Machine Learning
CMPSCI 589

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

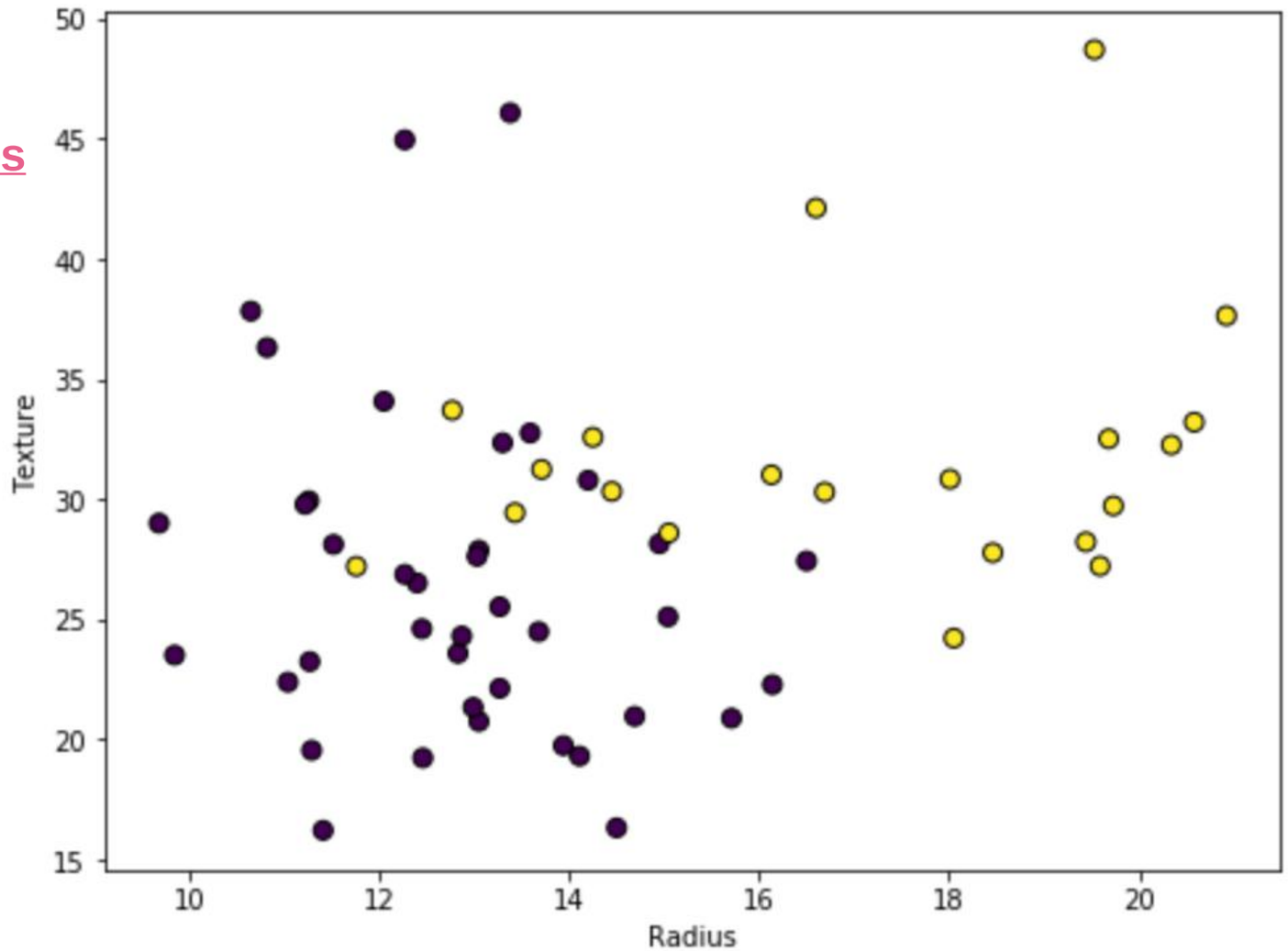
Decision Trees (1/2)

UMassAmherst
College of Information & Computer Sciences

Review: The *k*-nearest neighbors (*k*-NN) algorithm

- Instances belonging to **a same class** are likely to be **near** in the space of input attributes

- Instances belonging to **different classes** are likely to be **far away** in the space of input attributes



- To classify a **new instance** **(with unknown class/label)** its ***k* nearest instances vote**

  - Based, e.g., on Euclidean distance

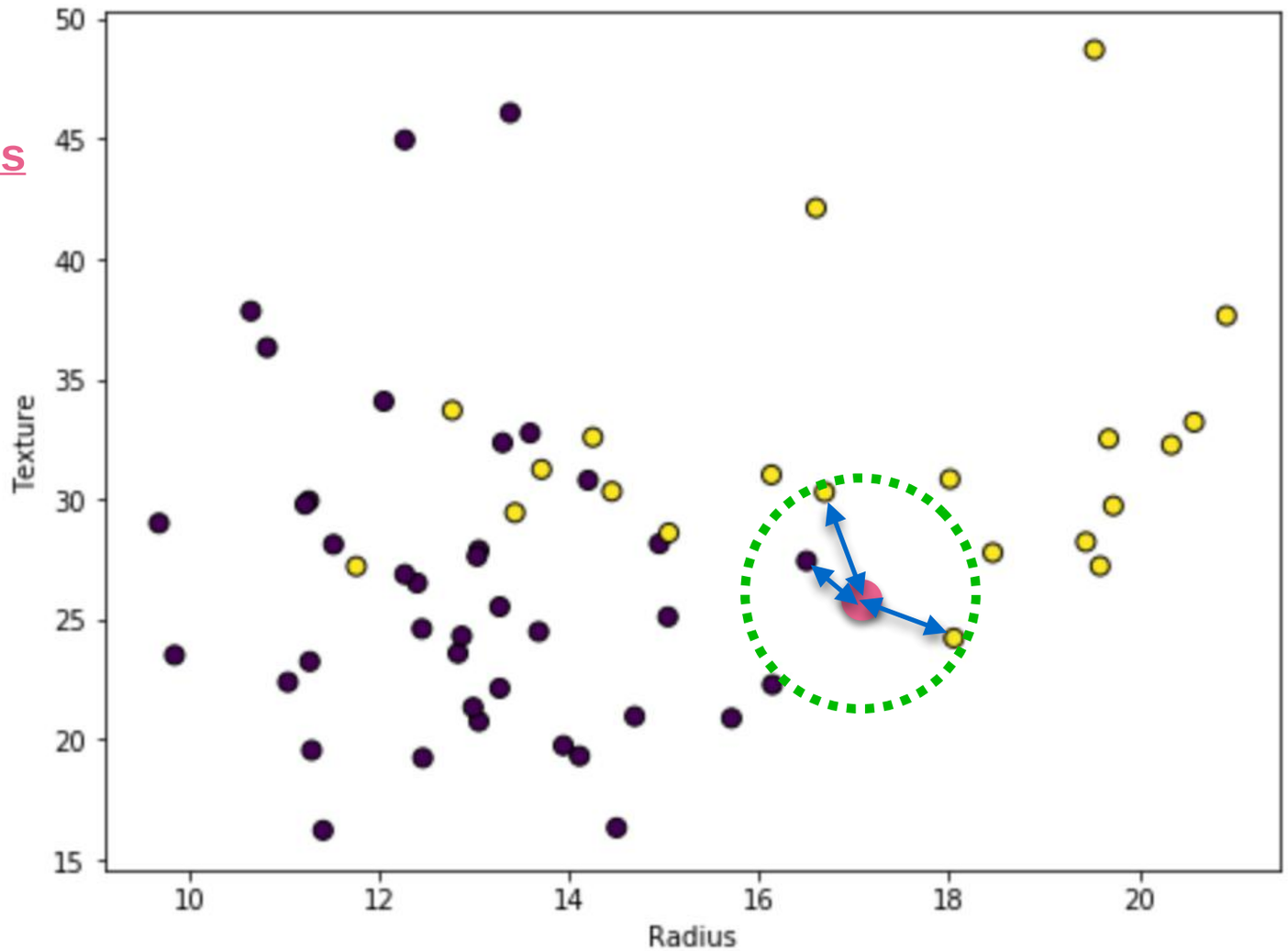
$$d(x_i, x_j) = \sqrt{\sum_{d=1}^{D} (x_i^d - x_j^d)^2}$$

Review: The *k*-nearest neighbors (*k*-NN) algorithm

- Instances belonging to **a same class** are likely to be **near** in the space of input attributes

- Instances belonging to **different classes** are likely to be **far away** in the space of input attributes

- To classify a **new instance** **(with unknown class/label)** its ***k* nearest instances vote**



Assume k=3
What is the predicted class?

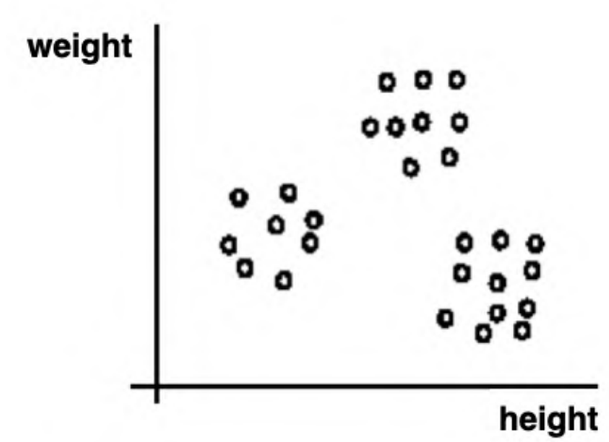
Feature Normalization

- When computing distances
  - it is sometimes necessary to pre-process training data to “adjust" the magnitude of attributes
  - so that all attributes have an equal contribution when computing distances

---

- Consider the following instances, representing the height and weight of a person
  - $x_i = [1.3\text{m},\ 90\text{kg}]$
  - $x_j = [2.1\text{m},\ 110\text{kg}]$

$$d(x_i,\ x_j) = = \sqrt{(1.3 - 2.1)^2 + (90 - 110)^2} = 20$$



- The magnitude of this distance value is mostly due to the weight difference
  - That is, differences in height would have almost no effect on the distance
  - But when “comparing" two people, what matters most?
    - Weight or height? → Both should contribute equally
- **It is necessary to normalize attributes so that all are in the same range of values!**

Feature Normalization

	Attribute 1 Area	Attribute 2 #bedrooms	Attribute 3 #floors
House_1	2100	5	1
House_2	1410	3	2
House_3	1534	3	2
House_4	850	2	1

Intuition

- area attribute is in the range of 0-2100
- #bedroom attribute is in the range of 1-5

area has higher-magnitude values than #bedrooms

differences in area affect distances much more than differences in #bedrooms

Feature normalization

- transform all attributes/features to a same range
- typically [0, 1] or [-1, +1]

$$\max_i := \max(x_i^1, \ldots, x_i^n)$$

$$\min_i := \min(x_i^1, \ldots, x_i^n)$$

$$x_i^d := \frac{x_i^d - \min_d}{\max_d - \min_d}$$

Normalizing to [0, 1]

Feature Normalization

	Attribute 1 Area	Attribute 2 #bedrooms	Attribute 3 #floors
House_1	2100	5	1
House_2	**1410**	3	2
House_3	1534	3	2
House_4	850	2	1

Intuition

- area attribute is in the range of 0-2100
- #bedroom attribute is in the range of 1-5

area has higher-magnitude values than #bedrooms

differences in area affect distances much more than differences in #bedrooms

Feature normalization

- transform all attributes/features to a same range
- typically [0, 1] or [-1, +1]

$$\max_1 = 2100$$

$$\min_1 = 850$$

$$x_2^1 = \frac{1410 - 850}{2100 - 850} = 0.448$$

$$\max_i := \max(x_i^1, \ldots, x_i^n)$$

$$\min_i := \min(x_i^1, \ldots, x_i^n)$$

$$x_i^d := \frac{x_i^d - \min_d}{\max_d - \min_d}$$

→ Normalizing to [0, 1]

Feature Normalization

	Attribute 1 Area	Attribute 2 #bedrooms	Attribute 3 #floors
House_1	2100	5	1
House_2	**0.448**	3	2
House_3	1534	3	2
House_4	850	2	1

Intuition

- area attribute is in the range of 0-2100
- #bedroom attribute is in the range of 1-5

area has higher-magnitude values than #bedrooms

differences in area affect distances much more than differences in #bedrooms

Feature normalization

- transform all attributes/features to a same range
- typically [0, 1] or [-1, +1]

$$\max_1 = 2100$$

$$\min_1 = 850$$

$$x_2^1 = \frac{1410 - 850}{2100 - 850} = 0.448$$

$$\max_i := \max(x_i^1, \ldots, x_i^n)$$

$$\min_i := \min(x_i^1, \ldots, x_i^n)$$

$$x_i^d := \frac{x_i^d - \min_d}{\max_d - \min_d}$$

→ Normalizing to [0, 1]

Feature Normalization

	Attribute 1 Area	Attribute 2 #bedrooms	Attribute 3 #floors
House_1	2100	5	1
House_2	1410	3	2
House_3	1534	3	2
House_4	850	2	1

Intuition

- area attribute is in the range of 0-2100
- #bedroom attribute is in the range of 1-5

area has higher-magnitude values than #bedrooms

differences in area affect distances much more than differences in #bedrooms

Feature normalization

- transform all attributes/features to a same range
- typically [0, 1] or [-1, +1]

$$\max_i := \max(x_i^1, \ldots, x_i^n)$$

$$\min_i := \min(x_i^1, \ldots, x_i^n)$$

$$x_i^d := 2\left(\frac{x_i^d - \min_d}{\max_d - \min_d}\right) - 1$$

Normalizing to [-1, 1]

Feature Normalization

	Attribute 1 Area	Attribute 2 #bedrooms	Attribute 3 #floors
House_1	2100	5	1
House_2	**1410**	3	2
House_3	1534	3	2
House_4	850	2	1

Intuition

- area attribute is in the range of 0-2100
- #bedroom attribute is in the range of 1-5

area has higher-magnitude values than #bedrooms

differences in area affect distances much more than differences in #bedrooms

Feature normalization

- transform all attributes/features to a same range
- typically [0, 1] or [-1, +1]

$$\max_1 = 2100$$

$$\min_1 = 850$$

$$x_2^1 = 2\left(\frac{1410 - 850}{2100 - 850}\right) - 1 = -0.104$$

$$\max_i := \max(x_i^1, \ldots, x_i^n)$$

$$\min_i := \min(x_i^1, \ldots, x_i^n)$$

$$x_i^d := 2\left(\frac{x_i^d - \min_d}{\max_d - \min_d}\right) - 1$$

→ Normalizing to [-1, 1]

Feature Normalization

	Attribute 1 Area	Attribute 2 #bedrooms	Attribute 3 #floors
House_1	2100	5	1
House_2	**-0.104**	3	2
House_3	1534	3	2
House_4	850	2	1

Intuition

- area attribute is in the range of 0-2100
- #bedroom attribute is in the range of 1-5

area has higher-magnitude values than #bedrooms

differences in area affect distances much more than differences in #bedrooms

Feature normalization

- transform all attributes/features to a same range
- typically [0, 1] or [-1, +1]

$$\max_1 = 2100$$

$$\min_1 = 850$$

$$x_2^1 = 2\left(\frac{1410 - 850}{2100 - 850}\right) - 1 = -0.104$$

$$\max_i := \max(x_i^1, \ldots, x_i^n)$$

$$\min_i := \min(x_i^1, \ldots, x_i^n)$$

$$x_i^d := 2\left(\frac{x_i^d - \min_d}{\max_d - \min_d}\right) - 1$$

→ Normalizing to [-1, 1]

The *k*-nearest neighbors (*k*-NN) algorithm

- ***k*-NN sometimes works really well as a classifier**
  - but it does not lead to any insights about the data
  - e.g., which attributes are relevant to determine if a person is going to repay a loan?

Decision Trees

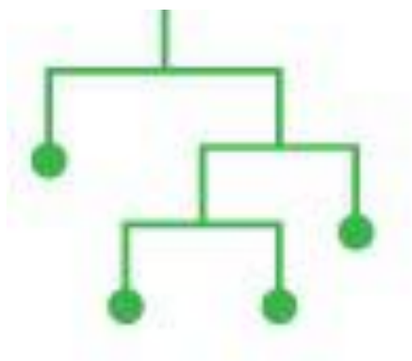
- ***k*-NN sometimes works really well as a classifier**
  - but it does not lead to any insights about the data
  - e.g., which attributes are relevant to determine if a person is going to repay a loan?

- **Decision Trees**
  - extract a set of classification rules to classify a given instance
    - like IF-ELSE statements, testing different attributes
  - ML model that is highly interpretable by humans

Decision Trees

	#friends	funny score	Likes
Post1	40	8	28
Post2	36	10	28
Post3	20	6	16
Post4	56	4	31
Post5	58	0	29
Post6	46	10	33

Decision Trees



```
if #friends >= 50
    if funny_score > 3
        Likes >= 30
    else
        Likes < 30
else
    if funny_score > 9
        Likes >= 30
    else
        Likes < 30
```

Learning to Play the 20 Questions Game

http://en.akinator.com/

Question N°1

Is your character a female?

Yes

No

Don't know

Probably

Probably not

Correct

Question N°2

Is your character real?

Yes

No

Don't know

Probably

Probably not

Correct

Question N°3

Is your character a famous youtuber?

Yes

No

Don't know

Probably

Probably not

Correct

Question N°4

Is your character older than 35 years old?

Yes

No

Don't know

Probably

Probably not

Correct

Question N°5

Is your character American ?

- Yes
- No
- Don't know
- Probably
- Probably not

Correct

Question N°6

Is your character an actor?

Yes

No

Don't know

Probably

Probably not

Correct

Question N° 7

Is your character the president or was he?

Yes

No

Don't know

Probably

Probably not

Correct

Question N° 8

Is your character black?

Yes

No

Don't know

Probably

Probably not

Decision Trees

- Which attributes of a person to test first, to guess as fast as possible?
  - Is the person a man or a woman?
  - Is the person older than 5 years old?

Information Gain

- G("Gender") = 0.9
- G("Lives_in_USA") = 0.73
- G("Is_Politician") = 0.36
- …

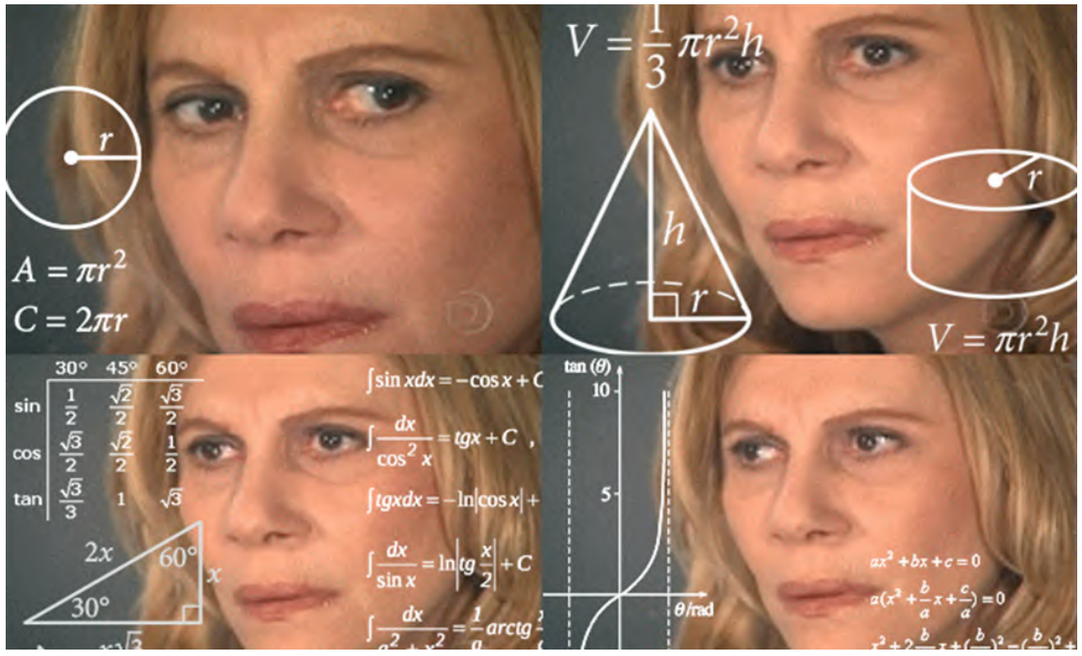
Entropy of a set (or dataset)

Decision Trees

- Which attributes of a person to test first, to guess as fast as possible?
  - Is the person a man or a woman?
  - Is the person older than 5 years old?

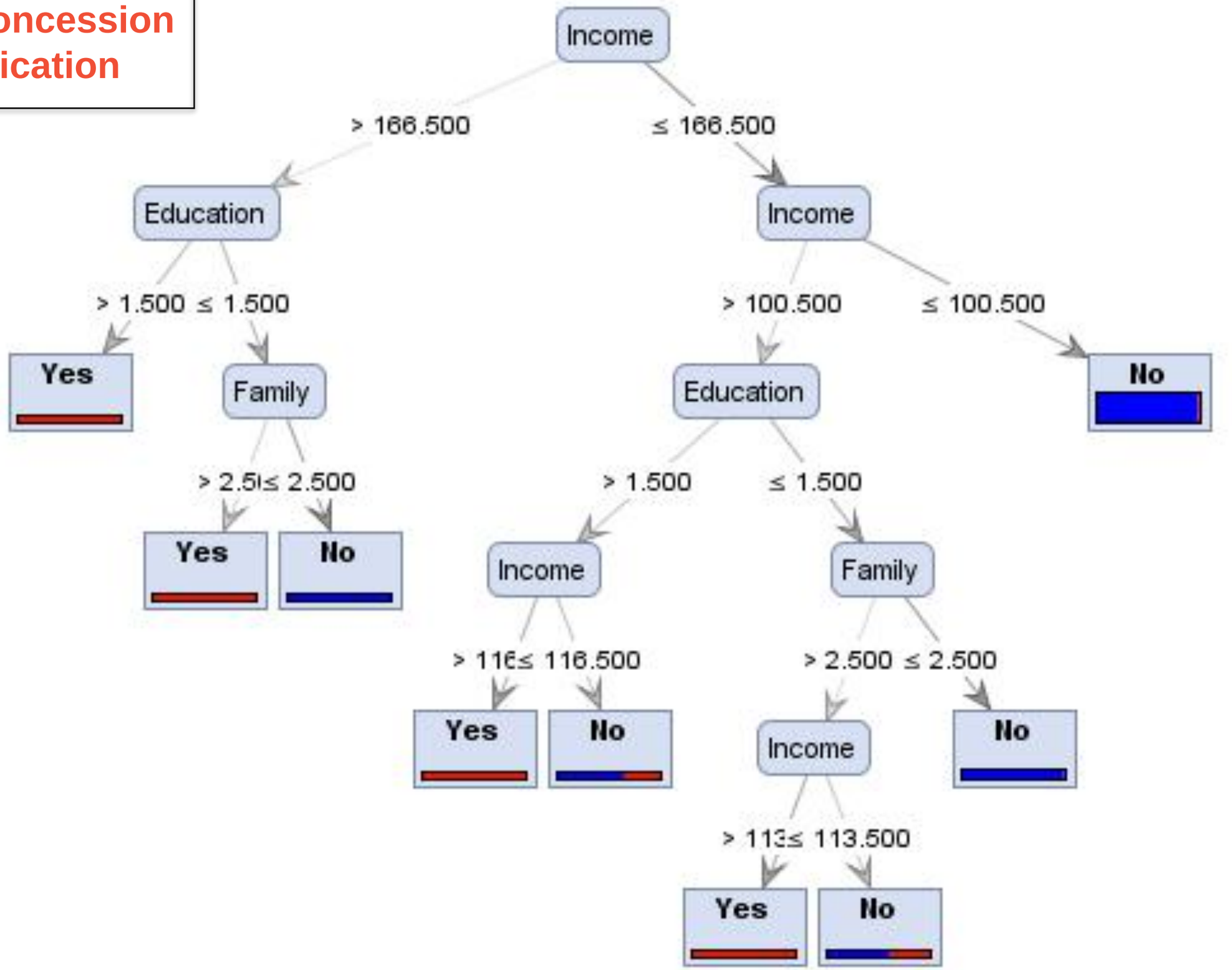
Information Gain

- G(“Gender") = 0.9
- G(“Lives_in_USA”) = 0.73
- G(“Is_Politician”) = 0.36
- …



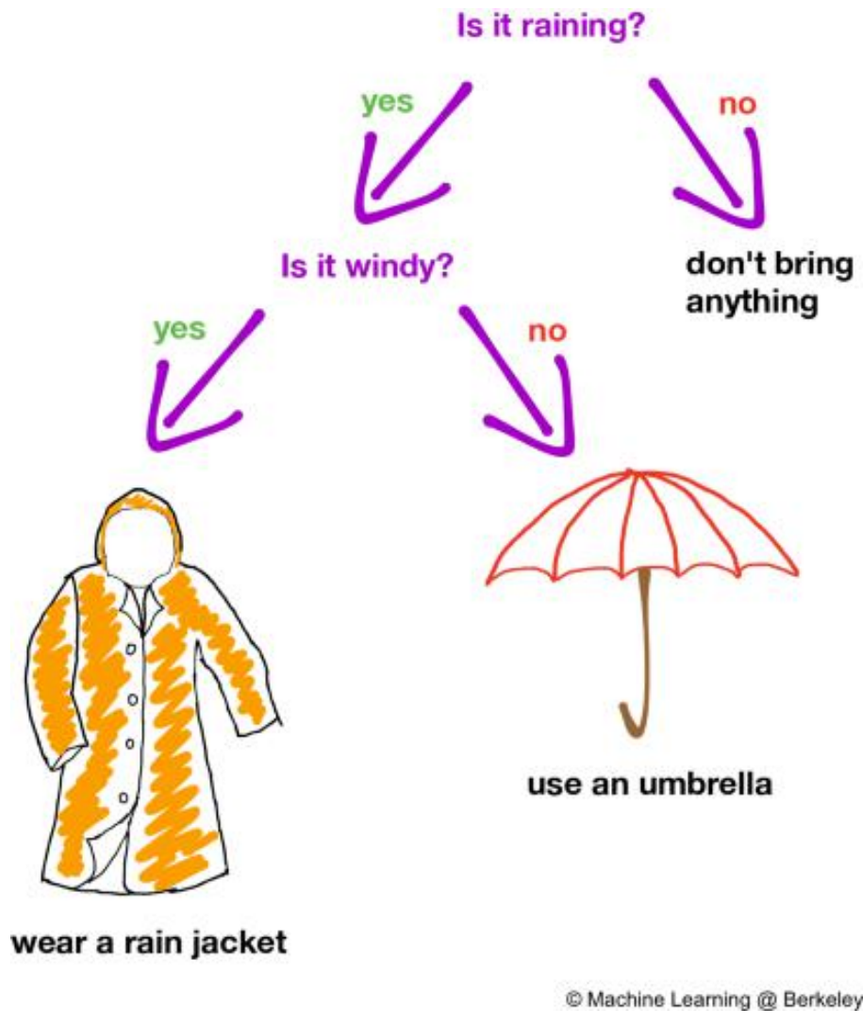
Decision Trees

Loan Concession Application



Decision Trees

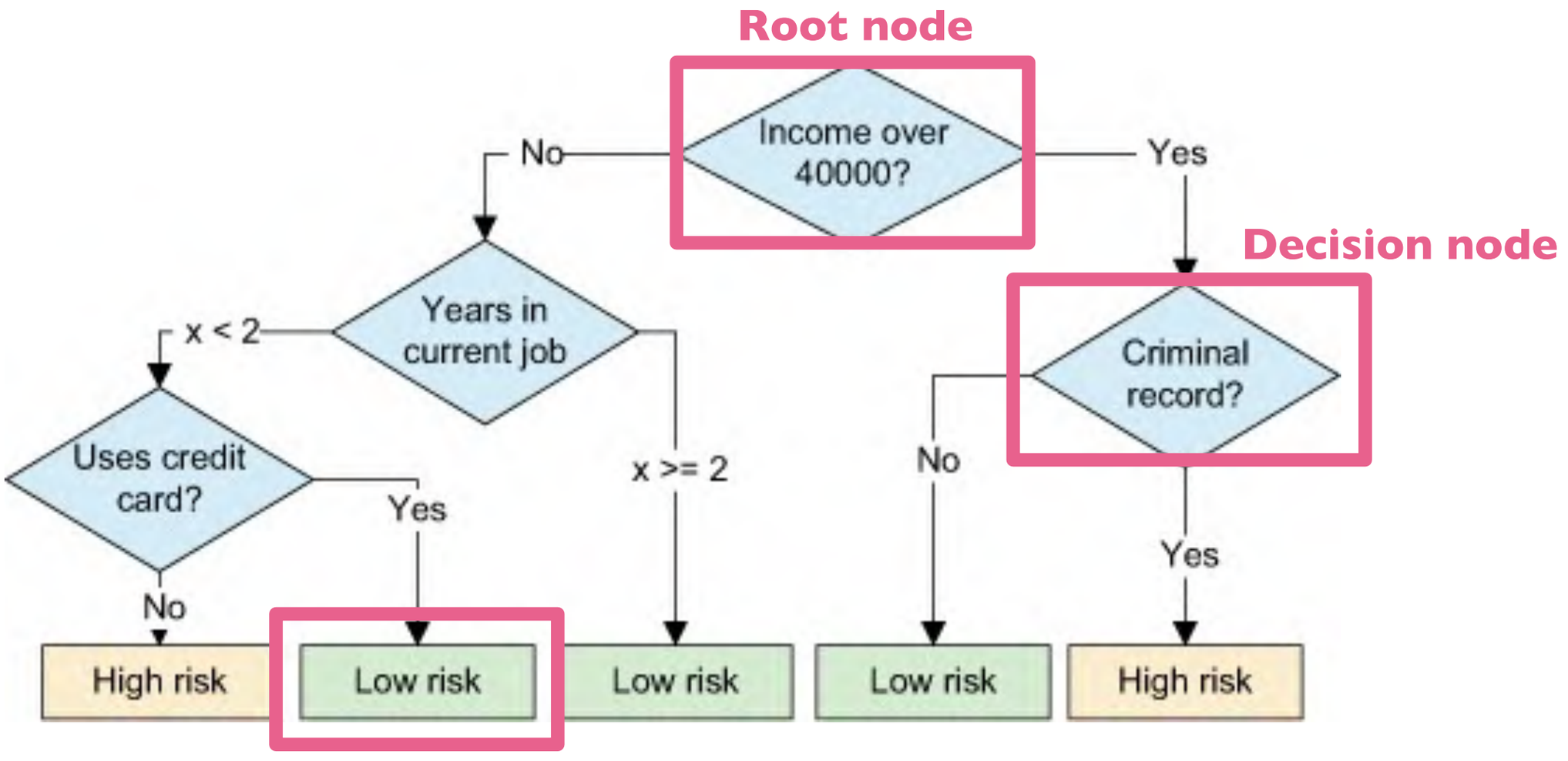
- **Decision Tree**
  - Simple ML model/algorithm
  - Widely used in practice
  - Easy to *interpret*



- Performs a series of tests on attributes of an instance
  - eventually leading to decision/prediction about the class of that instance

Decision Trees

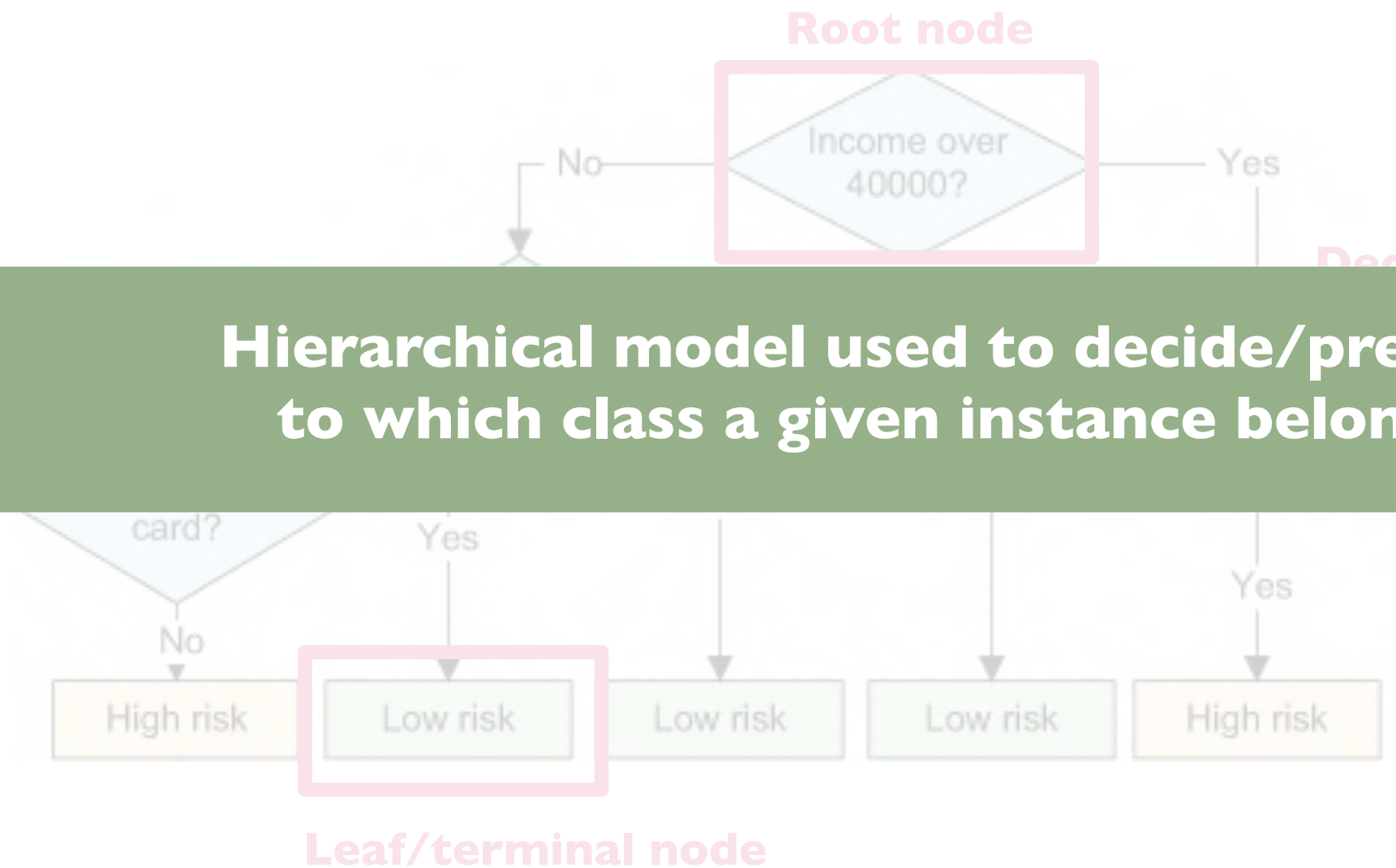
A decision tree for testing loan suitability



"Real-World Functional Programming with Examples in F# and C#". (Skeet and Petricek, 2010)

Decision Trees

A decision tree for testing loan suitability



Hierarchical model used to decide/predict to which class a given instance belongs

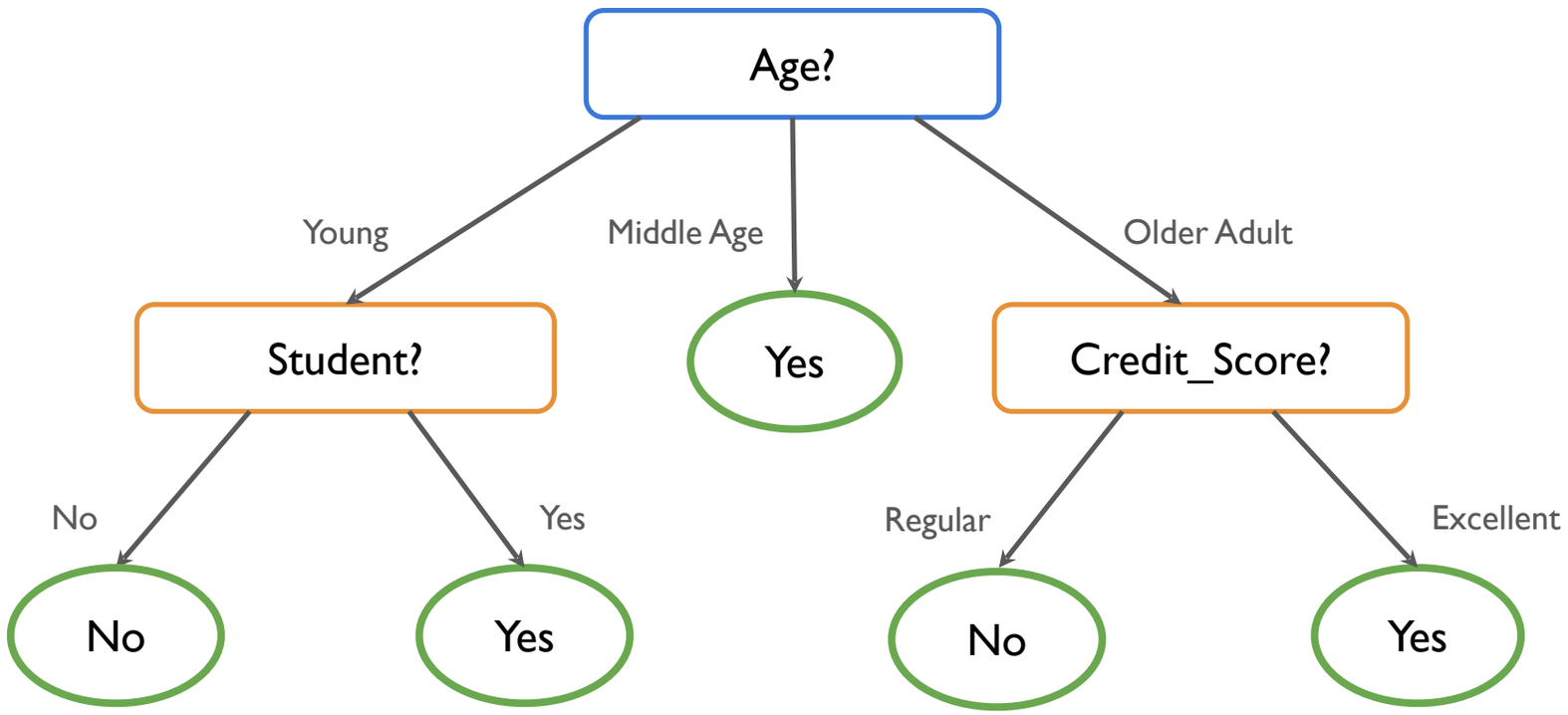
"Real-World Functional Programming with Examples in F# and C#". (Skeet and Petricek, 2010)

Decision Trees

How likely it is that a given client will buy a computer?

Each instance, x_i, is described by the following attributes

$x_i = [\text{Student}, \text{Age}, \text{Credit\_Score}]$, where $y_i \in \{\text{Yes}, \text{No}\}$



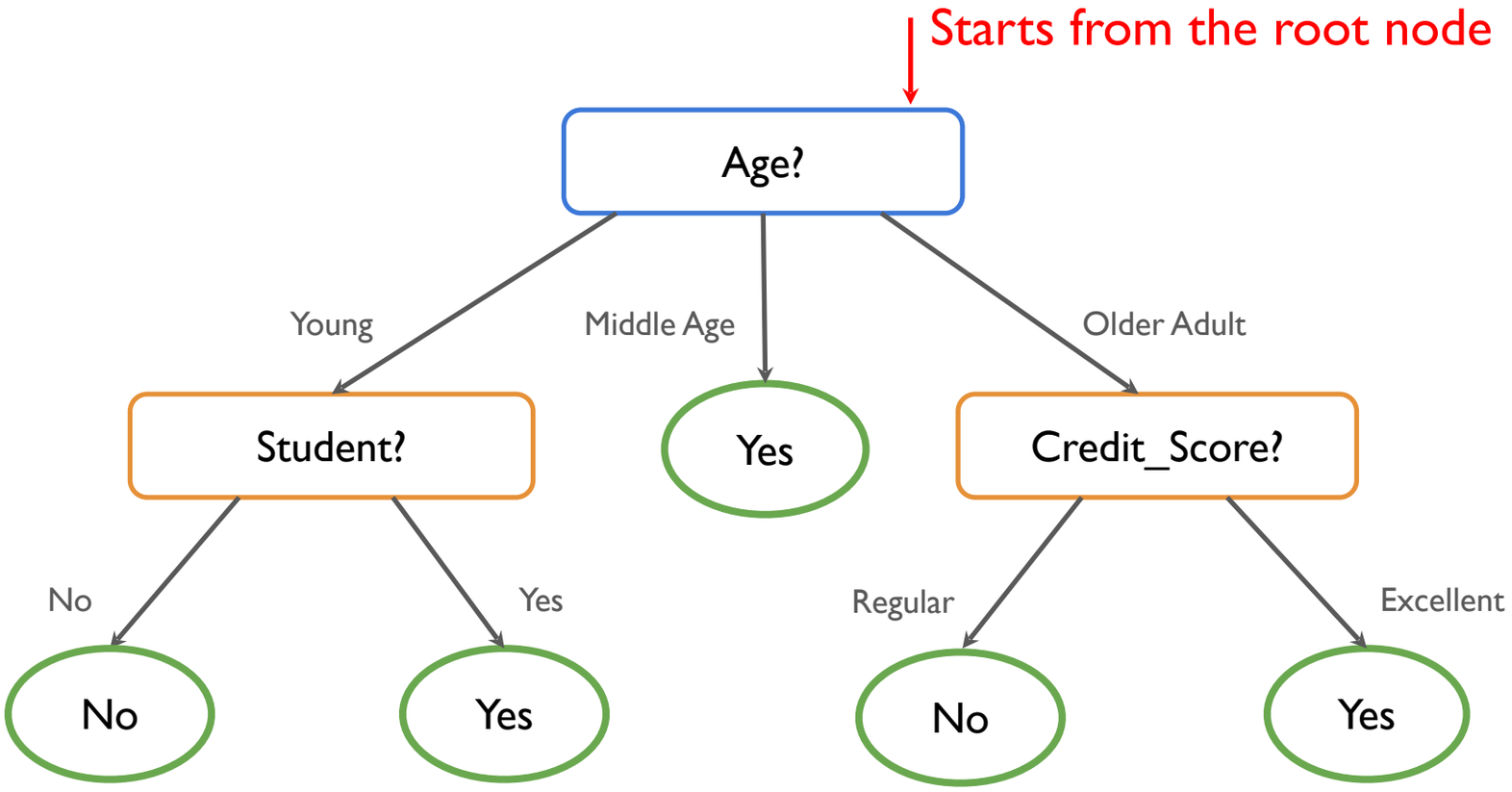
- Each non-leaf node tests the value of a given attribute
- Each branch corresponds to one possible value of that attribute
- Each leaf corresponds to predicting one particular class
- The path from the root node to a leaf defines a **classification rule**

Decision Trees

How likely it is that a given client will buy a computer?

Instance to be classified:

Student	Age	Credit_Score	Will_Buy_Computer
Yes	Young	Regular	??

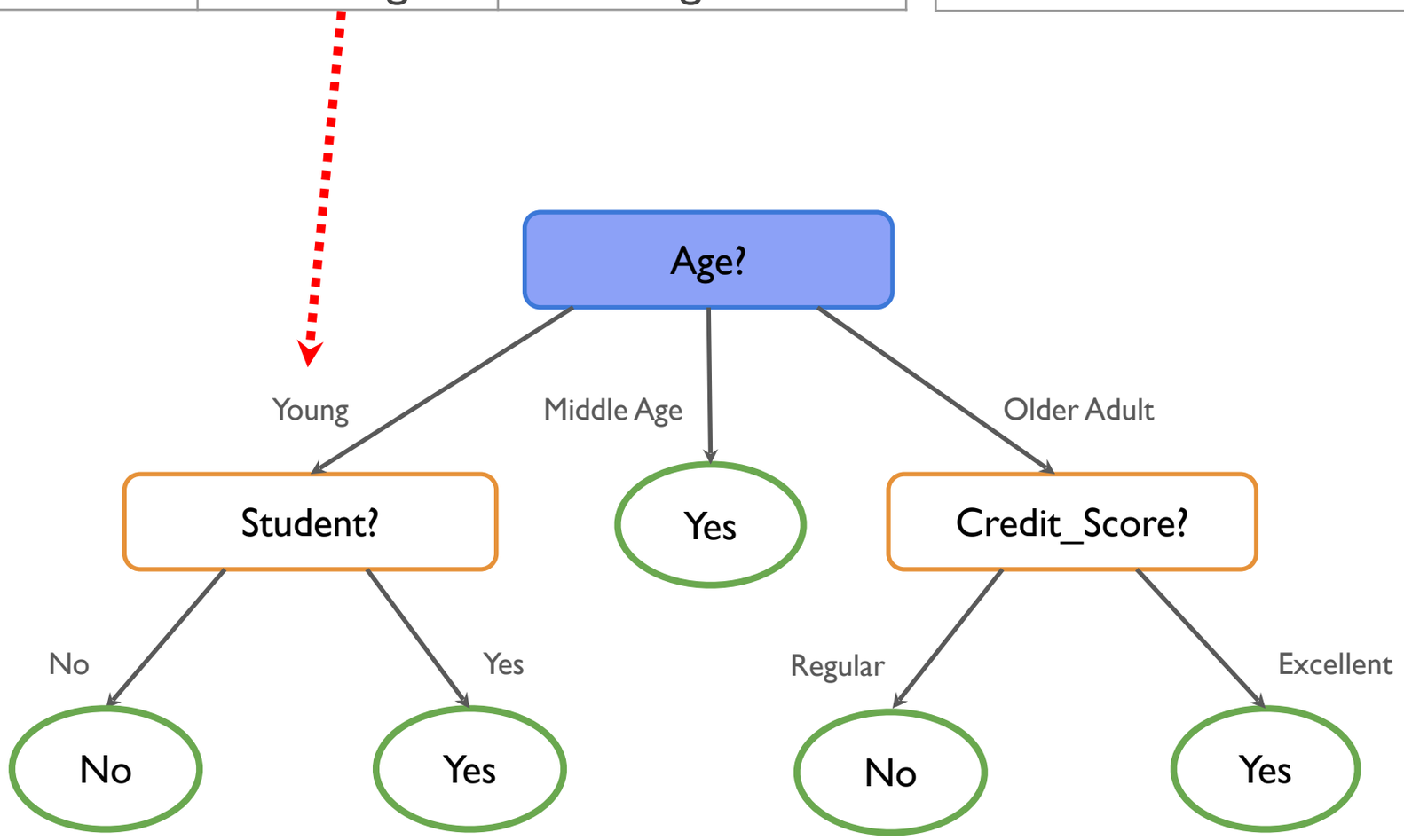


Decision Trees

How likely it is that a given client will buy a computer?

Instance to be classified:

Student	Age	Credit_Score	Will_Buy_Computer
Yes	Young	Regular	??

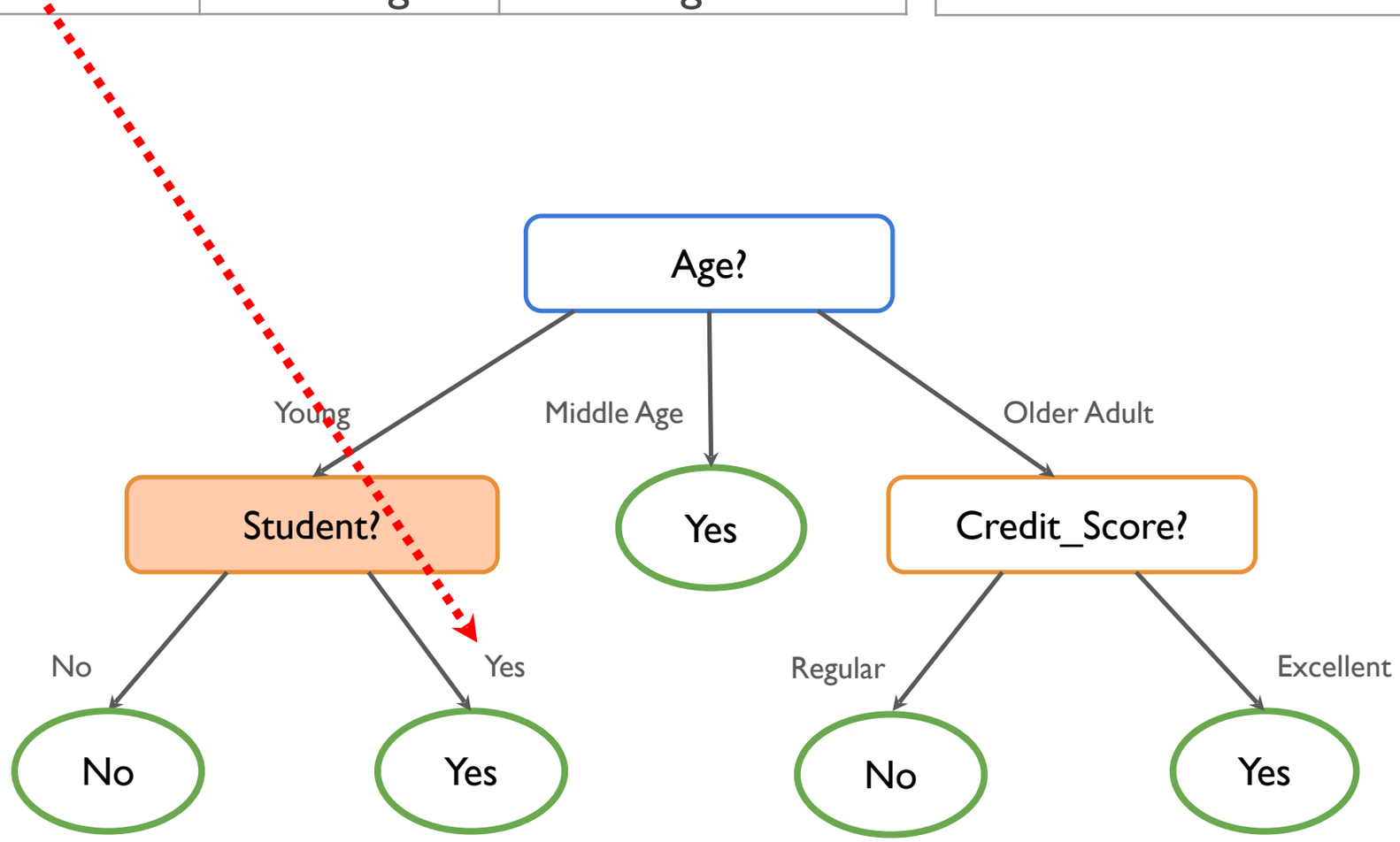


Decision Trees

How likely it is that a given client will buy a computer?

Instance to be classified:

Student	Age	Credit_Score	Will_Buy_Computer
Yes	Young	Regular	??



Decision Trees

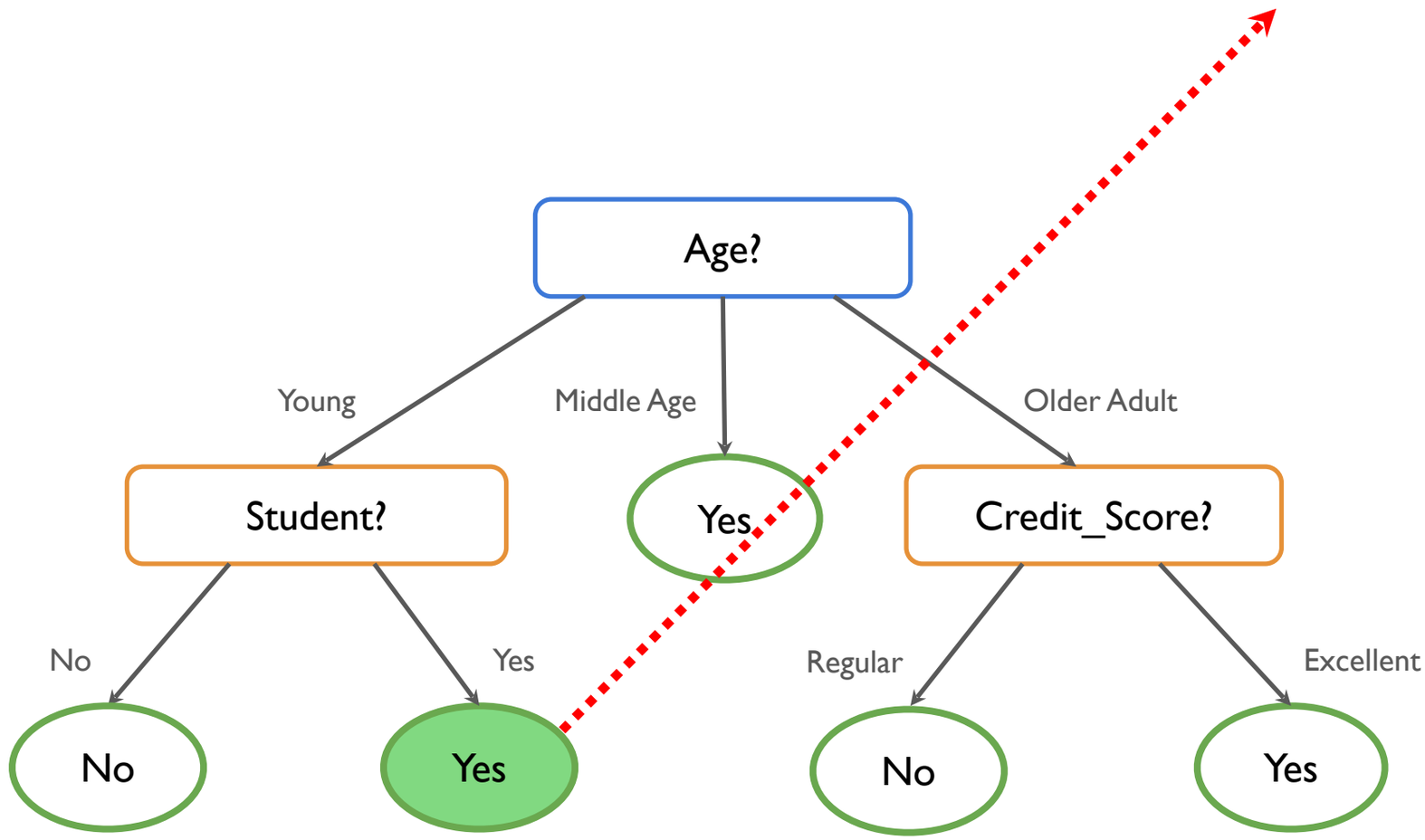
How likely it is that a given client will buy a computer?

Instance to be classified:

Student	Age	Credit_Score
Yes	Young	Regular

Predicted Class

Will_Buy_Computer
Yes

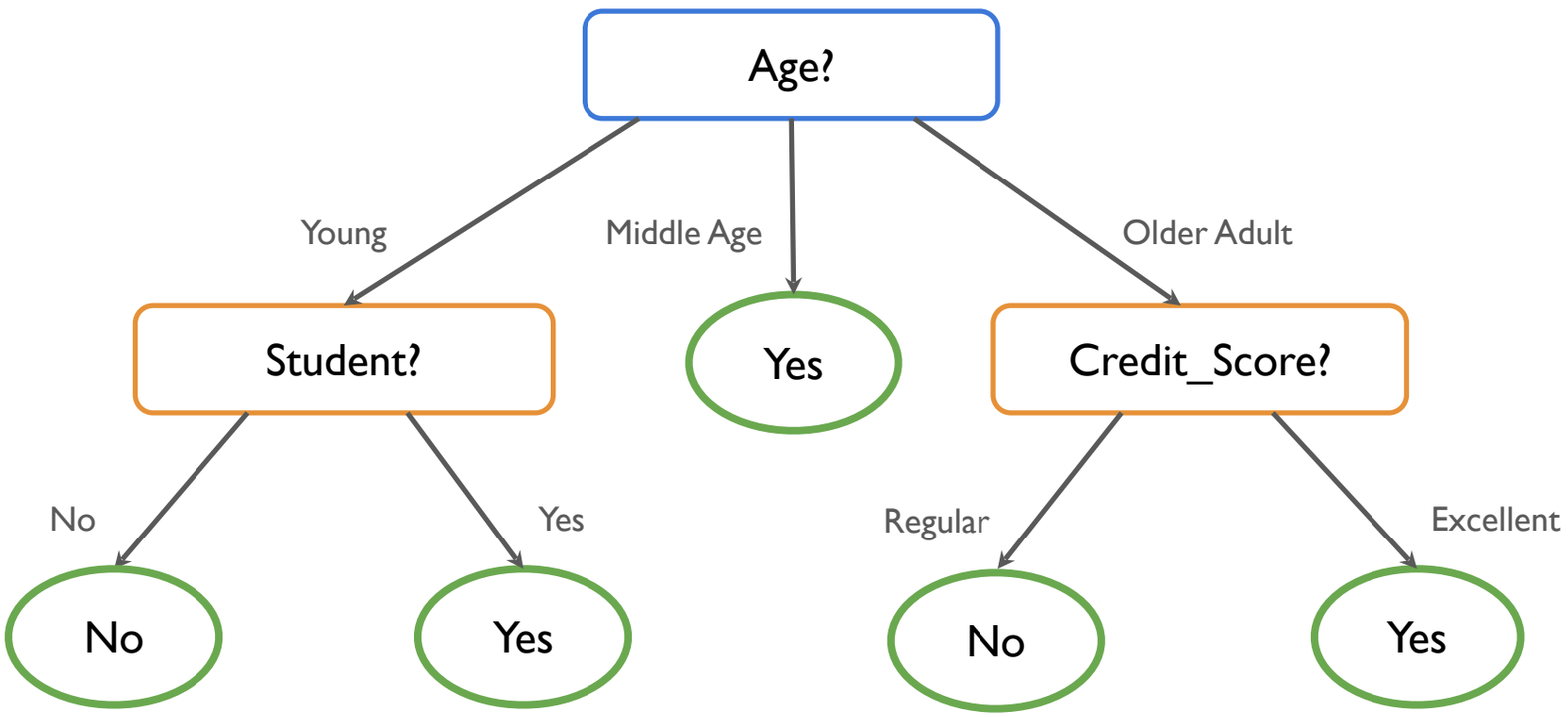


Decision Trees

How would the following instance be classified?

Instance to be classified:

Student	Age	Credit_Score	Will_Buy_Computer
No	Older Adult	Regular	??



Decision Trees

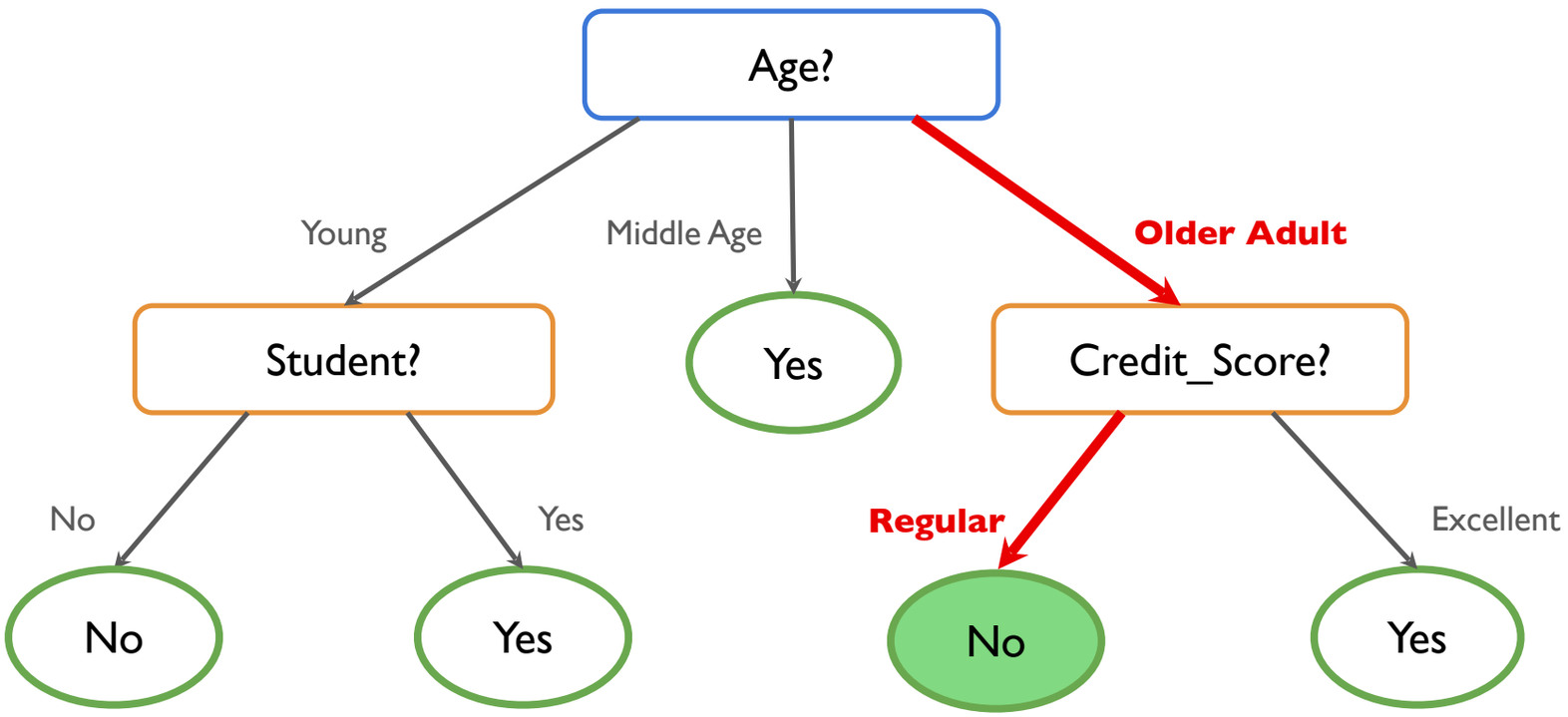
How would the following instance be classified?

Instance to be classified:

Student	Age	Credit_Score
No	Older Adult	Regular

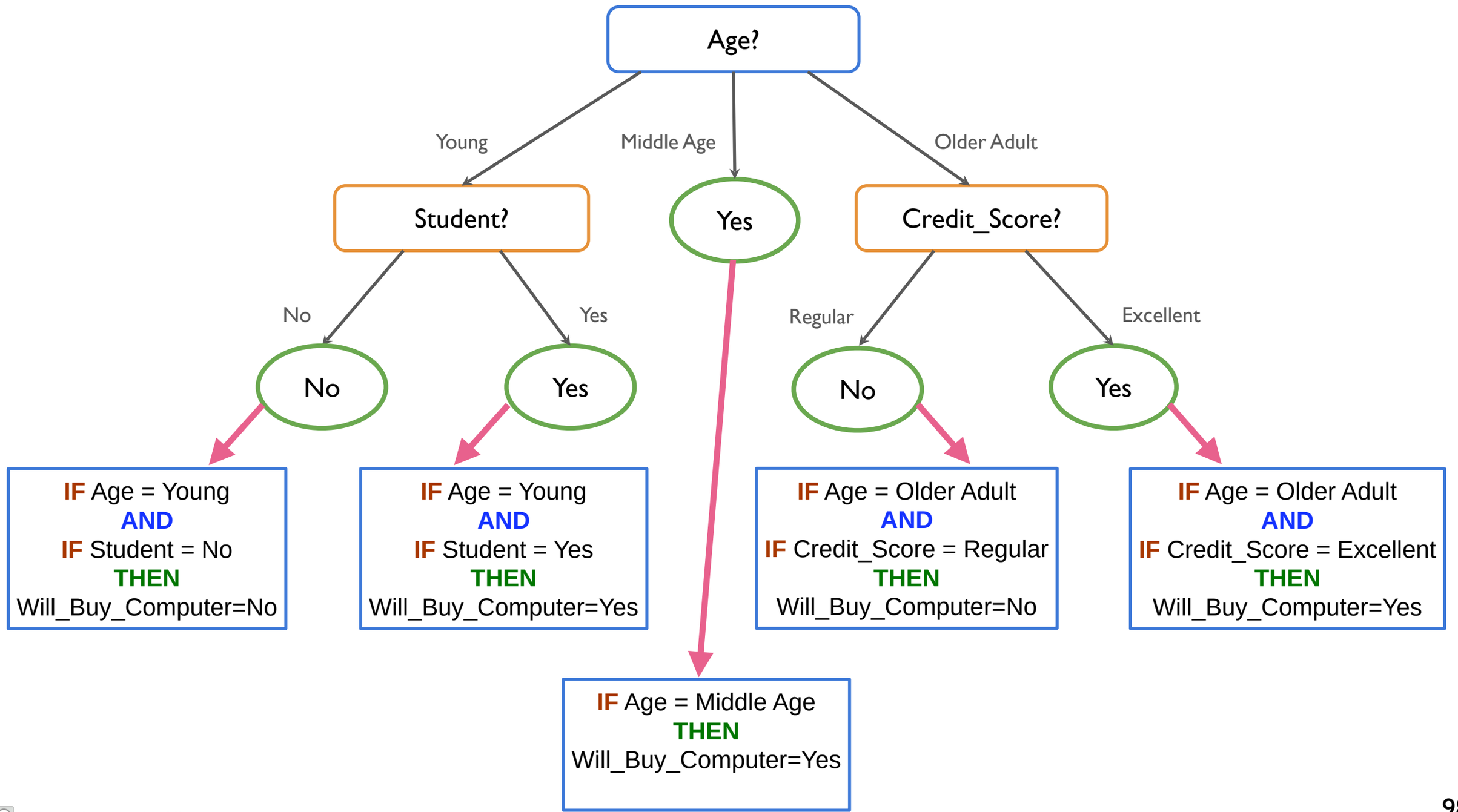
Predicted Class

Will_Buy_Computer
No



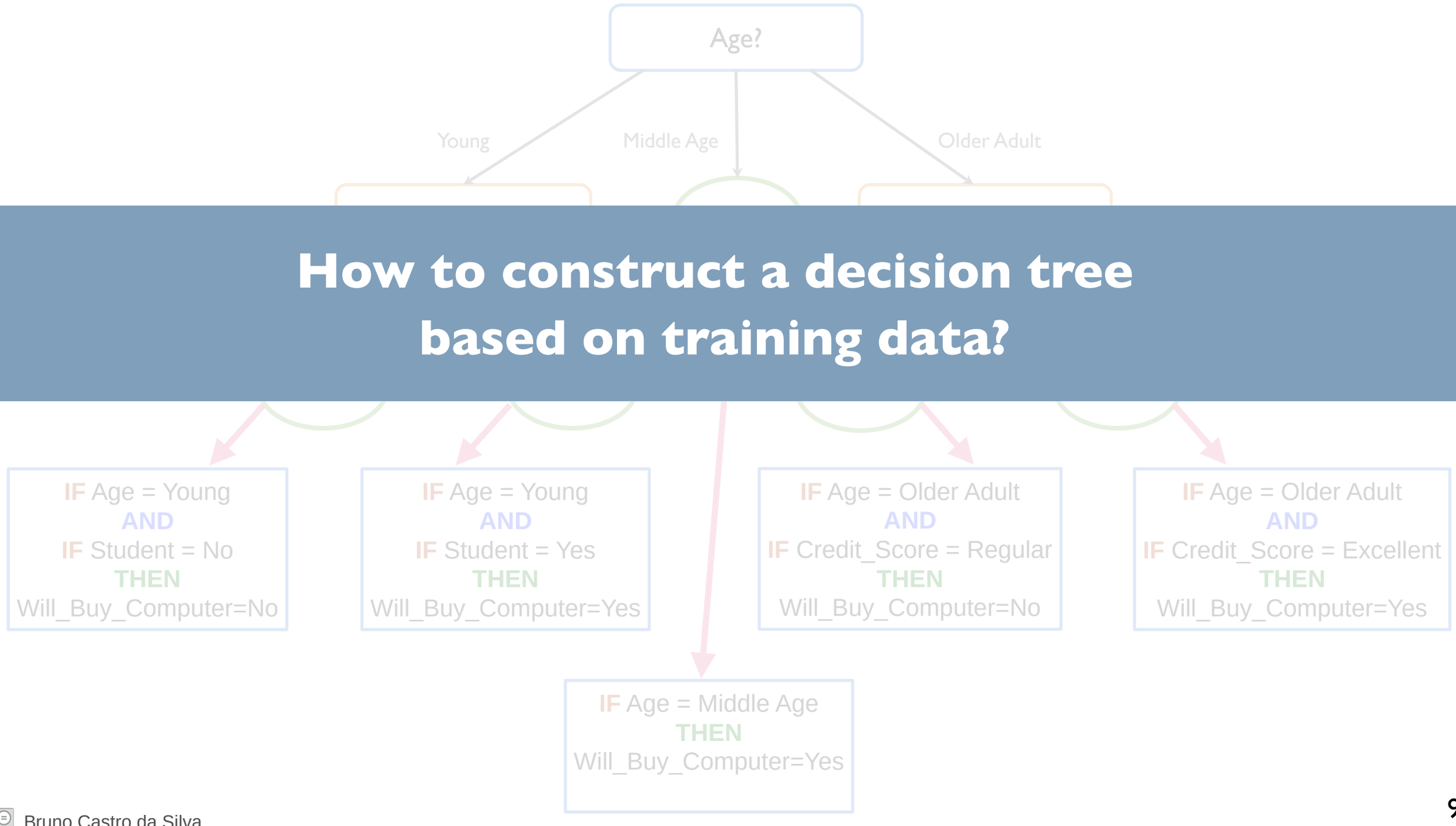
Decision Trees

Decision trees encode classification rules via (implicit) IF-ELSE statements



Decision Trees

Decision trees encode classification rules via (implicit) IF-ELSE statements



How to construct a decision tree based on training data?

Decision Trees

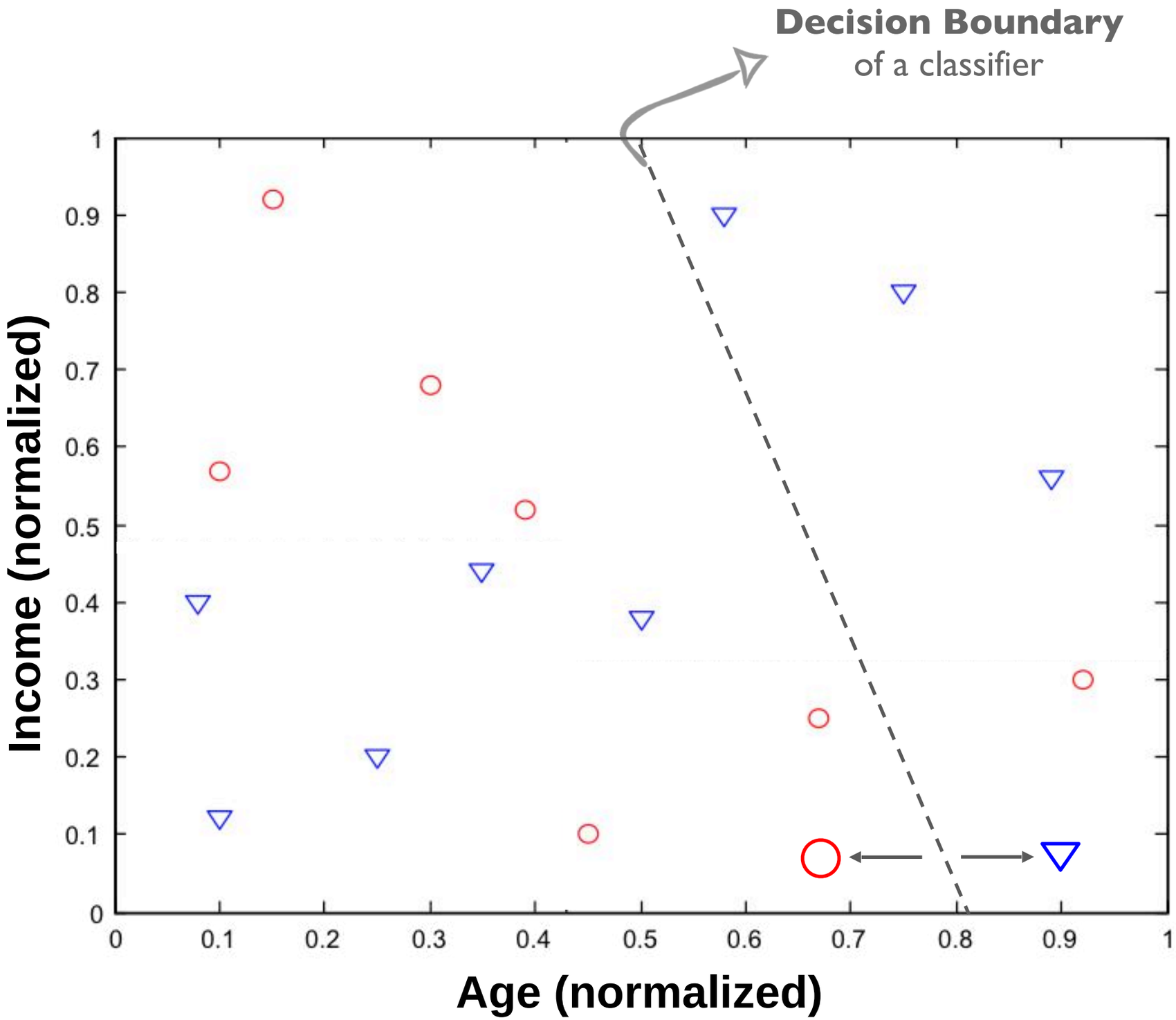
The predicted **class in each region** of space of attributes

depends on

the **percentage of instances of that class** in the region

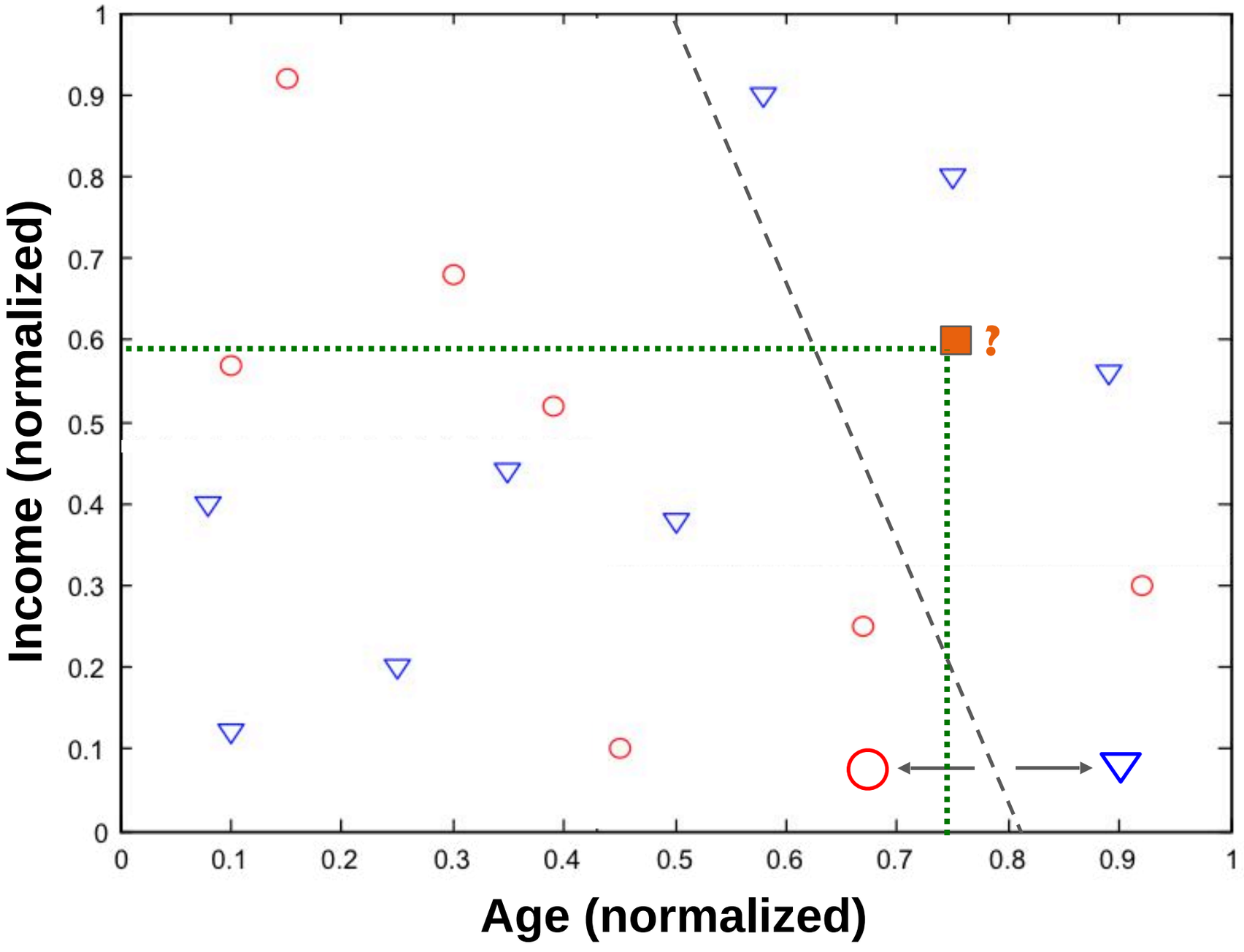
Will not repay loan

Will repay loan



Decision Trees

Which class is predicted for this instance?

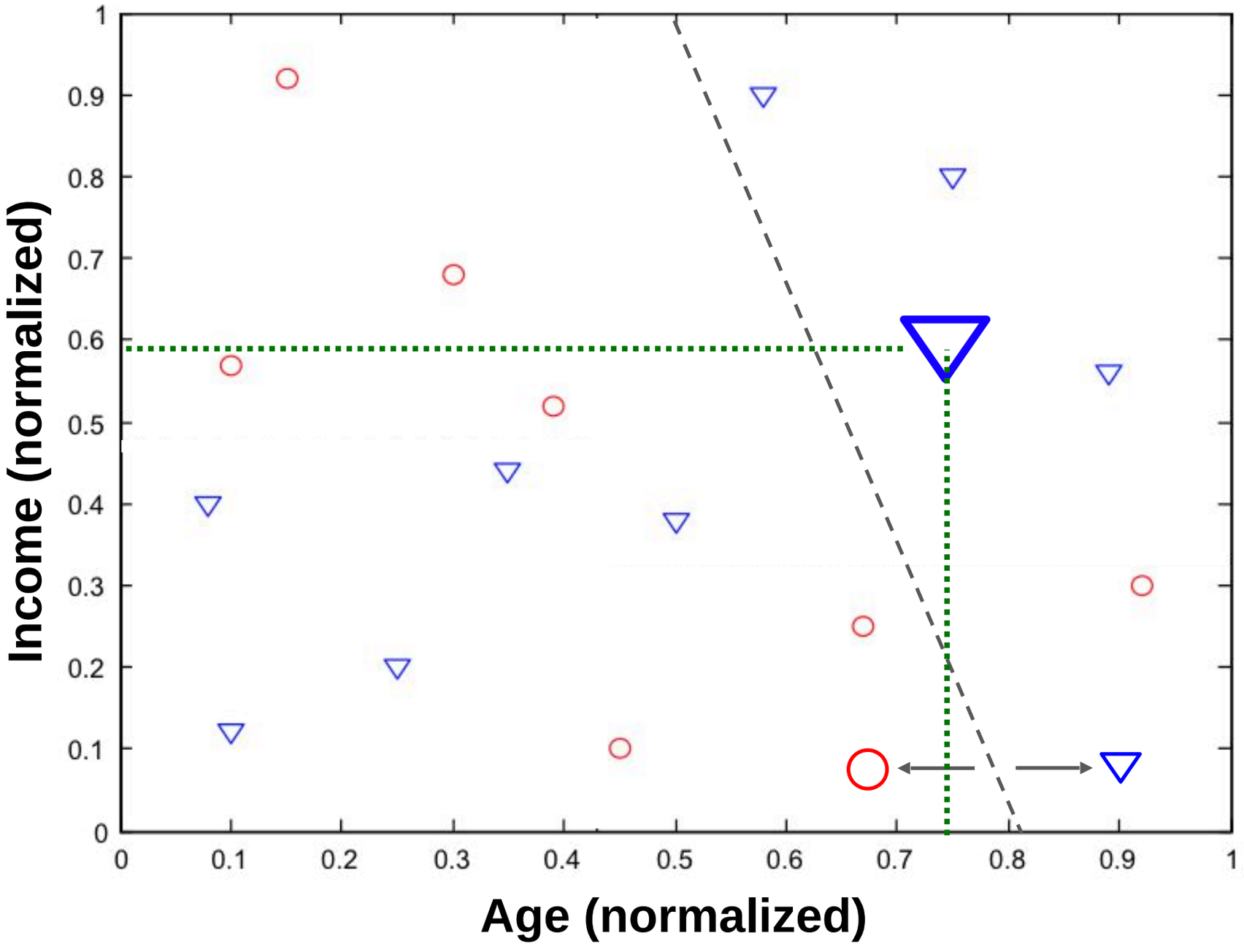


Will not repay loan

Will repay loan

Decision Trees

Which class is predicted for this instance?

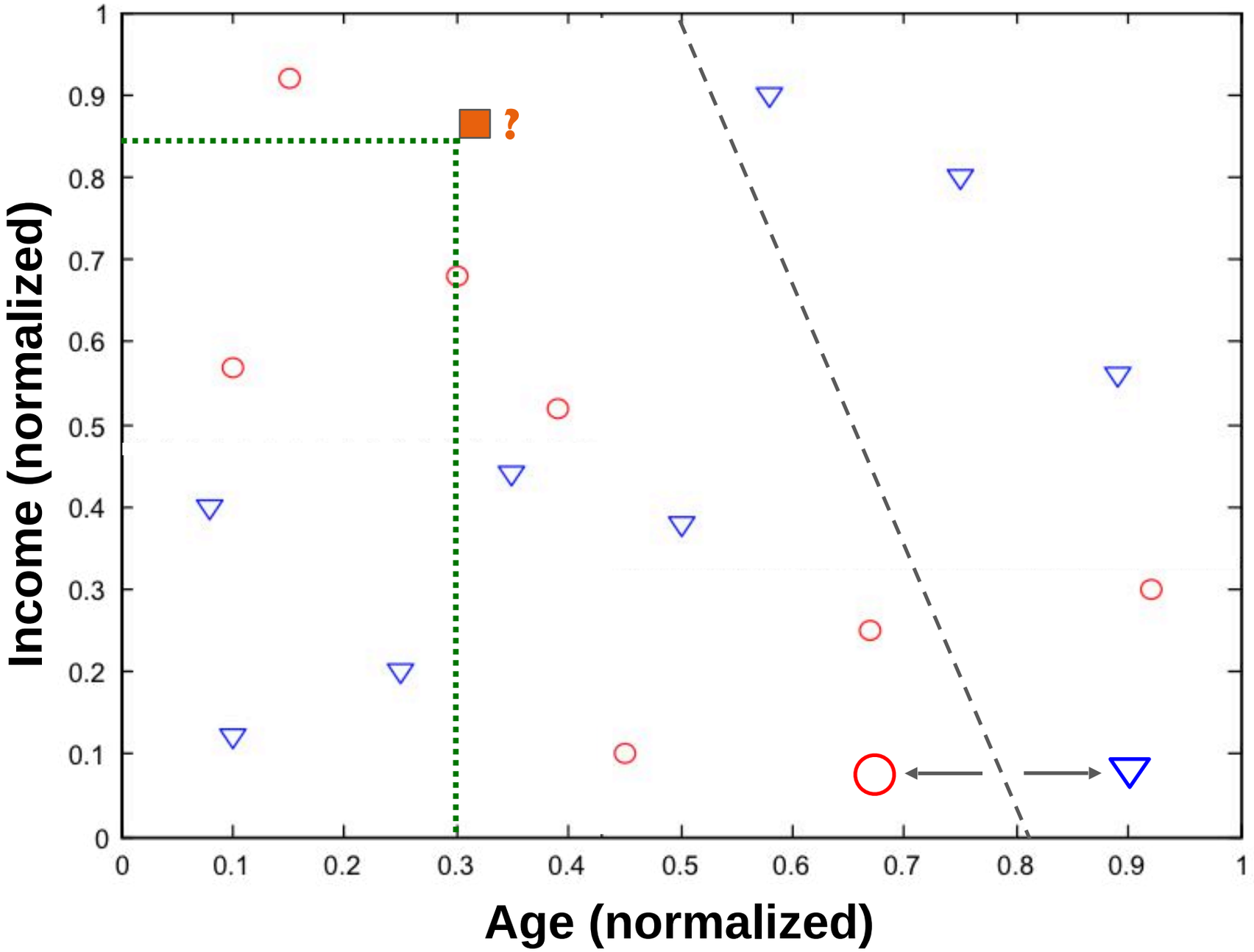


Will not repay loan

Will repay loan

Decision Trees

Which class is predicted for this instance?

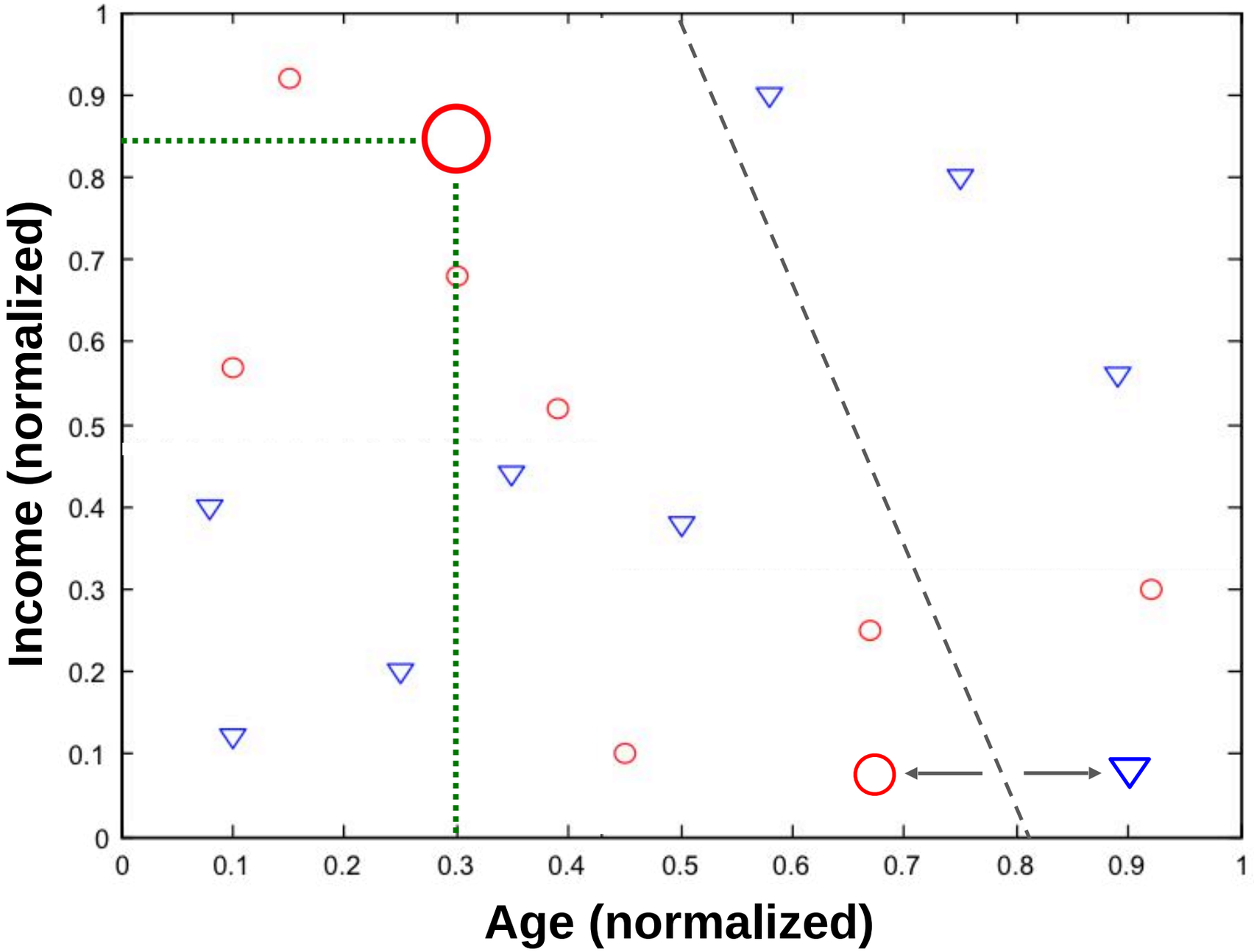


Will not repay loan

Will repay loan

Decision Trees

Which class is predicted for this instance?

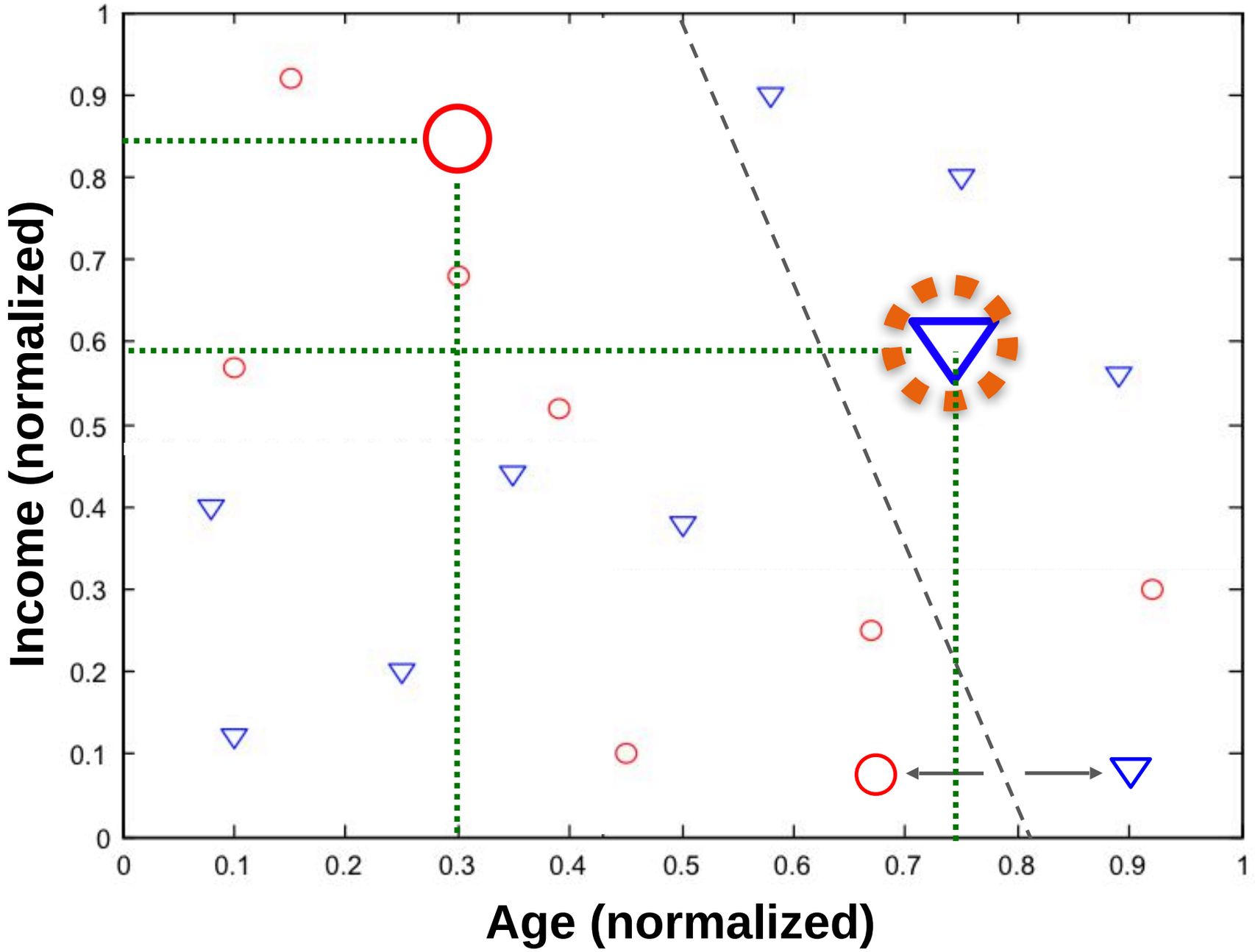


Will not repay loan

Will repay loan

Decision Trees

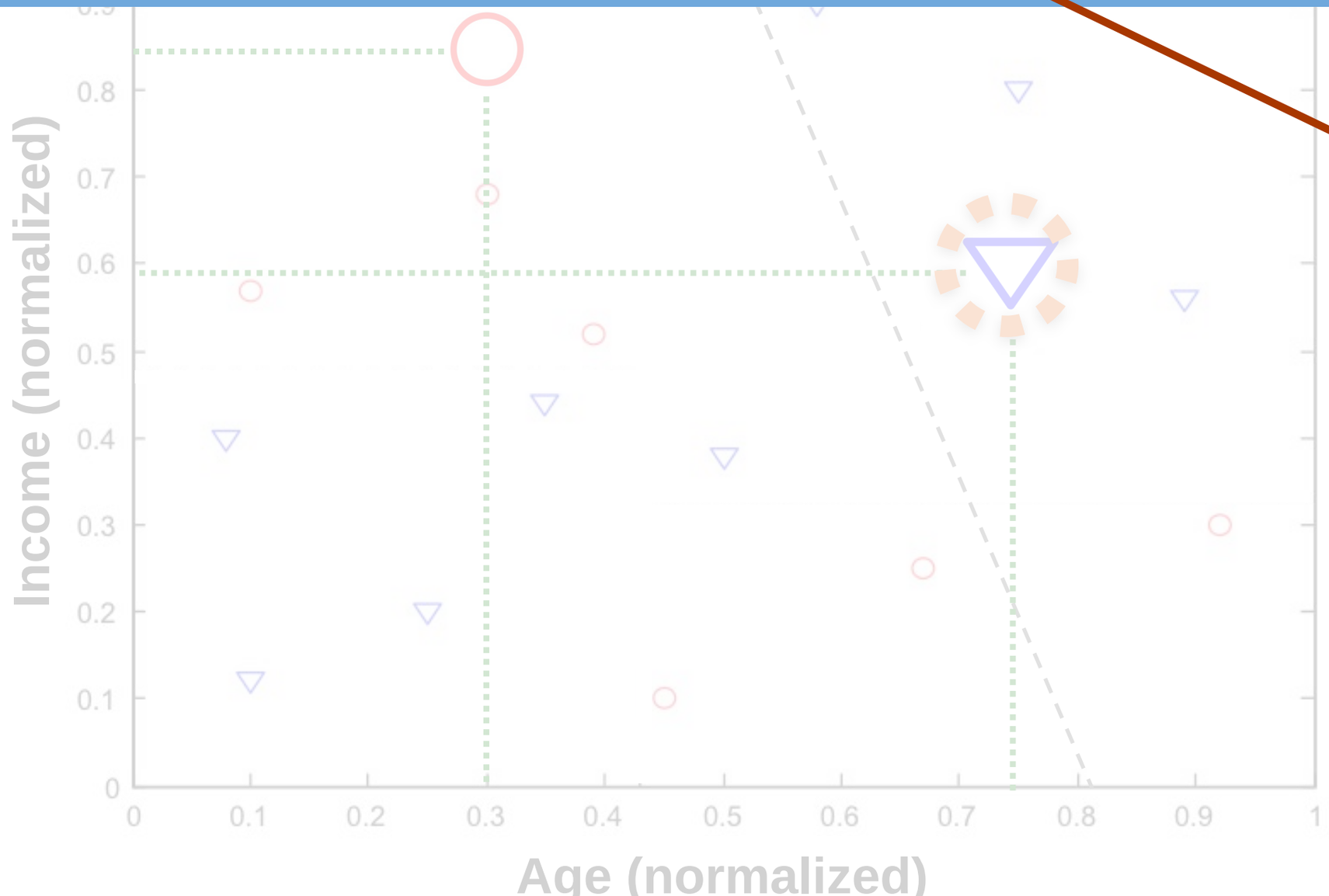
In which case do you think the classifier is more "certain" about its decision?



Decision Trees

If 100% of instances in a region belong to a same class,
we have 100% **certainty** when classifying a new instance in that region

Intuition: define decision boundaries by repeatedly partitioning the space of attributes to find regions that are as homogenous as possible (most instances belong to a same class)



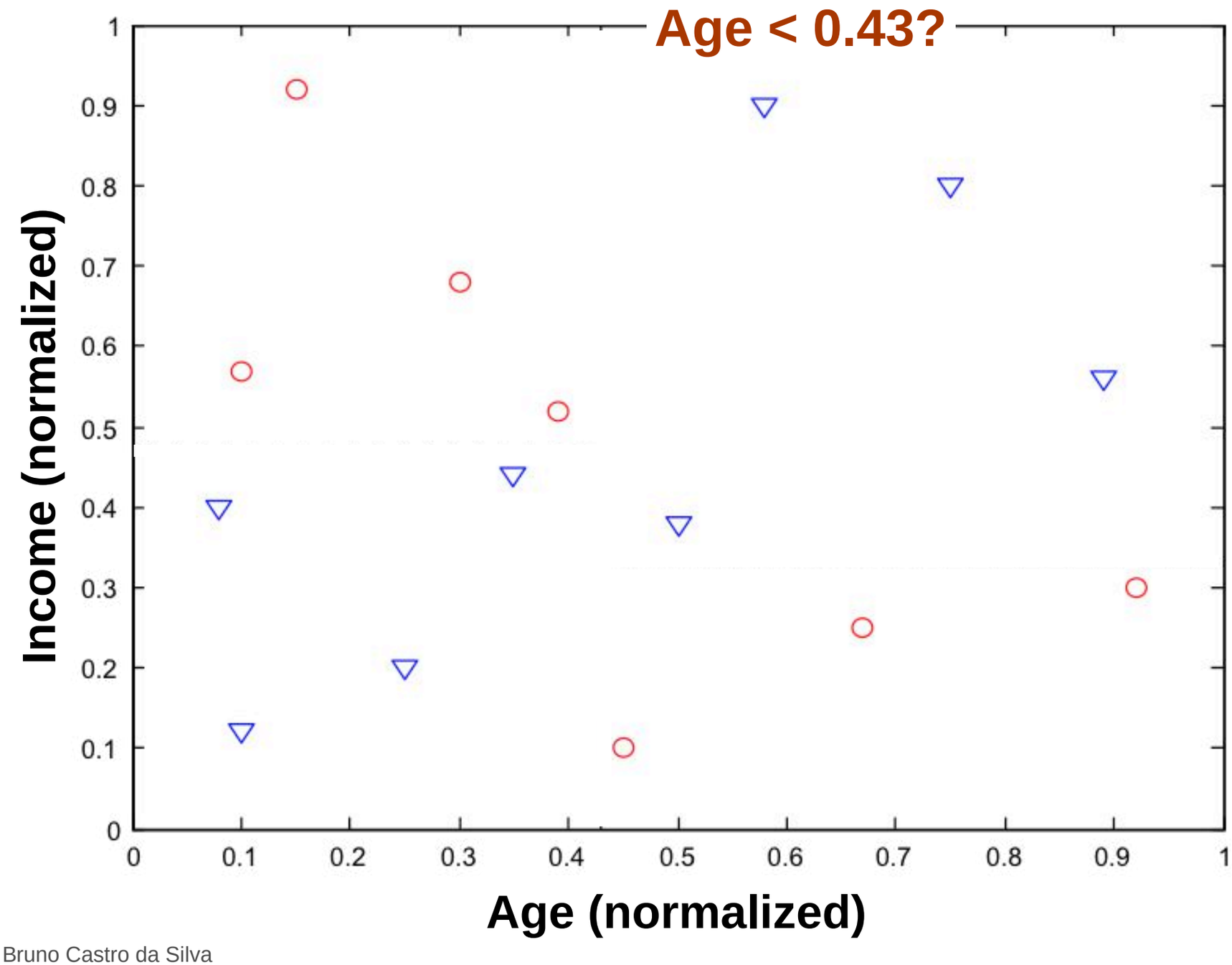
Testing the value of attributes

↓

Decision boundaries **parallel** to the axes

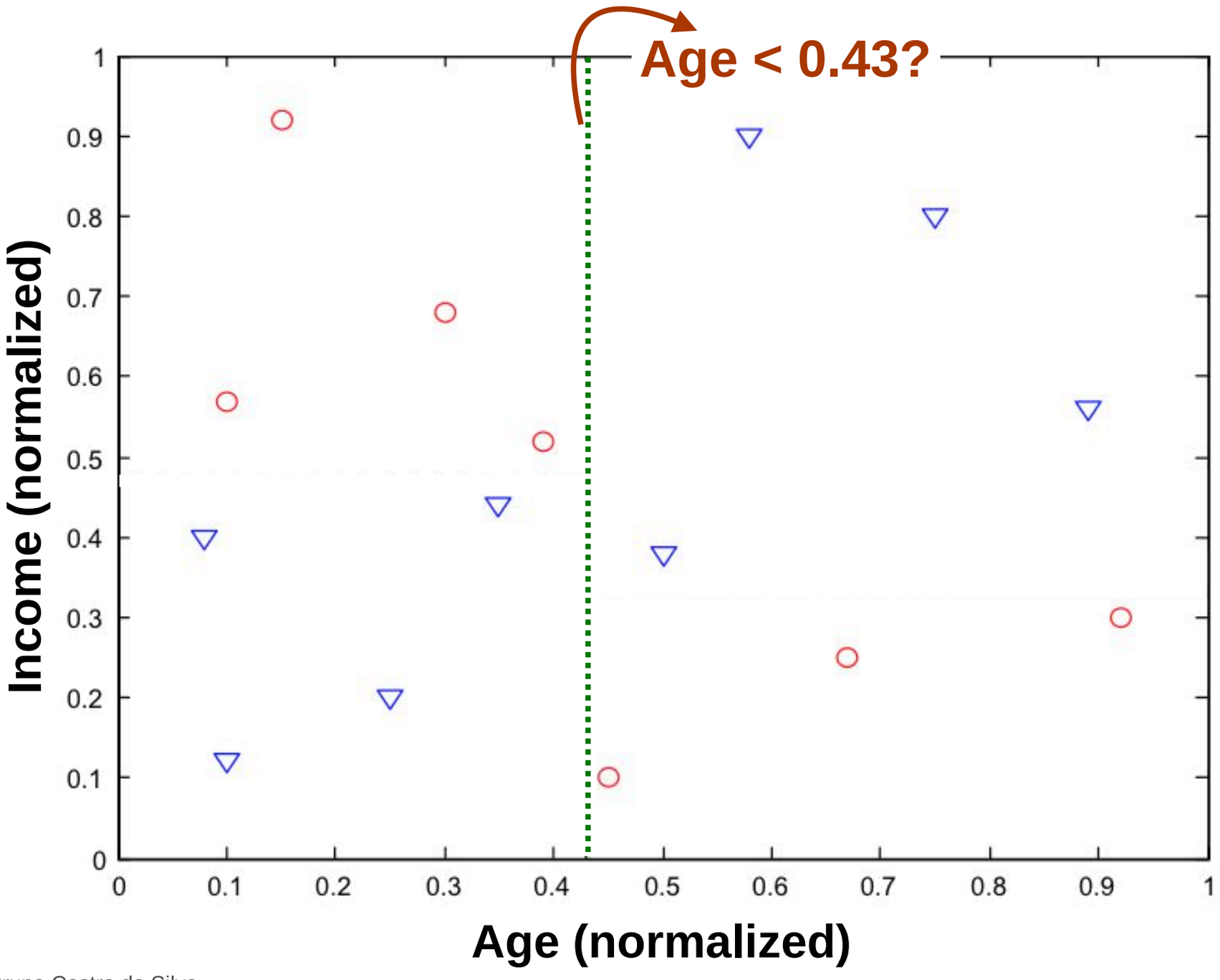
Decision Trees

Intuition: define decision boundaries by repeatedly partitioning the space of attributes to find regions that are as homogenous as possible (most instances belong to a same class)



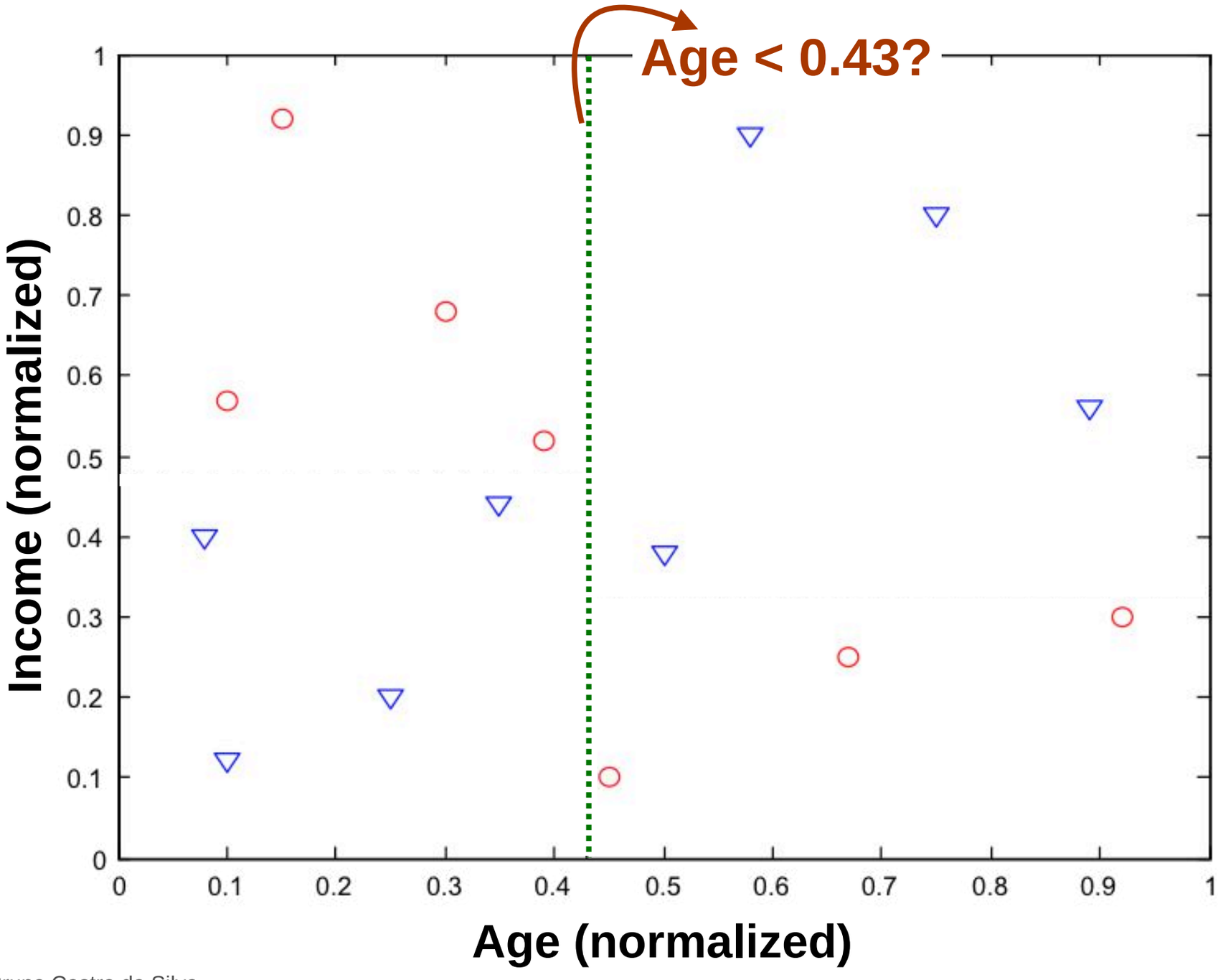
Decision Trees

Intuition: define decision boundaries by repeatedly partitioning the space of attributes to find regions that are as homogenous as possible (most instances belong to a same class)



Decision Trees

Intuition: define decision boundaries by repeatedly partitioning the space of attributes to find regions that are as homogenous as possible (most instances belong to a same class)

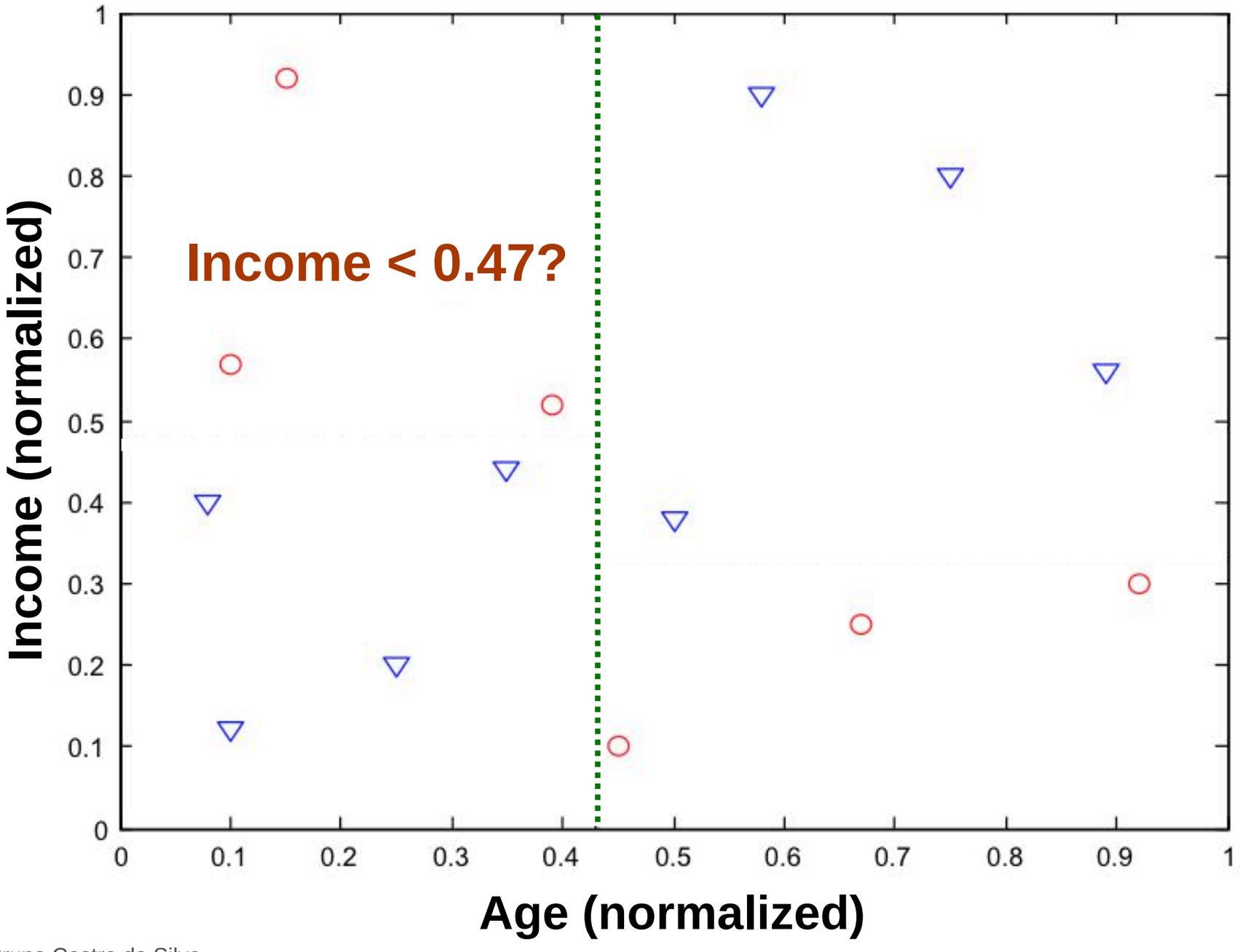


Regions/ partitions

Induced by the tests we perform on attributes

Decision Trees

Intuition: define decision boundaries by repeatedly partitioning the space of attributes to find regions that are as homogenous as possible (most instances belong to a same class)

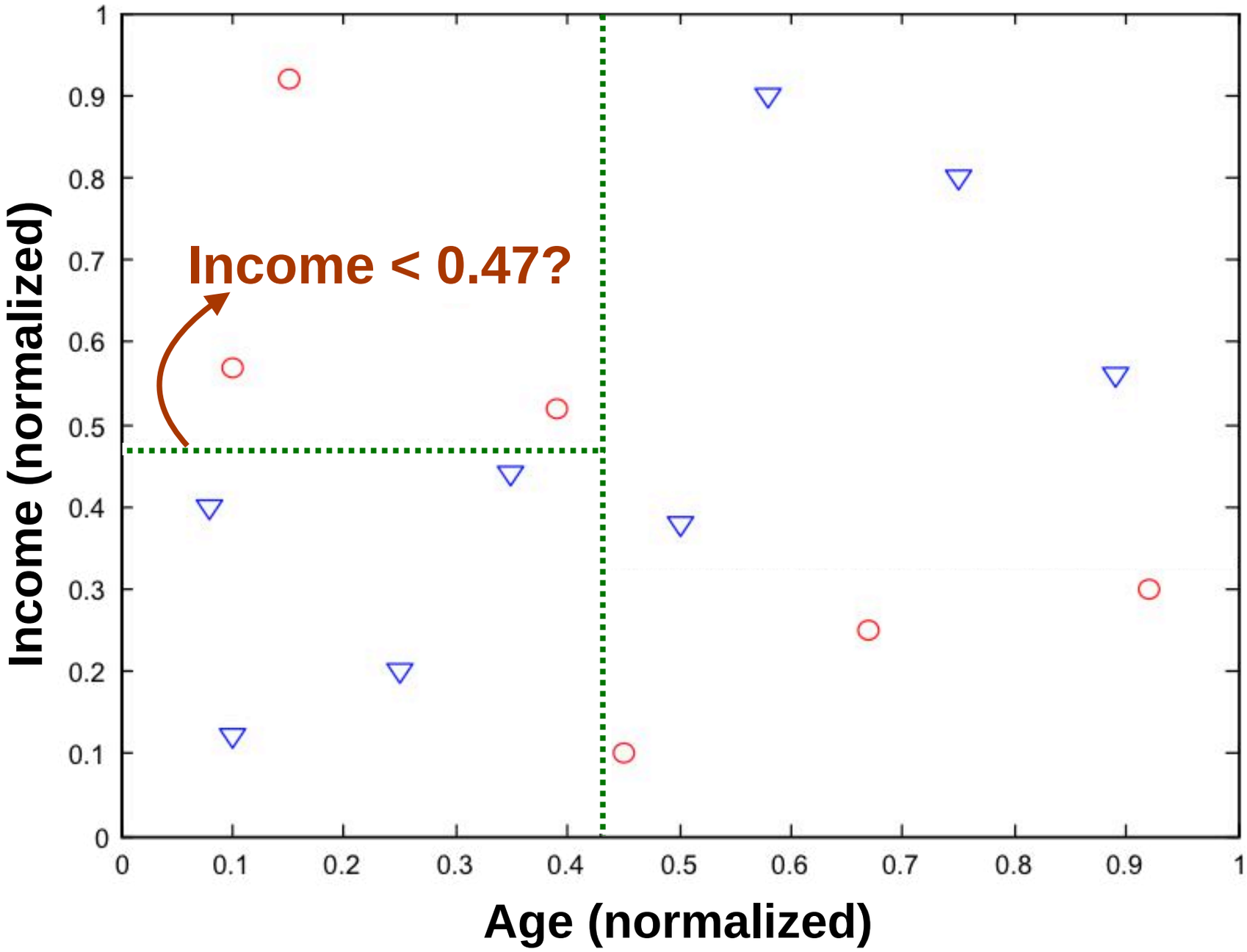


Regions/ partitions

Induced by the tests we perform on attributes

Decision Trees

Intuition: define decision boundaries by repeatedly partitioning the space of attributes to find regions that are as homogenous as possible (most instances belong to a same class)

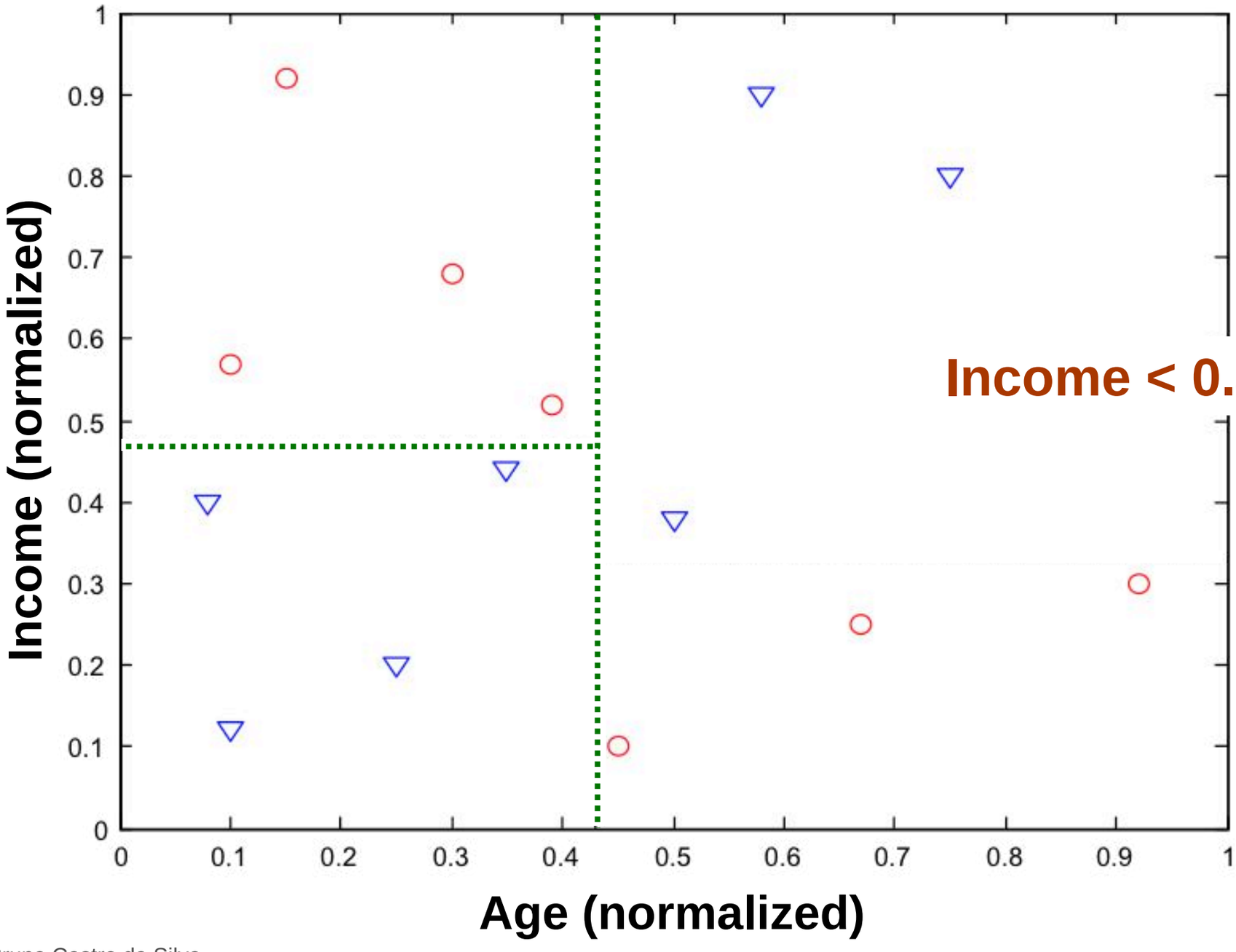


Regions/ partitions

Induced by the tests we perform on attributes

Decision Trees

Intuition: define decision boundaries by repeatedly partitioning the space of attributes to find regions that are as homogenous as possible (most instances belong to a same class)

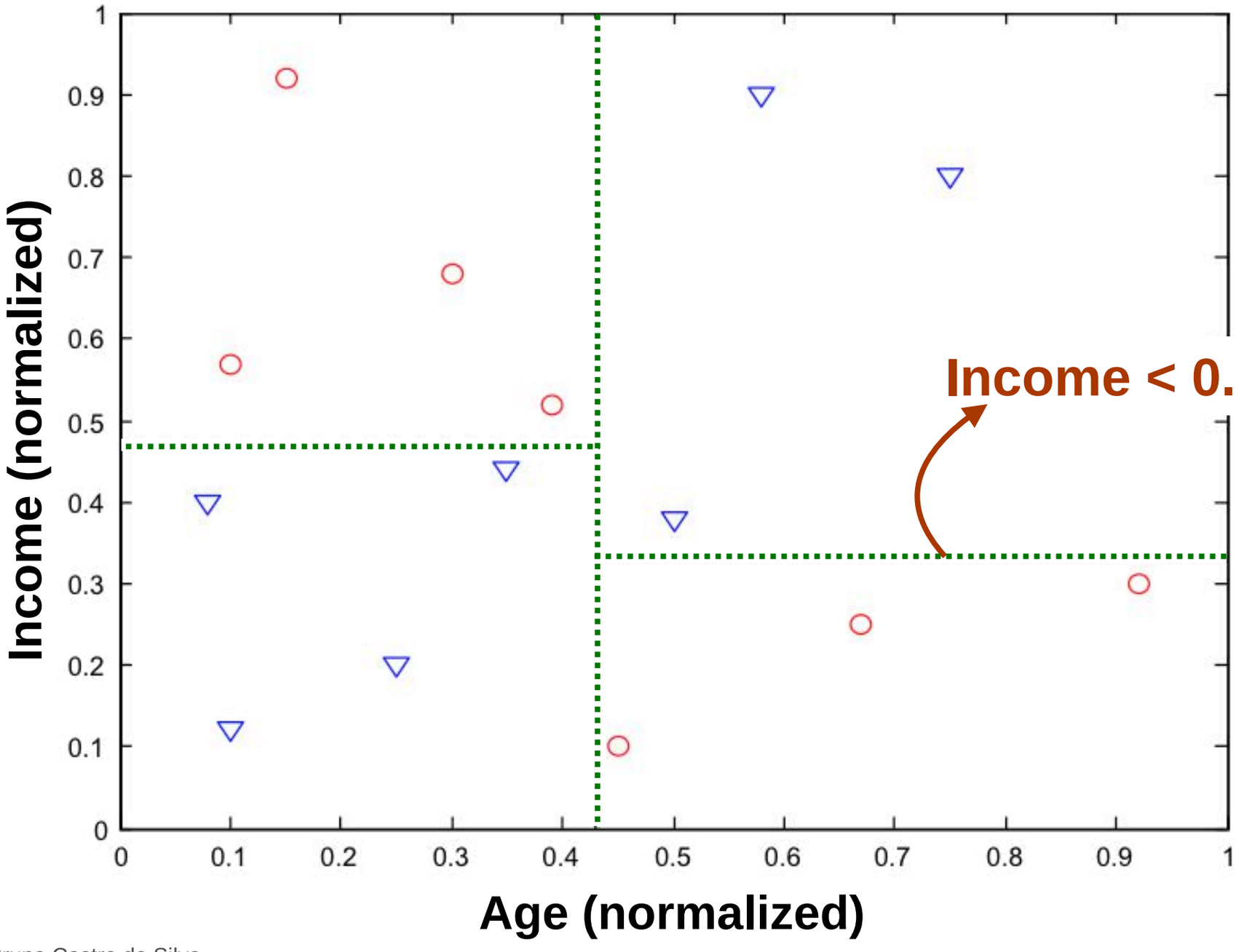


Regions/ partitions

Induced by the tests we perform on attributes

Decision Trees

Intuition: define decision boundaries by repeatedly partitioning the space of attributes to find regions that are as homogenous as possible (most instances belong to a same class)

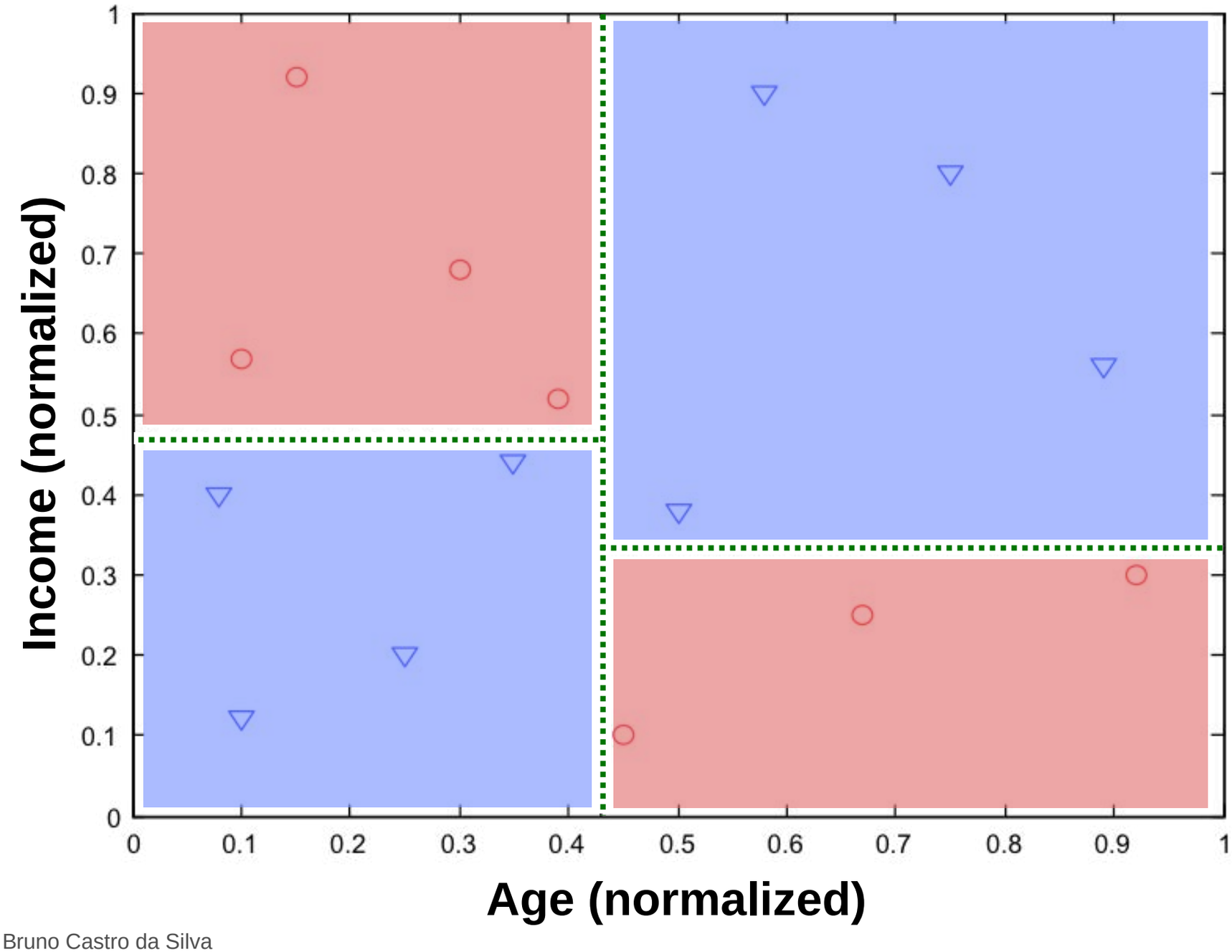


Regions/ partitions

Induced by the tests we perform on attributes

Decision Trees

Intuition: define decision boundaries by repeatedly partitioning the space of attributes to find regions that are as homogenous as possible (most instances belong to a same class)

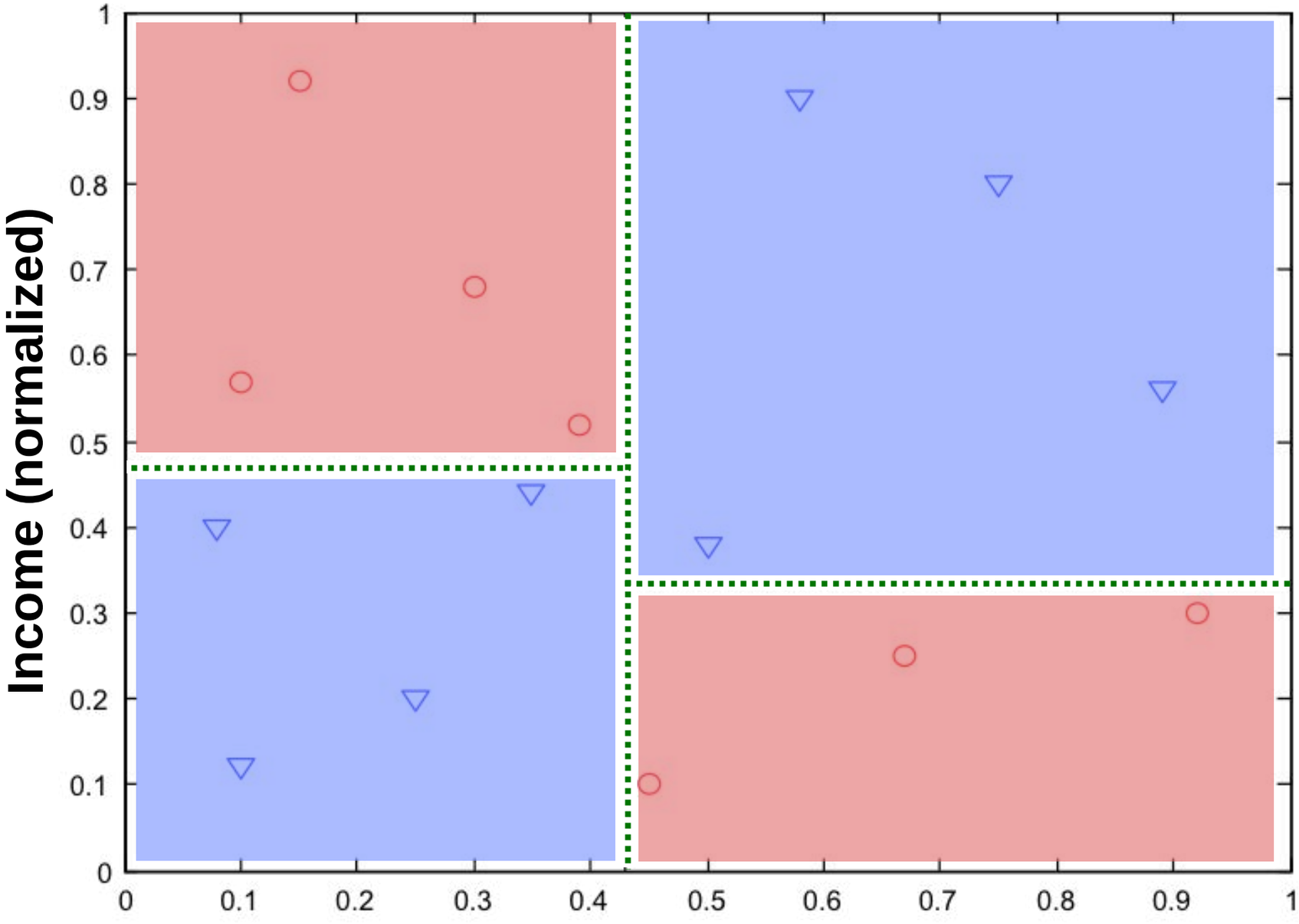


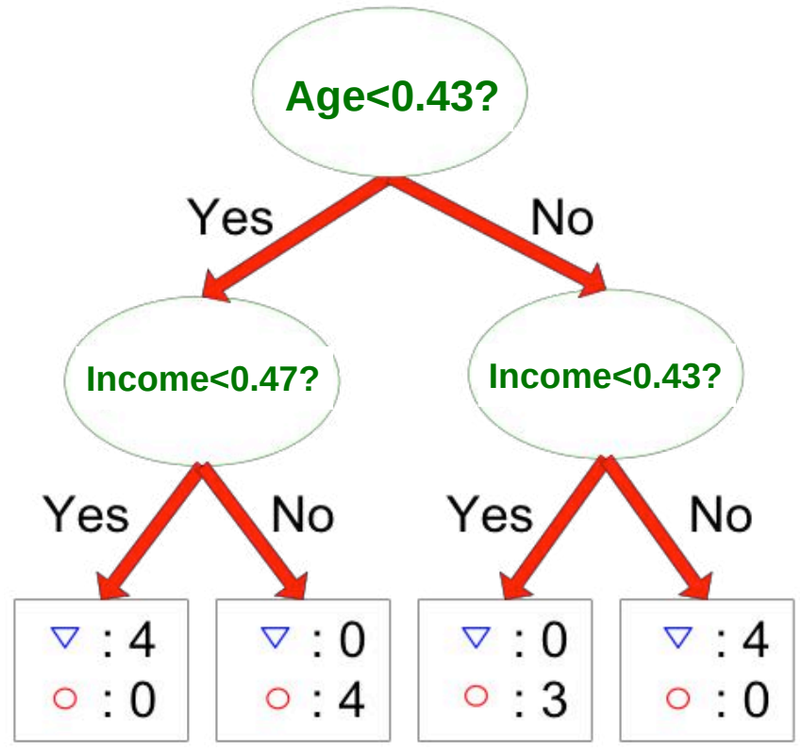
Regions/ partitions

Induced by the tests we perform on attributes

Decision Trees

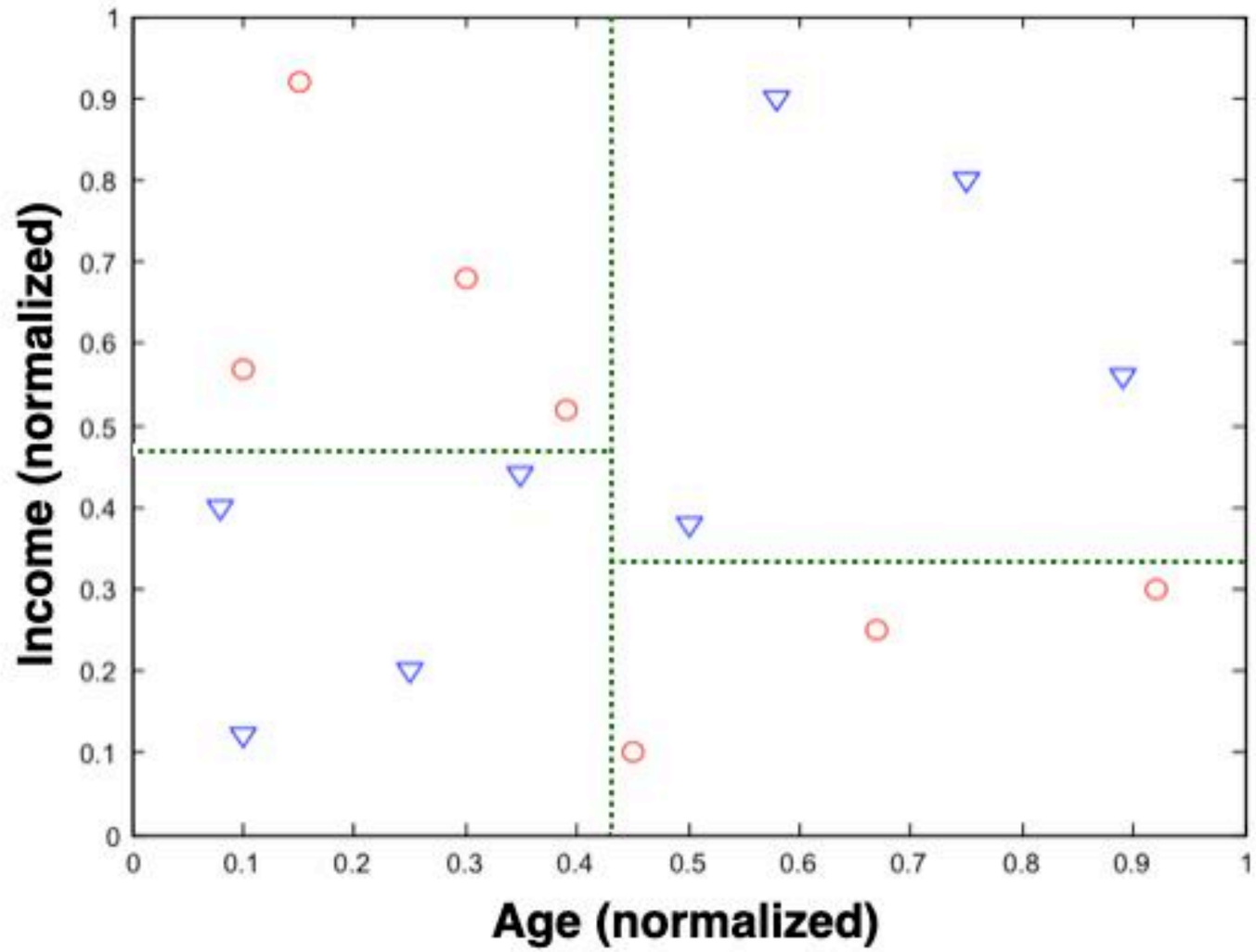
How would a programmer implement this decision tree?

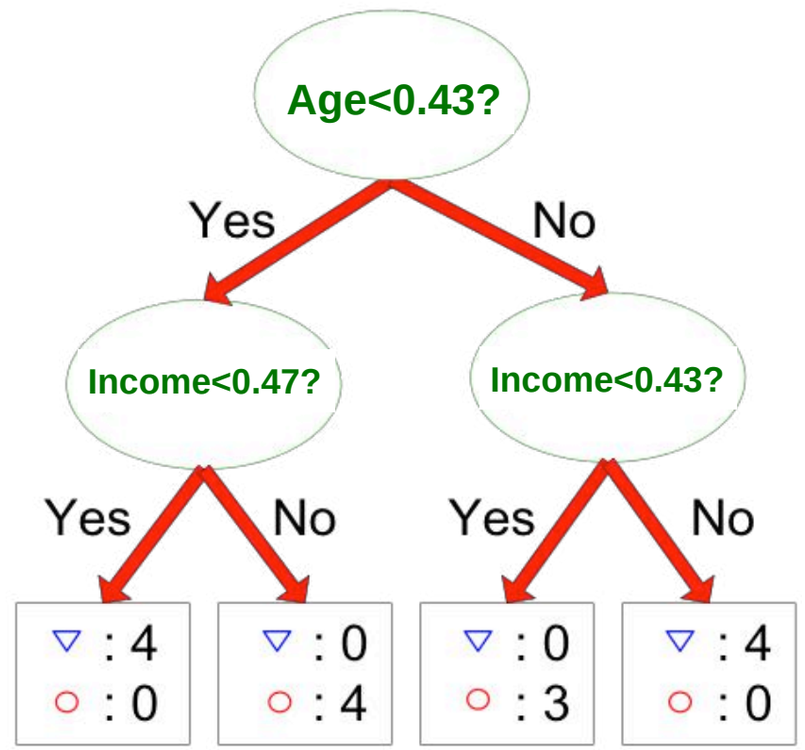




Decision Trees

- General idea
  - **Repeat** "until classifier is good enough"
    - Select the "best" attribute
    - Split the instances based on the value of this attribute (new decision rule)

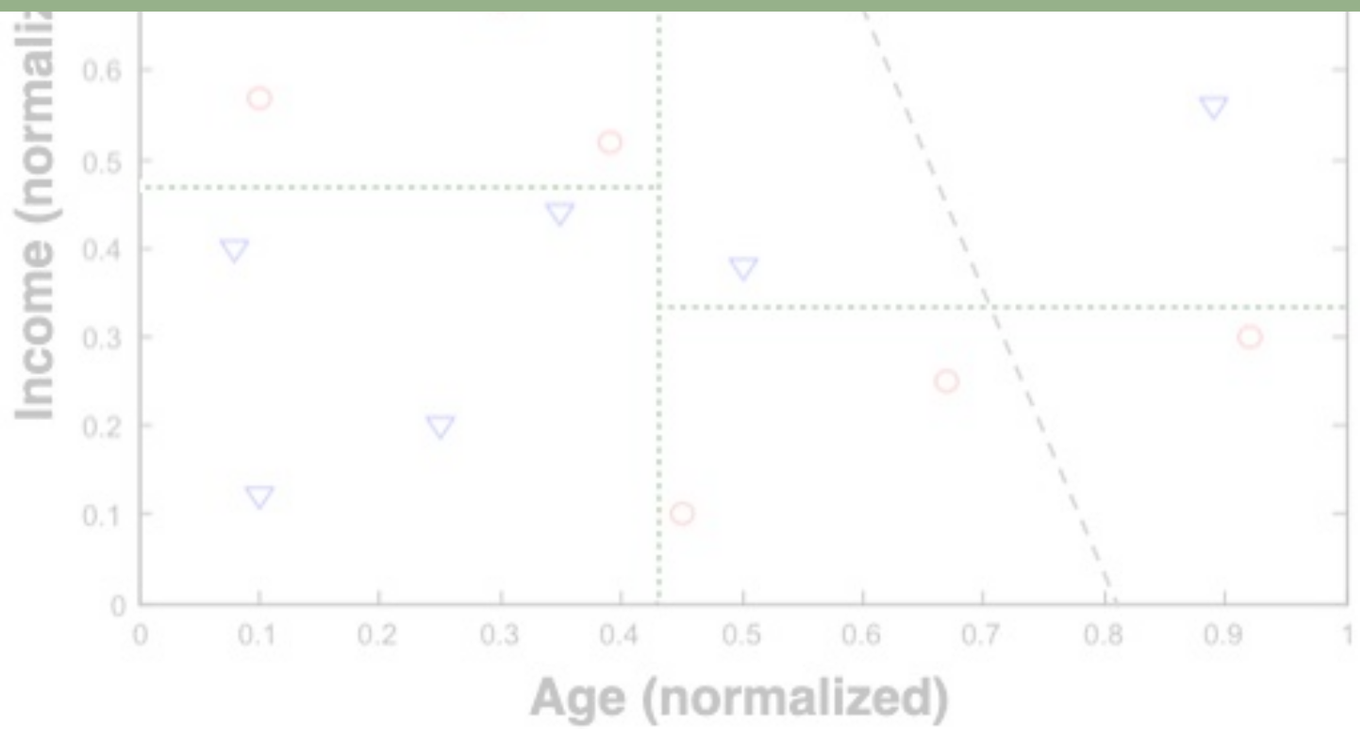




Decision Trees

- General idea
  - **Repeat** "until classifier is good enough"
    - Select the "best" attribute
    - Split the instances based on the value of this attribute (new decision rule)

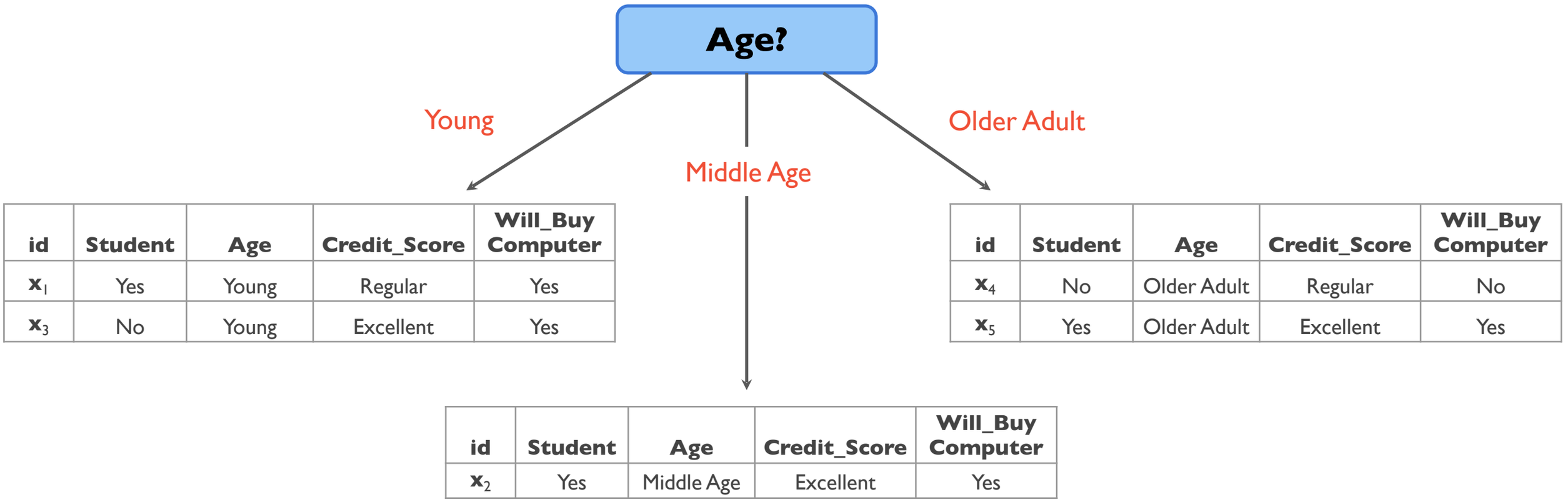
Divide-and-Conquer Strategy



Decision Trees

Divide-and-Conquer over data instances

id	Student	Age	Credit_Score	Will_Buy_Computer
x_1	Yes	Young	Regular	Yes
x_2	Yes	Middle Age	Excellent	Yes
x_3	No	Young	Excellent	No
x_4	No	Older Adult	Regular	No
x_5	Yes	Older Adult	Excellent	Yes



id	Student	Age	Credit_Score	Will_Buy Computer
x_1	Yes	Young	Regular	Yes
x_3	No	Young	Excellent	Yes

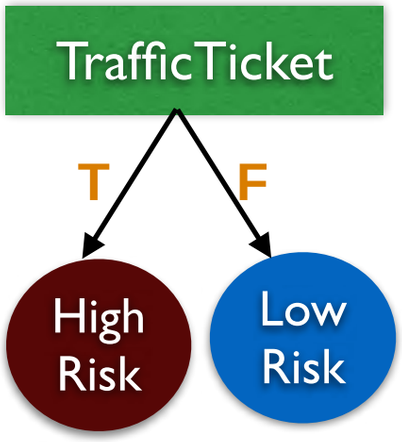
id	Student	Age	Credit_Score	Will_Buy Computer
x_4	No	Older Adult	Regular	No
x_5	Yes	Older Adult	Excellent	Yes

id	Student	Age	Credit_Score	Will_Buy Computer
x_2	Yes	Middle Age	Excellent	Yes

Learning a Decision Tree

- How to train a decision tree that correctly classifies these examples?

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	**High Risk**
Bob	30	M	No	**Low Risk**



Which attribute to test to determine a driver's label/class?

Learning a Decision Tree

- How to train a decision tree that correctly classifies these examples?

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	**High Risk**
Bob	30	M	No	**Low Risk**

But what if the training set is not so "well behaved"?

Learning a Decision Tree

- How to train a decision tree that correctly classifies these examples?

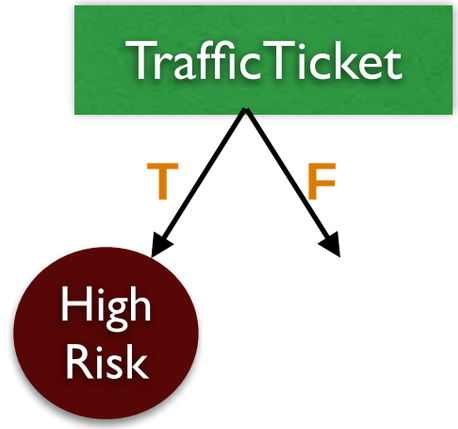
Name	*Age*	*Gender*	*TrafficTicket*	Class: High-Risk Driver
John	43	M	Yes	**High Risk**
Peter	18	M	No	**High Risk**
Anna	35	F	No	**Low Risk**
Paula	19	F	No	**High Risk**
Mark	90	M	Yes	**High Risk**
Marisa	19	F	Yes	**High Risk**
Bob	30	M	No	**Low Risk**

But what if the training set is not so "well behaved"?

Learning a Decision Tree

- How to train a decision tree that correctly classifies these examples?

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**	←

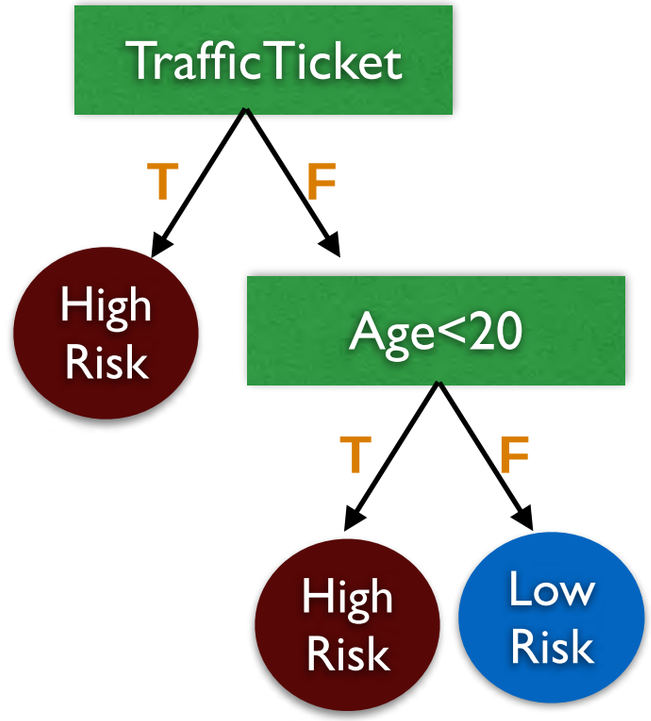


But what if the training set is not so "well behaved"?

Learning a Decision Tree

- How to train a decision tree that correctly classifies these examples?

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**

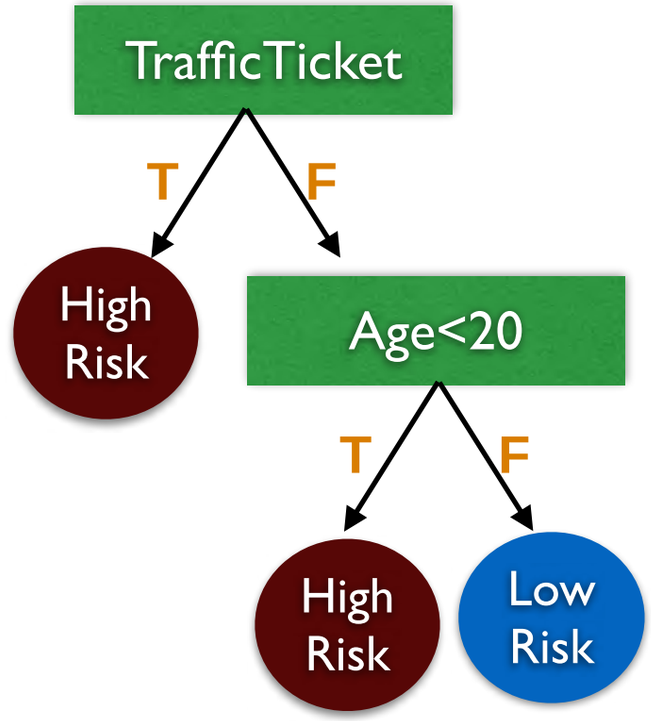


But what if the training set is not so "well behaved"?

Learning a Decision Tree

- How to train a decision tree that correctly classifies these examples?

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**

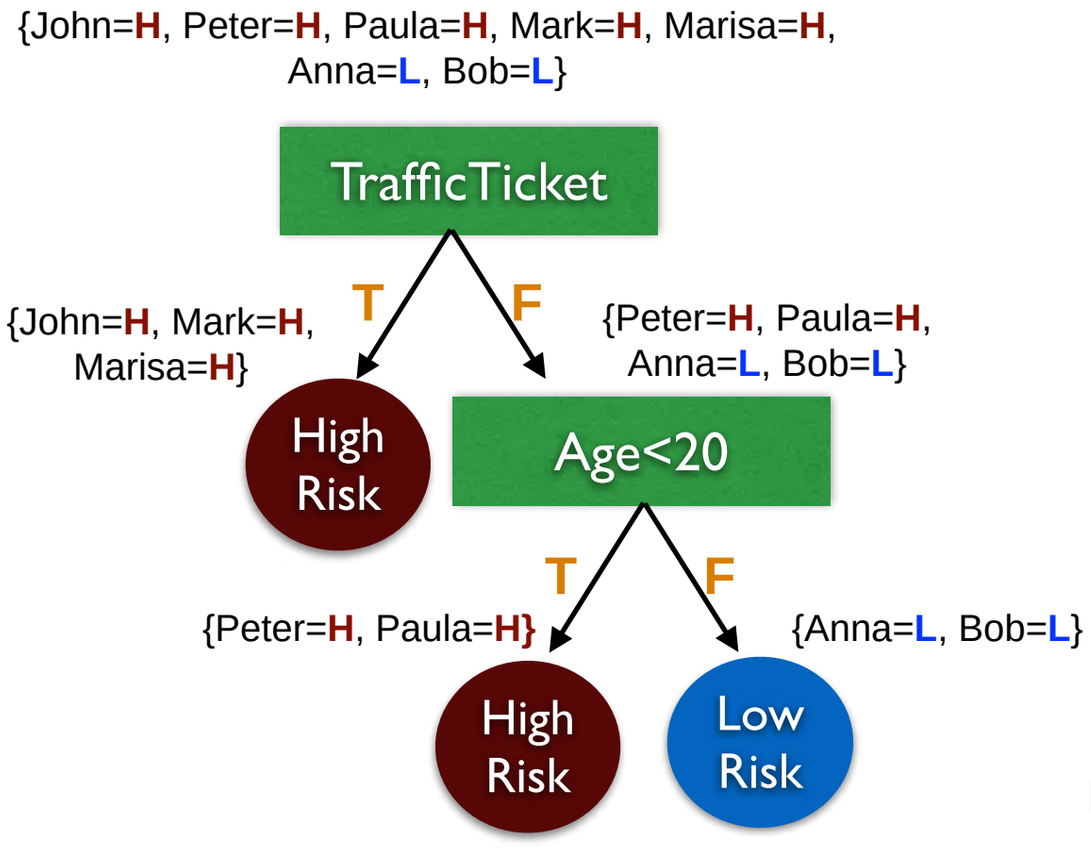


How to determine which attributes to test at each step along the tree?

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/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**



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/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)

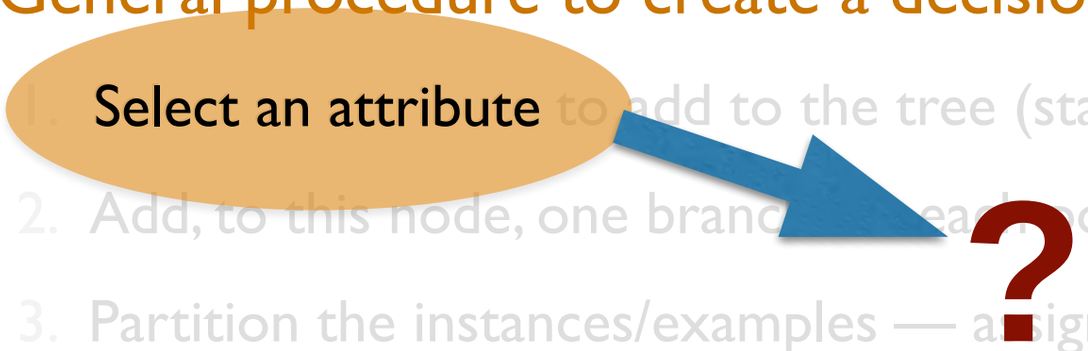
When to stop?

a. When all instances of a given partition/node belong to the same class
   Then, create a leaf node labeled with that class
b. When there are no more attributes that can be tested
   or
   When a partition is empty (i.e., there are no instances associated with it)
   Then, create a leaf node labeled with the majority class among the instances

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/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)

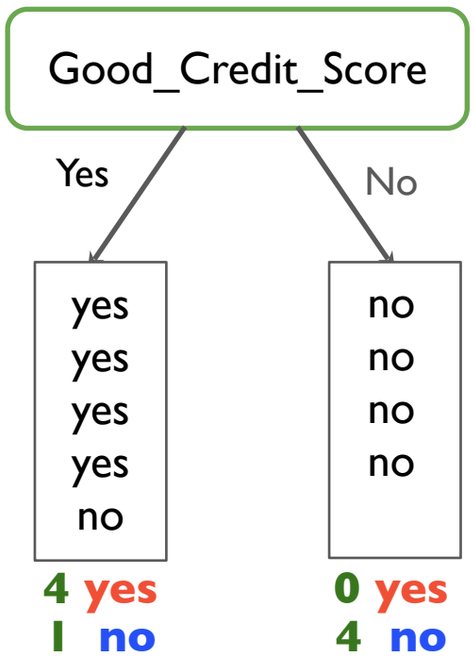


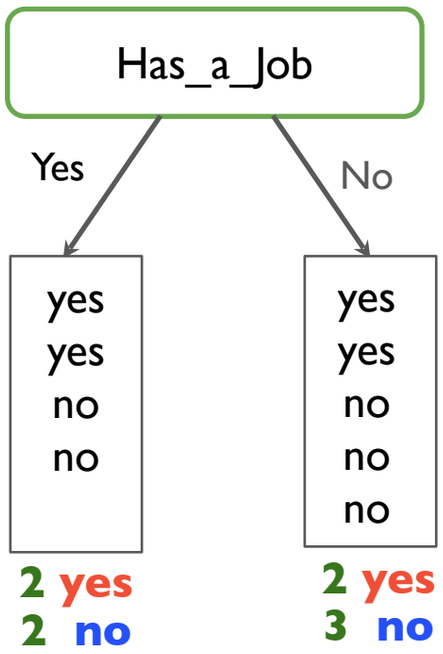
When to stop?

a. When all instances of a given partition/node belong to the same class
   Then, create a leaf node labeled with that class
b. When there are no more attributes that can be tested
   or
   When a partition is empty (i.e., there are no instances associated with it)
   Then, create a leaf node labeled with the majority class among the instances

Selecting an Attribute to Test

- Criterion/heuristic for selecting which attribute to test
  - Most informative attribute
    - attribute that best splits instances according to their classes
  - 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 homogenous as possible
      - all instances in that partition belong to the same class (in that case, add new leaf node)





"Good_Credit_Score?" seems to be more informative than "Has_a_Job"

Selecting an Attribute to Test

- Criterion/heuristic for selecting which attribute to test
  - Most informative attribute
    - attribute that best splits instances according to their classes
  - 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 homogenous as possible
      - all instances in that partition belong to the same class (in that case, add new leaf node)

"Goodness of Split"

"Good_Credit_Score?" seems to be more informative than "Has_a_Job"

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 homogenous 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?

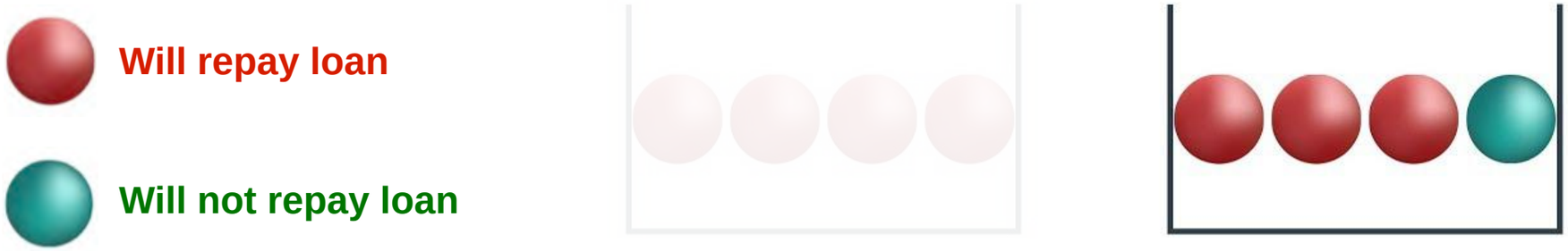
Do we have a good idea about whether the person is going to repay the loan or not?

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 homogenous 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?

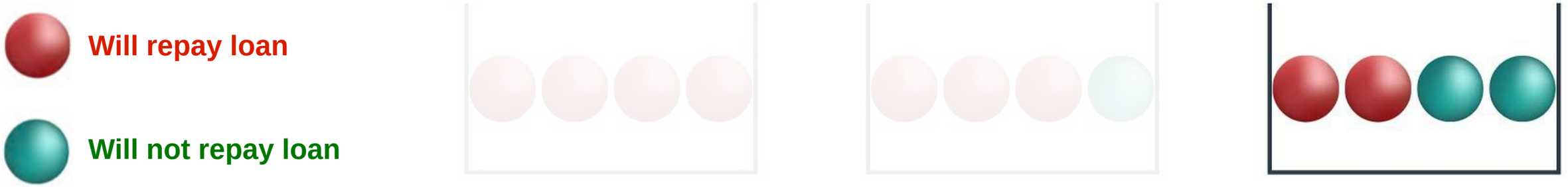
Do we have a good idea about whether the person is going to repay the loan or not?

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 homogenous 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?

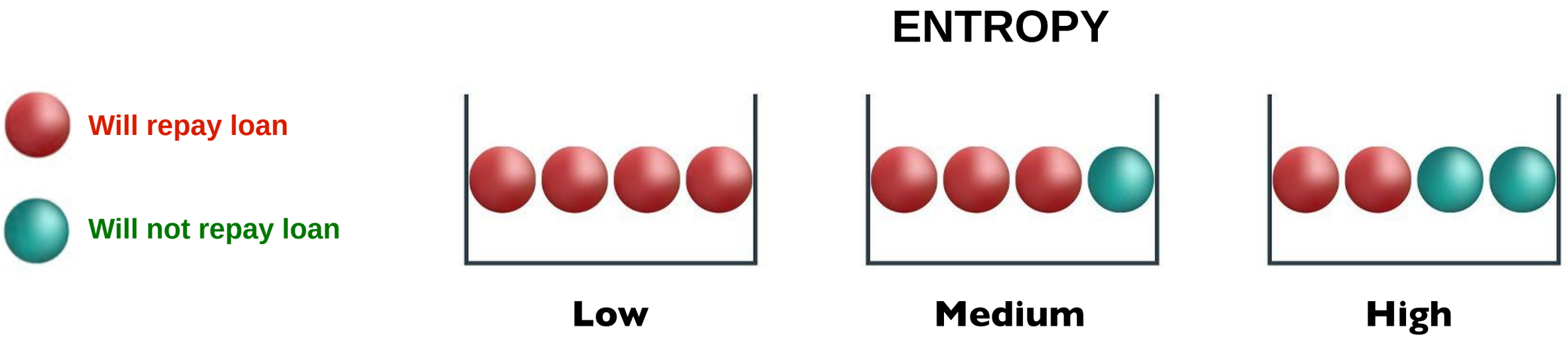
Do we have a good idea about whether the person is going to repay the loan or not?

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 homogenous as possible

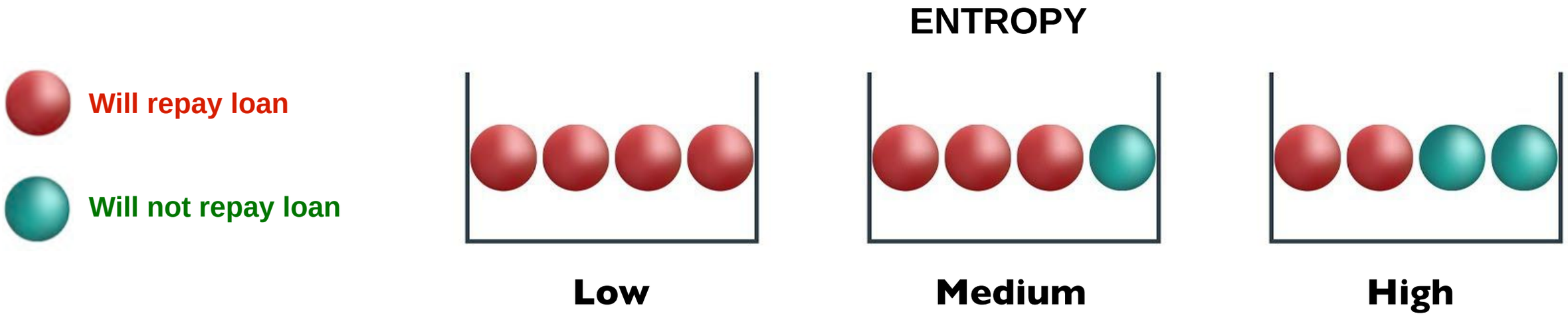
- How to quantify how homogenous a set of instances is?
  - ***Information, or entropy***
    - Information is measured in bits (or fractions of a bit)

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

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

{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 #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

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

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

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

I(2/4, 2/4) = -2/4 **log$_2$**(2/4) -2/4 **log$_2$**(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

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

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

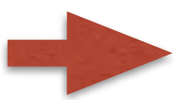
I(2/4, 2/4) = -2/4 **log$_2$**(2/4) -2/4 **log$_2$**(2/4)

= 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



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

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

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

I(2/2, 0/2) **=** -2/2 **log$_2$(**2/2**)** -0/2 **log$_2$(**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

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

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 **log$_2$**(2/2) -0/2 **log$_2$**(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

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

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

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

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

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 cla
  - 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**}

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

I(5/7, 2/7**) =** -5/7 **log₂(**5/7**)** -2/7 **log₂(**2/7**)**
= 0.8631

Selecting an Attribute to Test

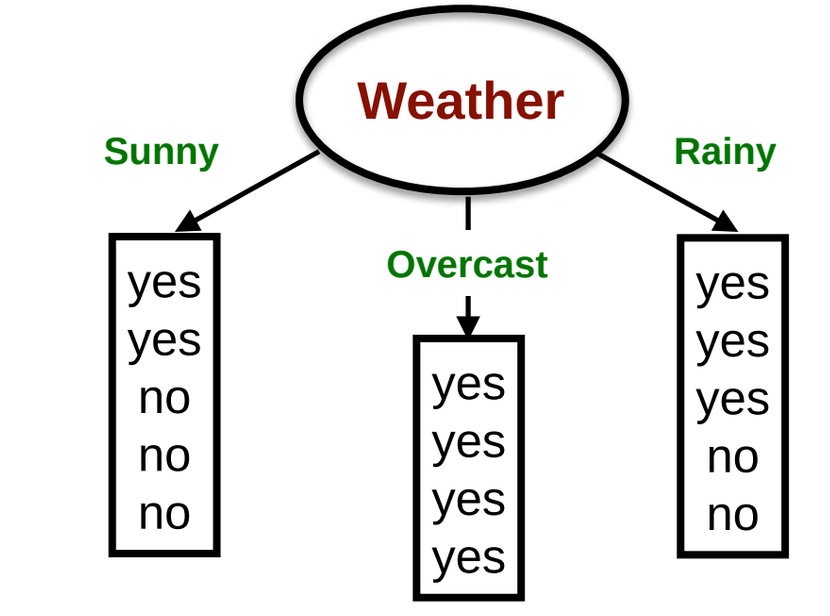
- Decision tree to predict whether a person will play tennis

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

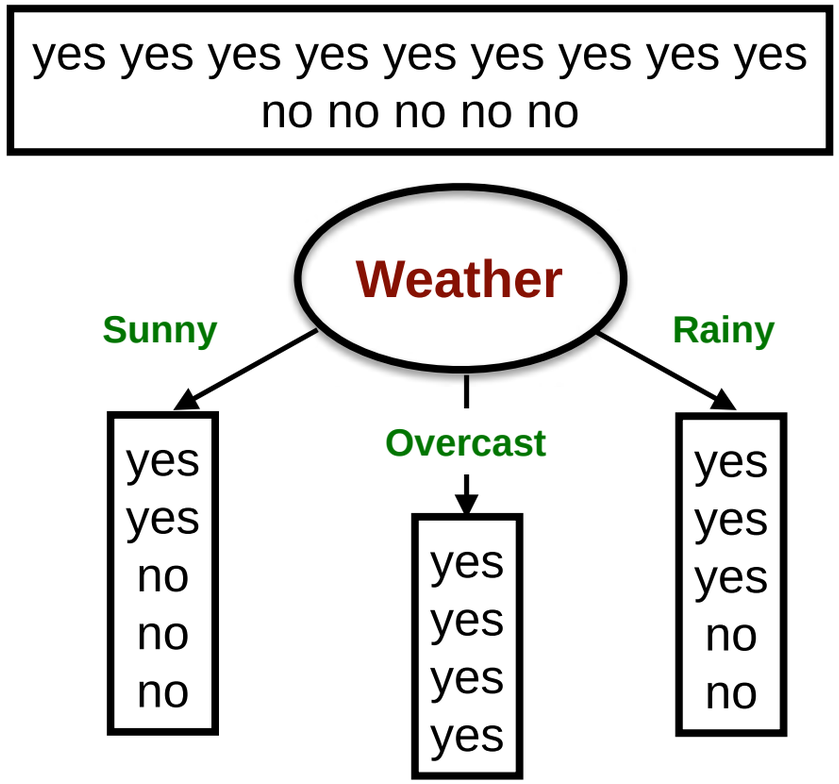
- **Entropy of the original dataset:**
  - **I**(9/14, 5/14) = -9/14 $\log_2$(9/14) -5/14 $\log_2$(5/14)
    = **0.940 bits**

- **Entropy of partitions resulting from testing Weather:**
  - **Weather=Sunny**
    - **I**(2/5, 3/5) = -2/5 $\log_2$(2/5) -3/5 $\log_2$(3/5)
      = 0.971 bits
  - **Weather=Overcast**
    - **I**(4/4, 0/4) = -4/4 $\log_2$(4/4) -0/4 $\log_2$(0/4)
      = 0 bits
  - **Weather=Rainy**
    - **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)x0.971 + (4/14)x0 + (5/14)x0.971 = **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:**
  - **I**(9/14, 5/14) = -9/14 $\log_2$(9/14) -5/14 $\log_2$(5/14)
    = **0.940 bits**

- **Entropy of partitions resulting from testing Weather:**
  - **Weather=Sunny**
    - **I**(2/5, 3/5) = -2/5 $\log_2$(2/5) -3/5 $\log_2$(3/5)
      = 0.971 bits
  - **Weather=Overcast**
    - **I**(4/4, 0/4) = -4/4 $\log_2$(4/4) -0/4 $\log_2$(0/4)
      = 0 bits
  - **Weather=Rainy**
    - **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)x0.971 + (4/14)x0 + (5/14)x0.971 = **0.693 bits**

By testing the attribute **Weather**, the entropy of the classes decreased by 0.940 - 0.693 = **0.247 bits**

Information Gain

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

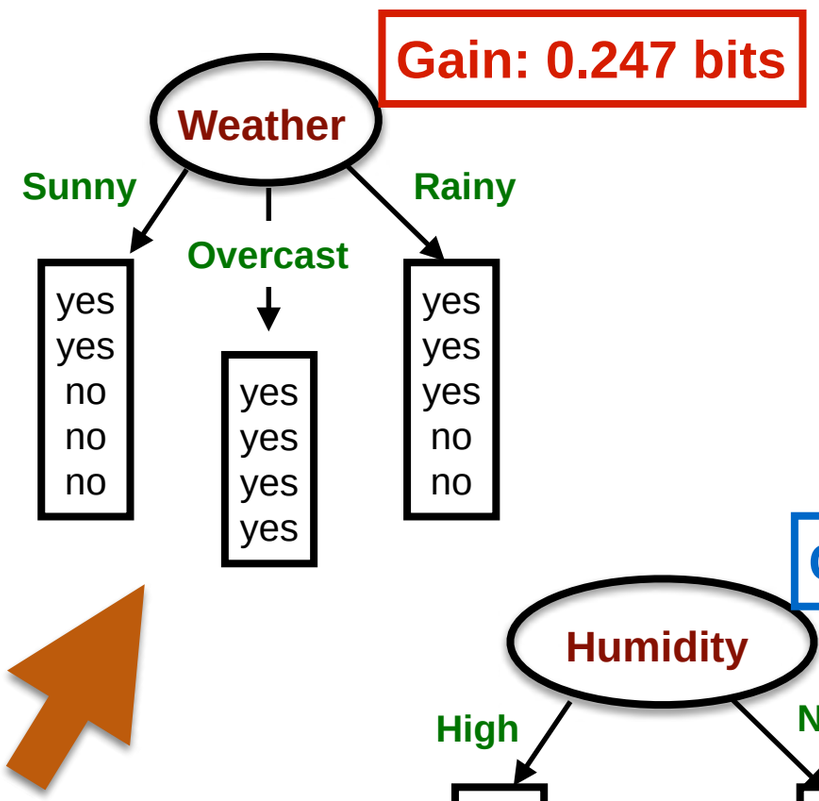
The algorithm will test, first, the attributes that result in higher information gain

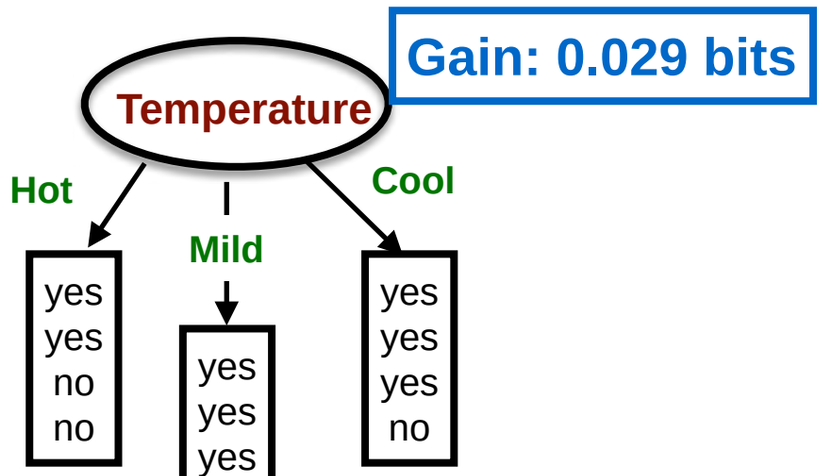
Selecting an Attribute to Test

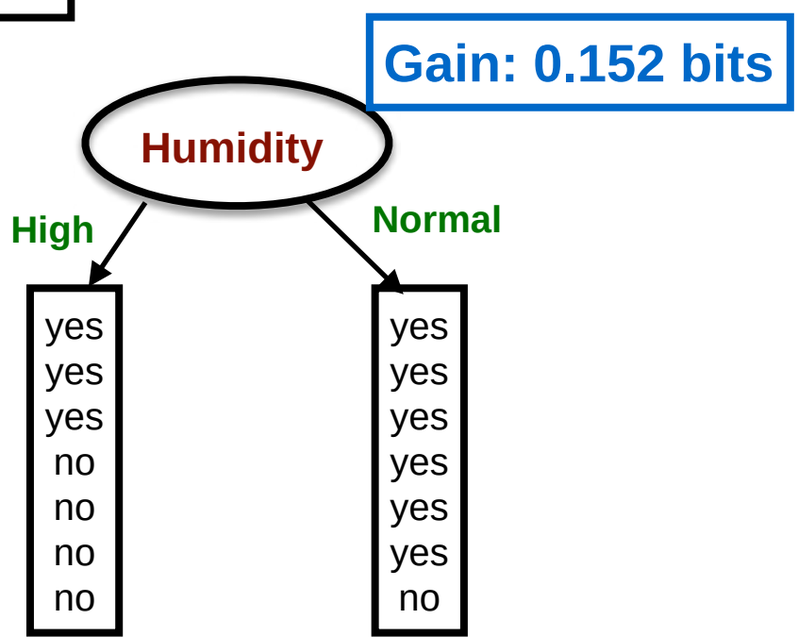
Information Gain

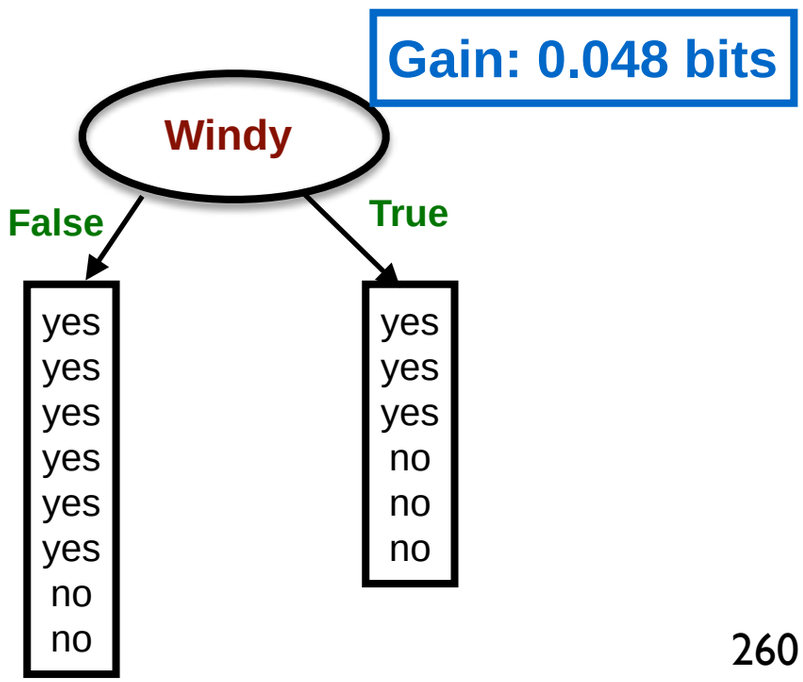
- quantifies how much information about the class is obtained by testing a given attribute

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

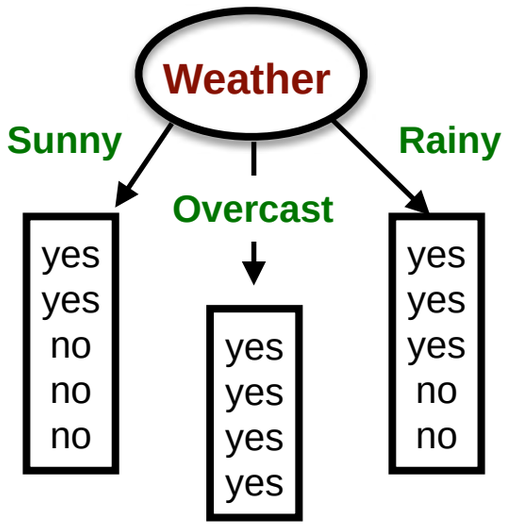






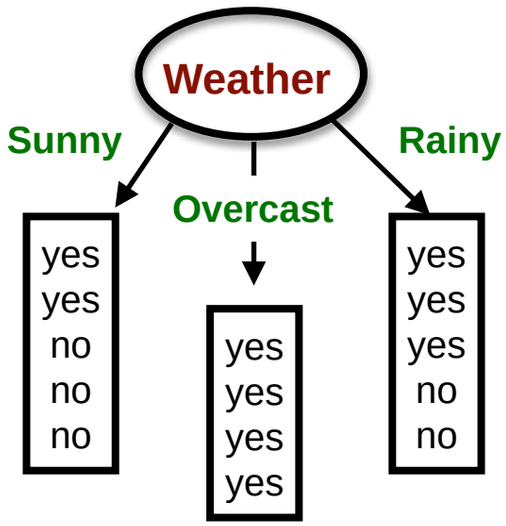


Selecting an Attribute to Test

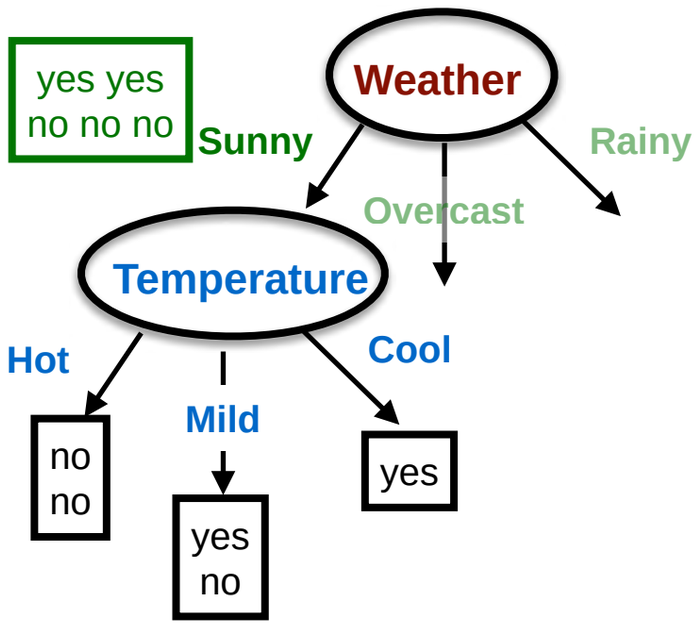


- We have decided that the 1st attribute to test is Weather
- What should be tested next, on the Sunny branch?
  - i.e., should we test Temperature, Windy, or Humidity?

Selecting an Attribute to Test

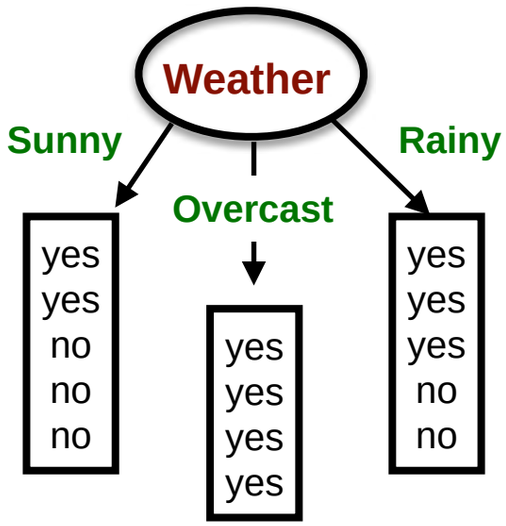


- We have decided that the 1st attribute to test is Weather
- What should be tested next, on the Sunny branch?
  - i.e., should we test Temperature, Windy, or Humidity?

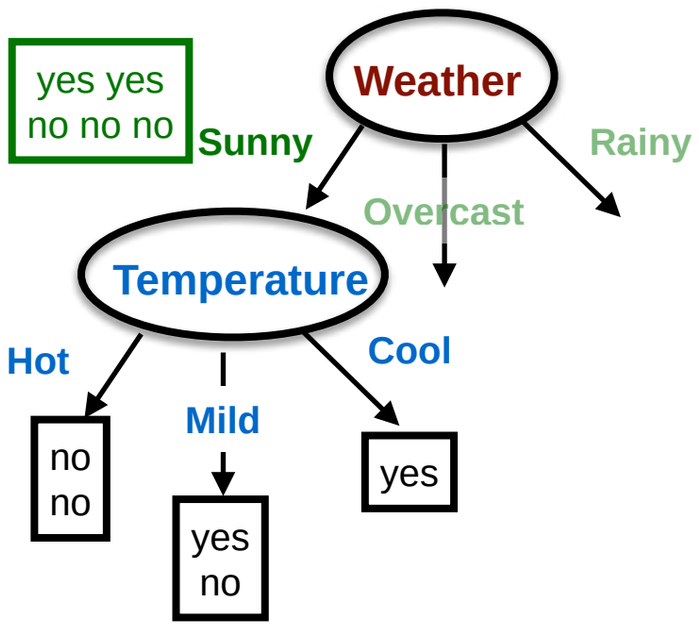


Gain: 0.571 bits

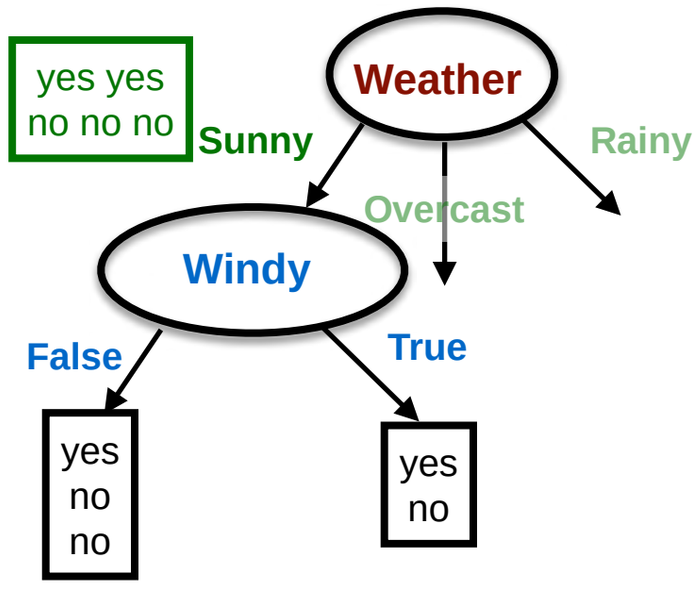
Selecting an Attribute to Test



- 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?

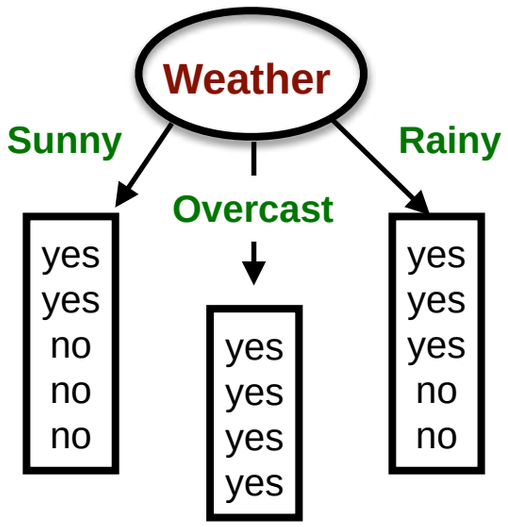


Gain: 0.571 bits

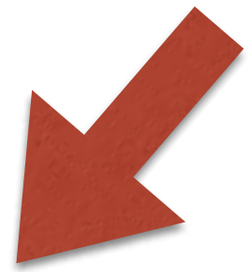


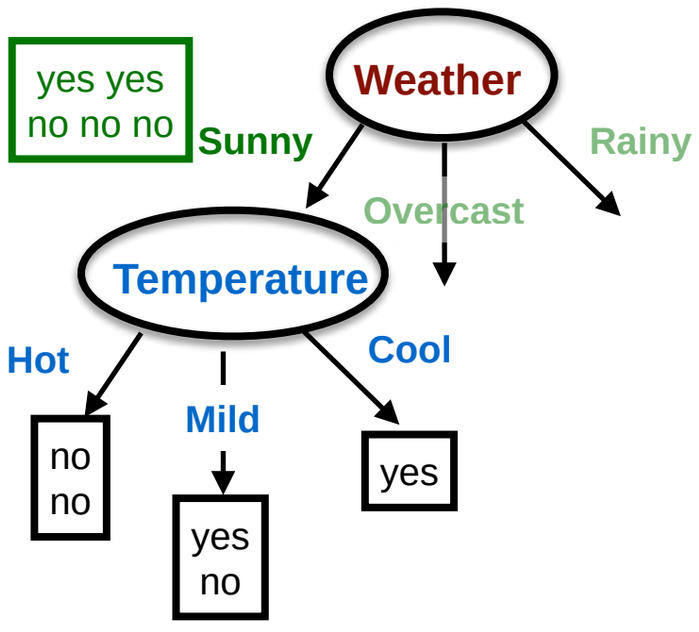
Gain: 0.020 bits

Selecting an Attribute to Test

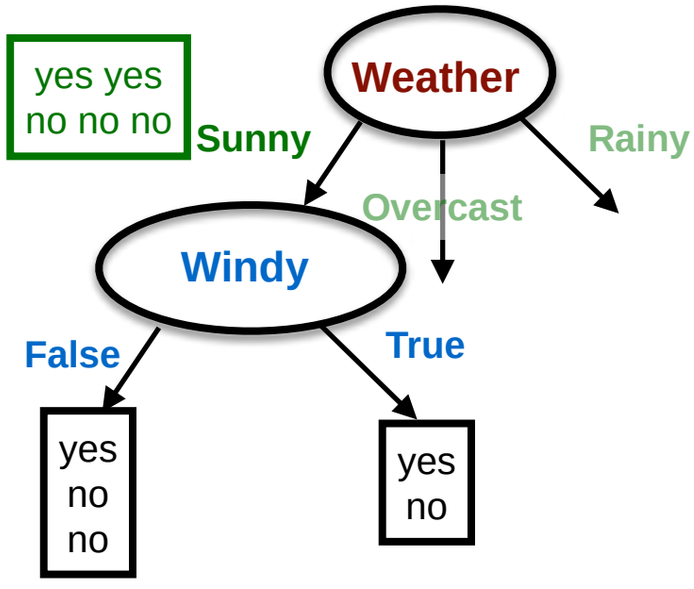


- We have decided that the 1st attribute to test is Weather
- What should be tested next, on the Sunny branch?
  - i.e., should we test Temperature, Windy, or Humidity?

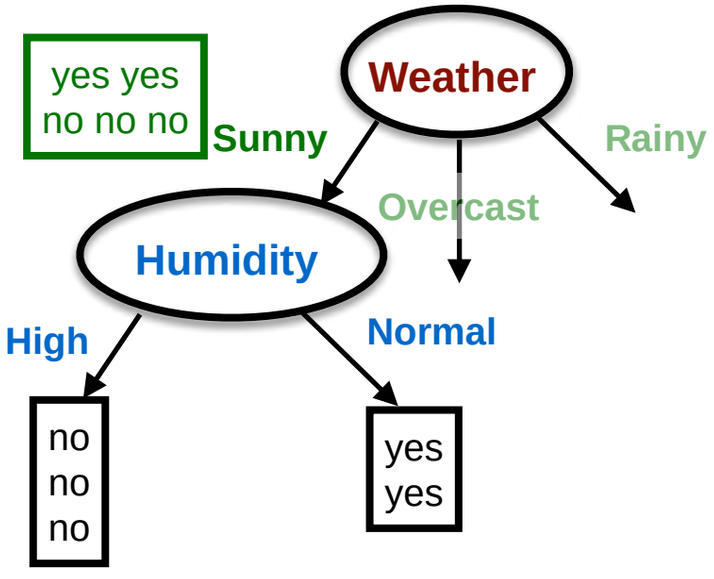




Gain: 0.571 bits

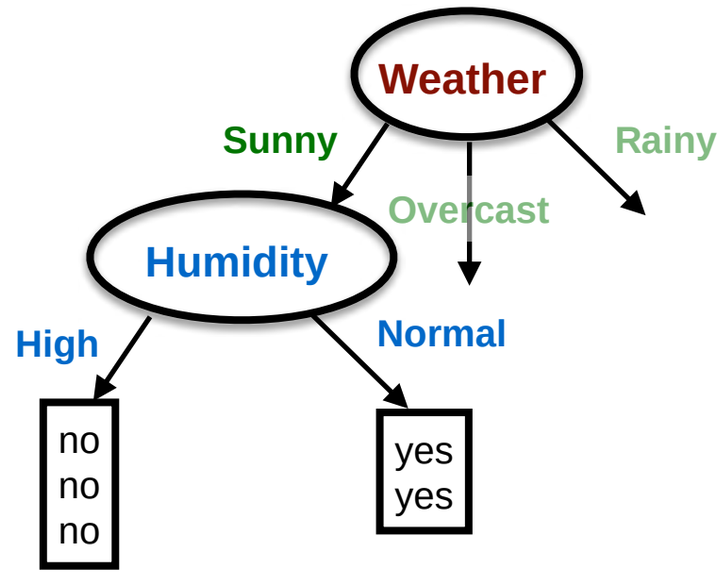


Gain: 0.020 bits

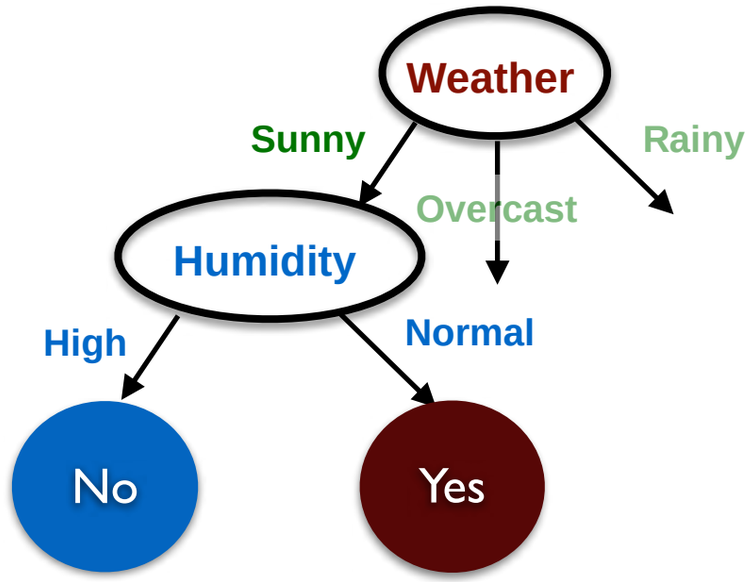


Gain: 0.971 bits

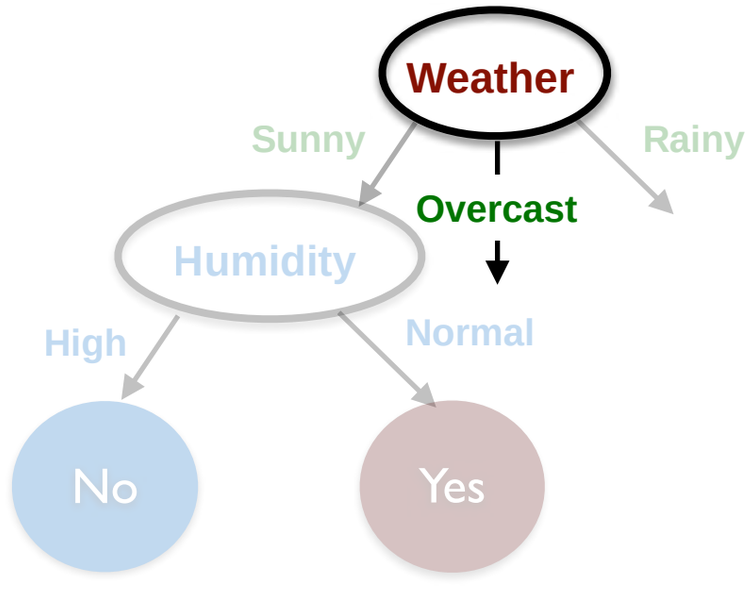
Selecting an Attribute to Test



Selecting an Attribute to Test



Selecting an Attribute to Test

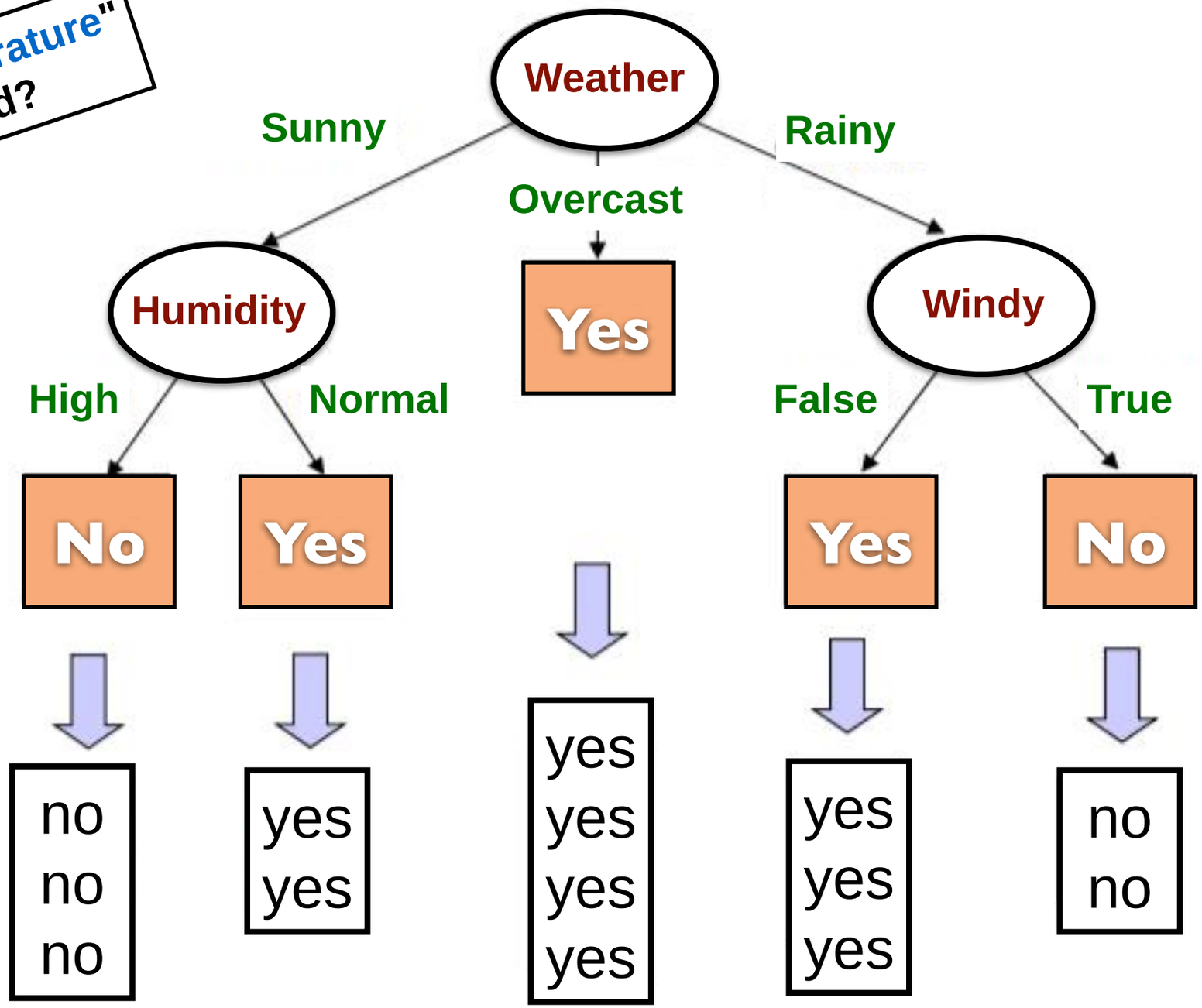


- What should be tested next, on the Overcast branch?
  - i.e., should we test Temperature, Windy, or Humidity?

Repeat the same process, recursively…

Learned Decision Tree

Where is "Temperature" being tested?



Decision Trees - Application

Decision trees for automated identification of cosmic-ray hits in Hubble Space Telescope images (1995)

- Automatically label objects in astronomical images (Hubble telescope)
  - stars / galaxies / cosmic rays?
  - noise or far-away star?
  - automatically counting objects in extremely high-resolution pictures

Machine Learning
CMPSCI 589

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

Decision Trees (1/2)

UMassAmherst
College of Information & Computer Sciences