Ricardo Aler Mur

- The main goal of this lecture is threefold:
    - To describe the basic machine learning pipeline (the sequence of processes that are typically followed in Machine Learning).
    - To explain some basic concepts: instances, attributes, and instance space.
    - To describe two of the basic Machine Learning algorithms: K-nearest neighbours and decision trees.

- About the KNN algorithm:
  - KNN is explained both for classication and regression.
  - Some of the advanges of KNN are explained (the main one being that it is not necessary to build an explicit model), and the inconveniences (slowness in classificating new instances, the problem with irrelevant attributes, the curse of dimensionality, and the problems with noise)
  - It is shown what the K means, its influence on classification and how to select it.

- About the decision tree algorithm:
  - The algorithm that builds decision trees is explained via an example
  - It is shown that the algorithm is recursive and basically amounts to choosing the best attribute at each node, by minimizing a measure called entropy.
  - It is shown how to extend the basic algorithm to regression tasks, via model trees and regression trees. In this case, the measure to minimize is variance, instead of entropy.

BASIC MODELS: NEAREST NEIGHBOURS AND TREE-BASED MODELS

Machine Learning Pipeline for classification (and regression)

Attributes, features, Input variables, Independent variables

Label, class, output variable, dependent variable

Test data

Sky	Temperature	Humidity	Wind	Tennis
Sunn y	85	85	No	No
Sunn y	80	90	Yes	No
Outc ast	83	86	No	Yes
Rainy	70	96	No	So
Rainy	68	80	No	Yes
Outc ast	64	65	Yes	Yes
Sunn y	72	95	No	No
Sunn y	69	70	No	Yes
Rainy	75	80	No	Yes
Sunn y	75	70	Yes	Yes
Outc ast	72	90	Yes	Yes
Outc ast	81	75	No	Yes
Rainy	71	91	Yes	No

Instances, examples

Training Data

Sky	Tempe rature	Humidity	Wind	Tennis
Sunny	60	65	No	?????

ML Algorithm

IF Sky = Sunny **AND** Humidity <= 75

THEN Tennis = Yes …

Model (Classifier)

Class = Yes

Prediction

DEFINITIONS: ATTRIBUTES AND LABEL

- **Attributes / features**
  - A feature is an individual measurable property of an instance, represented as a number
  - Types:
    - In general, real numbers:
      - 0, 1, 2
      - 1.3, 7.9, 10.798, …
    - Categorical: red / green / yellow
    - Ordinal: cold, lukewarm, hot
   - Typically, they are encoded as integer numbers

- **Class / label**
  - If categorical (e.g. cancer / no cancer) => classification problem
  - If real / integer number => regression problem

DEFINITIONS: INSTANCE SPACE

- Instances, examples, = tuples
  - In general, they inhabit a d-dimensional space (instance space)
  - (input, output) = $(\mathbf{x}_i, y) = (x_{i1}, x_{i2}, \ldots, x_{id}, y) \in (\mathbf{R^d}, Y)$
    - *Note: **boldface** means vector*
    - This instance has 4 inputs and 1 output. It inhabits a 4-dimensional space

Sky x_1	Temperature x_2	Humidity x_3	Wind x_4	Tennis y
Sunny	60	65	No	Yes

  - In 2-dimensions (2 attributes), each instance is a point in instance space
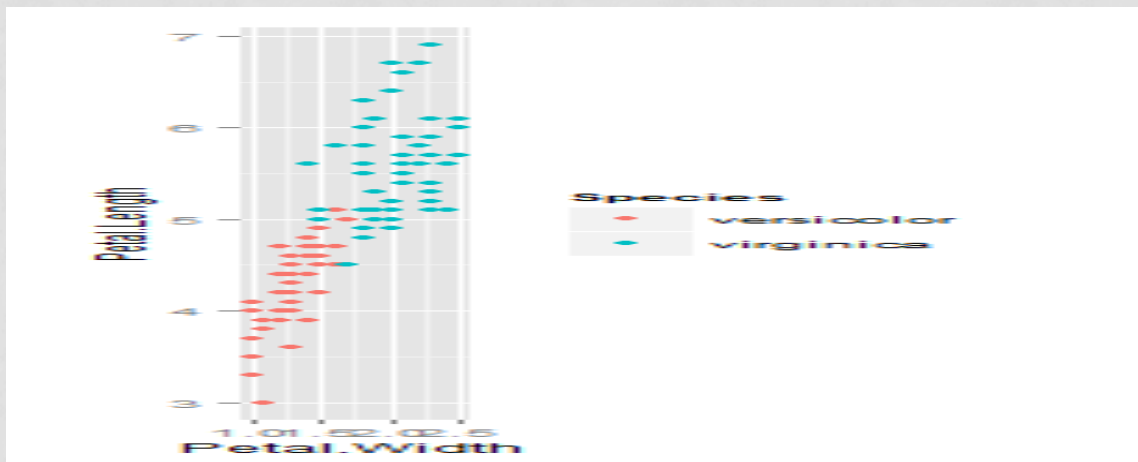
- Example: classify plants into two classes ("versicolor" / red vs. "virginica" / blue)
- 2 attributtes = (Petal.Width, Petal.Length) = 2 dimensions
- Classification = finding a boundary between the classes

Instance space in 2 dimensions
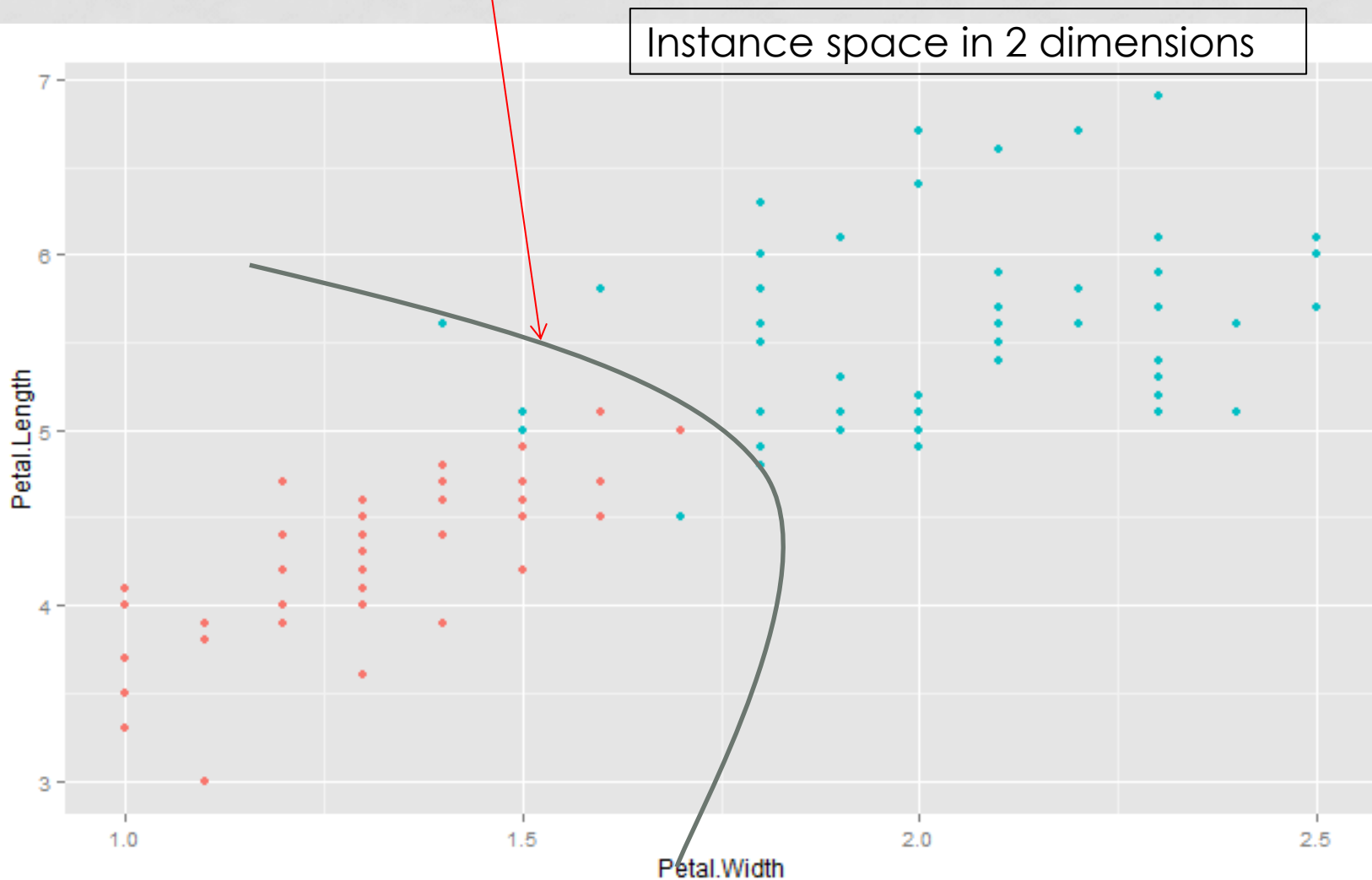
INSTANCE SPACE IN REGRESSION

- In the following case, there is one input variable and an output variable (continuous "label")
- Regression = finding a function that transforms the inputs into the output
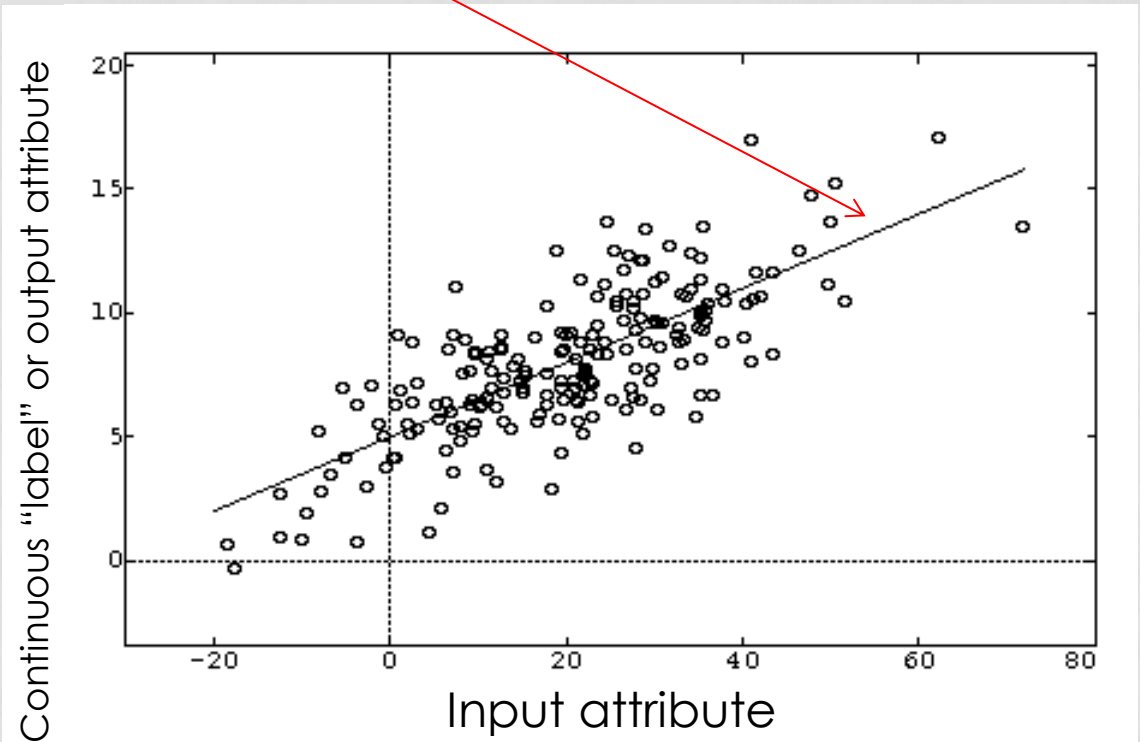
MODELS (CLASSIFICATION): RULES

Attributes, features,
Input variables,
Independent variables

Label, class, output variable,
dependent variable

Test data

Sky	Temperature	Humidity	Wind	Tennis
Sunny	85	85	No	No
Sunny	80	90	Yes	No
Outcast	83	86	No	Yes
Rainy	70	96	No	So
Rainy	68	80	No	Yes
Outcast	64	65	Yes	Yes
Sunny	72	95	No	No
Sunny	69	70	No	Yes
Rainy	75	80	No	Yes
Sunny	75	70	Yes	Yes
Outcast	72	90	Yes	Yes
Outcast	81	75	No	Yes
Rainy	71	91	Yes	No

Instances, examples

Training Data

Sky	Temperature	Humidity	Wind	Tennis
Sunny	60	65	No	?????

ML Algorithm

RULES

IF Sky = Sunny **AND** Humidity <= 75

THEN Tennis = Yes …

Model (Classifier)

Class = Yes

Prediction

MODELS: DECISION TREES

Attributes, features, Input variables, Independent variables

Label, class, output variable, dependent variable

Test data

Instances, examples

Sky	Temperature	Humidity	Wind	Tennis
Sunny	85	85	No	No
Sunny	80	90	Yes	No
Outcast	83	86	No	Yes
Rainy	70	96	No	So
Rainy	68	80	No	Yes
Outcast	64	65	Yes	Yes
Sunny	72	95	No	No
Sunny	69	70	No	Yes
Rainy	75	80	No	Yes
Sunny	75	70	Yes	Yes
Outcast	72	90	Yes	Yes
Outcast	81	75	No	Yes
Rainy	71	91	Yes	No

Training Data

Sky	Temperature	Humidity	Wind	Tennis
Sunny	60	65	No	?????

ML Algorithm

DECISION TREE
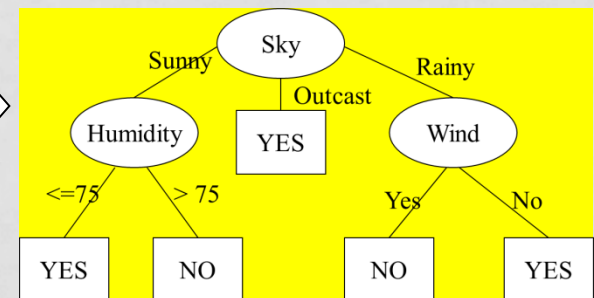
Model (Classifier)

Class = Yes

Prediction

MODELS: MANY OTHERS

Attributes, features,
Input variables,
Independent variables

Label, class, output variable,
dependent variable

Test data

Sky	Temperature	Humidity	Wind	Tennis
Sunny	85	85	No	No
Sunny	80	90	Yes	No
Outcast	83	86	No	Yes
Rainy	70	96	No	So
Rainy	68	80	No	Yes
Outcast	64	65	Yes	Yes
Sunny	72	95	No	No
Sunny	69	70	No	Yes
Rainy	75	80	No	Yes
Sunny	75	70	Yes	Yes
Outcast	72	90	Yes	Yes
Outcast	81	75	No	Yes
Rainy	71	91	Yes	No

Instances, examples

Training Data

Sky	Temperature	Humidity	Wind	Tennis
Sunny	60	65	No	?????

ML Algorithm

- Nearest neighbor
- Ensembles (bagging, boosting, stacking, …)
- Functions: neural networks, support vector machines, deep learning, …
- Naive bayes, bayesian networks

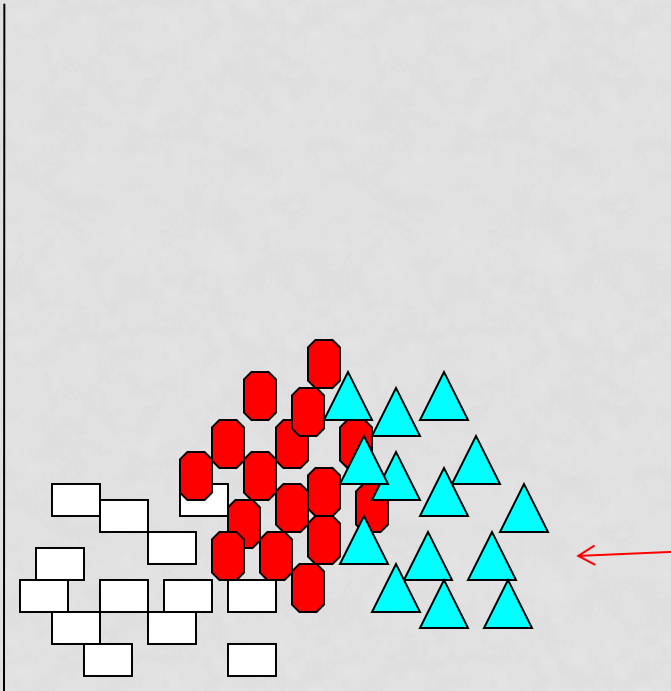Model (Classifier)

Class = Yes

Prediction

KNN: K-NEAREST NEIGHBOURS
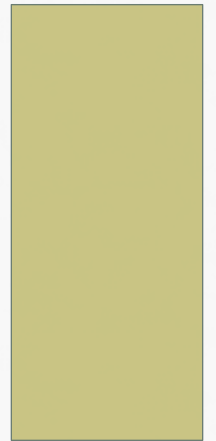
K-NEAREST NEIGHBORS (KNN)

Training

Height

Weight

☐ Child

🟥 Adult

🔺 Old

Model = all training instances are stored

K-NEAREST NEIGHBORS (KNN)

Testing

Height

K=1

Test instance
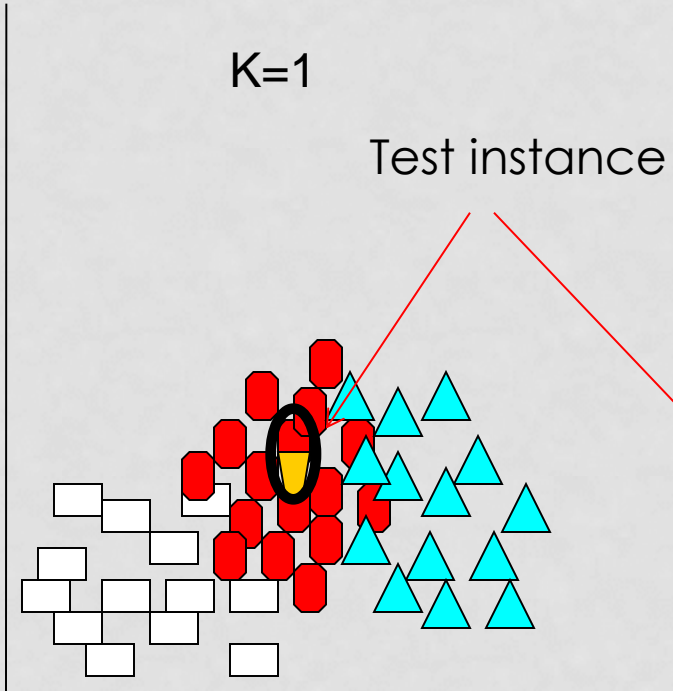
□ Child

■ Adult

▲ Aged

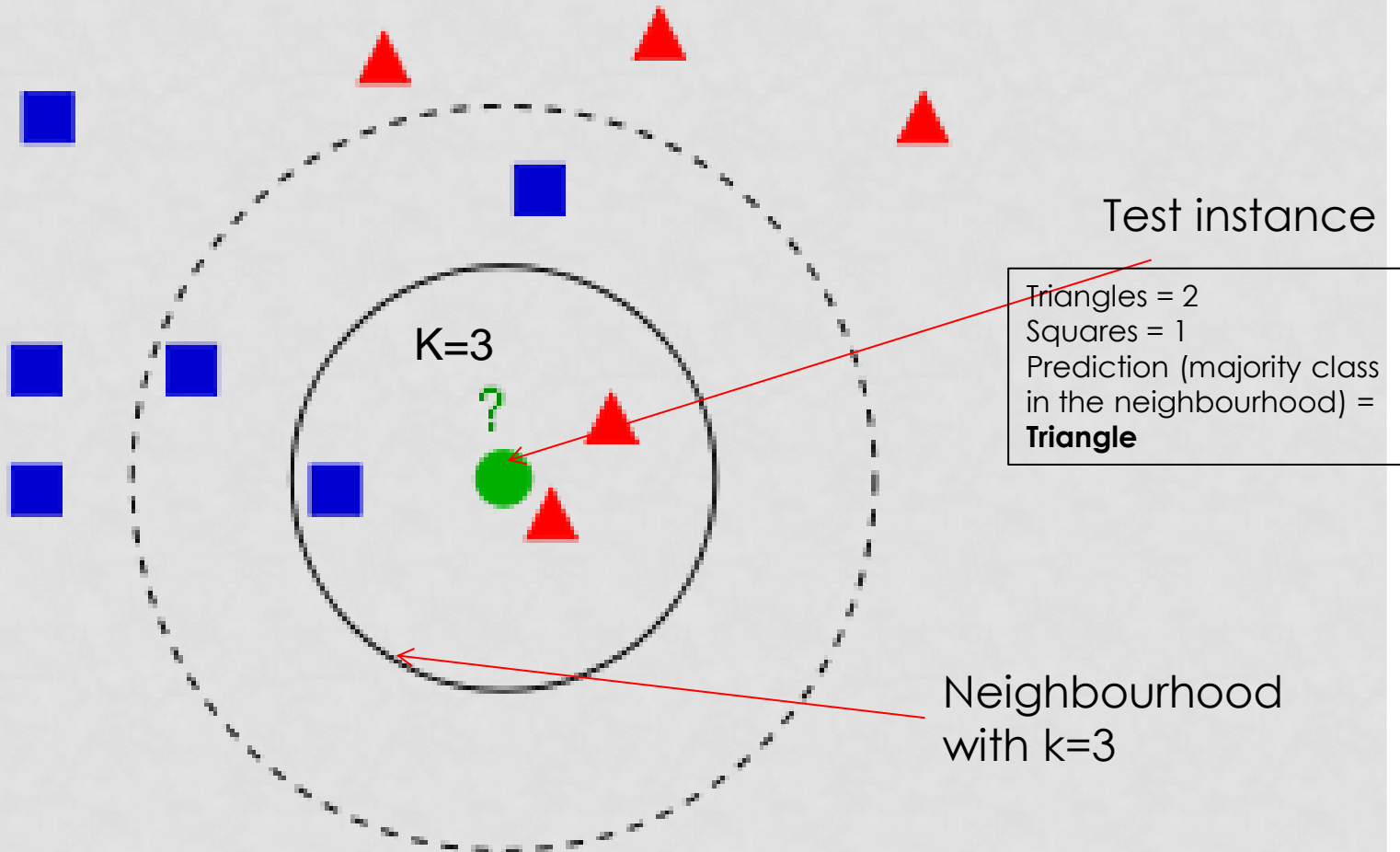Prediction = Adult (closest training instance)

Weight

K-NEAREST NEIGHBORS (KNN)

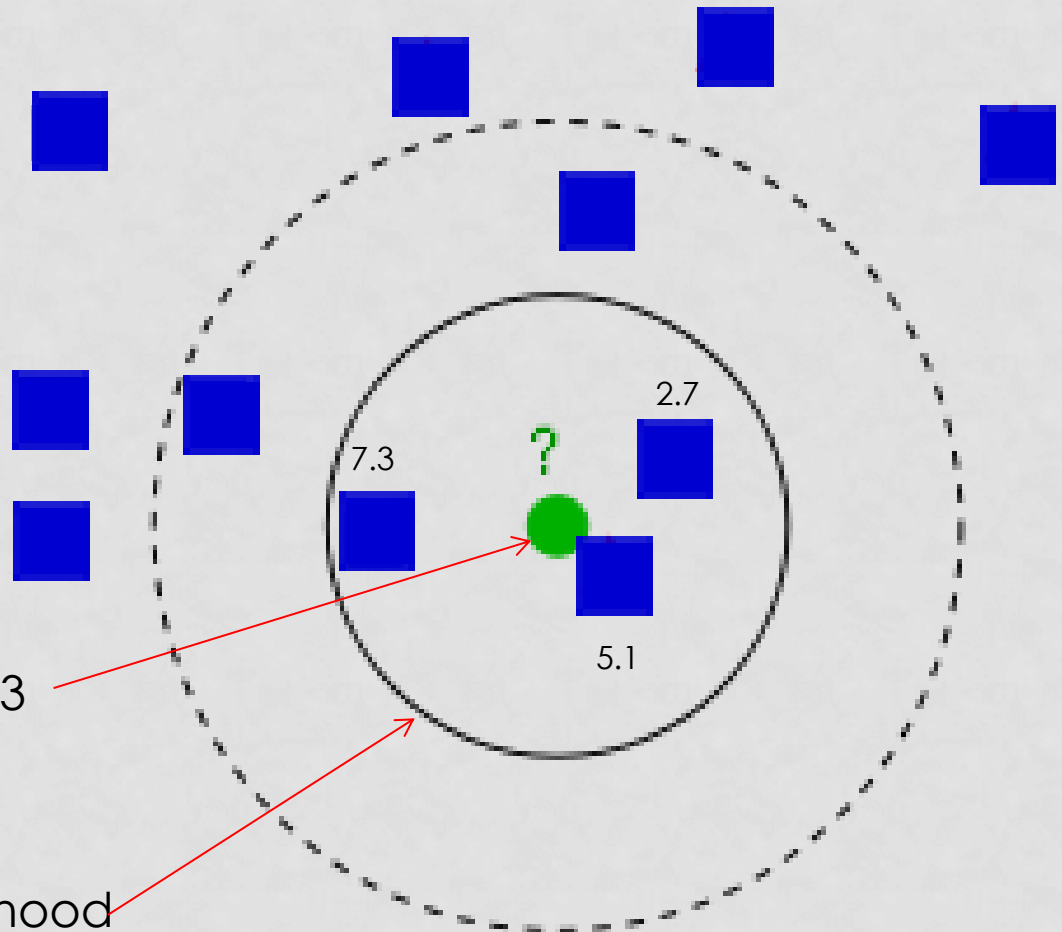K> 2 => classify new instances as the majority class of the k-nearest neighbours

K=3

Test instance

Triangles = 2
Squares = 1
Prediction (majority class in the neighbourhood) = **Triangle**

Neighbourhood with k=3

KNN FOR REGRESSION

It can be easily extended for **regression** by computing the average of the k-nearest neighbours

2.7

7.3

5.1

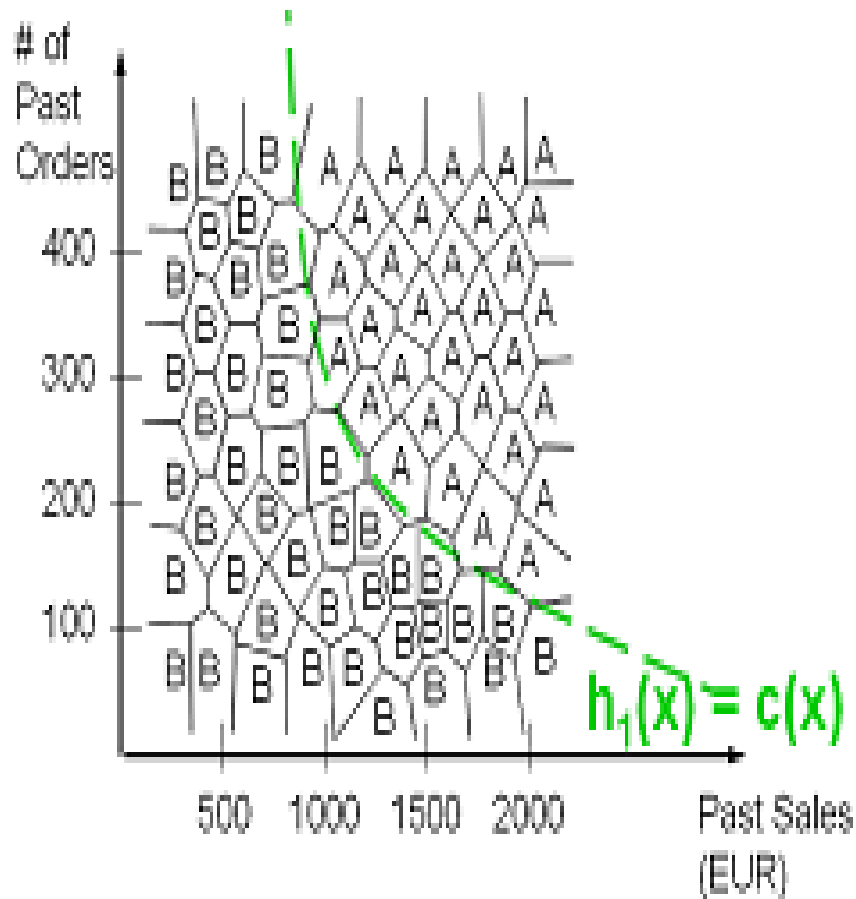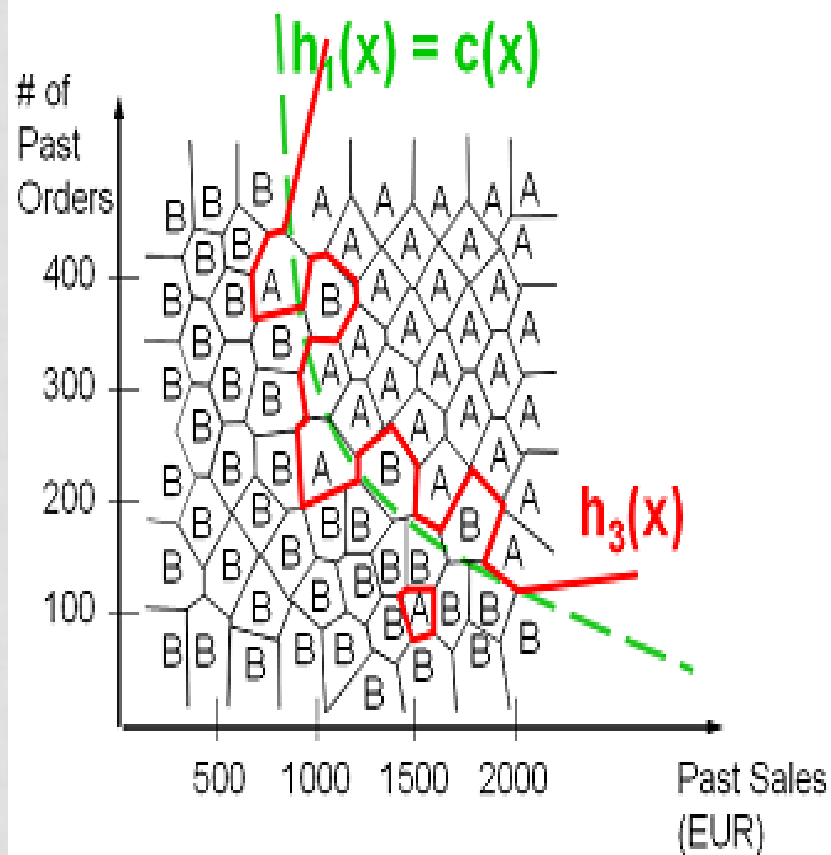Prediction = (7.3+2.7+5.1)/3

Neighbourhood with k=3

WHY USE K>1?

VORONOI TESSELLATION in Instance space

$h_1(x) = c(x)$

$h_2(x)$

$h_1(x) = c(x)$

K=1
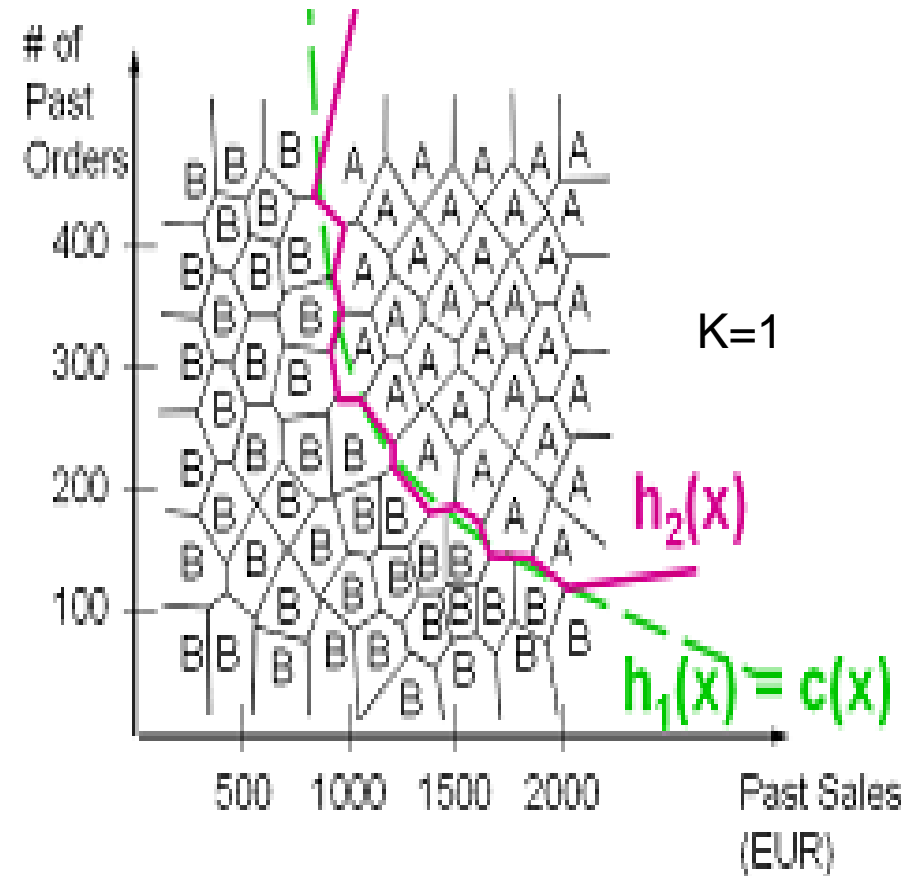
WHY USE K>1?

(a) 1-NN on noisy data
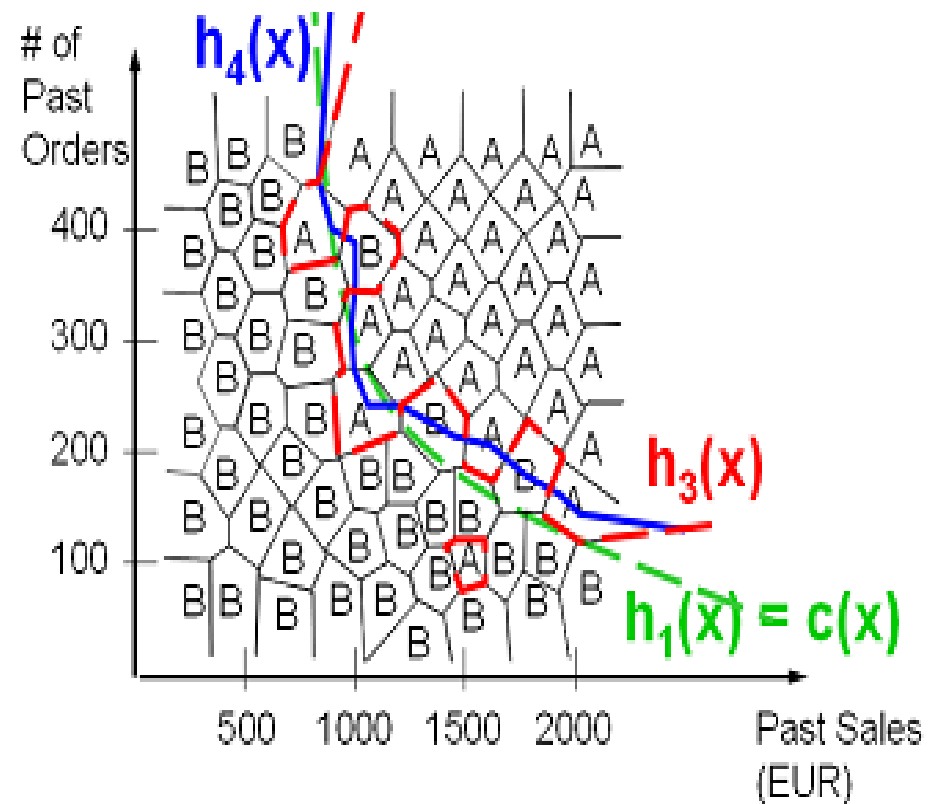
(b) 3-NN and noisy data

WHY USE K>1?

- With k=1, noisy instances (i.e. class overlap) have a large influence
- With k>1, more neighbors are considered and noisy instances have less influence (it is like averaging)
- But if k is very large, locality is lost
  - What is KNN if k=number of instances?
- If the number of classes is two, use odd k in order to avoid draws
- K is a parameter of the algorithm (a hyperparameter, actually) and the correct value has to be found out. Hyperparameter tuning is part of the ML pipeline, and we will talk about it in future lectures

SUMMARY OF KNN

- KNN classifies test instances as the majority class in the neighbourhood of the training set
- KNN is a lazy ML algorithm
  - During training, no model is constructed, but all training instances are stored (model = training instances)
- It is based on the idea that the best model of data is the data itself
- It can be easily extended for **regression** by computing the average of the k-nearest neighbours

LIMITATIONS OF KNN

- Slow (when testing): all distances to each training instance must be computed

- Large storage requirements (all training data is stored)

- Very sensitive to irrelevant attributes and the curse of dimensionality
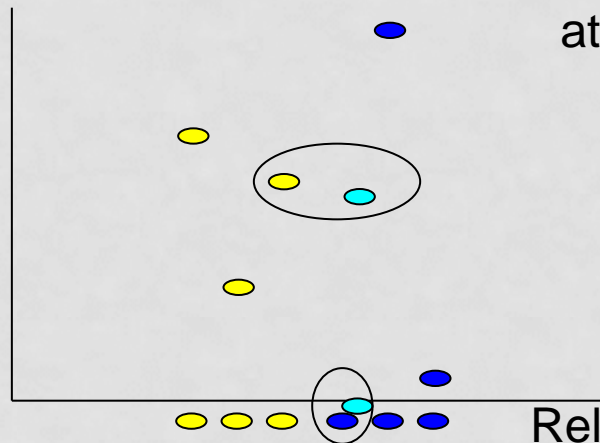
Irrelevant attributes

0 irrelevant atributtes
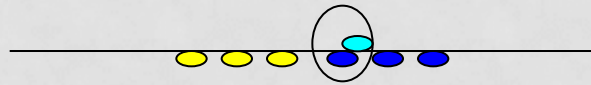
With the relevant attribute, classification is done properly

Irrelevant attribute

irrelevant attribute

1 irrelevant atribute

The irrelevant attribute makes distances confusing
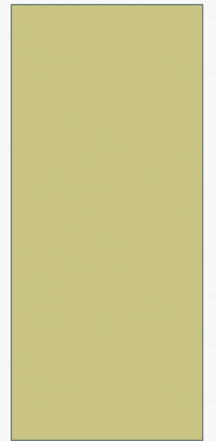
Relevant attribute

K=1

DISTANCES

- For numerical attributes, use the Euclidean distance:
  - 2D: $d(\mathbf{x_i},\mathbf{x_j})^2 = (x_{i1}-x_{j1})^2 + (x_{i2}-x_{j2})^2$
    - $d(\mathbf{x_i},\mathbf{x_j}) = \sqrt{((x_{i1}-x_{j1})^2 + (x_{i2}-x_{j2})^2)}$
  - dD: $d(\mathbf{x_i},\mathbf{x_j})^2 = (x_{i1}-x_{j1})^2 + (x_{i2}-x_{j2})^2 + \ldots + (x_{id}-x_{jd})^2$
- For nominal / categorical attributes, use the Hamming distance:
  - If attribute e es nominal (discrete, categorical, …), instead of $(x_{ie}-x_{je})^2$, the following is used: $\delta(x_{ie}, x_{je})$: 0 if $x_{ie}=x_{je}$ and 1 otherwise
- or transform the attribute to dummy variables / one-hot encoding)

NORMALIZATION

- It is important to normalize attributes, because ranges can be different (e.g. human body temperature ranges from 35° to 45° celsius, body height ranges from 0 to 2m, age ranges from 0 to 100 years, etc.)

- Otherwise, attributes with a large range have more weight on the distance

- Normalizing to 0-1 range:

  - $x'_{1j} = (x_{1j} - min_j)/(max_j - min_i)$

MODELS:
DECISION TREES (AND RULES) FOR CLASSIFICATION AND REGRESSION

MODELS: DECISION TREES

Attributes, features, Input variables, Independent variables

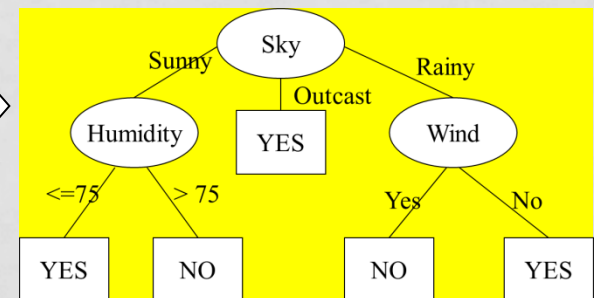Label, class, output variable, dependent variable

Test data

Sky	Temperature	Humidity	Wind	Tennis
Sunny	85	85	No	No
Sunny	80	90	Yes	No
Outcast	83	86	No	Yes
Rainy	70	96	No	Yes
Rainy	68	80	No	Yes
Outcast	64	65	Yes	Yes
Sunny	72	95	No	No
Sunny	69	70	No	Yes
Rainy	75	80	No	Yes
Sunny	75	70	Yes	Yes
Outcast	72	90	Yes	Yes
Outcast	81	75	No	Yes
Rainy	71	91	Yes	No

Instances, examples

Training Data

Sky	Temperature	Humidity	Wind	Tennis
Sunny	60	65	No	?????

ML Algorithm

DECISION TREE

Sky
Sunny / Outcast / Rainy
Humidity — YES — Wind
<=75 / > 75
YES / NO
Yes / No
NO / YES
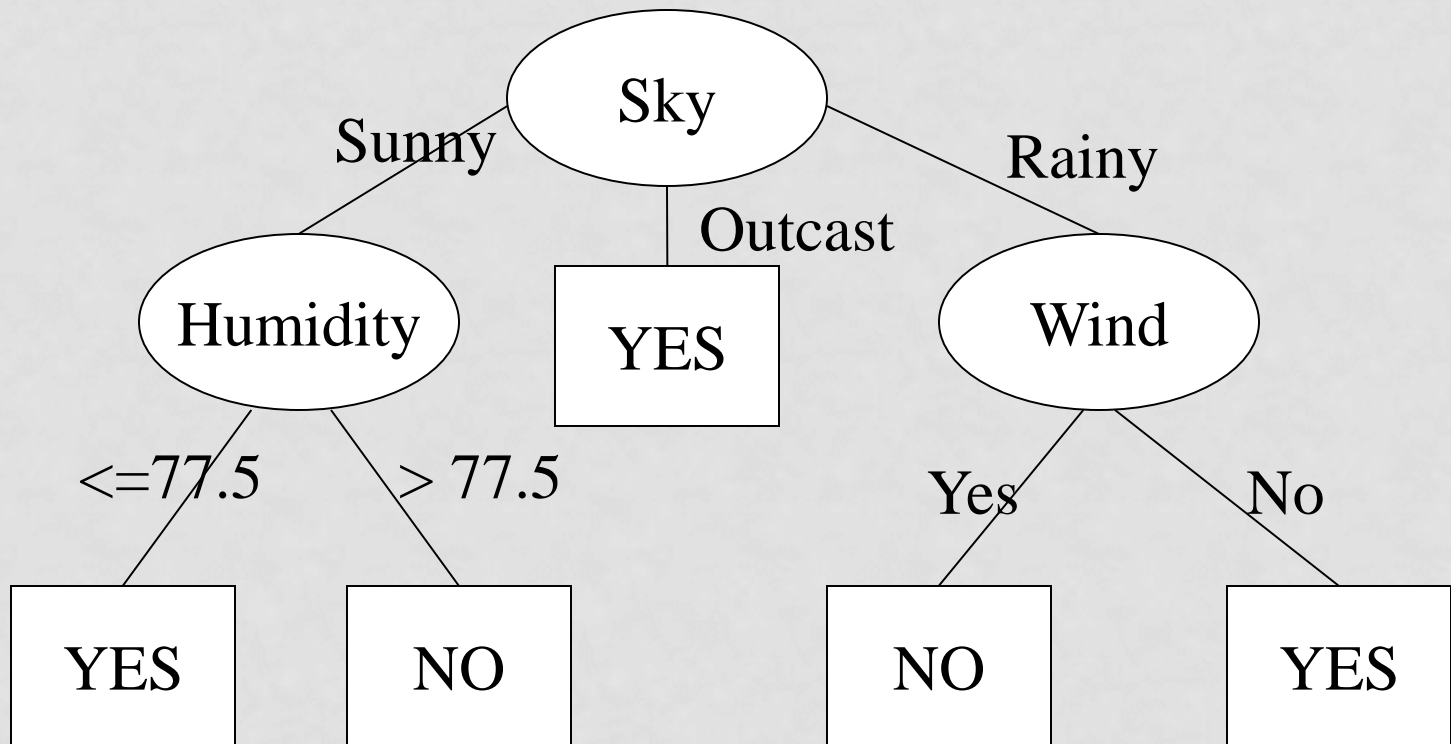
Model (Classifier)

Class = Yes

Prediction

Decision trees

Algorithms for building decision trees

- The most basic is ID3: decision trees are constructed recursively from the root to the leaves, each time selecting the best node (attribute) to put on the tree

- C4.5 (or J48), is able to deal with continuous attributes and uses statistical criteria to prevent overfitting the tree to the data (too large trees imply that data is memorized rather than generalized)

Simplified ID3 algorithm

1. Stop growing the tree if:
   1. All examples belong to the same class
   2. If there are no remaining instances or attributes
2. Otherwise, select the best attribute for that node, according to some criteria (entropy minimization, for instance)
3. Build recursively as many subtrees as values in the selected attribute

Simplified C4.5 algorithm

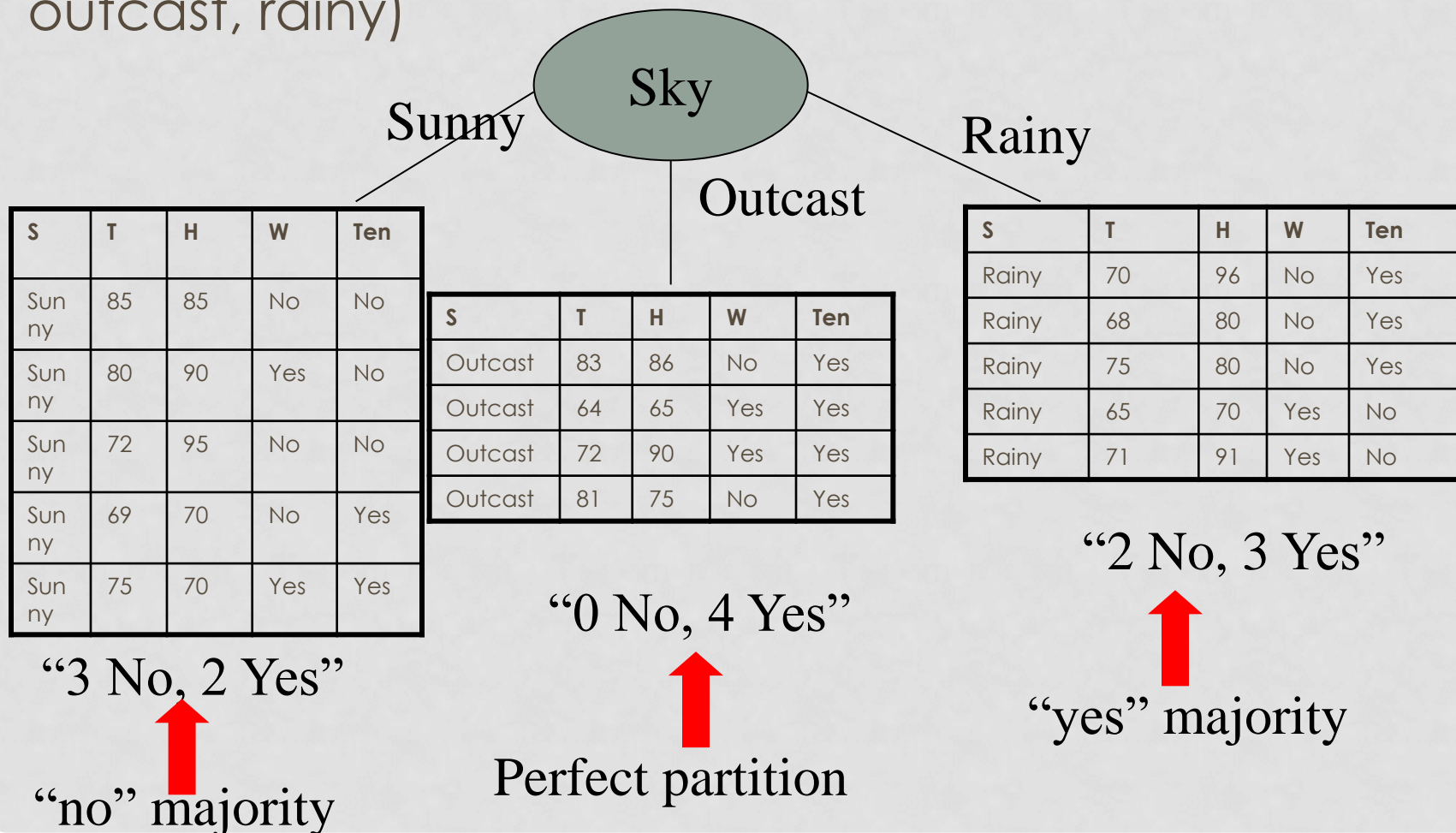1. Stop growing the tree if:
   1. All examples belong to the same class
   2. If there are no remaining instances or attributes
   3. If no improvements are expected by growing the tree
2. Otherwise, select the best attribute for that node, according to some criteria (entropy minimization, for instance)
3. Build recursively as many subtrees as values in the selected attribute

WHAT IS THE BEST NODE TO PUT IN THE ROOT OF THE TREE?

Sky?

Sunny   Outcast   Rainy

Humidity?

<=X   > X

Temperature?

<=X   > X

Wind?

Yes   No
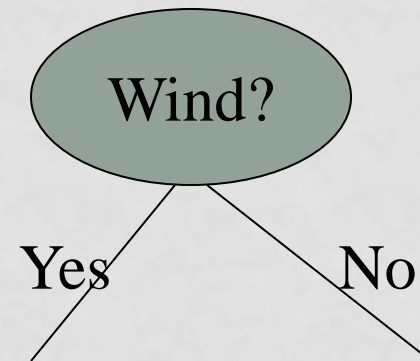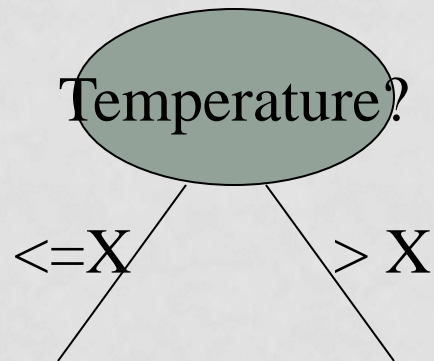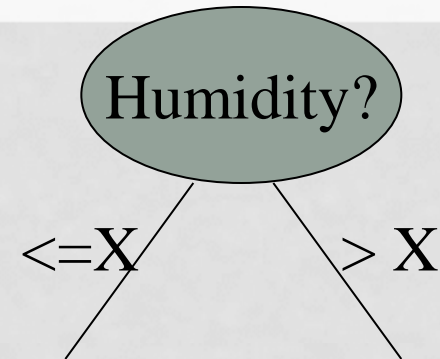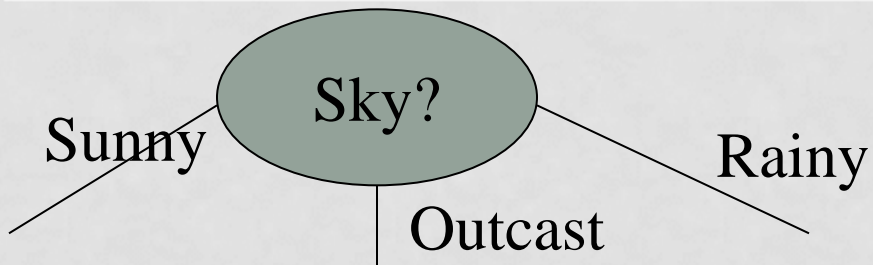
Let's try with attribute SKY

Sky generates as many partitions as values (3: sunny, outcast, rainy)

Sky

Sunny — Outcast — Rainy

S	T	H	W	Ten
Sunny	85	85	No	No
Sunny	80	90	Yes	No
Sunny	72	95	No	No
Sunny	69	70	No	Yes
Sunny	75	70	Yes	Yes

"3 No, 2 Yes"

"no" majority

S	T	H	W	Ten
Outcast	83	86	No	Yes
Outcast	64	65	Yes	Yes
Outcast	72	90	Yes	Yes
Outcast	81	75	No	Yes

"0 No, 4 Yes"

Perfect partition

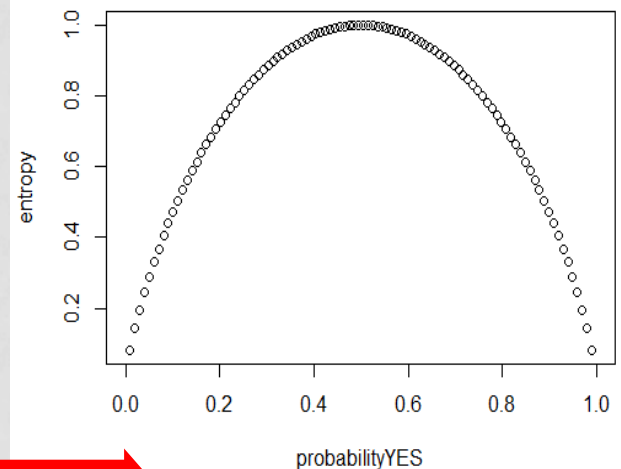| S	T	H	W	Ten
Rainy	70	96	No	Yes
Rainy	68	80	No	Yes
Rainy	75	80	No	Yes
Rainy	65	70	Yes	No
Rainy	71	91	Yes	No

"2 No, 3 Yes"

"yes" majority

How do we know if SKY is a good attribute?

- Perfect partition:
  - 0% No, 100% Yes
  - 100% No, 0% Yes
- Worse partition: 50% No, 50% Yes
- Entropy measures partition quality (the larger, the worse)

$$H(P) = -\sum_{Ci} p_{Ci} \log_2(p_{Ci})$$

$$H(P) = -(p_{yes} \log_2(p_{yes}) + p_{no} \log_2(p_{no}))$$
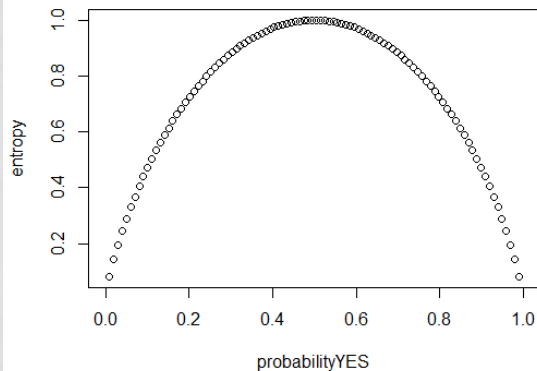$$p_{no} = (1 - p_{yes})$$

Proportion
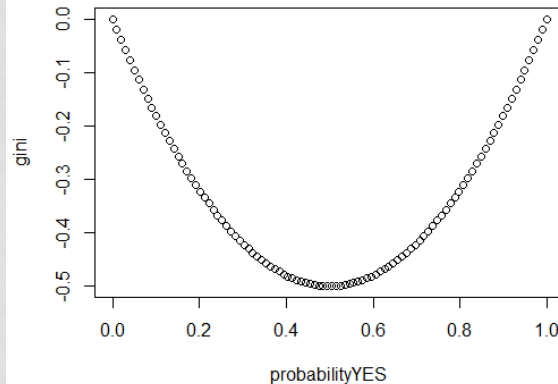of Yes

OTHER WAYS TO MEASURE PARTITION QUALITY

Entropy
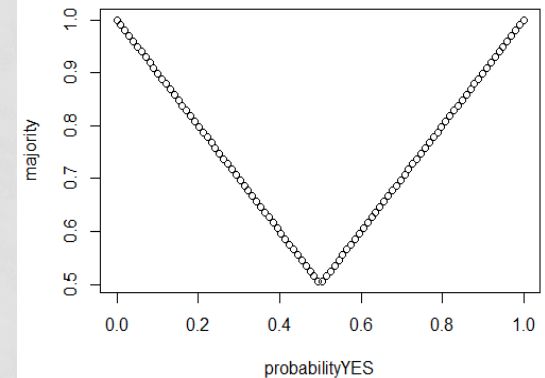
$$H(P) = -\sum_{Ci} p_{Ci} \log_2(p_{Ci})$$

Gini

$$Gini(P) = -\sum_{Ci} p_{Ci}(1 - p_{Ci})$$

Majority

$$M(P) = \max(p_{yes}, p_{no})$$

Average entropy for Sky

- Entropy for the three partitions of Sky:
  1. "3 No, 2 Yes": $H=-((3/5)*\log_2(3/5)+(2/5)*\log_2(2/5) = 0.97$
  2. "0 No, 4 Yes": $H=-((0/4)*\log_2(0/4)+1*\log_2(1))=0$
  3. "2 No, 3 Yes": $H=-((2/5)*\log_2(2/5)+(3/5)*\log_2(3/5))= 0.97$

- Average Sky entropy:
  - $HP=(5/14)*0.97+(4/14)*0+(5/14)*0.97 =$ **0.69**
  - Note: there are14 instances in the data set

WHAT TO DO FOR CONTINUOUS ATTRIBUTES?

C4.5 creates a binary (two-values) attribute by computing a threshold X

Nota: only some thresholds are shown. The best one is X=84 with average entropy = 0.83

Temperature   X?

$<=X$ $> X$
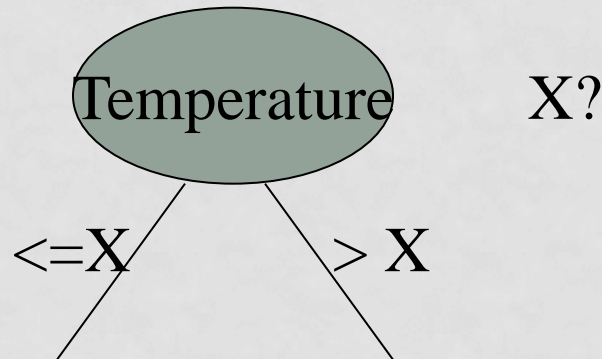
64 – Yes, 65-No, 68 – Yes, 69 – Yes, 70 – Yes, 71 – No, 72 – YesNo, 75 – YesYes, 80 – No, 81 – Yes, 83 – Yes, 85 - No

$HP = 0.83$

4 No, 9 Yes

X=84

1 No, 0 Yes

64 – Yes, 65-No, 68 – Yes, 69 – Yes, 70 – Yes, 71 – No, 72 – YesNo, 75 – YesYes, 80 – No, 81 – Yes, 83 – Yes, 85 - No

$HP = 0.93$

2 No, 4 Yes   X=71.5   3 No, 5 Yes

64 – Yes, 65-No, 68 – Yes, 69 – Yes, 70 – Yes, 71 – No, 72 – YesNo, 75 – YesYes, 80 – No, 81 – Yes, 83 – Yes, 85 - No

$HP = 0.89$

1 No, 4 Yes   X=70.5   4 No, 5 Yes

Possible thresholds

Possible thresholds are transitions from Yes to No, or from No to Yes:

64 – Yes, | 65-No, | 68 – Yes, 69 – Yes, 70 – Yes | 71 – No, | 72 – Yes | No, | 75 – Yes | Yes | 80 – No, | 81 – Yes, 83 – Yes | 85 - No

X=64.5 X=66.5 X=70.5 X=71.5 X=73.5 X=77.5 X=80.5 X=84
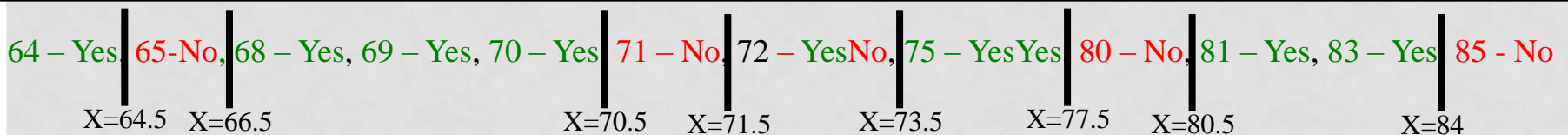
- The actual threshold may depend on the algorithm implementation. Some implementations use the average: Ej: 64.5 = (64+65)/2.
- Other implementations use the maximum of the left partition. In that case, the possible thresholds would have been:

64 – Yes, | 65-No, | 68 – Yes, 69 – Yes, 70 – Yes, | 71 – No, | 72 – Yes | No, | 75 – Yes | Yes, | 80 – No | 81 – Yes, 83 – Yes | 85 - No

X=64 X=65 X=70 X=71 X=72 X=75 X=80 X=83
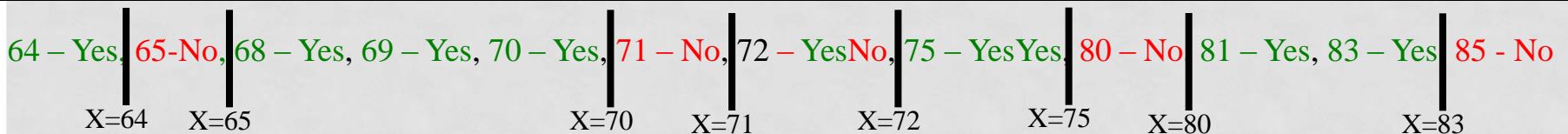
- Notice that entropy computed with the training data is the same in both cases, because in the two cases data is partitioned in the same way.

Humidity

Humidity

<=X   > X
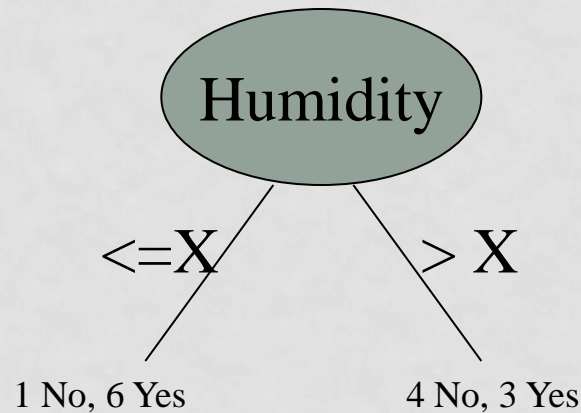
1 No, 6 Yes   4 No, 3 Yes

65-Yes, 70-NoYesYes, 75–Yes, 80-YesYes, | 85-No, 86-Yes, 90-NoYes, 91-No, 95-No, 96-Yes

X=82.5

1 No, 6 Yes   4 No, 3 Yes $HP = 0.79$

Note: there are other alternatives for the threshold, but this is the best one (minimum entropy)

WHAT IS THE BEST NODE TO PUT IN THE ROOT OF THE TREE?
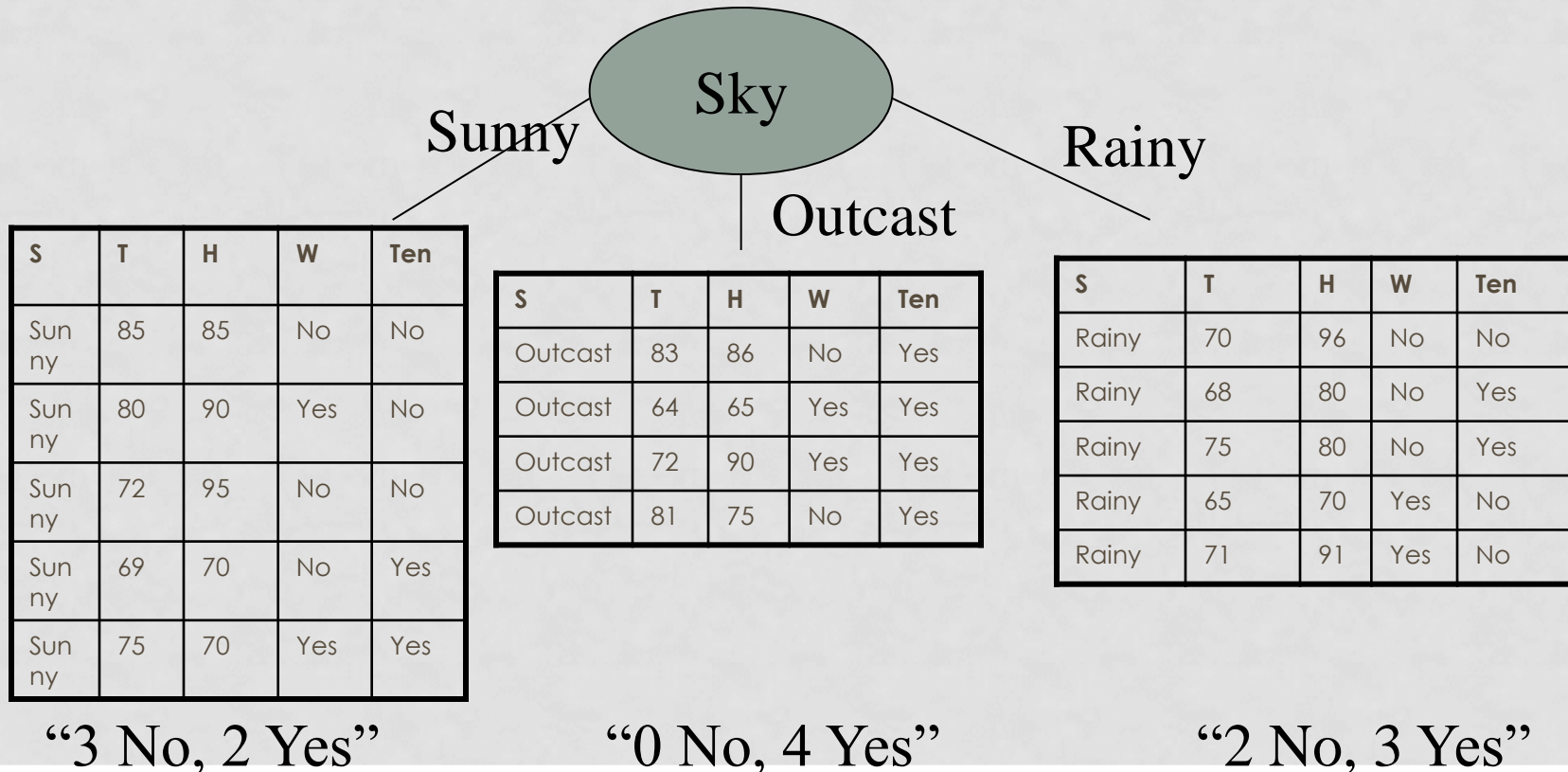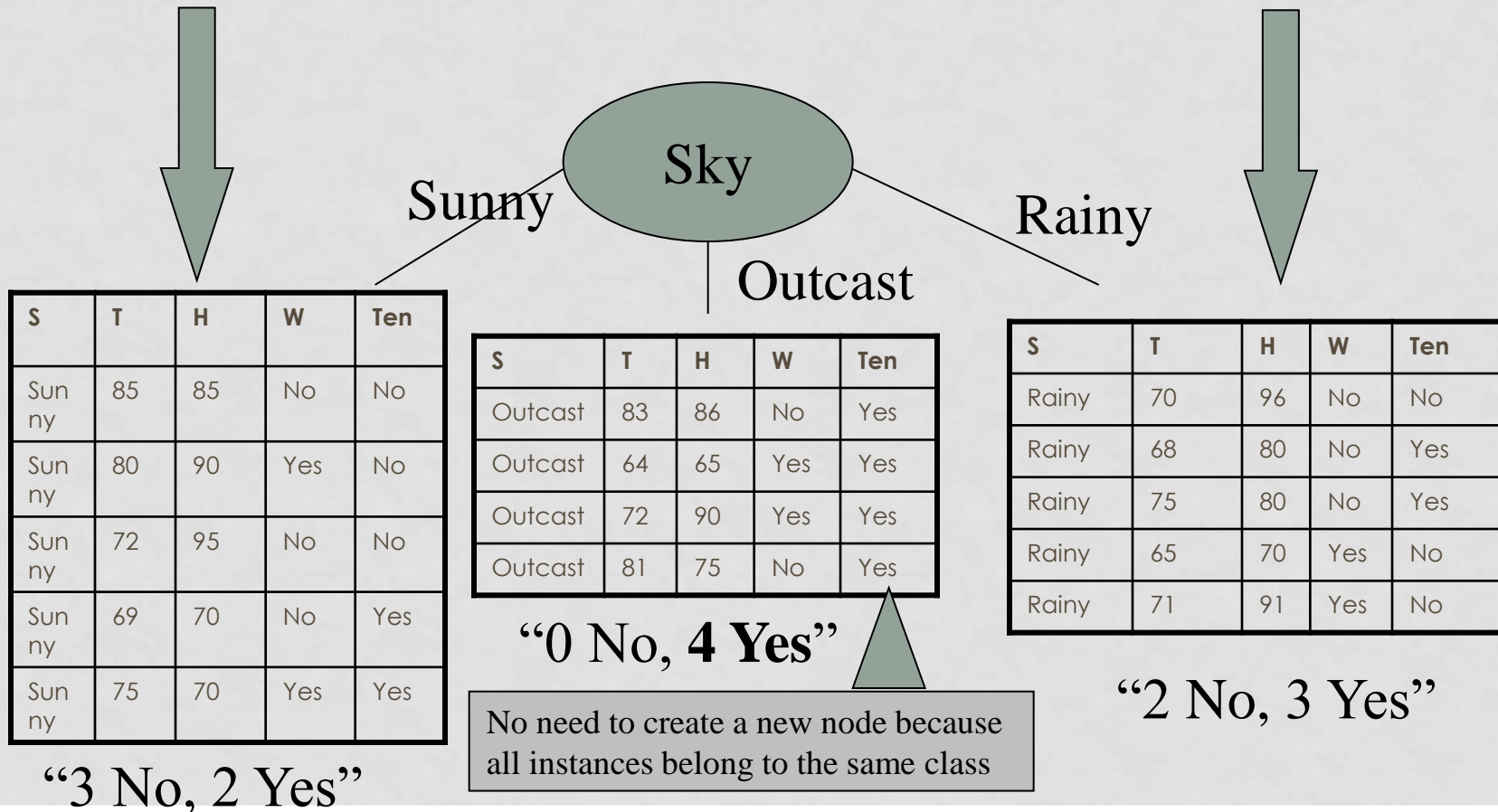
Recursive tree growth

Now that the attribute for the root node has been determined, the process continues recursively. Now, the algorithm has to construct three new subtrees.

S	T	H	W	Ten
Sunny	85	85	No	No
Sunny	80	90	Yes	No
Sunny	72	95	No	No
Sunny	69	70	No	Yes
Sunny	75	70	Yes	Yes

S	T	H	W	Ten
Outcast	83	86	No	Yes
Outcast	64	65	Yes	Yes
Outcast	72	90	Yes	Yes
Outcast	81	75	No	Yes

S	T	H	W	Ten
Rainy	70	96	No	No
Rainy	68	80	No	Yes
Rainy	75	80	No	Yes
Rainy	65	70	Yes	No
Rainy	71	91	Yes	No

"3 No, 2 Yes"   "0 No, 4 Yes"   "2 No, 3 Yes"
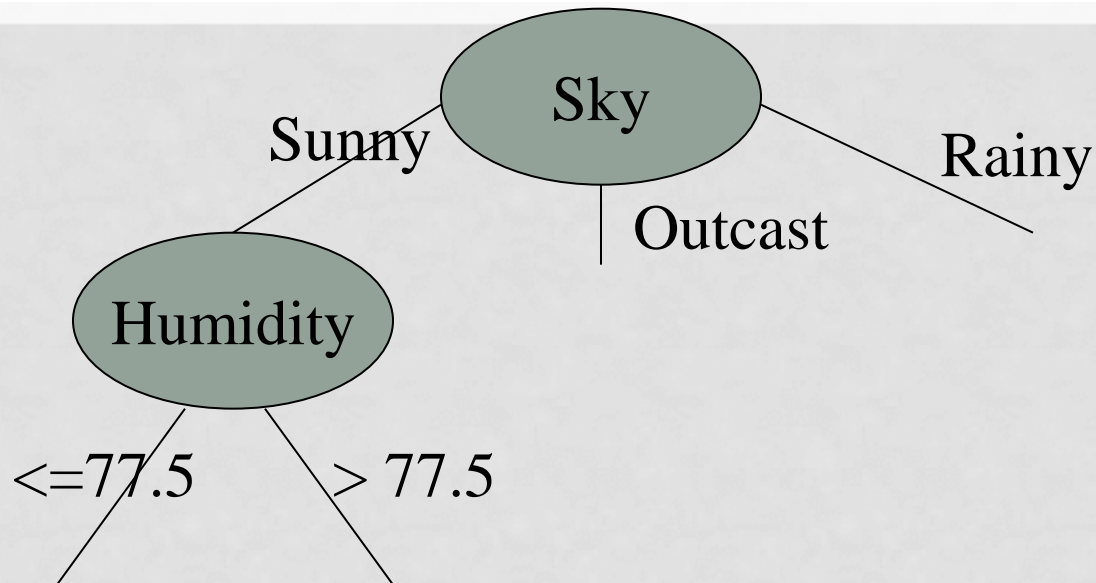
When to stop splitting data?

What is the best option, continue splitting, or classify with the majority class (No).

Use a statistical test. The decision might be based in too few instances.

Sky

Sunny → Outcast → Rainy

S	T	H	W	Ten
Sunny	85	85	No	No
Sunny	80	90	Yes	No
Sunny	72	95	No	No
Sunny	69	70	No	Yes
Sunny	75	70	Yes	Yes

"3 No, 2 Yes"

S	T	H	W	Ten
Outcast	83	86	No	Yes
Outcast	64	65	Yes	Yes
Outcast	72	90	Yes	Yes
Outcast	81	75	No	Yes

"0 No, **4 Yes**"

No need to create a new node because all instances belong to the same class

S	T	H	W	Ten
Rainy	70	96	No	No
Rainy	68	80	No	Yes
Rainy	75	80	No	Yes
Rainy	65	70	Yes	No
Rainy	71	91	Yes	No

"2 No, 3 Yes"

Why stop tree growth?

Sky

Sunny — Outcast — Rainy

Humidity

<=77.5

T	H	V	Ten
69	70	No	Yes
75	70	Yes	Yes

> 77.5

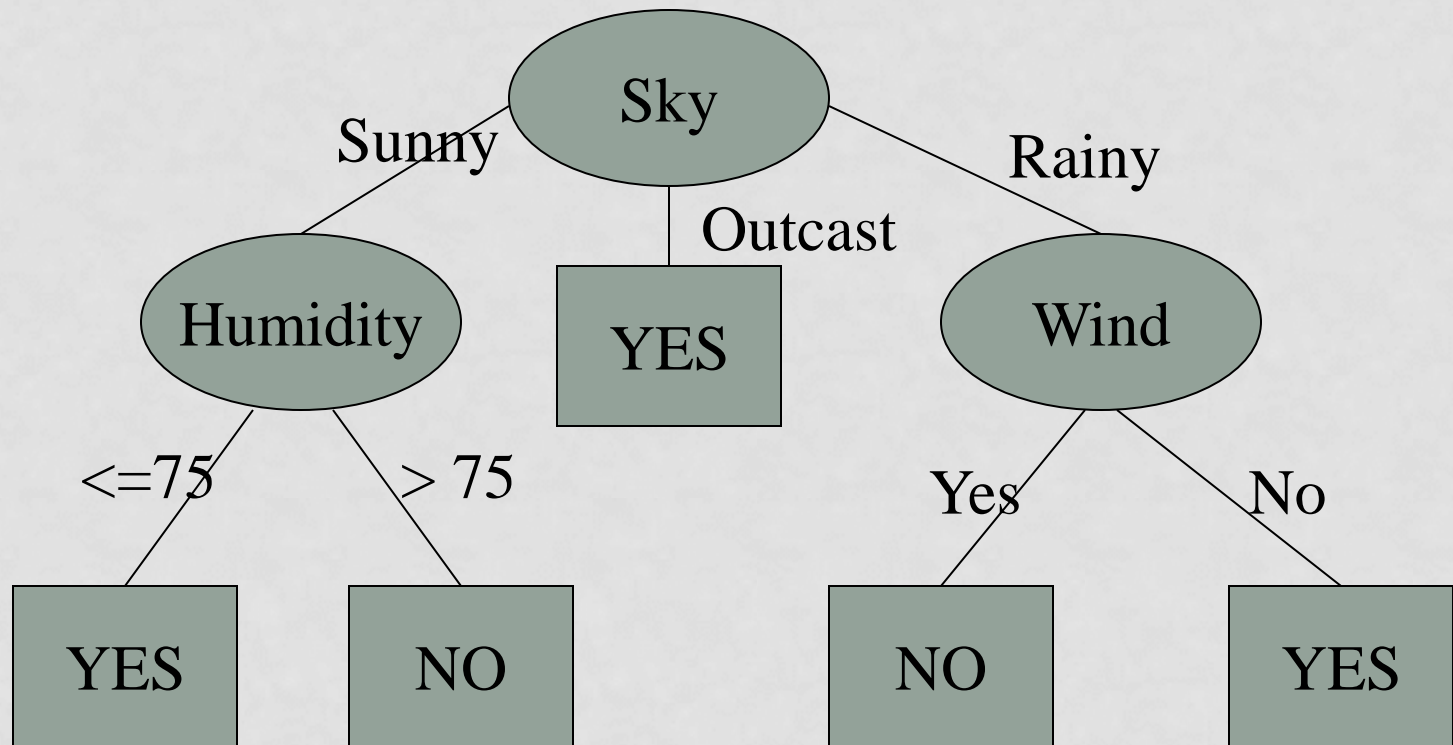| T	H	V	Ten
85	85	No	No
80	90	Yes	No
72	95	No	No

"2 Yes, 0 No"

"3 No, 0 Yes"

No need to create a new node because all instances belong to the same class
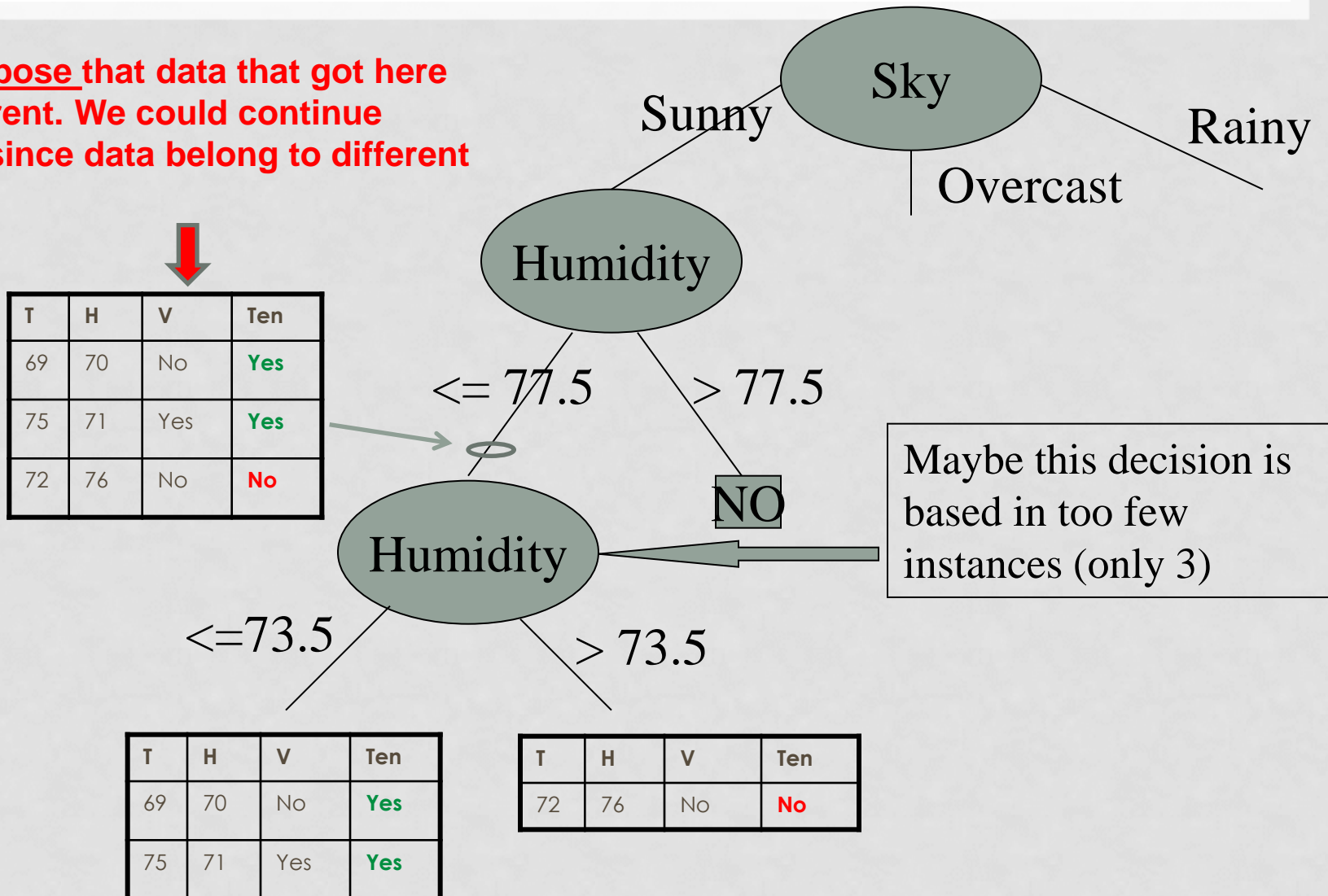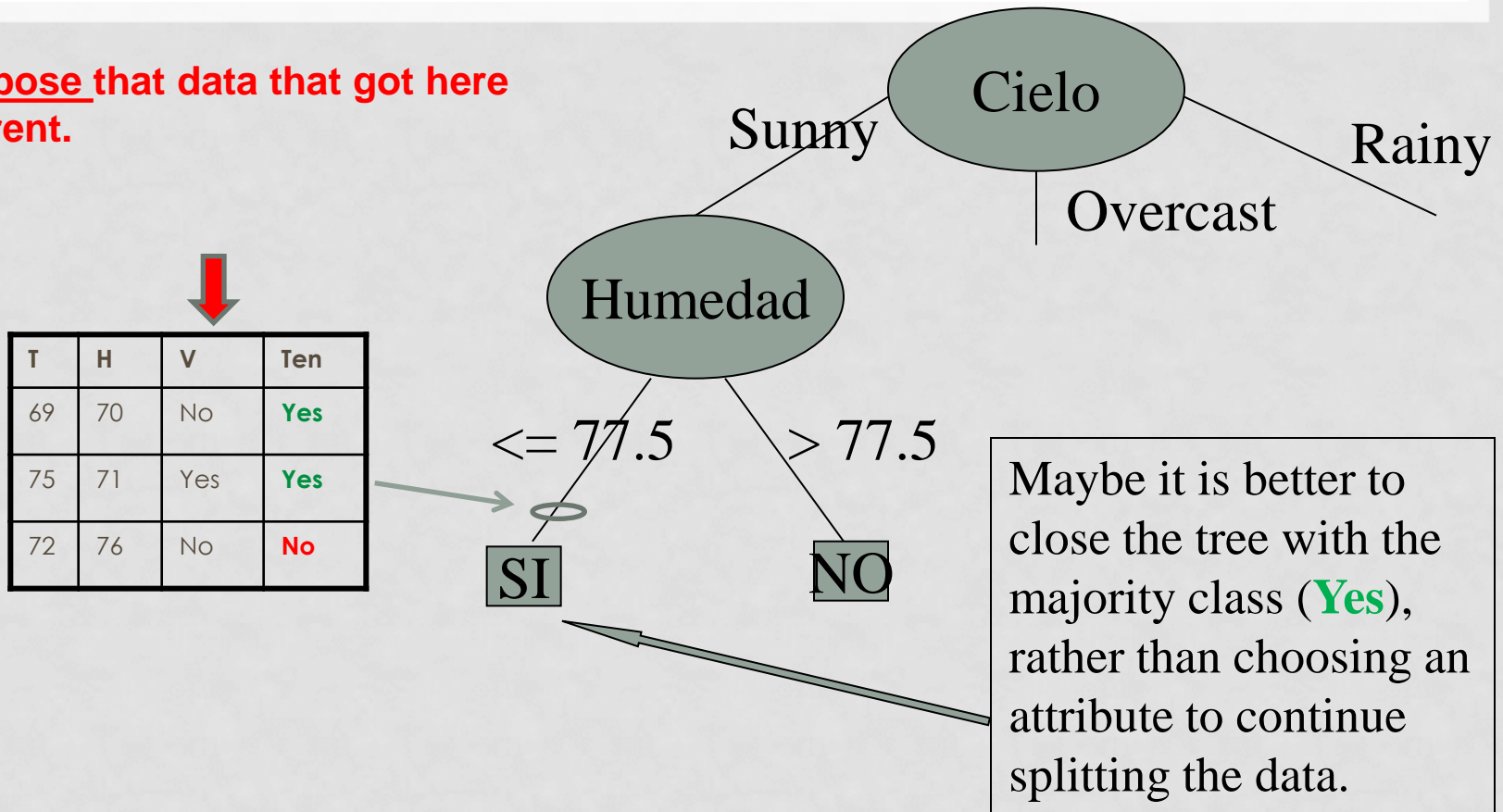
Recursive tree growth
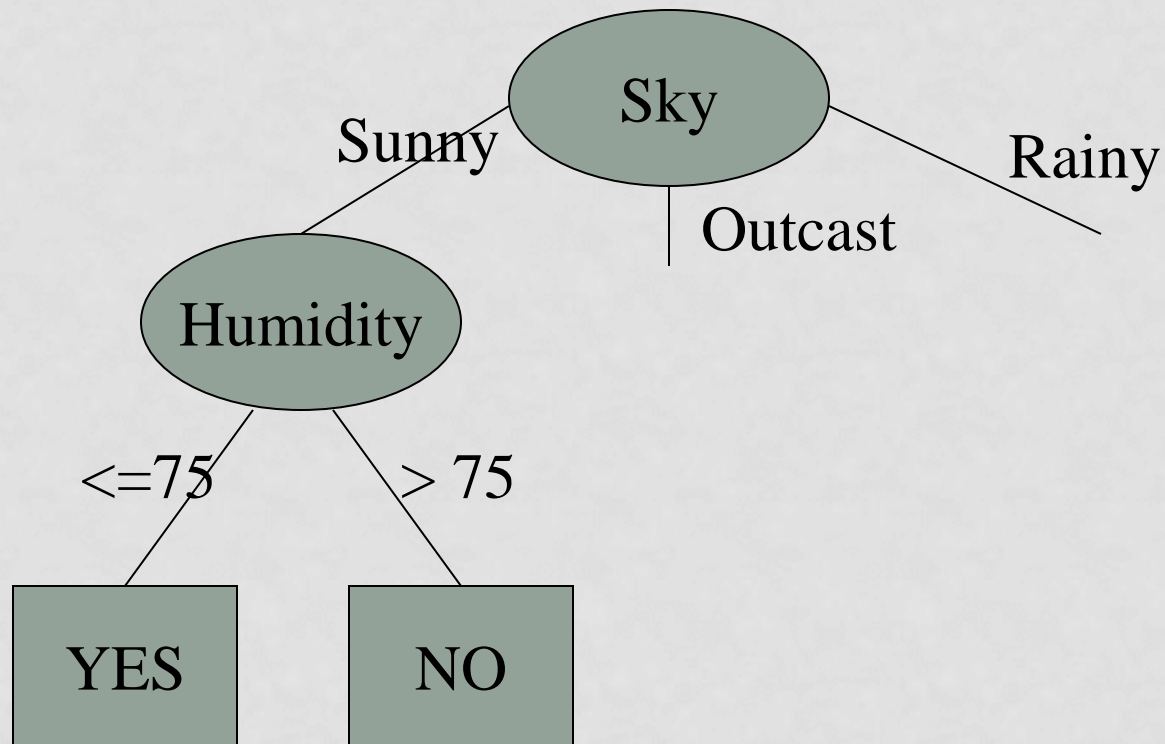
Recursive tree growth

Why to stop tree growth?

Let's suppose that data that got here was different. We could continue splitting since data belong to different classes.

T	H	V	Ten
69	70	No	**Yes**
75	71	Yes	**Yes**
72	76	No	**No**

Sky

Sunny

Overcast

Rainy

Humidity

<= 77.5   > 77.5

NO

Maybe this decision is based in too few instances (only 3)

Humidity

<=73.5   > 73.5

T	H	V	Ten
69	70	No	**Yes**
75	71	Yes	**Yes**

T	H	V	Ten
72	76	No	**No**

Why to stop tree growth?

Let's suppose that data that got here was different.

T	H	V	Ten
69	70	No	Yes
75	71	Yes	Yes
72	76	No	No

Cielo

Sunny

Overcast

Rainy

Humedad

<= 77.5

> 77.5

SI

NO

Maybe it is better to close the tree with the majority class (**Yes**), rather than choosing an attribute to continue splitting the data.

The algorithm uses a statistical criterion in order to determine whether it is worth to continue tree growth, whether the sample is too small, etc.

Rules (created from the decision tree)

Obtain one rule from each path from the root to the leaves

IF Sky = Sunny **AND** Humidity <= 75 **THEN** Tennis = Yes

ELSE IF Sky = Sunny AND Humidity > 75 **THEN** Tennis = No

ELSE IF Sky = Outcast **THEN** Tennis = Yes

ELSE IF Sky = Rainy AND Wind = Yes **THEN** Tennis = Yes

ELSE Tennis = No
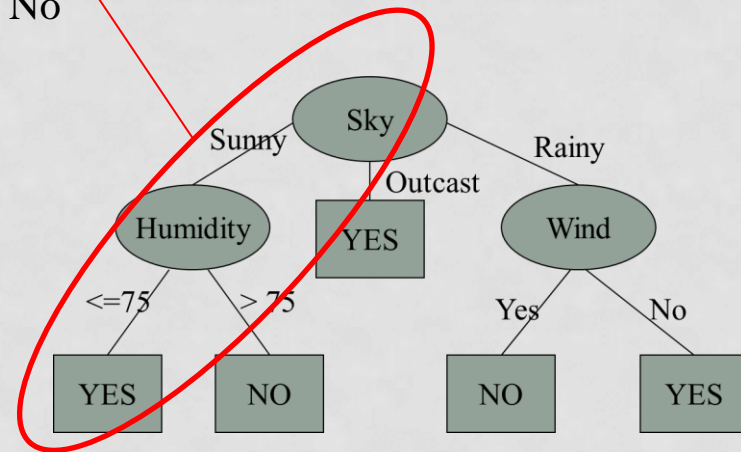
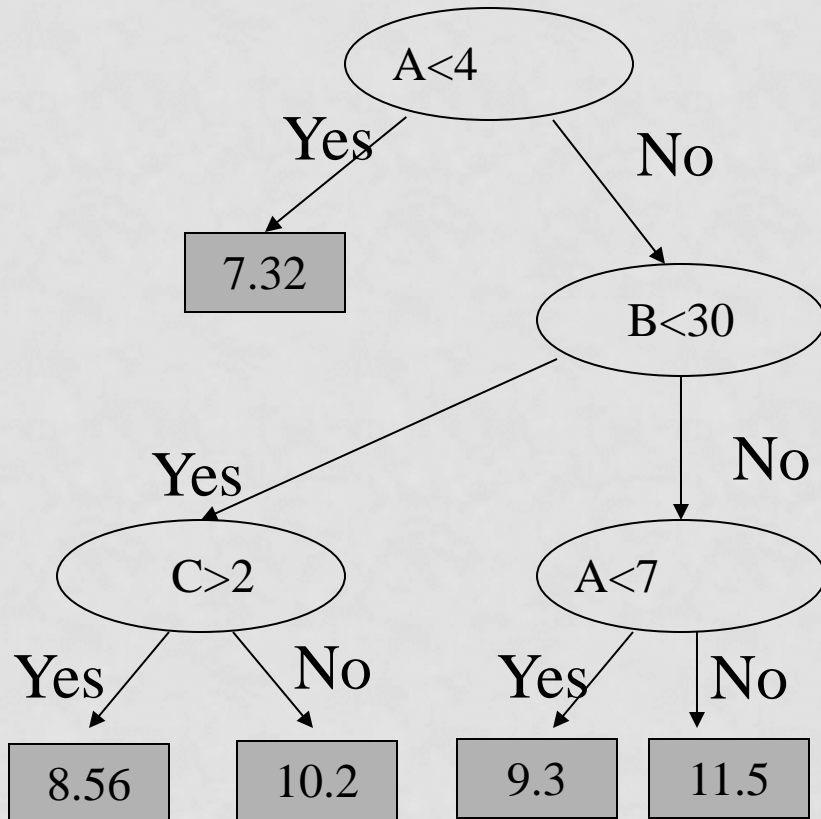But there are algorithms that build rules directly from data

TREES FOR REGRESSION

- What to do if the class / label is continuous (rather than discrete?
  - Answer: variance reduction (instead of entropy reduction)
- Two types:
  - Model trees
  - Regression trees

Example of linear regression
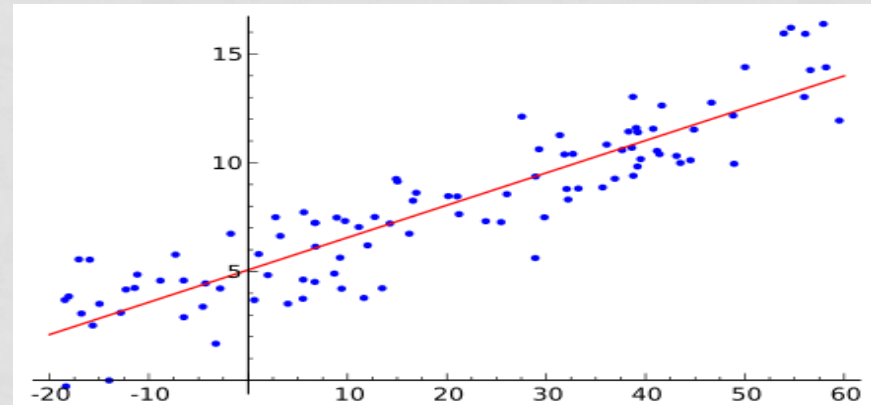
TREES FOR REGRESSION: TWO TYPES

Regression tree

Model tree

Constants
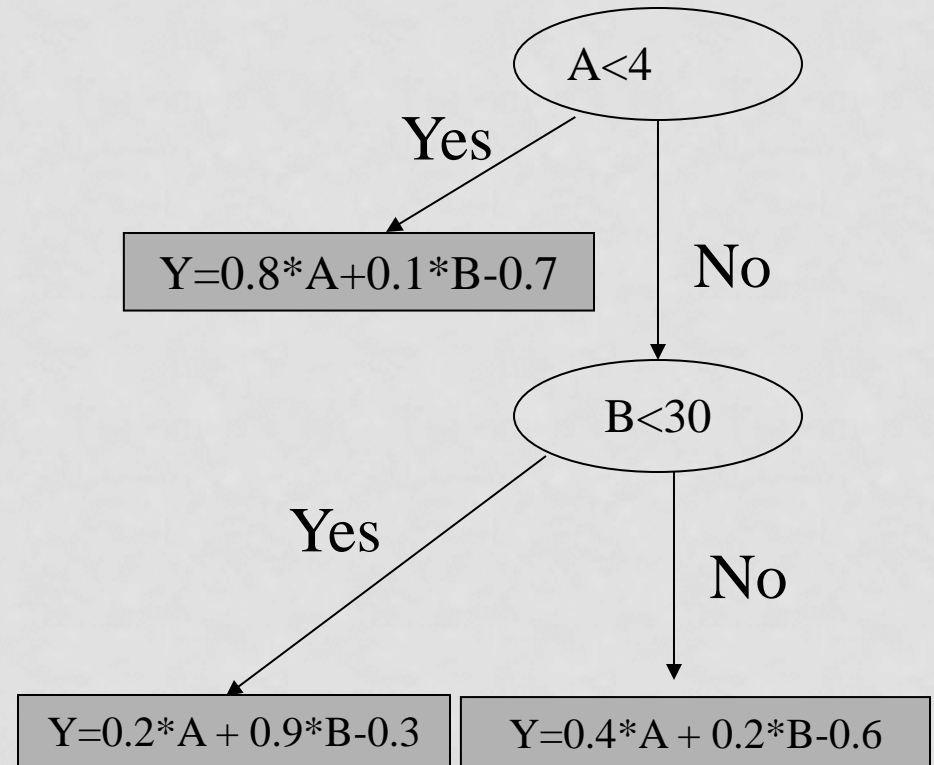
Linear models in the leaves
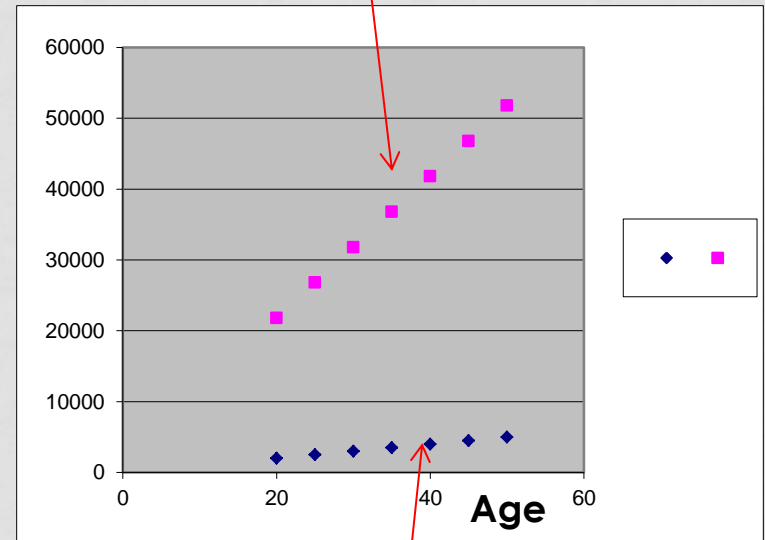
EXAMPLE

Training data

Data Miner?	Age	Salary
Yes	20	2000
Yes	25	2500
Yes	30	3000
Yes	35	3500
Yes	40	4000
Yes	45	4500
Yes	50	5000
No	20	2000
No	25	2050
No	30	2100
No	35	2150
No	40	2200
No	45	2250
No	50	2300

Data miner

Not data miner

MODEL TREES. EXAMPLE
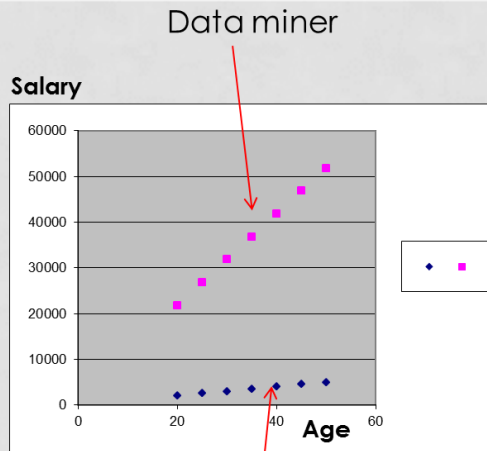
Salary

Data miner

Not data miner

Data miner?

Yes — Salary = 2000+(age-20)*100

No — Salary = 2000+(age-20)*10

REGRESSION TREES. EXAMPLE

Data miner

Salary

Not data miner

Data miner?

Yes

No

Salary = 3500

Salary = 2150
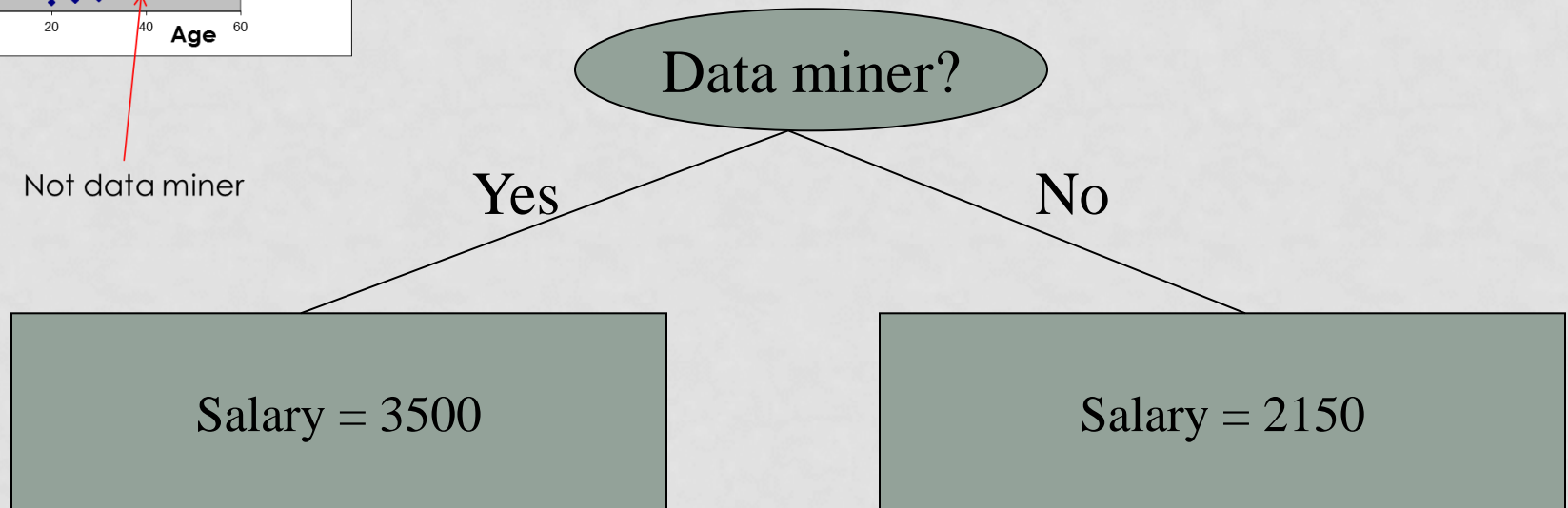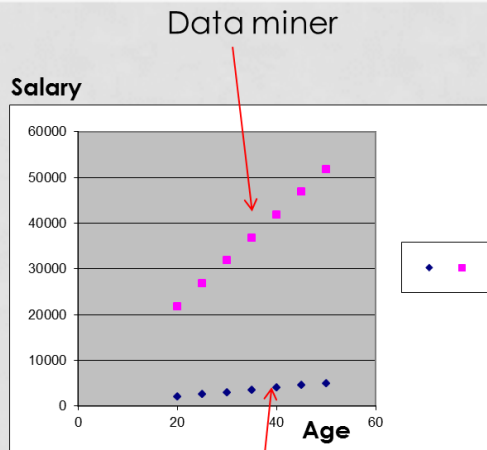
In the leaves, we can see the average salary for data miners (3500 euros) and the average salary for non-data miners (2150 euros)

TREES FOR REGRESSION

- Regression and model trees are built similarly, except that in the leaves
    - For regression trees, the average output value is computed
    - For model trees, a linear model is constructed (M5 (Quinlan, 93))

- Trees for regression are built similarly than trees for classification (decision trees), except that **standard deviation** is reduced (instead of entropy)

- The tree is built recursively until a stopping condition is reached:
    - Too few examples (2)
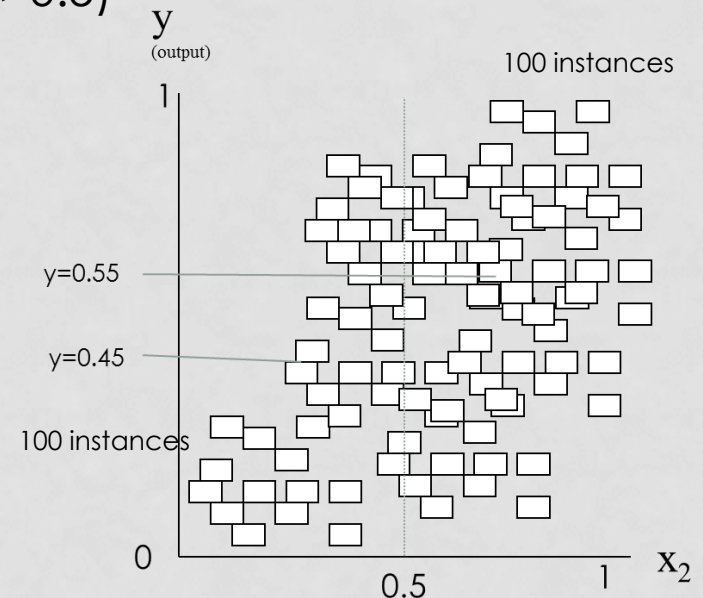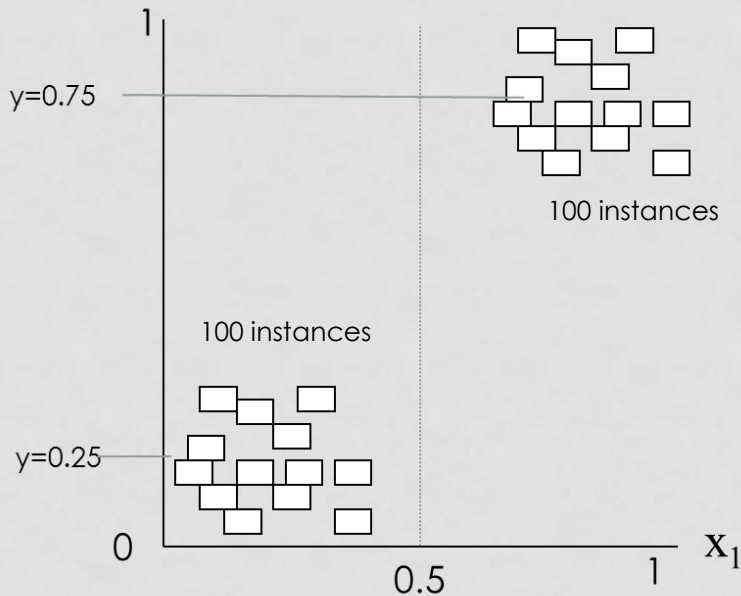   - Standard deviation smaller than 5% of the original sd

WHAT IS THE BEST NODE TO PUT IN THE ROOT OF THE TREE?
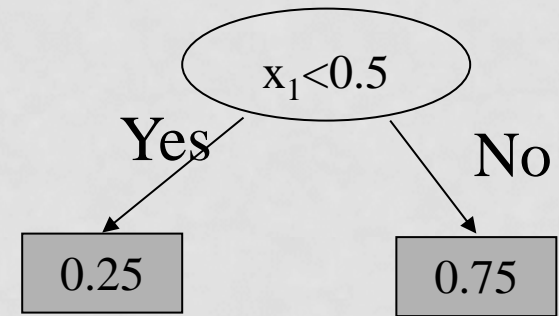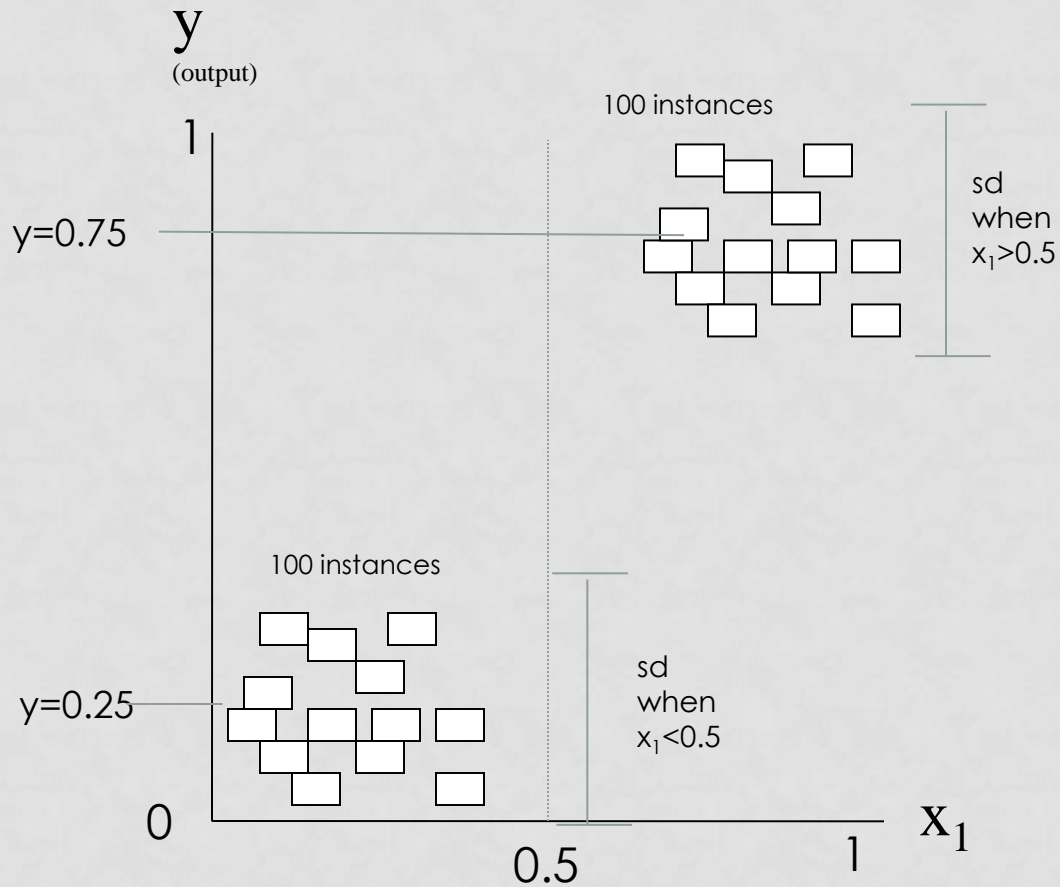
Which attribute is better?
x_1 or x_2?
We will choose the attribute for which the average standard deviation (sd) **after the partition** is small:

$$100/200 * sd(x_i<0.5) + 100/200 * sd(x_i>0.5)$$

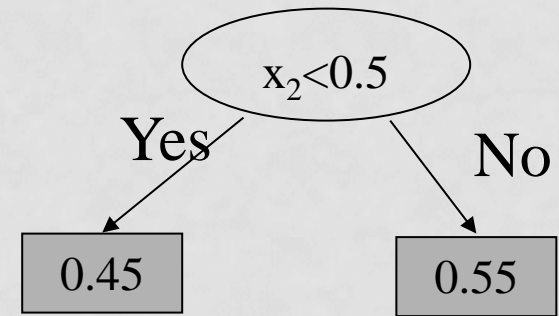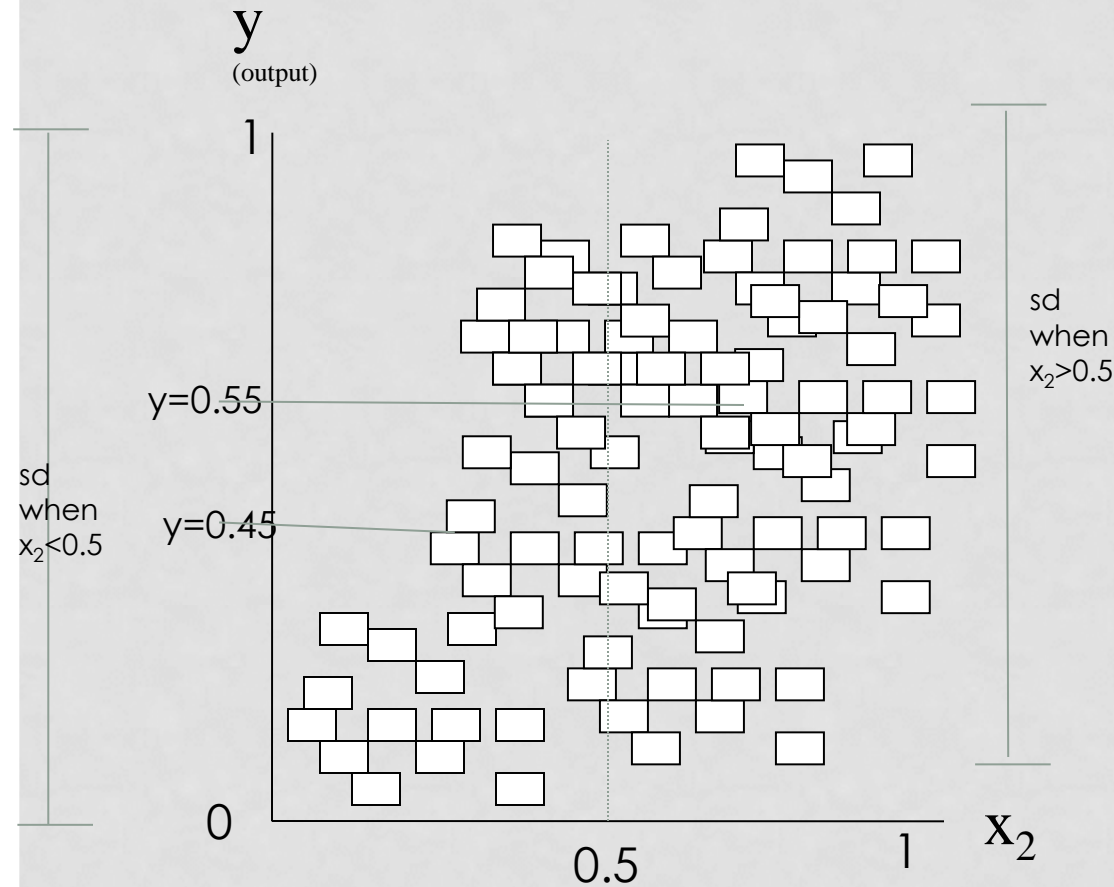# WHAT IS THE BEST NODE TO PUT IN THE ROOT OF THE TREE?

y
(output)

100 instances

sd when $x_1 > 0.5$

y=0.75

$x_1 < 0.5$

Yes   No

0.25   0.75

It can be noticed that x_1 is quite predictive

100 instances

sd when $x_1 < 0.5$

y=0.25

Instances after the partition are very spread

0   0.5   1 x_1

$\frac{1}{2} * sd(x_1 < 0.5) + \frac{1}{2} * sd(x_1 > 0.5)$ is small

WHAT IS THE BEST NODE TO PUT IN THE ROOT OF THE TREE?

y
(output)

1

sd when $x_2>0.5$

y=0.55

sd when $x_2<0.5$

y=0.45

0

0.5   1 x_2

$x_2<0.5$

Yes   No

0.45   0.55

It can be noticed that x_2 is not predictive

Instances after the partition are very spread

½* sd($x_2<0.5$) + ½* sd($x_2>0.5$) is very large

REFERENCES

- M5P: Model trees
- Wang, Y., & Witten, I. H. (1996). Induction of model trees for predicting continuous classes.
- http://researchcommons.waikato.ac.nz/bitstream/handle/10289/1183/uow-cs-wp-1996-23.pdf?sequence=1