

Ávore de Decisão

Mini-projeto 2 - Sistemas Inteligentes

Prof. Germano Crispim Vasconcelos

Equipe 6

- Alice Oliveira - aoqb
- Ana Albuquerque - alas3
- Enrique Laborão - elm2
- Victória Luisi - vlsc
- Olivia Tavares - obtmc

Setup

Importando dependências, conectando Google Drive, lendo datasets e criando os respectivos dataframes.

Dependências

In []:

```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import classification_report, accuracy_score, precision_score, recall_score, f1_score, roc_auc_score, confusion_matrix
```

Drive

In []:

```
from google.colab import drive
drive.mount('/content/drive')

base_path = '/content/drive/MyDrive/'
```

Mounted at /content/drive

Titanic Dataset - <https://www.kaggle.com/c/titanic/data> (<https://www.kaggle.com/c/titanic/data>)

In []:

```
titanic_train_dataset_path = base_path + 'titanic/train.csv'
titanic_test_dataset_path = base_path + 'titanic/test.csv'
```

Lower Back Pain Symptoms Dataset - <https://www.kaggle.com/datasets/sammy123/lower-back-pain-symptoms-dataset> (<https://www.kaggle.com/datasets/sammy123/lower-back-pain-symptoms-dataset>)

In []:

```
pain_dataset_path = base_path + 'Dataset_spine.csv'
```

Lendo datasets

In []:

```
titanic_df = pd.read_csv(titanic_train_dataset_path)
titanic_test_df = pd.read_csv(titanic_test_dataset_path) # Não possui a coluna de classificação (Survived)
```

In []:

```
pain_df = pd.read_csv(pain_dataset_path)
pain_df = pain_df.drop(pain_df.columns[[13]], axis=1) # Removendo a coluna 13
```

Funções

In []:

```
def predict_and_report(title, classifier, x_test, y_test, y_train, x_train):
    y_pred = classifier.predict(x_test)
    report = classification_report(y_test, y_pred)
    print(title + "- base de teste")
    print(report)
    plot_confusion_matrix(y_test, y_pred)
    print("-----")
    y_pred = classifier.predict(x_train)
    report = classification_report(y_train, y_pred)
    print(title + "- base de treino")
    print(report)
```

In []:

```
def plot_confusion_matrix(y_test, y_pred):
    cm = confusion_matrix(y_test, y_pred)
    fig, ax = plt.subplots(figsize=(2,2))
    sns.heatmap(cm, annot=True, fmt='d')
```



Projeto: Titanic Dataset



Análise e limpeza do Dataset

Valores faltantes

A primeira análise feita é verificar os valores faltantes. **Três colunas possuem valores nulos: Age, Cabin e Embarked.** Cada coluna terá um tratamento apropriado.

In []:

```
titanic_df.shape
```

Out[]:

```
(891, 12)
```

In []:

```
#titanic_df.info()
titanic_df.isnull().sum()
```

Out[]:

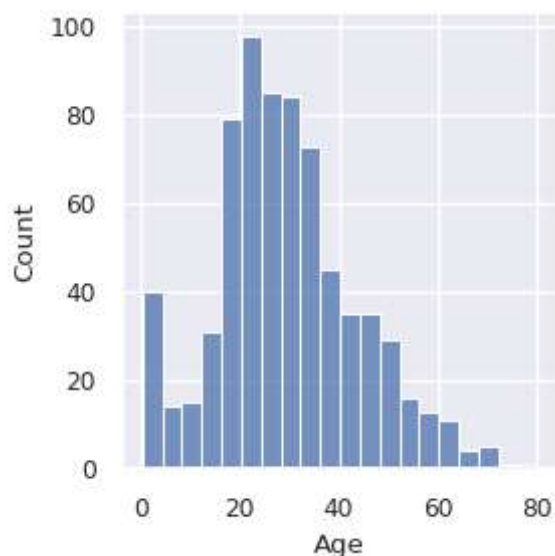
```
PassengerId      0
Survived          0
Pclass           0
Name             0
Sex              0
Age             177
SibSp            0
Parch           0
Ticket           0
Fare             0
Cabin           687
Embarked         2
dtype: int64
```

Age

- Como é uma coluna de valores numéricos, que a maioria dos valores não é nulo e que acreditamos ter grande relevância para a sobrevivência ou não, será utilizado um método de substituir os valores nulos por um outro valor.
- No plot vemos que se aproxima de uma distribuição assimétrica para a direita, então, então podemos **substituir os valores faltantes pela mediana**. [1](https://vitalflux.com/pandas-impute-missing-values-mean-median-mode/#How_to_decide_which_imputation_technique_to_use)
- **▶ Idealmente, deveríamos analisar a correlação entre a idade e outras features, observando a existência de uma correlação, então os valores nulos deveriam ser substituídos pela mediana do subconjunto (separado pelas outras features correlacionadas).** Não fizemos isso por não achar necessário para o escopo do projeto.

In []:

```
sns.set(font_scale=0.8)
sns.displot(titanic_df['Age'], height=3)
plt.show()
```



In []:

```
titanic_df['Age'] = titanic_df['Age'].fillna(titanic_df['Age'].median())
```

Cabin

- A maioria dos valores são nulos (687 de 891), não são numéricos e possuem muitos valores únicos.
- Poderia ser feita uma tentativa de extrair informações relevantes dos dados que existem, visto que as strings consistem de uma letra seguida por um ou dois números, podemos deduzir que está relacionada a posição da cabine no navio, podendo então influenciar na sobrevivência de um passageiro. [2] (<https://www.kaggle.com/code/ccastleberry/titanic-cabin-features/notebook>)
- Entretanto, optaremos por **descartar a coluna** devido a quantidade de valores faltantes.

In []:

```
titanic_df['Cabin'].value_counts()
```

Out[]:

```
B96 B98      4
G6           4
C23 C25 C27  4
C22 C26      3
F33          3
..
E34          1
C7           1
C54          1
E36          1
C148         1
Name: Cabin, Length: 147, dtype: int64
```

In []:

```
titanic_df = titanic_df.drop(columns=['Cabin'])
```

Embarked

- Como se tratam de valores categóricos, os dois valores faltantes serão substituídos pela moda.

In []:

```
titanic_df['Embarked'].value_counts()
```

Out[]:

```
S    644
C    168
Q     77
Name: Embarked, dtype: int64
```

In []:

```
titanic_df['Embarked'] = titanic_df['Embarked'].fillna(titanic_df['Embarked'].mode()[0])
```

In []:

```
titanic_df['Embarked'].value_counts()
```

Out[]:

```
S    646
C    168
Q     77
Name: Embarked, dtype: int64
```

Colunas consideradas irrelevantes

Algumas colunas possuem informações irrelevantes: identificadores que não apresentam relação direta com algo que possa definir a sobrevivência ou são valores aleatórios:

- PassengerId
- Ticket

In []:

```
titanic_df = titanic_df.drop(columns=['PassengerId'])
titanic_df = titanic_df.drop(columns=['Ticket'])
```

Substituição dos valores categóricos por numéricos

Name

- Essa coluna é composta apenas de valores únicos, mas todos os indivíduos possuem títulos (Mr., Mrs., Miss., etc).
- É possível extrair a informação do título do passageiro e criar uma nova coluna.

In []:

```
titanic_df['Name'].unique().size
```

Out[]:

891

In []:

```
titles = titanic_df.Name.str.extract(' ([A-Za-z]+)\.', expand=False)
print(titles.size)
titles.value_counts()
```

891

Out[]:

```
Mr          517
Miss        182
Mrs         125
Master       40
Dr           7
Rev          6
Mlle         2
Major        2
Col          2
Countess     1
Capt        1
Ms           1
Sir          1
Lady         1
Mme          1
Don          1
Jonkheer     1
Name: Name, dtype: int64
```

- Como Mr, Miss, Mrs e Master são os únicos títulos que se repetem mais de dez vezes, vamos optar por substituir o restante por 'Other'. E outros títulos que equivalem a esses títulos também serão substituídos.
- O seguinte mapeamento será feito:
 - Mr = 0, Mrs = 1, Miss = 2, Master = 3, Other = 4

In []:

```
titles = titles.replace(['Lady', 'Countess', 'Capt', 'Col', 'Don', 'Dr', 'Major', 'Rev',  
'Sir', 'Jonkheer', 'Dona'], 'Other')  
titles = titles.replace('Mlle', 'Miss')  
titles = titles.replace('Ms', 'Miss')  
titles = titles.replace('Mme', 'Mrs')  
titles = titles.map({"Mr": 0, "Mrs": 1, "Miss": 2, "Master": 3, "Other": 4})  
titles.value_counts()
```

Out[]:

```
0    517  
2    185  
1    126  
3     40  
4     23  
Name: Name, dtype: int64
```

In []:

```
titanic_df = titanic_df.drop(columns=['Name'])
```

In []:

```
titanic_df['Title'] = titles
```

Outras colunas

In []:

```
titanic_df['Sex'].unique()
```

Out[]:

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

In []:

```
titanic_df['Embarked'].unique()
```

Out[]:

```
array(['S', 'C', 'Q'], dtype=object)
```

Antes de treinar o modelo, é necessário converter algumas colunas de uma string para um valor numérico.

- Sex (male = 0, female = 1)
- Embarked (S = 0, C = 1, Q = 2)

In []:

```
titanic_df['Sex'] = titanic_df['Sex'].map({'male': 0, 'female': 1})
```

In []:

```
titanic_df['Embarked'] = titanic_df['Embarked'].map({'S': 0, 'C': 1, 'Q': 2})
```

Combinar colunas

- Podemos combinar as colunas Sibsp e Parch para ter uma coluna representando o tamanho da família.
- Fizemos alguns testes combinando as colunas e não houve melhoria significativa nos scores.

In []:

```
#family = titanic_df['SibSp'] + titanic_df['Parch']  
#titanic_df['Family'] = family
```

In []:

```
#titanic_df = titanic_df.drop(columns=['SibSp', 'Parch'])
```

Dataset após limpeza

In []:

```
titanic_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 891 entries, 0 to 890  
Data columns (total 9 columns):  
#   Column      Non-Null Count  Dtype  
---  -  
0   Survived    891 non-null    int64  
1   Pclass      891 non-null    int64  
2   Sex         891 non-null    int64  
3   Age        891 non-null    float64  
4   SibSp       891 non-null    int64  
5   Parch       891 non-null    int64  
6   Fare        891 non-null    float64  
7   Embarked    891 non-null    int64  
8   Title       891 non-null    int64  
dtypes: float64(2), int64(7)  
memory usage: 62.8 KB
```

Criando modelo

In []:

```
titanic_x = titanic_df.drop(columns=['Survived'])  
titanic_y = titanic_df['Survived']  
  
titanic_x_train, titanic_x_test, titanic_y_train, titanic_y_test = train_test_split(titanic_x, titanic_y, test_size = 0.2)
```

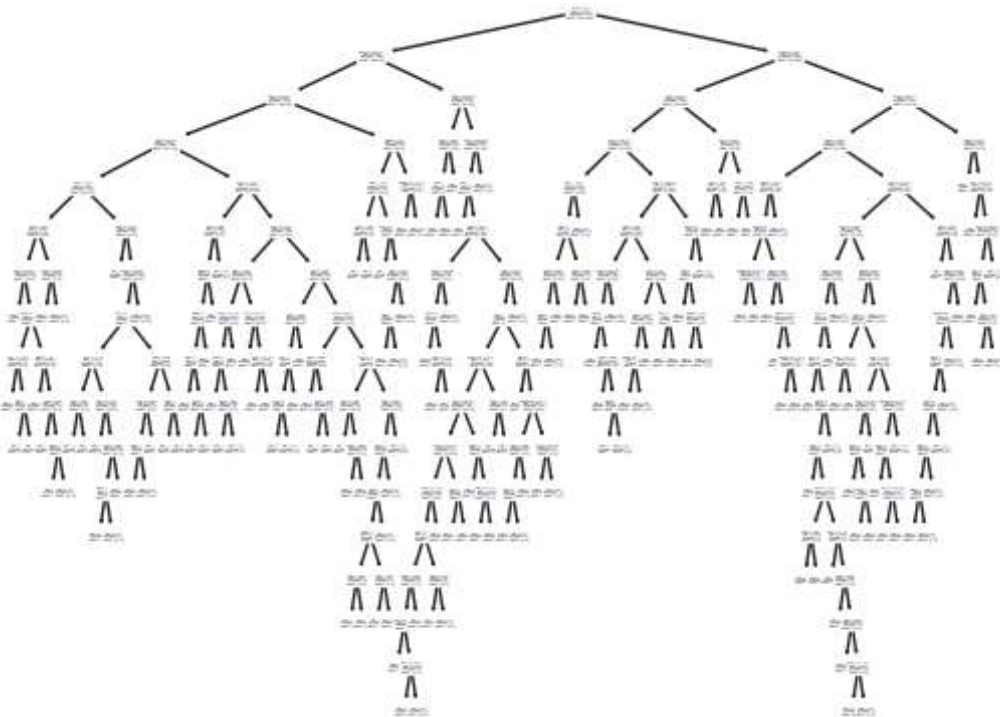
1. Classificador com parâmetros default

In []:

```
classifier1 = DecisionTreeClassifier()  
classifier1.fit(titanic_x_train, titanic_y_train)
```


In []:

```
plot_tree(classifier1, feature_names=titanic_df.drop(columns=['Survived']).columns)
plt.show()
```



In []:

```
titanic_y_pred = classifier1.predict(titanic_x_test)
titanic_report = classification_report(titanic_y_test, titanic_y_pred)
print(titanic_report)
```

	precision	recall	f1-score	support
0	0.79	0.85	0.82	105
1	0.76	0.68	0.71	74
accuracy			0.78	179
macro avg	0.77	0.76	0.77	179
weighted avg	0.78	0.78	0.77	179

2. Utilizando GridSearchCV

In []:

```
parameters1 = {
    'criterion': ["gini", "entropy", "log_loss"],
    'splitter': ["best", "random"],
    'max_depth': [3, 10, 30, None],
    'max_features': ["sqrt", "log2", None],
    'min_samples_split': [2, 16, 32],
    'max_leaf_nodes': [5, 10, 40, None],
    'random_state': [13]
}
```

In []:

```
grid = GridSearchCV(estimator = DecisionTreeClassifier(), param_grid = parameters1)
grid.fit(titanic_x_train, titanic_y_train)
```

In []:

```
print(grid.best_score_)
print(grid.best_params_)
```

0.8385403329065302

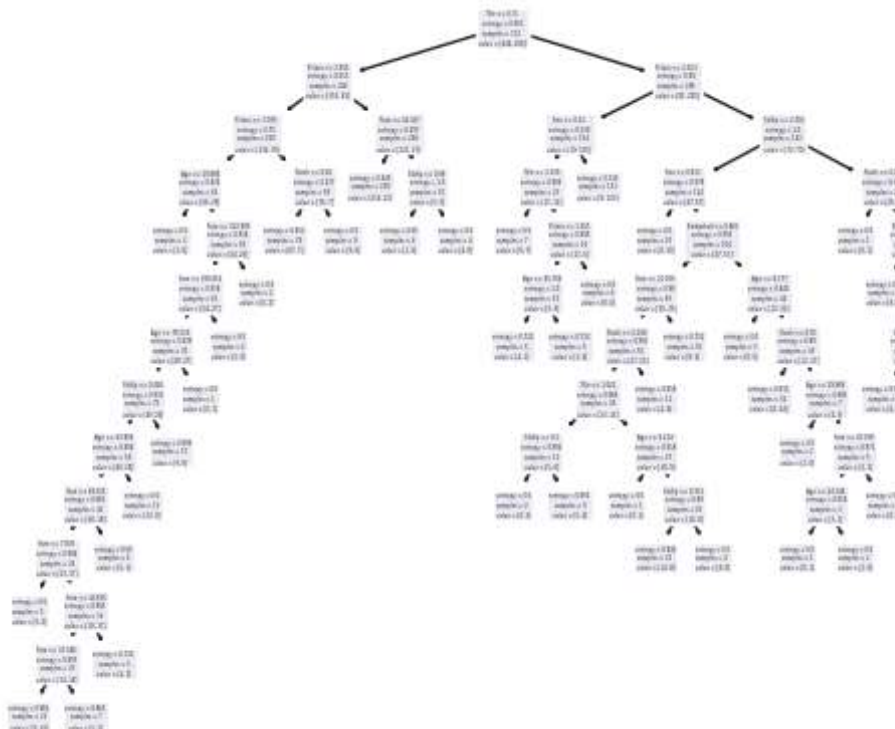
```
{'criterion': 'entropy', 'max_depth': 30, 'max_features': None, 'max_leaf_nodes': 40, 'min_samples_split': 2, 'random_state': 13, 'splitter': 'random'}
```

In []:

```
classifier2 = DecisionTreeClassifier(criterion=grid.best_params_['criterion'],
                                    max_depth=grid.best_params_['max_depth'],
                                    max_features=grid.best_params_['max_features'],
                                    max_leaf_nodes=grid.best_params_['max_leaf_nodes'],
                                    min_samples_split=grid.best_params_['min_samples_split'],
                                    random_state=grid.best_params_['random_state'],
                                    splitter=grid.best_params_['splitter'])
classifier2.fit(titanic_x_train, titanic_y_train)
```

In []:

```
plot_tree(classifier2, feature_names=titanic_df.drop(columns=['Survived']).columns)
plt.show()
```



In []:

```
titanic_y_pred = classifier2.predict(titanic_x_test)
titanic_report = classification_report(titanic_y_test, titanic_y_pred)
print(titanic_report)
```

	precision	recall	f1-score	support
0	0.82	0.90	0.85	105
1	0.83	0.72	0.77	74
accuracy			0.82	179
macro avg	0.82	0.81	0.81	179
weighted avg	0.82	0.82	0.82	179

3. Tentando obter uma árvore menor com score similar

In []:

```
parameters2 = {
    'criterion': ["gini", "entropy", "log_loss"],
    'splitter': ["best", "random"],
    'max_depth': [2, 4, 6],
    'max_features': ["sqrt", "log2", None],
    'min_samples_split': [2, 4, 8],
    'max_leaf_nodes': [4, 7, 10],
    'random_state': [13]
}
```

In []:

```
grid2 = GridSearchCV(estimator = DecisionTreeClassifier(), param_grid = parameters2)
grid2.fit(titanic_x_train, titanic_y_train)
```

In []:

```
print(grid2.best_score_)
print(grid2.best_params_)
```

0.8357234314980794

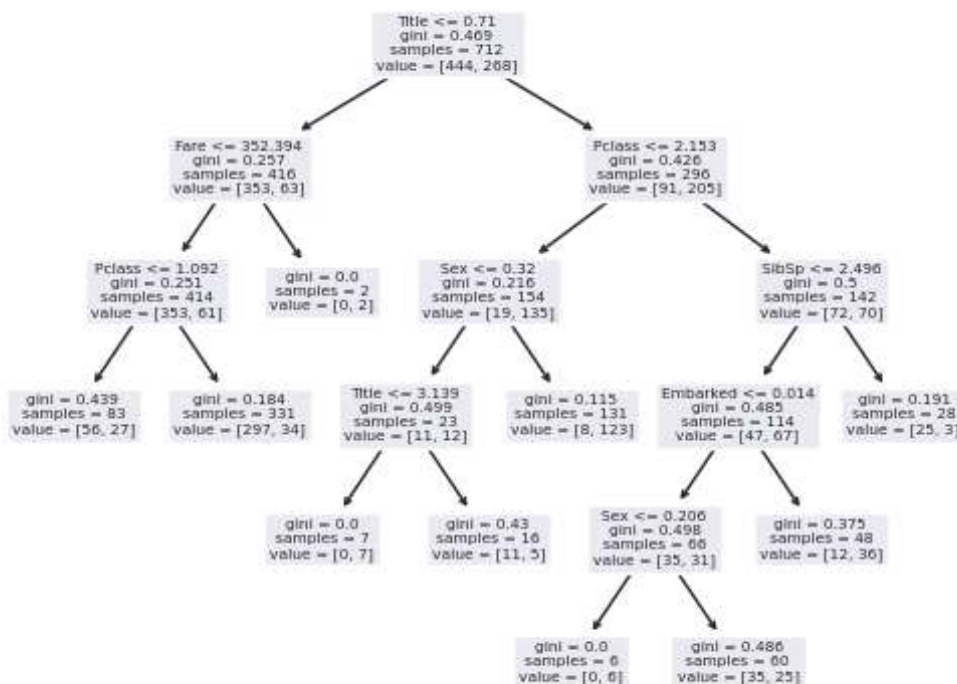
```
{'criterion': 'gini', 'max_depth': 6, 'max_features': None, 'max_leaf_node
s': 10, 'min_samples_split': 2, 'random_state': 13, 'splitter': 'random'}
```

In []:

```
classifier3 = DecisionTreeClassifier(criterion=grid2.best_params_['criterion'],
                                     max_depth=grid2.best_params_['max_depth'],
                                     max_features=grid2.best_params_['max_features'],
                                     max_leaf_nodes=grid2.best_params_['max_leaf_node
s'],
                                     min_samples_split=grid2.best_params_['min_samples_s
plit'],
                                     random_state=grid2.best_params_['random_state'],
                                     splitter=grid2.best_params_['splitter'])
classifier3.fit(titanic_x_train, titanic_y_train)
```

In []:

```
plot_tree(classifier3, feature_names=titanic_df.drop(columns=['Survived']).columns)
plt.show()
```



In []:

```
titanic_y_pred = classifier3.predict(titanic_x_test)
titanic_report = classification_report(titanic_y_test, titanic_y_pred)
print(titanic_report)
```

	precision	recall	f1-score	support
0	0.80	0.94	0.87	105
1	0.89	0.68	0.77	74
accuracy			0.83	179
macro avg	0.85	0.81	0.82	179
weighted avg	0.84	0.83	0.83	179

Utilizando Random Forests

Aplicando o RandomForestClassifier com a configuração default no problema Titanic.

In []:

```
parameters3 = {
    'criterion': ["gini", "entropy", "log_loss"],
    'n_estimators': [10, 50, 100], # Novo
    'max_depth': [2, 4, 6],
    'max_features': ["sqrt", "log2", None],
    'min_samples_split': [2, 4, 8],
    'max_leaf_nodes': [4, 7, 10],
    'random_state': [13]
}
```

In []:

```
grid3 = GridSearchCV(estimator = RandomForestClassifier(), param_grid = parameters3)
grid3.fit(titanic_x_train, titanic_y_train)
```

In []:

```
print(grid3.best_score_)
print(grid3.best_params_)
```

0.8413473850093569

```
{'criterion': 'gini', 'max_depth': 6, 'max_features': 'sqrt', 'max_leaf_no
des': 10, 'min_samples_split': 4, 'n_estimators': 10, 'random_state': 13}
```

In []:

```
classifier4 = RandomForestClassifier(criterion=grid3.best_params_['criterion'],
                                   max_depth=grid3.best_params_['max_depth'],
                                   max_features=grid3.best_params_['max_features'],
                                   max_leaf_nodes=grid3.best_params_['max_leaf_node
s'],
                                   min_samples_split=grid3.best_params_['min_samples_s
plit'],
                                   random_state=grid3.best_params_['random_state'],
                                   n_estimators=grid3.best_params_['n_estimators'])
classifier4.fit(titanic_x_train, titanic_y_train)
```

In []:

```
titanic_y_pred = classifier4.predict(titanic_x_test)
titanic_report = classification_report(titanic_y_test, titanic_y_pred)
print(titanic_report)
```

	precision	recall	f1-score	support
0	0.82	0.92	0.87	105
1	0.87	0.72	0.79	74
accuracy			0.84	179
macro avg	0.85	0.82	0.83	179
weighted avg	0.84	0.84	0.83	179



Utilizando AD no problema de Detecção de Lombalgia

In []:

```
pain_x = pain_df.drop(columns=['Class_att'])
pain_y = pain_df['Class_att']

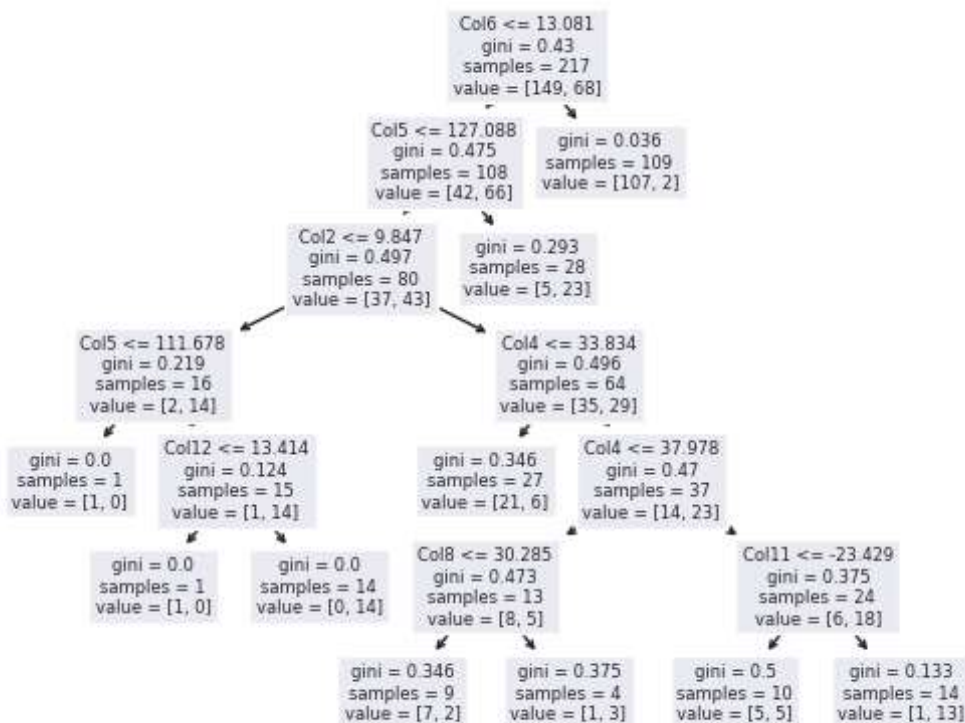
pain_x_train, pain_x_test, pain_y_train, pain_y_test = train_test_split(pain_x, pain_y,
test_size = 0.3)
```

In []:

```
classifier5 = DecisionTreeClassifier(criterion=grid2.best_params_['criterion'],
max_depth=grid2.best_params_['max_depth'],
max_features=grid2.best_params_['max_features'],
max_leaf_nodes=grid2.best_params_['max_leaf_node
s'],
min_samples_split=grid2.best_params_['min_samples_s
plit'],
random_state=grid2.best_params_['random_state'],
splitter=grid2.best_params_['splitter'])
classifier5.fit(pain_x_train, pain_y_train)
```

In []:

```
plot_tree(classifier5, feature_names=pain_df.drop(columns=['Class_att']).columns)
plt.show()
```



In []:

```
pain_y_pred = classifier5.predict(pain_x_test)
pain_report = classification_report(pain_y_test, pain_y_pred)
print(pain_report)
```

	precision	recall	f1-score	support
Abnormal	0.81	0.92	0.86	61
Normal	0.79	0.59	0.68	32
accuracy			0.81	93
macro avg	0.80	0.76	0.77	93
weighted avg	0.80	0.81	0.80	93



Conclusão

Titanic

- Configuração padrão teve os menores scores e houve overfitting.

In []:

```
predict_and_report("AD - Configuração padrão", classifier1, titanic_x_test, titanic_y_test, titanic_y_train, titanic_x_train)
```

AD - Configuração padrão- base de teste

	precision	recall	f1-score	support
--	-----------	--------	----------	---------

0	0.79	0.85	0.82	105
1	0.76	0.68	0.71	74

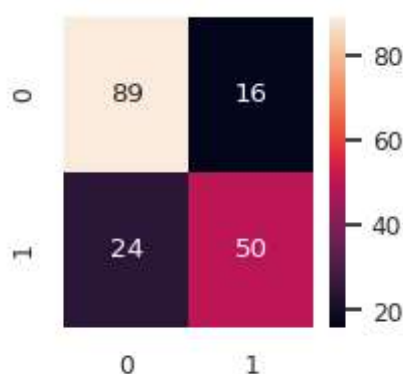
accuracy			0.78	179
macro avg	0.77	0.76	0.77	179
weighted avg	0.78	0.78	0.77	179

AD - Configuração padrão- base de treino

	precision	recall	f1-score	support
--	-----------	--------	----------	---------

0	0.98	1.00	0.99	444
1	0.99	0.96	0.98	268

accuracy			0.98	712
macro avg	0.98	0.98	0.98	712
weighted avg	0.98	0.98	0.98	712



- Configuração 1 teve melhoria nos scores.

In []:

```
print(grid.best_params_)
```

```
{'criterion': 'entropy', 'max_depth': 30, 'max_features': None, 'max_leaf_nodes': 40, 'min_samples_split': 2, 'random_state': 13, 'splitter': 'random'}
```


In []:

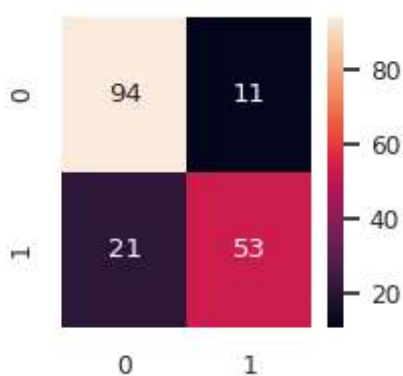
```
predict_and_report("AD - Configuração 1", classifier2, titanic_x_test, titanic_y_test,
titanic_y_train, titanic_x_train)
```

AD - Configuração 1- base de teste

	precision	recall	f1-score	support
0	0.82	0.90	0.85	105
1	0.83	0.72	0.77	74
accuracy			0.82	179
macro avg	0.82	0.81	0.81	179
weighted avg	0.82	0.82	0.82	179

AD - Configuração 1- base de treino

	precision	recall	f1-score	support
0	0.88	0.93	0.90	444
1	0.87	0.78	0.83	268
accuracy			0.88	712
macro avg	0.87	0.86	0.86	712
weighted avg	0.87	0.88	0.87	712



- Configuração 2 teve scores similares (mas é uma árvore consideravelmente menor)

In []:

```
print(grid2.best_params_)
```

```
{'criterion': 'gini', 'max_depth': 6, 'max_features': None, 'max_leaf_node  
s': 10, 'min_samples_split': 2, 'random_state': 13, 'splitter': 'random'}
```

In []:

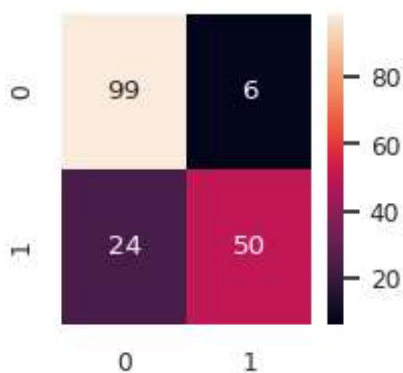
```
predict_and_report("AD - Configuração 2", classifier3, titanic_x_test, titanic_y_test,
titanic_y_train, titanic_x_train)
```

AD - Configuração 2- base de teste

	precision	recall	f1-score	support
0	0.80	0.94	0.87	105
1	0.89	0.68	0.77	74
accuracy			0.83	179
macro avg	0.85	0.81	0.82	179
weighted avg	0.84	0.83	0.83	179

AD - Configuração 2- base de treino

	precision	recall	f1-score	support
0	0.82	0.95	0.88	444
1	0.90	0.65	0.75	268
accuracy			0.84	712
macro avg	0.86	0.80	0.82	712
weighted avg	0.85	0.84	0.83	712



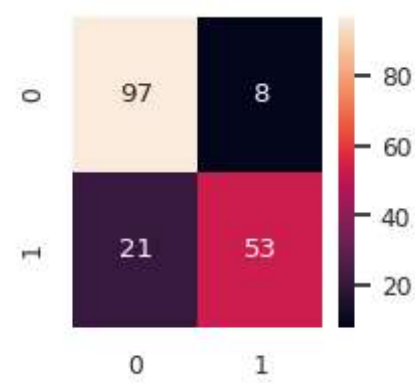
- RF apresentou scores similares.

In []:

```
predict_and_report("RF - Configuração 1", classifier4, titanic_x_test, titanic_y_test,
titanic_y_train, titanic_x_train)
```

RF - Configuração 1- base de teste					
	precision	recall	f1-score	support	
0	0.82	0.92	0.87	105	
1	0.87	0.72	0.79	74	
accuracy			0.84	179	
macro avg	0.85	0.82	0.83	179	
weighted avg	0.84	0.84	0.83	179	

RF - Configuração 1- base de treino					
	precision	recall	f1-score	support	
0	0.84	0.93	0.88	444	
1	0.86	0.71	0.78	268	
accuracy			0.85	712	
macro avg	0.85	0.82	0.83	712	
weighted avg	0.85	0.85	0.84	712	



Detecção de Lombalgia

In []:

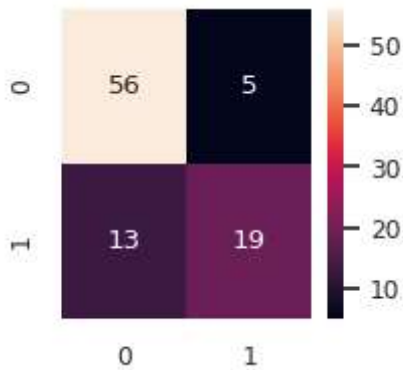
```
predict_and_report("AD - Configuração 1", classifier5, pain_x_test, pain_y_test, pain_y_train, pain_x_train)
```

AD - Configuração 1- base de teste

	precision	recall	f1-score	support
Abnormal	0.81	0.92	0.86	61
Normal	0.79	0.59	0.68	32
accuracy			0.81	93
macro avg	0.80	0.76	0.77	93
weighted avg	0.80	0.81	0.80	93

AD - Configuração 1- base de treino

	precision	recall	f1-score	support
Abnormal	0.90	0.95	0.93	149
Normal	0.88	0.78	0.83	68
accuracy			0.90	217
macro avg	0.89	0.87	0.88	217
weighted avg	0.90	0.90	0.90	217



- Acurácia com MLP: 0.82
- Acurácia com AD (configuração 2): 0.81
- Testando outras configurações considerando esse dataset poderíamos obter resultados melhores.