

UNIVERSITÉ DE SHERBROOKE  
DÉPARTEMENT D'INFORMATIQUE

IFT 870 - Forage de données  
TP#3 : Fonctions prédictives  
Hiver 2025

---

Le but de ce devoir est de développer et pratiquer des méthodes pour le réglage des hyperparamètres de méthodes prédictives.

Ce devoir est à faire en équipe de deux ou trois. Il devra être complété avant le vendredi 28 mars 2025 à 23h59. Vous devez remettre, sur `turnin.dinf.usherbrooke.ca`, un seul fichier Ipython notebook (nommé `tp3.ipynb`) contenant votre rapport et vos scripts Python pour ce devoir.

---

**Description des tâches à réaliser :**

On vous fournit un ensemble de données stockées dans un fichier au format `.csv` (`TP3_data.csv`). L'ensemble des données contient 10,000 observations représentées suivant 4 variables (`Attribut1`, `Attribut2`, `Attribut3`, `Attribut4`). Les données sont segmentées en 20 classes (0,1,...,19). La classe d'une observation est représentée par la valeur de la variable `Classe` dans le fichier de données. L'objectif du TP est d'implémenter, utiliser et comparer les résultats de fonctions de réglage d'hyperparamètres de type `GridSearch` pour le modèle de classification `sklearn.neighbors.KNeighborsClassifier`. Les hyperparamètres à régler sont : `n_neighbors` dans l'intervalle de valeurs entières [1,20], et `p` dans l'intervalle de valeurs entières [1,10].

1. **Implémentation de fonctions :** Vous devez implémenter :

- (a) une fonction `model_score(X,y,class_model,params)` qui prend en paramètre une matrice `X` de données, un vecteur `y` de classes correspondant, un modèle de classification (exemple : `class_model = KNeighborsClassifier()`), et un paramétrage exemple : `params = {'n_neighbors': 10, 'p': 5}`). La fonction retourne la moyenne de score de précision (accuracy) par validation croisée en divisant les données en 5 parties.
- (b) une fonction `bruteforce_optimisation(class_model,grille_param,X,y)` qui prend en paramètre un modèle de classification, une grille de paramètres (exemple : `grille_param = {'n_neighbors': range(1,11), 'p': range(1,6)}`), une matrice `X` de données, et un vecteur `y` de classes correspondant. La fonction explore tout l'espace de recherche des paramétrages et retourne un seul paramétrage de score maximal (exemple : `max_params = {'n_neighbors': 10, 'p': 5}`).

- (c) une fonction **randomize\_optimisation(class\_model, grille\_param, X, y, sample\_percent)** qui prend les mêmes paramètres que la fonction **bruteforce\_optimisation**, plus le pourcentage de paramètres échantillonnés **sample\_percent** (exemple : **sample\_percent = 30** ). La fonction explore uniquement l'espace de recherche échantillonné des paramètres et retourne un seul paramétrage de score maximal dans cet espace.
- (d) une fonction **halving\_optimisation(class\_model, grille\_param, X, y, n\_splitting)** qui prend les mêmes paramètres que la fonction **bruteforce\_optimisation**, plus le nombre de parties dans lequel les données et les paramètres sont divisés (exemple : **n\_splitting = 5**). La fonction explore tout l'espace de recherche des paramètres de taille  $P$ , puis un espace de taille  $P \times (n\_splitting - 1) / n\_splitting$ , puis  $P \times (n\_splitting - 2) / n\_splitting$ , etc., jusqu'à ne conserver qu'un paramétrage. De même, elle commence avec un échantillon des données de taille  $N \times 1 / n\_splitting$ ,  $N \times 2 / n\_splitting$ , etc., jusqu'à  $N \times n\_splitting / n\_splitting$  qui correspond à 100% des données. Elle retourne le seul paramétrage conservé à la fin.
- (e) une fonction **bayesian\_optimisation(class\_model, grille\_param, X, y, s\_size, n\_iter)** qui prend les mêmes paramètres que la fonction **bruteforce\_optimisation**, plus le nombre de paramètres échantillonnés à chaque itération du processus (exemple : **s\_size = 5**), et le nombre d'itération de la fonction (exemple : **n\_iter = 100**). L'optimisation bayésienne utilise une fonction d'approximation pour estimer la fonction de score par échantillonnage. Dans ce TP, la fonction d'approximation utilisée est le modèle de régression **sklearn.gaussian\_process.GaussianProcessRegressor**. La prédiction avec ce modèle retourne un vecteur des moyennes et un vecteur des écart-types des distributions prédictives à chaque donnée. La méthode commence par générer un échantillon **E** de **s\_size** paramètres qu'elle utilise pour estimer (fit) la fonction d'approximation. Puis, elle répète **n\_iter** fois le processus suivant :
- elle utilise la fonction d'approximation pour prédire les scores (moyennes et écart-types) de tous les paramètres ;
  - elle trouve la moyenne maximum prédite **max\_pred\_moy** ;
  - elle échantillonne **s\_size** paramètres ;
  - elle utilise la fonction d'approximation pour prédire les scores (moyennes et écart-types) des paramètres échantillonnés ;
  - les résultats de cette prédiction sont transformés en une distribution de probabilité en utilisant la fonction de distribution cumulative (**cdf**), comme suit : **probabilite = cdf((moyennes - max\_pred\_moy) / (ecart\_types + 10<sup>-6</sup>))** ;
  - le paramétrage de probabilité maximum est choisi : **max\_param** ;
  - **max\_param** est ajouté à l'échantillon **E**, puis **E** est utilisé pour ré-estimer la fonction d'approximation.
- Après la dernière itération, le paramétrage de score maximum dans l'échantillon **E** est choisi.

## 2. Comparaison de fonctions :

- (a) Proposez un partitionnement des données en données d'entraînement et données de test.
- (b) Présentez un graphique des scores d'entraînement, et un graphique des scores de test, pour tous les paramétrages de l'espace de recherche, sous forme de heatmap.
- (c) Appliquez les fonctions suivantes pour le réglage des hyperparamètres, et commentez les résultats :
  - `bruteforce_optimisation`
  - `randomize_optimisation` avec `sample_percent = 30`
  - `halving_optimisation` avec `s_splitting = 5`
  - `bayesian_optimisation` avec `s_size = 5` et `n_iter = 100`
  - `model_selection.GridSearchCV` avec `scoring='accuracy'`
  - `model_selection.RandomizedSearchCV` avec `n_iter = 60` et `scoring='accuracy'`
  - `model_selection.HalvingGridSearchCV` avec `factor = 5` et `scoring='accuracy'`
  - `skopt.gp_minimize` avec comme modèle de score `1 - accuracy`
- (d) En faisant varier le partitionnement des données et les paramètres des fonctions de réglage des hyperparamètres, comparez les performances des fonctions en termes de temps de calcul, et de capacité à trouver un paramétrage optimal. Commentez les résultats.

## Remise du travail

Pour soumettre votre travail, connectez-vous, dans un navigateur, au serveur <http://turnin.dinf.usherbrooke.ca> en utilisant votre CIP, puis choisissez le cours IFT870 (BIN710) et le projet TP3. Chargez votre fichier `tp3.ipynb` et soumettez-le. Le nom de votre fichier de remise doit être exactement `tp3.ipynb`. Indiquez bien les noms des deux membres de l'équipe dans le fichier. Ne faites qu'une seule soumission par équipe. Ne remettez pas d'autre fichier.