Data Science – Fall 2025/2026

INTRODUCTION TO Machine Learning

Lecture 3
Decision Tree

Topics

- Introduction
- Formal Definitions
- Construction of the Decision Tree
- Application on Weather DATASET

:

- Quinlan's Method
- Training & Test Datasets
- Cross-Vaidation Technique
- Regression Tree
- Validity

Introduction

Decision Tree is :

- Supervised classification/regression algorithm
- Non-parametric statistical method
- Allows classifying a set of individuals described by qualitative and quantitative variables.
- Produces the most homogeneous classes possible
- Classifications are easy to undesrtand for the user

Introduction

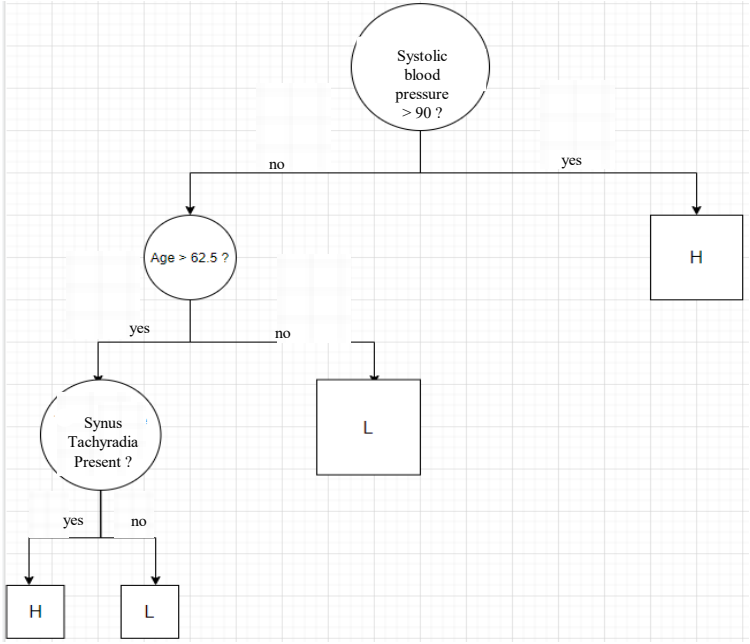
Most popular application areas:

- Banking sector: Identifying a client's credit risk based on their probability of default.
- Medicine: Identifying at-risk patients and disease trends.
- Marketing: Identifying customer churn rates.
- Challenges in environmental, economic, financial, and other areas of decision-making

Introduction

- In a hospital, for each new patient experiencing a heart attack, 15 variables are measured during the first 18 hours. These variables include blood pressure, age, and 13 other characteristics summarizing the various symptoms

- The objective of the study is to identify high-risk patients



Introduction

Representation

- The root: χ
- A node: a subset of χ (represented by a circle).
- Terminal nodes: subsets that are no longer divided (represented by boxes).
- Each terminal node is labeled with a target class, which is one of the values y of a target attribute Y.

Construction of the Decision Tree

- Obtained by repeatedly splitting χ into subsets.
- The variable that best separates the data is selected.
- This process is repeated for each subgroup.
- It stops when the subgroups reach the minimum size or when no further improvement is possible.

Introduction

Thus, constructing a decision tree requires:

- Selecting the splits.
- Deciding whether to declare a node as terminal or to continue splitting it.
- Assigning a class to each terminal node.

Notations

- n: sample size (number of observations).
- K: number of classes of the target variable Y
- $N(t)$: number of observations in node t.
- $N_k(t)$: number of observations of class $k \in \{1, 2, \ldots, K\}$ in node t.
- $p\ (k|t)$: proportion of observations in node t that belong to class $k \in \{1, 2, \ldots, K\}$.

$$p\ (k|t) = \frac{N_k\ (t)}{N\ (t)}$$

$p\ (t)$= Vector of proportions corresponding to node t

$$p\ (t) = [\, p(1|t), p\ (2|t)\ , \ldots , p\ (k|t)\ ]$$

Preliminary Definitions

Definition 1.
A measure of impurity of a node t in a decision tree with a target variable Y having K classes is a function of the form:

$$Imp\,(t) = \varphi\,(p\,(t)),$$

where φ is a non-negative function of $p\,(t)$, that satisfies the following conditions:

- φ reaches its unique maximum at $p\,(t) = (1/K, 1/K, \ldots, 1/K)$
- φ reaches its minimum at $[1, 0, \ldots, 0], [0, 1, \ldots, 0], \ldots, [0, 0, \ldots, 1]$
- φ is a symmetric function of $p\,(1|t), p\,(2|t), \ldots, p\,(k|t)$.

Remark: $Imp(t)$ is maximal when all classes are mixed in equal proportions (uniform/equiprobable distribution) and minimal when the node contains only a single class (total certainty)

Preliminary Definitions

Examples of Impurity measures: $Imp\,(t) = \varphi\,(p\,(t))$,

- **Entropy**

$$imp(t) = \mathcal{H}(t) = -\sum_{k=1}^{k=K} log_2(p(k|t)p(k|t)) \,,$$
with $0log_2(0) = 0$

- **Gini Impurity**

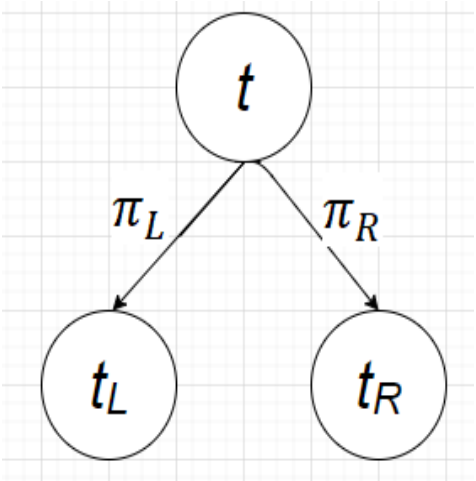
$$imp(t) = \varphi(t) = 1 \; - \sum_{k=1}^{k=K} p^2(k|t),$$

Remark: A node is pure if it contains data from only one class. In this case:

$$\mathcal{H}(t) = \varphi(t) = 0$$

To Split or not to Split

We consider a node t with two child nodes and t_R and t_L. This splitting operation is denoted by S,



With:

- π_L: Proportion of observations from t that go to t_L
- π_R: Proportion of observations from t that go to t_R

To Split or not to Split

The quality of a split S for a node t, denoted $\Phi(S, t)$, is defined by the variation in the impurity measure:

$$\Phi(S, t) = imp(t) - \pi_L\, imp(t_L) - \pi_R\, imp(t_R)$$

The idea is to choose a split S that mximizes $\Phi(S, t)$.

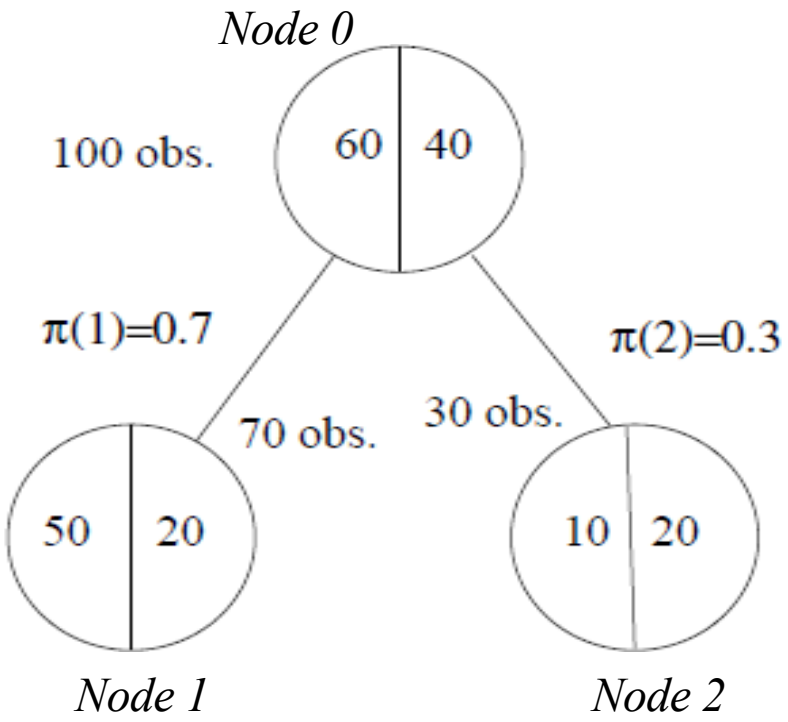
Definition 2.

The global impurity of a decision tree (denoted T), is given by:

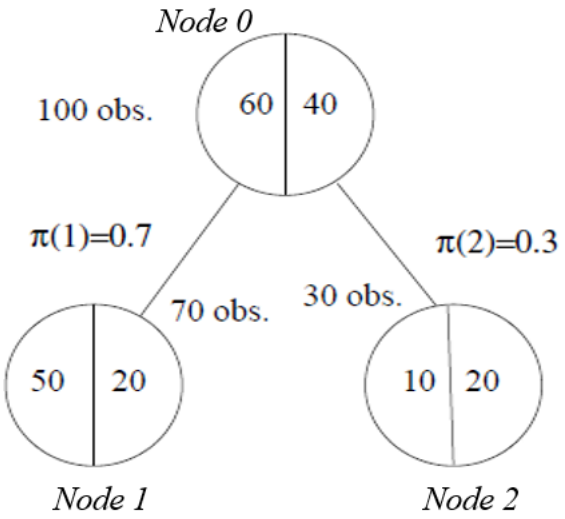
$$imp(T) = \sum_{t \in \widehat{T}} \pi_t imp(t),$$

where $\widehat{T}$ is the set of terminal nodes of T and π_t is the proportion of the global population within node t

Numerical Example



Numerical Example



$$Imp(t_0) = -\frac{60}{100}\log_2\left(\frac{60}{100}\right) - \frac{40}{100}\log\left(\frac{40}{100}\right) = 0.971,$$
$$Imp(t_1) = -\frac{50}{70}\log_2\left(\frac{50}{70}\right) - \frac{20}{70}\log\left(\frac{20}{70}\right) = 0.863,$$
$$Imp(t_2) = -\frac{10}{30}\log_2\left(\frac{10}{30}\right) - \frac{20}{30}\log\left(\frac{20}{30}\right) = 0.918,$$
$$\Delta Imp(t_0) = Imp(t_0) - 0.7\,Imp(t_1) - 0.3\,Imp(t_2) = 0.091.$$

Rules of Split

Let $X = [X_1, X_2, \ldots, X_p]$ be the vector of explanatory variables present in a given context. The splits of the nodes in a decision tree must satisfy the following conditions:

- Each split depends on only one variable
- For quantitative X_i ,the splitting criterion is of the form:
  Is $X_i \leq c$, with $c \in \mathbb{R}$
- If X_i is categorical with values in $B = \{b_1, b_2, \ldots, b_m\}$,the splitting criterion is of the form: Is $X_i \in A$, with $A \subseteq B$
- At each node, the variables X_i are considered one by one in order to:

  1) Find the best split for each X_i
  2) Choose the best variable in terms of split quality

Stopping and Assignement Rule

Stop splitting if:

- The change in the node's impurity measure is below a certain threshold.
- The tree depth exceeds a predefined value.
- The number of observations is below a predefined value.
- Alternatively, pruning can be used instead of stopping rules.

Assign the class defined as follows:

$$Y(t) = \arg \max_{k=1,\ldots,K} p(k|t),$$

to a terminal node t.

Stopping and Assignement Rule

- The previous construction approach provides the maximal tree T_{max}.

- T_{max} can lead to a very unstable predictive model because it is highly dependent on the samples used to build the tree.

- This is a case of **overfitting**.

- The solution is a tree pruning procedure

Pruning and Complexity Cost

Definition 3.

The classification error rate of a terminal node t, is denoted $R(t)$ and is calculated as follows:

$$R(t) = \sum_{k=1, k \neq Y(t)}^{K} p(k|t)$$

where,

$$Y(t) = \arg \max_{k=1,\ldots,K} p(k|t)$$

***Y*(*t*) is the class attributed to the node *t*.**

Pruning and Complexity Cost

Definition 4.

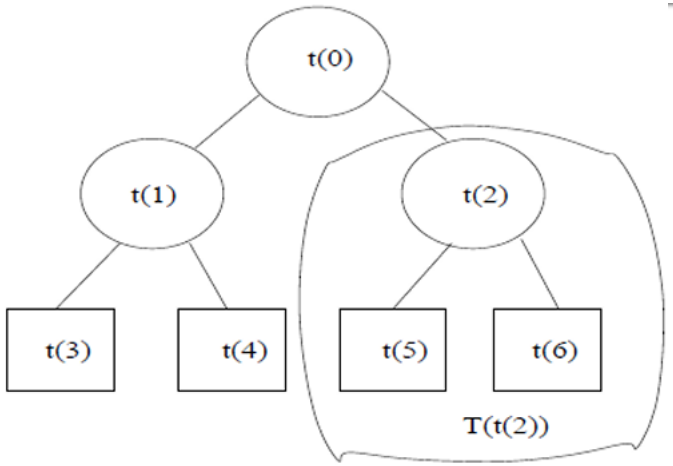
Let $\widehat{T} = \{t_1, t_2, .., t_m\}$ be the terminal nodes of a decision tree *T*. The error rate of classification of *T* is:

$$R(T) = \sum_{i=1}^{i=m} \frac{N(t_i)}{n} R(t_i)$$

Additionnaly, we define *size(T)* = *card*($\widehat{T}$), and $\alpha > 0$ a complexity parameter to imopose a certain penalty on large trees. The penalized error rate of *T* is:

$$R_\alpha(T) = R(T) + \alpha\, size(T)$$

Pruning and Complexity Cost



We denote *T(t),* as the subtree of *T* rooted in node *t.*

The error of subtree $T(t_2)$ and the error of the node t_2, the root of $T(t_2)$, are given, respectively by :

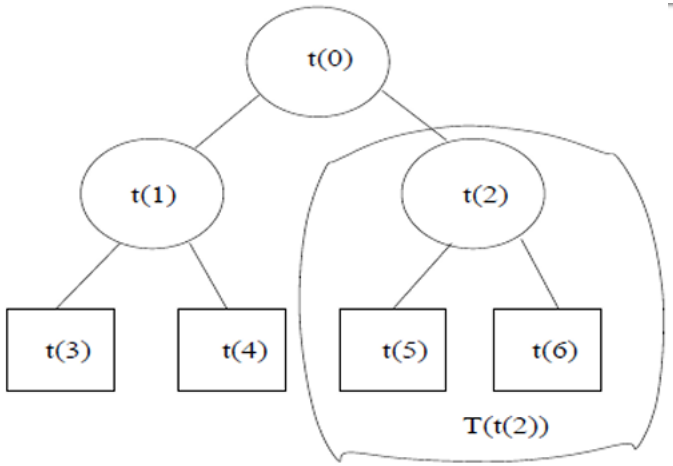
$$R_\alpha(T(t_2)) = R(T(t_2)) + \alpha\, size\big(T(t_2)\big)$$

$$= \frac{1}{n'}\left[N(t_5)\, R(t_5) + N(t_6)\, R(t_6)\right] + 2\alpha,$$

where n' is the *size* of the sample of the susbtree rooted in t_2

$$R_\alpha(t_2) = R(t_2) + \alpha = \sum_{k=1,k\neq Y(t_2)}^{K} p(k|t_2) + \alpha$$

Pruning and Complexity Cost



The pruning is efficient if:

$$R_\alpha(t_2) \leq R_\alpha\big(T(t_2)\big) \Leftrightarrow g(t_2, T) = \frac{R(T(t_2) - R(t_2))}{size(T(t_2) - 1} \leq \alpha$$

The function $g(t, T)$ can be calculated for each node in the tree

Application1 : Weather Dataset

The table consists of 14 observations. The goal is to explain the behavior (whether to play a game or not) of 14 individual, based on weather forecasts.

Number	Sunshine	Temperature	Humidity	Wind	Play
1	Sunny	Mild	High	No	No
2	Sunny	Mild	High	Yes	No
3	Overcast	Mild	High	No	Yes
4	Rain	Warm	High	No	Yes
5	Rain	Cool	Normal	No	Yes
6	Rain	Cool	Normal	Yes	No
7	Overcast	Cool	Normal	Yes	Yes
8	Sunny	Warm	High	No	No
9	Sunny	Cool	Normal	No	Yes
10	Rain	Warm	Normal	No	Yes
11	Sunny	Warm	Normal	Yes	Yes
12	Overcast	Warm	High	Yes	Yes
13	Overcast	Mild	Normal	No	Yes
14	Rain	Warm	High	Yes	No

Application1 : Weather Dataset

- We start by calculating the information gains for each feature (explicative variable)

Variable	*Impurity Variation*
Sunshine	**0.247**
Temperature	0.029
Humidity	0.152
Wind	0.048

- So, the root of the decision tree is the variable *Sunshine*

- Now, the attribute *Sunshine* can take 3 values. We repeat the calculation from the previous step for each of the different values

Number	Sunshine	Temperature	Humidity	Wind	Play
1	Sunny	Hot	High	No	No
2	Sunny	Hot	High	Yes	No
8	Sunny	Warm	High	No	No
9	Sunny	Cool	Normal	No	Yes
11	Sunny	Warm	Normal	Yes	Yes

Application1 : Weather Dataset

- Information gains of the value ‘Sunny’

Variable	*Impurity Variation*
Temperature	0.571
Humidity	**0.971**
Wind	0.02

- For the value ‘Overcast’ we have a pure node.

Number	Sunshine	Temperature	Humidity	Wind	Play
3	Overcast	Hot	High	No	Yes
7	Overcast	Cool	Normal	Yes	Yes
12	Overcast	Warm	High	Yes	Yes
13	Overcast	Hot	Normal	No	Yes

Application1 : Weather Dataset

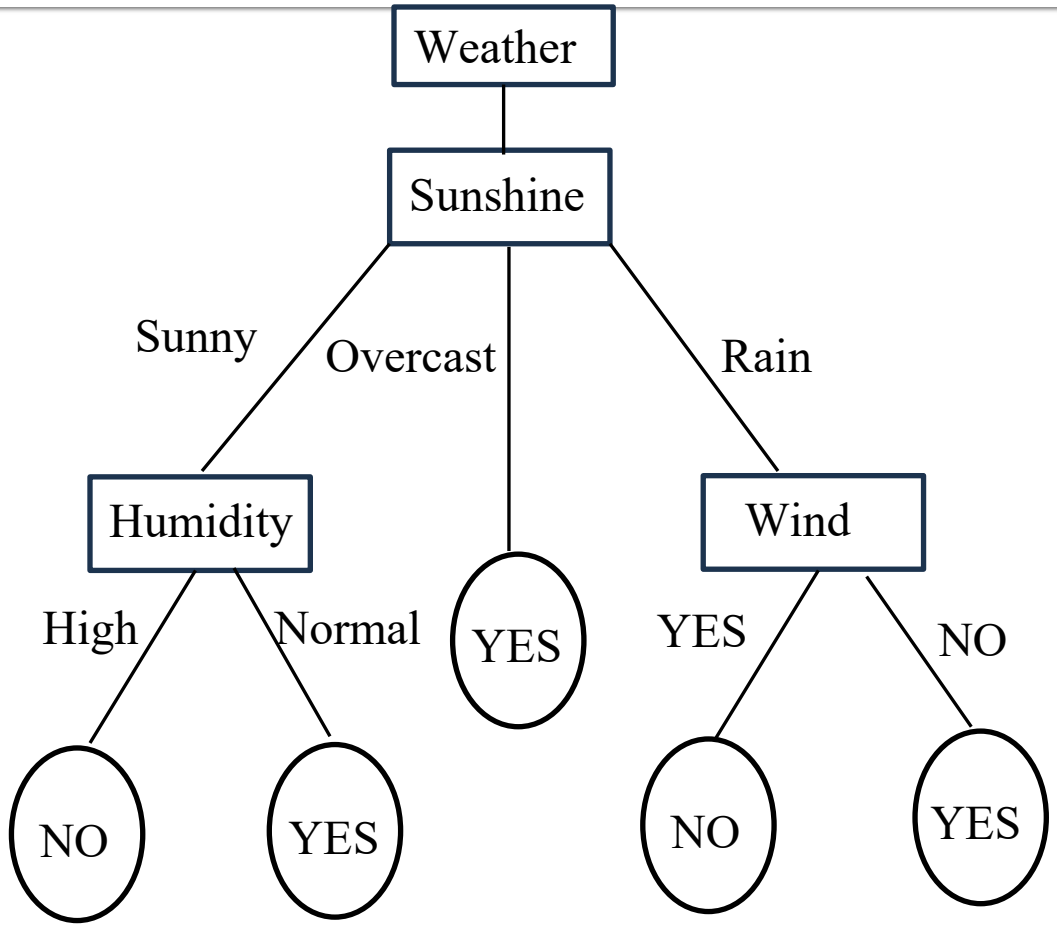
- The distribution of the dependant variable for the value ‘Rain’ is:

Number	Sunshine	Temperature	Humidity	Wind	Play
4	Rain	Warm	High	No	Yes
5	Rain	Cool	Normal	No	Yes
6	Rain	Cool	Normal	Yes	No
10	Rain	Warm	Normal	No	Yes
14	Rain	Warm	High	Yes	No

- So,

Variable	***Impurity Variation***
Temperature	0.02
Humidity	0.02
Wind	**0.971**

Application1 : Weather Dataset



Final Decision Tree

Atrributes Significance

- The decision tree that has just been constructed provides information about the level of significance of the attributes with respect to the classification of the dependent variable.

- The attribute *"Temperature"* is not used in the tree; this indicates that it is not statistically significant for determining the class of the dependent variable (to play or to play not).

- If the attribute *"Sunshine"* equals *"sun"*, the attribute *"Wind"* is no longer significant. If the attribute *"Sunshine"* equals *"rain"*, then the attribute *"Humidity"* is not significant.

Application1 Bis: Weather Dataset

Case of a Numerical Attribute

Number	Sunshine	Temperature (°C)	Humidity (%)	Wind	Play
1	Sunny	27.5	85	No	No
2	Sunny	25	90	Yes	No
3	Overcast	26.5	86	No	Yes
4	Rain	20	96	No	Yes
5	Rain	19	80	No	Yes
6	Rain	17.5	70	Yes	No
7	Overcast	17	65	Yes	Yes
8	Sunny	21	95	No	No
9	Sunny	16.5	70	No	Yes
10	Rain	22.5	80	No	Yes
11	Sunny	22.5	70	Yes	Yes
12	Overcast	21	90	Yes	Yes
13	Overcast	25.5	75	No	Yes
14	Rain	20.5	91	Yes	No

Application1 Bis: Weather Dataset

In order to determine the threshold at which to split a numerical variable:

Sort the numerical variable in ascending order

Number	Sunshine	Temperature (°C)	Humidity (%)	Wind	Play
9	Sunny	16.5	70	No	Yes
7	Overcast	17	65	Yes	Yes
6	Rain	17.5	70	Yes	No
5	Rain	19	80	No	Yes
4	Rain	20	96	No	Yes
14	Rain	20.5	91	Yes	No
8	Sunny	21	95	No	No
12	Overcast	21	90	Yes	Yes
10	Rain	22.5	80	No	Yes
11	Sunny	22.5	70	Yes	Yes
2	Sunny	25	90	Yes	No
13	Overcast	25.5	75	No	Yes
3	Overcast	26.5	86	No	Yes
1	Sunny	27.5	85	No	No

Application1 Bis: Weather Dataset

Number	Sunshine	Temperature (°C)	Humidity (%)	Wind	Play
9	Sunny	16.5	70	No	Yes
7	Overcast	17	65	Yes	Yes
6	Rain	17.5	70	Yes	No
5	Rain	19	80	No	Yes
4	Rain	20	96	No	Yes
14	Rain	20.5	91	Yes	No
8	Sunny	21	95	No	No
12	Overcast	21	90	Yes	Yes
10	Rain	22.5	80	No	Yes
11	Sunny	22.5	70	Yes	Yes
2	Sunny	25	90	Yes	No
13	Overcast	25.5	75	No	Yes
3	Overcast	26.5	86	No	Yes
1	Sunny	27.5	85	No	No

- Do not separate two successive observations that belong to the same class
- If the split is made between two values v and $w(v < w)$, the threshold s is set to $s = \frac{v + w}{2}$
- Choose s so that the information gain is maximal.

Application1 Bis: Weather Dataset

For the variable *Temperature*, the possible values are:

17.25, 18.25, 20.25, 21, 23.75, 25.25, 27.

For $s = 17.25$, the information gains is:

$$\begin{aligned}\Phi_{\text{Temperature}}(s = 17.25, t_0) &= \Delta Imp(t_0) = Imp(t_0) - \pi_L Imp(t_L) - \pi_R Imp(t_R), \\ &= \left[-\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right)\right] - \frac{2}{14} \times 0 \\ &\quad - \frac{12}{14} \times \left[-\frac{7}{12}\log_2\left(\frac{7}{12}\right) - -\frac{5}{12}\log_2\left(\frac{5}{12}\right)\right], \\ &= 0.1.\end{aligned}$$

In a similar way, in function of the threshold s, the information gains is:

s	$\Phi_{\text{Temperature}}(s, t_0)$
17.25	0.1
18.25	...
20.25	...
21	...
23.75	...
25.25	...
27	...

Application1 Bis: Weather Dataset

Remark

We have shown how to choose the threshold for a given numerical attribute. Thus, this method is applied to each numerical attribute, and for each one, we determine the threshold s_{opt} that produces the maximum information gain.

Finally, the attribute chosen to perform the split is the one, among the numerical and categorical attributes, that produces the maximum information gain.

Missing Values

Missing Attribute Values:

- Some values of the explicative variables are missing.
- Some values of the target variable are missing.
- During the classification phase, if an attribute's value is missing, it's impossible to decide which branch to choose to classify the object.

The Most Popular Solutions:

- Discard instances that have missing values.
- Imputation**:** Mean, Mode, Median, using expert assistance, etc.
- **Quinlan's Method** : Use the decision tree itself to determine the missing value of an attribute

Python Parameter

`min_samples_split`

Minimum number of samples required to split an internal node

Default= 2

`min_samples_leaf`

Minimum number of samples required to be at a leaf (terminal node)

Default = 1

`ccp_alpha`

Complexity parameter used for *cost-complexity pruning*

Default: **0.0** (no pruning by default)

Quinlan's Method

Quinlan's Method in the context of decision trees (ID3 or C4.5 algorithms) is used to handle missing values but *not only for estimating* them; it's part of how Quinlan's algorithms deal with incomplete data during both training and classification.

" The C4.5 method does not solve the missing value problem directly by filling in the blanks or dropping missing values. Instead, it works around the problem by modifying the tree-building algorithm itself."

Quinlan's Method

Quinlan introduced a **probabilistic approach** to handle missing attribute values.

During Training (building the tree) when an attribute value is missing for a training instance Quinlan's method **splits the instance probabilistically** among the possible attribute values.

If an attribute *A* has values {High, Medium, Low} and an instance has A = ?, then that instance is **distributed fractionally** across branches according to the **observed frequencies** of High, Medium, and Low in the data.

This allows the algorithm to:

- still use all training data (no loss),
- **estimate the contribution** of the instance based on the distribution of known values.

Quinlan's Method

Person	Income	Student	Buys_Computer
1	High	No	No
2	High	Yes	Yes
3	Medium	No	Yes
4	?	Yes	Yes
5	Low	No	No

Quinlan's Method

Step 1: Frequency distribution of **known** Income values

Income	Count
High	2
Medium	1
Low	1

Estimate probabilities for each Income value:

$$P(High) = 2/4 = 0.5, P(Medium) = 1/4 = 0.25, P(Low) = 1/4 = 0.25$$

Quinlan's Method

Step 2: Compute information gain

When calculating the information gain for the attribute Income, Quinlan's method does not ignore Person 4 (with missing Income). Instead, Person 4 is fractionally counted in all three branches, weighted by the probabilities above.

Income	Persons contributing	Weight contribution from Person 4
High	1, 2 (+ part of 4)	0.5
Medium	3 (+ part of 4)	0.25
Low	5 (+ part of 4)	0.25

Quinlan's Method

Step1: Parent (root) entropy

Target counts (using all 5 instances, missing values allowed for target):

Count_YES = 3

Count_NO = 2

Total = 5

Entropy :

$$H(S) = -\sum_{c} p\,(c) log_2 p(c)$$

$$p(Yes) = 3/5 = 0.6$$

contribution(yes) = $-0.6 log_{2_2}(0.6) = 0.44$

$$p(No) = 2/5 = 0.4$$

contribution(no) = $-0.4 log_{2_2}(0.4) = 0.52$

Parent entropy:

$$H(S) = 0.44 + 0.52 = 0.97$$

Quinlan's Method

Step2: Build Branch Counts (with fractional instance 4)

High branch:
Full members: person1 (No), person2 (Yes)
Add fractional from person4: +0.5 to Yes
Counts:
Yes = $1 + 0.5 = 1.5$
No = 1
Branch total = 2.5
Probabilities in High branch:

$$p_H(Yes) = 1.5/2.5 = 0.6$$
$$p_H(No) = 1/2.5 = 0.4$$

$$Entropy\ High\ = H(High) = -0.60.6log_{2_2}(0.6) - 0.4log_{2_2}(0.4)) = 0.97$$

High branch has same class proportions as the parent, so same entropy

Quinlan's Method

Medium branch:
Full: person3 (Yes)
Add fractional: +0.25 (person4)
Counts:
Yes = $1 + 0.25 = 1.25$
No = 0
Branch total = 1.25
Probabilities:
$p(Yes) = 1.25/1.25 = 1.0$,$p(No) = 0$
Entropy = 0(pure class)

Quinlan's Method

Low branch:
Full: person5 (No)
Add fractional: +0.25 (person4)
Counts:
Yes = $0 + 0.25$
No = 1
Branch total = 1.25
Probabilities:

$$p_L(Yes) = 0.25/1.25 = 0.2$$
$$p_L(No) = 1/1.25 = 0.8$$

$$\text{Entrlow}(Low) = H(Low) = -0.2log_{2_2}(0.2) - 0.8log_{2_2}(0.8) = 0.72$$

Quinlan's Method

Weight each branch entropy by the branch's proportion of the overall (total = 5):

Branch totals:

High total = 2.5 → weight = 2.5 / 5 = 0.5

Medium total = 1.25 → weight = 1.25 / 5 = 0.25

Low total = 1.25 → weight = 1.25 / 5 = 0.25

Weighted entropy:

$$H_{split} = 0.5 \times H(High) + 0.25 \times H(Medium) + 0.25 \times H(Low)$$

$$H_{split} = 0.48 + 0 + 0.18 = 0.66$$

Quinlan's Method

5) Calculate the Information Gain for splitting on Income

$$\text{Gain}(S, Income) = H(S) - H_{split}$$
$$= 0.973 - 0.669 = 0.304$$

So the **information gain** for Income (using Quinlan's fractional handling of the missing value) is **0.304**

Quinlan's Method

In order to highlight the advantage of Quinlan's method, first, remove the observation containing the missing value from the dataset. Then, recalculate the information gain using the variable **Income** to split.

Generalities

Problems

- The algorithm does not backtrack or question its choices : A Greedy nature of classical decision trees. Once a split is chosen at a node, it is never questioned.
- We can obtain low error on the training set, but also low predictive power (Overfitting phenomenon)

- Choice of parameters (*min_samples_split, min_samples_leaf, ccp_alpha*): wich value is the optimal value ?

Solutions

- Pruning: Fix overfitting by simplifying the tree after it's built to improve generalization.

- Cross-Validation technique: for evaluating a model's true performance and tuning its parameters

Training & Test Datasets

Concept: Split data to build a model that works in the real world.

- **Training Set**

Used to **teach** the model by finding patterns and relationships. This is where the model *learns*.

- **Test Set**

Used to **evaluate** the model's performance on new, unseen data. This simulates a real-world scenario.

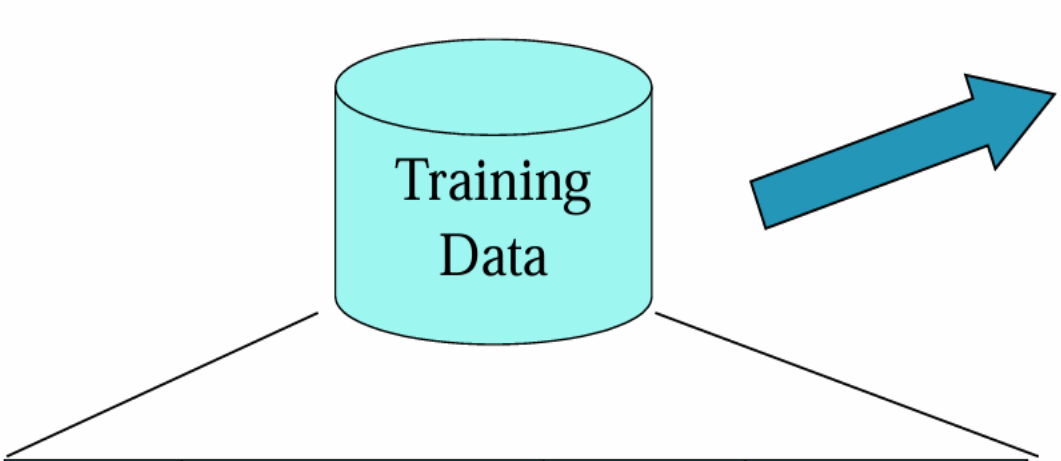
- **Goal: Prevent Overfitting**

This separation ensures the model learns general principles, not just the specific details of the training data, so it can make accurate predictions on future data.

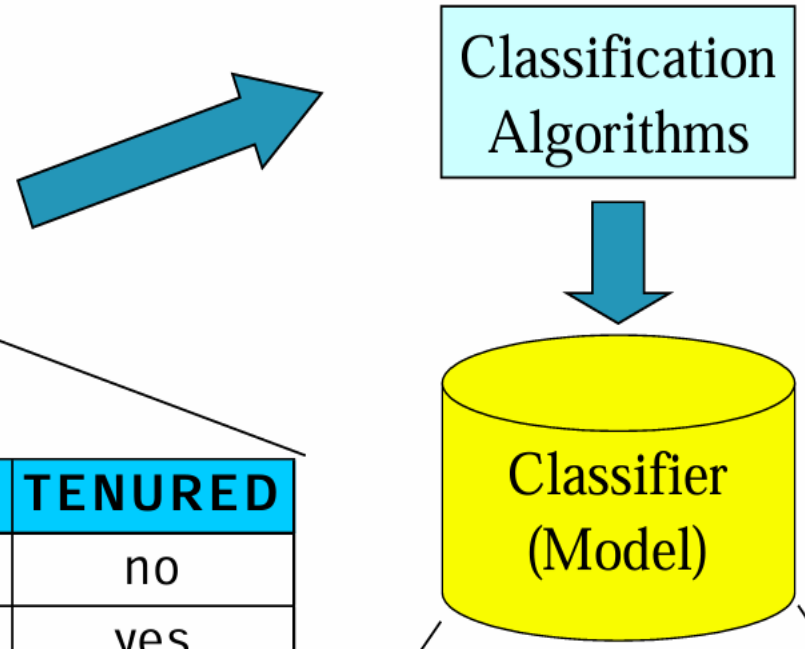
- **Standard Practice:** The 80/20 Split

A common rule of thumb is to use **80%** of the data for training and **20%** for testing, ensuring a fair and reliable evaluation.

Training & Test Datasets

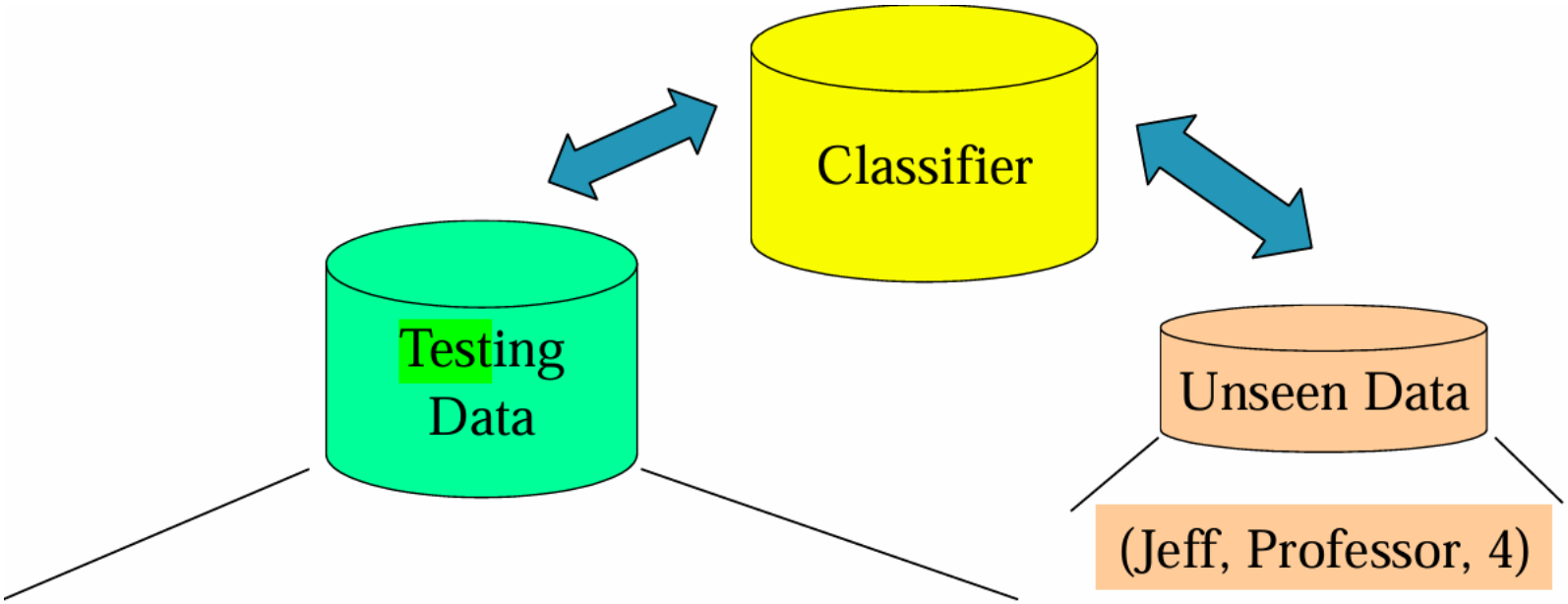


NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no



IF rank = ‘professor’
OR years > 6
THEN tenured = ‘yes’

Training & Test Datasets

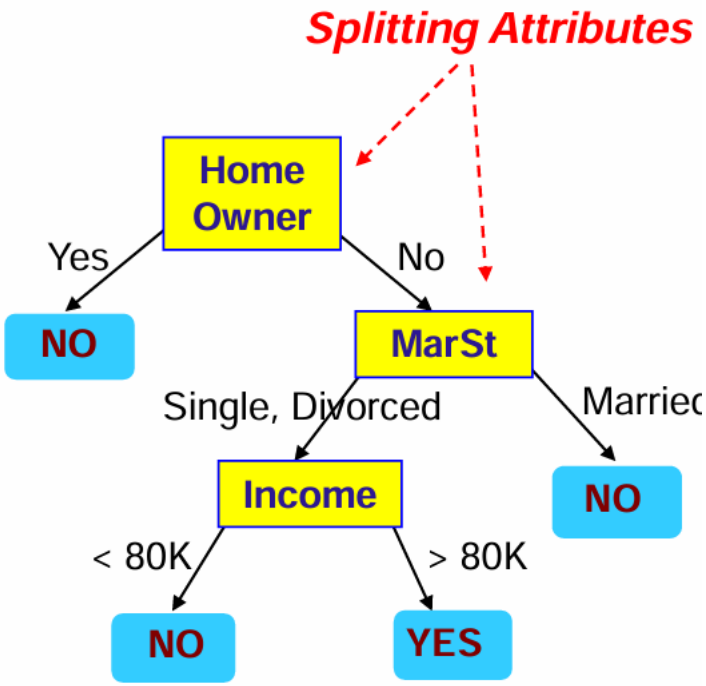


NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Merlisa	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes

ANOTHER EXAMPLE

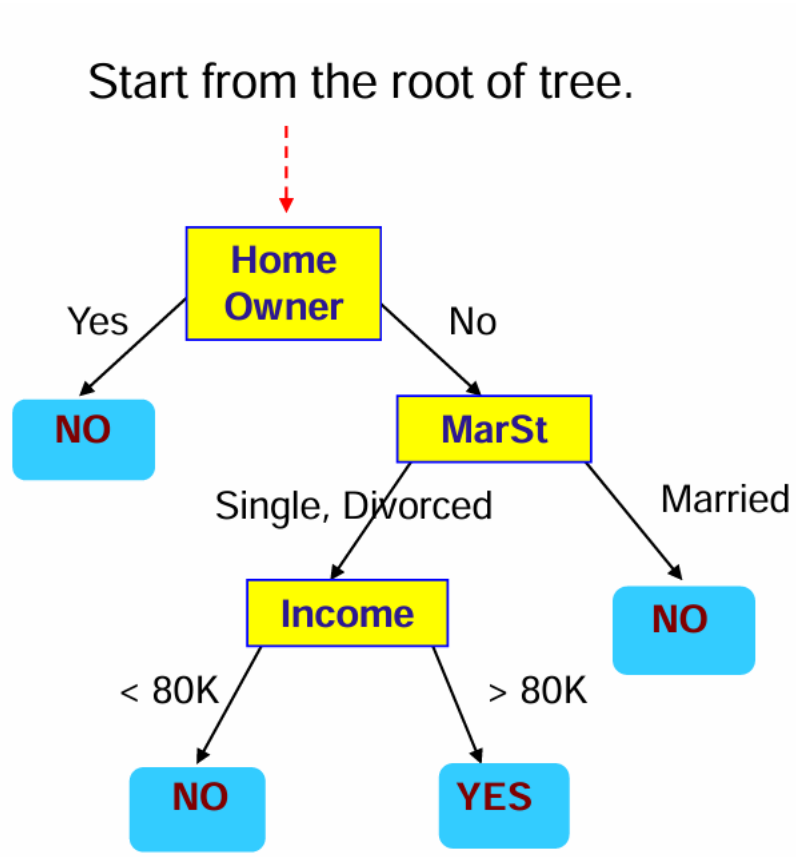
	categorical	categorical	continuous	class
ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Training Data



Model: Decision Tree

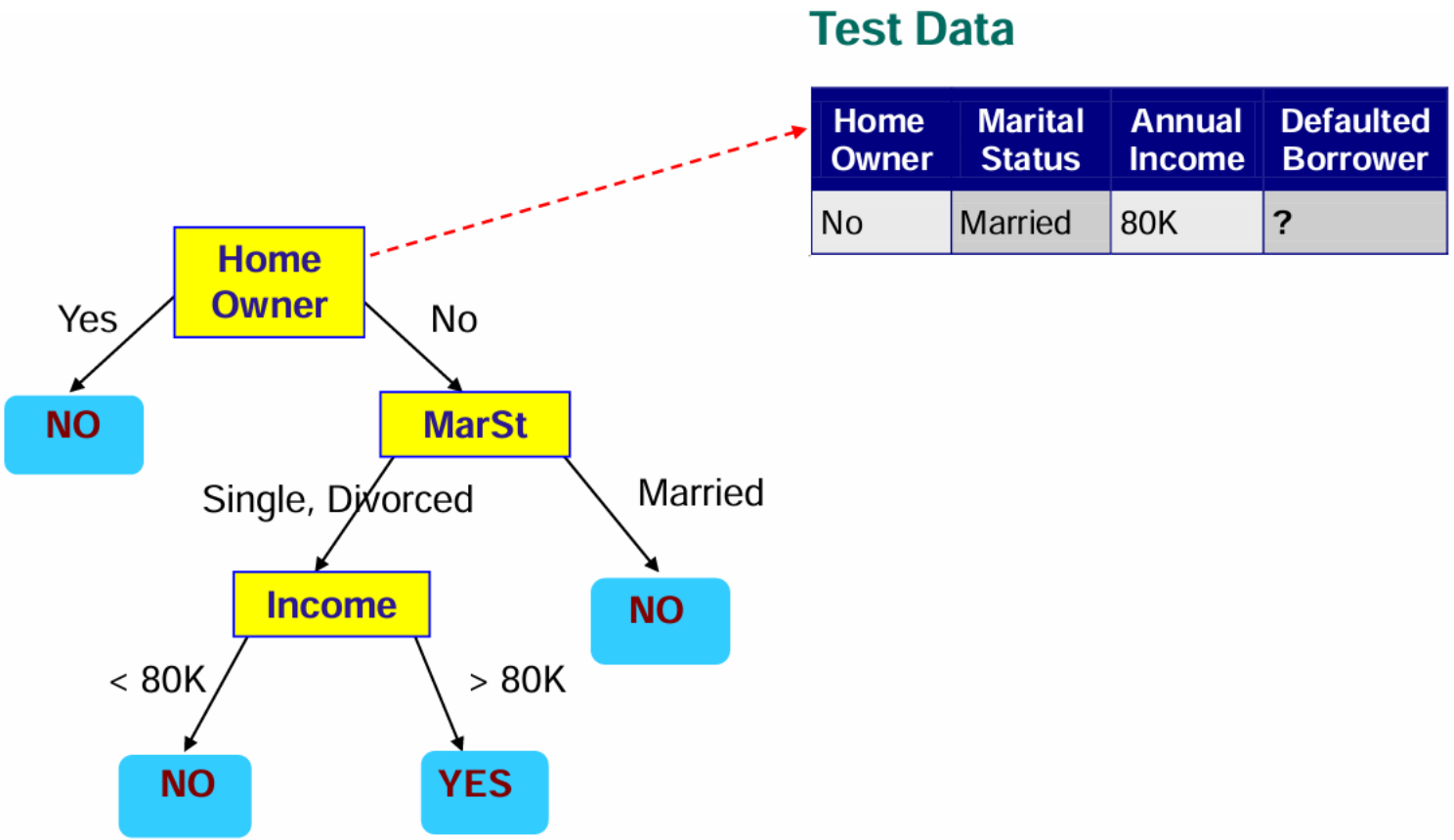
ANOTHER EXAMPLE



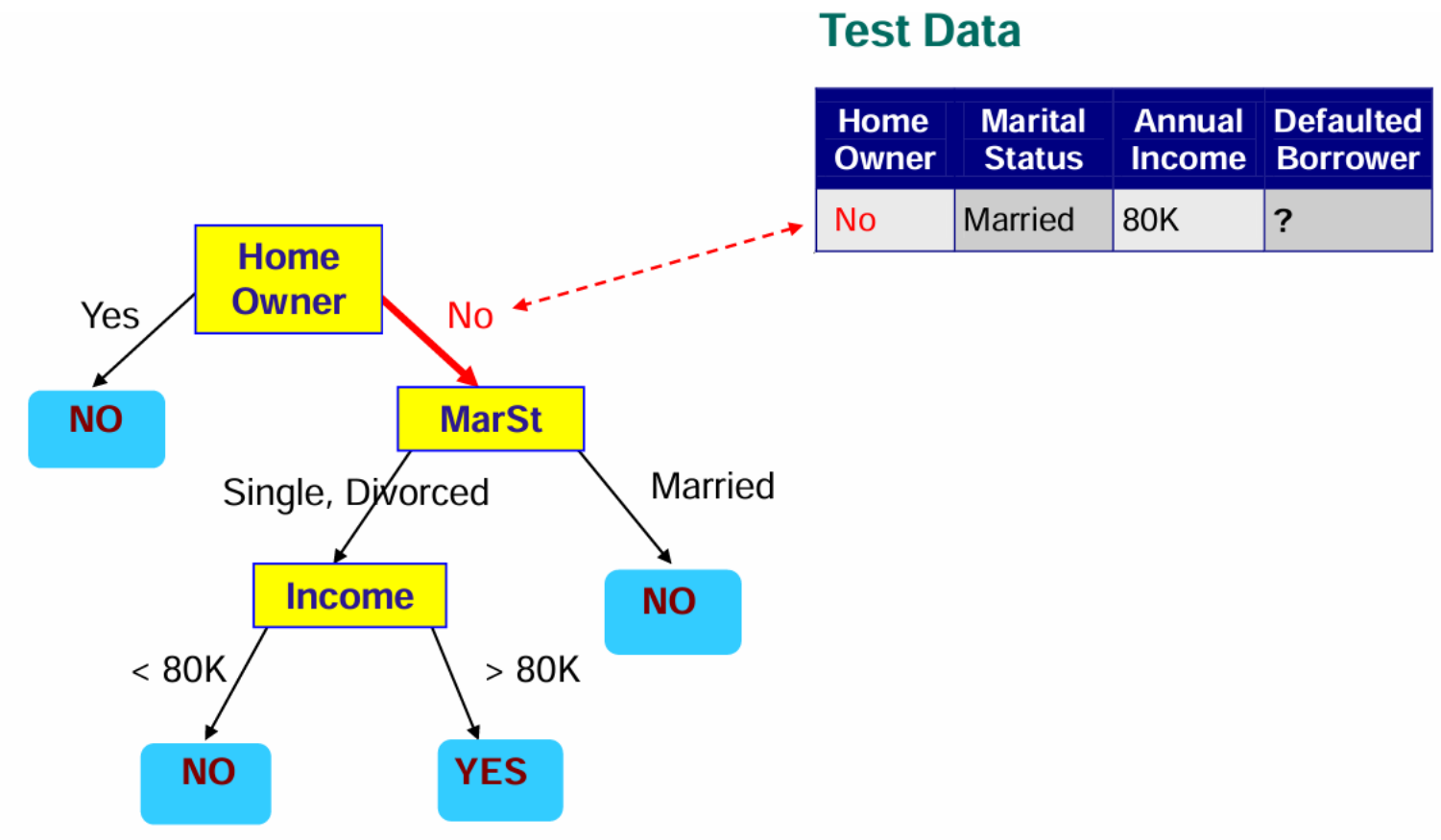
Test Data

Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?

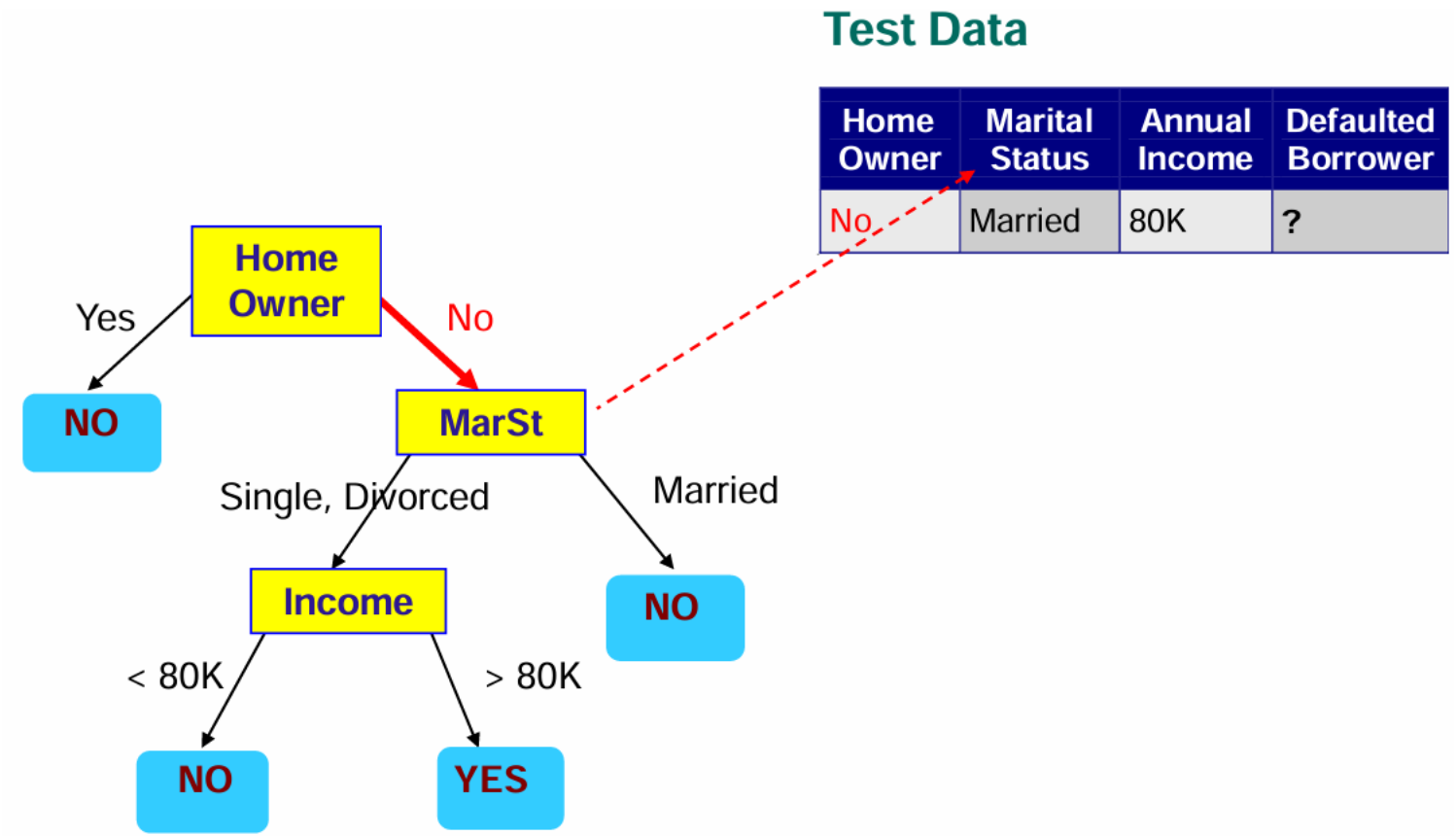
ANOTHER EXAMPLE



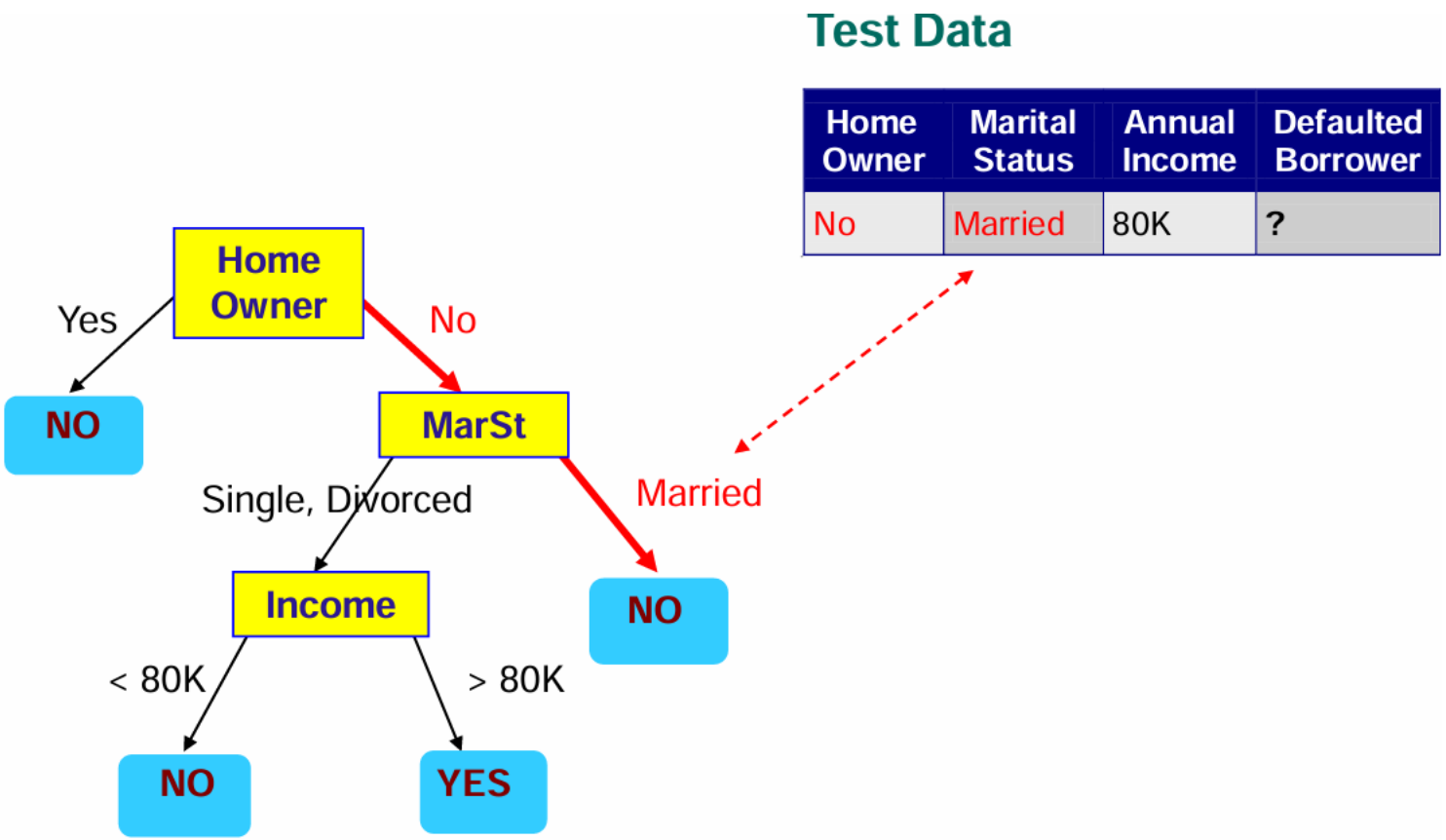
ANOTHER EXAMPLE



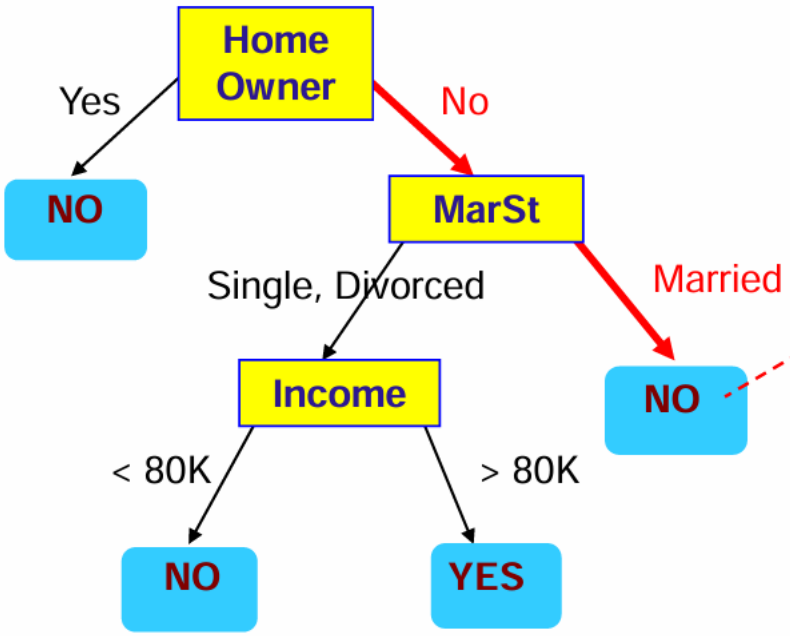
ANOTHER EXAMPLE



ANOTHER EXAMPLE



ANOTHER EXAMPLE



Test Data

Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?

Assign Defaulted to "No"

CROSS-VALIDATION

• **Data Splitting**
The dataset is divided into *k* subsets (folds) of equal size.

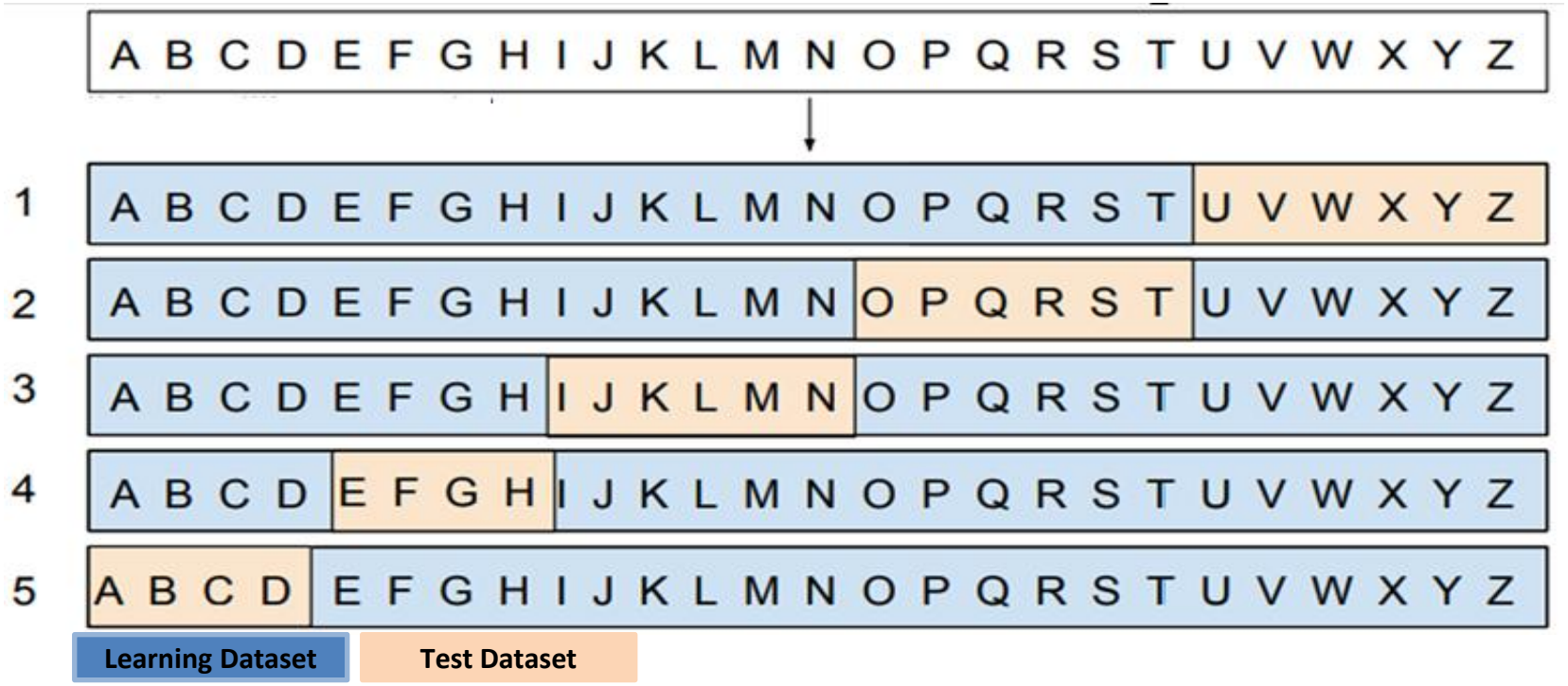
• **Training & Testing Rotation**
The model is trained *k* times. Each time, one fold is used for testing, and the remaining for training.

• **Performance Averaging**
The average of all test results gives a more reliable performance estimate.

• **Advantage**
Reduces overfitting and provides a better measure of model generalization.

CROSS-VALIDATION

The model is trained 5 times. Each time, one fold is used for testing, and the remaining for training



$$CV(\alpha) = \frac{1}{5}\sum_{v=1}^{5} MSE_v^{(test)}$$

REGRESSION DECISION TREE

In the case of a quantitative response variable (regression), for example Y = salary ,in thousands of dollars, the regression tree algorithm operates as follows:

1. Partitioning the predictor space:

The space of explanatory variables $(X_1, X_2, \ldots, X_p)$ is recursively divided into K **exhaustive and mutually exclusive regions** $R_1, R_2, \ldots, R_K$, ($R_i \in \mathbb{R}^p$)

2. Prediction within regions:

For any observation that falls into a region R_k ,the predicted value of Y is the **average of the observed Y values** in that region.

The construction of the tree relies on three key elements:

•A splitting criterion to determine the optimal partition of the predictor space;

•A stopping rule to decide when a node becomes terminal;

•A pruning procedure to prevent overfitting and improve generalization.

REGRESSION DECISION TREE

To determine the best split: find regions $R_1, \ldots, R_i, \ldots, R_k$, that minimize the flollowing loss function:

$$\sum_{k=1}^{K} \sum_{y_i \in R_k} \left(y_i - \overline{y}_{R_k}\right)^2,$$

where $\overline{y}_{R_k} = \frac{1}{Card(R_k)} \sum_{i \in R_k} y_i$

At each node, we search for the predictor variable X_j and the split threshold s that **maximize the reduction in heterogeneity** of the child nodes. These child regions are denoted by R_1(left) and R_2(right), defined as:

$$R_1(j, s) = \{observations: X_j \leq s\}, R_2(j, s) = \{observations: X_j > s\}$$

Thus, the objective is to find j and s that **minimize the loss function**:

$$\sum_{y_i \in R_1(j,s)} \left(y_i - \overline{y}_{R_1(j,s)}\right)^2 + \sum_{y_i \in R_2(j,s)} \left(y_i - \overline{y}_{R_2(j,s)}\right)^2.$$

REGRESSION DECISION TREE

Stopping rule: Use the parameters *min_samples_split* and *min_samples_leaf*

Pruning: Construct a **sequence of nested subtrees** based on a penalty on tree complexity. i.e. , for a given value of the complexity parameter **α**, find subtree $T \subseteq T_{max}$ that **minimizes**:

$$\sum_{m=1}^{|T|} \sum_{y_i \in R_m} \left(y_i - \bar{y}_{R_m}\right)^2 + \alpha |T|,$$

where $|T|$ is the number of terminal nodes and **α** is the complexity parameter.

α is **estimated using V-fold cross-validation**.

Numerical Example

i	(X_1) (experience, yrs)	(X_2) (education, yrs)	(Y) (salary, $k)
1	1	10	30
2	2	12	35
3	4	11	50
4	6	13	80
5	7	14	90
6	9	16	120

search for the best first split among candidate thresholds on **both** X_1 and X_2.

Numerical Example

The total mean $\bar{Y}$ is: $\bar{Y} = (30 + 35 + 50 + 80 + 90 + 120)/6 = 67.5$

.

The total variance that we want to reduce by splitting is:

$$\mathrm{Var}_{\mathrm{total}} = \frac{1}{6}\sum_{i=1}^{6}(Y_i - \bar{Y})^2 = 1031.25.$$

Numerical Example

Candidate split points
For numeric variables we test midpoints between sorted distinct values.

X_1sorted: $[1, 2, 4, 6, 7, 9]$: candidate thresholds: $1.5,\ 3.0,\ 5.0,\ 6.5,\ 8.0.$

X_2sorted: $[10, 11, 12, 13, 14, 16]$: candidate thresholds $10.5,\ 11.5, 12.5,\ 13.5,\ 15.0$

For each candidate threshold s and for each variable, split the data into:
Left group: observations with variable $\leq s$,
Right group: observations with variable $> s$,
Compute: mean and variance inside each group. The weighted average variance after splitis calculated:

$$\mathrm{Var}_{after} = \frac{n_L}{n}\mathrm{Var}_L + \frac{n_R}{n}\mathrm{Var}_R,$$

And the reduction of the split is calculated as follows: $\mathrm{reduction} = \mathrm{Var}_{total} - \mathrm{Var}_{after}$.

The best split is the one with **largest reduction** in variance.

Numerical Example

Variable & split	left / right sizes	weighted var after split	reduction in variance
$X_1 \leq 1.5$	$n_L = 1,\ n_R = 5$	750.000	**281.250**
$X_1 \leq 3.0$	2, 4	418.750	**612.500**
$X_1 \leq 5.0$	3, 3	**180.5556**	**850.6944** ← best
$X_1 \leq 6.5$	4, 2	328.125	**703.125**
$X_1 \leq 8.0$	5, 1	480.000	**551.250**
$X_2 \leq 10.5$	1, 5	750.000	**281.250**
$X_2 \leq 11.5$	2, 4	653.125	**378.125**
$X_2 \leq 12.5$	3, 3	**180.5556**	**850.6944** ← tied best
$X_2 \leq 13.5$	4, 2	328.125	**703.125**
$X_2 \leq 15.0$	5, 1	480.000	**551.250**

After the first split, the algorithm recursively applies the same procedure inside each child node (left and right) to find further splits, until a stopping rule is met : minimum node size or/and minimum reduction and/or max depth. Pruning may then be applied to avoid overfitting.

Validity

Test	Covid-19	No Covid-19	Total
Positive	56	49	105
Negative	14	461	475
Total	70	510	580

True Positive (TP): Observations that were identified as positive by the model and are actually positive

False Negative (FP): Observations that were identified as positive by the model and are not actually positive

True Negative (TN): Observations that were identified as negative by the model and are actually negative

Flase Negative (FN): Observations that were identified as positive by the model and are not actually negative

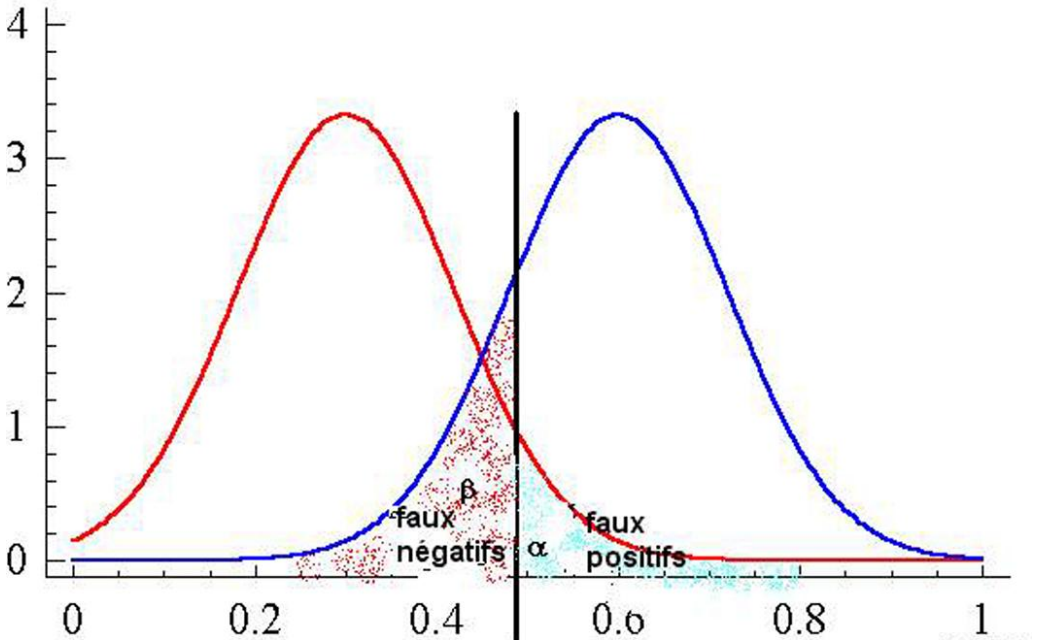
Specificity (Sp) : Proportion of true negatives among the non-sick = TN /(TN +FP) = 461/(461+49) = 90%

Sensibility (Se) : Proportion of true positives among the sick = TP /(TP +FN) = 56/(56+14) = 80%

Validity

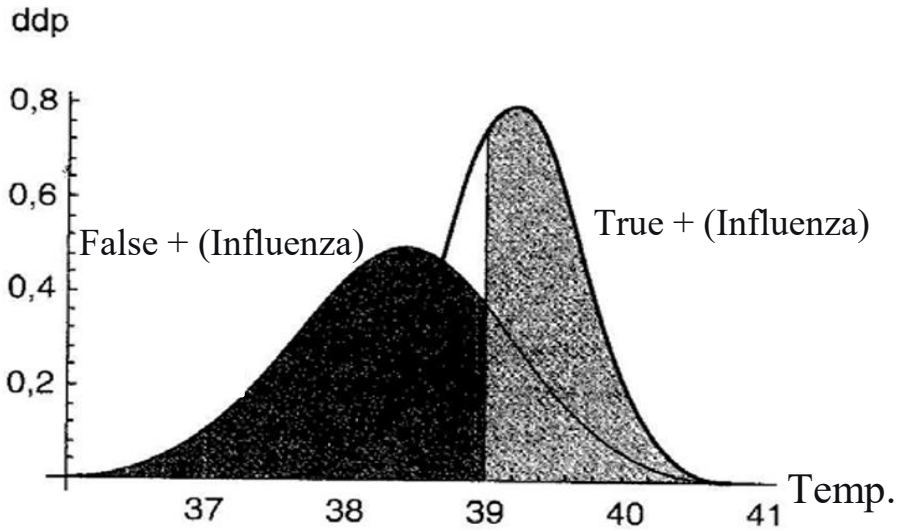
A reliable model should be **specific** and **sensitive**.

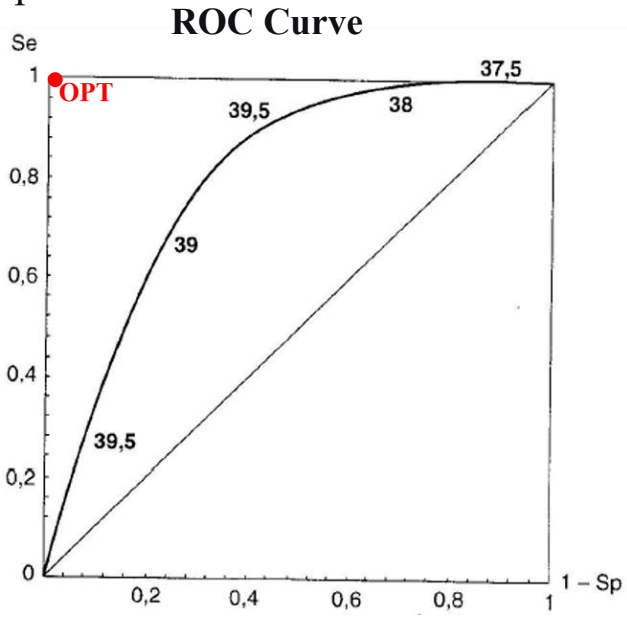
In practice, a gain in sensitivity is achieved at the cost of a loss in specificity, and vice versa.



Validity

The ROC (Receiver Operating Characteristic) curve visually represents this trade-off by plotting the True Pos. Rate **(Sensitivity)** against the False Pos. Rate **(1 - Specificity)** across all possible decision thresholds.





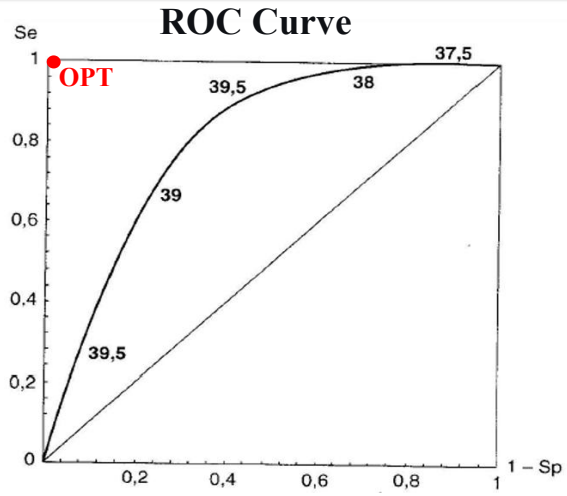
The optimal classifier would have FPR = 0 and TPR = 1. So its point on the ROC space would be at: $(0, 1)$. The top-left corner of the ROC plot.

But in reality the point (0, 1) is almost never reachable, becausebthere is **no model with 0 error,** and there is always a **trade-off** between sensitivity and specificity**.**

The goal is to find the **threshold (α)** that gives the **best balance** between:
high sensitivity and low (1 – specificity)

Validity

How to find that **optimal** point ?

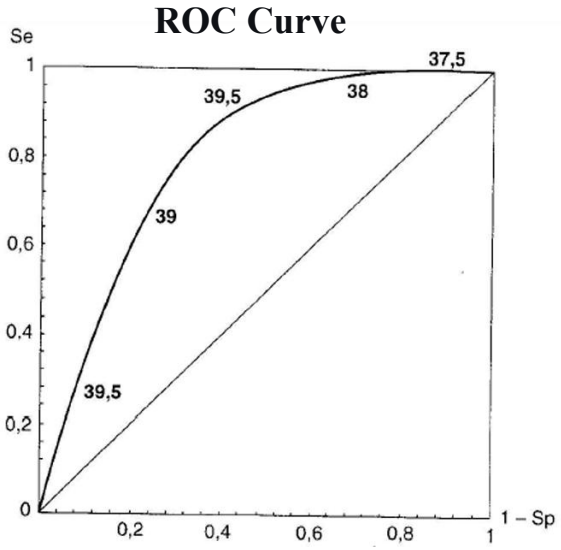


Find the point **closest to (0, 1)** in the ROC space.
For each point $P\ (1 - Spec.(\alpha)\ , 1 - Sens.(\alpha))$ of the curve ROC, the eucildian distance between *P* and the optimal point *OPT(0,1)* is calculated as follows:

$$dist(P,\ OPT),\ = \sqrt{(1 - Sp(\alpha) - 0)^2 + (Se(\alpha) - 1)^2}$$

The **optimal α** is the one corresponding to the point having the smallest distance.
The corresponding **threshold α** is considered **optimal**, since it balances both sensitivity and specificity best.

Validity



A **good model** has a curve that rises quickly toward the top-left corner high sensitivity and low false positive rate.

A **random model** (no discrimination ability) lies along the diagonal line (area = 0.5).

The **Area Under the Curve** (AUC) measures overall model quality