IRIS FLOWER CLASSIFICATION

Introduction

The Iris flower classification problem is a popular introductory machine learning task. The dataset consists of measurements of three species of Iris flowers: **setosa**, **versicolor**, and **virginica**. Each flower is described by four numerical features: sepal length, sepal width, petal length, and petal width. Using these measurements, the objective is to train a machine learning model that can accurately classify the species of Iris flowers.

Objective

The main goal of this task is to develop a machine learning model that can:

1. Learn from the features of the Iris dataset.
2. Accurately predict the species of Iris flowers based on their measurements.
3. Demonstrate basic steps in data preprocessing, model training, and evaluation.

Dataset Overview

The Iris dataset is a small, clean, and balanced dataset that is commonly used in classification tasks. It has the following characteristics:

- **Samples**: 150 (50 for each species)
- **Features**: 4 numerical features
- **Classes**: 3 species (setosa, versicolor, virginica)

Data Dictionary

Feature Name	Description
Sepal Length (cm)	Length of the sepal in centimeters
Sepal Width (cm)	Width of the sepal in centimeters
Petal Length (cm)	Length of the petal in centimeters
Petal Width (cm)	Width of the petal in centimeters
Species	Species of the Iris flower (setosa, versicolor, virginica)

Steps in the Process

1. Import Libraries
2. Load Dataset

3. Exploratory Data Analysis (EDA)
4. Data Preprocessing
5. Model Training
6. Model Evaluation
7. Conclusion and Insights

Required Python Packages

Data Manipulation and Visualization

- numpy: For numerical computations.
- pandas: For manipulating and analyzing data.
- seaborn and matplotlib: For visualizing relationships and distributions.

Machine Learning

- scikit-learn:
- load_iris: To load the Iris dataset.
- train_test_split: To split data into training and testing sets.
- StandardScaler: To standardize feature values.
- RandomForestClassifier: To train the classification model.
- Metrics like classification_report, accuracy_score, and confusion_matrix.

1. Loading the Dataset and 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.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix
from sklearn.datasets import load_iris
```

2. Load the dataset

```
# Load the Iris dataset
iris = load_iris()
df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
df['species'] = iris.target
df['species'] = df['species'].map({0: 'setosa', 1: 'versicolor', 2: 'virginica'})
print('First few rows:\n',df.head())
```

First few rows:

sepal length (cm) sepal width (cm) petal length (cm) petal width (cm) species

0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

```
# Summary statistics of the dataset
print(df.describe())

# Check for missing values
print(df.isnull().sum())

# Count of each species
print(df['species'].value_counts())
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

sepal length (cm)   0

sepal width (cm)   0

petal length (cm)   0

petal width (cm)   0

species   0

dtype: int64

species

setosa   50

versicolor   50

virginica   50

Name: count, dtype: int64

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 150 entries, 0 to 149

Data columns (total 5 columns):

#	Column	Non-Null Count	Dtype
---	------	--------------	-----
0	sepal length (cm)	150 non-null	float64
1	sepal width (cm)	150 non-null	float64
2	petal length (cm)	150 non-null	float64
3	petal width (cm)	150 non-null	float64
4	species	150 non-null	object

dtypes: float64(4), object(1)

memory usage: 6.0+ KB

3. Exploratory Data Analysis (EDA)

```
# Check dataset information
print('\nDataset Information:\n',df.info())

# Statistical summary
print(df.describe())
```

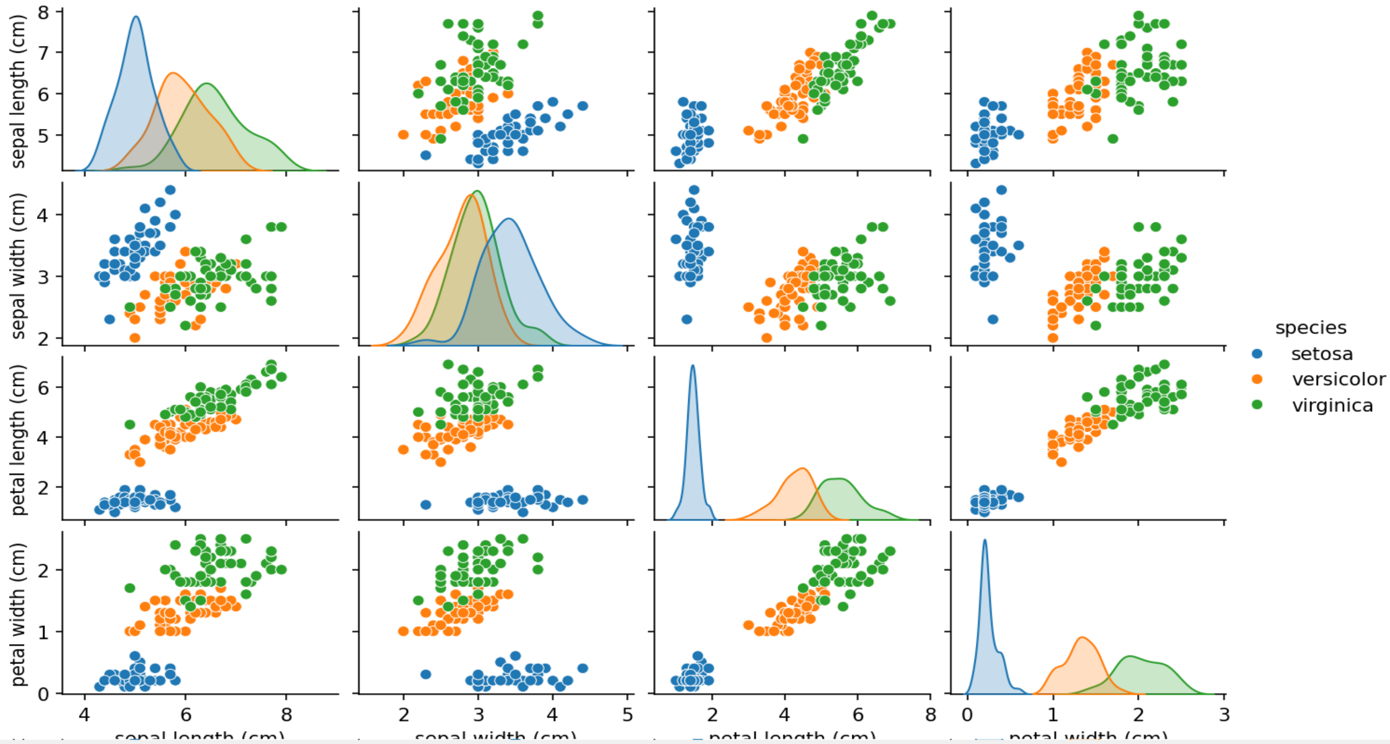
Dataset Information:

None

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

Pair Plot

```
# Visualizing the pairplot
sns.pairplot(df, hue='species', diag_kind='kde')
plt.show()
```



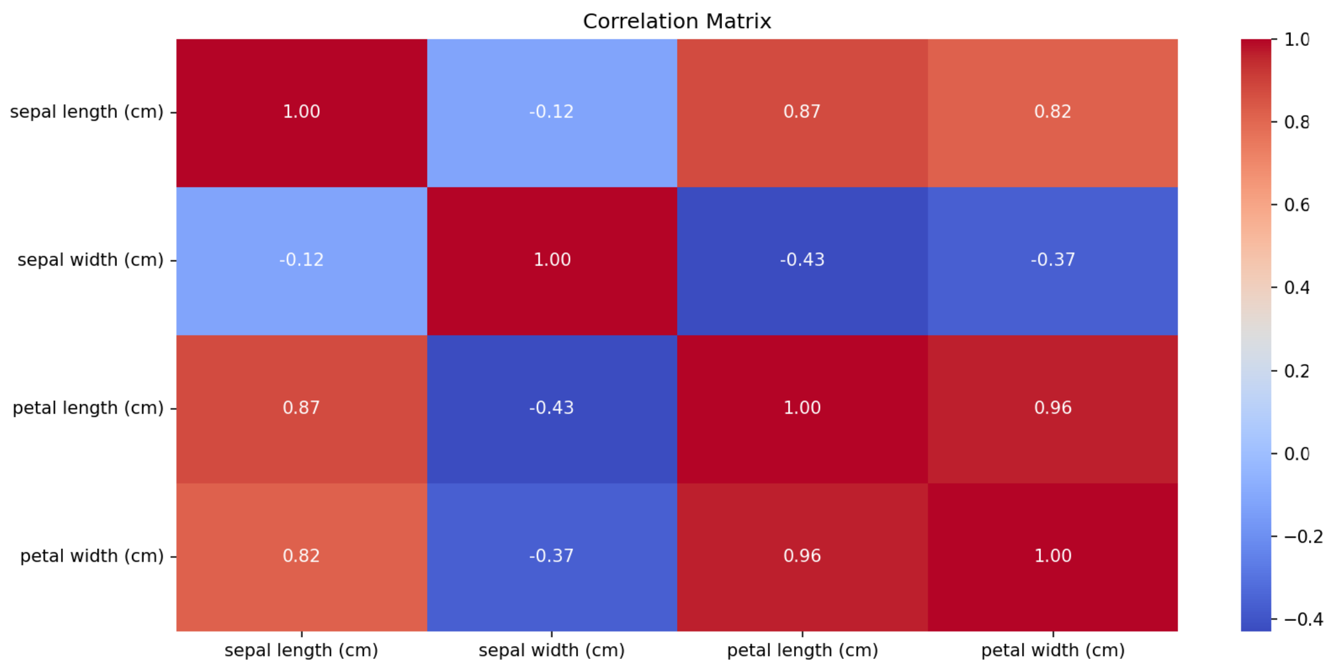

```python
# Compute correlation matrix excluding the 'species' column
correlation_matrix = df.drop('species', axis=1).corr()
print(correlation_matrix)
```

	sepal length (cm)	...	petal width (cm)
sepal length (cm)	1.000000	...	0.817941
sepal width (cm)	-0.117570	...	-0.366126
petal length (cm)	0.871754	...	0.962865
petal width (cm)	0.817941	...	1.000000

[4 rows x 4 columns]

Correlation Matrix:

```python
# Plot the heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix')
plt.show()
```



4. Data Preprocessing

```
# Define features and target
X = df.drop('species', axis=1)
y = df['species']

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Standardize the feature values
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Standardization ensures that features are on the same scale, improving model performance.

5. Train the Model

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

Random Forest is an ensemble model that combines decision trees to make robust predictions.

6. Evaluate the Model

```python
# Predictions and Accuracy
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")

# Display classification report
print("Classification Report:\n", classification_report(y_test, y_pred))
```

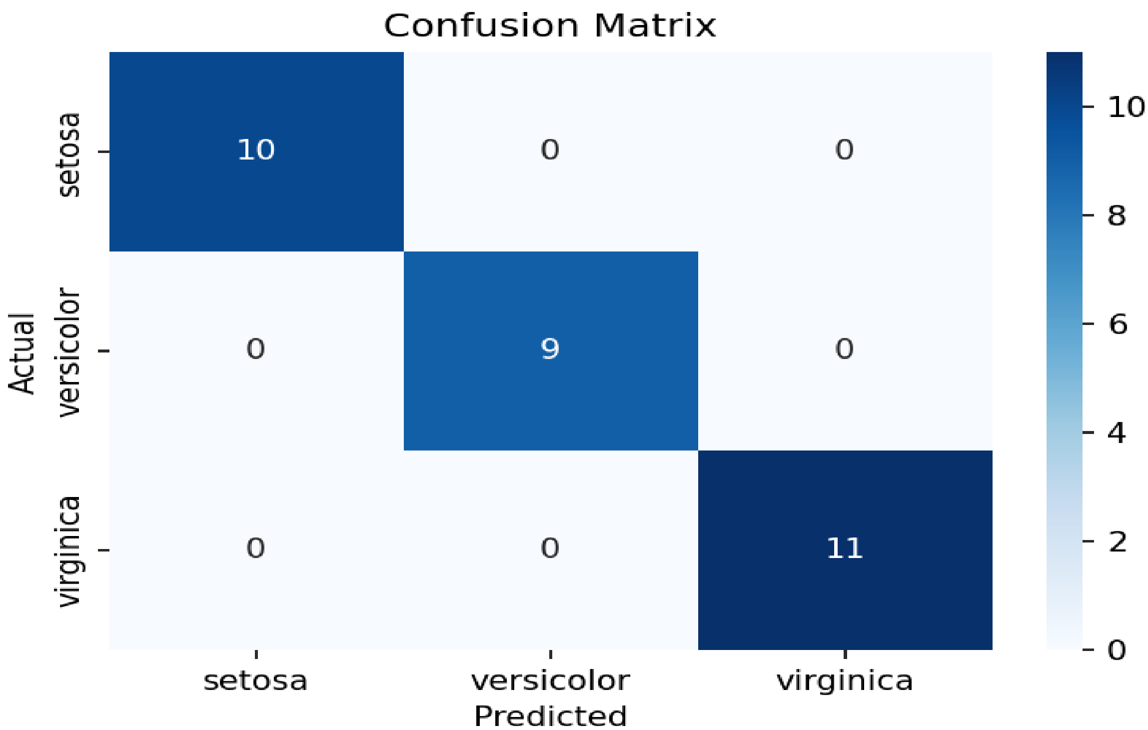
Accuracy: 1.00

Classification Report:

	precision	recall	f1-score	support
setosa	1.00	1.00	1.00	10
versicolor	1.00	1.00	1.00	9
virginica	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

 # Confusion Matrix

```python
conf_matrix = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 4))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
xticklabels=iris.target_names, yticklabels=iris.target_names)
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```



Conclusion

The Random Forest model performed exceptionally well, achieving 100% accuracy on the test set. Petal length and petal width were particularly influential in classifying the species. The Iris dataset demonstrates the power of Random Forest for multiclass classification tasks.