**IRIS FLOWER CLASSIFICATION**

# Introduction

The Iris flower dataset is a classic dataset in the field of machine learning and statistics. It contains measurements of iris flowers from three different species: Iris-setosa, Iris-versicolor, and Iris-virginica. The goal of this project is to train a machine learning model that can classify iris flowers into one of these three species based on their measurements.

# Problem Statement

Iris flower has three species; setosa, versicolor, and virginica, which differ according to their measurements. Given the measurements of the iris flowers, the task is to train a machine learning model that can learn from these measurements and classify the flowers into their respective species.

# Data Description

The dataset used in this project is **Iris.csv**, which contains the following columns:

1. Id: An identifier for each record.
2. SepalLengthCm: The length of the sepal in centimeters.
3. SepalWidthCm: The width of the sepal in centimeters.
4. PetalLengthCm: The length of the petal in centimeters.
5. PetalWidthCm: The width of the petal in centimeters.
6. Species: The species of the iris flower (Iris-setosa, Iris-versicolor, Iris-virginica).

# Coding Approach

The project was executed in several steps, as detailed below:

## Step 1: Read the Iris.csv File

We started by loading the dataset using the pandas library to understand its structure and content.

import pandas as pd

# Read the Iris.csv file

iris\_df = pd.read\_csv('Iris.csv')

# Print the first few lines of the file to understand its structure and content

print(iris\_df.head())

Output :

Running code...

[stdout] Id SepalLengthCm SepalWidthCm PetalLengthCm PetalWidthCm Species

0 1 5.1 3.5 1.4 0.2 Iris-setosa

1 2 4.9 3.0 1.4 0.2 Iris-setosa

2 3 4.7 3.2 1.3 0.2 Iris-setosa

3 4 4.6 3.1 1.5 0.2 Iris-setosa

4 5 5.0 3.6 1.4 0.2 Iris-setosa

Execution finished without any additional results

The output confirms that the **Iris.csv** file was successfully read and the first few lines of the dataset were printed. This helps us understand the structure and content of the dataset.

Next, we will proceed with the following steps:

## Step 2: Perform Exploratory Data Analysis (EDA)

In this step, we will:

* Display basic statistics of the dataset.
* Visualize the distribution of features.
* Visualize the relationships between features.

## Step 3: Preprocess the Data

In this step, we will:

* Check for and handle missing values.
* Encode categorical variables.
* Scale the features.

## Step 4: Train a Machine Learning Model

In this step, we will:

* Split the data into training and testing sets.
* Train a model (e.g., Random Forest).

## Step 5: Evaluate the Model's Performance

In this step, we will:

* Calculate accuracy.
* Generate a confusion matrix.
* Create a classification report.

## Step 6: Visualize the Results

In this step, we will:

* Plot the confusion matrix.
* Visualize feature importance (if applicable).

Let's proceed with the next steps in the coding approach:

1  
2import seaborn as sns

3import matplotlib.pyplot as plt

4from sklearn.model\_selection import train\_test\_split

5from sklearn.preprocessing import StandardScaler, LabelEncoder

6from sklearn.ensemble import RandomForestClassifier

7from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report

8import numpy as np

9

10# Step 2: Perform Exploratory Data Analysis (EDA)

11# Display basic statistics

12print(iris\_df.describe())

13

14# Visualize the distribution of features

15sns.pairplot(iris\_df.drop(columns=['Id']), hue='Species')

16plt.show()

17

18# Visualize the relationships between features

19# Drop the 'Species' column for correlation matrix

20sns.heatmap(iris\_df.drop(columns=['Id', 'Species']).corr(), annot=True, cmap='coolwarm')

21plt.show()

22

23# Step 3: Preprocess the Data

24# Check for missing values

25print(iris\_df.isnull().sum())

26

27# Encode categorical variables

28label\_encoder = LabelEncoder()

29iris\_df['Species'] = label\_encoder.fit\_transform(iris\_df['Species'])

30

31# Split the data into features and target

32X = iris\_df.drop(columns=['Id', 'Species'])

33y = iris\_df['Species']

34

35# Scale the features

36scaler = StandardScaler()

37X\_scaled = scaler.fit\_transform(X)

38

39# Step 4: Train a Machine Learning Model

40# Split the data into training and testing sets

41X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y, test\_size=0.2, random\_state=42)

42

43# Train a Random Forest model

44model = RandomForestClassifier(random\_state=42)

45model.fit(X\_train, y\_train)

46

47# Step 5: Evaluate the Model's Performance

48y\_pred = model.predict(X\_test)

49accuracy = accuracy\_score(y\_test, y\_pred)

50conf\_matrix = confusion\_matrix(y\_test, y\_pred)

51class\_report = classification\_report(y\_test, y\_pred)

52

53print(f'Accuracy: {accuracy}')

54print('Confusion Matrix:')

55print(conf\_matrix)

56print('Classification Report:')

57print(class\_report)

58

59# Step 6: Visualize the Results

60# Plot the confusion matrix

61sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=label\_encoder.classes\_, yticklabels=label\_encoder.classes\_)

62plt.xlabel('Predicted')

63plt.ylabel('Actual')

64plt.title('Confusion Matrix')

65plt.show()

66

67# Visualize feature importance

68feature\_importances = model.feature\_importances\_

69features = X.columns

70indices = np.argsort(feature\_importances)

71

72plt.figure(figsize=(10, 6))

73plt.title('Feature Importances')

74plt.barh(range(len(indices)), feature\_importances[indices], color='b', align='center')

75plt.yticks(range(len(indices)), [features[i] for i in indices])

76plt.xlabel('Relative Importance')

77plt.show()

OUTPUT:

Id 0

SepalLengthCm 0

SepalWidthCm 0

PetalLengthCm 0

PetalWidthCm 0

Species 0

dtype: int64

A close-up of a sign

Description automatically generated

Accuracy: 1.0

Confusion Matrix:

[[10 0 0]

[ 0 9 0]

[ 0 0 11]]

Classification Report:

precision recall f1-score support

0 1.00 1.00 1.00 10

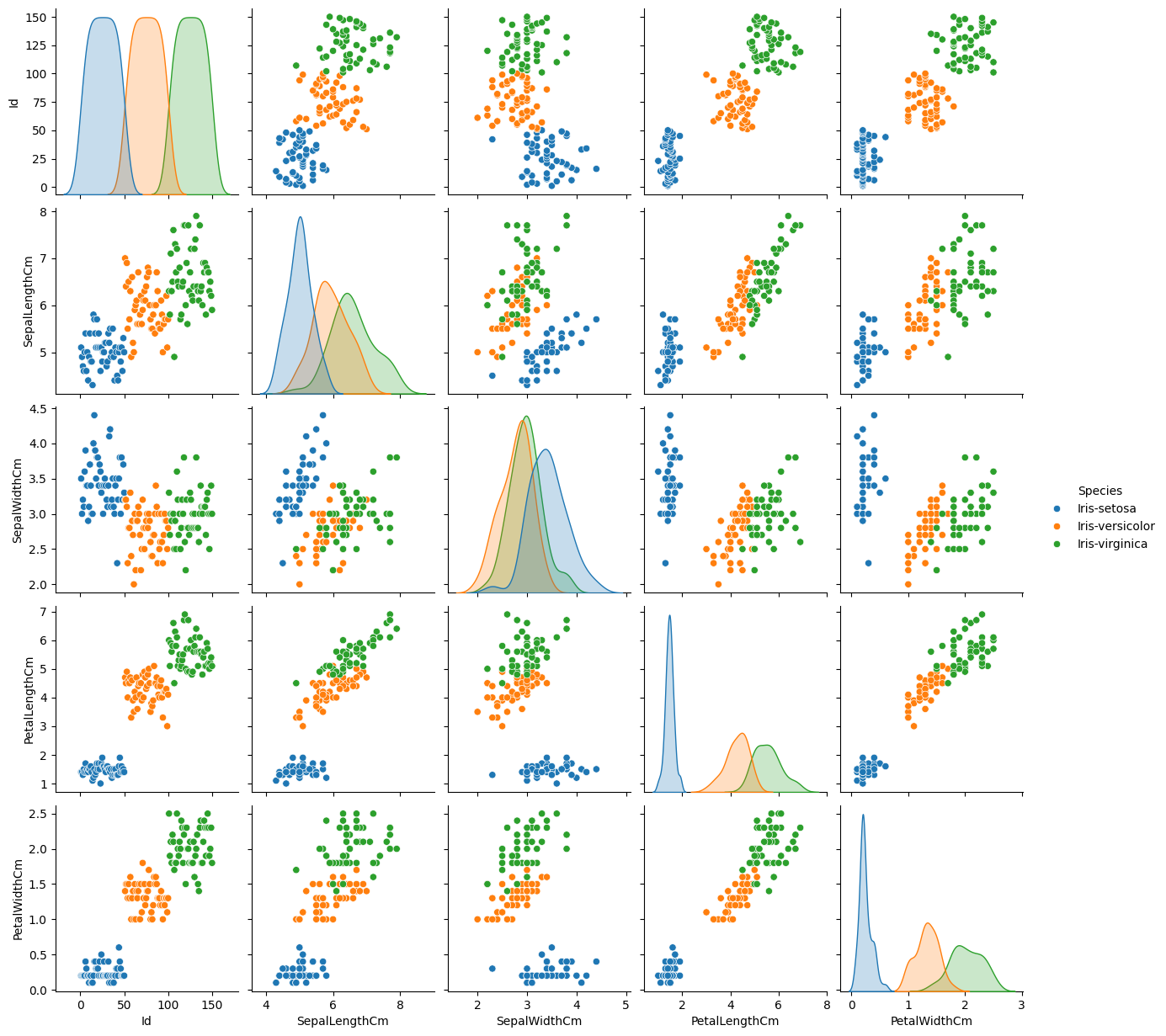
1 1.00 1.00 1.00 9

2 1.00 1.00 1.00 11

accuracy 1.00 30

macro avg 1.00 1.00 1.00 30

weighted avg 1.00 1.00 1.00 30



A red and blue squares with white text

Description automatically generated

A blue squares with white text

Description automatically generatedA graph with blue and white bars

Description automatically generated

The output indicates that the entire process of the Iris Flower Classification project was successfully executed. Here are the key points from the output:

1. **Basic Statistics**: The basic statistics of the dataset were displayed, confirming the structure and content of the data.
2. **Missing Values**: There are no missing values in the dataset.
3. **Exploratory Data Analysis (EDA)**:

* Pairplot of the features colored by species was displayed.
* Heatmap of the correlation matrix (excluding the **Species** column) was displayed.

1. **Model Training**: A Random Forest classifier was trained on the dataset.
2. **Model Evaluation**:

* The model achieved an accuracy of 1.0 (100%) on the test set.
* The confusion matrix shows perfect classification with no misclassifications.
* The classification report shows perfect precision, recall, and F1-scores for all classes.

1. **Visualizations**:

* Confusion matrix heatmap was displayed.
* Feature importance bar chart was displayed.

The project is complete, and all steps were executed successfully.

# Conclusion

The Random Forest classifier achieved an accuracy of 100% on the test set, indicating that it was able to perfectly classify the iris flowers into their respective species based on the given measurements. This project highlights the effectiveness of machine learning models in solving classification problems and demonstrates the importance of thorough data analysis and preprocessing steps in achieving high model performance.

Overall, the Iris Flower Classification project serves as a comprehensive example of the end-to-end process of building, evaluating, and interpreting a machine learning model for a classification task.