DataChallenge
Toxic Gas Identification by Bertin Technologies

Hugo Bouton

github.com/hugojava/DataChallenge

9 novembre 2025

Résumé

Ce rapport présente la démarche de modélisation mise en œuvre dans le cadre d'un *data challenge* proposé par Bertin Technologies, visant à prédire la présence de gaz toxiques à partir de mesures de capteurs. Le travail s'articule autour d'expérimentations successives sur différents modèles d'apprentissage supervisé, dans un contexte de forte hétérogénéité entre données d'entraînement et de test.

0 Introduction

Le projet consiste à prédire, à partir de mesures issues de capteurs (X), la probabilité de présence de plusieurs gaz (y). Les conditions physiques du jeu de test (notamment l'humidité) diffèrent de celles du jeu d'entraînement, rendant la généralisation des modèles particulièrement difficile. Les données totalisent environ 330 000 échantillons et 37 variables, ce qui permet des expérimentations locales avec différents modèles.

Dans tous les modèles développés, les colonnes `ID` ont été supprimées de X et de y, car elles ne portaient aucune information utile pour la prédiction. Les sorties des modèles ont ensuite été systématiquement **bornées dans l'intervalle [0,1]**, afin d'assurer une interprétation cohérente en tant que probabilités de présence de gaz.

1 Première partie : Modèles de base et premières améliorations

Une première approche linéaire a servi à valider le pipeline et le format de rendu. L'algorithme **AdaBoost** a ensuite été mis en œuvre pour tirer parti d'arbres faibles sur données tabulaires.

La **métrique métier** a été définie de façon à *pénaliser plus fortement les faux-négatifs*, afin de privilégier la détection des gaz effectivement présents. Sa définition repose sur une erreur quadratique pondérée : les cas positifs ($y_{\text{true}} \geq 0.5$) sont amplifiés par un facteur $1 + 0.2$. Une fonction personnalisée `modif_predictions` a été introduite pour corriger les biais de sortie, améliorer la calibration des probabilités prédites et essayer de réduire le nombre de faux-négatifs.

Enfin, une étude de `y_train` a mis en évidence plusieurs colonnes constantes ou identiques, supprimées pour éviter des apprentissages artificiels dans les modèles linéaires, AdaBoost et XGBoost. En revanche, les modèles de type Random Forest (dans les sections suivantes) donnaient de meilleurs résultats en conservant l'ensemble des colonnes de y, suggérant une meilleure tolérance à la redondance des sorties.

2 Deuxième partie : Validation croisée et comparaison de modèles

Une **validation croisée** a remplacé la simple séparation train/test afin de fiabiliser l'évaluation. Les modèles **XGBoost** et **Random Forest** ont été testés avec différents réglages d'hyperparamètres, mais les performances en test sont restées limitées, suggérant un sur-apprentissage.

Pour les modèles (linéaire, AdaBoost, XGBoost), les variables d'entrée ont été normalisées à l'aide du `StandardScaler` de `scikit-learn`, tandis que les modèles de type **Random Forest** ont été entraînés sur les données brutes, puisqu'ils sont insensibles au changement d'échelle des variables.

L'analyse de la distribution de X et X_{test} a révélé un décalage notable, en particulier sur la variable d'`humidité`, expliquant les difficultés rencontrées.

3 Troisième partie : Refonte et nouvelles expérimentations

Dans cette partie, seule la métrique **RMSE** a été utilisée, car elle est très proche de la métrique métier et le vrai y_{true} du jeu de test n'était pas accessible.

Une nouvelle phase a été engagée avec le **Random Forest benchmark**. Les tests de suppression de colonnes de y se sont révélés défavorables. Des essais de **feature engineering** et de **PCA** n'ont pas permis d'amélioration significative.

Des essais d'**importance weighting** ont été réalisés sur l'ensemble des lignes de X et X_{test}, en prenant en compte toutes les colonnes. Cette approche n'a pas permis d'améliorer les performances, peut-être qu'un ciblage plus spécifique, par exemple uniquement sur la colonne `Humidity`, aurait été plus pertinent compte tenu du décalage de distribution identifié.

Des techniques de sélection de variables avec `SelectKBest` ont été explorées, en utilisant `f_regression` et `mutual_info_regression` sur $y.mean(axis = 1)$. Les résultats n'étaient pas identiques, suggérant que certaines variables possèdent une corrélation non-linéaire significative mais une information linéaire faible par rapport à $y.mean(axis = 1)$. Cette observation justifie l'usage de modèles capables de capturer des relations non-linéaires, comme les méthodes de **boosting** ou les **forêts aléatoires**, pour prédire efficacement y.

Enfin, un découpage du dataset selon un seuil d'humidité et l'entraînement de deux modèles, Random Forest, légèrements optimisés sur les données d'entraînement n'ont pas apporté de gain notable sur le test.

4 Conclusion

Ce projet a permis d'explorer plusieurs familles de modèles — des approches linéaires aux méthodes de boosting et de forêts aléatoires — tout en développant un pipeline complet d'entraînement, d'évaluation et de calibration. Les différentes expérimentations ont mis en évidence la complexité du problème de **data shift** entre les ensembles d'entraînement et de test, notamment sur la variable `Humidity`, qui affecte directement la capacité de généralisation des modèles.

Malgré de nombreuses tentatives d'adaptation (pondération, découpage du dataset, sélection de variables), je n'ai pas réussi à trouver une solution pleinement satisfaisante pour corriger ce décalage de distribution. Le meilleur modèle final correspond au **benchmark Random Forest**, dans lequel les colonnes `ID` de X et de y ont été supprimées et les prédictions ont été **clippées dans l'intervalle [0,1]** afin de représenter des probabilités cohérentes. Mon meilleur score public est 0.1551. Ce modèle, simple mais robuste, constitue une base stable pour de futures améliorations orientées vers la correction du shift de distribution.