Data Challenge

MS2A - Apprentissage statistique

November 9, 2025

Data Challenge - Identification de gaz toxiques

Leïna Montreuil
M1 mathématiques et application
Sorbonne université
leina.montreuil@etu.sorbonne-universite.fr

Maxime Sangnier
Sorbonne Université

0.1 Introduction

La détection précoce des gaz toxiques est cruciale pour la sécurité industrielle et militaire. Ce travail vise à prédire le niveau d'alarme associé à divers agents chimiques à partir de mesures de capteurs portatifs. Le rapport présente le prétraitement des données, la sélection et la conception du modèle final, puis les performances sur les ensembles public et privé, avant de discuter les difficultés rencontrées et les enseignements tirés. Lien du challenge : https://challengedata.ens.fr/participants/challenges/156/

0.2 Analyse

Une première phase d'exploration a porté sur la compréhension des données. À l'aide de `ydata-profiling`, j'ai analysé les distributions, valeurs manquantes et corrélations entre variables, complétées par des visualisations (*violin plots*, histogrammes) pour comparer les capteurs (`M4--M15`, `S1--S3`, `R`) et l'humidité entre `x_train` et `x_test`.

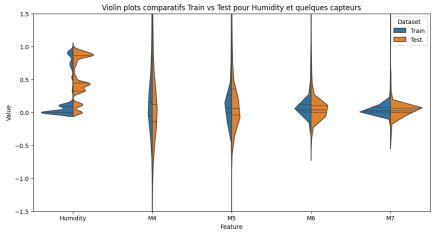

Figure 1: Comparaison des distributions par *violin plot*

Les analyses ont révélé des distributions similaires pour la plupart des capteurs, mais une différence marquée pour la variable `Humidity`, plus élevée dans `x_test`. Cela suggère un changement de conditions physiques plutôt qu'une mesure liée au gaz. Les corrélations et *scatter plots* n'ont montré aucun lien notable entre l'humidité et les autres variables. Les tests de modèles avec et sans cette variable ont confirmé qu'elle dégradait la généralisation, et a donc été exclue du jeu d'entraînement.

0.3 Prétraitement des données

Les jeux de données ont été séparés en trois ensembles distincts : un ensemble d'entraînement,de validation et de test.
Les variables d'entrée ont été standardisées à l'aide de la classe `StandardScaler` de `scikit-learn`
À la suite de l'analyse exploratoire, la variable `Humidity` a été retirée du jeu de données. En effet, son inclusion dégradait la généralisation des modèles en raison de sa distribution très différente entre l'entraînement et le test.

0.4 Phase de selection du modèle

J'ai commencé le data challenge assez tôt dans l'année, en testant d'abord des modèles que je connaissais déjà. J'ai ainsi exploré des modèles linéaires, des méthodes à noyaux, puis des réseaux de neurones de type MLP, avec lesquels j'ai rapidement obtenu de bons résultats. Au fur et à mesure de l'avancement du cours, j'ai appris que, pour les données tabulaires, les modèles les plus performants étaient souvent les forêts aléatoires et les méthodes de boosting. J'ai donc comparé ces approches à mes MLP, mais malgré de nombreuses recherches d'hyperparamètres à l'aide d'une recherche aléatoire (Randomized Search) et beaucoup de temps d'attente, mes réseaux de neurones restaient les plus performants.
Pour aboutir à la version finale du MLP, j'ai mené de nombreux ajustements successifs en observant systématiquement les courbes de perte d'entraînement et de validation. J'ai notamment fait varier la taille et le nombre de couches, le taux d'apprentissage, ajouté des couches de normalisation (BatchNorm et LayerNorm), un planificateur de taux d'apprentissage (scheduler), un arrêt anticipé (early stopping), data

augmentation, modifier la taille des batchs, et testé différentes fonctions d'activation (LeakyRelu, Gelu, Relu, Tanh...) afin d'améliorer la stabilité et la généralisation du modèle. J'avais également envisagé de tester les skip connections, découvertes dans le cours de vision par ordinateur, mais je n'ai pas pu le faire faute de temps, le cours ayant eu lieu tardivement.

0.5 Modèle retenu

J'ai testé plusieurs modèles supervisés : forêts aléatoires, méthodes de boosting (`XGBoost`, `LightGBM` avec et sans loss custom), *Kernel Ridge* et *Lasso*, *SVM regressor* avant d'obtenir mes meilleurs résultats avec un MLP. Ce modèle a été retenu pour ses meilleures performances, sa facilité d'ajustement via la visualisation des pertes d'entraînement et de validation, et sa rapidité d'exécution (environ 5 à 10 minutes par entraînement), permettant d'explorer efficacement différents compromis biais/variance. L'architecture et les principaux hyperparamètres sont présentés dans le tableau 1. **Score public : 0,14143** *(score privée non connue à ce jour mais le score privé du 19/10, j'ai d'ailleurs sélectionné le mauvais modèle mais j'avais obtenu 0,1490*

Table 1: Architecture et hyperparamètres du modèle MLP retenu

Type de modèle	Réseau de neurones multicouche (MLP)
Nombre de couches cachées	3
Taille des couches cachées	[128, 64, 32]
Fonction d'activation	ReLU
Régularisation	Dropout ($p = 0.2$)
Nombre de paramètres	13 207 pour 162 346 exemples d'entrainement
Fonction de perte	MSE personnalisée comme dans l'énoncé (`LossCustom`)
Optimiseur	Adam (lr $= 2 \times 10^{-3}$, weight decay $= 10^{-4}$)
Scheduler	ReduceLROnPlateau (facteur = 0.5, patience = 5)
Critère d'arrêt	Early stopping (patience = 10)
Taille de batch	1024
Nombre maximal d'époques	250

0.6 Erreurs rencontrées et réussites

Au début du projet, j'ai exploré les distributions globales des variables afin de formuler quelques hypothèses sur le comportement des capteurs. Cependant, je n'avais pas comparé en détail les distributions entre `xtrain` et `xtest`, ce qui a entraîné une longue stagnation des performances.J'ai ensuite recentré mes efforts sur l'analyse des variables plutôt que sur la complexité des modèles. À l'aide de `ydata-profiling` et de *violin plots*, j'ai mis en évidence une différence marquée sur la variable `Humidity`, découverte déterminante pour améliorer la généralisation. Le temps de calcul et la limite de deux soumissions par jour ont constitué des contraintes : certaines méthodes (modèles à noyaux, forêts aléatoires avec recherche d'hyperparamètres) demandaient plusieurs heures d'exécution, voire plantaient sur `Google Colab`. Je craignais d'explorer trop de modèles différents, au risque d'épuiser inutilement les essais disponibles. Ceci m'a souvent conduit à concentrer mes efforts sur l'analyse approfondie des modèles déjà prometteurs plutôt que sur la diversification des approches. Finalement, au bout d'un moment je suis arrivée à une stagnation au niveau de me résultats et malgré mes recherches, je n'ai plus réussi à les améliorer. Pour améliorer les performances, j'ai approfondi la littérature et mes cours (RDFIA/DeepL) afin de tester des méthodes de normalisation et schédulers. J'ai également exploré des approches de boosting et de forêts aléatoires que je n'avais pas encore explorées. Surtout, j'ai mené une analyse approfondie des hyperparamètres des MLP, ce qui m'a permis de mieux comprendre leur impact sur la stabilité et la performance du modèle.