Massively Parallel Algorithms
Classification & Prediction Using Random Forests

G. Zachmann
University of Bremen, Germany
cgvr.cs.uni-bremen.de

Classification Problem Statement

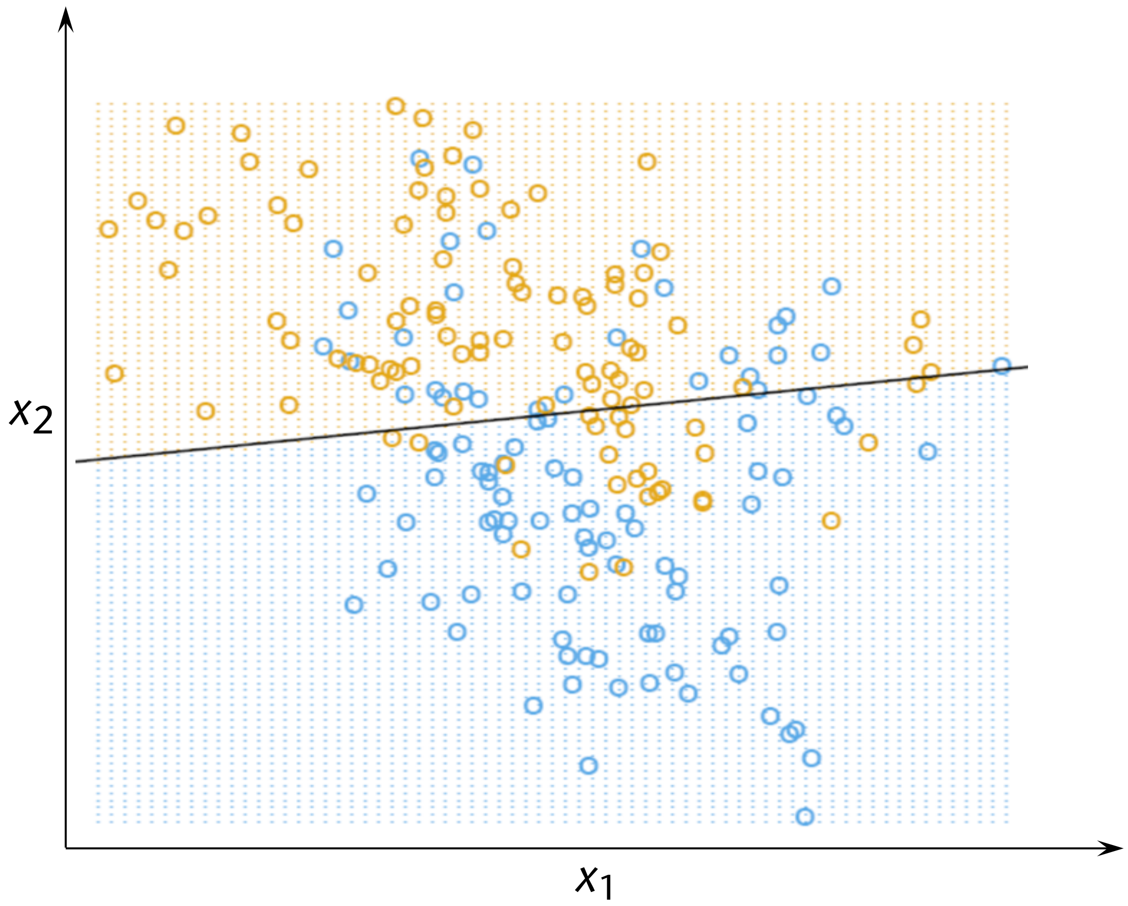
- Given a set of points $\mathcal{L} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \in \mathbb{R}^d$ and for each such point a label $y_i \in \{l_1, l_2, \ldots, l_n\}$
  - Each label represents a class, all points with the same label are in the same class
- Wanted: a method to decide for a *not-yet-seen* point $\mathbf{x}$ which label it most probably has, i.e., a method to *predict class labels*
  - We say that we learn a classifier C from the training set $\mathcal{L}$:

$$C : \mathbb{R}^d \to \{l_1, l_2, \ldots, l_n\}$$

- Typical applications:
  - Computer vision (object recognition, ...)
  - Credit approval
  - Medical diagnosis
  - Treatment effectiveness analysis

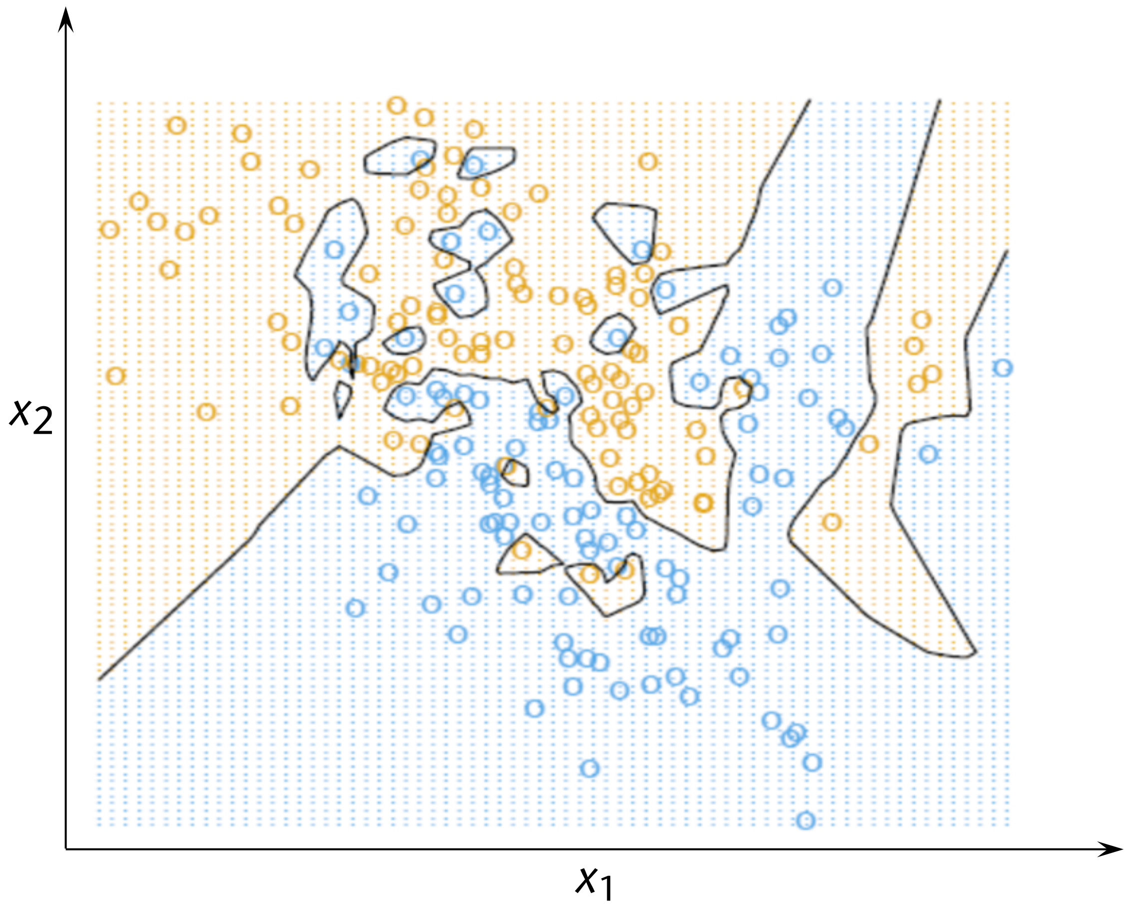
One Possible Solution: Linear Regression

- Assume we have only two classes (e.g., "blue" and "yellow")
- Fit a plane through the data



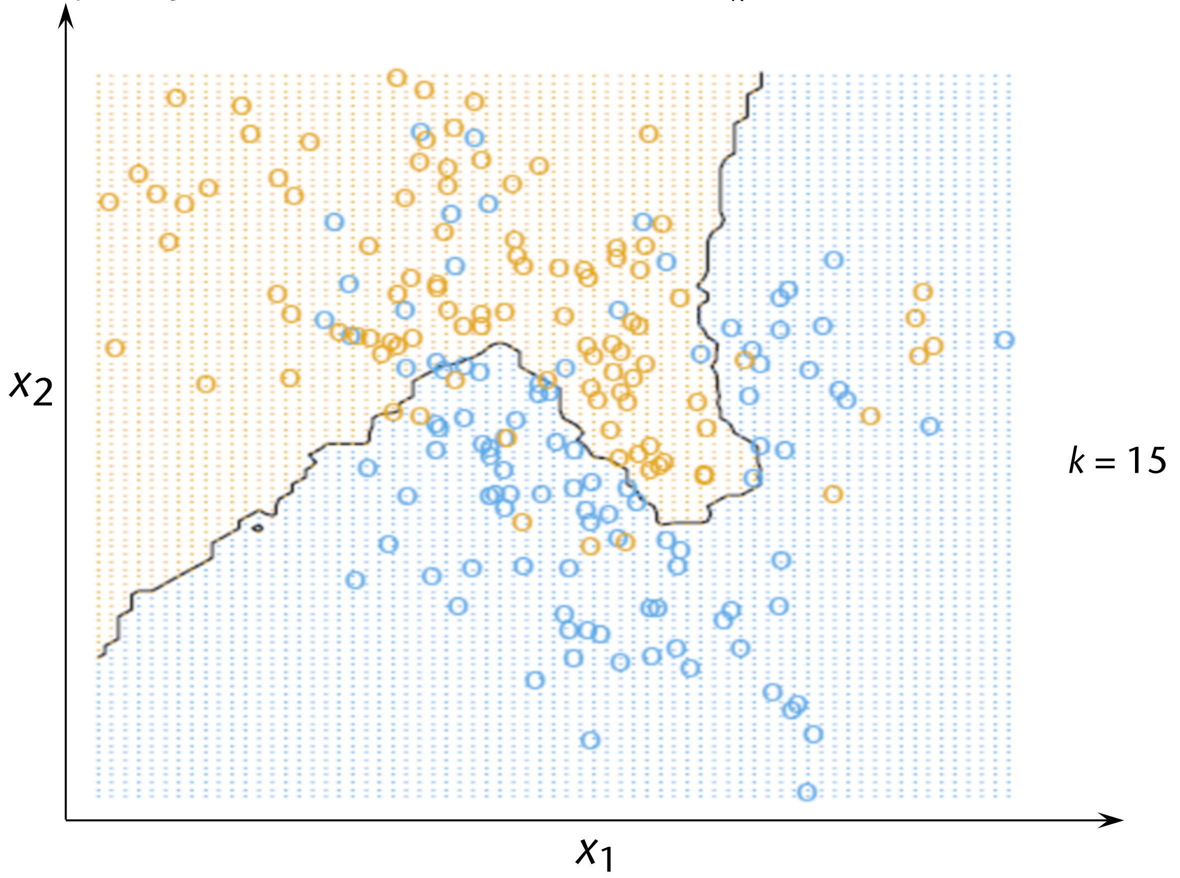
Another Solution: Nearest Neighbor (NN)

- For the new point **x**, find the nearest neighbor $\mathbf{x}^* \in \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \in \mathbb{R}^d$
- Assign the class l^* to **x**



Improvement: k-NN

- Instead of the 1 nearest neighbor, find the k nearest neighbors of $\mathbf{x}$, $\{\mathbf{x}_{i_1}, \ldots, \mathbf{x}_{i_k}\} \subset \mathcal{L}$
- Assign the majority of the labels $\{l_{i_1}, \ldots, l_{i_k}\}$ to $\mathbf{x}$

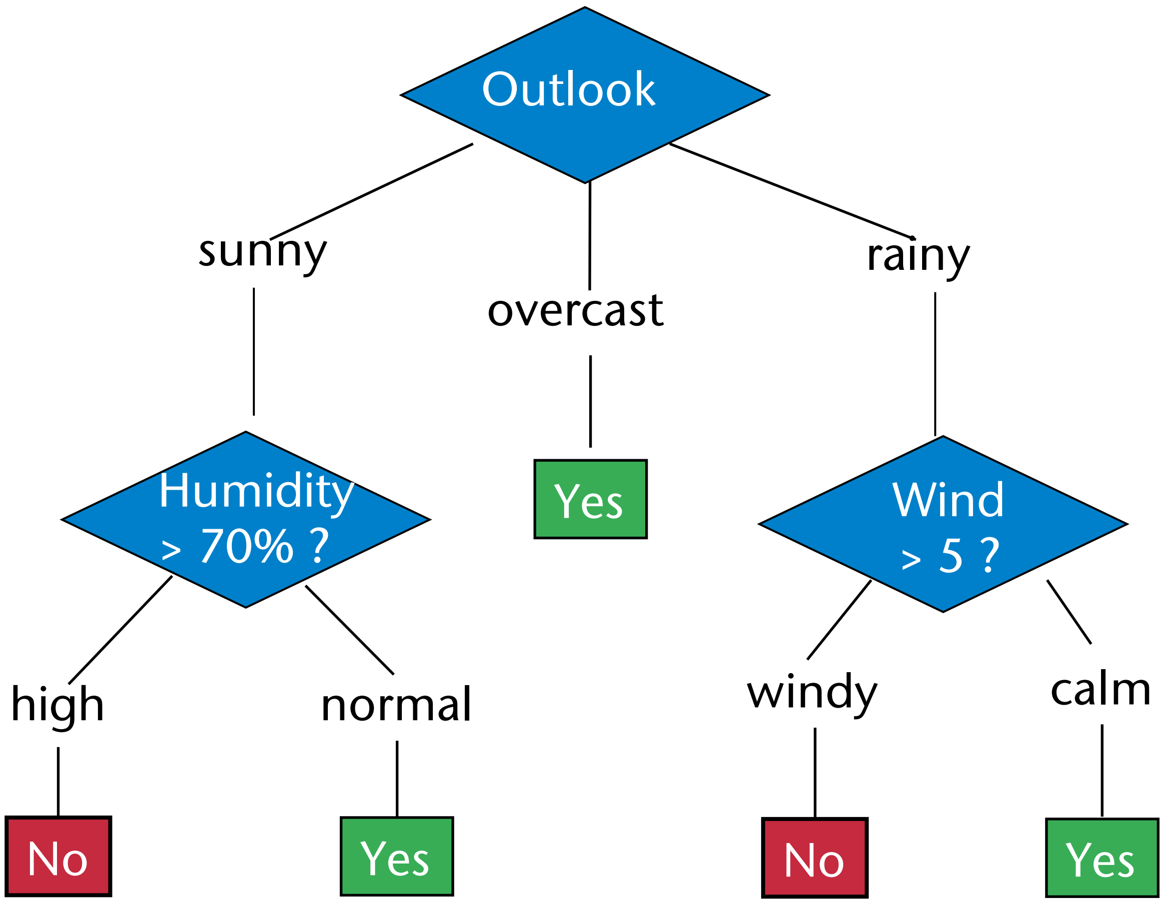


More Terminology

- The coordinates/components $x_{i,j}$ of the points $\mathbf{x}_i$ have special names: independent variables, predictor variables, features, …
  - Specific name of the $x_{i,j}$ depends on the domain
- The space where the $\mathbf{x}_i$ live (i.e., $\mathbb{R}^d$) is called feature space
- The labels y_i are also called target, dependent variable, response variable, …
- The set $\mathcal{L}$ is called the training set / learning set (will become clear later)

Decision Trees

- Simple example: decide whether to play tennis or not



A new sample (observation):
(Outlook=rainy,
Wind=calm,
Humidity=high)

Pass it down the tree ⟶ decision is yes.

- The *feature space* = "all" weather conditions
  - Based on the attributes

    outlook ∈ { sunny, overcast, rainy },

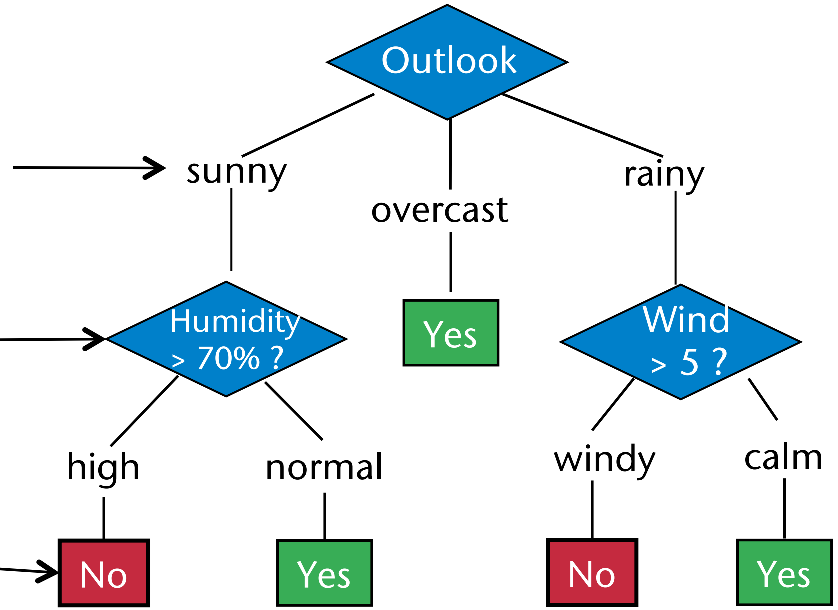
    humidity ∈ [0,100] percent ,

    wind ∈ {0, 1, …, 12} Beaufort
  - Here, our feature space is mixed continuous/discrete
- Anatomy of a decision tree:

Branches represent different values or ranges of the attribute(s)

Each node tests one or more attribute(s). This is sometimes called a weak classifier

Leaves are the eventual decisions (= classes/labels)



Another Example

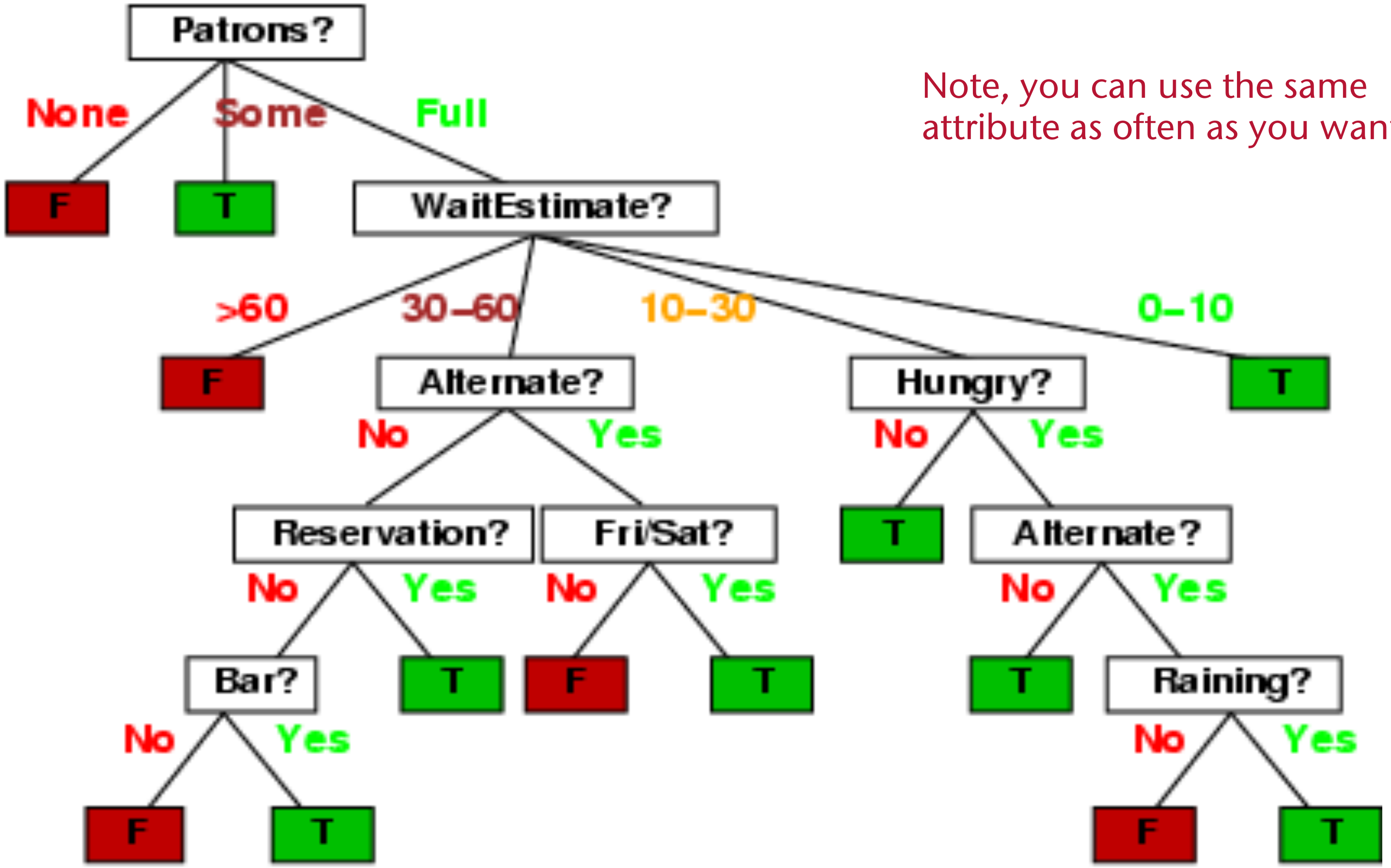
- "Please wait to be seated" ...
- Decide: *wait* or *go* some place else?
- Variables that could influence your decision:
  - Alternate: is there an alternative restaurant nearby?
  - Bar: is there a comfortable bar area to wait in?
  - Fri/Sat: is today Friday or Saturday?
  - Hungry: are we hungry?
  - Patrons: number of people in the restaurant (None, Some, Full)
  - Price: price range ($, $$, $$$)
  - Raining: is it raining outside?
  - Reservation: have we made a reservation?
  - Type: kind of restaurant (French, Italian, Thai, Burger)
  - WaitEstimate: estimated waiting time (0-10, 10-30, 30-60, >60)

- You collect data to base your decisions on:

Example	Attributes										Target
	Alt	Bar	Fri	Hun	Pat	$Price$	$Rain$	Res	$Type$	Est	$Wait$
X_1	T	F	F	T	Some	\$\$\$	F	T	French	0–10	T
X_2	T	F	F	T	Full	\$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	\$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	\$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	\$\$\$	F	T	French	>60	F
X_6	F	T	F	T	Some	\$\$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	\$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	\$\$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	\$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	\$\$\$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	\$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	\$	F	F	Burger	30–60	T

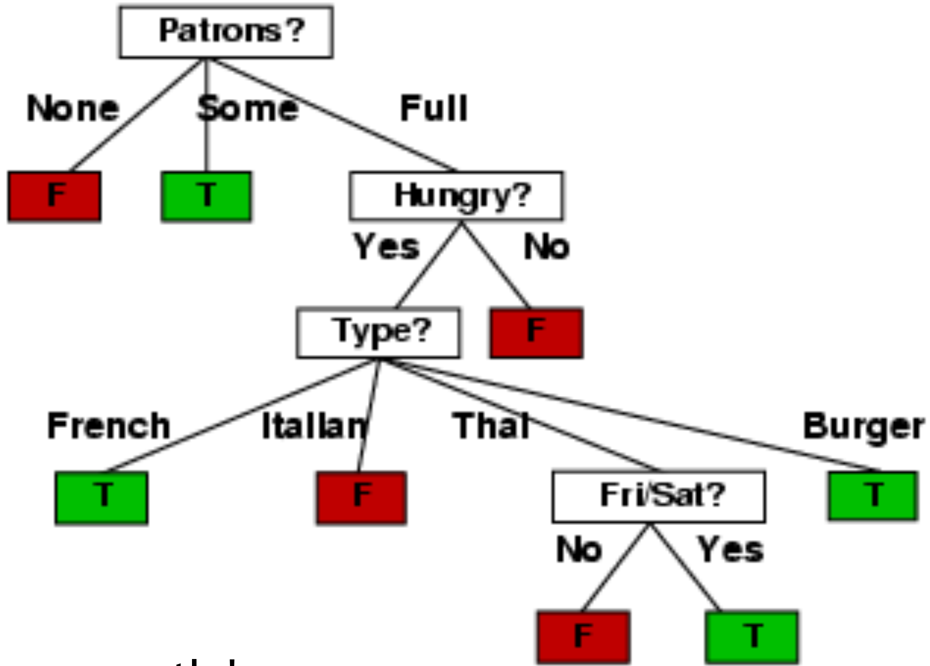
- Feature space: 10-dimensional, 6 Boolean attributes, 3 discrete attributes, one continuous attribute

- A decision tree that classifies all "training data" correctly:



Note, you can use the same attribute as often as you want

- A better decision tree:

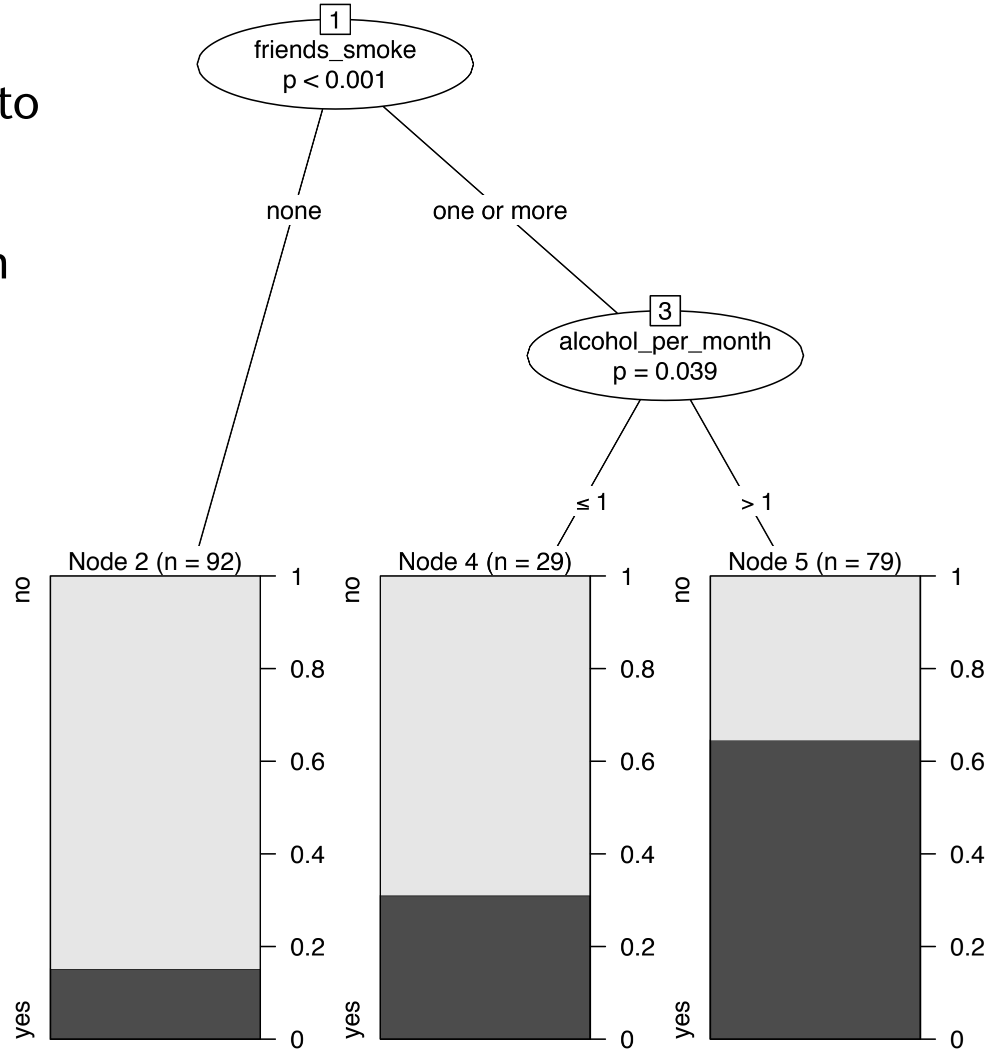


  - Also classifies all training data correctly!
  - Decisions can be made faster
- Questions:
  - How to construct (optimal) decision trees methodically?
  - How well does it generalize? (what is its generalization error?)

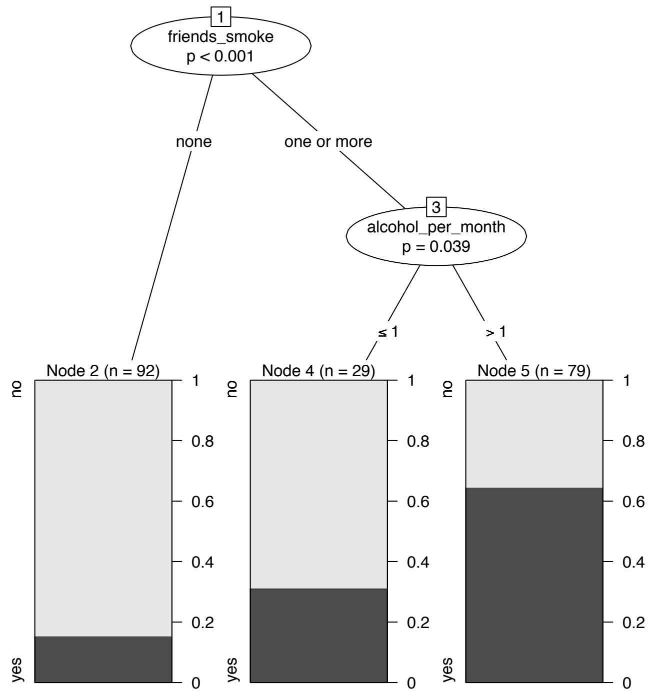
Construction (= Learning) of Decision Trees

- By way of the following example
- Goal: predict adolescents' intention to smoke within next year
  - Binary response variable *IntentionToSmoke*
- Four predictor variables (= attributes):
  - *LiedToParents* (bool) = subject has ever lied to parents about doing something they would not approve of
  - *FriendsSmoke* (bool) = one or more of the 4 best friends smoke
  - *Age* (int) = subject's current age
  - *AlcoholPerMonth* (int) = # times subject drank alcohol during past month
- Training data:
  - Kitsantas et al.: *Using classification trees to profile adolescent smoking behaviors*. 2007
  - 200 adolescents surveyed

- A decision tree:
  - Root node splits all points into *two subsets*
  - Node 2 = all data points with *FriendsSmoke* = False
  - Node 2 contains 92 points, 18% have label "yes", 82% have label "no"
  - Ditto for the other nodes

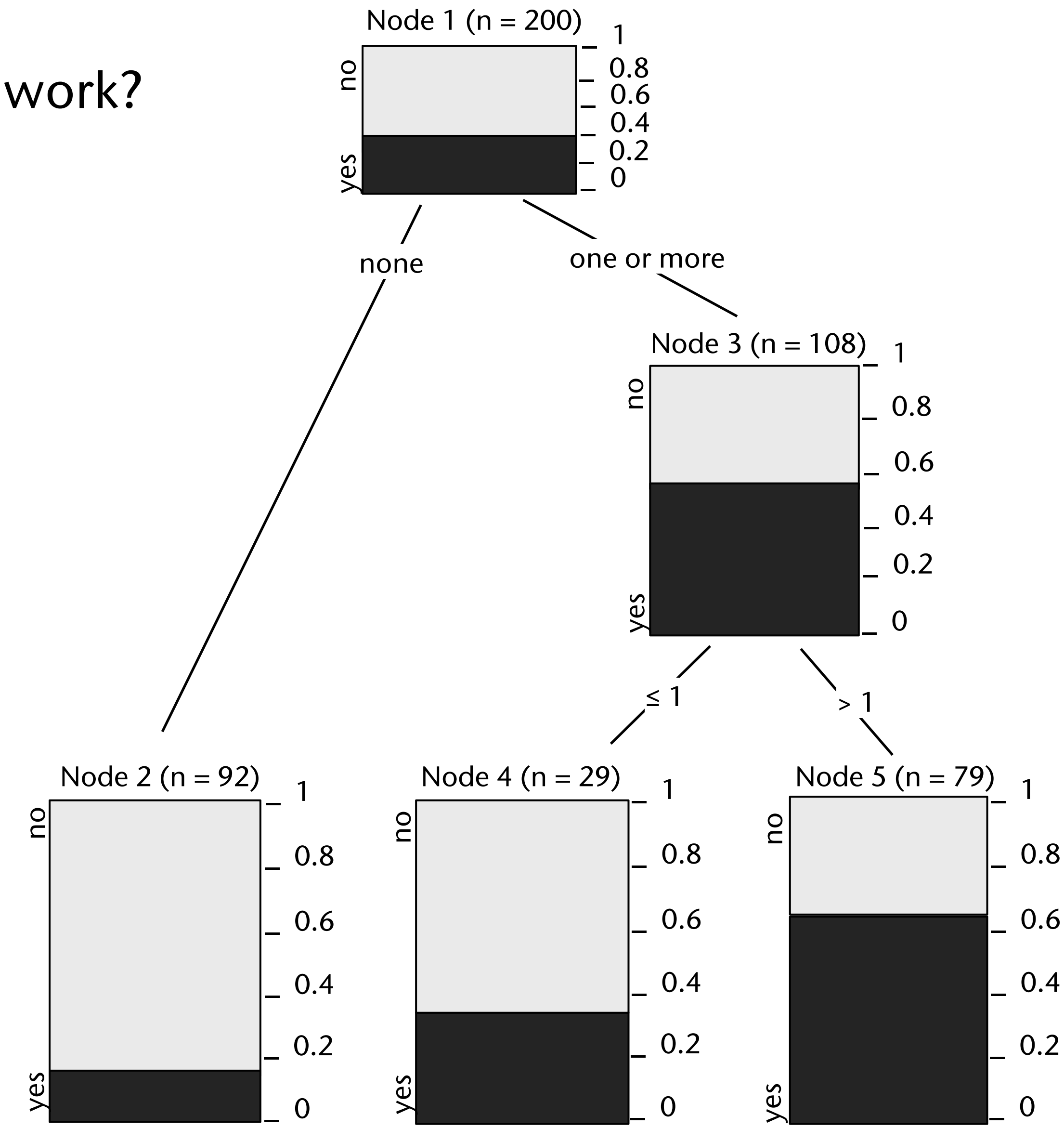


- Observation: a decision tree partitions feature space into rectangular regions:

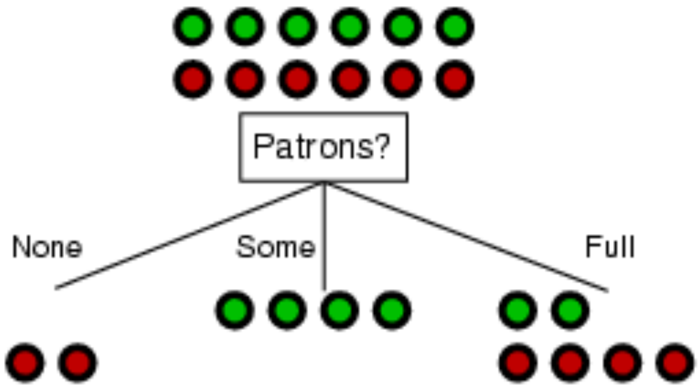


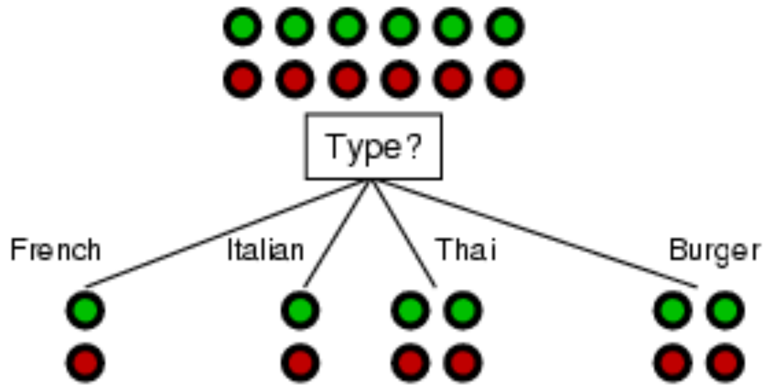
Selection of Splitting Variable and Cutpoint

- Why does our example work?
  - In the root node, *IntentionToSmoke*=yes is 40%
  - In node 2, *IntentionToSmoke*=yes is 18%, while in node 3 *IntentionToSmoke*=yes is 60%
  - So, after first split we can make better predictions



- Ideally, a good attribute (and cutpoint) splits the samples into subsets that are "all positive" or "all negative"
- Example (restaurant):





To wait or not to wait is still at 50%

Goals for Splitting Nodes

- We want (summed diversity within children) < (diversity in parent)
- Data points should be
  - Homogeneous (by labels) within leaves
  - Different between leaves
- Goal: try to increase purity within subsets
  - *Optimization* goal in each node: find the *attribute* and a *cutpoint* that splits the set of samples into two subsets with *optimal purity*
  - This attribute is the "most discriminative" one for that data (sub-) set
- Question: what is a good measure of purity for two given subsets of our training set?

Information Gain

- Enter the information theoretic concept of information gain
- Imagine different events:
  - The outcome of rolling a dice = 6
  - The outcome of rolling a *biased* dice = 6
  - Each situation has a different amount of uncertainty whether or not the event will occur
- Information = *amount of reduction in uncertainty* (= amount of surprise if a specific outcome occurs)

- Let Y be a random variable; then we make one observation of the variable Y (e.g., we draw a random ball out of a box) $\longrightarrow$ value y
- The information we obtain if event "$Y = y$" occurs is

$$I[Y = y] = \log_2 \frac{1}{p(y)} = -\log p(y)$$

  - "If the probability of this event happening is small and it happens, then the information is large"
- Examples:
  - Observing the outcome of coin flip $\longrightarrow$ $I = -\log \frac{1}{2} = 1$
  - Observing the outcome of dice = 6 $\longrightarrow$ $I = -\log \frac{1}{6} = 2.58$

Entropy

- A random variable Y (= experiment) can assume different values $y_1, \ldots, y_n$ (i.e., the experiment can have different outcomes)
- What is the *average* information we obtain by observing the random variable?
  - In probabilistic terms: what is the *expected amount of information*? $\longrightarrow$ captured by the notion of entropy
- Definition: Entropy
  Let Y be a random variable. The entropy of Y is

$$H(Y) = E[\mathcal{I}(Y)] = \sum_i p(y_i)\mathcal{I}[Y = y_i] = -\sum_i p(y_i) \log p(y_i)$$

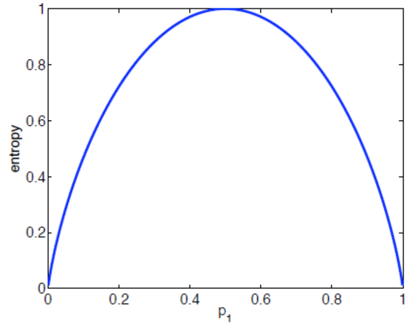
- Example: if Y can assume 8 values, and all are equally likely, then

$$H(Y) = -\sum_{i=1}^{8} \frac{1}{8} \log \frac{1}{8} = \log 2^3 = 3 \text{ bits}$$

- In general, if there are k possible outcomes, then

$$H(Y) \leq \log k$$

  - Equality holds when all outcomes are equally likely
- With k = 2 (two outcomes), entropy looks like this:



- The more the probability distribution deviates from uniformity the lower the entropy
- *Entropy* measures the *purity*:

4 "yes"
4 "no"

8 "yes"
0 "no"

This distribution is less uniform
Entropy is lower
The node is purer

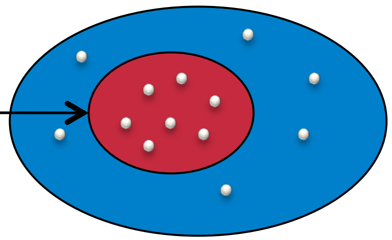
Conditional Entropy

- Now consider a random variable Y (e.g., the different classes/labels) with an attribute X (e.g., the first variable, $x_{i,1}$, of the data points, $\mathbf{x}_i$)
  - With every drawing of Y, we also get a value for the associated attribute X
- Assume that X is discrete, i.e., $x_i \in \{1, 2, ..., z\}$
- We now consider only cases of Y that fulfill some *condition*, e.g., x_i=1
- The entropy of Y, provided that it assumes only values with x_i =1:

$$H(Y|x_i = 1) = -\sum_i p(y_i|x_i = 1) \log \underbrace{p(y_i|x_i = 1)}$$

Probability of y_i occurring as a value of Y, where we consider only the subset of values of Y that have attribute x_i =1

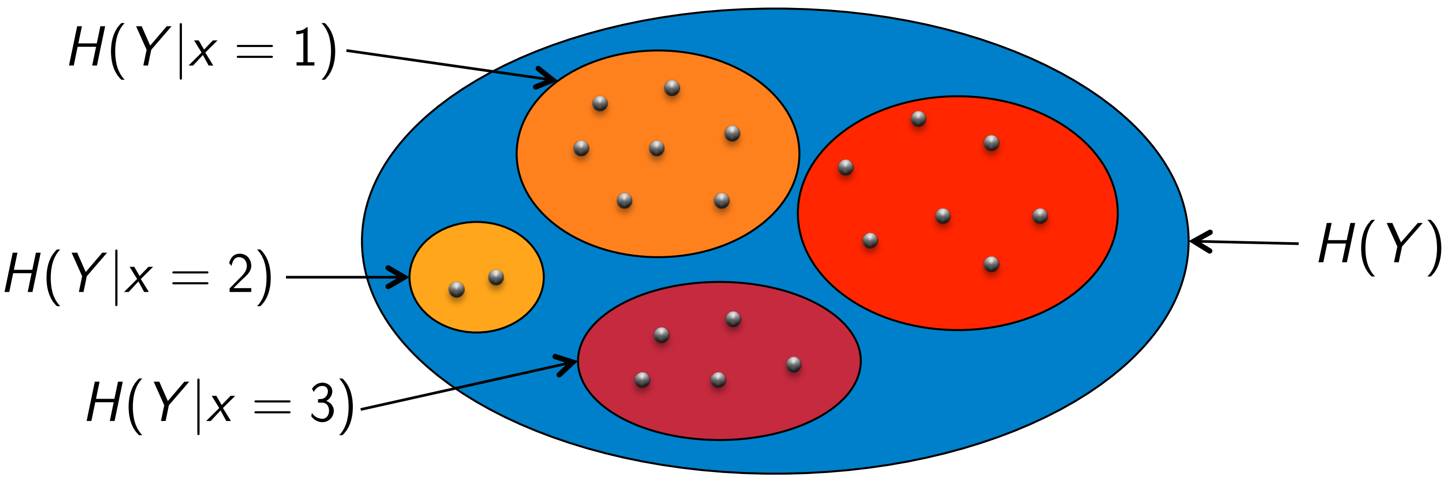
Subset with x_i =1



- Overall conditional entropy:

$$H(Y|X) = \sum_{k=1}^{z} p(x=k) \cdot H(Y|x=k)$$

$$= -\sum_{k=1}^{z} \underbrace{p(x=k)}_{\text{Probability that the attribute } X \text{ has value } k} \sum_{i} p(y_i|x_i=k) \log p(y_i|x_i=k)$$



Information Gain

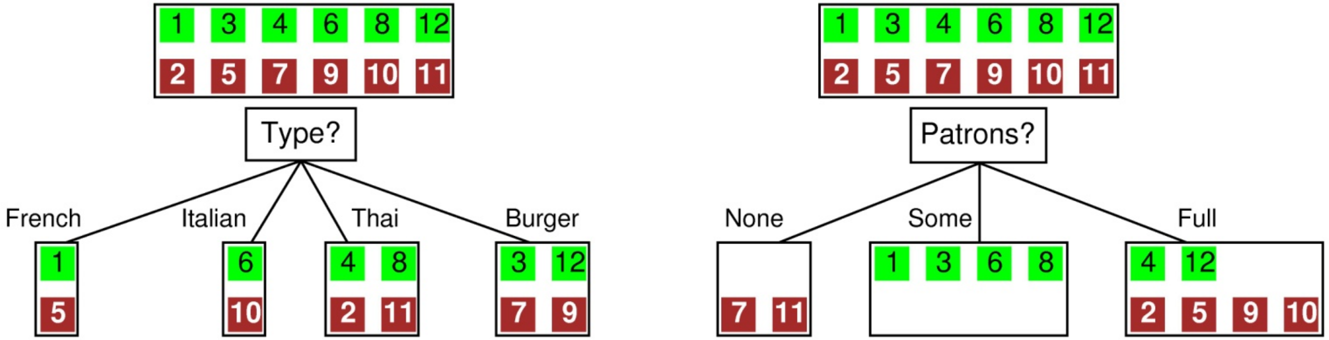
- How much information do we gain if we disclose the value of some attribute?
- Information gain = (information *before* split) – (information *after* split) = *reduction of uncertainty* by knowing attribute X
- The information gained by a split in a node of a decision tree:

$$IG(Y,X) = H(Y) - H(Y|X)$$

- Goal: choose the attribute with the largest IG
  - In case of scalar attributes, also choose the *optimal cutpoint*

Example

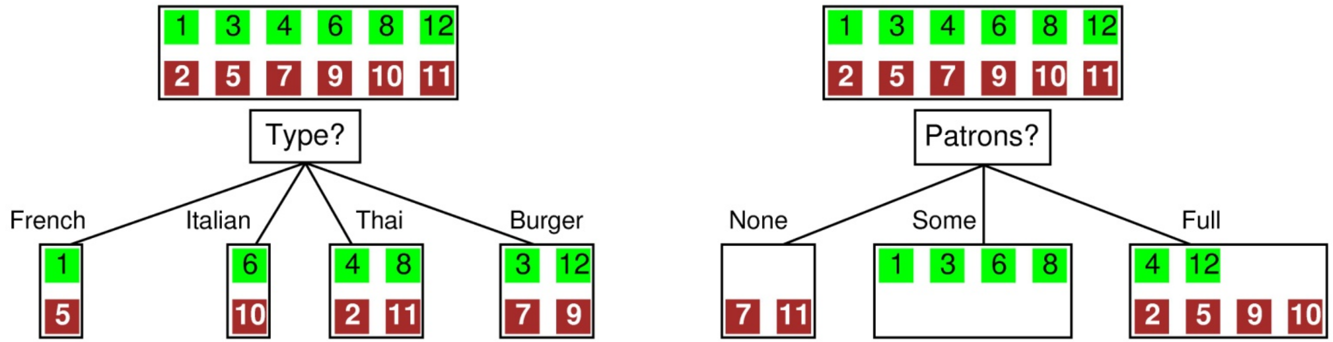
- Consider 2 options to split the root node of the restaurant example



- Random variable $Y \in \{ \text{"yes"}, \text{"no"} \}$
- At the root node:

$$H(Y) = p(y = \text{“yes”}) \log \frac{1}{p(y = \text{“yes”})} + p(y = \text{“no”}) \log \frac{1}{p(y = \text{“no”})}$$

$$= \frac{1}{2} \log 2 + \frac{1}{2} \log 2 = 1$$



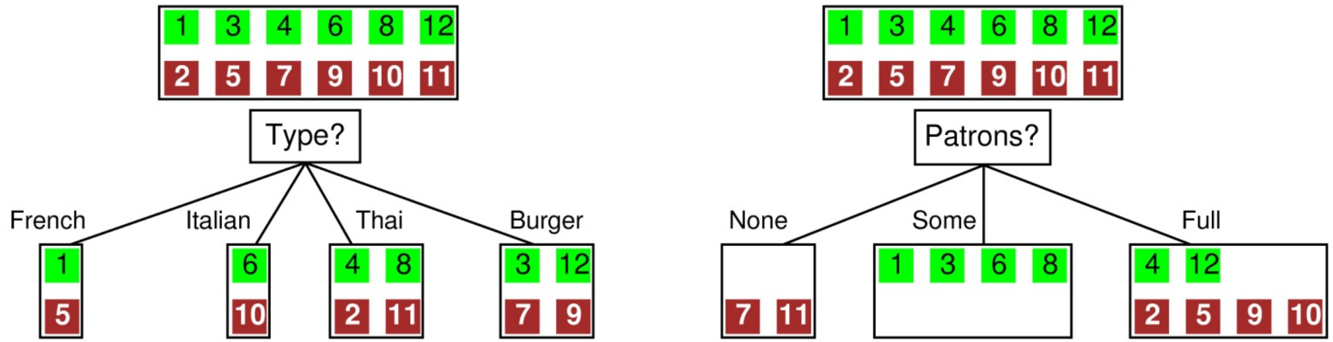
- Conditional entropy for right option:

$$H(Y \mid n) = p(n{=}\text{“none”})\cdot H(Y|n{=}\text{“none”}) + p(n{=}\text{“some”})\cdot H(Y|n{=}\text{“some”}) + p(n{=}\text{“full”})\cdot H(Y|n{=}\text{“full”})$$

where n = the attribute "#patrons" ∈ { "none", "some", "full" }

$$H(Y|\#\text{patrons}) = \frac{2}{12}\big(p(y{=}\text{“no”})\log p(y{=}\text{“no”}) + p(y{=}\text{“yes”})\log p(y{=}\text{“yes”})\big) + \frac{4}{12}\big(p(y{=}\text{“no”})\log p(y{=}\text{“no”}) + p(y{=}\text{“yes”})\log p(y{=}\text{“yes”})\big) + \frac{6}{12}\big(p(y{=}\text{“no”})\log p(y{=}\text{“no”}) + p(y{=}\text{“yes”})\log p(y{=}\text{“yes”})\big)$$

$$H(Y|\#\text{patrons}) = \frac{2}{12}(1\log 1 + 0\log 0) + \frac{4}{12}(0\log 0 + 1\log 1) + \frac{6}{12}\left(\frac{4}{6}\log\frac{6}{4} + \frac{2}{6}\log\frac{6}{2}\right)$$



- Conditional entropy for left option:

$$
\begin{aligned}
H(Y|\text{type}) = \;& \frac{2}{12}\Big(p(y=\text{“no”})\log p(y=\text{“no”}) + p(y=\text{“yes”})\log p(y=\text{“yes”})\Big) + \\
& \frac{2}{12}\Big(p(y=\text{“no”})\log p(y=\text{“no”}) + p(y=\text{“yes”})\log p(y=\text{“yes”})\Big) + \\
& \frac{4}{12}\Big(p(y=\text{“no”})\log p(y=\text{“no”}) + p(y=\text{“yes”})\log p(y=\text{“yes”})\Big) + \\
& \frac{4}{12}\Big(p(y=\text{“no”})\log p(y=\text{“no”}) + p(y=\text{“yes”})\log p(y=\text{“yes”})\Big) +
\end{aligned}
$$

$$
H(Y|\text{type}) = 2\cdot\frac{2}{12}\Big(\frac{1}{2}\log\frac{2}{1} + \frac{1}{2}\log\frac{2}{1}\Big) \; + \; 2\cdot\frac{4}{12}\Big(\frac{2}{4}\log\frac{4}{2} + \frac{2}{4}\log\frac{4}{2}\Big)
$$

- Compare the information gains:

$$IG(Y, \#\text{patrons}) = H(Y) - H(Y|\#\text{patrons})$$
$$= 1 - 0.585$$

$$IG(Y, \text{type}) = H(Y) - H(Y|\text{type})$$
$$= 1 - 1$$

  - So, the attribute "#patrons" gives us more information about *Y*

- Compute the *IG* obtained by a split induced by *each attribute*
  - In this case, the optimum is achieved by the attribute "#patrons" for splitting the set of data points in the root

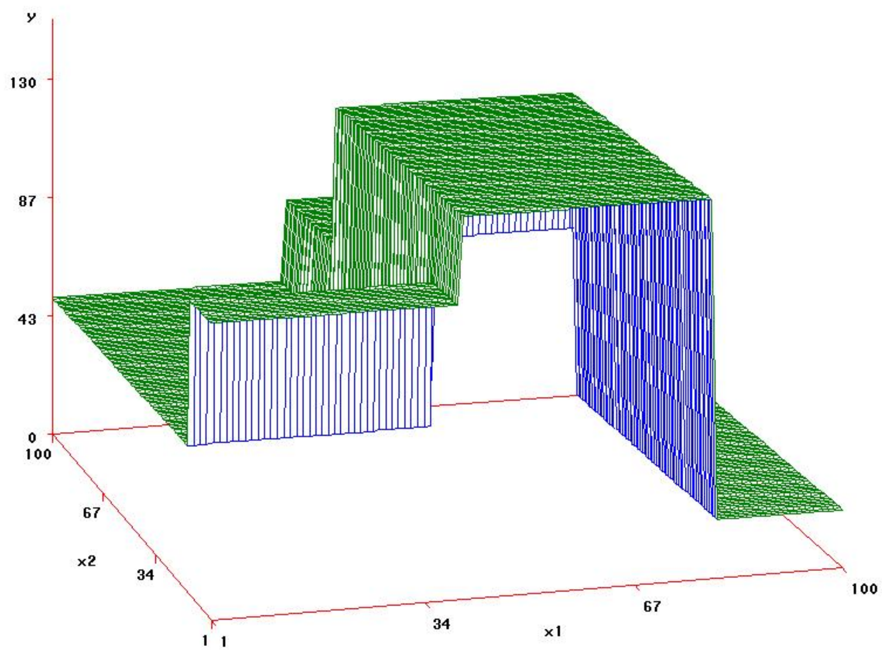
Bits and Pieces

- If there are no attributes left:
  - Can happen during learning of the decision tree, when a node contains data points with same attributes but different labels
  - This constitutes error / noise
  - Stop construction here, use majority vote (discard erroneous point)
- If there are leaves with no data points:
  - While classifying a new data point
  - Just choose the majority vote of the parent node

Expressiveness of Decision Trees

- Assume all variables (attributes and labels) are Boolean
- What is the class of Boolean functions that can be represented by a decision tree?
- Answer: all Boolean functions!
- Proof (simple):
  - Given any Boolean function
  - Convert it to a truth table
  - Consider each row as a data point, output = label
  - Construct a DT over all data points / rows

- If Y is a discrete, numerical variable, then DTs can be regarded as piecewise constant functions over the feature space:



- DTs can approximate *any* function

Problems of Decision Trees

- Error propagation:
  - Learning a DT is based on a series of local decisions
  - What happens, if one of the nodes implements the wrong decision? (e.g., because of an outlier)
  - The whole subtree will be wrong!
- Overfitting: in general, it means the learner performs extremely well on the training data, but very poorly on unseen data ⟶ high generalization error
  - When overfitting occurs, the DT has learned the noise in the data

- Example for the instability of single decision trees: