

# 11 Principais Algoritmos

Muitos algoritmos de aprendizado de máquina variam de simples a complexos em sua abordagem, e juntos fornecem uma poderosa biblioteca de ferramentas para analisar e prever padrões a partir de dados

Este artigo abordará algoritmos de aprendizagem de máquina que são comumente usados na comunidade de ciência de dados, um resumo conciso de cada um e apontar algumas das principais características.

Com isso em mente, vou começar com alguns dos algoritmos mais fundamentais e depois mergulhar em alguns algoritmos mais novos como **CatBoost**, **Gradient Boost** e **XGBoost**.

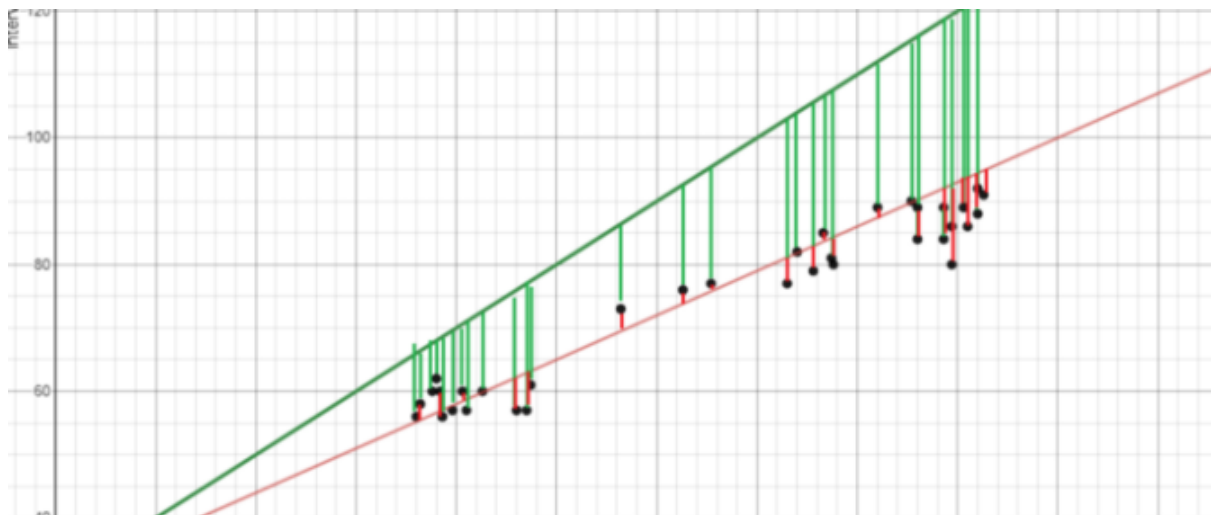
## 1. Regressão Linear

A Regressão Linear é um dos algoritmos mais fundamentais utilizados para **modelar relações entre uma variável dependente e uma ou mais variáveis independentes**.

Em termos mais simples, envolve **encontrar a "linha de melhor ajuste"** que representa duas ou mais variáveis.

A linha de melhor ajuste é encontrada minimizando as distâncias quadradas entre os pontos e a linha de melhor ajuste – isso é conhecido como minimizar a soma dos resíduos quadrados.

Um residual é simplesmente igual ao **valor previsto menos o valor real**.



Comparando a linha verde de melhor ajuste à linha vermelha, observe como as linhas verticais (os resíduos) são muito maiores para a linha verde do que a linha vermelha. Isso faz sentido porque a linha verde está tão longe dos pontos que não é uma boa representação dos dados

## 2. Regressão Logística

A regressão logística é semelhante à regressão linear, mas é usada para **modelar a probabilidade de um número discreto de desfechos**, tipicamente dois.

De relance, a regressão logística soa muito mais complicada do que a regressão linear, mas na verdade **só tem um passo a mais**.

Primeiro, você calcula uma pontuação usando uma equação semelhante à equação para a linha de melhor ajuste para regressão linear.

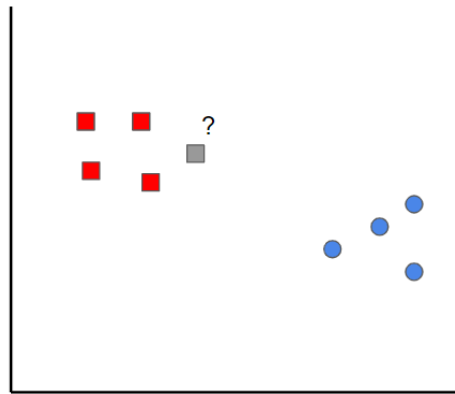
O passo extra é alimentar a pontuação calculada anteriormente na função sigmoid abaixo para que você obtenha uma probabilidade em troca.

Essa probabilidade pode então ser convertida em uma saída binária, 1 ou 0.

Para encontrar os pesos da equação inicial e calcular a pontuação, métodos como **descida gradiente ou probabilidade**

máxima são usados.

### 3.k-Vizinhos mais próximos



K-vizinhos mais próximos é uma ideia simples. Primeiro, você começa com dados que já estão classificados (ou seja, os pontos de dados vermelho e azul). Em seguida, quando você adiciona um novo ponto de dados, você classifica-o olhando para os pontos classificados mais próximos.

Qualquer classe que obtenha mais votos determina como o novo ponto é classificado.

Neste caso, se definirmos  $k=1$ , podemos ver que o primeiro ponto mais próximo da amostra cinza é um ponto de dados vermelho. Portanto, o ponto seria classificado como vermelho.

Algo a ter em mente é que se o valor de  $k$  é definido muito baixo, ele pode estar sujeito a outliers. Por outro lado, se o valor de  $k$  é definido muito alto, então ele pode ignorar as aulas com apenas algumas amostras.

### 4.Naive Bayes (Probabilidade Condicional)

Naive Bayes é um algoritmo de classificação, isso significa que é usado quando a variável de saída é discreta.

Pode parecer um algoritmo assustador porque requer conhecimento matemático preliminar em probabilidade condicional e Teorema de Bayes, mas é um conceito extremamente simples e "ingênuo":

Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
overcast	hot	high	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	cool	normal	FALSE	yes
rainy	cool	normal	TRUE	no
overcast	cool	normal	TRUE	yes
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
rainy	mild	normal	FALSE	yes
sunny	mild	normal	TRUE	yes
overcast	mild	high	TRUE	yes
overcast	hot	normal	FALSE	yes
rainy	mild	high	TRUE	no

Suponha que tenhamos dados de entrada sobre as características do tempo (perspectiva, temperatura, umidade, vento) e se você jogou golfe ou não (ou seja, última coluna).

O que o Naive Bayes essencialmente faz é **comparar a proporção entre cada variável de entrada e as categorias na variável de saída**. Isso pode ser mostrado na tabela abaixo:

outlook			temperature			humidity			windy			play?	
	yes	no		yes	no		yes	no		yes	no	yes	no
sunny	2/9	3/5	hot	2/9	2/5	high	3/9	4/5	FALSE	6/9	2/5	9/14	5/14
overcast	4/9	0/5	mild	4/9	2/5	normal	6/9	1/5	TRUE	3/9	3/5		
rainy	3/9	2/5	cool	3/9	1/5								

Por exemplo, na seção de **temperatura**, estava quente por dois dos nove dias que jogamos golfe (ou seja, sim).

Em termos matemáticos, podemos escrever isso como **a probabilidade de ser quente DADO que jogamos golfe**.

A notação matemática é  $P(\text{hot}|\text{yes})$ , isso é conhecido como probabilidade condicional.

A partir disso, podemos prever se vamos jogar golfe ou não para qualquer combinação de características meteorológicas.

Imagine que temos um novo dia com as seguintes características:

- Perspectiva: ensolarado
- Temperatura: amena
- Umidade: normal
- Windy: falso

Primeiro, vamos calcular a probabilidade de jogar golfe:

outlook			temperature			humidity			windy			play?	
	yes	no		yes	no		yes	no		yes	no	yes	no
sunny	2/9	3/5	hot	2/9	2/5	high	3/9	4/5	FALSE	6/9	2/5	9/14	5/14
overcast	4/9	0/5	mild	4/9	2/5	normal	6/9	1/5	TRUE	3/9	3/5		
rainy	3/9	2/5	cool	3/9	1/5								

Usando o gráfico acima, podemos obter as seguintes informações:

$$P(\text{yes}) = 9/14$$

$$P(\text{outlook} = \text{sunny}|\text{yes}) = 2/9$$

$$P(\text{temperature} = \text{mild}|\text{yes}) = 4/9$$

$$P(\text{humidity} = \text{normal}|\text{yes}) = 6/9$$

$$P(\text{windy} = \text{false}|\text{yes}) = 6/9$$

Agora podemos simplesmente inserir essas informações na seguinte fórmula:

$$P(\text{yes}|X) \propto P(X|y) * P(y)$$

$$P(\text{yes}|X) \propto P(x_1|y) * P(x_2|y) * P(x_3|y) * P(x_4|y) * P(y)$$

$$P(\text{yes}|X) \propto P(\text{sunny}|\text{yes}) * P(\text{mild}|\text{yes}) * P(\text{normal}|\text{yes}) * P(\text{false}|\text{yes}) * P(\text{yes})$$

$$P(\text{yes}|X) \propto \frac{2}{9} * \frac{4}{9} * \frac{6}{9} * \frac{6}{9} * \frac{9}{14}$$

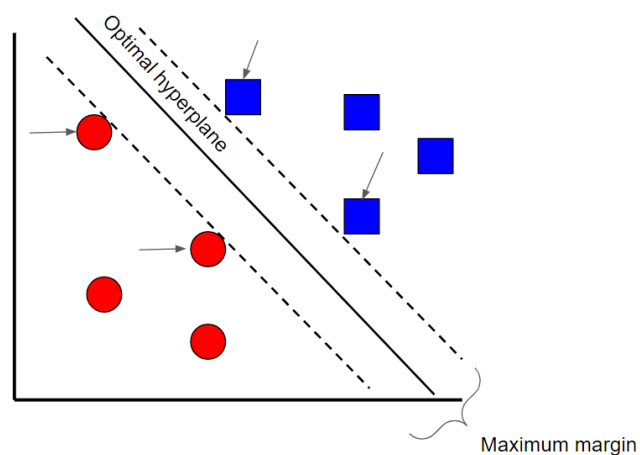
$$P(\text{yes}|X) \propto \mathbf{0.0282}$$

Da mesma forma completariamos a mesma sequência de etapas para  $P(\text{não} | X)$ .

$$P(\text{no}|X) \propto \mathbf{0.0069}$$

Porque  $P(\text{sim} | X) > P(\text{não} | X)$ , então podemos prever que essa pessoa jogaria golfe dado que a perspectiva é ensolarada, a temperatura é amena, a umidade é normal, e não está ventando.

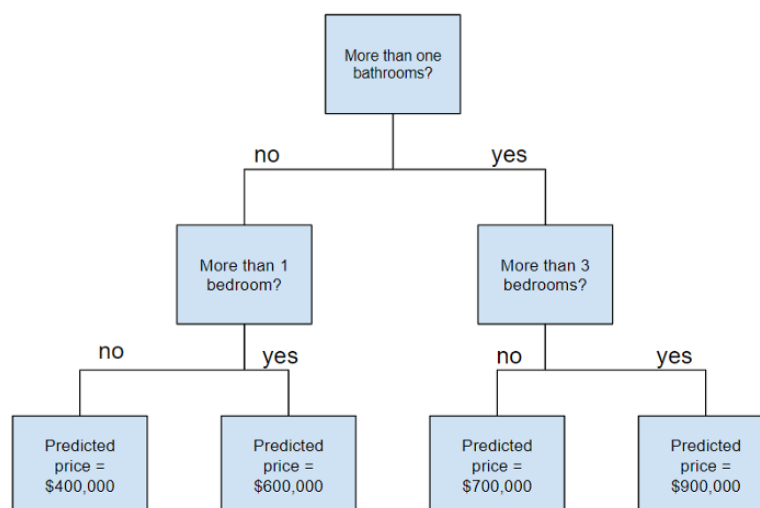
## 5. SVM (Classificação Supervisionada)



Uma Máquina vetorial de suporte é uma **técnica de classificação supervisionada** que pode realmente ficar muito complicada, mas é bastante intuitiva no nível mais fundamental.

Vamos supor que existem duas classes de dados. Uma **máquina vetorial de suporte encontrará um hiperplano / um limite entre as duas classes de dados** que maximiza a **margem** entre as duas classes (veja acima). Existem muitos planos que podem separar as duas classes, mas apenas um plano pode maximizar a margem ou distância entre as classes.

## 6.Árvore de Decisão



## Florestas Aleatórias (Conjunto de Árvores de Decisão)

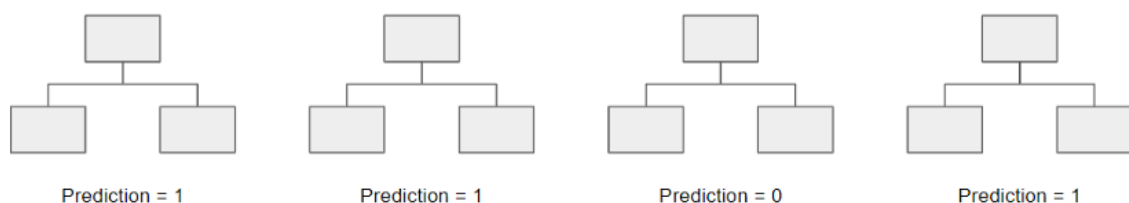
- O **aprendizado em conjunto ensemble**: é um método onde vários algoritmos de aprendizagem são usados em conjunto. O objetivo de fazê-lo é que ele **permite alcançar um desempenho preditivo maior do que se você fosse usar um algoritmo individual por si só**.
- A **amostragem de bootstrap sampling**: é um método de resamplagem que usa **amostragem aleatória com substituição**.

- **Bagging:** quando usamos o agregado dos conjuntos de dados bootstrapped para tomar uma decisão.

Florestas aleatórias são uma técnica de aprendizado de conjunto que se constrói a partir de árvores de decisão.

Florestas aleatórias envolvem a criação de **múltiplas árvores de decisão** usando conjuntos de dados originais e selecionando aleatoriamente um subconjunto de variáveis em cada etapa da árvore de decisão.

O modelo então seleciona o modo de todas as previsões de cada árvore de decisão (bagging). Qual é o objetivo disso? Ao confiar em um modelo de "maioria ganha" com ensemble, reduz o risco de erro de uma árvore individual.



Por exemplo, se criássemos uma árvore de decisão, a terceira, ela preveria 0. Mas se dependêssemos do modo de todas as 4 árvores de decisão, o valor previsto seria 1. Este é o poder das florestas aleatórias.

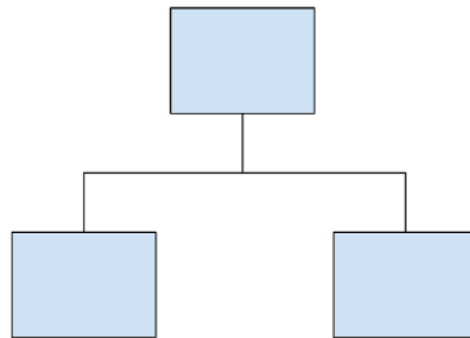
## 7. AdaBoost (constrói tocos)

AdaBoost, ou Adaptive Boost, também é um algoritmo de conjunto que aproveita métodos de bagging e impulsionamento para desenvolver um preditor aprimorado.

AdaBoost é semelhante a Random Forests no sentido de que as previsões são tomadas de muitas árvores de decisão. No entanto, existem três principais diferenças que tornam o AdaBoost único:



## Toco: um nó e duas folhas



Exemplo de um toco.

1. Primeiro, **AdaBoost cria uma floresta de tocos** em vez de árvores. Um toco é uma árvore que é feita de apenas um nó e duas folhas
2. Em segundo lugar, os tocos que são criados não são igualmente ponderados na decisão final (previsão final). **Tocos que criam mais erros terão menos a dizer na decisão final.**
3. Por fim, a ordem em que os tocos são feitos é importante porque **cada toco visa reduzir os erros que os tocos anteriores cometeram.**

Em essência, o AdaBoost adota uma abordagem mais iterativa no sentido de que **busca melhorar iterativamente através dos erros anteriores cometidos.**

## 8. Gradient Boost (constrói árvores)

**Gradient Boost** é um algoritmo de **conjunto ensemble** que usa métodos de impulsionamento para desenvolver um preditor aprimorado.

Em muitos aspectos, o Gradient Boost é semelhante ao AdaBoost, mas há algumas diferenças importantes:

- Ao contrário do AdaBoost, que constrói tocos, o **Gradient Boost constrói árvores com geralmente 8-32 folhas.**


- O Gradient Boost vê o problema de aumento como um problema de otimização, onde usa uma função de perda e tenta minimizar o erro. É por isso que é chamado de **aumento gradiente**, como é inspirado pela descida gradiente.
- Por fim, as árvores são usadas para prever os resíduos das amostras (previsões - observados).

Embora o último ponto possa ter sido confuso, o que precisamos saber é que o Gradient Boost começa construindo uma árvore para tentar encaixar os dados, e **as árvores subsequentes construídas após o objetivo de reduzir os resíduos (erro)**. Ele faz isso concentrando-se nas áreas onde os alunos existentes tiveram um desempenho ruim, semelhante ao AdaBoost.

## XGBoost vs. LightGBM vs. CatBoost

CatBoost vs. Light GBM vs. XGBoost

I recently participated in this Kaggle competition (WIDS Datathon by Stanford) where I was able to land up in Top 10 using various


 <https://towardsdatascience.com/catboost-vs-light-gbm-vs-xgboost-5f93620723db>

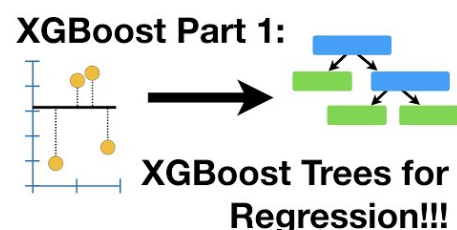


## 9. XGBoost (Muito poderoso)

XGBoost Part 1: Regression

XGBoost is an extreme machine learning algorithm, and that means it's got lots of parts. In this video, we focus on the unique regression trees

 <https://www.youtube.com/watch?v=0tD8wVaFm6E&t=3s>



XGBoost é um dos algoritmos mais populares e amplamente utilizados hoje em dia porque é simplesmente **muito poderoso**.

É semelhante ao Gradient Boost, mas tem alguns **recursos extras** que o tornam muito mais forte, incluindo:

- **Um encolhimento proporcional dos nós de folhas (poda):** usado para melhorar a generalização do modelo
- **Newton Boosting:** fornece uma rota direta para o mínimo do que descida gradiente, tornando-o muito mais rápido
- **Um parâmetro extra de randomização:** reduz a correlação entre árvores, melhorando a força do conjunto
- **Penalização única das árvores**

## 10. **LightGBM** (algoritmo de impulsioneamento)

LightGBM é outro tipo de algoritmo de impulsioneamento que tem se mostrado mais rápido e às vezes mais preciso que o XGBoost.

O que torna o **LightGBM diferente** é que ele usa uma técnica única chamada **Goss** (Unique-Side Sampling, **amostragem unilateral** baseada em gradiente) para filtrar as instâncias de dados para encontrar um valor dividido.

Isso é diferente do XGBoost, que usa algoritmos pré-classificados e baseados em histogramas para encontrar a melhor divisão.

## 11. **CatBoost** (requisitos de baixa latência)

CatBoost é outro algoritmo **baseado em Descida gradiente** que tem algumas diferenças sutis que o tornam único:

- CatBoost implementa **árvores simétricas**, que ajudam na **diminuição do tempo de previsão**, e tem uma profundidade de árvore mais rasa por padrão (seis);
- CatBoost aproveita permutações aleatórias semelhantes à forma como o XGBoost tem um parâmetro de randomização;
- Ao contrário do XGBoost, no entanto, o CatBoost lida com recursos categóricos de forma mais elegante, usando conceitos como **impulsioneamento ordenado e codificação de resposta**.

No geral, o que torna o CatBoost tão poderoso são seus requisitos de **baixa latência**, o que se traduz em ser cerca de **oito vezes mais rápido** que o XGBoost.