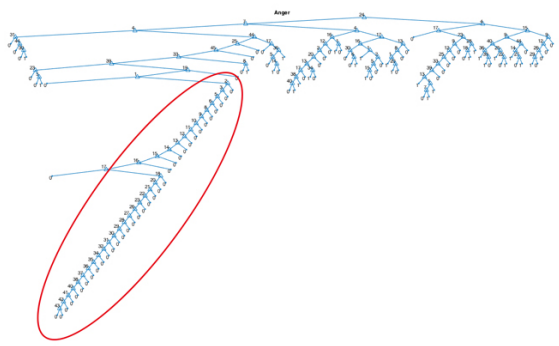
CO395 Machine Learning
CBC Assignment #2
Decision Tree Learning

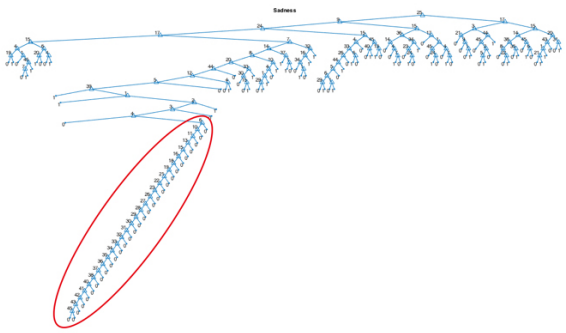
Group 11
Baisheng Song(bs2111) Jiayun Ding(jd1611)
Yiming Lin(yl8411) Yan Liu(ybl11)

November 2, 2014

1 About Clean Dataset

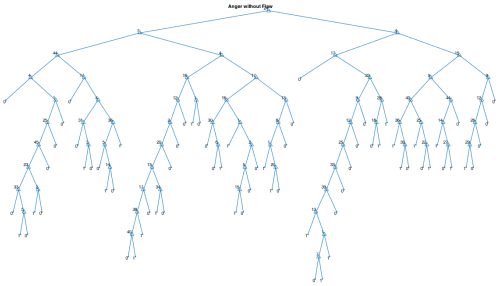
The clean dataset provided in datafiles is not absolutely clean. There are some flaws in diagrams, particularly in diagrams for Emotion 1 and Emotion 5, as shown on figures below: The progrom explored all 45 attributes in these two diagrams, and these two longest leaves are both classified as false(0).

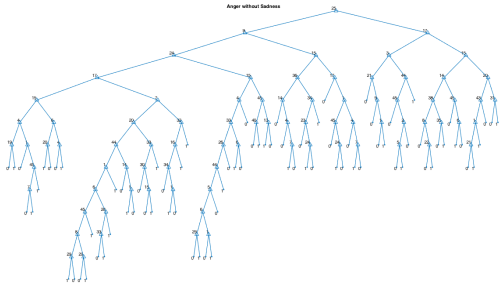




Further investigation was carried out by us and eventually we figured out there are two example rows(458,679) with exactly the same attributes but different targets in the clean dataset, in other words the dataset is not absolutely clean. This small flaw resulted in long leaves on the trees.

We deleted these two examples as wells as the corresponding targets. We trained new tree sets. Emotion 2,3,4,6 remained the same, while Emotion 1,5 have been changed. As shown in the following diagrams.





For the consistency of the coursework, the remaining content of the report will be based on the unchanged clean dataset as provided.

We have provided both trained tree sets generated from "absolute clean dataset" and "provided clean dataset" in our codes. Please perform unseen data on both tree sets.

2 Implementation

For the examples' testing, the codes written in the function `n_fold_validation()` follows several steps. Firstly, in `split_dataset()` function, we separate the example matrix into a test matrix and a training matrix with the portion 1:9 in the column, similarly for the corresponding target vector. In the case of 10 fold validation, this procedure needs to be done 10 times with the testing matrix set shifting from the top N/10 part to the bottom N/10 in the whole example set, and for each time the rest of the example set act as the training set.

The next step is to generate all the decision trees, six trees for six emotions in each turn of the validation process. The six trees' root node are stored in a 6*1 vector. To form those trees, we wrote the function `decision_tree_learning()` by following the steps described in part two in the manual, which takes an 9/10*N*45 training matrix, a 1*45 attribute matrix and an 9/10*N*1 binary vector as the inputs and outputs a decision tree for testing. The attribute matrix is created to be a set of continuous integers counting from 1 to 45. Based on the corresponding number of the emotions, the provided target vector can be split into six separate binary target vectors by using the equation `map_emotion()`. The decision tree is defined by a series of nodes, which are produced by the factory function `make_node`. The structure of the nodes is defined by the coursework specifications. In terms of forming a decision tree, each root node is added according to the chosen best attribute.

The function `choose_best_decision_attribute()` calculates and compares the information gain for all the remaining attributes. By means of the function $Gain(attribute) = Entropy(p, n) - Remainder(attribute)$, the entropy for the training data is calculated first by counting 0s and 1s in the binary target vector and using the entropy equation. Then, for each remaining attribute, the numbers of p0, p1, n0 and n1 are counted by comparing 0s and 1s in the example matrix and the binary target vector. Using those counted results, the $Remainder(attribute)$ can be calculated and hence the gain for each remaining attribute. The gain is stored into a 1*M matrix (M<=45), and the row index with the maximum gain can then be found. The corresponding best attribute can then be matched with this row index.

When an attribute has been used, it is deleted in the attribute vector. Therefore, the size of the attribute matrix is reduced. Based on the chosen best attribute, the example matrix and the binary target vector are both separated into two matrices with smaller size according to the 0 and 1 element in the example matrix. Those smaller sized examples, binary targets and attribute matrix are then recurred to form sub-trees with the root node stored in *tree.kids*. When the reduced-size binary target vector only has a unique value 0 or 1, the decision tree is completed. Otherwise, the tree branches until either the attribute matrix or the example matrix has been reduced to be empty. In this case, `majority_value()` function is used to make the mode of the element in the reduced-size binary target vector as the class of the leaf node. This function simply uses the default function `mode()` in Matlab.

Next step is using `testTrees()`, it test examples on the given trees. This function passes all trees, examples and a given strategy number to `evaluate()`. In the evaluation, the boolean classification and the depth are created for each example with each tree using `evaluate_boolean()`. Then evaluate function maps boolean classification into real emotion using function `use_strategy()`. If only one tree is classified, it will go into predicted result; if more than one tree is classified, it will use a strategy to choose between positive classifications. This will be discussed in the ambiguity question(5.2).

In the final step of `n_fold_validation()` function, we include the `calculate_confusion_matrix()` function which takes the testing target vector as the *actual_results* and the *predicted_results* vector as the prediction. This function forms a 6 by 6 confusion matrix with actual results being the column and predicted results being the row. Then, it maps the actual and predicted results to the corresponding position in the 6*6 matrix, and the value in this position is increased by one. After mapping all the columns in the target vector and the *predicted_results* vector, the confusion matrix for this particular testing set is ready.

For every group of testing set and training set, the above four steps are carried out. Therefore, ten sets of *predicted_results* and confusion matrix are generated in total for a ten-fold-validation process. Both sets of results are returned in a 1*10 cell array.

For the average result calculations, we have several functions. `combine_confusion_matrix()` is used to combine multiple matrices into one matrix. `recall_precision_rate()` takes predicted results and actual results as input, uses `calculate_confusion_matrix()` to create a confusion matrix and returns the calculated recall and precision on each class as a matrix according to the given formula. `f_measure()` takes an alpha variable(in this case is 1) and *recall_precision_matrix* as inputs and outputs f_measure matrix using the formula. `classification_rate()` is simply two vectors and calculating (sum of same values/total).

3 Tree Diagrams

3.1 Emotion 1 - Anger

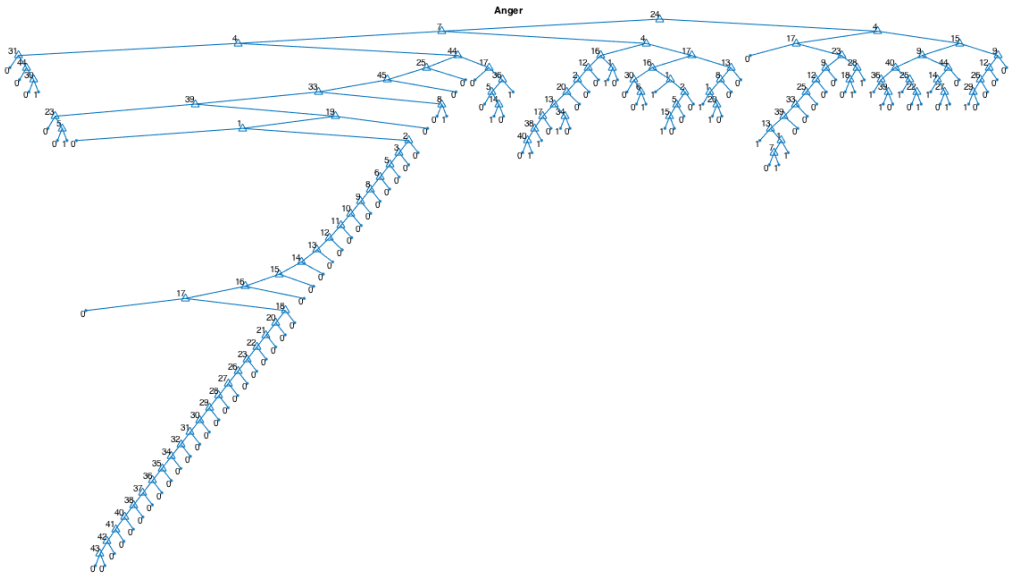


Figure 1: Trained decsion tree on the clean dataset for Emotion 1 - Anger

3.2 Emotion 2 - Disgust

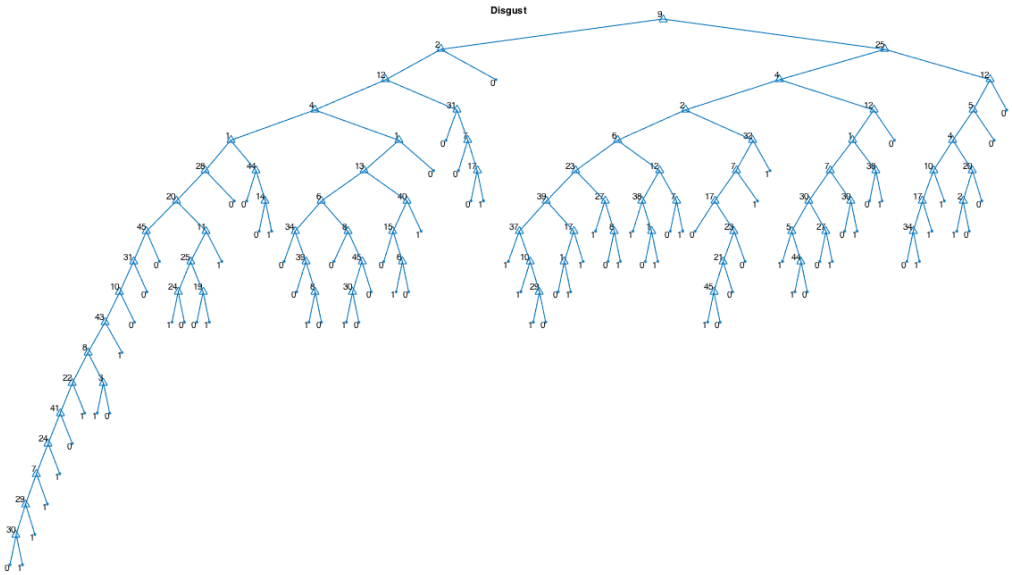


Figure 2: Trained decsion tree on the clean dataset for Emotion 2 - Disgust

3.3 Emotion 3 - Fear

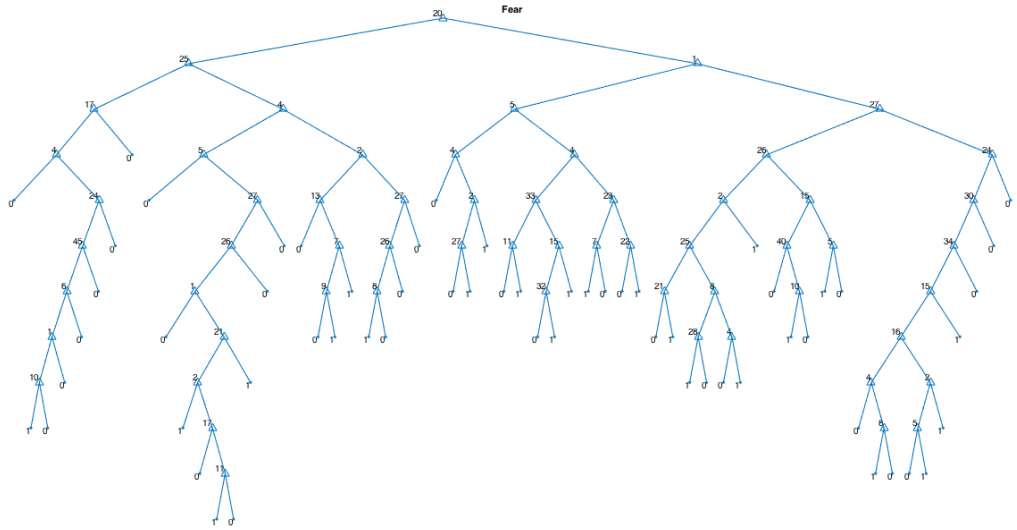


Figure 3: Trained decsion tree on the clean dataset for Emotion 3 - Fear

3.4 Emotion 4 - Happiness

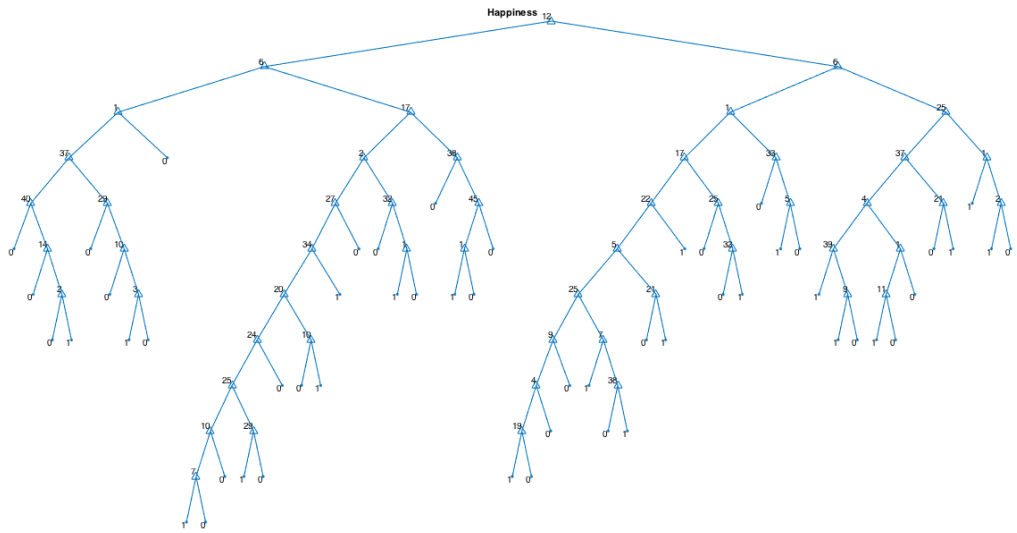


Figure 4: Trained decsion tree on the clean dataset for Emotion 4 - Happiness

3.5 Emotion 5 - Sadness

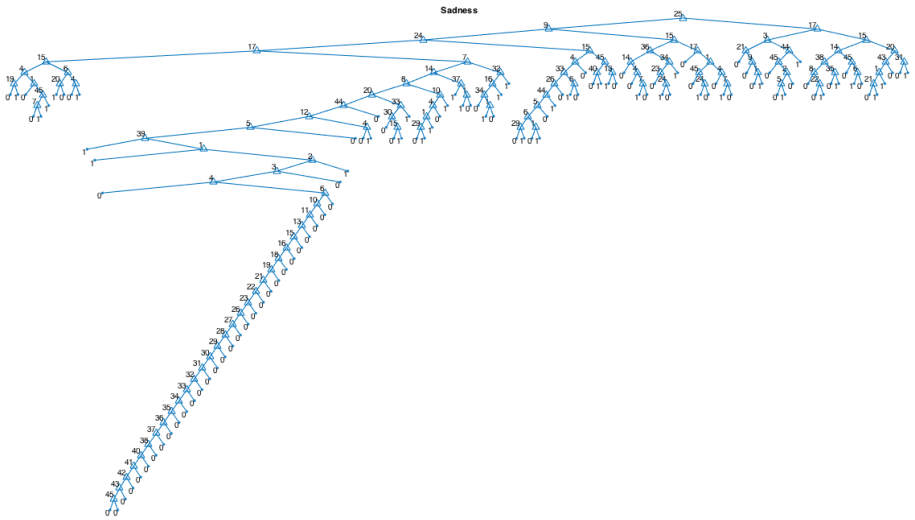


Figure 5: Trained decsion tree on the clean dataset for Emotion 5 - Sadness

3.6 Emotion 6 - Surprise

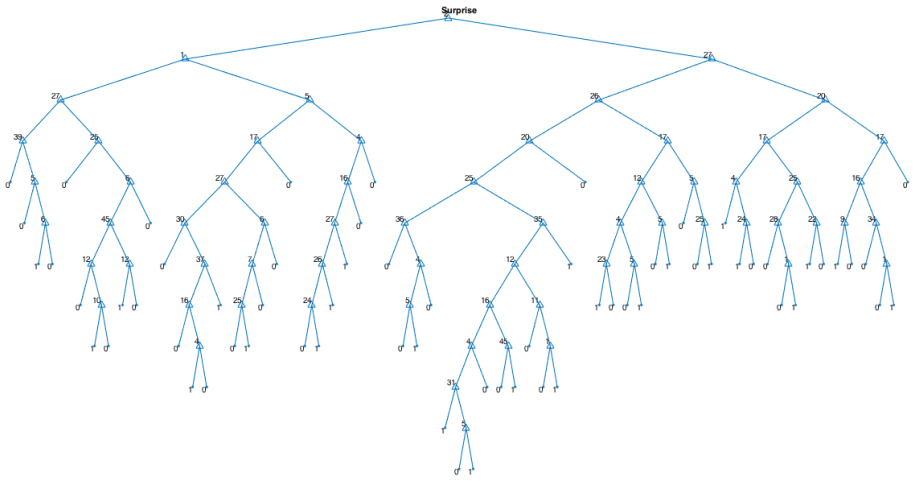


Figure 6: Trained decsion tree on the clean dataset for Emotion 6 - Surprise

4 Evaluation

The results below are for the maximum depth strategy. Evaluations of other strategies(5 in total) are presented in the next section on the question about ambiguity.

4.1 Clean Dataset

4.1.1 Confusion Matrix

		Predicted Class					
		1	2	3	4	5	6
Actual Class	1	**78**	30	3	5	15	1
	2	11	**165**	1	9	8	4
	3	14	8	**81**	3	4	9
	4	3	9	3	**180**	13	8
	5	21	21	4	7	**76**	3
	6	5	9	15	3	11	**164**

Table 1: Confusion Matrix for the *Clean* Dataset (Strategy 3 - Maximum Depth)

4.1.2 Recall, Precision and F_1 Measure

		Recall	Precision	F_1
Actual class	1	59%	59%	59%
	2	83%	68%	75%
	3	68%	76%	72%
	4	83%	87%	85%
	5	58%	60%	59%
	6	79%	87%	83%

Table 2: Recall, Precision and F_1 Measure for the *Clean* Dataset (Strategy 3 - Maximum Depth)

4.1.3 Average Classification Rate

$$CR = \frac{744}{1004} = 74.1\%$$

Figure 7: Classification Rate for the *Clean* Dataset (Strategy 3 - Maximum Depth)

4.2 Noisy Dataset

4.2.1 Confusion Matrix

		Predicted Class					
		1	2	3	4	5	6
Actual Class	1	**34**	10	18	2	16	8
	2	22	**122**	18	13	8	4
	3	24	12	**106**	16	10	19
	4	15	18	17	**144**	5	10
	5	27	5	12	3	**55**	8
	6	11	7	38	7	14	**143**

Table 3: Confusion Matrix for the *Noisy* Dataset (Strategy 3 - Maximum Depth)

4.2.2 Recall, Precision and F_1 Measure

		Recall	Precision	F_1
Actual class	1	39%	26%	31%
	2	65%	70%	68%
	3	57%	51%	54%
	4	69%	78%	73%
	5	50%	51%	50%
	6	65%	75%	69%

Table 4: Recall, Precision and F_1 Measure for the *Noisy* Dataset (Strategy 3 - Maximum Depth)

4.2.3 Average Classification Rate

$$CR = \frac{604}{1004} = 60.2\%$$

Figure 8: Classification Rate for the *Noisy* Dataset (Strategy 3 - Maximum Depth)

4.3 Discussion of Result

The cross-validation revealed that some emotions are classified much better than the other ones.In the clean dataset, emotion 4 and 6 achieved 85% and 83% F1 measures respectively. Emotion 2 and 3 are also pretty good; the F1 measures for them are 75% and 72% respectively. However, their recall and precision rates are closer to 60% to 70% instead of 80%. In the case of the noisy dataset, there is a particularly bad classification performance for emotion 1 which is anger. The recall and precision rates are 39% and 26% respectively. These values dropped from the ones obtained in the clean dataset for all other emotions as well, and however they are still acceptable.

5 Questions

5.1 Noisy-Clean Datasets

Yes. There is some difference in the performance when using the clean and noisy datasets.

As expected, the trees predict better on the clean dataset than the noisy dataset. The Classification Rate for clean dataset is 74.1%, the Error Rate for clean dataset is 25.9%; the Classification Rate for noisy dataset is 60.2%, the Error Rate for noisy dataset is 39.8%. In the case of noisy dataset, the trees have been trained on data that itself contains classification errors. We cannot expect trees trained on an unreliable dataset to perform better than the trees trained on a reliable and infallible dataset.

Some trees perform much better than others in both datasets, especially emotion 2, 4 and 6. However, some emotions like 1, 3 and 5 have a fall back in noisy dataset. The reason of this may be there are more noises for these emotions than others, and these trees are incorrectly trained on a large deviation.

5.2 Ambiguity

We tried five strategies of determining targets to examples that have been positively classified by more than one tree:

1. Choose a random class from positive predicted classes.
2. Choose the class with minimum explored depth.
3. Choose the class with maximum explored depth.
4. Choose the class with maximum explored depth; if more than one classes have the same explored depth, choose the one with ess nodes(minimum size).
5. Choose the class with maximum explored depth; if more than one classes have the same depth, choose the one with more nodes(maximum size)l.

Advantage for strategy 1 - randomized choice is the simplest, easiest and most obvious strategy to implement. It provides a good benchmark(pivot) for comparing methods. Disadvantage: The performance is not the worst, but certainly not the best among all strategies.

Advantage for strategy 2 is that the smaller number of depth a tree needs to explore in order to classify a target, the more certain it should be that the emotion is correctly classified. However, the disadvantage is that fewer nodes may lead to coincidence, The performance proved the disadvantage of this strategy.

The strategy 3 - Maximum depth strategy is similar to the Minimum depth strategy, but we choose the class with the maximum depth instead. The advantage for this strategy is that probability of coincidence would be small. Both strategy 2 and 3 require storing an extra N*6 matrix for depth, this increases the memory requirement. The performance proved it is a better strategy than the previous one.

The intuition behind strategy 4 is what if there are still classes with the same depth, we choose sizes(number of nodes) of each tree as a parameter. In strategy 4, when an example is positively classified to two trees that have the same depth, the tree with a minimum size will be chosen.Advantage: The larger occupation of depth in total size (depth/size), the more certain that the example belongs to this tree. Disadvantage: the large number of depth/size is due to the small size of the tree, therefore, the probability of coincidence increases.

This strategy proved to be wrong by the actual performance, but the opposite strategy 5 has a better performance especially in noisy dataset, because larger size of a tree means the probability of incidentally exploring the path is relatively low, which is an advantage. We do not see the disadvantage for this strategy until the pruning exmaple, and the actual performance also indicates this is the best strategy among all 5 strategies. Both strategy 4 and 5 require storing an extra N*6 matrix to store depth. These matrices increase the memory requirement in comparison to strategy 3.

Confusion Matrixs, Recalls, Precisions, F1 Measures and Classification Rates for both clean and noisy datasets on all five strategies are shown below. Comparison among Strategy 1,2,3 and comparison among Strategy 3,4,5 will be discussed.

Evaluation of Strategy 1 - Randomized Choice on Clean Dataset

		Predicted Class					
		1	2	3	4	5	6
Actual Class	1	**81**	15	5	8	16	7
	2	15	**137**	8	13	16	9
	3	6	8	**81**	0	8	16
	4	3	12	5	**180**	10	6
	5	16	15	8	8	**77**	8
	6	2	5	14	9	13	**164**

Table 5: Confusion Matrix for the *Clean* Dataset (Strategy 1)

		Recall	Precision	F_1
Actual class	1	61%	66%	64%
	2	69%	71%	70%
	3	68%	67%	68%
	4	83%	82%	83%
	5	58%	55%	57%
	6	79%	78%	79%

Table 6: Recall, Precision and F_1 Measure for the *Clean* Dataset (Strategy 1)

$$CR = \frac{720}{1004} = 71.7\%$$

Figure 9: Classification Rate for the *Clean* Dataset (Strategy 1)

Evaluation of Strategy 1 - Randomized Choice on Noisy Dataset

		Predicted Class					
		1	2	3	4	5	6
Actual Class	1	**26**	12	15	11	17	7
	2	13	**124**	18	16	9	7
	3	15	17	**103**	22	11	18
	4	9	12	18	**153**	6	11
	5	21	8	9	6	**53**	13
	6	10	10	19	12	16	**153**

Table 7: Confusion Matrix for the *Noisy* Dataset (Strategy 1)

		Recall	Precision	F_1
Actual class	1	29%	28%	29%
	2	66%	68%	67%
	3	56%	59%	56%
	4	73%	70%	71%
	5	48%	47%	48%
	6	70%	73%	71%

Table 8: Recall, Precision and F_1 Measure for the *Noisy* Dataset (Strategy 1)

$$CR = \frac{612}{1001} = 61.1\%$$

Figure 10: Classification Rate for the *Noisy* Dataset (Strategy 1)

Evaluation of Strategy 2 - Minimum Depth on Clean Dataset

		Predicted Class					
		1	2	3	4	5	6
Actual Class	1	**70**	12	23	11	8	8
	2	29	**122**	30	6	4	7
	3	6	18	**70**	8	8	9
	4	13	19	5	**168**	6	5
	5	14	16	16	19	**63**	4
	6	4	31	8	8	2	**154**

Table 9: Confusion Matrix for the *Clean* Dataset (Strategy 2)

		Recall	Precision	F_1
Actual class	1	53%	51%	52%
	2	62%	56%	59%
	3	59%	46%	52%
	4	78%	76%	77%
	5	48%	69%	57%
	6	74%	82%	78%

Table 10: Recall, Precision and F_1 Measure for the *Clean* Dataset (Strategy 2)

$$CR = \frac{647}{1004} = 64.4\%$$

Figure 11: Classification Rate for the *Clean* Dataset (Strategy 2)

Evaluation of Strategy 2 - Minimum Depth on Noisy Dataset

		Predicted Class					
		1	2	3	4	5	6
Actual Class	1	**21**	13	20	16	10	8
	2	24	**106**	15	30	5	7
	3	25	15	**84**	35	8	20
	4	28	13	12	**138**	5	13
	5	14	11	9	21	**44**	11
	6	40	11	14	14	8	**133**

Table 11: Confusion Matrix for the *Noisy* Dataset (Strategy 2)

		Recall	Precision	F_1
Actual class	1	24%	13%	18%
	2	57%	63%	60%
	3	45%	55%	49%
	4	66%	54%	60%
	5	40%	55%	46%
	6	60%	69%	65%

Table 12: Recall, Precision and F_1 Measure for the *Noisy* Dataset (Strategy 2)

$$CR = \frac{526}{1001} = 52.5\%$$

Figure 12: Classification Rate for the *Noisy* Dataset (Strategy 2)

Evaluation of Strategy 3 - Maximum Depth on Clean Dataset

		Predicted Class					
		1	2	3	4	5	6
Actual Class	1	**78**	30	3	5	15	1
	2	11	**165**	1	9	8	4
	3	14	8	**81**	3	4	9
	4	3	9	3	**180**	13	8
	5	21	21	4	7	**76**	3
	6	5	9	15	3	11	**164**

Table 13: Confusion Matrix for the *Clean* Dataset (Strategy 3)

		Recall	Precision	F_1
Actual class	1	59%	59%	59%
	2	83%	68%	75%
	3	68%	76%	72%
	4	83%	87%	85%
	5	58%	60%	59%
	6	79%	87%	83%

Table 14: Recall, Precision and F_1 Measure for the *Clean* Dataset (Strategy 3)

$$CR = \frac{744}{1004} = 74.1\%$$

Figure 13: Classification Rate for the *Clean* Dataset (Strategy 3)

Evaluation of Strategy 3 -Maximum Depth on Noisy Dataset

		Predicted Class					
		1	2	3	4	5	6
Actual Class	1	**34**	10	18	2	16	8
	2	22	**122**	18	13	8	4
	3	24	12	**106**	16	10	19
	4	15	18	17	**144**	5	10
	5	27	5	12	3	**55**	8
	6	11	7	38	7	14	**143**

Table 15: Confusion Matrix for the *Noisy* Dataset (Strategy 3)

		Recall	Precision	F_1
Actual class	1	39%	26%	31%
	2	65%	70%	68%
	3	57%	51%	54%
	4	69%	78%	73%
	5	50%	51%	50%
	6	65%	75%	69%

Table 16: Recall, Precision and F_1 Measure for the *Noisy* Dataset (Strategy 3)

$$CR = \frac{604}{1001} = 60.3\%$$

Figure 14: Classification Rate for the *Noisy* Dataset (Strategy 3)

Evaluation of Strategy 4 - Maximum Depth with Minimum Size on Clean Dataset

		Predicted Class					
		1	2	3	4	5	6
Actual Class	1	**73**	31	6	5	14	3
	2	10	**162**	4	10	8	4
	3	11	10	**82**	4	4	10
	4	2	8	3	**182**	13	8
	5	17	20	7	8	**76**	4
	6	2	8	17	3	11	**166**

Table 17: Confusion Matrix for the *Clean* Dataset (Strategy 4)

		Recall	Precision	F_1
Actual class	1	55%	63%	59%
	2	81%	68%	74%
	3	67%	68%	68%
	4	84%	86%	85%
	5	58%	60%	59%
	6	80%	85%	83%

Table 18: Recall, Precision and F_1 Measure for the *Clean* Dataset (Strategy 4)

$$CR = \frac{739}{1004} = 73.6\%$$

Figure 15: Classification Rate for the *Clean* Dataset (Strategy 4)

Evaluation of Strategy 4 - Maximum Depth with Minimum Size on Noisy Dataset

		Predicted Class					
		1	2	3	4	5	6
Actual Class	1	**32**	10	15	5	18	8
	2	14	**122**	16	17	10	8
	3	20	13	**102**	19	12	21
	4	11	15	14	**150**	5	14
	5	22	6	9	6	**57**	10
	6	10	8	31	9	13	**149**

Table 19: Confusion Matrix for the *Noisy* Dataset (Strategy 4)

		Recall	Precision	F_1
Actual class	1	36%	29%	32%
	2	65%	70%	68%
	3	55%	55%	55%
	4	72%	73%	73%
	5	52%	50%	51%
	6	68%	71%	69%

Table 20: Recall, Precision and F_1 Measure for the *Noisy* Dataset (Strategy 4)

$$CR = \frac{612}{1001} = 61.1\%$$

Figure 16: Classification Rate for the *Noisy* Dataset (Strategy 4)

Evaluation of Strategy 5 - Maximum Depth with Maximum Size on Clean Dataset

		Predicted Class					
		1	2	3	4	5	6
Actual Class	1	**75**	27	3	5	21	1
	2	10	**163**	1	9	13	4
	3	12	7	**78**	3	8	11
	4	3	9	2	**177**	16	9
	5	15	20	2	6	**86**	3
	6	3	9	11	3	13	**168**

Table 21: Confusion Matrix for the *Clean* Dataset (Strategy 5)

		Recall	Precision	F_1
Actual class	1	57%	64%	60%
	2	82%	69%	75%
	3	66%	80%	72%
	4	82%	87%	85%
	5	65%	55%	60%
	6	81%	86%	83%

Table 22: Recall, Precision and F_1 Measure for the *Clean* Dataset (Strategy 5)

$$CR = \frac{747}{1004} = 74.4\%$$

Figure 17: Classification Rate for the *Clean* Dataset (Strategy 5)

Evaluation of Strategy 5 - Maximum Depth with Maximum Size on Noisy Dataset

		Predicted Class					
		1	2	3	4	5	6
Actual Class	1	**34**	7	20	2	16	9
	2	22	**115**	23	13	9	5
	3	24	7	**109**	16	11	20
	4	15	9	22	**147**	6	10
	5	26	5	13	3	**55**	8
	6	11	7	38	4	14	**146**

Table 23: Confusion Matrix for the *Noisy* Dataset (Strategy 5)

		Recall	Precision	F_1
Actual class	1	39%	26%	31%
	2	62%	77%	68%
	3	58%	48%	53%
	4	70%	80%	75%
	5	50%	50%	50%
	6	66%	74%	70%

Table 24: Recall, Precision and F_1 Measure for the *Noisy* Dataset (Strategy 5)

$$CR = \frac{688}{1001} = 68.7\%$$

Figure 18: Classification Rate for the *Noisy* Dataset (Strategy 5)

Comparison of Strategies

	Classification Rate *Clean* Dataset	Classification Rate *Noisy* Dataset
Strategy 3 - Maximum Depth	74.1%	60.3%
Strategy 1 - Randomized	71.7%	61.1%
Strategy 2 - Minimum Depth	64.4%	61.1%

Table 25: Aggregated Classification Rate for Randomized Strategy as Pivot

Table 25 presents classification rates for randomized choice, maximum depth and minimum depth. Firstly, we expected strategy 2 to work reasonably well but has performed worst among these three strategies. However, the opposite strategy 3 performed best in the case of the clean dataset and very closely to the strategy 1 in the noisy dataset. Since the classification rates for strategy 3 and 1 in the noisy dataset are very close and, for strategy 1, it includes some random assignment, so it might just got lucky and assigned the correct example in most cases where there was a conflict. If we ran the simulation several times more, strategy 1 might have been performing a bit worse. Taking all of this into consideration, strategy 3 is the best one among these three strategies.

	Classification Rate *Clean* Dataset	Classification Rate *Noisy* Dataset
Strategy 5 - Max Depth with Max Nodes	**74.4%**	**68.7%**
Strategy 3 - Maximum Depth	74.1%	60.3%
Strategy 4 - Max Depth with Min Nodes	73.6%	60.1%

Table 26: Aggregated Classification Rate for Maximum Depth as Pivot

Table 26 presents classification rates for maximum depth, maximum depth with minimum nodes and the maximum depth with minimum nodes. In strategy 3, if there are two class still with the same depth, it will choose the first one which is a randomization. So we tried to develop a better algorithm. Again, we expected strategy 4 to work reasonably well but has actually performed worst among these three strategies. However, the opposite strategy 5 performed best in both cases especially in the noisy dataset. However, in the clean dataset, classification rates for these three strategies are very close, maybe some strategies got lucky. So deviation should be taken into consideration, and we are not confident strategy 5 must be better than 3 and 4. We are presuming more nodes means better structured and less chance to get wrong.

After running the pruning examples, whether nodes size should be used in strategies becomes more uncertain. Before the optimal size of the tree, more nodes mean better structure and less cost; however, after the optimal size, more nodes mean overfitting and more cost. So we decided to use strategy 3 as the default strategy.

5.3 Pruning

The `classregtree()` function creates a decision tree for classification. This decision tree is evaluated using the test function with two different methods: cross-validation and resubstitution. The first one uses a 10-fold cross-validation where the input samples are divided into 10 subsamples. The tree is trained on 9 samples, and the 10th sample is used as a validation. The resubstituion method uses the same dataset for both training and testing.

The `test()` function calculates the cost of the tree by summing the estimated probability of each terminal node times the nodes cost itself. The cost of a node can be calculated by misclassification costs of the observations in that node. Together with the cost, the test function also returns the standard error of each cost value, the number of terminal nodes for each subtree, and an estimated best level of pruning.

The `prune()` function takes a tree and returns the same tree with zero-sized subobjects removed.

Finally `plot()` function plots the costs and the corresponding tree sizes on the graph based on the data calculated for both cross validation and resubstitution methods.

Figure 19 is based on the clean dataset. The cost here can be treated as error rate, and tree size is the number of terminal nodes. There are two curves: red and blue. The red and blue curves indicate the cost trend for resubstitution and 10-cross validation respectively as the tree size increases. The red curve continuously decreases until zero as the tree size increases. It is mainly because the resubstitution method uses the same dataset for both training and testing. As long as the tree is large enough that can cover any cases, the error rate is extremely small. However, the blue curve shows some difference compared with the red one. The cost for the blue curve decreases dynamically until the tree size increase to close to 20. Then the curve decreases a bit between tree size 20 and 50; it is because the tree is well trained at this stage. However, after tree size 50, the cost increases slightly. The cause for this increase is that after around size 50, the tree is overfitting, therefore some false examples are classified as true, it results in an increase in error rate.

Figure 20 is based on the noisy dataset. The overall trend is roughly the same as a clean one; the tree is also overfitting after 50 of tree size. Except there is a sharp spike for the blue curve at around 70 tree size. The spike may due to a critical point where many false examples are classified as true, and it suddenly increases the misclassification rate. Moreover, the cost for the noisy dataset at any point is larger than the one for the clean dataset at the same 'tree size' point. Because the classification tree for the noisy dataset is trained based on noisy examples, which means the tree itself contains misclassification paths, so the misclassification rate on other data based on the trained tree is relatively high. Also, the end of two lines for noisy dataset is further than clean dataset, this may because noisy dataset would have more same examples with different targets.

From the observation of these two graphs, we can conclude that the optimal size for our decision tree regarding this test set is about 41 for clean dataset and about 52 for noisy dataset. One more comment, if there is no noise in the clean dataset, the optimal size is about 65.

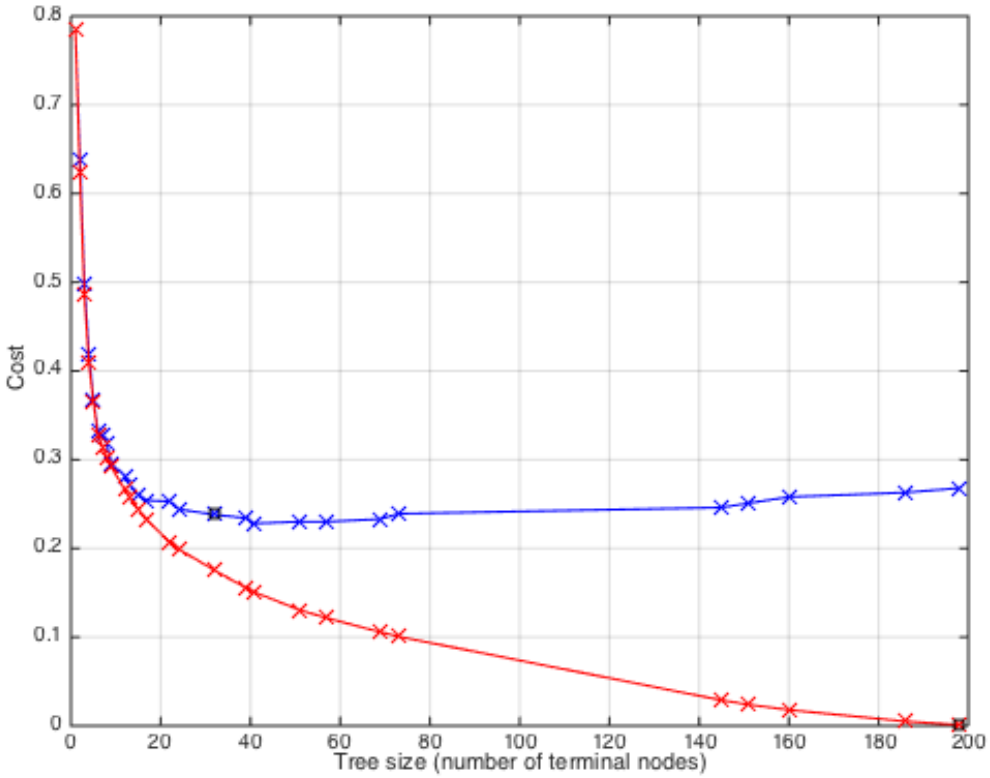


Figure 19: Pruning example for clean data

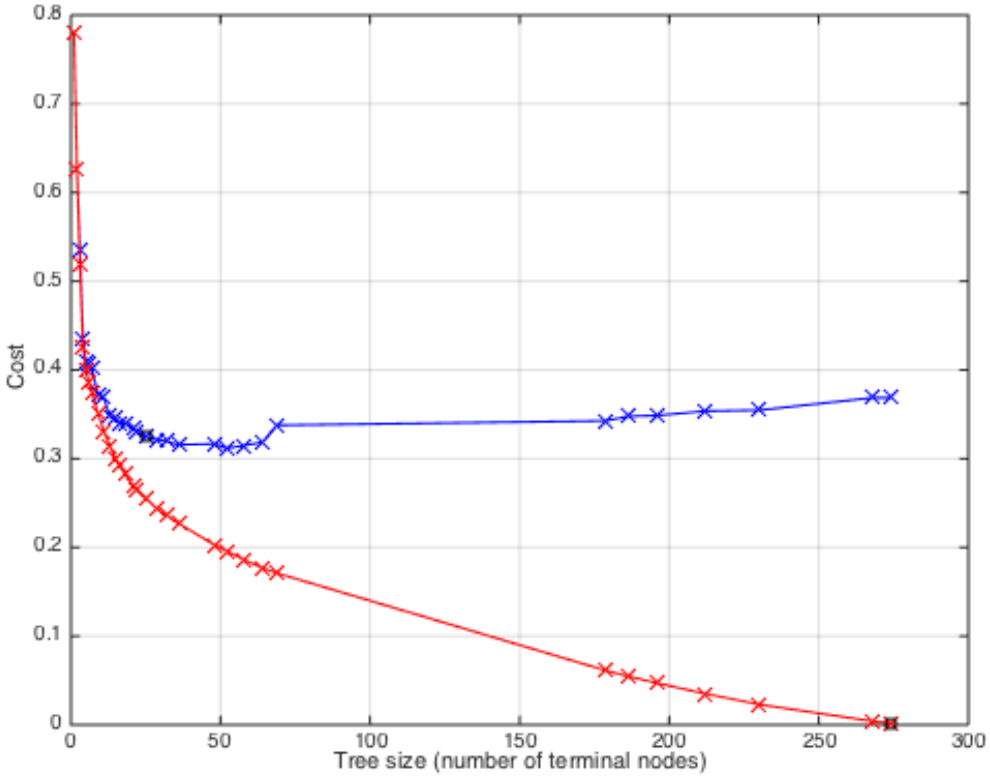


Figure 20: Pruning example for noisy data

6 Conclusion

We provide five strategies for ambiguity as discussed above. But the reliability of strategy 5 is not absolutely assured. For the certainty and stability of testing trees, strategy 3 is chosen to be the default strategy in order to avoid any overfitting problem.

There is a small flaw in the clean dataset. At least one of two rows with the same example attributes and different targets should be noise. This flaw is affecting our assignment a lot; however the report is based on provided clean dataset without any modification. There are two trained tree sets submitted for provided clean dataset and modified absolutely clean dataset.

There are still lots of points need to be improved for the running speed and less using memory. In addition, the coupling level between classes is a bit strong, this will affect the reuse of the code. We believe we will do better on Neutral Network and CBR assignments.