Instituto Politécnico de Viana do Castelo

LICENCIATURA EM ENGENHARIA INFORMÁTICA

Árvores de Decisão em Python – Dataset weather

Jorge Ribeiro

- jribeiro@estg.ipvc.pt

Índice

Árvores de decisão – O que é

Árvore de decisão é um tipo de algoritmo de aprendizagem supervisionada (com uma variável alvo pré-definida), muito utilizada em problemas de classificação.

Ela funciona para ambas as variáveis categóricas e contínuas de entrada e de saída.

Na árvore de decisão, dividimos a população ou amostra
em dois ou mais conjuntos homogêneos (ou sub-populações) com base nos divisores/diferenciadores
mais significativos das variáveis de entrada.

Tipos de Árvores de decisão

Os tipos de árvore de decisão baseiam-se no tipo de variável de destino que temos. São dois tipos:

Árvore de decisão de variável categórica: árvore de decisão que tem a variável de destino categórica, chamada assim de árvore de decisão de variável categórica. Exemplo: – No cenário anterior, onde a variável alvo era “O aluno joga tênis ou não”, os valores são SIM ou NÃO.

Árvore de decisão de variável contínua: cuja variável alvo é contínua. Exemplo: – Digamos que o problema seja prever se um cliente vai renovar o prêmio de seguro que paga a uma companhia de seguros (Sim ou Não). Nesse problema, sabemos que a renda do cliente é uma variável significativa, mas a companhia de seguros não conhece a renda de todos seus clientes. Agora, como sabemos que esta é uma variável importante, então podemos construir uma árvore de decisão para prever a renda do cliente com base na sua ocupação, no produto segurado e várias outras variáveis. Neste caso, estamos a prever valores para uma variável contínua.

Terminologias das Árvores de decisão

1.Nó Raiz: Representa a população inteira ou amostra, sendo ainda dividido em dois ou mais conjuntos homogêneos.

1.Divisão: É o processo de dividir um nó em dois ou mais sub-nós.

1.Nó de Decisão: Quando um sub-nó é dividido em sub-nós adicionais.

1.Folha ou Nó de Término: Os nós não divididos são chamados Folha ou Nó de Término.

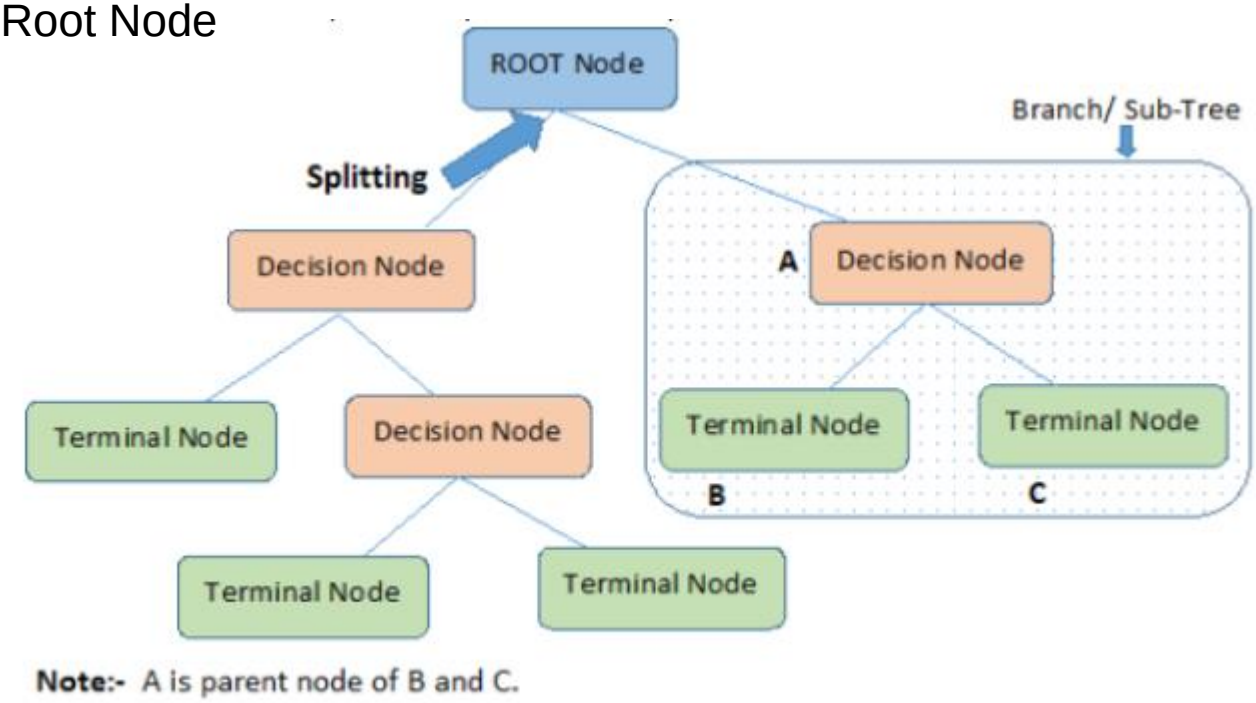
1.Poda: O processo de remover sub-nós de um nó de decisão é chamado poda. Podemos dizer que é o processo oposto ao de divisão.

1.Ramo: Uma sub-seção da árvore inteira é chamada de ramo.

1.Nó pai e nó filho: Um nó que é dividido em sub-nós é chamado de nó pai. Os sub-nós são os nós filhos do nó pai.

Árvores de decisão

Nó = Node
Divisão = splitting
Nó de decisão = Decision Node
Terminal Node = Nó de Término ou folha
Ramo = Branch
Nó Raiz = Root Node



Vantagens

1.Fácil de entender: A visualização de uma árvore de decisão torna o problema fácil de compreender, mesmo para pessoas que não tenham perfil analítico. Não requer nenhum conhecimento estatístico para ler e interpretar. Sua representação gráfica é muito intuitiva e permite relacionar as hipóteses também facilmente.

1.Útil em exploração de dados: A árvore de decisão é uma das formas mais rápidas de identificar as variáveis mais significativas e a relação entre duas ou mais variáveis. Com a ajuda de árvores de decisão, podemos criar novas variáveis/características que tenham melhores condições de predizer a variável alvo.

3.Menor necessidade de limpar dados: Requer menos limpeza de dados em comparação com outras técnicas de modelagem. Até um certo nível, não é influenciado por pontos fora da curva “outliers” nem por valores faltantes (“missing values”).

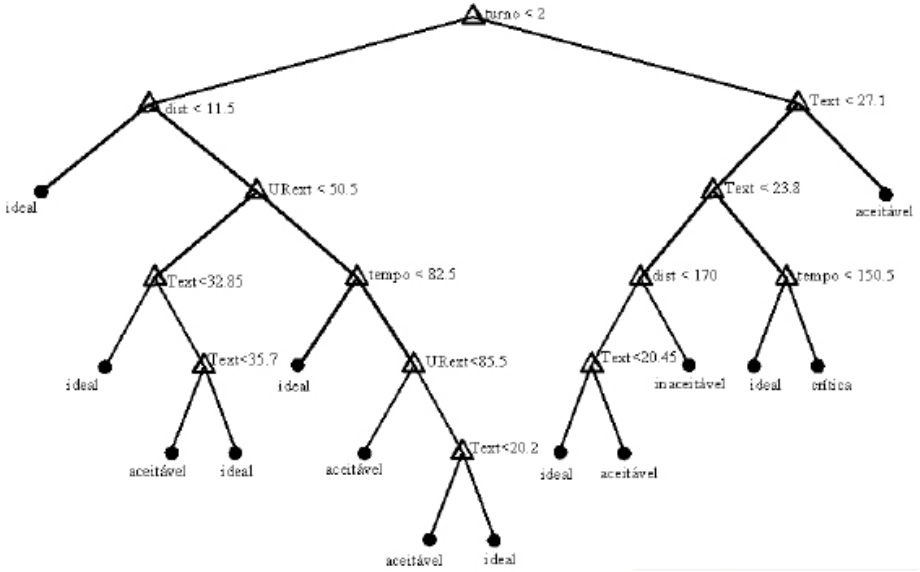
4.Não é restrita por tipos de dados: Pode manipular variáveis numéricas e categóricas.

5.Método não paramétrico: A árvore de decisão é considerada um método não-paramétrico. Isto significa que as árvores de decisão não pressupõe a distribuição do espaço nem a estrutura do classificador.

Desvantagens

1.Sobreajuste ("Over fitting"): Sobreajuste é uma das maiores dificuldades para os modelos de árvores de decisão. Este problema é resolvido através da definição de restrições sobre os parâmetros do modelo e da poda.

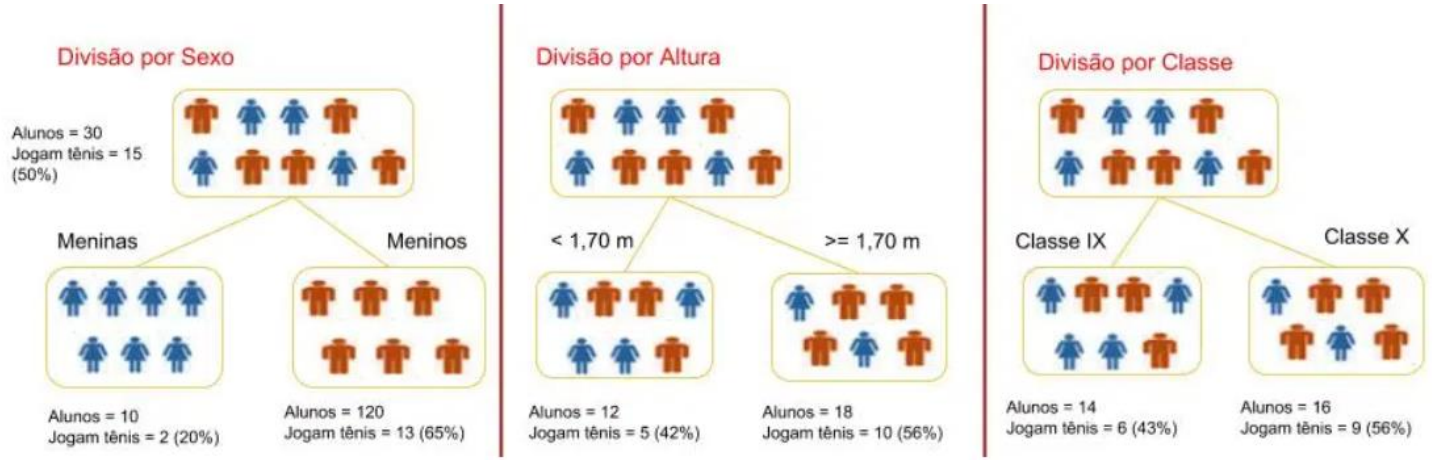
1.Não adequado para variáveis contínuas: ao trabalhar com variáveis numéricas contínuas, a árvore de decisão perde informações quando categoriza variáveis em diferentes categorias.



Árvores de decisão – Exemplo

Digamos que uma amostra de 30 alunos tem três variáveis: Sexo, Classe (IX ou X) e Altura. Digamos também que 15 destes 30 jogam tênis no recreio. Neste problema, precisamos de segregar os alunos (que jogam ténis) com base nas três variáveis à nossa disposição.

É então aí que entra a árvore de decisão. Ela segregará os alunos com base nos valores das três variáveis e identificará a variável que cria os melhores conjuntos homogêneos de alunos (que são heterogêneos entre si). No gráfico abaixo que representa o nosso exemplo, pode-se ver que a variável Sexo é capaz de identificar os melhores conjuntos homogêneos em comparação com as variáveis Altura e Classe.



Árvores de Regressão versus Árvores de Classificação

Nós de término (ou folhas) estão na parte inferior da árvore de decisão. Isto significa que as árvores de decisão são desenhadas de cabeça para baixo. Dessa forma, as folhas são o fundo e as raízes são os topos.

Ambas as árvores trabalham de forma quase semelhantes entre si. Vamos olhar para as diferenças primárias e para as semelhanças entre as árvores de classificação e regressão:

1. As árvores de regressão são utilizadas quando a variável dependente é contínua. As árvores de classificação são usadas quando a variável dependente é categórica.
2. No caso da árvore de regressão, o valor obtido pelos nós de término nos dados de treinamento é o valor médio das suas observações. Assim, a uma nova observação de dados atribui-se o valor médio correspondente.

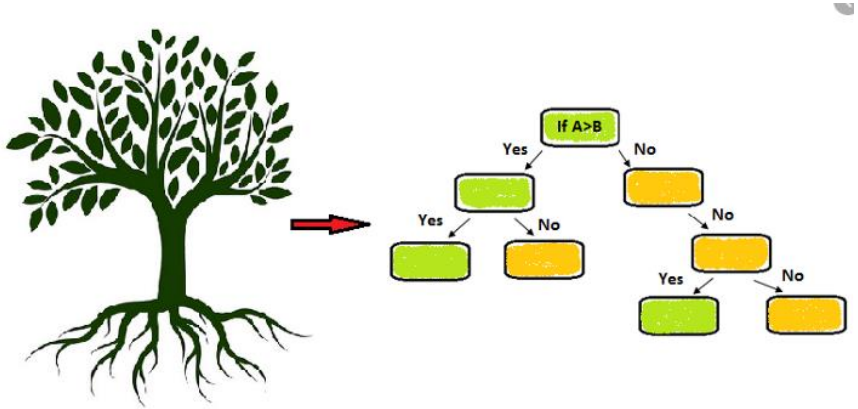
Árvores de decisão

1. No caso da árvore de classificação, o valor (classe) obtido pelo nó de término nos dados de treinamento é a moda das suas observações. Assim, a uma nova observação de dados atribui-se o valor da moda correspondente.
2. Ambas as árvores dividem o espaço preditor (variáveis independentes) em regiões distintas e não sobrepostas.
3. Ambas as árvores seguem uma abordagem “top-down” e “gananciosa” conhecida como divisão binária recursiva (“recursive binary splitting”). É chamada de “top-down” porque começa do topo da árvore quando todas as observações estão disponíveis em uma única região e, sucessivamente, divide o espaço preditor em dois novos ramos no sentido inferior da árvore. É conhecida como “gananciosa” porque o algoritmo se preocupa apenas com a divisão atual (procura a melhor variável disponível) e não com divisões futuras, que poderiam levar a uma árvore melhor.
4. Esse processo de divisão é continuado até que um critério de parada definido pelo usuário seja alcançado. Por exemplo: podemos dizer ao algoritmo que pare uma vez que o número de observações por nó torna-se menor que 50.
5. Em ambos os casos, o processo de divisão resulta em árvores totalmente crescidas até que os critérios de parada sejam atingidos. Mas, a árvore totalmente crescida é susceptível de sobrecarga de dados, levando a má precisão em dados não conhecidos. Isso nos traz à ‘poda’. A poda é uma das técnicas usadas para lidar com o sobreajuste.

Como fazer a divisão dos nós de uma árvore?

- A decisão de fazer as divisões dos nós afeta muito a precisão de uma árvore. Os critérios de decisão são diferentes para árvores de classificação e de regressão.

- Árvores de decisão usam vários algoritmos para decidir dividir um nó em dois ou mais sub-nós. A criação de sub-nós aumenta a homogeneidade dos sub-nós resultantes. Em outras palavras, pode-se dizer que a pureza do nó aumenta em relação à variável alvo. A árvore de decisão divide os nós em todas as variáveis disponíveis e seleciona a divisão que resulta em sub-nós mais homogêneos.



Índice Gini

O Coeficiente de Gini consiste em um número entre 0 e 1, onde 0 corresponde à completa igualdade (no caso do rendimento, por exemplo, toda a população recebe o mesmo salário) e 1 corresponde à completa desigualdade (onde uma pessoa recebe todo o rendimento e as demais nada recebem).

1. Funciona com a variável alvo categórica “Sucesso” ou “Falha”
2. Executa apenas divisões binárias
3. Quanto maior o valor de Gini, maior a homogeneidade
4. O CART (Árvore de Classificação e Regressão) usa o método Gini para criar divisões binárias

Passos para calcular o Gini de uma divisão:

1. Cálculo do Gini para sub-nós, calcule a soma dos quadrados da probabilidade de sucesso e da de fracasso (p ^ 2 + q ^ 2)
2. Calcule o Gini para a divisão, use a pontuação de Gini ponderada de cada nó dessa divisão

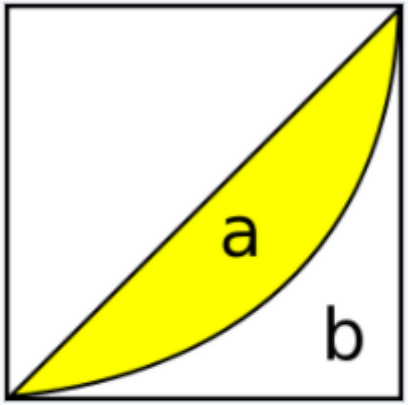
Índice Gini

Usando o exemplo anterior dos alunos que jogam ténis

Feminino	Masculino
Alunos = 10 Jogam tênis = 2 (20%)	Alunos = 120 Jogam tênis = 13 (65%)

0,2 = p sucesso 0,8 = p fracasso para o caso feminino

1. Gini para o sub-nó Feminino = (0,2 ^ 2) + (0,8 ^ 2) = 0,68
2. Gini para o sub-nó Masculino = (0,65 ^ 2) + (0,35 ^ 2) = 0,55
3. Gini ponderado da Divisão por Sexo = (10 / 30) * 0,68 + (20 / 30) * 0,55 = **0,59**



Representação gráfica do Coeficiente de Gini. O eixo horizontal representa a percentagem de pessoas , e o eixo vertical, a percentagem da renda. A diagonal representa a igualdade perfeita de renda, o coeficiente de Gini = a / (a + b).

Qui-Quadrado

É um algoritmo para descobrir a significância estatística entre as diferenças dos sub-nós e do nó pai. O qui-quadrado é medido pela soma dos quadrados das diferenças entre as freqüências observadas e esperadas da variável alvo.

1. Funciona com a variável alvo categórica “Sucesso” ou “Falha”.
2. Pode executar duas ou mais divisões.
3. Quanto maior o valor do qui-quadrado, maior é a significância estatística das diferenças entre sub nó e nó pai.
4. O qui-quadrado de cada nó é calculado usando a fórmula:

   Qui-quadrado = ((Real – Esperado) ^ 2 Esperado) ^ (1 / 2)
5. Gera a árvore chamada de CHAID (Chi-square Automatic Interaction Detector)

Passos para calcular o Qui-quadrado para uma divisão

1. Calcular o qui-quadrado para o nó individual calculando o desvio para ambos Sucesso e Falha
2. Calcular o qui-quadrado da divisão usando a soma de todos os qui-quadrados de Sucesso e Falha de cada nó da divisão

Qui-Quadrado

É um algoritmo para descobrir a significância estatística entre as diferenças dos sub-nós e do nó pai. O qui-quadrado é medido pela soma dos quadrados das diferenças entre as freqüências observadas e esperadas da variável alvo.

1. Funciona com a variável alvo categórica “Sucesso” ou “Falha”.
2. Pode executar duas ou mais divisões.
3. Quanto maior o valor do qui-quadrado, maior é a significância estatística das diferenças entre sub nó e nó pai.
4. O qui-quadrado de cada nó é calculado usando a fórmula:

   Qui-quadrado = ((Real – Esperado) ^ 2 Esperado) ^ (1 / 2)
5. Gera a árvore chamada de CHAID (Chi-square Automatic Interaction Detector)

Passos para calcular o Qui-quadrado para uma divisão

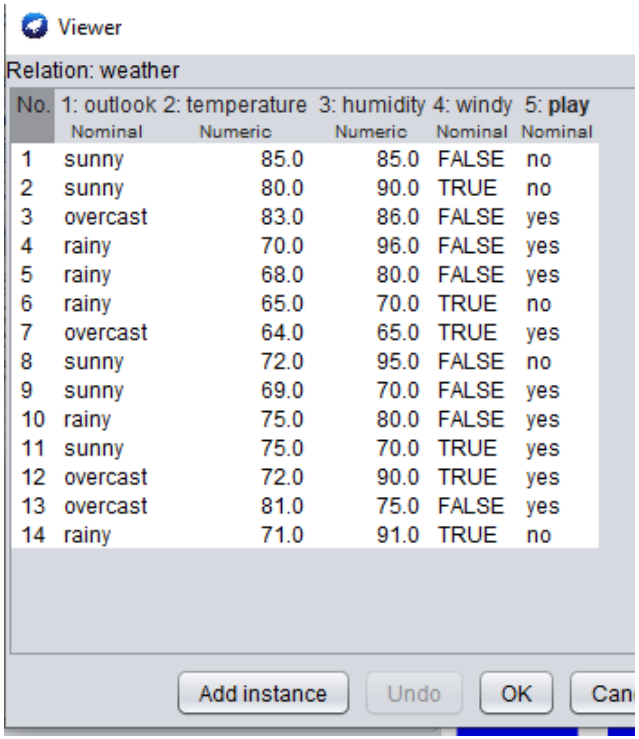
1. Calcular o qui-quadrado para o nó individual calculando o desvio para ambos Sucesso e Falha
2. Calcular o qui-quadrado da divisão usando a soma de todos os qui-quadrados de Sucesso e Falha de cada nó da divisão

Implementação em Python

EXEMPLO 1 – Weather (weather.numeric2.csv)

Data set

- **Objetivo da Inteligência Artificial**: Classificação/Previsão
- Algoritmos de Inteligência Artificial: Árvores de Decisão e Redes Neuronais Artificiais
- Ferramentas a utilizar: Weka Tool e Microsoft Azure Machine Learning;
- Conjunto de Dados (“praticar desporto”); variável independente (“Classe”) “play”:



No.	1: outlook Nominal	2: temperature Numeric	3: humidity Numeric	4: windy Nominal	5: play Nominal
1	sunny	85.0	85.0	FALSE	no
2	sunny	80.0	90.0	TRUE	no
3	overcast	83.0	86.0	FALSE	yes
4	rainy	70.0	96.0	FALSE	yes
5	rainy	68.0	80.0	FALSE	yes
6	rainy	65.0	70.0	TRUE	no
7	overcast	64.0	65.0	TRUE	yes
8	sunny	72.0	95.0	FALSE	no
9	sunny	69.0	70.0	FALSE	yes
10	rainy	75.0	80.0	FALSE	yes
11	sunny	75.0	70.0	TRUE	yes
12	overcast	72.0	90.0	TRUE	yes
13	overcast	81.0	75.0	FALSE	yes
14	rainy	71.0	91.0	TRUE	no

rows 14 columns 5

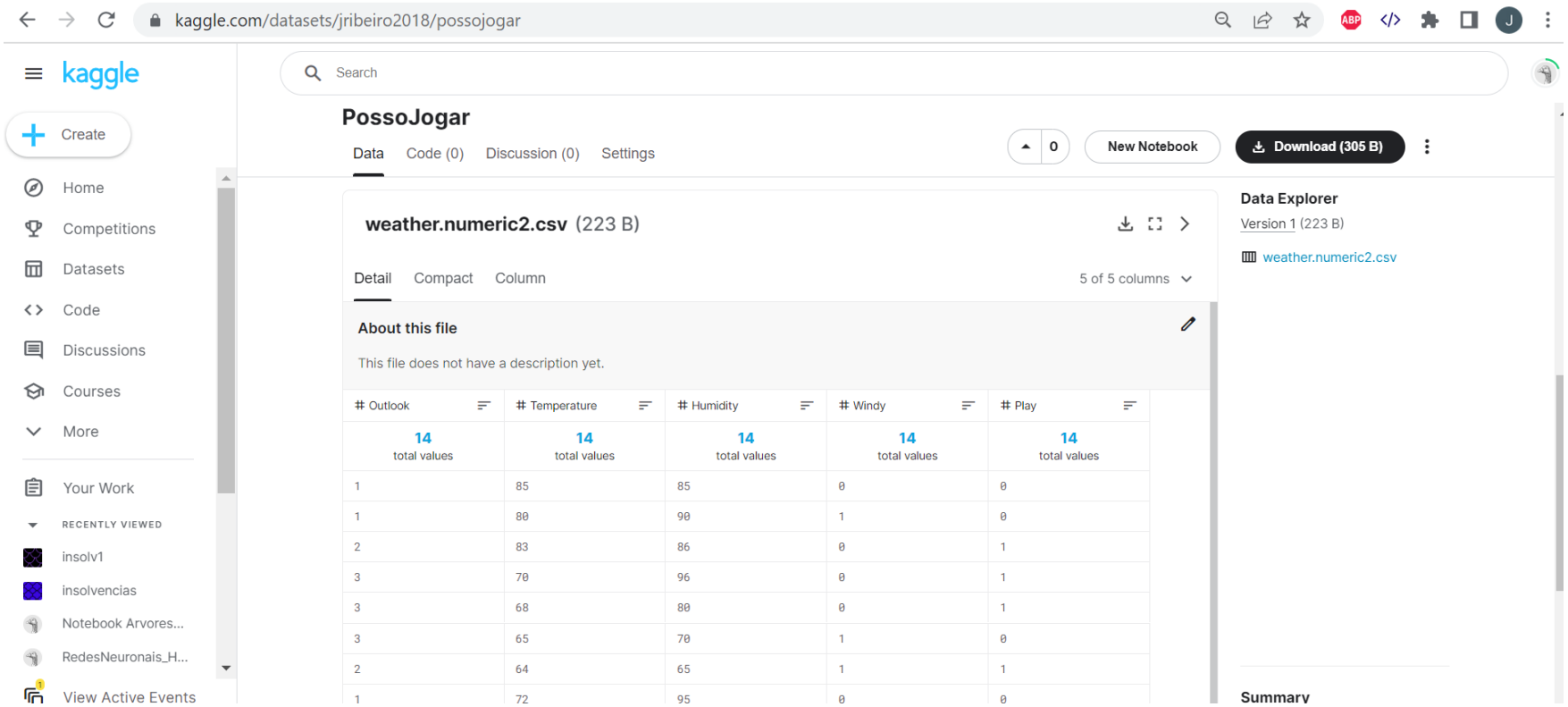
view as

outlook	temperature	humidity	windy	play
sunny	85	85	false	no
sunny	80	90	true	no
overcast	83	86	false	yes
rainy	70	96	false	yes
rainy	68	80	false	yes
rainy	65	70	true	no
overcast	64	65	true	yes
sunny	72	95	false	no
sunny	69	70	false	yes
rainy	75	80	false	yes
sunny	75	70	true	yes
overcast	72	90	true	yes
overcast	81	75	false	yes
rainy	71	91	true	no

	A	B	C	D
1	Outlook,Temperature,Humidity,Windy,Play			
2	1,85,85,0,0			
3	1,80,90,1,0			
4	2,83,86,0,1			
5	3,70,96,0,1			
6	3,68,80,0,1			
7	3,65,70,1,0			
8	2,64,65,1,1			
9	1,72,95,0,0			
10	1,69,70,0,1			
11	3,75,80,0,1			
12	1,75,70,1,1			
13	2,72,90,1,1			
14	2,81,75,0,1			
15	3,71,91,1,0			

EXEMPLO 1 – Weather (weather.numeric2.csv)

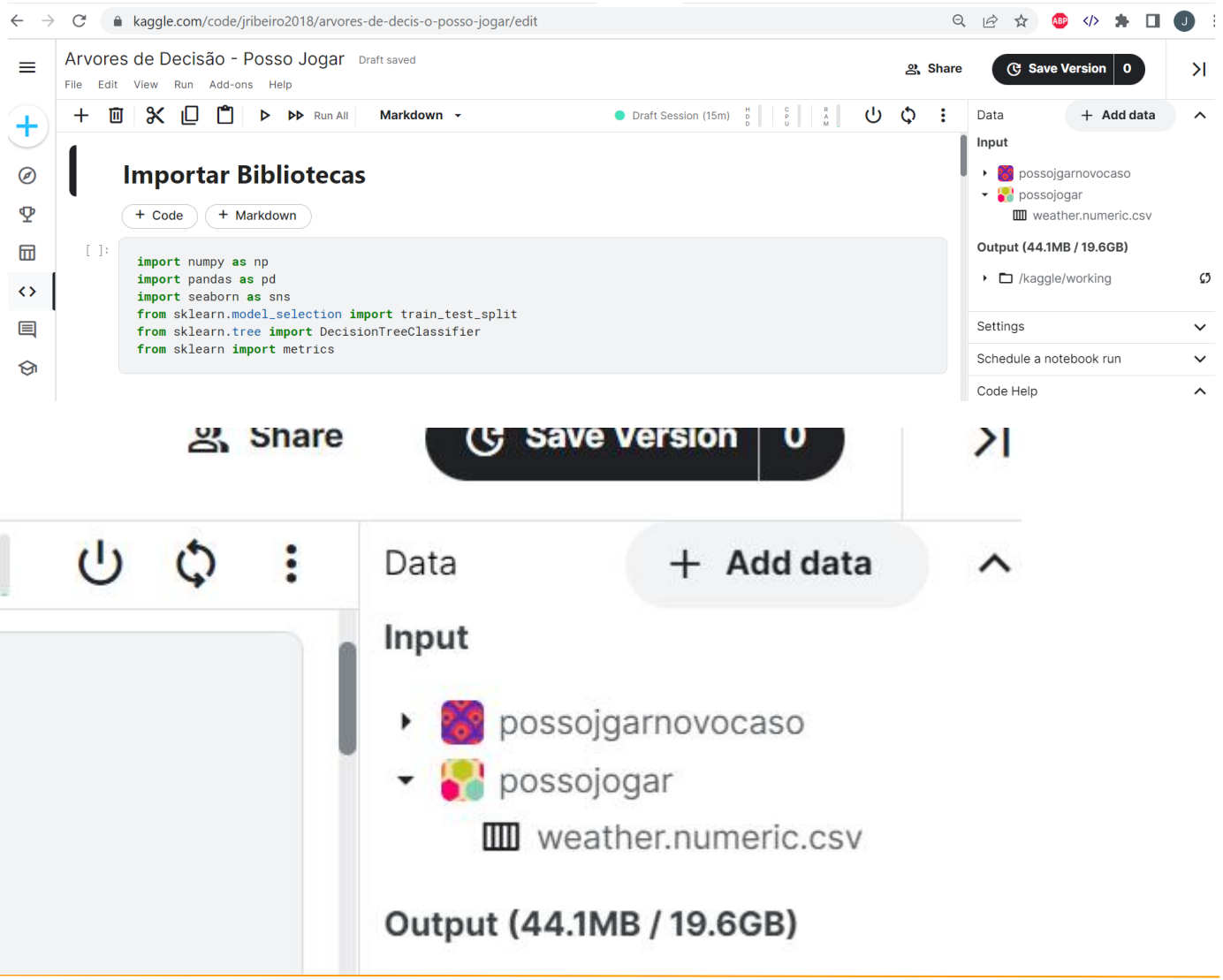
Data set



EXEMPLO 1 – Weather (weather.numeric2.csv)

Data set

	A	B	C	D
1	outlook,temperature, humidity,windy,play			
2	sunny,85,85,FALSE,no			
3	sunny,80,90,TRUE,no			
4	overcast,83,86,FALSE,yes			
5	rainy,70,96,FALSE,yes			
6	rainy,68,80,FALSE,yes			
7	rainy,65,70,TRUE,no			
8	overcast,64,65,TRUE,yes			
9	sunny,72,95,FALSE,no			
10	sunny,69,70,FALSE,yes			
11	rainy,75,80,FALSE,yes			
12	sunny,75,70,TRUE,yes			
13	overcast,72,90,TRUE,yes			
14	overcast,81,75,FALSE,yes			
15	rainy,71,91,TRUE,no			

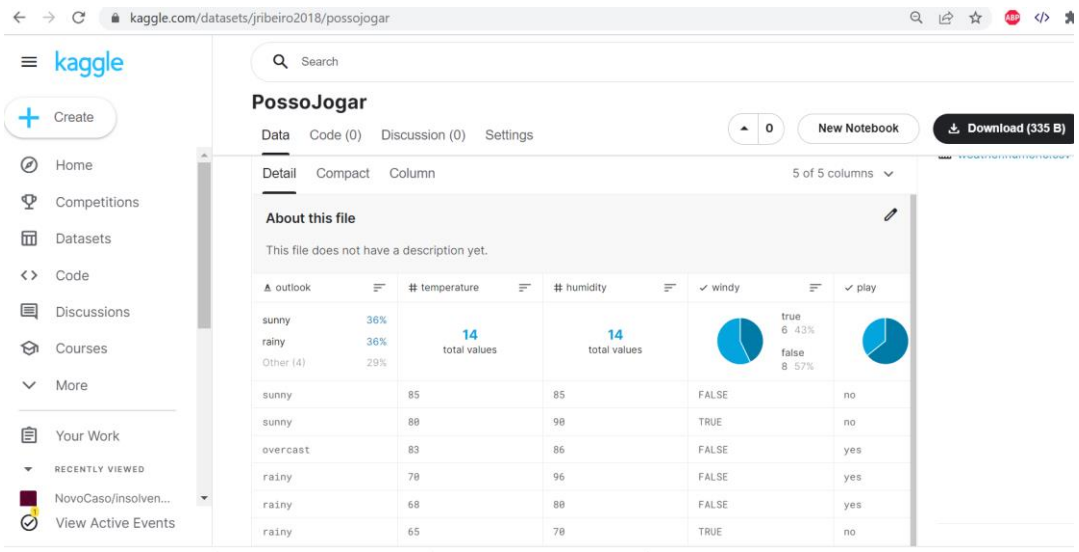


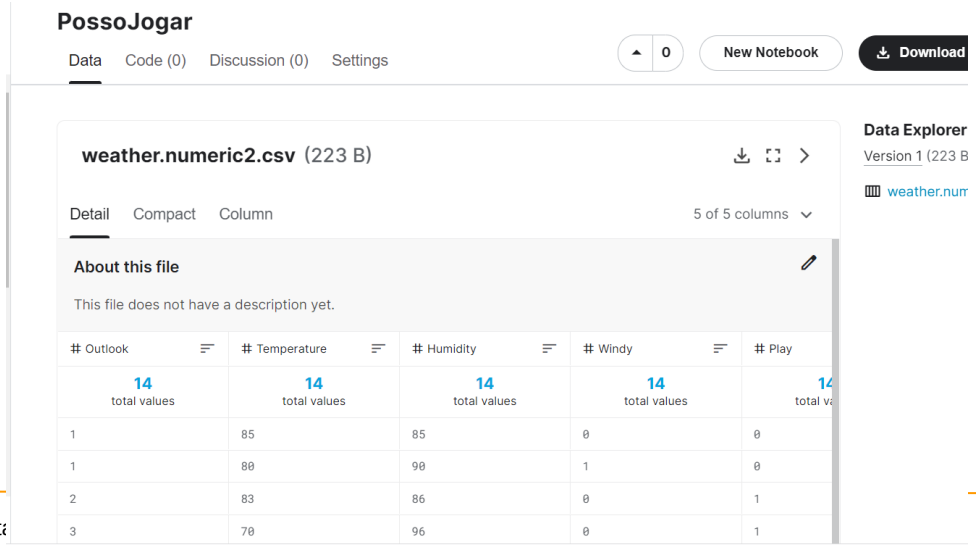
EXEMPLO 1 – Weather (weather.numeric2.csv)

Data set

	A	B	C	D
1	outlook,temperature, humidity,windy,play			
2	sunny,85,85,FALSE,no			
3	sunny,80,90,TRUE,no			
4	overcast,83,86,FALSE,yes			
5	rainy,70,96,FALSE,yes			
6	rainy,68,80,FALSE,yes			
7	rainy,65,70,TRUE,no			
8	overcast,64,65,TRUE,yes			
9	sunny,72,95,FALSE,no			
10	sunny,69,70,FALSE,yes			
11	rainy,75,80,FALSE,yes			
12	sunny,75,70,TRUE,yes			
13	overcast,72,90,TRUE,yes			
14	overcast,81,75,FALSE,yes			
15	rainy,71,91,TRUE,no			

	A	B	C	D
1	Outlook,Temperature,Humidity,Windy,Play			
2	1,85,85,0,0			
3	1,80,90,1,0			
4	2,83,86,0,1			
5	3,70,96,0,1			
6	3,68,80,0,1			
7	3,65,70,1,0			
8	2,64,65,1,1			
9	1,72,95,0,0			
10	1,69,70,0,1			
11	3,75,80,0,1			
12	1,75,70,1,1			
13	2,72,90,1,1			
14	2,81,75,0,1			
15	3,71,91,1,0			





- **EXEMPLO 1 – Weather (weather.numeric2.csv)**
- **Data set**

Importar Bibliotecas

+ Code + Markdown

[ ]:

```
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics
```

EXEMPLO 1 – Weather (weather.numeric2.csv)

Data set

Importar Dataset

```
dPossoJogar = pd.read_csv("../input/possojogar/weather.numeric2.csv")
dPossoJogar.head(2)     #Mostra as primeiras n linhas do conjunto de dados
dPossoJogar.tail(5)     #Mostra as últimas n linhas do conjunto
```

[59]:

	Outlook	Temperature	Humidity	Windy	Play
9	3	75	80	0	1
10	1	75	70	1	1
11	2	72	90	1	1
12	2	81	75	0	1
13	3	71	91	1	0

EXEMPLO 1 – Weather (weather.numeric2.csv)

Data set

Distribuição de frequência de valores em variáveis

```
for col in dPossoJogar:

    print(dPossoJogar[col].value_counts())
```

```
sunny       5
rainy       5
overcast    4
Name: outlook, dtype: int64
72    2
75    2
85    1
80    1
83    1
70    1
68    1
65    1
64    1
69    1
81    1
71    1
Name: temperature, dtype: int64
```

Acerca do Dataset

[7]:
```
# Dimensão do Dataset
dPossoJogar.shape
```

[7]: (14, 5)

```
# Primeiras 5 linhas do Dataset
dPossoJogar.head()
```

[8]:

	outlook	temperature	humidity	windy	play
0	sunny	85	85	False	no
1	sunny	80	90	True	no
2	overcast	83	86	False	yes
3	rainy	70	96	False	yes

EXEMPLO 1 – Weather (weather.numeric2.csv)

Data set

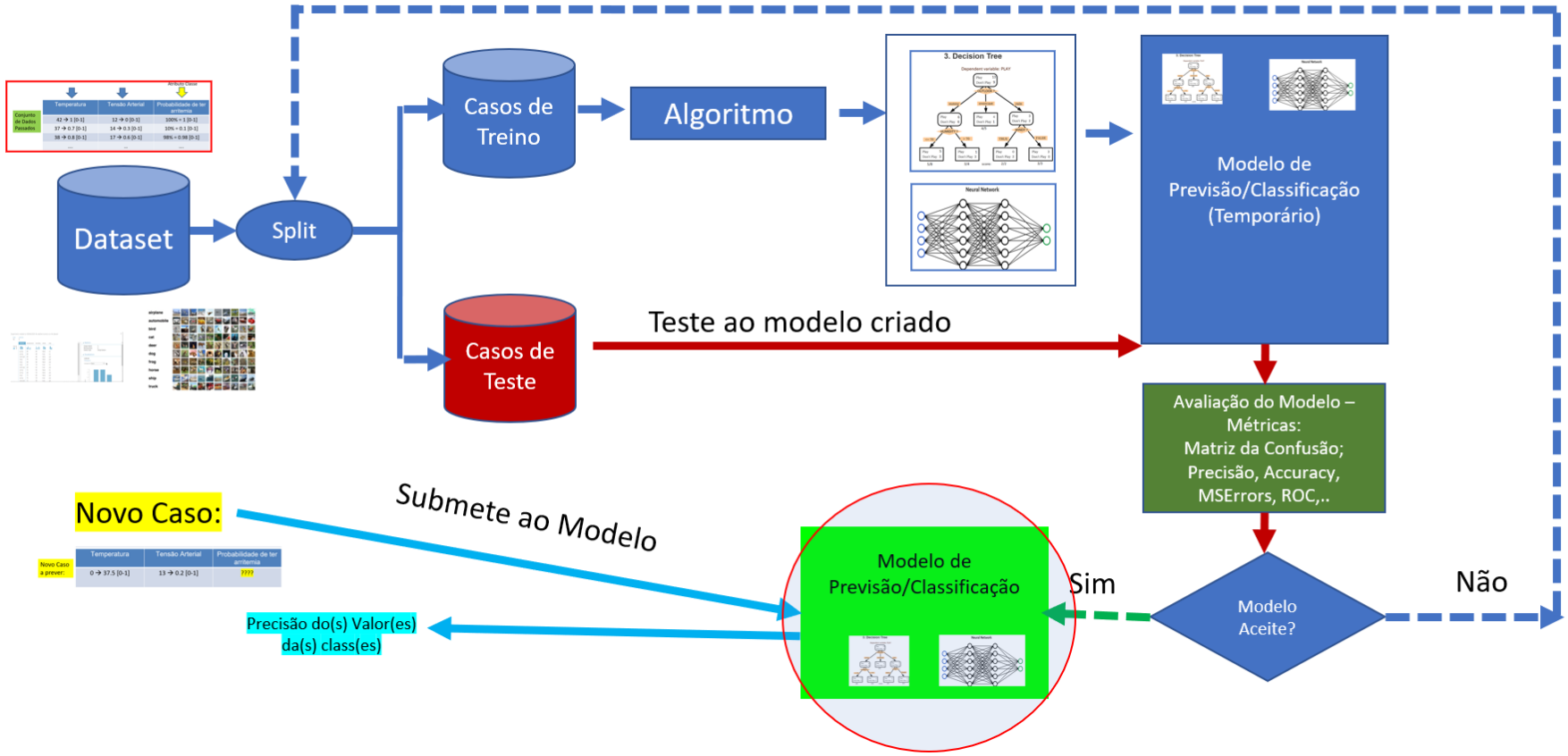
```
dPossoJogar.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14 entries, 0 to 13
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Outlook      14 non-null     int64
 1   Temperature  14 non-null     int64
 2   Humidity     14 non-null     int64
 3   Windy        14 non-null     int64
 4   Play         14 non-null     int64
dtypes: int64(5)
memory usage: 688.0 bytes
```

EXEMPLO 1 – Weather (weather.numeric2.csv)

Usando Python Kaggle – O Modelo – Árvores de Decisão:

- **Construção de um classificador/Previsão –**



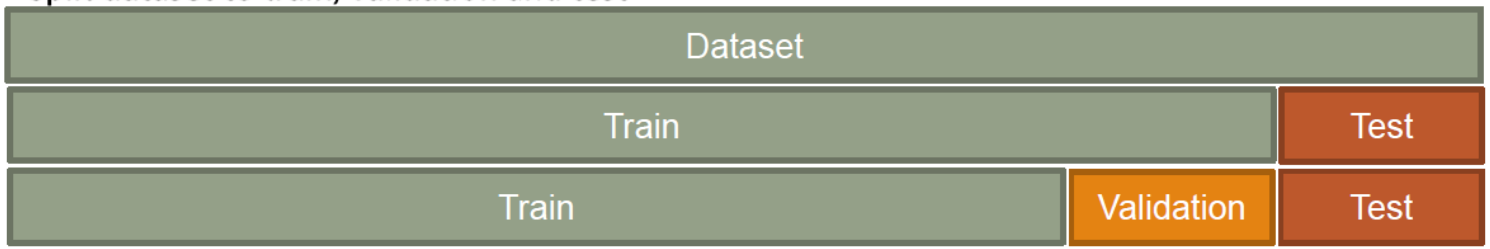
Usando Python Kaggle – O Modelo – Árvores de Decisão:

- Construção de um classificador/Previsão – Cross Validation/Split

Machine Learning Pipeline

1. **Prepare your dataset**

- Split dataset to train, validation and test

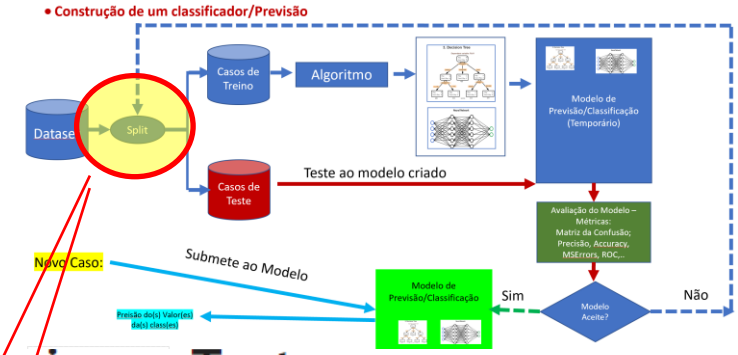


- Normalize your dataset to $\mu = 0$ and $\sigma = 1$
- Remove false data (NaN)
- Randomly shuffle the dataset

EXEMPLO 1 – Weather (weather.numeric2.csv)

Usando Python Kaggle – O Modelo – Árvores de Decisão:

- Construção de um classificador/Previsão – Cross Validation/Split



Dividir o dataset em dois conjuntos - Treino e Teste

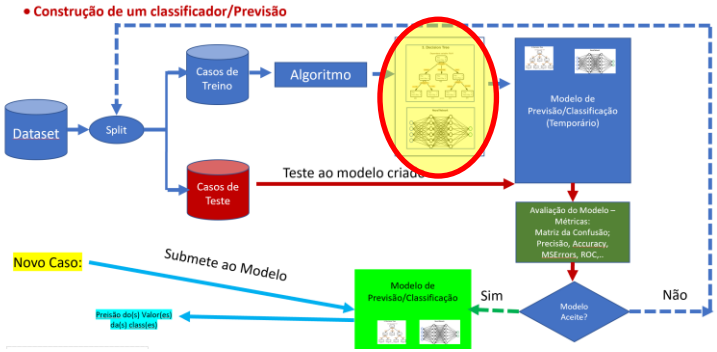
```
x = dPossoJogar.drop('Play', axis=1)    #Todas as tabelas que são usadas para fazer a predict
y = dPossoJogar.Play                    #Resultados Posso Jogar corresponde a 1 não Posso JOgar a 0

## Funcao train_test_split
x_train, x_teste, y_train, y_teste = train_test_split(x, y, test_size=0.50, random_state=18)
```

+ Code + Markdown

```
#cross-validation settings
kfold = model_selection.KFold(n_splits=10, random_state=seed)
```

EXEMPLO 1 – Weather (weather.numeric2.csv)

Usando Python Kaggle – O Modelo – Árvores de Decisão:



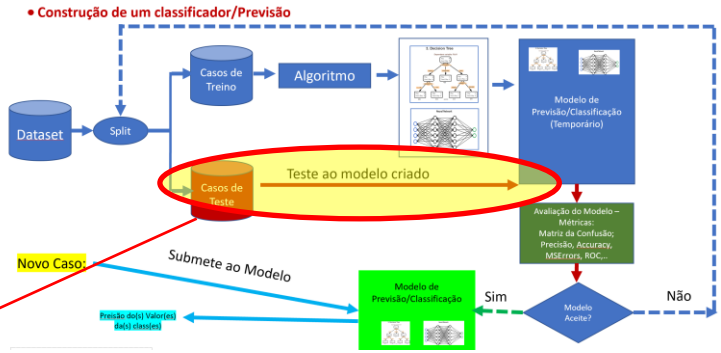
Treinar o modelo criado - DecisionTree_Class_Model

```
DecisionTree_Class_Model = DecisionTreeClassifier()
DecisionTree_Class_Model.fit(x_train, y_train)
```

```
[6]: DecisionTreeClassifier()
```

EXEMPLO 1 – Weather (weather.numeric2.csv)

Usando Python Kaggle – O Modelo – Árvores de Decisão:



Previsão

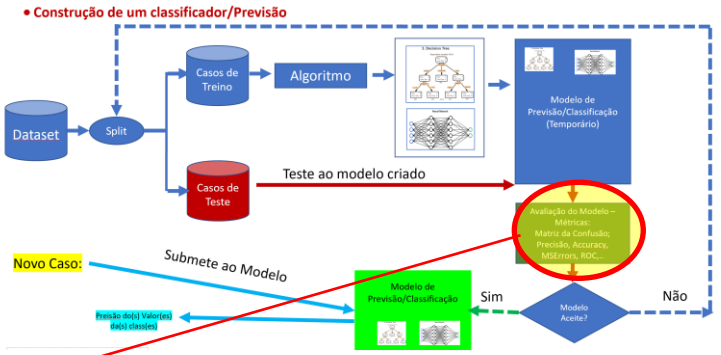
```
y_pred = DecisionTree_Class_Model.predict(x_teste)
print("Previsão", y_pred)
```

```
Previsão [1 1 1 1 1 0 1]
```

+ Code + Markdown

EXEMPLO 1 – Weather (weather.numeric2.csv)

Usando Python Kaggle – O Modelo – Árvores de Decisão:



Resultado - Accuracy

```
accuracy = metrics.accuracy_score(y_teste, y_pred)
print("Acuracy - Resultado:", accuracy)
```

```
Acuracy - Resultado: 0.42857142857142855
```

```
Precision: 0.696970
Recall: 0.541176
F1 score: 0.609272
```

$$Precision = \frac{TP}{TP+FP} \times 100\%$$

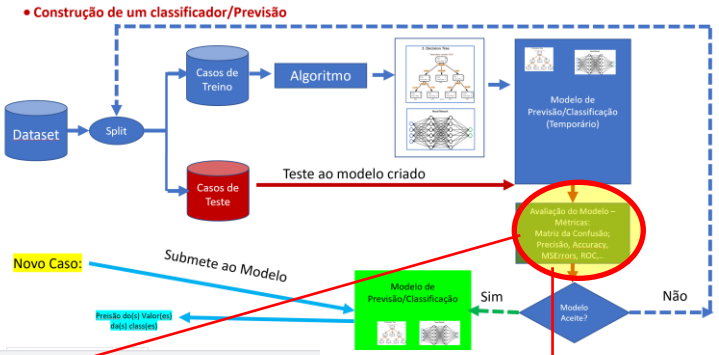
$$Recall = \frac{TP}{TP+FN} \times 100\%$$

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall}$$

```
X_train, X_test, y_train, y_test = model_selection
model.fit(X_train, y_train)
precision = precision_score(y_test, pred)
print('Precision: %f' % precision)
# recall: tp / (tp + fn)
recall = recall_score(y_test, pred)
print('Recall: %f' % recall)
# f1: tp / (tp + fp + fn)
f1 = f1_score(y_test, pred)
```

EXEMPLO 1 – Weather (weather.numeric2.csv)

Usando Python Kaggle – O Modelo – Árvores de Decisão:



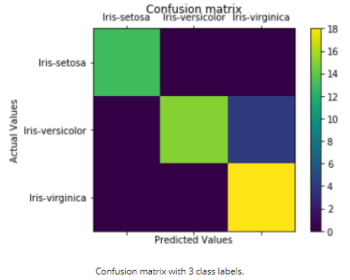
Confusion Matrix

```
from sklearn.metrics import confusion_matrix

pd.DataFrame(
    confusion_matrix(y_teste, y_pred),
    columns=['Previsão de Jogar', 'Previsão de Não Jogar'],
    index=['Realidade de Jogar', 'Realidade de Não Jogar']
)
```

[14]:

	Previsão de Jogar	Previsão de Não Jogar
Realidade de Jogar	0	3
Realidade de Não Jogar	1	3

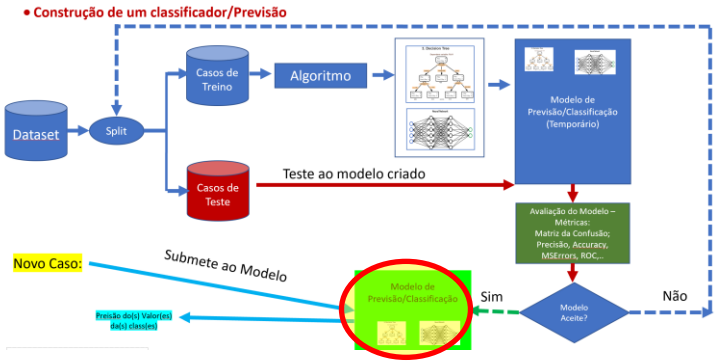
```
[[13  0  0]
 [ 0 15  4]
 [ 0  0 18]]
```



Confusion matrix with 3 class labels.

EXEMPLO 1 – Weather (weather.numeric2.csv)

Usando Python Kaggle – O Modelo – Árvores de Decisão:



Criar o Modelo e Gerar a Árvore de Decisão

```
from sklearn import tree

model = tree.DecisionTreeClassifier()
model.fit(x_train, y_train)

tree.export_graphviz(model, out_file='tree.dot', feature_names=x.columns)
from subprocess import call

call(['dot', '-T', 'png', 'tree.dot', '-o', 'tree.png'])
```

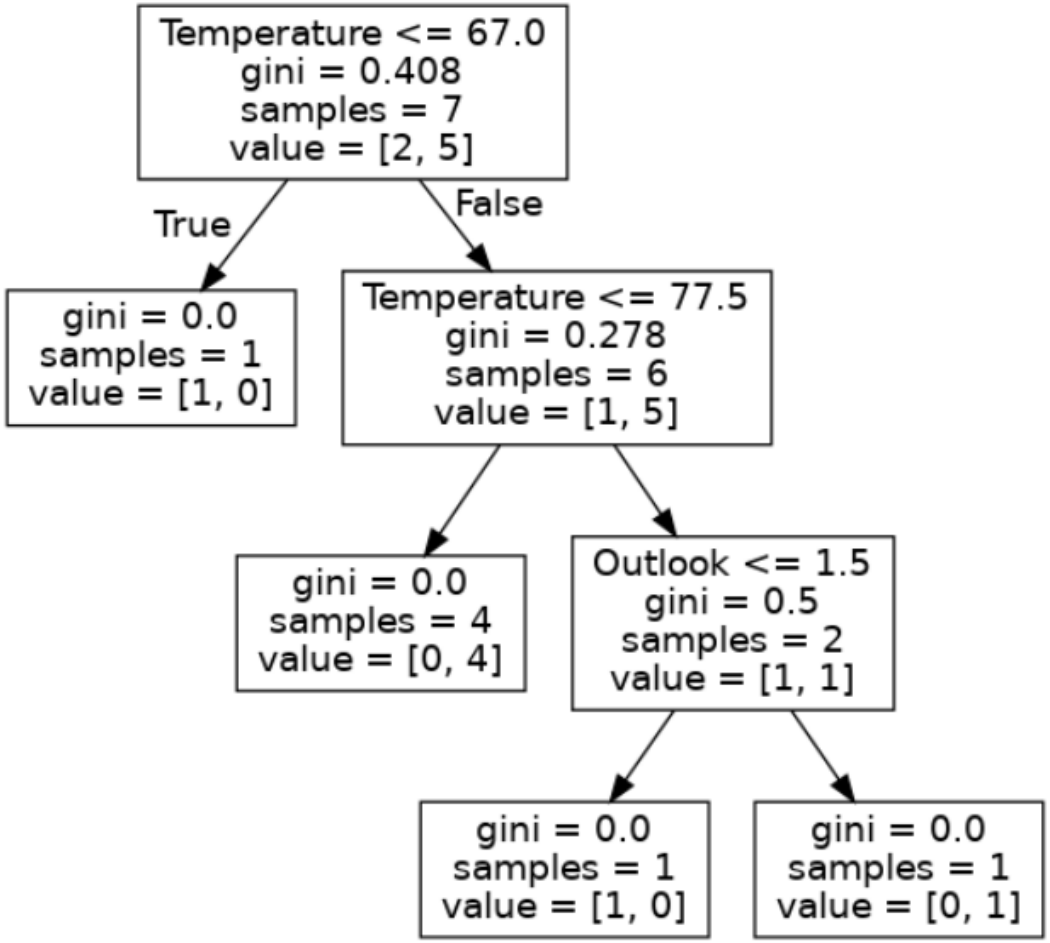
```
: 0
```

EXEMPLO 1 – Weather (weather.numeric2.csv)

Usando Python Kaggle – O Modelo – Árvores de Decisão:

```
from IPython.display import Image
Image(filename = 'tree.png')
```

16]:



Guardar modelo num ficheiro Output

```
import pickle

filename = 'model-output.sav'
pickle.dump(model, open(filename, 'wb'))
```

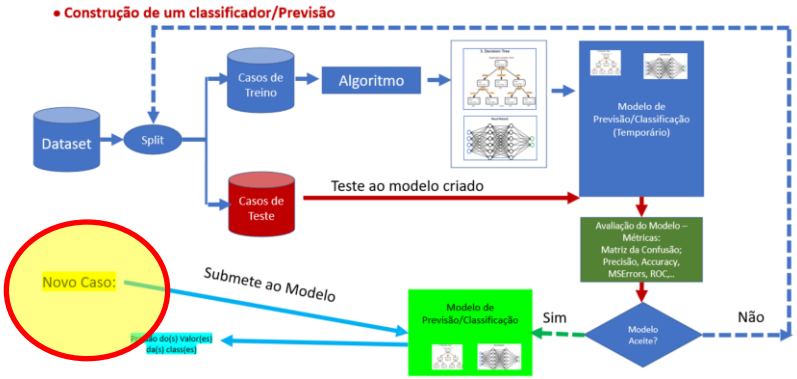
Carregar o Modelo e ver o Seu Resultado

```
loaded_model = pickle.load(open(filename, 'rb'))
result = loaded_model.score(x_teste, y_teste)
print("Score do Modelo:",result)
```

```
Score do Modelo: 0.5714285714285714
```

EXEMPLO 1 – Weather (weather.numeric2.csv)

Usando Python Kaggle – O Modelo – Árvores de Decisão:



Fazer uma previsão final

```
import pickle
loaded_model = pickle.load(open('./model-output.sav', 'rb'))    #Importar o modelo guardado

dNew = pd.read_csv("../input/possojogarnovocaso/weather.numeric_novocaso.csv")
dNew = dNew.drop('Play  ', axis=1)   #Remover a tabela Insolvencia
dNew.head(2)     #Mostra as primeiras n linhas do conjunto de dados

xNew = dNew
yNew = loaded_model.predict(xNew)

print("Previsão de Jogar:", yNew[0])
```

```
Previsão de Jogar: 0
```

Bibliografia

- https://www.kaggle.com/code/hamelg/python-for-data-29-decision-trees/notebook
- http://math.furman.edu/~dcs/courses/math47/R/library/lmtest/html/dwtest.html
- https://data.library.virginia.edu/diagnostic-plots/
- https://community.rstudio.com/t/error-after-r-update-lib-c-program-files-r-r-3-5-0-library-is-not-writable/7947/2
- https://medium.com/data-hackers/implementando-regress%C3%A3o-linear-simples-em-python-91df53b920a8

o teu • de partida

www.ipvc.pt