TP 7 - Arbres de décision Iris

NOM : GHODBANI

PRÉNOM : Fares

Groupe : Miage

Réalisé avec Canva et Visual studio code

Introduction

Ce TP a pour objectif d'étudier l'apprentissage supervisé à travers les arbres de décision en utilisant le jeu de données Iris.
Le but est de construire un modèle capable de classer automatiquement les fleurs en trois espèces à partir de quatre caractéristiques : longueur et largeur des sépales et des pétales.
Nous suivrons les étapes décrites dans le fichier consignes.py, depuis l'importation des bibliothèques jusqu'à la visualisation de l'arbre final dans le fichier figure_p7ia.png.
Les remarques présentées dans ce compte-rendu proviennent des résultats des requêtes exécutées dans ce fichier.
Ce TP permet d'évaluer la performance du modèle, d'interpréter l'arbre obtenu et de comprendre les relations entre les caractéristiques des fleurs et leur classification.

1. Importation des bibliothèques

Nous avons importé les bibliothèques nécessaires (pandas, numpy, matplotlib, sklearn.tree, sklearn.metrics) et la fonction train_test_split.

2. Chargement du dataset Iris

Le jeu de données Iris a été chargé avec load_iris().

3. Construction du DataFrame donnees

Les quatre caractéristiques ont été placées dans un DataFrame Pandas.

4. Nommer correctement les colonnes

Les colonnes ont été renommées :
sepal_length, sepal_width, petal_length, petal_width.
Remarque : cela améliore la lisibilité et permet de comprendre facilement les décisions de l'arbre.

5. Construction du DataFrame target

La colonne target contient les classes :

- 0 = setosa
- 1 = versicolor
- 2 = virginica

Remarque : cette étape permet de séparer clairement les caractéristiques (X) et les labels (y).

6. Séparation en train/test

70 % des données pour l'entraînement, 30 % pour le test (random_state=1).

Remarque : cette séparation assure la reproductibilité des résultats et permet de tester la généralisation du modèle.

7. Création et entraînement du modèle

DecisionTreeClassifier avec :

- criterion="entropy"
- max_depth=3
- min_samples_leaf=5

Remarques :

- Profondeur limitée pour éviter le surapprentissage.
- L'entropy est adaptée pour un dataset équilibré.
- L'arbre s'ajuste correctement aux données.

8. Prédiction sur le jeu de test

Prédictions générées avec predict().

Remarque : les prédictions sont conformes aux attentes et permettent d'évaluer la performance du modèle.

9. Précision du modèle

Précision : 95.56 %

Remarque : excellente performance sur le dataset Iris.

10. Matrice de confusion et rapport

```
[[14  0  0]
 [ 0 17  1]
 [ 0  1 12]]
```

Remarques :

- **Setosa classée parfaitement.**
- **Une erreur pour Versicolor et Virginica, ce qui est attendu car ces classes se chevauchent légèrement.**

F1-scores très élevés (≈0.92–1.00).

11–13. Erreurs du modèle

- **MAE :** 0.0444
- **MSE :** 0.0444
- **RMSE :** 0.2108

Remarque : valeurs très faibles, confirmant la qualité du modèle.

14. Visualisation de l'arbre

- Split principal sur petal_length, séparant parfaitement Setosa.
- Versicolor et Virginica séparés grâce à petal_width et sepal_length.

Remarque : arbre clair, interprétable, et conforme à la littérature sur le dataset Iris.

Conclusion

Ce TP a permis de mettre en pratique l'apprentissage supervisé avec les arbres de décision.

- Modèle précis et facile à interpréter.
- Split sur les pétales est déterminant pour la classification.
- Setosa parfaitement identifiable, tandis que Versicolor et Virginica présentent un léger chevauchement.
- Les mesures d'erreur sont très faibles, confirmant l'efficacité du modèle.

Ce TP illustre la puissance et la lisibilité des arbres de décision pour les problèmes de classification simples.