CentraleSupélec

Apprentissage Automatique (2020-2021)

RAPPORT DE PROJET
Kaggle Challenge Drinking Water Potability

Membres du groupe 3
Jérôme Auguste
Marco Boucas
Ariane Dalens

Professeur
Myriam Tami

GitHub

November 6, 2021

1 Introduction

1.1 Contexte

Outre les problèmes économiques ou politiques, l'accès à l'eau potable est une problématique toujours d'actualité notamment dans les pays en voie de développement. C'est avant tout un élément indispensable et essentiel à la santé, un droit fondamental de la personne et un élément d'une politique efficace de protection de la santé. Malheureusement, tous n'ont pas cette chance, en effet, certains pays sont dépourvus d'un approvisionnement et assainissement en eau, ce qui impacte non seulement la situation économique du pays, mais également les capacités de soin de la région.

1.2 Problématique

L'objectif est de trouver un modèle permettant de prédire si l'eau d'un cours d'eau est potable ou non afin de savoir si nous devons investir dans le développement d'un réseau d'assainissement de l'eau.
La tâche sera donc de la **classification binaire**, nous assignerons dans la suite de ce projet la valeur 1 pour "Potable" et 0 pour "Non Potable".

1.3 Les objectifs du projet

L'objectif principal du projet est d'établir un modèle de prédiction fiable de potabilité ($Potability \in \{0, 1\}$) de l'eau selon ses 9 features numériques données:

Feature	Type
ph	float
Sulfate	float
Trihalomethanes	float
Hardness	float
Solids	float
Chloramines	float
Conductivity	float
Organic_carbon	float
Turbidity	float
Potability (target)	categorical

Compte tenu de l'aspect santé de cette problémateique, il nous semble primoridial de privilégier des modèles qui ne vont pas prédire la potabilité d'une eau qui ne l'est pas, c'est pourquoi il faut minimiser les faux positifs, la métrique que nous allons privilégier est la **précision** ($precision = \frac{TP}{TP+FP}$, avec TP pour Vrai Positif et FP for Faux Positif). Toutefois, cette métrique seule introduit un biais dans le cas de données déséquilibrées (ex: une grande proportion d'eau potable). Il est donc important de prendre en compte le F1-score (F1-score $= 2\frac{precision*recall}{precision+recall}$) qui pourra équilibrer la métrique.

1.4 Stratégie retenue

Après étude des différents modèles, nous avons sélectionné 3 d'entre eux (les 3 meilleurs en termes de précision et F1-score): **Random Forest**, **SVM** et **K-nn**.

Ceux-ci sont utilisés collaborativement par un système de vote pour tirer les avantages de chacun. Le vote se fait de manière "hard", c'est à dire que la classe est sélectionnée à partir du vote majoritaire des modèles.

Nous obtenons avec ce modèle un f1-score pondéré de **61%** et une précision de **56%** soit respectivement 7 et 13 points au dessus de notre baseline.

2 Stratégie de résolution du problème

Pour attaquer ce problème, nous avons procédé de la manière suivante:

1. Analyse des données, des outliers et des missing values & feature engineering
2. Création d'une pipeline de preprocessing des données
3. Exploration des différents modèles
4. Sélection des meilleurs modèles et recherche des hyperparamètres
5. Conclusion sur les modèles utilisés

2.1 Analyse des données & traitement des outliers

2.1.1 Analyse exploratoire

Nous avons cherché en premier lieu à observer les distributions de nos données et observons que globalement les distributions de chacun des features présentent des **formes gaussiennes**, de moyennes et écart-types différents.

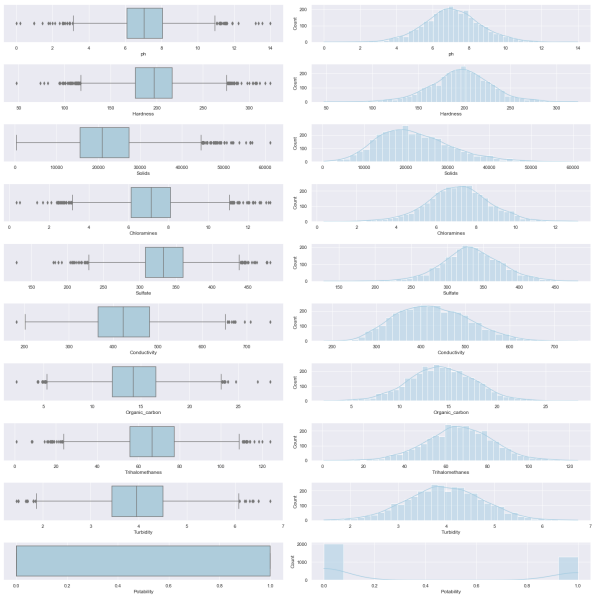

Figure 1: Distribution des différents features

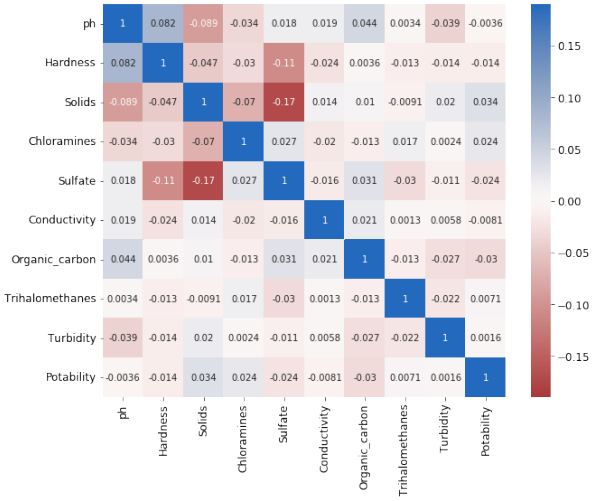

Figure 3: Matrice de corrélation des features

Ceci nous amènera donc à mettre à l'échelle nos données en utilisant un **StandardScaler**.

On observe aussi une répartition équivalente en séparant les deux classes (potable et non potable) :

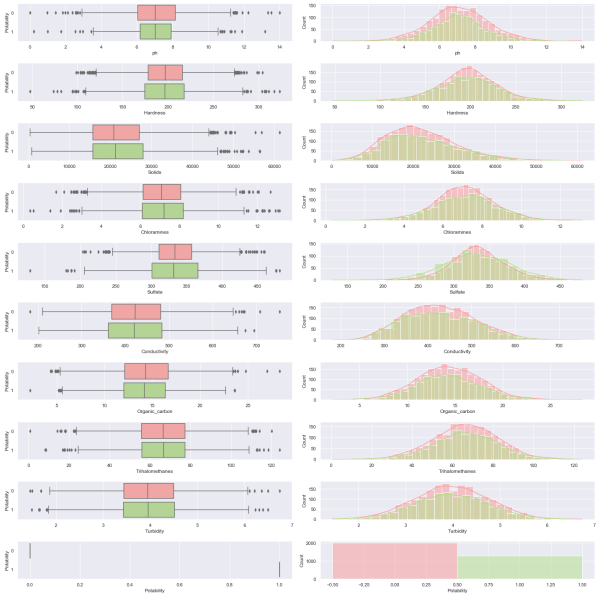

Figure 2: Distribution des différents features par label

2.1.2 Gestion des données manquantes

Dans l'analyse des données, nous nous concentrons sur les valeurs manquantes qui pourraient entraîner un biais dans le modèle. Voici la répartition des données manquantes:

Feature	n° of Null values	% of data
ph	491	15.0 %
Sulfate	781	23.8 %
Trihalomethanes	162	4.9 %

On observe que 3 de nos features présentent des valeurs manquantes (*Sulfate, ph, Trihalomethanes*). Le pourentage de valeurs manquantes est trop élevé pour les ignorer (et les supprimer). On se focalisera donc sur une **méthode d'imputation**.

Il faut par la suite vérifier la source de ces valeurs manquantes afin de les catégoriser comme MCAR, MAR, MNAR. Pour cela nous analysons la répartition des données manquantes par feature:

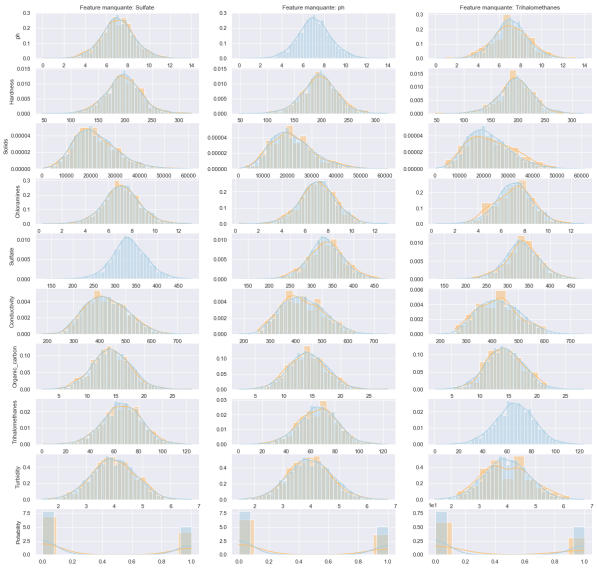

Figure 4: Distribution des features sur les données incomplètes

On remarque alors que la distribution des données manquantes ne semble pas dépendante de la valeur d'autres features (la distribution ne semble pas changer sur les valeurs incomplètes). Nous nous laissons donc le choix de la méthode d'imputation. Toutefois, afin de conserver la distribution des features et ne pas introduire un biais trop élevé, nous avons décidé d'utiliser une **imputation multivariée non stochastique**. Bien que cette méthode risque de renforcer la corrélation entre les features, cette dernière est préservée par le faible taux de données incomplètes.

2.1.3 Gestion des outliers

La répartition des valeurs pour les eaux potables [Figure 2] permet de détecter des outliers (en dehors de $Q1 - 1.5 * IQR$ et $Q3 + 1.5 * IQR$). Ainsi, nous avons décidé de supprimer les outliers (uniquement pour Potability = 1) ce qui représente 10% des données tout en conservant le dataset.

2.2 Exploration de modèles

Nous avons cherché à tester de manière assez exhaustives les différents modèles de classifications usuels

2.3 Résultats

2.3.1 Définition d'une baseline

Certaines recommandations sont fournies par l'Organisation Mondiale de la Santé quant à la définition de la potabilité de l'eau. Notamment:

- Valeur pH: $6.5 < \text{pH} < 8.5$
- Solids: recommandé $TDS < 500$ mg/L, limite $TDS < 1000$ mg/L
- Chloramines: Concentration de Chlorine < 4 mg/L
- Conductivity: < 400 μS/cm
- Organic Carbon: TOC < 2 mg/L dans l'eau potable/traîtée, TOC < 4 mg/L dans l'eau utilisée pour le traitement
- Trihalomethanes: Niveau THM < 80 ppm
- Turbidity: recommended 5.00 NTU

En mettant nos données et nos recherches au regard de ces recommandations, nous pourrons discuter de la validité de ces limites, ainsi que celle de notre modèle.
Nous avons défini un Classifier basé sur ces données qui vérifie que l'eau tient compte des recommandations de valeurs de pH, de Trihalomethanes et de Conductivité. Les autres règles ne semblent pas applicables car l'ordre de grandeur des données ne correspond pas à celle des règles (ce qui pourrait s'expliquer par une différence d'unité que nous ne connaissons pas).

2.3.2 Résultats des modèles, tableau synthétique

	name	cv_accuracy_score	cv_f1_score	cv_precision_score	cv_recall_score
4	SVC	0.630682	0.526340	0.516925	0.539425
3	LogisticRegression	0.509659	0.440650	0.390864	0.505163
5	KNeighborsClassifier	0.619318	0.432740	0.503768	0.380310
7	RandomForestClassifier	0.652273	0.400769	0.591136	0.291177
1	DummyClassifier	0.525568	0.387653	0.383581	0.368491
2	DecisionTreeClassifier	0.618182	0.371685	0.507627	0.302897
8	ExtraTreesClassifier	0.632955	0.323120	0.526382	0.285252
10	GradientBoostingClassifier	0.619318	0.317933	0.511796	0.228778
9	AdaBoostClassifier	0.603409	0.310686	0.462858	0.234848
0	OMSClassifier	0.573864	0.264706	0.213740	0.342466
6	GaussianNB	0.611932	0.189716	0.474419	0.118883

Figure 5: Comparaison des modèles selon 4 scores

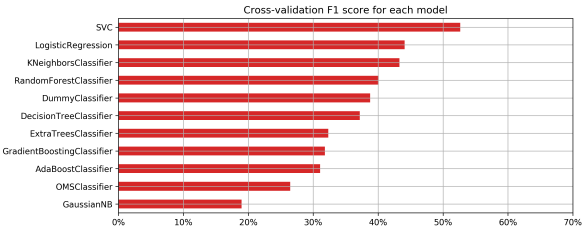

Figure 6: Classement des modèles selon le F1-score

En comparant les différents modèles présents dans le tableau récapitulatif [Figure 5] sur les scores

sélectionnés (F1-score et precision), on remarque qu'aucun des modèles (à l'exception des SVM) n'est particulièrement plus performant que les autres et qu'ils restent dans une fourchette de scores assez fine. Par ailleurs les règles appliquées par l'OMS ne permettent pas de classifier efficacement les données.

2.3.3 Fine tunning des modèles les plus performants

Nous avons gardé les trois modèles les plus pertinents au vu des résultats précédents: RandomForest, SVM et K-NN (en excluant la LogisticRegression en raison d'une accuracy et une precision en dessous des autres modèles). Nous chercherons donc maintenant à les fine tuner pour optimiser les résultats.
Dans le cadre de notre recherche d'hyperparamètres, nous allons procéder par gridsearch, en évaluant le modèles pour toutes les combinaisons de paramètres sélectionnés et en les classant par score. A la fin, nous sélectionneront le modèle qui performe le mieux sur l'ensemble de ces combinaisons de paramètres.
Afin de garantir un choix le plus équilibré possible, nous calculerons à la fois l'accuracy, le f1-score ainsi que la précision du modèle, et nous chercherons le modèle qui performera le mieux en terme de f1-score, sous condition que ses performances ne soient pas trop dégradées du point de vue des autres métriques.

K-Nearest Neighbours
K-Nearest Neighbours (our *Knn*) est un modèle de machine learning **transductif** dont la méthode de prédiction revient à chercher les éléments du set de données les plus proches (le choix de la distance est un hyperparamètre), et de retourner la classe prédominante pour ces **K** voisins.
Pour ce modèle, certains paramètres sont connus pour avoir une influence majeure sur la résultante finale (et donc les performances du modèle). Dans le cadre de notre recherche d'hyperparamètres, nous allons chercher les valeurs de K (nombre de voisins), que le choix de la distance utilisée *metric* ainsi que le choix de la classe prédominante (qui peut se faire avec des poids uniformes ou inversement proportionnel à la distance).

Suite à la grid search, nous sélectionnons les paramètres suivants qui maximisent le f1-score tout en gardant des scores très bon sur la précision et l'accuracy:

K	10
Pondération	Inverse de la distance

Pour ce modèle, nous obtenons un score de f1-score de **61%** moyenné, avec une accuracy de **64%** et une précision de **52%**.

Random Forest
Dans le cadre du cours, nous avons pu étudier les Random Forest (ou *Forêts aléatoires*), ce sont des estimateurs qui adaptent un certain nombre d'arbres de décision sur divers sous-échantillons de l'ensemble de données et utilise la moyenne pour améliorer la performance globale du modèle tout en évitant à ce dernier d'overfit sur le training set.
Compte tenu du risque d'overfit important, nous surveillerons bien évidemment le nombre d'estimateurs ainsi que la profondeur maximale de chacun d'entre eux.

Suite à la grid search, nous sélectionnons les paramètres suivants qui maximisent le f1-score tout en gardant des scores très bon sur la précision et l'accuracy:

Profondeur Maximale	20
Nombre d'estimateurs	100

Pour ce modèle, nous obtenons un score de f1-score de **63%** moyenné, avec une accuracy de **66%** et une précision de **57%**.

SVM
Les SVM (ou *Support -Vector Machines*) qui visent à trouver un hyperplan dans l'espace des features qui sépare au mieux les données de training, par la recherche de vecteurs de support qui maximiseront la marge entre les 2 catégories. Dans le cadre de données non linéairement séparables, comme dans le cas présent au vu des différents graphiques, on peut néanmoins espérer qu'il existe une transformation vers un espace dans lesquel les données le sont. L'implémentation réelle ne nécessite pas d'expliciter cette transformation, mais d'obtenir le produit scalaire dans cet espace secondaire (technique connue sous le nom du *kernel trick*).
Dans le cadre de notre optimisation de modèle, nous nous intéresseront particulièrement à 3 paramètres: C, le coefficient de régulation, le choix du kernel ainsi que le fait de prendre en compte le fait que les données soient "unbalanced" (ce qui est notre cas) (*class_weight*).

Suite à la grid search, nous sélectionnons les paramètres suivants qui maximisent le f1-score tout en gardant des scores très bon sur la précision et l'accuracy:

C	20
Kernel	rbf

Pour ce modèle, nous obtenons un score de f1-score de **63%** moyenné, avec une accuracy de **64%** et une précision de **53%**.

Conclusion
Voici les scores obtenus par les 3 modèles par cross-validation:

Model	F1-score	Accuracy	Precision
Knn	63%	64%	53%
Random Forest	63%	66%	57%
SVM	63%	64%	53%

	precision	recall	f1-score	support
0	0.67	0.84	0.75	464
1	0.54	0.32	0.40	279
accuracy			0.64	743
macro avg	0.61	0.58	0.57	743
weighted avg	0.62	0.64	0.62	743

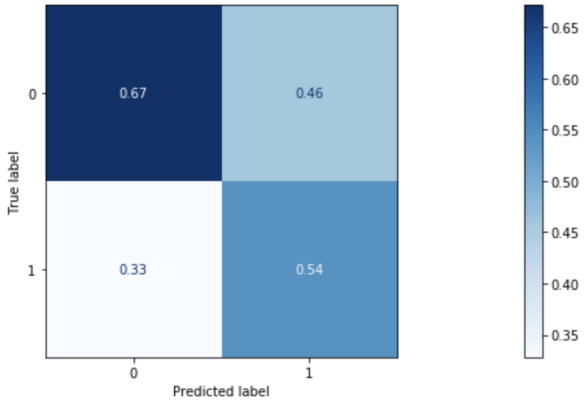

Figure 7: Modèle final

2.3.4 Modèle final - Voting classifier

Comme nous avons pu le voir, les résultats, bien que corrects, ne sont pas significativement beaucoup plus élevés que les résultats obtenus avec le modèle découlant des règles de l'OMS. Pour essayer de remédier à cela, et construire un modèle plus robuste face aux perturbations dans les données, nous avons décidé de continuer dans notre recherche et de produire un Voting Classifier.
Le principe est simple, et peut se résumer par ce fameux proverbe homérique *"L'union fait la force"*. Plutôt que devoir subir les forces et faiblesses d'un seul modèle, nous pouvons combiner plusieurs modèles ensemble et résoudre le problème de manière démocratique (sans vote blanc).
Basé sur les 3 modèles précédents, agrémentés des paramètres obtenus suite à une grid search, nous pouvons donc construire ce modèle final.
Afin de pouvoir comparer ce modèle, nous proposons également en parallèle le modèle construit à partir des régulations de l'OMS, pour pouvoir comparer notre approche avec un modèle qui peut servir de référence.

	precision	recall	f1-score	support
0	0.62	0.81	0.70	1998
1	0.43	0.21	0.29	1278
accuracy			0.58	3276
macro avg	0.52	0.51	0.49	3276
weighted avg	0.54	0.58	0.54	3276

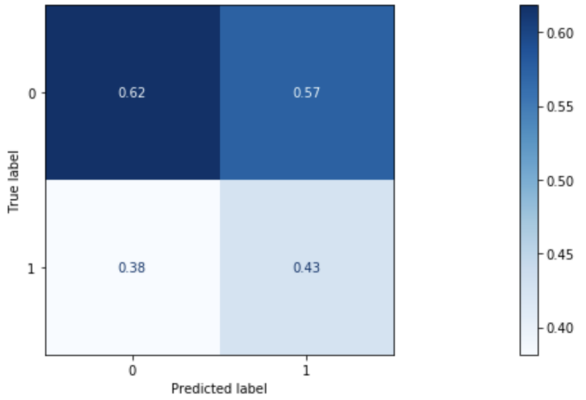

Figure 8: OMS model

2.3.5 Explicabilité modèle

Il est vrai que compte tenu du fait que le modèle soit une combinaison de différents modèles, l'explicabilité du modèle final est bien plus obscure que le modèle OMS. Tout d'abord, la couche *VotingClassifier* ne rajoute que peu de complexité, car très facile à expliquer et à comprendre.
Concernant le Knn, ce modèle est explicite de part la méthode de prédiction, il suffit de présenter les

éléments du dataset les plus proches ainsi que leur potabilité.
Pour le Random Forest et le SVM par contre, la tâche se révèle plus ardue. Autant pour le Random Forest, il est théoriquement possible d'afficher les arbres de décision de la forêt, et ainsi présenter le raisonnement du modèle ainsi que les raisons d'une telle classification.

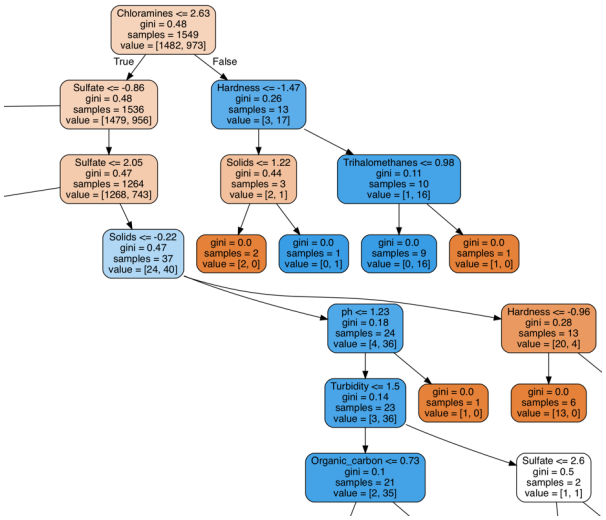

Figure 9: Extrait d'un Arbre de Décision

Néanmoins, une telle présentation laisse à désirer, et compte tenu du fait que ce modèle contient plusieurs dizaines d'arbres, on ne peut considérer un tel modèle correct d'un point de vue explicabilité.
Concernant le SVM, nous pouvons abandonner tout espoir de rendre un tel modèle explicable. Considérant les méthodes actuellement disponibles en ligne, excepté l'affichage de *"l'importance"* de chaque feature, on peut dérouler le raisonnement et les raisons qui ont poussé le modèle à prédire une valeur.
Finalement, du point de vue de la pipeline et du preprocessing, il n'y a pas de difficultés particulières de ce côté, car pas de raisonnement spécifique (sauf pour les valeurs anormales, qui sont exclues d'avance et qui peuvent potentiellement être causé par des erreurs de manipulation lors de la prise de mesures.

2.4 Pour aller plus loin

On remarque que le modèle OMS est particulièrement peu performant et que les performances de notre modèles laissent à désirer. Ce qui est assez étonnant et laisse à penser que les données ne sont pas forcément réelles et possiblement partiellement erronées, il serait intéressant, autant pour améliorer notre connaissance des données que pour améliorer l'explicabilité finale du modèle, de s'intéresser à la captation des données préalable.
Concernant le choix des features présentes dans le dataset et chargées de représenter chaque échantillon d'eau. Car après certaines recherches sur le site de l'OMS et autres réglementations, il pourrait être intéressant de regarder certaines features qui peuvent plus facilement départager les eaux entre potable et non-potable.
Considérant l'impact humain actuel, il pourrait être intéressant de regarder l'empreinte carbone d'un tel modèle (en inférence et au cours de l'entraînement et de la recherche des modèles), afin de regarder si l'utilisation de machine learning dans ce cas d'utilisation est réellement pertinente.

3 Conclusion

Pour conclure sur ce projet, qui pour nous s'est révélé être une mise en pratique des connaissances du cours et un réel apprentissage, il faut néanmoins tempérer les résultats et questionner la pertinence de l'usage d'un modèle de machine learning dans de telles situations.
Tout d'abord, au vu des résultats obtenus et surtout de l'importance de la tâche en question (on parle en effet de détecter si une eau est propre à la consommation), il n'est pas forcément judicieux d'utiliser de telles méthodes pour ce genre de situation. Une expertise professionnelle et un diagnostique plus poussé semblent nécessaires afin d'aboutir à une conclusion plus pertinente et potentiellement explicable.
Il faut néanmoins relever, que même si les données sont loin d'être parfaites, et bien que l'on ignore leur méthode de prélèvement, elles ne sont pas complètement aléatoires. On peut en effet noter que nos modèles sont meilleurs qu'une solution *dummy*, et donc que les features sont effectivement en partie responsable ou non de la potabilité d'une eau. Néanmoins, cela nous amène à nous questionner sur le choix de telles features, car certaines mesures, comme la concentration en nitrates, en plomb et autres métaux lourds peuvent également entrer en compte (en tout cas, prises en compte dans les réglementation de l'OMS) et ainsi pourraient nous aider à mieux déterminer la potabilité de l'eau.