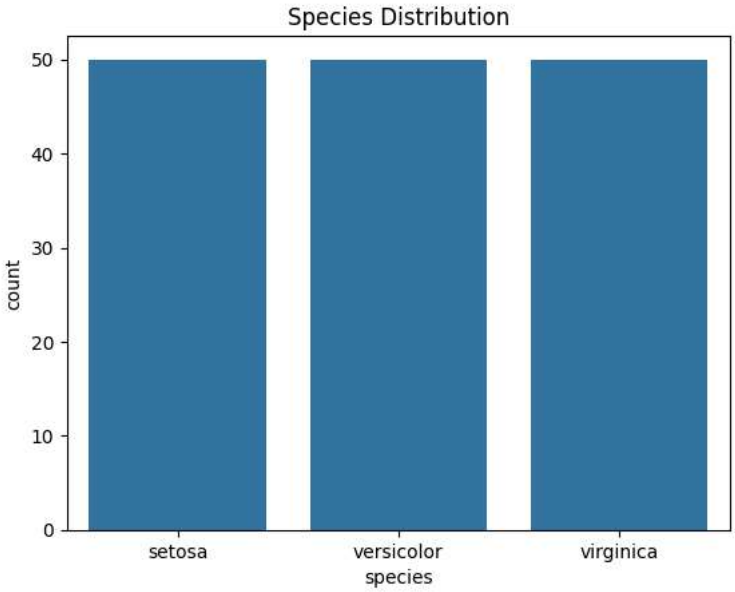
```
#  Iris Flower Classifiction


# Importing  libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report


# Load Iris dataset
iris = load_iris()
data = pd.DataFrame(data=iris.data, columns=iris.feature_names)
data['species'] = iris.target
data['species'] = data['species'].map({0: 'setosa', 1: 'versicolor', 2: 'virginica'})
print(data.head())  # Check first 5 rows
print(data.info())  # Check data types and missing values
```

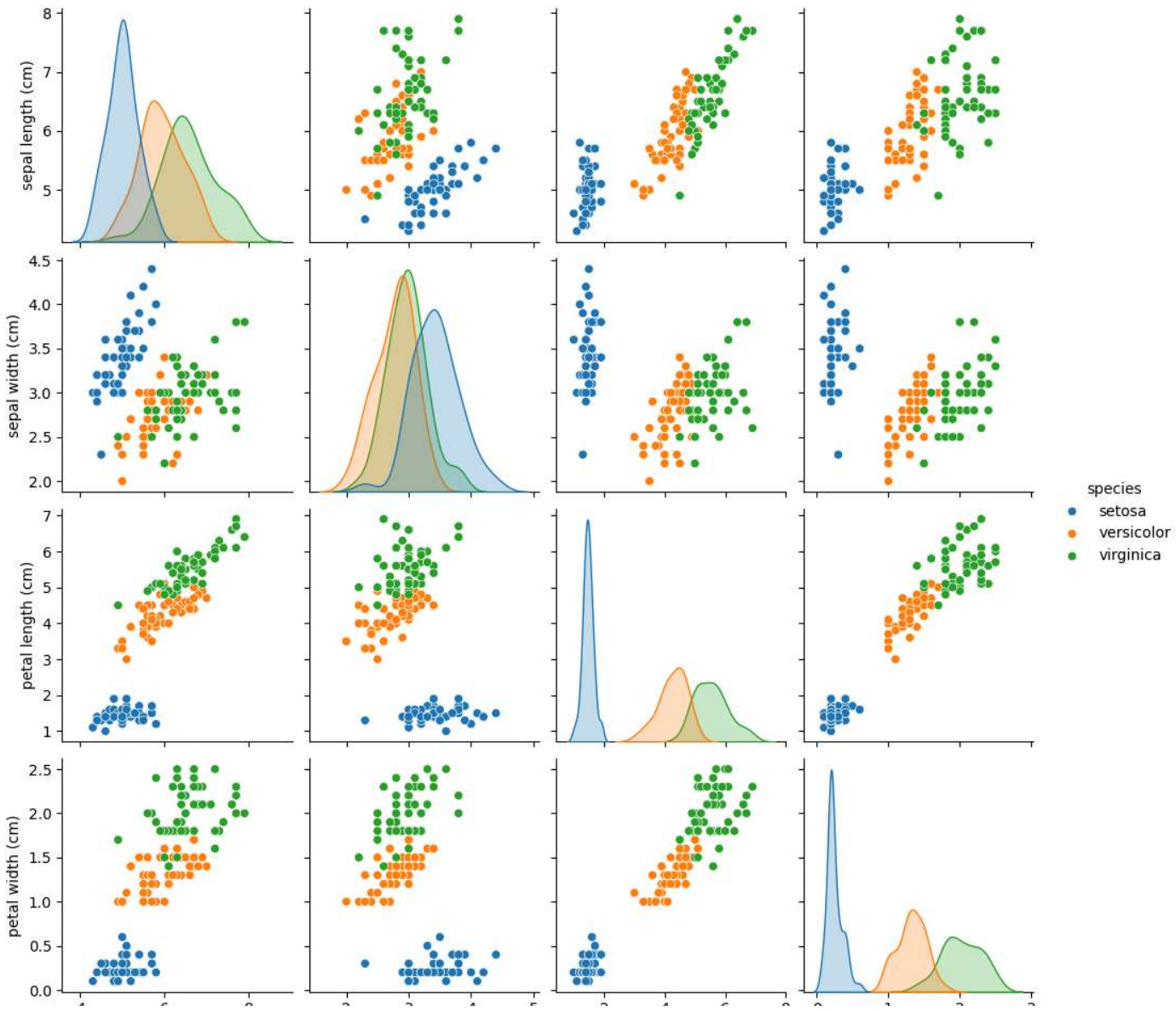
```
    sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2

  species
0  setosa
1  setosa
2  setosa
3  setosa
4  setosa
<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
None
```

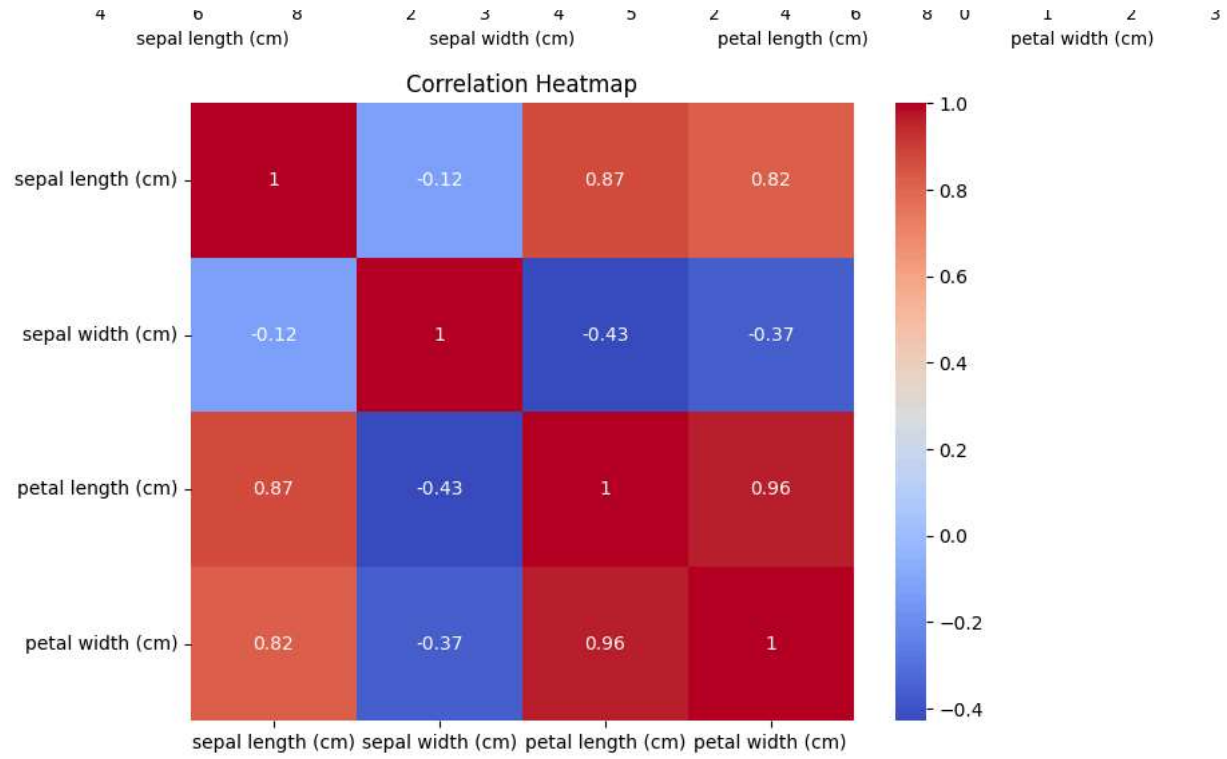
```
# Step 1: Libraries
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris

# Step 2: Load data as DataFrame
iris_data = load_iris()
data = pd.DataFrame(iris_data.data, columns=iris_data.feature_names)
data['species'] = iris_data.target_names[iris_data.target]

# Step 3: Check for missing values
print("Missing values:")
print(data.isnull().sum())

# Step 4: Species distribution
sns.countplot(x='species', data=data)  # ✅ data use karo, iris nahi
plt.title('Species Distribution')
plt.show()

# Step 5: Pair plot
sns.pairplot(data, hue='species')  # ✅ no extra space
plt.show()

# ✅ Sahi Tarika
plt.figure(figsize=(8, 6))
sns.heatmap(data.drop('species', axis=1).corr(), annot=True, cmap='coolwarm')
```

```
plt.title('Correlation Heatmap')
plt.show()
```

```
Missing values:
sepal length (cm)    0
sepal width (cm)     0
petal length (cm)    0
petal width (cm)     0
species              0
dtype: int64
```








```
# Step : Load data
iris = load_iris()
data = pd.DataFrame(iris.data, columns=iris.feature_names)
data['species'] = iris.target_names[iris.target]

# Step : Features and target
X = data.drop('species', axis=1)  # Features
y = data['species']               # Target

# Step : Scale features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Step : Split data
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

print(" Sab kuch chal gaya! Data split ho chuka hai.")
print("X_train shape:", X_train.shape)
print("y_test shape:", y_test.shape)
```

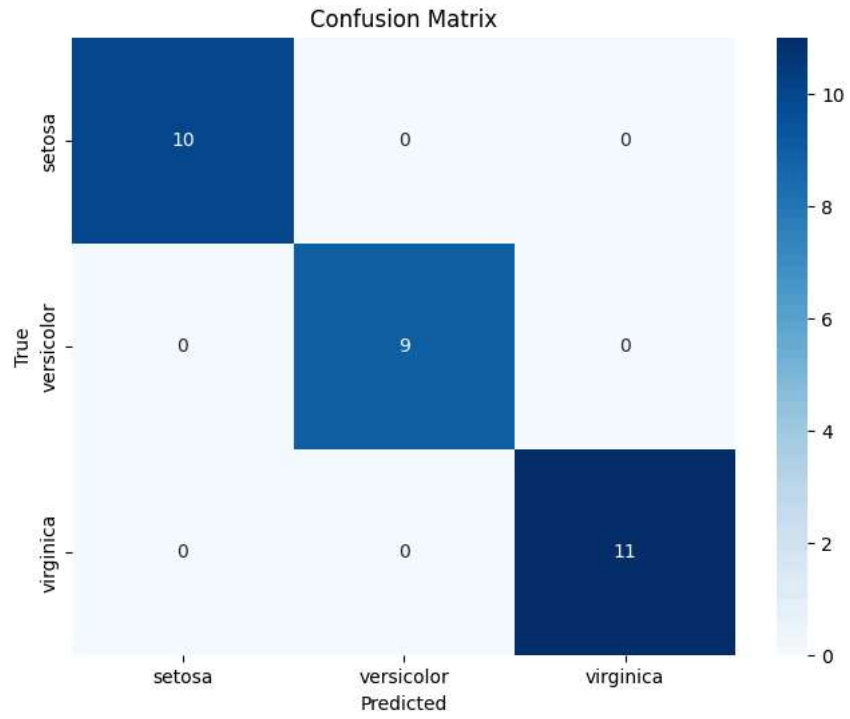
```
 Sab kuch chal gaya! Data split ho chuka hai.
X_train shape: (120, 4)
y_test shape: (30,)
```

```
# Train the model

model= LogisticRegression(random_state=42)
model.fit(X_train,y_train)
y_pred=model.predict(X_test)
print('Accuracy:',accuracy_score(y_test,y_pred))
print('Confusion Matrix:\n',confusion_matrix(y_test,y_pred))
print('Classification Report:\n',classification_report(y_test,y_pred))
```

```
Accuracy: 1.0
Confusion Matrix:
 [[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
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
```

```
# Step 1: Libraries
from sklearn.metrics import confusion_matrix, classification_report
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris


# Step 3: Confusion Matrix
cm = confusion_matrix(y_test, y_pred)

# Step 4: Plot
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=iris.target_names,
            yticklabels=iris.target_names)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')

# Step 5: Save safely (folder banao agar nahi hai)
import os
os.makedirs('plots', exist_ok=True)  # folder banao agar nahi hai
plt.savefig('plots/confusion_matrix.png')

# Step 6: Dikha do
plt.show()

# Step 7: Classification Report
print(classification_report(y_test, y_pred, target_names=iris.target_names))
```



	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

```
# Step 1: Libraries
import joblib
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression

# Step 2: Load data
iris = load_iris()
X = iris.data
y = iris.target

# Step 3: Scale features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Step 4: Train model
model = LogisticRegression()
model.fit(X_scaled, y)

# Step 5: Save model and scaler
joblib.dump(model, 'iris_classifier.pkl')
joblib.dump(scaler, 'scaler.pkl')

# Step 6: Example prediction
sample = [[7.2,3.2,6.9,2.0]]  # Sepal length, width, petal length, width
sample_scaled = scaler.transform(sample)
prediction = model.predict(sample_scaled)

# Convert number to species name
print('Predicted Species:', iris.target_names[prediction[0]])
```

```
Predicted Species: virginica
```

```
# 🟢 Step 1: Train a fresh model (Run this first)
from sklearn.datasets import load_iris
```

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
import joblib

# Load data
iris = load_iris()
X, y = iris.data, iris.target

# Check: kya teeno classes mojood hain?
print("Unique labels:", set(y))  # Must be {0, 1, 2}
print("Target names:", iris.target_names)  # ['setosa' 'versicolor' 'virginica']

# Scale the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42, stratify=y

# Train model
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)

# Test accuracy
```