University of Stuttgart | ANALYTIC COMPUTING

Machine Learning Exercise (SS 21)

Assignment 5: Decision Tree (Solution)

Dr. Decky Aspandi
decky.aspandi-latif@ipvs.uni-stuttgart.de

Akram Hosseini
Akram.Hosseini@ipvs.uni-stuttgart.de

Daniel Frank
daniel.frank@ipvs.uni-stuttgart.de

This assignment sheet consists of 10 pages with the following 3 tasks:

Submit your solution in ILIAS as a single PDF file.[1] Make sure to list your full name and immatriculation number at the start of the file. Optionally, you can *additionally* upload source files (e.g. PPTX files). Remember to fill out the exercise slot and exercise presentation polls linked in ILIAS. If you have any questions, feel free to ask them in the excercise forum in ILIAS.

Submission is open until Tuesday, 8th of June 2021, 23:59 PM.

[1] Your drawing software probably allows to export as PDF. An alternative option is to use a PDF printer. If you create multiple PDF files, use a merging tool (like pdfarranger) to combine the PDFs into a single file.

Task 1: Inductive Construction (40 Points)

Given the dataset in Table 1:

1. **Task (30 Points)**: Construct a decision tree by hand using the top-down algorithm presented in the lecture. Your stop criteria are zero entropy or a depth of 2 (the root node is at depth 0, the first layer of inner nodes are at depth 1, ...). Draw your final decision tree and provide your computation steps (information gain per relevant attribute for each split, class frequencies for each node, . . .).
2. **Task (10 Points)**: Compute the error rate of your decision tree on the training data.

Depth $= 0$:

Compute entropy for root node:

	$-$	$+$	H
root	115	125	0.9987

Compute information gain for each possible split of the root node:

F_1	$-$	$+$	H	
0	15	65	0.6962	$IG = 0.2674$
1	30	50	0.9544	
2	70	10	0.5436	

F_2	$-$	$+$	H	
0	70	50	0.9799	$IG = 0.0316$
1	45	75	0.9544	

F_3	$-$	$+$	H	
0	50	70	0.9799	$IG = 0.0113$
1	65	55	0.9950	

F_4	$-$	$+$	H	
0	50	70	0.9799	$IG = 0.0113$
1	65	55	0.9950	

Because F_1 yields the highest information gain, we perform the split accordingly and construct the following tree:

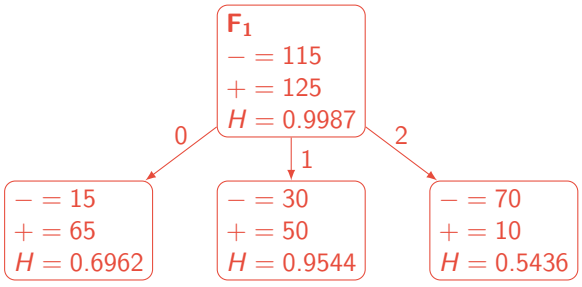


Table 1 Dataset consists of 4 categorical features ($F_1 \in \{0,1,2\}$, $F_2 \in \{0,1\}$, $F_3 \in \{0,1\}$, $F_4 \in \{0,1\}$) and a binary classification target with labels $\{-,+\}$.

F_1	F_2	F_3	F_4	Instances	
				$-$	$+$
1	1	0	1	10	0
2	1	0	0	0	10
0	1	0	1	0	10
2	0	0	1	10	0
2	1	0	1	10	0
0	0	1	0	10	0
0	0	1	1	0	10
0	0	0	0	0	10
1	1	0	0	5	5
1	0	1	0	5	5
1	0	1	1	10	0
2	0	1	0	10	0
2	0	1	1	10	0
0	1	1	0	0	10
0	1	1	1	0	10
0	1	0	0	0	10
0	0	0	1	5	5
1	1	1	0	0	10
2	0	0	0	10	0
1	1	1	1	0	10
2	1	1	0	10	0
1	0	0	0	0	10
2	1	1	1	10	0
1	0	0	1	0	10

Depth=1:

For each child node in the new layer, we repeat the process of computing IGs and splitting. Because we have split on F_1, we do not need to consider for future splits.

Compute IG of all possible splits for child node $F_1 = 0$:

F_2	$-$	$+$	H
0	15	25	0.9544
1	0	40	0.0000

$IG = 0.2190$

F_3	$-$	$+$	H
0	5	35	0.5436
1	10	30	0.8113

$IG = 0.0188$

F_4	$-$	$+$	H
0	10	30	0.8113
1	5	35	0.5436

$IG = 0.0188$

We perform a split at F_2, as it achieves the highest IG.

Compute IG of all possible splits for child node $F_1 = 1$:

F_2	$-$	$+$	H
0	15	25	0.9544
1	15	25	0.9544

$IG = 0.0000$

F_3	$-$	$+$	H
0	15	25	0.9544
1	15	25	0.9544

$IG = 0.0000$

F_4	$-$	$+$	H
0	10	30	0.8113
1	20	20	1.0000

$IG = 0.0487$

We perform a split at F_4, as it achieves the highest IG.

Compute IG for all possible for child node $F_1 = 2$:

F_2	$-$	$+$	H
0	40	0	0.0000
1	30	10	0.8113

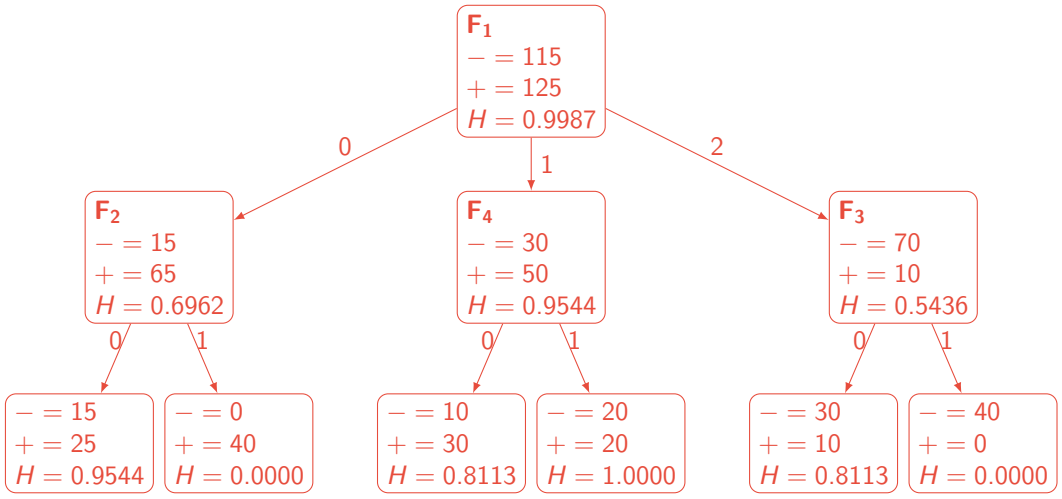
$IG = 0.1379$

F_3	$-$	$+$	H	
0	30	10	0.8113	$IG = 0.1379$
1	40	0	0.0000	

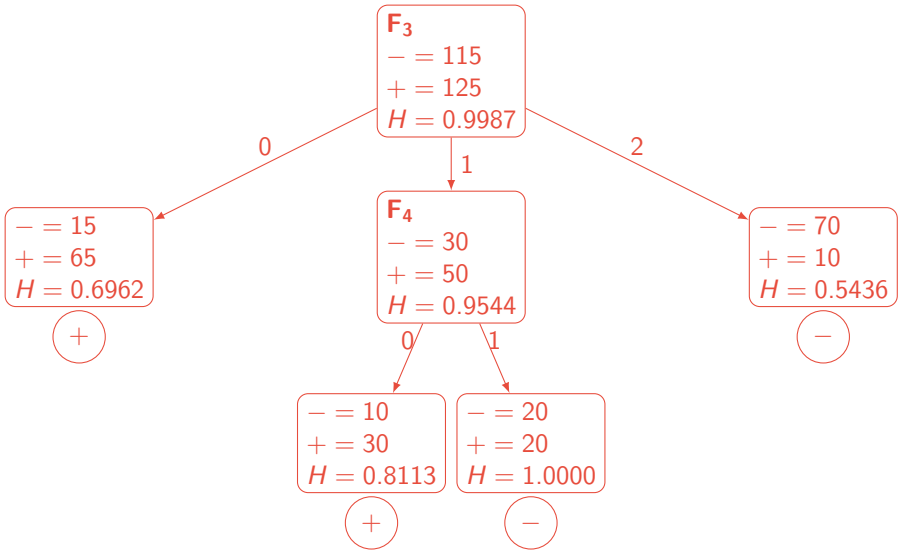
F_4	$-$	$+$	H	
0	30	10	0.8113	$IG = 0.1379$
1	40	0	0.0000	

All splits yield the same IG, we therefore arbitrarily choose to split at F_3.

We have determined splits for all nodes at depth 1 and can construct the following tree:



Depth $= 2$:
Our stopping criterion is a depth of 2, therefore our tree construction is finished. The classification decision of each leave node is determined by the majority of class labels. We can see that for the leaves of node F_2 and F_3 the same class is predicted, which allows us to prune them without any performance loss. The leaf $F_4 = 1$ has an equal number of negative and positive labels, accordingly we have to arbitrarily select one class label as classification decision. We choose the label $-$. This yields the following tree:



Error rate: Finally, we can compute the overall error rate on the training data:

$$E = \frac{15 + 10 + 20 + 10}{80 + 40 + 40 + 80} = \frac{55}{240} = 0.23$$

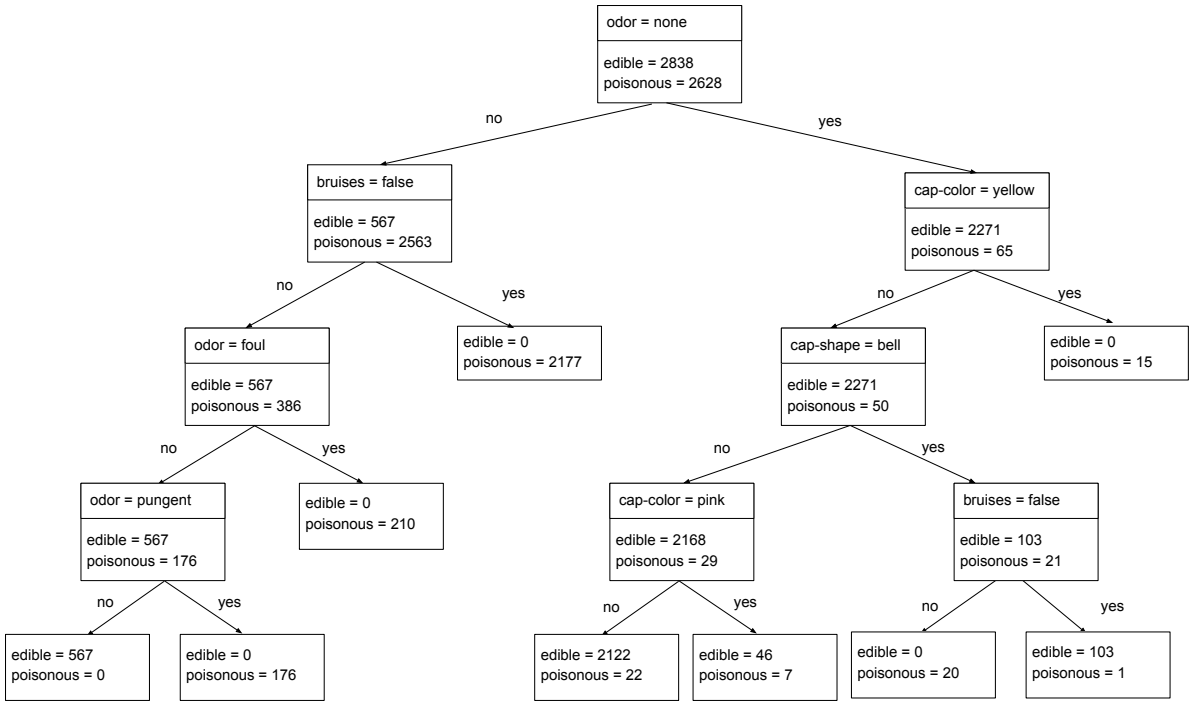


Figure 1 Decision tree of depth 4 classifies mushrooms as edible or poisonous. Inner nodes have a decision rule at the top, the left child is chosen if the rule does not apply and the right child is chosen if the rule applies. For each leaf and inner node, the class frequencies for the training data are summarized.

Task 2: Minimal Error Pruning (35 Points)

Figure 1 shows a decision tree for classifying mushrooms as either edible or poisonous. We constructed it with scikit-learn with a maximum depth of 4. Because the implementation of scikit-learn uses a slightly different algorithm (CART, in the lecture: ID3), which allows continuous data, we have to apply one-hot encoding to each variable resulting in a binary decision tree.

1. **Task (35 Points)**: Prune two decisions (inner nodes) of the decision tree based on the error rate. First, compute the overall error rate of the original tree. Second, consider the removal of each viable node by computing the error rate after removing it from the tree. Third, identify the node with the lowest error rate and prune it. Repeat this process to remove a second inner node. Draw the decision tree after each pruning step. Provide your calculations for the pruning step.

Note: A node is viable for pruning, if all its children are leaf nodes. After pruning the chosen inner node turns into a leaf node. We define the overall error rate as (please check lecture's slides):

$$\frac{\text{number of misclassified samples in each leaf}}{\text{total number of samples}}$$

The error rate for the original tree is:

$$E = \frac{22 + 7 + 1}{5466} = 0.0054$$

The error rate after removing "odor = pungent" is:

$$\frac{176 + 22 + 7 + 1}{5466} = 0.0376$$

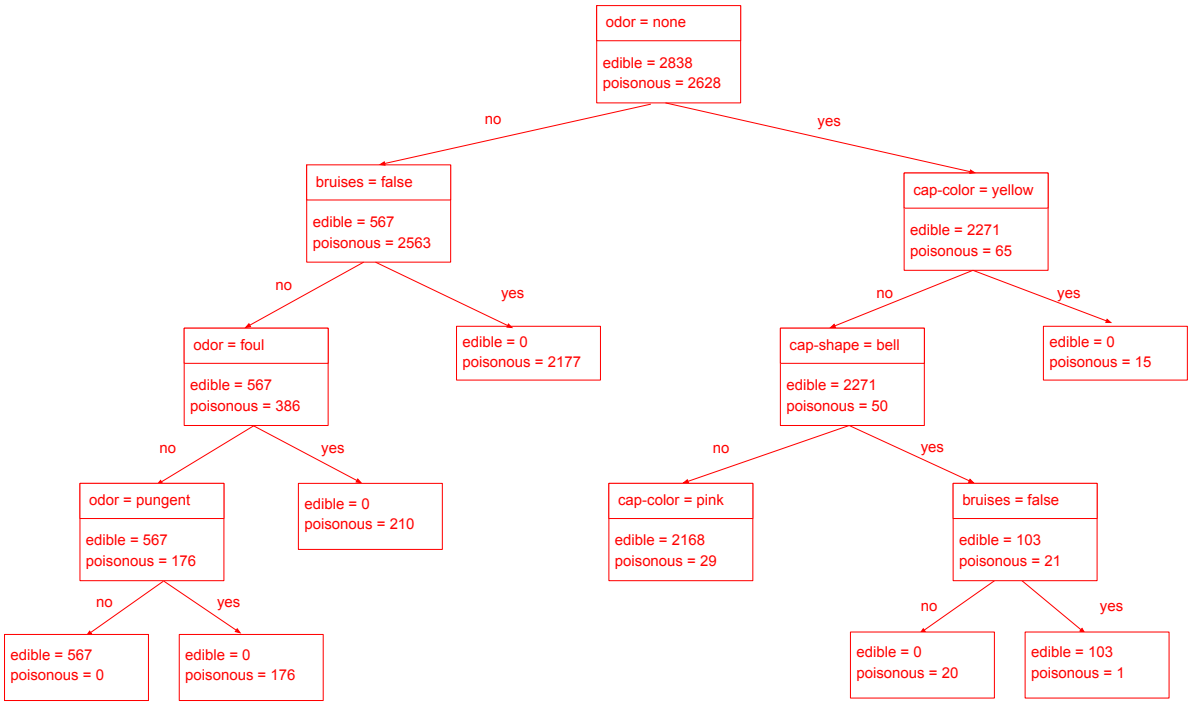
The error rate after removing "cap-color = pink" is:

$$\frac{29+1}{5466} = 0.0054$$

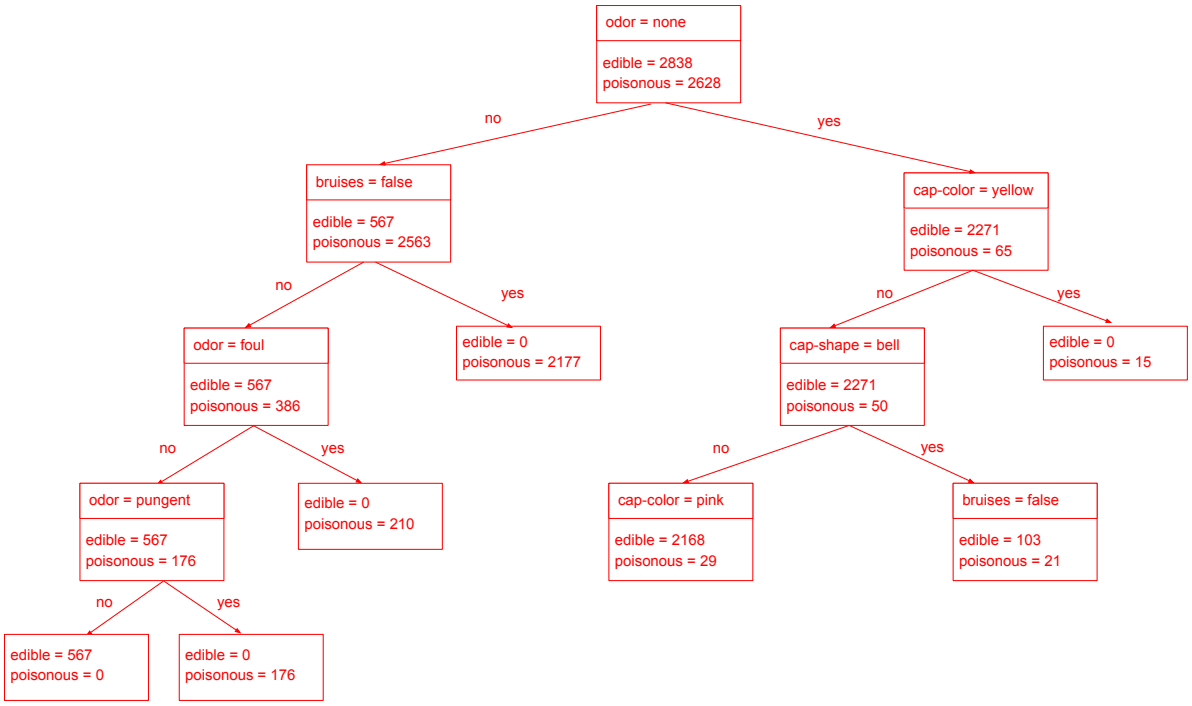
The error rate after removing "bruises = false" is:

$$\frac{22+7+21}{5466} = 0.00915$$

Our first pruning operation removes the node "cap-color = pink", since the error rate does not chance due to its removal. This results in the following tree:



For the second pruning operation, we can reuse our previous computations. We prune the node "bruises = false", as it leads to the lowest increase in error rate. Accordingly, our final pruned tree follows:


odor = none
edible = 2838
poisonous = 2628
no
yes
bruises = false
edible = 567
poisonous = 2563
cap-color = yellow
edible = 2271
poisonous = 65
no
yes
no
yes
odor = foul
edible = 567
poisonous = 386
edible = 0
poisonous = 2177
cap-shape = bell
edible = 2271
poisonous = 50
edible = 0
poisonous = 15
no
yes
no
yes
odor = pungent
edible = 567
poisonous = 176
edible = 0
poisonous = 210
cap-color = pink
edible = 2168
poisonous = 29
bruises = false
edible = 103
poisonous = 21
no
yes
edible = 567
poisonous = 0
edible = 0
poisonous = 176

Task 3: Regression with Decision Trees and kNN (25 Points)

Decision trees and k-nearest-neighbors are not limited to classification but can also be used for regression. Research and explain the following two questions in sufficient detail:

1. **Task (15 Points)**: How does the construction of regression trees differ to classification trees? How is a prediction computed in a regression tree?

   To build a regression tree we start in a similar way for constructing decision tree. However, the differences between construction of a classification tree and a regression tree is that the output variable in regression is numerical (continuous values) while that for classification is categorical (discrete). Entropy is not suitable for continuous variables, we need another calculation to compute the prediction in regression trees, for example the Mean Square Error (MSE).

2. **Task (10 Points)**: How can kNN be used for regression? Please cite your sources.

   In order to use kNN for regression, first we need to look for the k nearest points to the new data point, then we should calculate the average distance of those points, and finally connect these points gives us the regression line.