SICO70
SISTEMAS INTELIGENTES 2

Aula 10 - Ensembles

Prof. Rafael G. **Mantovani**

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Apucarana - PR, Brasil

Universidade Tecnológica Federal do Paraná (UTFPR)
Engenharia de Computação

Roteiro

1. Introdução
2. *Voting*
3. *Boosting*
4. *Bagging*
5. *Random Forest*
6. Referências

Roteiro

1. **Introdução**
2. ***Voting***
3. ***Boosting***
4. ***Bagging***
5. ***Random Forest***
6. **Referências**

Introdução

- **Ensembles/Comitês**
  - Problema?
    - duas cabeças pensam melhor do que uma
    - mais cabeças ainda pensa melhor do que duas
    - decisão baseada em conjunto (comitês)
    - para ML os resultados são impressivos

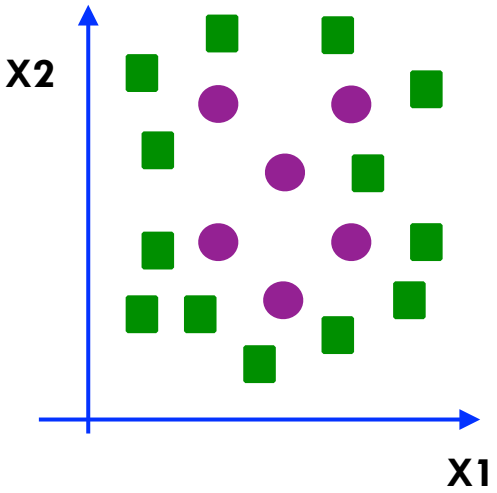
Introdução

- **Ensembles**
  - Random Forest (RF)
    - é um dos algoritmos de AM mais poderosos/robustos atualmente
    - funcionamento/implementação relativamente simples
  - *Ensembles* são soluções vencedoras em competições de AM
    - *Netflix Prize Competition*
      - alguém gostar de um filme → gostos

Introdução

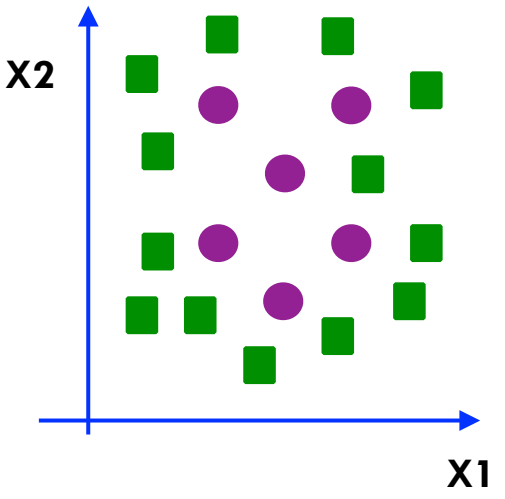
- **Ideia geral**
  - ter vários algoritmos, cada um deles com resultados levemente diferentes em um mesmo *dataset*
    - alguns aprendem alguns padrões muito bem
    - outros aprendem bem outros padrões
  - colocar todos eles juntos
    - resultado do conjunto é melhor do que suas performances isoladas

Introdução



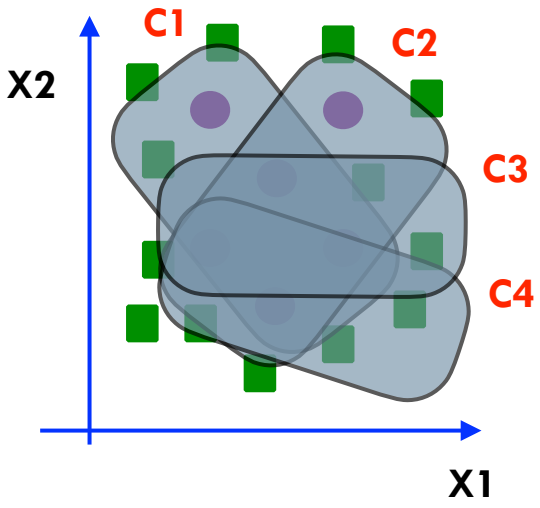
Problema original

Introdução



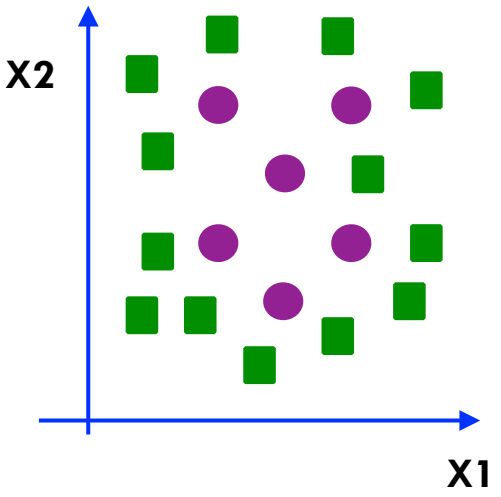
Problema original

diferentes classificadores (Ci)



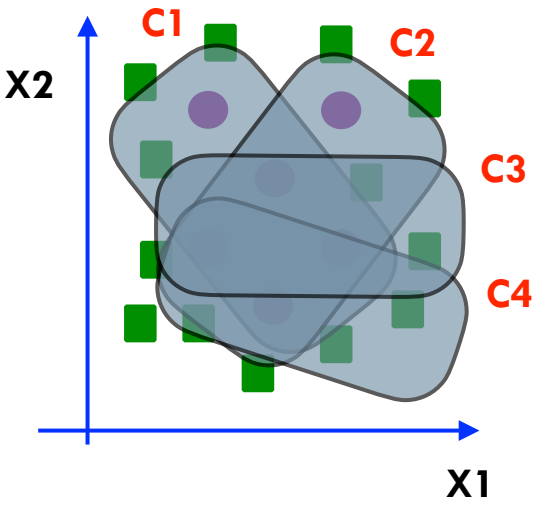
diferentes superfícies

Introdução



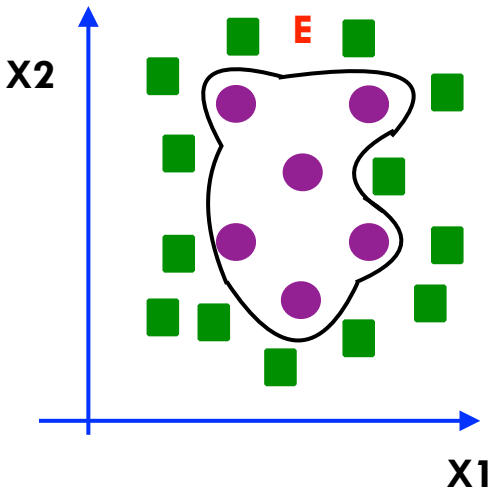
Problema original

diferentes classificadores (Ci)



diferentes superfícies

Ensemble (E)



superfície + complexa

Introdução

- Algumas questões que carecem de respostas?

- Q1: **quais** algoritmos usamos como ***base learners***?

- Q2: como podemos garantir que esses ***base learners*** **aprendem** coisas **diferentes**?

- Q3: como podemos **combinar** esses resultados?

Roteiro

1. Introdução
2. *Voting*
3. *Boosting*
4. *Bagging*
5. *Random Forest*
6. Referências

Voting

- **Voting**
  - Uma forma simples de combinar diferentes learners
  - predição
    - votação majoritária (classificação)
    - média/mediana (regressão)
  - classificador baseado no voto majoritário é comumente dito ser "*hard voting*"

Voting

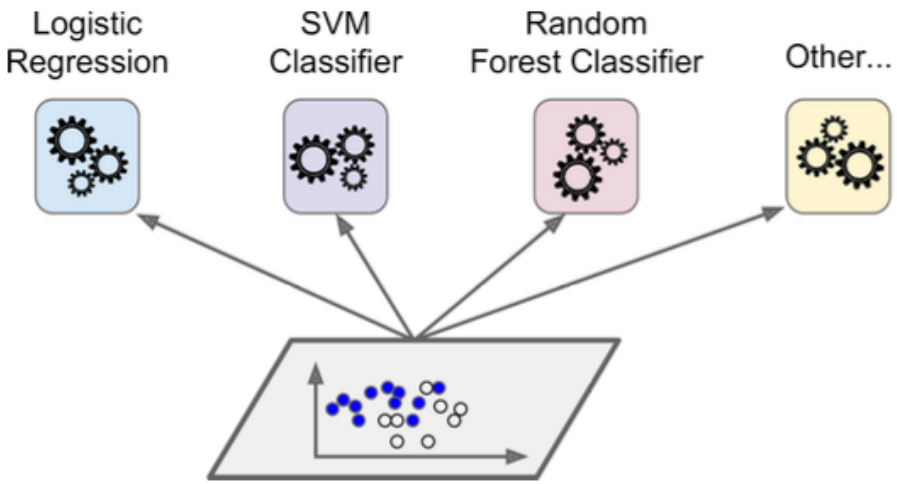
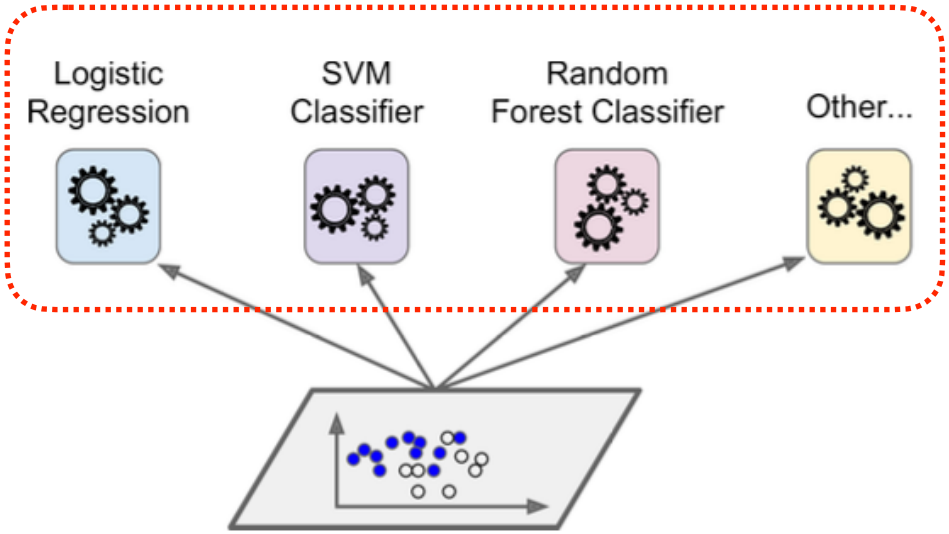


Figura de: Aurélien Gerón (2019)

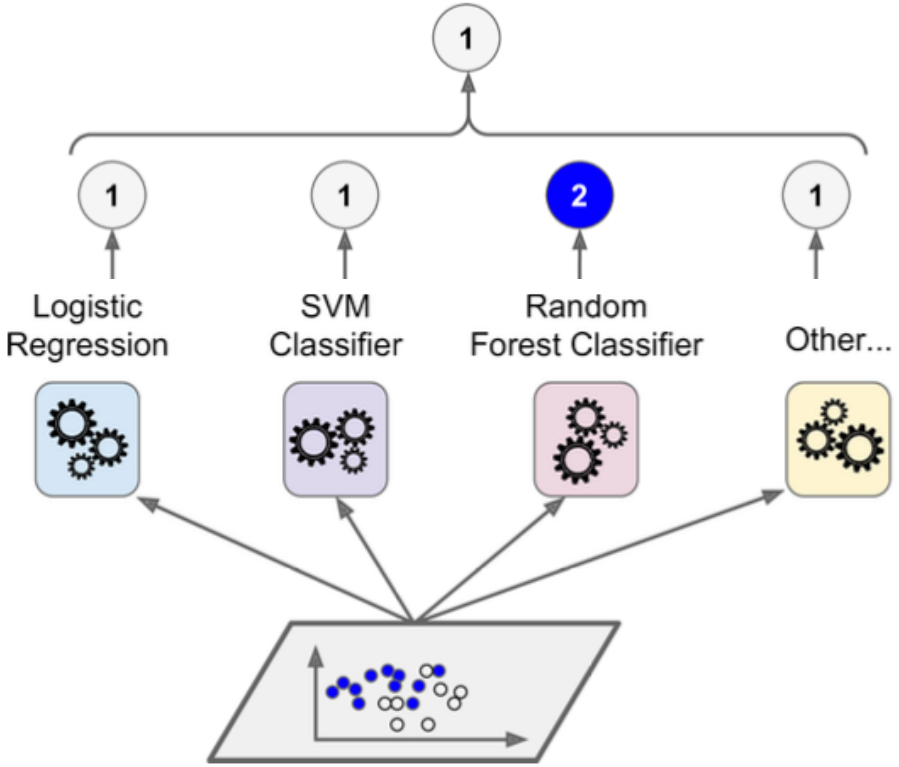
Voting



diferentes base learners (N)

Figura de: Aurélien Gerón (2019)

Voting



Predições

diferentes base learners (N)

Figura de: Aurélien Gerón (2019)

Voting

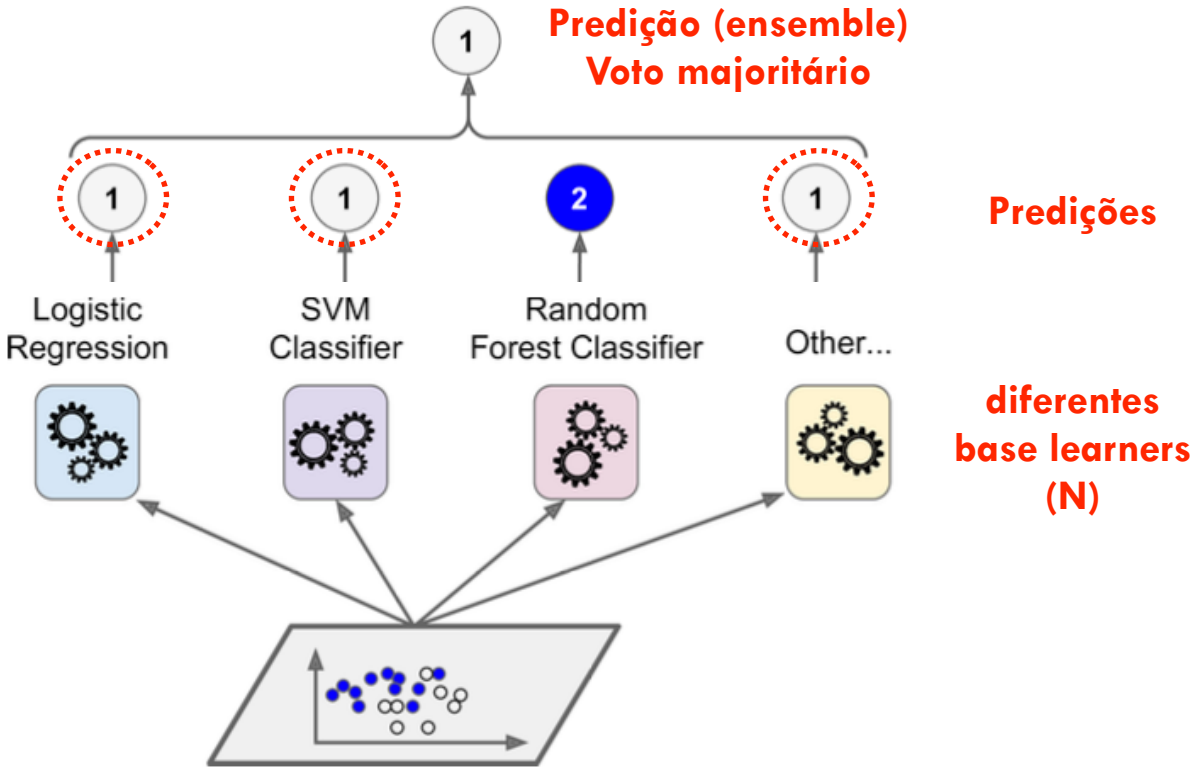
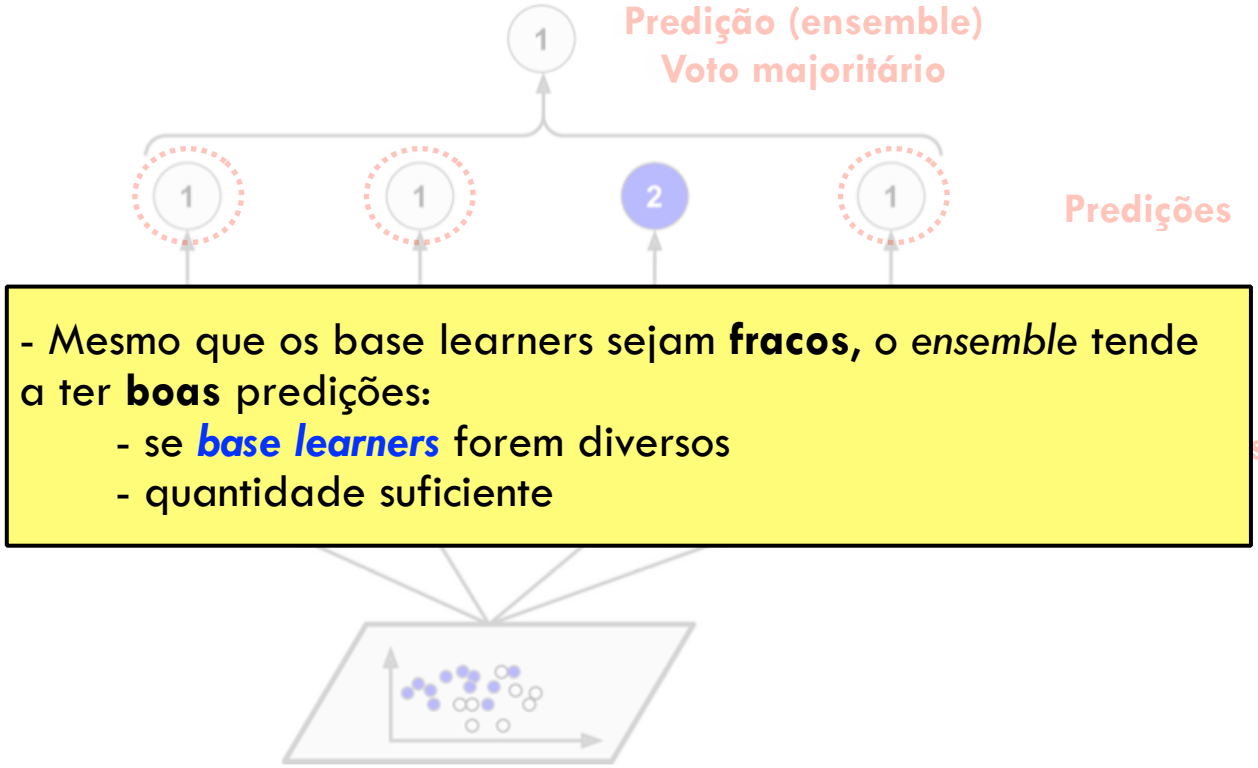


Figura de: Aurélien Gerón (2019)

Voting



- Mesmo que os base learners sejam **fracos,** o *ensemble* tende a ter **boas** predições:
  - se ***base learners*** forem diversos
  - quantidade suficiente

Figura de: Aurélien Gerón (2019)

Voting

- **Características**
  - algoritmos também podem prever **probabilidades**
    - predição é a média das probabilidades entre os diferentes *base learners* (soft voting)
  - tende a alcançar melhores performances, pois probabilidades mais “fortes" (com mais certeza) tendem a ter um peso maior/confiante no voto do comitê

Roteiro

1. Introdução
2. *Voting*
3. ***Boosting***
4. *Bagging*
5. *Random Forest*
6. Referências

Boosting

- um dos métodos mais populares para Ensembles
  - coleção de base learners fracos
  - cada um tem desempenho um pouco superior do que um palpite aleatório
  - colocando todos eles em um mesmo algoritmo é possível criar um *learner* com boas predições

- **Ideia geral:** treinar alguns classificadores sequencialmente, cada um deles tentando corrigir os erros do predecessor

Boosting

- vários algoritmos dentro desse supergrupo (*Boosting*)
  - primeiras ideias → Freund & Schapire (1999)
  - ainda é um dos algoritmos mais usados em Aprendizado de Máquina (AM)
  - comitês iterativos
  - corrigir os erros das predições anteriores
  - mais famoso: **AdaBoost**

AdaBoost

- **AdaBoost** (*Adaptive Boosting*)
  - inovação → é usar pesos para cada uma das amostras classificadas
    - pesos são *inputs* do algoritmo e atualizados frequentemente
    - atualizar os pesos das amostras classificadas erradamente pelos classificadores anteriores
  - é um algoritmo iterativo e sequencial
    - o base learner de uma iteração depende dos resultados da iteração anterior

AdaBoost

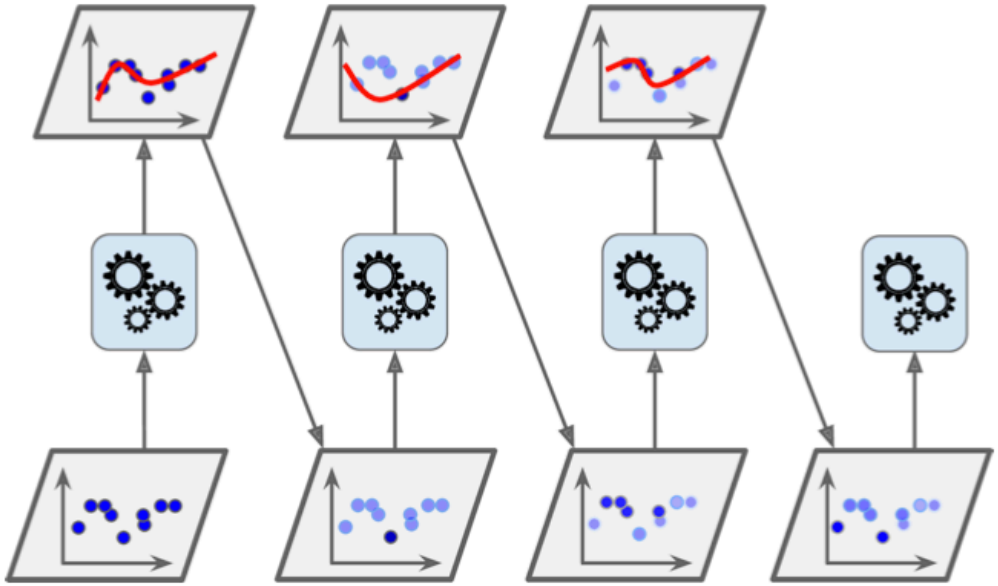


Figura de: Aurélien Gerón (2019)

AdaBoost

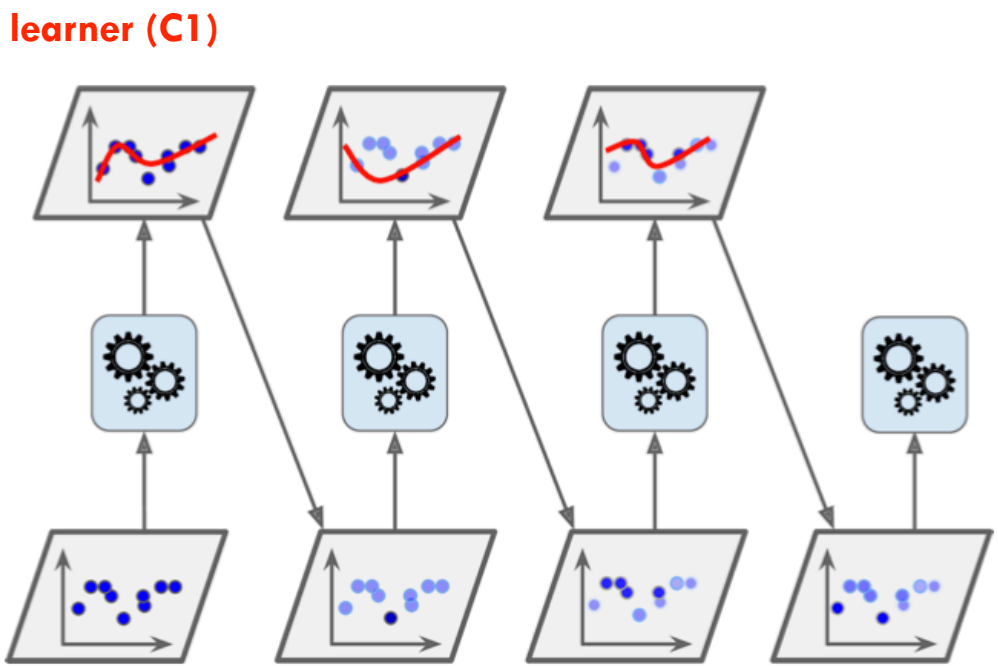


Figura de: Aurélien Gerón (2019)

AdaBoost

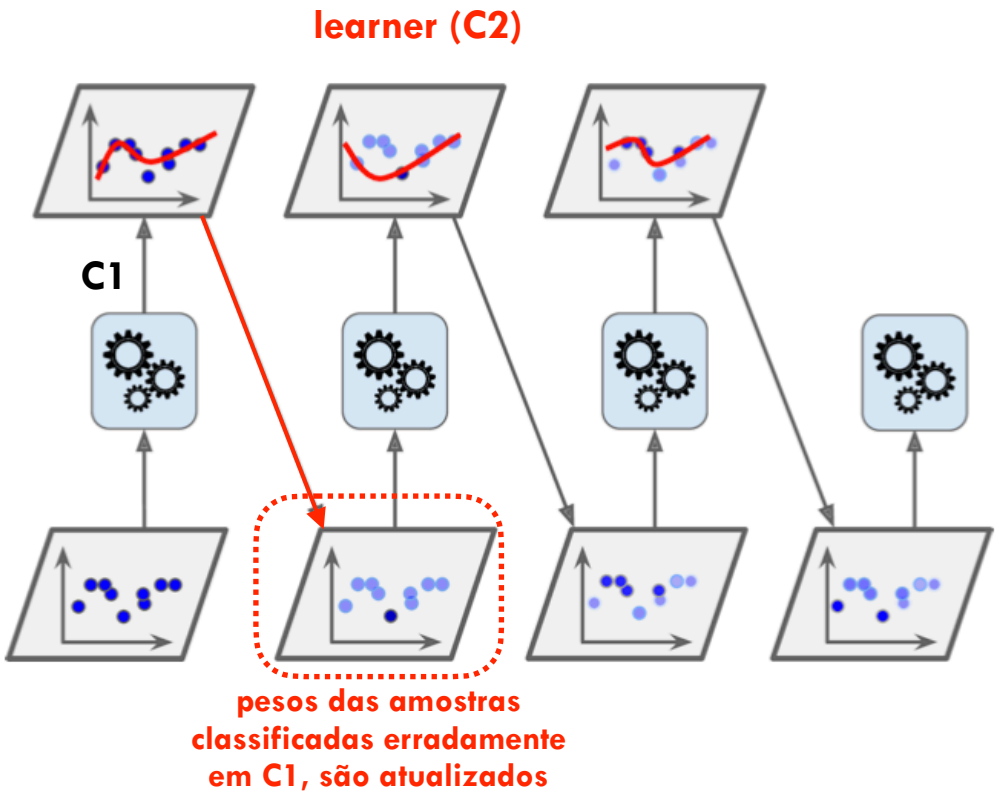


Figura de: Aurélien Gerón (2019)

AdaBoost

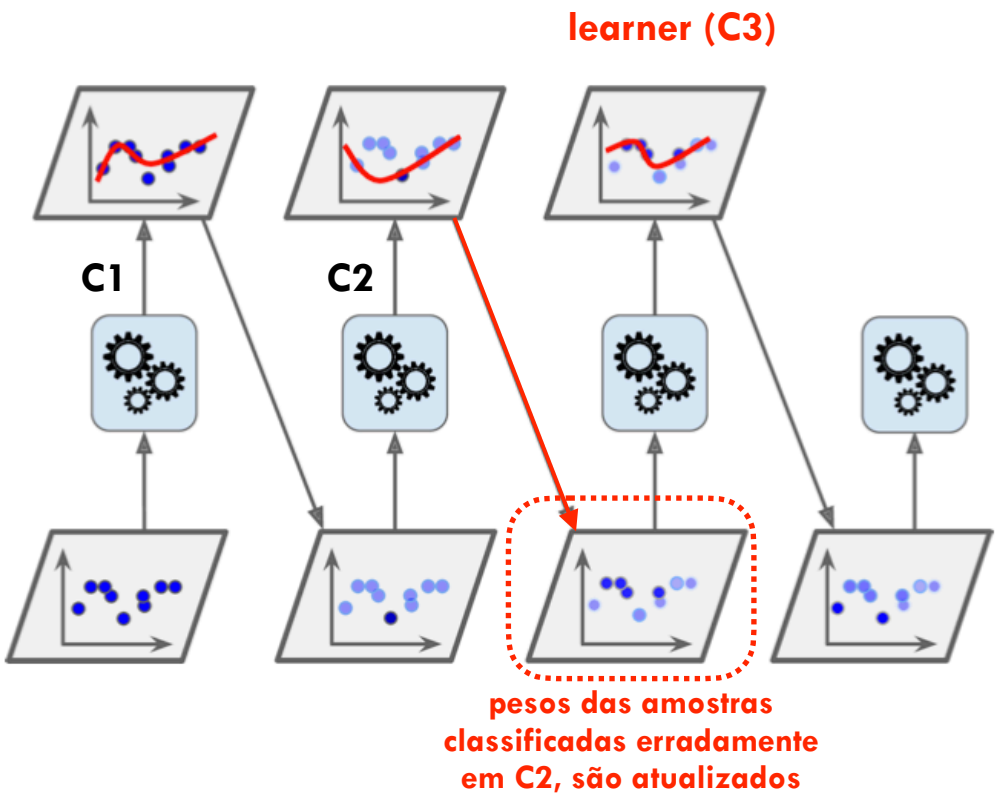


Figura de: Aurélien Gerón (2019)

AdaBoost

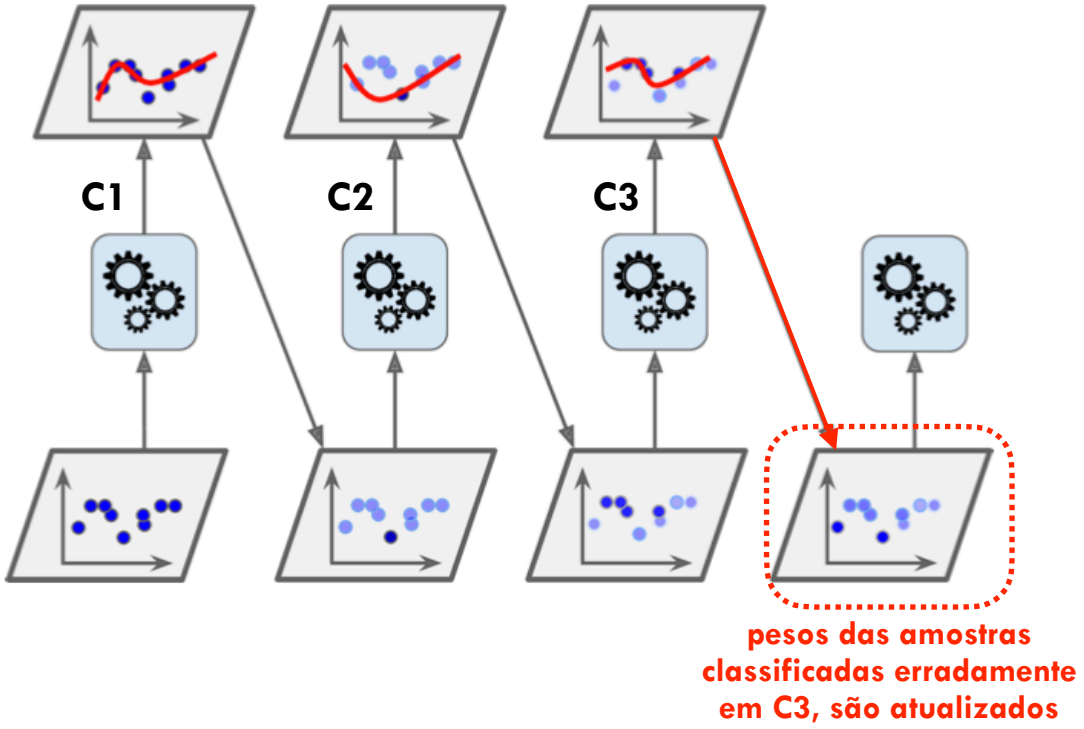


Figura de: Aurélien Gerón (2019)

AdaBoost

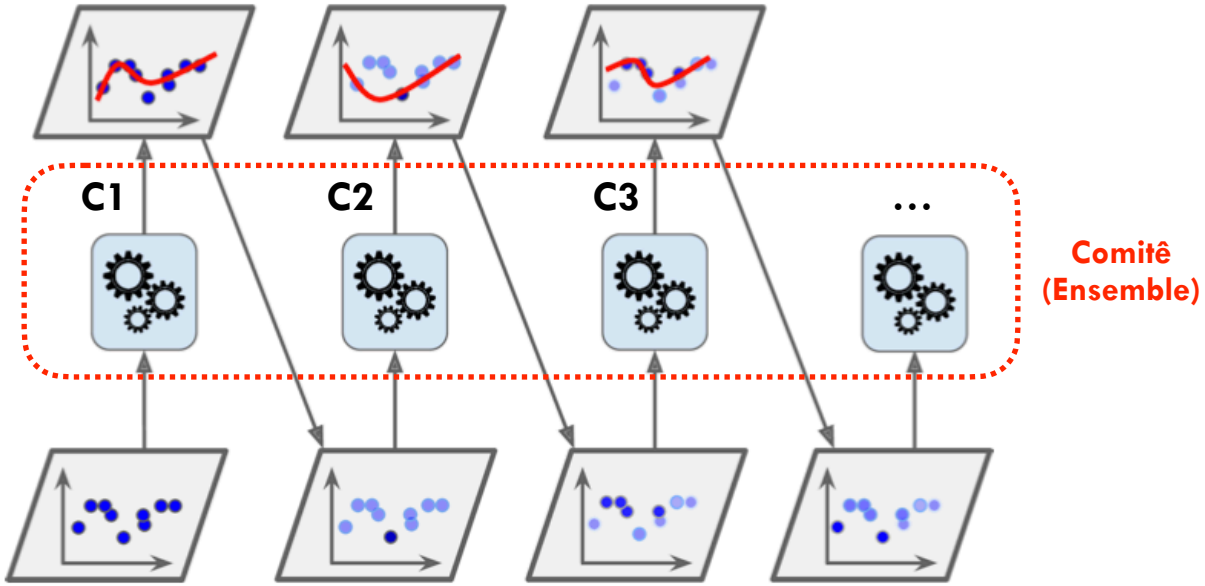


Figura de: Aurélien Gerón (2019)

AdaBoost

- **Como funciona?**
  - N é o tamanho do conjunto de treinamento (exemplos)
  - pesos de cada amostra são inicializados com o mesmo valor: 1 / N
  - A cada iteração uma estimativa de erro (ε) é computada
    - soma dos pesos das amostras classificadas erradamente
    - os pesos das amostras classificadas erradamente são ajustados por um fator α:

$$\alpha = (1 - \epsilon) / \epsilon$$

AdaBoost

- pesos das amostras classificadas corretamente não são modificados
- o conjunto total de pesos é normalizado, de forma que a soma de todos eles = 1

- Treinamento **termina** quando:
  - ou um número de iterações (base learners) é satisfeito
  - ou todas as amostras são corretamente classificadas
  - ou uma das amostras tem peso > 0.5

AdaBoost

Pseudocódigo:

1. Inicializar todos os pesos w com $1/N$
2. Enquanto $0 < \varepsilon_t < 1/2$, e $t < T$, T é o número máximo de iterações
   - 2.1 treinar o classificador em $\{S, w_t\}$
   - 2.2 computar o erro de treinamento
   - 2.3 calcular o peso da iteração (α_t)
   - 2.4 atualizar o peso das amostras classificadas erradamente
   - 2.5 normalizar o vetor dos pesos, de forma que a soma = 1

Output:

- T classificadores: comitê/ensemble
- α: vetor de pesos das iterações

AdaBoost

- **Predições** (pós-treinamento)
  - computa as predições de todos os base learners (iterações) e as pondera usando os valores de α_j computados em cada uma das iterações
  - classe predita pelo ensemble → classe que recebe a maior parte dos votos ponderados

$$\hat{y}(x) = \underset{k}{argmax} \sum_{\substack{j=1 \\ \hat{y}(x)=k}}^{M} \alpha_j$$

Roteiro

1. Introdução
2. *Voting*
3. *Boosting*
4. *Bagging*
5. *Random Forest*
6. Referências

Bagging

- Proposto por Breiman (1996)

- **Ideia** geral:
  - algoritmo simples
  - treinar o mesmo base learner em diferentes sub-amostras do problema/*dataset* original
  - amostragens com reposição (*sampling with replacement*)
    - mesmo exemplo pode ser amostrado várias vezes
  - Bagging (bootstrap aggregation)

Bagging

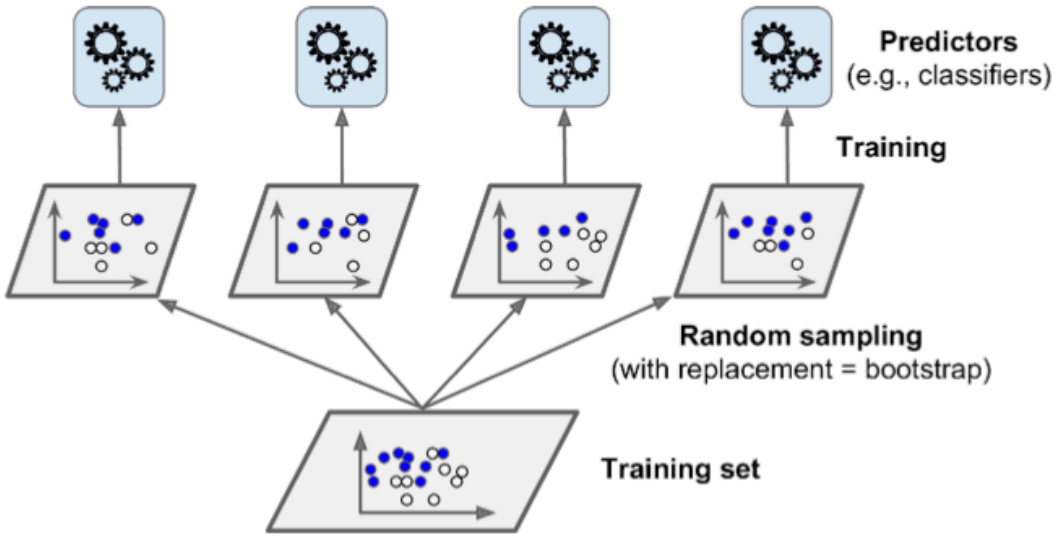


Figura de: Aurélien Gerón (2019)

Bagging

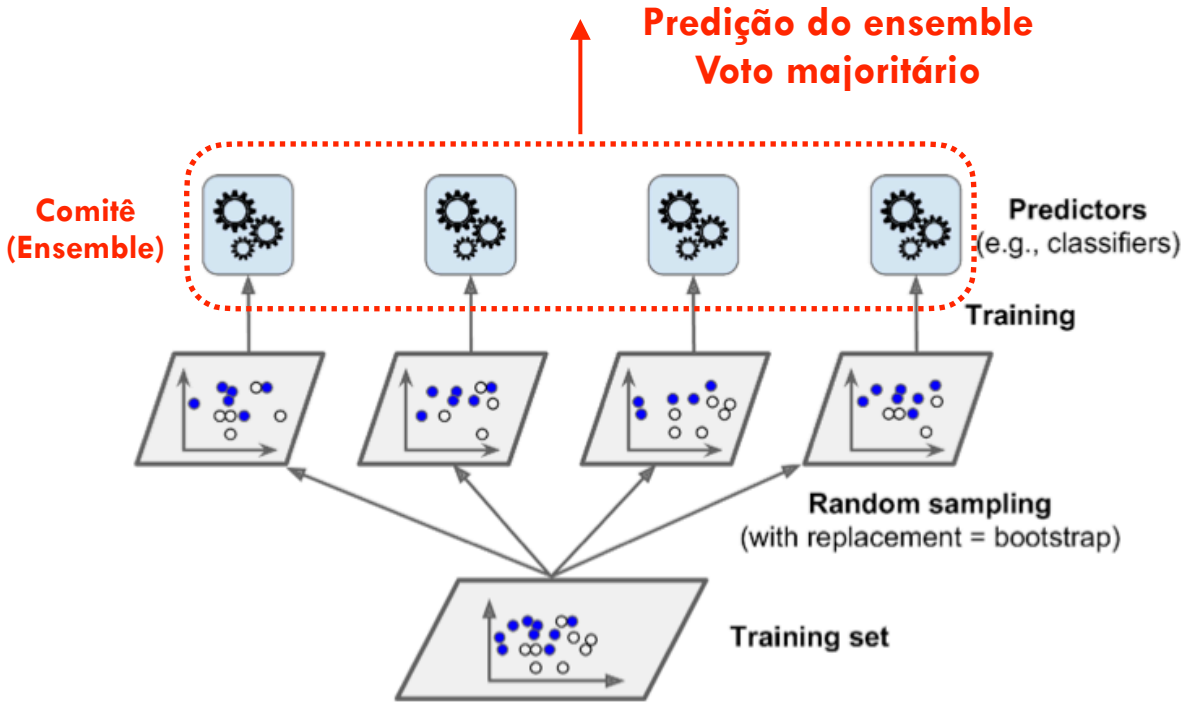


Figura de: Aurélien Gerón (2019)

Bagging

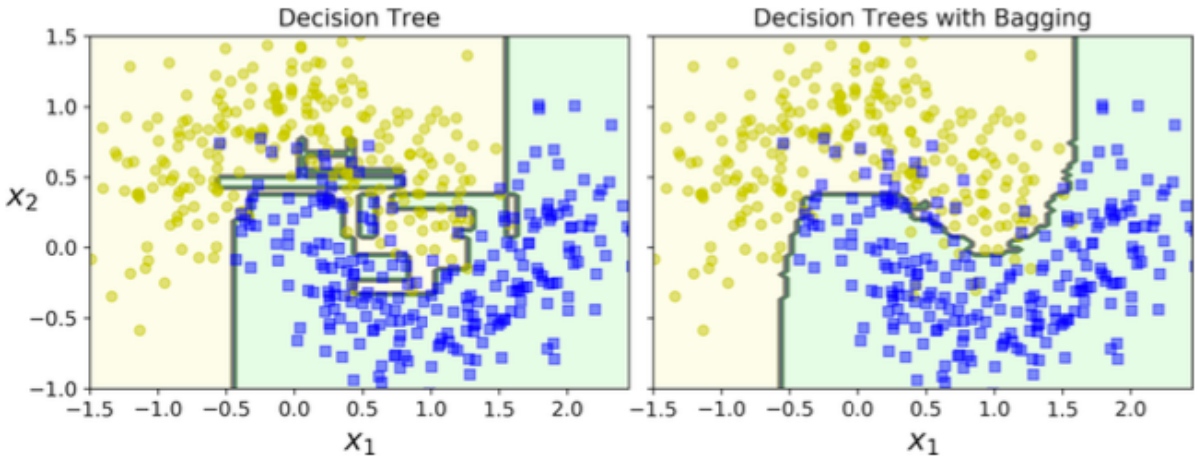


Figura de: Aurélien Gerón (2019)
dataset: moon, Bagging com 500 DTs

Bagging

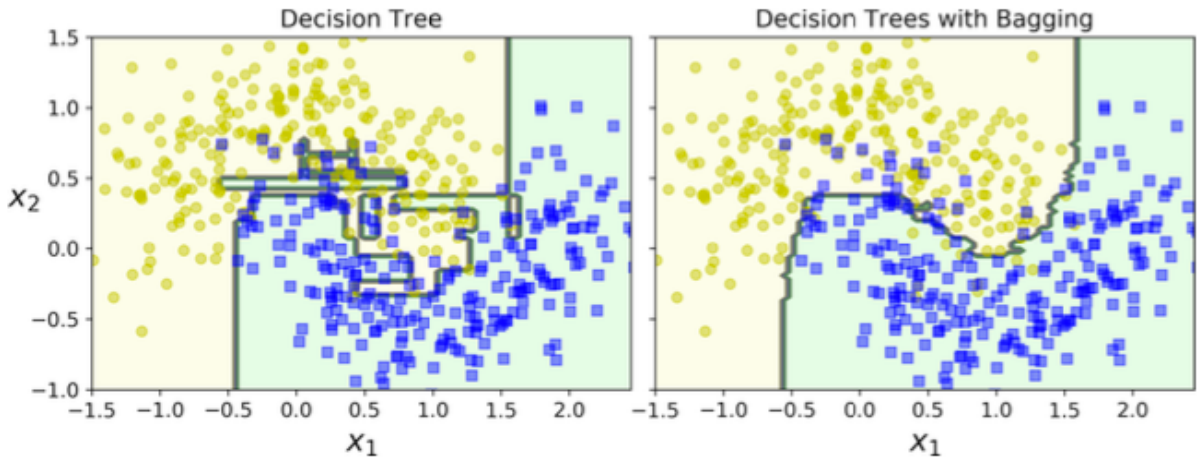


Figura de: Aurélien Gerón (2019)
dataset: moon, Bagging com 500 DTs

- ***Bagging*** (ensemble) **generaliza melhor**
- **superfície** de decisão é menos **irregular**
- frequentemente: Bagging > DT

Bagging

- **Características**
  - amostragem é igual ao tamanho do conjunto de treinamento
  - realiza um número finito de amostras (B)
  - beneficia de ser composto de muitos base learners que aprendem padrões levemente diferentes

Bagging

Input:

- B: quantidade de amostras/classificadores no comitê

1. Realizar B amostragens com reposição
2. Treinar um classificador para cada amostra $b \in B$
3. Combinar a saída dos classificadores
   - voto majoritário (classificação)
   - mediana (regressão)

Output:

- B classificadores treinados nas amostras de treinamento

Roteiro

1. Introdução
2. *Voting*
3. *Boosting*
4. *Bagging*
5. *Random Forest*
6. Referências

Random Forest

- Proposto por Breiman (2001)
  - *Random Forest* → Floresta Aleatória
  - Ensemble de árvores de decisão

- **Ideia** geral:
    - algoritmo melhorado a partir do *Bagging*
    - adiciona mais um nível de **aleatoriedade** na criação do ensemble
    - limita a quantidade de *features* usadas para gerar cada árvore de decisão
        - selecionadas aleatoriamente

Random Forest

- **Características**
  - incrementar a aleatoriedade (*subset* de *features*) torna o treinamento do algoritmo mais rápido
    - poucas features para cada árvore
  - introduz um novo hiperparâmetro:
    - quantidade de features na subamostra
    - literatura reporta que RF não são sensíveis a essa escolha
    - comum: $\sqrt{F}$, com F = número de features do *dataset*

Random Forest

- **Características**
  - segundo hiperparâmetro é o número de árvores (T)
    - s1: definir T a priori
    - s2: adicionar árvores enquanto o erro de treinamento não para de reduzir
  - bootstrap + subset de features
    - reduz a variância do algoritmo sem afetar o *bias*
    - não há necessidade em podar as árvores

Random Forest

Input:

- dataset: com N exemplos, e F features
- T: quantidade de árvores

1. Para cada uma das **T** árvores

1.1 Realizar uma amostragem de tamanho **N** dos exemplos fazendo reposição (*bootstrap with replacement*)

1.2. Aleatoriamente, selecionar apenas $\sqrt{F}$ features da amostra

1.3. Treinar um classificador com esse *subset* gerado

2. Combinar a saída dos classificadores

- voto majoritário (classificação)
- mediana (regressão)

Output:

- **T** classificadores treinados nas amostras de treinamento

Random Forest

- Importância relativa dos atributos
  - Gini index
  - quanto de impureza é reduzido em média (sobre todas as árvores) quando um atributo específico (*feature*) está na árvore
  - quais os atributos mais importantes/descritivos para um problema

Random Forest

- **Comparativo Geral**
  - RF | *Bagging* são algoritmos que podem ser executados em paralelo
  - Boosting é sequencial (dependência dos modelos)
  - Para um mesmo número de árvores
    - Boosting tende a ser melhor
    - mas RF consegue criar um ensemble maior com o mesmo custo computacional (e ser melhor)
  - RF é robusto e tem desempenhos bons em datasets pequenos e grandes

Hands on

Vamos exercitar :)
[Google Colab - Exemplo 01]
[Google Colab - Exemplo 02]

Roteiro

1. Introdução
2. *Voting*
3. *Boosting*
4. *Bagging*
5. *Random Forest*
6. Referências

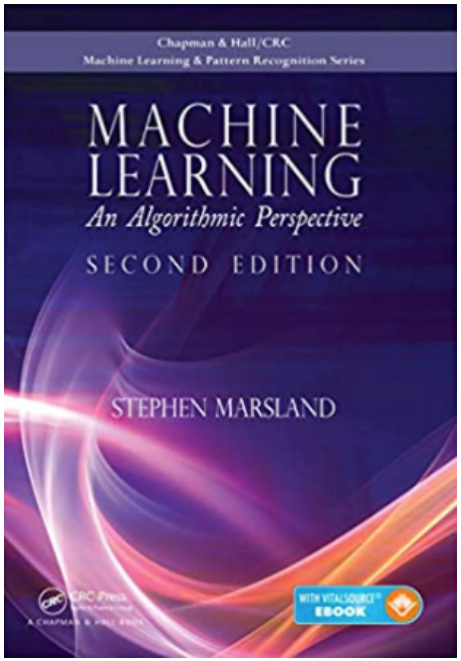
Síntese

- *Comitês / Ensembles*
  - *Voting*
  - *Boosting*
  - *Bagging*
  - *Random Forest*

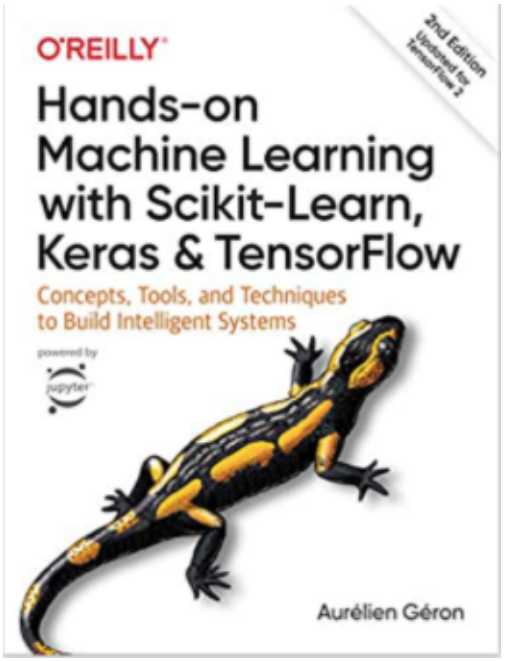
Roteiro

1. Introdução
2. *Voting*
3. *Boosting*
4. *Bagging*
5. *Random Forest*
6. Síntese / Próximas Aulas
7. Referências

Literatura Sugerida



(Marsland, 2014)



(Géron, 2019)

Obrigado :)

Rafael G. Mantovani
rafaelmantovani@utfpr.edu.br