Machine Learning
CMPSCI 589

Bruno C. da Silva
bsilva@cs.umass.edu

Decision Trees (2/2)

UMassAmherst
College of Information & Computer Sciences

Review: Learning a Decision Tree

- General procedure to create a decision tree
  1. Select an attribute to add to the tree (starting from the root) → new node
  2. Add, to this node, one branch for each possible value of the selected attribute
  3. Partition the instances (training examples)
     - Assign each instance to its corresponding branch, based on the value of that instance's attribute
  4. Repeat these steps, recursively, for each resulting partition (i.e., for each children node)

Name	***Age***	***Gender***	***TrafficTicket***	**Class: High-Risk Driver**
John	43	M	Yes	**High Risk**
Peter	18	M	No	**High Risk**
Anna	35	F	No	**Low Risk**
Paula	19	F	No	**High Risk**
Mark	90	M	Yes	**High Risk**
Marisa	19	F	Yes	**High Risk**
Bob	30	M	No	**Low Risk**

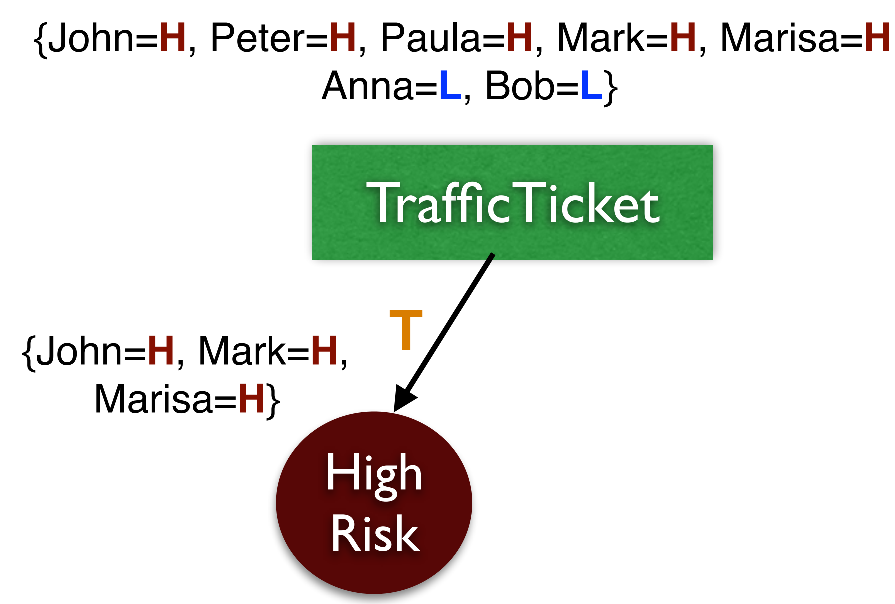
{John=**H**, Peter=**H**, Paula=**H**, Mark=**H**, Marisa=**H**,
Anna=**L**, Bob=**L**}

TrafficTicket

Review: Learning a Decision Tree

- General procedure to create a decision tree
  1. Select an attribute to add to the tree (starting from the root) → new node
  2. Add, to this node, one branch for each possible value of the selected attribute
  3. Partition the instances (training examples)
     - Assign each instance to its corresponding branch, based on the value of that instance's attribute
  4. Repeat these steps, recursively, for each resulting partition (i.e., for each children node)

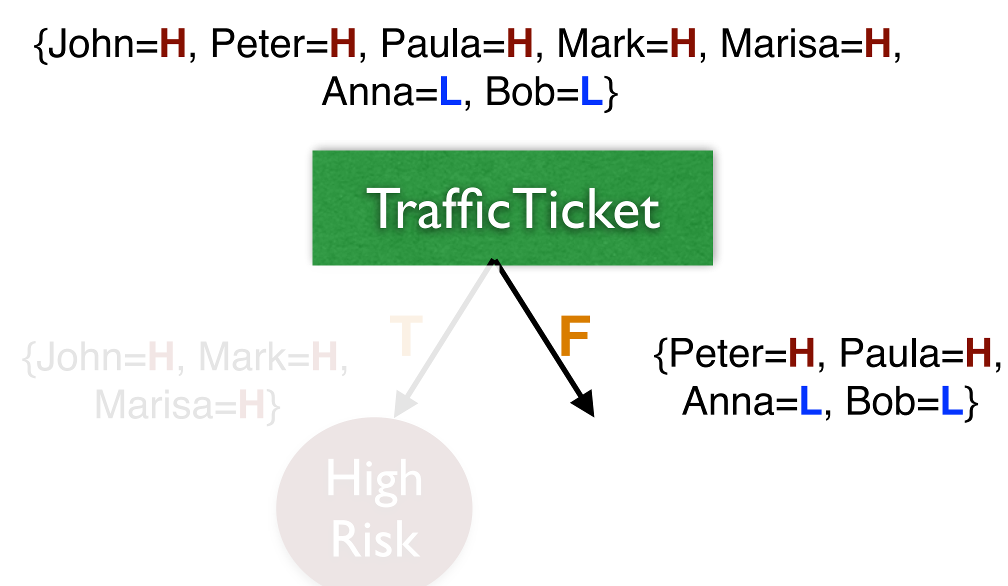
Name	***Age***	***Gender***	***TrafficTicket***	**Class: High-Risk Driver**
John	43	M	Yes	**High Risk**
Peter	18	M	No	**High Risk**
Anna	35	F	No	**Low Risk**
Paula	19	F	No	**High Risk**
Mark	90	M	Yes	**High Risk**
Marisa	19	F	Yes	**High Risk**
Bob	30	M	No	**Low Risk**



Review: Learning a Decision Tree

- General procedure to create a decision tree
  1. Select an attribute to add to the tree (starting from the root) → new node
  2. Add, to this node, one branch for each possible value of the selected attribute
  3. Partition the instances (training examples)
     - Assign each instance to its corresponding branch, based on the value of that instance's attribute
  4. Repeat these steps, recursively, for each resulting partition (i.e., for each children node)

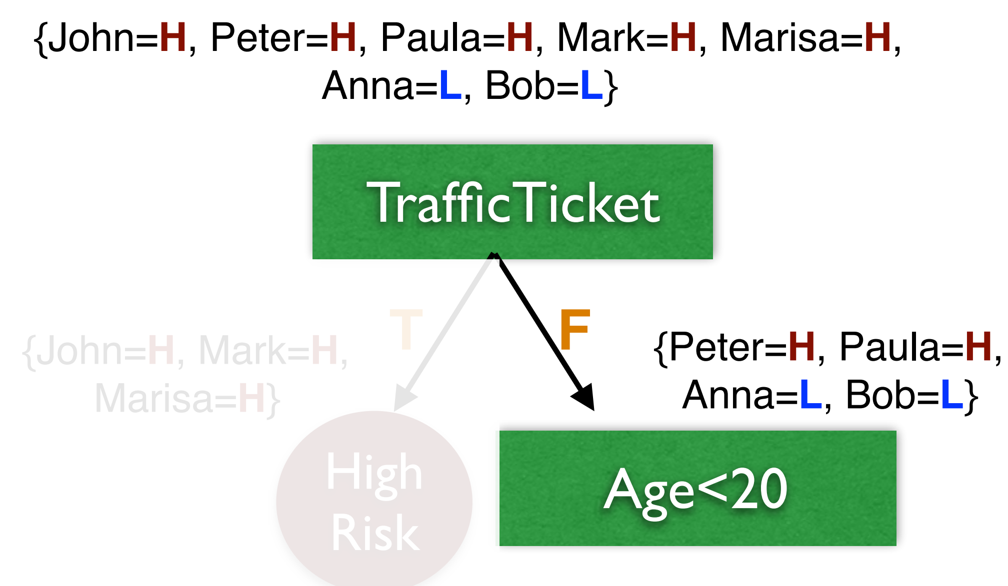
Name	*Age*	*Gender*	*TrafficTicket*	Class: High-Risk Driver
John	43	M	Yes	**High Risk**
Peter	18	M	No	**High Risk**
Anna	35	F	No	**Low Risk**
Paula	19	F	No	**High Risk**
Mark	90	M	Yes	**High Risk**
Marisa	19	F	Yes	**High Risk**
Bob	30	M	No	**Low Risk**



Review: Learning a Decision Tree

- General procedure to create a decision tree
    1. Select an attribute to add to the tree (starting from the root) → new node
    2. Add, to this node, one branch for each possible value of the selected attribute
    3. Partition the instances (training examples)
        - Assign each instance to its corresponding branch, based on the value of that instance's attribute
    4. Repeat these steps, recursively, for each resulting partition (i.e., for each children node)

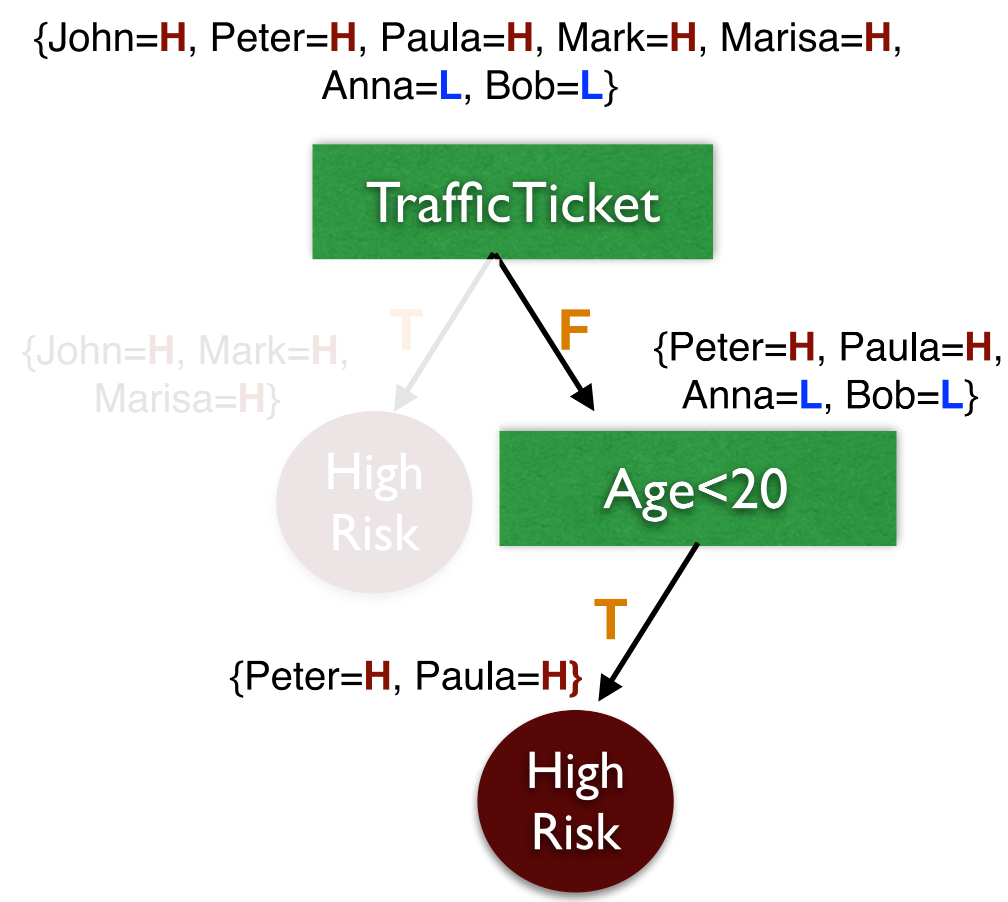
Name	***Age***	***Gender***	***TrafficTicket***	**Class: High-Risk Driver**
John	43	M	Yes	**High Risk**
Peter	18	M	No	**High Risk**
Anna	35	F	No	**Low Risk**
Paula	19	F	No	**High Risk**
Mark	90	M	Yes	**High Risk**
Marisa	19	F	Yes	**High Risk**
Bob	30	M	No	**Low Risk**



Review: Learning a Decision Tree

- General procedure to create a decision tree
  1. Select an attribute to add to the tree (starting from the root) → new node
  2. Add, to this node, one branch for each possible value of the selected attribute
  3. Partition the instances (training examples)
     - Assign each instance to its corresponding branch, based on the value of that instance's attribute
  4. Repeat these steps, recursively, for each resulting partition (i.e., for each children node)

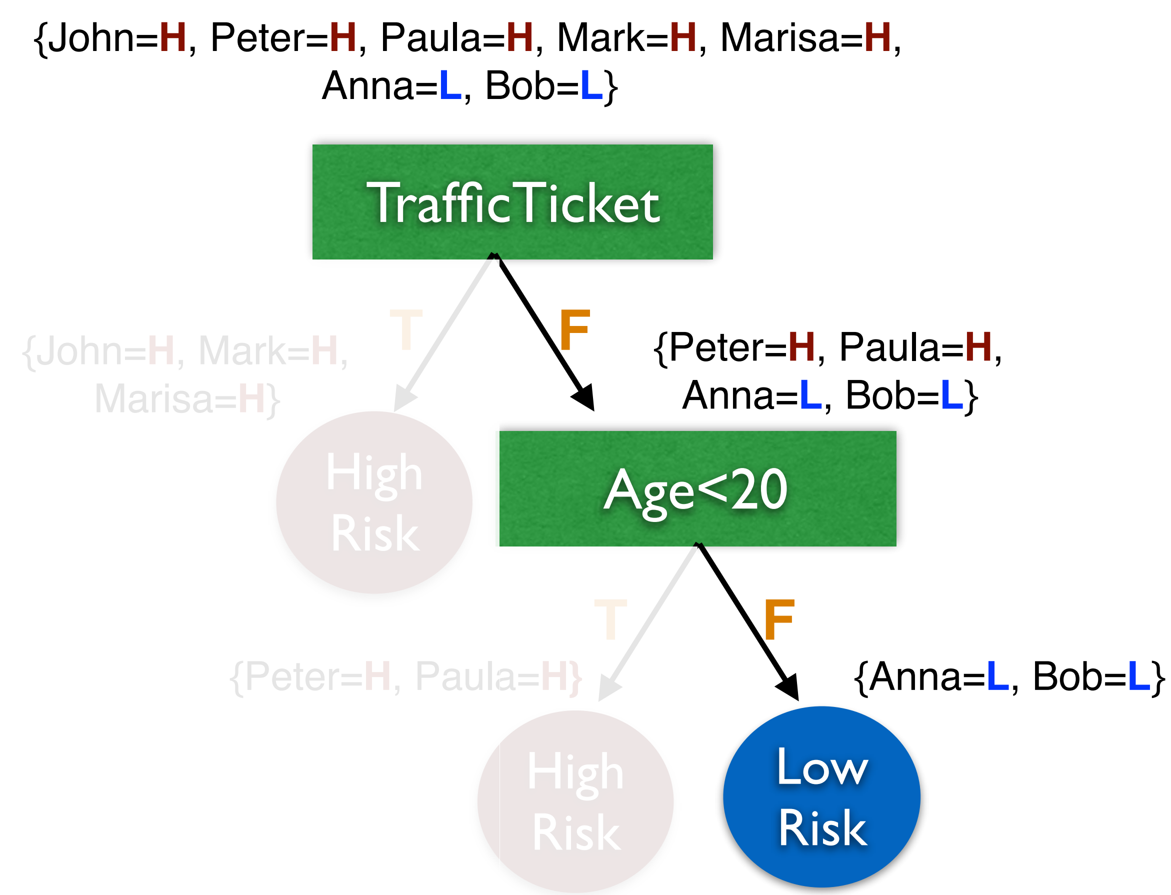
Name	***Age***	***Gender***	***TrafficTicket***	**Class: High-Risk Driver**
John	43	M	Yes	**High Risk**
Peter	18	M	No	**High Risk**
Anna	35	F	No	**Low Risk**
Paula	19	F	No	**High Risk**
Mark	90	M	Yes	**High Risk**
Marisa	19	F	Yes	**High Risk**
Bob	30	M	No	**Low Risk**



Review: Learning a Decision Tree

- General procedure to create a decision tree
    1. Select an attribute to add to the tree (starting from the root) → new node
    2. Add, to this node, one branch for each possible value of the selected attribute
    3. Partition the instances (training examples)
        - Assign each instance to its corresponding branch, based on the value of that instance's attribute
    4. Repeat these steps, recursively, for each resulting partition (i.e., for each children node)

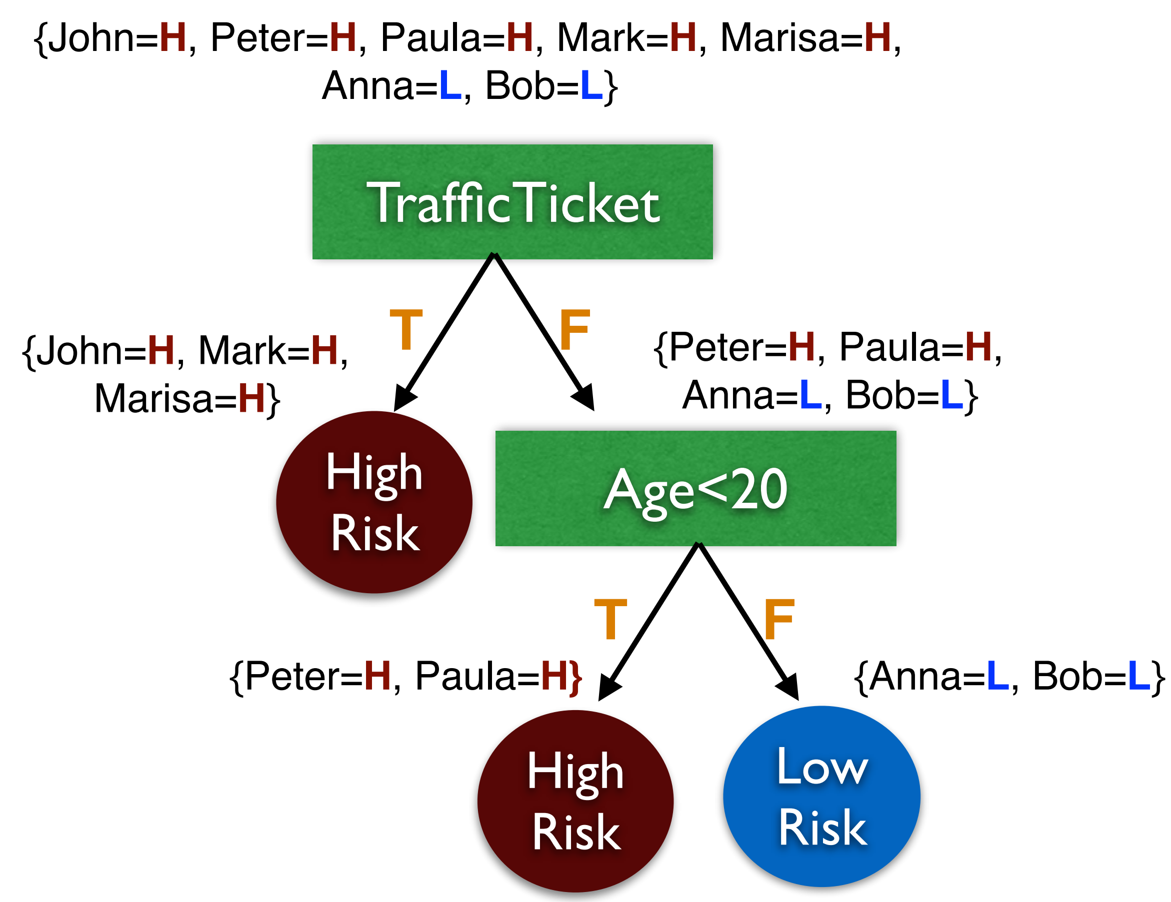
Name	*Age*	*Gender*	*TrafficTicket*	Class: High-Risk Driver
John	43	M	Yes	**High Risk**
Peter	18	M	No	**High Risk**
Anna	35	F	No	**Low Risk**
Paula	19	F	No	**High Risk**
Mark	90	M	Yes	**High Risk**
Marisa	19	F	Yes	**High Risk**
Bob	30	M	No	**Low Risk**



Review: Learning a Decision Tree

- General procedure to create a decision tree
    1. Select an attribute to add to the tree (starting from the root) → new node
    2. Add, to this node, one branch for each possible value of the selected attribute
    3. Partition the instances (training examples)
        - Assign each instance to its corresponding branch, based on the value of that instance's attribute
    4. Repeat these steps, recursively, for each resulting partition (i.e., for each children node)

Name	***Age***	***Gender***	***TrafficTicket***	**Class: High-Risk Driver**
John	43	M	Yes	**High Risk**
Peter	18	M	No	**High Risk**
Anna	35	F	No	**Low Risk**
Paula	19	F	No	**High Risk**
Mark	90	M	Yes	**High Risk**
Marisa	19	F	Yes	**High Risk**
Bob	30	M	No	**Low Risk**



Review: Learning a Decision Tree

- General procedure to create a decision tree
  1. Select an attribute to add to the tree (starting from the root) → new node
  2. Add, to this node, one branch for each possible value of the selected attribute
  3. Partition the instances (training examples)
     - Assign each instance to its corresponding branch, based on the value of that instance's attribute
  4. Repeat these steps, recursively, for each resulting partition (i.e., for each children node)

?

Selecting an Attribute to Test

- Ideally, we should select an attribute such that
  - "all instances of class A go to one branch, all instances of class B to the other branch"
  - i.e., attribute that results in partitions whose instances are as <u>homogenous</u> as possible

- How to quantify how homogenous a set of instances is?
  - ***Information, or entropy***
  - Information is measured in bits (or fractions of a bit)

Let's suppose we test Age, and the instances associated with Age=Young look like this

Was that a useful attribute to test?

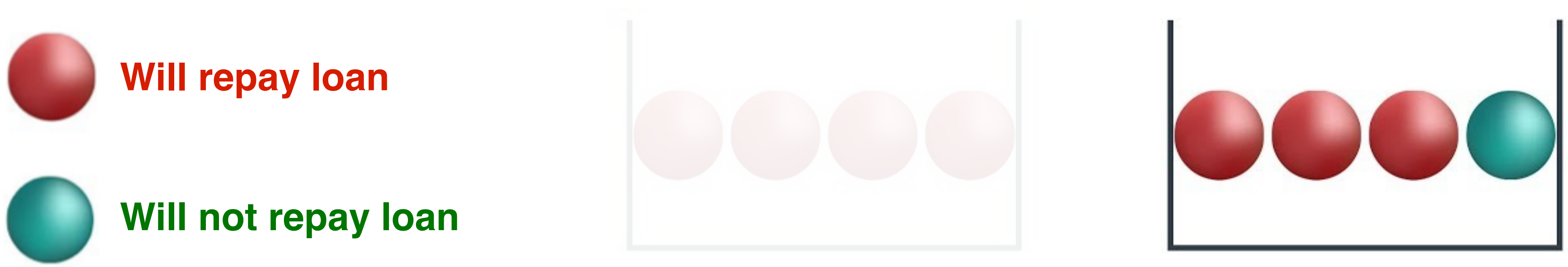
Do we have a good idea about whether the person is going to repay the loan or not?

Selecting an Attribute to Test

- Ideally, we should select an attribute such that
    - “all instances of class A go to one branch, all instances of class B to the other branch”
    - i.e., attribute that results in partitions whose instances are as <u>homogenous</u> as possible

- How to quantify how homogenous a set of instances is?
    - ***Information, or entropy***
    - Information is measured in bits (or fractions of a bit)

Let’s suppose we test Age, and the instances associated with Age=Young look like this



Was that a useful attribute to test?

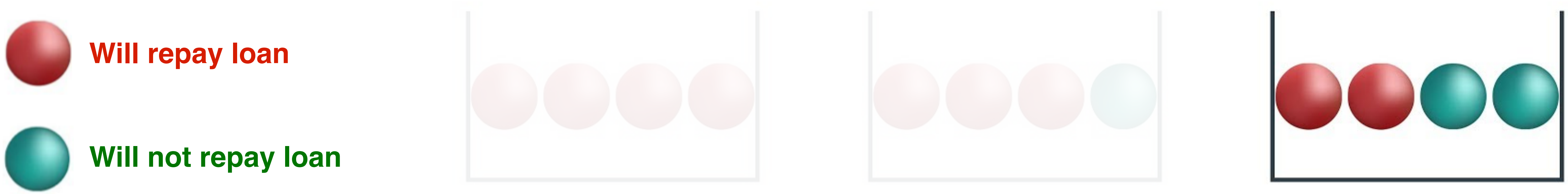
Do we have a good idea about whether the person is going to repay the loan or not?

Selecting an Attribute to Test

- Ideally, we should select an attribute such that
  - "all instances of class A go to one branch, all instances of class B to the other branch"
  - i.e., attribute that results in partitions whose instances are as <u>homogenous</u> as possible

- How to quantify how homogenous a set of instances is?
  - ***Information, or entropy***
  - Information is measured in bits (or fractions of a bit)

Let's suppose we test Age, and the instances associated with Age=Young look like this



Was that a useful attribute to test?

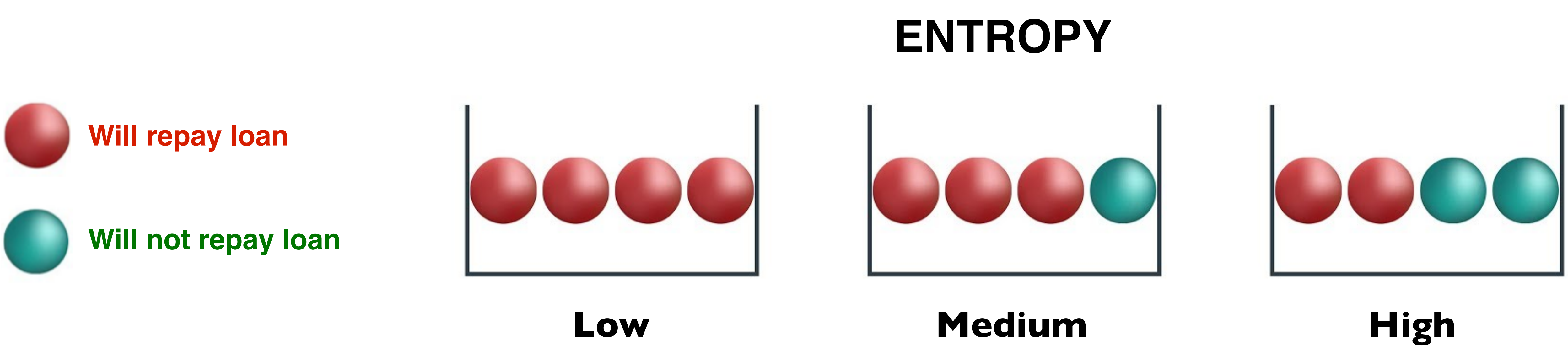
Do we have a good idea about whether the person is going to repay the loan or not?

Selecting an Attribute to Test

- Ideally, we should select an attribute such that
  - "all instances of class A go to one branch, all instances of class B to the other branch"
  - i.e., attribute that results in partitions whose instances are as <u>homogenous</u> as possible

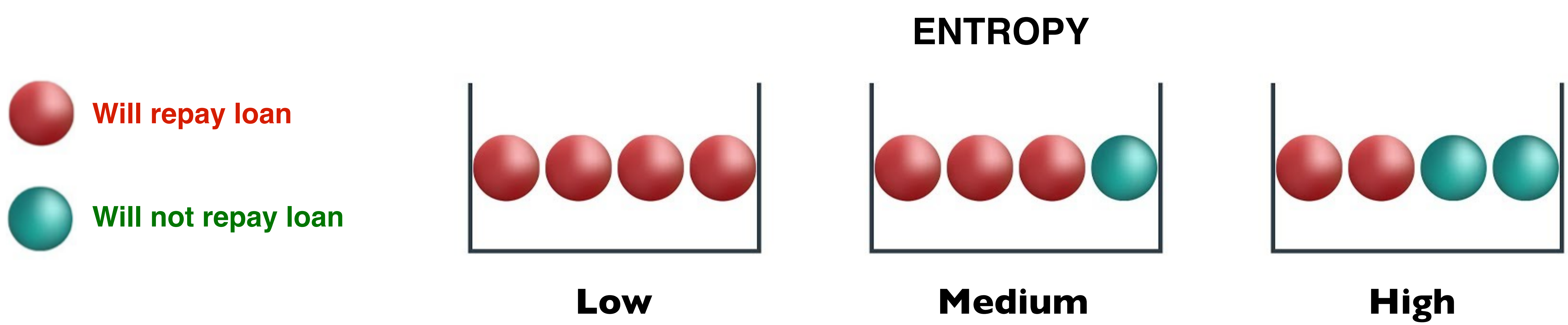
- How to quantify how homogenous a set of instances is?
  - ***Information, or entropy***
  - Information is measured in bits (or fractions of a bit)

Let's suppose we test Age, and the instances associated with Age=Young look like this



Selecting an Attribute to Test

- How to quantify how homogenous a set of instances is?
  - ***Information, or entropy***
  - Information is measured in bits (or fractions of a bit)



- Intuitively, quantifies how random a given quantity (e.g., class) is within a dataset
- Associated with how hard it is to predict the class based on an attribute

- Higher entropy
  - → instances of a same class are all mixed up
  - → testing the attribute that resulted in that partition of the data was not very useful

Selecting an Attribute to Test

- How to quantify how homogenous a set of instances is?
  - ***Information, or entropy***
  - Information is measured in bits (or fractions of a bit)

- Given a distribution of labels/classes in a partition of the data
  - how much information is required to predict the class
  - this is the *entropy* of that distribution

$$I(p_1, p_2, \ldots, p_n) = -p_1 \log_2(p_1) - p_2 \log_2(p_2) \ldots - p_n \log_2(p_n)$$

Probability that class #1 (H) appears in the partition of the data

{John=**H**, Peter=**H**, Paula=**H**, Mark=**H**, Marisa=**H**, Anna=**L**, Bob=**L**}

Selecting an Attribute to Test

- How to quantify how homogenous a set of instances is?
  - ***Information, or entropy***
  - Information is measured in bits (or fractions of a bit)

- Given a distribution of labels/classes in a partition of the data
  - how much information is required to predict the class
  - this is the *entropy* of that distribution

$$I(p_1, p_2, \ldots, p_n) = -p_1 \log_2(p_1) - p_2 \log_2(p_2) \ldots - p_n \log_2(p_n)$$

Probability that class #2 (L) appears in the partition of the data

{John=**H**, Peter=**H**, Paula=**H**, Mark=**H**, Marisa=**H**, Anna=**L**, Bob=**L**}

Selecting an Attribute to Test

- How to quantify how homogenous a set of instances is?
  - ***Information, or entropy***
  - Information is measured in bits (or fractions of a bit)

- Given a distribution of labels/classes in a partition of the data
  - how much information is required to predict the class
  - this is the *entropy* of that distribution

$$I(p_1, p_2, \ldots, p_n) = -p_1 \log_2(p_1) - p_2 \log_2(p_2) \ldots - p_n \log_2(p_n)$$

High entropy because the class is completely undetermined (50% instances are H, 50% instances are L)

{John=**H**, Peter=**H**, Anna=**L**, Bob=**L**}

Pr(H) = 2/4
Pr(L) = 2/4

I(2/4, 2/4) **=** -2/4 **log₂**(2/4) -2/4 **log₂**(2/4)

= 1

Selecting an Attribute to Test

- How to quantify how homogenous a set of instances is?
  - ***Information, or entropy***
  - Information is measured in bits (or fractions of a bit)

- Given a distribution of labels/classes in a partition of the data
  - how much information is required to predict the class
  - this is the *entropy* of that distribution

$$I(p_1, p_2, \ldots, p_n) = -p_1 \log_2(p_1) - p_2 \log_2(p_2) \ldots - p_n \log_2(p_n)$$

Low entropy because the class is completely determined (100% instances are H!)

{John=**H**, Peter=**H**}

Pr(H) = 2/2
Pr(L) = 0/2

I(2/2, 0/2) = -2/2 **log₂**(2/2) -0/2 **log₂**(0/2)

= 0

Selecting an Attribute to Test

- How to quantify how homogenous a set of instances is?
  - ***Information, or entropy***
  - Information is measured in bits (or fractions of a bit)

- Given a distribution of labels/classes in a partition of the data
  - how much information is required to predict the class
  - this is the *entropy* of that distribution

“Medium" entropy because the class is almost determined (almost sure it is H, but there’s still some uncertainty)

$$I(p_1, p_2, \ldots, p_n) = -p_1 \log_2(p_1) - p_2 \log_2(p_2) \ldots - p_n \log_2(p_n)$$

{John=**A**, Peter=**A**, Paula=**A**, Mark=**A**, Marisa=**A**, Anna=**B**, Bob=**B**}

I(5/7, 2/7) = -5/7 **log**$_2$(5/7) -2/7 **log**$_2$(2/7) = 0.8631

Pr(A) = 5/7
Pr(B) = 2/7

Selecting an Attribute to Test

- Decision tree to predict whether a person will play tennis

Weather	Temperature	Humidity	Windy	PlayTennis
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

- Which attribute to test first?
- Let's consider testing Weather

Selecting an Attribute to Test

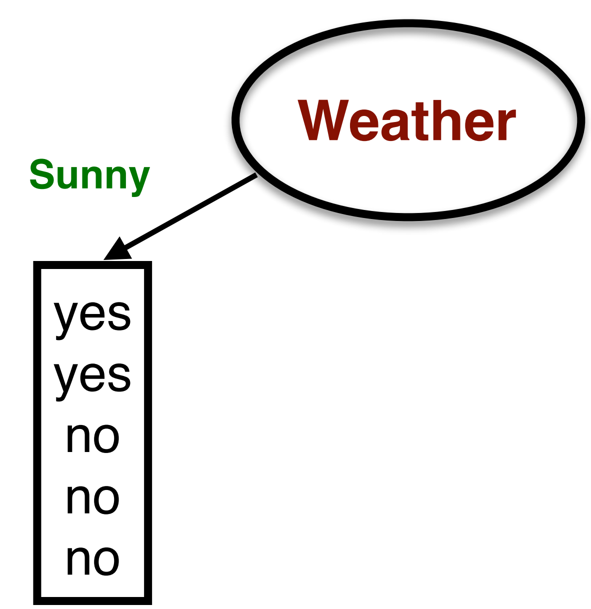
- Decision tree to predict whether a person will play tennis

Weather	Temperature	Humidity	Windy	PlayTennis
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

- Which attribute to test first?
- Let’s consider testing Weather

Original dataset: 9 instances “Yes"
5 instances "No"

yes yes yes yes yes yes yes yes yes
no no no no no



Selecting an Attribute to Test

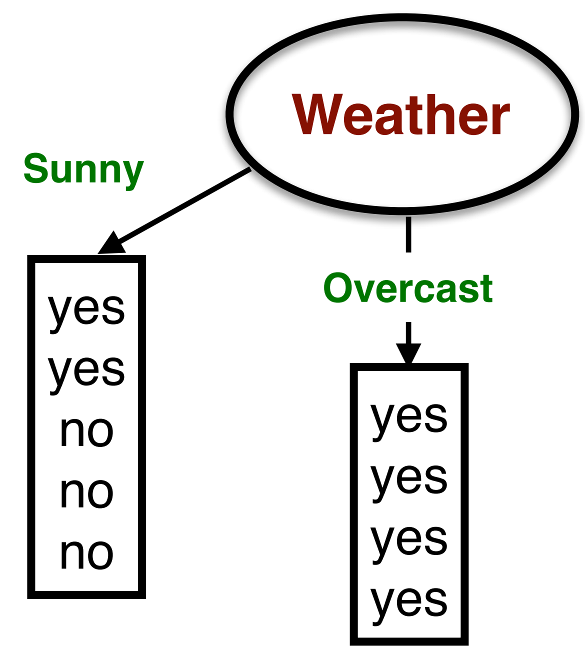
- Decision tree to predict whether a person will play tennis

Weather	Temperature	Humidity	Windy	PlayTennis
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

- Which attribute to test first?
- Let's consider testing Weather

Original dataset: 9 instances "Yes"
5 instances "No"

yes yes yes yes yes yes yes yes yes
no no no no no



Selecting an Attribute to Test

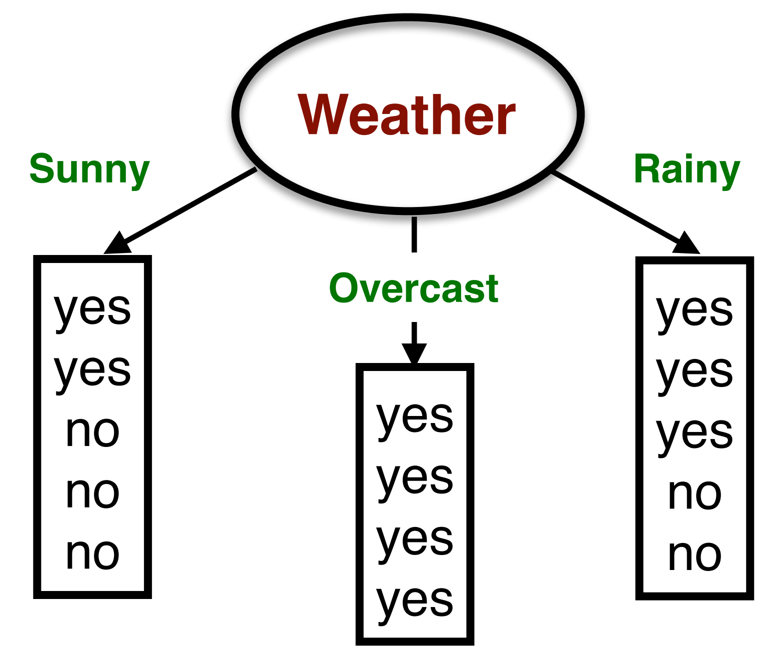
- Decision tree to predict whether a person will play tennis

Weather	Temperature	Humidity	Windy	PlayTennis
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

- Which attribute to test first?
- Let's consider testing Weather

Original dataset: 9 instances "Yes"
5 instances "No"

yes yes yes yes yes yes yes yes yes
no no no no no



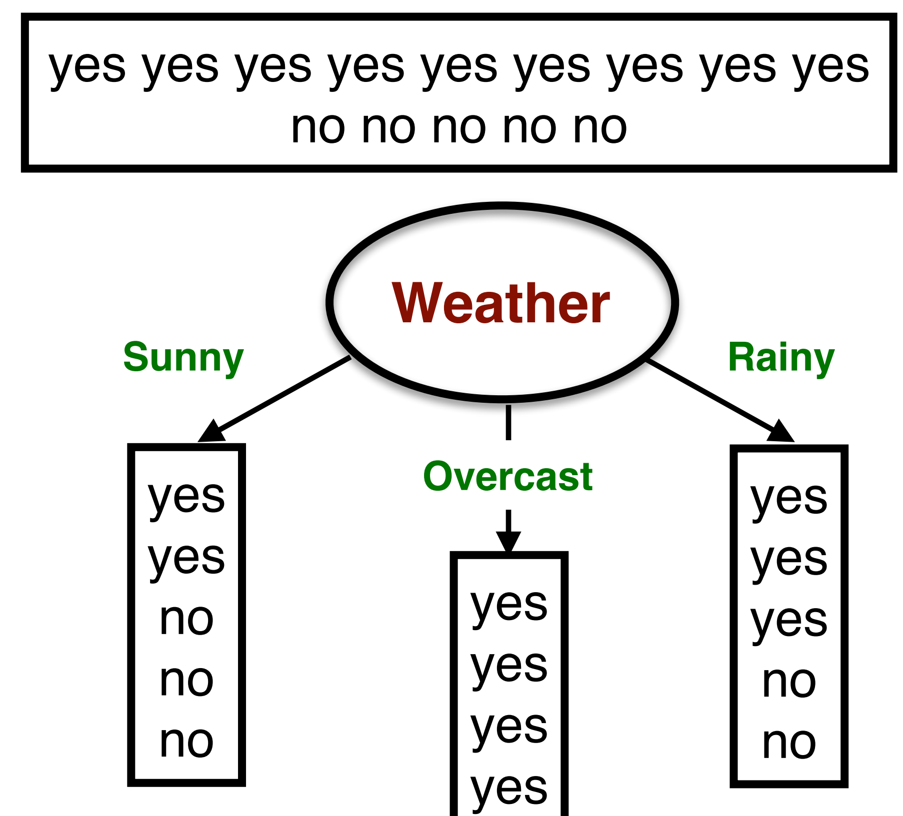
Selecting an Attribute to Test

- Decision tree to predict whether a person will play tennis

- Entropy of the original dataset:
  - $\mathbf{I}(9/14, 5/14) = -9/14 \log_2(9/14) - 5/14 \log_2(5/14)$
    $= \mathbf{0.940\ bits}$
- Entropy of partitions resulting from testing Weather:
  - **Weather=Sunny**
    - $\mathbf{I}(2/5, 3/5) = -2/5 \log_2(2/5) - 3/5 \log_2(3/5)$
      $= 0.971$ bits
  - **Weather=Overcast**
    - $\mathbf{I}(4/4, 0/4) = -4/4 \log_2(4/4) - 0/4 \log_2(0/4)$
      $= 0$ bits
  - **Weather=Rainy**
    - $\mathbf{I}(3/5, 2/5) = -3/5 \log_2(3/5) - 2/5 \log_2(2/5)$
      $= 0.971$ bits
- Average entropy of the resulting partitions
  - $(5/14) \times 0.971 + (4/14) \times 0 + (5/14) \times 0.971 = \mathbf{0.693\ bits}$

- Which attribute to test first?
- Let's consider testing Weather

Original dataset: 9 instances "Yes"
5 instances "No"



Selecting an Attribute to Test

- Decision tree to predict whether a person will play tennis

- **Entropy of the original dataset:**
  - $\mathbf{I}(9/14, 5/14) = -9/14 \log_2(9/14) - 5/14 \log_2(5/14)$
    $= \mathbf{0.940\ bits}$

- **Entropy of partitions resulting from testing Weather:**
  - **Weather=Sunny**
    - $\mathbf{I}(2/5, 3/5) = -2/5 \log_2(2/5) - 3/5 \log_2(3/5)$
      $= 0.971$ bits
  - **Weather=Overcast**
    - $\mathbf{I}(4/4, 0/4) = -4/4 \log_2(4/4) - 0/4 \log_2(0/4)$
      $= 0$ bits
  - **Weather=Rainy**
    - $\mathbf{I}(3/5, 2/5) = -3/5 \log_2(3/5) - 2/5 \log_2(2/5)$
      $= 0.971$ bits

- **Average entropy of the resulting partitions**
  - $(5/14)\times 0.971 + (4/14)\times 0 + (5/14)\times 0.971 = \mathbf{0.693\ bits}$

By testing the attribute **Weather**, the entropy of the classes decreased by $0.940 - 0.693 = \mathbf{0.247\ bits}$

Information Gain

- quantifies how much information about the class is obtained by testing a given attribute

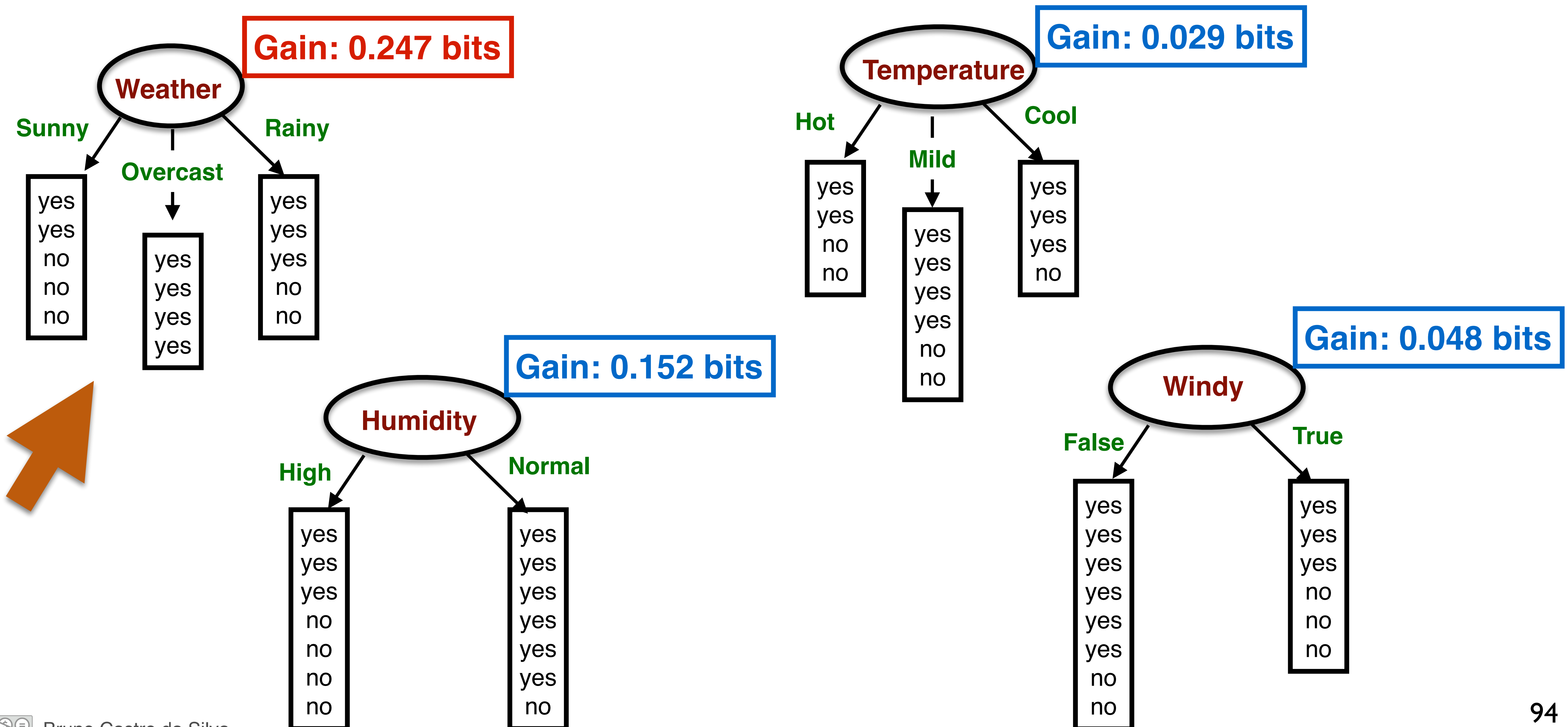
> The algorithm will test, first, the attributes that result in higher information gain

Selecting an Attribute to Test

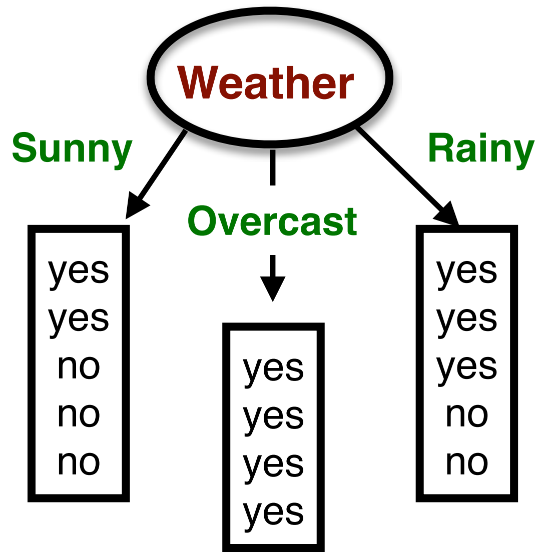
Information Gain

- quantifies how much information about the class is obtained by testing a given attribute

The algorithm will test, first, the attributes that result in higher information gain

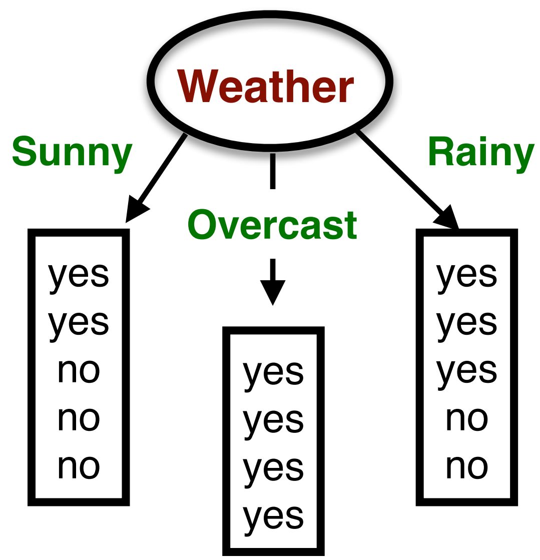


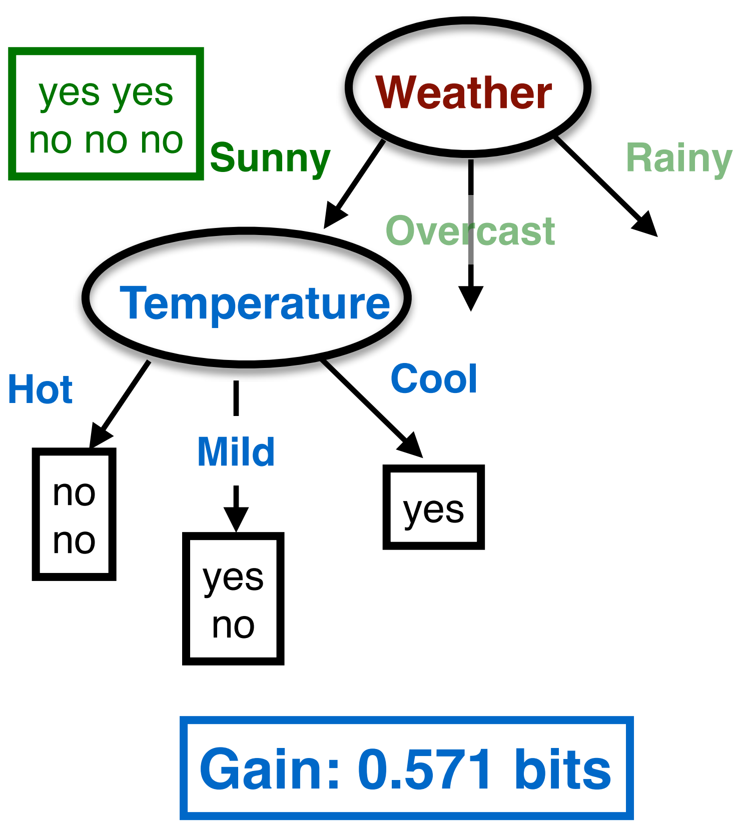
Selecting an Attribute to Test



- We have decided that the 1st attribute to test is Weather
- What should be tested next, on the Sunny branch?
  - i.e., should we test Temperature, Windy, or Humidity?

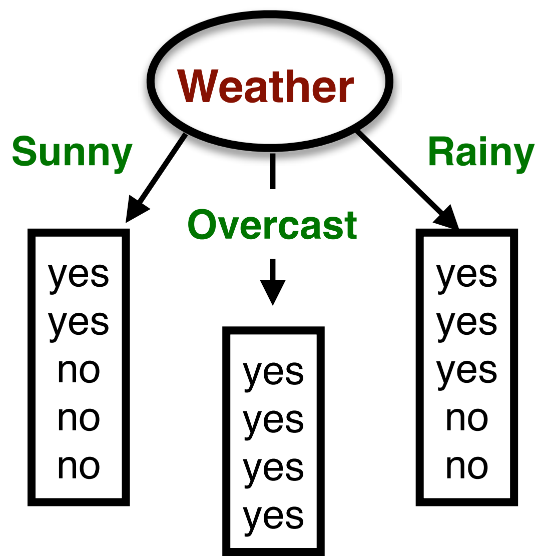
Selecting an Attribute to Test



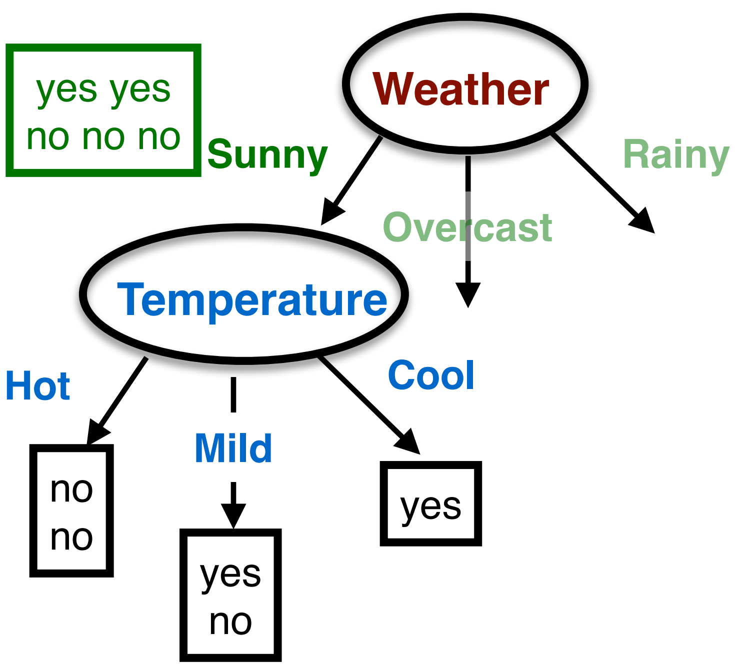


- We have decided that the 1st attribute to test is Weather
- What should be tested next, on the Sunny branch?
  - i.e., should we test Temperature, Windy, or Humidity?

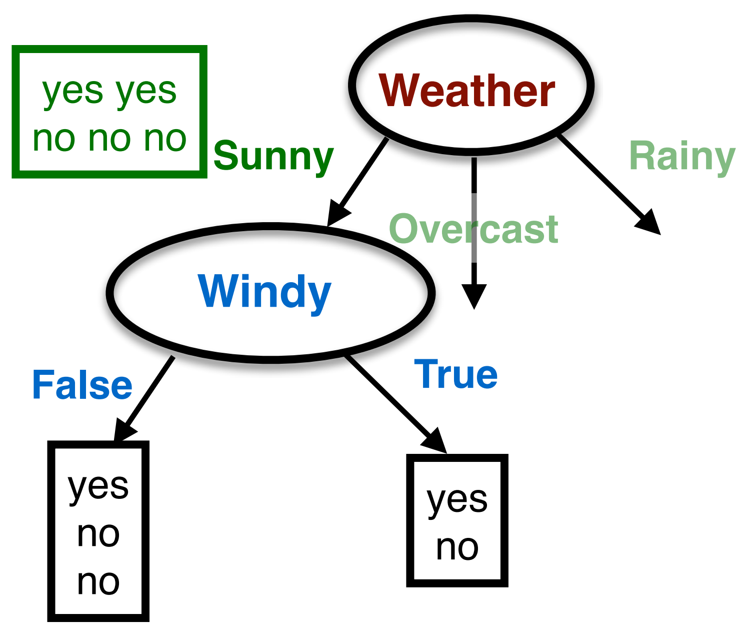
Selecting an Attribute to Test



- We have decided that the 1^{st} attribute to test is Weather
- What should be tested next, on the Sunny branch?
  - i.e., should we test Temperature, Windy, or Humidity?

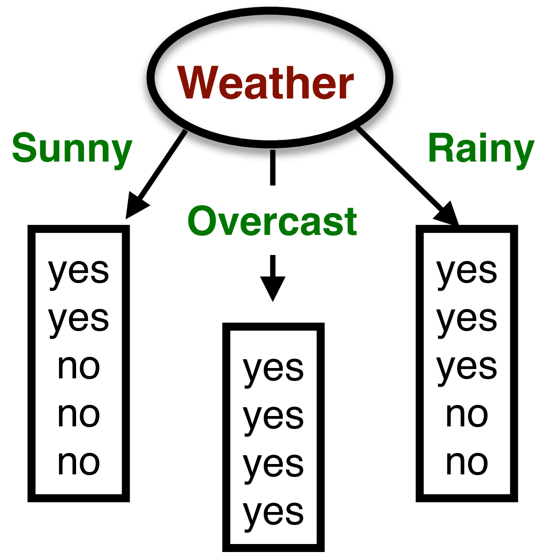


Gain: 0.571 bits

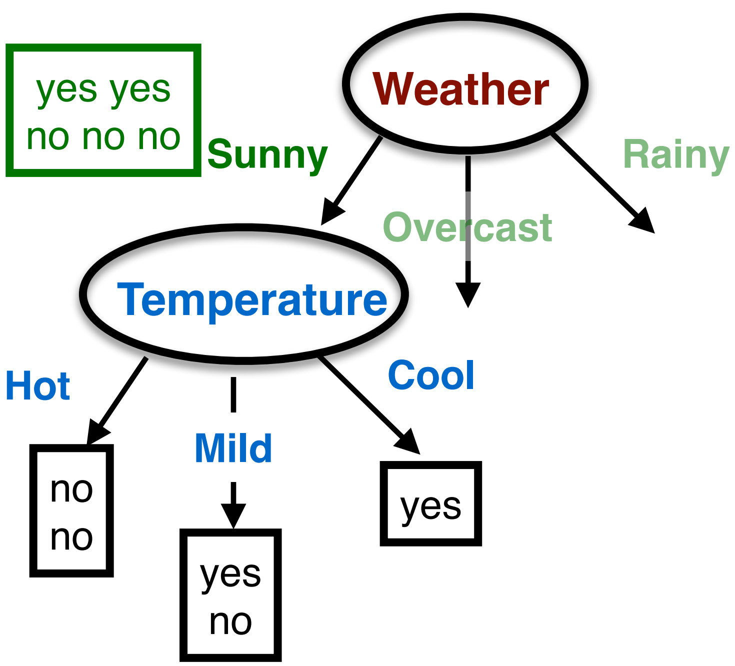


Gain: 0.020 bits

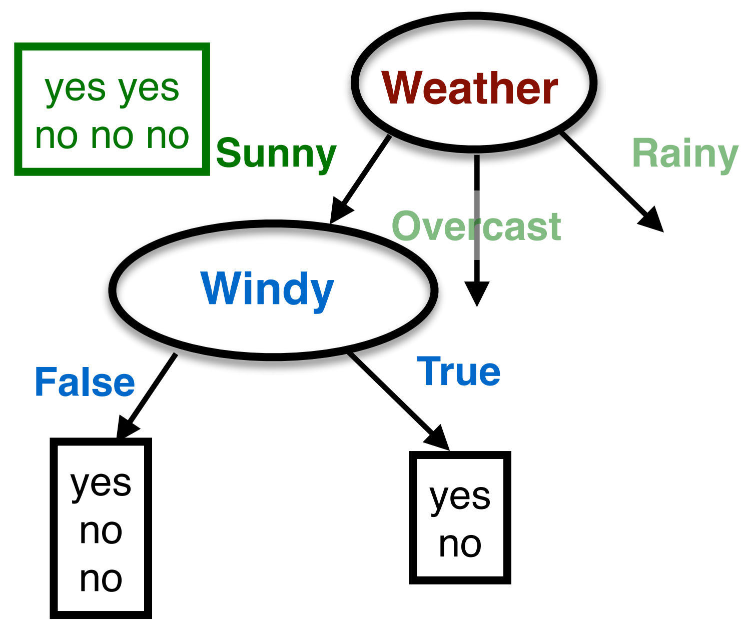
Selecting an Attribute to Test



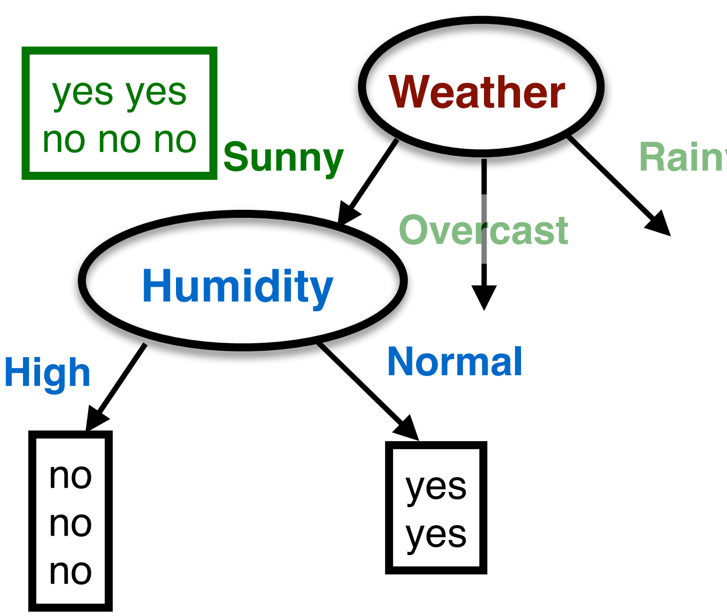
- We have decided that the 1st attribute to test is Weather
- What should be tested next, on the Sunny branch?
  - i.e., should we test Temperature, Windy, or Humidity?



Gain: 0.571 bits

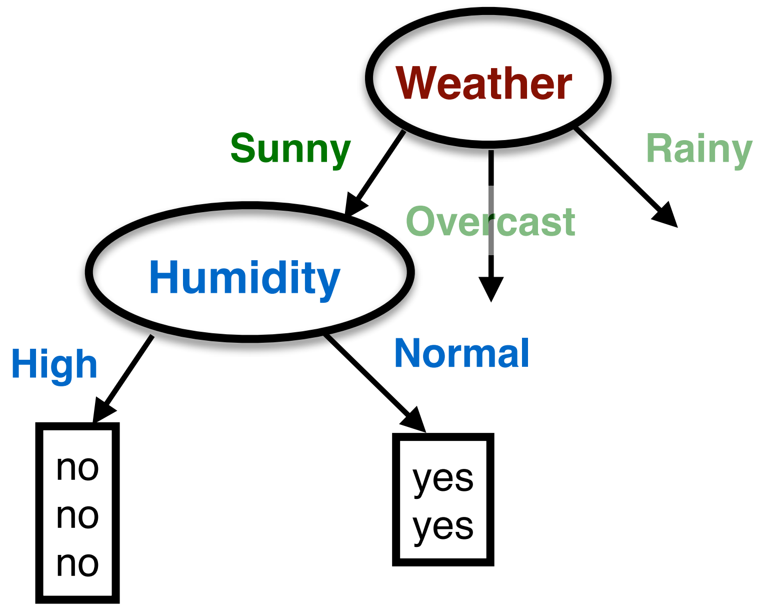


Gain: 0.020 bits

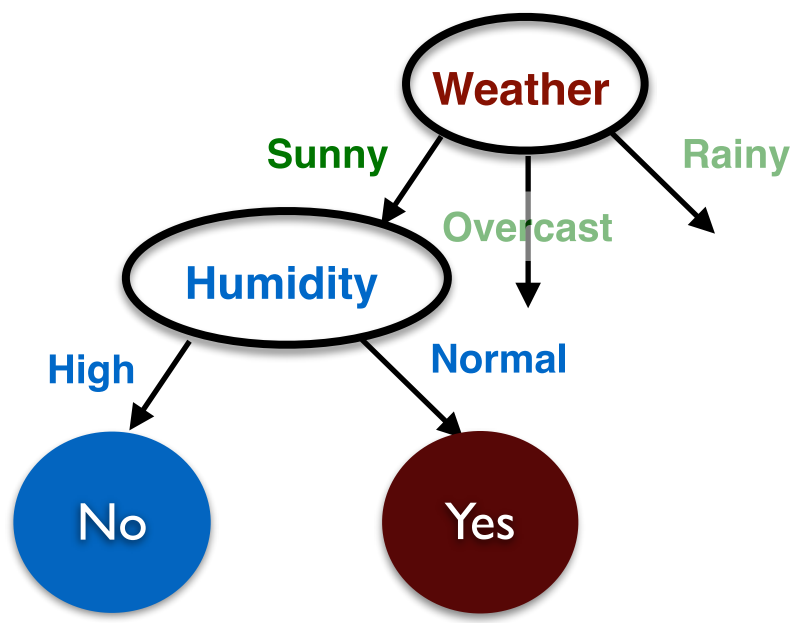


Gain: 0.971 bits

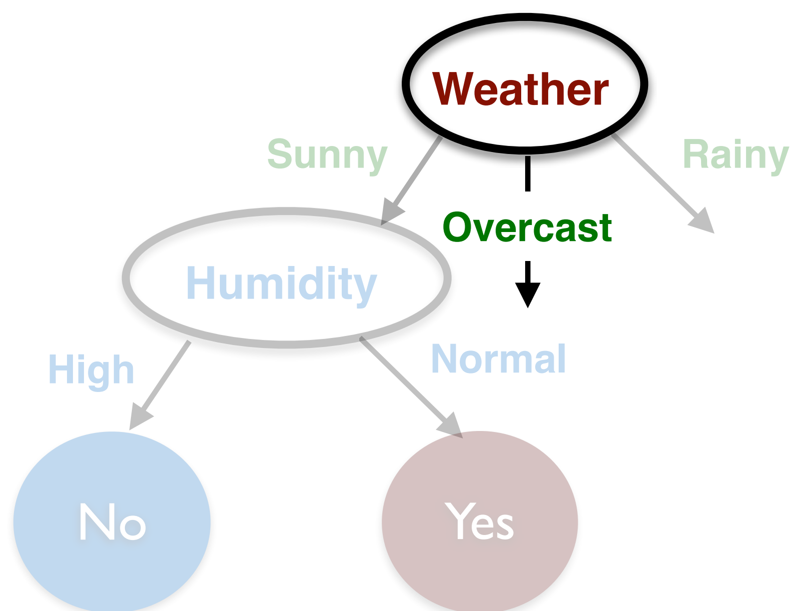
Selecting an Attribute to Test



Selecting an Attribute to Test



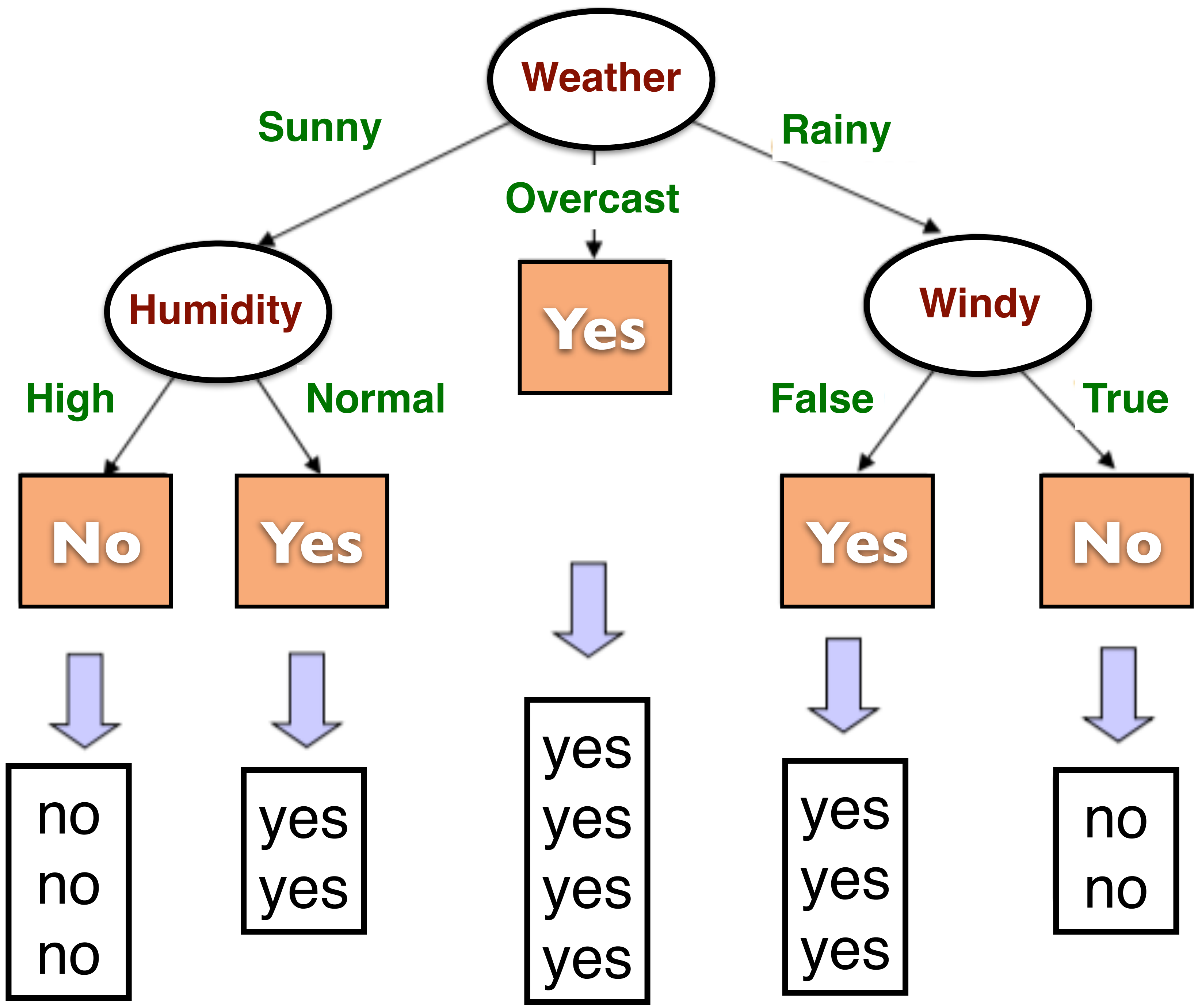
Selecting an Attribute to Test



- What should be tested next, on the Overcast branch?
  - i.e., should we test Temperature, Windy, or Humidity?

Repeat the same process, recursively…

Learned Decision Tree



Criteria for Selecting an Attribute to Test

- We have discussed one possible criterion for selecting which attribute to test
  - **Information Gain**

- Many other criteria have been proposed — each with different properties
- Intuitively:
  - A split that keeps the same proportion of classes in each partition is **useless**
  - A split where the instances in each partition have the same class is **useful**!

Criteria for Selecting an Attribute to Test

- We have discussed <u>one</u> possible criterion for selecting which attribute to test
  - **Information Gain**

- Many other criteria have been proposed — each with different properties
- Intuitively:
  - A split that keeps the same proportion of classes in each partition is **useless**
  - A split where the instances in each partition have the same class is **useful**!

- Main criteria for selecting which attribute to test:
  - **Information Gain** - <u>ID3</u> Algorithm (Quilan, 1987)
  - **Information Gain Ratio** - <u>C4.5</u> Algorithm (Quilan, 1988)
  - **Gini Impurity** - <u>CART</u> Algorithm (Breiman, 1984)

Criteria for Selecting an Attribute to Test

- We have discussed <u>one</u> possible criterion for selecting which attribute to test
  - **Information Gain**

- Many other criteria have been proposed — each with different properties
- Intuitively:
  - A split that keeps the same proportion of classes in each partition is **useless**
  - A split where the instances in each partition have the same class is **useful**!

All algorithms are based on the same underlying tree-learning strategy
Differ with respect to the criterion used to select which attribute to test at each point

- **Information Gain Ratio** - <u>C4.5</u> Algorithm (Quilan, 1988)
- **Gini Impurity** - <u>CART</u> Algorithm (Breiman, 1984)

Function: decision_tree(D, L)

Pseudocode

- Simply describes the learning process illustrated earlier through examples
- Specifies how to handle edge cases
  - E.g., when a split results in a branch with zero examples
    - If splitting on Age, the branch where Age = Young has no instances

You can use this as a reference
when implementing the algorithm

A Decision Tree Learning algorithm

Function: decision_tree(D, L)

Input: A dataset $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ with n training instances
A list, L, of attributes that can still be tested

- Create a new node, N

- If all instances in D belong to the same class, y
  Define node N as a leaf node labeled with y and return it
- If there are no more attributes that can be tested (i.e., if $L = \emptyset$)
  Define node N as a leaf node labeled with the majority class in D, and return it

// stopping
// criteria

- Let A be the best attribute to split the dataset D **// Select splitting attribute according to some criterion**
- Define node N as a decision node that tests attribute A

- Main criteria for selecting which attribute to test:
  - **Information Gain** - ID3 Algorithm (Quilan, 1987)
  - **Information Gain Ratio** - C4.5 Algorithm (Quilan, 1988)
  - **Gini Impurity** - CART Algorithm (Breiman, 1984)

A Decision Tree Learning algorithm

Function: decision_tree(D, L)

Input: A dataset $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ with n training instances
A list, L, of attributes that can still be tested

- Create a new node, N

- If all instances in D belong to the same class, y
  Define node N as a leaf node labeled with y and return it
- If there are no more attributes that can be tested (i.e., if $L = \emptyset$)
  Define node N as a leaf node labeled with the majority class in D, and return it

// stopping
// criteria

- Let A be the best attribute to split the dataset D **// Select splitting attribute according to some criterion**
- Define node N as a decision node that tests attribute A
- Remove A from the list of attributes that can still be tested: $L := L - \{A\}$

- Let V be a list with all different values of attribute A considering the instances in dataset D
- For each attribute value $v \in V$:
  - Let D_v be the partition of D containing all instances whose attribute $A = v$
  - If D_v is empty
    Let T_v be a leaf node labeled with the majority class in D
  - Else
    Let T_v be a sub-tree responsible for classifying the instances in D_v: $T_v :=$ **decision_tree(** D_v, L **)**
  - Create an edge from node N to the root of T_v, where the edge is labeled with attribute value v

// creates
// sub-trees

- Return N

Information Gain

- This is the criterion discussed earlier → results in a method known as **ID3**
- Intuitively, it selects the attribute A that maximizes the difference between:
  - The entropy of the original dataset D (before splitting it based on A)
  - The average entropy of the resulting partitions if we split dataset D based on A

Formally:

The following equations simply describe mathematically the quantities we computed earlier through examples

You can use these as a reference when implementing the algorithm

Information Gain

- This is the criterion discussed earlier → results in a method known as **ID3**
- Intuitively, it selects the attribute A that maximizes the difference between:
  - The entropy of the original dataset D (before splitting it based on A)
  - The average entropy of the resulting partitions if we split dataset D based on A

Formally:

- Let p_i be the probability that the label i occurs in instances in a dataset D
- Let $I(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$ be the entropy of an arbitrary dataset D, where m is the number of classes/labels
- Assume that the attribute A can take up v values

  (that is, if we split D based on attribute A, we will end up with v partitions)
- Let $\text{Info}_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} I(D_j)$ be the average entropy of the partitions resulting from splitting D based on A
- Let $\text{Gain}_A(D) = I(D) - \text{Info}_A(D)$ be the **Information Gain** resulting from splitting based on attribute A
- At each step, the algorithm splits the instances based on the attribute A with **highest Information Gain**

Information Gain

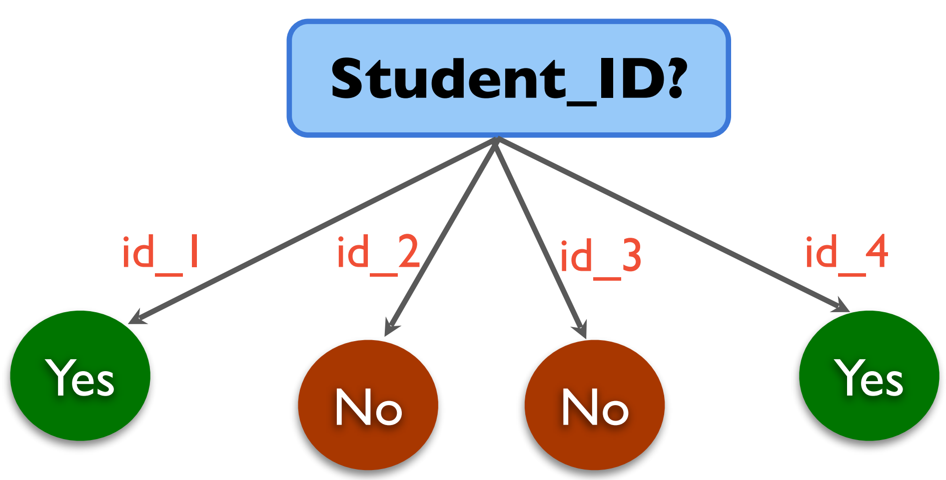
- This is the criterion discussed earlier → results in a method known as **ID3**
- Intuitively, it selects the attribute A that maximizes the difference between:
    - The entropy of the original dataset D (before splitting it based on A)
    - The average entropy of the resulting partitions if we split dataset D based on A

Information Gain

- This is the criterion discussed earlier → results in a method known as **ID3**
- Intuitively, it selects the attribute A that maximizes the difference between:
  - The entropy of the original dataset D (before splitting it based on A)
  - The average entropy of the resulting partitions if we split dataset D based on A

- Often results in a decision tree that is not necessarily the "simplest" one
- Intuitively, it often chooses attributes with many possible values (like Student_ID, Name, etc)

Student_ID	**Student**	**Age**	**Credit_Score**	**Will_Buy_Computer**
id_1	Yes	Young	Regular	Yes
id_2	Yes	Middle Age	Excellent	No
id_3	No	Young	Excellent	No
id_4	No	Older Adult	Regular	Yes



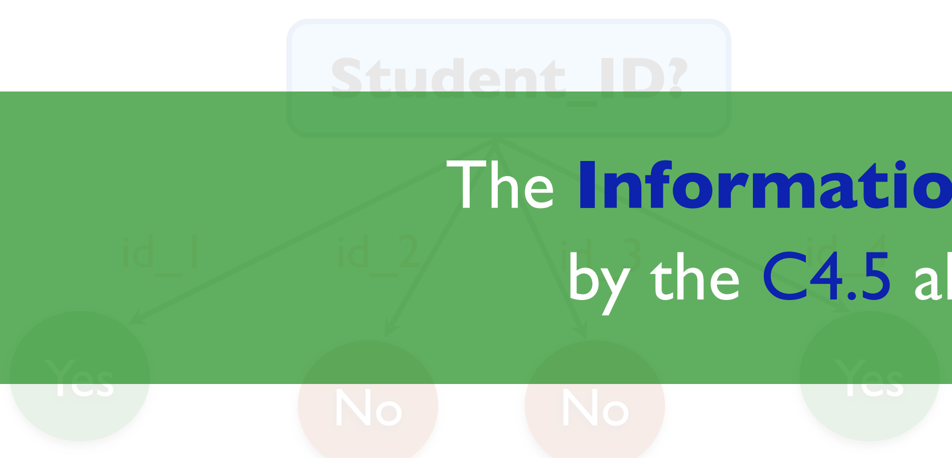
- Perfect split!
- With just one test, can "predict" the class perfectly
- But it is clearly overfitting ("memorizing" the dataset)

Information Gain

- This is the criterion discussed earlier → results in a method known as **ID3**
- Intuitively, it selects the attribute A that maximizes the difference between:
  - The entropy of the original dataset D (before splitting it based on A)
  - The average entropy of the resulting partitions if we split dataset D based on A

- Often results in a decision tree that is not necessarily the "simplest" one
- Intuitively, it often chooses attributes with many possible values (like Student_ID, Name, etc)

Student_ID	Student	Age	Credit_Score	Will_Buy_Computer
id_1	Yes	Young	Regular	Yes
id_2	Yes	Middle Age	Excellent	No
id_3	No	Young	Excellent	No
id_4	No	Older Adult	Regular	Yes



- Perfect split!

The **Information Gain Ratio** criterion, implemented by the C4.5 algorithm, tries to mitigate this issue

Criteria for Selecting an Attribute to Test

- We have discussed <u>one</u> possible criterion for selecting which attribute to test
    - **Information Gain**

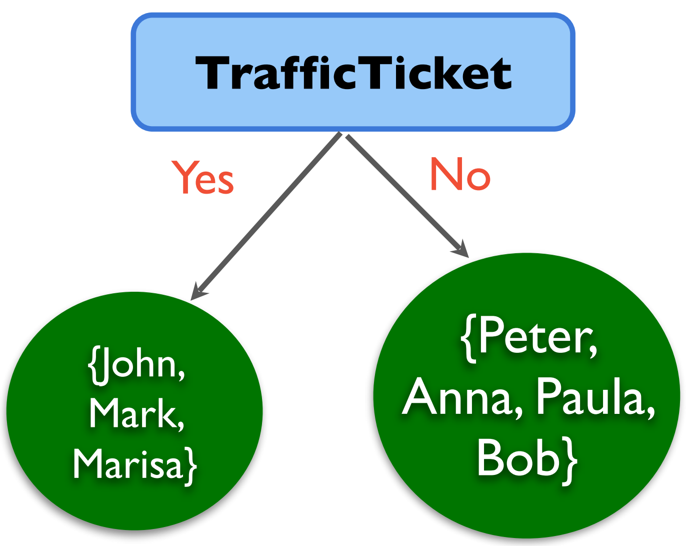
- Many other criteria have been proposed — each with different properties
- Intuitively:
    - A split that keeps the same proportion of classes in each partition is **useless**
    - A split where the instances in each partition have the same class is **useful**!

- Main criteria for selecting which attribute to test:
    - **Information Gain** - <u>ID3</u> Algorithm (Quilan, 1987)
    - **Information Gain Ratio** - <u>C4.5</u> Algorithm (Quilan, 1988)
    - **Gini Impurity** - <u>CART</u> Algorithm (Breiman, 1984)

Information Gain Ratio

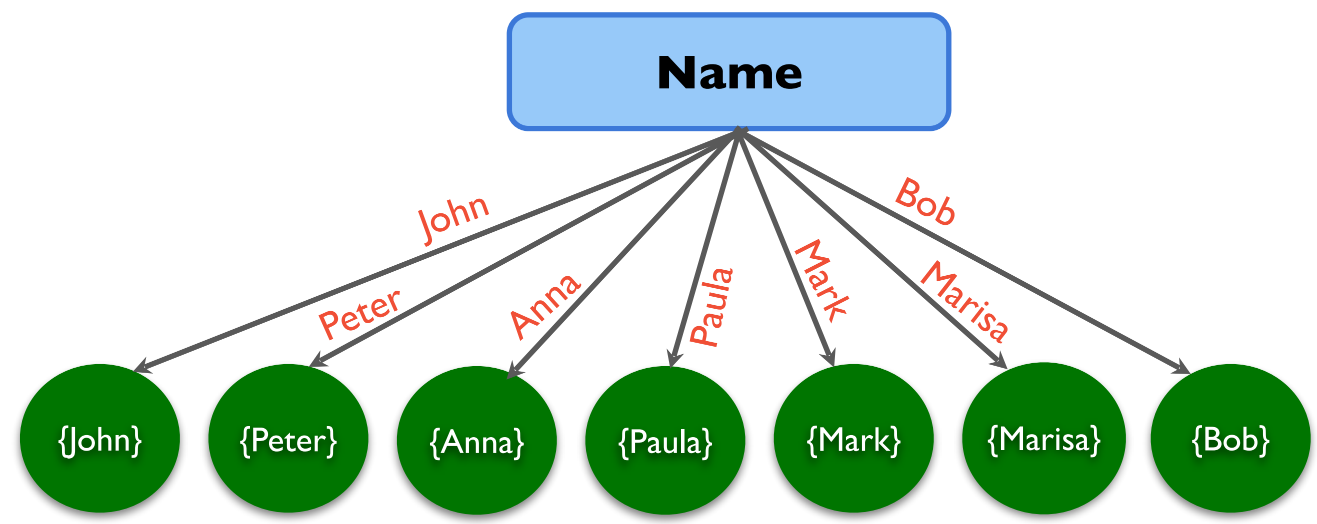
Name	*Age*	*Gender*	*TrafficTicket*	Class: High-Risk Driver
John	43	M	Yes	**High Risk**
Peter	18	M	No	**Low Risk**
Anna	35	F	No	**Low Risk**
Paula	19	F	No	**Low Risk**
Mark	90	M	Yes	**High Risk**
Marisa	19	F	Yes	**Low Risk**
Bob	30	M	No	**Low Risk**

- “Adjusts" Information Gain criterion to lessen the bias towards attributes that create many branches
- Intuition:



$$|D_1| = 3 \qquad |D_2| = 4$$

$$\Pr(D_1) = \frac{3}{7} \qquad \Pr(D_2) = \frac{4}{7}$$

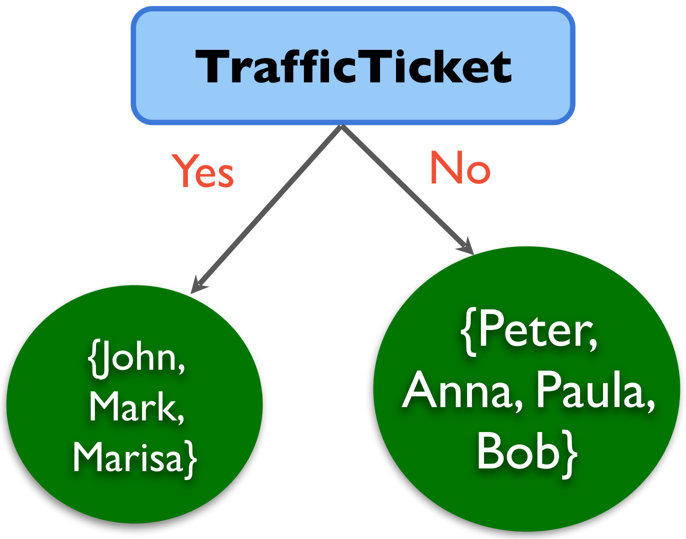


$$|D_1| = 1 \quad |D_2| = 1 \quad |D_3| = 1 \quad |D_4| = 1 \quad |D_5| = 1 \quad |D_6| = 1 \quad |D_7| = 1$$

$$\Pr(D_1) = \frac{1}{7} \quad \Pr(D_2) = \frac{1}{7} \quad \Pr(D_3) = \frac{1}{7} \quad \Pr(D_4) = \frac{1}{7} \quad \Pr(D_5) = \frac{1}{7} \quad \Pr(D_6) = \frac{1}{7} \quad \Pr(D_7) = \frac{1}{7}$$

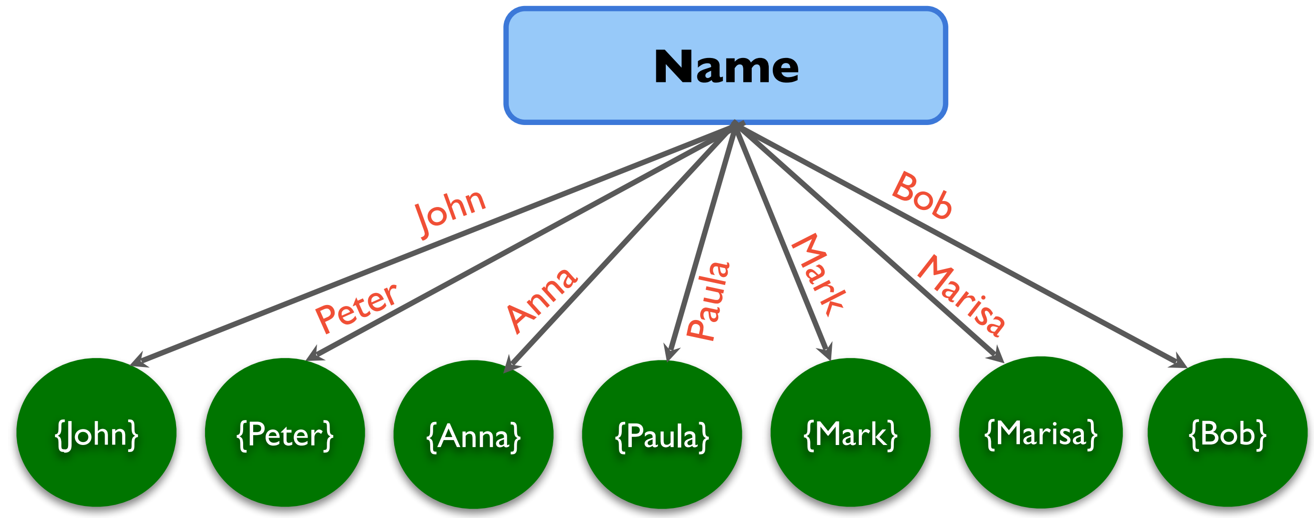
Information Gain Ratio

- "Adjusts" Information Gain criterion to lessen the bias towards attributes that create many branches
- Intuition:



$|D_1| = 3$ $\quad \Pr(D_1) = \frac{3}{7}$

$|D_2| = 4$ $\quad \Pr(D_2) = \frac{4}{7}$



$|D_1| = 1$ $\quad \Pr(D_1) = \frac{1}{7}$

$|D_2| = 1$ $\quad \Pr(D_2) = \frac{1}{7}$

$|D_3| = 1$ $\quad \Pr(D_3) = \frac{1}{7}$

$|D_4| = 1$ $\quad \Pr(D_4) = \frac{1}{7}$

$|D_5| = 1$ $\quad \Pr(D_5) = \frac{1}{7}$

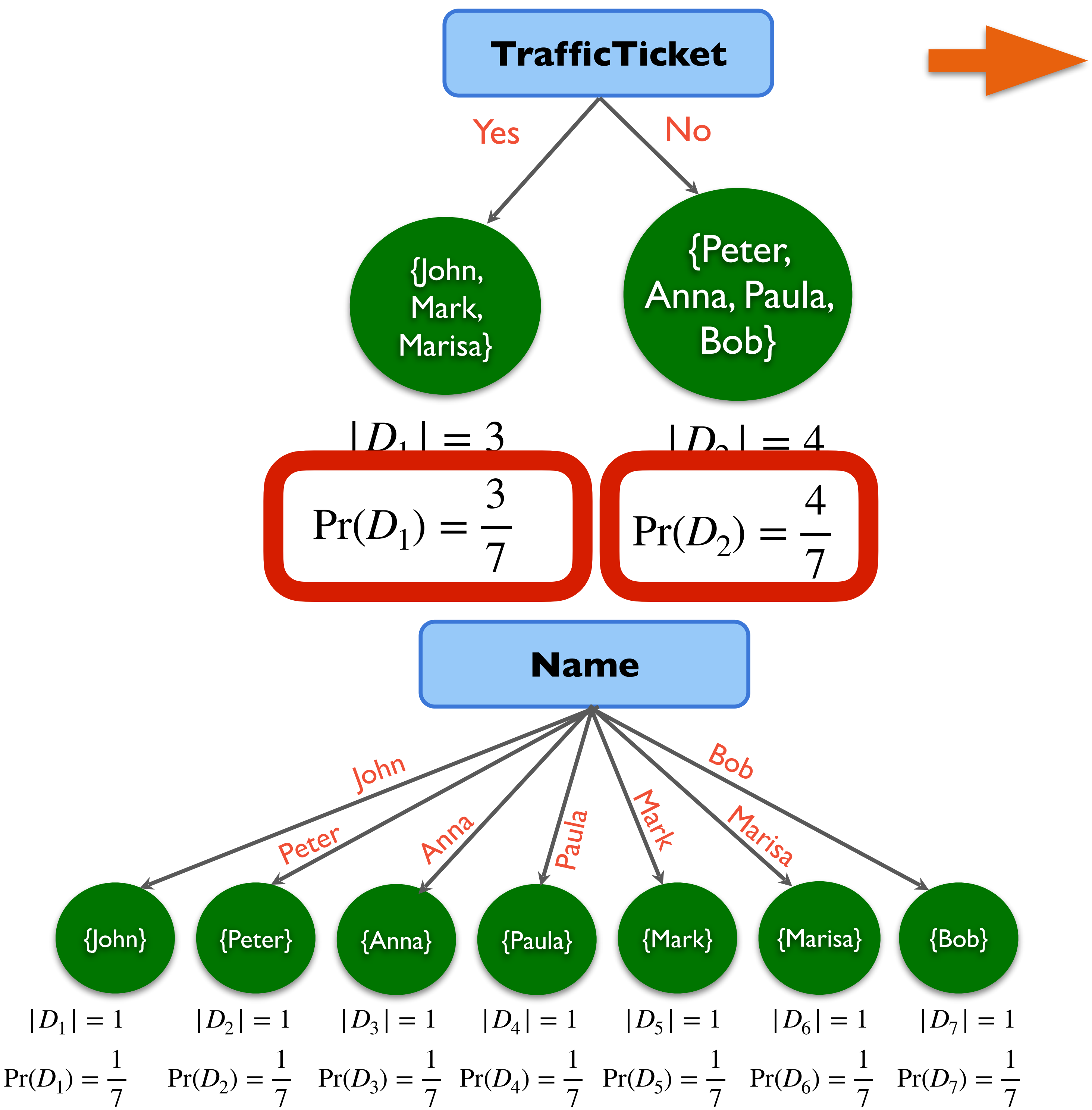
$|D_6| = 1$ $\quad \Pr(D_6) = \frac{1}{7}$

$|D_7| = 1$ $\quad \Pr(D_7) = \frac{1}{7}$

- If there are lots of branches (e.g., if we split by Name, there are as many branches as attribute values!)
  - Then these probabilities will be very similar/homogenous
- How to quantify how "homogeneous" these quantities are?
  - We've seen something like this before… Entropy!

Information Gain Ratio

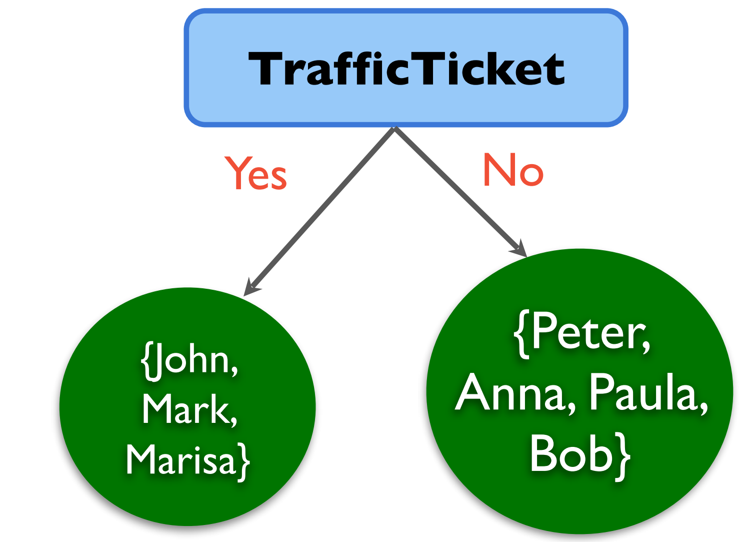
- How to quantify how "homogeneous" these quantities are?
    - We've seen something like this before… Entropy!



$$-\frac{3}{7}\log_2\left(\frac{3}{7}\right) - \frac{4}{7}\log_2\left(\frac{4}{7}\right)$$

Information Gain Ratio

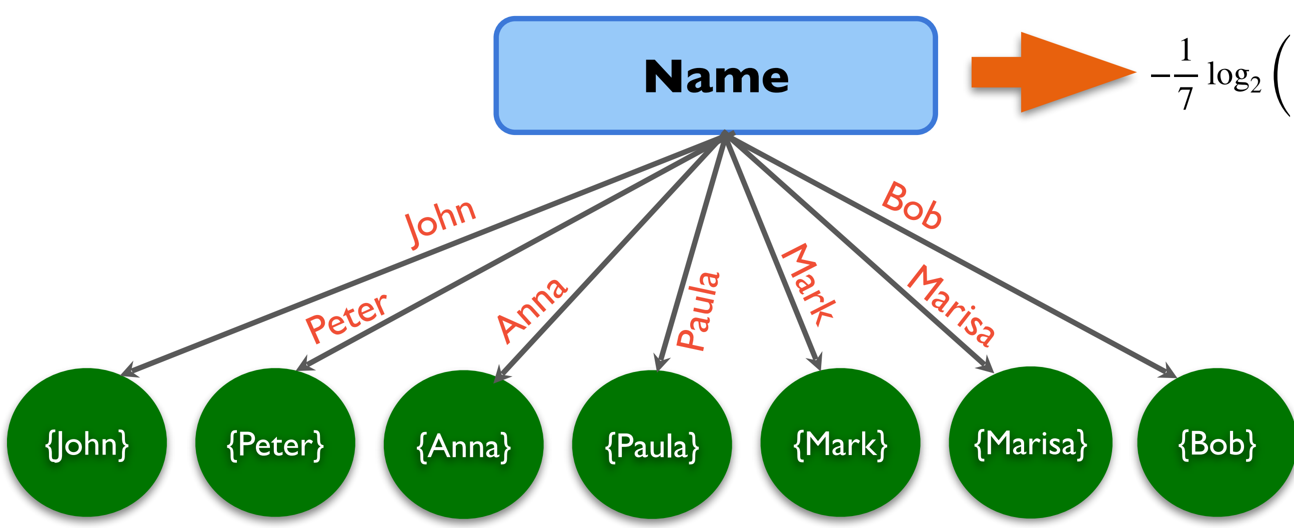
- How to quantify how "homogeneous" these quantities are?
  - We've seen something like this before… Entropy!



$$-\frac{3}{7}\log_2\left(\frac{3}{7}\right)-\frac{4}{7}\log_2\left(\frac{4}{7}\right)$$

$|D_1| = 3$ $\qquad$ $|D_2| = 4$

$\Pr(D_1) = \frac{3}{7}$ $\qquad$ $\Pr(D_2) = \frac{4}{7}$



$$-\frac{1}{7}\log_2\left(\frac{1}{7}\right)-\frac{1}{7}\log_2\left(\frac{1}{7}\right)-\frac{1}{7}\log_2\left(\frac{1}{7}\right)-\frac{1}{7}\log_2\left(\frac{1}{7}\right)-\frac{1}{7}\log_2\left(\frac{1}{7}\right)-\frac{1}{7}\log_2\left(\frac{1}{7}\right)-\frac{1}{7}\log_2\left(\frac{1}{7}\right)$$

$|D_1| = 1$ $|D_2| = 1$ $|D_3| = 1$ $|D_4| = 1$ $|D_5| = 1$ $|D_6| = 1$ $|D_7| = 1$

$\boxed{\Pr(D_1) = \frac{1}{7}}$ $\boxed{\Pr(D_2) = \frac{1}{7}}$ $\boxed{\Pr(D_3) = \frac{1}{7}}$ $\boxed{\Pr(D_4) = \frac{1}{7}}$ $\boxed{\Pr(D_5) = \frac{1}{7}}$ $\boxed{\Pr(D_6) = \frac{1}{7}}$ $\boxed{\Pr(D_7) = \frac{1}{7}}$

Information Gain Ratio

- How to quantify how "homogeneous" these quantities are?
  - We've seen something like this before… Entropy! → which is called, in this context, **Split_Info**

$|D_1| = 3$ $\quad$ $|D_2| = 4$

$\Pr(D_1) = \frac{3}{7}$ $\quad$ $\Pr(D_2) = \frac{4}{7}$

$$-\frac{3}{7}\log_2\left(\frac{3}{7}\right) - \frac{4}{7}\log_2\left(\frac{4}{7}\right)$$

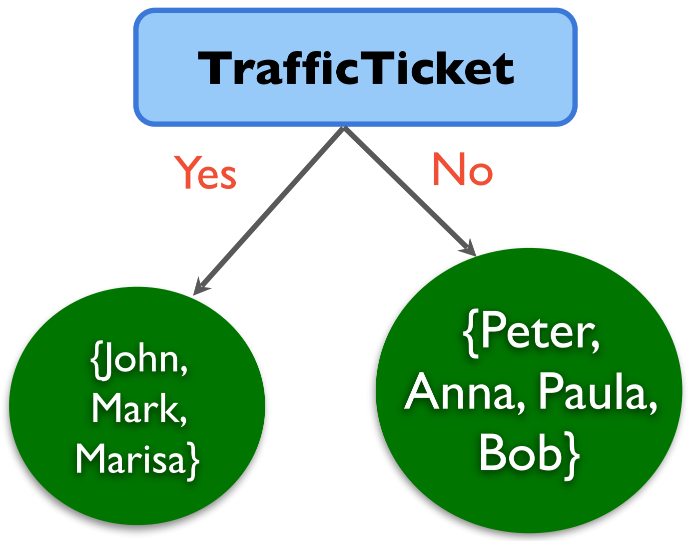
$$-\frac{1}{7}\log_2\left(\frac{1}{7}\right) - \frac{1}{7}\log_2\left(\frac{1}{7}\right) - \frac{1}{7}\log_2\left(\frac{1}{7}\right) - \frac{1}{7}\log_2\left(\frac{1}{7}\right) - \frac{1}{7}\log_2\left(\frac{1}{7}\right) - \frac{1}{7}\log_2\left(\frac{1}{7}\right) - \frac{1}{7}\log_2\left(\frac{1}{7}\right)$$

$|D_1| = 1$ $\quad$ $|D_2| = 1$ $\quad$ $|D_3| = 1$ $\quad$ $|D_4| = 1$ $\quad$ $|D_5| = 1$ $\quad$ $|D_6| = 1$ $\quad$ $|D_7| = 1$

$\Pr(D_1) = \frac{1}{7}$ $\quad$ $\Pr(D_2) = \frac{1}{7}$ $\quad$ $\Pr(D_3) = \frac{1}{7}$ $\quad$ $\Pr(D_4) = \frac{1}{7}$ $\quad$ $\Pr(D_5) = \frac{1}{7}$ $\quad$ $\Pr(D_6) = \frac{1}{7}$ $\quad$ $\Pr(D_7) = \frac{1}{7}$

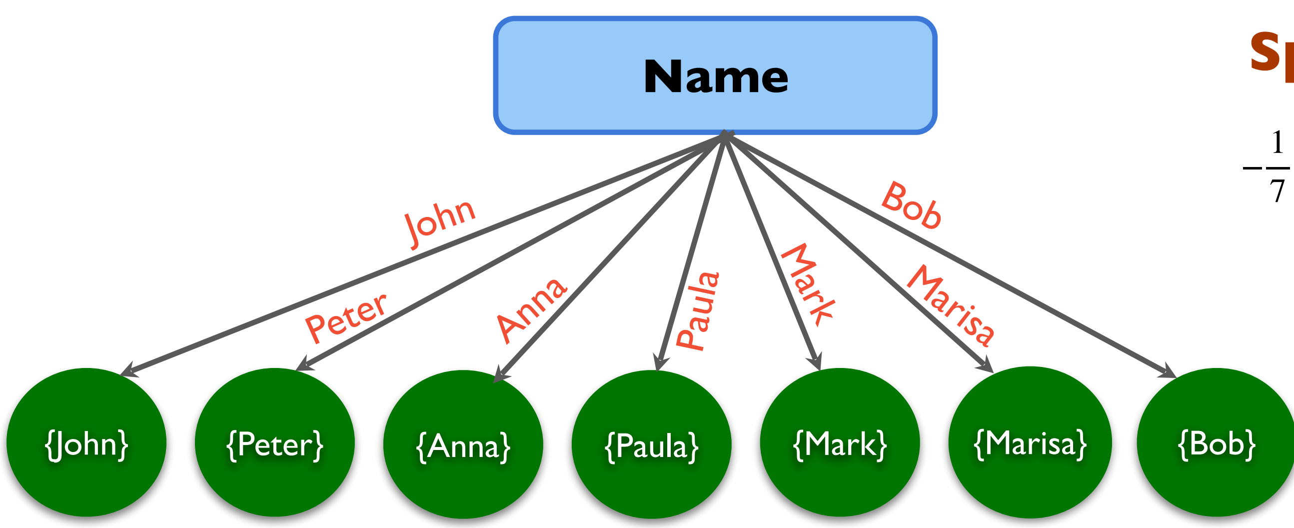
Information Gain Ratio

- How to quantify how "homogeneous" these quantities are?
  - We've seen something like this before… Entropy! → which is called, in this context, **Split_Info**



Split_Info(TrafficTicket) = $-\frac{3}{7}\log_2\left(\frac{3}{7}\right)-\frac{4}{7}\log_2\left(\frac{4}{7}\right)$

$$= 0.98$$



Split_Info(Name) =

$$-\frac{1}{7}\log_2\left(\frac{1}{7}\right)-\frac{1}{7}\log_2\left(\frac{1}{7}\right)-\frac{1}{7}\log_2\left(\frac{1}{7}\right)-\frac{1}{7}\log_2\left(\frac{1}{7}\right)-\frac{1}{7}\log_2\left(\frac{1}{7}\right)-\frac{1}{7}\log_2\left(\frac{1}{7}\right)-\frac{1}{7}\log_2\left(\frac{1}{7}\right)$$

$$= 2.8$$

Information Gain Ratio

- How to quantify how "homogeneous" these quantities are?
  - We've seen something like this before… Entropy! → which is called, in this context, **Split_Info**

Name	*Age*	*Gender*	*TrafficTicket*	**Class: High-Risk Driver**
John	43	M	Yes	**High Risk**
Peter	18	M	No	**Low Risk**
Anna	35	F	No	**Low Risk**
Paula	19	F	No	**Low Risk**
Mark	90	M	Yes	**High Risk**
Marisa	19	F	Yes	**Low Risk**
Bob	30	M	No	**Low Risk**

Split_Info(TrafficTicket) = $-\frac{3}{7}\log_2\left(\frac{3}{7}\right)-\frac{4}{7}\log_2\left(\frac{4}{7}\right) = 0.98$

Split_Info(Name) = $-\frac{1}{7}\log_2\left(\frac{1}{7}\right)-\frac{1}{7}\log_2\left(\frac{1}{7}\right)-\frac{1}{7}\log_2\left(\frac{1}{7}\right)-\frac{1}{7}\log_2\left(\frac{1}{7}\right)-\frac{1}{7}\log_2\left(\frac{1}{7}\right)-\frac{1}{7}\log_2\left(\frac{1}{7}\right)-\frac{1}{7}\log_2\left(\frac{1}{7}\right) = 2.8$

The larger value of Split_Info for Name suggests that this is a worse split than TrafficTicket

Information Gain Ratio

- The **Information Gain Ratio** combines two "measures" of how good a split (based on attribute A) is
  - Its **Information Gain**, as previously defined → $\text{Gain}_A(D)$ → higher is better
  - Its **Split_Info** → $\text{Split\_Info}(A)$ → higher is worse

$$\text{Gain\_Ratio}(A, D) = \frac{\text{Gain}_A(D)}{\text{Split\_Info}(A)}$$

Name	Age	Gender	TrafficTicket	Class: High-Risk Driver
John	43	M	Yes	High Risk
Peter	18	M	No	Low Risk
Anna	35	F	No	Low Risk
Paula	19	F	No	Low Risk
Mark	90	M	Yes	High Risk
Marisa	19	F	Yes	Low Risk
Bob	30	M	No	Low Risk

Split_Info(TrafficTicket) = $-\frac{3}{7}\log_2\left(\frac{3}{7}\right) - \frac{4}{7}\log_2\left(\frac{4}{7}\right) = 0.98$

Split_Info(Name) = $-\frac{1}{7}\log_2\left(\frac{1}{7}\right) - \frac{1}{7}\log_2\left(\frac{1}{7}\right) - \frac{1}{7}\log_2\left(\frac{1}{7}\right) - \frac{1}{7}\log_2\left(\frac{1}{7}\right) - \frac{1}{7}\log_2\left(\frac{1}{7}\right) - \frac{1}{7}\log_2\left(\frac{1}{7}\right) - \frac{1}{7}\log_2\left(\frac{1}{7}\right) = 2.8$

The larger value of Split_Info for Name suggests that this is a worse split than TrafficTicket

Information Gain Ratio

- The **Information Gain Ratio** combines two "measures" of how good a split (based on attribute A) is
  - Its **Information Gain**, as previously defined → $\text{Gain}_A(D)$ → higher is better
  - Its **Split_Info** → $\text{Split\_Info}(A)$ → higher is worse

$$\text{Gain\_Ratio}(A, D) = \frac{\text{Gain}_A(D)}{\text{Split\_Info}(A)}$$

Name	Age	Gender	TrafficTicket	Class: High-Risk Driver
John	43	M	Yes	High Risk
Peter	18	M	No	Low Risk
Anna	35	F	No	Low Risk
Paula	19	F	No	Low Risk
Mark	90	M	Yes	High Risk
Marisa	19	F	Yes	Low Risk
Bob	30	M	No	Low Risk

$\text{Gain}_{\text{TrafficTicket}}(D) = 0.466$

$\text{Split\_Info}(\text{TrafficTicket}) = 0.98$

$\text{Gain}_{\text{Name}}(D) = 0.86$

$\text{Split\_Info}(\text{Name}) = 2.8$

In terms of **Information Gain** only, Name looks like **a good split**

However, its **Split_Info** suggests that Name it's **a bad split**

Let's combine these into a single score that takes both into account

Gain_Ratio!

Information Gain Ratio

- The **Information Gain Ratio** combines two "measures" of how good a split (based on attribute A) is
  - Its **Information Gain**, as previously defined → $\text{Gain}_A(D)$ → higher is better
  - Its **Split_Info** → $\text{Split\_Info}(A)$ → higher is worse

$$\text{Gain\_Ratio}(A, D) = \frac{\text{Gain}_A(D)}{\text{Split\_Info}(A)}$$

In terms of **Information Gain** only, Name looks like **a good split**
However, its **Split_Info** suggests that Name it's **a bad split**

↓

Let's combine these into a single score that takes both into account
Gain_Ratio!

Gain$_{\text{TrafficTicket}}$(D) $= 0.466$

Split_Info(TrafficTicket) $= 0.98$

$$\text{Gain\_Ratio}(\text{TrafficTicket}, D) = \frac{0.466}{0.98} = 0.475$$

Gain$_{\text{Name}}$(D) $= 0.86$

Split_Info(Name) $= 2.8$

$$\text{Gain\_Ratio}(\text{Name}, D) = \frac{0.86}{2.8} = 0.307$$

Information Gain Ratio

- The **Information Gain Ratio** combines two "measures" of how good a split (based on attribute A) is
  - Its **Information Gain**, as previously defined → $\text{Gain}_A(D)$ → higher is better
  - Its **Split_Info** → $\text{Split\_Info}(A)$ → higher is worse

$$\text{Gain\_Ratio}(A, D) = \frac{\text{Gain}_A(D)}{\text{Split\_Info}(A)}$$

In terms of **Information Gain** only, Name looks like **a good split**

However, its **Split_Info** suggests that Name it's **a bad split**

Let's combine these into a single score that takes both into account

Gain_Ratio!

$$\text{Gain\_Ratio}(\text{TrafficTicket}, D) = \frac{0.466}{0.98} = 0.475$$

This criterion "understands" that splitting based on TrafficTicket is better than splitting based on Name

$$\text{Gain\_Ratio}(\text{Name}, D) = \frac{0.86}{2.8} = 0.307$$

Criteria for Selecting an Attribute to Test

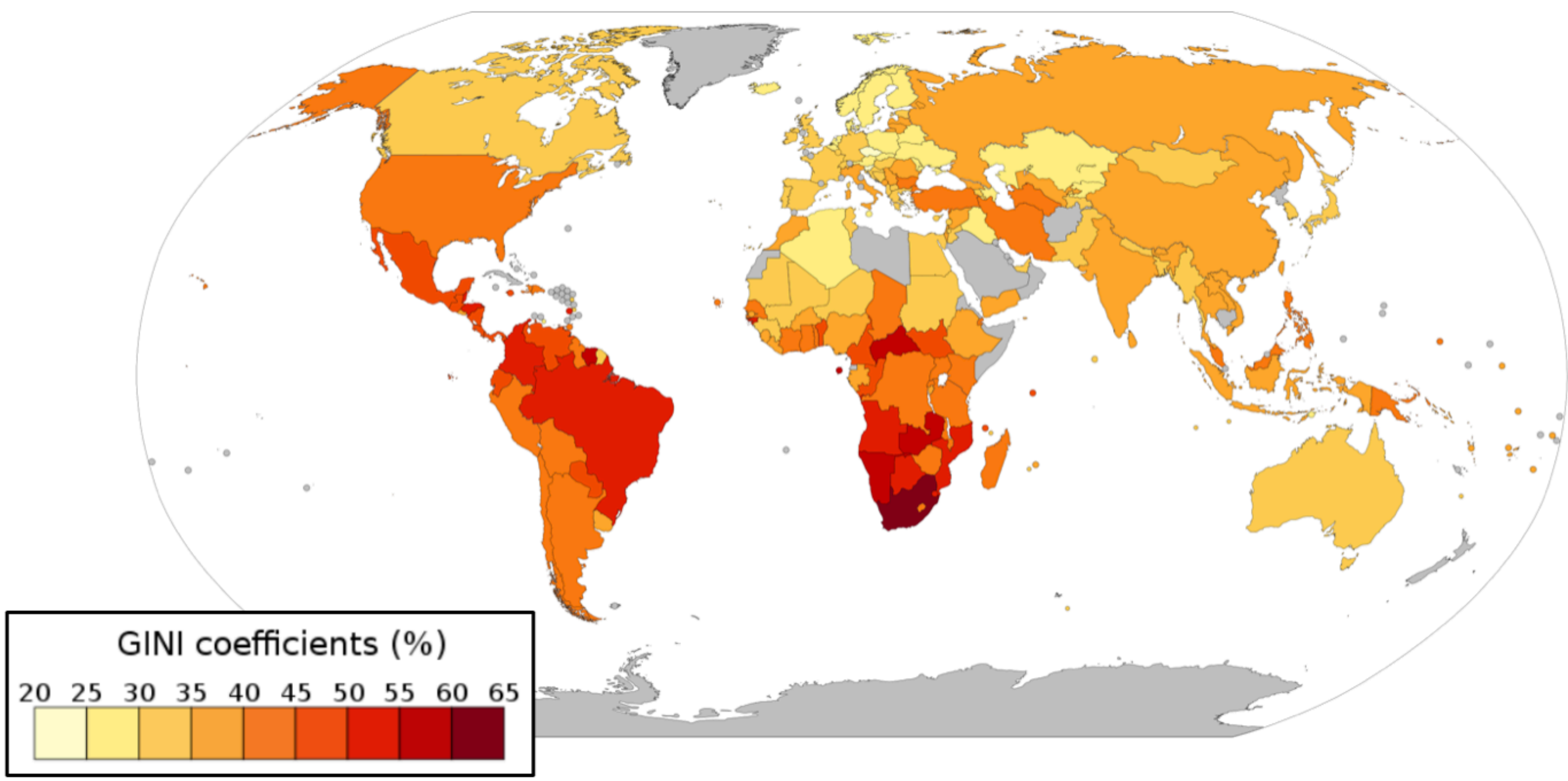
- We have discussed one possible criterion for selecting which attribute to test
  - **Information Gain**

- Many other criteria have been proposed — each with different properties
- Intuitively:
  - A split that keeps the same proportion of classes in each partition is **useless**
  - A split where the instances in each partition have the same class is **useful**!

- Main criteria for selecting which attribute to test:
  - **Information Gain** - ID3 Algorithm (Quilan, 1987)
  - **Information Gain Ratio** - C4.5 Algorithm (Quilan, 1988)
  - **Gini Impurity** - CART Algorithm (Breiman, 1984)

Gini Criterion

- Originally proposed to quantify how uneven income is across a population



- Gini coefficient → how uneven income/wealth distribution across a population is
  - Gini = 1 → very uneven income/wealth distribution across a population
  - Gini = 0 → very even income/wealth distribution across a population

Gini Criterion

- Gini coefficient → how uneven income/wealth distribution across a population is
- In the context of decision trees
  - how uneven (or non-homogeneous) are the classes after a split

Gini Criterion

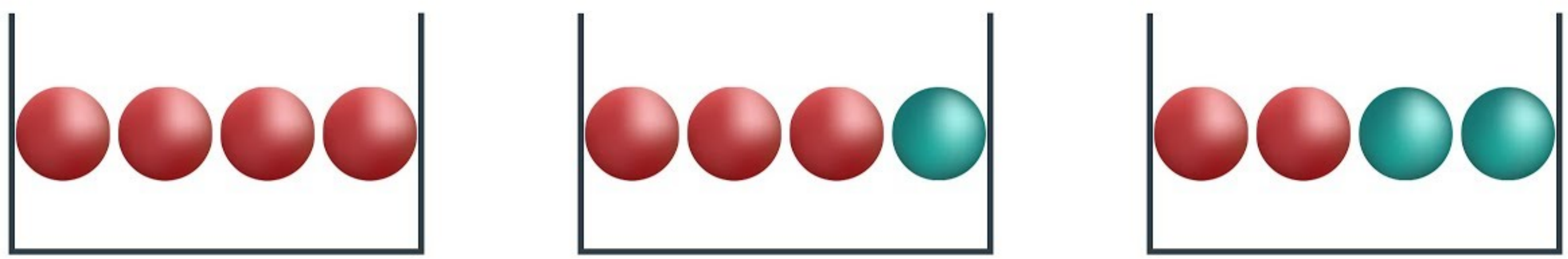
- Gini coefficient → how uneven income/wealth distribution across a population is
- In the context of decision trees
  - how uneven (or non-homogeneous) are the classes after a split

Gini(D) = 1 - (Pr( )² + Pr( )²)

Let's suppose we test Age, and the instances associated with Age=Young look like this


Will repay loan


Will not repay loan



Even	**"Medium"**	**Uneven**
Pr( ) = 1	Pr( ) = 3/4	Pr( ) = 2/4
Pr( ) = 0	Pr( ) = 1/4	Pr( ) = 2/4

Gini $\Rightarrow$

$$1-(1^2+0^2) = 0$$

$$1-\left((3/4)^2+(1/4)^2\right) = 0.375$$

$$1-\left((2/4)^2+(2/4)^2\right) = 0.5$$

Gini Criterion

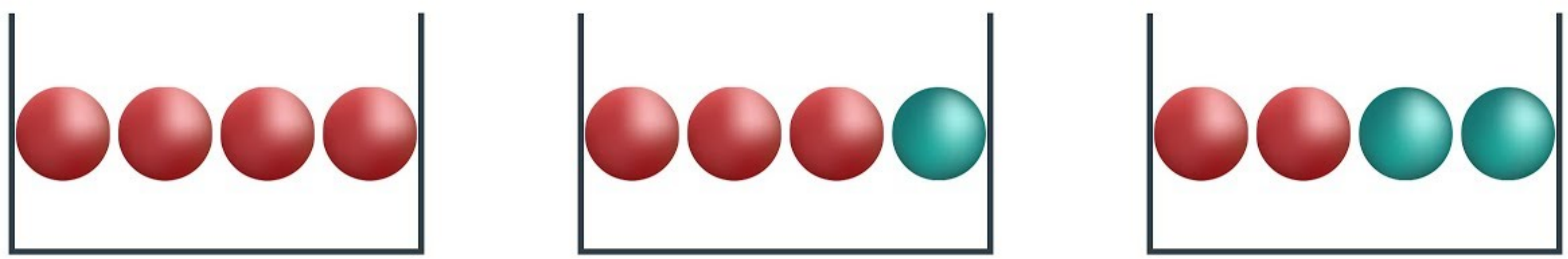
- Gini coefficient → how uneven income/wealth distribution across a population is
- In the context of decision trees
  - how uneven (or non-homogeneous) are the classes after a split

Gini(D) = 1 - (Pr( )² + Pr( )²)

Let's suppose we test Age, and the instances associated with Age=Young look like this

Will repay loan

Will not repay loan



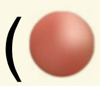
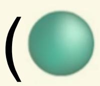
Even	**"Medium"**	**Uneven**
$1-(1^2+0^2) = 0$	$1-\left((3/4)^2+(1/4)^2\right) = 0.375$	$1-\left((2/4)^2+(2/4)^2\right) = 0.5$

Gini ⇒

more homogenous partition → ideal result of a split
(smaller value of the Gini coefficient)

Gini Criterion

- In the context of decision trees
  - how uneven (or non-homogeneous) are the classes after a split

Gini(D) = 1 - (Pr()^2 + Pr()^2)

- More generally, if there are m classes in a dataset D

$$\text{Gini}(D) \;=\; 1 - \left(\sum_{i=1}^{m} (p_i)^2 \right)$$

where p_i be the probability that the label/class i occurs in instances in a dataset D

Gini Criterion

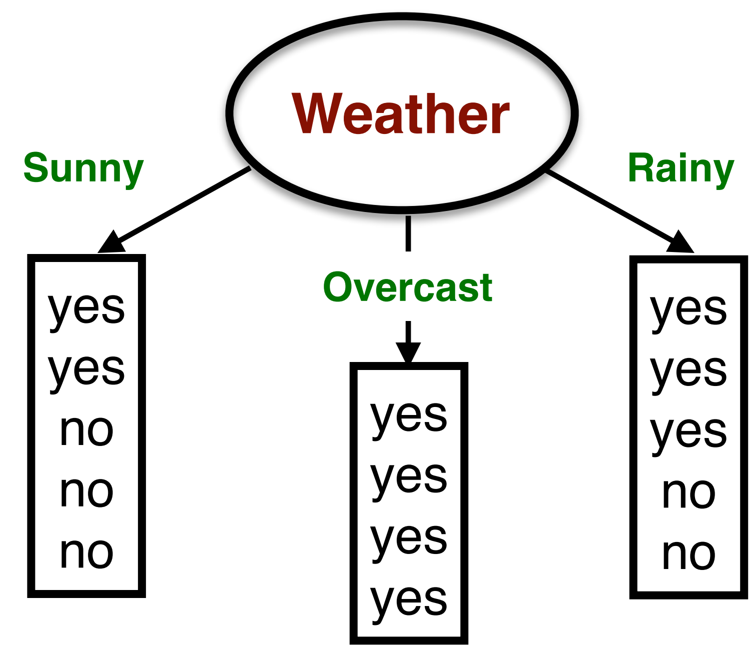
- Decision tree to predict whether a person will play tennis

Weather	Temperature	Humidity	Windy	PlayTennis
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

- Let's consider testing Weather

Original dataset: 9 instances "Yes"
5 instances "No"

yes yes yes yes yes yes yes yes yes
no no no no no



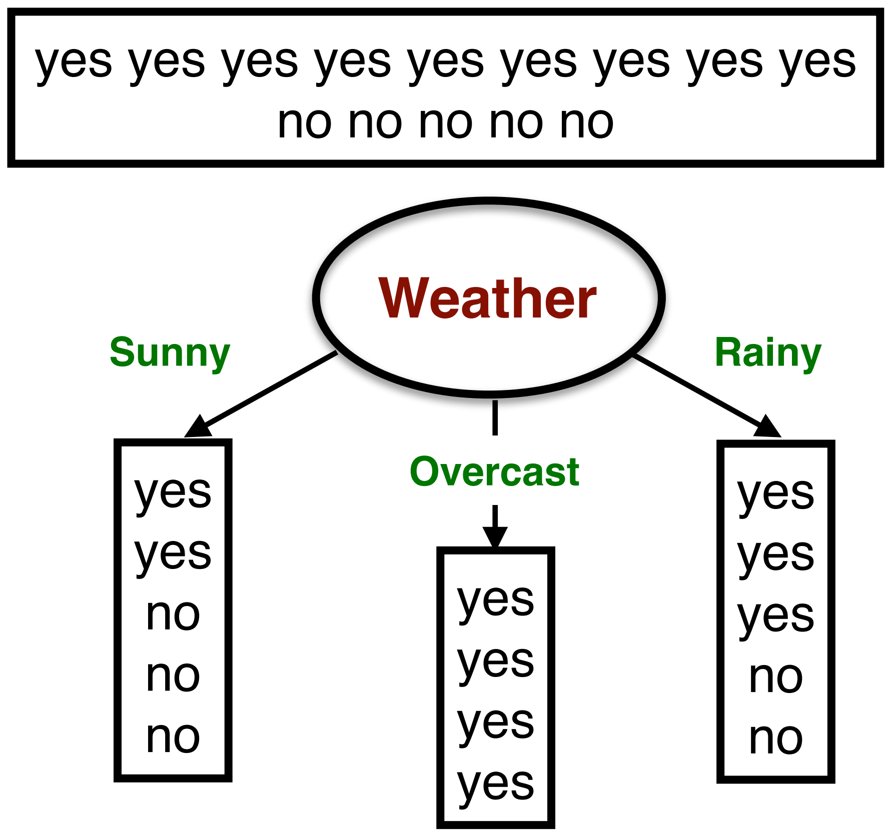
Gini Criterion

- Decision tree to predict whether a person will play tennis

- **Gini coefficient of the original dataset:**
  - **Gini**$(9/14, 5/14) = 1 - ((9/14)^2 + (5/14)^2)$
    $= \mathbf{0.459}$
- **Gini coeff. of partitions resulting from testing Weather:**
  - **Weather=Sunny**
    - $\mathbf{Gini}_{Sunny}(2/5, 3/5) = 1 - ((2/5)^2 + (3/5)^2)$
      $= 0.48$
  - **Weather=Overcast**
    - $\mathbf{Gini}_{Overcast}(4/4, 0/4) = 1 - ((4/4)^2 + (0/4)^2)$
      $= 0$
  - **Weather=Rainy**
    - $\mathbf{Gini}_{Rainy}(3/5, 2/5) = 1 - ((3/5)^2 + (2/5)^2)$
      $= 0.48$
- **Average Gini coefficient of the resulting partitions**
  - $(5/14)\times 0.48 + (4/14)\times 0 + (5/14)\times 0.48 = \mathbf{0.3428}$

- Let's consider testing Weather

Original dataset: 9 instances "Yes"
5 instances "No"



Gini Criterion

- Decision tree to predict whether a person will play tennis

- **Gini coefficient of the original dataset:**
  - **Gini**(9/14, 5/14) = 1 - ($(9/14)^2 + (5/14)^2$) = **0.459**
- **Gini coeff. of partitions resulting from testing Weather:**
  - **Weather=Sunny**
    - $\textbf{Gini}_{Sunny}(2/5, 3/5) = 1 - ((2/5)^2 + (3/5)^2) = 0.48$
  - **Weather=Overcast**
    - $\textbf{Gini}_{Overcast}(4/4, 0/4) = 1 - ((4/4)^2 + (0/4)^2) = 0$
  - **Weather=Rainy**
    - $\textbf{Gini}_{Rainy}(3/5, 2/5) = 1 - ((3/5)^2 + (2/5)^2) = 0.48$
- **Average Gini coefficient of the resulting partitions**
  - (5/14)x0.48 + (4/14)x0 + (5/14)x0.48 = **0.3428**

Testing the attribute **Weather**:
Gini(Weather) = **0.3428**

- Now proceed similarly as when selecting attributes via Information Gain…

- Compute Gini coefficient of each candidate attribute
- Split dataset using the attribute with the **lowest Gini coefficient**

Gini Criterion

Formally:

- Let p_i be the probability that the label i occurs in instances in a dataset D
- $\text{Gini}(D) = 1 - \left(\sum_{i=1}^{m} (p_i)^2\right)$ is the Gini coefficient of an arbitrary dataset D (m is the number of classes/labels)
- Assume that the attribute A can take up v values

  (that is, if we split D based on attribute A, we will end up with v partitions)
- Let $\text{Gini}_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \text{Gini}(D_j)$ be the Gini coefficient associated with splitting D based on A
- At each step, the algorithm splits the instances based on the attribute A with **lowest Gini coefficient**

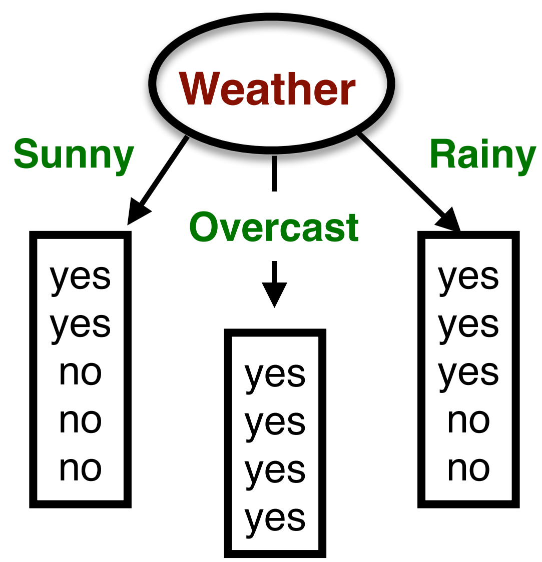
Criteria for Selecting an Attribute to Test

- Main criteria for selecting which attribute to test:
  - **Information Gain** - ID3 Algorithm (Quilan, 1987)
  - **Information Gain Ratio** - C4.5 Algorithm (Quilan, 1988)
  - **Gini Impurity** - CART Algorithm (Breiman, 1984)

- **Empirically:**
  - Information Gain Ratio is almost always better than Information Gain
    - in terms of predictive power and complexity of the resulting decision trees
  - However, in practice
    - which criterion will work best depends heavily on the application
    - should test them all and compare the resulting performances

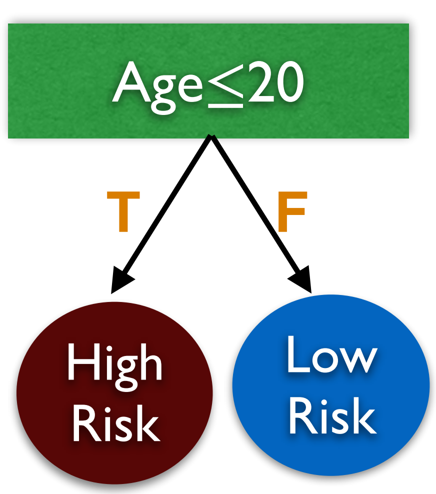
Dealing with Numerical Attributes

- So far we have studied how to select which **categorical attribute** to split



One branch per possible value of the attribute

- How do we decide a splitting point/value in case of **numerical attributes**?



Consider deciding how to split the attribute **Age**

Pick a threshold value, V
Generate two branches/disjoint partitions:

- one partition with instances s.t. **Age** $\leq V$
- one partition with instances s.t. **Age** $> V$

Dealing with Numerical Attributes

- How do we decide a splitting point/value in case of **numerical attributes**?

  - one partition with instances s.t. **Age** $\leq V$
  - one partition with instances s.t. **Age** $> V$

1) Sort the instances according to the value of the attribute

Name	***Age***	***Gender***	***TrafficTicket***	**Class: High-Risk Driver**
John	43	M	Yes	**High Risk**
Peter	18	M	No	**High Risk**
Anna	35	F	No	**Low Risk**
Paula	19	F	No	**High Risk**
Mark	90	M	Yes	**High Risk**
Marisa	21	F	Yes	**High Risk**
Bob	30	M	No	**Low Risk**

Dealing with Numerical Attributes

- How do we decide a splitting point/value in case of **numerical attributes**?

  - one partition with instances s.t. **Age** $\leq V$
  - one partition with instances s.t. **Age** $> V$

1) Sort the instances according to the value of the attribute

Name	***Age***	***Gender***	***TrafficTicket***	**Class: High-Risk Driver**
Peter	18	M	No	**High Risk**
Paula	19	F	No	**High Risk**
Marisa	21	F	Yes	**High Risk**
Bob	30	M	No	**Low Risk**
Anna	35	F	No	**Low Risk**
John	43	M	Yes	**High Risk**
Mark	90	M	Yes	**High Risk**

Dealing with Numerical Attributes

- How do we decide a splitting point/value in case of **numerical attributes**?

  - one partition with instances s.t. **Age** $\leq V$
  - one partition with instances s.t. **Age** $> V$

1) Sort the instances according to the value of the attribute

2) Evaluate splits done using as threshold the mean values between consecutive Ages

Name	*Age*	*Gender*	*TrafficTicket*	Class: High-Risk Driver
Peter	18	M	No	High Risk
Paula	19	F	No	High Risk
Marisa	21	F	Yes	High Risk
Bob	30	M	No	Low Risk
Anna	35	F	No	Low Risk
John	43	M	Yes	High Risk
Mark	90	M	Yes	High Risk

Age≤18.5

Dealing with Numerical Attributes

- How do we decide a splitting point/value in case of **numerical attributes**?

  - one partition with instances s.t. **Age** $\leq V$
  - one partition with instances s.t. **Age** $> V$

1) Sort the instances according to the value of the attribute

2) Evaluate splits done using as threshold the mean values between consecutive Ages

Name	*Age*	*Gender*	*TrafficTicket*	Class: High-Risk Driver
Peter	18	M	No	High Risk
Paula	19	F	No	High Risk
Marisa	21	F	Yes	High Risk
Bob	30	M	No	Low Risk
Anna	35	F	No	Low Risk
John	43	M	Yes	High Risk
Mark	90	M	Yes	High Risk

Age≤18.5

Age≤20

Dealing with Numerical Attributes

- How do we decide a splitting point/value in case of **numerical attributes**?

  - one partition with instances s.t. **Age** $\leq V$
  - one partition with instances s.t. **Age** $> V$

1) Sort the instances according to the value of the attribute

2) Evaluate splits done using as threshold the mean values between consecutive Ages

Name	*Age*	*Gender*	*TrafficTicket*	Class: High-Risk Driver
Peter	18	M	No	High Risk
Paula	19	F	No	High Risk
Marisa	21	F	Yes	High Risk
Bob	30	M	No	Low Risk
Anna	35	F	No	Low Risk
John	43	M	Yes	High Risk
Mark	90	M	Yes	High Risk

Age≤18.5

Age≤20

Age≤25.5

Dealing with Numerical Attributes

- How do we decide a splitting point/value in case of **numerical attributes**?

  - one partition with instances s.t. **Age** $\leq V$
  - one partition with instances s.t. **Age** $> V$

1) Sort the instances according to the value of the attribute

2) Evaluate splits done using as threshold the mean values between consecutive Ages

Name	***Age***	***Gender***	***TrafficTicket***	**Class: High-Risk Driver**
Peter	18	M	No	High Risk
Paula	19	F	No	High Risk
Marisa	21	F	Yes	High Risk
Bob	30	M	No	Low Risk
Anna	35	F	No	Low Risk
John	43	M	Yes	High Risk
Mark	90	M	Yes	High Risk

$Age \leq 18.5$

$Age \leq 20$

$Age \leq 25.5$

$Age \leq 32.5$

Dealing with Numerical Attributes

- How do we decide a splitting point/value in case of **numerical attributes**?

  - one partition with instances s.t. **Age** $\leq V$
  - one partition with instances s.t. **Age** $> V$

1) Sort the instances according to the value of the attribute

2) Evaluate splits done using as threshold the mean values between consecutive Ages

Name	*Age*	*Gender*	*TrafficTicket*	Class: High-Risk Driver
Peter	18	M	No	High Risk
Paula	19	F	No	High Risk
Marisa	21	F	Yes	High Risk
Bob	30	M	No	Low Risk
Anna	35	F	No	Low Risk
John	43	M	Yes	High Risk
Mark	90	M	Yes	High Risk

- Age≤18.5
- Age≤20
- Age≤25.5
- Age≤32.5
- Age≤39

Dealing with Numerical Attributes

- How do we decide a splitting point/value in case of **numerical attributes**?

  - one partition with instances s.t. **Age** $\leq V$
  - one partition with instances s.t. **Age** $> V$

1) Sort the instances according to the value of the attribute

2) Evaluate splits done using as threshold the mean values between consecutive Ages

Name	*Age*	*Gender*	*TrafficTicket*	Class: High-Risk Driver
Peter	18	M	No	High Risk
Paula	19	F	No	High Risk
Marisa	21	F	Yes	High Risk
Bob	30	M	No	Low Risk
Anna	35	F	No	Low Risk
John	43	M	Yes	High Risk
Mark	90	M	Yes	High Risk

Age≤18.5

Age≤20

Age≤25.5

Age≤32.5

Age≤39

Age≤66.5

Dealing with Numerical Attributes

- How do we decide a splitting point/value in case of **numerical attributes**?

  - one partition with instances s.t. **Age** $\leq V$
  - one partition with instances s.t. **Age** $> V$

1) Sort the instances according to the value of the attribute

2) Evaluate splits done using as threshold the mean values between consecutive Ages

Age≤18.5	Age≤20	Age≤25.5	Age≤32.5	Age≤39	Age≤66.5

3) Pick the split threshold that maximizes the criterion of interest ***(Info. Gain, Gini, etc.)***

- It has been shown that, for most commonly-used splitting criteria
  - testing only thresholds that correspond to such mean values is sufficient

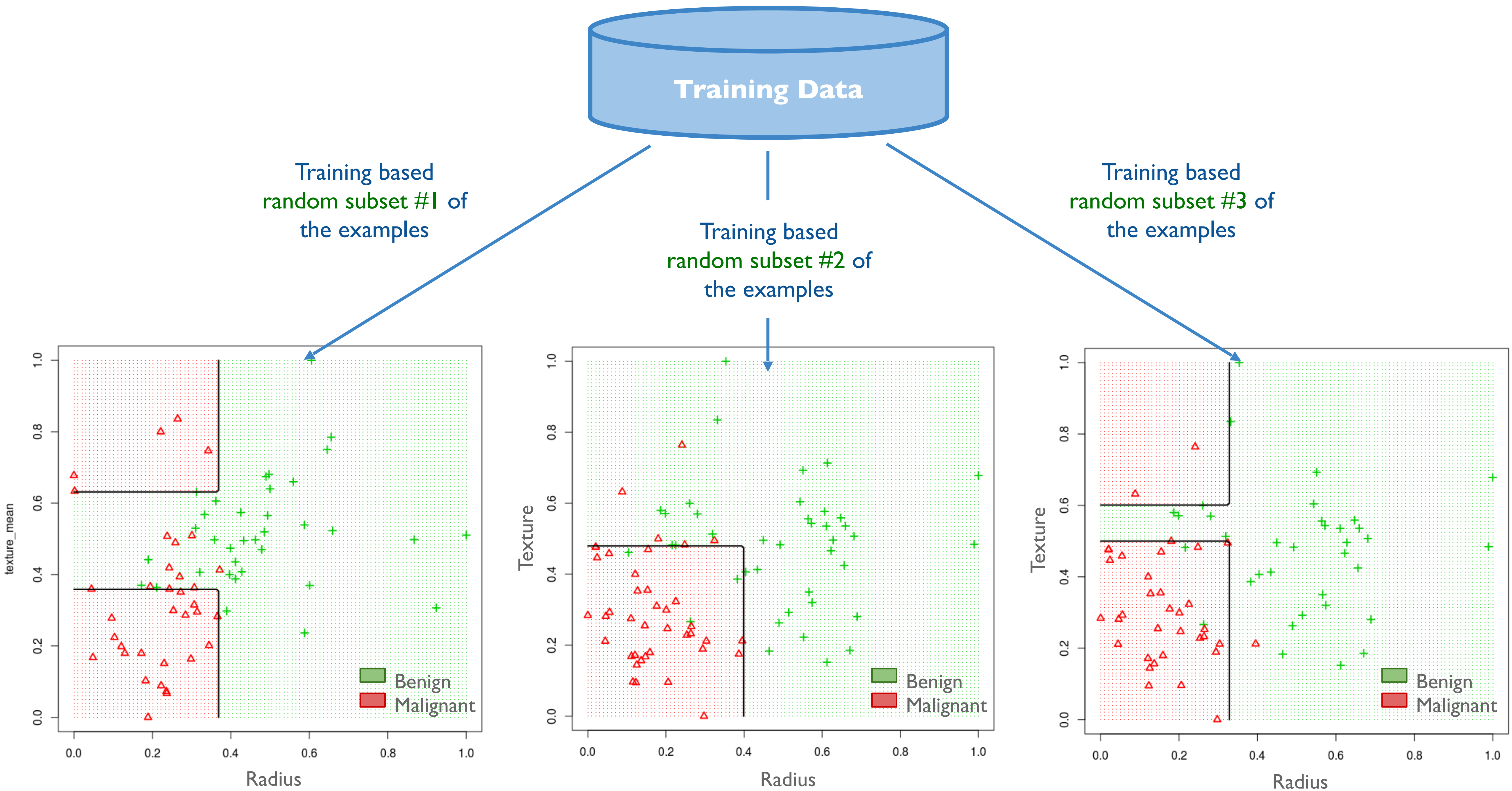
Decision Trees: Pros and Cons

- **Pros:**
  - Simple for humans to understand and interpret
  - Handles both numerical and categorical attributes
  - Requires little data preparation *(e.g., no need to normalize attributes)*
  - Performs well with large datasets
  - “Automatically" ignores irrelevant attributes not useful to predict the class/label

- **Cons:**
  - Non-robust: small variations in the dataset can generate completely different trees

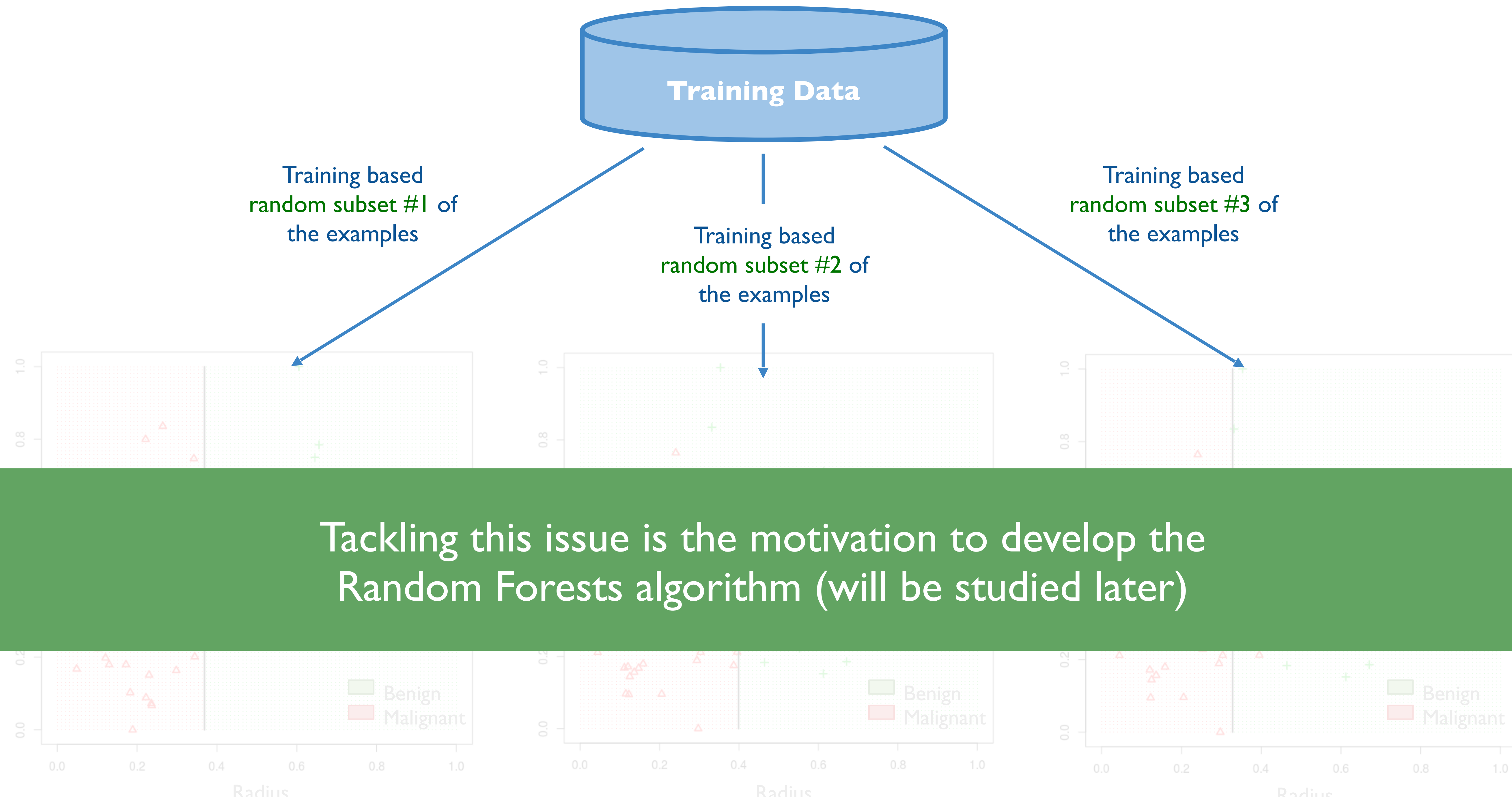
Decision Trees Can be Non-Robust

Decision boundary can be **heavily influenced** by **small changes** to the training data



Decision Trees Can be Non-Robust

Decision boundary can be **heavily influenced** by **small changes** to the training data



Tackling this issue is the motivation to develop the Random Forests algorithm (will be studied later)

Decision Trees: Pros and Cons

- **Pros:**
  - Simple for humans to understand and interpret
  - Handles both numerical and categorical attributes
  - Requires little data preparation *(e.g., no need to normalize attributes)*
  - Performs well with large datasets
  - “Automatically" ignores irrelevant attributes not useful to predict the class/label

- **Cons:**
  - Non-robust: small variations in the dataset can generate completely different trees
  - Often generate overly-complicated trees that overfit to training data
    - i.e., that do not generalize well (make correct predictions) to new instances
  - Although it is possible to deal with numerical attributes, it is time-consuming
    - estimates suggest that processing them takes ~70% of execution time (Catlett, 1991)

Machine Learning
CMPSCI 589

Bruno C. da Silva
bsilva@cs.umass.edu

Decision Trees (2/2)

UMassAmherst
College of Information & Computer Sciences