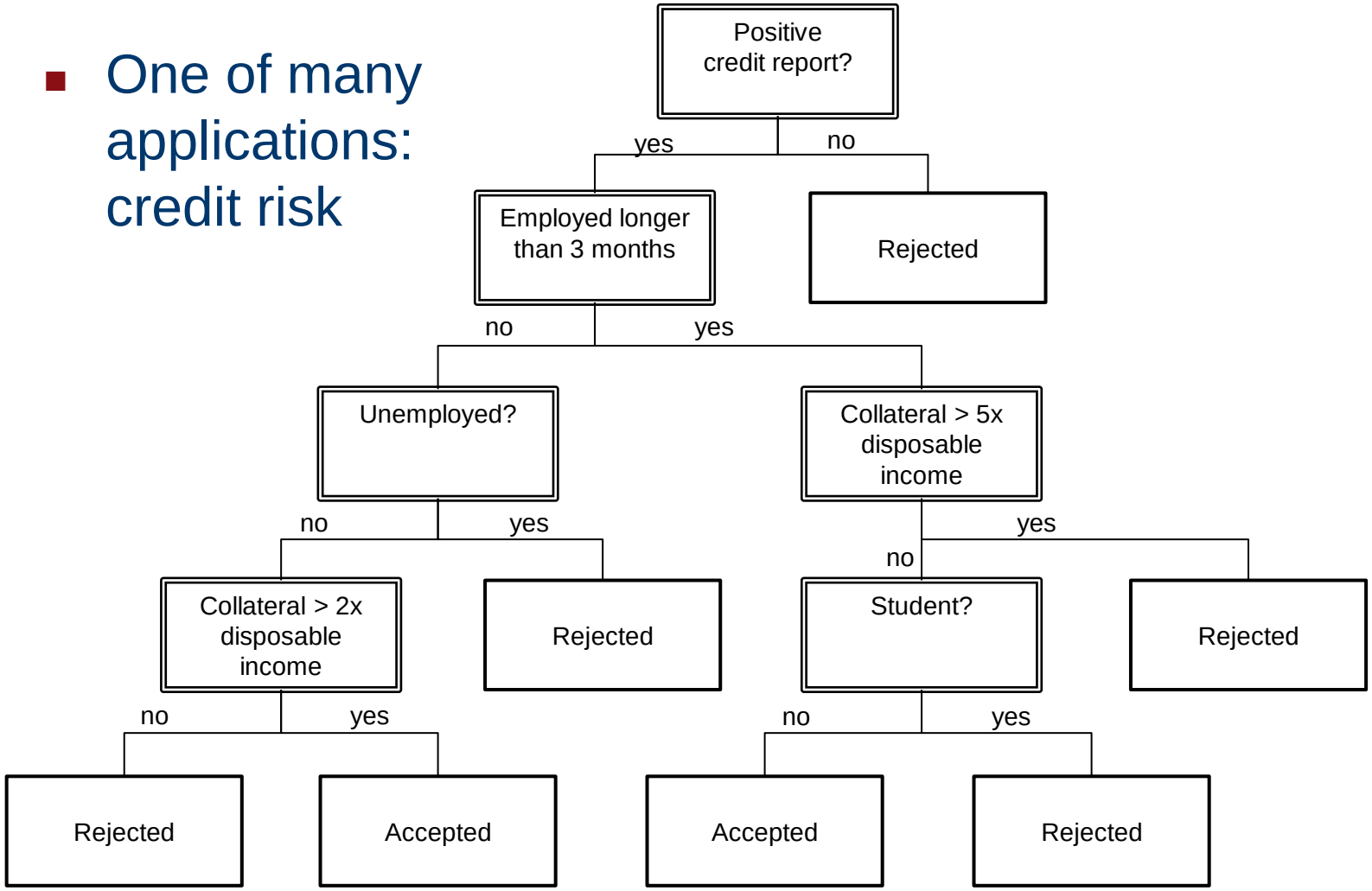
Universität Potsdam
Institut für Informatik
Lehrstuhl Maschinelles Lernen

Intelligent Data Analysis
Decision Trees

Paul Prasse, Niels Landwehr, Tobias Scheffer

Decision Trees

- One of many applications: credit risk



Decision Trees– Why?

- Simple to interpret.
- Provides a classification plus a justification for it.
  - “Rejected, because subject has been employed less than 3 months and has collateral < 2 x disposable income“.
- Can be learned from samples.
  - Simple learning algorithm.
  - Efficient, scalable.
- Classification and regression trees.
- Classification, regression & model trees often are components of complex (e.g., risk) models.

Classification

- Input: Instance (Object) $\mathbf{x} \in X$.
  - Instances are represented as a vector of attributes.
  - An instance is an assignment to the attributes.
  - Instances will also be called feature vectors.

$$\mathbf{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_m \end{pmatrix}$$

e.g. features like color, test results,…

- Output: Class $y \in Y$; finite set Y.
  - e.g., {accepted, rejected}; {spam, not spam}.
  - The class is also referred to as the target attribute.

Classifier learning

- Input: Training data.
  - $L = \langle (\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n) \rangle$
  - $\mathbf{x}_i = \begin{pmatrix} x_{i1} \\ \vdots \\ x_{in} \end{pmatrix}$
- Output: Classifier.
  - $f : X \to Y$

e.g. a decision tree:
path along the edges to a leaf
provides the classification

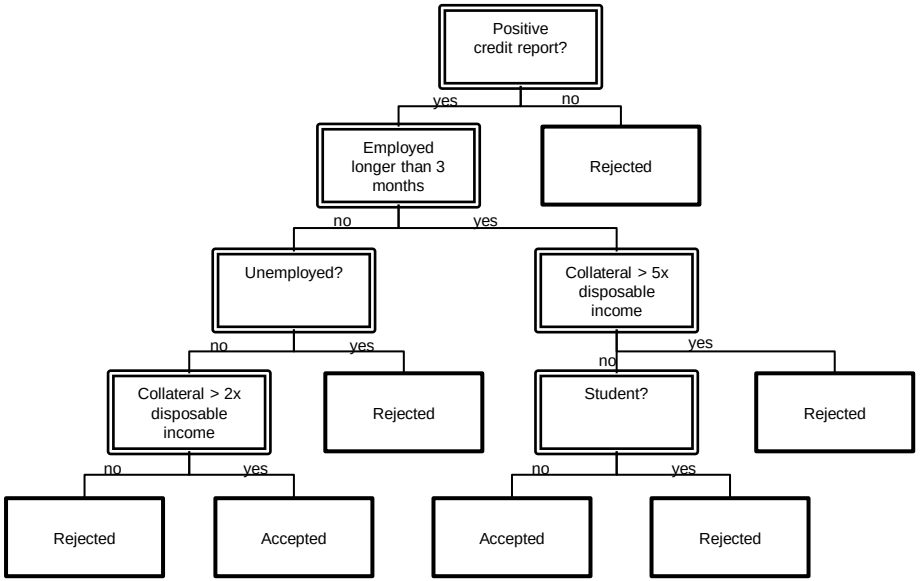
Regression

- Input: Instance (Object) $\mathbf{x} \in X$.
  - Instances are represented as vectors (bold letters) of attributes (italic letters).
  - An instance is an assignment of attribute values.
- Output: Function value $y \in Y$; continuous valued.

- Learning Problem: continuous-valued training data.
  - $L = \langle (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n) \rangle$
  - e.g. $\langle (\mathbf{x}_1, 3.5\ ), \ldots, (\mathbf{x}_n, -2.8\ ) \rangle$

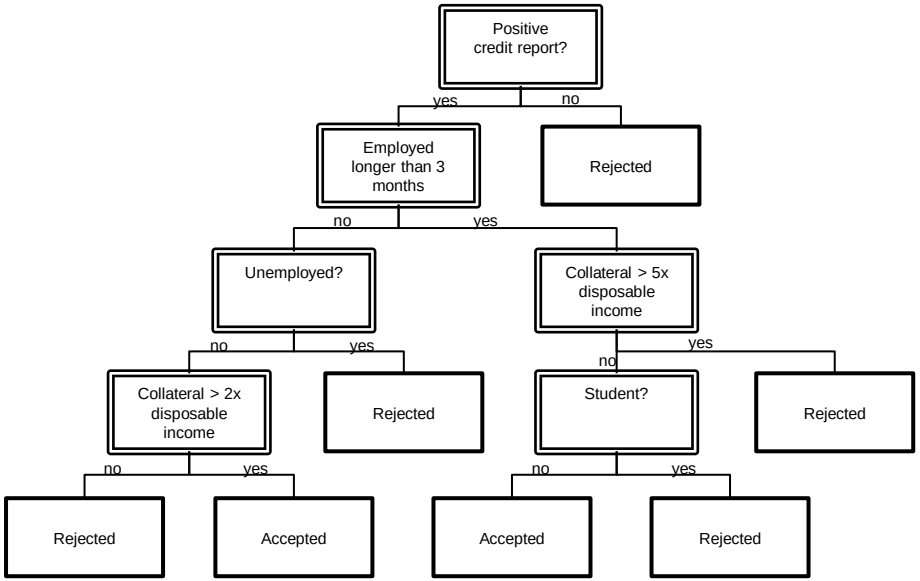
Decision Trees

- Test nodes: Test if the value of an attribute satisfies a condition; follow the branch corresponding to the outcome of this test.
- Terminal nodes: Provides a value to output.



Decision Trees

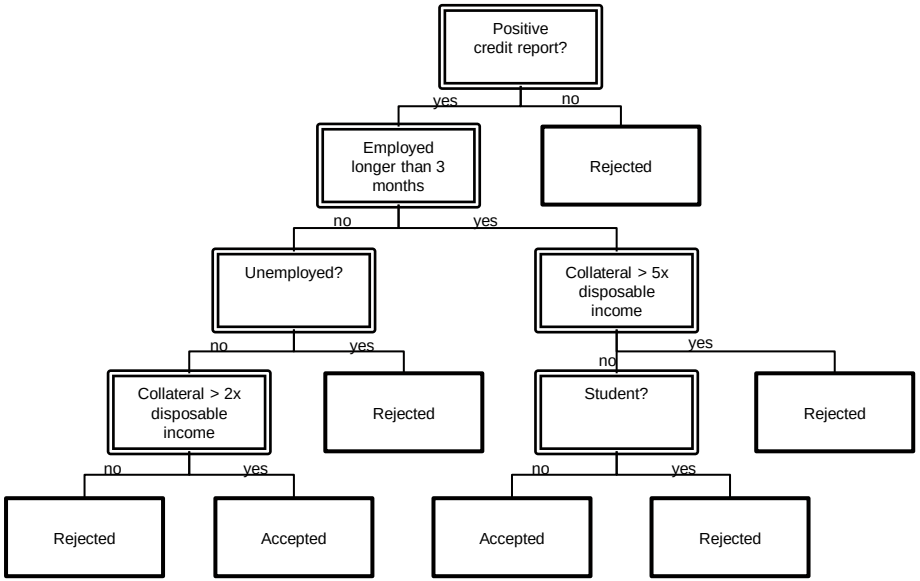
- Decision trees can be represented as logical operations / sentences, like:
  - $\wedge, \vee, \mathrm{XOR}$
  - $(A \wedge B) \vee (C \wedge \neg D \wedge E)$



Application of Decision Trees

- Test nodes: conduct a test, choose the appropriate branching, & proceed recursively.

- Terminal nodes: return the value as a class.



Application of Decision Trees

- Decision trees make sense if…
  - Instances can be described as attribute-value pairs.
  - Target attribute has a discrete range of values.
    - It is also possible to extend model trees to a range of continuous values.
  - Interpretability of predictions is desirable.

- Application areas:
  - Medical Diagnosis
  - Credit (Risk) Assessment
  - Object Recognition in Computer Vision

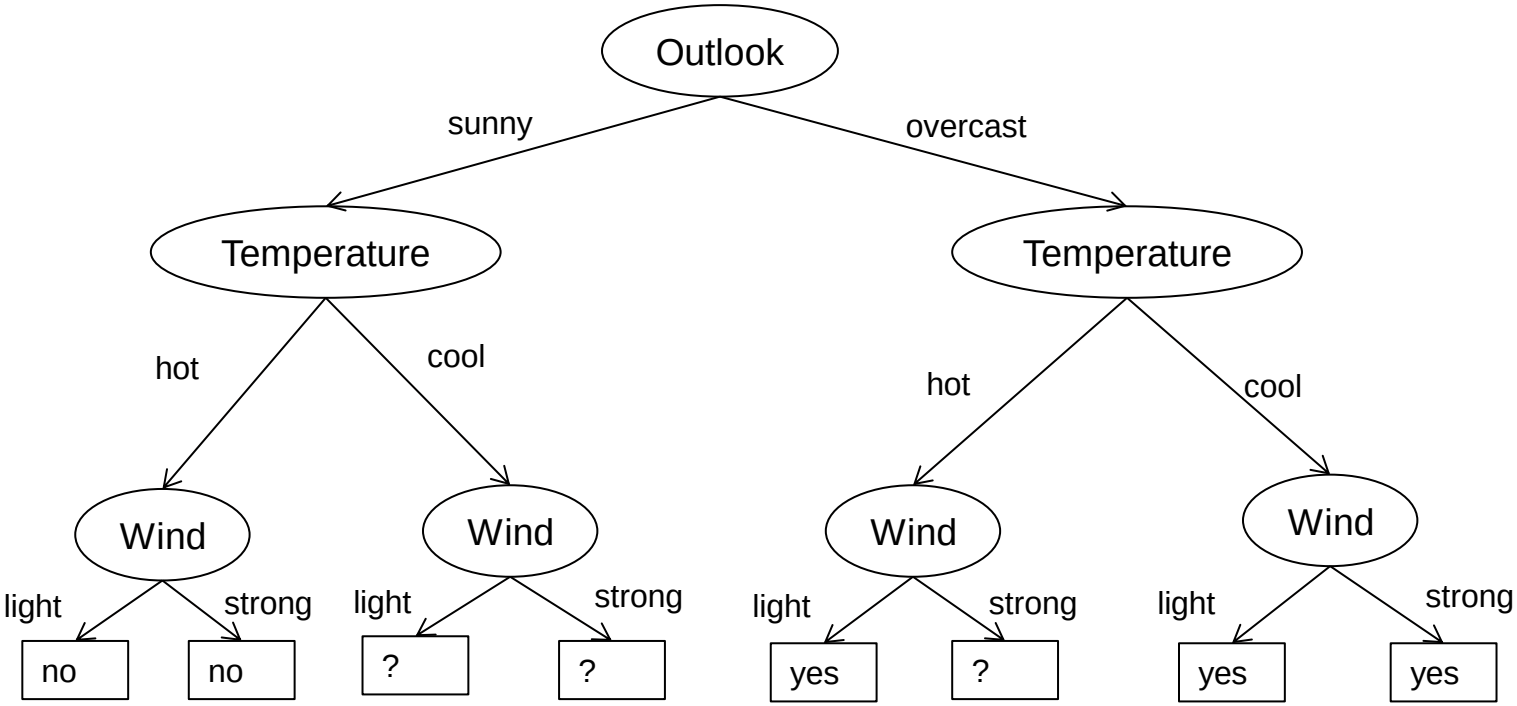
Learning of Decision Trees

Day	Outlook	Temperature	Humidity	Wind	Tennis?
1	sunny	hot	high	light	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	light	yes
4	rainy	mild	high	light	yes
5	rainy	cool	normal	light	yes
6	rainy	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	light	no
9	sunny	cool	normal	light	yes
10	rainy	mild	normal	light	yes

- Find the decision tree that at least provides the correct class for the training data.
- Trivial Way: create a tree that merely reproduces the training data.

Learning of Decision Trees

Day	Outlook	Temperature	Wind	Tennis?
1	sunny	hot	light	no
2	sunny	hot	strong	no
3	overcast	hot	light	yes
4	overcast	cool	strong	yes
5	overcast	cool	light	yes



Learning of Decision Trees

- A more elegant way: From the trees that are consistent with the training data, choose the smallest (as few nodes as possible).
- Small trees are good because:
  - they are easier to interpret;
  - they generalize better in many cases.
  - There are more samples per leaf node. Hence, the class decision in each leaf is supported by more samples.

Complete Search for the Smallest Tree

- How many decision trees are there?
  - Assume m binary attributes and two classes.

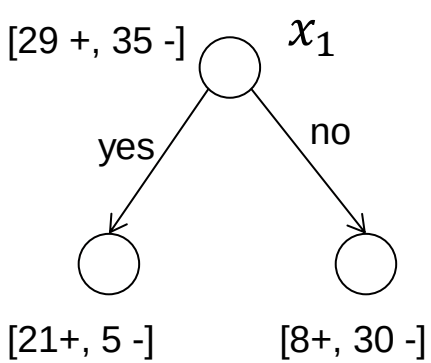
- What is the complexity of a complete search for the smallest tree?

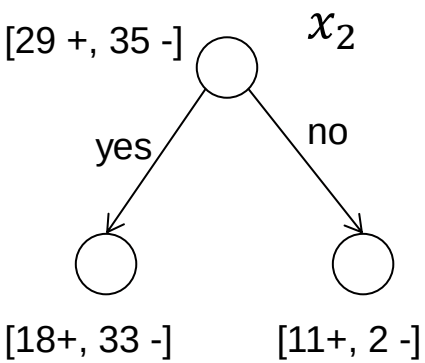
Learning of Decision Trees

- Greedy algorithm that finds a small tree (instead of the smallest tree) but is polynomial in the number of attributes.
- Idea for Algorithm?

Greedy Algorithm – Top-Down Construction

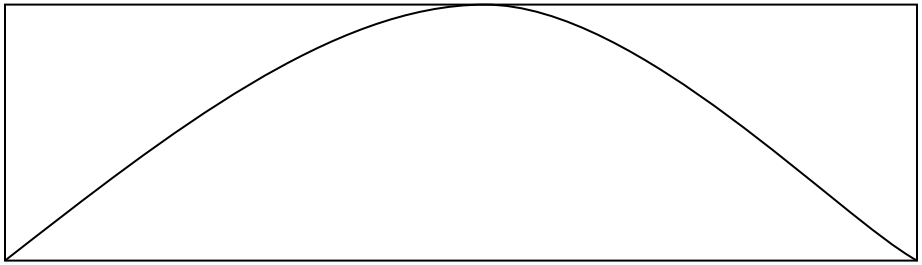
- So long as the training data are not perfectly classified:
  - Choose the "best" attribute A that separates the training data.
  - Create a new successor node for every value of attribute A and repartition the training data accordingly into the resulting tree.
- Problem: Which attribute is the best?





Split Criterium: Entropy

- Entropy = a measure of uncertainty.



- S is a set of training data.
- p_+ is the fraction of positive samples in S.
- p_- is the fraction of negative samples in S.
- Entropy measure the uncertainty in S (by +/-):

$$H(S) = -p_+ \log_2 p_+ - p_- \log_2 p_-$$

Split Criterium: Entropy

- $H(S)$ = Expected number of bits required to encode the target class for samples drawn randomly from the set S.
  - Optimal, shortest code

- Entropy of random variable y:
  - $H(y) = -\sum_v p(y = v) \log_2 p(y = v)$
- Empirical entropy of random variable y on data L:
  - $H(L, y) = -\sum_v \hat{p}_L(y = v) \log_2 \hat{p}_L(y = v)$

Split Criterium: Conditional Entropy

- Conditional entropy of random variable y given event $x = v$:

$$H_{|x=v}(y) = H(y|x = v)$$
$$= -\sum_{v'} p(y = v'|x = v)\log_2 p(y = v'|x = v)$$

- Empirical conditional entropy on data L:

$$H_{|x=v}(L, y) = H(L, y|x = v)$$
$$= -\sum_{v'} \hat{p}_L(y = v'|x = v)\log_2 \hat{p}_L(y = v'|x = v)$$

Split Criterium: Information Gain

- Entropy of the class variable: Uncertainty about the correct classification.
- Information Gain:
  - Mutual information of an attribute.
  - Reduction of the entropy in the class variable after the test of the value of this attribute.

$$IG(x) = H(y) - \sum_v p(x = v) H_{|x=v}(y)$$

- Information Gain on Data L:

$$IG(L, x) = H(L, y) - \sum_v \hat{p}_L(x = v) H_{|x=v}(L, y)$$

Example – Information Gain

IG = 0,05 IG = 0,46 IG = 0,05

Sample	x_1: Credit > 3 x Income?	x_2: Employed Longer than 3 Months?	x_3: Student?	y: Credit Repaid?
1	Yes	Yes	No	Yes
2	Yes	No	No	No
3	No	Yes	Yes	Yes
4	No	No	Yes	No
5	No	Yes	No	Yes
6	No	No	No	Yes

$$H(L,y) = -\sum_v \hat{p}_L(y=v)\log_2 \hat{p}_L(y=v)$$

≈0.92

$$IG(L,x) = H(L,y) - \sum_v \hat{p}_L(x=v)H_{|x=v}(L,y)$$

Example II – Information Gain

- $IG(L,x)$= Expected reduction in the uncertainty through the split using attribute x.
- Which split is better?

- $H(L,y) = -\left(\frac{29}{64}\log_2\frac{29}{64}+\frac{35}{64}\log_2\frac{35}{64}\right) = 0{,}99$
- $IG(L,x_1) = 0{,}99 - \left(\frac{26}{64}*H(L_{|x_1=ja},y) + \frac{38}{64}*H(L_{|x_1=nein},y)\right) = 0{,}26$
- $IG(L,x_2) = 0{,}99 - \left(\frac{51}{64}*H(L_{|x_2=ja},y) + \frac{13}{64}*H(L_{|x_2=nein},y)\right) = 0{,}11$

Information Gain / Gain Ratio

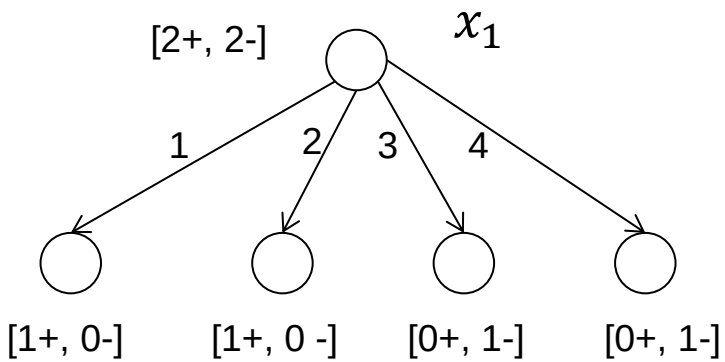
- Motivation:
  - Predicting whether a student will pass an exam.
  - How high is the information gain of the attribute “Matriculation number“?
  - The information content of this test is huge.

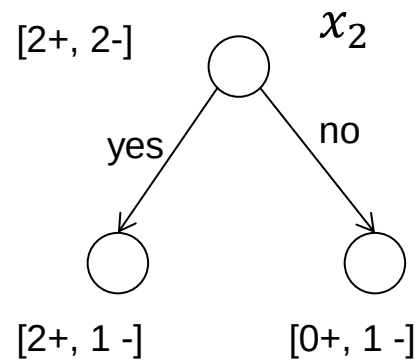
Information Gain / Gain Ratio

- Motivation:
  - Predicting whether a student will pass an exam.
  - How high is the information gain of the attribute "Matriculation number"?
  - The information content of this test is huge.
- Idea: Penalize information content of a test.
  - $GainRatio(L,x) = \frac{IG(L,x)}{SplitInfo(L,x)}$
  - $SplitInfo(L,x) = -\sum_v \frac{|L_{|x=v}|}{|L|} \log_2 \frac{|L_{|x=v}|}{|L|}$

Example: Info Gain Ratio

- Which split is better?





$IG(L, x_1) = 1$

$SplitInfo(L, x_1) = -4\left(\frac{1}{4}\log_2\frac{1}{4}\right) = 2$

$GainRatio(L, x_1) = 0{,}5$

$IG(L, x_2) = 1 - \left(\frac{3}{4} * 0{,}92\right) = 0{,}68$

$SplitInfo(L, x_2) = -\left(\frac{3}{4}\log_2\frac{3}{4} + \frac{1}{4}\log_2\frac{1}{4}\right) = 0{,}81$

$GainRatio(L, x_2) = 0{,}84$

Algorithm ID3

- Preconditions:
  - Classifier learning,
  - All attributes have a fixed, discrete range of values.
- Idea: recursive algorithm.
  - Choose the attribute that maximally reduces the uncertainty with respect to the target class
    - using Information Gain, Gain Ratio, or Gini Index
  - Then recursively call the algorithm by splitting the data according to the values of the chosen attribute.
  - Continue splitting a branch until all the samples remaining for that branch are all of the same class.
- Original reference:

  J.R. Quinlan: „Induction of Decision Trees". 1986

Algorithm ID3

- Input: $L = \langle(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\rangle$, available attributes = $(x_1, \dots, x_n)$

- If all samples in L have the same class y,
  - Return a terminal node with class y.
- If available attributes=∅,
  - Return a terminal node with the most prevalent class in L.
- Otherwise construct & return a test node:
  - Determine the best available attribute:

$$x_* = \underset{x_i \in available}{\operatorname{argmax}} \; IG(L, x_i)$$

  - For all values v_j of this attribute:
    - Split the training set, $L_j = \left\langle (\mathbf{x}_k, y_k) \in L \middle| x_{k*} = v_j \right\rangle$
    - Recursion: Branch for value $v_j = ID3\left(L_j, available \backslash x_*\right)$.

Algorithm ID3: Example

Sample	x_1: Credit > 3 x Income?	x_2: Employed Longer than 3 Months?	x_3: Student?	y: Credit Repaid?
1	Yes	Yes	No	Yes
2	Yes	No	No	No
3	No	Yes	Yes	Yes
4	No	No	Yes	No
5	No	Yes	No	Yes
6	No	No	No	Yes

- Input: $L = \langle(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\rangle$, available attributes = $(x_1, \dots, x_n)$

- If all samples in L have the same class y,
  - Return a terminal node with class y.
- If available attributes=$\emptyset$,
  - Return a terminal node with the most prevalent class in L.
- Otherwise construct & return a test node:
  - Determine the best available attribute:

$$x_* = \underset{x_i \in available}{\operatorname{argmax}} \; IG(L, x_i)$$

  - For all values v_j of this attribute:
    - Split the training set, $L_j = \left\langle(\mathbf{x}_k, y_k) \in L \middle| x_{k*} = v_j\right\rangle$
    - Recursion: Branch for value $v_j = ID3\left(L_j, available \backslash x_*\right)$.

Continuous Attributes

- ID3 chooses an attribute with the largest information content and then constructs a branch for every value of this attribute.
- Problem: that only works for discrete attributes.
- Attributes like size, income, & distance have infinitely many values.

- Ideas?

Continuous Attributes

- Idea: Binary decision trees with "≤"-tests.
- Problem: Infinitely many thresholds for binary tests.
- Idea: Only finitely many values occur in the training data.
- Example:
  - A continuous attribute has the following values:
    - 0,2; 0,4; 0,7; 0,9
  - Possible Splits:
    - $\leq 0{,}2; \leq 0{,}4; \leq 0{,}7; \leq 0{,}9$
- Other Possibilities:
  - Map the attribute into a discrete range of values.

Algorithm C4.5

- A further enhancement on ID3
- Improvements:
  - Allows for continuously-valued attributes
  - Allows for training data with missing attribute values
  - Allows for attributes with costs
  - Pruning
- Not the last version: see C5.0

- Original Reference:

  J.R. Quinlan: ”C4.5: Programs for Machine Learning”. 1993

Algorithm C4.5

- Input: $L = \langle(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\rangle$

- If all samples in L have the same class y,
  - Return a terminal node with class y.
- If all samples in L are identical:
  - Return a terminal node with the most prevalent class in L.
- Otherwise construct the best test node, by iterating over all attributes.
  - Discrete attributes: treated as in ID3.
  - Continuous attributes: $[x_* \leq v_*] = \underset{x_i \,\&\, v \in L}{\operatorname{argmax}} IG(L, [x_i \leq v])$
  - If the best attribute is discrete, recursively split like ID3.
  - If the best attribute is continuous, split the training set as:
    - $L_{left} = \langle(\mathbf{x}_k, y_k) \in L | x_{k*} \leq v_*\rangle$ & $L_{right} = \langle(\mathbf{x}_k, y_k) \in L | x_{k*} > v_*\rangle$
    - Recursion: left branch= $C4.5(L_{left})$, right branch= $C4.5(L_{right})$

Information Gain for Continuous Attributes

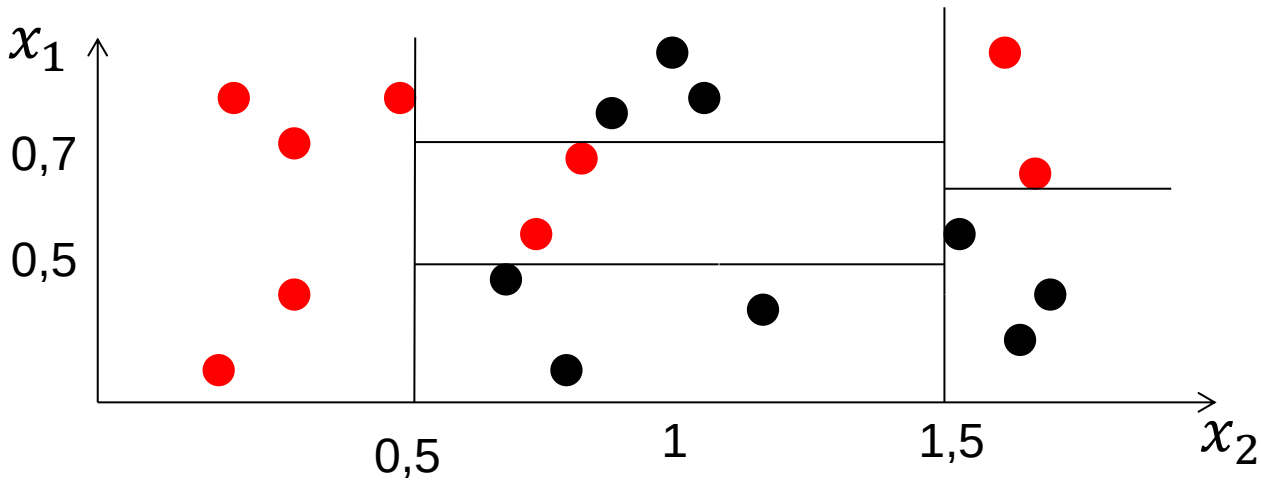
- Information Gain of a Tests "$[x \leq v]$":

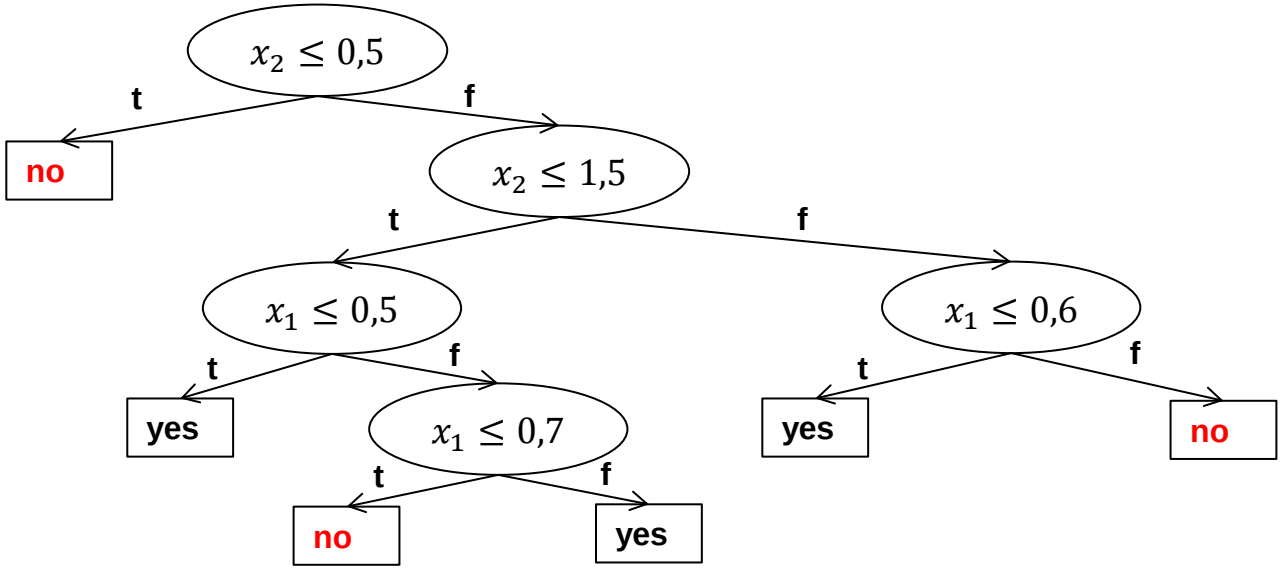
$$IG([x \leq v]) = H(y) - p([x \leq v])H_{[x \leq v]}(y) - p([x > v])H_{[x > v]}(y)$$

- Empirical Information Gain:

$$IG(L, [x \leq v]) = H(L, y) - \hat{p}_L([x \leq v])H_{[x \leq v]}(L, y)$$
$$-\hat{p}_L([x > v])H_{[x > v]}(L, y)$$

C4.5: Example


x_1
0,7
0,5
0,5
1
1,5
x_2


$x_2 \leq 0{,}5$
t
f
no
$x_2 \leq 1{,}5$
t
f
$x_1 \leq 0{,}5$
$x_1 \leq 0{,}6$
t
f
t
f
yes
$x_1 \leq 0{,}7$
yes
no
t
f
no
yes

Pruning

- Problem: Leaf nodes that only are supported by a single (or very few) samples, often do not provide a good classification.
- Pruning: Remove test nodes whose leaves have less than a minimum number of samples.
- This policy results in leaf nodes labeled with the most prevalently occurring class.

Pruning with a Threshold

- For all leaf nodes: If fewer than r training samples are placed into a leaf node,
  - Remove the parent test node of this leaf.
  - Replace it with a new leaf node that predicts the majority class from its training data.

- Regularization parameter r.
- This parameter is adjusted via cross validation.

Reduced Error Pruning

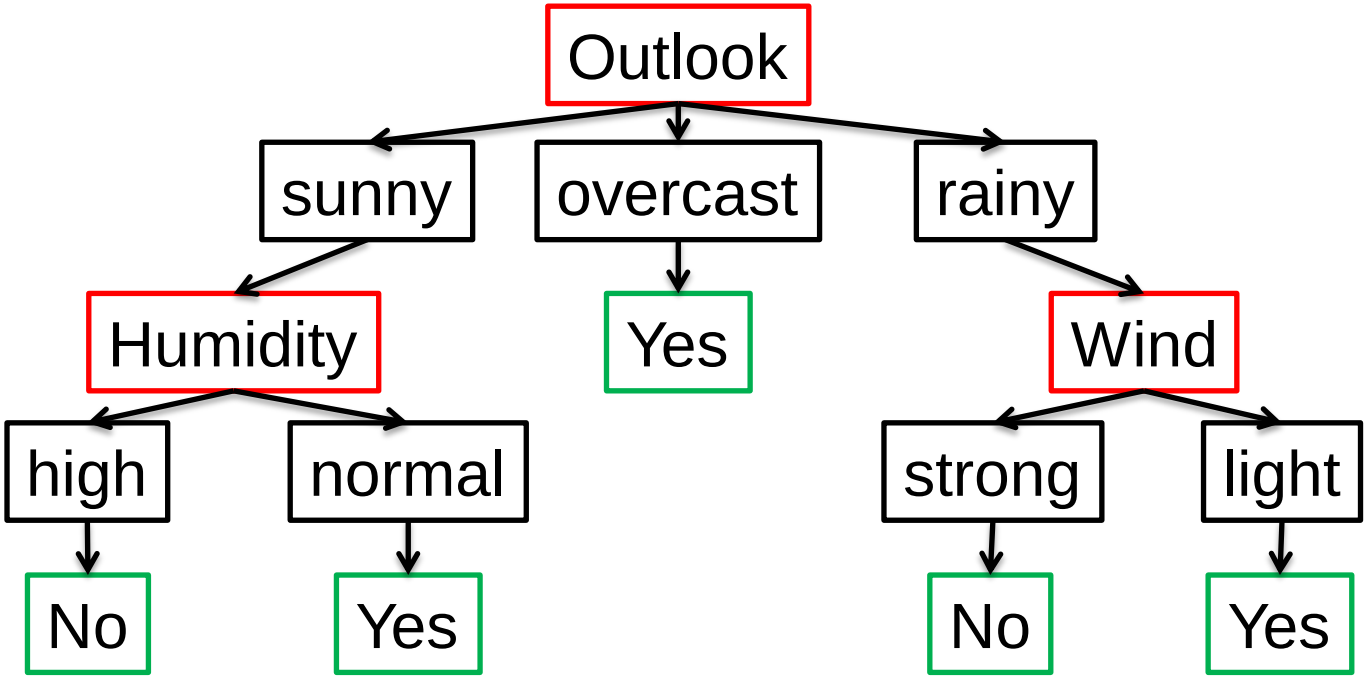
- Splitting of the training data into a training set and a pruning validation set.
- After the construction of the tree with the training set:
  - Attempt to merge two leaf nodes by removing their adjoining test node,
  - Continue as long as the error rate is reduced with respect to the pruning validation set.

Conversion of Trees into Rules

- Path through the tree: yields the conditions for the rule
- Class: specifies the rules conclusion

- Pruning of Rules: Test which conditions can be omitted without increasing the error rate.

Conversion of Trees into Rules



$R_1 = If\ (Outlook = sunny) \wedge (Humidity = high), then\ don't\ play\ tennis$

$R_2 = If\ (Outlook = sunny) \wedge (Humidity = normal), then\ play\ tennis$

$R_3 = If\ (Outlook = overcast), then\ play\ tennis$

$R_4 = If\ (Outlook = rainy) \wedge (Wind = strong), then\ don't\ play\ tennis$

$R_5 = If\ (Outlook = rainy) \wedge (Wind = light), then\ play\ tennis$