TP : Construire une Forêt Aléatoire en Python

🎯 Objectif

Ce TP a pour but de découvrir le fonctionnement d'un modèle **Random Forest** (forêt aléatoire) en Python à l'aide de scikit-learn. Il s'appuie sur :

- Le jeu de données Iris
- Un modèle d'ensemble basé sur plusieurs arbres de décision
- L'utilisation de l'indice de Gini comme critère

Rappel : Qu'est-ce qu'une forêt aléatoire ?

Une **forêt aléatoire** est un ensemble de plusieurs arbres de décision.

Chaque arbre :

- est entraîné sur un sous-échantillon différent (tiré avec remise = *bootstrap*)
- sélectionne un sous-ensemble aléatoire d'attributs à chaque division

La décision finale est prise par **vote majoritaire** (en classification).

1. Installation & Imports

```
pip install scikit-learn matplotlib
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
```

2. Chargement des données Iris

```
iris = load_iris()
X = iris.data
y = iris.target

print("Caractéristiques :", iris.feature_names)
print("Classes :", iris.target_names)
```

3. Séparer en données d'entraînement et de test

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
```

4. Création du modèle Random Forest

```
clf = RandomForestClassifier(
    n_estimators=100,       # nombre d'arbres
    criterion='gini',       # critère d'impureté
    max_depth=4,            # profondeur max des arbres
    random_state=42
)
```

Ensuite, on l'entraîne :

```
clf.fit(X_train, y_train)
```

5. Évaluation de la forêt

```
y_pred = clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)

print("Précision sur le test :", accuracy)
```

6. Importance des variables

```
importances = clf.feature_importances_
for feature, score in zip(iris.feature_names, importances):
    print(f"{feature} : {score:.4f}")
```

7. Prédiction sur une nouvelle fleur

```
X_new = [[5.9, 3.0, 5.1, 1.8]]
prediction = clf.predict(X_new)

print("Classe prédite :", iris.target_names[prediction[0]])
```

8. Code complet

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt

# Chargement
iris = load_iris()
X, y = iris.data, iris.target
```

```
# Split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# Modèle
clf = RandomForestClassifier(n_estimators=100, criterion='gini',
                             max_depth=4, random_state=42)
clf.fit(X_train, y_train)

# Évaluation
y_pred = clf.predict(X_test)
print("Précision :", accuracy_score(y_test, y_pred))

# Importance des attributs
importances = clf.feature_importances_
for name, val in zip(iris.feature_names, importances):
    print(f"{name} : {val:.4f}")

# Prédiction d'une nouvelle fleur
X_new = [[5.9, 3.0, 5.1, 1.8]]
pred = clf.predict(X_new)
print("Classe prédite :", iris.target_names[pred[0]])
```

🎓 Activités supplémentaires

- Changer `n_estimators` (ex. 10, 50, 200) et observer l'impact
- Visualiser un des arbres de la forêt (avec `clf.estimators_[0]`)