CO395 Group 57

Dongxiao Huang, Zheng Xun Phang, Yufeng Zhang

Implementation

How do we select the best attribute at each node? For all attributes, we compute the information gain if the dataset is split on that attribute:

$$\text{Gain(Attribute)} = \mathcal{I}(p,n) - \left[\frac{p_0+n_0}{p+n}\mathcal{I}(p_0,n_0) + \frac{p_1+n_1}{p+n}\mathcal{I}(p_1,n_1)\right]$$

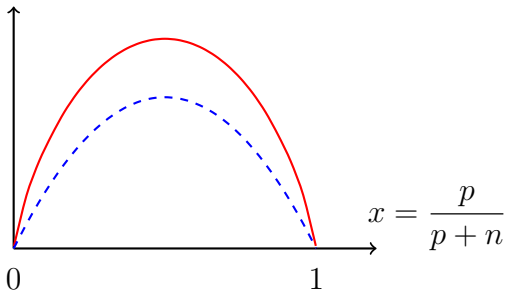
$$\begin{aligned} p_k &= \text{Number of positive examples with (attribute} = k) \\ n_k &= \text{Number of negative examples with (attribute} = k) \\ p &= p_0 + p_1 = \text{Number of positive examples before split} \\ n &= n_0 + n_1 = \text{Number of negative examples before split} \end{aligned}$$

There are two common ways to measure information.

Entropy:
$$\mathcal{I}(p,n) = -\frac{p}{p+n}\log\frac{p}{p+n} - \frac{n}{p+n}\log\frac{n}{p+n} \qquad \text{if } p,n \neq 0$$
$$\mathcal{I}(p,0) = \mathcal{I}(0,n) = 0$$

Gini impurity:
$$\mathcal{I}(p,n) = \frac{p}{p+n}\left(1-\frac{p}{p+n}\right) + \frac{n}{p+n}\left(1-\frac{n}{p+n}\right)$$

We used entropy since it is stated in the specification, but Gini impurity is faster to compute. Both metrics should give similar results since their graphs have a similar shape:



Entropy: $-x\log x - (1-x)\log(1-x)$

Gini: $2x(1-x)$

N.B. The *height* of those graphs is irrelevant because minimizing a function is equivalent to minimizing any positive multiple of that function. What matters is their *shape*.

The decision tree algorithm is given in the specification, so it's unnecessary to repeat it here. We try to use NumPy functions instead of Python loops for performance since the former run on vectorized C code.

To evaluate our decision tree, we performed K-fold cross validation as follows:

1. Shuffle the dataset and split it into $K = 10$ parts
2. For each $k \in \{1, \cdots, K\}$ we train the decision tree on the dataset *excluding* part k and then test the decision tree on part k. During testing, we aggregate the predictions from 6 trees (see Ambiguity Question) and increment the relevant cells in the confusion matrix.

Evaluation

Each cell of the confusion matrix is a total, not an average, over all folds of cross validation. There were $n = 1000$ test examples in total.

Warning: It makes no sense to compute classification rate for each emotion separately. For example, if the actual emotion is Anger then (Fear, Surprise) counts as a true negative for Anger even though it is a misclassification! In other words, if an instance of Fear is misclassified as Surprise, it should not be taken as evidence that our trees are good at detecting Anger.

Clean Data

Actual Class	Predicted Class					
	Anger	Disgust	Fear	Happiness	Sadness	Surprise
Anger	97	16	3	4	9	2
Disgust	19	159	1	7	9	3
Fear	7	4	90	2	3	12
Happiness	1	11	2	191	5	5
Sadness	18	15	5	9	79	6
Surprise	1	3	11	5	7	179

We add up the relevant cells in the confusion matrix above to compute these summary statistics:

	Anger	Disgust	Fear	Happiness	Sadness	Surprise
Precision	0.678	0.764	0.804	0.876	0.705	0.865
Recall	0.740	0.803	0.763	0.888	0.598	0.869
F_1 score	0.708	0.783	0.783	0.882	0.648	0.867
—	0.920	0.912	0.950	0.949	0.914	0.945

The last row in the table above is the "classification rate" of each emotion. As explained earlier, these figures are meaningless. Instead we compute the classification rate of all 6 trees as

$$\frac{97 + 159 + \cdots + 179}{1000} = 0.795$$

Noisy Data

Actual Class	Predicted Class					
	Anger	Disgust	Fear	Happiness	Sadness	Surprise
Anger	30	10	17	8	16	7
Disgust	13	141	14	11	4	4
Fear	12	15	116	21	8	15
Happiness	8	11	12	166	3	8
Sadness	22	10	5	5	59	9
Surprise	6	8	15	11	7	173

Summary statistics as before:

	Anger	Disgust	Fear	Happiness	Sadness	Surprise
Precision	0.330	0.723	0.648	0.748	0.608	0.801
Recall	0.341	0.754	0.620	0.798	0.536	0.786
F_1 score	0.335	0.738	0.634	0.772	0.570	0.794
—	0.881	0.900	0.866	0.902	0.911	0.910

$$\text{Classification rate} = \frac{30 + 141 + \cdots + 173}{1000} = 0.685$$

The F_α score by van Rijsbergen (1975) is a harmonic mean of precision P and recall R:

$$\frac{1+\alpha^2}{F_\alpha} = \frac{1}{P} + \frac{\alpha^2}{R} \qquad \Rightarrow \qquad F_\alpha = \frac{(1+\alpha^2)PR}{\alpha^2 P + R}$$

which allows us to adjust the weight given to precision or recall. This is useful if, for example, it is essential to detect Anger in an image when it is present. This averaging also enables F_α to deal with imbalanced classes.

On clean data, our 6 trees classify the emotions *Disgust*, *Fear*, *Happiness* and *Surprise* with high accuracy ($F_1 > 0.78$) while *Sadness* is harder to detect.

On noisy data, our 6 trees classify the emotions *Disgust*, *Happiness* and *Surprise* with high accuracy ($F_1 > 0.73$) while *Anger* and *Sadness* are harder to recognize.

We expected emotions to be most frequently misclassified as *Anger* because if none of our 6 trees detect any emotion in an image, then we classify it as *Anger* by default. However, this is not evident from the confusion matrices. In fact, none of our test observations was classified in this manner.

The Facial Action Coding System in the specification has more rules that map to *Anger* and *Fear* than to other emotions. This suggests there are more ways for a face to express anger and fear (through action units) than other emotions, so images are likely to be misclassified as such. However, this hypothesis isn't supported by the confusion matrices.

We could have created another label "No emotion" for images that cannot be classified as any of the 6 emotions. However the specification requires our testTrees(...) function generate predictions in $\{1, \cdots, 6\}$.

Miscellaneous

Noisy-Clean Datasets Question

Overall performance:
Not surprisingly, the noisy dataset has lower overall performance. Its classification rate is 0.685 compared to 0.795 for clean data.

Per emotion performance:
The conclusion is the same when we analyse each emotion separately: precision, recall and F_1 score are lower for noisy data.

Noise degrades the classification accuracy of any machine learning model because it won't be learning the true underlying function that maps features to the target variable. For decision trees, this means taking many wrong branches to fit the noise.

Another intuition is that as noise in the data increases, the features that predict each emotion will become random, and so a decision tree will effectively be guessing randomly.

Among the 6 emotions, *Anger* is most susceptible to noise because a fair number of noisy test examples will not be classified as having any of the 6 emotions, so it is labelled as *Anger* by default. As a result, its F_1 score is 0.335 only.

Ambiguity Question

If no emotion is detected, then we predict *Anger*. But if our 6 trees predict that a given face has more than one emotion, we considered several methods of selecting only one emotion:

1. Select the first detected emotion in alphabetical order

Example: If *Fear* and *Surprise* are detected, then we predict *Fear*.

Advantage: Easy to implement
Disadvantage: This method is arbitrary and will not improve the classification rate

2A. Select the emotion that was detected from the shallowest node

Example: A face gets a positive result from our *Anger* and *Fear* trees. In the *Anger* tree, the face belongs to a node that is 3 levels deep. In the *Fear* tree, it belongs to a node that is 6 levels deep. We classify this face as *Anger*.

Advantage: This is more principled than method 1 and simpler than method 3. It may work well if a shallow tree is sufficient to fit the data, since it ignores deeper but overfitting trees.
Disadvantage: It may not perform well if a deep tree is necessary, since it select the shallowest but underfitting tree.

2B. Select the emotion that was detected from the deepest node

This contrasts with method 2A above.

3. Toggle each action unit in turn and take a majority vote

This is our preferred method. Its rationale is: adding or removing an action unit is akin to hiding a part of a face, which does not change its underlying emotion. Of course, if we change too many action units from present to absent or vice versa, then we cannot expect our trees to classify the face correctly. Hence, we only toggle one action unit at a time.

Example: A face has action units 3 and 12 and it gets a positive result from our *Anger* and *Fear* trees. We toggle each action unit in turn and test it on our 6 trees as follows.

Action Units	Result
1, 3, 12	Fear, Sadness
2, 3, 12	(none)
12	Disgust, Fear
3, 4, 12	Anger, Fear
⋮	⋮

Since *Fear* is the most frequent result, it will be our prediction.

Advantage: This method has the best classification rate among the 3. It is robust to noise.
Disadvantage: Test time increases with the number of action units.

The classification rates for the methods above are computed using 10-fold cross validation:

Classification Rate	Plan 1	Plan 2A	Plan 2B	Plan 3
Clean Data	0.726	0.719	0.715	0.795
Noisy Data	0.594	0.610	0.595	0.685

Pruning Question

The function $pruning\_example(x, y)$ takes two inputs: x is an $n \times p$ matrix of features and y is a n-dimensional vector of target variables.

It builds a classification tree and prunes it to various sizes (as measured by number of leaves instead of depth). It computes the classification cost for each tree size. Finally it returns a plot of classification cost against tree size, which also marks the smallest tree size whose cost is within 1 standard error of the minimum cost.

The two curves in the plot correspond to two methods of computing classification costs.

- Resubstitution: Tests a tree using the same data that was used to fit the tree (i.e. computes in-sample error)
- 10-fold cross validation: See our explanation in the Implementation section earlier. Validation error is an out-of-sample error if we don't use it to tune a hyperparameter such as maximum depth, otherwise it's only an estimate of out-of-sample error.

In figures 1 and 2, we find that classification costs have similar behaviour on both clean and noisy data: Resubstitution error keeps falling as the tree gets larger, while validation error falls initially and then rises.

This is expected because a deeper tree will always fit a dataset better than a shallower tree. However if a tree is too deep, it will overfit the training data but fail to generalize to unseen data, which is manifested in the increasing validation error.

Differences between clean and noisy data:

- For a fixed tree size, resubstitution and validation errors are higher on noisy data than clean data, since noise by definition makes it harder for a tree to fit the data.
- For the same reason, noisy data require more branches, and hence more leaves, to classify all training examples. It took about 275 vs 200 leaves to achieve zero in-sample error.
- The gap between resubstitution and validation errors rises faster for noisy data than clean data, which shows that overfitting is more severe when the data have noise.

Optimal tree size for clean data is about 22 leaves while that for noisy data is 25 leaves.

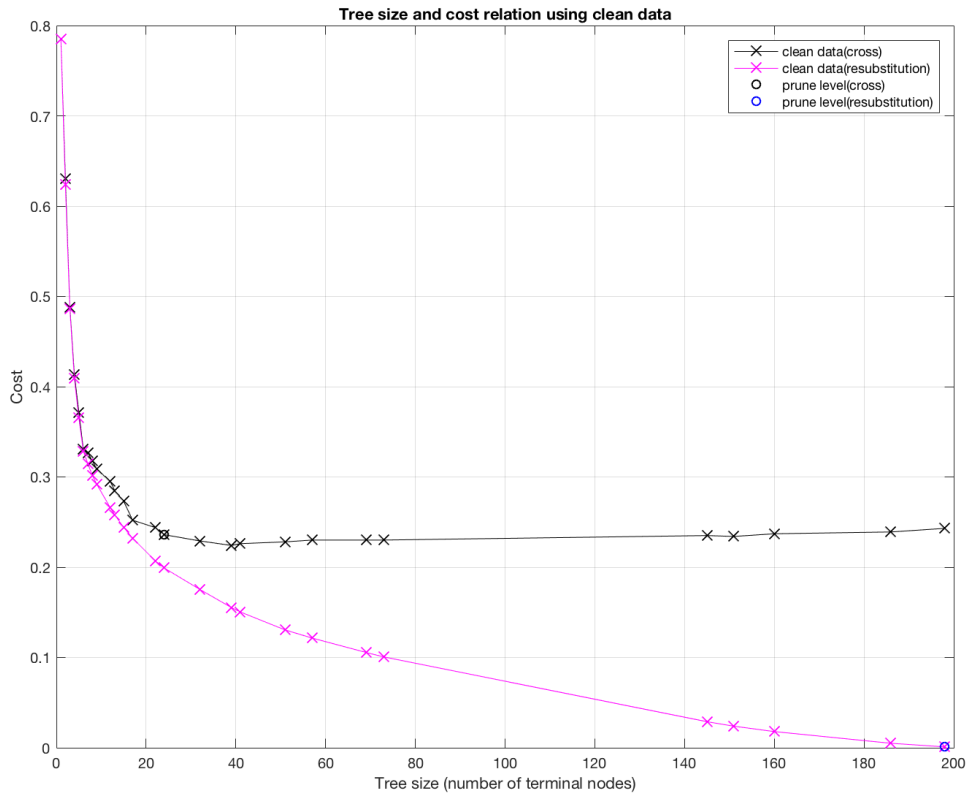


Figure 1: Clean Data

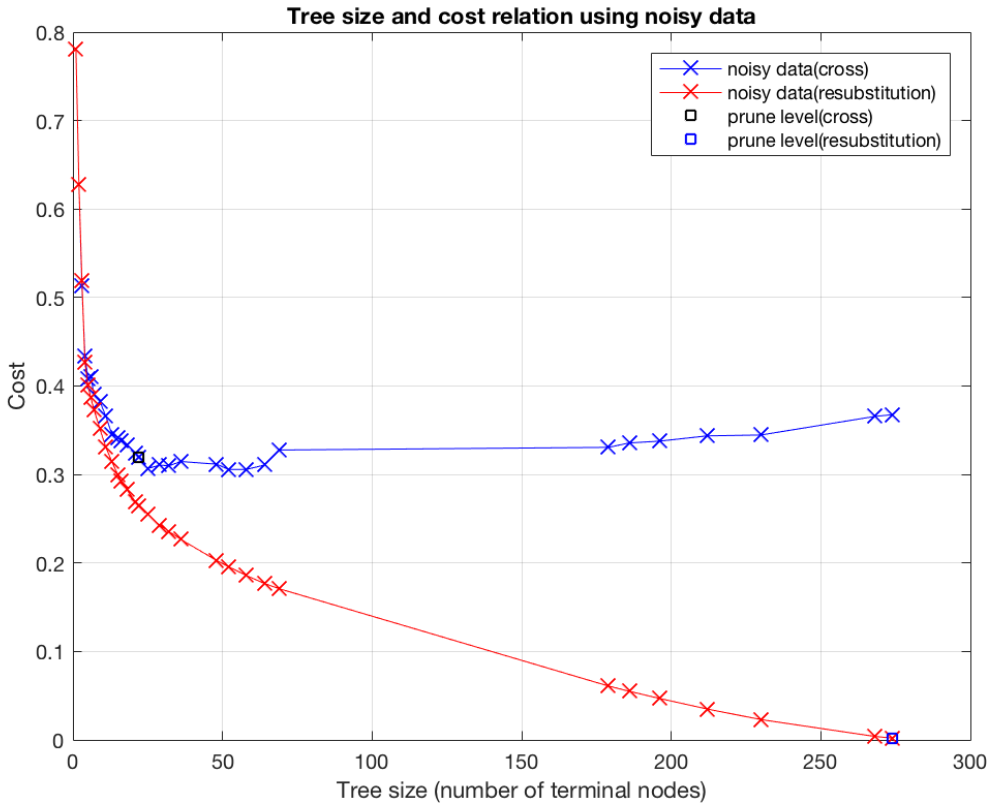


Figure 2: Noisy Data

Tree Diagrams

Figure 3 to 8 are the trees trained on the entire clean dataset (1004 examples).

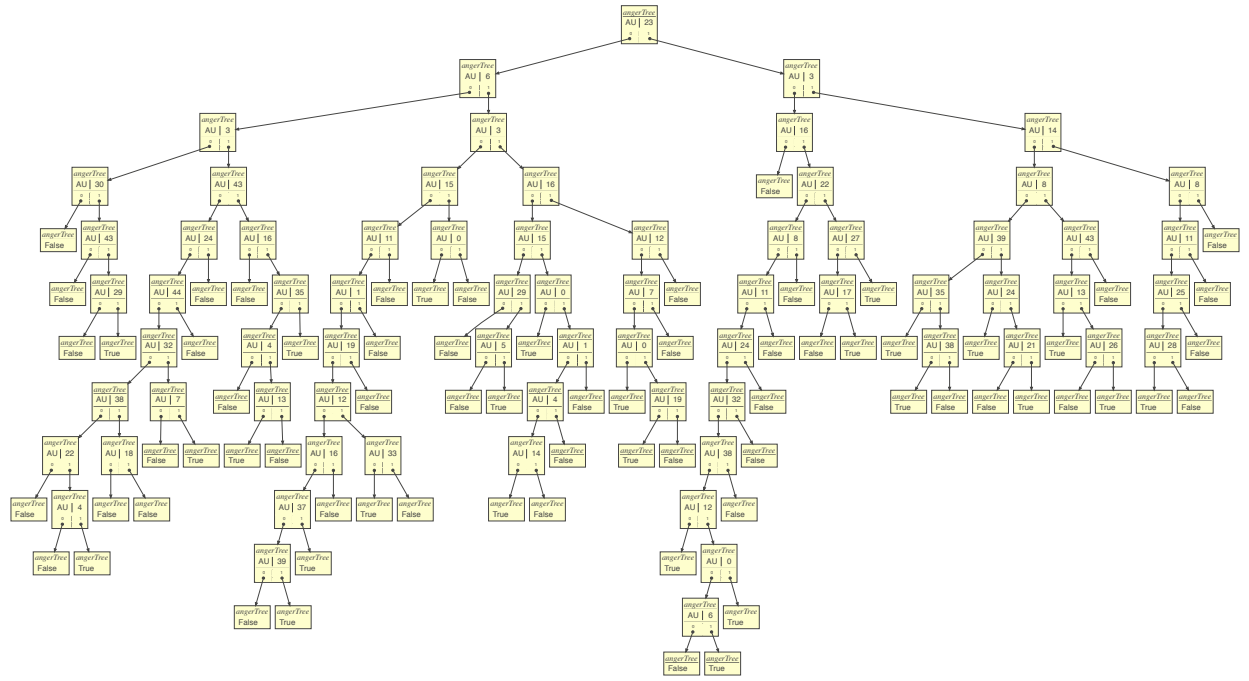


Figure 3: Anger Tree

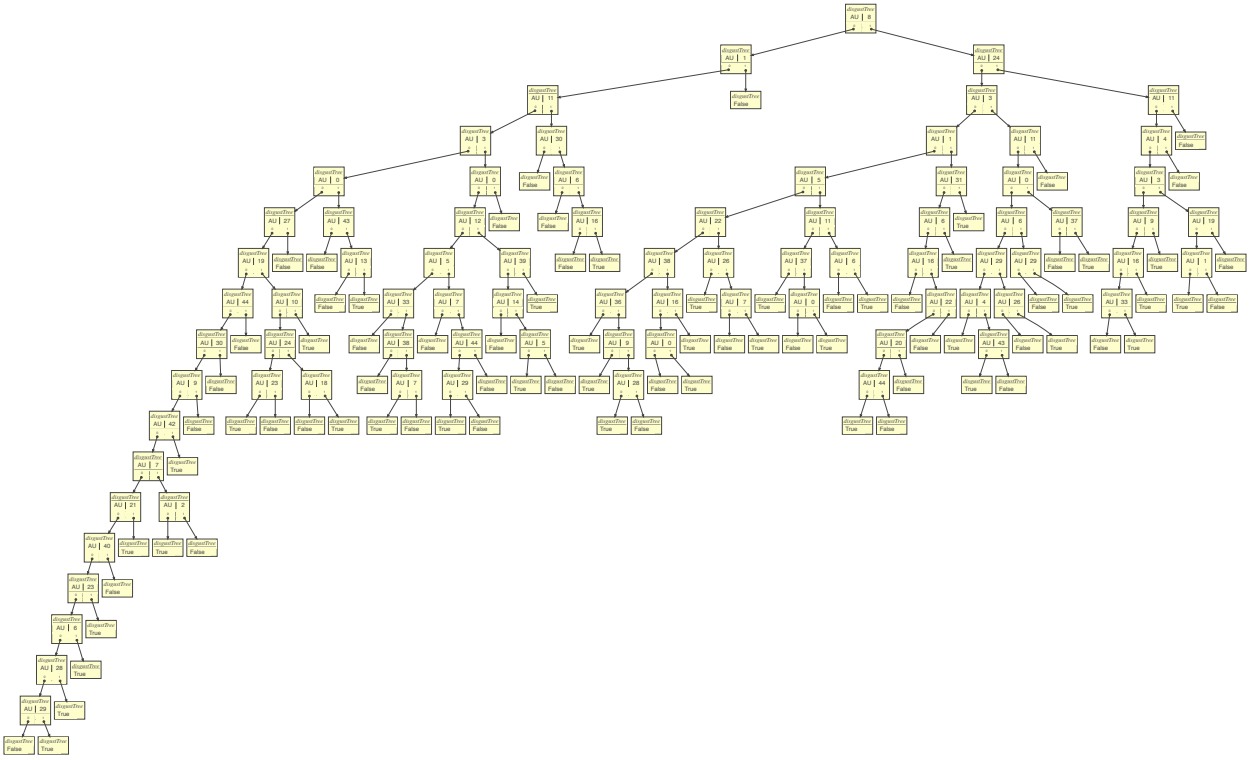


Figure 4: Disgust Tree

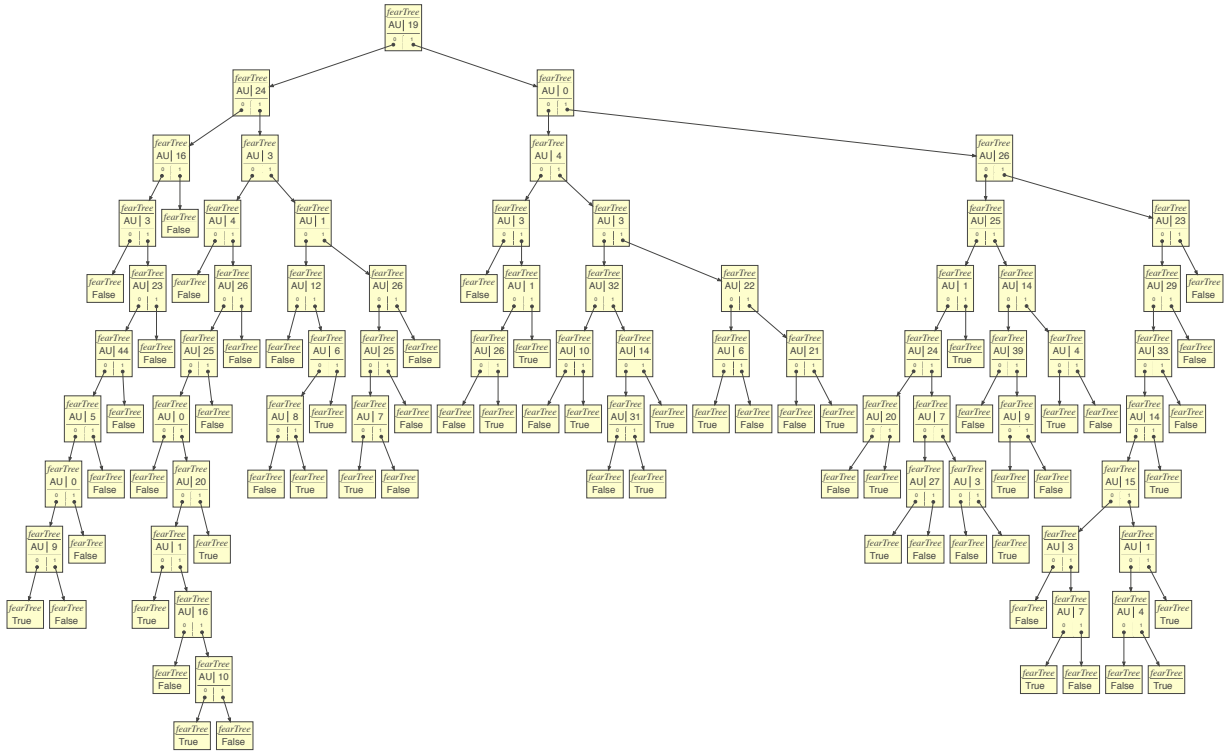


Figure 5: Fear Tree

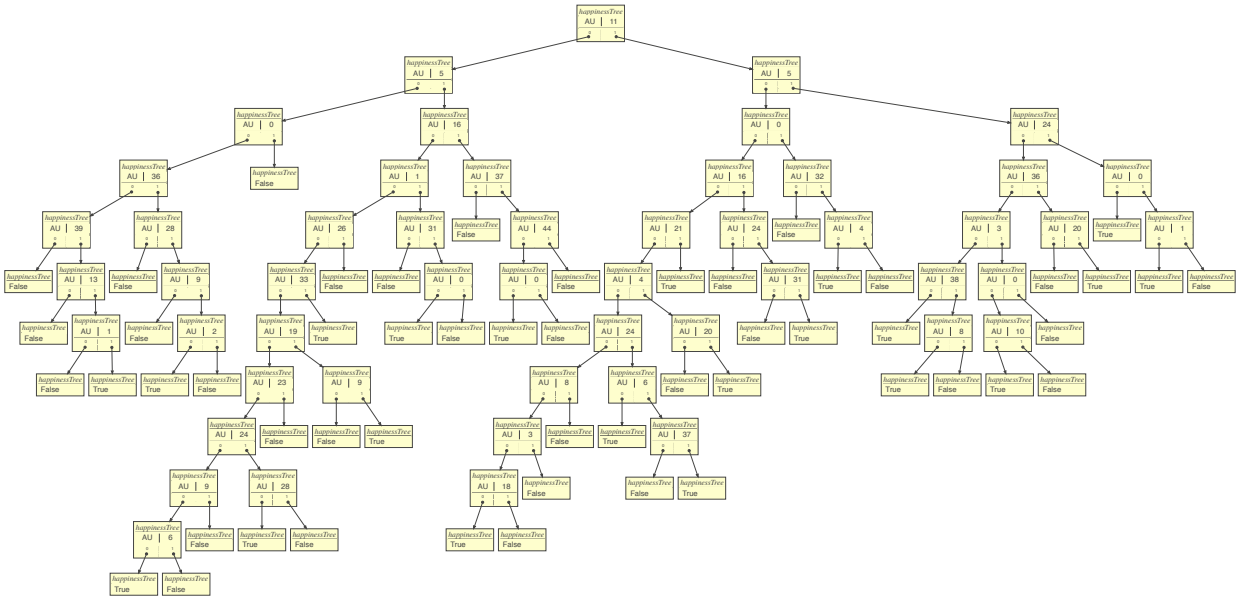


Figure 6: Happiness Tree

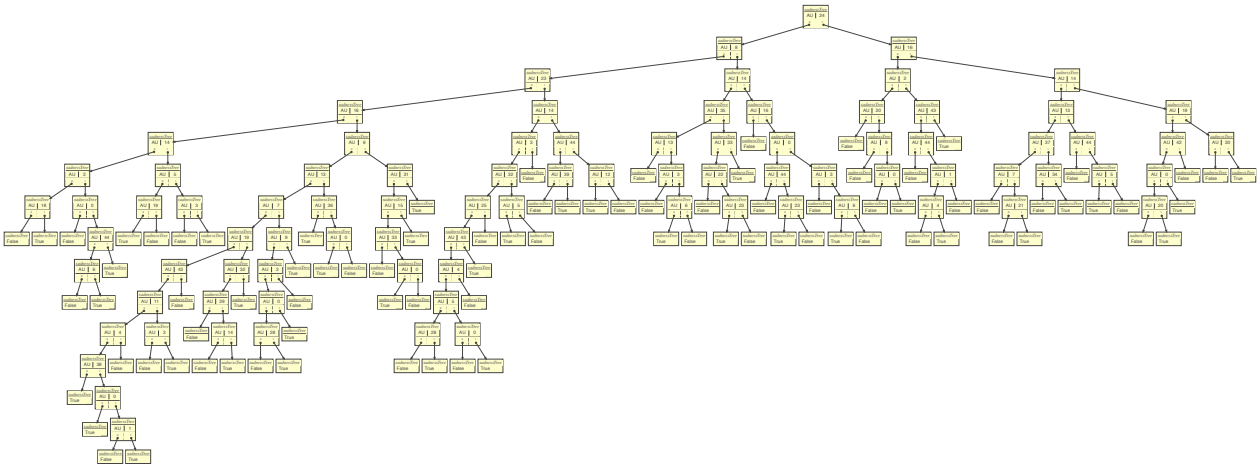


Figure 7: Sadness Tree

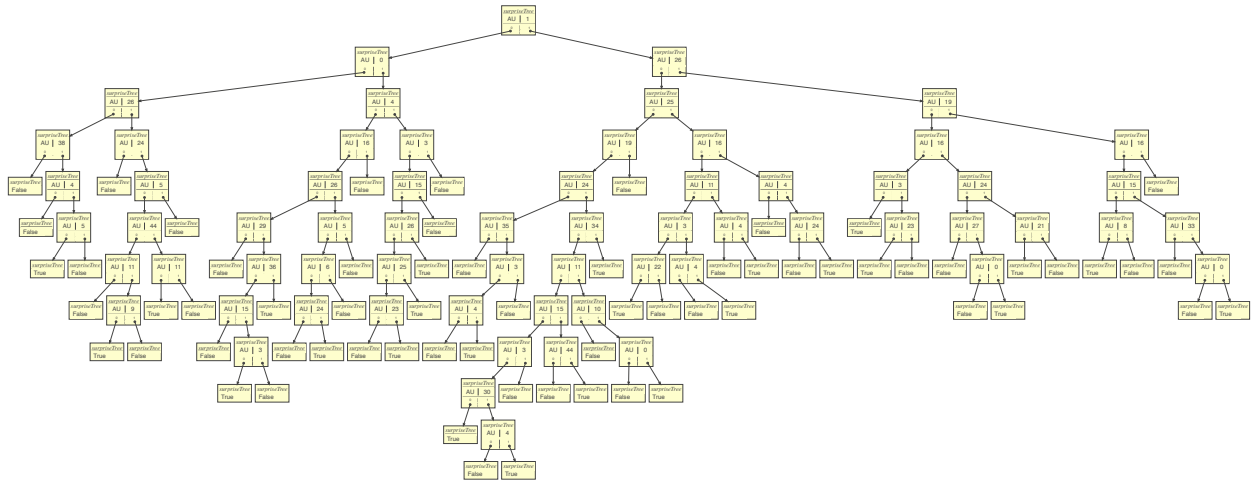


Figure 8: Surprise Tree