Mini Rapport TD8

Structure de Donnée et Algorithme

I- Explication des types choisis pour manipuler l'arbre de décision

Type « matrice_donnees »

Je ne l'utilise qu'afin de basculer les données vers une autre structure : « table_donnees ».

Type « table_donnees »

J'ai choisi d'utiliser un pointeur de tableau afin de contenir tous les paramètres des différents individus plantes du document. J'ai fait le transfert depuis la matrice car cette dernière me paraissait beaucoup trop compliquée à utiliser (notamment en ce qui concerne les divisions successives de cette dernière lors de la création de l'arbre de décision).

Type « individu »

Celui-ci est utilisé au sein d'une seule et unique fonction : celle pour calculer la médiane corrigée. Il me fallait construire un tableau d'elements pouvant contenir la valeur Y de chaque individu du nœud, ainsi que les valeurs du paramètre Xi testé.

II- Comment est construit automatiquement l'arbre de décision ?

La fonction « creer_arbre_recursivement » prend en paramètre un pointeur sur la racine de l'arbre ainsi que tous les paramètres rentrés par l'utilisateur (hauteur_max…). A partir de ce nœud on détermine le paramètre de décision à utiliser à l'aide de la fonction « parametre_division_a_utiliser ». La particularité de cette fonction c'est de renvoyer 0 s'il n'est pas possible de diviser l'arbre et dans le cas contraire le numéro du paramètre que l'on peut donc utiliser. La médiane est également déterminée grâce a cette fonction (on entre l'adresse mémoire de la médiane) ce qui évite de la recalculer, c'est un gain de temps.

Ensuit, s'il est possible de diviser le nœud en 2 fils, on créé un fils droit et un fils gauche à l'aide des fonction correspondantes (2 fonctions sont nécessaires car on scinde le tableau interne du parent de deux manières différentes), et on les associe au parent.

Enfin, on re-appel la fonction afin de construire le reste de l'arbre.

III- Comment est codé chaque fonctionnalité ?

1) Afficher la hauteur de l'arbre

Elle appel la fonction « rechercher_hauteur_arbre » : l'arbre en entrée est un pointeur constant comme cela on ne le modifiera pas. On on créé un autre temporaire dans le but de parcourir l'arbre. La variable hauteur_max extérieure a cette 2ème fonction permet d'etre incrémente dès que le pointeur tmp se pose sur un nœud ayant une hauteur plus importante. Ce dernier parcourant tout l'arbre a la fin sa valeur est celle de la hauteur de l'arbre de décision.

2) Afficher la largeur de l'arbre

De la même manière que la fonction « rechercher_hauteur_arbre » , on parcourt l'arbre avec tmp. Dès que l'on détecte une feuille, le compteur de feuille s'incrémente. Ainsi lorsqu'il n'y a plus de nœud à parcourir cette dernière variable a pour valeur le nombre total de feuilles.

3) Afficher l'arbre

Adaptation de la fonction arborescence avec une fonction affichage_en_fonction_du_type_de_noeud, cette derniere est importante car on affichera pas de la même manière un fils droit et un fils gauche : car le chemin n'est pas le même. Et dans mon affichage il faut mettre « >= » ou « < » à bon escient. Cette fonction se charge juste de cela.

4) Afficher feuilles de l'arbre

De la même manière, on parcourt l'arbre avec tmp. Sauf que dès que l'on détecte une feuille, la fonction « retrouver_chemin_feuille » se charge de remonter l'arbre jusqu'à la racine en stockant dans 3 tableau les branches, les médianes utilisées, les paramètres utilisées. Il suffit ensuite de lire ces tableaux à l'envers afin de retracer le bon chemin.

5) Prédire

Element_a_predire renvoi simplement un tableau contenant les caractéristiques de l'individu inconnu (donc tableaux taille 4). Ensuite il suffit de parcourir le tableau grâce a tmp et tant que l'on n'arrive pas à une feuille de passer de branches en branches en fonction des paramètres et des médianes utilisées. Enfin on affiche la proportion *100 de la feuille correspondante à l'individu.

IV- Predire ces individus :

1) 100%

2)20%

3)20%

4)3.4%

5)20%

6)3.4%

7)20%

8)3.4%

9) 20%

10)3.4%