```
# Use the Optical Recognition of Handwritten Disease database:
# 1. Apply classifiers such as Naive Bayes, KNN, Random Forest, and Decision Tree.
# 2. List the performance of these classifiers in tabular form, including:
# Precision, Recall, F1 Score, Accuracy
# 3. Compare the F1 Score to justify the best classifier for the problem

import numpy as np
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

# Paths to the data files
train_file_path = 'C:/Users/HP/OneDrive/Desktop/ml 7th sem codes/datasets/optical/optdigits.tra'
test_file_path = 'C:/Users/HP/OneDrive/Desktop/ml 7th sem codes/datasets/optical/optdigits.tes'

# Load the training and testing data
train_data = pd.read_csv(train_file_path, header=None)
test_data = pd.read_csv(test_file_path, header=None)

# Separate features and labels
X_train = train_data.iloc[:, :-1].values  # All columns except the last one
y_train = train_data.iloc[:, -1].values   # The last column as labels

X_test = test_data.iloc[:, :-1].values    # All columns except the last one
y_test = test_data.iloc[:, -1].values     # The last column as labels

# Dictionary to store model names and their accuracies
model_accuracies = {}
```

```
# Applying classifiers
# Naive Bayes
nb_model = GaussianNB()
nb_model.fit(X_train, y_train)
y_pred_nb = nb_model.predict(X_test)
model_accuracies['Naive Bayes'] = accuracy_score(y_test, y_pred_nb)

# K-Nearest Neighbors
knn_model = KNeighborsClassifier()
knn_model.fit(X_train, y_train)
y_pred_knn = knn_model.predict(X_test)
model_accuracies['KNN'] = accuracy_score(y_test, y_pred_knn)

# Decision Tree
dt_model = DecisionTreeClassifier()
dt_model.fit(X_train, y_train)
y_pred_dt = dt_model.predict(X_test)
model_accuracies['Decision Tree'] = accuracy_score(y_test, y_pred_dt)

# Random Forest
rf_model = RandomForestClassifier()
rf_model.fit(X_train, y_train)
y_pred_rf = rf_model.predict(X_test)
model_accuracies['Random Forest'] = accuracy_score(y_test, y_pred_rf)

# Generate classification reports and compare
reports = {
    'Naive Bayes': classification_report(y_test, y_pred_nb, output_dict=True),
    'KNN': classification_report(y_test, y_pred_knn, output_dict=True),
    'Decision Tree': classification_report(y_test, y_pred_dt, output_dict=True),
```

```
    'Random Forest': classification_report(y_test, y_pred_rf, output_dict=True),
}

# Display results in tabular form
results_table = pd.DataFrame({
    'Classifier': list(reports.keys()),
    'Precision': [r['weighted avg']['precision'] for r in reports.values()],
    'Recall': [r['weighted avg']['recall'] for r in reports.values()],
    'F1 Score': [r['weighted avg']['f1-score'] for r in reports.values()],
    'Accuracy': list(model_accuracies.values())
})

print(results_table)

# Compare F1 Scores
best_classifier = results_table.loc[results_table['F1 Score'].idxmax()]
print("\nBest Classifier based on F1 Score:")
print(best_classifier)
```