



Soutenance de Projet

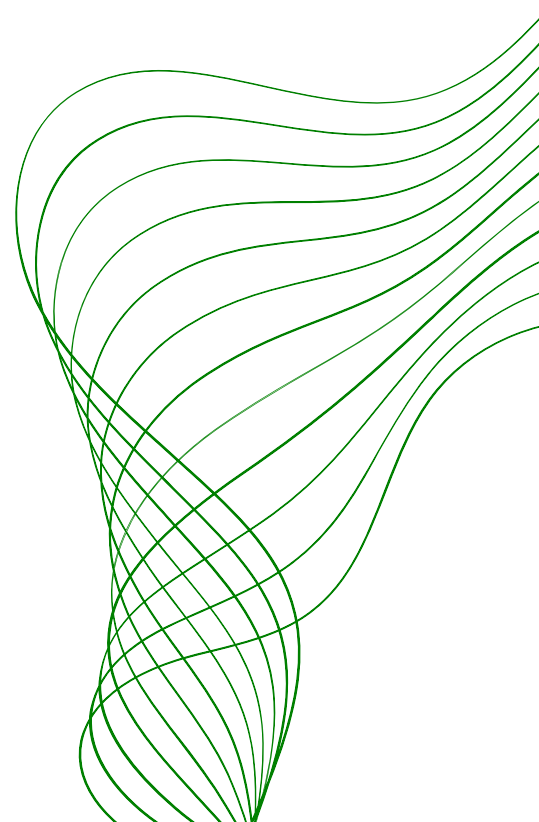
Anticiper les besoins en consommation des bâtiments

Par Chakir Fikri





Plan

1. Problématique
 2. Présentation du jeu de données
 3. Feature Engineering
 4. Modélisation et résultats
- 

1. Problématique

La ville de Seattle souhaite atteindre son objectif de ville neutre en émissions de CO2 avant 2050.

En 2016, les agents de la ville ont effectué des relevés de consommation énergétiques de plusieurs bâtiments afin de connaître leurs quantités de CO2 émises. Cependant, ces relevés sont fastidieux et coûteux à obtenir.

*J'ai été missionné pour élaborer des algorithmes d'apprentissages capables de prédire les quantités d'énergie totale consommée et de CO2 émise par an, **pour les bâtiments non résidentiels pour lesquels les relevés énergétiques n'ont pas (encore) été réalisés.** Les prédictions doivent se fonder sur les caractéristiques structurelles des bâtiments (date de construction, situation géographique, surface et usages).*

2. Présentation du jeu de données

Le jeu de données, après nettoyage, suppressions des bâtiments résidentiels et suppression des caractéristiques sans intérêts pour notre étude, est composé de 1637 propriétés décrites par 26 caractéristiques.

Les caractéristiques peuvent être décrites comme suit :

- 1 caractéristique temporelle : la date de construction (***YearBuilt***).
- 5 caractéristiques de situation géographique :
ZipCode, CouncilDistrictCode, Neighborhood, Latitude et Longitude.
- 5 caractéristiques décrivant le type d'usage de la propriété :
BuildingType, PrimaryPropertyType, LargestPropertyUseType, SecondLargestPropertyUseType et ThirdLargestPropertyUseType.
- 8 caractéristiques de taille ou d'étendue, dont 5 de surface.
- 7 caractéristiques énergétiques (quantité de gaz, électricité, etc).

3. Feature engineering

ENCODAGE DES VARIABLES CATÉGORIELLES

Les variables ont été encodées en appliquant le **LabelEncoder** *sur l'ensemble des variables catégorielles* et non sur chaque variable catégorielle une à une, afin de conserver la correspondance naturelle entre les variables. Cela signifie qu'un unique numéro a été attribué à chaque chaîne de caractère présente dans le data-set. Par conséquent, une chaîne présente dans plusieurs variables a le même numéro dans toutes ces variables.

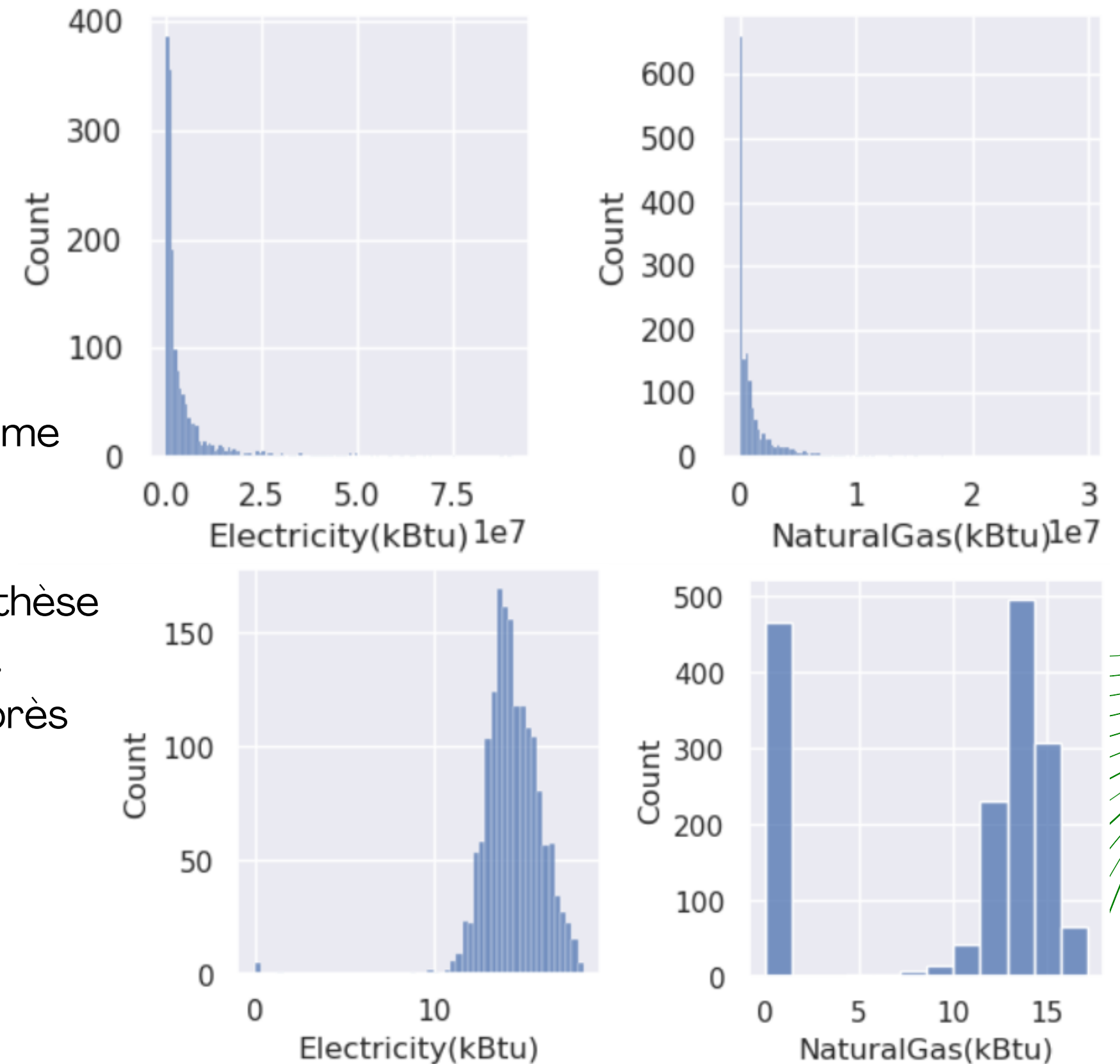
TRANSFORMATION DES VARIABLES :

Nous avons tracé les distributions des variables. 14 variables ont une amplitude et un skewness élevés, c'est à dire que leurs valeurs s'étalent sur des ordres de grandeurs élevés et leur distributions sont concentrées sur la gauche et étalées à droite. Voici comme exemple les distributions de deux de ces variables :

La fonction logarithme a été appliquée à ces variables pour deux raisons :

- 1) Réduire leurs amplitudes ;
- 2) Rendre leurs distributions plus proches de celle d'une gaussienne.

Le premier point est important car certains algorithmes comme la régression linéaire fonctionnent mal lorsque les amplitudes des variables diffèrent beaucoup ;
le second, car beaucoup de modèles sont basés sur une hypothèse de normalité qui est en réalité rarement vérifiée *stricto-sensu*.
Ci-contre, les distributions des deux variables précédentes après passage au *log*.



CRÉATION DE NOUVEAUX PRÉDICTEURS

Démarche générale :

Les variables énergétiques ne peuvent pas être incluses telles quelles parmi les prédicteurs, car le but est justement de s'en affranchir.

Par conséquent et afin d'éviter toute fuite de données (*data leakage*), l'idée générale a été d'extraire des variables énergétiques des informations qu'il sera possible de retrouver à partir des données structurelles des bâtiments, qui elles, seront toujours disponibles.

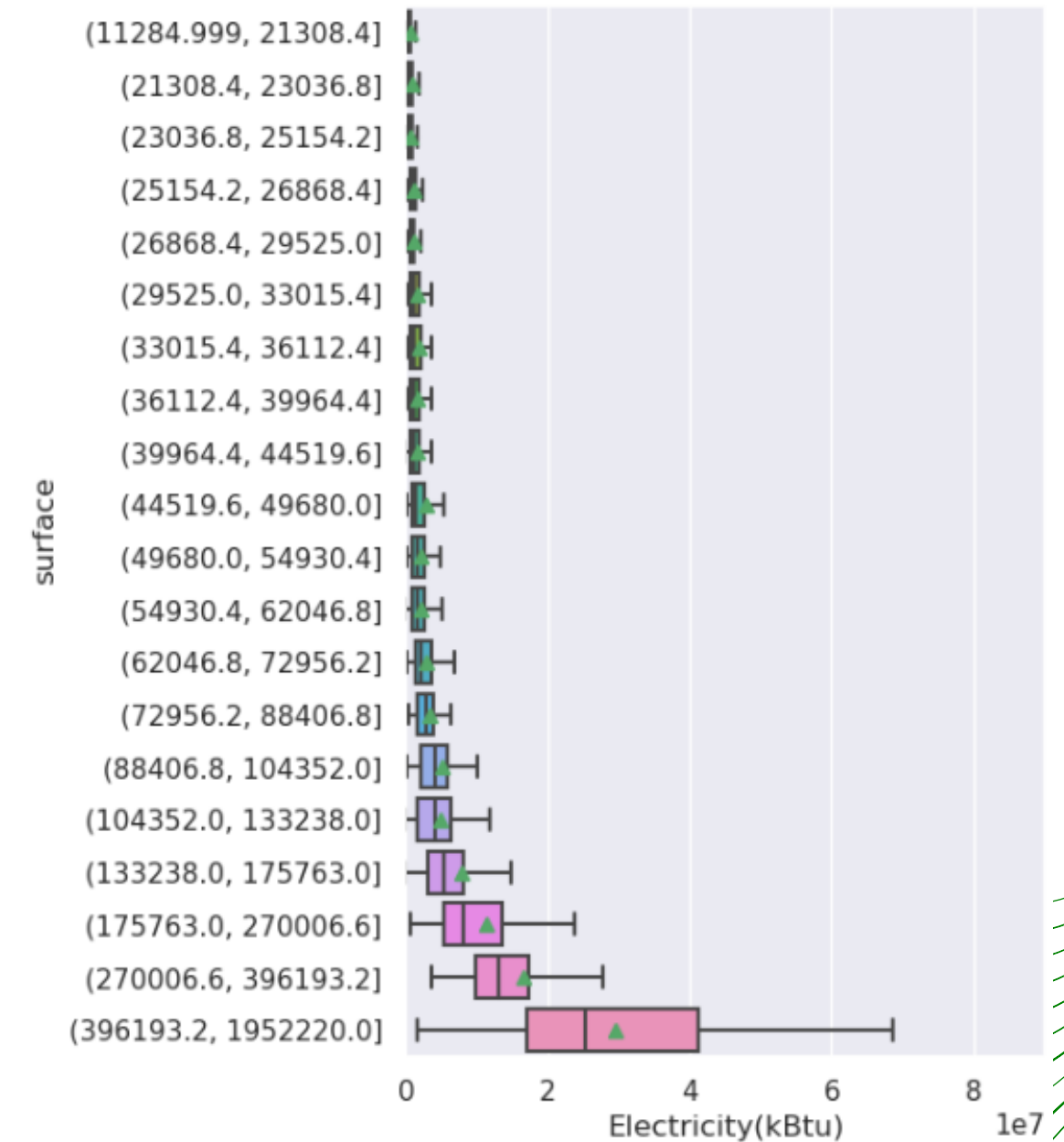
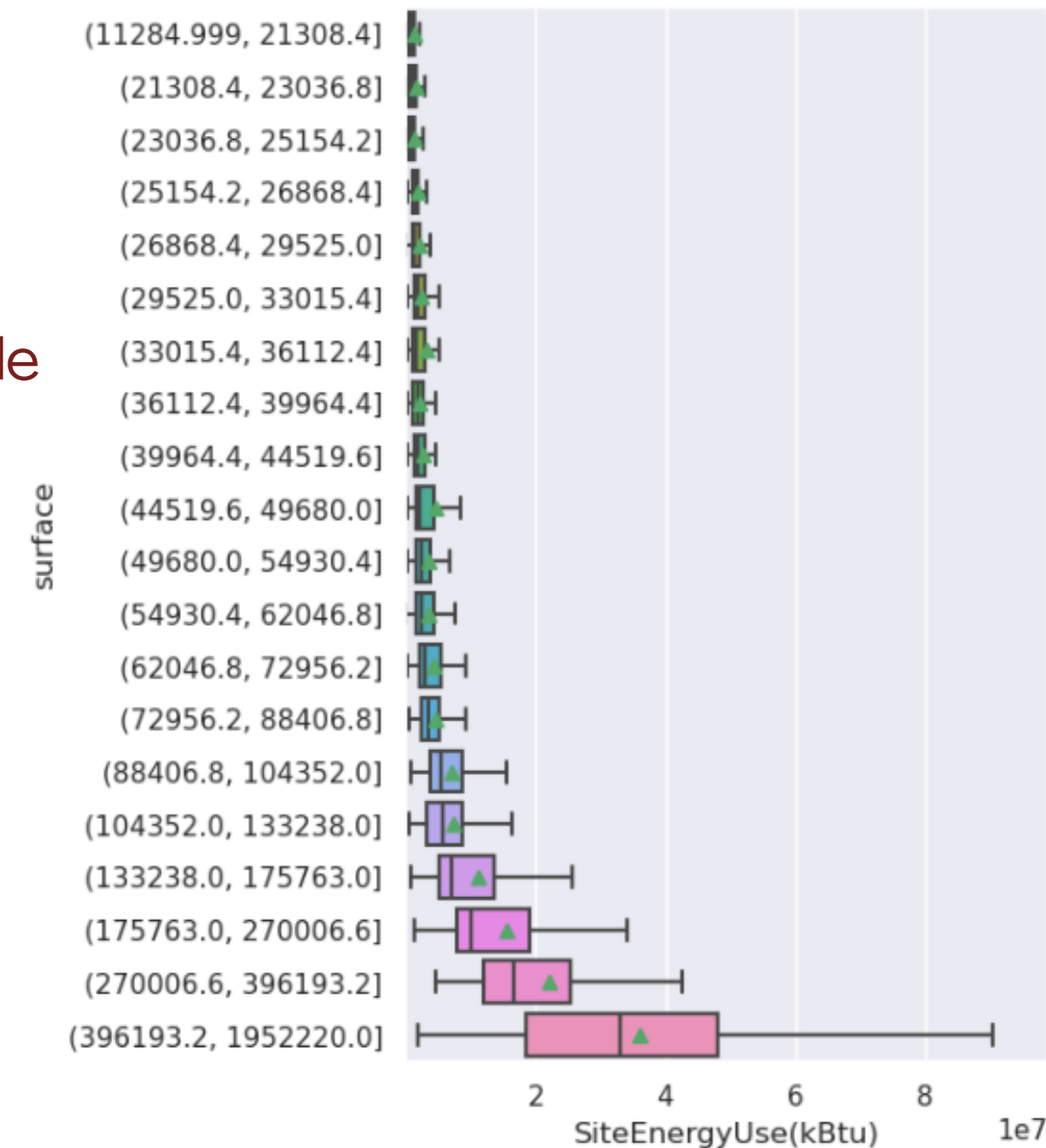
Nous avons donc cherché des correspondances entre certaines variables structurelles et les variables énergétiques.

Surface et Énergie :

Nous avons discrétisé la variable **PropertyGFATotal** en 20 classes de même effectif et affiché, pour chaque variable énergétique, les boxplots de chaque classe. Ci-dessous par exemple, les boxplots pour les caractéristiques d'énergie totale et d'Electricité.

On peut observer que, mis à part pour les 4 ou 5 dernières classes, les boxplots sont écrasés et donc on peut sans commettre d'erreur significative assimiler les valeurs de chaque classe à leur médiane ou leur moyenne.

De même pour les autres variables car leurs boxplots ont une structure similaire.



Nous avons donc créer six nouveaux prédicteurs en procédant comme suit :

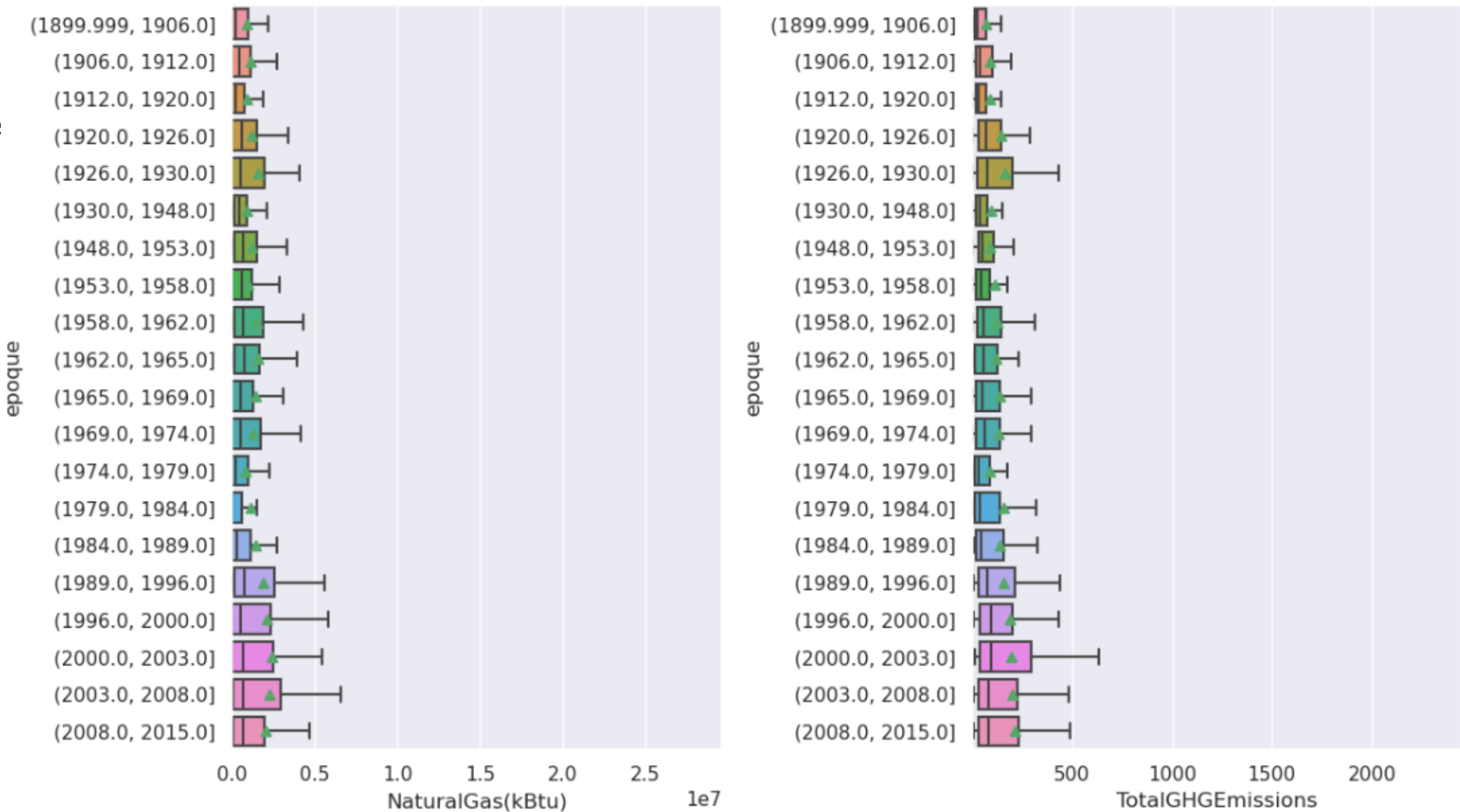
- 1) Discrétisation de la variable **PropertyGFATotal** en un maximum de classes (1560 classes) ;
- 2) Pour chaque propriété d'une classe surfacique donnée, sa valeur pour la variable énergétique considérée est égale à la médiane des valeurs énergétiques de cette classe.
- 3) Passage au log pour les raisons déjà mentionnées.

Date de construction et Énergie :

Nous avons discrétisé la variable **YearBuilt** en 20 classes de même effectif et affiché, pour chaque variable énergétique, les boxplots de chaque classe. Ci-dessous par exemple, les boxplots pour les caractéristiques de gaz naturel et de CO2 émis.

On peut observer que les boxplots sont assez écrasés devant l'amplitude des variable et donc on peut sans commettre d'erreur significative assimiler les valeurs de chaque classe à leur médiane ou leur moyenne.

De même pour les autres variables car leurs boxplots ont une structure similaire.



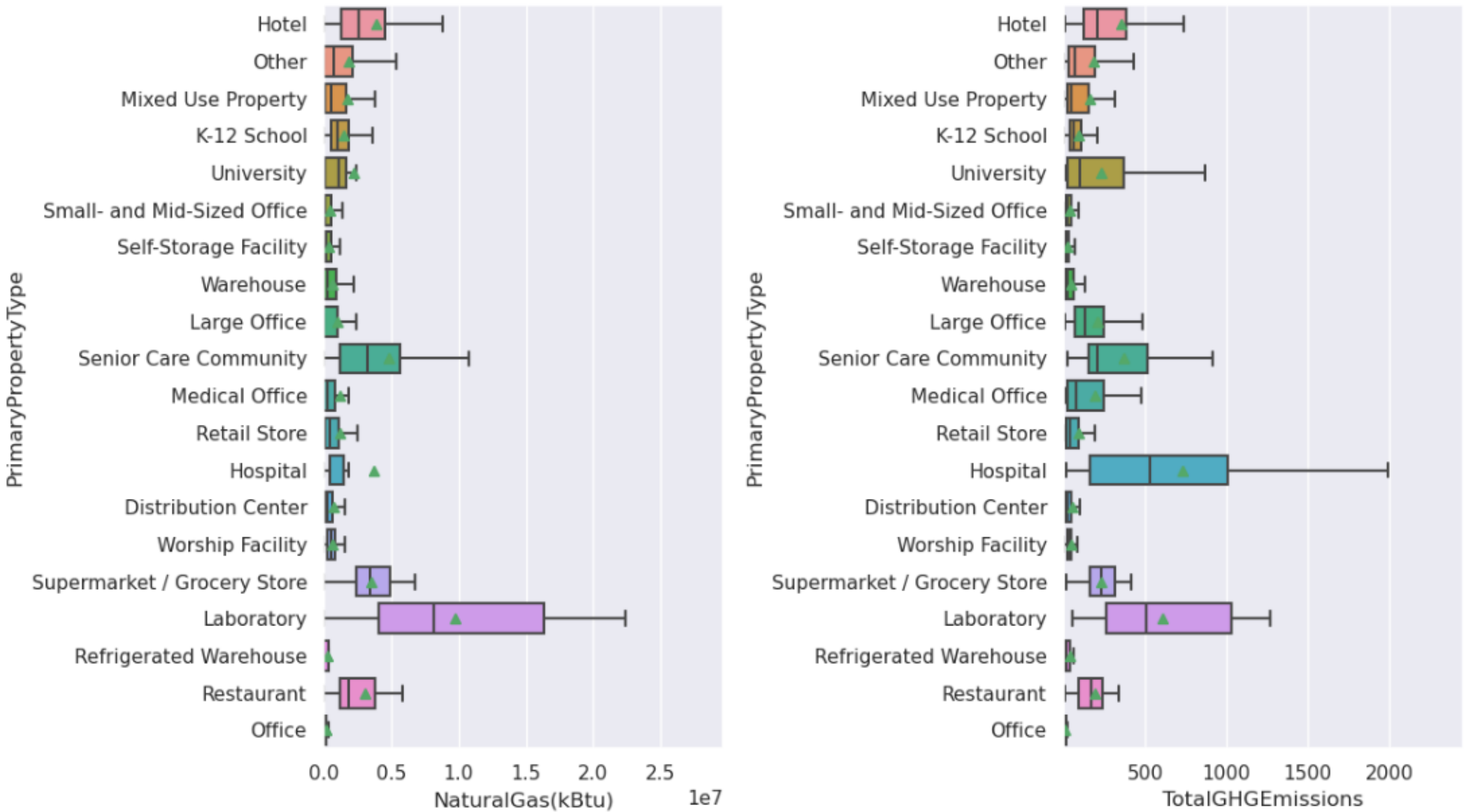
Nous avons donc créer cinq nouveaux prédicteurs en procédant comme suit :

- 1) Discrétisation de la variable **YearBuilt** en un maximum de classes (90 classes) ;
- 2) Pour chaque propriété d'une classe temporelle donnée, sa valeur pour la variable énergétique considérée est égale à la médiane des valeurs énergétiques de cette classe.
- 3) Passage au *log* pour les raisons déjà mentionnées.

Type de propriété et Énergie :

Nous avons affiché pour chaque variable énergétique, les *boxplots* dans chaque classe de la variable modale **PrimaryPropertyType**. Ci-dessous par exemple, les boxplots pour les caractéristiques de gaz naturel et de CO2 émis. Ici, la variable est modale et les classes ne sont pas de même effectif.

Contrairement aux deux cas précédents, il y a ici une plus grande variété de boxplots. et certains sont très étalés. Néanmoins, pour plus de la moitié des classes, les boxplots sont écrasés et on peut donc sans commettre d'erreur significative assimiler les valeurs de chaque classe à leur médiane ou leur moyenne, *pour ces classes*. Elles regroupent plus des deux-tiers des effectifs. Pour les autres classes, les valeurs peuvent également être assimilées à leur médiane ou leur moyenne avec un plus grand risque d'erreur cette fois. Les nouvelles variables ainsi obtenus sont donc moins bien corrélées aux variables énergétiques initiales.



Nous avons donc créer cinq nouveaux prédicteurs en attribuant, pour chaque variable énergétique (à l'exception de *SteamUse(kBtu)*) et chaque propriété d'un type donné, la valeur médiane des valeurs énergétiques des propriétés de même type.

4. Modélisation et résultats

Démarche générale :

1) Entraînement de modèles linéaires sur les données structurelles uniquement, sans inclure les nouveaux prédicteurs créés lors du feature engineering. (Pour la première cible uniquement)

»»» **Établissement d'un Benchmark de performance.**

2) Tests sur **les données enrichies** de différents modèles par ordre croissant de complexité : modèles linéaires régularisés (ElasticNet) puis modèles à arbres (Arbre de décision, Forêt aléatoire puis GradientBoosting). Recherches des hyperparamètres optimaux au moyen d'une recherche sur grille avec validation croisée.

3) Les performances ont été calculées et les modèles optimisés en utilisant comme métrique la MAPE (Mean Absolute Percentage Error), qui permet de connaître le pourcentage d'erreur moyenne commise par prédiction (la cible ne contient pas de valeurs nulles).

4) Les performances ont été calculées sur les ensembles d'entraînement et de test afin d'**évaluer l'overfit des modèles.**

5) Les modèles les plus performants ont été comparés pour choisir le modèle final selon plusieurs critères : simplicité du modèle, performance, robustesse (qui overfit peu) et vitesse d'exécution.

6) Le modèle finalement choisi a été optimisé de façon plus fine (*fine tuning*) et évalué sur un dataset de validation finale qui avait été créé au départ.

PREMIÈRE CIBLE : LA QUANTITÉ DE CO2 ÉMISE

Régressions linéaires sur les données uniquement structurelles :

Pour la régression linéaire, on obtient une MAPE de 3.68 sur l'ensemble de test, soit une erreur moyenne de 368% par prédiction !

Avec les régressions régularisées, on obtient des résultats à peine meilleurs avec une MAPE de 3.5.

>>> On devrait pouvoir facilement obtenir de meilleurs résultats avec un ensemble de prédiction mieux élaboré.

Tests de modèles linéaires sur les données enrichies :

Après avoir réalisé une recherche sur grille pour un ElasticNet et testé le modèle optimal trouvé, on obtient une MAPE de 2.85. C'est effectivement mieux qu'avec les données précédentes mais cela reste largement insuffisant.

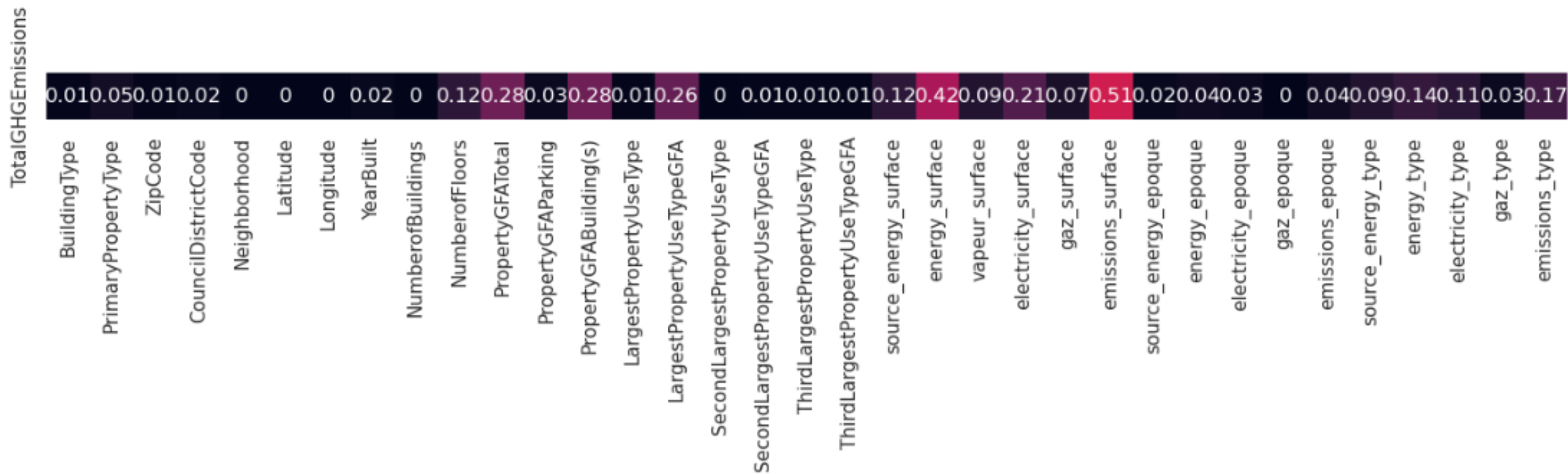
>>> On en déduit que les modèles linéaires ne conviennent pas pour prédire la cible telle quelle. Il faut ou tester d'autres modèles plus complexes ou modifier la cible au moyen d'une opération réversible afin de rendre linéaire sa relation avec les prédictors.

Nous avons suivi cette deuxième voie qui s'est avérée très payante !

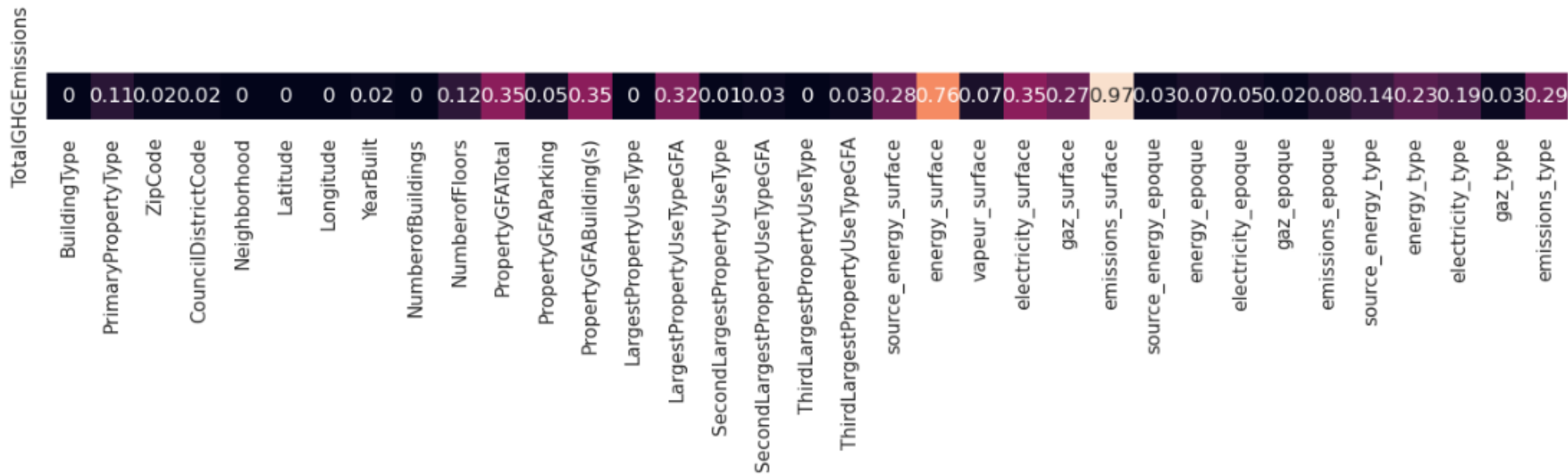
Passage au *log* de la cible :

Cette idée de passer au *log* la cible part du constat suivant : Certains des nouveaux prédicteurs ont été construits à partir du logarithme de la cible, donc la cible passée au *log* devrait être beaucoup mieux corrélée aux prédicteurs que la cible initiale, ce qui s'est avéré être le cas.

Voici la heatmap des corrélations de Pearson entre les prédicteurs et la cible initiale :



Voici la heatmap des corrélations avec la cible passée au *log* :



Après passage de la cible au log, nous avons à nouveau cherché le modèle linéaire régularisé optimal au moyen d'une recherche sur grille avec validation croisée . Après avoir testé le *Lasso* obtenu, on obtient une MAPE de 0.14, soit une erreur moyenne de 14% par prédiction, ce qui est satisfaisant.

On peut constater que le passage au *log* a permis une multiplication par 20 de la performance de la régression linéaire !

Tests de modèles plus complexes :

Nous avons testé des modèles à arbres *sur la cible initiale*, par ordre de complexité : arbre de décision, forêt aléatoire et GradientBoosting. Voici en résumé les résultats obtenus :

- **Arbre de décision** : MAPE = 0.24, overfit très important.
- **Forêt aléatoire** : MAPE = 0.21, overfit assez important.
- **GradientBoost** : MAPE = 0.22, overfit important.

>>> Le Lasso obtenu est plus simple, plus performant et n'overfit pas, c'est donc le modèle que j'ai retenu.

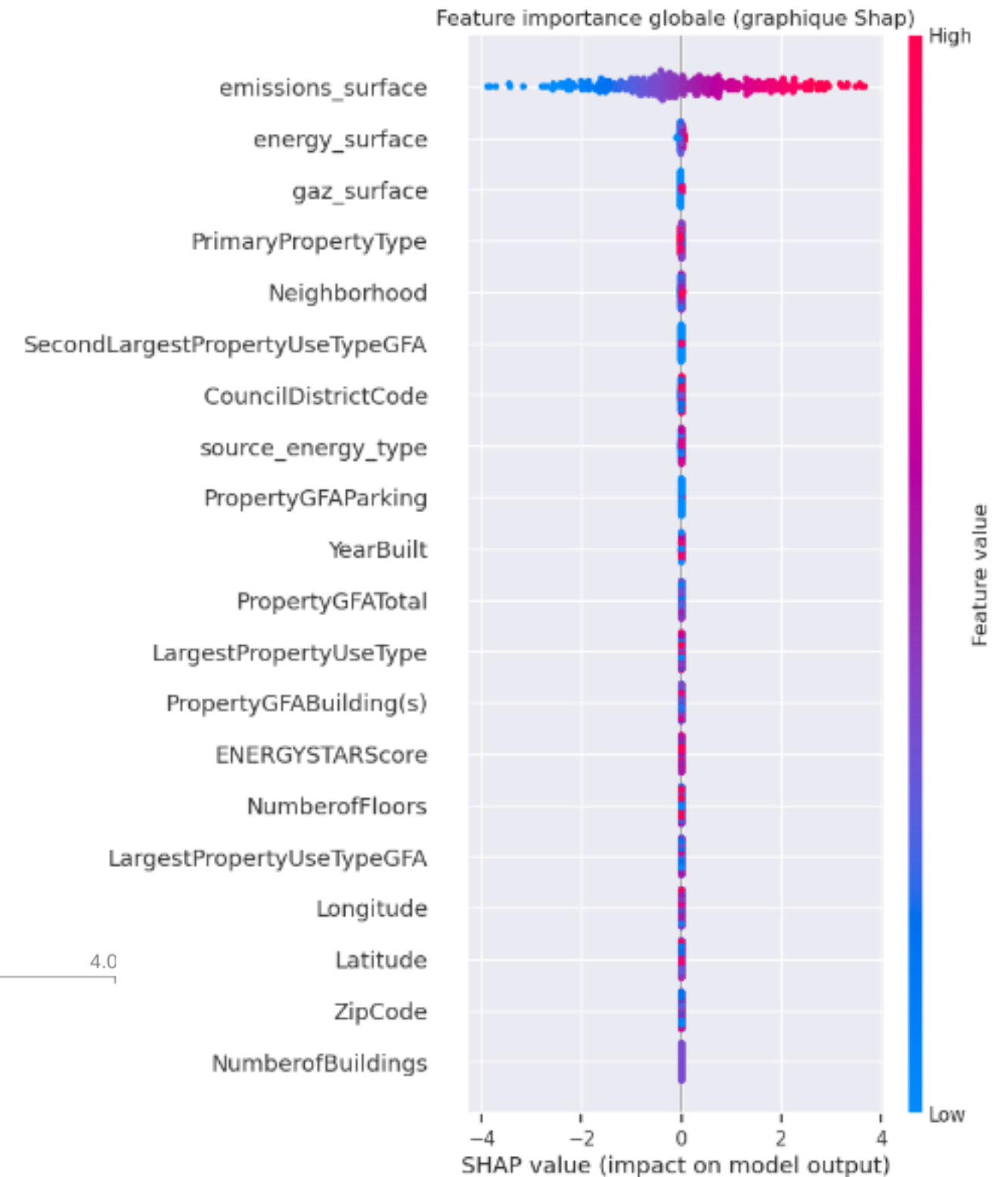
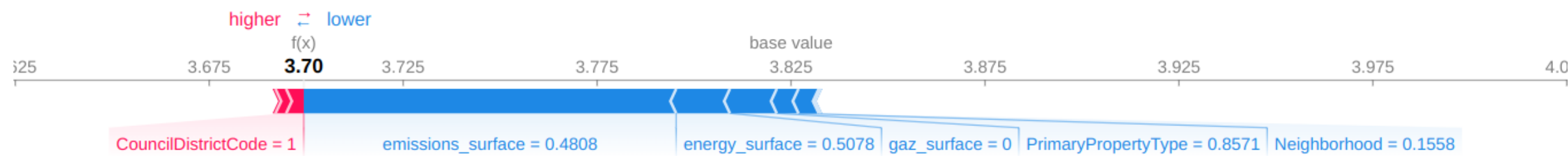
Feature Importance :

Après avoir optimisé le Lasso finalement choisi et l'avoir validé, nous avons réalisé une étude de feature importance globale puis locale. Dans le cas d'un modèle linéaire comme le Lasso, l'importance d'un prédicteur dans la détermination de la cible est simplement donnée par son coefficient de régression : plus ce dernier est grand en valeur absolue, plus ce prédicteur est important.

Ci-contre le graphique d'importance globale des prédicteurs obtenu en utilisant la bibliothèque **SHAP** (SHapley Additive exPlanations). On peut constater :

- l'importance éclipsante du prédicteur **emissions_surface** qui s'explique par le fait que, par construction, ce prédicteur est très fortement corrélé au *log* de la cible ;
- que l'**ENERGYSTARScore** n'a aucune influence dans prédiction. Cette variable ne peut donc contribuer à prédire les émissions CO2.

On peut faire les mêmes observations au niveau local, pour une prédiction donnée. Voici par exemple le graphique de feature importance pour la 33ème ligne du dataset de test.



DEUXIÈME CIBLE : LA QUANTITÉ D'ÉNERGIE CONSOMMÉE

Pour cette cible, tous les modèles ont été testés directement sur les données enrichies.

Tests de modèles linéaires sur les données enrichies :

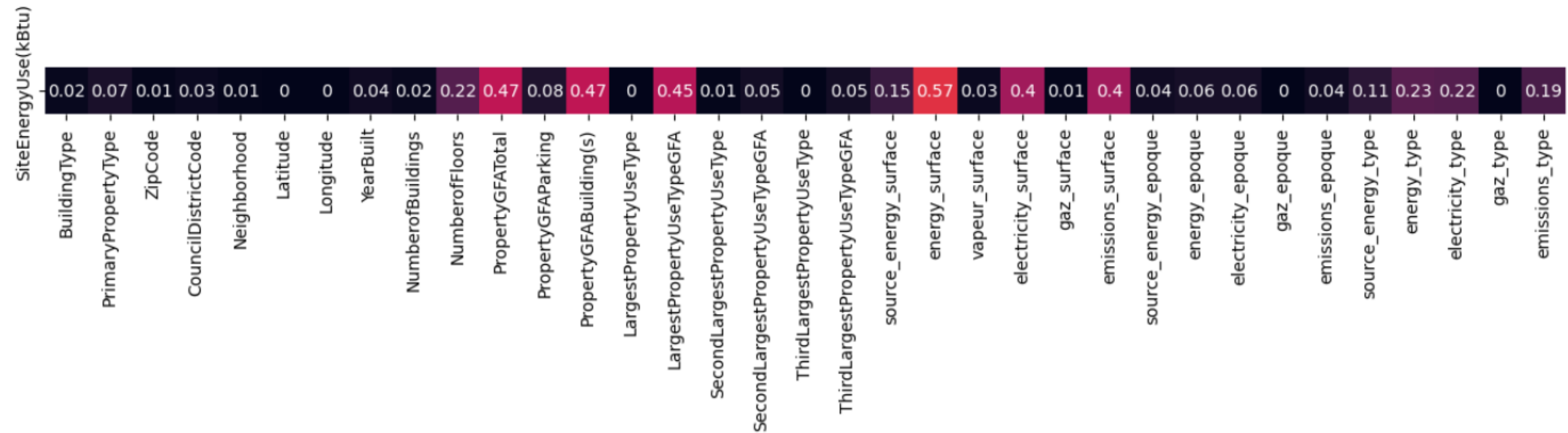
Après avoir réalisé une recherche sur grille pour un ElasticNet et testé le modèle optimal trouvé, on obtient une MAPE de 1.63, soit une erreur moyenne de 163% par prédiction. C'est mieux que pour la cible précédente mais très insuffisant.

>>> À nouveau, on en déduit que les modèles linéaires ne conviennent pas pour prédire la cible telle quelle. Il faut ou tester d'autres modèles plus complexes ou modifier la cible au moyen d'une opération réversible afin de rendre linéaire sa relation avec les prédicteurs.

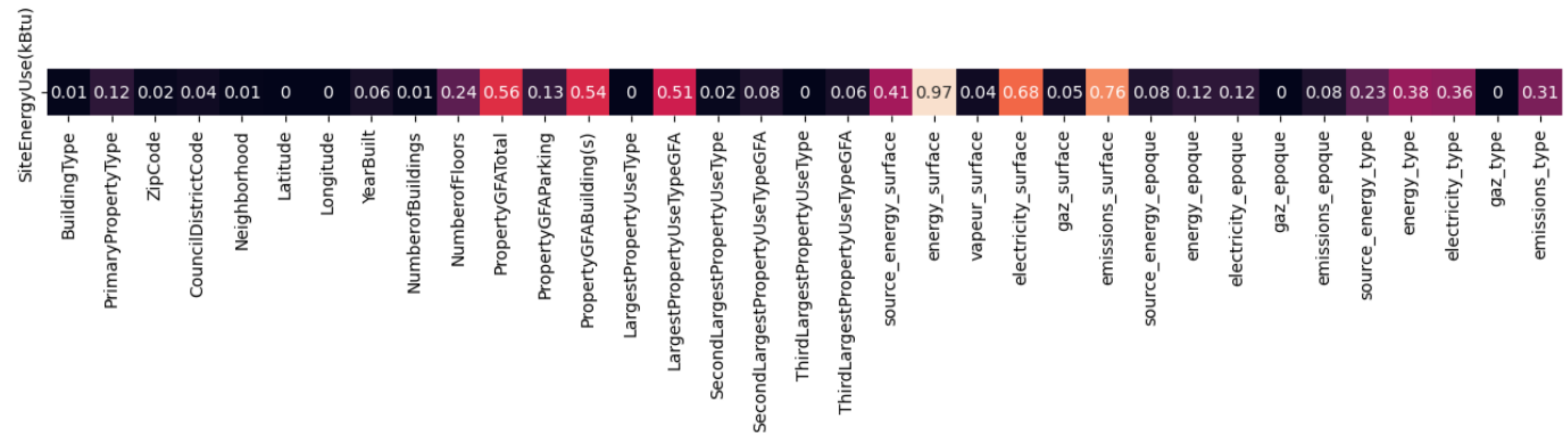
Nous avons suivi cette deuxième voie en passant au *log* la cible. Cette démarche c'est à nouveau avérée très fructueuse !

Passage au *log* de la cible :

Voici la heatmap des corrélations de Pearson entre les prédicteurs et la cible initiale :



Voici la heatmap des corrélations avec la cible passée au *log* :



Tests de modèles par ordre croissant de complexité.

Cette fois, tous les modèles ont été testés sur le log de la cible.

Comme pour la cible précédente, différents modèles ont été testés et optimisés par ordre croissant de complexité, au moyen d'une recherche sur grille avec validation croisée .

Voici en résumé les résultats obtenus :

- **Modèles linéaires** : Lasso, MAPE = 0.07, pas d'overfit.
- **Arbre de décision** : MAPE = 0.16, overfit très important.
- **Forêt aléatoire** : MAPE = 0.10, overfit assez important.
- **GradientBoost** : MAPE = 0.10, overfit assez important.

>>> Le Lasso obtenu est plus simple, plus performant et robuste, c'est donc encore une fois le modèle que j'ai retenu.

Feature importance.

On peut constater, en observant les coefficients de régression, le poids écrasant de la feature **energy_surface** dans la prédiction de cette cible. Cela était prévisible car cette feature a un coefficient de corrélation de Pearson égal à 0.97 avec la cible passée au *log*.