Programme 5: Classification with support vector machines (SVM)

Problem Statement

The goal of this case study is to predict the quality of red wine based on various physicochemical properties using a Support Vector Machine (SVM). The dataset winequality_red.csv contains several features such as fixed acidity, volatile acidity, citric acid, and more, as well as a target variable indicating the quality rating of the wine on a scale from 0 to 10. Accurate prediction of wine quality can be valuable for wine producers and consumers by helping to assess and ensure quality standards.

```
import pandas as pd  # Import the pandas library for data manipulation

from sklearn.model_selection import train_test_split  # Import function to split data into training and testing sets

from sklearn.preprocessing import StandardScaler  # Import StandardScaler to standardize features

from sklearn.svm import SVC  # Import Support Vector Classifier for SVM modeling

from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score  # Import various metrics to evaluate the model


# Load the dataset

data = pd.read_csv('winequality_red.csv')  # Read the dataset from a CSV file into a DataFrame


# Display the details of the dataset

print(data.info())  # Print summary information about the DataFrame, including the data types and non-null counts of each column


# Features and target variable

X = data.drop('quality', axis=1)  # Drop the 'quality' column to create the feature set (X) which contains all columns except 'quality'

y = data['quality']  # Create the target variable (y) which contains only the 'quality' column


# Preprocessing: Standardize the features

scaler = StandardScaler()  # Create an instance of StandardScaler

X_scaled = scaler.fit_transform(X)  # Fit the scaler to the feature data and transform it, standardizing the features


# Split the data into training and testing sets
```

```
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3, random_state=42)  # Split the data into training and testing sets with 30% of the data used for testing and a fixed random seed for reproducibility


# Initialize and train the SVM model with balanced class weights

model = SVC(kernel='rbf', gamma='scale', C=1.0, class_weight='balanced', probability=True)  # Create an instance of SVC with radial basis function kernel, automatic gamma scaling, regularization parameter C set to 1.0, balanced class weights, and probability estimates enabled

model.fit(X_train, y_train)  # Train the SVM model using the training data


# Make predictions on the test set

y_pred = model.predict(X_test)  # Use the trained model to predict the labels for the test set


# Calculate and print accuracy

accuracy = accuracy_score(y_test, y_pred)  # Calculate the accuracy of the predictions

print(f"\nAccuracy: {accuracy:.4f}")  # Print the accuracy with 4 decimal places


print("\nConfusion Matrix:")

cm = confusion_matrix(y_test, y_pred)  # Compute the confusion matrix to evaluate the performance of the classification

print(cm)  # Print the confusion matrix


precision = precision_score(y_test, y_pred, average=None)  # Calculate precision scores for each class, without averaging

recall = recall_score(y_test, y_pred, average=None)  # Calculate recall scores for each class, without averaging

f1 = f1_score(y_test, y_pred, average=None)  # Calculate F1 scores for each class, without averaging


print("\nPrecision for each class:")

for i, p in enumerate(precision):  # Iterate through each class's precision score

    print(f"Class {i}: {p:.4f}")  # Print the precision score for each class with 4 decimal places


print("\nRecall for each class:")

for i, r in enumerate(recall):  # Iterate through each class's recall score

    print(f"Class {i}: {r:.4f}")  # Print the recall score for each class with 4 decimal places
```

```
print("\nF1 Score for each class:")
for i, f in enumerate(f1):  # Iterate through each class's F1 score
    print(f"Class {i}: {f:.4f}")  # Print the F1 score for each class with 4 decimal places
```