Download libraries and access the dataset file

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

```
Mounted at /content/drive
```

```
!pip install sweetviz
```

```
import numpy as np

# data processing
import pandas as pd


# data visualization
import seaborn as sns
%matplotlib inline
from matplotlib import pyplot as plt
from matplotlib import style

# Algorithms
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import balanced_accuracy_score
from sklearn import tree

# Statistic visualization
import sweetviz as sv

#Tree Visualization
import graphviz


from zipfile import ZipFile
zip_file = ZipFile('/content/drive/MyDrive/IA/titanic.zip')
```

First Step

- Create variables for the 'train', 'test' and 'gender' data
- Check the composition of each table
- Switch string values to integer or float values
- Use Sweetviz to identify columns with low correlation with the target variable ('Died')
- Remove some columns.

Creating variables and check the composition of each table

[ ] ↳ *5 células ocultas*

Gathering informations about the tables

[ ] ↳ *3 células ocultas*

Searching for null values, replacing them with the most common value in the column, and converting strings to integers

[ ] ↳ *5 células ocultas*

Sweetviz

[ ] ↳ *1 célula oculta*

Removing columns

[ ] ↳ *3 células ocultas*

Second Step

- Supervised Learning
- Final Decision on Results
- Decision Tree Visualization
- Graphs to Illustrate Decision Making with Different Parameters

Supervised Learning and Final Decision

In this stage, I chose to use a max_depth of 1 because the decision in the Decision Tree tends to have higher accuracy in the level 1 of the tree.

```
X_train = data_train.drop(['Survived'], axis = 1).values
y_train = data_train['Survived'].values

X_test = data_test.values
y_test = data_gender['Survived'].values

clf = DecisionTreeClassifier(min_samples_split = 10, max_depth = 1, random_state = 0)
clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
score = balanced_accuracy_score(y_test, y_pred)

print(f'Accurancy rate = {round(score*100,2)} %')
```

```
Accurancy rate = 100.0 %
```

Decision Tree Classifier and Graphs

Decision Tree Classifier

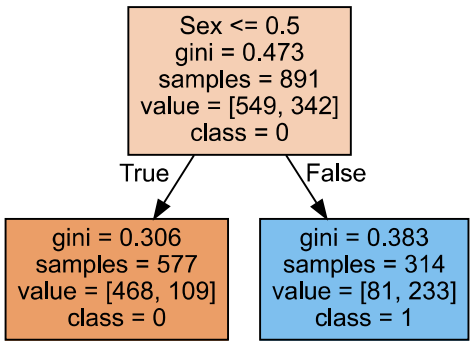
In the image, you'll notice that the tree makes decisions primarily based on its right side. With this information, another approach to achieving 100% accuracy without overfitting is to set min_samples_split to 315.

```
attributes = data_train.columns.values.tolist()
if attributes[0] == 'Survived':
  del attributes[0]

decisoes = np.unique(data_train['Survived'].values).tolist()

j = 0
for i in decisoes:
  decisoes[j] = str(i)
  j += 1


dot_data = tree.export_graphviz(clf, out_file=None,
                                feature_names=attributes,
                                class_names=decisoes,
                                filled=True)
graph = graphviz.Source(dot_data, format="png")
graph
```



Graphs Showing the difference of choosing different parameters on classifier function.

[ ] ↳ *8 células ocultas*