UNIVERSITÉ DE SHERBROOKE
DÉPARTEMENT D'INFORMATIQUE

IFT 870 BIN 710 - Forage de données

TP#4 : Fonctions prédictives

Hiver 2026

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. Il devra être complété avant le vendredi 17 avril 2026 à 23h59. Vous devez remettre, sur `turnin.dinf.usherbrooke.ca`, un seul fichier Ipython notebook (nommé `tp4.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` (`TP4_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 `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 paramètrages 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 paramétrages sont divisés (exemple : `n_splitting = 5`). La fonction explore tout l'espace de recherche des paramètres de taille `P`, puis un epace 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 paramétrages 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 vector des écart-types des distributions prédictives à chaque donnée. La méthode commence par générer un échantillon `E` de `s_size` paramétrages 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 paramétrages ;
- elle trouve la moyenne maximum prédite `max_pred_moy` ;
- elle échantillonne `s_size` paramétrages ;
- elle utilise la fonction d'approximation pour prédire les scores (moyennes et écart-types) des paramétrages 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^-6))` ;
- 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 fureteur, au serveur `http://turnin.dinf.usherbrooke.ca` en utilisant votre CIP, puis choisissez le cours `IFT870 (BIN710)` et le projet `TP4`. Chargez votre fichier `tp4.ipynb` et soumettez-le. Le nom de votre fichier de remise doit être exactement `tp4.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.