Decision Trees

CMT307 Session 6

Yuhua Li
liy180@cardiff.ac.uk

Classification: Definition

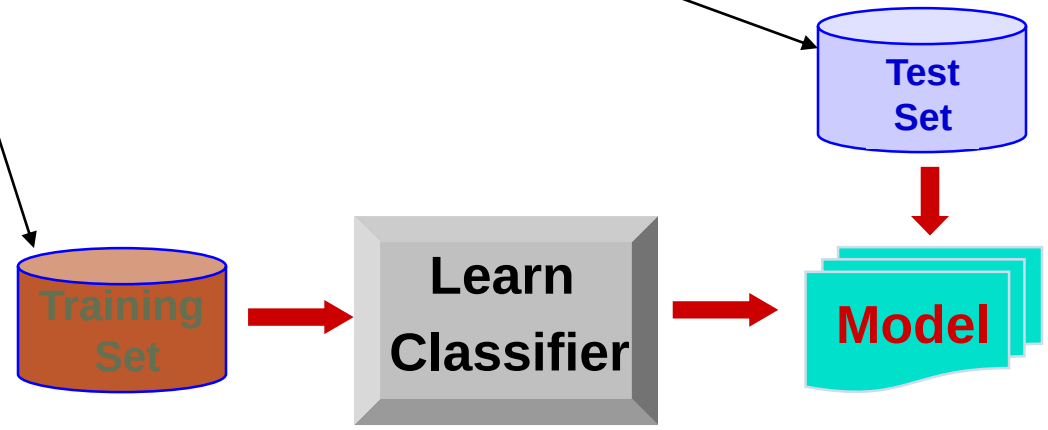
- Given a collection of records (*training set*)
  - Each record contains a set of *attributes*, one of the attributes is the *class*.
- Find a *model* for class attribute as a function of the values of other attributes.
- Goal: previously unseen records should be assigned a class as accurately as possible.
  - 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.

Classification Example

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

Refund	Marital Status	Taxable Income	Cheat
No	Single	75K	?
Yes	Married	50K	?
No	Married	150K	?
Yes	Divorced	90K	?
No	Single	40K	?
No	Married	80K	?



Decision Tree

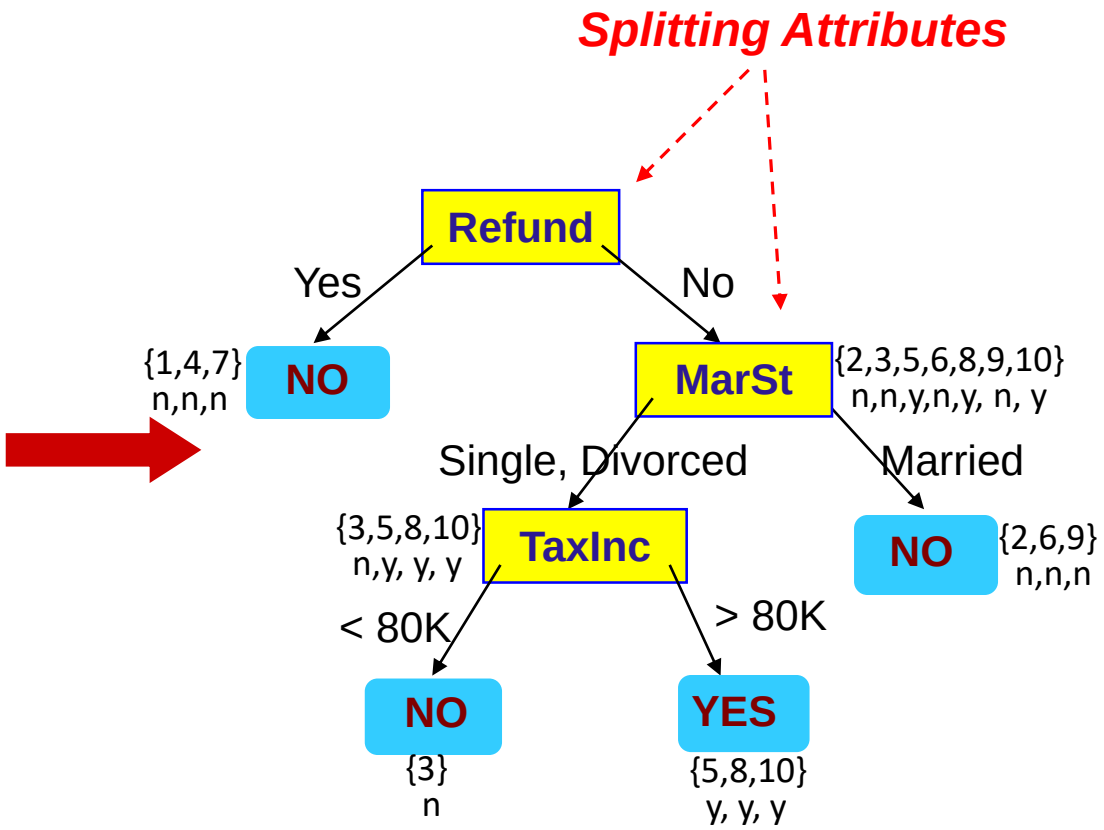
- A **root node** that has no incoming edges and zero or more outgoing edges
- An **internal node** is a test on an attribute
- A **branch** represents an outcome of the test
- A **leaf** or **terminal node** represents a class label
- At each node, one attribute is chosen to **split** training examples into distinct classes as much as possible

- A new case is classified by following a matching path to a leaf node.

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

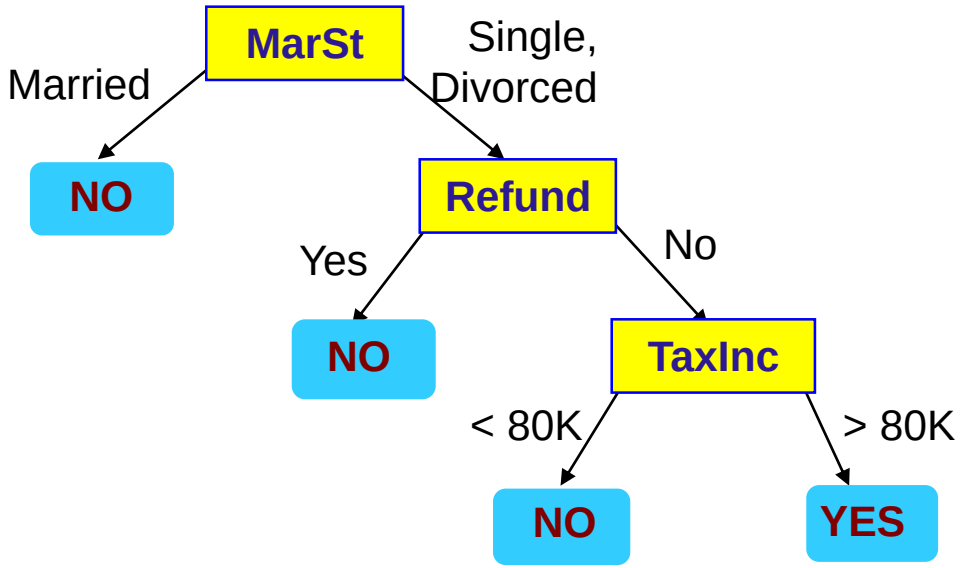
Training Data



Model: Decision Tree

Example of Decision Tree

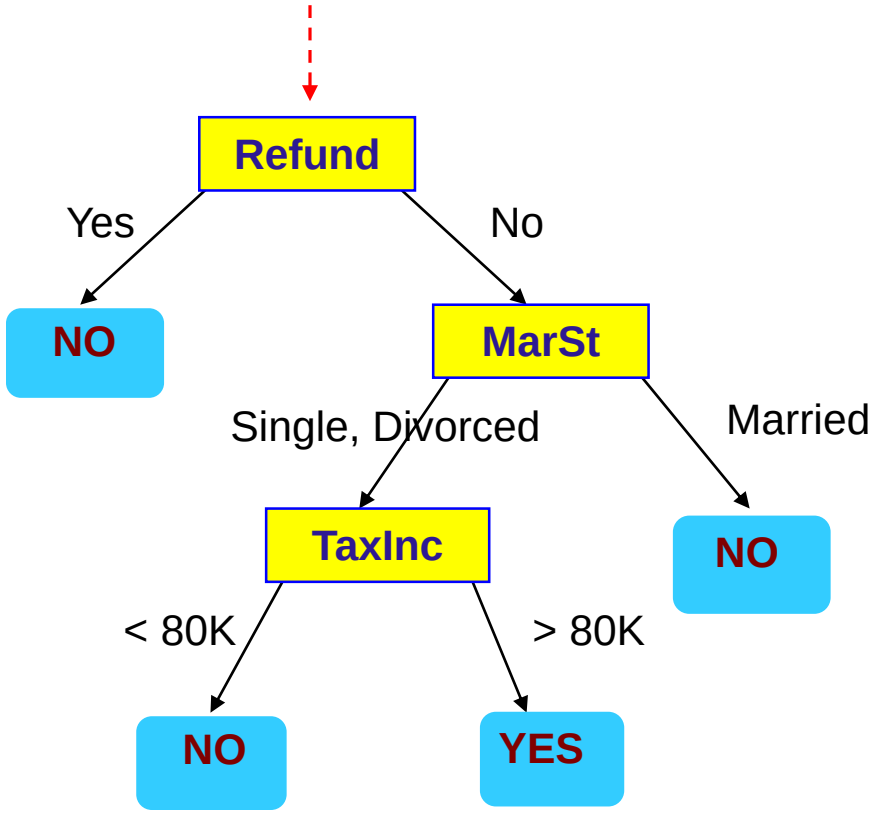
	categorical	categorical	continuous	class
Tid	**Refund**	**Marital Status**	**Taxable Income**	**Cheat**
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single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



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

Apply Model to Test Data

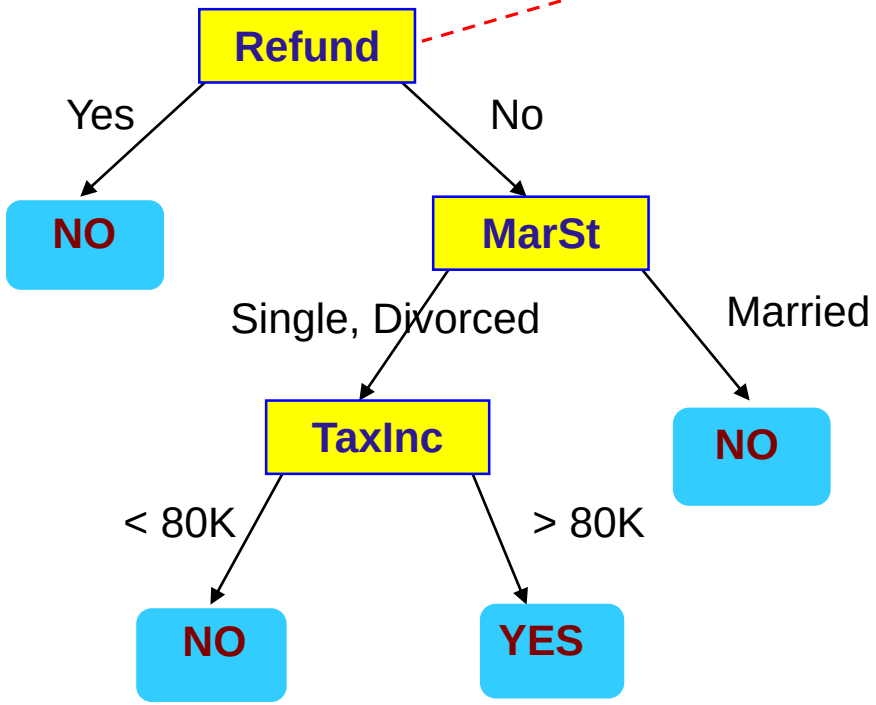
Start from the root of tree.



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

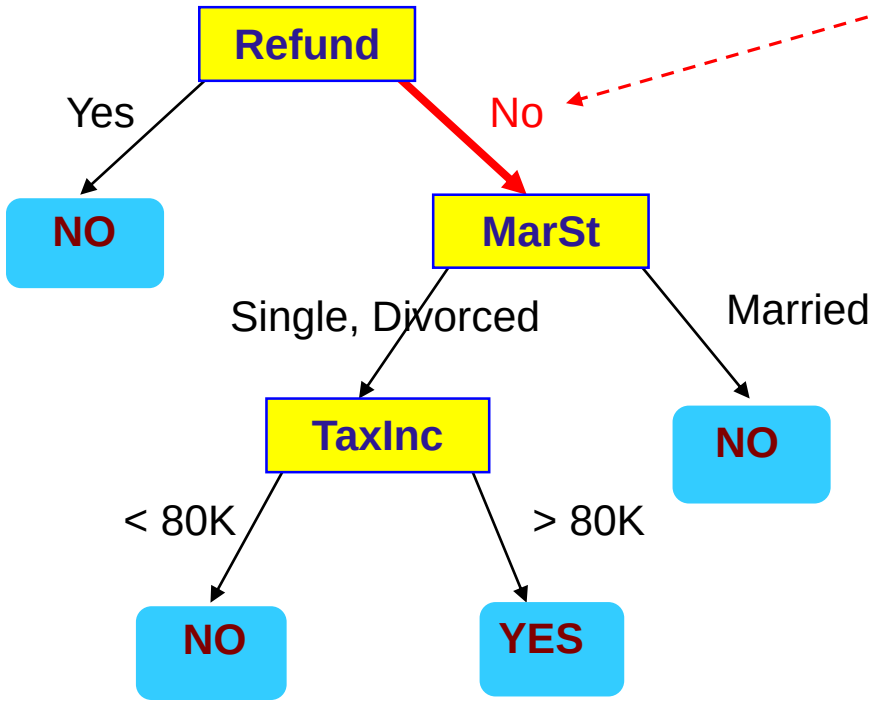
Apply Model to Test Data



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

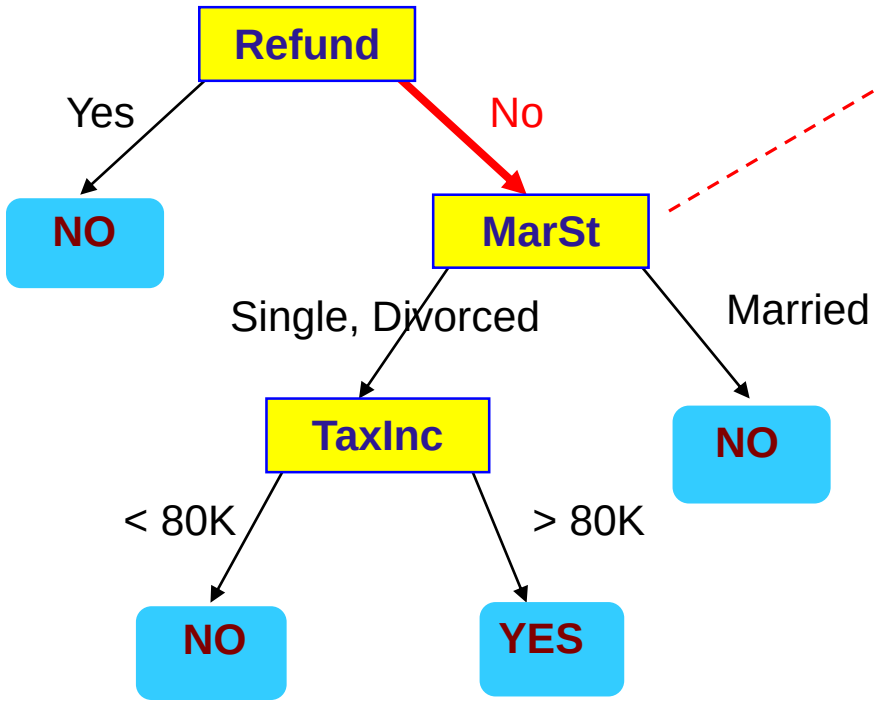
Apply Model to Test Data



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

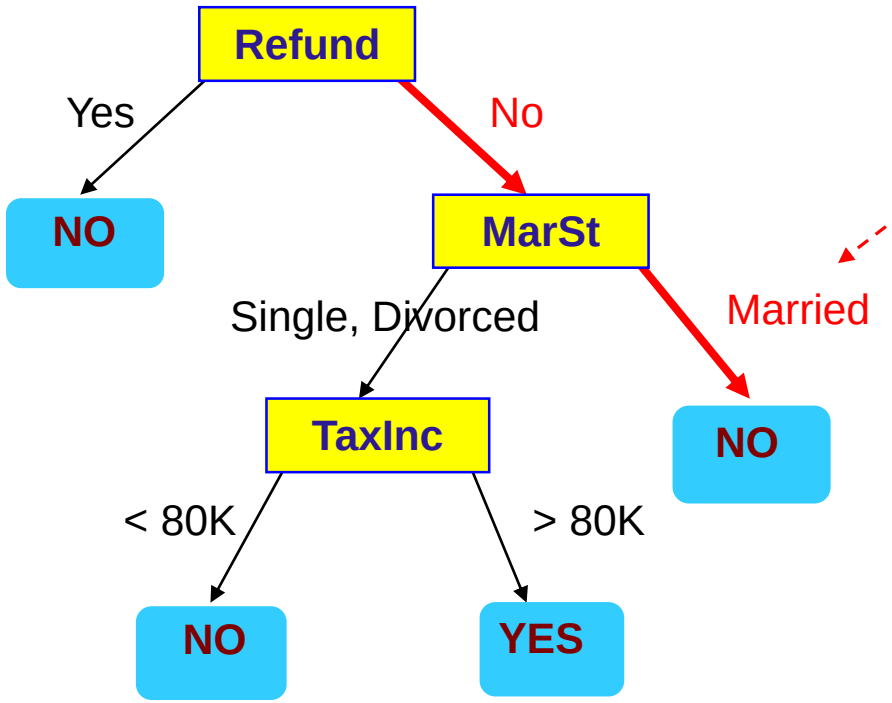
Apply Model to Test Data



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

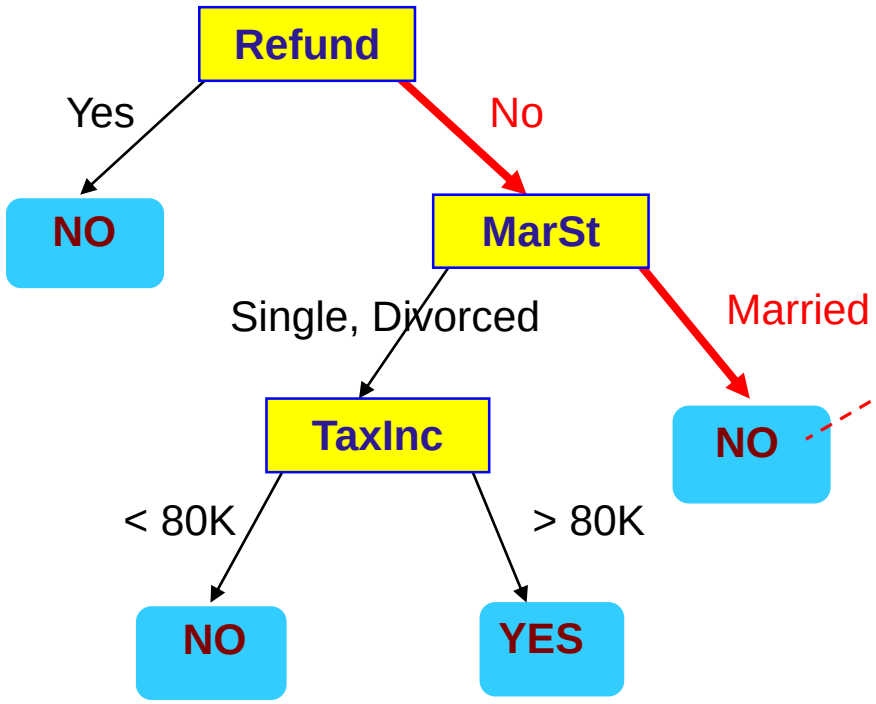
Apply Model to Test Data



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	?

Assign Cheat to “No”

Decision Tree Algorithms

- Many Algorithms:
  - Hunt's (one of the earliest)
  - CART (Classification And Regression Tree)
  - ID3 (Iterative Dichotomiser 3)
  - C4.5 (successor of ID3)
  - SLIQ (Supervised Learning In Quest), SPRINT

Choosing the Splitting Attribute

- At each node, available attributes are evaluated on the basis of separating the classes of the training examples. A Goodness function is used for this purpose.
- Typical goodness functions:
  - information gain (Entropy) used in (ID3/C4.5)
  - GINI index (IBM Intelligent Miner)

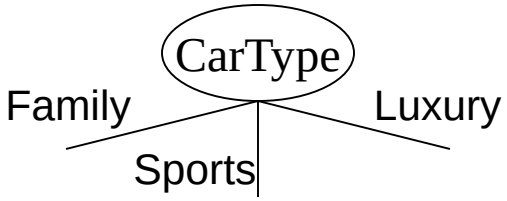
How to Specify Test Condition?

- Depends on attribute types
  - Nominal: takes values from an unordered set. Marital status, Gender, Colour, etc.
  - Continuous: takes values from ordered set. Age, Salary, etc.

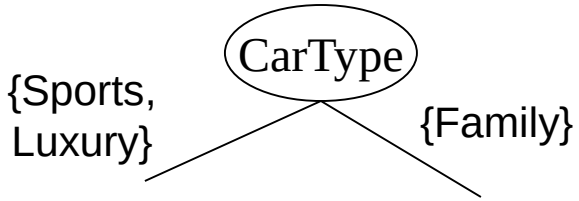
- Depends on number of ways to split
  - 2-way split
  - Multi-way split

Splitting Based on Nominal Attributes

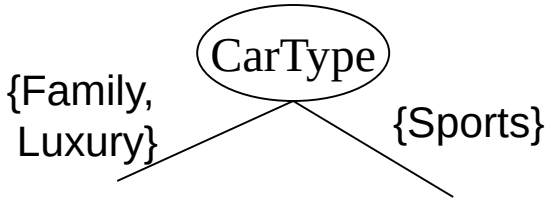
- Multi-way split: Use as many partitions as distinct values.



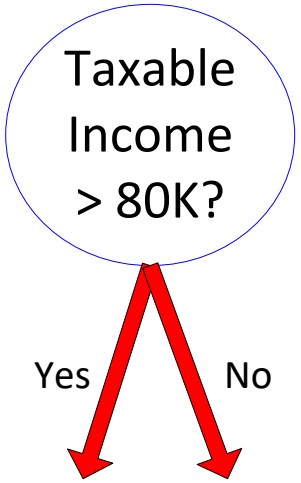
- Binary split: Divides values into two subsets.
  Need to find optimal partitioning.



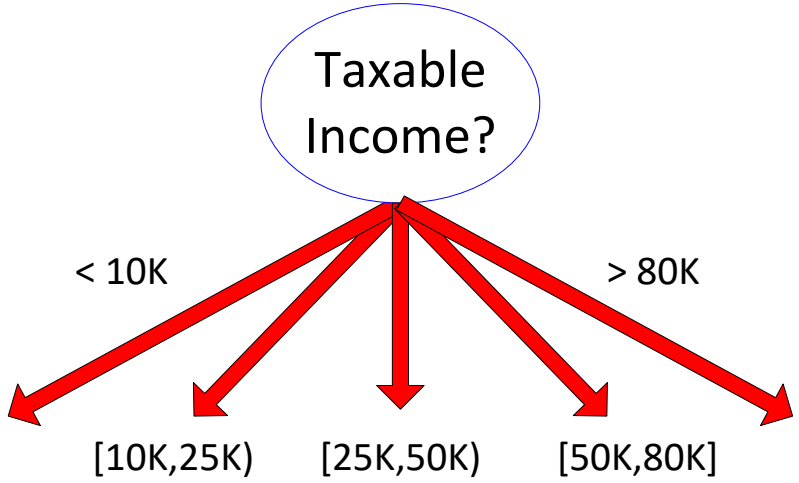
OR



Splitting Based on Continuous Attributes



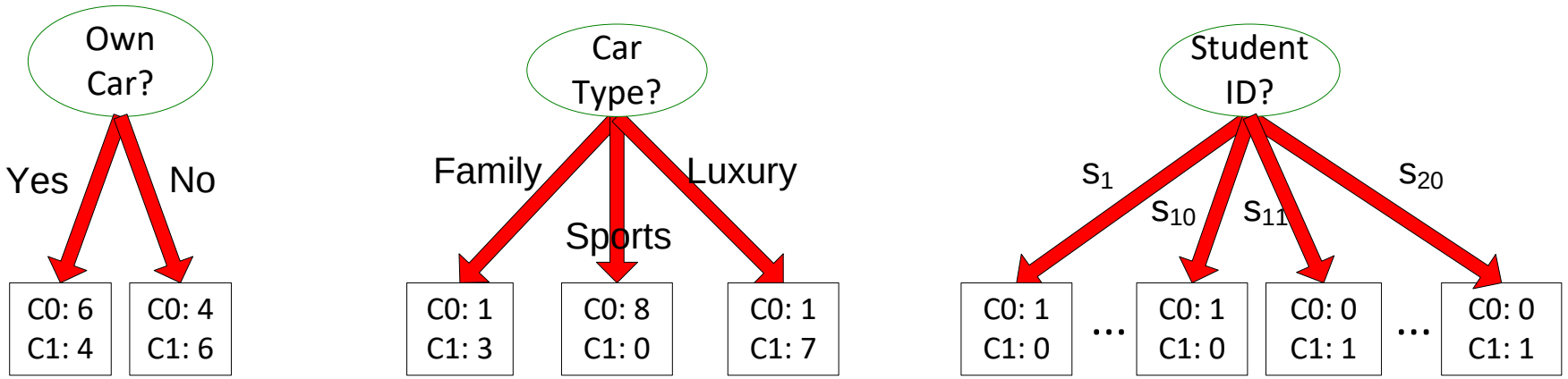
(i) Binary split



(ii) Multi-way split

How to determine the Best Split

Before Splitting: 10 records of class 0 (C0), 10 records of class 1 (C1)



Which test condition is the best?

How to determine the Best Split

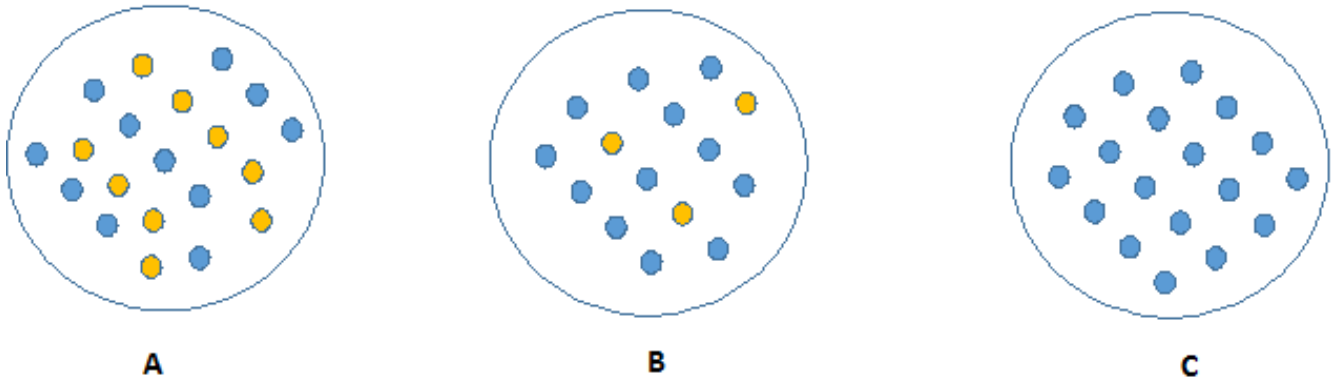
- Greedy approach:
  - Nodes with homogeneous class distribution are preferred
- Need a measure of node impurity:

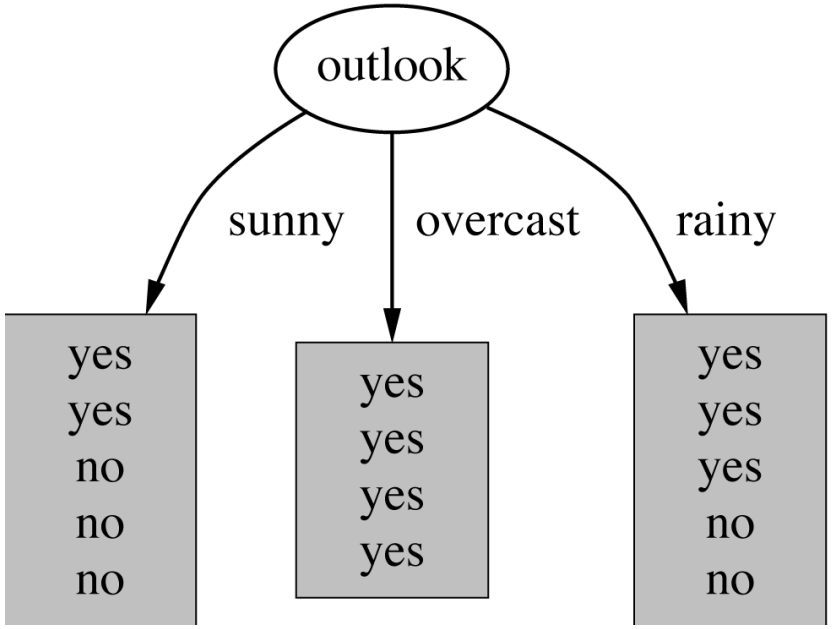
C0: 5 C1: 5	C0: 9 C1: 1
Non-homogeneous, **High degree of impurity**	**Homogeneous,** **Low degree of impurity**

A criterion for attribute selection

- Which is the best attribute?
  - The one which will result in the smallest tree
  - Heuristic: choose the attribute that produces the “purest” nodes
- Popular *impurity criterion*: *information gain*
  - Information gain increases with the average purity of the subsets that an attribute produces
- Strategy: choose attribute that results in greatest information gain

Node purity / homogeneity





Computing information

- Given a probability distribution, the information required to predict an event is the distribution's ***entropy*** *(*Shannon entropy)
- Generally, entropy refers to disorder or uncertainty
- Shannon entropy was introduced by Claude E. Shannon in 1948

Splitting Criteria based on Information Gain

- 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 relative frequency of class j at node t).

  - Measures homogeneity of a node.
    - Maximum entropy when records are equally distributed among all classes implying least information
    - Minimum entropy (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 = – 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

  - Measures Reduction in Entropy achieved because of the split. Choose the split that achieves most reduction (maximizes GAIN)
  - Used in ID3 and C4.5

Splitting based on *Gini* Index

- If a data set *T* contains examples from *c* classes, gini index, *gini*(*T*) is defined as

$$gini(T) = 1 - \sum_{j=1}^{c} p_j^2$$

Example:
a node with 3 classes, each having samples [0, 49, 5]
gini(*T*) = 1 – (0/54)² – (49/54)² – (5/54)² ≈ 0.168.

where p_j is the relative frequency of class *j* in *T.*

- If a data set *T* of *N* points is split into two subsets T_1 and T_2 with sizes N_1 and N_2 respectively, the *gini* index of the split data contains examples from *c* classes, the *gini* index *gini*(*T*) is defined as

$$gini_{split}(T) = \frac{N_1}{N} gini(T_1) + \frac{N_2}{N} gini(T_2)$$

- The attribute provides the smallest $gini_{split}(T)$ is chosen to split the node (*need to enumerate all possible splitting points for each attribute*).

Gini Index

- **Gini index measures the quality of split by computing node impurity as,**

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

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

- **When a node p is split into k partitions (children), 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 i,

n = number of records at node p.

Weather Data: Play or not Play?

- Training Examples

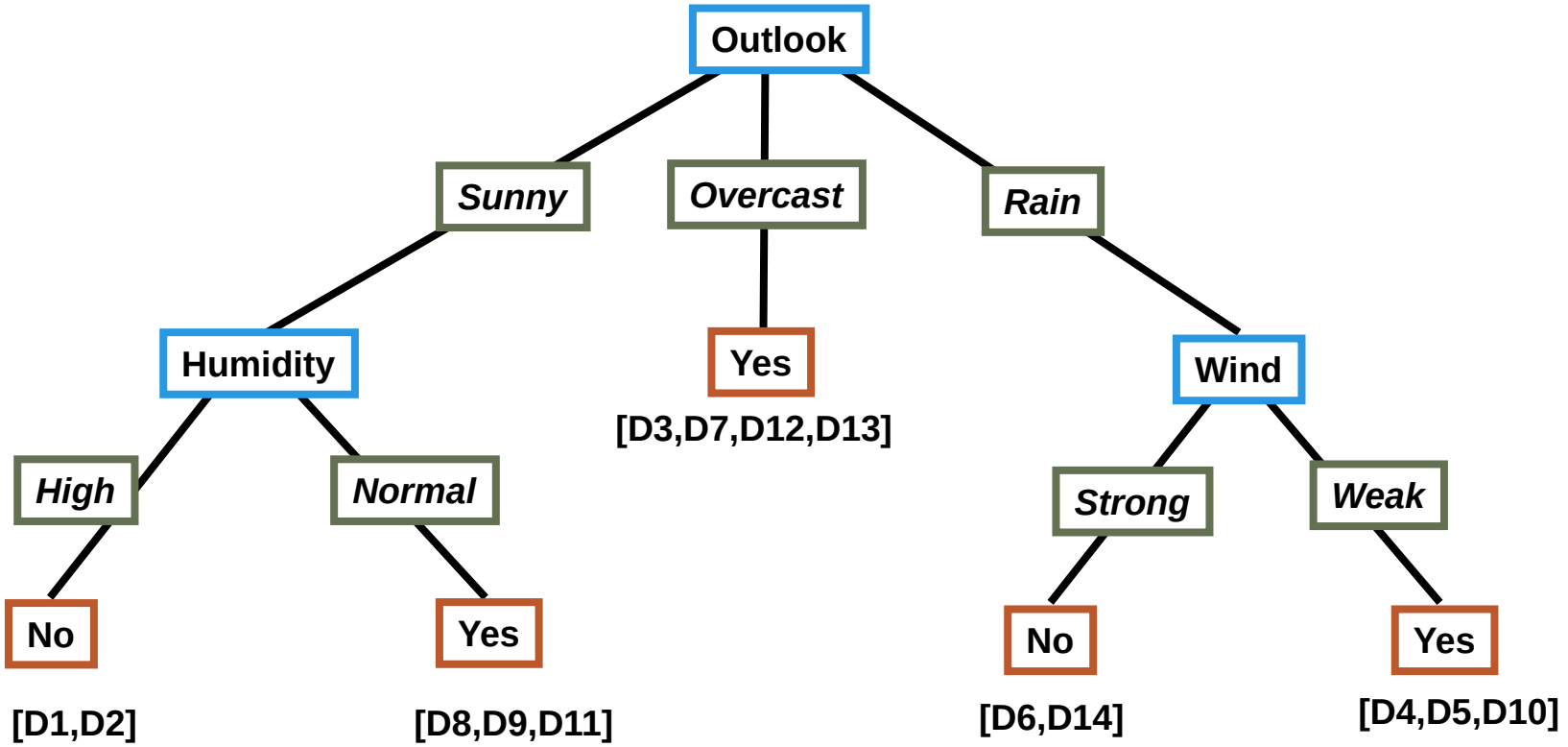
Day	Outlook	Temp.	Humidity	Wind	Play Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Weak	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Strong	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Decision Tree for PlayTennis

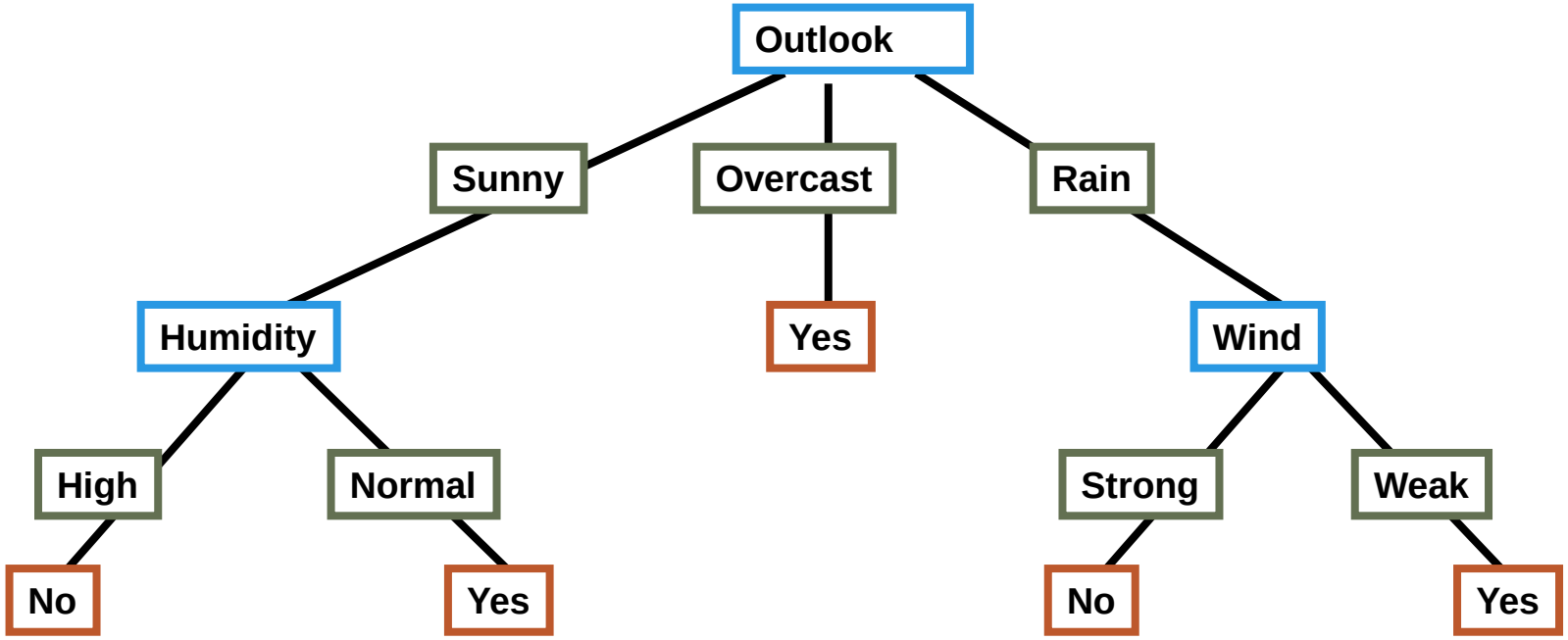
- Attributes and their values:
  - Outlook: *Sunny, Overcast, Rain*
  - Humidity: *High, Normal*
  - Wind: *Strong, Weak*
  - Temperature: *Hot, Mild, Cool*

  - Target concept - Play Tennis: *Yes, No*

Decision Tree for PlayTennis from ID3 Algorithm



Converting a Tree to Rules



R_1: If (Outlook=Sunny) $\wedge$ (Humidity=High) Then PlayTennis=No
R_2: If (Outlook=Sunny) $\wedge$ (Humidity=Normal) Then PlayTennis=Yes
R_3: If (Outlook=Overcast) Then PlayTennis=Yes
R_4: If (Outlook=Rain) $\wedge$ (Wind=Strong) Then PlayTennis=No
R_5: If (Outlook=Rain) $\wedge$ (Wind=Weak) Then PlayTennis=Yes

Regression Trees

- Training of regression trees is the same as that of classification tree, except to split the training set in a way that minimizes the MSE
- If a data set T of N points is split into two subsets T_1 and T_2 with sizes N_1 and N_2 respectively, then the cost function is

$$J(k, t_k) = \frac{N_1}{N} \mathrm{MSE}_{T_1} + \frac{N_2}{N} \mathrm{MSE}_{T_2}$$

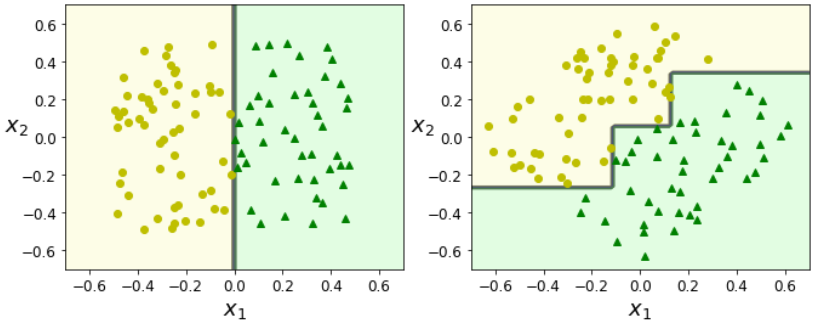
where

$$\begin{cases} \mathrm{MSE}_{\mathrm{node}} = \sum_{i \in \mathrm{node}} \left(\hat{y}_{\mathrm{node}} - y^{(i)} \right)^2 \\ \hat{y}_{\mathrm{node}} = \frac{1}{N_{\mathrm{node}}} \sum_{i \in \mathrm{node}} y^{(i)} \end{cases}$$

Remarks

- Pros
  - Interpretable decision process: white box model
  - Decision trees requires little data preparation: no need of feature scaling or centering at all

- Cons
  - Splits are perpendicular to an axis, so sensitive to training set rotation
    - Solution: use PCA to get a better orientation of the training data
  - Very sensitive to small variations in the training data, leading to different trees.
    - Solution: use Random Forest to limit instability



Reading

- Aurélien Géron. Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools and Techniques to Build Intelligent Systems. O'Reilly, 2017.
  Chapter 6.