Decision Tree for Titanic Data

Importing the Libs

```
!pip install plotly --upgrade
!pip install yellowbrick --upgrade
```

```
Requirement already satisfied: plotly in c:\users\novae\appdata\local\programs\python\python39\lib\site-packa
Requirement already satisfied: six in c:\users\novae\appdata\local\programs\python\python39\lib\site-package
Requirement already satisfied: tenacity>=6.2.0 in c:\users\novae\appdata\local\programs\python\python39\lib\
WARNING: You are using pip version 21.2.3; however, version 22.0.4 is available.
You should consider upgrading via the 'C:\Users\novae\AppData\Local\Programs\Python\Python39\python.exe -m p
Collecting yellowbrick
  Downloading yellowbrick-1.4-py3-none-any.whl (274 kB)
Requirement already satisfied: scipy>=1.0.0 in c:\users\novae\appdata\local\programs\python\python39\lib\sit
Requirement already satisfied: cycler>=0.10.0 in c:\users\novae\appdata\local\programs\python\python39\lib\s
Requirement already satisfied: numpy>=1.16.0 in c:\users\novae\appdata\local\programs\python\python39\lib\si
Requirement already satisfied: matplotlib!=3.0.0,>=2.0.2 in c:\users\novae\appdata\local\programs\python\pyt
Requirement already satisfied: scikit-learn>=1.0.0 in c:\users\novae\appdata\local\programs\python\python39\
Requirement already satisfied: packaging>=20.0 in c:\users\novae\appdata\local\programs\python\python39\lib\
Requirement already satisfied: pillow>=6.2.0 in c:\users\novae\appdata\local\programs\python\python39\lib\si
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\novae\appdata\local\programs\python\python39\lib
Requirement already satisfied: python-dateutil>=2.7 in c:\users\novae\appdata\local\programs\python\python39
Requirement already satisfied: fonttools>=4.22.0 in c:\users\novae\appdata\local\programs\python\python39\li
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\novae\appdata\local\programs\python\python39\li
Requirement already satisfied: six>=1.5 in c:\users\novae\appdata\local\programs\python\python39\lib\site-pa
Requirement already satisfied: joblib>=0.11 in c:\users\novae\appdata\local\programs\python\python39\lib\sit
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\novae\appdata\local\programs\python\python39
Installing collected packages: yellowbrick
Successfully installed yellowbrick-1.4
WARNING: You are using pip version 21.2.3; however, version 22.0.4 is available.
You should consider upgrading via the 'C:\Users\novae\AppData\Local\Programs\Python\Python39\python.exe -m p
```

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

Base de dados de Titanic

- Fonte (adaptado): https://www.kaggle.com/c/titanic/data

Exploração dos Dados

```
base = pd.read_csv('./content/titanic.csv')
```

```
base.shape
```

```
    (891, 12)
```

```
base.head(10)
base.describe()
```

It is noticed that there are no inconsistent values

```
base.columns
```

```
    Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
           'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
          dtype='object')
```

It does not need the "PassengerId", "Name", "Cabin", "Ticket"

```
base = base.drop('PassengerId', axis = 1)
base = base.drop('Name', axis = 1)
base = base.drop('Ticket', axis = 1)
base = base.drop('Cabin', axis = 1)
```

Missing values handling

```
base.isnull().sum()
```

```
    Survived      0
    Pclass        0
    Sex           0
```

```
Age         177
SibSp         0
Parch         0
Fare          0
Embarked      2
dtype: int64
```

It is Filling the null values with the not null values mean.

```
base['Age'].fillna(base['Age'].mean(), inplace = True)
base.shape
```

```
(891, 8)
```

It is dropping the NaN values of 'Embarked'

```
base= base.drop(base[base['Embarked'].isna()].index)
base.shape
```

```
(889, 8)
```

```
base.isnull().sum()
```

```
Survived    0
Pclass      0
Sex         0
Age         0
SibSp       0
Parch       0
Fare        0
Embarked    0
dtype: int64
```

```
base
```

Data Visualization

```
sns.countplot(x = base['Survived']);
```

```
plt.pie(list(base.Embarked.value_counts().to_dict().values()),
        labels=list(base.Embarked.value_counts().to_dict().keys()),
        colors=sns.color_palette('husl',3),
        autopct='%.0f%%')
plt.show()
```

Exploratory Analysis

```
grafico = px.treemap(base, path=['Survived', 'Pclass', 'Embarked'])
grafico.show()
```

```
grafico = px.parallel_categories(base, dimensions=['Pclass', 'Sex', 'Survived'])
grafico.show()
```

```
grafico = px.scatter_matrix(base, dimensions=['SibSp', 'Parch', 'Fare', 'Age'], color = 'Survived')
grafico.show()
```

```
plt.subplots(figsize=(16,12))
sns.heatmap(
    base.corr(),
    annot=True,
    square=True,
    cbar=True
)
```

Divion between predictor and class

```
base.columns
```

```
    Index(['Survived', 'Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare',
           'Embarked'],
          dtype='object')
```

```
X_Titanic = base.iloc[:, 1:8].values
X_Titanic
```

```
    array([[3, 'male', 22.0, ..., 0, 7.25, 'S'],
           [1, 'female', 38.0, ..., 0, 71.2833, 'C'],
           [3, 'female', 26.0, ..., 0, 7.925, 'S'],
           ...,
           [3, 'female', 29.69911764705882, ..., 2, 23.45, 'S'],
           [1, 'male', 26.0, ..., 0, 30.0, 'C'],
           [3, 'male', 32.0, ..., 0, 7.75, 'Q']], dtype=object)
```

```
Y_Titanic = base.iloc[:, 0].values
```

```
Y_Titanic
```

```
array([0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1,
       1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1,
       1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1,
       0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1,
       0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1,
       0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0,
       0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1,
       0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0,
       1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1,
       0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0,
       1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0,
       0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1,
       0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1,
       0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1,
       1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0,
       0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1,
       0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1,
       1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0,
       0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0,
       0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0,
       0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0,
       0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1,
       1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0,
       0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0,
       1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0,
                0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0,
```

Categorical attribute handling

LabelEncoder

preparing our data for our models

```
from sklearn.preprocessing import LabelEncoder
X_Titanic[:,1] = LabelEncoder().fit_transform(X_Titanic[:,1])
X_Titanic[:,1]
```

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

```
X_Titanic[:,6] = LabelEncoder().fit_transform(X_Titanic[:,6])
X_Titanic[:,6]
```

```
    array([2, 0, 2, 2, 2, 1, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 1, 2, 2, 0, 2, 2,
           1, 2, 2, 2, 0, 2, 1, 2, 0, 0, 1, 2, 0, 2, 0, 2, 2, 0, 2, 2, 0, 0,
           1, 2, 1, 1, 0, 2, 2, 2, 0, 2, 0, 2, 2, 0, 2, 2, 0, 2, 2, 0, 0, 2,
           2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2,
           2, 2, 2, 2, 2, 2, 2, 0, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2,
           0, 2, 2, 0, 2, 1, 2, 0, 2, 2, 2, 0, 2, 2, 0, 1, 2, 0, 2, 0, 2, 2,
           2, 2, 0, 2, 2, 2, 0, 0, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
           0, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 0, 2, 2,
           0, 2, 2, 2, 0, 2, 2, 2, 2, 1, 2, 1, 2, 2, 2, 2, 2, 0, 0, 1, 2, 1,
           2, 2, 2, 2, 0, 2, 2, 2, 0, 1, 0, 2, 2, 2, 2, 1, 0, 2, 2, 0, 2, 2,
           2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 1, 2,
           2, 0, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 0, 2, 0, 2, 1, 2, 2, 2, 1,
           2, 2, 2, 2, 2, 2, 2, 2, 0, 1, 2, 2, 2, 1, 2, 1, 2, 2, 2, 2, 0, 2,
           2, 2, 1, 2, 0, 0, 2, 2, 0, 0, 2, 2, 0, 1, 1, 2, 1, 2, 2, 0, 0, 0,
           0, 0, 0, 2, 2, 2, 2, 2, 2, 2, 0, 2, 2, 1, 2, 2, 0, 2, 2, 2, 0, 1,
           2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0,
           2, 0, 2, 2, 2, 1, 1, 2, 0, 0, 2, 1, 2, 0, 0, 1, 0, 0, 2, 2, 0, 2,
           0, 2, 0, 0, 2, 0, 0, 2, 2, 2, 2, 2, 2, 1, 0, 2, 2, 2, 0, 2, 2, 2,
           2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 2, 2, 2, 2, 2, 2,
           2, 0, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
           2, 2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 0, 0, 2, 0, 2, 2, 2, 1, 2, 2, 2,
           2, 2, 2, 2, 2, 1, 0, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0,
           2, 2, 0, 2, 2, 2, 2, 2, 0, 2, 0, 0, 2, 2, 2, 2, 1, 1, 2, 2, 0, 2,
           2, 2, 2, 1, 2, 2, 0, 2, 2, 2, 1, 2, 2, 2, 2, 0, 0, 0, 1, 2, 2, 2,
           2, 2, 0, 0, 0, 2, 2, 2, 0, 2, 0, 2, 2, 2, 2, 0, 2, 2, 0, 2, 2, 0,
           2, 1, 0, 2, 2, 0, 0, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2,
           1, 2, 2, 2, 2, 0, 2, 2, 0, 2, 0, 0, 2, 2, 0, 2, 2, 2, 0, 2, 1, 2,
           2, 2, 2, 0, 0, 2, 2, 2, 2, 0, 2, 2, 2, 0, 2, 2, 2, 1, 1, 2, 2, 2,
           2, 2, 2, 0, 2, 0, 2, 2, 2, 1, 2, 2, 1, 2, 2, 0, 2, 2, 2, 2, 2, 2,
           2, 2, 0, 2, 2, 0, 0, 2, 0, 2, 2, 2, 2, 2, 1, 1, 2, 2, 1, 2, 0, 2,
           0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 1, 0, 2,
           2, 2, 0, 2, 2, 2, 2, 2, 0, 2, 0, 2, 2, 2, 1, 0, 2, 0, 2, 0, 1, 2,
           2, 2, 2, 2, 0, 0, 2, 2, 2, 2, 2, 0, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2,
           1, 2, 2, 2, 0, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2,
           1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 0, 1, 1, 2, 2,
           2, 2, 0, 2, 2, 1, 2, 1, 2, 0, 2, 2, 2, 2, 2, 2, 1, 2, 0, 1, 2, 2,
           0, 2, 2, 2, 2, 0, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
           2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 1, 2, 0, 1, 0, 2, 0, 2, 2, 0, 2, 2,
           2, 0, 2, 2, 0, 0, 2, 2, 2, 0, 2, 0, 2, 2, 0, 2, 2, 2, 2, 2, 0, 0,
           2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 0, 0, 2, 2, 2, 0, 2, 2,
           2, 2, 2, 1, 2, 2, 2, 0, 1], dtype=object)
```

Escalation of values

```
from sklearn.preprocessing import MinMaxScaler
X_Titanic =  MinMaxScaler().fit_transform(X_Titanic)
X_Titanic
```

```
    array([[1.        , 1.        , 0.27117366, ..., 0.        , 0.01415106,
            1.        ],
           [0.        , 0.        , 0.4722292 , ..., 0.        , 0.13913574,
            0.        ],
           [1.        , 0.        , 0.32143755, ..., 0.        , 0.01546857,
            1.        ],
           ...,
           [1.        , 0.        , 0.36792055, ..., 0.33333333, 0.04577135,
            1.        ],
           [0.        , 1.        , 0.32143755, ..., 0.        , 0.0585561 ,
            0.        ],
           [1.        , 1.        , 0.39683338, ..., 0.        , 0.01512699,
            0.5       ]])
```

Division of bases into training and testing

```
from sklearn.model_selection import train_test_split
X_Titanic_treinamento, X_Titanic_teste, Y_Titanic_treinamento, Y_Titanic_teste = train_test_split(X_Titanic, Y_Tit
```

```
X_Titanic_treinamento.shape, Y_Titanic_treinamento.shape
```

```
    ((666, 7), (666,))
```

```
X_Titanic_teste.shape, Y_Titanic_teste.shape
```

```
    ((223, 7), (223,))
```

Saving the variables

```
import pickle
with open('titanic.pkl', mode = 'wb') as f:
  pickle.dump([X_Titanic_treinamento, Y_Titanic_treinamento, X_Titanic_teste, Y_Titanic_teste], f)
```

Training the Model with a Decision Tree 76,23% of precision

```
from sklearn.tree import DecisionTreeClassifier
arvore_Titanic = DecisionTreeClassifier(criterion='entropy')
arvore_Titanic.fit(X_Titanic_treinamento, Y_Titanic_treinamento)
```

```
    DecisionTreeClassifier(criterion='entropy')
```

```
arvore_Titanic.feature_importances_

    array([0.11216816, 0.24608242, 0.28688116, 0.04836535, 0.03000124,
           0.24917688, 0.02732478])
```

```
arvore_Titanic.classes_

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

```
from sklearn import  tree
#previsores = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width']
fig, axes = plt.subplots(nrows = 1,ncols = 1,figsize = (20,20))
tree.plot_tree(arvore_Titanic, class_names = ['0','1'], filled=True);
```

```
previsoes_Titanic = arvore_Titanic.predict(X_Titanic_teste)
```

```
from sklearn.metrics import accuracy_score, classification_report
accuracy_score(Y_Titanic_teste, previsoes_Titanic)
```

```
    0.7668161434977578
```

```
from yellowbrick.classifier import ConfusionMatrix
cm = ConfusionMatrix(arvore_Titanic)
cm.fit(X_Titanic_treinamento, Y_Titanic_treinamento)
cm.score(X_Titanic_teste, Y_Titanic_teste)
```

```
print(classification_report(Y_Titanic_teste, previsoes_Titanic))
```

```
              precision    recall  f1-score   support

           0       0.80      0.81      0.80       132
           1       0.72      0.70      0.71        91

    accuracy                           0.77       223
   macro avg       0.76      0.76      0.76       223
weighted avg       0.77      0.77      0.77       223
```

```
def trainAndAnalyze(model):
    model.fit(X_Titanic_treinamento, Y_Titanic_treinamento)
    print("Model score:")
    print(model.score(X_Titanic_teste, Y_Titanic_teste))
    print("Classification Report")
    print(classification_report(Y_Titanic_teste, model.predict(X_Titanic_teste)))
    print("Confusion Matrix:")
    confMatrix = ConfusionMatrix(model)
    confMatrix.fit(X_Titanic_treinamento, Y_Titanic_treinamento)
    confMatrix.score(X_Titanic_teste,Y_Titanic_teste)
    fig, axes = plt.subplots(nrows = 1,ncols = 1,figsize = (20,20))
    tree.plot_tree(model, class_names = ['0','1'], filled=True)
    print("Tree:")
```

Changing some parameters in our Decision Tree Classifier

We suspect that if we define a maximum number of levels for our tree, we can avoid overfitting

```
tree2 = DecisionTreeClassifier(
    criterion='entropy',
    max_depth=6,
    splitter='best',
    max_features=None,
    min_impurity_decrease=0.01
)
trainAndAnalyze(tree2)
```

```
trainAndAnalyze(
    DecisionTreeClassifier(
        criterion='entropy',
        max_depth=50,
        min_samples_split=10,
        splitter='random',
        max_features='auto',
        min_impurity_decrease=0.001
```

```
            )
        )
```

```
trainAndAnalyze(
    DecisionTreeClassifier(
        criterion='gini',
        max_depth=50,
        min_samples_split=6,
        splitter='best',
        max_features='log2',
        min_impurity_decrease=0.001
    )
)
```

```
trainAndAnalyze(
        DecisionTreeClassifier(
            criterion='gini',
            max_depth=10,
            min_samples_split=5,
            splitter='best',
            max_features='log2',
            min_impurity_decrease=0.001,
            min_weight_fraction_leaf=0.001,
            class_weight='balanced'
        )
    )
```

```
trainAndAnalyze(
    DecisionTreeClassifier(
        criterion='entropy',
        max_depth=10,
        min_samples_split=5,
        splitter='best',
        max_features='log2',
        min_impurity_decrease=0.001,
        min_weight_fraction_leaf=0.001,
        class_weight='balanced'
    )
)
```

```
trainAndAnalyze(
    DecisionTreeClassifier(
        criterion='entropy',
        max_depth=10,
        min_samples_split=10,
        splitter='best',
        max_features='log2',
        min_impurity_decrease=0.001,
        min_weight_fraction_leaf=0.01,
        class_weight='balanced'
    )
)
```