Data Mining

Classification

- Part 1 -

Outline

1. What is Classification?

– Goal: Previously unseen records should be assigned a class from a given set of classes as accurately as possible.

– Approach:
  - Given a collection of records (*training set*)
    - each record contains a set of *attributes*
    - one of the attributes is the *class (label)* that should be predicted.
  - Learn a *model* for the class attribute as a function of the values of other attributes.

– Variants:
  - single-class problems (class labels e.g. true/false or fraud/no fraud)
  - multi-class problems (class labels e.g. low, medium, high)

Introduction to Classification

A Couple of Questions:

- What is this?
- Why do you know?
- How have you come to that knowledge?

Introduction to Classification

- Goal: Learn a model for recognizing a concept, e.g. trees
- Training data:

"tree"

"tree"

"tree"

"not a tree"

"not a tree"

"not a tree"

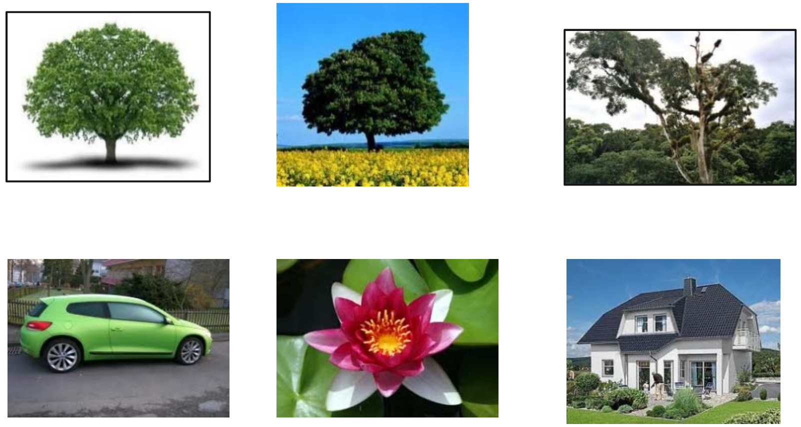
Introduction to Classification

– We (or the learning algorithm) look at positive and negative examples (training data)

– … and derive a model

  e.g., "Trees are big, green plants that have a trunk."

– Goal: Classification of unseen instances



Tree?

Tree?

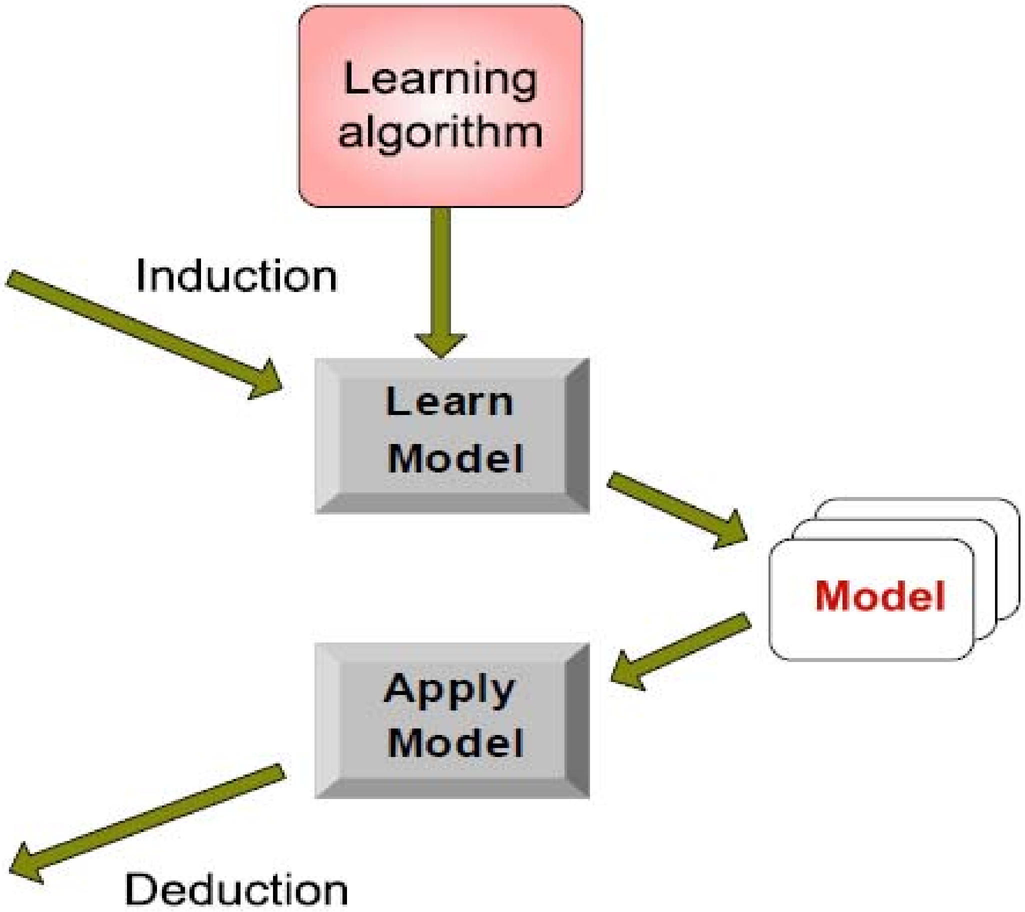
Warning:
Models are only approximating examples!
Not guaranteed to be correct or complete!

Model Learning and Model Application Process

Class/Label Attribute

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set



Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Unseen Records

Classification Examples

– Credit Risk Assessment
  - Attributes: your age, income, debts, …
  - Class: Are you getting credit by your bank?

– Marketing
  - Attributes: previously bought products, browsing behaviour
  - Class: Are you a target customer for a new product?

– Tax Fraud
  - Attributes: the values in your tax declaration
  - Class: Are you trying to cheat?

– SPAM Detection
  - Attributes: words and header fields of an e-mail
  - Class: Is it a spam e-mail?

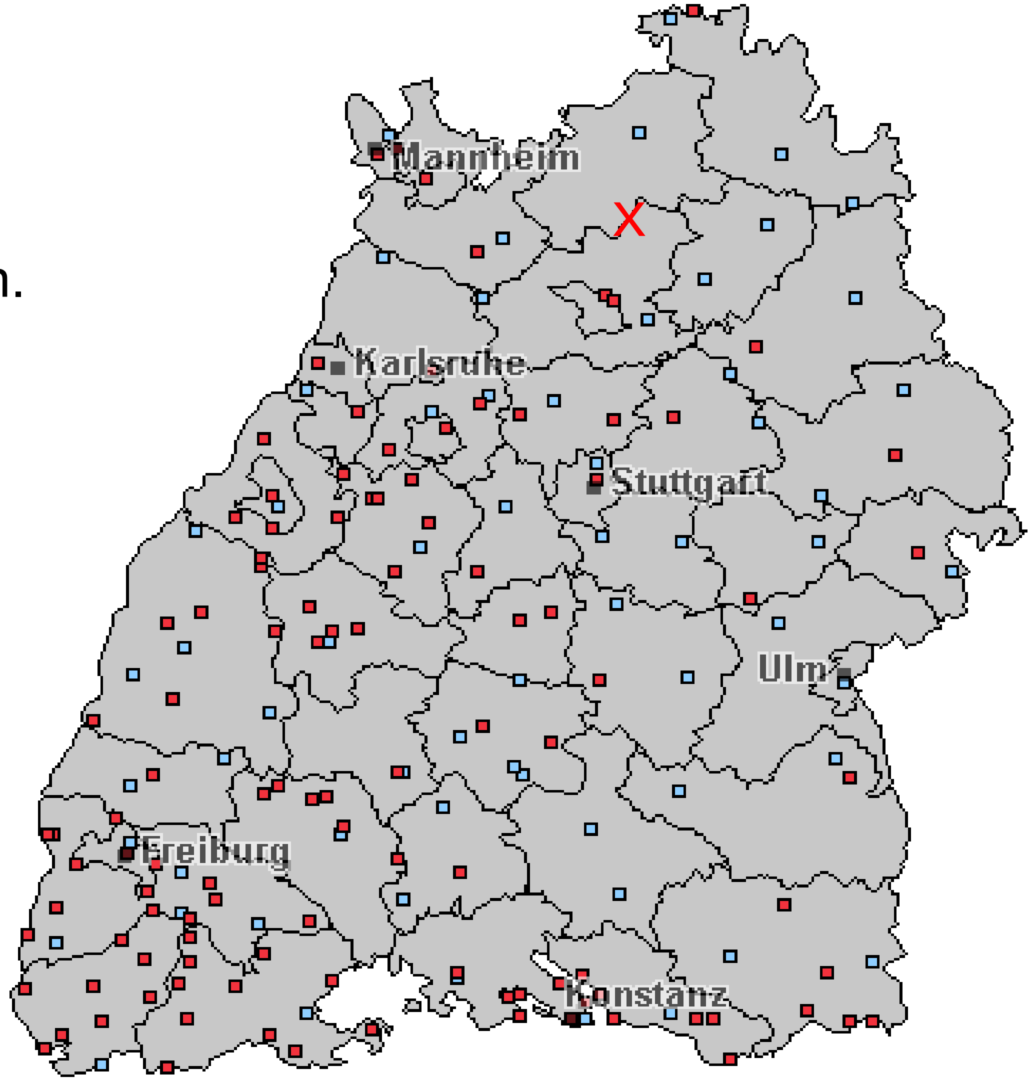
Classification Techniques

1. K-Nearest-Neighbors
2. Decision Trees
3. Rule Learning
4. Naïve Bayes
5. Support Vector Machines
6. Artificial Neural Networks
7. Deep Neural Networks
8. Many others …

2. K-Nearest-Neighbors

Example Problem

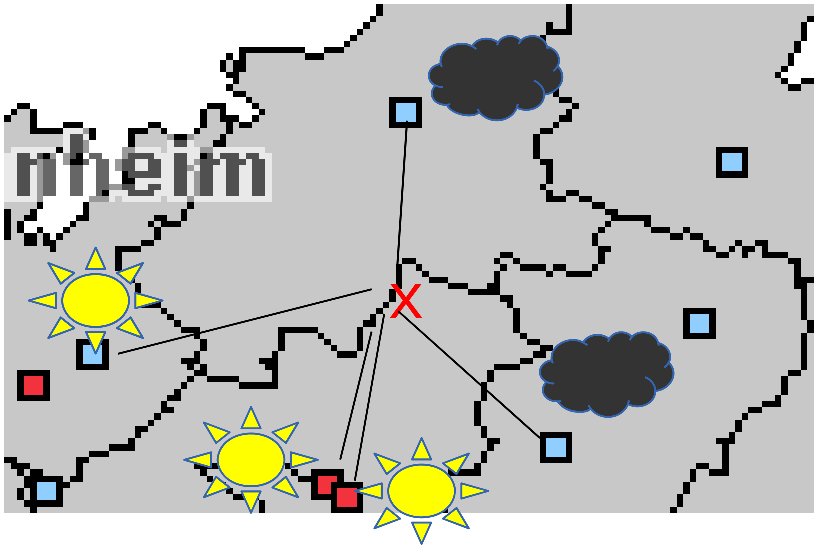
- Predict what the current weather is in a certain place
- where there is no weather station.
- How could you do that?



Basic Idea

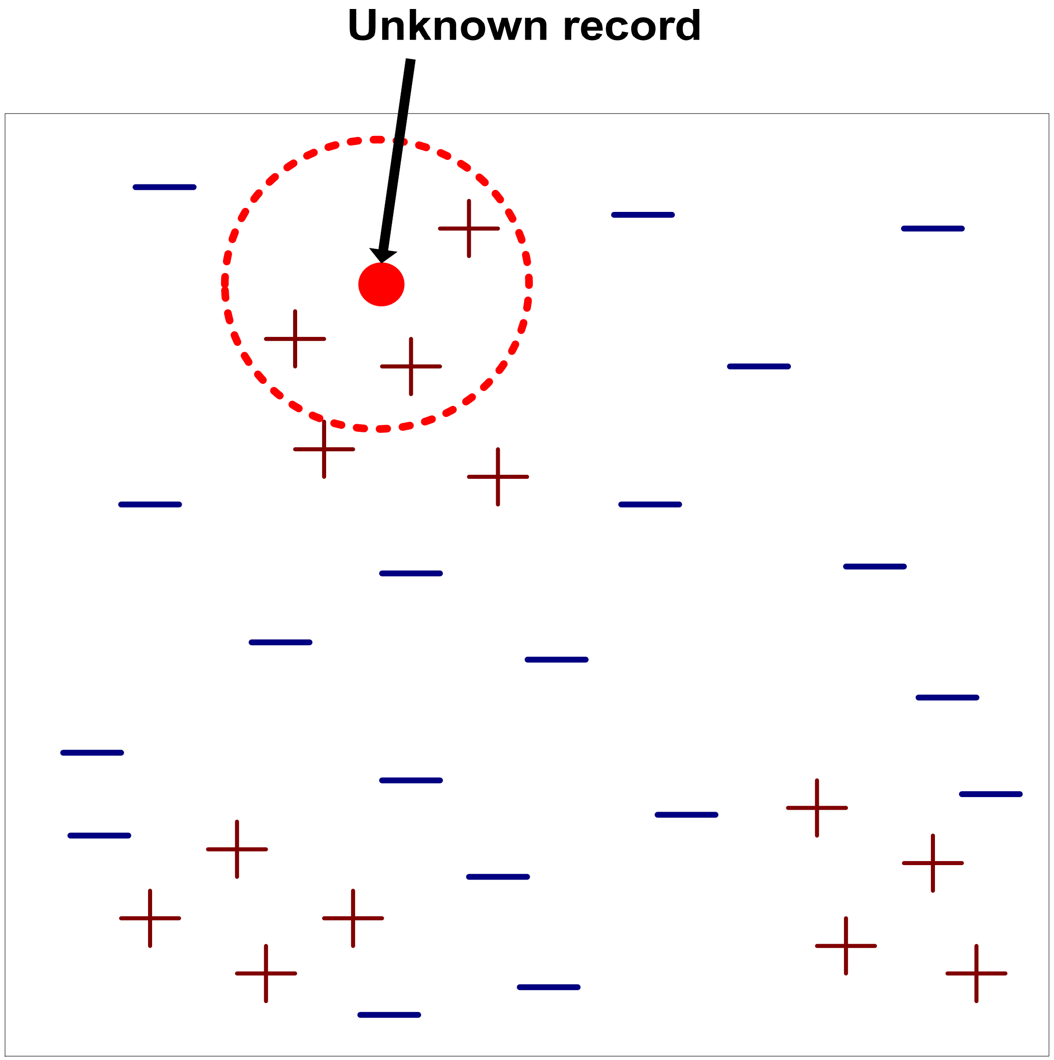
– Use the average of the nearest stations

– Example:
  - 3x sunny
  - 2x cloudy
  - result = sunny



– This approach is called K-Nearest-Neighbors
  - where k is the number of neighbors to consider
  - in the example: k=5
  - in the example: “near” denotes geographical proximity

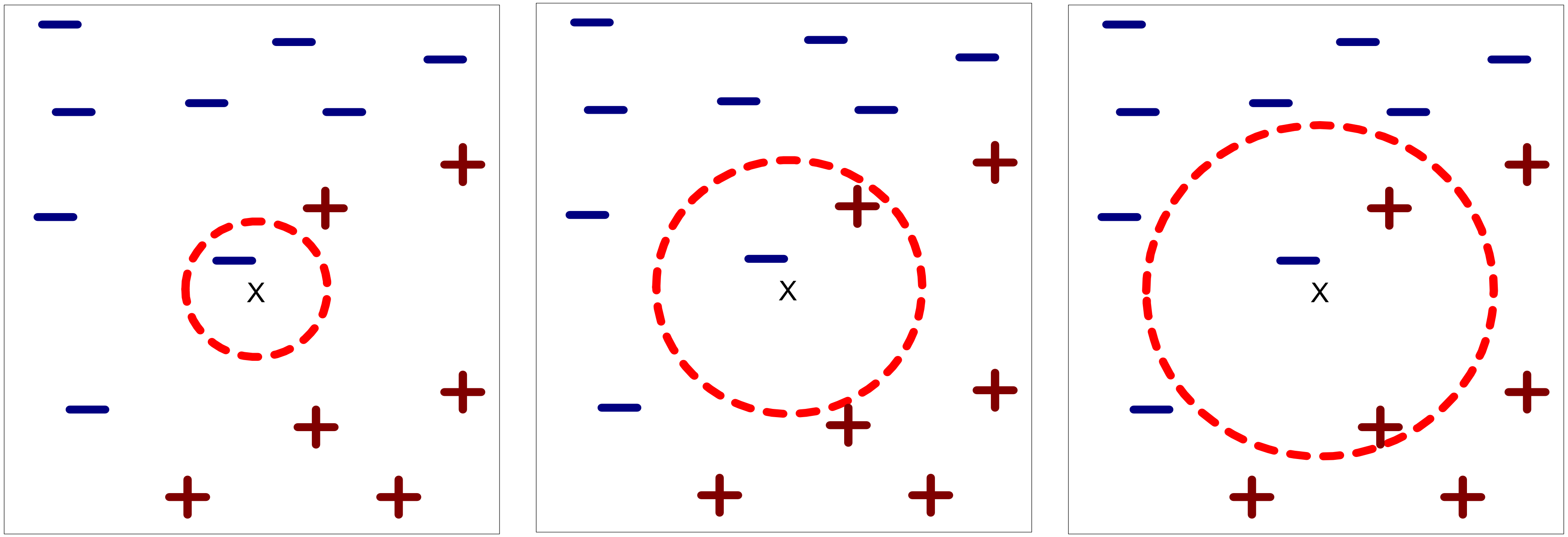
K-Nearest-Neighbors Classifiers



- Require three things
  - The set of stored records
  - A distance measure to compute distance between records
  - The value of k, the number of nearest neighbors to consider
- To classify an unknown record:
  1. Compute distance to each training record
  2. Identify k-nearest neighbors
  3. Use class labels of nearest neighbors to determine the class label of unknown record
     - by taking majority vote or
     - by weighing the vote according to distance

Examples of K-Nearest Neighbors

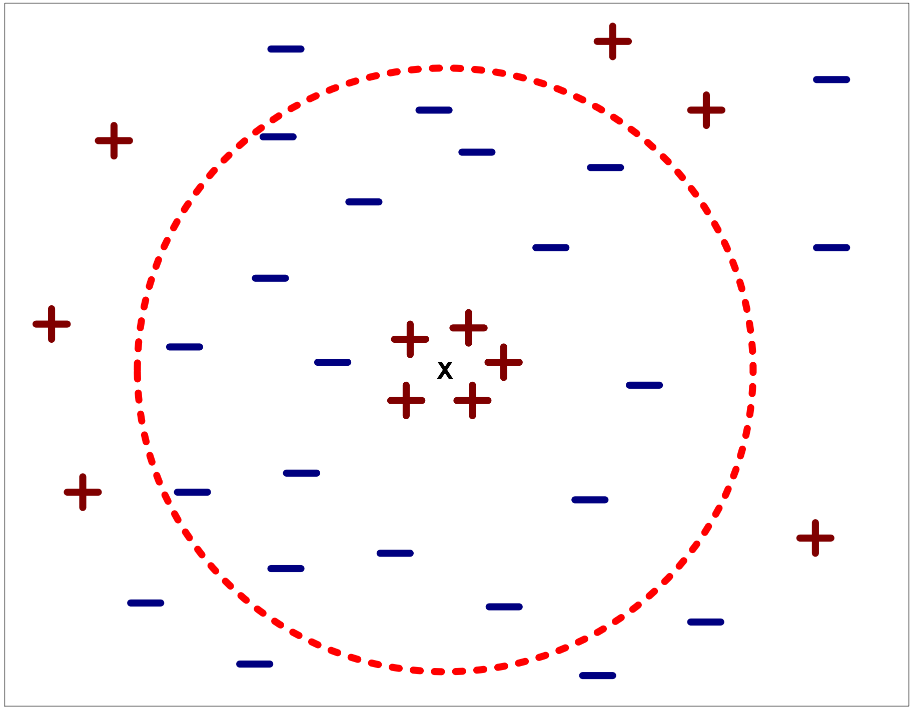
The k-nearest neighbors of a record x are data points that have the k smallest distances to x.



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

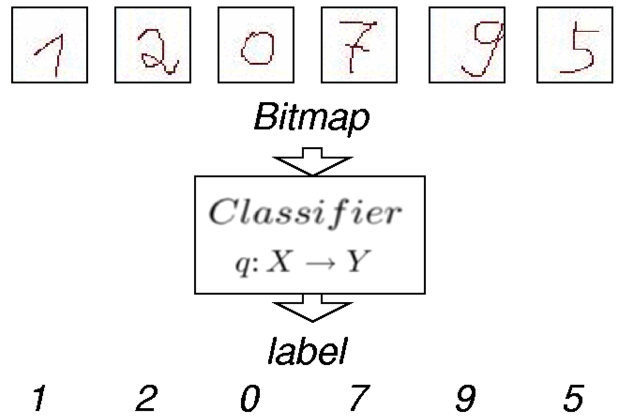
Choosing a Good Value for K

– If k is too small, the result is sensitive to noise points
– If k is too large, the neighborhood may include points from other classes



– Rule of thumb: Test k values between 1 and 10.

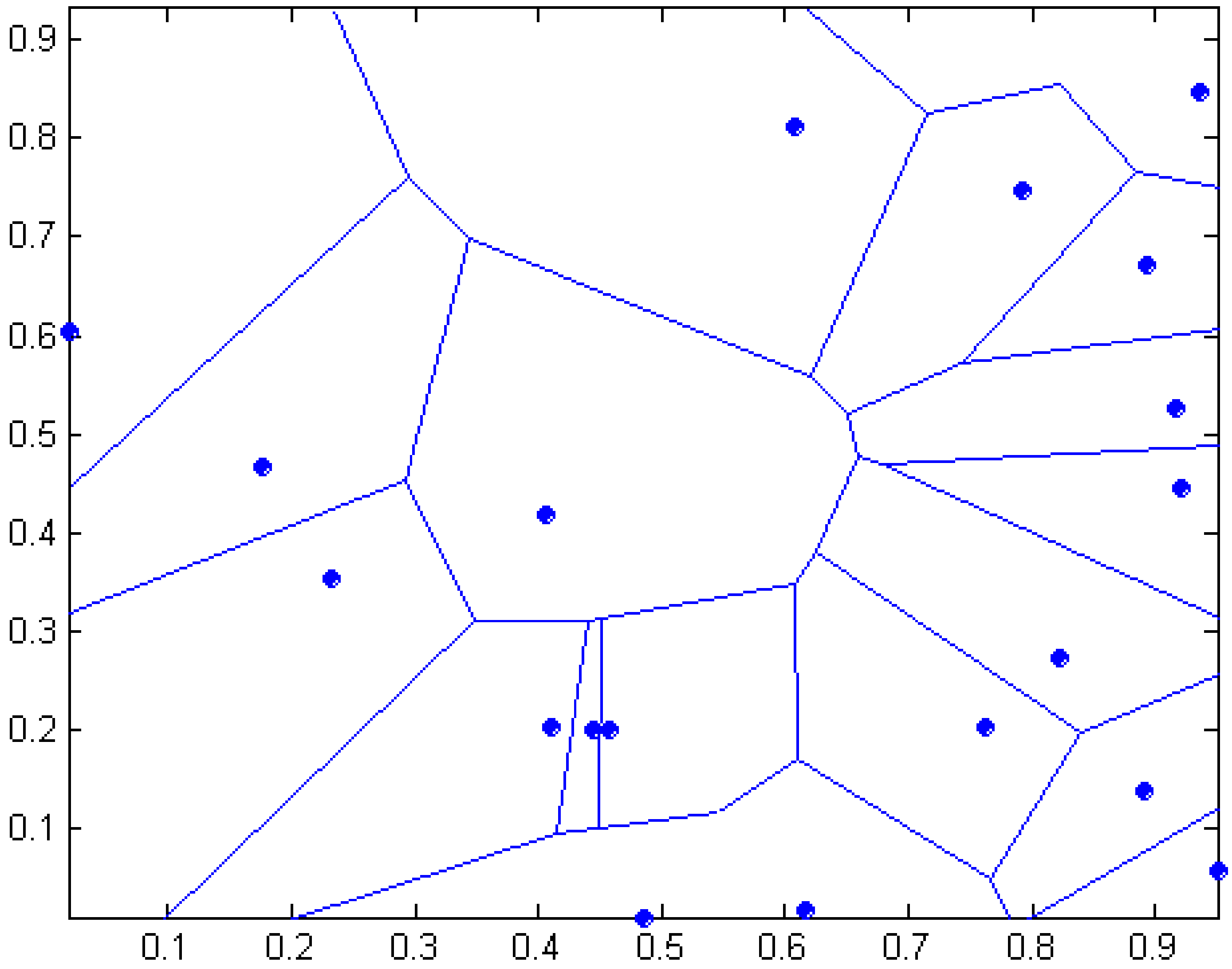
Discussion of K-Nearest-Neighbor Classification



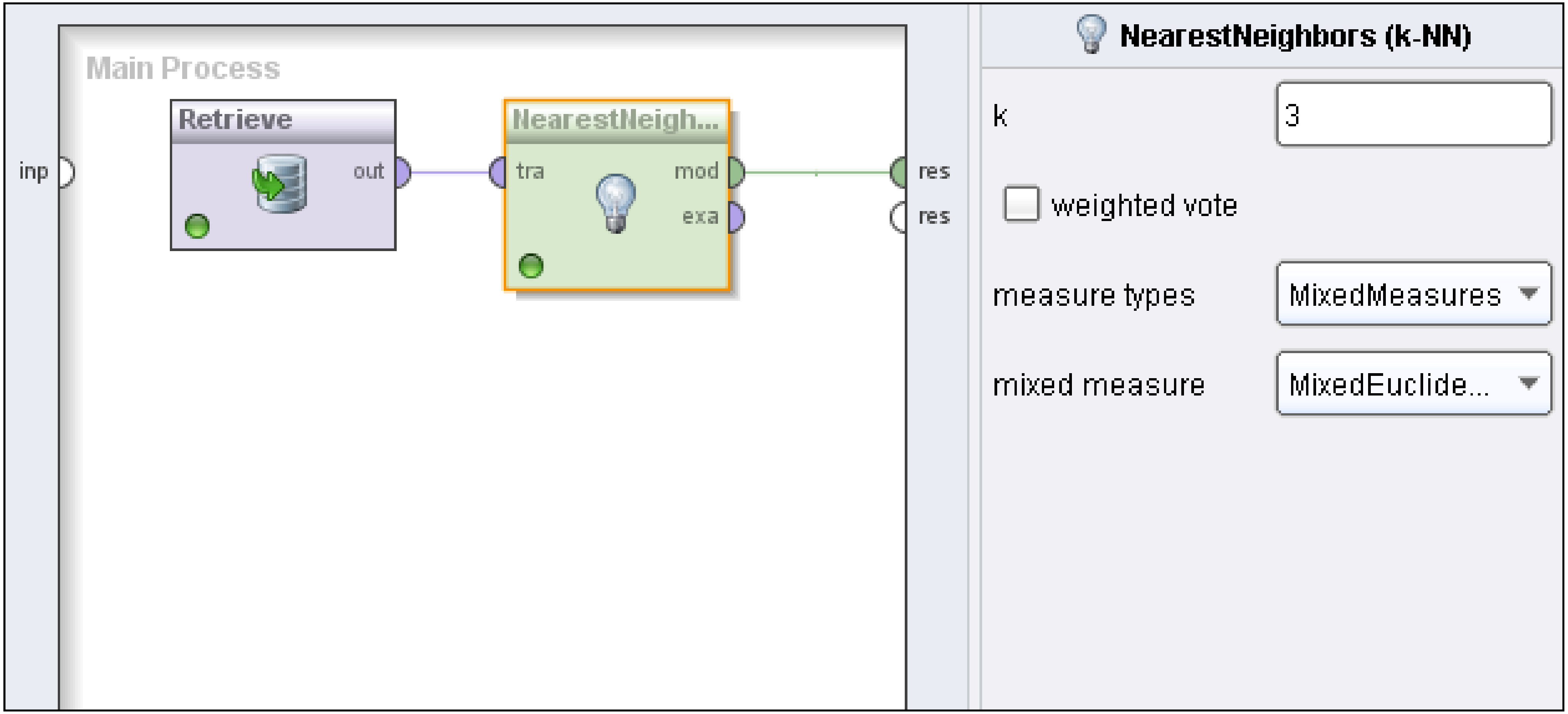
– Often very accurate
  - for instance for optical character recognition (OCR)
– … but slow
  - as training data needs to be searched

– Assumes that all attributes are equally important
  - remedy: attribute selection or attribute weights
– Can handle decision boundaries which are not parallel to the axes (unlike decision trees)

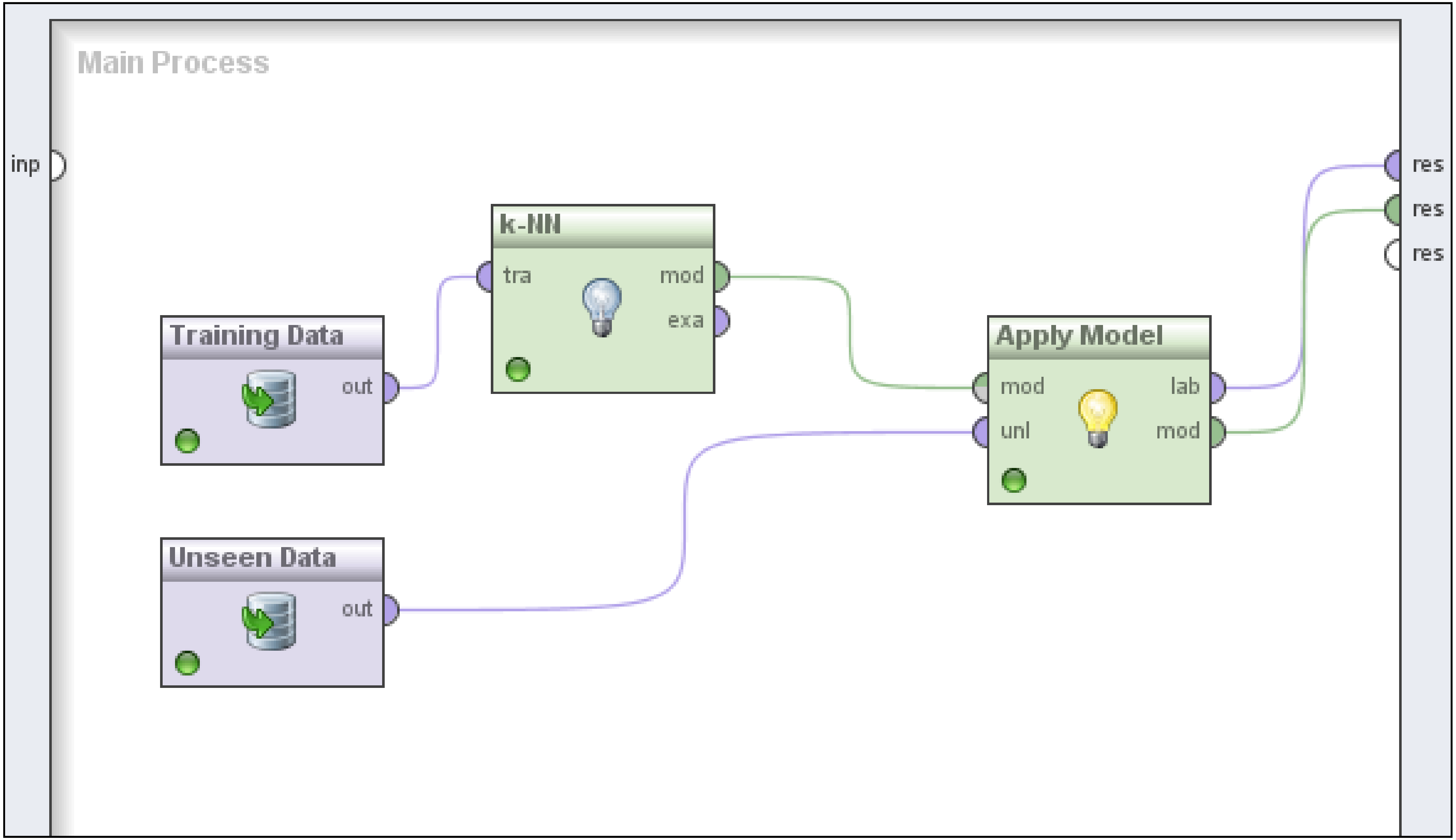
Decision Boundaries of a 1-NN Classifier



KNN in RapidMiner



Applying the Model



Resulting Dataset

Prediction

Confidence scores

Result Overview | ExampleSet (Ret olf-Testset)

Data

Statistics

Charts

Advanced Charts

Annotation

ExampleSet (14 examples, 4 spec attributes, 4 regular attribute)

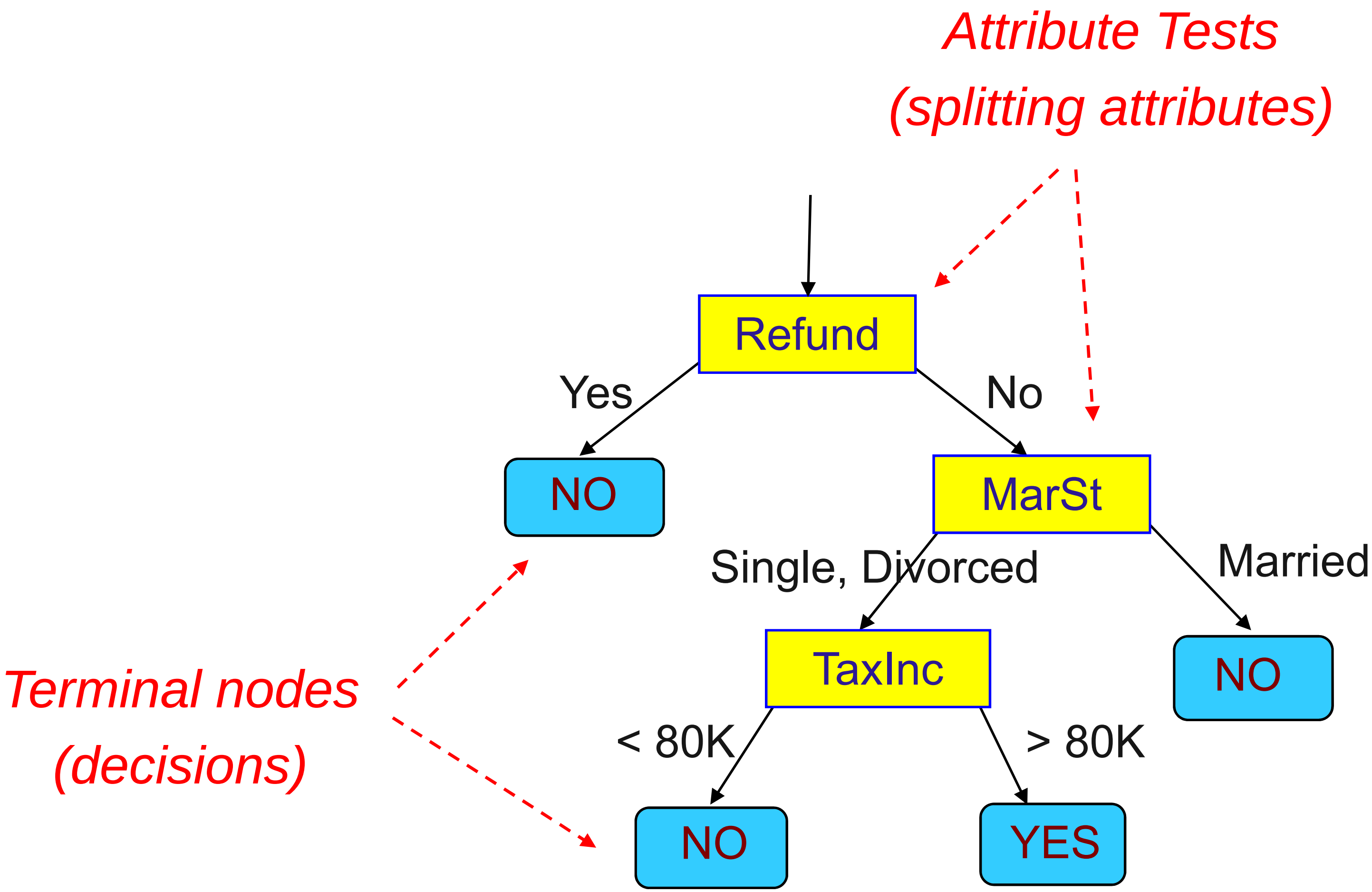
Row No.	Play	prediction(Play)	confidence(no)	confidence(yes)	Outlook	Temper
1	yes	no	0.667	0.333	sunny	85
2	no	no	0.667	0.333	overcast	80
3	yes	yes	0.333	0.667	overcast	83
4	yes	yes	0.333	0.667	rain	70
5	yes	yes	0.333	0.667	rain	68
6	no	yes	0.333	0.667	rain	65
7	yes	yes	0.333	0.667	overcast	64
8	no	yes	0.333	0.667	sunny	72
9	yes	yes	0.333	0.667	sunny	69
10	no	yes	0.333	0.667	sunny	75
11	yes	yes	0.333	0.667	sunny	68
12	yes	yes	0.333	0.667	overcast	72
13	no	yes	0	1	overcast	81
14	yes	yes	0.333	0.667	rain	71

Lazy versus Eager Learning

- Lazy Learning
  - Instance-based learning approaches, like KNN, are also called lazy learning as no explicit knowledge (model) is learned
  - Single goal: Classify unseen records as accurately as possible

- Eager Learning
  - but actually, we might have two goals
    1. classify unseen records
    2. understand the application domain as a human
  - Eager learning approaches generate models that are (might be) interpretable by humans
  - Examples of eager techniques: Decision Tree Learning, Rule Learning

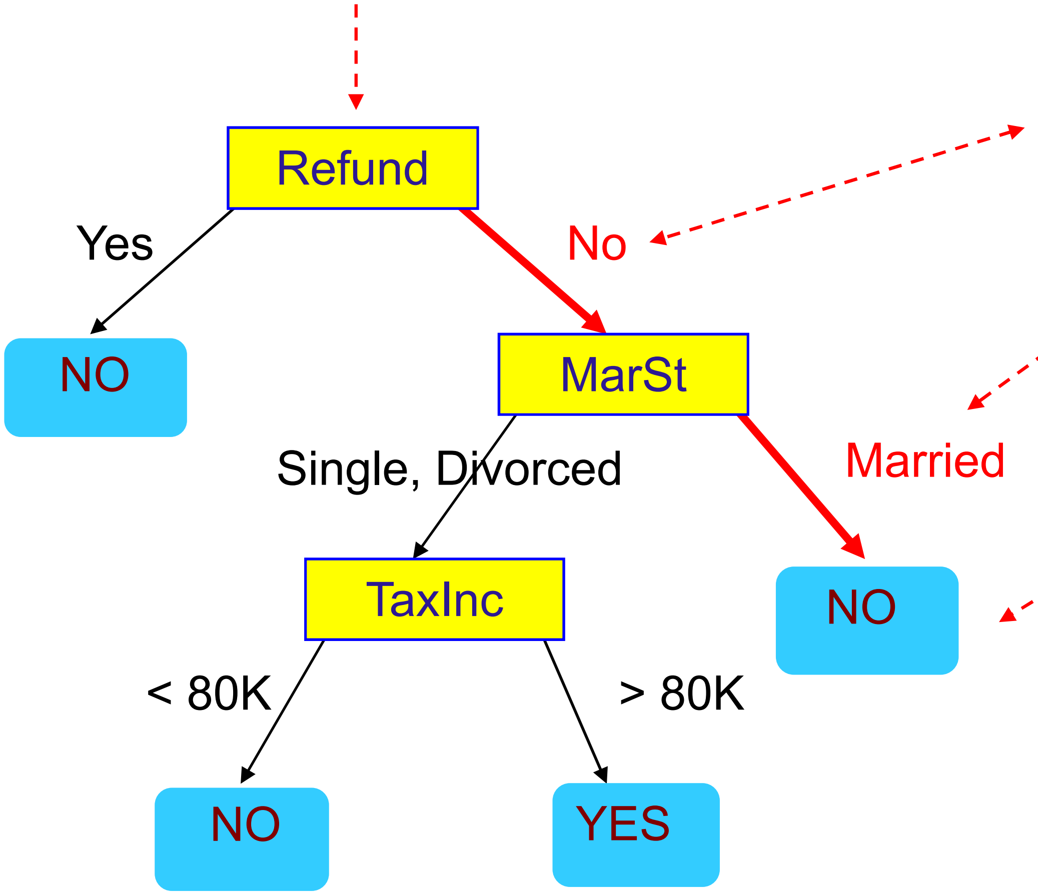
3. Decision Tree Classifiers



Decision trees encode a procedure for taking a classification decision.

Applying a Decision Tree to Unseen Data

Start from the root of tree.

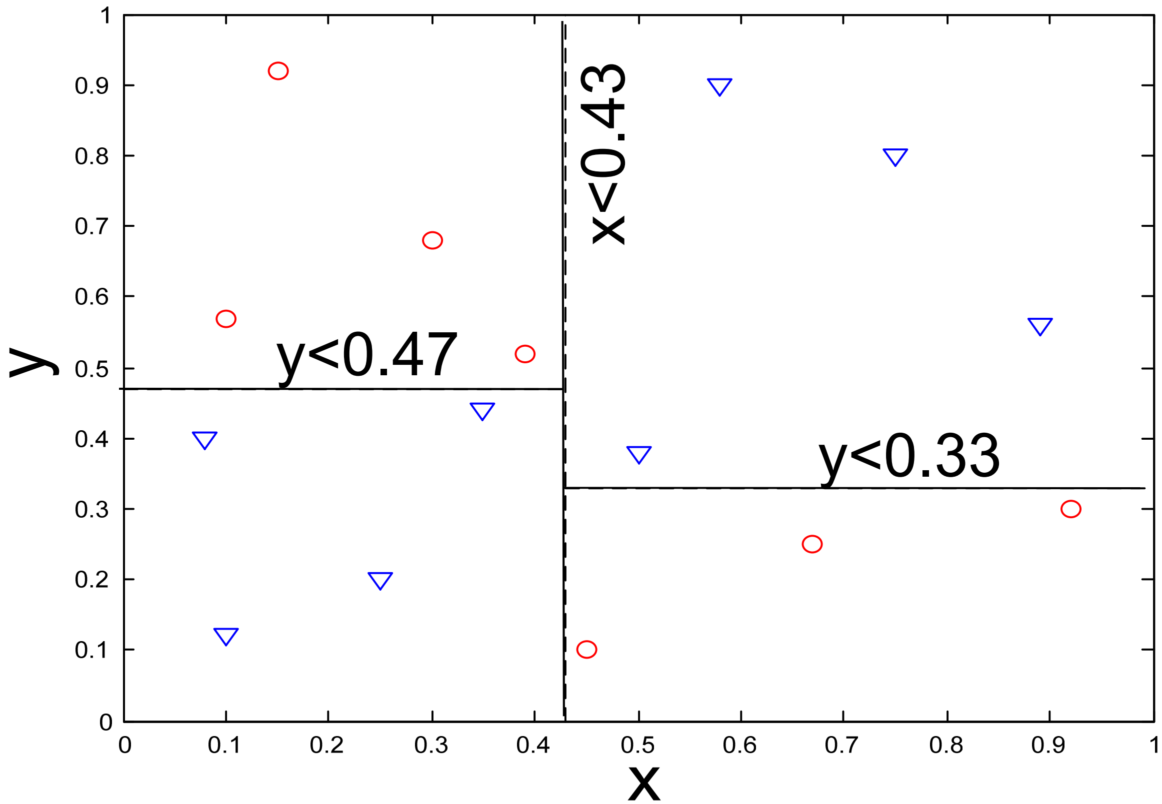


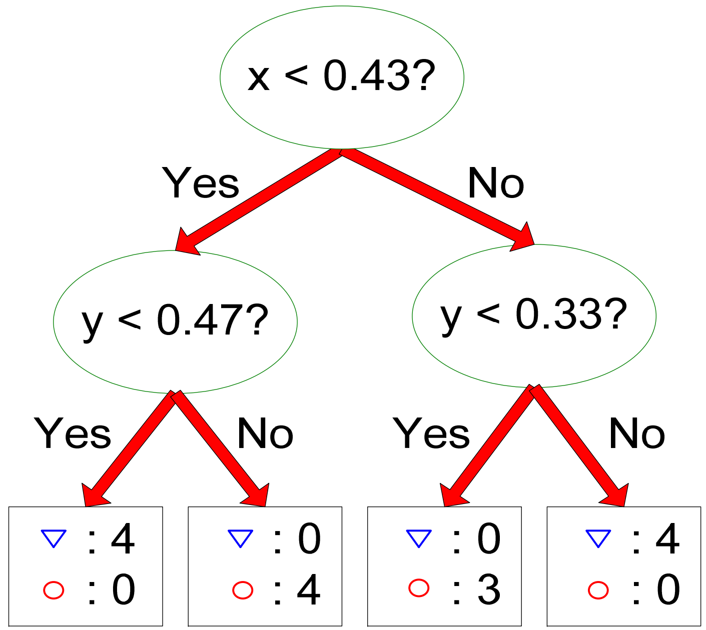
Unseen Record

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Assign Cheat to “No”

Decision Boundary





The decision boundaries are parallel to the axes because the test condition involves a single attribute at-a-time.

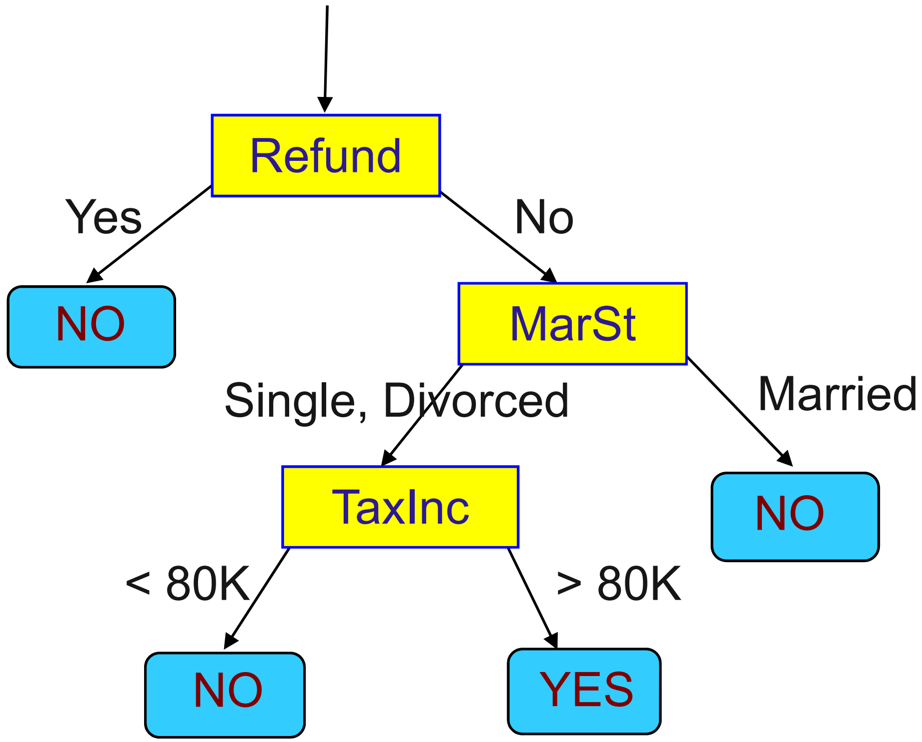
Decision Tree Induction

categorical categorical continuous class

Tid	**Refund**	**Marital Status**	**Taxable Income**	**Cheat**
1	Yes	Single	125K	**No**
2	No	Married	100K	**No**
3	No	Single	70K	**No**
4	Yes	Married	120K	**No**
5	No	Divorced	95K	**Yes**
6	No	Married	60K	**No**
7	Yes	Divorced	220K	**No**
8	No	Single	85K	**Yes**
9	No	Married	75K	**No**
10	No	Single	90K	**Yes**

Training Data

Learning Algorithm



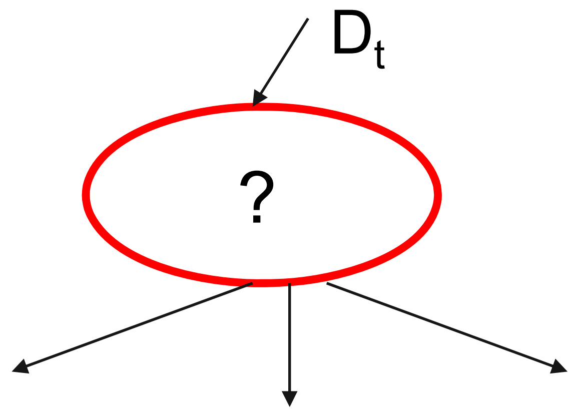
Model: Decision Tree

Decision Tree Induction

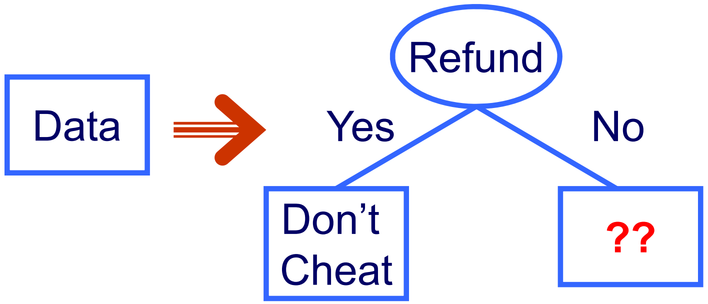
- How to learn a decision tree from training data?
- Finding an optimal decision tree is NP-hard.
- Tree building algorithms thus use a greedy, top-down, recursive partitioning strategy to induce a reasonable solution.
- Many different algorithms have been proposed:
  - Hunt's Algorithm
  - ID3
  - C4.5
  - CHAID

Hunt's Algorithm

– Let D_t be the set of training records that reach a node t
– General procedure:
  • If D_t only contains records that belong to the same class y_t, then t is a leaf node labeled as y_t
  • If D_t contains records that belong to more than one class, use an attribute test to split the data into subsets having a higher purity.
  • Recursively apply the procedure to each subset.



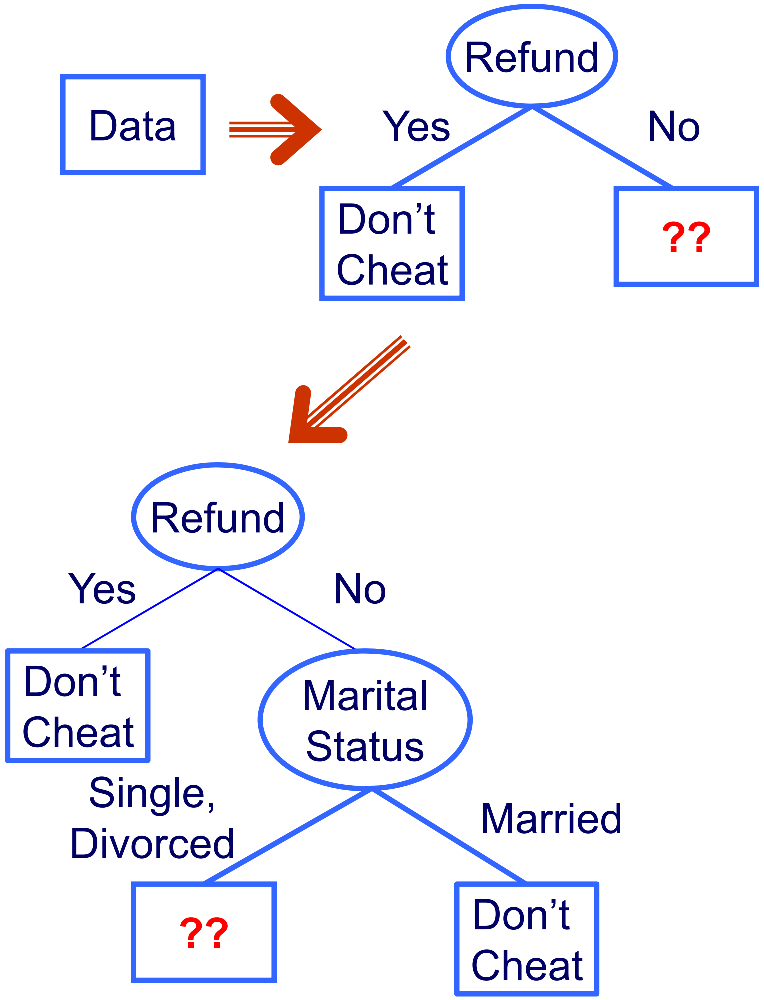
Hunt’s Algorithm – Step 1



Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

1. We test all possible splits and measure the purity of the resulting subsets
2. We find the split on Refund to produce the purest subsets

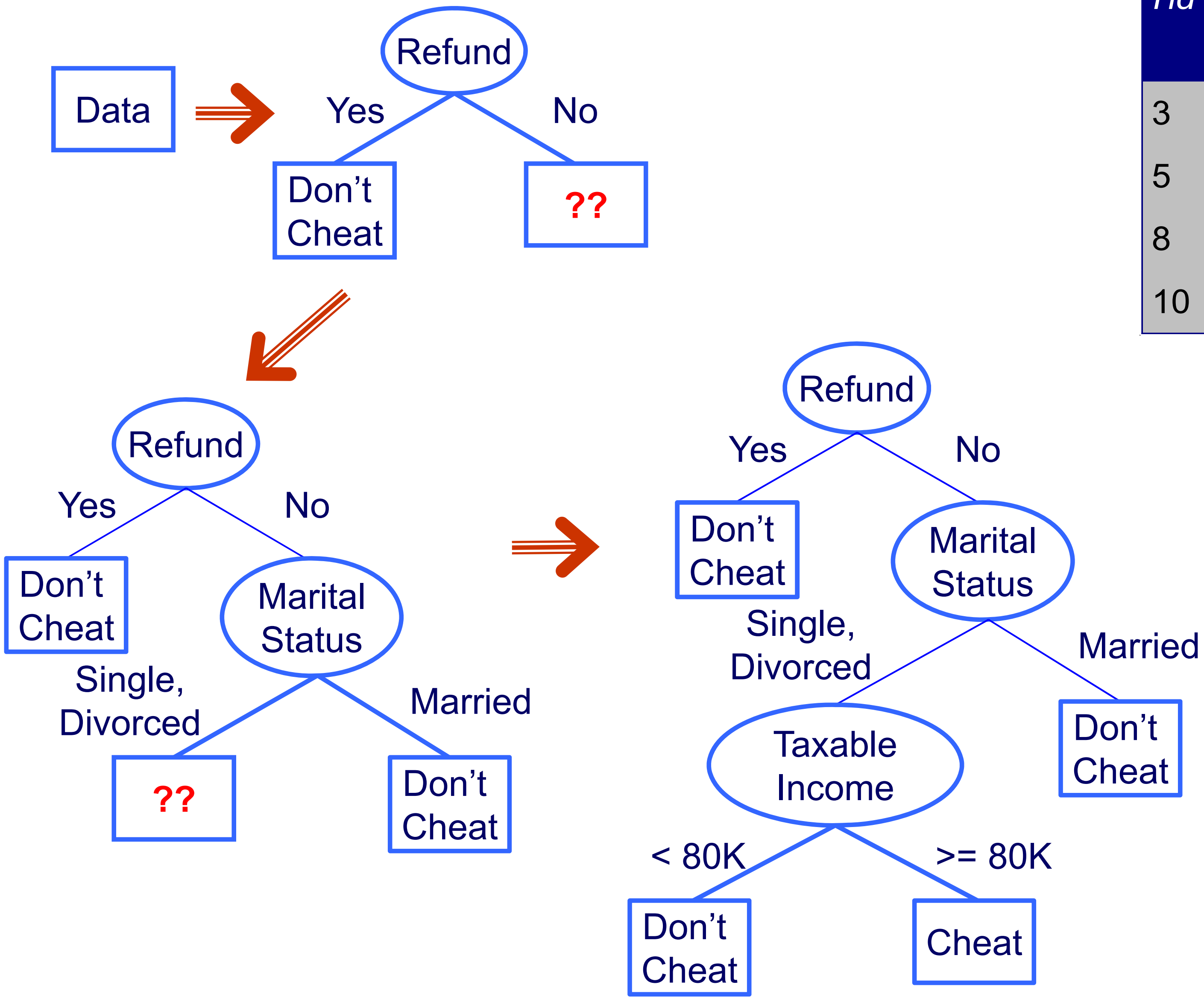
Hunt’s Algorithm – Step 2



Tid	**Refund**	**Marital Status**	**Taxable Income**	**Cheat**
2	No	Married	100K	**No**
3	No	Single	70K	**No**
5	No	Divorced	95K	**Yes**
6	No	Married	60K	**No**
8	No	Single	85K	**Yes**
9	No	Married	75K	**No**
10	No	Single	90K	**Yes**

1. We further examine the Refund=No records
2. Again, we test all possible splits
3. We find the split on Marital Status to produce the purest subsets

Hunt's Algorithm – Step 3



Tid	**Refund**	**Marital Status**	**Taxable Income**	**Cheat**
3	No	Single	70K	**No**
5	No	Divorced	95K	**Yes**
8	No	Single	85K	**Yes**
10	No	Single	90K	**Yes**

1. We further examine the Marital Status=Single or =Divorced records
2. We find a split on Taxable Income to produce pure subsets

Tree Induction Issues

1. Determine how to split the records
   - How to specify the attribute test condition?
   - How to determine the best split?
2. Determine when to stop splitting

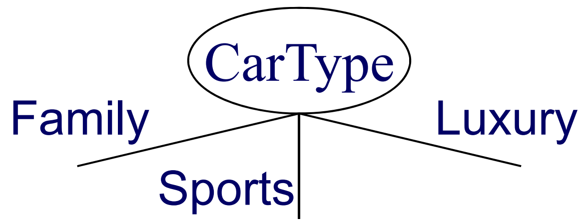
3.1 How to Specify the Attribute Test Condition?

1. Depends on attribute types
   - Nominal
   - Ordinal
   - Continuous

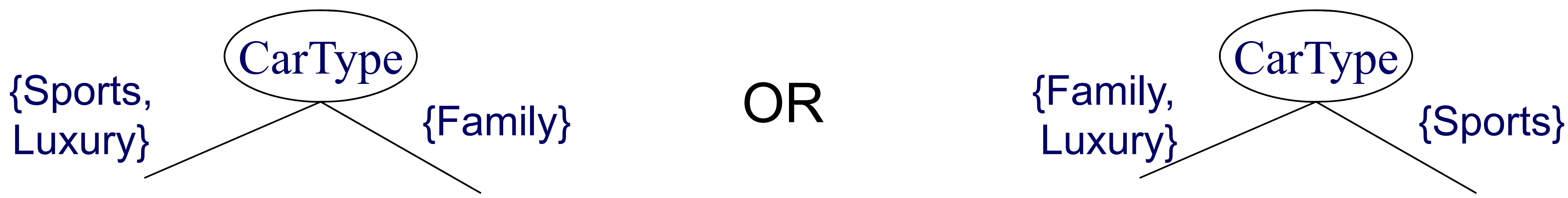
2. Depends on number of ways to split
   - 2-way split
   - Multi-way split

Splitting Based on Nominal Attributes

– Multi-way split: Use as many partitions as distinct values.

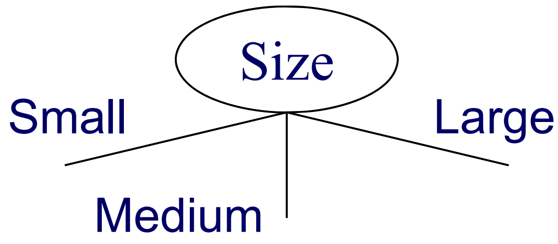


– Binary split: Divides values into two subsets. Need to find optimal partitioning.

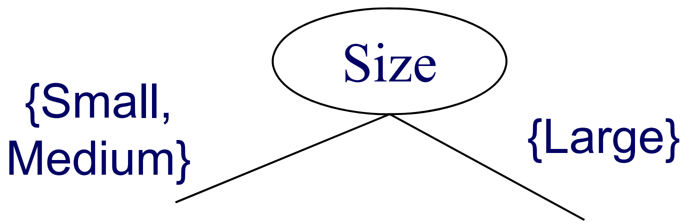


Splitting Based on Ordinal Attributes

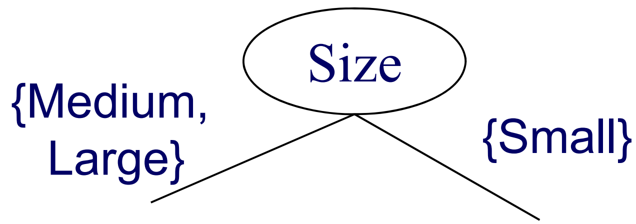
- Multi-way split: Use as many partitions as distinct values.



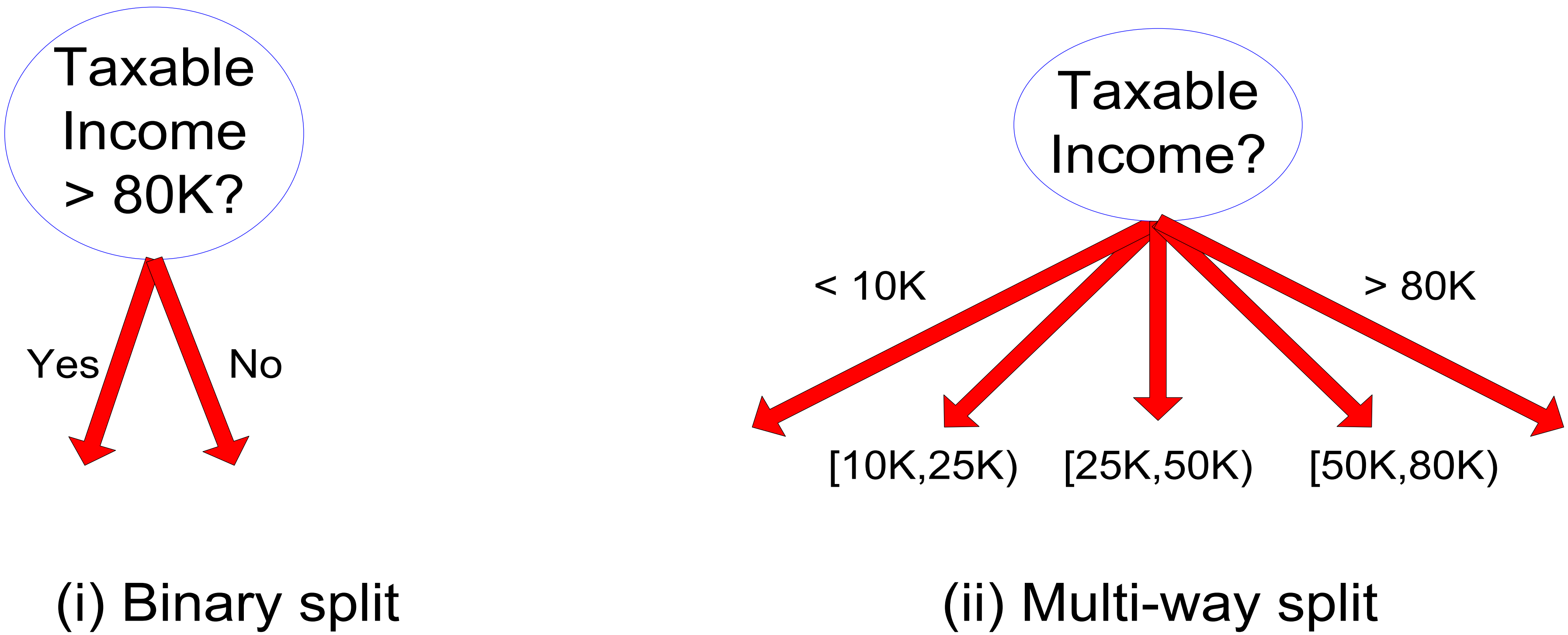
- Binary split: Divides values into two subsets while keeping the order. Need to find optimal partitioning.



OR



Splitting Based on Continuous Attributes



Splitting Based on Continuous Attributes

– Different ways of handling continuous attributes

- Discretization to form an ordinal categorical attribute
  - equal-interval binning
  - equal-frequency binning
  - binning based on user-provided boundaries
- Binary Decision: $(A < v)$ or $(A \geq v)$
  - usually sufficient in practice
  - find the best splitting border v based on a purity measure (see below)
  - can be compute intensive

Discretization Example

- Values of the attribute, e.g., age of a person:
  - 0, 4, 12, 16, 16, 18, 24, 26, 28
- Equal-interval binning – for bin width of e.g., 10:
  - Bin 1: 0, 4 [-,10) bin
  - Bin 2: 12, 16, 16, 18 [10,20) bin
  - Bin 3: 24, 26, 28 [20,+) bin

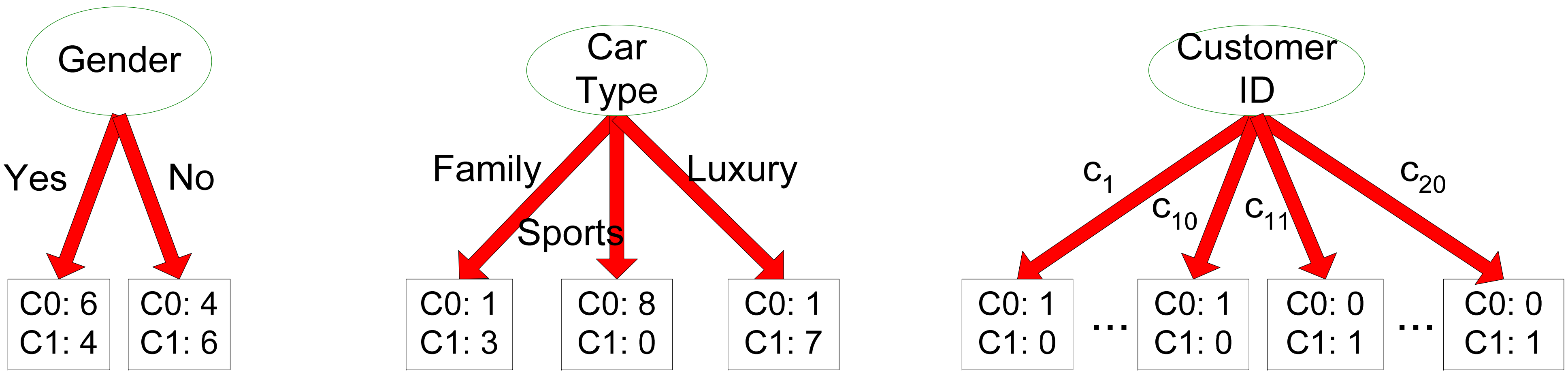
  – denote negative infinity, + positive infinity
- Equal-frequency binning – for bin density of e.g., 3:
  - Bin 1: 0, 4, 12 [-, 14) bin
  - Bin 2: 16, 16, 18 [14, 21) bin
  - Bin 3: 24, 26, 28 [21,+] bin

3.2 How to Find the Best Split?

Customer Id	Gender	Car Type	Shirt Size	Class
1	M	Family	Small	C0
2	M	Sports	Medium	C0
3	M	Sports	Medium	C0
4	M	Sports	Large	C0
5	M	Sports	Extra Large	C0
6	M	Sports	Extra Large	C0
7	F	Sports	Small	C0
8	F	Sports	Small	C0
9	F	Sports	Medium	C0
10	F	Luxury	Large	C0
11	M	Family	Large	C1
12	M	Family	Extra Large	C1
13	M	Family	Medium	C1
14	M	Luxury	Extra Large	C1
15	F	Luxury	Small	C1
16	F	Luxury	Small	C1
17	F	Luxury	Medium	C1
18	F	Luxury	Medium	C1
19	F	Luxury	Medium	C1
20	F	Luxury	Large	C1

Before splitting the dataset contains:

- 10 records of class 0 and
- 10 records of class 1



Which test condition is the best?

How to Find the Best Split?

– Greedy approach: Test all possible splits and use the one that results in the most homogeneous (= pure) nodes.

– Need a measure of node impurity:

C0: 5 C1: 5	C0: 9 C1: 1
Non-homogeneous, High degree of impurity	Homogeneous, Low degree of impurity

– Common measures of node impurity:
  1. GINI Index
  2. Entropy
  3. GainRATIO

Comparing different Splits

Before Splitting:

C0	N_{00}
C1	N_{01}

→ M0

C0	N_{10}
C1	N_{11}

↓ M1

C0	N_{20}
C1	N_{21}

↓ M2

M12

C0	N_{30}
C1	N_{31}

↓ M3

C0	N_{40}
C1	N_{41}

↓ M4

M34

Largest purity gain? M0 – M12 versus M0 – M34

3.2.1 Measure of Impurity: GINI Index

– GINI Index for a given node t :

$$GINI(t) = 1 - \sum_j [p(j \mid t)]^2$$

$p(j \mid t)$ is the relative frequency of class j at node t.

– Minimum (0.0) when all records belong to one class

– Maximum ($1 - 1/n_c$) when records are equally distributed among all classes. n_c = number of classes

C1	**0**
C2	**6**
Gini=0.000	

C1	**1**
C2	**5**
Gini=0.278	

C1	**2**
C2	**4**
Gini=0.444	

C1	**3**
C2	**3**
Gini=0.500	

Examples for computing GINI

$$GINI(t) = 1 - \sum_j [p(j \mid t)]^2$$

C1	**0**
C2	**6**

P(C1) = 0/6 = 0   P(C2) = 6/6 = 1

Gini = 1 – $P(C1)^2$ – $P(C2)^2$ = 1 – 0 – 1 = 0

C1	**1**
C2	**5**

P(C1) = 1/6   P(C2) = 5/6

Gini = 1 – $(1/6)^2$ – $(5/6)^2$ = 0.278

C1	**2**
C2	**4**

P(C1) = 2/6   P(C2) = 4/6

Gini = 1 – $(2/6)^2$ – $(4/6)^2$ = 0.444

Splitting Based on GINI

- When a node p is split into k partitions (children), the GINI index of each partition is weighted according to the partition's size.
- The quality of the overall split is computed as:

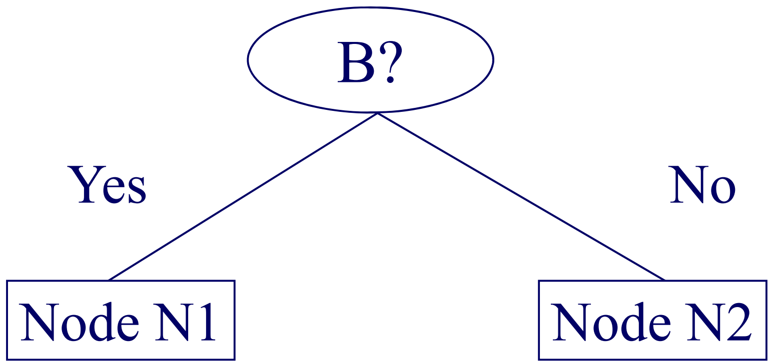
$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)$$

where: n_i = number of records at child i

n = number of records at node p

Example: Calculating the Purity Gain of a Possible Split

Split into two partitions



	Parent
C1	**6**
C2	**6**
Gini = 0.500	

GINI(N1)
$= 1 - (5/7)^2 - (2/7)^2$
$= 0.408$

GINI(N2)
$= 1 - (1/5)^2 - (4/5)^2$
$= 0.32$

	N1	N2
C1	**5**	**1**
C2	**2**	**4**
Gini=0.371		

$\text{GINI}_{\text{Split}}$
$= 7/12 * 0.408 +$
$5/12 * 0.32$
$= 0.371$

Purity Gain = 0.5 – 0.371 = 0.129

Categorical Attributes: Computing Gini Index

For each distinct attribute value, gather counts for each class.

Multi-way split

	CarType		
	Family	Sports	Luxury
C1	1	2	1
C2	4	1	1
Gini	0.393		

Two-way split (find best partition of values)

	CarType	
	{Sports, Luxury}	{Family}
C1	3	1
C2	2	4
Gini	0.400	

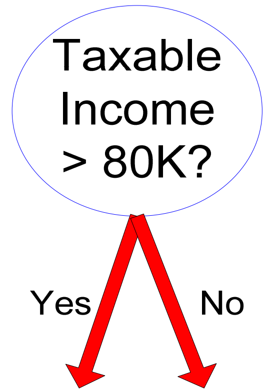
	CarType	
	{Sports}	{Family, Luxury}
C1	2	2
C2	1	5
Gini	0.419	

Continuous Attributes: Computing Gini Index

– How to find the best binary split for a continuous attribute?

– Efficient computation:

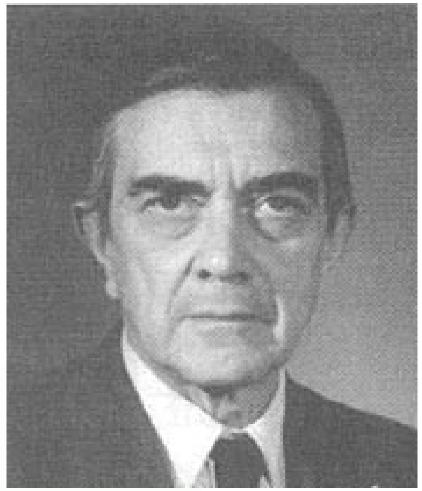
1. Sort the attribute on values
2. Linearly scan these values, each time updating the count matrix and computing the gini index
3. Choose the split position that has the smalest gini index

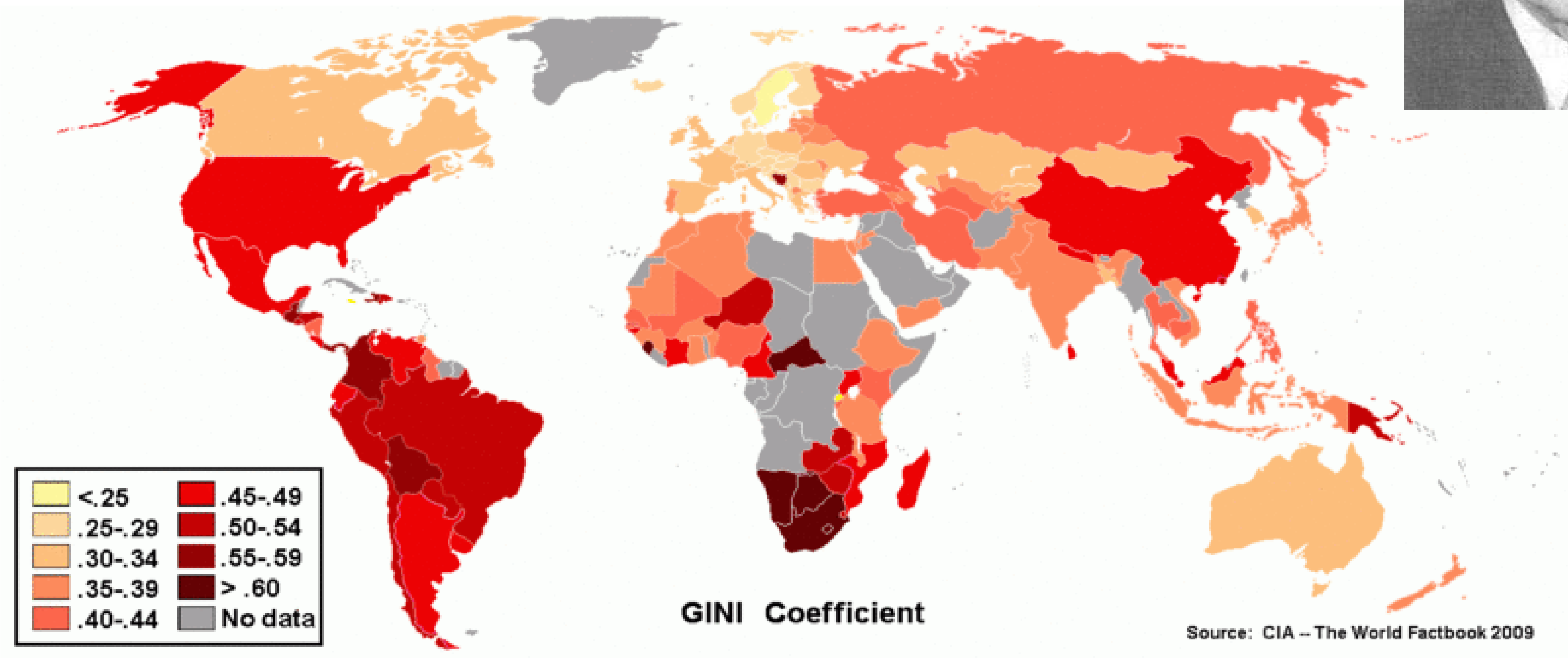


	Taxable Income																					
Sorted Values		60		70		75		85		90		95		100		120		125		220		
Split Positions	55		65		72		80		87		92		97		110		122		172		230	
	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>
Yes	0	3	0	3	0	3	0	3	1	2	2	1	3	0	3	0	3	0	3	0	3	0
No	0	7	1	6	2	5	3	4	3	4	3	4	3	4	4	3	5	2	6	1	7	0
Gini	0.420		0.400		0.375		0.343		0.417		0.400		*0.300*		0.343		0.375		0.400		0.420	

Gini Index

- Named after Corrado Gini (1885-1965)
- Used to measure the distribution of income
  - 1: somebody gets everything
  - 0: everybody gets an equal share





3.2.2 Alternative Splitting Criterion: Information Gain

– Calculating the information gain relies on the entropy of each node.

– Entropy of a given node t:

$$Entropy(t) = -\sum_{j} p(j \mid t) \log_2 p(j \mid t)$$

$p(j \mid t)$ is the relative frequency of class j at node t

– Entropy measures homogeneity of a node
  - Minimum (0.0) when all records belong to one class.
  - Maximum ($\log n_c$) when records are equally distributed among all classes.

Examples for Computing Entropy

$$Entropy(t) = -\sum_j p(j|t) \log_2 p(j|t)$$

C1	**0**
C2	**6**

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1

Entropy = – 0 $\log_2$ 0 – 1 $\log_2$ 1 = – 0 – 0 = 0

C1	**1**
C2	**5**

P(C1) = 1/6 P(C2) = 5/6

Entropy = – (1/6) $\log_2$ (1/6) – (5/6) $\log_2$ (5/6) = 0.65

C1	**2**
C2	**4**

P(C1) = 2/6 P(C2) = 4/6

Entropy = – (2/6) $\log_2$ (2/6) – (4/6) $\log_2$ (4/6) = 0.92

Splitting Based on Information Gain

– Information Gain:

$$GAIN_{split} = Entropy(p) - \left(\sum_{i=1}^{k} \frac{n_i}{n} Entropy(i) \right)$$

Parent Node p is split into k partitions;

n_i is number of records in partition i

– Information gain measures the entropy reduction of a split.

– We choose the split with the largest reduction (maximal GAIN)

– Disadvantage: Tends to prefer splits that result in large number of partitions, each being small but pure (split by ID attribute?)

3.2.3 Alternative Splitting Criterion: GainRATIO

- GainRATIO is designed to overcome the tendency to generate a large number of small partitions.
- GainRATIO adjusts information gain by the entropy of the partitioning (SplitINFO).
- Higher entropy of the partitioning (large number of small partitions) is penalized!

$$GainRATIO_{split} = \frac{GAIN_{Split}}{SplitINFO}$$

$$SplitINFO = -\sum_{i=1}^{k} \frac{n_i}{n} \log \frac{n_i}{n}$$

Parent Node p is split into k partitions

n_i is the number of records in partition i

3.3 Overfitting

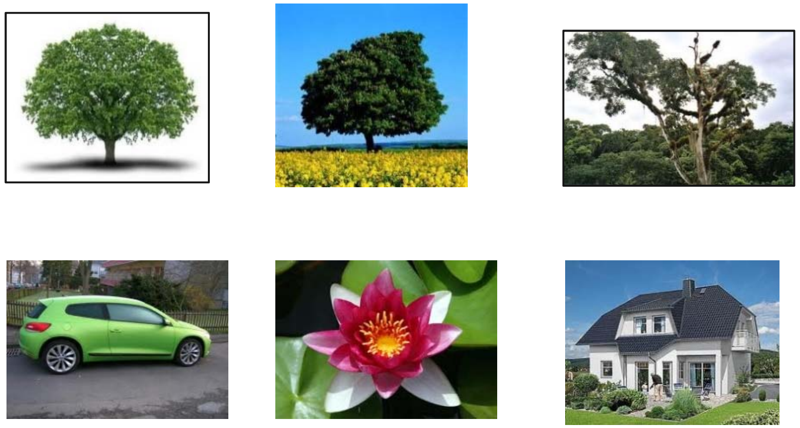
– Problem: Learned models can fit the training data too closely and thus work poorly on unseen data.

– Possible model fitting the training data:

"Trees are big, green plants that have a trunk and no wheels.“

– Unseen instance:

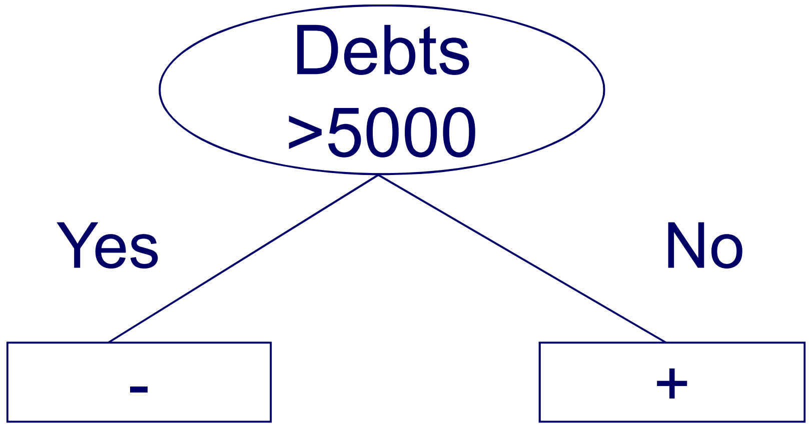
Training data



– Goal: Find good compromise between specificness and generality of a model.

Overfitting: Second Example

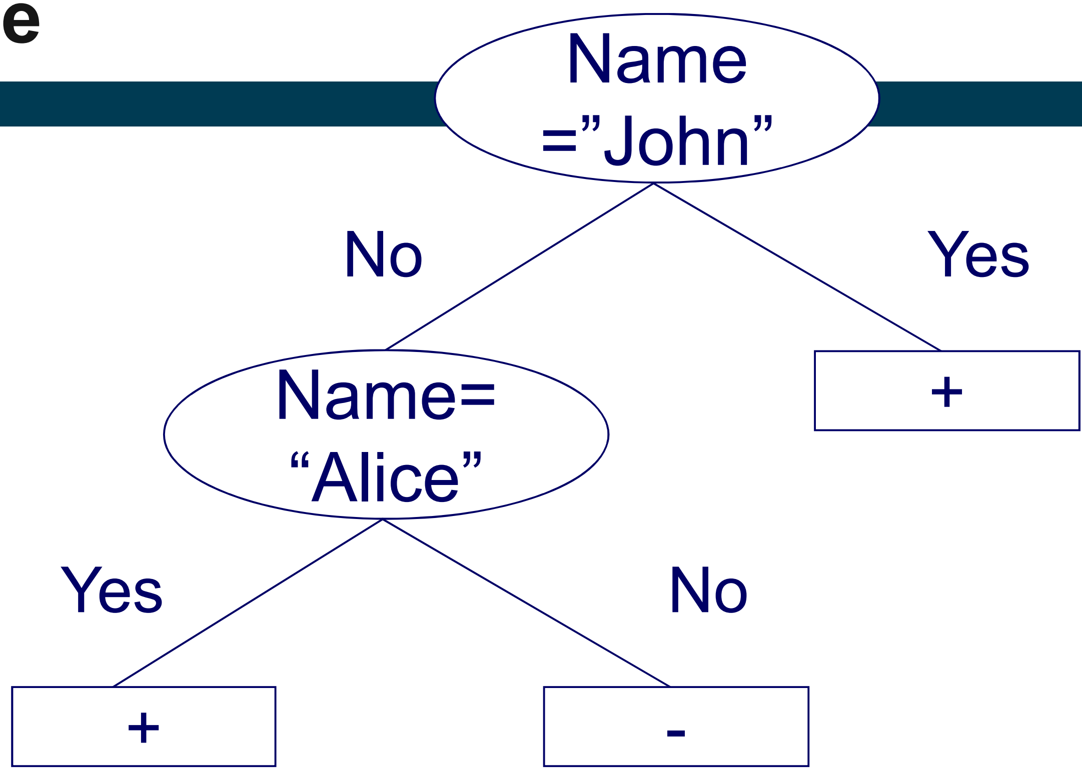
– Example: Predict credit rating
  - possible decision tree:



Name	Net Income	Job status	Debts	Rating
John	40000	employed	0	+
Mary	38000	employed	10000	-
Stephen	21000	self-employed	20000	-
Eric	2000	student	10000	-
Alice	35000	employed	4000	+

Overfitting: Second Example

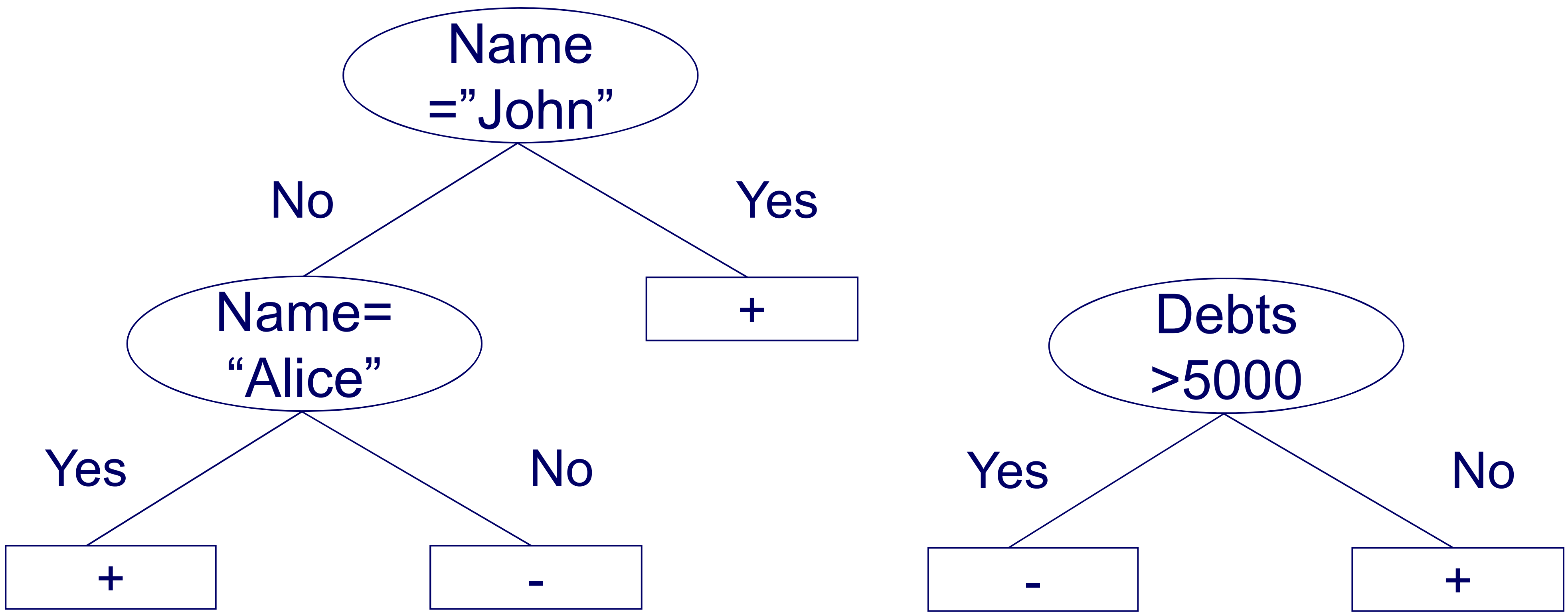
– Example: Predict credit rating
  - alternative decision tree:



Name	Net Income	Job status	Debts	Rating
John	40000	employed	0	+
Mary	38000	employed	10000	-
Stephen	21000	self-employed	20000	-
Eric	2000	student	10000	-
Alice	35000	employed	4000	+

Overfitting: Second Example

- Both trees seem equally good
  - as they classify all instances in the training set correctly
  - Which one do you prefer?



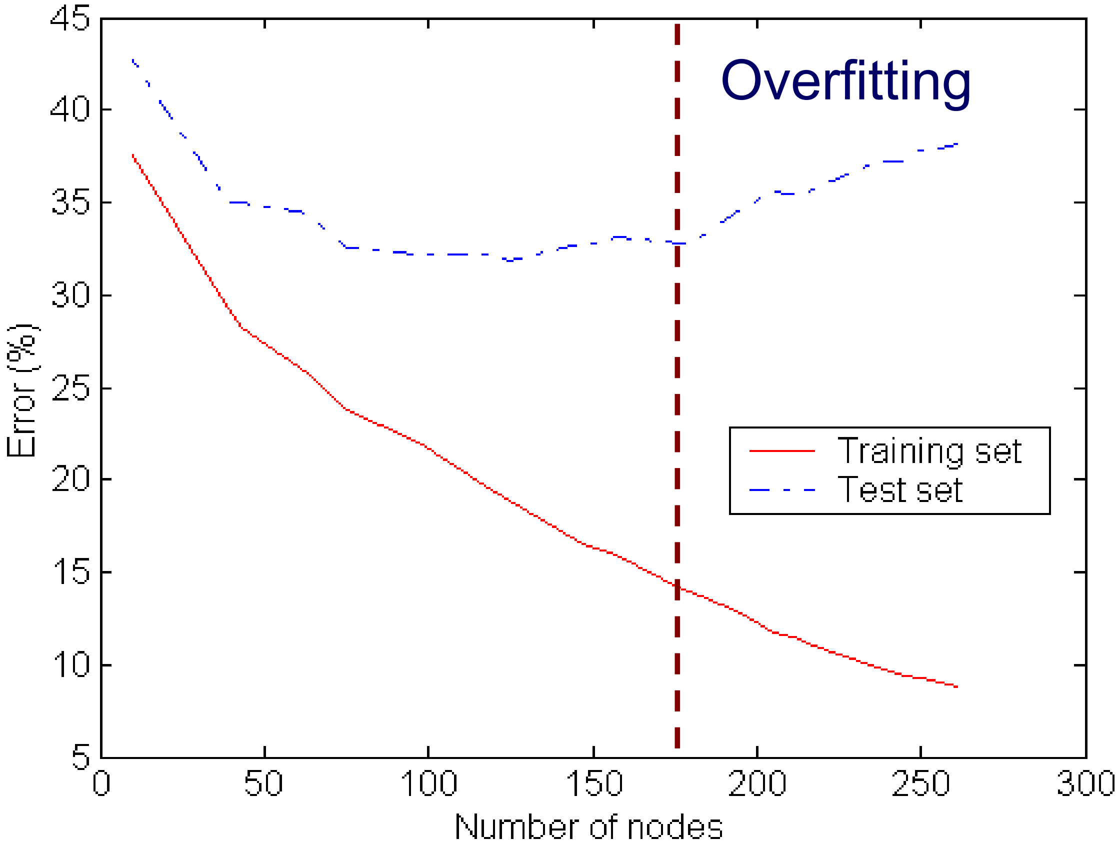
Occam's Razor

- Named after William of Ockham (1287-1347)
- A fundamental principle of science
  - if you have two theories
  - that explain a phenomenon equally well
  - choose the simpler one

- Example:
  - phenomenon: the street is wet
  - theory 1: it has rained
  - theory 2: a beer truck has had an accident, and beer has spilled. The truck has been towed, and magpies picked the glass pieces, so only the beer remains

Overfitting: Symptoms and Causes

– Symptoms:
  1. decision tree too deep and
  2. too many branches
  3. model works well on training set but performs bad on test set

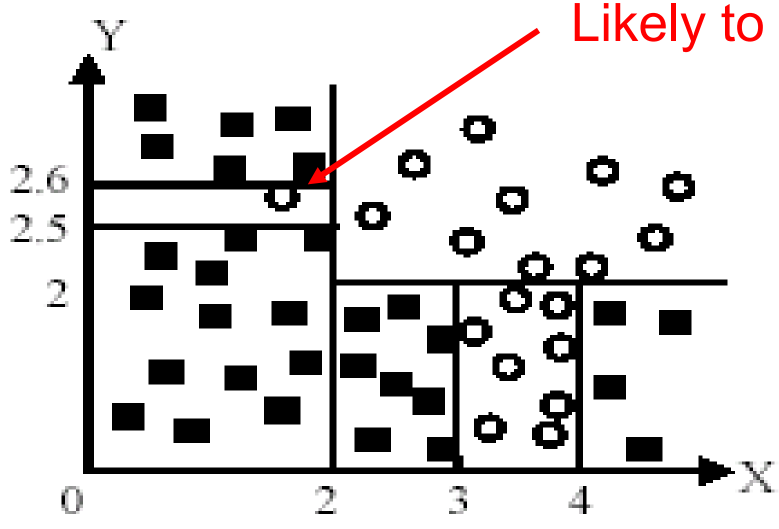


– Typical causes of overfitting
  1. noise / outliers
  2. too little training data
  3. poor learning algorithm

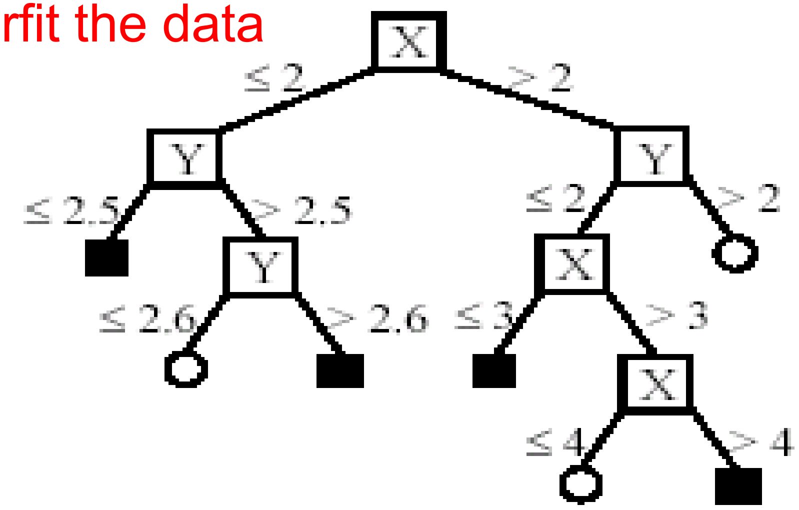
An overfitted model likely does not **generalize** well to **unseen data.**

Example of an Outlier causing Overfitting

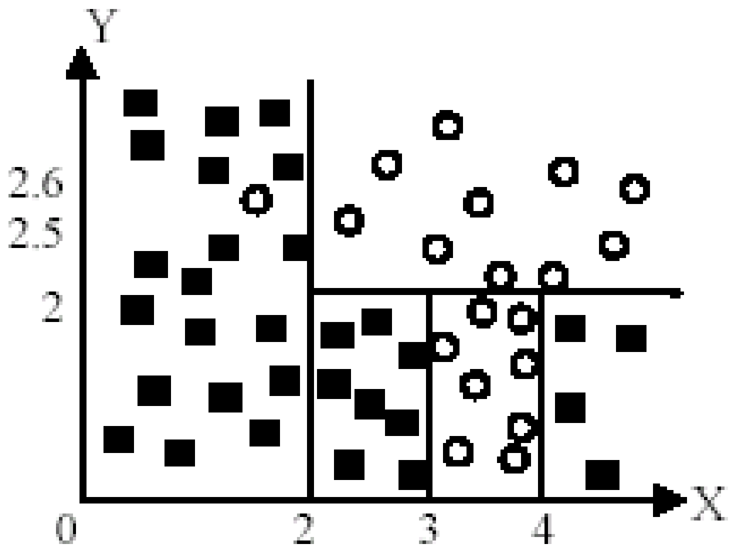
Likely to overfit the data

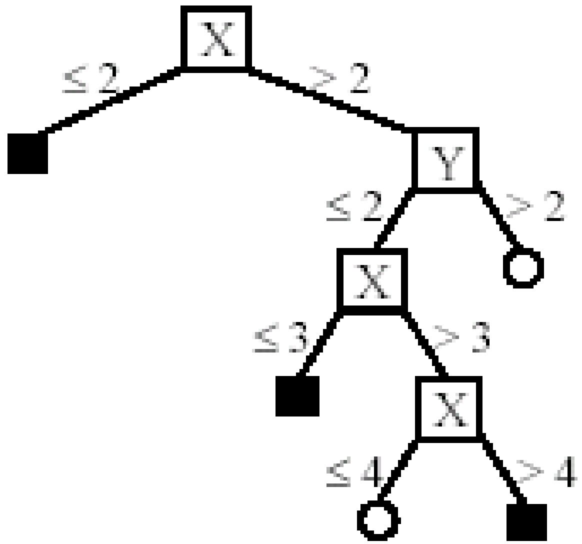


(A) A partition of the data space

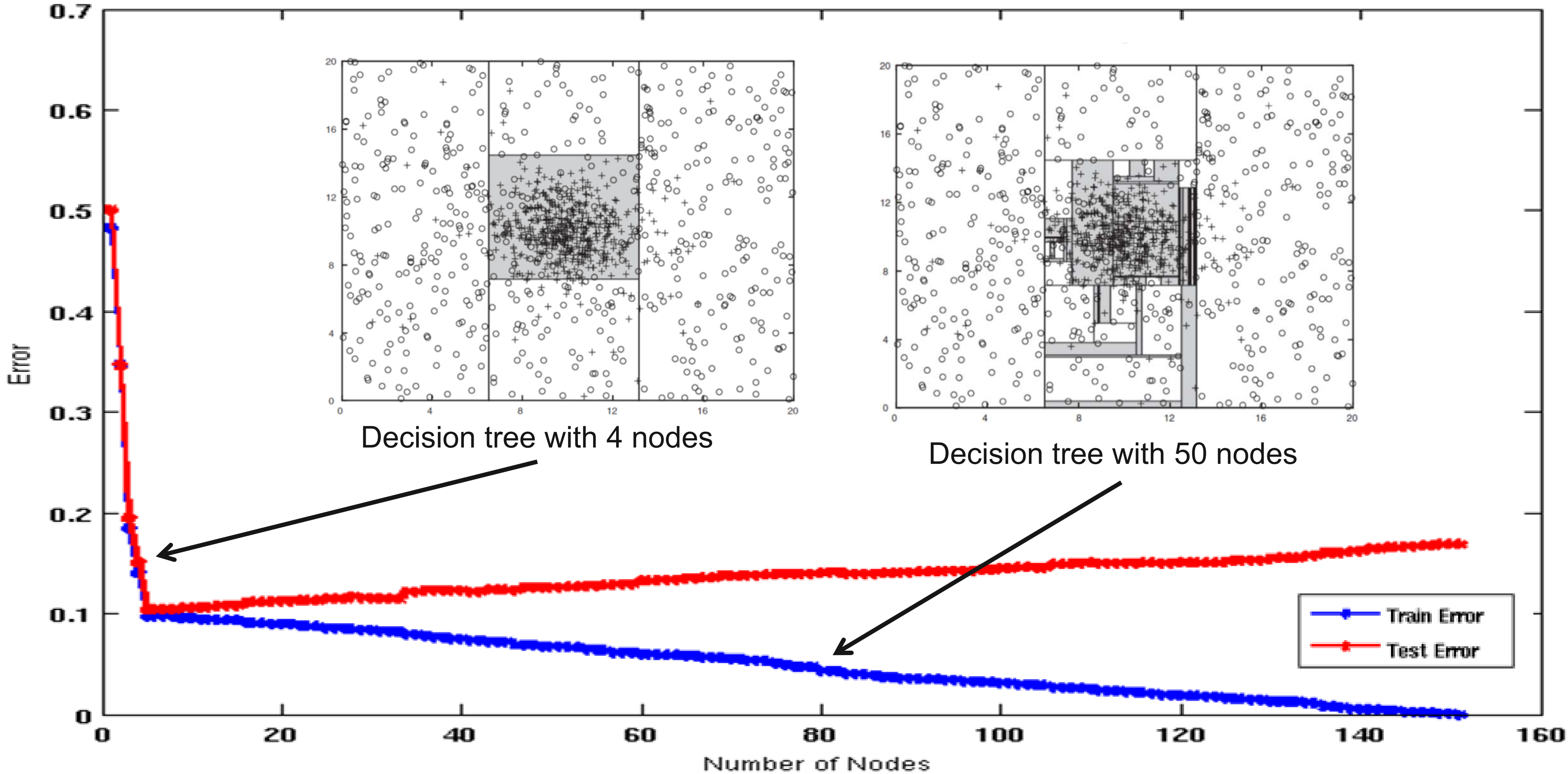


(B). The decision tree





Underfitting versus Overfitting Model

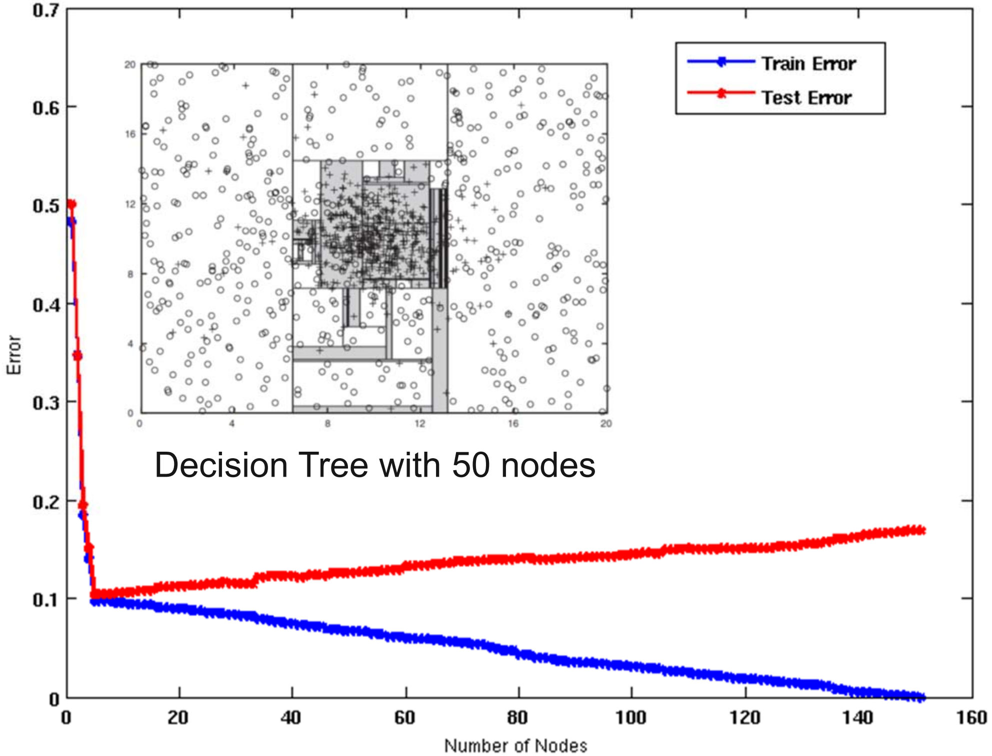


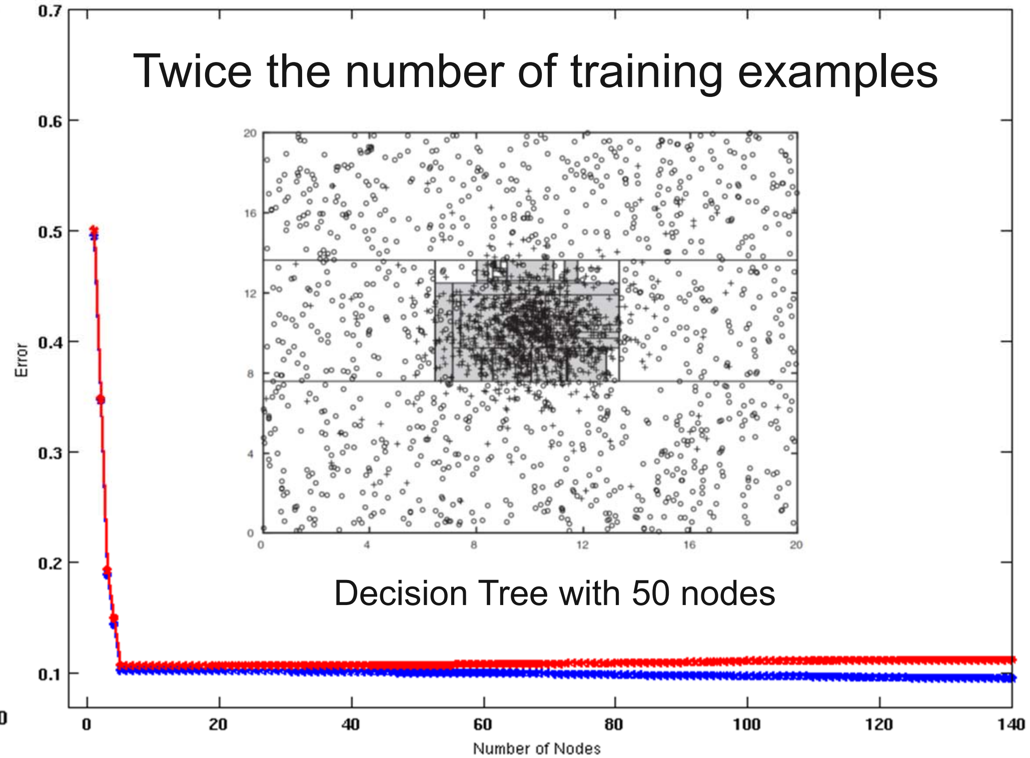
Underfitting: when model is too simple, both training and test errors are large

Overfitting: when model is too complex, training error is small but test error is large

How to Address Overfitting?

– Add more training data!





- If training data is under-representative, testing errors increase and training errors decrease on increasing number of nodes
- Increasing the size of training data reduces the difference between training and testing errors at a given number of nodes

How to Address Overfitting?

– Pre-Pruning (Early Stopping)
  - Stop the algorithm before tree becomes fully-grown
  - Normal stopping conditions for a node (no pruning):
    - Stop if all instances belong to the same class
    - Stop if all the attribute values are the same
  - Early stopping conditions (pre-pruning):
    - Stop if number of instances within a leaf node is less than some user-specified threshold (e.g. leaf size < 4)
    - Stop if expanding the current node only slightly improves the impurity measure (e.g. gain < 0.01)

How to Address Overfitting?

– Post-Pruning

- Grow decision tree to its entirety
- **Subtree Replacement**
  1. Trim the nodes of the decision tree in a bottom-up fashion
  2. Estimate generalization error before and after trimming
     - using a validation set (see in two slides)
     - or by putting a penalty on model complexity (see in two slides)
  3. If generalization error improves after trimming, replace sub-tree by a leaf node. Class label of leaf node is determined from majority class of instances in the sub-tree
- **Subtree Raising**
  - Replace subtree with most frequently used branch

Examples of Post-Pruning

Decision Tree:

```
depth = 1 :
|  breadth > 7 : class 1
|  breadth <= 7 :
|  |  breadth <= 3 :
|  |  |  ImagePages > 0.375 : class 0
|  |  |  ImagePages <= 0.375 :
|  |  |  |  totalPages <= 6 : class 1
|  |  |  |  totalPages > 6 :
|  |  |  |  |  breadth <= 1 : class 1
|  |  |  |  |  breadth > 1 : class 0
|  |  width > 3 :
|  |  |  MultiIP = 0:
|  |  |  |  ImagePages <= 0.1333 : class 1
|  |  |  |  ImagePages > 0.1333 :
|  |  |  |  |  breadth <= 6 : class 0
|  |  |  |  |  breadth > 6 : class 1
|  |  |  MultiIP = 1:
|  |  |  |  TotalTime <= 361 : class 0
|  |  |  |  TotalTime > 361 : class 1
depth > 1 :
|  MultiAgent = 0:
|  |  depth > 2 : class 0
|  |  depth <= 2 :
|  |  |  MultiIP = 1: class 0
|  |  |  MultiIP = 0:
|  |  |  |  breadth <= 6 : class 0
|  |  |  |  breadth > 6 :
|  |  |  |  |  RepeatedAccess <= 0.0322 : class 0
|  |  |  |  |  RepeatedAccess > 0.0322 : class 1
|  MultiAgent = 1:
|  |  totalPages <= 81 : class 0
|  |  totalPages > 81 : class 1
```

Subtree Raising

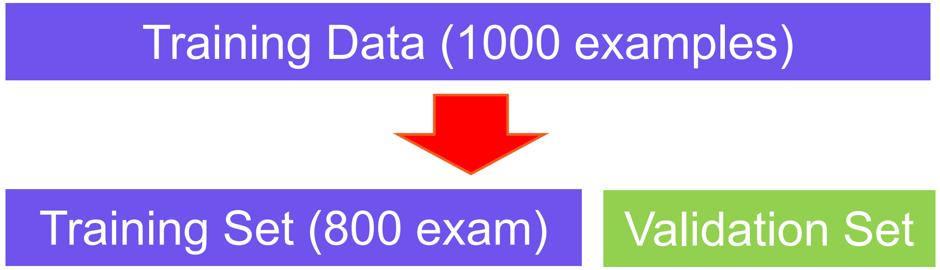
Simplified Decision Tree:

```
depth = 1 :
|  ImagePages <= 0.1333 : class 1
|  ImagePages > 0.1333 :
|  |  breadth <= 6 : class 0
|  |  breadth > 6 : class 1
depth > 1 :
|  MultiAgent = 0: class 0
|  MultiAgent = 1:
|  |  totalPages <= 81 : class 0
|  |  totalPages > 81 : class 1
```

Subtree Replacement

Validation Set versus Penalty on Model Complexity

– Validation Set



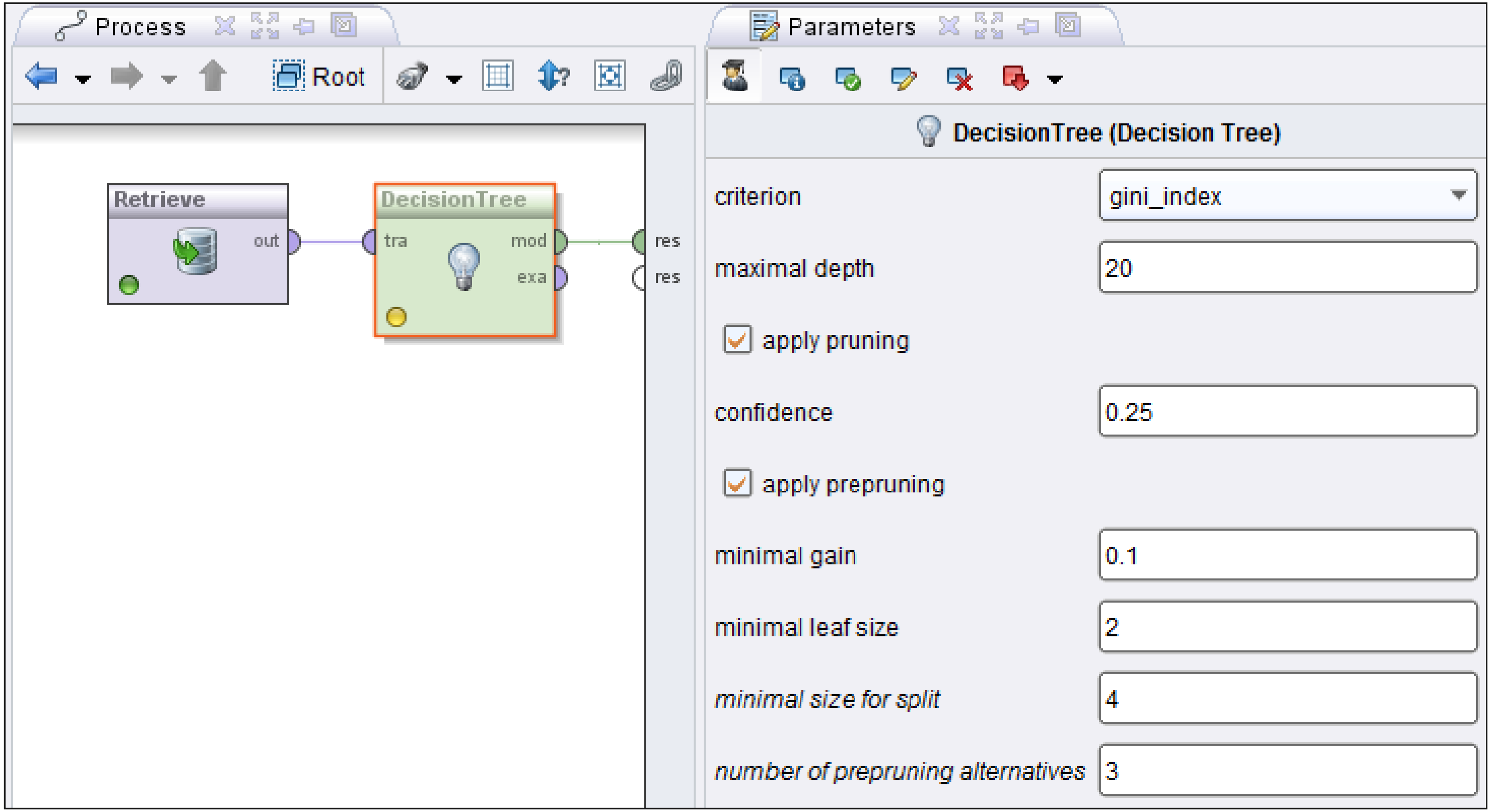
- Divide training data into two parts:
  - Training set:
    - use for model building
  - Validation set:
    - use for estimating generalization error
    - Note: validation set is not the same as test set (see Model Evaluation later)
- Drawback: Less data available for training

– Penalty on Model Complexity

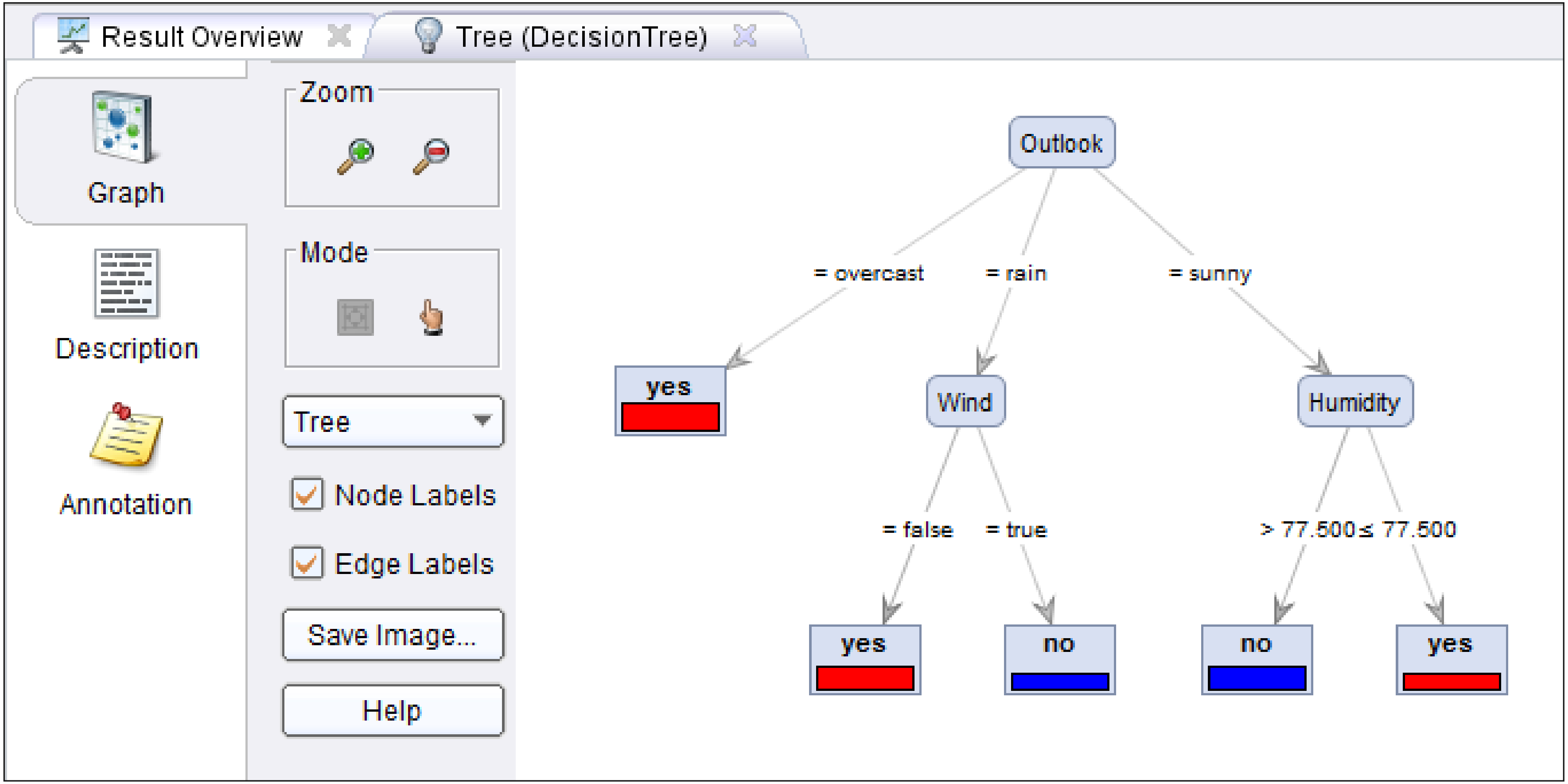
- Gen. Error(Model) = Train. Error(Model, Train. Data) + Complexity(Model)
- Complexity(Model) = α * number of leaf nodes
- User-provided penalty factor: e.g. $\alpha = 0.01$
- Advantage: All data available for training
- Drawlback: Unclear how to estimate α

Decision Trees in RapidMiner

The Decision Tree operator implements flexible learning algorithm which includes discretization, pre- and post-pruning.



Learned Decision Tree



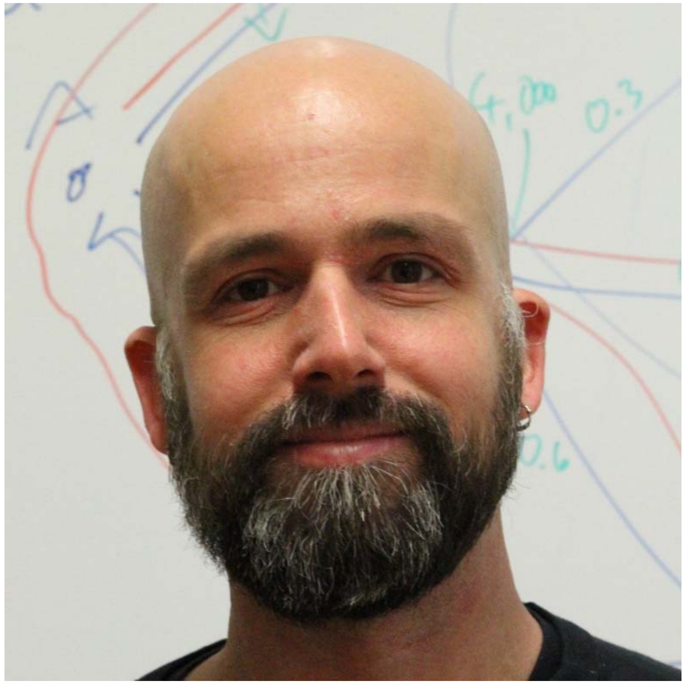
3.4 Discussion of Decision Trees

– Advantages
  - Inexpensive to construct
  - Extremely fast at classifying unknown records
  - Easy to interpret by humans for small-sized trees (eager learning)
  - Can easily handle redundant or irrelevant attributes
  - Accuracy is comparable to other classification techniques for many low dimensional data sets

– Disadvantages
  - Space of possible decision trees is exponentially large. Greedy approaches are often unable to find the best tree.
  - Can only represent decision boundaries parallel to the axes

Next week: Heiko Paulheim will give the Lecture.

1. What is Classification?
2. K-Nearest-Neighbors
3. Decision Trees
4. Model Evaluation
5. Rule Learning
6. Naïve Bayes
7. Artificial Neural Networks
8. Support Vector Machines
9. Parameter Tuning



Prof. Dr. Heiko Paulheim