TP à rendre

ING 1 GI GM 2022-2023

Programmation Procédurale

Modalités :

- Pour chaque sujet vous devez déposer le rendu sur git et m'envoyer le lien par mail au plus tard le **10/12/2022 à minuit.**
- Vous devez faire obligatoirement un **makefile** et un **redame** pour chaque rendu et pensez à séparer votre projet en plusieurs fichiers.
- Vous pouvez constituer des groupes de deux étudiants au maximum.

Sujet 1 : Prédire le prix d'une habitation qu'on veut mettre en location dans la plateforme Airbnb

Aujourd'hui, des millions d'hôtes et de voyageurs choisissent Airbnb pour publier leur annonce ou réserver des logements uniques partout dans le monde. Il s'agit d'un service de plateforme communautaire payant de location de logements. Elle permet, par exemple, à des particuliers de louer tout ou une partie de leur propre habitation comme logement d'appoint.

Lorsqu'un particulier décide de mettre en location une habitation dans la plateforme il aura besoin d'évaluer le prix de dernier en fonction des prix proposés dans le marché. Si le prix trop élevé il risque de ne pas trouver d'hôte et s'il est trop bas ce sera moins intéressant pour le propriétaire.

Dans ce TP vous allez implémenter en langage C l'algorithme des K plus proches voisins pour prédire le prix d'un logement x qu'on veut mettre en location.

Le fichier airbnbParis.csv contient des logements publiés dans la plateforme Airbnb en 2019 mais uniquement en ile de France. Chaque ligne du fichier est un logement et pour chaque logement ses caractéristiques sont séparées par des virgules.

L'algorithme des k plus proche voisins ou K-nearest neighbors (KNN) est un algorithme de machine Learning qui appartient à la classe des algorithmes d'apprentissage supervisé simple et facile à mettre en œuvre qui peut être utilisé pour résoudre les problèmes de classification et de régression.

Travail à faire

Partie 1 : Implémentation de l'algorithme KNN

Pour prédire le prix d'un logement l'algorithme consiste à trouver les k logements les plus similaires au logement x que l'on veut louer et à calculer le prix moyen. Pour cela la métrique de similarité qui sera utilisée est la distance euclidienne donnée par la formule ci-dessous.

$$\text{distance} = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

Chaque logement a n caractéristiques ou attributs. Les notations $xi\ et\ yi$ représentent, respectivement, les valeurs numériques de la caractéristique $i$pour les logements x et y. Ici la distance entre deux logements sera calculée en fonction d'une seule caractéristique à la fois (KNN univarié). La distance euclidienne s'applique uniquement sur des caractéristiques à valeur numérique.

Par exemple on peut calculer la similarité du logement x par rapport aux logements existants selon le nombre de personnes qu'il peut accueillir, le nombre de chambres qu'il dispose ou le nombre de lits etc.

a) Calculer la distance de similarité du logement x avec chaque logement du fichier airbnb_donnees_propores.csv.

   Indication : un logement est représenté par une structure avec différents champs. Créer un type « *Logement* » avec les différents champs nécessaires pour l'algorithme.

b) Randomiser et trier le tableau de logements.

   Indication : randomiser le tableau consiste à permuter de façon aléatoire les éléments. Si le tableau est trié sans être randomisé avant on aura toujours les mêmes k premiers logements. Pour être plus proche de la réalité le tableau doit être randomisé et trié après par ordre de croissance de la distance de similarité. Plus la distance de similarité est petite plus le logement est similaire à x.

c) Sélectionner les k premiers logements du tableau ou de la liste randomisée et trié puis calculer la moyenne des prix. Le prix trouvé sera le prix prédit pour le logement x. Tester votre programme en faisant varier la valeur de k.

d) Entrainer d'autres modèles en utilisant d'autres caractéristiques.

Indication : Modifier le type « Logement » en remplaçant le champ distance par un tableau de distances. Donc chaque logement aura un tableau de distances ou on mettra dans chaque case sa similarité avec le logement x selon un critère donné. Ici nous considérons 3 critères : le nombre de personnes que peut accueillir un logement « *accommodates* », le nombre de chambres « *bedrooms* », le nombre de lits « *beds* ».

Partie 2 (Bonus) : Evaluation de la performance du modèle de prédiction.

Pour tester la qualité des prédictions le dataset airbnb_donnees_propres est séparée en deux partitions appelées données d'entrainement et données de test contenant respectivement 80% et 20% des lignes du dataset airbnb_donnees_propres.

e) Créer un tableau de logements nommé tabEntrainement pour charger le fichier airbnbEntrainement.csv
f) Créer un tableau de logements nommé tabTest pour chager le fichier airbnbTest.csv
g) Utiliser les données de tabEntraiment dans l'algorithme implémenté dans la partie 1 pour faire une prédiction pour chaque élément de tabTest. Stocker dans un tableau tabPrediction de même taille que tabTest le prix obtenu pour chaque élément de tabTest.

Maintenant pour chaque logement de tabTest vous avez le prix réel et le prix prédits par l'algorithme. L'évaluation de la qualité de votre algorithme consiste à comparer les prix réels et les prix prédits.

La MAE est une métrique très populaire utilisée pour tester la qualité des prédictions. Elle correspond à la moyenne des valeurs absolues des erreurs, définie par la formule suivante :

$$\mathrm{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n} = \frac{\sum_{i=1}^{n} |e_i|}{n}.$$

Avec ei la différence entre le prix réel et le prix prédit, n le nombre d'éléments de tabTest.

h) Calculer MAE de votre modèle.
i) Reprenez la question d) en calculant les mae pour différents modèles avec des valeurs de k différents.
j) Trouver la valeur de k qui donne de meilleure performance. Plus la mae est faible plus votre modèle est performant.

Sujet 2 : Ordonnancement de processus

L'ordonnancement consiste, pour le système d'exploitation, à optimiser l'utilisation du processeur en lui affectant tour à tour différentes tâches à exécuter (processus). Il peut y en avoir des centaines à la fois sur une machine. L'ordonnanceur va répartir le temps de calcul entre les programmes, afin que tous puissent avancer dans leur exécution de manière satisfaisante, la plupart des ordonnanceurs modernes utilisent des files pour garder en mémoire de façon optimale les programmes à exécuter. La file est parfaitement adaptée à l'ordonnancement : les programmes qui demandent du temps de calcul sont insérés en bout de file, et ceux qui seront défilés pour obtenir effectivement du temps processeur sont ceux qui attendent depuis le plus longtemps

Un processus est caractérisé par un nom, une durée d'exécution et une priorité. La priorité est un entier compris entre 0 et 5 par exemple. Un ordonnanceur est caractérisé par une file de processus.

(a) Créer les structures permettant de modéliser un processus et un ordonnanceur et écrire les programmes suivants :
(b) ajout_activite qui ajoute une activité passée en paramètre à la file de processus de l'ordonnanceur.
(c) step qui effectue un "tour" d'ordonnancement comme suit : si la file est vide, on le dit et on ne fait rien. S'il y a au moins une activité dans la file, on défile et on exécute l'activité en affichant son nom et sa durée et en attendant le temps correspondant à la durée de l'activité.
(d) run qui itère step jusqu'à obtenir une file de processus vide.
(e) Ecrire un programme principal.