

Aprendizado de Máquina Supervisionado –IMD3002

Aula 15 – Árvore de Decisão

João Carlos Xavier Júnior

jcxavier@imd.ufrn.br

Árvore de Decisão

Árvore de Decisão

❑ Definição:

- ❖ Um fluxograma com a estrutura de uma árvore.
- ❖ **Nó** interno representa um **teste sobre um atributo**.
- ❖ Cada **ramo** representa um **resultado do teste**.
- ❖ **Folhas** representam as **classes**.

❑ A geração de uma árvore consiste de duas fases:

- ❖ Construção da árvore (particionamento de atributos).
- ❖ Fase da poda (identifica e remove ruídos ou *outliers*).

❑ Uso da árvore: classificação de amostras desconhecidas.

- ❖ Testa os valores dos atributos da amostra “contra” a árvore.

Árvore de Decisão

❑ Algoritmo Básico:

- ❖ A árvore é construída recursivamente no sentido *top-down* (divisão para conquista).
- ❖ Atributos “**testes**” são selecionados com base em heurísticas ou medidas estatísticas (ex., ganho de informação).
- ❖ Medidas utilizadas: Entropia e índice Gini.

❑ Condições de parada do particionamento:

- ❖ Todas as amostras de um nó pertencem a **mesma classe**.
- ❖ Não existem mais atributos para particionamento.

Árvore de Decisão

□ Entropia:

- ❖ É uma medida de aleatoriedade (*impureza*) de uma variável.
- ❖ A entropia de uma variável nominal X que pode tomar i valores:

$$\textit{entropia}(X) = - \sum_i p_i \log_2 p_i$$

Árvore de Decisão

Conjunto de treinamento:

Nº exemplar	Céu	Temperatura	Umidade	Vento	Classe
1	sol	alta	alta	não	não joga
2	sol	alta	alta	sim	não joga
3	nublado	alta	alta	não	joga
4	chuva	alta	alta	não	joga
5	chuva	baixa	normal	não	joga
6	chuva	baixa	normal	sim	não joga
7	nublado	baixa	normal	sim	joga
8	sol	suave	alta	não	não joga
9	sol	baixa	normal	não	joga
10	chuva	suave	normal	não	joga
11	sol	suave	normal	sim	joga
12	nublado	suave	alta	sim	joga
13	nublado	alta	normal	não	joga
14	chuva	suave	alta	sim	não joga

9 instâncias
para jogar

5 instâncias
para não jogar

Árvore de Decisão

❑ Exemplo:

- ❖ Suponha que S é uma coleção de 14 instâncias, onde 9 são positivas e 5 são negativas.
- ❖ Notação: $[9+, 5-]$
- ❖ A entropia de S em relação a esta classificação booleana é
- ❖ dada por:

$$\begin{aligned} \text{entropia}([9+, 5-]) = \\ - \left(\frac{9}{14} \right) \log_2 \left(\frac{9}{14} \right) - \left(\frac{5}{14} \right) \log_2 \left(\frac{5}{14} \right) = 0,940 \end{aligned}$$

Árvore de Decisão

❑ Calcular o Ganho (S, Vento):

❖ Valores de (Vento) = {não ou weak, sim ou strong}.

❖ S: [9+, 5-]

▪ $S_{\text{não}} = [3+, 2-]$

▪ $S_{\text{sim}} = [3+, 6-]$

$$\text{Ganho}(S, \text{Vento}) = E(S) - \sum_{v \in \{\text{não}, \text{sim}\}} \frac{|S_v|}{|S|} E(Sv)$$

$$\text{Ganho}(S, \text{Vento}) = E(S) - (8/14) E(\text{não}) - (6/14) E(\text{sim})$$

$$\text{Ganho}(S, \text{Vento}) = - (6/8) * \log_2(6/8) - (2/8) * \log_2(2/8) \Rightarrow 0,811$$

$$\text{Ganho}(S, \text{Vento}) = - (3/6) * \log_2(3/6) - (3/6) * \log_2(3/6) \Rightarrow 1,000$$

$$\text{Ganho}(S, \text{Vento}) = 0,940 - (8/14) 0,811 - (6/14) 1,000 = 0,048$$

Árvore de Decisão

❑ Qual atributo deve ser primeiro nó da árvore?

- ❖ Determinar o ganho de informação (Gain) para cada atributo.
- ❖ Selecionar aquele cujo ganho de informação é o **mais alto**:
 - $\text{Ganho}(S, \text{Outlook}) = 0,246$
 - $\text{Ganho}(S, \text{Humidity}) = 0,151$
 - $\text{Ganho}(S, \text{Wind}) = 0,048$
 - $\text{Ganho}(S, \text{Temperature}) = 0,029$

Árvore de Decisão

□ Ganho de informação (atributos numéricos):

- ❖ Um teste num atributo numérico produz uma partição binária do conjunto de exemplos:
 - Instâncias onde $\text{valor_do_atributo} < \text{ponto_referência}$;
 - Instâncias onde $\text{valor_do_atributo} > \text{ponto_referência}$.
- ❖ Escolha do ponto de referência:
 - Ordenar os exemplos por ordem crescente dos valores do atributo numérico.
 - Qualquer ponto intermediário entre dois valores diferentes e consecutivos dos valores observados no conjunto de treinamento pode ser utilizado como possível ponto de referência.

Árvore de Decisão

❑ Conjunto de treinamento:

Nº exemplar	Céu	Temperatura	Umidade	Vento	Classe
1	sol	85	85	não	não joga
2	sol	80	90	sim	não joga
3	nublado	83	78	não	joga
4	chuva	70	96	não	joga
5	chuva	68	80	não	joga
6	chuva	65	70	sim	não joga
7	nublado	64	65	sim	joga
8	sol	72	95	não	não joga
9	sol	69	70	não	joga
10	chuva	75	80	não	joga
11	sol	75	70	sim	joga
12	nublado	72	90	sim	joga
13	nublado	81	75	não	joga
14	chuva	71	80	sim	não joga

9 instâncias
para jogar

5 instâncias
para não jogar

Árvore de Decisão

❑ Calculando ganho (atributo Umidade):

Umidade	Classe
85	não
90	não
78	sim
96	sim
80	sim
70	não
65	sim
95	não
70	sim
80	sim
70	sim
90	sim
75	sim
80	não



Umidade	Classe	
65	sim	68
70	não	70
70	sim	70
70	sim	73
75	sim	77
78	sim	79
80	sim	80
80	sim	80
80	não	83
85	não	88
90	não	90
90	sim	93
95	não	96
96	sim	

$$v_p = \frac{(v_i + v_j)}{2}$$



$$v_p = \frac{(75 + 78)}{2} \Rightarrow 77$$

Árvore de Decisão

❑ Calculando ganho (atributo Umidade):

Umidade	Classe	
65	sim	68
70	não	70
70	sim	70
70	sim	73
75	sim	77
78	sim	79
80	sim	80
80	sim	80
80	não	83
85	não	88
90	não	90
90	sim	93
95	não	96
96	sim	



Umidade $< 77 \Rightarrow$ 5 instâncias

Onde:

4/5 (sim) e 1/5 (não)

Umidade $> 77 \Rightarrow$ 9 instâncias

Onde:

5/9 (sim) e 4/9 (não)

Árvore de Decisão

❑ Calculando ganho (atributo Umidade):

$$\text{info}(\text{umidade} < 77) = -\frac{4}{5} \log_2\left(\frac{4}{5}\right) - \frac{1}{5} \log_2\left(\frac{1}{5}\right) \Rightarrow 0,721928$$

$$\text{info}(\text{umidade} > 77) = -\frac{5}{9} \log_2\left(\frac{5}{9}\right) - \frac{4}{9} \log_2\left(\frac{4}{9}\right) \Rightarrow 0,991076$$

$$\text{info}(\text{umidade}) = \frac{5}{14} * 0,721928 + \frac{9}{14} * 0,991076 \Rightarrow 0,894952$$

$$\text{Ganho}(\text{umidade}) = 0,940 - 0,894952 \Rightarrow 0,045$$

Árvore de Decisão

❑ Calculando ganho (atributo Umidade):

Umidade	Classe	
65	sim	68
70	não	70
70	sim	70
70	sim	73
75	sim	77
78	sim	79
80	sim	80
80	sim	80
80	não	83
85	não	88
90	não	90
90	sim	93
95	não	96
96	sim	



?

?

?

?

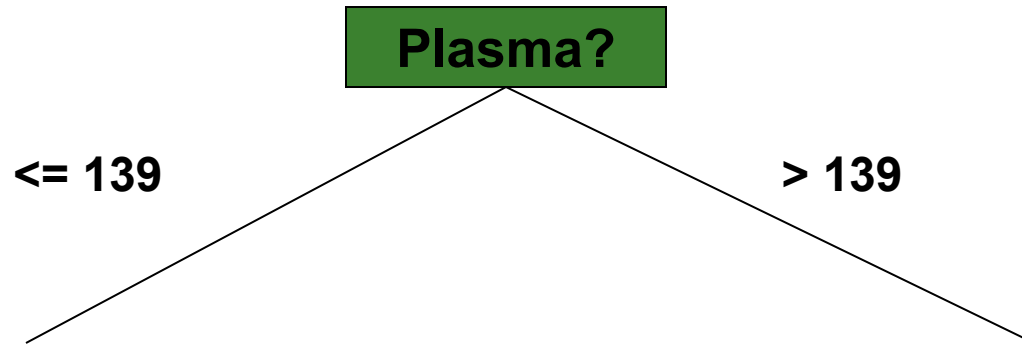
Repetir o processo para outros pontos intermediários ...

Árvore de Decisão

□ Treinamento: Pima Indians Diabetes Database (subset)

#	Pregnant	Plasma	BloodPressure	Skin	Insulin	BodyMass	Pedigree	Age	Diagnóstico
1	6	148	72	35	0	33,6	627	50	tested_positive
2	1	85	66	29	0	26,6	351	31	tested_negative
3	8	183	64	0	0	23,3	672	32	tested_positive
4	1	89	66	23	94	28,1	167	21	tested_negative
5	0	137	40	35	168	43,1	2288	33	tested_positive
6	5	116	74	0	0	25,6	201	30	tested_negative
7	3	78	50	32	88	31,0	248	26	tested_positive
8	10	115	0	0	0	35,3	134	29	tested_negative
9	2	197	70	45	543	30,5	158	53	tested_positive
10	4	110	92	0	0	37,6	191	30	tested_negative
11	10	168	74	0	0	38,0	537	34	tested_positive
12	10	139	80	0	0	27,1	1441	57	tested_negative
13	1	189	60	23	846	30,1	398	59	tested_positive
14	5	166	72	19	175	25,8	587	51	tested_positive
15	7	100	0	0	0	30,0	484	32	tested_positive
16	0	118	84	47	230	45,8	551	31	tested_positive
17	7	107	74	0	0	29,6	254	31	tested_positive
18	1	103	30	38	83	43,3	183	33	tested_negative
19	1	115	70	30	96	34,6	529	32	tested_positive
20	3	126	88	41	235	39,3	704	27	tested_negative

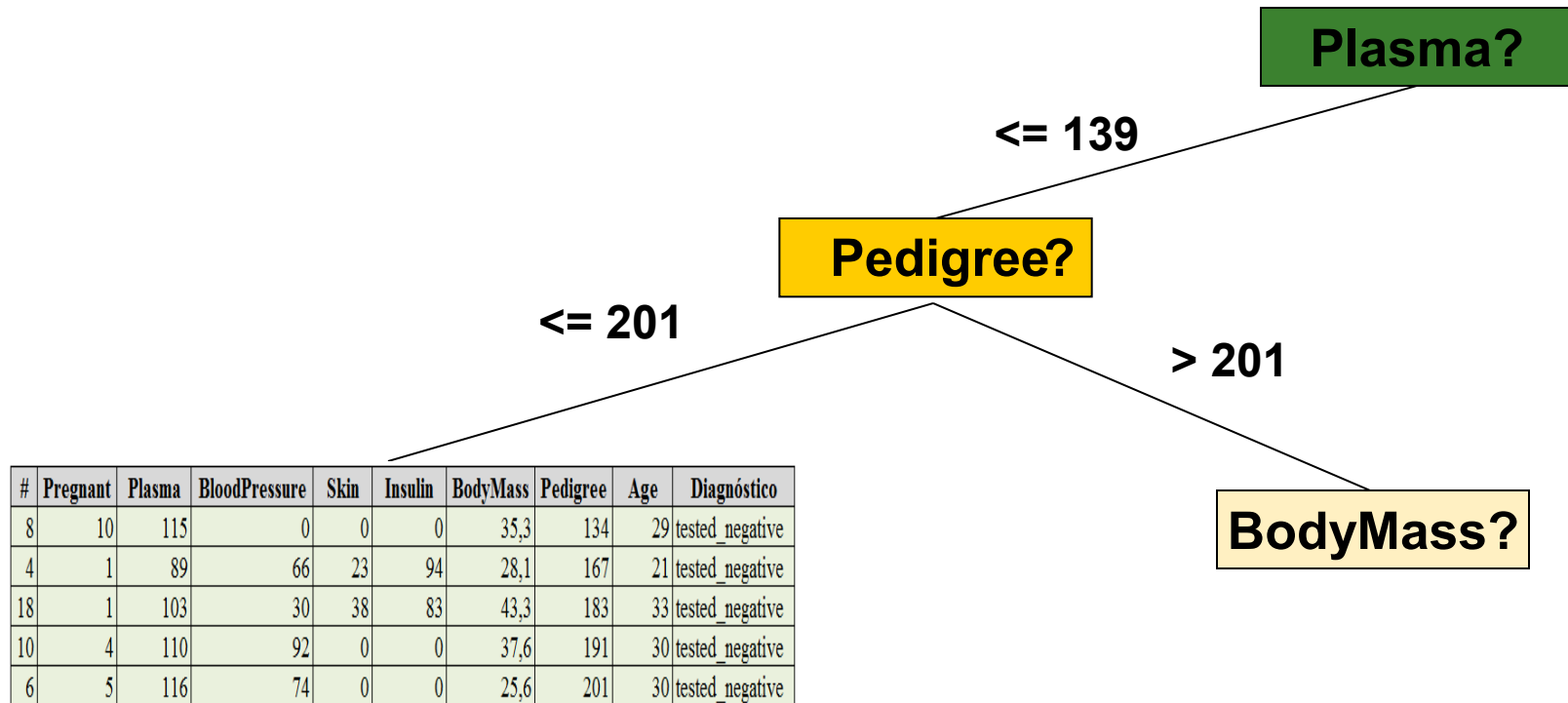
Árvore de Decisão



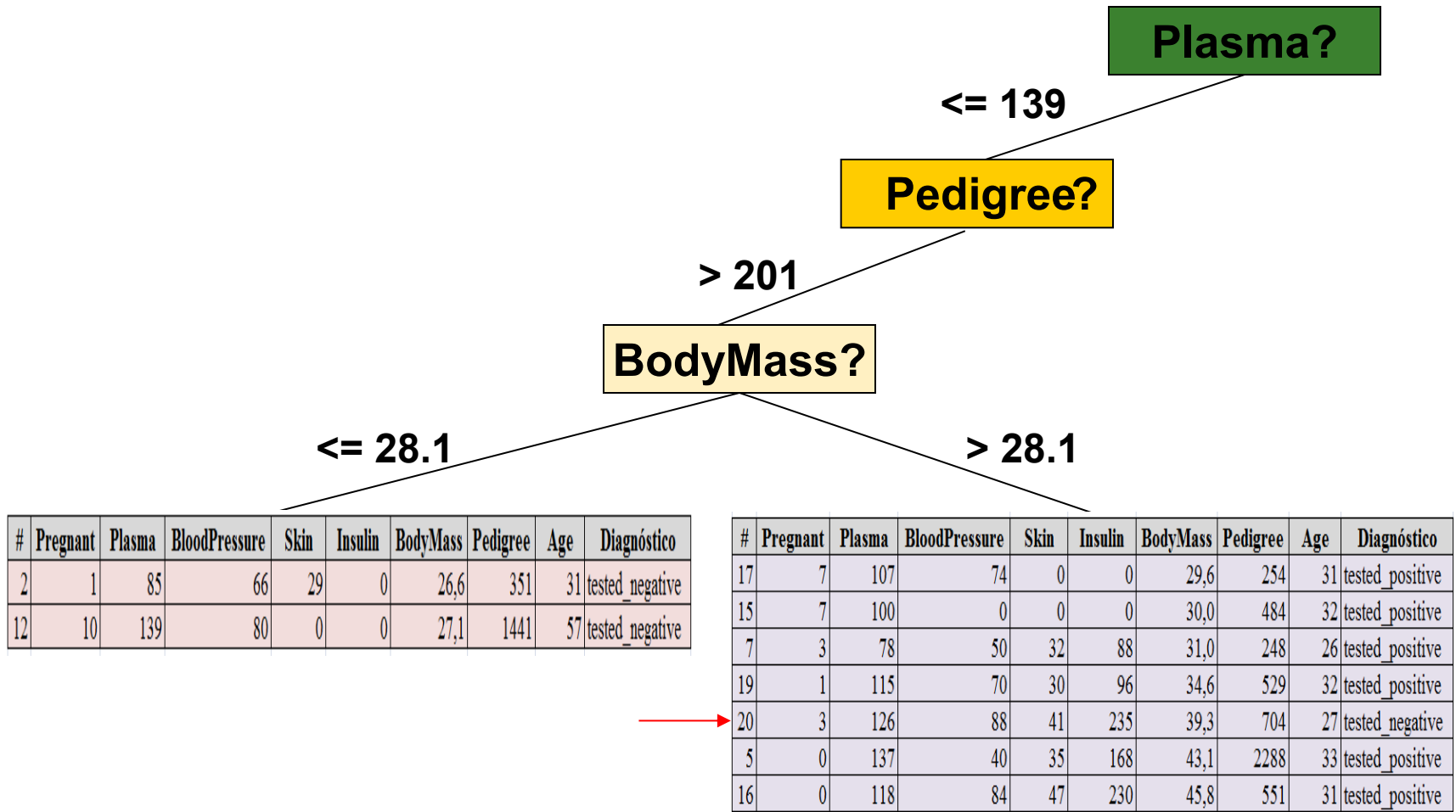
#	Pregnant	Plasma	BloodPressure	Skin	Insulin	BodyMass	Pedigree	Age	Diagnóstico
7	3	78	50	32	88	31,0	248	26	tested_positive
2	1	85	66	29	0	26,6	351	31	tested_negative
4	1	89	66	23	94	28,1	167	21	tested_negative
15	7	100	0	0	0	30,0	484	32	tested_positive
18	1	103	30	38	83	43,3	183	33	tested_negative
17	7	107	74	0	0	29,6	254	31	tested_positive
10	4	110	92	0	0	37,6	191	30	tested_negative
8	10	115	0	0	0	35,3	134	29	tested_negative
19	1	115	70	30	96	34,6	529	32	tested_positive
6	5	116	74	0	0	25,6	201	30	tested_negative
16	0	118	84	47	230	45,8	551	31	tested_positive
20	3	126	88	41	235	39,3	704	27	tested_negative
5	0	137	40	35	168	43,1	2288	33	tested_positive
12	10	139	80	0	0	27,1	1441	57	tested_negative

#	Pregnant	Plasma	BloodPressure	Skin	Insulin	BodyMass	Pedigree	Age	Diagnóstico
1	6	148	72	35	0	33,6	627	50	tested_positive
14	5	166	72	19	175	25,8	587	51	tested_positive
11	10	168	74	0	0	38,0	537	34	tested_positive
3	8	183	64	0	0	23,3	672	32	tested_positive
13	1	189	60	23	846	30,1	398	59	tested_positive
9	2	197	70	45	543	30,5	158	53	tested_positive

Árvore de Decisão



Árvore de Decisão



Árvore de Decisão

□ Geração de regras:

Teste	#	Pregnant	Plasma	BloodPressure	Skin	Insulin	BodyMass	Pedigree	Age	Diagnóstico
IF Plasma > 139	1	6	148	72	35	0	33,6	627	50	tested_positive
	14	5	166	72	19	175	25,8	587	51	tested_positive
	11	10	168	74	0	0	38,0	537	34	tested_positive
	3	8	183	64	0	0	23,3	672	32	tested_positive
	13	1	189	60	23	846	30,1	398	59	tested_positive
	9	2	197	70	45	543	30,5	158	53	tested_positive
IF Plasma <= 139	8	10	115	0	0	0	35,3	134	29	tested_negative
and	4	1	89	66	23	94	28,1	167	21	tested_negative
Pedigree <= 201	18	1	103	30	38	83	43,3	183	33	tested_negative
	10	4	110	92	0	0	37,6	191	30	tested_negative
	6	5	116	74	0	0	25,6	201	30	tested_negative
IF Plasma <= 139 and	2	1	85	66	29	0	26,6	351	31	tested_negative
Pedigree > 201 and	12	10	139	80	0	0	27,1	1441	57	tested_negative
BodyMass <= 28.1	-	-	-	-	-	-	-	-	-	-
IF Plasma <= 139	17	7	107	74	0	0	29,6	254	31	tested_positive
and	15	7	100	0	0	0	30,0	484	32	tested_positive
Pedigree > 201	7	3	78	50	32	88	31,0	248	26	tested_positive
and	19	1	115	70	30	96	34,6	529	32	tested_positive
BodyMass > 28.1	20	3	126	88	41	235	39,3	704	27	tested_negative
	5	0	137	40	35	168	43,1	2288	33	tested_positive
	16	0	118	84	47	230	45,8	551	31	tested_positive

Árvore de Decisão

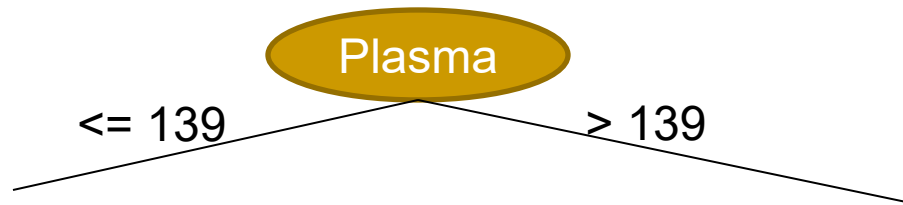
❑ Como classificar a seguinte instância:

7	103	66	32	0	39.1	344	31
---	-----	----	----	---	------	-----	----

Árvore de Decisão

❑ Como classificar a seguinte instância:

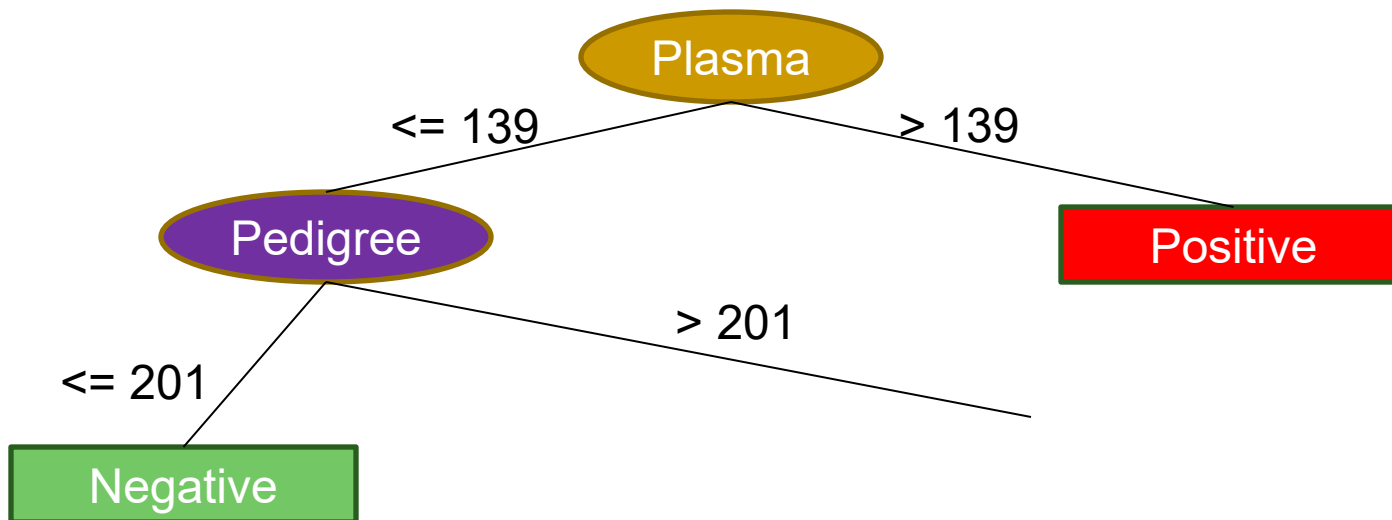
7	103	66	32	0	39.1	344	31
---	-----	----	----	---	------	-----	----



Árvore de Decisão

❑ Como classificar a seguinte instância:

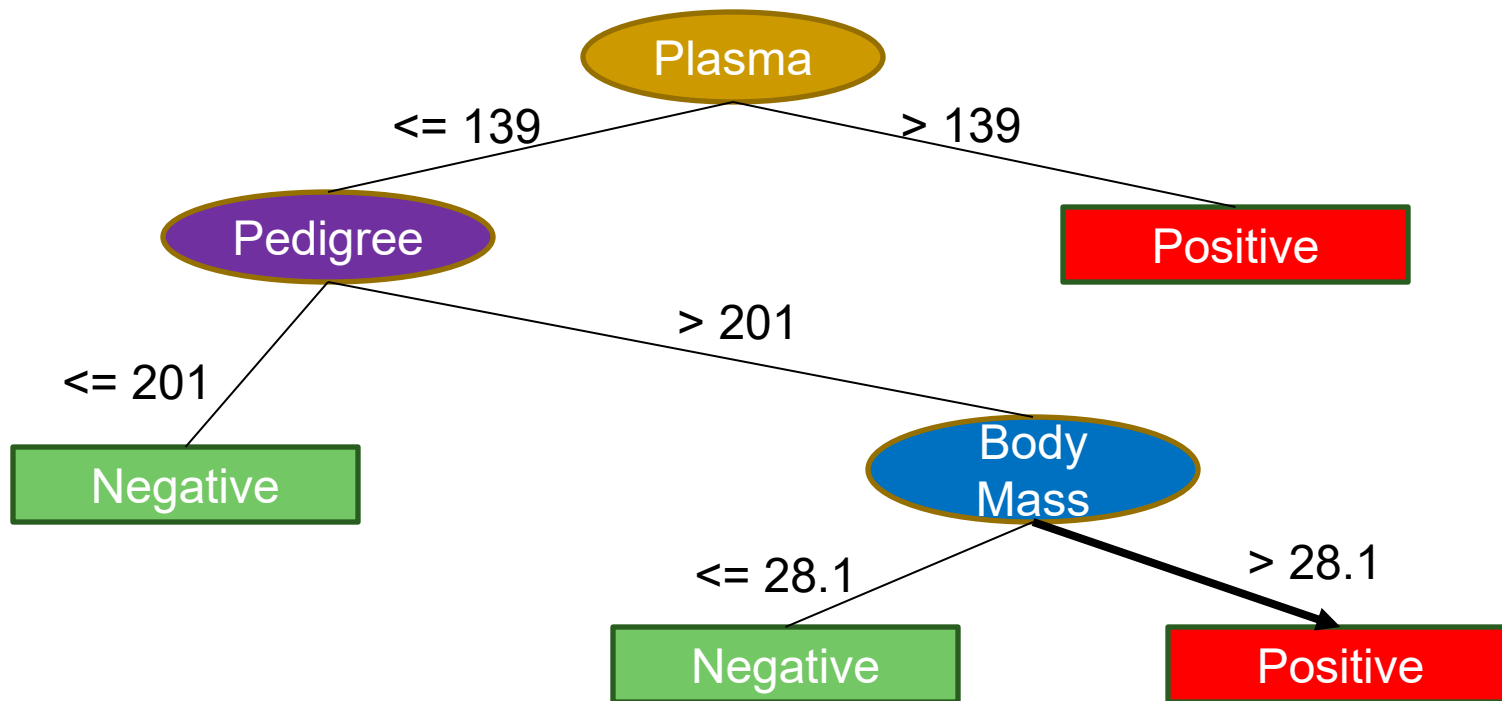
7	103	66	32	0	39.1	344	31
---	-----	----	----	---	------	-----	----



Árvore de Decisão

❑ Como classificar a seguinte instância:

7	103	66	32	0	39.1	344	31
---	-----	----	----	---	------	-----	----



Árvore de Decisão

❑ Como classificar a seguinte instância:

7	103	66	32	0	39.1	344	31
---	-----	----	----	---	------	-----	----



7	103	66	32	0	39.1	344	31	tested_positive
---	-----	----	----	---	------	-----	----	-----------------

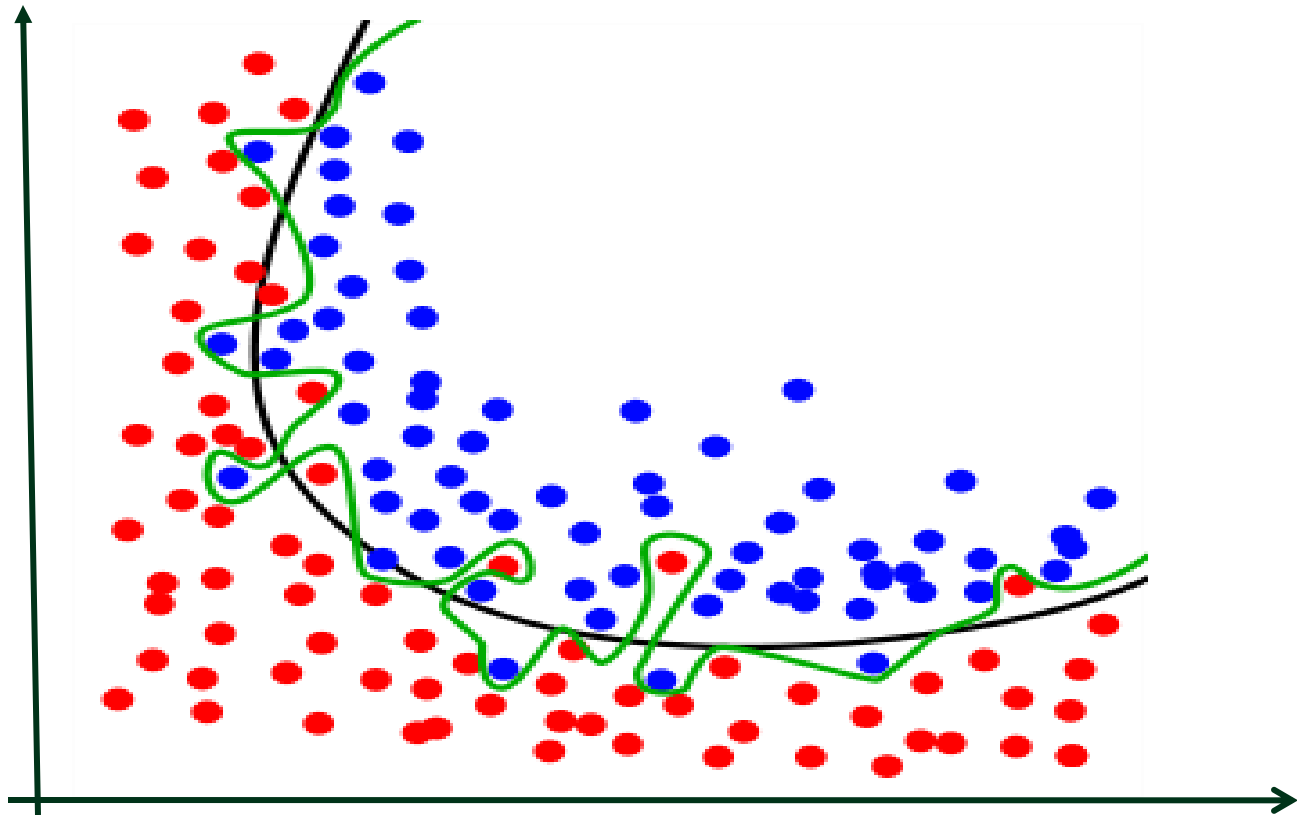
Árvore de Decisão

❑ Sobre-ajustamento ou Overfitting:

- ❖ O algoritmo de partição recursiva gera estruturas que podem obter um ajuste **perfeito** aos exemplos de treinamento.
- ❖ O aprendizado é muito específico ao conjunto de treinamento, **não permitindo generalizar** para o conjunto de teste.

Árvore de Decisão

❑ Overfitting:



Árvore de Decisão

❑ Problema de Overfitting:

- ❖ Para melhorar o modelo, utilizam-se métodos de poda (*pruning*) na árvore, cujo objetivo é melhorar a taxa de acerto do modelo para novas amostras que não foram utilizadas no treinamento.
- ❖ Existem diversas formas de realizar uma poda, e todas elas são classificadas como:
 - Pré-poda
 - Ou
 - Pós-poda

Árvore de Decisão

□ Poda:

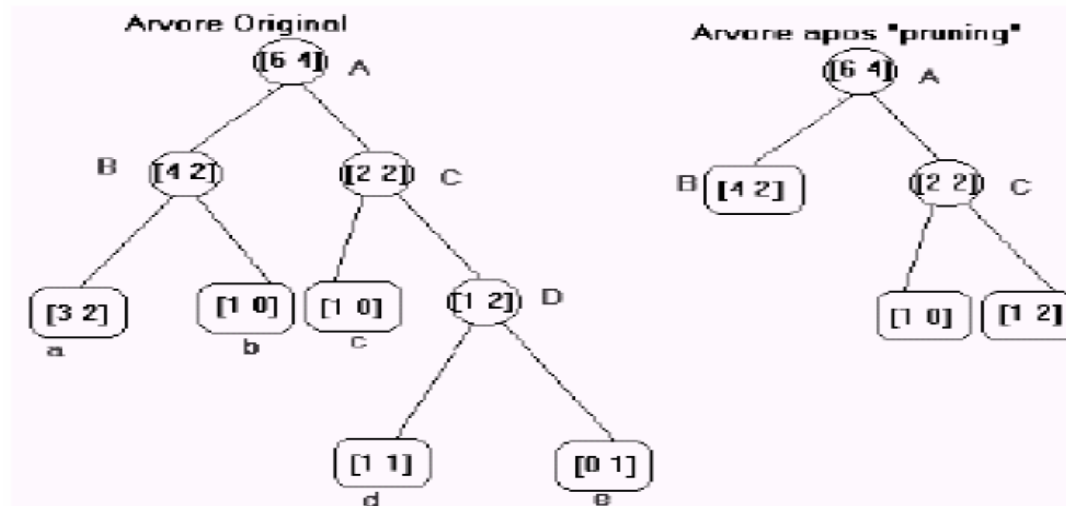
- ❖ **Pré-Poda:** durante a geração da árvore (baseada em critérios).
 - Ganho de informação for menor que um valor **pré-estabelecido**, então um determinado nó n vira folha;
 - Número mínimo de instâncias em um nó n a ser considerado para divisão;
 - Número mínimo de instâncias em um nó terminal (folha);
 - Profundidade máxima da árvore.

Árvore de Decisão

❑ Poda:

❖ **Pós-Poda:** realizada após a construção da árvore.

- Para cada nó interno da árvore, é calculada a taxa de erro, e caso esse nó vire folha (e tudo abaixo dele seja eliminado).



Árvore de Decisão

Tenho como saber se uma
AD está ou não
apresentando Overfitting?



Árvore de Decisão

❑ Overfitting:

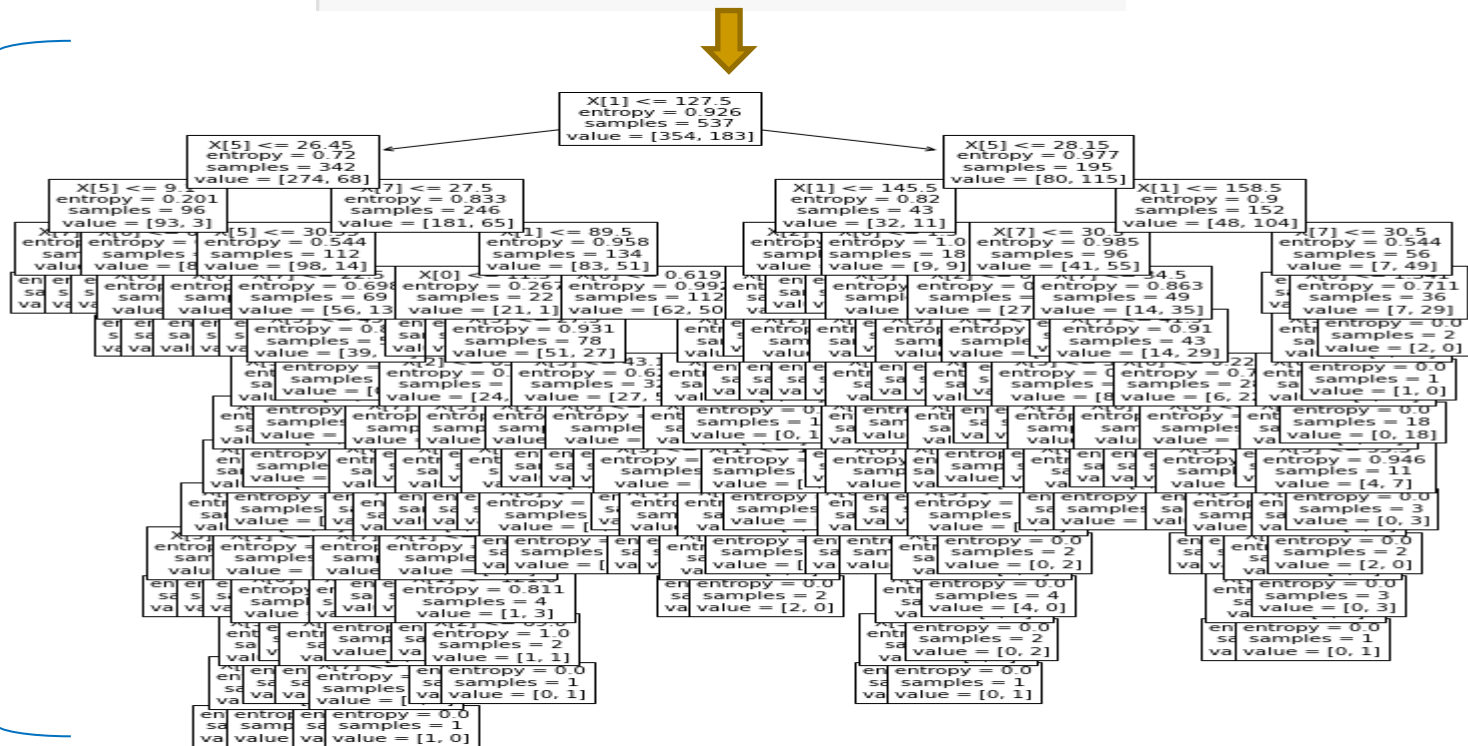
- Por padrão, o modelo de árvore de decisão pode crescer até sua profundidade total.
- A poda refere-se a uma técnica para remover as partes da árvore de decisão para evitar o crescimento em toda a sua profundidade.
- Ao ajustar os hiperparâmetros do modelo de árvore de decisão, pode-se podar as árvores e evitar que elas sejam superajustadas.

Árvore de Decisão

Overfitting:

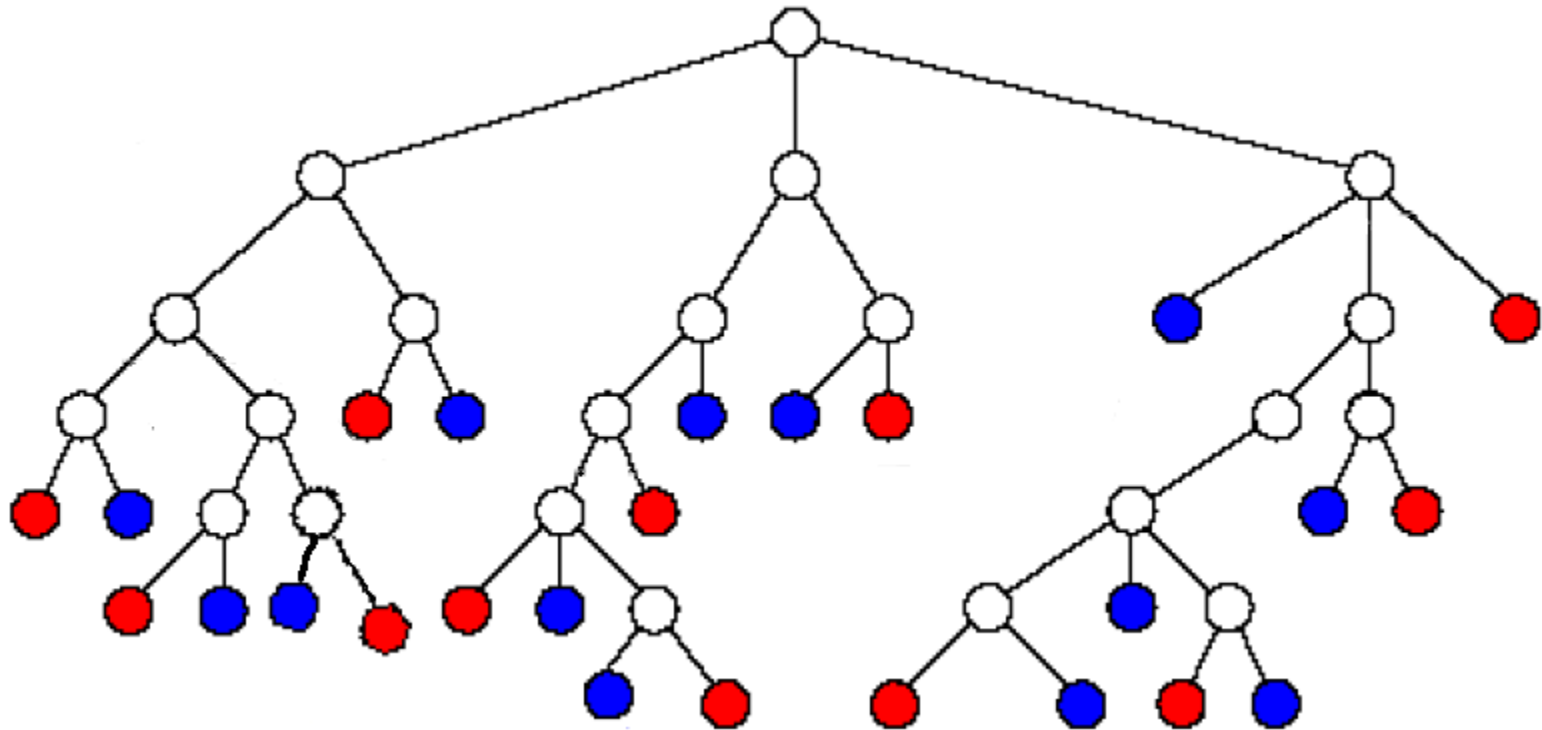
```
# Create Decision Tree classifier object  
dtt = DecisionTreeClassifier(criterion="entropy")
```

[768 rows x 9 columns]
Accuracy: 0.732



Árvore de Decisão

□ Pós-Poda:

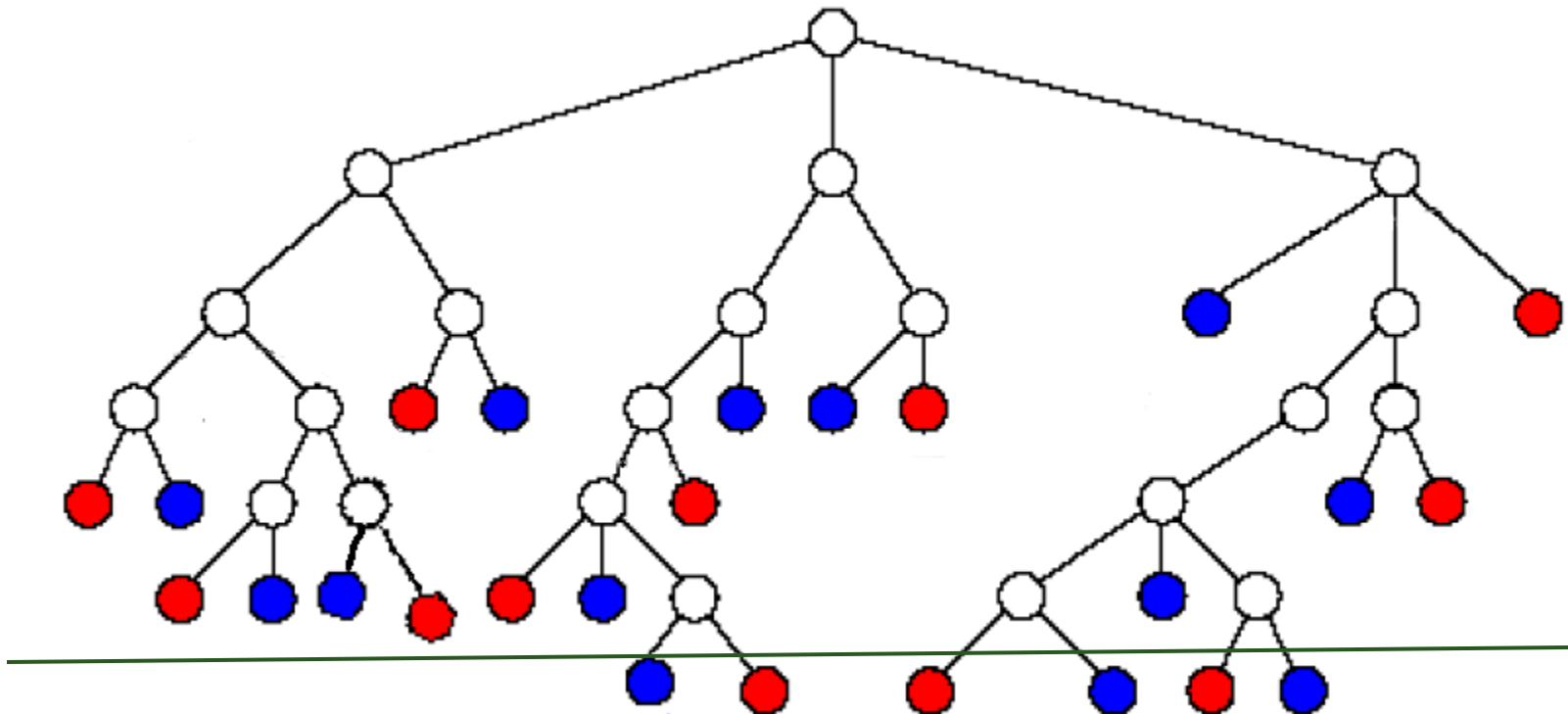


`max_depth = 6`

`Acc = 0,7662`

Árvore de Decisão

□ Pós-Poda:

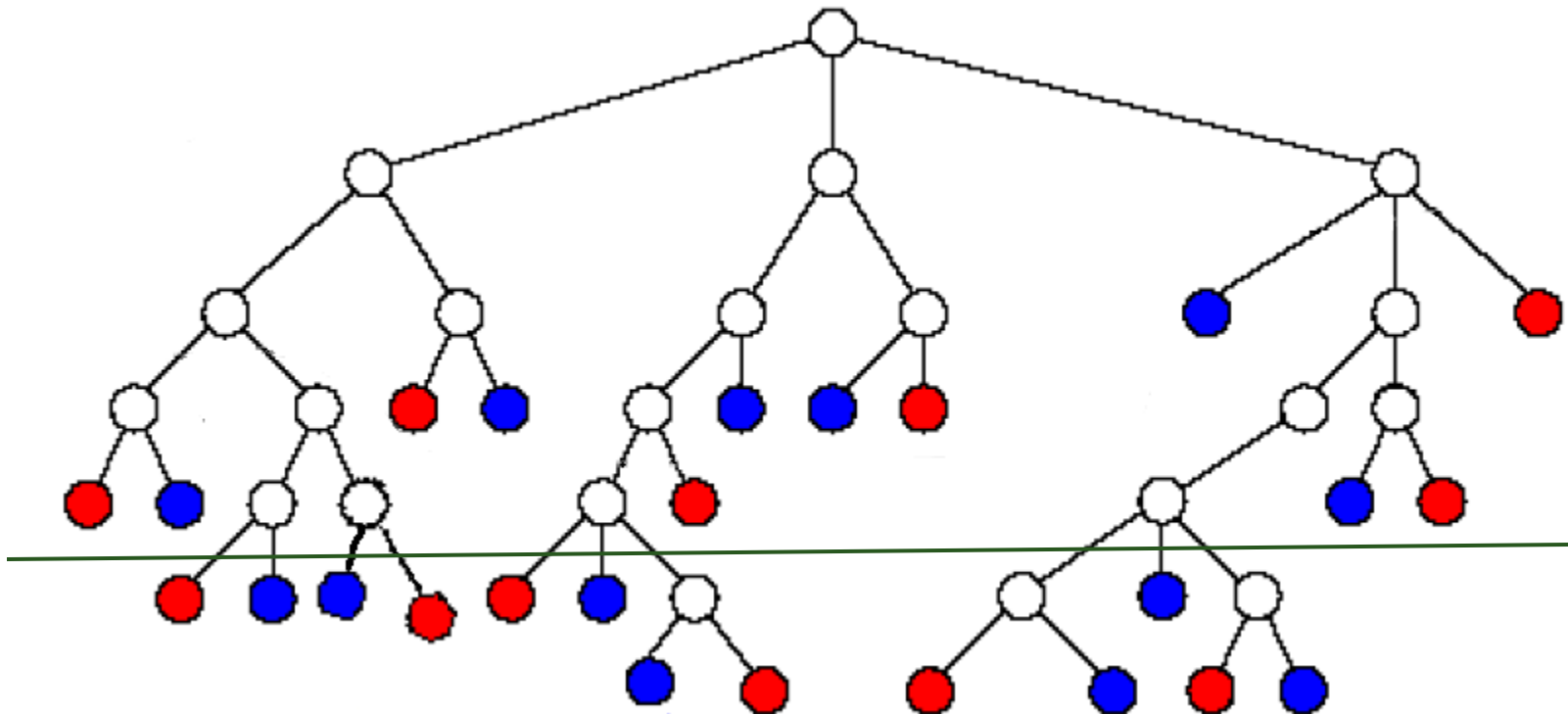


`max_depth = 5`

`Acc = 0,7682`

Árvore de Decisão

□ Poda:

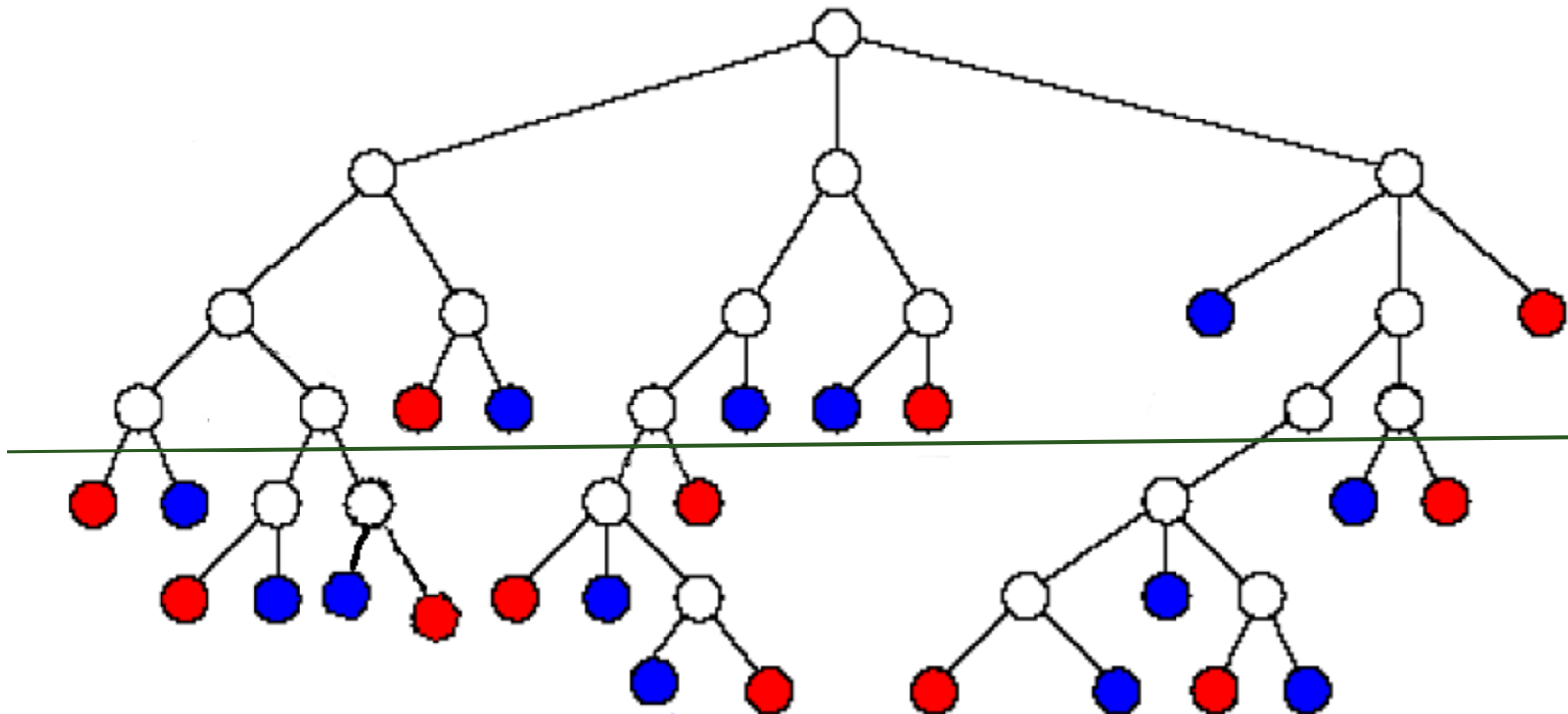


`max_depth = 4`

`Acc = 0,7712`

Árvore de Decisão

□ Pós-Poda:



`max_depth = 3`

`Acc = 0,7545`

Dúvidas ...



Algoritmos mais conhecidos

- ❑ **ID3 (Iterative Dichotomiser 3) (Quilan,1986):**
 - Os atributos devem ser obrigatoriamente categóricos.
- ❑ **C4.5 (J48 no Weka) (Quilan, 1993):**
 - Um algoritmo para geração de árvores de decisão, sucessor do algoritmo ID3.
 - Considera atributos **numéricos** e **categóricos**.
- ❑ **CART (Classification And Regression Trees) (Breiman et al., 1984):**
 - Produz árvores de classificação ou regressão, dependendo se as variáveis são categóricas ou numéricas.

Overview

□ Vantagens:

- Estrutura de fácil manipulação;
- Produzem modelos que podem ser facilmente interpretados por humanos;
- Muito rápido para classificar amostras desconhecidas.

□ Desvantagens:

- Pouca robustez a dados de **grande dimensão**;
- Acurácia afetada por **atributos pouco relevantes**.

Árvore de Decisão

Prática

Árvore de Decisão

- ❑ Parâmetros e valores default para Árvore de Decisão:

`sklearn.tree.DecisionTreeClassifier`

```
class sklearn.tree.DecisionTreeClassifier(*, criterion='gini', splitter='best', max_depth=None, min_samples_split=2,  
min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None,  
min_impurity_decrease=0.0, class_weight=None, ccp_alpha=0.0)
```

[\[source\]](#)

<https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html>

Árvore de Decisão

❑ Parâmetros e valores default para Árvore de Decisão:

Parameters:

criterion : {*"gini"*, *"entropy"*, *"log_loss"*}, *default="gini"*

The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "log_loss" and "entropy" both for the Shannon information gain, see [Mathematical formulation](#).

splitter : {*"best"*, *"random"*}, *default="best"*

The strategy used to choose the split at each node. Supported strategies are "best" to choose the best split and "random" to choose the best random split.

max_depth : *int*, *default=None*

The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

min_samples_split : *int or float*, *default=2*

The minimum number of samples required to split an internal node:

- If int, then consider `min_samples_split` as the minimum number.
- If float, then `min_samples_split` is a fraction and `ceil(min_samples_split * n_samples)` are the minimum number of samples for each split.

<https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html>

Árvore de Decisão

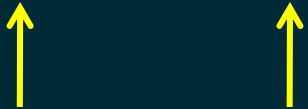
❑ Configurando a AD:

```
# Create Decision Tree classifier object
clf = DecisionTreeClassifier(criterion="entropy", max_depth=3)

# Train Decision Tree Classifier
clf = clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)

# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```



```
Accuracy: 0.7705627705627706
```

Árvore de Decisão

❑ Utilizando DecisionTreeClassifier (Holdout):

```
[1] ### Carregar as Libraries
import numpy as np
import pandas as pd

### Importing Dataset
dataset = pd.read_csv('Diabetes.csv',encoding='utf-8')

[2] # Obtendo os nomes das colunas Numéricas
tipos_numericos = ['int32', 'int64', 'float16', 'float32', 'float64']
cols_num = dataset.select_dtypes(include=tipos_numericos)

## Selecionando os atributos numéricos
colunas_numericas = list(cols_num.columns)

## Pegar a classe
coluna_classe = dataset['classe']

## Separando os atributos da classe
X = dataset[colunas_numericas] # Features
y = dataset.classe            # Target variable (classe)
```

Árvore de Decisão

❑ Utilizando **DecisionTreeClassifier**:

```
[3] ## Carregando o algoritmo / método / técnica Decision Tree
    from sklearn.tree import DecisionTreeClassifier
    from sklearn.model_selection import train_test_split
    from sklearn import metrics
    from sklearn.metrics import confusion_matrix
```

```
[4] # Separando dataset em duas partes: treinamento e teste
    # 70% training and 30% test
    X_train_70, X_test_30, y_train_70, y_test_30 = train_test_split(X, y, test_size=0.3, random_state=1)
```

Árvore de Decisão

❑ Utilizando **DecisionTreeClassifier**:

```
[6] # Create Decision Tree classifier object
    dtc = DecisionTreeClassifier(criterion="entropy", max_depth=3)

    # Train Decision Tree Classifier
    dtc = dtc.fit(X_train_70,y_train_70)

    #Predict the response for test dataset
    y_pred = dtc.predict(X_test_30)

    # Model Accuracy
    acuracia = metrics.accuracy_score(y_test_30, y_pred)
    print('Accuracy: %.3f' % acuracia)

    # Matriz de confusão p/ 30%
    confusion_matrix(y_test_30, y_pred)
```

```
Accuracy: 0.771
array([[124,  22],
       [ 31,  54]])
```

Árvore de Decisão

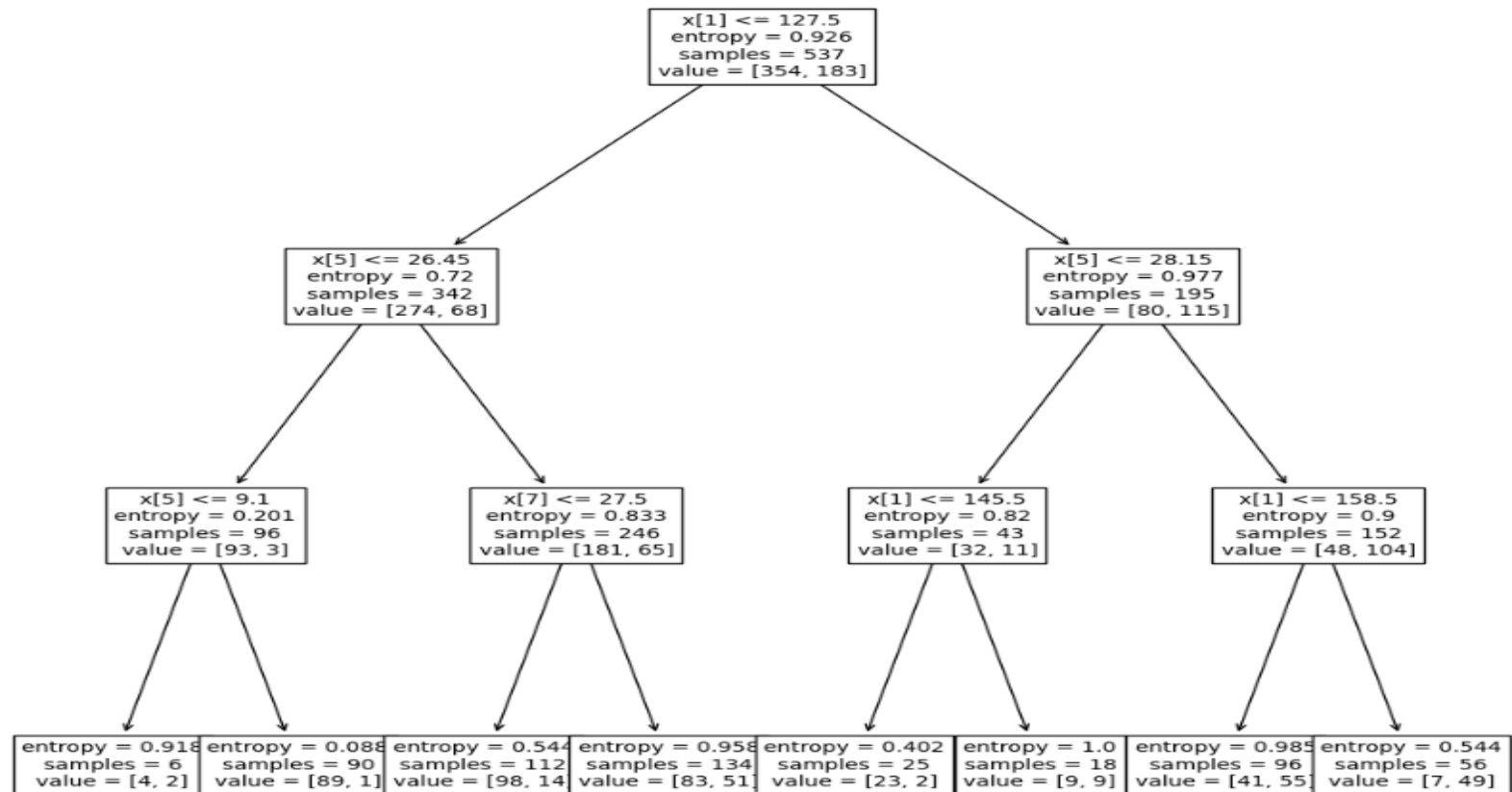
❑ Utilizando **DecisionTreeClassifier**:

```
[7] # Mostrando (plotando) a árvore gerada
    from sklearn import tree
    import matplotlib.pyplot as plt
    plt.figure(figsize=(12,12)) # set plot size (denoted in inches)
    tree.plot_tree(dtc, fontsize=10)
    plt.show()

    # Matriz de confusão p/ 30%
    confusion_matrix(y_test_30, y_pred)
```


Árvore de Decisão

□ Visualizando a **árvore** gerada:



Árvore de Decisão

❑ Utilizando DecisionTreeClassifier (k fold cv):

```
[3] ## Carregando o algoritmo / método / técnica Decision Tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_val_predict
from sklearn.metrics import confusion_matrix

[4] # 10-fold CV
kf = KFold(n_splits=10, random_state=1, shuffle=True)

# Create Decision Tree classifier object
dtc = DecisionTreeClassifier(criterion="entropy", max_depth=3)

# Model Accuracy
scores = cross_val_score(dtc, X, y, scoring='accuracy', cv=kf)
print('Accuracy: %.3f (%.3f)' % (scores.mean(), scores.std()))

# Matriz de confusão p/ k fold
y_pred = cross_val_predict(dtc, X, y, cv=kf)
confusion_matrix(y, y_pred)
```

```
Accuracy: 0.752 (0.042)
array([[424,  76],
       [114, 154]])
```

Árvore de Decisão

❑ Utilizando DecisionTreeClassifier (k fold cv):

```
[9] cnt = 1
# split() method generate indices to split data into training and test set.
for train_index, test_index in kf.split(X, y):
    print(f'Fold:{cnt}, Train set: {len(train_index)}, Test set:{len(test_index)}')
    cnt += 1

print()

# Model Accuracy
scores = cross_val_score(dtc, X, y, scoring='accuracy', cv=kf)
print('Accuracy: %.3f (%.3f)' % (mean(scores), std(scores)))
```

```
Fold:1, Train set: 691, Test set:77
Fold:2, Train set: 691, Test set:77
Fold:3, Train set: 691, Test set:77
Fold:4, Train set: 691, Test set:77
Fold:5, Train set: 691, Test set:77
Fold:6, Train set: 691, Test set:77
Fold:7, Train set: 691, Test set:77
Fold:8, Train set: 691, Test set:77
Fold:9, Train set: 692, Test set:76
Fold:10, Train set: 692, Test set:76
```

```
Accuracy: 0.752 (0.042)
```

Árvore de Decisão

Tenho como saber se uma
AD está ou não
apresentando Overfitting?

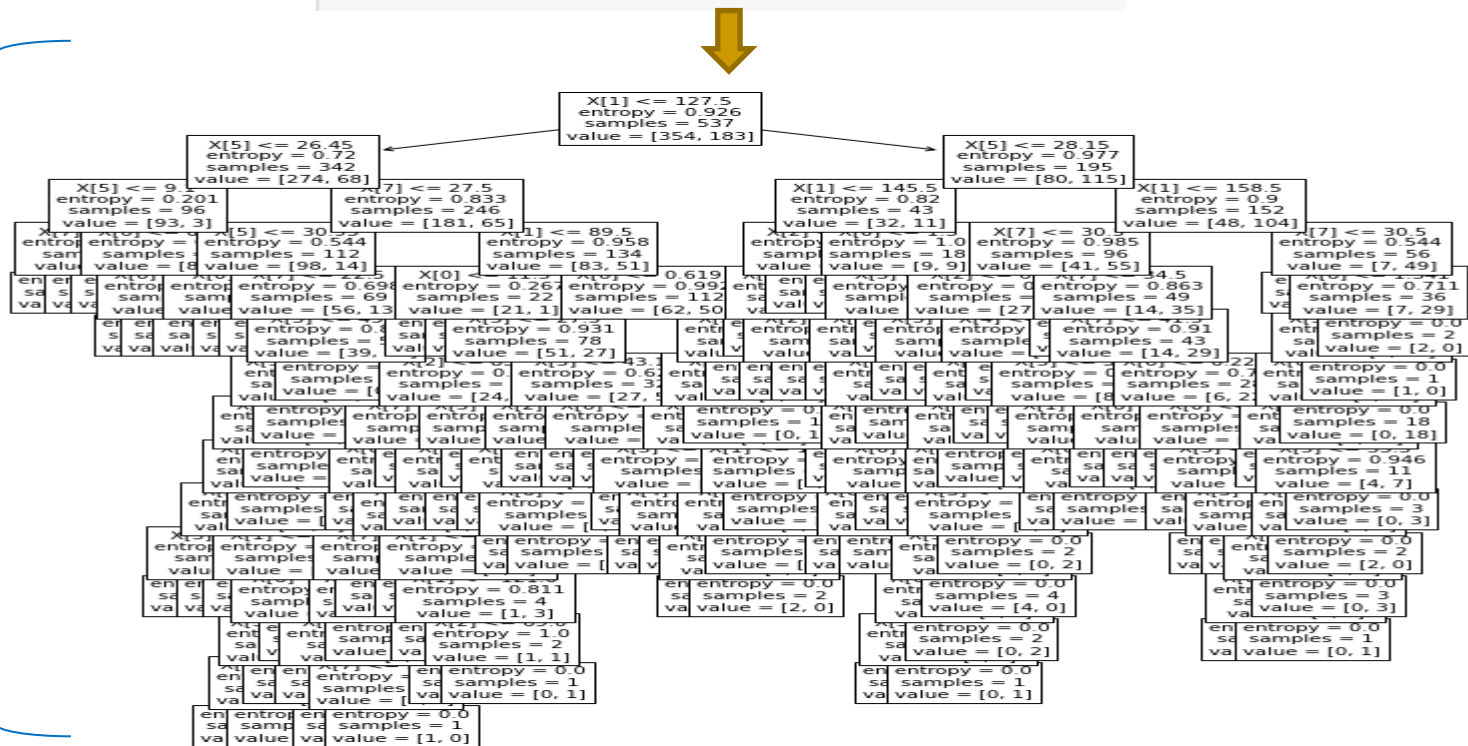


Árvore de Decisão

Overfitting:

```
# Create Decision Tree classifier object  
dtt = DecisionTreeClassifier(criterion="entropy")
```

[768 rows x 9 columns]
Accuracy: 0.732



Árvore de Decisão

❑ Pós-poda:

```
# -----  
# usar Decision Tree Classifier no sklearn  
# Load libraries  
from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classifier  
from sklearn.model_selection import train_test_split # Import train_test_split function  
from sklearn import metrics #Import scikit-learn metrics module for accuracy calculation  
from sklearn.metrics import confusion_matrix  
  
#split dataset in features and target variable  
feature_cols = ['preg', 'plas', 'pres', 'skin', 'insu', 'mass', 'pedi', 'age']  
X = dados[feature_cols] # Features  
y = dados.classe # Target variable  
  
# Split dataset into training set and test set  
X_train_70, X_test_30, y_train_70, y_test_30 = train_test_split(X, y, test_size=0.3, random_state=1) # 70% training and 30% test  
  
vals = [2,3,4,5,6,7,8,9,10]  
  
for i in vals:  
    # Create Decision Tree classifier object  
    dtc = DecisionTreeClassifier(criterion="entropy", max_depth=i)  
    dtc = dtc.fit(X_train_70,y_train_70)  
    y_pred = dtc.predict(X_test_30)  
    print(f'Acuracia:{metrics.accuracy_score(y_test_30, y_pred)},Max_depth:{i}')
```

DT_pos-poda.py

Árvore de Decisão

❑ Pós-poda:

```
Acuracia:0.7705627705627706,Max_depth:2  
Acuracia:0.7705627705627706,Max_depth:3  
Acuracia:0.7748917748917749,Max_depth:4  
Acuracia:0.7705627705627706,Max_depth:5  
Acuracia:0.7662337662337663,Max_depth:6  
Acuracia:0.7878787878787878,Max_depth:7  
Acuracia:0.7575757575757576,Max_depth:8  
Acuracia:0.7229437229437229,Max_depth:9  
Acuracia:0.7142857142857143,Max_depth:10
```

DT_pos-poda.py

Obrigado!!!

