

COMPONENTE CURRICULAR:	Projeto Aplicado II
NOME COMPLETO DO ALUNO:	Amarilis Oliveira dos Reis Nicole Xavier do Nascimento Lourenço Netto Ribeiro Correa Lucas José de Carvalho Anastacio
RA:	10443156 10437680 10441018 10441680

**CLASSIFICAÇÃO DE ESPÉCIES DE FLORES COM BASE EM  
CARACTERÍSTICAS FÍSICAS**

## Sumário

1. APRESENTAÇÃO DO GRUPO.....	3
2. PREMISSAS DO PROJETO.....	3
2.1 Definição da organização.....	3
2.2 Área de atuação.....	3
2.3 Dados que serão utilizados.....	4
3. OBJETIVOS E METAS.....	4
4. CRONOGRAMA DE ATIVIDADES.....	5
5. DEFINIÇÃO DOS PACOTES USADOS.....	5
6. ANÁLISE EXPLORATÓRIA DA BASE DE DADOS E TRATAMENTO.....	6
6.1 O QUE A CÉLULA ANTERIOR NOS INFORMA.....	7
6.2 O QUE A CÉLULA ANTERIOR NOS INFORMA.....	8
7. VISUALIZAÇÃO.....	8
7.1 O QUE A CÉLULA ANTERIOR NOS INFORMA.....	9
7.2 O QUE A CÉLULA ANTERIOR NOS INFORMA.....	10
7.3 O QUE A CÉLULA ANTERIOR NOS INFORMA.....	11
8. DEFINIÇÃO DAS BASES TEÓRICAS.....	12
9. CÁLCULO DA ACURÁCIA.....	14
9.1 INTERPRETAÇÃO DA MATRIZ DE CONFUSÃO.....	15
9.2 SOBRE A ACURÁCIA DE 100%.....	16
10. LINK DO GITHUB.....	16

## 1. Apresentação do grupo

Somos um grupo de estudantes de Ciência de Dados, e estamos desenvolvendo este projeto como parte de nossa formação acadêmica, aplicando conceitos fundamentais da área em um estudo prático. Nosso objetivo é explorar técnicas de aprendizado de máquina para resolver um problema de classificação de espécies de flores do gênero “*Iris*”, consolidando nosso conhecimento sobre análise de dados, modelagem preditiva e avaliação de modelos.

## 2. Premissas do projeto

Para contextualizar nossa solução, criamos a IrisScan, uma organização fictícia projetada para representar um cenário realista no qual nossa aplicação poderia ser implementada.

### 2.1 Definição da organização

A IrisScan é uma empresa fictícia dedicada ao desenvolvimento de soluções tecnológicas para a identificação e classificação de espécies vegetais. Seu foco é combinar ciência e inovação para facilitar o trabalho de pesquisadores e profissionais da botânica por meio do uso de inteligência artificial. A empresa busca oferecer ferramentas que automatizam a identificação de flores e plantas, tornando o processo mais rápido e preciso.

### 2.2 Área de atuação

A IrisScan atua no setor de tecnologia aplicada à botânica e agricultura. Suas soluções utilizam aprendizado de máquina e análise de dados para auxiliar pesquisadores, agrônomos e entusiastas na identificação e estudo de espécies vegetais. O objetivo é integrar a tecnologia ao conhecimento biológico, proporcionando ferramentas acessíveis para monitoramento, preservação ambiental e pesquisas científicas.

### 2.3 Dados que serão utilizados

Neste estudo, utilizamos o conjunto de dados Iris, um dos mais conhecidos na área de aprendizado de máquina. Ele contém informações sobre três espécies de flores (Iris setosa, Iris versicolor e Iris virginica), com quatro características numéricas:

- Comprimento da sépala
- Largura da sépala
- Comprimento da pétala
- Largura da pétala

O conjunto de dados conta com 150 amostras balanceadas, sendo 50 de cada espécie, e será utilizado para treinar um modelo capaz de classificar automaticamente a espécie de uma flor com base em suas características físicas.

## 3. Objetivos e metas

O principal objetivo deste projeto é desenvolver um modelo de aprendizado de máquina que consiga classificar corretamente a espécie de uma flor com base em seus atributos físicos.

Para isso, definimos as seguintes metas:

- Realizar uma análise exploratória do conjunto de dados para entender suas características e padrões.
- Aplicar técnicas de pré-processamento para garantir a qualidade dos dados.
- Treinar e avaliar diferentes modelos de classificação, como K-Nearest Neighbors (KNN), Regressão Logística e Árvores de Decisão, comparando seus desempenhos.
- Alcançar uma precisão mínima de 90% na classificação das espécies.
- Produzir visualizações e insights que ajudem a interpretar os resultados obtidos.

## 4. Cronograma de atividades

Para organizar o desenvolvimento do projeto, seguimos o seguinte cronograma de atividades:

Fase	Atividade	Período
Etapa 1	Definição do problema e pesquisa sobre o conjunto de dados.	03/03
Etapa 2	Análise exploratória e pré processamento dos dados; implementação e testes de modelos de classificação; avaliação de desempenho e ajustes nos modelos.	31/03
Etapa 3	Apresentação de produtos e storytelling.	28/04
Etapa 4	Elaboração do relatório final e apresentação do projeto.	26/05

## 5. Definição dos pacotes usados

A linguagem de programação escolhida foi o Python, pois trabalharemos com o Scikit (comumente se referindo a Scikit-Learn) o qual é uma biblioteca de aprendizado de máquina de código aberto para Python. Ela fornece ferramentas simples e eficientes para mineração e análise de dados, construídas sobre NumPy, SciPy e Matplotlib. Ela inclui implementações de vários algoritmos de aprendizado de máquina para classificação, regressão, agrupamento, redução de dimensionalidade e muito mais.

## 6. Análise exploratória da base de dados e tratamento

O conjunto de dados Iris é um dos mais estudados em Machine Learning e já possui diversas análises exploratórias disponíveis. No entanto, nós realizamos nossa própria análise para entender melhor os padrões dos dados.

```
[1]: import sklearn
print(sklearn.__version__) # Verificando a versão instalada
```

1.6.1

```
[2]: from sklearn import datasets

# Carregando o conjunto de dados
iris = datasets.load_iris()
print("Conjunto de dados carregado com sucesso!")
```

Conjunto de dados carregado com sucesso!

```
[3]: # Converter para DataFrame
import pandas as pd
df = pd.DataFrame(iris.data, columns=iris.feature_names)

# Adicionar coluna das espécies
df['species'] = iris.target
df['species'] = df['species'].map({0: 'setosa', 1: 'versicolor', 2: 'virginica'})

# Mostrar as primeiras linhas
df.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

species  
0 setosa  
1 setosa  
2 setosa  
3 setosa  
4 setosa

[4]: `df.info()`

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
#   Column                Non-Null Count  Dtype
---  -
0   sepal length (cm)      150 non-null    float64
1   sepal width (cm)       150 non-null    float64
2   petal length (cm)      150 non-null    float64
3   petal width (cm)       150 non-null    float64
4   species                150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

## 6.1 O que a célula anterior nos informa:

Número de linhas e colunas

Tipos de dados de cada coluna

Se há valores ausentes

[5]: `df.describe()`

```
count    sepal length (cm)  sepal width (cm)  petal length (cm)  \
mean          5.843333          3.057333          3.758000
std           0.828066          0.435866          1.765298
min           4.300000          2.000000          1.000000
25%           5.100000          2.800000          1.600000
50%           5.800000          3.000000          4.350000
75%           6.400000          3.300000          5.100000
max           7.900000          4.400000          6.900000

count    petal width (cm)
mean          1.199333
std           0.762238
min           0.100000
25%           0.300000
50%           1.300000
75%           1.800000
max           2.500000
```

## 6.2 O que a célula anterior nos informa:

Média, mínimo, máximo, desvio padrão para cada característica

O intervalo de valores

```
[6]: df.isnull().sum()
```

```
[6]: sepal length (cm)    0
      sepal width (cm)   0
      petal length (cm)  0
      petal width (cm)   0
      species           0
      dtype: int64
```

A saída é 0 para todas as colunas porque o conjunto de dados Iris não contém valores ausentes.

```
[7]: df['species'].value_counts()
```

```
[7]: species
      setosa      50
      versicolor  50
      virginica   50
      Name: count, dtype: int64
```

Cada espécie (Setosa, Versicolor, Virginica) tem 50 amostras.

## 7. Visualização

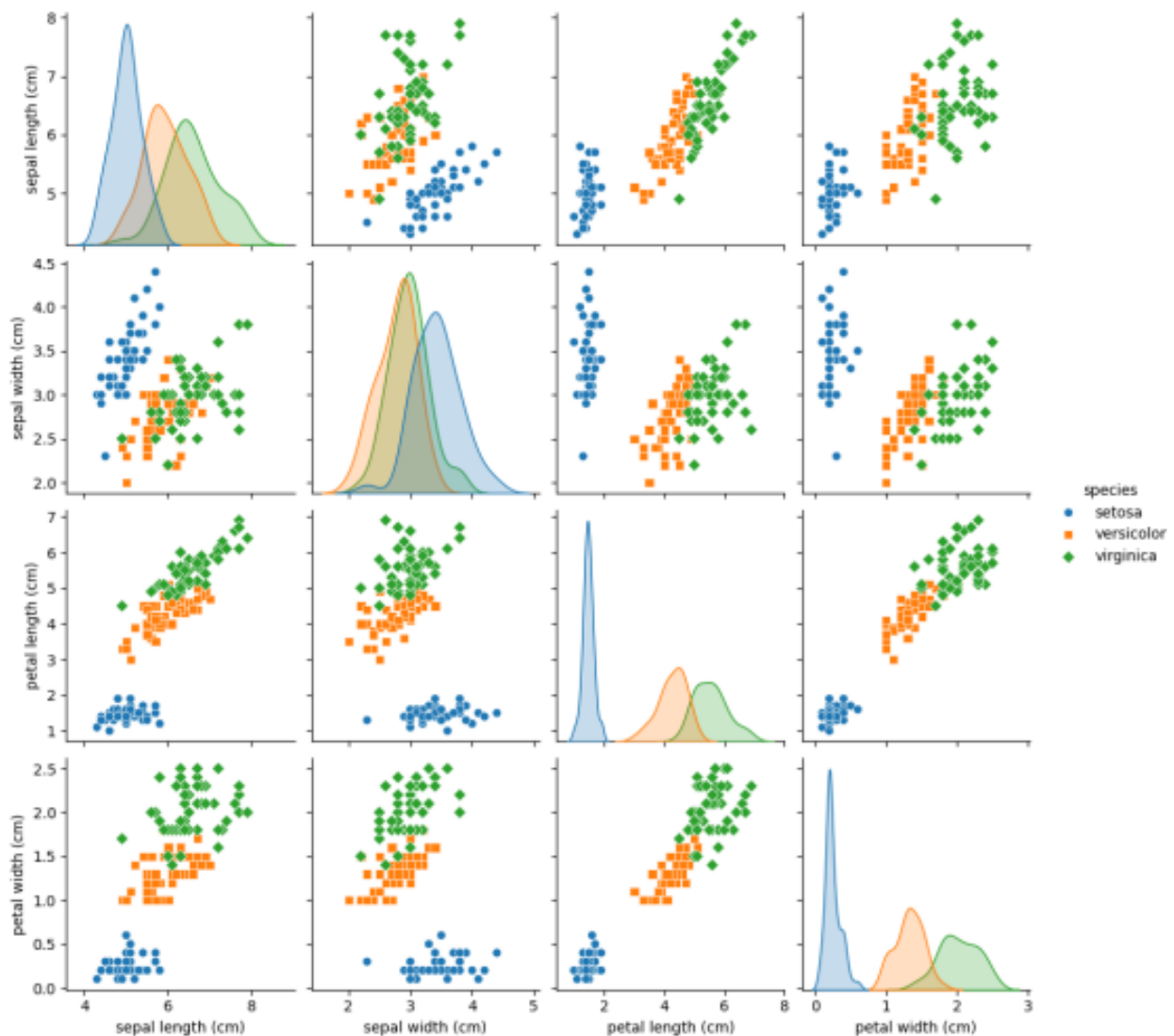
Etapa 1: Importar bibliotecas necessárias

```
[8]: import matplotlib.pyplot as plt
      import seaborn as sns
```

Etapa 2: Pairplot – Relacionamentos gerais de recursos

```
[9]: sns.pairplot(df, hue="species", diag_kind="kde", markers=["o", "s", "D"])
      plt.show()
```





### 7.1 O que a célula anterior nos informa:

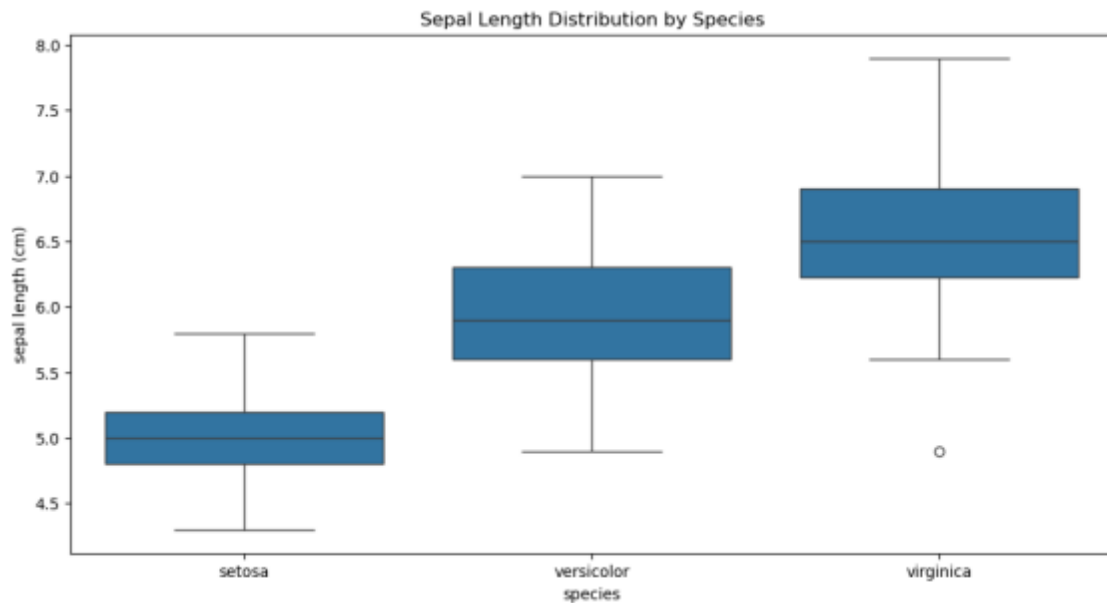
Diagramas de dispersão comparando todas as características.

Cores diferentes representam espécies diferentes.

Diagramas diagonais mostram a distribuição de cada característica.

### Etapa 3: Boxplots – Distribuições de recursos

```
[10]: plt.figure(figsize=(12, 6))
sns.boxplot(data=df, x="species", y="sepal length (cm)")
plt.title("Sepal Length Distribution by Species")
plt.show()
```



## 7.2 O que a célula anterior nos informa:

Como o comprimento da sépala varia entre as espécies.

Se alguma espécie tem uma extensão maior ou valores atípicos.

Etapa 4: Gráfico de violino – Distribuições de recursos com densidade

```
[11]: plt.figure(figsize=(12, 6))
sns.violinplot(data=df, x="species", y="petal width (cm)", inner="quartile")
plt.title("Petal Width Distribution by Species")
plt.show()
```



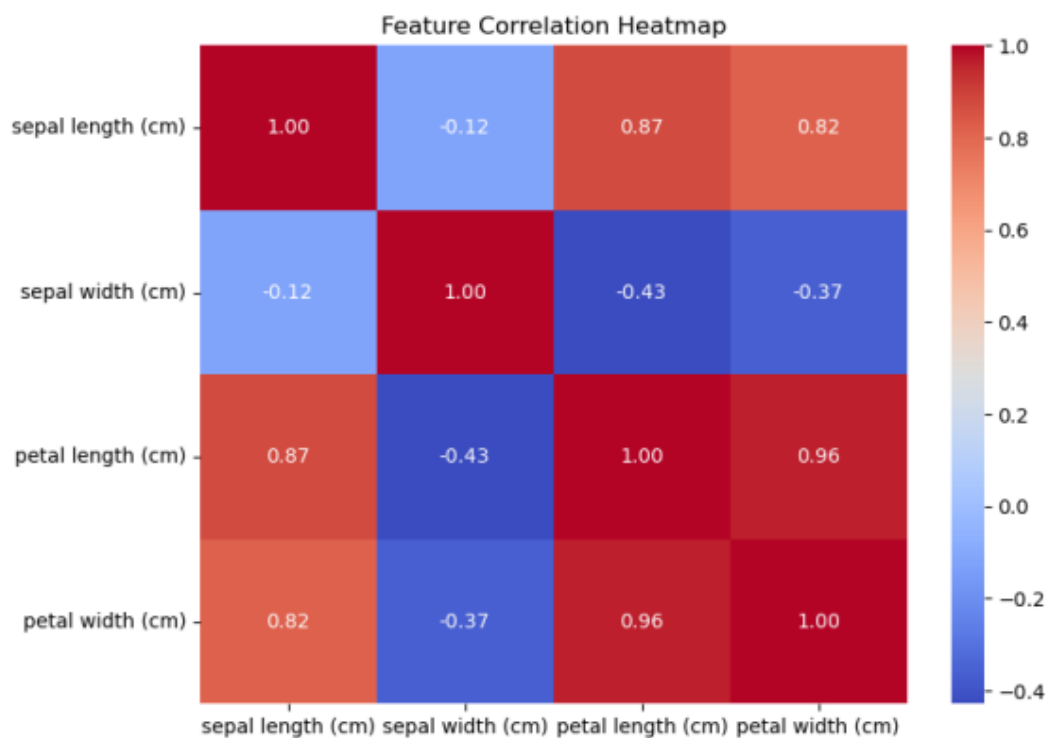
Por que usar um gráfico de violino:

Combina um boxplot e um gráfico de densidade. Mostra onde a maioria dos pontos de dados estão concentrados.

### Etapa 5: Mapa de calor – Correlação entre recursos

[12]:

```
plt.figure(figsize=(8,6))
sns.heatmap(df.drop(columns=["species"]).corr(), annot=True, cmap="coolwarm",
            fmt=".2f")
plt.title("Feature Correlation Heatmap")
plt.show()
```



### 7.3 O que a célula anterior nos informa:

Quais recursos são altamente correlacionados (por exemplo, comprimento da pétala e largura da pétala).

Ajuda a decidir quais recursos podem ser redundantes.

## 8. Definição das bases teóricas

Para realizar a classificação das flores de Íris a partir de imagens, utilizamos conceitos de Visão Computacional, Aprendizado de Máquina e Redes Neurais Convolucionais (CNNs).

**Visão Computacional:** a visão computacional é um campo da Inteligência Artificial que permite que máquinas “vejam” e processem imagens. No nosso projeto, usamos a biblioteca OpenCV para:

- Carregar e processar imagens (ajustar tamanho, normalizar pixels, etc.).
- Converter imagens para um formato adequado para redes neurais.
- Aplicar técnicas de aumento de dados (data augmentation), como espelhamento e rotação, para melhorar o aprendizado do modelo.

**Redes Neurais Convolucionais (CNNs):** Redes Neurais Convolucionais (Convolutional Neural Networks – CNNs) são um tipo de rede neural projetada para processar imagens. Diferente de redes neurais tradicionais, as CNNs utilizam camadas de convolução para detectar padrões visuais, como bordas, formas e texturas.

**A arquitetura básica de uma CNN inclui:**

**Camadas de Convolução:** Aplicam filtros para extrair características importantes da imagem.

**Camadas de Pooling:** Reduzem a dimensionalidade das imagens, mantendo as informações mais relevantes.

**Camadas Fully Connected (Densas):** Responsáveis pela classificação final.

Usaremos um modelo pré-treinado (Transfer Learning, como MobileNetV2 ou ResNet50) para evitar treinar um modelo do zero, o que economiza tempo e melhora o desempenho, principalmente se tivermos um conjunto de dados pequeno.

## Transfer Learning (Aprendizado por Transferência)

O Transfer Learning consiste em reutilizar um modelo já treinado em grandes bases de imagens (como ImageNet) para resolver um novo problema. Em vez de treinar uma CNN do zero, ajustamos a camada final do modelo para classificar as três espécies de Íris.

Isso é útil porque:

- Modelos pré-treinados já aprenderam a identificar formas e padrões básicos.
- Funciona bem mesmo com poucas imagens.
- Reduz o tempo de treinamento e melhora a precisão.

## Algoritmo de Otimização e Função de Perda

Nosso modelo será treinado usando:

- Função de perda: Categorical Crossentropy – usada para problemas de classificação multiclasse.
- Otimização: Adam (Adaptive Moment Estimation) – um dos otimizadores mais eficientes para ajustar os pesos da rede neural.

A equação da função de perda Crossentropy para 3 classes é:

$$\text{Loss} = - \sum_{i=1}^3 y_i \log (\hat{y}_i)$$

Onde: •  $y_i$  é o valor real da classe (1 para a classe correta, 0 para as demais).

- $\hat{y}_i$  é a probabilidade prevista pelo modelo.

## 9. Cálculo da acurácia

Etapa 1: Dividindo os Dados (Divisão de Treinamento-Teste)

Antes de treinar um modelo, dividimos o conjunto de dados em:

- Conjunto de treinamento (por exemplo, 80%) – Usado para treinar o modelo
- Conjunto de teste (por exemplo, 20%) – Usado para avaliar o desempenho do modelo

```
[13]: from sklearn.model_selection import train_test_split

# Definir características (X) e alvo (y)
X = df.drop(columns=["species"]) # Características
y = df["species"] # Alvo

# Dividir dados entre 80% treinamento e 20% teste

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

Etapa 2: Treinando um modelo

Classificador simples (regressão logística):

```
[14]: from sklearn.linear_model import LogisticRegression

# Inicializar e treinar o modelo
model = LogisticRegression()
model.fit(X_train, y_train)

[14]: LogisticRegression()
```

Etapa 3: Fazendo previsões

Após o treinamento, usamos o modelo para prever espécies no conjunto de teste:

```
[15]: y_pred = model.predict(X_test)
```

Etapa 4: Calculando a pontuação de acurácia

Agora, calculamos a acurácia usando `accuracy_score` do Scikit-learn:

```
[16]: from sklearn.metrics import accuracy_score

# Calcular acurácia
accuracy = accuracy_score(y_test, y_pred)
print(f"Model Accuracy: {accuracy:.2f}")
```

Model Accuracy: 1.00

Etapa 5: Implementando KNN (K-Nearest Neighbors)

```
[20]: # Importar o classificador KNN do sklearn
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix

# Inicializar o classificador KNN
knn = KNeighborsClassifier(n_neighbors=3)

# Treinar o modelo nos dados de treinamento
knn.fit(X_train, y_train)

# Fazer previsões sobre os dados de teste
y_pred = knn.predict(X_test)

# Calcular acurácia

accuracy = accuracy_score(y_test, y_pred)

# Print da acurácia
print(f"KNN Model Accuracy: {accuracy:.2f}")

# Matriz de confusão
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(cm)
```

KNN Model Accuracy: 1.00

Confusion Matrix:

```
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```

## 9.1 Interpretação da Matriz de Confusão:

A primeira linha ([10, 0, 0]) significa que todas as 10 flores Setosa foram corretamente classificadas como Setosa.

A segunda linha ([0, 9, 0]) significa que todas as 9 flores Versicolor foram corretamente classificadas como Versicolor.

A terceira linha ([0, 0, 11]) significa que todas as 11 flores Virginica foram corretamente classificadas como Virginica.

Como não há classificações erradas (nenhum valor fora da diagonal), seu modelo previu perfeitamente todas as espécies.

## 9.2 Sobre a acurácia de 100%

O conjunto de dados Iris é frequentemente considerado um conjunto de dados “de brinquedo” (“toy dataset”), o que significa que é muito bem estruturado e separável com modelos simples. Como é um conjunto de dados pequeno e fácil de aprender, o modelo memorizou os exemplos de treinamento em vez de aprender padrões generalizáveis. Isso pode resultar em uma acurácia muito alta, como no caso deste modelo (100%).

## 10. Link do GitHub

<https://github.com/iamni2001/IriScan>