SVM - One-Versus-One (OVO)

SVM - One-Versus-One (OVO)

Théorie

Les **machines à vecteurs de support (SVM)** sont des modèles de classification supervisés qui cherchent à maximiser la marge de séparation entre les classes. Pour un problème **multi-classe**, l'approche **One-Versus-One (OVO)** entraîne un SVM pour chaque paire de classes, ce qui permet une meilleure séparation lorsque les classes sont bien distinctes.

Hyperparamètres

Nous allons tester un seul hyperparamètre pour réduire le temps d'entraînement : - **Paramètre de régularisation (`C`)** : contrôle la pénalisation des erreurs de classification (valeurs entre `0.1` et `1`).

Exemple en Python

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.svm import SVC
from sklearn.multiclass import OneVsOneClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Chargement des ensembles de données
train_data = pd.read_csv('covertype_train.csv')
val_data = pd.read_csv('covertype_val.csv')
```

```
test_data = pd.read_csv('covertype_test.csv')

# Préparation des données
X_train = train_data.drop('Cover_Type', axis=1)
y_train = train_data['Cover_Type']

X_val = val_data.drop('Cover_Type', axis=1)
y_val = val_data['Cover_Type']

X_test = test_data.drop('Cover_Type', axis=1)
y_test = test_data['Cover_Type']

# Standardisation des données
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_val = scaler.transform(X_val)
X_test = scaler.transform(X_test)

# Recherche du meilleur hyperparamètre (C seulement)
C_values = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]  # Entre 0.1 et 1
train_accuracies = []
val_accuracies = []

for C in C_values:
    model = OneVsOneClassifier(SVC(kernel='rbf', C=C))
    model.fit(X_train, y_train)

    y_train_pred = model.predict(X_train)
    y_val_pred = model.predict(X_val)

    train_accuracies.append(accuracy_score(y_train, y_train_pred))
    val_accuracies.append(accuracy_score(y_val, y_val_pred))

# Sélection du meilleur hyperparamètre
best_C = C_values[val_accuracies.index(max(val_accuracies))]
print(f"Meilleur hyperparamètre : C={best_C}")

# Affichage du graphique
plt.figure(figsize=(8, 6))
plt.plot(C_values, train_accuracies, marker='o', linestyle='dashed', label='Train Accuracy')
plt.plot(C_values, val_accuracies, marker='s', linestyle='dashed', label='Validation Accuracy
plt.xlabel("Paramètre de régularisation (C)")
```

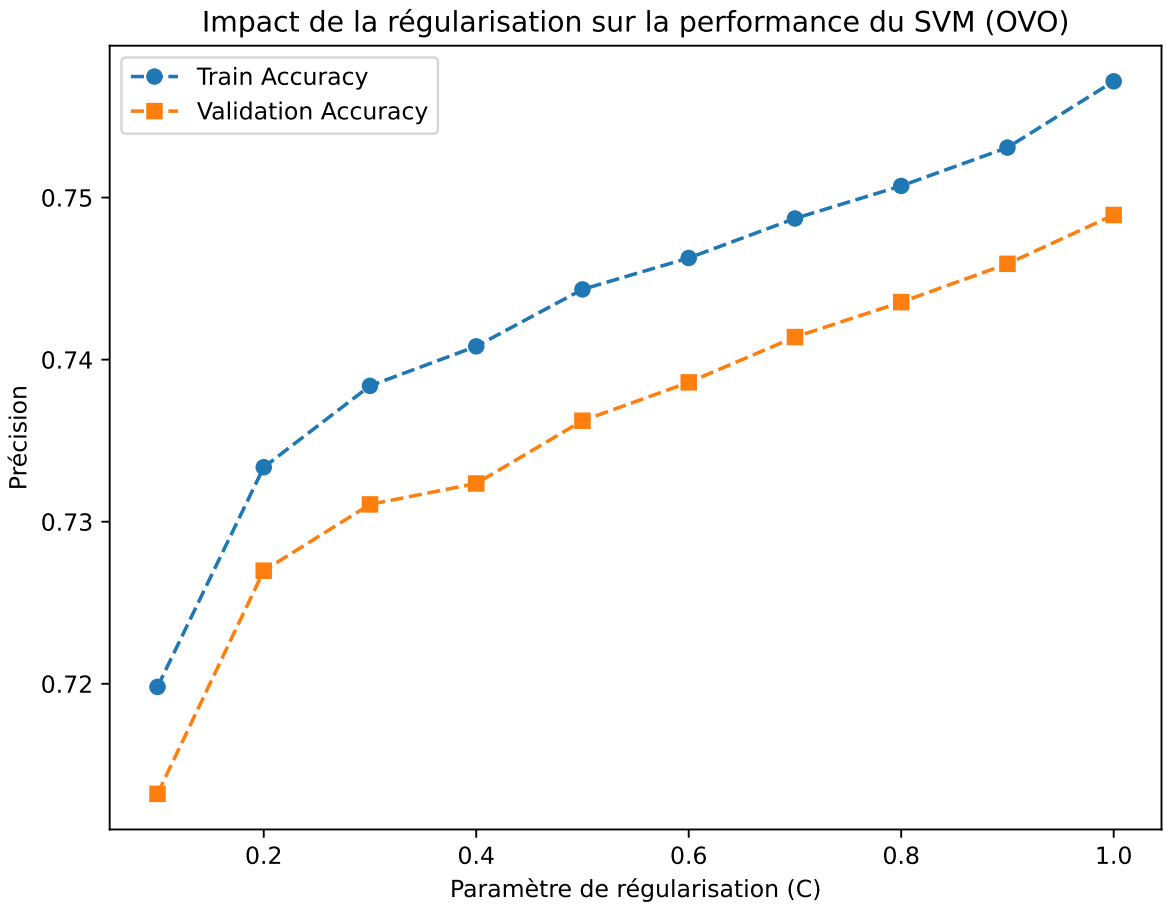
```
plt.ylabel("Précision")
plt.title("Impact de la régularisation sur la performance du SVM (OVO)")
plt.legend()
plt.show()

# Modèle final avec le meilleur hyperparamètre
final_model = OneVsOneClassifier(SVC(kernel='rbf', C=best_C))
final_model.fit(X_train, y_train)
y_test_pred = final_model.predict(X_test)

# Affichage de la matrice de confusion
conf_matrix = confusion_matrix(y_test, y_test_pred)
print("\nMatrice de confusion :")
print(conf_matrix)

print("\nÉvaluation sur l'ensemble de test")
print(classification_report(y_test, y_test_pred))
```

```
Meilleur hyperparamètre : C=1
```




```
Matrice de confusion :
[[1223  454    0    0    0    0   31]
 [ 372 1834   42    0    0    9    4]
 [   0   23  252    0    0    6    0]
 [   0    0   18    0    0    3    0]
 [   4   64    6    0    0    0    0]
 [   0   39   98    0    0    7    0]
 [  56    0    0    0    0    0  103]]

Évaluation sur l'ensemble de test
              precision    recall  f1-score   support

           1       0.74      0.72      0.73      1708
           2       0.76      0.81      0.78      2261
           3       0.61      0.90      0.72       281
```

```
           4       0.00      0.00      0.00        21
           5       0.00      0.00      0.00        74
           6       0.28      0.05      0.08       144
           7       0.75      0.65      0.69       159

    accuracy                           0.74      4648
   macro avg       0.45      0.45      0.43      4648
weighted avg       0.71      0.74      0.72      4648


/home/ensai/.local/lib/python3.10/site-packages/sklearn/metrics/_classification.py:1531: Unde
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/home/ensai/.local/lib/python3.10/site-packages/sklearn/metrics/_classification.py:1531: Unde
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/home/ensai/.local/lib/python3.10/site-packages/sklearn/metrics/_classification.py:1531: Unde
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```