TP: Construction d'un Arbre de Décision en Python (Indice de Gini)

1. Objectif du TP

Ce TP vise à découvrir les principes de base pour construire un **arbre de décision** en Python, en utilisant l'*indice de Gini* comme critère d'impureté. Nous allons :

- Charger un jeu de données (Iris) via `scikit-learn`.
- Entraîner un modèle d'arbre de décision avec `criterion='gini'`.
- Évaluer la précision du modèle.
- (Optionnel) Visualiser l'arbre.

2. Jeu de données : Iris

Le jeu de données *Iris*, inclus par défaut dans **scikit-learn**, contient 150 échantillons de fleurs réparties en 3 espèces :

- **Setosa**
- **Versicolor**
- **Virginica**

Chaque fleur est décrite par 4 attributs : longueur/largeur des sépales et pétales (en cm). L'objectif est de prédire l'espèce à partir de ces caractéristiques.

3. Code Python Pas à Pas

3.1 Installation et imports

Si ce n'est pas déjà fait, installez `scikit-learn` :

```
pip install scikit-learn
```

Ensuite, dans un fichier *.py* (ou notebook Jupyter), importez les librairies nécessaires :

```
import numpy as np
import pandas as pd
```

```
# Chargement du dataset Iris
from sklearn.datasets import load_iris

# Pour séparer en train/test
from sklearn.model_selection import train_test_split

# Pour créer l'arbre de décision
from sklearn.tree import DecisionTreeClassifier

# Pour visualiser l'arbre
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt
```

3.2 Chargement du dataset Iris

```
# Charger les données Iris
iris = load_iris()
X = iris.data       # 4 attributs (long/large sépale, long/large pétale)
y = iris.target     # espèce (0=Setosa, 1=Versicolor, 2=Virginica)

print("Nombre d'échantillons :", X.shape[0])
print("Nombre de caractéristiques :", X.shape[1])
print("Espèces :", iris.target_names)
```

Vous verrez apparaître :

- 150 échantillons
- 4 caractéristiques
- 3 classes : ['setosa' 'versicolor' 'virginica']

3.3 Séparation entraînement / test

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

print("Taille des données d'entraînement :", X_train.shape)
print("Taille des données de test :", X_test.shape)
```

Cela crée un ensemble d'entraînement (80 %) et un ensemble de test (20 %).

3.4 Création de l'arbre de décision

On crée un `DecisionTreeClassifier` avec le critère "gini", et on peut limiter la profondeur pour éviter le sur-apprentissage (exemple : `max_depth=4`).

```
clf = DecisionTreeClassifier(criterion='gini', max_depth=4, random_state=42)
clf.fit(X_train, y_train)
```

3.5 Évaluation du modèle

On prédit sur l'ensemble de test et on calcule la précision (accuracy).

```
score = clf.score(X_test, y_test)
print("Précision de l'arbre de décision :", score)
```

On obtient généralement un score autour de **0.9** ou plus.

3.6 Visualisation :

Si vous avez `graphviz` installé, vous pouvez tracer l'arbre :

```
plt.figure(figsize=(10,6))
plot_tree(clf,
          feature_names=iris.feature_names,
          class_names=iris.target_names,
          filled=True)
plt.show()
```

Cela produit un diagramme montrant chaque nœud (les conditions sur les attributs) et les feuilles (classes prédites).

3.7 Prédiction d'une classe en utilisant l'arbre :

```
# Nouvelle fleur (vecteur avec 4 caractéristiques)
nouvelle_fleur = [[5.1, 3.5, 1.4, 0.2]]  # format : liste de liste !
# Prédiction
prediction = clf.predict(nouvelle_fleur)

# Afficher le résultat
print("Indice de classe :", prediction)
```

```
print("Nom de la classe :", iris.target_names[prediction[0]])
```

4. Code Complet

```
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
import matplotlib.pyplot as plt

# 1. Charger les données Iris
iris = load_iris()
X = iris.data
y = iris.target

# 2. Séparer en train/test (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# 3. Créer et entraîner l'arbre (critère Gini)
clf = DecisionTreeClassifier(criterion='gini', max_depth=3, random_state=42)
clf.fit(X_train, y_train)

# 4. Évaluer sur le test
score = clf.score(X_test, y_test)
print("Précision :", score)

# 5. Visualisation
plt.figure(figsize=(10,6))
plot_tree(clf,
          feature_names=iris.feature_names,
          class_names=iris.target_names,
          filled=True)
plt.show()
# Nouvelle fleur (vecteur avec 4 caractéristiques)
nouvelle_fleur = [[5.1, 3.5, 1.4, 0.2]]  # format : liste de liste !
# Prédiction
prediction = clf.predict(nouvelle_fleur)

# Afficher le résultat
print("Indice de classe :", prediction)
```

```
print("Nom de la classe :", iris.target_names[prediction[0]])
```

5. Activités supplémentaires

1. **Modifier la profondeur :** Testez `max_depth=None` ou `max_depth=2` et observez l'impact sur la précision.
2. **Prédire sur un nouvel exemple :** Donnez par exemple un point : **X_new** = [[5.1, 3.2, 1.0, 0.2]]