Iris Flower Classification Using Machine Learning

Data Science Project

Tools

- Python
- Pandas , NumPy
- Seaborn & Matplotlib
- Scikit-learn
- Jupyter Notebook

By : Komal Digwal

Project Overview

The **Iris Flower Classification** project focuses on building a machine learning model to classify Iris flowers into one of three species — **Setosa**, **Versicolor**, or **Virginica** — based on their **sepal and petal measurements**.
This dataset is a classic and widely used dataset for **introductory classification problems**, ideal for beginners learning supervised machine learning.

📌 Objectives:

- Understand the relationship between flower measurements and species
- Perform data analysis and visualization
- Train a classification model using labeled data
- Predict the species of an Iris flower based on its features

☐ Key Concepts Covered:

- Data Preprocessing
- Exploratory Data Analysis (EDA)
- Classification using ML Algorithms (e.g., Logistic Regression, KNN)
- Model Evaluation using Accuracy and Confusion Matrix

Dataset Description

The **Iris dataset** is one of the most famous datasets in pattern recognition and classification. It contains **150 samples** from three species of Iris flowers:

- Setosa
- Versicolor
- Virginica

Each sample includes **four numerical features** (measurements in centimeters) :

Column Name	Description
SepalLengthCm	Length of the sepal
SepalWidthCm	Width of the sepal
PetalLengthCm	Length of the petal
PetalWidthCm	Width of the petal
Species	The class/target label (flower species)

Dataset Info :

- **Total Samples :** 150
- **Features :** 4 numeric features
- **Target Variable :** Species (3 categories)
- **Missing Values :** None

Codes

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
import os
```

Load the Dataset

```
import kagglehub
path = kagglehub.dataset_download("arshid/iris-flower-dataset")
print("Path to dataset files:", path)
print(os.listdir(path))
```

Output

```
Path to dataset files: C:\Users\Admin\.cache\kagglehub\datasets\arshid\iris-flower-dataset\versions\1
['IRIS.csv']
```

Read

```
df = pd.read_csv("C:/Users/Admin/.cache/kagglehub/datasets/arshid/iris-flower-dataset/versions/1/IRIS.csv")
df.head()
```

Data Info and Clean-Up

```
print("Dataset Information:")
print(df.info())
```

Output's

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

```
Dataset Information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   sepal_length   150 non-null    float64
 1   sepal_width    150 non-null    float64
 2   petal_length   150 non-null    float64
 3   petal_width    150 non-null    float64
 4   species        150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
None
```

Summary Statistics

```
print("\nSummary Statistics:")
print(df.describe())
```

Missing Values

```
print("\nMissing Values:")
print(df.isnull().sum())
```

Output's

```
Summary Statistics:
       sepal_length  sepal_width  petal_length  petal_width
count    150.000000   150.000000    150.000000   150.000000
mean       5.843333     3.054000      3.758667     1.198667
std        0.828066     0.433594      1.764420     0.763161
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
```

```
Missing Values:
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

Column Names

```
print("\nColumn Names:")
print(df.columns)
```

Output

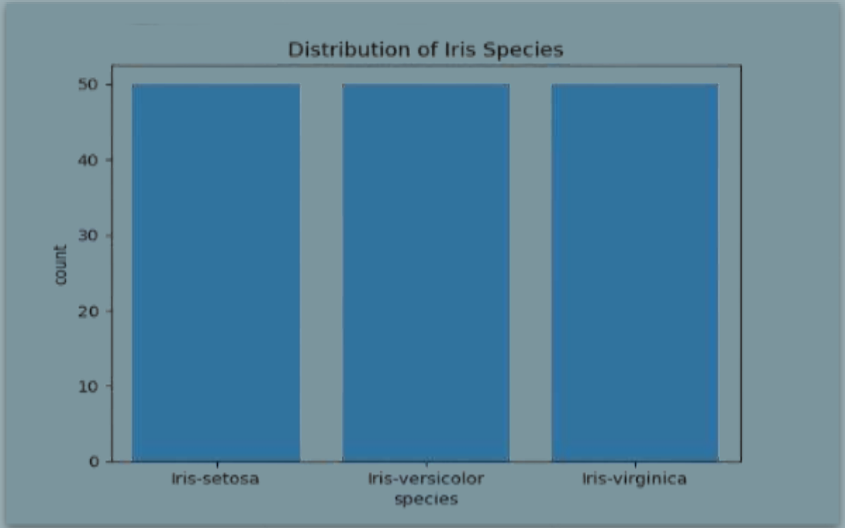
```
Column Names:
Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'species'],
dtype='object')
```

EDA – Visualizations

Countplot for Species Distribution

```
sns.countplot(x='species', data=df)
plt.title('Distribution of Iris Species’)
plt.show()
```

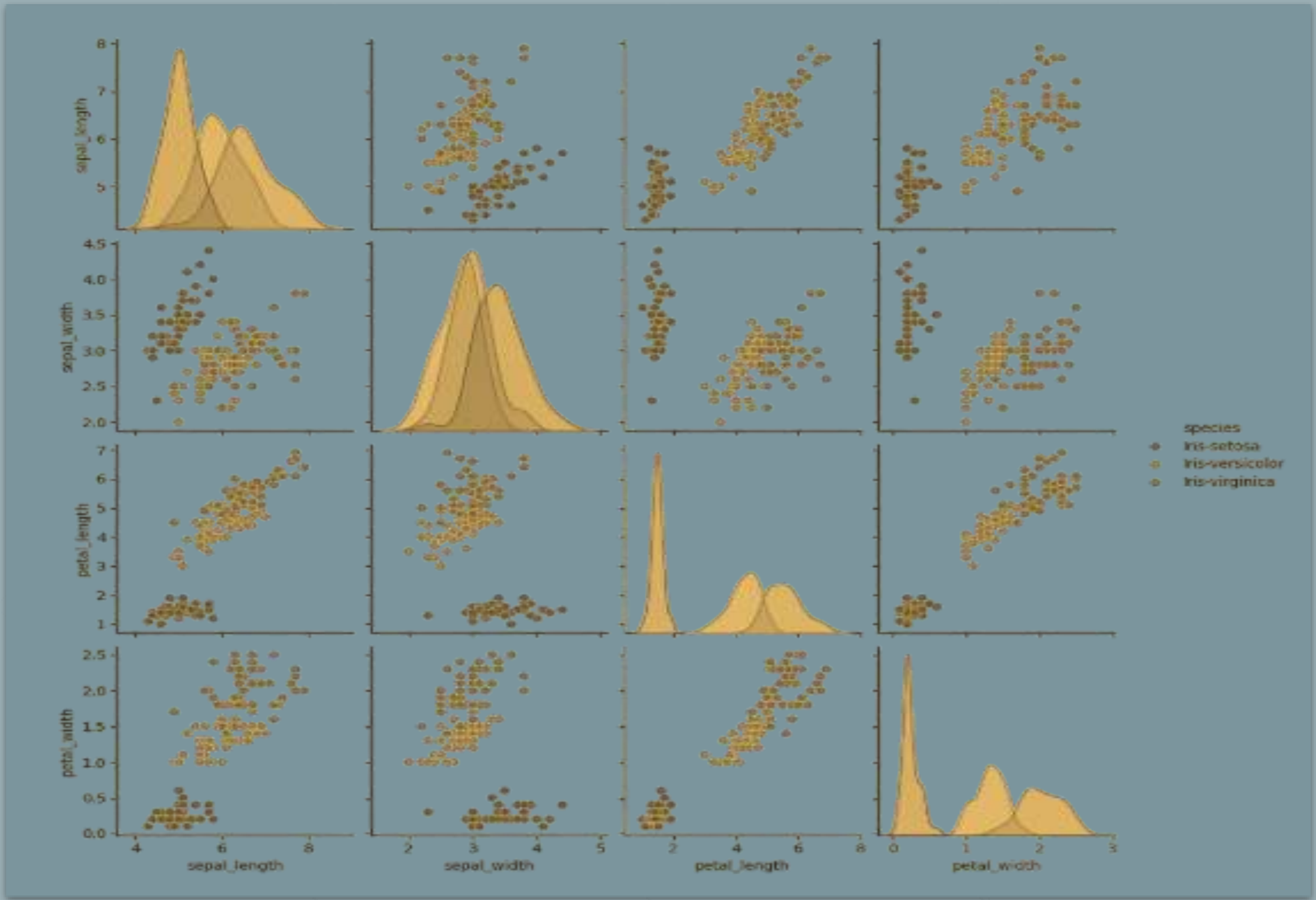
Output



Pairplot for Feature Relationships

```
sns.pairplot(df, hue='species')
plt.show()
```

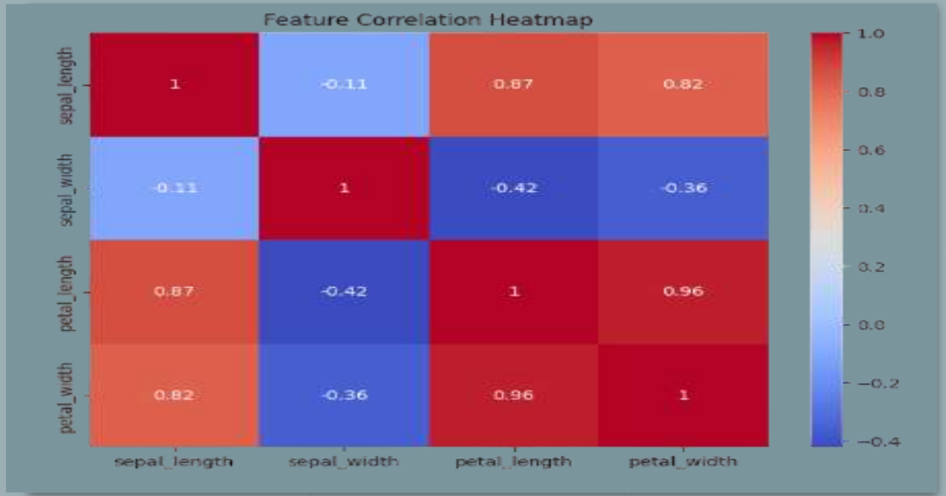
Output



Correlation Heatmap

```
plt.figure(figsize=(8, 6))
sns.heatmap(df.corr(numeric_only=True), annot=True, cmap='coolwarm’)
plt.title("Feature Correlation Heatmap")
plt.show()
```

Output



Data Preparation

```
X = df.drop('species', axis=1)
y = df['species’]
le = LabelEncoder()
y_encoded = le.fit_transform(y)
print("Unique Classes:", le.classes_)
```

Output

```
Unique Classes: ['Iris-setosa' 'Iris-versicolor' 'Iris-virginica']
```

Train-Test Split

```
X_train, X_test, y_train, y_test = train_test_split(X, y_encoded, test_size=0.2, random_state=42)
```

Model Training

Random Forest Classifier

```
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
y_pred_rf = rf_model.predict(X_test)
```

Logistic Regression

```
lr_model = LogisticRegression(max_iter=200)
lr_model.fit(X_train, y_train)
y_pred_lr = lr_model.predict(X_test)
```

Model Evaluation

Random Forest Evaluation

```
print("Random Forest Classifier Results:")
print("Accuracy:", accuracy_score(y_test, y_pred_rf))
print("\nClassification Report:")
print(classification_report(y_test, y_pred_rf))
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred_rf))
```

Output

```
Random Forest Classifier Results:
Accuracy: 1.0

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

Confusion Matrix:
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```

Model Comparison

```
print("Model Comparison:")
print("Random Forest Accuracy:", accuracy_score(y_test, y_pred_rf))
print("Logistic Regression Accuracy:", accuracy_score(y_test, y_pred_lr))
```

Output

```
Unique Classes: ['Iris-setosa' 'Iris-versicolor' 'Iris-virginica']
```

Train-Test Split

```
X_train, X_test, y_train, y_test = train_test_split(X, y_encoded, test_size=0.2, random_state=42)
```

Model Training

Random Forest Classifier

```
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
y_pred_rf = rf_model.predict(X_test)
```

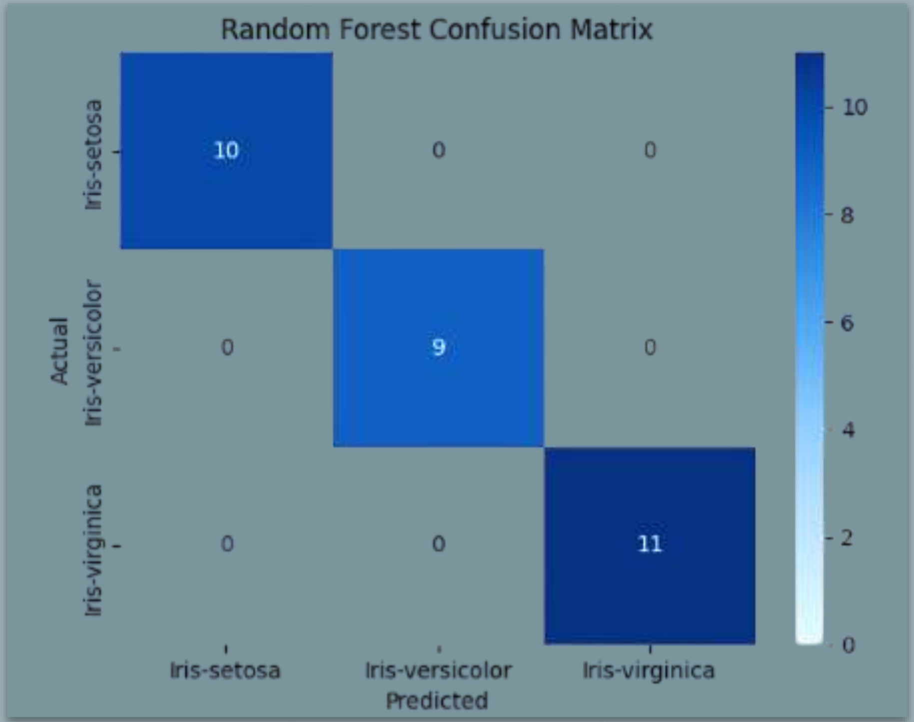
Output

```
Model Comparison:
Random Forest Accuracy: 1.0
Logistic Regression Accuracy: 1.0
```

Visualization

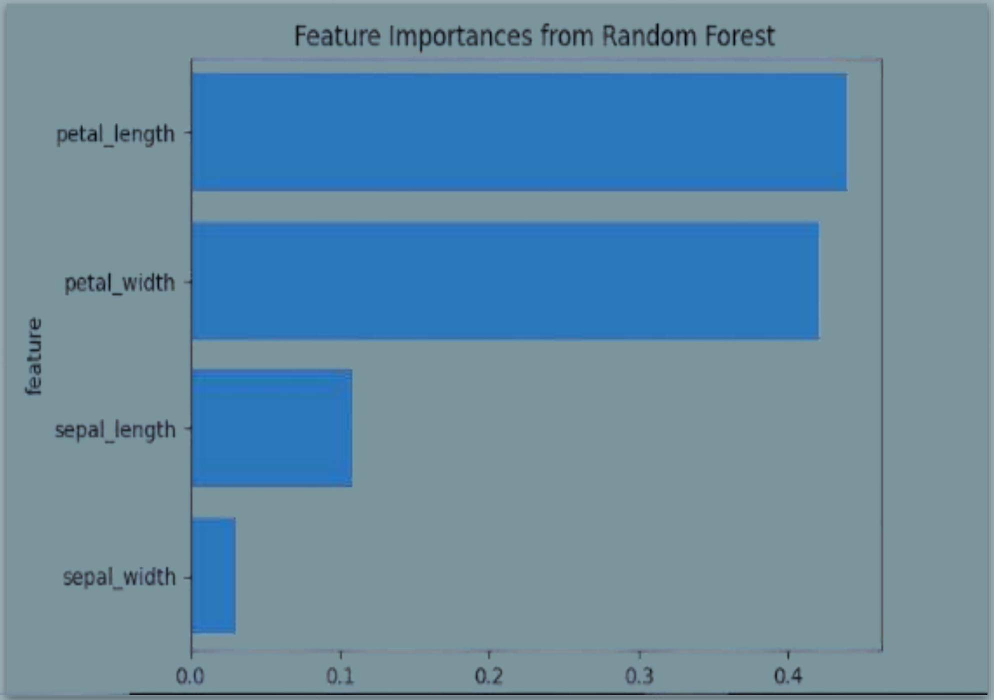
```
cm = confusion_matrix(y_test, y_pred_rf)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=le.classes_, yticklabels=le.classes_)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Random Forest Confusion Matrix')
plt.show()
```

Output



```python
feature_importances = pd.DataFrame({
    'feature': X.columns,
    'importance': rf_model.feature_importances_}).sort_values(by='importance', ascending=False)
print(feature_importances)
sns.barplot(x='importance', y='feature', data=feature_importances)
plt.title("Feature Importances from Random Forest")
plt.show()
```

Output




```
        feature  importance
2  petal_length    0.439994
3   petal_width    0.421522
0  sepal_length    0.108098
1   sepal_width    0.030387
```