Imperial College London

Decision Trees Report

Imperial College London

Department of Computing

Introduction to Machine Learning

Authors:
Niusha Alavi, Olle Nilsson, Paween Pitimanaaree, Alfred Tingey

Date: November 11, 2019

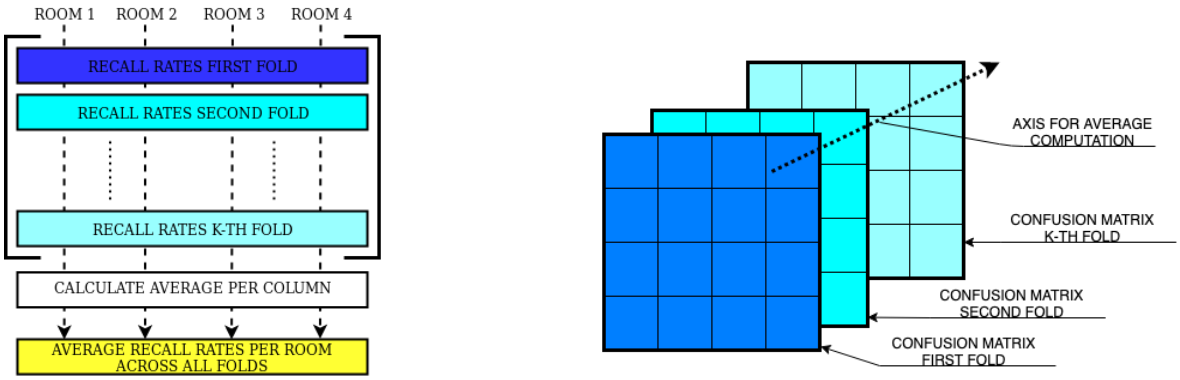
1 Method of Implementation

1.1 Decision Tree Algorithm

To implement the decision tree for the given dataset we defined five main functions to recursively split our dataset into left and right datasets by maximizing its gain value. Our implementation is as follows:

- Firstly, we defined a function that computes the entropy value of our data, denoted by: $H(dataset) = -\sum_{k=1}^{k=K} p_k * log_2(p_k)$ where K is the total number of labels in our data, and p_k denotes the proportion of label k in our data.
- To split the dataset into 2 parts with the most information gain, we calculate the gain for all attributes and values. Then we compare all the information gain for each attribute and pick an attribute and value with the highest gain.
- For each attribute, we sort the data by the value of the attribute in ascending order. Then we split the data at every unique value of the attribute and choose the split that maximizes the information gain following the formula: $Gain(S_{all}, S_{left}, S_{right}) = H(S_{all}) - Remainder(S_{left}, S_{right})$ where $Remainder$ denotes the sum of the information gain of the left and right datasets.
- Finally, we create the decision tree by splitting the data recursively by calculating the highest information gain in each dataset that we create.

1.2 10-fold Cross Validation



(a) (b)

Figure 1: Averages Calculation Methods: (a) Label specific metrics, using recall rate as an example. (b) Confusion Matrix.

For the validation of the decision trees a general K-fold algorithm was implemented. The data set is split in K equal size folds (K=10 was used here). One fold is put aside as validation data and a decision tree is trained on the remaining 9 using the algorithm described in section 1.1.

The validation data is then evaluated using the trained tree and a confusion matrix, which is populated based on the predicted and known true labels for each sample. From the confusion matrix the recall rates, precision rates and F1 scores for each room are calculated, together with the overall classification rate as follows:

Classification rate: the number of correctly classified examples divided by the total amount of examples. Additionally, we can use the classification error as a metric: $Error = 1 - CR$.

Recall: the number of correctly classified positive examples divided by the total number of positive examples.

Precision: the number of correctly classified positive examples divided by the total number of predicted positive examples.

F-1 Score: F-1 $= 2 * (Precision * Recall)/(Precision + Recall)$

This process is repeated using each of the 10 folds as the validation data. For each fold the confusion matrix and label specific measures such as recall are stacked in arrays, as illustrated in Fig. 1, and the average across all folds is calculated by averaging along the first dimension of these arrays.

1.3 Pruning Algorithm

To implement the pruning of our tree we used a **depth first search** to find the nodes that are beneficial to prune. We use pruning to stop our decision tree from over-fitting to our dataset. Our implementation is as follows:

- We search through the tree to find a node that is a parent of two unchecked leaves which we haven't attempted to prune before. This node is changed into a leaf with a label determined by the majority label of samples in the subset of the training set used to build that node's sub-tree structure.
- A copy of the tree before pruning is kept and a marker is put on the node that we just pruned such that the next iteration of the algorithm knows not to try and prune that node again.
- The classification rate of the unpruned and the pruned tree are compared and the best tree is chosen. If the classification rates of the pruned and unpruned trees are the same, then we choose the pruned tree to reduce complexity.
- If the pruned tree is better, it is kept and the tree is searched again to find the next node that can be pruned. We then repeat the process above across all nodes that can be pruned. We carry forward any prune that performs better than the previous iteration of the tree.

1.4 10-fold Cross Validation with Pruning

A similar method as described in section 1.2, integrated with the pruning algorithm from section 1.3, is used to evaluate the effect of pruning on the accuracy of the decision trees. Each tree that is trained is pruned to find the configuration of pruned nodes with the best performance on the validation data. This means that we are fitting our tree to the validation data and therefore need to put aside an additional fold of test data to evaluate the performance of the pruned tree on data that is separate from both the training data and the validation data, which effectively means running through the K-fold Cross Validation described in section 1.2, with $K = 9$, ten times. We do this once with each of the ten folds as the test data.

2 Results 10-fold Cross Validation

2.1 Clean Dataset Results

The cross validation results for the clean data set are shown by the average confusion matrix in Table. 1 and the averages of the various performance metrics are shown in Table. 2. In general the results are very accurate for both the overall score and the label specific measures. Only room 2 and room 3 are confused for each other on more than one occasion on average.

Room	1 Predicted	2 Predicted	3 Predicted	4 Predicted
1 Actual	50	0	0	0
2 Actual	0	47	3	0
3 Actual	0	2	47	1
4 Actual	0	0	1	49

Table 1: Average Confusion Matrix for Clean Data. Entries Rounded to Nearest Integer

Metric	Room 1	Room 2	Room 3	Room 4
Recall	0.989	0.948	0.942	0.979
Precision	0.983	0.963	0.933	0.979
F1	0.986	0.955	0.937	0.978
CR	96.4%			

Table 2: Metrics for Clean Data

2.2 Noisy Dataset results

The cross validation results for the noisy data set are shown by the average confusion matrix in Table. 3 and the averages of the various performance metrics in Table. 4. The results are

far more confused, and there is a general performance decrease in the prediction of all rooms.

Room	1 Predicted	2 Predicted	3 Predicted	4 Predicted
1 Actual	40	3	3	3
2 Actual	2	41	4	3
3 Actual	4	3	41	3
4 Actual	3	2	4	41

Table 3: Average Confusion Matrix for Noisy Data. Entries Rounded to Nearest Integer

Metric	Room 1	Room 2	Room 3	Room 4
Recall	0.796	0.800	0.811	0.810
Precision	0.799	0.820	0.790	0.809
F1	0.795	0.808	0.800	0.808
CR	80.4%			

Table 4: Metrics for Noisy Data

2.3 Clean vs. Noisy Results

The performance results of our decision tree on the noisy data is far worse than on the clean. This is consistent over all rooms and performance measures. The overall classification rate of our algorithm performs 16% worse on the noisy data than on the clean data.

The reason that our algorithm performs worse on the noisy data lies in the nature of the decision tree algorithm, which will always attempt to find a pattern between the attribute values (WiFi-signals) for samples and their corresponding label. Noise may cause samples with different labels to have similar attribute values, which will cause contradictions in labels of similar samples and thus cause the algorithm to create further splits in the data set in order to perfectly classify the training-data. Consequently, this will cause problems in generalising the classification to unseen data since noise is random. The additional splitting of our data due to noise in the training data set is unlikely to follow the general trend of the "good" samples in our data set and will cause an increased rate of miss-classification on unseen data from the same distribution.

3 Results 10-fold Cross Validation and Pruning

To increase the accuracy of our decision tree on noisy data we have to introduce pruning on our decision tree to attempt to locate the splits caused by noisy examples and remove them. The cross validation results for pruned trees are shown by the averages of the various performance metrics in Table. 5 for the clean data set, and in Table. 6 for the noisy data set.

3.1 Clean Data set

Metric	Room 1	Room 2	Room 3	Room 4
Recall	0.997	0.948	0.937	0.979
Precision	0.977	0.961	0.932	0.992
F1	0.987	0.954	0.934	0.985
CR	96.5%			

Table 5: Metrics for Clean Data with Pruned Decision Tree

3.2 Noisy Data set

Metric	Room 1	Room 2	Room 3	Room 4
Recall	0.906	0.871	0.856	0.881
Precision	0.877	0.886	0.865	0.889
F1	0.890	0.877	0.859	0.884
CR	88.0%			

Table 6: Metrics for Noisy Data with Pruned Decision Tree

3.3 Influence of Pruning on Results

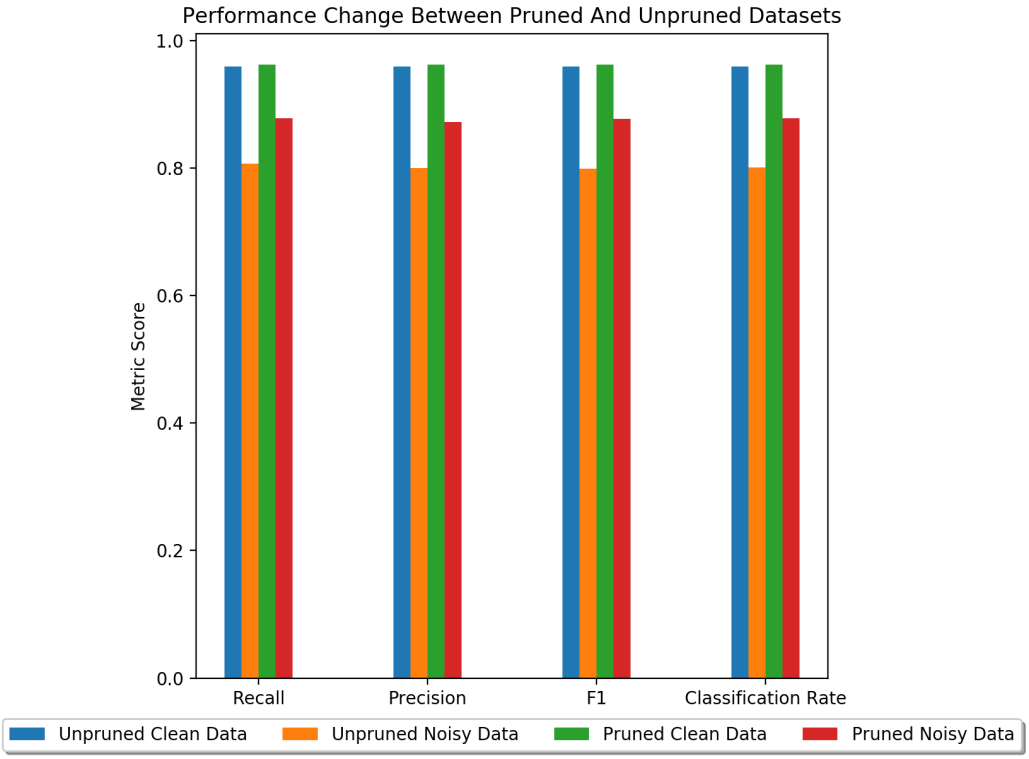


Figure 2: Influence of Pruning on Noisy and Clean Data

Pruning our decision tree has a different influence when applied to the clean and noisy datasets. On the one hand, for the clean dataset, pruning has a negligible effect on the clean data with an average increase of only 0.1% on the test data. On the other hand, pruning significantly increases the accuracy of our decision tree on the noisy dataset by an average of 7.6%. Pruning our noisy dataset reduces the difference in accuracy of classification between the noisy and clean datasets. There is a significant increase in performance on our noisy dataset when the decision tree is pruned because there are lots of 'noisy examples' in our noisy data. This means that the training data distribution might be different from the test data. Therefore, the decision tree has a higher chance of over-fitting the training data and does not generalize well on the data. The results in the change of performance of pruning our decision trees for the clean and noisy datasets are shown in Fig. 2.

3.4 Influence of Maximal Depth on Results

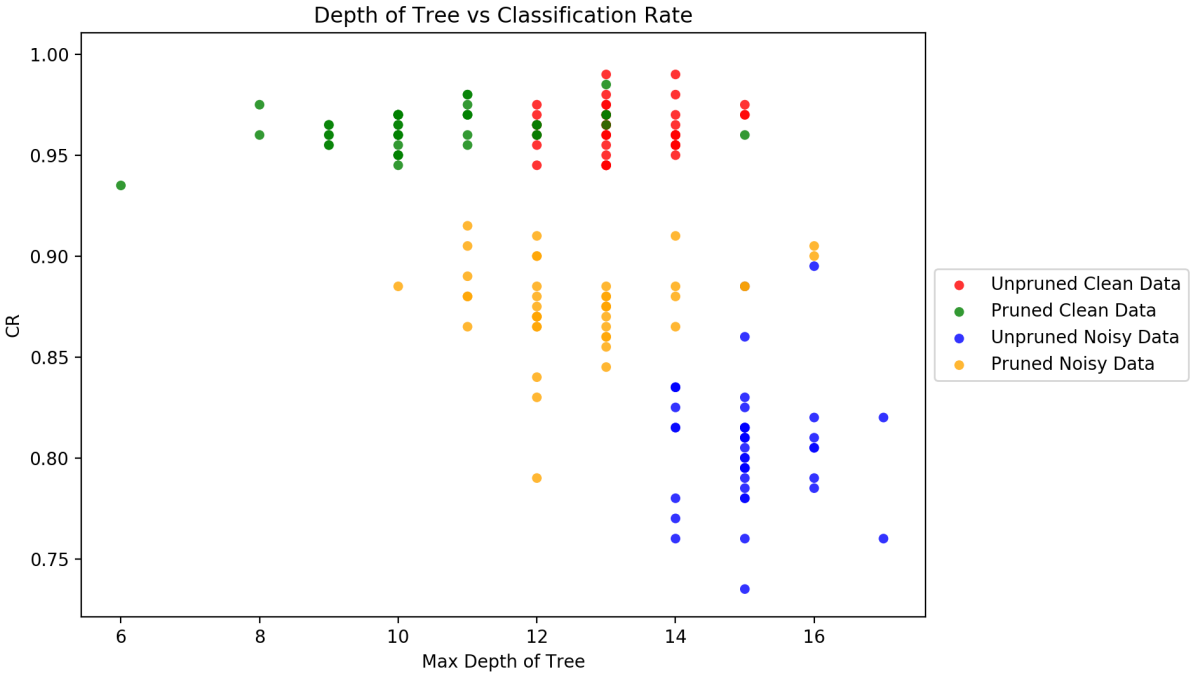


Figure 3: Depth vs Classification Rate of Different Decision Trees on Datasets

Fig. 3 shows the depth and classification rate of 40 randomly selected decision trees from each of the unpruned clean and noisy data and their pruned counterparts. Clearly, pruning trees trained in noisy data will in general improve the performance, while trees trained on the clean data seem fairly invariant to pruning. Since we only continue to prune while the classification rate continues to improve, we can conclude that pruning to even lower depths will start to reduce the classification rates. Therefore there seems to exist a maximum for the classification rate at some depth of a decision tree trained on a given data set, where below and above this depth the classification rate will drop. For the clean data, the trained trees seem to end up close to this depth during training, but for the noisy data the additional splitting of the data due to noisy samples causes the tree to be trained to a higher than optimal depth. This means that the tree will overfit the data and, thus, generalises poorly to unseen data, leading to worse classification rates on the test data.

In conclusion, for any real application using decision trees where noise in the data is inevitable, pruning will be an essential step. This is because the prediction accuracy is significantly better when reducing the depth of trees trained on noisy data.

4 Visualising the Decision Trees

The following figures show the visualisation of our decision trees acting on the noisy and clean data. We only show the decision trees up to a maximal depth of 5.

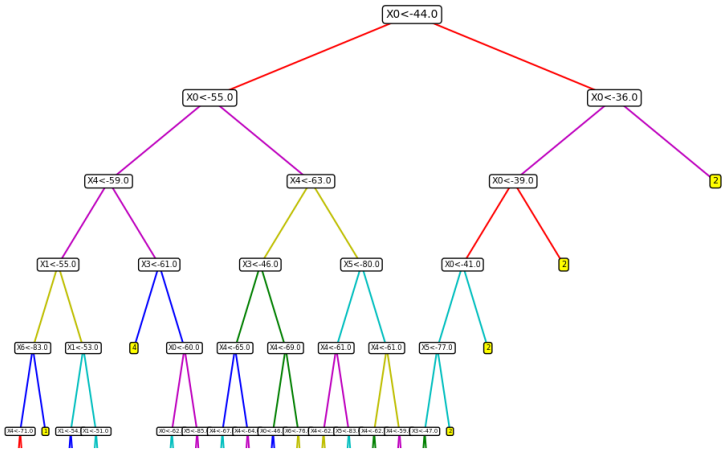


Figure 4: Decision tree trained on clean data set - partial depth

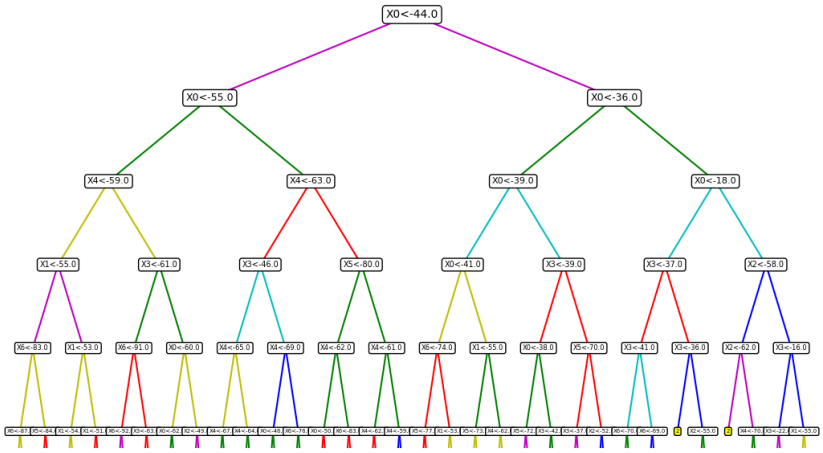


Figure 5: Decision tree trained on noisy data set - partial depth