Universidade Federal do Rio do Janeiro

Relatório: trabalho árvore de decisão

Prevendo Sobreviventes do Titanic

Aline Freire de Rezende
DRE: 116110571
Larissa Monteiro da Fonseca Galeno
DRE: 116083017

1. Código

Nosso código se encontra no Google Colab, no link.

2. Dataset

O dataset escolhido foi sobre os passageiros do Titanic (fonte). Nele, tivemos que apagar colunas de zeros e linhas sobre alguns passageiros, pois faltavam informações. Assim tivemos que transformar o arquivo CSV em excel para facilitar a edição. Havia algumas colunas cujas informações não conseguimos entender, então utilizamos uma outra fonte para ajudar na interpretação (Titanic - Machine Learning for Disaster).

Nele há as seguintes instâncias:

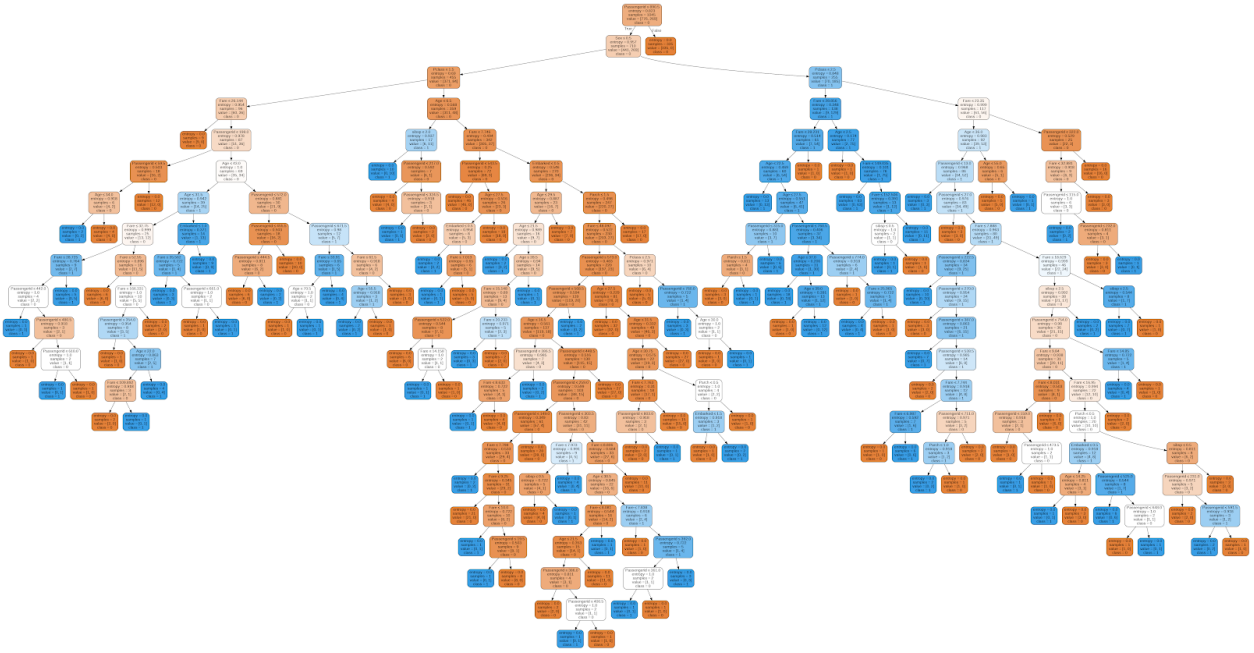
Variável	**Definição**	**Tipo do dado**
Passengerid	ID do passageiro	Número inteiro
Age	Idade em anos (se está estimado tem .5)	Número racional
Fare	Preço da passagem	Número racional
Sex	Gênero	0: Homens 1: Mulheres
Sibsip	Número de parente e/ou cônjuges embarcados	Número inteiro
Parch	Número de pais e/ou filhos embarcados	Número inteiro
Pclass	Classe do Ticket	1: Primeira classe 2: Segunda Classe 3: Terceira Classe
Embarked	Porto que embarcou	0: Cherbourg 1: Queenstown 2: Southampton
Survived	Se sobreviveu	0: Não 1: Sim

O objetivo é que nosso programa consiga determinar se uma pessoa sobreviverá ou não.

3. Experimento base

Em primeira instância, utilizamos todos os atributos no programa, para começar com a informação completa. Além disso, decidimos começar o experimento de forma simples: os parâmetros iniciais foram 80% treinamento - 20% teste, o atributo de construção da árvore foi entropia, e deixamos o shuffle em seu valor padrão, True.

Nesse caso base, obtivemos uma acurácia de 0.816793893129771, e a árvore de decisão foi:

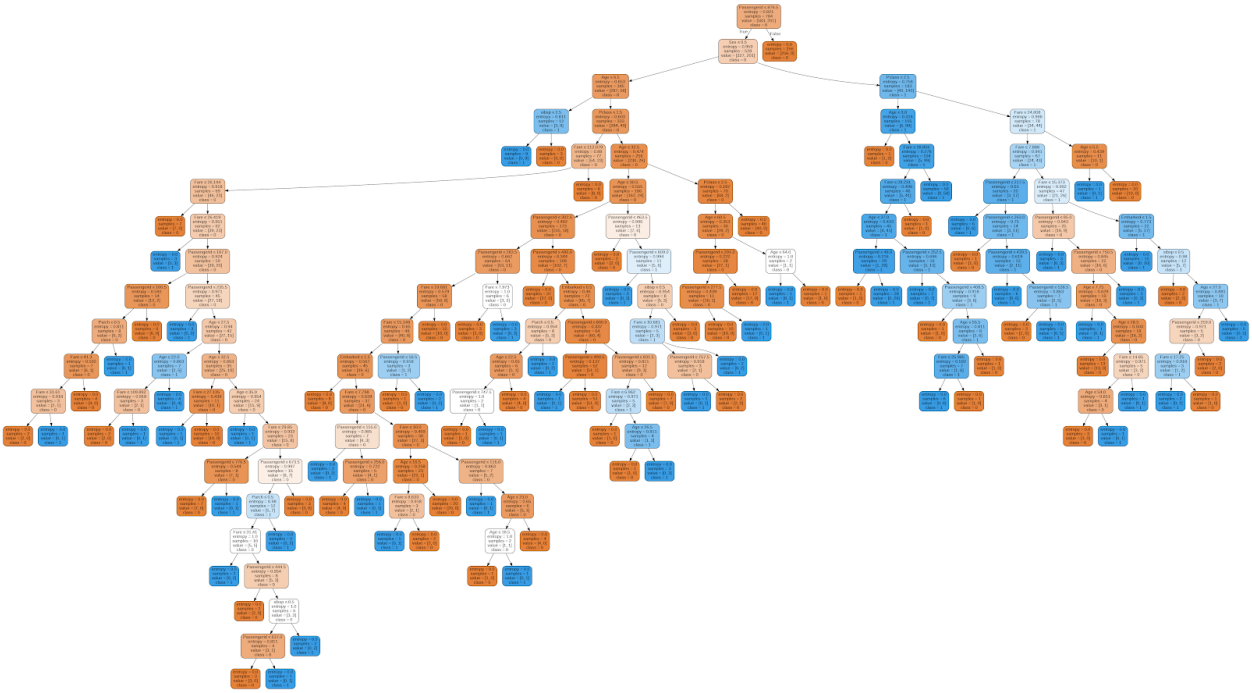


Com esse teste pudemos perceber que para um determinado caso o programa encontra a resposta rapidamente.

4. Testes

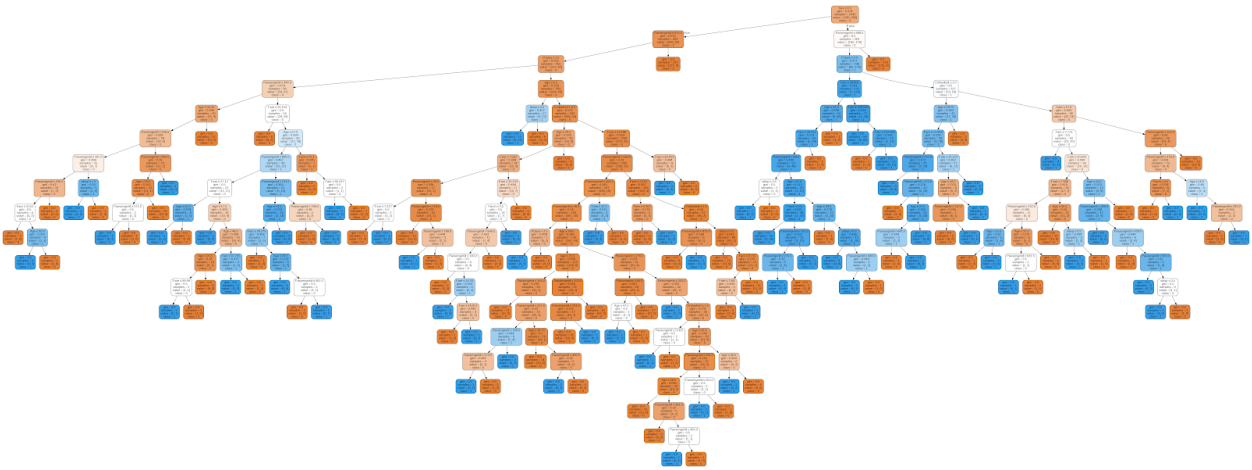
Começamos o teste mudando a divisão de treinamento e teste para entender como o programa iria se comportar com tamanhos diferentes de treino e teste:

- Alteramos o tamanho do treinamento/teste para 60% - 40%, mantivemos o atributo de construção da árvore em entropia, e com shuffle na divisão de treinamento e teste:
    - Acurácia: 0.8279158699808795;
    - Percebemos que não houve mudanças significativas;
    - Árvore:



Novamente, para tentar entender o comportamento do dataset, utilizamos como critério Gini ao invés da Entropia:

- Mantivemos o tamanho do treinamento/teste em 80% - 20%, alteramos o atributo de construção da árvore para gini, e com shuffle na divisão de treinamento e teste:
    - Acurácia: 0.7900763358778626;
    - Percebemos que houve uma pequena piora da acurácia, mas também não muito significativa;
    - Árvore:



Percebemos que na Entropia tinha um caso que a resposta era encontrada rapidamente (a uma distância da raiz), enquanto em Gini isso não ocorreu. Dessa forma, entendemos que a Entropia faz mais sentido para o caso que estamos trabalhando e seguimos os testes usando esse critério.

Assim, buscando aumentar a acurácia, trocamos o parâmetro de Shuffle, utilizando o valor “False” nos próximos testes.

- Mantivemos o tamanho do treinamento/teste em 80% - 20%, mantivemos o atributo de construção da árvore em entropia, e sem shuffle na divisão de treinamento e teste:
  - Acurácia: 1.0;
  - Houve 100% de acurácia;
  - Árvore: link para imagem
  - Por mais que tenha dado acurácia de 100% desconfiamos do resultado, pois a probabilidade disso ocorrer parece baixa.

- Alteramos o tamanho do treinamento/teste para 60% - 40%, mantivemos o atributo de construção da árvore em entropia, e sem shuffle na divisão de treinamento e teste:
  - Acurácia: 0.7208413001912046;
  - Percebemos que com a diminuição do treinamento, houve uma piora significativa em relação ao experimento base;
  - Árvore: link para imagem

Após os testes, introduzimos a divisão com K Fold esperando resultados melhores e mais consistentes em relação ao caso base, uma vez que ocorreriam diferentes testes e treinamentos. Mantivemos o parâmetro de shuffle do K Fold como True, pois acreditamos que dessa forma não teríamos um resultado “viciado”, mesmo podendo ter uma acurácia pior.

- Introduzimos k-fold com 200 folds, mantivemos o atributo de construção da árvore em entropia, e com shuffle na divisão de treinamento e teste:
  - Média da acurácia: 0.8266666666666665;
  - Percebemos que não houve mudanças significativas;

A fins de teste, colocamos o shuffle como False, esperando que houvesse uma melhora na média da acurácia. O resultado objetivo foi de 0.832142857142857, ou seja, não houve mudanças significativas.

5. Comparação com Kaggle

Vimos, ao comparar com outros resultados no Kaggle, que as acurácias não variavam muito em relação às nossas. Os métodos de cálculo foram variados, alguns usaram regressão ou RandomForestClassifier, por exemplo.

Nos casos em que testamos, nossa média de acurácia foi de 0.80. No Kaggle, encontramos os seguintes resultados:

- 0.71 neste caso;
- 0.80 neste caso;
- 0.89 neste caso;
- 0.82 neste caso;
- 0.80 neste caso;

6. Conclusão

Ao longo do trabalho, nos deparamos com algumas dificuldades:

- Não conseguimos entender como fazer a poda, usamos como fonte a documentação da biblioteca (fonte, fonte), porém não ficou claro como aplicar no código. Entretanto, acreditamos que para o nosso dataset a poda poderia dar resultados positivos.
- De acordo que estávamos testando a base vimos que pelo fato de termos mais pessoas que não sobreviveram (casos negativos) poderia gerar problemas para o aprendizado da máquina. Com isso, também ficamos na dúvida se a acurácia era a melhor forma de medir a performance do nosso código.
- Como nosso experimento base já deu uma acurácia alta ficamos com dificuldade de saber em qual parâmetro alterar para provocar alguma mudança expressiva na acurácia.

Apesar das dificuldades, o trabalho foi de grande proveito para entendermos como trabalhar com uma base maior, ter uma maior domínio das bibliotecas e conseguir interpretar alguns resultados.