

# Combinação de Modelos

1001513 – Aprendizado de Máquina 2  
Turma A – 2023/2  
Prof. Murilo Naldi



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



# Agradecimentos

- Pessoas que colaboraram com a produção deste material: Diego Silva, Ricardo Cerri, Moacir Ponti
- Intel IA Academy

# Discussão

Conhecemos diversos algoritmos de AM, mas qual deles é o melhor?

- Cada algoritmo explora estratégias diferentes
- Como explorar essas diferenças?



## Discussão

Modelos múltiplos são preditores individuais que são combinados ou agregados de alguma maneira para (tentar) “fortalecer” uma predição única.



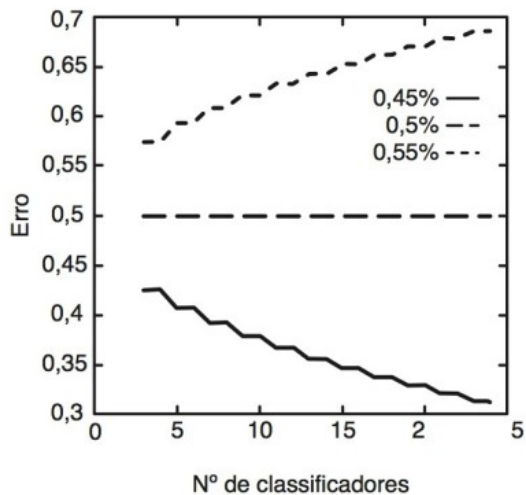
# Modelos múltiplos

## Hansen e Salamon (1990)

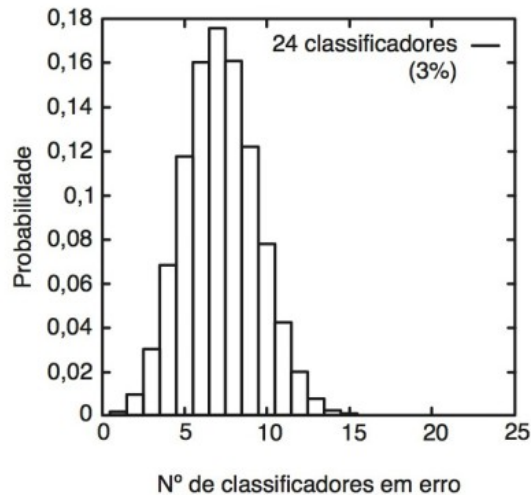
- Modelos múltiplos são bons apenas se os modelos individuais cometem erros independentes
- Os autores provam que a taxa de **erro** decresce linearmente com o número de modelos se:
  - Todos os modelos têm a mesma taxa de erro
  - A taxa de erro é menor que 0,5
  - Todos cometem erros independentes

# Modelos múltiplos

\*Cada um com 30% de erro



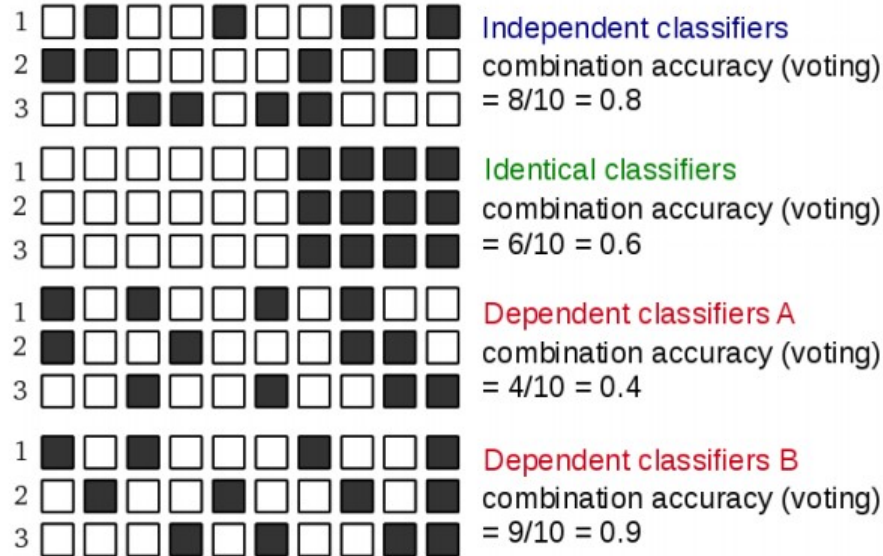
(a) Evolução da taxa de erro



(b) Probabilidade de erro

# Modelos múltiplos

[3 classifiers - accuracy = 6/10 = 0.6]    ☐ correct    ☒ incorrect



# Modelos múltiplos

## Conclusões “gerais”

- Os erros devem ser independentes
  - Ou, melhor ainda, negativamente correlacionados



# Modelos múltiplos

## Conclusões “gerais”

- Cada modelo deve ser melhor do que o modelo de escolhas aleatórias

# Modelos múltiplos

Duas questões centrais:

- Como **combinar** previsões de modelos diferentes?
- Como **gerar** modelos diferentes?



## Modelos múltiplos

Modelos **heterogêneos** vs. **Homogêneos**

- É preciso garantir diversidade
  - Mas como?

# Modelos múltiplos

- Sabemos que os erros tem que ser independentes
  - Como medir? Ex: *inter-rated agreement*

$$\kappa = \frac{2(ad-bc)}{(a+c)(c+d)+(a+b)(b+d)}$$

a = os dois acertaram

b = apenas o primeiro acertou

c = apenas o segundo acertou

d = os dois erraram

# Modelos múltiplos

- Podemos dividir os métodos de combinação em categorias:
  - Votação vs. seriação
  - Estáticos vs. dinâmicos

# Métodos de votação vs seriação

- Votação é o método mais comum
  - Votação simples/uniforme
  - Votação ponderada
    - Todos são independentes!

# Métodos de votação vs seriação

- Seriação considera os componentes em série
  - Possui “*scores*” na combinação
  - *Scores* tem que “se comportar como probabilidades”
  - Na maioria dos casos existe uma “ordem”

# Métodos de votação vs seriação

Regras para  $m$  modelos com  $K$  classes:

→ Soma

$$S_k = \sum_{i=1}^m P_{ik}$$

→ Média

$$S_k = \frac{1}{m} \sum_{i=1}^m P_{ik}$$

→ Média geométrica

$$S_k = \sqrt[m]{\prod_{i=1}^m P_{ik}}$$

KITTLER, J. et al. On combining classifiers. IEEE Trans. on Pattern Analysis and Machine Intelligence, v. 20, n. 3, p. 226-239, 1998



# Métodos de votação vs seriação

Regras para  $m$  modelos com  $K$  classes:

→ Produto

$$S_k = \prod_{i=1}^m P_{ik}$$

→ Máximo

$$S_k = \max_i P_{ik}$$

→ Mínimo

$$S_k = \min_i P_{ik}$$

KITTLER, J. et al. On combining classifiers. IEEE Trans. on Pattern Analysis and Machine Intelligence, v. 20, n. 3, p. 226-239, 1998

# Métodos dinâmicos vs estáticos

- Métodos **estáticos** consideram **todos os modelos** na combinação
  - A escolha não muda
  - Ou seja, é isso aí que a gente viu até agora

# Métodos dinâmicos vs estáticos

- Métodos **dinâmicos** realizam uma **seleção de modelos** para cada exemplo a ser classificado
  - Baseado na hipótese que modelos diferentes erram mais em determinadas regiões do espaço que outros
  - Objetivo: melhorar resultado

# Métodos dinâmicos

## Model Applicability Induction (**MAI**)

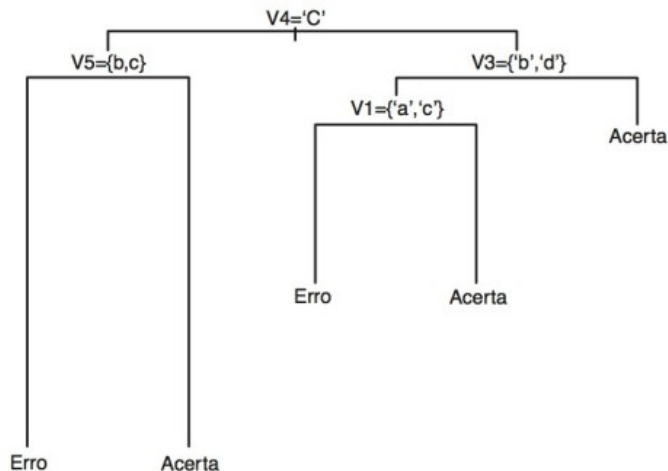
- Utiliza **meta-aprendizado** para determinar se utiliza ou não cada um dos modelos induzidos

V1	V2	V3	V4	V5	Classe	V1	V2	V3	V4	V5	Erro
t	a	c	t	a	membro	t	a	c	t	a	+
t	g	c	t	a	membro	t	g	c	t	a	-
g	t	a	c	t	não membro	g	t	a	c	t	+
a	a	t	t	g	membro	a	a	t	t	g	+
t	c	g	a	t	não membro	t	c	g	a	t	-
a	g	g	g	g	membro	a	g	g	g	g	+

# Métodos dinâmicos

## Model Applicability Induction (**MAI**)

- Utiliza **meta-aprendizado** para determinar se utiliza ou não cada um dos modelos induzidos (metamodelo)

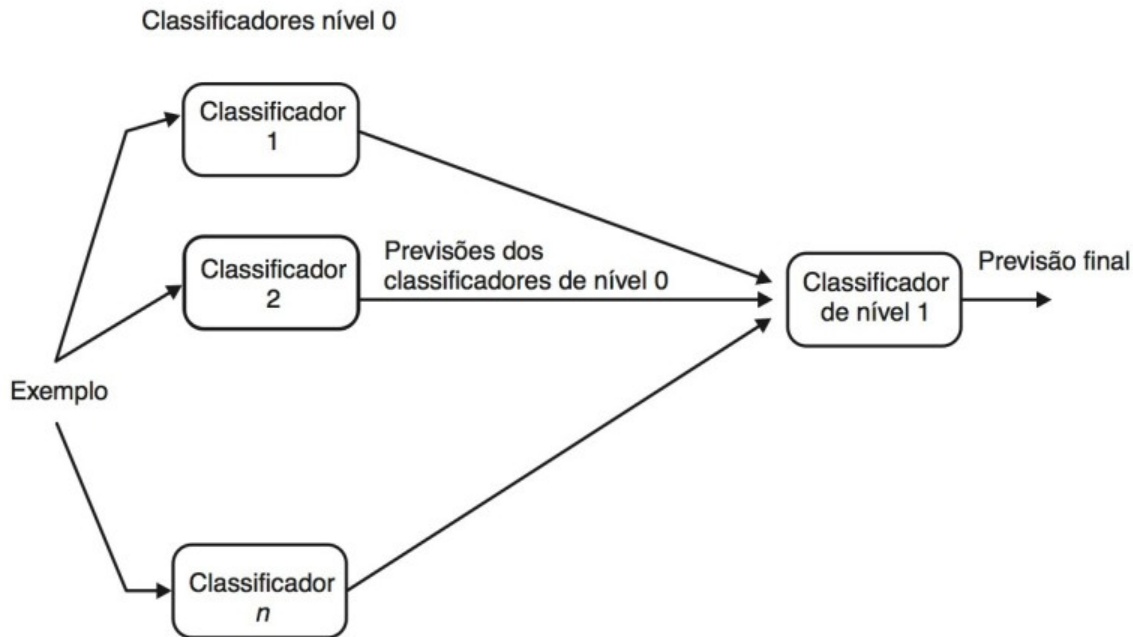


# Combinação de modelos heterogêneos

- Diferentes modelos são induzidos por diferentes algoritmos
- Há diversas formas de combinar resultados
  - Já vimos algumas
- Há maneiras mais “complexas”
  - Vamos ver um exemplo

# Combinação de modelos heterogêneos

## Generalização por pilha



# Combinação de modelos heterogêneos

## Generalização por pilha

V1	V2	V3	V4	V5	Classe	P <sub>1,1</sub>	P <sub>1,2</sub>	P <sub>2,1</sub>	P <sub>2,2</sub>	P <sub>3,1</sub>	P <sub>3,2</sub>	Classe
t	a	c	t	a	Membro	0,51	0,49	0,13	0,87	0,12	0,88	Membro
t	g	c	t	a	Membro	0,19	0,81	0,07	0,93	0,81	0,19	Membro
g	t	a	c	t	Não Membro	0,68	0,32	0,55	0,45	0,69	0,31	Não Membro
a	a	t	t	g	Membro	0,74	0,26	0,66	0,34	0,94	0,06	Membro
t	c	g	a	t	Não Membro	0,62	0,38	0,01	0,99	0,78	0,22	Não Membro
a	g	g	g	g	Membro	0,65	0,35	0,90	0,10	0,55	0,45	Membro

Conjunto de dados original

Conjunto de dados de  $Nível_1$

Score do modelo 3 para a classe 1



# Combinação de modelos homogêneos

- Há várias estratégias de combinar resultados de modelos gerados pelo mesmo algoritmo de aprendizado

Mas... cadê a diversidade?



# Combinação de modelos homogêneos

- Geralmente, isso é feito por manipulações no conjunto de treinamento
- Funciona bem para algoritmos instáveis
  - Aqueles que mudam bastante com pouca alteração nos dados (lembra algum?)

# Combinação de modelos homogêneos

Manipulações nos dados estão relacionadas ao conceito de **reamostragem**

- De exemplos, de atributos

Ou injeção de aleatoriedade e perturbação de exemplos

# Injeção de aleatoriedade

Injetar aleatoriedade é uma técnica comum em AM

- Ex: inicializar uma rede neural com diferentes pesos

Vamos ver outros exemplos

# Combinação de modelos homogêneos

## *Bootstrap aggregating* - **Bagging**

- Cria conjuntos de exemplos por amostragem com reposição
- O tamanho dos conjuntos de dados originados são o mesmo que do conjunto original

# Combinação de modelos homogêneos

## *Bootstrap aggregating* - **Bagging**

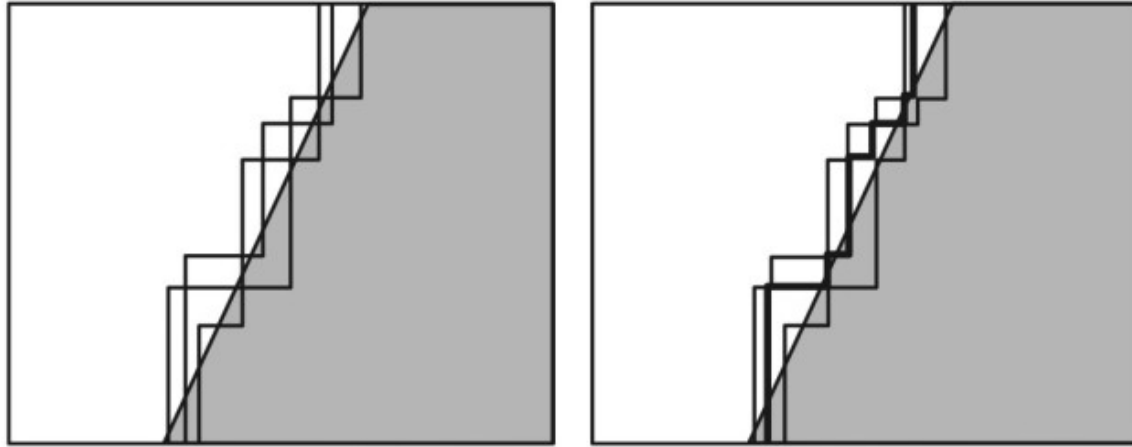
- Portanto, o novo conjunto pode conter repetições, enquanto alguns exemplos ficam “de fora”
  - Cada amostra, para  $n$  grande, tem aproximadamente 36,8% de duplicações
- Um classificador é gerado para cada amostragem

# Combinação de modelos homogêneos

## *Bootstrap aggregating - **Bagging***

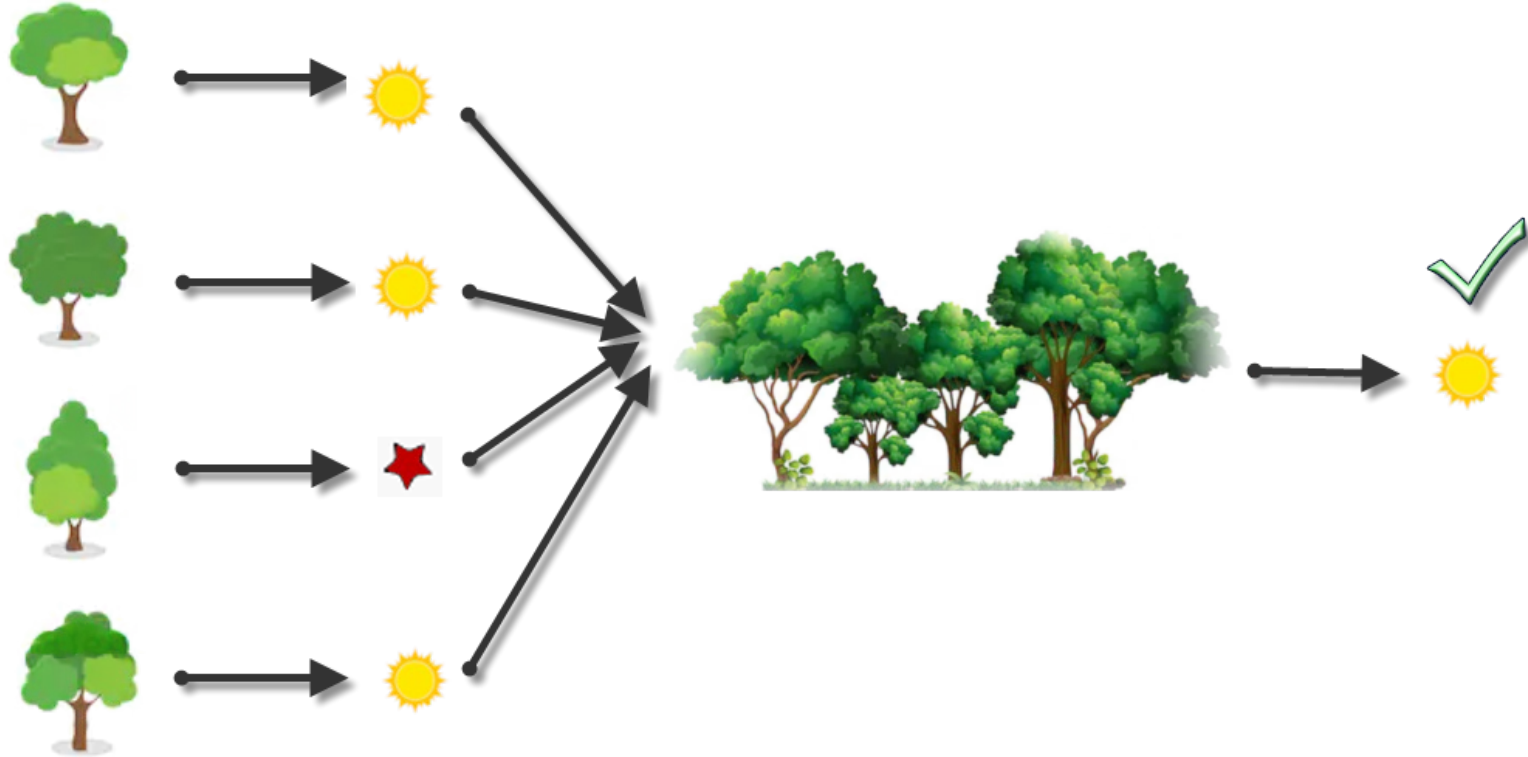
- Por que isso funciona?
  - Exemplo em árvores de decisão:
    - Pode mudar os atributos selecionados
    - Pode mudar os pontos de corte
    - Em cada mudança, pode afetar a distribuição dos dados nos filhos de cada nó de decisão

# Ilustração Bagging





# Random Forest



# Random Forest

- Considere um conjunto de dados  $D$ , com  $m$  atributos.
- Realizamos uma amostragem por *bagging*

# Random Forest

- Considere um conjunto de dados  $D$ , com  $m$  atributos.
- Realizamos uma amostragem por *bagging*
- Criamos uma árvore de decisão a partir desse conjunto

# Random Forest

- Considere um conjunto de dados  $D$ , com  $m$  atributos.
- Realizamos uma amostragem por *bagging*
- Criamos uma árvore de decisão a partir desse conjunto
- Mas, selecionamos  $i$  ( $i \ll m$ ) atributos aleatoriamente para determinar os nós de decisão em cada nível
  - Não utilizamos quaisquer estratégias de poda

# Random Forest

- Considere um conjunto de dados  $D$ , com  $m$  atributos.
- Realizamos uma amostragem por *bagging*
- Criamos uma árvore de decisão a partir desse conjunto
- Mas, selecionamos  $i$  ( $i \ll m$ ) atributos aleatoriamente para determinar os nós de decisão em cada nível
  - Não utilizamos quaisquer estratégias de poda
- Repetimos o processo  $T$  vezes (em que  $T$  é o número de árvores de decisão)

# Random Forest

id	Att1	Att2	Att3	Att4	Classe
1					
2					
3					
4					
5					

# Random Forest

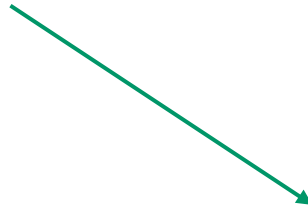


id	Att1	Att2	Att3	Att4	Classe
1					
2					
3					
4					
5					

# Random Forest



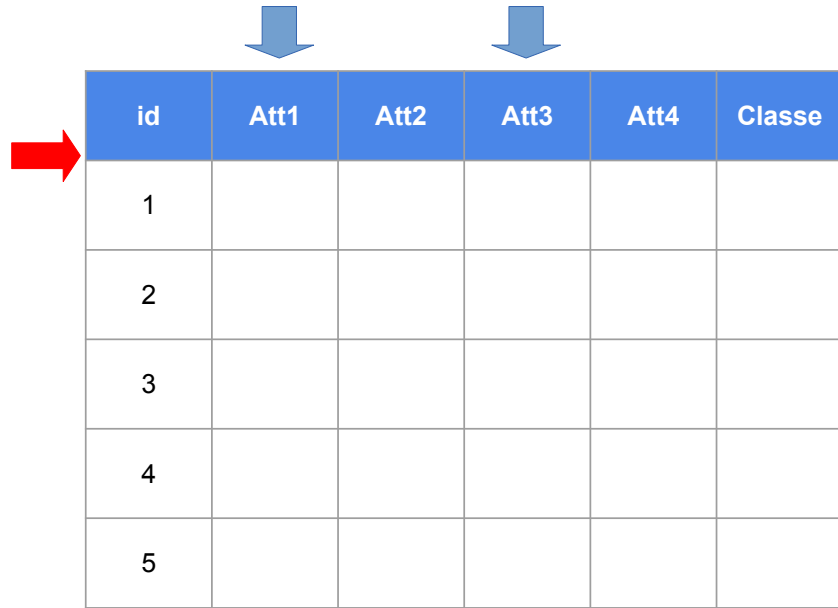
id	Att1	Att2	Att3	Att4	Classe
1					
2					
3					
4					
5					



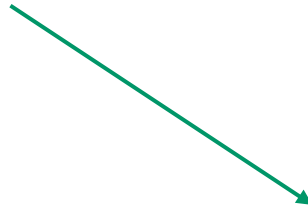
id	Att1	Att3	Classe



# Random Forest

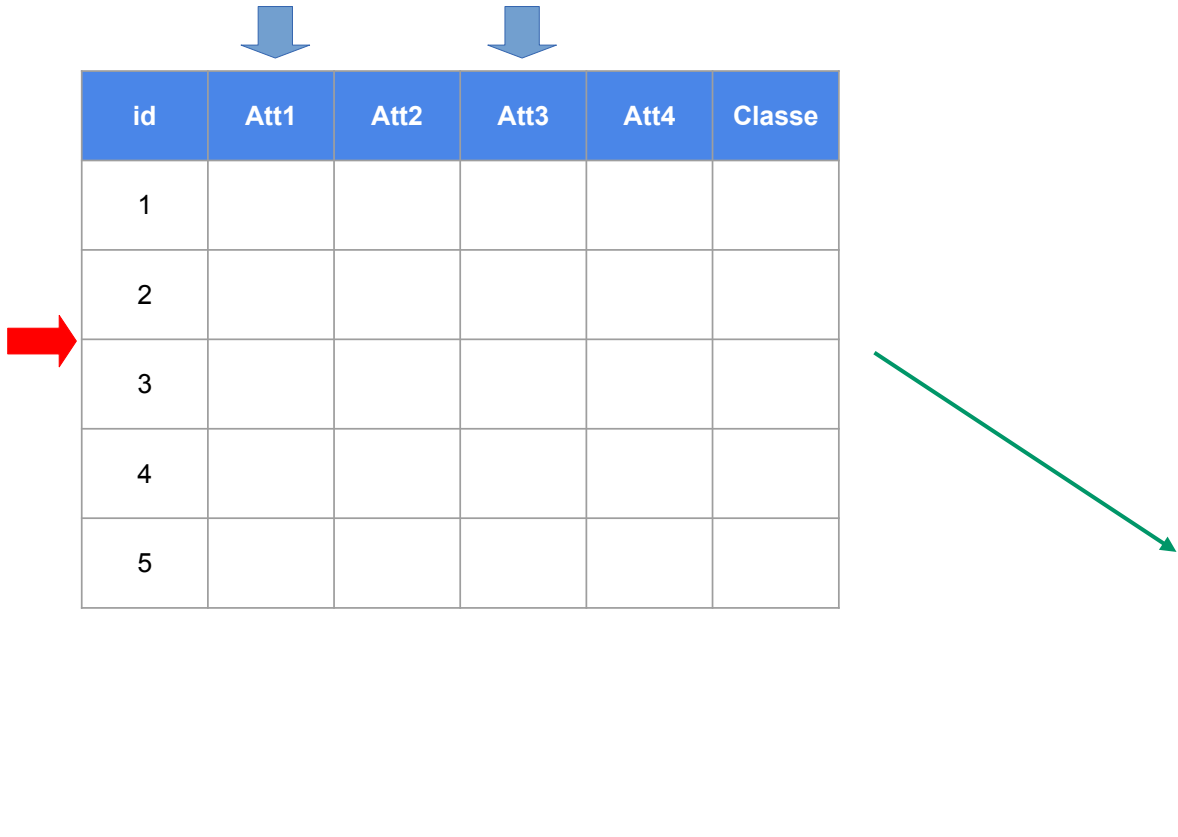


id	Att1	Att2	Att3	Att4	Classe
1					
2					
3					
4					
5					

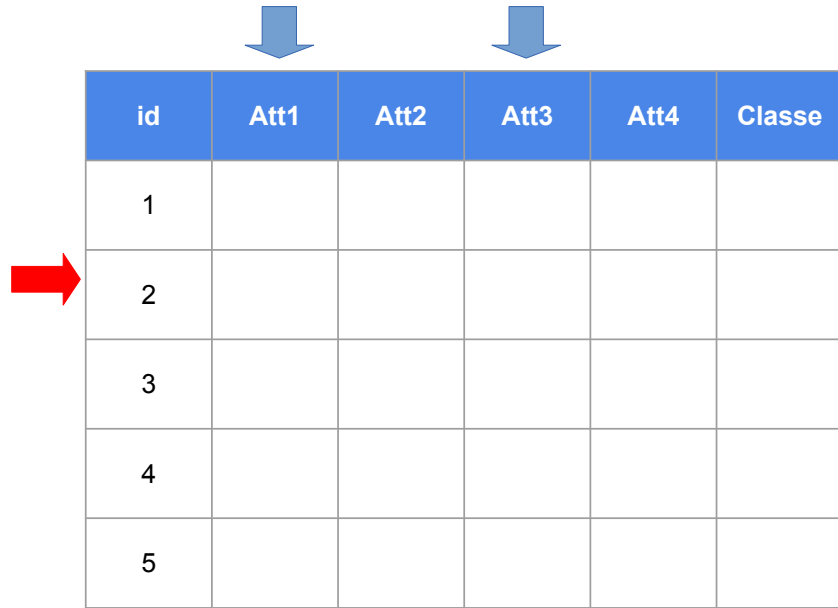


id	Att1	Att3	Classe
1			

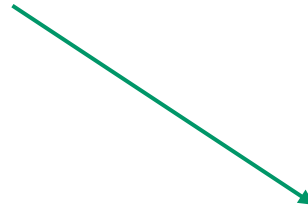
# Random Forest



# Random Forest

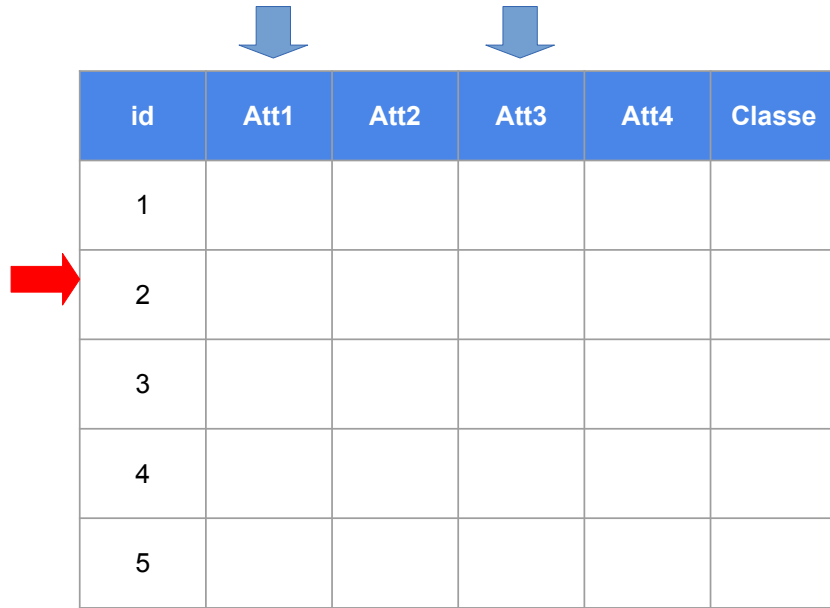


id	Att1	Att2	Att3	Att4	Classe
1					
2					
3					
4					
5					

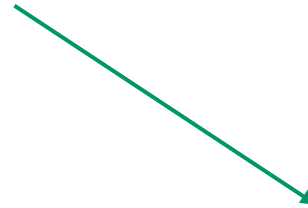


id	Att1	Att3	Classe
1			
4			
3			

# Random Forest

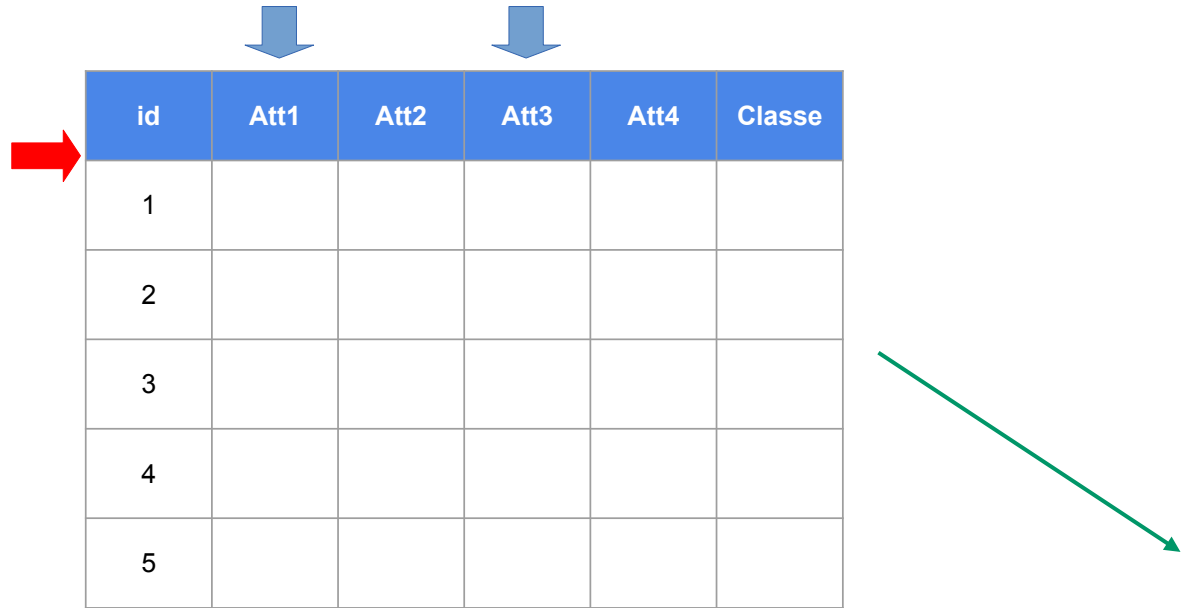


id	Att1	Att2	Att3	Att4	Classe
1					
2					
3					
4					
5					



id	Att1	Att3	Classe
1			
4			
3			
3			

# Random Forest

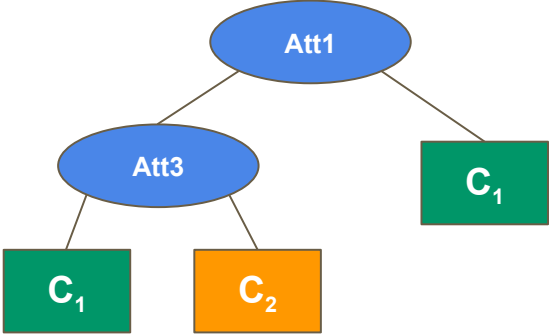


id	Att1	Att2	Att3	Att4	Classe
1					
2					
3					
4					
5					

id	Att1	Att3	Classe
1			
4			
3			
3			
1			

# Random Forest

id	Att1	Att2	Att3	Att4	Classe
1					
2					
3					
4					
5					



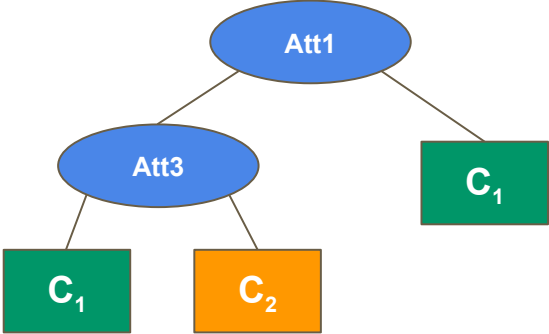
↑

id	Att1	Att3	Classe
1			
4			
3			
3			
1			

# Random Forest

id	Att1	Att2	Att3	Att4	Classe
1					
2					
3					
4					
5					

Out-of-bag

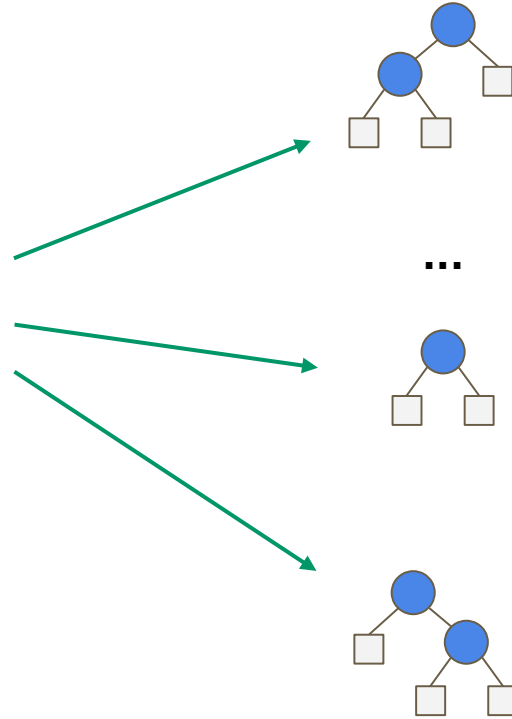


↑

id	Att1	Att3	Classe
1			
4			
3			
3			
1			

# Random Forest

id	Att1	Att2	Att3	Att4	Classe
1					
2					
3					
4					
5					





## Combinação de modelos homogêneos

- Sendo um classificador fraco um classificador cuja capacidade de generalização seja pouco melhor que a escolha aleatória
  - Poderia um conjunto de classificadores fracos ter a performance de um classificador forte?

## Combinação de modelos homogêneos

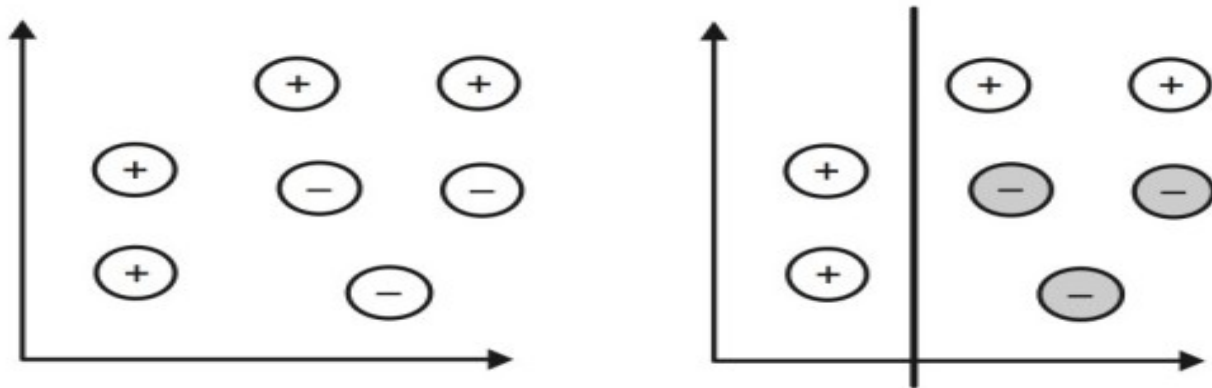
- Sendo um classificador fraco um classificador cuja capacidade de generalização seja pouco melhor que a escolha aleatória
  - Poderia um conjunto de classificadores fracos ter a performance de um classificador forte?
  - Sim, com *boosting*!

# Boosting

- Ao associar um peso adequado para cada exemplo de treinamento, o classificador pode alcançar uma acurácia relativamente elevada
- O algoritmo é iterativo:
  - A cada iteração o peso de cada exemplo é alterado
  - Se o exemplo for classificado corretamente, seu peso diminui segundo uma função
  - Se for classificado errado, seu peso aumenta
  - Constrói-se um novo classificador com esses pesos e repete-se o procedimento

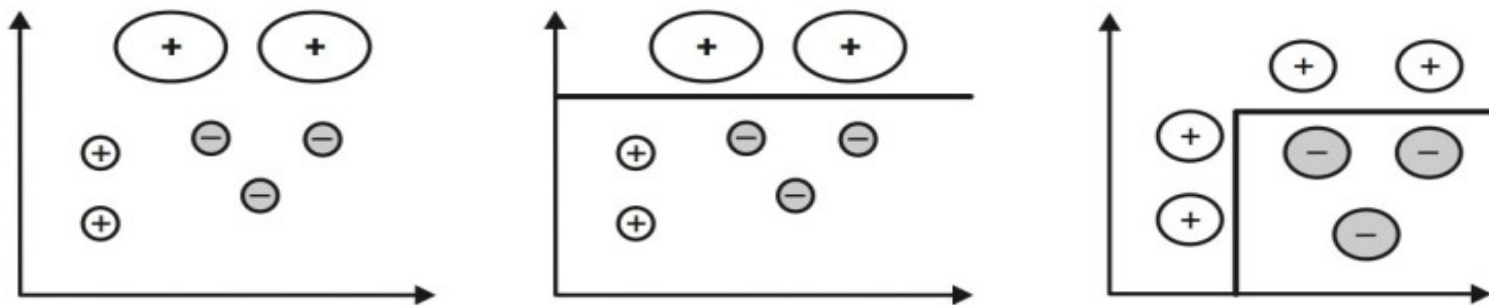
# Boosting

- A classificação é dada pela combinação de todos esses modelos aprendidos nas diferentes iterações
- Exemplo - iteração #2



# Boosting

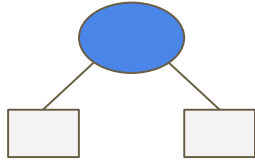
- A classificação é dada pela combinação de todos esses modelos aprendidos nas diferentes iterações
- Exemplo - iteração #2



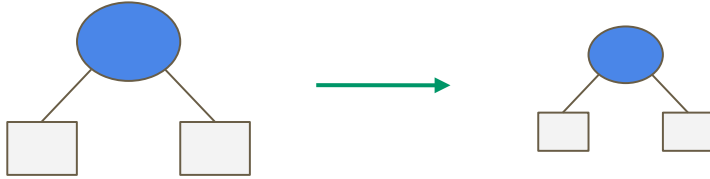
# AdaBoost

- Bom exemplo de *boosting*: AdaBoost
  - Baseado em amostragem aleatória
  - Cada exemplo é ponderado pela técnica de *boosting* descrita anteriormente para construir “*stumps*”
  - Os pesos influenciam na probabilidade de selecionar o exemplo e na construção da árvore
  - O erro dos *stumps* são utilizados para recalcular os pesos e ponderar a votação (na combinação)

# Exemplo

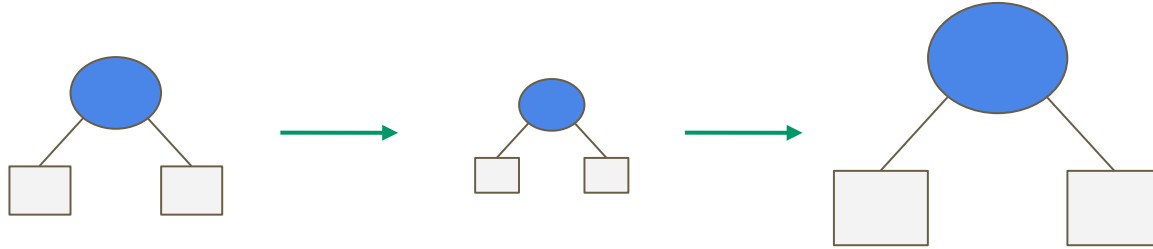


# Exemplo

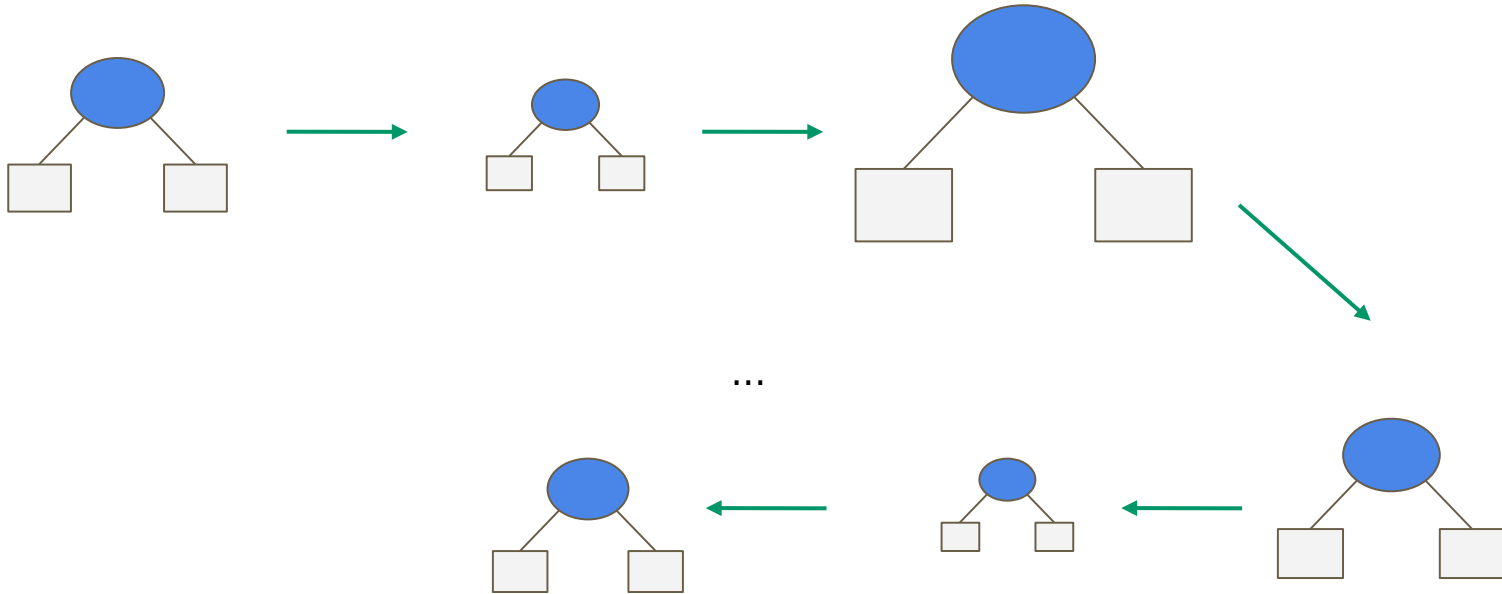




# Exemplo



# Exemplo



# *Gradient Boosting*

- Inicia-se com uma “folha de decisão”
- Ajusta os pesos, constrói uma árvore...
  - Muito parecido, mas a limitação é diferente
  - O cálculo do erro parece mais com redes neurais

# *T.Hanks*

- Em agradecimento ao Diego Silva, temos
  - Random Forrest

