Estimation du volume consommé à partir de capteurs de température

Sacha Alidadi Heran

21 Aout 2023

Contents

1 Introduction

1.1 Présentation de l'entreprise

BDR Thermea est un leader mondial en matière de solutions de chauffage intelligentes. Cette entreprise est née en 2009 de la fusion de 3 entreprises en 2009 Baxi, De Dietrich, Remeha.

Elle est connue pour ses solutions de chauffage et d'eau chaude, fournissant des produits à travers l'Europe de l'Ouest, notamment au Royaume-Uni, en France, en Allemagne, en Espagne, aux Pays-Bas et en Italie.

L'entreprise se distingue par ses chiffres impressionnants. Chaque année, 1,4 million d'utilisateurs finaux choisissent les produits de BDR Thermea. Elle emploie plus de 6 100 personnes et ses produits sont vendus dans plus de 100 pays, générant un chiffre d'affaires de 2,1 milliards d'euros.

BDR Thermea se compose de plusieurs marques à travers le monde, y compris Baxi, De Dietrich, Remeha, Brötje, Chappee, Sofath, Oertli, Baymak, Heatrae Sadia, Megaflo, et Potterton. Mon stage a eu lieu au sein de BDR Thermea France dans l'ancien siège social de De Dietrich.

Au sein de cette entreprise, j'étais rattaché au département R&D BDU PAC chargé des développements de Pompes à chaleur.

1.2 Contexte

Les ingénieurs de BDR Thermea France, en particulier ceux travaillant sur le chauffe-eau Elensio, cherchent à comprendre et optimiser le volume d'eau consommé par leurs produits. Voici à quoi ressemble le produit Elensio

Figure 1: Photo du chauffe-eau Elensio

Cette connaissance approfondie est essentielle à l'amélioration continue des

produits. Toutefois, l'utilisation de débitmètres pour suivre le volume d'eau présente des défis. Non seulement ces équipements ont un coût élevé, mais leur production et leur utilisation ont également une empreinte carbone significative.

Pour résoudre ce problème, il est venu l'idée d'utiliser les températures du ballon d'eau chaude pour prédire le volume d'eau consommée. Cette idée est intéressante, car quand on consomme de l'eau dans le ballon, de l'eau chaude sort du haut du ballon et de l'eau froide entre en bas du ballon ce qui diminue la température en bas. Il y a donc une corrélation entre ces deux variables.

Dans une tentative précédente de résoudre ce dilemme, un modèle d'apprentissage automatique avait été développé. Il visait à déduire le volume d'eau consommé en se basant sur les températures mesurées à différents niveaux du ballon. Cependant, les performances de ce modèle étaient loin d'être satisfaisantes, nécessitant ainsi une révision et une amélioration.

1.3 Missions et objectifs

Face à ce défi, la mission assignée durant ce stage est claire : développer un modèle plus performant et précis pour estimer le volume d'eau consommé. Toutefois, cette mission ne sera pas sans contraintes. En effet, le matériel informatique embarqué dans le chauffe-eau Elensio impose des limites en termes de puissance de calcul. Ainsi, bien que les modèles d'apprentissage profond soient puissants, ils sont également gourmands en ressources. C'est pourquoi ils seront à éviter dans le cadre de ce projet. La quête sera donc celle d'un équilibre entre précision et efficacité, afin d'assurer une solution viable pour les chauffe-eaux créés par l'entreprise.

1.4 Approche utilisée

Lors du projet entre Mars et Mai, nous avons utilisé une approche à deux modèles. Le premier modèle est chargé de détecter un sous tirage à l'instant **t** en utilisant les températures du ballon à différents instants. Le deuxième modèle est chargé de déterminer le volume d'eau consommé à l'instant t en utilisant les résultats du premier modèle ainsi que les températures du ballon à l'instant **t**.

Cette approche s'est avérée peu intéressante pour deux raisons possibles:

- Pas assez de données
- Une approche à revoir

Alors entre ce projet et le stage, nous avons récolté plus de données. Maintenant, nous avons des données sur deux mois au lieu d'un.

Mais nous avons aussi décidé de changer d'approche. Au lieu d'utiliser deux modèles, nous utiliserons un seul modèle mais qui cette fois va détecter le volume d'eau consommé sur une tranche horaire.

Pour que le modèle n'ait pas trop de données à traiter, nous avons créé différentes features qui seront abordés dans la section suivante.

2 Contributions

2.1 Exploratory Data Analysis

2.1.1 Présentation générale du dataset

Le jeu de données, appelé "proc", contient une variété d'informations enregistrées toutes les secondes, du 28 février 2023 au 13 avril 2023. Ces données viennent d'un ballon d'eau chaude utilisé par un client de BDR Thermea. Elles comprennent des variables telles que la consommation d'électricité cumulée, le statut de chauffage du ballon, l'humidité et la température de l'air à l'extérieur du ballon, la consommation d'eau et les températures de l'eau à différents niveaux du ballon.

Durant le projet, l'humidité de l'air ne sera pas utilisée. Tout simplement car c'est une mesure physique qui n'est pas liée au problème. Elle a été utilisée pour d'autres projets mais on a oublié de l'enlever du dataset. Et il en va de même pour la consommation d'électricité.

Aussi nous n'utiliserons pas certaines températures comme celles des capteurs placés contre la cuve, car elles sont moins précises que celles des capteurs immergés dans la cuve.

Ce jeu de données est appelé "proc" pour "processed", ce qui veut dire que les données ont été filtrées. En effet, un filtre de Butterworth a été utilisé pour traiter ces données. Ce filtre, nommé d'après l'ingénieur britannique Stephen Butterworth, est une méthode de traitement du signal utilisée pour supprimer le bruit dans les données. Dans notre contexte, cela signifie qu'il nous aide à éliminer les fluctuations mineures et indésirables dans nos enregistrements de température, de consommation d'eau et d'électricité, nous permettant ainsi d'avoir une vue plus claire des tendances sous-jacentes.

Plus spécifiquement, on a appliqué un filtre passe bas enlevant toutes les fréquences de plus de 5 minutes. Ceci pour réduire le bruit de nos données et pour faciliter l'apprentissage du modèle.

Notre jeu de données contient 29 colonnes initialement, chacune représentant une caractéristique différente de notre système de ballon d'eau chaude.

Au total, il y a 18 colonnes de températures, 1 d'humidité, 1 sur le statut de chauffage du ballon, 5 sur la consommation d'eau, 1 sur la température de l'air et 1 sur la consommation de l'électricité. Il y a aussi 2 autres variables qui ne sont pas intéressantes pour le modèle.

Toutefois, pour améliorer la performance de notre modèle, nous avons décidé de créer des features supplémentaires à partir de ces données, ce qui nous donne 22 colonnes de plus. Par conséquent, notre dataset final comprend 51 colonnes au total.

Maintenant que nous avons ces informations, nous pouvons afficher les données de températures et de volume.

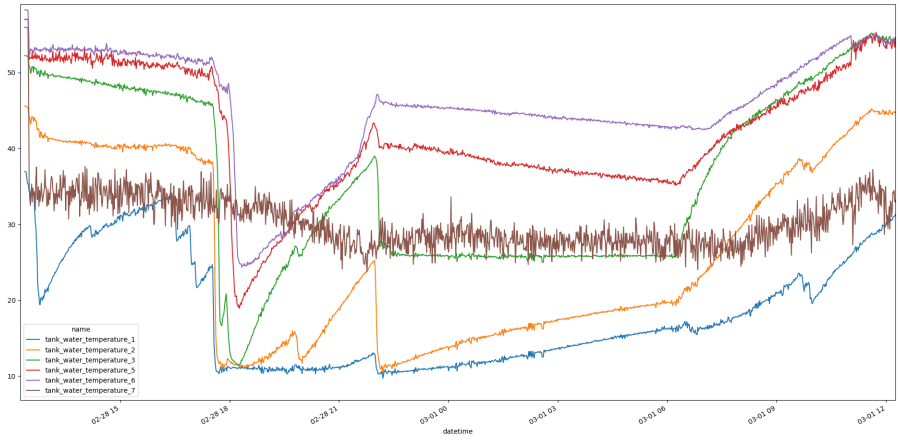

Figure 2: Graphique des températures des 6 capteurs à l'intérieur du ballon sur une journée. Plus le numéro du capteur est petit, plus il est placé en bas du ballon

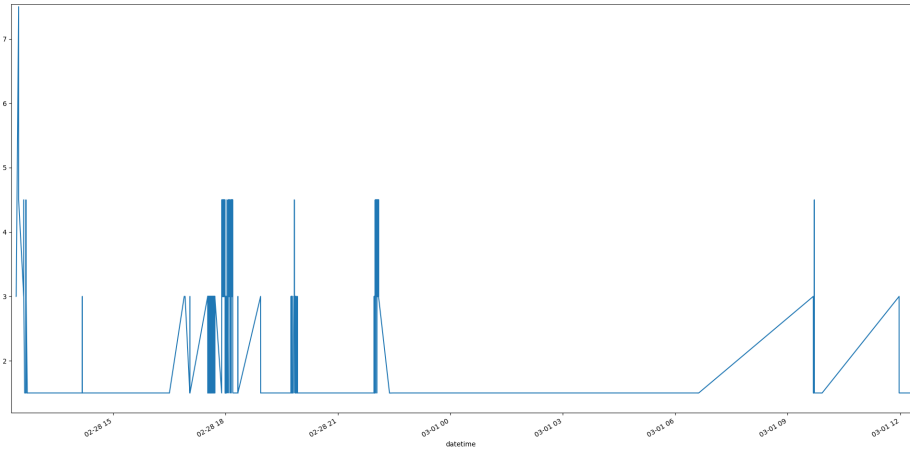

Figure 3: Graphique du volume consommé par le ballon sur une journée

Nous remarquons qu'à chaque pic dans le second graphique (qui correspond à un sous-tirage), la température dans le ballon diminue. En particulier, pour les capteurs placés en bas du ballon. Ce qui confirme que ces deux variables sont liées.

2.1.2 Identification de la target

Notre objectif est de prédire le volume d'eau consommé par le ballon d'eau chaude dans une période de X secondes, en utilisant les températures enregistrées pendant cette période. Cette quantité d'eau consommée, nommée "inlet_water_cumsum" dans notre dataset, est donc notre target (ou en termes de science des données, notre "variable target").

2.1.3 Types de variable

La majorité des variables de notre jeu de données sont des nombres à virgule flottante, ce qui signifie qu'elles peuvent représenter un large éventail de valeurs avec une précision décimale. Cependant, la variable "heat_command" agit effectivement comme une variable binaire, enregistrant un "0" quand le ballon n'est pas en train de chauffer, et un "1" quand il l'est.

De plus, une exception importante est la variable "category", qui est un objet. Elle contient des étiquettes catégorielles indiquant le niveau de sous-tirage d'eau à un moment donné. Les cinq catégories possibles sont "none" (pas de tirage), "low" (faible tirage), "medium" (tirage moyen), "high" (fort tirage), et "huge" (énorme tirage). Cette variable permet d'ajouter une dimension qualitative à notre analyse et permet de mieux répartir nos données d'entraînement et de test. L'idée est de diminuer le surapprentissage, en effet, si nos données d'entraînement ne contient que des sous-tirages de type "none" et "low", notre modèle va être bon sur ces catégories mais va être très mauvais sur les autres catégories.

Un autre point important sur la variable category, c'est que celles-ci ont été choisies de manière arbitraire. On verra que ça a engendré quelques problématiques plus tard.

2.1.4 Présentation des différentes colonnes et des features créés

Chaque colonne de notre dataset représente une caractéristique différente du système de ballon d'eau chaude, y compris des informations sur la consommation d'électricité et d'eau, le statut de chauffage, l'humidité de l'air, la température de l'air et de l'eau, et la catégorie de sous-tirage à un moment donné.

Avant d'aller plus loin, précisons ce qu'on a appelé une tranche horaire. Il s'agit d'une période de temps non glissante. Par exemple une période de 16:20 à 16:40.

Et durant ce projet, nous avons surtout travaillé sur des tranches horaires de 20 minutes. Comme notre dataset contient des données toutes les secondes et qu'il y a 1200 secondes dans 20 minutes, notre modèle devrait traiter au moins 1200 entrées. C'est beaucoup trop pour un modèle censé tourner sur un ordinateur embarqué.

Alors pour optimiser le temps de calcul de notre modèle, nous avons créé 22 nouvelles features à partir des données existantes. Ces features ont été crééeas à partir de nos données de température ainsi que l'énergie du ballon. Celles-ci représentent différentes statistiques, calculées sur une tranche horaire, de la

température de l'eau et de l'énergie enregistrées à différentes hauteurs dans le ballon. Ces statistiques comprennent la différence de température ou d'énergie entre le début et la fin de la tranche horaire (Toutes les variables commençant par delta), la différence entre le maximum et le minimum enregistré pendant la tranche horaire(Toutes les variables commençant par gap), la différence du maximum avec la tranche horaire précédente (Toutes les variables commençant par diffmax), et la différence de la médiane avec la tranche horaire précédente (Toutes les variables commençant par diffmed). L'intérêt de ces features est de prémacher le travail de notre modèle, par exemple avec les variables delta, s'il y a un sous-tirage dans notre tranche horaire, la température globale va diminuer dans le ballon et sera donc plus petite à la fin qu'au début. Ces variables seront donc positives et donc plus élevées comparé au cas où il n'y a pas de sous-tirage.

2.1.5 Identification des variables manquantes

Notre jeu de données initial ne contient aucune valeur manquante. Cependant, lors de la création des nouvelles features, des valeurs NaN (Not a Number) peuvent apparaître pour les tranches horaires où les calculs ne sont pas possibles, par exemple lors de la comparaison avec la tranche horaire précédente pour la première tranche horaire de notre dataset. Dans le processus ultérieur d'entraînement du modèle, ces valeurs NaN devront être traitées de manière appropriée.

2.1.6 Statistiques sur les sous-tirages

Notre variable target, "*inlet_water_cumsum*", indique le volume d'eau consommé sur une tranche de 1200 secondes. Pour mieux comprendre cette variable, nous avons examiné la répartition des différentes catégories de sous-tirage, qui sont déterminées par la quantité d'eau utilisée pendant chaque tranche horaire.

Nous avons constaté que la majorité des tranches horaires (85%) n'ont pas de sous-tirage d'eau ("none"). Les sous-tirages faibles ("low") représentent environ 11% des tranches horaires, tandis que les sous-tirages moyens ("medium"), élevés ("high") et énormes ("huge") ne représentent ensemble que 4% des tranches horaires.

Ce déséquilibre dans notre variable cible présente un défi pour l'entraînement de notre modèle. Par exemple, un modèle naïf qui prédit toujours 0L d'eau utilisée serait techniquement correct 85% du temps, mais il ne serait pas utile dans la pratique car il ne prédit pas correctement les cas de sous-tirage d'eau. De plus, le modèle peut être biaisé en faveur des sous-tirages faibles, qui sont plus courants que les autres types de sous-tirage.

Pour éviter ces biais, il sera important de créer un ensemble d'entraînement qui contient des proportions équilibrées de chaque type de sous-tirage. Les variations quotidiennes dans les types de sous-tirage rendent également cette approche nécessaire. Par exemple, le 23 Février 2023, tous les sous-tirages étaient de type "faible", tandis que le 8 Avril 2023, 93,7% des sous-tirages

étaient de type "énorme". Cela indique que tous les types de sous-tirage sont importants à prédire pour obtenir un modèle performant.

Enfin, si on fait la somme cumulée des volumes par catégories de sous-tirages, on remarque que plus de la moitiée du volume total d'eau consommée vient uniquement des sous-tirages "huge". Comme on peut le voir avec ce diagramme 4

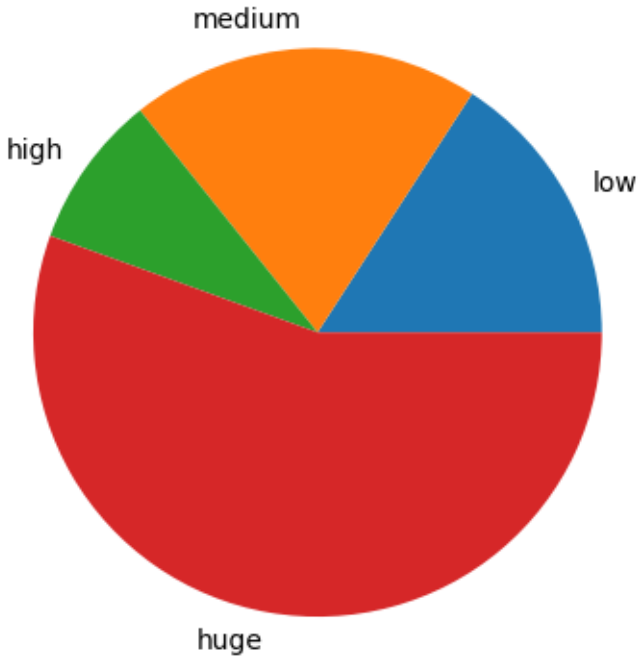

Figure 4: Diagramme en camembert des volumes totaux par type de sous-tirages

Ce diagramme suggère que individuellement, chaque catégorie de sous-tirage est importante. Mais si on regarde sur plusieurs mois, le modèle doit être très bon sur la catégorie "huge".

2.1.7 Compréhension des différentes variables

Pour comprendre les relations entre nos différentes variables, nous avons calculé des matrices de corrélation en utilisant trois coefficients différents : Pearson, Kendall et Spearman.

L'intérêt de dresser ces matrices de corrélation est de comprendre quelles sont les variables les moins corrélées à notre target afin de les éliminer lors de notre phase d'entraînement. Mais aussi de voir les variables les plus corrélées entre elles afin de ne pas entraîner notre modèle sur ces variables corrélées. En

effet, si on ne fait pas ça, le modèle risque d'être en surapprentissage, c'est-à-dire qu'il est très bon sur les données d'entrainement mais pas assez sur les données de test.

Le coefficient de corrélation de Pearson est une mesure de la corrélation linéaire entre deux variables. Il est calculé comme le rapport de la covariance des deux variables à leurs écarts-types respectifs, et varie entre -1 et 1.

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y}$$

Le coefficient de Kendall, également appelé τ de Kendall, est une mesure de la correspondance entre deux classements. Il est calculé comme la différence normalisée entre le nombre de paires concordantes et le nombre de paires discordantes parmi toutes les paires distinctes de l'ensemble de données.

$$\tau = \frac{(n_c - n_d)}{\frac{1}{2}n(n-1)}$$

où n_c est le nombre de paires concordantes, n_d est le nombre de paires discordantes, et n est le nombre total d'observations.

Le coefficient de Spearman, également appelé ρ de Spearman, est une mesure de la corrélation de rang. Il est calculé comme le coefficient de corrélation de Pearson entre les rangs des variables.

$$\rho_{rg_X, rg_Y} = \frac{cov(rg_X, rg_Y)}{\sigma_{rg_X} \sigma_{rg_Y}}$$

où rg_X et rg_Y sont les rangs des variables X et Y.

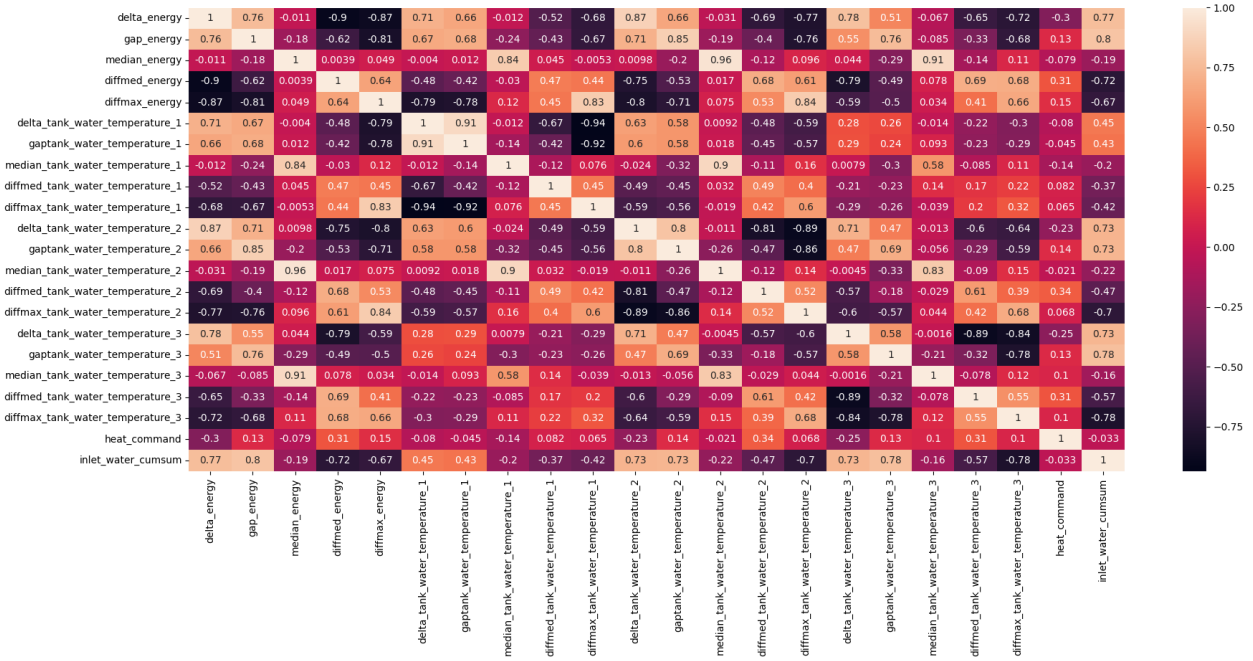

Figure 5: Matrice de corrélation de Pearson

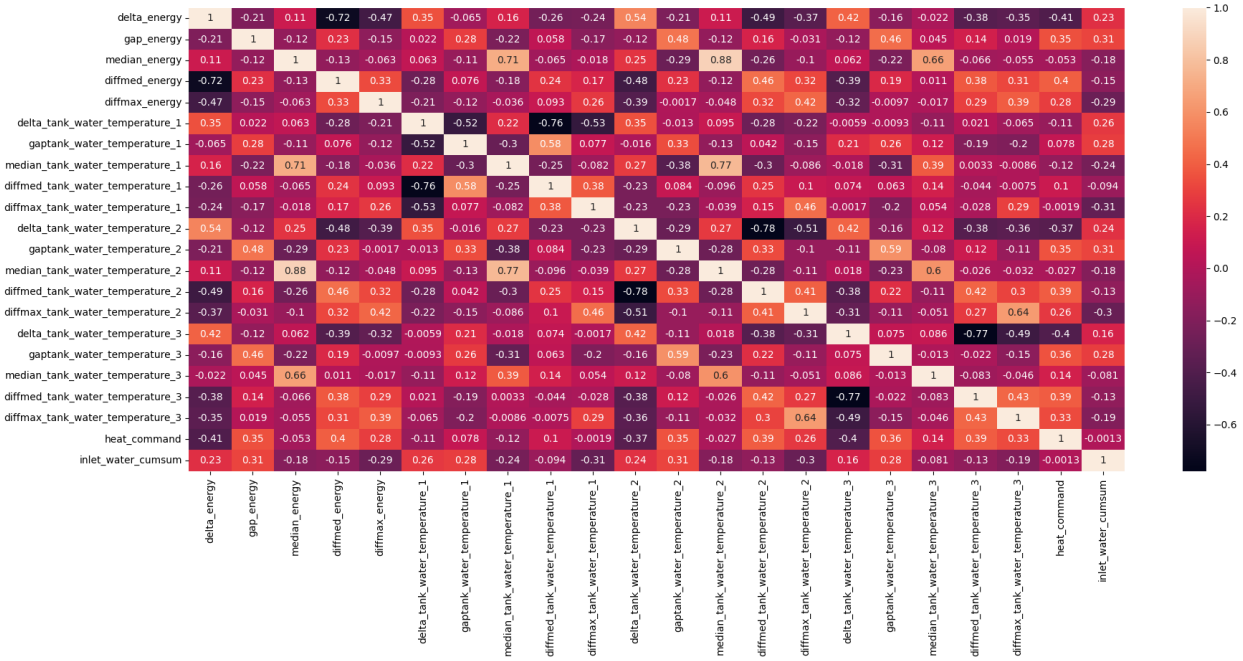

Figure 6: Matrice de corrélation de Kendall

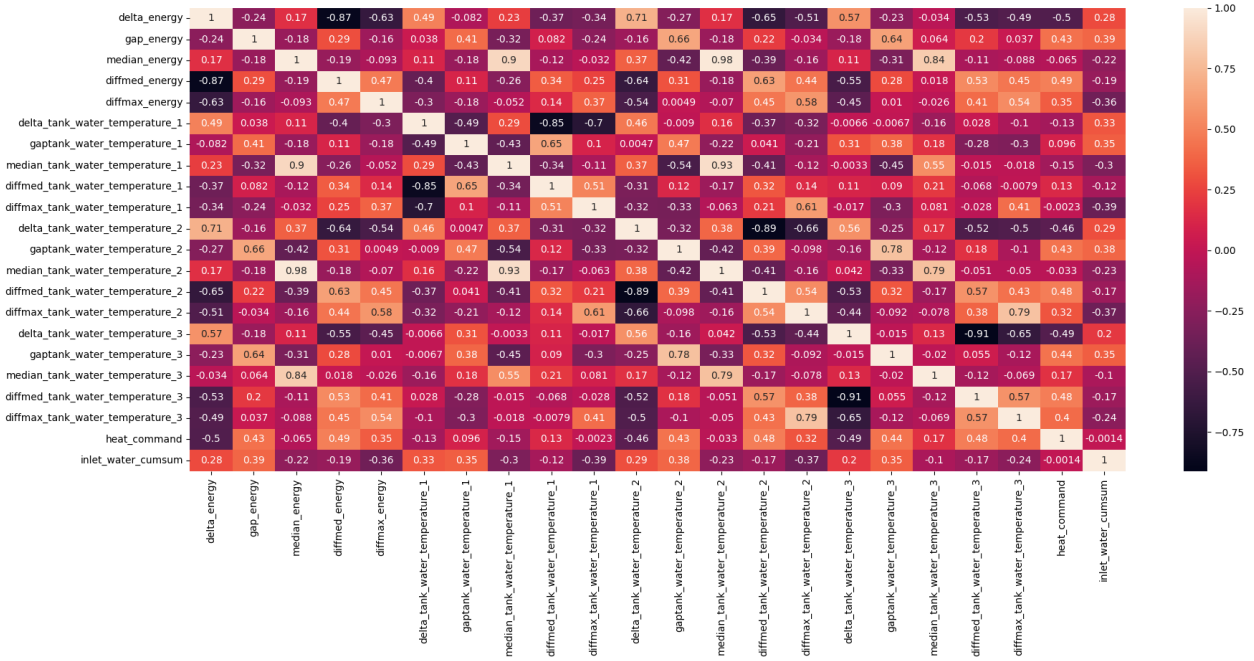

Figure 7: Matrice de corrélation de Spearman

D'après les matrices de corrélation, on peut remarquer que les variables *delta_energy*, *delta_tank_water_temperature_1*, *delta_tank_water_temperature_2*, *delta_tank_water_temperature_3* sont les plus corrélées à notre target. De plus, en analysant la matrice de corrélation avec les variables créées, on constate que les variables delta (fin-start) et gap (max-min) sont les plus fortement corrélées à notre cible.

Nous avons également utilisé une "clustermap" pour visualiser ces relations. La "clustermap" utilise une technique de clustering hiérarchique pour regrouper les variables qui sont fortement corrélées entre elles.

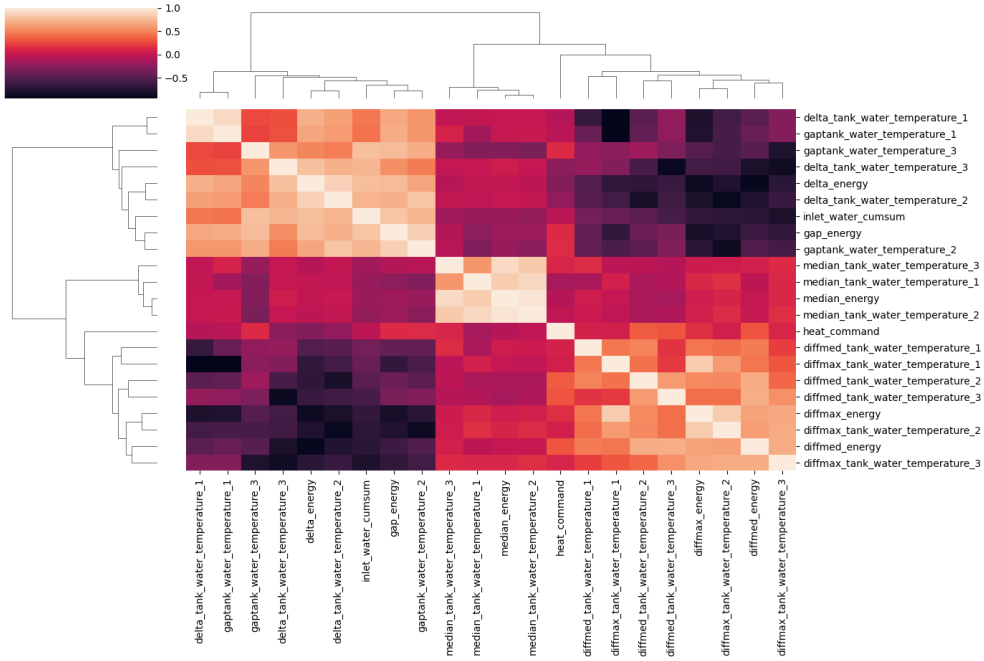

Figure 8: Clustermap de la matrice de corrélation de Pearson

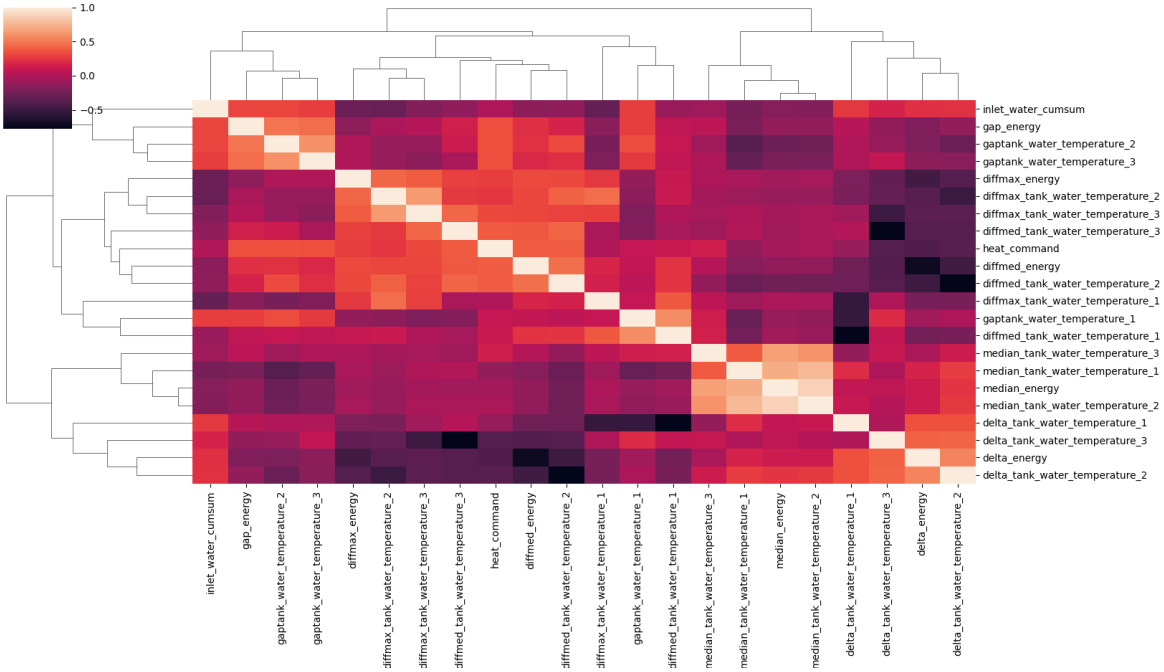

Figure 9: Clustermap de la matrice de corrélation de Kendall

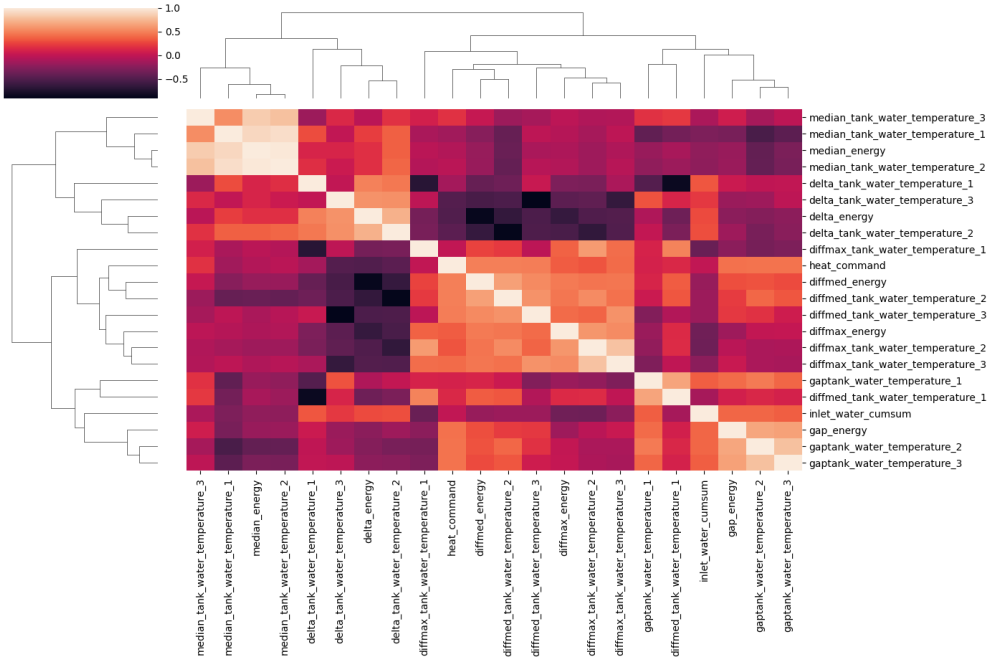

Figure 10: Clustermap de la matrice de corrélation de Spearman

2.1.8 Visualisation des relations features/target

Afin d'explorer les relations entre nos features et la target, nous avons tracé la distribution de chaque feature pour les différents types de sous-tirage.

Les graphiques suivants montrent la distribution des variables 'delta' et 'gap' pour les différents types de sous-tirage. On peut constater que les distributions des sous-tirages faibles, moyens, élevés et énormes sont significativement différentes de la distribution des non-sous-tirages.

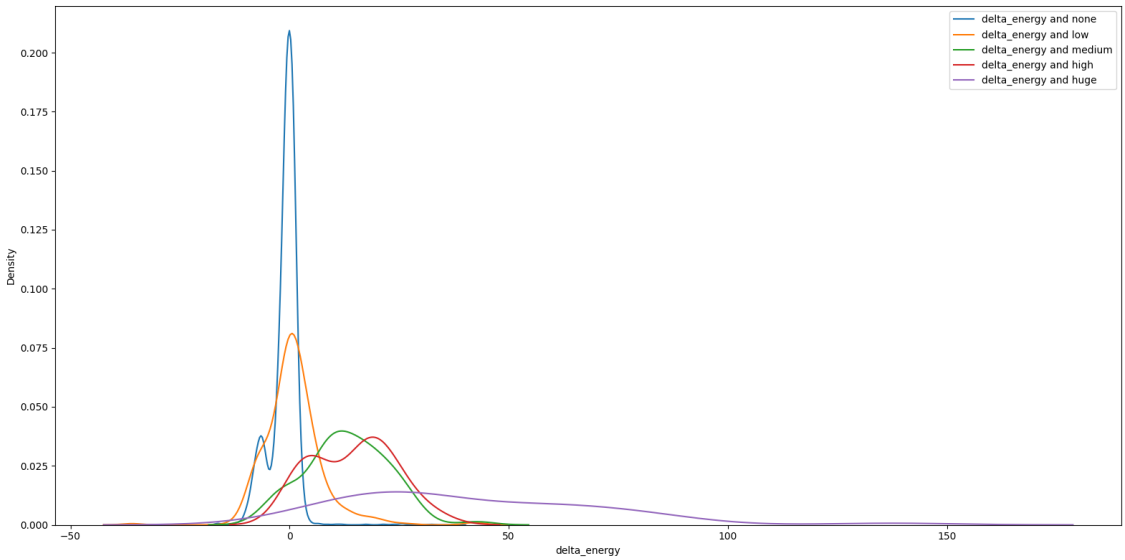

Figure 11: Distribution de ‘delta_energy‘ pour chaque type de sous-tirage

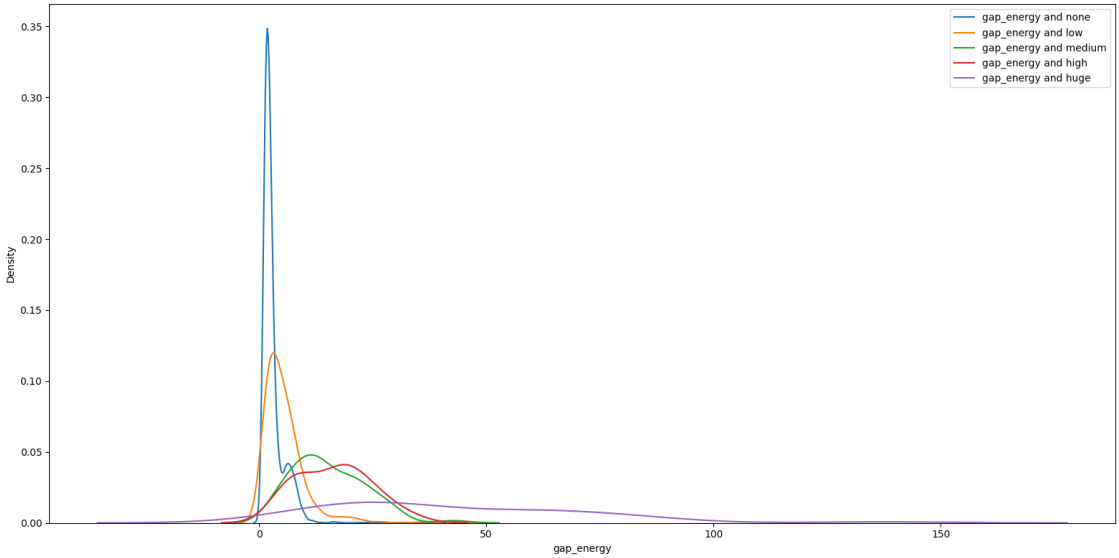

Figure 12: Distribution de ‘gap_energy‘ pour chaque type de sous-tirage

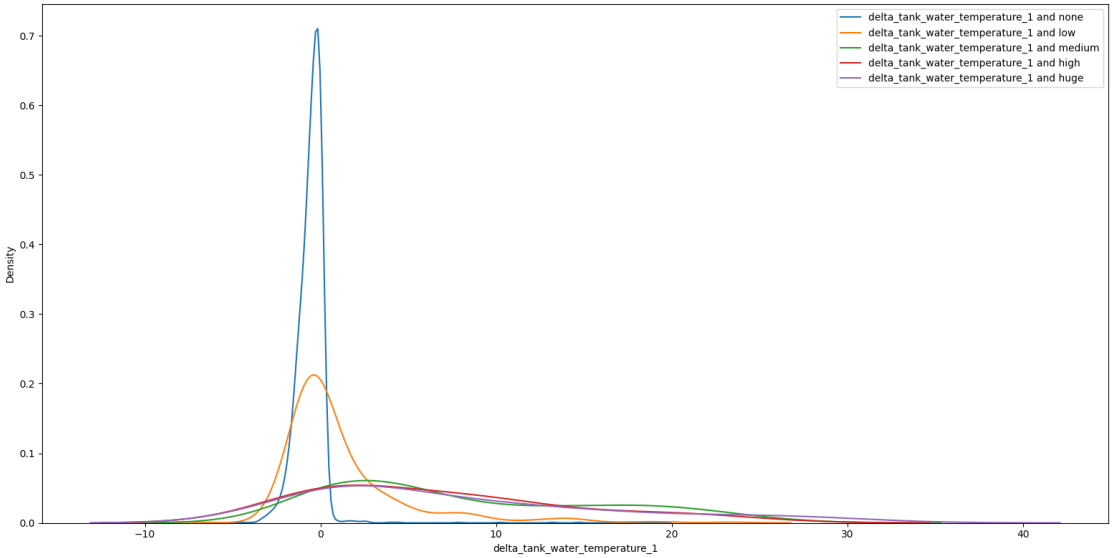

Figure 13: Distribution de ‘delta_tank_water_temperature_1‘ pour chaque type de sous-tirage

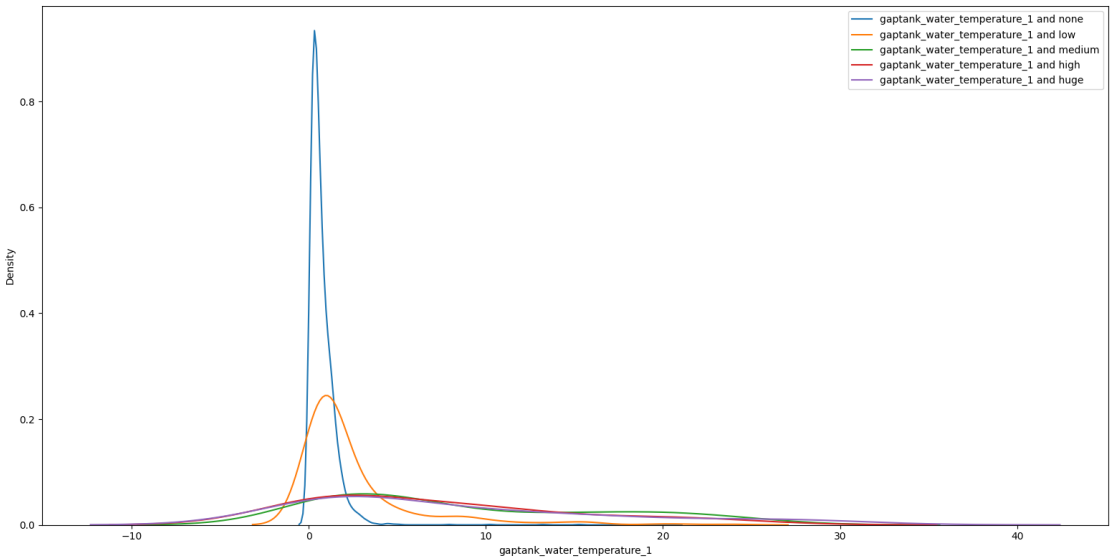

Figure 14: Distribution de ‘gaptank_water_temperature_1‘ pour chaque type de sous-tirage

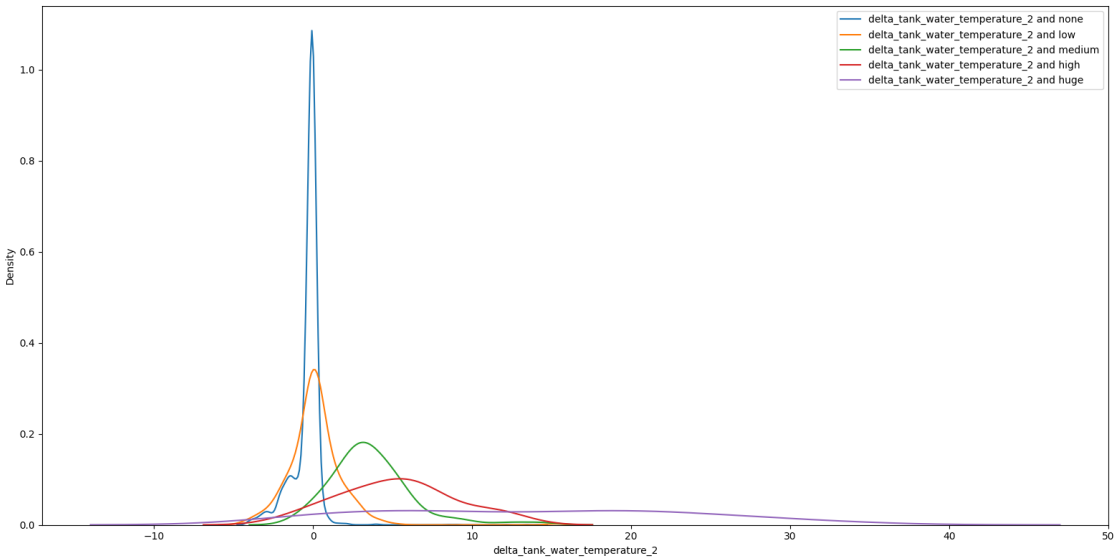

Figure 15: Distribution de ‘delta_tank_water_temperature_2‘ pour chaque type de sous-tirage

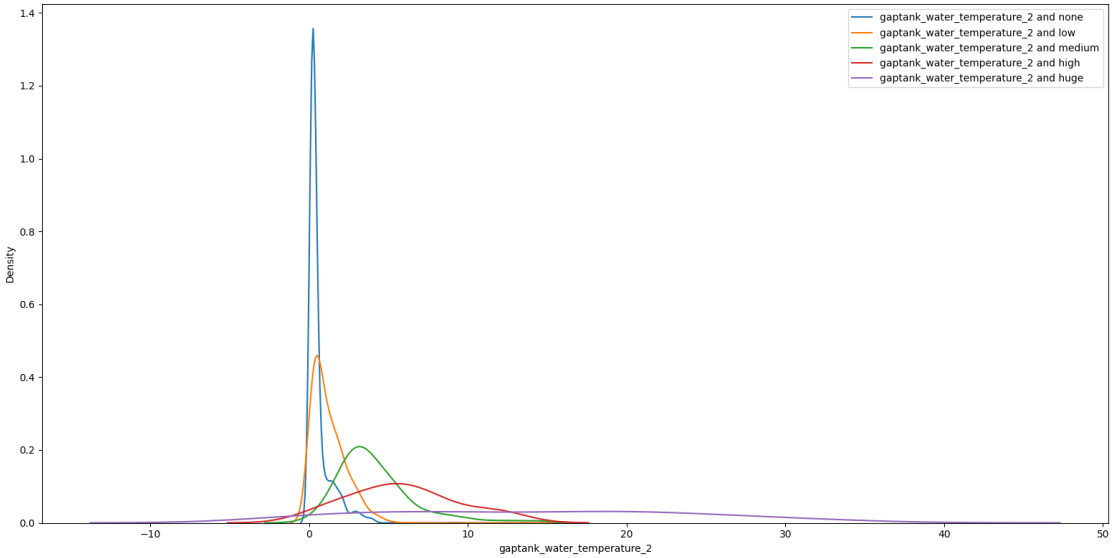

Figure 16: Distribution de ‘gaptank_water_temperature_2‘ pour chaque type de sous-tirage

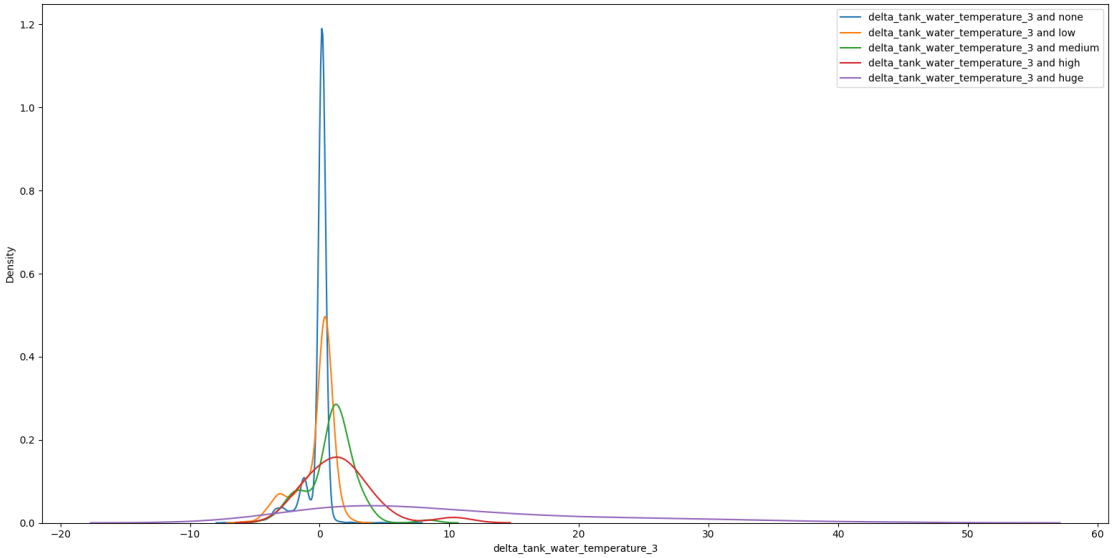

Figure 17: Distribution de ‘delta_tank_water_temperature_3‘ pour chaque type de sous-tirage

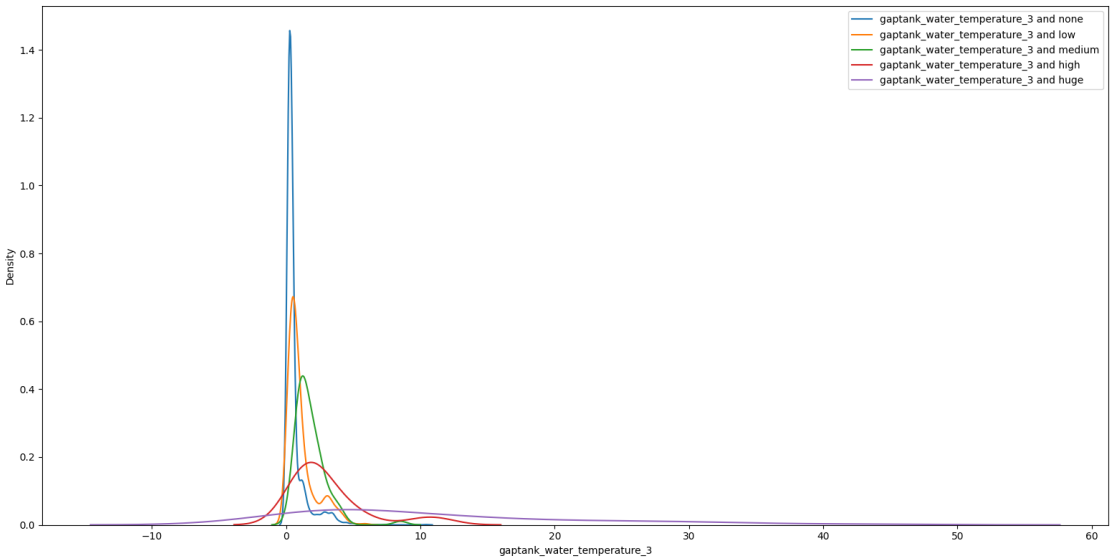

Figure 18: Distribution de ‘gaptank_water_temperature_3‘ pour chaque type de sous-tirage

Comme on peut le voir sur la figure 11, même si les densités des catégories none et low sont différentes, leurs maximums se confondent. Ce qui indique que le modèle pourrait avoir du mal à bien distinguer ces deux catégories-là. On pourrait se retrouver avec un modèle qui interprète un sous-tirage faible alors qu'il n'y en a pas ou inversement.

À l'inverse, pour la figure 12, les maximums ne se confondent pas pour ces catégories.

Ces observations nous conduisent à formuler l'hypothèse suivante : la différence de température et d'énergie est différente lorsqu'il y a un sous-tirage que lorsqu'il n'y en a pas. Pour tester cette hypothèse, nous formulons l'hypothèse nulle suivante (H0) : ‘delta_tank_water_temperature_X‘ ou ‘gap_tank_water_temperature_X‘ sont identiques qu'il y ait un sous-tirage ou non.

Pour évaluer cette hypothèse, nous avons procédé à un test de Student. Le test de Student, ou test t, est une technique statistique utilisée pour déterminer si la différence entre les moyennes de deux groupes est significative. Le test t est calculé comme suit :

$$t = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

où $\bar{X}_1$ et $\bar{X}_2$ sont les moyennes des deux échantillons, s_1^2 et s_2^2 sont leurs variances, et n_1 et n_2 sont leurs tailles respectives.

Avant de procéder au test, nous avons veillé à sélectionner deux échantillons de taille équivalente pour éviter tout biais dû à la différence de taille des groupes de sous-tirage. À la suite du test, nous avons constaté que l'hypothèse nulle est rejetée dans tous les cas, renforçant ainsi notre hypothèse initiale que la différence de température et d'énergie est liée au sous-tirage.

2.2 Modèles et métriques utilisés

2.2.1 Modèles employés

Au cours de ce stage, plusieurs algorithmes ont été explorés afin de mettre au point un modèle prédictif efficace et précis pour estimer le volume d'eau consommé. Ces modèles ont été choisis pour leur capacité à traiter des données linéaires ou non linéaires et leur facilité d'optimisation:

- Régression linéaire : un modèle basique permettant de comprendre les relations linéaires entre les variables.
- Random Forest : un ensemble d'arbres de décision qui a la capacité de gérer efficacement les relations non linéaires.
- AdaBoost : un algorithme de boosting adaptatif qui ajuste les poids des observations mal classées.
- GradientBoosting : un algorithme de boosting qui optimise une fonction de perte différentiable.

2.2.2 Métriques d'évaluation

L'évaluation du modèle est cruciale pour comprendre ses performances. Les métriques suivantes ont été choisies pour fournir une évaluation complète des prédictions du modèle, en mettant l'accent sur différentes facettes des erreurs:

- R^2 score : coefficient de détermination.
- (Mean Squared Error) : moyenne des carrés des erreurs.
- MAE (Mean Absolute Error) : moyenne des erreurs absolues.
- Max Erreur : l'erreur maximale observée.
- Erreur relative : calculée comme $\frac{valeur\ actuelle - valeur\ prédite}{valeur\ actuelle}$.
- Erreur cumulée entre le volume total prédit et le volume total réel.

Ces métriques ont été évaluées pour différentes catégories de sous-tirages : global, usage, none, low, medium, high, et huge.

Remarquons que l'on a ajouté deux catégories: global et usage. "Global" correspond à toutes les catégories de sous-tirage et "usage" correspond à toutes les catégories sauf "none". La catégorie "usage" est très importante car elle nous aidera à comprendre l'erreur que fait le modèle sans compter la catégorie "none". En effet, on verra qu'il est très facile pour un modèle de renvoyer 0, puisqu'il y a beaucoup de 0 dans notre variable target. Par conséquent l'erreur globale sera proche de 0 mais ne veut pas dire que le modèle est performant car l'erreur sur la catégorie "usage" peut être très élevée.

Nous ferons attention à la catégorie low car elle peut être biaisé. Par exemple si on suppose que tous les faibles sous tirages sont à 1 litre, et que notre modèle prédit 3 litres on aurait 300% d'erreur. Alors que pour la catégorie "huge" faire une erreur de 300 % sur un sous-tirage de 40 litres, revient à renvoyer 160 litres. Et d'après le diagramme en camembert 4, cette erreur aura un plus gros impact sur l'erreur cumulée qu'une erreur sur la catégorie "low".

2.2.3 Défis rencontrés

En travaillant sur l'optimisation des modèles et des features, un défi majeur a été identifié. En changeant les tranches horaires, la distribution des catégories de sous-tirages s'avère inégale. Par exemple, une tranche horaire de 5 minutes pourrait ne comporter que 0,1% de sous-tirages énormes, rendant difficile pour le modèle de prédire avec précision cette catégorie. Ceci suggère la nécessité de redéfinir les catégories de sous-tirages selon la tranche horaire choisie. Une plus grande attention doit donc être accordée à la manière dont ces catégories sont définies et utilisées lors de l'entraînement et de l'évaluation des modèles.

Nous avons également fait attention à ce que le modèle soit capable de faire de bonnes prédictions que l'on ait un énorme ou faible sous-tirage. Pour ça nous avons créé une fonction qui permet de répartir équitablement nos catégories de sous-tirages entre les données d'entraînement et de test.

2.3 Résultats

Avec plus de 5000 modèles entraînés, nous avons été obligés d'en écarter un grand nombre pour pas rendre ce rapport trop indigeste.

Alors nous avons sélectionné les 9 modèles les plus intéressants, ainsi que leurs avantages et inconvénients.

Mais avant, expliquons rapidement chaque modèle:

(1): AdaBoostRegressor(estimator=RandomForestRegressor(n_estimators=50),n_estimators=100)

Avec la combinaison de variables suivante:

(delta_3, delta_energy, gap_energy, heat_command)

Ce modèle a deux paramètres modifiés:

- estimator: L'estimateur utilisé par le modèle. Par défaut, il s'agit d'un arbre de décision, ici il a été remplacé par une random forest. Notons, que cette random forest contient 50 estimateurs.

- n_estimators: Le nombre d'estimateurs. Ici le modèle va entrainer 100 RandomForest à la suite. Chaque modèle tentant d'optimiser le résultat de son

prédecesseur.

(2): AdaBoostRegressor(estimator=RandomForestRegressor(n_estimators=200))

Avec la combinaison de variables suivante:

(delta_energy, gap1, gap3, gap_energy, diffmax_energy, heat_command)

gap1 désigne la variable gap_tank_water_temperature_1

(3): AdaBoostRegressor(estimator=RandomForestRegressor(n_estimators=200),learning_rate=0.1, loss='square',n_estimators=100)

Avec la combinaison de variables suivante:

(delta_3, delta_energy, gap1, gap3, gap_energy, diffmax_energy, heat_command)

Ce modèle a deux autres paramètres modifiés:

- learning_rate: Le poids qu'on applique à chaque itération. Par défaut égal à 1

- loss: À chaque itération, le modèle actuel tente d'optimiser la perte du précédent modèle. Le paramètre loss correspond à la manière dont la loss est calculée. Par défaut on calcule juste la différence entre les valeurs prédites et les vraies valeurs. Ici on calcule différence au carré.

(4): AdaBoostRegressor(estimator=RandomForestRegressor(n_estimators=200),loss='square')

Avec la combinaison de variables suivante:

(delta_3, delta_energy, gap1, gap3, gap_energy, diffmax_energy, heat_command)

(5): GradientBoostingRegressor()

Avec la combinaison de variables suivante:

(gap_energy, diffmax_energy)

(6): GradientBoostingRegressor()

Avec la combinaison de variables suivante:

(gap_energy)

(7): GradientBoostingRegressor(n_estimators=50)

Avec la combinaison de variables suivante:

(gaptank_water_temperature_3, gap_energy, diffmax_energy, heat_command)

Ici on a modifié un paramètre:

- n_estimator: Le nombre d'estimateur, qui est équivalent au paramètre d'AdaBoost.

(8): Pipeline(PolynomialFeatures()),(LinearRegression())])

Avec la combinaison de variables suivante:

(delta_3, diffmax_2, diffmax_3)

Pipeline correspond à une combinaison de deux transformations. D'abord les entrées sont traitées par la fonction PolynomialFeatures. Cette fonction renvoie une matrice des combinaisons polynomiales de degré 2 de nos entrées. Par exemple, si l'entrée est $[a, b]$, la fonction renvoie $[1, a, b, a^2, ab, b^2]$. Enfin, la régression linéaire est entrainé à partir des données fournies par notre première fonction.

(9): Pipeline(PolynomialFeatures()),(LinearRegression())])

Avec la combinaison de variables suivante:

(gap1, gap2, gap3)

Par soucis de simplicité, les modèles seront abrégés par leur numéro respectifs. Pour retrouver facilement le nom des modèles, les numéros des modèles dans le tableau renvoient vers le nom du modèle associé.

Enfin, il y a certaines abréviations dans le tableau que nous allons expliciter:

- R2 correspond au score R2
- RE 75% correspond à l'erreur relative à 75%
- RE 95% correspond à l'erreur relative à 95%
- RE max correspond à l'erreur relative maximale
- sumerr correspond à la somme des erreurs en pourcentage entre les vraies valeurs et les valeurs prédites.

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
$R2_{global}$	0.9361	0.9040	0.9358	0.9282	0.8146	0.7886	0.9074	0.9002	0.8512
$R2_{usage}$	0.9516	0.9291	0.9480	0.9485	0.7967	0.7731	0.9224	0.9469	0.8591
$R2_{none}$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
$R2_{low}$	-0.2944	-0.6786	-1.5118	-0.7872	-0.7170	-0.5330	-0.6142	-0.0887	-0.9245
$R2_{medium}$	-2.5646	-2.3094	-2.0446	-1.9046	-5.4415	-8.6283	-1.5552	-5.1467	-3.1771
$R2_{high}$	-21.7576	-10.2970	-23.7606	-15.3054	-28.0033	-52.3598	-18.1088	-36.3448	-41.5270
$R2_{huge}$	0.8974	0.8134	0.9172	0.8906	0.3576	0.3041	0.7888	0.9014	0.5933
MSE_{global}	2.3914	3.5951	2.4034	2.6880	6.9497	7.9240	3.4696	3.7343	5.5767
MSE_{usage}	10.2064	14.9527	10.9607	10.8500	42.8602	47.8542	16.3905	11.2068	29.7062
MSE_{none}	0.9979	1.5700	0.8776	1.2327	0.5371	0.7936	1.1624	2.4020	1.2679
MSE_{low}	2.8962	3.7557	5.6200	3.9987	3.8416	3.4301	3.6118	2.4359	4.3059
MSE_{medium}	7.3262	6.8015	6.2575	5.9696	13.2387	19.7884	5.2515	12.6330	8.5849
MSE_{high}	27.0497	13.4277	29.4305	19.3806	34.4733	63.4235	22.7127	44.3880	50.5475
MSE_{huge}	70.1250	127.4968	56.5921	74.7791	438.8940	475.4426	144.2902	67.3360	277.8585
MAE_{global}	0.4557	0.4883	0.3875	0.4167	0.5644	0.6974	0.4452	0.8660	0.8362
MAE_{usage}	1.7661	1.9119	1.8642	1.7218	2.9782	3.2256	2.0189	2.0371	2.7232
MAE_{none}	0.2220	0.2344	0.1242	0.1840	0.1333	0.2459	0.1641	0.6572	0.4992
MAE_{low}	0.9642	0.9216	1.2028	0.9133	2.9782	3.2256	1.0087	1.0550	1.0887
MAE_{medium}	2.1149	1.8920	1.9687	1.7287	2.8741	3.5500	1.6754	3.0941	2.4680
MAE_{high}	4.5332	3.1989	4.7674	3.8983	4.6421	6.5037	4.2672	5.8885	6.8130
MAE_{huge}	6.6157	9.8231	5.9013	7.5735	16.7608	18.0220	10.2355	6.5489	15.2195
$RE75\%\_global$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
$RE75\%\_usage$	0.8277	0.7564	0.9922	0.9382	0.9486	0.9392	0.9613	1.2806	1.1008
$RE75\%\_none$	-0.075	-0.0602	0.0	-0.0020	-0.0315	-0.0390	-0.0151	-0.2062	-0.1224
$RE75\%\_low$	0.9297	0.7564	0.9922	0.9382	0.9486	0.9392	0.9832	1.2806	2.2511
$RE75\%\_medium$	0.4147	0.4308	0.3880	0.3780	0.5112	0.8802	0.3209	0.5542	0.5052
$RE75\%\_high$	0.4555	0.3702	0.4638	0.3936	0.6517	0.9492	0.4810	0.6324	0.6127
$RE75\%\_huge$	0.2538	0.3295	0.2550	0.2595	0.5981	0.6848	0.3171	0.2378	0.5233
$sumerr_{global}$	-0.1118	-0.1461	-0.0569	-0.0940	0.0651	0.0824	0.0	-0.0979	-0.0330
$sumerr_{usage}$	0.0781	0.0543	0.0493	0.0632	0.1786	0.2923	0.1387	0.1093	0.1812
$sumerr_{none}$	-149.4370	-157.7545	-83.5997	-123.8328	-89.3610	-165.2509	-109.1503	-163.0587	-168.6677
$sumerr_{low}$	0.0433	-0.0237	-0.1317	0.0208	-0.1745	0.3089	-0.0206	0.0996	0.0512
$sumerr_{medium}$	0.0265	0.0720	-0.0861	0.0359	0.1146	0.2637	0.1772	0.2364	0.2371
$sumerr_{high}$	0.3898	0.2713	0.4099	0.3352	0.3991	0.5592	0.3467	0.1225	0.5858
$sumerr_{huge}$	0.0638	0.0435	0.0940	0.0490	0.2674	0.2627	0.1453	0.0728	0.1500
RE 95%_global	0.7147	0.6657	0.9745	0.8063	0.9247	0.9070	0.8989	0.8935	0.8549
RE 95%_usage	3.4330	3.7321	2.9267	3.9772	3.4284	5.2712	3,4108	10.7665	9.9465
RE 95%_none	-0.0366	-0.0270	0.0	0.0	-0.0315	-0.0390	-0.0063	-0.0417	-0.0312
RE 95%_low	5.9400	8.4700	6.3573	6.0280	5.2338	9.1604	4.3712	11.6277	13.4968
RE 95%_medium	0.6278	0.6673	0.5744	0.6571	1.0209	0.9411	0,6997	0.6896	0.7990
RE 95%_high	0.6496	0.3803	0.7384	0.5227	0.7596	0.9527	0.5582	0.7805	0.8151
RE 95%_huge	0.5629	0.5377	0.6798	0.4043	0.8655	0.9398	0.6303	0.4451	0.8212
RE max_global	43.9440	59.1300	62.4280	62.0730	45.6643	18.3491	52.3073	64.7378	61.6809
RE max_usage	43.9440	59.1300	62.4280	62.0730	45.6643	18.3491	52.3073	64.7378	61.6809
RE max_none	0.0	-0.0117	0.0	0.0	-0.0315	-0.0390	-0.0012	-0.0016	-0.0012
RE max_low	43.9440	59.1300	62.4280	62.0730	45.6643	18.3491	52.3073	64.7378	61.6809
RE max_medium	0.7281	0.7093	1.0610	0.7129	1.5295	0.9601	0.7637	0.7519	0.9393
RE max_high	0.6982	0.3828	0.8071	0.5550	0.7865	0.9535	0.5775	0.8176	0.8658
RE max_huge	0.6128	0.5439	0.7749	0.5019	0.9393	0.9625	0.6870	0.5843	0.9346

Table 1: Tableau comparatif des métriques des différents modèles

Maintenons que l'on a les résultats de nos modèles, nous pouvons analyser les avantages et les incovénients de chacun.

Modèle 1:
Avantages :

- Le coefficient de détermination (R2) est élevé pour les catégories Global, Usage et Huge, indiquant une bonne performance du modèle sur ces segments.
- La MAE est relativement faible pour la plupart des catégories, ce qui suggère que le modèle a tendance à faire des prédictions proches des valeurs réelles.
- Le troisième quartile de l'erreur relative est très faible pour les catégories Medium, High et Huge. Ce qui est très bien car ça veut dire que le modèle est capable de s'adapter à ces catégories.

Inconvénients :

- Le coefficient de détermination (R2) est négatif pour les catégories Low, Medium et High. Cela suggère que le modèle est moins performant pour ces segments.
- La MSE est relativement élevée pour les catégories Usage, Huge et High, indiquant une plus grande variabilité dans les erreurs de prédiction pour ces segments.
- L'erreur maximale (ME) est également élevée pour certaines catégories, ce qui signifie que le modèle peut parfois faire des prédictions très éloignées de la valeur réelle.

Modèle 2:
Avantages:

- Erreur Relative à 75%: L'erreur relative à 75% est de 0.76, c'est l'erreur la plus basse de tous les modèles.
- Erreur relative à peu près identique pour les catégories medium, high et huge: L'erreur relative à 75% pour les catégories medium, high, huge est de respectivement 0.43, 0.37, 0.33. Ce qui montre que le modèle peut s'adapter au type de sous-tirage

Inconvénients:

- Somme des erreurs (sumerr): La différence entre la somme des valeurs prédites et la somme des valeurs réelles est de 14%. C'est très élevé. On voudrait idéalement avoir une erreur en dessous de 10%.

Modèle 3:
Avantages:

- R2 score: Le R2 score est de 0.94, ce qui est excellent par rapport aux autres modèles
- Erreur Moyenne Quadratique (MSE): La MSE est de 2.4, c'est un score très bas comparé à d'autres modèles.
- Somme des erreurs (sumerr): La différence entre la somme des valeurs prédites et la somme des valeurs réelles est de 5%.

Inconvénients:

- Erreur relative à 75%: L'erreur relative à 75% est de 0.99. Ce qui signifie que 25% des prédictions font une erreur d'au moins 99%.

Modèle 4:
Avantages:

- R2 Score: La valeur R2 (0.928 pour les données globales) est assez élevée, ce qui indique que le modèle explique une grande partie de la variabilité des données. De plus, pour certaines catégories comme "usage", il atteint 0.949.
- Erreurs relatives: Les erreurs relatives à 25%, 50% et 75% sont proches de zéro pour la plupart des catégories, ce qui signifie que les prédictions du modèle sont généralement proches des valeurs réelles pour au moins 75% des points de données.
- Erreur absolue moyenne (MAE): La MAE pour les données globales est de 0.42, ce qui est relativement faible, indiquant que les prédictions du modèle sont en moyenne proches des valeurs réelles.

Inconvénients:

- Erreur maximale (ME): L'erreur maximale pour les données globales est de 62.1, ce qui signifie que dans le pire des cas, le modèle peut se tromper de 6200%.

Modèle 5:
Avantages:

- Somme des erreurs (sumerr): Au total, le modèle fait une erreur de prediction de 6% entre les vraies données et les données prédites.
- Erreurs relatives: Les erreurs relatives à 25%, 50%, et 75% sont proches de zéro pour la plupart des catégories, ce qui signifie que les prédictions du modèle sont généralement proches des valeurs réelles pour au moins 75% des points de données.

Inconvénients:

- R2 score pour les données "none": On remarque que quel que soit le modèle le R2 score pour les données "none" est 0. C'est normal car les données sont toutes identiques et donc la variance est nulle. De plus, comme le R2 score calcule un rapport entre l'erreur quadratique moyenne et la variance des données, il y a une division par zero. Pour remédier à ce problème, scikit-learn renvoie 1 si les prédictions sont parfaites et 0 sinon.
- R2 Score: Le score R2 pour les données globales est de 0.815, ce qui est assez bas par rapport aux autres modèles.
- Erreurs relatives à 95%: Les erreurs relatives à ces percentiles sont plus élevées que les autres modèles. Cela veut dire que 5% des données font au moins une erreur de 92%

Modèle 6:
Avantages:

- Erreurs relatives pour la catégorie low: L'erreur relative à 75% est de 93.2% ce qui est faible par rapport à d'autres modèles.
- Somme des erreurs (sumerr): La différence entre la somme des valeurs prédites et la somme des valeurs réelles est de 8%.
- Erreur maximale (ME): L'erreur maximale pour les données globales est de 18.3, c'est la plus petite erreur de tous ces modèles.

Inconvénients:

- R2 Score: Le score R2 pour les données globales est de 0.789, ce qui est assez bas par rapport aux autres modèles.
- Erreur absolue moyenne (MAE) et Erreur Quadratique Moyenne (MSE): La MAE pour les données globales est de 0.697, ce qui est élevé par rapport aux autres modèles. De même pour la MSE, qui est de 7.92

Modèle 7:
Avantages:

- R2 score: Le R2 score est de 0.90
- Somme des erreurs: La différence entre la somme des valeurs prédites et la somme des valeurs réelles est de 0%. En réalité, il y a une différence petite, mais l'erreur a été arrondie à 0
- Erreur relative à peu près identique pour les catégories medium, high, huge: L'erreur relative à 75% pour les catégories medium, high, huge est de respectivement 0.32, 0.48, 0.31. Ce qui montre que le modèle peut s'adapter au type de sous-tirage

Modèle 8:
Avantages:

- R2 score: Le R2 score est de 0.9, ce qui est très bon par rapport aux autres modèles
- Erreur Quadratique Moyenne (MSE): La MSE est de 3.7, ce qui est très faible comparé à d'autres modèles

Inconvénients:

- Erreur Absolue Moyenne (MAE): La MAE est de 0.86, c'est le modèle avec la plus grosse MAE des 9 modèles.
- Erreur relative à 75 et 95%: L'erreur relative à 75 et 95% pour la catégorie "usage" est de respectivement 128% et 1077%. C'est le modèle avec les plus grosses erreurs relatives
- Erreur relative qui varie selon les modèles: L'erreur relative selon les catégories medium, high et huge varient. Ce qui signifie que le modèle a du mal à s'adapter selon les catégories

Modèle 9:
Avantages:

- Somme des erreurs (sumerr): La différence entre la somme des valeurs prédites et la somme des valeurs réelles est de 3%. C'est la meilleure performance de 9 modèles.
- Erreur relative à peu près identique pour les catégories medium, high, huge: L'erreur relative à 75% pour les catégories medium, high, huge est de respectivement 0.5, 0.6, 0.5. Ce qui montre que le modèle peut s'adapter au type de sous-tirage

Inconvénients:

- Erreur relative pour tous les sous-tirages: Le troisième quartille de l'erreur relative est de 110%. Cela signifie que 25% des prédictions font au moins une erreur de 110%
- Erreur relative à 95%: L'erreur relative à 95% est de 995%. Cela signifie que 5% des prédictions font au moins une erreur de 995%
- Erreur Absolue Moyenne (MAE): La MAE est de 0.84, ce qui est très élevé comparé aux autres modèles

À partir de ces résultats, nous pouvons dire que le modèle (7) est le meilleur des 9 modèles.

Pour analyser en détail les performances de ce modèle nous pouvons afficher la densité des erreurs relatives pour chaque catégorie. Pour faire ça nous allons

utiliser un diagramme en violon, il s'agit d'un graphique statistique permettant de comparer la distribution des probabilités. Dans ce cas, nous avons également tracé une boîte à moustache. La boîte à moustache montre la médiane (le point en blanc), le premier et le deuxième quartiles (le haut et le bas du rectangle), ainsi que le maximum et le minimum (les extrémités de la figure).

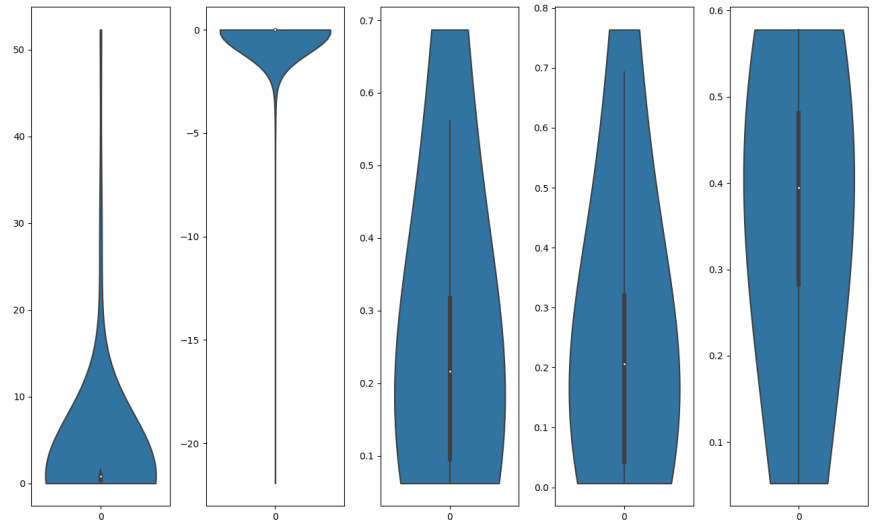

Figure 19: Diagramme en violon des erreurs relatives pour chaque catégorie de sous-tirage du modèle 7. Chaque graphique correspondent respectivement à "none", "low", "huge","medium","high"

Premièrement, nous pouvons remarquer que l'ordre de grandeur pour la grandeur pour la catégorie "none" est plus élevé que pour les autres catégories. C'est normal, car il n'y a pas d'erreur relative pour cette catégorie. En effet comme l'erreur relative se calcule en divisant la vraie valeur et que la vraie valeur dans cette catégorie est tout le temps égale à 0, on a du simplement calculer la différence entre la valeur prédite et la vraie valeur.

Ensuite comme indiqué sur le tableau, il y a de grosses valeurs abhérantes pour la catégorie "low" pouvant aller jusqu'à -5230%. Pour essayer de comprendre ce qui ne va pas, analysons un peu quelques prédictions faites par le modèle.

```
gaptank_water_temperature_3     0.826338
gap_energy                     10.166163
diffmax_energy                  -0.04437
heat_command                         0.0
actual                               3.8
predict                         3.485864
Name: 2023-03-02 11:20:00, dtype: Float64
gaptank_water_temperature_3    0.710451
gap_energy                     3.055346
diffmax_energy                -0.012588
heat_command                     1200.0
actual                              3.2
predict                        0.124686
Name: 2023-03-01 09:40:00, dtype: Float64
```

Figure 20: Prédiction du modèle sur deux entrées. La première prédiction fait une erreur 8%, la deuxième 96%

Sur cette image 20, les quatres premières lignes correspondent aux données d'entrées du modèle. La cinquième ligne correspond à la valeur que devrait renvoyer le modèle et la sixième est la valeur renvoyée par le modèle. La première partie montre une prédicition avec une erreur de 8% et la deuxième montre une autre avec une erreur de 96%. Nous pouvons constater que la variable gap_energy est beaucoup plus elevée dans la première prédiction que la deuxième. Ce qui nous amène à supposer que l'arbre de décision entraîné par GradientBoosting renvoie une valeur très faible quand la variable gap_energy est inférieur à un certain seuil. Ce qui est logique puisque cette variable qu'on a créée est très élevée quand il y a un sous-tirage puisque la température du ballon baisse à ce moment.

On remarque aussi que la variable heat_command est nulle lors de la première prédiction et non-nulle lors de la deuxième. On pourrait donc également supposer que le modèle renvoie une valeur faible quand cette variable est non-nulle. Ce qui est logique physiquement puisque si le ballon chauffe, c'est que la température du ballon est trop basse, et donc il y a peu de chances qu'il y ait un sous-tirage à ce moment.

On est donc amené à supposer que le modèle a été induit en erreur sur cette prediction. Pour être sur, regardons une autre prédiction.

```
gaptank_water_temperature_3    0.669077
gap_energy                     8.138509
diffmax_energy                -0.046161
heat_command                        0.0
actual                             2.65
predict                        2.390605
Name: 2023-03-10 16:20:00, dtype: Float64
gaptank_water_temperature_3    0.714435
gap_energy                     3.297329
diffmax_energy                 -0.00923
heat_command                      558.0
actual                             2.25
predict                        0.006337
Name: 2023-03-02 19:20:00, dtype: Float64
```

Figure 21: Prédiction du modèle sur deux entrées. La première prédiction fait une erreur 11%, la deuxième 99,7%

Sur la première partie 21 , on voit une prédiction avec une erreur de 11%, et sur la deuxième partie, l'erreur est de 99,7%. On remarque encore la même chose, la variable *gap_energy* est moins élevée dans la deuxième prédiction et la variable *heat_command* est non nulle. Tout ça nous donne des pistes d'amélioration que nous détaillerons un peu plus loin. D'abord, analysons deux autres modèles intéressants.

Ces modèles sont le modèle 1 et 2. Analysons rapidement leur diagramme en violon.

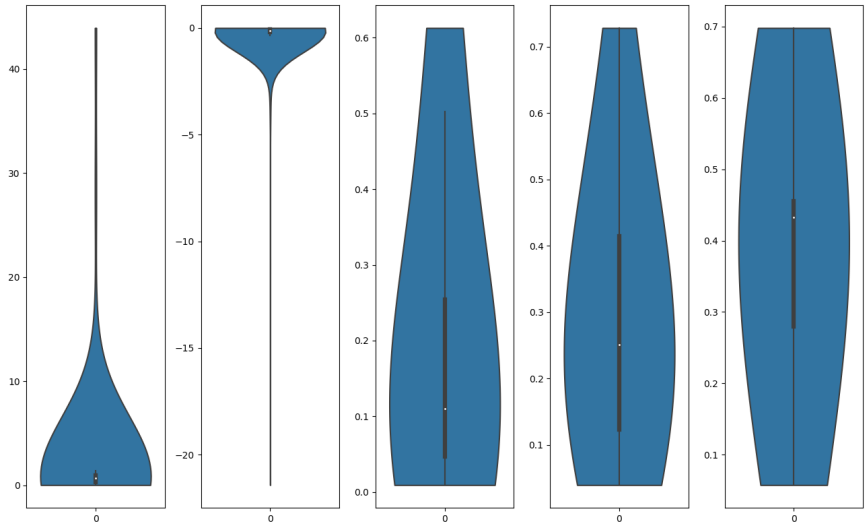

Figure 22: Diagramme en violon des erreurs relatives pour chaque catégorie de sous-tirage du modèle 1. Chaque graphique correspondent respectivement à "none", "low", "huge","medium","high"

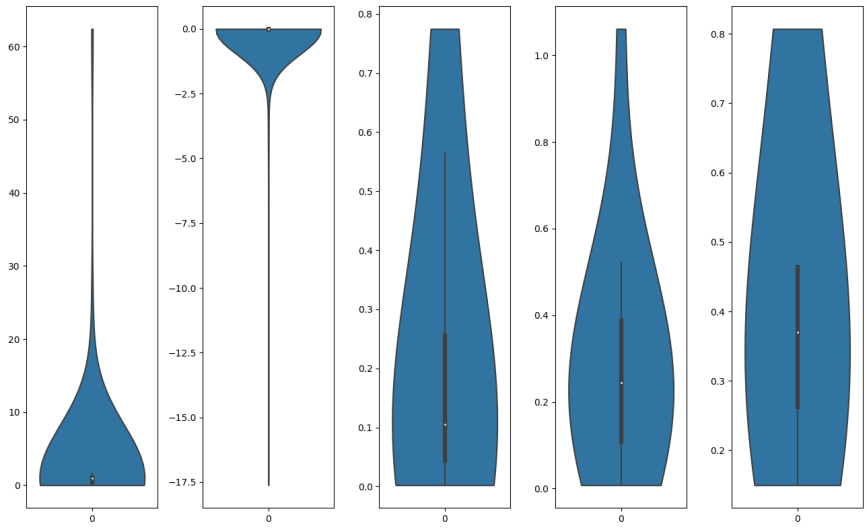

Figure 23: Diagramme en violon des erreurs relatives pour chaque catégorie de sous-tirage du modèle 2. Chaque graphique correspondent respectivement à "none", "low", "huge","medium","high"

Comme pour le modèle 7, il y a de grosses valeurs abhérantes pour la

catégorie "low" pouvant aller jusqu'à -5913%. En ce qui concerne les autres catégories (hors "none"), l'erreur maximale ne dépasse pas 72% ce qui est légèrement mieux que le modèle 7 avec une erreur maximale de 106%

Le modèle 7 reste quand même meilleur du fait de sa prédiction totale presque parfaite et de sa très bonne erreur relative.

3 Conclusion

3.1 Pistes d'amélioration

À partir de tout ce qu'on a vu, nous pouvons donner des idées d'amélioration du modèle:

- Avoir plus de données du client: Nous avons récolté les données d'un client sur 2 mois. Doubler ce nombre de données nous permettrait d'avoir plus de situations spécifiques où le modèle est mauvais.

- Redéfinir nos catégories: Comme mentionné précédemment dans le rapport, nous avons eu quelques problèmes avec la définition arbitraire de nos catégories de sous-tirages. Ainsi définir les catégories en fonction de la durée de la tranche horaire pourrait nous aider à donner de meilleures données pour notre modèle.

- Avoir des données de différents clients: Les données récoltées viennent d'un seul client. Mais rien ne nous dit que le modèle sera performant pour le ballon d'un autre client. En effet, tout le monde n'a pas la même manière de consommer de l'eau chaude. Par exemple, une famille nombreuse ne consommera pas l'eau chaude de la même manière qu'une personne seule. Ainsi récolter des données de différents clients permettrait au modèle d'être capable de s'adapter à n'importe quelle situation.

3.2 Bilan

Pour conclure, nous avons réussi à créer des modèles suffisamment performants pour prédire le volume d'eau consommé par tranche horaire en fonction des températures du ballon.

Cependant, ces modèles sont perfectibles à cause de leur incapacité à généraliser certaines situations spécifiques à cause d'un manque de données.

4 Bibliographie

- [1] Documentation de scikit-learn pour le choix des modèles et des hyper-paramètres
- [2] Documentation de scikit-learn sur les Random forest
- [3] Documentation de scikit-learn sur AdaBoost
- [4] Documentation de scikit-learn sur les GradientBoosting

- [5] Documentation de scikit-learn sur la régression linéaire
- [6] G.Tazin - Private communication