Random Forest - Forêt Aléatoire

Random Forest (Forêt Aléatoire)

Théorie

La forêt aléatoire est un algorithme d'apprentissage supervisé basé sur un ensemble d'arbres de décision. Elle fonctionne en combinant plusieurs arbres pour améliorer la précision et réduire le risque de surapprentissage.

Évaluation des performances

Lorsqu'on évalue un modèle de classification, plusieurs métriques sont utilisées :

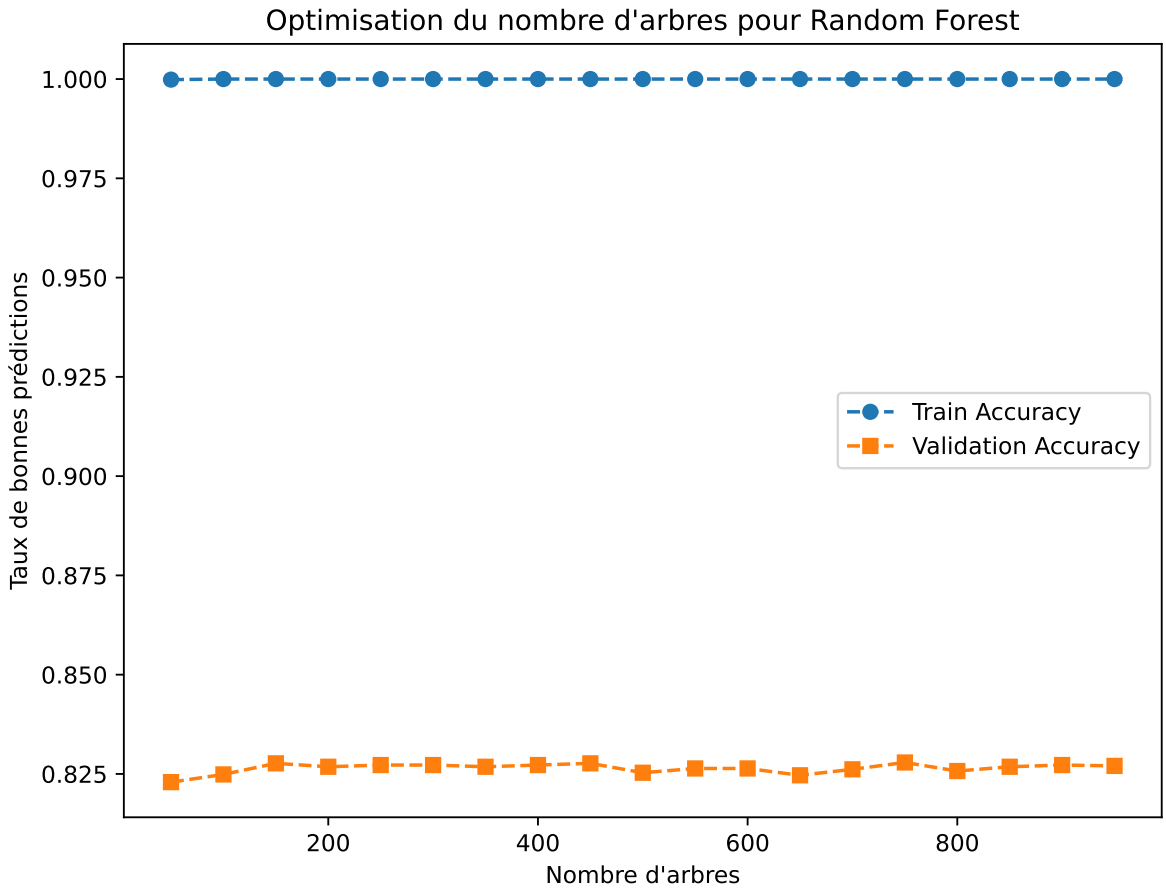
- **Matrice de confusion** : Tableau qui résume les performances du modèle en comparant les vraies classes aux classes prédites. Les lignes correspondent aux **classes réelles**, et les colonnes aux **classes prédites**.
- **Accuracy (Précision globale)** : Proportion des prédictions correctes parmi l'ensemble des données.
- **Precision (Précision par classe)** : Nombre de vrais positifs divisé par la somme des vrais positifs et des faux positifs. Indique la fiabilité des prédictions positives.
- **Recall (Rappel)** : Nombre de vrais positifs divisé par la somme des vrais positifs et des faux négatifs. Indique la capacité du modèle à détecter les échantillons positifs.
- **F1-score** : Moyenne harmonique entre précision et rappel, utile lorsque les classes sont déséquilibrées.

Exemple en Python

```python
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.model_selection import train_test_split

# Chargement des ensembles de données déjà préparés
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']

# Recherche du meilleur nombre d'arbres en utilisant l'ensemble de validation
n_estimators_range = range(50, 1000, 50)
train_accuracies = []
val_accuracies = []

for n in n_estimators_range:
    rf = RandomForestClassifier(n_estimators=n, random_state=42, n_jobs=-1)
    rf.fit(X_train, y_train)

    # Évaluation sur l'ensemble d'entraînement
    y_train_pred = rf.predict(X_train)
    train_accuracies.append(accuracy_score(y_train, y_train_pred))

    # Évaluation sur l'ensemble de validation
    y_val_pred = rf.predict(X_val)
    val_accuracies.append(accuracy_score(y_val, y_val_pred))

# Sélection du meilleur hyperparamètre basé sur l'ensemble de validation
best_n = n_estimators_range[val_accuracies.index(max(val_accuracies))]
print(f"Meilleur nombre d'arbres: {best_n}")
```

```
# Affichage du graphique comparant l'entraînement et la validation
plt.figure(figsize=(8, 6))
plt.plot(n_estimators_range, train_accuracies, marker='o', linestyle='dashed', label='Train A
plt.plot(n_estimators_range, val_accuracies, marker='s', linestyle='dashed', label='Validatio
plt.xlabel("Nombre d'arbres")
plt.ylabel("Taux de bonnes prédictions")
plt.title("Optimisation du nombre d'arbres pour Random Forest")
plt.legend()
plt.show()

# Modèle final avec le meilleur nombre d'arbres
rf = RandomForestClassifier(n_estimators=best_n, random_state=42, n_jobs=-1)
rf.fit(X_train, y_train)

# Évaluation sur l'ensemble de test
y_test_pred = rf.predict(X_test)

# Affichage de la matrice de confusion avec annotations
conf_matrix = confusion_matrix(y_test, y_test_pred)
print("\nMatrice de confusion (les lignes représentent les vraies classes et les colonnes les
print(conf_matrix)

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

```
Meilleur nombre d'arbres: 750
```




```
Matrice de confusion (les lignes représentent les vraies classes et les colonnes les classes
[[1400  300    0    0    1    0    7]
 [ 235 1992   17    0    4   13    0]
 [   0   21  241    1    0   18    0]
 [   0    0    8   11    0    2    0]
 [   3   45    1    0   25    0    0]
 [   0   21   44    1    1   77    0]
 [  35    2    0    0    0    0  122]]

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

           1       0.84      0.82      0.83      1708
           2       0.84      0.88      0.86      2261
           3       0.77      0.86      0.81       281
```

4	0.85	0.52	0.65	21
5	0.81	0.34	0.48	74
6	0.70	0.53	0.61	144
7	0.95	0.77	0.85	159
accuracy			0.83	4648
macro avg	0.82	0.67	0.73	4648
weighted avg	0.83	0.83	0.83	4648