4-Anticipez les besoins en consommation de bâtiments

Analyse Exploratoire et prédictions

Sommaire

- Partie 1 : Introduction
- Partie 2 : Nettoyage du jeu de données
- Partie 3 : Exploration des données
- Partie 4 : Predictions
- Partie 5 : Comparaison avec/sans la variable ‘EnergyStar’
- Partie 6 : Conclusion

1-Introduction

- Pour atteindre son objectif d' émission zéro carbone sur les **bâtiments non destinés à l'habitation** en 2050, **la ville de Seattle** s'intéresse à la consommation énergétique des bâtiments actuels (2016).
- **Nous voulons tenter de prédire les émissions de CO2 et la consommation totale d'énergie** de bâtiments **non destinés à l'habitation pour lesquels elles n'ont pas encore été mesurées**.
- Nous chercherons également à **évaluer l'intérêt de l'"ENERGY STAR Score" pour la prédiction d'émissions**
- **Le jeu de données :**
  - https://s3.eu-west-1.amazonaws.com/course.oc-static.com/projects/Data_Scientist_P4/2016_Building_Energy_Benchmarking.csv

2-Nettoyage du jeu de données

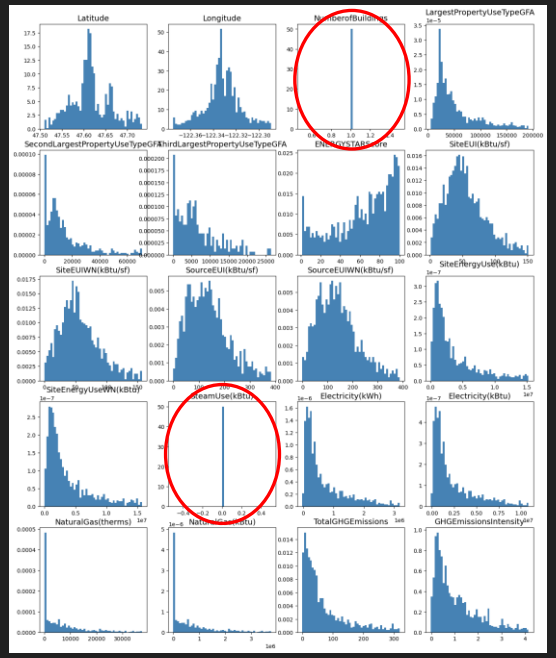
- Suppression des colonnes vides
- Suppression des lignes en doublon:
  - Traitement sur les colonnes ‘Latitude’ et ‘Longitude’ en doublon
- Réduction de la base aux bâtiments non destinés à l’habitation :
  - Filtrage sur la colonne ‘BuildingType’
- Vérification des types des variables:
  - ‘ZipCode’ convertie au format ‘object’
- Suppression des outliers par la méthode interquartile
- Nettoyage des étiquettes:
  - Suppression des colonnes ‘City’, ‘State’, 'NumberofBuildings’ et 'SteamUse(kBtu)' qui ne contiennent qu’une seule valeur
  - Remise en cohérence de la casse sur la colonne ‘Neighborhood’ (une même valeur peut être en majuscules ou en minuscules)
- Filtrage sur les variables compliant dans la colonne ‘Compliant’
- Suppression de la colonne ‘Outlier’:
  - Vide à ce stade

2-Nettoyage du jeu de données

- A ce stade la base contient **42 colonnes pour 1473 valeurs**

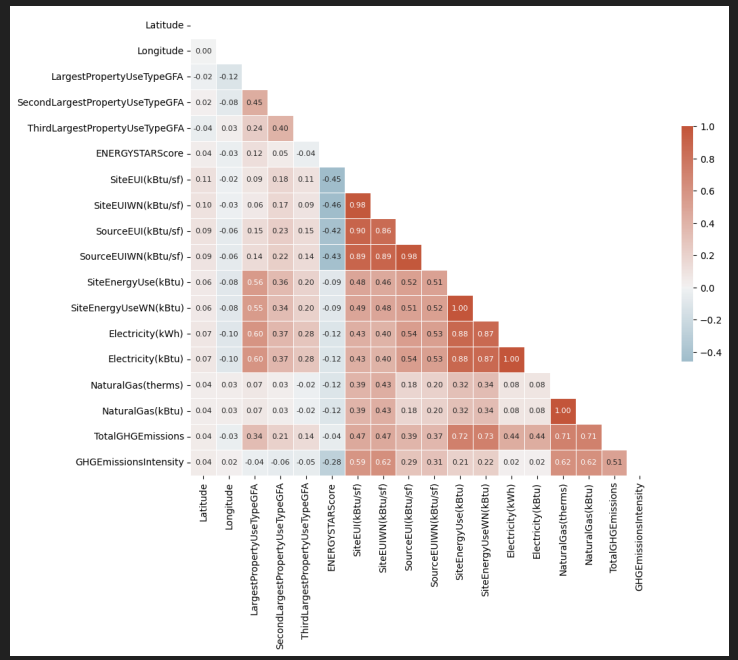
3-1 Analyse univariée et bivariée

- Analyse univariée



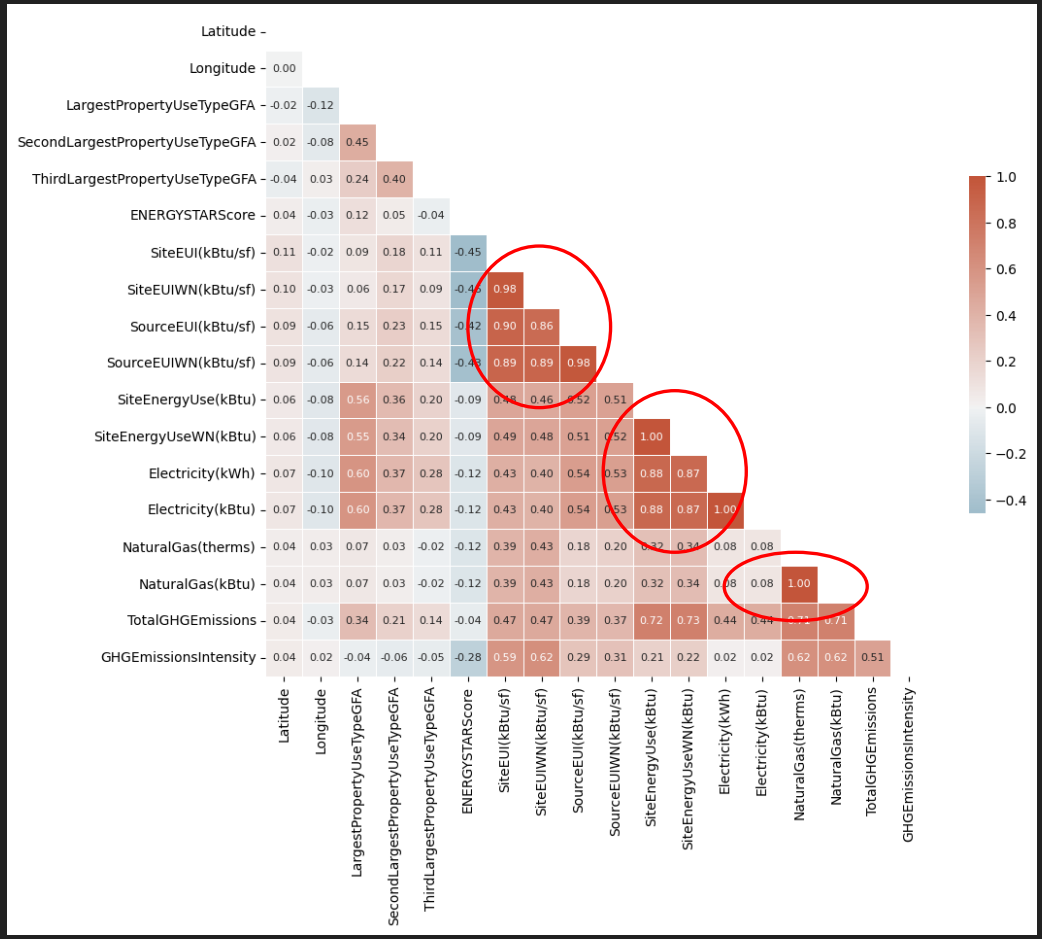
- On supprimes les deux colonnes à valeur unique

- Analyse bivariée



3-2 Feature Engineering- Variables numériques

- Analyse bivariée:
  - Forte colinéarité sur certaines variables



3-2 Feature Engineering - Variables numériques

- On remarque des cpaires de variables identiques mais avec une unité différente (**kBtu vs therms ou kWh**). Pour ces variables on conserve celles qui sont en kBtu et on supprime les autres (**'Electricity(kWh)','NaturalGas(therms)')**
- On remarque qu'on a deux types de variables fortement corrélées: **SiteEnergyUse et SourceEU** qui se déclinent sur 4 modes (**WN, kBtu, sf, I**). On choisit de conserver celles qui sont en WN(kBtu/sf) pour Source EUI et WN(kBtu) pour SiteEU. (**'SiteEUI(kBtu/sf)', 'SiteEUIWN(kBtu/sf)', 'SiteEnergyUse(kBtu)','SourceEUI(kBtu/sf)'**) sont supprimées

Variable 1	Variable 2	Pearson coeff
~~NaturalGas(therms)~~	NaturalGas(kBtu)	1.000000
Electricité(kBtu)	~~Electricity(kWh)~~	1.000000
~~SiteEnergyUse(kBtu)~~	SiteEnergyUseWN(kBtu)	0.996689
SourceEUIWN(kBtu/sf)	~~SourceEUI(kBtu/sf)~~	0.982078
~~SiteEUI(kBtu/sf)~~	~~SiteEUIWN(kBtu/sf)~~	0.980605
~~SourceEUI(kBtu/sf)~~	~~SiteEUI(kBtu/sf)~~	0.901777
SourceEUIWN(kBtu/sf)	~~SiteEUI(kBtu/sf)~~	0.893992
SourceEUIWN(kBtu/sf)	~~SiteEUIWN(kBtu/sf)~~	0.887587
~~SiteEnergyUse(kBtu)~~	~~Electricity(kWh)~~	0.880249
~~SiteEnergyUse(kBtu)~~	Electricity(kBtu)	0.880249
~~Electricity(kWh)~~	~~SiteEnergyUseWN(kBtu)~~	0.866459
Electricity(kBtu)	~~SiteEnergyUseWN(kBtu)~~	0.866459
~~SiteEUIWN(kBtu/sf)~~	~~SourceEUI(kBtu/sf)~~	0.857356
SiteEnergyUseWN(kBtu)	TotalGHGEmissions	0.725630
TotalGHGEmissions	~~SiteEnergyUse(kBtu)~~	0.718244
NaturalGas(kBtu)	TotalGHGEmissions	0.710061
~~NaturalGas(therms)~~	TotalGHGEmissions	0.710061

3-2 Feature Engineering - Variables numériques

- on change toutes les colonnes numériques possibles en **/sf** pour se libérer de la colinéarité avec PropertyGFATotal : **'PropertyGFAParking'**, **'LargestPropertyUseTypeGFA'**, **'SecondLargestPropertyUseTypeGFA'**, **'ThirdLargestPropertyUseTypeGFA'**, **'SiteEnergyUseWN(kBtu)'**, **'Electricity(kBtu)'**, **'NaturalGas(kBtu)'**, **'TotalGHGEmissions'**, **'GHGEmissionsIntensity'**

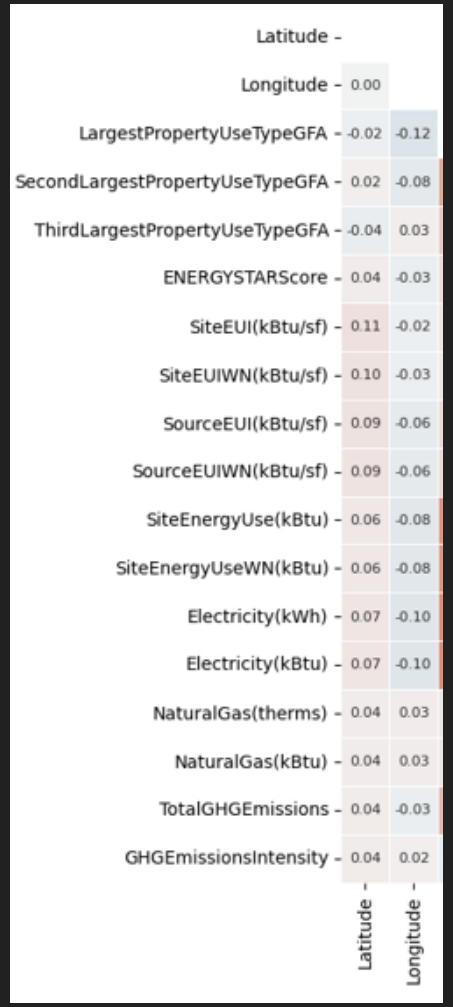
3-3 Feature Engineering - Variables catégorielles

- Pour pouvoir intégrer toutes ces données dans les modèles nous devons convertir les variables catégorielles en variables numériques.
  - Avec un **HotEncoding** chaque valeur des variables catégorielles est transformée en une variable numérique binaire (0-1). On risque d'avoir un nombre de variables trop important (>100) par rapport au nombre de valeurs => risques liés au « fléau de la dimensionalité ».
  - On opte pour un **TargetEncoding** des variables catégorielles qui attribue à chaque valeur une probabilité. Le nombre de dimensions ne change pas.
  - Les variables concernées : **'BuildingType'**, **'PrimaryPropertyType'**, **'Neighborhood'**, **'LargestPropertyUseType'**

3-3 Feature Engineering - synthèse

- A ce stade on a des colonnes peu remplies (dont la colonne ‘EnergyStar’) pour l’analyse. On les supprime.
- On remarque également que **11% des lignes ont une Longitude non renseignée**. C’est une quantité assez importante de lignes dont il serait dommage de se priver pour l’analyse. On impute la **longitude médiane** après avoir vérifié dans l’analyse bivariée qu’il n’y a pas de corrélation avec nos targets.

Extrait de la matrice de corrélation



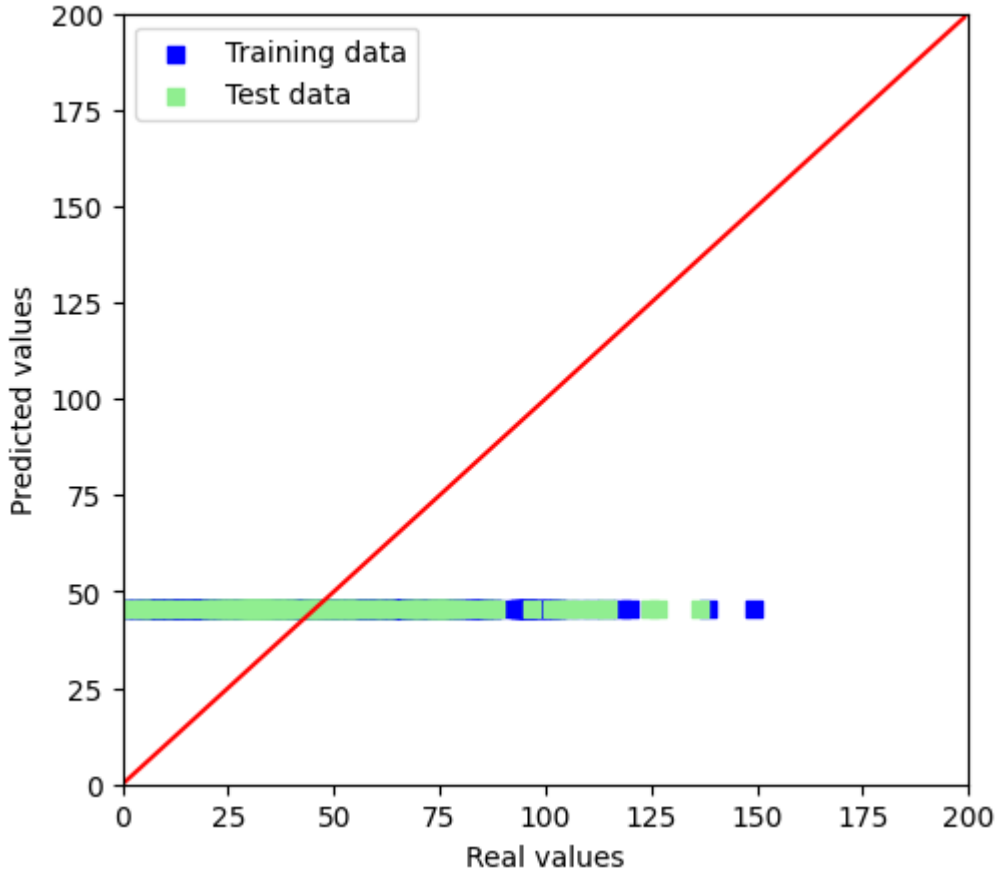
3-3 Feature Engineering - synthèse

- On a un jeu de données avec **20 variables et 1071 valeurs** prêtes pour l'analyse exploratoire.
- Nos targets sont :
  - **'SiteEnergyUseWN(kBtu)/sf'** : la consommation énergétique en kBtu moyennée sur 30 ans et ramenée à la surface
  - **'TotalGHGEmissions/sf'** : l'émission carbone en équivalent tonnes de CO2 ramenée à la surface

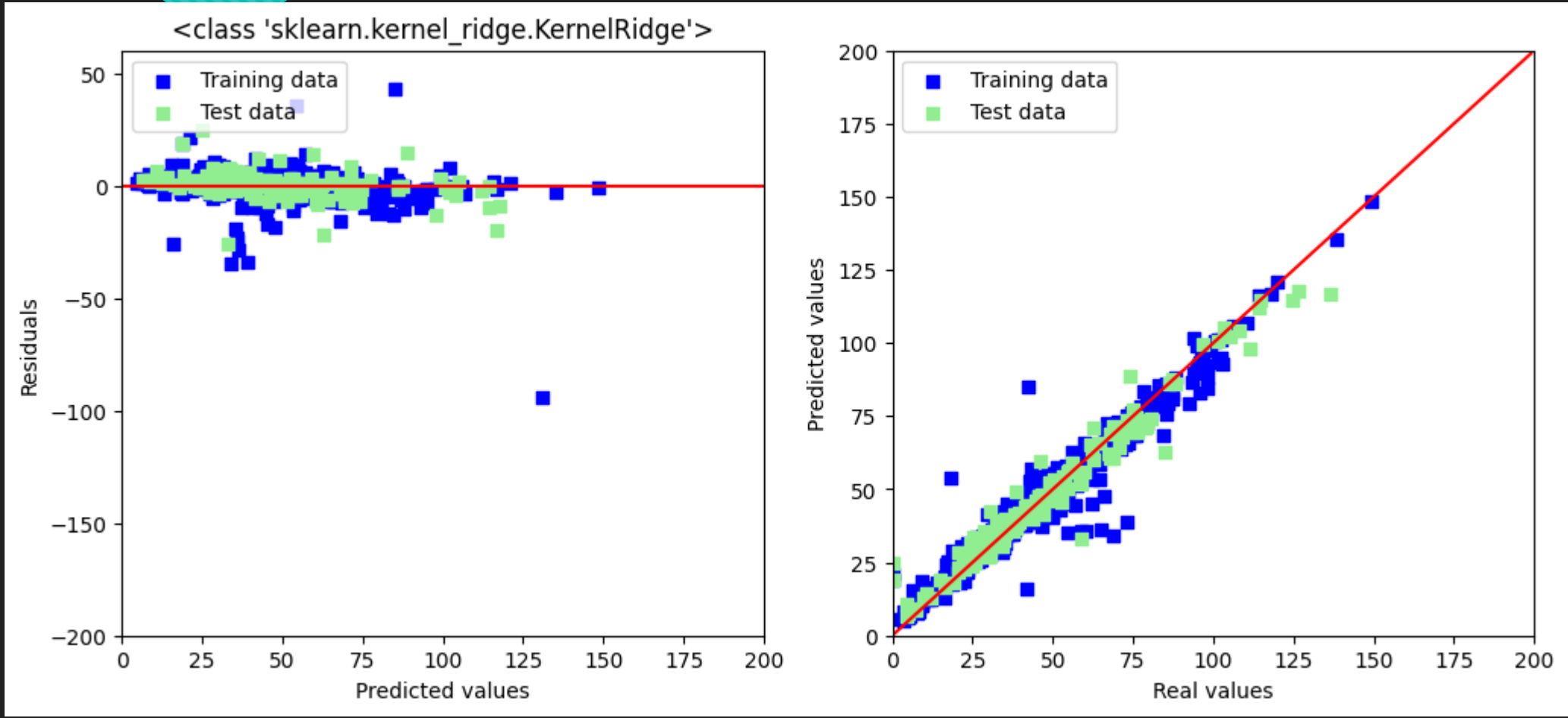
4-1 Prédictions variable 'SiteEnergyUseWN(kBtu)/sf'

- On va effectuer un entraînement sur différents modèles de régression sur un échantillon d'apprentissage (**80% de la base**) et les tester sur un échantillon de test (**20% de la base**):
  - Baseline : approche naïve Mean
  - Régression régularisée Ridge à noyau
  - Régression linéaire Lasso
  - Régression linéaire ElasticNet
  - Régression KNN
  - SVR à noyau
  - XG-boost
  - RandomForest
- On utilise **GridSearchCV(cv=5)** pour entraîner ces modèles
- Pour chaque mise en œuvre de modèle plusieurs types de visualisation seront présentées:
  - Un scatterplot valeur réelle vs valeur prédite
  - Un scatterplot valeur prédite vs résidu (=valeur réelle-valeur prédite)
  - Graphes relatifs aux paramètre du modèle
- Chaque modèle sera évalué à la fin dans un tableau récapitulatif selon son **coefficient de détermination optimal et l'erreur moyenne absolue sur l'échantillon de test** et **le temps de traitement**

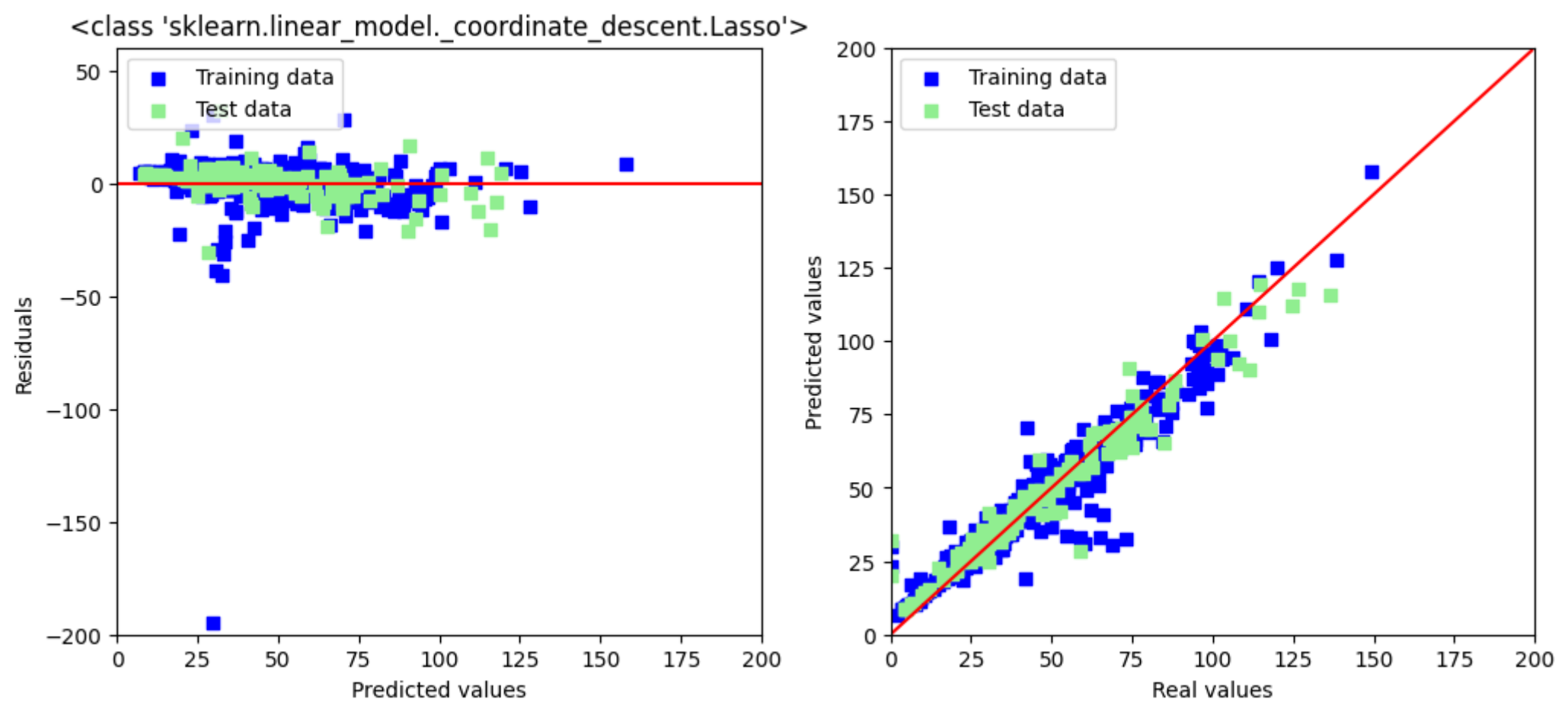
4-1-1 Baseline : Méthode Naive Mean



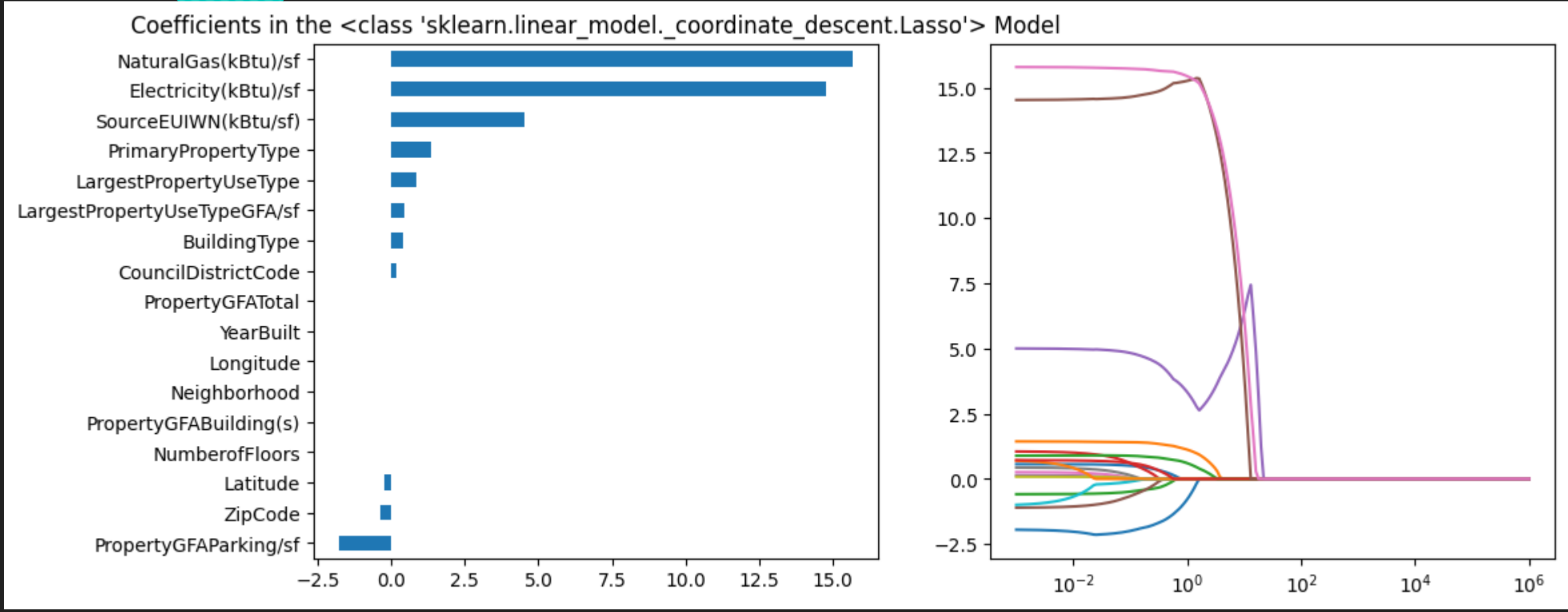
4-1-2 Régularisation KernelRidge



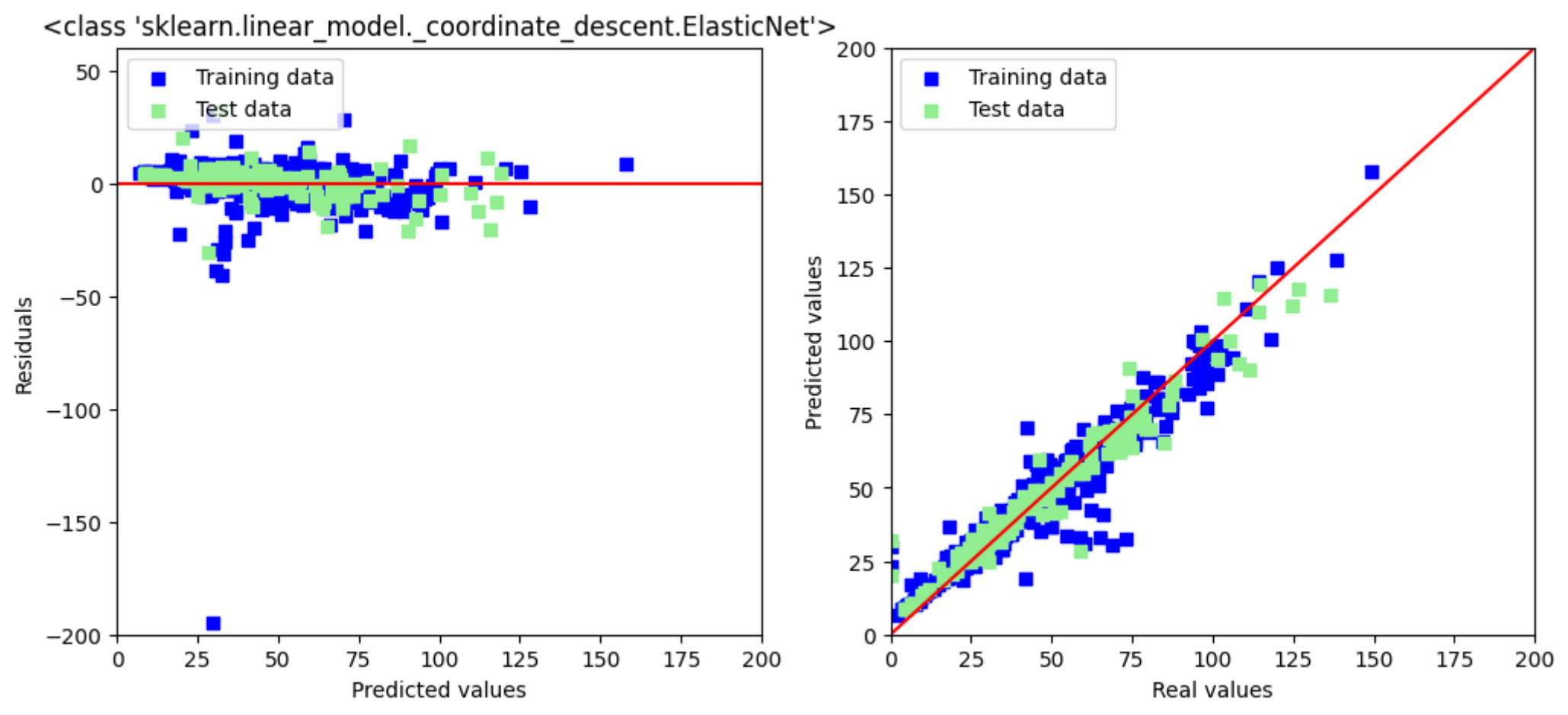
4-1-3 Régularisation Lasso



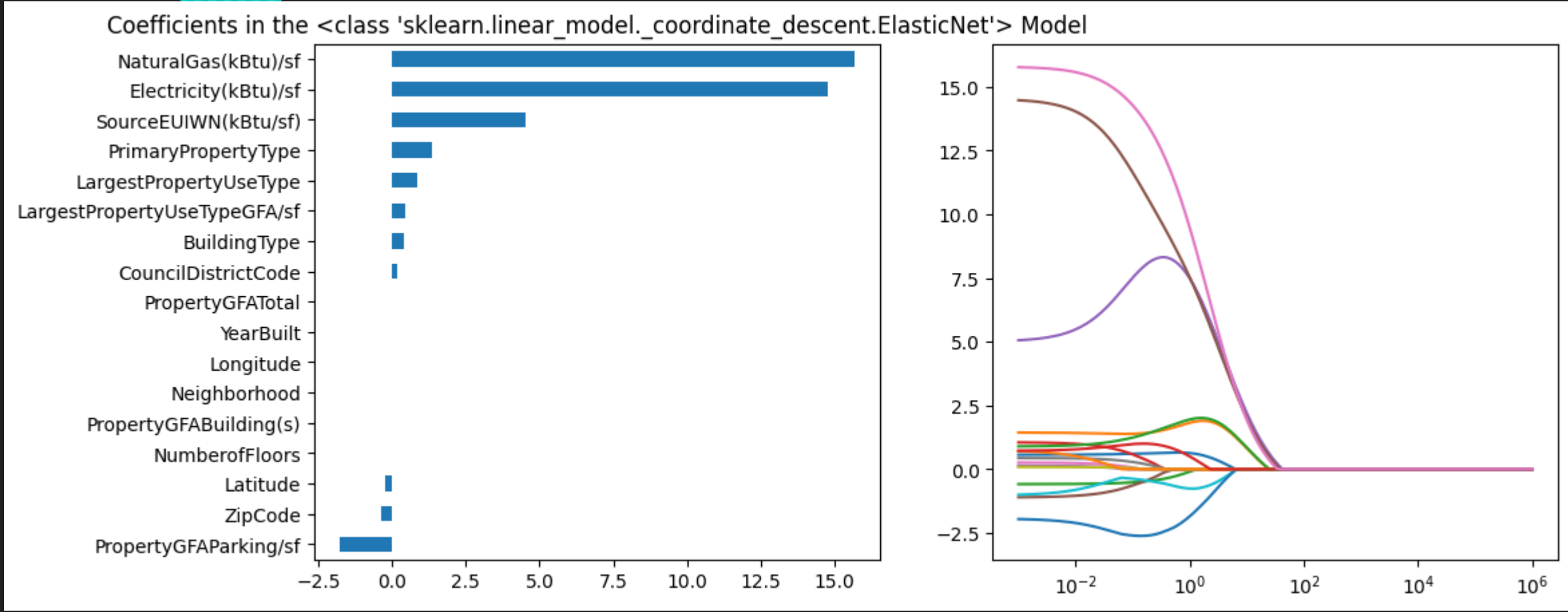
4-1-3 Régularisation Lasso



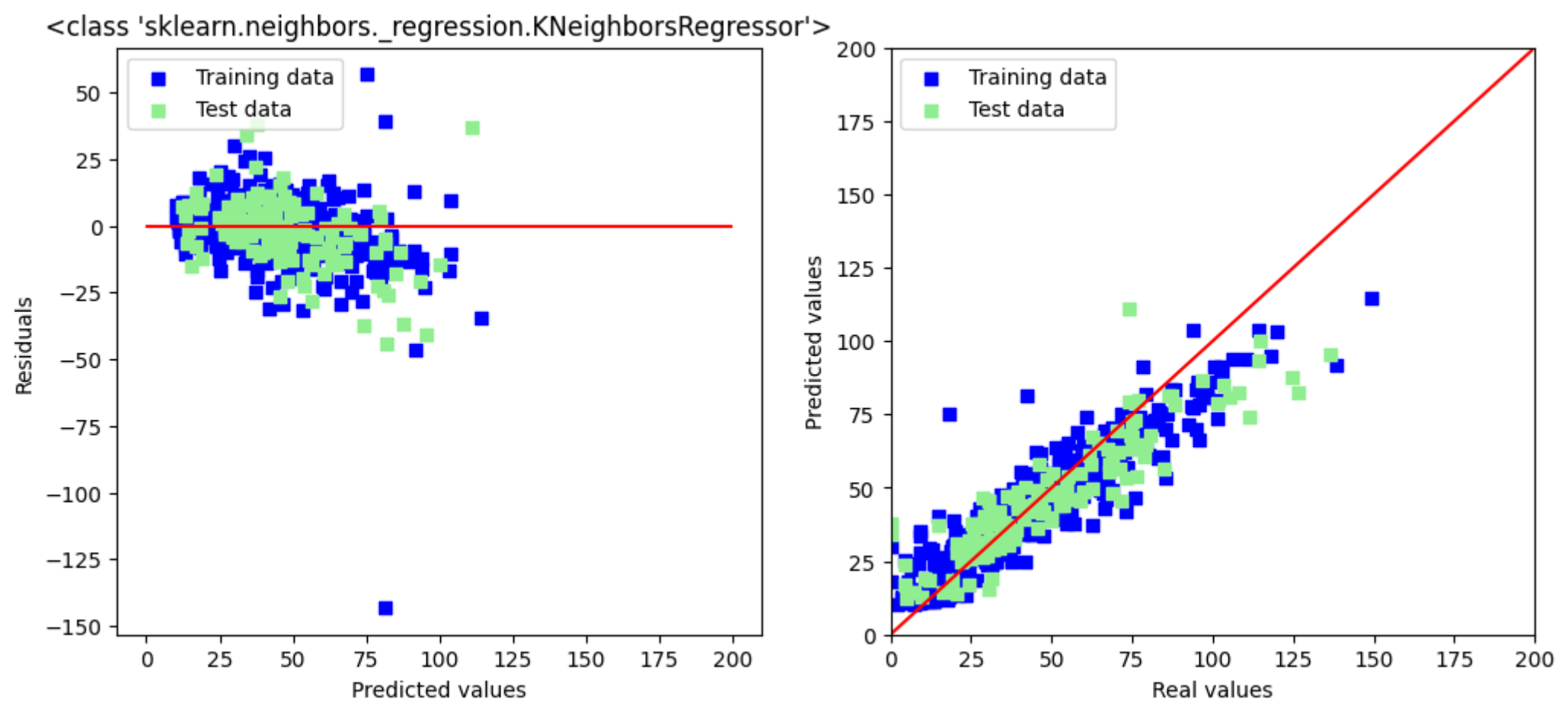
4-1-4 Régularisation ElasticNet



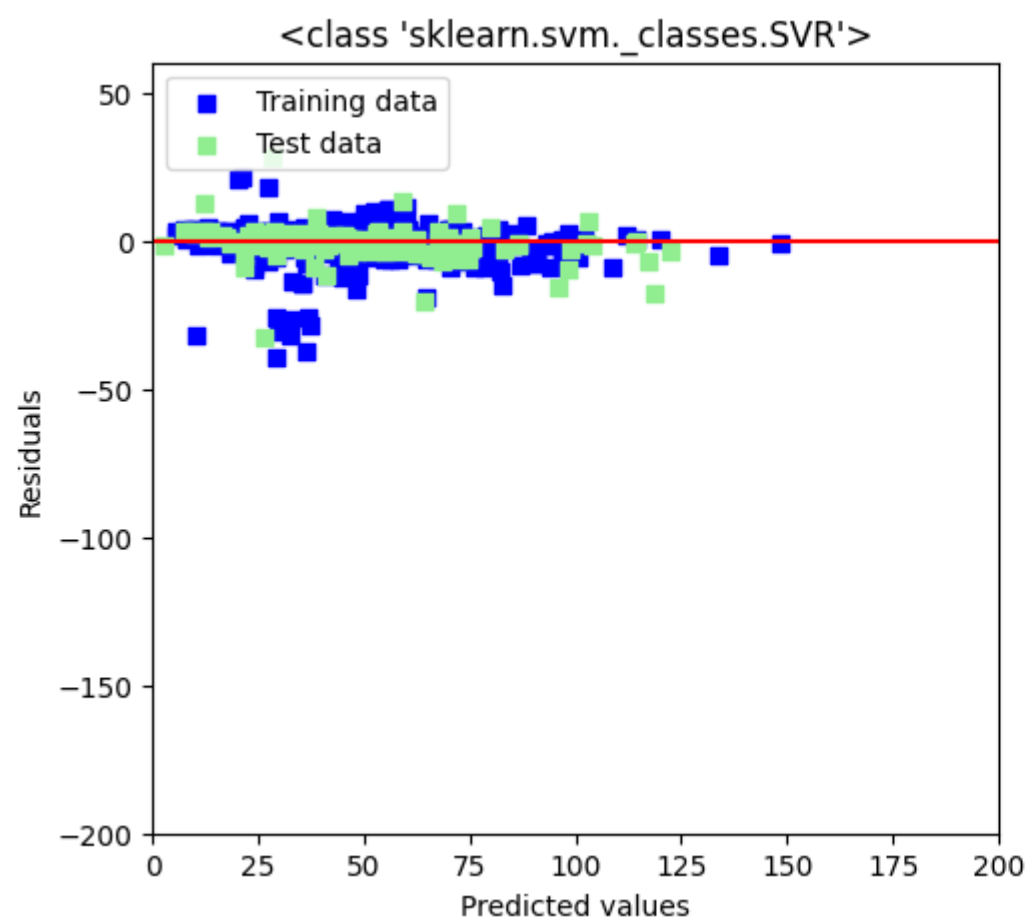
4-1-4 Régularisation ElasticNet

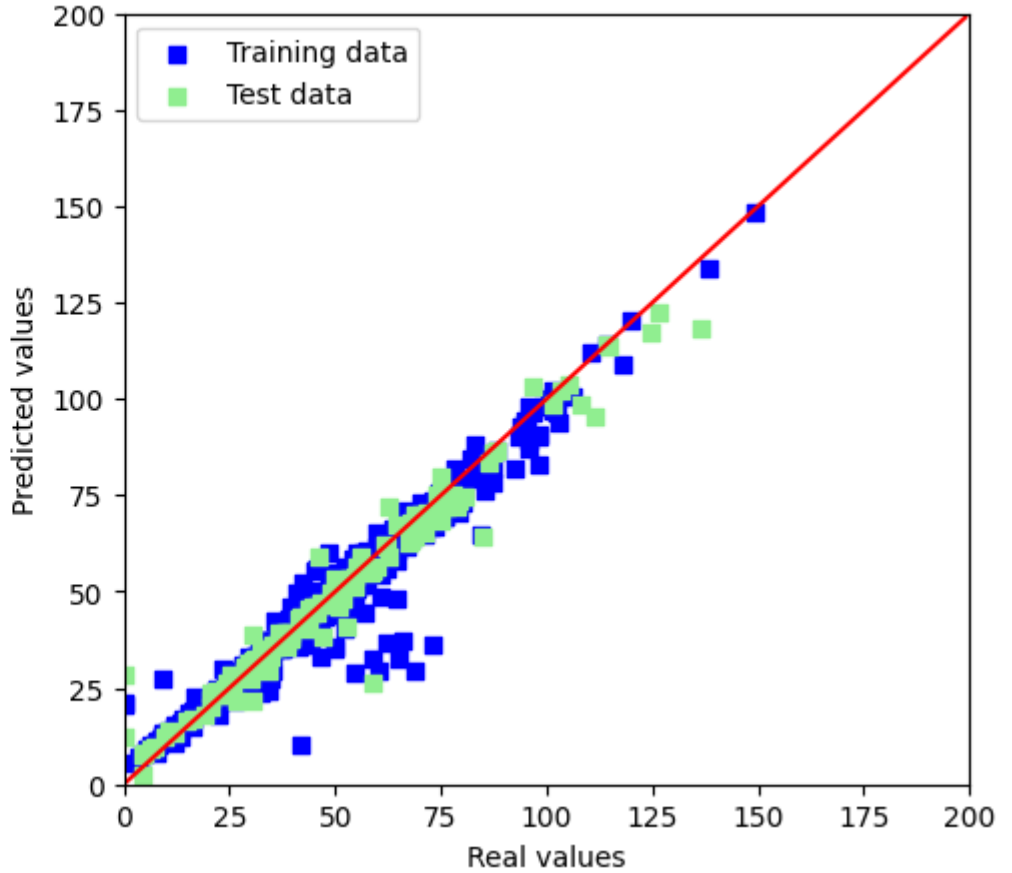


4-1-5 Régression KNN

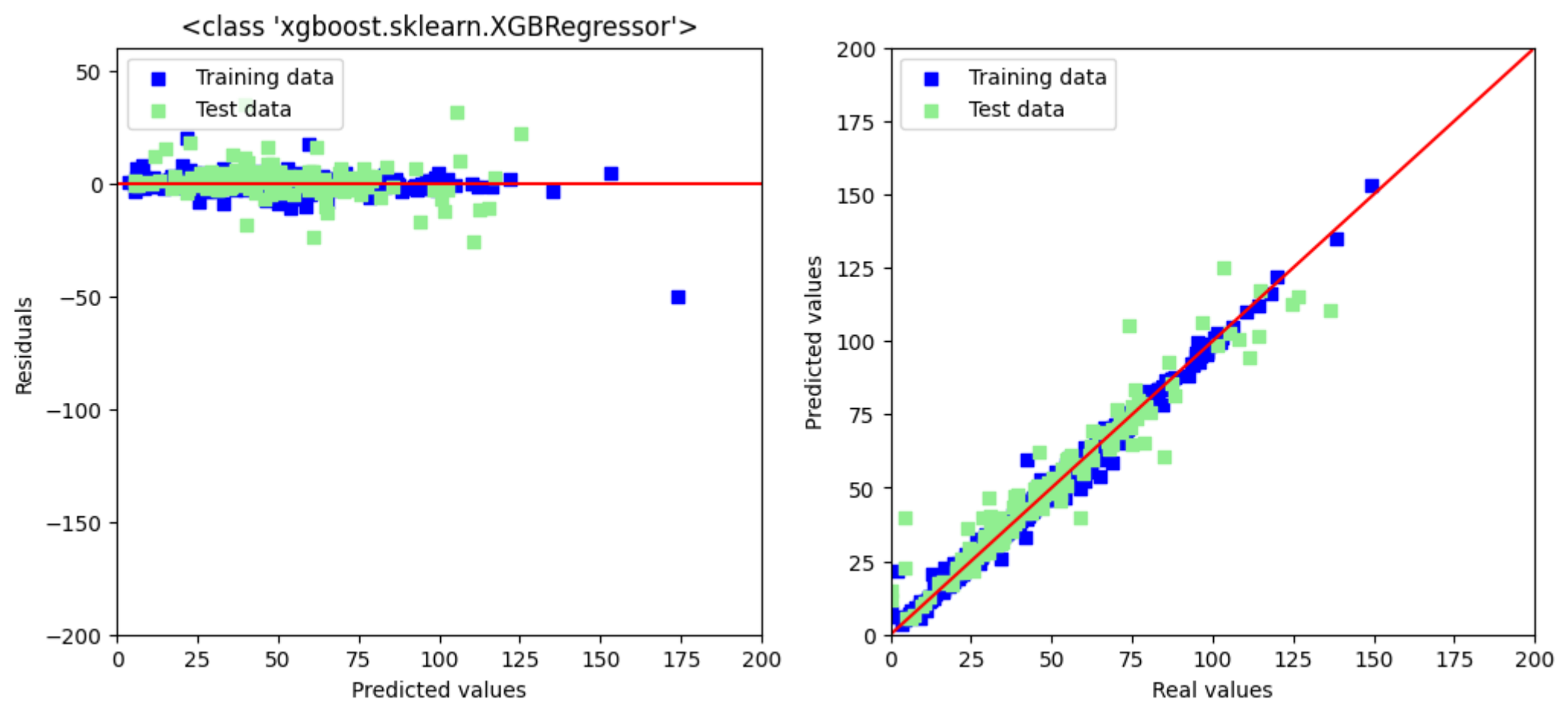


4-1-6 Régression Kernel SVR

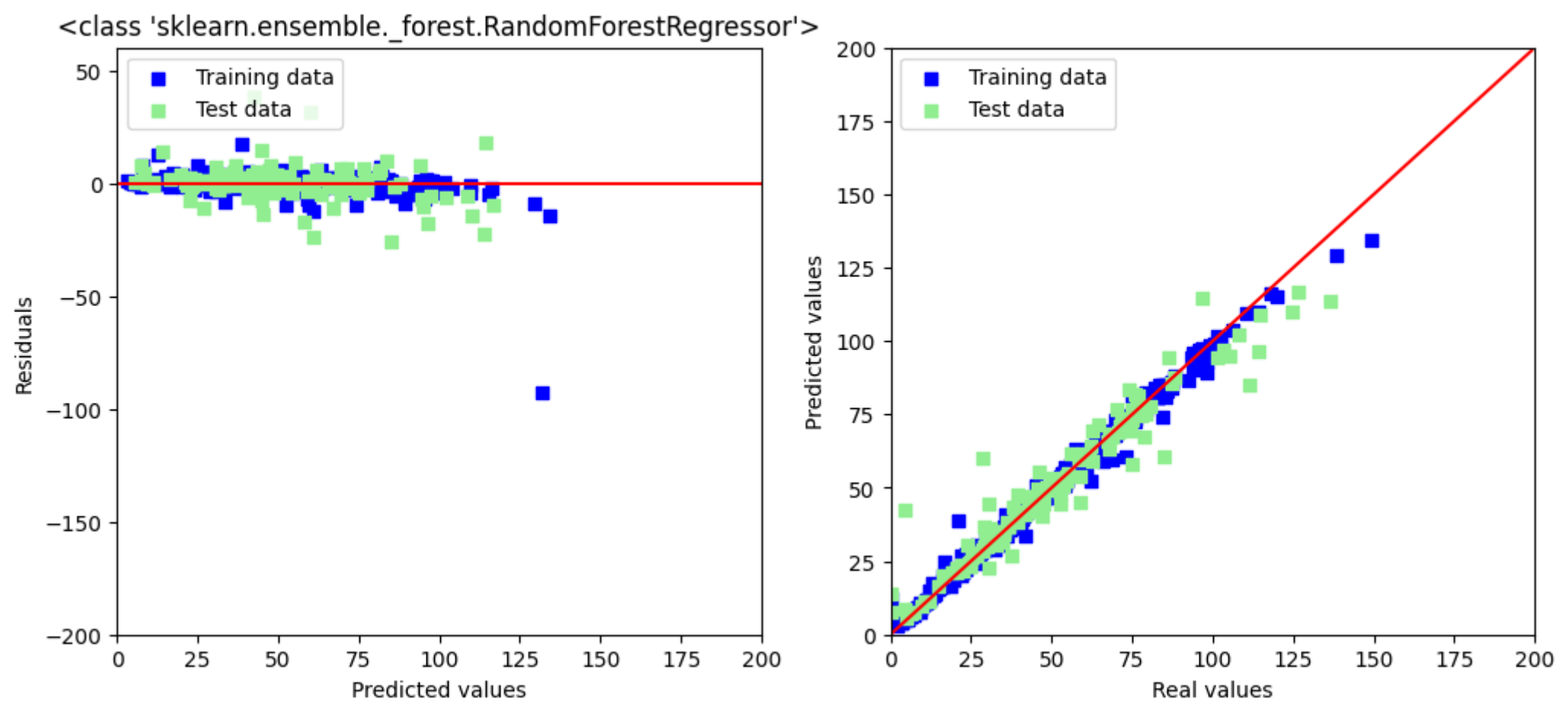




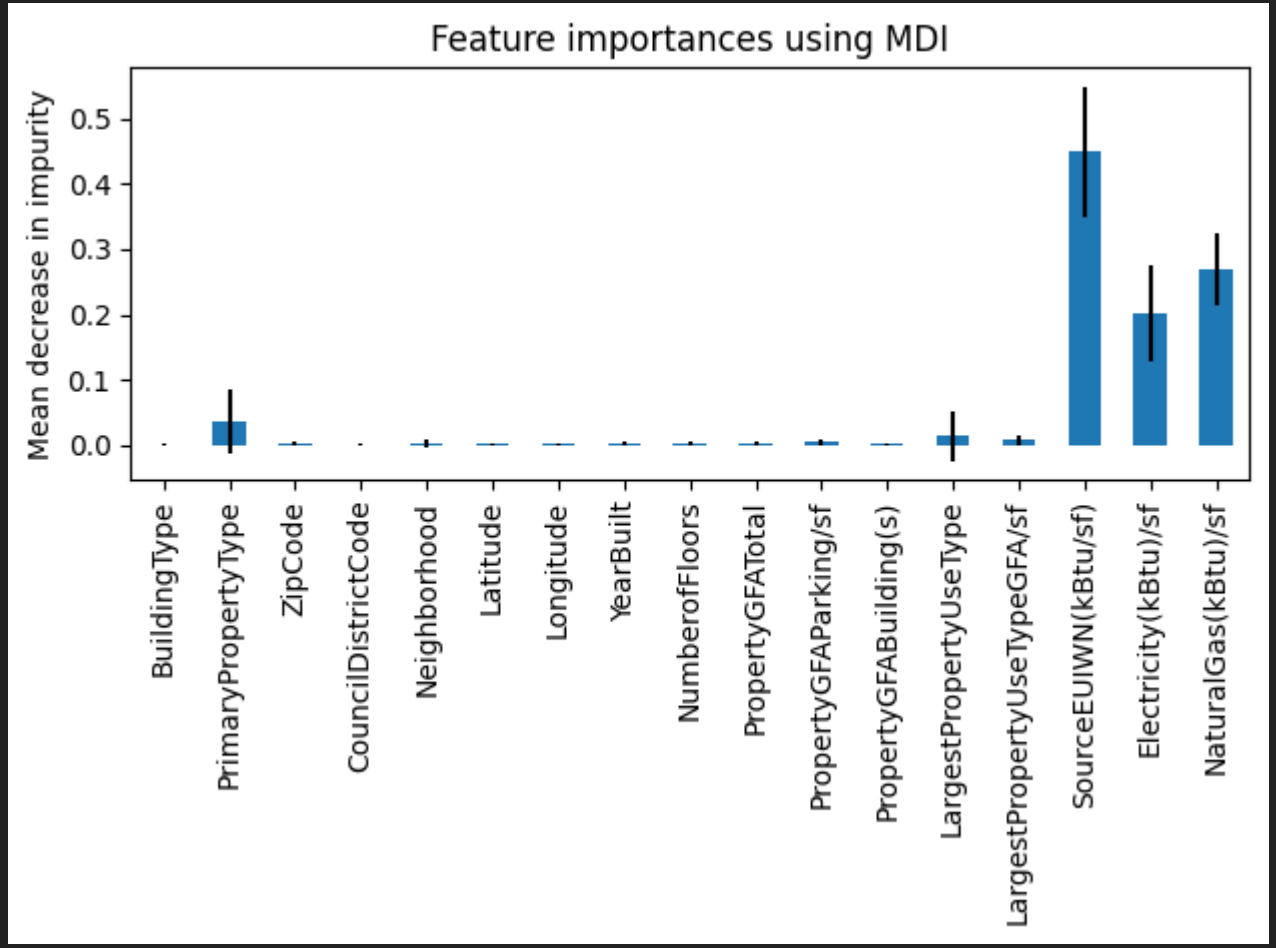
4-1-7 XGboost


<class 'xgboost.sklearn.XGBRegressor'>
Training data
Test data
Residuals
Predicted values
50
0
−50
−100
−150
−200
0
25
50
75
100
125
150
175
200
Training data
Test data
Predicted values
Real values
200
175
150
125
100
75
50
25
0

4-1-8 Random Forest



4-1-8 Random Forest



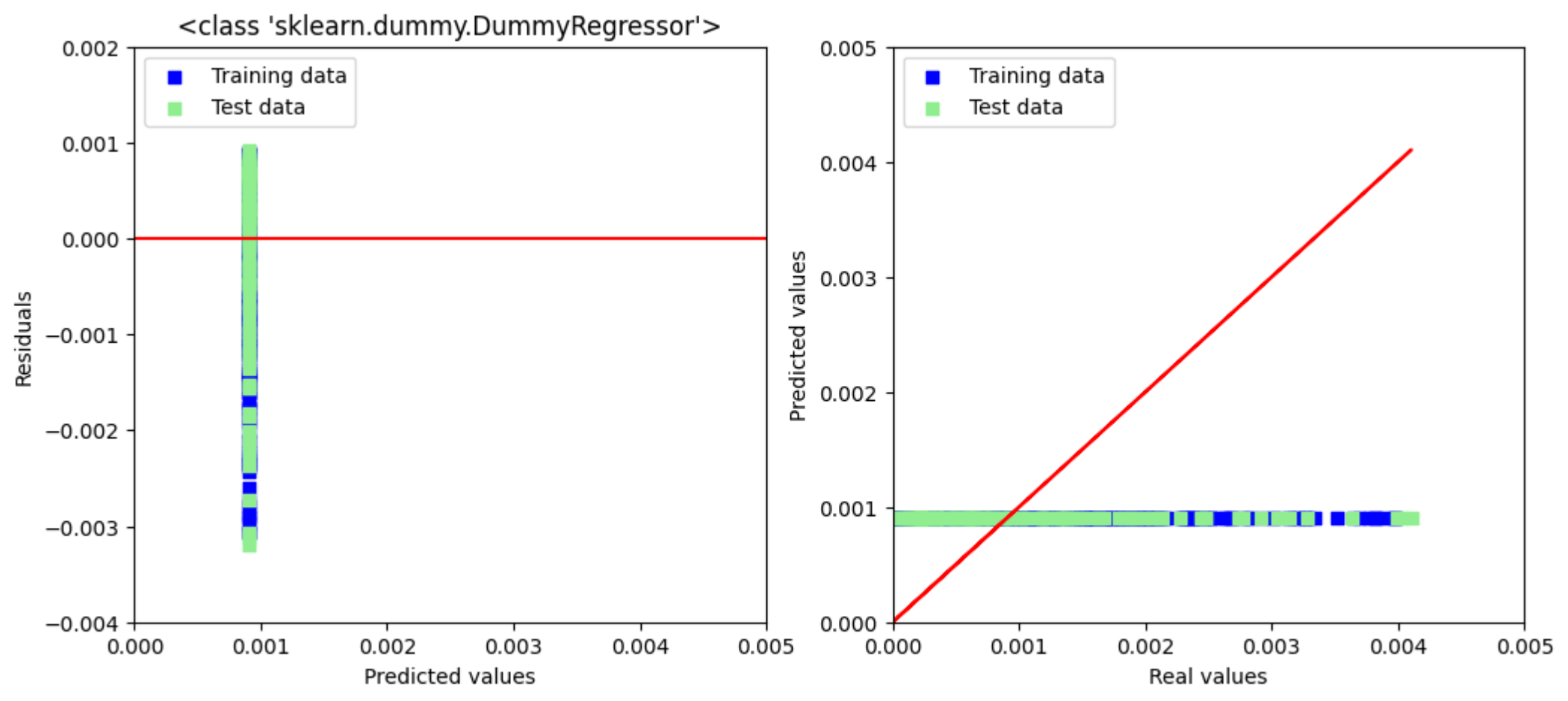
4-1-9 Synthèse modèle prédictif 'SiteEnergyUseWN(kBtu)/sf'

	modèle	meilleurs_paramètres	meilleur_score	MAE	process_time
0	DummyRegressor()	{'strategy': 'mean'}	-0.010691	19.934615	0.015623
1	KernelRidge()	{'alpha': 0.097712415353465 02, 'gamma': 0.01, ...	0.888619	3.497039	407.322750
2	Lasso()	{'alpha': 0.24945081352303167}	0.883485	3.303145	5.080350
3	ElasticNet()	{'alpha': 0.24945081352303167, 'l1_ratio': 1.0}	0.883485	3.303145	75.680071
4	KNeighborsRegressor()	{'n_neighbors': 4}	0.737283	8.829697	7.999009
5	**SVR()**	**{'C': 0.1, 'epsilon': 2, 'kernel': 'linear', '...**	**0.889822**	**2.880840**	**35.202421**
6	XGBRegressor()	{'colsample_bytree': 0.7, 'learning_rate': 0.0...	0.873170	3.447548	33.493420
7	RandomForestRegressor()	{}	0.877248	3.460685	9.556365

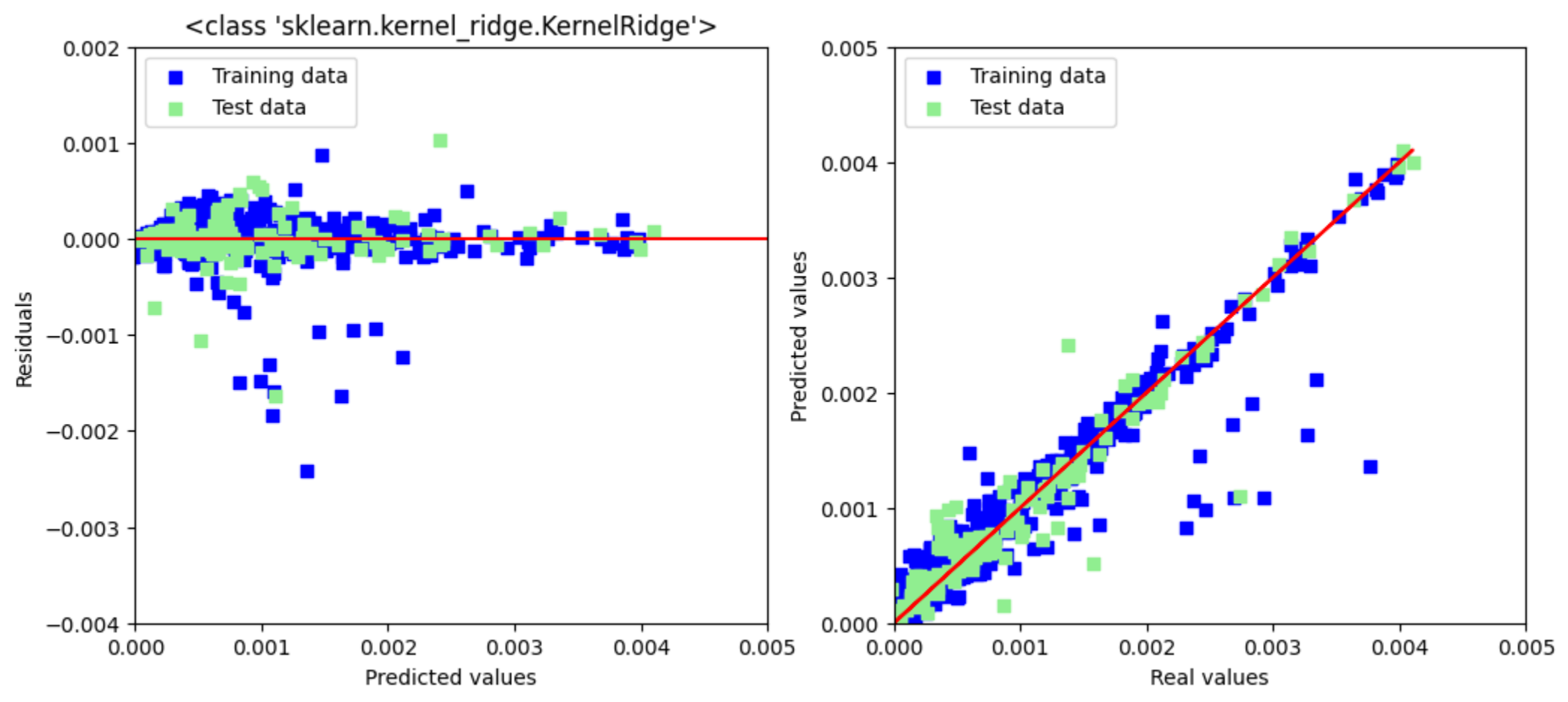
4-2 Exploration variable 'TotalGHGEmissions/sf'

- On va effectuer un entraînement sur différents modèles de régression sur un échantillon d'apprentissage (**80% de la base**) et les tester sur un échantillon de test (**20% de la base**):
  - Baseline : approche naïve Mean
  - Régression régularisée Ridge à noyau
  - Régression linéaire Lasso
  - Régression linéaire ElasticNet
  - Régression KNN
  - SVR à noyau
  - XG-boost
  - RandomForest
- On utilise **GridSearchCV(cv=5)** pour entraîner ces modèles
- Pour chaque mise en œuvre de modèle plusieurs types de visualisation seront présentées:
  - Un scatterplot valeur réelle vs valeur prédite
  - Un scatterplot valeur prédite vs résidu (=valeur réelle-valeur prédite)
  - Graphes relatifs aux paramètre du modèle
- Chaque modèle sera évalué à la fin dans un tableau récapitulatif selon son **coefficient de détermination optimal et l'erreur moyenne absolue sur l'échantillon de test** et **le temps de traitement**

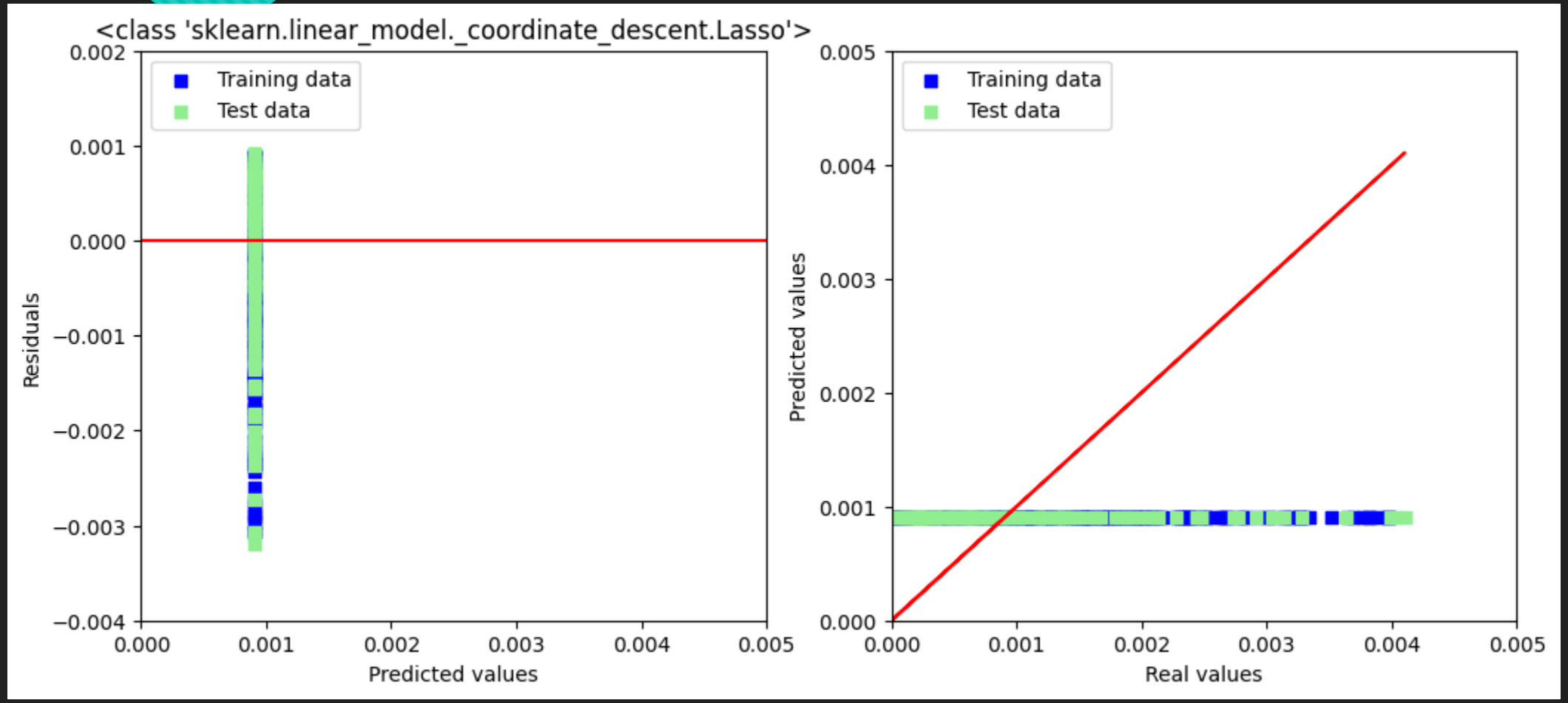
4-2-1 Baseline : Méthode Naive Mean



4-2-2 Régularisation KernelRidge

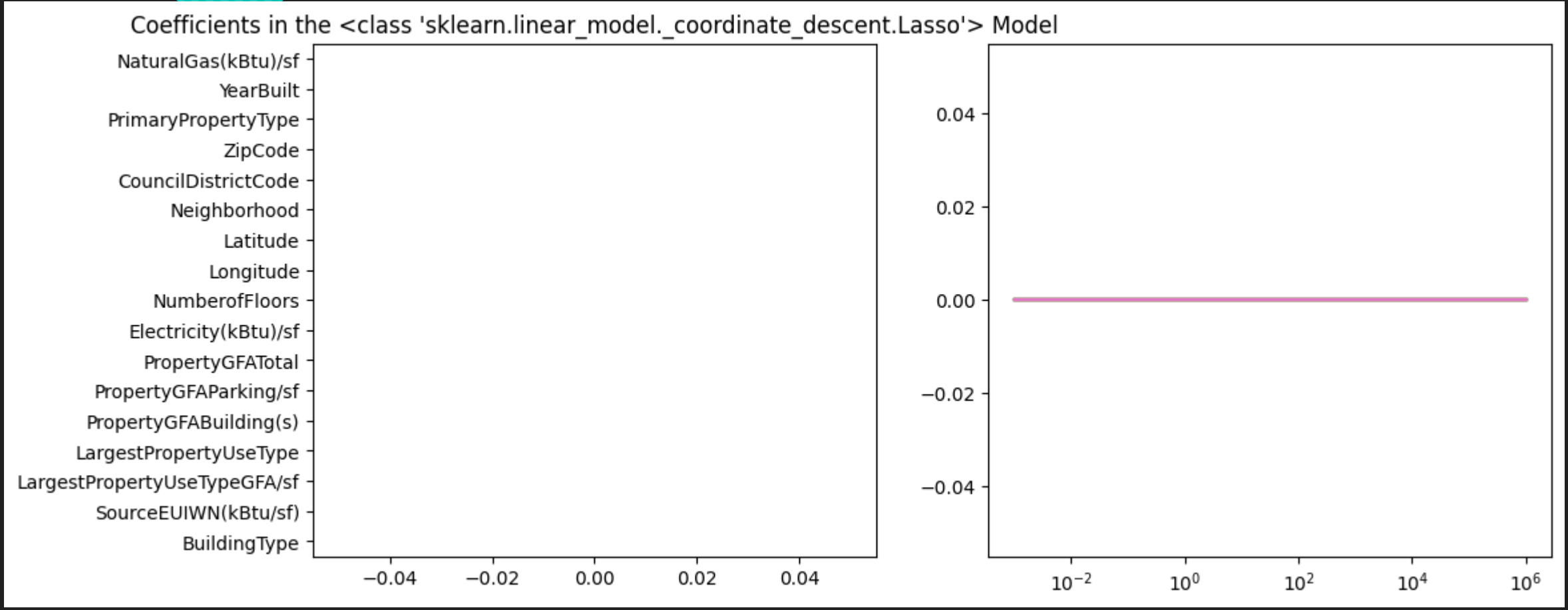


4-2-3 Régularisation Lasso

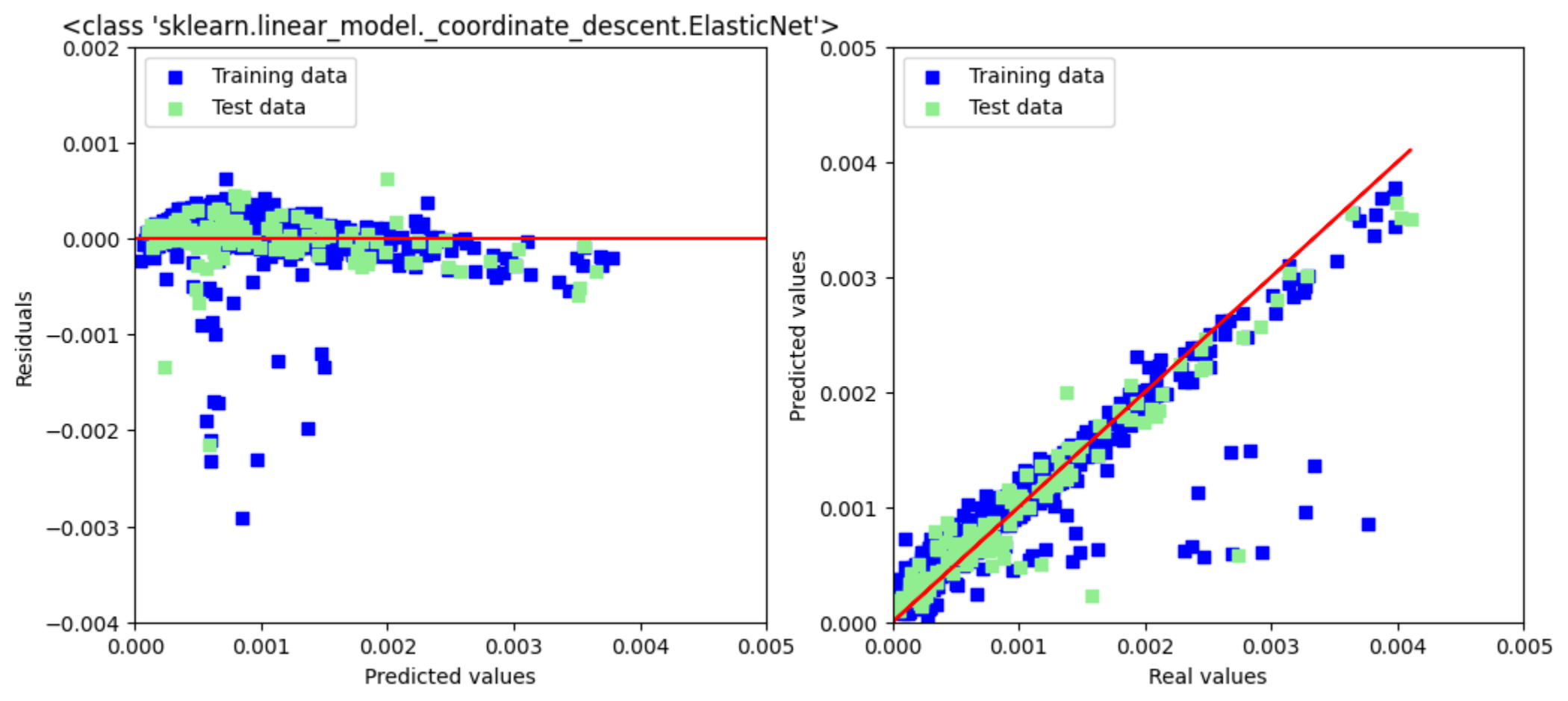


- La régularisation Lasso n'est pas meilleure que la baseline

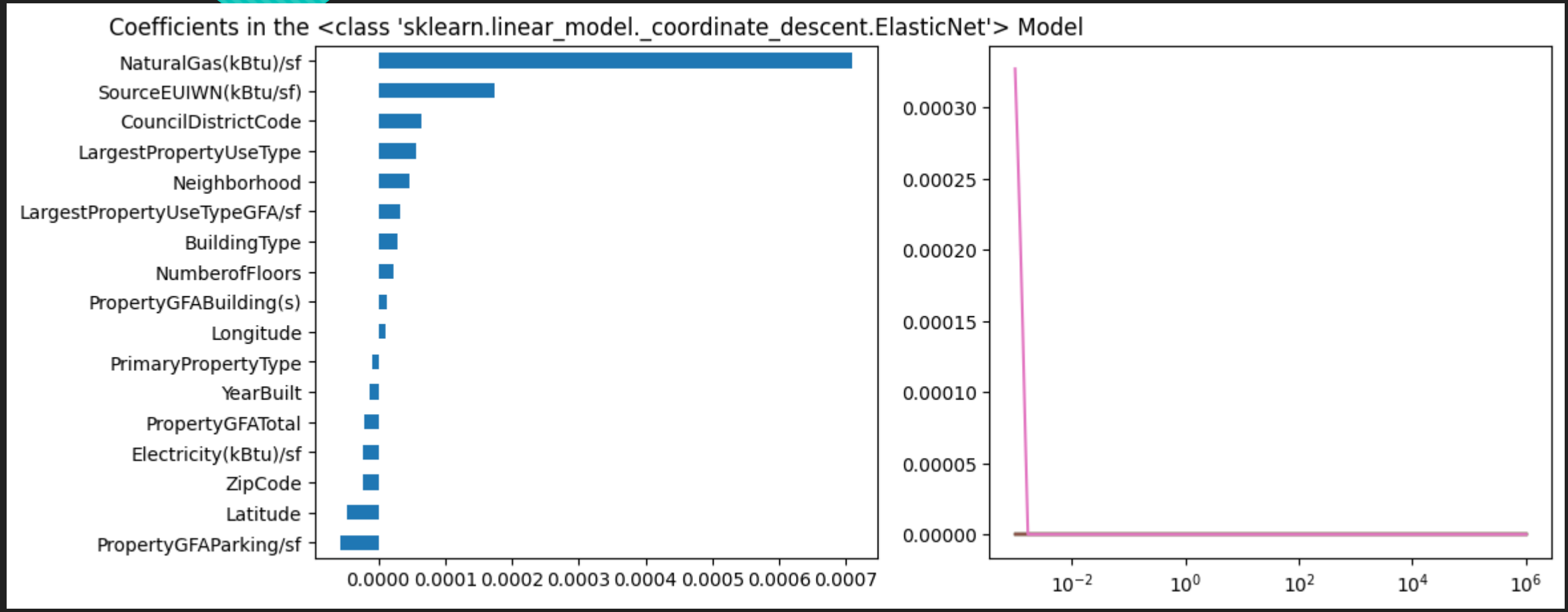
4-2-3 Régularisation Lasso



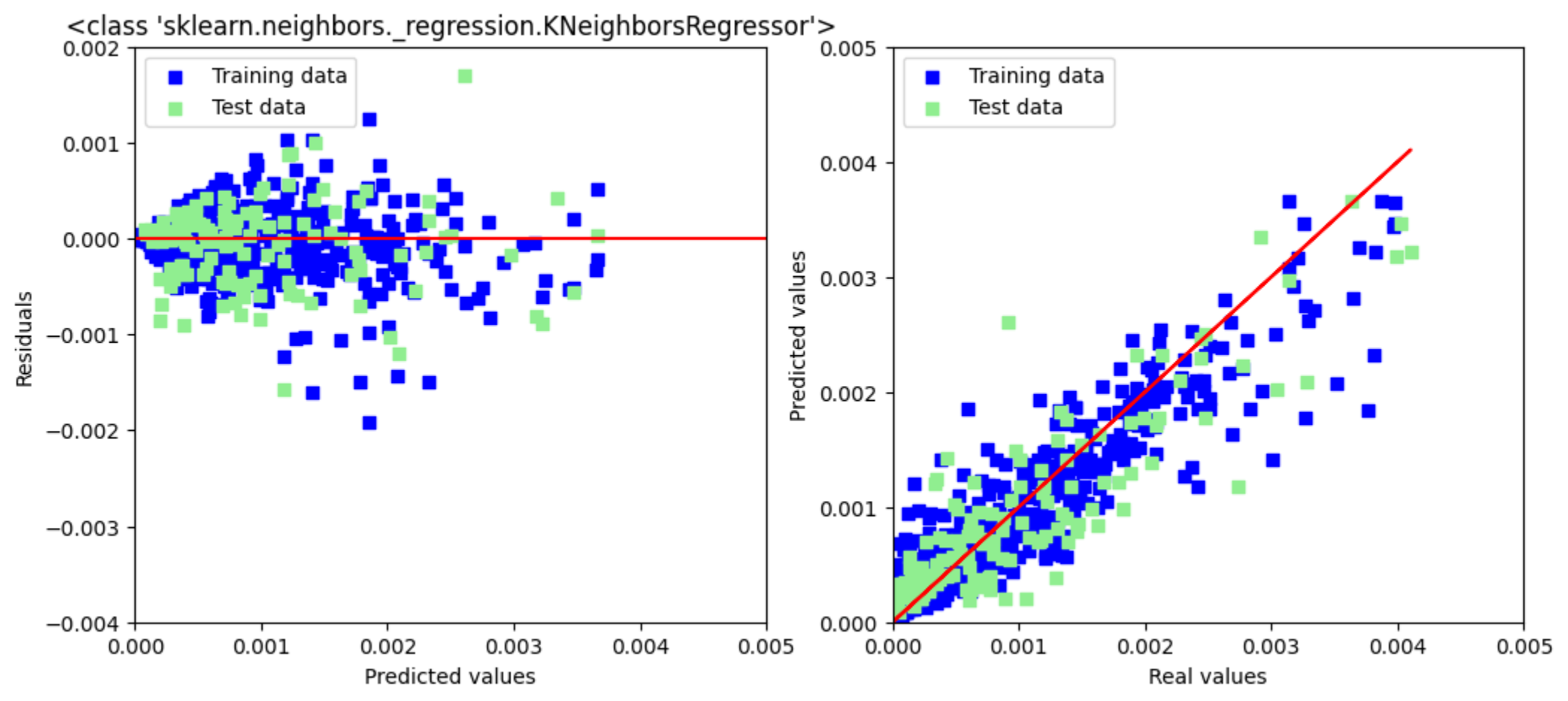
4-2-4 Régularisation ElasticNet



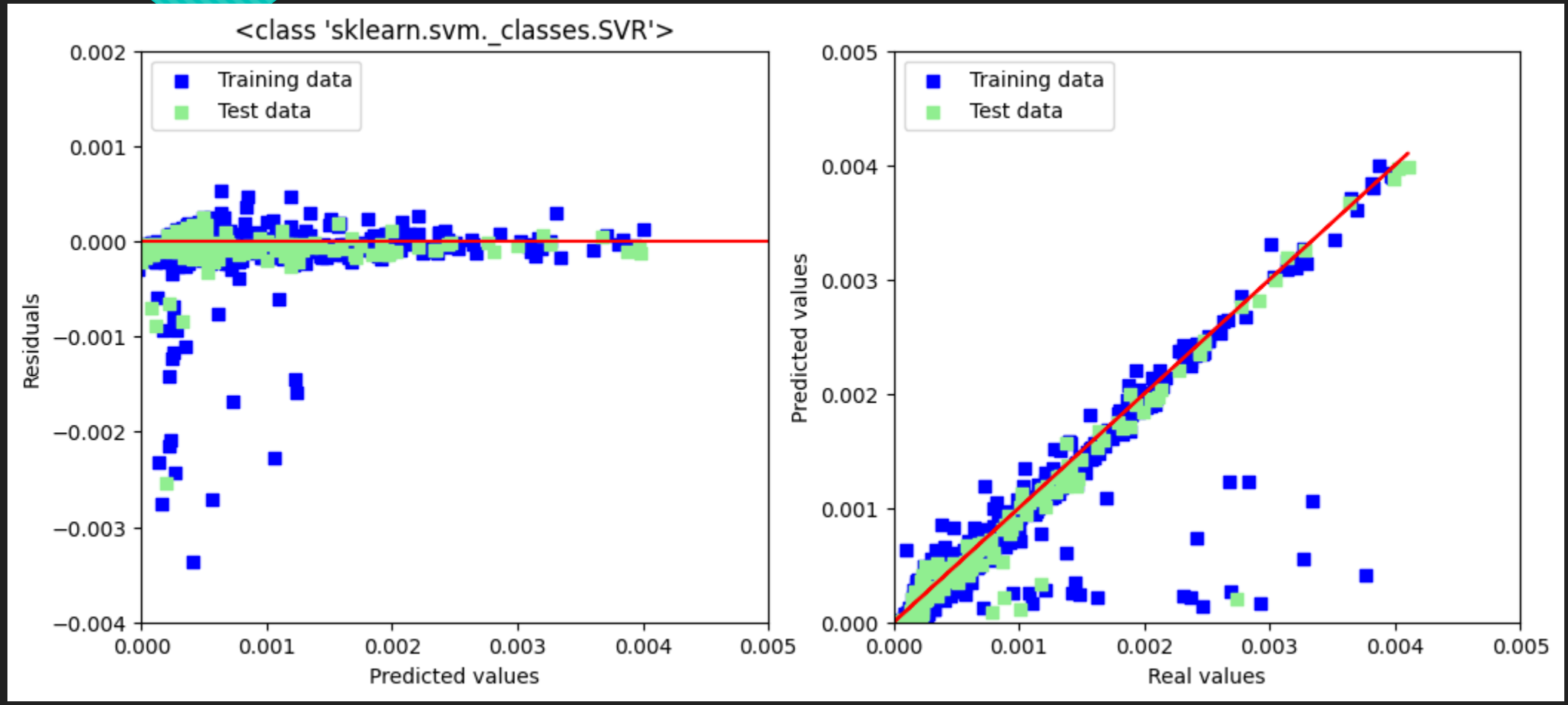
4-2-5 Régularisation ElasticNet



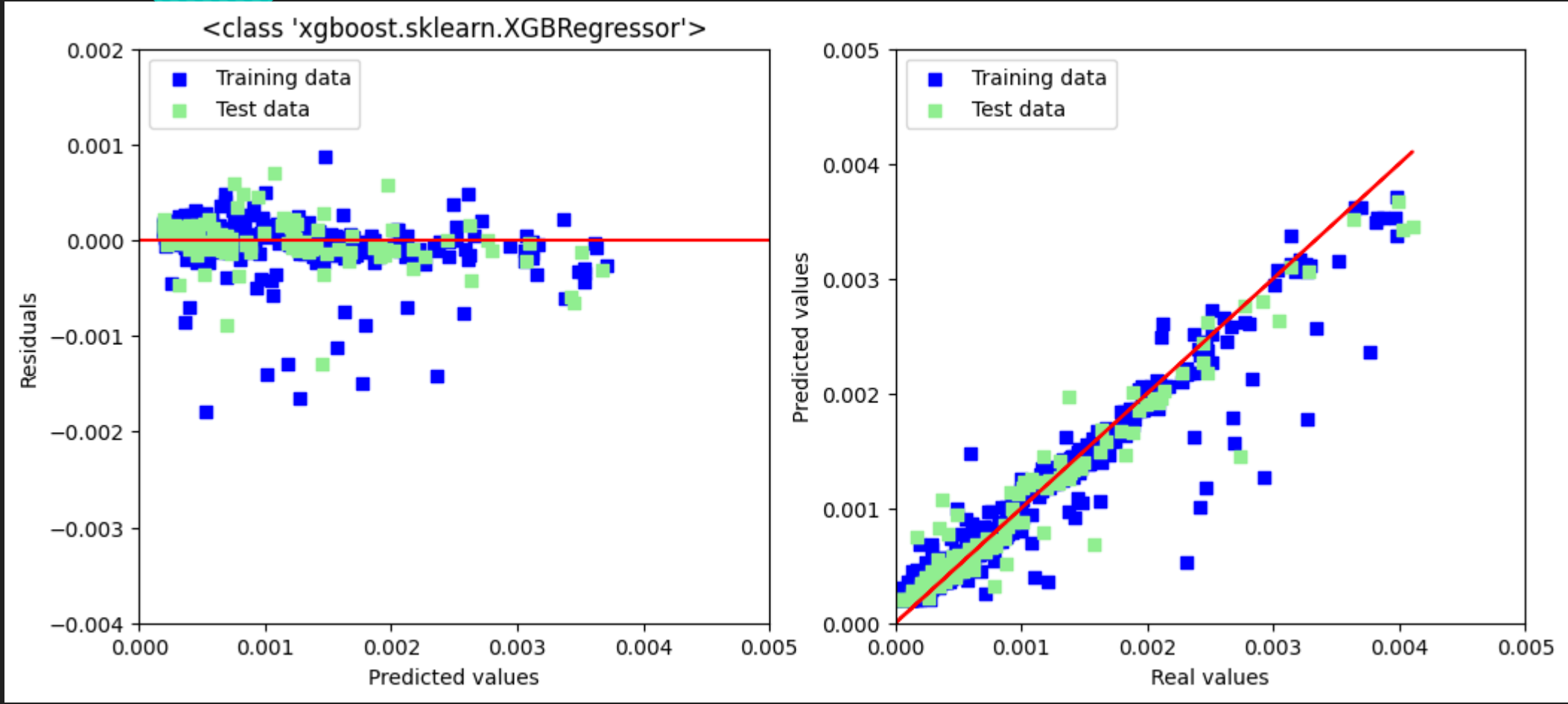
4-2-6 Régression KNN



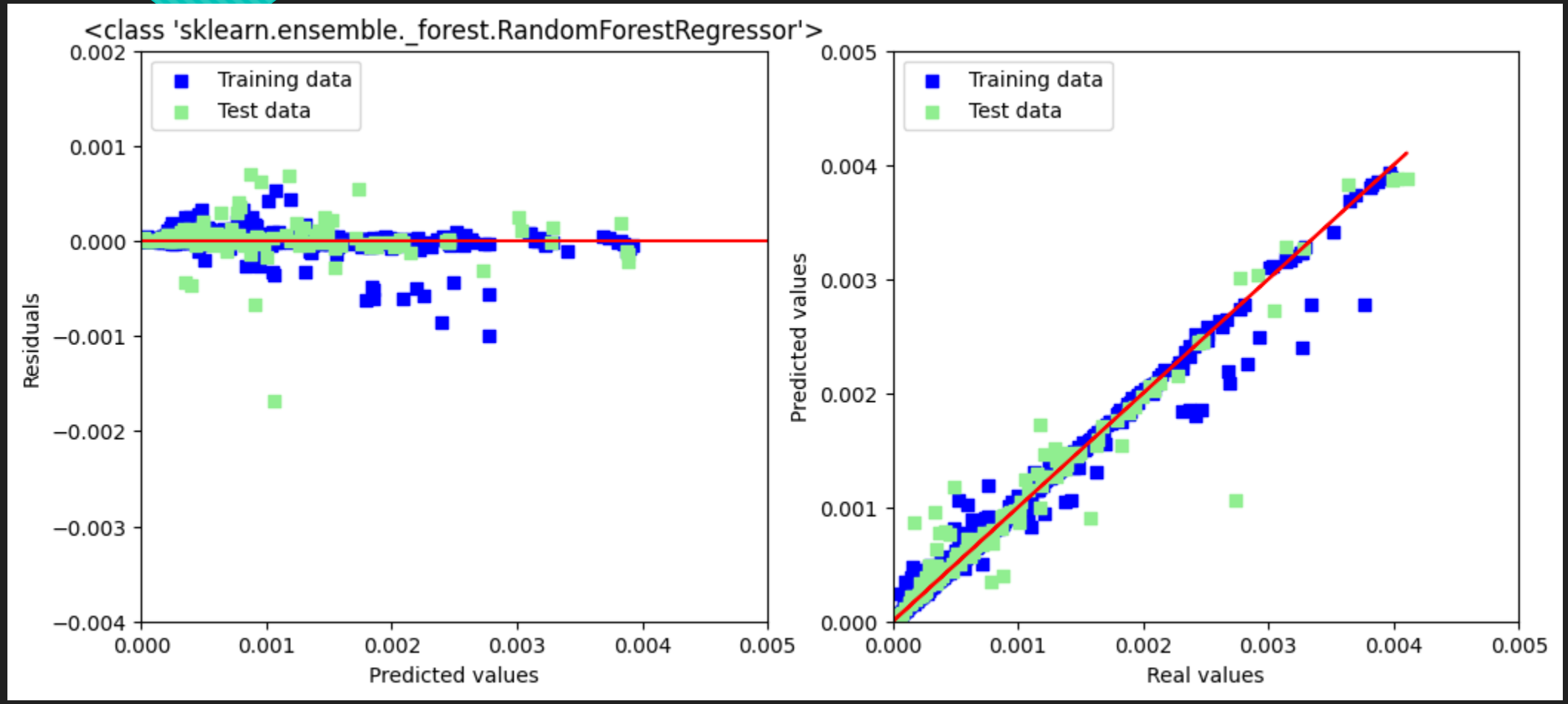
4-2-7 Régression Kernel SVR



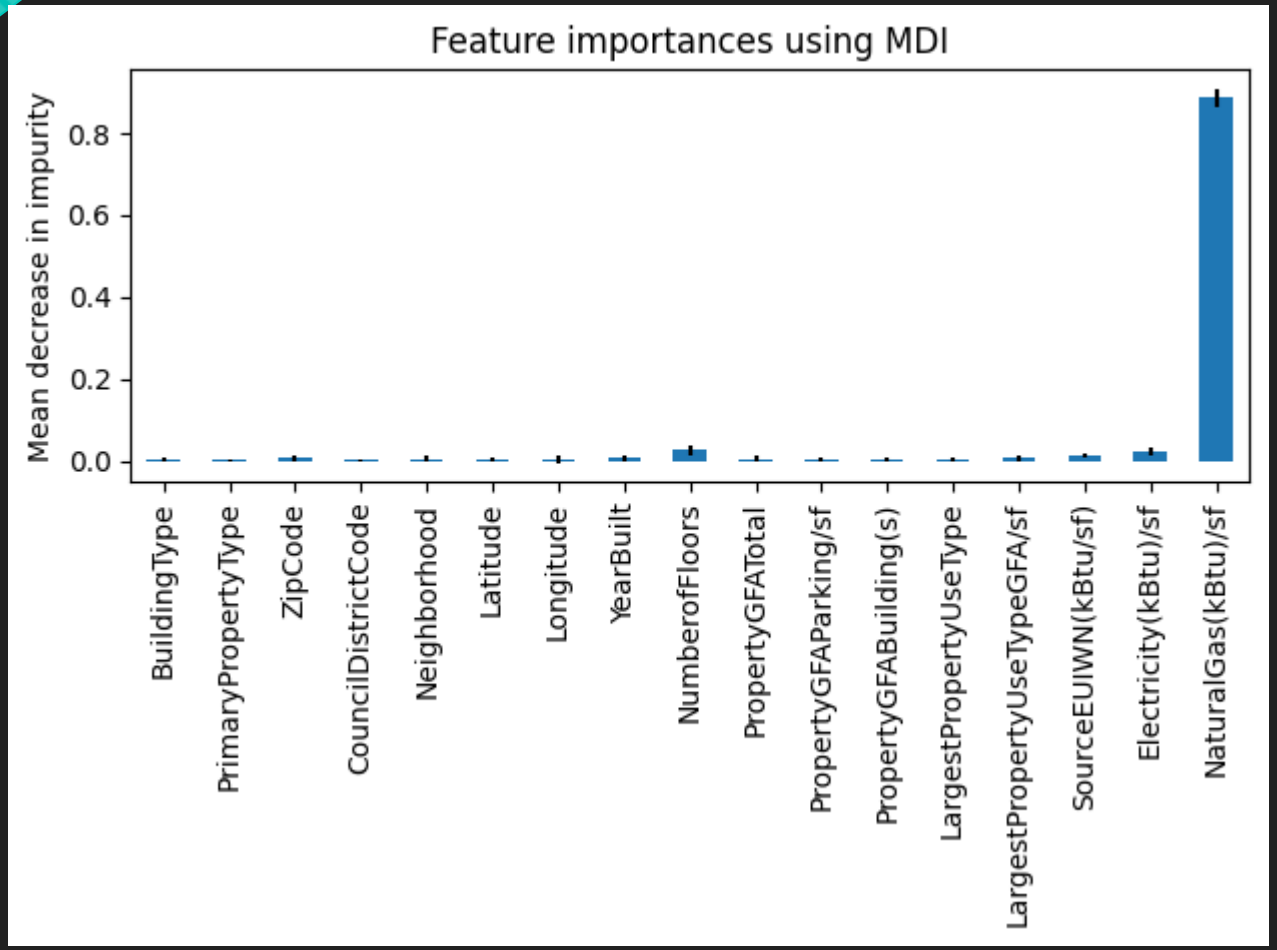
4-2-8 XGboost


<class 'xgboost.sklearn.XGBRegressor'>
Training data
Test data
Residuals
Predicted values
0.002
0.001
0.000
−0.001
−0.002
−0.003
−0.004
0.000
0.001
0.002
0.003
0.004
0.005
Training data
Test data
Predicted values
Real values
0.005
0.004
0.003
0.002
0.001
0.000

4-2-9 Random Forest



4-2-9 Random Forest



4-2-10 Synthèse modèle prédictif 'TotalGHGEmissions/sf'

	modèle	meilleurs_paramètres	meilleur_score	MAE	process_time
0	DummyRegressor()	{'strategy': 'mean'}	-0.010233	0.000725	0.011041
1	KernelRidge()	{'alpha': 0.1825183494319043, 'gamma': 0.01, ...	0.895662	0.000130	396.708819
2	Lasso()	{'alpha': 0.001}	-0.010233	0.000725	3.494279
3	ElasticNet()	{'alpha': 0.07149428986597581, 'l1_ratio': 0.0}	0.872433	0.000159	75.626973
4	KNeighborsRegressor()	{'n_neighbors': 3}	0.704190	0.000317	9.896932
5	SVR()	{'C': 0.0001, 'epsilon': 0, 'kernel': 'linear'...	0.859766	0.000150	8.442173
6	XGBRegressor()	{'colsample_bytree': 0.7, 'learning_rate': 0.0...	0.889256	0.000146	15.715476
7	**RandomForestRegressor()**	**{}**	**0.901588**	**0.000106**	**11.975214**

4-3 Synthèse pour les deux targets

Méthode	Score SiteEnergyUseWN(kBtu)/sf	Score TotalGHGEmissions/sf
DummyRegressor()	-0.010691	-0.010233
KernelRidge()	0.888619	0.895662
Lasso()	0.883485	-0.010233
ElasticNet()	0.883485	0.872433
KNeighborsRegressor()	0.737283	0.704190
SVR()	**0.889822**	0.859766
XGBRegressor()	0.873170	0.889256
RandomForestRegressor()	0.877248	**0.901588**

5 Comparaison avec et sans la variable EnergyStar

- On prend la meilleure méthode sélectionnée précédemment pour chacune des targets :
  - **SVR(kernel='linear', C=1, epsilon=1, max_iter=1000)** pour la target **EnergySiteEnergyUseWN(kBtu)/sf**
  - **RandomForestRegressor()** pour la target **TotalGHGEmissions/sf**
- On applique la méthode sur la portion de la base avec la colonne **EnergyStar** remplie (**702** lignes sur les **1071** à l'origine)
- On applique la méthode sur cette même portion de la base à laquelle on a enlevé la colonne **EnergyStar**
- On compare les scores

5-1 Comparaison avec la target SiteEnergyUseWN(kBtu)/sf

- On applique le modèle Kernel SVR optimisé sur la base de données avec et sans la colonne EnergyStar.
- Résultat :

	modèle	meilleurs_paramètres	Score sur la base modifiée	MAE
Sans la colonne EnergyStar	SVR()	{'C': 1.0, 'epsilon': 1, 'kernel': 'linear', '...	0.96167	2.440505
Avec la colonne EnergyStar	SVR()	{'C': 1.0, 'epsilon': 1, 'kernel': 'linear', '...	0.961416	2.447796

- Pour la target SiteEnergyUseWN(kBtu)/sf la variable EnergyStar améliore le score du modèle de 0.73 %

5-2 Comparaison avec la target TotalGHGEmissions/sf

- On applique le modèle RandomForest sur la base de données avec et sans la colonne EnergyStar.
- Résultat :

	modèle	meilleurs_paramètres	Score sur la base modifiée	MAE
Sans la colonne EnergyStar	RandomForestRegressor()	{}	0.94682	0.000096
Avec la colonne EnergyStar	RandomForestRegressor()	{}	0.942745	0.000103

- Pour la target TotalGHGEmissions/sf la variable EnergyStar améliore le score du modèle de 0.001 %

6 Conclusion

- Les modèles prédictifs Kernel SVR pour la consommation d'énergie et RandomForest pour les émissions de CO2 fournissent de très bons résultats de prédiction sur le jeu de données actuel (respectivement R2=89% et R2=90%)
- L'apport de l'information EnergyStar pour la prédiction est négligeable aussi bien pour la consommation d'énergie que pour les émissions de carbone. Les scores déjà élevés à la base expliquent l'absence d'amélioration.