Decision Trees Report

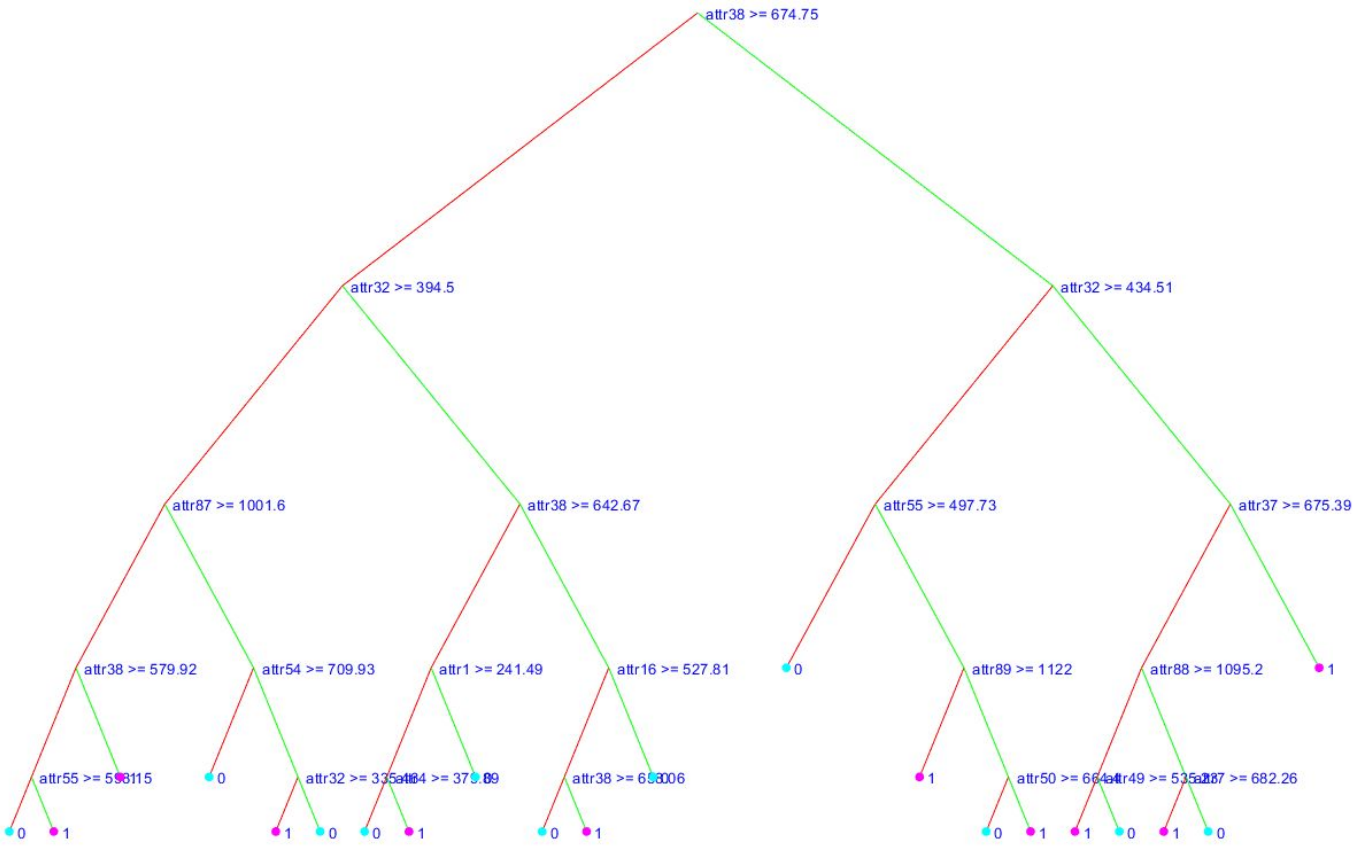
Group X

Jay Mehta
Julian Henjes
Sam Adejumo
Tom Staplehurst
James Thackway
James Armer
Gabrielle Hornshaw

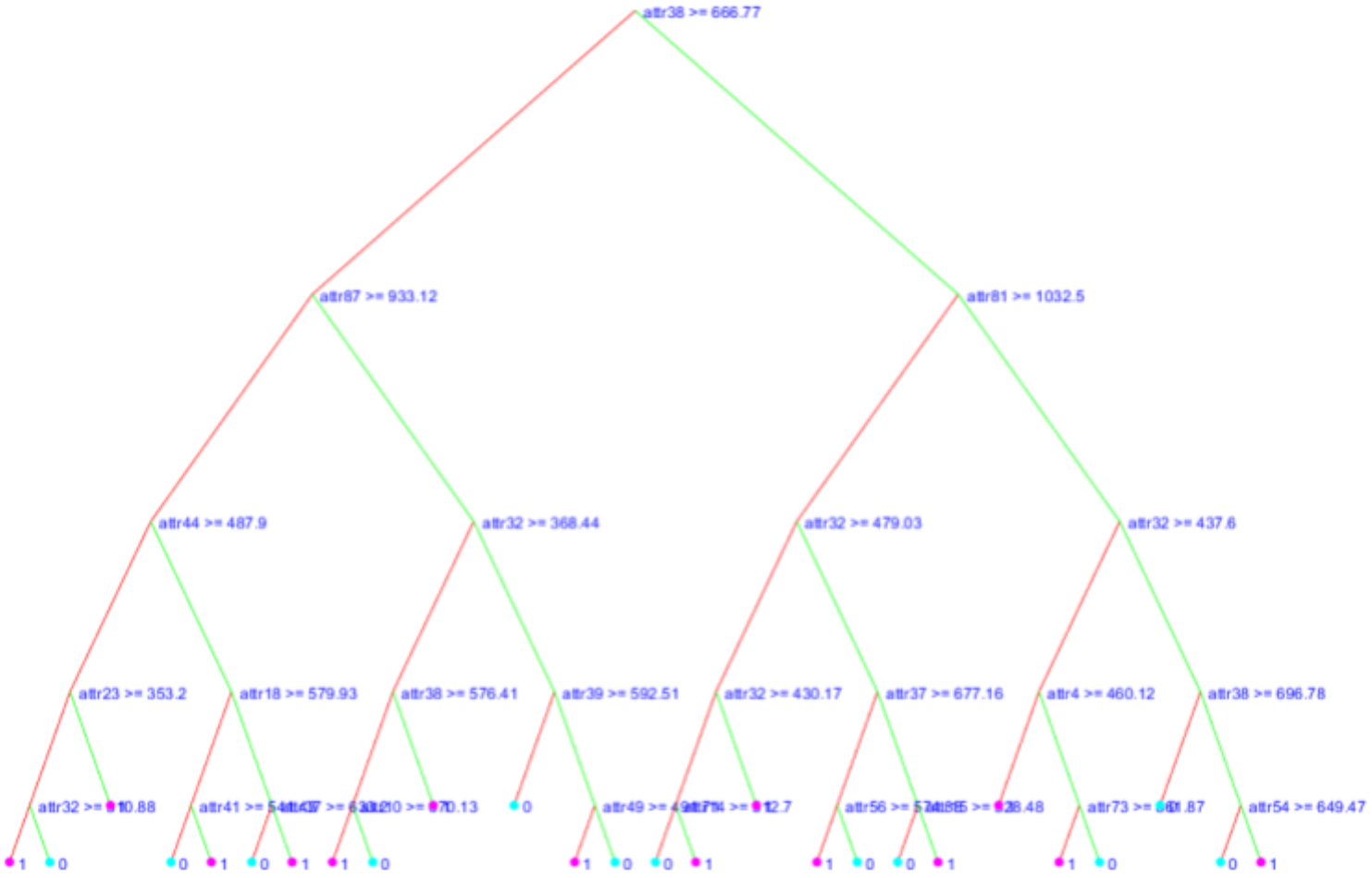
25/11/2019

Acquired Decision Tree(s)

For our final decision trees, we found that our code classified classes 1 and 3 the best (accuracies shown later on in this document). To acquire these trees we used 50,000 rows of data per input with 5 layers. We also implemented a basic pruning algorithm that would remove child nodes from a parent where they were both of the same class. This was done after training to improve the simplicity of our final trees. Below are the trees we found for Class 1 and Class 3. Similar trees can consequently be produced for the other classes. The accuracies of these trees were determined by using a 5-fold cross validation which is discussed later.



↑ Fig. 1 Final trained decision tree for Class 1



↑ Fig. 2 Final trained decision tree for Class 3

Tree Visualisation

We provided our own visualisation function, as we made Node() a class instead of a struct. We coloured nodes that classified as negative blue, and those that classified positive red (As seen in Fig 1 and Fig 2 above). Developing our decision tree this way was easier for us, as we were able to implement our training algorithm and a basic pruning algorithm.

Depth of Tree

To determine how deep to make our decision tree we tested 3, 4, 5 & 6 layers. To allow us to test this efficiently we did not use all the data available. However, we used a suitable amount to ensure our results were reliable. Here is an average accuracy across every class for each depth:

Depth	Average Accuracy over each AU
3	59.5%
4	64.3%
5	**68.1%**
6	63.4%

From these tests we gathered that a depth of 5 was the best to use, and any greater depth would be overfitting and reduce our test accuracy.

K-Fold Cross Validation

We used 50,000 rows of data per input, with a depth of 5 and 5 fold cross validation. We have also recorded the precision, recall and consequently the F1-Score. This is because accuracy does not need to be the only metric to determine how good the model is.

We decided to use 5 folds over 10 folds so that we could achieve results using a large dataset in a reasonable time. Below are the precision and recall rates for each of the 5 classes:

Class 1 (AU 6)

Fold	Precision	Recall	F1-Score	Accuracy
1	0.518	0.690	0.640	0.672
2	0.643	0.612	0.670	0.682
3	0.930	0.537	0.790	0.705
4	0.781	0.651	0.710	0.750
5	0.665	0.784	0.720	0.690

Average Accuracy for Class 1 = 69.976%
Average F1 Score for Class 1 = 0.706

Class 2 (AU 10)

Fold	Precision	Recall	F1-Score	Accuracy
1	0.473	0.677	0.557	0.532
2	0.739	0.656	0.695	0.649
3	0.759	0.897	0.822	0.7334
4	0.669	0.493	0.567	0.5144
5	0.658	0.642	0.650	0.5634

Average Accuracy for Class 2 = 59.850%
Average F1 Score for Class 2 = 0.658

Class 3 (AU 12)

Fold	Precision	Recall	F1-Score	Accuracy
1	0.813	0.668	0.733	0.707
2	0.722	0.739	0.730	0.679
3	0.840	0.631	0.721	0.686
4	0.713	0.798	0.753	0.690
5	0.856	0.742	0.795	0.753

Average Accuracy for Class 3 = 70.292%
Average F1 Score for Class 3 = 0.7464

Class 4 (AU 14)

Fold	Precision	Recall	F1-Score	Accuracy
1	0.378	0.596	0.462	0.531
2	0.598	0.482	0.534	0.634
3	0.511	0.174	0.260	0.563
4	0.505	0.574	0.538	0.525
5	0.608	0.247	0.352	0.516

Average Accuracy for Class 4 = 55.378%
Average F1 Score for Class 4 = 0.429

Class 5 (AU 17)

Fold	Precision	Recall	F1-Score	Accuracy
1	0.837	0.199	0.322	0.645
2	0.271	0.153	0.195	0.643
3	0.230	0.047	0.078	0.661
4	0.366	0.393	0.379	0.684
5	0.438	0.211	0.285	0.584

Average Accuracy for Class 5 = 64.366%
Average F1 Score for Class 5 = 0.252

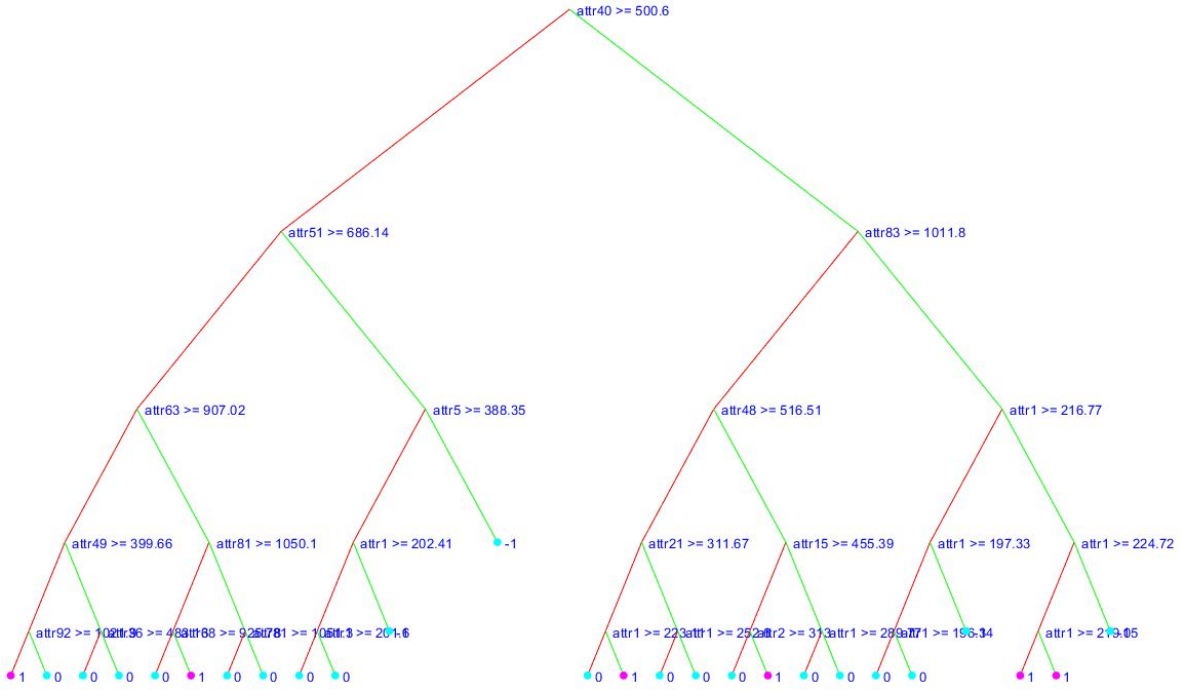
From our research we know that we want precision to be high when the cost of a false positive is high, and that we want recall to be high when the cost of a false negative is high. Given the context of our decision tree, the cost of false positive or a false negative when identifying facial features is fairly even, meaning that we would want a balanced precision and recall score - a high F1-Score.

Both of our highest accuracies had our highest F1-Scores (class 1 and 3), showing that they were balanced between precision and recall, and therefore our tree had classified that data set well. However for the 5th class the accuracy was the third highest attained but the F1-Score was the lowest, showing that the accuracy was not a good measure of how well our tree classified the data set. This could be because the accuracy was largely made up of true negatives, which aren't all that useful to use when we need to be positively classifying AUs. This could mean that there was an uneven class distribution for the 5th class, with that being the reason why our tree had a low F1-Score.

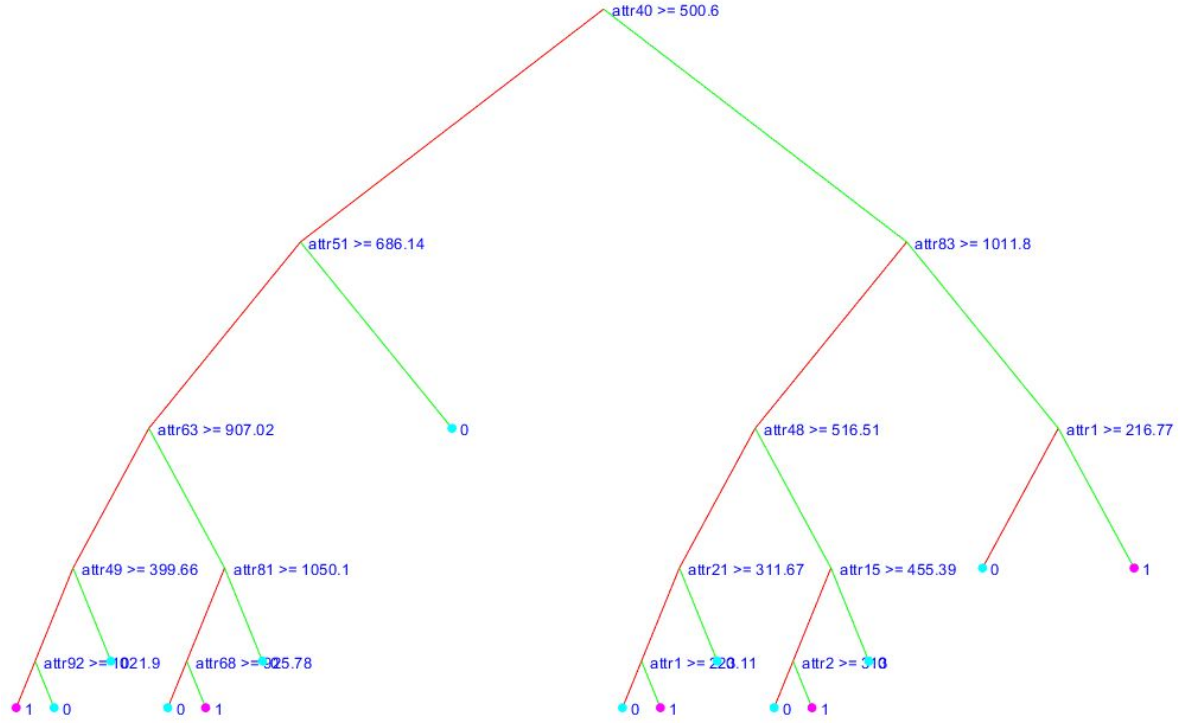
Pruning

Pruning is used on decision trees in machine learning to remove parts of the tree that do not provide power to classify instances to reduce the risk of overfitting as the larger the decision tree the bigger the risk. Following the principle of Occam’s razor, it is desirable to simplify the solution.

The way in which we prune the trees uses post order traversal until the child nodes of a node are both leaves with the same class. Under these circumstances, the parent node assumes the class of the children and removes the children.
More advanced pruning methods can solve problems of overfitting too, however our implementation considers completely redundant nodes, which only serves to improve model simplicity and validation speed.



↑ Fig 3. A tree with redundant nodes



↑ Fig 4. The pruned tree, with no redundant nodes

Multi-class Decision Trees

A binary decision tree predicts one of two potential classes, 1 or 0, at each of its leaf nodes. A singular decision tree that predicts multiple binary classes would have sets of classes that grow exponentially with the number of classes it's trying to predict (2^n , where *n* is the number of classes. In this case of 5 binary classes there are 32 different permutations of sets of classes from [0,0,0,0,0], [0,0,0,0,1]... [1,1,1,1,1], where each leaf node would predict one of these sets of binary classes.

The ID3 search query would have to be extended to also look at the best split in each of the classes, adding an extra dimension to the search. This would result in an increased time cost to training.

The accuracy of the tree would also be decreased, for example using a dataset of 80,000 features, there will be roughly 40,000 examples of class 0 and 40,000 example of class 1 (assuming the dataset is close to being evenly balanced). With 32 different potential sets of classes there would be roughly 2,500 examples of each permutation (assuming classes are independent as stated in the lab manual), this resulting small dataset of examples for each permutation would make predicting more difficult. A minimum of 5 layers would be needed to have only a single leaf node representing each permutation, so many more layers would be needed to have even slightly accurate tree.