Decision Trees

Mauro Sozio*

*slides adapted from the course: Introduction to data mining, Steinbach, Kumar

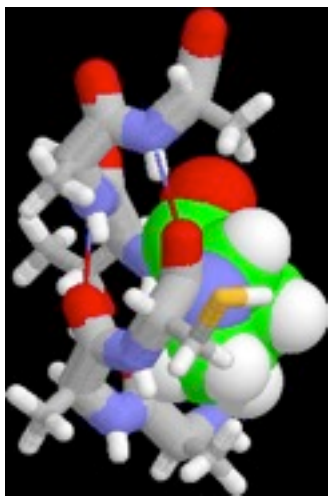
Classification: Definition

- 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

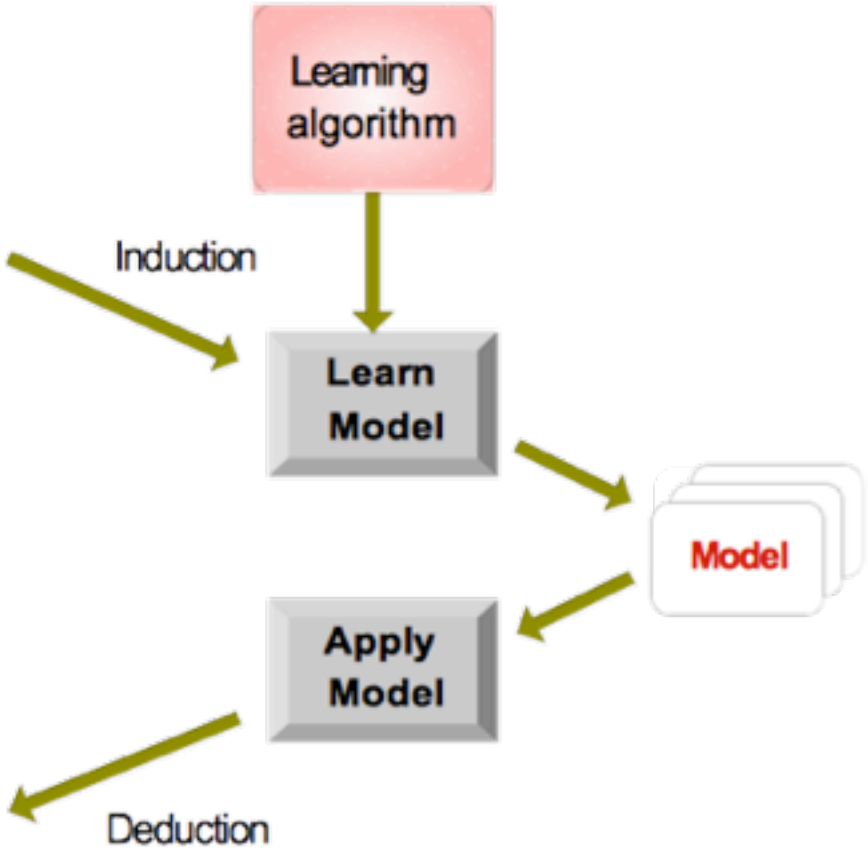
Illustrating Classification Task

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



Facts about Classification

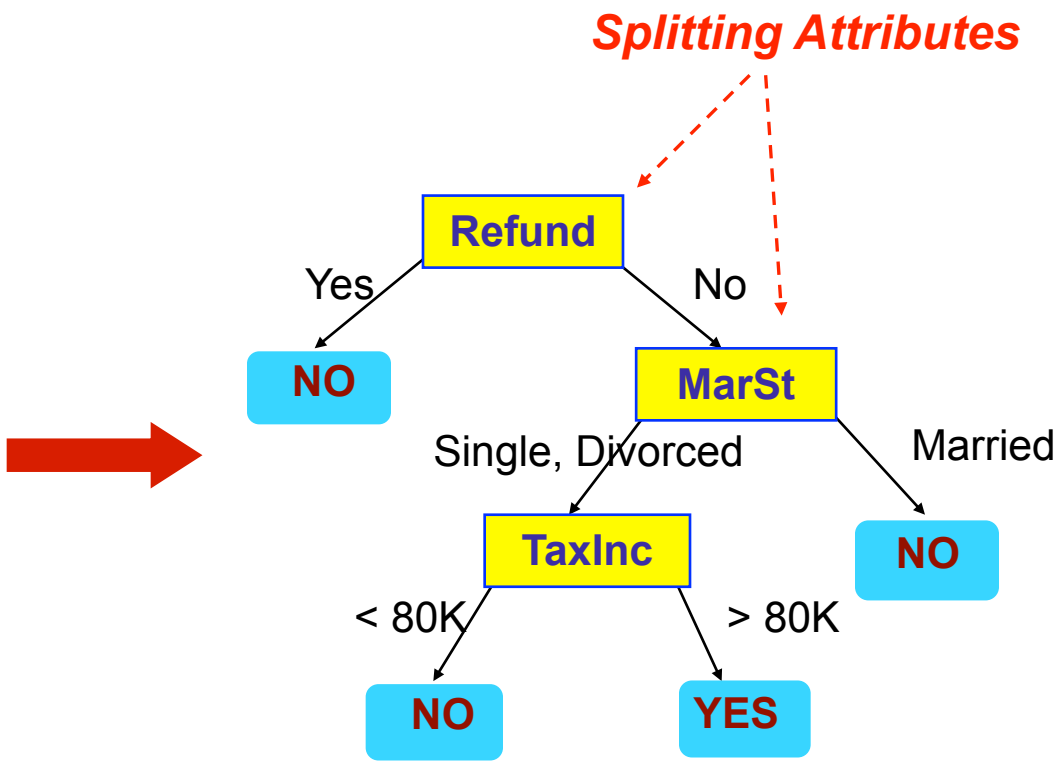
- Attributes might be discrete or continuous, however, the class must be discrete.
- If the class is continuous then regression.
- Both descriptive and predictive.
- Most suited for binaries or nominal attributes, less effective for ordinal because ignores orders.

Classification Techniques

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

Example of a Decision Tree

Tid	Refund (categorical)	Marital Status (categorical)	Taxable Income (continuous)	Cheat (Class)
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

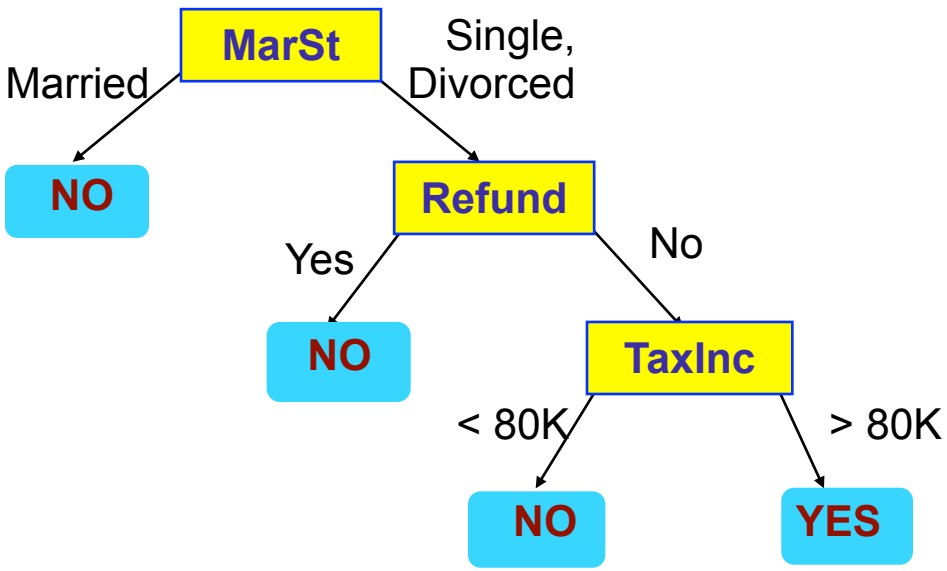


Training Data

Model: Decision Tree

Another Example of 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 could be more than one tree that fits the same data!

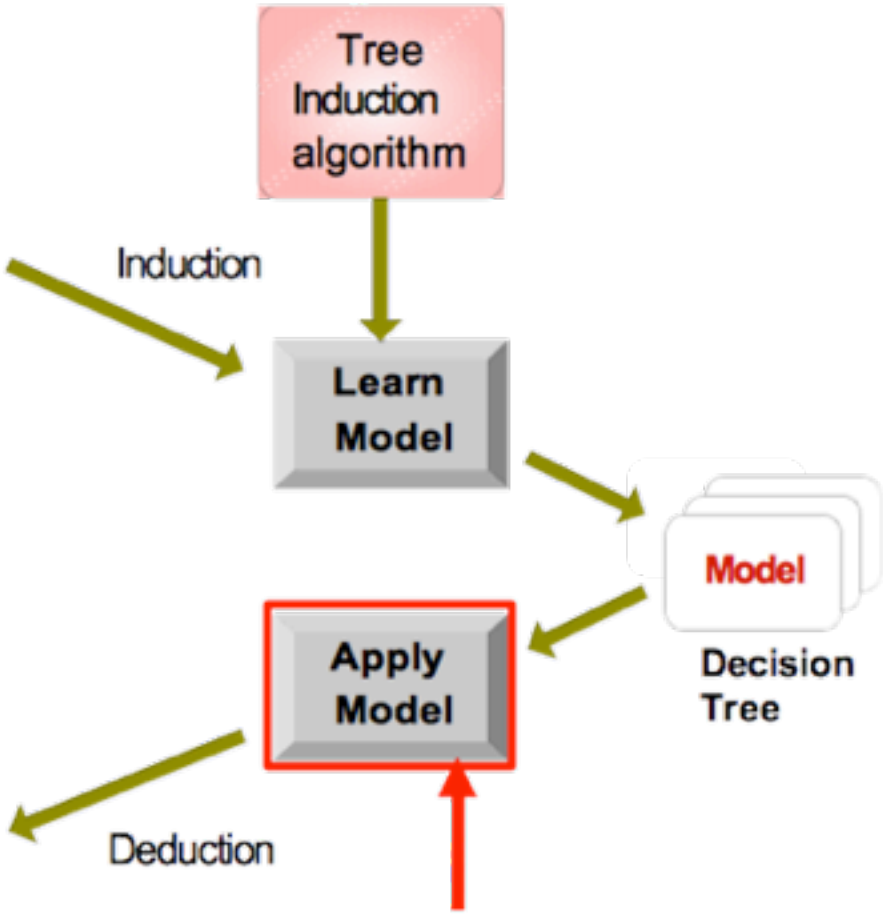
Decision Tree Classification Task

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

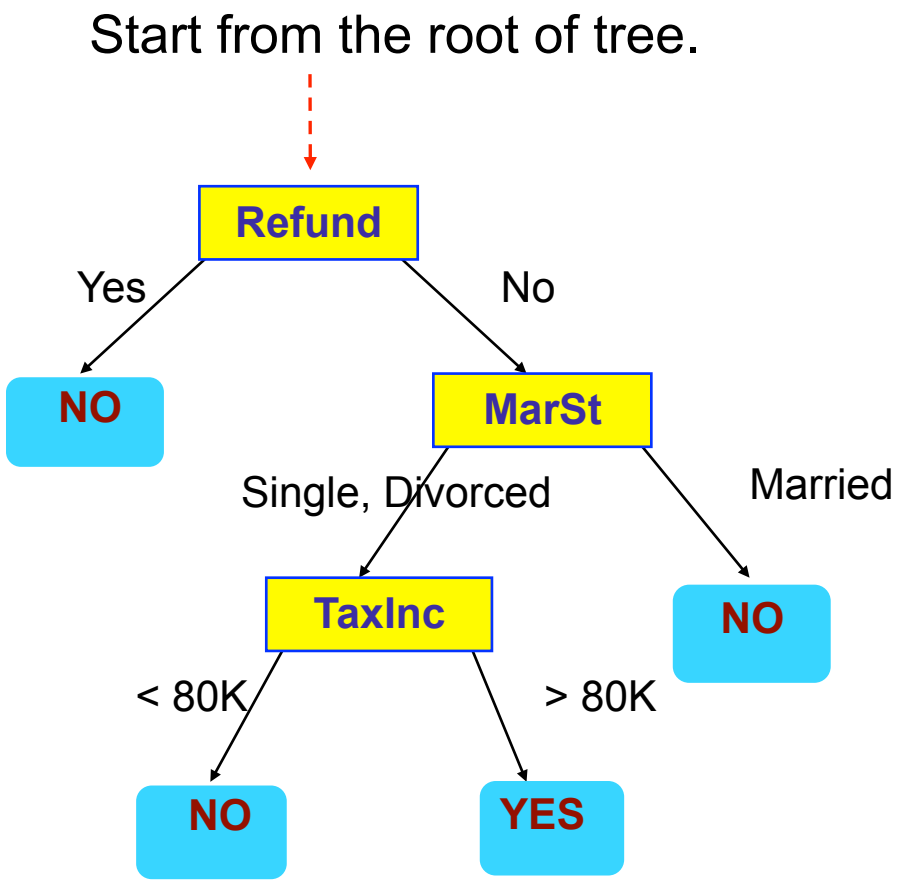
Training Set

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



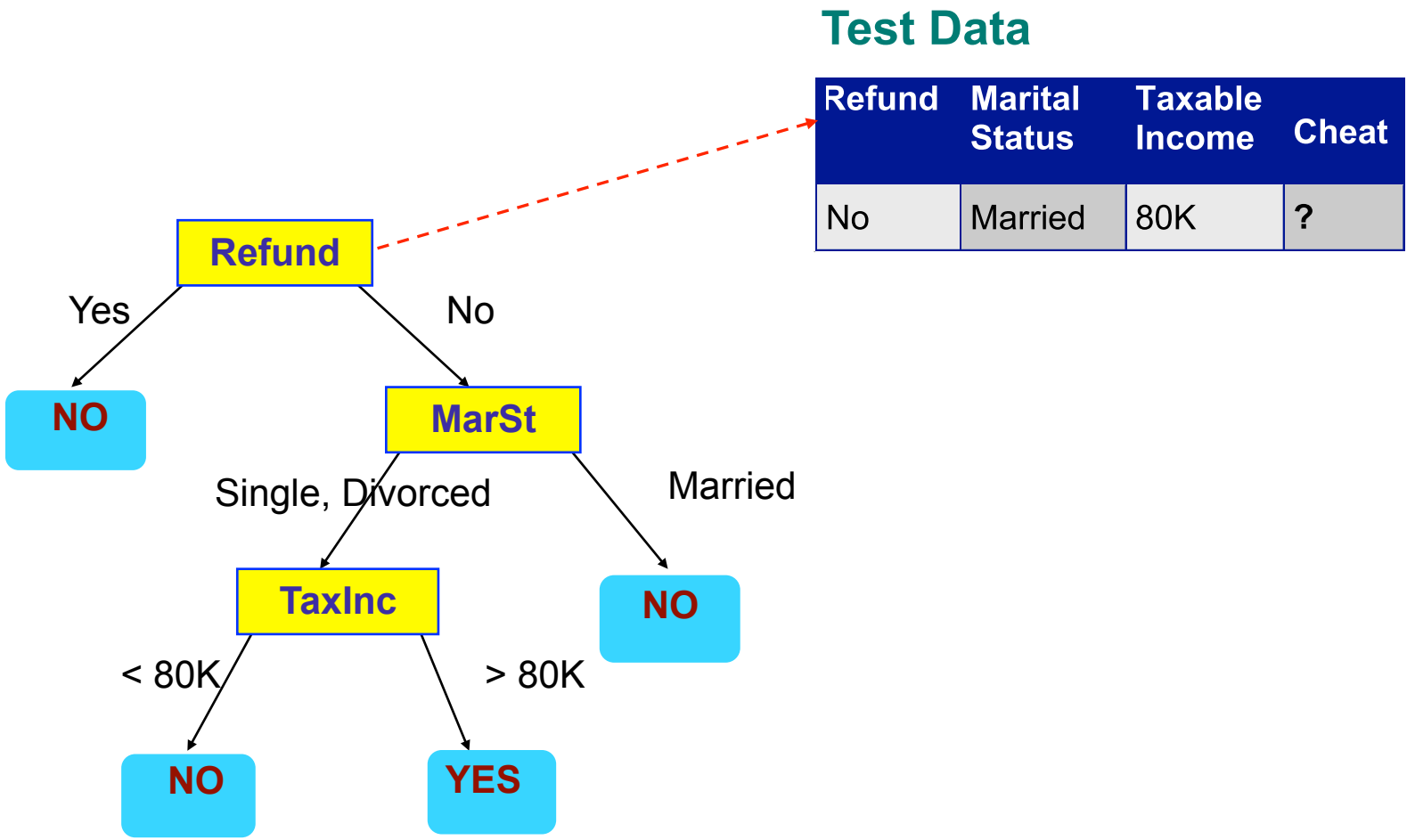
Apply Model to Test Data



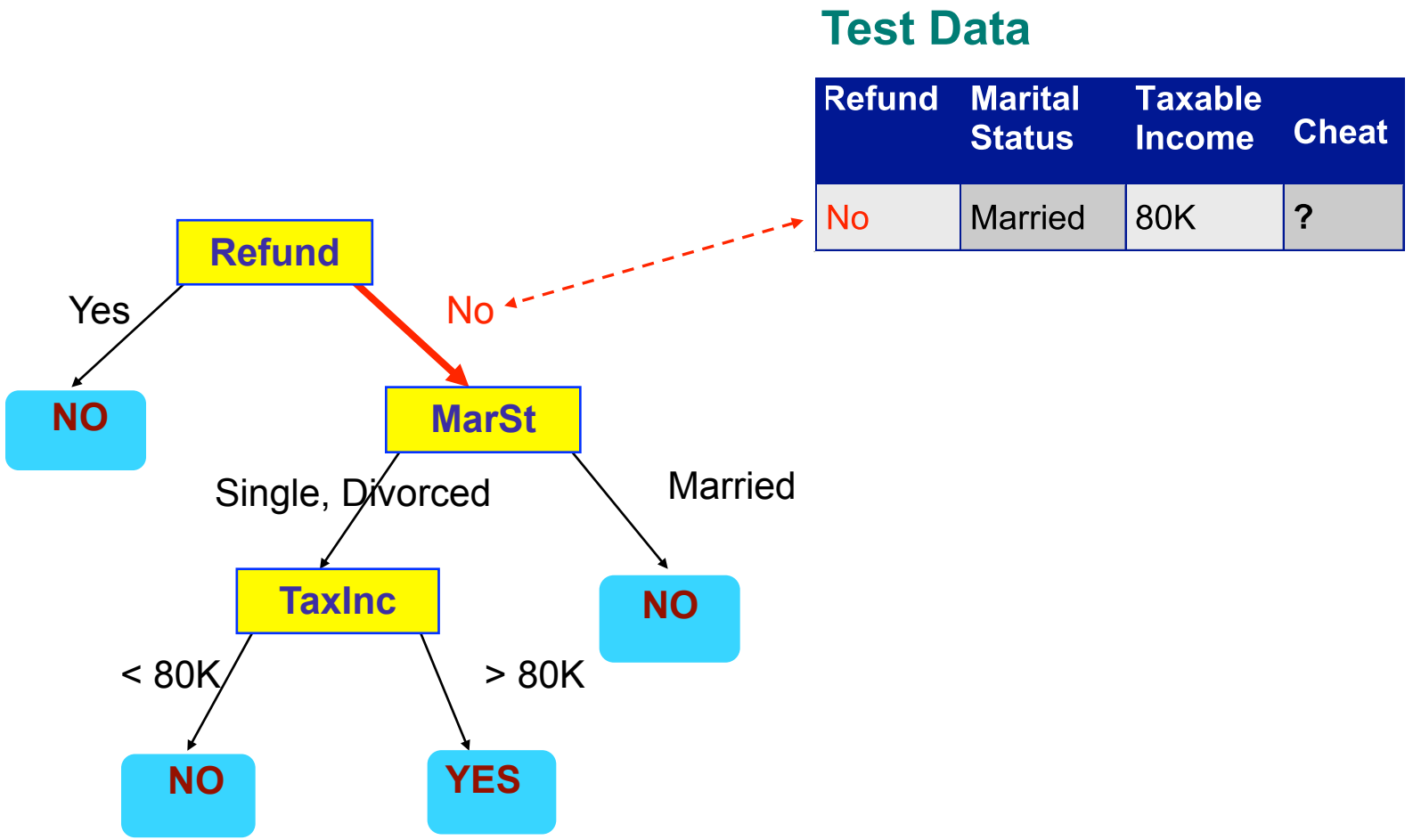
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

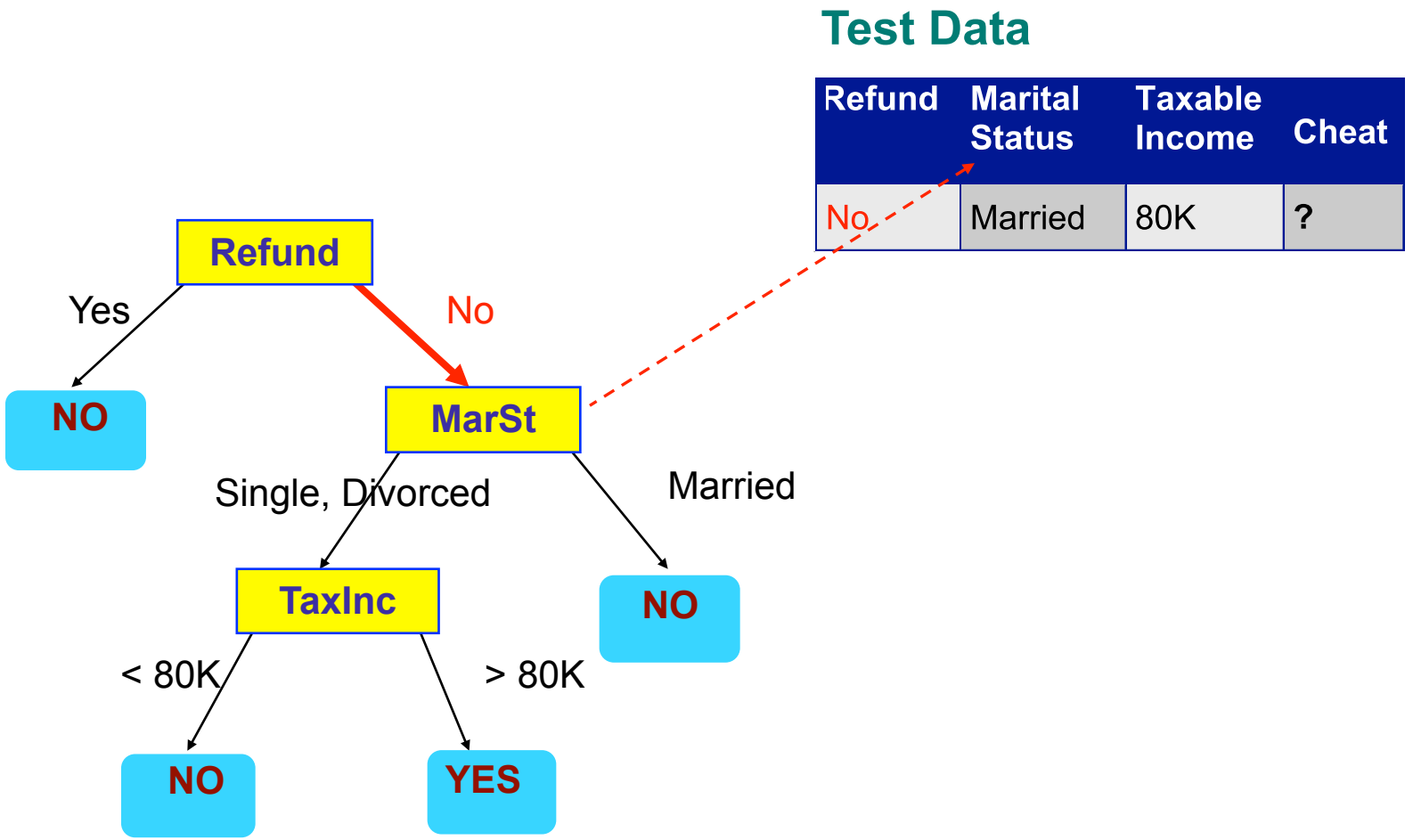
Apply Model to Test Data



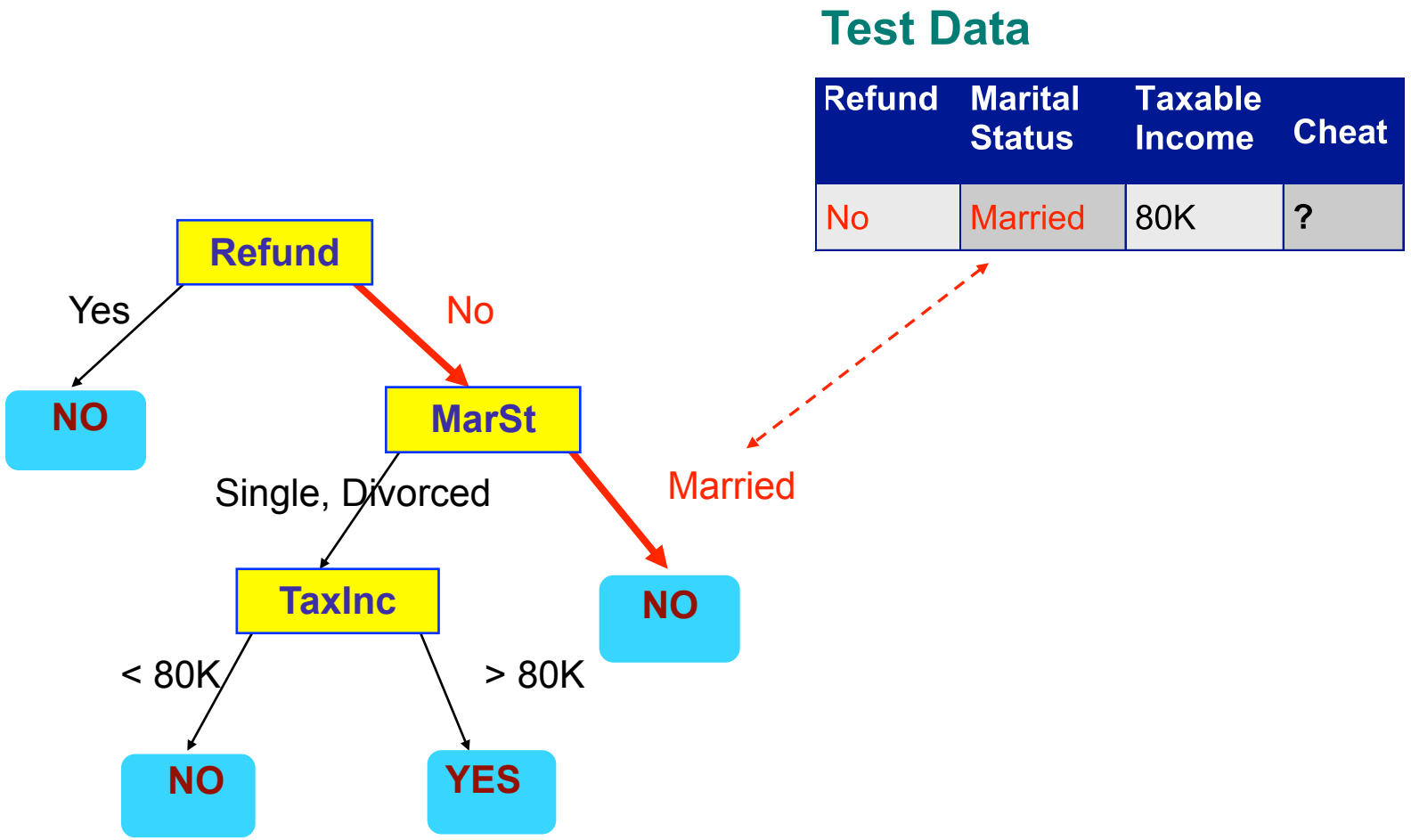
Apply Model to Test Data



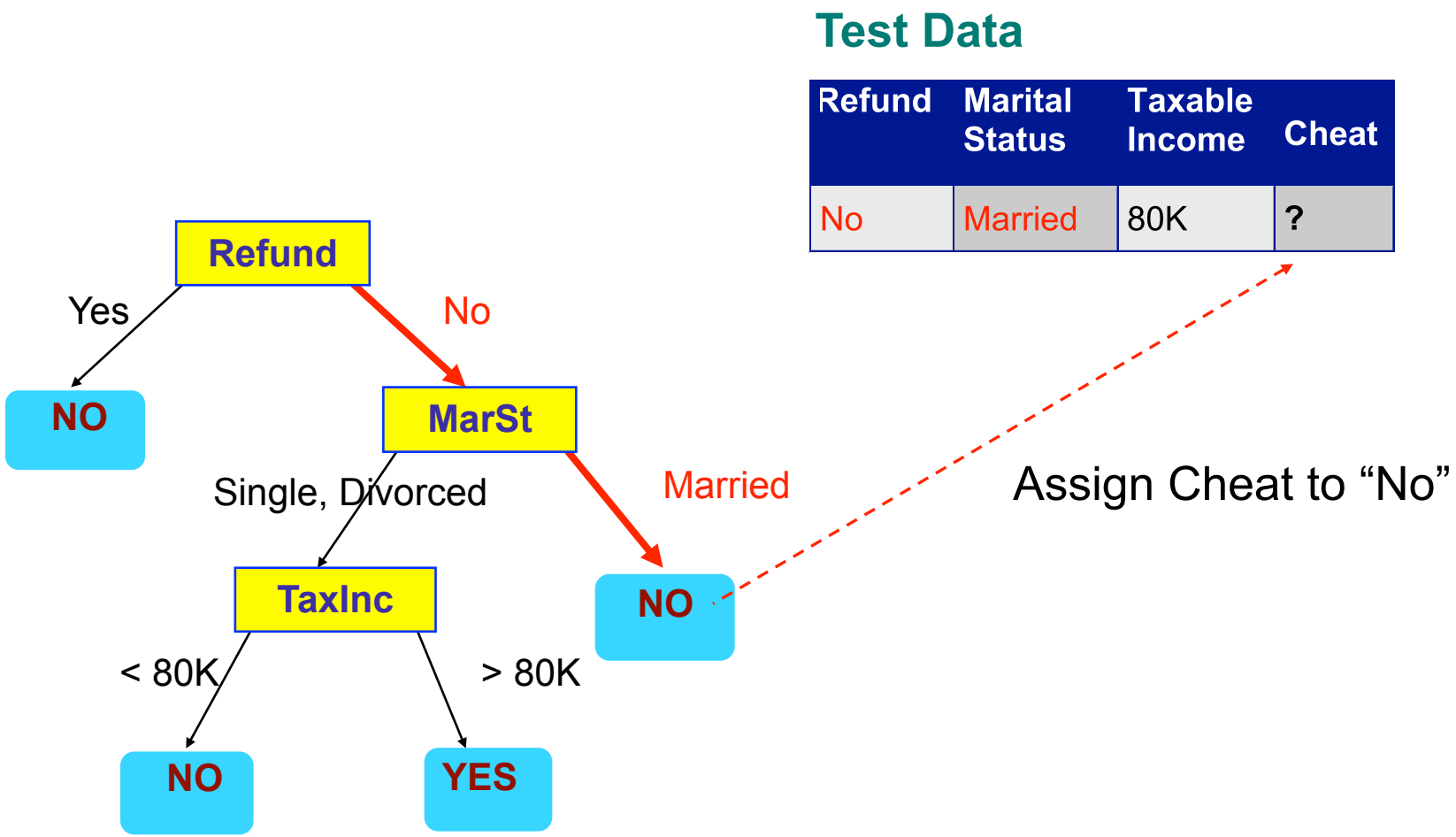
Apply Model to Test Data



Apply Model to Test Data



Apply Model to Test Data



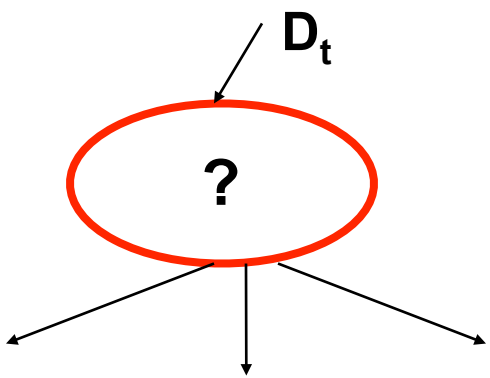
Decision Tree Induction

- 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, y_d
  - If D_t contains records that belong to more than one Class, use an attribute test to split the data into smaller subsets. Recursively apply the procedure to each subset.

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

Default: Don't Cheat

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

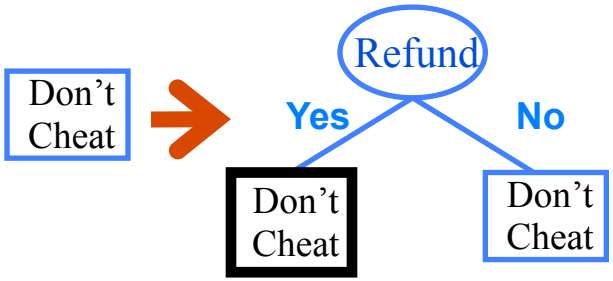
Hunt's Algorithm

Don't Cheat

Default: Don't Cheat

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

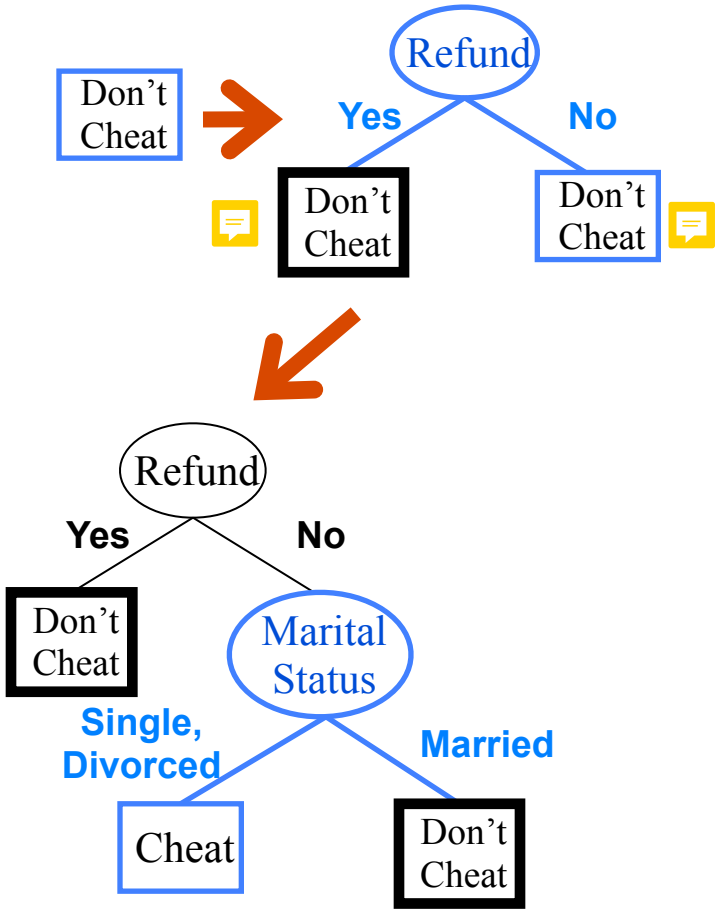
Hunt's Algorithm



Default: Don't Cheat

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

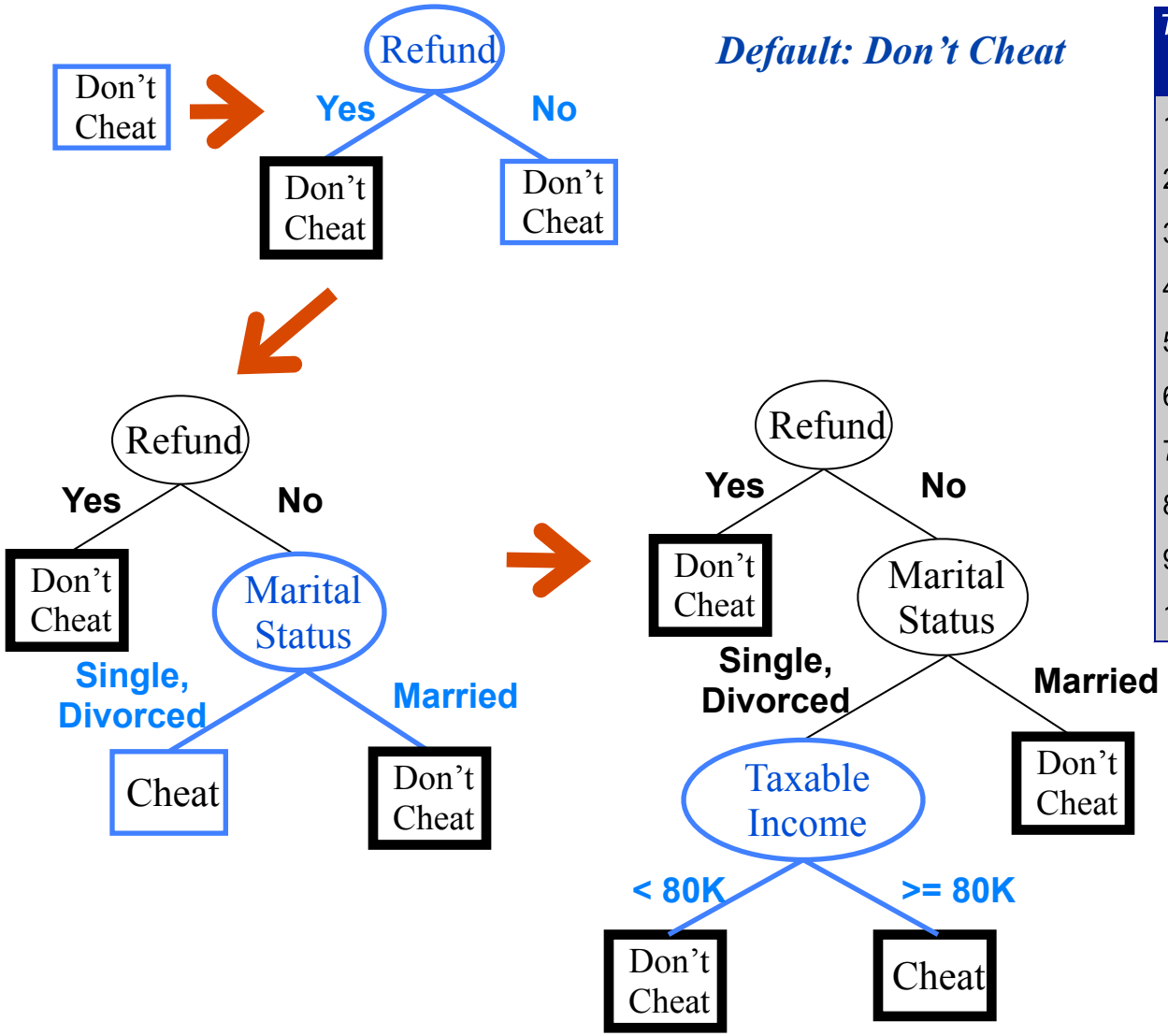
Hunt's Algorithm



Default: Don't Cheat

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



Default: Don't Cheat

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

General structure of tree induction

- Based on a greedy strategy:
  - Split the records based on an attribute test that optimizes certain criterion.

- We need to determine:
  - How to split the records:
    - How to specify the attribute test condition?
    - How to determine the best split?
  - When to stop splitting

General structure of tree induction

- Based on a greedy strategy:
  - Split the records based on an attribute test that optimizes certain criterion.

- We need to determine:
  - How to split the records:
    - How to specify the attribute test condition?
    - How to determine the best split?
  - When to stop splitting

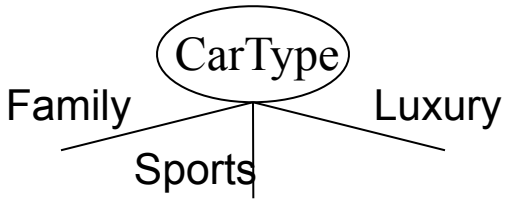
How to Specify Test Condition?

- Depends on attribute types:
  - Nominal (or categorical), have 2 or more categories, no order.E.g.:country (USA, Spain)
  - Ordinal, 2 or more categories but can ordered or ranked. E.g. T-Shirt size: S,M,L,XL,XXL
  - Continuous Ordinal, e.g. temperature, salary

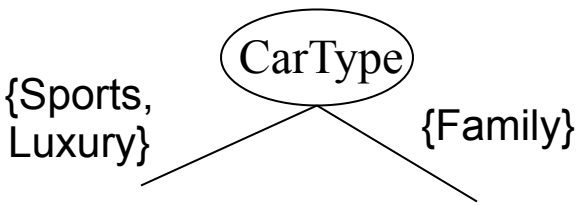
- Depends on number of ways to split
  - 2-way split
  - Multi-way split

Splitting Based on Nominal Attributes

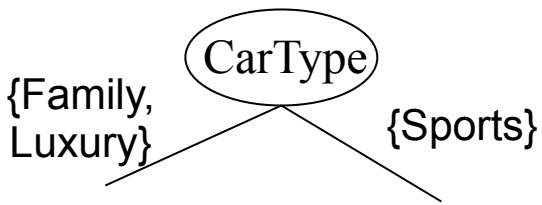
- Multi-way split: Use as many partitions as distinct values.



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

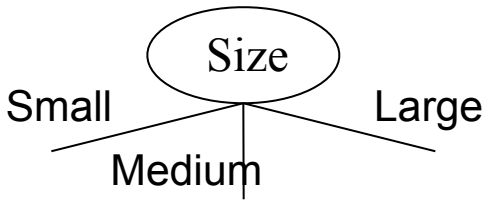


OR

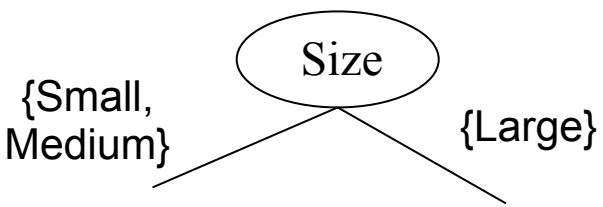


Splitting Based on Ordinal Attributes

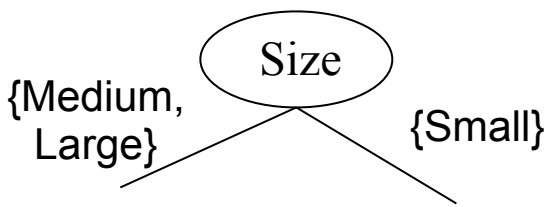
- Multi-way split: Use as many partitions as distinct values.



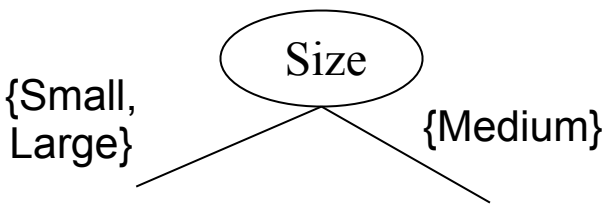
- Binary split: Divides values into two subsets. Need to find optimal partitioning.



OR



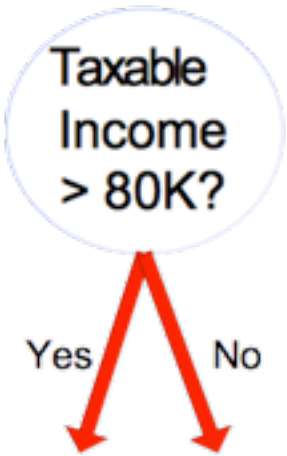
- What about this split?
- (wrong, it violates order)



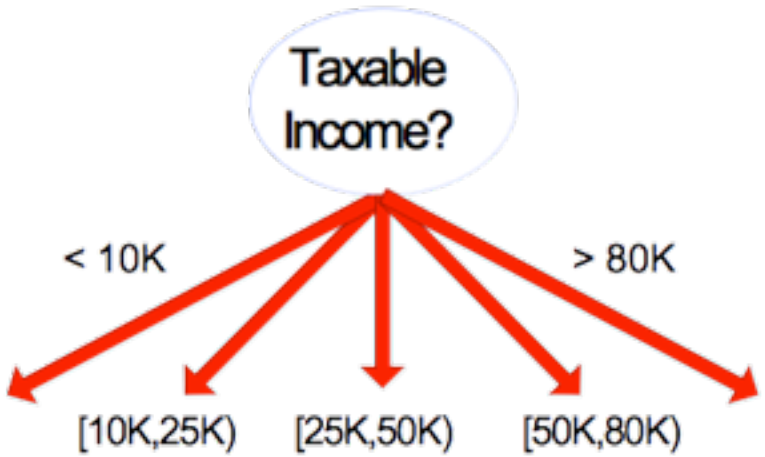
Splitting Based on Continuous Attributes

- Different ways of handling
  - Discretization
    - Static – discretize once at the beginning
    - Dynamic – ranges can be found by equal interval bucketing, equal frequency bucketing (percentiles), or clustering.
  - Binary Decision: $(A < v)$ or $(A \geq v)$
    - brute force approach for finding the best split (based on values in the data)
    - computationally more expensive

Splitting Based on Continuous Attributes



(i) Binary split



(ii) Multi-way split

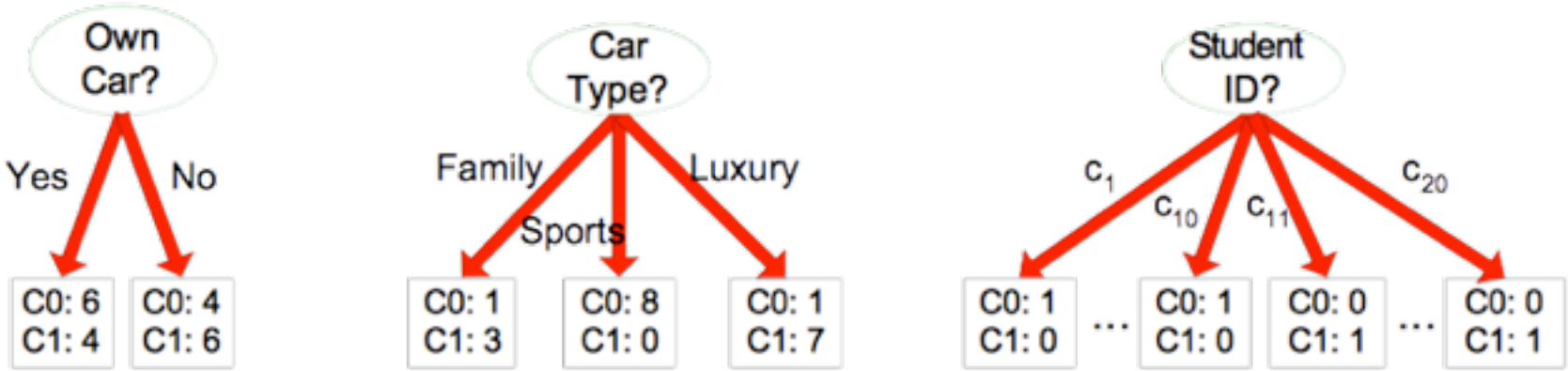
General structure of tree induction

- Based on a greedy strategy:
  - Split the records based on an attribute test that optimizes certain criterion.

- We need to determine:
  - How to split the records:
    - How to specify the attribute test condition?
    - How to determine the best split?
  - 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

Non-homogeneous,

High degree of impurity

C0:9
C1:1

Homogeneous,

Low degree of impurity

Measures of Node Impurity

- Gini Index
- Entropy
- MisClassification error

Measure of Impurity: GINI

- Gini Index for a given node t :

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

(NOTE: $p(j \mid t)$ is the probability of Class j at node t).

  - Maximum (1 - $1/n_c$) when records are equally distributed among all Classes, implying least interesting information
  - Minimum (0.0) when all records belong to one Class, implying most interesting information

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

- Used in CART, SLIQ, SPRINT.
- When a node p is split into k partitions (children), the quality of split is computed as the weighted average:

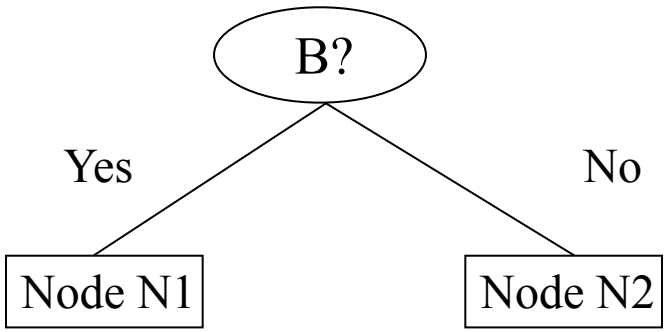
$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)$$

where, n_i = number of records at child i,

n = number of records at node p.

Binary Attributes: Computing GINI

- Splits into two partitions
- Effect of Weighing partitions:
  - Larger and Purer Partitions are sought for.



	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		

Weighted Average
= 7/12 * 0.408 +
5/12 * 0.32
= 0.371

Categorical Attributes: Find best GI

- 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	

Continuous Attributes: Find Best GI

- Input: N records, attribute A
- Output: value v with min Gini for A

- Naive:
  - let v1...vN be the values of A in the N records
  - compute Gini for each of them, return the min
  - requires O(N^2) operations, too expensive!

Continuous Attributes: Find Best GI

- Better strategy
  - Sort the records non-decreasingly: v1<=v2,...,<=vN
  - compute Gini index for each of them *incrementally*
  - return the candidate with minimum Gini
  - Running time: O(N log N)

Continuous Attributes: Find Best GI

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

Incremental computation of Gini

Candidates: 60,70,75.... Attribute: (TI)

- Gini(TI <= 60)=?
1 'No' record with TI <= 60 and
9 records with TI > 60, 3 'Yes', 6 'No' =>
GI = 1/10 x 0 + 9/10 x (1-(3/9)^2-(6/9)^2) = 0.4

- Gini(TI <= 70)=?
Update distribution: 1 'No 'record in (60,70] =>
1+1=2 'No' with TI <= 70, 3 'Yes' and 6-1=5
'NO' with TI > 70 => GI= 0.375

Alternative Splitting Criteria based on INFO

- Entropy at a given node t:

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

(NOTE: $p(j \mid t)$ is the probability of Class j at node t).

  - Measures homogeneity of a node.
    - Maximum (log n_c) when records are equally distributed among all Classes implying least information
    - Minimum (0.0) when all records belong to one Class, implying most information

Examples for computing Entropy

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

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

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

Entropy = – 1 log 1 = – 0 – 0 = 0

***note we define 0*log 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 Theory

- Information Gain:

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

Parent p is split into k partitions one for each value v_i for *att*

n_i is number of records in partition i (with value v_i)

  - Measures Reduction in Entropy achieved because of the split. Choose the split that achieves most reduction (maximizes GAIN)
  - Used in ID3 and C4.5
  - Disadvantage: Tends to prefer splits that result in large number of partitions, each being small but pure.

Splitting Based on INFO...

- Gain Ratio:

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

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

Parent Node, p is split into k partitions, SplitINFO > 0

n_i is the number of records in partition i with value v_i for the att.

- Adjusts Information Gain by the entropy of the partitioning (SplitINFO). Higher entropy partitioning (large number of small partitions) is penalized!
- Overcomes the disadvantage of Information Gain

Splitting Criteria based on Classification Error

- Classification error at a node t :

$$Error(t) = 1 - \max_i P(i \mid t)$$

- Measures misClassification error in a node.
  - Maximum (1 - $1/n_c$) when records are equally distributed among all Classes, implying least interesting information
  - Minimum (0.0) when all records belong to one Class, implying most interesting information

Examples for Computing Error

$$Error(t) = 1 - \max_i P(i \mid t)$$

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

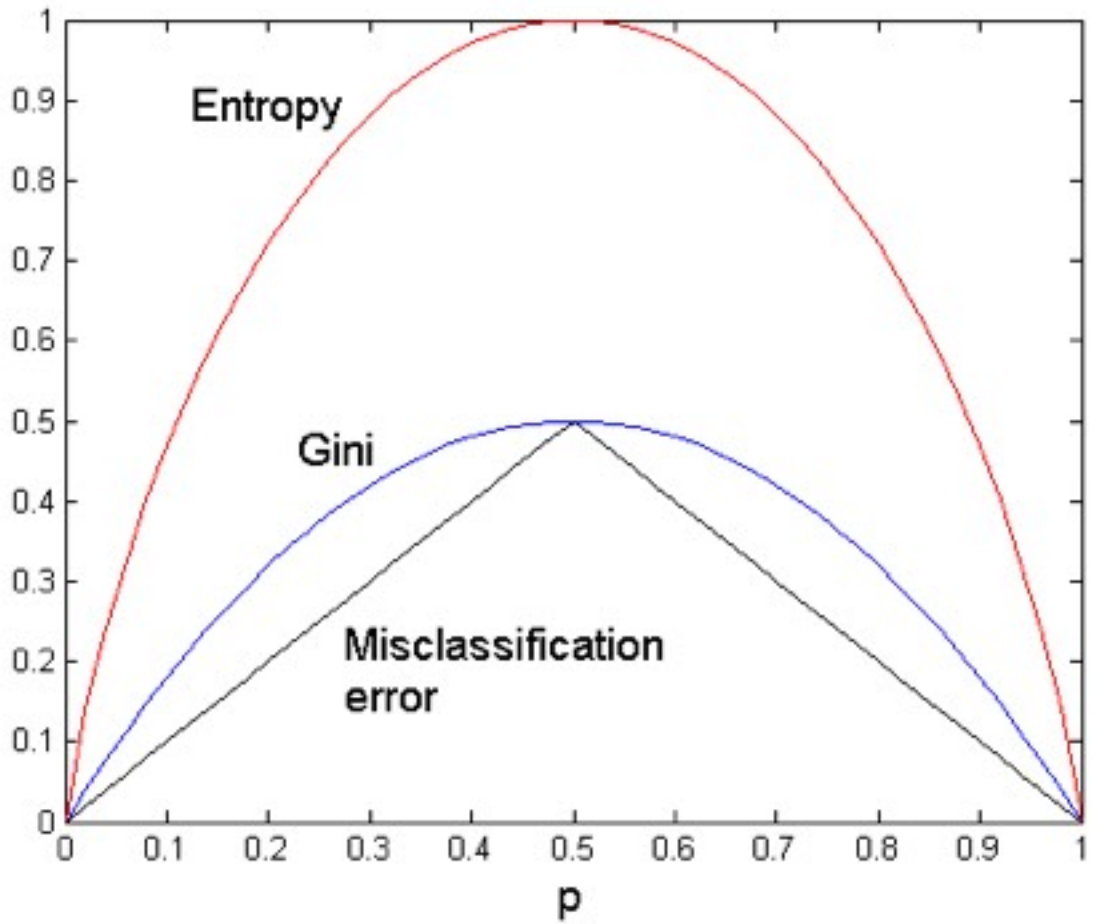
C1	**2**
C2	**4**

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

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

Comparison among Splitting Criteria

Binary Classification (2 Classes)



Comparison among Splitting Criteria

- More about splitting criteria:
  - Studies have shown that the choice of impurity measure has little effect on the final result
  - The strategy used to prune the tree (decrease its size and complexity) has a much bigger impact. More on this later...

General structure of tree induction

- Based on a greedy strategy:
  - Split the records based on an attribute test that optimizes certain criterion.

- We need to determine:
  - How to split the records:
    - How to specify the attribute test condition?
    - How to determine the best split?
  - When to stop splitting

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 (e.g. similar salaries)
- Early termination (to be discussed later)

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 tasks

Decision Trees

C4.5:

- #1 in the top 10 DM algorithm according to Springer LNCS (2008) see http://www.cs.umd.edu/~samir/498/10Algorithms-08.pdf
- Simple depth-first construction.
- Uses Information Gain
- Needs entire data to fit in memory.
- Unsuitable for Large Datasets.

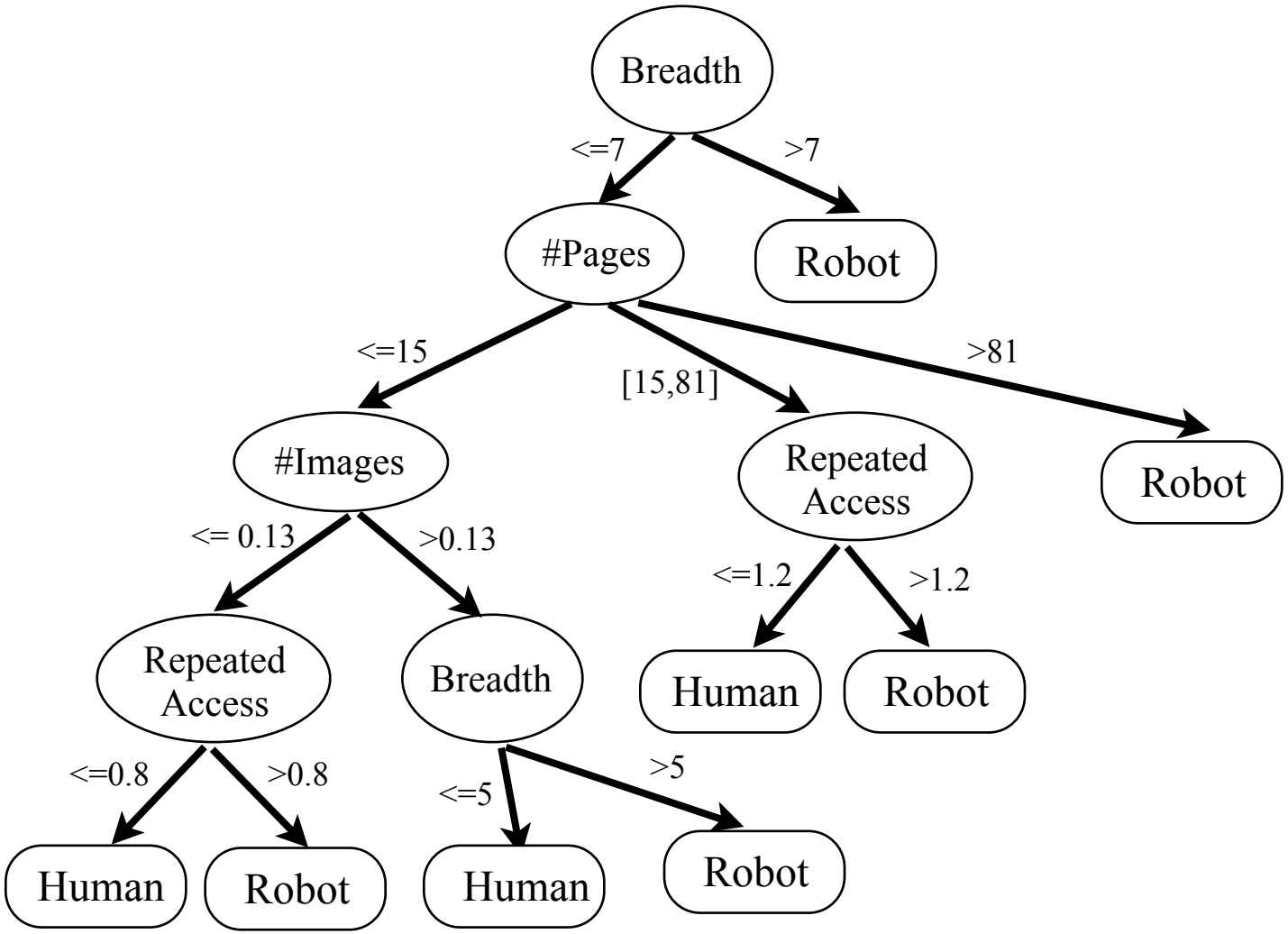
Most successful techniques by winners in Kaggle are gradient boosting (xgboost) and random forests, which build on decision trees.

Example: Web-Robot detection

- **Problem**: Given a Web log, detect which accesses are made by humans and which ones by robots (e.g. crawlers)

- **Input:** a list of accesses to a Web site with the info:
  - IP address
  - average breadth of page visits
  - average depth of page visits
  - repeated access: how many times a same web page is accessed on average
  - number of images retrieved

Decision Tree for Web Robot Detection



Results

- Web robots tend to access as many pages as possible => visits are broad but shallow.
- Web robots rarely download pictures.
- Web robots are more likely to make repeated requests to a same Web page (Web pages retrieved by humans are often cached).

Practical Issues of Classification

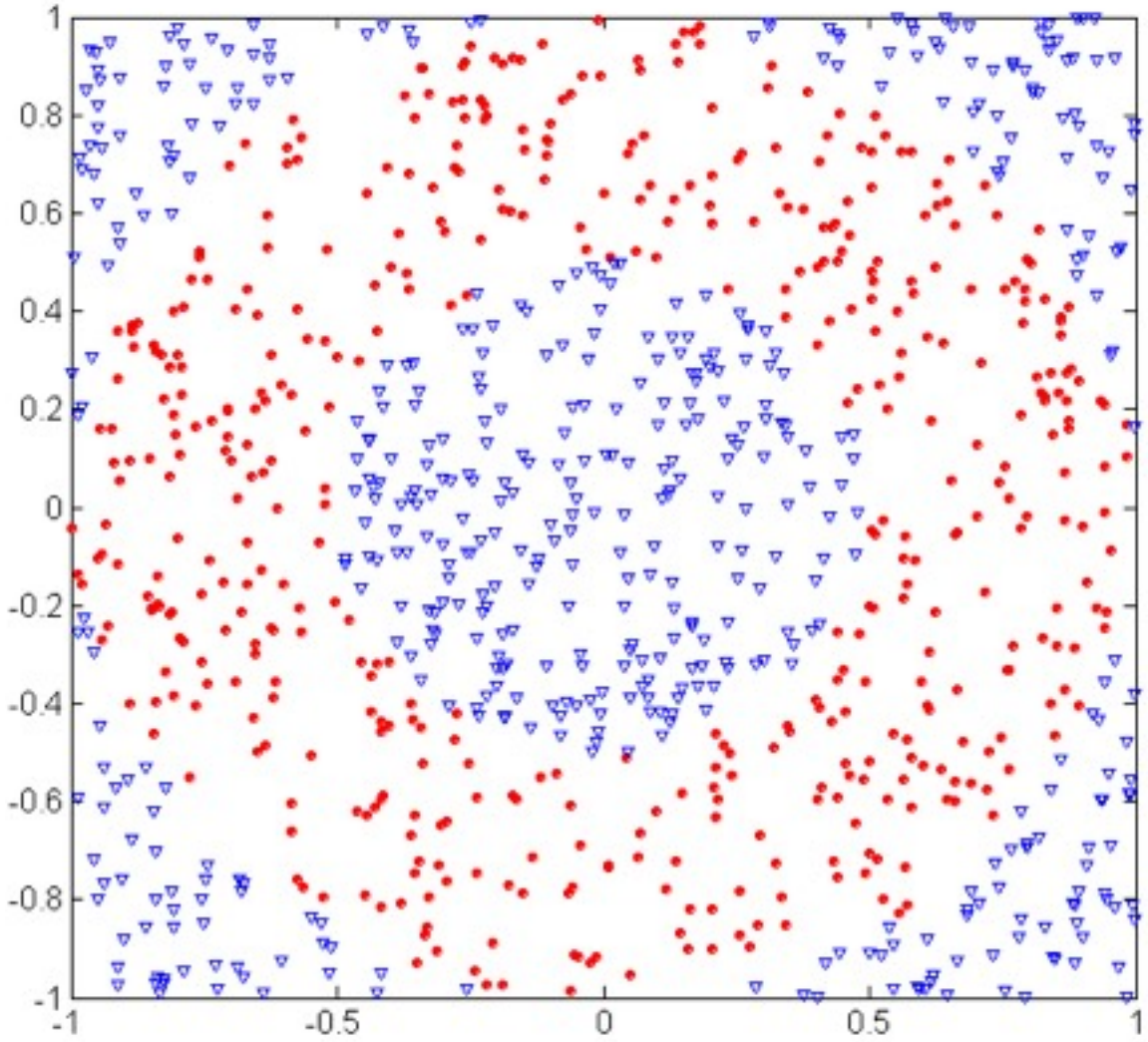
- Underfitting and Overfitting
- Missing Values
- Costs of Classification

Underfitting and Overfitting

- Our main goal is to find a model that predicts the Class values on unseen records. We might have:
  - underfitting: the model did not learn the structure of the data and performs poorly on both the training set and the test set.
  - overfitting: the model becomes too specialized, describes well the training data but not unseen records. Good in training set but poor in the test set.

- Might have overfitting when decision tree is too large e.g. one record per leaf, which means tree size > dataset size.

Underfitting and Overfitting (Example)



500 circular and 500 triangular data points.

Circular points:

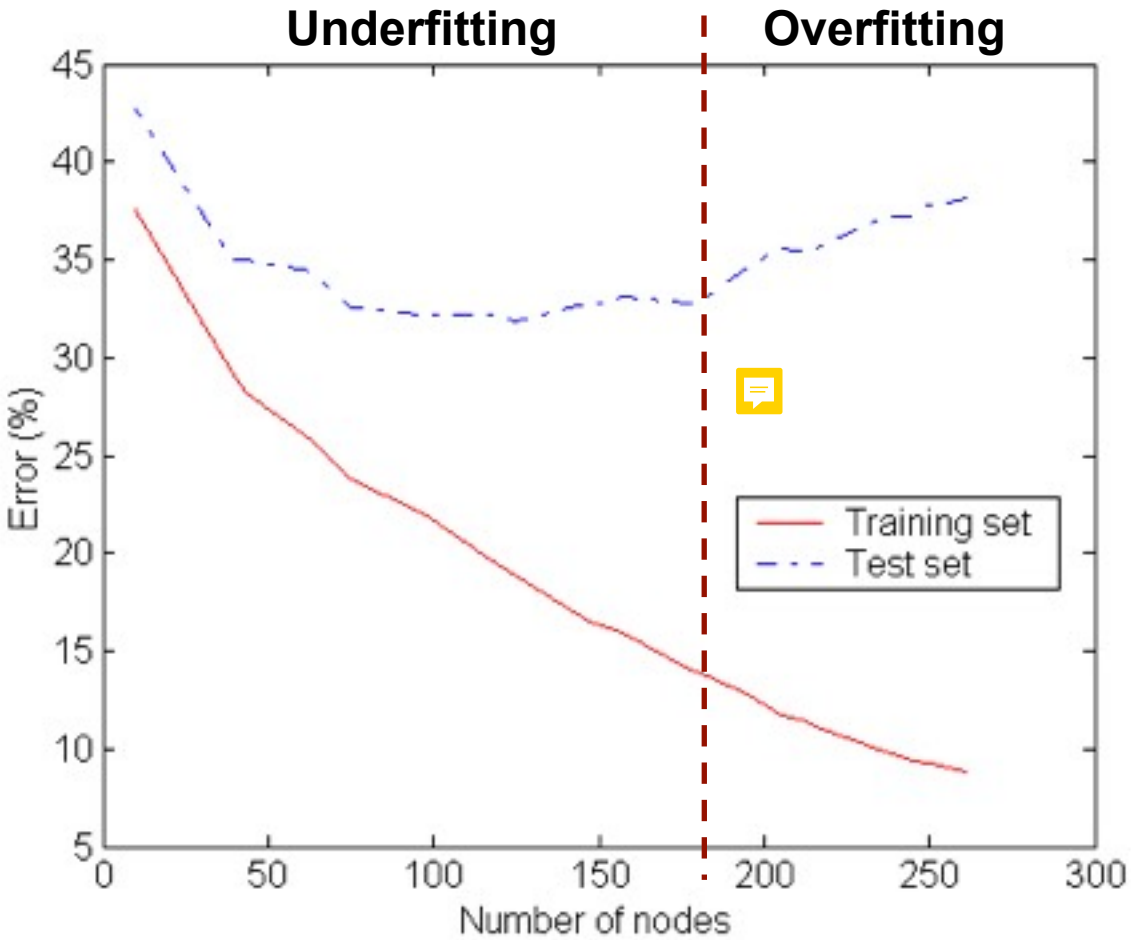
$0.5 \leq \text{sqrt}(x_1^2+x_2^2) \leq 1$

Triangular points:

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

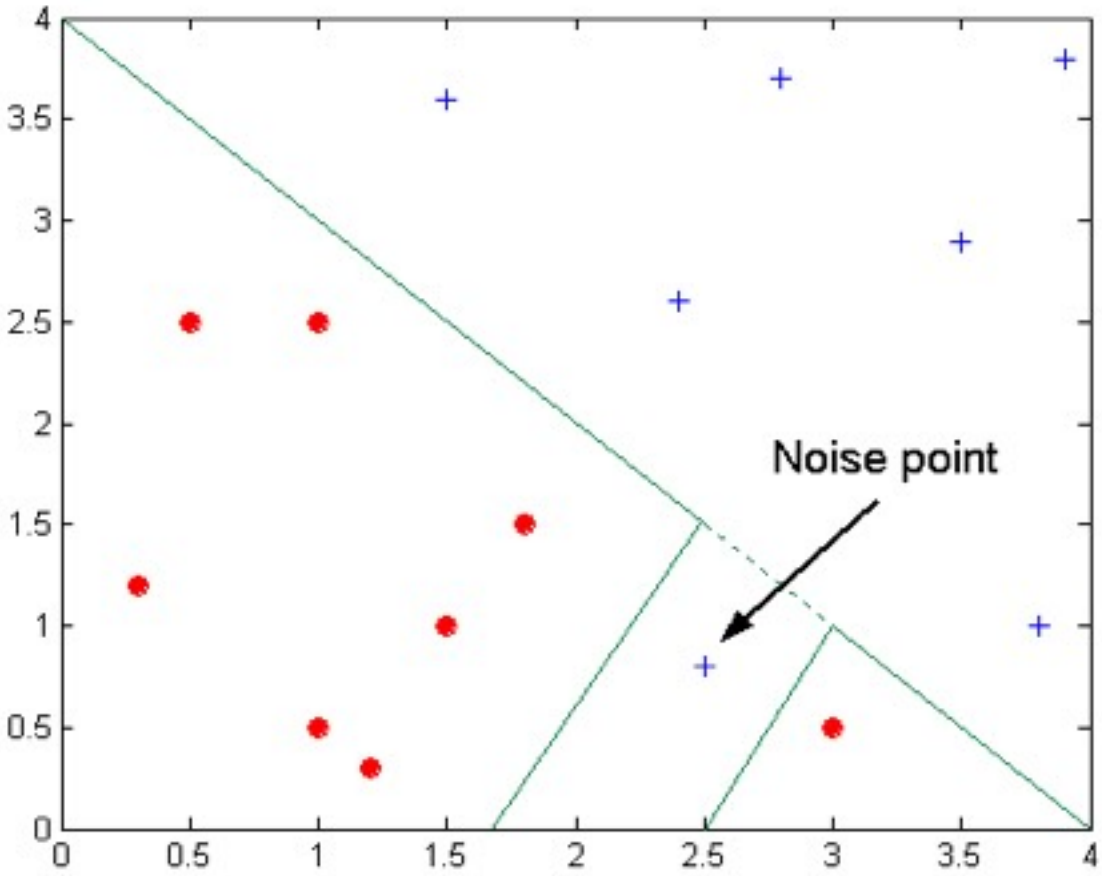
$\text{sqrt}(x_1^2+x_2^2) < 1$

Underfitting and Overfitting: Results



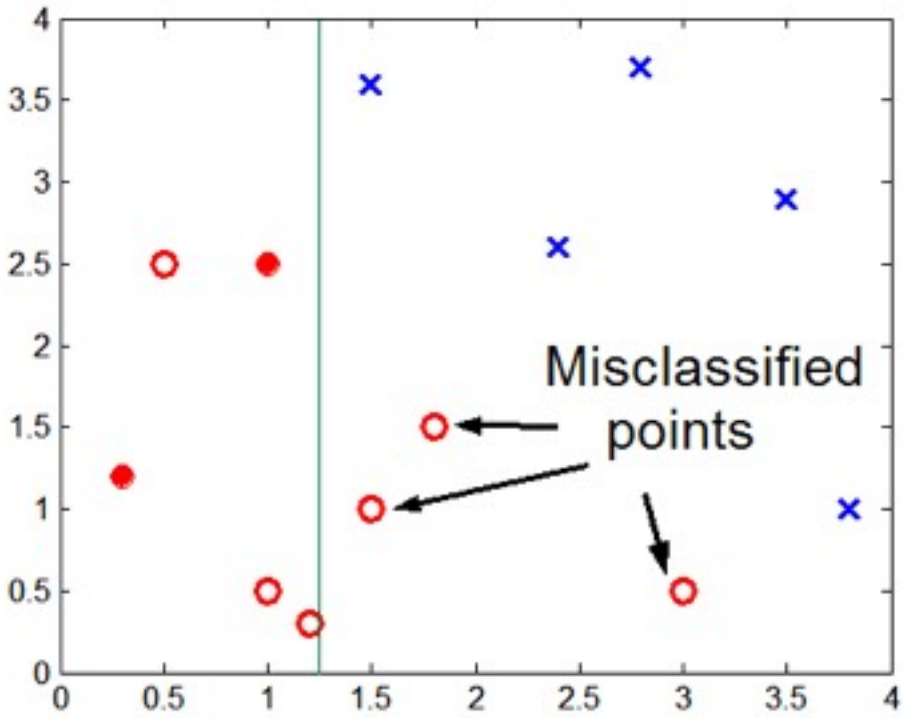
Results after building a decision tree on the previous dataset. Note that more sophisticated decision trees (larger # nodes) deliver good results in the training set but poor in the test set.

Overfitting due to Noise



Decision boundary is distorted by noise point

Overfitting due to Insufficient Examples



Lack of data points in the lower half of the diagram makes it difficult to predict exactly the Class labels of that region

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

Generalization and training errors

- Our main goal: the decision tree should generalize well to previously unseen data

- Two main errors
  - Training (aka resubstitution) errors: on training set
  - Generalization errors: errors on testing set

Occam's Razor (ORP)

- **ORP:** Given two models with 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

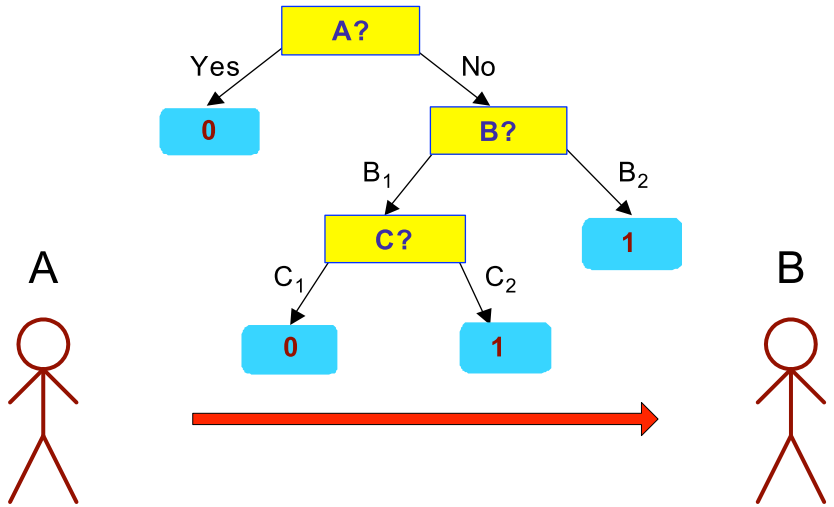
Estimating Generalization Errors

- **Optimistic approach:** ideal training test, generalization errors=training errors.

- Penalty for model complexity (not ideal training set, in line with ORP)
  - **Pessimistic estimate:**
    - Gener. error (Model)= Train. Error(Model, Train. Data) + alpha * Complexity(Model)
    - E.g. Complexity(Model) = num. leaves, alpha=0.5. *Tree with 30 leaves,10 errors on training (1000 records):*
      - *Gener. error = 10 + 30×0.5 = 25*
      - *Normalized gener. error = (10 + 30×0.5)/1000 = 2.5%*
  - **Minimum Description Length** (MDL)

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	?

A wishes to send data to B. B knows all attribute values but not the class values.

- Cost(Model,Data) = Cost(Data|Model) + Cost(Model)
  - Cost(Model) = cost of encoding the model
  - Cost(Data|Model) encodes misClassification errors.
- Prefer the model with smallest cost.

How to Address Overfitting

- Pre-Pruning (Early Stopping Rule)
  - 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 per node is less than some threshold
    - Stop if Class distribution of instances are independent of the available features (e.g., using χ^2 test to test variables indep.)
    - Stop if expanding the current node does not improve impurity measures (e.g., Gini or information gain).

How to Address Overfitting…

- Post-pruning (after building the tree)
  - Grow decision tree to its entirety
  - 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
  - Better than pre-pruning but more operations.

Example of Post-Pruning

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

Training Error (Before splitting) = 10/30

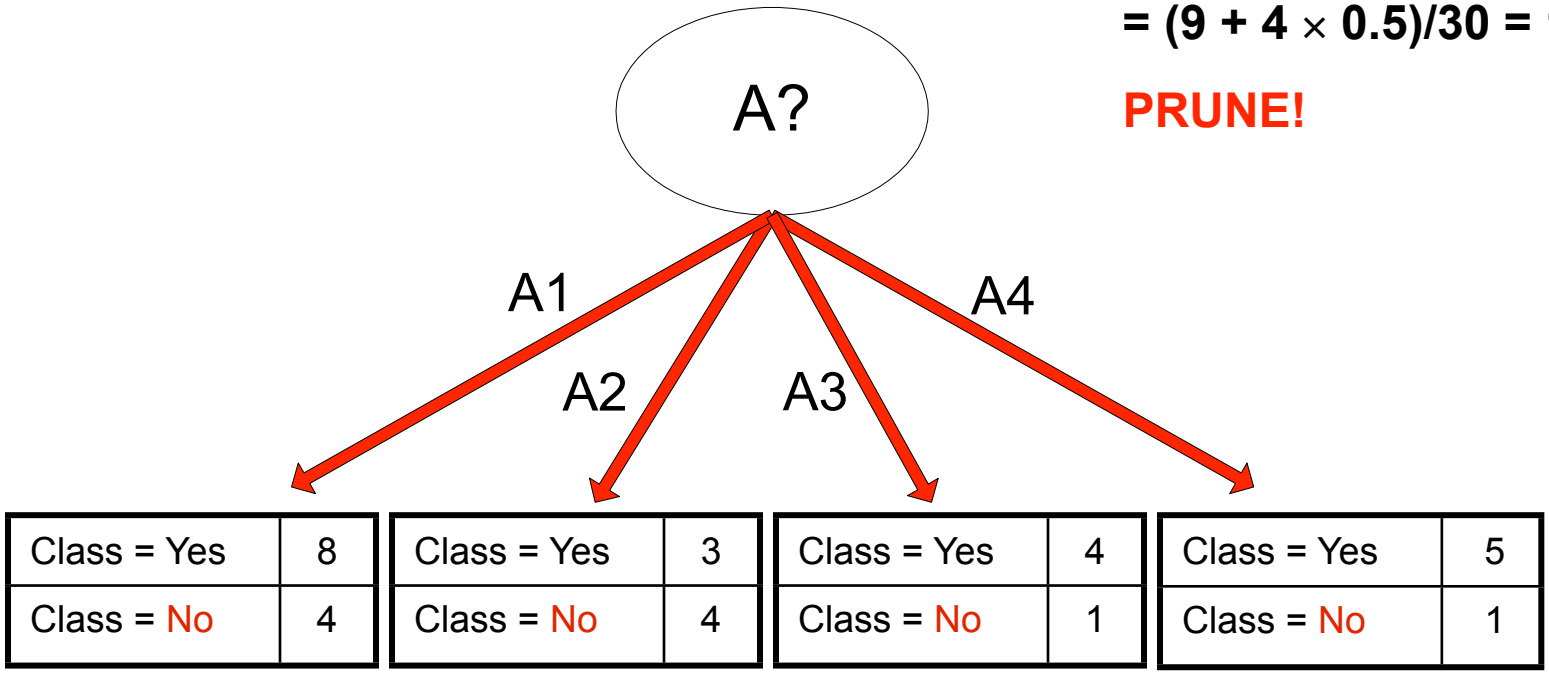
Generalization err =(10 + 0.5)/30 = 10.5/30

Training Error (After splitting) = 4+3+1+1= 9/30

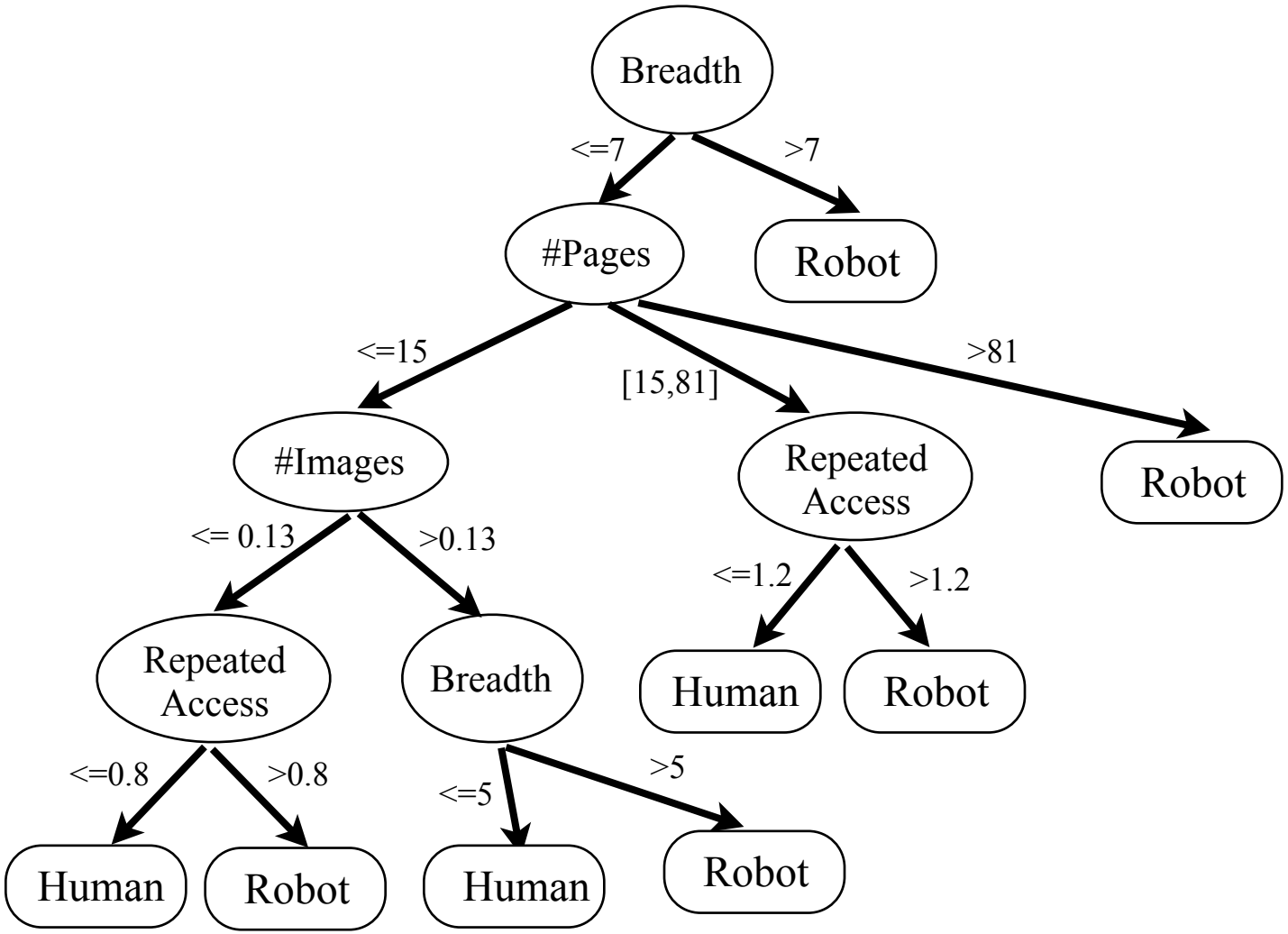
Generalization error (After splitting)

= (9 + 4 × 0.5)/30 = 11/30

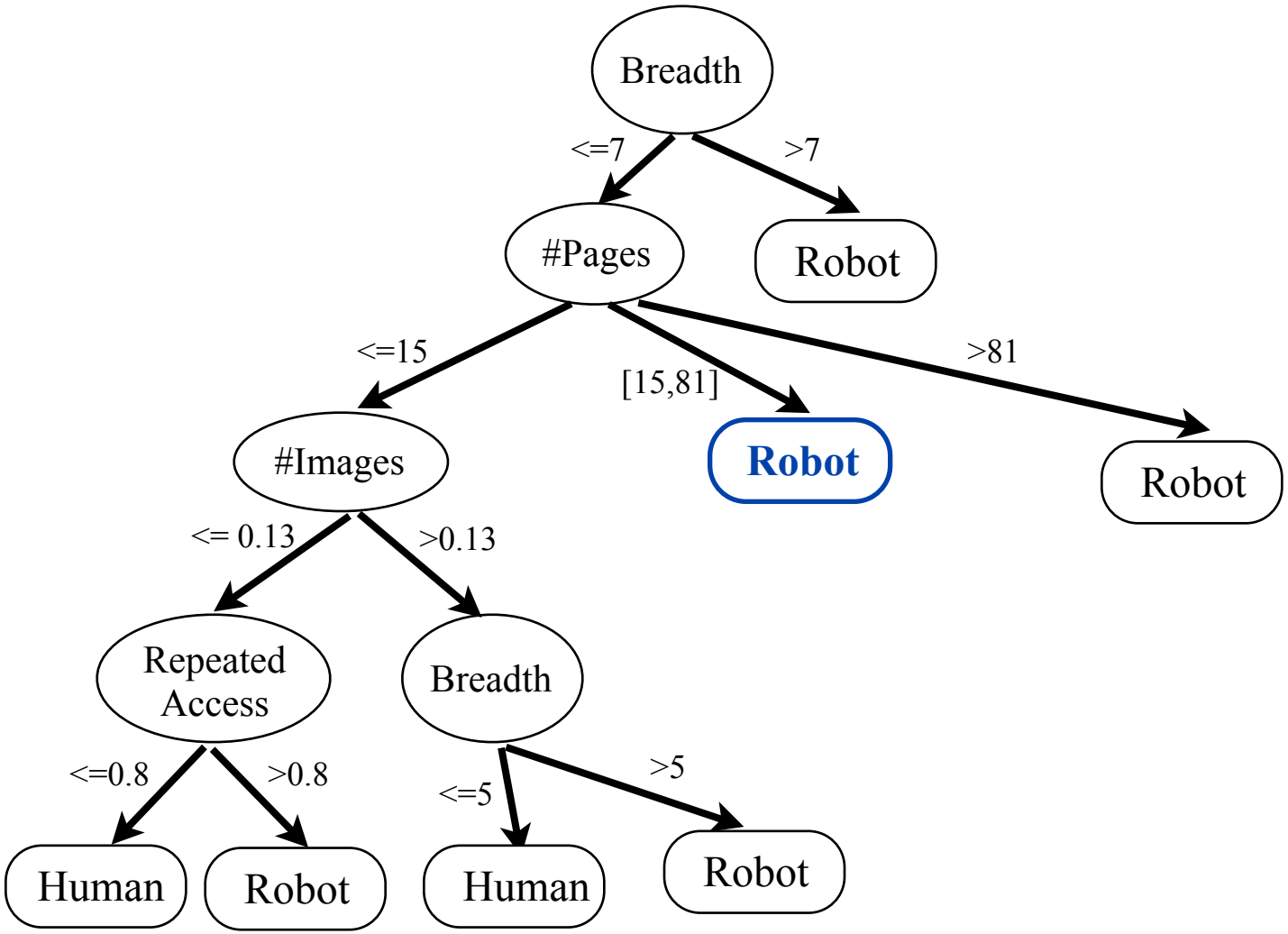
PRUNE!



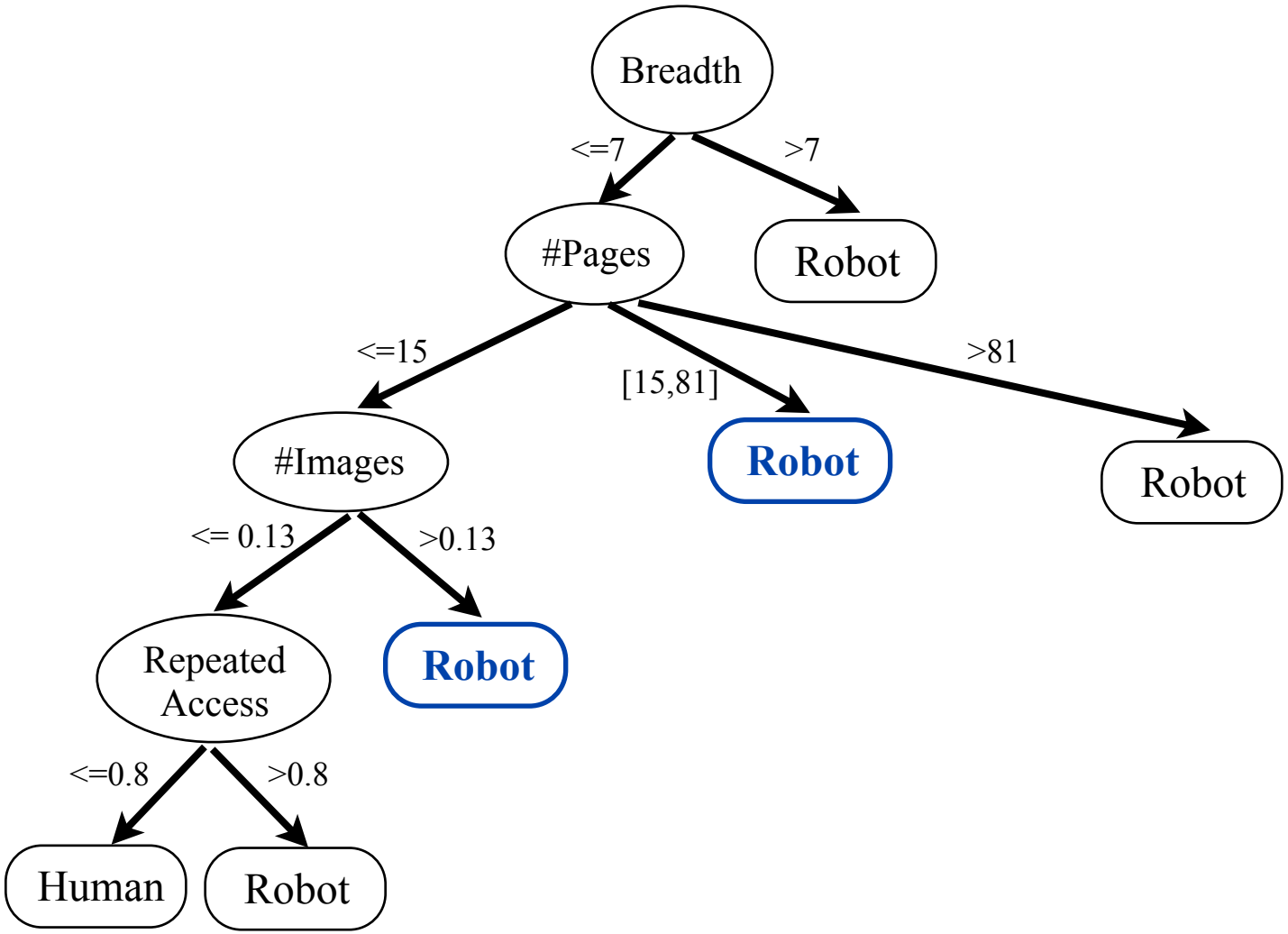
Decision Tree for Web Robot Detection



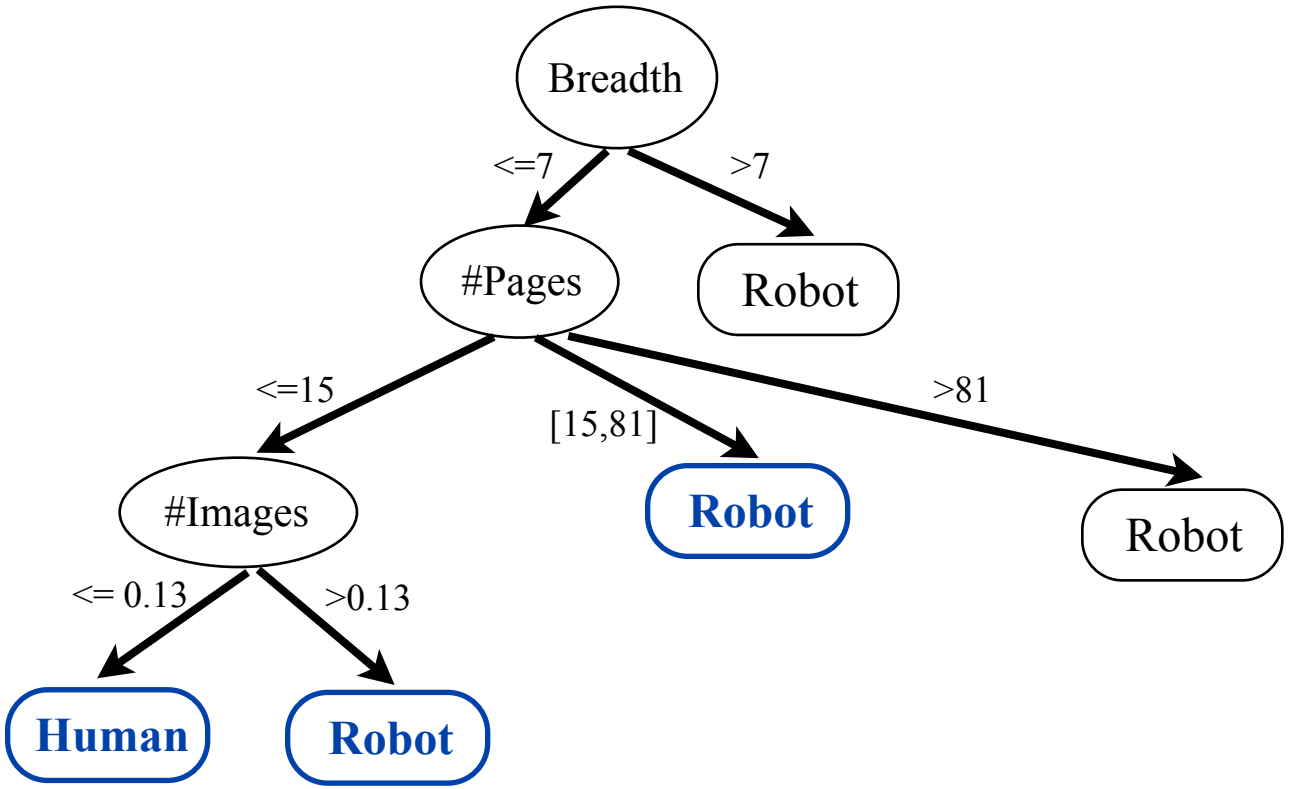
Decision Tree for Web Robot Detection



Decision Tree for Web Robot Detection



Decision Tree for Web Robot Detection



Decision Trees: Issues

- Data Fragmentation
- Search Strategy
- Expressiveness
- Tree Replication

Data Fragmentation

- Number of instances gets smaller as you traverse down the tree
- Number of instances at the leaf nodes could be too small to make any statistically significant decision

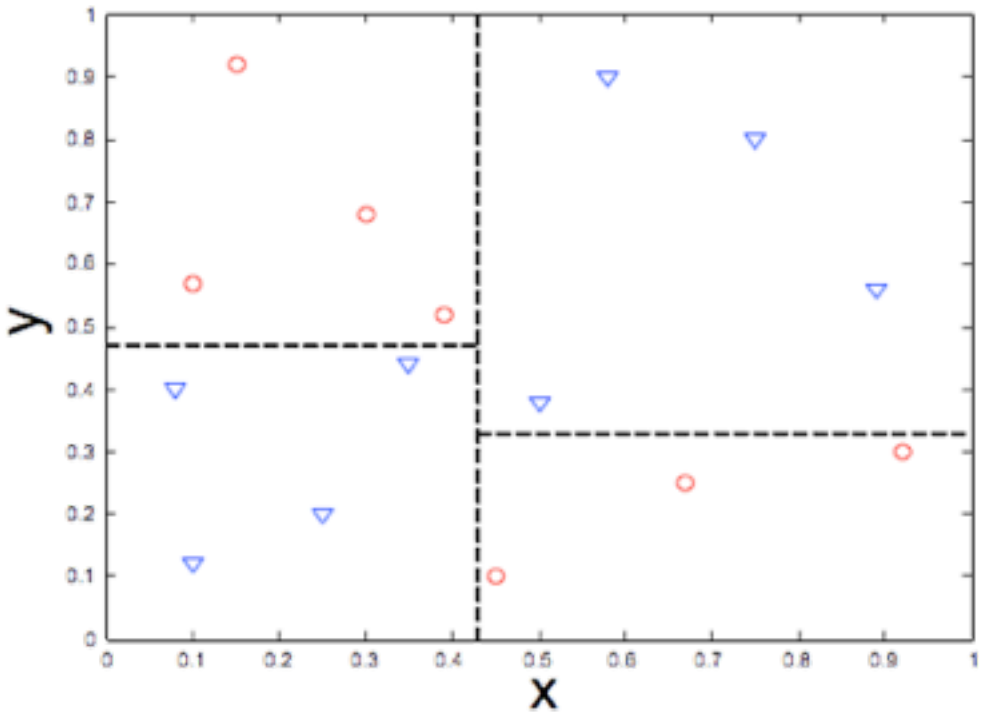
Search Strategy

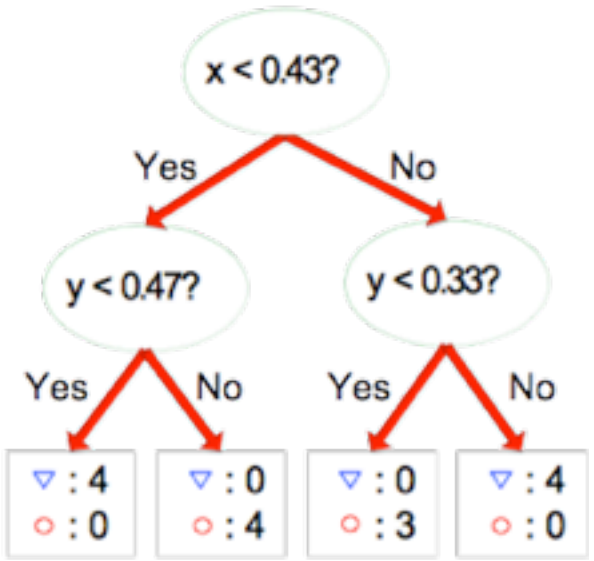
- Finding an optimal decision tree is NP-hard
- The algorithm presented so far uses a greedy, top-down, recursive partitioning strategy to induce a reasonable solution
- Other strategies?
  - Bottom-up
  - Bi-directional

Expressiveness

- Decision tree works often well for discrete functions:
  - Not always:
    - Example: parity function:
      - Class = 1 if even number of Boolean attributes with truth value = True
      - Class = 0 otherwise
    - For accurate modeling, must have a complete tree (2^d nodes, where d is the number of boolean attributes)
- Not expressive enough for modeling continuous variables
  - Particularly when test condition involves only a single attribute at-a-time

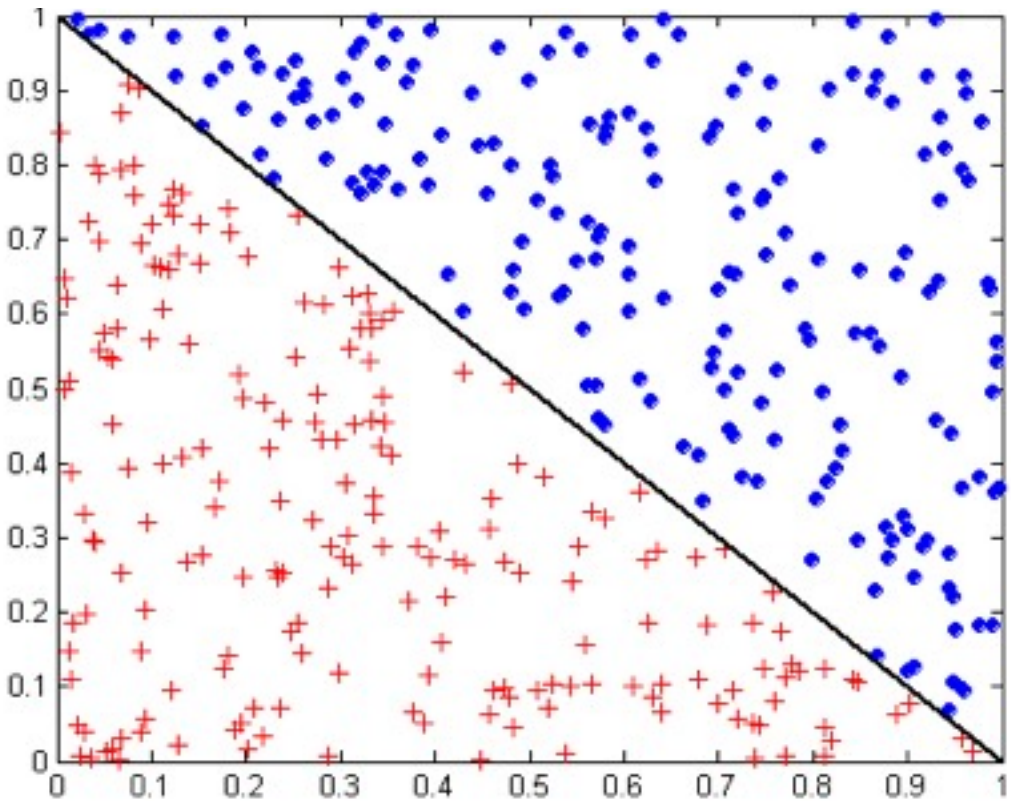
Decision Boundary



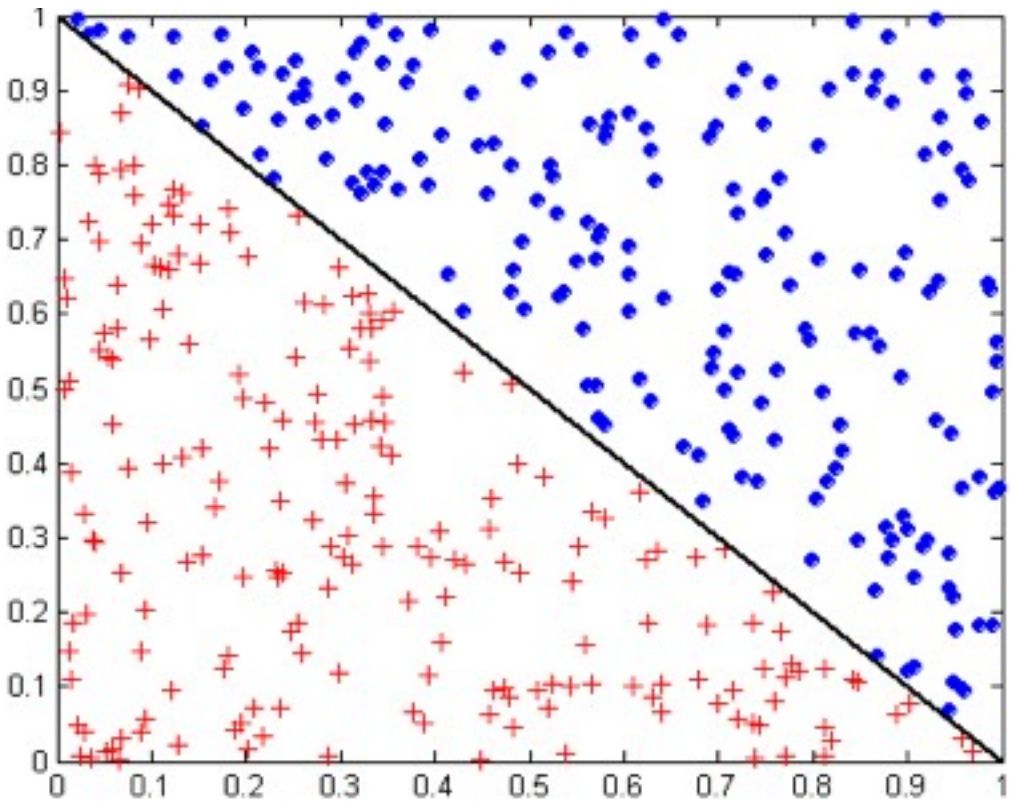


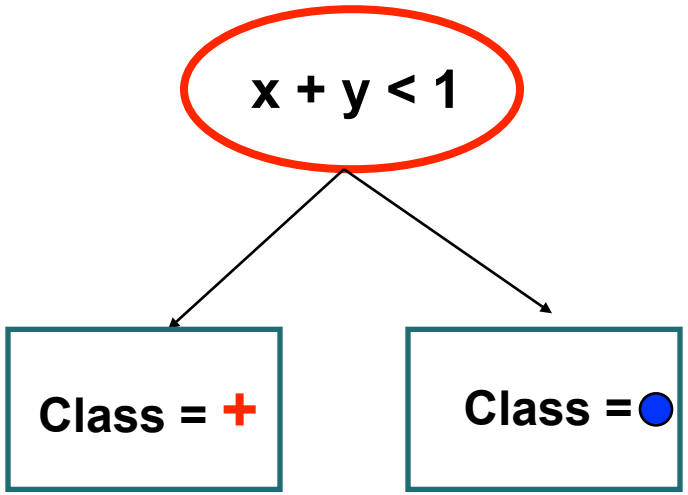
- Border line between two neighboring regions of different classes is known as decision boundary
- Decision boundary is parallel to axes because test condition involves a single attribute at-a-time

Oblique Decision Trees

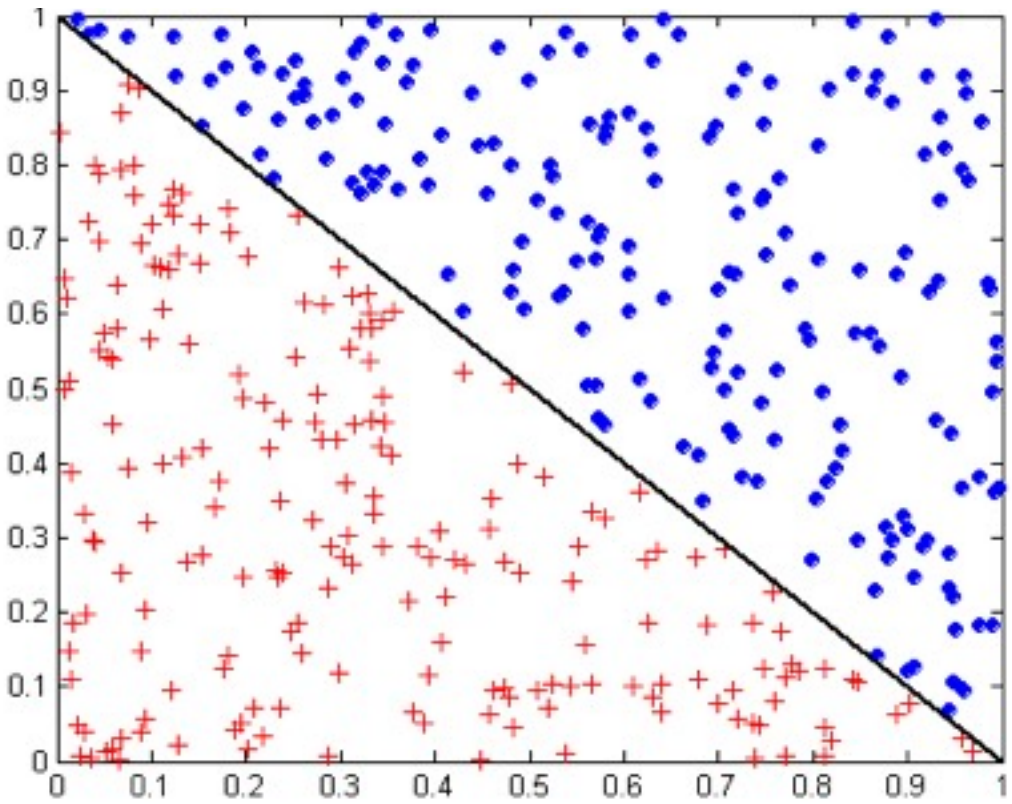


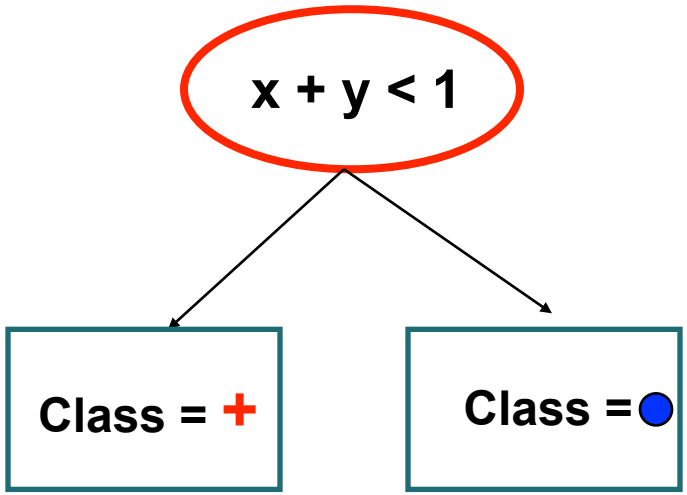
Oblique Decision Trees





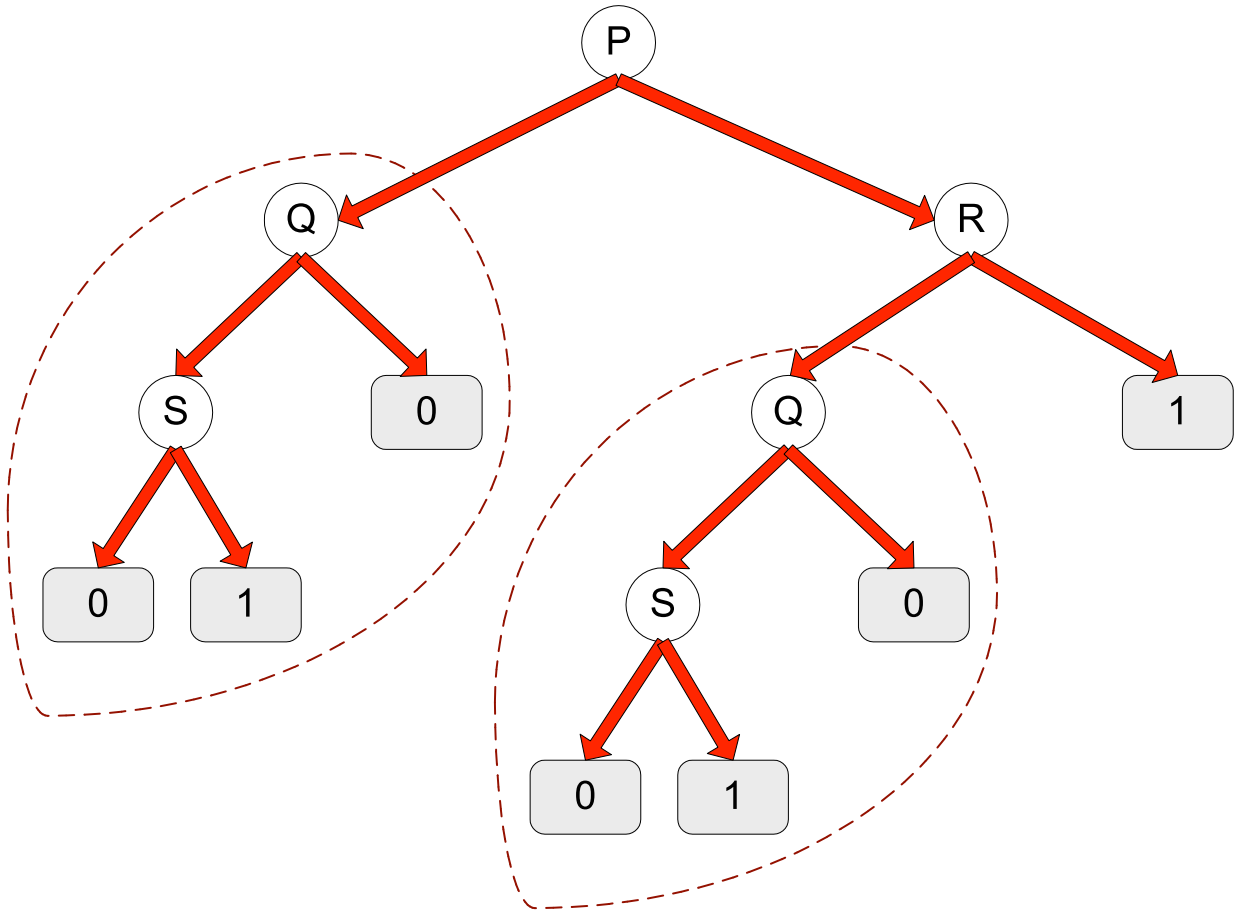
Oblique Decision Trees





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

Tree Replication



- **Replicas of the same subtree (more complicated than needed)**

Simple Decision Tree: A Pseudocode

- Pseudocode similar to C4.5 and CART. It builds a decision tree recursively which is then pruned. Let D be the set of records and A the set of attributes.

- Nodes are represented with a data structure containing:
  - a constraint on an attribute (e.g. a<= 10) (except roots)
  - a list of nodes (except the leaves) i.e. the children of the node.
  - the class value (only leaves)

- BuildDecisionTree(*D, A, minNum, alpha*)
  - create a *root* node
  - T=Build(*D, A, minNum, root*)
  - T=PostPrune(T,alpha, minNum)
  - **return** T

Simple Decision Tree: A Pseudocode

- Build(*D, A, minNum, node*)
  - If all records in D have same class *c*
    - node is a leaf with class c
    - **return**
  - if the # of records in D is < *minNum*
    - node is a leaf with the default class value *d*
    - ***return***
  - if A = empty set
    - the class of *node* is by majority or default if tie with majority class; **return**
  - let $a, v_1, \ldots, v_k$ be an attribute and the corresponding values maximizing information gain in D, respectively. Let D_i be the set of records satisfying the corresponding constraints on *a*.
  - A = A \ {a}
  - for i=*1,..,k*
    - create node n_i with corresponding constraint on *a* with value v_i
    - add n_i to the list of children of *node*
    - BuildDecisionTree(D_i, A, minNum, n_i)

Postpruning: Pseudocode

- The level of the root is 1. The level of a node other than the root is equal to the level of its parent + 1. levels(T) is the max level of any node in T.

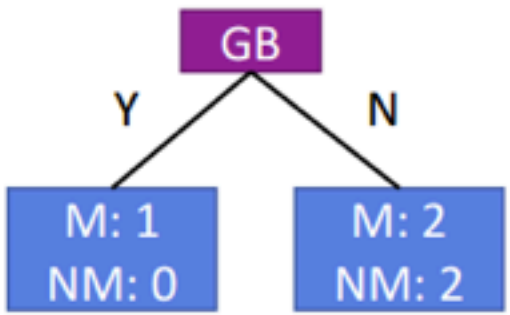
- PostPrune(*T,alpha, minNum*)
- For l=levels(T) downto 1
  - For every node n at level l in T
    - let P be the tree obtained by replacing the subtree rooted in n with a leaf, associated to all records in the subtree. Its class value is c if all records have class c; the default class if the number of records is < *minNum* or there is a tie between the default and the majority class; otherwise it is determined by majority.
    - if gen_error(P) < gen_error(T) then T = P.

Exercise: Decision Trees

Animal	Give Birth (GB)	Live in Water (LW)	Class
A1	Y	Y	M
A2	N	Y	M
A3	N	Y	NM
A4	N	N	M
A5	N	N	NM

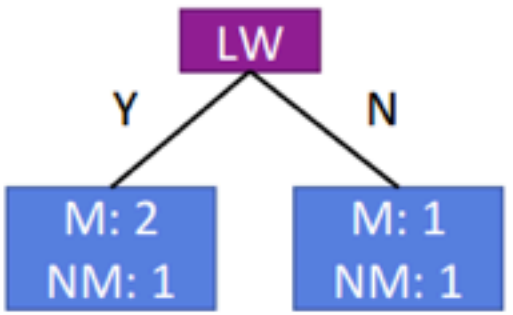
Construct a decision tree with the following properties:
1. Choose the best split based on GINI index;
2. Stopping rule: when gini value=0 or no further split is possible;
3. Class of a leaf is determined by majority rule. If ties use the default class (NM).

Building Decision Tree: Step 1



$GINI_1 = 0$ $GINI_2 = \frac{1}{2}$

$$GINI_{split} = \frac{1}{5} \times 0 + \frac{4}{5} \times \frac{1}{2} = \frac{2}{5}$$

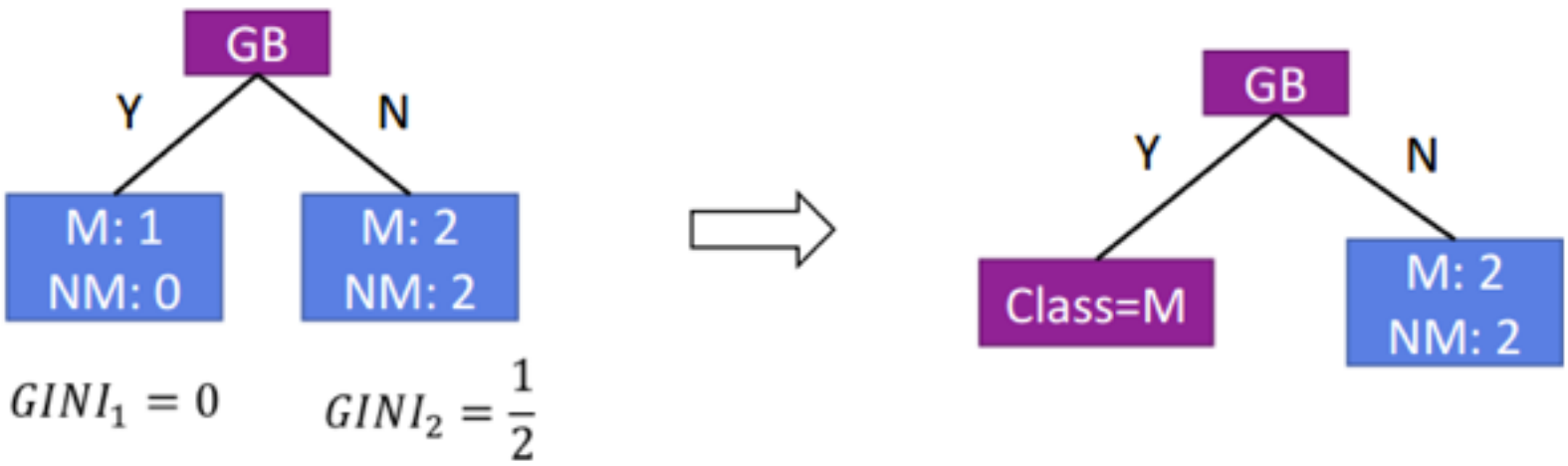


$GINI_1 = \frac{4}{9}$ $GINI_2 = \frac{1}{2}$

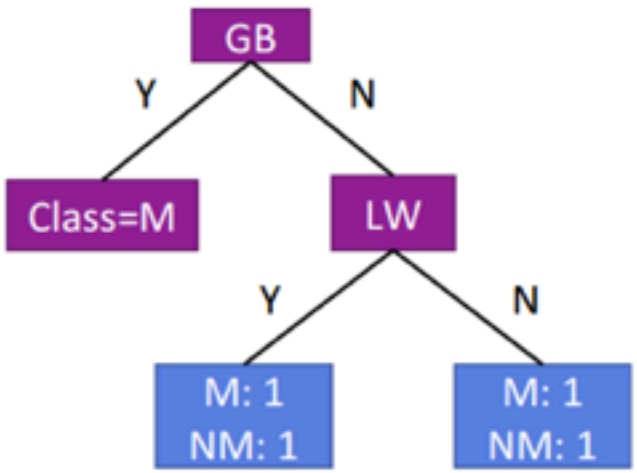
$$GINI_{split} = \frac{3}{5} \times \frac{4}{9} + \frac{4}{5} \times \frac{1}{2} = \frac{7}{15}$$

Building Decision Tree: Step 2

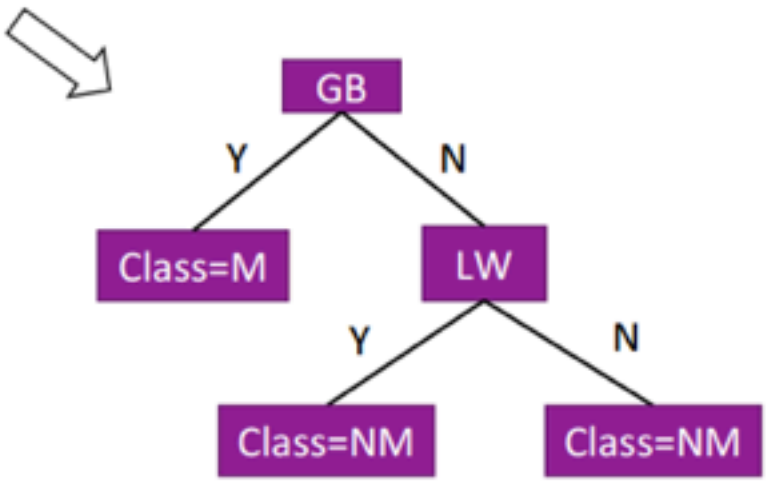
Left child node: Since $Gini_1$ is already 0, we don't need to split the left child node. We have a leaf node with the majority class.



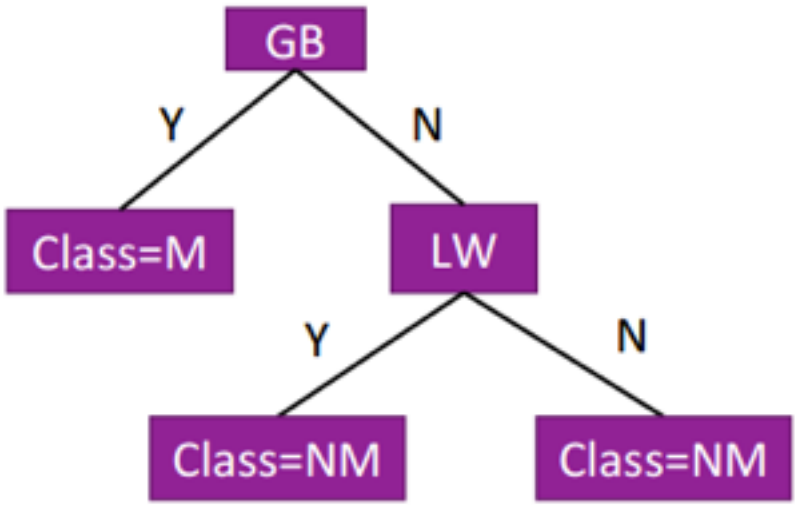
Building Decision Tree: Step 3



- No further split is possible
- Number of ***M*** records = Number of ***NM*** records for both child nodes
- You can choose an arbitrary class label, or use a default one (e.g., ***NM***)

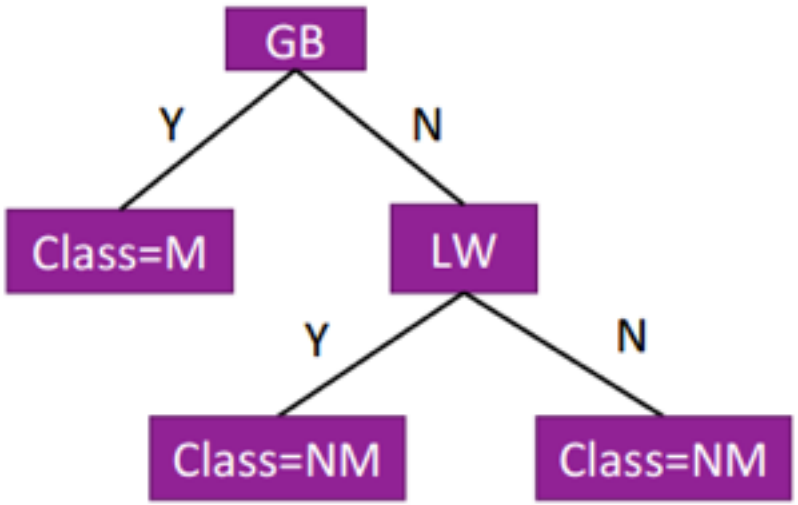


Decision trees: predicting



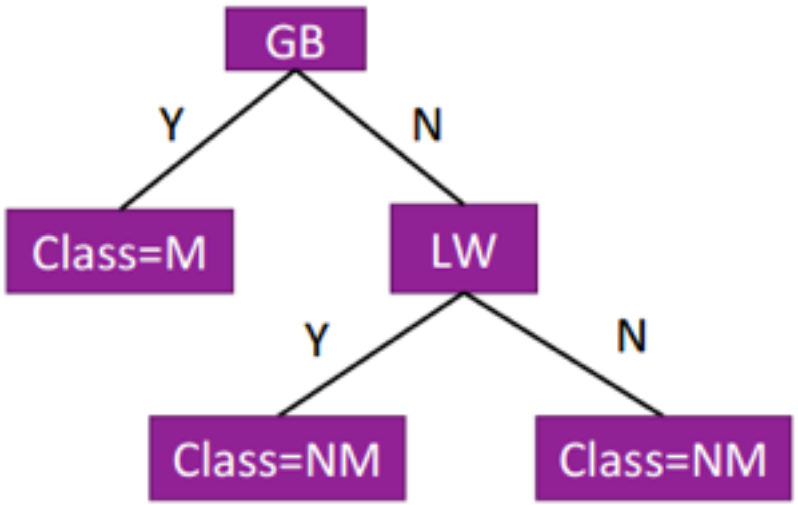
What is the class of GB='N', LW='Y'?

Decision trees: predicting



What is the class of GB='N', LW='Y'? NM

Decision trees: post pruning

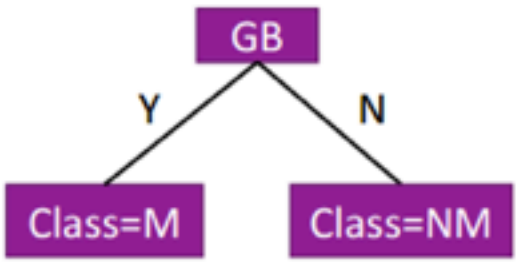


Should we prune the subtree rooted in LW?

Decision trees: post pruning

Training Error = Number of misclassified records
Generalization Error = Training Error + 0.5 * number of leaves

Before Splitting

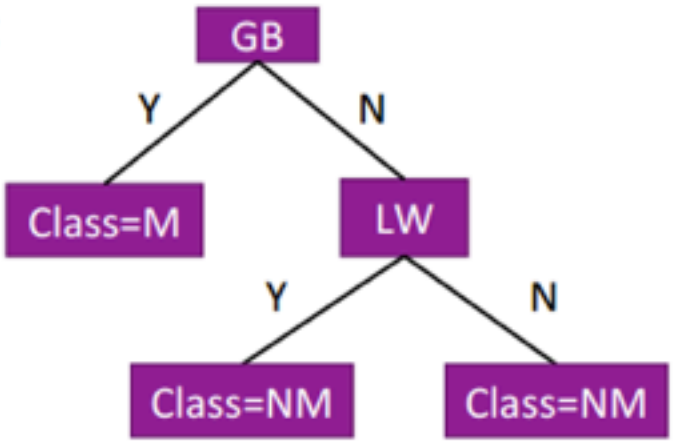


Training Error = 2

Number of leaves = 2

Generalization Error = 3

After Splitting



Training Error = 2

Number of leaves = 3

Generalization Error = 3.5

So we prune the sub-tree!

References

- Chapter 4 of Introduction to Data mining, Pang-Ning Tan, M. Steinbach, V. Kumar, March 2006. Also available online: http://www-users.cs.umn.edu/~kumar/dmbook/ch4.pdf

- Chapter 19 of DATA MINING AND ANALYSIS

Fundamental Concepts and Algorithms, M. Zaki, W. Meira Jr.