Aula 05 - Árvores

1001524 – Aprendizado de Máquina I
2023/1 - Turmas A, B e C

Prof. Dr. Murilo Naldi

naldi@ufscar.br

Agradecimentos

- Parte do material utilizado nesta aula foi cedido pelos professores André Carvalho, Ricardo Campello, Diego Silva e Alan Valejo
- Parte do material utilizado nesta aula foi disponibilizado por M. Kumar no endereço:
  - www-users.cs.umn.edu/~kumar/dmbook/index.php
- Agradecimentos a Intel Software e a

  Intel IA Academy pelo material

  disponibilizado e recursos didáticos

Aulas Anteriores

- Visualização e estatísticas de resumo
- Pré-processamento
- Classificação e Validação de resultados

Conteúdo

- Resumo
  - Usando Árvores de Decisão
  - Gerando um modelo
    - Medida de Impureza
    - Tipos de atributos
    - Poda
  - Relação com Regras de Decisão

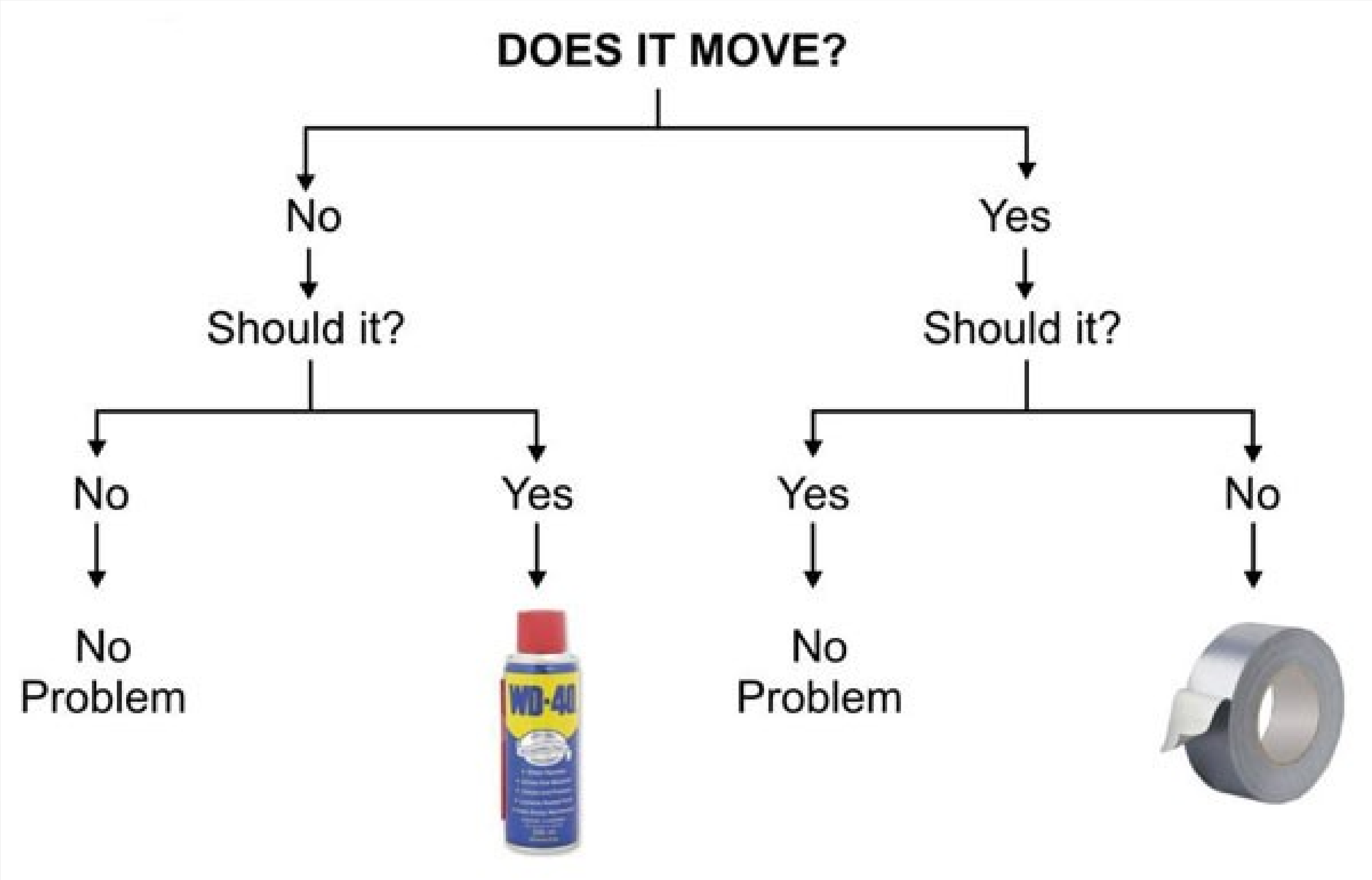
Árvores de Decisão (ADs)

- São algoritmos de classificação e tomada de decisão que utilizam a estratégia de divisão e conquista:
  - Divide problemas difíceis em problemas mais simples
  - Problema complexo é decomposto em sub-problemas menores
  - Estratégia é aplicada recursivamente a cada subproblema

Árvores de Decisão (ADs)

- Uma das técnicas mais utilizadas
  - Eficaz, eficiente e produz modelos interpretáveis
- Árvore é composta por:
  - Nó raiz
    - Nenhuma aresta de entrada e $n \geq 0$ arestas de saída
  - Nós intermediários
    - 1 aresta de entrada e $n > 1$ arestas de saída
  - Nós folhas (terminais)
    - 1 aresta de entrada e nenhuma aresta de saída

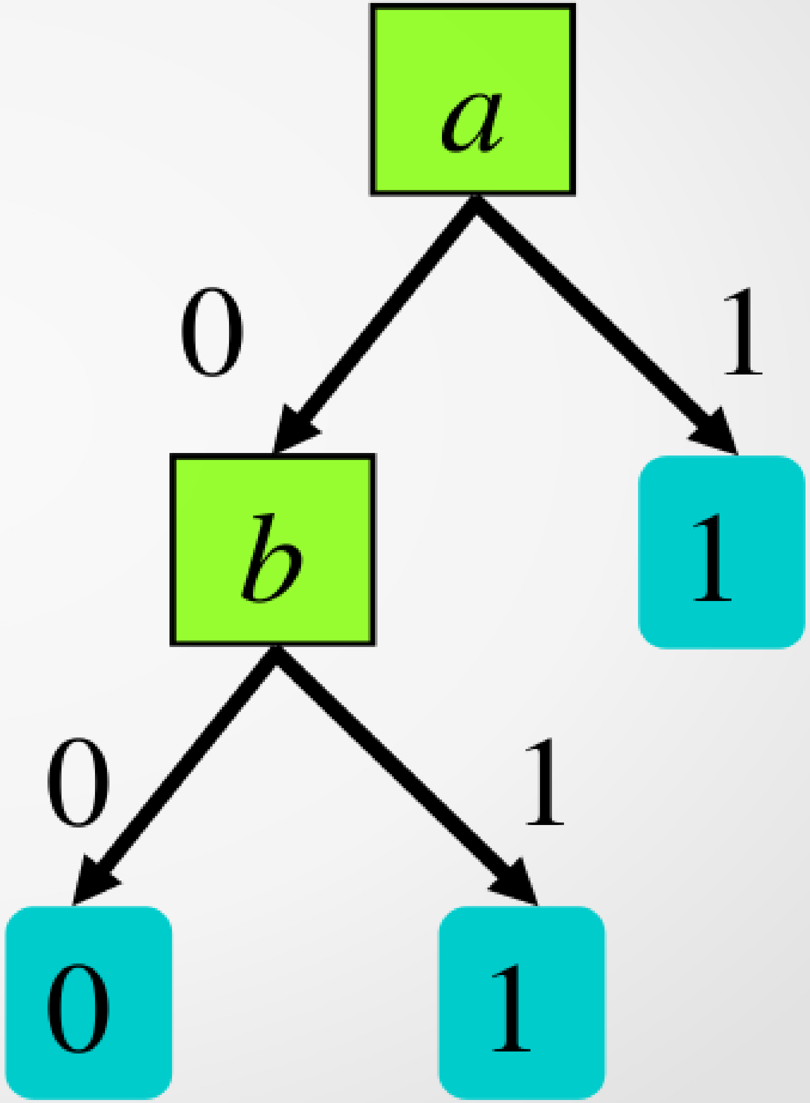
Exemplo simples



Outro exemplo

a OR b

a	b	a v b
0	0	0
0	1	1
1	0	1
1	1	1



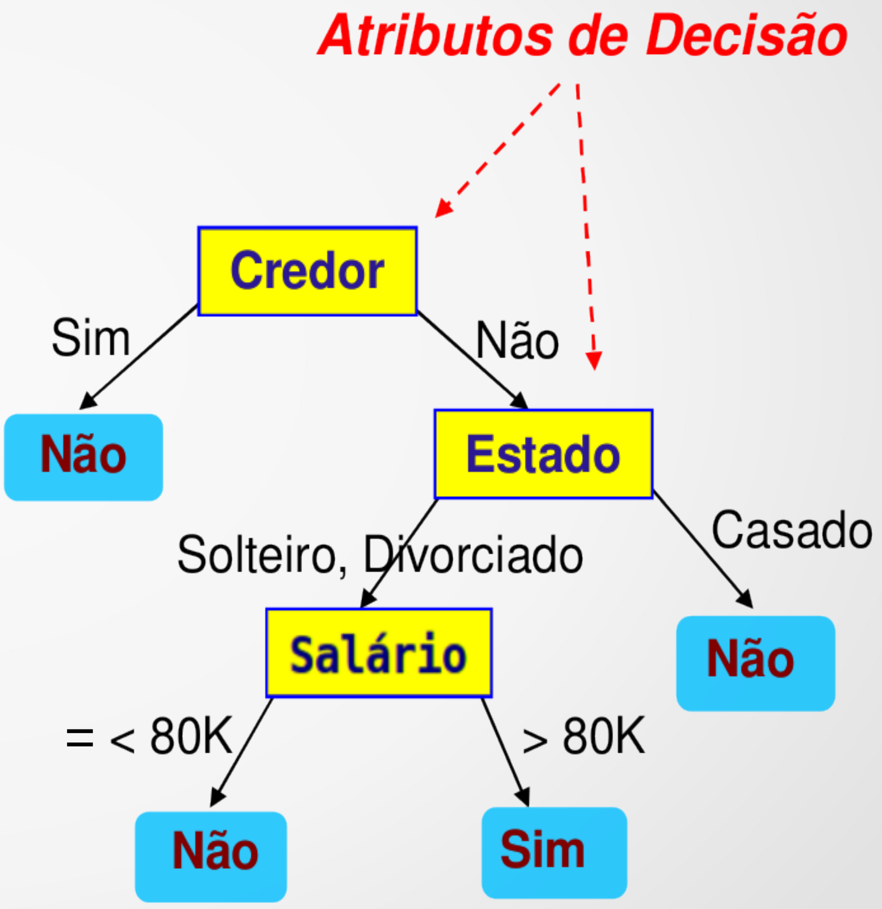
Exercícios

- Encontrar árvore de decisão para:
  - A AND b
  - A XOR b
  - (a AND b) OR (b AND c)

Atributos de decisão

Id	É Credor	Estado Civil	Salário	Calote
1	Sim	Solteiro	125K	Não
2	Não	Casado	100K	Não
3	Não	Solteiro	70K	Não
4	Sim	Casado	120K	Não
5	Não	Divorciado	95K	Sim
6	Não	Casado	60K	Não
7	Sim	Divorciado	220K	Não
8	Não	Solteiro	85K	Sim
9	Não	Casado	75K	Não
10	Não	Solteiro	90K	Sim

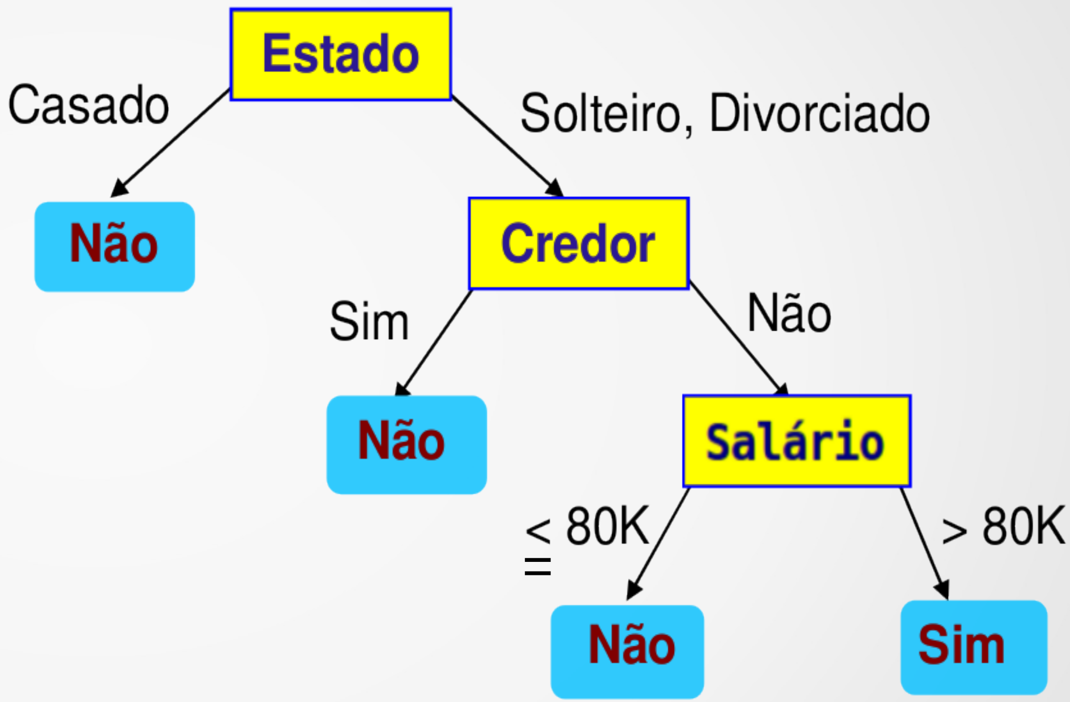
Dados de Treinamento



Modelo: Árvore de Decisão

Outro exemplo

Id	*É Credor*	*Estado Civil*	*Salário*	*Calote*
1	Sim	Solteiro	125K	Não
2	Não	Casado	100K	Não
3	Não	Solteiro	70K	Não
4	Sim	Casado	120K	Não
5	Não	Divorciado	95K	Sim
6	Não	Casado	60K	Não
7	Sim	Divorciado	220K	Não
8	Não	Solteiro	85K	Sim
9	Não	Casado	75K	Não
10	Não	Solteiro	90K	Sim



Diferentes árvores podem ser ajustadas para os mesmos dados !

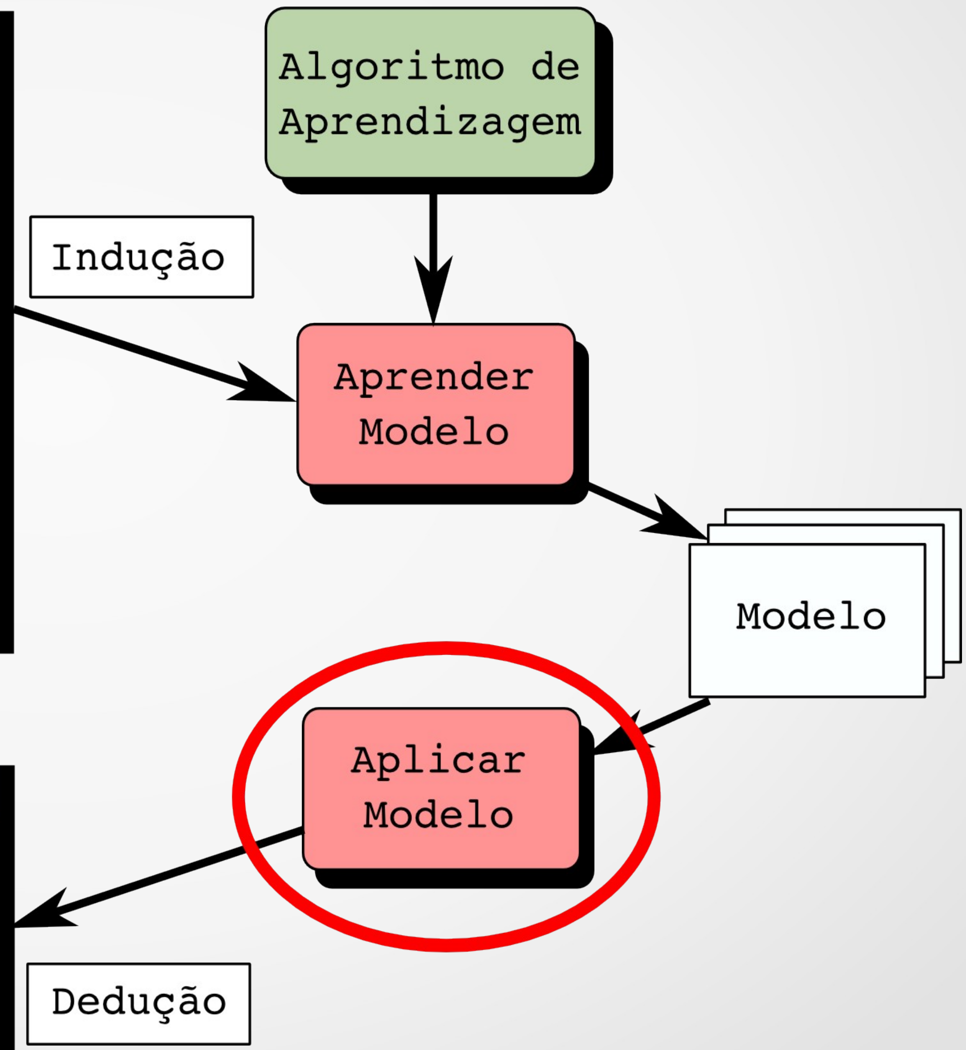
Aplicar Modelo

Conjunto de Treinamento

ID	Atrib.1	Atrib.2	Atrib.3	Classe
1	Sim	Grande	125K	Não
2	Não	Médio	100K	Não
3	Não	Pequeno	70K	Não
4	Sim	Médio	120K	Não
5	Não	Grande	95K	Sim
6	Não	Médio	60K	Não
7	Sim	Grande	220K	Não
8	Não	Pequeno	85K	Sim
9	Não	Médio	75K	Não
10	Não	Pequeno	90K	Sim

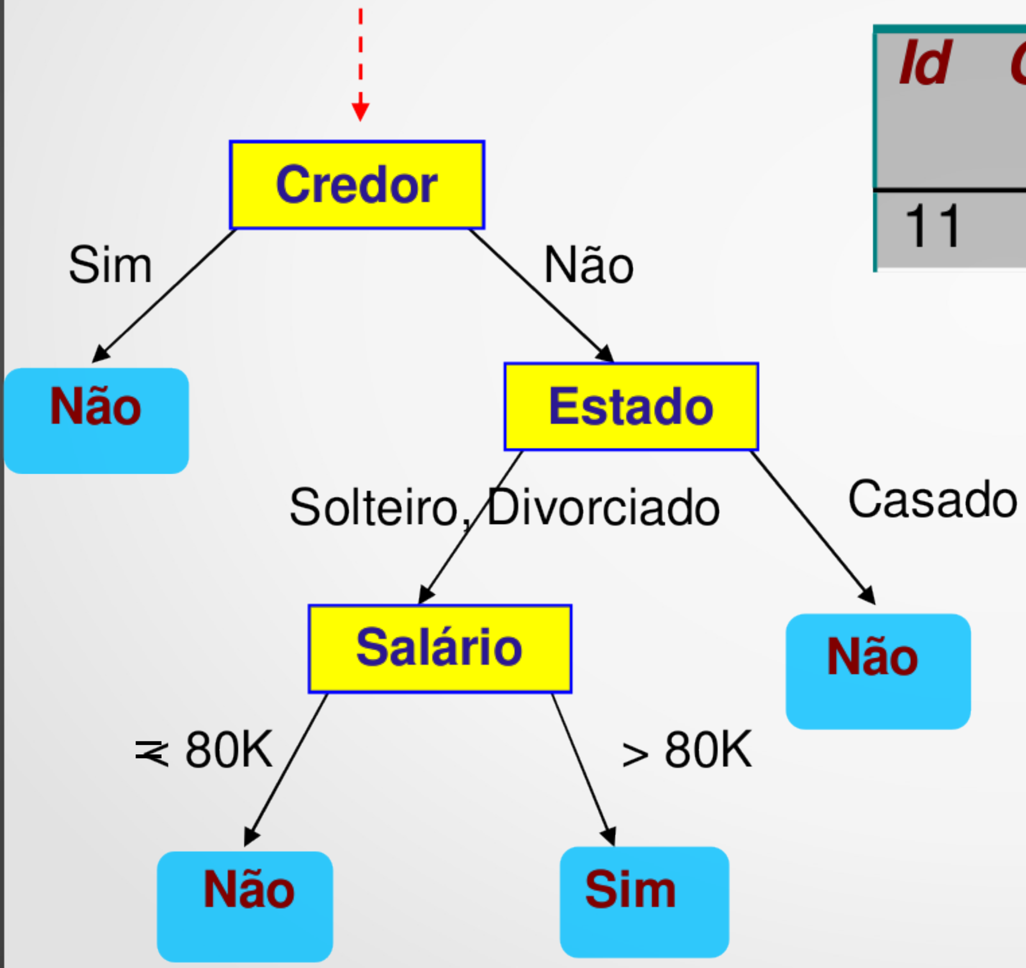
Conjunto de Teste

ID	Atrib.1	Atrib.2	Atrib.3	Classe
11	Não	Pequeno	55K	?
12	Sim	Médio	80K	?
13	Sim	Grande	110K	?
14	Não	Pequeno	95K	?
15	Não	Grande	67K	?



Exemplo: Aplicar Modelo

Começar da Raiz



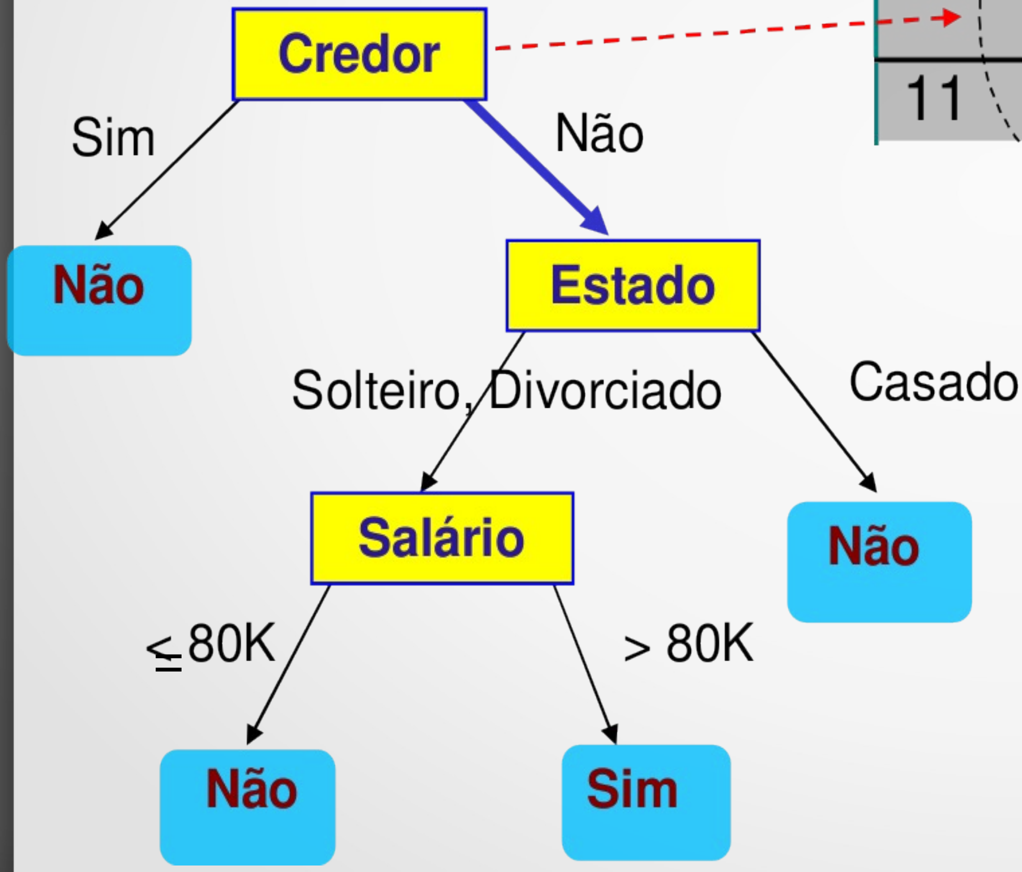
Dados de Teste

Id	*Credor*	*Estado Civil*	*Salário*	*Calote*
11	Não	Casado	80K	???

Exemplo: Aplicar Modelo

Dados de Teste

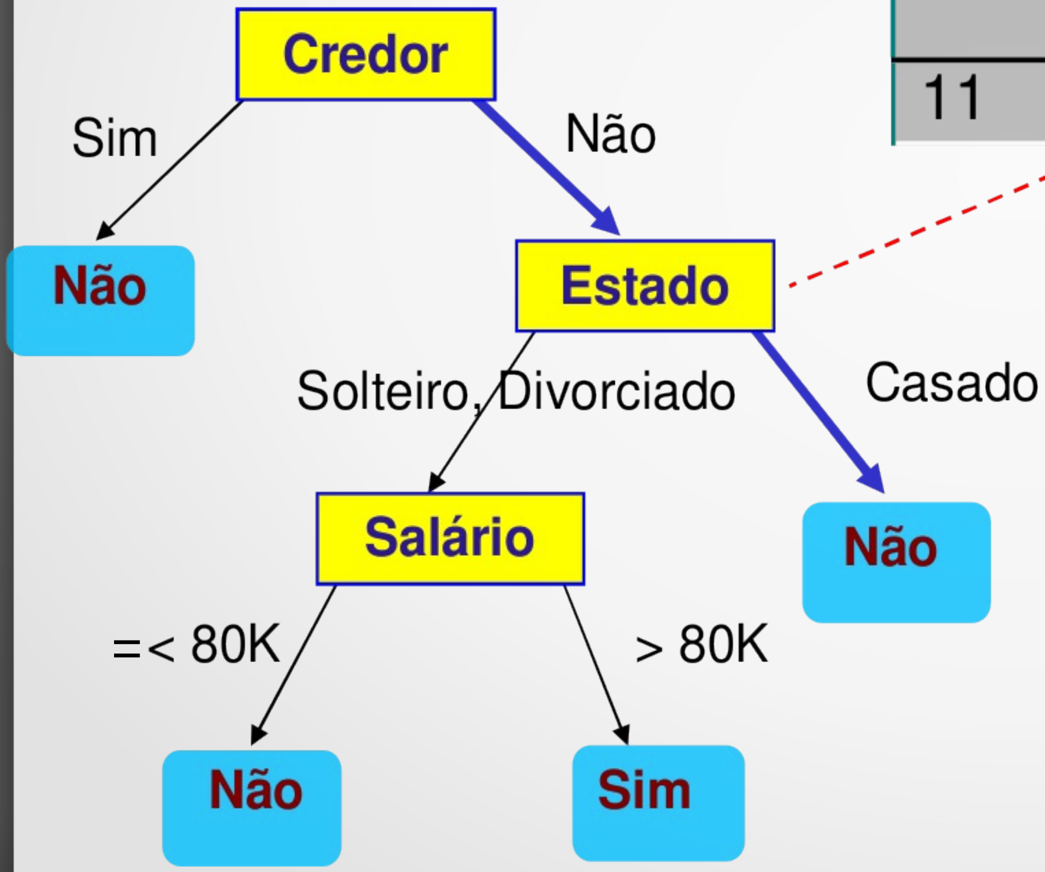
Id	*Credor*	*Estado Civil*	*Salário*	*Calote*
11	Não	Casado	80K	???



Exemplo: Aplicar Modelo

Dados de Teste

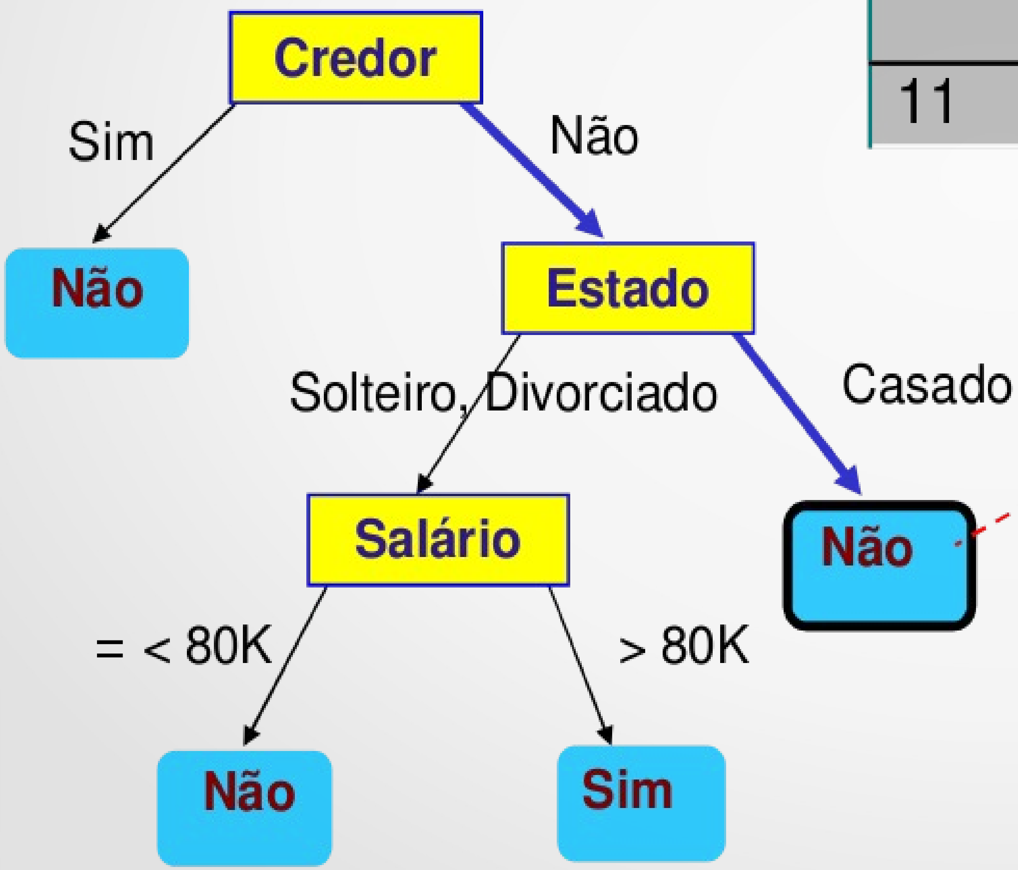
Id	*Credor*	*Estado Civil*	*Salário*	*Calote*
11	Não	Casado	80K	???



Exemplo: Aplicar Modelo

Dados de Teste

Id	*Credor*	*Estado Civil*	*Salário*	*Calote*
11	Não	Casado	80K	???

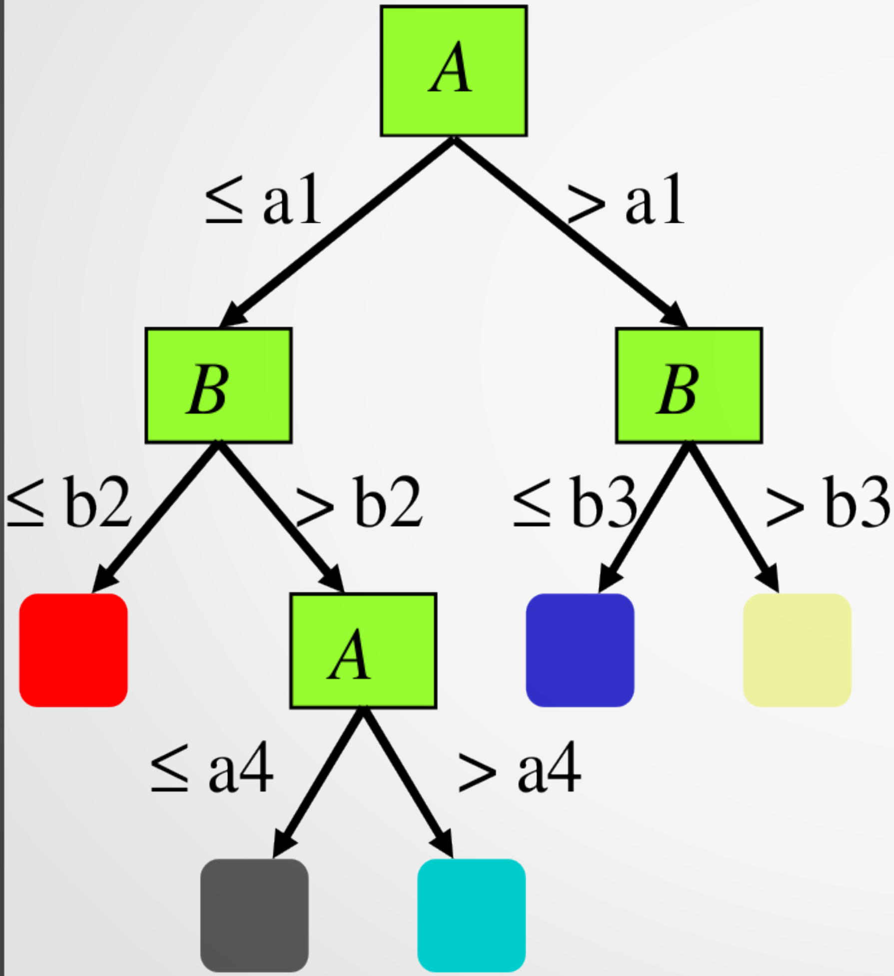


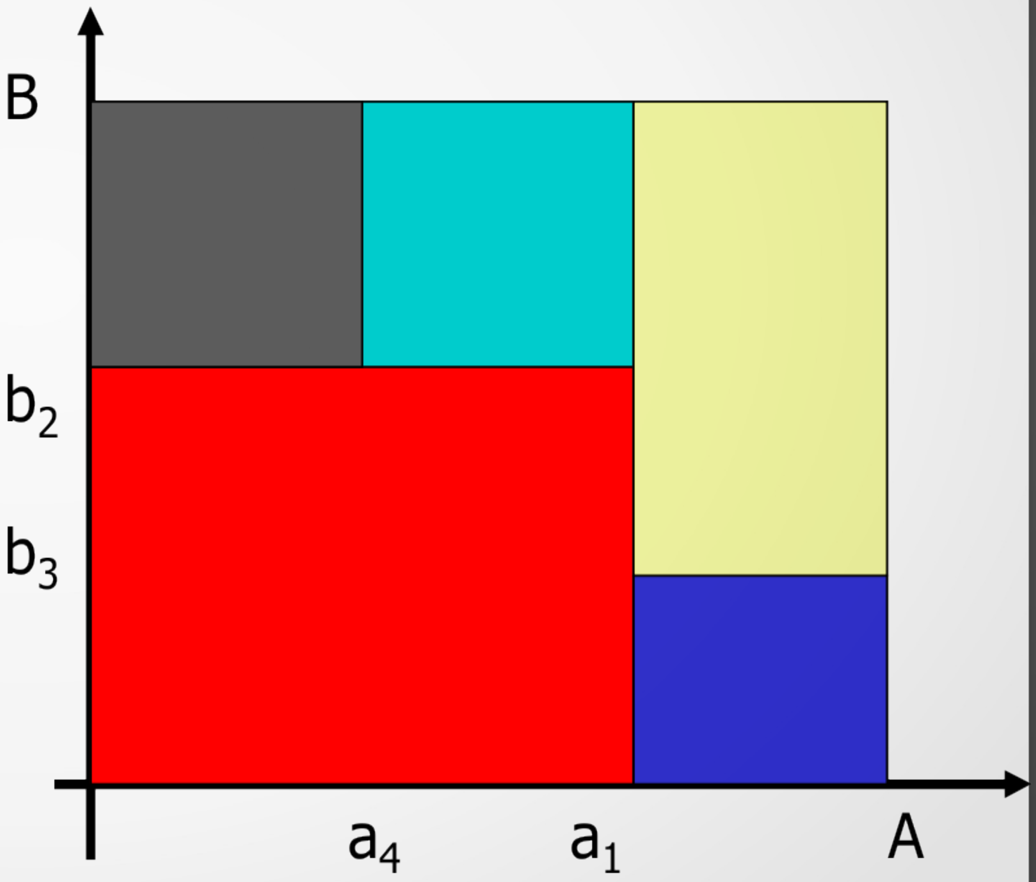
Atribui classe **Não**

Árvores e Regras

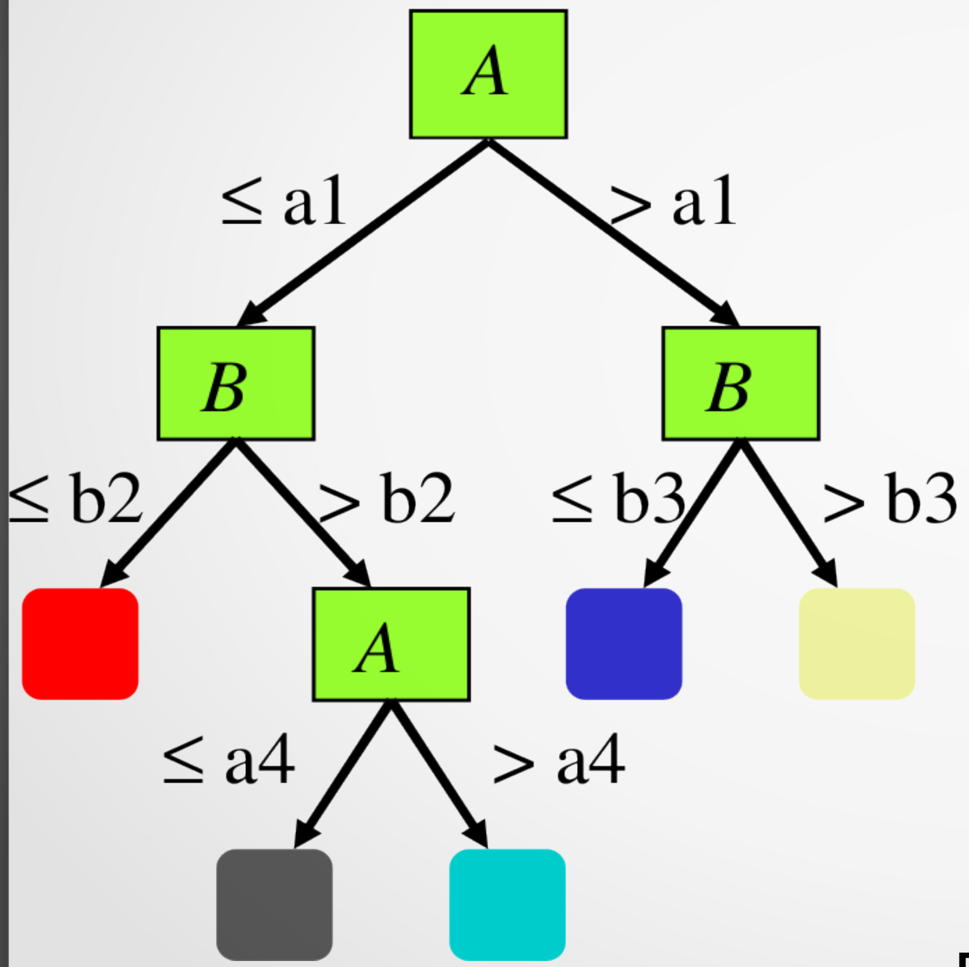
- Cada percurso da raiz a um nó folha representa uma regra de classificação
- Cada nó folha
  - Está associado a uma classe
  - Corresponde a uma região do domínio dos atributos
  - Hiper-retângulo
    - Interseção de hiper-retângulos é um conjunto vazio
    - União é o espaço total

Árvores e Regras





Árvores e Regras



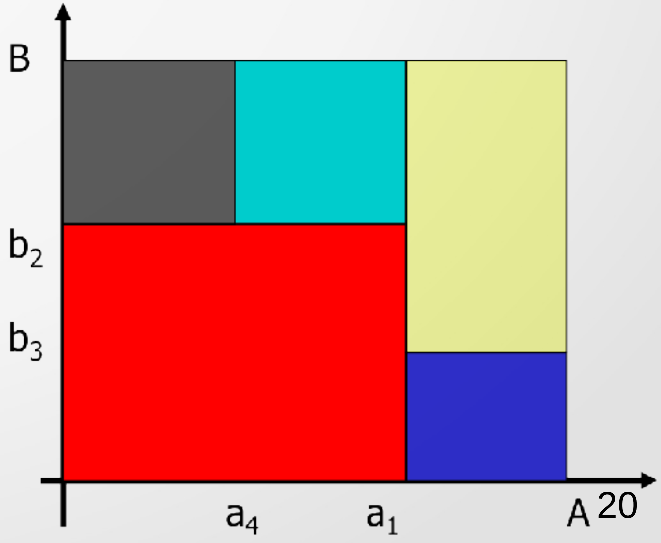
Regras: disjunções de conjunções lógicas

1. Se A ≤ a_1 E B ≤ b_2
Então Classe = Vermelha
OU
2. Se A > a_1 E B ≤ b_3
Então Classe = Azul
OU
...

Exercício: complete as regras !

Espaço de Hipóteses

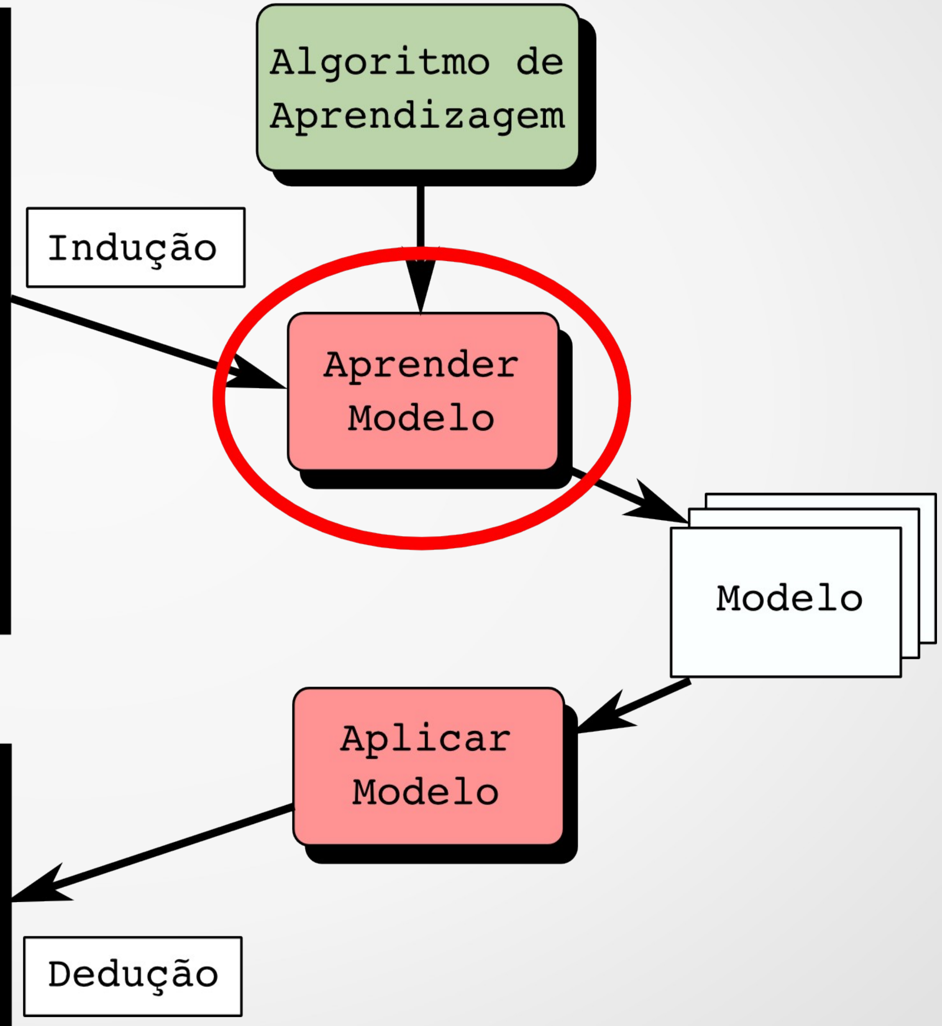
- Uma árvore de decisão específica ou o conjunto de regras correspondente representa uma hipótese no espaço de hipóteses para a função de classificação a ser aproximada
- Qualquer realização particular de um modelo é uma hipótese!



Aprender Modelo

Conjunto de Treinamento

ID	Atrib.1	Atrib.2	Atrib.3	Classe
1	Sim	Grande	125K	Não
2	Não	Médio	100K	Não
3	Não	Pequeno	70K	Não
4	Sim	Médio	120K	Não
5	Não	Grande	95K	Sim
6	Não	Médio	60K	Não
7	Sim	Grande	220K	Não
8	Não	Pequeno	85K	Sim
9	Não	Médio	75K	Não
10	Não	Pequeno	90K	Sim



Conjunto de Teste

ID	Atrib.1	Atrib.2	Atrib.3	Classe
11	Não	Pequeno	55K	?
12	Sim	Médio	80K	?
13	Sim	Grande	110K	?
14	Não	Pequeno	95K	?
15	Não	Grande	67K	?

Indução de ADs

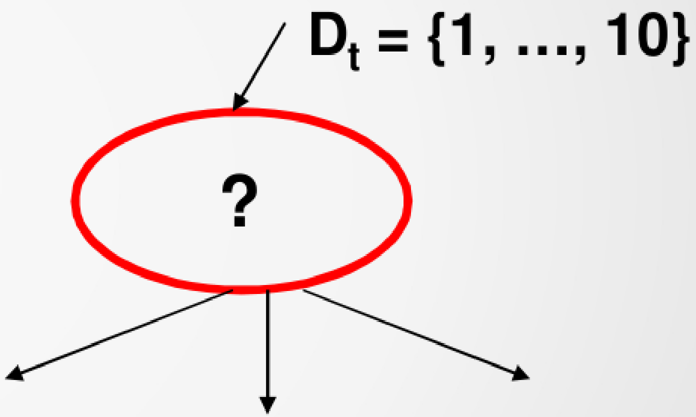
- Existem vários algoritmos
  - Hunt’s Concept Learning System
    - Um dos primeiros
    - Base de vários algoritmos atuais
  - ID3, C4.5, J4.8, C5.0
  - CART, Random-Forest
  - ...

Algoritmo de *Hunt*

- Seja D_t o conjunto de objetos que atingem o nó *t* (não classificados), o algoritmo de Hunt será:
  - Passo 1. Se todos os objetos de D_t pertencem à mesma classe c_t , então *t* é um nó folha rotulado como c_t
  - Passo 2. Se D_t contém objetos que pertencem a mais de uma classe, então *t* deve ser um nó interno
    - Passo 2.1. O nó deve conter uma condição de teste sobre algum valor dos atributos que não foi selecionado acima na árvore
    - Passo 2.2. Um nó filho é criado para cada saída da condição de teste (valor do atributo) e os objetos em D_t são distribuídos neles
    - Passo 2.3. O algoritmo é aplicado recursivamente para cada nó filho

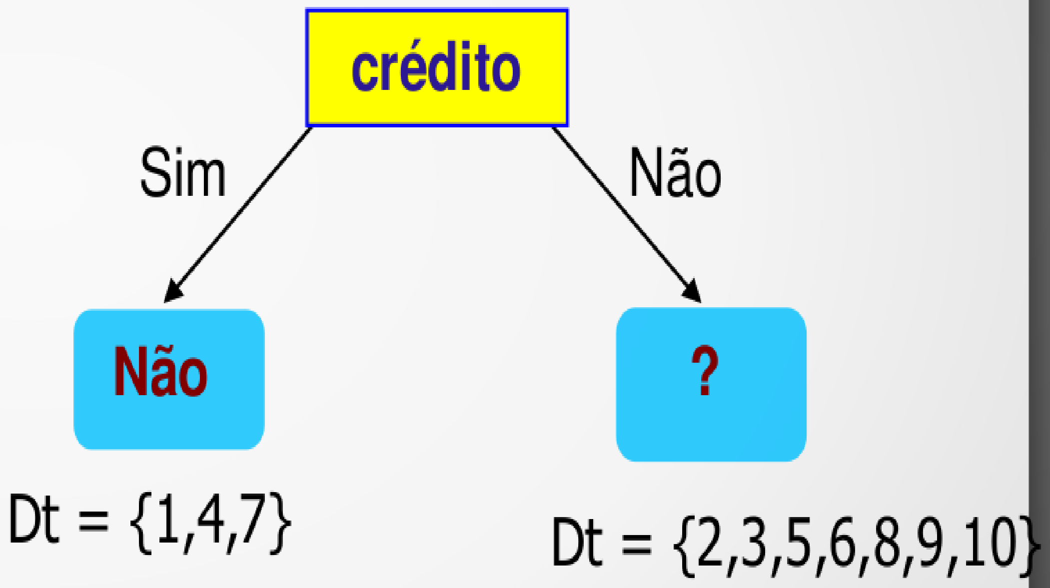
Algoritmo de Hunt

Id	Crédito	Estado Civil	Renda	Deve
1	Sim	Solteiro	125K	Não
2	Não	Casado	100K	Não
3	Não	Solteiro	70K	Não
4	Sim	Casado	120K	Não
5	Não	Divorciado	95K	Sim
6	Não	Casado	60K	Não
7	Sim	Divorciado	220K	Não
8	Não	Solteiro	85K	Sim
9	Não	Casado	75K	Não
10	Não	Solteiro	90K	Sim



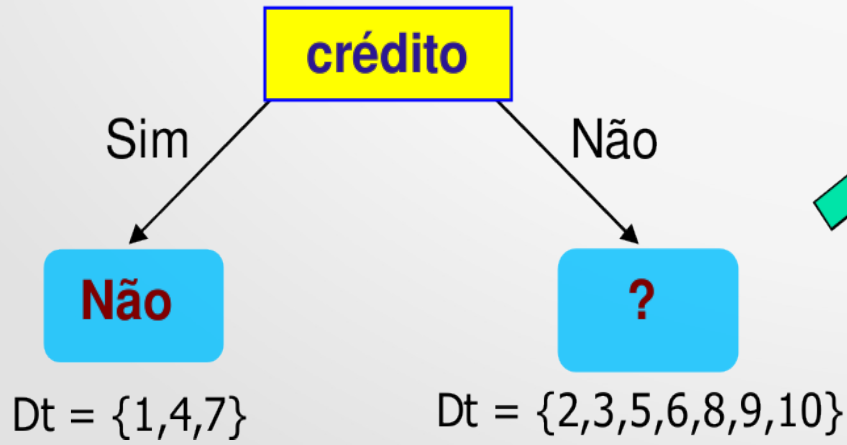
Algoritmo de Hunt

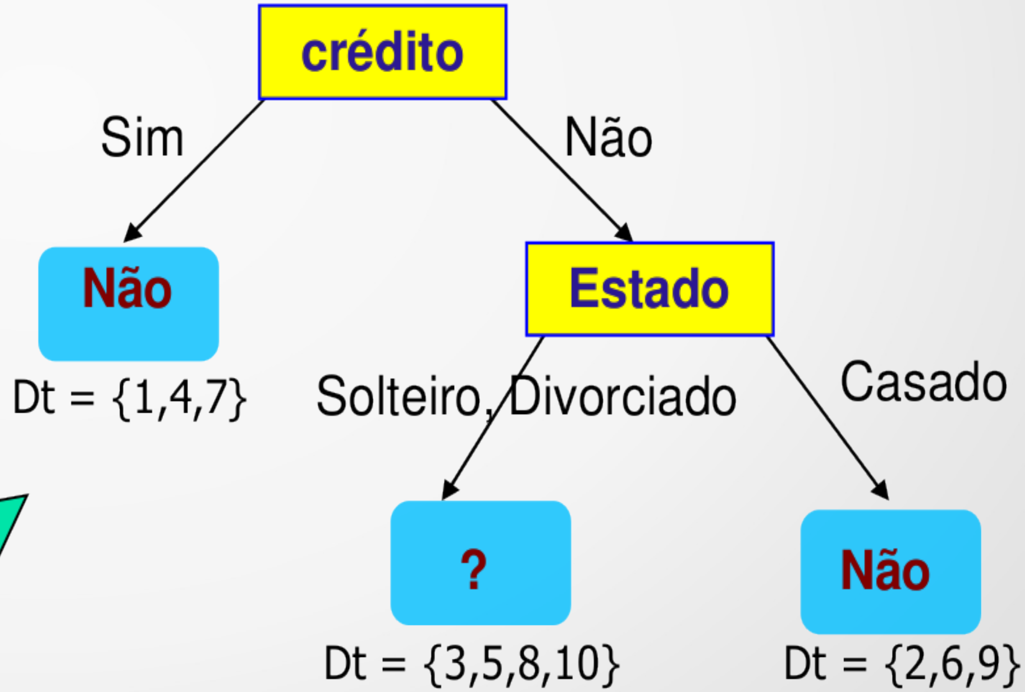
Id	Crédito	Estado Civil	Renda	Deve
1	Sim	Solteiro	125K	Não
2	Não	Casado	100K	Não
3	Não	Solteiro	70K	Não
4	Sim	Casado	120K	Não
5	Não	Divorciado	95K	Sim
6	Não	Casado	60K	Não
7	Sim	Divorciado	220K	Não
8	Não	Solteiro	85K	Sim
9	Não	Casado	75K	Não
10	Não	Solteiro	90K	Sim



Algoritmo de Hunt

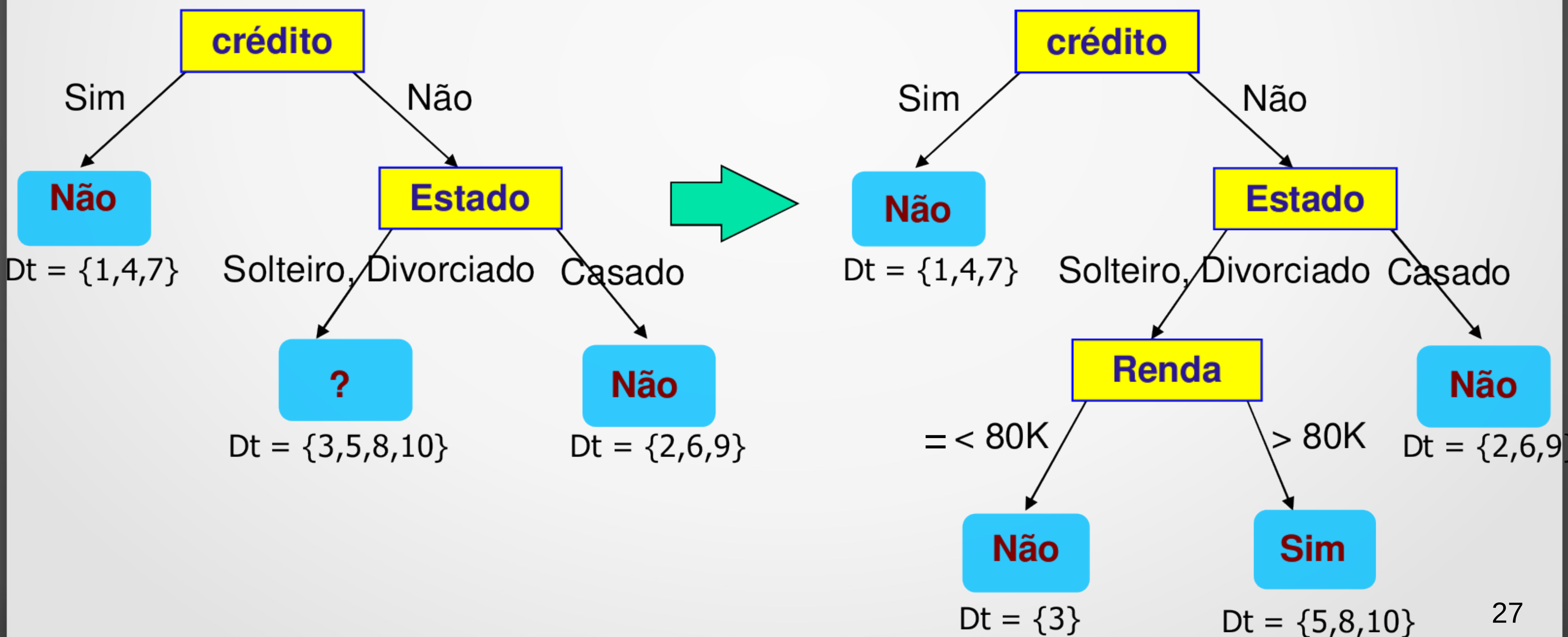
Id	Crédito	Estado Civil	Renda	Deve
1	Sim	Solteiro	125K	Não
2	Não	Casado	100K	Não
3	Não	Solteiro	70K	Não
4	Sim	Casado	120K	Não
5	Não	Divorciado	95K	Sim
6	Não	Casado	60K	Não
7	Sim	Divorciado	220K	Não
8	Não	Solteiro	85K	Sim
9	Não	Casado	75K	Não
10	Não	Solteiro	90K	Sim





Algoritmo de Hunt

Id	Crédito	Estado Civil	Renda	Deve
1	Sim	Solteiro	125K	Não
2	Não	Casado	100K	Não
3	Não	Solteiro	70K	Não
4	Sim	Casado	120K	Não
5	Não	Divorciado	95K	Sim
6	Não	Casado	60K	Não
7	Sim	Divorciado	220K	Não
8	Não	Solteiro	85K	Sim
9	Não	Casado	75K	Não
10	Não	Solteiro	90K	Sim



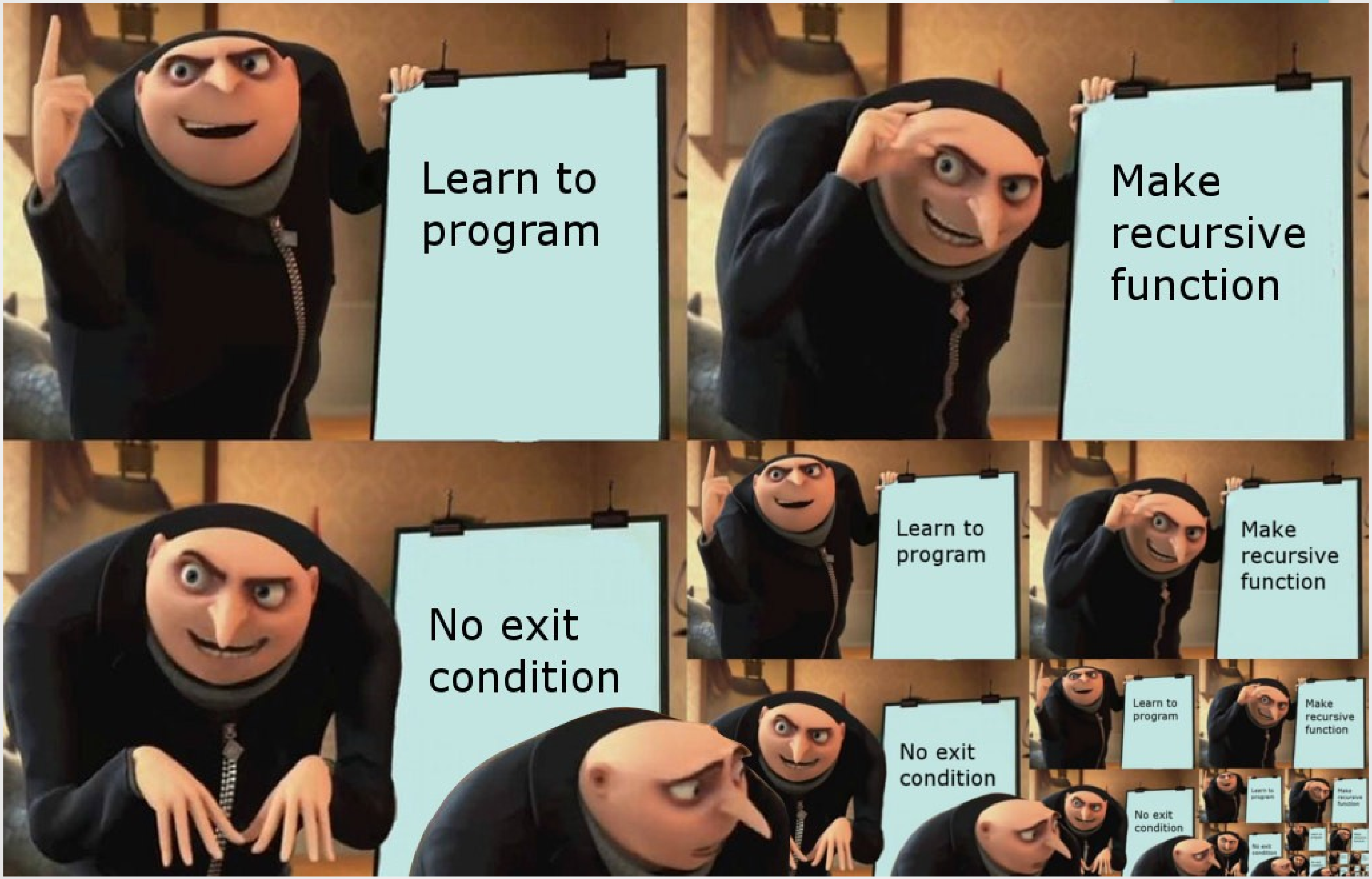
Algoritmo de Hunt

- Problema: o algoritmo rudimentar apresentado anteriormente garantidamente funciona apenas se:
  - Houver ao menos um objeto para cada combinação possível dos valores dos atributos preditores; e
  - Havendo mais de um, devem pertencer todos à mesma classe

Algoritmo de Hunt

- Solução (dada que essas hipóteses são muito restritivas):
  - Se D_t for vazio para um determinado nó *t*, rotular o nó com a classe majoritária dos objetos do nó pai
  - Se D_t for composto de objetos pertencentes a classes distintas em um dado nó *t* e não há mais atributos disponíveis, rotular o nó com a classe majoritária desses objetos

Critério de Parada



Critério de Parada

- Chamada recursiva pode ser finalizada:
  - Quando todos os dados do nó atual possuem o mesmo rótulo
  - Quando os dados do nó atual ainda possuem rótulos de classes diferentes, porém possuem os “mesmos valores” (categóricos) para todos os atributos preditores
    - o que significa que todos os atributos já terão sido incluídos no caminho a partir da raiz, não havendo mais atributos disponíveis

Decisões importantes

- Decisões importantes a serem consideradas durante a indução:
  - Como dividir os objetos?
    - Como escolher o atributo de divisão?
    - Qual a melhor divisão para aquele atributo?
  - Quando parar de dividir os objetos?

Condição de Teste

- Depende do tipo do atributo
  - Binário
  - Nominal
  - Ordinal
  - Contínuo
- Depende do número de divisões
  - 2 divisões
  - Mais que 2 divisões

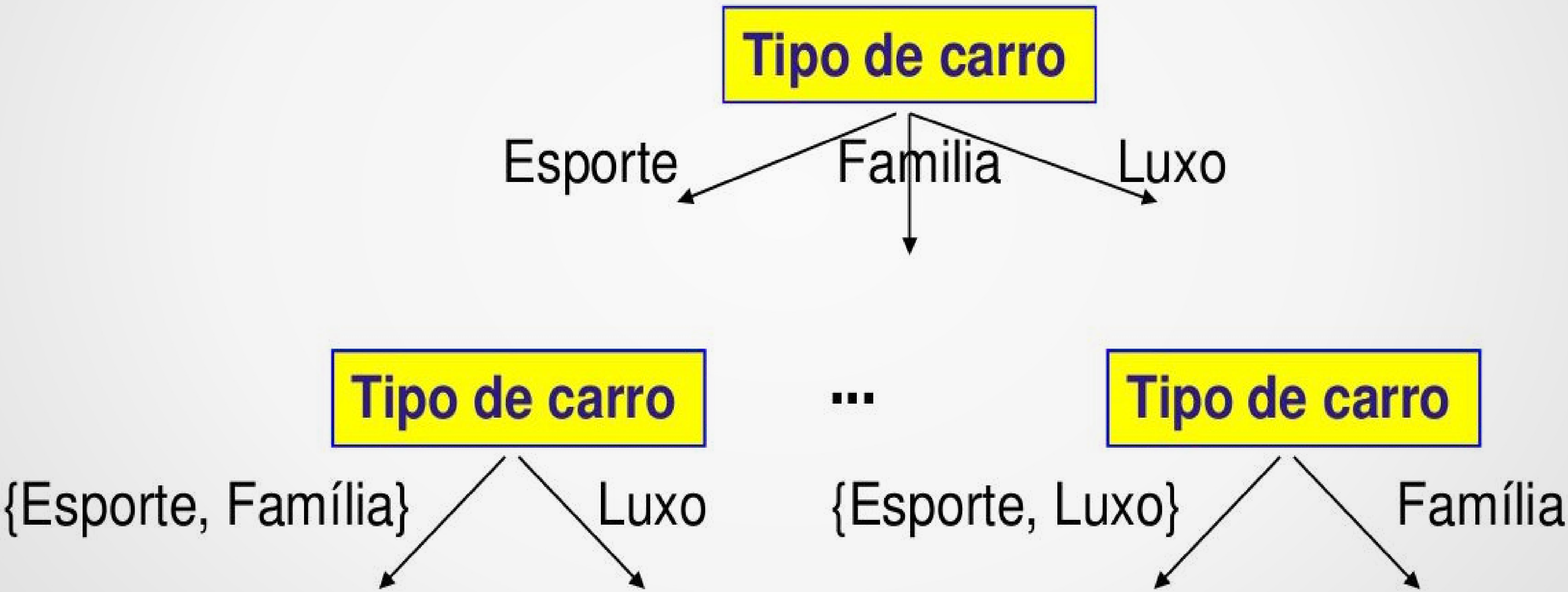
Atributos Binários

- Teste mais simples
  - Apenas dois possíveis resultados

Atributos Nominais

- Pode assumir mais que dois valores
- Duas formas de condição de teste
  - Usar tantos ramos quantos forem os possíveis valores do atributo
  - Unir valores em cada ramo
    - disjunção lógica

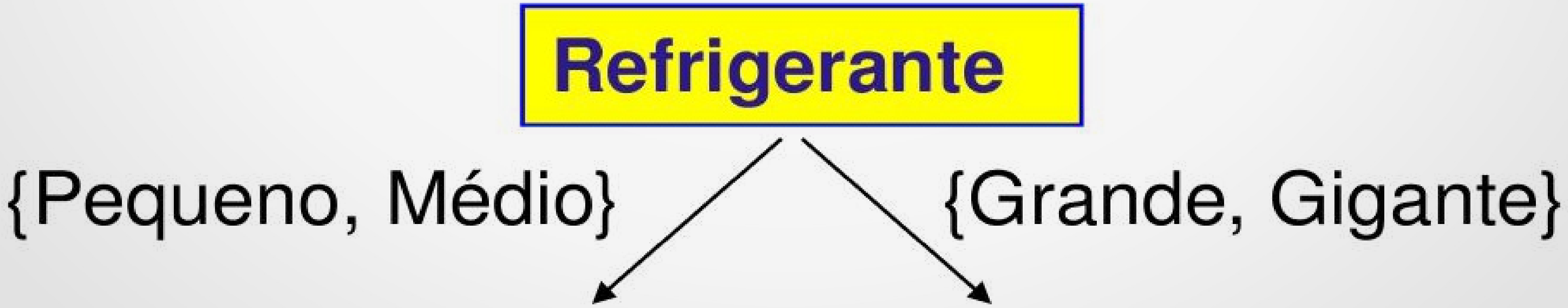
Exemplo



Tipo de carro
Esporte
Familia
Luxo
Tipo de carro
...
Tipo de carro
{Esporte, Família}
Luxo
{Esporte, Luxo}
Família

Atributos Ordinais

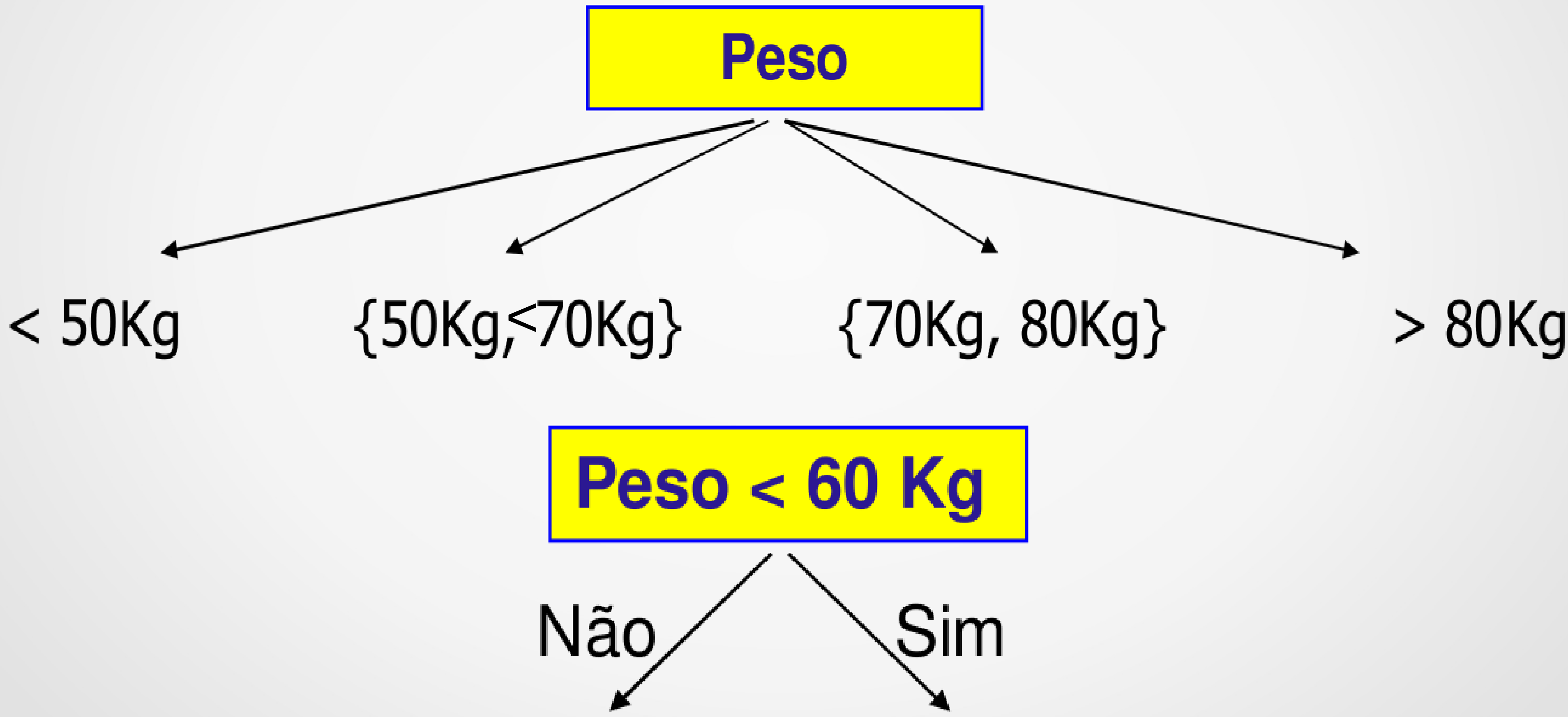
- Duas alternativas
  - Usar tantos ramos quantos forem os possíveis valores do atributo
  - Unir valores em cada ramo
    - sem violar relação de ordem



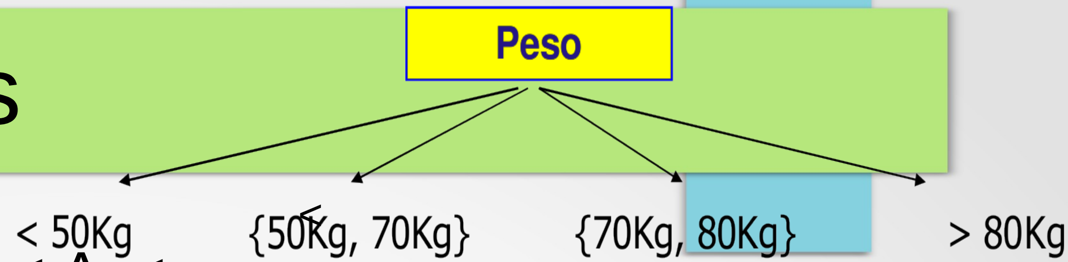
Atributos Contínuos

- Consultas sobre intervalos: $x < A_i < y$
  - Usar estratégia de discretização
- Condição de teste pode ser expressa por:
  - Comparação: $A_i < x$
    - Escolher valor x de A_i que gera melhor divisão
      - Ponto de referência

Exemplos

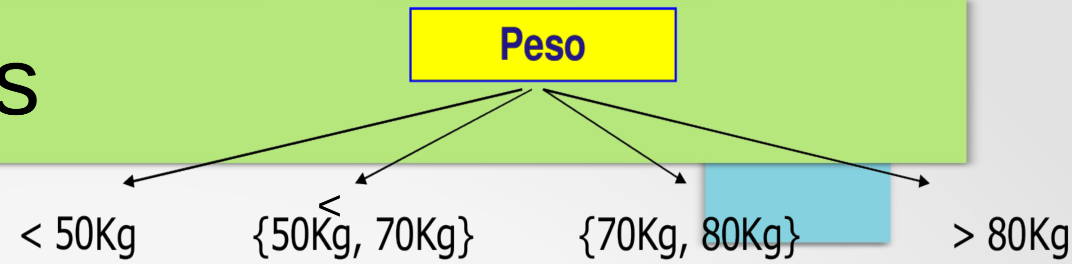


Atributos Contínuos

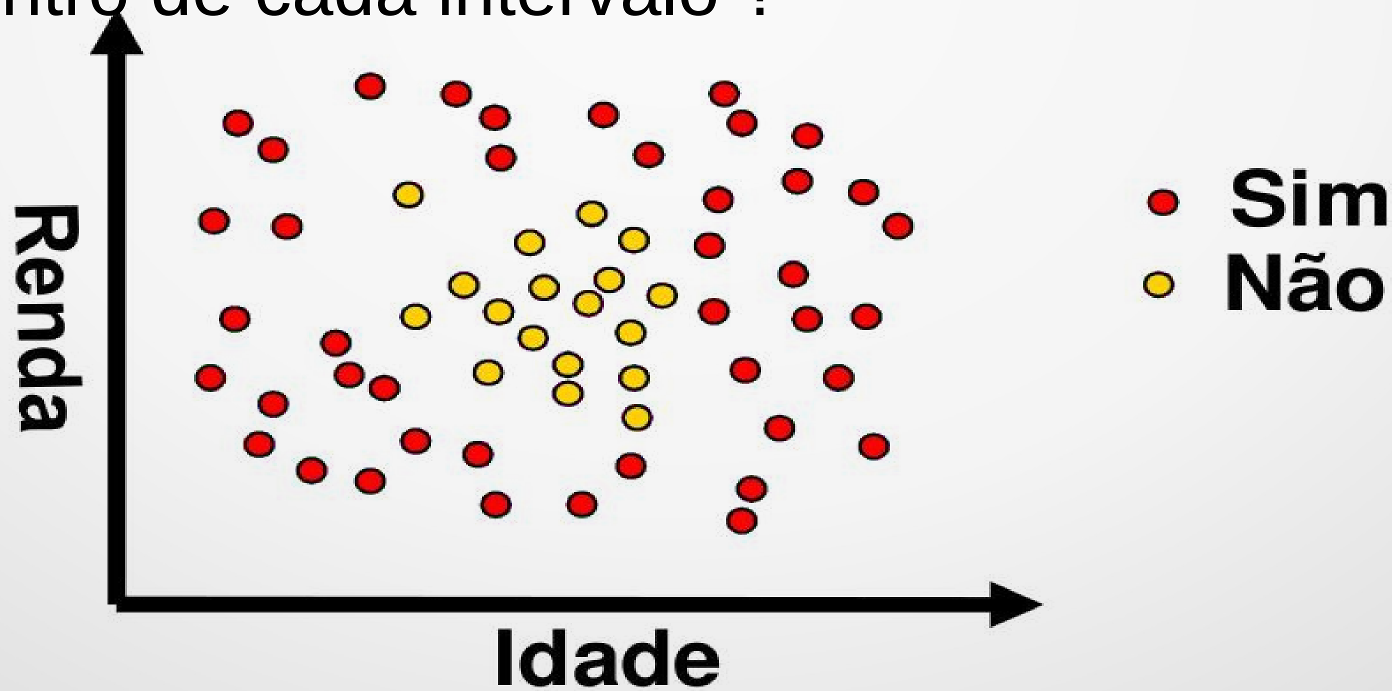


- Consultas sobre Intervalos: $x < A_i < y$
  - Vantagem:
    - Com a discretização, o atributo pode ser manipulado pelo algoritmo como um atributo nominal qualquer
      - não requer modificações no algoritmo de indução de ADs
  - Desvantagens:
    - Tenta esgotar a capacidade de divisão do atributo
    - Tende a gerar muitos ramos desnecessários na árvore
      - antes que parte dos objetos do nó da árvore em questão possam ser melhor pré-classificados por outro atributo
- Qual o número ideal de intervalos...???

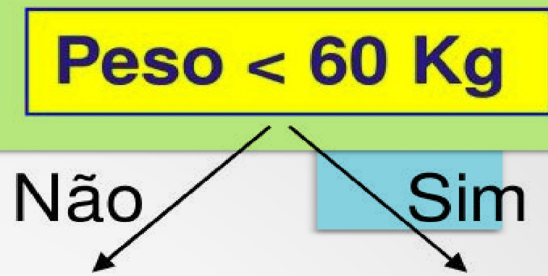
Atributos Contínuos



- Qual seria uma boa discretização supervisionada dos dados abaixo para cada atributo?
  - Quantos intervalos são necessários para obter o máximo grau possível de pureza de classe(s) dentro de cada intervalo ?

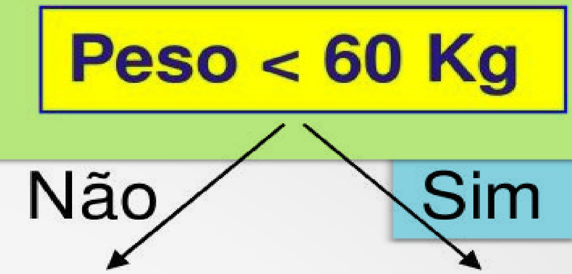


Atributos Contínuos



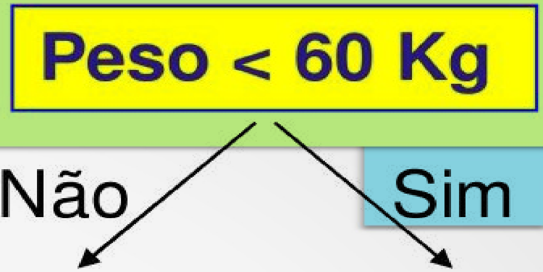
- Comparações: $A_i < x$
  - Atributo não é removido do conjunto de candidatos à divisão
  - Pode gerar ramos mais profundos (regras mais complexas)
    - requer modificações no algoritmo básico de indução, em especial no que diz respeito à interrupção ou poda da árvore (p. ex. algo. C4.5)

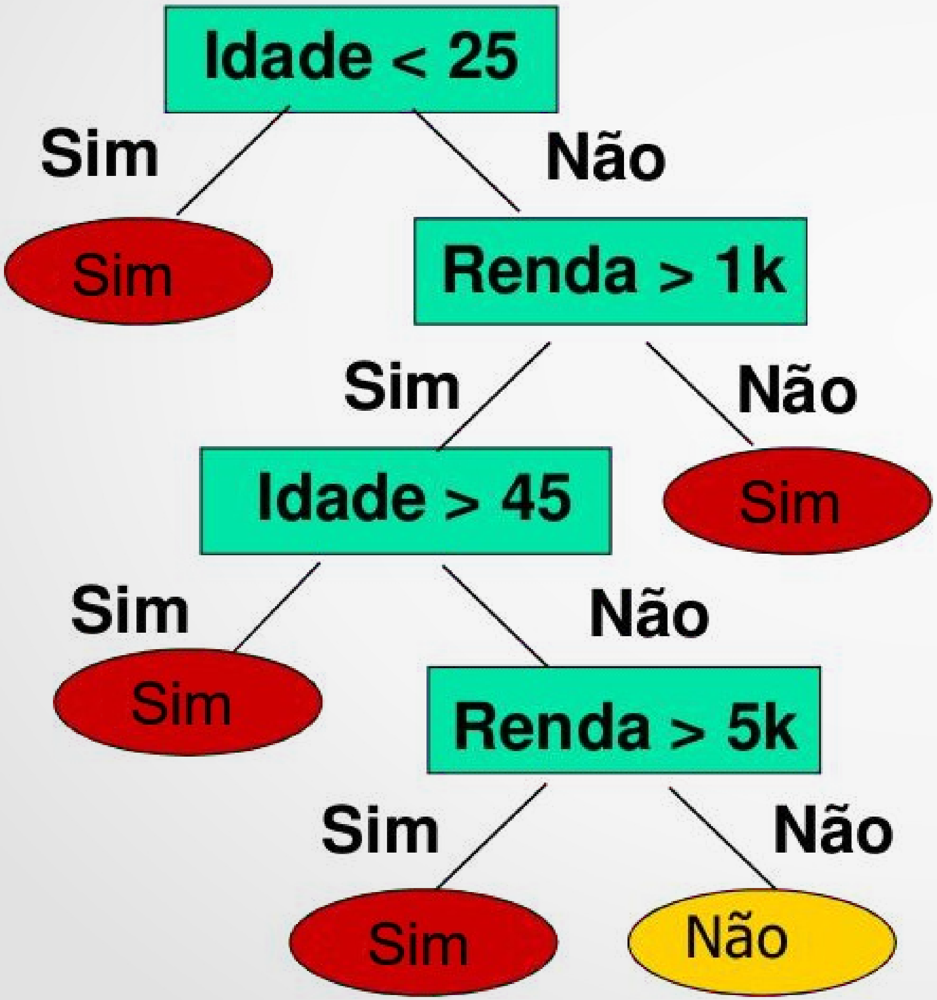
Atributos Contínuos

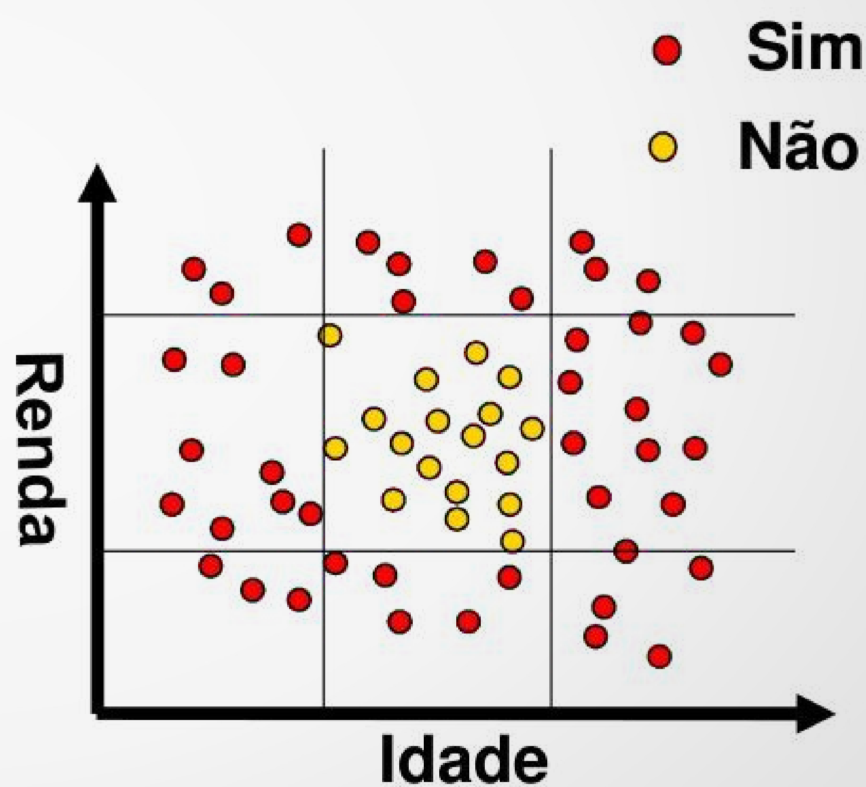


- Em contrapartida:
  - escolha do atributo e do valor de comparação é “por demanda”
  - árvore mais flexível -> maior poder de discriminação
- escolha do atributo e do valor de comparação é muito mais simples
  - 1 ponto de discretização -> no. fixo de intervalos = 2
  - tendência a minimizar a largura da árvore

Atributos Contínuos



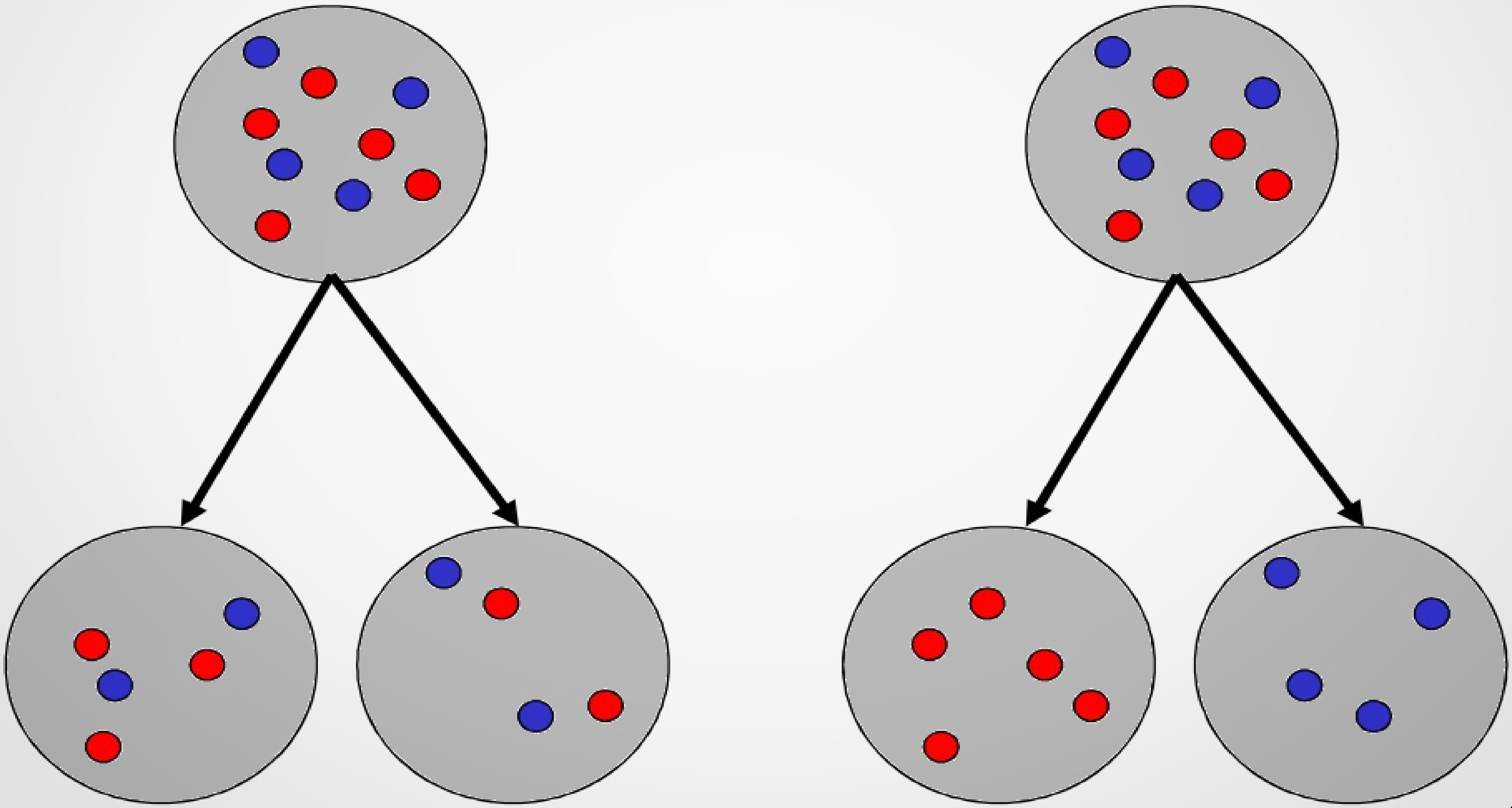




Medidas para Escolha de Atributo

- Existem várias medidas para determinar a melhor forma de dividir os objetos
- Medidas de impureza
  - Definidas em termos da distribuição de classes dos dados antes e após a divisão
  - Baseadas na ideia que:
    - A melhor partição é aquela em que todos os dados são divididos em grupos com uma mesma classe
    - Quanto mais balanceadas as classes, pior

Medidas para Escolha de Atributo

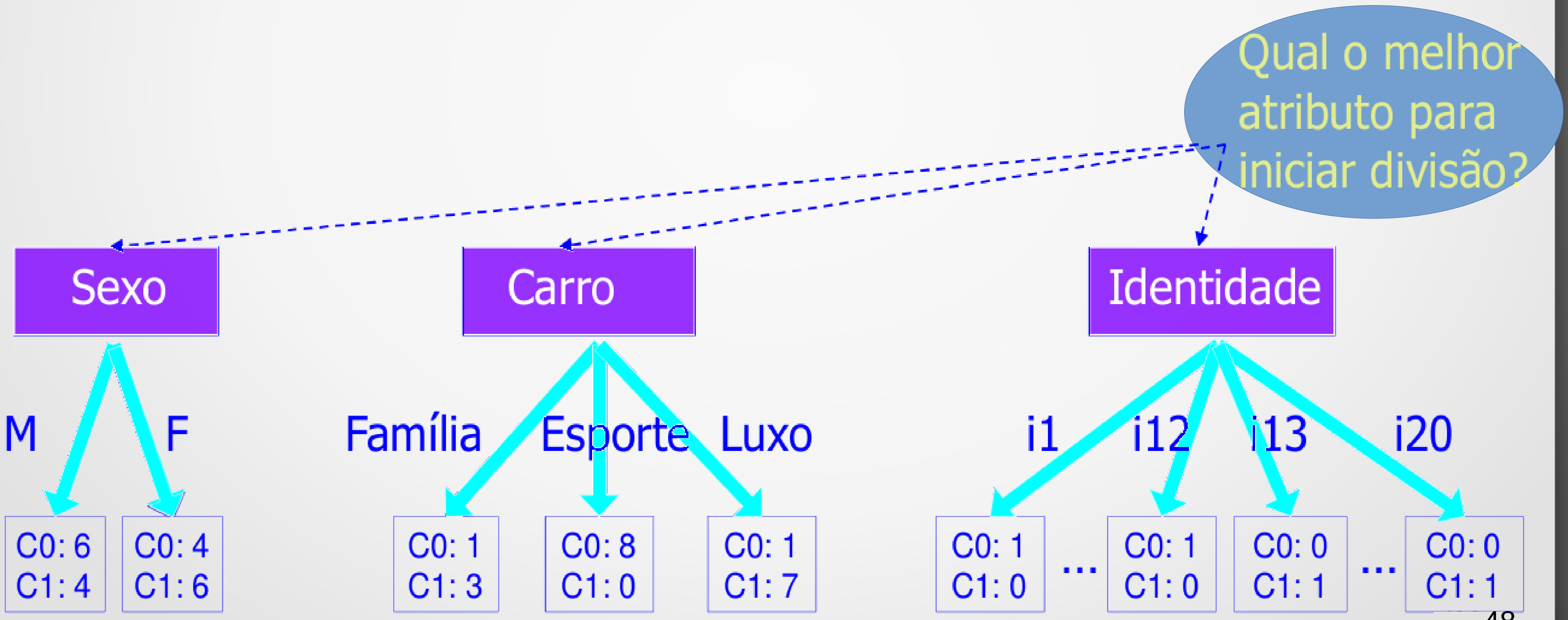


Medidas de impureza

- Medidas diferentes geram partições diferentes dos dados
- Exemplos de medidas de impureza:
  - Entropia
  - Gini
  - Erro de classificação
  - Qui-quadrado

Medidas de impureza

- Supor que D possui antes da divisão:
  - 10 exemplos da classe 0 (C0: 10)
  - 10 exemplos da classe 1 (C1: 10)



Medidas de impureza

- Abordagem gulosa
  - Prefere nós com distribuição mais homogênea (pura) de classes
  - Necessário uma medida de (im)pureza

C0: 5	C0: 9
C1: 5	C1: 1
Muito heterogênea	**Muito homogênea**
Alto grau de impureza	**Baixo grau de impureza**

Medidas de impureza

$$\text{Entropia}(t) = -\sum_{i=1}^{c} p(i \mid t) \log_2 p(i \mid t)$$

$$\text{Gini\_Index}(t) = 1 - \sum_{i=1}^{c} \left[p(i \mid t)\right]^2$$

$$\text{Erro\_Class}(t) = 1 - \max_{i \in \{1,\ldots,c\}} \left[p(i \mid t)\right]$$

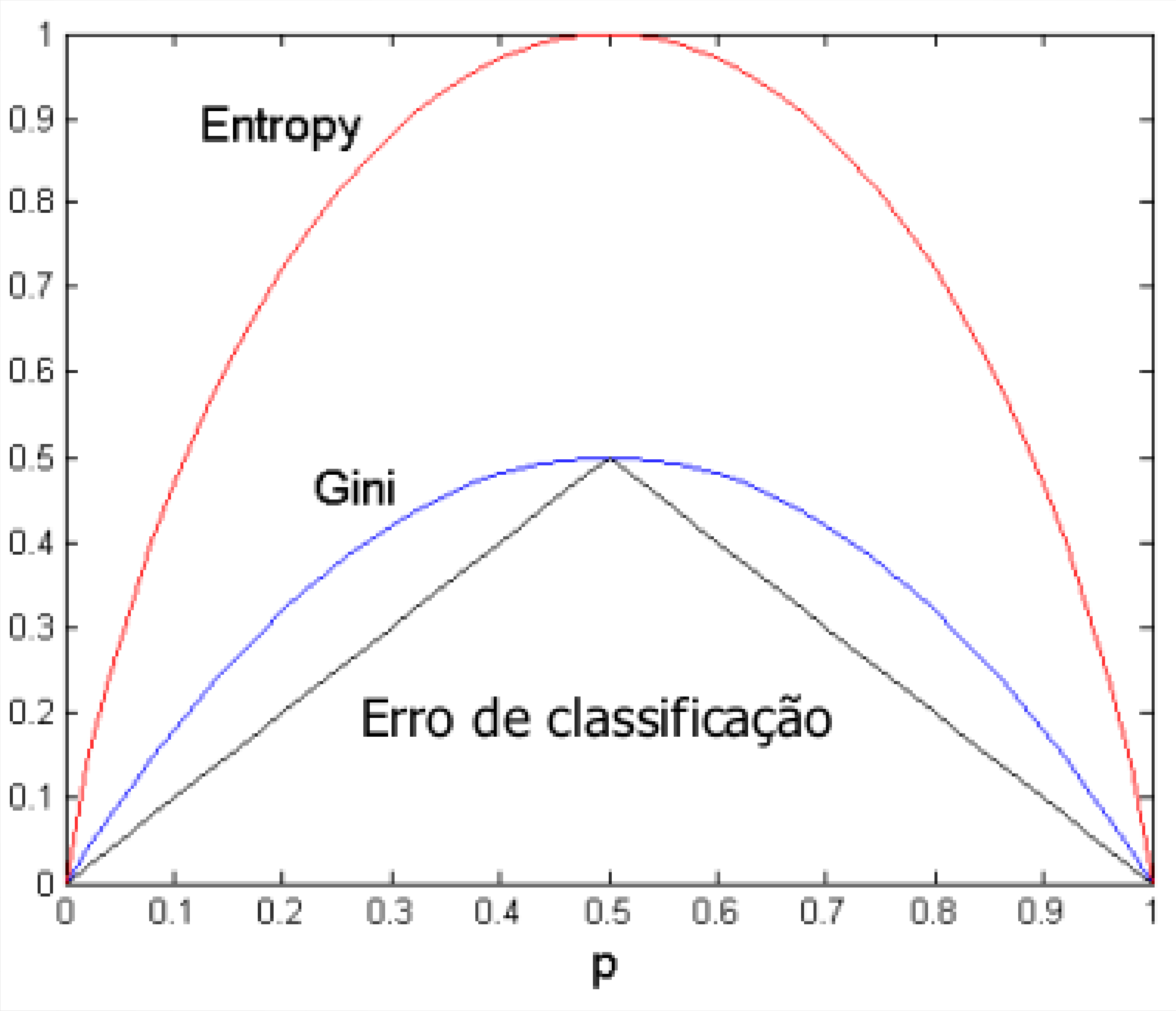
onde:

p(i|t) = fração de dados pertencente à classe i em um nó t

c = número de classes

$0 \log_2 0 = 0$

Comparação: Duas Classes



Comparação

- Valor máximo:
  - Entropia: ($\log_2 c$)
  - Gini e Erro de classificação: ($1 - 1/c$)
  - Quando os dados estão igualmente distribuídos entre todas as classes
    - Informação menos interessante (menos informação)
- Valor mínimo: 0 (para todos)
  - Quando os dados pertencem a uma classe
    - Informação mais interessante

Exemplo

- Calcular a medida de impureza Gini para os dados abaixo:

$$\text{Gini\_Index}(t) = 1 - \sum_{i=1}^{c} [p(i \mid t)]^2$$

C1	**0**
C2	**6**
Gini=?	

C1	**1**
C2	**5**
Gini=?	

C1	**2**
C2	**4**
Gini=?	

C1	**3**
C2	**3**
Gini=?	

Exemplo

$P(C1) = 0/6 = 0 \quad P(C2) = 6/6 = 1$

$\text{Gini} = 1 - P(C1)^2 - P(C2)^2 = 1 - 0 - 1 = 0$

$P(C1) = 1/6 \quad P(C2) = 5/6$

$\text{Gini} = 1 - (1/6)^2 - (5/6)^2 = 0.278$

$P(C1) = 2/6 \quad P(C2) = 4/6$

$\text{Gini} = 1 - (2/6)^2 - (4/6)^2 = 0.444$

$P(C1) = 3/6 \quad P(C2) = 3/6$

$\text{Gini} = 1 - (3/6)^2 - (3/6)^2 = 0.500$

C1	**0**
C2	**6**
Gini=?	

C1	**1**
C2	**5**
Gini=?	

C1	**2**
C2	**4**
Gini=?	

C1	**3**
C2	**3**
Gini=?	

Exercícios

- Fazer os mesmos cálculos para as medidas de entropia e classificação

$$\text{Entropia}(t) = -\sum_{i=1} p(i \mid t) \log_2 p(i \mid t)$$

$$\text{Erro\_Class}(t) = 1 - \max_{i \in \{1,\ldots,c\}} [p(i \mid t)]$$

C1	**0**
C2	**6**
E=?	

C1	**1**
C2	**5**
E=?	

C1	**2**
C2	**4**
E=?	

C1	**3**
C2	**3**
E=?	

C1	**0**
C2	**6**
Class=?	

C1	**1**
C2	**5**
Class=?	

C1	**2**
C2	**4**
Class=?	

C1	**3**
C2	**3**
Class=?	

Medidas de impureza

- Usadas para avaliar a qualidade de cada condição de teste candidata
  - Compara-se o grau de impureza antes e após a divisão
  - Quanto maior a diferença, melhor o atributo
- Exemplos:
  - Ganho de Informação: usada, por exemplo, pelo algoritmo ID3
  - Média Ponderada de Gini: usada, por exemplo, pelo algoritmo CART

Medida de Ganho

$$\Delta = I(v_{pai}) - \sum_{t=1}^{k} \frac{N(v_t)}{N} I(v_t)$$

Soma ponderada pela proporção de objetos em cada um dos k nós filhos

- onde:

  $I(v_t)$: mede o grau de impureza do nó filho v_t

  $N(v_t)$: no. de objetos do nó filho vt

  N: no. de objetos do nó original (v_{pai})

- Quando a medida de impureza é Entropia, Δ mede o Ganho de Informação (Δ_{info})

Medida de Ganho

$$\Delta = I(v_{pai}) - \sum_{t=1}^{k} \frac{N(v_t)}{N} I(v_t)$$

- Note que o primeiro termo será constante para todos os atributos e, portanto, pode ser omitido para comparar os Δs associados a cada atributo
- Isso é feito no critério da média ponderada de Gini

Média Ponderada de Gini

- Quando um nó é dividido em k filhos, a qualidade da divisão é definida por:

$$\text{Gini}_{divisão} = \sum_{t=1}^{k} \frac{N(v_t)}{N} \text{Gini}(v_t)$$

➡ Quanto menor melhor

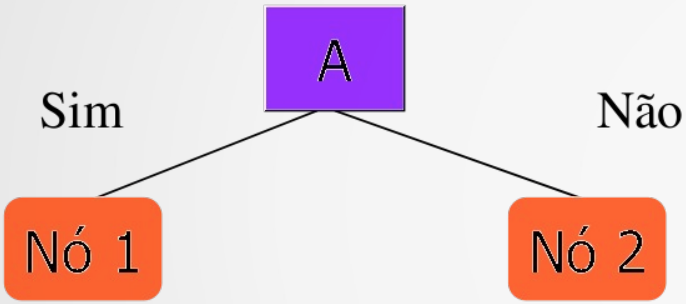
onde:

$N(v_t)$: no. de objetos do nó filho v_t

N: no. de objetos do nó original (pai)

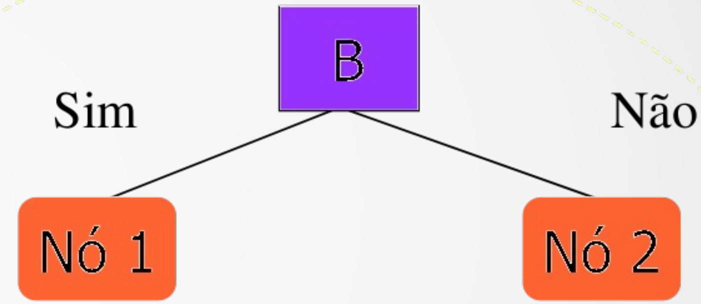
Divisão de Atributos Binários

	Pai
C1	6
C2	6
Gini = 0.500	



	Nó 1	*Nó 2*
C1	4	2
C2	3	3
Gini_d = 0.486		

$\text{Gini}_{\text{divisão}} = (7/12)\text{x}0.49 + (5/12)\text{x}0.48$
$= 0.486$

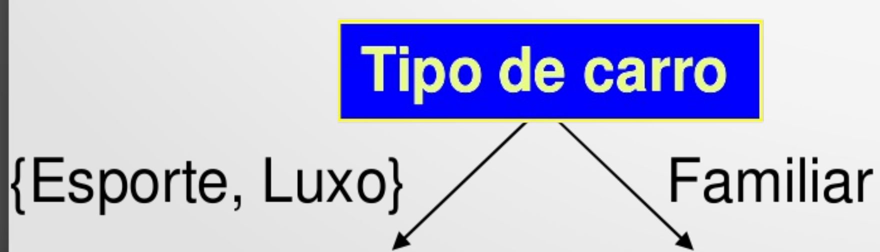


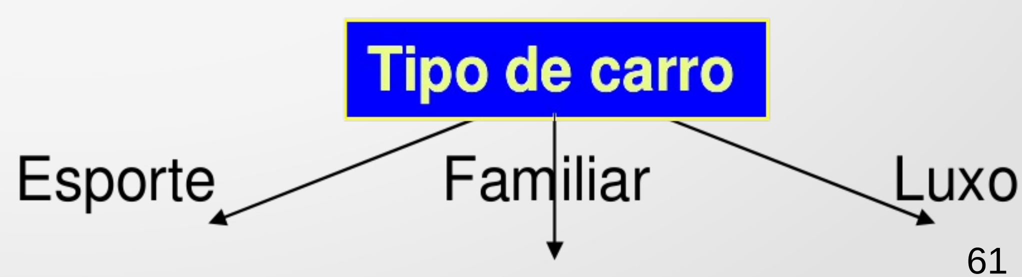
	Nó 1	*Nó 2*
C1	1	5
C2	4	2
Gini_d = 0.375		

$\text{Gini}_{\text{divisão}} = (\text{exercício})$
$= 0.375$

Divisão de Atributos Nominais

- Duas alternativas
  - Divisão binária (k = 2):
    - requer busca pela melhor binarização
    - custo computacional adicional
  - Divisão múltipla (k > 2):
    - Tende a produzir partições mais puras
    - Porém, tende a privilegiar atributos com muitos valores...





Exercício

- Definir a melhor divisão considerando divisão binária e divisão múltipla para:

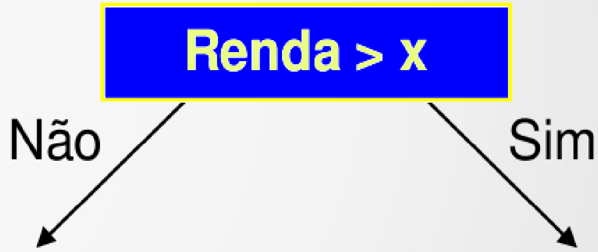
	Tipo de Carro		
	Família	Esporte	Luxo
C1	1	2	1
C2	4	1	1
$Gini_d$	???		

	Tipo de Carro	
	{Esporte, Luxo}	{Família}
C1	3	1
C2	2	4
$Gini_d$	???	

	Tipo de Carro	
	{Esporte}	{Família, Luxo}
C1	2	2
C2	1	5
$Gini_d$	???	

Divisão de Atributos Contínuos

Id	Crédito	Estado Civil	Renda	Deve
1	Sim	Solteiro	125K	Não
2	Não	Casado	100K	Não
3	Não	Solteiro	70K	Não
4	Sim	Casado	120K	Não
5	Não	Divorced	95K	Sim
6	Não	Casado	60K	Não
7	Sim	Divorced	220K	Não
8	Não	Solteiro	85K	Sim
9	Não	Casado	75K	Não
10	Não	Solteiro	90K	Sim



Divisão de Atributos Contínuos

- Por comparação
  - Vários candidatos

    para ponto de referência
    - No. possíveis divisões = no. valores distintos
- Cada valor candidato **x** possui uma matriz de contagens associada a ele
  - Contagens das classes em cada uma das duas partições ($A_i \leq$ **x** e $A_i >$ **x**)

Divisão de Atributos Contínuos

- Força Bruta
  - Método mais simples
  - Testar todos os valores **x** presentes nos dados para o atributo
    - Para cada **x**, calcular sua medida de ganho (Δ_{info} ou $Gini_d$) usando as matrizes de contagens das partições resultantes
  - Computacionalmente ineficiente:
  - $O(N^2)$
  - Trabalho repetitivo

Divisão de Atributos Contínuos

- Cálculo Cumulativo
  - Método mais eficiente: O(N log N)
  - Ordenar os valores do atributo
  - Para o menor valor
    - Calcular matriz de contagens
    - Calcular medida de ganho associada (Δ_{info} ou $Gini_d$)
  - Para cada valor, a partir do menor
    - Atualizar matriz de contagens cumulativamente
    - Calcular medida de ganho associada
  - Escolher a posição com medida de ganho ótima
    - Maior Δ_{info} ou menor $Gini_d$

Exemplo

Valores ordenados

Deve	Não	Não	Não	Sim	Sim	Sim	Não	Não	Não	Não
Renda										
	60	70	75	85	90	95	100	120	125	220

Candidatos a pto. de ref.

	55		65		72		80		87		92		97		110		122		172		230	
	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>
Sim	0	3	0	3	0	3	0	3	1	2	2	1	3	0	3	0	3	0	3	0	3	0
Não	0	7	1	6	2	5	3	4	3	4	3	4	3	4	4	3	5	2	6	1	7	0
Gini	0.420		0.400		0.375		0.343		0.417		0.400		*0.300*		0.343		0.375		0.400		0.420	

* Nota: O exemplo acima assume o uso de desigualdades estritas (< e >) no teste, por isso toma valores candidatos intermediários aos valores do atributo, ao invés desses próprios

Exemplo

Primeiro Candidato: x = 55

< 55

Classe sim: 0

Classe não: 0

Gini N1 = 0

> 55

Classe sim: 3

Classe não: 7

Gini N2 = 0.420

Ginid = 0x0 + 1x0.420 = 0.420

Exemplo

Segundo Candidato: x = 65

Atualiza distribuição do último candidato

< 65

Classe sim: 0

Classe não: 1 (0 + 1)

Gini N1 = ?

> 65

Classe sim: 3

Classe não: 6 (7 – 1)

Gini N2 = ?

Ginid = 0.400

Cálculo Cumulativo Melhorado

- Só é preciso considerar valores entre dois objetos adjacentes com classes diferentes !
  - Não – Sim ou Sim – Não
  - Reduz de 11 para 2 o número de pontos de referência candidatos no exemplo anterior !

Exemplo

Valores ordenados

Candidatos a pto. de ref.

Deve	Não	Não	Não	Sim	Sim	Sim	Não	Não	Não	Não
	Renda									
	60	70	75	85	90	95	100	120	125	220

	55		65		72		80		87		92		97		110		122		172		230	
	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>
Sim	0	3	0	3	0	3	0	3	1	2	2	1	3	0	3	0	3	0	3	0	3	0
Não	0	7	1	6	2	5	3	4	3	4	3	4	3	4	4	3	5	2	6	1	7	0
Gini	0.420		0.400		0.375		0.343		0.417		0.400		*0.300*		0.343		0.375		0.400		0.420	

* Nota: O exemplo acima assume o uso de desigualdades estritas (< e >) no teste, por isso toma valores candidatos intermediários aos valores do atributo, ao invés desses próprios

Taxa de Ganho

- Medidas como Entropia e Gini favorecem atributos com muitos valores
  - podem gerar muitos subconjuntos dos dados de treinamento
  - subconjuntos menores tendem a ser mais puros
  - porém, são mais susceptíveis a se especializar nos dados de treinamento
    - preditores ruins da função de classificação para dados não vistos
    - exemplo extremo: no. do RG ou CPF para classificação de risco de crédito

Taxa de Ganho

- Alternativas para minimizar este problema
  - Usar apenas divisões binárias (abordagem usada pelo algoritmo CART)
  - Usar alguma punição para a quantidade de valores do atributo
    - Exemplo: Taxa de Ganho (abordagem usada pelo algoritmo C4.5)

Taxa de Ganho

- Definida para a Entropia:

$$\text{Taxa de Ganho} = \frac{\Delta_{\text{info}}}{S_{\text{info}}}$$

$$S_{\text{info}} = -\sum_{t=1}^{k} p(v_t)\log_2 p(v_t)$$

onde:

k: número de divisões (valores do atributo A_i)

$p(v_t)$ = fração de objetos cujo valor do atributo $A_i = v_t$

S_{info} = entropia do conjunto de objetos com relação aos valores do atributo A_i

↑ no. objetos com valores distintos ↑ Sinfo ↓ taxa de ganho

Viés Indutivo

- Informalmente, o viés (*bias*) indutivo de um algoritmo é uma tendência deste em privilegiar uma ou um conjunto de hipóteses frente às demais
- Pode ser caracterizado como:
  - De restrição (ou linguagem): restringe o espaço de hipóteses
  - De busca (ou preferência): polariza a escolha dentre as possíveis realizações do modelo

Overfitting

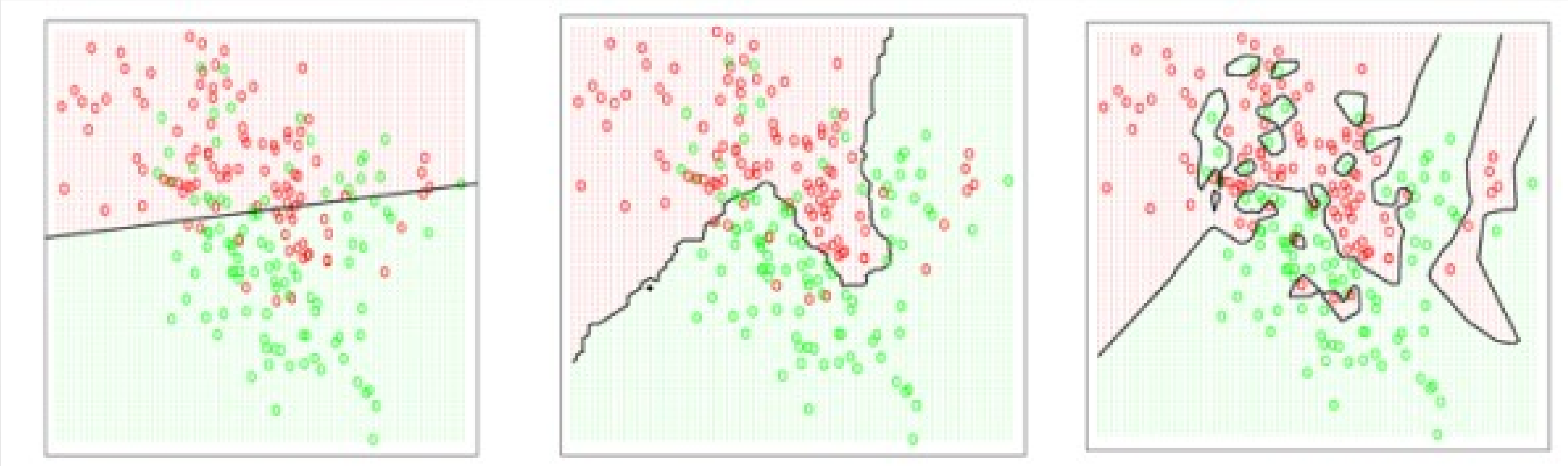
- Ausência de viés restritivo em ADs significa que pode haver um ajuste perfeito aos dados de treinamento
  - Chamado de *overfitting*
  - Problema:
    - o conjunto de dados não for muito representativo
    - contaminado com ruído

Overfitting

Overfitting

- Partição recursiva dos dados:
  - Decisões são baseadas em conjuntos cada vez menores de objetos
  - Níveis mais profundos podem ter muito poucos objetos
    - Presença de ruído afeta cada vez mais a decisão para esses nós
    - Reduz capacidade de generalização (desempenho em objetos não vistos)

Generalização e *Overfitting*



Fonte: https://mathbabe.org/2012/11/20/columbia-data-science-course-week-12-predictive-modeling-data-leakage-model-evaluation/

Overfitting

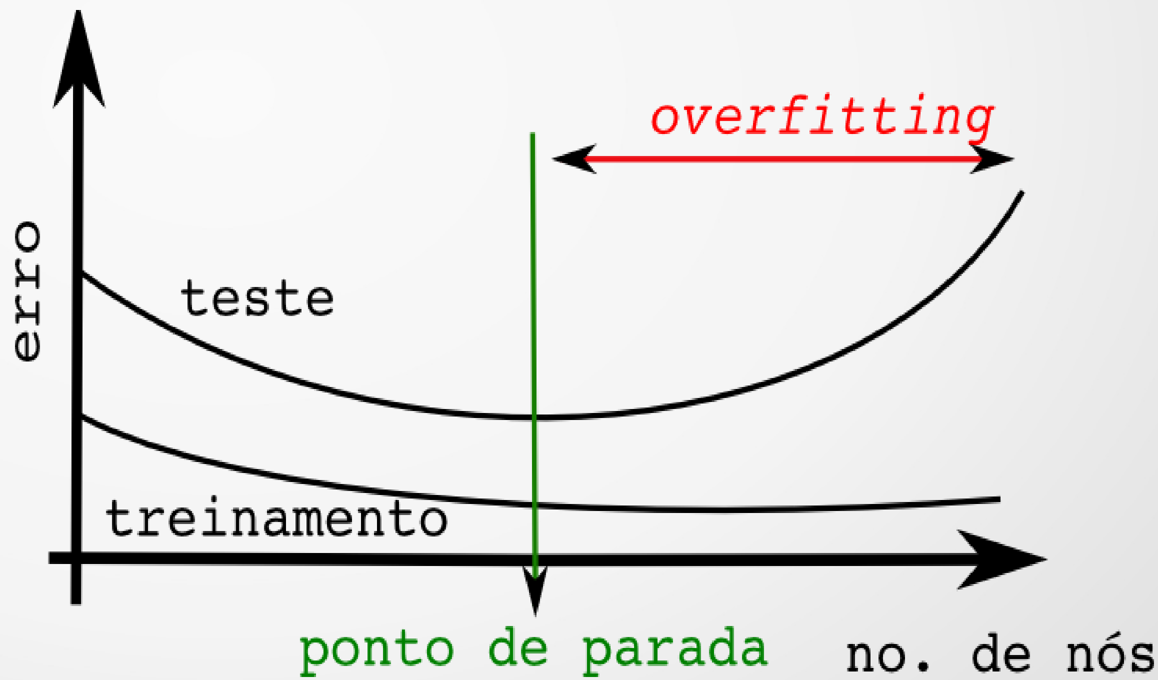
- Navalha de Occam (*Occam's razor*)
- Quanto mais simples a solução, melhor
- Explicação dos dados por uma hipótese mais complexa pode ser apenas uma coincidência
- Árvore de decisão pode ser simplificada...

Simplificação de ADs

- Duas Abordagens:
  - Interromper a priori o crescimento da árvore
    - com base no desempenho em dados de teste
    - com base em um compromisso entre desempenho em dados de teste e complexidade do modelo
  - Podar o modelo a posteriori

Interrupção do Crescimento

- Quando o desempenho em dados não usados no treinamento não mais melhora de forma significativa
- Muito comum
  em redes neurais,
  não se mostra a
  melhor
  abordagem
  em ADs



Poda da Árvore

- Elimina sub-árvores cujas existências reduzem o desempenho do modelo em dados de teste
- Demanda calcular, para cada nó, a variação no desempenho de classificação após a

  eliminação dos seus

  descendentes
- Mais eficaz que interrupção,

  com maior custo computacional

Poda da Árvore

1) Tomar um conjunto de dados de teste (não vistos no treinamento)
2) Percorrer a árvore segundo percurso pós-fixado. Para cada nó t, calcular:
    1) E_t: o erro de classificação dos objetos de teste que chegam até aquele nó, como se o nó fosse uma folha associada à classe da maioria
    2) S_t: a soma dos erros E_i de cada uma das folhas descendentes de t
    3) V_t: variação do erro após poda dos descendentes ($V_t = S_t – E_t$)
3) Podar os descendentes daquele nó t com maior valor positivo de V_t
4) Atualizar os valores de S_t e V_t para os nós ancestrais de t
5) Retornar ao passo 3 enquanto houver valores positivos de V_t

Vantagens das ADs

- Rápida classificação de novos dados
- Interpretação da hipótese induzida
  - Fácil para árvores relativamente pequenas
    - ou seja, com poucas regras...
- Determina quais atributos são importantes
  - Seleção de atributos embarcada !!!
    - Pode ser estendida para também levar em conta o custo da utilização de cada atributo...

Vantagens das ADs

- Principais algoritmos tratam tanto atributos categóricos como atributos numéricos
- Desempenho muitas vezes comparável ou até superior a outros bons classificadores
  - depende da natureza dos dados
- Algoritmos podem ser adaptados para tratar instâncias com valores ausentes (e.g. C4.5)

Desvantagens das ADs

- Limitação de hipótese a hiper-retângulos
  - Exceto para Árvores Oblíquas (mais complexas...)
- Baixo desempenho em problemas com muitas classes e poucos dados
  - Representatividade das regiões hiper-retangulares...
- Custo computacional de indução e simplificação do modelo pode ser elevado
  - especialmente para os algoritmos mais sofisticados

Algoritmos

- ID3 : trata atributos categóricos
  - Desenvolvido por Quinlan (1993)
  - Usa ganho de informação para a seleção
  - Faz simplificação por poda
- C4.5: sucessor do ID3 manipula qualquer atributo
  - Download em:

  http://www.cse.unsw.edu.au/~quinlan/c4.5r8.tar.gz

Outros Algoritmos

- J4.8 : Versão Java do C4.5 implementada no software Weka
- PART : Variante do J4.8 também disponível no Weka
- C5.0: Sucessor do C4.5 comercial
- CART (Classification and Regression Trees): Árvores Oblíquas, ou seja, hipóteses não mais restritas a partições hiper-retangulares

Aplicando Árvore com *holdout* Iris

- Código

```
import pandas as pd
#Importa o método de Árvores de Decisão do Sklearn
from sklearn.tree import DecisionTreeClassifier
# Localização do arquivo
filepath = 'data/Iris_Data.csv'
# Importando os dados
data = pd.read_csv(filepath)
#Colocando os dados em ordem aleatória
randomdata = (data.sample(n=150, replace=False))
#Aplicando hold out
traindata = randomdata.iloc[:135,:]
testdata = randomdata.iloc[135:,:]
#Cria uma instância de classe
DTC = DecisionTreeClassifier(criterion='gini', max_features=4, max_depth=5)
#Ajusta a Árvore aos dados de treino
DTC = DTC.fit(traindata.iloc[:,0:4], traindata.iloc[:,4])
#Classe real
print(testdata.iloc[:,4])
#Classe predita
print(DTC.predict(testdata.iloc[:,0:4]))
```

Aplicando Árvore com *holdout* Iris

- Saída = Classes

```
113      Iris-virginica
104      Iris-virginica
13          Iris-setosa
51      Iris-versicolor
34          Iris-setosa
86      Iris-versicolor
109      Iris-virginica
99      Iris-versicolor
96      Iris-versicolor
138      Iris-virginica
3           Iris-setosa
82      Iris-versicolor
91      Iris-versicolor
27          Iris-setosa
52      Iris-versicolor
```

- Saída Predita

```
['Iris-virginica'
'Iris-virginica'
'Iris-setosa'
'Iris-versicolor'
 'Iris-setosa'
'Iris-versicolor'
'Iris-virginica'
'Iris-versicolor'
 'Iris-versicolor'
'Iris-virginica'
'Iris-setosa'
'Iris-versicolor'
 'Iris-versicolor'
'Iris-setosa'
'Iris-virginica']
```

Exercício

- Seja o seguinte cadastro de pacientes:

Nome	Febre	Enjôo	Manchas	Dores	Diagnóstico
João	sim	sim	pequenas	sim	doente
Pedro	não	não	grandes	não	saudável
Maria	sim	sim	pequenas	não	saudável
José	sim	não	grandes	sim	doente
Ana	sim	não	pequenas	sim	saudável
Leila	não	não	grandes	sim	doente

Exercício

- Usando medida de entropia:
  - Induzir uma árvore de decisão capaz de distinguir:
    - Pacientes potencialmente saudáveis
    - Pacientes potencialmente doentes
  - Testar a árvore para novos casos
    - (Luis, não, não, pequenas, sim)
    - (Laura, sim, sim, grandes, sim)

Exercício Computacional

- Baixar da UCI as bases de dados IRIS e GLASS
  - Induzir uma árvore de decisão
  - Dividir os dados da seguinte forma:
    - 50% para treinamento
    - 25% para teste
    - 25% para validação
  - Fazer o mesmo para outros algoritmos e comparar resultados