

# Aula 04- Classificação

1001524 – Aprendizado de Máquina I  
2023 - Turmas A, B, C  
Prof. Dr. Murilo Naldi

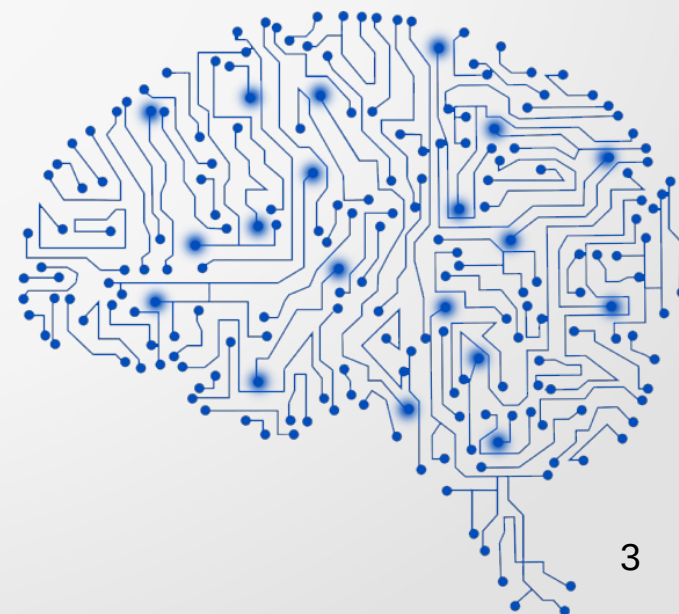
[naldi@ufscar.br](mailto:naldi@ufscar.br)

# Agradecimentos

- Parte do material utilizado nesta aula foi cedido pelos professores André C.P.L.F de Carvalho e Ricardo J.G.B. Campello e, por esse motivo, o crédito deste material é deles
- Parte do material utilizado nesta aula foi disponibilizado por M. Kumar no endereço:
  - [www-users.cs.umn.edu/~kumar/dmbook/index.php](http://www-users.cs.umn.edu/~kumar/dmbook/index.php)
- Agradecimentos a Intel Software e a Intel IA Academy pelo material disponibilizado e recursos didáticos

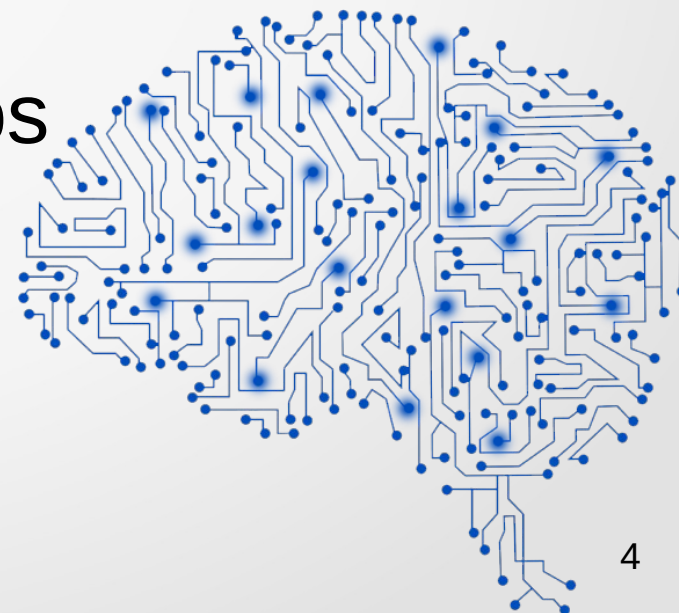
# Aprendizado de máquina

- Vimos que aprendizado de máquina permite que computadores possam aprender e inferir através dos dados
  - Com diversas aplicações
    - Filtro de Spam
    - Busca na rede
    - Roteamento
    - Detecção de fraude
    - Recomendação de conteúdo
    - Reconhecimento de padrões



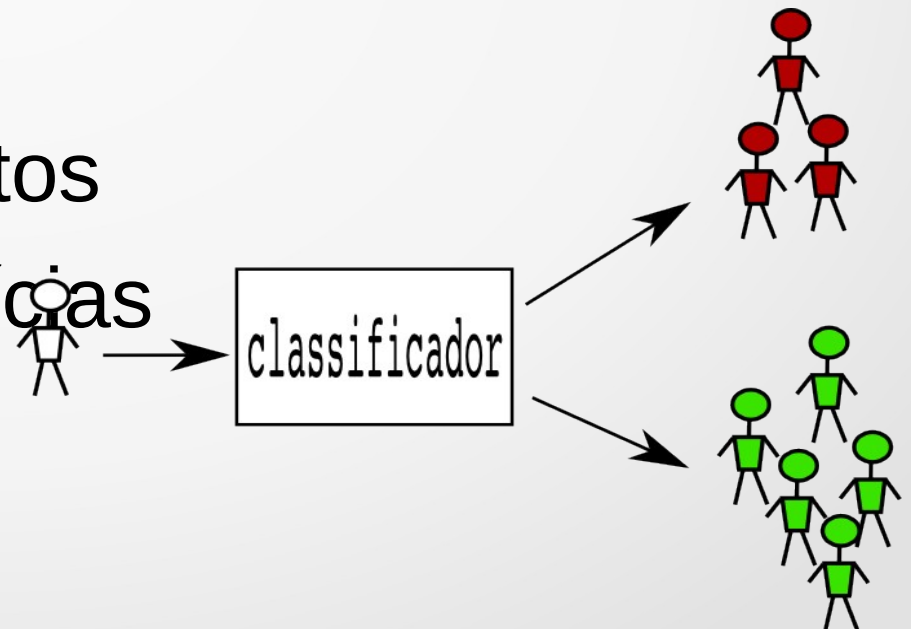
# Aprendizado de máquina

- Dentre as tarefas podemos citar
  - Métodos supervisionados
    - Classes são conhecidas
      - Classificação
      - Regressão
  - Métodos não supervisionados
    - Não se sabe a estrutura
      - Agrupamento



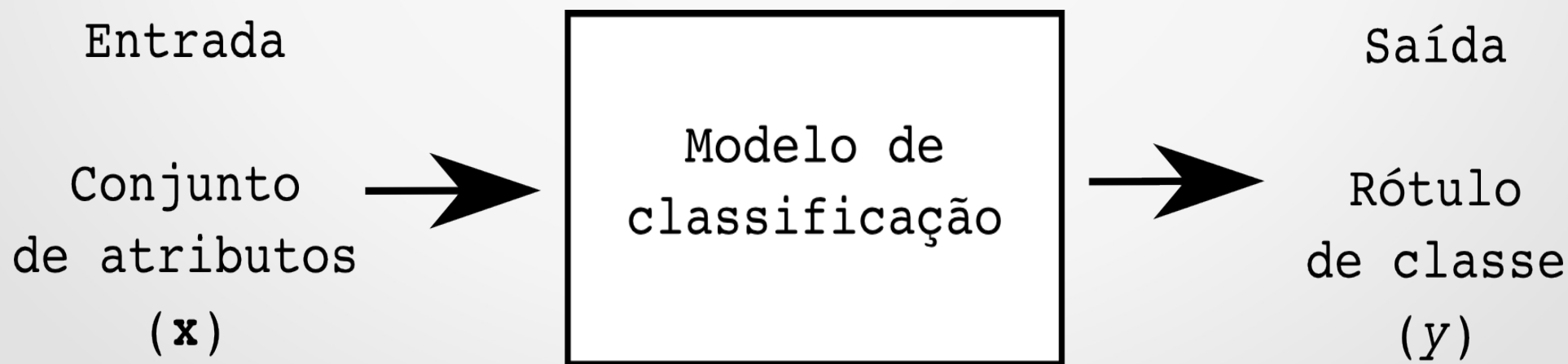
# Classificação

- Consiste na tarefa de organizar objetos em categorias pré-definidas
- Exemplo:
  - Perfis de cliente
  - Tipos de paciente
  - Categorias de produtos
  - Classificação de notícias



# Modelo de classificação

- Mapeia diferentes conjuntos de atributos  $\mathbf{x}$  para uma das classes pré-definidas classe  $y$
- Pode ser **descritivo** (resumo dos dados) ou **preditivo** (prevê o rótulo de um dado não conhecido)



# Vocabulário

- A seguir, veremos alguns nomes utilizados nas tarefas de classificação

sepal length	sepal width	petal length	petal width	species
6.7	3.0	5.2	2.3	virginica
6.4	2.8	5.6	2.1	virginica
4.6	3.4	1.4	0.3	setosa
6.9	3.1	4.9	1.5	versicolor
4.4	2.9	1.4	0.2	setosa
4.8	3.0	1.4	0.1	setosa
5.9	3.0	5.1	1.8	virginica
5.4	3.9	1.3	0.4	setosa
4.9	3.0	1.4	0.2	setosa
5.4	3.4	1.7	0.2	setosa

# Vocabulário

- Atributo objetivo ou alvo: classe ou categoria que possui os valores a serem preditos. Espera-se dependência dos outros atributos.

Objetivo


sepal length	sepal width	petal length	petal width	species
6.7	3.0	5.2	2.3	virginica
6.4	2.8	5.6	2.1	virginica
4.6	3.4	1.4	0.3	setosa
6.9	3.1	4.9	1.5	versicolor
4.4	2.9	1.4	0.2	setosa
4.8	3.0	1.4	0.1	setosa
5.9	3.0	5.1	1.8	virginica
5.4	3.9	1.3	0.4	setosa
4.9	3.0	1.4	0.2	setosa
5.4	3.4	1.7	0.2	setosa



# Vocabulário

- Atributos: valores que definem o atributo alvo por algum tipo de relação de dependência

Atributos



sepal length	sepal width	petal length	petal width	species
6.7	3.0	5.2	2.3	virginica
6.4	2.8	5.6	2.1	virginica
4.6	3.4	1.4	0.3	setosa
6.9	3.1	4.9	1.5	versicolor
4.4	2.9	1.4	0.2	setosa
4.8	3.0	1.4	0.1	setosa
5.9	3.0	5.1	1.8	virginica
5.4	3.9	1.3	0.4	setosa
4.9	3.0	1.4	0.2	setosa
5.4	3.4	1.7	0.2	setosa

# Vocabulário

- Exemplo: objeto, tupla ou padrão do conjunto de dados que pode ser visto como um ponto

Exemplo →

sepal length	sepal width	petal length	petal width	species
6.7	3.0	5.2	2.3	virginica
6.4	2.8	5.6	2.1	virginica
4.6	3.4	1.4	0.3	setosa
6.9	3.1	4.9	1.5	versicolor
4.4	2.9	1.4	0.2	setosa
4.8	3.0	1.4	0.1	setosa
5.9	3.0	5.1	1.8	virginica
5.4	3.9	1.3	0.4	setosa
4.9	3.0	1.4	0.2	setosa
5.4	3.4	1.7	0.2	setosa

# Vocabulário

- Rótulo: valor do atributo alvo no ao qual o classificador de

Rótulo

sepal length	sepal width	petal length	petal width	species
6.7	3.0	5.2	2.3	virginica
6.4	2.8	5.6	2.1	virginica
4.6	3.4	1.4	0.3	setosa
6.9	3.1	4.9	1.5	versicolor
4.4	2.9	1.4	0.2	setosa
4.8	3.0	1.4	0.1	setosa
5.9	3.0	5.1	1.8	virginica
5.4	3.9	1.3	0.4	setosa
4.9	3.0	1.4	0.2	setosa
5.4	3.4	1.7	0.2	setosa

# Modelo baseado em similaridade

- Um dos modelos mais simples é o modelo baseado em similaridade
- Exemplo: algoritmo dos  $k$  vizinhos mais próximos ( $k$ -NN)
- Utiliza um conjunto de pares entrada e saída como modelo
- A entrada é comparada com os pares do modelo e uma saída é gerada a partir dos exemplos mais próximos

# O que é classificação?

- Baseado em uma informação de compra anterior, que flor recomendar a um cliente?

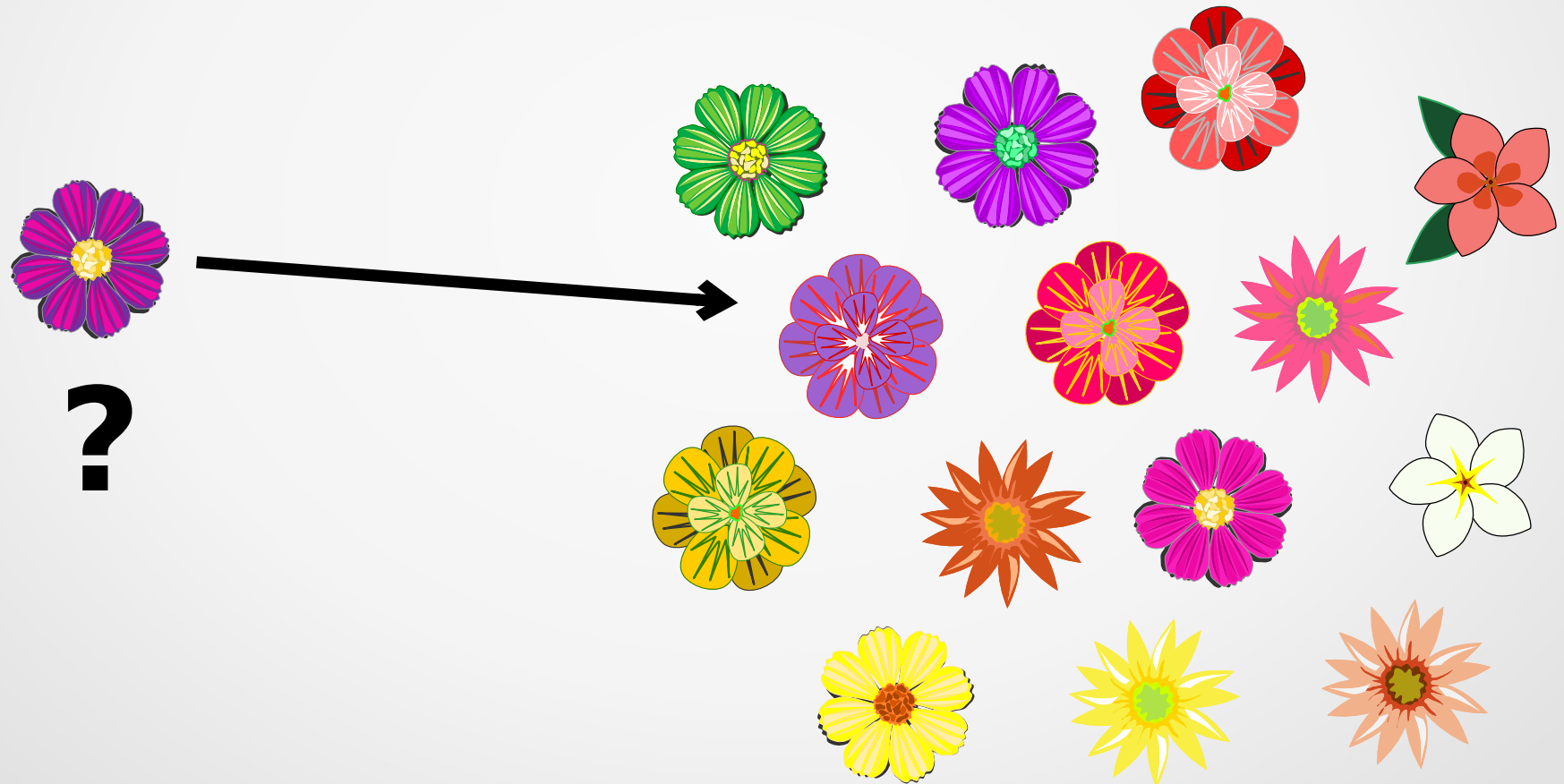


?



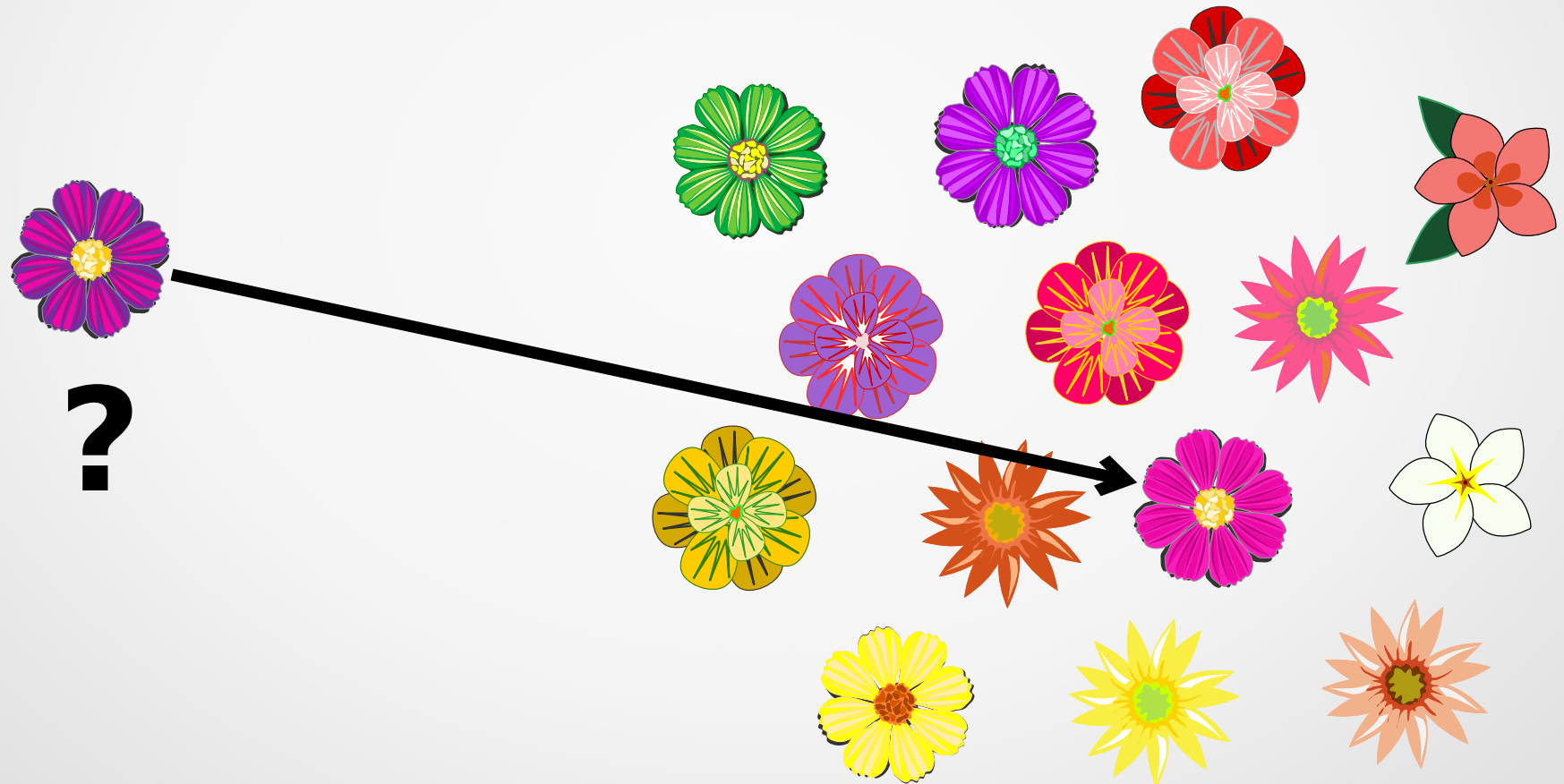
# O que é classificação?

- Baseado em uma informação de compra anterior, que flor recomendar a um cliente?



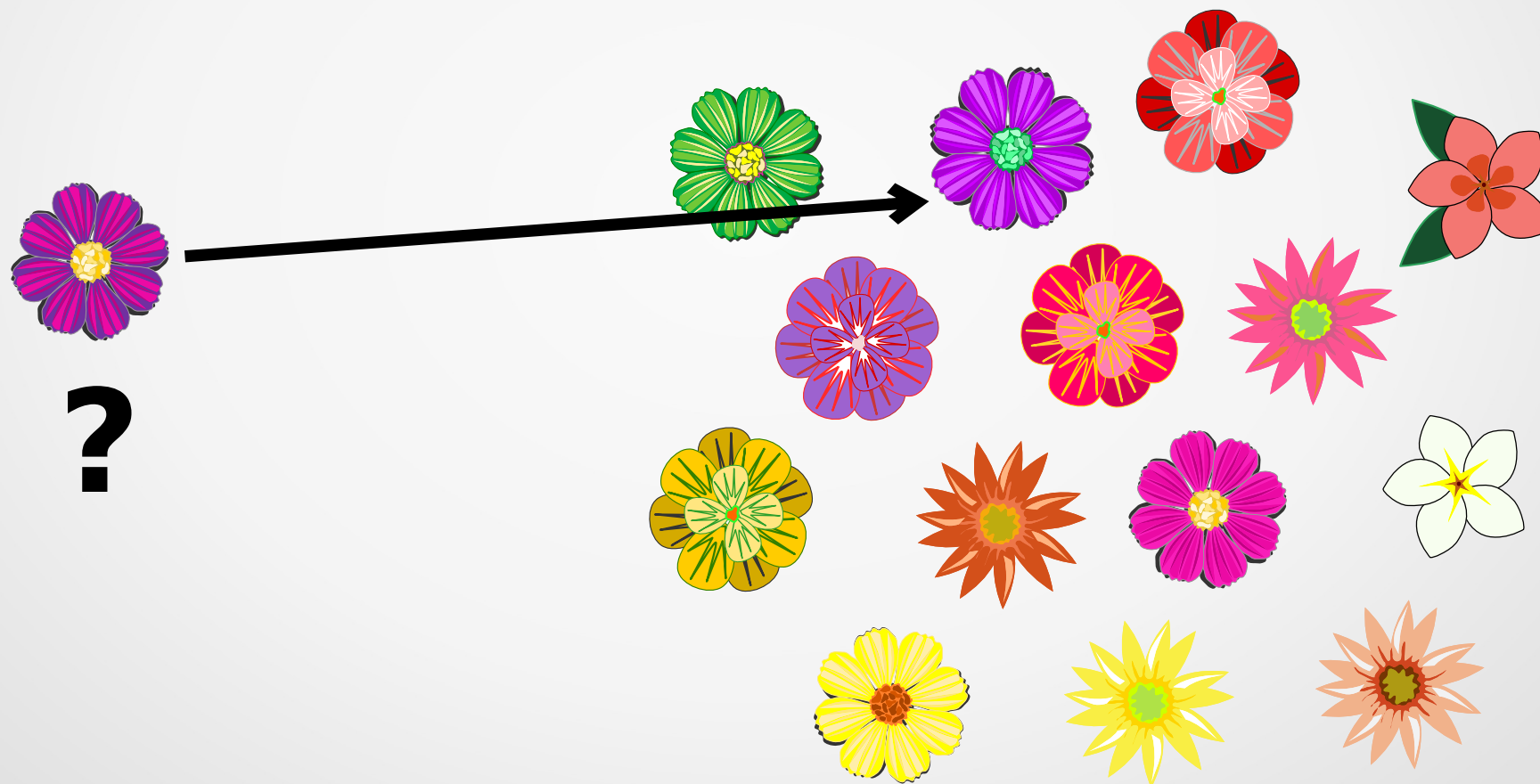
# O que é classificação?

- Baseado em uma informação de compra anterior, que flor recomendar a um cliente?



# O que é classificação?

- Baseado em uma informação de compra anterior, que flor recomendar a um cliente?





## $k$ vizinhos mais próximos

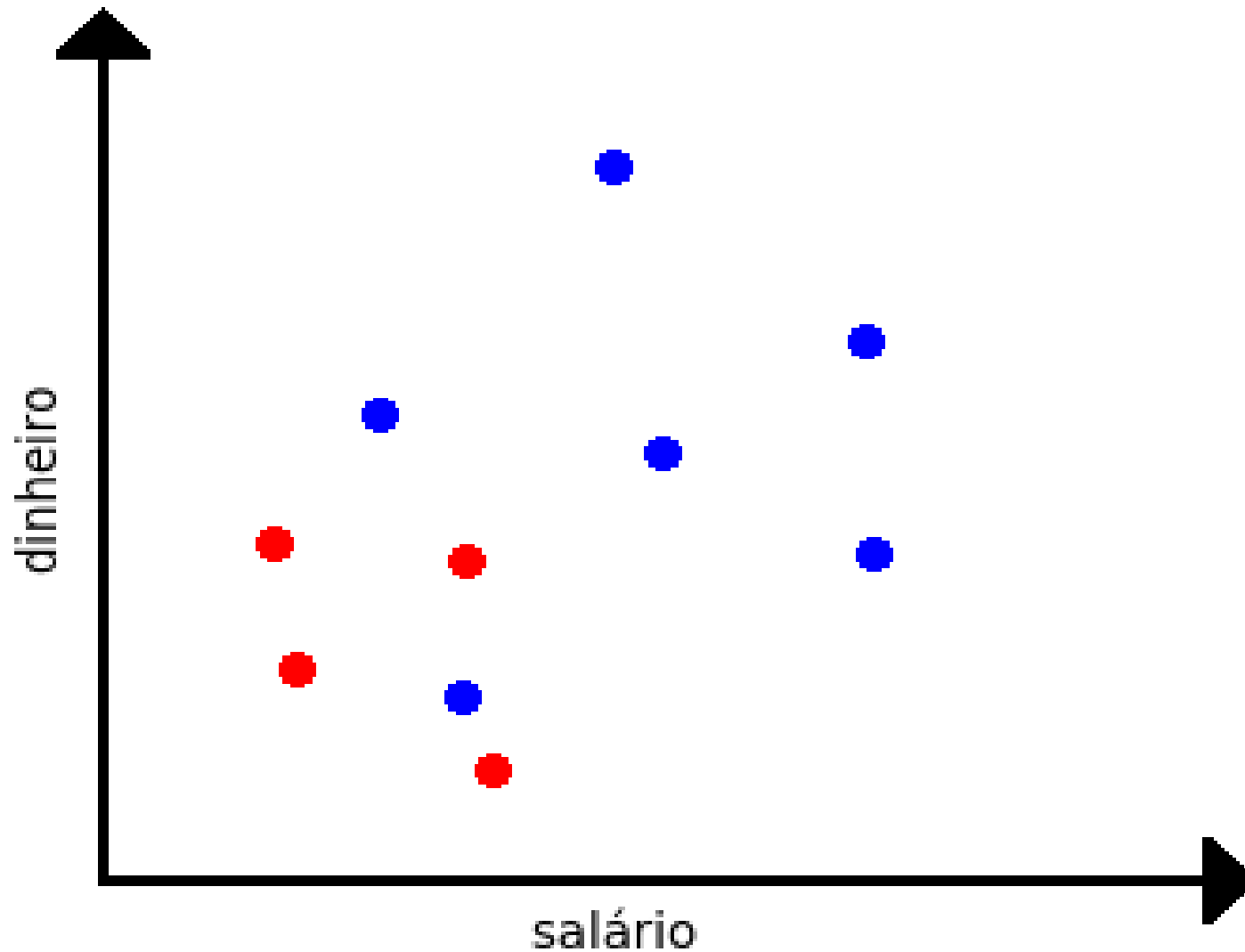
- É preciso de um conjunto de exemplos (entradas) rotulados (com as respectivas saídas).
- Exemplo aplicação:
  - Agente de análise de crédito
    - Entrada: dados bancários de clientes
    - Atributos: dinheiro e salário
    - Saída: classe do objeto
    - Crédito: sim ou não



# Dados bancários rotulados

Cliente	Dinheiro	Salário	Crédito
x1	300	360	não
x2	1270	980	sim
x3	740	280	sim
x4	240	720	não
x5	1140	1800	sim
x6	1810	1300	sim
x7	750	670	não
x8	520	1090	sim
x9	820	70	não
x10	1830	690	sim

# Dados bancários rotulados



# Hipótese

- Dado um novo cliente, o agente deverá predizer se ele deve receber crédito ou não (*classe*).
- A hipótese é que os  $k$  vizinhos mais próximos/semelhantes irão definir a classe do novo cliente.
- $k$  é um valor dado pelo usuário.

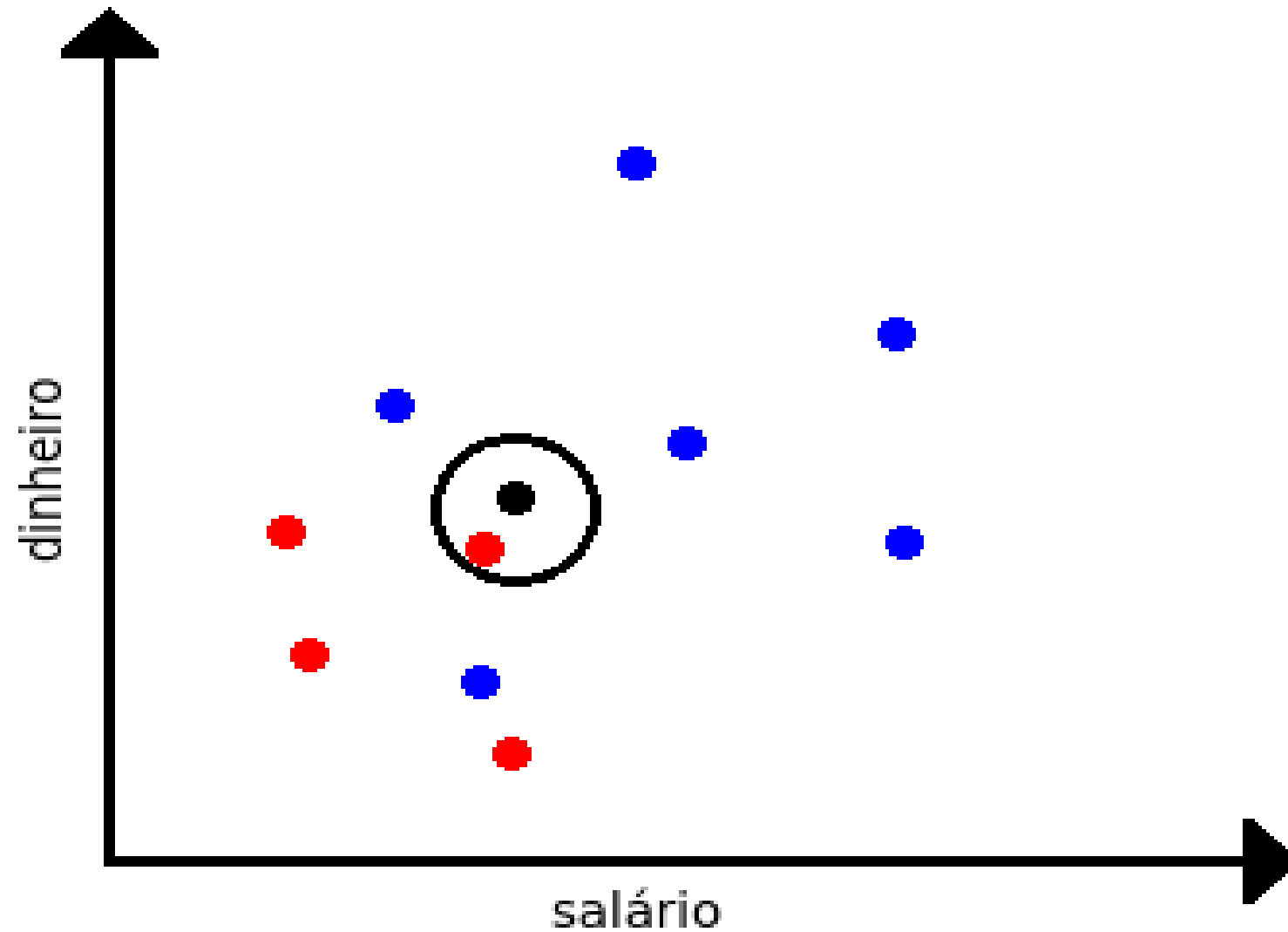
# Classificação

- Consiste em prever a classe de um dado/entrada inédito a partir do modelo induzido
- Exemplo: novo cliente  $x_{11}$  com dinheiro 860 e salário 720
- Calcula distância euclidiana  $d(x_{11}, ?)$  para todos dados do modelo e vê qual é o dado mais semelhante ( $k = 1$ ).

# Dados bancários rotulados

Cliente	Dinheiro	Salário	Crédito	$\bullet d(x_{11}, ?)$
x1	300	360	não	66,57
x2	1270	980	sim	48,55
x3	740	280	sim	45,61
x4	240	720	não	62,00
x5	1140	1800	sim	111,57
x6	1810	1300	sim	111,31
x7	750	670	não	<b>12,08</b>
x8	520	1090	sim	50,25
x9	820	70	não	65,12
x10	1830	690	sim	97,05

# Dados bancários rotulados



# Resultado

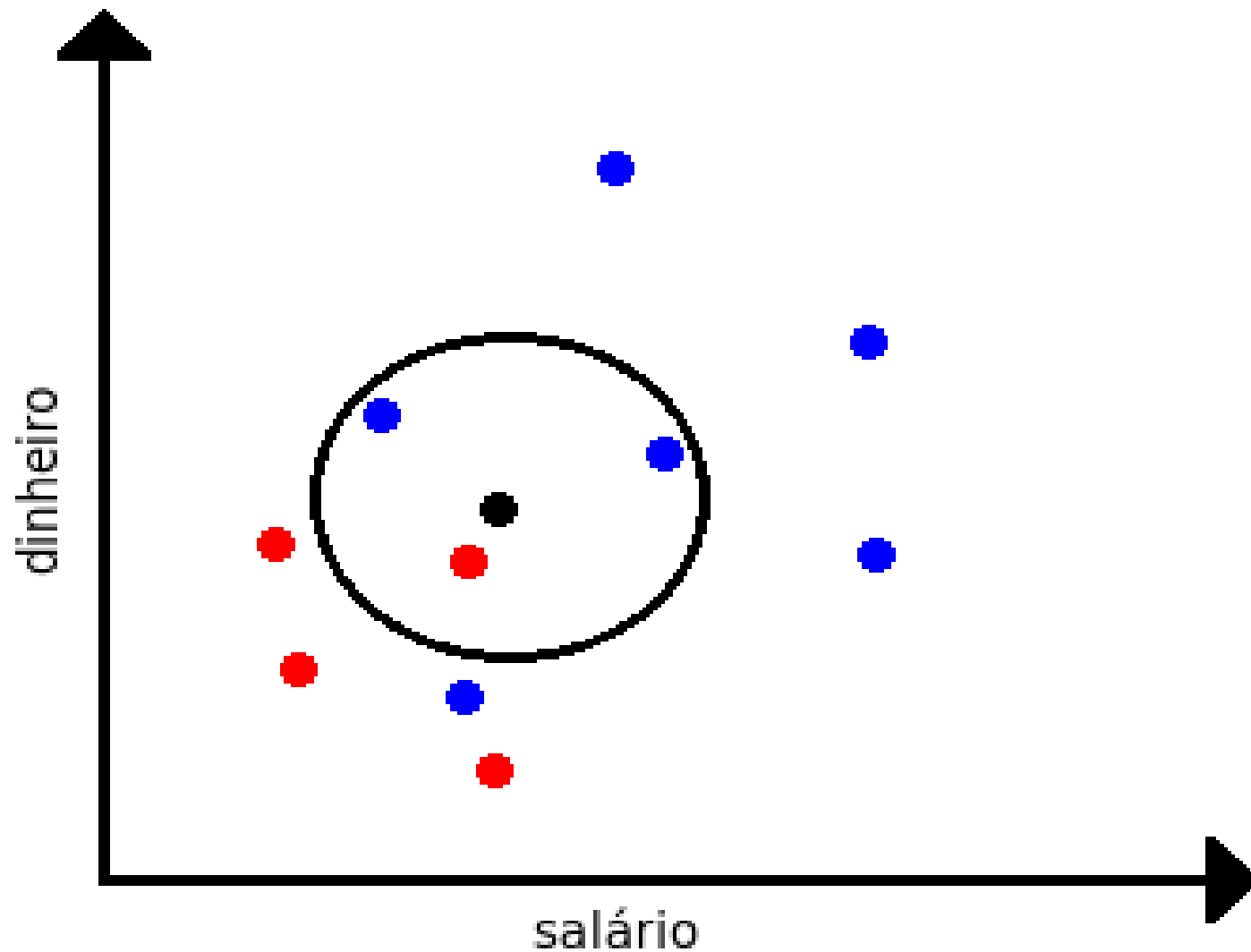
- Como o novo cliente  $x_{11}$  está mais próximo do cliente  $x_7$ , então ele será classificado como **não**, ou seja, não receberá crédito
- Mas será que essa é a melhor forma de classificar  $x_{11}$ ?
- E se fizermos  $k = 3$ ?



# Dados bancários rotulados

Cliente	Dinheiro	Salário	Crédito	$d(x_{11}, ?)$
x1	300	360	não	66,57
x2	1270	980	sim	<b>48,55</b>
x3	740	280	sim	<b>45,61</b>
x4	240	720	não	62,00
x5	1140	1800	sim	111,57
x6	1810	1300	sim	111,31
x7	750	670	não	<b>12,08</b>
x8	520	1090	sim	50,25
x9	820	70	não	65,12
x10	1830	690	sim	97,05

# Dados bancários rotulados



# Resultado

- Com  $k = 3$ , os clientes **x7**, **x2** e **x3** são os mais próximos (semelhantes) ao cliente **x11**.
- Como a maioria pertence a classe **sim**, então o **x11** também poderá receber crédito.
- Importante definir bem qual será o valor de  $k$  antes de aplicar a classificação.

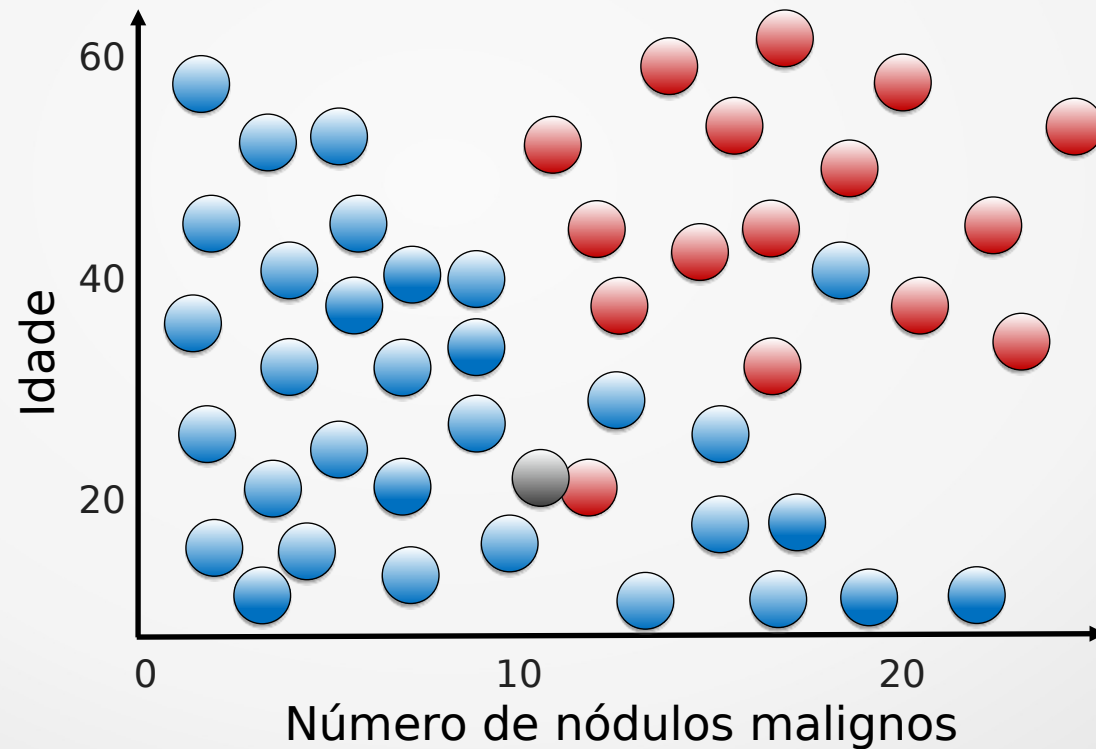
# Vantagens e desvantagens

- Vantagens:
  - $k$ -NN não precisa de treinamento
  - Incremental
- Desvantagens:
  - Armazena todos os dados de treino
  - Consulta todos os dados armazenados (proporcional ao custo computacional)
  - Problemas com dimensionalidade dos dados

# Sensibilidade a medida

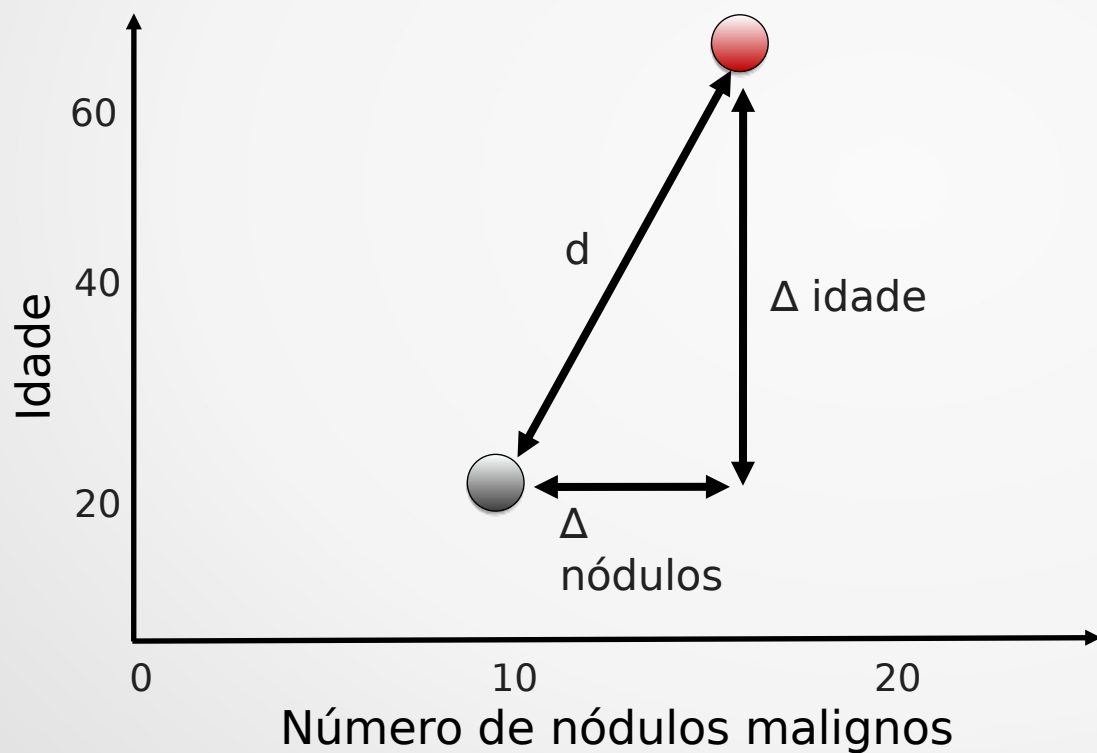
- Considere a aplicação sobre dados de pacientes com câncer

● Sobreviveu  
● Não  
Sobreviveu



# Sensibilidade a medida

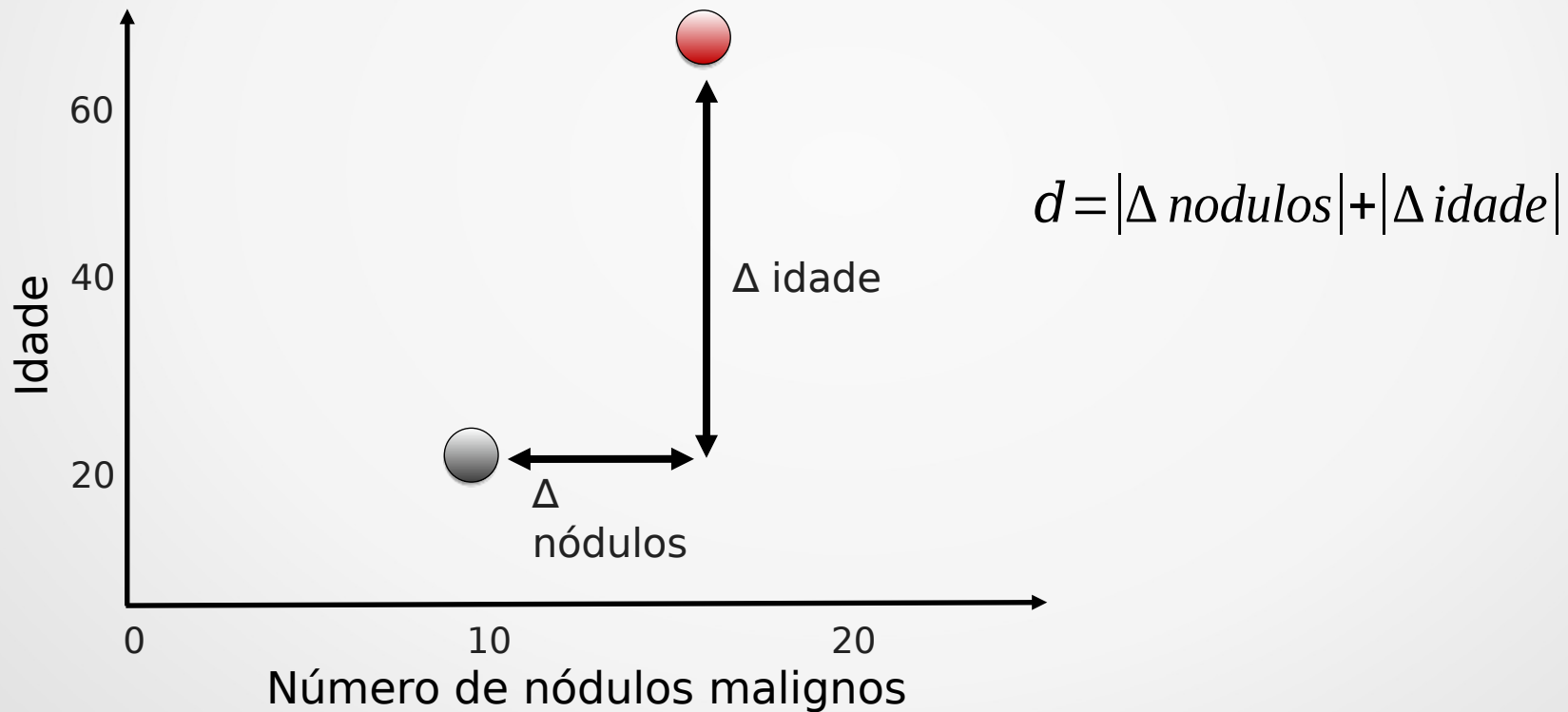
- Distância Euclideana (L2)



$$d = \sqrt{\Delta \text{ nodulos}^2 + \Delta \text{ idade}^2}$$

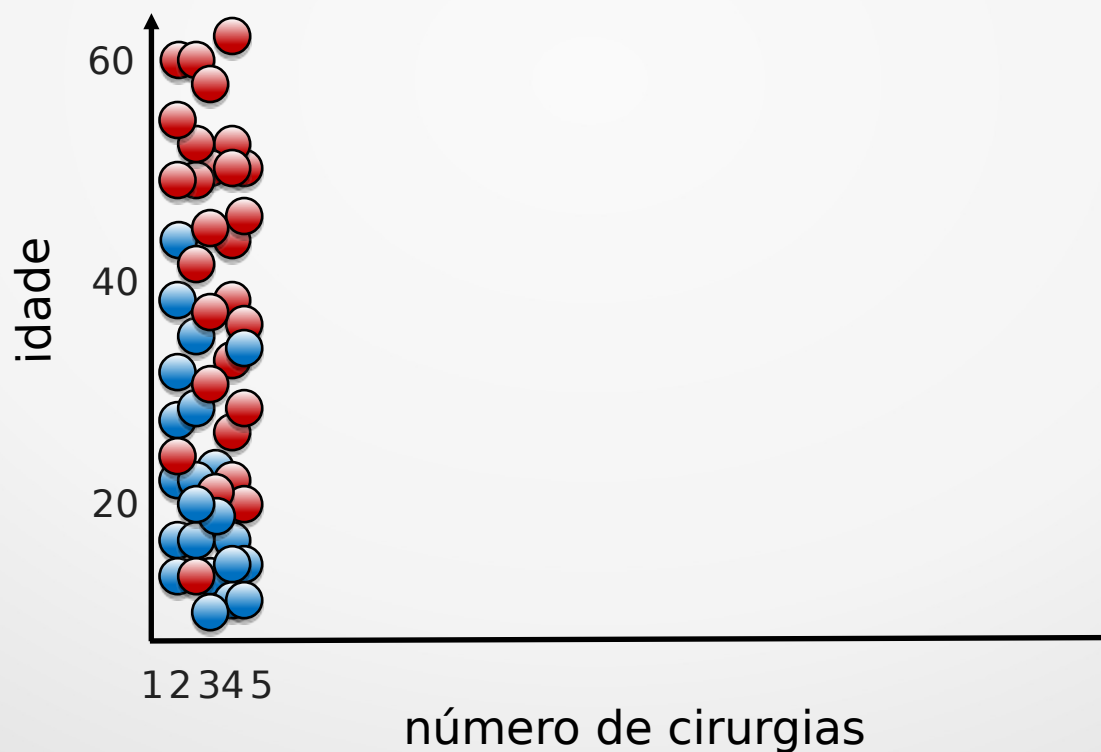
# Sensibilidade a medida

- Distância de Manhattan (L1 ou bloco cidade)



# Escala é importante

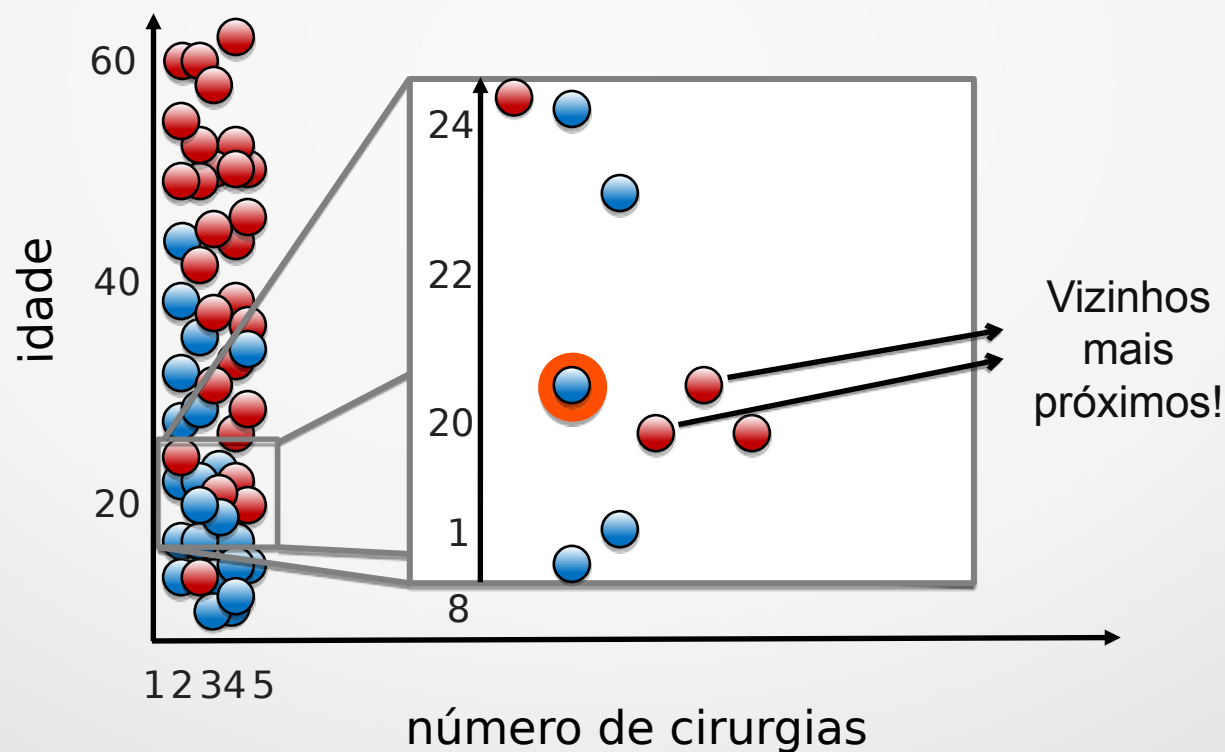
- A escala dos atributos influencia diretamente no cálculo da distância entre os objetos
  - Padronização pode ser recomendada





# Escala é importante

- A escala dos atributos influencia diretamente no cálculo da distância entre os objetos
  - Padronização pode ser recomendada



# Visão Geral Classificação

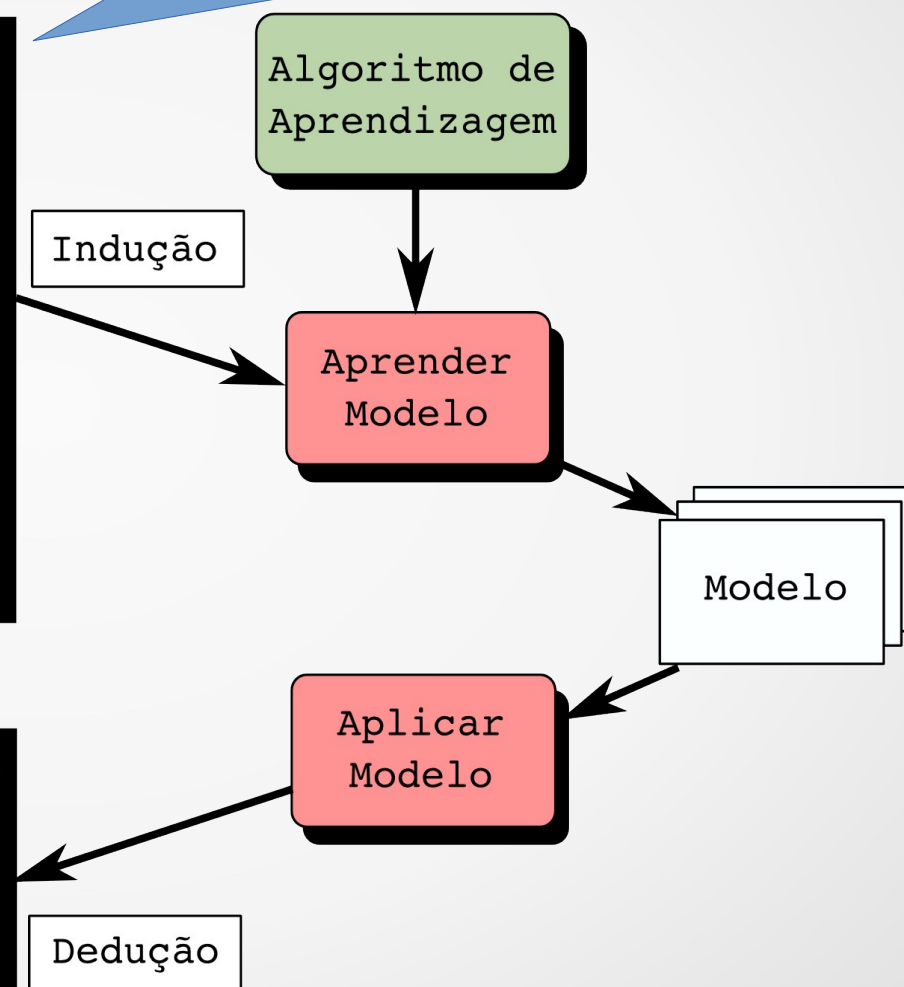
Conjunto de Treinamento

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	?

Tudo começa com conjunto de dados



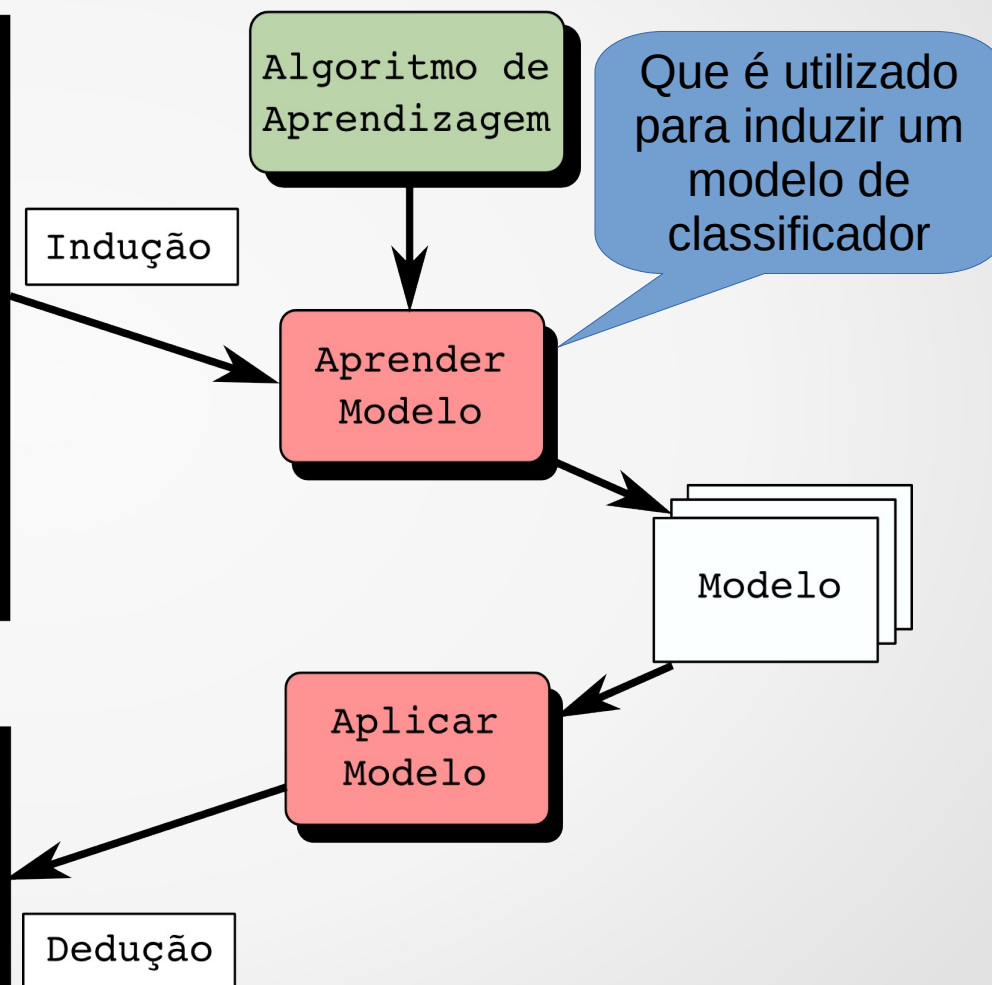
# Visão Geral Classificação

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	?



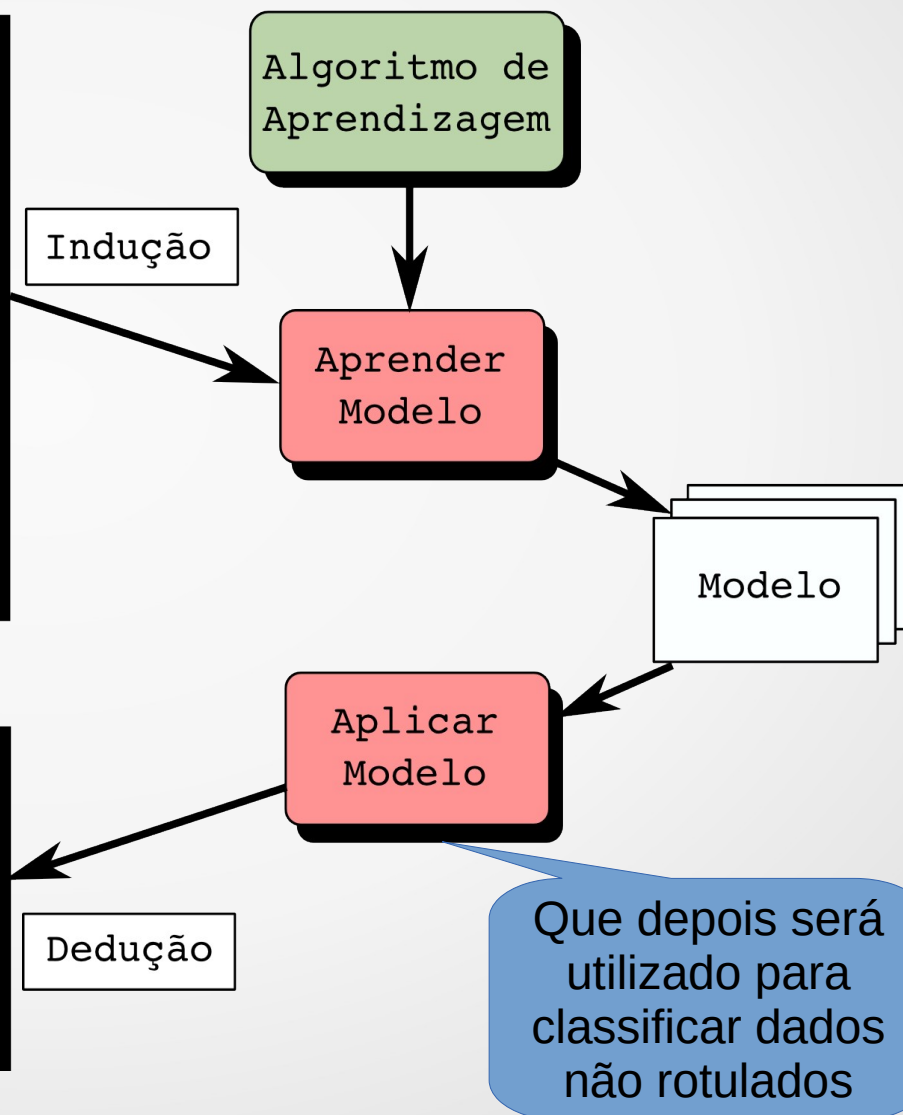
# Visão Geral Classificação

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	?



# Algumas perguntas

- Existem diversos algoritmos de classificação
  - E diversas formas de induzir um modelo
- Como avaliar o desempenho de um modelo?
- Como obter estimativas confiáveis?
- Como comparar o desempenho relativo de diferentes modelos?

# Algumas perguntas

- Existem diversos algoritmos de classificação
  - E diversas formas de induzir um modelo
- **Como avaliar o desempenho de um modelo?**
- Como obter estimativas confiáveis?
- Como comparar o desempenho relativo de diferentes modelos?

# Avaliação de desempenho

- Foco na capacidade preditiva de um modelo
  - Ignora custo computacional...
- Matriz confusão
  - Matriz que possibilita visualizar a confusão que um modelo faz ao predizer as classes de um problema

# Matriz de confusão

- Seja  $f_{(ab)}$  é a frequência em que a classe real é  $a$  e a classe prevista é  $b$
- Seja  $n$  o número total de objetos

CLASSE REAL	CLASSE PREDITA		
		1	0
	1	$f_{(11)}$	$f_{(10)}$
	0	$f_{(01)}$	$f_{(00)}$

$f_{(11)}$ : VP (verdadeiro positivo)

$f_{(10)}$ : FN (falso negativo)

$f_{(01)}$ : FP (falso positivo)

$f_{(00)}$ : VN (verdadeiro negativo)



# Métricas para avaliação de desempenho

- Medida de acurácia:
  - método bastante utilizado...

$$Acuracia = \frac{Num. \text{ previsoes corretas}}{Total \text{ de previsoes}} = \frac{f_{11} + f_{00}}{f_{11} + f_{00} + f_{01} + f_{10}}$$

# Métricas para avaliação de desempenho

- Taxa de erro:
  - complemento da acurácia

$$\begin{aligned} \textit{Taxa de erro} &= \frac{\textit{Num. de previsões erradas}}{\textit{Total de previsões}} \\ &= \frac{f_{10} + f_{01}}{f_{00} + f_{10} + f_{01} + f_{11}} \end{aligned}$$

# Limitações da acurácia

- Considere um problema de 2 classes
  - Número de exemplos Classe 0 = 9990
  - Número de exemplos de Classe 1 = 10
- Se o modelo prevê tudo para ser classe 0, a precisão é  $9990/10000 = 99,9\%$ 
  - A precisão é enganosa porque o modelo não detectar qualquer exemplo uma classe
- Taxa de erro possui mesma limitação

# Outras métricas

- Precisão

$$prel(h) = \frac{f_{11}}{f_{11} + f_{01}}$$

- Predição negativa

$$nrel(h) = \frac{f_{00}}{f_{00} + f_{10}}$$

- Suporte

$$sup(h) = \frac{f_{11}}{n}$$

- Sensitividade (*Recall*)

$$sen(h) = \frac{f_{11}}{f_{11} + f_{10}}$$

- Especificidade

$$espc(h) = \frac{f_{00}}{f_{01} + f_{00}}$$

- Cobertura

$$cob(h) = \frac{f_{11} + f_{01}}{n}$$

# Medida F

- Em alguns casos é preciso utilizar uma medida que desconsidere os verdadeiros negativos
  - geralmente 0 indica falta de uma característica
- Nestes casos, é utilizada a medida F

$$\begin{aligned} medidaF(h) &= \frac{2 * prel(h)sen(h)}{prel(h) + sen(h)} \\ &= \frac{2 * f_{11}}{2 * f_{11} + f_{01} + f_{10}} \end{aligned}$$

# Matriz de custos

- Penalização imposta ao sistema no caso deste cometer um dado tipo de erro
- O custo é a soma da frequência multiplicada por essas penalizações

CLASSE REAL	CLASSE PREDITA		
		1	0
	1	$C_{(1/1)}$	$C_{(1/0)}$
	0	$C_{(0/1)}$	$C_{(0/0)}$

$C_{(i|j)}$ : custo de classificar um objeto da classe  $j$  erroneamente como classe  $i$

# Exemplo

Matriz de custo	CLASSE PREDITA		
CLASSE REAL	$C_{(ij)}$	1	0
	1	-1	100
	0	1	0

Modelo $M_1$	CLASSE PREDITA		
CLASSE REAL		1	0
	1	150	40
	0	60	250

Precisão = 80%  
Custo = 3910

Modelo $M_2$	CLASSE PREDITA		
CLASSE REAL		1	0
	1	250	45
	0	5	200

Precisão = 90%  
Custo = 4255

# Algumas perguntas

- Existem diversos algoritmos de classificação
  - E diversas formas de induzir um modelo
- Como avaliar o desempenho de um modelo?
- **Como obter estimativas confiáveis?**
- Como comparar o desempenho relativo de diferentes modelos?

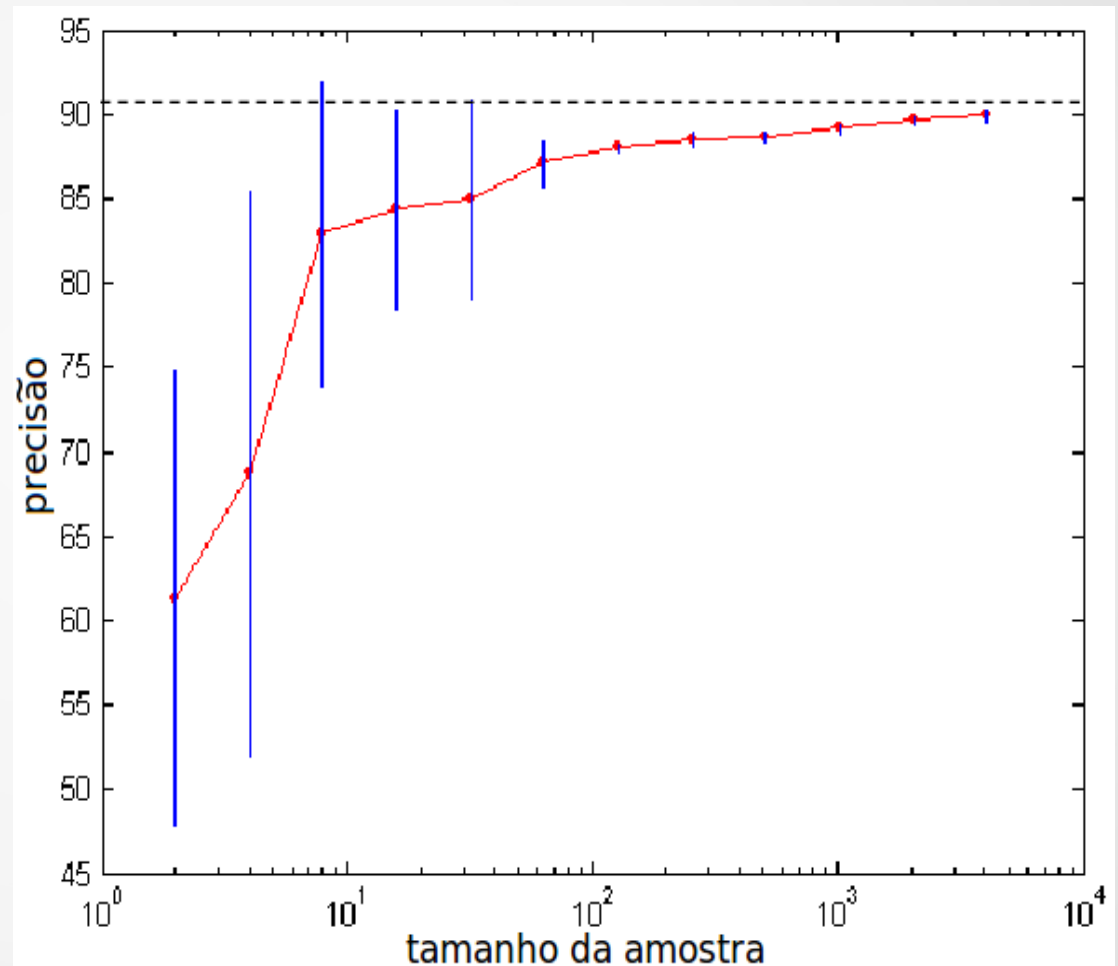


# Como estimar confiabilidade

- Como obter uma estimativa confiável do desempenho de um classificador?
- Desempenho de um modelo pode depender de outros fatores além do algoritmo de aprendizagem:
  - distribuição de classe
  - custo de má classificação
  - tamanho dos conjuntos de treinamento e teste

# Curva de aprendizado

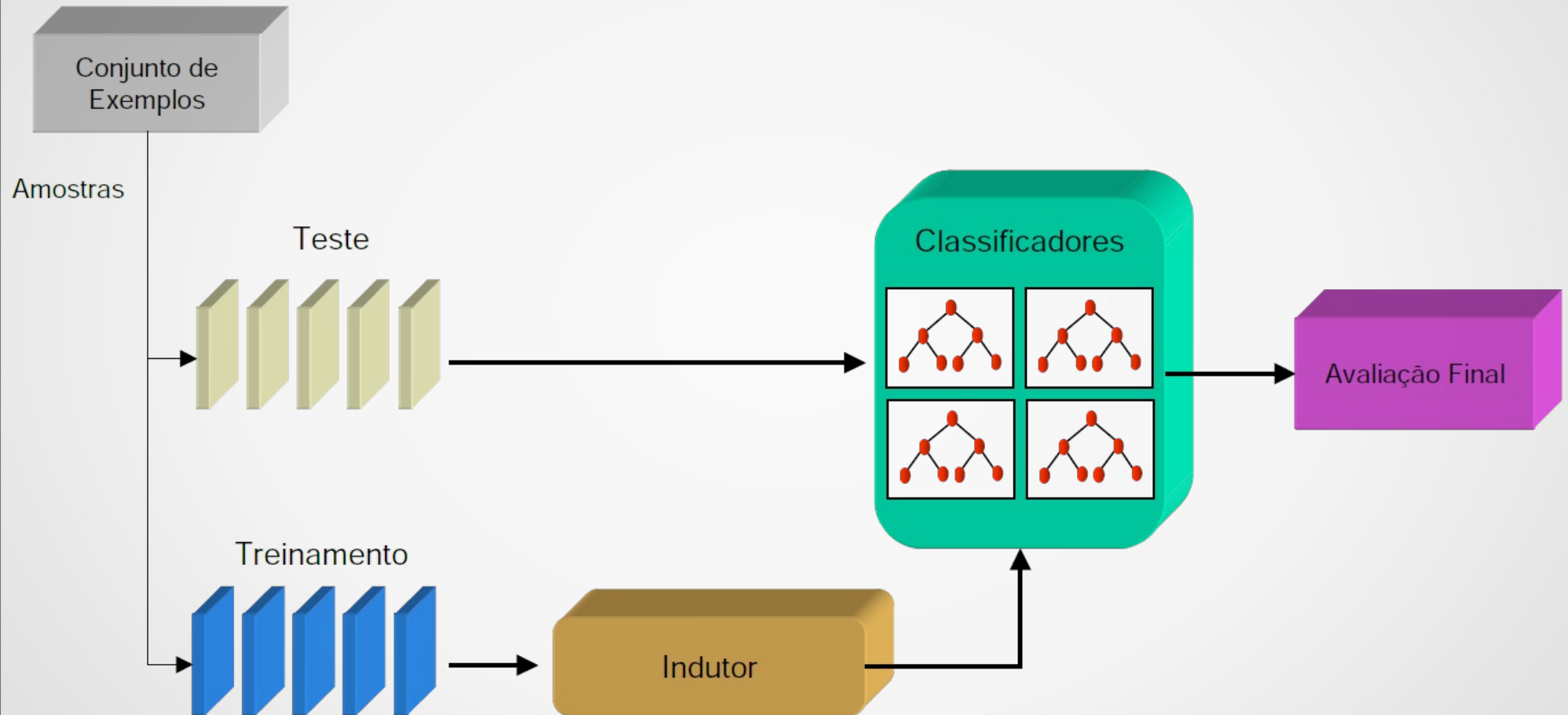
- Curva de aprendizagem mostra como a precisão varia com o tamanho da amostra
- Efeito do pequeno tamanho da amostra:
  - Viés na estimativa
  - Variância da estimativa



# Métodos de amostragem

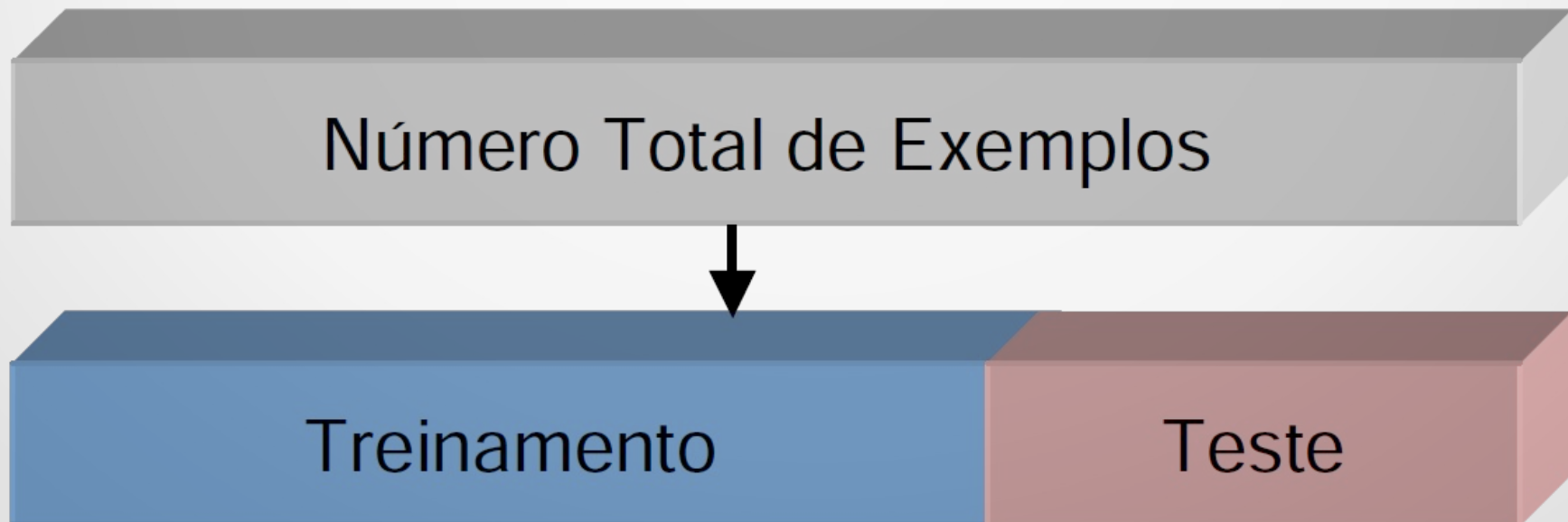
- É importante, ao estimar uma medida verdadeira (por exemplo, o erro verdadeiro), que a amostra seja **aleatória**, isto é, os exemplos não devem ser pré-selecionados
- Para problemas reais, normalmente é tomada uma amostra de tamanho  $n$  e o objetivo consiste em estimar uma medida para aquela população em particular (não para todas as populações)
- Alguns métodos para estimar medidas (**estimadores**) são descritos a seguir...

# Métodos de amostragem



# Holdout

- Este estimador divide os exemplos em uma porcentagem fixa de exemplos  $p$  para treinamento e  $(1-p)$  para teste,
- Valores típicos são  $p = 2/3$  e  $(1-p) = 1/3$ , embora não existam fundamentos teóricos sobre estes valores

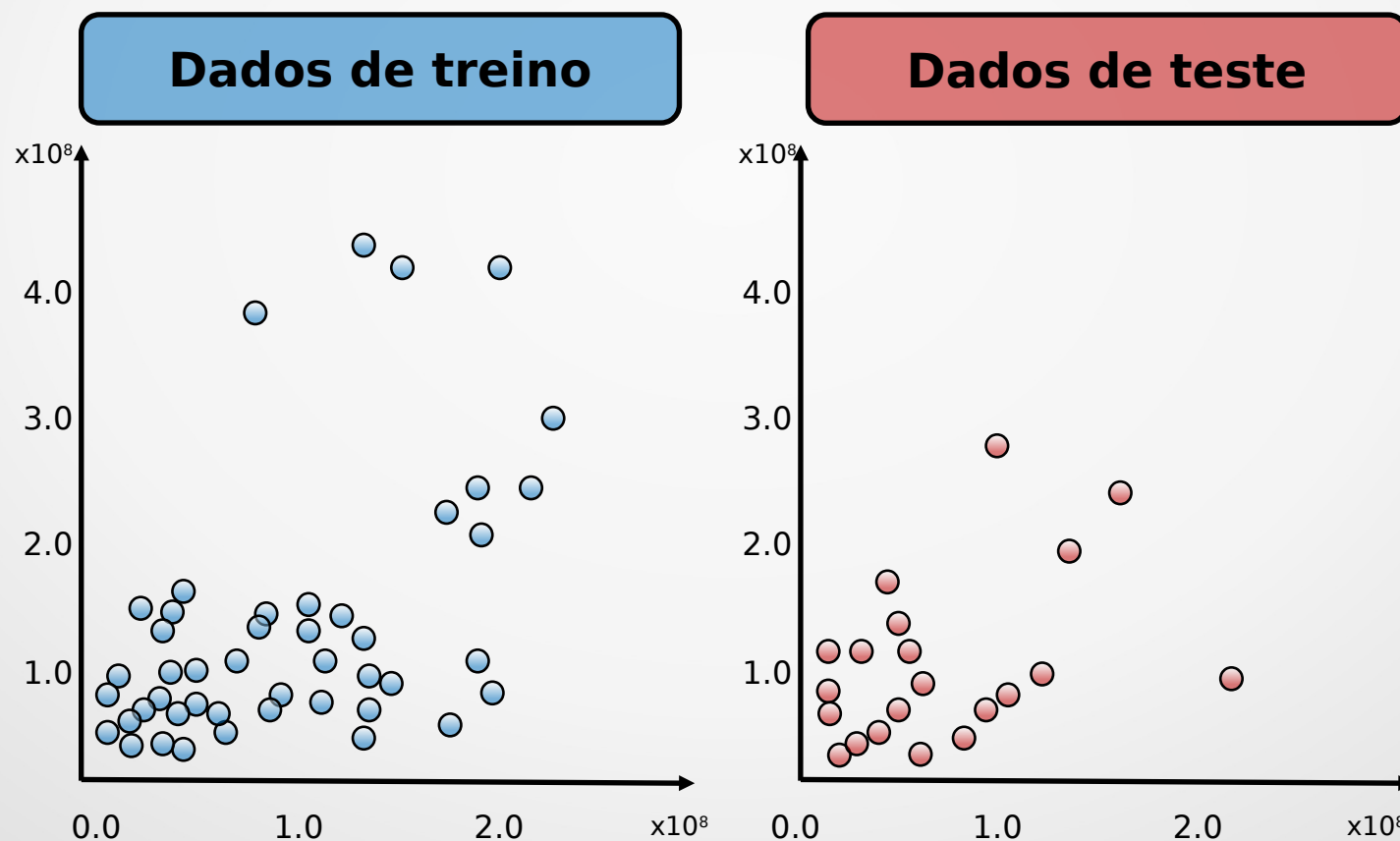


# Exemplo de *holdout*

ID	Proprietário	E. Civil	Renda	Investe	
1	Sim	Solteiro	1.500,00	Pouco	Dados de treino
2	Não	Casado	812,00	Muito	
3	Não	Solteiro	2.345,67	Não	
4	Sim	Casado	4.768,00	Muito	
5	Não	Divorciado	734,00	Não	
6	Não	Casado	3.900,00	Pouco	
7	Sim	Divorciado	2.100,00	Muito	
8	Sim	Casado	3.500,00	Pouco	Dados de teste
9	Não	Solteiro	5.000,00	Muito	
10	Sim	Solteiro	2.547,00	Pouco	
12	Sim	Casado	1.532,00	Não	

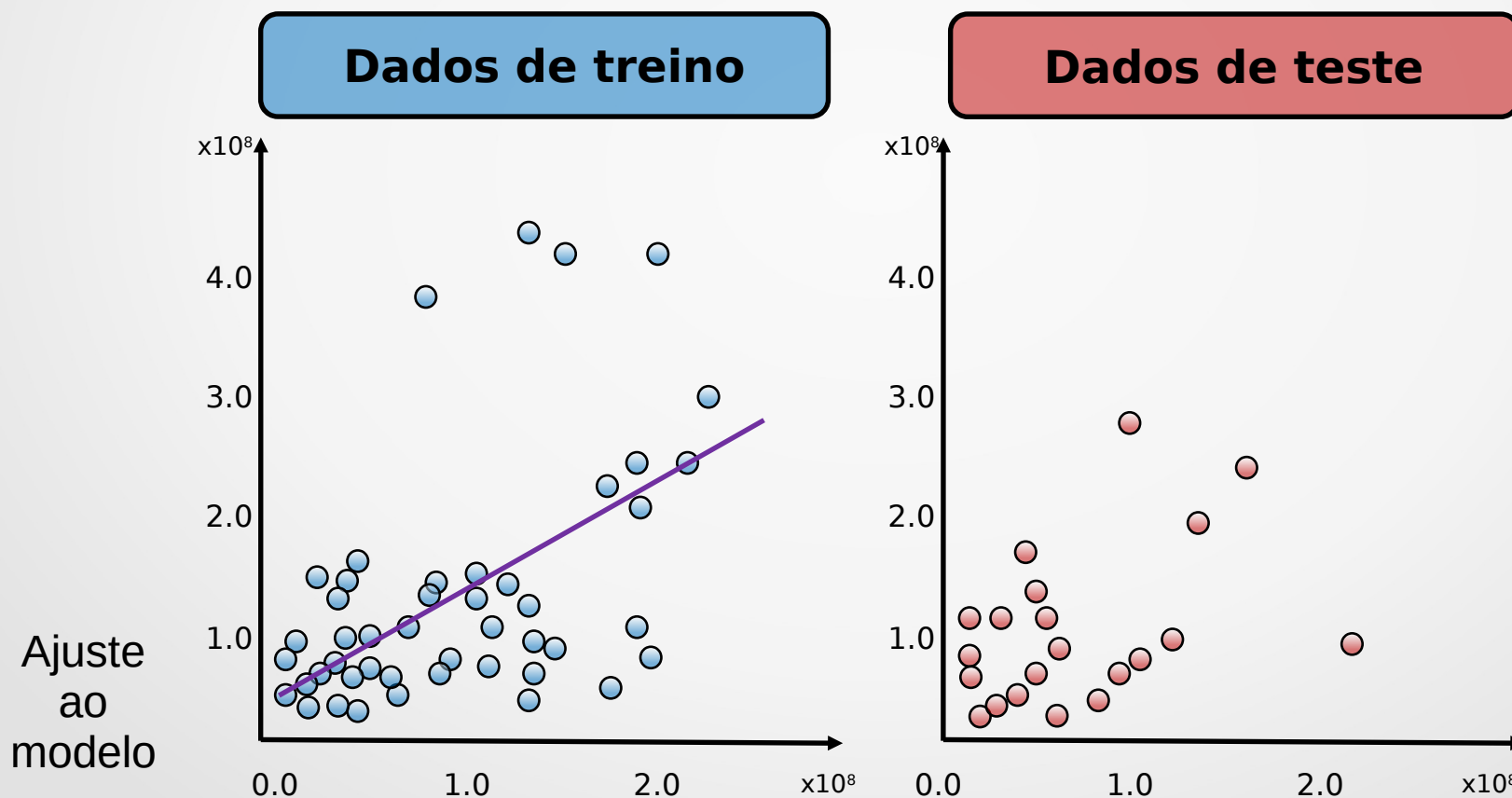
# Usando *holdout*

- Dados de treino são usados para gerar o modelo
  - Dados de teste para verificar sua qualidade



# Usando *holdout*

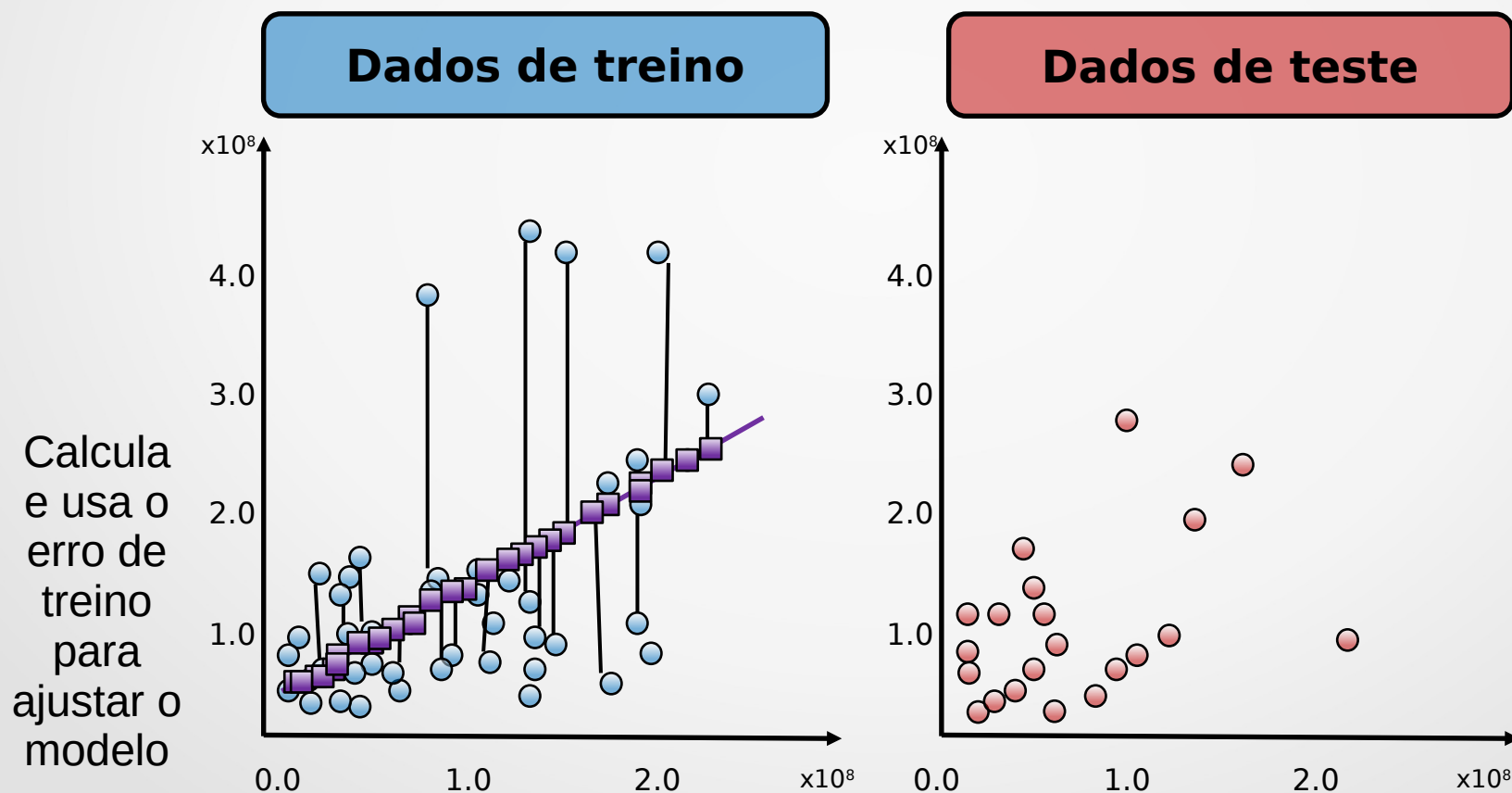
- Dados de treino são usados para gerar o modelo
  - Dados de teste para verificar sua qualidade





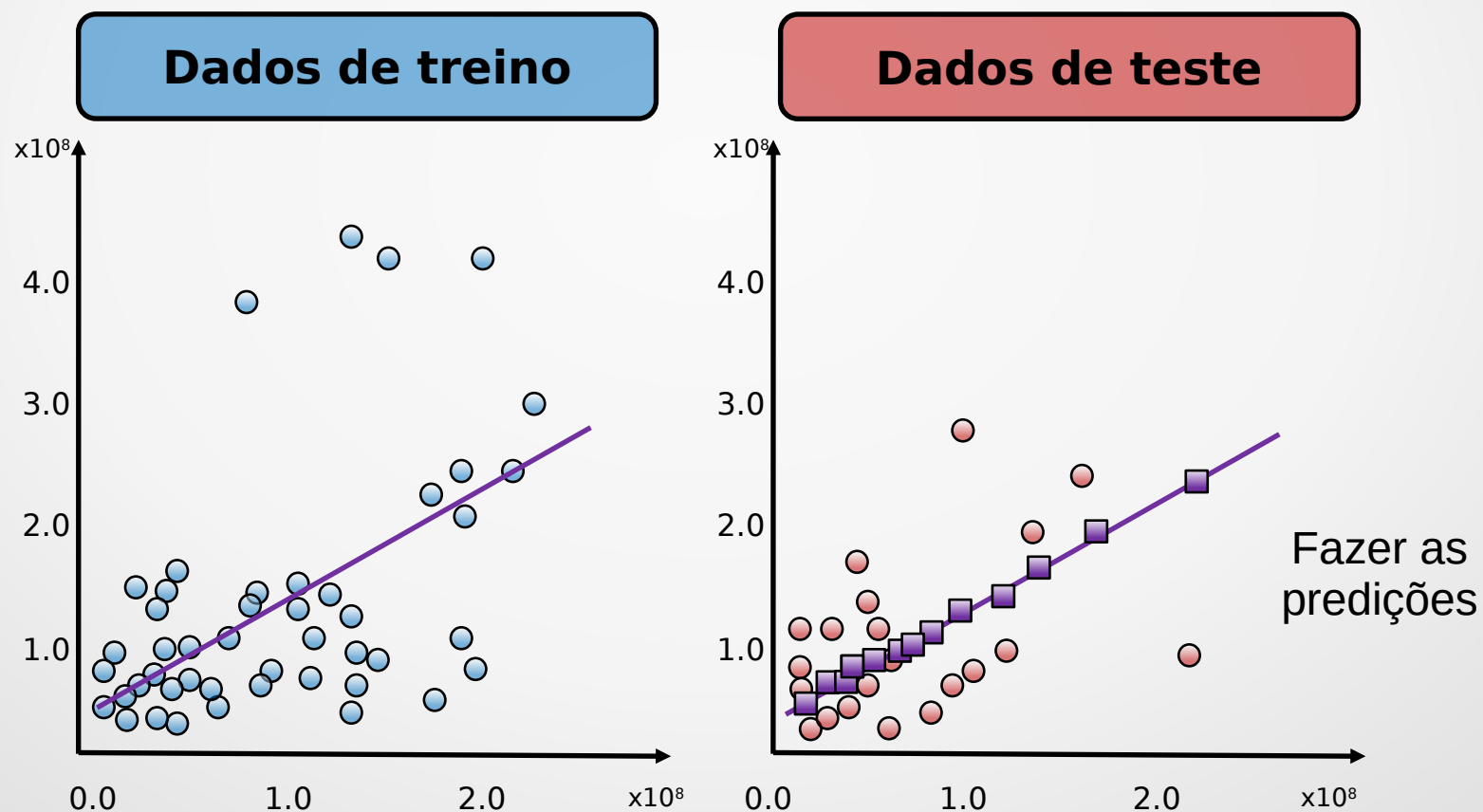
# Usando *holdout*

- Dados de treino são usados para gerar o modelo
  - Dados de teste para verificar sua qualidade



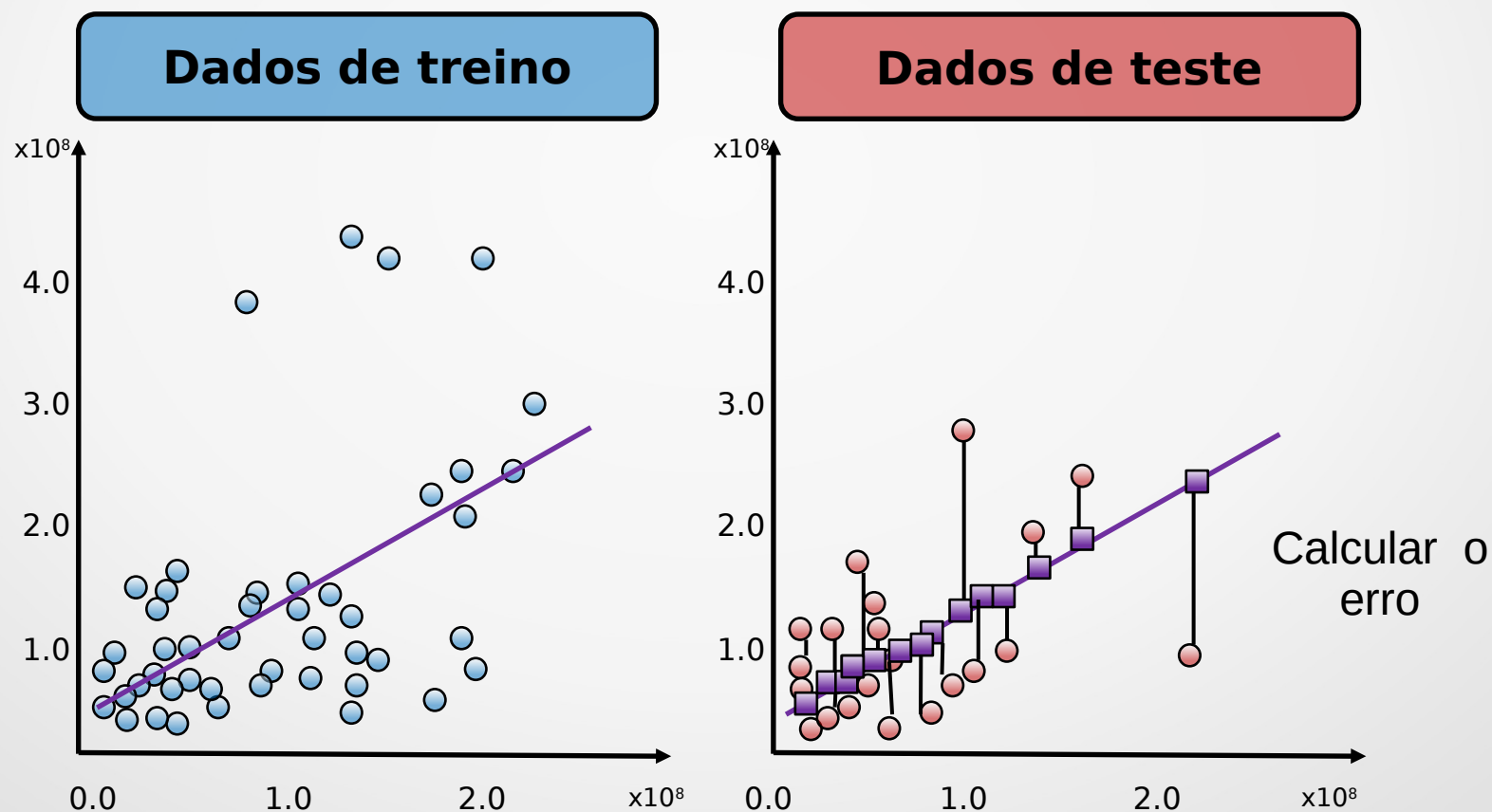
# Usando *holdout*

- Dados de treino são usados para gerar o modelo
  - Dados de teste para verificar sua qualidade



# Usando *holdout*

- Dados de treino são usados para gerar o modelo
  - Dados de teste para verificar sua qualidade



# Aplicando *knn* com *holdout* Iris

- Código

```
#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
KNN = KNeighborsClassifier(n_neighbors=3)

#Ajusta a modelo aos dados de treino
KNN = KNN.fit(traindata.iloc[:,0:4],traindata.iloc[:,4])

#Classe real
print(testdata.iloc[:,4])

#Classe predita
print(KNN.predict(testdata.iloc[:,0:4]))
```

# Aplicando *knn* com *holdout* Iris

- Saída = Classes

```
100      Iris-virginica
66       Iris-versicolor
19       Iris-setosa
41       Iris-setosa
65       Iris-versicolor
35       Iris-setosa
121      Iris-virginica
147      Iris-virginica
25       Iris-setosa
145      Iris-virginica
23       Iris-setosa
27       Iris-setosa
6        Iris-setosa
13       Iris-setosa
62       Iris-versicolor
```

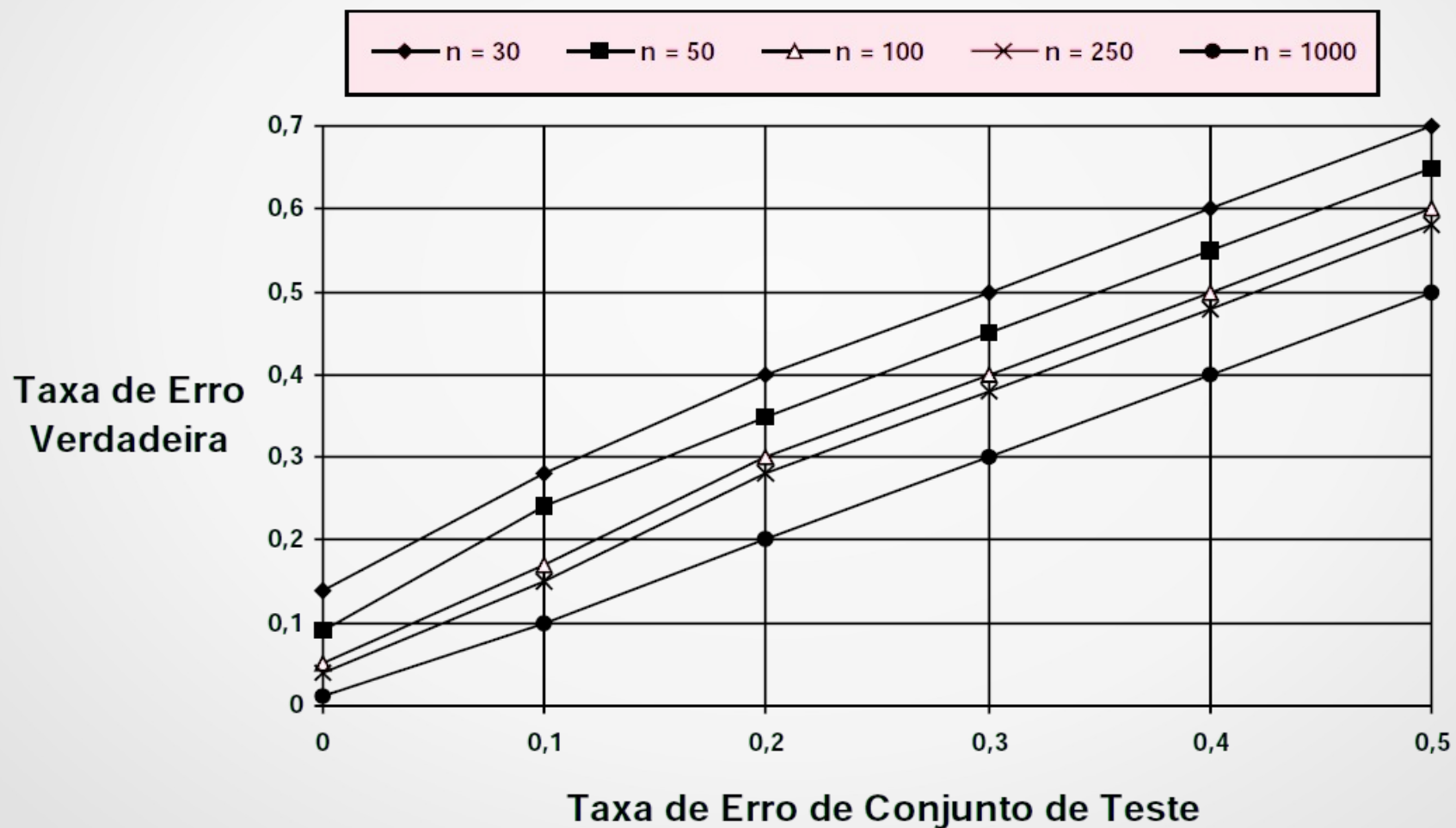
- Saída Predita

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

# Métodos de validação

- Quantos casos de teste são necessários para uma estimativa precisa?
- Quantos casos deve conter cada conjunto de treinamento e teste?
- No gráfico a seguir é mostrada a relação entre a taxa de erro do conjunto de teste e a taxa de erro verdadeira máxima para conjuntos de teste de vários tamanhos, com 95% de confiabilidade.

# Qualidade da predição



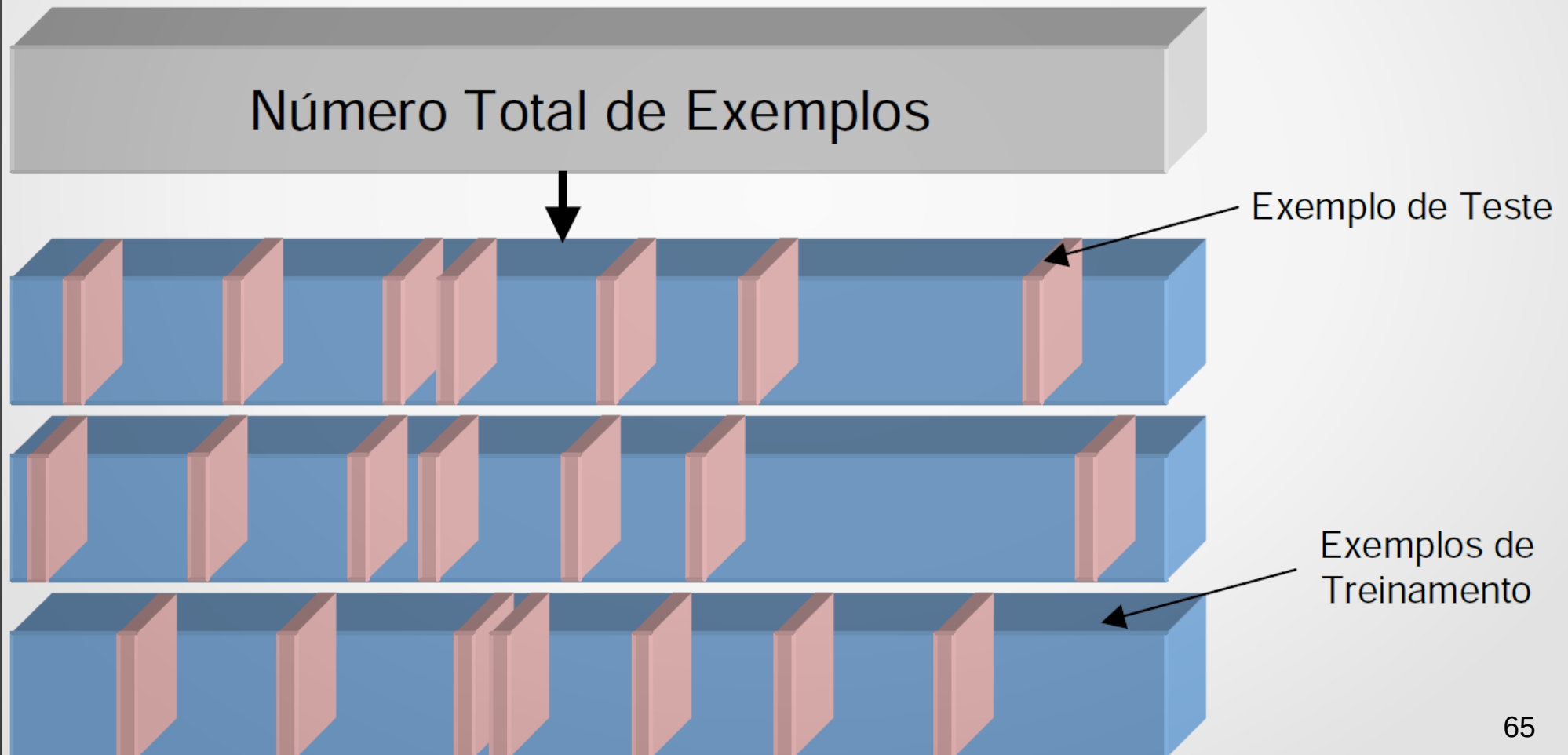
# Qualidade da predição

- Por exemplo, para um conjunto de teste de 50 exemplos, se a taxa de erro no conjunto de teste for 0%, há uma alta probabilidade (95%) que a taxa de erro verdadeira seja no máximo 10%.
- Se isso acontecer com 1000 exemplos de teste, a taxa de erro verdadeira será, com alta probabilidade, menor que 1%



# Amostragem Aleatória

- $L$  hipóteses,  $L \ll n$ , são induzidas a partir de  $L$  amostras do conjunto de dados, ou seja, *holdout*  $L$  vezes



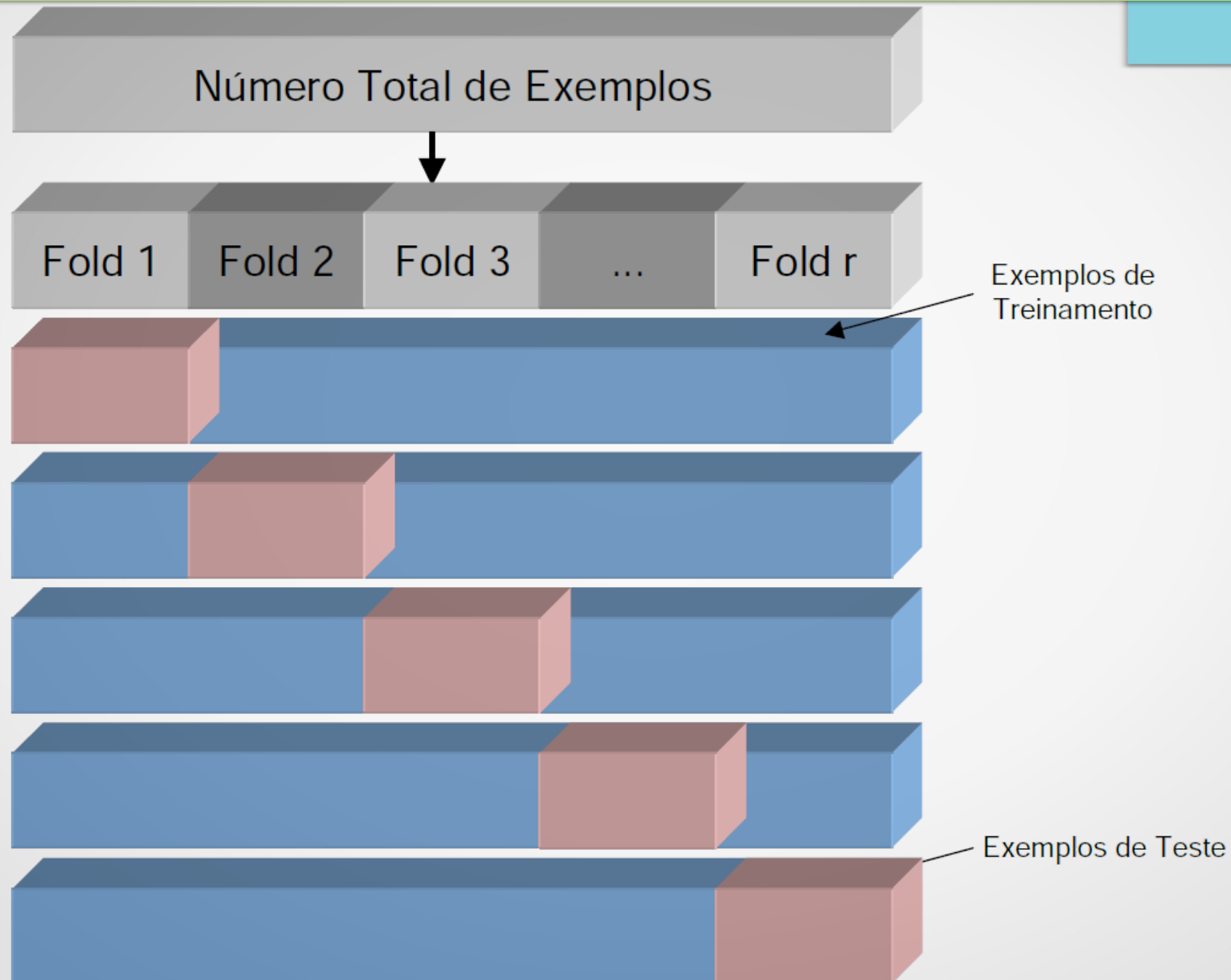
# Amostragem aleatória

- O erro final é calculado como sendo a média dos erros de todas as hipóteses induzidas e calculados em conjuntos de teste independentes e extraídos aleatoriamente
- Amostragem aleatória pode produzir melhores estimativas de erro que o estimador *holdout*

# Validação Cruzada

- Em *r-fold cross-validation* (CV) os exemplos são aleatoriamente divididos em  $r$  partições mutuamente exclusivas (*folds*) de tamanho aproximadamente igual
- Os exemplos nos  $(r-1)$  *folds* são usados para treinamento e a hipótese induzida é testada no *fold* remanescente
- Este processo é repetido  $r$  vezes, cada vez considerando um *fold* diferente para teste
- O erro é a média dos erros entre os  $r$  folds

# Validação Cruzada



# Validação cruzada com knn e Iris

- Código

```
#Importando validação cruzada
from sklearn.model_selection import
cross_val_score, KFold, StratifiedKFold

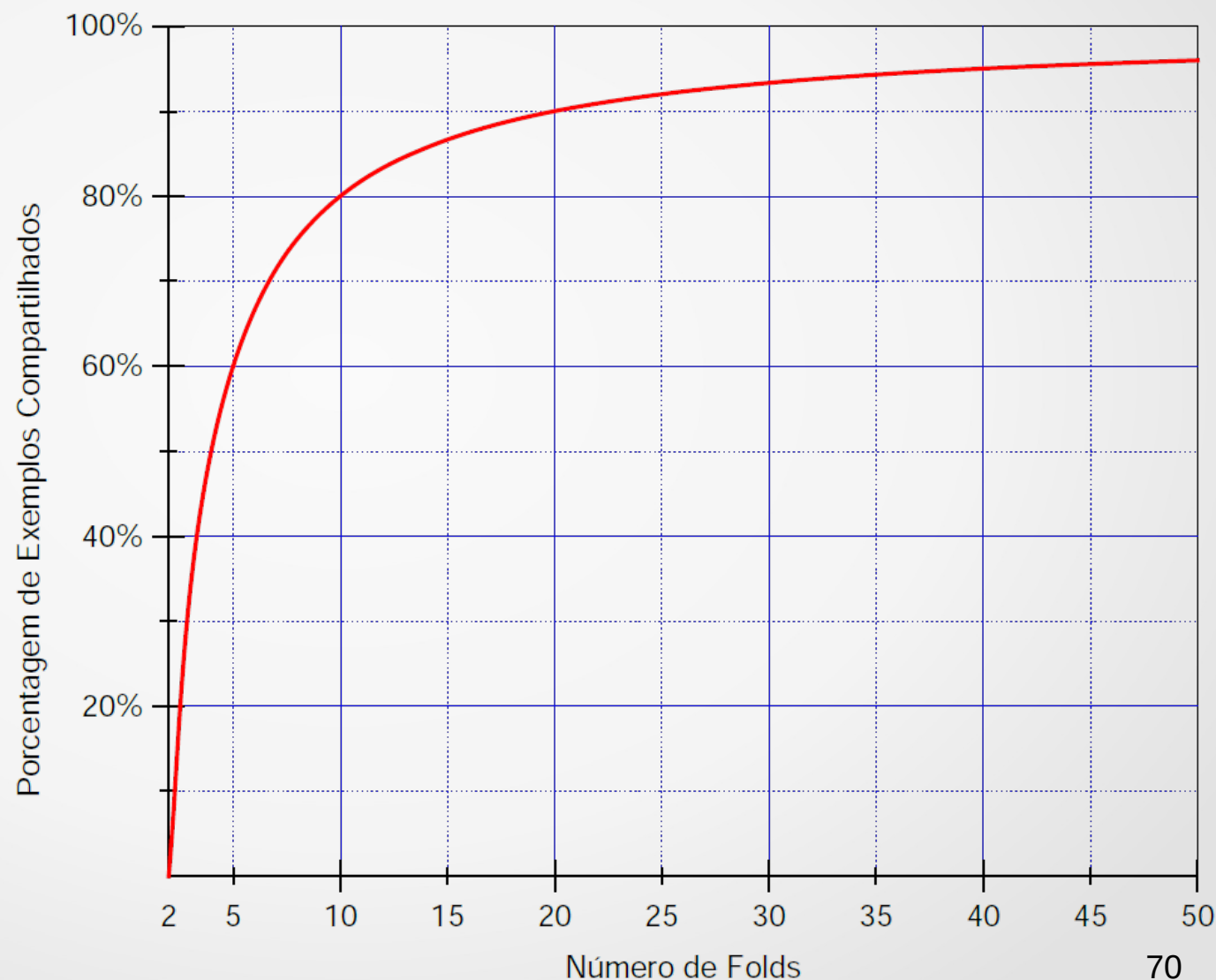
#Aplica validação cruzada
print(cross_val_score(KNN, data.iloc[:,0:4],
data.iloc[:,4], cv=10))
```

- Saída de valores de acurácia

```
[1.          0.93333333  1.          0.93333333
0.86666667  1.
0.93333333  1.          1.          1.          ]
```

# Validação Cruzada

- Qual melhor valor para  $r$ ?
- Quando maior o valor de  $r$ , maior o conjunto de treinamento
- Mais parecida é cada amostra



# Validação Cruzada Estratificada

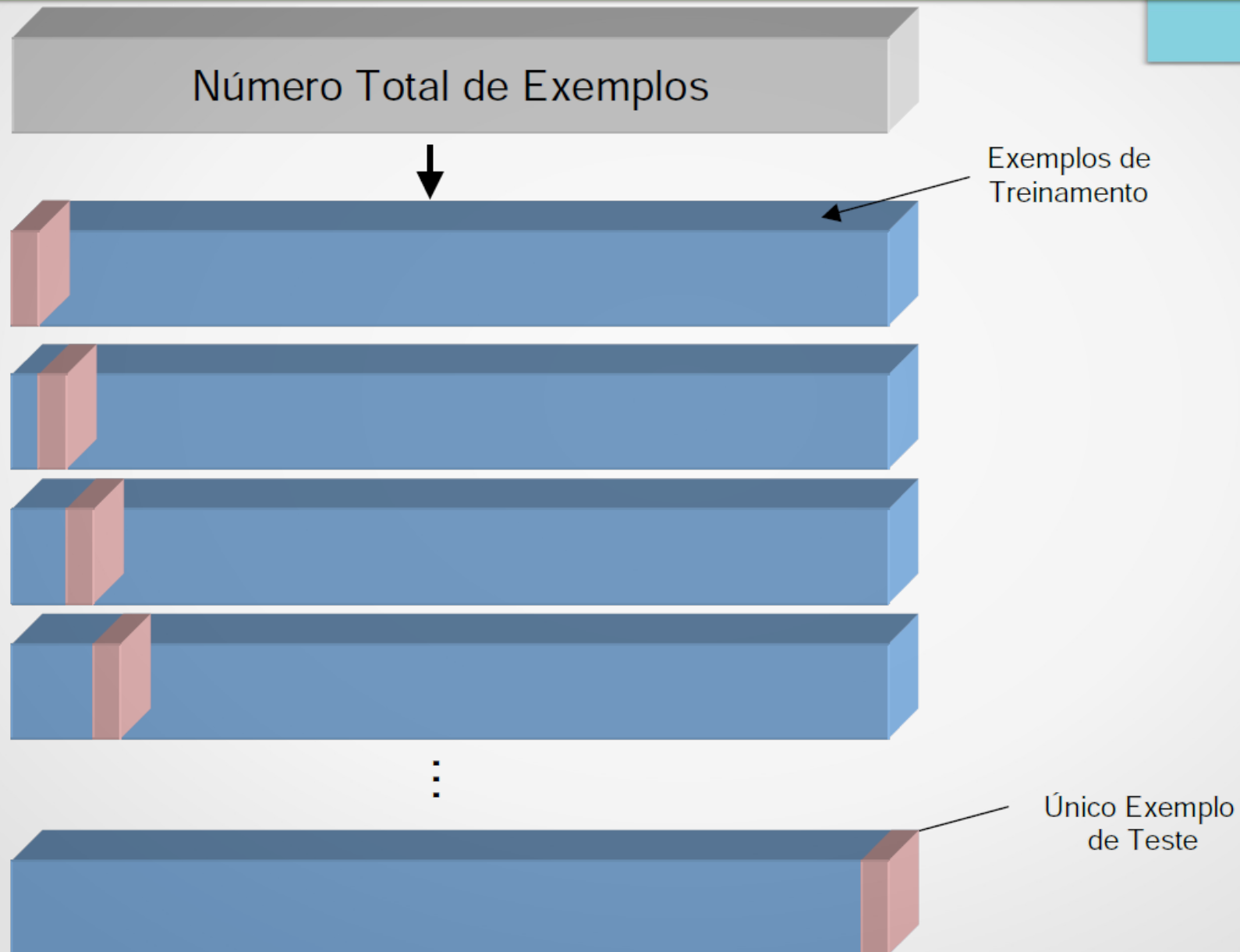
- Similar à validação cruzada, mas a distribuição de classe (proporção de exemplos em cada uma das classes) é mantida ao gerar os *folds* mutuamente exclusivos
- Isto significa, por exemplo, que se o conjunto original de exemplos possui duas classes com distribuição de 20% e 80%, então cada *fold* também terá esta mesma proporção de classes

## *Leave-one-out*

- Como a validação cruzada, em que cada partição possui um objeto, ou seja,  $r=n$
- É computacionalmente dispendioso e, por isso, usado em **amostras pequenas**
- Embora o *leave-one-out* é um estimador praticamente não tendencioso (ou seja, o estimador, após várias aplicações, tende para a taxa de erro verdadeira), sua variância para pequenas amostras é alta.



# Leave-one-out

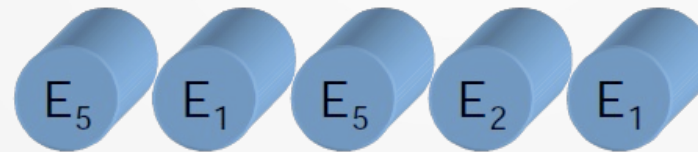
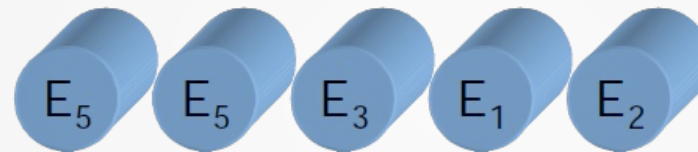
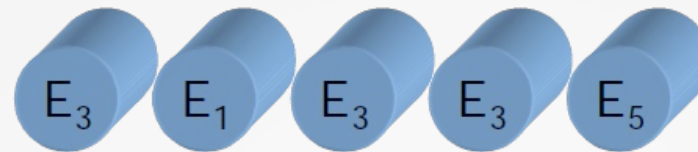
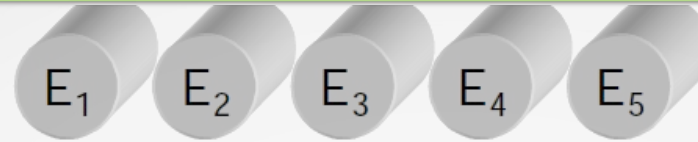


# Bootstrapping

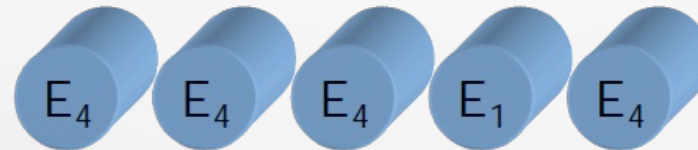
- Amostragens são feitas com reposição
  - Excelente para conjuntos de dados pequenos, especialmente se for feito estratificado
- Mais conhecida é a versão e0, que consiste em amostrar com reposição objetos para o conjunto de treino
  - Os que não foram selecionados são amostrados para o conjunto de teste

# Bootstrapping e0

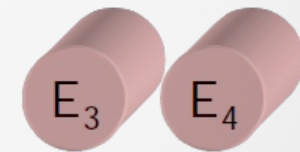
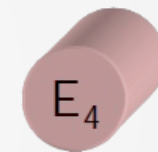
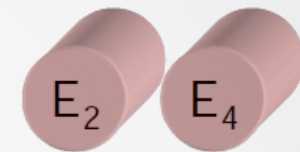
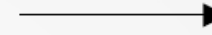
Conjunto Completo  
de Exemplos



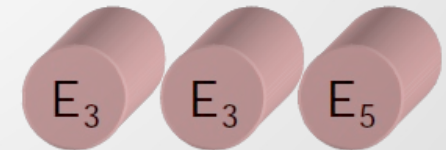
⋮



Conjuntos de Treinamento



⋮



Conjuntos de Teste

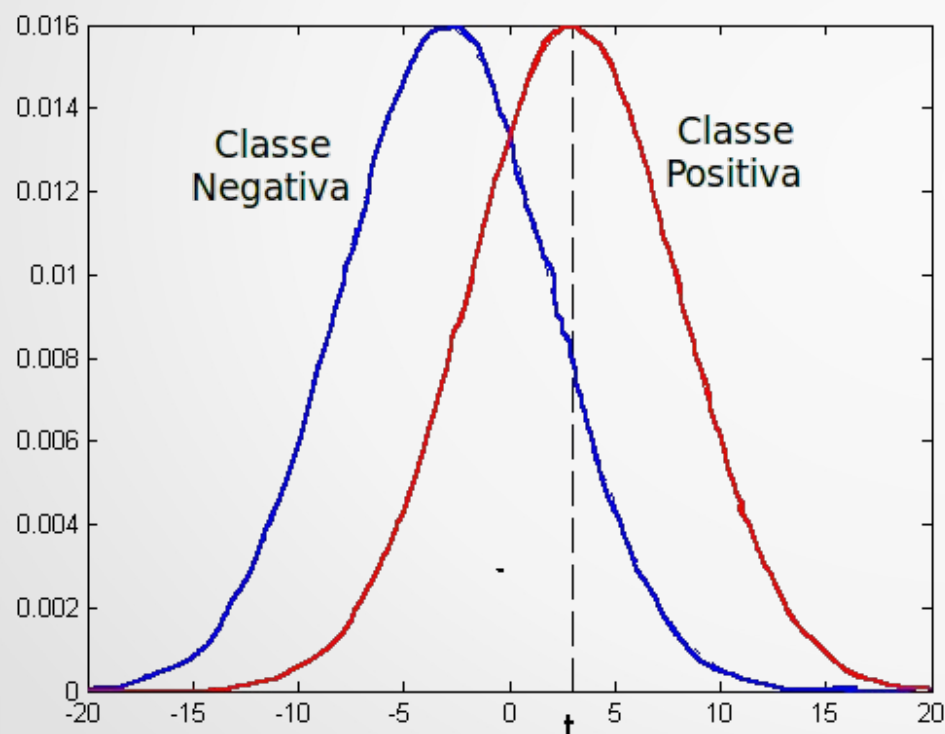
# Algumas perguntas

- Existem diversos algoritmos de classificação
  - E diversas formas de induzir um modelo
- Como avaliar o desempenho de um modelo?
- Como obter estimativas confiáveis?
- **Como comparar o desempenho relativo de diferentes modelos?**

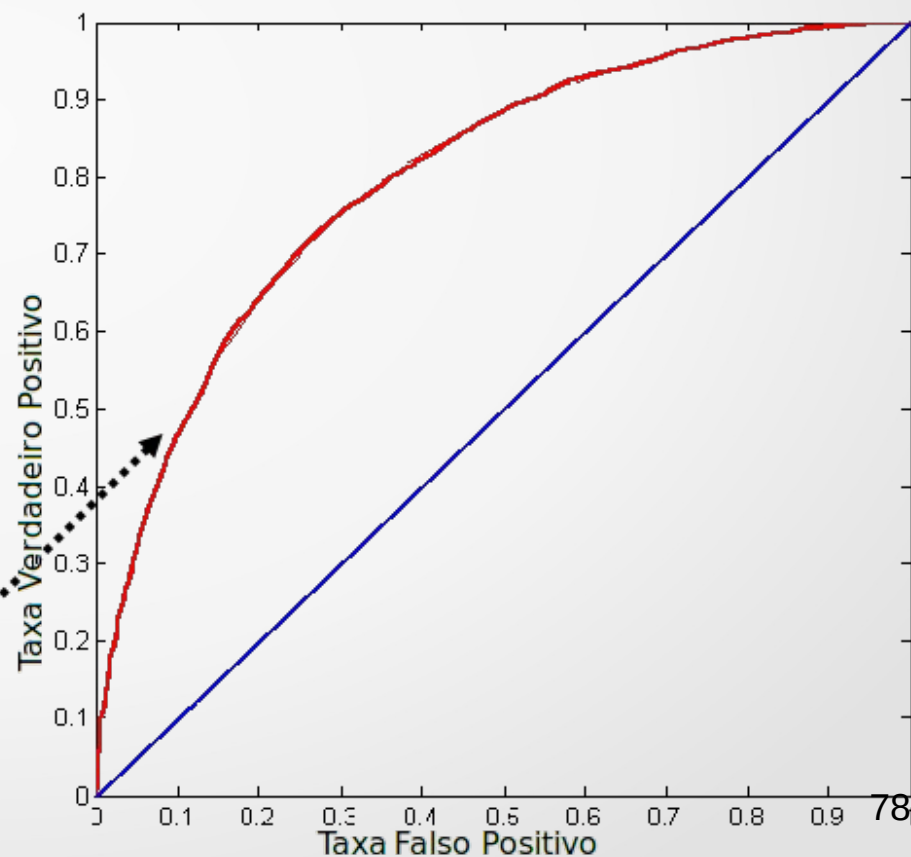
# Receiver Operating Characteristic (ROC)

- Desenvolvido em 1950 para a teoria de detecção de sinais para analisar sinais ruidosos
- Curva ROC compara VP (no eixo y) contra FP (no eixo x)
- Desempenho de cada modelo gerador é representado como um ponto na curva ROC
- Diferentes modelos são gerados mudando o limiar do algoritmo, a distribuição de amostra ou matriz de custos

- Conjunto unidimensional com 2 classes
- Se  $x > t$ , o objeto pertence a classe +, senão pertence a classe -

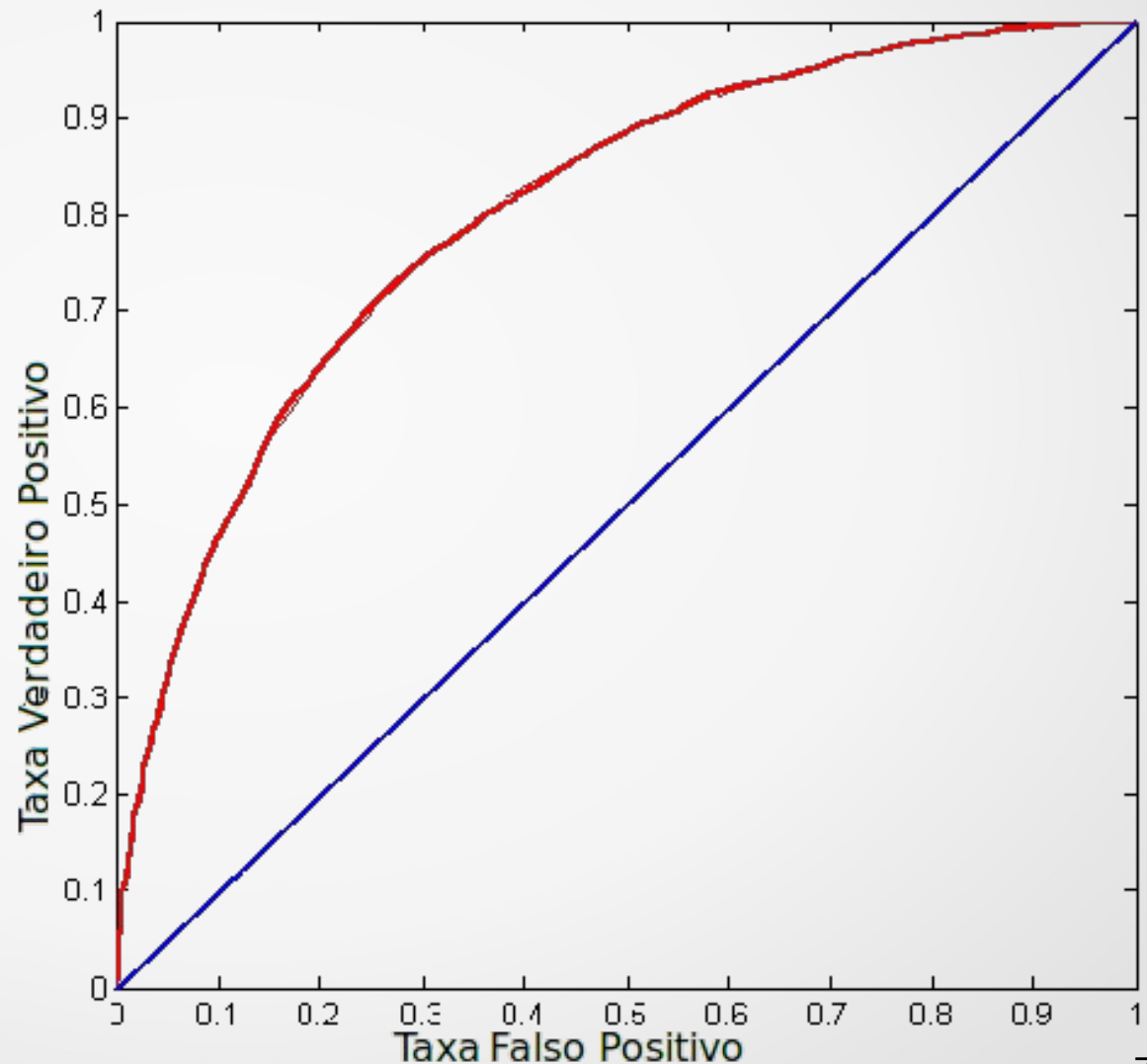


No limiar  $t$ :  
 VP = 50, FP = 12, FN = 50, VN = 88



# Curva ROC

- (VP,FP):
  - (0,0): declara tudo como classe -
  - (1,1): declara tudo como classe +
  - (1,0): ideal
- Linha diagonal:
  - Sorteio aleatório
- Abaixo da linha:
  - a predição é contrária a classe verdadeira



# Construindo a curva ROC

- Use classificador que produz probabilidade a posteriori para cada modelo de teste  $P(+|A)$
- Classificar os casos de acordo com  $P(+|A)$  ordem decrescente
- Aplicar limiar para cada valor único de  $P(+|A)$
- Contar o número de VP, FP, VN, FN em cada limiar
  - taxa VP,  $VPR = VP / (VP + FN)$
  - taxa FP,  $FPR = FP / (FP + VN)$

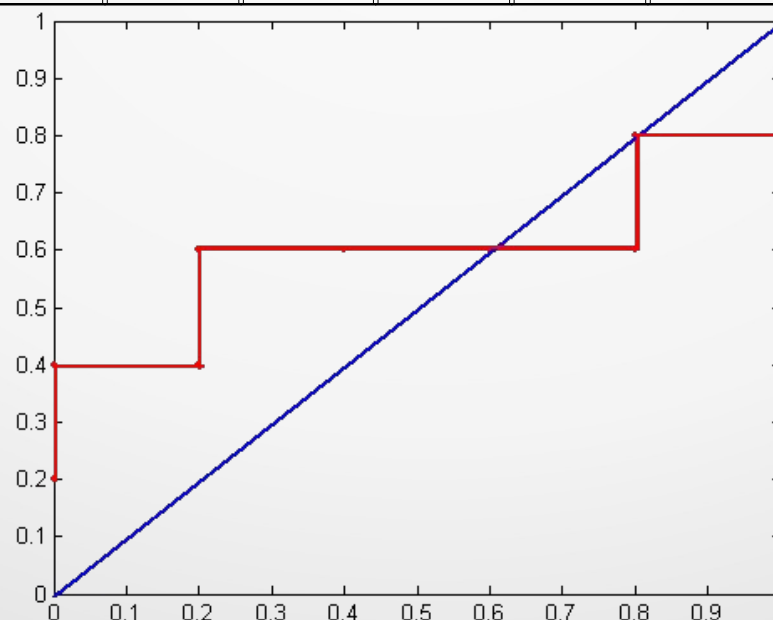
Modelo	$P(+ A)$	Classe
1	0.95	+
2	0.93	+
3	0.87	-
4	0.85	-
5	0.85	-
6	0.85	+
7	0.76	-
8	0.53	+
9	0.43	-
10	0.25	+



# Construindo a curva ROC

Class	+	-	+	-	-	-	+	-	+	+	
Limiar >=	0.25	0.43	0.53	0.76	0.85	0.85	0.85	0.87	0.93	0.95	1.00
TP	5	4	4	3	3	3	3	2	2	1	0
FP	5	5	4	4	3	2	1	1	0	0	0
TN	0	0	1	1	2	3	4	4	5	5	5
FN	0	1	1	2	2	2	2	3	3	4	5
TPR	1	0.8	0.8	0.6	0.6	0.6	0.6	0.4	0.4	0.2	0
FPR	1	1	0.8	0.8	0.6	0.4	0.2	0.2	0	0	0

Curva ROC:



# Teste de significância

- Dado dois modelos:
  - M1: precisão = 85%, testado em 30 casos
  - M2: precisão = 75%, testado em 5000 casos
- Podemos dizer M1 é **melhor** do que M2?
- Quanta **confiança** podemos colocar na precisão do M1 e M2?
- Pode a diferença de medir o desempenho ser explicada como **resultado de flutuações aleatórias** no conjunto de teste?

# Teste de hipóteses

- Testes de hipóteses são testes estatísticos que comparam duas amostras segundo uma determinada hipótese
  - Exemplo: geralmente se assume uma hipótese (**hipótese nula**) que os resultados obtidos de um modelo não poderiam ser obtidos por um outro, ou seja, que os resultados de ambos não provêm da mesma população
- No exemplo anterior, se aplica o teste de hipótese para verificar se a diferença de precisão de M1 é **significante** em relação à M2

# Testes de hipóteses

- Entre dois classificadores
  - teste –  $t$ 
    - Assume que os dados seguem a distribuição normal
  - Wilcoxon
    - Teste não paramétrico que utiliza *ranks*
- Múltiplas comparações
  - ANOVA
    - Assume que os dados seguem a distribuição normal
  - Friedman
    - Teste não paramétrico que utiliza *ranks*