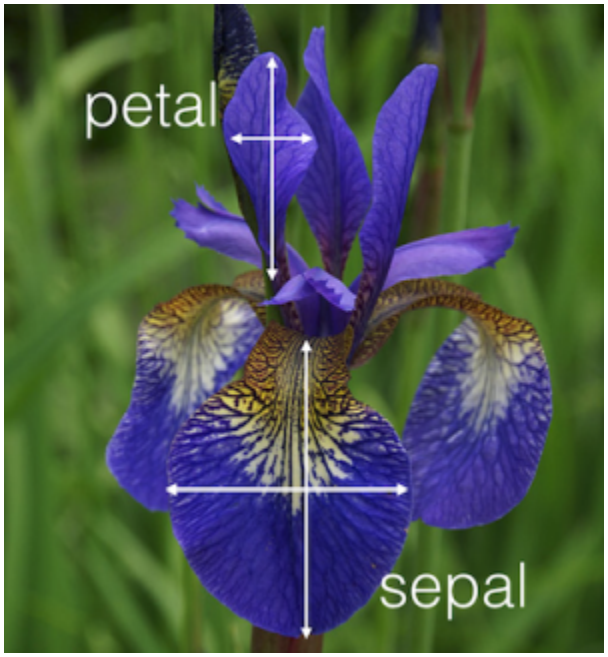
Guía Paso a Paso

Modelo de Clasificación con Scikit-learn en Google Colab

Dataset Iris

🎯 Objetivo

Construir un modelo de clasificación que prediga a qué especie pertenece una flor (setosa, versicolor, virginica) en función de sus características (largo y ancho del sépalo y pétalo), utilizando Python, Scikit-learn y Google Colab.



✅ No necesitas instalar nada en tu computadora, solo tener una cuenta de Google.

🚀 Paso 1: Abre Google Colab

- Ve a https://colab.research.google.com/
- Haz clic en **"Nuevo cuaderno"**
- Cambia el nombre a: `modelo_clasificacion_iris`

📦 Paso 2: Importar librerías necesarias

En la primera celda, escribe este código:

```
import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt


from sklearn.datasets import load_iris

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification_report, confusion_matrix,
ConfusionMatrixDisplay
```

🌸 Paso 3: Cargar el dataset Iris

```
# Cargar dataset iris

iris = load_iris()


# Convertirlo en un DataFrame

df = pd.DataFrame(data=iris.data, columns=iris.feature_names)

df['target'] = iris.target

df['species'] = df['target'].apply(lambda i: iris.target_names[i])


# Mostrar las primeras filas

df.head()
```

🔍 Paso 4: Explorar y visualizar los datos

```
# Información general

df.info()
```

```
# Estadísticas
df.describe()

# Visualizar relaciones entre variables
sns.pairplot(df, hue="species")
plt.show()
```

Preguntas que pueden hacerle a ChatGPT:

- *¿Qué representa cada columna del dataset iris?*
- *¿Por qué se usa pairplot?*
- *¿Qué significa target?*

🧪 Paso 5: Separar variables y etiquetas

```
X = df[iris.feature_names]  # variables (características)
y = df['target']            # etiquetas (clases)
```

✂️ Paso 6: Dividir en entrenamiento y prueba

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

🌲 Paso 7: Crear el modelo de clasificación

Usaremos un modelo **Random Forest**, fácil y efectivo para principiantes:

```
modelo = RandomForestClassifier(random_state=42)
modelo.fit(X_train, y_train)
```

🔮 Paso 8: Hacer predicciones

y_pred = modelo.predict(X_test)

📊 Paso 9: Evaluar el modelo

9.1 Reporte de clasificación

print(classification_report(y_test, y_pred, target_names=iris.target_names))

9.2 Matriz de confusión visual

cm = confusion_matrix(y_test, y_pred)

disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=iris.target_names)

disp.plot(cmap='Blues')

plt.title("Matriz de Confusión")

plt.show()

¿Qué me indica la matriz de confusión?

La matriz de confusión es una herramienta fundamental para evaluar el rendimiento de un modelo de clasificación. Te indica cuántas veces el modelo acertó o se equivocó al predecir las clases, comparando las predicciones con los valores reales.

¿Qué contiene?

Es una tabla de doble entrada donde:

- Las filas representan las clases reales (etiquetas verdaderas).
- Las columnas representan las clases predichas por el modelo.

Por ejemplo, para el dataset Iris con 3 clases (setosa, versicolor, virginica), una matriz típica podría verse así:

	Pred: setosa	Pred: versicolor	Pred: virginica
Real: setosa			
Real: versicolor			
Real: virginica			

```
# Calcular la Accuracy

accuracy = (cm.diagonal().sum()) / cm.sum()

print(f'Accuracy: {accuracy:.2f}')

# Calculate Precision for each class

precision = np.diag(cm) / np.sum(cm, axis = 0)

print("Precision for each class:")

for i, p in enumerate(precision):

   print(f"  Class {iris.target_names[i]}: {p:.2f}")

# Calculate Precision for each class

precision = np.diag(cm) / np.sum(cm, axis = 0)

print("Precision for each class:")

for i, p in enumerate(precision):

   print(f"  Class {iris.target_names[i]}: {p:.2f}")

# Calculate Precision for each class

precision = np.diag(cm) / np.sum(cm, axis = 0)

print("Precision for each class:")

for i, p in enumerate(precision):

   print(f"  Class {iris.target_names[i]}: {p:.2f}")
```

📈 Paso 10: Visualización de importancia de características

importancias = modelo.feature_importances_

features = iris.feature_names

El siguiente paso es muy importante ¿En qué radica su importancia?

plt.figure(figsize=(8, 4))

sns.barplot(x=importancias, y=features)

plt.title("Importancia de características")

plt.xlabel("Importancia")

plt.ylabel("Características")

plt.show()

Paso 11: Guardar el modelo en formato .pkl"

Para utilizar JOBLIB con la finalidad de guardar, asegúrate de tenerlo instalado en COLAB, puede usar:

```
!pip install joblib
```

Ahora sí, ya puedes importarlo:

import joblib

joblib.dump(modelo, 'modelo_iris.pkl')

Descargar el archivo en tu computadora

from google.colab import files

files.download('modelo_iris.pkl')

Paso 12: Cargar el modelo en otro Colab",

```
from google.colab import files

uploaded = files.upload()  # Sube el archivo modelo_iris.pkl


import joblib

modelo_cargado = joblib.load('modelo_iris.pkl')


# Usar el modelo cargado

modelo_cargado.predict(X_test)
```

🤖 ¿Cómo usar ChatGPT como asistente?

🧩 Puedes preguntarle:

- *¿Qué significa la métrica precision/recall/f1-score?*
- *¿Qué otro modelo de clasificación puedo probar?*
- *Explícame este bloque de código línea por línea.*
- *¿Qué significa overfitting o underfitting?*

✅ ¿Qué aprendiste?

- Cómo cargar y explorar un dataset clásico en Python.
- Cómo construir un modelo de clasificación con Scikit-learn.
- Cómo interpretar métricas como precisión, recall, F1 y matriz de confusión.
- Cómo visualizar relaciones y la importancia de características.
- Cómo apoyarte en ChatGPT para entender código, conceptos y errores.