Name- Adarsh Upadhyay

Reg No. - 23MCA0237

The Wheat Seeds Dataset involves the prediction of species given measurements of seeds from different varieties of wheat.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, confusion_matrix,
precision_score, recall_score, f1_score

# Load dataset
dataset = pd.read_csv("/content/wheat-seeds.csv")
X = dataset.iloc[:, :7]
y = dataset.iloc[:, 7]

# Split dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)

# Standardize features
sc = StandardScaler()
X_train_std = sc.fit_transform(X_train)
X_test_std = sc.transform(X_test)

def classifiers():
  classifiers = {
    'SVM (Linear)': SVC(kernel='linear', random_state=42),
    'SVM (Poly)': SVC(kernel='poly', random_state=42),
    'SVM (RBF)': SVC(kernel='rbf', random_state=42),
    'SVM (Sigmoid)': SVC(kernel='sigmoid', random_state=42),
    'KNN': KNeighborsClassifier(),
    'Decision Tree': DecisionTreeClassifier(random_state=42),
```

```
    'MLPC': MLPClassifier( hidden_layer_sizes=(100,50),max_iter=1000,
random_state=42),
    'Naive Bayes': GaussianNB()
  }
  return classifiers

def predictMatrix(classifiers):
  # Store metrics for each classifier
  metrics = {
    'Accuracy': [],
    'Precision': [],
    'Recall': [],
    'F1 Score': []
  }

  for name, clf in classifiers.items():
    # Train classifier
    clf.fit(X_train_std, y_train)

    # Predict
    y_pred = clf.predict(X_test_std)

    # Calculate metrics
    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred, average='weighted')
    recall = recall_score(y_test, y_pred, average='weighted')
    f1 = f1_score(y_test, y_pred, average='weighted')

    metrics['Accuracy'].append(accuracy)
    metrics['Precision'].append(precision)
    metrics['Recall'].append(recall)
    metrics['F1 Score'].append(f1)

  return metrics

def printMetrics(classifiers, metrics):
  # Print metrics
  for metric_name, metric_values in metrics.items():
    print(f"{metric_name}:")
    for classifier_name, value in zip(classifiers.keys(),
metric_values):
        print(f"{classifier_name}: {value:.4f}")
    print()

def plotMetrics(classifiers, metrics):
  # Plotting
  plt.figure(figsize=(12, 8))
  bar_width = 0.2
  index = np.arange(len(classifiers))
  for i, (metric_name, metric_values) in enumerate(metrics.items()):
```

```
    plt.bar(index + i * bar_width, metric_values, bar_width,
label=metric_name)

  plt.xlabel('Classifiers')
  plt.ylabel('Score')
  plt.title('Classifier Performance Comparison')
  plt.xticks(index + bar_width * (len(metrics) - 1) / 2,
classifiers.keys(), rotation=45, ha='right')
  plt.legend()
  plt.show()

classifiers = classifiers()
metrics = predictMatrix(classifiers)

printMetrics(classifiers, metrics)
```

```
Accuracy:
SVM (Linear): 0.9206
SVM (Poly): 0.8095
SVM (RBF): 0.9365
SVM (Sigmoid): 0.9206
KNN: 0.9048
Decision Tree: 0.8889
MLPC: 0.9365
Naive Bayes: 0.9048

Precision:
SVM (Linear): 0.9250
SVM (Poly): 0.8833
SVM (RBF): 0.9385
SVM (Sigmoid): 0.9250
KNN: 0.9067
Decision Tree: 0.8940
MLPC: 0.9459
Naive Bayes: 0.9067

Recall:
SVM (Linear): 0.9206
SVM (Poly): 0.8095
SVM (RBF): 0.9365
SVM (Sigmoid): 0.9206
KNN: 0.9048
Decision Tree: 0.8889
MLPC: 0.9365
Naive Bayes: 0.9048

F1 Score:
SVM (Linear): 0.9197
SVM (Poly): 0.8157
SVM (RBF): 0.9361
```

```
SVM (Sigmoid): 0.9197
KNN: 0.9054
Decision Tree: 0.8904
MLPC: 0.9353
Naive Bayes: 0.9054


plotMetrics(classifiers, metrics)
```