Published in Data Hackers

Leonardo Berlatto Follow
Apr 25, 2021 · 6 min read

Como Funciona uma Árvore de Decisão

A intuição por trás do algoritmo de árvore de decisão, da matemática à aplicação

Photo by veeterzy on Unsplash

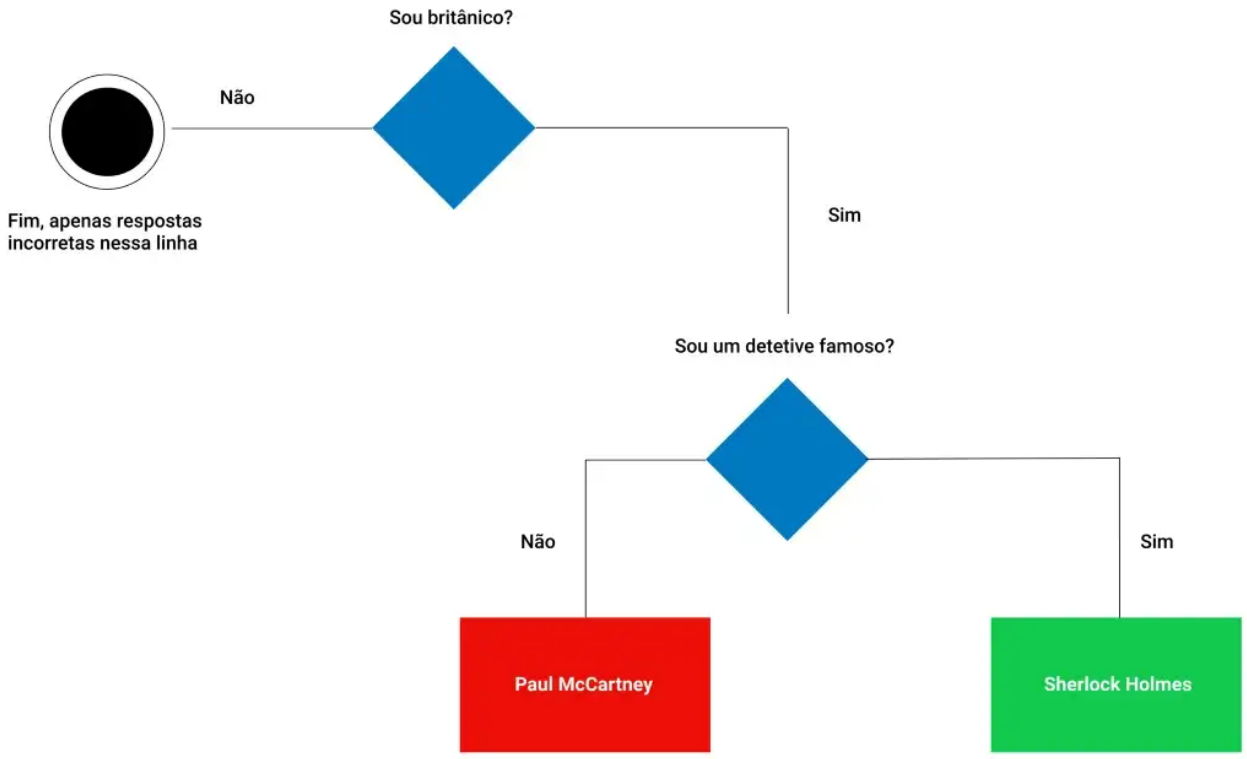


Diagrama do funcionamento do **Quem Sou Eu** onde o personagem era Sherlock Holmes

Mas hoje não estamos aqui para falar sobre jogos. A questão aqui é: você sabia que existe um algoritmo de Machine Learning que aprende de forma similar a como jogamos Q*uem Sou Eu*? Não? Então vem comigo!

A Árvore de Decisão

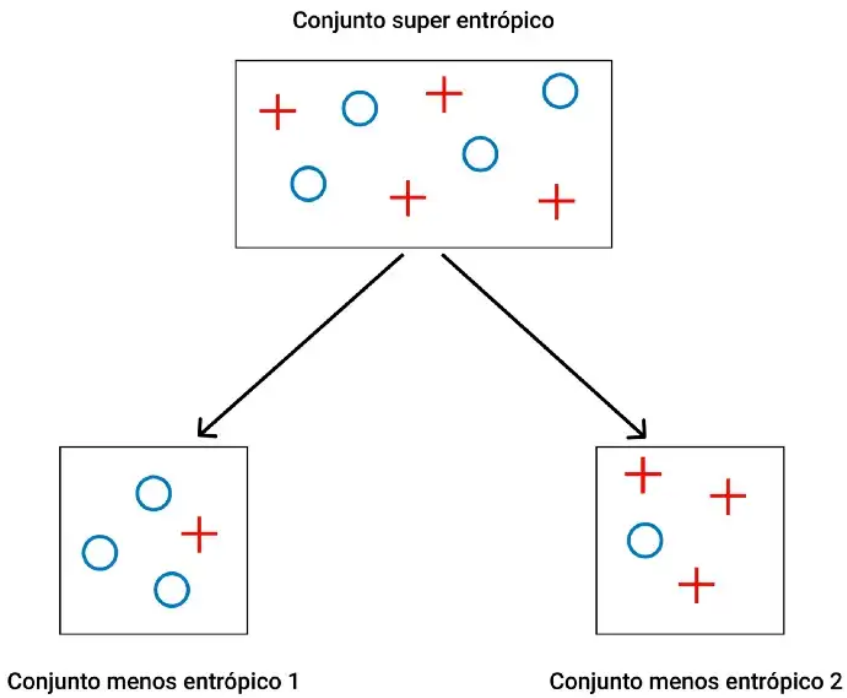
Árvore de Decisão é um dos mais elementares algoritmos de Machine Learning para modelos de classificação, conhecido principalmente por ser um dos algoritmos menos *mágicos* e fáceis de entender o seu funcionamento completo. Como se trata de um modelo de classificação, sua função é prever uma classe dentro de um conjunto finito de opções. Um exemplo de aplicação são modelos que preveem se é recomendável ou não liberar crédito para um cliente com base em seu score do Serasa, salário e outros dados.

em qual classe esse dado se encaixa melhor.

> *Ok, ok. Mas de que forma o algoritmo vai saber como construir essa árvore?*

Entropia

Entropia pode ser definida como a medida que nos diz o quanto nossos dados estão desorganizados e misturados. Quanto maior a entropia, menor o *ganho de informação* e vice-versa. Nossos dados ficam menos entrópicos conforme dividimos os dados em conjuntos capazes de representar apenas uma classe do nosso modelo.



Divisão de um conjunto entrópico em dois menos entrópicos

A partir desse momento, nosso objetivo se torna construir nossa árvore tendo o conjunto de dados inteiro como raiz e criar ramificações baseadas em condições que minimizem a entropia e aumentem o ganho de informação. O valor da entropia de um dado pode ser calculado pela seguinte fórmula:

Fórmula da Entropia

A entropia costuma variar entre 0 e nosso número de classes-1, assumindo seu valor máximo quando as probabilidades de cada classe ocorrer, representadas como p(x) na fórmula, são iguais.

Ganho de Informação

O ganho de informação é uma medida que nos diz o quão bem uma *feature* do conjunto de dados separa os registros conforme as suas classes. Ele é calculado no momento em que estamos comparando a entropia do estado atual dos dados com a entropia que seria obtida com uma nova ramificação, dado através da fórmula abaixo:

$$ganho = Entropia(pai) - \sum_{i=1}^{n} peso(filho_i) * entropia(filho_i)$$

Fórmula do ganho de informação dada uma possível ramificação

O número de filhos presentes no somatório depende de como a ramificação será feita, mas normalmente são gerados dois nós filhos. Onde o peso para cada um desses filhos é calculado dividindo o total de elementos em um nó filho dividido pelo número de elementos no nó pai:

$$peso(filho) = \frac{n°amostrasFilho}{n°amostrasPai}$$

Fórmula para o peso de cada filho na hora de calcular o ganho de informação

Uma vez sabendo como calcular o ganho de informação, calculamos ele para cada uma das *features* em nosso dataset e comparamos os resultados para saber qual teve o maior ganho. Sendo assim, nossa ramificação será feita com base na coluna que apresentou o melhor ganho e, então, repetiremos o processo recursivamente para cada lado da ramificação, parando quando o ganho de informação ser igual a 0.

Photo by Ravi Roshan on Unsplash

Tamanho da Árvore e Overfitting

Nosso objetivo com uma Árvore de Decisão é chegar a uma entropia o mais baixa possível. Para isso, nossa árvore precisa crescer e ir formando diversas ramificações, o que é bom!.. até certo ponto.

Quanto maior a altura da nossa árvore, ou seja, quanto mais níveis horizontais de ramificações ela ter, mais adequado aos dados de treinamento nosso modelo estará. No final, se nossa árvore ter uma altura muito grande, isso pode acabar causando *overfitting* no nosso modelo, que consequentemente pode apresentar predições imprecisas com dados reais.

Demo em Python

Para a nossa demo de hoje, estaremos usando um dataset sobre vinhos do UCI. Nesse

Import das bibliotecas e leitura do dataset

Agora que podemos manipular nossos dados, vamos prepará-los para o treinamento. Nessa etapa, é importante normalizar os dados — deixá-los em uma mesma escala — para que nosso modelo não considere uma variável mais importante que outra. Além disso, devemos dividir nossos dados em um conjunto de treinamento e outro de validação para que possamos checar a precisão do modelo pós-treinamento.

scaling_train_test_split.ipynb hosted with ❤ by GitHub view raw

Separando o dataset em conjuntos de treino e teste e aplicando feature scaling

Com os dados preparados, o que falta fazer agora é o treinamento em si. Vamos criar uma função que, recebendo uma altura de árvore, treina o modelo com base nessa altura e então executar de 1 a 20.

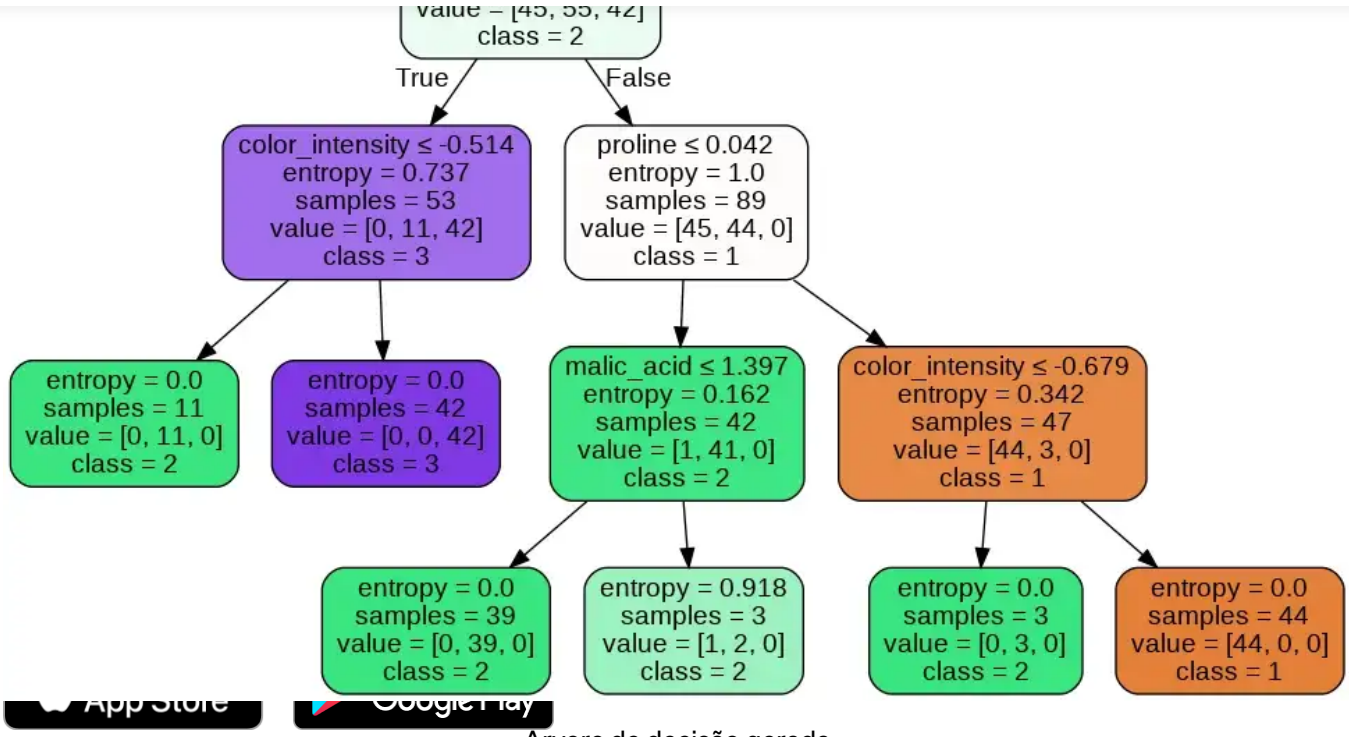
training_accuracy.ipynb hosted with ❤ by GitHub view raw

Treinamento dos dados em 20 alturas diferentes

No final, conseguimos uma precisão máxima de 94,44% ao testar o modelo com os dados de teste. Esse é um indicador excelente! Foi possível notar também que nossa precisão caiu para 91,66% em alturas maiores que três, não deixando de ser uma precisão ótima. Para exportarmos nossa árvore com os melhores resultados, usaremos o trecho de código abaixo:

tree.ipynb hosted with ❤ by **GitHub** **view raw**

Criação do arquivo contendo a imagem da árvore de decisão



Árvore de decisão gerada

A imagem acima representa a árvore de decisão do nosso modelo. Cada nó contém uma série de informações úteis como:

1. A condição da ramificação que gera os nós filhos(leve em conta que os valores continuam numa mesma escala), onde nós à esquerda atendem e os a direita não atendem essa condição;
2. O valor da entropia para os amostras do nó atual;
3. O número de amostras presente no nó atual;
4. A classe mais presente no nó atual.

Conclusão

Com esse tutorial, foi possível conhecermos um dos modelos de classificação mais práticos do mundo de Machine Learning. Caso você tenha se interessado por modelos de classificação e curtido o modelo de Árvore de Decisão em específico, recomendo dar uma olhada no modelo de Random Forest.

Referências

- Curso de Inteligência Artificial para todos do Professor Diogo Cortiz
- Artigo do KDnuggets sobre Decision Trees
- Artigo Decision Trees and Decision-making