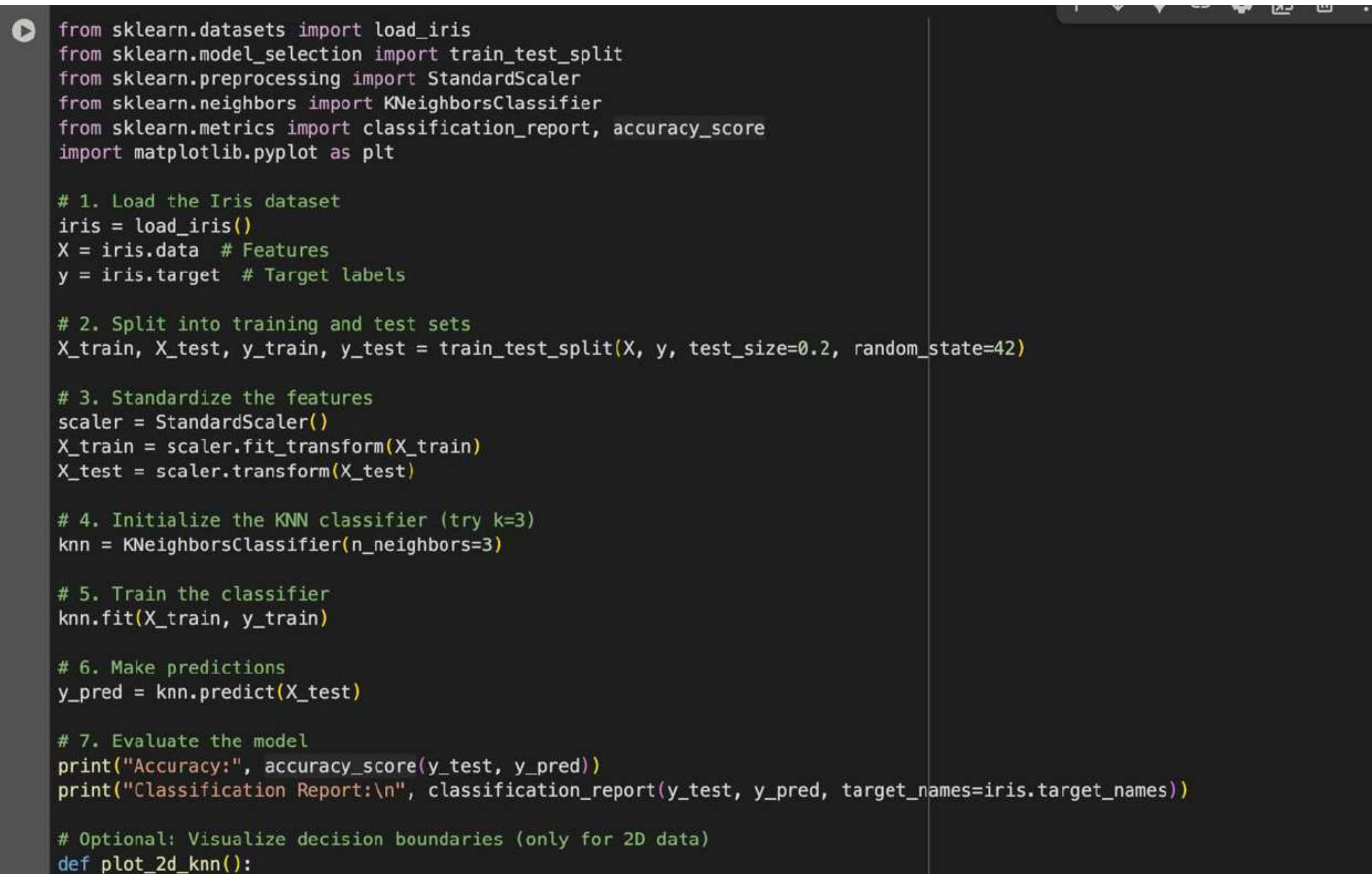
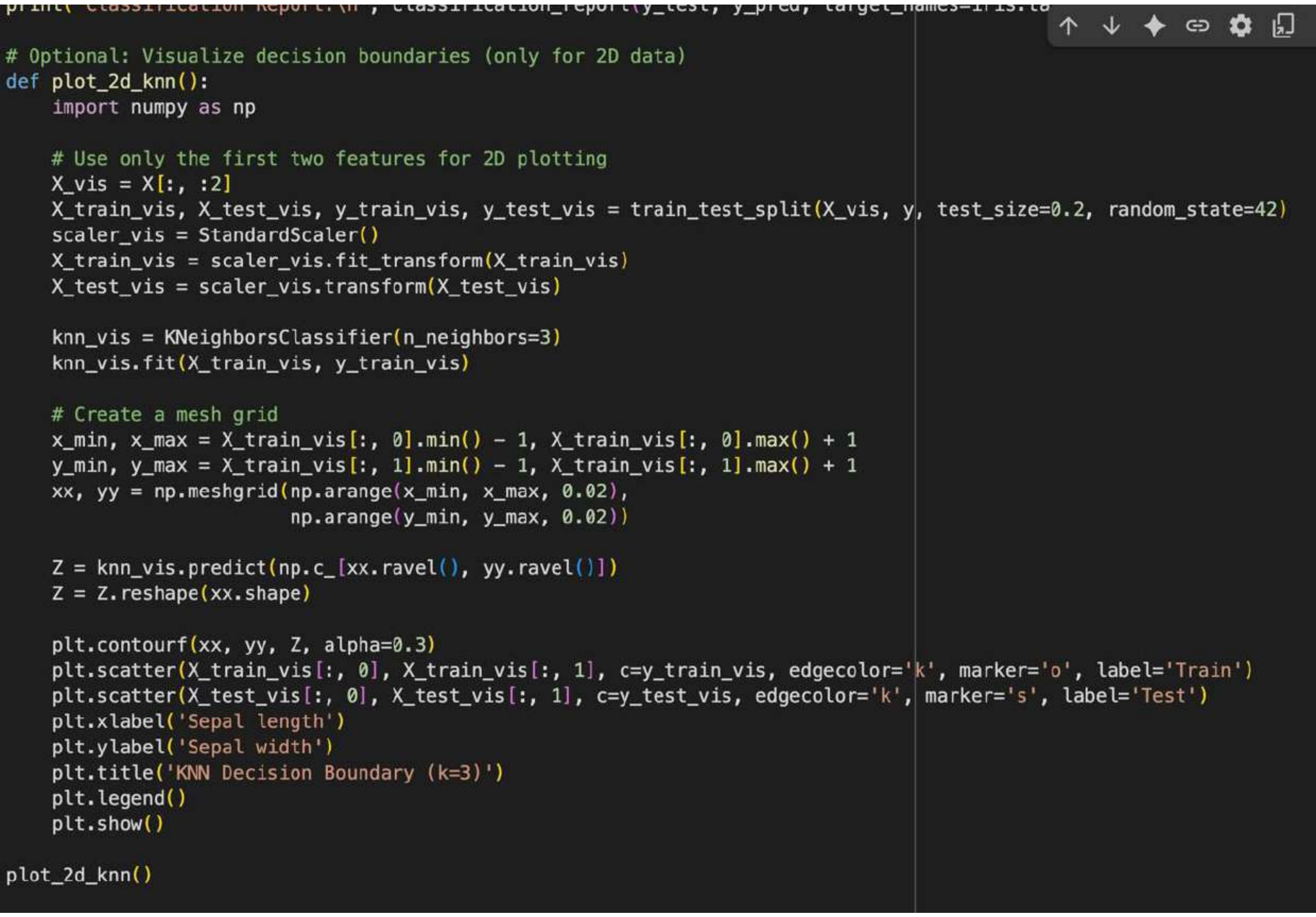
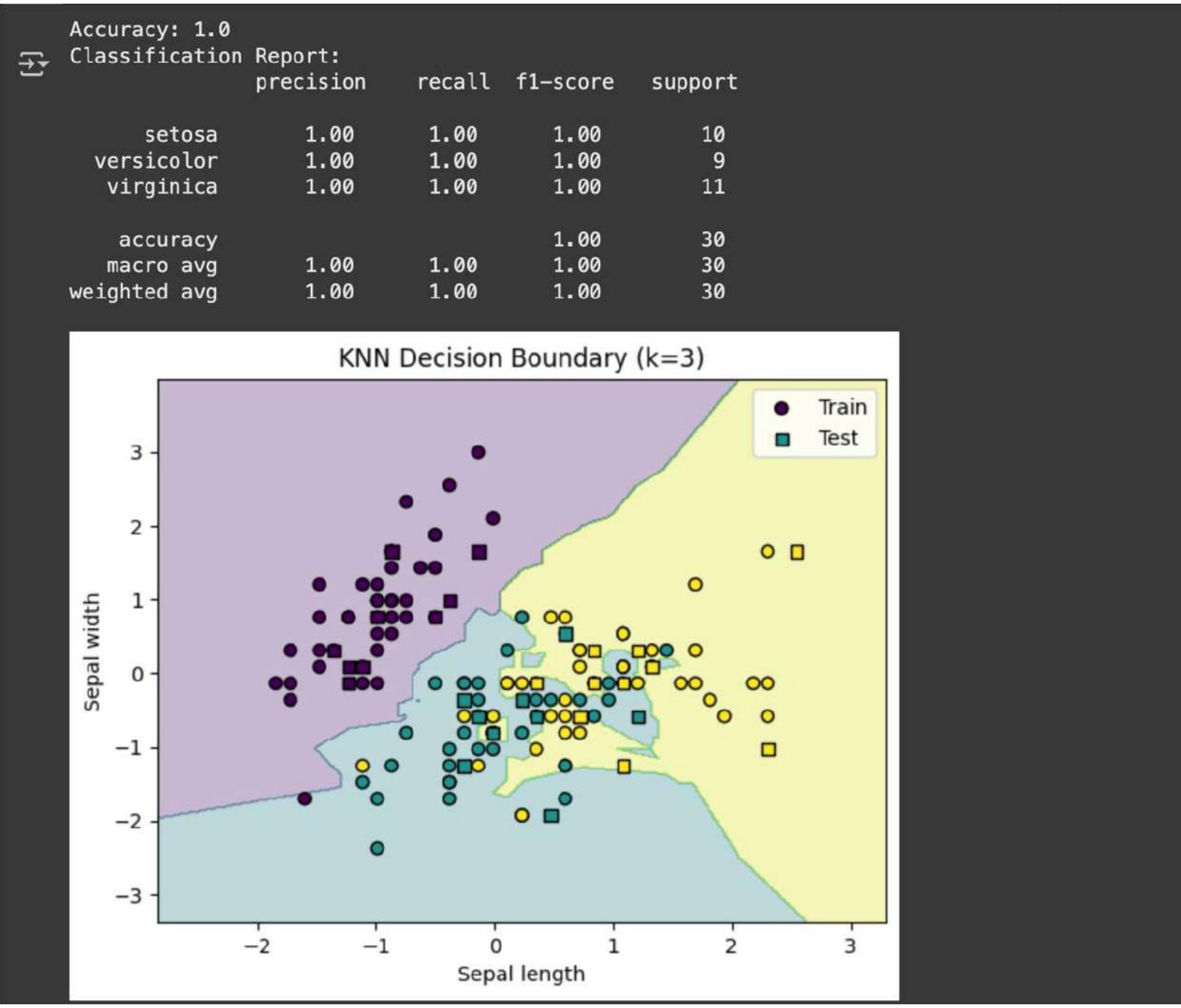


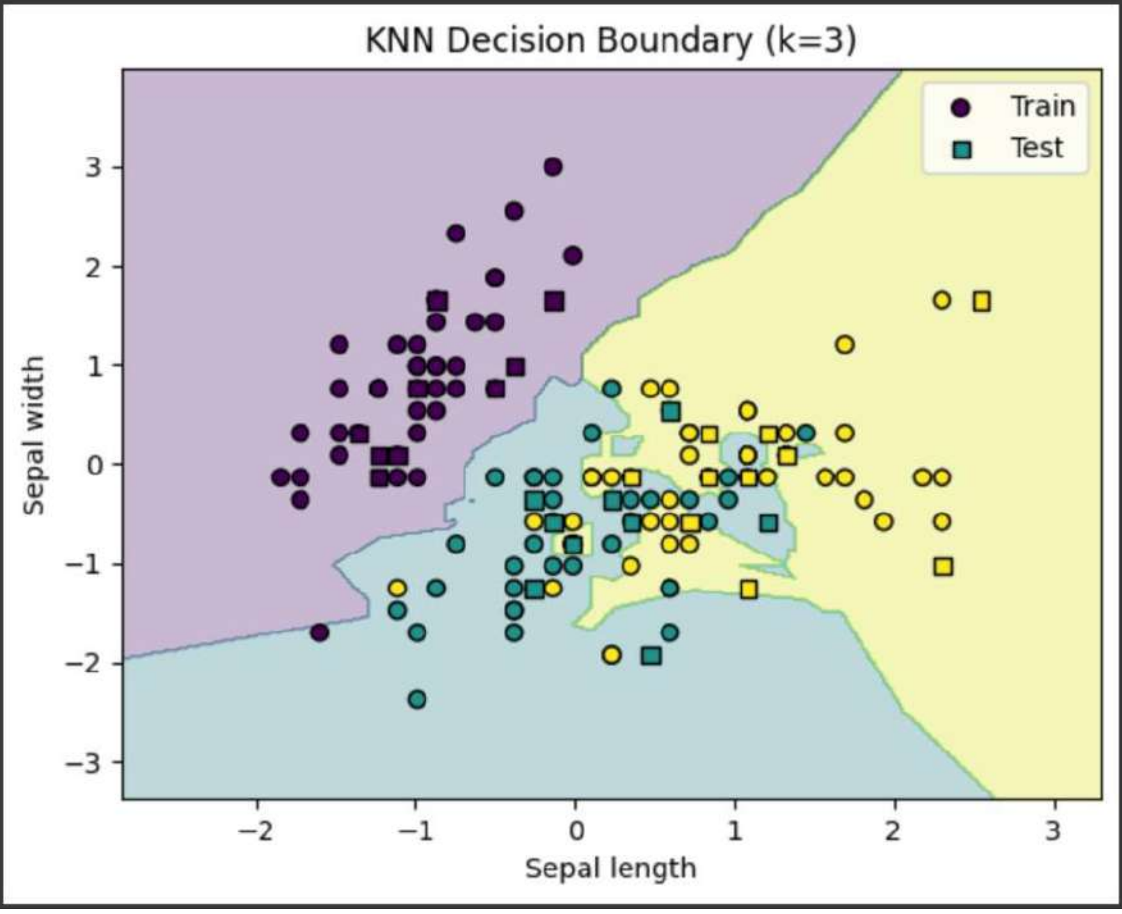
```python
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, accuracy_score
import matplotlib.pyplot as plt

# 1. Load the Iris dataset
iris = load_iris()
X = iris.data  # Features
y = iris.target  # Target labels

# 2. Split into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 3. Standardize the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# 4. Initialize the KNN classifier (try k=3)
knn = KNeighborsClassifier(n_neighbors=3)

# 5. Train the classifier
knn.fit(X_train, y_train)

# 6. Make predictions
y_pred = knn.predict(X_test)

# 7. Evaluate the model
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred, target_names=iris.target_names))

# Optional: Visualize decision boundaries (only for 2D data)
def plot_2d_knn():
```



```python
print('Classification Report:\n', classification_report(y_test, y_pred, target_names=iris.ta

# Optional: Visualize decision boundaries (only for 2D data)
def plot_2d_knn():
    import numpy as np

    # Use only the first two features for 2D plotting
    X_vis = X[:, :2]
    X_train_vis, X_test_vis, y_train_vis, y_test_vis = train_test_split(X_vis, y, test_size=0.2, random_state=42)
    scaler_vis = StandardScaler()
    X_train_vis = scaler_vis.fit_transform(X_train_vis)
    X_test_vis = scaler_vis.transform(X_test_vis)

    knn_vis = KNeighborsClassifier(n_neighbors=3)
    knn_vis.fit(X_train_vis, y_train_vis)

    # Create a mesh grid
    x_min, x_max = X_train_vis[:, 0].min() - 1, X_train_vis[:, 0].max() + 1
    y_min, y_max = X_train_vis[:, 1].min() - 1, X_train_vis[:, 1].max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.02),
                         np.arange(y_min, y_max, 0.02))

    Z = knn_vis.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)

    plt.contourf(xx, yy, Z, alpha=0.3)
    plt.scatter(X_train_vis[:, 0], X_train_vis[:, 1], c=y_train_vis, edgecolor='k', marker='o', label='Train')
    plt.scatter(X_test_vis[:, 0], X_test_vis[:, 1], c=y_test_vis, edgecolor='k', marker='s', label='Test')
    plt.xlabel('Sepal length')
    plt.ylabel('Sepal width')
    plt.title('KNN Decision Boundary (k=3)')
    plt.legend()
    plt.show()

plot_2d_knn()
```



```
Accuracy: 1.0
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
```


KNN Decision Boundary (k=3)
Train
Test
Sepal width
Sepal length