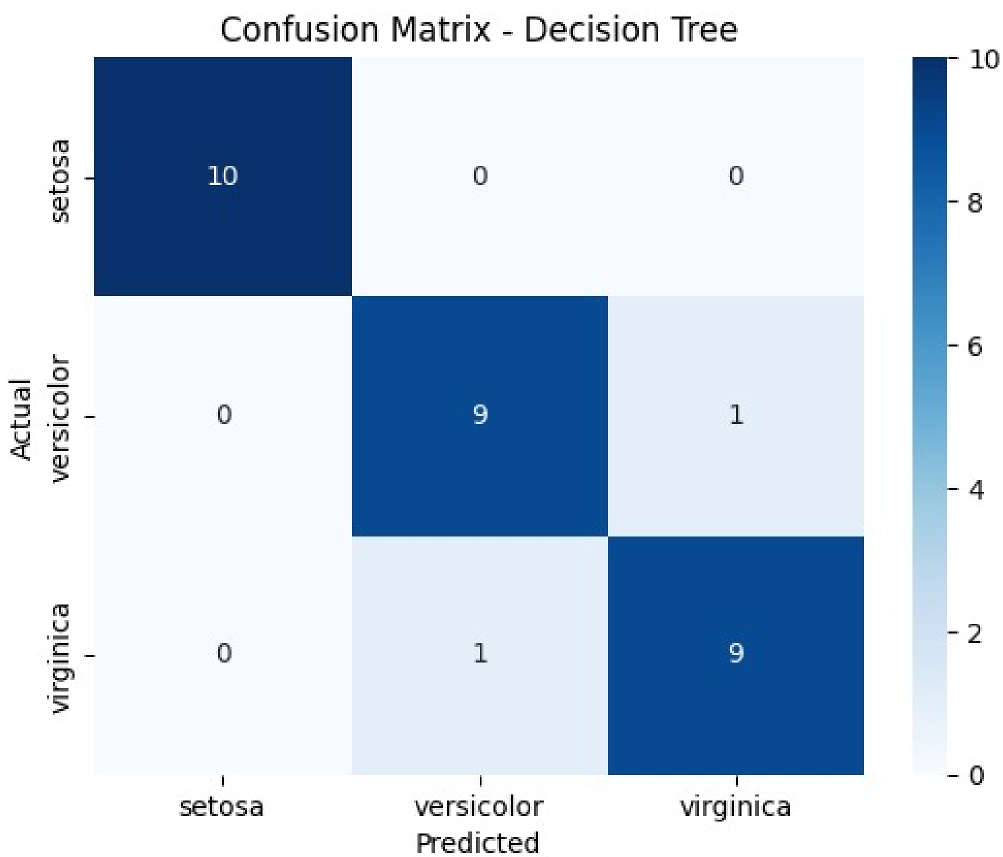
```python
# Import necessary libraries
import pandas as pd
import numpy as np
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC

# Load the Iris dataset
iris = load_iris()
X = iris.data
y = iris.target
feature_names = iris.feature_names
target_names = iris.target_names

# Convert to DataFrame for better readability
data = pd.DataFrame(X, columns=feature_names)
data['Target'] = y

# Display dataset information
print("First 5 rows of the dataset:")
print(data.head())
print("\nSummary statistics:")
print(data.describe())

# Check for missing values
print("\nChecking for missing values:")
print(data.isnull().sum())

# Feature scaling
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.2, random_state=42, stratify=y
)

print("\nDataset split: ")
print(f"Training samples: {X_train.shape[0]}")
print(f"Test samples: {X_test.shape[0]}")

# Define models
```

```
models = {
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "Random Forest": RandomForestClassifier(random_state=42),
    "K-Nearest Neighbors": KNeighborsClassifier(),
    "Logistic Regression": LogisticRegression(random_state=42,
max_iter=500),
    "SVM": SVC(kernel='linear', random_state=42)
}

# Train and evaluate each model
results = {}
for model_name, model in models.items():
    print(f"\nTraining {model_name}...")
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    # Classification report and confusion matrix
    print(f"\nClassification Report for {model_name}:")
    print(classification_report(y_test, y_pred,
target_names=target_names))

    print(f"\nConfusion Matrix for {model_name}:")
    cm = confusion_matrix(y_test, y_pred)
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
xticklabels=target_names, yticklabels=target_names)
    plt.title(f"Confusion Matrix - {model_name}")
    plt.xlabel("Predicted")
    plt.ylabel("Actual")
    plt.show()

    # Store results
    results[model_name] = {
        "accuracy": model.score(X_test, y_test),
        "classification_report": classification_report(y_test, y_pred,
output_dict=True)
    }

# Compare accuracy scores
print("\nModel Performance Summary:")
for model_name, metrics in results.items():
    print(f"{model_name} - Accuracy: {metrics['accuracy']:.2f}")

# Insights and Suggestions
print("\nInsights and Suggestions:")
print("1. Evaluate using different hyperparameters for the models.")
print("2. Test with other datasets (e.g., Breast Cancer or Wine
Quality).")
print("3. Use advanced techniques such as GridSearchCV for
hyperparameter tuning.")
```

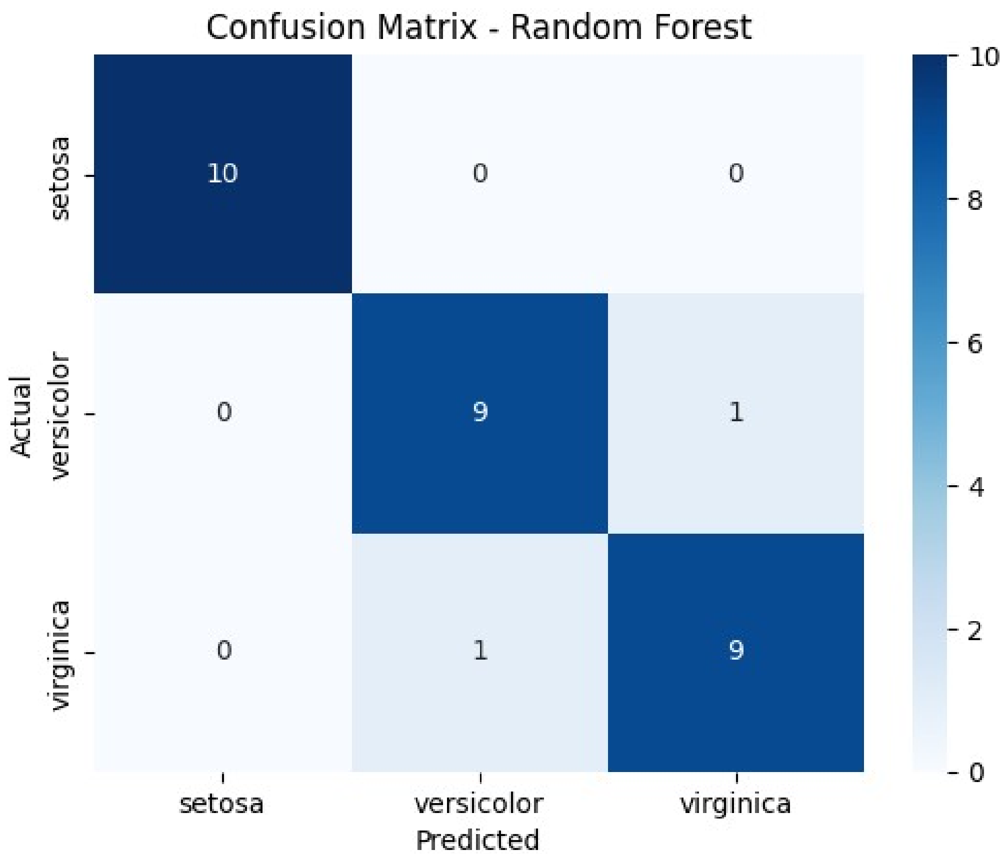
```
First 5 rows of the dataset:
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width
(cm)  \
0                5.1               3.5                1.4
0.2
1                4.9               3.0                1.4
0.2
2                4.7               3.2                1.3
0.2
3                4.6               3.1                1.5
0.2
4                5.0               3.6                1.4
0.2

   Target
0       0
1       0
2       0
3       0
4       0

Summary statistics:
       sepal length (cm)  sepal width (cm)  petal length (cm)  \
count         150.000000        150.000000         150.000000
mean            5.843333          3.057333           3.758000
std             0.828066          0.435866           1.765298
min             4.300000          2.000000           1.000000
25%             5.100000          2.800000           1.600000
50%             5.800000          3.000000           4.350000
75%             6.400000          3.300000           5.100000
max             7.900000          4.400000           6.900000

       petal width (cm)      Target
count        150.000000  150.000000
mean           1.199333    1.000000
std            0.762238    0.819232
min            0.100000    0.000000
25%            0.300000    0.000000
50%            1.300000    1.000000
75%            1.800000    2.000000
max            2.500000    2.000000

Checking for missing values:
sepal length (cm)    0
sepal width (cm)     0
petal length (cm)    0
petal width (cm)     0
Target               0
dtype: int64
```

```
Dataset split:
Training samples: 120
Test samples: 30

Training Decision Tree...

Classification Report for Decision Tree:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        10
  versicolor       0.90      0.90      0.90        10
   virginica       0.90      0.90      0.90        10

    accuracy                           0.93        30
   macro avg       0.93      0.93      0.93        30
weighted avg       0.93      0.93      0.93        30


Confusion Matrix for Decision Tree:
```



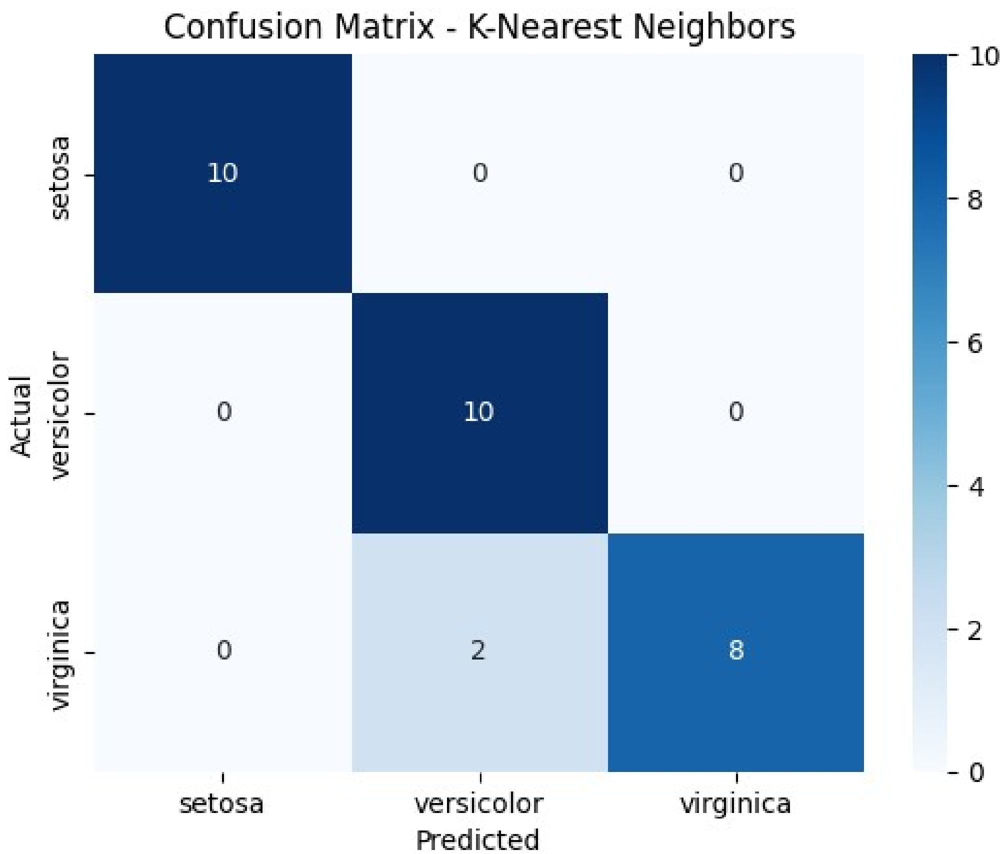

```


Training Random Forest...

```

```
Classification Report for Random Forest:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        10
  versicolor       0.90      0.90      0.90        10
   virginica       0.90      0.90      0.90        10

    accuracy                           0.93        30
   macro avg       0.93      0.93      0.93        30
weighted avg       0.93      0.93      0.93        30


Confusion Matrix for Random Forest:
```



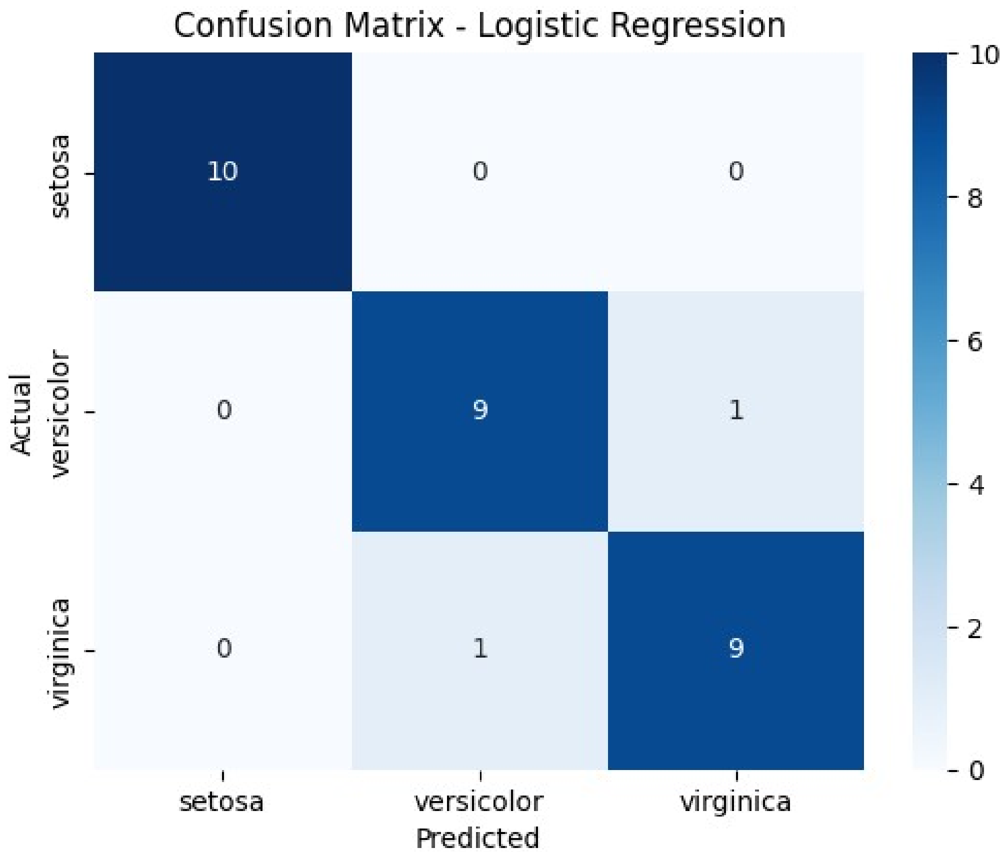

```

Training K-Nearest Neighbors...

Classification Report for K-Nearest Neighbors:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        10
  versicolor       0.83      1.00      0.91        10
   virginica       1.00      0.80      0.89        10
```

```
    accuracy                           0.93        30
   macro avg       0.94      0.93      0.93        30
weighted avg       0.94      0.93      0.93        30


Confusion Matrix for K-Nearest Neighbors:
```

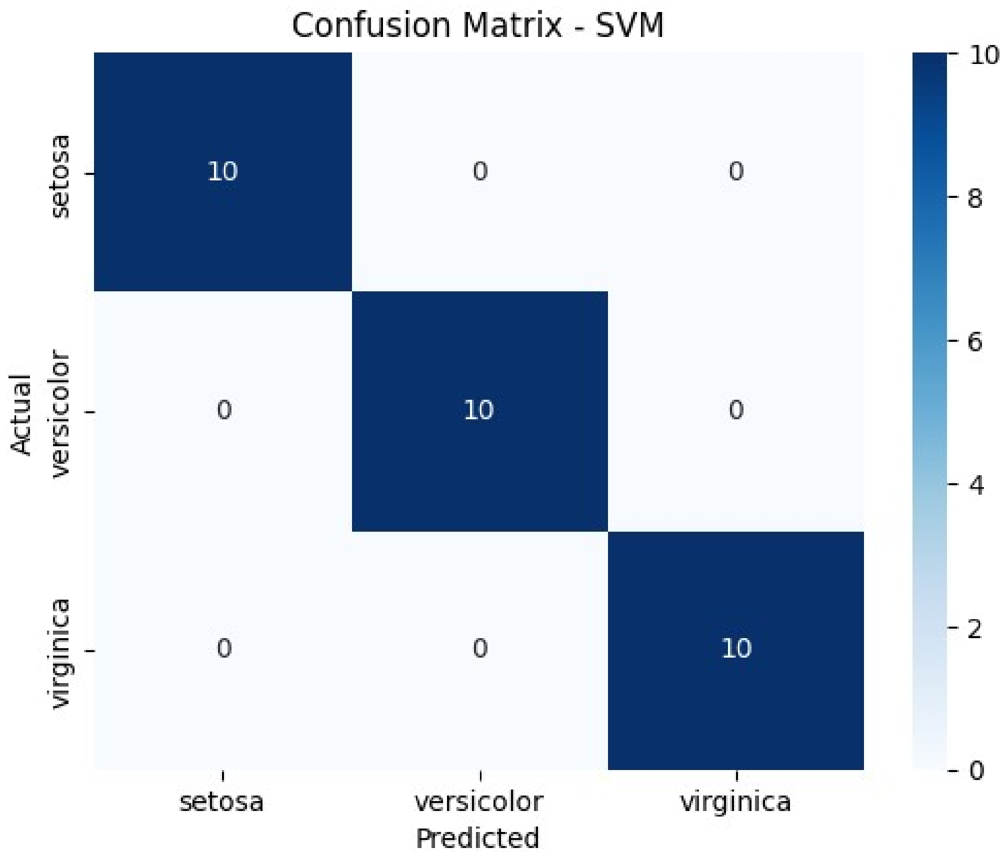



```
Training Logistic Regression...

Classification Report for Logistic Regression:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        10
  versicolor       0.90      0.90      0.90        10
   virginica       0.90      0.90      0.90        10

    accuracy                           0.93        30
   macro avg       0.93      0.93      0.93        30
weighted avg       0.93      0.93      0.93        30
```

```
Confusion Matrix for Logistic Regression:
```




```
Training SVM...

Classification Report for SVM:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        10
  versicolor       1.00      1.00      1.00        10
   virginica       1.00      1.00      1.00        10

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30


Confusion Matrix for SVM:
```




```
Model Performance Summary:
Decision Tree - Accuracy: 0.93
Random Forest - Accuracy: 0.93
K-Nearest Neighbors - Accuracy: 0.93
Logistic Regression - Accuracy: 0.93
SVM - Accuracy: 1.00

Insights and Suggestions:
1. Evaluate using different hyperparameters for the models.
2. Test with other datasets (e.g., Breast Cancer or Wine Quality).
3. Use advanced techniques such as GridSearchCV for hyperparameter
tuning.
```