IRIS FLOWER CLASSIFICATION

Name: YASH BANSAL

Roll Number: 202401100300283

Course: CSE(AI)

Section: D

Class Roll No.: 63

INTRODUCTION

The iris flower, scientifically known as Iris, is a distinctive genus of flowering plants. Within this genus, there are three primary species: Iris setosa, Iris versicolor, and Iris virginica. These species exhibit variations in their physical characteristics, particularly in the measurements of their sepal length, sepal width, petal length, and petal width.

Objective:

The objective of this project is to develop a machine learning model capable of learning from the measurements of iris flowers and accurately classifying them into their respective species. The model's primary goal is to automate the classification process based on the distinct characteristics of each iris species.

Project Details:

- **Iris Species:** The dataset consists of iris flowers, specifically from the species setosa, versicolor, and virginica.
- **Key Measurements:** The essential characteristics used for classification include sepal length, sepal width, petal length, and petal width.
- **Machine Learning Model:** The project involves the creation and training of a machine learning model to accurately classify iris flowers based on their measurements.

This project's significance lies in its potential to streamline and automate the classification of iris species, which can have broader applications in botany, horticulture, and environmental monitoring.

Methodology

1. Import Required Libraries

- The first step is to import the necessary libraries. The libraries used in this code include:
    - **numpy**: For numerical operations and handling arrays.
    - **pandas**: For data manipulation and analysis.
    - **seaborn & matplotlib**: For data visualization (e.g., plotting heatmaps, pairplots).
    - **sklearn**: For machine learning tasks, including data preprocessing, model training, and evaluation.
    - **google.colab.files**: For uploading and downloading files in a Google Colab environment.

2. Upload & Load CSV File

- The code requests the user to upload the iris_data.csv file manually using the files.upload() function from the google.colab library.
- After uploading the CSV file, the code reads the dataset into a Pandas DataFrame using pd.read_csv() and prints the first 5 rows of the dataset for inspection.

3. Data Preprocessing & Cleaning

- **Missing Values**: The code checks for any missing values in the dataset using the isnull().sum() function. It prints the count of missing values in each column.
- **Dataset Information**: It prints the dataset's structure, including column names, data types, and any missing values, using df.info().
- **Unique Values in Target Column**: It checks the unique values in the target variable (species) to identify the different species in the dataset.
- **Extract Features and Target**: The features (input variables) are extracted by selecting all columns except the last one, and the target variable (species) is extracted by selecting the last column.
- **Label Encoding**: Since the target variable (species) contains categorical values (i.e., species names), the code uses **LabelEncoder** from sklearn to convert these categorical labels into numerical values. This is necessary for training machine learning models.

4. Correlation Matrix

- The correlation matrix is computed for the numeric features (sepal length, sepal width, petal length, and petal width) using the corr() method.
- The correlation matrix is visualized using a **heatmap** plotted by seaborn. The heatmap helps to identify the relationships between different features:
    - Correlation values close to 1 indicate a strong positive relationship.
    - Correlation values close to -1 indicate a strong negative relationship.
    - Correlation values close to 0 indicate no significant relationship.

5. Data Visualization

- The **pairplot** function from seaborn is used to visualize the relationships between the features in the dataset, with different species indicated by different colors. This helps in understanding the distribution and separability of the classes based on the features.

6. Train-Test Split

- The dataset is split into **training (80%)** and **testing (20%)** sets using the train_test_split function from sklearn.model_selection.
- The training set is used to train the model, while the testing set is used to evaluate its performance.
- The shapes of the training and testing sets are printed to confirm the split.

7. Model Training

- A **Random Forest Classifier** is initialized and trained on the training data. The model uses 100 estimators (trees) in the forest and is trained using the fit() method.
- The Random Forest algorithm is an ensemble learning method that works by constructing multiple decision trees and aggregating their results for classification.

8. Model Evaluation

- **Predictions**: After training the model, predictions are made on the testing data using the predict() method.
- **Accuracy**: The accuracy of the model is calculated by comparing the predicted values with the actual test labels. The accuracy score is computed using the accuracy_score() function from sklearn.metrics.

- **Classification Report**: A detailed classification report is generated using classification_report(), which includes the precision, recall, and F1-score for each class, along with the macro and weighted averages.

9. Save & Download Report

- The accuracy score and classification report are saved to a text file (classification_report.txt).
- The report file is made available for download using the files.download() function from google.colab.

CODE

```
# ================================
# 📌 Step 1: Import Required Libraries
# ================================
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
from google.colab import files

# ================================
# 📌 Step 2: Upload & Load CSV File
# ================================
print("📂 Please upload the 'iris_data.csv' file:")
uploaded = files.upload()  # Manually upload the CSV file

# Define filename (Make sure it matches the uploaded file)
filename = "iris_data.csv"

# Read the CSV file into a Pandas DataFrame
```

```
df = pd.read_csv(filename)

# Display the first 5 rows of the dataset
print("\n🔍 Preview of Dataset:")
print(df.head())

# ================================
# 📌 Step 3: Data Preprocessing & Cleaning
# ================================

# ✅ Check for Missing Values
print("\n❓ Checking for Missing Values:")
print(df.isnull().sum())  # Displays the count of missing values in each column

# ✅ Display Dataset Information (Column Names, Data Types, Missing Data)
print("\n📊 Dataset Information:")
print(df.info())

# ✅ Checking Unique Values in Target Column
print("\n🎯 Unique Classes in Target Column:")
print(df.iloc[:, -1].unique())

# ✅ Extract Features (X) & Target Column (y)
X = df.iloc[:, :-1]  # Select all columns except the last one (Features)
y = df.iloc[:, -1]   # Select the last column (Target variable - Species)
```

```
# ✅ Convert Target Labels to Numerical Values
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)  # Convert species names into numbers (Setosa → 0, Versicolor → 1, Virginica → 2)
print("\n🆔 Label Encoding Applied to Target Column")

# ================================
# 📌 Step 4: Correlation Matrix (Fixed)
# ================================

# ✅ Compute Correlation Matrix (Only for Numeric Features)
corr_matrix = df.iloc[:, :-1].corr()  # Exclude non-numeric columns

# ✅ Plot Correlation Heatmap
plt.figure(figsize=(8,6))  # Set figure size
sns.heatmap(corr_matrix, annot=True, cmap="coolwarm", fmt=".2f", linewidths=0.5)
plt.title("📊 Correlation Matrix of Features")
plt.show()

# ✅ Interpretation:
# - Correlation values range from -1 to 1.
# - Values close to +1 indicate strong positive correlation.
# - Values close to -1 indicate strong negative correlation.
# - Values near 0 indicate no significant correlation.
```

```
# ================================
# 📌 Step 5: Data Visualization
# ================================

# ✅ Pairplot - Relationship Between Features Colored by Species
df_encoded = df.copy()
df_encoded["Species"] = y  # Replace categorical values with encoded values
sns.pairplot(df_encoded, hue="Species", palette="husl")
plt.show()

# ================================
# 📌 Step 6: Train-Test Split
# ================================

# ✅ Split dataset into Training (80%) and Testing (20%) sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# ✅ Print Dataset Shapes
print(f"\n🧩 Training Set Shape: {X_train.shape}")
print(f"🧪 Testing Set Shape: {X_test.shape}")

# ================================
# 📌 Step 7: Model Training
# ================================
```

```
# ✅ Initialize and train a Random Forest Classifier
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# ================================
# 📌 Step 8: Model Evaluation
# ================================

# ✅ Make Predictions on Test Data
y_pred = model.predict(X_test)

# ✅ Compute Accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"\n✅ Model Accuracy: {accuracy:.2f}")

# ✅ Generate Detailed Classification Report
report = classification_report(y_test, y_pred)

# ✅ Print Classification Report
print("\n📜 Classification Report:\n", report)

# ================================
# 📌 Step 9: Save & Download Report
# ================================
```

```
# ✅ Save classification report to a text file
report_filename = "classification_report.txt"
with open(report_filename, "w") as f:
    f.write(f"Model Accuracy: {accuracy:.2f}\n\n")
    f.write("Classification Report:\n")
    f.write(report)

# ✅ Download the report file
files.download(report_filename)
print(f"\n📁 Report saved and ready to download: {report_filename}")
```

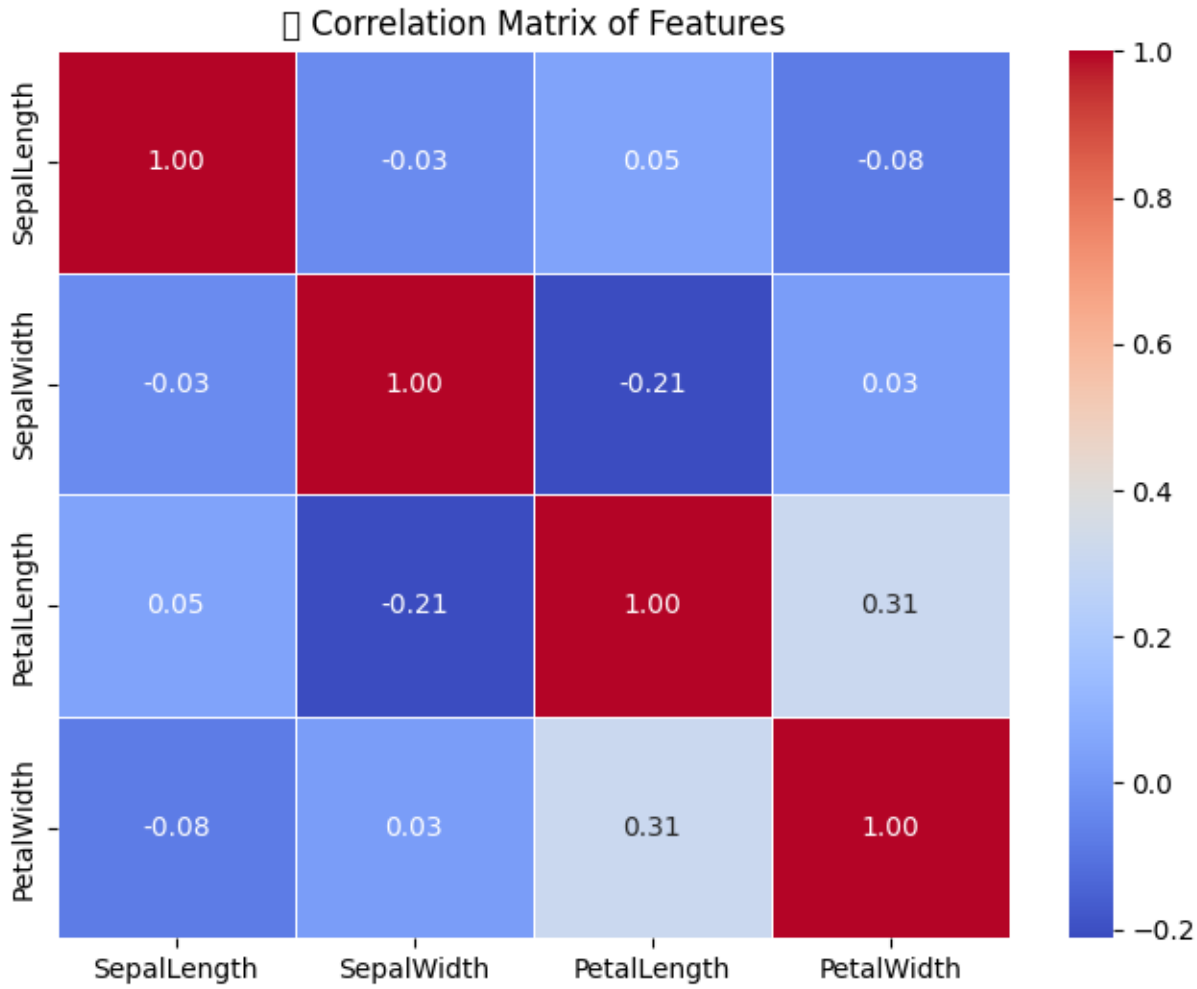
RESULT

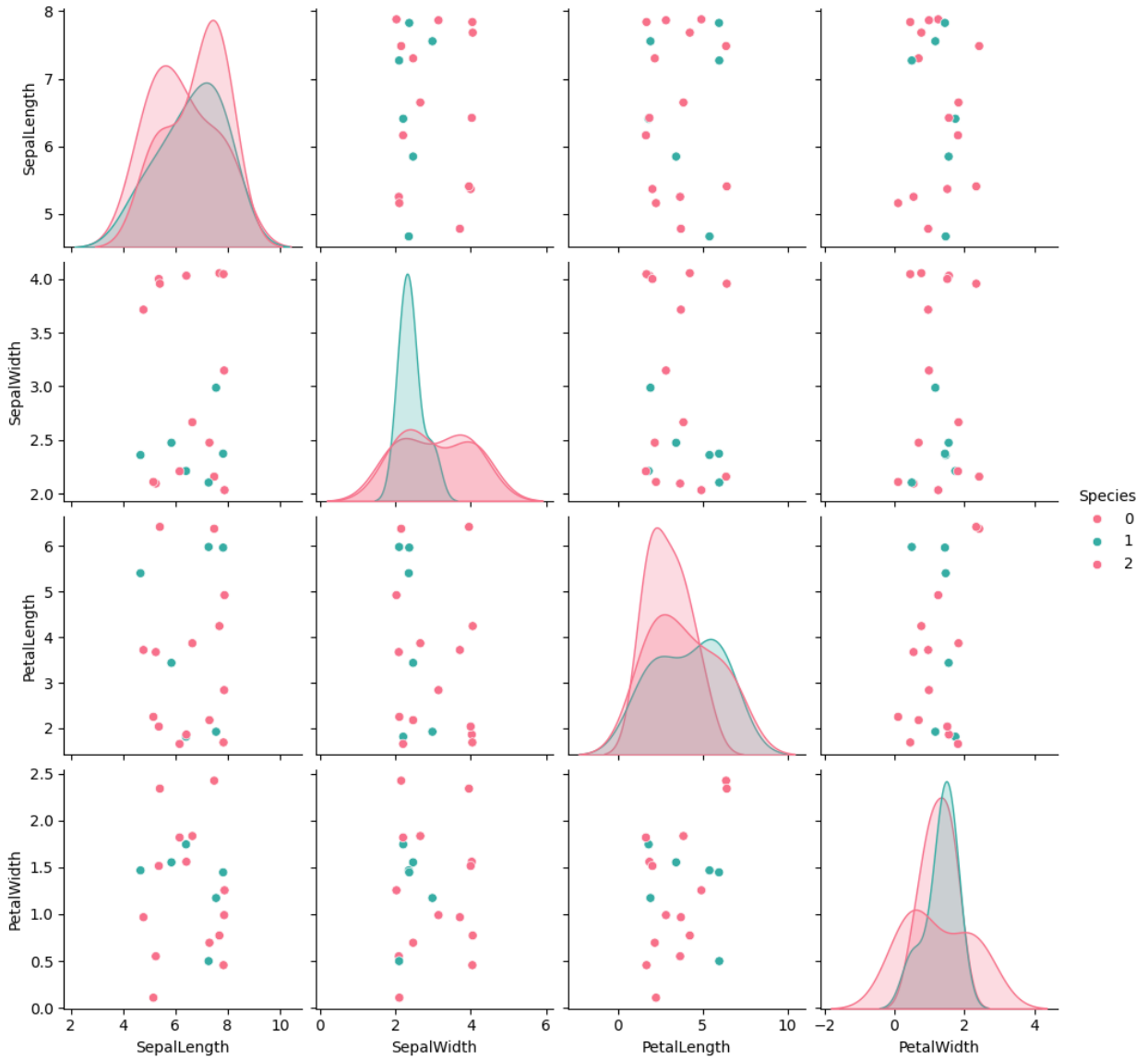
```
Please upload the 'iris_data.csv' file:
Choose Files iris_data.csv
• iris_data.csv(text/csv) - 1710 bytes, last modified: 3/11/2025 - 100% done
Saving iris_data.csv to iris_data.csv

🔍 Preview of Dataset:
   SepalLength  SepalWidth  PetalLength  PetalWidth     Species
0     7.303275    2.475025     2.176049    0.695003      Setosa
1     7.556928    2.987381     1.921585    1.172615  Versicolor
2     5.254016    2.093516     3.672564    0.550424   Virginica
3     6.409620    2.211042     1.812869    1.745372  Versicolor
4     7.684009    4.056479     4.244270    0.772148      Setosa

❓ Checking for Missing Values:
SepalLength    0
SepalWidth     0
PetalLength    0
PetalWidth     0
Species        0
dtype: int64

📊 Dataset Information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20 entries, 0 to 19
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   SepalLength  20 non-null     float64
 1   SepalWidth   20 non-null     float64
 2   PetalLength  20 non-null     float64
 3   PetalWidth   20 non-null     float64
 4   Species      20 non-null     object
dtypes: float64(4), object(1)
memory usage: 932.0+ bytes
```






```
Training Set Shape: (16, 4)
Testing Set Shape: (4, 4)

Model Accuracy: 0.25

Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         1
           1       0.00      0.00      0.00         2
           2       0.25      1.00      0.40         1

    accuracy                           0.25         4
   macro avg       0.08      0.33      0.13         4
weighted avg       0.06      0.25      0.10         4

/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted sam
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted sam
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted sam
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))

Report saved and ready to download: classification_report.txt
```

References

1. Fisher, R.A. (1936). *The use of multiple measurements in taxonomic problems*. Annals of Eugenics, 7(2), 179-188.
2. Breiman, L. (2001). *Random forests*. Machine Learning, 45(1), 5-32. DOI: 10.1023/A:1010933404324
3. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). *Scikit-learn: Machine learning in Python*. Journal of Machine Learning Research, 12, 2825-2830.
4. Waskom, M.L., Botvinnik, O., Ostblom, S., et al. (2020). *Seaborn: statistical data visualization*. Journal of Open Source Software, 5(51), 2296.