

A Jupyter Notebook toolbar with icons for up, down, search, undo, redo, settings, and a full-screen view.

```
# Imports
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, accuracy_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Load dataset
iris = load_iris()
X = iris.data
y = iris.target
class_names = iris.target_names

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

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

# Define classifiers
classifiers = {
    "KNN": KNeighborsClassifier(n_neighbors=3),
    "Logistic Regression": LogisticRegression(),
    "Decision Tree": DecisionTreeClassifier(),
    "Random Forest": RandomForestClassifier(),
    "Naive Bayes": GaussianNB(),
```

```

    "SVM": SVC()
}

# Results storage
results = []

# Evaluate each classifier
for name, model in classifiers.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    report = classification_report(y_test, y_pred, output_dict=True)
    acc = accuracy_score(y_test, y_pred)
    conf_mat = confusion_matrix(y_test, y_pred)

    results.append({
        "Model": name,
        "Accuracy": acc,
        "Precision": report['weighted avg']['precision'],
        "Recall": report['weighted avg']['recall'],
        "F1-Score": report['weighted avg']['f1-score']
    })

# Print report
print(f"===== {name} =====")
print("Classification Report:")
print(classification_report(y_test, y_pred, target_names=class_names))
print("Confusion Matrix:")
print(conf_mat)
print()

# Convert results to DataFrame
df_results = pd.DataFrame(results)

# Show results table
print("=== Summary of Statistical Performance ===")

```

```
print(df_results)

# Plotting the performance comparison
plt.figure(figsize=(10, 6))
sns.barplot(data=df_results.melt(id_vars="Model"), x="Model", y="value", hue="variable")
plt.title("Classifier Performance Comparison on Iris Dataset")
plt.ylabel("Score")
plt.ylim(0, 1.1)
plt.grid(True, linestyle="--", alpha=0.5)
plt.show()
```

===== SVM =====

Classification Report:

	precision	recall	f1-score	support
setosa	1.00	1.00	1.00	19
versicolor	1.00	1.00	1.00	13
virginica	1.00	1.00	1.00	13
accuracy			1.00	45
macro avg	1.00	1.00	1.00	45
weighted avg	1.00	1.00	1.00	45

Confusion Matrix:

```
[[19  0  0]
 [ 0 13  0]
 [ 0  0 13]]
```

=== Summary of Statistical Performance ===

	Model	Accuracy	Precision	Recall	F1-Score
0	KNN	1.000000	1.000000	1.000000	1.000000
1	Logistic Regression	1.000000	1.000000	1.000000	1.000000
2	Decision Tree	1.000000	1.000000	1.000000	1.000000
3	Random Forest	1.000000	1.000000	1.000000	1.000000
4	Naive Bayes	0.977778	0.979365	0.977778	0.977745
5	SVM	1.000000	1.000000	1.000000	1.000000