

Aprendizado de Máquina – IMD1101

Aula 08 – Aprendizado Supervisionado de Máquina 01

Treinamento e Teste

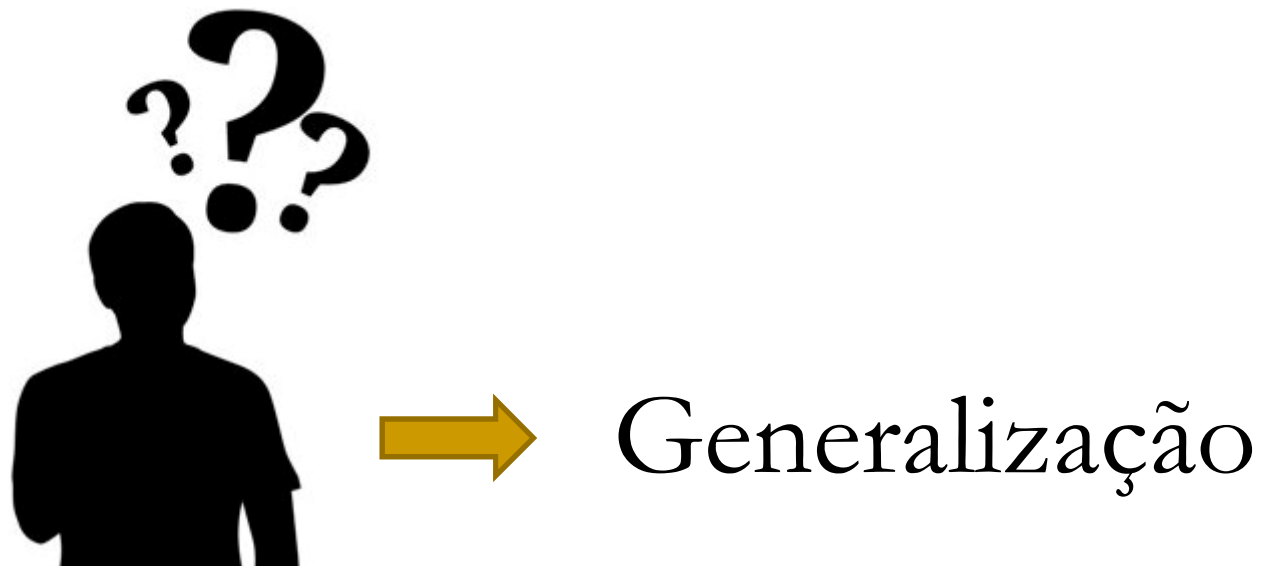
❑ Como treinar e testar os algoritmos?



<http://iasdcentralcampinas.org.br/treinamentos-apac/>

Treinamento

- ❑ O que se busca alcançar com treinamento?



- ❑ **Generalização:** a habilidade de classificar padrões de teste que não foram utilizados durante o treinamento.

Treinamento e Teste

- ❑ **Validação cruzada:** é uma técnica para avaliar a **capacidade de generalização** de um modelo, a partir de um conjunto de dados.
- ❑ Empregada em **problemas de predição**.
- ❑ Busca-se então estimar o quão **acurado** é este modelo na prática, ou seja, o seu desempenho para um **novo conjunto de dados**.
- ❑ Particiona-se o conjunto de dados em subconjuntos mutualmente exclusivos. Formas: *holdout* e *k-fold*.

Treinamento e Teste

- ❑ **Método holdout:** consiste em dividir o conjunto total de dados em dois subconjuntos mutuamente exclusivos, um para **treinamento** e outro para **teste** (validação).
- ❑ O conjunto de dados pode ser **separado** em **quantidades** iguais ou não. É possível ter $2/3$ dos dados para **treinamento** e o $1/3$ restante para **teste**.
- ❑ Esta abordagem é indicada para **grande quantidade** de dados. Quando o conjunto é **pequeno**, o **erro calculado** na predição pode sofrer **muita variação**.

Treinamento e Teste

- ❑ **Método k-fold:** consiste em dividir o conjunto total de dados em k subconjuntos mutuamente exclusivos do mesmo tamanho.
- ❑ Um subconjunto é utilizado para teste e os $k - 1$ restantes são utilizados para estimação dos parâmetros e calcula-se a acurácia do modelo.

Holdout

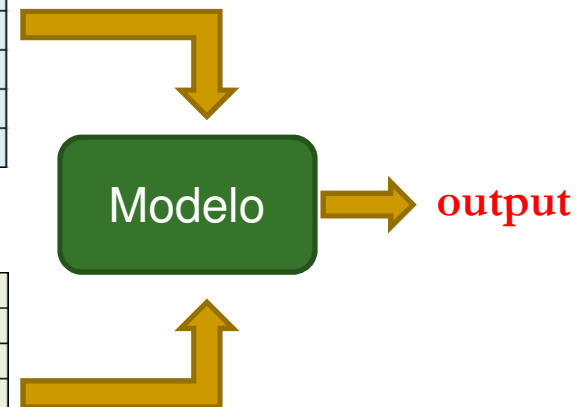
□ Exemplo 2/3 – 1/3:

Training

N#	preg	plasma	pressure	skin	insulin	mass	pedigree	age	class
2	3	126	88	41	235	39,3	0,704	27	tested_negative
4	0	137	40	35	168	43,1	2,288	33	tested_positive
5	3	78	50	32	88	31,0	0,248	26	tested_positive
6	2	197	70	45	543	30,5	0,158	53	tested_positive
7	11	143	94	33	146	36,6	0,254	51	tested_positive
8	10	125	70	26	115	31,1	0,205	41	tested_positive
9	13	145	82	19	110	22,2	0,245	57	tested_negative
11	5	166	72	19	175	25,8	0,587	51	tested_positive
13	1	103	30	38	83	43,3	0,183	33	tested_negative
15	3	126	88	41	235	39,3	0,704	27	tested_negative

Test

1	1	115	70	30	96	34,6	0,529	32	?
3	1	89	66	23	94	28,1	0,167	21	?
10	1	189	60	23	846	30,1	0,398	59	?
12	0	118	84	47	230	45,8	0,551	31	?
14	1	115	70	30	96	34,6	0,529	32	?



k-fold

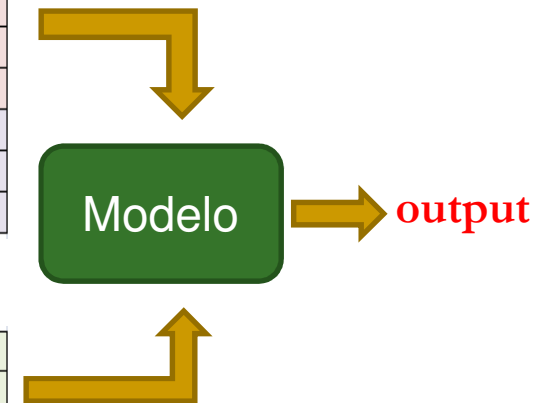
□ Exemplo (5-fold):

Training

N#	preg	plasma	pressure	skin	insulin	mass	pedigree	age	class
1	1	115	70	30	96	34,6	0,529	32	tested_negative
2	3	126	88	41	235	39,3	0,704	27	tested_negative
4	0	137	40	35	168	43,1	2,288	33	tested_positive
5	3	78	50	32	88	31,0	0,248	26	tested_positive
6	2	197	70	45	543	30,5	0,158	53	tested_positive
7	11	143	94	33	146	36,6	0,254	51	tested_positive
8	10	125	70	26	115	31,1	0,205	41	tested_positive
9	13	145	82	19	110	22,2	0,245	57	tested_negative
11	5	166	72	19	175	25,8	0,587	51	tested_positive
12	0	118	84	47	230	45,8	0,551	31	tested_positive
13	1	103	30	38	83	43,3	0,183	33	tested_negative
15	3	126	88	41	235	39,3	0,704	27	tested_negative

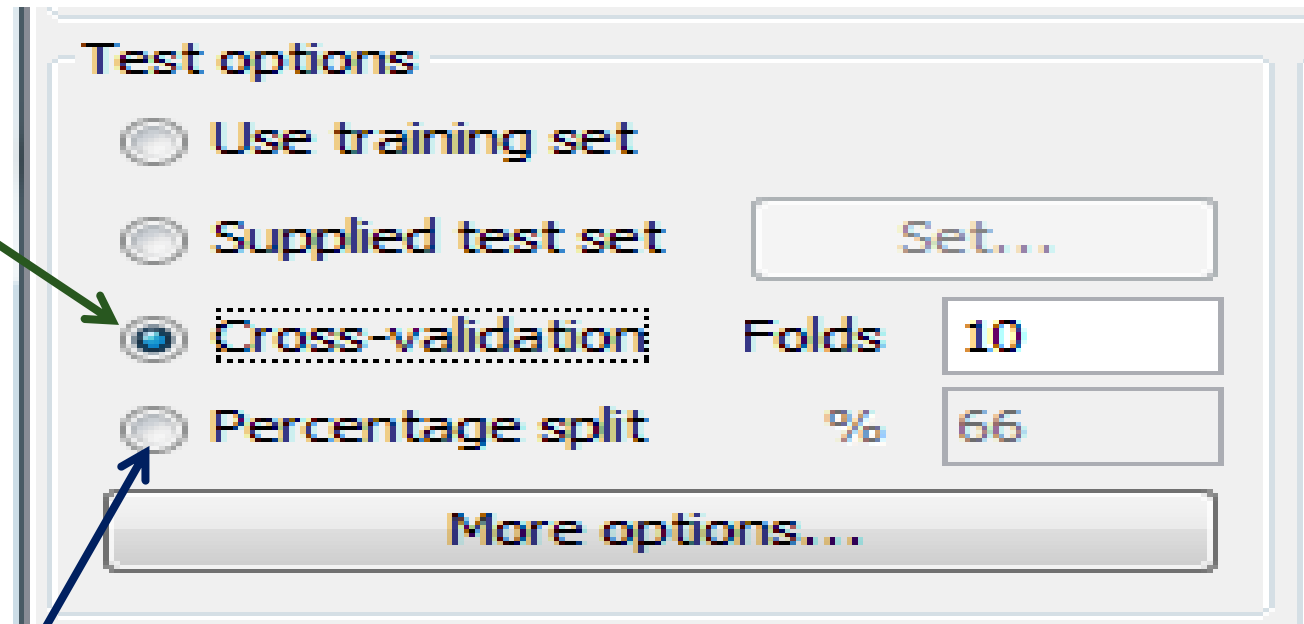
Test

3	1	89	66	23	94	28,1	0,167	21	?
10	1	189	60	23	846	30,1	0,398	59	?
14	1	115	70	30	96	34,6	0,529	32	?



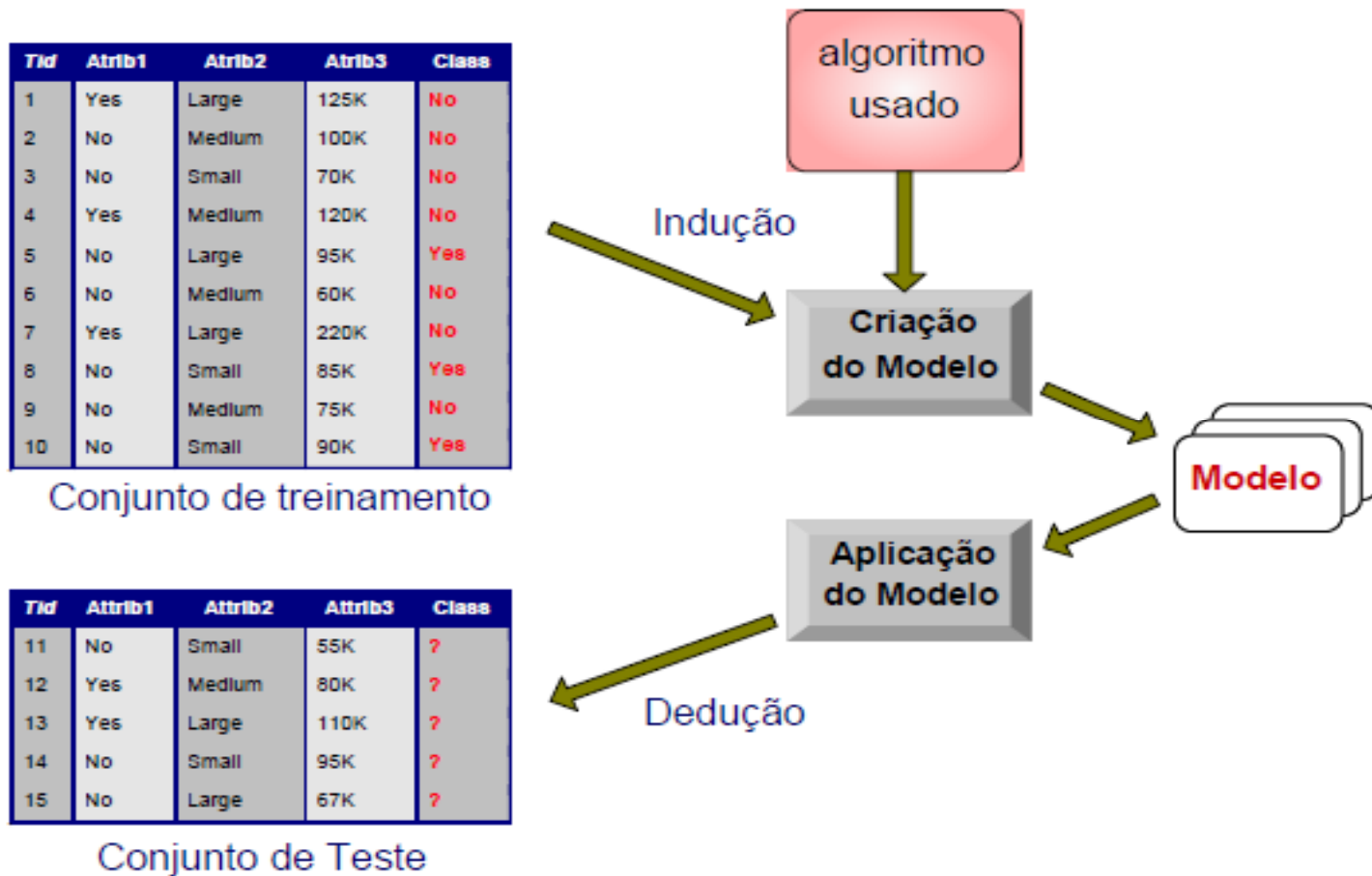
Treinamento e Teste

k-fold



Holdout

O que é Classificação?



<http://www.ime.unicamp.br/~wanderson/Aulas/>

Onde aplicar Classificação?

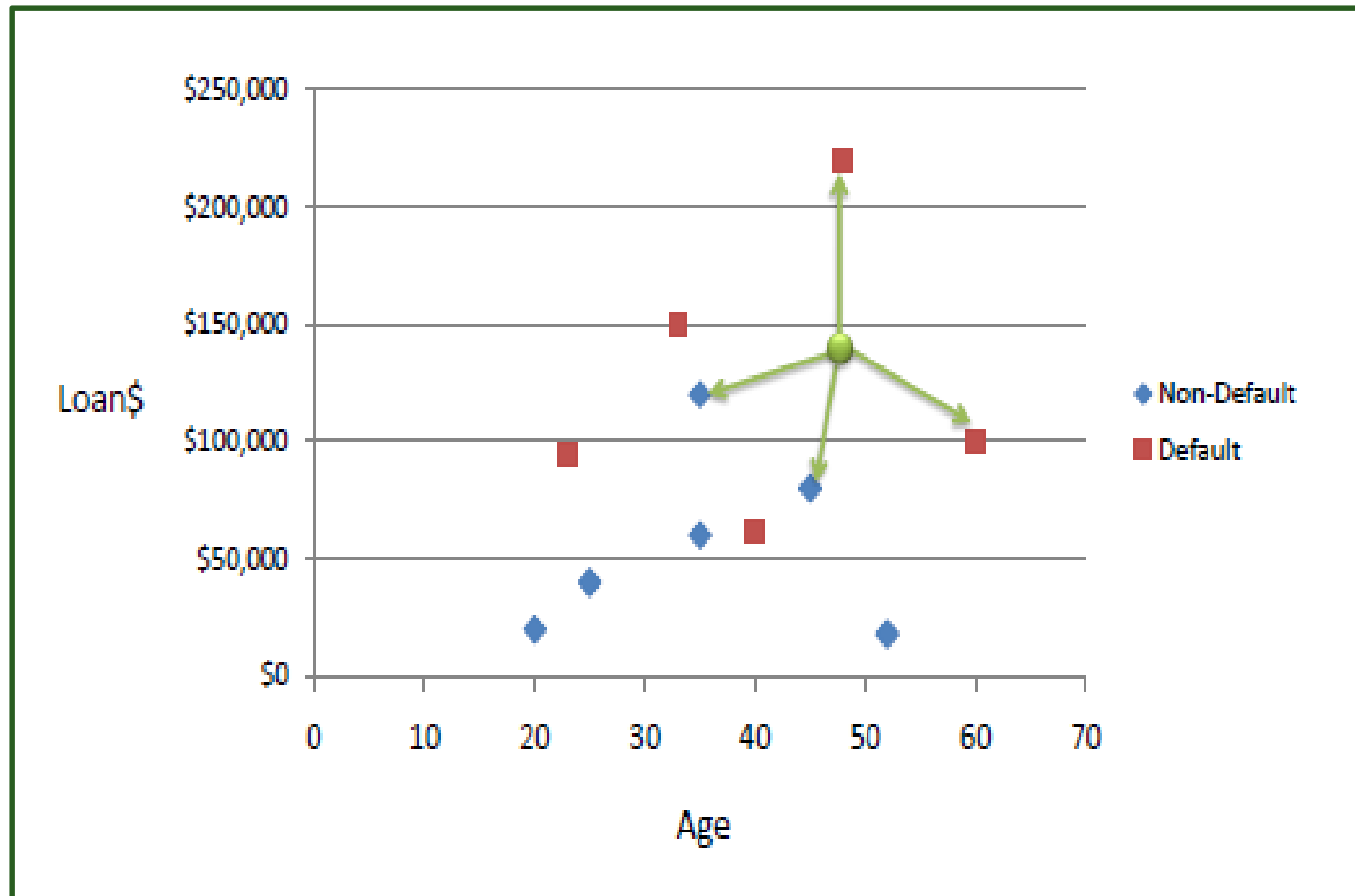
- ❑ Classificar tumores como benigno ou maligno.
- ❑ Classificar transações de cartão de crédito como legítima ou fraudulenta.
- ❑ Classificar estruturas secundárias de proteínas como *alpha-helix*, *beta-sheet* ou *random coil*.
- ❑ Avaliar riscos de empréstimos, previsão de tempo, etc.
- ❑ Sistema de alerta de geada.
- ❑ Qualquer sistema que **tome decisão**.

Boas Características - Classificador

- ❑ Precisão
- ❑ Velocidade
 - ❖ Tempo para construir o modelo.
 - ❖ Tempo para usar o modelo.
- ❑ Robustez
 - ❖ Capacidade de lidar com ruídos e valores faltantes (missing).
- ❑ Escalabilidade
 - ❖ Eficiência em banco de dados residentes em disco.
- ❑ Interpretabilidade
 - ❖ Clareza fornecida pelo modelo.

k-NN (k Nearest Neighbor)

k-NN (k Nearest Neighbor)



http://www.saedsayad.com/k_nearest_neighbors.htm

k-NN (k Nearest Neighbor)

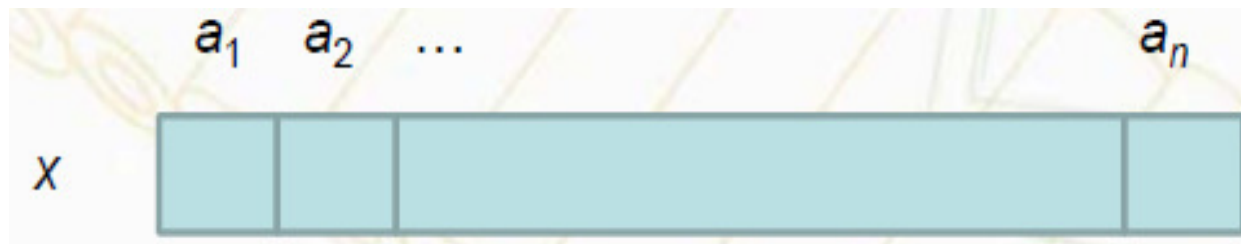
- ❑ Algoritmo de aprendizado mais simples.
- ❑ Algoritmo baseado em Instâncias.
- ❑ Este algoritmo supõe que todos os padrões (instâncias) são pontos no espaço n-dimensional R^n .
- ❑ Os vizinhos mais próximos de um padrão são definidos em termos da **distância Euclidiana**.
- ❑ A regra dos vizinhos mais próximos: classificar um ponto x atribuindo a ele o rótulo mais frequente dentre as k amostras mais próximas (esquema de votação).

k-NN (k Nearest Neighbor)

- Vamos considerar uma instância arbitrária x que é descrita pelo vetor de características:

$$x = \langle a_1(x), a_2(x), \dots, a_n(x) \rangle$$

onde $a_r(x)$ representa o valor do r -ésimo atributo da instância x .



<http://www.ppgia.pucpr.br/~alekoe/AM/2013/>

k-NN (k Nearest Neighbor)

- Então a distância Euclidiana entre duas instâncias x_i e x_j é definida como $d(x_i, x_j)$, onde:

$$d(x_i, x_j) \equiv \sqrt{\sum_{r=1}^n (a_r(x_i) - a_r(x_j))^2}$$

k-NN (k Nearest Neighbor)

❑ Algoritmo de Treinamento:

- ❖ Não há um treinamento explícito.
 - Classificam exemplos nunca vistos por meio de exemplos similares conhecidos.
 - Os próprios padrões são utilizados como base para a resposta do k-NN.
 - Método é denominado de *lazy*, pois necessitam manter os exemplos na memória para classificar novos exemplos.

❑ Para cada padrão de treinamento $\langle \mathbf{x}, f(\mathbf{x}) \rangle$, faça:

- ❖ Adicione o exemplo a lista de exemplos_de_treinamento.

k-NN (k Nearest Neighbor)

□ Algoritmo de Classificação:

- ❖ Dado um padrão (instância) de consulta \mathbf{x}_q a ser classificado.
 - Seja $\mathbf{x}_1, \dots, \mathbf{x}_k$ as k instâncias (padrões) do exemplos_de_treinamento que são mais próximos a \mathbf{x}_q .
 - Retorne a **classe** mais comum **entre os vizinhos**.

k-NN (k Nearest Neighbor)

□ Um exemplo:

# Inst	Attr 1	Attr 2	Attr 3	Class
1	0,50	0,80	0,90	A
2	0,40	0,50	0,40	B
3	0,20	0,50	0,15	C
4	0,50	0,40	0,60	B
5	0,80	0,75	0,87	A

- Dado um exemplo: $(0.67; 0.75; 0.58)$, é fornecido que $k=3$.
- A qual classe este padrão pertence?

k-NN (k Nearest Neighbor)

□ Para os cinco padrões ficaria:

$$1: \sqrt{(0.5-0.67)^2 + (0.8-0.75)^2 + (0.9-0.58)^2} = \sqrt{0.0289+0.025+0.1024} = \sqrt{0.1338} = 0.37$$

$$2: \sqrt{(0.4-0.67)^2 + (0.5-0.75)^2 + (0.4-0.58)^2} = \sqrt{0.0729+0.0625+0.0324} = \sqrt{0.1678} = 0.41$$

$$3: \sqrt{(0.2-0.67)^2 + (0.5-0.75)^2 + (0.15-0.58)^2} = \sqrt{0.2209+0.2025+0.1849} = \sqrt{0.6083} = 0.68$$

$$4: \sqrt{(0.5-0.67)^2 + (0.4-0.75)^2 + (0.6-0.58)^2} = \sqrt{0.0289+0.1225+0.0004} = \sqrt{0.1518} = 0.39$$

$$5: \sqrt{(0.8-0.67)^2 + (0.75-0.75)^2 + (0.87-0.58)^2} = \sqrt{0.0169+0+0.0841} = \sqrt{0.101} = 0.32$$

Os três vizinhos mais próximos

k-NN (k Nearest Neighbor)

❑ Para os cinco padrões ficaria:

$$1: \sqrt{(0.5-0.67)^2 + (0.8-0.75)^2 + (0.9-0.58)^2} = \sqrt{0.0289+0.025+0.1024} = \sqrt{0.1338} = 0.37$$

$$2: \sqrt{(0.4-0.67)^2 + (0.5-0.75)^2 + (0.4-0.58)^2} = \sqrt{0.0729+0.0625+0.0324} = \sqrt{0.1678} = 0.41$$

$$3: \sqrt{(0.2-0.67)^2 + (0.5-0.75)^2 + (0.15-0.58)^2} = \sqrt{0.2209+0.2025+0.1849} = \sqrt{0.6083} = 0.68$$

$$4: \sqrt{(0.5-0.67)^2 + (0.4-0.75)^2 + (0.6-0.58)^2} = \sqrt{0.0289+0.1225+0.0004} = \sqrt{0.1518} = 0.39$$

$$5: \sqrt{(0.8-0.67)^2 + (0.75-0.75)^2 + (0.87-0.58)^2} = \sqrt{0.0169+0+0.0841} = \sqrt{0.101} = 0.32$$



Classe A

k-NN (k Nearest Neighbor)

- ❑ E se houvesse empate, o que fazer?
- ❑ No exemplo anterior, se $k = 4$, haveria um empate.
- ❑ Solução:
 - ❖ Ponderar a contribuição de cada um dos k vizinhos de acordo com a distância do ponto \mathbf{x}_q ;
 - ❖ Maior peso para os vizinhos mais próximos;
 - ❖ Como ficaria a resposta para o exemplo anterior?

k-NN (k Nearest Neighbor)

$$\begin{aligned} 1: & \sqrt{(0.5-0.67)^2 + (0.8-0.75)^2 + (0.9-0.58)^2} = \sqrt{0.0289+0.025+0.1024} = \sqrt{0.1338} = 0.37 \\ 2: & \sqrt{(0.4-0.67)^2 + (0.5-0.75)^2 + (0.4-0.58)^2} = \sqrt{0.0729+0.0625+0.0324} = \sqrt{0.1678} = 0.41 \\ 3: & \sqrt{(0.2-0.67)^2 + (0.5-0.75)^2 + (0.15-0.58)^2} = \sqrt{0.2209+0.2025+0.1849} = \sqrt{0.6083} = 0.68 \\ 4: & \sqrt{(0.5-0.67)^2 + (0.4-0.75)^2 + (0.6-0.58)^2} = \sqrt{0.0289+0.1225+0.0004} = \sqrt{0.1518} = 0.39 \\ 5: & \sqrt{(0.8-0.67)^2 + (0.75-0.75)^2 + (0.87-0.58)^2} = \sqrt{0.0169+0+0.0841} = \sqrt{0.101} = 0.32 \end{aligned}$$

k-NN (k Nearest Neighbor)

$$1: \sqrt{(0.5-0.67)^2 + (0.8-0.75)^2 + (0.9-0.58)^2} = \sqrt{0.0289+0.025+0.1024} = \sqrt{0.1338} = 0.37$$

$$2: \sqrt{(0.4-0.67)^2 + (0.5-0.75)^2 + (0.4-0.58)^2} = \sqrt{0.0729+0.0625+0.0324} = \sqrt{0.1678} = 0.41$$

$$3: \sqrt{(0.2-0.67)^2 + (0.5-0.75)^2 + (0.15-0.58)^2} = \sqrt{0.2209+0.2025+0.1849} = \sqrt{0.6083} = 0.68$$

$$4: \sqrt{(0.5-0.67)^2 + (0.4-0.75)^2 + (0.6-0.58)^2} = \sqrt{0.0289+0.1225+0.0004} = \sqrt{0.1518} = 0.39$$

$$5: \sqrt{(0.8-0.67)^2 + (0.75-0.75)^2 + (0.87-0.58)^2} = \sqrt{0.0169+0+0.0841} = \sqrt{0.101} = 0.32$$

Classe A

Classe B

Overview

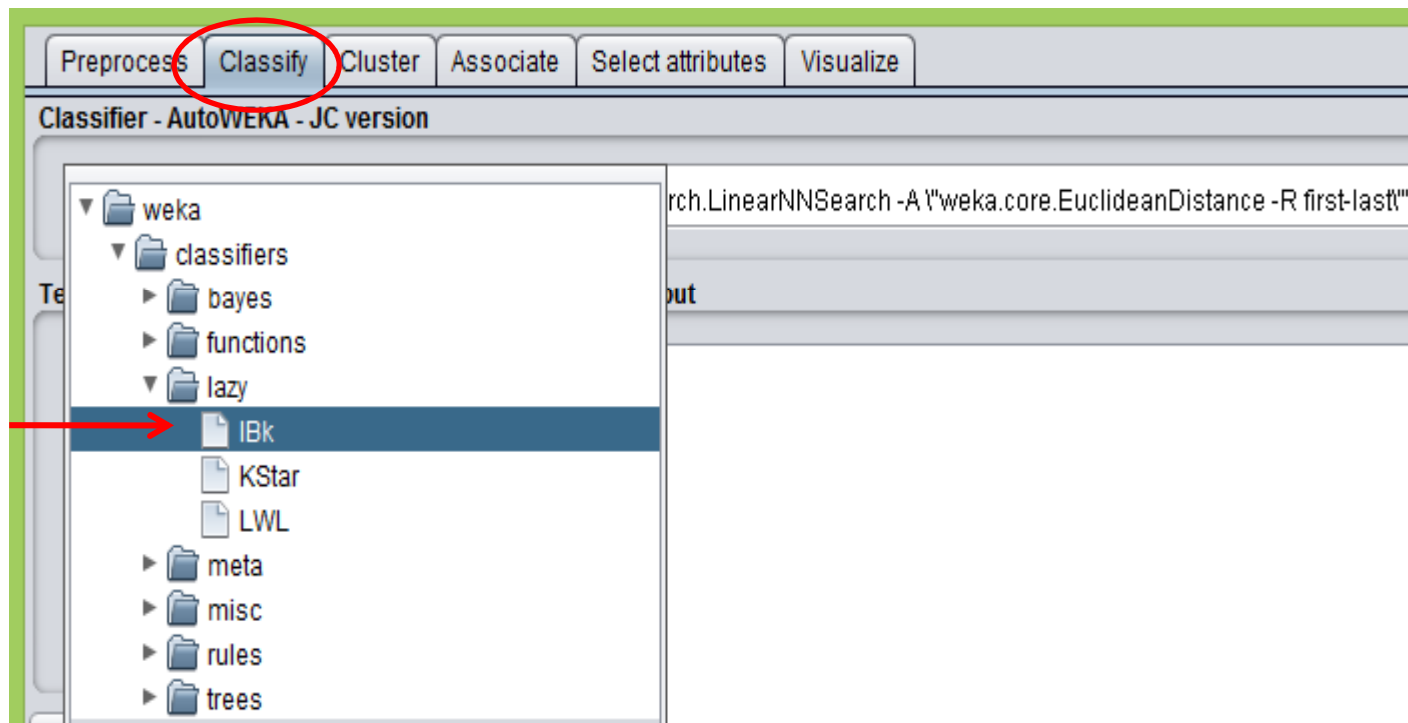
- ❑ Métodos de aprendizagem baseados em instâncias **não** necessitam formar uma **hipótese explícita** da função alvo sobre o espaço das instâncias.
- ❑ Eles formam uma aproximação local da função alvo para cada nova instância a “classificar”.
- ❑ O **k-NN** é um algoritmo baseado em instâncias para aproximar funções alvo de valor real ou de valor discreto, assumindo que as instâncias correspondem a pontos em um espaço d -dimensional.

Overview

- ❑ O valor da função alvo para um novo ponto é estimada a partir dos valores conhecidos dos k exemplos de treinamento mais próximos.
- ❑ **Vantagens:**
 - ❖ Habilidade para modelar funções alvo complexas por uma coleção de aproximações locais menos complexas.
 - ❖ A informação presente nos exemplos de treinamento nunca é perdida.
- ❑ **Dificuldades:**
 - ❖ Tempo?

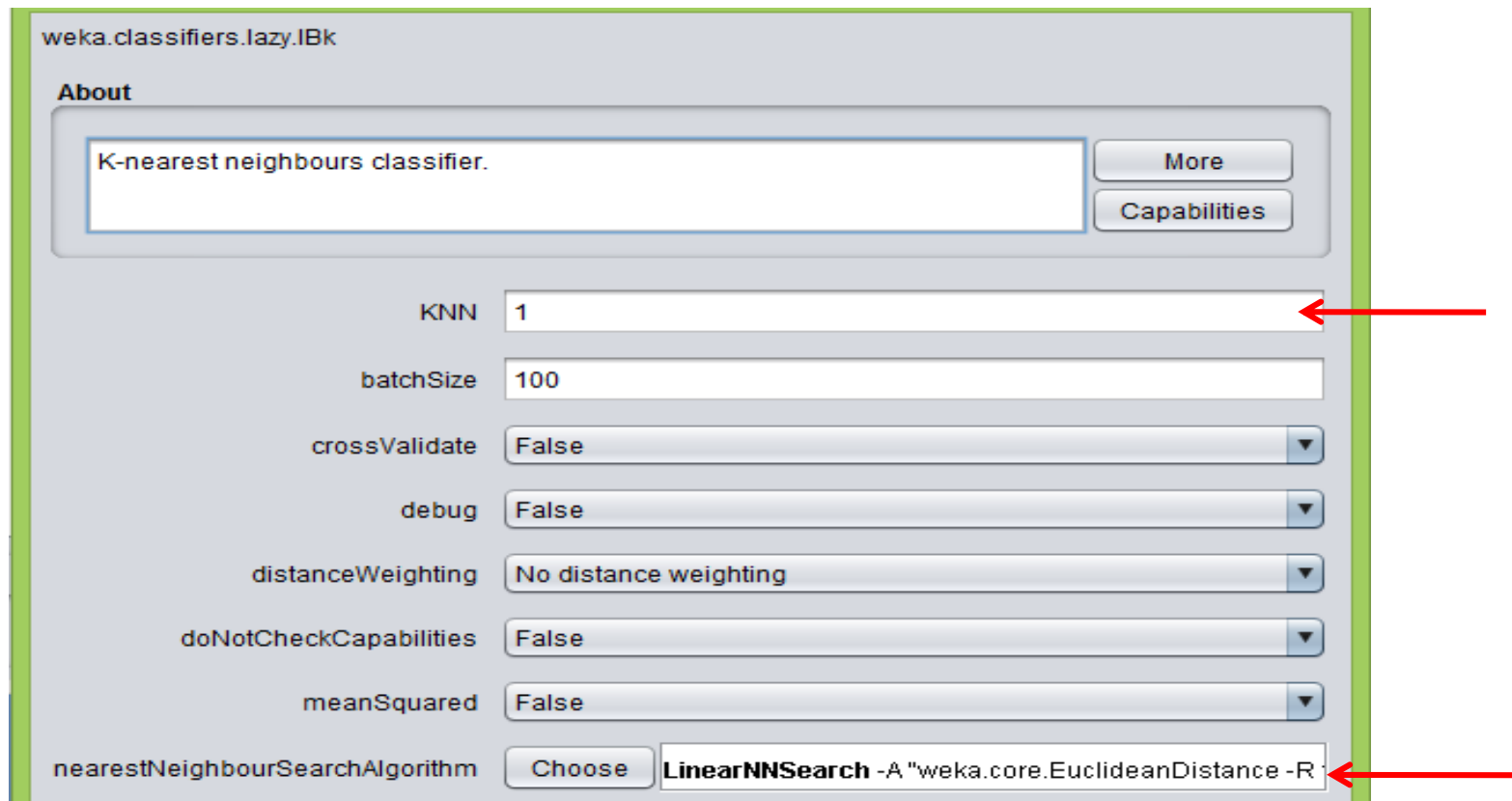
k-NN

- ❑ Utilizando **IBK** (WEKA):



k-NN

❑ Configurando o IBK:



weka.classifiers.lazy.IBk

About

K-nearest neighbours classifier.

More

Capabilities

KNN 1

batchSize 100

crossValidate False

debug False

distanceWeighting No distance weighting

doNotCheckCapabilities False

meanSquared False

nearestNeighbourSearchAlgorithm Choose LinearNNSearch -A "weka.core.EuclideanDistance -R

k-NN

❑ Abrindo o dataset Diabetes:

Current relation

Relation: pima_diabetes
Instances: 768

Attributes: 9
Sum of weights: 768

Attributes

All None Invert Pattern

No.	Name
1	<input checked="" type="checkbox"/> preg
2	<input type="checkbox"/> plas
3	<input type="checkbox"/> pres
4	<input type="checkbox"/> skin
5	<input type="checkbox"/> insu
6	<input type="checkbox"/> mass
7	<input type="checkbox"/> pedi
8	<input type="checkbox"/> age
9	<input type="checkbox"/> class

Remove

Selected attribute

Name: preg
Missing: 0 (0%)
Distinct: 17
Type: Numeric
Unique: 2 (0%)

Statistic	Value
Minimum	0
Maximum	17
Mean	3.845
StdDev	3.37

Class: class (Nom) Visualize All

Value	Frequency
0	246
1	103
2	75
3	125
4	50
5	45
6	66
7	24
8	11
9	19
10	2
11	1
12	1

<https://www.dropbox.com/sh/fhkqy2wybxjl0n5/AAABevgbnnM4HSdPgeUU6tgPa?dl=0>

k-NN

❑ Configurando treinamento e teste:

Test options


☐ Use training set

☐ Supplied test set Set...

☒ Cross-validation Folds 10

☐ Percentage split % 66

More options...



Test options


☐ Use training set

☐ Supplied test set Set...

☐ Cross-validation Folds 10

☒ Percentage split % 70

More options...



k-NN

❑ Analisando os resultados 1-NN:

```
Classifier output

=== Run information ===

Scheme:      weka.classifiers.lazy.IBk -K 1 -W 0 -A "weka.core.neighboursearch
Relation:    pima_diabetes
Instances:    768
Attributes:  9
              preg
              plas
              pres
              skin
              insu
              mass
              pedi
              age
              class
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

IB1 instance-based classifier
using 1 nearest neighbour(s) for classification
```


k-NN

□ Analisando os resultados 1-NN:

Classifier output

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances	539	70.1823 %
Incorrectly Classified Instances	229	29.8177 %
Kappa statistic	0.3304	
Mean absolute error	0.2988	
Root mean squared error	0.5453	
Relative absolute error	65.7327 %	
Root relative squared error	114.3977 %	
Total Number of Instances	768	

=== Detailed Accuracy By Class ===

	Precision	Recall	F-Measure	Class
	0,759	0,794	0,776	tested_negative
	0,580	0,530	0,554	tested_positive
Weighted Avg.	0,696	0,702	0,698	

=== Confusion Matrix ===

a	b	<-- classified as
397	103	a = tested_negative
126	142	b = tested_positive

k-NN

Analizando os resultados 3-NN:

```
Classifier output

=== Run information ===

Scheme:      weka.classifiers.lazy.IBk -K 3 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance\" -S 0"
Relation:    pima_diabetes
Instances:    768

Correctly Classified Instances      558      72.6563 %
Incorrectly Classified Instances    210      27.3438 %
Kappa statistic                    0.3822
Mean absolute error                 0.3092
Root mean squared error             0.4525
Relative absolute error             68.0324 %
Root relative squared error         94.9365 %
Total Number of Instances          768

=== Detailed Accuracy By Class ===
               Precision    Recall    F-Measure   Class
               0,774        0,820    0,796      tested_negative
               0,622        0,552    0,585      tested_positive
Weighted Avg.  0,721        0,727    0,722

=== Confusion Matrix ===
      a    b  <-- classified as
    410  90 |   a = tested_negative
    120 148 |   b = tested_positive
```

k-NN

❑ Analisando os resultados 1-NN:

```
Classifier output

=== Run information ===

Scheme:      weka.classifiers.lazy.IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.c
Relation:    pima_diabetes
Instances:    768
Test mode:    split 70.0% train, remainder test

=== Summary ===

Correctly Classified Instances      169      73.4783 %
Incorrectly Classified Instances    61      26.5217 %
Kappa statistic                     0.3903
Mean absolute error                  0.2661
Root mean squared error              0.514
Relative absolute error              59.2025 %
Root relative squared error          110.1644 %
Total Number of Instances           230

=== Detailed Accuracy By Class ===
               Precision  Recall  F-Measure  Class
0,813         0,797     0,805     tested_negative
0,573         0,597     0,585     tested_positive
Weighted Avg. 0,738         0,735     0,736

=== Confusion Matrix ===
      a  b  <-- classified as
126  32 |  a = tested_negative
 29  43 |  b = tested_positive
```

k-NN

❑ Analisando os resultados 3-NN:

```
Classifier output

=== Run information ===

Scheme:      weka.classifiers.lazy.IBk -K 3 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.Euclid
Relation:    pima_diabetes
Instances:    768
Test mode:   split 70.0% train, remainder test

=== Summary ===
Correctly Classified Instances      169      73.4783 %
Incorrectly Classified Instances    61      26.5217 %
Kappa statistic                    0.3616
Mean absolute error                 0.3104
Root mean squared error             0.4417
Relative absolute error             69.0573 %
Root relative squared error        94.6539 %
Total Number of Instances          230

=== Detailed Accuracy By Class ===
               Precision    Recall    F-Measure  Class
               0,790        0,835      0,812      tested_negative
               0,587        0,514      0,548      tested_positive
Weighted Avg. 0,727        0,735      0,730

=== Confusion Matrix ===
      a  b  <-- classified as
132  26 |  a = tested_negative
 35  37 |  b = tested_positive
```

Árvore de Decisão

Árvore de Decisão

❑ Definição:

- ❖ Um fluxograma com a estrutura de uma árvore.
- ❖ Nó interno representa um testes sobre um atributo.
- ❖ Cada ramo representa um resultado do teste.
- ❖ Folhas representam as classes.

❑ A geração de uma árvore consiste de duas fases:

- ❖ Construção da árvore (particionamento de atributos).
- ❖ Fase da poda (identifica e remove ruídos ou outliers).

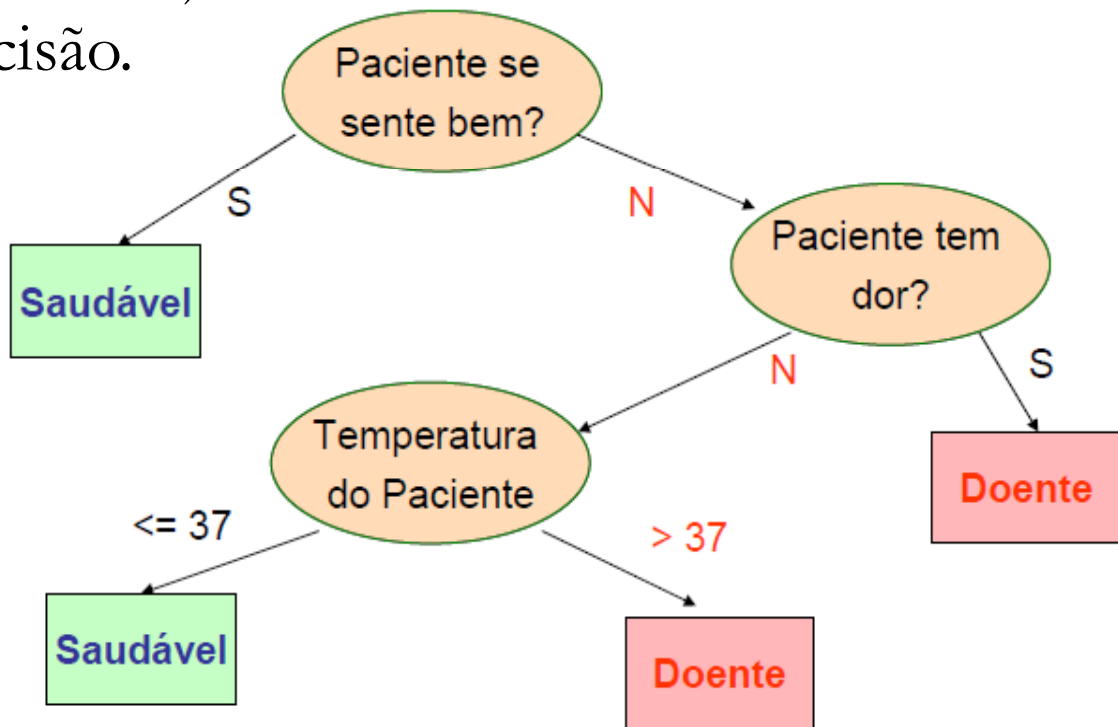
❑ Uso da árvore: classificação de amostras desconhecidas.

- ❖ Testa os valores dos atributos da amostra “contra” a árvore.

Árvore de Decisão

❑ Funcionamento:

- ❖ Lista de perguntas → respostas “**sim**” ou “**não**”.
- ❖ Hierarquicamente arranjadas.
- ❖ Levam a uma decisão.



Árvore de Decisão

❑ Geração de regras:

Se (paciente está bem = sim) então
 classe = **saudável**

Senão

 Se (paciente tem dor = sim) então
 classe = **doente**

 Fimse

Fimse

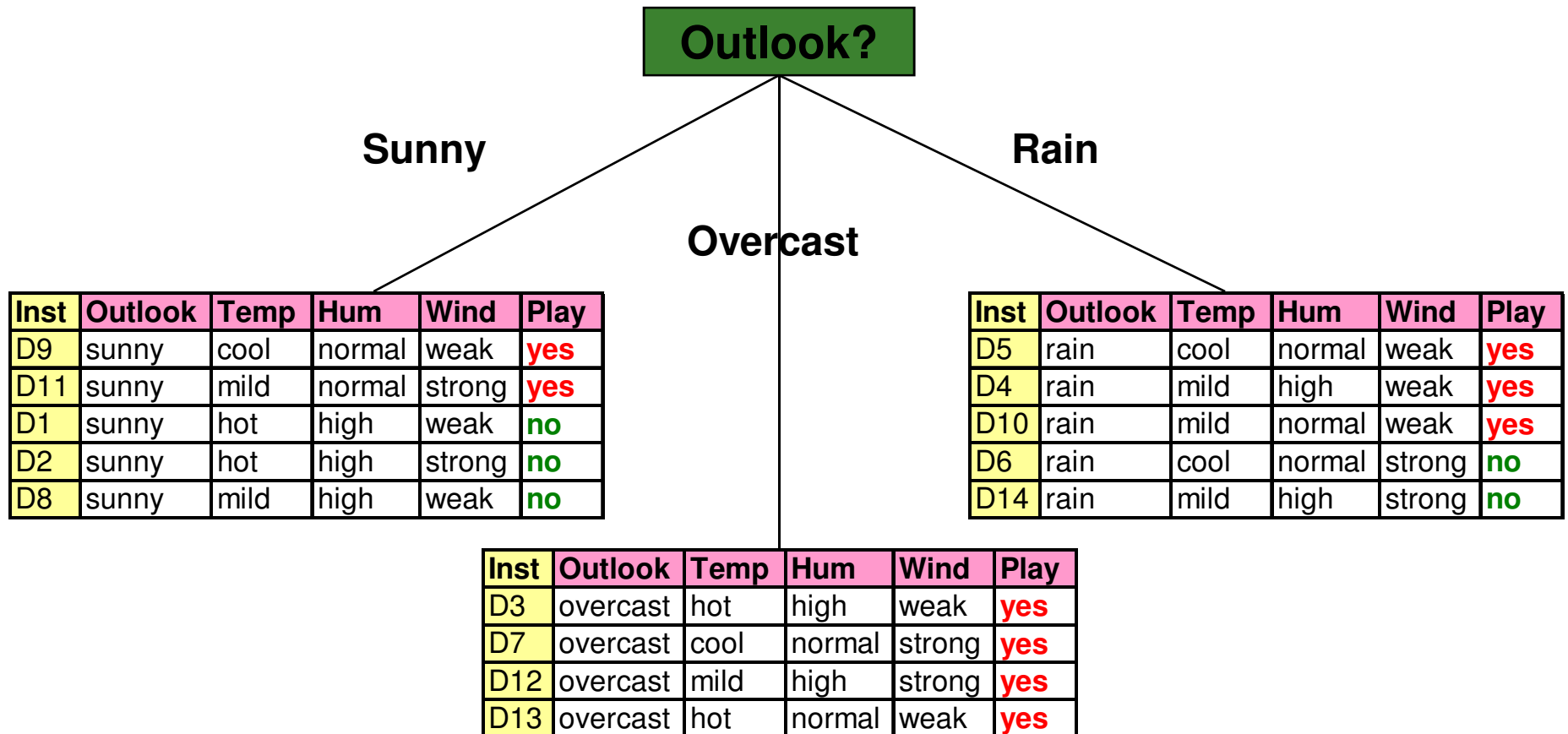
Árvore de Decisão

□ Treinamento:

Base de Dados “Tempo”

Instância	Outlook	Temperature	Humidity	Wind	Play
D1	sunny	hot	high	weak	no
D2	sunny	hot	high	strong	no
D3	overcast	hot	high	weak	yes
D4	rain	mild	high	weak	yes
D5	rain	cool	normal	weak	yes
D6	rain	cool	normal	strong	no
D7	overcast	cool	normal	strong	yes
D8	sunny	mild	high	weak	no
D9	sunny	cool	normal	weak	yes
D10	rain	mild	normal	weak	yes
D11	sunny	mild	normal	strong	yes
D12	overcast	mild	high	strong	yes
D13	overcast	hot	normal	weak	yes
D14	rain	mild	high	strong	no

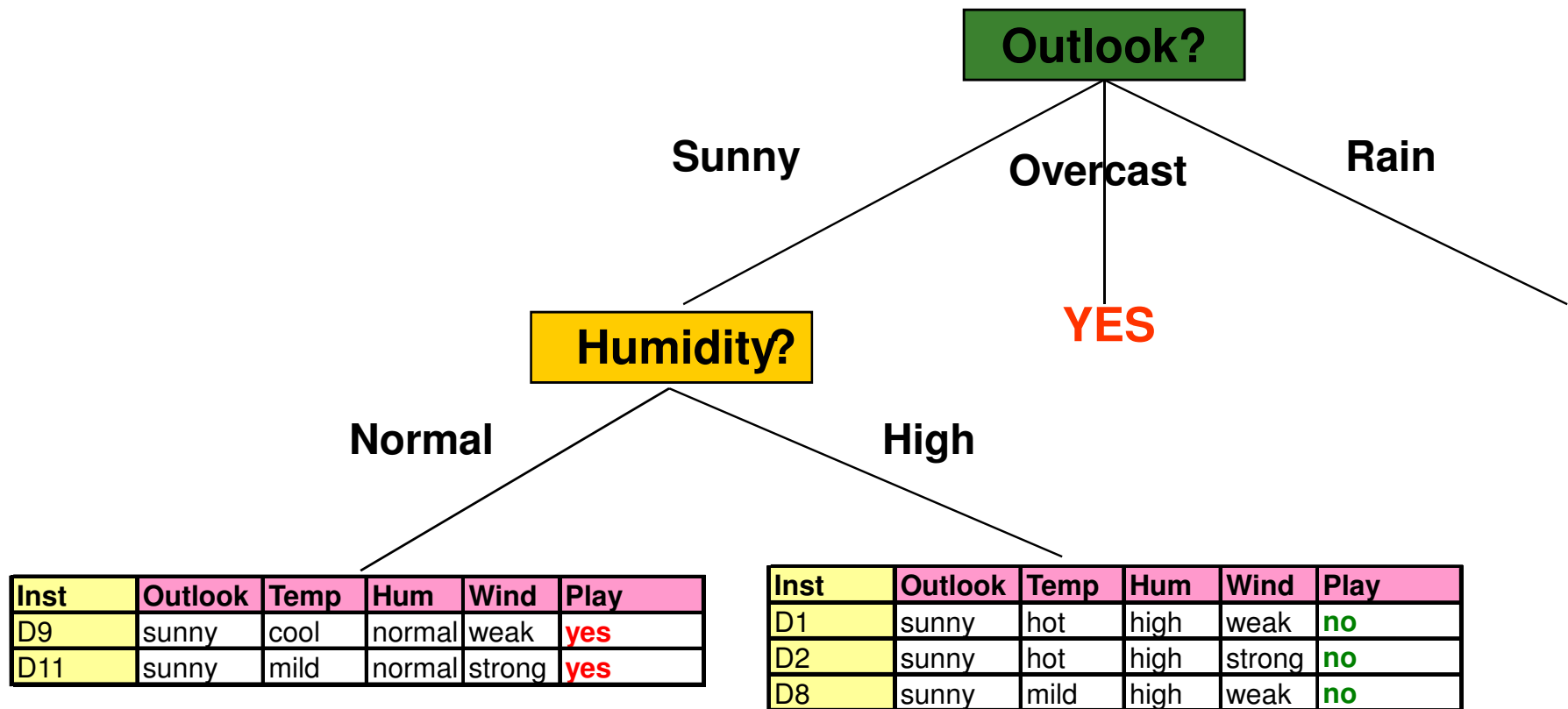
Árvore de Decisão



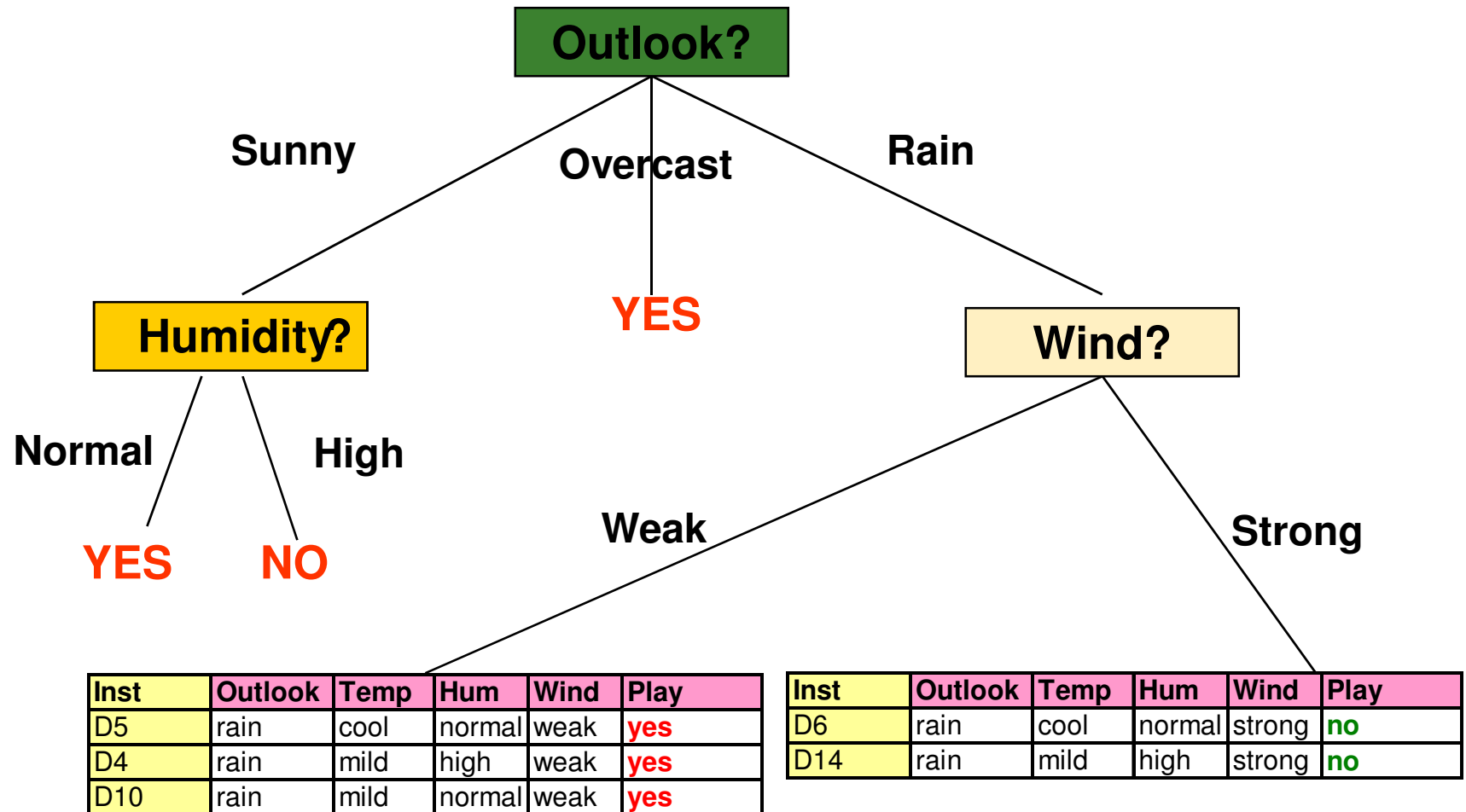
Árvore de Decisão

Teste	Exemplo	Outlook	Temperature	Humidity	Wind	Play?
If outlook=sunny	D1	Sunny	Hot	High	Weak	No
	D2	Sunny	Hot	High	Strong	No
	D8	Sunny	Mild	High	Weak	No
	D9	Sunny	Cool	Normal	Weak	Yes
	D11	Sunny	Mild	Normal	Strong	Yes
If outlook=overcast	D3	Overcast	Hot	High	Weak	Yes
	D7	Overcast	Cold	Normal	Strong	Yes
	D12	Overcast	Mild	High	Strong	Yes
	D13	Overcast	Hot	Normal	Weak	Yes
If outlook=rain	D4	Rain	Mild	High	Weak	Yes
	D5	Rain	Cool	Normal	Weak	Yes
	D6	Rain	Cool	Normal	Strong	No
	D10	Rain	Mild	Normal	Weak	Yes
	D14	Rain	Mild	High	Strong	No

Árvore de Decisão



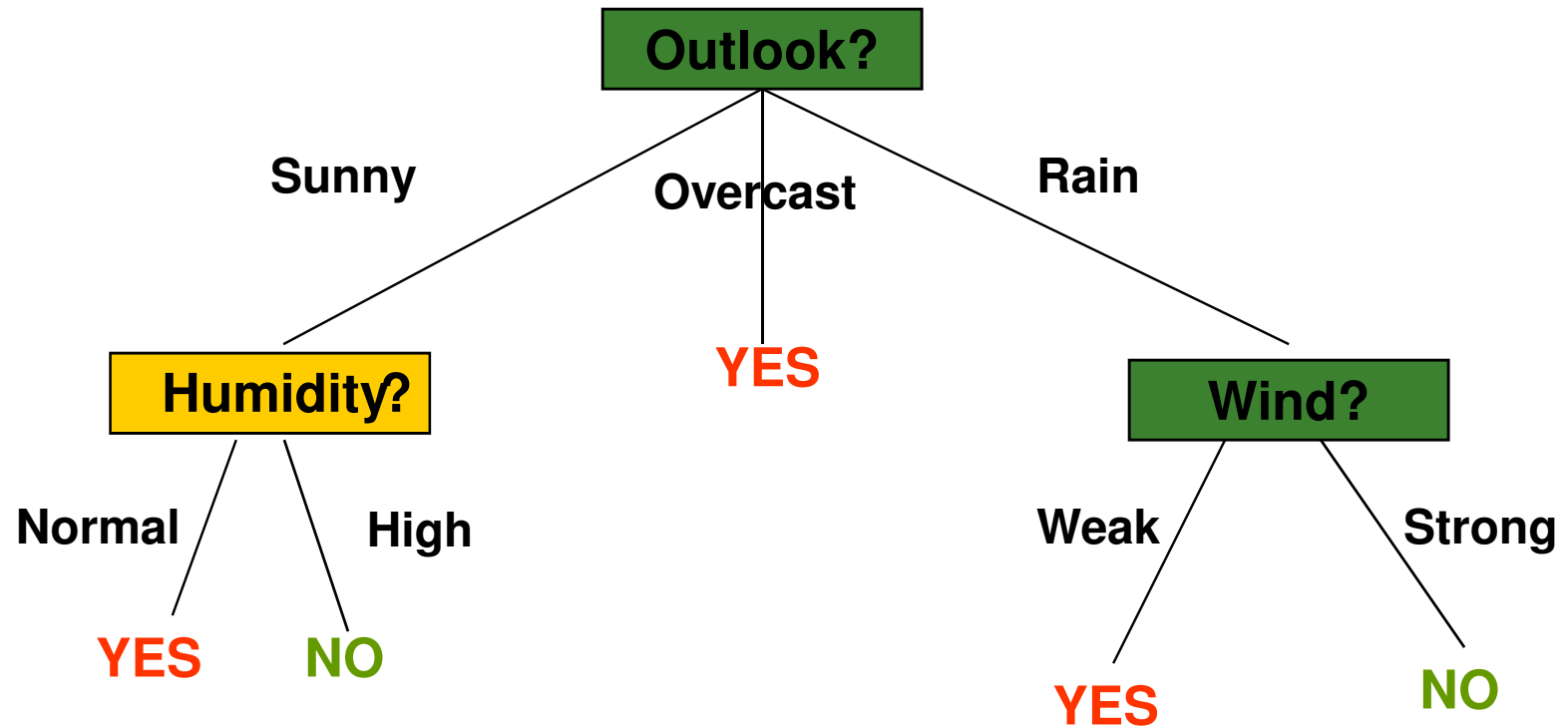
Árvore de Decisão



Árvore de Decisão

Teste	Exemplo	Outlook	Temperature	Humidity	Wind	Play?
If outlook=sunny and humidity=high	D1	Sunny	Hot	High	Weak	No
	D2	Sunny	Hot	High	Strong	No
	D8	Sunny	Mild	High	Weak	No
If outlook=sunny and humidity=nomal	D9	Sunny	Cool	Normal	Weak	Yes
	D11	Sunny	Mild	Normal	Strong	Yes
If outlook=overcast	D3	Overcast	Hot	High	Weak	Yes
	D7	Overcast	Cold	Normal	Strong	Yes
	D12	Overcast	Mild	High	Strong	Yes
	D13	Overcast	Hot	Normal	Weak	Yes
If outlook=rain and wind=strong	D6	Rain	Cool	Normal	Strong	No
	D14	Rain	Mild	High	Strong	No
If outlook=rain and wind=weak	D4	Rain	Mild	High	Weak	Yes
	D5	Rain	Cool	Normal	Weak	Yes
	D10	Rain	Mild	Normal	Weak	Yes

Árvore de Decisão



Árvore de Decisão

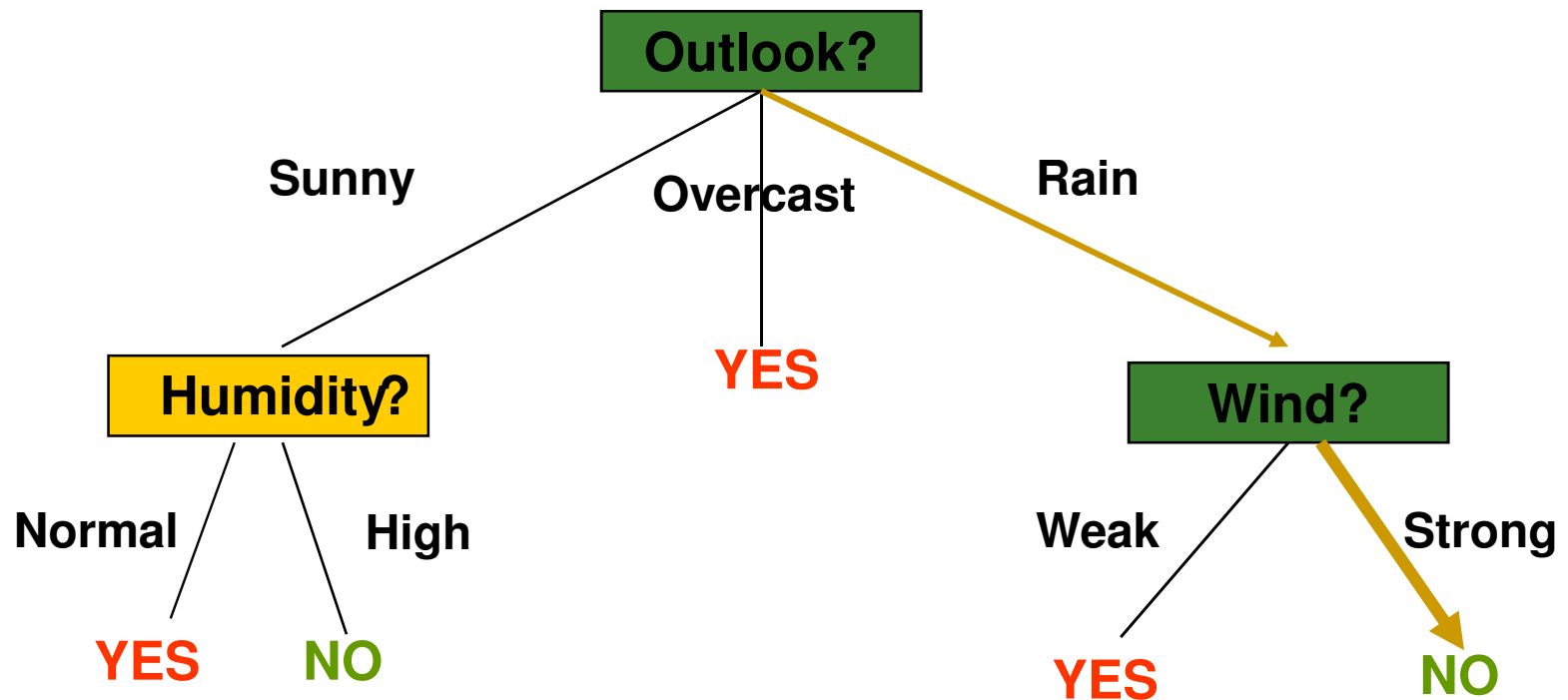
Teste	Exemplo	Outlook	Temperature	Humidity	Wind	Play?
If outlook=sunny and humidity=high	D1	Sunny	Hot	High	Weak	No
	D2	Sunny	Hot	High	Strong	No
	D8	Sunny	Mild	High	Weak	No
If outlook=sunny and humidity=nomal	D9	Sunny	Cool	Normal	Weak	Yes
	D11	Sunny	Mild	Normal	Strong	Yes
If outlook=overcast	D3	Overcast	Hot	High	Weak	Yes
	D7	Overcast	Cold	Normal	Strong	Yes
	D12	Overcast	Mild	High	Strong	Yes
	D13	Overcast	Hot	Normal	Weak	Yes
If outlook=rain and wind=strong	D6	Rain	Cool	Normal	Strong	No
	D14	Rain	Mild	High	Strong	No
If outlook=rain and wind=weak	D4	Rain	Mild	High	Weak	Yes
	D5	Rain	Cool	Normal	Weak	Yes
	D10	Rain	Mild	Normal	Weak	Yes

Árvore de Decisão

❑ Como classificar a seguinte instância:

@data

rainy, hot, normal, strong, ???



Árvore de Decisão

❑ Algoritmo Básico:

- ❖ A árvore é construída recursivamente no sentido top-down (divisão para conquista).
- ❖ Os atributos são **nominais** (se numéricos, eles são **discretizados**).
- ❖ Atributos “**testes**” são selecionados com base em heurísticas ou medidas estatísticas (ex., **ganho de informação**).

❑ Condições de parada do particionamento:

- ❖ Todas as amostras de um nó pertencem a **mesma classe**.
- ❖ Não existem mais atributos para particionamento.

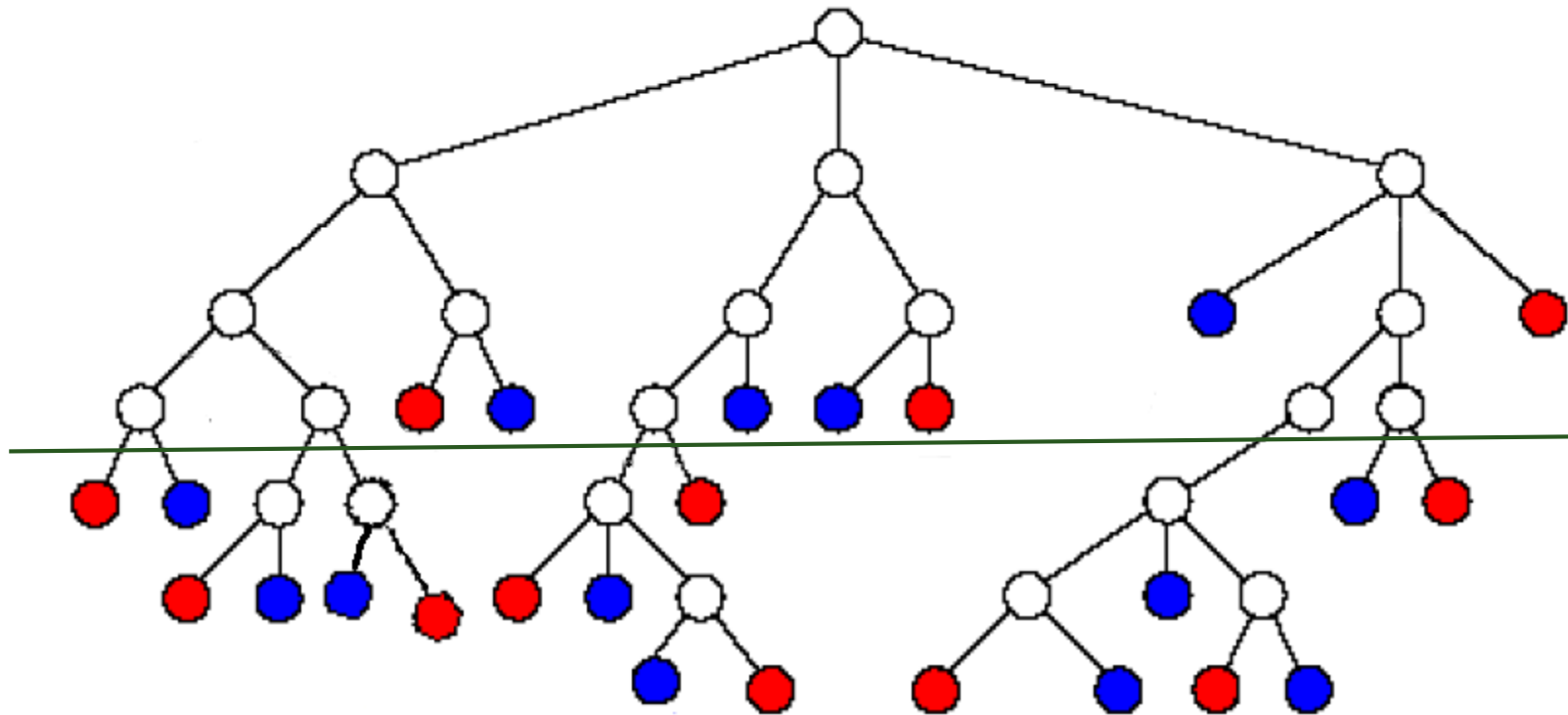
Árvore de Decisão

□ Poda:

- ❖ Técnica para lidar com ruído e “**Overfitting**”.
- ❖ **Pré-Poda**: durante a geração da Hipótese.
- ❖ **Pós-Poda**: é gerado um **Classificador** que explique os exemplos.
 - Após isso, elimina-se algumas partes (**cortes em ramos da árvore**) generalizando a Hipótese.

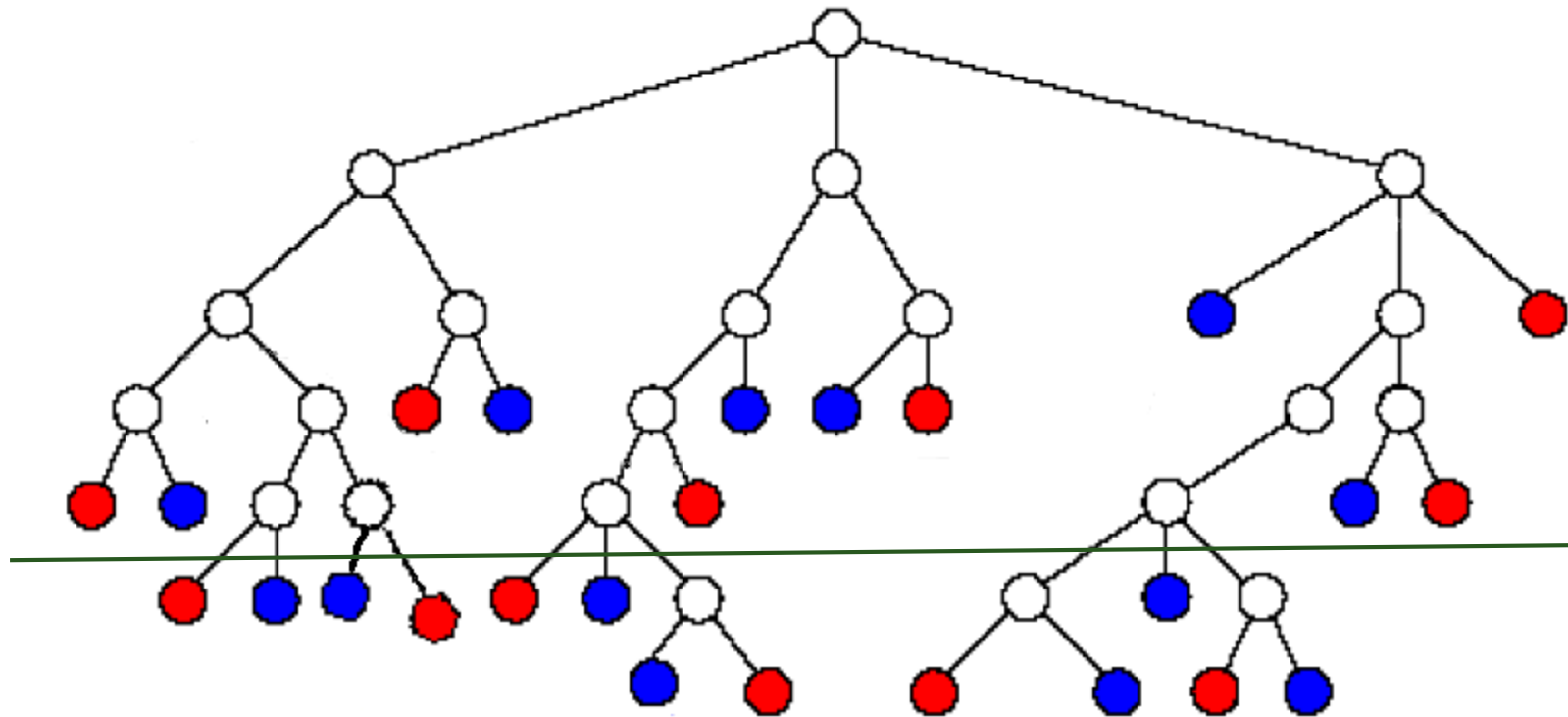
Árvore de Decisão

□ Poda:



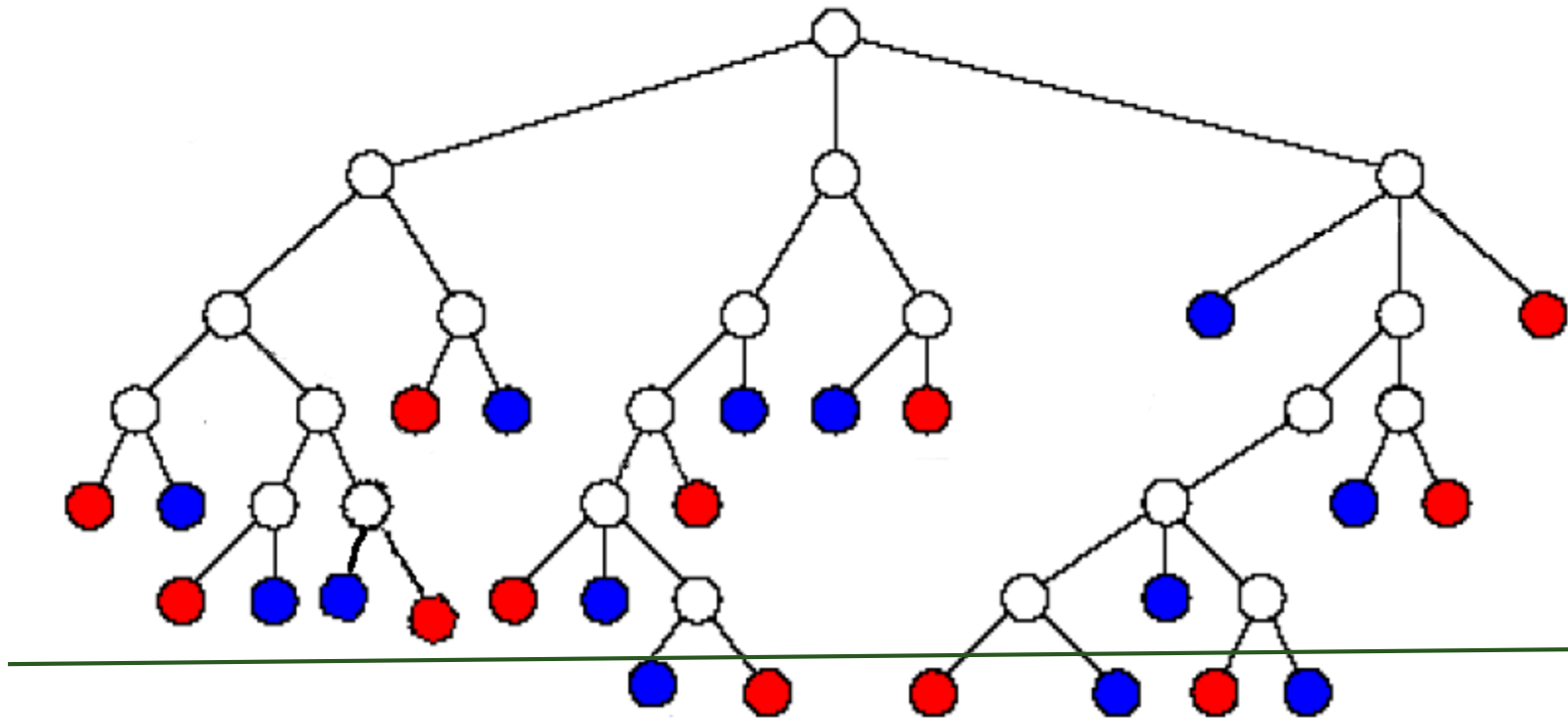
Árvore de Decisão

□ Poda:



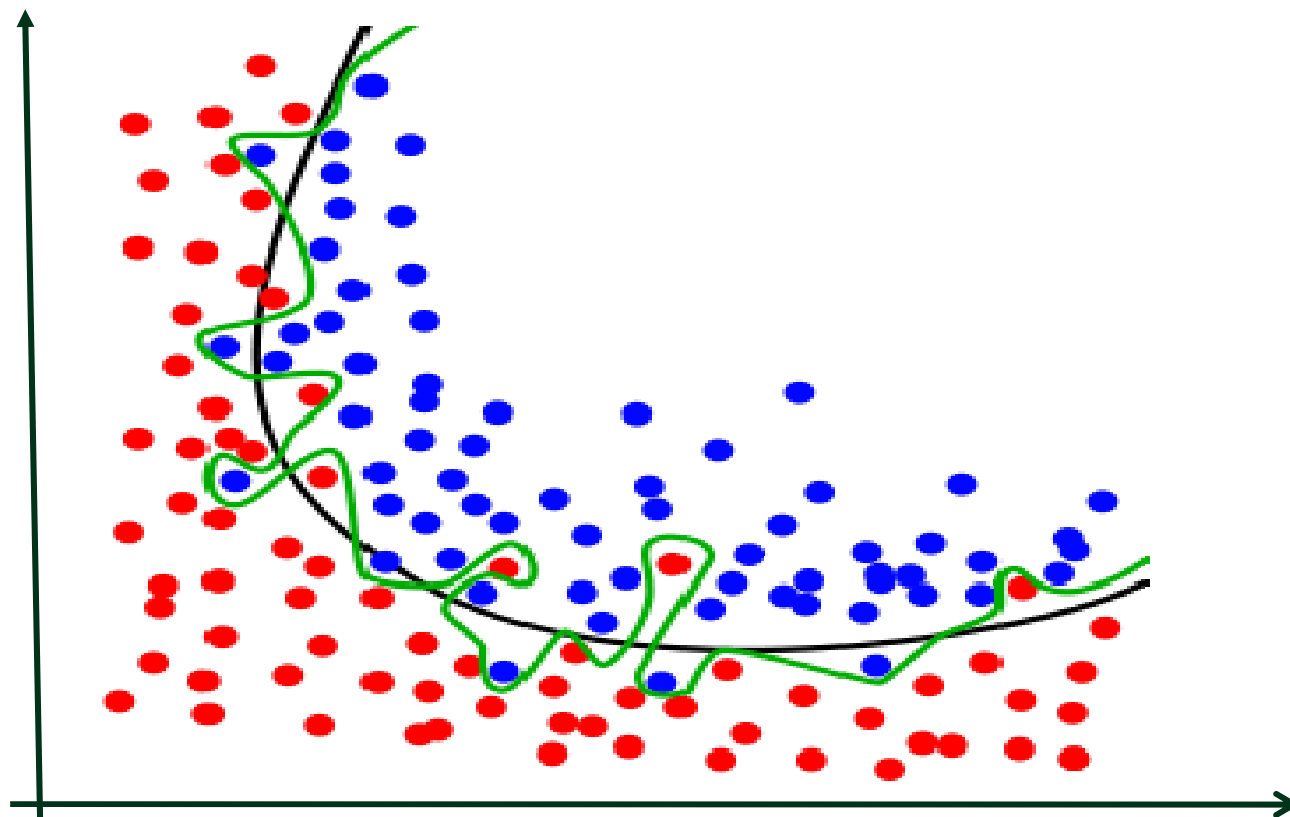
Árvore de Decisão

□ Poda:



Árvore de Decisão

❑ Overfitting:



Algoritmos mais conhecidos

- ❑ **ID3 (Iterative Dichotomiser 3) (Quilan,1986):**
 - Os atributos devem ser obrigatoriamente categóricos.
- ❑ **C4.5 (J48 no Weka) (Quilan, 1993):**
 - Um algoritmo para geração de árvores de decisão, sucessor do algoritmo ID3.
 - Considera atributos numéricos e categóricos.
- ❑ **CART (Classification And Regression Trees) (Breiman et al., 1984):**
 - Produz árvores de classificação ou regressão, dependendo se as variáveis são categóricas ou numéricas.

Overview

❑ Vantagens:

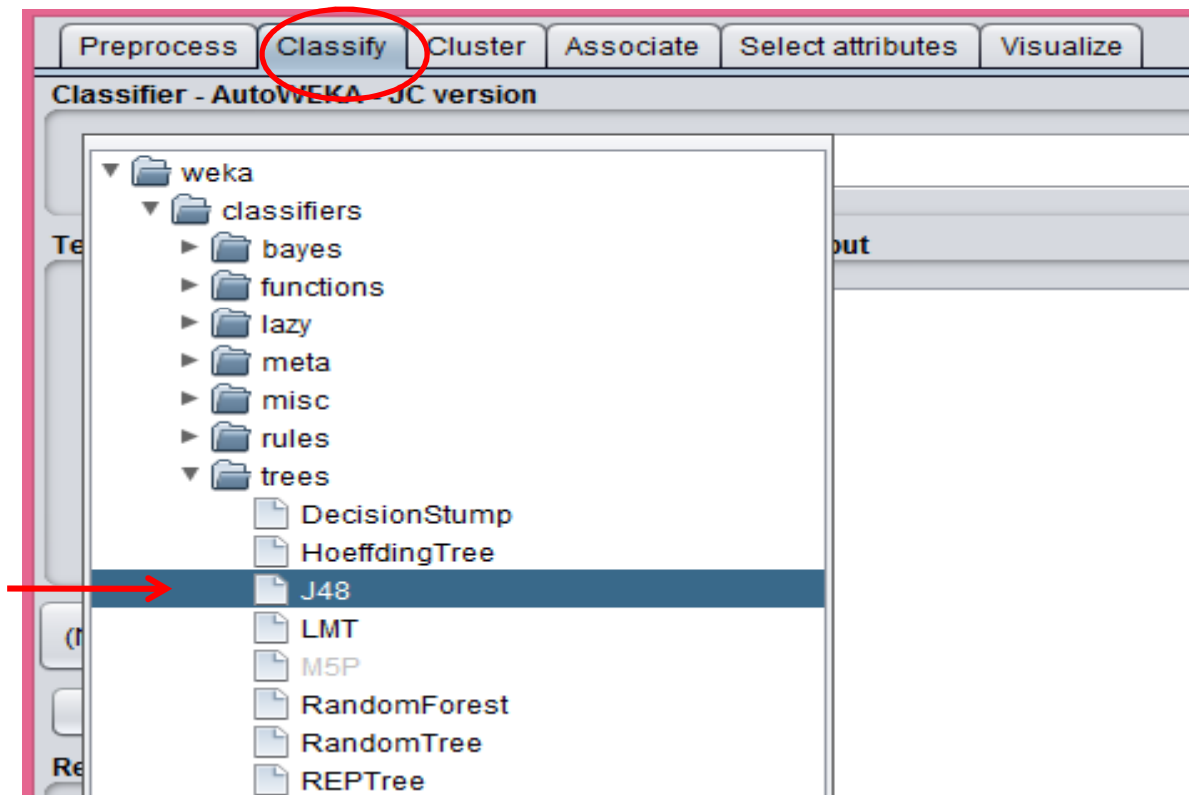
- Estrutura de fácil manipulação;
- Produzem modelos que podem ser facilmente interpretados por humanos;
- Muito rápido para classificar amostras desconhecidas.

❑ Desvantagens:

- Pouca robustez a dados de grande dimensão;
- Acurácia afetada por **atributos pouco relevantes**;
- Dificuldade em lidar **com dados contínuos**.

Árvore de Decisão

- ❑ Utilizando **J48** (WEKA):



Árvore de Decisão

❑ Configurando o J48:

weka.classifiers.trees.J48

About

Class for generating a pruned or unpruned C4. More
Capabilities

batchSize 100

binarySplits False

collapseTree True

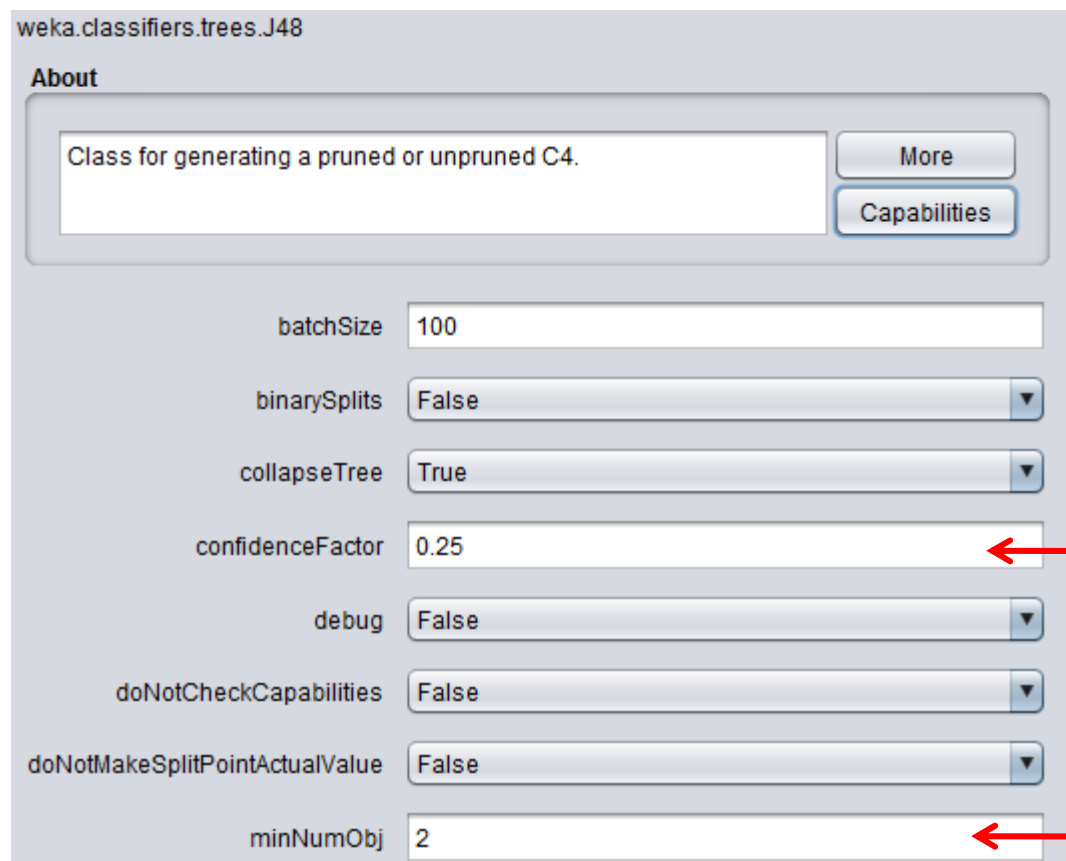
confidenceFactor 0.25

debug False

doNotCheckCapabilities False

doNotMakeSplitPointActualValue False

minNumObj 2



Árvore de Decisão

❑ Analisando os resultados J48:

```
Classifier output

=== Run information ===

Scheme:      weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:    pima_diabetes
Instances:   768
Attributes:  9
              preg
              plas
              pres
              skin
              insu
              mass
              pedi
              age
              class
Test mode:   10-fold cross-validation
```

Árvore de Decisão

☐ Analisando os resultados J48:

```
Classifier output

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances      567      73.8281 %
Incorrectly Classified Instances    201      26.1719 %
Kappa statistic                     0.4164
Mean absolute error                  0.3158
Root mean squared error              0.4463
Relative absolute error              69.4841 %
Root relative squared error          93.6293 %
Total Number of Instances           768

=== Detailed Accuracy By Class ===
                Precision    Recall    F-Measure    Class
                0,790        0,814        0,802        tested_negative
                0,632        0,597        0,614        tested_positive
Weighted Avg.  0,735        0,738        0,736

=== Confusion Matrix ===
  a  b  <-- classified as
407 93 |  a = tested_negative
108 160 | b = tested_positive
```

Árvore de Decisão

☐ Analisando os resultados J48:

Classifier output

=== Summary ===

Correctly Classified Instances	176	76.5217 %
Incorrectly Classified Instances	54	23.4783 %
Kappa statistic	0.4889	
Mean absolute error	0.3206	←
Root mean squared error	0.4239	
Relative absolute error	71.3381 %	
Root relative squared error	90.8521 %	
Total Number of Instances	230	

=== Detailed Accuracy By Class ===

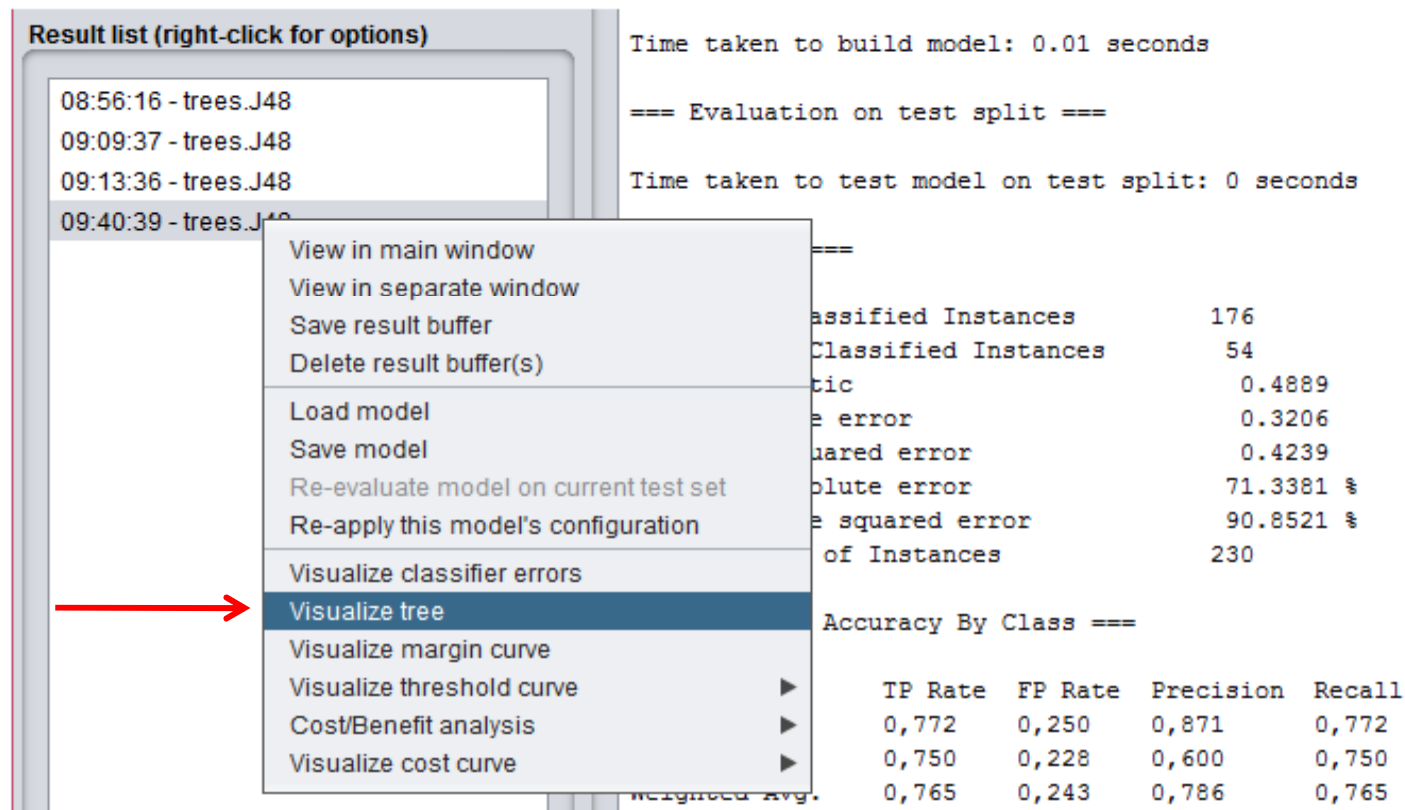
	Precision	Recall	F-Measure	Class
	0,871	0,772	0,819	tested_negative
	0,600	0,750	0,667	tested_positive
Weighted Avg	0,786	0,765	0,771	

=== Confusion Matrix ===

a	b	<-- classified as
122	36	a = tested_negative
18	54	b = tested_positive

Árvore de Decisão

❑ Visualizando a árvore....



The screenshot displays a software interface for decision tree analysis. On the left, a 'Result list (right-click for options)' window shows a list of results with timestamps and filenames (e.g., '08:56:16 - trees.J48'). A right-click context menu is open over this list, with a red arrow pointing to the 'Visualize tree' option. The menu includes options like 'View in main window', 'View in separate window', 'Save result buffer', 'Delete result buffer(s)', 'Load model', 'Save model', 'Re-evaluate model on current test set', 'Re-apply this model's configuration', 'Visualize classifier errors', 'Visualize tree', 'Visualize margin curve', 'Visualize threshold curve', 'Cost/Benefit analysis', and 'Visualize cost curve'. On the right, the main window displays model performance metrics. It shows 'Time taken to build model: 0.01 seconds' and 'Time taken to test model on test split: 0 seconds'. Below this, it states '=== Evaluation on test split ==='. A table of metrics follows, including 'Classified Instances' (176), 'Unclassified Instances' (54), 'Entropy' (0.4889), 'Information Error' (0.3206), 'Squared Error' (0.4239), 'Absolute Error' (71.3381 %), 'Mean Squared Error' (90.8521 %), and 'Number of Instances' (230). At the bottom, it shows 'Accuracy By Class ===' with a table of TP Rate, FP Rate, Precision, and Recall for each class.

Accuracy By Class ===	
TP Rate	FP Rate
0,772	0,250
0,750	0,228
0,765	0,243

Precision	Recall
0,871	0,772
0,600	0,750
0,786	0,765

Árvore de Decisão

📄 Analisando os resultados J48:

```

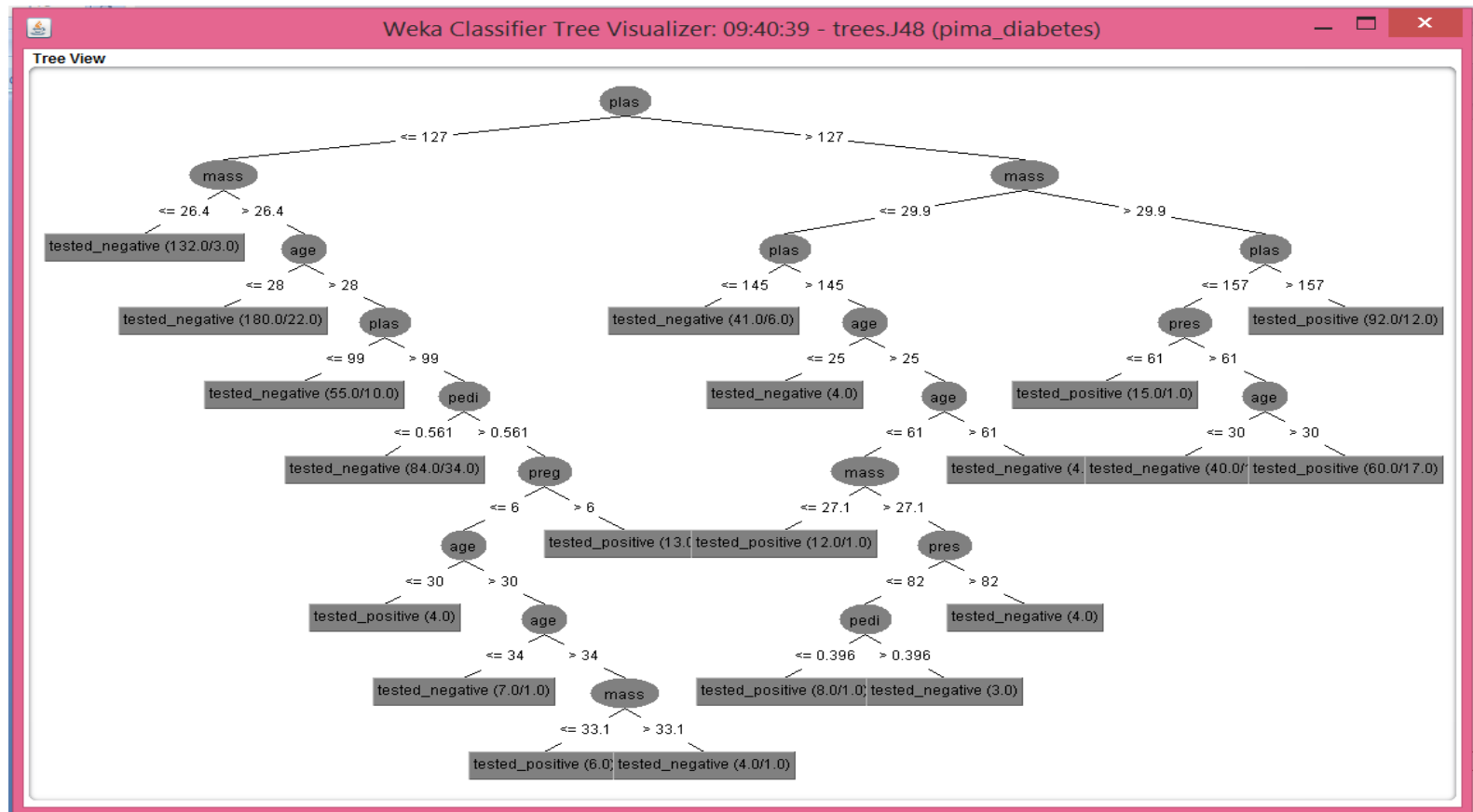
plas <= 127
| mass <= 26.4: tested_negative (132.0/3.0)
| mass > 26.4
| | age <= 28: tested_negative (180.0/22.0)
| | age > 28
| | | plas <= 99: tested_negative (55.0/10.0)
| | | plas > 99
| | | | pedi <= 0.561: tested_negative (84.0/34.0)
| | | | pedi > 0.561
| | | | | preg <= 6
| | | | | age <= 30: tested_positive (4.0)
| | | | | age > 30
| | | | | | age <= 34: tested_negative (7.0/1.0)
| | | | | | age > 34
| | | | | | mass <= 33.1: tested_positive (6.0)
| | | | | | mass > 33.1: tested_negative (4.0/1.0)
| | | | | preg > 6: tested_positive (13.0)
plas > 127
| mass <= 29.9
| | plas <= 145: tested_negative (41.0/6.0)
| | plas > 145
| | | age <= 25: tested_negative (4.0)
| | | age > 25
| | | | age <= 61
| | | | mass <= 27.1: tested_positive (12.0/1.0)
| | | | mass > 27.1
| | | | | pres <= 82
| | | | | | pedi <= 0.396: tested_positive (8.0/1.0)
| | | | | | pedi > 0.396: tested_negative (3.0)
| | | | | pres > 82: tested_negative (4.0)
| | | | age > 61: tested_negative (4.0)
| mass > 29.9
| | plas <= 157
| | | pres <= 61: tested_positive (15.0/1.0)
| | | pres > 61

```

- Number of Leaves : 20
- Size of the tree : 39

Árvore de Decisão

Visualizando a árvore....



Dúvidas ...

