🔷 1. Import Required Libraries

✅ We import the dataset, the classifier, and a tool to measure accuracy.

```
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
```

The Iris dataset is one of the most famous and simple datasets in machine learning.

🌼 It includes 150 flowers from the Iris species.

📏 Each flower has 4 features:

Sepal length

Sepal width

Petal length

Petal width

🎯 The goal is to predict the species:

Iris Setosa

Iris Versicolor

Iris Virginica

```
from sklearn.datasets import load_iris
import pandas as pd

# Load the dataset
iris = load_iris()

# Create a DataFrame from the data
df = pd.DataFrame(iris.data, columns=iris.feature_names)
df["target"] = iris.target
df["species"] = pd.Categorical.from_codes(iris.target, iris.target_names)

# Show the first 5 rows
df.head()
```

Visualize with a Pair Plot

```
import seaborn as sns
import matplotlib.pyplot as plt

# Pairplot of the features
sns.pairplot(df, hue="species")
plt.show()
```

🔷 2. Load the Dataset

✅ X holds the input data; y holds the class we want to predict.

```
iris = load_iris()
X = iris.data    # özellikler (sepal length, petal width, vs.)
y = iris.target  # etiketler (0=setosa, 1=versicolor, 2=virginica)
```

🔷 3. Split the Dataset

✅ We keep 80% of the data for training and 20% for testing.

```
from sklearn.model_selection import train_test_split

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

🔷 4. Create and Train the Model

✅ The model learns from the training data — like answering a flow of yes/no questions.

```
model = DecisionTreeClassifier()
model.fit(X_train, y_train)
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

🔷 5. Make Predictions and Evaluate the Model

✅ The model makes predictions. We check how many are correct.

```
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)

print("Accuracy:", accuracy)
```

```
Accuracy: 1.0
```

✅ Accuracy: 1.0 – What Does It Mean? The model predicted all test samples correctly.

This looks great, but it can also mean the model is overfitting.

Iris dataset is small and clean, so high accuracy is common.

In real-world data, 100% accuracy is rare.

Let's Predicted A Flower Type

```
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# 1. Load the dataset
iris = load_iris()
X = iris.data  # Features (sepal & petal sizes)
y = iris.target  # Target (flower types)

# 2. Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 3. Train the Decision Tree model
model = DecisionTreeClassifier()
model.fit(X_train, y_train)

# 4. Evaluate the model using test data
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("✅ Model Accuracy on Test Data:", round(accuracy * 100, 2), "%")

# 5. Predict using user input
print("\n🔍 Now let's make a prediction based on your input!")
print("Enter flower measurements:")
sepal_length = float(input("Sepal length (cm): "))
sepal_width = float(input("Sepal width (cm): "))
petal_length = float(input("Petal length (cm): "))
petal_width = float(input("Petal width (cm): "))

# 6. Format the input and make prediction
new_data = [[sepal_length, sepal_width, petal_length, petal_width]]
prediction = model.predict(new_data)

# 7. Show the result
print("🌸 Predicted Flower Type:", iris.target_names[prediction[0]])
```

```
✅ Model Accuracy on Test Data: 100.0 %

🔍 Now let's make a prediction based on your input!
Enter flower measurements:
Sepal length (cm): 5.1
Sepal width (cm): 3.5
Petal length (cm): 1.4
Petal width (cm): 0.2
🌸 Predicted Flower Type: setosa
```

Sepal length (cm): 5.1 Sepal width (cm): 3.5 Petal length (cm): 1.4 Petal width (cm): 0.2