**Pour acceder le code voila le lien de github :**

**https://github.com/bilalerrabia/TP1-**

1. **Importation des bibliothèques nécessaires :**

```
from sklearn.tree import DecisionTreeClassifier, export_text, plot_tree
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.metrics import accuracy_score
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
```

✓ 0.3s

+ Code + Markdown

2. **Chargement des données :**

```
data = {
    'Prévisions': ['Ensoleillé', 'Ensoleillé', 'Nuageux', 'Pluie', 'Pluie', 'Pluie', 'Nuageux', 'Ensoleillé', 'Ensoleillé', 'Pluie', 'Ensoleillé', 'Nuageux',
    'Température': ['Chaud', 'Chaud', 'Chaud', 'Moyen', 'Frais', 'Frais', 'Frais', 'Moyen', 'Frais', 'Moyen', 'Moyen', 'Moyen', 'Chaud', 'Moyen'],
    'Humidité': ['Haute', 'Haute', 'Haute', 'Haute', 'Normale', 'Normale', 'Normale', 'Haute', 'Normale', 'Normale', 'Normale', 'Haute', 'Normale', 'Haute'],
    'Vent': ['Non', 'Oui', 'Non', 'Non', 'Non', 'Oui', 'Oui', 'Non', 'Non', 'Non', 'Oui', 'Oui', 'Non', 'Oui'],
    'Jouer': ['Non', 'Non', 'Oui', 'Oui', 'Oui', 'Non', 'Oui', 'Non', 'Oui', 'Oui', 'Oui', 'Oui', 'Oui', 'Non']
}
# Conversion en DataFrame
df = pd.DataFrame(data)
# Encodage des variables catégoriques
X = pd.get_dummies(df[['Prévisions', 'Température', 'Humidité', 'Vent']])
y = df['Jouer']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)

clf = DecisionTreeClassifier(criterion='entropy', random_state=0)
clf.fit(X_train, y_train)
```

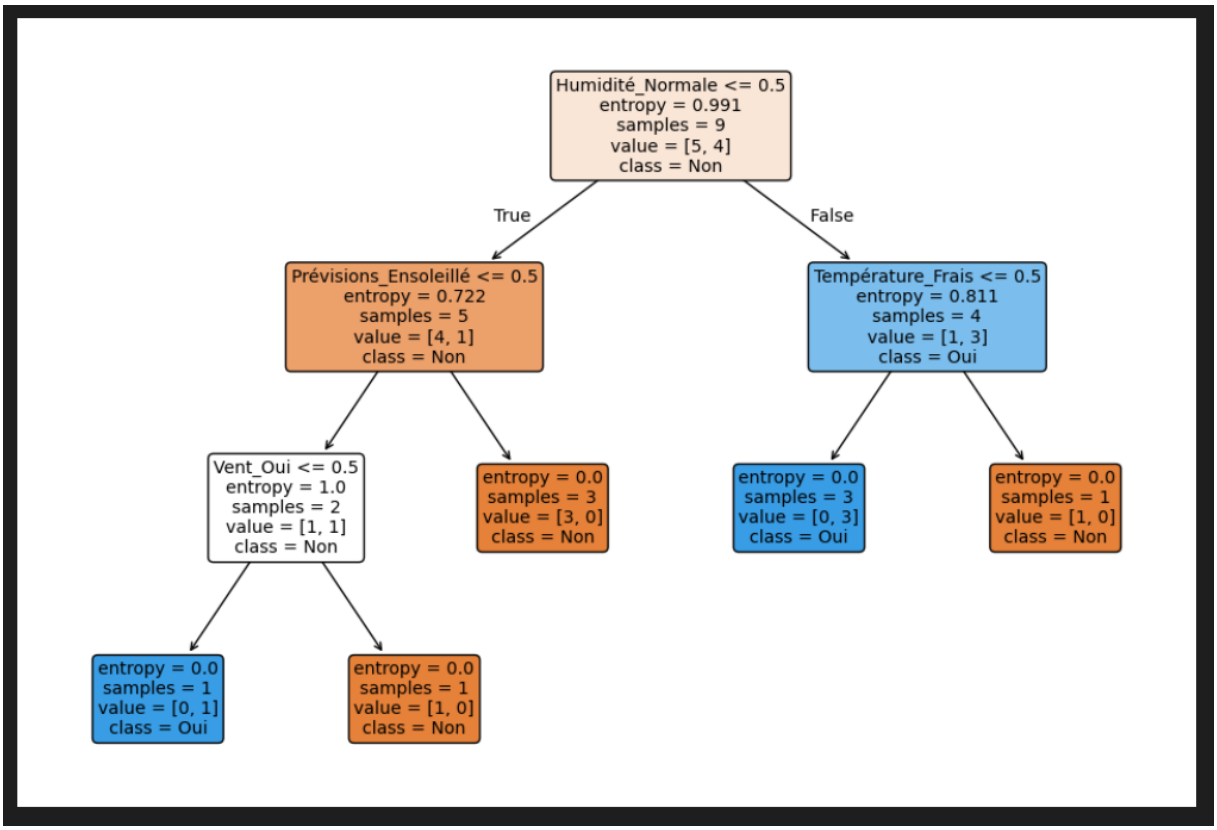
✓ 0.1s Python

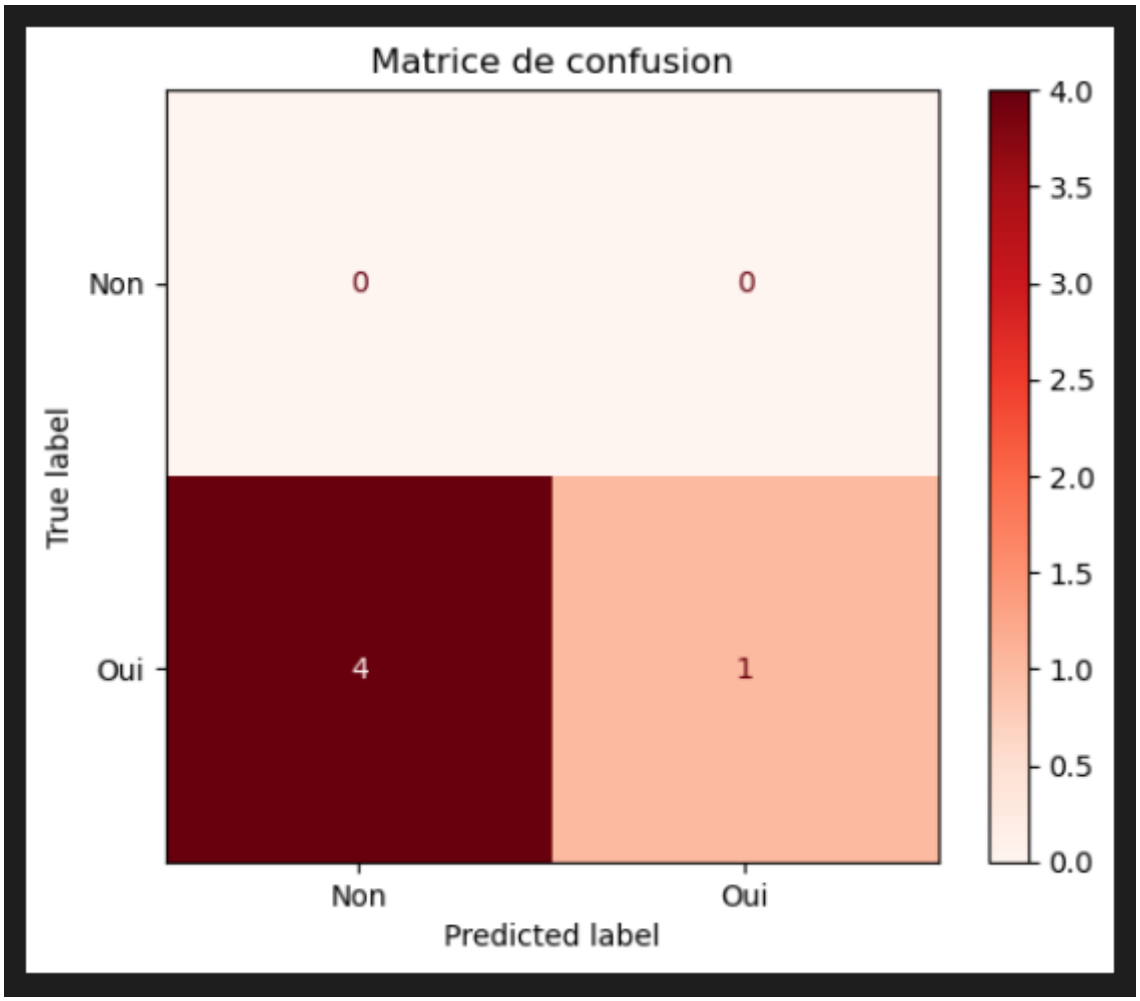
DecisionTreeClassifier

3. **Évaluation sur l'ensemble de test :**

```
score = clf.score(X_test, y_test)
print(f"Précision sur l'ensemble de test : {score * 100:.2f}%")
# 3.    Affichage de l'arbre de décision (texte)
print("Arbre de décision:\n")
print(export_text(clf, feature_names=list(X.columns)))

plt.figure(figsize=(12, 8))
plot_tree(clf, feature_names=X.columns, class_names=clf.classes_, filled=True, rounded=True)
plt.show()
# 4.     Matrice de confusion
y_pred = clf.predict(X_test)
cm = confusion_matrix(y_test, y_pred, labels=clf.classes_)
ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=clf.classes_).plot(cmap='Reds')
plt.title("Matrice de confusion")
plt.show()
# 5.Calculer le taux de reconnaissance (accuracy), Recall et F1-score
accuracy = accuracy_score(y_test, y_pred)
# Afficher le résultat
print(f"Taux de reconnaissance (accuracy) : {accuracy * 100:.2f}%")

recall = recall_score(y_test, y_pred, pos_label='Oui')
print(f"Rappel (Recall) : {recall:.2f}")
#cacul de F1-score
f1 = f1_score(y_test, y_pred, pos_label='Oui')
print(f"F1-Score : {f1:.2f}")

for depth in range(1,6):
    model = DecisionTreeClassifier(max_depth=depth, random_state=42)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    print(f'Depth: {depth}, Accuracy: {accuracy:.2f}')
```

✓ 1.2s


Humidité_Normale <= 0.5
entropy = 0.991
samples = 9
value = [5, 4]
class = Non
True
False
Prévisions_Ensoleillé <= 0.5
entropy = 0.722
samples = 5
value = [4, 1]
class = Non
Température_Frais <= 0.5
entropy = 0.811
samples = 4
value = [1, 3]
class = Oui
Vent_Oui <= 0.5
entropy = 1.0
samples = 2
value = [1, 1]
class = Non
entropy = 0.0
samples = 3
value = [3, 0]
class = Non
entropy = 0.0
samples = 3
value = [0, 3]
class = Oui
entropy = 0.0
samples = 1
value = [1, 0]
class = Non
entropy = 0.0
samples = 1
value = [0, 1]
class = Oui
entropy = 0.0
samples = 1
value = [1, 0]
class = Non


Matrice de confusion
Non
Oui
True label
0
0
4
1
Non
Oui
Predicted label
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0

**Question : pourquoi accuracy score est tres petit = 20 % ?**

La précision de 20 % est très probablement due à la **taille extrêmement petite de l'ensemble de test**. Voici une explication détaillée :

## Problème principal : Taille de l'ensemble de test

- la base de données contient seulement **14 échantillons**.
- Lorsque nous divisons les données avec `test_size=30%`, l'ensemble de test ne contient que **4 ou 5 échantillons**.
- Si le modèle prédit mal **un seul échantillon**, cela peut déjà réduire considérablement la précision. Par exemple :
    - Si l'ensemble de test contient 5 échantillons et que le modèle se trompe sur 4 d'entre eux, la précision sera de **20 %** (1 correct / 5 total).

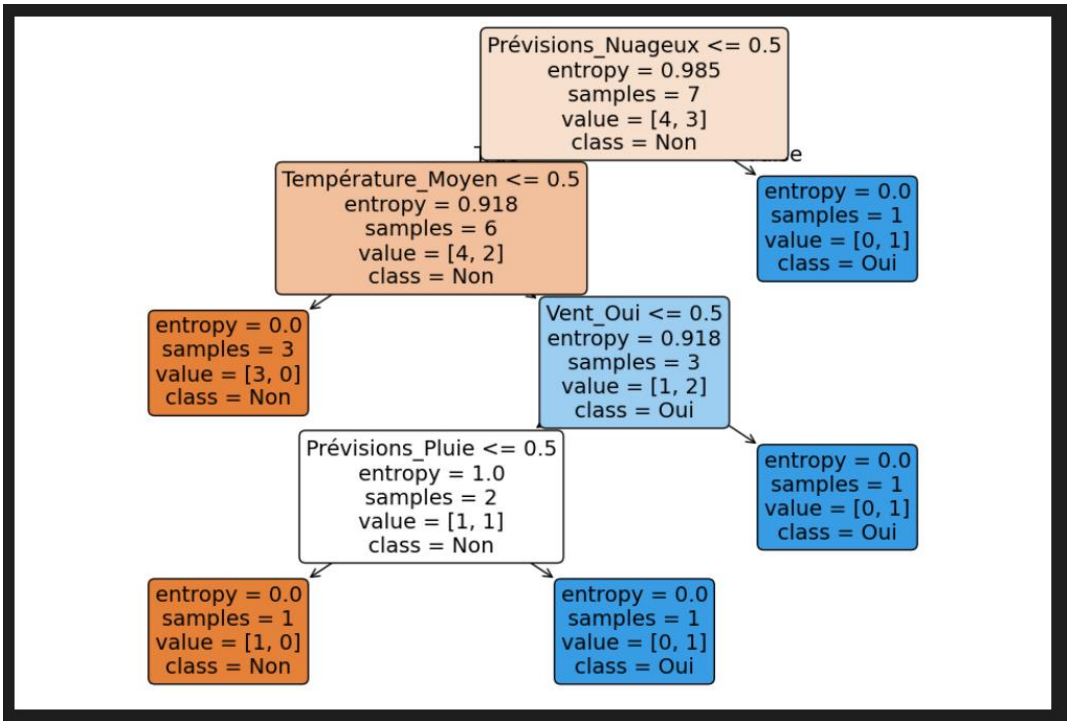
**Question : QU'IL EST IMPACT DE DEPTH SUR L'APPRENTISSAGE DE NOTRE ARBRE ?**

La profondeur (depth) de notre arbre De décision influence sa complexité :

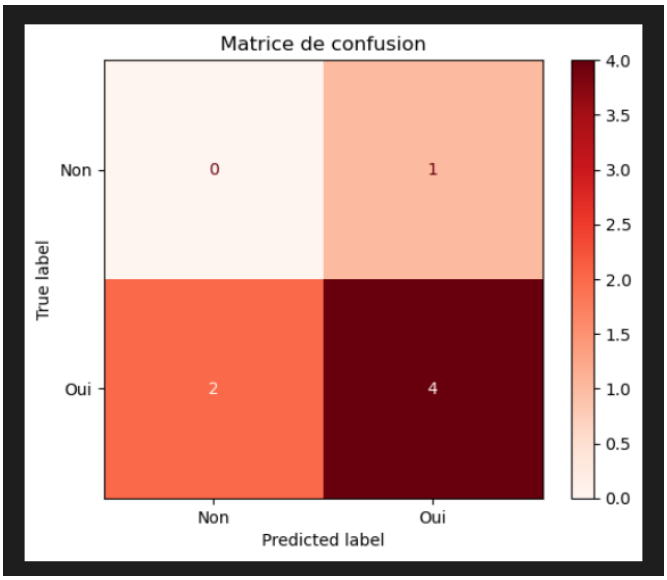
- Profondeur faible (ex : 1-2) : Sous-apprentissage (underfitting), précision faible.
- Profondeur optimale (ex : 3-4) : Bon équilibre, généralisation efficace.
- Profondeur élevée (ex : 5+) : Surapprentissage (overfitting), risque de mémoriser le bruit.

Dans notre cas, `depth = 3` donne une précision de 100%, mais il faut vérifier avec un plus grand ensemble de test ou une validation croisée pour éviter l'overfitting.

On va modifier la taille de test size data par 50% ET l'autre 50% , on va donner pour l'apprentissage et on va constater les resultats




Apres la modification test = 7 echantillion  4  vrai et 3 faux




Voila les nouveaux resultats : AVANT F1 = 20% et Apres la modification : F1 = 73 % et accuracy = 57, 14 %

```
Taux de reconnaissance (accuracy) : 57.14%
Rappel (Recall) : 0.67
F1-Score : 0.73
Depth: 1, Accuracy: 0.57
Depth: 2, Accuracy: 0.57
Depth: 3, Accuracy: 0.71
Depth: 4, Accuracy: 0.57
Depth: 5, Accuracy: 0.57
```