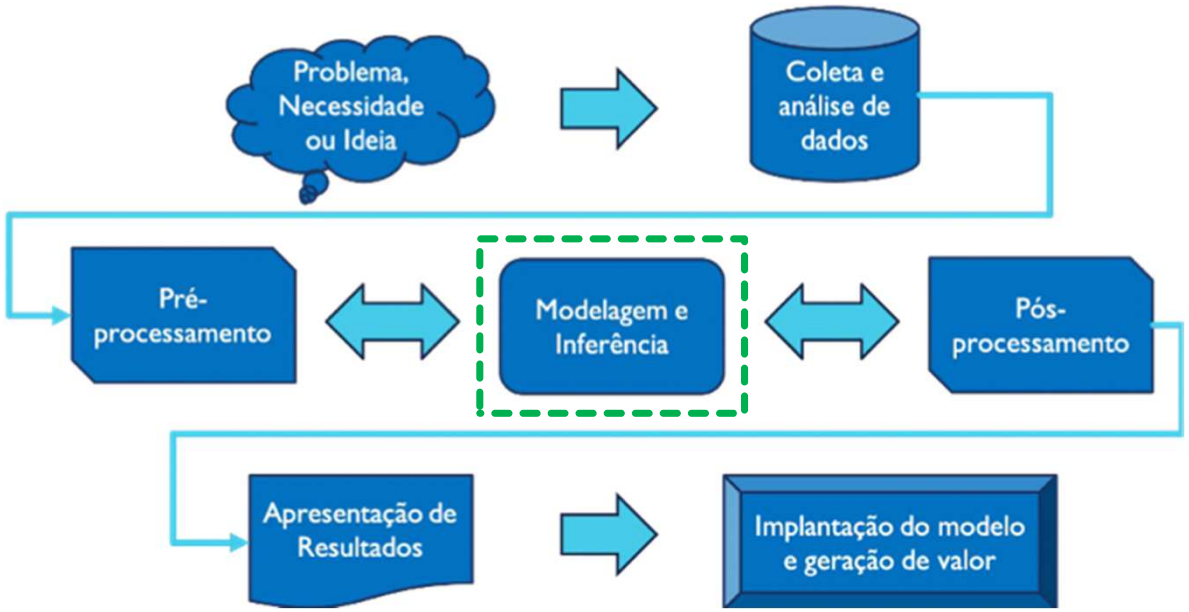
ESPECIALIZAÇÃO EM CIÊNCIA DE DADOS – PUC-RIO
MACHINE LEARNING

AULA 2: PROBLEMAS DE CLASSIFICAÇÃO

Tatiana Escovedo, PhD.
tatiana@inf.puc-rio.br

Problemas de Classificação

ESQUEMA BÁSICO DE UM PROJETO DE CD

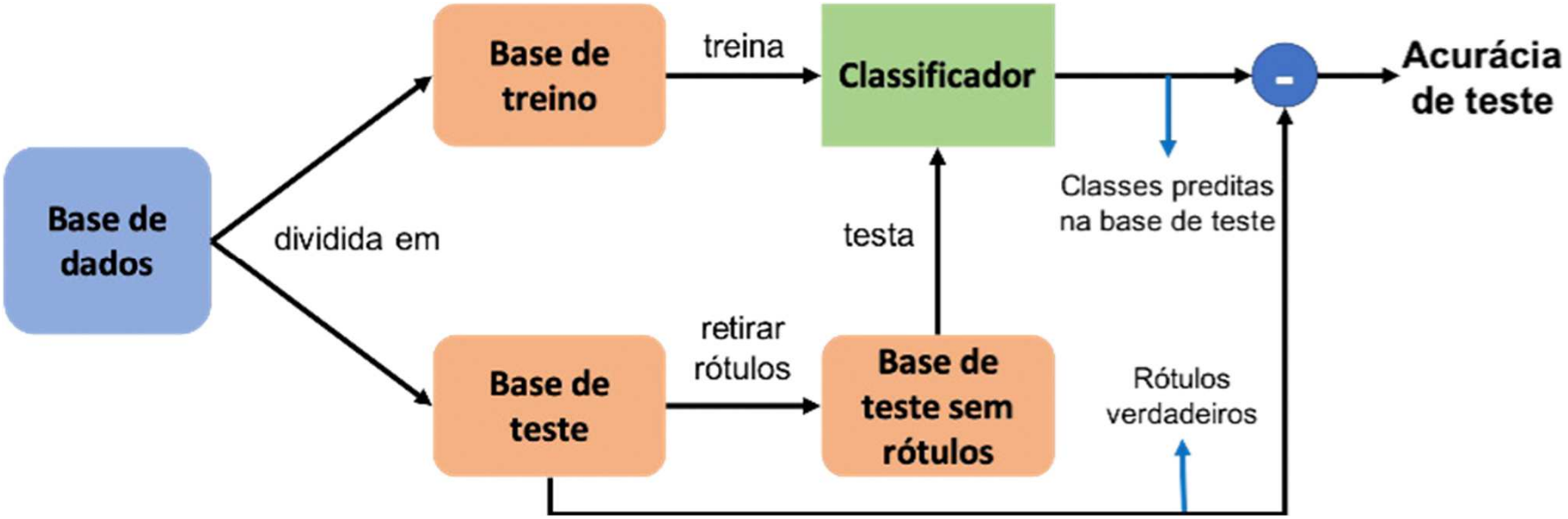


- 1 **Elencar** os modelos possíveis e passíveis para cada tipo de problema
- 2 **Estimar** os parâmetros que compõem os modelos, baseando-se nas instâncias e variáveis pré-processadas
- 3 **Avaliar** os resultados de cada modelo, usando métricas e um processo justo de comparação

CLASSIFICAÇÃO

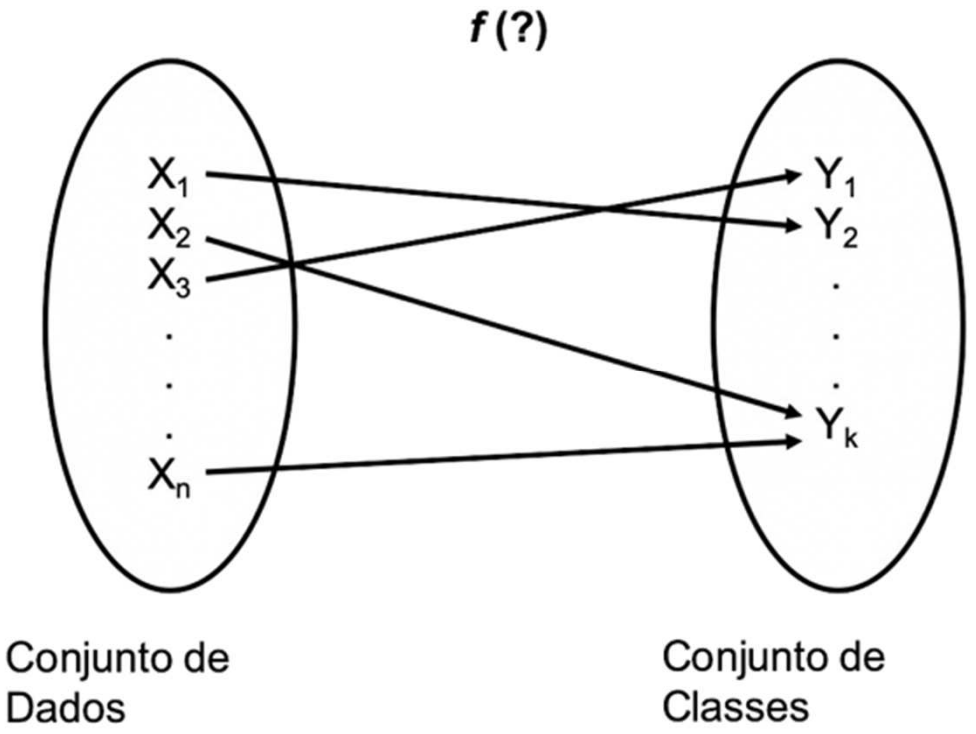
- Uma das tarefas de KDD mais importantes e mais populares
- Utiliza aprendizado **supervisionado**
- Utiliza dois grupos: **X**, com os atributos a serem utilizados na previsão do valor (atributos previsores ou de predição) e **Y**, com o atributo para o qual se deve fazer a predição do valor (atributo-alvo)
- O atributo alvo é **categórico**

CLASSIFICAÇÃO



DEFINIÇÃO INFORMAL

- Busca por uma função que permita **associar** corretamente cada registro X_i de um conjunto de dados a um único rótulo categórico, Y_i, denominado **classe**.
- Uma vez identificada, esta função pode ser aplicada a **novos** registros para **prever** as classes em que eles se enquadram.



DEFINIÇÃO FORMAL

- Dada uma coleção de exemplos ***f***, obter uma função (hipótese) ***h*** que seja uma aproximação de ***f***.
- A imagem de ***f*** é formada por rótulos de classes retirados de um conjunto finito e toda hipótese ***h*** é chamada de **Classificador**.
- O aprendizado consiste na busca da hipótese ***h*** que mais se aproxime da função original ***f***.

AVALIAÇÃO: ACURÁCIA DO CLASSIFICADOR

- Medida de desempenho: **acurácia**, ou taxa de acerto do classificador:

$$Acc(h) = 1 - Err(h)$$

- A acurácia é uma função da **taxa de erro** (ou taxa de classificação incorreta):

$$Err(h) = \frac{1}{n}\sum_{i=1}^{n} \|y_i \neq h(i)\|$$

Onde:

- O operador *//**E**//* retorna 1 se a expressão *E* for verdadeira e 0 em caso contrário.
- ***n*** é o número de exemplos (registros da base de dados)
- y_i é a classe real associada ao i-ésimo exemplo
- ***h(i)*** é a classe indicada pelo classificador para o i-ésimo exemplo

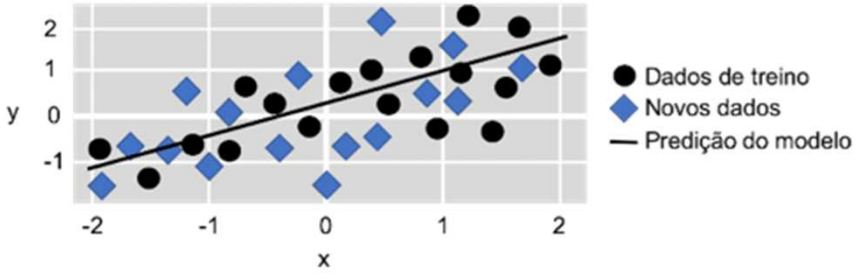
CLASSIFICAÇÃO: PROBLEMAS

- Uma vez identificada uma hipótese (classificador), esta pode ser muito **específica** para o conjunto de treinamento utilizado.
- Caso este conjunto não corresponda a uma amostra suficientemente representativa da população, o Classificador pode ter um bom desempenho no conjunto de **treinamento**, mas não no conjunto de **teste**.
  - ***Overfitting:*** o classificador se ajustou em excesso ao conjunto de treinamento.
- O algoritmo de aprendizado pode ter parametrizações inadequadas.
  - ***Underfitting:*** o classificador se ajustou pouco ao conjunto de treinamento

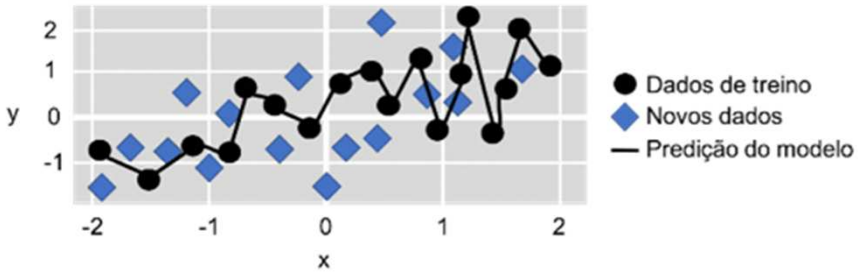
OVERFITTING

- Geralmente, o erro de generalização (teste) é maior que o erro de treinamento. Idealmente, estes erros são próximos.
- Se o erro de generalização é **grande demais**, pode estar ocorrendo ***overfitting***: o modelo está memorizando os padrões de treinamento em vez de descobrir regras ou padrões generalizados.
- Modelos mais simples tendem a generalizar melhor.

Um modelo bem ajustado irá apresentar erros da mesma magnitude tanto nos novos dados quanto nos dados de treino.

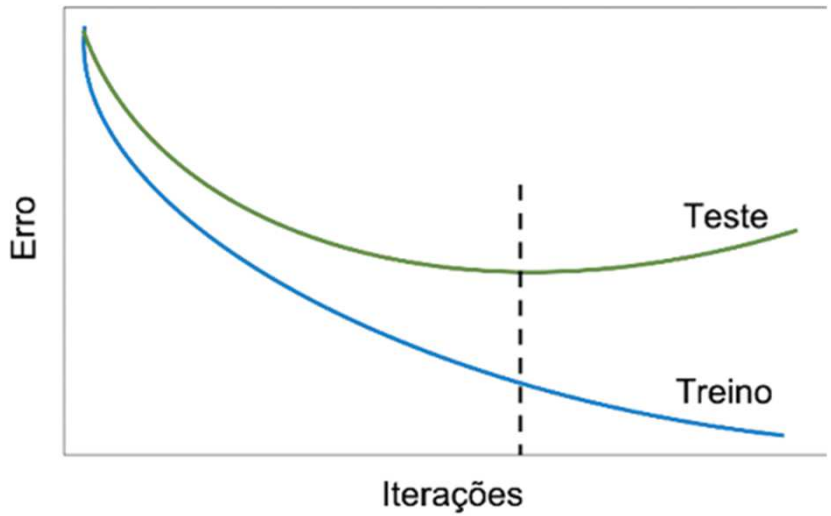


Um modelo com overfitting "memoriza" os dados de treino e apresentará erros muito maiores nos novos dados.



DILEMA BIAS X VARIÂNCIA (FLEXIBILIDADE X QUALIDADE)

- Na aprendizagem supervisionada, ao mesmo tempo em que o modelo preditivo precisa ser suficientemente **flexível** para aproximar os dados de treinamento, o processo de treinamento deve **evitar** que o modelo absorva os **ruídos** da base.
- O modelo deve **capturar** as regularidades dos dados de treinamento, mas também **generalize** bem para dados desconhecidos.
- **Solução:** escolher o momento adequado para interromper o treinamento (pode usar validação cruzada).



AVALIAÇÃO: MATRIZ DE CONFUSÃO

- A **matriz de confusão** oferece um detalhamento do desempenho do modelo de classificação: mostra, para cada classe, o número de classificações corretas em relação ao número de classificações indicadas pelo modelo.

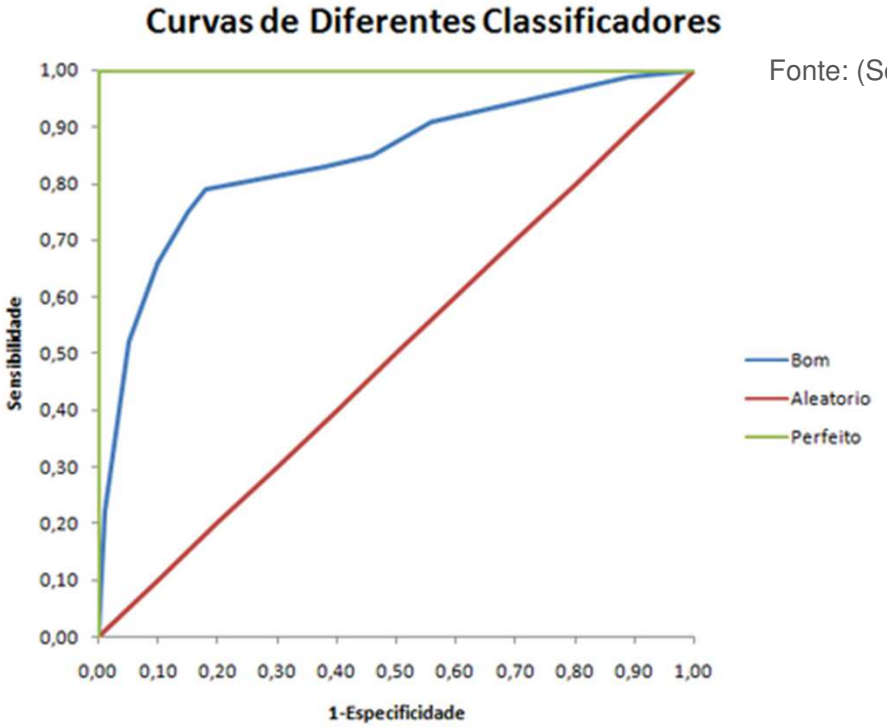
- Falso Positivo (FP) também é conhecido como **Alarme Falso**; e Falso Negativo (FN) como **Alarme Defeituoso**.

Classes	Predita C1	Predita C2
Verdadeira C1	Verdadeiros Positivos	Falsos Negativos
Verdadeira C2	Falsos Positivos	Verdadeiros Negativos

Classes	Predita C1	Predita C2	...	Predita Cn
Verdadeira C1	M(C1, C1)	M(C1, C2)	...	M(C1, Cn)
Verdadeira C2	M(C2, C1)	M(C2, C2)	...	M(C2, Cn)
...	...	...	...	...
Verdadeira Cn	M(Cn, C1)	M(Cn, C2)	...	M(Cn, Cn)

OUTRAS MÉTRICAS DE AVALIAÇÃO

- **Taxa de Falsos Positivos** (TFP = FP/(FP+VN))
- **Curva ROC** (*Receiver Operating Characteristic*): contrasta os benefícios de uma classificação correta (TVP: sensibilidade) e o custo de uma classificação incorreta (TFP: 1-especificidade).
  - ***Sensibilidade:*** capacidade de identificar corretamente os indivíduos que **apresentam** a característica de interesse.
  - ***Especificidade:*** capacidade em identificar corretamente os indivíduos que **não** apresentam a condição de interesse.



Fonte: (Souza, 2009)

TEOREMA NFL (NO FREE LUNCH)

- **Não existe** um algoritmo de aprendizado que seja **superior** a todos os demais quando considerados todos os problemas de classificação possíveis.
- A cada problema, os algoritmos disponíveis devem ser **experimentados** a fim de identificar aqueles que obtêm melhor desempenho.

- **Classificador nulo:** sempre retorna apenas uma classe (geralmente a mais frequente). Nosso classificador deve ser melhor que o nulo!

EXEMPLO

Sexo	País	Idade	Compra
M	França	25	Sim
M	Inglaterra	21	Sim
F	França	23	Sim
F	Inglaterra	34	Sim
F	França	30	Não
M	Alemanha	21	Não
M	Alemanha	20	Não
F	Alemanha	18	Não
F	França	34	Não
M	França	55	Não

Clientes e suas compras em um tipo de literatura

Se País = Alemanha Então Compra = Não

Se País = Inglaterra Então Compra = Sim

Se País = França e Idade ≤ 25 Então Compra = Sim

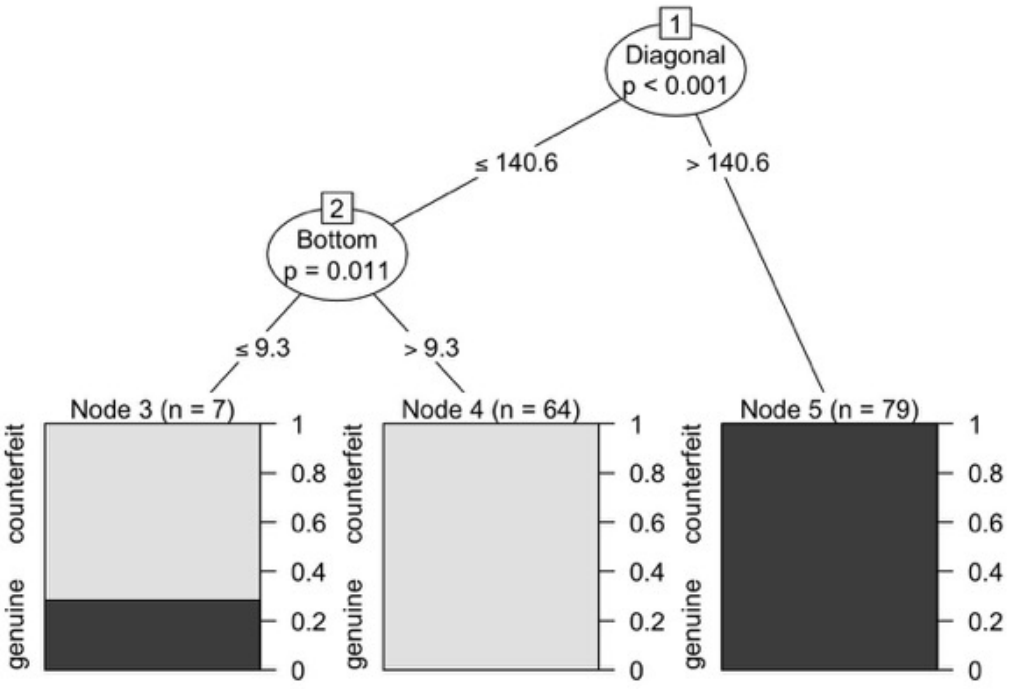
Se País = França e Idade > 25 Então Compra = Não

Árvore de Decisão

ÁRVORE DE DECISÃO

- Um dos modelos de decisão mais **simples**
- Usam amostras das características dos dados para criar **regras** de decisão no formato de **árvore**
- Inspiradas na forma que **humanos** tomam decisão
- Apresentam a informação **visualmente**, de uma forma fácil de entender
- Podem ser usadas para problemas de **classificação ou regressão**
- Reduzem os dados em um **conjunto de regras** que podem ser usadas para uma decisão de classificação ou gerar uma predição

ÁRVORE DE CLASSIFICAÇÃO



- Aliam **acurácia** e **interpretabilidade**
- Possibilitam **seleção automática de variáveis** para compor suas estruturas
- Cada nó interno representa uma **decisão** sobre um atributo que determina como os dados estão particionados pelos seus nós filhos.
- Para classificar um novo exemplo, basta **testar** os valores dos atributos na árvore e percorrê-la até se atingir um nó folha (classe predita).

ÁRVORE DE CLASSIFICAÇÃO

- Há diferentes algoritmos para sua elaboração:
  - CHAID
  - Ctree
  - **C4.5**
  - CART
  - Hoeffding Tree

- Em geral, a construção da árvore é realizada de acordo com alguma abordagem **recursiva** de particionamento do conjunto de dados.
- Todos os algoritmos são bem parecidos.
- A principal distinção está nos processos de:
  - Seleção de variáveis
  - Critério para particionar
  - Critério de parada para o crescimento da árvore

C4.5

- Utiliza conceitos e medidas da **Teoria da Informação**.
- Inicialmente, a **raiz** da árvore contém todo o conjunto de dados com exemplos misturados de várias classes.
- Um **predicado** (ponto de separação) é escolhido como sendo a condição que melhor separa ou discrimina as classes. Um predicado é um dos atributos previsores do problema e induz uma divisão do conjunto de dados em dois ou mais conjuntos disjuntos, cada um deles associado a um **nó filho**.
- Cada novo **nó** abrange um subconjunto do conjunto de dados original que é recursivamente separado até que o subconjunto associado a cada **nó folha** consista inteira ou predominantemente de registros de uma mesma classe.

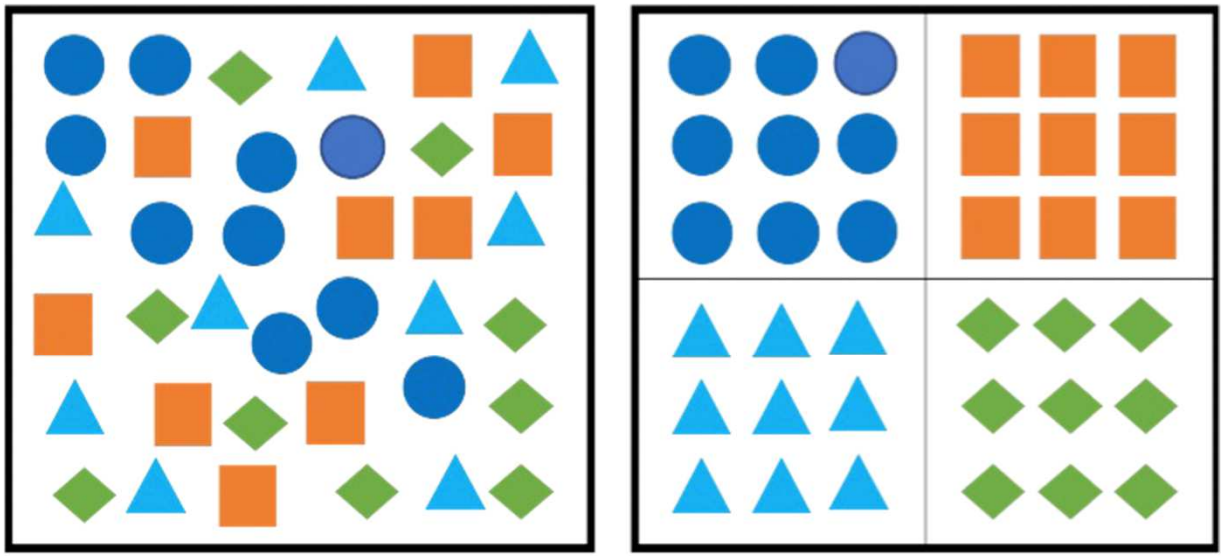
C4.5: FASE DE CONSTRUÇÃO

- Uma árvore é gerada pelo **particionamento recursivo** do conjunto de dados de treinamento.
- O conjunto de treinamento é separado em dois ou mais subconjuntos por meio da definição de **predicados** sobre os conjuntos de valores de cada atributo previsor.
- O processo de **divisão** é repetido **recursivamente** até que todos ou a maioria dos exemplos de cada subconjunto pertençam a uma **classe**.
- Os **nós folhas** da árvore abrangem todo o conjunto de treinamento.
- Todos os nós em uma determinada altura devem ser processados antes do processamento do nível subsequente.

C4.5: FASE DE CONSTRUÇÃO

- Duas operações principais durante o processo de construção:
  - (1) Avaliação de **pontos de separação** em potencial para identificação de qual o melhor entre eles
  - (2) Criação das **partições**, usando o melhor ponto de separação selecionado, para os casos pertencentes a cada nó.
- Uma vez determinado o melhor ponto de separação de cada nó, as partições podem ser criadas pela simples aplicação do critério de partição selecionado.

ENTROPIA



Alta entropia

Baixa entropia

C4.5: (1) AVALIAÇÃO DOS PONTOS DE SEPARAÇÃO

(i) Cálculo da entropia de *T*

- A **entropia** de um conjunto de dados é um valor entre 0 e 1 que representa a sua **pureza**.
- Se em um determinado conjunto todos os registros são da mesma classe, a entropia é 0.
- Se cada registro é de uma classe diferente, a entropia é 1.

$$H(T) = -\sum_{j=1}^{k} \frac{freq(c_j, t)}{|T|} \times log_2 \frac{freq(c_j, t)}{|T|}$$

- ***T***: conjunto de registros de entrada
- ***freq*(c_j, *t*)**: quantidade de registros da classe c_j em T
- **/*T*/**: número total de registros em T
- ***k***: número de classes distintas que ocorrem em T

C4.5: (1) AVALIAÇÃO DOS PONTOS DE SEPARAÇÃO

(ii) Cálculo do valor esperado da entropia

- $info_{Ai}(T)$: valor esperado da entropia uma vez que T é dividido com os valores do atributo A_i
- A_i deve ser escolhido dentre os atributos ainda não utilizados na definição da árvore.
- Considerar que existam valores no domínio A_i e que A_i induz uma partição sobre T, resultando em subconjuntos $\{T\}$.

$$info_{A_i}(T) = j = 1 \sum_{j=i}^{n} \frac{|T_j|}{|T|} \times H(T_j)$$

- $\boldsymbol{T}$: conjunto de registros de entrada
- $\boldsymbol{T_j}$: cada um dos subconjuntos da partição induzida por A_i sobre T
- $|\boldsymbol{T_j}|$: quantidade de registros em T_j
- $\boldsymbol{n}$: número de atributos

C4.5: (1) AVALIAÇÃO DOS PONTOS DE SEPARAÇÃO

(iii) Cálculo do ganho de informação do atributo A_j com relação a T

- Representa a redução na impureza na situação em que os valores de A_j são usados para subdividir T

$$GInfo(A_j, T) = H(T) - info_{A_i}(T)$$

- Após calcular $GInfo(A_j, T)$ para cada atributo previsor remanescente, o C4.5 seleciona o atributo com maior ganho de informação para associar ao nó da árvore.

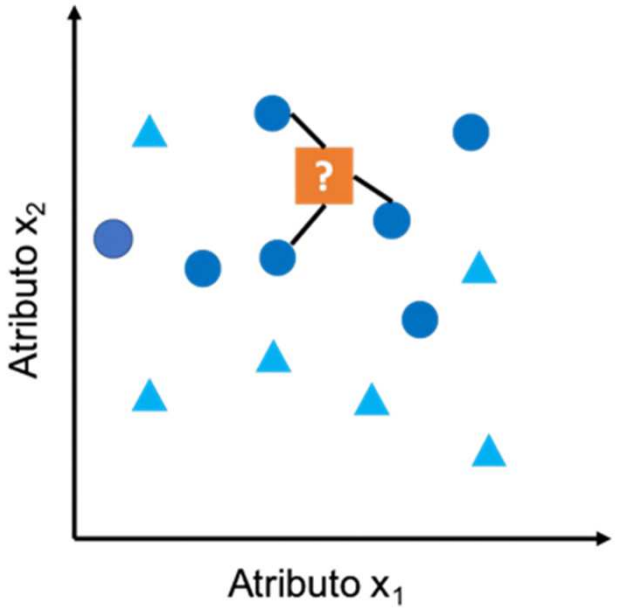
C4.5: ALGORITMO

- **1.** Calcular a entropia do conjunto *T* completo
- **2.** Para cada atributo
- **2.1.** Calcular o ganho de informação
- **3.** Selecionar o atributo com maior ganho de informação para o nó raiz da árvore
- **4.** Subdividir o conjunto *T*
- **5.** Repetir o procedimento para cada nó gerado

KNN

KNN

- **KNN: k-Nearest Neighbours (k-Vizinhos Mais Próximos)**
- Algoritmo **simples** de entender e que **funciona** muito bem na prática.
- Algoritmo **não-paramétrico**: não assume premissas sobre a distribuição dos dados.
- Considera que os exemplos **vizinhos** são **similares** ao exemplo que se deseja classificar.



Considera que os registros do conjunto de dados correspondem a pontos no R^n, em que cada atributo corresponde a uma dimensão deste espaço.

KNN: FUNCIONAMENTO

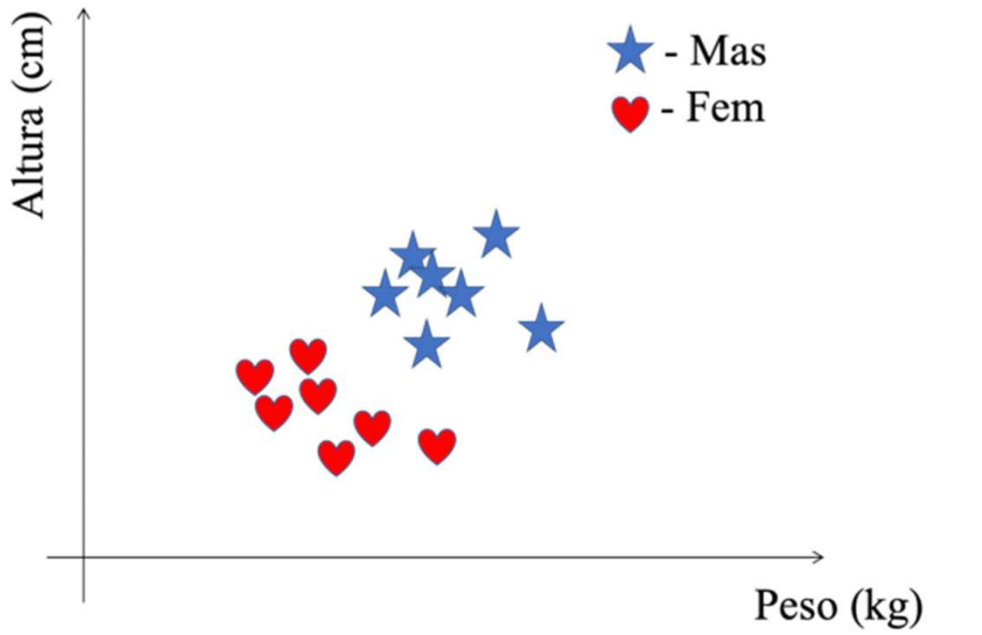
- O conjunto de dados (rotulado) é armazenado. Quando um novo registro deve ser classificado, ele é comparado a todos os registros do conjunto de treinamento para identificar ***k*** (parâmetro de entrada) **vizinhos mais próximos** (mais semelhantes) de acordo com alguma métrica de distância.
- A classe do novo registro é determinada por **inspeção** das classes desses vizinhos mais próximos, de acordo com a métrica selecionada.
- Na maioria das implementações, os atributos são **normalizados**, para que tenham a mesma contribuição na predição da classe

KNN: ALGORITMO

1. Definição da métrica de distância, critério de desempate e valor de *k*
2. Cálculo da distância do novo registro a cada um dos registros existentes no conjunto de referência.
3. Identificação dos *k* registros do conjunto de referência que apresentaram menor distância em relação ao novo registro (mais similares).
4. Apuração da classe mais frequente entre os *k* registros identificados no passo anterior (usando votação majoritária)

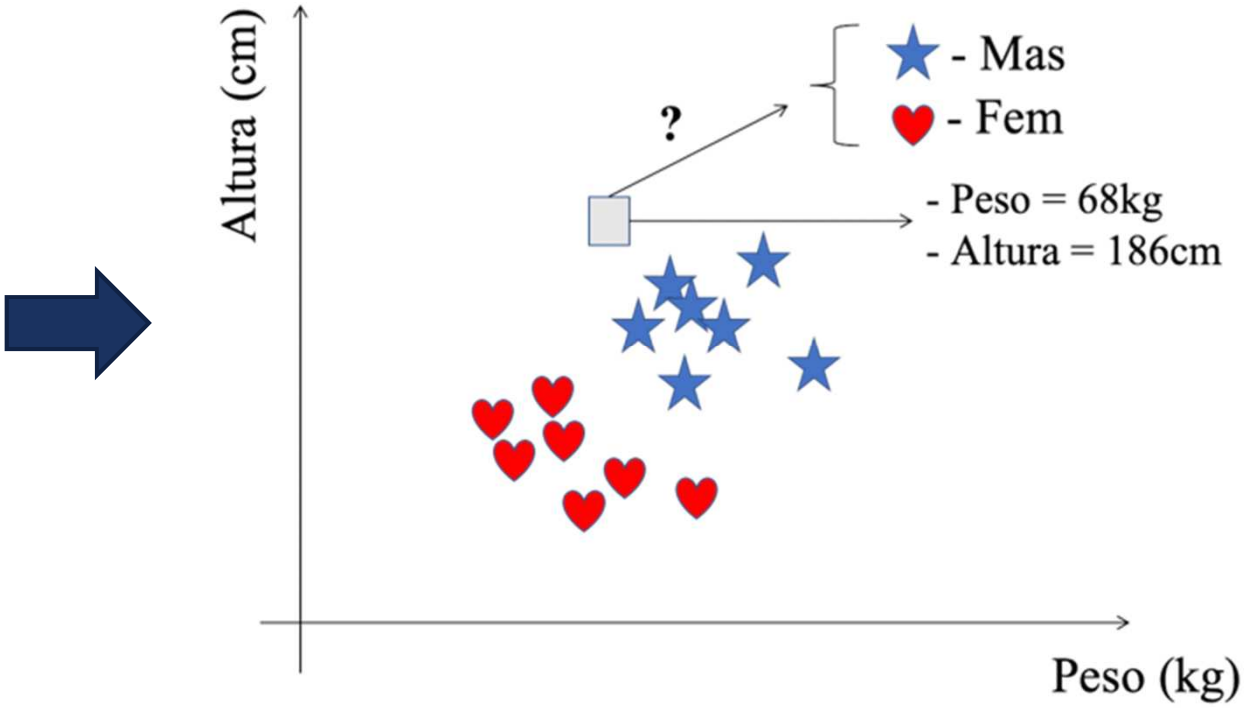
KNN: EXEMPLO

id	Peso (kg)	Altura (cm)	Sexo
1	55	167	F
2	52	160	F
3	48	155	F
...	...	...	...
12	70	175	M
13	74	170	M
14	80	180	M



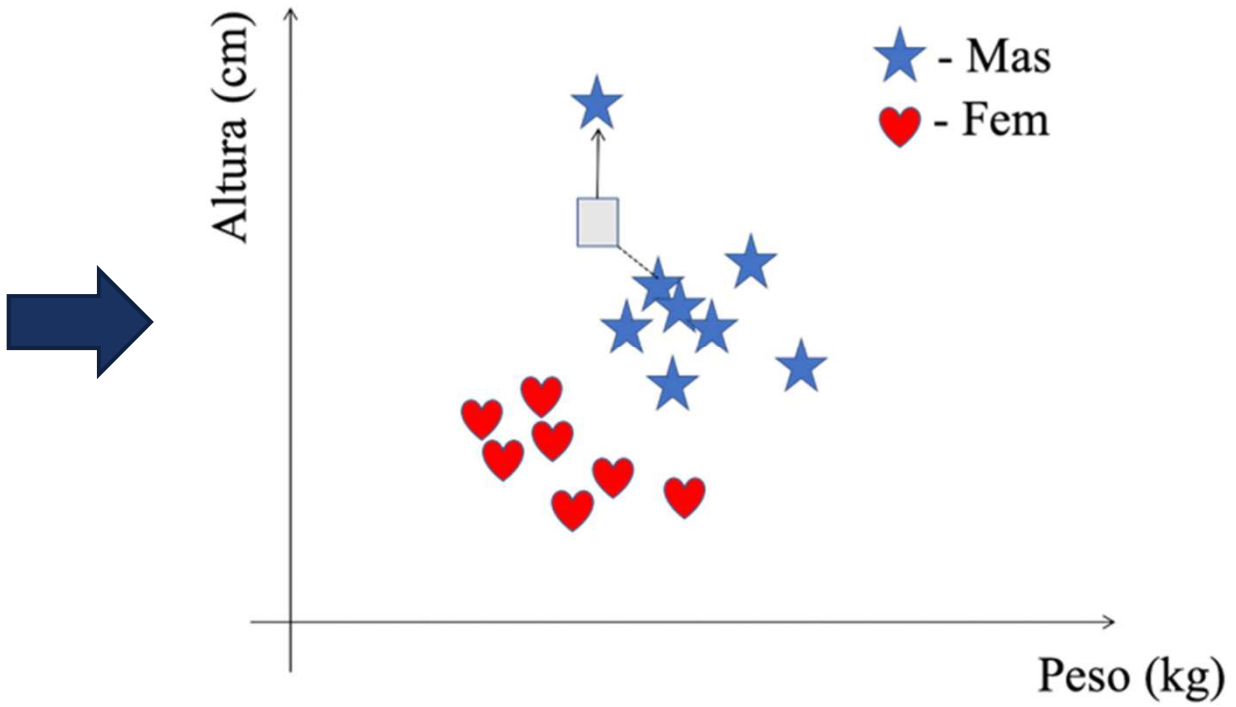
KNN: EXEMPLO 1

- Imagine que neste momento chega um novo indivíduo, com 68 kg e altura 186 cm, e deseja-se determinar o sexo deste indivíduo.



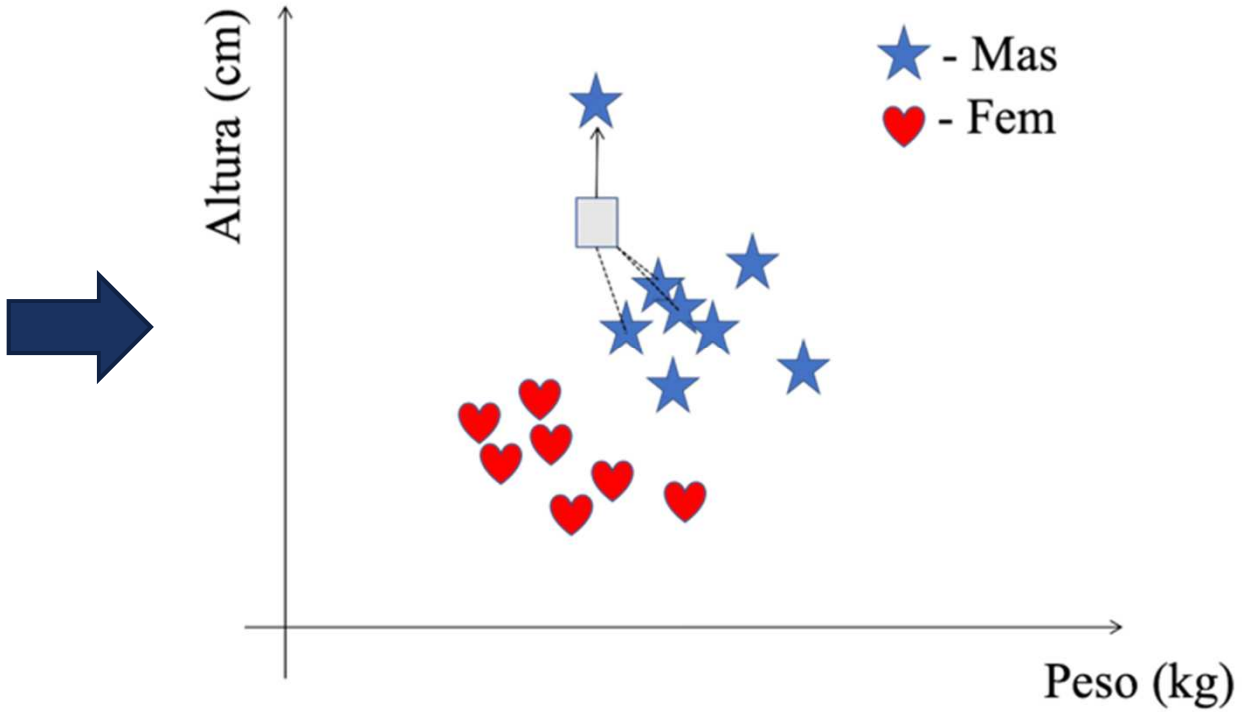
KNN: EXEMPLO 1

- Podemos utilizar o KNN para determinar o sexo deste novo indivíduo, que é dado pela ¸informação dos k vizinhos mais próximos.
- Precisamos então determinar o valor de k.
- Para **k = 1**:



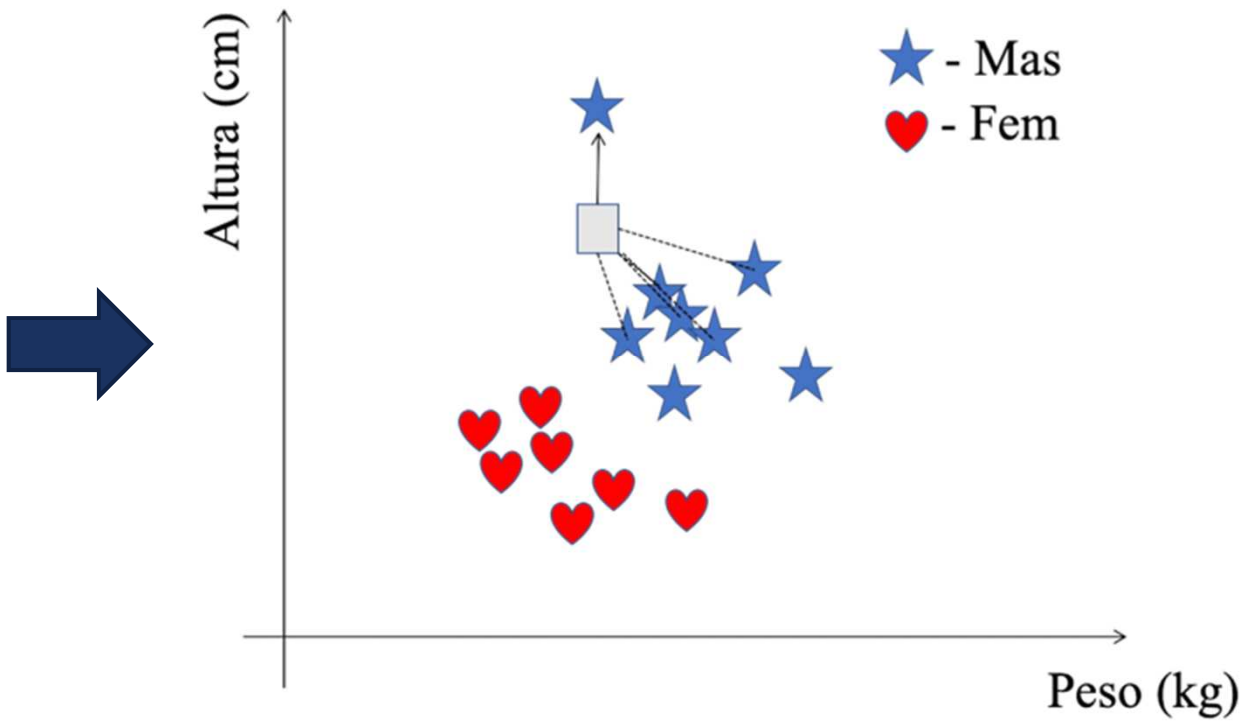
KNN: EXEMPLO

- Para **k = 3**:



KNN: EXEMPLO

- Para **k = 5**:



KNN: PERGUNTAS

1. Que tipo de **distância** usar?

2. Qual o **valor** adequado de ***k***?

PERGUNTA 1) QUE TIPO DE DISTÂNCIA USAR?

Euclidiana: $d(x_j, y_j) = \sqrt{\sum_{j=1}^{J}(x_j - y_j)^2}$

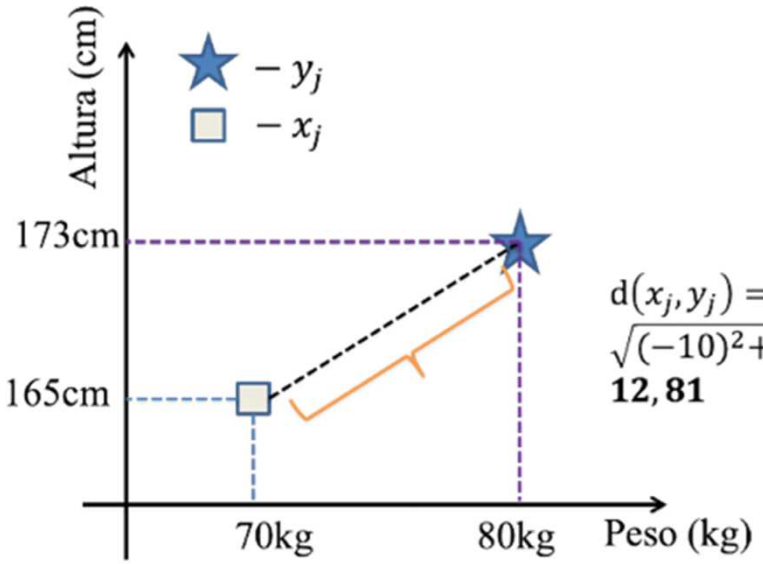
Manhattan: $d(x_j, y_j) = \sum_{j=1}^{J}|x_j - y_j|$

Minkowski: $d(x_j, y_j) = \max(|x_j - y_j|)$

DISTÂNCIA EUCLIDIANA

- **Euclidiana**

$$d(x_j, y_j) = \sqrt{\sum_{j=1}^{J}(x_j - y_j)^2}$$



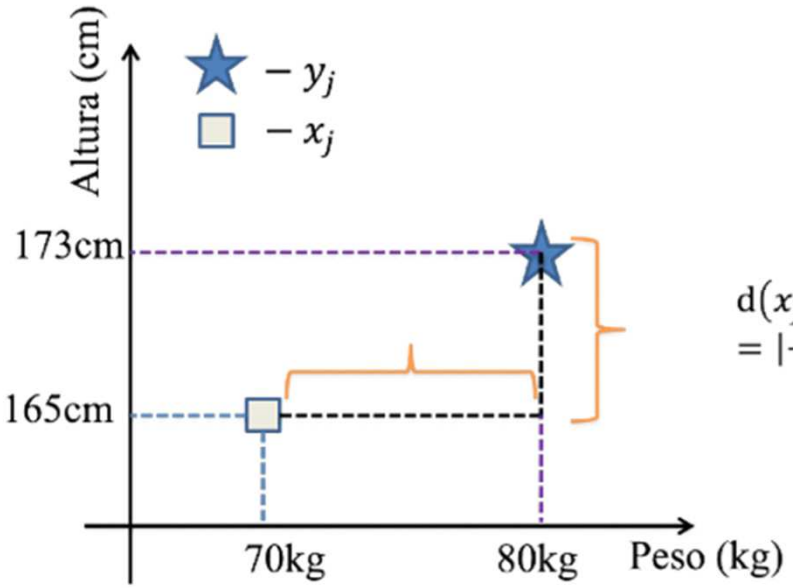
Euclidiana

$$d(x_j, y_j) = \sqrt{(70-80)^2 + (165-173)^2} = \sqrt{(-10)^2 + (-8)^2} = \sqrt{100+64} = \sqrt{164} = \mathbf{12,81}$$

DISTÂNCIA DE MANHATTAN

- **Manhattan**

$$d(x_j, y_j) = \sum_{j=1}^{J} |x_j - y_j|$$



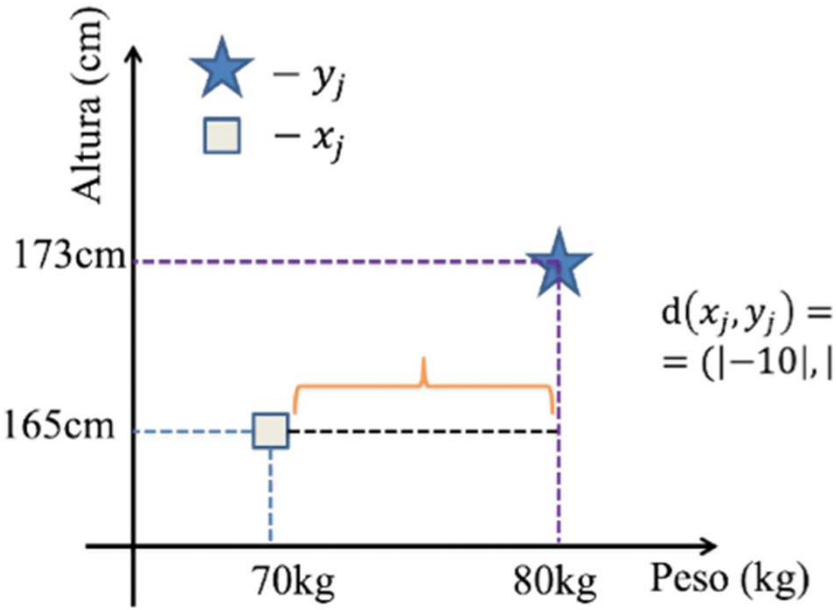
Manhattan

$$d(x_j, y_j) = |70 - 80| + |165 - 173|$$
$$= |-10| + |-8| = 10 + 8 = \mathbf{18}$$

DISTÂNCIA DE MINKOWSKI

- **Minkowski**

$$d(x_j, y_j) = \max_j(|x_j - y_j|)$$



Minkowski

$$d(x_j, y_j) = \max(|70 - 80|, |165 - 173|) =$$
$$= (|-10|, |-8|) = \max(10,8) = \mathbf{10}$$

PERGUNTA 1) QUE TIPO DE DISTÂNCIA USAR?

- Para cada distância, há um resultado **diferente**!
- Qual distância escolher?
  - A decisão é **experimental**: devem ser criados diversos modelos diferentes, variando os parâmetros de distância, e aquele que apresentar melhor resultado (considerando, por exemplo, a acurácia de teste) será o melhor candidato.

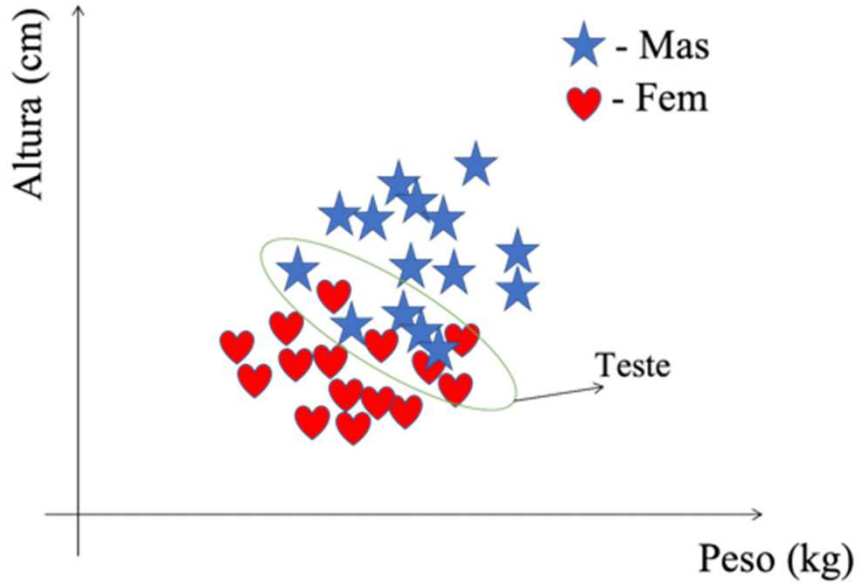
PERGUNTA 2) QUAL O VALOR ADEQUADO DE K

- Normalmente determinado em função do conjunto de dados.
- Em geral, quanto maior o valor de *k*, menor o efeito de eventuais ruídos no conjunto de referência, mas valores grandes de *k* tornam mais difusas as fronteiras entre as classes existentes.
- Decidido experimentalmente, usando **validação cruzada**.

VALIDAÇÃO CRUZADA PARA DETERMINAR O VALOR DE K

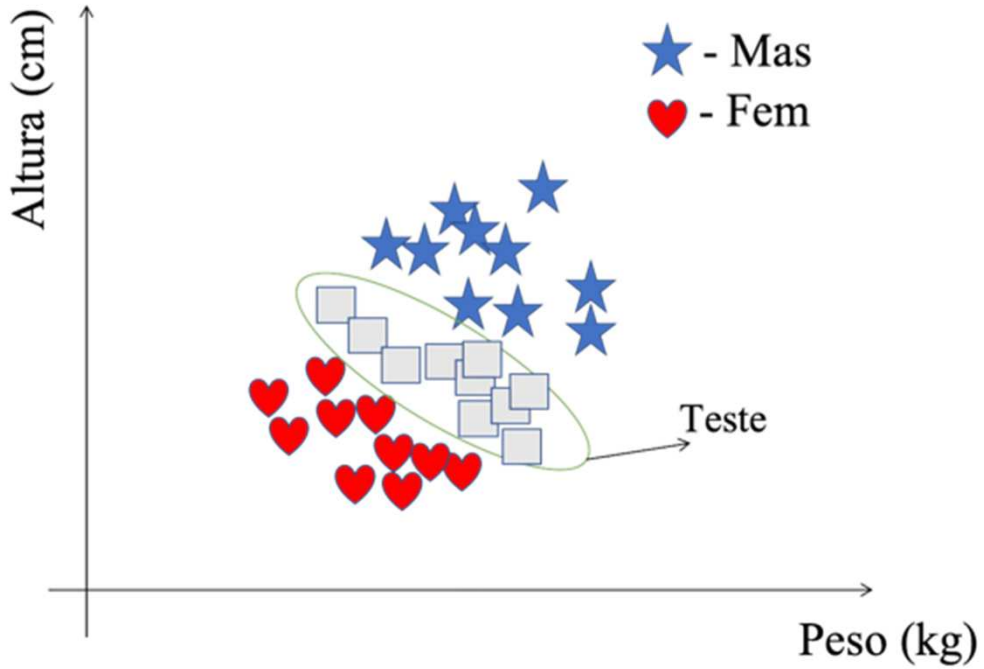
1. Dividir os dados em **treino** e **teste**:

	id	Peso (kg)	Altura (cm)	Sexo
Treinamento	1	55	167	F
	2	52	160	F
	3	48	155	F
	...	...	...	...
	9	58	165	F
	10	64	169	F
	11	78	179	M
	...	...	...	...
	18	70	175	M
	19	74	170	M
	20	80	180	M
Teste	21	57	162	F
	22	59	161	F
	...	...	...	...
	29	72	170	M
	30	69	165	M



VALIDAÇÃO CRUZADA PARA DETERMINAR O VALOR DE K

2. Assumir que não conhecemos os rótulos dos exemplos do conjunto de teste:



VALIDAÇÃO CRUZADA PARA DETERMINAR O VALOR DE K

3. Aplicar o modelo e calcular a acurácia de teste para ***k = 1***, ***k = 3*** e ***k = 5***:

k	Acurácia
1	6/10 = 60%
2	7/10 = **70%**
3	6/10 = 60%

- Seleciona-se o **k** com maior acurácia com os dados de teste (no caso, 3).
- Sugere-se variar **k** de 1 a 20 e também as diferentes distâncias.

KNN: LIMITAÇÕES

- A **performance** de classificação pode ser lenta em datasets grandes.
- É **sensível a características irrelevantes**, uma vez que todas as características contribuem para o cálculo da distância e consequentemente, para a predição.
- É necessário **testar** os valores de k e a métrica de distância a utilizar.

Naïve Bayes (Bayes Ingênuo)

NAÏVE BAYES: MOTIVAÇÃO

- Classificador **genérico**, de aprendizado **dinâmico**, **rápido** computacionalmente e que só precisa de um pequeno número de dados de treino.
- Especialmente adequado para **grande número** de atributos ou características.
- Primeiras **aplicações comerciais** foram para filtrar *spam* de e-mails (década de 90), usado para categorizar textos baseado na frequência das palavras usadas.
- Muito utilizado para previsões em tempo real, sistemas embarcados, classificação de textos e análise de sentimentos nas redes sociais.

NAÏVE BAYES: MOTIVAÇÃO

- **Naïve (Ingênuo)** porque desconsidera completamente a correlação entre os atributos (características).
  - Se determinado animal é considerado um “Gato” se tiver bigodes, orelhas em pé e aproximadamente 30 cm de altura , o algoritmo não vai levar em consideração a correlação entre esses fatores, tratando cada um de forma independente.
- **Bayes** porque baseia-se no **Teorema de Bayes**, estando relacionado com o cálculo de probabilidades condicionais

$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)},$$

em que A e B são eventos e $P(B) \neq 0$.

NAÏVE BAYES: CONSTRUÇÃO

- **Problema:** Definir a Liberação de Crediário no Varejo.
- Queremos estimar se um novo cliente, João, será um **bom pagador** ou **mau pagador**.

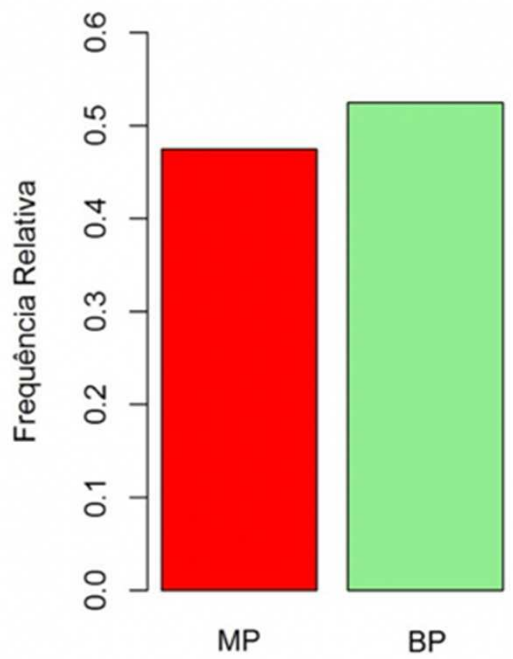
Cliente	Renda	Idade	Est. Civil	Pagador
Ana	360	25	C	MP
Bernardo	350	33	S	MP
Carlos	1100	56	S	MP
...	...	...	...	...
Xavier	600	33	S	MP
Zé	800	29	S	BP
João	**750**	**38**	**C**	**?**

NAÏVE BAYES: CONSTRUÇÃO

1. Calcular a frequência relativa de bons e maus pagadores considerando toda a base de dados.

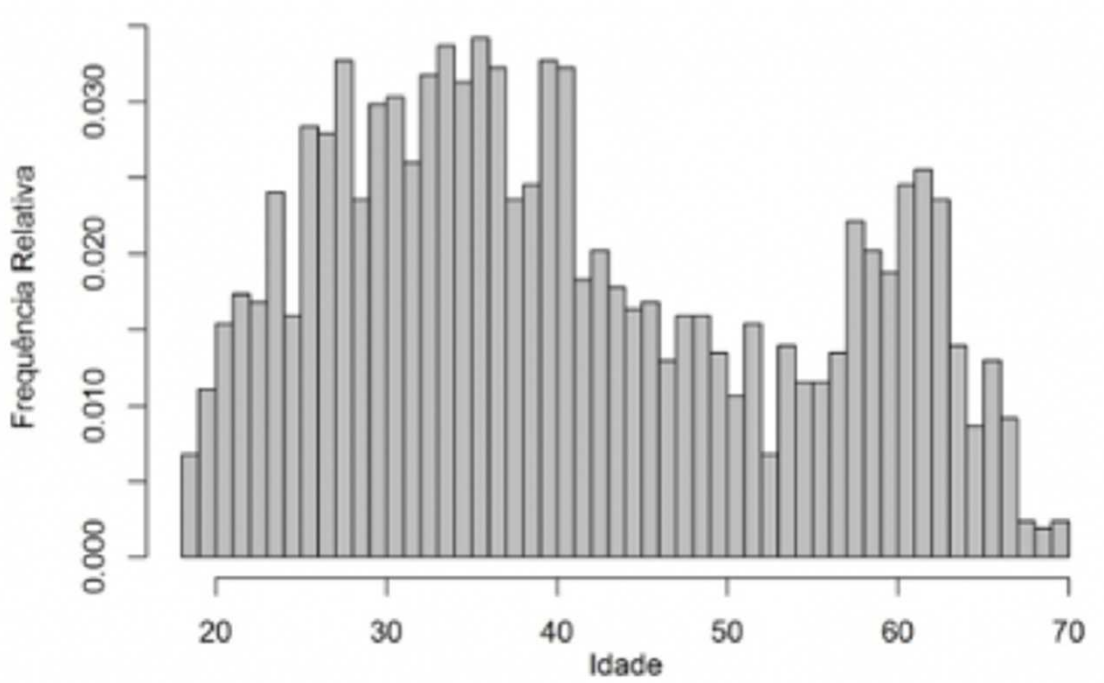
Para tal, calcula-se a frequência relativa, dividindo-se o total de pessoas pertencentes a cada classe pelo total de pessoas na base de dados.

Cliente	Pagador
Ana	MP
Bernardo	MP
Carlos	MP
...	...
Xavier	MP
Zé	BP
João	**?**



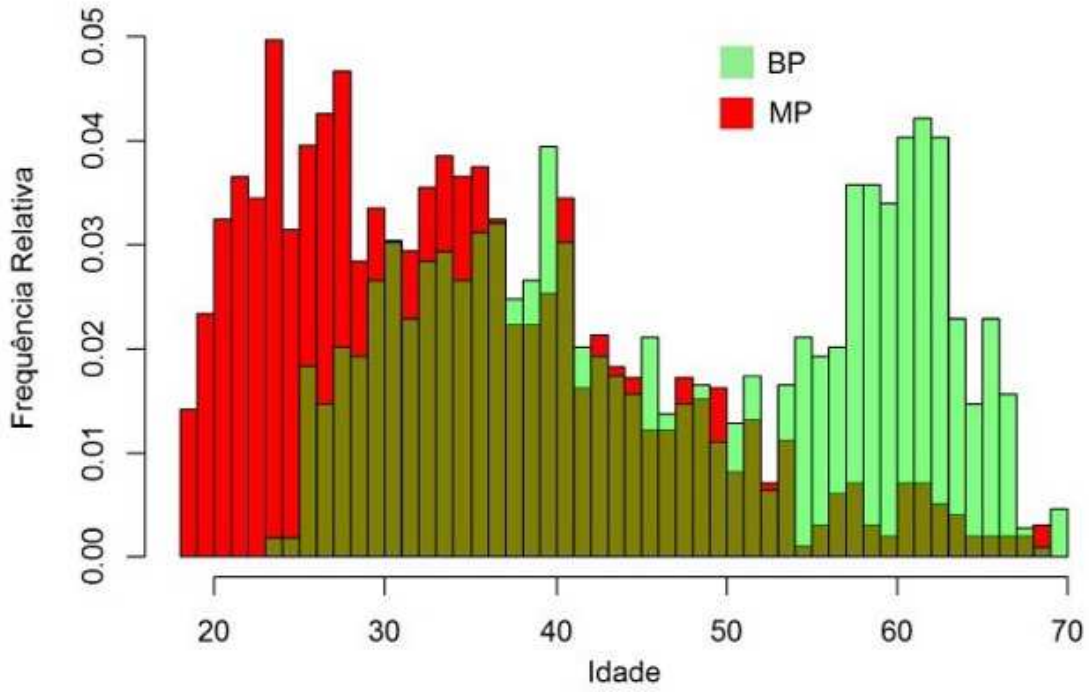
NAÏVE BAYES: CONSTRUÇÃO

2. Examinar a idade dos clientes, construindo um histograma de frequência relativa.



NAÏVE BAYES: CONSTRUÇÃO

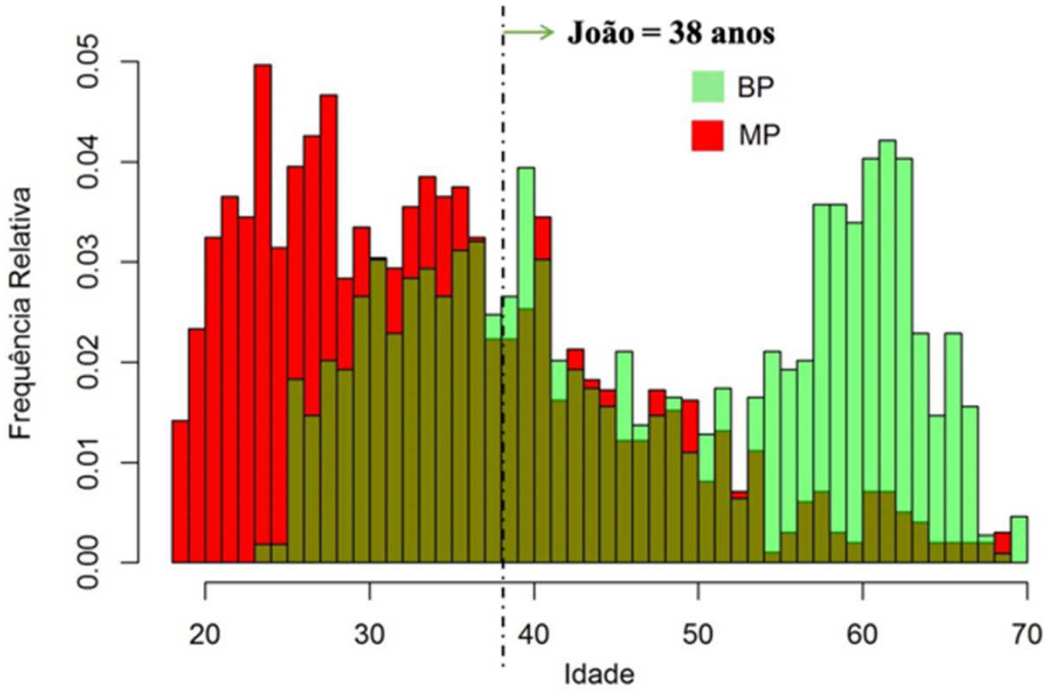
- Para facilitar a análise, vamos verificar a idade dos clientes por classe.
- Em verde, estão representados os bons pagadores e em vermelho os maus pagadores. As regiões em marrom representam a interseção das duas classes.



NAÏVE BAYES: MOTIVAÇÃO

- **Pergunta:** Dado o cliente João com a idade de 38 anos, qual a probabilidade dele pertencer aos MP ou BP?

- O Naive Bayes responde este problema calculando a probabilidade de João pertencer a uma das classes em função da frequência da sua idade (38 anos) nos grupos BP e MP e do histórico de bons e maus pagadores.



NAÏVE BAYES: CONSTRUÇÃO

Surgem novas Perguntas:

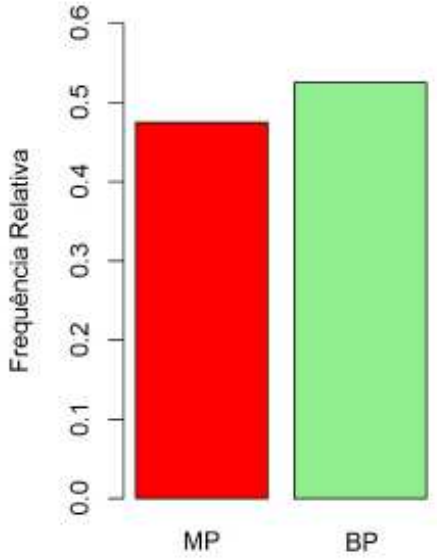
1. Como estimar a frequência histórica de bons e maus pagadores e a frequência da idade 38 anos nos dois grupos?
2. Como usar ambas as informações para inferir a probabilidade de João pertencer ao grupo BP e MP?

NAÏVE BAYES: CONSTRUÇÃO

- Vamos tratar primeiramente a primeira parte pergunta 1: *Como estimar a frequência histórica de bons e maus pagadores?*
- Já sabemos que a frequência histórica de bons e maus pagadores pode ser estimada dividindo o total de pessoas que pertençam a esta classe pelo total de pessoas da nossa base de dados histórica:

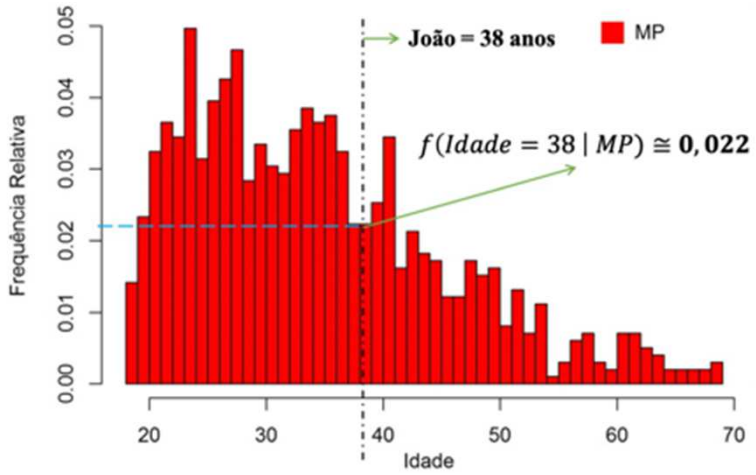
$$f(BP) = \frac{\#clientesBP}{\#\ clientes} = 0{,}53$$

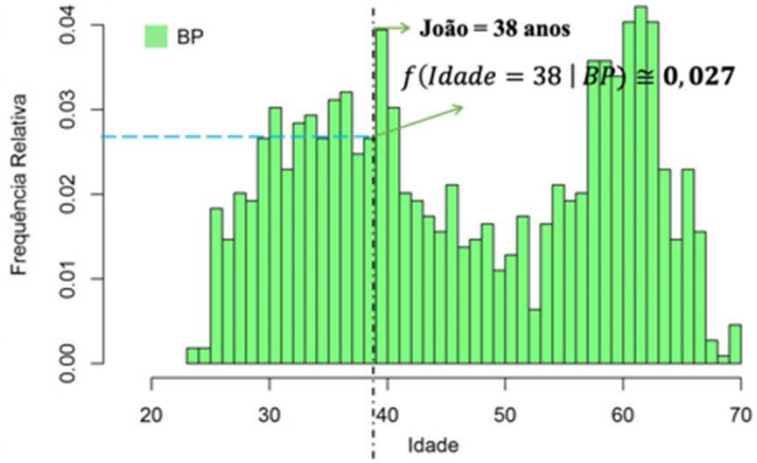
$$f(MP) = \frac{\#clientesMP}{\#\ clientes} = 0{,}47$$



NAÏVE BAYES: CONSTRUÇÃO

- 2ª parte da pergunta 1: *Como estimar a a frequência da idade 38 anos nos dois grupos?*
- <u>Forma 1:</u> Por **Histograma**: verificar a frequência relativa correspondente a 38 anos nos histogramas individuais de bons e maus pagadores.



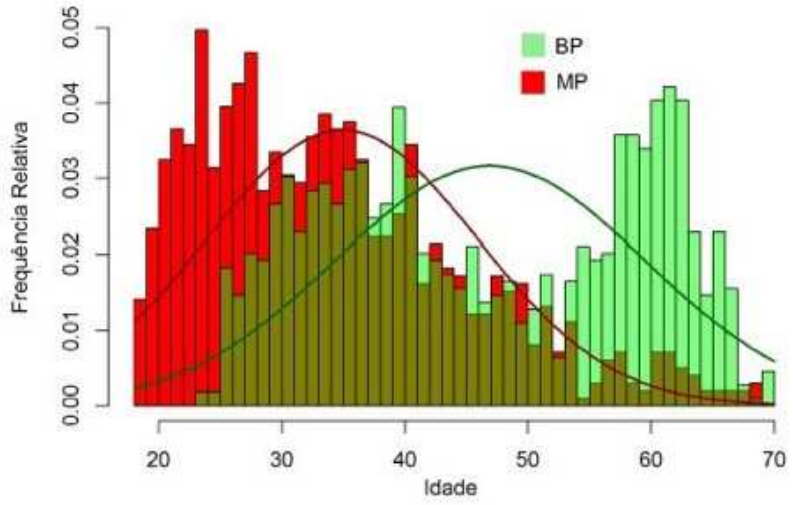


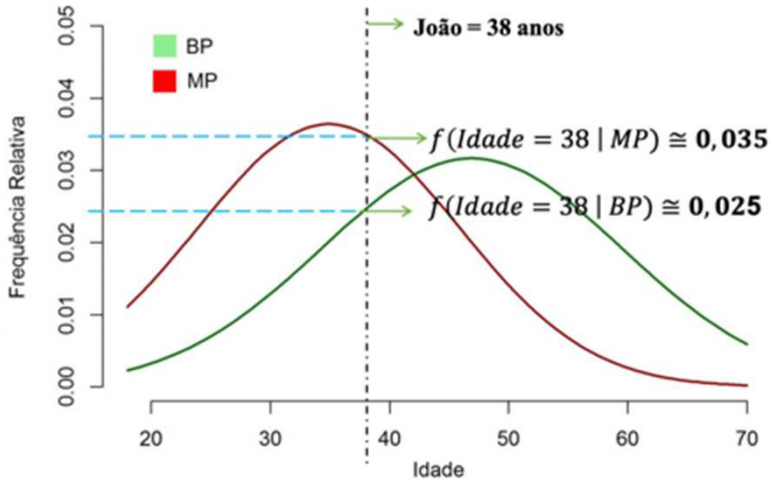
$$f(idade = 38|MP) = 0,022$$

$$f(idade = 38|BP) = 0,027$$

NAÏVE BAYES: CONSTRUÇÃO

- Forma 2: utilizar a **função de densidade** para estimar a frequência de idade em cada grupo (mais usual).





$$f(idade = 38|MP) = 0{,}035$$
$$f(idade = 38|BP) = 0{,}025$$

NAÏVE BAYES: CONSTRUÇÃO

Até agora, temos estimadores para as frequências:

- **1.** De Bons/Maus Pagadores:
  - *f(MP)* = 0,47
  - *f(BP)* = 0,53
- **2.** Da idade de 38 anos no Grupo de Bons/Maus Pagadores:
  - *f(Idade = 38|MP)* ≅ 0,035
  - *f(Idade = 38|BP)* ≅ 0,025

Como aproveitar essa informação para a Classificação?

$$f(MP|idade = 38) = ?$$

$$f(BP|idade = 38) = ?$$

Vamos usar a Regra de Bayes.

NAÏVE BAYES: CONSTRUÇÃO

A **Regra de Bayes**, é baseada no Teorema de Bayes e afirma que:

- a probabilidade a posteriori de uma hipótese ser validada pelos dados (*P(H|X)*)
- é proporcional a distribuição de probabilidade dos dados (função de verossimilhança– *P(X|H)*)
- multiplicada pela probabilidade a priori da hipótese ser verdadeira (*P(H)*).

NAÏVE BAYES: CONSTRUÇÃO

- **Regra de Bayes:** $P(MP|Idade = 38) \propto P(Idade = 38|MP) * P(MP)$

Verossimilhança

Posteriori

Priori

  - **Probabilidade a Priori:** pré-concebida antes dos fatos, o que nesse caso é expresso no desprezo da Idade do Indivíduo
  - **Verossimilhança:** compatibilidade entre a evidência (Idade = 38 anos) e a Classe MP
  - **Probabilidade a Posteriori:** resultante da combinação entre a observação dos fatos (Verossimilhança) e a visão "preconceituosa" sobre as Classes

NAÏVE BAYES: CONSTRUÇÃO

- **Exemplo:** Prever se o cliente é *BP* ou *MP* dada a sua categoria de idade (Júnior ou Sênior).

Id	Fx. Etária	Pagador
A	Senior	BP
B	Junior	MP
C	Senior	BP
D	Junior	MP
E		?

1. Calcular as probabilidades de *E* se enquadrar em cada uma das duas classes:

$$P(E = MP) \propto \frac{\#MP}{\#clientes} = 2/4$$

$$P(E = BP) \propto \frac{\#BP}{\#clientes} = 2/4$$

NAÏVE BAYES: CONSTRUÇÃO

- **Inserindo um novo fato (verossimilhança):** Idade - *E* é Sênior.

Id	Fx. Etária	Pagador
A	Senior	BP
B	Junior	MP
C	Senior	BP
D	Junior	MP
E	**Senior**	?

2. Calcular as probabilidades de *E* se enquadrar em cada uma das duas classes, sabendo que *E* é Sênior:

$$P(E = MP | idade = Sênior) \propto ?$$

$$P(E = BP | idade = Sênior) \propto ?$$

NAÏVE BAYES: CONSTRUÇÃO

3. Aplicando a Regra de Bayes, sabemos que basta multiplicar a verossimilhança pela probabilidade a priori para calcular a probabilidade a posteriori. Assim:

- $$\underbrace{P(E = MP|Idade = Senior)}_{\text{Posteriori}} \propto \underbrace{\frac{\#Senior\ e\ MP}{\#MP}}_{FatoNovo} * \underbrace{\frac{\#MP}{\#Clientes}}_{Priori} = \frac{0}{2} * \frac{2}{4} = 0$$

- $$\underbrace{P(E = BP|Idade = Senior)}_{\text{Posteriori}} \propto \underbrace{\frac{\#Senior\ e\ BP}{\#BP}}_{FatoNovo} * \underbrace{\frac{\#BP}{\#Clientes}}_{Priori} = \frac{2}{2} * \frac{2}{4} = 1/2$$

NAÏVE BAYES: CONSTRUÇÃO

4. Vamos fazer uma normalização das probabilidades, para que as duas somadas resultem em 1. Ao se normalizar, têm-se:

$$P(E = MP | idade = Sênior) \propto \frac{0}{\frac{1}{2} + 0} = 0$$

$$P(E = BP | idade = Sênior) \propto \frac{1/2}{\frac{1}{2} + 0} = 1$$

NAÏVE BAYES: CONSTRUÇÃO

Decisão:

- Como P(E = BP|Idade = Senior) > P(E = MP|Idade = Senior), então, o Cliente *E* **tem maiores chances de pertencer ao grupo BP.**

Id	Fx. Etária	Pagador
A	Senior	BP
B	Junior	MP
C	Senior	BP
D	Junior	MP
E	**Senior**	**BP**

NAÏVE BAYES: CONSTRUÇÃO

- Vamos então usar a Regra de Bayes para calcular a probabilidade do cliente João (do problema original), com 38 anos, pertencer às classes *BP* e *MP*:

$$P(MP|idade = 38) \propto P(idade = 38|MP) * P(MP)$$

$$P(MP|idade = 38) \propto 0{,}035 * 0{,}47 = 0{,}016$$

$$P(BP|idade = 38) \propto P(idade = 38|BP) * P(BP)$$

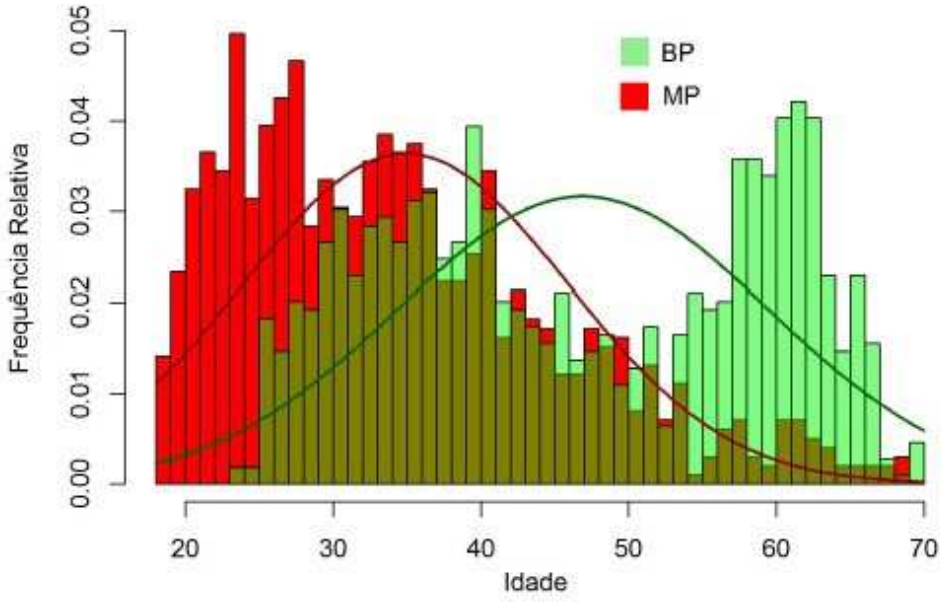
$$P(BP|idade = 38) \propto 0{,}025 * 0{,}53 = 0{,}013$$

NAÏVE BAYES: CONSTRUÇÃO

- **Pendências Finais:**
  - **1.** Como definir uma função densidade de probabilidade para cada grupo?
  - **2.** Como lidar com mais informação além da Idade, por exemplo, renda, estado civil, etc.?

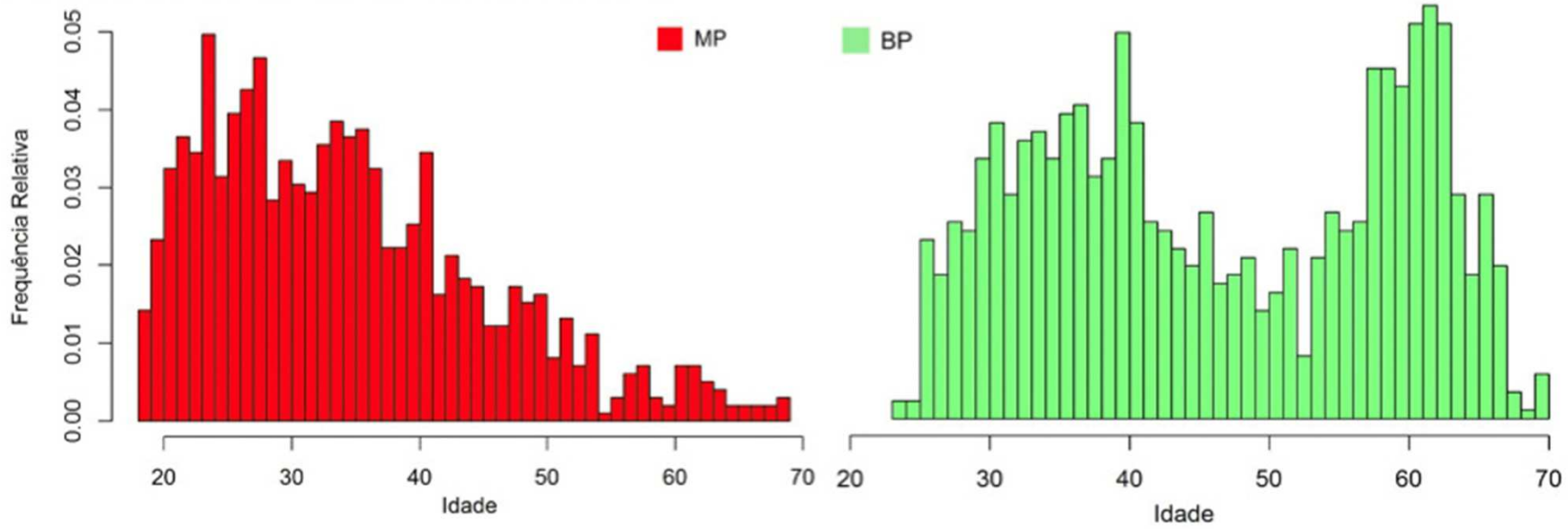
NAÏVE BAYES: CONSTRUÇÃO

- **1.** Como definir uma função densidade de probabilidade para cada grupo?



NAÏVE BAYES: CONSTRUÇÃO

- **1º Passo:** Vamos pensar um grupo por vez:



NAÏVE BAYES: CONSTRUÇÃO

- **2º Passo:** Escolher um tipo de distribuição que mais se aproxime dos dados. Usualmente, opta-se pela distribuição **normal**:

Matematicamente:

$$f_x(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-\mu)^2}{\sigma^2}}$$

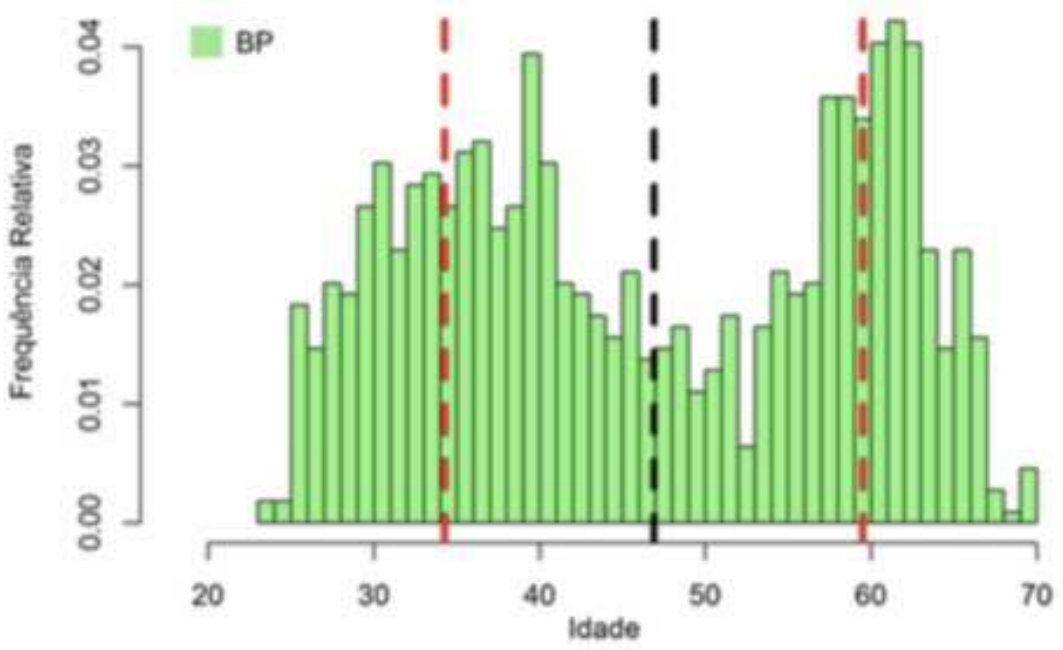
Parâmetros:

média (μ) e variância (σ^2).

NAÏVE BAYES: CONSTRUÇÃO

Dist. Normal para Idade e BP

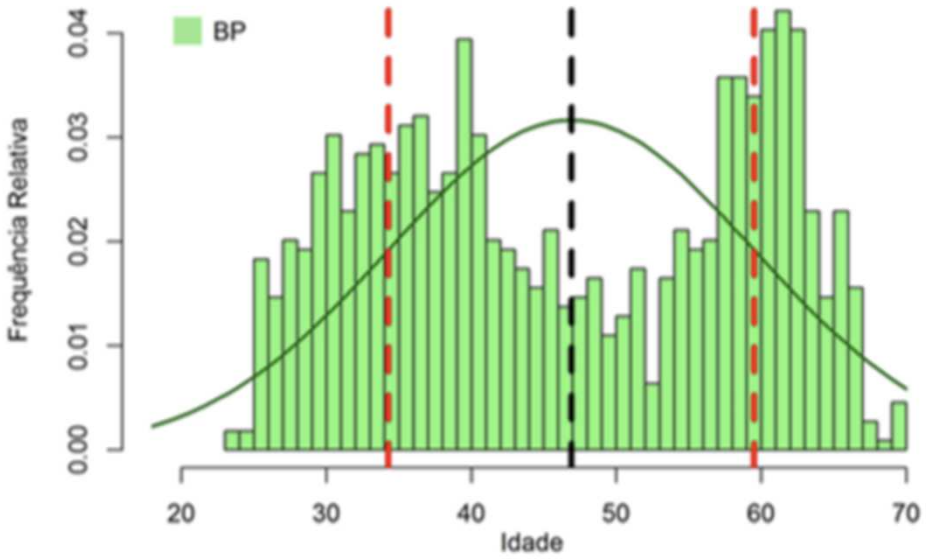
- **1º Passo:** Calcular a Média e Variância da Amostra

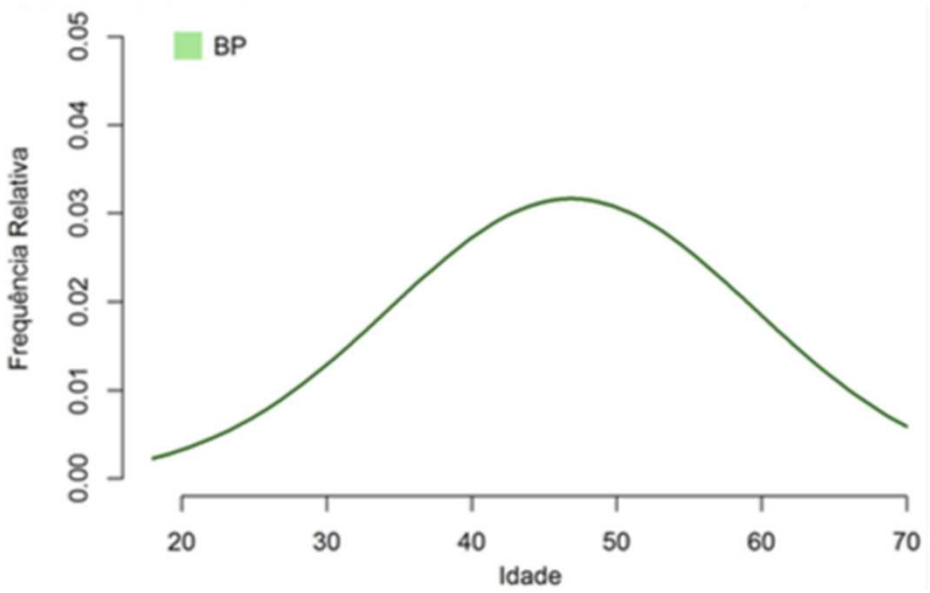


NAÏVE BAYES: CONSTRUÇÃO

Dist. Normal para Idade e BP

- **2º Passo:** Parametrizar a função Normal com estes valores





NAÏVE BAYES: CONSTRUÇÃO

- **Pendências Finais:**
  - **1.** Como definir uma função densidade de probabilidade para cada grupo?
  - **2. Como lidar com mais informação além da Idade, por exemplo, Renda, Valor da Entrada, etc.?**

NAÏVE BAYES: CONSTRUÇÃO

- **Problema Original:** Definir a Liberação de um Empréstimo Pessoal

- **Vontade:** usar as demais informações na Regra de Bayes

Cliente	Renda	Idade	Est. Civil	Pagador
Ana	360	25	C	MP
Bernardo	350	33	S	MP
Carlos	1100	56	S	MP
...	...	...	...	...
Xavier	600	33	S	MP
Zé	800	29	S	BP
João	**750**	**38**	**C**	**?**

NAÏVE BAYES: CONSTRUÇÃO

- Lembrando o que já temos

$$f(MP) = 0{,}047$$
$$f(BP) = 0{,}053$$
$$f(idade = 38|MP) \cong 0{,}035$$
$$f(idade = 38|BP) \cong 0{,}025$$
$$f(MP|idade = 38) = 0{,}016$$
$$f(BP|idade = 38) = 0{,}013$$

- Agora queremos:

$$f(MP|idade = 38\ e\ renda = 750) = ?$$
$$f(BP|idade = 38\ e\ renda = 750) = ?$$

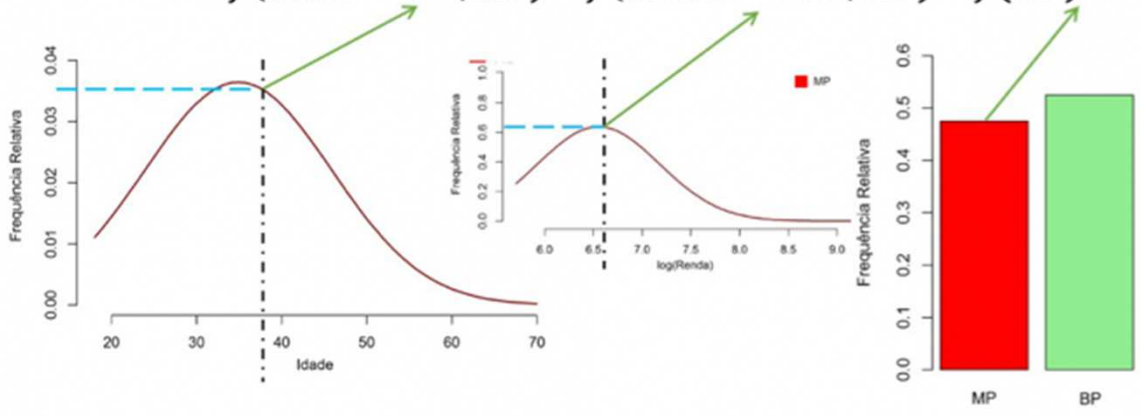
NAÏVE BAYES: CONSTRUÇÃO

- Para calcular as respostas, basta fazer o cálculo de maneira desacoplada:

$$P(MP|idade = 38\ e\ renda = 750) \propto P(idade = 38\ e\ renda = 750|MP) * P(MP) \approx$$

$$\approx P(idade = 38|MP) * P(renda = 750|MP) * P(MP)$$

$$P(MP \mid Idade = 38, Renda = 750)$$
$$\propto f(Idade = 38 \mid MP) * f(Renda = 750 \mid MP) * f(MP)$$



NAÏVE BAYES: CONSTRUÇÃO

- Aplicando o Naive Bayes (após uma normalização):

Cliente	Renda	Idade	Pagador	P(Cliente=MP)	Decisão
Ana	360	25	MP	0,6	MP
Bernardo	350	33	MP	0,3	BP
Carlos	1100	56	MP	0,6	MP
...	...	...	...	...	...
Xavier	600	33	MP	0,6	MP
Zé	800	29	BP	0,1	BP
João	**750**	**38**	**?**	**0,3**	**BP**

NAÏVE BAYES: DEFINIÇÃO FORMAL

Sejam:

- $X(A_1, A_2, \ldots, A_n, C)$ um conjunto de dados
- $c_1, c_2, \ldots, c_k$ são as **classes** do problema (valores possíveis do atributo alvo C)
- R um **registro** que deve ser classificado
- $a_1, a_2, \ldots, a_k$ os valores que R assume para os **atributos** previsores $A_1, A_2, \ldots, A_n$, respectivamente.

NAÏVE BAYES: DEFINIÇÃO FORMAL

O algoritmo consiste em:

1. Calcular as **probabilidades condicionais** $P(C=c_i|R)$, $i = 1, 2, \ldots, k$
2. Indicar como **saída** do algoritmo a classe c tal que $P(C=c|R)$ seja **máxima**, quando considerados todos os valores possíveis do atributo alvo C.

- A intuição por trás do algoritmo é dar **mais peso** para as classes **mais frequentes**.

Support Vector Machines (SVM)

SUPPORT VECTOR MACHINES (MÁQUINAS DE VETOR DE SUPORTE)

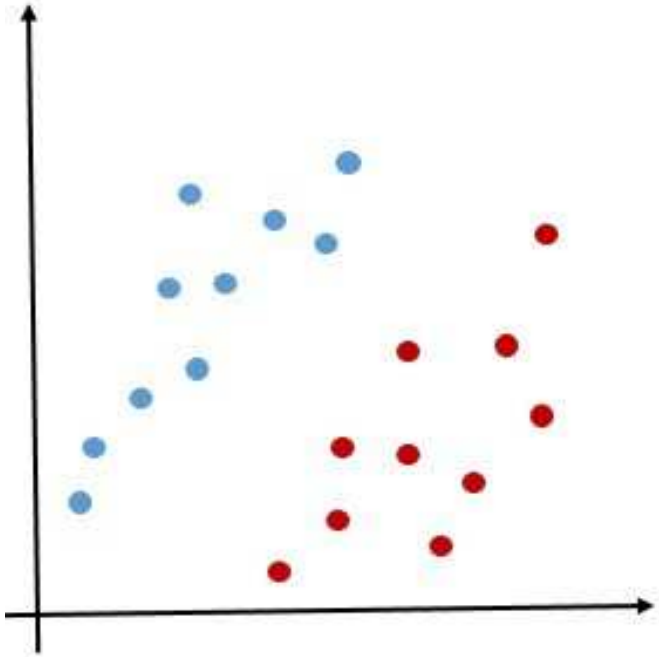
- Um dos algoritmos mais **efetivos** para Classificação, e também pode ser utilizado para Regressão (menos usado), podendo ser usado em dados **lineares** ou **não lineares**.
- O primeiro *paper* foi publicado em 1992, mas conceitos teóricos são explorados desde 1960.
- Apesar do **treinamento** dos modelos de SVM costumar ser **lento**, costumam apresentar **boa acurácia** e conseguem modelar fronteiras de decisão **complexas** e **não-lineares**.
  - Não é adequado para conjuntos de dados muito grandes ou com grande quantidade de ruídos.
- **Menos** propensos a ***overfitting*** se comparados com outros métodos.
- **Aplicações:** reconhecimento de manuscritos, reconhecimento de voz, *text mining*, etc.

SVM: RESUMO

- Essencialmente, o SVM realiza um mapeamento **não linear** para **transformar** os dados de treino originais em uma **dimensão maior**.
- Nesta nova dimensão, o algoritmo busca pelo **hiperplano** que separa os dados linearmente de forma ótima.
  - Com um mapeamento apropriado para uma dimensão suficientemente alta, dados de duas classes podem ser **sempre** separados por um hiperplano.
- O SVM encontra este hiperplano usando **vetores de suporte** (exemplos essenciais para o treinamento) e **margens**, definidas pelos vetores de suporte.

SVM: EXEMPLO

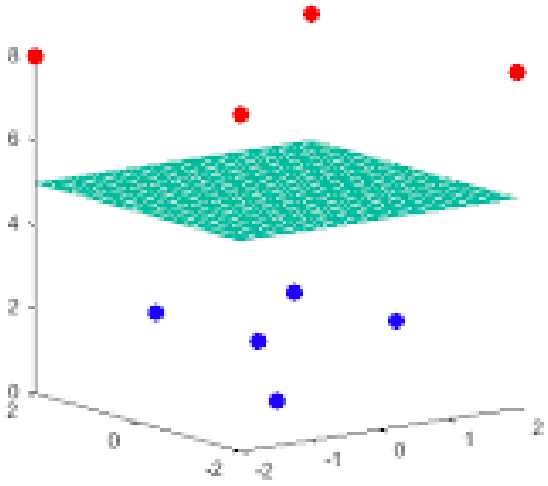
- Considere um conjunto de dados de treinamento **linearmente separável** na forma $\{x_i, y_i\}$, em que:
  - x_i corresponde ao vetor de 2 atributos previsores
  - $y_i \in \{-1,1\}$, às duas classes possíveis do problema.
- O conjunto de dados de entrada é utilizado para construir uma função de decisão $f(x)$ tal que:
  - Se $f(x) > 0$, então $y_i = 1$
  - Se $f(x) < 0$, então $y_i = -1$



SVM: EXEMPLO

- O SVM constrói **classificadores lineares**, que separam os conjuntos de dados por meio de um **hiperplano** (generalização do conceito de plano para dimensões > 3).

- Em um espaço *p*-dimensional, um **hiperplano** é um subespaço achatado de dimensão ***p*-1**, que não precisa passar pela origem.



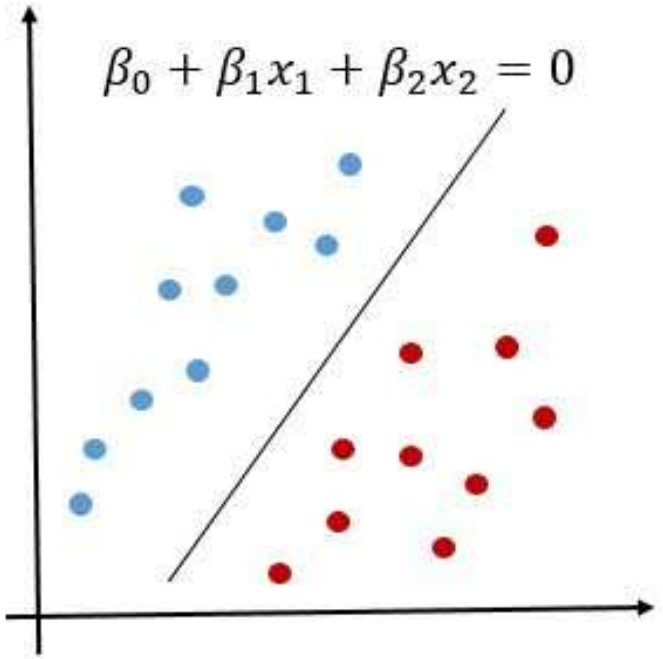
SVM: EXEMPLO

Para **d = 2**:

- O hiperplano é uma **reta** e sua equação é $\beta_0 + \beta_1 x_1 + \beta_2 x_2 = 0$,

(β_1 e β_2 são parâmetros que determinam a inclinação da reta e β_o o ponto de corte do eixo y)

- O **propósito** do SVM é determinar parâmetros da reta (β_0, β_1 e β_2), que permitem **separar** os conjuntos dos dados de treinamento em duas classes possíveis:
  - $\beta_0 + \beta_1 x_1 + \beta_2 x_2 < 0$
  - $\beta_0 + \beta_1 x_1 + \beta_2 x_2 > 0$



SVM: EXEMPLO

- Da mesma forma que um plano tem **2** dimensões e divide um conjunto tridimensional em 2 espaços de dimensão **3**, um hiperplano em um espaço **n**-dimensional tem **n-1** dimensões e divide este espaço em 2 subespaços de dimensão **n**.

Para **d = p**:

- A equação do hiperplano é

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ ... + \beta_n x_n = 0$$

- Neste caso, o hiperplano também divide o espaço p-dimensional em 2 metades:
  - $\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ ... + \beta_n x_n < 0$
  - $\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ ... + \beta_n x_n > 0$

SVM: EXEMPLO

Voltando ao nosso exemplo…

- Como afirmamos que o problema é linearmente separável, **é possível** construir um hiperplano que separe as observações de treino perfeitamente de acordo com seus rótulos de classe.

- Este hiperplano tem as seguintes propriedades:
  - $\beta_0 + \beta_1 x_1 + \beta_2 x_2 > 0, \text{se } y_i = 1$
  - $\beta_0 + \beta_1 x_1 + \beta_2 x_2 < 0, \text{se } y_i = -1$

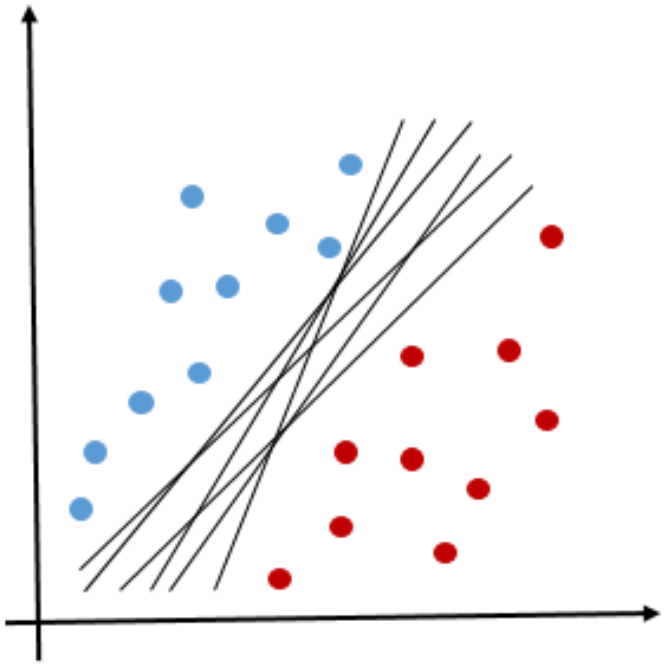
- O hiperplano pode então ser usado para construir um classificador: o exemplo de teste recebe a classe dependendo de que lado do hiperplano estiver localizado.

SVM: EXEMPLO

- Porém, no exemplo, **infinitas** retas (ou hiperplanos) dividem corretamente o conjunto de treinamento em duas classes.

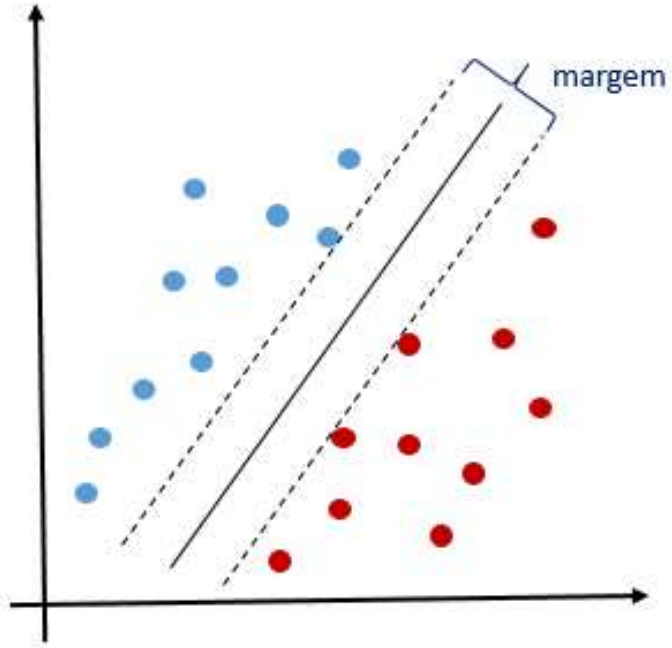
- **É preciso definir qual delas usar…**

- O SVM deve realizar um processo de **escolha** da reta separadora, dentre o conjunto infinito de retas possíveis.



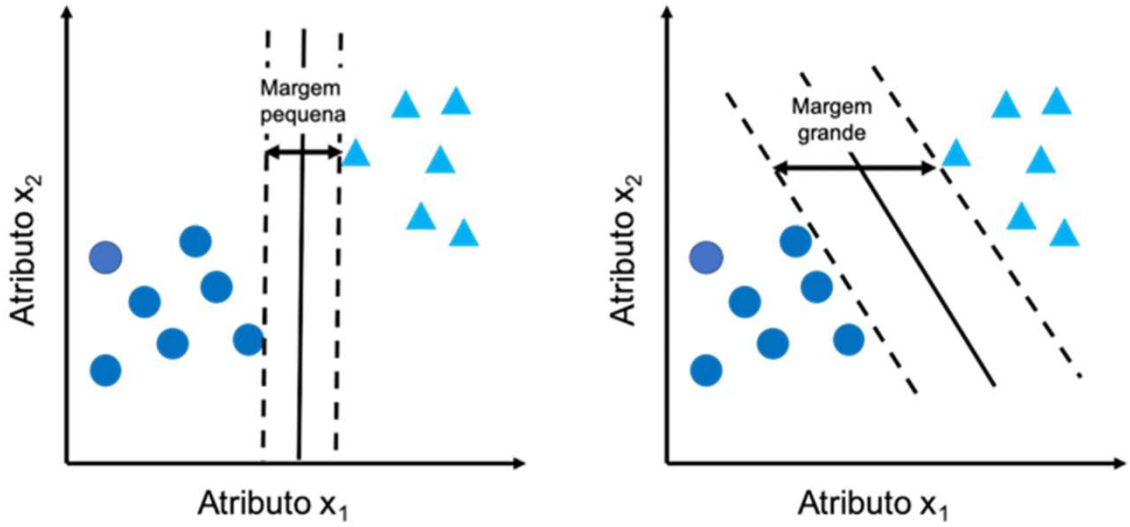
SVM: MARGEM E VETOR DE SUPORTE

- Na figura, é apresentado apenas um **Classificador Linear** (reta sólida) e duas retas paralelas ao classificador.
- Cada uma delas é movida a partir da posição da reta sólida, e determina quando a reta paralela intercepta o **primeiro ponto** do conjunto de dados.
- A **margem** é a distância construída entre estas duas retas paralelas pontilhadas.



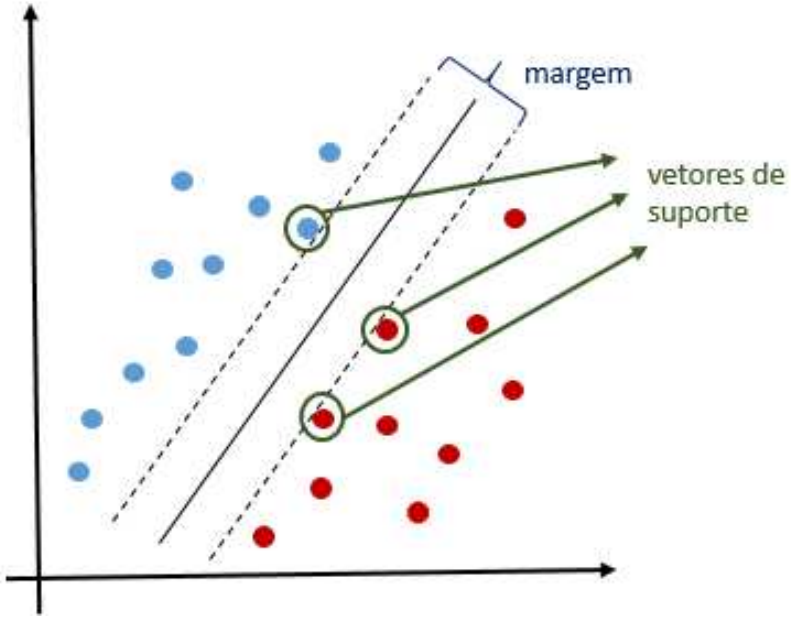
SVM: MARGEM E VETOR DE SUPORTE

- Assim como existem infinitas retas que separam os pontos em duas classes, há diversos **tamanhos de margem possíveis** dependendo da reta escolhida como classificador.



SVM: MARGEM E VETOR DE SUPORTE

- O classificador associado ao valor máximo de margem é denominado **Classificador Linear de Margem Máxima**.
- Os pontos do conjunto de dados de treinamento que são interceptados pelas linhas da margem são denominados **vetores de suporte** e são os pontos mais difíceis de classificar.
- OBS: apesar deste classificador ser geralmente bom, pode haver *overfitting* se o número de dimensões for grande.



SVM: OTIMIZAÇÃO

- O SVM realiza um processo de **otimização**, por meio do qual são determinados os parâmetros do classificador linear (no exemplo, β_0, β_1 e β_2).

- O objetivo é determinar os valores dos parâmetros que produzam o **valor máximo** para o comprimento da margem.

SVM: OTIMIZAÇÃO

- A reta correspondente ao classificador linear é dita ótima porque se ela for deslocada em alguma das duas direções das retas perpendiculares a ele, a probabilidade é menor de haver um erro de classificação.

- A posição do classificador linear correspondente ao comprimento de margem máximo é a **mais segura possível** com relação a eventuais erros de classificação: quanto maior a distância de *x* para o hiperplano, maior a confiança sobre a classe a que *x* pertence.

SVM: OTIMIZAÇÃO

- A solução do problema de otimização do SVM pode ser obtida usando técnicas de **programação quadrática**, muito conhecidas na área de Pesquisa Operacional, que consistem em **otimizar** uma função quadrática sujeita a restrições lineares. A solução deste problema está fora do escopo deste curso.

- Uma vez obtidos os valores dos parâmetros, a função de decisão para classificar um novo exemplo x_i é dada pelo sinal de f(x):

$$Se\ f(x_i) > 0, então\ y_i = 1$$

$$Se\ f(x_i) < 0, então\ y_i = -1$$

SVM: SOFT-MARGIN

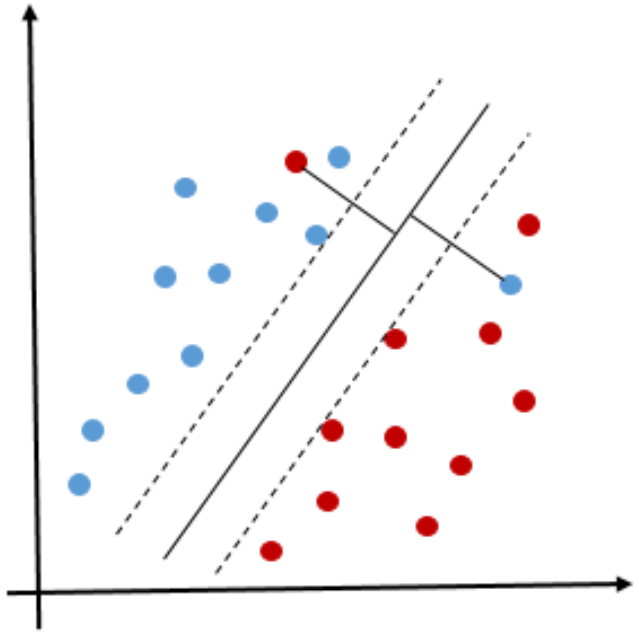
- Na prática, os dados reais não costumam ser perfeitamente separáveis por um hiperplano.
- Além disso, o Hiperplano Margem Máxima é extremamente sensível a mudanças em uma única observação, o que sugere que pode ocorrer *overfitting* nos dados de treino.
- Podemos querer considerar um classificador baseado em um hiperplano que não separe perfeitamente as duas classes, com o objetivo de aumentar a robustez nas observações individuais e melhorar a classificação na maioria das observações de treino.
  - Pode valer a pena classificar erroneamente algumas observações de treino a fim de melhorar a classificação nas observações restantes.

SVM: SOFT-MARGIN

- O **classificador soft-margin** permite que algumas observações do conjunto de treino violem a linha de separação:
  - O hiperplano é escolhido para separar corretamente **a maior parte** das observações em duas classes, mas pode classificar incorretamente algumas observações.
  - Um conjunto adicional de coeficientes é introduzido na otimização para permitir uma “folga” à margem, mas aumentam a complexidade do modelo.

SVM: SOFT-MARGIN

- Neste exemplo, o classificador iria cometer **erros** na classificação para os dois pontos destacados, que podem ser considerados ruído.



SVM: SOFT-MARGIN

- O parâmetro de custo **C** define a “rigidez” da margem e controla o *trade-off* entre o **tamanho da margem** e o **erro do classificador**.
- Quanto maior o valor de C, mais pontos podem ficar dentro da margem e maior o erro de classificação, mas menor é a chance de *overfitting*.
- Se **C = 0**, a margem é rígida e temos o **Classificador de Margem Máxima**, que na prática, não produz bons resultados.

SVM: NÚMERO DE VETORES DE SUPORTE

- O **número total de vetores de suporte** depende da quantidade de **folga** permitida nas margens e da **distribuição** dos dados.
  - Quanto **maior** a folga permitida, **maior** o número de vetores de suporte e **mais lenta** será a classificação dos dados de teste, pois a complexidade computacional do SVM está relacionada com o número de vetores de suporte.

SVM: KERNEL TRICK

- Na prática, o SVM é implementado usando funções **kernel**: objetos matemáticos que permitem que trabalhemos um espaço de dimensão maior.
  - Em vez de se utilizar as observações em si, é utilizado o seu produto interno. A previsão de um novo exemplo é feita calculando o produto escalar entre o exemplo (x) e cada vetor de suporte (x_i).

- Os tipos de kernel mais utilizados são linear, polinomial e radial (RBF).

$$K(x, x_i) = \sum(x \times x_i)$$

Linear

$$K(x, x_i) = 1 + \sum(x \times x_i)^d$$

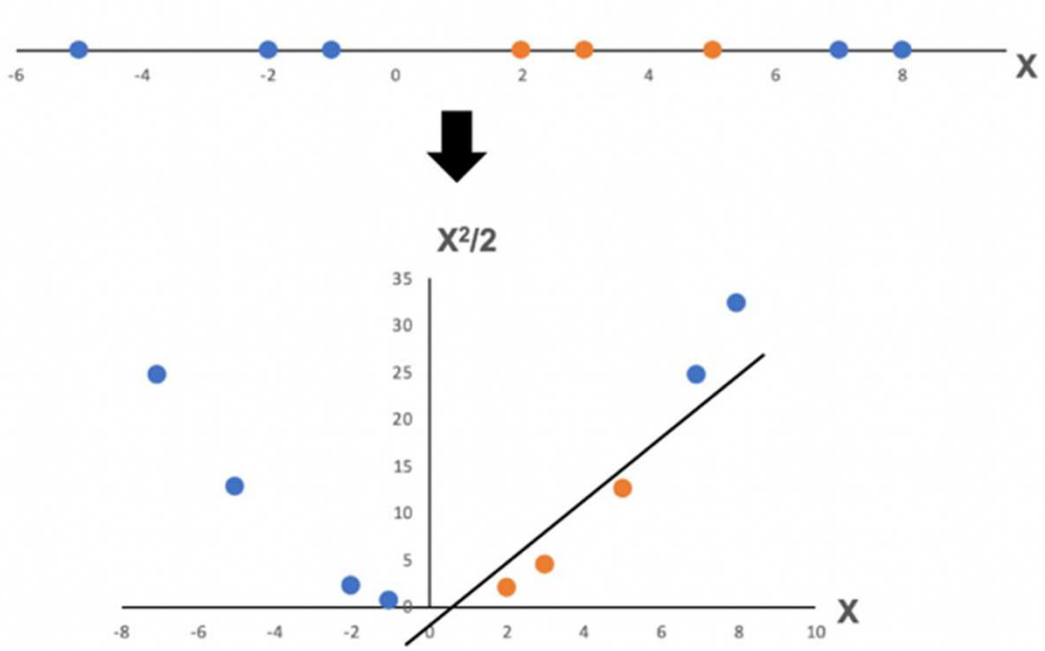
Polinomial

$$K(x, x_i) = e^{-gamma \times \sum((x - x_i)^2)}$$

Radial

SVM: KERNEL TRICK

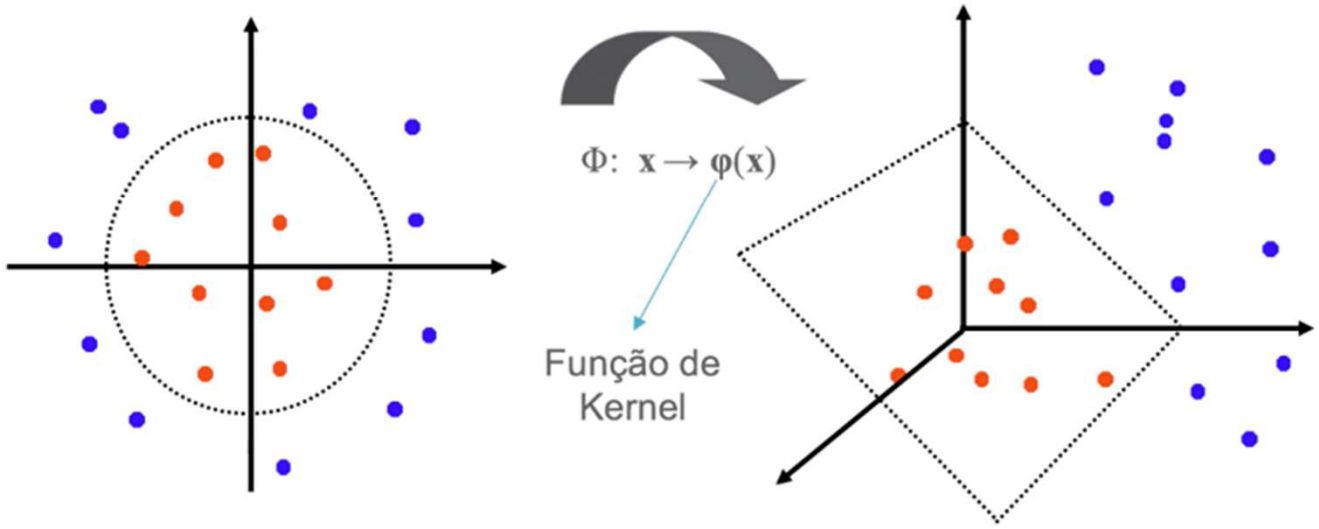
- Para um conjunto de dados que não é linearmente separável, o SVM utiliza **funções kernel** para mapear o conjunto de dados para um espaço de dimensão **maior** que a original e o classificador é ajustado neste novo espaço.
- O SVM é, na verdade, a combinação do **classificador linear** com um **kernel não-linear**.



SVM: KERNEL TRICK

- Embora estejamos aumentando a dimensão do espaço, a **complexidade diminui,** pois antes a classificação só era possível usando classificadores não lineares, e agora pode ser feita apenas com um hiperplano, que é uma superfície de decisão linear.

$$\phi(x_1, x_2) \rightarrow (z_1, z_2, z_3) = (x_1^2, \sqrt{2x_1x_2}, x_2^2)$$



Fonte: *Data Science Academy*

SVM: MÚLTIPLAS CLASSES

- O SVM também pode ser aplicado a problemas de classificação que envolvem **múltiplas classes**.
  - Usa-se o algoritmo para treinar um modelo de classificação que informe se o registro é da classe $\boldsymbol{c_i}$ (região positiva) ou se é de alguma **outra** classe diferente de c_i (região negativa).
  - Assim, constrói-se *p-1* modelos de classificação, onde *p* é o número de classes possíveis no problema.
  - O novo exemplo é classificado em cada um dos modelos gerados. A classificação final é a classe mais frequentemente atribuída.

SVM: RESUMO

- Dado um conjunto de treinamento, deseja-se realizar uma estimativa da **margem de comprimento máximo**.
  - Os **vetores de suporte** são os pontos (do conjunto de dados) que delimitam a margem e determinam a localização do classificador linear: são os pontos mais próximos da reta do classificador linear.
  - O **comprimento da margem** é igual à distância entre as duas retas paralelas ao classificador linear e que forma a fronteira de classificação.
  - Por construção, todos os vetores de suporte possuem a **mesma distância** em relação a reta do classificador linear (a metade do comprimento da margem).