APPRENTISSAGE STATISTIQUE - MS ENSAI
TP1 : k-PPV ET ARBRES DE DÉCISIONS. 2025-2026

1 Objectifs du TP

Nous allons mettre en application l'algorithme des k plus proches voisins et l'algorithme CART d'arbres de décisions sur deux jeux de données très classiques. En fonction de votre temps disponible et de vos envies, vous pouvez étudier un ou plusieurs problèmes supplémentaires proposés ci-après

Dans chaque cas, il s'agit d'effectuer rapidement une première exploration descriptive des données et une analyse statistique univariée des variables d'intérêt supposées, puis d'appliquer d'abord l'algorithme k-NN et ensuite l'algorithme CART.

Pour chacun des deux problèmes, il est demandé de découper en plusieurs parties l'ensemble des données, pour former des échantillons d'apprentissage et des échantillons de test. Une des méthodes possibles (validation croisée de type V-Fold) consiste à séparer le jeu de données en 5 blocs de tailles égales, puis de diviser chaque bloc en un échantillon d'entraînement comprenant 80% des données et un échantillon de test. Pour chaque bloc, on estime le prédicteur à partir du sous-échantillon d'entraînement, puis on calcule son risque empirique sur le sous-échantillon de test (et cela pour chaque valeur de k que l'on souhaite tester).

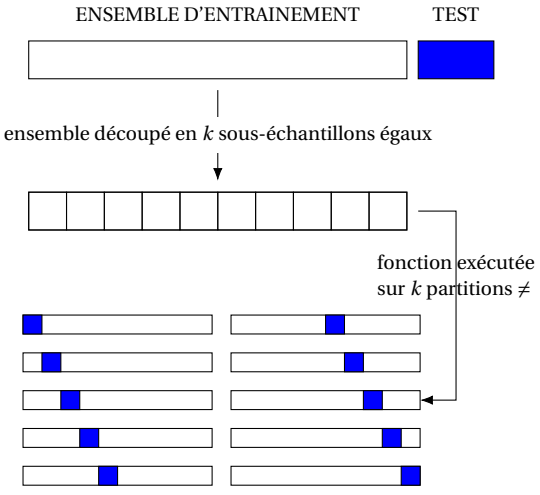

FIGURE 1 – Validation croisée k-Fold pour $k = 10$.

Vous devrez évaluer la qualité des prédictions à l'aide des techniques usuelles.

2 Reconnaissance de chiffres manuscrits

La base de données est tirée de l' « UC Irvine Machine Learning Repository ». C'est un jeu de données regroupant 5620 images de chiffres manuscrits compris entre 0 et 9. On a demandé à plusieurs personnes de tracer des chiffres, d'abord de façon posée, puis de façon plus rapide. Les images de ces chiffres ont été discrétisées en matrices de pixels de 8 lignes sur 8 colonnes, dont chaque coefficient est un entier compris entre 0 et 16 représentant le niveau de gris du pixel.

Les données sont disponibles à l'adresse https://archive.ics.uci.edu/ml/datasets, mais on peut les charger directement à partir d'une bibliothèque Python. Nous n'utilisons que la partie test du jeu de données qui comprend 1797 images.

1°. Les données sont disponibles via la commande `load_digits` de la bibliothèque `sklearn.datasets`. Charger les données, vérifier la taille de l'échantillon et afficher quelques un des chiffres manuscrits à l'aide de la commande `imshow`.

2°. Calculer et afficher quelques statistiques descriptives.

3°. À l'aide la commande `TSNE`, utiliser cet algorithme pour afficher en deux dimensions une représentation de la base de données. Il sera nécessaire de transformer les matrices 8×8 en vecteurs colonnes à 64 coordonnées. Que pensez-vous du résultat?

4°. Découper les données en échantillons d'entrainement et de test (20% pour le test, 80% pour l'entrainement). Le classifieur k-NN est donné par la fonction `KNeighborsClassifier` de `sklearn.neighbors`. Les paramètres les plus importants sont la valeur de k (`n_neighbors`), les poids (`weights`) et le paramètre p de la métrique de Minkowski. Créer un vecteur contenant plusieurs valeurs de ces paramètres, puis à l'aide de la fonction `GridSearchCV` de `sklearn.model_selection` récupérer les meilleures valeurs des paramètres. `GridSearchCV` opère une validation croisée de type k-fold sur les données dont on peut fixer la valeur (on prendra `cv=5`).

5°. À l'aide de la fonction `classification_report`, évaluer la qualité de la prédiction. Afficher la matrice de confusion.

6°. Importer et utiliser les fonctions `DecisionTreeClassifier` et `cross_val_predict` pour construire un arbre de décisions. Évaluer ses performances et comparer avec la méthode des ppv.

7°. Avec la fonction `plot_tree` afficher l'arbre obtenu. Qu'en pensez-vous? Comment expliquez-vous l'écart de performances entre les deux méthodes?

8°. Si vous avez le temps, reprendre l'exercice avec le jeu de données MNIST, qui est beaucoup plus fourni (60 000 images). Pour télécharger les données, installer `python-mnist`.

3 Iris de Fisher

1°. Charger le jeu de données des iris de Fisher, en même temps que les bibliothèques habituelles de traitement des données.

2°. Découper le jeu de données en réservant 75% des données pour l'entrainement et le reste pour les tests. À l'aide de la fonction `tree.DecisionTreeClassifier` construire un arbre de décisions, l'entraîner et évaluer ses performances.

3°. Afficher l'arbre de décision à l'aide des fonctions `tree.export_graphviz`, `pydotplus` et `tree.plot_tree`. Jouer sur la taille de la figure et celle de la police de caractères pour que l'arbre soit lisible.

4°. Reprendre le jeu de données et appliquer une validation croisée afin d'optimiser l'arbre. Vous pouvez utiliser par exemple la fonction `cross_val_predict` puis `classification_report`. Comparer les résutlats.

5°. Tester la méthode k-NN sur le jeu de données en déterminant la valeur de k optimale ; pour cela, faire une boucle sur les différentes valeurs de k à tester et utiliser la fonction `cross_val_score`. Avec les paramètres, l'appel ressemblera à : `cross_val_score(knn, X_train, Y_train, cv=10, scoring='accuracy')`. Comparer les performances des deux méthodes.

4 Reconnaissance de formes de chiffres manuscrits : un second jeu

Ici, la base de données est tirée de l' « UC Irvine Machine Learning Repository ». C'est un jeu de données regroupant 1593 images de chiffres manuscrits compris entre 0 et 9. Les images de ces chiffres ont été discrétisées en tableaux de 16 lignes sur 16 colonnes, puis un seuillage a été effectué sur les niveaux de gris. Finalement, un chiffre manuscrit est représenté par un tableau 16×16 formé de 0 et de 1. Ce tableau est transformé en vecteur ligne (de 256 coordonnées binaires) dans la base de données. Pour chaque ligne, 10 colonnes sont ajoutées et contiennent, sous forme de variables catégorielles, les valeurs réelles des chiffres correspondants.

La base de données est disponible ici : https://archive.ics.uci.edu/ml/datasets.

Reprendre les questions de la section 2 et les appliquer à ce jeu de données.

5 Prévision du niveau de revenus annuels

La base de données « Census Income Data Set » est disponible sur cette page et contient 48842 individus dont on a observé 15 variables (âge, CSP, niveau d'éducation, métier, situation professionnelle, sexe, etc.). Il s'agit de prédire le niveau de revenu annuel des individus en fonction des 14 premières variables : est-il plus grand ou plus petit que 50K par an ?

Séparer la base de données en un échantillon d'apprentissage (les 32561 premières lignes) et un échantillon test (les 16281 dernières lignes) et évaluer la différence de performances des classifieurs pour une régression logistique, une classification par la méthode des k plus proches voisins et un arbre de décision.

6 Regression par k-ppv et arbres de décision

1°. Créer un nuage de points autour d'une sinusoïde et utiliser un régresseur à base de k-ppv pour approcher la courbe initiale (utiliser la fonction `KNeighborsRegressor`). Faire varier le paramètre k et évaluer la qualité de l'approximation en fonction de k. Modifier également le paramètre poids.

2°. Effectuer le même travail avec un arbre de décision (utiliser la fonction `DecisionTreeRegressor`). Faire varier la profondeur de l'arbre et évaluer la qualité de l'approximation en fonction de ce paramètre.