Sistemas Inteligentes

K-NEAREST NEIGHBORS

k-nearest neighbors

Ideia:

- Exemplos similares devem possuir rótulos similares
- Classificar os novos exemplos como exemplos do treinamento similares

Algoritmo:

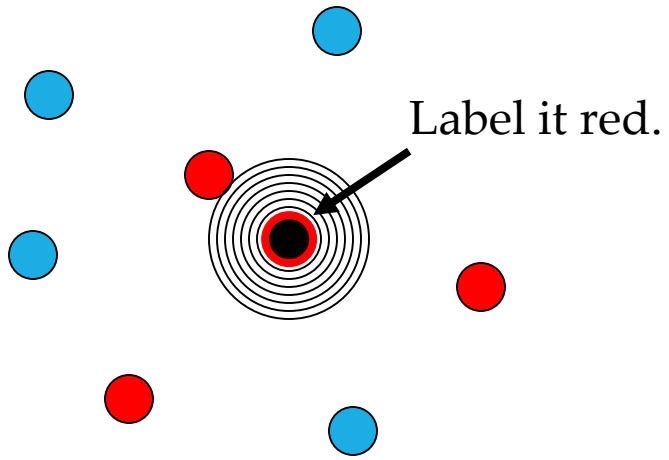
- Para um dado $\mathbf{x}$, determinar a qual classe ele pertence da seguinte forma:
  - Encontre o dado de treinamento que mais se assemelhar a $\boldsymbol{x}$;
  - Classifique x com o mesmo rótulo do dado de treinamento

Questões:

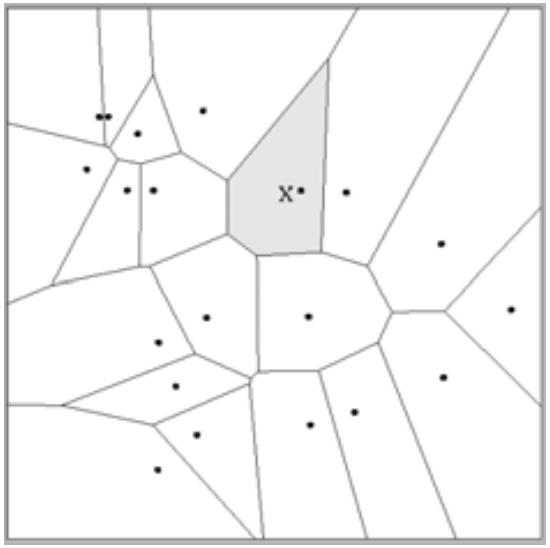
- Como quantificar a similaridade?
- Quantos dados de treinamento devo considerar?
- Como resolver inconsistências?

1-Vizinho mais próximo

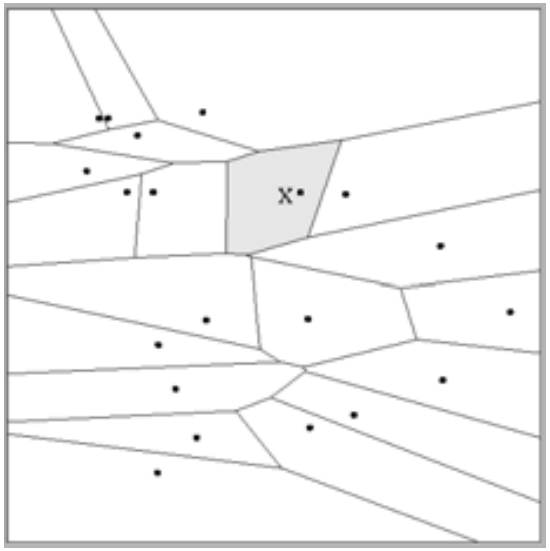
Um dos classificadores mais simples, que explora a ideia de classificar o novo dado de acordo com o dado de treinamento mais próximo a ele.



Métricas de distância



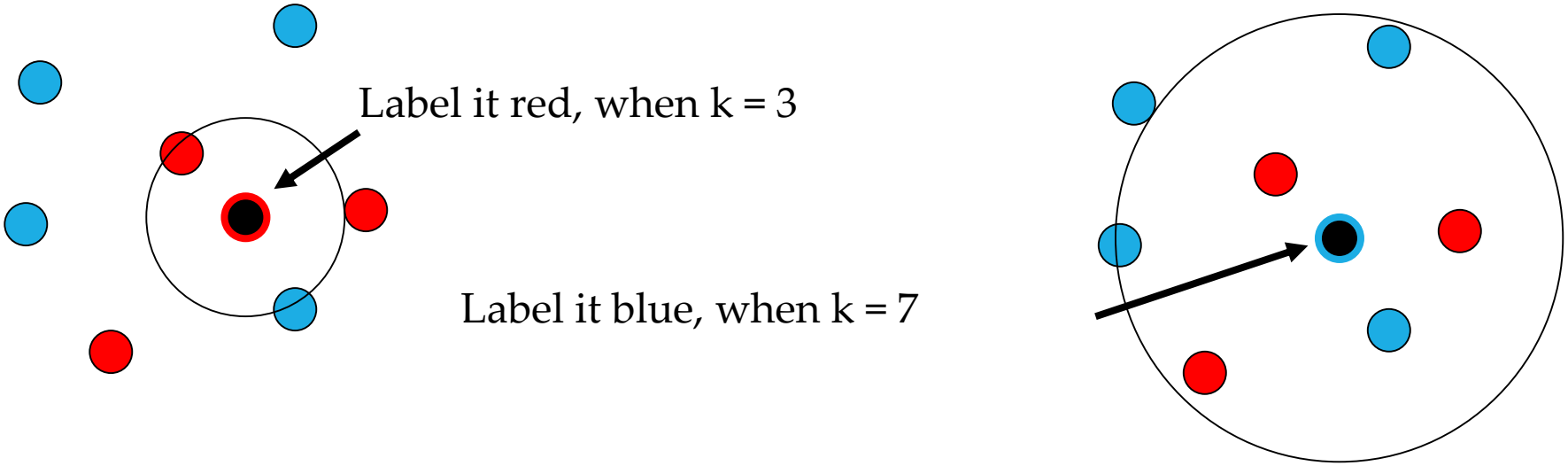
$\text{Dist}(\mathbf{a},\mathbf{b}) = (a_1 - b_1)^2 + (a_2 - b_2)^2$



$\text{Dist}(\mathbf{a},\mathbf{b}) = (a_1 - b_1)^2 + (3a_2 - 3b_2)^2$

k – vizinhos mais próximos

Generalização do método anterior. O novo dado é classificado de acordo com o rótulo mais frequente dos seus k vizinhos mais próximos.



Vantagens e Desvantagens

Simples de implementar e possui poucos parâmetros:

- Aumentar o número de vizinhos pode ser benéfico para reduzir a sensibilidade ao ruído nos dados. Por outro lado, aumentar o número de vizinhos torna o processo de classificação mais custoso
- Regra geral: escolha um valor k não muito grande, mas também não muito pequeno

Necessita de uma medida de distância que seja compatível com a noção de similaridade desejada para a tarefa de classificação.

Pode ser proibitivo utilizar o algoritmo para grandes bases de dados, pois a classificação deve comparar os dados com todos os exemplos de treinamento.

Precisão na predição cai rapidamente com o aumento no número de atributos considerados.

Sistemas Inteligentes

DECISION TREES

Exemplo de classificação

	binary	categorical	continuous	class
Tid	**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

Figure 4.6. Training set for predicting borrowers who will default on loan payments.

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Test Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

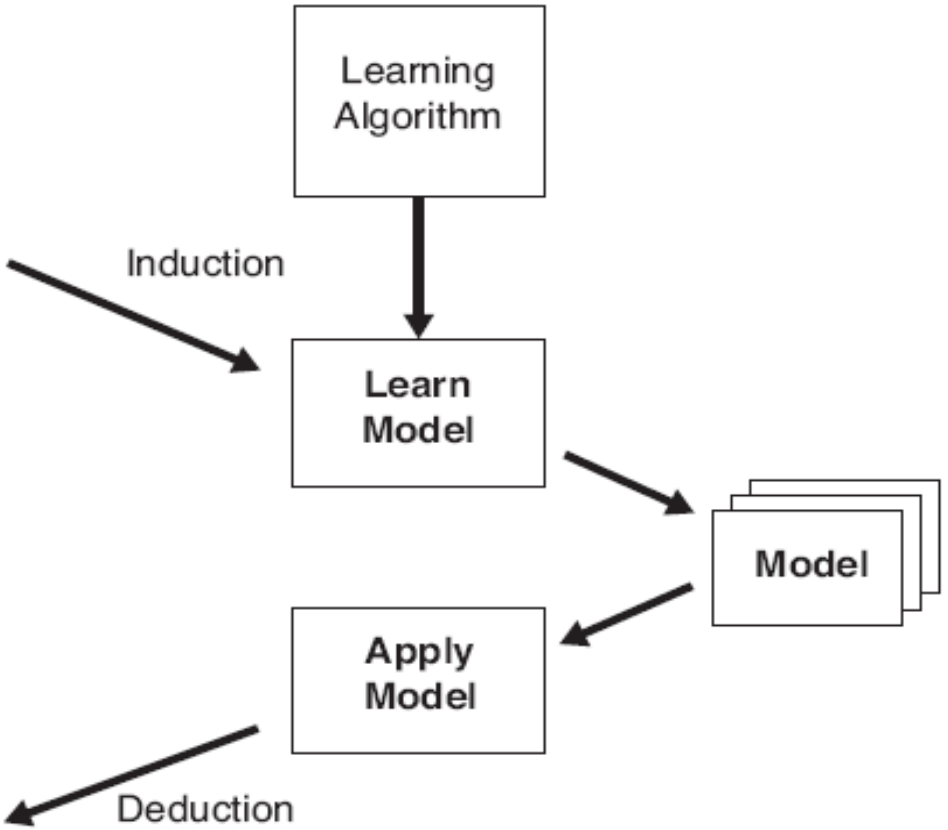


Figure 4.3. General approach for building a classification model.

Decision Trees

Classificador cujo modelo segue uma estrutura de árvore

- Um nó interno denota um teste a ser realizado sobre um determinado atributo
- Uma derivação representa um resultado do teste
- As folhas representam os rótulos das classes

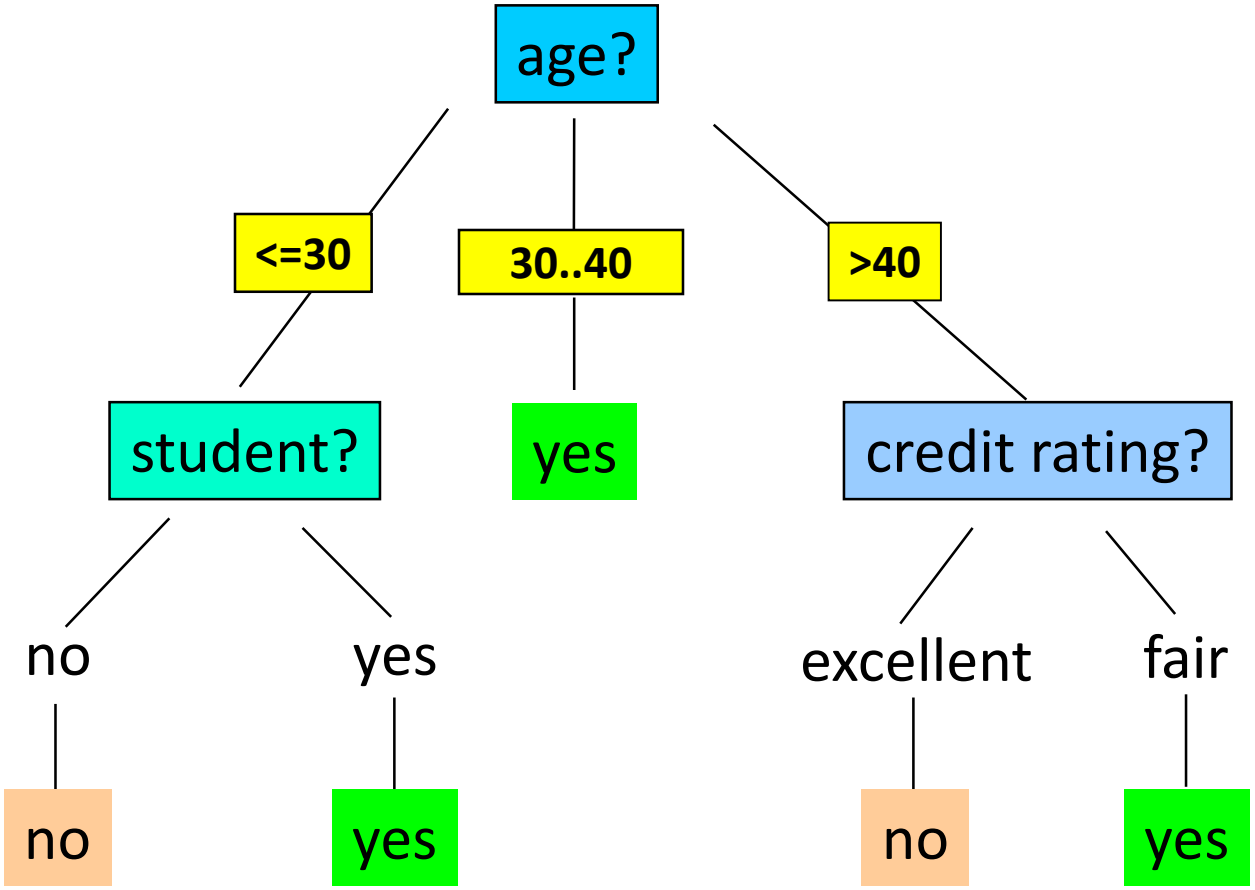
A geração da árvore de decisão consiste em duas etapas

- **Construção da árvore**
  - No início, todos os exemplos de treinamento estão no nó raiz
  - Particionam-se os exemplos de maneira recursiva, baseado em determinados atributos
- **Poda da árvore**
  - Identificar e remover ramos que refletem o ruído ou *outliers*

Dados para treinamento para árvore de decisão

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Decision Tree "*buys_computer*"



Algoritmo de Hunt

X_t: conjunto de dados de treinamento no nó t

$y = \{y_1, \ldots, y_c\}$: rótulos das classes

Passo 1: Se todos os dados em X_t pertencem a uma mesma classe y_t, então t é um nó folha e deve ser rotulado como y_t

Passo 2: Se X_t contém dados que pertencem a mais de uma classe

- Selecione um atributo que será utilizado no teste para particionar os dados em subconjuntos menores
- Crie um nó filho para cada possível resultado do teste
- Aplique o algoritmo recursivamente para cada nó filho

Construção da *Decision Tree* - exemplo

	binary	categorical	continuous	class
Tid	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

Figure 4.6. Training set for predicting borrowers who will default on loan payments.

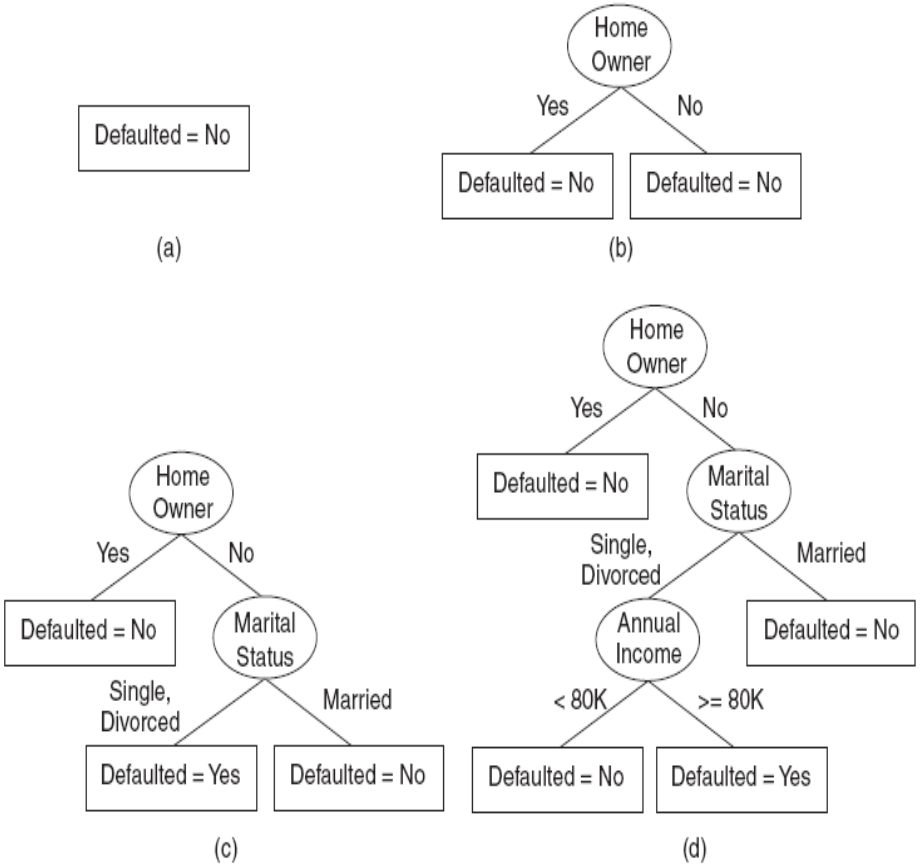


Figure 4.7. Hunt's algorithm for inducing decision trees.

Bifurcação de Nó

Atributos Binários

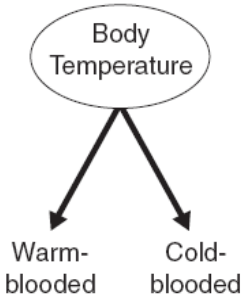


Figure 4.8. Test condition for binary attributes.

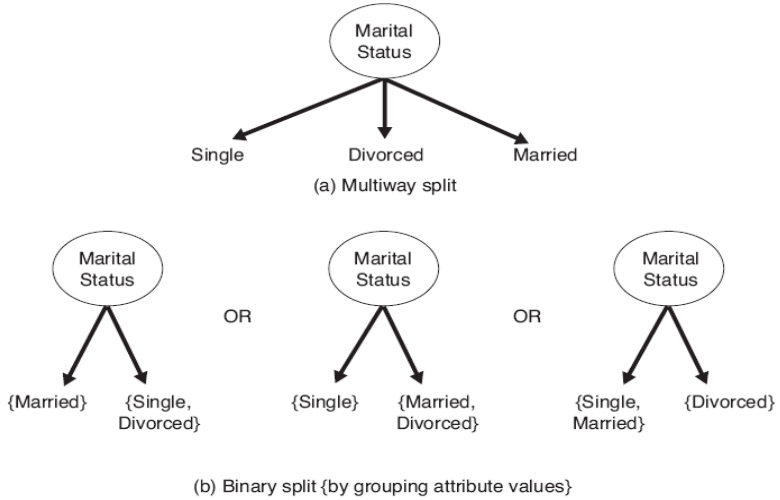


Figure 4.9. Test conditions for nominal attributes.

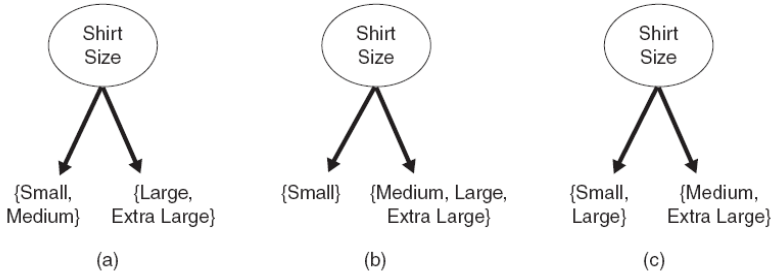


Figure 4.10. Different ways of grouping ordinal attribute values.

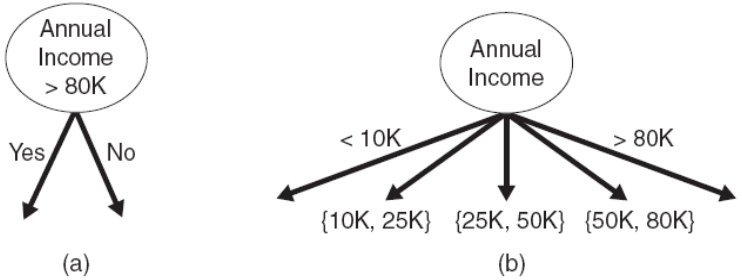


Figure 4.11. Test condition for continuous attributes.

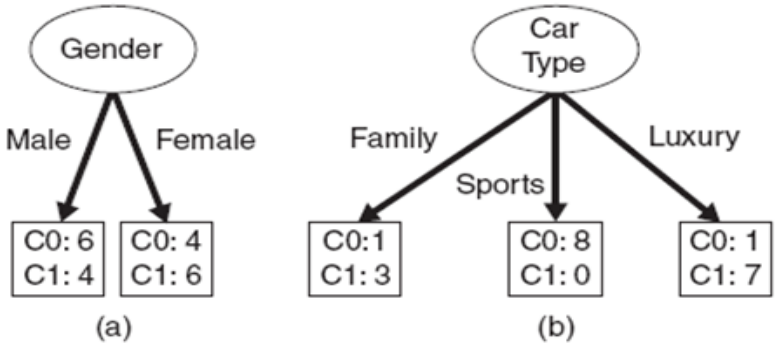
Como selecionar a bifurcação?

Considere que

$p(i|t)$: fração dos dados no nó t que pertencem à classe i

A melhor bifurcação é selecionada com base no grau de **impureza** de seus nós filhos

- Uma distribuição de classes (0,1) possui alta pureza
- Uma distribuição de classes (0.5,0.5) possui a menor pureza (maior impureza)



Quais são as medidas de impureza?

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

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

$$\text{Classification error}(t) = 1 - \max_i \left[p(i \mid t) \right]$$

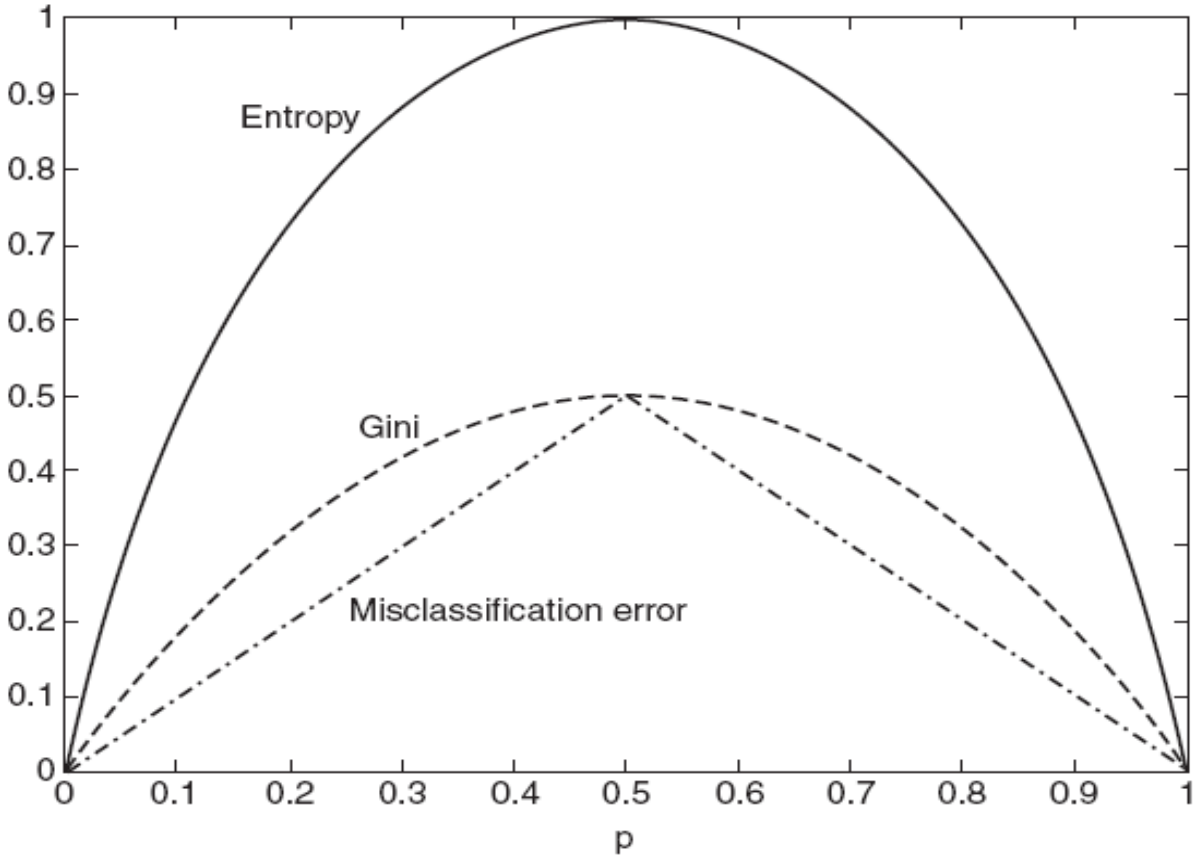


Figure 4.13. Comparison among the impurity measures for binary classification problems.

Medidas de Impureza

Ganho de um teste: compare a impureza de um nó pai com a impureza dos nós filhos

$$\Delta = I(parent) - \sum_{j=1}^{k} \frac{N(v_j)}{N} I(v_j)$$

Maximizar o ganho = minimizar a impureza ponderada dos nós filhos

- Se I(pai) = Entropy(pai), então a quantidade Δ_{info} é denominada de ganho de informação

Calculando o ganho - exemplo

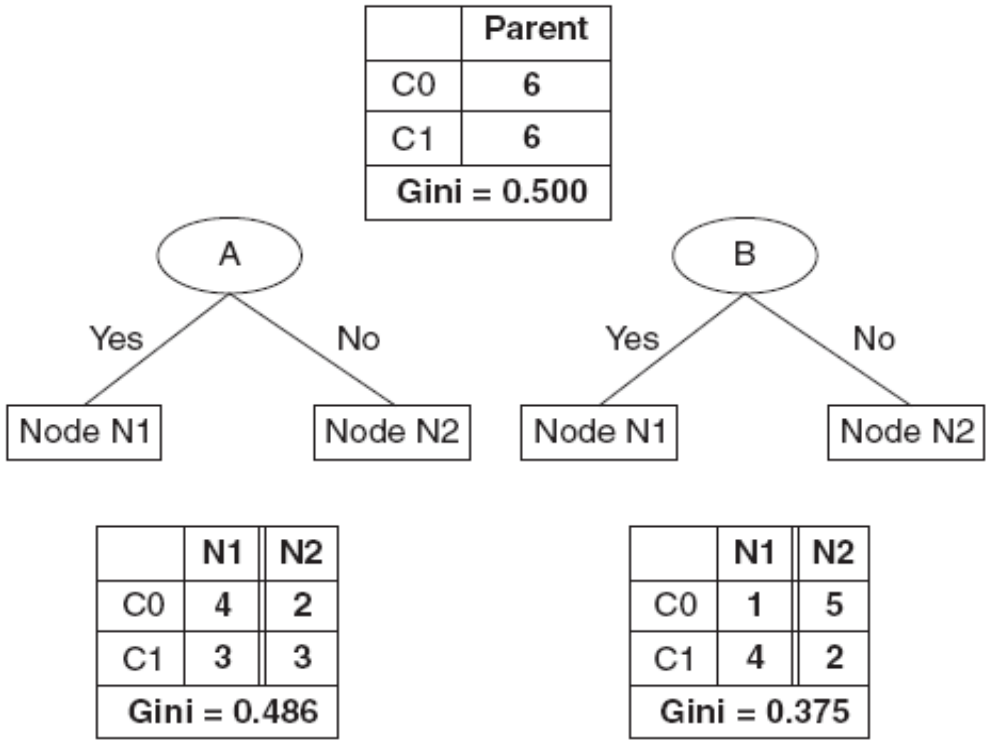


Figure 4.14. Splitting binary attributes.

A estratégia é boa para se determinar a bifurcação?

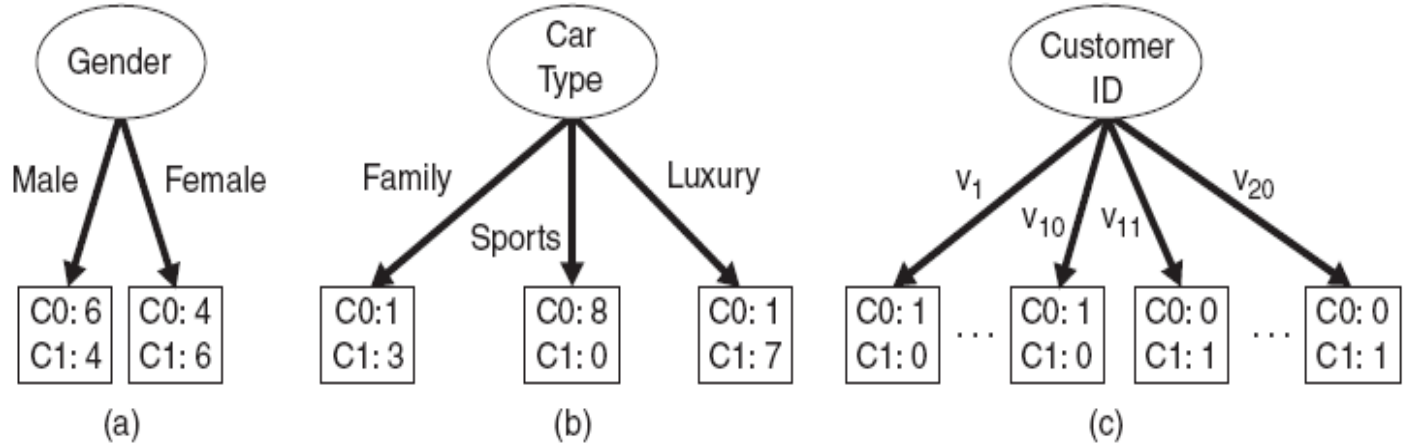


Figure 4.12. Multiway versus binary splits.

Medidas baseadas apenas na impureza favorecem atributos com grande número de valores! E uma condição de teste com vários possíveis resultados pode não ser apropriada, pois o # de padrões em cada partição é muito pequeno para se fazer predições.

Alternativa: usar o *Gain Ratio*

O *Gain Ratio* é definido como

$$Gain\ ratio\ = \frac{\Delta_{info}}{Split_{info}}$$

onde

$$Split_{Info}\ =\ -\Sigma_{i=1\ldots k} p(v_i)\log\big(p(v_i)\big)$$

para

- k: número total de bifurcações

Se cada atributo possui o mesmo número de padrões, $Split_{Info}\ = \log k$

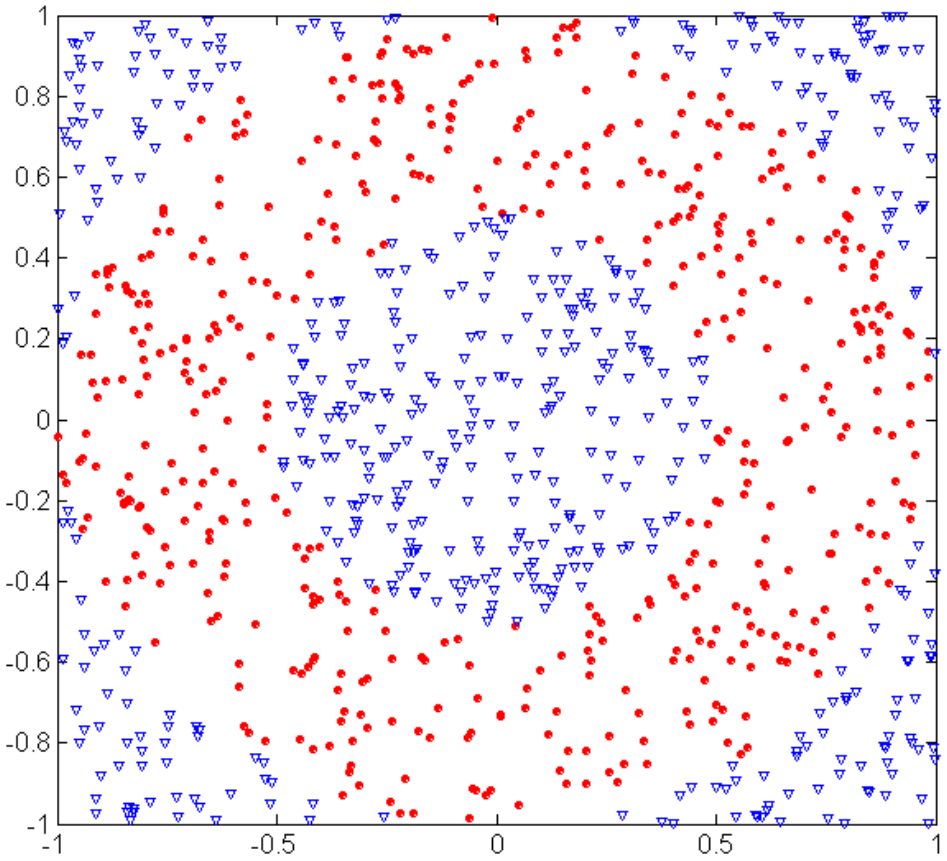
- Quanto maior o número de bifurcações → maior o valor de $Split_{info}$ → menor o *Gain Ratio*

Construindo Decision Trees (pseudo código)

GenDecTree(Sample **S**, Features **F**)

1. **If stopping_condition(S,F)** = true **then**
   a. **leaf = createNode()**
   b. **leaf.label= Classify(S)**
   c. **return leaf**
2. **root = createNode()**
3. **root.test_condition = findBestSplit(S,F)**
4. **V = {v| v** is a possible outcome of **root.test_condition}**
5. **for *each*** value **vϵV**:
   a. S_v**: = {s | root.test_condition(s)** = **v** and **s ϵ S}**;
   b. **child = TreeGrowth(**S_v **,F)** ;
   c. Add **child** as a descent of **root** and label the edge **(root→child) as v**
6. **return root**

Exemplo de aplicação



500 círculos and 500 triângulos (dados).

Círculos:

$0.5 \leq \text{sqrt}(x_1^2+x_2^2) \leq 1$

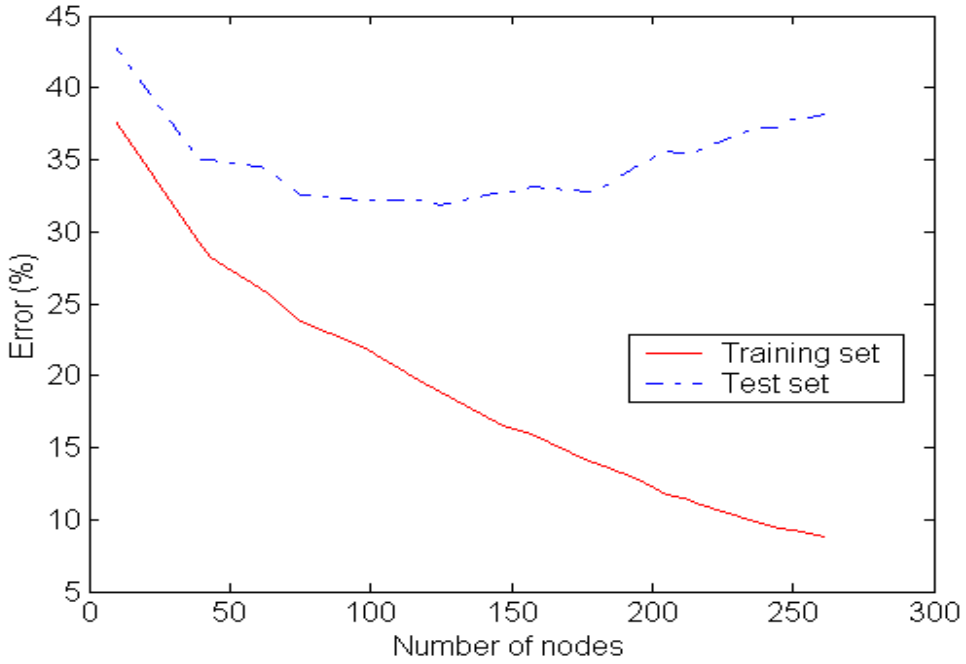
Triângulos

$\text{sqrt}(x_1^2+x_2^2) > 1$ or

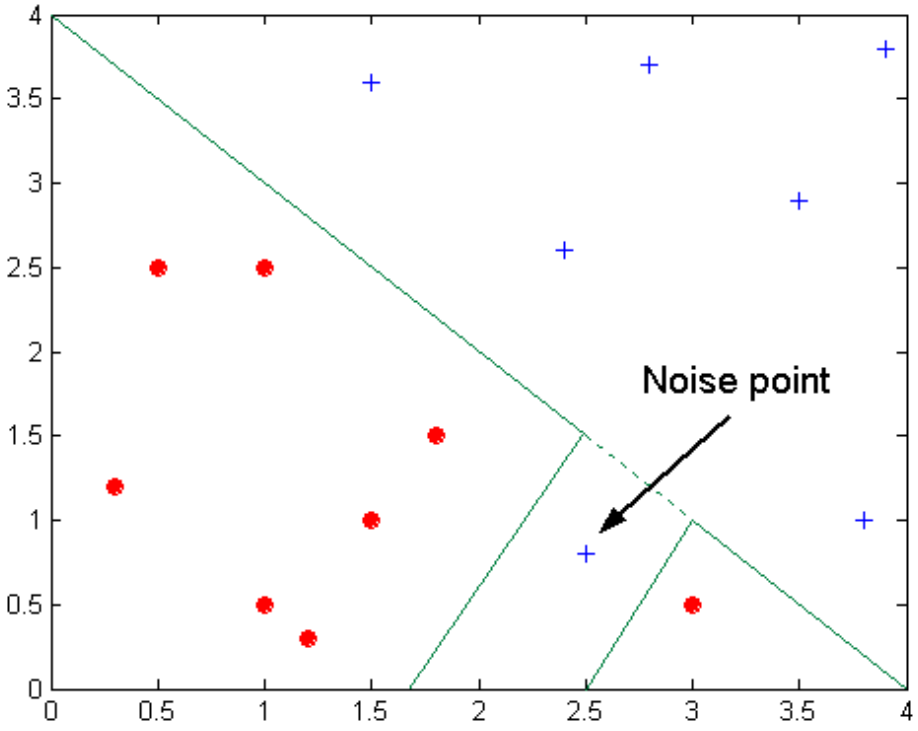
$\text{sqrt}(x_1^2+x_2^2) < 0.5$

Underfitting x *Overfitting*

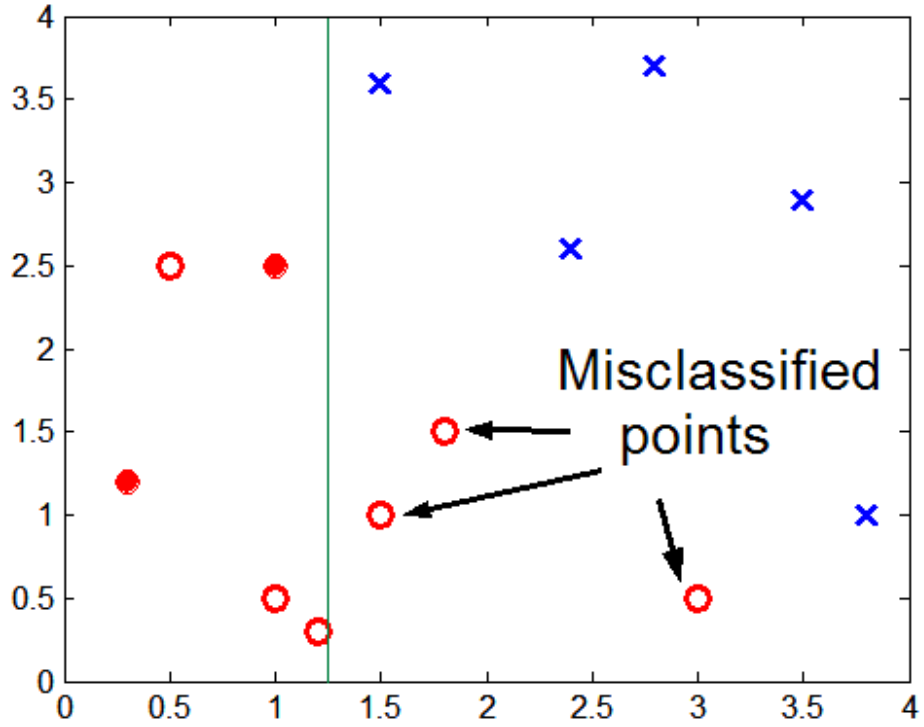
Quando o modelo é muito simples, não é capaz de reduzir o erro de treinamento, tampouco o de teste. Quando há muita flexibilidade no modelo é necessário tomar cuidado para não obter uma solução com *overfitting*.



Overfitting devido ao ruído



Erros devido a pouco treinamento



Overfitting: como lidar com isso?

Dados dois modelos com erro de generalização similares, deve-se dar preferência ao modelo mais simples → Navalha de Occam

- Modelos mais complexos têm maior chance de gerar *overfitting*
- Necessário considerar a complexidade do modelo na sua avaliação

Outra abordagem: poda após geração da árvore

- Obtenha a árvore de decisão completa
- Podar os nós da árvore de decisão em uma abordagem "*bottom-up*"
- Se o erro de generalização melhorar depois da poda, substitua a sub árvore podada por um nó folha, cuja classe é determinada pela classe da maioria dos padrões associadas à sub árvore
- Algoritmos como o "*Minimum Description Length*" (MDL) podem ser utilizados nesse processo
- Pode-se realizar a poda enquanto a árvore está sendo criada também (*pre-prunning*)

Avaliando os Modelos

Uma maneira usual de representar a performance do classificador é através da ***Confusion matrix***

		Predicted Class	
		Class = 1	**Class = 0**
Actual Class	**Class = 1**	**TP**	**FN**
	Class = 0	**FP**	**TN**

TP: True Positive; TN: True Negative; FP: False Positive; FN: False Negative

Diferentes tipos de erro podem ser mais ou menos relevantes, dependendo da aplicação

- Por exemplo, no diagnóstico automático de câncer a partir de imagens, o que é mais importante: TP ou FN?

Acurácia, Taxa de Erro, Sensibilidade e Especificidade

Acurácia: porcentagem dos dados de teste que foram corretamente classificados.

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP}$$

Taxa de Erro: 1 – acurácia, ou

$$Error\ rate = \frac{FP + FN}{TP + TN + FN + FP}$$

A\P	C	¬C	
C	**TP**	**FN**	**P**
¬C	**FP**	**TN**	**N**
	P'	**N'**	**All**

Se houver um desbalanço na quantidade de dados – por exemplo, uma das classes é rara, as métricas anteriores podem ocultar informações sobre o desempenho do classificador

Alternativas:

Sensibilidade: Taxa de True Positive

$$Sensitivity = \frac{TP}{TP + FN}$$

Especificidade: Taxa de True Negative

$$Specificity = \frac{TN}{FP + TN}$$

Precisão, Recall e medida F

Precisão: porcentagem de dados classificados como positivos que são efetivamente positivos

$$precision = \frac{TP}{TP + FP}$$

Recall: porcentagem dos dados positivos que o classificador classificou corretamente como positivos

$$recall = \frac{TP}{TP + FN}$$

Medida F (ou F-score): media harmônica entre a precisão e o recall

$$F = \frac{2 \times precision \times recall}{precision + recall}$$

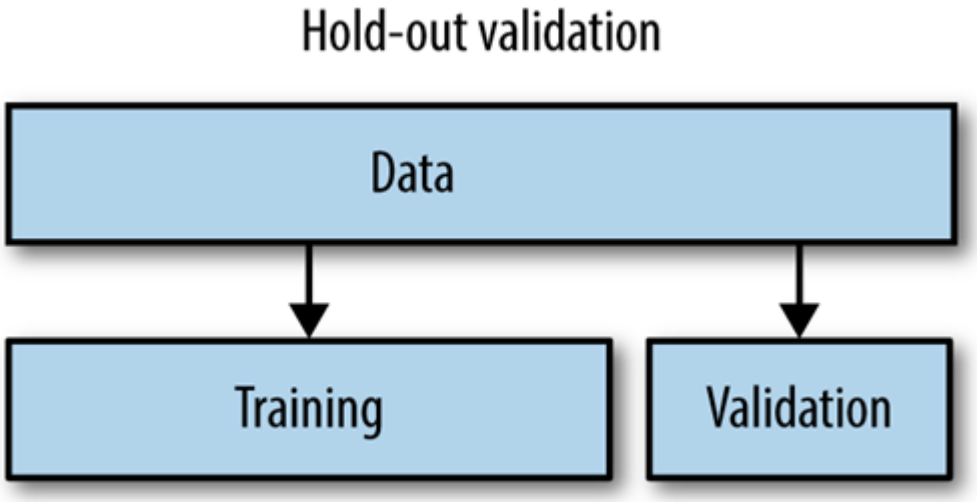
Exemplo

- Precision = 90/230 = 39.13% Recall = 90/300 = 30.00%

Actual Class\Predicted class	cancer = yes	cancer = no	Total	Recognition(%)
cancer = yes	**90**	**210**	300	30.00 (*sensitivity*
cancer = no	**140**	**9560**	9700	98.56 (*specificity)*
Total	230	9770	10000	96.40 (*accuracy*)

Método de validação *Holdout*

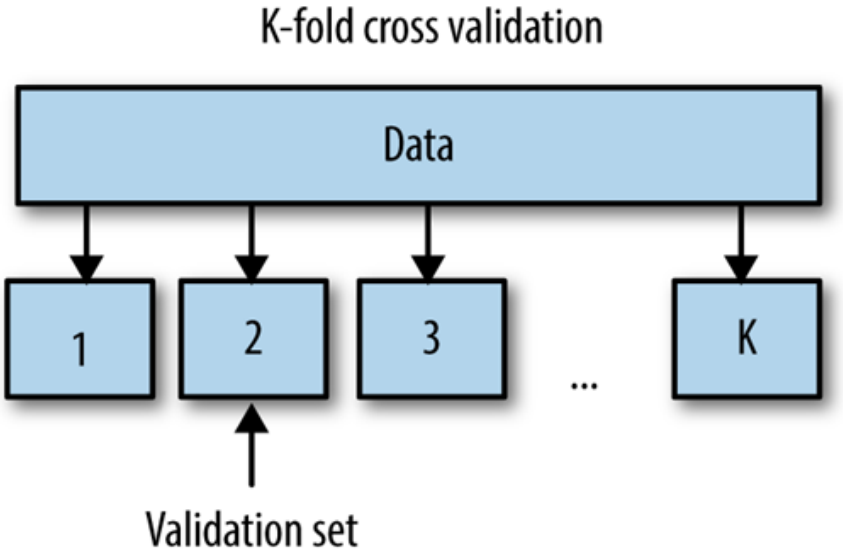
- Dados são divididos em dois conjuntos
  - Treinamento (e.g., 2/3), que é utilizado para construção do modelo
  - Teste (e.g., 1/3) para avaliação de desempenho
- Amostragem aleatória: variação do *holdout*
  - Repetir o método *holdout* k vezes, dividindo aleatoriamente os conjuntos. O desempenho global será a média de todos os desempenhos.



K-fold Cross-Validation

Particionar os dados em k conjuntos mutuamente exclusivos, de mesmo tamanho

- A cada iteração, utilize um dos subconjuntos como teste (os demais como dados de treinamento). Desempenho global será a média dos desempenhos
- Leave-one-out: caso particular da validação k-fold, em que k é igual o número de dados disponíveis



Bootstrap Validation

Funciona bem para conjunto de dados pequenos

- Dados de treinamento são amostrados aleatoriamente a partir dos dados disponíveis (com substituição), i.e., um mesmo dado pode ser sorteado mais de uma vez
- Os dados que não forem sorteados formam o conjunto de teste