Imperial College London
395 Machine Learning CBC Report
Assignment 2: Decision Trees

Marta GARNELO ABELLANAS[1], Patrick GLAUNER [*1], Beatriz Isabel LOPEZ ANDRADE[1], and Anthony WILSON[1]

[1]Group 2

October 30, 2014

Abstract

This report describes the implementation of a decision tree learning algorithm for emotion recognition based on action units. Two datasets are used for the training process: one with clean and one with noisy data and the optimal decision trees for each set are selected by 10-fold cross-validation. The resulting decision trees are then evaluated using various performance measures and discussed in the context of the questions proposed by the assignment.

1 Introduction

Decision trees are powerful and widely used learning algorithms with many applications in academia and industry. They can be applied to classification and regression problems. In this assignment, binary decision trees based on ID3 are used and combined. Other decision tree learning algorithms however, support multi-class classification.

2 Implementation

This section covers various implementation parts that are of special interest. Other key parts are covered later in this report, as part of the results and evaluation in Chapter **??** and in the Q&A in Chapter 4.

Information theory is key to decision tree learning based on ID3 to select attributes with the highest **importance** or **information gain**. This can be expressed in terms of **entropy**. For a subset of the training set containing only positive or negative examples, it was encountered that MATLAB computes $log(0) := NaN$ - not a number - instead of throwing an exception. This caused the learning algorithm to compute very large decision trees and therefore needed to be handled with an extra condition in the entropy calculation. Computing the information gain is subsequently straight-forward and is done for all remaining attributes from which the with the largest information gain is chosen. If there are multiple attributes with the same maximum information gain, the last attribute of these candidates is chosen in a loop. As this is done for implementation reasons, choosing any other candidate would generate a slightly different decision tree. This strategy was not further evaluated as a uniform distribution of training examples is assumed and a preference for the last candidate should therefore not make a difference on sufficiently large datasets.

In this coursework, **10-fold cross-validation** is used. The examples are split into ten sets of equal size. For the 1004-example test set, this makes ten folds that include 100 different test examples each

*Group leader

and the remaining four examples are left unused to allow all folds to have exactly the same size. For each fold of the cross-validation process, 10% of the examples are retained for **testing**, another 10% for **validation**, and the remaining 80% are used for **training**. It is important to note that for each fold, different validation and test sets are used, and they do not overlap with other validation and test sets from different folds, respectively. This functionality is implemented in the function `generate_folds`, which calculates the fold size and divides the examples into the sets by computing the indexes of the validation, test, and training sets for each of the ten folds.

Once the trees are obtained for a given training set, predictions are made by the predictor and the **performance measures** are computed. All evaluation measures code is in a separate package `+evaluation` so that it can be easily reused in the following assignments. Computing the confusion matrix given the actual values and the predictions is done by MATLAB's `confusionmat`.

In cross-validation, the **average classification rate** of the six classes is carried out for each of the ten folds:

$$Rate_{Classification}(D) = 1 - Error(D) \tag{1}$$

The error per class on the training set can be computed as defined in [2]:

$$Error(D) = \frac{1}{N_{test}} \sum_{n=1}^{N_{test}} (1 - correct(h(x^{(i)}), x^{(i)})) \tag{2}$$

The average classification rate is the mean of the classification rates of the six classes. The set of trees with the highest value among all the ten sets is chosen as the best classifier. Then, the test error of this set of trees is calculated using the test set of examples of the corresponding fold. Note that the test sets of the rest of the folds are not used at all.

3 Results and evaluation

Three different predictors were implemented and evaluated. They are explained in detail in Chapter 4 as part of the Q&A. Based on performance measures explained later in this chapter, one of them was chosen. The decision trees shown in Figures 1, 2 and 3 are the trees learned for the entire clean dataset using that predictor. The decision trees shown in Figures **??**, **??** and **??** are the trees learned for the entire noisy dataset. Comparing the trees learned for the clean and noisy datasets, one can see that in average the trees for the noisy dataset are larger. This makes sense as noisy data is less general and requires more specific trees.

Tables **??**, **??** and **??** contain the performance measures of each predictor generated by cross-validation for the clean dataset. In addition, tables **??**, **??** and **??** contain the corresponding performance measures for the noisy dataset. The first performance measure is the **confusion matrix**, a structure where columns represent instances of a predicted class and rows the actual number of examples in that class. This type of visualization is especially useful for detecting confusions between classes. From the confusion matrices obtained with the trees from the clean data we can see that there are no outstanding confusions between classes for none of the three different classifiers. The majority of the labeled examples lie along the diagonal, which means they have been correctly classified and the remaining misclassifications are spread relatively evenly across the classes. Upon closer inspection methods one and two seem to have more similar effects, as they have three classes (1, 3 and 6) that yield almost identical results. In the case of the trees trained with noisy data the values behave similarly, although the number of correctly classified examples along the diagonal has decreased overall. Classes 1 and 5 seem to be the most problematic for all classifiers, most likely as a consequence of being underrepresented in the noisy dataset.

While confusion matrices are quite visual they are too complex for direct comparison with other classifications, especially as the number of classes increases. **Classification rate**, **recall**, **precision** and **F1 score** are single values that result from the confusion matrix and therefore provide comparable measures of a classification. The recall value describes the amount of examples belonging to a class that were correctly assigned to that class. Precision on the other hand measures how many of the examples within a predicted class actually belong to it. In the case of clean training data most values lie within the range of 0.7-0.95. An exception to this are the labels from classes 1 and 5 that exhibit lower values in some of the cases. This could again be a result of low numbers of examples from that class, as both class 1 and

5 are also underrepresented in the clean data. Another class that has fewer examples in comparison to the rest of the clean data is class 3 and while the effects of this are less striking than for class 1 and 5, both the recall and precision for this class are overall lower than the rest. If we observe the trees from classes 1 and 5 they both contain an exceptionally long single branch in their structure. This type of long branch increases the size of the tree and in this case also correlates with worse classification rates. The trees obtained from noisy data provide similar results. The values are again overall high ranging from 0.6 to 0.95. The outliers are classes 1 and 5 as indicated by the confusion matrix above. In the case of the second classifier this is especially prominent as the precision has decreased from 0.73 to 0.23 in class 1 and from 0.58 to 0.43 in class 5 between the clean and the noisy data sets. The recall also dropped significantly from 0.64 to 0.40 and from 0.70 to 0.40 for each rating respectively.

As a result of this, the classification rate, a measure of how many predictions were correct out of the total number of examples, varies significantly between data sets. Classification method number two that had the highest average classification rate with the clean data set, for example, has now the lowest classification rate of the three methods. This might be a consequence of the algorithm itself, as it chooses the class with the shortest decision path in case of ambiguities. For clean data this method should work fine, but if there is noise in the tree a wrong attribute will have a bigger impact on a short decision path. The average classification rate for classifier number three, that selects the classes with the longest path, shows a different trend. While the average classification rate for the noisy set is lower than the one for the clean set, it is still significantly higher than the noisy one from the second algorithm.

The classifier that exhibits the smallest decrease in the average classification rate, however, is classifier number one. A possible explanation is the fact that this algorithm assesses the classification depending on the size of the tree. As we have seen before the trees that seemed the most problematic for classification are the ones that have single long branches that result in bigger trees. By sanctioning these trees by a factor proportional to their size this problem should be reduced. We have therefore chosen the first classifier as our final classifier. We also believe that it is more realistic to develop a classifier that is strong with noisy data rather that one that only provides the right results with clean data (as is the case with classifier number two). Furthermore, as noisy data are omnipresent in engineering an algorithm that is robust to noise will have more real-world applications than one for clean data.

4 Q&A

Noisy-Clean Datasets Question
Is there any difference in the performance when using the clean and noisy datasets? If yes/no explain why. Discuss the differences in the overall performance and per emotion.

Please see Chapter **??** for a comprehensive comparison.

Ambiguity Question
Each example needs to get only a single emotion assigned to it, between 1 and 6. Explain how you made sure this is always the case in your decision tree algorithm. Describe the different approaches you followed (at least two) to solve this problem and the advantages/disadvantages of each approach. Compare the performance of your approaches on both clean and noisy datasets and explain if your findings are consistent with what you described above.

For examples which were classified as positive by multiple different trees, we implemented and tested three different approaches for choosing a final classification. The first approach was to take all the trees that classified the example as positive, sum the nodes in each tree, and chose the minimum. The corresponding emotion will then be used as the final classification. The tree with the least nodes is likely to provide the most general hypothesis and therefore is least likely to be susceptible to overfitting. A problem with this approach is that as the number of trees to compare grows, and when the trees themselves are very large, this operation can be more expensive in terms of computation than alternative solutions.

Our second approach is to observe the leaf node in each tree which provides the final classification for the example, and compare the depths (i.e. the number of nodes along the path to classifying the example in each tree). As the number, and size of the trees grow, this solution will be computationally less expensive compared to the first solution, but we are still left with the decision of whether to take the path with the maximum or minimum length. By choosing the maximum, we are selecting a very specific hypothesis potentially vulnerable to overfitting. In theory the minimum should provide a more accurate

classification, and by ignoring the overall size of the tree and focussing on only the nodes in the tree that are directly encountered by the example, we might receive a more accurate classification compared to using the first solution.

In a similar way, if none of the trees classifies a given example, the number of nodes along the path to classify this new example is taken into account. In this case, the emotion assigned is the one corresponding to the longest path, as the decision provided by the classifier should be the least reliable.

In practice however, the results were not quite as expected. Solution one (takes min total tree nodes) returned an error of 26% on the clean data and performed best on the noisy data, with an error of just 29%. The solution which in theory is favorable, performed very well on the clean data (error of 23%), but worst on the noisy data (error of 35%). The third solution (max nodes in path) gave an error of 25% for clean data, and 30% for noisy data. Subsequently, solution one was chosen as the best predictor, as it performs the best on the noisy dataset and still reasonably on the clean one in comparison to the other predictors.

Pruning Question
Briefly explain how the `pruning_example` *function works. One figure with two different curves should be generated for each dataset (clean and noisy). Include the two figures in your report and explain what these curves are and why they have this shape. What is the difference between them? What is the optimal tree size in each case?*

Utilizing MATLAB functions, `pruning_example` learns a decision tree for a training set and the corresponding classes. It expects the tree's error rates for a pruning sequence of that tree. The stronger the pruning, the less terminal nodes of a tree are left. Figure 4 contains the **learning curves** for both datasets. They describe the error rate of a learning algorithm for a growing amount of terminal nodes in the tree. The red learning curve is the **training error**. For more parameters - or terminal nodes in this case - the training error usually decreases and can even go down to 0 for sufficiently many parameters [1]. This can be seen in both graphs. For a noisy dataset, more terminal nodes are required to fit the decision tree model to the dataset to have a zero training error. The more parameters, the more likely such a model is prone to **overfitting**. Overfits only match the training data well, but fail to generalize to new examples. The blue learning curve is the **test error**. The more a model overfits, the more the training error usually increases. This can also be seen in both graphs with a larger test error for the noisy dataset than for the clean dataset. Therefore, a pruned tree reduces the overfitting and subsequently the test error decreases.

The used function `classregtree.test` also returns "`bestlevel` containing the estimated best level of pruning" [3], which is the expected best number of terminal nodes. For the clean dataset it estimates $N = 8$ and for the noisy dataset $N = 12$ terminal nodes. Another good selection strategy for the optimal number of tree nodes is the global minimum of the test error curve. For the clean dataset this is $N = 57$ and for the noisy one $N = 52$. In comparison, this leads to bigger trees which may be more likely to overfit. Furthermore, longer tree traversals slow down predictions. As stated in the MATLAB documentation, the selection of test examples, or more precisely cross-validation examples in the MATLAB context, includes randomness and results can slightly vary in different runs.

5 Conclusions and prospects

Overall, decision trees are useful tools for classification. Benefits of these algorithms include the fact that changes made to the algorithm can be understood by direct comparison of the structures before and after the respective changes. Furthermore, decision trees are very visual which allows for easy understanding of the use of information theory in these types of classifiers.

We have generated a decision tree with average classification rates of 74% and 71% for clean and noisy data respectively. We are particularly pleased with the relatively small increase of 3% error rate, when the classifier is applied to noisy data. This implies that our algorithm would perform well on real-world data that may contain incorrect or missing values.

The final structure of that tree was retrieved using 10-fold cross-validation and the specific algorithm of the chosen classifier outperforms the other two predictors that we implemented. Interestingly, this predictor calculates the classification by taking into account and penalizing large trees. This method

proves to be successful even when applied to data sets that are noisy. As a result of this robustness, the classification rates between clean and noisy data are less prominent than in the case of our other algorithms.

Despite achieving above chance rates overall, there is still room for improvement as the highest classification rates lie within the range of 71% to 74%. This could be a result of the relatively small sample size, as algorithms that involve the measurement of entropy require a high number of samples for each label in order to select the attributes with the information gain accurately. Furthermore decision tree algorithms are relatively simple compared to similar classifiers, so a higher accuracy might be achieved with a more complex predictor.

In the upcoming assignment, a neural network will be implemented which can be compared on performance level to the obtained decision trees. We expect the multi-class prediction part to be much simpler as neural networks output probabilities and support multi-class classification by nature.

References

[1] Andrew Ng, *Machine Learning*, Coursera. `http://www.coursera.org/course/ml` (retrieved October 27, 2014)

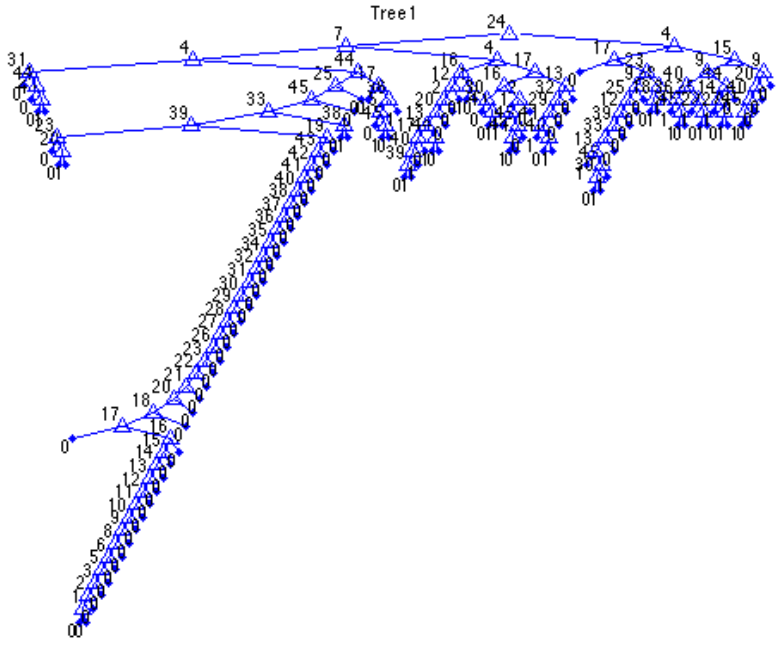
[2] Maja Pantic, *Machine Learning (Course 395) Computer Based Coursework Manual*, Imperial College London, 2014. `http://ibug.doc.ic.ac.uk/courses/machine-learning-course-395/` (retrieved October 27, 2014)

[3] MATLAB R2014b Documentation, *classregtree.test*, MathWorks. `http://www.mathworks.co.uk/help/stats/classregtree.test.html` (retrieved October 27, 2014)

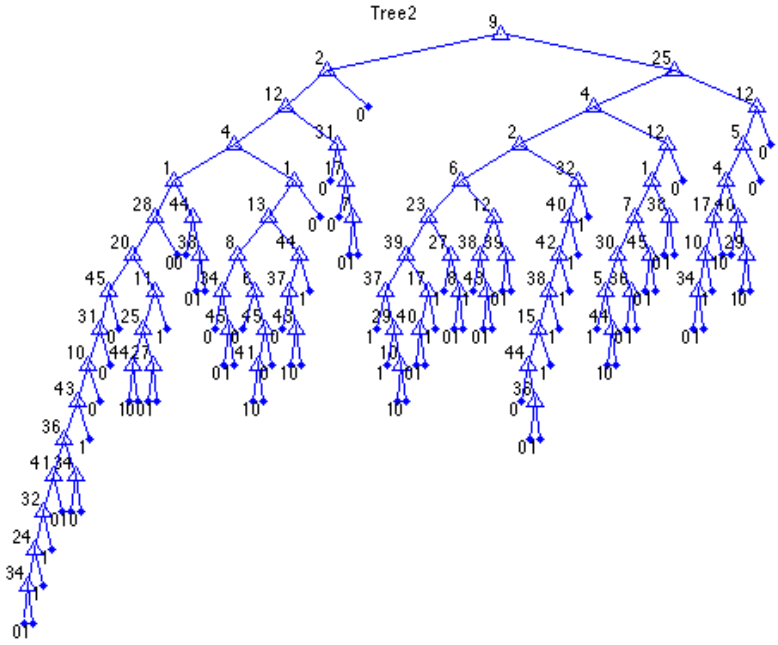
[4] Tom Mitchell, *Machine Learning*, McGraw-Hill Science/Engineering/Math, 1997, ISBN 0070428077.

Appendices

A Learned decision trees

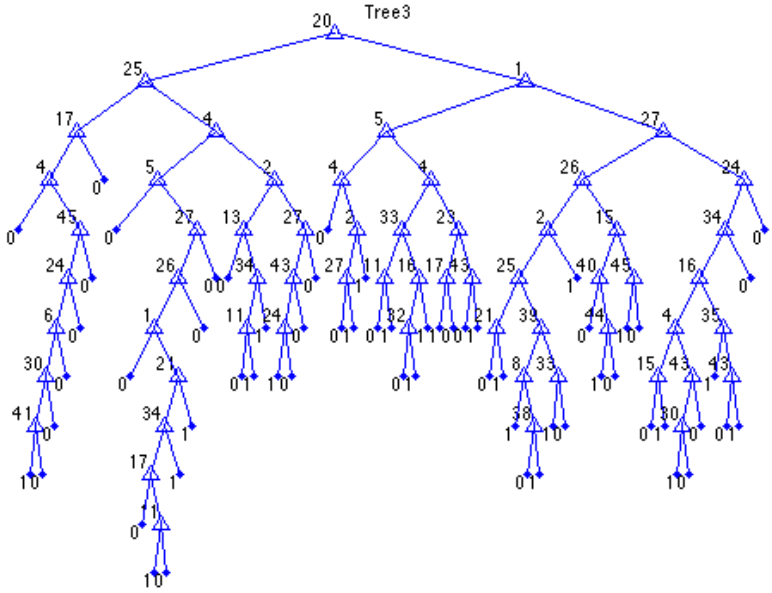


(a) $y = 1$

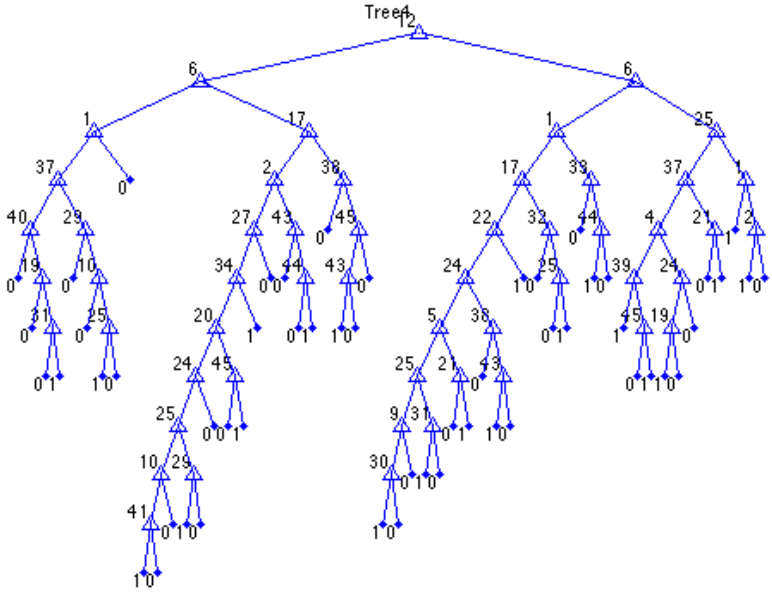


(b) $y = 2$

Figure 1: Decision trees for $y \in \{1, 2\}$ on clean dataset

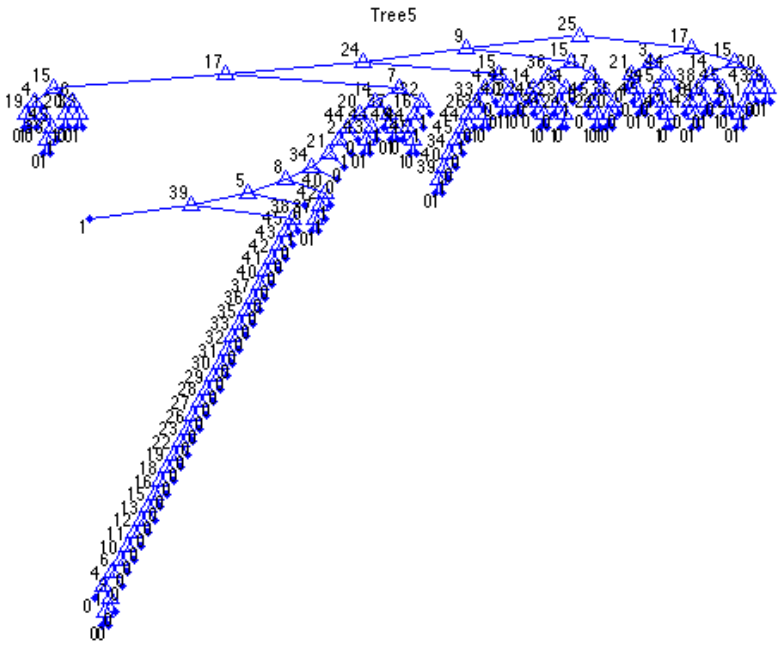


(a) $y = 3$

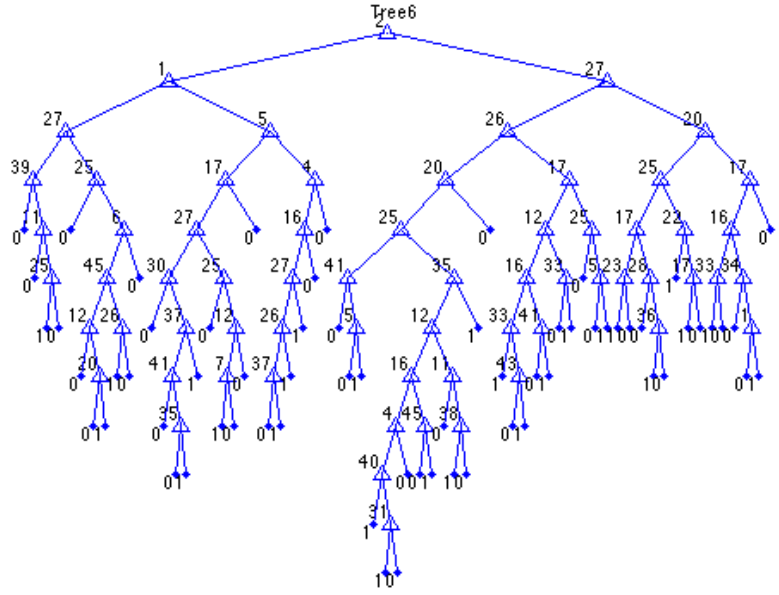


(b) $y = 4$

Figure 2: Decision trees for $y \in \{3, 4\}$ on clean dataset

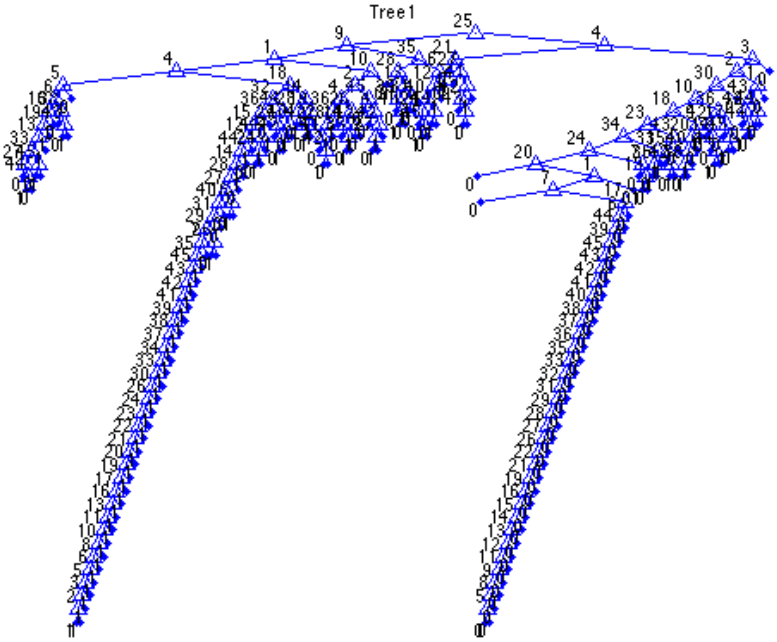


(a) $y = 5$

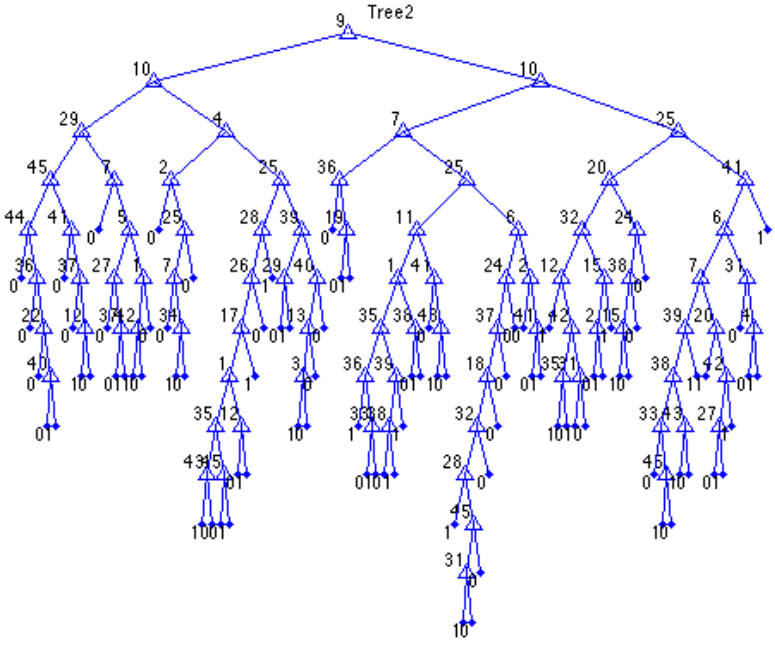


(b) $y = 6$

Figure 3: Decision trees for $y \in \{5, 6\}$ on clean dataset

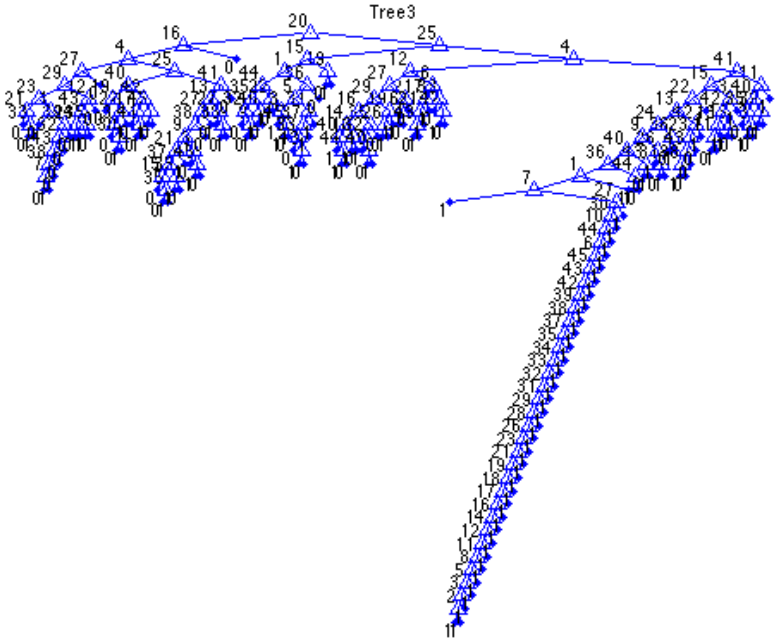


(a) $y = 1$

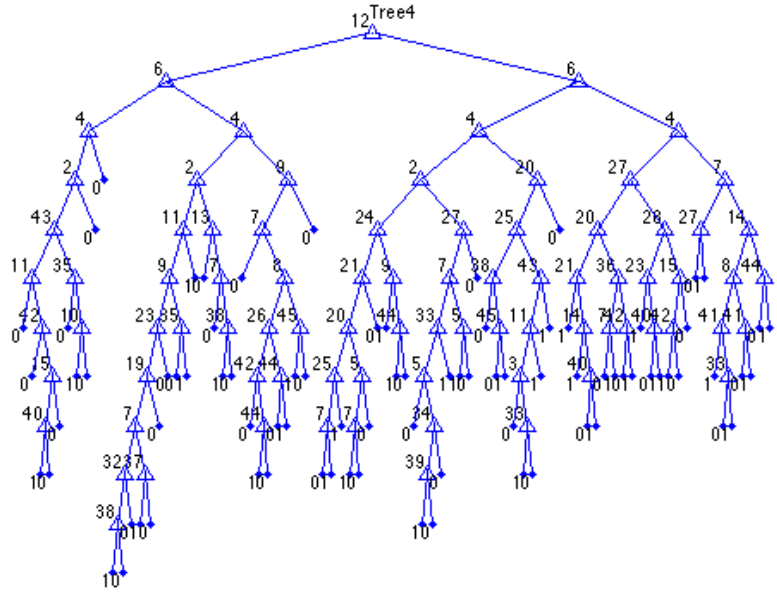


(b) $y = 2$

Figure 4: Decision trees for $y \in \{1, 2\}$ on noisy dataset

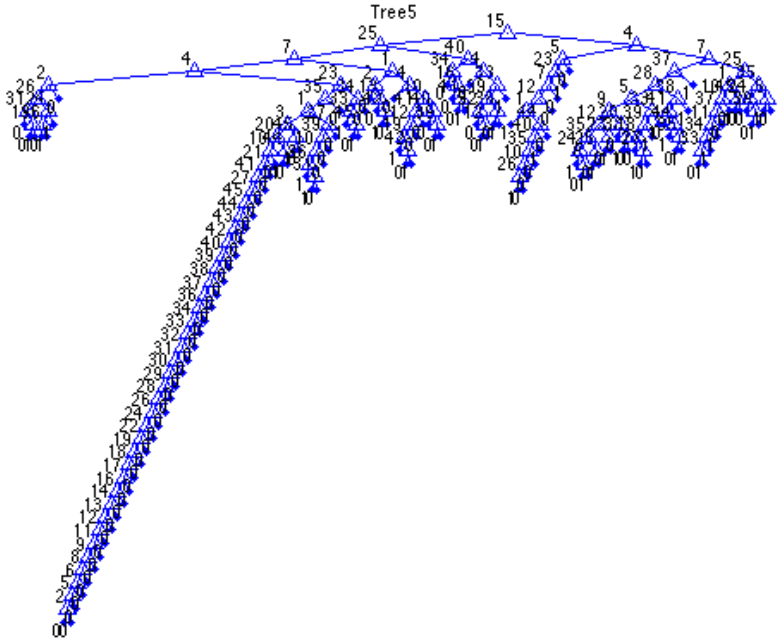


(a) $y = 3$

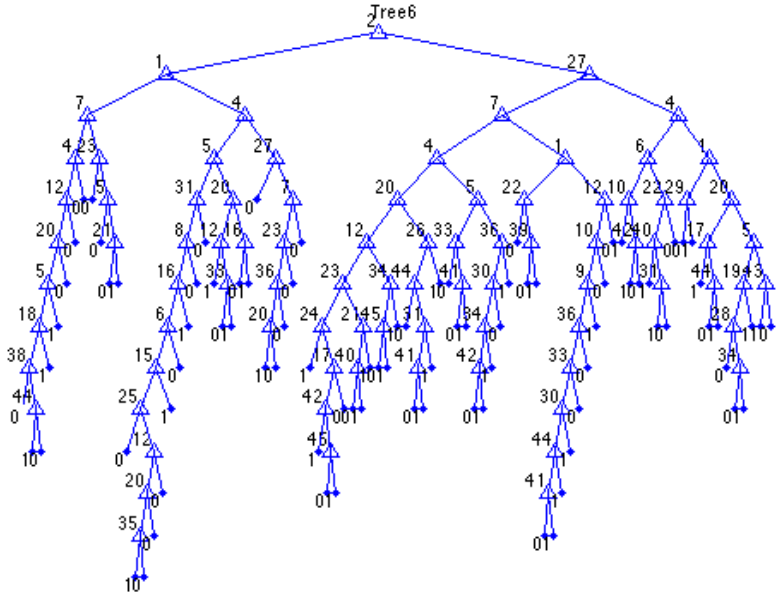


(b) $y = 4$

Figure 5: Decision trees for $y \in \{3, 4\}$ on noisy dataset



(a) $y = 5$



(b) $y = 6$

Figure 6: Decision trees for $y \in \{5, 6\}$ on noisy dataset

B Performance measures

		Prediction					
		1	2	3	4	5	6
Actual class	1	11	1	1	2	3	1
	2	2	16	1	2	0	0
	3	1	2	11	0	0	0
	4	0	0	0	14	0	1
	5	1	1	0	1	6	1
	6	0	1	3	0	1	16

	Class					
	1	2	3	4	5	6
Precision	0.73	0.76	0.69	0.74	0.60	0.84
Recall	0.58	0.76	0.79	0.93	0.60	0.76
F_1	0.65	0.76	0.73	0.82	0.60	0.80

Average classification rate: 0.74

Table 1: Performance measures for **clean dataset** of predictor 1

		Prediction					
		1	2	3	4	5	6
Actual class	1	11	1	1	2	3	1
	2	2	18	0	1	0	0
	3	1	2	11	0	0	0
	4	0	0	1	14	0	0
	5	1	1	0	1	7	0
	6	0	1	3	0	1	16

	Class					
	1	2	3	4	5	6
Precision	0.73	0.78	0.69	0.78	0.64	0.94
Recall	0.58	0.86	0.79	0.93	0.70	0.76
F_1	0.65	0.82	0.73	0.85	0.67	0.84

Average classification rate: 0.77

Table 2: Performance measures for **clean dataset** of predictor 2

		Prediction					
		1	2	3	4	5	6
Actual class	1	11	1	2	1	4	0
	2	1	19	1	0	0	0
	3	1	2	11	0	0	0
	4	0	1	1	12	0	1
	5	1	1	0	1	7	0
	6	0	1	4	0	1	15

	Class					
	1	2	3	4	5	6
Precision	0.79	0.76	0.58	0.86	0.59	0.94
Recall	0.58	0.90	0.79	0.80	0.70	0.71
F_1	0.67	0.83	0.67	0.83	0.64	0.81

Average classification rate: 0.75

Table 3: Performance measures for **clean dataset** of predictor 3

		Prediction					
		1	2	3	4	5	6
Actual class	1	2	1	1	0	1	0
	2	1	16	1	0	1	0
	3	1	3	16	2	0	1
	4	0	2	1	20	0	0
	5	1	2	0	0	4	2
	6	1	2	1	1	3	13

	Class					
	1	2	3	4	5	6
Precision	0.33	0.62	0.80	0.87	0.44	0.81
Recall	0.40	0.84	0.70	0.87	0.44	0.62
F_1	0.36	0.71	0.74	0.87	0.44	0.70

Average classification rate: 0.71

Table 4: Performance measures for **noisy dataset** of predictor 1

		Prediction					
		1	2	3	4	5	6
Actual class	1	3	2	1	0	1	0
	2	1	11	1	0	0	1
	3	3	2	15	0	0	3
	4	1	2	3	15	2	1
	5	4	3	0	0	2	1
	6	1	0	1	1	0	19

	Class					
	1	2	3	4	5	6
Precision	0.23	0.55	0.71	0.94	0.40	0.76
Recall	0.43	0.79	0.65	0.63	0.20	0.86
F_1	0.30	0.65	0.68	0.75	0.27	0.81

Average classification rate: 0.65

Table 5: Performance measures for **noisy dataset** of predictor 2

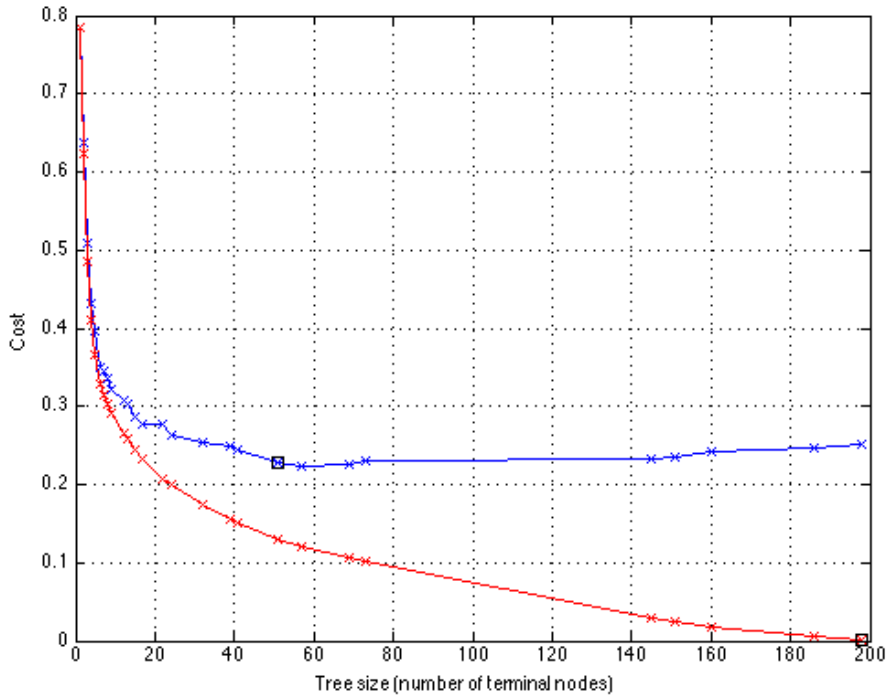
		Prediction					
		1	2	3	4	5	6
Actual class	1	4	0	1	0	0	0
	2	2	13	2	1	1	0
	3	1	3	17	1	0	1
	4	0	2	3	17	0	1
	5	2	1	0	0	6	0
	6	1	2	1	0	4	13

	Class					
	1	2	3	4	5	6
Precision	0.40	0.62	0.71	0.89	0.55	0.87
Recall	0.80	0.68	0.74	0.74	0.67	0.62
F_1	0.53	0.65	0.72	0.81	0.60	0.72

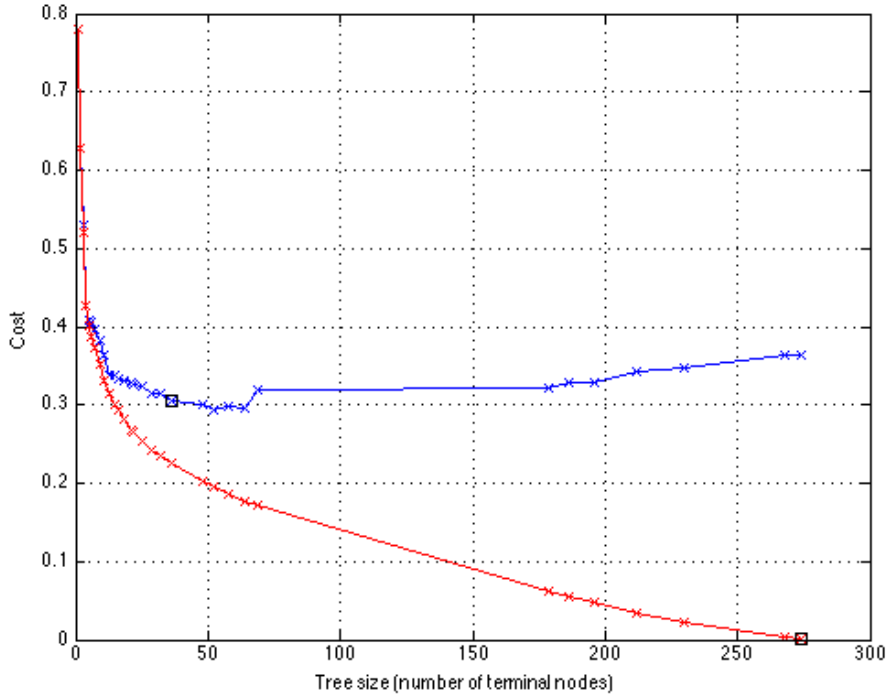
Average classification rate: 0.70

Table 6: Performance measures for **noisy dataset** of predictor 3

C Pruning learning curves



(a) Clean dataset



(b) Noisy dataset

Figure 7: Pruning learning curves for both datasets