

Links úteis (deletar depois)

- https://vitalflux.com/pandas-impute-missing-values-mean-median-mode/#How_to_decide_which_imputation_technique_to_use (https://vitalflux.com/pandas-impute-missing-values-mean-median-mode/#How_to_decide_which_imputation_technique_to_use)
- <https://towardsdatascience.com/machine-learning-with-the-titanic-dataset-7f6909e58280> (<https://towardsdatascience.com/machine-learning-with-the-titanic-dataset-7f6909e58280>)
- <https://www.kaggle.com/code/startupsci/titanic-data-science-solutions/notebook> (<https://www.kaggle.com/code/startupsci/titanic-data-science-solutions/notebook>)
- <https://www.kaggle.com/competitions/titanic/discussion/395092> (<https://www.kaggle.com/competitions/titanic/discussion/395092>)

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 [1]:

```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

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

Drive

In [2]:

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

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

Mounted at /content/drive

IRIS Dataset - <https://www.kaggle.com/datasets/arshid/iris-flower-dataset>
(<https://www.kaggle.com/datasets/arshid/iris-flower-dataset>)

In [3]:

```
iris_dataset_path = base_path + 'IRIS.csv'
```

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

In [4]:

```
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 [5]:

```
iris_df = pd.read_csv(iris_dataset_path)
```

In [182]:

```
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
```

In []:

```
pain_df = pd.read_csv(pain_dataset_path)
pain_df.head()
```

Funções

In [7]:

```
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)
```

Experimento: Iris Flower Dataset (pode deletar antes de entregar)

Análise do Dataset

In [8]:

```
iris_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 150 entries, 0 to 149  
Data columns (total 5 columns):  
#   Column          Non-Null Count  Dtype  
---  ---  
0   sepal_length    150 non-null   float64  
1   sepal_width     150 non-null   float64  
2   petal_length    150 non-null   float64  
3   petal_width     150 non-null   float64  
4   species         150 non-null   object  
dtypes: float64(4), object(1)  
memory usage: 6.0+ KB
```

In [9]:

```
iris_df['species'].unique()
```

Out[9]:

```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

In [10]:

```
iris_df['species'].value_counts()
```

Out[10]:

```
Iris-setosa      50  
Iris-versicolor  50  
Iris-virginica   50  
Name: species, dtype: int64
```

In []:

```
#iris_df.describe()
```

In []:

```
#sns.pairplot(iris_df, hue='species')
```

In []:

```
#iris_df.hist(bins=50, figsize=(5,5))  
#plt.show()
```

In []:

```
#sns.set_style("whitegrid")  
#data = np.random.normal(size=(20, 6)) + np.arange(6) / 2  
#sns.set(rc={'figure.figsize':(7,4)})  
#sns.boxplot(data = iris_df.iloc[:, :-1])
```

🔧 Criando modelo

In [15]:

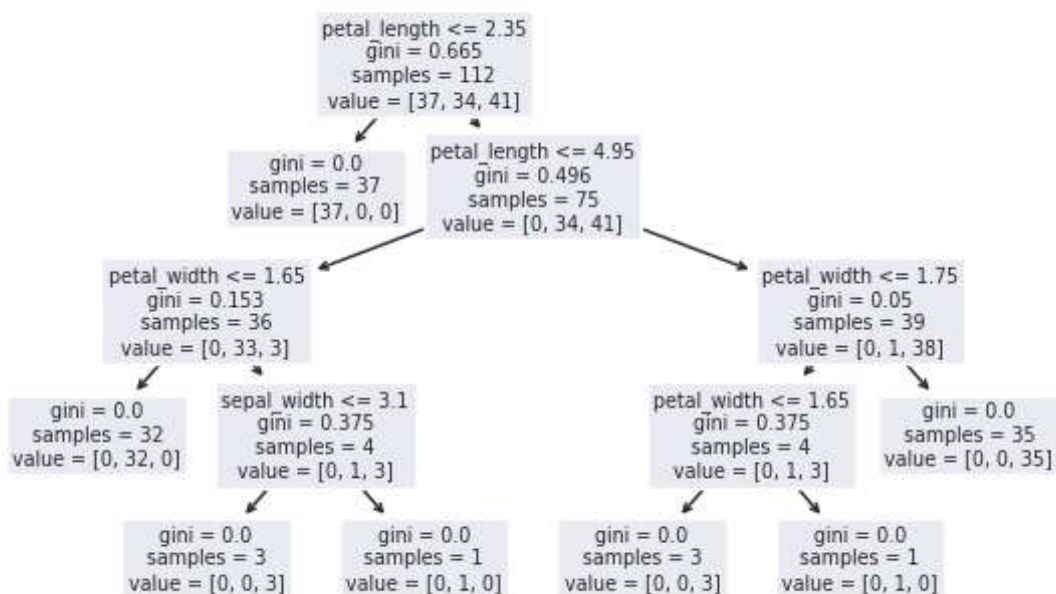
```
iris_x = iris_df.iloc[:, :-1]  
iris_y = iris_df.iloc[:, -1]  
iris_x_train, iris_x_test, iris_y_train, iris_y_test = train_test_split(iris_x, iris_y,  
test_size = 0.25, random_state = 0)
```

In []:

```
classifier = DecisionTreeClassifier()  
classifier.fit(iris_x_train, iris_y_train)
```

In [17]:

```
plot_tree(classifier, feature_names=iris_df.columns[: -1])  
plt.show()
```



In [18]:

```
iris_y_pred = classifier.predict(iris_x_test)
```

In [19]:

```
report = classification_report(iris_y_test, iris_y_pred)
```

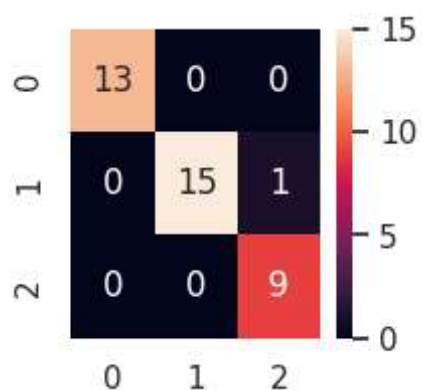
In [20]:

```
print(report)
```

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	13
Iris-versicolor	1.00	0.94	0.97	16
Iris-virginica	0.90	1.00	0.95	9
accuracy			0.97	38
macro avg	0.97	0.98	0.97	38
weighted avg	0.98	0.97	0.97	38

In [21]:

```
plot_confusion_matrix(iris_y_test, iris_y_pred)
```



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 [183]:

```
titanic_df.shape
```

Out[183]:

(891, 12)

In [184]:

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

Out[184]:

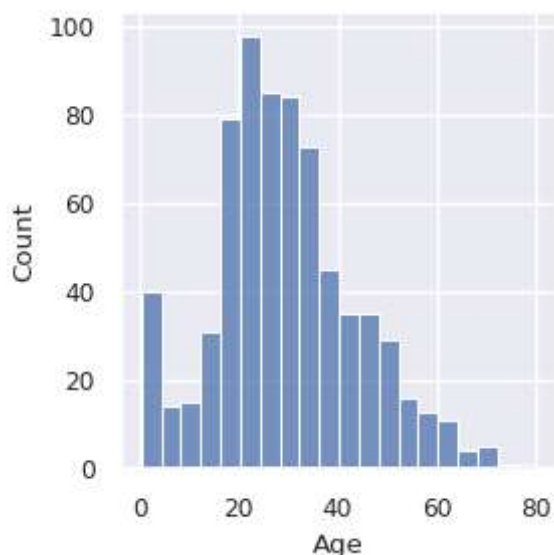
```
PassengerId      0
Survived          0
Pclass           0
Name             0
Sex              0
Age             177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin           687
Embarked         2
dtype: int64
```

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).**

In [185]:

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



In [186]:

```
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 [187]:

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

Out[187]:

```
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 [188]:

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

Embarked

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

In [189]:

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

Out[189]:

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

In [190]:

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

In [191]:

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

Out[191]:

```
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 [192]:

```
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 [193]:

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

Out[193]:

```
891
```


In [194]:

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

891

Out[194]:

```
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 [195]:

```
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[195]:

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

In [196]:

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

In [197]:

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

Outras colunas

In [198]:

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

Out[198]:

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

In [199]:

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

Out[199]:

```
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 [200]:

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

In [201]:

```
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.

In [202]:

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

In [203]:

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

Dataset após limpeza

In [204]:

```
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
```

In [205]:

```
titanic_df.head()
```

Out[205]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	Title
0	0	3	0	22.0	1	0	7.2500	0	0
1	1	1	1	38.0	1	0	71.2833	1	1
2	1	3	1	26.0	0	0	7.9250	0	2
3	1	1	1	35.0	1	0	53.1000	0	1
4	0	3	0	35.0	0	0	8.0500	0	0

Criando modelo

In [207]:

```
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 []:

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

In [209]:

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



In [210]:

```
titanic_y_pred = classifier.predict(titanic_x_test)
```

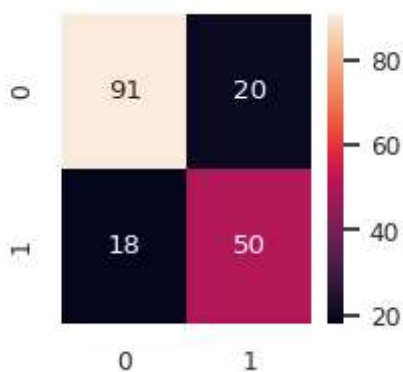
In [211]:

```
titanic_report = classification_report(titanic_y_test, titanic_y_pred)
print(titanic_report)
```

	precision	recall	f1-score	support
0	0.83	0.82	0.83	111
1	0.71	0.74	0.72	68
accuracy			0.79	179
macro avg	0.77	0.78	0.78	179
weighted avg	0.79	0.79	0.79	179

In [212]:

```
plot_confusion_matrix(titanic_y_test, titanic_y_pred)
```





Utilizando Random Forests

In []:



Utilizando AD no problema de Detecção de Lombalgia

In []:



Conclusão