Projet 4 : Anticipez les besoins en consommation de bâtiments

Problématique

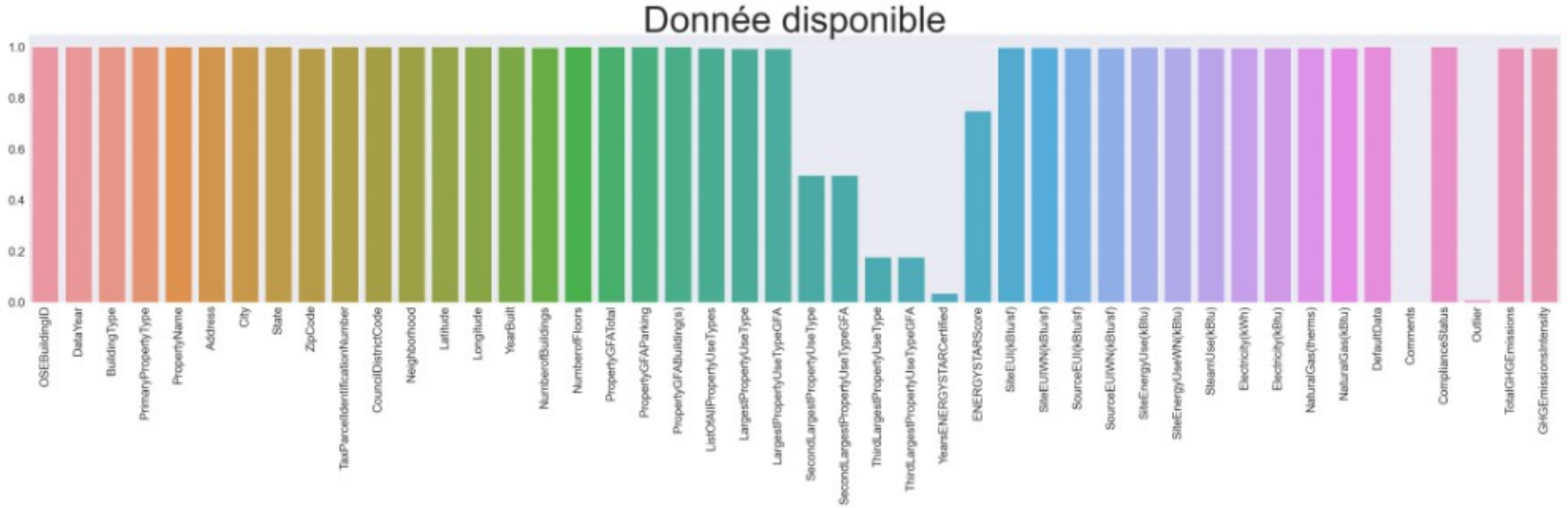
Problématique : Prédire les émissions de CO_2 et la consommation totale d'énergie de bâtiments non destinés à l'habitation

Évaluer l'intérêt de l' "ENERGY STAR Score" pour la prédiction d'émissions

Données des Bâtiments à Seattle

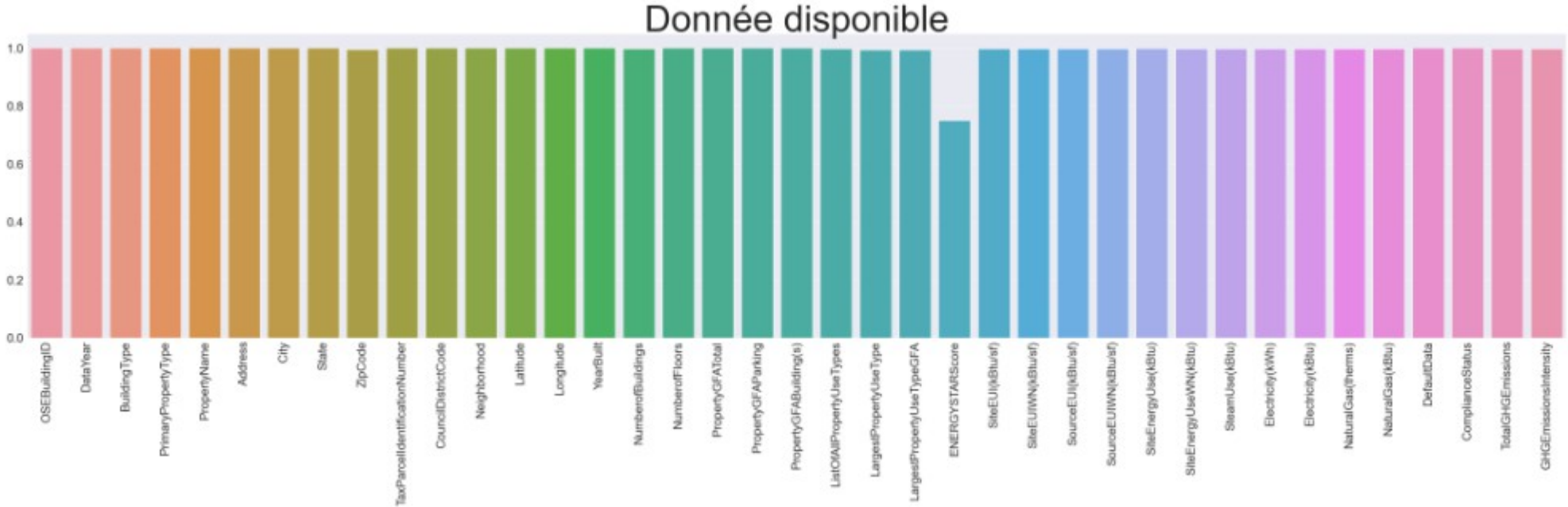
- Donnée collecté par la ville sur la consommation en énergie des bâtiments résidentiel et non résidentiel
- 3376 lignes sur 46 colonnes
- Contient des informations telles que le GFA (Gross Floor Area), la position géographique, la consommation d'énergie des bâtiments, le type de bâtiments…
- Certaines colonnes ne sont presque pas renseignées
- L'Énergie Star Score n'est pas renseignée pour tous les bâtiments

Données disponibles



Préparation des données

- On commence par se débarrasser des colonnes qui ne sont pas remplis à au moins 50 %



Création de la distance Haversine

- On a la latitude et la longitude de chaque bâtiment, mais cette information n’est pas très utile en l’état.
- Avec la longitude et la latitude, on calcule la distance harversine entre le centre de Seattle et le bâtiment.

Sélection des colonnes et des lignes

- On se débarrasse des colonnes qui ne sont pas pertinentes pour répondre à la problématique
- Par exemple, l'adresse, le nom de la propriété, l'ID…
- On se débarrasse également des colonnes qui sont redondantes.
- Par exemple, la même information dans une unité différente pour les colonnes numériques.
- A l'aide de la colonne BuildingType, on se débarrasse des lignes contenant des bâtiments résidentiels.

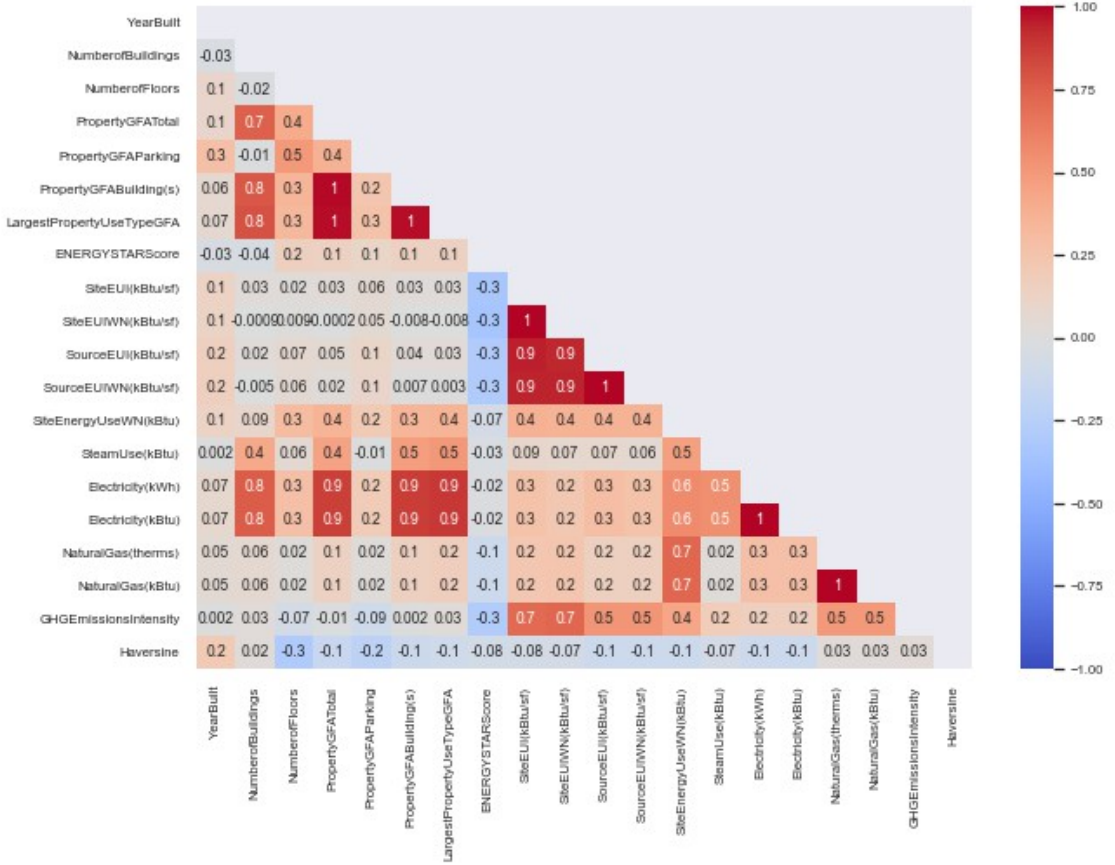
Colonnes catégoriel

- On rassemble certaines catégories dans les colonnes catégorielles pour éviter de se retrouver avec un trop grand nombre de catégories.
- Par exemple : K-12 School et University ont été regroupé dans la catégorie School.
- On transforme les colonnes ‘numériques’ qui correspondent à des catégories identifiées par des nombres.

Corrélation

- A l’aide d’un diagramme de corrélation, on se débarrasse des colonnes numériques qui sont trop corrélées.
- On se débarrasse des colonnes qui ont plus de 70 % de corrélations (on en garde une et on supprime les autres)

Corrélation

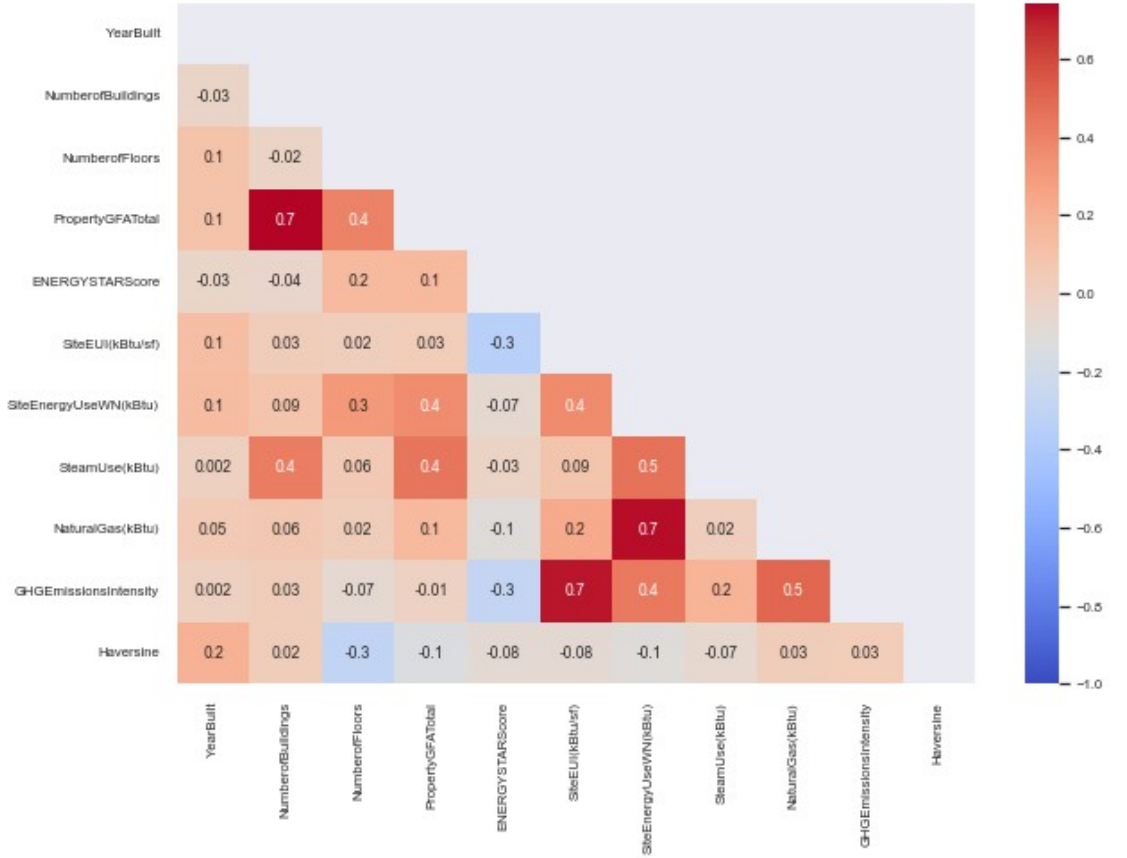


Corrélation

Par exemple :

- Electricity(kwH) et Electricity(kBtu) sont la même information dans différentes unités (on remarque d'ailleurs une corrélation de 1)
- PropretyGFABuilding(s) et NumberofBuildings ont une corrélation de 0,8. C'est trop élevés. On va se débarrasser de l'un d'entre eux.

Corrélation



One Hot Encoding et sélection des features

- On fait le One Hot Encoding des colonnes catégoriel
- On sélectionne les colonnes qui sont suffisamment corrélées avec nos deux targets (SiteEnergyUse(kBtu) et TotalGHGEmissions) tout en évitant celles qui sont trop corrélés (entre 0.05 et 0.82)
- On supprime également les colonnes de la même nature que la target.

Modélisation

- On va chercher un modèle de machine learning qui nous permettra de créer un modèle capable de faire une prédire nos targets (SiteEnergyUse(kBtu) et TotalGHGEmissions) à partir de certaines informations que nous avons sélectionné.
- On divisera notre dataframe en un set d'apprentissage, et un set de test. On voudra éviter que notre modèle overfit, c'est-à-dire qu'il soit trop performant avec le set d'entraînement, par rapport à la validation croisée.
- On testera construira le modèle avec et sans l'EnergyStarScore pour évaluer son utilité.

Modélisation

- On entraînera notre modèle avec le log de notre target, afin d’améliorer ses performances.

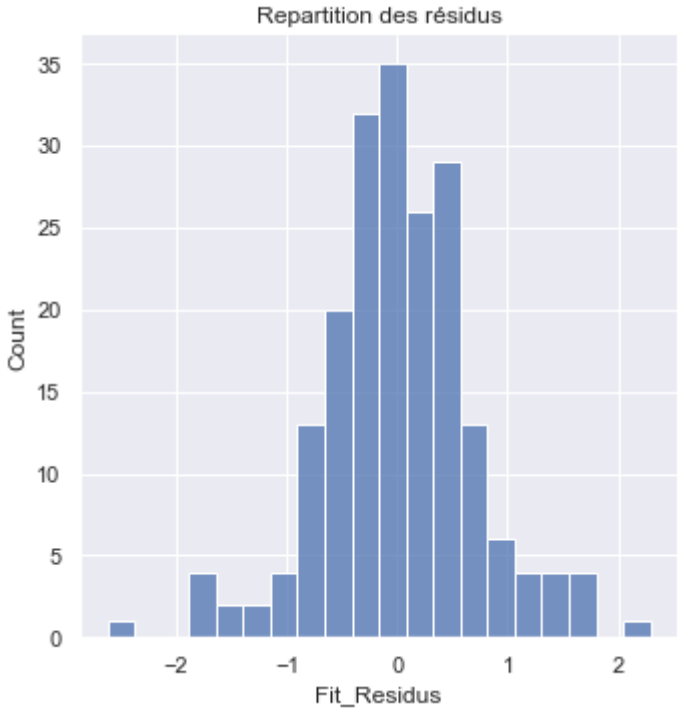
Prédiction du SiteEnergyUse(kBtu)

	model	best_params	r2_score_cv	r2_score_train	r2_score_test	mean_res	std_res	time (s)	commentaire
0	Baseline			0.000000	-0.002380	0.058362	1.196231		
1	LinearRegression()	{}	0.55914	0.582693	0.547749	0.040892	0.803420	1.64776	
2	LinearRegression()	{}	0.511942	0.533006	0.457034	0.007626	0.881424	1.168745	No energyStar
3	Ridge()	{'alpha': 0.001}	0.55914	0.582693	0.547749	0.040892	0.803420	1.571679	
4	Ridge()	{'alpha': 0.001}	0.511942	0.533006	0.457034	0.007627	0.881424	0.944736	No energyStar
5	Lasso()	{'alpha': 0.023535353535353534}	0.562294	0.578363	0.542502	0.038072	0.808217	0.178465	
6	Lasso()	{'alpha': 0.02}	0.513528	0.529794	0.455056	0.007631	0.883029	0.17575	No energyStar
7	SVR()	{'kernel': 'rbf'}	0.66407	0.756774	0.636263	0.019010	0.721204	0.235083	
8	SVR()	{'kernel': 'rbf'}	0.60661	0.679639	0.549604	-0.041988	0.801710	0.25088	No energyStar
9	RandomForestRegressor()	{'max_depth': 3, 'max_features': 'auto', 'min_samples_leaf': 6, 'min_samples_split': 10, 'n_estimators': 50}	0.701219	0.737772	0.668814	0.016234	0.688225	0.871226	
10	RandomForestRegressor()	{'max_depth': 3, 'max_features': 'auto', 'min_samples_leaf': 4, 'min_samples_split': 12, 'n_estimators': 47}	0.674035	0.706732	0.621889	-0.002926	0.735565	0.75797	No energyStar
11	GradientBoostingRegressor()	{'n_estimators': 86}	0.785376	0.888046	0.790105	0.055136	0.545264	1.565155	
12	GradientBoostingRegressor()	{'n_estimators': 52}	0.702849	0.777588	0.650263	0.005903	0.707409	1.478229	No energyStar

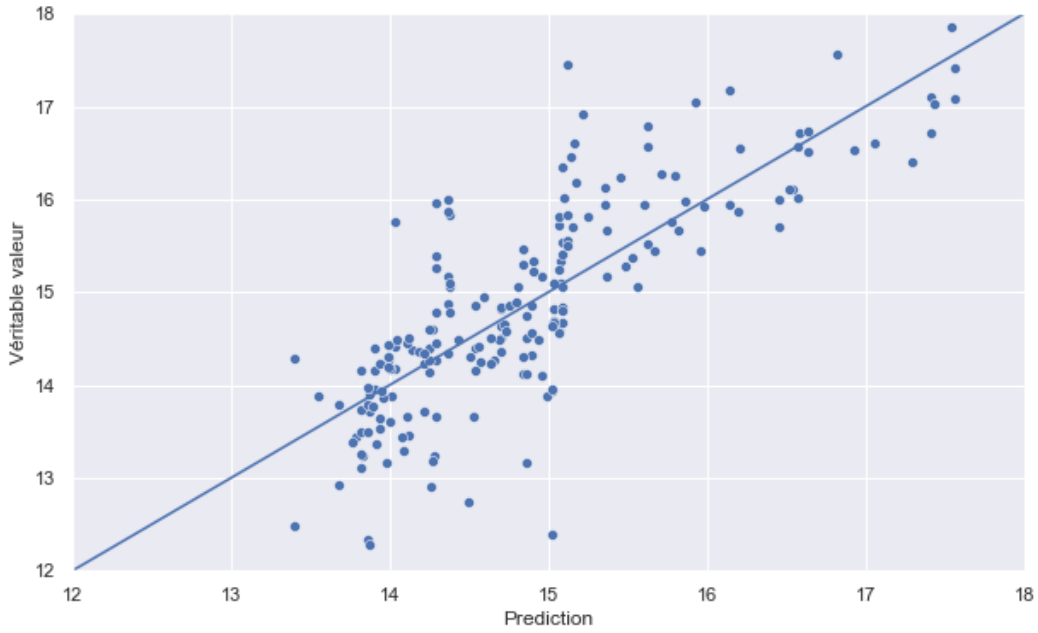
Prédiction du SiteEnergyUse(kBtu)

- Le Gradient Boosting semble être le plus performant, mais il overfit beaucoup trop.
- On choisit le RandomForestRegressor car il a de bonne performance et overfit beaucoup moins.
- On regarde la distribution des résidus et on plot la prédiction contre les valeurs réelles, pour vérifier qu'il n'y a pas de biais.
- On va regarder l'importance de chaque feature également.

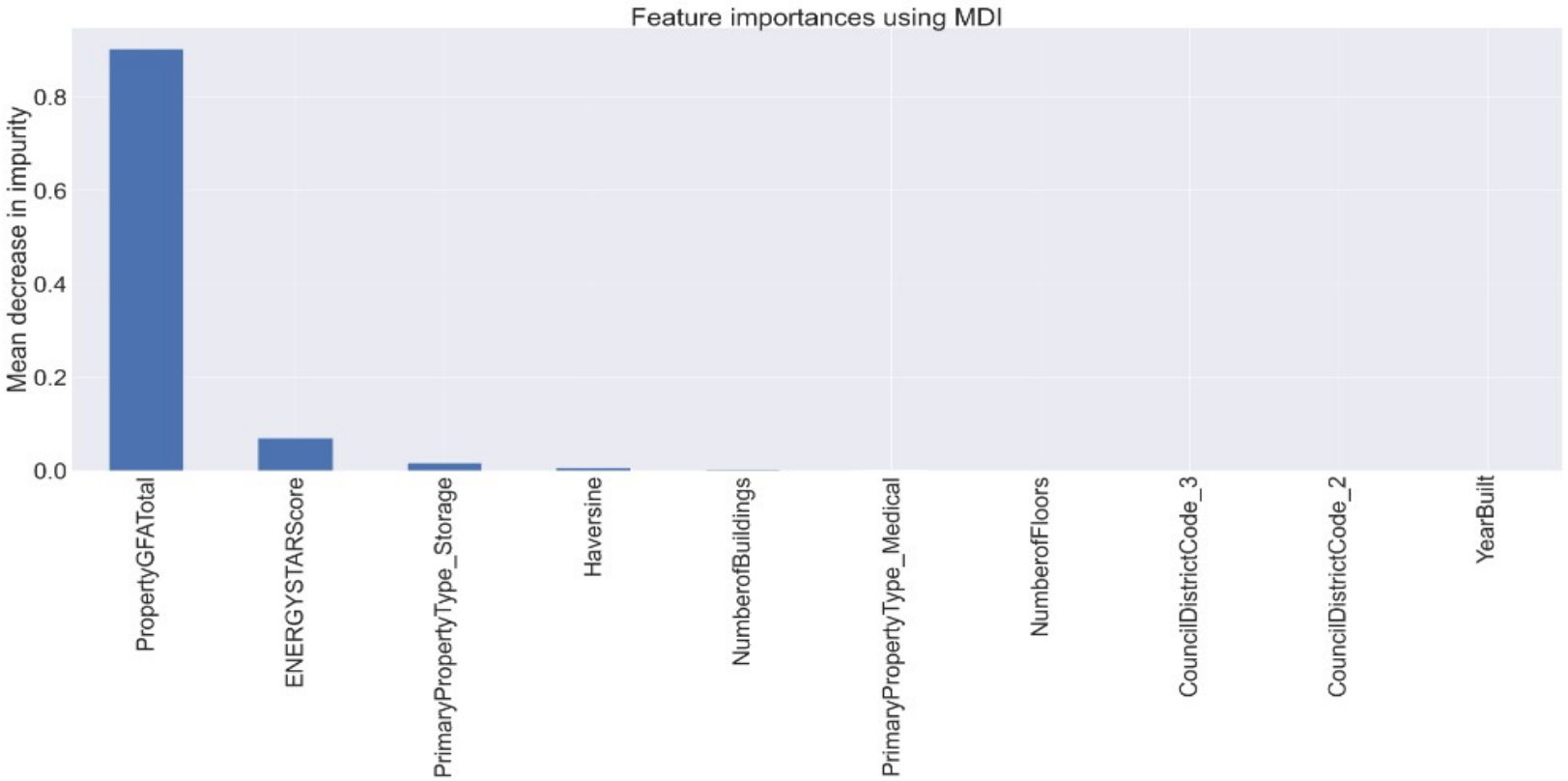
Prédiction du SiteEnergyUse(kBtu)



Moyenne des résidus : 0.01



Feature importance : SiteEnergyUse(kBtu)



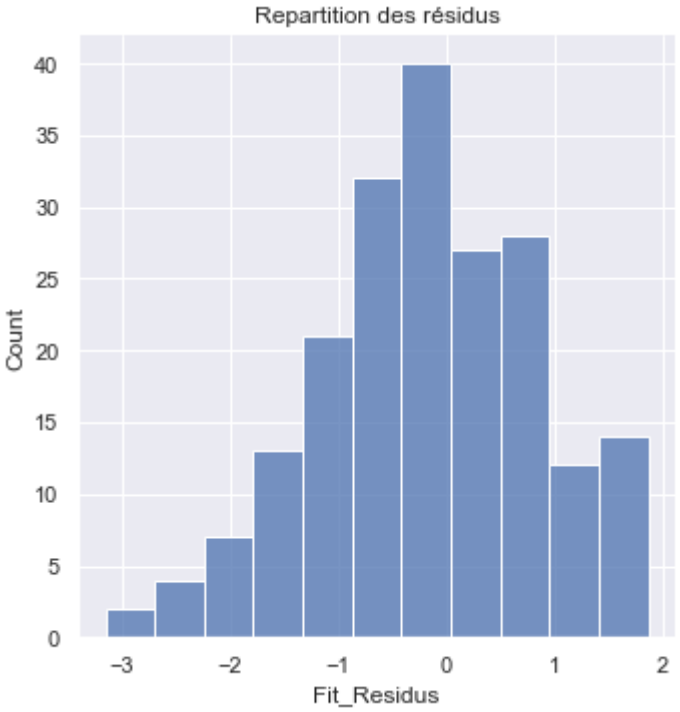
Prédiction du TotalGHGEmissions

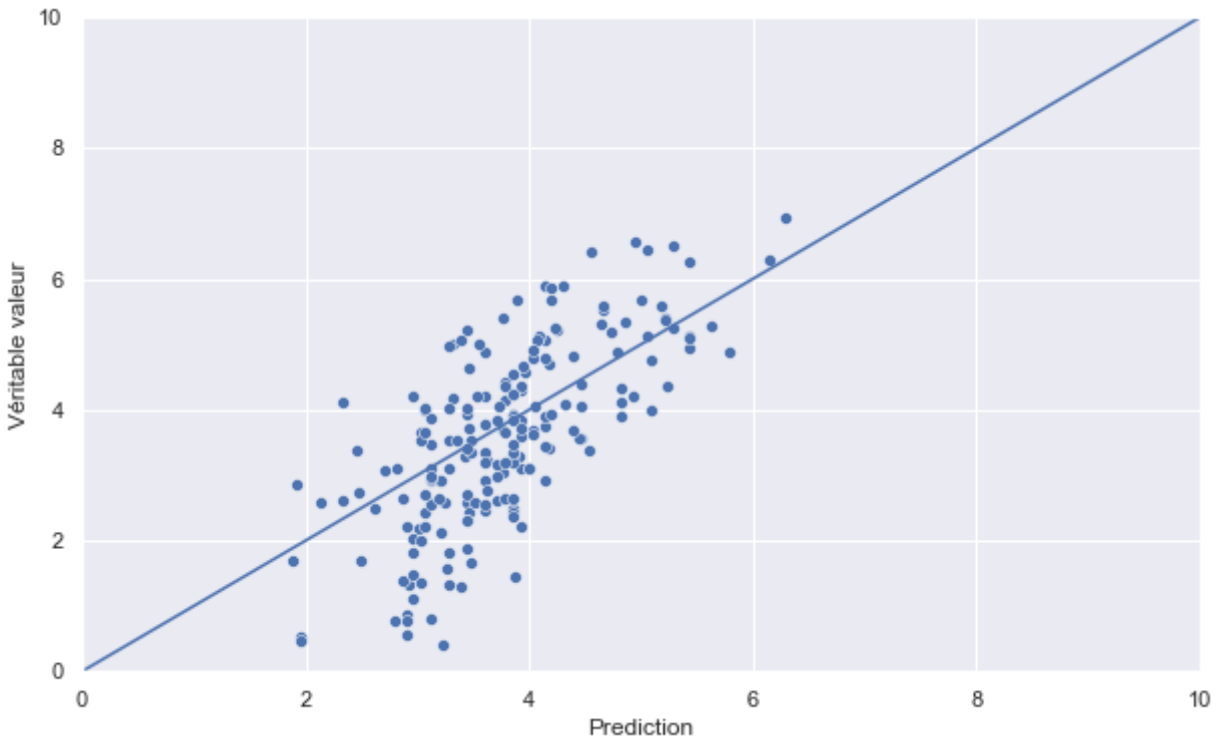
	model	best_params	r2_score_cv	r2_score_train	r2_score_test	mean_res	std_res	time (s)	commentaire
0	Baseline			-2.220446e-16	-0.059685	0.346245	1.417269		
1	LinearRegression()	{}	0.379778	3.997010e-01	0.342383	0.185732	1.134208	1.609277	
2	LinearRegression()	{}	0.349053	3.680387e-01	0.274833	0.174368	1.194237	1.153278	No energyStar
3	Ridge()	{'alpha': 24.569164629827903}	0.382231	3.986010e-01	0.330802	0.197900	1.142375	1.497459	
4	Ridge()	{'alpha': 26.3281546564802}	0.351452	3.669900e-01	0.265386	0.187263	1.200215	0.971655	No energyStar
5	Lasso()	{'alpha': 0.03}	0.385601	3.956489e-01	0.321407	0.204792	1.149399	0.192249	
6	Lasso()	{'alpha': 0.03}	0.354282	3.641034e-01	0.258518	0.195018	1.204719	0.187507	No energyStar
7	SVR()	{'kernel': 'rbf'}	0.426938	5.218869e-01	0.360859	0.276636	1.098764	0.241549	
8	SVR()	{'kernel': 'rbf'}	0.395403	4.730933e-01	0.316681	0.231835	1.148391	0.240184	No energyStar
9	RandomForestRegressor()	{'max_depth': 3, 'max_features': 'auto', 'min_samples_leaf': 3, 'min_samples_split': 6, 'n_estimators': 25}	0.479187	5.177816e-01	0.417548	0.152828	1.070788	0.438773	
10	RandomForestRegressor()	{'max_depth': 3, 'max_features': 'auto', 'min_samples_leaf': 4, 'min_samples_split': 6, 'n_estimators': 20}	0.445103	4.831506e-01	0.398014	0.169672	1.086458	0.346658	No energyStar
11	GradientBoostingRegressor()	{'max_depth': 2, 'max_features': 'auto', 'min_samples_leaf': 4, 'min_samples_split': 6, 'n_estimators': 30}	0.507497	5.577698e-01	0.473657	0.184281	1.011573	0.235997	
12	GradientBoostingRegressor()	{'max_depth': 2, 'max_features': 'auto', 'min_samples_leaf': 3, 'min_samples_split': 5, 'n_estimators': 30}	0.465437	5.076807e-01	0.414031	0.183172	1.069324	0.229073	No energyStar

Prédiction du TotalGHGEmissions

- Cette fois, on choisit le Gradiant Boosting, il possède les meilleures performances et n’overfit pas trop.
- On regarde la distribution des résidus et on plot la prédiction contre les valeurs réelles, pour vérifier qu’il n’y a pas de biais.
- On va regarder l'importance de chaque feature également.

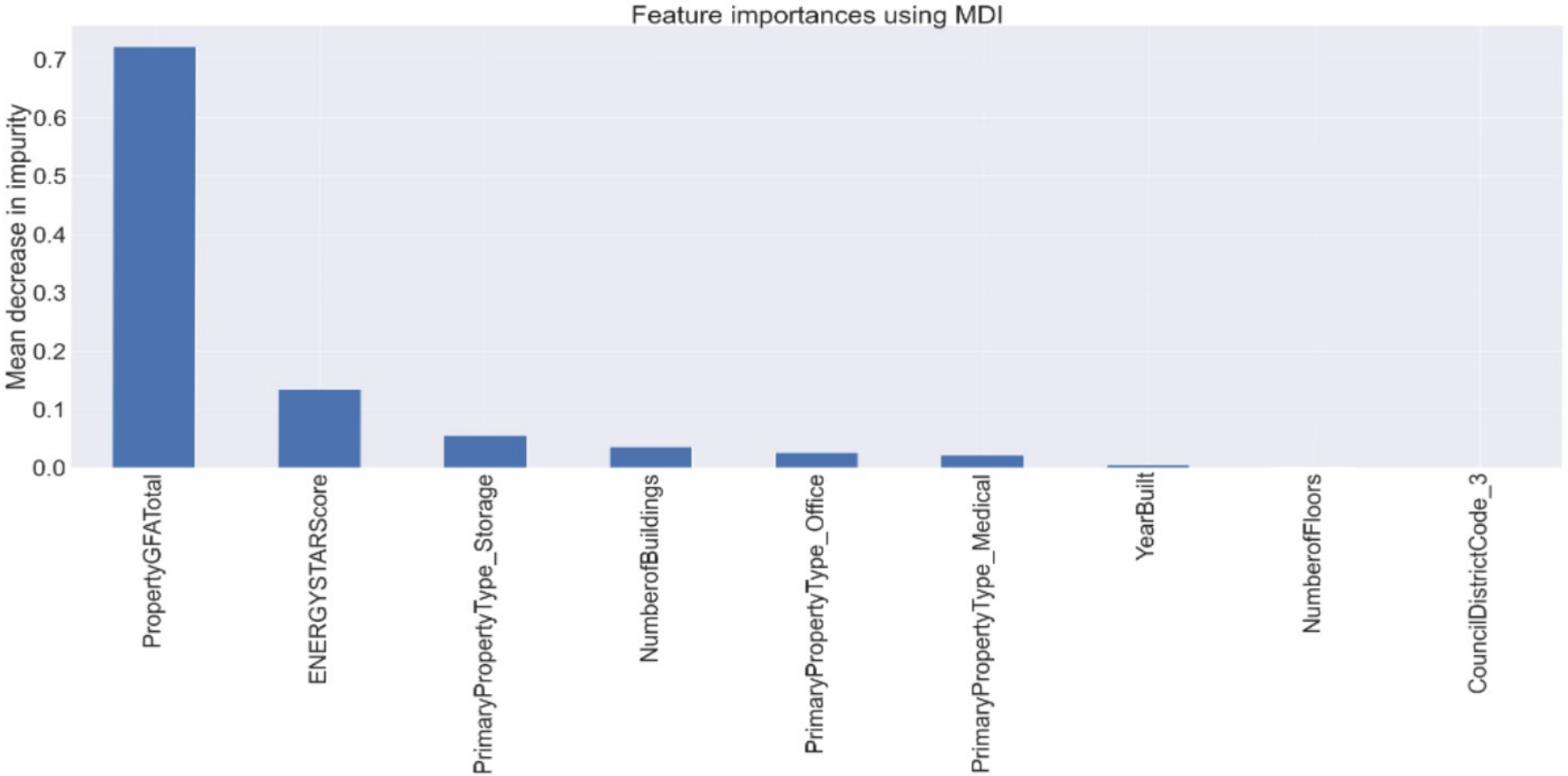
Prédiction du TotalGHGEmissions





Moyenne des résidus : 0.18

Feature importance : TotalGHGEmissions



EnergyStarScore

- Ne pas utiliser l’EnergyStarScore donne un modèle moins performant systématiquement.
- On remarque une amélioration d'à peu prês 4% du score à chaque fois.
- Il est donc utile de continuer à le calculer pour les futures améliorer la précision des prédictions.