Training samples: 105, Testing sample: 45

Model Performance Comparison:

Model	Accuracy	Precision	Recall	F1 score
Random Forest	1.000	1.00	1	1
Decision Tree	~~0.955~~ 1	~~0.9511~~	1	1
Naive Bayes	0.955	0.971	0.976	0.97

Confusion matrix for random forest

$$\begin{bmatrix} 19 & 0 & 0 \\ 0 & 13 & 0 \\ 0 & 0 & 13 \end{bmatrix}$$

Confusion matrix for decision Tree

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

Confusion matrix for Naive Bayes:

[19 0 0
0 12 1
0 0 13]

Lab 3 Study of classifiers with respect to statistical parameters.

06-08-25

To study the performance of different classifiers using statistical parameters like accuracy, precision, recall and F1 score

Pseudocode:

1. Import required libraries
pandas, scikit learn, metrics train test split
2. Load the data set
split ~~into features and~~ data into train (70%) and test (30%) sets
3. Define 3 models: Random Forest, Decision Tree, Naive Bayes
4. For each model:
Train on training data
Predict test data
Calculate metrics: Accuracy, Precision (macro), Recall (macro & weighted), F1 mac
Display confusion matrix and confusion matrices.

OBSERVATIONS

Random Forest gives highest accuracy and best metrics

Decision Tree and Naive Bayes have slightly lower but comparable performance

Confusion matrices show most misclassification happen b/w similar classes.

Macro and weighted recalls are similar due to balanced dataset.

```
x_train = x_train / 255.0
x_test = x_test / 255.0

# One-hot encode labels
y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)

model = Sequential([
    Flatten(input_shape=(28, 28)),        # Flatten 28x28 image → 784 vector
    Dense(128, activation='relu'),        # Hidden layer with ReLU
    Dense(64, activation='relu'),         # Second hidden layer
    Dense(10, activation='softmax')       # Output layer (10 classes)
])

model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5, batch_size=32, validation_split=0.1)

test_loss, test_acc = model.evaluate(x_test, y_test)
print(f"Test Accuracy: {test_acc*100:.2f}%")

import numpy as np

predictions = model.predict(x_test)
print("Predicted Label:", np.argmax(predictions[0]))

```