Experiments 7/8/2025

STUDY THE CLASSIFIERS WITH RESPECT TO STATISTICAL PARAMETERS

Aim: To implement different classifiers on iris dataset

Objective:

→ To implement Decision tree and KNN classifier and Logistic Regression.

→ Train both the classifier on a dataset to compare the output.

→ To determine the better performing classifier for the given dataset.

Pseudocode:

→ Import the necessary libraries.

→ Load the iris dataset.

→ Preprocess and normalize the data

→ Split the dataset into training and testing data

→ Compare the classifiers using evaluation metrics after training and fitting the classifier on the training data

Observation

Model	Accuracy
KNN	1.0
Logistic Regression	1.0
Decision Tree	1.0

All three classifiers - KNN, Decision Tree and logistic Regression achieve similar accuracy on the iris dataset because the iris dataset is clean, balanced.

All three algorithms are capable of handling multi-class classification effectively.

Statistical Parameters are values that summarize or describe important characteristics of a dataset.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

Out of predicted positives how many were actually correct.

$$\text{Recall} = \frac{TP}{TP + FN}$$

Out of actual positives, how many were correctly predicted.

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Result: successfully implemented and different classifiers on iris dataset.

Decision Tree, KNN, Logistic Regression Comparison.

Model	Accuracy	Precision	Recall	F1-Score
KNN	1.00	1.00	1.00	1.00
Logistic Regression	1.00	1.00	1.00	1.00
Decision Tree	1.00	1.00	1.00	1.00

[3]:
```
!pip install scikit-learn
import pandas as pd
import numpy as np
from sklearn.datasets import load_iris
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

```
Defaulting to user installation because normal site-packages is not writeable
Collecting scikit-learn
  Downloading scikit_learn-1.7.1-cp310-cp310-manylinux2014_x86_64.manylinux_2_17_x86_64.whl.metadata (11 kB)
Requirement already satisfied: numpy>=1.22.0 in /home/jupyter-ra2311047010041/.local/lib/python3.10/site-pac
learn) (2.2.3)
Collecting scipy>=1.8.0 (from scikit-learn)
  Downloading scipy-1.15.3-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (61 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 62.0/62.0 kB 114.1 kB/s eta 0:00:00 0:00:01
Collecting joblib>=1.2.0 (from scikit-learn)
  Downloading joblib-1.5.1-py3-none-any.whl.metadata (5.6 kB)
Collecting threadpoolctl>=3.1.0 (from scikit-learn)
  Downloading threadpoolctl-3.6.0-py3-none-any.whl.metadata (13 kB)
Downloading scikit_learn-1.7.1-cp310-cp310-manylinux2014_x86_64.manylinux_2_17_x86_64.whl (9.7 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 9.7/9.7 MB 200.3 kB/s eta 0:00:0000:0100:02
Downloading joblib-1.5.1-py3-none-any.whl (307 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 307.7/307.7 kB 330.0 kB/s eta 0:00:00a 0:00:01
Downloading scipy-1.15.3-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (37.7 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 37.7/37.7 MB 205.5 kB/s eta 0:00:0000:0100:05
Downloading threadpoolctl-3.6.0-py3-none-any.whl (18 kB)
[notice] To update, run: pip install --upgrade pip
```

4]:
```
iris = load_iris()
X = iris.data
y = iris.target
```

1]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

2]:
```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

3]:
```
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
```

3]:
```
▾ KNeighborsClassifier
▸ Parameters
```

4]:
```
y_pred = knn.predict(X_test)
```

5]:
```
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 1.0
```

```
▸ Parameters
```

1]:
```
y_pred = knn.predict(X_test)
```

5]:
```
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 1.0
```

```
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 1.0

Confusion Matrix:
 [[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       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
```

In [18]:
```
from sklearn.tree import DecisionTreeClassifier

model = DecisionTreeClassifier()
model.fit(X_train, y_train)
y_pred_dt = model.predict(X_test)
```

In [19]:
```
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 1.0

Confusion Matrix:
 [[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       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
```

In [20]:
```
# No scaling
knn_no_scaling = KNeighborsClassifier(n_neighbors=3)
knn_no_scaling.fit(X_train, y_train)
y_pred_no_scaling = knn_no_scaling.predict(X_test)
```

```
Accuracy WITHOUT normalization: 1.0
Confusion Matrix:
 [[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```

In [5]:

```python
# Step 1: Import required libraries
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Step 2: Load the Iris dataset
iris = load_iris()
X = iris.data        # Features
y = iris.target      # Target labels

# Step 3: Split into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 4: Train Logistic Regression model
logreg = LogisticRegression(max_iter=200)
logreg.fit(X_train, y_train)

# Step 5: Predict on test data
y_pred_logreg = logreg.predict(X_test)

# Step 6: Evaluate the model
print("✅ Accuracy:", accuracy_score(y_test, y_pred))
print("\n📊 Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\n📄 Classification Report:\n", classification_report(y_test, y_pred, target_names=iris.target_names
```

```
✅ Accuracy: 1.0
```

```
📊 Confusion Matrix:
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]

📝 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
```

In [21]:

```python
from sklearn.metrics import accuracy_score
import pandas as pd

# Assuming you already have predictions from each model:
# y_pred_knn, y_pred_logreg, y_pred_dt, y_pred_rf, y_pred_svm

results = [
    {"Model": "KNN", "Accuracy": accuracy_score(y_test, y_pred_knn)},
    {"Model": "Logistic Regression", "Accuracy": accuracy_score(y_test, y_pred_logreg)},
    {"Model": "Decision Tree", "Accuracy": accuracy_score(y_test, y_pred_dt)},

]
# Create DataFrame to show results
results_df = pd.DataFrame(results)
print(results_df.sort_values(by="Accuracy", ascending=False))
```

```
                 Model  Accuracy
0                  KNN       1.0
```