Imperial College London

Machine Learning

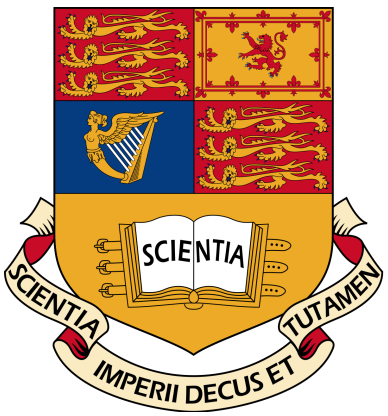
Group 18

Imperial College London

Department of Computing

# Decision Trees Coursework

*Authors:*
Adrian Catană (ac7815)
Andrei Isăilă (ii515)
Cristian Matache (crm15)
Oana Ciocioman (oc1115)




February 23, 2018

# Contents

# Chapter 1

# Implementation

## 1.1 Decision trees

The first solution for Emotion Recognition was using a standard algorithm for building decision trees. These structures would lately be used for providing an emotion choice on a certain set of attributes. As we will see later, this technique has the advantages of being easy to understand and easy to implement, but there are some variations of it and further improvements that would grow the overall performance.

### 1.1.1 Cross validation

The first important step of our implementation is the 10-fold cross-validation (even though our design allows any choice for splitting - by increasing the number of folds total accuracy grows, but this is expected since we are training our trees on a broader set of data and testing on less). The idea is to divide the data into 3 parts as follows: 80% training data, 10% validation data and the other 10% is left for testing purposes. The training algorithm behaves as described in the specification. After the training has completed, we test each tree on the validation data and based on how precise each tree is we give it a priority value. The precision of each tree is determined by calculating the error rate i.e. the quotient of number of test examples classified incorrectly and the total number of examples. The tree's priority will be used in cases of ambiguity later in the classification process. After the validation phase ended we simply test our classifier on the remaining data.

### 1.1.2 Training

In the training phase of each fold we are constructing 6 different trees for each emotion: anger, disgust, fear, happiness, sadness and surprise. The building process is done using the ID3 algorithm (see T. Mitchell, Machine Learning, page 56). As a result, each tree has as node labels a particular attribute and each node has 2 children, one for each boolean possible value of this specific attribute.

Note that when the algorithm is choosing the best attribute it follows the ideas based on information gain (e.g. using entropy as a measure of surprise), even though other measures such as the Gini impurity (which like entropy shows how heterogeneous/distributed some input is over a dataset) could have been used.

### 1.1.3 Prediction

Having the list of decision trees on each fold iteration, we must make predictions for the testing data that we split above. That is, for each list of attributes that we have in the testing dataset, we traverse our tree from the root to a leaf that will give us a boolean value, 0 or 1, depending on whether this certain set of attributes has been discovered to represent an emotion or not. The decision on whether we should go to a child or another is given by the boolean value of the specific attribute in the attributes set.

At this stage, we have a list of predictions obtained from the results computed from each tree, but we only need to choose one emotion as a final result. There are some cases presented in the Ambiguity section below which we will discuss one by one.

## 1.2 Random forests

### 1.2.1 Motivation

Being curious in improving the accuracy even further, we considered that the most natural way to continue was to use random decision forests. Since we already had implemented its building blocks - decision trees - we decided to experiment with this approach. The papers and resources we consulted reinforced our opinion. Also, forests are widely used in many practical situations so it was both a motivation to try them (from a learning perspective) and a confirmation that it will give better results.

### 1.2.2 How it works

The first thing to do in this case was to find the relation between the concept of **bootstrap aggregation** (Bagging) and our existing code base.

#### 1.2.2.1 What we already had - Trees

We were already building trees to binary classify every emotion. To predict, we were running all test samples against those 6 trees (one for each emotion), seeing what are the possible emotions and adopting a strategy to choose between them.

#### 1.2.2.2 What was next - Bagging

For random forests, we are using a number of N decision trees (instead of 1) per emotion, so $6 \cdot N$ trees in total, that can do binary classification on each emotion. Each of these trees are trained on a set of K randomly selected examples from the training set. This sampling is done uniformly and with replacement, so an example can be used twice to train the same tree.

The prediction phase is the point where we actually make extensive use of the ensemble technique of voting. That is, test examples are run against N trees for each emotion. Next, for every emotion some or all of those N trees will decide whether the test example is part of their class. Counting these positives is called voting. To give a final prediction we choose the most voted emotion. In case of equality, we go back to the strategies as we would do in plain decision trees.

#### 1.2.2.3 In code

The implementation followed closely the steps described above. We worked with certain features of Pandas library for sampling and slicing. As we used the existing implementation, we train the trees in the forest with all the attributes (Action Units). We found that we could either use random forests this way or do random sampling also on a number of P attributes much less than total number (= 45). The cross validation is done in the same fashion as for decision trees.

### 1.2.3 Observations

Indeed the accuracy was drastically increased from around 70% to over 83% and the model proved to be more robust than plain decision trees as this ensemble technique reduces variance. Also, understanding in detail every step of a more complex system was very beneficial for our own learning curve.

### 1.2.4 Parameter tuning

The higher the number of trees in the forest, the more accurate the prediction becomes. Obviously, there is a trade-off between robustness and accuracy on one hand and the expensiveness of the computation on the other hand. This problem can be reduced to finding reasonable values for parameters N and K. In our research we found that typical values for K are around 0.66 of the size of the training set. Out of pure curiosity we tried more values for K out of this range, but we concluded to remain in the recommended range. We used N = 5 and 10 (trees per emotion) for testing because it is relatively fast to compute and significantly higher robustness and accuracies can be observed. The final version is using K = 600 and N = 15 trees per emotion -> 180 trees in the whole forest altogether with a Min/Max strategy as it will be described later on.

## 1.3 Visualisation

Since trees are a complex data structure widely used throughout the project we decided to facilitate their visualisation. As the decision trees can grow quite large we opted for 2 types of display which:

- Depict the binary structure of the tree, but can have overlapping nodes

- Distinguish the contents of all nodes, so no overlapping nodes occur

Thus, both forms can be found in the appendix. In our assignment we made extensive use of Networkx library in order to create a better tree visualization. Please **note** that there is an issue with the latest version of Networkx. It was already raised on Github but have not been solved yet. Instead, please make sure to use version 1.11 which works well (this is actually the version that will be installed by running **sudo pip3 install -r requirements.txt**).

## 1.4 Parallel Computation

The idea behind parallelizing our algorithms' work was that it takes a long time to build all the trees/forests and then to traverse them, while these operations are quite independent. An improvement in overall computational time would help us during parameter tuning and testing.

During decision trees construction, we make a separate process for each emotion. This lowered the overall execution time on our testing system from 298.37 seconds to 168.12 (the algorithms were tested on the clean/noisy datasets). The same principle applies for the decision forest algorithm, yielding a progress from 878.44 seconds to 434.50.

An even more extreme approach that we tried was concerning tree traversals. Once we had all the trees for one emotion, we could test them asynchronously against some fixed datasets, boosting up the overall speed, especially for large tree structures. This proved to be unscalable: by creating so many processes we would have risked to lower down the overall performance due to scheduler/resource allocation issues.

However, a massive enhancement for the future would be to run subsets of processes on different machines in a map-reduce fashion.

## 1.5 Command line arguments

To handle these multiple combinations of testing our model(s), we enabled the specification of command line arguments when running. For example: python(3) main.py tree multi - runs the decision tree on multiple processes. More information can be found in the README.md file.

# Chapter 2

# Evaluation

## 2.1 Decision Trees

### 2.1.1 Confusion Matrix

The tables below represent the confusion matrices computed based on the values obtained by performing the pure decision trees algorithm both on clean and noisy data. We can notice that the values on the principal diagonal are higher than the other ones in both tables. Since the accuracy of the implemented algorithm was average we optimized the algorithm by using Random Forests. The data in the tables below is row-normalized and it is based on the matrix computed by elementwise averages of the 10 confusion matrices obtained at each step of the 10-fold cross validation.

#### 2.1.1.1 Clean Data

The accuracy of one decision tree on **clean** data, with random strategy is: **69.4%**

| Predicted / Expected | Anger | Disgust | Fear | Happiness | Sadness | Surprise |
|---|---|---|---|---|---|---|
| **Anger** | 0.597015 | 0.134328 | 0.059701 | 0.052239 | 0.111940 | 0.044776 |
| **Disgust** | 0.097938 | 0.695876 | 0.036082 | 0.056701 | 0.067010 | 0.046392 |
| **Fear** | 0.051724 | 0.060345 | 0.689655 | 0.051724 | 0.034483 | 0.112069 |
| **Happiness** | 0.041475 | 0.059908 | 0.009217 | 0.811060 | 0.050691 | 0.027650 |
| **Sadness** | 0.115942 | 0.115942 | 0.043478 | 0.072464 | 0.579710 | 0.072464 |
| **Surprise** | 0.009756 | 0.043902 | 0.078049 | 0.029268 | 0.043902 | 0.795122 |

Table 1. Confusion matrix for clean data

#### 2.1.1.2 Noisy Data

The accuracy of one decision tree on **noisy** data, with random strategy is: **62.6%**

| Predicted / Expected | Anger | Disgust | Fear | Happiness | Sadness | Surprise |
|---|---|---|---|---|---|---|
| **Anger** | 0.311927 | 0.155963 | 0.165138 | 0.064220 | 0.220183 | 0.082569 |
| **Disgust** | 0.036364 | 0.745455 | 0.090909 | 0.048485 | 0.054545 | 0.024242 |
| **Fear** | 0.107345 | 0.124294 | 0.548023 | 0.062147 | 0.062147 | 0.096045 |
| **Happiness** | 0.036866 | 0.046083 | 0.092166 | 0.728111 | 0.032258 | 0.064516 |
| **Sadness** | 0.153846 | 0.096154 | 0.115385 | 0.048077 | 0.480769 | 0.105769 |
| **Surprise** | 0.021834 | 0.021834 | 0.109170 | 0.087336 | 0.039301 | 0.720524 |

Table 2. Confusion matrix for noisy data

#### 2.1.1.3 Observations

We can notice from both confusion matrices that the algorithm performs quite well, giving an accuracy of 69.4 % when using the clean data and an accuracy of 62.6 % when using the noisy data. Furthermore, the emotions that are recognized in most cases are **Happiness** and **Surprise**. On the other hand, **Anger** and **Sadness** lower the accuracy of our decision trees algorithm since the emotions can sometimes be mistaken in the real life too. From the first line of the confusion matrix for noisy data we can observe that Anger and Sadness were confused in most of the times.

### 2.1.2 Measurements

*LEGEND:*
*CR - Classification Rate, F1 - Performance of the Classifier UAR - Unweighted Average Recall*

#### 2.1.2.1 Clean Data

The average **Classification Rate** is: **89.8%**

The average **Precision** is: **69.4%**

| Metric / Emotion | CR | Recall class 1 | Precision class 1 | F1 class 1 | UAR | Recall class 2 | Precision class 2 | F1 class 2 |
|---|---|---|---|---|---|---|---|---|
| **Anger** | 0.88002 | 0.59701 | 0.65329 | 0.62388 | 0.76682 | 0.93663 | 0.92076 | 0.92863 |
| **Disgust** | 0.88024 | 0.69587 | 0.62674 | 0.65950 | 0.80649 | 0.91711 | 0.93780 | 0.92734 |
| **Fear** | 0.91052 | 0.68965 | 0.75274 | 0.71982 | 0.82217 | 0.95469 | 0.93895 | 0.94675 |
| **Happiness** | 0.92477 | 0.81105 | 0.75555 | 0.78232 | 0.87929 | 0.94752 | 0.96164 | 0.95453 |
| **Sadness** | 0.87861 | 0.57971 | 0.65302 | 0.61418 | 0.75905 | 0.93839 | 0.91778 | 0.92797 |
| **Surprise** | 0.91529 | 0.79512 | 0.72384 | 0.75781 | 0.86722 | 0.93932 | 0.95820 | 0.94867 |

Table 3. Measurements for decision trees algorithm performed on clean data

#### 2.1.2.2 Noisy Data

The average **Classification Rate** is: **86.3%**

The average **Precision** is: **58.0%**

| Metric / Emotion | CR | Recall class 1 | Precision class 1 | F1 class 1 | UAR | Recall class 2 | Precision class 2 | F1 class 2 |
|---|---|---|---|---|---|---|---|---|
| **Anger** | 0.82594 | 0.31192 | 0.46682 | 0.37397 | 0.62033 | 0.92874 | 0.87094 | 0.89892 |
| **Disgust** | 0.88352 | 0.74545 | 0.62654 | 0.68084 | 0.82829 | 0.91113 | 0.94708 | 0.92876 |
| **Fear** | 0.82920 | 0.54802 | 0.48896 | 0.51680 | 0.71673 | 0.88544 | 0.90736 | 0.89627 |
| **Happiness** | 0.90297 | 0.72811 | 0.70120 | 0.71440 | 0.83302 | 0.93794 | 0.94520 | 0.94156 |
| **Sadness** | 0.84538 | 0.48076 | 0.54067 | 0.50896 | 0.69954 | 0.91831 | 0.89840 | 0.90824 |
| **Surprise** | 0.89123 | 0.72052 | 0.65881 | 0.68828 | 0.82294 | 0.92537 | 0.94303 | 0.93412 |

Table 4. Measurements for decision trees algorithm performed on noisy data

#### 2.1.2.3 Observations

The metrics above indicate a big difference in terms of *Precision* (more than 11 %) when running the algorithm on clean/noisy data. The average *Classification Rate* alone is not a very reliable characteristic for the correctness and accuracy of our algorithm since we can easily notice the discrepancy between average precision and average classification rate.

## 2.2 Random Forests

As a consequence of the observations noted before, we optimized the *Decision Trees Algorithm* by using *Random Forests* and *Min-Max Depth* in order to maximize the average precision and recall.

### 2.2.1 Confusion Matrix

#### 2.2.1.1 Clean Data

The accuracy of five decision trees per emotion on **clean** data, with random strategy is: **81.7%**

| Predicted / Expected | Anger | Disgust | Fear | Happiness | Sadness | Surprise |
|---|---|---|---|---|---|---|
| **Anger** | 0.757576 | 0.075758 | 0.045455 | 0.007576 | 0.098485 | 0.015152 |
| **Disgust** | 0.061321 | 0.797170 | 0.009434 | 0.037736 | 0.084906 | 0.009434 |
| **Fear** | 0.036036 | 0.009009 | 0.810811 | 0.009009 | 0.045045 | 0.090090 |
| **Happiness** | 0.009302 | 0.023256 | 0.009302 | 0.930233 | 0.023256 | 0.004651 |
| **Sadness** | 0.083333 | 0.108333 | 0.033333 | 0.025000 | 0.725000 | 0.025000 |
| **Surprise** | 0.014019 | 0.000000 | 0.070093 | 0.014019 | 0.018692 | 0.883178 |

Table 5. Confusion matrix for clean data

#### 2.2.1.2 Noisy Data

The accuracy of five decision trees per emotion on **clean** data, with random strategy is: **74.4%**

| Predicted / Expected | Anger | Disgust | Fear | Happiness | Sadness | Surprise |
|---|---|---|---|---|---|---|
| **Anger** | 0.506849 | 0.123288 | 0.068493 | 0.068493 | 0.164384 | 0.068493 |
| **Disgust** | 0.076531 | 0.770408 | 0.056122 | 0.045918 | 0.030612 | 0.020408 |
| **Fear** | 0.065000 | 0.080000 | 0.685000 | 0.040000 | 0.070000 | 0.060000 |
| **Happiness** | 0.041860 | 0.027907 | 0.074419 | 0.800000 | 0.023256 | 0.032558 |
| **Sadness** | 0.106383 | 0.031915 | 0.063830 | 0.053191 | 0.670213 | 0.074468 |
| **Surprise** | 0.017937 | 0.008969 | 0.053812 | 0.044843 | 0.044843 | 0.829596 |

Table 6. Confusion matrix for noisy data

#### 2.2.1.3 Observations

The most significant observation regarding the confusion matrices computed for the *Random Forests Algorithm* is the increase in the accuracy of our emotion recognition on both clean and noisy data (12.3 % for clean data, 11.8 % for noisy data). Similarly, **Happiness** and **Surprise** were detected with the highest accuracy, while **Anger** and **Sadness** were sometimes mistaken.

### 2.2.2 Measurements

#### 2.2.2.1 Clean Data

The average **Classification Rate** is: **93.9%**

The average **Precision** is: **81.6%**

| Metric / Emotion | CR | Recall class 1 | Precision class 1 | F1 class 1 | UAR | Recall class 2 | Precision class 2 | F1 class 2 |
|---|---|---|---|---|---|---|---|---|
| **Anger** | 0.92559 | 0.75757 | 0.78783 | 0.77241 | 0.85838 | 0.95919 | 0.95188 | 0.95552 |
| **Disgust** | 0.93013 | 0.79716 | 0.78653 | 0.79181 | 0.87694 | 0.95672 | 0.95932 | 0.95802 |
| **Fear** | 0.94053 | 0.81081 | 0.82868 | 0.81965 | 0.88864 | 0.96647 | 0.96232 | 0.96439 |
| **Happiness** | 0.97281 | 0.93023 | 0.90881 | 0.91939 | 0.95578 | 0.98133 | 0.98598 | 0.98365 |
| **Sadness** | 0.90910 | 0.72500 | 0.72836 | 0.72667 | 0.83546 | 0.94592 | 0.94505 | 0.94548 |
| **Surprise** | 0.95647 | 0.88317 | 0.85953 | 0.87119 | 0.92715 | 0.97113 | 0.97650 | 0.97381 |

Table 7. Measurements for random forests algorithm performed on clean data

#### 2.2.2.2 Noisy Data

The average **Classification Rate** is: **90.3%**

The average **Precision** is: **70.6%**

| Metric / Emotion | CR | Recall class 1 | Precision class 1 | F1 class 1 | UAR | Recall class 2 | Precision class 2 | F1 class 2 |
|---|---|---|---|---|---|---|---|---|
| **Anger** | 0.86652 | 0.50684 | 0.62223 | 0.55864 | 0.72265 | 0.93845 | 0.90489 | 0.92137 |
| **Disgust** | 0.91638 | 0.77040 | 0.73901 | 0.75438 | 0.85799 | 0.94558 | 0.95368 | 0.94961 |
| **Fear** | 0.89472 | 0.68499 | 0.68385 | 0.68442 | 0.81083 | 0.93666 | 0.93697 | 0.93682 |
| **Happiness** | 0.92459 | 0.80000 | 0.76013 | 0.77955 | 0.87475 | 0.94951 | 0.95957 | 0.95451 |
| **Sadness** | 0.88951 | 0.67021 | 0.66800 | 0.66910 | 0.80179 | 0.93338 | 0.93399 | 0.93368 |
| **Surprise** | 0.92894 | 0.82959 | 0.76423 | 0.79557 | 0.88920 | 0.94881 | 0.96532 | 0.95699 |

Table 8. Measurements for random forests algorithm performed on noisy data

#### 2.2.2.3 Observations

From the measurements above we can observe a remarkable improvement of the average precision and recall. *Random Forests Algorithm* is performing even better on noisy data than on clean data when using *Decision Trees Algorithm*. The metrics values tend to be higher than the ones obtained before which implies a more accurate emotion recognition.

# Chapter 3

# Questions

## 3.1 Noisy-Clean Datasets

From the information collected in the tables in Chapter 2 we can distinguish significant differences in performance when the algorithm is run on clean or noisy data. An overall better performance can be noticed on clean data rather than on noisy data. The reason behind this comes from the difficulty in characterizing different types of emotion. Since sadness and anger can be confused by humans as well, training a machine to recognize this emotions can be even harder.

First of all, both the decision trees and random forests algorithms applied on the clean data give an accuracy with about 7 % higher than the one obtained on the noisy data. Furthermore, there is a difference of 11 % in the average precision obtained on clean/noisy data. Secondly, the emotions mostly recognized were Happiness and Surprise on both datasets, while Anger and Sadness were often confused.

## 3.2 Ambiguity

Let the list of predictions from the prediction section (**1.1.2**) above be P. We face several cases:

- **P is empty** - no tree accepted the current example as a valid emotion.
- **P has one element** - only 1 tree accepted the example as a valid emotion.
- **P has at least 2 elements** - more than one tree accepted the example as a valid emotion.

The case where a single tree recognized the example as being an emotion is clear and requires no further explanation, we can safely classify it as being that particular emotion. For the other 2 we can come up with various heuristic to improve the classification process.

#### 3.2.0.1 Random

- **P is empty** - take a random emotion from all 6 available.
- **P has at least 2 elements** - take a random emotion from the ones accepted.

This approach resulted in an accuracy of **67.3%**.

#### 3.2.0.2 Min/Max Depth and Precision

This is where the precision parameter calculated on the validation data is used. Each Tree has a precision associated, which acts as a priority value.

- **P is empty**
  - **First criterion:** Choose the emotion whose tree rejected the test data at the highest depth. Our rationale is that if a tree needs more attributes in order to reject the example, then it is more likely that the example can in fact be accepted.
  - **Second criterion:** If more trees rejected the test data at the same depth, choose the tree which has the lowest precision as it is more likely to be wrong.
- **P has at least 2 elements**

- **First criterion:** Choose the emotion whose tree accepted the test data at the lowest depth. The intuition behind this is that we want to choose the more general emotion i.e the emotion which needs less attributes in order to be classified. It is more likely that an emotion which takes more attributes in order to be validated is a more specific case which might only be present in our training data and is not something that is generally true.
- **Second criterion:** If more trees accepted the test data at the same depth, choose the tree which has the highest precision as it is more likely to be right.

#### 3.2.0.3 Conclusions

Generally, Min/Max Depth and Precision strategy yields higher performances than the Random one when used with Decision Trees as well as when used with Random Forests. For example, in case of decision trees the **Random** strategy's accuracy on clean data was about 69% while Min/Max approach resulted in an improved accuracy of 75%. The enhancement on random forests when the number of trees is large is much less impactful, but if we run a forest with N=10 trees per emotion (so 60 in total) we managed to raise the accuracy from 83.8% in Random to 84.2% in Min/Max. However, there can be a lot of influnce by chance so this is not a very precise way of measuring the overall performance.

## 3.3 Pruning

Generally, pruning is a method widely used in AI to manage the trade-off between speed and accuracy. Pruning hastens the computation without impacting the final classification rate too much. That is, whenever a search problem over a tree occurs, pruning removes branches that would not bring an advantage (in terms of CR) if they were to be explored. As the final tree is simpler, it also reduces overfitting.

### 3.3.1 How it works

The function builds a classification tree and then computes the cost of the tree and smaller trees using both a 10-fold cross-validation method and a resubstitution one (this will use the entire data for both training and testing). These test functions will give us the expected best level where the pruning can happen, the number of terminal nodes for each subtree, together with the cost/error of the tree.

The costs/errors are then plot against the number of nodes in the pruned subtrees (cost per tree size). The blue line represents the error using 10-fold cross-validation, while the red one is for resubstitution - this measure under-estimates the true error.

We follow the ideas from "T. Mitchell - Machine Learning, section 3.7. Issues in Decision tree learning": the red line "grows each branch of the tree just deeply enough to perfectly classify the training examples. [...] in fact it can lead to difficulties when there is noise in the data, or when the number of training examples is too small to produce a representative sample of the true target function."

That is, the red line is trained over the entire dataset; testing against a subset of these data will always give less errors than the blue line, since it was not trained on the testing examples. The best coordinate will be the one that has a variance no greater than one standard error above the minimum value along the red line.

### 3.3.2 Noisy-Clean comparison

The difference between the two graphs is as expected: in the noisy case, the area between the blue and red lines is increasing with the number of nodes more than in the clean case. This is due to the fact that noisy data is more likely to introduce high variance. In short, the main point is that the wider the gap between the two lines the more the overfitting.

### 3.3.3 Optimal tree size

For our plots, such coordinates are marked using squares (this is actually just plotting nodes(bestLevel + 1), as expected). For clean data, the size is 51 for cross-validation and 198 for resubstitution (this can also be seen by printing nodes(bestLevel + 1) and nodes2(bestLevel2 + 1)), while for noisy data we get 22 for blue and 274 for red.

# Appendix A

# Appendix

## A.1 Decision Trees Visualization

### A.1.1 Anger

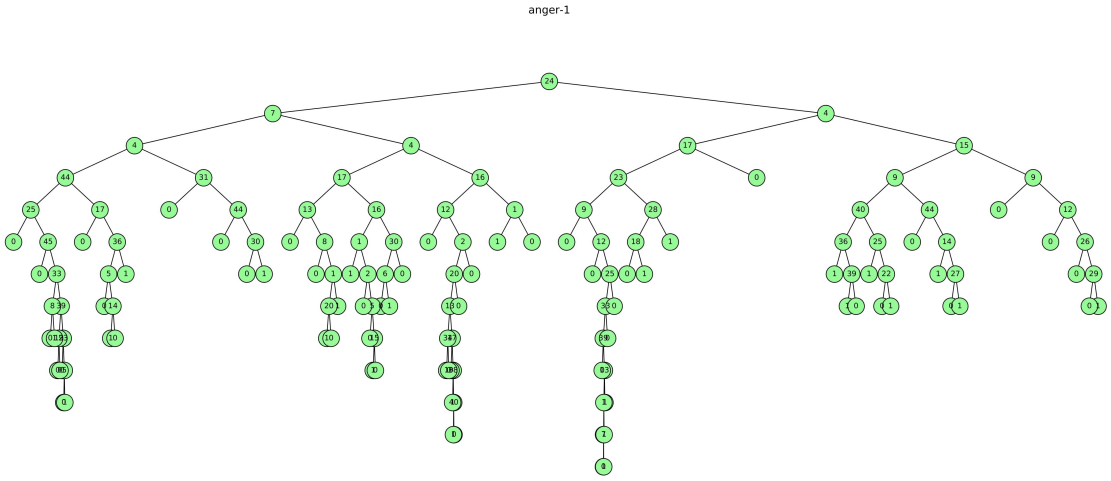



Figure A.1: Decision Tree Type 1 for Anger

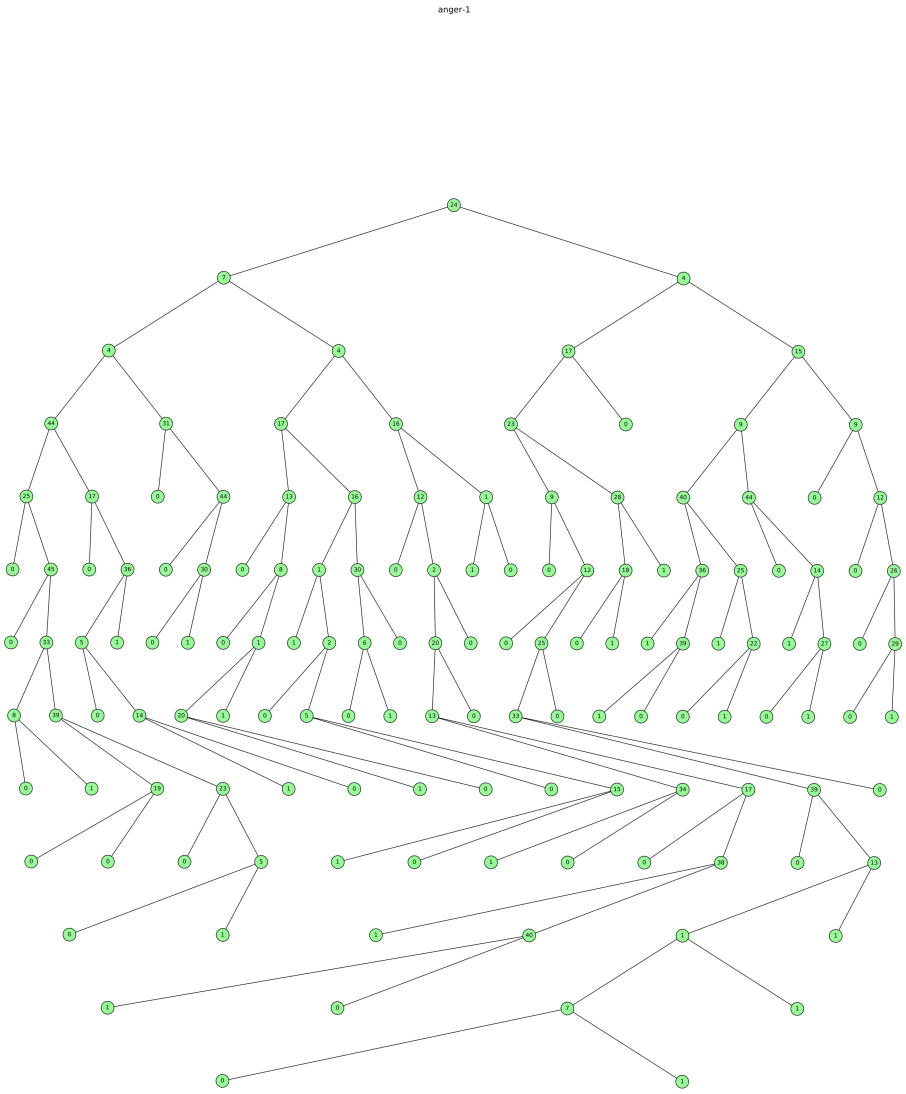



Figure A.2: Decision Tree Type 2 for Anger

### A.1.2 Disgust

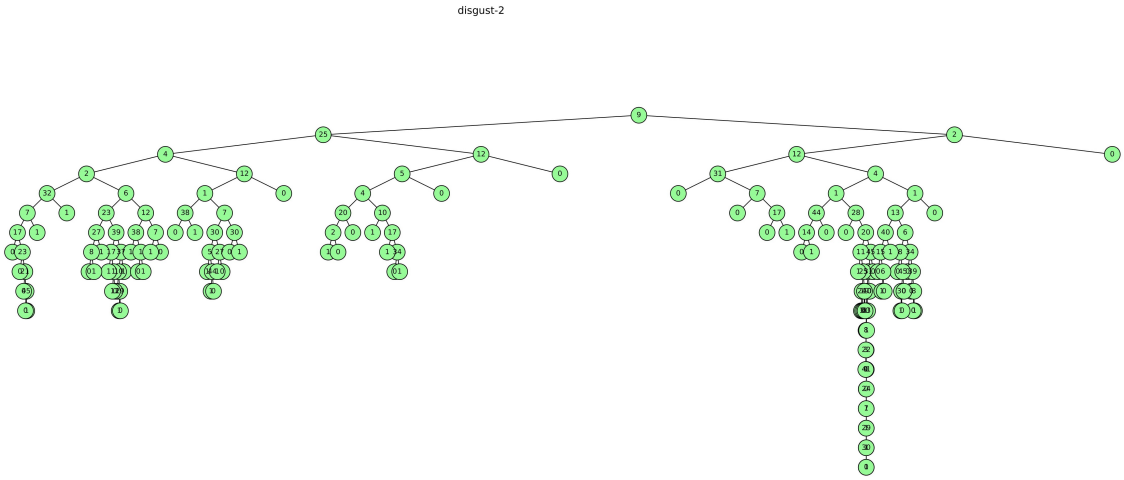



Figure A.3: Decision Tree Type 1 for Disgust

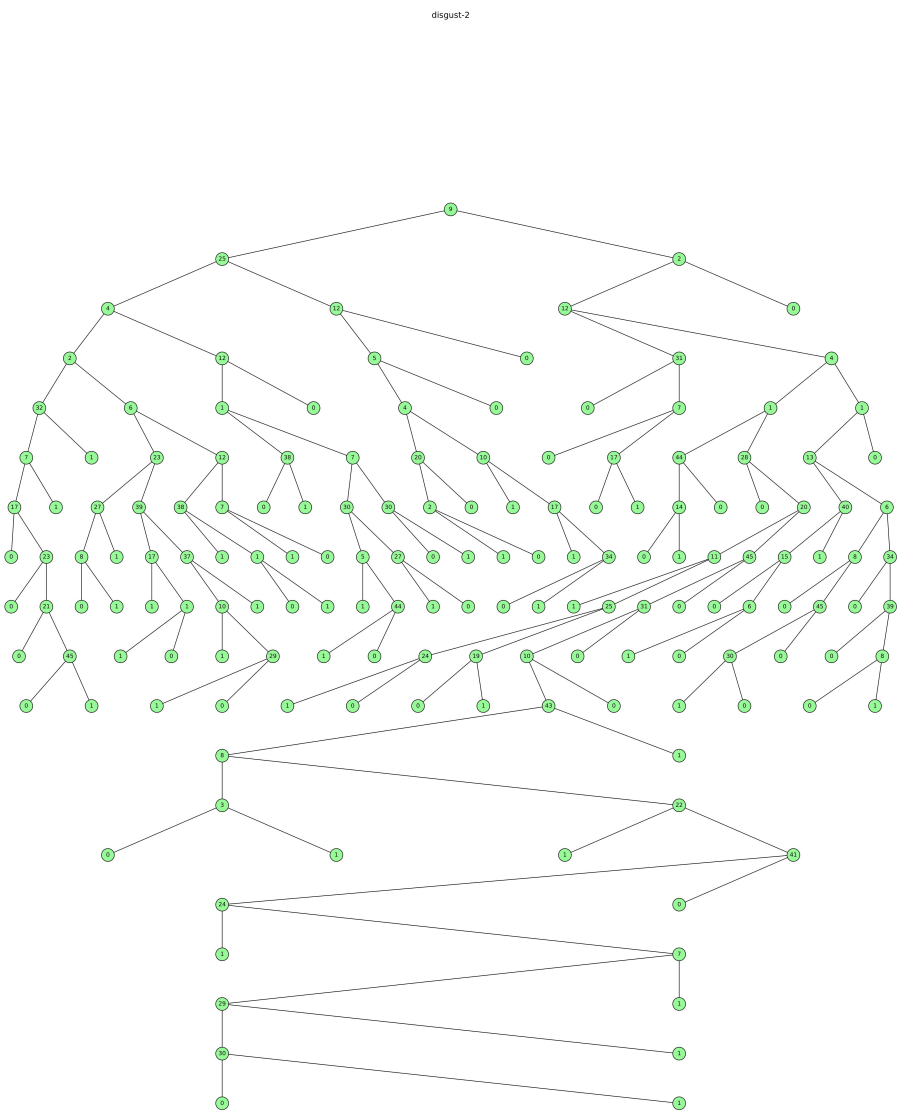



Figure A.4: Decision Tree Type 2 for Disgust

### A.1.3 Fear

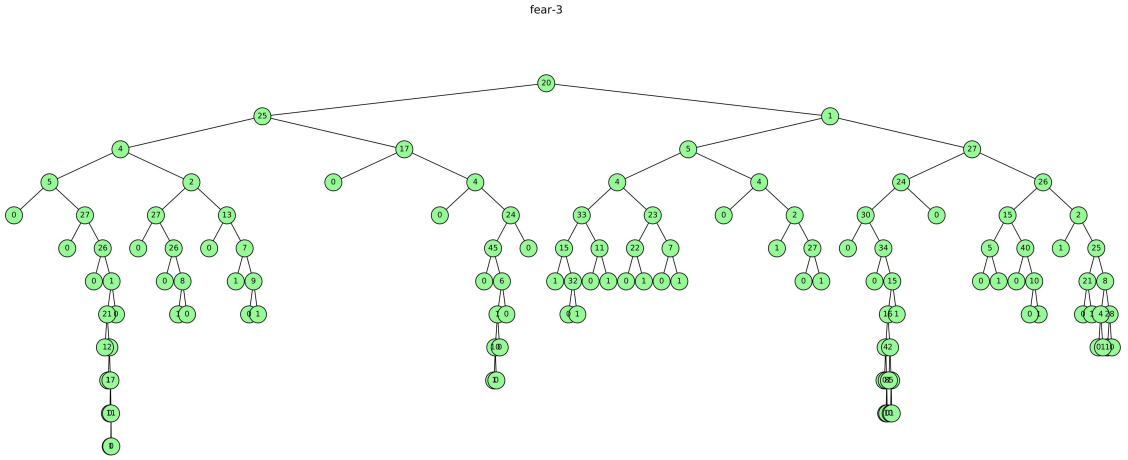



Figure A.5: Decision Tree Type 1 for Fear

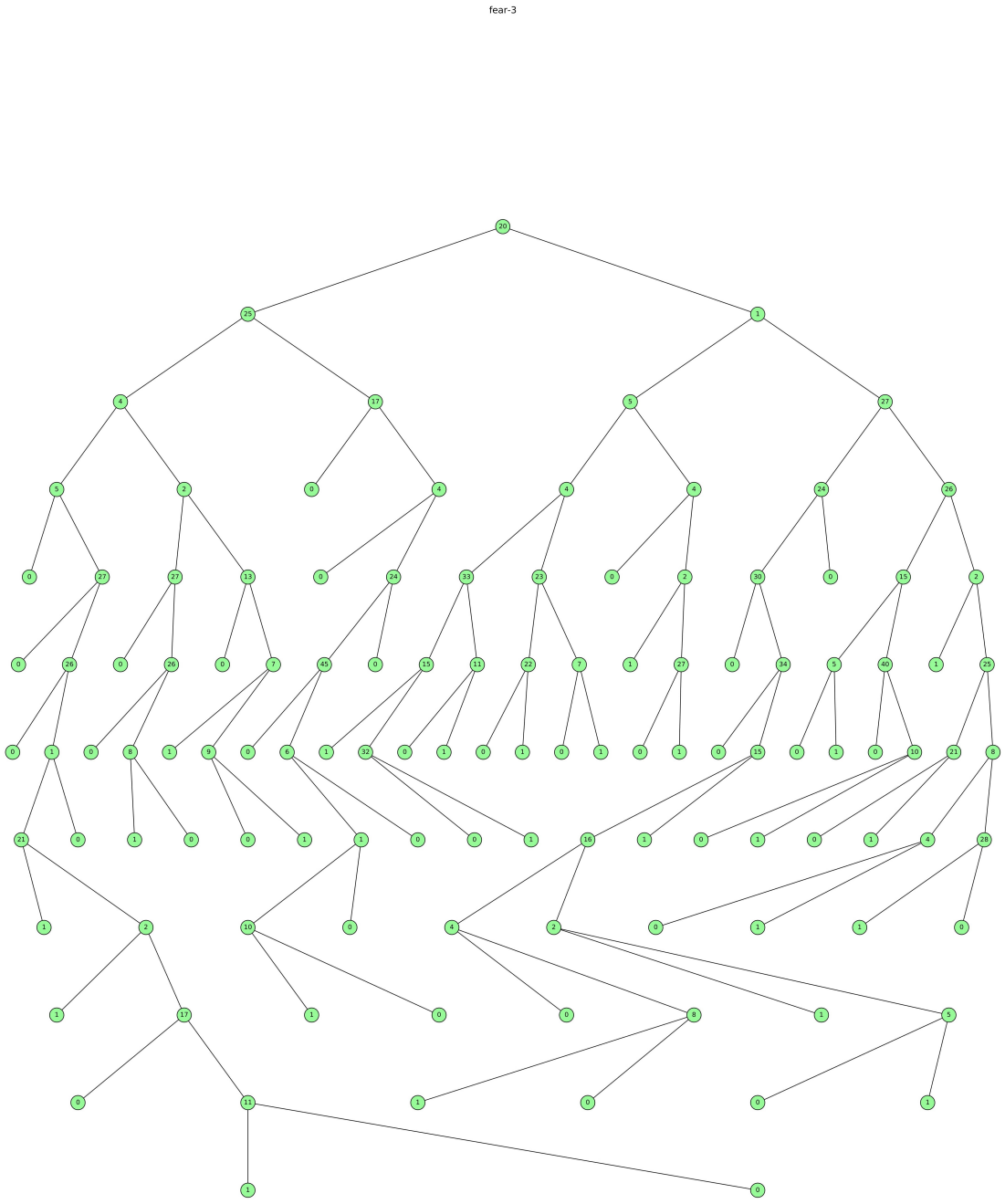



Figure A.6: Decision Tree Type 2 for Fear

### A.1.4 Happiness

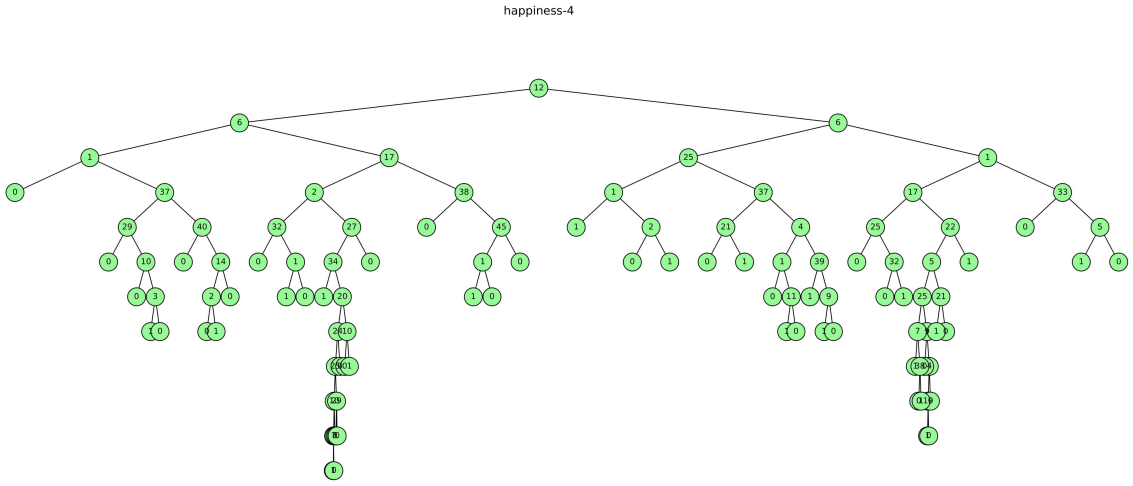



Figure A.7: Decision Tree Type 1 for Happiness

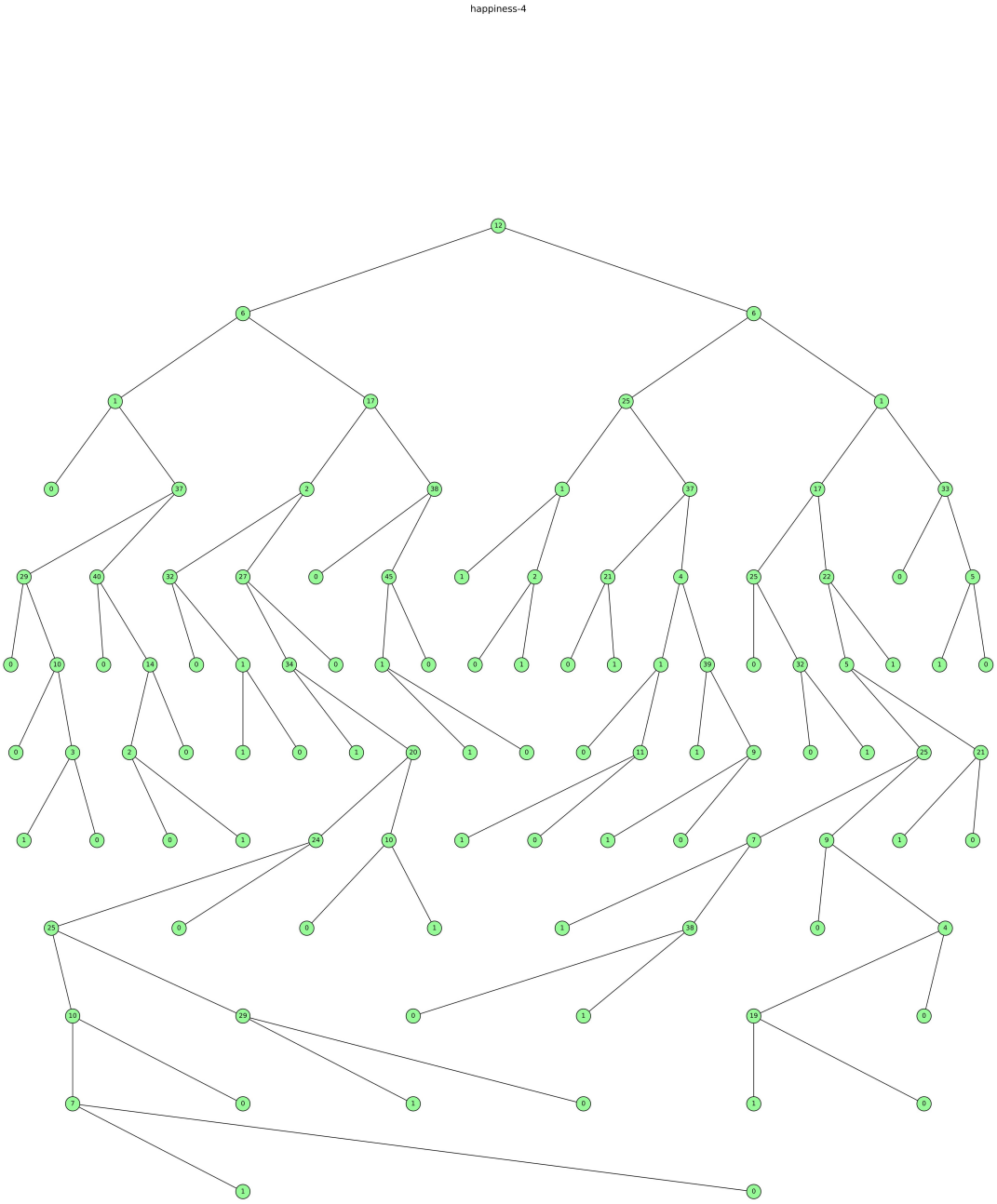



Figure A.8: Decision Tree Type 2 for Happiness

### A.1.5 Sadness

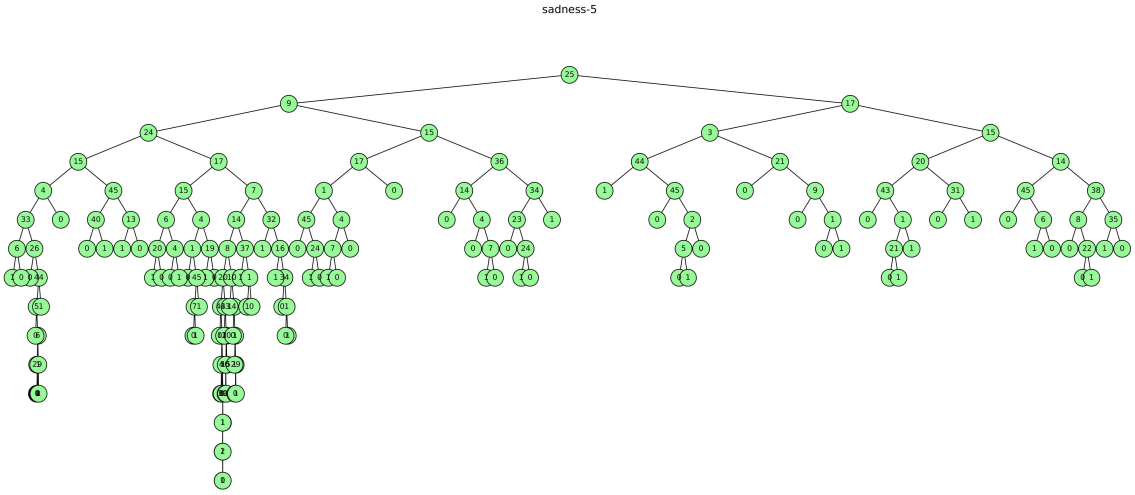



Figure A.9: Decision Tree Type 1 for Sadness

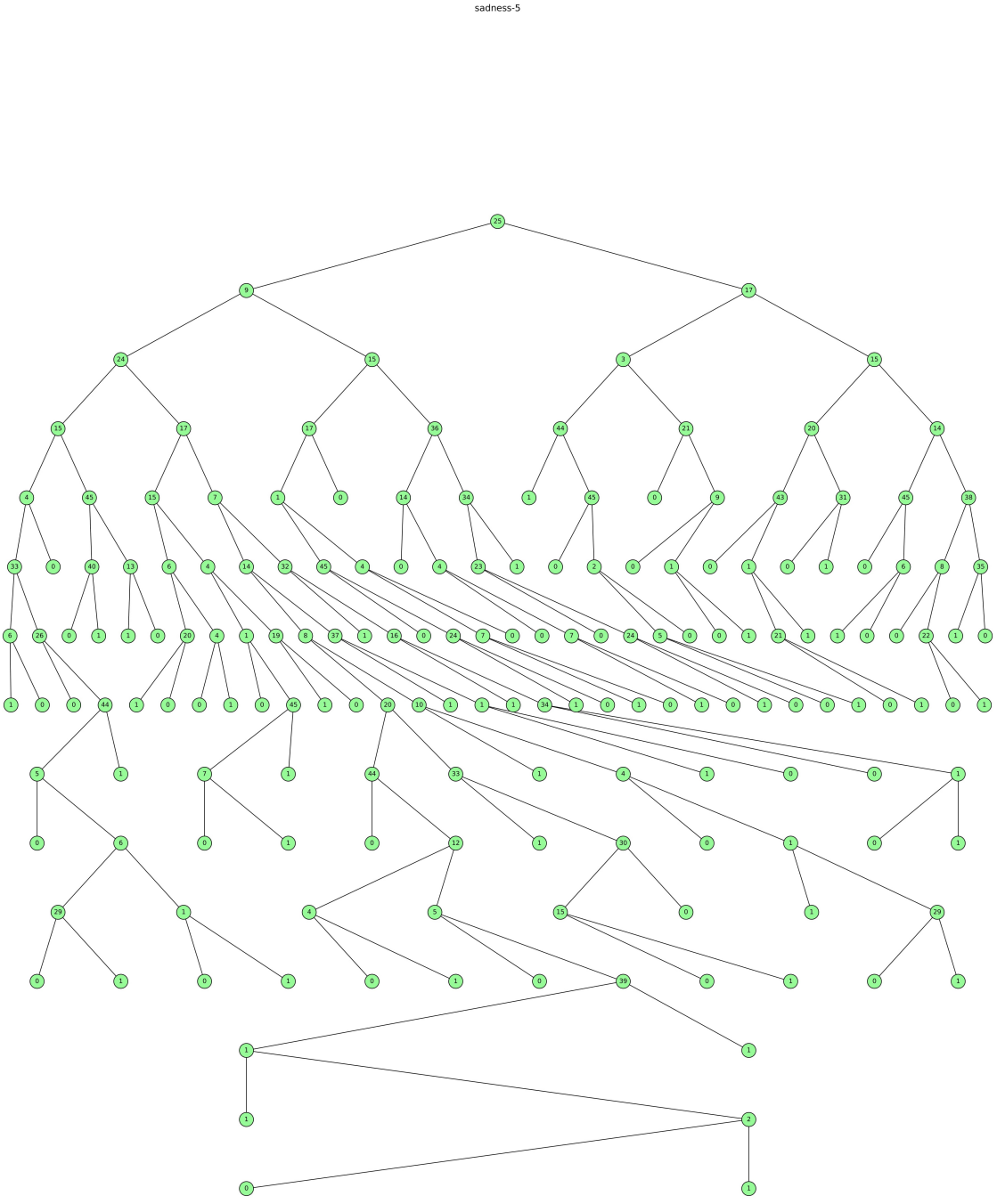



Figure A.10: Decision Tree Type 2 for Sadness

### A.1.6 Surprise

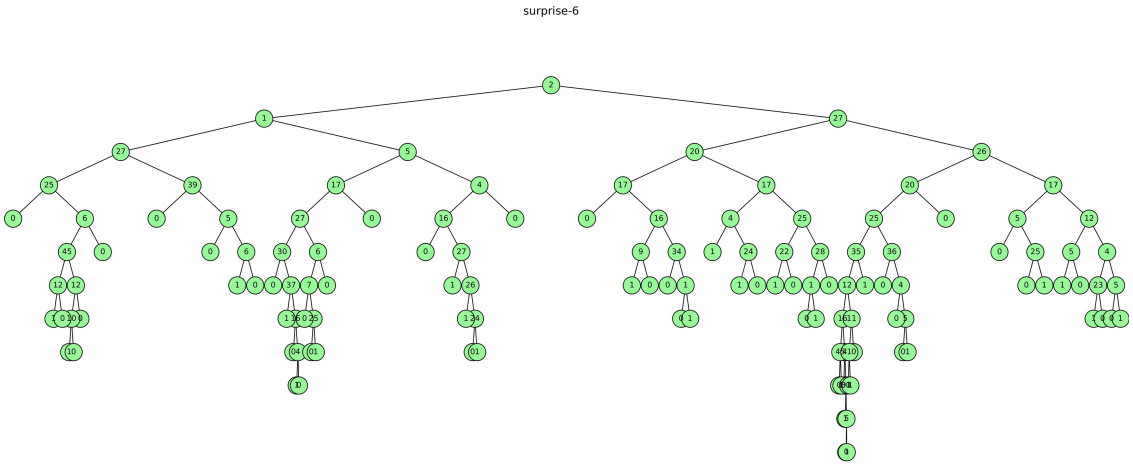



Figure A.11: Decision Tree Type 1 for Surprise

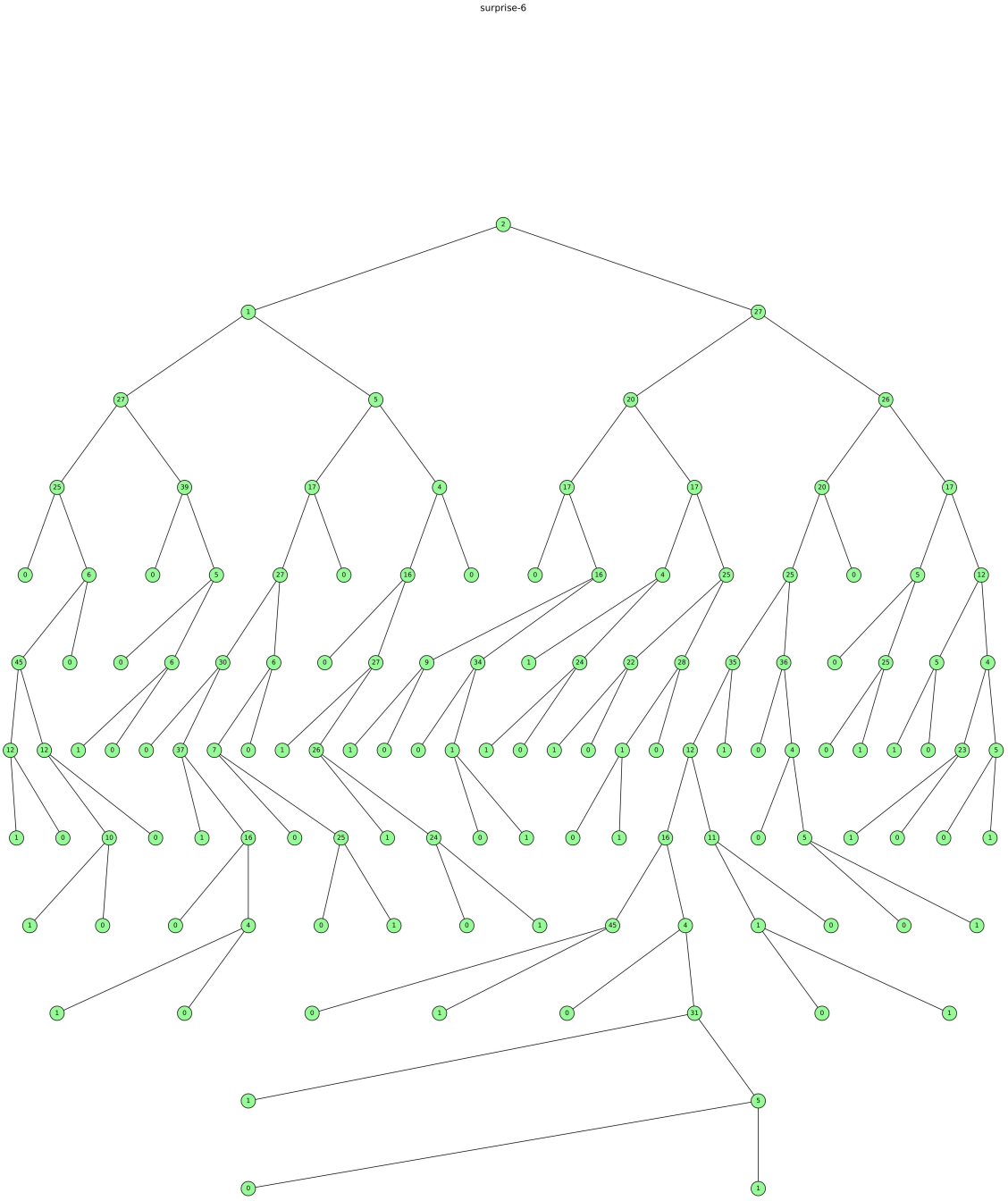



Figure A.12: Decision Tree Type 2 for Surprise

## A.2 Pruning Example

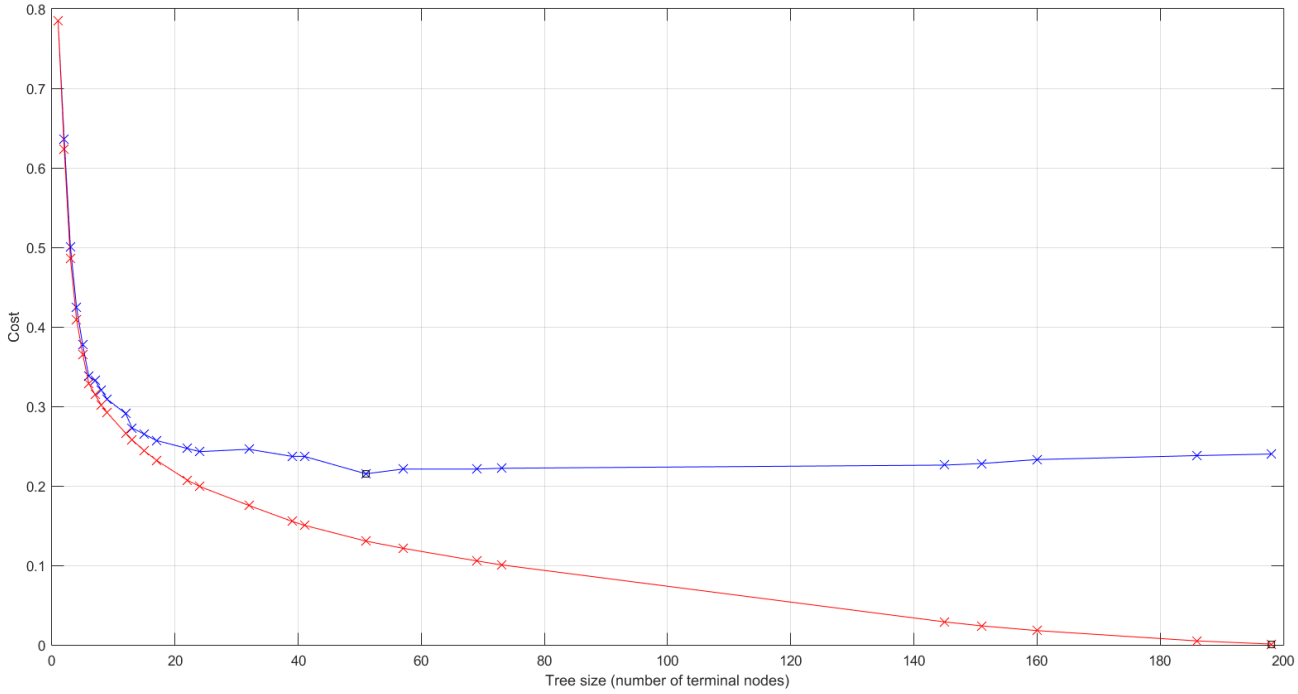



Figure A.13: pruning_example function on clean data

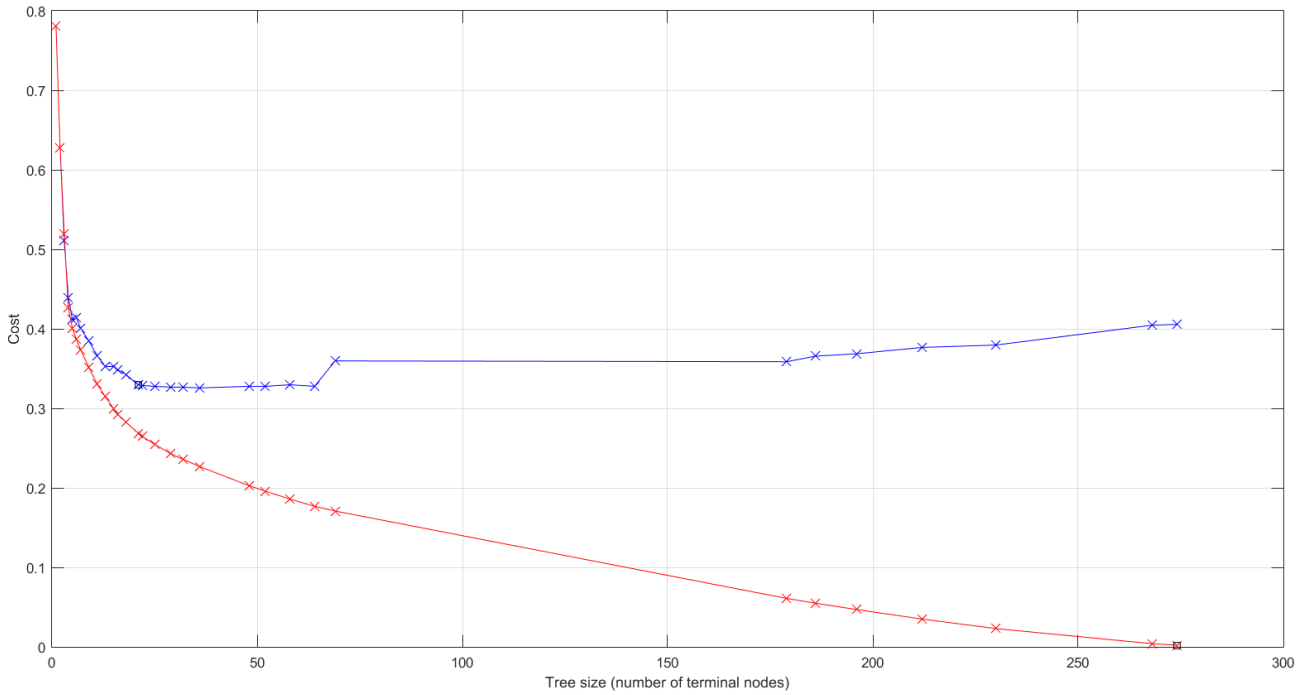



Figure A.14: pruning_example function on noisy data

# Bibliography

[1] Machine Learning by Tom Mitchell
*http://www.cs.ubbcluj.ro/~gabis/ml/ml-books/McGrawHill%20-%20Machine%20Learning%20-Tom%20Mitchell.pdf*

[2] Stanford Lecture
*https://web.stanford.edu/class/stats202/content/lec20.pdf*

[3] The Elements of Statistical Learning by Trevor Hastie, Robert Tibshirani, Jerome Friedman
*https://web.stanford.edu/~hastie/Papers/ESLII.pdf*

[4] Wikipedia - Random Forest
*https://en.wikipedia.org/wiki/Random_forest*