

# **Machine Learning: Classification Algorithms – Decision Trees**



---

Won Kim



# Classification Algorithms

---

- **Decision Trees**
- Logistic Regression
- Support Vector Machine
- Bayesian Classifier



# Roadmap: Decision Tree

---

- Decision Tree
- Decision Tree Regression
- Decision Tree Pruning



# Acknowledgments

---

- <http://www.cs.kent.edu/~jin/DM07/ClassificationDecisionTree.ppt>



---



# **A Lightning Review (from the Data Science course)**

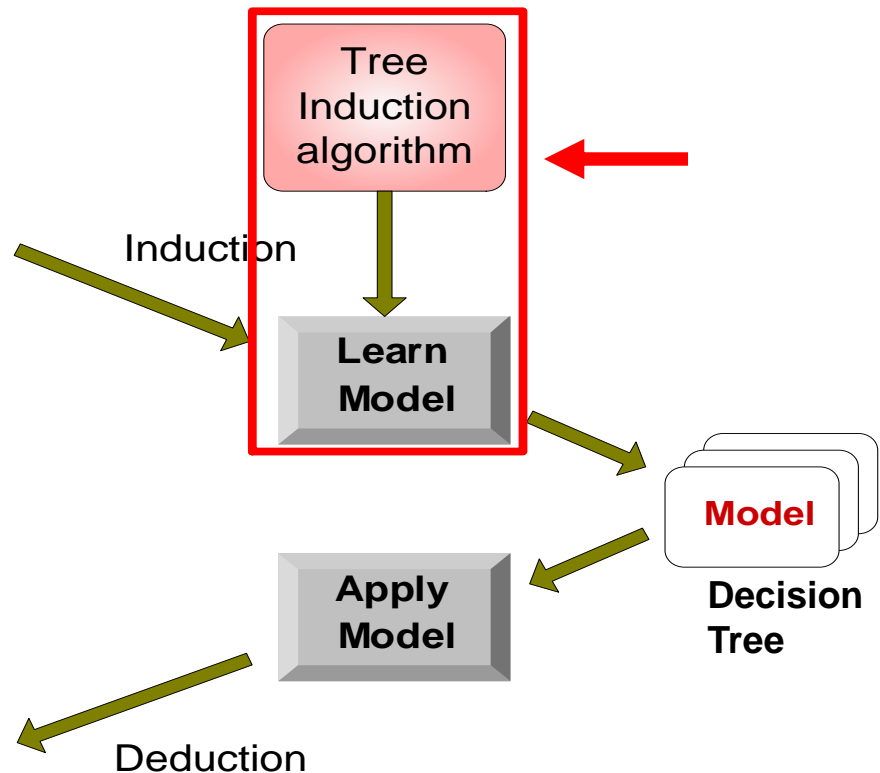
# Decision Tree for Classification Task

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



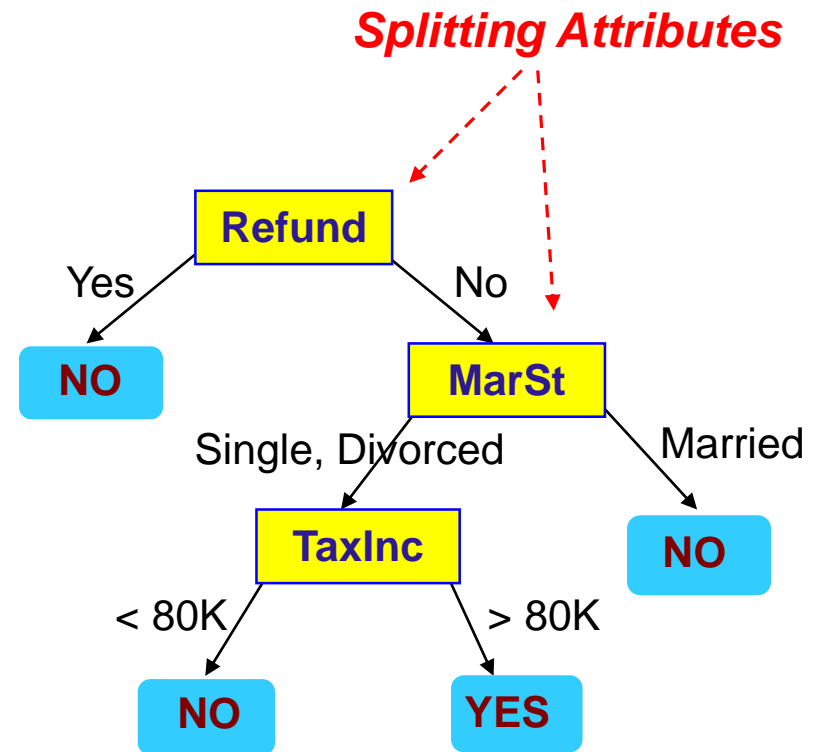
# Example 1

(\* predicting tax cheaters \*)

categorical  
categorical  
continuous  
target

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

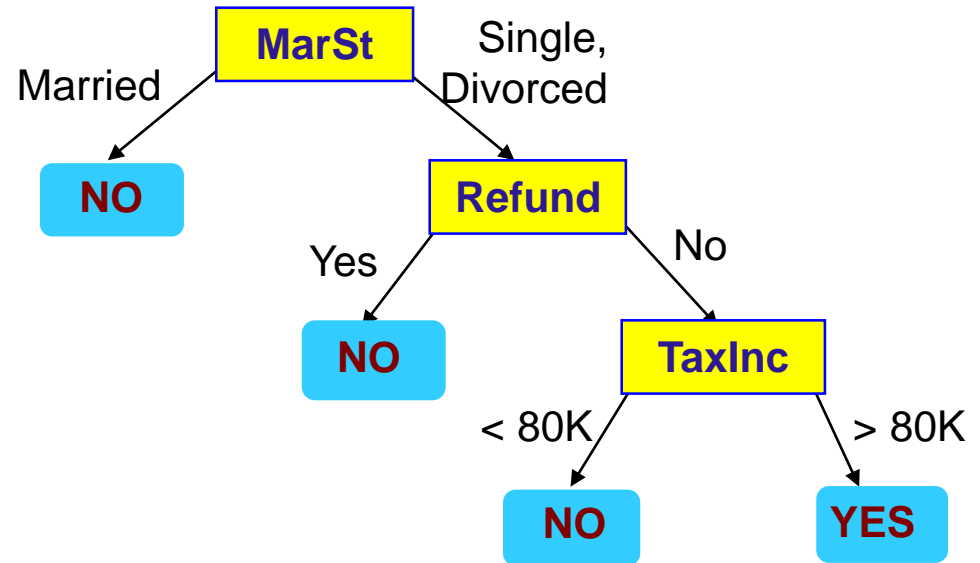


Model: Decision Tree

## Example 2

categorical  
categorical  
continuous  
target

<i>Tid</i>	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

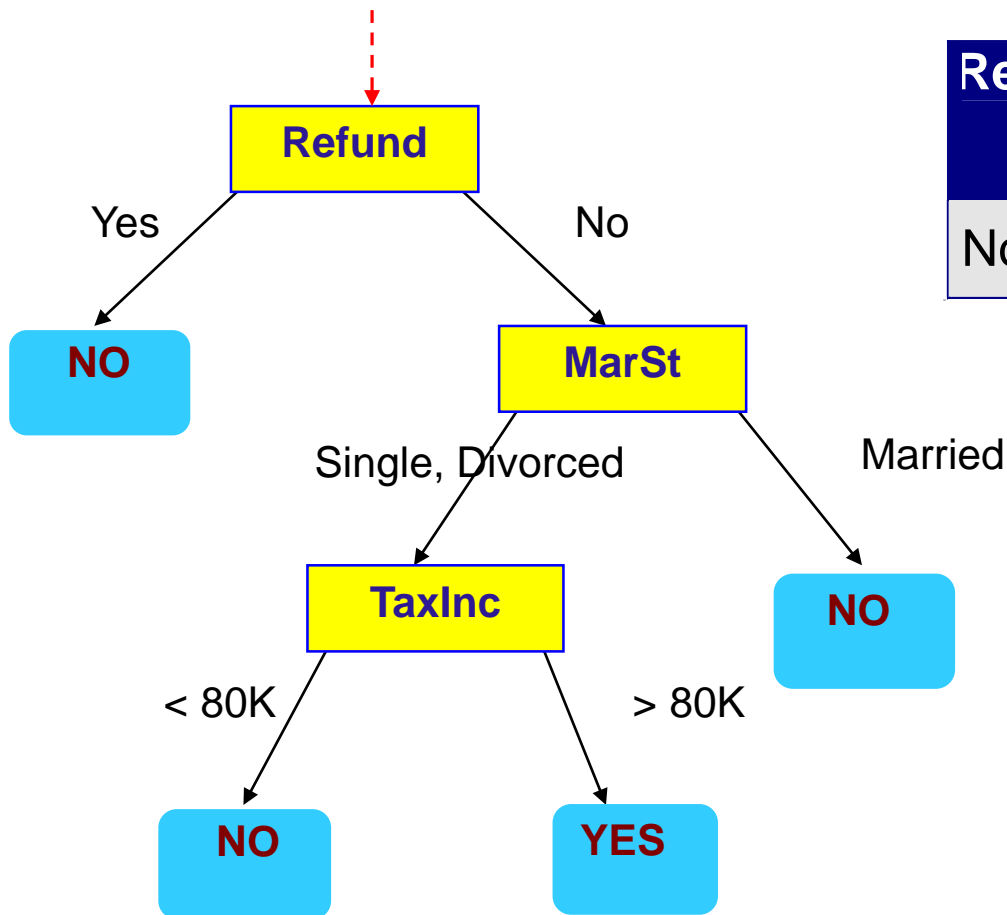


There could be more than one tree (hundreds!) that fits the same data.



# Apply Model to Test Data (1/2)

Start from the root of tree.



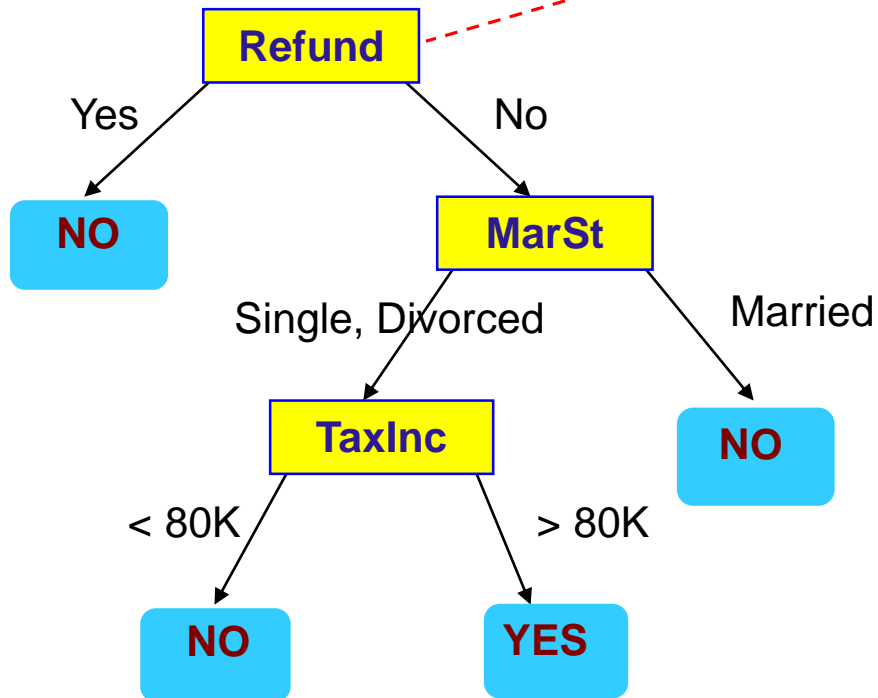
Sample Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

# Apply Model to Test Data (2/2)

Sample Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?





# Issues in Tree Induction

- Tree deduction is easy, but induction is not.
- Determine how to split the records
  - How to specify the attribute test condition?
  - How to determine the best split?
- Determine when to stop splitting
- **\*\* note:** record=sample=observation=instance  
attribute=feature=independent variables  
=predictor variable  
target=class=dependent variable



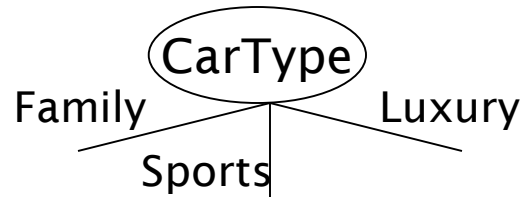
# How to Specify the Test Condition?

---

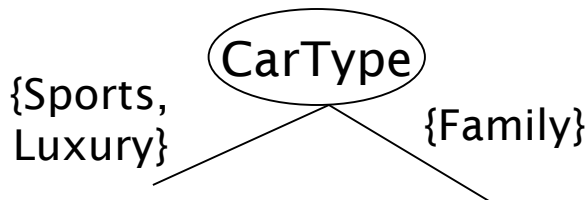
- Depends on attribute types
  - categorical attribute
    - nominal attribute (no order; gender (male, female))
    - Ordinal attribute (order: "S, M, L, XL")
  - Continuous attribute (integer, real)
- Depends on the number of ways to split
  - 2-way split (e.g., gender)
  - multi-way split (e.g., days of the week)

# Splitting Nominal Attributes

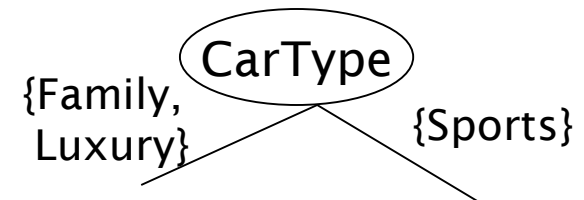
- **Multi-way split:** Use as many partitions as distinct values.



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

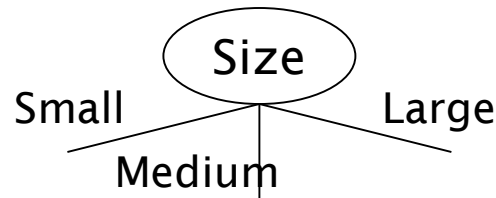


OR

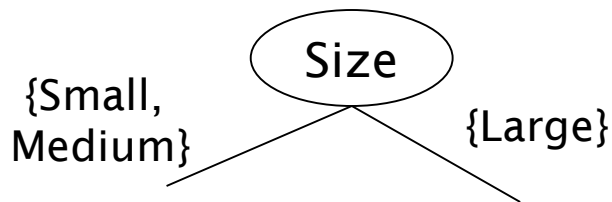


# Splitting Ordinal Attributes

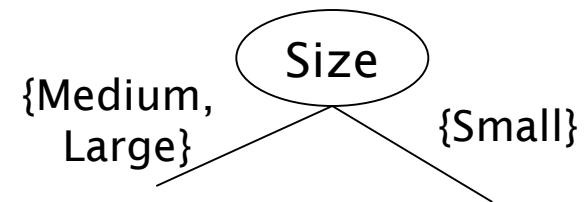
- **Multi-way split:** Use as many partitions as distinct values.



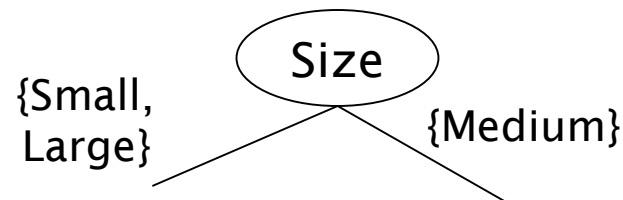
- **Binary split:** Divide values into two subsets.  
Need to find optimal partitioning.



OR



- What about this split?

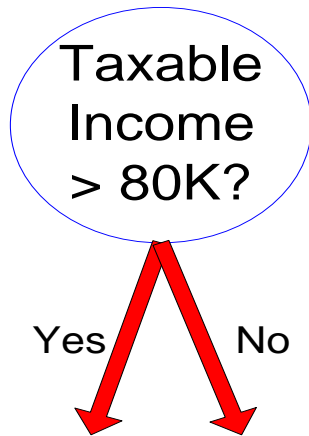




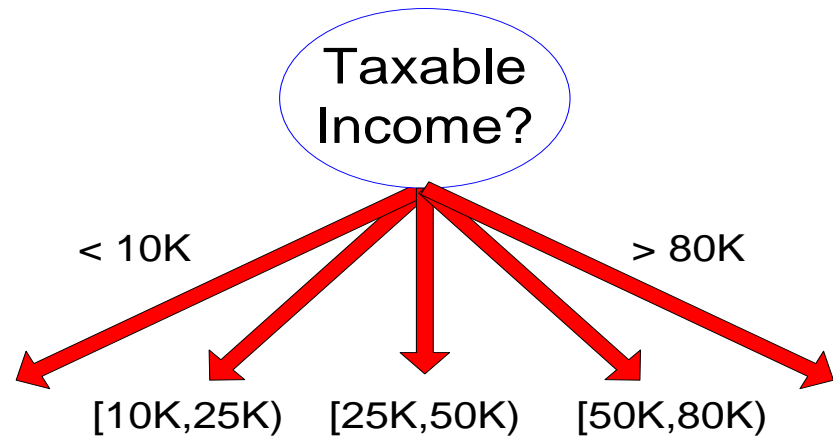
# Splitting Continuous Attributes

- Two ways
  - **Discretization** to form an ordinal categorical attribute (e.g., age group (10, 20, 30...), score group)
    - Static – discretize once at the beginning
    - Dynamic – ranges can be found by equal interval bucketing, equal frequency bucketing (percentiles), or clustering
  - **Binary Decision**: (age < 40) or (age ≥ 40)
    - consider all possible splits and find the best cut (i.e., age “40”)
    - in general compute intensive (later in this class)

# Example: Splitting a Continuous Attribute



(i) Binary split



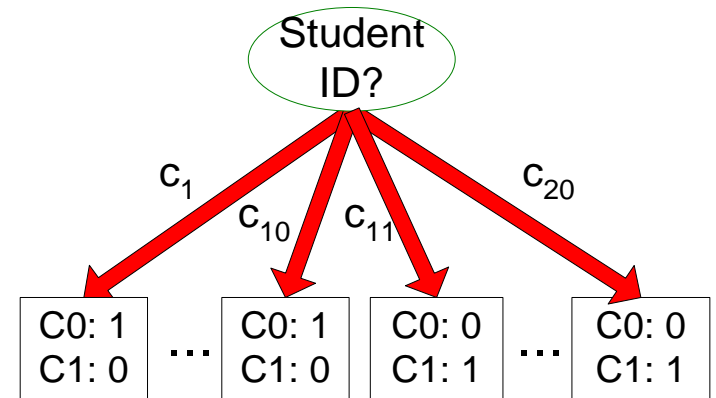
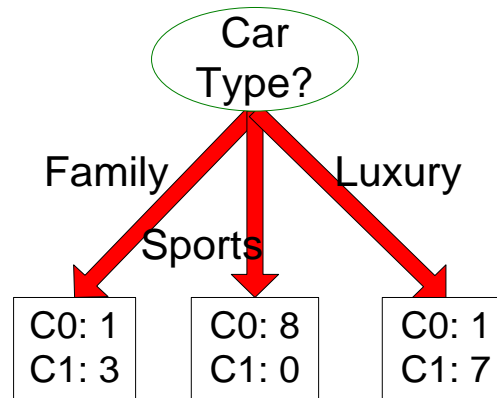
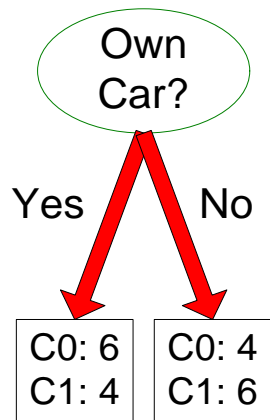
(ii) Multi-way split



# Determining the Best Split (1/2)

StudentID	OwnCar	CarType	Grade
-----------	--------	---------	-------

Before Splitting: 10 records of class 0 (grade good)(C0)  
10 records of class 1 (grade poor)(C1)



Which test condition is the best?

## Determining the Best Split (2/2)

- Nodes with **homogeneous/pure** class distribution are preferred
  - \* “homogeneous/pure” means **“all elements are of the same type(class)”**
  - (e.g.) all records for the node have the same value for Grade.
- Need a measure of node “impurity”

C0: 5
C1: 5

non-homogeneous,  
high degree of impurity  
(like coin toss)

C0: 9
C1: 1

homogeneous,  
low degree of impurity  
prediction possible



---

# End of Quick Review



# Measures (Metrics) of Node Impurity

---

- Entropy
  - (studied in Data Science: will be skipped)
  - Information Gain, Gain Ratio
- Gini Index
- Classification Error
- Standard Deviation (Variance Reduction)



# Roadmap: Measures of Impurity

---

- Gini index
- Classification error



# Gini Index

---

- Gini index (Gini coefficient, or Gini ratio)
- Developed by the Italian statistician and sociologist Corrado Gini in 1912
- Used in economics as a measurement of inequality in income or wealth distribution of a nation's residents
- Used in CART, SLIQ, SPRINT decision trees
- Default in Scikit-learn implementation of Random Forest, AdaBoost, and Gradient Boosting

# Computing the Gini Index

- Gini Index for a given node  $t$  :

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

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

- Maximum ( $1 - 1/n_c$ ) when records are **equally distributed** among all classes, implying least interesting information ( $n_c$  is the number of classes)
- Minimum (0.0) when all records belong to **one class**, implying most interesting information

C1	<b>0</b>	C1	<b>1</b>	C1	<b>2</b>	C1	<b>3</b>
C2	<b>6</b>	C2	<b>5</b>	C2	<b>4</b>	C2	<b>3</b>
<b>Gini=0.000</b>		<b>Gini=0.278</b>		<b>Gini=0.444</b>		<b>Gini=0.500</b>	

# Illustration: Computing the Gini Index

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

C1	<b>0</b>
C2	<b>6</b>

$$P(C1) = 0/6 = 0 \quad P(C2) = 6/6 = 1$$

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

C1	<b>1</b>
C2	<b>5</b>

$$P(C1) = 1/6 \quad P(C2) = 5/6$$

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

C1	<b>2</b>
C2	<b>4</b>

$$P(C1) = 2/6 \quad P(C2) = 4/6$$

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





# In-Class Exercise

---

- Compute the Gini index (using the formula)
  - when there are 2 records, and both have the same class (C1).



# Answer

---

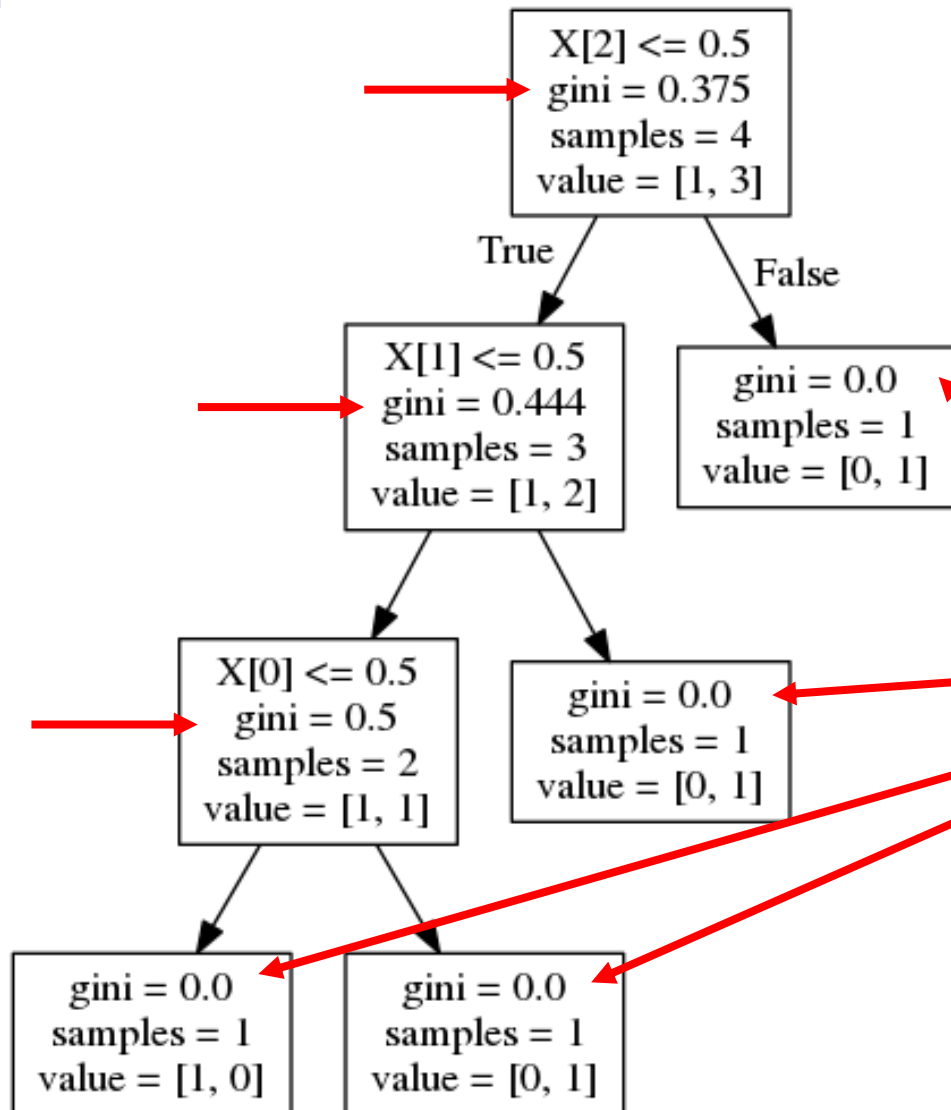
$$GINI(t) = 1 - \sum_j [p(j|t)]^2$$

$$P(C1) = 2/2 = 1$$

$$Gini = 1 - P(C1)^2 = 1 - 1 = 0$$

- Note: When there is only one class in a decision tree node, the Gini index is the minimum.
- A low Gini index is more desirable.

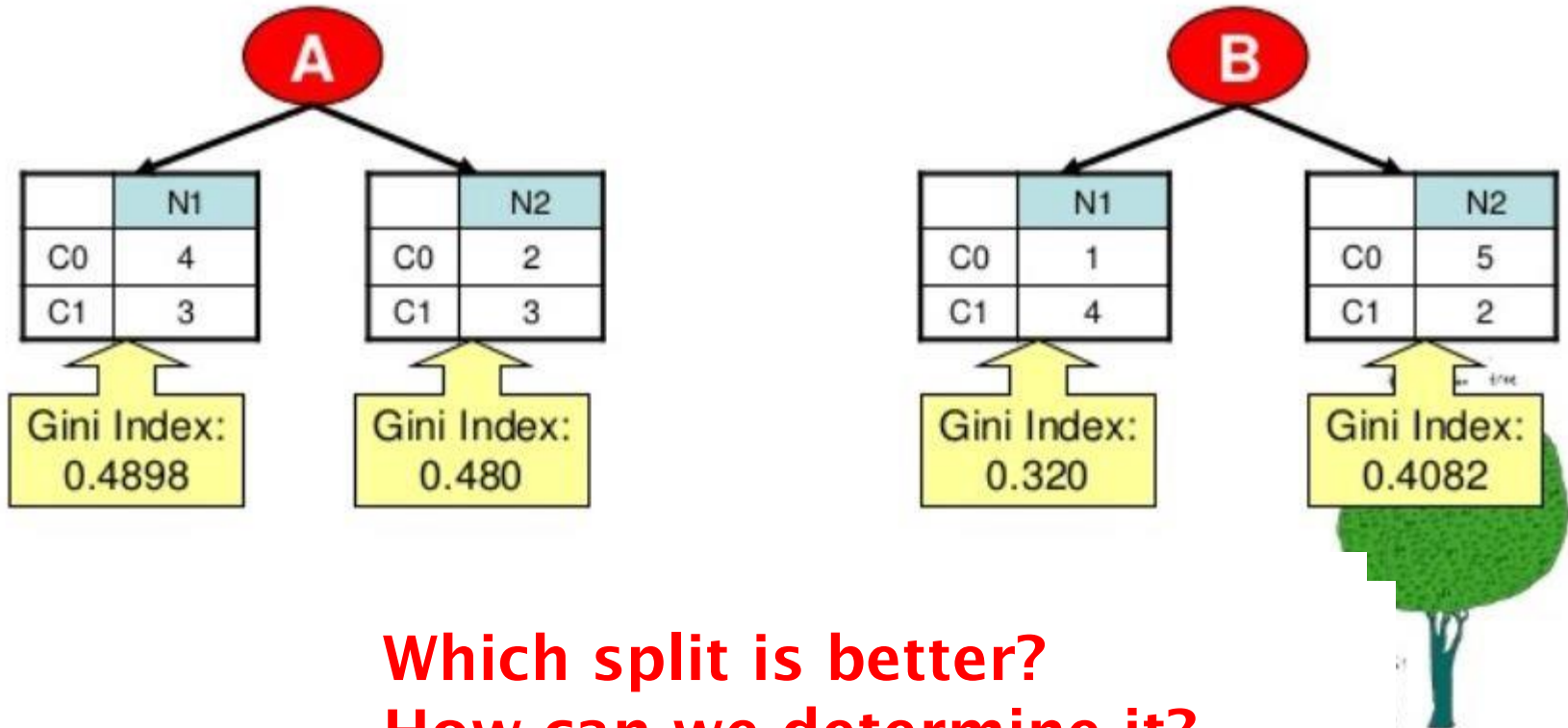
# Graphical Representation of a Gini Decision Tree



Note:  
The nodes with  
gini=0.0 are leaf  
nodes

# We can compute Gini index. Is that enough (Are we done)?

Suppose there are two ways (A and B) to split the data into smaller subset.





# Information Gain vs. Gini Gain

---

- When using entropy as measure of node impurity, information gain has to be computed.
- Similarly, when using Gini index as measure of node impurity, Gini gain has to be computed.

# Splitting Criterion: Gini Gain

(Below,  $N_{xx}$  is a distribution,  $M_x$  is Gini index)

Before  
Splitting:

C0	<b>N00</b>
C1	<b>N01</b>

→  $M_0$

A?

Yes

No

Node N1

Node N2

C0	<b>N10</b>
C1	<b>N11</b>

C0	<b>N20</b>
C1	<b>N21</b>

$M_1$

$M_2$

$M_{12}$

B?

Yes

No

Node N3

Node N4

C0	<b>N30</b>
C1	<b>N31</b>

C0	<b>N40</b>
C1	<b>N41</b>

$M_3$

$M_4$

$M_{34}$

Gain =  $(M_0 - M_{12})$  vs.  $(M_0 - M_{34})$



# Selecting the Best Node Split

- For each split (e.g., "A", "B" in the previous page)
  - Compute the Gini index for the node before split ("parent":  $M_0$  in the previous page).
  - Split the node into  $k$  partitions ("children") ( $k=2$  for split "A" and  $k=2$  for split "B" in the previous page)
  - Compute the Gini for each partition ("child":  $M_1$  and  $M_2$  for "A";  $M_3$  and  $M_4$  for "B")
  - Compute the weighted Gini for the  $k$  partitions ( $M_{12}$ ,  $M_{34}$ )
  - Compute the **Gini Gain**  
(Gini for "parent" – weighted Gini for the  $k$  partitions)
- Select the split that gives the maximum Gini Gain



# Weighted Gini of the Partitions

- When a node  $p$  (parent) is split into  $k$  partitions (children), the weighted Gini of the  $k$  children 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 parent node  $p$ .

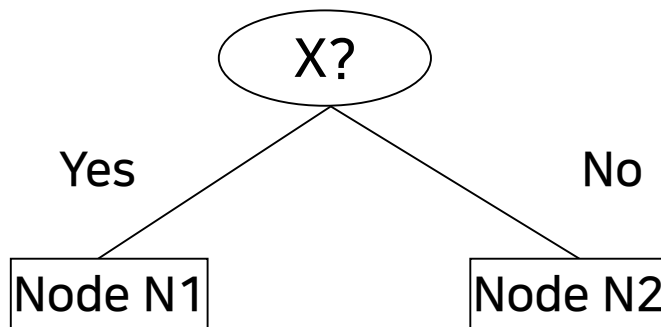


# Example: Computing the Gini Gain (1/2)

- Compute the Gini of a (parent) node

	<b>Parent</b>
C1	<b>6</b>
C2	<b>6</b>
<b>Gini = 0.500</b>	

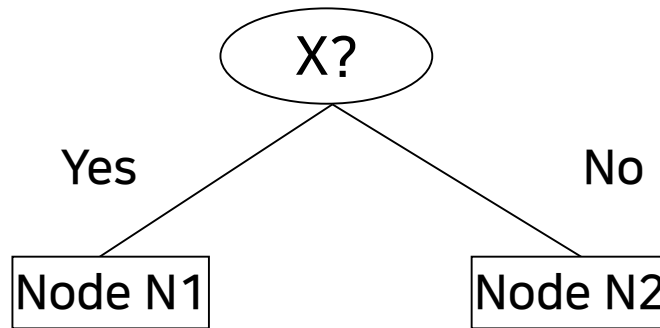
- Split the node into two partitions



	<b>N1</b>	<b>N2</b>
C1	<b>5</b>	<b>1</b>
C2	<b>2</b>	<b>4</b>

## Example: Computing the Gini Gain (2/2)

- Compute the Gini for each partition



	<b>N1</b>	<b>N2</b>
C1	<b>5</b>	<b>1</b>
C2	<b>2</b>	<b>4</b>

$$\text{Gini}(N1) = 1 - (5/7)^2 - (2/7)^2 = 0.41$$

$$\text{Gini}(N2) = 1 - (1/5)^2 - (4/5)^2 = 0.32$$

- Compute the weighted Gini for the 2 partitions

$$\text{Gini}(\text{Children}) = 7/12 * 0.41 + 5/12 * 0.32 = 0.372$$

- Gini Gain =  $0.5 - 0.372 = 0.128$



# GINI Walkthrough Example

- <https://sefiks.com/2018/08/27/a-step-by-step-cart-decision-tree-example/>
- Dataset: 14 samples
- 4 predictor attributes (weather conditions)
  - outlook, temperature, humidity, wind
- target (class): (yes/no) decision to play golf



# Dataset

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No



# Computing Steps (1/29)

## Determine the Root Node

- Let us consider all 4 attributes, one by one.
- First, Outlook
  - **outlook** is a nominal attribute. It can be sunny, overcast or rain.
  - The following **count matrix** summarizes the (yes/no) decision for each of the **outlook** values.

Outlook	Yes	No	Number of records
Sunny	2	3	5
Overcast	4	0	4
Rain	3	2	5



## Computing Steps (2/29)

- Gini indexes for **outlook**

$$\text{Gini}(\text{Outlook}=\text{Sunny}) = 1 - (2/5)^2 - (3/5)^2 = 1 - 0.16 - 0.36 = 0.48$$

$$\text{Gini}(\text{Outlook}=\text{Overcast}) = 1 - (4/4)^2 - (0/4)^2 = 0$$

$$\text{Gini}(\text{Outlook}=\text{Rain}) = 1 - (3/5)^2 - (2/5)^2 = 1 - 0.36 - 0.16 = 0.48$$

Now calculate the weighted sum of the Gini indexes for the **outlook** attribute.

$$\begin{aligned}\text{Gini}(\text{Outlook}) &= (5/14) \times 0.48 + (4/14) \times 0 + (5/14) \times 0.48 \\ &= 0.171 + 0 + 0.171 = 0.342\end{aligned}$$



## Computing Steps (3/29)

- Next, Temperature
  - **temperature** is a nominal attribute. It can be hot, cool or mild.
  - The following, count matrix, summarizes the (yes/no) decision for each of the **temperature** values.

Temperature	Yes	No	Number of records
Hot	2	2	4
Cool	3	1	4
Mild	4	2	6



# Computing Steps (4/29)

- Gini indexes for **temperature**

$$\text{Gini}(\text{Temp}=\text{Hot}) = 1 - (2/4)^2 - (2/4)^2 = 0.5$$

$$\text{Gini}(\text{Temp}=\text{Cool}) = 1 - (3/4)^2 - (1/4)^2 = 1 - 0.5625 - 0.0625 = 0.375$$

$$\text{Gini}(\text{Temp}=\text{Mild}) = 1 - (4/6)^2 - (2/6)^2 = 1 - 0.444 - 0.111 = 0.445$$

Now calculate the weighted sum of the Gini index for the **temperature** attribute

$$\begin{aligned}\text{Gini}(\text{Temp}) &= (4/14) \times 0.5 + (4/14) \times 0.375 + (6/14) \times 0.445 \\ &= 0.142 + 0.107 + 0.190 = 0.439\end{aligned}$$





# Computing Steps (5/29)

- Next, Humidity
  - **humidity** is a binary attribute. It can be high or normal.
  - The following summarizes the (yes/no) decision for each of the **humidity** values.

Humidity	Yes	No	Number of records
High	3	4	7
Normal	6	1	7



# Computing Steps (6/29)

- Gini indexes for **humidity**

$$\text{Gini}(\text{Humidity}=\text{High}) = 1 - (3/7)^2 - (4/7)^2 = 1 - 0.183 - 0.326 = 0.489$$

$$\text{Gini}(\text{Humidity}=\text{Normal}) = 1 - (6/7)^2 - (1/7)^2 = 1 - 0.734 - 0.02 = 0.244$$

Now, calculate the weighted sum for the **humidity** attribute.

$$\text{Gini}(\text{Humidity}) = (7/14) \times 0.489 + (7/14) \times 0.244 = 0.367$$



# Computing Steps (7/29)

- Next, Wind
  - **wind** is a binary attribute. It can be strong or weak.
  - The following summarizes the (yes/no) decision for each of the **wind** values.

Wind	Yes	No	Number of records
Weak	6	2	8
Strong	3	3	6



# Computing Steps (8/29)

- Gini indexes for **wind**

$$\text{Gini}(\text{Wind}=\text{Weak}) = 1 - (6/8)^2 - (2/8)^2 = 1 - 0.5625 - 0.0625 = 0.375$$

$$\text{Gini}(\text{Wind}=\text{Strong}) = 1 - (3/6)^2 - (3/6)^2 = 1 - 0.25 - 0.25 = 0.5$$

Now, calculate the weighted sum of the Gini indexes for the **wind** attribute.

$$\text{Gini}(\text{Wind}) = (8/14) \times 0.375 + (6/14) \times 0.5 = 0.428$$



# Computing Steps (8/29)

- Gini indexes for **wind**

$$\text{Gini}(\text{Wind}=\text{Weak}) = 1 - (6/8)^2 - (2/8)^2 = 1 - 0.5625 - 0.0625 = 0.375$$

$$\text{Gini}(\text{Wind}=\text{Strong}) = 1 - (3/6)^2 - (3/6)^2 = 1 - 0.25 - 0.25 = 0.5$$

Now, calculate the weighted sum of the Gini indexes for the **wind** attribute.

$$\text{Gini}(\text{Wind}) = (8/14) \times 0.375 + (6/14) \times 0.5 = 0.428$$



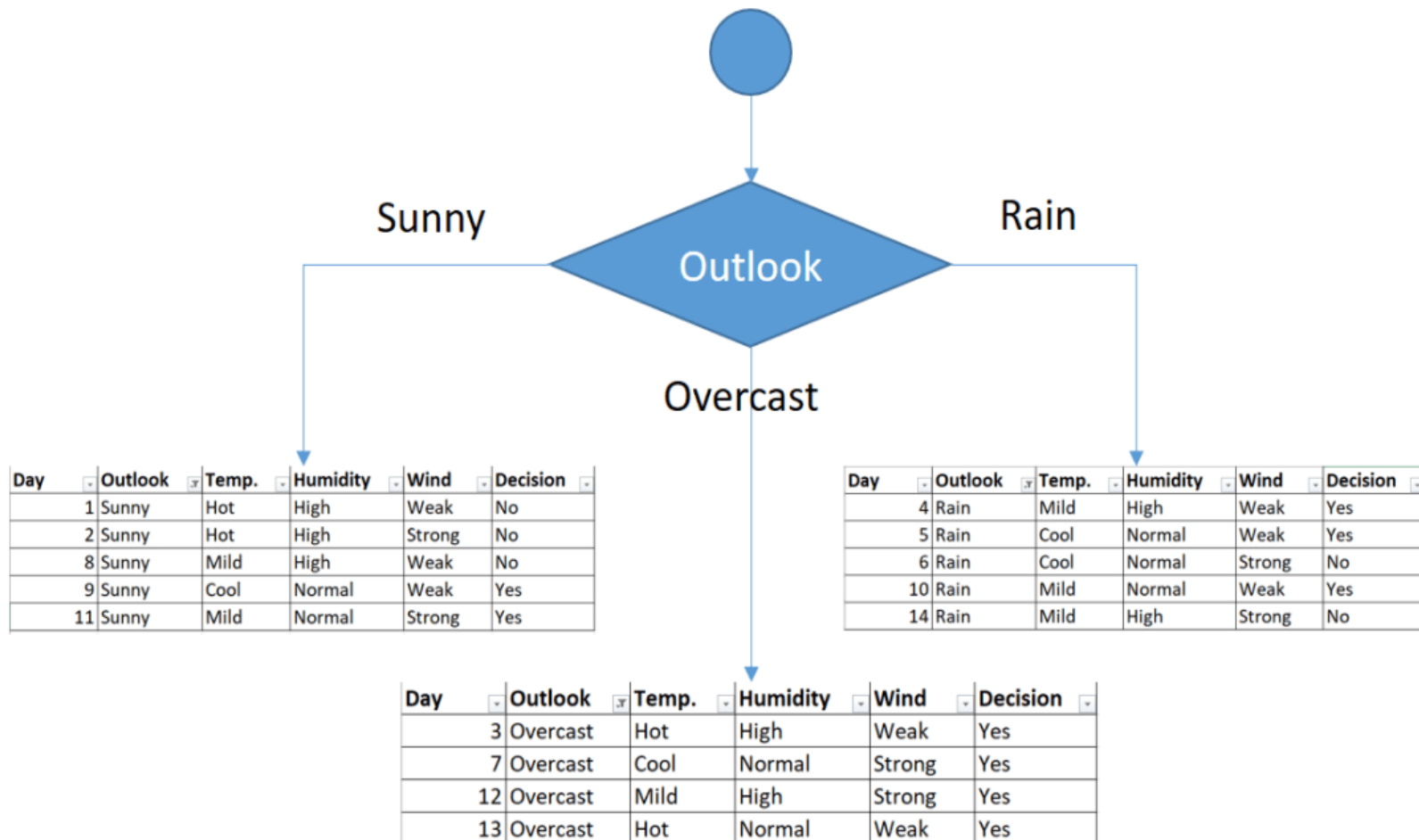
## Computing Steps (9/29)

- Now, select the attribute that results in the maximum Gini gain as the root node
- Select **outlook**

attribute	Weighted Gini index
Outlook	0.342
Temperature	0.439
Humidity	0.367
Wind	0.428

# Computing Steps (10/29)

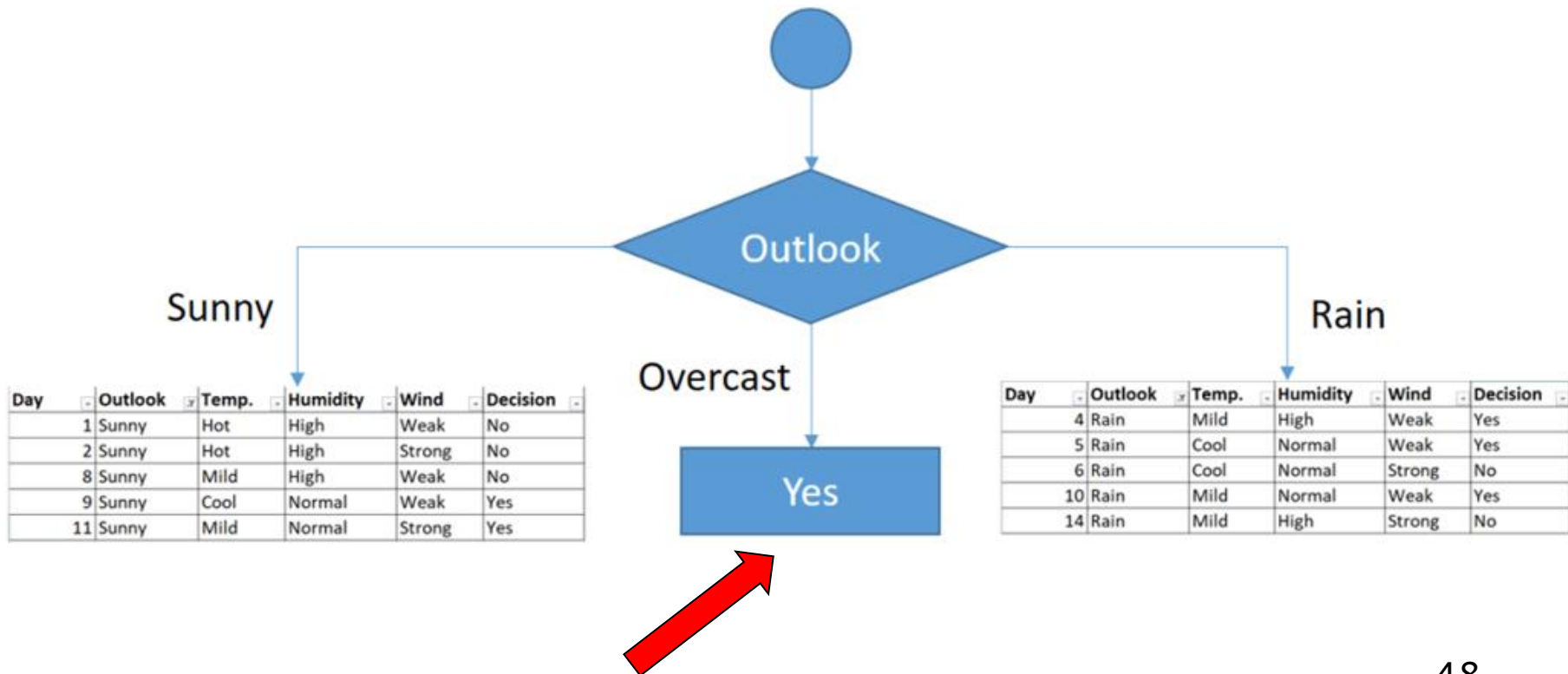
- The resulting intermediate decision tree



# Computing Steps (11/29)

## The Next Level

- The sub-dataset in the overcast leaf has only “yes” decisions. This means that **overcast** leaf is done.
- The tree now becomes as follows.







# Computing Steps (12/29)

- Look at the sub-dataset for the **sunny outlook**.
- We need to find the Gini index scores for the **temperature, humidity and wind** attributes.

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes



# Computing Steps (13/29)

- temperature for the sunny outlook

Temperature	Yes	No	Number of records
Hot	0	2	2
Cool	1	0	1
Mild	1	1	2



## Computing Steps (14/29)

- Compute Gini index for **sunny** and **temperature**

$$\text{Gini}(\text{Outlook}=\text{Sunny and Temp.}=\text{Hot}) = 1 - (0/2)^2 - (2/2)^2 = 0$$

$$\text{Gini}(\text{Outlook}=\text{Sunny and Temp.}=\text{Cool}) = 1 - (1/1)^2 - (0/1)^2 = 0$$

$$\begin{aligned}\text{Gini}(\text{Outlook}=\text{Sunny and Temp.}=\text{Mild}) &= 1 - (1/2)^2 - (1/2)^2 \\ &= 1 - 0.25 - 0.25 = 0.5\end{aligned}$$

$$\text{Gini}(\text{Outlook}=\text{Sunny and Temp.})$$

$$= (2/5) \times 0 + (1/5) \times 0 + (2/5) \times 0.5 = 0.2$$



# Computing Steps (15/29)

- **humidity** for the **sunny outlook**

Humidity	Yes	No	Number of records
High	0	3	3
Normal	2	0	2



## Computing Steps (16/29)

---

- Gini index for **sunny** and **humidity**

$$\text{Gini}(\text{Outlook}=\text{Sunny and Humidity}=\text{High}) = 1 - (0/3)^2 - (3/3)^2 = 0$$

$$\text{Gini}(\text{Outlook}=\text{Sunny and Humidity}=\text{Normal}) = 1 - (2/2)^2 - (0/2)^2 = 0$$

$$\text{Gini}(\text{Outlook}=\text{Sunny and Humidity}) = (3/5) \times 0 + (2/5) \times 0 = 0$$



# Computing Steps (17/29)

- wind for the sunny outlook

Wind	Yes	No	Number of records
Weak	1	2	3
Strong	1	1	2



## Computing Steps (18/29)

---

- Gini index for **sunny** and **wind**

$$\text{Gini}(\text{Outlook}=\text{Sunny and Wind}=\text{Weak}) = 1 - (1/3)^2 - (2/3)^2 = 0.266$$

$$\text{Gini}(\text{Outlook}=\text{Sunny and Wind}=\text{Strong}) = 1 - (1/2)^2 - (1/2)^2 = 0.2$$

$$\text{Gini}(\text{Outlook}=\text{Sunny and Wind}) = (3/5) \times 0.266 + (2/5) \times 0.2 = 0.466$$



## Computing Steps (19/29)

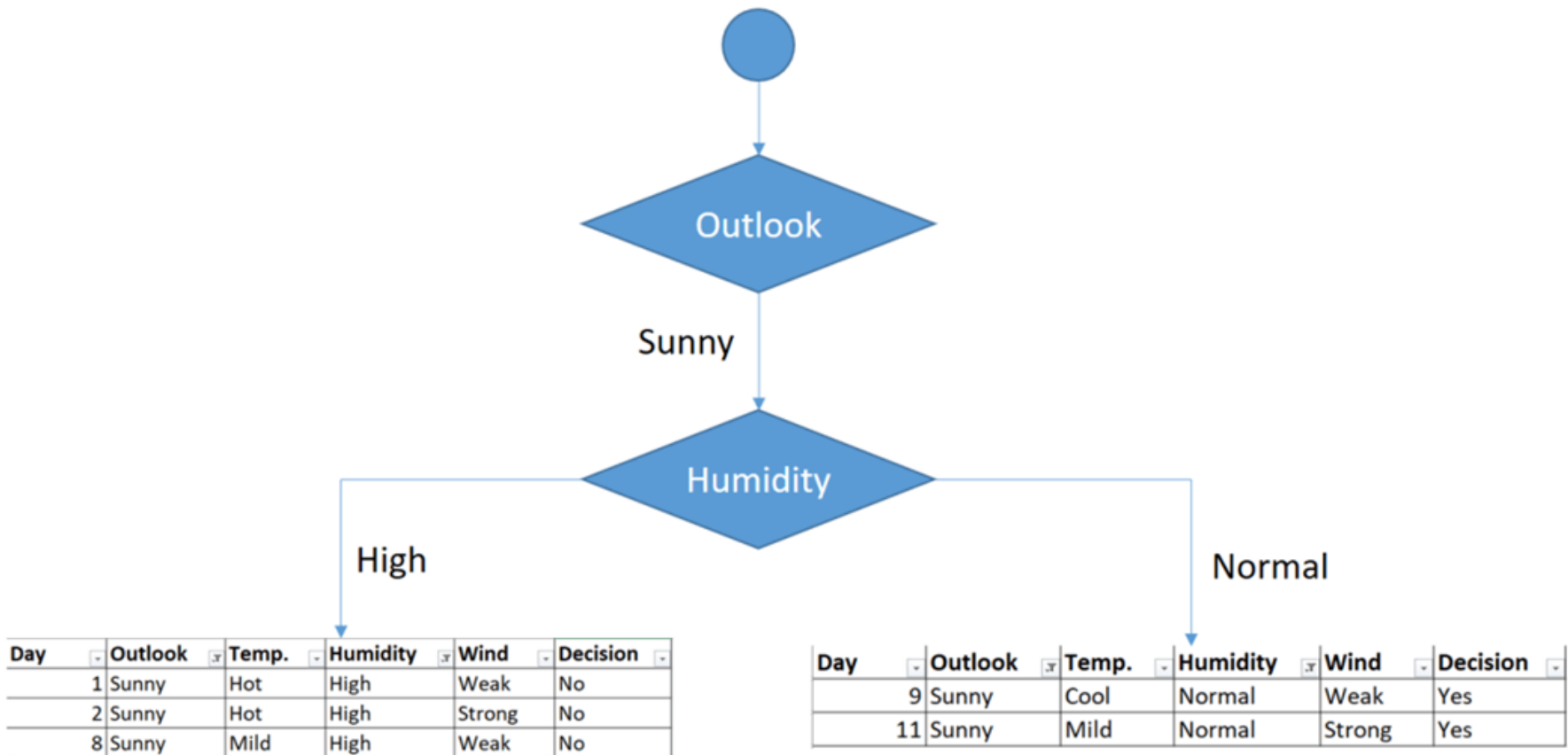
- **humidity** is the best attribute for the **sunny** outlook

attribute	Weighted Gini index
Temperature	0.2
Humidity	0
Wind	0.466



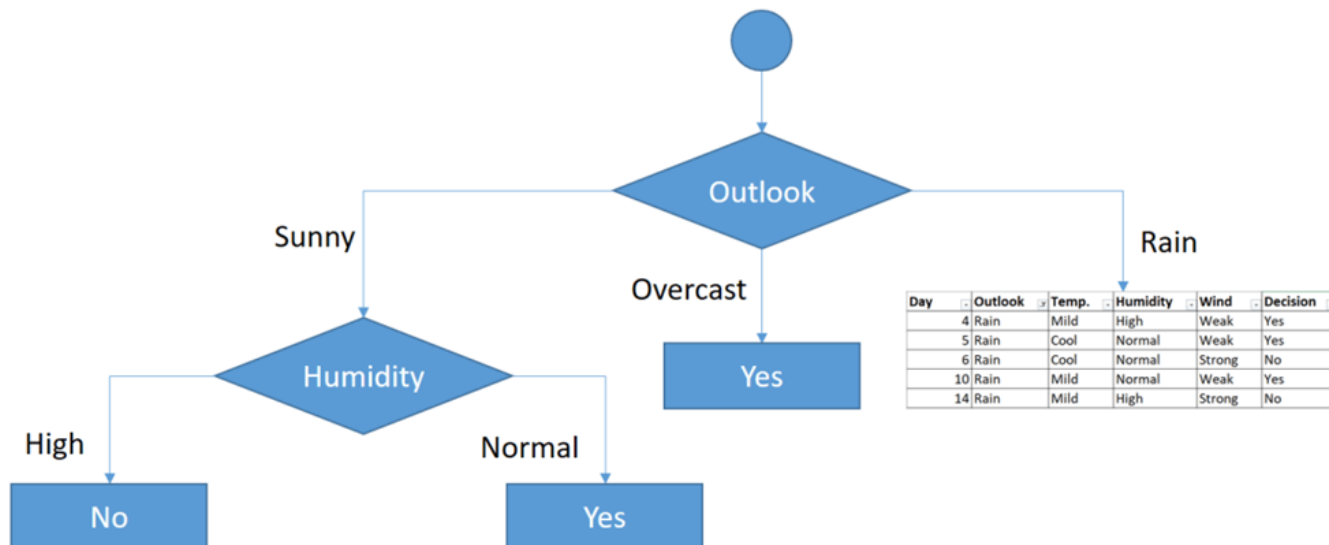
# Computing Steps (20/29)

- The resulting subtree is as follows.



# Computing Steps (21/29)

- Look at the sub-datasets for high and normal **humidity**
- The decision is always “no” for high **humidity** and **sunny outlook**.
- Similarly, the decision is always “yes” for normal **humidity** and **sunny outlook**.
- The **sunny outlook** branch is done.





## Computing Steps (22/29)

- Now look at the **rain outlook**.
- We calculate Gini index scores for **temperature**, **humidity** and **wind** attributes when **outlook** is **rain**.

Day	Outlook	Temp.	Humidity	Wind	Decision
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
10	Rain	Mild	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No



## Computing Steps (23/29)

- temperature for the rain outlook

Temperature	Yes	No	Number of records
Cool	1	1	2
Mild	2	1	3

$$\text{Gini}(\text{Outlook}=\text{Rain and Temp.}=\text{Cool}) = 1 - (1/2)^2 - (1/2)^2 = 0.5$$

$$\text{Gini}(\text{Outlook}=\text{Rain and Temp.}=\text{Mild}) = 1 - (2/3)^2 - (1/3)^2 = 0.444$$

$$\text{Gini}(\text{Outlook}=\text{Rain and Temp.}) = (2/5) \times 0.5 + (3/5) \times 0.444 = 0.466$$



## Computing Steps (24/29)

- **humidity** for the **rain outlook**

Humidity	Yes	No	Number of records
High	1	1	2
Normal	2	1	3

$$\text{Gini}(\text{Outlook}=\text{Rain and Humidity}=\text{High}) = 1 - (1/2)^2 - (1/2)^2 = 0.5$$

$$\text{Gini}(\text{Outlook}=\text{Rain and Humidity}=\text{Normal}) = 1 - (2/3)^2 - (1/3)^2 = 0.444$$

$$\text{Gini}(\text{Outlook}=\text{Rain and Humidity}) = (2/5) \times 0.5 + (3/5) \times 0.444 = 0.466$$



## Computing Steps (25/29)

- wind for the rain outlook

Wind	Yes	No	Number of records
Weak	3	0	3
Strong	0	2	2

$$\text{Gini}(\text{Outlook}=\text{Rain and Wind}=\text{Weak}) = 1 - (3/3)^2 - (0/3)^2 = 0$$

$$\text{Gini}(\text{Outlook}=\text{Rain and Wind}=\text{Strong}) = 1 - (0/2)^2 - (2/2)^2 = 0$$

$$\text{Gini}(\text{Outlook}=\text{Rain and Wind}) = (3/5) \times 0 + (2/5) \times 0 = 0$$



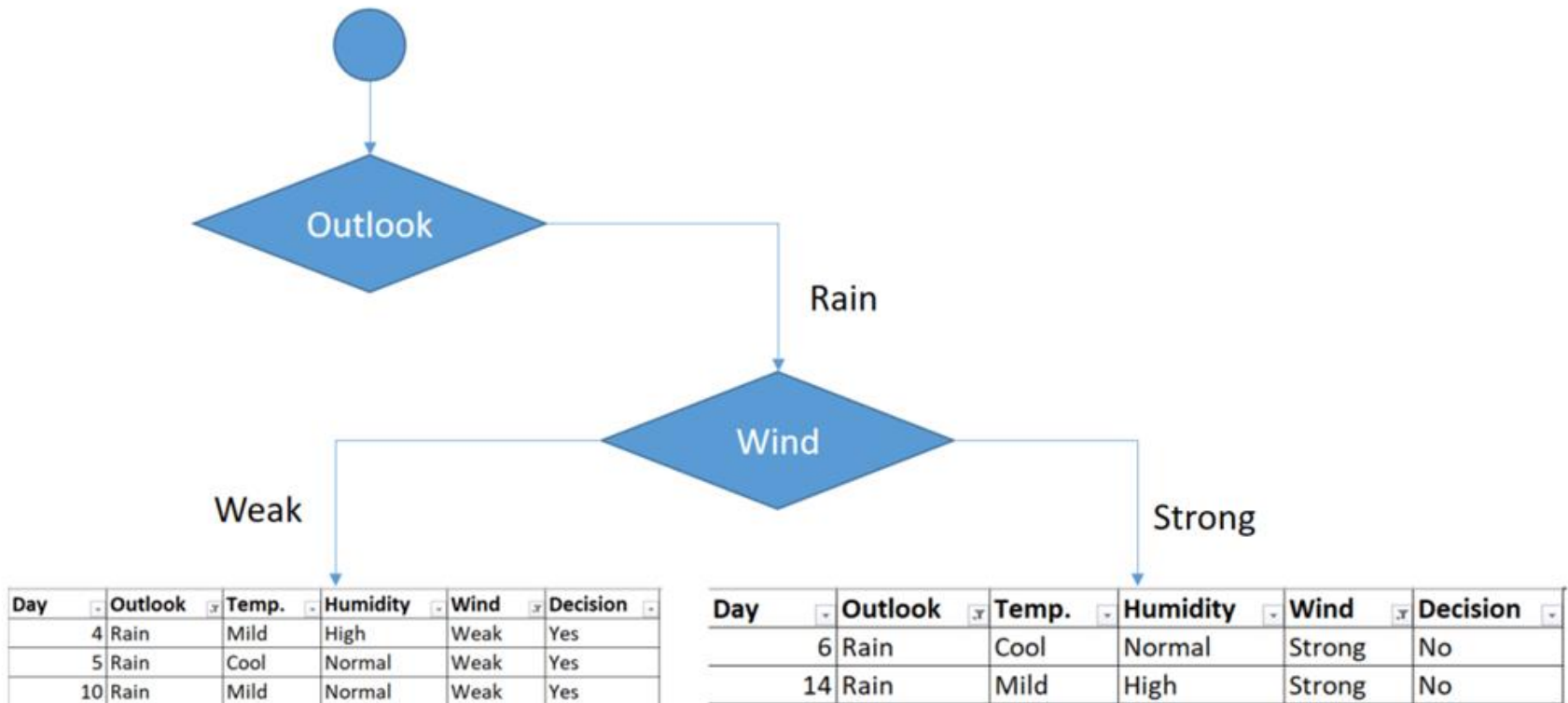
## Computing Steps (26/29)

- **wind** is the best attribute of the **rain outlook**

attribute	Weighted Gini index
Temperature	0.466
Humidity	0.466
Wind	0

# Computing Steps (27/29)

- The resulting subtree is as follows.







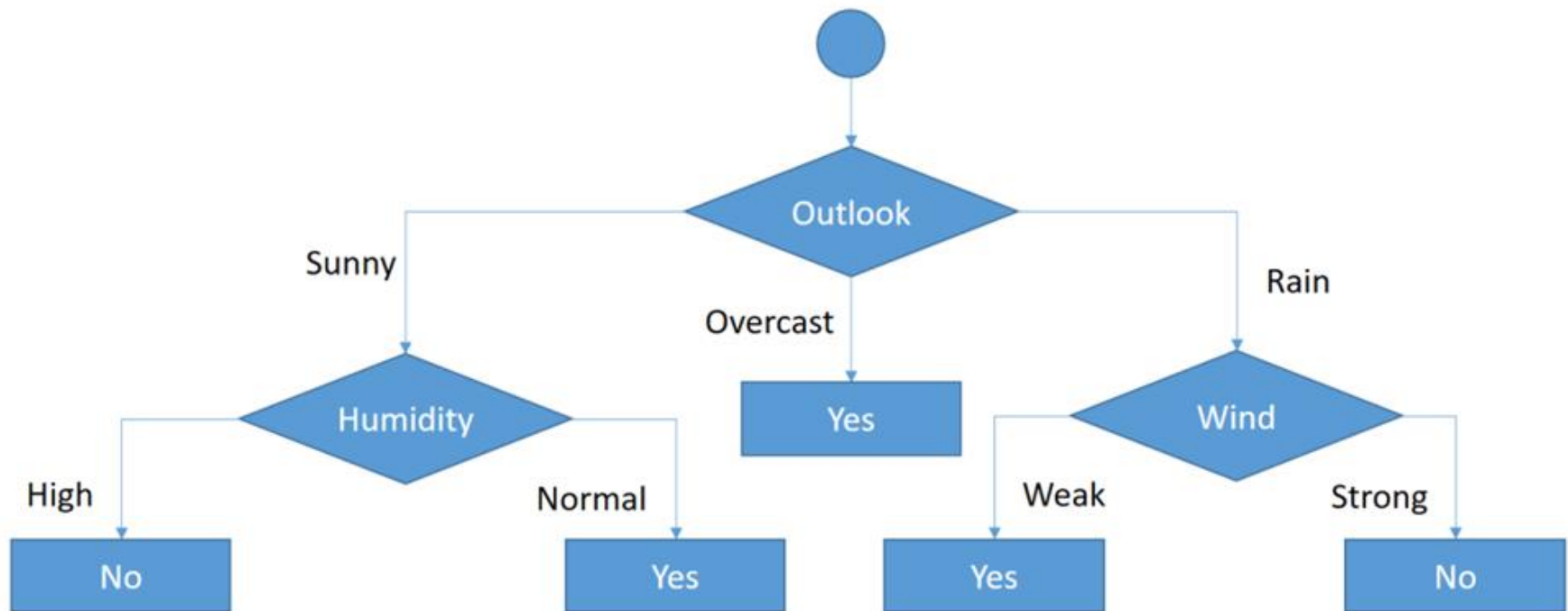
## Computing Steps (28/29)

---

- Look at the sub-datasets for the weak and strong **wind** and **rain outlook**.
- The decision is always “yes” when **wind** is weak.
- The decision is always “no” if **wind** is strong.
- This means that the **rain wind** branch is done.

# Computing Steps (29/29)

- Final form of the decision tree.





## Homework 1

- By hand computation, build a decision tree for the following dataset, using Gini index; **and submit in a WORD file. (\* Do NOT stop at the root node. \*)**
- A publisher wants to know if a new novel manuscript (by two authors) will be a **hit** based on a past history of the two authors.

Novelist	Novel Genre	Hit (Yes/No)
Isaac Asimov	science fiction	yes
Isaac Asimov	fantasy	yes
Isaac Asimov	adventure	no
Isaac Asimov	science fiction	yes
Tom Clancy	science fiction	no
Tom Clancy	fantasy	no
Tom Clancy	adventure	yes

# Computing the Gini Index for Categorical Attributes (Summarized)

- As we have seen, for each distinct value, gather count 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)

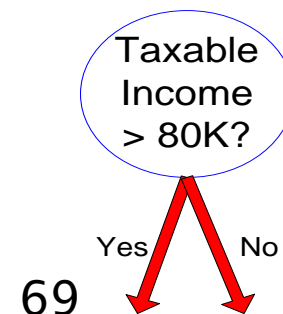
	CarType	
	{Sports, Luxury}	{Family}
C1	3	1
C2	2	4
Gini	0.400	

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

# Selecting the Best Splitting Value for a Continuous Attribute: Naïve Method

- Use binary decisions based on one splitting value  $v$ 
  - number of possible splitting values  
= number of distinct values  
(possibly very many)
- Each splitting value  $v$  has a count matrix associated with it
  - Class count in each of the partitions,  $A < v$  and  $A \geq v$
- Simple method to choose best  $v$ 
  - For each  $v$ , scan the dataset twice to gather count matrix and compute its Gini index
  - Computationally Inefficient! Repetition of work.

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



# Selecting the Best Splitting Value for a Continuous Attribute: Efficient Method

- For each attribute
  - Sort the attribute values.
  - Sequentially scan these values, each time updating the count matrix and computing Gini index.
  - Choose split positions as intermediate between two neighboring attribute values, and compute Gini index for each.  
(e.g., 72 for 70 and 75 and 172 for 125 and 220).
  - Choose the split position that has the least Gini index (97)

Cheat	No		No		No		Yes		Yes		Yes		No		No		No		No			
S →	Taxable Income																					
	60		70		75		85		90		95		100		120		125		220			
	55		65		72		80		87		92		97		110		122		172		230	
IS →	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>
	Yes	0	3	0	3	0	3	0	3	1	2	2	1	3	0	3	0	3	0	3	0	3
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		<u>0.300</u>		0.343		0.375		0.400		0.420	



## Homework 2

---

- Write an algorithm (pseudo code) for the efficient method. How many times is the dataset scanned (after sorting)?
- Using the naïve method, find the best splitting value for Taxable Income, using the example data ([on page 69](#)). (\* You must show all your work.) How many times is the dataset scanned?
- What is the computational complexity of the naïve method and efficient method (excluding sorting).
  - Hint: describe in terms of Big Oh.



# Roadmap: Measures of Impurity

---

- Gini index
- Classification error





## Classification (Miscalculation) Error

- Similar to the Gini index, but simpler
- Classification error at a node  $t$  :

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

( $p(i | t)$ ) is the relative frequency of class  $i$  at node  $t$ )

- Measures classification error made by a node.
  - Maximum ( $1 - 1/n_c$ ) when records are equally distributed among all classes, implying least interesting information ( $n_c$  is the number of classes)
  - Minimum (0.0) when all records belong to one class, implying most interesting information

# Illustration: Computing the Classification Error

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

C1	<b>0</b>
C2	<b>6</b>

$$P(C1) = 0/6 = 0 \quad P(C2) = 6/6 = 1$$

$$Error = 1 - \max(0, 1) = 1 - 1 = 0$$

C1	<b>1</b>
C2	<b>5</b>

$$P(C1) = 1/6 \quad P(C2) = 5/6$$

$$Error = 1 - \max(1/6, 5/6) = 1 - 5/6 = 1/6$$

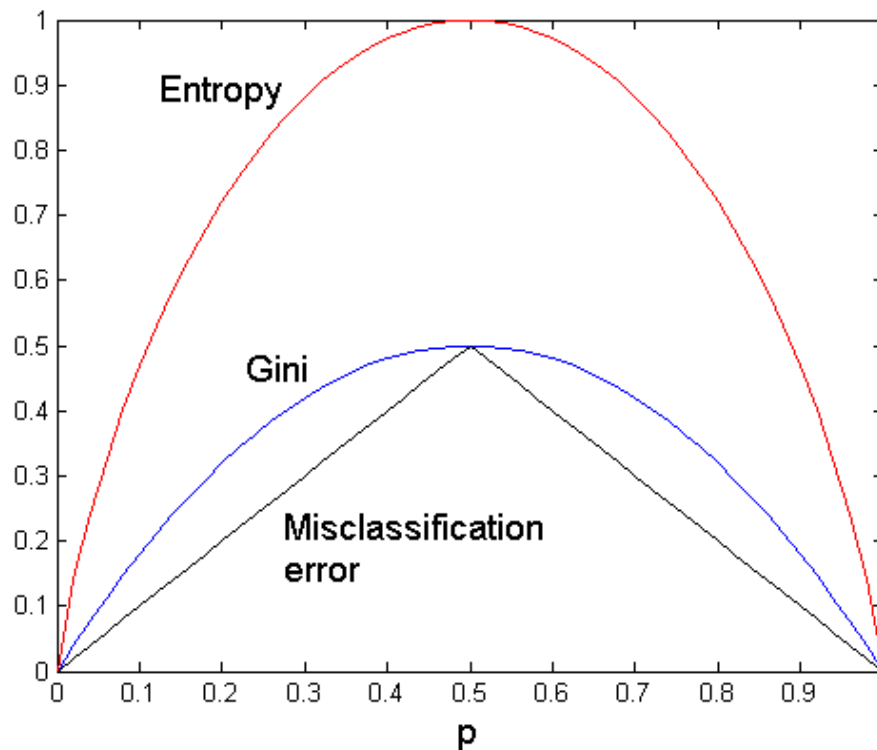
C1	<b>2</b>
C2	<b>4</b>

$$P(C1) = 2/6 \quad P(C2) = 4/6$$

$$Error = 1 - \max(2/6, 4/6) = 1 - 4/6 = 1/3$$

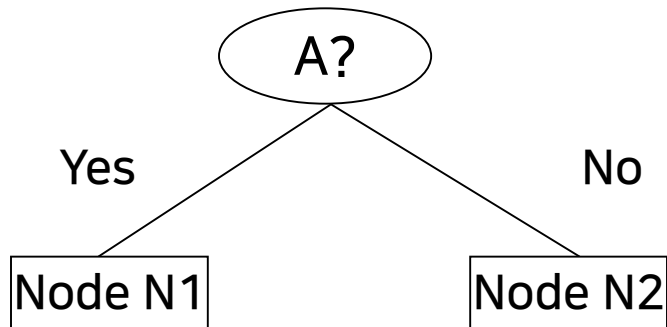
# Comparing the Splitting Criteria

- (for a 2-class problem)
- note: Impurity for entropy is higher than for Gini and classification error. This is why in theory Gini index is better than entropy.



# Classification Error vs Gini

## In-Class Exercise: Compute Classification Errors corresponding to Gini Index



	Parent
C1	7
C2	3
<b>Gini = 0.42</b>	

**Gini(N1)**

$$= 1 - (3/3)^2 - (0/3)^2 \\ = 0$$

	N1	N2
C1	3	4
C2	0	3

**Gini(N2)**

$$= 1 - (4/7)^2 - (3/7)^2 \\ = 0.489$$

**Gini(Children)**

$$= 3/10 * 0 \\ + 7/10 * 0.489 \\ = 0.342$$



# Advantages of Decision Trees

---

- Inexpensive to construct (for small datasets)
- Fast at classifying unknown records
- Accuracy is comparable to other classification techniques (for simple datasets)
- Easy to interpret for small-sized trees
- Can handle both continuous and categorical variables
- Can be handle both classification and regression tasks.
- Default for ensemble learning models (e.g., random forest, AdaBoost, Gradient Boosting)



# Weaknesses of Decision Trees

- Error-prone with too many classes (targets)
  - number of training examples becomes smaller quickly in a tree with many levels/branches.
- Expensive to train
  - sorting, combination of attributes, etc.
- Greedy algorithm
  - no global optimization (makes best split decision at each level of the tree, and drop all others)



# Roadmap

---

- Decision Trees
- **Decision Tree Regression**
- Decision Tree Pruning



# Classification vs. Regression

---

- Many machine learning models can be used for both classification and regression tasks
  - (e.g.) decision tree, Support Vector Machine, neural networks
- One key difference between classification and regression is the type of target attribute.
  - Classification: predicts a categorical attribute
  - Regression: predicts a continuous attribute





# Acknowledgments

---

- [https://www.saedsayad.com/decision\\_tree\\_reg.htm](https://www.saedsayad.com/decision_tree_reg.htm)
- <https://acadgild.com/blog/using-decision-trees-for-regression-problems>



## ID3 and C4.5

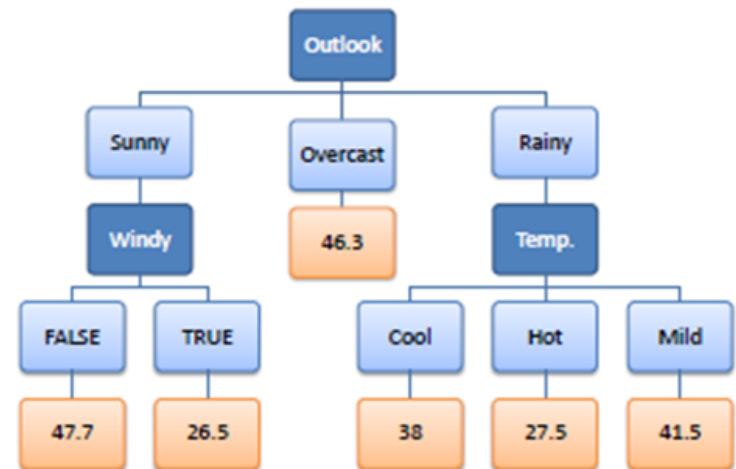
---

- Invented by J.Ross Quinlan
- Can be used for regression by replacing Information Gain with Standard Deviation Reduction (or Variance Reduction)

# Walkthrough Example: Building a Decision Tree for Regression

- Number of hours played tennis, based on the outlook, temperature, humidity and wind of the weather.
- Numerical (continuous valued) target attribute

Predictors				Target
Outlook	Temp.	Humidity	Windy	Hours Played
Rainy	Hot	High	False	26
Rainy	Hot	High	True	30
Overcast	Hot	High	False	48
Sunny	Mild	High	False	46
Sunny	Cool	Normal	False	62
Sunny	Cool	Normal	True	23
Overcast	Cool	Normal	True	43
Rainy	Mild	High	False	36
Rainy	Cool	Normal	False	38
Sunny	Mild	Normal	False	48
Rainy	Mild	Normal	True	48
Overcast	Mild	High	True	62
Overcast	Hot	Normal	False	44
Sunny	Mild	High	True	30





## Computing Steps (1/12)

---

- Select the root node predictor (attribute) from the 4 predictors
- To do it, we need to compute the **standard deviations** for each attribute **(not Gini index)**, and select the one that gives the largest **standard deviation reduction (SDR)** **(not Gini Gain)**.


## Computing Steps (2/12)

- Compute the standard deviation for the **target attribute**

Hours Played
25
30
46
45
52
23
43
35
38
46
48
52
44
30

$$\text{Count} = n = 14$$

$$\text{Average} = \bar{x} = \frac{\sum x}{n} = 39.8$$


$$\text{Standard Deviation} = S = \sqrt{\frac{\sum (x - \bar{x})^2}{n}} = 9.32$$

$$\text{Coefficient of Variation} = CV = \frac{S}{\bar{x}} * 100\% = 23\%$$



## Note

---

- Standard Deviation (S) is used for deciding tree branching.
- Coefficient of Deviation (CV), along with count (n), is used to decide when to stop branching.
- Average (Avg) is the value in the leaf nodes.

# Computing Steps (3/12)

- Compute the standard deviation for **two attributes** (target and one predictor)
  - \* **T: target**    **X: predictor (Outlook)**    **c: each value of Outlook**

$$S(T, X) = \sum_{c \in X} P(c)S(c)$$

		Hours Played (StDev)	Count
Outlook	Overcast	3.49	4
	Rainy	7.78	5
	Sunny	10.87	5
			14



(Use HoursPlayed values in the dataset on page 83 to compute StDev for Overcast, Rainy, and Sunny).

$$\begin{aligned} S(\text{Hours}, \text{Outlook}) &= P(\text{Overcast}) * S(\text{Overcast}) + P(\text{Rainy}) * S(\text{Rainy}) + P(\text{Sunny}) * S(\text{Sunny}) \\ &= (4/14) * 3.49 + (5/14) * 7.78 + (5/14) * 10.87 \\ &= 7.66 \end{aligned}$$

## Computing Steps (4/12)

- The standard deviation for the target = 9.32
- Compute the standard deviation reduction for each of the 4 predictors.

		Hours Played (StDev)
Outlook	Overcast	3.49
	Rainy	7.78
	Sunny	10.87
SDR=1.66		

		Hours Played (StDev)
Temp.	Cool	10.51
	Hot	8.95
	Mild	7.65
SDR=0.17		

		Hours Played (StDev)
Humidity	High	9.36
	Normal	8.37
SDR=0.28		

		Hours Played (StDev)
Windy	False	7.87
	True	10.59
SDR=0.29		





## Computing Steps (5/12)


- Outlook is the predictor with the largest SDR.

$$SDR(T, X) = S(T) - S(T, X)$$

$$\begin{aligned} SDR(\text{Hours}, \text{Outlook}) &= S(\text{Hours}) - S(\text{Hours}, \text{Outlook}) \\ &= 9.32 - 7.66 = 1.66 \end{aligned}$$

# Computing Steps (6/12)

- The dataset is split based on the 3 values of the selected predictor attribute (Outlook).



A decision tree diagram on the left shows the 'Outlook' attribute branching into three categories: 'Sunny', 'Overcast', and 'Rainy'. Each category is represented by a blue rounded rectangle. Lines connect the 'Outlook' box to each of the three sub-category boxes.

Outlook	Temp	Humidity	Windy	Hours Played
Sunny	Mild	High	FALSE	45
Sunny	Cool	Normal	FALSE	52
Sunny	Cool	Normal	TRUE	23
Sunny	Mild	Normal	FALSE	46
Sunny	Mild	High	TRUE	30

Overcast	Temp	Humidity	Windy	Hours Played
Overcast	Hot	High	FALSE	46
Overcast	Cool	Normal	TRUE	43
Overcast	Mild	High	TRUE	52
Overcast	Hot	Normal	FALSE	44

Rainy	Temp	Humidity	Windy	Hours Played
Rainy	Hot	High	FALSE	25
Rainy	Hot	High	TRUE	30
Rainy	Mild	High	FALSE	35
Rainy	Cool	Normal	FALSE	38
Rainy	Mild	Normal	TRUE	48



## Computing Steps (7/12)

---

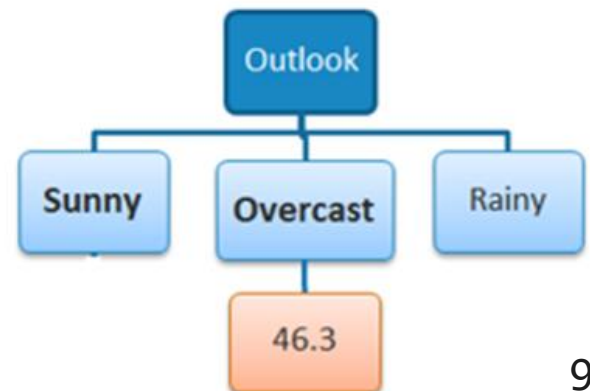
- This process is run recursively on the non-leaf branches, until all data is processed.
- We apply some branching termination criteria.
  - Coefficient of deviation(CV) for a branch becomes less than a threshold (e.g., 10%).
  - The number of records that remain in the branch is less than a threshold (e.g., 4).

## Computing Steps (8/12)

- The **Overcast** branch can stop, because its CV ( $\text{StDev} \div \text{AVG} = 8\%$ ) is less than the threshold (10%).
- Average of the 4 target values (46, 43, 52, 44) becomes the value in the leaf node.

Outlook - Overcast

		Hours Played (StDev)	Hours Played (AVG)	Hours Played (CV)	Count
Outlook	Overcast	3.49	46.3	8%	4
	Rainy	7.78	35.2	22%	5
	Sunny	10.87	39.2	28%	5



# Computing Steps (9/12)

- The **Sunny** branch needs further branching.
- The **Windy** predictor is selected, as it has the largest SDR (7.62).

Outlook - Sunny

Temp	Humidity	Windy	Hours Played
Mild	High	FALSE	45
Cool	Normal	FALSE	52
Cool	Normal	TRUE	23
Mild	Normal	FALSE	46
Mild	High	TRUE	30
			S = 10.87
			AVG = 39.2
			CV = 28%

		Hours Played (StDev)	Count
Temp	Cool	14.50	2
	Mild	7.32	3

$$SDR = 10.87 - ((2/5) * 14.5 + (3/5) * 7.32) = 0.678$$

		Hours Played (StDev)	Count
Humidity	High	7.50	2
	Normal	12.50	3

$$SDR = 10.87 - ((2/5) * 7.5 + (3/5) * 12.5) = 0.370$$

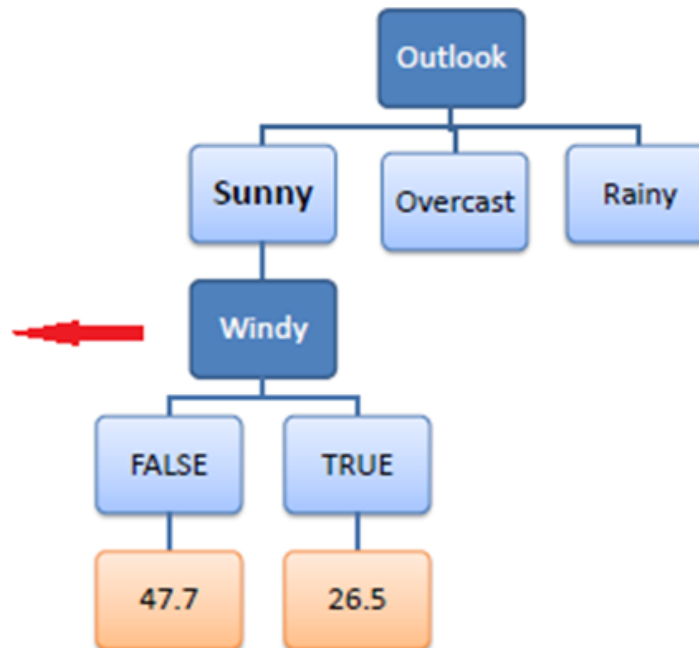
		Hours Played (StDev)	Count
Windy	False	3.09	3
	True	3.50	2

$$SDR = 10.87 - ((3/5) * 3.09 + (2/5) * 3.5) = 7.62$$

# Computing Steps (10/12)

- The **Windy** branch can stop, because the number of records for both branches (True and False) is  $< 4$ .
- The average for each branch is assigned to the respective leaf node.

Temp.	Humidity	Windy	Hours Played
Mild	High	FALSE	45
Cool	Normal	FALSE	52
Mild	Normal	FALSE	46
Cool	Normal	TRUE	23
Mild	High	TRUE	30



# Computing Steps (11/12)

- The **Rainy** branch needs further branching.
- The **Temp** predictor is selected, as it has the largest SDR (4.18).

## Outlook - Rainy

Temp	Humidity	Windy	Hours Played
Hot	High	FALSE	25
Hot	High	TRUE	30
Mild	High	FALSE	35
Cool	Normal	FALSE	38
Mild	Normal	TRUE	48
			S = 7.78
			AVG = 35.2
			CV = 22%

		Hours Played (StDev)	Count
Temp	Cool	0	1
	Hot	2.5	2
	Mild	6.5	2

$$SDR = 7.78 - ((1/5)*0 + (2/5)*2.5 + (2/5)*6.5) = 4.18$$

		Hours Played (StDev)	Count
Humidity	High	4.1	3
	Normal	5.0	2

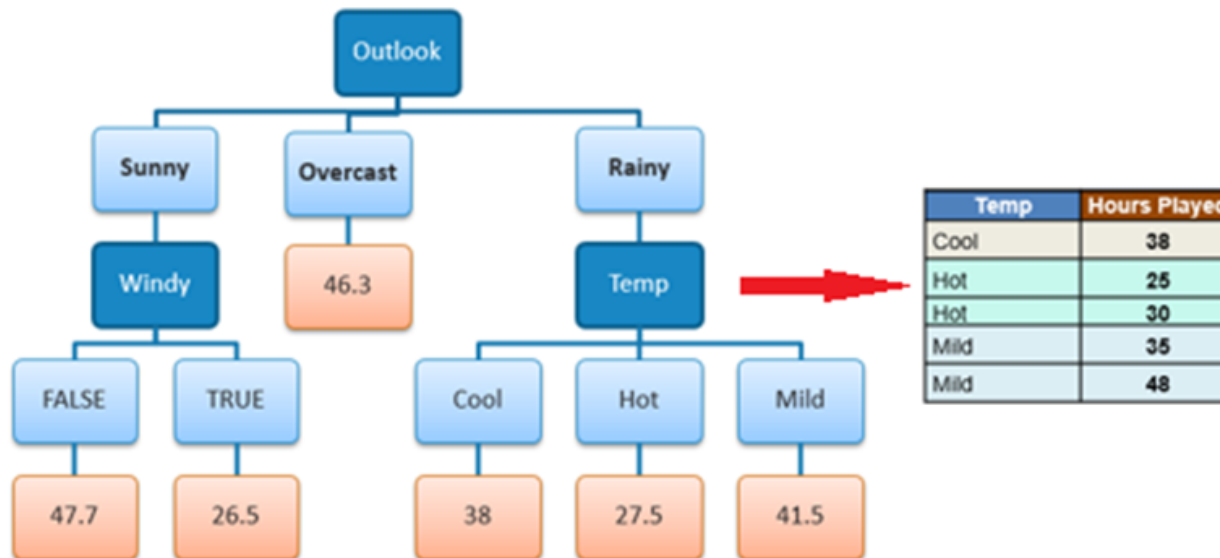
$$SDR = 7.78 - ((3/5)*4.1 + (2/5)*5.0) = 3.32$$

		Hours Played (StDev)	Count
Windy	False	5.6	3
	True	9.0	2

$$SDR = 7.78 - ((3/5)*5.6 + (2/5)*9.0) = 0.82$$

# Computing Steps (12/12)

- All 3 branches of **Temp** (Cool, Hot, Mild) can stop, as they each have  $\leq 3$  data in them.
- The average of each branch is assigned to the respective leaf node.







## Exercise

---

- By hand calculation, work out the Walkthrough example, including computations not explained.



# Decision Tree Regression Using Pandas and Scikit Learn

```
from sklearn.model_selection import train_test_split  
from sklearn import datasets  
from sklearn.tree import DecisionTreeRegressor
```

```
x_training_set, x_test_set, y_training_set, y_test_set =  
    train_test_split(X,y,test_size=0.10,random_state=42,  
                    shuffle=True)
```

```
model = DecisionTreeRegressor(max_depth=5, random_state=0)  
model.fit(x_training_set, y_training_set)
```



## Homework 3

---

- (1) By hand calculation, build a decision tree for regression, using the dataset shown on the next page.
  - The dataset has 14 data samples, each with a student's major, year, gender, and StudyHours. StudyHours is the target attribute.
- (2) Write a program in Python to do the above, **using** Pandas, Numpy and Scikitlearn.
- **Submit both in a WORD file.**



# Dataset to Use

Major	Year	Gender	StudyHours
SW	2	F	20
Math	3	M	20
Art	3	F	15
English	3	M	28
Math	3	F	26
English	3	M	17
Math	3	F	26
SW	3	F	40
SW	3	M	33
English	3	M	18
Math	3	M	25
Math	3	F	30
SW	3	F	45
Art	3	M	20



# Roadmap

---

- Decision Trees
- Decision Tree Regression
- Decision Tree Pruning



# Acknowledgments

---

- <http://www.cs.kent.edu/~jin/DM07/ClassificationDecisionTree.ppt>
- <https://www.quora.com/How-can-I-find-a-real-step-by-step-example-of-a-decision-tree-pruning-to-overcome-overfitting>
- [www-scf.usc.edu/~csci567/index.html](http://www-scf.usc.edu/~csci567/index.html)
- [https://en.wikipedia.org/wiki/Decision\\_tree\\_pruning](https://en.wikipedia.org/wiki/Decision_tree_pruning)



# Pruning the Decision Tree

- Pruning means dropping some of the subtrees that are not worth including in the model.
  - Results in a reduced tree height and/or reduced number of branches
- Purposes of pruning
  - Limit the computation costs in tree induction and deduction
  - Address overfitting (of a complex model)
- Two ways of pruning
  - **Pre-pruning**: early stopping of tree induction
  - **Post-pruning**: pruning the fully grown tree



# Roadmap: Decision Tree Pruning

---

- Pre-pruning
- Post-pruning





## Pre-Pruning (1/2)

---

- Pre-pruning means not constructing a full decision tree by stopping subtree construction at some nodes.
- Pre-pruning has the risk of stopping the tree induction prematurely.
- **Note:** Decision-tree induction already uses pre-pruning. It takes the greedy approach and selects the “best split” at each step, giving up all other splits.
  - Further, it stops tree induction for a node
    - if all records belong to the same class
    - if all the attribute values are the same



## Pre-Pruning (2/2)

---

- Pre-pruning can be more aggressive by setting more restrictive **early-stopping conditions**.
  - Stop if the number of records is less than some user-specified threshold
  - Stop if expanding the current node does not improve impurity measures (e.g., Gini or information gain).



# Roadmap: Decision Tree Pruning

---

- Pre-pruning
- Post-pruning



# Post-Pruning

- Remove some nodes of the fully grown decision tree in a **bottom-up** or **top-down** fashion.
- If, at a node, **classification error** does not become worse after removing a subtree rooted at that node, replace the subtree with a leaf node.
  - In this case, the content of the leaf node is determined from (in a classification task) the majority class or (in a regression task) average of the records in the subtree.



# Post-Pruning Methods

---

- Reduced error pruning (We will cover only this.)
- Pessimistic pruning
- Cost-complexity pruning
- Rule post-pruning
- ...



# Reduced Error Pruning

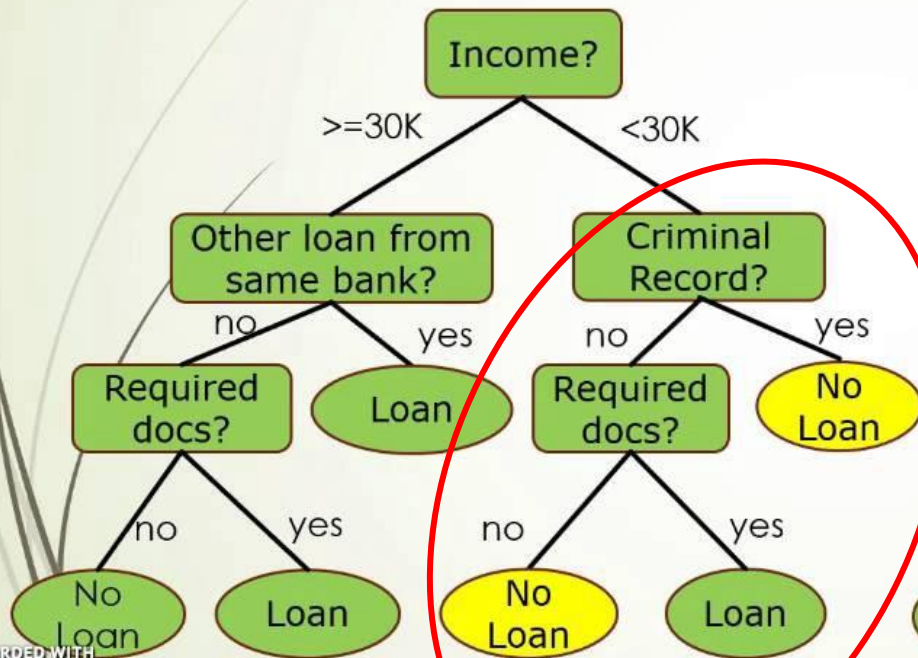
- **Bottom-up**, starting at the leaves, each node is a candidate for pruning.
- Pruning consists of replacing a subtree with the most common class of the subtree.
- Nodes are removed only if the resulting tree performs no worse on the **validation set** (\* not test set).
- Pruning stops when accuracy no longer increases.

# Illustration: Pruning and Replacing a Subtree with a Leaf Node

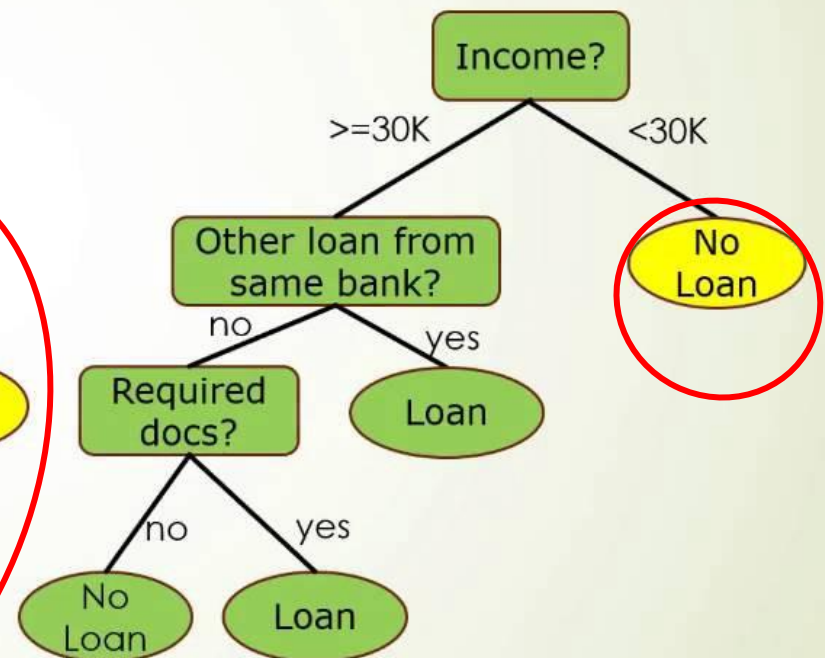
10

## Tree Pruning Example

An Unpruned Decision Tree



A Pruned Decision Tree





## (Reminder) Validation Set

- To tune a trained model (before testing and predicting), we need to use a validation set.
- The validation set comes out of the train set.
- Test (prediction) is done only with the test set.
- The initial full training dataset is split into train set (X\_train, y\_train) and test set (X\_test)
- The train set is further split (using train\_test\_split) into (reduced) train set and validation set.  

```
X_train, X_val, y_train, y_val = train_test_split(X_train,  
y_train, test_size=0.3, random_state=12)
```



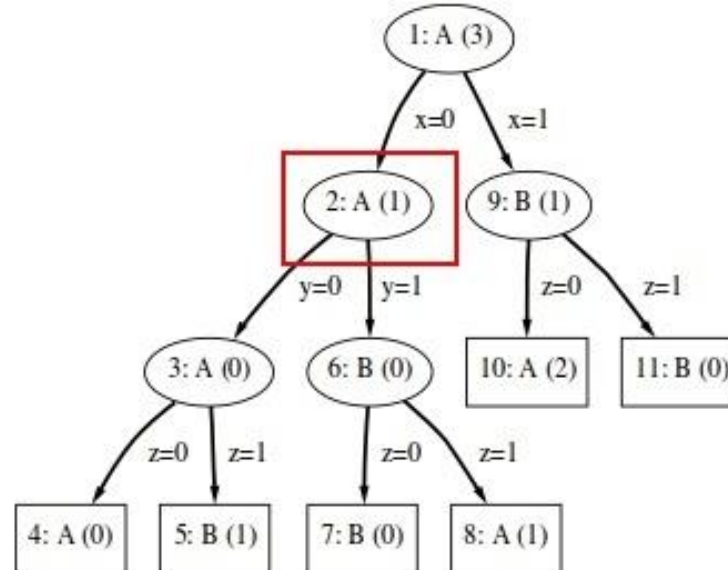
# Step by Step Illustration (1/5)

- Example validation set
  - target class values: A or B
  - Predictor variables: x, y, z

x	y	z	class
0	0	1	A
0	1	1	B
1	1	0	B
1	0	0	B

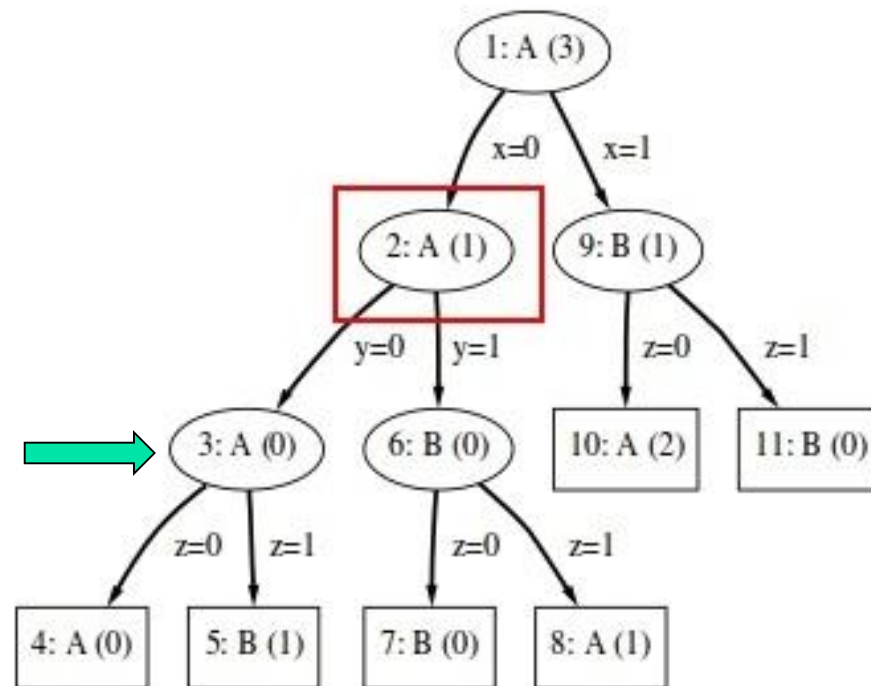
## Step-by-Step Illustration (2/5)

- In the validation test, in every node the target value and number of misclassified records are recorded (in parenthesis).
- In the example tree below (each node with node ID), node(2) (in a red box) misclassifies one record from the validation set.



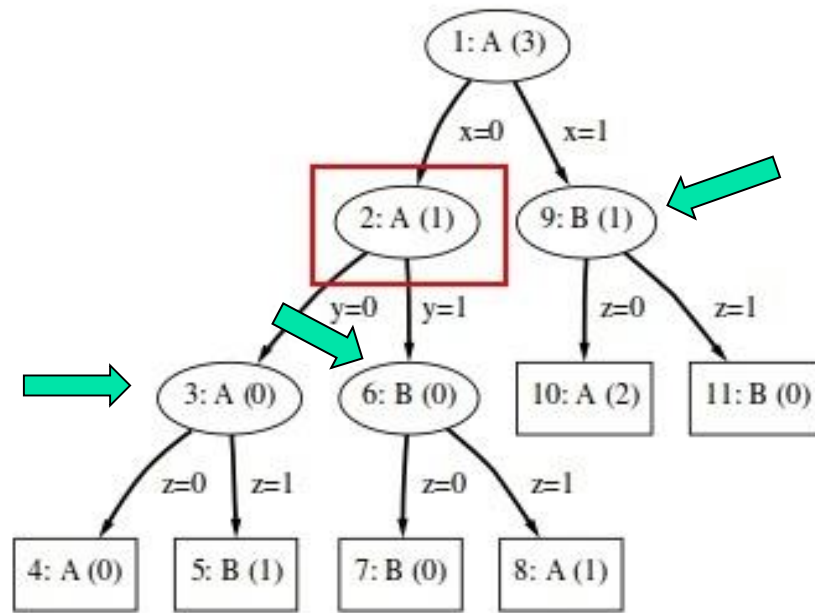
## Step-by-Step Illustration (3/5)

- Node 3 in the tree can be made into a leaf node, since it does not make worse error than as a subtree.



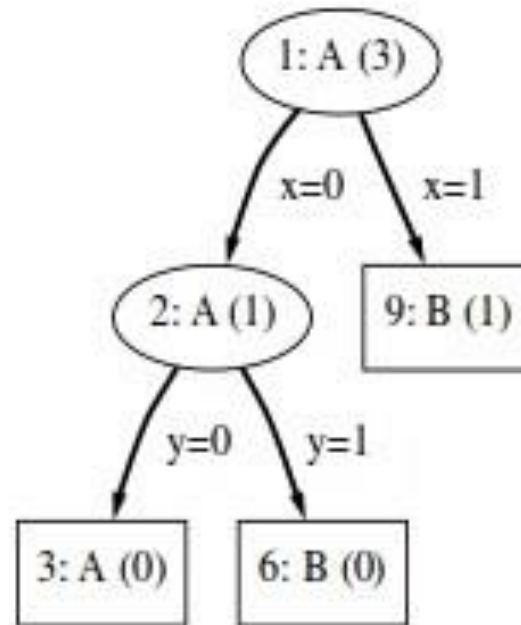
## Step-by-Step Illustration (4/5)

- Same consideration for nodes 6 and 9.
- However, node 2 cannot be made into a leaf, since it makes one error, while as a subtree, with the newly-created leaves 3 and 6, it makes no error.



## Step-by-Step Illustration (5/5)

- The following is the final pruned tree.
- This decision tree is used against the test set for deduction.





# End of Class

---