Overall good job on the report, with only minor points removed for not explaining the tie break in selecting the best attribute and missing justification of using both min and max depth for ambiguity.

CO395 - CBC Group 61
Decision Tree Coursework

Belen Barbed, Devin Nanayakkara, Piotr Pomienski, Benjamin Withers

February 13, 2018

Abstract

This report covers our implementation of decisions trees that determines emotions from input vectors of Action Units. The code was written in MATLAB and evaluation of the performance of our implementation was also done in MATLAB. An analysis is given within, along with some answers to some questions posed in the coursework specification.

1 Introduction

Emotion recognition is something most people do effortlessly, our brains can perform this subconsciously. People can usually immediately tell whether the person they see is happy, sad, or angry, yet it is very difficult to make a machine perform the same task. Even with advanced facial feature recognition, standard algorithms are not always up to the task. Machine learning offers a potential solution to this problem, as instead of formulating a complicated computer program, the machine can learn to distinguish different emotions, much like a real person would.

For this task, we were given examples of 6 basic emotions - anger, disgust, fear, happiness, sadness and surprise - and their corresponding facial expressions described using the Facial Action Coding System (FACS). It splits up images of the human face into multiple smaller sections, allowing for easier identification of which muscles, or Action Units, are being used for any given expression. There were around 1000 examples, using which the algorithm would be both taught and evaluated.

2 Implementation

2.1 Cross-Validation

For this exercise, 10-fold cross-validation was used. The entire input sample set x was partitioned into 10 equal size sets, or folds. 9 folds would be used as the training data for each iteration of the training and testing process, with the remaining fold being selected as the test set. The fold used for the test set would be different for each iteration, ensuring that each fold would be used exactly once after all 10 iterations and this was done manually instead of using a loop.

The actual target set y was also partitioned into 10 sets, which were used to build up and train decision trees within each iteration. The trees, along with the test set, would then used to generate predicted targets. These predicted targets would be appended to a vector of all predicted targets across all 10 iterations and used at the end, along with the actual target set y, to generate the average confusion matrix and associated performance metrics.

2.2 Selection of Best Attribute

For each iteration whilst building/training the decision trees, the algorithm checks at each node which attribute contributes the most toward achieving the actual target. This is determined by calculating the information gain. The attribute which results in the highest Information Gain, will

be declared as the best decision attribute for the respective node. The Information Gain of an attribute is calculated using the following equations.

p : number of positive examples
n : number of negative examples
p_0 : number of positive examples of the data subset where the attribute has the value 0
n_0 : number of negative examples of the data subset where the attribute has the value 0
p_1 : number of positive examples of the data subset where the attribute has the value 1
n_1 : number of negative examples of the data subset where the attribute has the value 1

$$InformationGain(attribute) = Entropy(p,n) - \frac{p_0+n_0}{p+n}Entropy(p_0,n_0) - \frac{p_1+n_1}{p+n}Entropy(p_1,n_1) \quad (1)$$

$$Entropy(p,n) = -\frac{p}{p+n}\log_2\left(\frac{p}{p+n}\right) - \frac{n}{p+n}\log_2\left(\frac{n}{p+n}\right) \quad (2)$$

When the dataset arrives to the *choose_ best_ decision_ attribute* function, the Information Gain for each attribute is calculated with respect to the actual targets. Formally, the attribute with the highest information gain is returned. However, there are instances when the best attribute cannot be chosen from the Information Gain. This is due to the fact that the same example results in both positive and negative targets. For example, a dataset may have 2 equal examples but the associated targets may have one positive and the other negative. Hence, there exists no information to build the tree further. In such cases the algorithm randomly selects a target which results in a leaf node.

Why randomly? How can you justify the correctness of the algorithm?

2.3 Computing Average Results

The average confusion matrix is obtained by getting the average across all 10 iterations. The average recall rates, average precision rates, F_1-measures, and average classification rates are calculated using the average confusion matrix determined in the previous step. Rates and measures mentioned were calculated using the following equations.

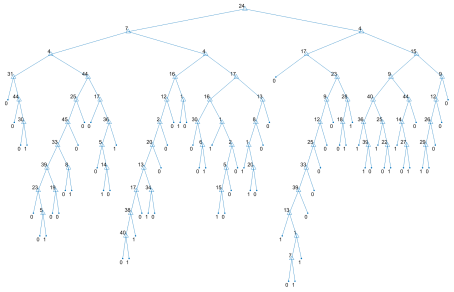
TP : True Positives, TN : True Negatives, FP : False Positives, FN : $FalseNegatives$

$$RecallRate = \frac{TP}{TP+FN} \times 100 \quad (3) \qquad PrecisionRate = \frac{TP}{TP+FP} \times 100 \quad (4)$$

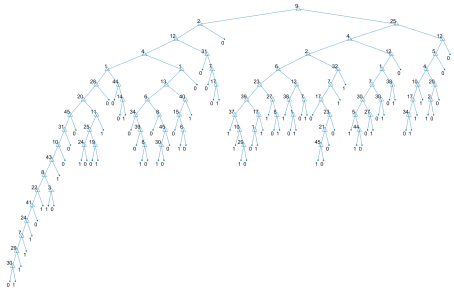
$$F_1 = 2 \times \frac{PrecisionRate * RecallRate}{PrecisionRate + RecallRate} \quad (5) \qquad ClassificationRate = \frac{TP+TN}{TP+TN+FP+FN} \quad (6)$$

3 Tree Diagrams

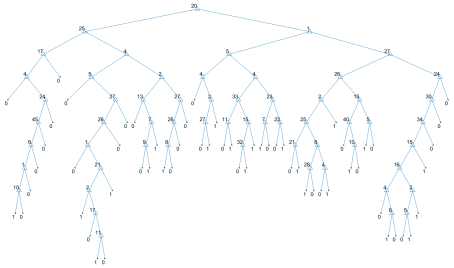
The Decision Tree Visualisations shown in Figure 1 below were generated in MATLAB using the GPLAB toolbox[1] drawtree function. Larger versions of these diagrams are available in the Appendix.



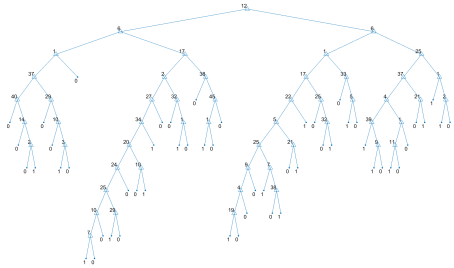
(a) Class 1 - Anger



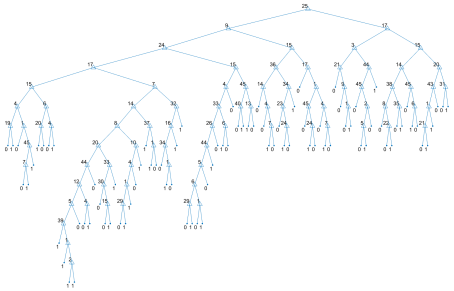
(b) Class 2 - Disgust



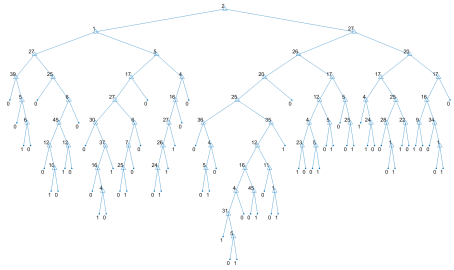
(c) Class 3 - Fear



(d) Class 4 - Happiness



(e) Class 5 - Sadness



(f) Class 6 - Surprise

Figure 1: Decision Tree Visualisations

4 Results

Each example must be classified as one emotion class. There are instances where the example cannot be classified into an emotion class. In such cases all class trees arrive at a 0 indicating that this example does not belong to this class. Hence 3 strategies are introduced to deal with such instances. Strategy 1 is the random method where a class is randomly assigned to an unclassified example. Strategy 2 is the minimum depth method where the class assigned is based on the least depth travelled through the nodes in a class tree. Strategy 3 is the maximum depth method where the class assigned is the deepest tree reached by the unclassified example. More information on these strategies can be found in Section 5.2 (Ambiguity).

4.1 Strategy 1: Random Method

		Predicted Class					
		1	2	3	4	5	6
Actual Class	1	8.500	1.900	0.500	0.800	1.100	0.400
	2	1.300	14.300	0.400	1.600	1.500	0.700
	3	0.800	0.500	8.100	0.200	1.100	1.200
	4	0.300	1.300	1.000	17.500	0.500	1.000
	5	1.800	1.500	1.000	1.100	7.000	0.800
	6	0.100	0.700	1.000	1.000	0.900	17.000

Table 1: Average Confusion Matrix - Random Method - Clean Dataset

	Class						
	1	2	3	4	5	6	Average
Recall Rate	64.394	72.222	68.067	81.019	53.030	82.126	70.143
Precision Rate	66.406	70.792	67.500	78.829	57.851	80.569	70.325
F1 Measure	65.385	71.500	67.782	79.909	55.336	81.340	70.209
CR	91.036	88.645	92.331	91.235	88.745	92.231	90.704

Table 2: Evaluation Results (%) - Random Method - Clean Dataset

By observing Table 2 on the clean dataset, emotion Class 3 and emotion Class 6 have the highest accuracy with classification rates 92.3% and 92.2% respectively. The emotion Class 5 has a classification rate of 88.7%, however it has the lowest recall rate and precision rate with 53.0% and 57.9% respectively. This implies that almost half the actual positive examples are missed and about 42% of the predicted positive examples are false positives. Hence emotion Class 5 is the most confused.

		Predicted Class					
		1	2	3	4	5	6
Actual Class	1	2.400	0.900	2.000	0.900	2.000	0.600
	2	1.100	13.000	1.400	1.800	0.600	0.800
	3	1.800	1.500	10.300	2.400	0.800	1.900
	4	1.000	1.200	1.200	15.500	0.700	1.300
	5	1.800	0.900	0.900	1.100	5.400	0.900
	6	1.300	1.100	1.900	1.300	1.200	15.2000

Table 3: Average Confusion Matrix - Random Method - Noisy Dataset

	Class						
	1	2	3	4	5	6	Average
Recall Rate	27.273	69.519	55.080	74.163	49.091	69.091	57.370
Precision Rate	25.532	69.892	58.192	67.391	50.467	73.430	57.484
F1 Measure	26.374	69.705	56.593	70.615	49.770	71.194	57.375
CR	86.613	88.711	84.216	87.113	89.111	87.712	87.246

Table 4: Evaluation Results (%) - Random Method - Noisy Dataset

By observing Table 4 on the noisy dataset, emotion Class 5 and emotion Class 2 have the highest accuracy with classification rates 89.1% and 88.7% respectively. However emotion Class 5, misses more than half the actual positive examples (49% recall rate) and almost half the predicted positive examples are incorrect (50% precision rate). But emotion Class 1 is the most confused having a recall rate of 27.3% (high false negatives) and a 25.5% precision rate (high false positives).

4.2 Strategy 2: Minimum Depth Method

		Predicted Class					
		1	2	3	4	5	6
Actual Class	1	8.900	1.100	0.700	0.800	1.200	0.500
	2	1.800	13.600	0.500	1.100	1.500	1.300
	3	0.400	0.400	8.400	0.300	0.800	1.600
	4	0.200	1.500	0.500	18.100	0.900	0.400
	5	2.200	1.300	0.900	1.100	7.000	0.700
	6	0.300	0.400	1.200	0.900	0.800	17.100

Table 5: Average Confusion Matrix - Minimum Depth Method - Clean Dataset

	Class						
	1	2	3	4	5	6	Average
Recall Rate	67.424	68.687	70.588	83.796	53.030	82.609	71.022
Precision Rate	64.493	74.317	68.852	81.166	57.377	79.167	70.895
F1 Measure	65.926	71.391	69.710	82.460	55.118	80.851	70.909
CR	90.837	89.143	92.729	92.331	88.645	91.932	90.936

Table 6: Evaluation Results (%) - Minimum Depth Method - Clean Dataset

By observing Table 6 on the clean dataset, emotion Class 3 and emotion Class 4 have the highest accuracy with classification rates 92.7% and 92.3% respectively. The most confused emotion is Class 5 which has only a 53% recall rate (high false negatives) and a 57% precision rate (high false positives).

		Predicted Class					
		1	2	3	4	5	6
Actual Class	1	2.100	1.100	2.000	1.000	1.800	0.800
	2	1.700	12.600	1.800	1.500	0.500	0.600
	3	1.600	2.300	10.000	2.200	1.000	1.600
	4	1.100	1.500	0.900	15.800	0.500	1.100
	5	2.000	1.100	0.400	1.100	5.600	0.800
	6	1.200	0.600	1.800	1.300	1.600	15.500

Table 7: Average Confusion Matrix - Minimum Depth Method - Noisy Dataset

	Class						
	1	2	3	4	5	6	Average
Recall Rate	23.864	67.380	53.476	75.598	50.909	70.455	56.947
Precision Rate	21.649	65.625	59.172	68.996	50.909	75.980	57.055
F1 Measure	22.703	66.491	56.180	72.146	50.909	73.113	56.924
CR	85.714	87.313	84.416	87.812	89.211	88.611	87.179

Table 8: Evaluation Results (%) - Minimum Depth Method - Noisy Dataset

By observing Table 8 on the noisy dataset, the highest accuracy can be found in emotions Class 5 and Class 6 with classification rates 89.2% and 88.6% respectively. Similar to Table 4, emotion class 5 has only 50.9% recall rate and 50.9% precision rate. Hence almost half the actual positive examples are missed and almost half the predicted positive examples are false positives. However emotion Class 1 is the most confused due to very low recall rate (23.9%) and precision rate (21.6%).

4.3 Strategy 3: Maximum Depth Method

		Predicted Class					
		1	2	3	4	5	6
Actual Class	1	8.300	1.700	0.400	0.900	1.300	0.600
	2	1.200	15.000	0.700	1.000	1.200	0.700
	3	1.100	0.900	8.100	0.000	0.600	1.200
	4	0.200	1.300	0.600	17.600	1.100	0.800
	5	2.000	1.300	0.500	1.200	7.800	0.400
	6	0.400	1.000	1.300	0.300	0.800	16.900

Table 9: Average Confusion Matrix - Maximum Depth Method - Clean Dataset

	Class						
	1	2	3	4	5	6	Average
Recall Rate	62.879	75.758	68.067	81.481	59.091	81.643	71.486
Precision Rate	62.879	70.755	69.828	83.810	60.938	82.039	71.708
F1 Measure	62.879	73.171	68.936	82.629	60.000	81.840	71.576
CR	90.239	89.044	92.729	92.629	89.641	92.530	91.135

Table 10: Evaluation Results (%) - Maximum Depth Method - Clean Dataset

By observing Table 10 on the clean dataset, the emotions in Class 3, Class 4, and Class 6 have the highest accuracy with classification rates 92.7%, 92.6% and 92.5% respectively. The most confused emotion is found in Class 5 with a 59.1% recall rate and a 60.9% precision rate despite having an accuracy (classification rate) of 89.6%.

		Predicted Class					
		1	2	3	4	5	6
Actual Class	1	2.800	0.800	1.600	0.800	2.000	0.800
	2	1.400	12.800	1.300	1.700	0.800	0.700
	3	1.800	1.400	10.600	1.600	1.000	2.300
	4	1.300	1.700	1.900	14.100	0.900	1.000
	5	2.200	0.700	0.900	0.900	5.300	1.000
	6	1.200	1.100	2.400	1.400	1.600	14.300

Table 11: Average Confusion Matrix - Maximum Depth Method - Noisy Dataset

	Class						
	1	2	3	4	5	6	Average
Recall Rate	31.818	68.449	56.684	67.464	48.182	65.000	56.266
Precision Rate	26.168	69.189	56.684	68.780	45.690	71.144	56.276
F1 Measure	28.718	68.817	56.684	68.116	46.903	67.933	56.195
CR	86.114	88.412	83.816	86.813	88.012	86.513	86.613

Table 12: Evaluation Results (%) - Maximum Depth Method - Noisy Dataset

By observing Table 12 on the noisy dataset, the most accurate emotion is in Class 2 with a classification rate of 88.4% followed by Class 5 with 88.0%. However similar to Table 4 and Table 8, Class 5 has a low recall rate of (48.1%) and a low precision rate of (45.7%). But emotion Class 1 is recognised to be confused because most of the actual positives are missed (31.8% recall rate) and most of the predicted positives are false positives (26.2%).

5 Questions

5.1 Is there any difference in the performance when using the clean and noisy datasets?

	Class						
	1	2	3	4	5	6	Average
Random Clean	91.036	88.645	92.331	91.235	88.745	92.231	90.704
Random Noisy	86.613	88.711	84.216	87.113	89.111	87.712	87.246
Min. Depth Clean	90.837	89.143	92.729	92.331	88.645	91.932	90.936
Min. Depth Noisy	85.714	87.313	84.416	87.812	89.211	88.611	87.179
Max. Depth Clean	90.239	89.044	92.729	92.629	89.641	92.530	91.135
Max. Depth Noisy	86.114	88.412	83.816	86.813	88.012	86.513	86.613
Average Clean	90.704	88.944	92.596	92.065	89.010	92.231	
Average Noisy	86.147	88.145	84.149	87.246	88.778	87.612	

Table 13: Performance comparison (using CR as %) of clean and noisy datasets by method

As shown in Table 13, there is noticeable difference in the performance of the trees depending on the dataset they are sorting. The noisy set consistently performs 3-5% worse than the clean one regardless of the method of disambiguation used. This difference comes from the errors within the noisy dataset that were generated by the automated system for AU recognition.

This difference, however, is not consistent across classes, as some (2 and 5) have very little performance differences between the noisy and clean datasets. This can be due to the fact that the automated system is especially good at recognising those emotions and so the noisy set is more accurate for those classes than for others. On the other hand, class 3 has noticeably worse CR in the noisy dataset, which shows that the automated system is not particularly good at recognising this emotion.

2

5.2 Ambiguity

As explained in Section 4 (Results), there are instances where the example cannot be classified into an emotion class, as the tress for all 6 classes return 0. To disambiguate these cases, three different methods were implemented: random, minimum tree depth, and maximum tree depth.

5.2.1 Random

The random method involves assigning a random class to the example, having a uniform distribution of probability across the 6 classes.

This is the easiest method to implement, as it requires no knowledge of how the example traversed each tree. Because of this, it was expected that the approach would heavily underperform the others, as it is just blindly assigning classes to the ambiguous examples. This was not found to be the case, however.

This method performs the worst out of the three in the clean dataset, as expected - albeit by a negligible amount of 0.232%. However, it yields better results than the two others on the noisy dataset by an equally small amount, which was not expected. This finding can make sense if one considers the automated system to fail to correctly classify examples in a random manner.

5.2.2 Minimum Depth

Minimum tree depth is a method by which the example will be assigned the class of the tree it travelled the least depth into.

Compared to random, the minimum depth method requires more computation and knowledge of how the example traversed each of the 6 class trees. It will result in choosing the class that most quickly determined the result for said example.

As expected, minimum depth performed better than random and worse than maximum depth in the clean dataset, but unexpectedly fell behind random in the noisy set.

5.2.3 Maximum Depth

Maximum tree depth involves choosing the class in which the deepest leaf was reached by the unclassified example. This results in choosing the class that was the most undecided when classifying the example. The objective is to not pick classes that have quickly discarded the example as a negative for their emotion. This method was expected to perform the best out of all implemented.

As expected, this method performed the best out of the three in the clean dataset, by a 0.199% margin compared to minimum tree depth. However, it fared the worst in the noisy set, which was an unexpected result.

You cannot use both minimum and maximum depth, without justifying why, because the principles are the opposite, should one not be way better than the other?

5.3 Pruning

The given pruning_example code creates decision trees and uses those to predict responses. It takes the same input as our implementation of the decision trees. This code generates cost, node and best level vectors for the trees using two methods: resubstitution and cross-validation. These vectors most likely can be used to estimate the error rate or performance of the learning algorithm used.

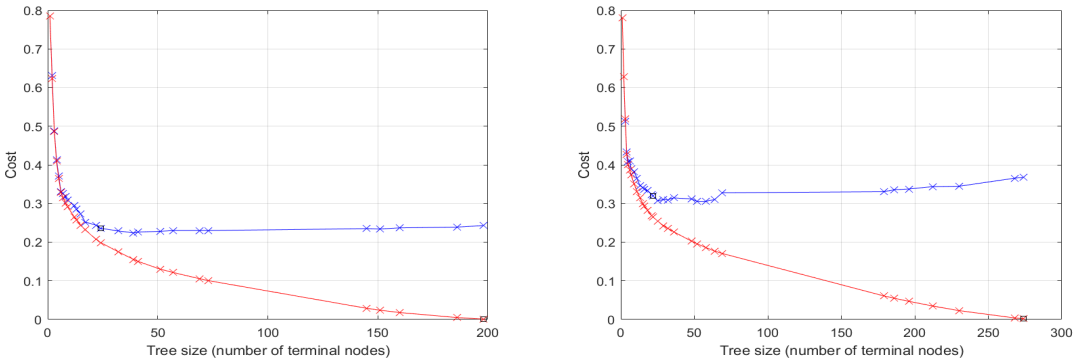
Resubstitution uses the entire sample set given (y) to estimate the cost, node and best level vectors over a classifier, or when used normally, even the error rate. Resubstitution is a faster alternative to cross-validation, however the predicted error rate accuracy would drop if exposed to an unseen data set.

Cross-validation is the method we used for our implementation and the pruning example does almost the same. The sample set is partitioned into 10 sets as well, however in this case, 1 set is used for validation and the other 9 for training. Just like before, this is repeated 10 times such that each of the 10 sets is used as the validation set. The average error estimate is then calculated and may be used to help identifying any "overfitting" involved with running the validation process on the training data.

After validation, the trees are pruned using MATLAB's prune function. This most likely reduces the number of leaves and branches in the tree in order to reduce computational cost and potentially reduce any overfitting.

The pruning_example code then plots graphs (shown in Figure 2) with cost on the y-axis and tree size (in terms of terminal nodes from the nodes vector) on the x-axis. The red line is resubstitution and blue is cross-validation and two graphs were generated, one for the clean dataset and another for the noisy dataset. One can see that cross-validation results in additional cost compared to resubstitution for increasing tree size. Cross-validation seems to be heavily influenced by noisy data, whereas resubstitution is unaffected. In both cases cross-validation remains at constant cost for increasing tree size after around 50 nodes, whereas resubstitution continues to cause a reduction in cost. This suggests that resubstitution is worth doing for larger tree sizes and cross-validation is better at smaller tree sizes.

Good job!



(a) Clean Dataset (b) Noisy Dataset

Figure 2: Pruning Example on Clean and Noisy Datasets

References

[1] S. Silva, "GPLAB - A Genetic Programming Toolbox for MATLAB", Gplab.sourceforge.net, 2016. [Online]. Available: http://gplab.sourceforge.net/. [Accessed: 10- Feb- 2018].

6 Appendix - Tree Visualisation Diagrams

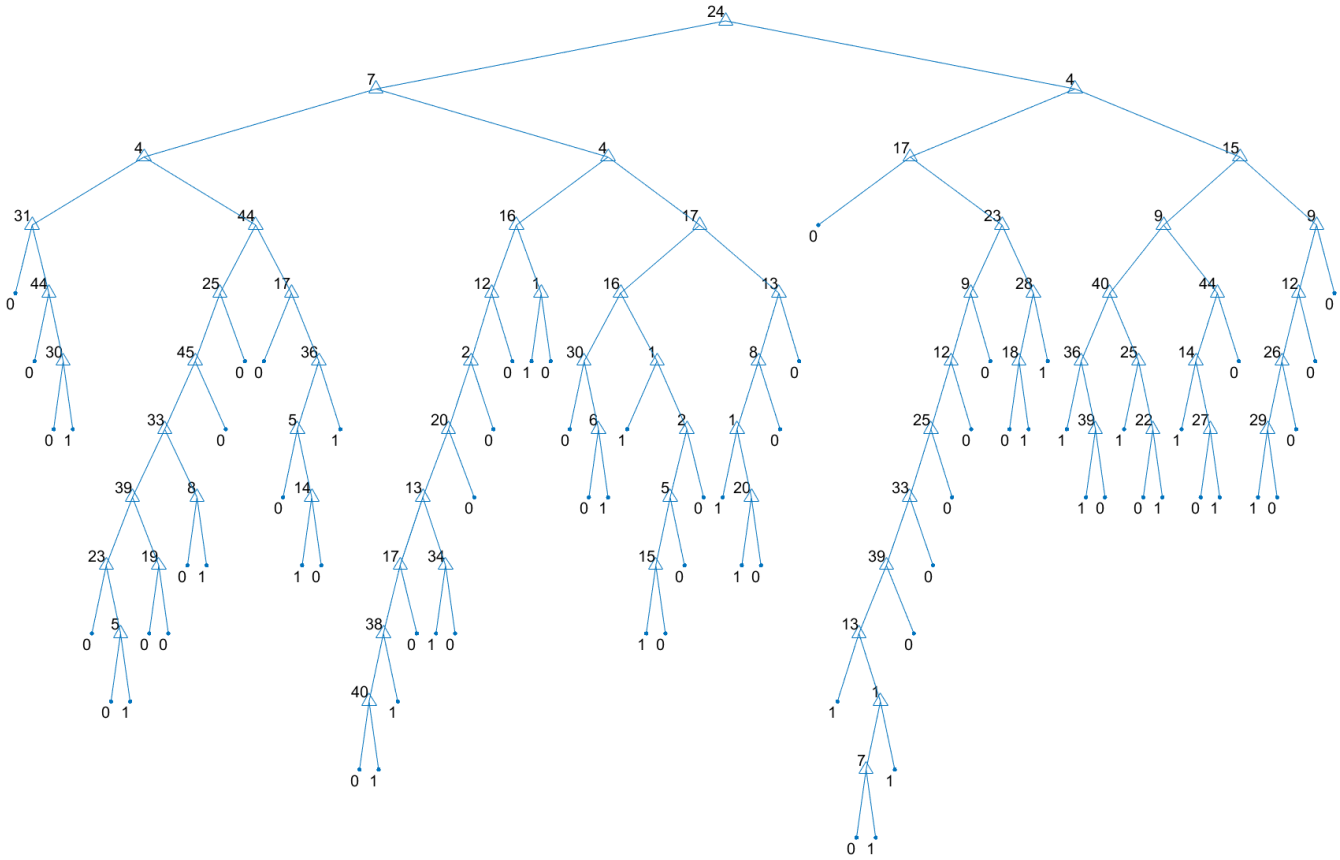


Figure 3: Class 1 - Anger

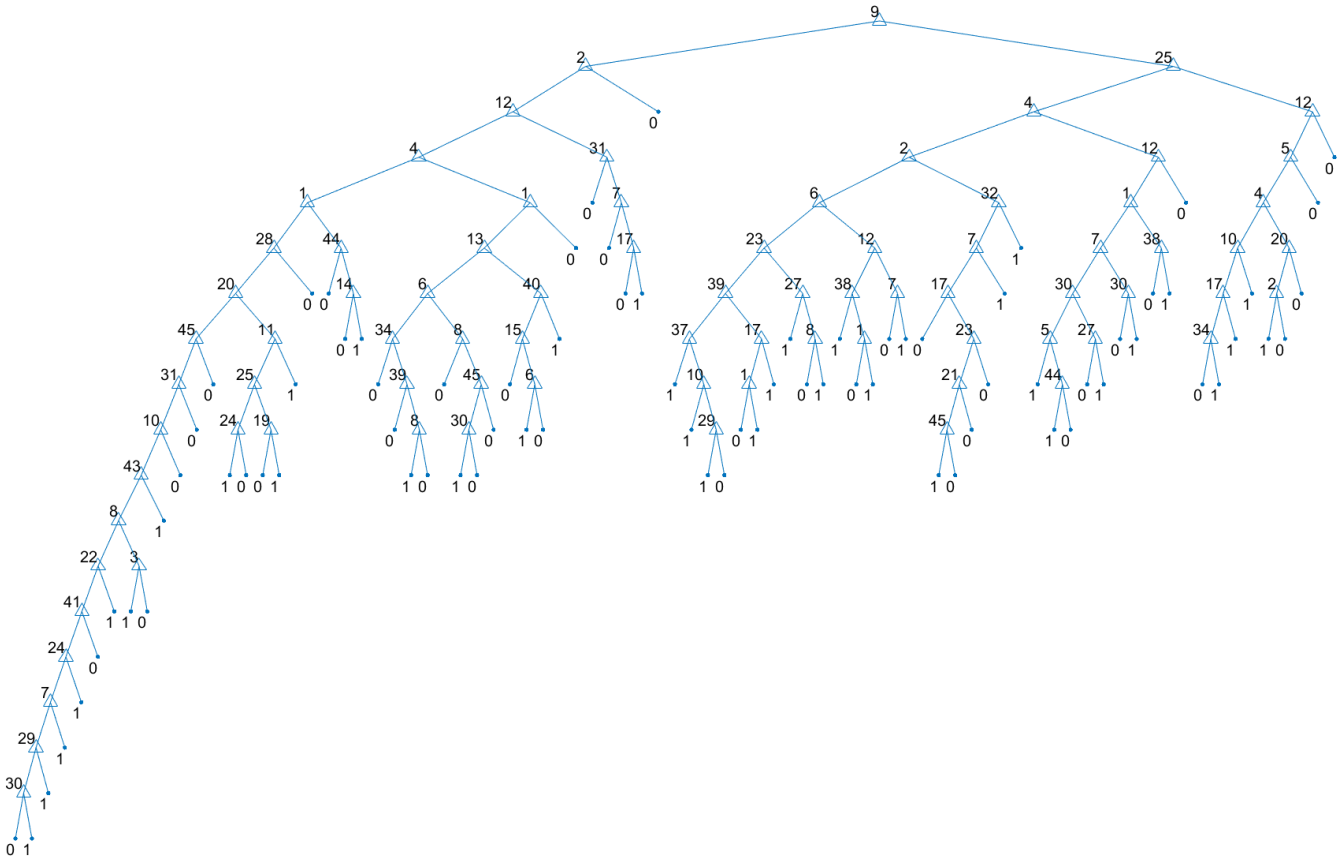


Figure 4: Class 2 - Disgust

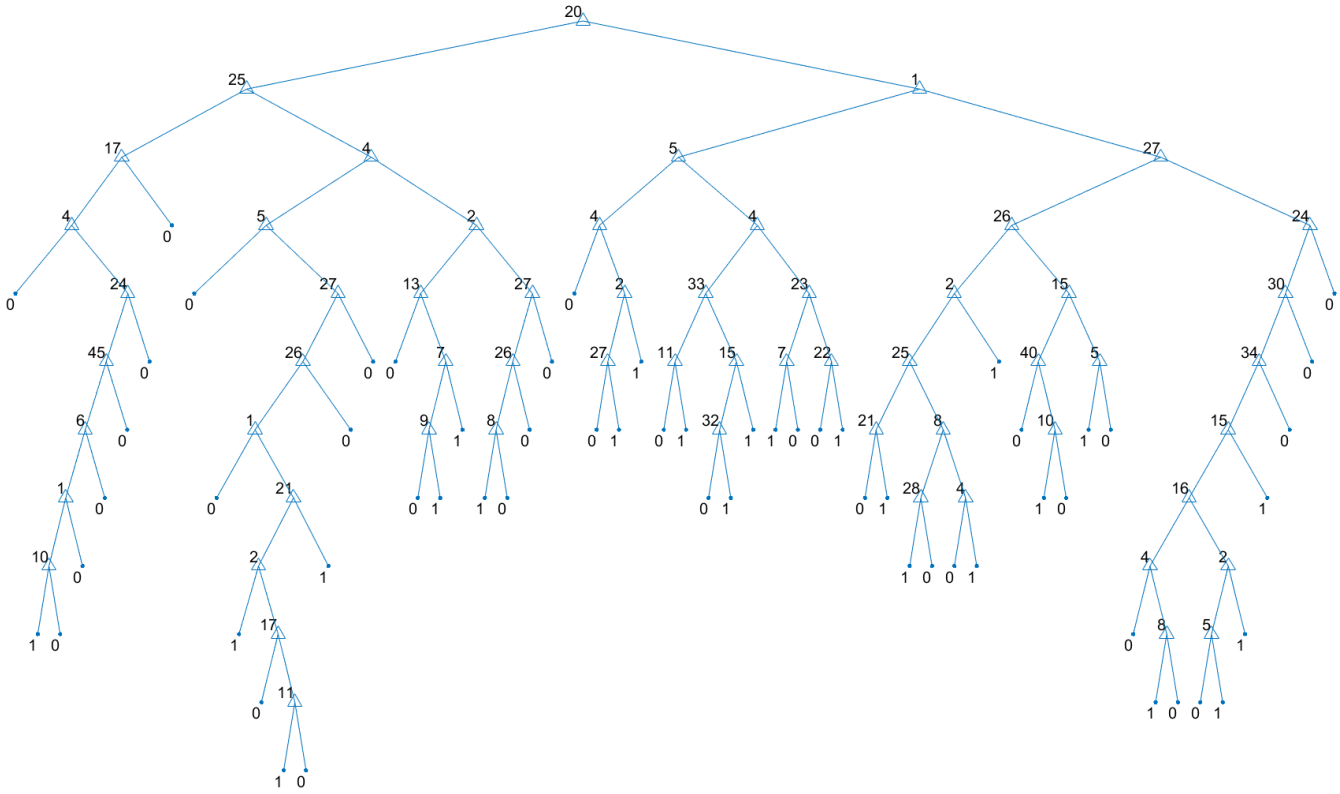


Figure 5: Class 3 - Fear

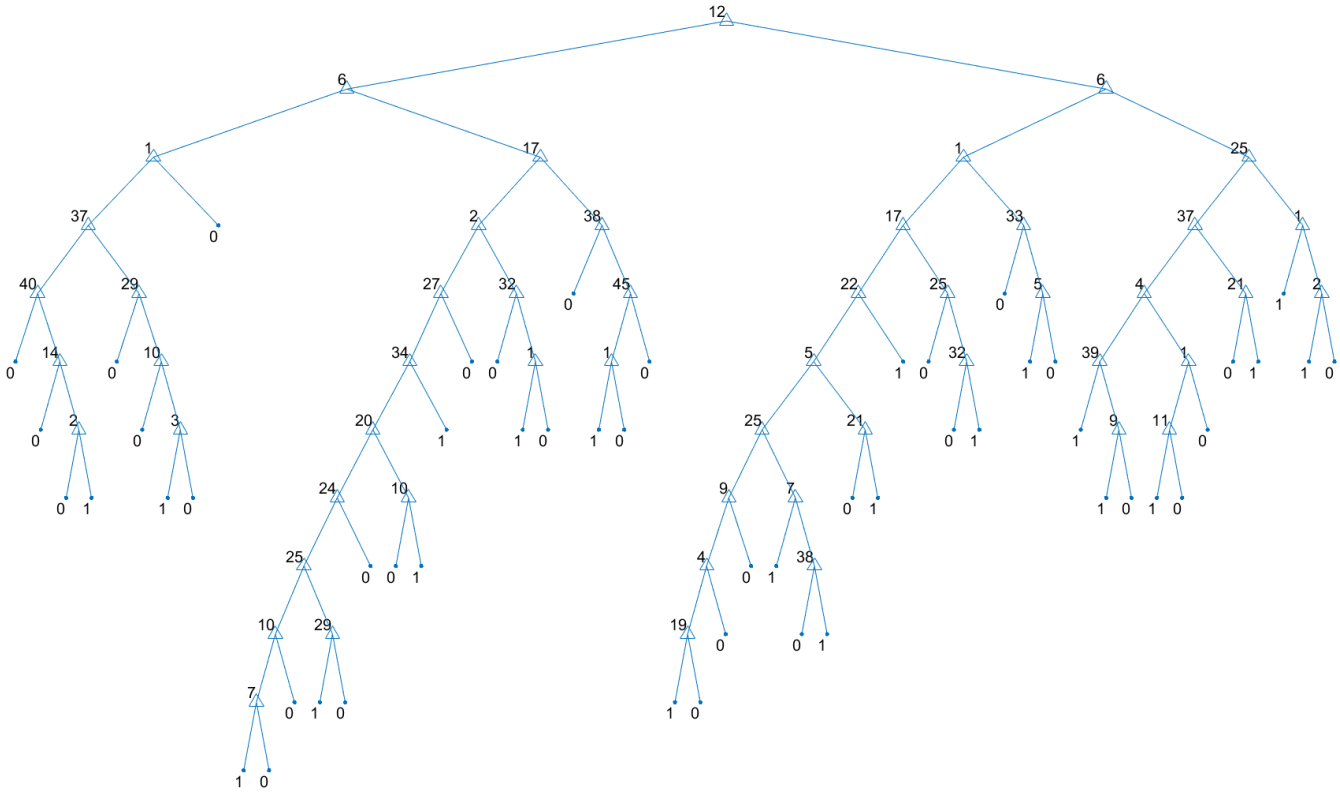


Figure 6: Class 4 - Happiness

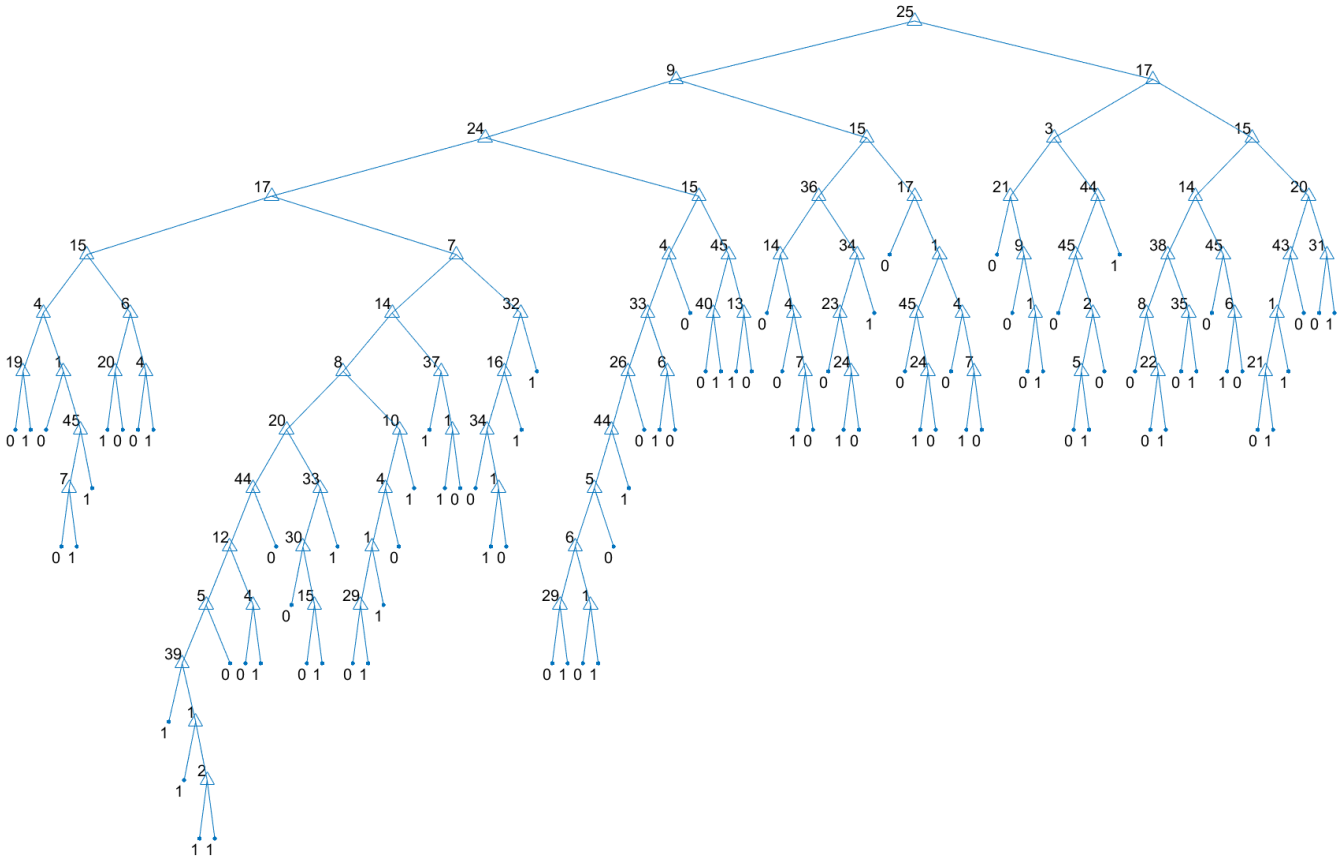


Figure 7: Class 5 - Sadness

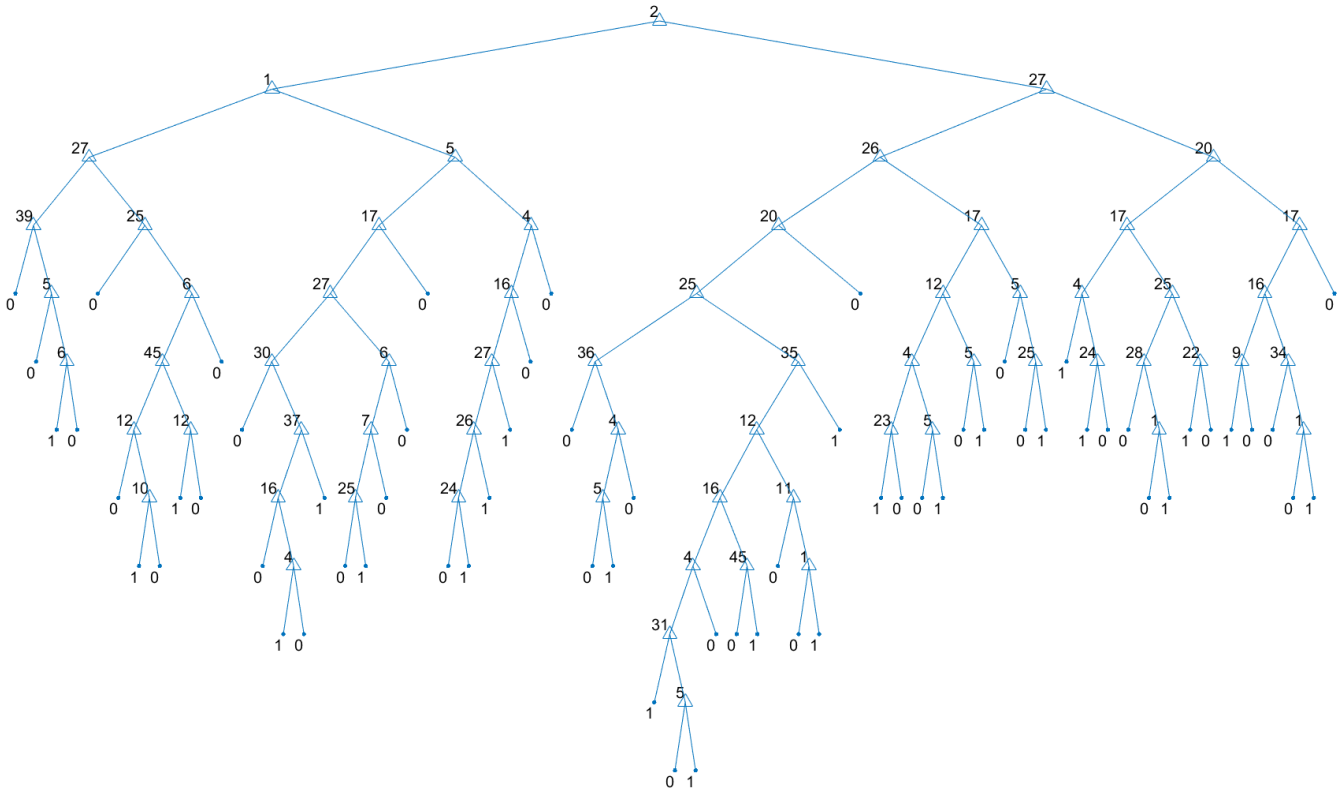


Figure 8: Class 6 - Surprise