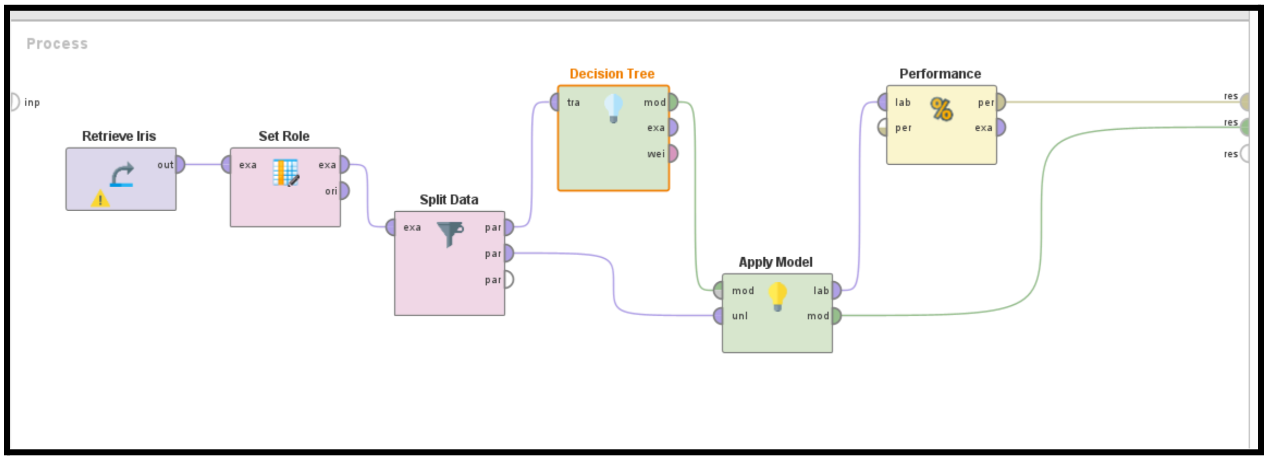

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
import numpy as np
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt

# --- CONFIGURATION: Change these for a new dataset ---
# ⚠️ 1. Change the file name
DATASET_FILE = 'Iris.csv'

# ⚠️ 2. Change the name of the target/label column
TARGET_COLUMN = 'Species' # Change this to your new target column name
(e.g., 'Target', 'Diagnosis')

# ⚠️ 3. Change the name of the unique ID column to drop (or set to None if
no ID column exists)
ID_COLUMN_TO_DROP = 'Id' # Change this to the ID column (or set to None)
# --- END CONFIGURATION ---



# --- 1. Preprocess data. Split data into train and test set ---

# Load the dataset
print("Loading data...")
```

```python
try:
    df = pd.read_csv(DATASET_FILE)
    print(f"Successfully loaded '{DATASET_FILE}'.")
except FileNotFoundError:
    print(f"Error: '{DATASET_FILE}' not found. Make sure the file is in the same directory.")
    exit()

# Drop the specified ID column if it exists
if ID_COLUMN_TO_DROP and ID_COLUMN_TO_DROP in df.columns:
    df = df.drop(ID_COLUMN_TO_DROP, axis=1)
    print(f"Dropped ID column: '{ID_COLUMN_TO_DROP}'")
else:
    print("No specified ID column to drop, or column not found.")


# Separate features (X) and target (y)
# Features (X) will now be all columns *except* the target column
try:
    X = df.drop(TARGET_COLUMN, axis=1)
    y = df[TARGET_COLUMN]
    print(f"Features set (X) columns: {X.columns.tolist()}")
    print(f"Target variable (y): '{TARGET_COLUMN}'")
except KeyError:
    print(f"Error: Target column '{TARGET_COLUMN}' not found in the dataset.")
    exit()

# Split the data into training (80%) and testing (20%) sets
# random_state ensures reproducibility
print("Splitting data into 80% training and 20% testing...")
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Display shapes of the split data
print(f"X_train shape: {X_train.shape}")
print(f"X_test shape: {X_test.shape}")
print("-" * 50)
```

```python
# --- 2. Build a Classification model using the inbuilt library function
on training data ---

# Initialize the Decision Tree Classifier
dt_model = DecisionTreeClassifier(random_state=42)

# Train the model on the training data
print("Training Decision Tree Classifier...")
dt_model.fit(X_train, y_train)
print("Training complete.")
print("-" * 50)

# Make predictions on the test set
y_pred = dt_model.predict(X_test)

# --- 3. Calculate metrics based on test data using an inbuilt function
---

# Calculate Accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Decision Tree Accuracy: {accuracy:.4f}")

# Generate a Classification Report (Precision, Recall, F1-Score)
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

# Generate Confusion Matrix
conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(conf_matrix)

# Interpretation of Results
print("\n--- Model Evaluation Summary ---")
print(f"The Decision Tree achieved an accuracy of {accuracy:.4f}.")


# 7. Visualize the Tree (Optional but recommended)
plt.figure(figsize=(20, 10))
# Get feature names from X columns (This is now fully dynamic)
feature_names = X.columns.tolist()
```

```python
# Get class names from the model (This is also fully dynamic)
class_names = dt_model.classes_.astype(str).tolist() # Convert class names to string for plot_tree

plot_tree(dt_model, feature_names=feature_names, class_names=class_names, filled=True)
plt.title(f"Decision Tree Visualization for {DATASET_FILE}")
# Save the plot to a file
PLOT_FILENAME = "decision_tree_output.png"
plt.savefig(PLOT_FILENAME)


print(f"Decision Tree plot saved as '{PLOT_FILENAME}'")
```