

Random Forest - Forêt Aléatoire

Théorie

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

Hyperparamètre utilisé

Nous allons optimiser :

- **Nombre d'arbres (`n_estimators`)** : sélectionné 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 du meilleur `n_estimators` et évaluation

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier
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/covertypes_train.csv')
val_data = pd.read_csv('../data/covertypes_val.csv')
test_data = pd.read_csv('../data/covertypes_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 n_estimators
n_estimators_range = range(50, 1000, 50)
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)
    acc = accuracy_score(y_val, rf.predict(X_val))
    val_accuracies.append((n, acc))

# Sélection du meilleur nombre d'arbres
best_n, best_val_acc = max(val_accuracies, key=lambda x: x[1])

# Affichage du graphique
plt.figure(figsize=(8, 6))
plt.plot(n_estimators_range, [acc for n, acc in val_accuracies],
    ↪ marker='o', linestyle='dashed', label="Validation")
plt.xlabel("Nombre d'arbres")
plt.ylabel("Précision sur validation")
plt.title("Impact du nombre d'arbres sur la performance de Random
    ↪ Forest")
plt.legend()
plt.show()

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

# Matrice de confusion
conf_matrix = confusion_matrix(y_test, y_test_pred)

```

```

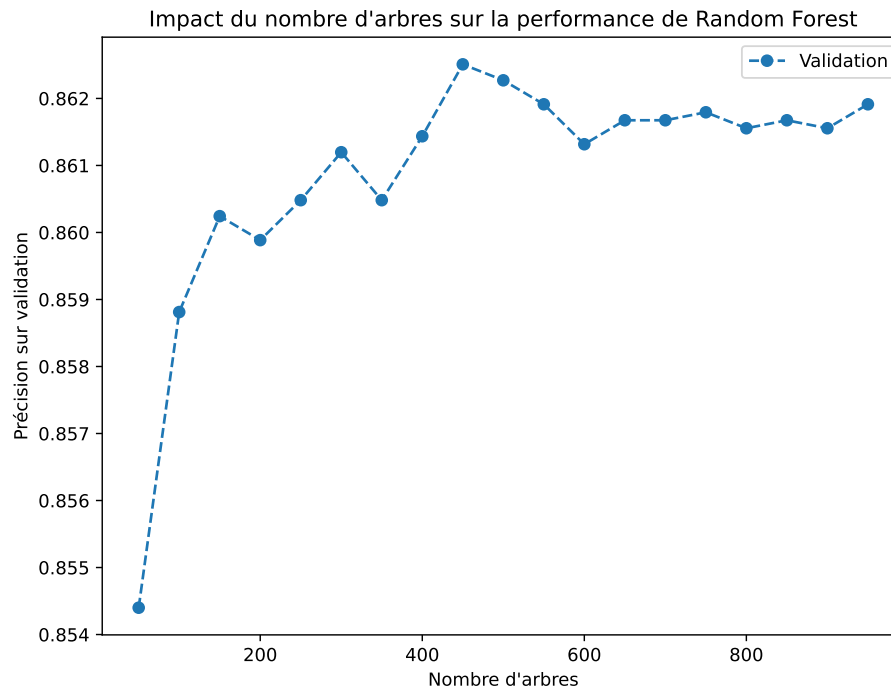
# Calcul des taux de bien classés par classe
class_accuracies = conf_matrix.diagonal() /
    ↪ conf_matrix.sum(axis=1)
overall_test_accuracy = accuracy_score(y_test, y_test_pred)

# Affichage des résultats
print(f"\n Meilleur nombre d'arbres sur l'échantillon de
    ↪ validation : {best_n}")
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%}")

```



Meilleur nombre d'arbres sur l'échantillon de validation : 450
Taux de bien classés sur l'échantillon de validation avec cet hyperparamètre : 86.25%

Matrice de confusion sur l'échantillon de test, avec le meilleur hyperparamètre :

[1753	312	0	0	5	2	47]
[247	2490	40	1	20	31	4]
[0	15	1352	11	1	51	0]
[0	0	37	71	0	2	0]
[3	86	16	0	273	2	0]
[1	17	128	2	0	546	0]
[43	4	0	0	0	0	774]]

Taux de bien classés par classe sur l'échantillon de test, avec le meilleur hyperparamètre

Classe 1	: 82.73%
Classe 2	: 87.89%
Classe 3	: 94.55%
Classe 4	: 64.55%
Classe 5	: 71.84%
Classe 6	: 78.67%
Classe 7	: 94.28%

Taux de bien classés sur l'échantillon de test avec le meilleur hyperparamètre : 86.55%