Report

Machine Learning Classification Report: Iris Flower Classification

Name: SHUBHAM KUMAR

Uni. Roll No. :202401100300243

2. Introduction

The Iris dataset is a well-known dataset in the field of machine learning, commonly used for classification problems. It contains **150 samples** of **three different species** of the iris flower (**Setosa, Versicolor, and Virginica**), with four features: **sepal length, sepal width, petal length, and petal width**.

This report presents a **machine learning model** that classifies iris flowers based on their features using a **Random Forest Classifier**. The implementation is done in Python using Google Colab and includes steps like **data preprocessing, visualization, model training, and evaluation**.

3. Methodology

The methodology followed in this project consists of the following steps:

Step 1: Importing Required Libraries

We begin by importing essential Python libraries for handling data, visualization, and machine learning.

Step 2: Data Upload & Loading

The dataset is manually uploaded into Google Colab and loaded into a Pandas DataFrame for further analysis.

Step 3: Data Preprocessing & Cleaning

- Checking for missing values.
- Displaying dataset information (column names, data types, missing data).
- Encoding categorical target labels into numerical values (0, 1, 2 for the three iris species).

Step 4: Data Visualization

- **Pairplot:** Helps visualize feature relationships based on species.
- **Correlation Heatmap:** Displays feature relationships to understand their dependencies.

Step 5: Splitting Dataset

- The dataset is divided into **80% training** and **20% testing** data using `train_test_split()`.

Step 6: Model Training

- A **Random Forest Classifier** with **100 estimators** is used for classification.

Step 7: Model Evaluation

- **Accuracy Score:** Measures the model's performance.
- **Classification Report:** Provides precision, recall, F1-score, and support for each class.

Step 8: Report Generation & Download

- The classification report is saved as a text file and made available for download.

# CODE

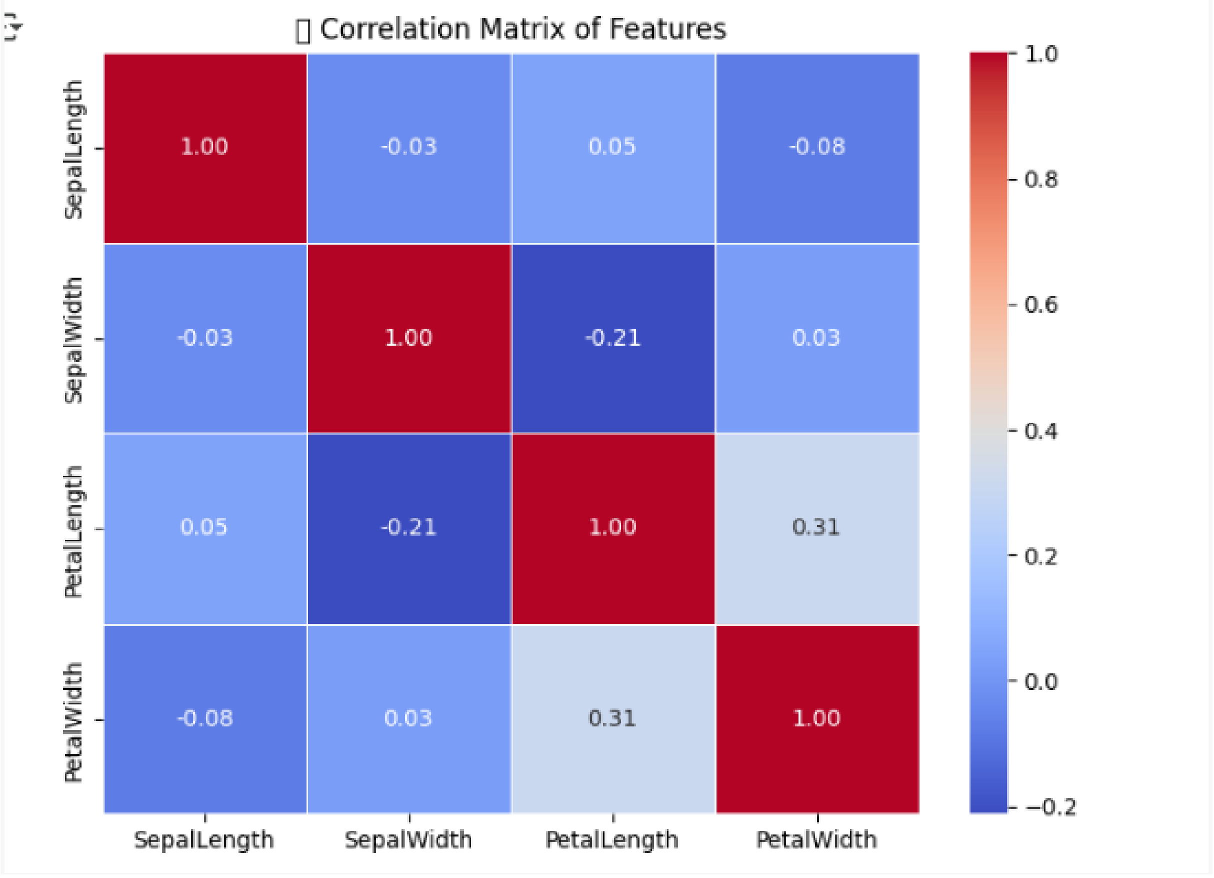
```
#  Step 1: Import Required Libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report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

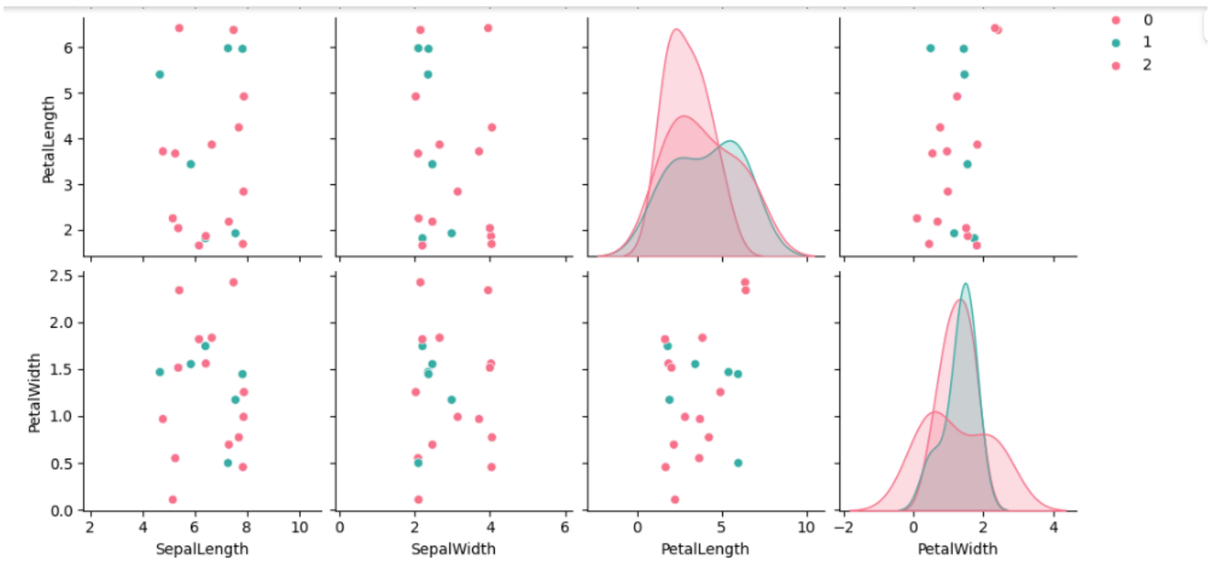
```
# Plot Correlation Heatmap
plt.figure(figsize=(8,6))  # Set figure size
sns.heatmap(corr_matrix, annot=True, cmap="coolwarm", fmt=".2f",
linewidths=0.5)
plt.title("□ Correlation Matrix of Features")
plt.show()

# 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}")
```

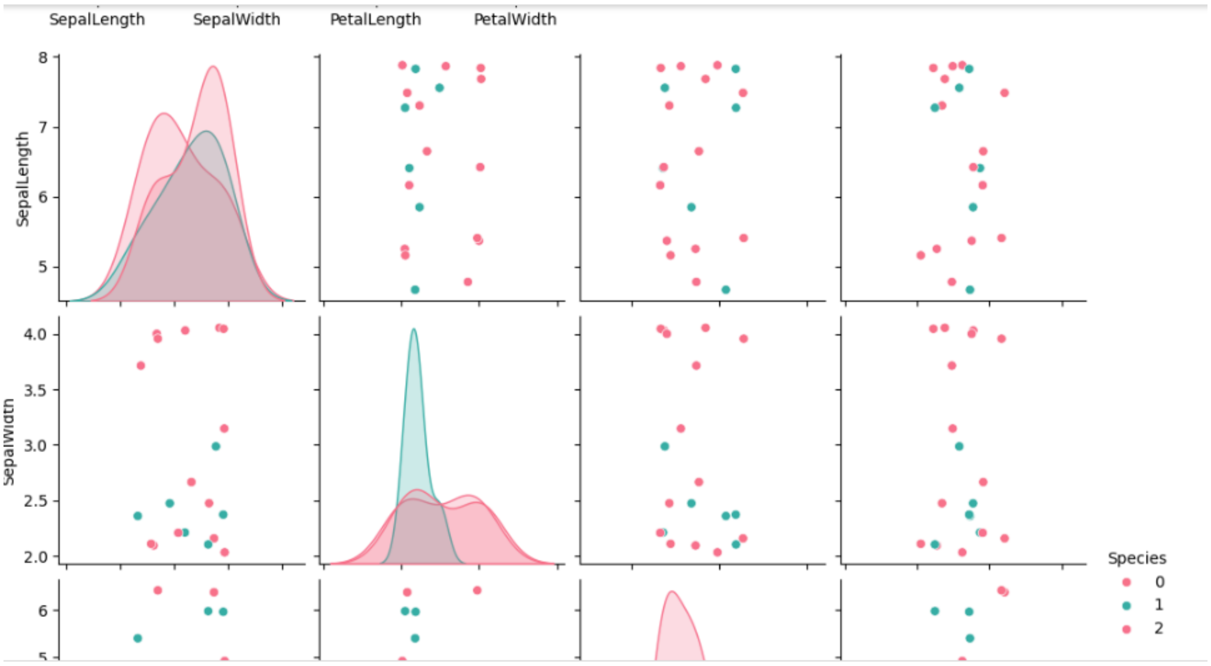
Screenshots :

```
    fig.canvas.print_figure(bytes_io, **kw)
```





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



REFERENCES

1. Fisher, R.A. (1936). *The Use of Multiple Measurements in Taxonomic Problems*. Annals of Eugenics, 7(2), 179–188.
2. Pedregosa, F., et al. (2011). *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research, 12, 2825–2830.
3. Breiman, L. (2001). *Random Forests*. Machine Learning, 45(1), 5-32.
4. Seaborn Documentation. (n.d.). Retrieved from: https://seaborn.pydata.org/
5. Scikit-learn Documentation. (n.d.). Retrieved from: https://scikit-learn.org/stable/