Projet implémentation de l’arbre de décision

ZIDAN Lama N étudiant: 2196961

ZIDAN Loubna N étudiant: 2196960

Version python : python3.10

L’algorithme fonctionne bien.

Voici les étapes que nous avons suivi pour implémenter l'arbre de décision:

1. Choix de la métrique d'impureté : Au commencement, nous devons sélectionner une métrique d'impureté qui servira à évaluer la qualité de nos partitions. Nous utiliserons l'indice de Gini, qui calcule l'impureté d'un nœud en utilisant le nombre de chaque catégorie cible parmi tous les enregistrements correspondant à ce nœud. L'impureté totale de Gini est obtenue en soustrayant de 1 la somme des carrés des proportions des nombres dans toutes les catégories cibles, le tout multiplié par le nombre d'enregistrements.

2. fonction pour calculer l'impureté d'un dataset : Nous développons ensuite une fonction capable de calculer l'impureté du dataset en utilisant l'indice de Gini que nous avons choisi précédemment.

3. Implémentation de la fonction gain d'information : Nous mettons en place une fonction qui calcule le gain d'information lors de la division d'un dataset. Le gain d'information est la différence entre l'impureté du dataset avant la division et l'impureté pondérée des sous-ensembles après la division. Cette fonction nous aidera à choisir la meilleure division de chaque nœud en choisissant la division qui maximise le gain d'information.

4. Trouver le meilleur point de division : Pour chaque feature et chaque valeur unique de ce feature, nous partitionnons le dataset en deux sous-ensembles. Nous calculons ensuite le gain d'information pour cette partition en utilisant la fonction gain d'information. Nous sélectionnons le feature et la valeur qui maximisent le gain d'information pour effectuer la division.

5. Créer un arbre de décision: en choisissant le feature et le seuil de division qui minimisent l'impureté de Gini à chaque nœud, puis en créant récursivement des nœuds enfants jusqu'à atteindre une certaine profondeur maximale (4 ici). Si l'impureté est minimale ou si nous avons atteint la profondeur maximale, nous créons un nœud feuille avec la classe la plus représentée. Ensuit, prédire en suivant l'arbre de la racine à une feuille en utilisant les critères de division à chaque nœud.

6. Implémenter une classe pour le classificateur basé sur l'arbre de décision. C'est à dire une interface pour utiliser l'arbre de décision que nous avons construit avec la classe Tree.

7. Chargement des données Iris, entrainement et évaluation. L'évaluation va être calculé avec les trois métriques couramment utilisés avec la classification et qui sont:

    - Précision ou(Accuracy): C'est la proportion de prédiction correcte parmi l'ensemble de prédiction. il est calculé comme suit: (Nombre de vrais positifs + Nombre de vrais négatifs) / (Nombre total de prédictions)
    - Rappel ou Sensibilité : mesure la capacité du modèle à identifier correctement les cas vrais positifs parmi tous les cas positifs. il est calculé comme suit: Nombre de vrais positifs / (Nombre de vrais positifs + Nombre de faux négatifs)
    - F1-Score: C'est une mesure qui utilise la précision et le rappel en calculant la moyenne. il est calculé comme suit: 2 * (Précision * Rappel) / (Précision + Rappel)

8. Représentation de l'arbre.

Liens consultés pour nous aider à comprendre l'algorithme :

Coder un algorithme d'arbre de décision from scratch - LES MODELES D'ARBRES #4 - YouTube

Coder un algorithme d'arbre de décision from scratch - LES MODELES D'ARBRES #3 - YouTube

How to implement Decision Trees from scratch with Python - YouTube

Decision Trees (mlu-explain.github.io)

Decision Tree Algorithm, Explained - KDnuggets

A Step by Step CHAID Decision Tree Example - Sefik Ilkin Serengil (sefiks.com)