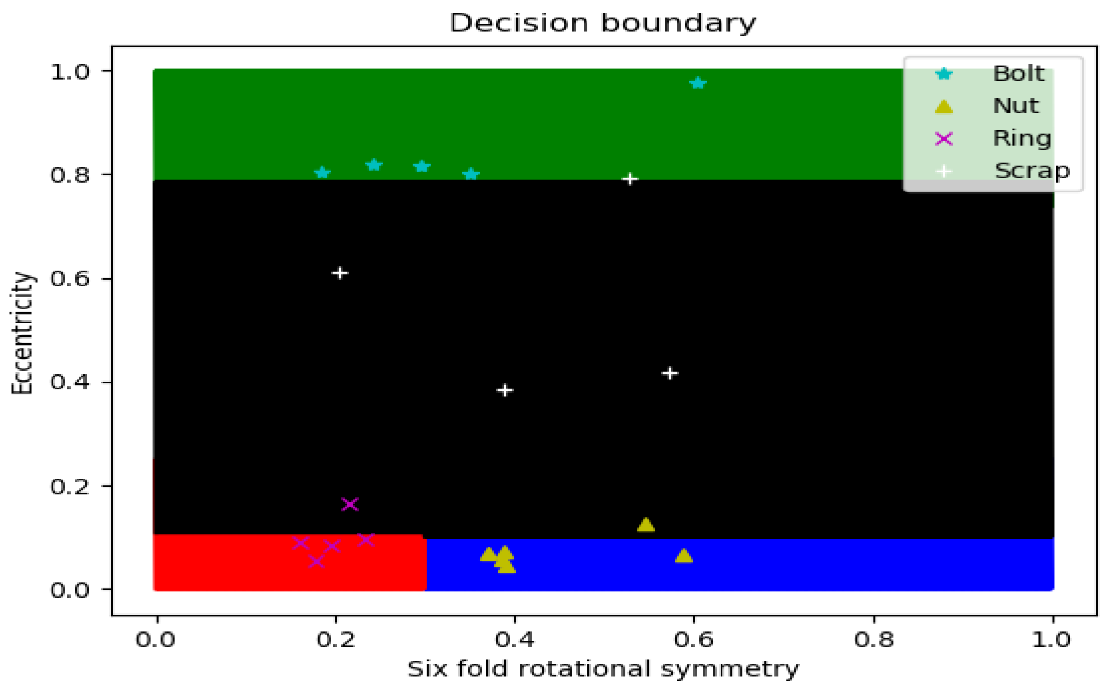
Confusion matrix:



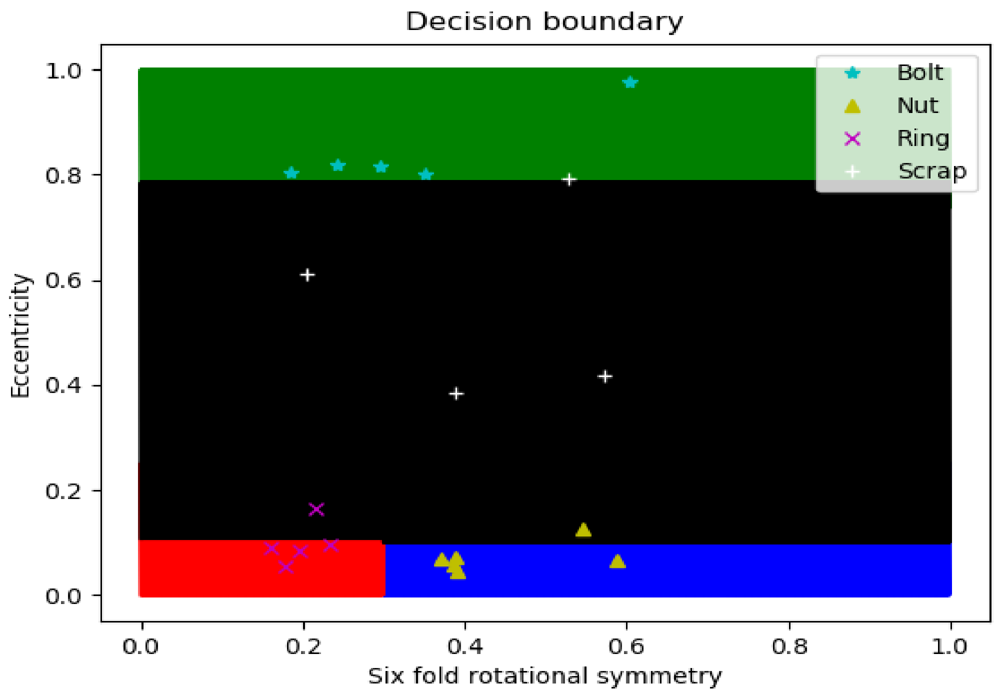
```
Krishnas-MacBook-Air:project2 krishnatg$ py executeMLP.py MLPweights10000.csv test_data.csv
----- Confusion Matrix -----
+---------------------------------------------+------+-----+------+-------+
|        Assigned (down) \ Actual (across)        | Bolt | Nut | Ring | Scrap |
+---------------------------------------------+------+-----+------+-------+
|                     Bolt                    |  5   |  0  |  0   |   0   |
+---------------------------------------------+------+-----+------+-------+
|                     Nut                     |  0   |  6  |  0   |   0   |
+---------------------------------------------+------+-----+------+-------+
|                     Ring                    |  0   |  0  |  5   |   0   |
+---------------------------------------------+------+-----+------+-------+
|                    Scrap                    |  0   |  0  |  0   |   4   |
+---------------------------------------------+------+-----+------+-------+
Profit: 203
Accuracy: 1.0
Mean per class accuracy: 1.0
```

- Decision Trees:

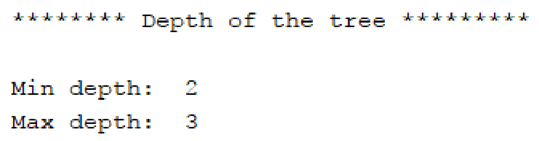
Before Pruning-



After Pruning-



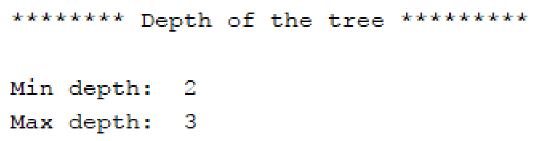
c) **The tree metrics before pruning:**



```
******** Depth of the tree *********

Min depth:  2
Max depth:  3
```

The tree metrics after pruning:



```
******** Depth of the tree *********

Min depth:  2
Max depth:  3
```

Before Pruning:

```
------------- Before pruning --------------
+--------------------------------------------------+------+-----+------+-------+
|        Predicted (down) \ Actual (across)        | Bolt | Nut | Ring | Scrap |
+--------------------------------------------------+------+-----+------+-------+
|                       Bolt                       |  5   |  0  |  0   |   1   |
+--------------------------------------------------+------+-----+------+-------+
|                       Nut                        |  0   |  6  |  0   |   0   |
+--------------------------------------------------+------+-----+------+-------+
|                       Ring                       |  0   |  0  |  5   |   0   |
+--------------------------------------------------+------+-----+------+-------+
|                      Scrap                       |  0   |  0  |  0   |   3   |
+--------------------------------------------------+------+-----+------+-------+

********** Recognition rate ***********
Mean per class accuracy is:  0.9375
Accuracy is:  95.0
***************************************

************ Profit ***********
Total profit computed is: 199
*******************************
```

After Pruning:

```
----------- After pruning ------------
+--------------------------------------------------+------+-----+------+-------+
|        Predicted (down) \ Actual (across)        | Bolt | Nut | Ring | Scrap |
+--------------------------------------------------+------+-----+------+-------+
|                       Bolt                       |  5   |  0  |  0   |   1   |
+--------------------------------------------------+------+-----+------+-------+
|                       Nut                        |  0   |  6  |  0   |   0   |
+--------------------------------------------------+------+-----+------+-------+
|                       Ring                       |  0   |  0  |  5   |   0   |
+--------------------------------------------------+------+-----+------+-------+
|                      Scrap                       |  0   |  0  |  0   |   3   |
+--------------------------------------------------+------+-----+------+-------+

********** Recognition rate ***********
Mean per class accuracy is:  0.9375
Accuracy is:  95.0
***************************************

************ Profit ***********
Total profit computed is: 199
*******************************
```

Discussion:

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

As seen in the results above, the MLP performs a little worse than the DT after 1000 epochs but after 10,000 epochs, the MLP performs perfect classification of the test data.

This meets the expectations that we had at the start. We feel that the DT performed this well (95% accurate) only because the non-linearity of the data was in one class. If the entire dataset was non-linear (across all four classes), we would have seen the DT not do as well as it did here. The MLP would have still performed exceedingly well even then.

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

In the first run, i.e. in 0 epochs, the MLP is basically running a model with randomly assigned weights. As expected, the model ran poorly. It classified all samples except one as Scrap. As seen in the result shown previously, they make no sense.

After 10 epochs, the accuracy of the model is seen to have increased to 25%, but the profit has gone down. Again, these number of iterations of learning are clearly not enough for the model to be making any sensible classifications. It classified all but one sample as a Ring.

After 100 epochs, the model is seen to have behaved exactly the same way as it did after 10 epochs. This was ideally not expected but not unusual because of the way the weights are assigned randomly at the beginning.

After 1000 epochs, we see a huge improvement in the model. The classification boundaries are quite evident, and the accuracy of the model is 90%. The profit is quite good.

After 10,000 epochs, the model classifies the testing data perfectly. It has a 100% accuracy, the decision boundaries are clear, and all the data points are plotted within the expected regions. The plot of the testing data shows that the model has not overfit; perfect classification of a dataset that was not used to train the model exhibits the lack of any overfitting.

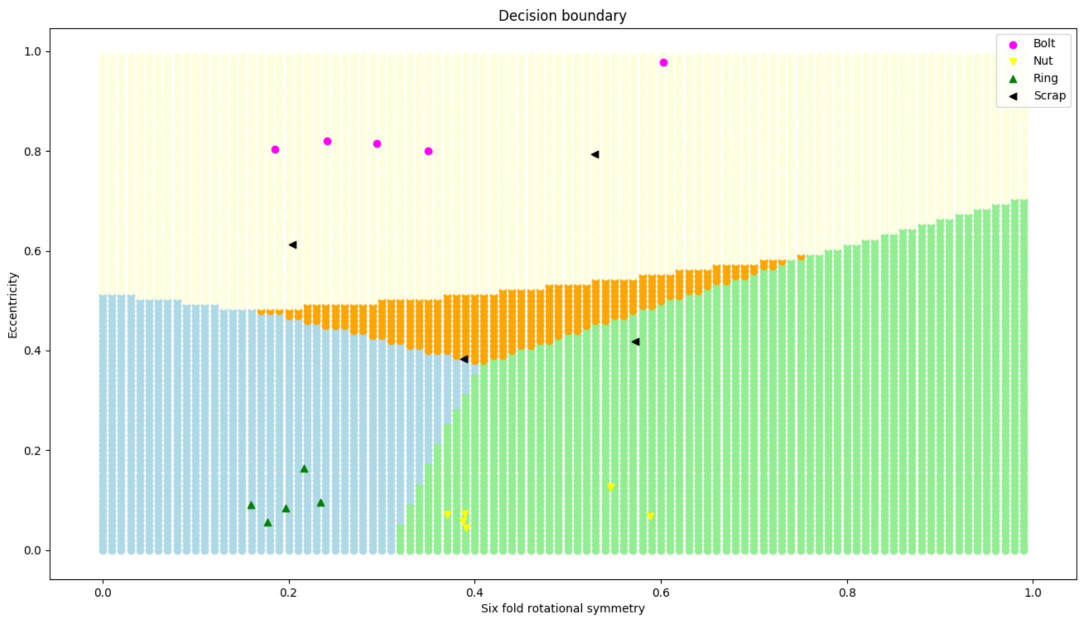
The DT is not able to figure out a relationship at higher degrees between the attributes and the labels. The DT is able to have an accuracy of 95% on the testing data after initially performing its training on the train data. Unlike MLP, we do not see an increase in the accuracy or improvement in the model when we perform multiple iterations.

Also, there is no improvement in the accuracy on pruning the tree using the Chi-Squared test with a significance level of 1%. Even though we get the same tree after pruning, we did not expect the accuracy of the model to change and it did not.

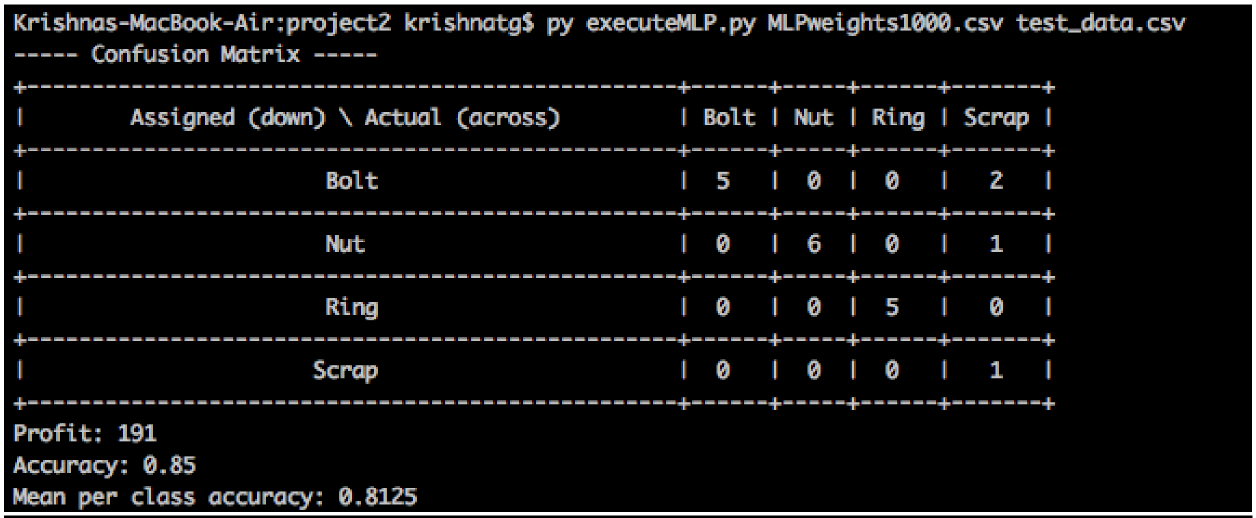
Bonus:

We ran the MLP with 3 nodes in our hidden layer and following graphs and results were seen:

Decision boundary after 1000 epochs (3 hidden nodes)



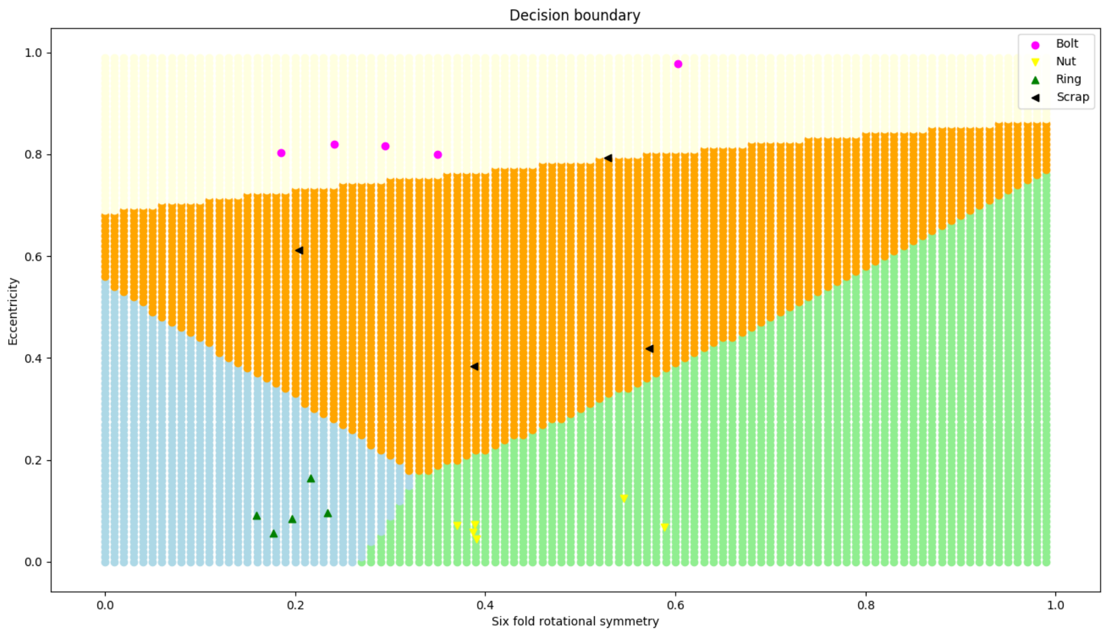
Confusion matrix



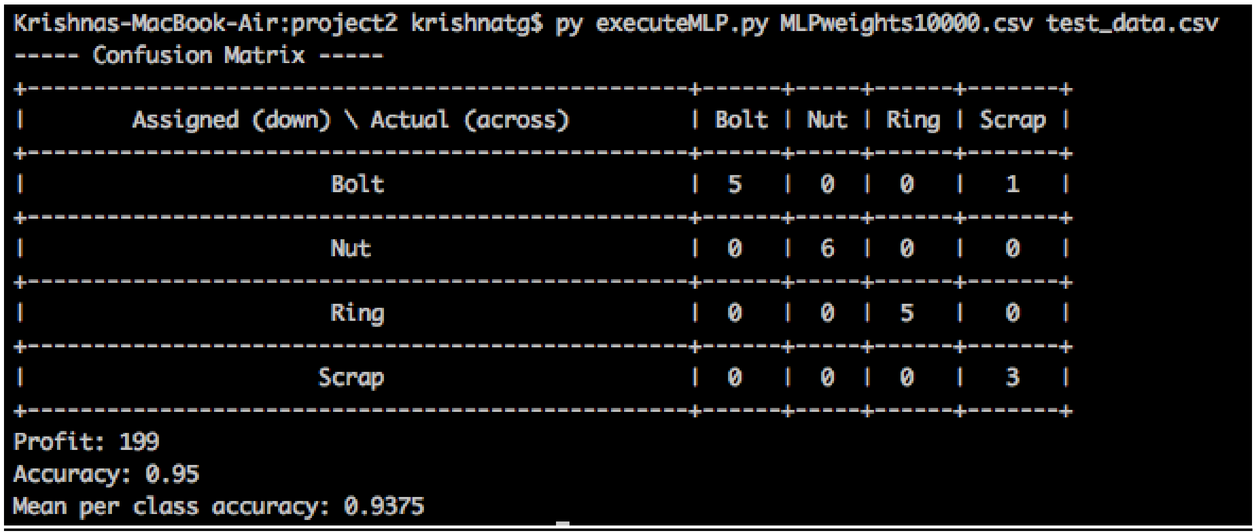
```
Krishnas-MacBook-Air:project2 krishnatg$ py executeMLP.py MLPweights1000.csv test_data.csv
----- Confusion Matrix -----
+----------------------------------------------+------+-----+------+-------+
|        Assigned (down) \ Actual (across)     | Bolt | Nut | Ring | Scrap |
+----------------------------------------------+------+-----+------+-------+
|                     Bolt                     |  5   |  0  |  0   |   2   |
+----------------------------------------------+------+-----+------+-------+
|                     Nut                      |  0   |  6  |  0   |   1   |
+----------------------------------------------+------+-----+------+-------+
|                     Ring                     |  0   |  0  |  5   |   0   |
+----------------------------------------------+------+-----+------+-------+
|                    Scrap                     |  0   |  0  |  0   |   1   |
+----------------------------------------------+------+-----+------+-------+
Profit: 191
Accuracy: 0.85
Mean per class accuracy: 0.8125
```

This network had a reduced ability to learn so the results seen after 1000 epochs is worse than what was seen with the network when it had 5 nodes in the hidden layer. After 10000 epochs, this network still showed a level of prediction accuracy lesser than the network with 5 hidden nodes.

<u>Decision boundary after 10000 epochs (3 hidden nodes)</u>



<u>Confusion matrix</u>



```
Krishnas-MacBook-Air:project2 krishnatg$ py executeMLP.py MLPweights10000.csv test_data.csv
----- Confusion Matrix -----
+---------------------------------------------+------+-----+------+-------+
|        Assigned (down) \ Actual (across)      | Bolt | Nut | Ring | Scrap |
+---------------------------------------------+------+-----+------+-------+
|                     Bolt                    |  5   |  0  |  0   |   1   |
+---------------------------------------------+------+-----+------+-------+
|                     Nut                     |  0   |  6  |  0   |   0   |
+---------------------------------------------+------+-----+------+-------+
|                     Ring                    |  0   |  0  |  5   |   0   |
+---------------------------------------------+------+-----+------+-------+
|                     Scrap                   |  0   |  0  |  0   |   3   |
+---------------------------------------------+------+-----+------+-------+
Profit: 199
Accuracy: 0.95
Mean per class accuracy: 0.9375
```