



Projet 4: Anticipez les besoins en consommation électrique de bâtiments

QUENTIN STEPNIEWSKI

OPENCLASSROOMS

28 JUILLET 2020

Sommaire

1. Introduction – Présentation de la problématique
2. Présentation des données utilisées
3. Nettoyage de la base de données
4. Mise en place de modèles de prédiction
5. Comparaison des différents modèles
6. Conclusion sur les modèles et la problématique

1. Introduction – Présentation de la problématique



Problématique principale

- Prédire les performances énergétiques de bâtiments situés à Seattle
- Se passer des relevés faits sur place (opérations très coûteuses et fastidieuses)

Objectifs de l'étude

- Prédire la consommation en énergie des bâtiments
- Prédire les émissions en CO₂ des bâtiments
- Evaluer l'intérêt de « **l'ENERGYSTARScore*** »

*ENERGYSTARScore :

- Indicateur à échelle nationale permettant de refléter les performances énergétiques d'un bâtiment
- Score allant de 1 (mauvaise performance) à 100 (excellente performance)
- Un score de 50 représente la médiane nationale

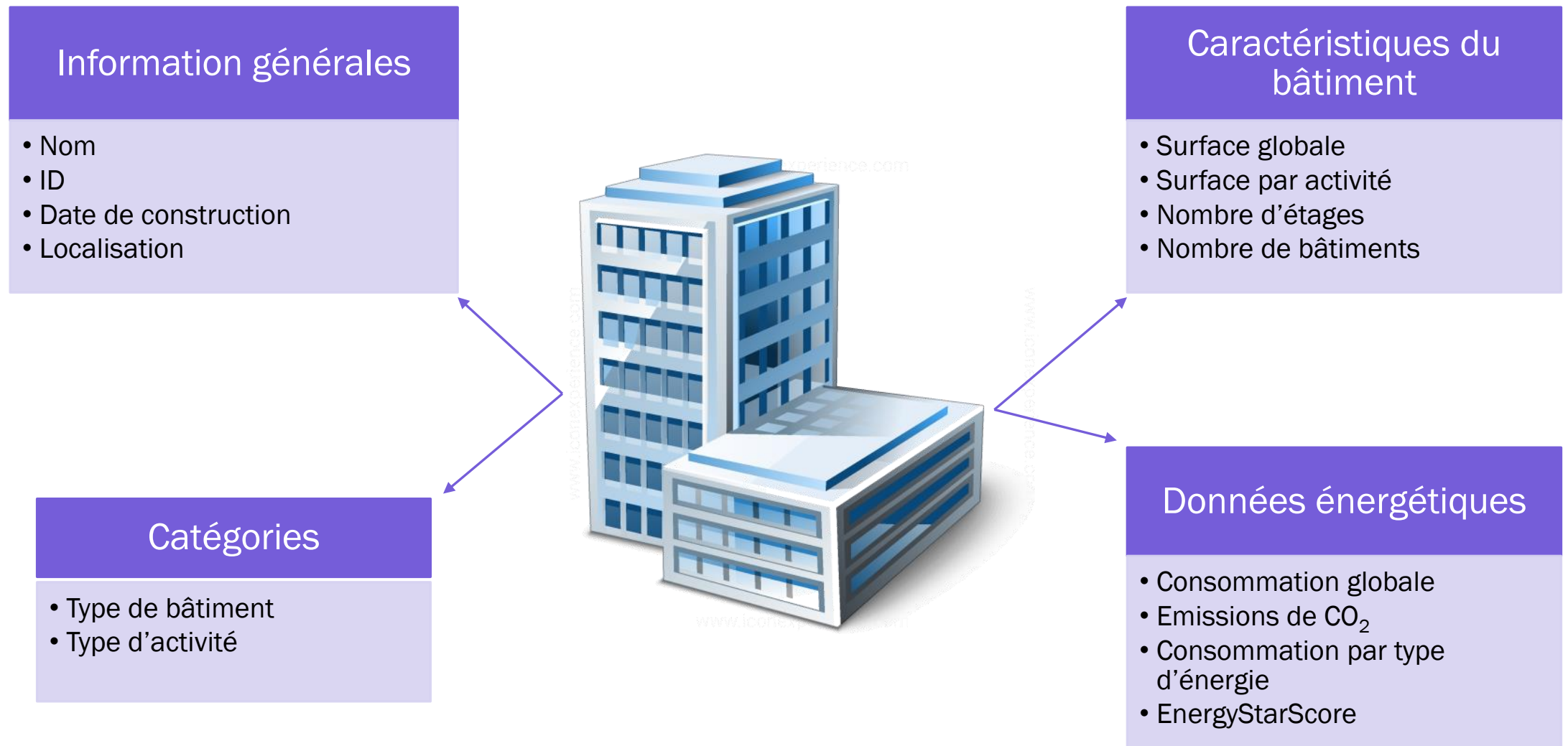
2. Présentation des données utilisées

2 bases de données concernant les bâtiments de Seattle :

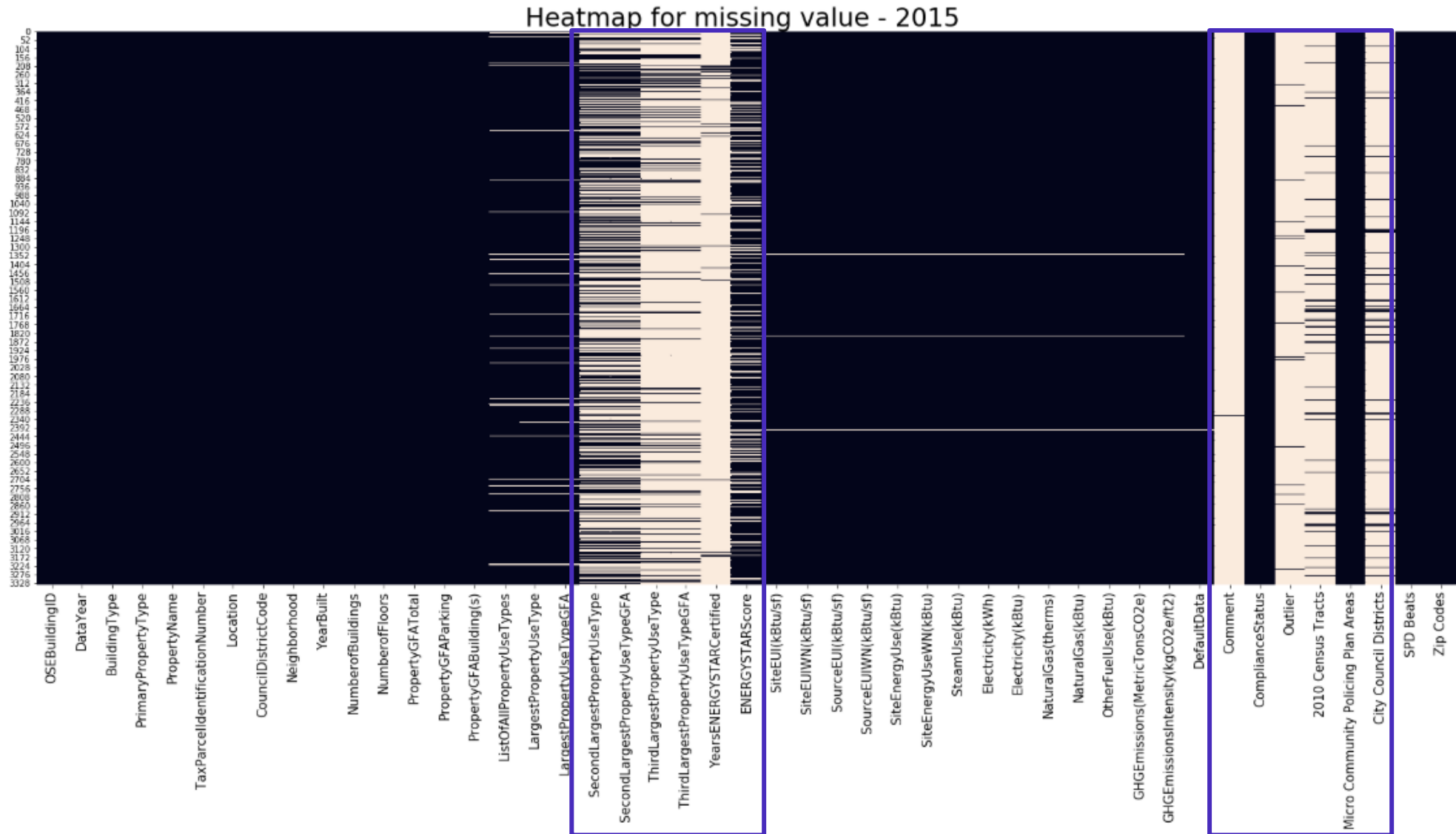
- Données de 2015 :
 - 3340 lignes
 - 47 colonnes
- Données de 2016 :
 - 3376 lignes
 - 46 colonnes



2. Présentation des données utilisées



2. Présentation des données utilisées



3. Nettoyage de la base de données



Gestion des targets

Fusion des 2 datasets (2015 – 2016) :

- Suppression des colonnes non communes
- Gestion des features identiques avec un nom différent

Choix et gestion des variables cibles :

Rappel des objectifs du projet

- Prédire la consommation d'énergie  SiteEnergyUseWN(kBtu)
- Prédire les émissions de CO₂  GHGEmissions(MetricTonsCO2e)

** Suppression de toutes les autres variables concernant les émissions/la consommation*

3. Nettoyage de la base de données

Gestion des targets

- Gestion des doublons

OSEBuildingID	max	min	diff_perc
1	249.98	249.43	0.220018
2	295.86	263.51	10.934226
3	2089.28	2061.48	1.330602
5	1936.34	286.43	85.207660
8	507.70	505.01	0.529840
...
50049	8.70	7.97	8.390805
50055	31.46	30.69	2.447552
50057	627.97	395.26	37.057503
50058	5.46	5.42	0.732601
50059	6.74	6.74	0.000000

```
study_duplicates['OSEBuildingID'].value_counts().describe()
```

```
count    3284.0
```

```
mean      2.0
```

```
std       0.0
```

```
min       2.0
```

```
25%      2.0
```

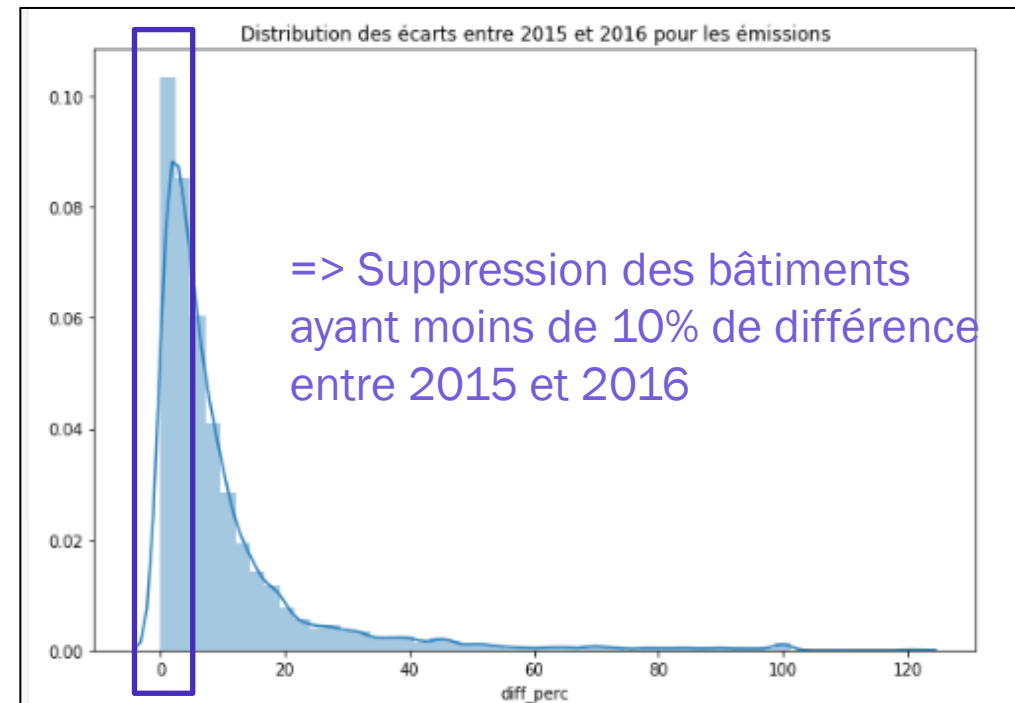
```
50%      2.0
```

```
75%      2.0
```

```
max       2.0
```

```
Name: OSEBuildingID, dtype: float64
```

=> 1642 bâtiments présents à la fois en 2015 et 2016

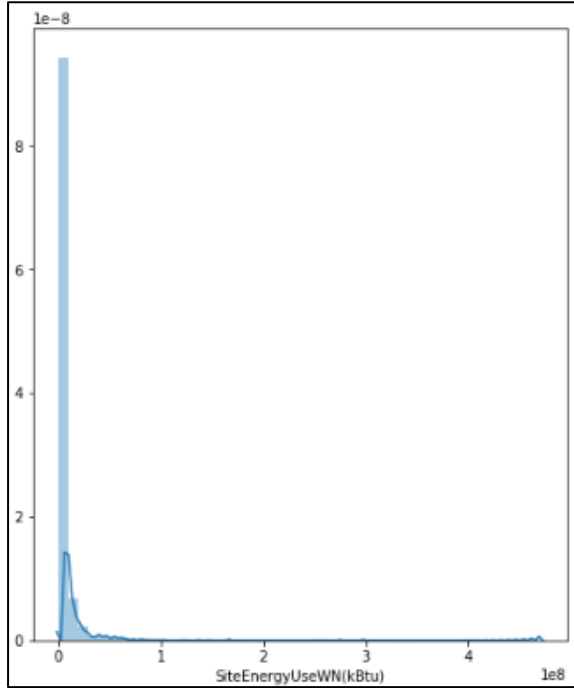


3. Nettoyage de la base de données

Gestion des targets

- Gestion des doublons
- Gestion des valeurs aberrantes (émission et/ou consommation nulle)
- Gestion de l'échelle des targets

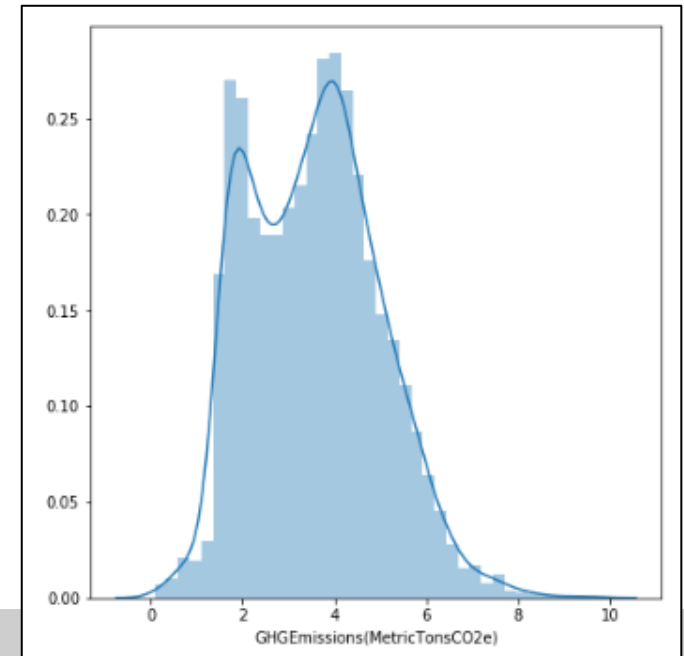
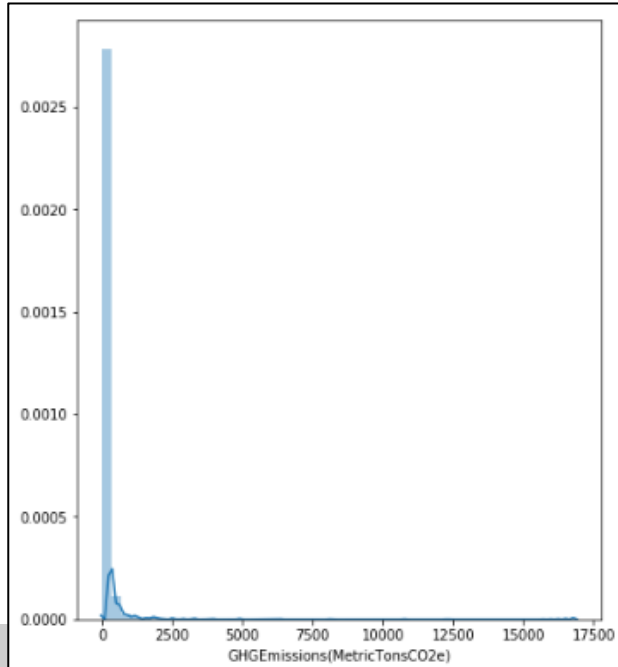
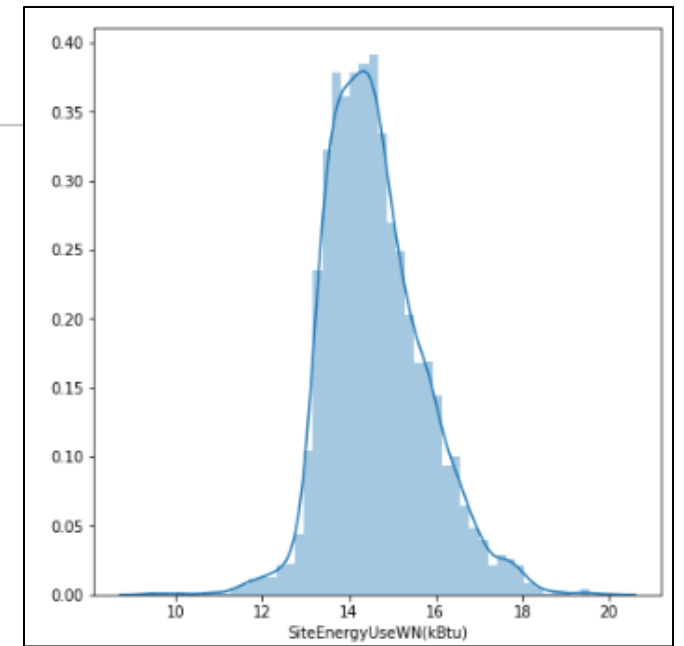
de la base de données



```
skewness for SiteEnergyUseWN(kBtu): 15.077735359310045  
skewness for GHGEmissions(MetricTonsCO2e): 20.959079985948026
```



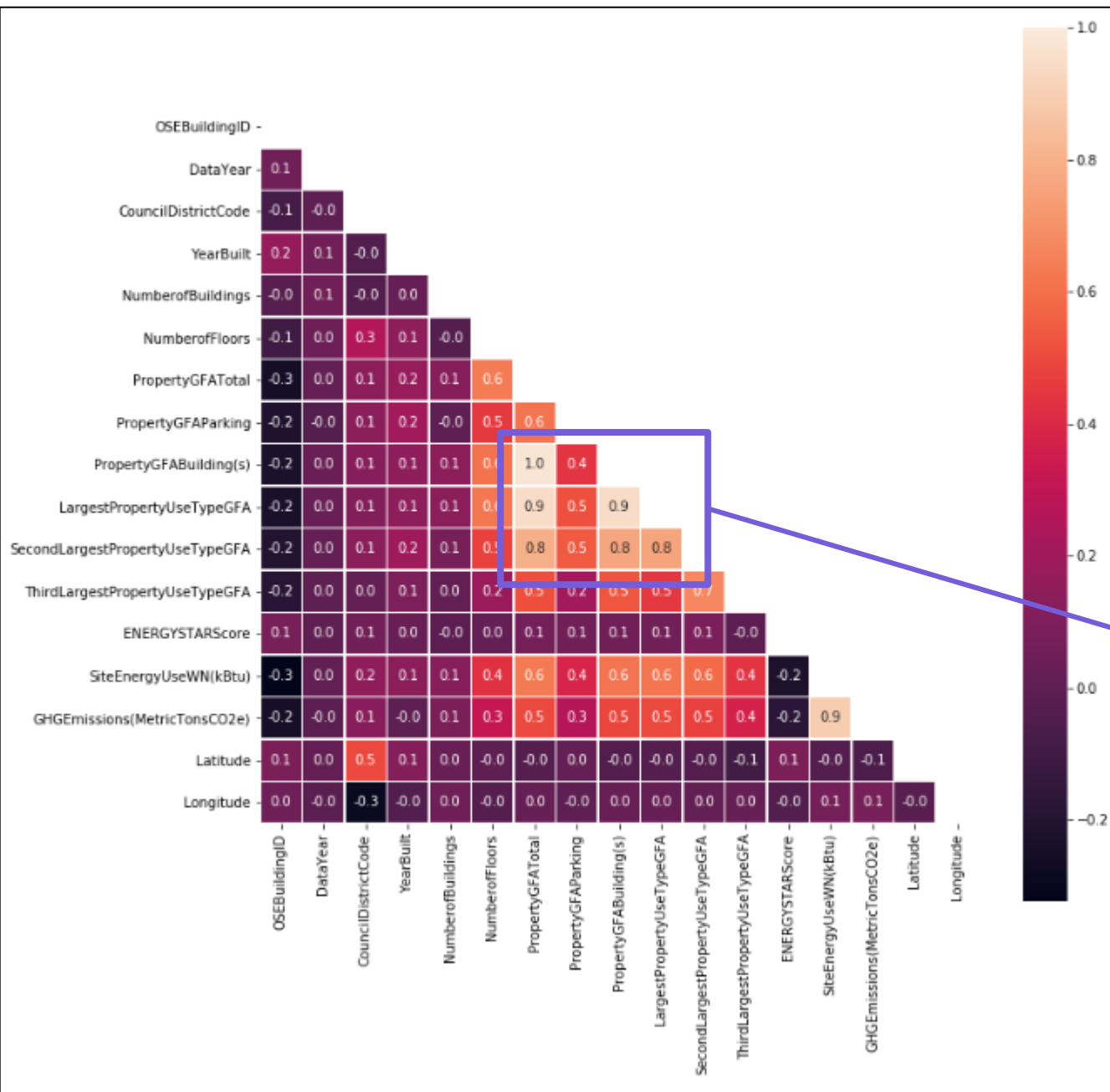
`np.log1p()`



3. Nettoyage de la base de données

Gestion des features

- Suppression des bâtiments dont l'ENERGYSTARScore n'est pas renseigné
- Gestion de valeurs aberrantes (NumberOfBuilding = 0 ou NumberOfFloor = 0)
- Suppression des variables inutiles (PropertyName, State, Comments, ...)
- Etude des corrélations entre les variables



pas renseigné

NumberofFloor = 0)

omments, ...)

Corrélation importante entre les différentes variables de surface :

- PropertyGFATotal
- Building
- LargestPropertyUseTypeGFA

=> Variables à inspecter entre elles par la suite

3. Nettoyage de la base de données

Gestion des features

- Suppression des bâtiments dont l'ENERGYSTARScore n'est pas renseigné
- Gestion de valeurs aberrantes (NumberOfBuilding = 0 ou NumberOfFloor = 0)
- Suppression des variables inutiles (PropertyName, State, Comments, ...)
- Etude des corrélations entre les variables
- Gestion des outliers sur les surfaces

Vérification:

$$\text{PropertyGFATotal} = \text{GFABuilding} + \text{GFAParking}$$

OK : On pourra donc
supprimer
PropertyGFATotal

Vérification:

$$\text{PropertyGFATotal} = \text{LargestTypeGFA} + \text{SecondTypeGFA} + \text{ThirdTypeGFA}$$

NOK : Suppression des
bâtiments dont la
différence entre ces
valeurs est > 99%

3. Nettoyage de la base de données

Gestion des features

- Suppression des bâtiments dont l'ENERGYSTARScore n'est pas renseigné
- Gestion de valeurs aberrantes (NumberOfBuilding = 0 ou NumberOfFloor = 0)
- Suppression des variables inutiles (PropertyName, State, Comments, ...)
- Etude des corrélations entre les variables
- Gestion des outliers sur les surfaces
- Création de 2 variables :
 - *BuildingAge* (DataYear – YearBuilt)
 - *Volume* (BuildingGFA * Hauteur moyenne d'un étage [8 feet, trouvée sur internet])

Réduction du nombre de catégories pour les variable 'UseType' (Réduction à ~10 catégories)

```
typedict = {'Other - Entertainment/Public Assembly':'Entertainment',
            'Fitness Center/Health Club/Gym':'Entertainment',
            'Museum':'Entertainment',
            'Worship Facility':'Entertainment',
            'Movie Theater':'Entertainment',
            'Convention Center':'Entertainment',
            'Other - Recreation':'Entertainment',
            'Swimming Pool':'Entertainment',
            'Entertainment':'Entertainment',

            'Medical Office':'Hospital',
            'Hospital (General Medical & Surgical)':'Hospital',
            'Other/Specialty Hospital':'Hospital',
            'Urgent Care/Clinic/Other Outpatient':'Hospital',
            'Outpatient Rehabilitation/Physical Therapy':'Hospital',
            'Hospital':'Hospital',

            'Hotel':'Hotel',
            'Multifamily Housing':'Hotel',
            'Senior Care Community':'Hotel',
            'Residence Hall/Dormitory':'Hotel',
            'Residential Care Facility':'Hotel',
            'Other - Lodging/Residential':'Hotel',
            'Prison/Incarceration':'Hotel',
            'Single Family Home':'Hotel',
            'Mid-Rise Multifamily':'Hotel',
            'Low-Rise Multifamily':'Hotel',
            'High-Rise Multifamily':'Hotel',
            'Residence Hall':'Hotel',

            'Manufacturing/Industrial Plant':'Manufacturing',
            'Manufacturing':'Manufacturing',

            'Police Station':'Office',
            'Office':'Office',
            'Government Office':...
```

de données

Score n'est pas renseigné

ing = 0 ou NumberOfFloor = 0)

One Hot Encoding sur ces features :

=> Transformation d'une feature à n catégories en n features booléennes

e [8 feet, trouvée sur internet])

3. Nettoyage de la base de données

Gestion des features

- Suppression des bâtiments dont l'ENERGYSTARScore n'est pas renseigné
- Gestion de valeurs aberrantes (NumberOfBuilding = 0 ou NumberOfFloor = 0)
- Suppression des variables inutiles (PropertyName, State, Comments, ...)
- Etude des corrélations entre les variables
- Gestion des outliers sur les surfaces
- Création de 2 variables :
 - *BuildingAge* (DataYear – YearBuilt)
 - *Volume* (BuildingGFA * Hauteur moyenne d'un étage [8 feet, trouvée sur internet])
- Re-catégorisation des variables PropertyUseType + OneHotEncoding
- Target Encoding de la variable BuildingType

Dataset
après
fusion
2015-
2016

- 6716 lignes
- 46 colonnes

DataSet
Final

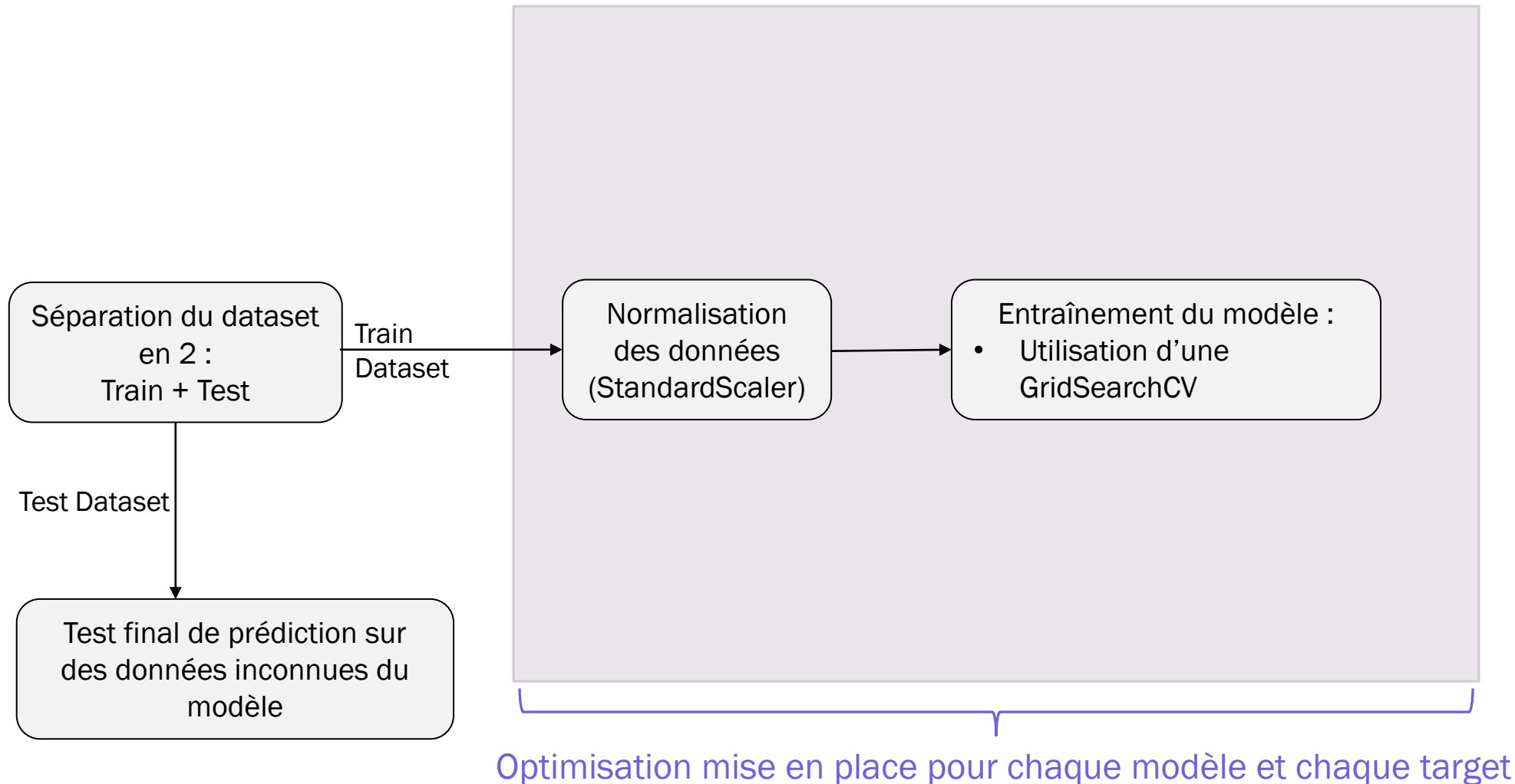
- 3478 lignes
- 46 colonnes
(dont 2
targets et 33
OHE features)

**TargetEncoding :*

Remplacer chaque catégorie d'une variable par la valeur de la moyenne de la target pour cette catégorie

4. Mise en place de modèles de prédiction

Principe mis en place pour la prédiction :



4. Mise en place de la prédiction

GridSearchCV

Crossvalidation

Optimisation des hyperparamètres du modèle

	Train Set					Score
Split 1	Validation	Train	Train	Train	Train	$R^2 = 0,92$
Split 2	Train	Validation	Train	Train	Train	$R^2 = 0,89$
Split 3	Train	Train	Validation	Train	Train	$R^2 = 0,81$
Split 4	Train	Train	Train	Validation	Train	$R^2 = 0,86$
Split 5	Train	Train	Train	Train	Validation	$R^2 = 0,90$
Test Dataset						$R^2 = 0,88$

Test final de prédiction sur des données inconnues du modèle

*Hyperparamètres :

Paramètres intrinsèques d'un modèle (ex: le nombre de voisin à observer pour un KNN Classifier)

=> GridSearchCV va construire un modèle pour chaque combinaison d'hyperparamètres

Optimisation mise en place pour chaque modèle et chaque target

4. Mise en place de GridSearchCV pour la prédiction

GridSearchCV

Crossvalidation

Optimisation des
hyperparamètres du modèle

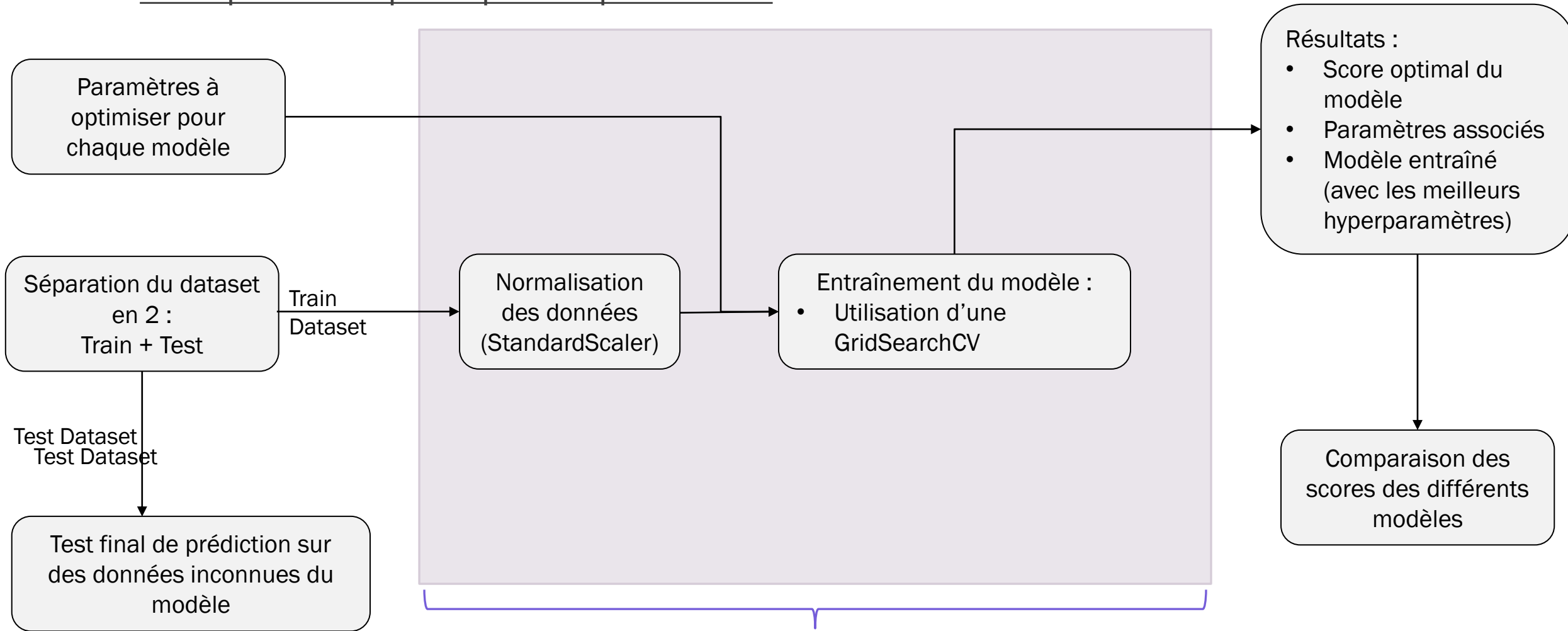
	Train Set					Modèle 1	Modèle 2	Modèle 3	...
Split 1	Validation	Train	Train	Train	Train	$R^2 = 0,92$	$R^2 = 0,84$	$R^2 = 0,86$...
Split 2	Train	Validation	Train	Train	Train	$R^2 = 0,89$	$R^2 = 0,91$	$R^2 = 0,86$...
Split 3	Train	Train	Validation	Train	Train	$R^2 = 0,81$	$R^2 = 0,83$	$R^2 = 0,92$...
Split 4	Train	Train	Train	Validation	Train	$R^2 = 0,86$	$R^2 = 0,91$	$R^2 = 0,92$...
Split 5	Train	Train	Train	Train	Validation	$R^2 = 0,90$	$R^2 = 0,87$	$R^2 = 0,91$...
Test Dataset						$R^2 = 0,88$	$R^2 = 0,87$	$R^2 = 0,89$...

Test final de prédiction sur
des données inconnues du
modèle

Optimisation mise en place pour chaque modèle et chaque target

4. Mise en place de modèles de prédiction

Principe mis en place pour la prédiction :



Optimisation mise en place pour chaque modèle et chaque target

4. Mise en place de modèles de prédiction

Modèles étudiés :

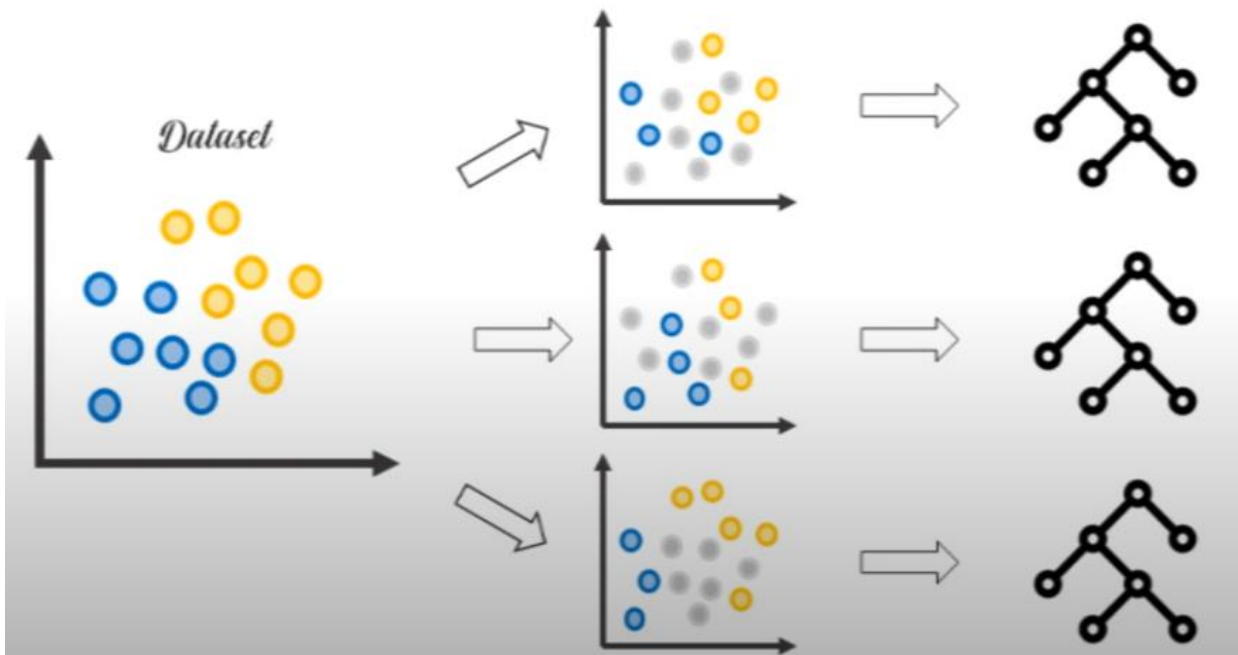
- ElasticNet
- SVR
- MLP
- Bagging (RandomForest)
- Boosting (GradientBoosting)

4. Mise en place de modèles de prédiction

Bagging (RandomForestRegressor)

**Bagging* :

- entraîner plusieurs modèles (ici plusieurs arbres de décision) sur une portion aléatoire du dataset (Bootstrapping)
- prendre le résultat moyen des modèles



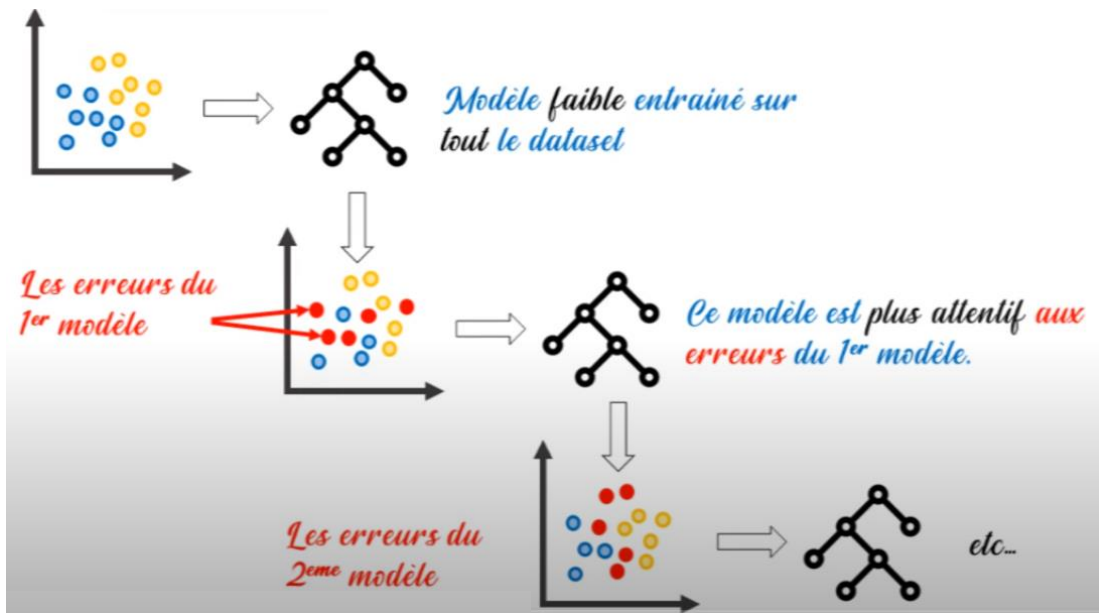
- Chaque modèle sera un '**apprenant fort**' (biais faible mais forte variance)
- La foule de modèle va permettre de **réduire** cette **variance** pour obtenir un résultat de prédiction plus robuste
- RandomForest se base sur des arbres de décision, les hyperparamètres :
 - `n_estimators` (nombre d'arbres)
 - `max_depth` (profondeur max de l'arbre)
 - `min_samples_leaf` (nombre mini d'individus à avoir à gauche ET à droite d'un nœud)
 - `Max_feature` (nombre de features à observer au moment de faire une séparation)

4. Mise en place de modèles de prédiction

Boosting (GradientBoostingRegressor)

**Boosting* :

- Entraîner des modèles de manière séquentielle (les uns après les autres)
- Chaque modèle cherche à s'améliorer sur les erreurs faites par le modèle précédent



- Ici, les modèles sont plutôt des '**apprenants faibles**' (biais important)
- La foule de modèles va permettre de **réduire** ce **biais** et donc créer des modèles performants
- GradientBoosting se base également sur des arbres de décision: **mêmes hyperparamètres**. On rajoute simplement l'aspect de `learning_rate` pour notre descente de gradient:
 - `Learning_rate` (le 'pas' utilisé pour la descente de gradient)

4. Mise en place de modèles de prédiction

ElasticNet	SVR	MLP	RandomForest	GradienBoosting
Alpha logspace[10^{-5} : 10^5]	Gamma logspace[10^{-5} : 10^5]	Hidden_layer_size [(10,10,10) , (20,20,20) , (30,30,30)]	N_estimators [10 , 50 , 100 , 150 , 200 , 250 , 300]	N_estimators [10 , 50 , 100 , 150 , 200 , 250 , 300]
L1_ratio [0 , 0.1 , 0.2 , ... , 0.9 , 1]	C [1 , 10 , 100 , 1000]	Max_iter [50 , 100 , 200 , 500]	Max_depth linspace[10:110]	Max_depth linspace[10:110]
Tol [0.1 , 0.01 , 0.001 , 0.0001]	Epsilon [0.001 , 0.01 , 0.1 , 1]	Batch_size [100, 200 , 500]	Min_samples_leaf [1 , 3 , 5 , 7 , 9 , 11]	Min_samples_leaf [1 , 3 , 5 , 7 , 9 , 11]
		Learning_rate_init [0.0005 , 0.001 , 0.005]	Max_features ['auto','sqrt']	Learning_rate [0.08 ,0.1 ,0.12 ,0.15],

5. Comparaison des différents modèles

Résultats obtenus :

Consommation

RMSE of Median based Model for Consumption: 1.1747302901939065

	r2	best_param_r2	neg_root_mean_squared_error	best_param_neg_root_mean_squared_error
ElasticNet	0.567912	{'alpha': 0.029470517025518096, 'l1_ratio': 0....	-0.70129	{'alpha': 0.04714868363457394, 'l1_ratio': 0.0...
SVR	0.711884	{'gamma': 0.004281332398719391, 'C': 10, 'epsi...	-0.572557	{'gamma': 0.004281332398719391, 'C': 10, 'epsi...
MLP	0.756861	{'batch_size': 100, 'hidden_layer_sizes': (20,...	-0.532793	{'batch_size': 100, 'hidden_layer_sizes': (30,...
RandomForest	0.810972	{'n_estimator': 200, 'max_depth': 95.714285714...	-0.461663	{'n_estimator': 250, 'max_depth': 38.571428571...
GradientBoosting	0.81658	{'n_estimator': 150, 'max_depth': 65.0, 'min_s...	-0.45415	{'n_estimator': 100, 'max_depth': 110.0, 'min_...

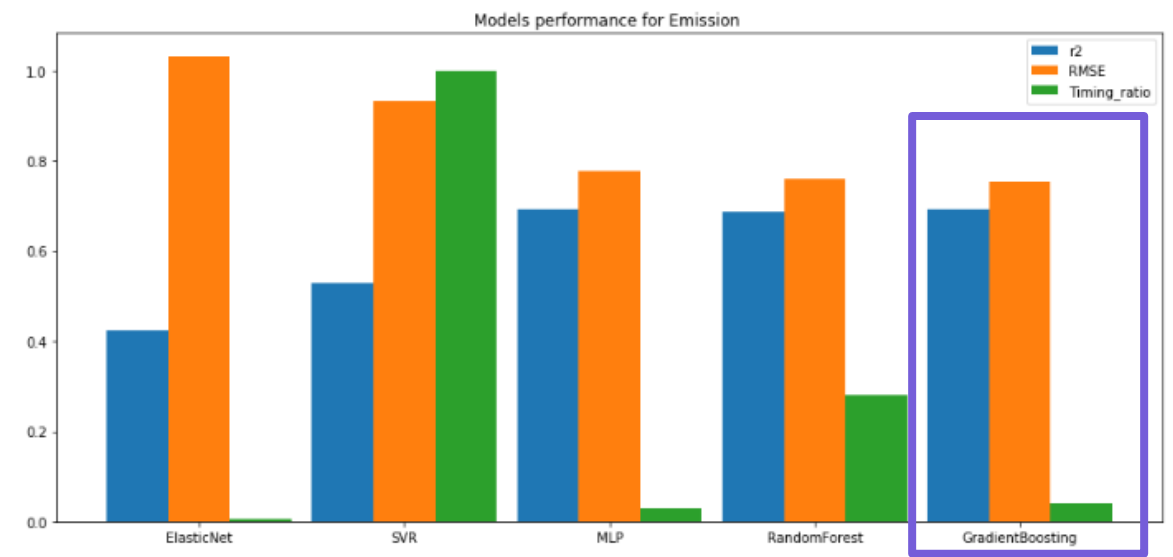
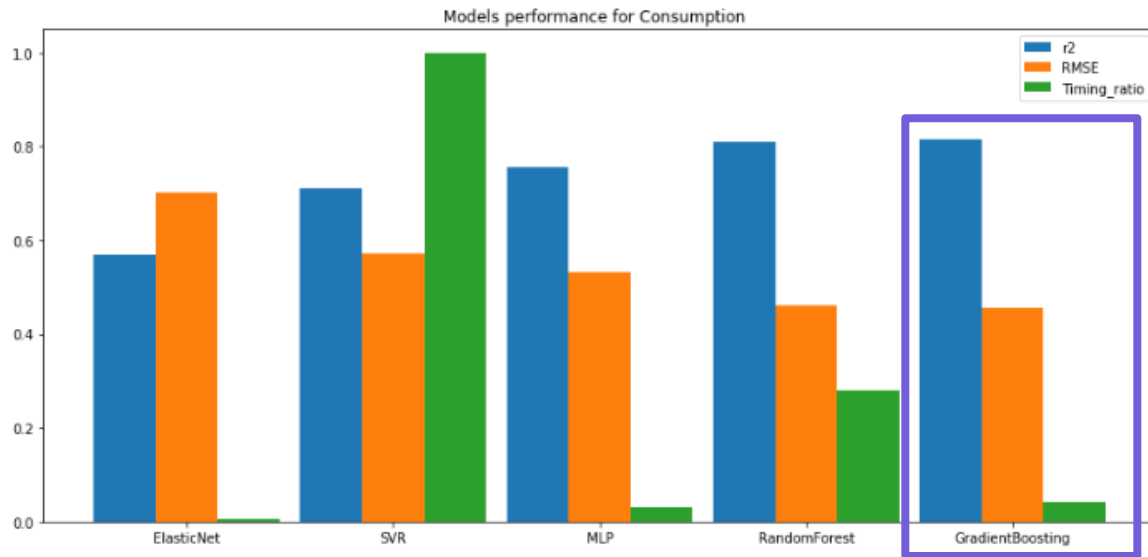
Emission

RMSE of Median based Model for Emission: 1.8646778855508095

	r2	best_param_r2	neg_root_mean_squared_error	best_param_neg_root_mean_squared_error
ElasticNet	0.424899	{'alpha': 0.018420699693267165, 'l1_ratio': 0....	-1.03131	{'alpha': 0.018420699693267165, 'l1_ratio': 0....
SVR	0.529505	{'gamma': 0.004281332398719391, 'C': 10, 'epsi...	-0.932794	{'gamma': 0.004281332398719391, 'C': 10, 'epsi...
MLP	0.692366	{'batch_size': 100, 'hidden_layer_sizes': (30,...	-0.776892	{'batch_size': 100, 'hidden_layer_sizes': (30,...
RandomForest	0.68594	{'n_estimator': 150, 'max_depth': 110.0, 'min_...	-0.760681	{'n_estimator': 300, 'max_depth': 38.571428571...
GradientBoosting	0.691715	{'n_estimator': 150, 'max_depth': 80.0, 'min_s...	-0.753731	{'n_estimator': 100, 'max_depth': 110.0, 'min_...

5. Comparaison des différents modèles

Comparaison des modèles :

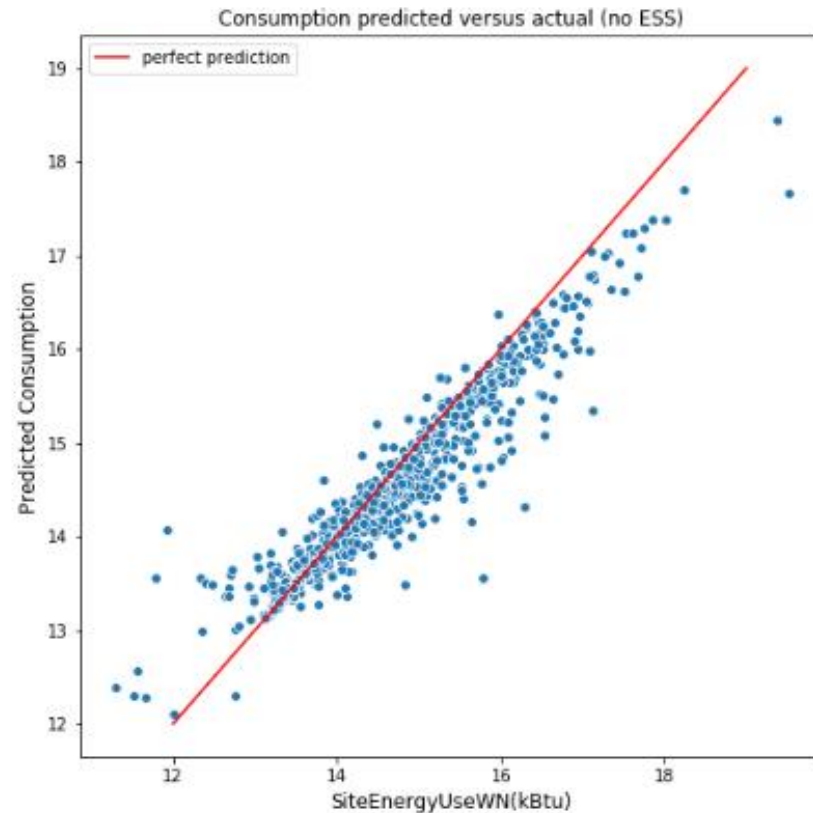


Ici, on choisira donc le modèle GradientBoosting qui démontre les meilleures performances :

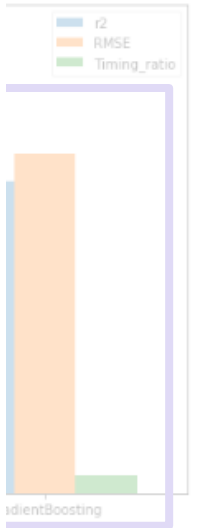
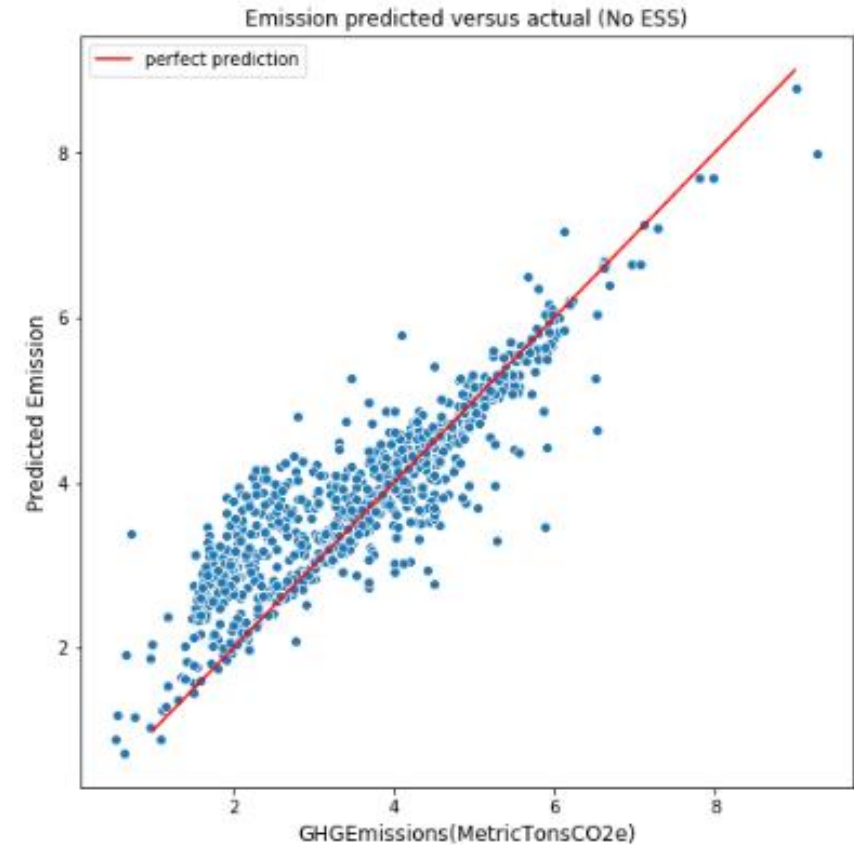
- Validation score consommation (RMSE= 0,45)
- Validation score émission (RMSE= 0,75)
- Parmi les meilleurs en temps d'exécution

5. Comparaison des différents modèles

Comparaison des modèles :



qui démontre



Ici, on c

- Valid
- Valid
- Parmi les meilleurs en temps d'exécution

5. Comparaison des différents modèles

Évaluation de l'intérêt de l'ENERGYSTARScore :

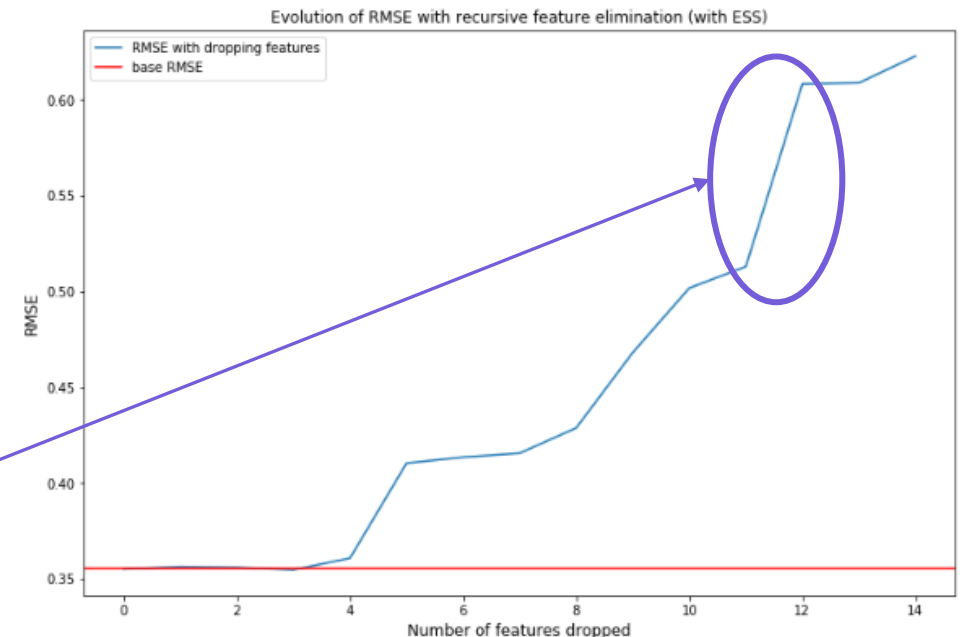
Modèle GradientBoosting avec EnergyStarScore :

- Meilleur score de validation en consommation (RMSE= 0,36) → ~20% d'amélioration
- Meilleur score de validation en émission (RMSE= 0,70) → ~7% d'amélioration

Recursive feature Elimination (sklearn + fonction créée pour le projet) :

	Feature_name	Rank_feature
5	LargestPropertyUseTypeGFA	1
4	PropertyGFABuilding(s)	2
43	Volume	3
8	ENERGYSTARScore	4
2	NumberofFloors	5
0	BuildingType_Consumption	6
9	BuildingAge	7
6	SecondLargestPropertyUseTypeGFA	8
17	LargestPropertyUseType_Storage	9
3	PropertyGFAParking	10
14	LargestPropertyUseType_Retail	11
7	ThirdLargestPropertyUseTypeGFA	12
12	LargestPropertyUseType_Hotel	13
13	LargestPropertyUseType_Office	14
16	LargestPropertyUseType_Special Storage	15

	Feature_dropped	Loop_score
0	SecondLargestPropertyUseType_	-0.355177
1	ThirdLargestPropertyUseType_	-0.356343
2	NumberofBuildings	-0.355998
3	CouncilDistrictCode_	-0.354757
4	LargestPropertyUseType_	-0.360814
5	ThirdLargestPropertyUseTypeGFA	-0.410389
6	PropertyGFAParking	-0.413493
7	NumberofFloors	-0.415579
8	BuildingType_Consumption	-0.428755
9	BuildingAge	-0.468077
10	SecondLargestPropertyUseTypeGFA	-0.501744
11	ENERGYSTARScore	-0.512946
12	Volume	-0.608242
13	PropertyGFABuilding(s)	-0.60876
14	LargestPropertyUseTypeGFA	-0.622771



6. Conclusion sur les modèles et la problématique

Prédiction des émissions et de la consommation sans passer par le relevé annuel :

	Consommation		Emission	
	GradientBoosting (sans ESS)	GradientBoosting (avec ESS)	GradientBoosting (sans ESS)	GradientBoosting (avec ESS)
Score Validation (RMSE)	0,45	0,36	0,75	0,70
Score Test (RMSE)	0,39	0,31	0,71	0,66

- D'après ce tableau, on voit qu'il y a un impact non-négligeable de l'ENERGYSTARScore sur nos prédictions.
- Son utilisation semble donc très importante, il pourrait être intéressant d'évaluer (avec le métier) un ratio effort/gain pour cet indicateur afin de statuer définitivement sur son maintien.
- On pourrait également songer à mettre en place des solutions moins coûteuses pour obtenir des features permettant d'améliorer notre précision (questionnaire sur les sources d'énergies ?)

Merci de votre attention