Data Mining: Classification

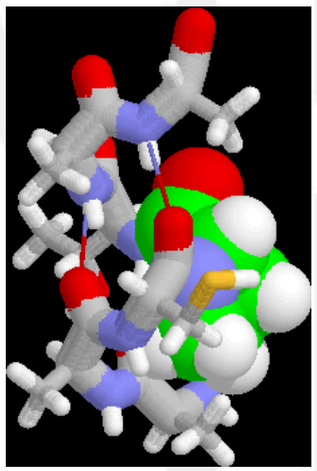
Tom Heskes

Classification: Definition

- Given a collection of records (*training set*)
  - Each record contains a set of *attributes*, one of the attributes is the *class*.
- Find a *model* for class attribute as a function of the values of other attributes.
- Goal: previously unseen records should be assigned a class as accurately as possible.
  - A *test set* is used to determine the accuracy of the model. Usually, the given data set is divided into training and test sets, with training set used to build the model and test set used to validate it.

Examples of Classification Task

- Predicting tumor cells as benign or malignant
- Classifying credit card transactions as legitimate or fraudulent
- Classifying secondary structures of protein as alpha-helix, beta-sheet, or random coil
- Categorizing news stories as finance, weather, entertainment, sports, etc



Classification Techniques

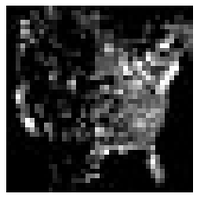
- Decision Tree based Methods
- Rule-based Methods
- Memory based reasoning
- Neural Networks
- Naïve Bayes and other Bayesian Networks
- Support Vector Machines
- ….

Classification tasks from the UCI ML repository

- See http://archive.ics.uci.edu/ml/
- Currently > 200 data sets for classification tasks

Hits

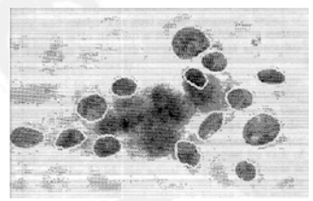
- **416585:** Iris



- **293122:** Adult

- **255322:** Wine

- **211707:** Breast Cancer Wisconsin (Diagnostic)



- **198442:** Car Evaluation

- **163045:** Abalone

Breast Cancer Wisconsin (Diagnostic)

- **Data Set Characteristics:** Multivariate
- **Number of Instances:** 569
- **Area:** Life Sciences
- **Attribute Characteristics:** Real
- **Number of Attributes:** 32
- **Date Donated:**1995-11-01
- **Associated Tasks:** Classification
- **Missing Values?** No
- **Source:** Dr. William H. Wolberg, General Surgery Dept. University of Wisconsin, Clinical Sciences Center
- **Data Set Information:** Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image.

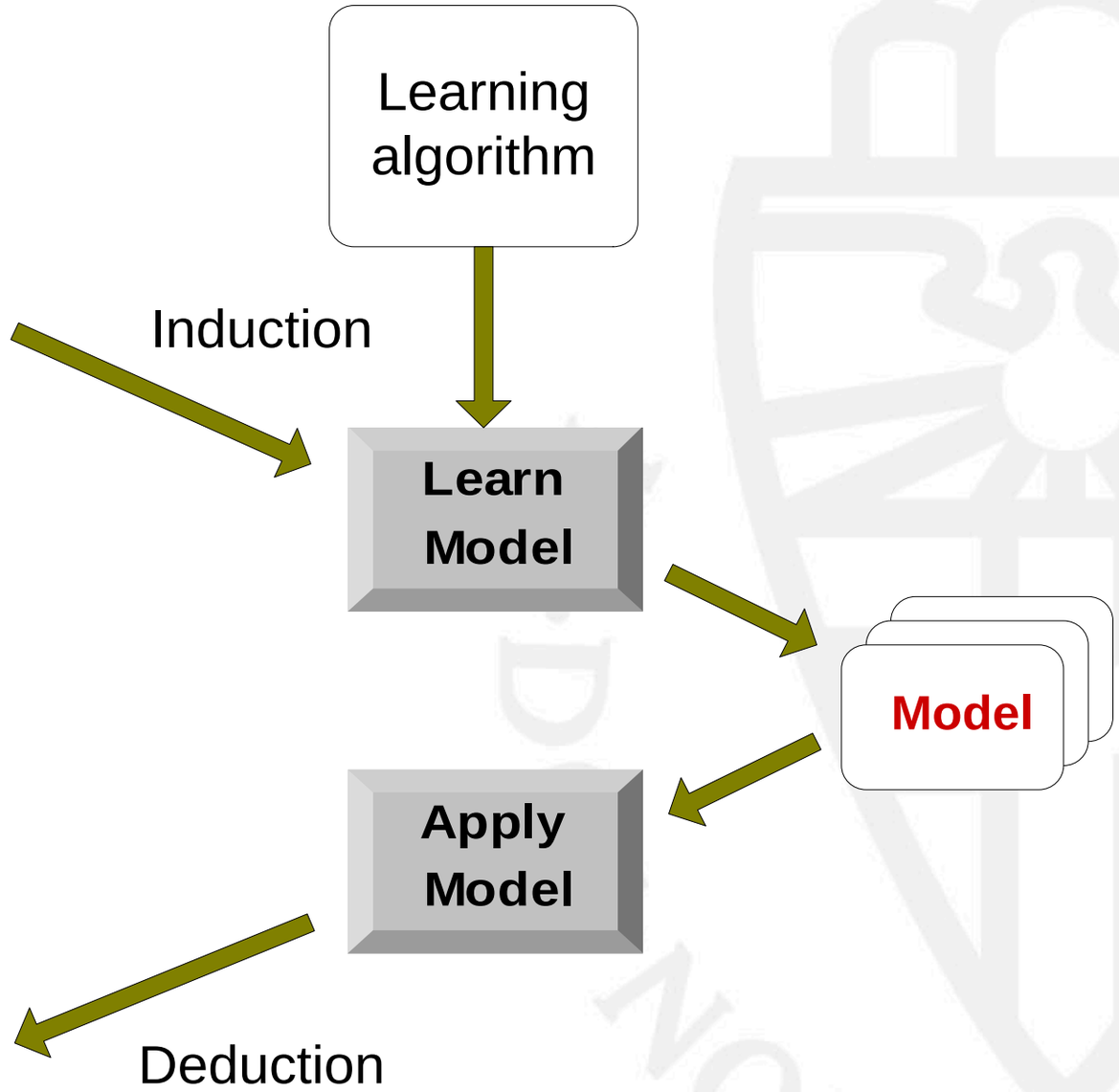
Classification Task

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	?

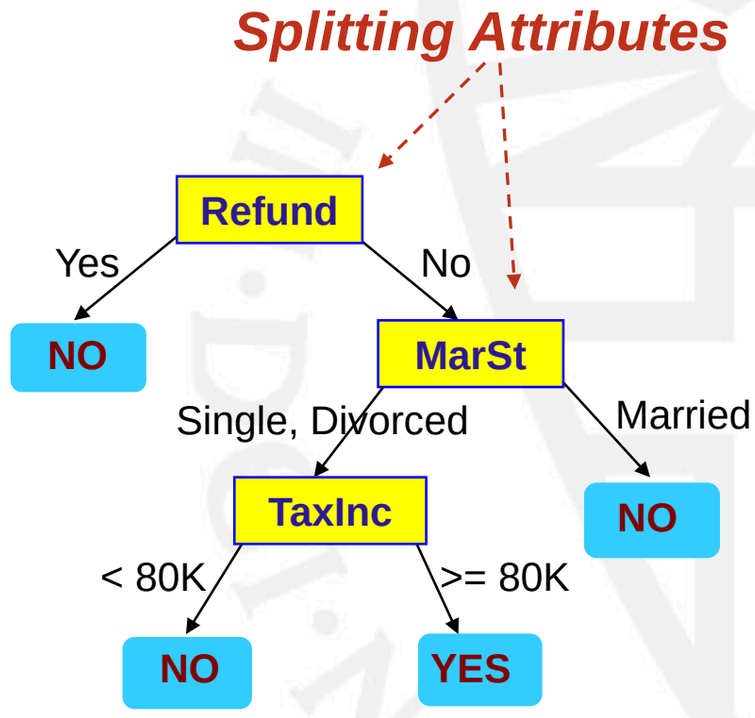
Test Set



Example of a Decision Tree

	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**

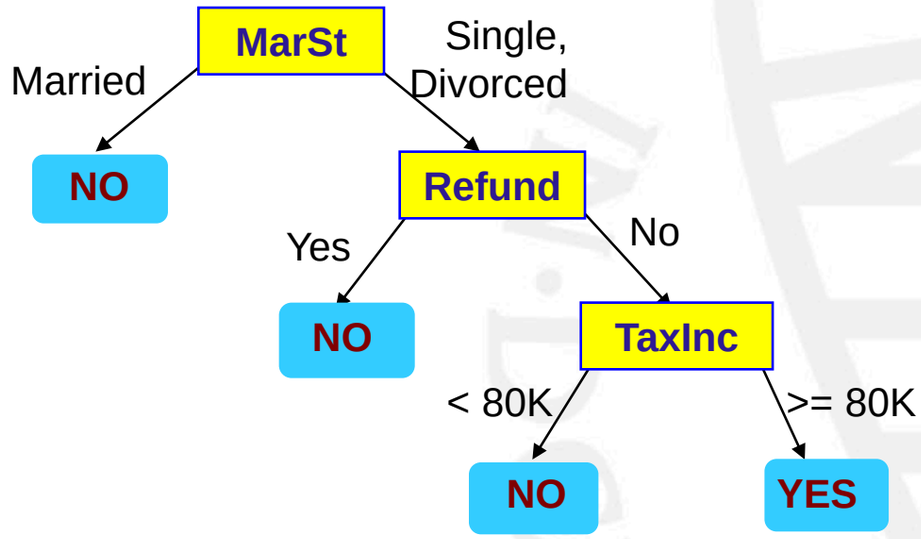
Training Data



Model: Decision Tree

Another Example of a Decision Tree

	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**



There can be more than one tree that fits the same data!

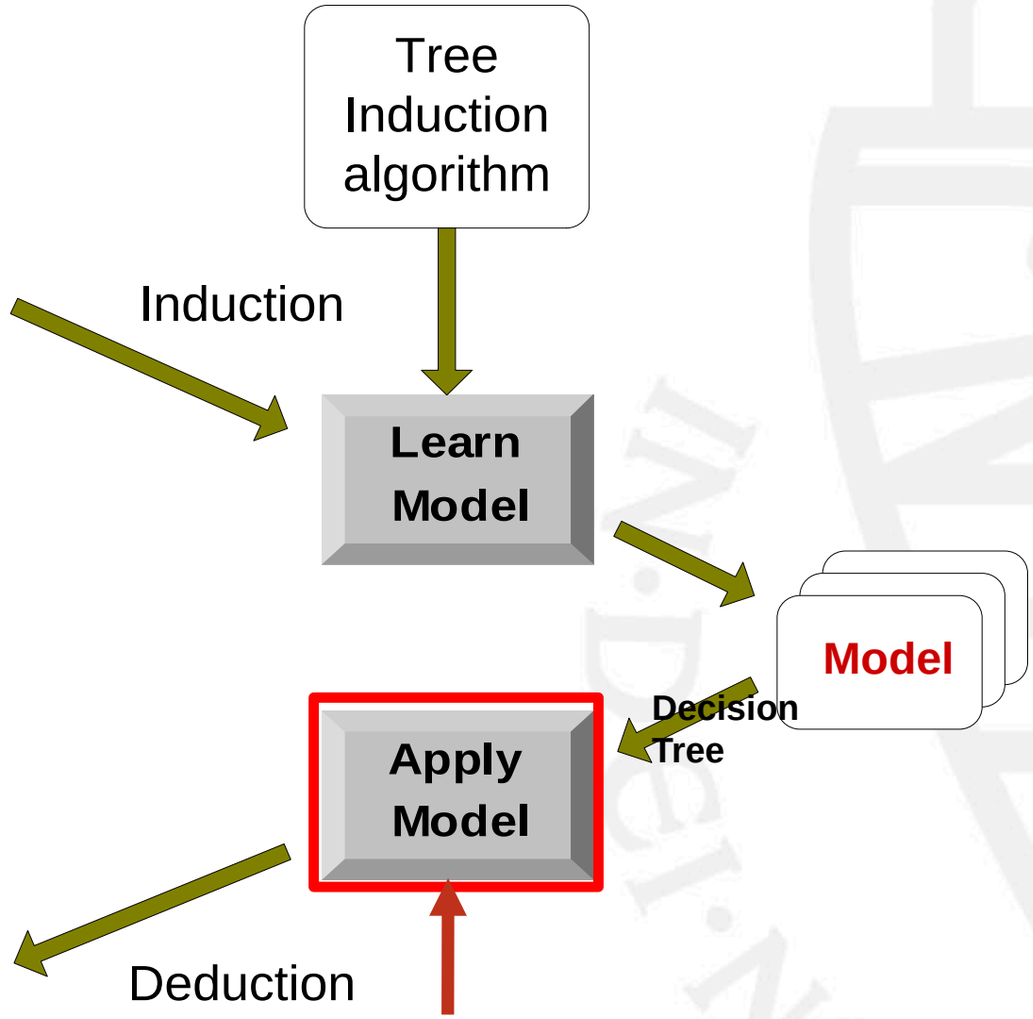
Decision Tree Classification Task

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

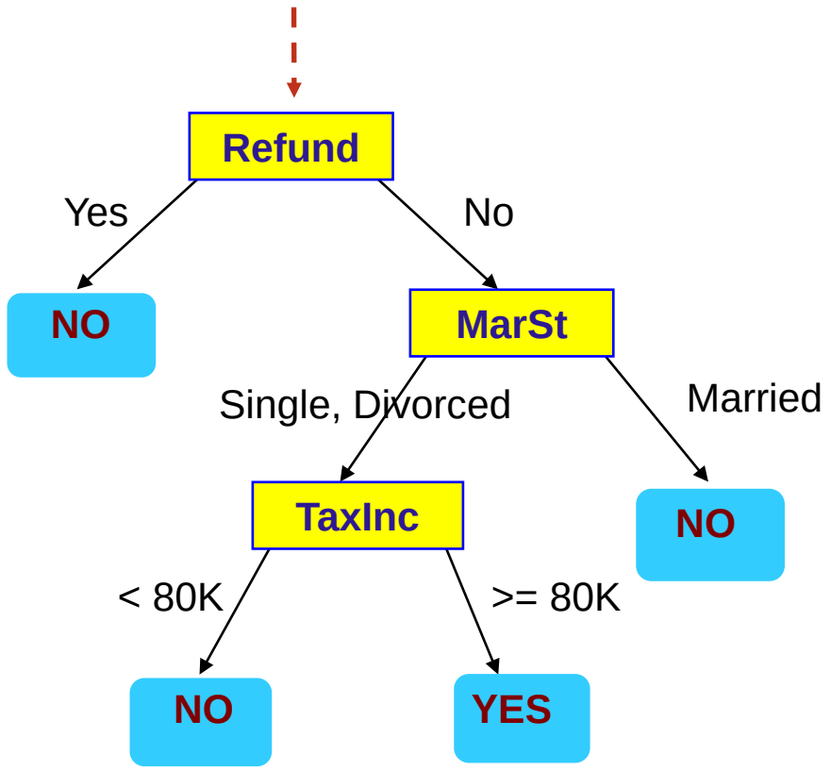
Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



Apply Model to Test Data

Start from the root of tree



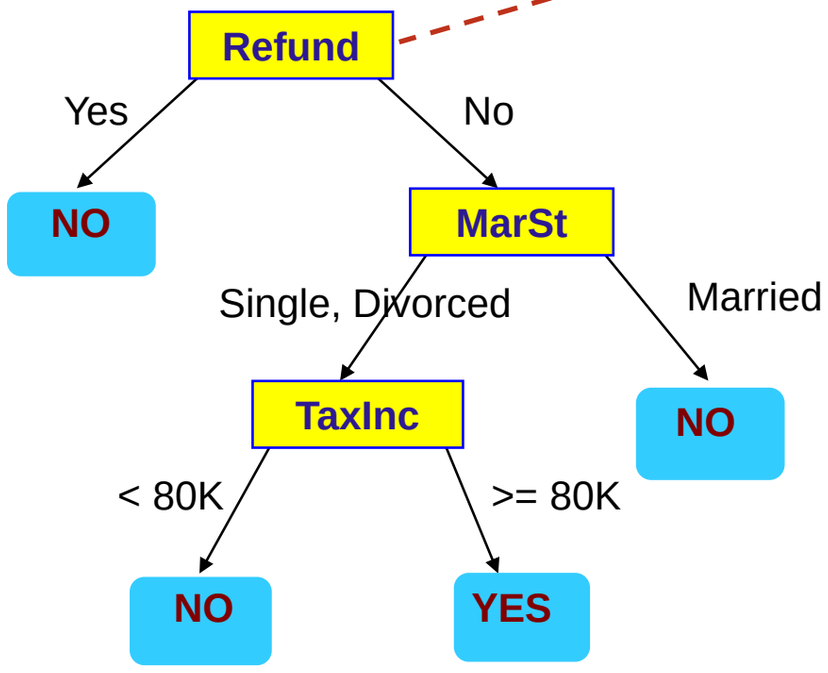
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Apply Model to Test Data

Test Data

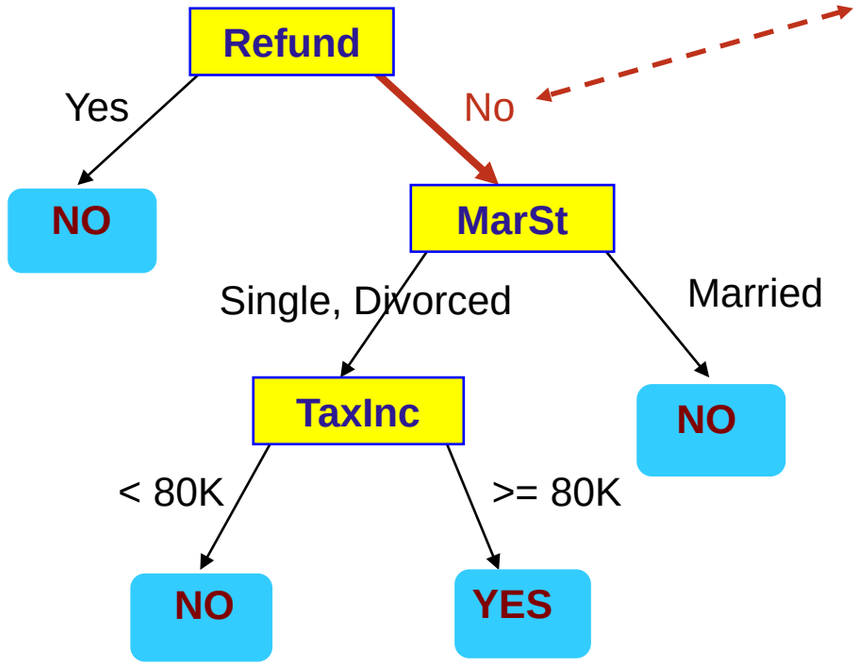
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data

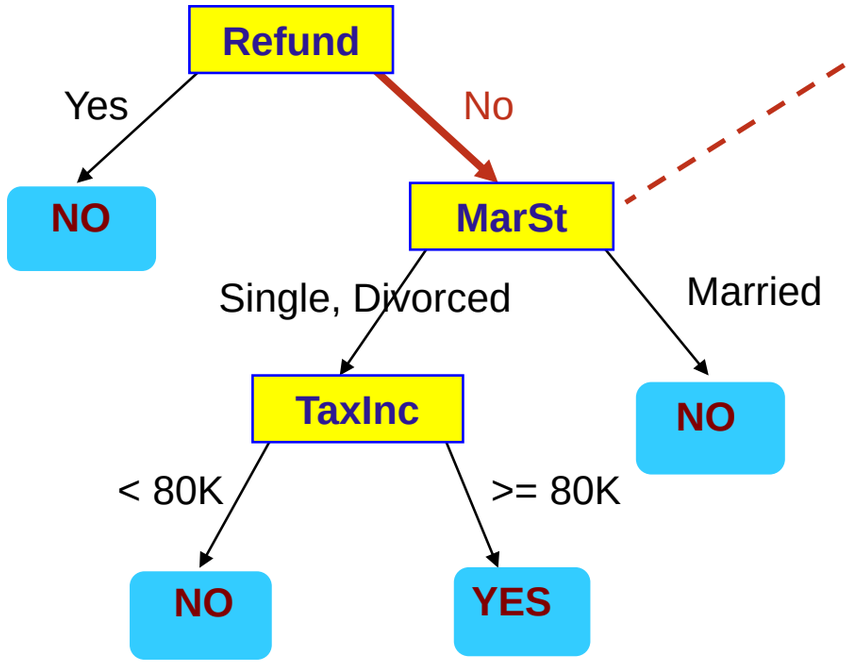
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

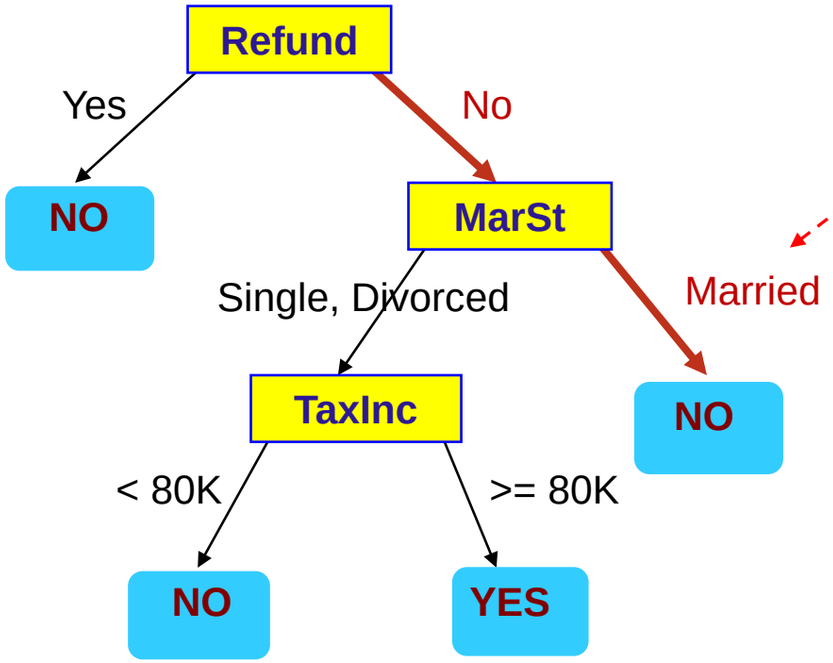


Apply Model to Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



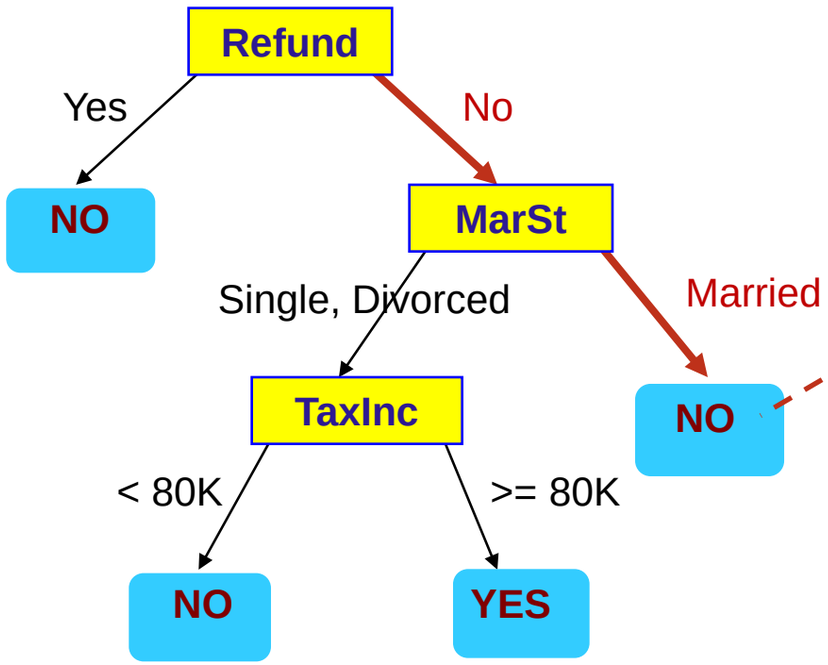
Apply Model to Test Data



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Apply Model to Test Data



Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Assign Cheat to “No”

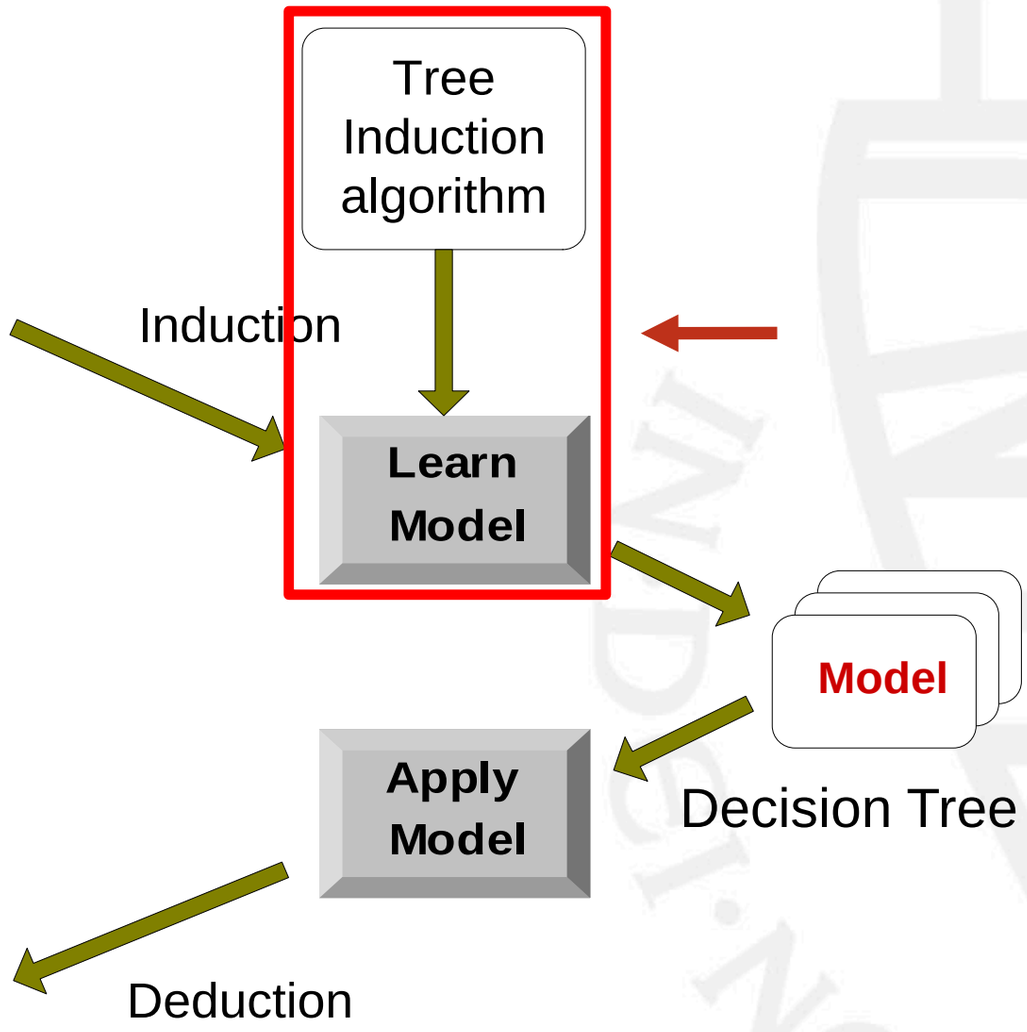
Decision Tree Classification Task

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	?

Test Set



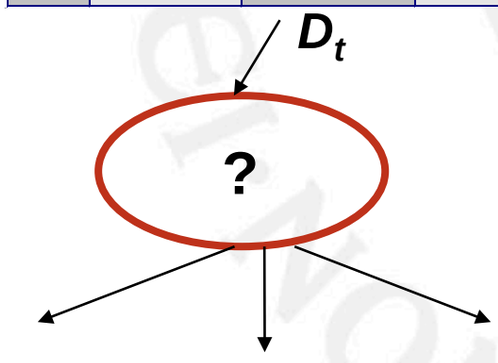
Decision Tree Induction

- Many Algorithms:
  - Hunt's Algorithm (one of the earliest)
  - CART
  - ID3, C4.5
  - SLIQ,SPRINT

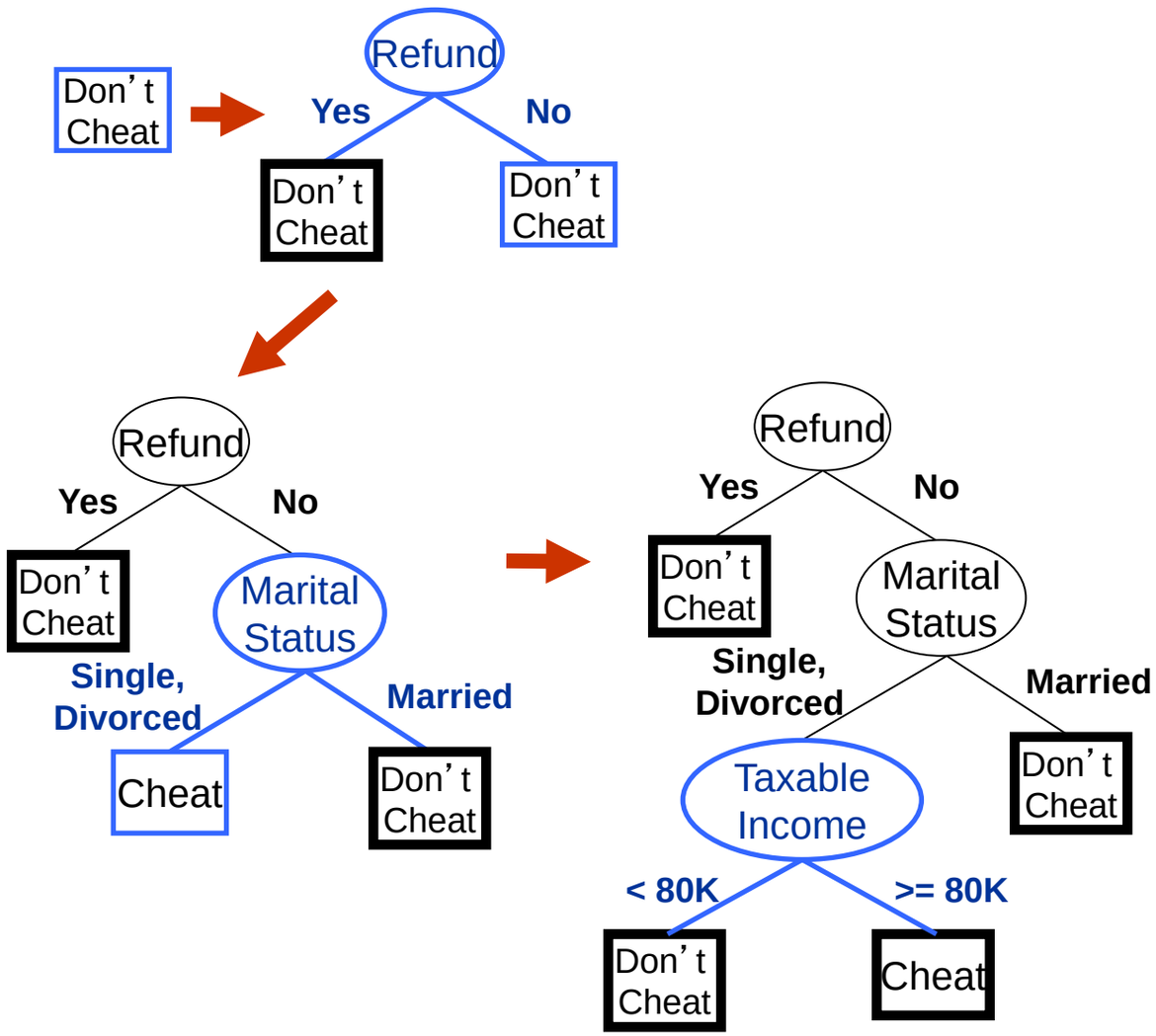
General Structure of Hunt's Algorithm

- Let D_t be the set of training records that reach a node t
- General Procedure:
  - If D_t contains records that belong the same class y_t, then t is a leaf node labeled as y_t
  - If D_t is an empty set, then t is a leaf node labeled by the default class computed as the majority vote of the class labels of the records in D_t
  - If D_t contains records that belong to more than one class then a) select best attribute to split the data into smaller subsets, b) recursively apply the procedure to each of the resulting subsets

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**



Hunt's Algorithm



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

Tree Induction

- Greedy strategy.
  - Split the records based on an attribute test that optimizes certain criterion.

- Issues
  - Determine how to split the records
    - How to specify the attribute test condition?
    - How to determine the best attribute to split on?
  - Determine when to stop splitting

Tree Induction

- Greedy strategy.
  - Split the records based on an attribute test that optimizes certain criterion.

- Issues
  - Determine how to split the records
    - How to specify the attribute test condition?
    - How to determine the best attribute to split on?
  - Determine when to stop splitting

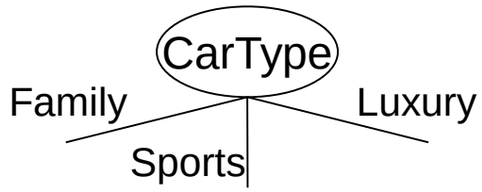
Splitting an Attribute: How to Specify Test Condition?

- Depends on attribute types
  - Nominal
  - Ordinal
  - Continuous

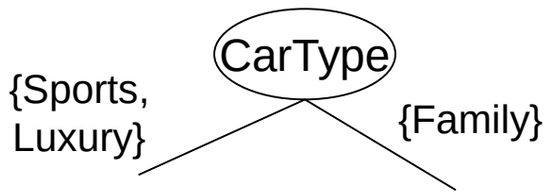
- Depends on number of ways to split
  - 2-way split
  - Multi-way split

Splitting Based on Nominal Attributes

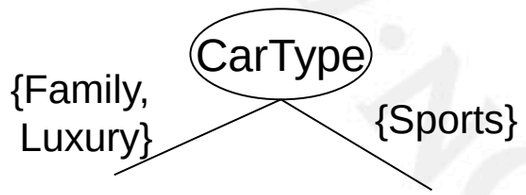
- Multi-way split: Use as many partitions as distinct values



- Binary split: Divides values into two subsets; find coherent partitioning

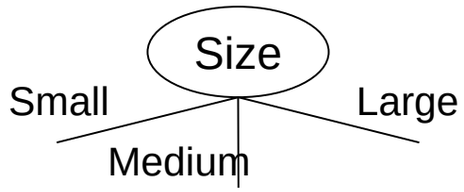


OR

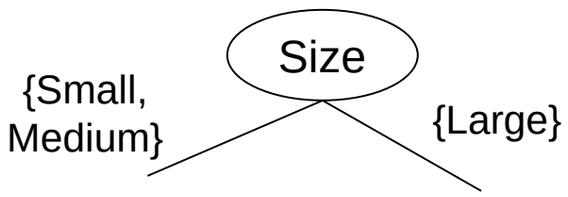


Splitting Based on Ordinal Attributes

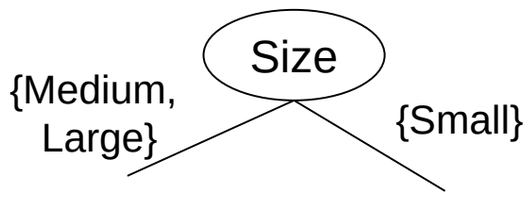
- Multi-way split: Use as many partitions as distinct values



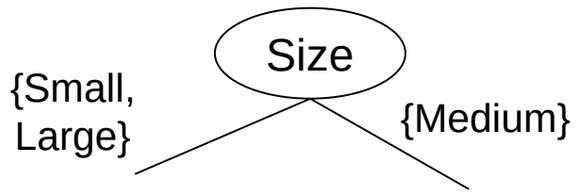
- Binary split: Divides values into two subsets; find optimal partitioning



OR

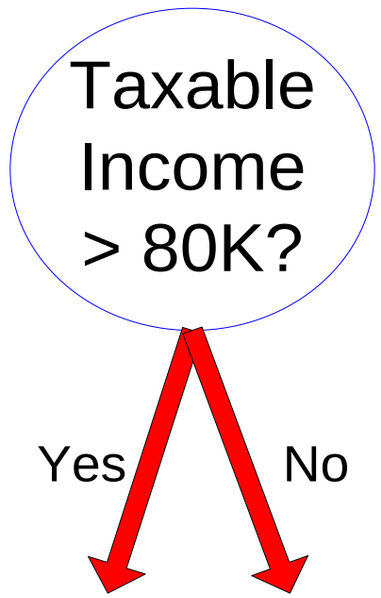


- What about this split?

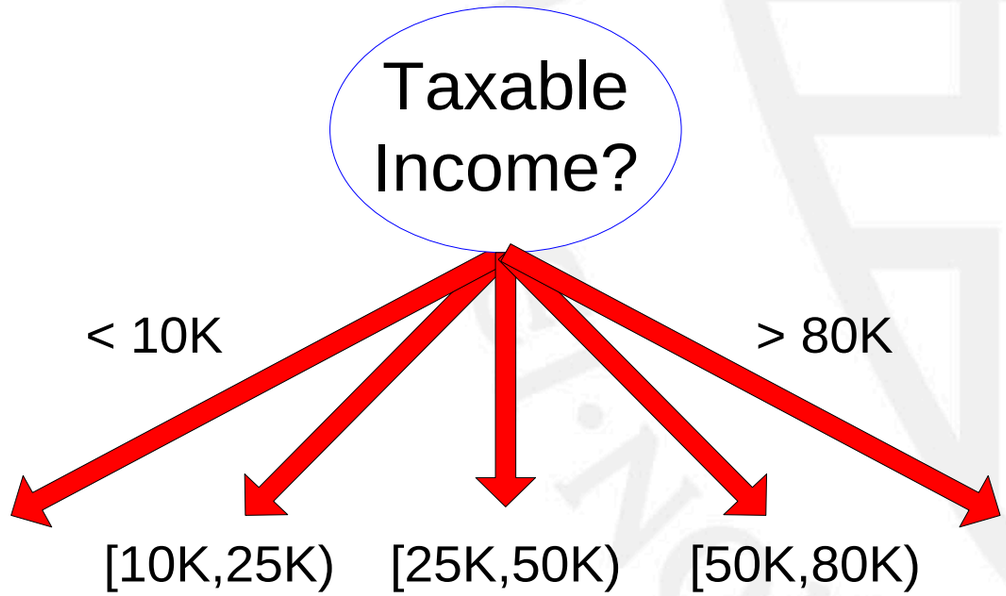


Splitting Based on Continuous Attributes

- Different ways of handling:
  - Discretization to form an ordinal categorical attribute
  - Binary Decision: ($A < v$) or ($A \geq v$)
- Consider all possible splits and finds the best cut
- Can be computationally intensive



(i) Binary split



(ii) Multi-way split

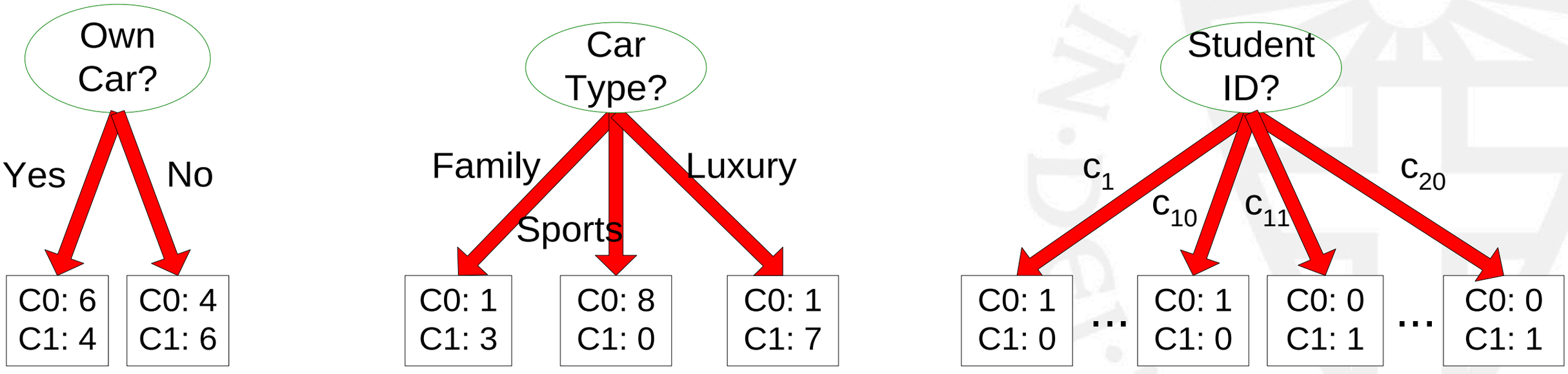
Tree Induction

- Greedy strategy
  - Split the records based on an attribute test that optimizes certain criterion.

- Issues
  - Determine how to split the records
    - How to specify the attribute test condition?
    - How to determine the best attribute to split on?
  - Determine when to stop splitting

How to determine the Best Split

Before Splitting: 10 records of class 0,
10 records of class 1



Which test condition is the best?

How to determine the Best Split

- Greedy approach:
  - Nodes with homogeneous class distribution are preferred
- Need a measure of node impurity:

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

Use Measures of Node Impurity

1. GINI index
2. Entropy
3. Misclassification error

Measure of Impurity: GINI Index

- GINI index for a given node *t* :

$$GINI(t) = 1 - \sum_{c} [p(c \mid t)]^2$$

  where *p(c|t)* is the relative frequency of class *c* at node *t*

- Maximum (1 - 1/n_c) when records are equally distributed among all classes, implying highest impurity.

- Minimum (0) when all records belong to one class, implying highest purity

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: Computing GINI of a Node

$$GINI(t) = 1 - \sum_c [p(c \mid t)]^2$$

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

Quality of a Split Based on GINI

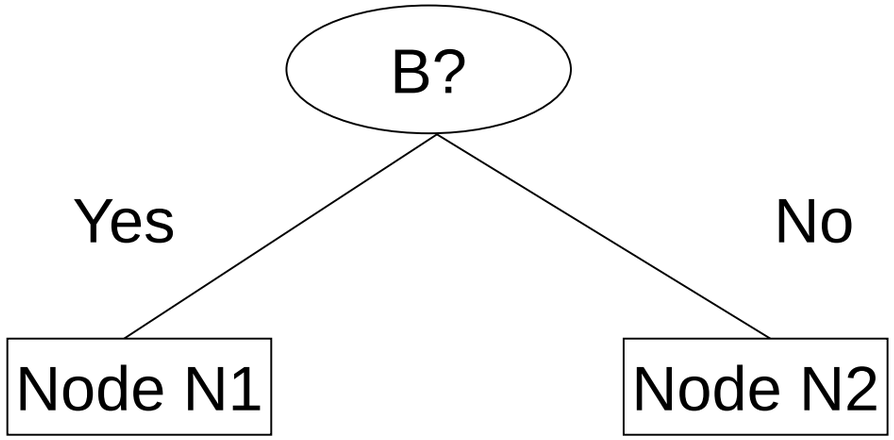
- Used in the decision tree algorithms CART, SLIQ, SPRINT.
- When a node *p* is split into *k* partitions (children nodes), the quality of split is computed as

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

where n_i = number of records at child node *i*,
n = number of records at node *p*

Computing GINI Index for a Binary Attribute

	Parent
C1	6
C2	6
Gini = 0.500	



	N1	N2
C1	5	1
C2	2	4
Gini=0.37		

Gini(N1)
$= 1 - (5/7)^2 - (2/7)^2$
$= 0.41$

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

Gini(Children)
$= 7/12 \times 0.41 +$
$5/12 \times 0.32$
$= 0.37$

GINI Index: Categorical Attributes

- For each distinct value, gather counts for each class in the dataset
- Use the count matrix to make decisions

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)

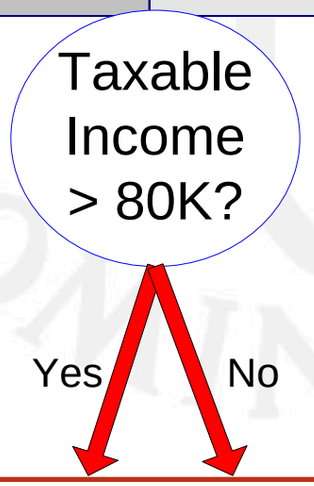
	CarType	
	{Sports, Luxury}	{Family}
C1	3	1
C2	2	4
Gini	0.400	

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

GINI Index: Continuous Attributes

- Use binary decisions based on one value
- Simple (inefficient) method to choose best splitting value:
  - For each splitting value *v*, scan the database to gather the class count matrix and compute its Gini index
  - Select a *v* with the smallest Gini index

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**



Binary Split Selection: Continuous Attributes

- For each attribute *a*
  - Sort the values of occurring in the records at that node
  - Scan these values, each time updating the count matrix and computing the GINI index of that split
- Choose the split position that has the smallest GINI index

Cheat	No	No	No	Yes	Yes	Yes	No	No	No	No

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	

Measure of Impurity: Entropy

- Entropy at a given node *t*:

$$Entropy(t) = -\sum_{c} p(c \mid t) \log p(c \mid t)$$

where $p(c|t)$ is the relative frequency of class *c* at node *t*

- Measures homogeneity of a node
  - Maximum (log n_c) when records are equally distributed among all classes implying highest impurity
  - Minimum (0) when all records belong to one class, implying highest purity

Examples for Computing Entropy of a Node

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

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

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

Entropy = – 0 log 0 – 1 log 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

Quality of a Split Using Entropy: 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

- Measures reduction in entropy achieved when using *split* to partition the values of attribute *a*
- Choose the split with maximum GAIN
- Used in ID3 and C4.5
- Disadvantage: tends to prefer splits that result in large number of partitions, each being small but pure

Entropy of a vector: Matlab code

```
function result = ent(Y)
% Calculates the entropy of a vector of values

% Get frequency table
tab = tabulate(Y);
prob = tab(:,3) / 100;

% Filter out zero-entries
prob = prob(prob~=0);

% Compute entropy
result = -sum(prob .* log2(prob));
```

GainRATIO: Adjusted Information Gain

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

Parent node p is split into k partitions
n_i is the number of records in partition i

- Adjusts Information Gain by the entropy of the partitioning (*SplitINFO*)
- Higher entropy partitioning (large number of small partitions) is penalized!
- Used in C4.5
- Designed to overcome the disadvantage of Information Gain

Measure of Impurity: Classification Error

- Classification error at a node *t*:

$$Error(t) = 1 - \max_c p(c \mid t)$$

- Measures misclassification error made by a node
  - Maximum (1 - 1/n_c) when records are equally distributed among all classes, implying highest impurity
  - Minimum (0) when all records belong to one class, implying highest purity

Examples for Computing Error

$$Error(t) = 1 - \max_c p(c \mid t)$$

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

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

Error = 1 – max (0, 1) = 1 – 1 = 0

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

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

Error = 1 – max (1/6, 5/6) = 1 – 5/6 = 1/6

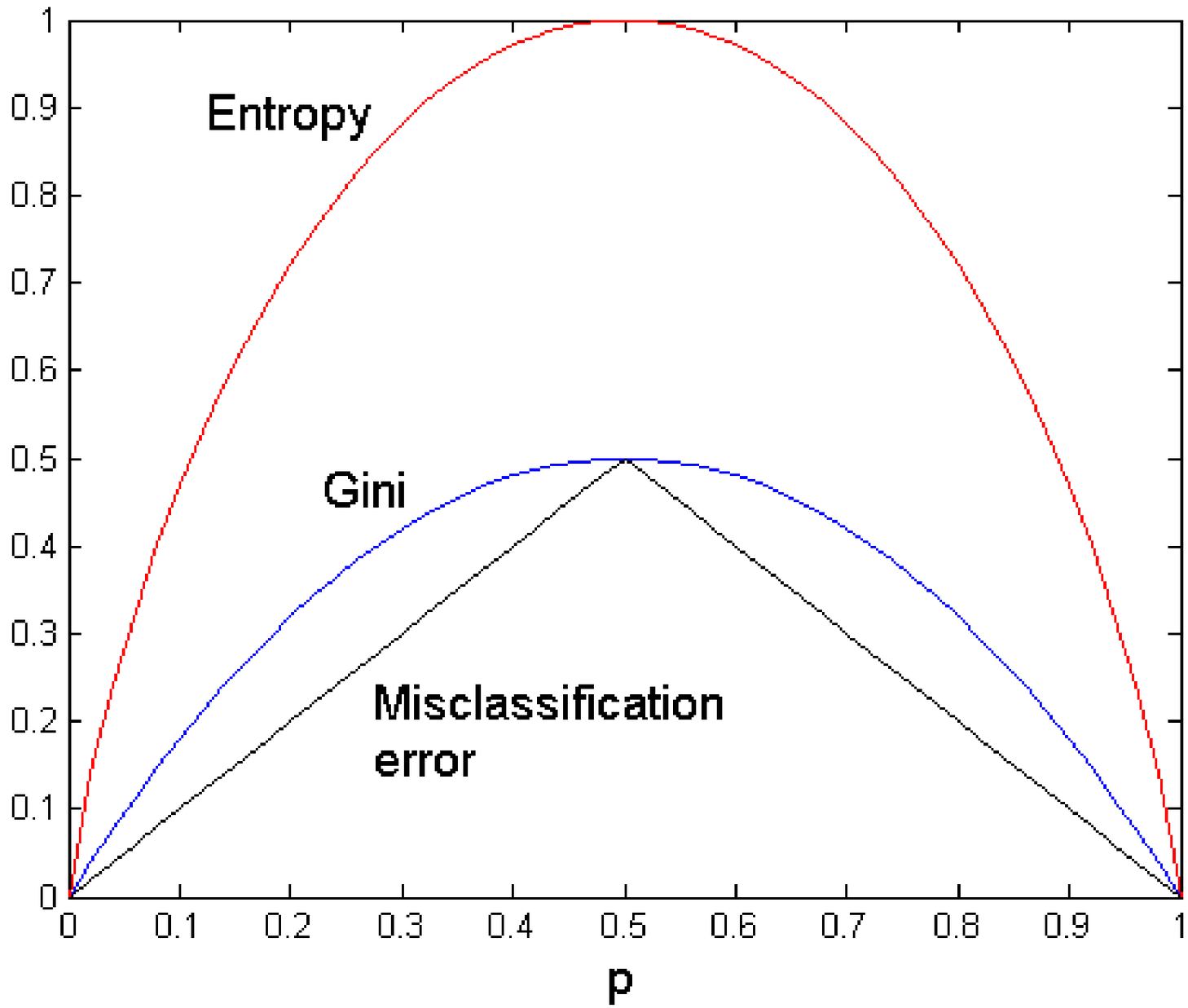
C1	**2**
C2	**4**

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

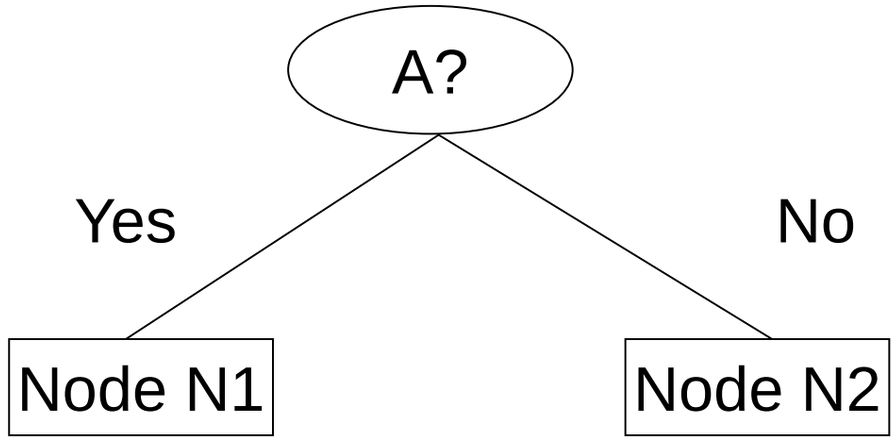
Error = 1 – max (2/6, 4/6) = 1 – 4/6 = 1/3

Comparison among Splitting Criteria

For a 2-class problem:



Misclassification Error vs Gini



	Parent
C1	7
C2	3
Gini = 0.42 Error = 0.3	

Gini(N1)
$= 1 - (3/3)^2 - (0/3)^2$
$= 0$

Gini(N2)
$= 1 - (4/7)^2 - (3/7)^2$
$= 0.4898$

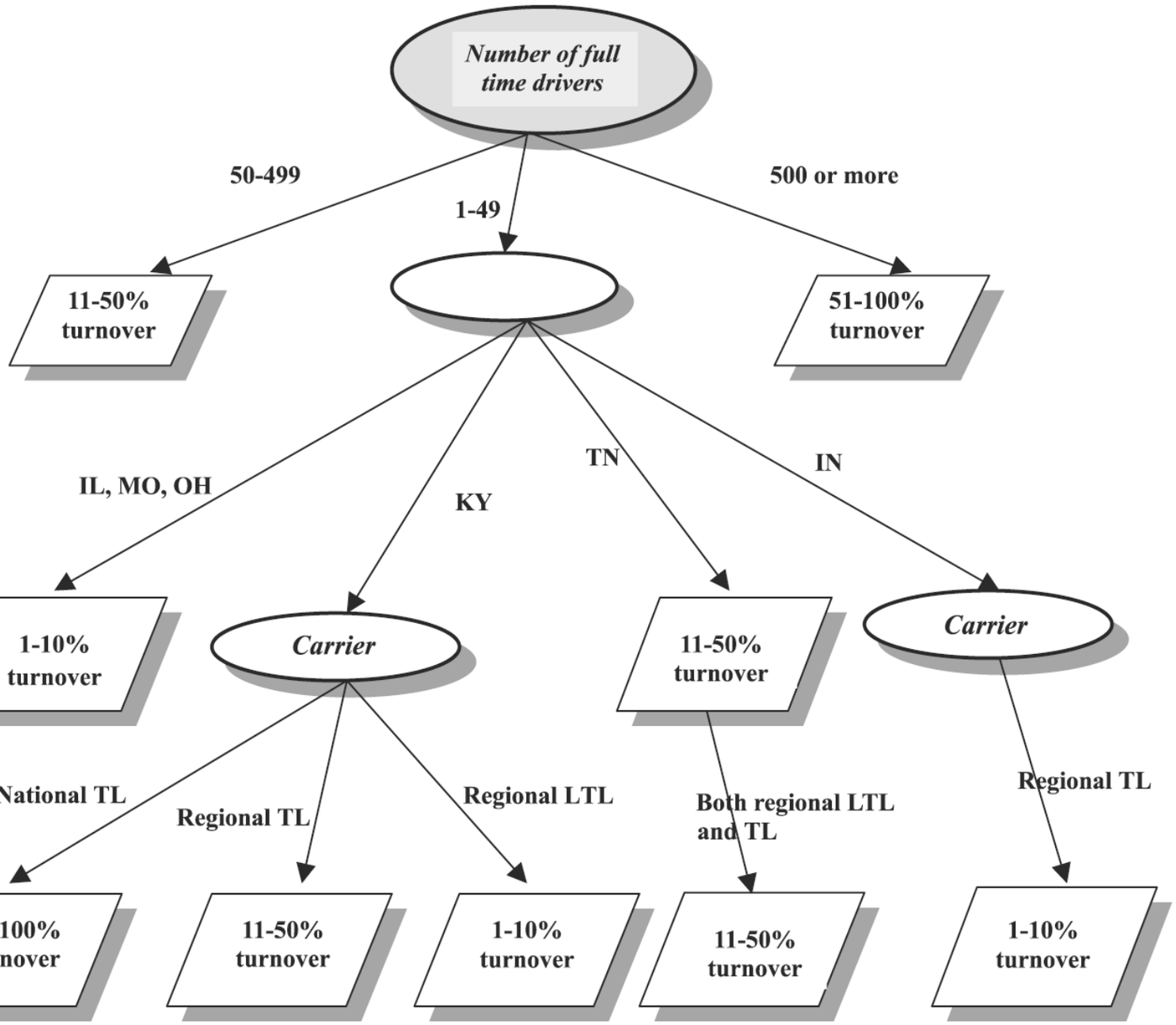
	N1	N2
C1	3	4
C2	0	3
Gini=0.3429 Error = 0.3		

Gini(Children)
$= 3/10 \times 0$
$+ 7/10 \times 0.4898$
$= 0.3429$

Stopping Criteria for Tree Induction

- Stop expanding a node when all the records belong to the same class
- Stop expanding a node when all the records have similar attribute values
- Early termination (to be discussed later)

Example: Truck Driver Turnover



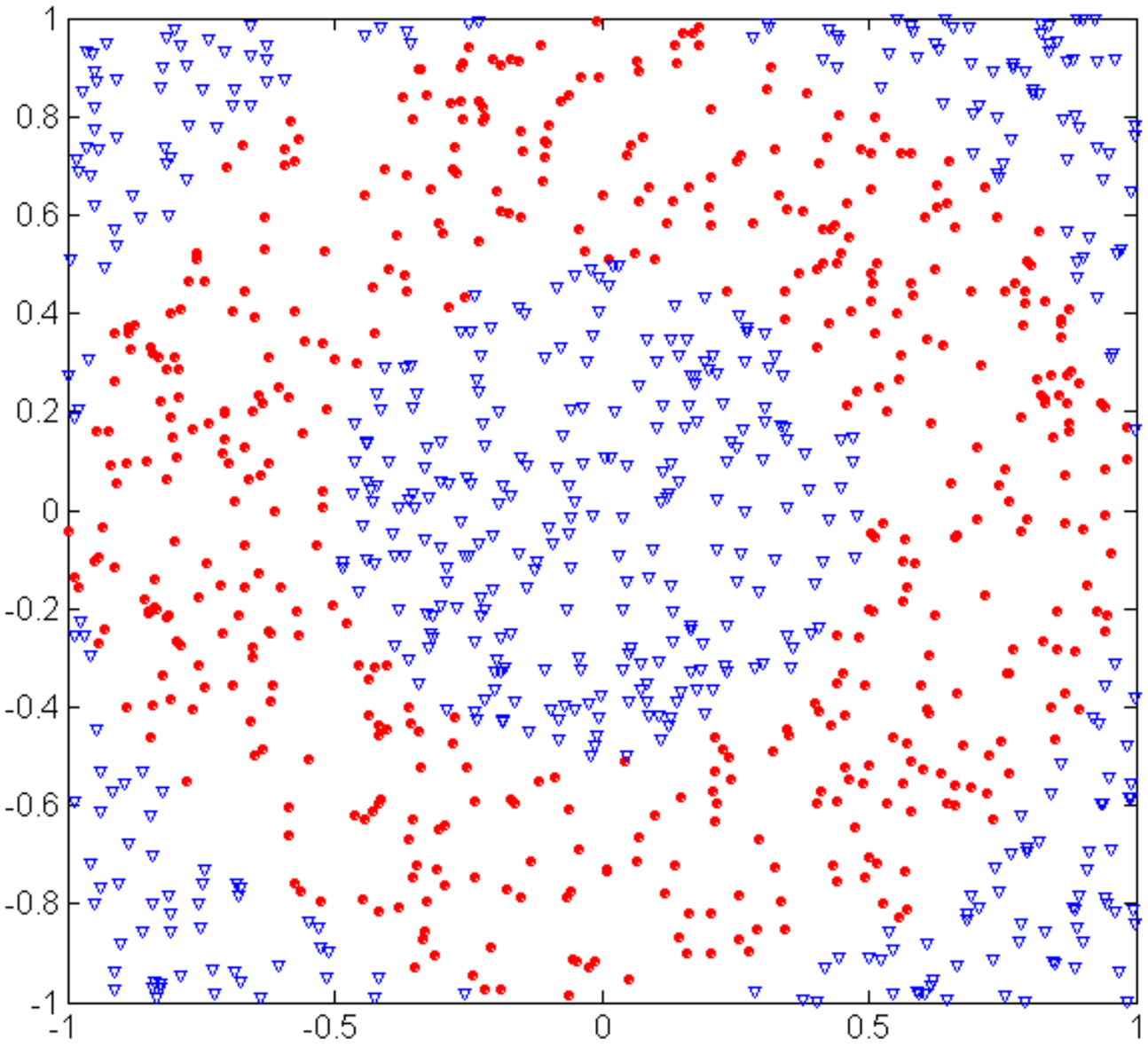
Decision Tree Based Classification: Advantages

- Inexpensive to construct
- Extremely fast at classifying unknown records
- Easy to interpret for small-sized trees
- Accuracy is comparable to other classification techniques for many simple data sets

Example: C4.5

- Simple depth-first construction
- Uses Information Gain
- Sorts continuous attributes at each node
- Needs entire data to fit in memory
- Unsuitable for large datasets
  - Needs out-of-core sorting
- Software freely available from:
  http://www.cse.unsw.edu.au/~quinlan/c4.5r8.tar.gz

Underfitting and Overfitting (Example)



500 circular and 500 triangular data points.

Circular points:

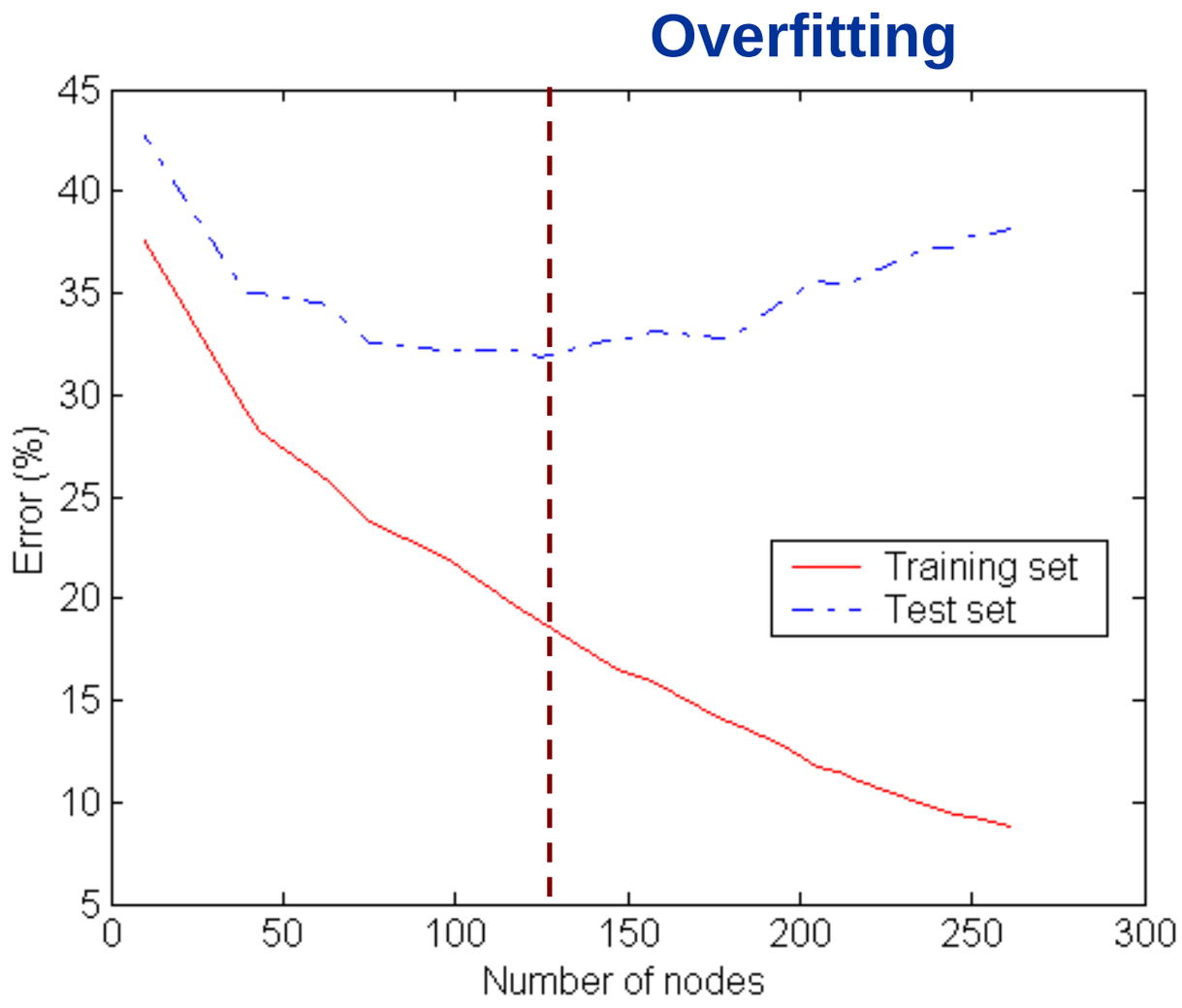
$0.5 \leq \text{sqrt}(x_1^2+x_2^2) \leq 1$

Triangular points:

$\text{sqrt}(x_1^2+x_2^2) < 0.5$ or

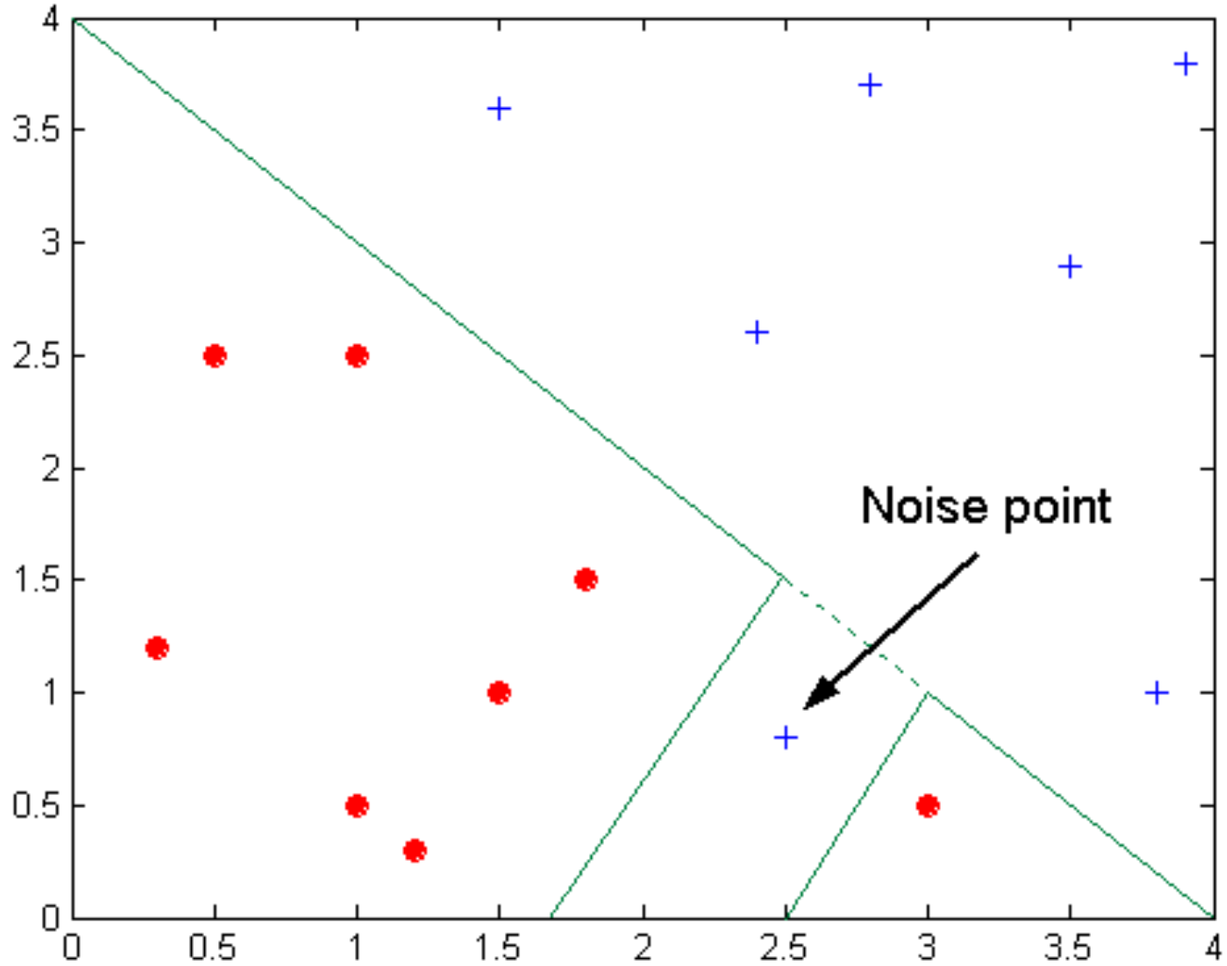
$\text{sqrt}(x_1^2+x_2^2) > 1$

Underfitting and Overfitting



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

Overfitting Due to Noise



Decision boundary is distorted by noise point

Notes on Overfitting

- Overfitting results in decision trees that are more complex than necessary
- Training error no longer provides a good estimate of how well the tree will perform on previously unseen records
- Need new ways for estimating errors

Estimating Generalization Error

- Training / re-substitution error: error on training data
- Generalization error: error on independent test data

- Methods for estimating generalization error
  - Optimistic approach: generalization error = training error

  - Pessimistic approach:
    - For each leaf node, add 0.5 to training error:

$$generalization\ error = training\ error + N \times 0.5$$

      with N the number of leaf nodes
    - For a tree with 30 leaf nodes and 10 errors out of 1000 training instances:
      training error = 10/1000 = 0.01; generalization error = $(10+30 \times 0.5)/1000 = 0.025$

  - Reduced error pruning (REP):
    - uses validation data set to estimate generalization error

Occam's Razor

- Given two models of similar generalization errors, one should prefer the simpler model over the more complex model

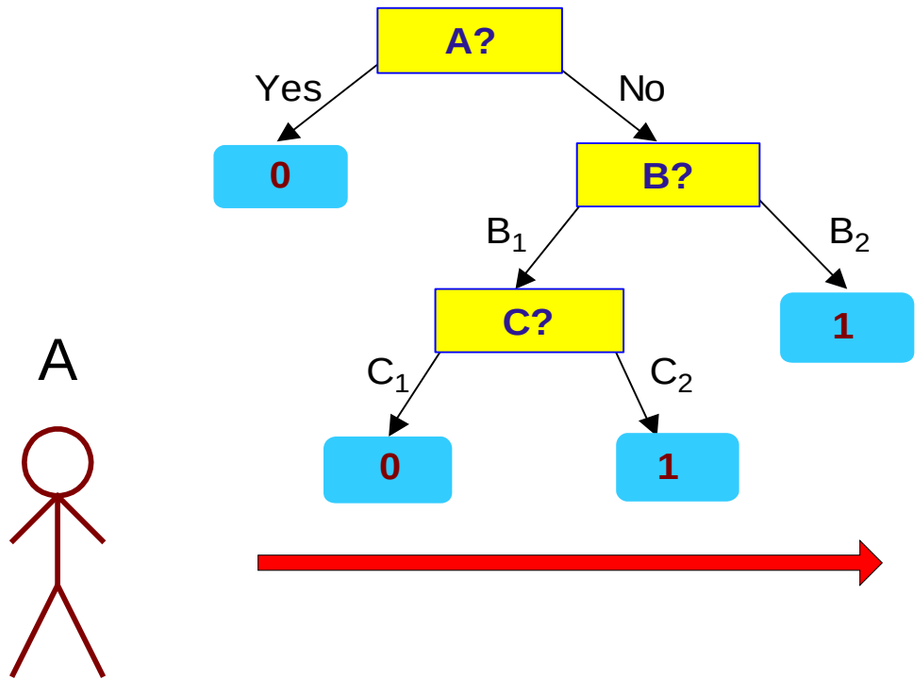
- For complex models, there is a greater chance that it was fitted accidentally by errors in data

- Therefore, one should include model complexity when evaluating a model

Ockham chooses a razor

Minimum Description Length (MDL)

X	y
X_1	1
X_2	0
X_3	0
X_4	1
…	…
X_n	1



X	y
X_1	?
X_2	?
X_3	?
X_4	?
…	…
X_n	?

- Cost(Model,Data) = Cost(Data|Model) + Cost(Model)
  - Cost is the number of bits needed for encoding
  - Search for the least costly model
- Cost(Data|Model) encodes the misclassification errors.
- Cost(Model) uses node encoding (number of children) plus splitting condition encoding

How to Address Overfitting: Pre Pruning

- Pre pruning (early stopping rules)
  - Stop the algorithm before it becomes a fully-grown tree

- Typical stopping conditions for a node:
  - Stop if all instances belong to the same class
  - Stop if all the attribute values are the same

- More restrictive conditions:
  - Stop if number of instances is less than some user-specified threshold
  - Stop if class distribution of instances are independent of the available features (e.g., using a χ^2 test)
  - Stop if expanding the current node does not improve impurity measures (e.g., Gini or information gain)

How to Address Overfitting: Post Pruning

- Post pruning
  - Grow decision tree to its entirety and then try to prune

- Trim the nodes of the decision tree in a bottom-up fashion

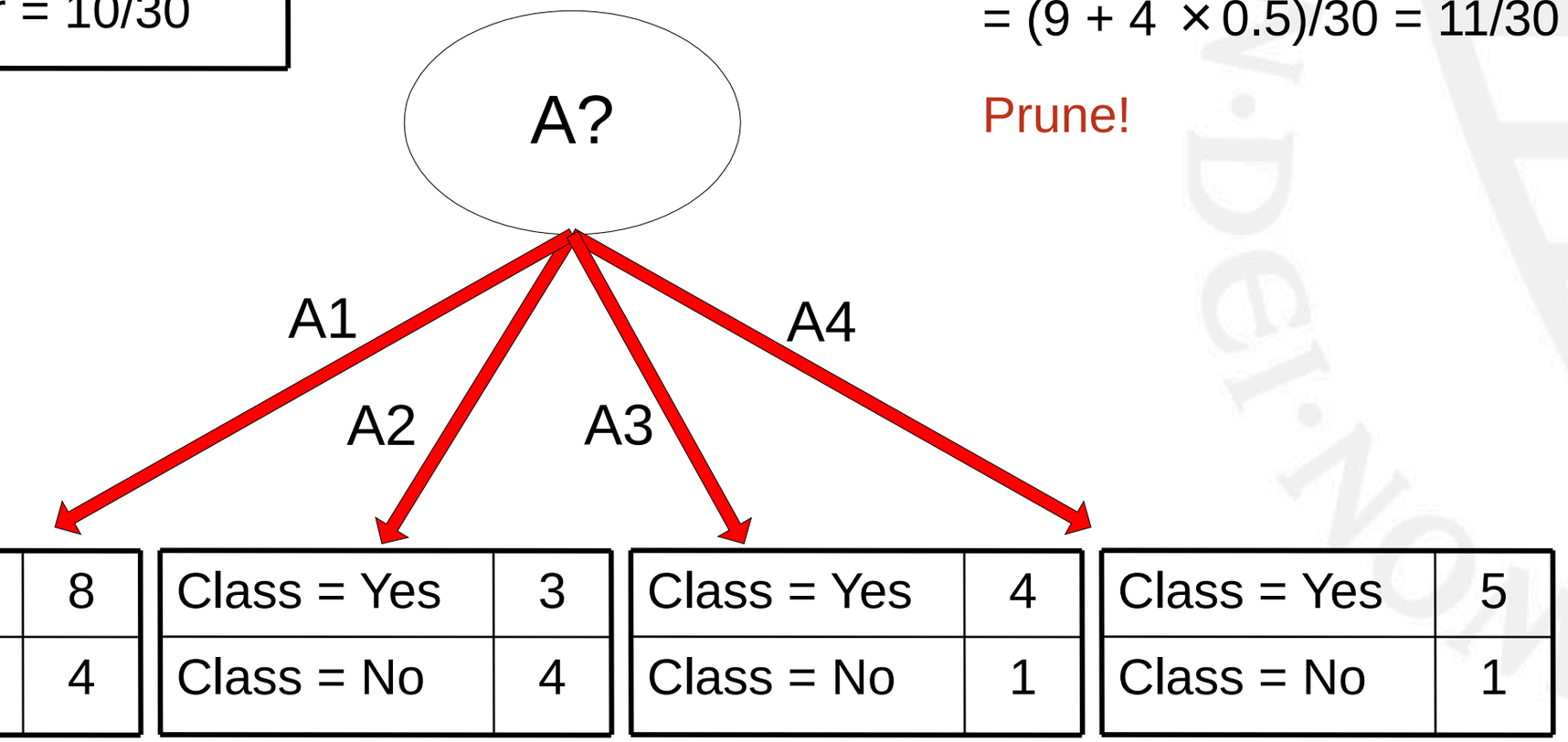
- If generalization error improves after trimming, replace sub-tree by a leaf node

- Class label of leaf node is determined from majority class of instances in the sub-tree that was removed

- Can use MDL for post-pruning

Example of Post-Pruning

Class = Yes	20
Class = No	10
Error = 10/30	



Class = Yes	8
Class = No	4

Class = Yes	3
Class = No	4

Class = Yes	4
Class = No	1

Class = Yes	5
Class = No	1

Training Error (Before splitting) = 10/30

Pessimistic error = (10 + 0.5)/30 = 10.5/30

Training Error (After splitting) = 9/30

Pessimistic error (After splitting)

= (9 + 4 ×0.5)/30 = 11/30

Prune!

Search Strategy

- Finding an optimal decision tree is NP-hard
- The algorithm presented so far uses a greedy, top-down, recursive partitioning strategy to induce a reasonable solution
- Other strategies?
  - Bottom-up
  - Bi-directional

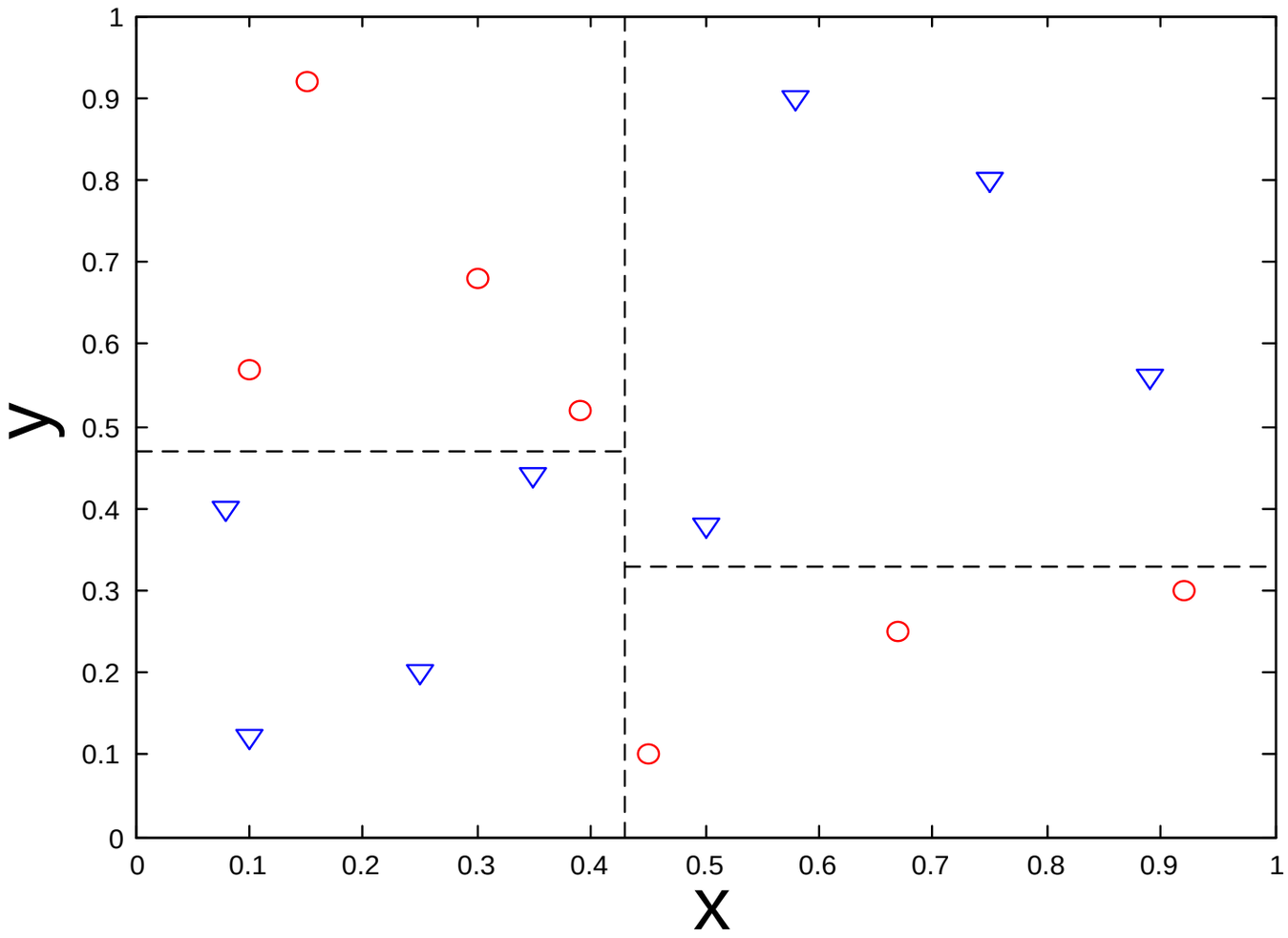
Expressiveness

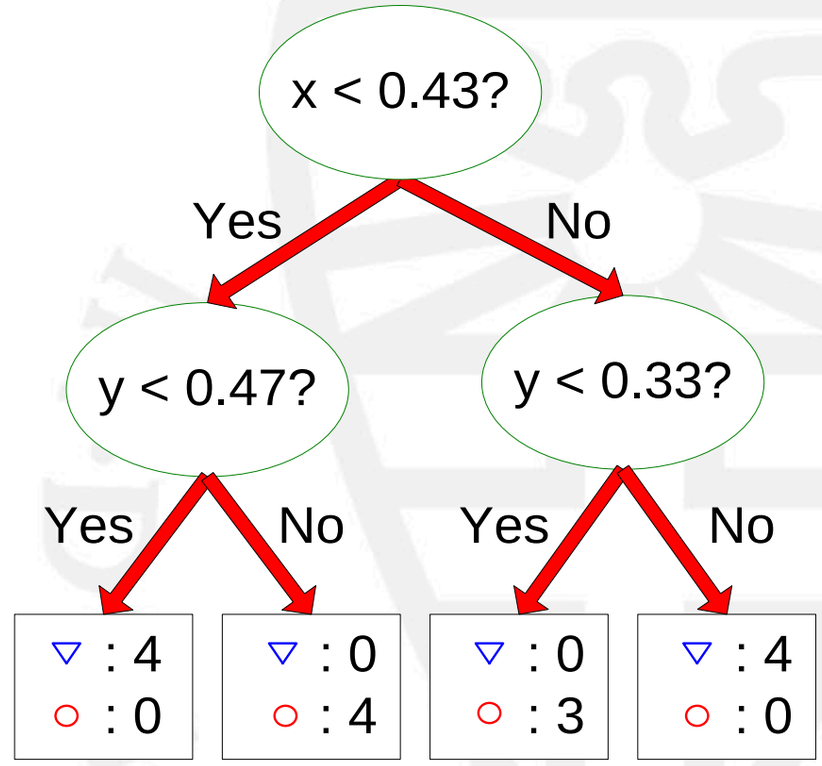
- Decision trees provide an expressive representation for learning discrete-valued function

- But they do not generalize well to all types of Boolean functions
  - E.g., parity function:
    - Class = 1 if there is an even number of Boolean attributes with truth value = True
    - Class = 0 if there is an odd number of Boolean attributes with truth value = True
  - Decision tree has depth equal to number of attributes…

- Not expressive enough for modeling continuous variables
  - Particularly when test condition involves only a single attribute at a time

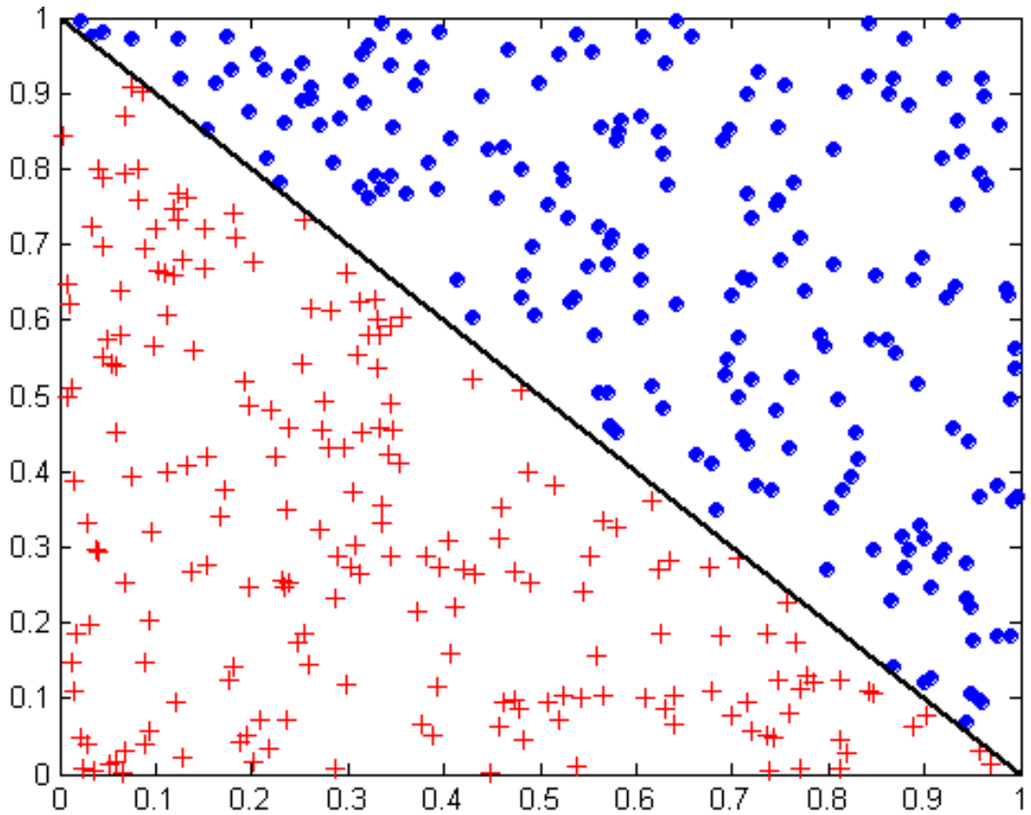
Decision Boundary

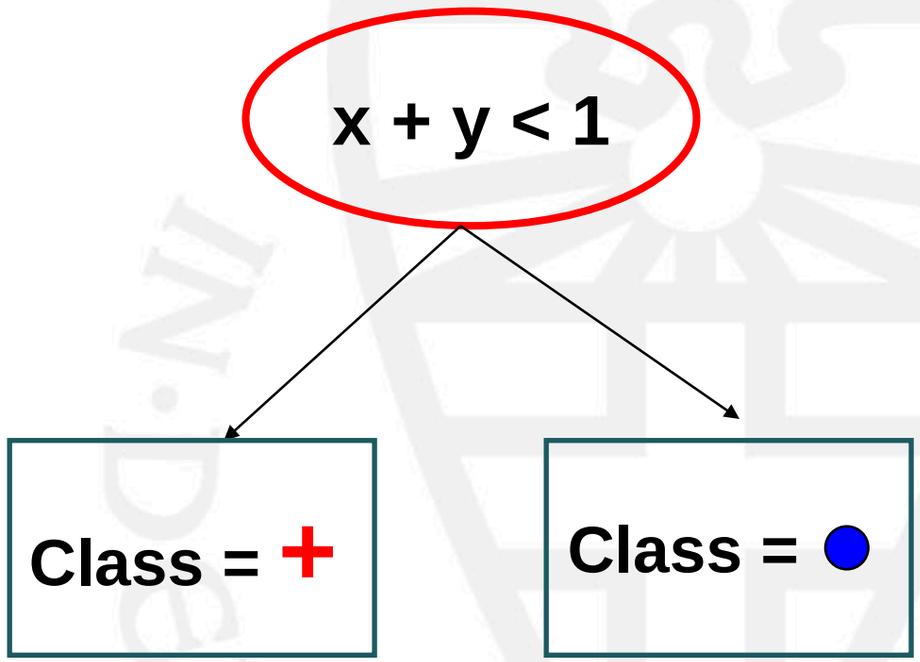




- Decision boundary is parallel to axes because test condition involves a single attribute at a time

Oblique Decision Trees





- Test condition may involve multiple attributes
- More expressive representation
- Finding optimal test condition is computationally expensive