0. Índice

1. Importando bibliotecas

Voltar ao índice

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.tree import plot_tree
from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix

from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
```

2. Carregando um dataframe

Voltar ao índice

```
titanic = sns.load_dataset('titanic')
titanic.head()
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adul
0	0	3	male	22.0	1	0	7.2500	S	Third	man	
1	1	1	female	38.0	1	0	71.2833	C	First	woman	
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	
3	1	1	female	35.0	1	0	53.1000	S	First	woman	
4	0	3	male	35.0	0	0	8.0500	S	Third	man	

```
titanic.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   survived     891 non-null    int64
 1   pclass       891 non-null    int64
 2   sex          891 non-null    object
 3   age          714 non-null    float64
 4   sibsp        891 non-null    int64
 5   parch        891 non-null    int64
 6   fare         891 non-null    float64
 7   embarked     889 non-null    object
 8   class        891 non-null    category
 9   who          891 non-null    object
 10  adult_male   891 non-null    bool
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object
 13  alive        891 non-null    object
 14  alone        891 non-null    bool
```

```
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
```

```
titanic.dtypes
```

```
survived          int64
pclass            int64
sex              object
age             float64
sibsp             int64
parch             int64
fare            float64
embarked         object
class          category
who              object
adult_male         bool
deck           category
embark_town      object
alive            object
alone              bool
dtype: object
```

3. Dados duplicados

Voltar ao índice

```
titanic.drop_duplicates()
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	ad
0	0	3	male	22.0	1	0	7.2500	S	Third	man	
1	1	1	female	38.0	1	0	71.2833	C	First	woman	
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	
3	1	1	female	35.0	1	0	53.1000	S	First	woman	
4	0	3	male	35.0	0	0	8.0500	S	Third	man	
...	...	...	...	...	...	...	...	...	...	...	
885	0	3	female	39.0	0	5	29.1250	Q	Third	woman	
887	1	1	female	19.0	0	0	30.0000	S	First	woman	
888	0	3	female	NaN	1	2	23.4500	S	Third	woman	
889	1	1	male	26.0	0	0	30.0000	C	First	man	
890	0	3	male	32.0	0	0	7.7500	Q	Third	man	

784 rows × 15 columns

```
titanic.shape
```

```
(891, 15)
```

```
titanic = titanic.drop_duplicates()
titanic.shape
```

```
(784, 15)
```

```
titanic.tail()
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adu
885	0	3	female	39.0	0	5	29.125	Q	Third	woman	
887	1	1	female	19.0	0	0	30.000	S	First	woman	
888	0	3	female	NaN	1	2	23.450	S	Third	woman	
889	1	1	male	26.0	0	0	30.000	C	First	man	
890	0	3	male	32.0	0	0	7.750	Q	Third	man	

```
titanic.reset_index(drop=True, inplace=True)
```

```
titanic.tail()
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adu
779	0	3	female	39.0	0	5	29.125	Q	Third	woman	
780	1	1	female	19.0	0	0	30.000	S	First	woman	
781	0	3	female	NaN	1	2	23.450	S	Third	woman	
782	1	1	male	26.0	0	0	30.000	C	First	man	
783	0	3	male	32.0	0	0	7.750	Q	Third	man	

4. Identificando e tratando dados ausentes

Voltar ao índice

https://scikit-learn.org/stable/modules/tree.html#id2

A árvore de decisão requer pouca preparação de dados. Outras técnicas geralmente requerem normalização de dados, variáveis fictícias precisam ser criadas e valores em branco precisam ser removidos. Observe, entretanto, que **este módulo não oferece suporte a valores ausentes**.

```
titanic.isna().sum()
```

```
survived         0
pclass           0
sex              0
age            106
sibsp            0
parch            0
fare             0
embarked         2
class            0
who              0
adult_male       0
deck           582
embark_town      2
alive            0
alone            0
dtype: int64
```

```
percentage = (titanic.isnull().sum() / len(titanic)) * 100
percentage
```

```
survived        0.000000
pclass          0.000000
sex             0.000000
age            13.520408
sibsp           0.000000
parch           0.000000
fare            0.000000
embarked        0.255102
class           0.000000
who             0.000000
adult_male      0.000000
deck           74.234694
embark_town     0.255102
alive           0.000000
alone           0.000000
dtype: float64
```

```
# dropar todas as colunas que tenha pelo menos 1 NA
titanic_sem_na = titanic.dropna(axis=1)
```

```
titanic_sem_na.head()
```

	survived	pclass	sex	sibsp	parch	fare	class	who	adult_male	alive	a
0	0	3	male	1	0	7.2500	Third	man	True	no	
1	1	1	female	1	0	71.2833	First	woman	False	yes	
2	1	3	female	0	0	7.9250	Third	woman	False	yes	
3	1	1	female	1	0	53.1000	First	woman	False	yes	
4	0	3	male	0	0	8.0500	Third	man	True	no	

```
titanic_sem_na.shape
```

```
(784, 11)
```

```
titanic_sem_na.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 784 entries, 0 to 783
Data columns (total 11 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   survived    784 non-null    int64
 1   pclass      784 non-null    int64
 2   sex         784 non-null    object
 3   sibsp       784 non-null    int64
 4   parch       784 non-null    int64
 5   fare        784 non-null    float64
 6   class       784 non-null    category
 7   who         784 non-null    object
 8   adult_male  784 non-null    bool
 9   alive       784 non-null    object
 10  alone       784 non-null    bool
dtypes: bool(2), category(1), float64(1), int64(4), object(3)
memory usage: 51.6+ KB
```

5. Dados categorizados

[Voltar ao índice]

https://scikit-learn.org/stable/modules/tree.html#id2

CART (árvores de classificação e regressão) é muito semelhante a C4.5, mas difere porque oferece suporte a variáveis de destino numéricas (regressão) e não calcula conjuntos de regras. A CART constrói árvores binárias usando o recurso e o limite que geram o maior ganho de informação em cada nó.

scikit-learn usa uma versão otimizada do algoritmo CART; entretanto, a **implementação do scikit-learn não suporta variáveis categorizadas** por enquanto.

```
titanic_sem_na.head(2)
```

	survived	pclass	sex	sibsp	parch	fare	class	who	adult_male	alive	a
0	0	3	male	1	0	7.2500	Third	man	True	no	
1	1	1	female	1	0	71.2833	First	woman	False	yes	

- **survived** - se o passageiro sobreviveu ou não, ou seja, nossa target
- **pclass** - classe em que o passageiro estava (primeira, segunda, terceira)
- **sex** - genero do passageiro (masculino ou feminino)
- **sibsp** - quantidade de irmãos/esposos/esposas no navio (0 a 8)
- **parch** - quantidade de pais/filhos a bordo
- **fare** - preço do ticket
- **class** - igual a pclass
- **who** - se é homem, mulher ou criança
- **adult_male** - se é um homem adulto
- **alive** - igual a survived
- **alone** - se estava sozinho a bordo

Survived e alive

```
titanic_sem_na.survived.value_counts()
```

```
survived
0    461
1    323
Name: count, dtype: int64
```

```
titanic_sem_na.survived.value_counts(normalize=True)
```

```
survived
0    0.58801
```

```
1    0.41199
Name: proportion, dtype: float64
```

```
titanic_sem_na.alive.value_counts()
```

```
alive
no     461
yes    323
Name: count, dtype: int64
```

```
titanic_sem_na[['alive','survived','sibsp']].groupby(['alive','survived']).count()
```

alive	survived	sibsp
no	0	461
yes	1	323

```
titanic_sem_na = titanic_sem_na.drop('alive',axis=1)
```

pclass e class

```
titanic_sem_na['pclass'].unique()
```

```
array([3, 1, 2], dtype=int64)
```

```
titanic_sem_na['class'].unique()
```

```
['Third', 'First', 'Second']
Categories (3, object): ['First', 'Second', 'Third']
```

```
titanic_sem_na[['pclass','class','sibsp']].groupby(['pclass','class']).count()
```

```
C:\Users\Soldado\AppData\Local\Temp\ipykernel_28572\2776273745.py:1: FutureWarning: T
  titanic_sem_na[['pclass','class','sibsp']].groupby(['pclass','class']).count()
```

pclass	class	sibsp
1	First	214
	Second	0
	Third	0
2	First	0
	Second	165
	Third	0
3	First	0
	Second	0
	Third	405

```
titanic_sem_na = titanic_sem_na.drop('pclass',axis=1)
```

Outras variáveis

```
titanic_sem_na['sex'].unique()
```

```
array(['male', 'female'], dtype=object)
```

```
titanic_sem_na['sibsp'].unique()
```

```
array([1, 0, 3, 4, 2, 5, 8], dtype=int64)
```

```
titanic_sem_na['parch'].unique()
```

```
array([0, 1, 2, 5, 3, 4, 6], dtype=int64)
```

```
titanic_sem_na['who'].unique()
```

```
array(['man', 'woman', 'child'], dtype=object)
```

```
titanic_sem_na[['sex','who','sibsp']].groupby(['sex','who']).count()
```

sex	who	sibsp
female	child	42
	woman	251
male	child	40
	man	451

```
titanic_sem_na.head(2)
```

	survived	sex	sibsp	parch	fare	class	who	adult_male	alone
0	0	male	1	0	7.2500	Third	man	True	False
1	1	female	1	0	71.2833	First	woman	False	False

Transformando em dummie (flag)

```
# titanic_encoded = titanic_sem_na.copy()
titanic_encoded = pd.get_dummies(titanic_sem_na, columns=['class','who'], drop_first=True)
titanic_encoded.head(20)
```

	survived	sex	sibsp	parch	fare	adult_male	alone	class_Second	class_Th
0	0	male	1	0	7.2500	True	False	False	T
1	1	female	1	0	71.2833	False	False	False	Fa
2	1	female	0	0	7.9250	False	True	False	T
3	1	female	1	0	53.1000	False	False	False	Fa
4	0	male	0	0	8.0500	True	True	False	T
5	0	male	0	0	8.4583	True	True	False	T
6	0	male	0	0	51.8625	True	True	False	Fa
7	0	male	3	1	21.0750	False	False	False	T
8	1	female	0	2	11.1333	False	False	False	T
9	1	female	1	0	30.0708	False	False	True	Fa
10	1	female	1	1	16.7000	False	False	False	T
11	1	female	0	0	26.5500	False	True	False	Fa
12	0	male	0	0	8.0500	True	True	False	T
13	0	male	1	5	31.2750	True	False	False	T
14	0	female	0	0	7.8542	False	True	False	T
15	1	female	0	0	16.0000	False	True	True	Fa
16	0	male	4	1	29.1250	False	False	False	T
17	1	male	0	0	13.0000	True	True	True	Fa
18	0	female	1	0	18.0000	False	False	False	T
19	1	female	0	0	7.2250	False	True	False	T

Mapping

```
titanic_encoded.sex.unique()
```

```
array(['male', 'female'], dtype=object)
```

```
titanic_encoded.sex = titanic_encoded.sex.map({'female': 1, 'male':0})
```

```
titanic_encoded.sex.unique()
```

```
array([0, 1], dtype=int64)
```

Mudando alguns tipos de dados

```
titanic_encoded.dtypes
```

```
survived          int64
sex               int64
sibsp             int64
parch             int64
fare            float64
adult_male         bool
alone              bool
class_Second       bool
class_Third        bool
who_man            bool
who_woman          bool
dtype: object
```

```
titanic_encoded.adult_male.astype(int)
```

```
0      1
1      0
2      0
3      0
4      1
      ..
779    0
780    0
781    0
782    1
783    1
Name: adult_male, Length: 784, dtype: int32
```

```
titanic_encoded.adult_male = titanic_encoded.adult_male.astype(int)
titanic_encoded.alone = titanic_encoded.alone.astype(int)
```

```
titanic_encoded.dtypes
```

```
survived          int64
sex               int64
sibsp             int64
parch             int64
fare            float64
adult_male        int32
alone             int32
class_Second       bool
class_Third        bool
who_man            bool
who_woman          bool
dtype: object
```

```
titanic_encoded.columns
```

```
Index(['survived', 'sex', 'sibsp', 'parch', 'fare', 'adult_male', 'alone',
       'class_Second', 'class_Third', 'who_man', 'who_woman'],
      dtype='object')
```

6. Separando as variáveis explicativas da target

Voltar ao índice

```
y = titanic_encoded.survived
```

```
X = titanic_encoded.drop('survived',axis=1)
```

7. Árvore de classificação com todas as variáveis

Voltar ao índice

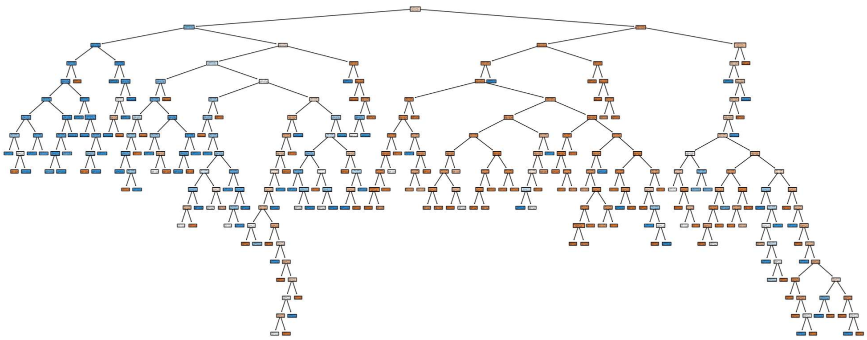
```
clf_dt = DecisionTreeClassifier(random_state=100)
clf_dt
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier(random_state=100)
```

```
clf_dt = clf_dt.fit(X,y)
clf_dt
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier(random_state=100)
```

```
plt.figure(figsize=(25, 10))
plot_tree(clf_dt,
          filled=True,
          class_names=['Died', 'Survived'],
          feature_names=X.columns);
```



```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=100)
```

```
# Fazendo a previsão e avaliando o erro
y_pred = clf_dt.predict(X_test)

from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,y_pred)
```

```
array([[138,   6],
       [ 14,  78]], dtype=int64)
```

```
predict = clf_dt.predict(X)
```

```
accuracy_score(y, predict)
```

```
0.9170918367346939
```

8. Separando entre treino e teste

Voltar ao índice

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=100)
```

```
clf = DecisionTreeClassifier(random_state=100)
clf = clf.fit(X_train,y_train)
```

```
y_chapeu_teste = clf.predict(X_test)
y_chapeu_teste
```

```
array([1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1,
       1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0,
       1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0,
       0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1,
       0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1,
       1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1,
       1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0,
       0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0,
       1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=int64)
```

```
y_pred = clf.predict(X_train)
confusion_matrix(y_train,y_pred)
```

```
array([[306,  11],
       [ 24, 207]], dtype=int64)
```

```
y_pred = clf_dt.predict(X_test)
confusion_matrix(y_test,y_pred)
```

```
array([[138,   6],
       [ 14,  78]], dtype=int64)
```

9. Pre pruning

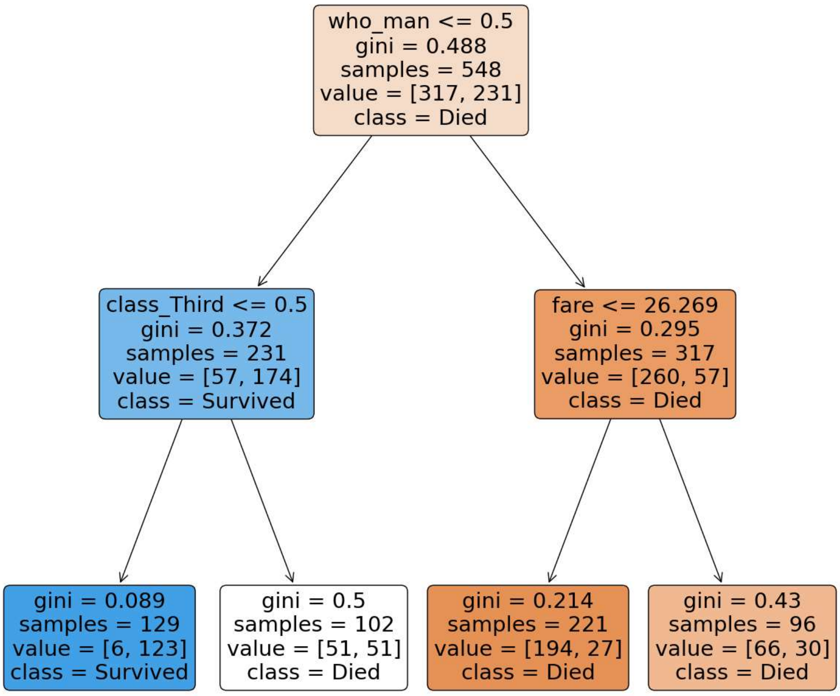
Voltar ao índice

Profundidade

```
clf = DecisionTreeClassifier(max_depth=2 ,random_state=100)
clf = clf.fit(X_train,y_train)
predict = clf.predict(X_test)
accuracy_score(y_test, predict)
```

```
0.7923728813559322
```

```
plt.figure(figsize=[15,15])
plot_tree(clf,
         filled=True,
         rounded=True,
         class_names=['Died', 'Survived'],
         feature_names=X_train.columns);
```

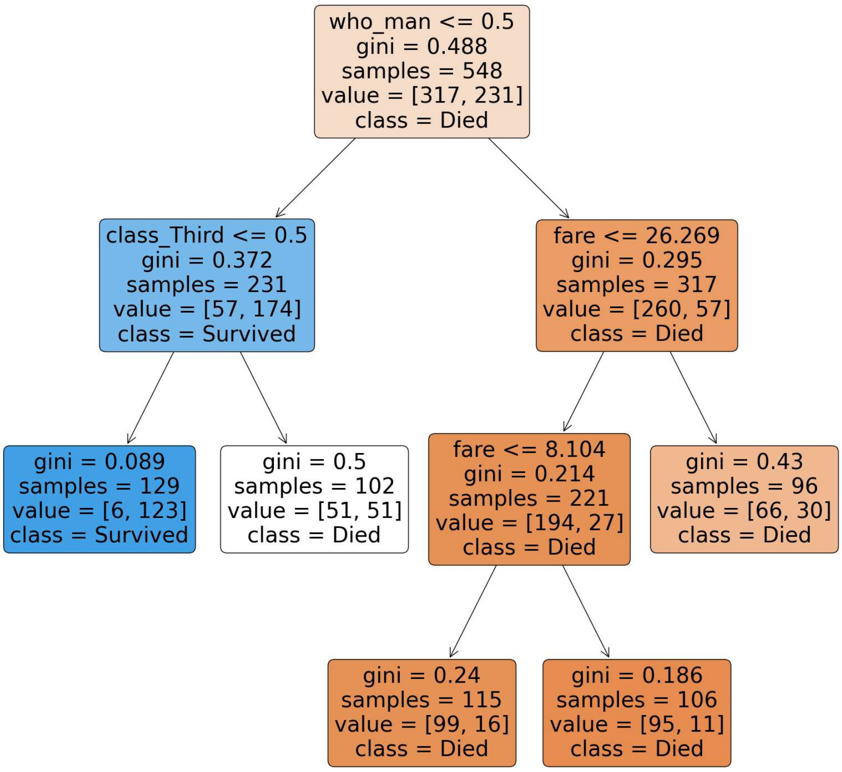


Amostras na folha

```
clf = DecisionTreeClassifier(min_samples_leaf=80, random_state=100)
clf = clf.fit(X_train,y_train)
predict = clf.predict(X_test)
accuracy_score(y_test, predict)
```

```
0.7923728813559322
```

```
plt.figure(figsize=[20,20])
plot_tree(clf,
         filled=True,
         rounded=True,
         class_names=['Died', 'Survived'],
         feature_names=X_train.columns);
```

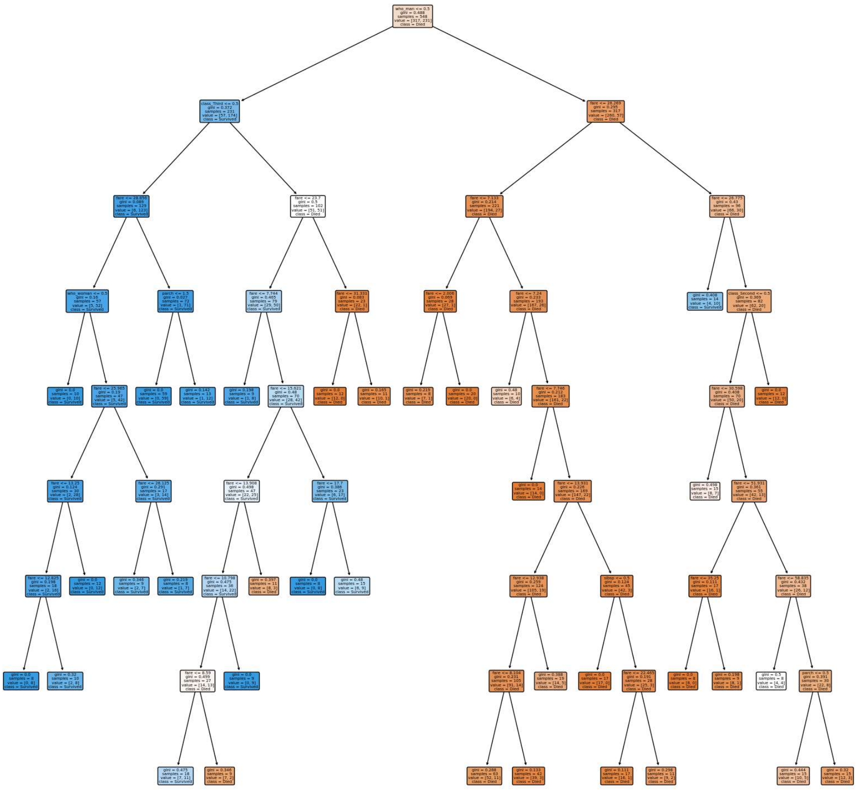

who_man <= 0.5
gini = 0.488
samples = 548
value = [317, 231]
class = Died
class_Third <= 0.5
gini = 0.372
samples = 231
value = [57, 174]
class = Survived
fare <= 26.269
gini = 0.295
samples = 317
value = [260, 57]
class = Died
gini = 0.089
samples = 129
value = [6, 123]
class = Survived
gini = 0.5
samples = 102
value = [51, 51]
class = Died
fare <= 8.104
gini = 0.214
samples = 221
value = [194, 27]
class = Died
gini = 0.43
samples = 96
value = [66, 30]
class = Died
gini = 0.24
samples = 115
value = [99, 16]
class = Died
gini = 0.186
samples = 106
value = [95, 11]
class = Died

Amostras na folha e profundidade

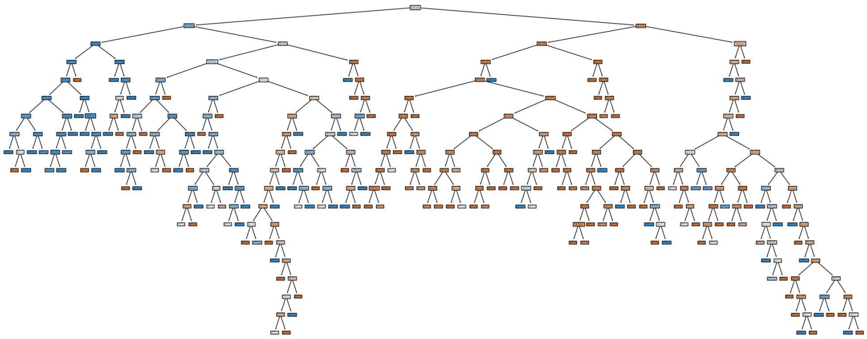
```
clf = DecisionTreeClassifier(max_depth=8, min_samples_leaf=8, random_state=100)
clf = clf.fit(X_train,y_train)
predict = clf.predict(X_test)
accuracy_score(y_test, predict)
```

```
0.7754237288135594
```

```
plt.figure(figsize=[20,20])
plot_tree(clf,
        filled=True,
        rounded=True,
        class_names=['Died', 'Survived'],
        feature_names=X_train.columns);
```



```
plt.figure(figsize=(25, 10))
plot_tree(clf_dt,
          filled=True,
          class_names=['Died', 'Survived'],
          feature_names=X.columns);
```



10. Post pruning

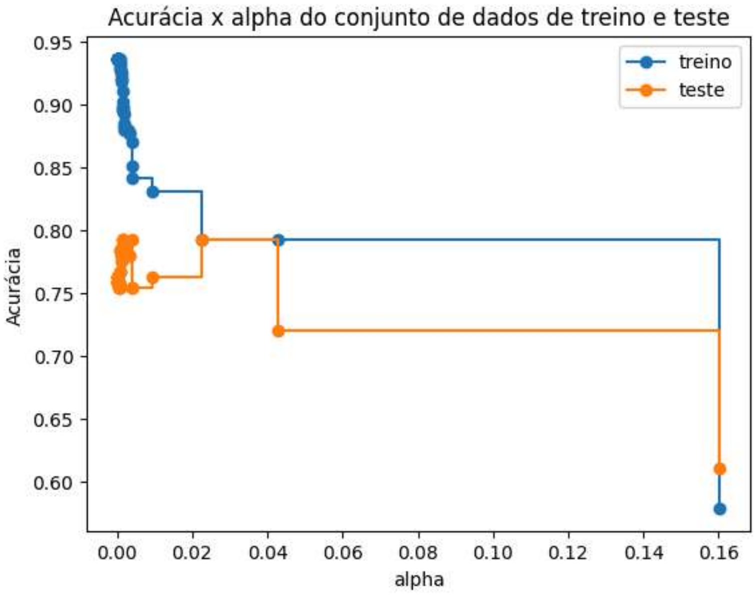
https://scikit-learn.org/stable/auto_examples/tree/plot_cost_complexity_pruning.html

```
clf = DecisionTreeClassifier(random_state=100)
path = clf.cost_complexity_pruning_path(X_train, y_train)
ccp_alphas, impurities = path.ccp_alphas, path.impurities
```

```
clfs = []
for ccp_alpha in ccp_alphas:
    clf = DecisionTreeClassifier(random_state=0, ccp_alpha=ccp_alpha)
    clf.fit(X_train, y_train)
    clfs.append(clf)
```

```
train_scores = [clf.score(X_train, y_train) for clf in clfs]
test_scores = [clf.score(X_test, y_test) for clf in clfs]

fig, ax = plt.subplots()
ax.set_xlabel("alpha")
ax.set_ylabel("Acurácia")
ax.set_title("Acurácia x alpha do conjunto de dados de treino e teste")
ax.plot(ccp_alphas, train_scores, marker='o', label="treino",
        drawstyle="steps-post")
ax.plot(ccp_alphas, test_scores, marker='o', label="teste",
        drawstyle="steps-post")
ax.legend()
plt.show()
```




```
pd.DataFrame({'alpha': ccp_alphas.tolist(), 'score': test_scores})
```