Arbre de Décision - CART

Théorie

L'algorithme **CART** (**Classification and Regression Trees**) est un modèle d'apprentissage supervisé qui construit un **arbre de décision** en divisant l'espace des caractéristiques en sous-ensembles homogènes.

Hyperparamètre utilisé

Nous allons optimiser :

- **Profondeur maximale de l'arbre (`max_depth`)** : sélectionnée en fonction de la précision sur l'ensemble de validation.

Métriques d'évaluation

Nous afficherons :

- **Matrice de confusion** : récapitulant les erreurs de classification.
- **Taux de bien classés sur l'échantillon de validation** avec le meilleur hyperparamètre.
- **Taux de bien classés sur l'échantillon de test** avec ce même hyperparamètre.
- **Taux de bien classés par classe sur l'échantillon de test**.

Recherche de la meilleure `max_depth` et évaluation

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, accuracy_score
import warnings

#  Suppression des avertissements inutiles
warnings.filterwarnings("ignore", category=UserWarning)

#  Chargement des ensembles de données
```

```
train_data = pd.read_csv('../data/covertype_train.csv')
val_data = pd.read_csv('../data/covertype_val.csv')
test_data = pd.read_csv('../data/covertype_test.csv')

#  Préparation des données
X_train, y_train = train_data.drop('Cover_Type', axis=1),
 ↪  train_data['Cover_Type']
X_val, y_val = val_data.drop('Cover_Type', axis=1),
 ↪  val_data['Cover_Type']
X_test, y_test = test_data.drop('Cover_Type', axis=1),
 ↪  test_data['Cover_Type']

#  Recherche du meilleur hyperparamètre max_depth
depth_range = range(1, 30)
val_accuracies = []

for depth in depth_range:
    tree = DecisionTreeClassifier(max_depth=depth,
 ↪  random_state=42)
    tree.fit(X_train, y_train)
    acc = accuracy_score(y_val, tree.predict(X_val))
    val_accuracies.append((depth, acc))

# Sélection de la meilleure profondeur d'arbre
best_depth, best_val_acc = max(val_accuracies, key=lambda x:
 ↪  x[1])


#  Affichage du graphique
plt.figure(figsize=(8, 6))
plt.plot(depth_range, [acc for depth, acc in val_accuracies],
 ↪  marker='o', linestyle='dashed', label="Validation")
plt.xlabel("Profondeur de l'arbre")
plt.ylabel("Précision sur validation")
plt.title("Impact de la profondeur de l'arbre sur la performance
 ↪  de CART")
plt.legend()
plt.show()

#  Modèle final avec la meilleure profondeur
final_model = DecisionTreeClassifier(max_depth=best_depth,
 ↪  random_state=42)
final_model.fit(X_train, y_train)
y_test_pred = final_model.predict(X_test)

#  Matrice de confusion
```

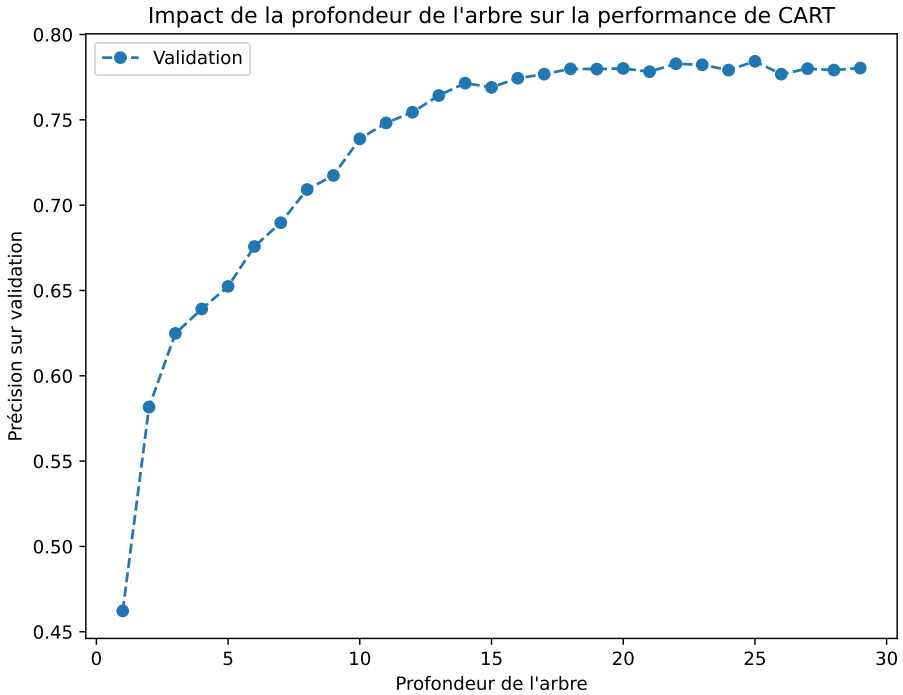
```python
conf_matrix = confusion_matrix(y_test, y_test_pred)

#  Calcul des taux de bien classés par classe
class_accuracies = conf_matrix.diagonal() /
 ↪  conf_matrix.sum(axis=1)
overall_test_accuracy = accuracy_score(y_test, y_test_pred)

#  Affichage des résultats
print(f"\n  Meilleure profondeur de l'arbre sur l'échantillon de
 ↪  validation : {best_depth}")
print(f"Taux de bien classés sur l'échantillon de validation avec
 ↪  cet hyperparamètre : {best_val_acc:.2%}")
print("\n  Matrice de confusion sur l'échantillon de test, avec le
 ↪  meilleur hyperparamètre :")
print(conf_matrix)

print("\n  Taux de bien classés par classe sur l'échantillon de
 ↪  test, avec le meilleur hyperparamètre  :")
for i, acc in enumerate(class_accuracies, start=1):
    print(f"Classe {i} : {acc:.2%}")

print(f"\n  Taux de bien classés sur l'échantillon de test avec le
 ↪  meilleur hyperparamètre : {overall_test_accuracy:.2%}")
```



```
 Meilleure profondeur de l'arbre sur l'échantillon de validation : 25
Taux de bien classés sur l'échantillon de validation avec cet hyperparamètre : 78.43%

 Matrice de confusion sur l'échantillon de test, avec le meilleur hyperparamètre :
[[1581  449    1    0   11    1   76]
 [ 459 2177   43    1   93   52    8]
 [   2   46 1216   24   13  129    0]
 [   0    0   23   74    0   13    0]
 [  10   83   13    0  263   11    0]
 [   4   40  128    4    4  514    0]
 [  61   14    0    0    0    0  746]]

 Taux de bien classés par classe sur l'échantillon de test, avec le meilleur hyperparamètre
Classe 1 : 74.61%
Classe 2 : 76.84%
Classe 3 : 85.03%
Classe 4 : 67.27%
Classe 5 : 69.21%
Classe 6 : 74.06%
Classe 7 : 90.86%

 Taux de bien classés sur l'échantillon de test avec le meilleur hyperparamètre : 78.35%
```