



Inteligência Artificial e Big Data

Aula 08

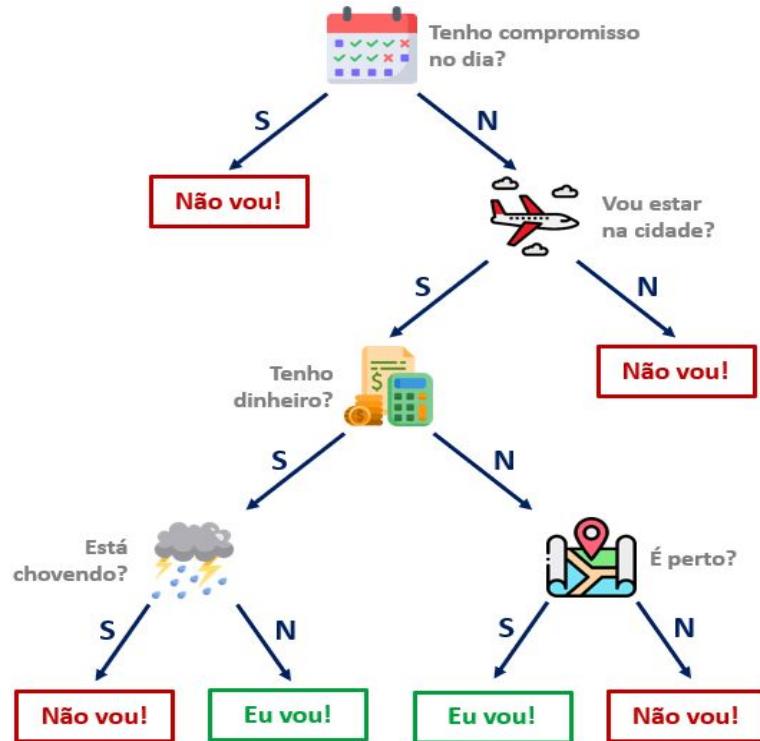
Prof. Me Daniel Vieira

Agenda

- 1- Árvore de decisão - Classificação
- 2- Árvore de decisão
- 3- Estudo de caso
- 4- Matriz de confusão
- 5- Acurácia, precisão, recall, F1 - Score
- 6- Situações de aprendizagem

Árvore de decisão

É um algoritmo de aprendizado supervisionado que cria uma estrutura na forma de árvore para tomar decisões. Cada nó interno representa um teste em uma característica, cada ramo representa um resultado possível desse teste e cada folha representa uma classe ou valor de saída



Árvore de decisão código

```
import matplotlib.pyplot as plt  
from sklearn import datasets  
from sklearn import tree  
  
iris = datasets.load_iris()  
  
data = iris.data  
target = iris.target  
data  
target
```

```
data = iris.data  
data  
# Imprima as primeiras linhas dos dados  
print("Dados:")  
print(data[:5]) # Imprime as cinco  
primeiras linhas dos dados  
  
# Imprima as primeiras linhas do alvo  
print("\nAlvo:")  
print(target[:5]) # Imprime as cinco  
primeiras linhas do alvo
```

Árvore de decisão código

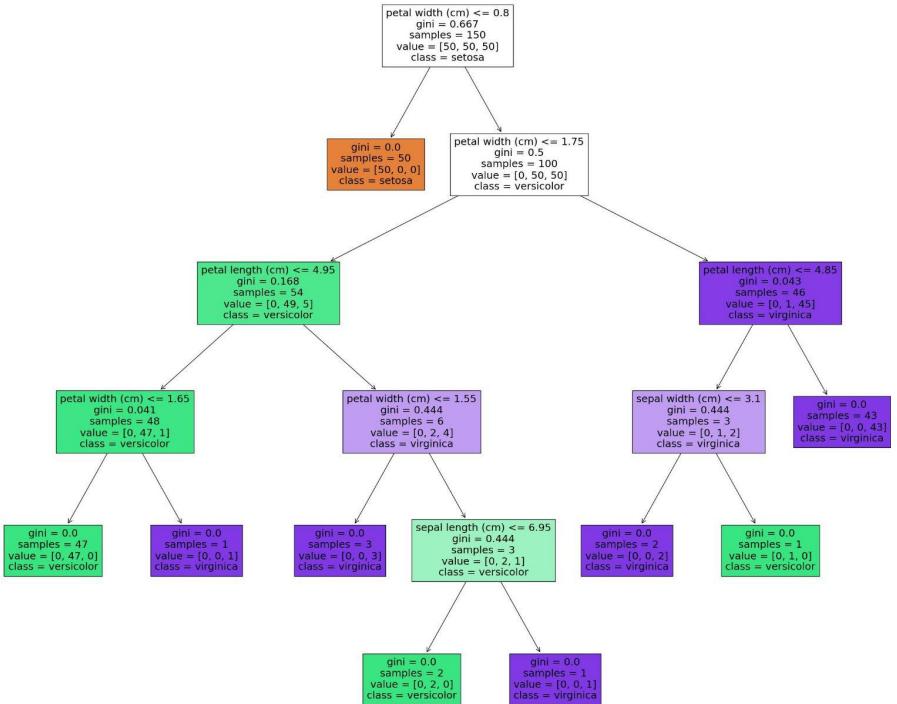
```
X,y = iris.data,iris.target
```

```
fig = plt.figure(figsize=(30,25))
tree.plot_tree(classificador,feature_names=iris.feature_names,
class_names=iris.target_names,filled=True)
```

Árvore de decisão código

```
X,y = iris.data,iris.target  
classificador = tree.DecisionTreeClassifier()  
classificador.fit(X,y)  
fig = plt.figure(figsize=(30,25))  
tree.plot_tree(classificador,feature_names=iris.feature_names  
,class_names=  
iris.target_names,filled=True)  
plt.savefig('arvore.jpg')
```

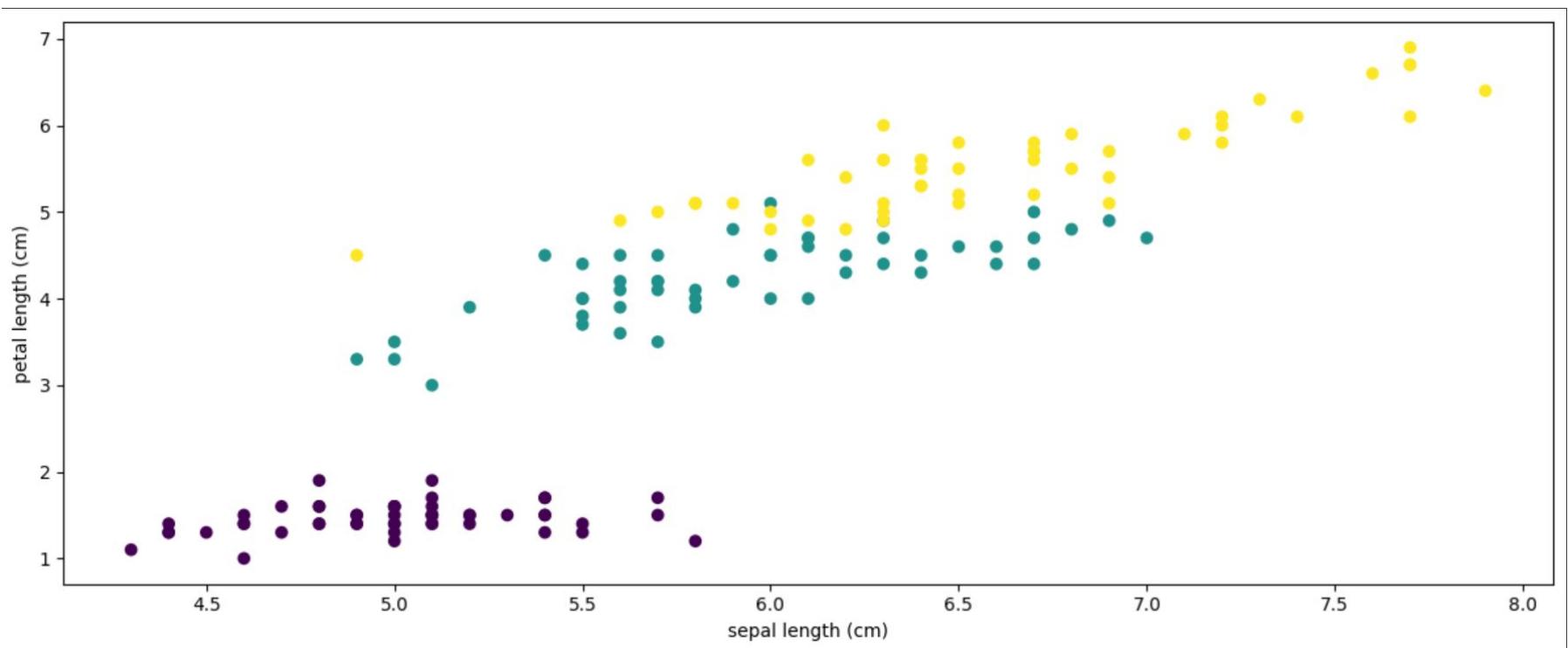
Árvore de decisão código



Árvore de decisão código

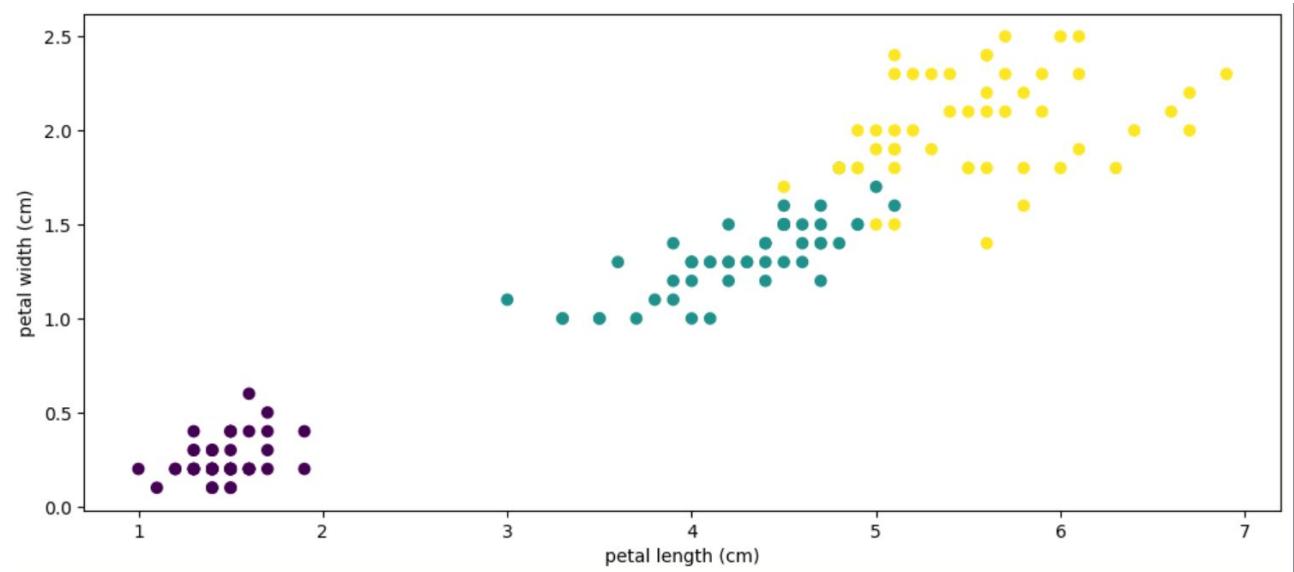
```
plt.figure(figsize=(15,10))
plt.scatter(iris.data[:,0],iris.data[:,2],c = iris.target)
plt.xlabel(iris.feature_names[0])
plt.ylabel(iris.feature_names[2])
plt.tight_layout()
plt.show()
```

Árvore de decisão código



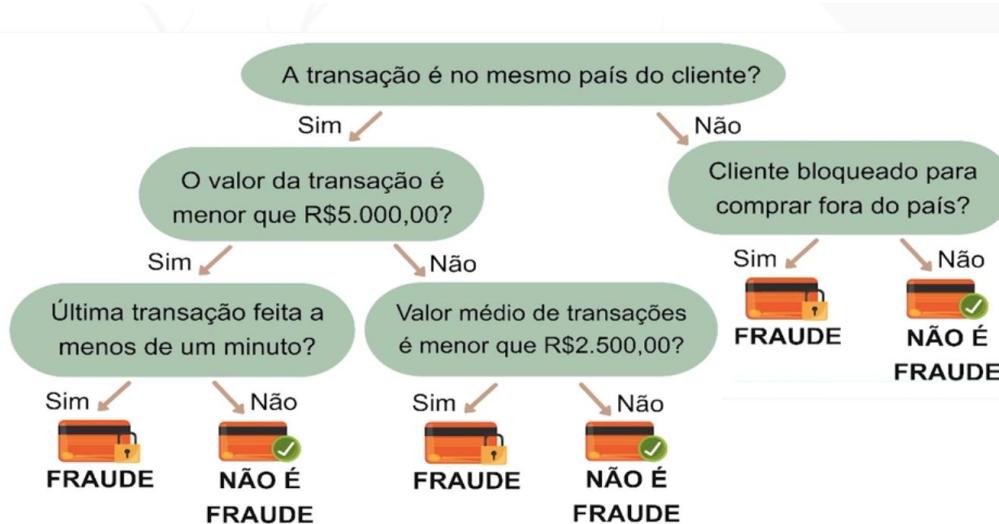
Árvore de decisão código

```
plt.figure(figsize=(15,10))
plt.scatter(iris.data[:,2],iris.data[:,3],c=iris.target)
plt.xlabel(iris.feature_names[2])
plt.ylabel(iris.feature_names[3])
```



Estudo de caso

Classificar uma transação com o cartão de crédito como fraude ou não



Estudo de caso

```
import pandas as pd  
df = pd.read_csv("creditcard.csv",sep=",")
```

✓ 0.7s

Python

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.128539	-0.189115	0.133558	-0.021053	149.62	0
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.167170	0.125895	-0.008983	0.014724	2.69	0
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.139097	-0.055353	-0.09752	378.66	0
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575	0.647376	-0.221929	0.062723	0.061458	123.50	0
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267	-0.206010	0.502292	0.219422	0.215153	69.99	0

5 rows × 31 columns

Estudo de caso

```
n_transacoes = df['Class'].count()  
  
print("Número de transações: ", n_transacoes)  
n_transacoes = df['Class'].count()  
n_fraudes = df['Class'].sum()  
  
print("Número de transações: ", n_transacoes)  
print("Número de fraudes: ", n_fraudes)
```

```
n_transacoes = df['Class'].count()  
n_fraudes = df['Class'].sum()  
n_normais = n_transacoes - n_fraudes  
  
print("Número de transações: ", n_transacoes)  
print("Número de fraudes: ", n_fraudes)  
print("Número de transações normais: ", n_normais)
```

Estudo de caso

```
n_transacoes = df['Class'].count()
n_fraudes = df['Class'].sum()
n_normais = n_transacoes - n_fraudes
fraudes_porc = n_fraudes / n_transacoes
normais_porc = n_normais / n_transacoes

print("Número de transações: ", n_transacoes)
print("Número de fraudes: ", n_fraudes, "%."2f"
%(fraudes_porc*100))
print("Número de transações normais: ", n_normais,
"%(."2f" %(normais_porc*100))
X = df.drop('Class', axis=1).values
y = df['Class'].values
```

Estudo de caso

```
from sklearn.model_selection import  
StratifiedShuffleSplit  
    validador = StratifiedShuffleSplit(n_splits=1,  
test_size=0.1, random_state=0)  
for treino_id, teste_id in validador.split(X, y):  
    x_train, x_test = X[treino_id], X[teste_id]  
    y_train, y_test = y[treino_id], y[teste_id]
```

Estudo de caso

```
from sklearn import tree
classificador_arvore_decisao = tree.DecisionTreeClassifier()
arvore = classificador_arvore_decisao.fit(x_train, y_train)
classificador_arvore_decisao = tree.DecisionTreeClassifier()
arvore = classificador_arvore_decisao.fit(x_train, y_train)
y_pred = arvore.predict(x_test)

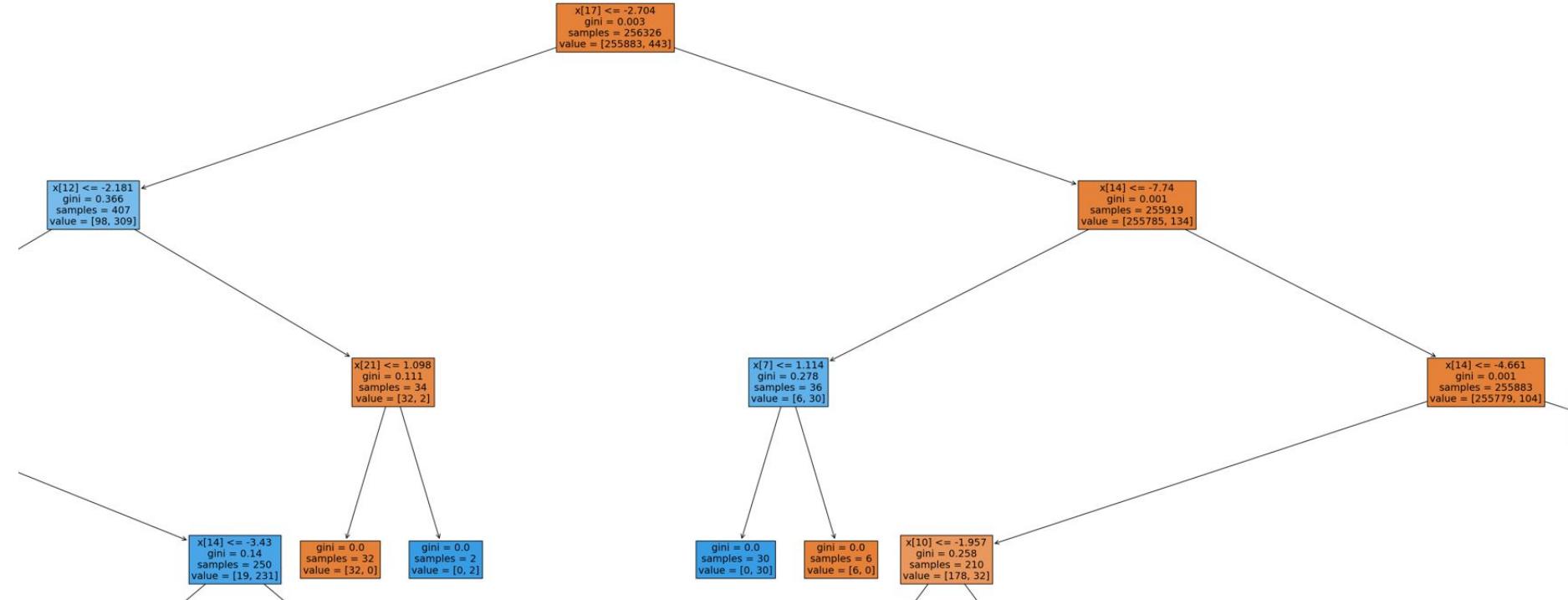
def validador(X, y):
    validador = StratifiedShuffleSplit(n_splits=1, test_size=0.1, random_state=0)
    for treino_id, teste_id in validador.split(X, y):
        X_train, X_test = X[treino_id], X[teste_id]
        y_train, y_test = y[treino_id], y[teste_id]
    return X_train, X_test, y_train, y_test
```

Estudo de caso

```
def executar_classificador(classificador, X_train, X_test, y_train):
    arvore = classificador.fit(X_train, y_train)
    y_pred = arvore.predict(X_test)
    return y_pred

classificador_arvore_decisao = tree.DecisionTreeClassifier()
y_pred_arvore_decisao = executar_classificador(classificador_arvore_decisao, x_train, x_test, y_train)
import matplotlib.pyplot as plt
def salvar_arvore(classificador,nome):
    plt.figure(figsize=(200,100))
    tree.plot_tree(classificador,filled = True, fontsize=14)
    plt.savefig(nome)
    plt.close()
#criacao da figura da arvore de decisao
salvar_arvore(classificador_arvore_decisao, "arvore_decisao1.png")
```

Árvore de decisão criada



Métricas para avaliar a árvore de decisão

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix

def validar_arvore(y_test, y_pred):
    print(accuracy_score(y_pred, y_test))
    print(confusion_matrix(y_pred, y_test))
validar_arvore(y_test,y_pred_arvore_decisao)
```

Métricas para avaliar a árvore de decisão

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
def validar_arvore(y_test, y_pred):
    print(accuracy_score(y_test, y_pred))
    print(precision_score(y_test, y_pred))
    print(recall_score(y_test, y_pred))
    print(confusion_matrix(y_test, y_pred))
validar_arvore(y_test, y_pred)
```

Métricas para avaliar a árvore de decisão

```
#validacao arvore de decisao
validar_arvore(y_test, y_pred_arvore_decisao)
#execucao do classificador DecisionTreeClassifier
classificador_arvore_decisao = tree.DecisionTreeClassifier(max_depth=10, random_state=0)
y_pred_arvore_decisao = executar_classificador(classificador_arvore_decisao, X_train,
X_test, y_train)
validar_arvore(y_test, y_pred_arvore_decisao)

#execucao do classificador DecisionTreeClassifier
classificador_arvore_decisao = tree.DecisionTreeClassifier(max_depth=10,
random_state=0,min_samples_leaf=10)
y_pred_arvore_decisao = executar_classificador(classificador_arvore_decisao, X_train,
X_test, y_train)
```

Matriz de confusão

		Valor previsto	
		Positivo	Negativo
Valor Real	Positivo	Verdadeiros Positivos	Falsos Negativos
	Negativo	Falsos Positivos	Verdadeiros Negativos

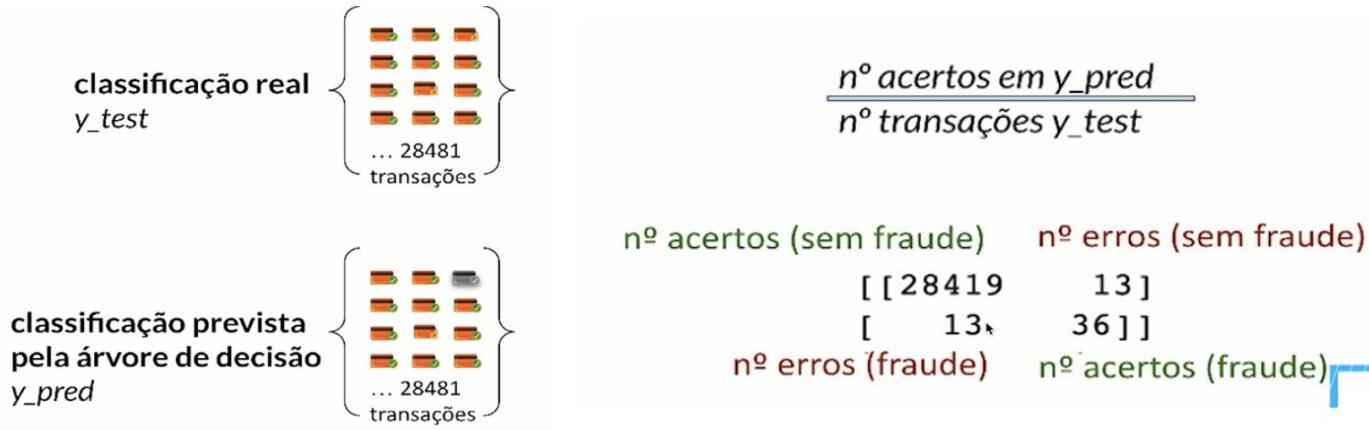
Matriz de confusão

		Predito	
		Falha	Não falha
Real	Falha	15	10
	Não falha	25	50

Métricas para avaliar a árvore de decisão (Acurácia)

$$Acurácia = \frac{Acertos\,(A)}{Acertos\,(A) + Erros\,(E)}$$

Métricas para avaliar a árvore de decisão (Acurácia)



Soma dados da diagonal

Métricas para avaliar a árvore de decisão (Acurácia)

$$Acurácia = \frac{Verdadeiros Positivos (VP) + Verdadeiros Negativos (VN)}{Total}$$

$$\begin{aligned} Total = & \text{ } Verdadeiros Positivos (VP) + Verdadeiros Negativos (VN) \\ & + Falsos Positivos (FP) + Falsos Negativos (FN) \end{aligned}$$

Métricas para avaliar a árvore de decisão (Acurácia)

$$Acurácia = \frac{Verdadeiros Positivos (VP) + Verdadeiros Negativos (VN)}{Total}$$

$$\begin{aligned} Total = & \text{ } Verdadeiros Positivos (VP) + Verdadeiros Negativos (VN) \\ & + Falsos Positivos (FP) + Falsos Negativos (FN) \end{aligned}$$

Métricas para avaliar a árvore de decisão (Acurácia)

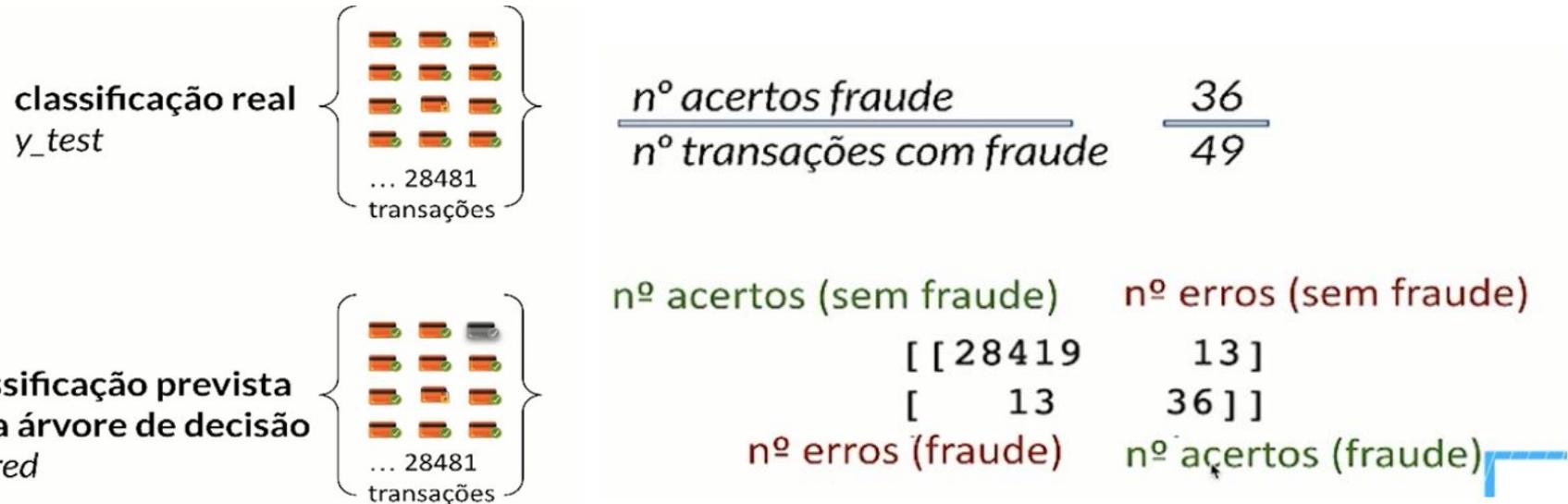
		Preditos	
		Falha	Não falha
Real	Falha	15	10
	Não falha	25	50

		Valor previsto	
		Positivo	Negativo
Valor Real	Positivo	Verdadeiros Positivos	Falsos Negativos
	Negativo	Falsos Positivos	Verdadeiros Negativos

$$\text{Acurácia} = \frac{\text{Verdadeiros Positivos (VP)} + \text{Verdadeiros Negativos (VN)}}{\text{Total}}$$

$$\text{Acurácia} = \frac{15 + 50}{15 + 50 + 25 + 10} = 65\%$$

Métricas para avaliar a árvore de decisão (Precisão)



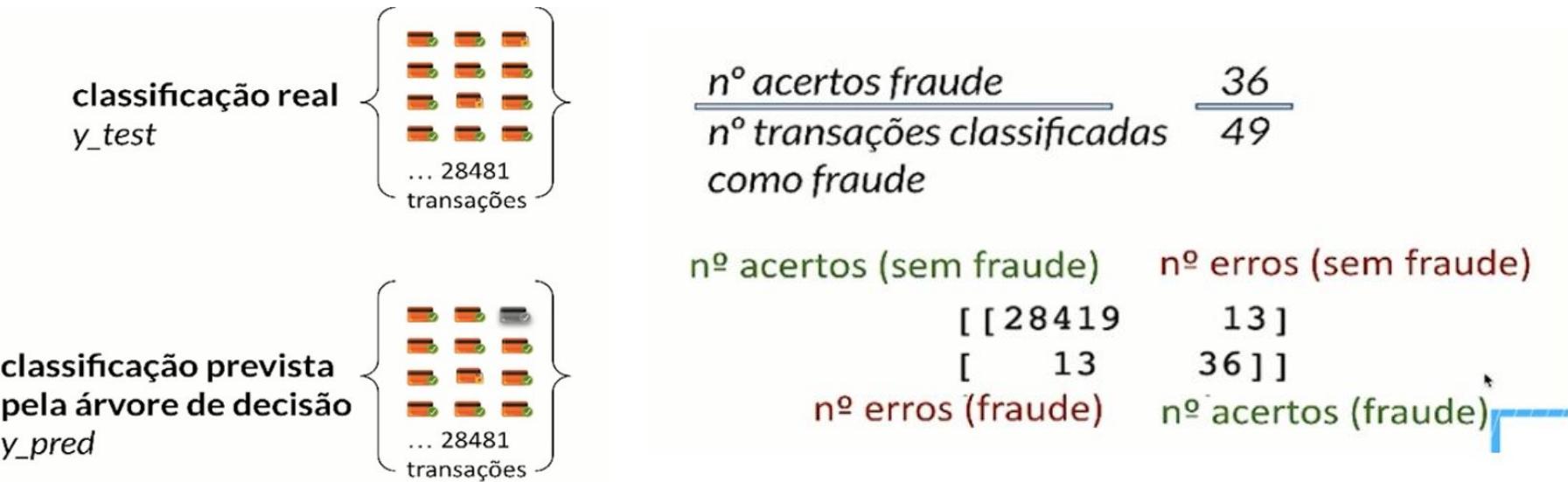
Soma dados da segunda linha da matriz de confusão

Métricas para avaliar a árvore de decisão (Precisão)

$$Precisão = \frac{Verdadeiros Positivos (VP)}{Verdadeiros Positivos (VP) + Falsos Positivos (FP)}$$

$$Precisão = \frac{15}{15 + 25} = 37,5\%$$

Métricas para avaliar a árvore de decisão (Recall)



Métricas para avaliar a árvore de decisão (Recall)

$$Revocação = \frac{Verdadeiros Positivos (VP)}{Verdadeiros Positivos (VP) + Falsos Negativos (FN)}$$

$$Revocação = \frac{15}{15 + 10} = 60\%$$

Métricas para avaliar a árvore de decisão (F1-Score)

$$F1 - Score = \frac{2 * Precisão * Revocação}{Precisão + Revocação}$$

$$F1 - Score = \frac{2 * 0,375 * 0,6}{0,375 + 0,6} = 46,15\%$$

Exercícios

1) Criar um script para classificar uma bebida como boa, ruim, péssima a partir das notas dos clientes

```
notas = 8, 6, 5, 9, 4, 3, 7, 2, 1, 5, 6, 9, 8] # notas dos clientes (0-10)
classes = 'boa', 'ruim', 'ruim', 'boa', 'péssima', 'péssima', 'boa', 'péssima',
'péssima', 'ruim', 'ruim', 'boa', 'boa']
```

Exercício 2

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn import tree
import matplotlib.pyplot as plt
import seaborn as sns

#Dados treinamento
notas = np.array([8,6,5,9,4,3,7,2,1,5,6,9,8])
classes = np.array(['bom','ruim','ruim','bom','pessimo','pessimo',
 'bom','pessimo', 'pessimo','ruim','ruim','bom','bom'])
```

Exercício 1

```
#dividir dados em teste e treino
notas_treino, notas_teste, classes_treino, classes_teste = train_test_split(notas.reshape(-1,1),
classes.reshape(-1,1), test_size=0.2, random_state=42)

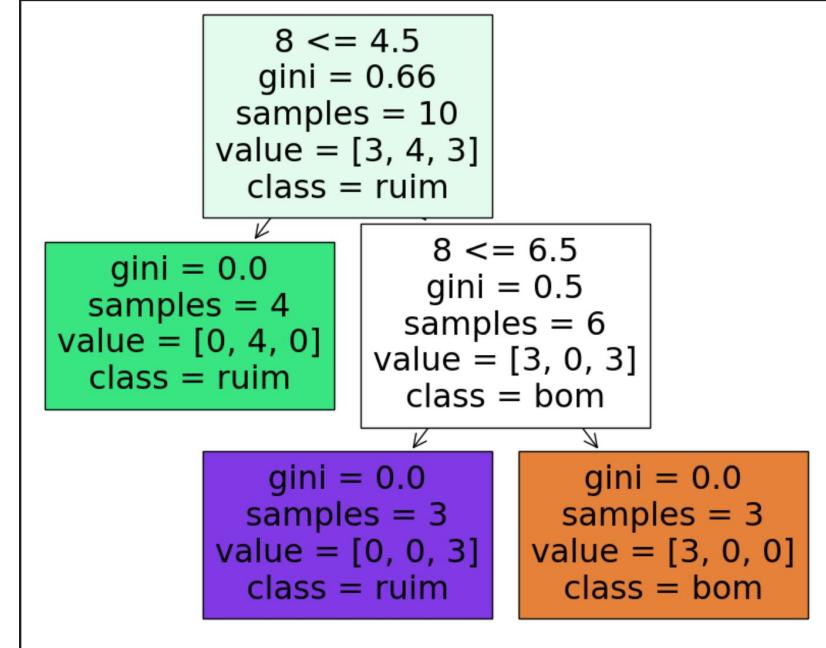
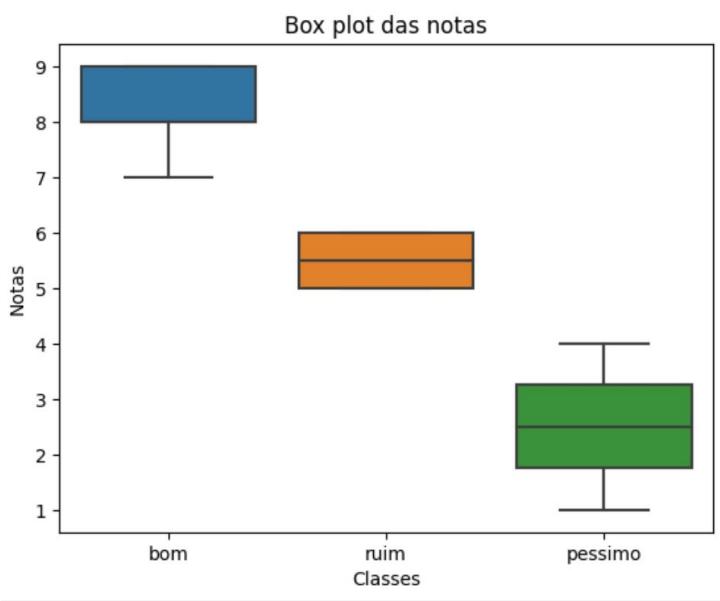
modelo = DecisionTreeClassifier()
modelo.fit(notas_treino,classes_treino)

accuracy = accuracy_score(classes_teste,previsoes)
print(accuracy)

fig = plt.figure(figsize=(10,8))
tree.plot_tree(modelo,feature_names= notas.tolist(), class_names = classes.tolist(), filled= True)
```

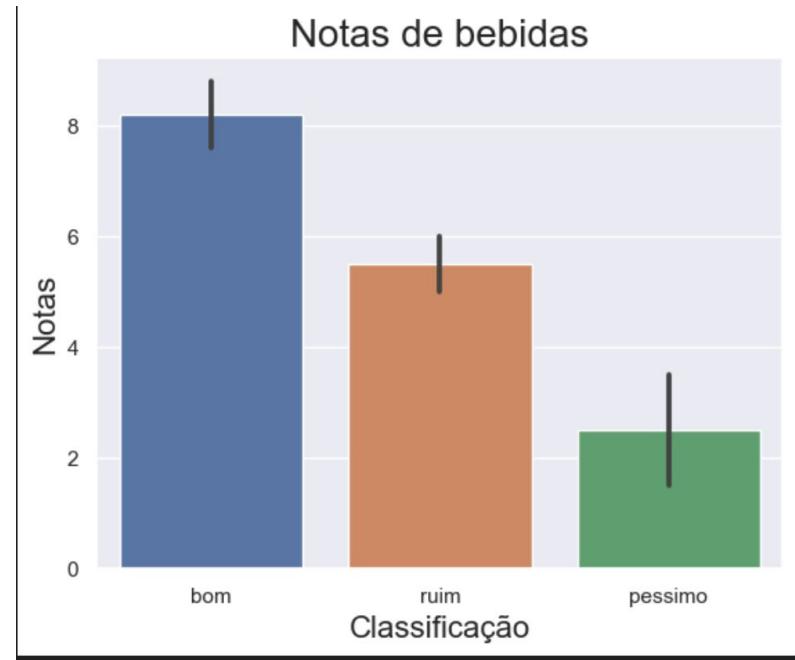
Exercício 2

```
sns.boxplot(x = classes, y = notas)  
plt.title('Box plot das notas')  
plt.ylabel('Notas')  
plt.xlabel('Classes')
```



Exercício 2

```
sns.barplot(y=notas, x = classes)
sns.set(font_scale=1)
plt.title('Notas de bebidas', fontsize=20)
plt.xlabel('Classificação', fontsize=16)
plt.ylabel('Notas', fontsize=16)
```



Situações de aprendizagem

1) Suponha que você está trabalhando em uma fábrica de produtos eletrônicos que produz dispositivos móveis, como smartphones e tablets. Você coletou dados sobre diferentes máquinas usadas na produção e deseja classificá-las como "Máquinas de Montagem" ou "Máquinas de Teste" com base em suas características. As características incluem a velocidade de operação, a complexidade das tarefas que executam e a quantidade de manutenção necessária. Use uma árvore de decisão para classificar as máquinas.

*

Velocidade de Operação	Complexidade da Tarefa	Manutenção Necessária	Classificação
10	Baixa	Baixa	Montagem
5	Alta	Alta	Teste
8	Média	Média	Montagem
6	Alta	Alta	Teste
12	Baixa	Baixa	Montagem
4	Alta	Média	Teste

Situações de aprendizagem

2) Imagine que você trabalha em uma usina nuclear e precisa classificar as máquinas industriais em duas categorias: "Seguras" e "Não Seguras". Você coletou dados sobre características das máquinas, como idade, histórico de manutenção, número de falhas anteriores e nível de automação. Use uma árvore de decisão para determinar se uma máquina é segura ou não com base nessas características.

Idade (anos)	Histórico de Manutenção	Número de Falhas Anteriores	Nível de Automação	Classificação
5	Bom	0	Alto	Segura
10	Ruim	3	Baixo	Não Segura
3	Excelente	0	Médio	Segura
8	Regular	2	Alto	Não Segura
1	Excelente	0	Médio	Segura
15	Ruim	5	Baixo	Não Segura

Situações de aprendizagem

3) Você é um engenheiro de qualidade em uma fábrica de automóveis e deseja classificar as máquinas de produção de acordo com sua contribuição para a qualidade dos produtos finais. As características incluem a precisão na montagem, a velocidade de produção e a taxa de retrabalho. Crie uma árvore de decisão para classificar as máquinas em "Alta Qualidade" e "Baixa Qualidade". *

Precisão na Montagem	Velocidade de Produção	Taxa de Retrabalho	Classificação
Alta	Média	Baixa	Alta Qualidade
Média	Baixa	Alta	Baixa Qualidade
Alta	Alta	Baixa	Alta Qualidade
Média	Alta	Baixa	Alta Qualidade
Baixa	Baixa	Alta	Baixa Qualidade
Baixa	Média	Alta	Baixa Qualidade

Situações de aprendizagem

4) Em uma planta de produção industrial, você deseja classificar as máquinas com base em sua eficiência energética. As características incluem o consumo de energia, o tempo de operação e o tipo de energia utilizada (por exemplo, elétrica ou a gás). Use uma árvore de decisão para classificar as máquinas em "Eficientes" e "Ineficientes" em termos de uso de energia.

Consumo de Energia (kWh)	Tempo de Operação (horas)	Tipo de Energia	Classificação
1000	200	Elétrica	Eficiente
3000	500	Gás	Ineficiente
1500	300	Elétrica	Eficiente
2500	400	Gás	Ineficiente
1200	250	Elétrica	Eficiente
3500	600	Gás	Ineficiente

Situações de aprendizagem

5) Suponha que você é responsável por determinar quando realizar a manutenção preventiva em diferentes máquinas em uma fábrica de processamento de alimentos. As características incluem a idade das máquinas, o número de horas de operação desde a última manutenção e o histórico de falhas. Use uma árvore de decisão para classificar as máquinas em "Necessidade de Manutenção" e "Não Necessidade de Manutenção".

Idade (anos)	Horas Desde a Última Manutenção	Histórico de Falhas	Classificação
2	50	Nenhuma	Necessita
6	200	2	Necessita
1	20	Nenhuma	Necessita
4	150	1	Não Necessita
3	100	3	Necessita
5	180	2	Não Necessita

Situações de aprendizagem

<https://docs.google.com/forms/d/1IkXeARrcTRLBx9kQCzXqvnzGrdprIH1DGPESNXpa3uA/edit>



Obrigado!

Prof. Me Daniel Vieira

Email: danielvieira2006@gmail.com

Linkedin: Daniel Vieira

Instagram: Prof daniel.vieira95

