**✅ Cell 1**

python

CopyEdit

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

* **Problem Step**: Import required libraries.
* **Definition**:
  + pandas: For data manipulation and loading CSV files.
  + matplotlib.pyplot: For creating basic visualizations.
  + seaborn: For advanced and aesthetic visualizations.
* **Why**: These tools are necessary for data preprocessing, analysis, and visualization throughout the notebook.

**✅ Cell 2**

python

CopyEdit

df = pd.read\_csv("iris.csv")

df.head()

* **Problem Step**: Load dataset and display top records.
* **Definition**:
  + pd.read\_csv(): Reads a CSV file into a DataFrame.
  + .head(): Displays the first 5 rows.
* **Why**: To load and take a quick look at the structure of the Iris dataset.

**✅ Cell 3**

python

CopyEdit

df.shape

* **Problem Step**: Get dataset dimensions.
* **Definition**:
  + .shape: Returns number of rows and columns in a DataFrame.
* **Why**: Helps verify the dataset size (e.g., for cleaning or splitting).

**✅ Cell 4**

python

CopyEdit

df.info()

* **Problem Step**: Check data types and missing values.
* **Definition**:
  + .info(): Displays data types, non-null values, and memory usage.
* **Why**: Useful to assess data quality and types for preprocessing.

**✅ Cell 5**

python

CopyEdit

df.describe()

* **Problem Step**: Get statistical summary.
* **Definition**:
  + .describe(): Returns count, mean, std deviation, min, max, etc., for numeric columns.
* **Why**: Provides a numeric overview of each feature's distribution.

**✅ Cell 6**

python

CopyEdit

df.isnull().sum()

* **Problem Step**: Check for null values.
* **Definition**:
  + .isnull(): Flags missing values.
  + .sum(): Sums them per column.
* **Why**: Missing values can lead to errors in model training or analysis.

**✅ Cell 7**

python

CopyEdit

df["species"].value\_counts()

* **Problem Step**: Analyze label distribution.
* **Definition**:
  + .value\_counts(): Counts unique values in a column.
* **Why**: To confirm whether the dataset is balanced across species classes.

**✅ Cell 8**

python

CopyEdit

sns.pairplot(df, hue="species")

plt.show()

* **Problem Step**: Visualize relationships and class separability.
* **Definition**:
  + pairplot(): Creates scatter plots between every pair of features.
  + hue: Color-codes based on class labels.
* **Why**: Helps assess how well-separated the classes are, a key insight for classification.

**✅ Cell 9**

python

CopyEdit

sns.heatmap(df.corr(), annot=True, cmap='coolwarm')

plt.show()

* **Problem Step**: Visualize correlation between features.
* **Definition**:
  + df.corr(): Calculates correlation matrix.
  + sns.heatmap(): Plots correlation values.
* **Why**: To identify strongly correlated features that may influence modeling.

**✅ Cell 10**

python

CopyEdit

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

* **Problem Step**: Import modeling tools.
* **Definition**:
  + train\_test\_split: Splits data into training and testing sets.
  + LogisticRegression: A basic classifier for multiclass classification.
  + accuracy\_score, classification\_report, confusion\_matrix: Tools to evaluate performance.
* **Why**: These are essential for building and assessing the model.

**✅ Cell 11**

python

CopyEdit

X = df.drop("species", axis=1)

y = df["species"]

* **Problem Step**: Define features and target.
* **Definition**:
  + .drop(): Removes a column from the DataFrame.
* **Why**: Splits independent variables (X) and the label (y) for modeling.

**✅ Cell 12**

python

CopyEdit

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

* **Problem Step**: Split data.
* **Definition**:
  + train\_test\_split(): Divides data for training and testing.
  + test\_size=0.2: 20% test, 80% train.
* **Why**: To evaluate model performance on unseen data.

**✅ Cell 13**

python

CopyEdit

model = LogisticRegression()

model.fit(X\_train, y\_train)

* **Problem Step**: Train the model.
* **Definition**:
  + fit(): Trains the model using the training data.
* **Why**: To learn patterns from data and map features to species.

**✅ Cell 14**

python

CopyEdit

y\_pred = model.predict(X\_test)

* **Problem Step**: Make predictions.
* **Definition**:
  + predict(): Uses the trained model to classify new data.
* **Why**: To assess how well the model generalizes to new data.

**✅ Cell 15**

python

CopyEdit

print("Accuracy:", accuracy\_score(y\_test, y\_pred))

print("Confusion Matrix:\n", confusion\_matrix(y\_test, y\_pred))

print("Classification Report:\n", classification\_report(y\_test, y\_pred))

* **Problem Step**: Evaluate model performance.
* **Definition**:
  + accuracy\_score: Ratio of correct predictions.
  + confusion\_matrix: Shows prediction vs. actual class matrix.
  + classification\_report: Shows precision, recall, F1-score for each class.
* **Why**: To comprehensively evaluate how well the model performs.