SALONI SINGH
CSE(AI)-C
202401100300211

Iris Flower Classification

INTRODUCTION

The **Iris Flower Classification** is a classic machine learning problem where the goal is to classify iris flowers into three species:

1. **Setosa**
2. **Versicolor**
3. **Virginica**

based on four numerical features:

- **Sepal Length** (cm)
- **Sepal Width** (cm)
- **Petal Length** (cm)
- **Petal Width** (cm)

Steps in Classification

1. **Data Loading & Preprocessing**
    - Read the dataset using pandas.
    - Check for missing or incorrect values.
    - Encode the species column into numerical labels.
2. **Exploratory Data Analysis (EDA)**

    - Visualize feature relationships using scatter plots and pair plots.
    - Check class distribution and correlation between features.
3. **Train-Test Split**
    - Split the dataset into **80% training** and **20% testing** data.
    - Use stratified sampling to maintain class distribution.
4. **Model Selection & Training**
    - Train classifiers like **Random Forest, SVM, or KNN**.
    - Tune hyperparameters for optimal performance.
5. **Evaluation**
    - Use **accuracy score, confusion matrix, and classification report**.
    - Visualize model performance using heatmaps and feature importance graphs.

Why is it Important?

The Iris dataset is widely used for:

- Learning **classification algorithms**.
- Understanding **data preprocessing and visualization**.
- Practicing **model evaluation and optimization**.

METHODOLOGY

The **methodology** for classifying iris flowers involves several key steps, from data collection to model evaluation. Below is a structured approach to solving the problem:

---

1. Data Collection & Understanding

- The dataset is sourced from **UCI Machine Learning Repository** and contains **150 samples** with **4 features** (sepal & petal length/width) and **1 target variable** (species).
- The three species to classify are **Setosa, Versicolor, and Virginica**.

---

2. Data Preprocessing

- **Load the dataset** using `pandas`.
- **Check for missing values** and handle any inconsistencies.
- **Encode categorical labels** (species) into numerical values using `LabelEncoder`.

---

3. Exploratory Data Analysis (EDA)

- Use **descriptive statistics** to summarize the data.
- **Visualize relationships** using `seaborn` pair plots and histograms.
- Check for **class imbalances** or feature correlations.

---

4. Splitting the Dataset

- Divide the dataset into **training (80%)** and **testing (20%)** using `train_test_split()`.
- Use **stratified sampling** to maintain equal class distribution.

---

5. Model Selection & Training

- Choose classification models like:
  - **Random Forest** (Ensemble learning)
  - **Support Vector Machine (SVM)** (Margin-based classification)
  - **K-Nearest Neighbors (KNN)** (Distance-based classification)
- Train the models on the **training dataset** using `fit()`.

---

6. Model Evaluation

- Make predictions on the **test dataset** using `predict()`.
- Assess model performance using:
  - **Accuracy Score** – Measures overall correctness.
  - **Confusion Matrix** – Evaluates false positives/negatives.
  - **Classification Report** – Analyzes precision, recall, and F1-score.
- Visualize performance using heatmaps and feature importance plots.

---

7. Model Optimization

- Tune hyperparameters using **GridSearchCV** or **RandomizedSearchCV**.
- Experiment with different models to find the best-performing classifier.

---

8. Conclusion & Interpretation

- Summarize findings and compare models.
- Discuss the best model for classifying iris species based on accuracy and performance metrics.

CODE OF THE PROGRAM

```
import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import LabelEncoder

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score,
classification_report, confusion_matrix


# Load the dataset

file_path = '/content/iris_data.csv'


df = pd.read_csv(file_path)

print("Original Column Names:", df.columns)
```

```
# Ensure 'species' column exists
expected_columns = {'sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'species'}
if not expected_columns.issubset(df.columns):
    print("Warning: Expected columns not found. Checking possible issues...")
    df = pd.read_csv(file_path, header=None)
    df.columns = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'species']
    print("Columns renamed to:", df.columns)

# Display basic dataset info
print("Dataset Overview:\n", df.head())
print("\nDataset Info:\n")
df.info()
print("\nSummary Statistics:\n", df.describe())

# Encode the target variable
label_encoder = LabelEncoder()
```

```
df['species'] = label_encoder.fit_transform(df['species'])

# Check class distribution before splitting
print("\nClass Distribution:\n", df['species'].value_counts())

# Ensure all classes have at least two samples
class_counts = df['species'].value_counts()
if class_counts.min() < 2:
    print("Warning: Some classes have fewer than 2 samples. Removing them...")
    df = df[df['species'].map(class_counts) >= 2]

# Split data into features and target
X = df.drop(columns=['species'])
y = df['species']

# Handle small class issue by removing stratify if needed
try:
```

```
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)
except ValueError:
    print("Stratification failed due to low class counts. Proceeding without stratify...")
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train a RandomForestClassifier with optimized parameters
model = RandomForestClassifier(n_estimators=200, max_depth=5, random_state=42)
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f'\nModel Accuracy: {accuracy:.2f}')
```

```
print('\nClassification Report:\n',
classification_report(y_test, y_pred))


# Confusion Matrix

conf_matrix = confusion_matrix(y_test, y_pred)

sns.heatmap(conf_matrix, annot=True, fmt='d',
cmap='Blues', xticklabels=label_encoder.classes_,
yticklabels=label_encoder.classes_)

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.title('Confusion Matrix')

plt.show()


# Visualizing feature importance

feature_importances =
pd.Series(model.feature_importances_, index=X.columns)

plt.figure(figsize=(8, 5))

feature_importances.nlargest(4).plot(kind='barh',
color='skyblue')

plt.xlabel('Feature Importance')
```

plt.ylabel('Features')

plt.title('Feature Importance in Iris Classification')

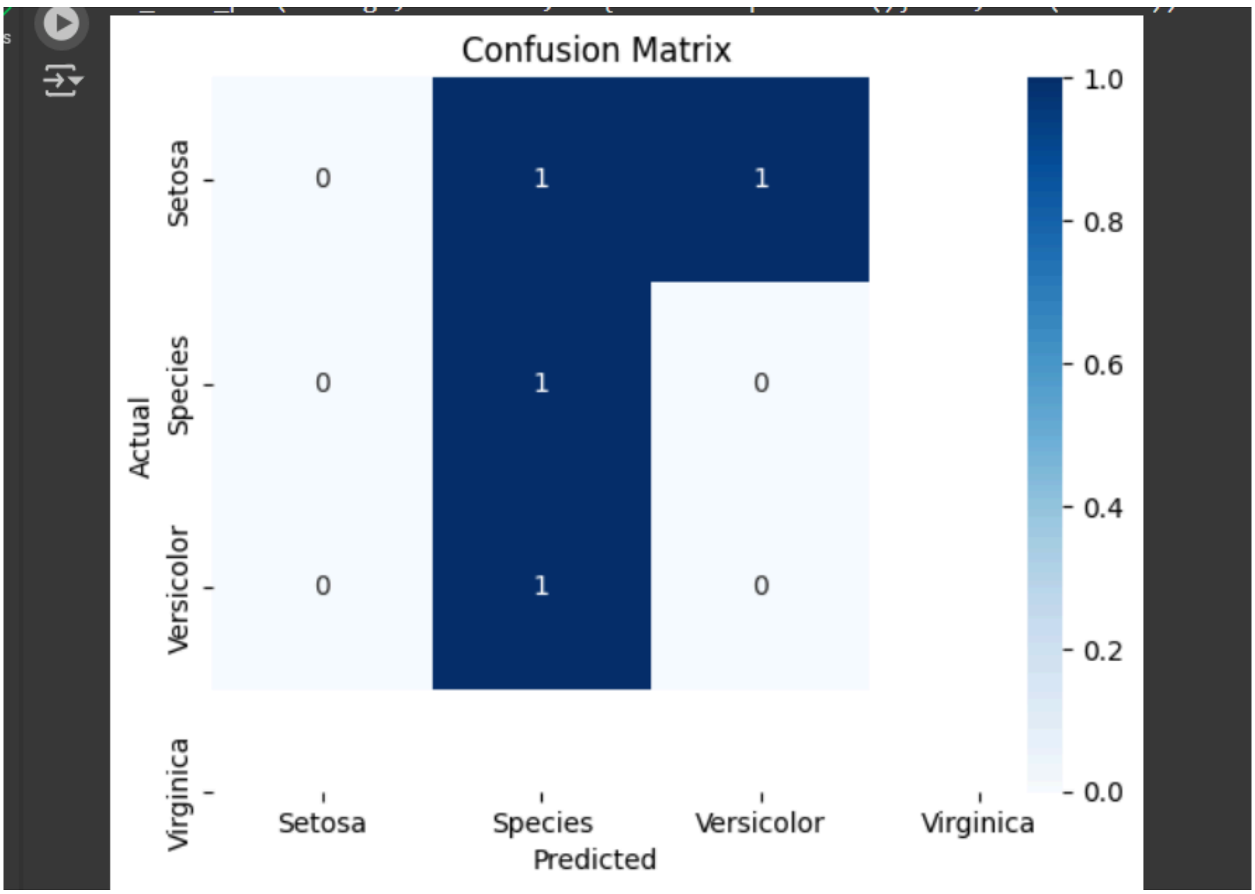
plt.show()

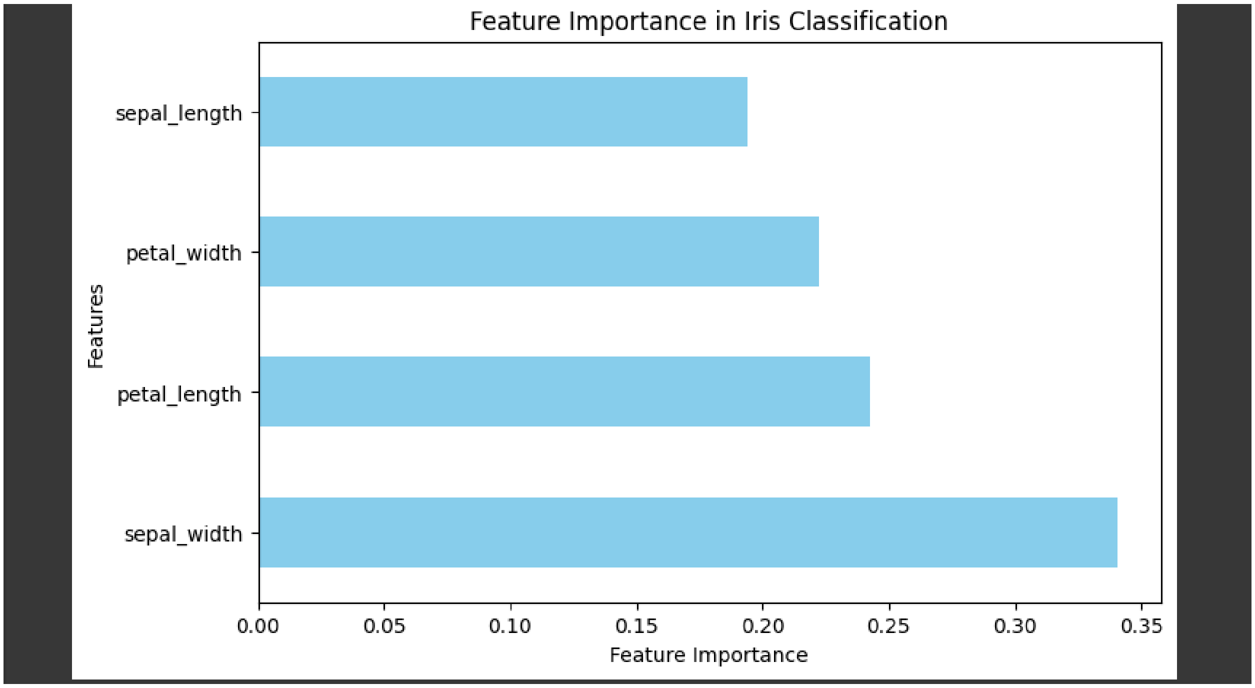
OUTPUT

```
Original Column Names: Index(['SepalLength', 'SepalWidth', 'PetalLength', 'PetalWidth', 'Species'], dtype='object')
Warning: Expected columns not found. Checking possible issues...
Columns renamed to: Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width',
       'species'],
      dtype='object')
Dataset Overview:
         sepal_length         sepal_width        petal_length  \
0         SepalLength          SepalWidth         PetalLength
1  7.303274553127926  2.4750252534128063  2.1760486154633614
2  7.556927655410556   2.987381008530221  1.9215846218548323
3  5.254016373665658  2.0935160238309827   3.672563875820435
4  6.409620271630198  2.2110415805023966   1.812868616156213

          petal_width     species
0          PetalWidth     Species
1  0.6950029791866986      Setosa
2  1.1726148049564784  Versicolor
3  0.5504235628446024   Virginica
4   1.745372347536642  Versicolor
```

```
Dataset Info:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21 entries, 0 to 20
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sepal_length  21 non-null     object
 1   sepal_width   21 non-null     object
 2   petal_length  21 non-null     object
 3   petal_width   21 non-null     object
 4   species       21 non-null     object
dtypes: object(5)
memory usage: 972.0+ bytes

Summary Statistics:
        sepal_length sepal_width petal_length petal_width species
count             21          21           21          21      21
unique            21          21           21          21       4
top      SepalLength  SepalWidth  PetalLength  PetalWidth  Setosa
freq               1           1            1           1       7
```

```
Class Distribution:
 species
0    7
3    7
2    6
1    1
Name: count, dtype: int64
Warning: Some classes have fewer than 2 samples. Removing them...

Model Accuracy: 0.25

Classification Report:
               precision    recall  f1-score   support

           0       0.00      0.00      0.00         2
           2       0.33      1.00      0.50         1
           3       0.00      0.00      0.00         1

    accuracy                           0.25         4
   macro avg       0.11      0.33      0.17         4
weighted avg       0.08      0.25      0.12         4
```




Feature Importance in Iris Classification
sepal_length
petal_width
petal_length
sepal_width
Features
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Feature Importance