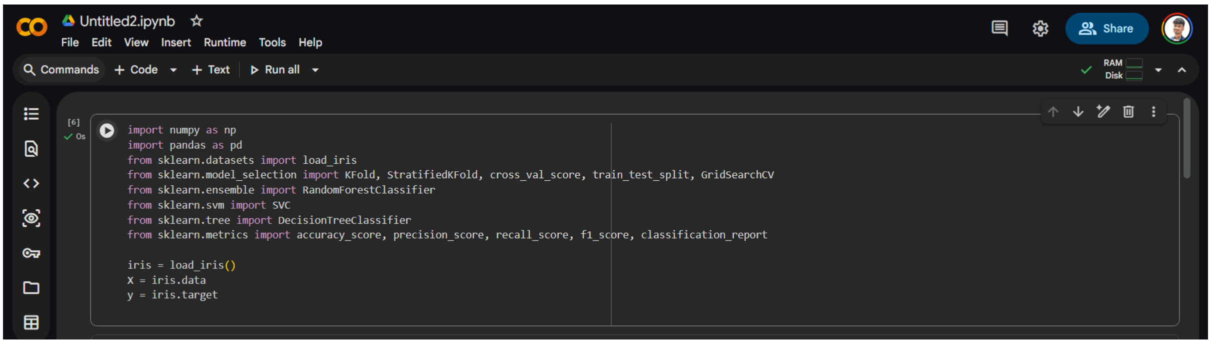
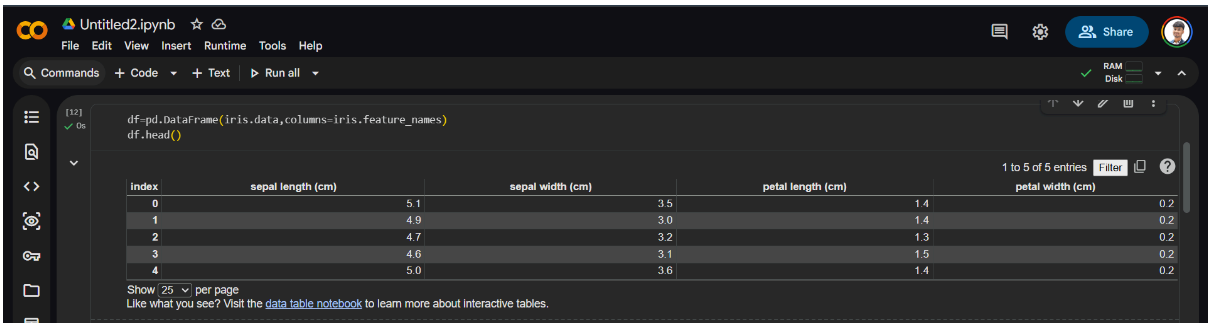

Untitled2.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text ▷ Run all
Share
RAM
Disk
[6]
✓ 0s

```
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import KFold, StratifiedKFold, cross_val_score, train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, classification_report

iris = load_iris()
X = iris.data
y = iris.target
```

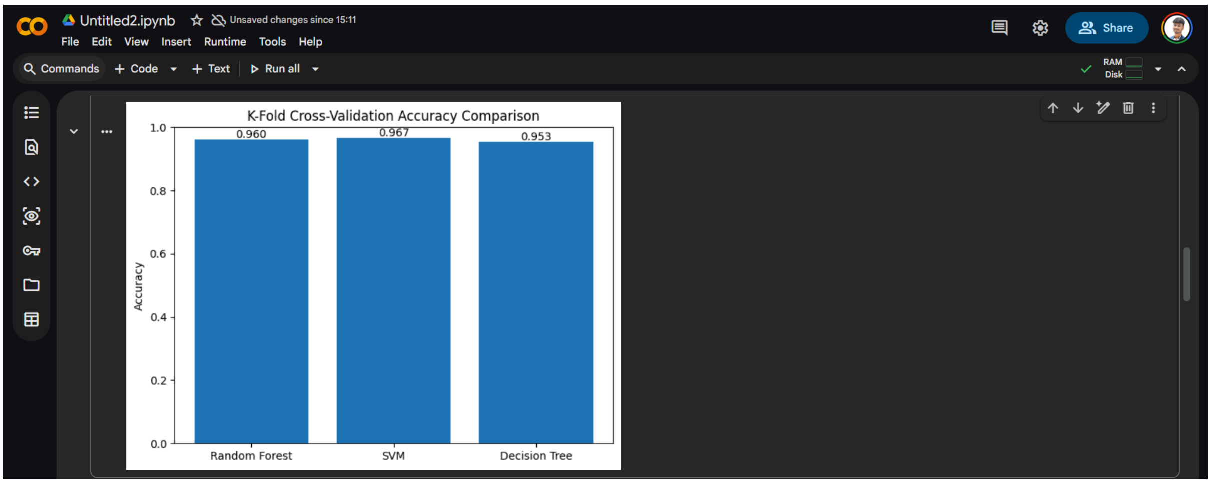

Untitled2.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text ▷ Run all
Share
RAM
Disk
[12]
✓ 0s

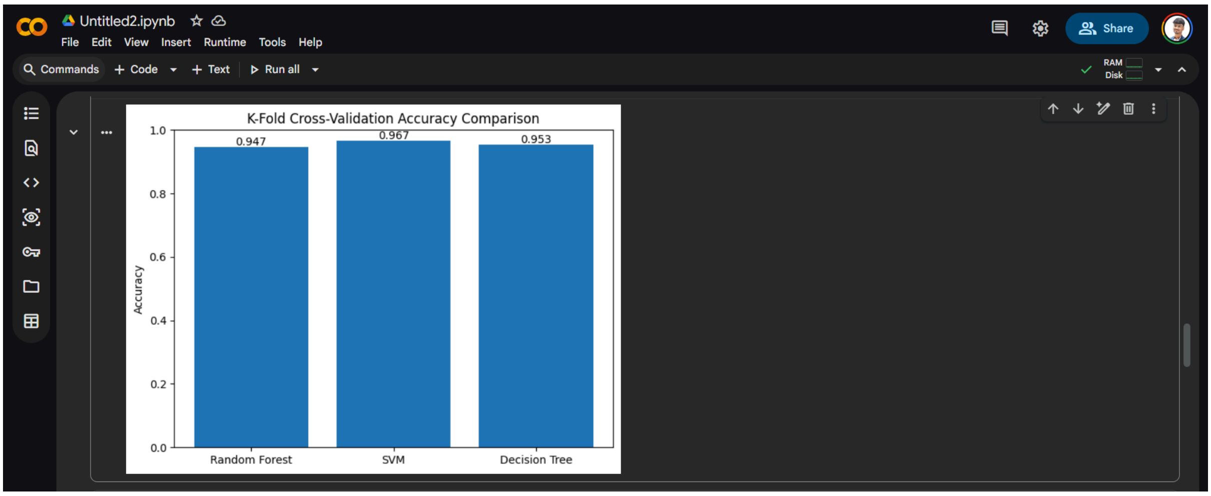
```
df=pd.DataFrame(iris.data,columns=iris.feature_names)
df.head()
```

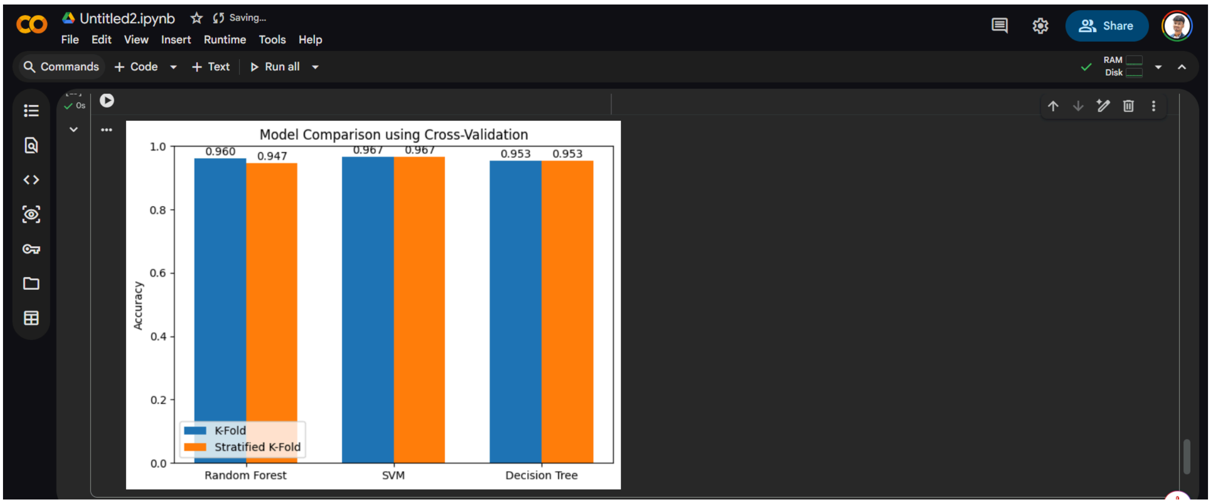
1 to 5 of 5 entries Filter

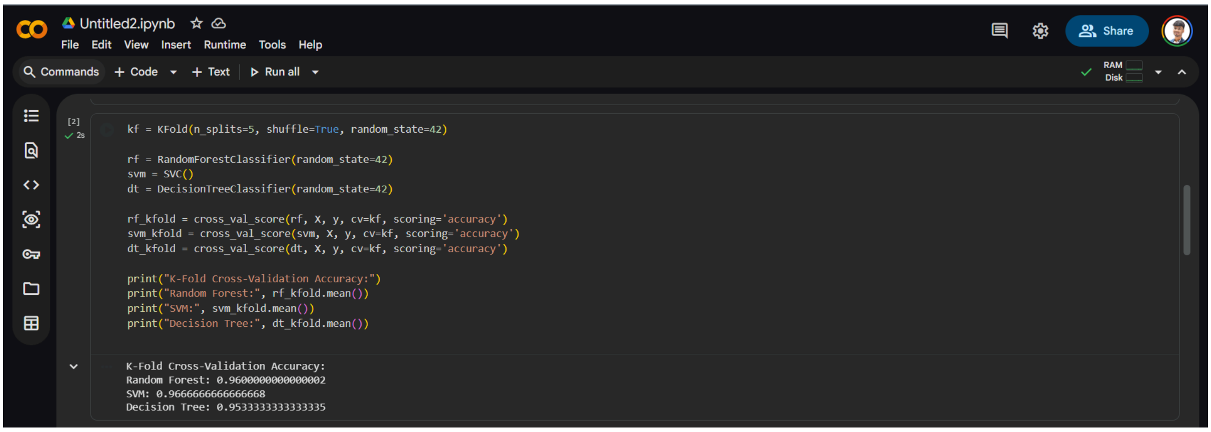
| index | sepal length (cm) | sepal width (cm) | petal length (cm) | petal width (cm) |
|---|---|---|---|---|
| 0 | 5.1 | 3.5 | 1.4 | 0.2 |
| 1 | 4.9 | 3.0 | 1.4 | 0.2 |
| 2 | 4.7 | 3.2 | 1.3 | 0.2 |
| 3 | 4.6 | 3.1 | 1.5 | 0.2 |
| 4 | 5.0 | 3.6 | 1.4 | 0.2 |

Show 25 per page
Like what you see? Visit the data table notebook to learn more about interactive tables.


Untitled2.ipynb
Unsaved changes since 15:11
File Edit View Insert Runtime Tools Help
Commands + Code + Text ▷ Run all
Share
RAM
Disk
K-Fold Cross-Validation Accuracy Comparison
Accuracy
1.0
0.8
0.6
0.4
0.2
0.0
0.960
0.967
0.953
Random Forest
SVM
Decision Tree

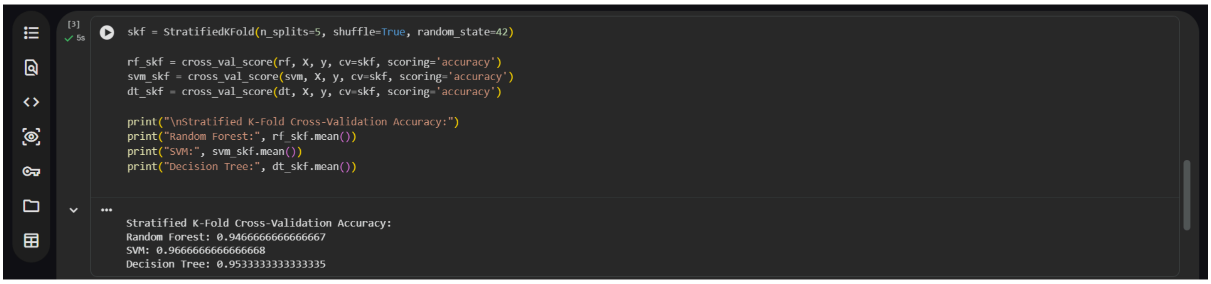

Untitled2.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text ▷ Run all
Share
RAM
Disk
K-Fold Cross-Validation Accuracy Comparison
Accuracy
1.0
0.8
0.6
0.4
0.2
0.0
0.947
0.967
0.953
Random Forest
SVM
Decision Tree





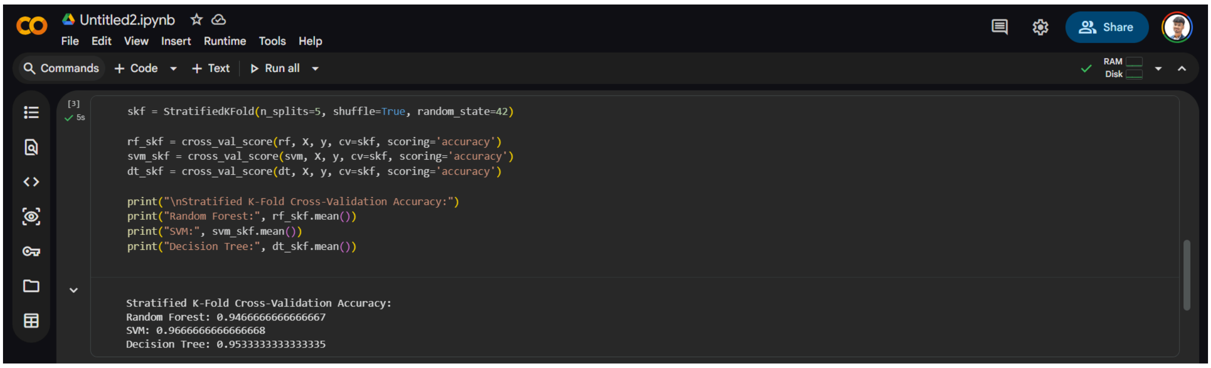
```
kf = KFold(n_splits=5, shuffle=True, random_state=42)

rf = RandomForestClassifier(random_state=42)
svm = SVC()
dt = DecisionTreeClassifier(random_state=42)

rf_kfold = cross_val_score(rf, X, y, cv=kf, scoring='accuracy')
svm_kfold = cross_val_score(svm, X, y, cv=kf, scoring='accuracy')
dt_kfold = cross_val_score(dt, X, y, cv=kf, scoring='accuracy')

print("K-Fold Cross-Validation Accuracy:")
print("Random Forest:", rf_kfold.mean())
print("SVM:", svm_kfold.mean())
print("Decision Tree:", dt_kfold.mean())
```

```
K-Fold Cross-Validation Accuracy:
Random Forest: 0.9600000000000002
SVM: 0.9666666666666668
Decision Tree: 0.9533333333333335
```



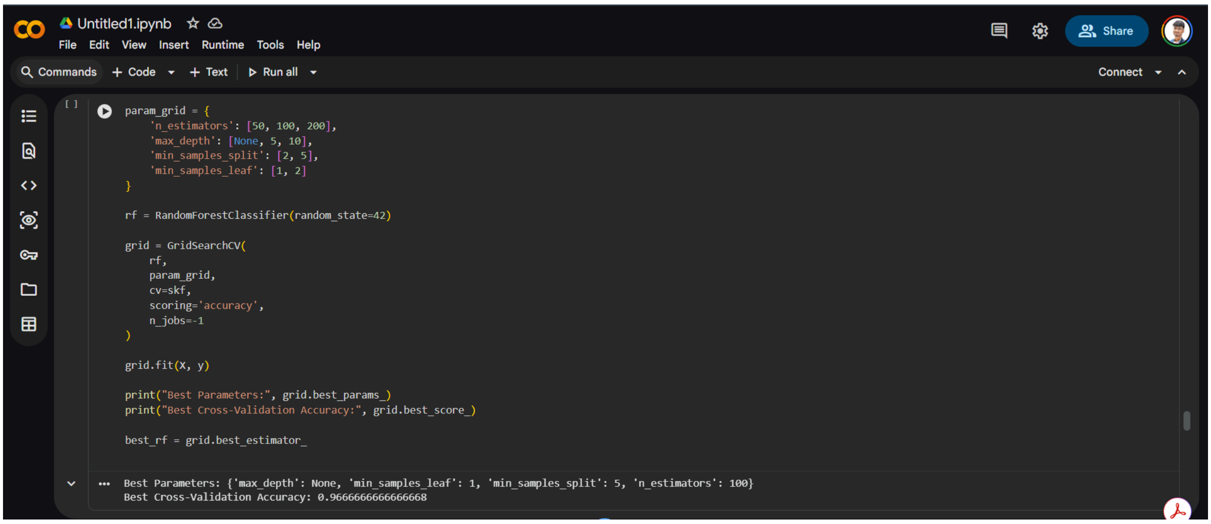
```
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

rf_skf = cross_val_score(rf, X, y, cv=skf, scoring='accuracy')
svm_skf = cross_val_score(svm, X, y, cv=skf, scoring='accuracy')
dt_skf = cross_val_score(dt, X, y, cv=skf, scoring='accuracy')

print("\nStratified K-Fold Cross-Validation Accuracy:")
print("Random Forest:", rf_skf.mean())
print("SVM:", svm_skf.mean())
print("Decision Tree:", dt_skf.mean())
```

```
Stratified K-Fold Cross-Validation Accuracy:
Random Forest: 0.9466666666666667
SVM: 0.9666666666666668
Decision Tree: 0.9533333333333335
```



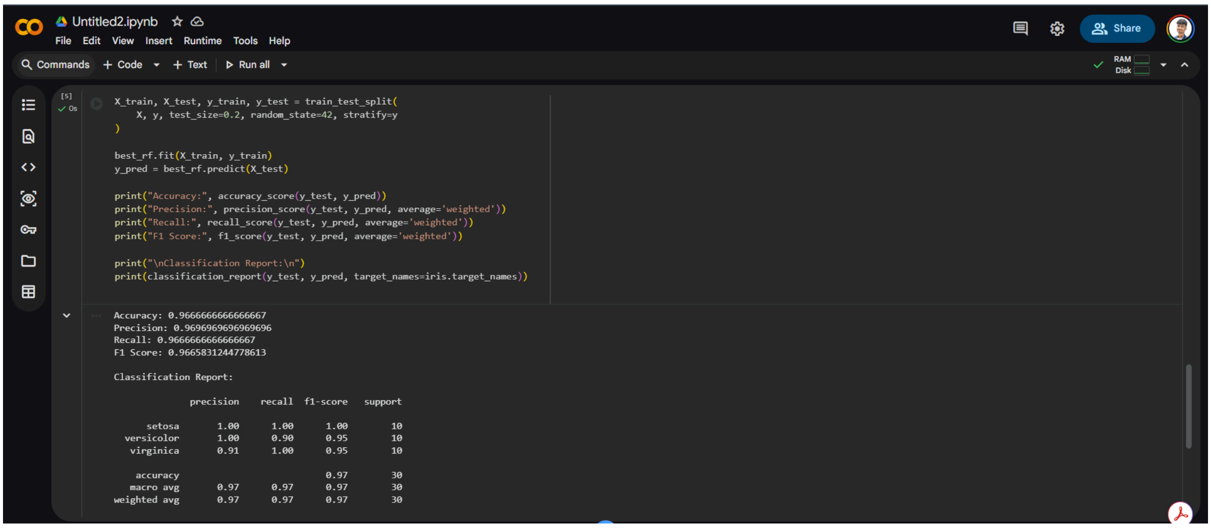
```
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

rf_skf = cross_val_score(rf, X, y, cv=skf, scoring='accuracy')
svm_skf = cross_val_score(svm, X, y, cv=skf, scoring='accuracy')
dt_skf = cross_val_score(dt, X, y, cv=skf, scoring='accuracy')

print("\nStratified K-Fold Cross-Validation Accuracy:")
print("Random Forest:", rf_skf.mean())
print("SVM:", svm_skf.mean())
print("Decision Tree:", dt_skf.mean())
```

```
Stratified K-Fold Cross-Validation Accuracy:
Random Forest: 0.9466666666666667
SVM: 0.9666666666666668
Decision Tree: 0.9533333333333335
```



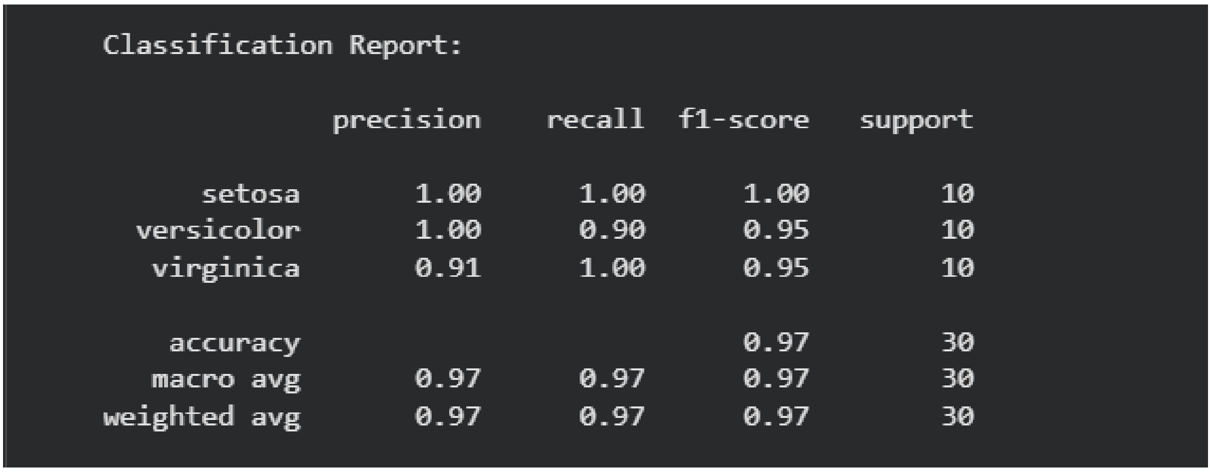
```
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 5, 10],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 2]
}

rf = RandomForestClassifier(random_state=42)

grid = GridSearchCV(
    rf,
    param_grid,
    cv=skf,
    scoring='accuracy',
    n_jobs=-1
)

grid.fit(X, y)

print("Best Parameters:", grid.best_params_)
print("Best Cross-Validation Accuracy:", grid.best_score_)

best_rf = grid.best_estimator_
```

```
Best Parameters: {'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 100}
Best Cross-Validation Accuracy: 0.9666666666666668
```



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

best_rf.fit(X_train, y_train)
y_pred = best_rf.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))
print("Precision:", precision_score(y_test, y_pred, average='weighted'))
print("Recall:", recall_score(y_test, y_pred, average='weighted'))
print("F1 Score:", f1_score(y_test, y_pred, average='weighted'))

print("\nClassification Report:\n")
print(classification_report(y_test, y_pred, target_names=iris.target_names))
```

```
Accuracy: 0.9666666666666667
Precision: 0.9696969696969696
Recall: 0.9666666666666667
F1 Score: 0.9665831244778613

Classification Report:

              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        10
  versicolor       1.00      0.90      0.95        10
   virginica       0.91      1.00      0.95        10

    accuracy                           0.97        30
   macro avg       0.97      0.97      0.97        30
weighted avg       0.97      0.97      0.97        30
```



```
Classification Report:

              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        10
  versicolor       1.00      0.90      0.95        10
   virginica       0.91      1.00      0.95        10

    accuracy                           0.97        30
   macro avg       0.97      0.97      0.97        30
weighted avg       0.97      0.97      0.97        30
```