MACHINE LEARNING BASIC CONCEPTS

Basic workflow for classification (and regression)

Attributes, **features**, predictors, input variables, independent variables, explanatory variables

Label, **class**, output variable, dependent variable, **response**, predictand, target

Instances, examples, data

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

Training Data / Available data

Future data

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

ML Algorithm / Method

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

THEN Tennis = Yes ...

Model (Classifier)

Class = Yes

Prediction

DEFINITIONS: ATTRIBUTES AND RESPONSE

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

- **Response**
  - If categorical (e.g. cancer / no cancer) => classification problem
    - If it has two values: binary classification, otherwise, multi-class problem
  - If real / integer number => regression problem

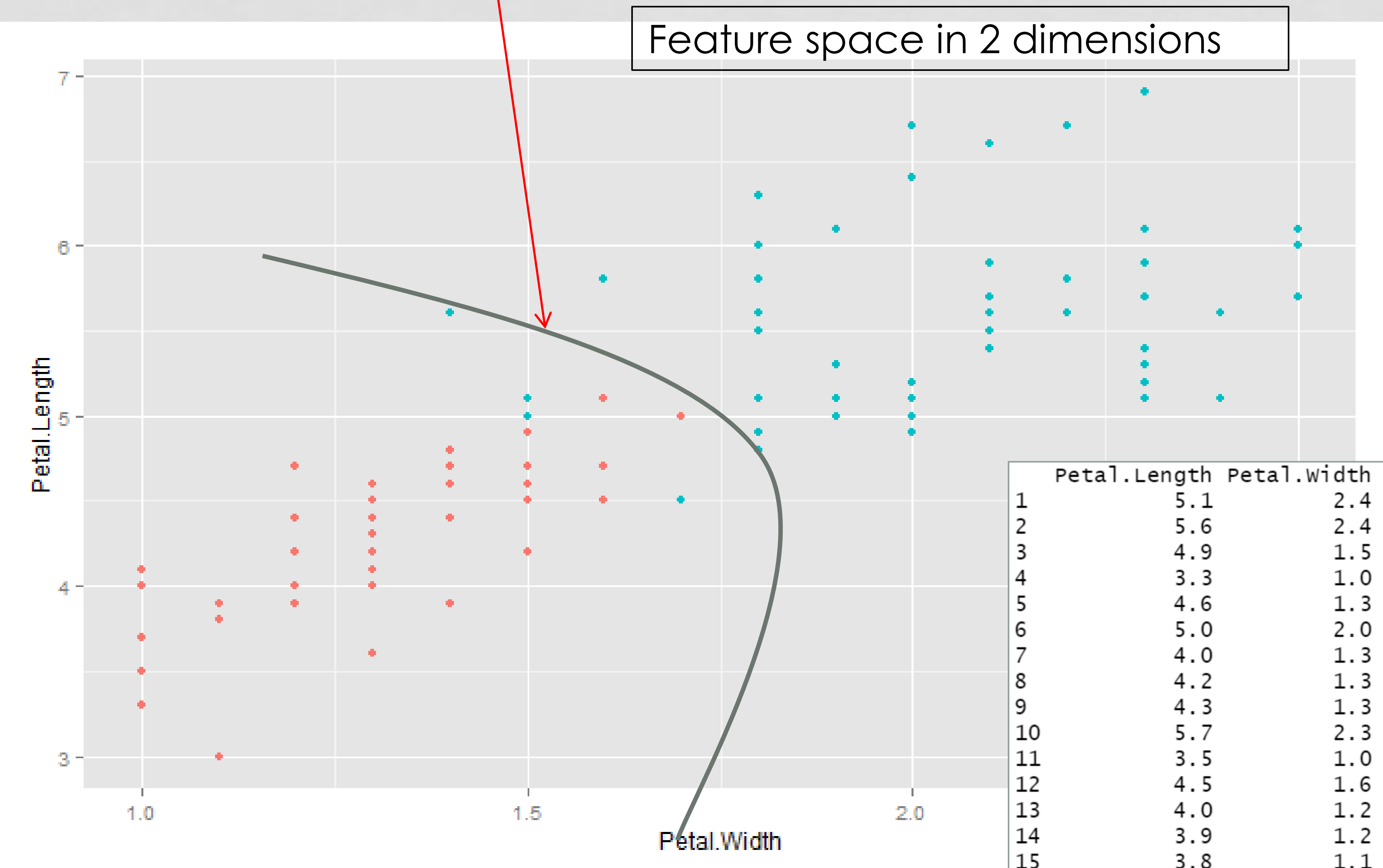
DEFINITIONS: FEATURE SPACE (INSTANCE SPACE)

- Instances, examples = tuples
  - In general, they inhabit a d-dimensional space (instance space)
  - (input, output) = ($\mathbf{x}_i$, y) = (x_{i1}, x_{i2}, …, x_{id}, y) $\in$ ($\boldsymbol{R}^d$, 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

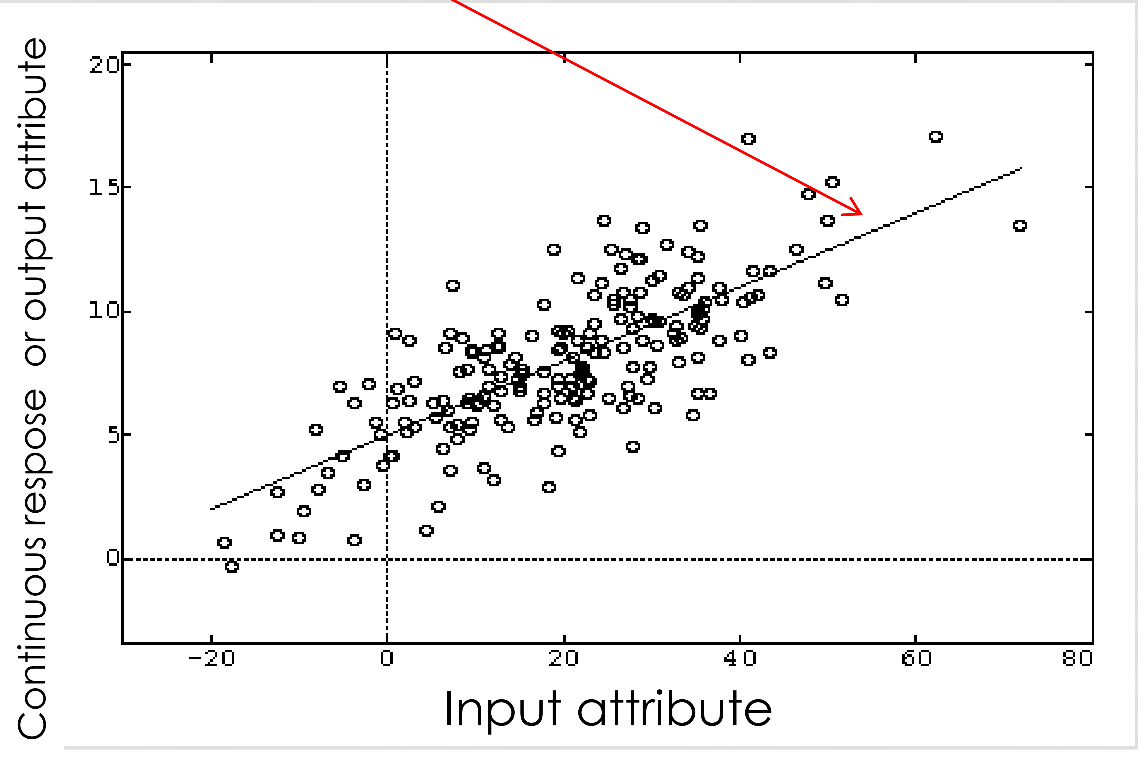
- Example: classify plants into two classes ("versicolor“ / red vs. "virginica“ / blue)
- 2 attributtes = (Petal.Width, Petal.Length) = 2 dimensions
- Classification = finding a boundary between the classes



	Petal.Length	Petal.Width	Species
1	5.1	2.4	virginica
2	5.6	2.4	virginica
3	4.9	1.5	versicolor
4	3.3	1.0	versicolor
5	4.6	1.3	versicolor
6	5.0	2.0	virginica
7	4.0	1.3	versicolor
8	4.2	1.3	versicolor
9	4.3	1.3	versicolor
10	5.7	2.3	virginica
11	3.5	1.0	versicolor
12	4.5	1.6	versicolor
13	4.0	1.2	versicolor
14	3.9	1.2	versicolor
15	3.8	1.1	versicolor

FEATURE 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

Instances, examples

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

Training Data

Future 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

Future data

Instances, examples

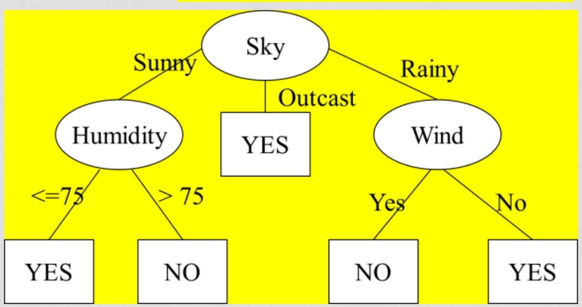
Sky	Temperature	Humidity	Wind	Tennis
Sunny	85	85	No	No
Sunny	80	90	Yes	No
Overcast	83	86	No	Yes
Rainy	70	96	No	So
Rainy	68	80	No	Yes
Overcast	64	65	Yes	Yes
Sunny	72	95	No	No
Sunny	69	70	No	Yes
Rainy	75	80	No	Yes
Sunny	75	70	Yes	Yes
Overcast	72	90	Yes	Yes
Overcast	81	75	No	Yes
Rainy	71	91	Yes	No

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

Instances, examples

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

Training Data

Future data

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

ML Algorithm

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

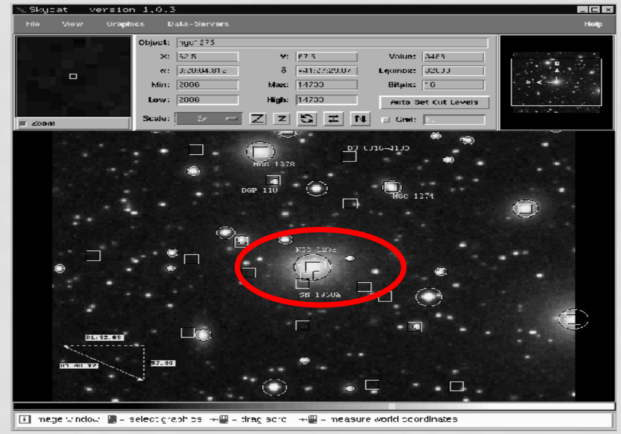
Model (Classifier)

Class = Yes

Prediction

KNN: K-NEAREST NEIGHBOURS

KNN is a lazy method, because the "model" is the data

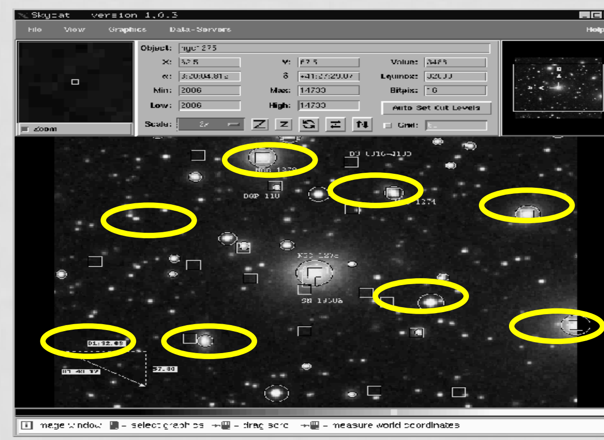


?

Training data

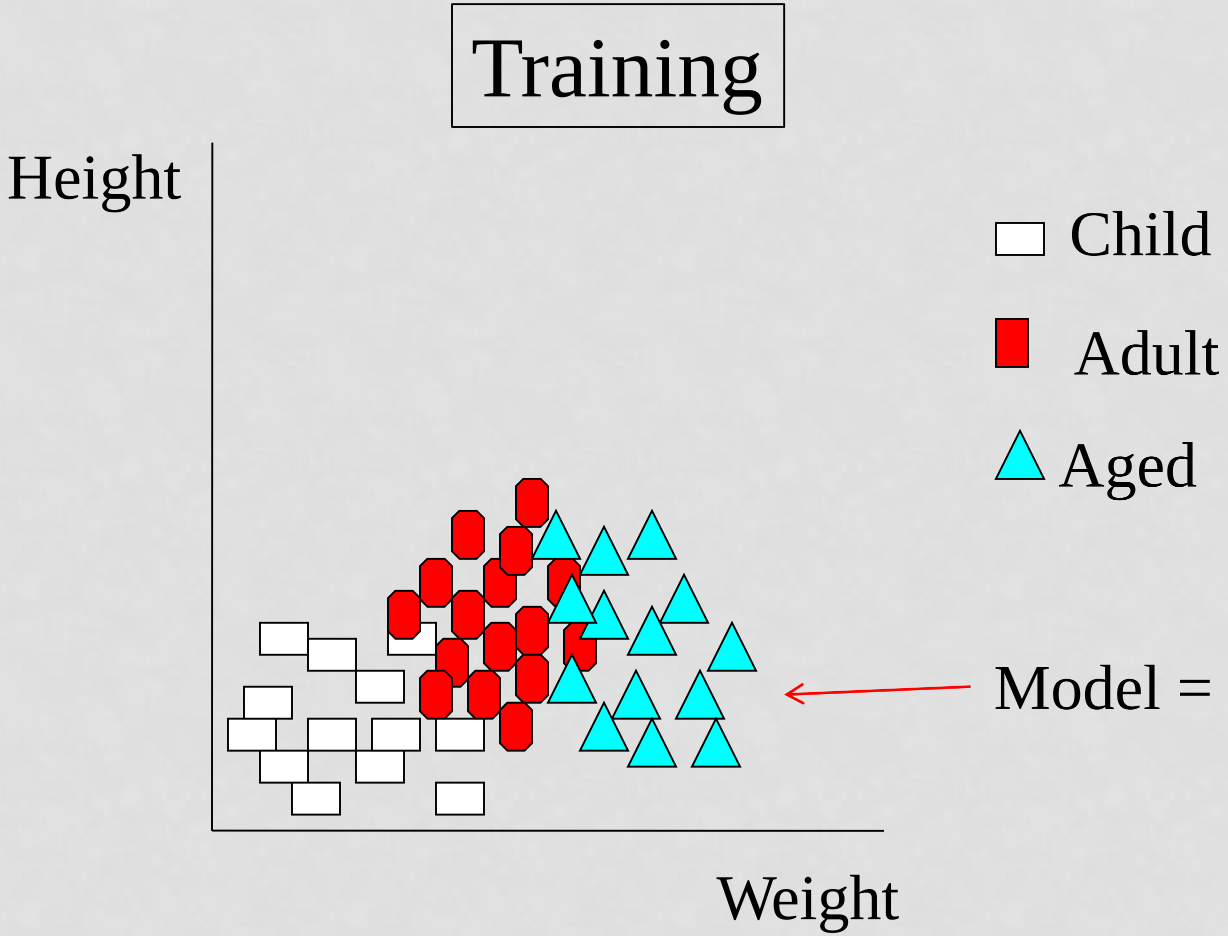
KNN

Model



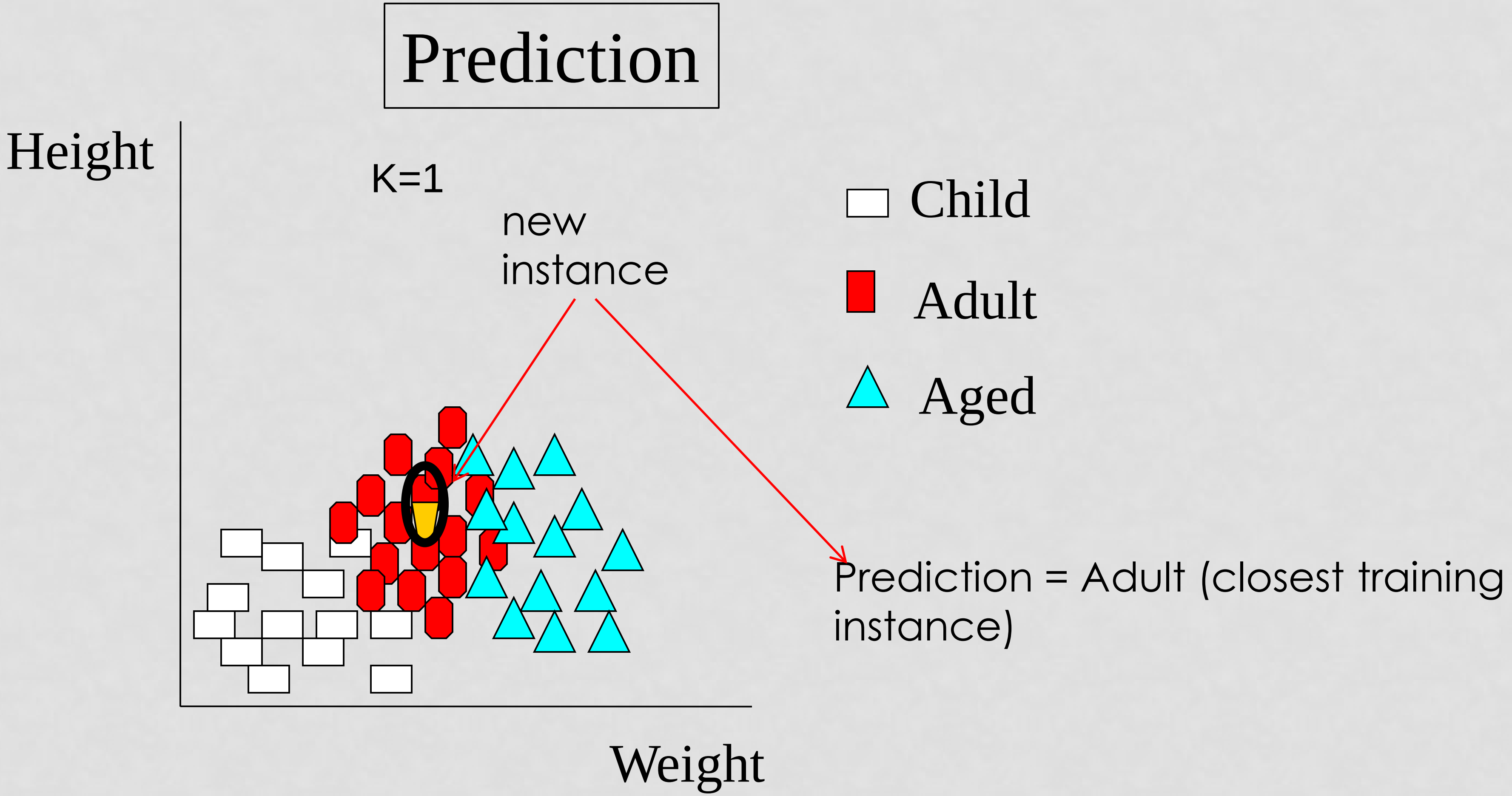
Spiral galaxy

K-NEAREST NEIGHBORS (KNN)



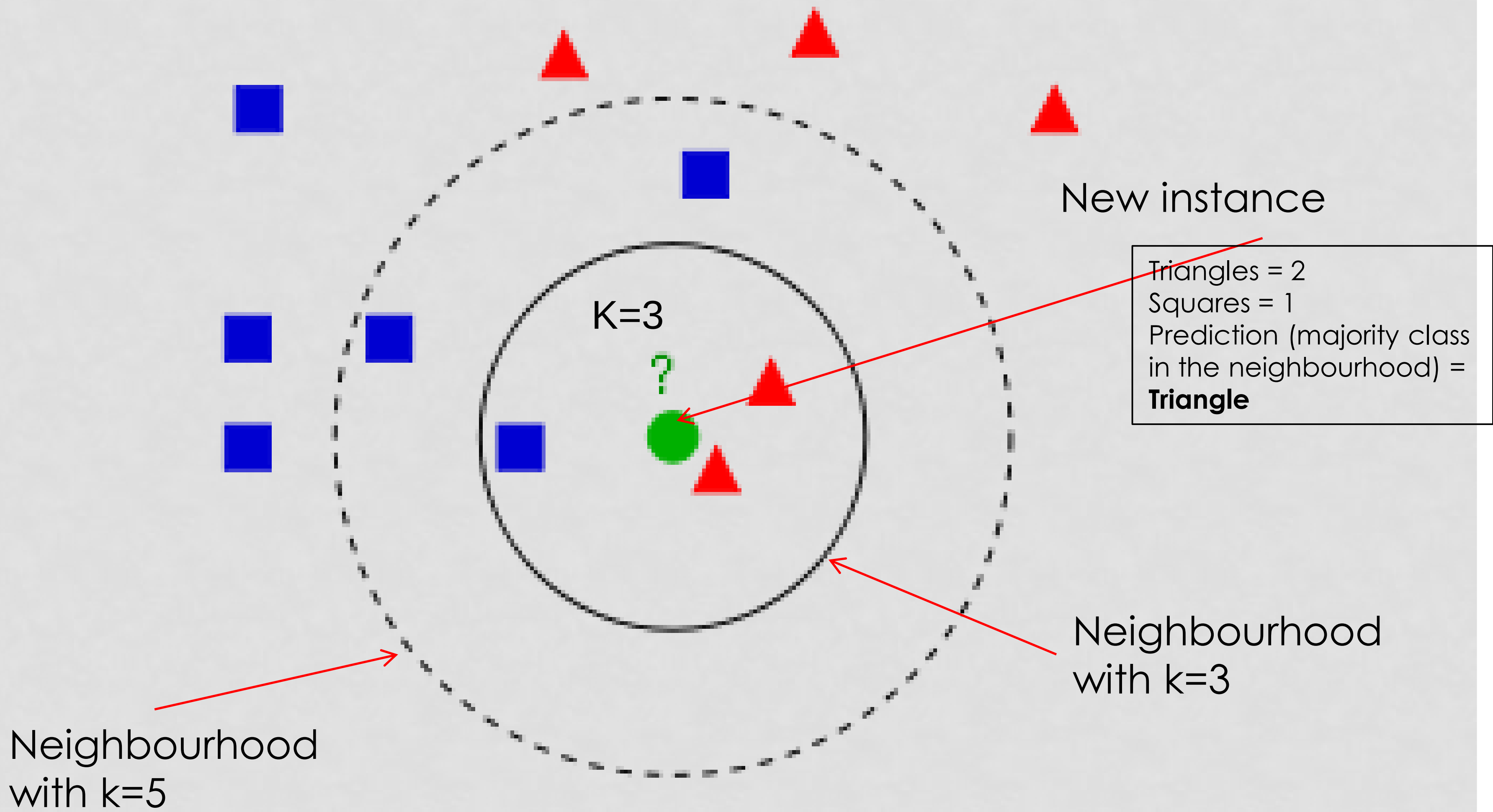
Model = all training instances are stored

K-NEAREST NEIGHBORS (KNN)



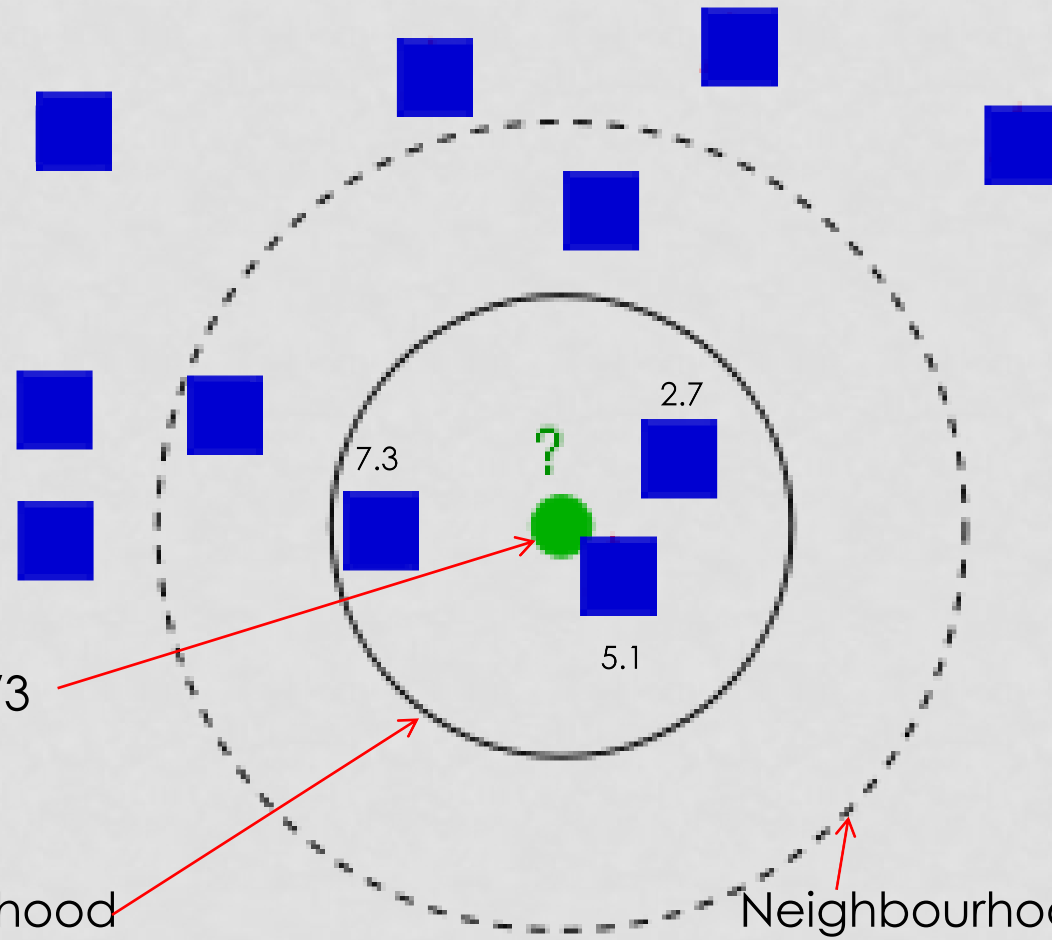
K-NEAREST NEIGHBORS (KNN)

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



KNN FOR REGRESSION

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



Prediction = (7.3+2.7+5.1)/3

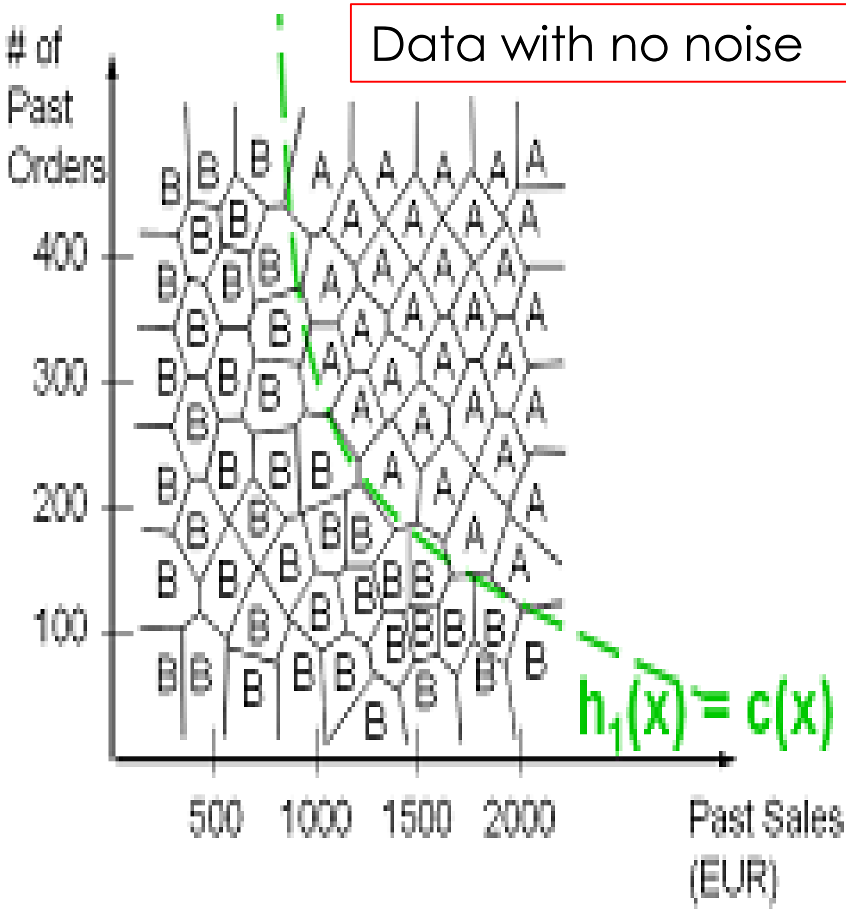
Neighbourhood with k=3

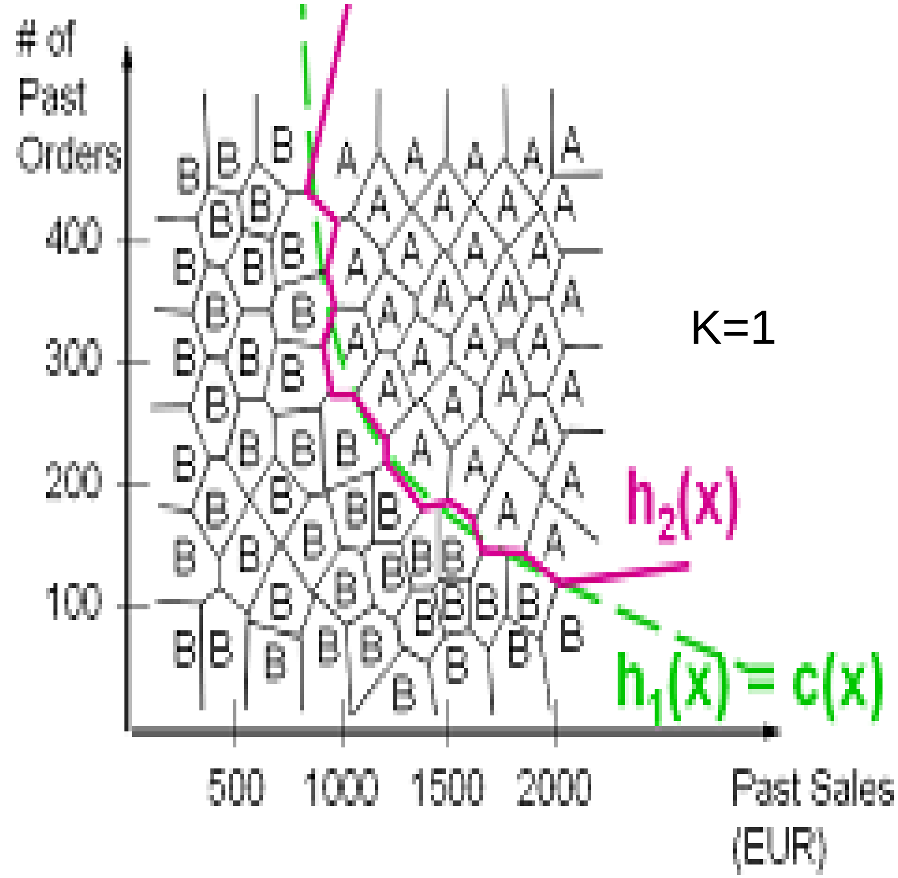
Neighbourhood with k=5

WHY USE K>1?

VORONOI TESSELLATION in Instance space

Data with no noise

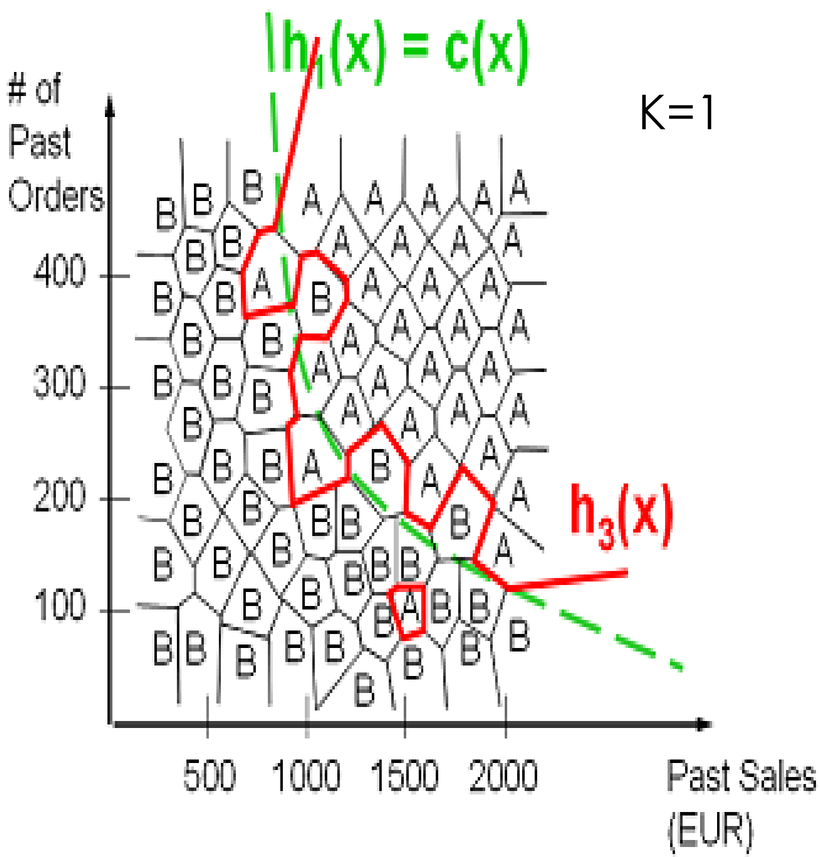




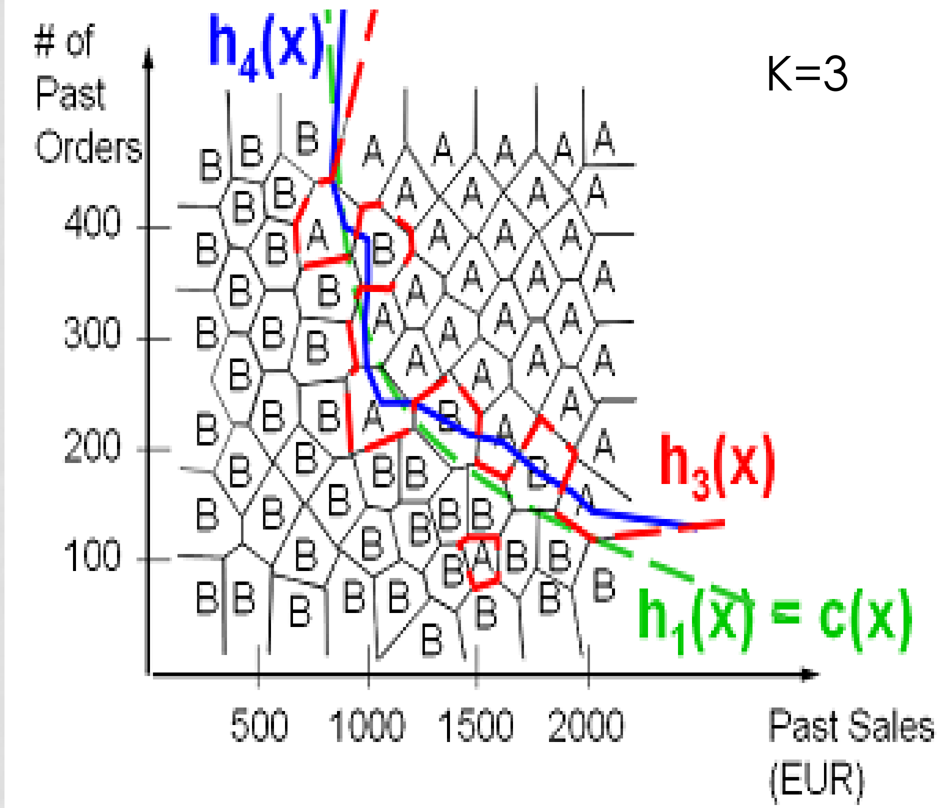
K=1

WHY USE K>1?

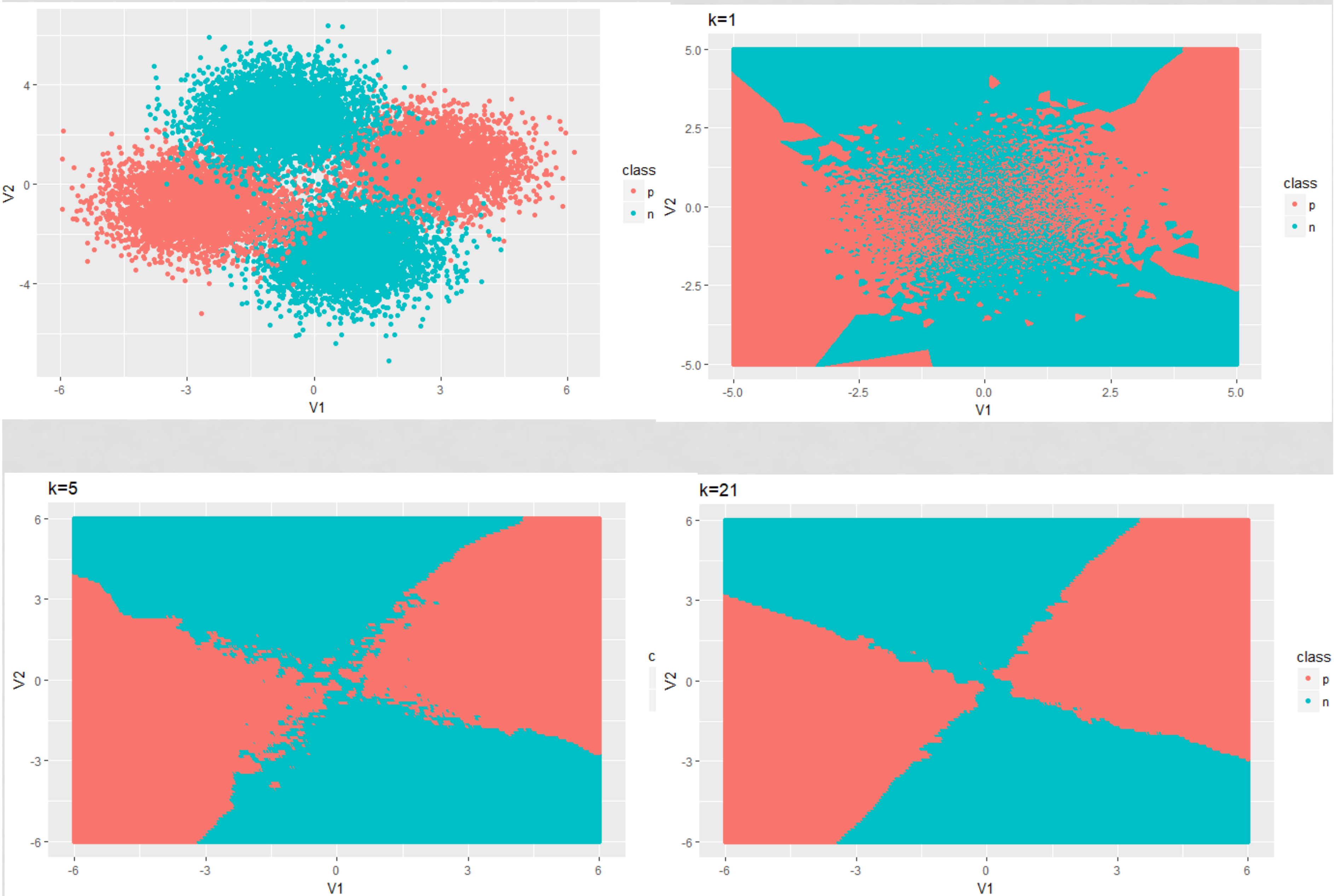
Data with **noise**



(a) 1-NN on noisy data



(b) 3-NN and noisy data



V2
V1
class
p
n
k=1
k=5
k=21

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 the main **hyper-parameter** of KNN and has to be tuned properly.

DISTANCES

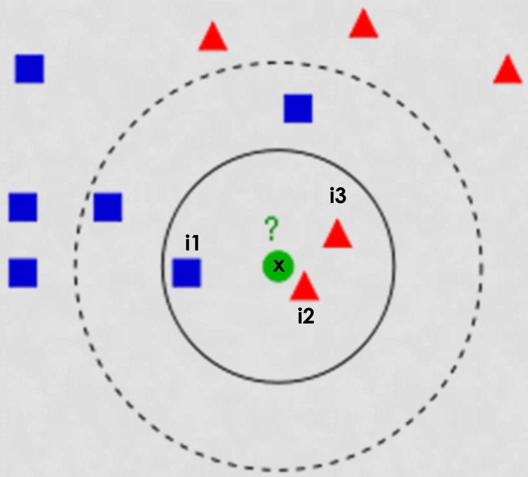
- instance #i: $\mathbf{x_i}$ = (weight$_i$, height$_i$) = (w_i, h_i)
- instance #k: $\mathbf{x_k}$ = (weight$_k$, height$_k$) = (w_k, h_k)
- For numerical attributes, the Euclidean distance is typically used:
  - 2D: $d(\mathbf{x_i},\mathbf{x_k})^2 = (w_i-w_k)^2 + (h_i-h_k)^2$
    - $d(\mathbf{x_i},\mathbf{x_k}) = \mathrm{sqrt}[\ (w_i-w_k)^2 + (h_i-h_k)^2\ ]$
  - dD: $d(\mathbf{x_i},\mathbf{x_k})^2 = (x_{i1}-x_{k1})^2 + (x_{i2}-x_{k2})^2 + \ldots + (x_{id}-x_{kd})^2$
- For nominal / categorical attributes: Hamming distance:
  - If attribute *e* es nominal (categorical), instead of $(x_{ie}-x_{ke})^2$, the following is used: $\delta(x_{ie}, x_{ke})$: 0 if $x_{ie}=x_{ke}$ and 1 otherwise
- or transform the attribute to dummy variables / one-hot encoding)

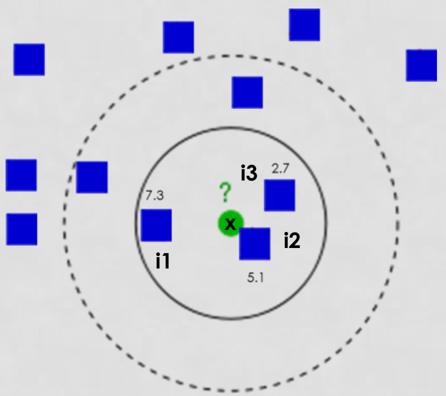
SCALING (NORMALIZATION)

- It is important to scale (normalize) attributes, because ranges can be different (e.g. human body temperature ranges from 35º to 45º celsius, body height ranges from 0 to 2m, body weight ranges from 0 to 100kg, age ranges from 0 to 100 years, etc.)
- Otherwise, attributes with a large range have more weight on the distance
- Scaling attribute x_1:
  - To 0-1 range (minmax): $x'_{1j} = \frac{x_{1j} - \min(x_1)}{\max(x_1) - \min(x_1)}$
  - Standarization: $x'_{1j} = \frac{x_{1j} - \bar{x}_1}{\sigma_1}$

OTHER DISTANCES

- In some cases, it may be better to weight the neighbors so that nearer neighbors contribute more to the classification or regression
- The inverse of the distance is typically used
- "uniform" (euclidean distance) vs. "distance" (inverse euclidean distance)





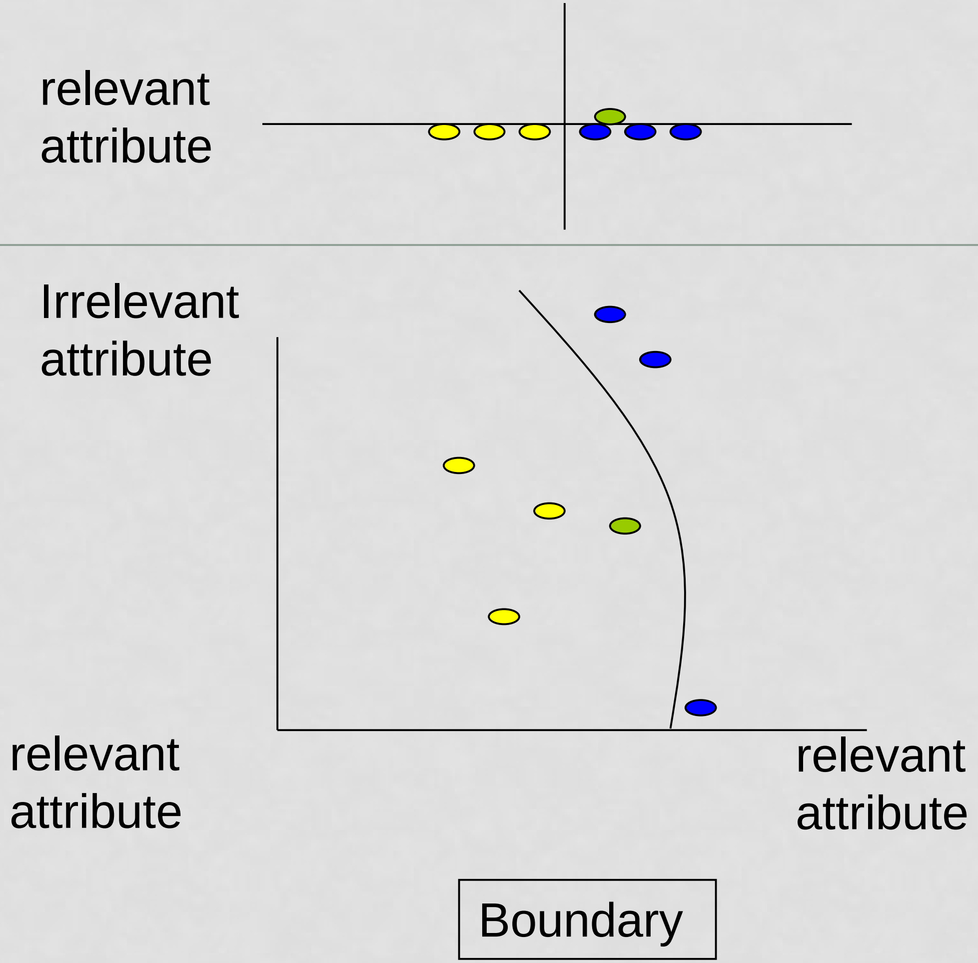
- Blue square: $\frac{1}{d1}$
- Red triangle: $\frac{1}{d2}+\frac{1}{d3}$

$$\frac{\frac{7.3}{d1}+\frac{5.1}{d2}+\frac{2.7}{d3}}{\frac{1}{d1}+\frac{1}{d2}+\frac{1}{d3}}$$

LIMITATIONS OF KNN

- Very sensitive to noise.
  - Solution: Large K's
- Slow (when testing): all distances to each training instance must be computed.
  - Solution: ball-trees
- Large storage requirements (all training data is stored).
  - Possible solution: pre-processing with "condensation" (store only the relevant instances)
- Very sensitive to irrelevant attributes and the curse of dimensionality.
  - Solutions: pre-processing with feature selection / feature extraction

Irrelevant attributes



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

MODELS: TREES (AND RULES) FOR CLASSIFICATION AND REGRESSION

MODELS: DECISION TREES

Attributes, features,
Input variables,
Independent variables

Label, class, output variable,
dependent variable

Instances, examples

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

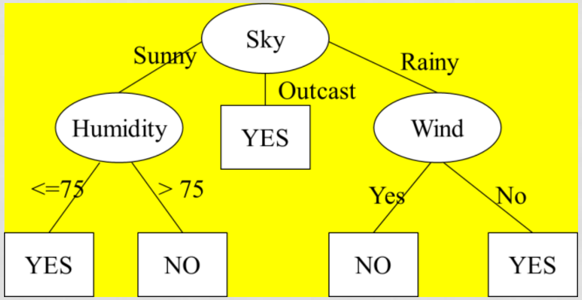
Training Data

Future instance

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

ML
Algorithm

DECISION TREE



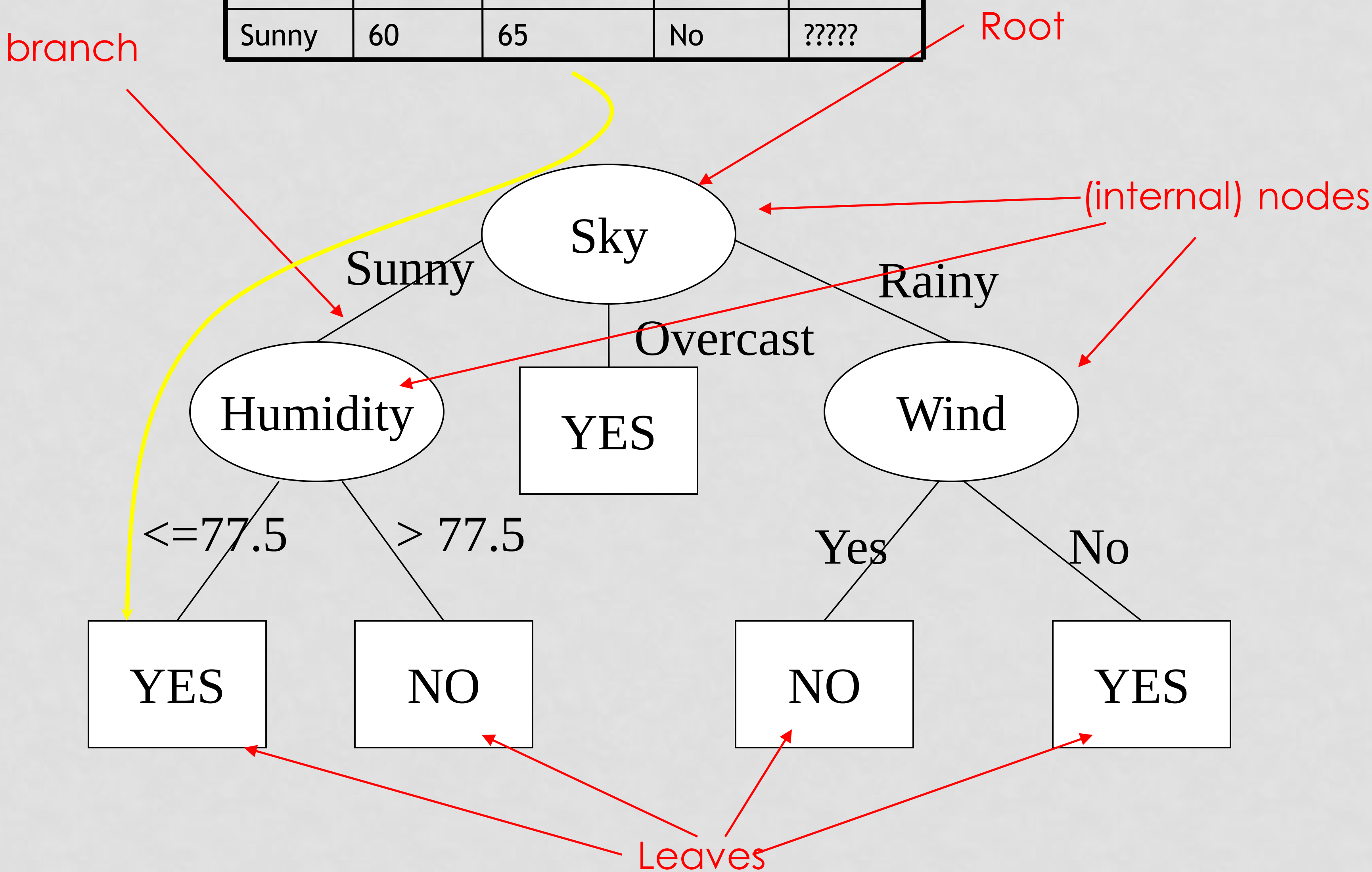
Model (Classifier)

Class = Yes

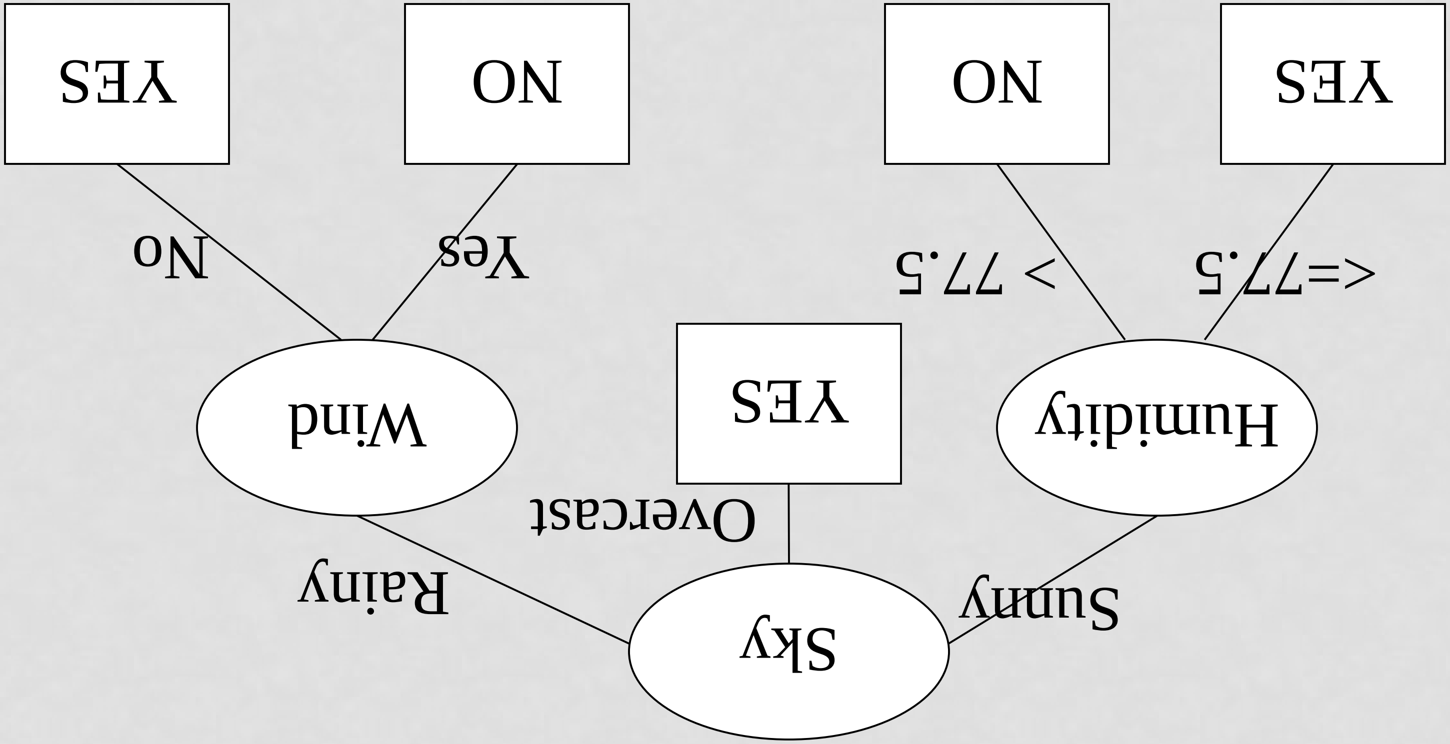
Prediction

Decision trees

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



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)

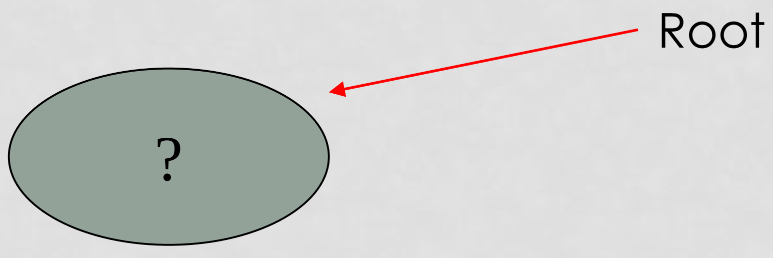
Simplified ID3 algorithm

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

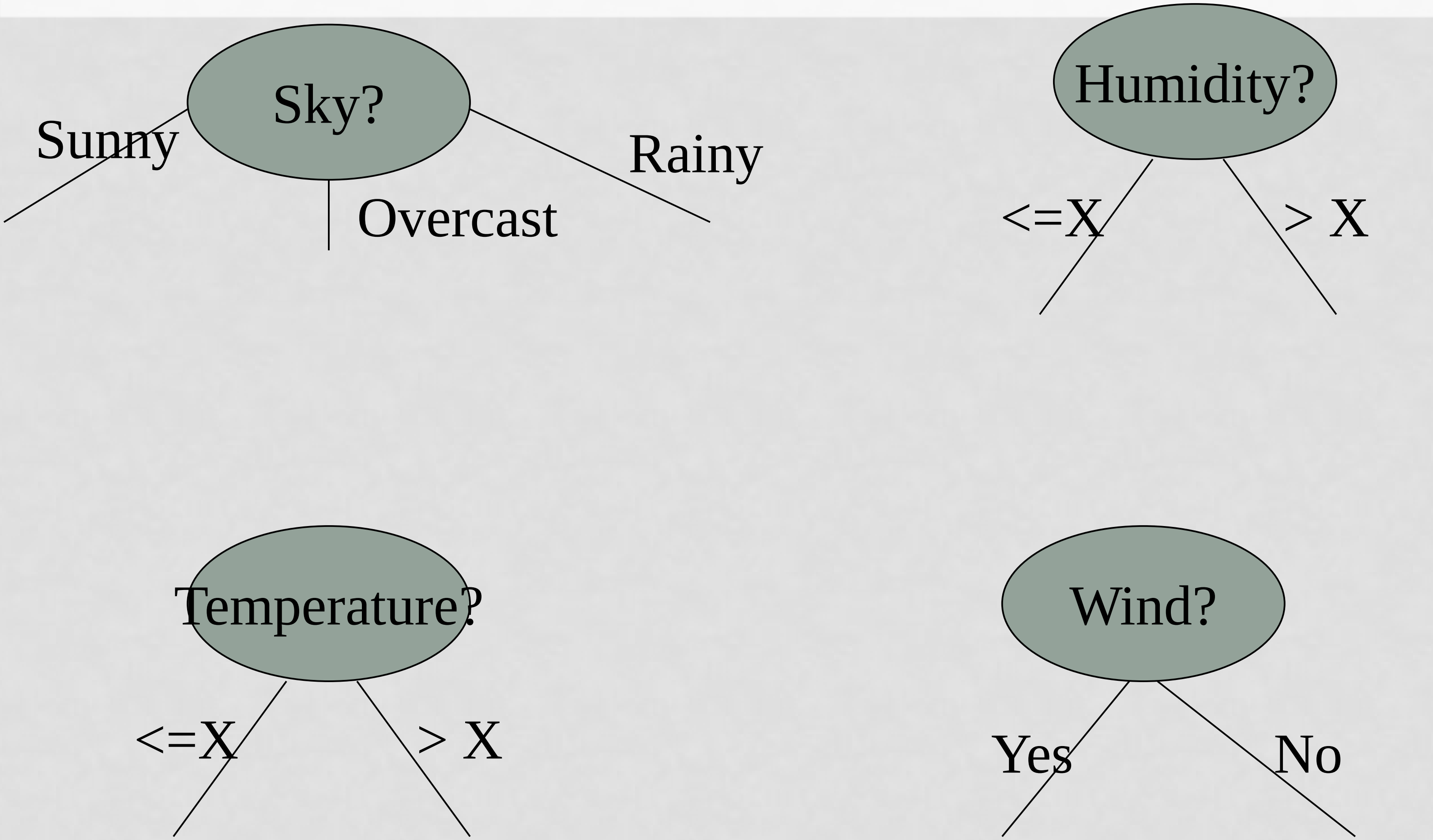
Simplified C4.5 algorithm

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

THE CONSTRUCTION OF THE TREE STARTS WITH AN EMPTY TREE, AT THE ROOT



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



Let's try with attribute SKY

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

Sky

Sunny

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

Overcast

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

"0 No, 4 Yes"

Perfect partition

Rainy

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"

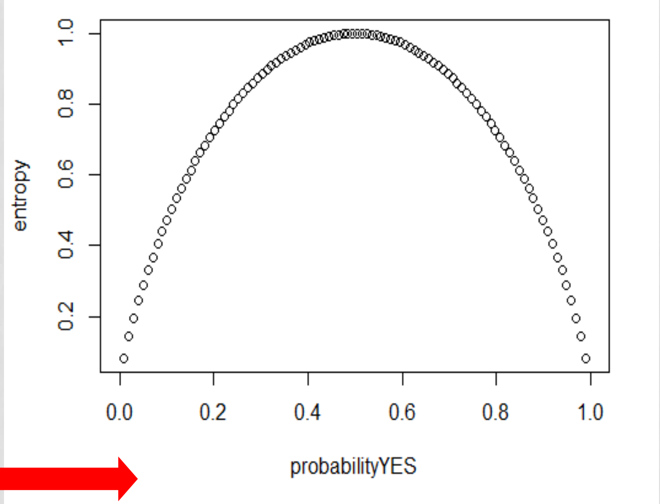
"yes" majority

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

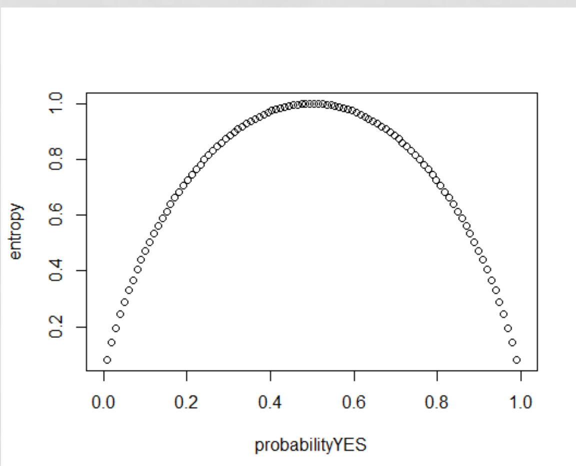
$$H(P) = -\sum_{Ci} p_{Ci} \log_2(p_{Ci})$$

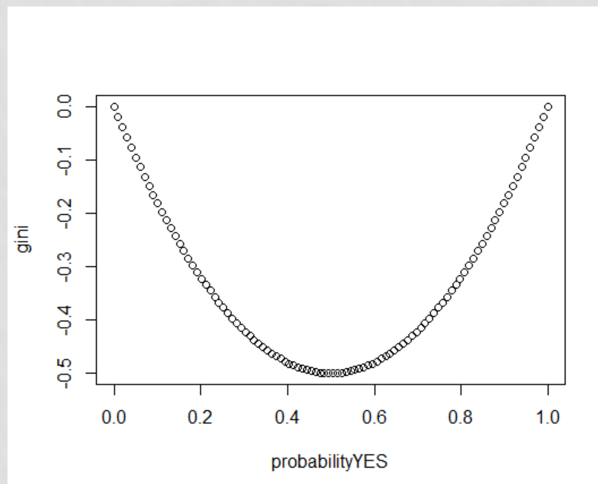
Gini

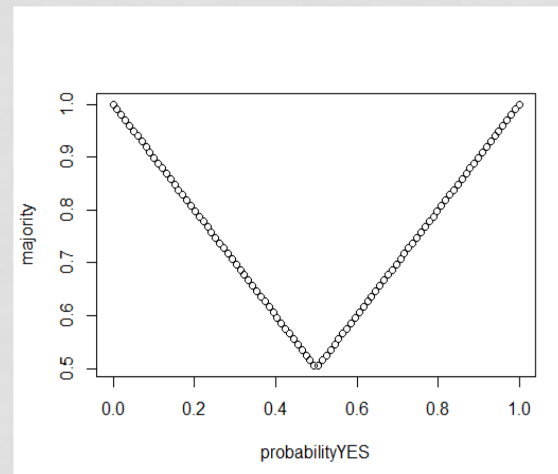
$$Gini(P) = -\sum_{Ci} p_{Ci}(1 - p_{Ci})$$

Majority

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







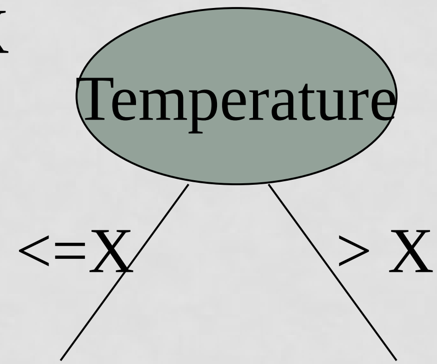
Average entropy for Sky

- 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?

A binary (two-values) attribute is created by computing a threshold X



X?

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

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

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 – YesNo, 75 – YesYes 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

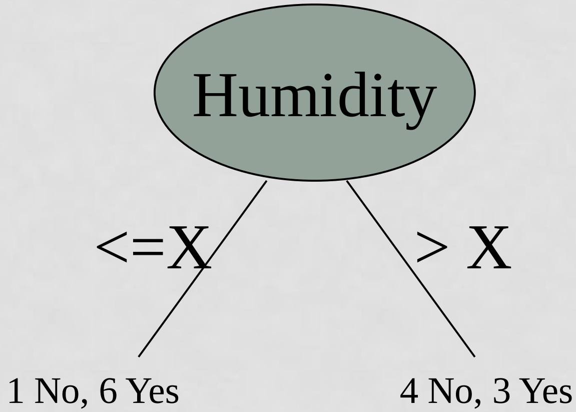
- The actual threshold may depend on the algorithm implementation. Some implementations use the average: Ej: 64.5 = (64+65)/2.
- Other implementations use the maximum of the left partition. In that case, the possible thresholds would have been:

64 – Yes, 65-No, 68 – Yes, 69 – Yes, 70 – Yes, 71 – No, 72 – YesNo, 75 – YesYes 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



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

X=82.5

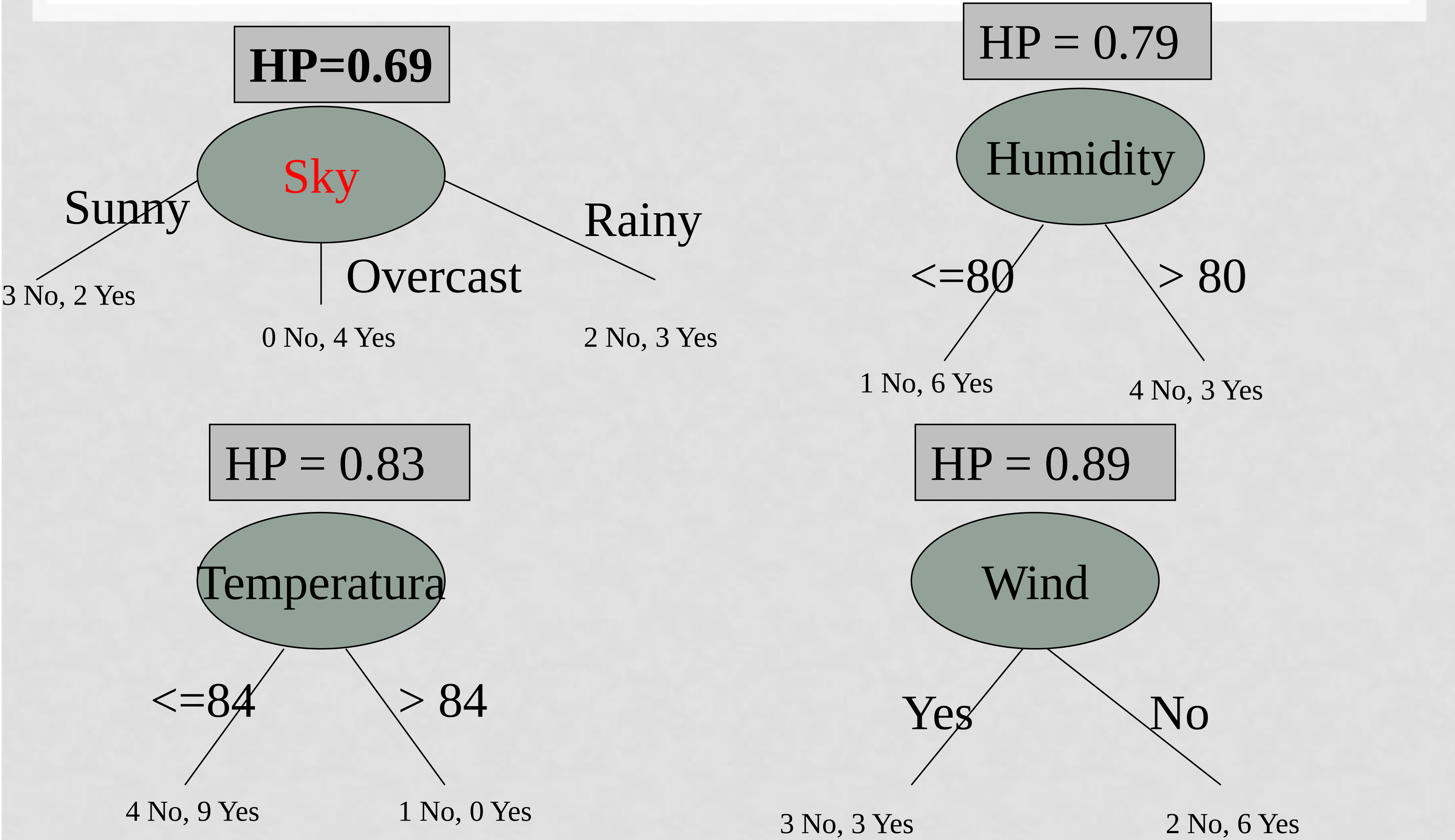
1 No, 6 Yes

4 No, 3 Yes

$\boxed{\text{HP} = 0.79}$

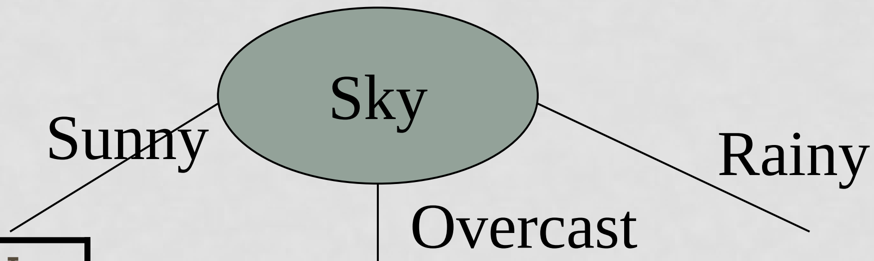
Note: there are other alternatives for the threshold, but this is the best one (minimum entropy)

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

"3 No, 2 Yes"

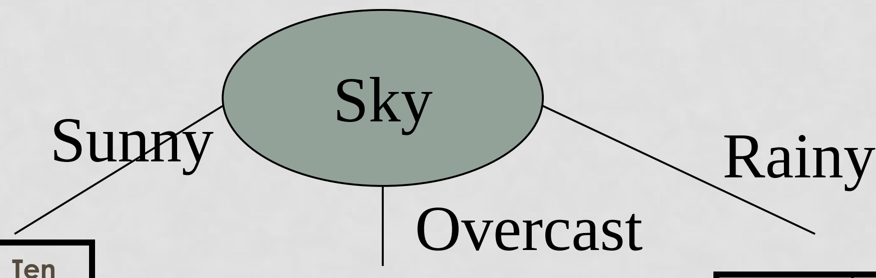
S	T	H	W	Ten
Overcast	83	86	No	Yes
Overcast	64	65	Yes	Yes
Overcast	72	90	Yes	Yes
Overcast	81	75	No	Yes

"0 No, 4 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

"2 No, 3 Yes"

When to stop splitting data?



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
Overcast	83	86	No	Yes
Overcast	64	65	Yes	Yes
Overcast	72	90	Yes	Yes
Overcast	81	75	No	Yes

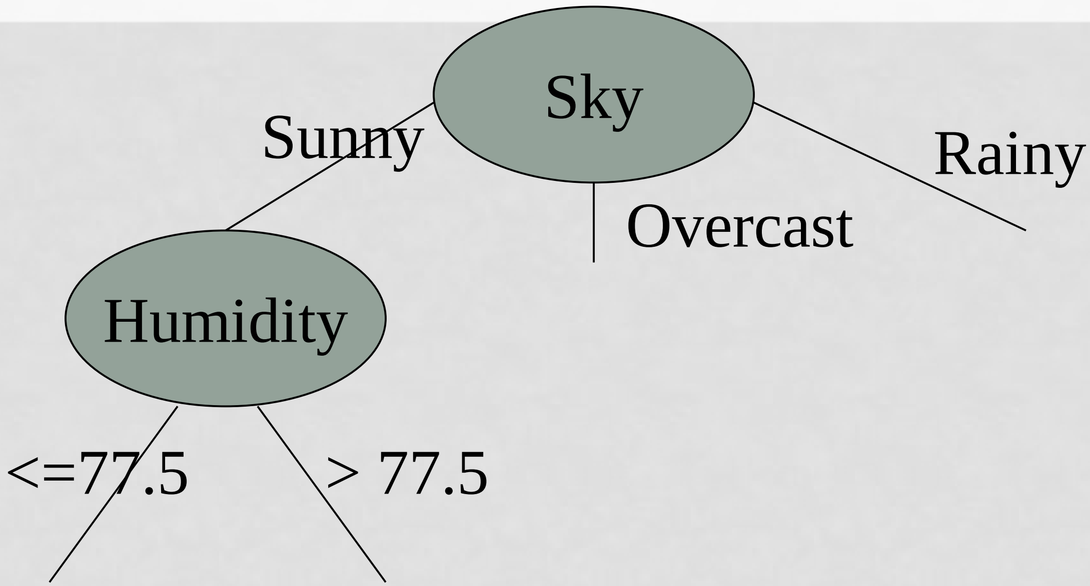
"0 No, **4 Yes**"

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?



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

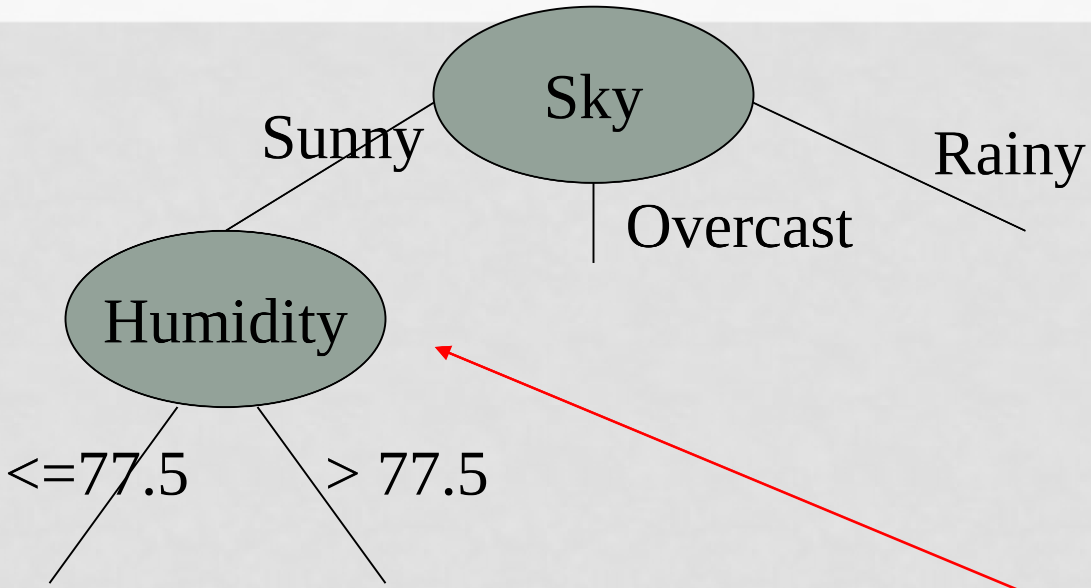
T	H	V	Ten
85	85	No	No
80	90	Yes	No
72	95	No	No

"**2 Yes,** 0 No"

"**3 No,** 0 Yes"

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

Why stop tree growth?



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

“**2 Yes,** 0 No”

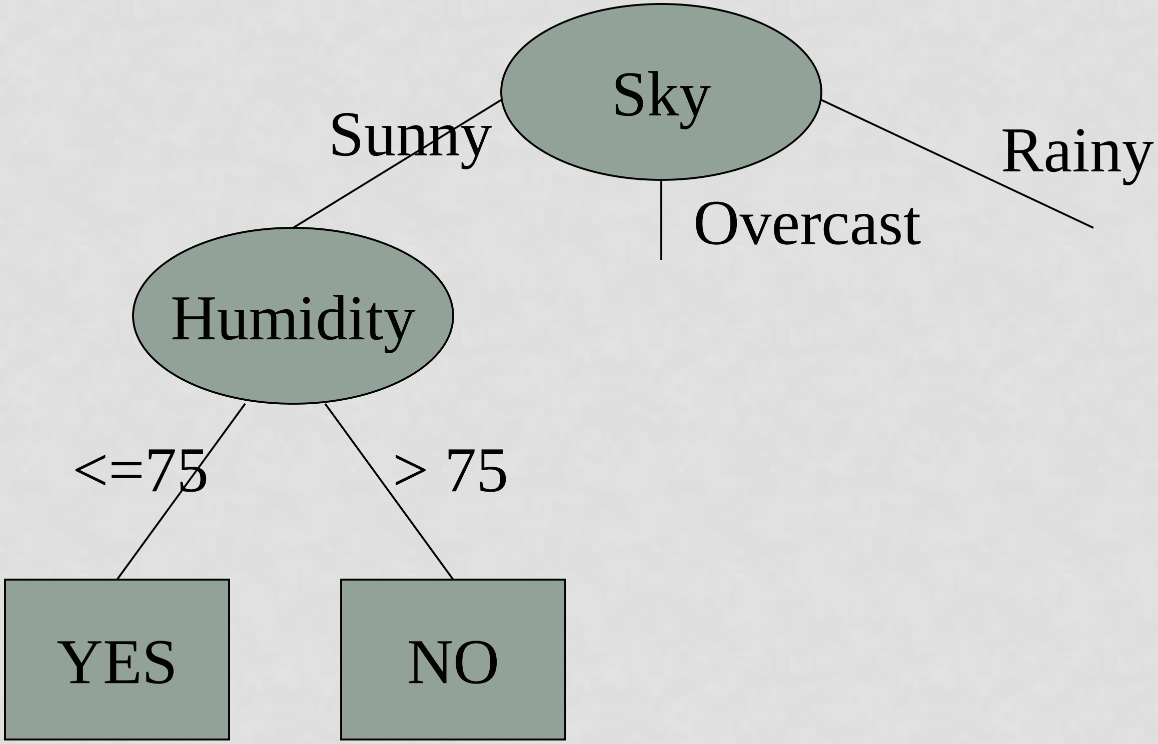
T	H	V	Ten
85	85	No	No
80	90	Yes	No
72	95	No	No

“**3 No,** 0 Yes”

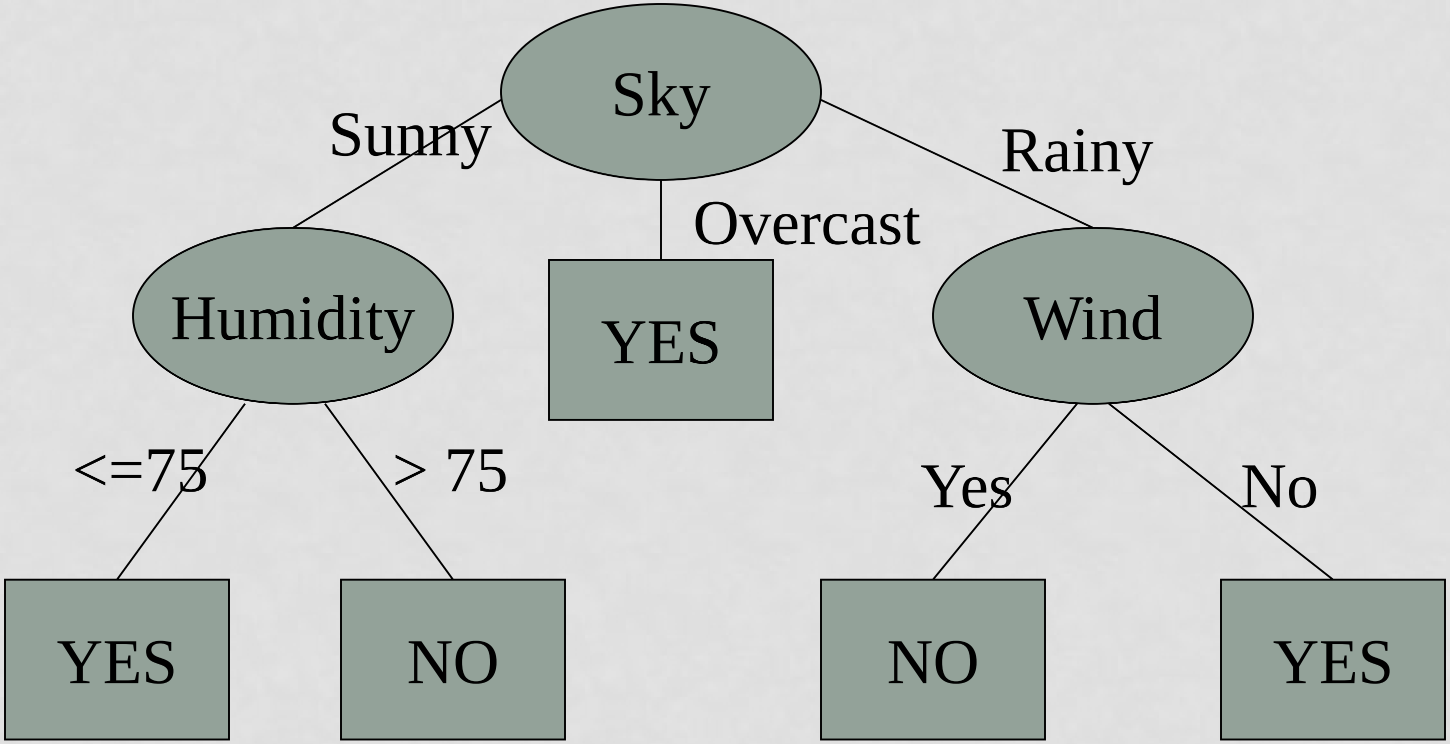
No need to create a new node because all instances belong to the same class

Question: would Sky be a candidate attribute for this node?

Recursive tree growth



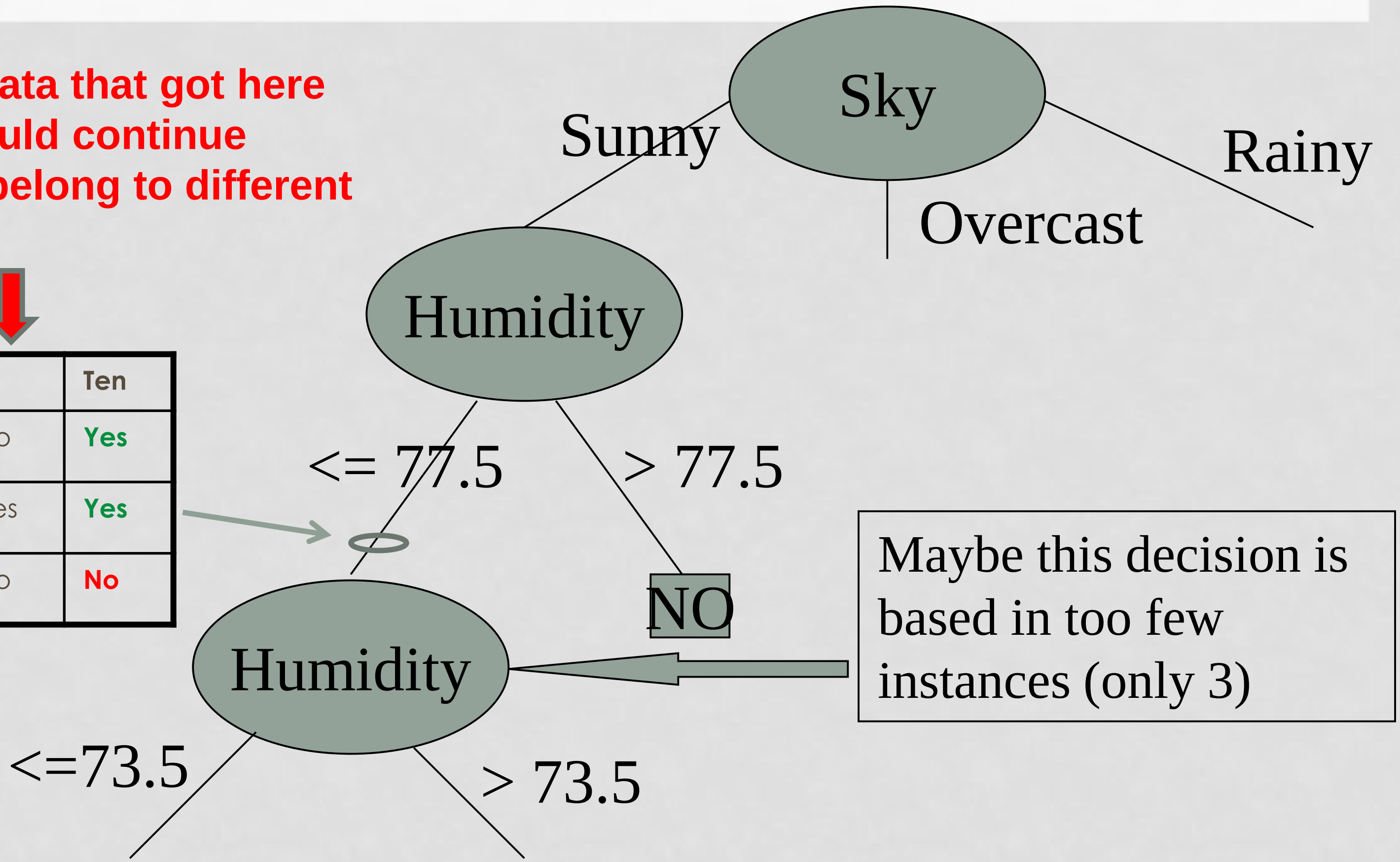
Recursive tree growth



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.

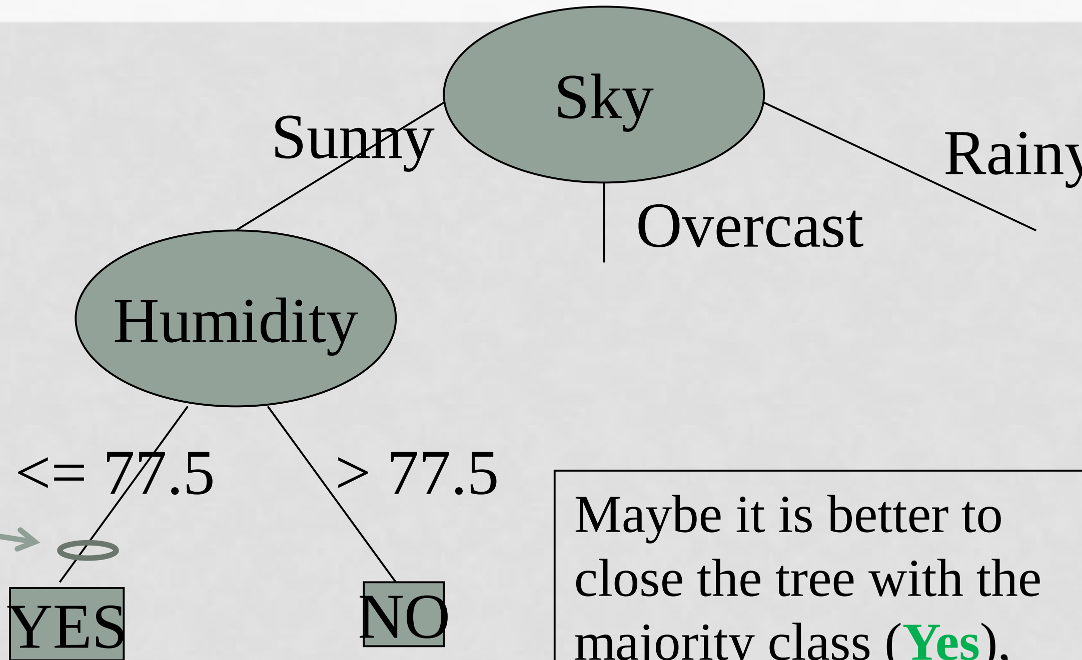
T	H	V	Ten
69	70	No	Yes
75	71	Yes	Yes
72	76	No	No



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?



T	H	V	Ten
69	70	No	Yes
75	71	Yes	Yes
72	76	No	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 may use a statistical criterion in order to determine whether it is worth to continue tree growth, whether the sample is too small, etc.
- Also, this can be controlled with the **min_samples_split** hyperparameter.

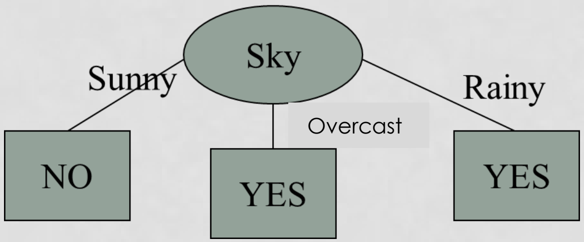
SOME HYPER-PARAMETERS OF DECISION TREES

- **max_depth**: maximum depth of the tree
- **min_samples_split**: minimum number of instances in order to continue subdividing the tree

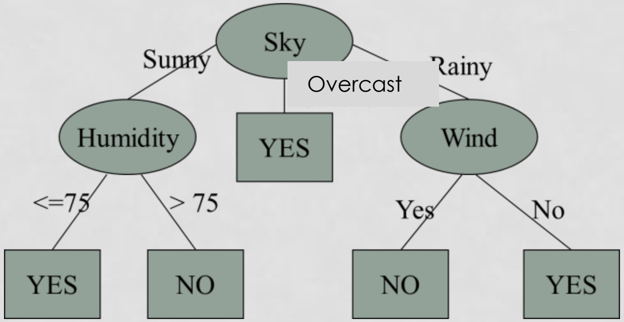
SOME HYPER-PARAMETERS OF DECISION TREES

- **max_depth**: maximum depth of the tree

max_depth = 1



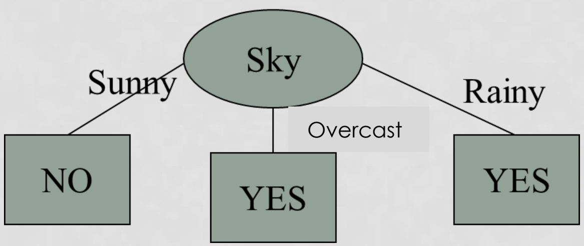
max_depth = 2



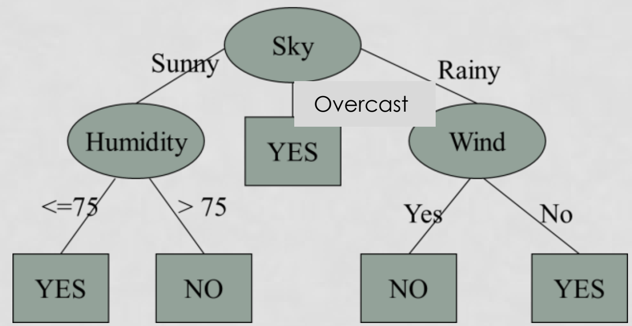
SOME HYPER-PARAMETERS OF DECISION TREES

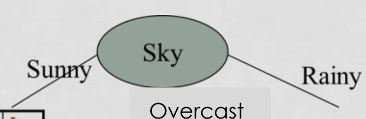
- **min_samples_split**: minimum number of instances in order to continue subdividing the tree

min_samples_split = 6



min_samples_split = 5





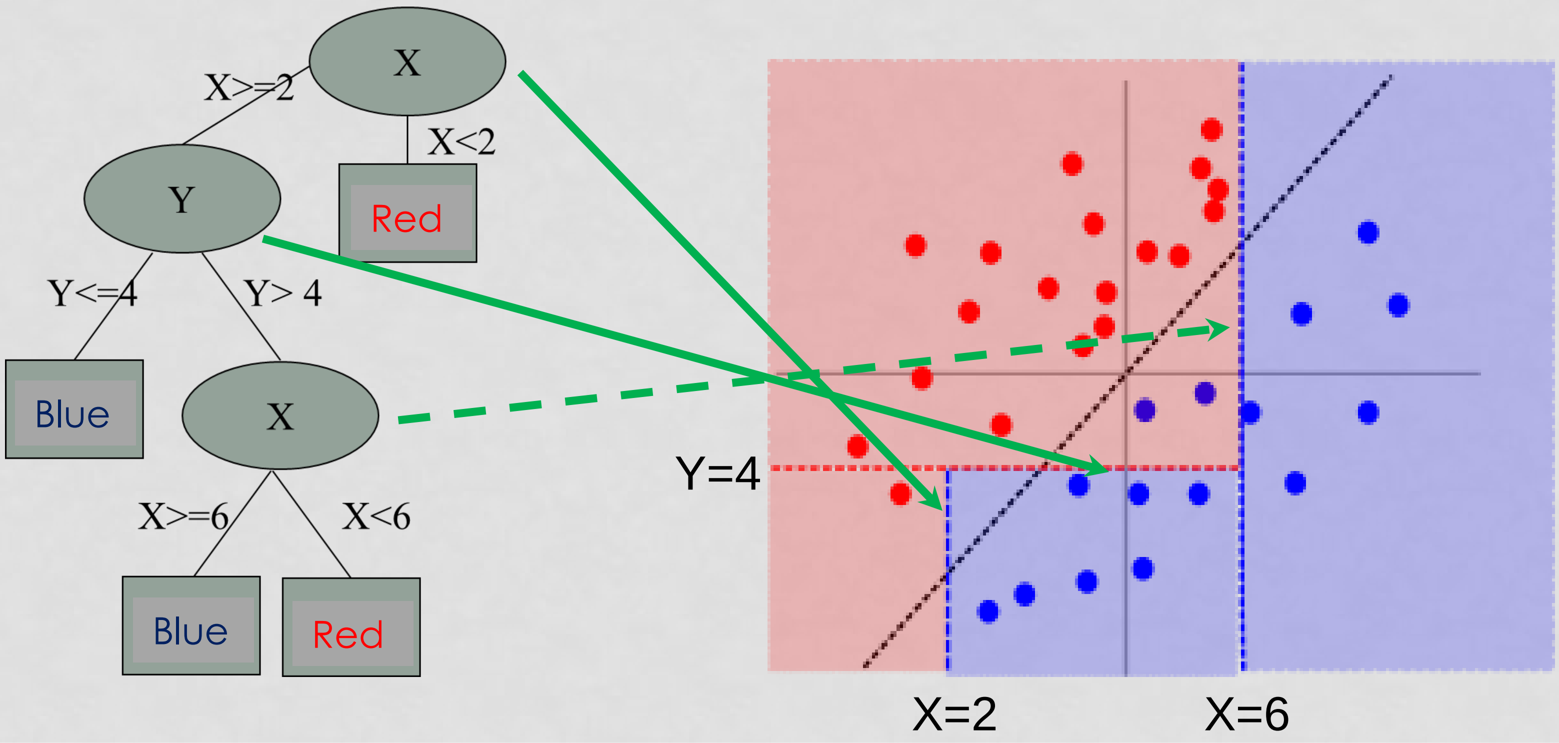
S	T	H	W	Ten
Sun ny	85	85	No	No
Sun ny	80	90	Yes	No
Sun ny	72	95	No	No
Sun ny	69	70	No	Yes
Sun ny	75	70	Yes	Yes

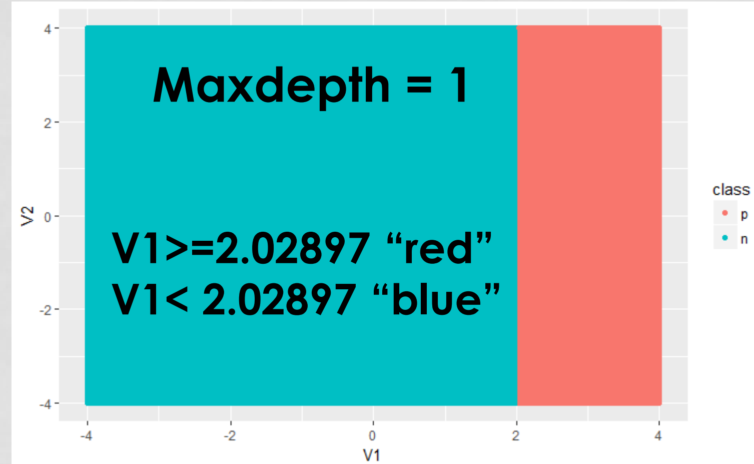
S	T	H	W	Ten
Outcast	83	86	No	Yes
Outcast	64	65	Yes	Yes
Outcast	72	90	Yes	Yes
Outcast	81	75	No	Yes

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

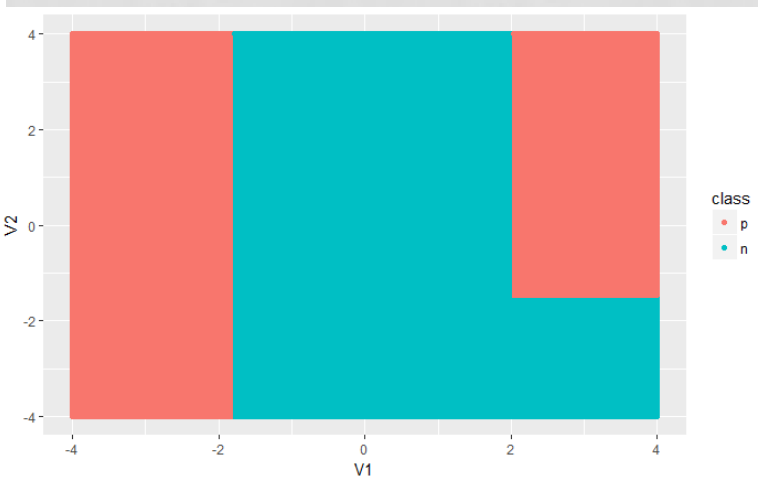
Boundaries in Feature Space

- Decision trees are non-linear. Boundary is made of piece-wise linear segments parallel to axis.
- Therefore, not very good for oblique boundaries (there are "oblique trees", but not widely used)

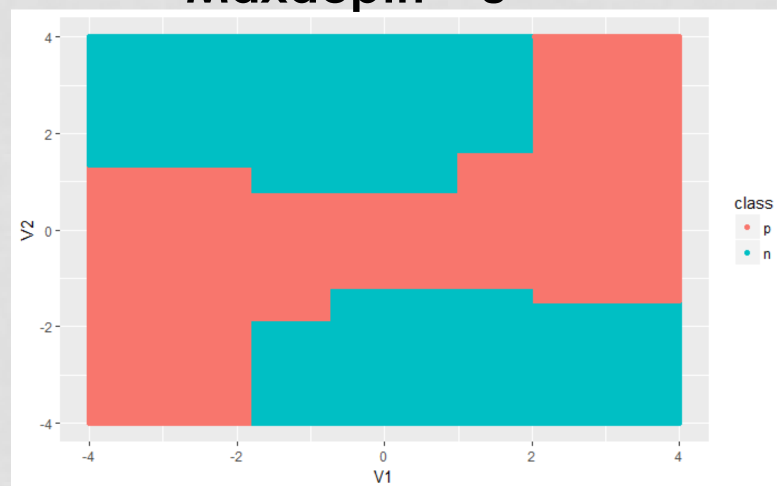




Maxdepth = 2



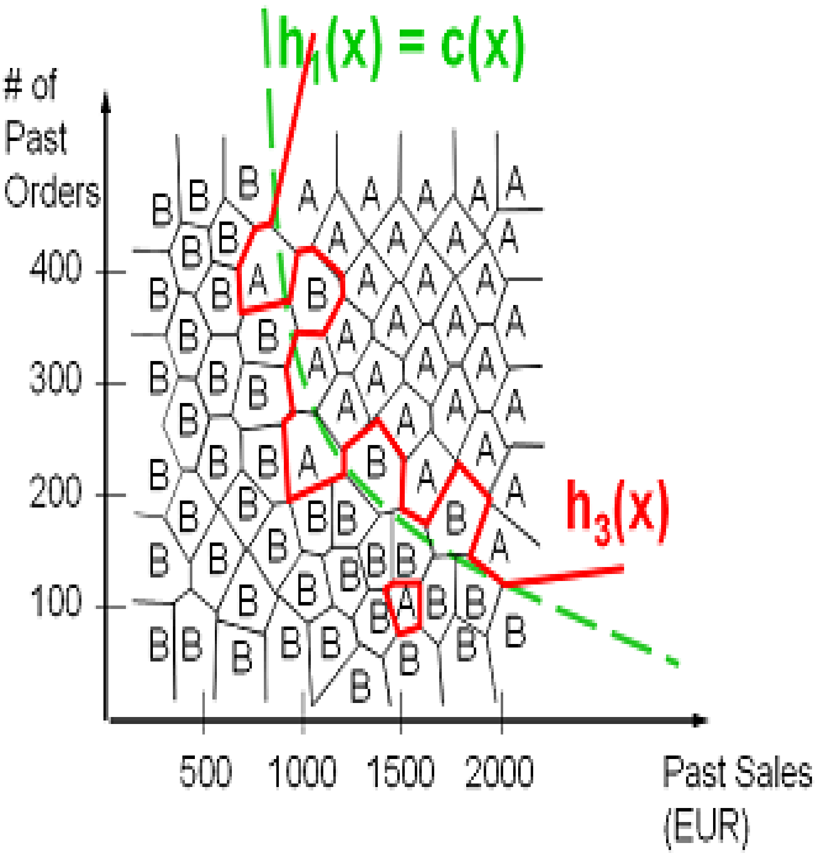
Maxdepth = 8



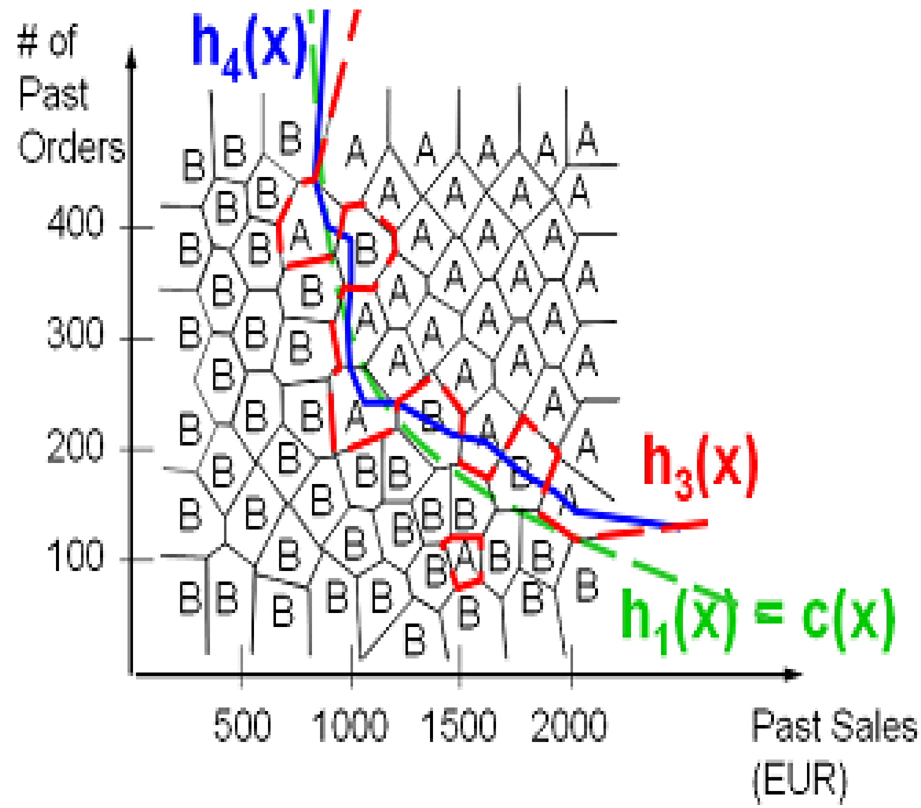
V1>=2.02897 “red”
V2>=-1.474671 “red”
V2< -1.474671 “blue”
V1< 2.02897 “blue”
6) V1< -1.797563 “red”
7) V1>=-1.797563 “blue”

Quite commonly, hyper-parameters control the **complexity** of the model

FOR KNN, LARGE K = SIMPLER MODEL



(a) 1-NN on noisy data



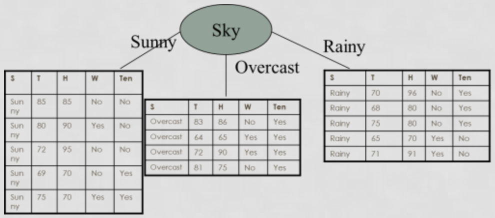
(b) 3-NN and noisy data

ENTROPY AND INFORMATION GAIN

- Sometimes, instead of entropy (H) (to minimize), information gain (IG) is used (to be maximized)
- IG is the difference between the entropy of the original partition H(P) and the weighted average entropy after using the attribute (HP_{Sky}).
- Maximizing IG is equivalent to minimizing entropy.

P

Cielo	Temp	Hum	Viento	Tenis
sol	85	85	no	no
sol	80	90	si	no
nubes	83	86	no	si
lluvia	70	96	no	si
lluvia	68	80	no	si
lluvia	65	70	si	no
nubes	64	65	si	si
sol	72	95	no	no
sol	69	70	no	si
lluvia	75	80	no	si
sol	75	70	si	si
nubes	72	90	si	si
nubes	81	75	no	si
lluvia	71	91	si	no



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
Overcast	83	86	No	Yes
Overcast	64	65	Yes	Yes
Overcast	72	90	Yes	Yes
Overcast	81	75	No	Yes

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

P_1 P_2 P_3

$IG = H(P) - HP_{Sky}(P_1, P_2, P_3)$

ENTROPY AND INFORMATION GAIN

- Choosing the attribute with highest IG is equivalent to choosing the attribute with smallest entropy, because H(P) is the same for all attributes.

$$IG_{Sky} = H(P) - HP_{Sky}(P_1, P_2, P_3)$$

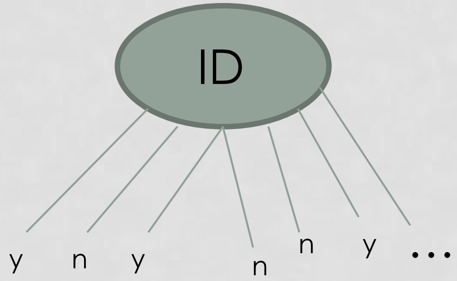
$$IG_{Humidity} = H(P) - HP_{Humidity}(P_1, P_2)$$

$$IG_{Temperature} = H(P) - HP_{Temperature}(P_1, P_2)$$

$$IG_{Wind} = H(P) - HP_{Wind}(P_1, P_2)$$

CATEGORICAL ATTRIBUTES WITH MANY VALUES

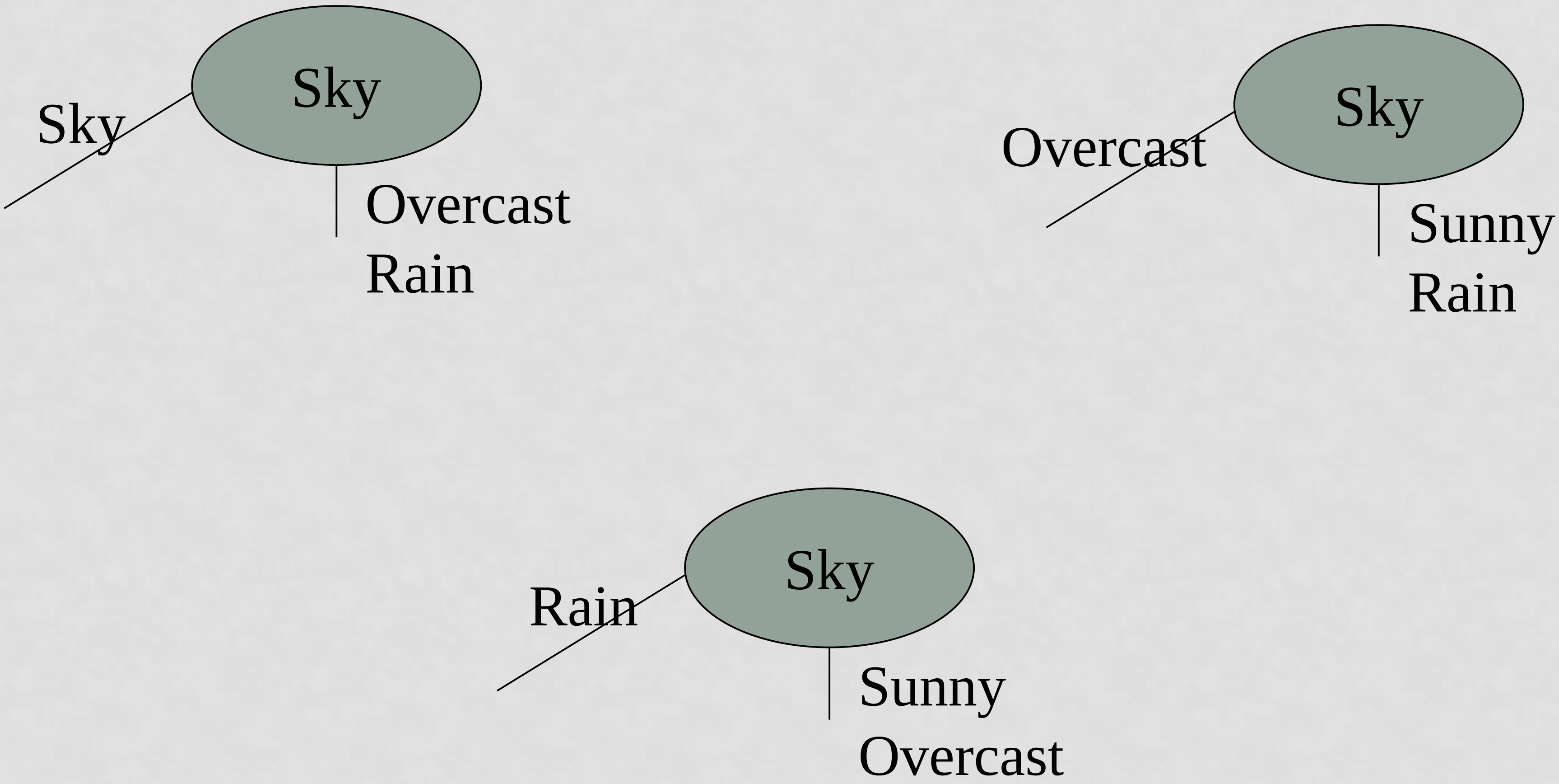
- Some attributes have many values and therefore many branches.
- In an extreme case (e.g. personal identity number ID), each branch is going to contain just one instance. Therefore, each partition is going to have zero entropy, and so ID. Despite ID's not being particularly useful for prediction (e.g. for predicting whether a loan is going to be returned)



- In less extreme cases, this kind of attributes reduces too much the number of instances that go down into each branch.
- A solution is to penalize attributes with lots of different values. A metric called "gain ratio" is typically used:
    - gain_ratio = information_gain / penalization_distinct_values
- Another solution is to use binary nodes.

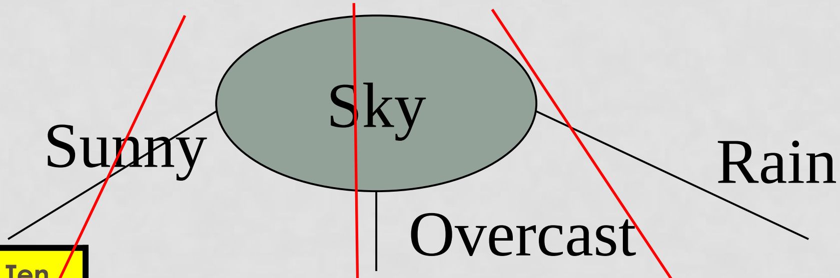
BINARY NODES FOR CATEGORICAL ATTRIBUTES

Sometimes, a trick similar to that of the threshold, can also be used for categorical attributes, so that all nodes are binary (two branches)



HANDLING "MISSING VALUES"

NA	85	85	No	No



C	T	H	V	Ten
Sun	85	85	No	No
Sun	80	90	Yes	No
Sun	72	95	No	No
Sun	69	70	No	Yes
Sun	75	70	Yes	Yes

C	T	H	V	Ten
Overc	83	86	No	Yes
Overc	64	65	Yes	Yes
Overc	72	90	Yes	Yes
Overc	81	75	No	Yes

C	T	H	V	Ten
Rain	70	96	No	Yes
Rain	68	80	No	Yes
Rain	75	80	No	Yes
Rain	65	70	Yes	No
Rain	71	91	Yes	No

3 + 5/14 No

2 Yes

0+4/14 No

4 Yes

2 + 5/14 No

3 Yes

Rules (created from the decision tree)

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

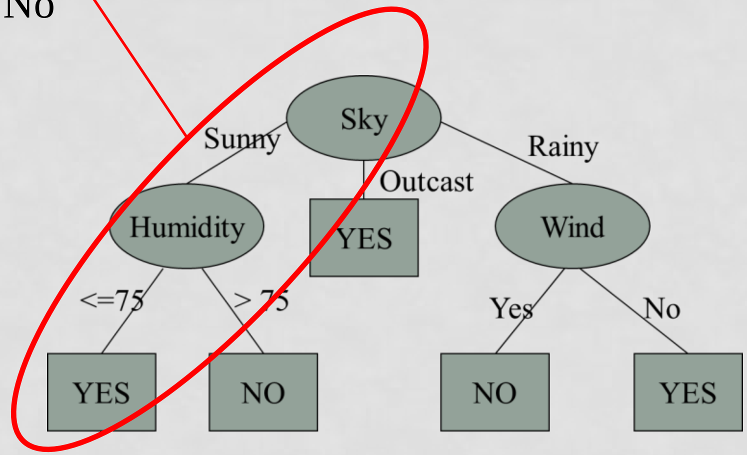
IF Sky = Sunny **AND** Humidity <= 75 **THEN** Tennis = Yes

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

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

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

ELSE Tennis = No

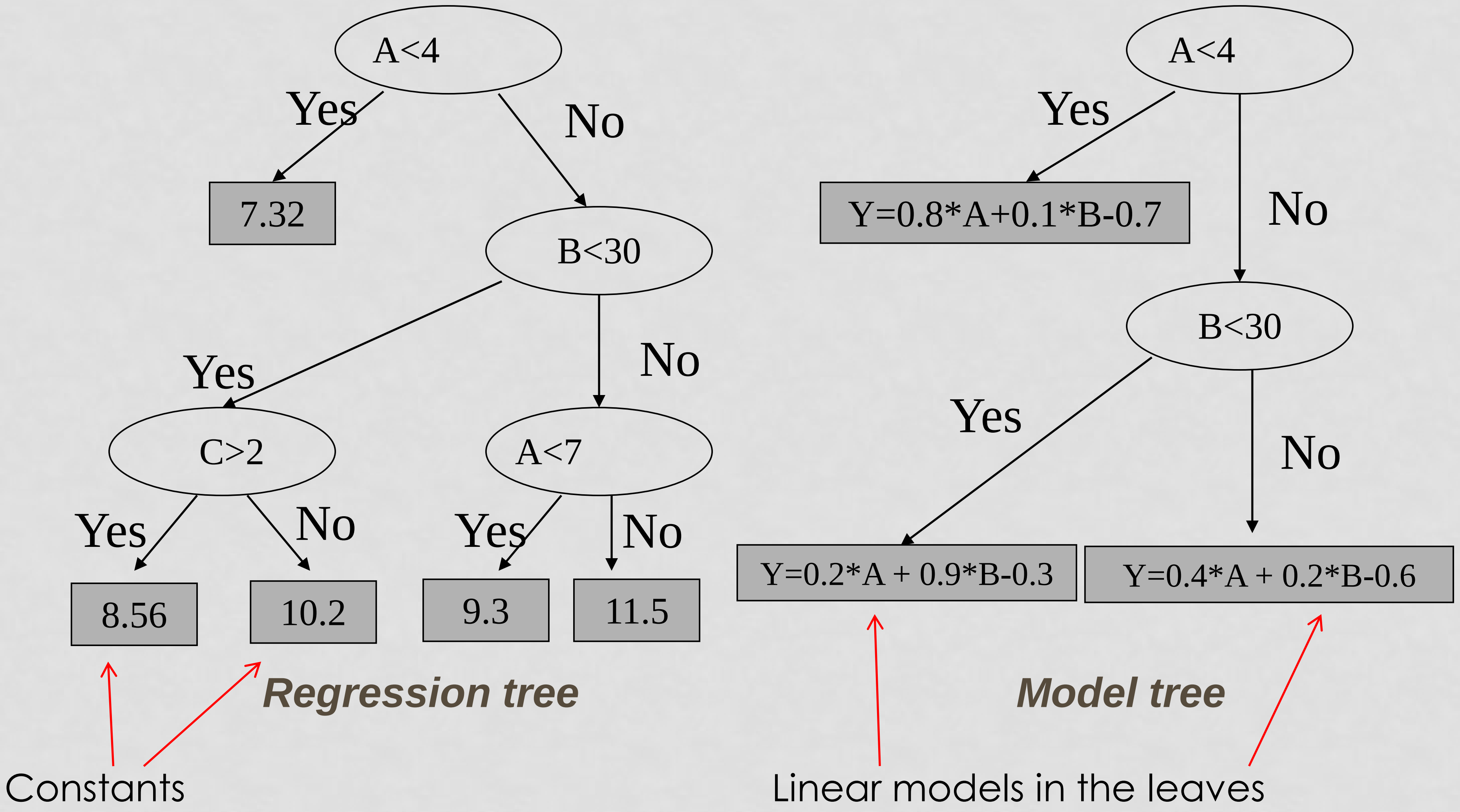


But there are algorithms that build rules directly from data

TREES FOR REGRESSION

- What to do if the response variable is continuous (rather than categorical)?
  - Answer: variance reduction (instead of entropy reduction)
- Two types:
  - Model trees
  - Regression trees

TREES FOR REGRESSION: TWO TYPES



EXAMPLE

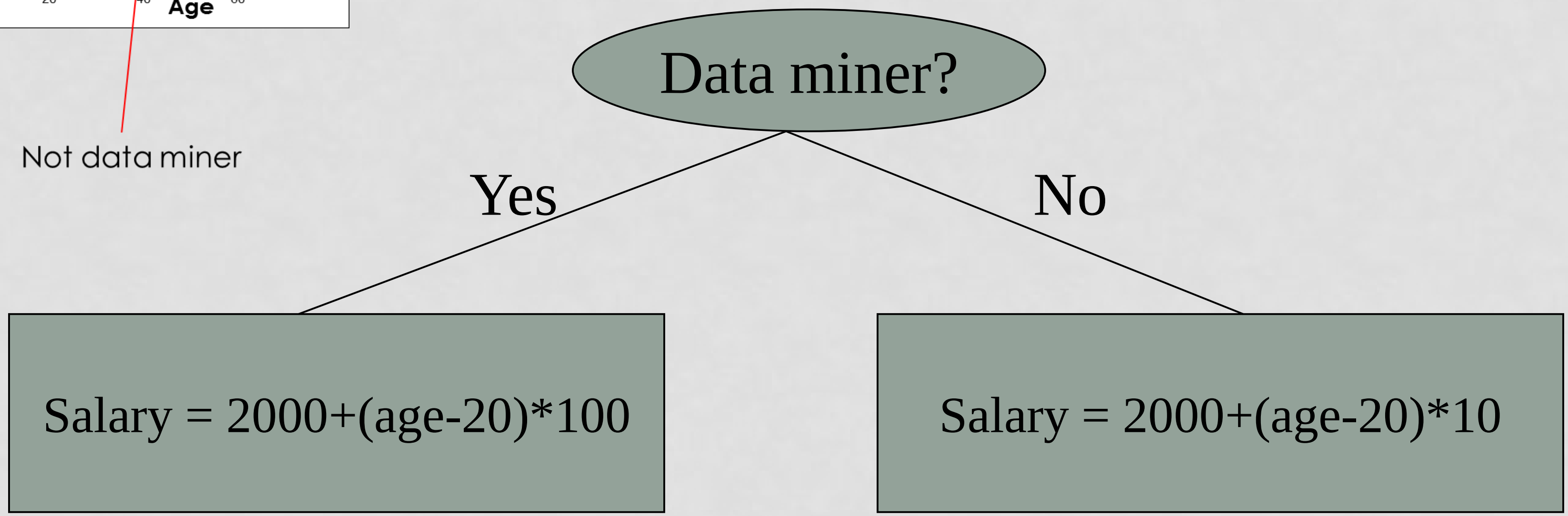
Training data

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

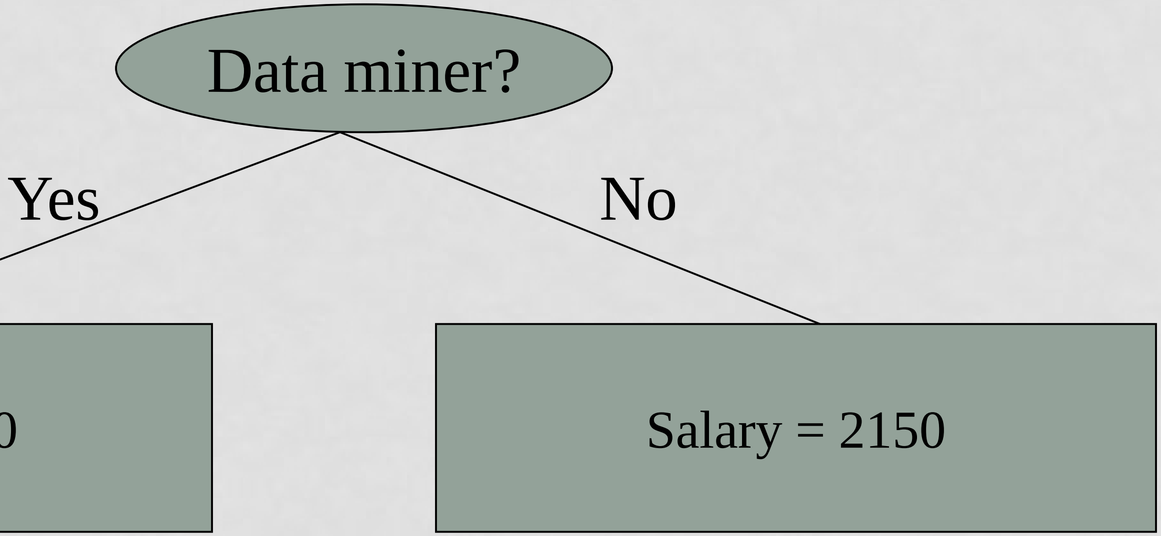
Data miner

Not data miner

MODEL TREES. EXAMPLE

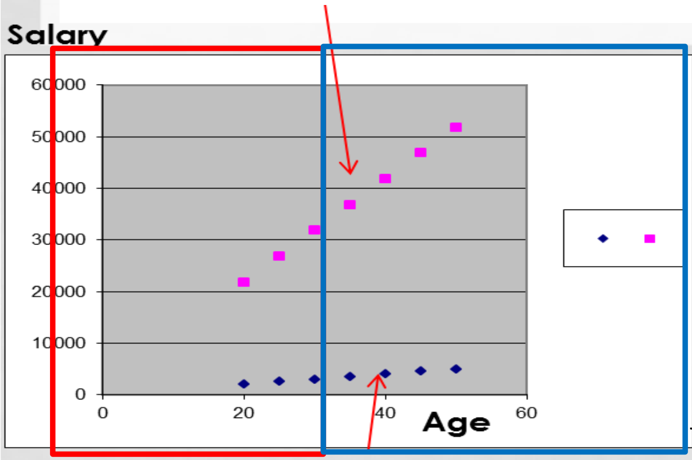


REGRESSION TREES. EXAMPLE

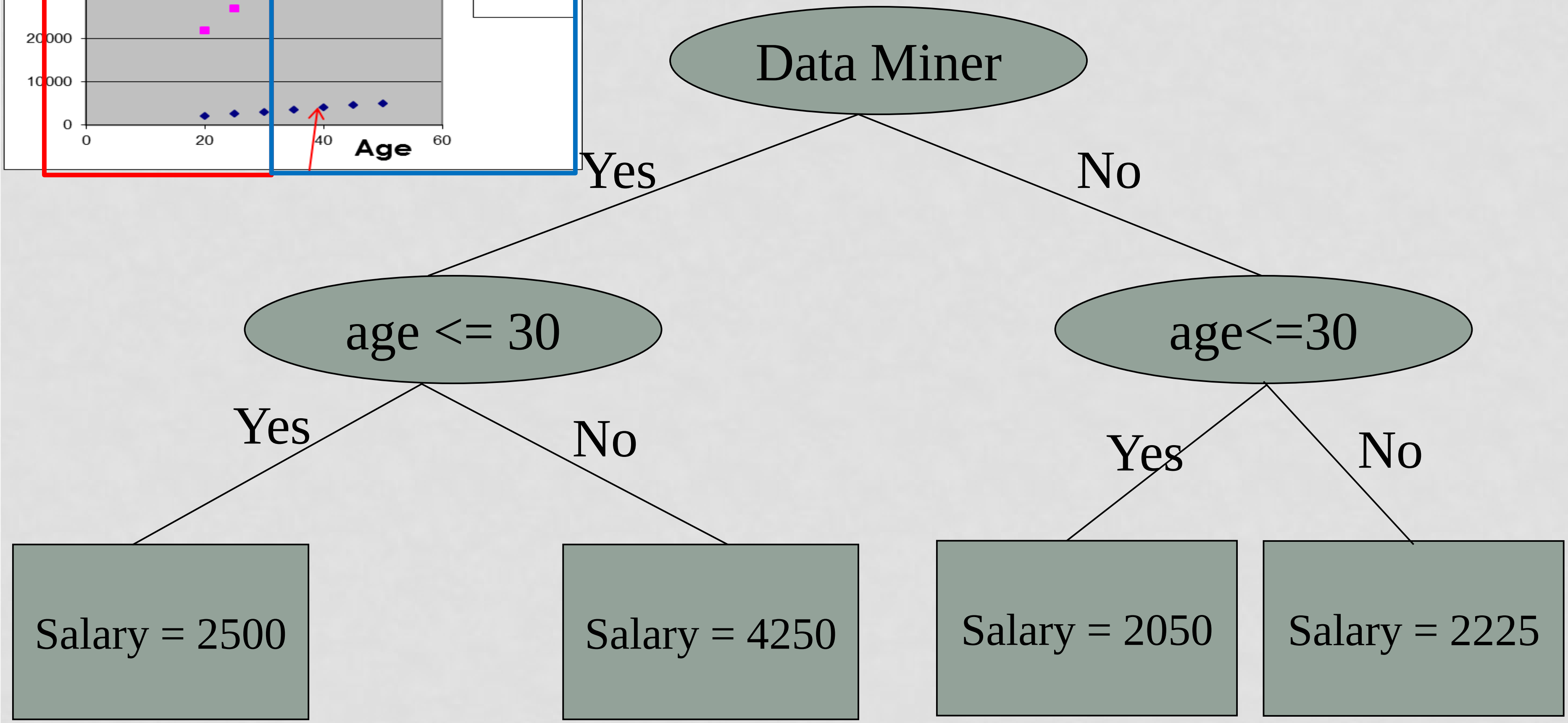


In the leaves, we can see the average salary for data miners (3500 euros) and the average salary for non-data miners (2150 euros)

REGRESSION TREES. EXAMPLE



A larger depth and using *age* allows the regression tree to approximate the problema, piece-wise.



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))

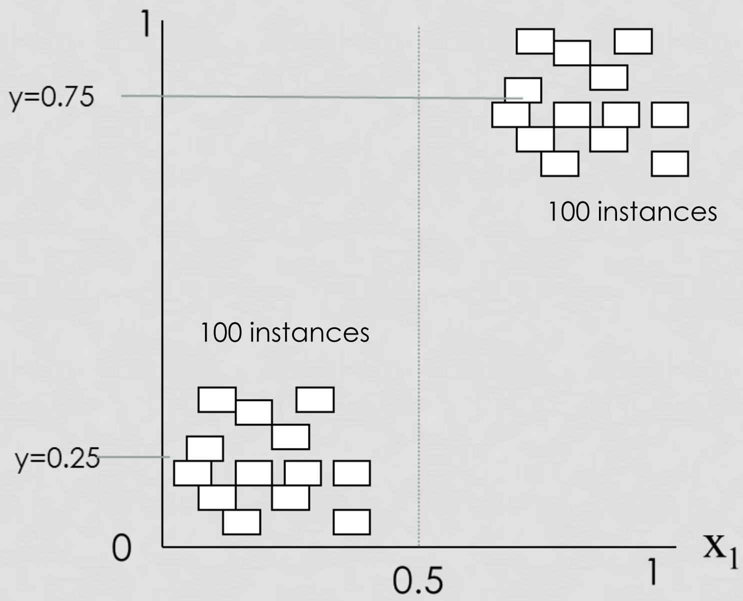
- Trees for regression are built similarly than trees for classification (decision trees), except that **standard deviation / variance** is reduced (instead of entropy)

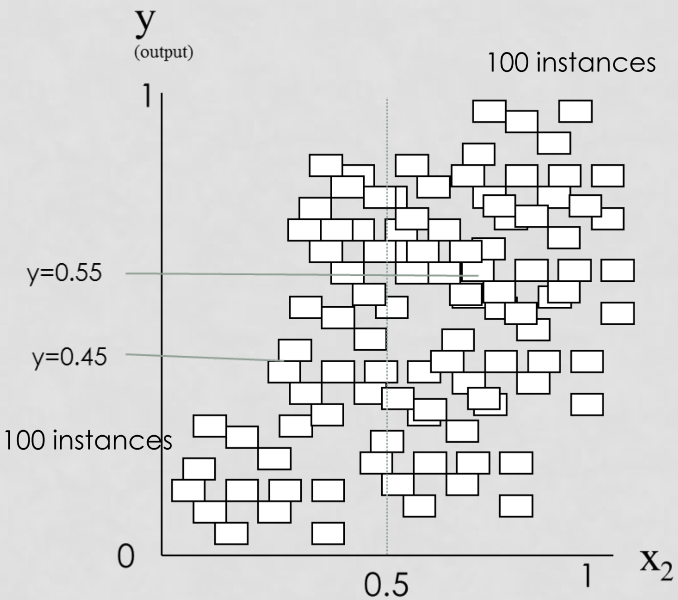
- The tree is built recursively until a stopping condition is reached (max_depth, minsplit, …)

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

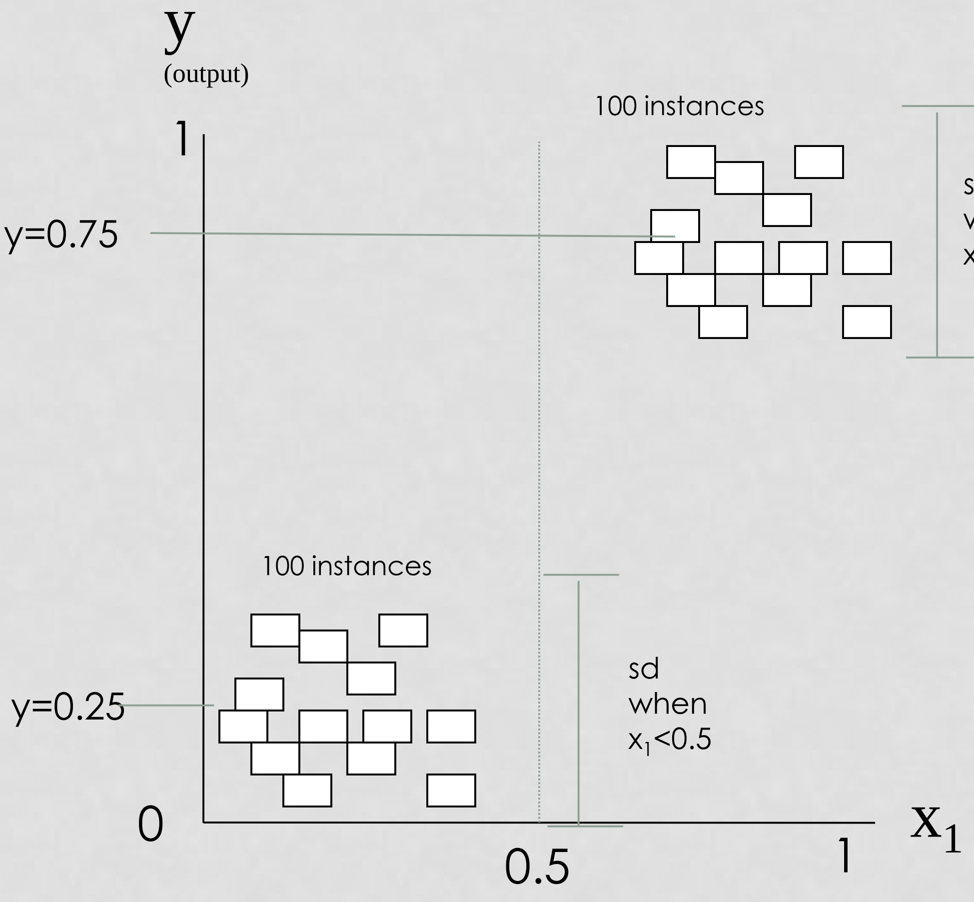
Which attribute is better?
x_1 or x_2?
We will choose the attribute for which the average standard deviation (sd) **after the partition** is small:

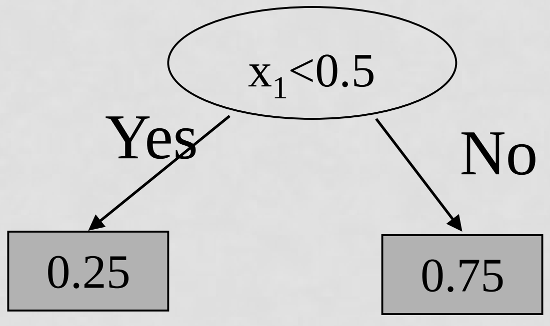
$$100/200 * sd(x_i<0.5) + 100/200 * sd(x_i>0.5)$$





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



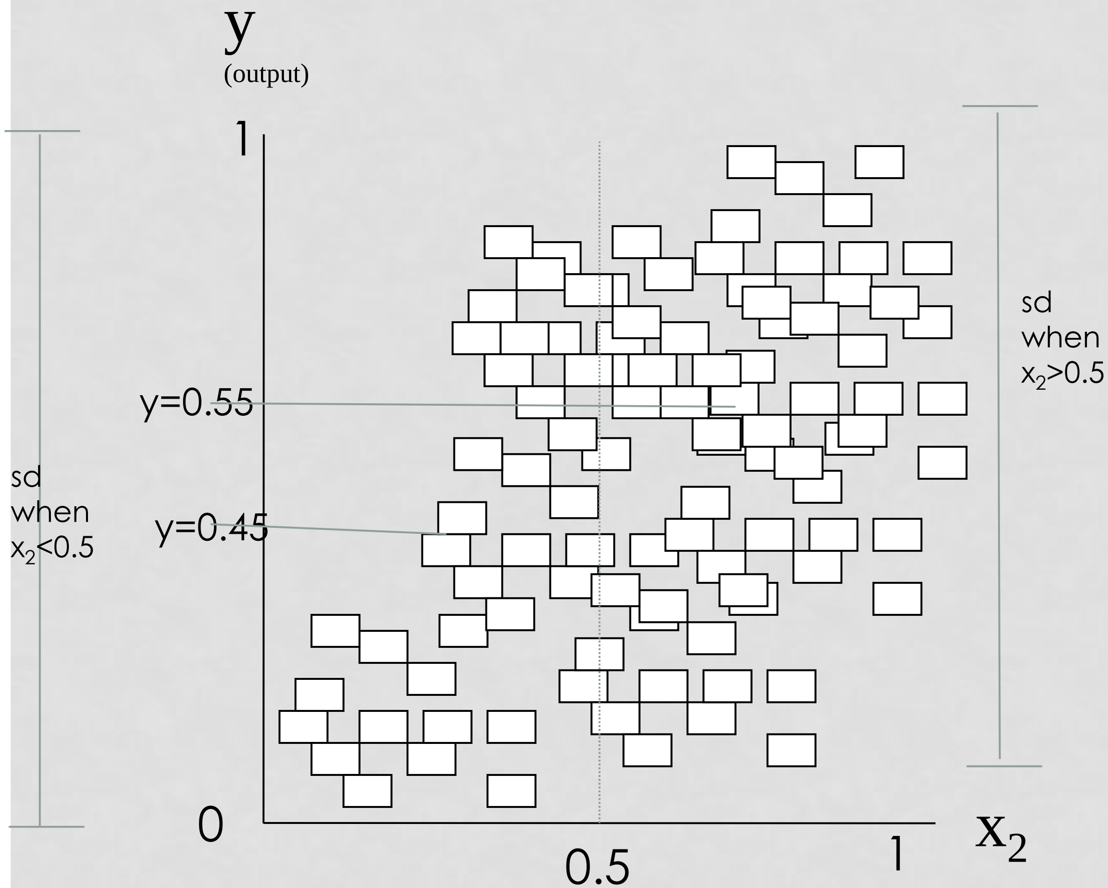


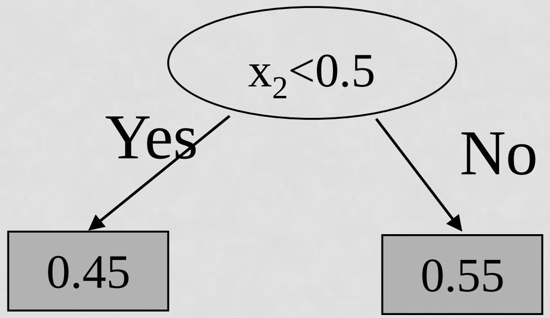
It can be noticed that x_1 is quite predictive

Instances after the partition are not spread

½*sd(x_1<0.5) + ½* sd(x_1>0.5) is small

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





It can be noticed that x_2 is not predictive

Instances after the partition are very spread

$\frac{1}{2} * sd(x_2<0.5) + \frac{1}{2} * sd(x_2>0.5)$ is very large

ACTUAL METHODS

- Classification and regression trees: C4.5, C5.0, CART
- Model trees: M5P, Cubist

ADVANTAGES OF TREES

- Interpretable.
  - Attributes closer to the root are the most relevant.
- They can handle naturally categorical attributes (no need for dummy variables).
- Training is fast (compared to other methods)
- They can handle multiple classes naturally.
- They can handle missing values.

DISADVANTAGES OF TREES

- Boundaries too simple for some problems (parallel to axes)
  - However, trees are the elements of advanced methods, such as Random Forests and Gradient Tree Boosting (ensembles).
- Hyper-parameters must be tuned carefully (maxdepth, ..), otherwise (deep) trees are prone to overfitting (bad generalization).