PRACTICE PROJECTS

1. Iris Flower Classification

Description: Classify iris flowers into three species (setosa, versicolor, virginica) based on features like petal and sepal length/width using classification algorithms.

Steps:-

- Load the Iris dataset (available in sklearn.datasets).

- Explore and visualize the data using Pandas and Matplotlib/Seaborn.

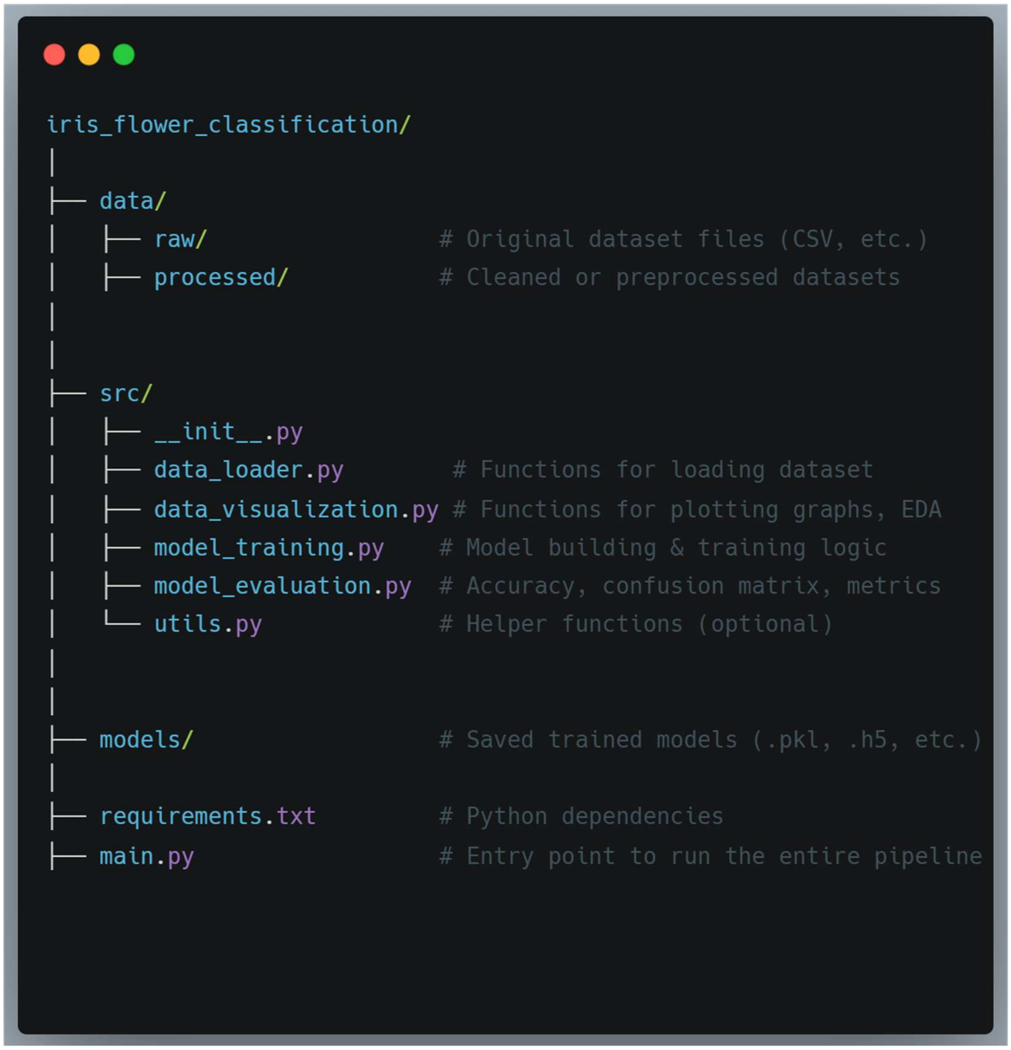
- Split the dataset into training and testing sets.

- Train a model using a classifier (e.g., Logistic Regression or KNN).

- Evaluate model accuracy using confusion matrix and classification report.

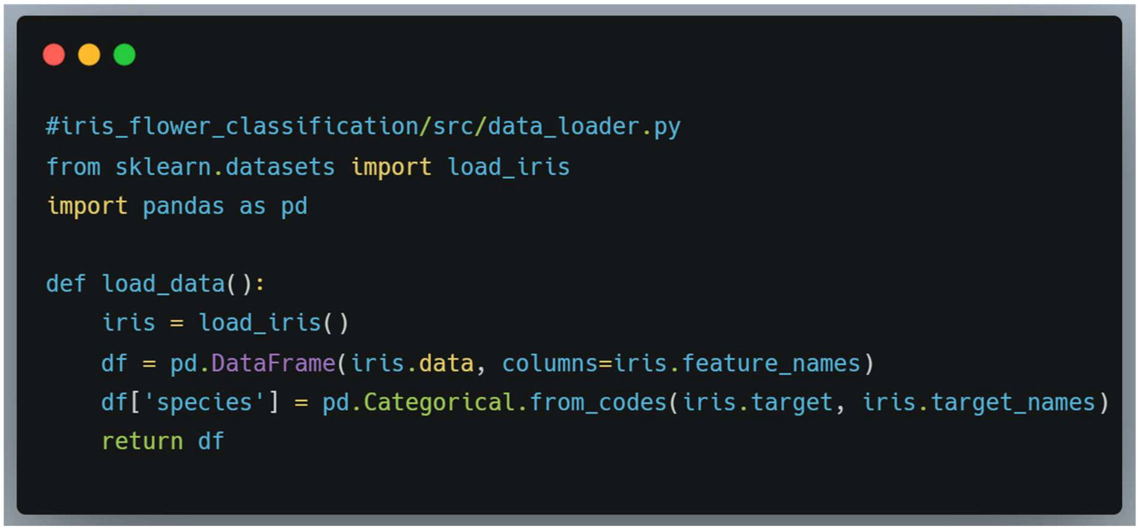
Project Code

Project folder structure



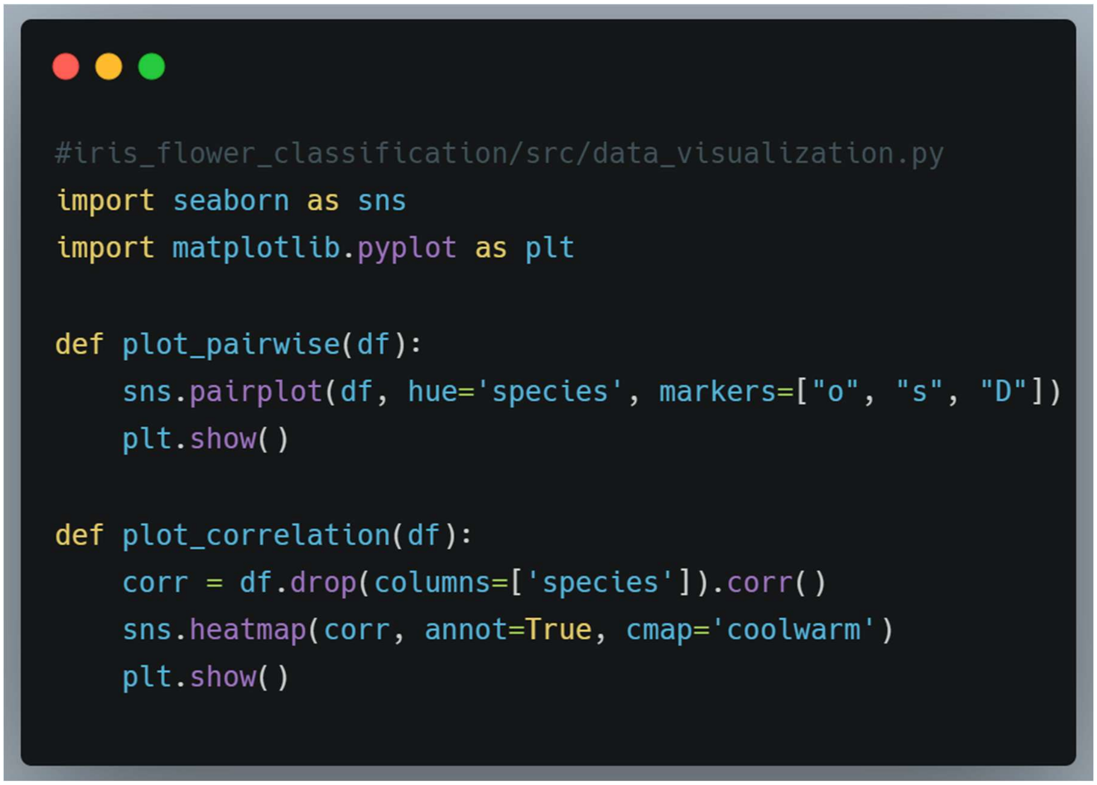
```
iris_flower_classification/
│
├── data/
│   ├── raw/                 # Original dataset files (CSV, etc.)
│   ├── processed/           # Cleaned or preprocessed datasets
│
│
├── src/
│   ├── __init__.py
│   ├── data_loader.py        # Functions for loading dataset
│   ├── data_visualization.py # Functions for plotting graphs, EDA
│   ├── model_training.py    # Model building & training logic
│   ├── model_evaluation.py  # Accuracy, confusion matrix, metrics
│   └── utils.py             # Helper functions (optional)
│
│
├── models/                  # Saved trained models (.pkl, .h5, etc.)
│
├── requirements.txt         # Python dependencies
├── main.py                  # Entry point to run the entire pipeline
```

#iris_flower_classification/src/data_loader.py



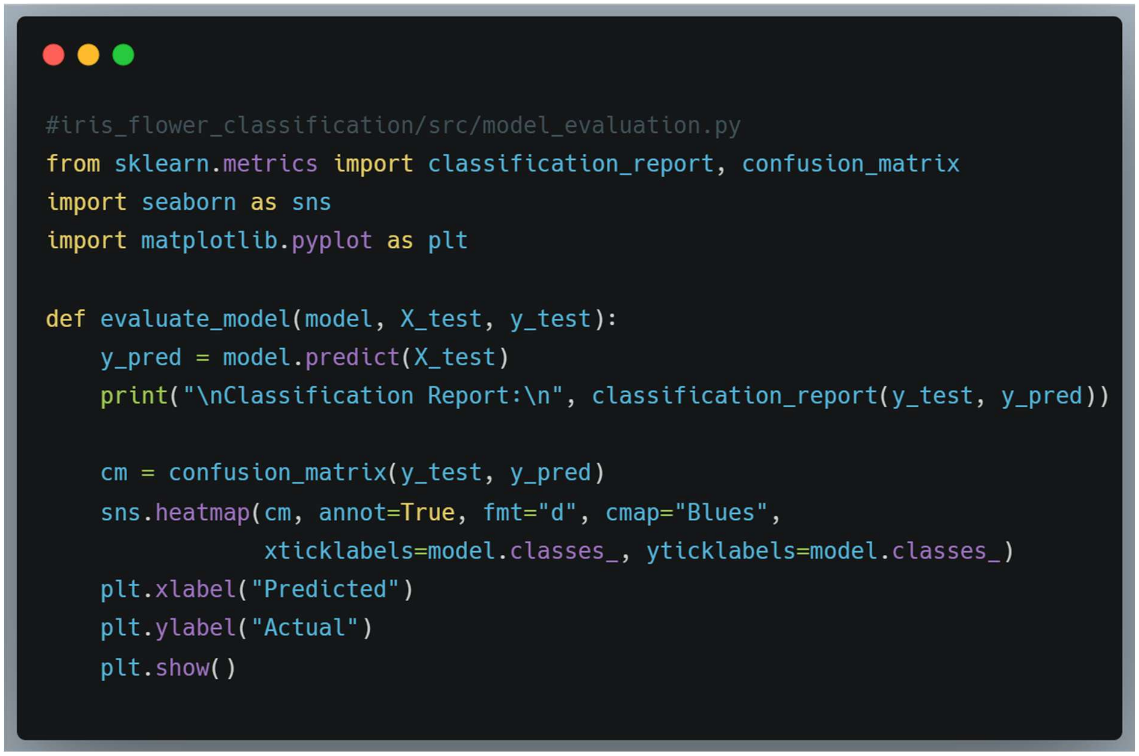
```python
#iris_flower_classification/src/data_loader.py
from sklearn.datasets import load_iris
import pandas as pd

def load_data():
    iris = load_iris()
    df = pd.DataFrame(iris.data, columns=iris.feature_names)
    df['species'] = pd.Categorical.from_codes(iris.target, iris.target_names)
    return df
```

#iris_flower_classification/src/data_visualization.py



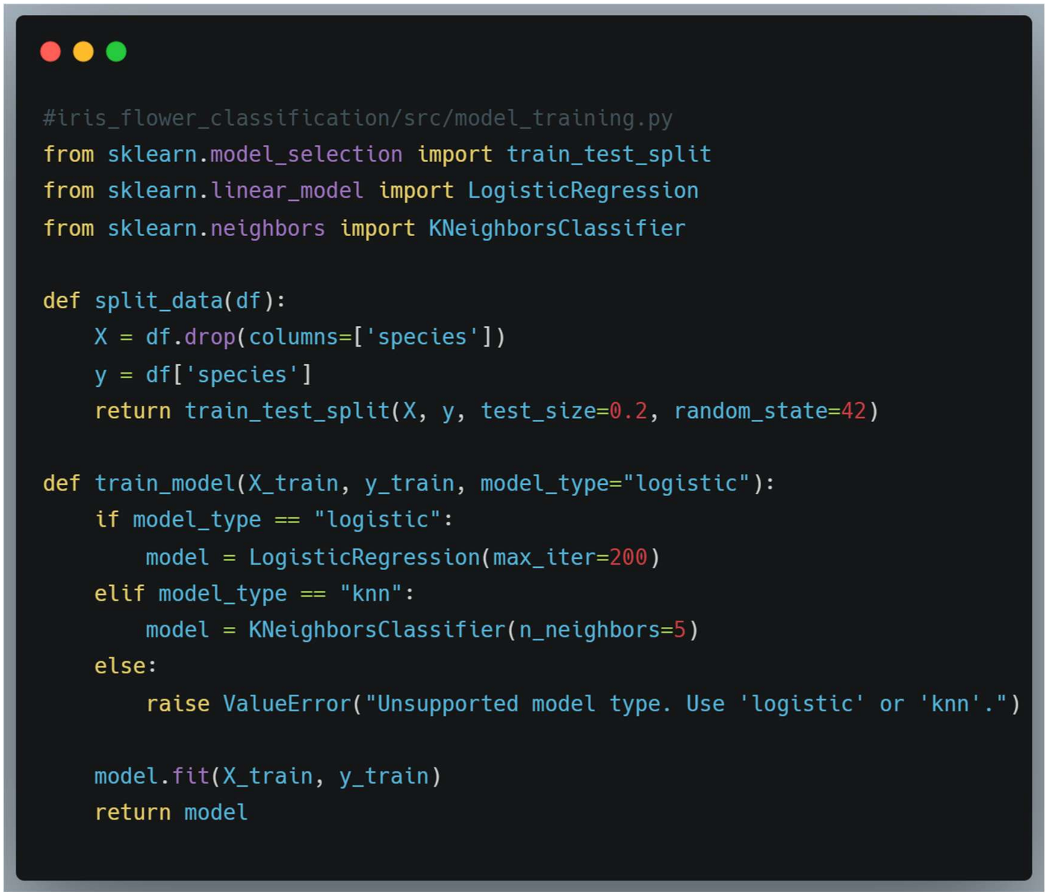
```python
#iris_flower_classification/src/data_visualization.py
import seaborn as sns
import matplotlib.pyplot as plt

def plot_pairwise(df):
    sns.pairplot(df, hue='species', markers=["o", "s", "D"])
    plt.show()

def plot_correlation(df):
    corr = df.drop(columns=['species']).corr()
    sns.heatmap(corr, annot=True, cmap='coolwarm')
    plt.show()
```

#iris_flower_classification/src/model_evaluation.py



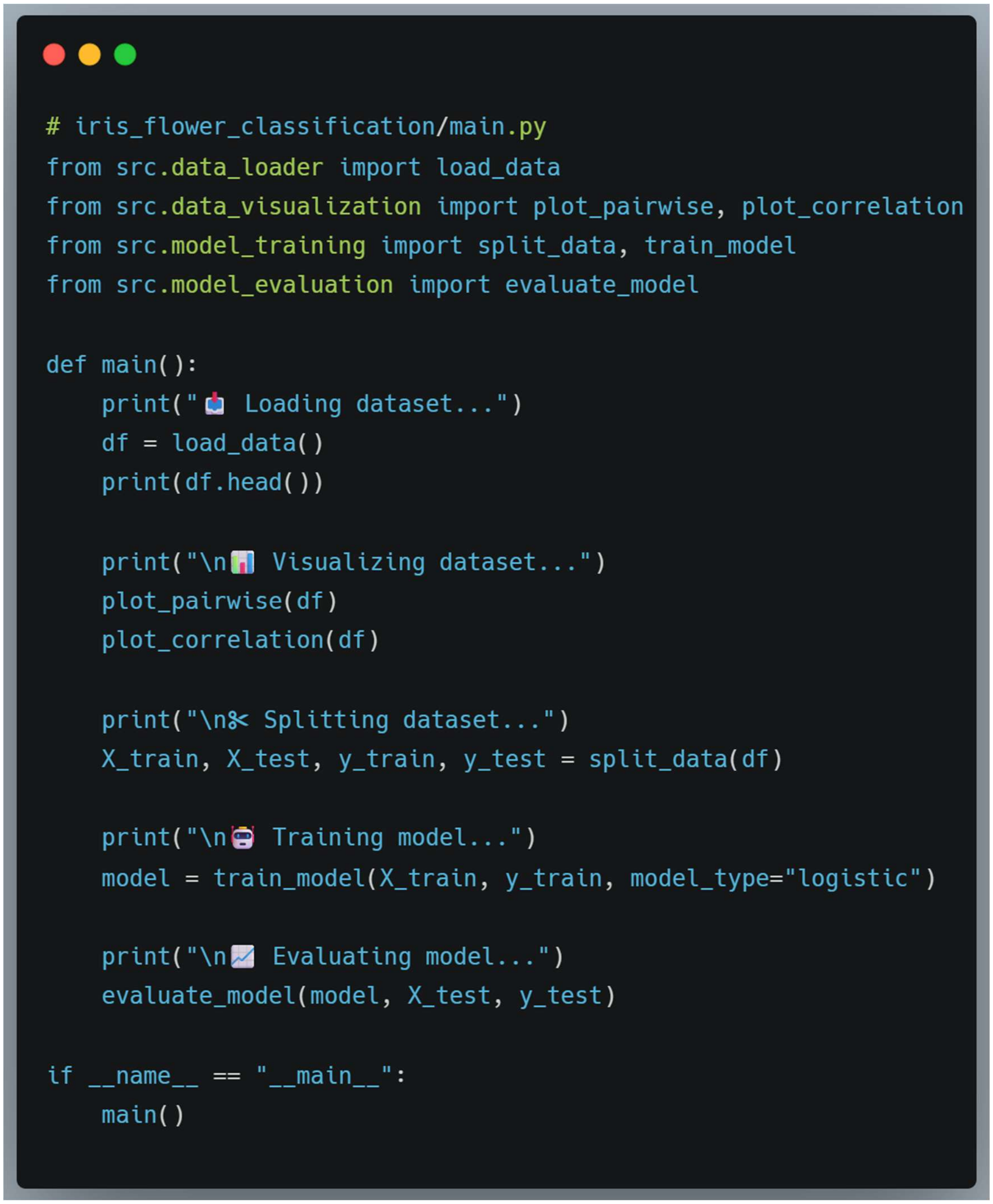
```python
#iris_flower_classification/src/model_evaluation.py
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

def evaluate_model(model, X_test, y_test):
    y_pred = model.predict(X_test)
    print("\nClassification Report:\n", classification_report(y_test, y_pred))

    cm = confusion_matrix(y_test, y_pred)
    sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
                xticklabels=model.classes_, yticklabels=model.classes_)
    plt.xlabel("Predicted")
    plt.ylabel("Actual")
    plt.show()
```

#iris_flower_classification/src/model_training.py



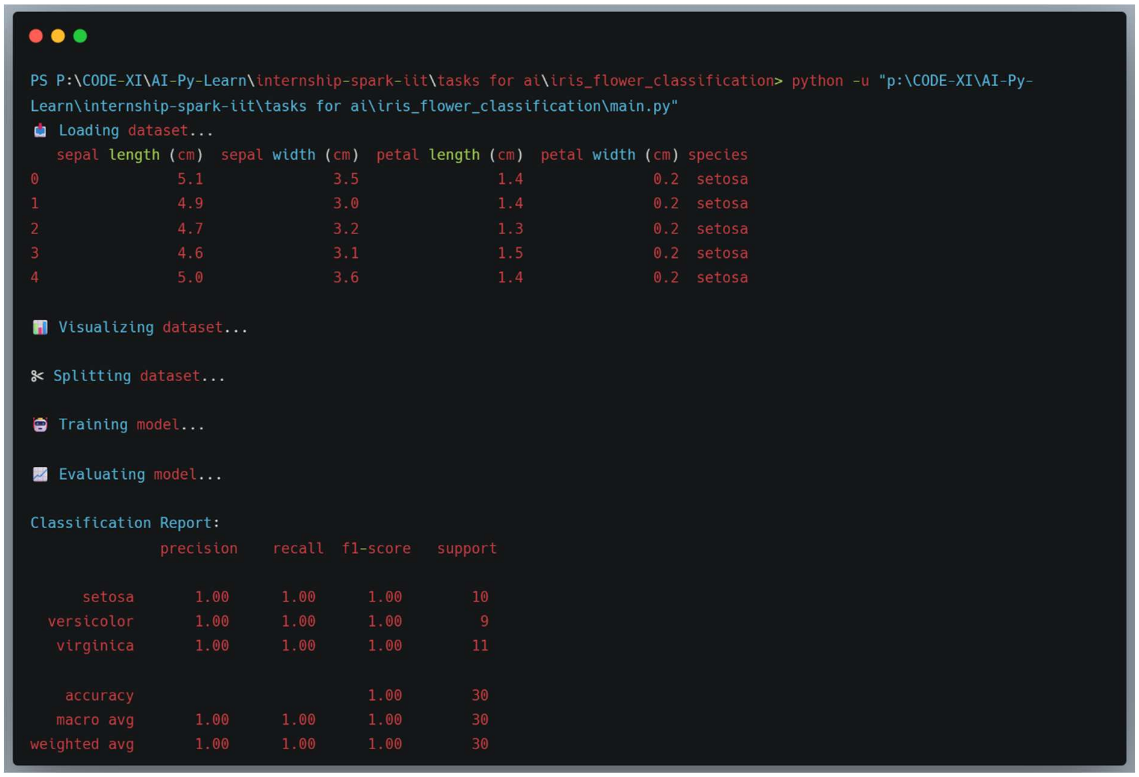
```python
#iris_flower_classification/src/model_training.py
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier

def split_data(df):
    X = df.drop(columns=['species'])
    y = df['species']
    return train_test_split(X, y, test_size=0.2, random_state=42)

def train_model(X_train, y_train, model_type="logistic"):
    if model_type == "logistic":
        model = LogisticRegression(max_iter=200)
    elif model_type == "knn":
        model = KNeighborsClassifier(n_neighbors=5)
    else:
        raise ValueError("Unsupported model type. Use 'logistic' or 'knn'.")

    model.fit(X_train, y_train)
    return model
```

#iris_flower_classification/main.py

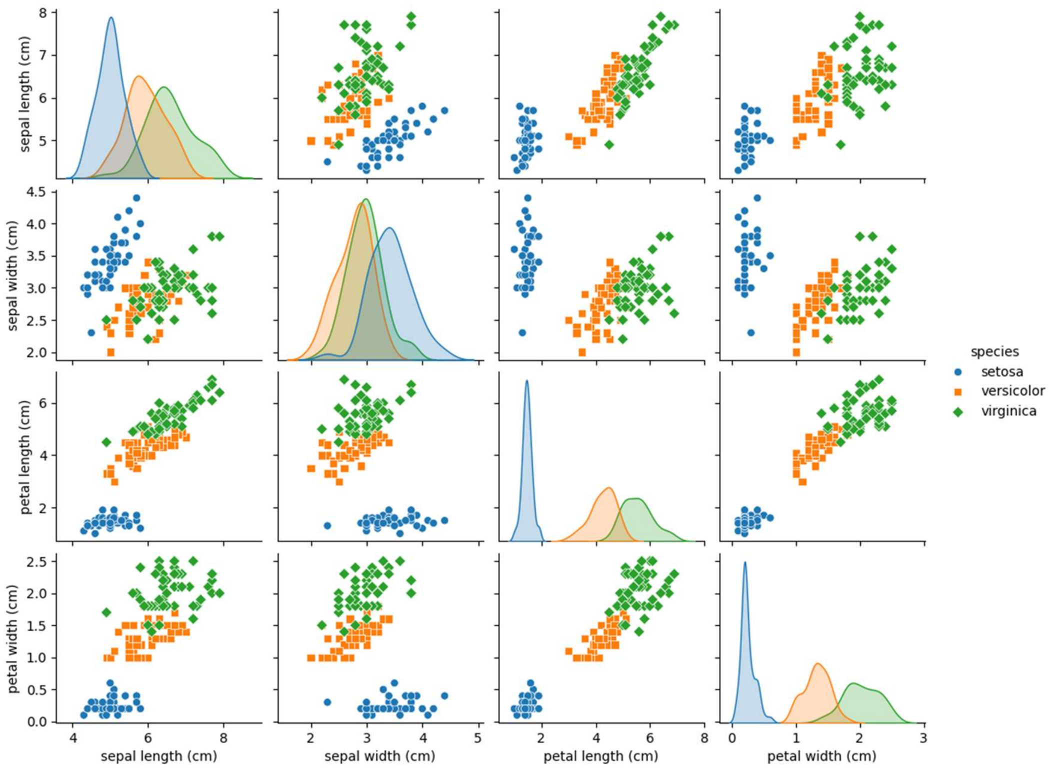


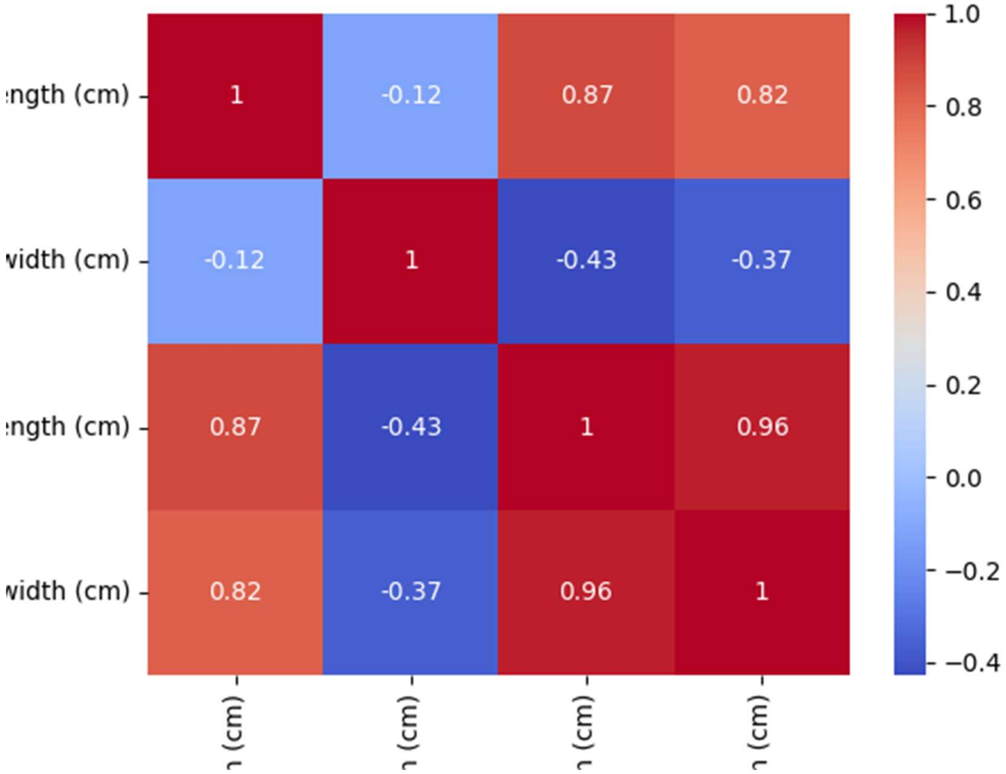
```python
# iris_flower_classification/main.py
from src.data_loader import load_data
from src.data_visualization import plot_pairwise, plot_correlation
from src.model_training import split_data, train_model
from src.model_evaluation import evaluate_model

def main():
    print("📥 Loading dataset...")
    df = load_data()
    print(df.head())

    print("\n📊 Visualizing dataset...")
    plot_pairwise(df)
    plot_correlation(df)

    print("\n✂ Splitting dataset...")
    X_train, X_test, y_train, y_test = split_data(df)

    print("\n🤖 Training model...")
    model = train_model(X_train, y_train, model_type="logistic")

    print("\n📈 Evaluating model...")
    evaluate_model(model, X_test, y_test)

if __name__ == "__main__":
    main()
```

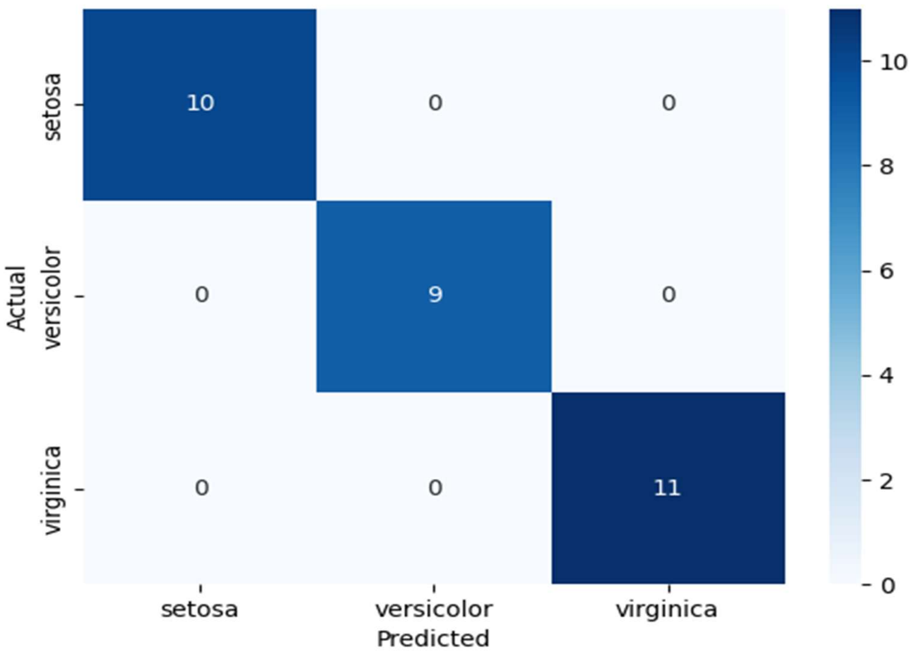
Project Output



```
PS P:\CODE-XI\AI-Py-Learn\internship-spark-iit\tasks for ai\iris_flower_classification> python -u "p:\CODE-XI\AI-Py-
Learn\internship-spark-iit\tasks for ai\iris_flower_classification\main.py"
📥 Loading dataset...
   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

📊 Visualizing dataset...

✂ Splitting dataset...

🤖 Training model...

📈 Evaluating model...

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
```




ength (cm)
width (cm)
ength (cm)
width (cm)
1
-0.12
0.87
0.82
-0.12
1
-0.43
-0.37
0.87
-0.43
1
0.96
0.82
-0.37
0.96
1
n (cm)
n (cm)
n (cm)
n (cm)
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4


Actual
setosa
versicolor
virginica
10
0
0
0
9
0
0
0
11
setosa
versicolor
virginica
Predicted
10
8
6
4
2
0