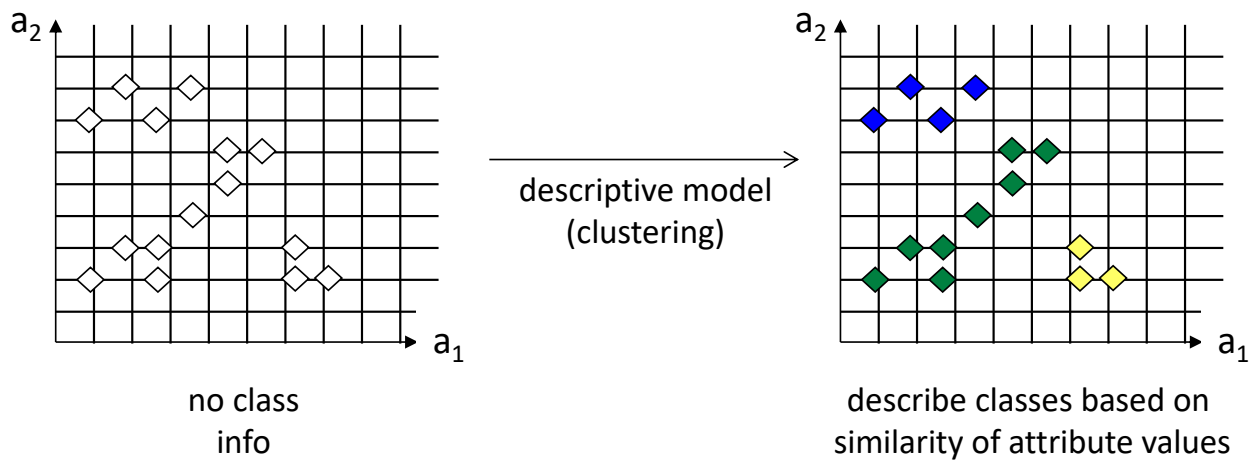


# **4. CLASSIFICATION**

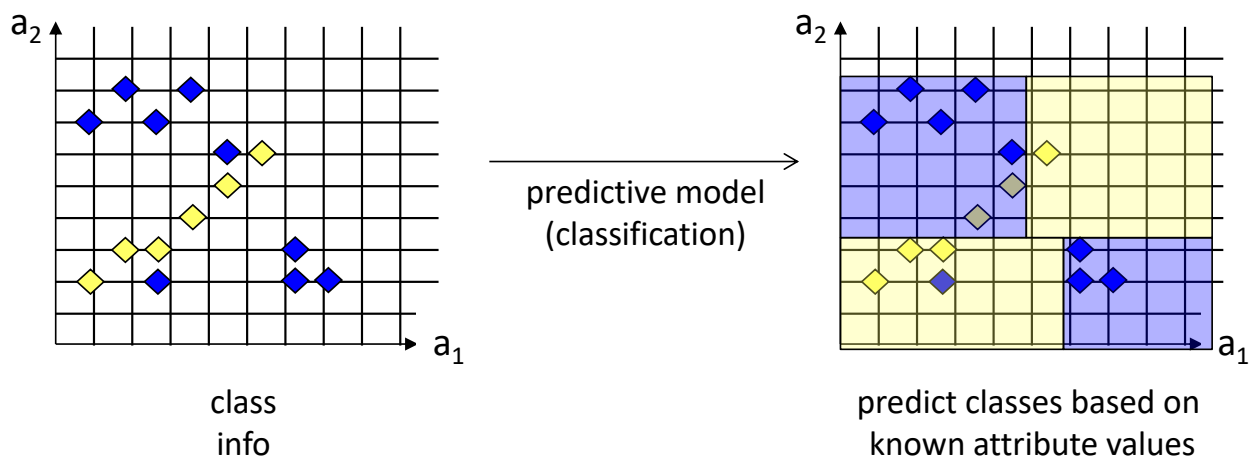
# Clustering and Classification

Given a dataset of *objects* described by *attributes*, build a model that assigns objects to a *class (or label)*



# Clustering and Classification

Given a dataset of *objects* described by *attributes*, build a model that assigns objects to a *class*



# Classification Problem

**Input:** set of objects with categorical/numerical attributes and one class label

**Output:** A model that returns the class label given the object attributes

- Model is a function represented as rules, decision trees, formulae

Classification belongs to *supervised* ML

- Objects have class information

# Classification: General Approach

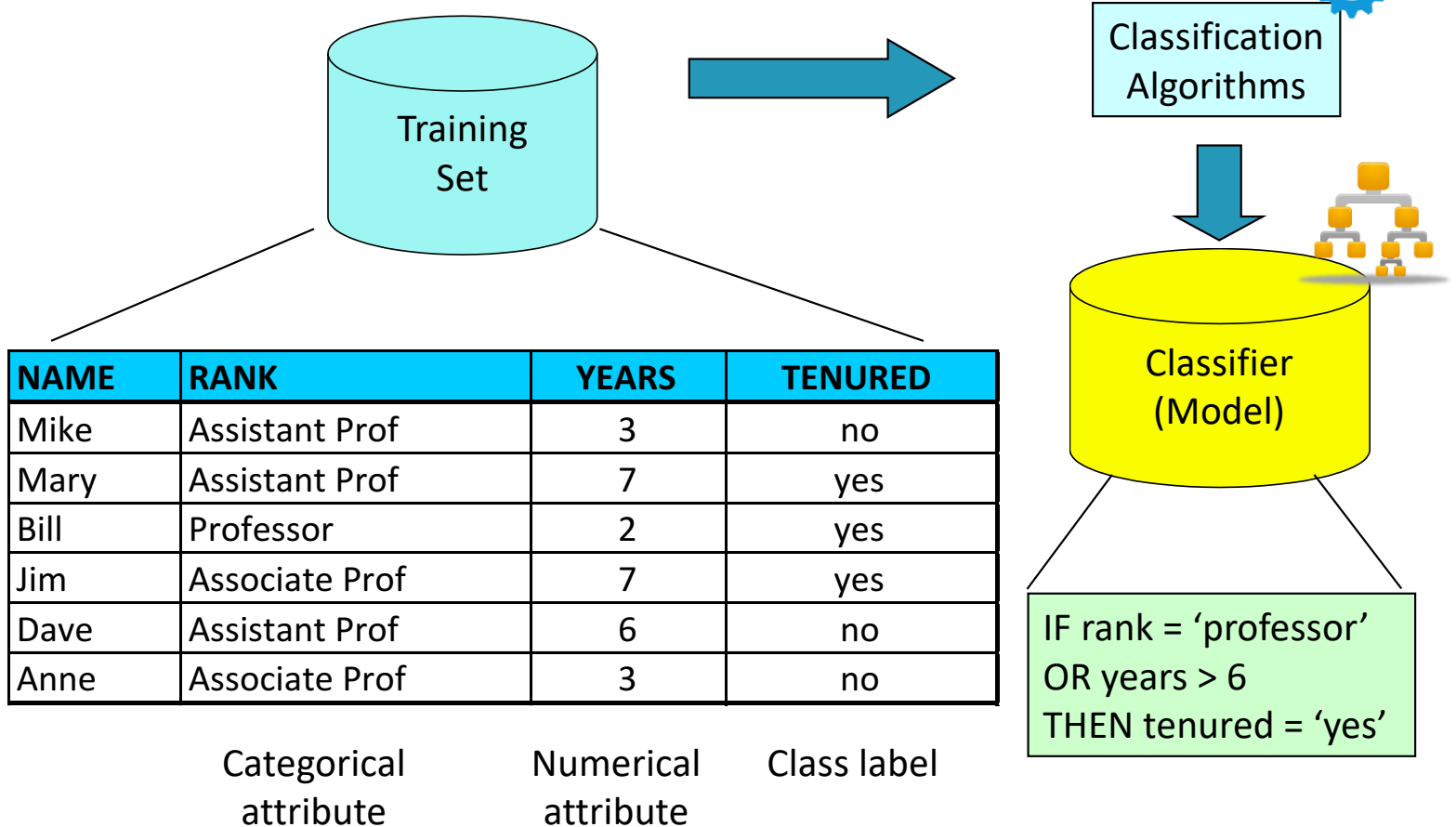
Model is learnt from a set of objects with known labels: **training set**

The quality of the model is evaluated by comparing the predicted class labels with those from a set of objects with known labels: **test set**

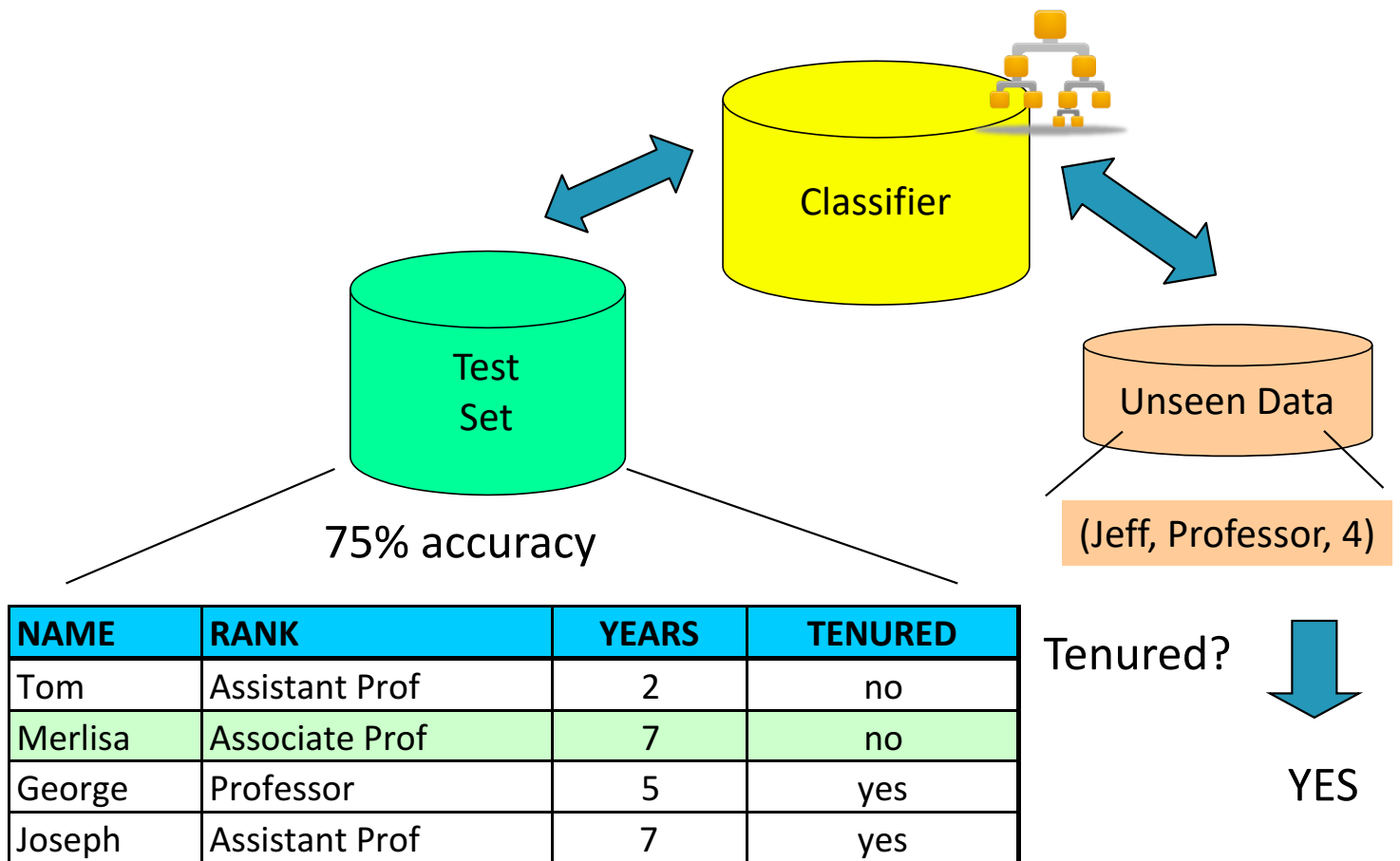
- Test set is independent of training set, otherwise over-fitting will occur

The model is applied to data with unknown labels: **prediction**

# Classification: Training



# Classification: Model Test and Usage



# Classification: Problem Formulation

## Problem

Given a database  $D$  with  $n$  data items described by  $d$  categorical/numerical attributes and one categorical attribute (class label  $C$ )

## Find

A function  $f: X^d \rightarrow C$

rules  
decision tree  
formula

## Such that

classifies *accurately* the items in the *training* set  
*generalises* well for the (unknown) items in the *test* set



# Characteristics of Classification Methods

Predictive accuracy

Speed and scalability

- Time to build the model
- Time to use the model
- In memory vs. on disk processing

Robustness

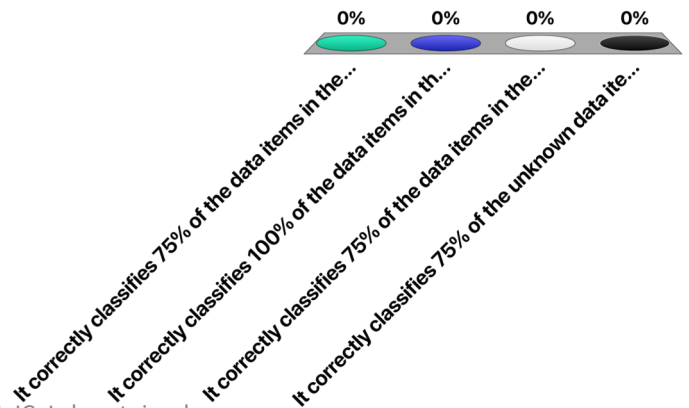
- Handling noise, outliers and missing values

Interpretability

- Understanding the model and its decisions (black box) vs. white box
- Compactness of the model

# If a classifier has 75% accuracy, it means that ...

- A. It correctly classifies 75% of the data items in the training set
- B. It correctly classifies 100% of the data items in the training set but only 75% in the test set
- C. It correctly classifies 75% of the data items in the test set
- D. It correctly classifies 75% of the unknown data items

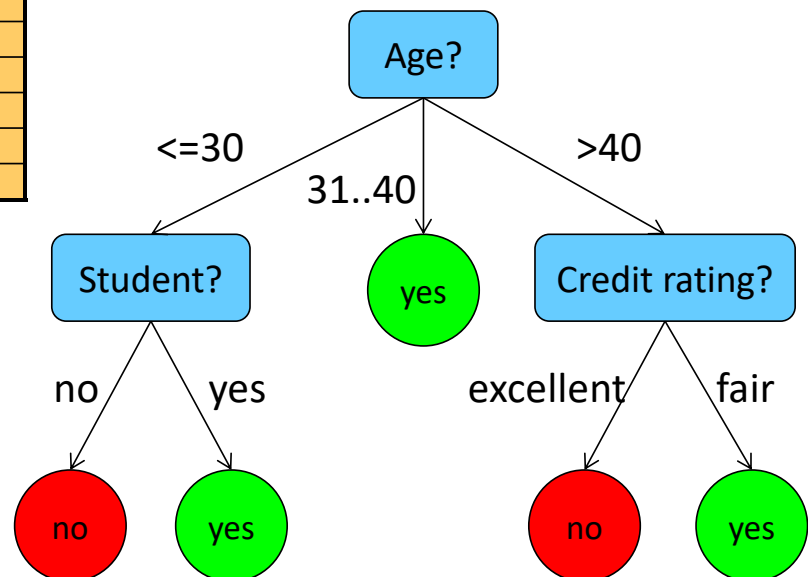


# Decision Trees

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31...40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31...40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31...40	medium	no	excellent	yes
31...40	high	yes	fair	yes
>40	medium	no	excellent	no

some columns are not used

- Nodes are tests on a single attribute
- Branches are attribute values
- Leaves are marked with class labels



# Decision Tree Induction: Algorithm

Tree construction (top-down divide-and-conquer strategy)

- At the beginning, all training samples belong to the root
- Examples are partitioned recursively based on a selected “most discriminative” attribute
- Discriminative power determined based on information gain (ID3/C4.5)

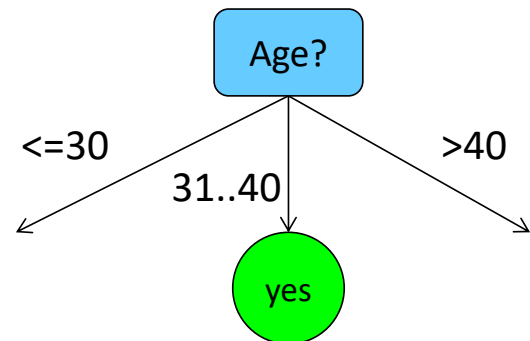
Partitioning stops if todo: understand all

- All samples belong to the same class → assign the class label to the leaf
- There are no attributes left → majority voting to assign the class label to the leaf
- There are no samples left

# Example: Decision Tree Induction

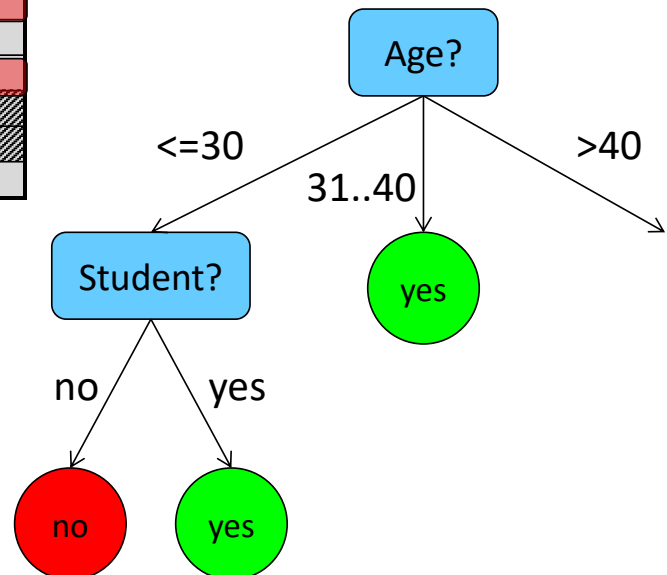
age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31...40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31...40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31...40	medium	no	excellent	yes
31...40	high	yes	fair	yes
>40	medium	no	excellent	no

buys\_computer is always yes if  
31 < age < 40



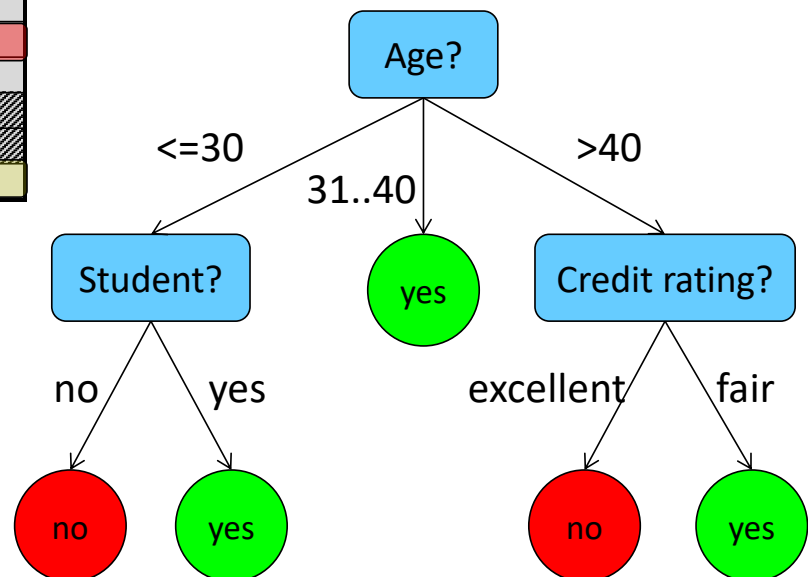
# Example: Decision Tree Induction

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31..40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31..40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31..40	medium	no	excellent	yes
31..40	high	yes	fair	yes
>40	medium	no	excellent	no



# Example: Decision Tree Induction

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31..40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31..40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31..40	medium	no	excellent	yes
31..40	high	yes	fair	yes
>40	medium	no	excellent	no



# Attribute Selection

At a given branch in the tree, the set of samples  $S$  to be classified has  $P$  positive and  $N$  negative instances

The entropy of the set  $S$  is Entropy of set is the entropy of the label

$$H(P, N) = -\frac{P}{P+N} \log_2 \frac{P}{P+N} - \frac{N}{P+N} \log_2 \frac{N}{P+N}$$

## Note

– If  $P=0$  or  $N=0$

$H(P, N) = 0 \rightarrow$  no uncertainty

– If  $P=N$

$H(P, N) = 1 \rightarrow$  maximal uncertainty



Entropy always computed wrt label

## Attribute Selection: Example

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31...40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31...40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31...40	medium	no	excellent	yes
31...40	high	yes	fair	yes
>40	medium	no	excellent	no

$$H_S = H(9, 5) = 0.94$$

Entropy of set = entropy of label (buys\_computer)

Age [ $\leq 30$ ]  $H(2, 3) = 0.97$

Age [31...40]  $H(4, 0) = 0$

Age [ $> 40$ ]  $H(3, 2) = 0.97$

Income [high]  $H(2, 2) = 1$

Income [med]  $H(4, 2) = 0.92$

Income [low]  $H(3, 1) = 0.81$

Student [yes]  $H(6, 1) = 0.59$

Student [no]  $H(3, 4) = 0.98$

Rating [fair]  $H(6, 2) = 0.81$

Rating [exc]  $H(3, 3) = 1$

Entropies are all computed wrt label

# Attribute Selection: Example

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31...40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31...40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31...40	medium	no	excellent	yes
31...40	high	yes	fair	yes
>40	medium	no	excellent	no

$$H_S = H(9, 5) = 0.94$$

$$H_{\text{Age}} = p([<=30]) \cdot H(2, 3) + p([31...40]) \cdot H(4, 0) + p([>40]) \cdot H(3, 2) = \\ = 5/14 \cdot 0.97 + 4/14 \cdot 0 + 5/14 \cdot 0.97 = 0.69$$

$$H_{\text{Income}} = p([high]) \cdot H(2, 2) + p([med]) \cdot H(4, 2) + p([low]) \cdot H(3, 1) = \\ = 4/14 \cdot 1 + 6/14 \cdot 0.92 + 4/14 \cdot 0.81 = 0.91$$

$$H_{\text{Student}} = p([yes]) \cdot H(6, 1) + p([no]) \cdot H(3, 4) = 7/14 \cdot 0.59 + 7/14 \cdot 0.98 = 0.78$$

$$H_{\text{Rating}} = p([fair]) \cdot H(6, 2) + p([exc]) \cdot H(3, 3) = 8/14 \cdot 0.81 + 6/14 \cdot 1 = 0.89$$

conditional entropy =  
probability of being in that subgroup x entropy of subgroup

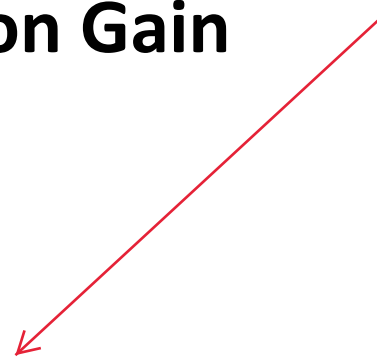
we split using the attribute with the smallest uncertainty.

conditional entropy =  
probability of being in that subgroup x entropy of subgroup

## Attribute Selection: Information Gain

Attribute A partitions S into  $S_1, S_2, \dots, S_v$

Entropy of attribute A is

$$H(A) = \sum_{i=1}^v \frac{P_i + N_i}{P + N} H(P_i, N_i)$$


The information gain obtained by splitting S using A is

$$Gain(A) = H(P, N) - H(A)$$

$H(P, N)$  = total / true  
uncertainty  
 $H(A)$  = uncertainty of  
category

$$Gain(Age) = 0.94 - 0.69 = 0.25$$

$$Gain(Income) = 0.94 - 0.91 = 0.03$$

$$Gain(Student) = 0.94 - 0.78 = 0.16$$

$$Gain(Rating) = 0.94 - 0.89 = 0.05$$

← split on age

if  $H(A) = H(P, N)$ , then gain = 0

if the uncertainty of the attribute is  
equal to the uncertainty of S, then  
gain is 0

the attribute has no impact on  
classification

Information Gain = Entropy of set before splitting - Entropy of set we split at

- Entropy of set before splitting is a "constant"
- We wanna reduce uncertainty so we choose the splitting attribute that has the lowest entropy

Given the distribution of positive and negative samples for attributes  $A_1$  and  $A_2$ , which is the best attribute for splitting?

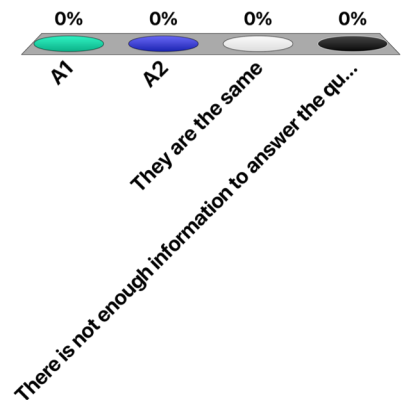
$A_1$	P	N
a	2	2
b	4	0
$A_2$	P	N
x	3	1
y	3	1

A.  $A_1$

B.  $A_2$

C. They are the same

D. There is not enough information to answer the question



# Pruning

The construction phase does not filter out noise → **overfitting**

## Pruning strategies

- Stop partitioning a node when large majority of samples is positive or negative, i.e.,  $N/(N+P)$  or  $P/(N+P) > 1 - \epsilon$
- Build the full tree, then replace nodes with leaves labelled with the majority class, if classification accuracy does not change
- Apply Minimum Description Length (MDL) principle

# Minimum Description Length Pruning

Let  $M_1, M_2, \dots, M_n$  be a list of candidate models (i.e., trees). The best model is the one that minimizes

$$L(M) + L(D | M)$$

where

- $L(M)$  is the length in bits of the description of the model (#nodes, #leaves, #arcs ...)
- $L(D | M)$  is the length in bits of the description of the data when encoded with the model (#misclassifications)

# Extracting Classification Rules from Trees

Represent the knowledge in the form of IF-THEN rules

- One rule is created for each path from the root to a leaf
- Each attribute-value pair along a path forms a conjunction
- The leaf node holds the class prediction

Rules are easier for humans to understand

## Example

IF <i>age</i> = " $\leq 30$ " AND <i>student</i> = "no"	THEN <i>buys_computer</i> = "no"
IF <i>age</i> = " $\leq 30$ " AND <i>student</i> = "yes"	THEN <i>buys_computer</i> = "yes"
IF <i>age</i> = "31...40"	THEN <i>buys_computer</i> = "yes"
IF <i>age</i> = " $> 40$ " AND <i>credit_rating</i> = "excellent"	THEN <i>buys_computer</i> = "yes"
IF <i>age</i> = " $> 40$ " AND <i>credit_rating</i> = "fair"	THEN <i>buys_computer</i> = "no"

# Decision Trees: Continuous Attributes

With continuous attributes we can not have a separate branch for each value

- use **binary decision trees**

## Binary decision trees

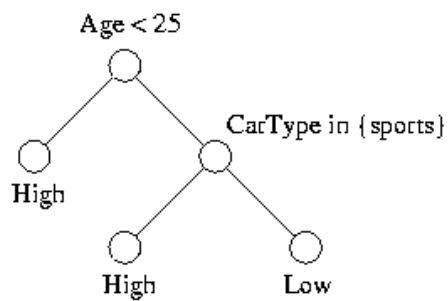
- For continuous attributes  $A$  a split is defined by  $\text{val}(A) < X$
- For categorical attributes  $A$  a split is defined by a subset  $X \subseteq \text{domain}(A)$



# Example: Binary Decision Tree

<i>rid</i>	Age	Car Type	Risk
0	23	family	High
1	17	sports	High
2	43	sports	High
3	68	family	Low
4	32	truck	Low
5	20	family	High

(a) Training Set



(b) Decision Tree

# Splitting Continuous Attributes

## Approach

- Sort the data according to attribute value
- Determine the value of X which maximizes information gain by scanning through the data items

X is the continuous attribute (eg age)

Only if the class label changes, a relevant decision point exists

<b>0</b>	<b>1</b>	<b>2</b>	<b>4</b>	<b>6</b>	<b>7</b>	<b>8</b>	<b>10</b>
P	P	P	N	N	N	P	P

Possible split points



Top Row Example: Age, we split at 25 because all labels are high

- Here, we need to decide whether to split using age or car type.
- For each attribute, we find the interval that maximizes information gain
  - We pick the attribute that results in the highest information gain

We actually should also compute between (32,43) and (43,58)

# Example

Attribute List

Age	Class	tid
17	High	1
20	High	5
23	High	0
32	Low	4
43	High	2
68	Low	3

Position of cursor in scan

← position 0

← position 3

← position 6

cursor position 0:

	H	L
C <sub>below</sub>	0	0
C <sub>above</sub>	4	2

cursor position 3:

	H	L
C <sub>below</sub>	3	0
C <sub>above</sub>	1	2

cursor position 6:

	H	L
C <sub>below</sub>	4	2
C <sub>above</sub>	0	0

$$H(P, N) = H(4, 2) = 0.918$$

$$H(A) = 0 + \frac{1}{2} H(1, 2) = 0.459$$

Gain = H(P, N) – H(A) = 0.459

Attribute List

Car Type	Class	tid
family	High	0
sports	High	1
sports	High	2
family	Low	3
truck	Low	4
family	High	5

Count Matrix

	H	L
family	2	1
sports	2	0
truck	0	1

splitting to {sports} and {family, truck}

$$H(A) = 0 + \frac{2}{3} H(2, 2) = 0.666$$

Gain = H(P, N) – H(A) = 0.251

# Scalability of Continuous Attribute Splits

## Naive implementation

- At each step the data set is split in subsets that are associated with a tree node

## Problem

- For evaluating which attribute to split, data needs to be sorted according to these attributes
- Becomes dominating cost

# Scalability of Continuous Attribute Splits

Idea: Presorting of data and maintaining order throughout tree construction

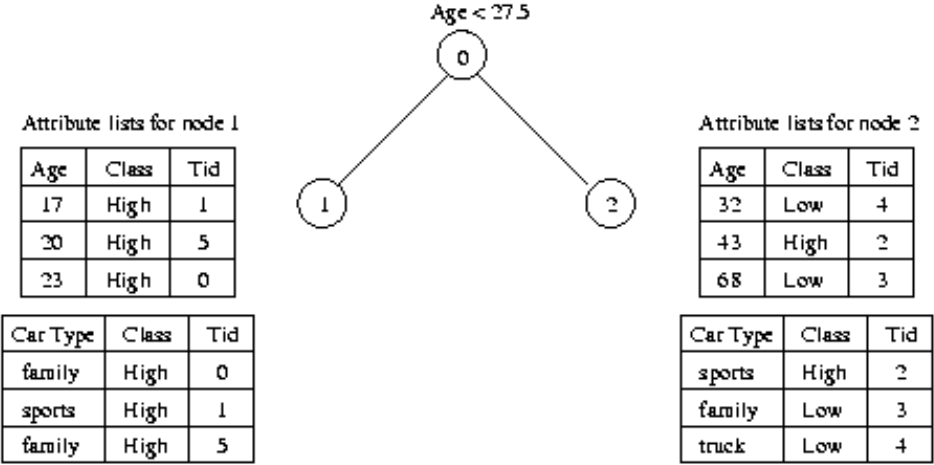
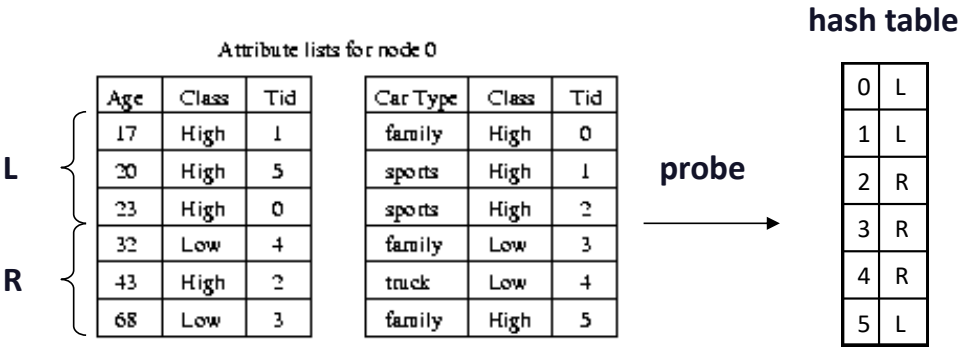
- Requires separate sorted attribute tables for each attribute

Updating attribute tables

- Attribute used for split: splitting attribute table straightforward
- Other attributes
  - Build Hash Table associating tuple identifiers (TIDs) of data items with partitions
  - Select data from other attribute tables by scanning and probing the hash table

# Example

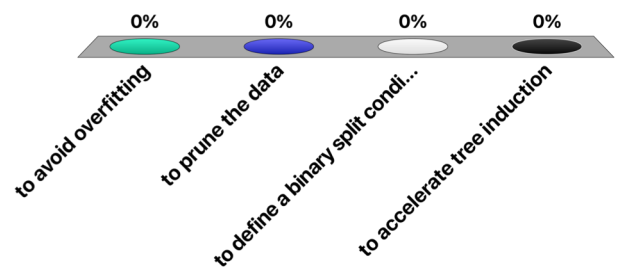
- scan table to see if it belongs to L node
- get its hash value
- index from car type table



age hash table

# When splitting a continuous attribute, its values need to be sorted ...

- A. to avoid overfitting
- B. to prune the data
- C. to define a binary split condition
- D. to accelerate tree induction



# Characteristics of Decision Tree Induction

## Strengths

- Automatic feature selection
- Minimal data preparation
- Non-linear model
- Easy to interpret and explain

## Weaknesses

- Sensitive to small perturbation in the data
- Tend to overfit
- Have to be re-trained from scratch with new data



# Decision Tree Induction: Properties

Model: flow-chart like tree structure

Score function: classification accuracy

Optimisation: top-down tree construction + pruning

Data Management: avoiding sorting during splits

# Classification Algorithms

Decision tree induction is a (well-known) example of a classification algorithm

## Alternatives

- Basic methods: Naïve Bayes, kNN, logistic regression, ..
- Ensemble methods: random forest, gradient boosting, ...
- Support vector machines
- Neural networks: CNN, rNN, LSTM, ...

After a weak learner is added, the data are reweighted: examples that are misclassified gain weight and examples that are classified correctly lose weight. Thus, future weak learners focus more on the examples that previous weak learners misclassified.

## Ensemble Methods

### Idea

- Take a collection of simple or **weak** learners
- Combine their results to make a single, **strong** learner

### Types

- **Bagging:** train learners in parallel on different samples of the data, then combine outputs through voting or averaging sampling with replacement
- **Stacking:** combine model outputs using a second-stage learner like linear regression
- **Boosting:** train learners on the filtered output of other learners - give higher weight to points that have been wrongly sampled

# Random Forests

Learn  $K$  different decision trees from independent samples of the data (bagging)

- vote between different learners, so models should not be too similar

Aggregate output: majority vote

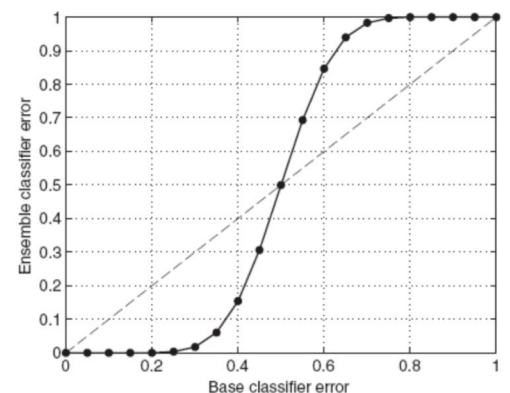
# Why do Ensemble Methods Work?

Assume there are 25 base classifiers

- Each classifier has error rate = 0.35
- Assume classifiers are independent

Probability that the ensemble classifier makes a wrong prediction

$$P(\text{wrong prediction}) = \sum_{i=13}^{25} \binom{25}{i} \varepsilon^i (1 - \varepsilon)^{25-i} = 0.06$$



Tan, Steinbach, Kumar

# Sampling Strategies

## Two sampling strategies

### Sampling data

- select a subset of the data → Each tree is trained on different data

### Sampling attributes

- select a subset of attributes → corresponding nodes in different trees (usually) don't use the same feature to split

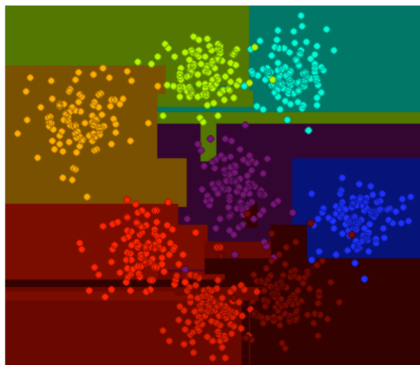
# Random Forests: Algorithm

1. Draw  $K$  bootstrap **samples of size  $N$**  from original dataset, with replacement (bootstrapping) all attributes taken
2. While constructing the decision tree, select a random set of  **$m$  attributes** out of the  $p$  attributes available to infer split (feature bagging) some attributes taken

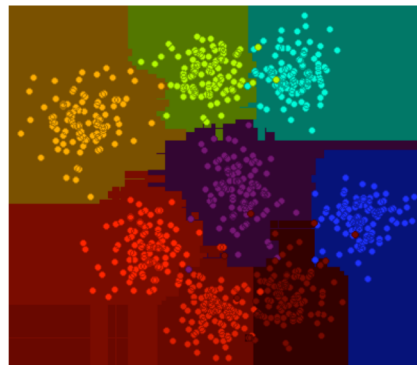
## Typical parameters

- $m \approx \sqrt{p}$ , or smaller
- $K \approx 500$

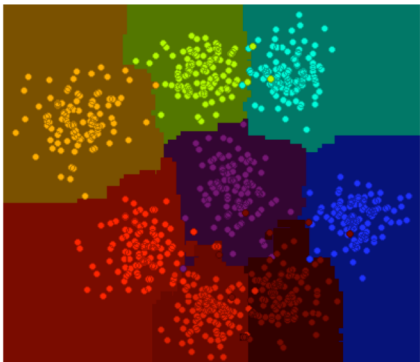
# Illustration of Random Forests



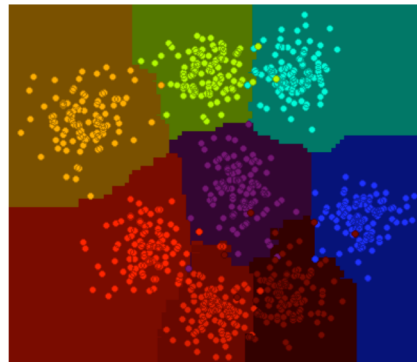
1 rCART



10 rCARTs



100 rCARTs



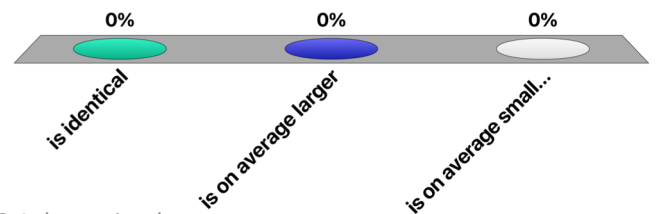
500 rCARTs



The computational cost for constructing a RF with K as compared to constructing K decision trees on the same data

- A. is identical
- B. is on average larger
- C. is on average smaller

we use lesser attributes. lesser branches



# Characteristics of Random Forests

## Strengths

- Ensembles can model extremely complex decision boundaries without overfitting
- Probably the most popular classifier for **dense data** ( $\leq$  a few thousand features)
- Easy to implement (train a lot of trees)
- Parallelizes easily, good match for MapReduce

# Characteristics of Random Forests

## Weaknesses

- Deep Neural Networks generally do better
- Needs many passes over the data – at least the max depth of the trees
- Relatively easy to overfit – hard to balance accuracy/fit tradeoff

# References

## Textbook

- Jiawei Han, Data Mining: concepts and techniques, Morgan Kaufman, 2000, ISBN 1-55860-489-8

## References

- Leo Breiman (2001) "Random Forests" Machine Learning, 45, 5-32.
- Shafer, John, Rakesh Agrawal, and Manish Mehta. "SPRINT: A scalable parallel classifier for data mining." *Proc. 1996 Int. Conf. Very Large Data Bases*. 1996.