Projet 4 :
Anticipez les besoins en consommation électrique de bâtiments

Raphaël GIRAUDOT

Sommaire

1. Contexte & Objectifs de la mission

La ville de Seattle à pour objectif de devenir une ville neutre en émissions de CO2 en 2050.

Actuellement, elle suit les émissions de CO2 et la consommation d'énergie de bâtiments grâce à des relevés annuels effectués par des agents de la ville.

Seulement, ces relevés sont longs et coûteux et la ville cherche donc à s'en passer à l'avenir.

Le but est donc, à partir de relevés effectués en 2015 et 2016, de pouvoir prédire les émissions de CO2 et la consommation totale d'énergie des bâtiments non destinée à l'habitation via une modélisation.

Comme on va chercher à estimer une quantitée on va réaliser des modélisation de type régression.

2. Nettoyage & Fusion des 2 fichiers

Uniformisation des colonnes :

- Dans data_2015 :
  - Création des colonnes : *'Address'*, *'City'*, *'State'*, *'ZipCode'*, *'Latitude'*, *'Longitude'* à partir de la colonne **'Location'**.
  - Retrait des colonnes : *'OtherFuelUse(kBtu)'*, *'2010 Census Tracts'*, *'Seattle Police Department Micro Community Policing Plan Areas'*, *'City Council Districts'*, *'SPD Beats'*, *'Zip Codes'*
- Dans data_2016 :
  - Renommage de *'Comments'* en *'Comment'*
- Dans les 2 fichiers :
  - *'GHGEmissions(MetricTonsCO2e)'* & *'TotalGHGEmissions'* -> **'TotalGHGEmissions(MetricTonsCO2e)'**
  - *'GHGEmissionsIntensity(kgCO2e/ft2)'* & *'GHGEmissionsIntensity'* -> **'GHGEmissionsIntensity(kgCO2e/ft2)'**

Ensuite on uniformise les types des colonnes pour pouvoir ensuite fusionner les deux fichiers.

col data_2015 **not in** data_2016	col data_2016 **not in** data_2015
Location	Address
OtherFuelUse(kBtu)	City
GHGEmissions(MetricTonsCO2e)	State
GHGEmissionsIntensity(kgCO2e/ft2)	ZipCode
Comment	Latitude
2010 Census Tracts	Longitude
Seattle Police Department Micro Community Policing Plan Areas	Comments
City Council Districts	TotalGHGEmissions
SPD Beats	GHGEmissionsIntensity
Zip Codes	

3. Récapitulatif du fichier créé

- Nombre d'individus (après retrait des bâtiment résidentiels):
  - 3287 dans le fichier créé (48,94% des individus de data_2015 & data_2016)
  - 1639 de data_2015 (49,86% du fichier)
  - 1648 de data_2016 (50,14% du fichier)

- 46 variables :
  - Identification : **OSEBuildingID, DataYear, PropertyName, TaxParcelIdentificationNumber**
  - Localisation : **Address, City, State, ZipCode, CouncilDistrictCode, Latitude, Longitude, Neighborhood,**
  - Usage : **BuildingType, PrimaryPropertyType, PropertyGFAParking, PropertyGFABuilding(s), ListOfAllPropertyUseTypes, LargestPropertyUseType, LargestPropertyUseTypeGFA, SecondLargestPropertyUseType, SecondLargestPropertyUseTypeGFA, ThirdLargestPropertyUseType, ThirdLargestPropertyUseTypeGFA,**
  - Informations générales : **YearBuilt, NumberofBuildings, NumberofFloors, PropertyGFATotal,**
  - Energie : **YearsENERGYSTARCertified, ENERGYSTARScore, SiteEUI(kBtu/sf), SiteEUIWN(kBtu/sf), SourceEUI(kBtu/sf), SourceEUIWN(kBtu/sf), SiteEnergyUse(kBtu), SiteEnergyUseWN(kBtu), SteamUse(kBtu), Electricity(kWh), Electricity(kBtu), NaturalGas(therms), NaturalGas(kBtu), TotalGHGEmissions(MetricTonsCO2e), GHGEmissionsIntensity(kgCO2e/ft2)**
  - Autres : **DefaultData, Comment, ComplianceStatus, Outlier**

4. Feature Engineering

On va créer de nouvelles variables à partir des variables du fichier :

- **‘BuildingAge’ = 'DataYear'-'YearBuilt'** : Âge du bâtiment

- **‘BuildingProp’ = 'PropertyGFABuilding(s)'/'PropertyGFATotal'*100** :
  Proportion de surface bâtie par rapport à la surface totale

- **‘TopUseTypeProp’ = 'LargestPropertyUseTypeGFA'/'PropertyGFATotal'*100** :
  Proportion de l’usage qui occupe le plus de surface par rapport à la surface totale

5. Feature Selection

Pour faciliter l'entraînement on va retirer des variables de notre DataSet

- Retrait des variables qui ne varient pas : ***City, State, ZipCode***
- Retrait des variables qui sont liées directement à une autre variable déjà présente
- Retrait des features très corrélées entre-elles
- Isole les variables d'identification : **'OSEBuildingID', 'DataYear', 'PropertyName', 'Address' ,'TaxParcelIdentificationNumber'**
- Retrait des variables qui concernent les relevés énergétiques

En tout 15 features restantes.

6. Analyse : Numerical Features

- ENERGYSTARScore : Score de performance énergétique

  De 0 à 100. 50% des bâtiments ont un score >72 soit une bonne performance énergétique.
  33,63% des individus n'ont pas d'ENERGYSTARScore

- NumberofFloors : Nombre d'étages

  La moitié des bâtiments ont, au maximum, 2 étages.

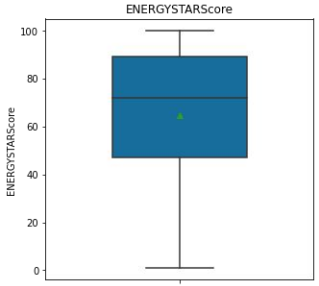
- NumbersofBuildings : Nombre de bâtiments

  Au moins 75% des individus = 1 bâtiment

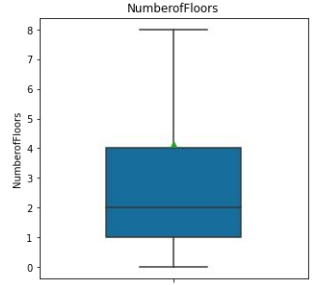
```
count    2211.000000
mean       64.815920
std        28.549886
min         1.000000
25%        47.000000
50%        72.000000
75%        89.000000
max       100.000000
```

```
nbNaCol(data, 'ENERGYSTARScore')
```

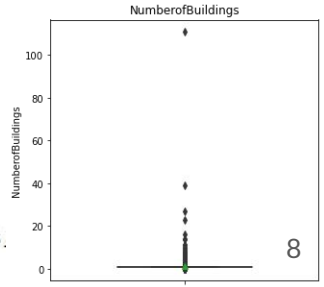
```
(1107, 33.36347197106691)
```




```
count    3310.000000
mean        4.126888
std         6.567333
min         0.000000
25%         1.000000
50%         2.000000
75%         4.000000
max        99.000000
```




```
count    3316.000000
mean        1.117310
std         2.219845
min         0.000000
25%         1.000000
50%         1.000000
75%         1.000000
max       111.000000
Name: NumberofBuildings
```



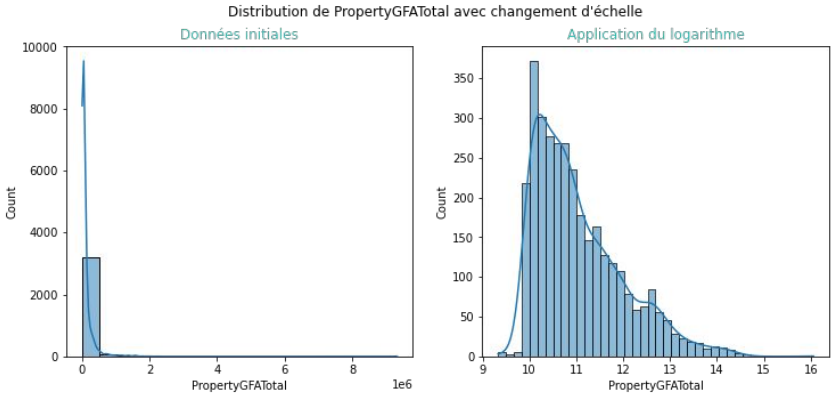
6. Analyse : Numerical Features

On observe que PropertyGFATotal présente une distribution étalée allant de 11 571 sq.ft à 9 320 156 sq.ft.

Afin de se ramener à une même échelle on réalise une transformation par le log() . On observe alors une distribution se rapprochant du distribution Normale, courbe en cloche.

On va donc ajouter une colonne ‘PropertyGFATotal_log’ dans nos features.

```
count    3.287000e+03
mean     1.157180e+05
std      2.512306e+05
min      1.128500e+04
25%      2.940850e+04
50%      4.905200e+04
75%      1.048430e+05
max      9.320156e+06
Name: PropertyGFATotal
```




```
count    3287.000000
mean       11.044373
std         0.948789
min         9.331230
25%        10.289039
50%        10.800636
75%        11.560219
max        16.047690
Name: PropertyGFATotal_log
```

6. Analyse : Numerical Features (Créées)

- BuildingProp : Proportion de Bâtiment

  De 0 à 100. Pour au moins 75% des individus la surface bâtie représente 100% de la propriété

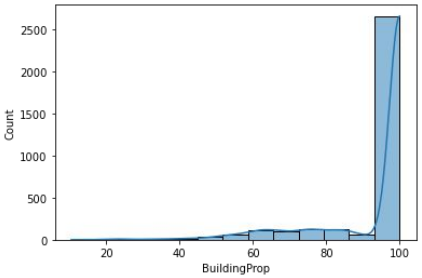
- TopUseTypeProp : Proportion de l'usage majoritaire de propriété

  De 0 à 100. On observe que pour 50% des individus l'usage majoritaire représente plus de 96% de la propriété

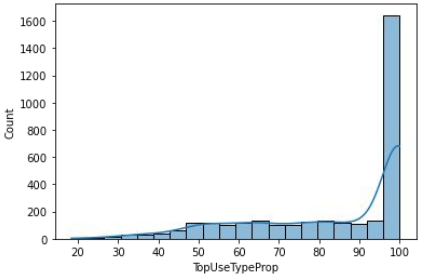
- BuildingAge : Age du bâtiment

  De 1 à 116 ans. En moyenne les bâtiments ont ≈54 ans, l'âge des bâtiments est assez bien répartie avec un âge médian à 50 ans proche de l'âge moyen.

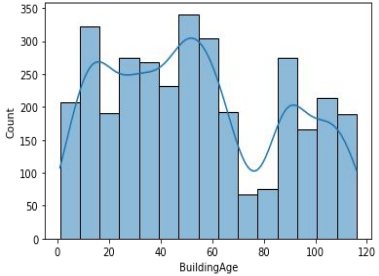
count	3316.000000
mean	93.857770
std	13.834463
min	10.497748
25%	100.000000
50%	100.000000
75%	100.000000
max	100.002922



count	3245.000000
mean	84.046597
std	20.120732
min	18.345763
25%	69.084931
50%	96.210593
75%	100.000000
max	100.000000

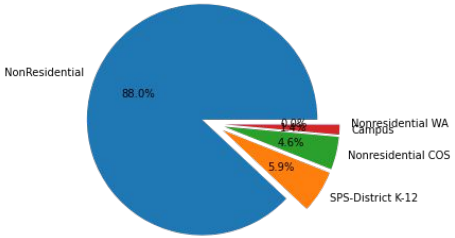


count	3318.000000
mean	53.965943
std	32.707800
min	1.000000
25%	27.000000
50%	50.000000
75%	86.000000
max	116.000000

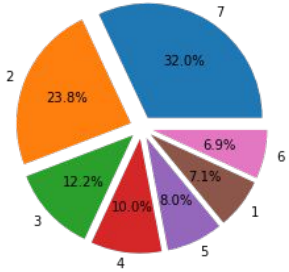


6. Analyse : Categorical Features

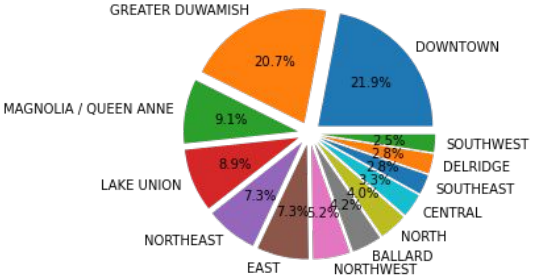
BuildingType :



CouncilDistrictCode :



Neighborhood :



- LargestPropertyUseType :

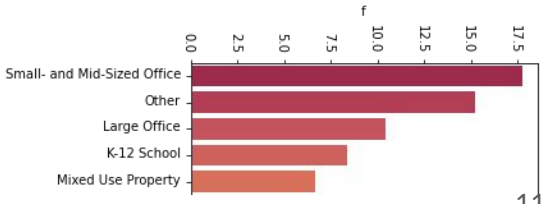
Usage de la propriété qui prends le plus de surface. Les 3 premières modalités concentrent plus de 50% des individus.

LargestPropertyUseType	n	f
OFFICE	975	30.027718
NON-REFRIGERATED WAREHOUSE	393	12.103480
K-12 SCHOOL	275	8.469356
RETAIL STORE	198	6.097937
OTHER	198	6.097937

- PrimaryPropertyUseType :

Usage principal de la propriété. La moitié des individus sont représentés par 4 modalités.

	PrimaryPropertyType	n	f
0	Small- and Mid-Sized Office	588	17.721519
1	Other	502	15.129596
2	Large Office	344	10.367691
3	K-12 School	275	8.288125
4	Mixed Use Property	220	6.630500

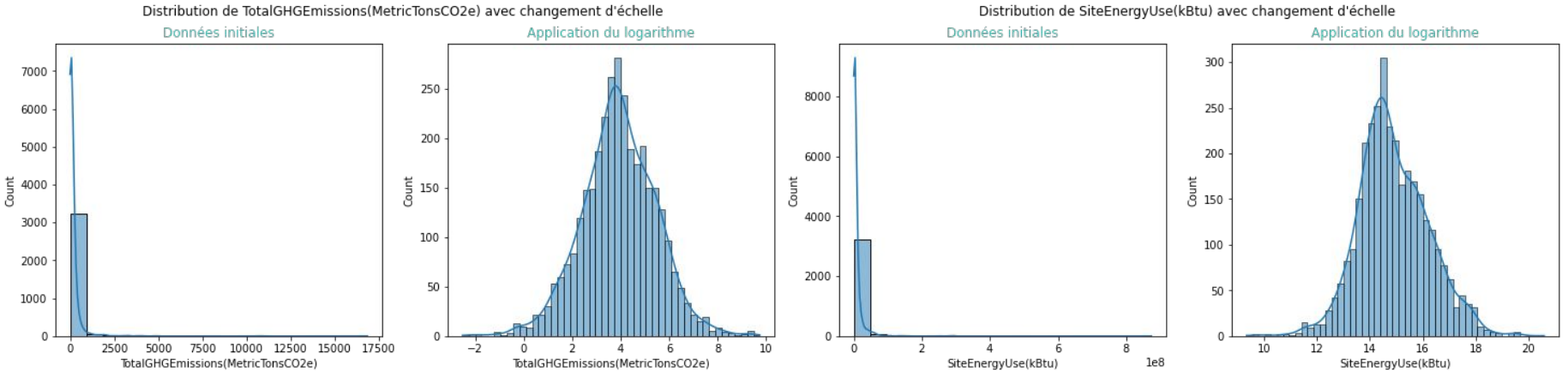


6. Analyse : Targets

La distribution des variables à prédire, nos targets, sont assez étalées vers des hautes valeurs.

On remarques qu'en appliquant une transformation logarithmique on se ramène à une distribution proche d'une distribution normale, avec une courbe en cloche.

On va donc entraîner nos différents modèles sur les données initiales et sur les données transformées pour voir si on à un gain de performance.



7. Préparation des données pour la modélisation

- Retrait des individus présentant des données manquantes dans les variables à prédire

- Création de 2 dataframes :
  - X : Dataframe contenant les features
  - y : Dataframe contenant les targets (SiteEnergyUse(kBtu) & TotalGHGEmissions(MetricTonsCO2e))

- Création du Train et Test set :
  - Train set : 80% des individus -> 2 629
  - Test set : 20% des individus -> 658

8. Preprocessing

Création de pipelines de preprocessing pour s'assurer que toutes les données subissent la même transformation avant d'entraîner nos modèles :

- Numerical features :
  - Simple Imputer : Impute par la médiane les valeurs manquantes
  - Standard scaler : Standardise les données
- Categorical features :
  - Simple Imputer : Impute les valeurs manquantes par 'missing'
  - OneHotEncoder : Encode les modalités des variables catégorielles

On englobe alors ces 2 pipelines dans un 'make_column_transformer', On en crée 2 :

- Avec les features 'de base'
- Avec les features 'de base' sauf pour 'PropertyGFATotal' qui est remplacée par 'PropertyGFATotal_log' et que l'on appliquera pour entraîner les modèles sur les targets transformée par un log

9. a. Modélisation - Méthodologie

On va donc maintenant utiliser plusieurs techniques de modélisation.

Pour évaluer les performances des modèles, on va utiliser :

- Les score de R^2 et la RMSE, sur cross-validation et sur l'ensemble de test
- Les temps d'exécution (fit & scoring) mais aussi le temps de recherche des paramètres optimaux pour les modèles avec hyperparamètres.

On va stocker ces métriques dans un tableau afin de pouvoir comparer facilement les performances de chaques modèles.

On enregistrera aussi la valeur des paramètres optimaux trouvé via le GridSearchCV.

9. a. Modélisation - Méthodologie

Pour chaques modèles :

1. Pipeline modèle = preprocessor + modèle
    a. *(suivant le modèle) définition des paramètres, hyperparamètres, à optimiser*
    b. *GridSearchCV pour trouver les paramètres qui renvoient le meilleur score R^2*
2. Cross-validation du modèle *(avec les paramètres optimaux)*
3. Entrainement du modèle sur le training_set
4. Score du modèle sur le test_set
5. Enregistrement des métriques dans le tableau 'df_score'

On teste chaque type de modèles sur nos 2 targets et sur les 2 targets avec transformation par le log(), on a donc 4 modèles différents pour chaque type de modélisation.

9. a. Modélisation - Méthodologie

Modèles testés :

1. Dummy Regressor
2. Régression :
    a. Linéaire
    b. Ridge
    c. Lasso
    d. ElasticNet
3. Méthodes ensemblistes :
    a. Bagging
    b. Random Forest
    c. Gradient Boosting

9. b. Modélisation - Dummy Regressor

C'est un modèle "bête" , c'est à dire qu'il va prédire une valeur unique, dans notre cas la moyenne de la target du training set, peut importe les features.

Avec un tel modèle on ne s'attends pas à avoir de bonnes performances mais il va nous servir de "baseline" c'est à dire que l'on va pouvoir comparer les performances des autres modèles à celles du Dummy Regressor.

	nom_modele	target	commentaires	r2_cv	rmse_cv	r2_test	rmse_test	fit_time	score_time	time_total
0	Dummy	SiteEnergyUse(kBtu)	mean	-0.005799	2.324259e+07	-0.000125	2.365766e+07	0.093754	0.033909	0.127663
1	Dummy	SiteEnergyUse(kBtu)	mean-Log(y)	-0.063034	2.377979e+07	-0.051329	2.425570e+07	0.085070	0.032159	0.117229
2	Dummy	TotalGHGEmissions(MetricTonsCO2e)	mean	-0.010259	6.168327e+02	-0.000698	7.355771e+02	0.083774	0.033911	0.117684
3	Dummy	TotalGHGEmissions(MetricTonsCO2e)	mean-Log(y)	-0.051433	6.281297e+02	-0.037648	7.490343e+02	0.091757	0.038895	0.130652

On remarques donc des performances médiocre, un r2 proche de 0, en cv et en test

9. b. Modélisation - Régression Linéaire

La Régression Linéaire va chercher a estimer des coefficients pour chaques features en cherchant à minimiser l'erreur globale.

	nom_modele	target	commentaires	r2_cv	rmse_cv	r2_test	rmse_test	fit_time	score_time	time_total
4	RegLin	SiteEnergyUse(kBtu)		0.560859	1.425792e+07	0.728881	1.231756e+07	0.272002	0.039219	0.311221
5	RegLin	SiteEnergyUse(kBtu)	Log(y)	0.614162	1.484360e+07	0.906225	7.244157e+06	0.234627	0.027926	0.262554
6	RegLin	TotalGHGEmissions(MetricTonsCO2e)		0.365145	4.458108e+02	0.670752	4.219282e+02	0.220010	0.026927	0.246937
7	RegLin	TotalGHGEmissions(MetricTonsCO2e)	Log(y)	0.481522	4.233367e+02	0.840200	2.939439e+02	0.257373	0.035850	0.293223

r2_test >> r2_ cv : bonne généralisation du modèle

log(y) >> y

9. b. Modélisation - Régression Ridge

La régression Ridge se base sur une régression linéaire que l'on va régulariser à l'aide de la norme l2 des coefficient de la régression linéaire.

Cela va permettre de réduire la variance de certaines variables et donc de réduire l'impact de certaines features du modèles.

On va tester plusieurs valeur d'alpha, le coefficient de régularisation, plus alpha est élevé plus la régularisation sera élevée.
alpha = 0 : pas de régularisation = régression linéaire.

	nom_modele	target	commentaires	r2_cv	rmse_cv	r2_test	rmse_test	fit_time	score_time	time_total
8	Ridge	SiteEnergyUse(kBtu)	(alpha, 1.0)	0.572036	1.410388e+07	0.711218	1.271246e+07	0.098736	0.030945	0.129681
9	Ridge	SiteEnergyUse(kBtu)	(alpha, 0.8, Log(y))	0.624592	1.463711e+07	0.897376	7.578253e+06	0.116662	0.029895	0.146556
10	Ridge	TotalGHGEmissions(MetricTonsCO2e)	(alpha, 1.0)	0.374101	4.453388e+02	0.669050	4.230167e+02	0.121624	0.033908	0.155532
11	Ridge	TotalGHGEmissions(MetricTonsCO2e)	(alpha, 1.0, Log(y))	0.511793	4.094782e+02	0.847702	2.869614e+02	0.105181	0.029952	0.135133

alpha_optimal = 1 (très proche de 1) : forte régularisation sachant que plage d'alpha testée : 0 -> 1

r2_test >> r2_ cv : bonne généralisation du modèle

log(y) >> y

9. b. Modélisation - Régression Lasso

La régression Lasso est aussi une méthode de régularisation d'une régression linéaire, par la norme l1 des coefficients.

Contrairement à la régression de Ridge, cette régularisation cherche à annuler certains coefficient de la régression linéaire, ce qui revient à éliminer, sélectionner, certaines features qui ne sont pas nécessaire à la prédiction.

Comme pour la régression de ridge on va tester plusieurs valeurs de alpha, coefficient de régularisation.

	nom_modele	target	commentaires	r2_cv	rmse_cv	r2_test	rmse_test	fit_time	score_time	time_total
12	Lasso	SiteEnergyUse(kBtu)	(alpha, 25849)	0.580892	1.400607e+07	0.706894	1.280727e+07	0.215395	0.025938	0.241333
13	Lasso	SiteEnergyUse(kBtu)	(alpha, 0.03, Log(y))	0.608708	1.329586e+07	0.459825	1.738648e+07	0.096764	0.028923	0.125687
14	Lasso	TotalGHGEmissions(MetricTonsCO2e)	(alpha, 0.9)	0.397180	4.399477e+02	0.656716	4.308278e+02	0.267247	0.032946	0.300193
15	Lasso	TotalGHGEmissions(MetricTonsCO2e)	(alpha, 0.05, Log(y))	0.267517	5.276179e+02	0.238858	6.415189e+02	0.104925	0.028473	0.133399

y : r2_test >> r2_ cv : bonne généralisation du modèle

log(y) : r2_test < r2_ cv : overfitting

r2_test (y) > r2_test(log(y))

9. b. Modélisation - Régression ElasticNet

La régression ElasticNet est régularisation utilisant une combinaison de la norme l1 et l2. Pour ce modèle on va devoir optimiser 2 paramètres :

- alpha : coefficient de régularisation
- l1_ratio : compris entre 1 et 0, si l1_ratio =0 -> utilisation de la norme l2 (Ridge).

	nom_modele	target	commentaires	r2_cv	rmse_cv	r2_test	rmse_test	fit_time	score_time	time_total
16	ElasticNet	SiteEnergyUse(kBtu)	(alpha, 0.1, - l1, 0.95)	0.589743	1.376844e+07	0.623808	1.450940e+07	0.633669	0.030896	0.664565
17	ElasticNet	SiteEnergyUse(kBtu)	(alpha, 0.1, - l1, 0.5, Log(y))	0.497777	1.639639e+07	0.376420	1.868059e+07	0.101730	0.031030	0.132760
18	ElasticNet	TotalGHGEmissions(MetricTonsCO2e)	(alpha, 0.1, l1, 0.95)	0.396133	4.547571e+02	0.617331	4.548708e+02	0.237298	0.028938	0.266236
19	ElasticNet	TotalGHGEmissions(MetricTonsCO2e)	(alpha, 0.1, l1, 0.7, Log(y))	0.224078	5.397447e+02	0.164915	6.719578e+02	0.101698	0.027930	0.129628

y : r2_test >> r2_ cv : bonne généralisation du modèle

log(y) : r2_test < r2_ cv : overfitting

r2_test (y) > r2_test(log(y))

9. b. Modélisation - Bagging

Cette méthode créer de nouveaux train sets de même taille que le train set par échantillonnage du trainset (bootstrapping).

On entraîne alors un estimateur, ici des arbres de décisions, pour chaque échantillon de bootstrap.

La prédiction finale est alors donnée par un méta-estimateur, qui combine l'ensemble des prédictions des modèle, ici la moyenne.

Ici le paramètre que l'on cherche à optimiser est le nombre d'estimateur de notre modèle.

	nom_modele	target	commentaires	r2_cv	rmse_cv	r2_test	rmse_test	fit_time	score_time	time_total
20	Bagging	SiteEnergyUse(kBtu)	(n_estimators:, 30)	0.621751	1.506088e+07	0.895782	7.636865e+06	6.873854	0.078008	6.951862
21	Bagging	SiteEnergyUse(kBtu)	(n_estimators:, 40, Log(y))	0.599755	1.561747e+07	0.742291	1.200906e+07	8.707860	0.110676	8.818536
22	Bagging	TotalGHGEmissions(MetricTonsCO2e)	(n_estimators:, 40)	0.581370	4.038613e+02	0.927506	1.979828e+02	9.462214	0.093359	9.555574
23	Bagging	TotalGHGEmissions(MetricTonsCO2e)	(n_estimators:, 40, Log(y))	0.515605	4.430565e+02	0.817461	3.141622e+02	8.422318	0.088790	8.511109

r2_test >> r2_ cv : bonne généralisation du modèle

r2_test (y) > r2_test(log(y))

9. b. Modélisation - Random Forest

La différence avec le bagging réside dans le fait que le Random Forest ne prends pas en compte toutes les features pour créer un noeud, l'algorithme sélectionne les features qui maximisent le gain d'information.

Les paramètres que l'on va chercher à optimiser ici sont :

- le nombre d'arbres de décision dans la forêt
- la profondeur maximale de chaque arbre
- le gain d'information minimum pour créer un nouveau noeud dans les arbres

	nom_modele	target	commentaires	r2_cv	rmse_cv	r2_test	rmse_test	fit_time	score_time	time_total
24	RandomForest	SiteEnergyUse(kBtu)	(n_estim, 200, - max_depth, None, - min_impuri...	0.629400	1.485904e+07	0.894802	7.672713e+06	47.050140	0.165509	47.215649
25	RandomForest	SiteEnergyUse(kBtu)	(n_estim, 200, - max_depth, None, - min_impuri...	0.599209	1.562721e+07	0.731347	1.226142e+07	43.796619	0.165504	43.962123
26	RandomForest	TotalGHGEmissions(MetricTonsCO2e)	(n_estim, 200, - max_depth, None, - min_impuri...	0.577085	4.052957e+02	0.906621	2.246994e+02	42.336397	0.150594	42.486991
27	RandomForest	TotalGHGEmissions(MetricTonsCO2e)	(n_estim, 200, - max_depth, None, - min_impuri...	0.516922	4.428240e+02	0.807603	3.225343e+02	42.201021	0.163614	42.364635

r2_test >> r2_ cv : bonne généralisation du modèle

r2_test (y) > r2_test(log(y))

9. b. Modélisation - Gradient Boosting

Au lieu d'entraîner nos arbres simultanément, on va chercher à optimiser nos arbres de décision en entraînant des arbres de plus en plus précis en se basant sur les arbres précédent. Les paramètres que l'on va chercher à optimiser sont les mêmes que pour le RandomForest.

	nom_modele	target	commentaires	r2_cv	rmse_cv	r2_test	rmse_test	fit_time	score_time	time_total
28	Gradient Boosting	SiteEnergyUse(kBtu)	(n_estim, 2000, - max_depth, 3, - min_impurity...	0.634331	1.482951e+07	0.929548	6.279025e+06	32.236940	0.159577	32.396518
29	Gradient Boosting	SiteEnergyUse(kBtu)	(n_estim, 1000, - max_depth, 3, - min_impurity...	0.785572	1.116561e+07	0.813886	1.020550e+07	16.173985	0.100741	16.274726
30	Gradient Boosting	TotalGHGEmissions(MetricTonsCO2e)	(n_estim, 2000, - max_depth, 3, - min_impurity...	0.575811	4.102282e+02	0.952193	1.607758e+02	31.722226	0.172048	31.894274
31	Gradient Boosting	TotalGHGEmissions(MetricTonsCO2e)	(n_estim, 500, - max_depth, 3, - min_impurity_...	0.634617	3.650292e+02	0.918683	2.096845e+02	8.182434	0.060919	8.243353

r2_test >> r2_ cv : bonne généralisation du modèle

r2_test (y) > r2_test(log(y))

9. c. Modélisation - Comparaison des meilleurs modèles

On va s'intéresser ici au score r2_test et sélectionner les modèles les plus performants. On peut aussi regarder la rmse_test sur ces modèles.

En effet, le score r2 est plus facilement “lisible” , par contre la rmse nous permettra d’échanger avec les équipes métier.

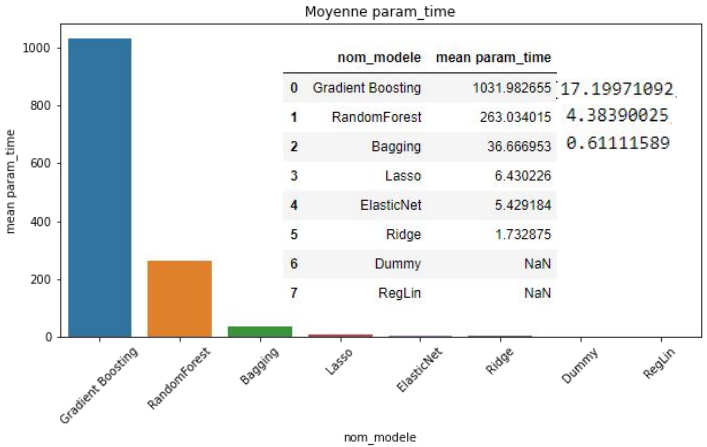
	nom_modele	target	target_transform	commentaires	r2_cv	rmse_cv	r2_test	rmse_test
28	Gradient Boosting	SiteEnergyUse(kBtu)	y	(n_estim, 2000, - max_depth, 3, - min_impurity...	0.634331	1.482951e+07	0.929548	6.279025e+06
5	RegLin	SiteEnergyUse(kBtu)	Log(y)	Log(y)	0.614162	1.484360e+07	0.906225	7.244157e+06
9	Ridge	SiteEnergyUse(kBtu)	Log(y)	(alpha, 0.8, Log(y))	0.624592	1.463711e+07	0.897376	7.578253e+06
20	Bagging	SiteEnergyUse(kBtu)	y	(n_estimators:, 30)	0.621751	1.506088e+07	0.895782	7.636865e+06
24	RandomForest	SiteEnergyUse(kBtu)	y	(n_estim, 200, - max_depth, None, - min_impuri...	0.629400	1.485904e+07	0.894802	7.672713e+06

	nom_modele	target	target_transform	commentaires	r2_cv	rmse_cv	r2_test	rmse_test
30	Gradient Boosting	TotalGHGEmissions(MetricTonsCO2e)	y	(n_estim, 2000, - max_depth, 3, - min_impurity...	0.575811	410.228158	0.952193	160.775751
22	Bagging	TotalGHGEmissions(MetricTonsCO2e)	y	(n_estimators:, 40)	0.581370	403.861283	0.927506	197.982801
31	Gradient Boosting	TotalGHGEmissions(MetricTonsCO2e)	Log(y)	(n_estim, 500, - max_depth, 3, - min_impurity_...	0.634617	365.029174	0.918683	209.684519
26	RandomForest	TotalGHGEmissions(MetricTonsCO2e)	y	(n_estim, 200, - max_depth, None, - min_impuri...	0.577085	405.295695	0.906621	224.699447
11	Ridge	TotalGHGEmissions(MetricTonsCO2e)	Log(y)	(alpha, 1.0, Log(y))	0.511793	409.478241	0.847702	286.961400

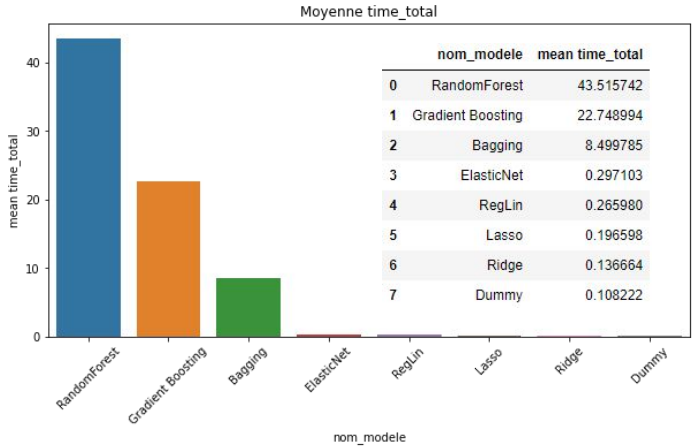
Pour les deux targets la méthode de Gradient Boosting donne les meilleurs résultats sur le jeu de test, sans transformation par le log des targets.

9. c. Modélisation - Sélection des meilleurs modèles

Du côté de la performance matérielle, on s'intéresse aux temps d'exécution de nos modèles.



	nom_modele	mean param_time
0	Gradient Boosting	1031.982655
1	RandomForest	263.034015
2	Bagging	36.666953
3	Lasso	6.430226
4	ElasticNet	5.429184
5	Ridge	1.732875
6	Dummy	NaN
7	RegLin	NaN



	nom_modele	mean time_total
0	RandomForest	43.515742
1	Gradient Boosting	22.748994
2	Bagging	8.499785
3	ElasticNet	0.297103
4	RegLin	0.265980
5	Lasso	0.196598
6	Ridge	0.136664
7	Dummy	0.108222

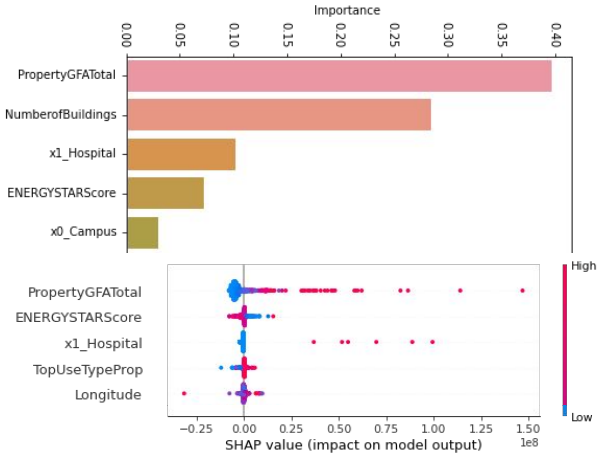
On remarque que les méthodes ensemblistes sont plus consommatrices en ressources, par contre elle sont beaucoup plus précises.

Dans notre cas, sachant que l'on effectue des prédictions annuelles, le temps d'exécution, et d'optimisation des paramètres, importe peu, on va préférer prendre du temps pour avoir des modèles précis, plutôt que de favoriser un temps d'exécution faible et une précision moindre.

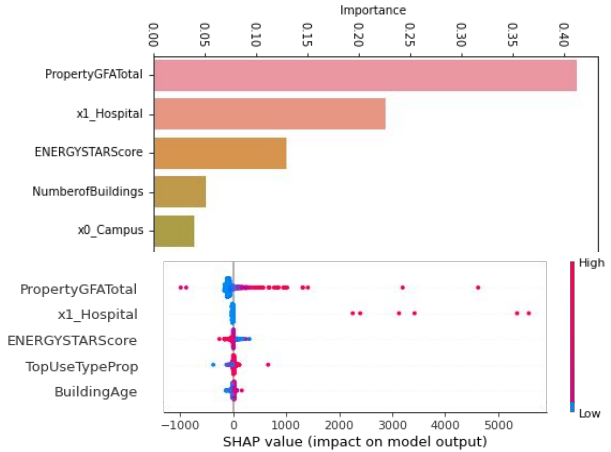
10. Modélisation - Importance des features

Maintenant que l'on à sélectionner nos modèles on va regarder quelles sont les features qui ont de l'importance dans ces modèles.

SiteEnergyUse(kBtu)



TotalGHGEmissions(MetricTonsCO2e)



On retrouve que les 5 features les plus importantes de nos modèles sont les mêmes, mais dans des proportions différentes. La surface totale reste la feature qui pèse le plus dans les deux modèles.

L'ENERGYSTARScore apparaît comme être une feature importante dans nos modèles on va donc s'intéresser à sa pertinence.

11. Pertinence de l'ENERGYSTARScore

Pour vérifier sa pertinence on va entraîner nos 2 modèles de GradientBoosting, avec les mêmes paramètres mais en retirant de nos features l'ENERGYSTARScore :

Avec : 0.929548 0.952193

Sans :

```
model.score(X_test_2, y_test['SiteEnergyUse(kBtu)'])
```

```
0.9504408451114349
```

```
model.score(X_test_2, y_test['TotalGHGEmissions(MetricTonsCO2e)'])
```

```
0.9657118542022046
```

On remarque qu'en retirant l'ENERGYSTARScore nos score R2 de nos 2 modèles ne baissent pas, ils augmentent même. (+0.02 & +0.013)

On peut donc en déduire que même si c'est une feature jugée comme importante dans les précédents modèles, en la retirant nos performances ne sont pas impactées négativement. On peut donc s'en passer pour la suite.

12. Conclusion & Pistes d'amélioration

On peut donc conclure qu'on obtient déjà de très bonnes performances. Malgré tout on pourrait explorer quelques pistes pour améliorer nos modélisations :

- Explorer d'autres pistes grâce au feature engineering :

  a. Créer d'autres Features (à partir du type de la voie par exemple)
  b. Réduire le nombre de modalités de certaines variables (manuellement ou en utilisant le package optbinning)

- Optimiser les modèles sur plus d'hyper paramètres : Comme on a pu voir la recherche d'hyper paramètres optimaux à l'aide du GridSearchCV prend énormément de temps, surtout sur des modèles ensemblistes et on a à pas pu tester de grandes plages de valeur et de combinaisons différentes.

- Dans le futur, on pourrait continuer à réaliser des relevés sur un nombre réduit de bâtiments pour continuer à étoffer notre base de données