

***INTELIGÊNCIA ARTIFICIAL***

**LABORATÓRIO # 5**

**ASSUNTO:** ÁRVORE DE DECISÃO USANDO PYTHON

**Materiais de Apoio**

**Site oficial do Scikit Learn:**

**scikit-learn.org/**

**Site oficial do Projeto Jupyter:**

**http://jupyter.org/**

**Instalação do Jupyter:**

[**http://jupyter.org/install**](http://jupyter.org/install)

**Vídeo BEM introdutório sobre Jupyter:**

**https://www.youtube.com/watch?v=m0FbNlhNyQ8**

**Vídeo interessante para Decision Tree em Python (inglês):**

[**https://www.youtube.com/watch?v=303yUAhD\_RE**](https://www.youtube.com/watch?v=303yUAhD_RE)

**Blog com 4 formas de Visualização de Árvores de Decisão:**

**https://mljar.com/blog/visualize-decision-tree/**

**Importante:**

**No Python, atenção aos tabs! O Python entende os tabs como begin/end ou seja, o que está dentro do corpo de uma função**

**Parte 1 – Importação de Bibliotecas**

***Dataset usado:*** *nenhum*

*import pandas as pd*

*from sklearn.tree import DecisionTreeClassifier*

*from sklearn import tree*

*from matplotlib import pyplot as plt*

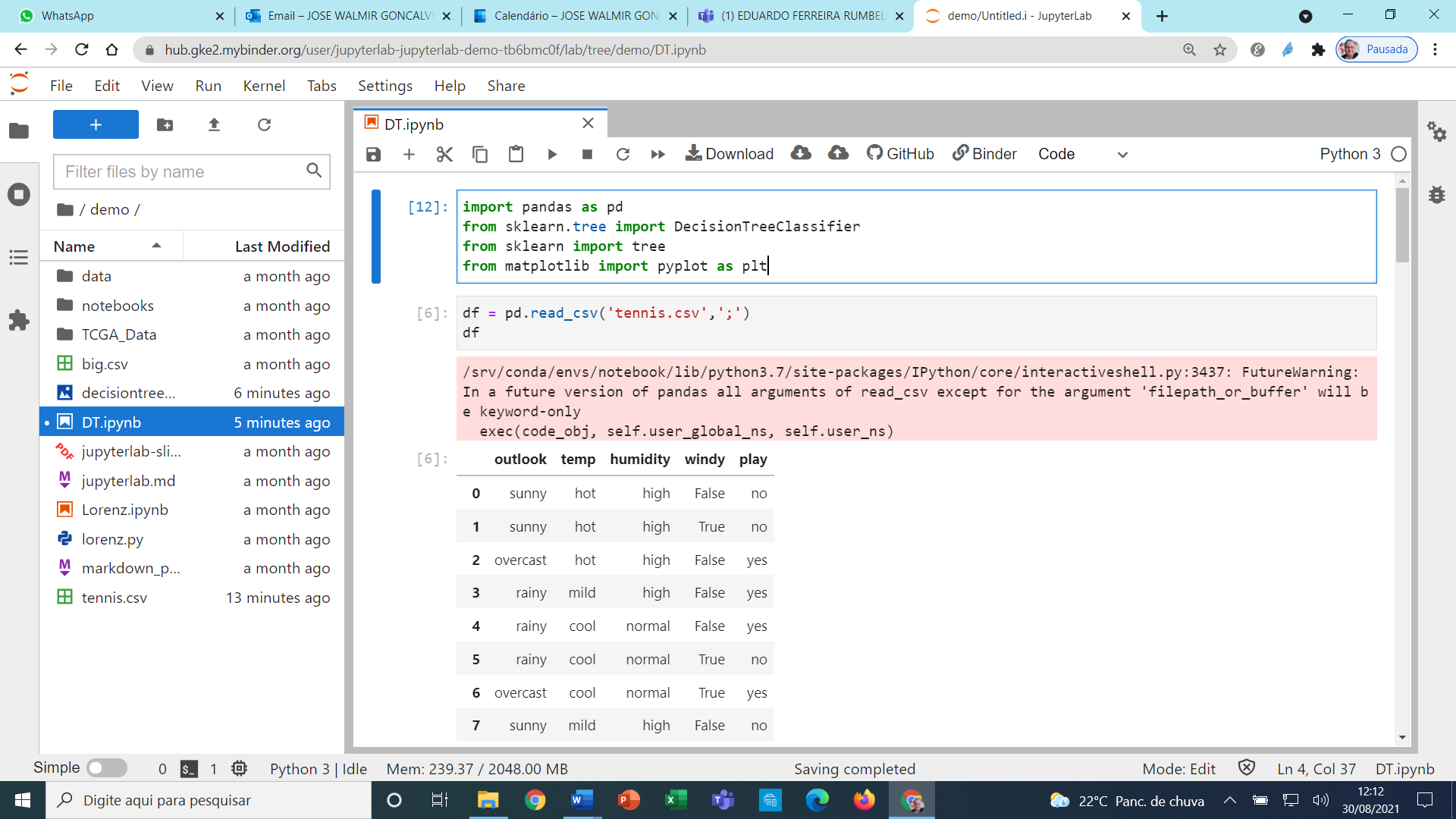
***Obs.*** Se precisar instalar alguma biblioteca, criar uma nova célula e instalar *pip install* ***nomedabiblioteca***

**Parte 2 – Leitura de dados usando Pandas, Definição e Transformação de dados de Classes**

***Dataset usado:*** *tennis.csv*

***Obs.*** Se estiver usando Jupyter Notebook (Cloud) ou Jupyter Lab (Cloud), pode fazer upload do arquivo tennis.csv.

No JupyterLab (Cloud), basta arrastar e soltar o arquivo na área dos arquivos (à esquerda). Após upload, se o arquivo estiver na área /demo, não precisa se preocupar com path, mas, se inserir na pasta /demo/data, usar ‘data/tennis.csv’.



*# leitura de dados (CSV), separados por ponto e vírgula no nosso caso*

*df = pd.read\_csv('tennis.csv',';')*

*df*

*# factorize – codifica os valores categóricos – árvore de decisão do Python precisa de dados numéricos*

*df['play'],class\_names = pd.factorize(df['play']) # False=0; True=1*

*print(class\_names)*

*print(df['play'].unique())*

*df['outlook'],\_ = pd.factorize(df['outlook'])*

*print(df['outlook'].unique())*

*df['temp'],\_ = pd.factorize(df['temp']) # Hot=0; Mild=1; Cool=2*

*print(df['temp'].unique())*

*df['humidity'],\_ = pd.factorize(df['humidity']) # High=0; Normal=1*

*print(df['humidity'].unique())*

*df['windy'],\_ = pd.factorize(df['windy']) # False=0; True=1*

*print(df['windy'].unique())*

*# mostra todo o dataset transformado para numérico*

*df*

*# mostra as primeirs linhas do dataset (útil em caso de grandes datasets)*

*df.head()*

*# mostra informações básicas do dataset como quais colunas (e seus tipos) e uso de memória*

*df.info()*

**Parte 3 – Treinamento e Testes – Abordagem 1**

***Abordagem 1 do Lab:*** *mesmos dados usados para Treinamento serão usados para Testes!*

*# Variáveis X são as colunas features e Variáveis Y trata-se da coluna target*

*# separa as colunas de features (colunas com os dados)*

*feature\_cols = ['outlook','temp','humidity','windy']*

*X\_train = df[feature\_cols] # Features (conj. dados)*

*# separa a coluna target (colunas com os dados das classes)*

*y\_train = df.play # Target variable (conj. dados)*

*# mostra as colunas separadas para treinamento*

*X\_train.columns # nomes das colunas features para treinamento*

*df.columns[4] # nome da coluna targer para reinamento*

*# constrói o modelo usando critério “Entropia”*

*clf = DecisionTreeClassifier(criterion='entropy')*

*# treina o modelo – observe que nessa abordagem está usando os mesmos dados para treinamento e testes (sem separação)*

*clf = clf.fit(X\_train,y\_train)*

*# mostra modelo processado*

*clf*

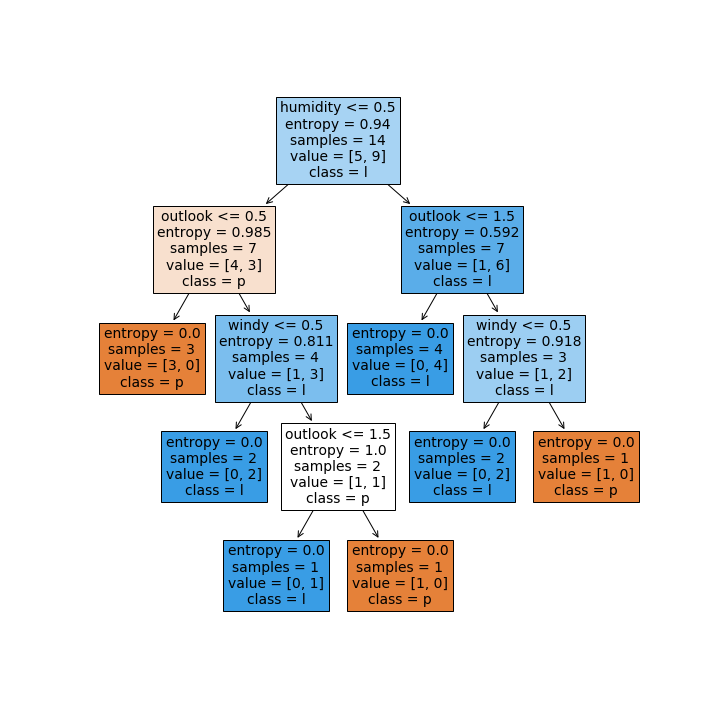
*# mostra a árvore*

*fig1 = plt.figure(figsize=(10,10))*

*\_ = tree.plot\_tree(clf, feature\_names=X\_train.columns,class\_names=df.columns[4], filled=True)*

*fig1.savefig("fig1.png")*

***Veja o resultado (Figura 1) na próxima página!!***

**

**Parte 4 – Treinamento e Testes – Abordagem 2**

***Abordagem 2 do Lab:*** *agora, vamos separar dados de Treinamento (70%) e Testes (30%)*

*# separa as colunas de features (colunas com os dados)*

*feature\_cols = ['outlook','temp','humidity','windy']*

*X\_train = df[feature\_cols] # Features (conj. dados)*

*# separa a coluna target (colunas com os dados das classes)*

*y\_train = df.play # Target variable (conj. dados)*

*# Uso do Modelo para fazer Classificações com os dados de Testes*

*# Para funcionar, deve ser importada a biblioteca:*

***from sklearn.model\_selection import train\_test\_split***

*# Separação de dados para Treinamento (train) e Testes (test); Variáveis X são as colunas features e Variáveis Y trata-se da coluna target*

*X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_train, y\_train, test\_size=0.3, random\_state=1)*

*# Visualização dos dados de Treinamento e Testes, já separados*

*X\_train*

*X\_test*

*# constrói o modelo usando critério “Entropia”*

*clf = DecisionTreeClassifier(criterion='entropy')*

*# treina o modelo – observe que nessa abordagem está usando os mesmos dados para treinamento e testes (sem separação)*

*clf = clf.fit(X\_train,y\_train)*

*# mostra modelo processado*

*clf*

*# mostra a árvore*

*fig2 = plt.figure(figsize=(10,10))*

*\_ = tree.plot\_tree(clf, feature\_names=X\_train.columns,class\_names=df.columns[4], filled=True)*

*fig2.savefig("fig2.png")*

*# uso do modelo para fazer predições (classificações) com os dados de testes*

*y\_pred = clf.predict(X\_test)*

*# como o nosso modelo performou ?*

*# Para funcionar, deve ser importada a biblioteca:*

***from sklearn import metrics***

*count\_misclassified = (y\_test != y\_pred).sum()*

*print('Amostras erroneamente classificadas: {}'.format(count\_misclassified))*

*accuracy = metrics.accuracy\_score(y\_test, y\_pred)*

*print('Acuracia: {:.2f}'.format(accuracy))*

*# Testando uma amostra especifica!*

*# Para funcionar, deve ser importada a biblioteca:*

***import numpy as np***

*# Para a Entrada outlook=1 ; temp=0; humidity=1; windy=0*

*# Qual a classe (saída)?*

*W\_test = np.array ([[1, 0, 1, 0]])*

*y\_pred = clf.predict(W\_test)*

*y\_pred*

*# Resposta: 1 (Yes) - Linha 12 do Dataset e que também está presente no Treinamento!!*

**Bom Trabalho!!**