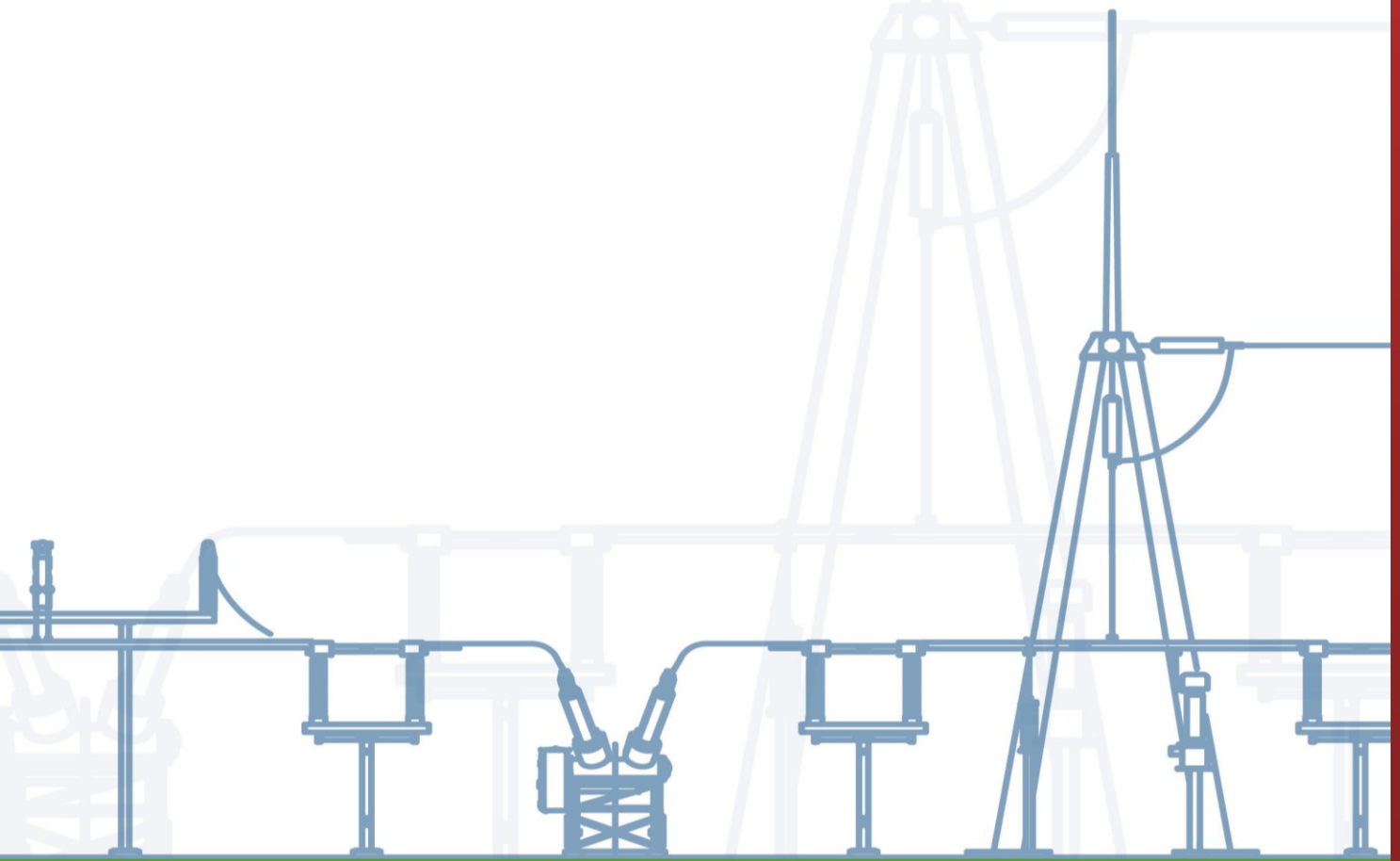
Seattle

ENERGY STAR



Prédire les émissions de CO2 et la consommation totale d'énergie.

Projet 4 – DS – Marc SELLAM - 09/2022

Sommaire :

- Page 1 Mission.
- Page 2 Informations fournies.
- Page 3 Les données fournies.
- Page 4 Première sélection de variables.
- Page 5 Les variables numériques.
- Page 6-8 Les variables catégorielles.
- Page 9-10 Préparation des données et création de 4 datasets.
- Page 11 Les variables des 4 datasets.
- Page 12 Variables explicatives et variables à prédire.
- Page 13 Choix des algorithmes.
- Page 14 Les modèles de prédiction : Régression linéaire.
- Page 15 Les modèles de prédiction : SVM.
- Page 16 Les modèles de prédiction : Arbre de décision.
- Page 17 Les modèles de prédiction : Adaboost.
- Page 18 Traitement des résultats.
- Page 19 Meilleurs résultats.
- Page 20 Conclusion.

Mission

- Je travaille pour la ville de Seattle. Pour atteindre son objectif de ville neutre en émissions de carbone en 2050, mon équipe s'intéresse de près à la consommation et aux émissions des bâtiments non destinés à l'habitation.

- A partir des relevés de la ville effectués en 2016, 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 en se basant sur les données structurelles des bâtiments (taille et usage des bâtiments, date de construction, situation géographique, ...).

- Evaluer l'intérêt de l'"ENERGY STAR Score" pour la prédiction d'émissions en l'intégrant dans la modélisation et juger de son intérêt.

Informations fournies

- Source des données :
  - https://data.seattle.gov/dataset/2016-Building-Energy-Benchmarking/2bpz-gwpy
  - Site internet Seattle Open Data dans lequel sont hébergées les données relatives à la ville de Seattle ainsi que la description de chacune des informations qui nous a été fournies.

- Téléchargement des données sur https://s3.eu-west-1.amazonaws.com/course.oc-static.com/projects/Data_Scientist_P4/2016_Building_Energy_Benchmarking.csv.

- Informations relatives à l'"ENERGY STAR Score".

Les données fournies

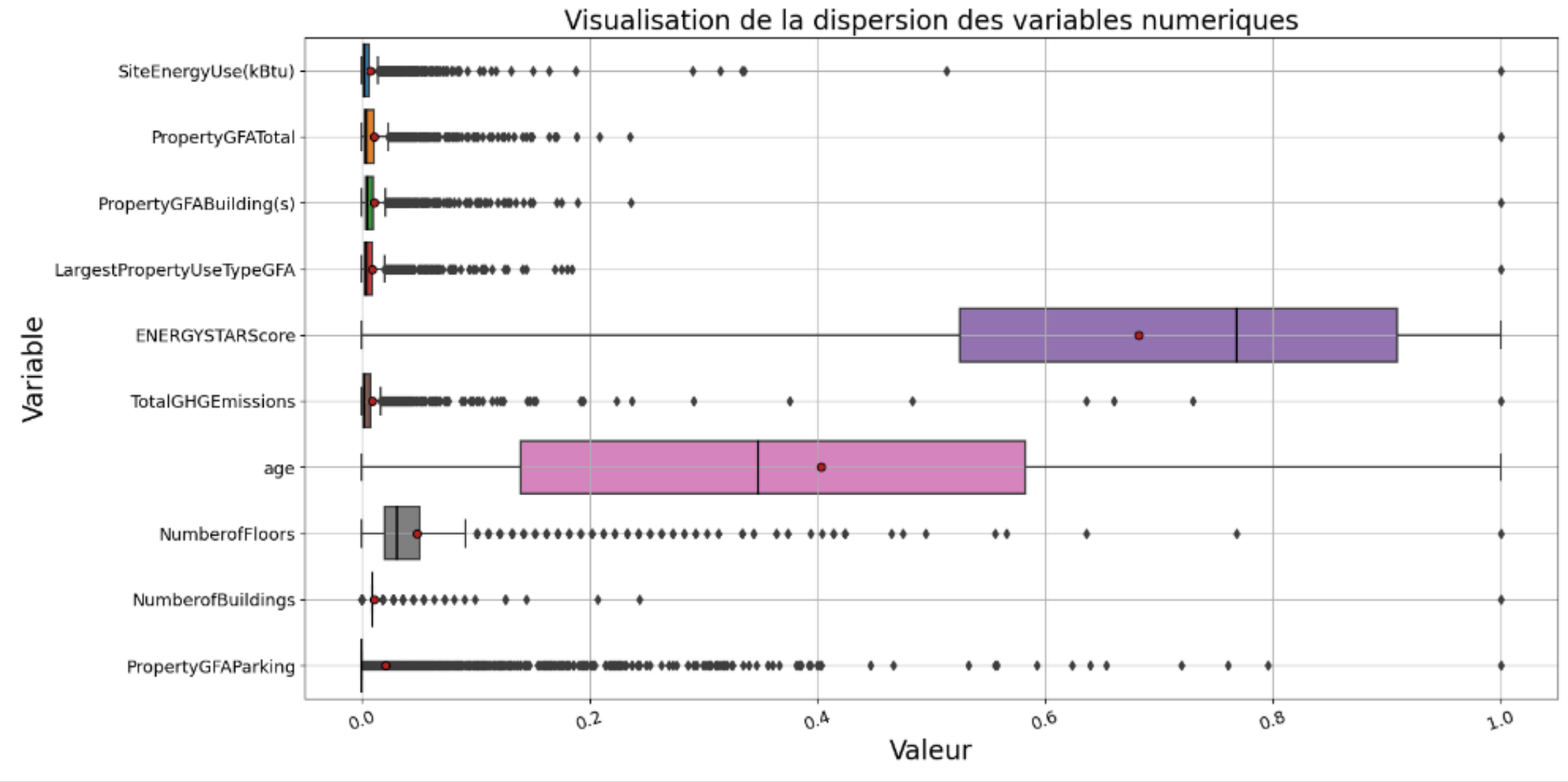
```
nb de lignes x nb de colonnes :  (3376, 46) .

% de données manquantes :
 Comments                          100.000000
Outlier                             99.052133
YearsENERGYSTARCertified            96.475118
ThirdLargestPropertyUseType         82.345972
ThirdLargestPropertyUseTypeGFA      82.345972
SecondLargestPropertyUseType        50.266588
SecondLargestPropertyUseTypeGFA     50.266588
ENERGYSTARScore                     24.970379
LargestPropertyUseTypeGFA            0.592417
LargestPropertyUseType               0.592417
ZipCode                              0.473934
ListOfAllPropertyUseTypes            0.266588
SourceEUIWN(kBtu/sf)                 0.266588
SourceEUI(kBtu/sf)                   0.266588
Electricity(kWh)                     0.266588
Electricity(kBtu)                    0.266588
NaturalGas(therms)                   0.266588
NaturalGas(kBtu)                     0.266588
TotalGHGEmissions                    0.266588
SteamUse(kBtu)                       0.266588
GHGEmissionsIntensity                0.266588
NumberofBuildings                    0.236967
SiteEUI(kBtu/sf)                     0.207346
SiteEUIWN(kBtu/sf)                   0.177725
SiteEnergyUseWN(kBtu)                0.177725
SiteEnergyUse(kBtu)                  0.148104
TaxParcelIdentificationNumber        0.000000
BuildingType                         0.000000
PrimaryPropertyType                  0.000000
ComplianceStatus                     0.000000
```

- 3376 bâtiments avec 46 variables que l'on peut regrouper:
  - Données de situation
  - Données structurelles ainsi que leur destination
  - Données énergétiques
  - Données Energy score
  - Données d'émission de gaz a effet de serre
  - Informations diverses

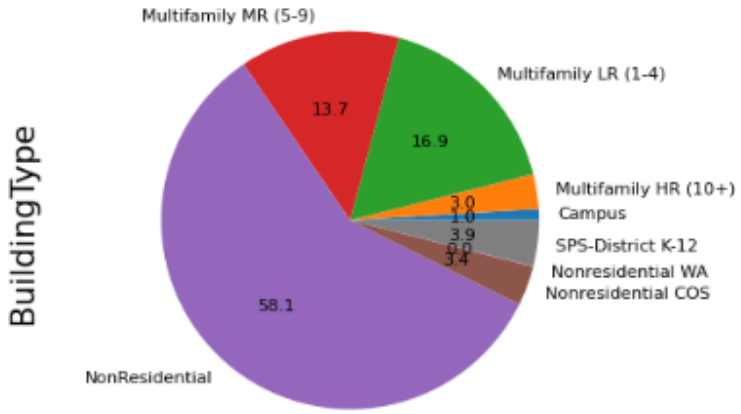
Première sélection de variables

	% manquants	type	unique	échantillon des 5 premieres valeurs
BuildingType	0.000000	object	8	['NonResidential', 'NonResidential', 'NonResidential', 'NonResidential', 'NonResidential']
PropertyGFAParking	0.000000	int64	496	[0, 15064, 196718, 0, 62000]
PropertyGFATotal	0.000000	int64	3195	[88434, 103566, 956110, 61320, 175580]
NumberofFloors	0.000000	int64	50	[12, 11, 41, 10, 18]
PropertyGFABuilding(s)	0.000000	int64	3193	[88434, 88502, 759392, 61320, 113580]
Neighborhood	0.000000	object	19	['DOWNTOWN', 'DOWNTOWN', 'DOWNTOWN', 'DOWNTOWN', 'DOWNTOWN']
PrimaryPropertyType	0.000000	object	24	['Hotel', 'Hotel', 'Hotel', 'Hotel', 'Hotel']
YearBuilt	0.000000	int64	113	[1927, 1996, 1969, 1926, 1980]
SiteEnergyUse(kBtu)	0.148104	float64	3355	[7226362.5, 8387933.0, 72587024.0, 6794584.0, 14172606.0]
NumberofBuildings	0.236967	float64	18	[1.0, 1.0, 1.0, 1.0, 1.0]
ListOfAllPropertyUseTypes	0.266588	object	467	['Hotel', 'Hotel, Parking, Restaurant', 'Hotel', 'Hotel', 'Hotel, Parking, Swimming Pool']
TotalGHGEmissions	0.266588	float64	2819	[249.98, 295.86, 2089.28, 286.43, 505.01]
ZipCode	0.473934	float64	56	[98101.0, 98101.0, 98101.0, 98101.0, 98121.0]
LargestPropertyUseType	0.592417	object	57	['Hotel', 'Hotel', 'Hotel', 'Hotel', 'Hotel']
LargestPropertyUseTypeGFA	0.592417	float64	3123	[88434.0, 83880.0, 756493.0, 61320.0, 123445.0]
ENERGYSTARScore	24.970379	float64	101	[60.0, 61.0, 43.0, 56.0, 75.0]
SecondLargestPropertyUseType	50.266588	object	51	[nan, 'Parking', nan, nan, 'Parking']
SecondLargestPropertyUseTypeGFA	50.266588	float64	1353	[nan, 15064.0, nan, nan, 68009.0]
ThirdLargestPropertyUseType	82.345972	object	45	[nan, 'Restaurant', nan, nan, 'Swimming Pool']
ThirdLargestPropertyUseTypeGFA	82.345972	float64	502	[nan, 4622.0, nan, nan, 0.0]

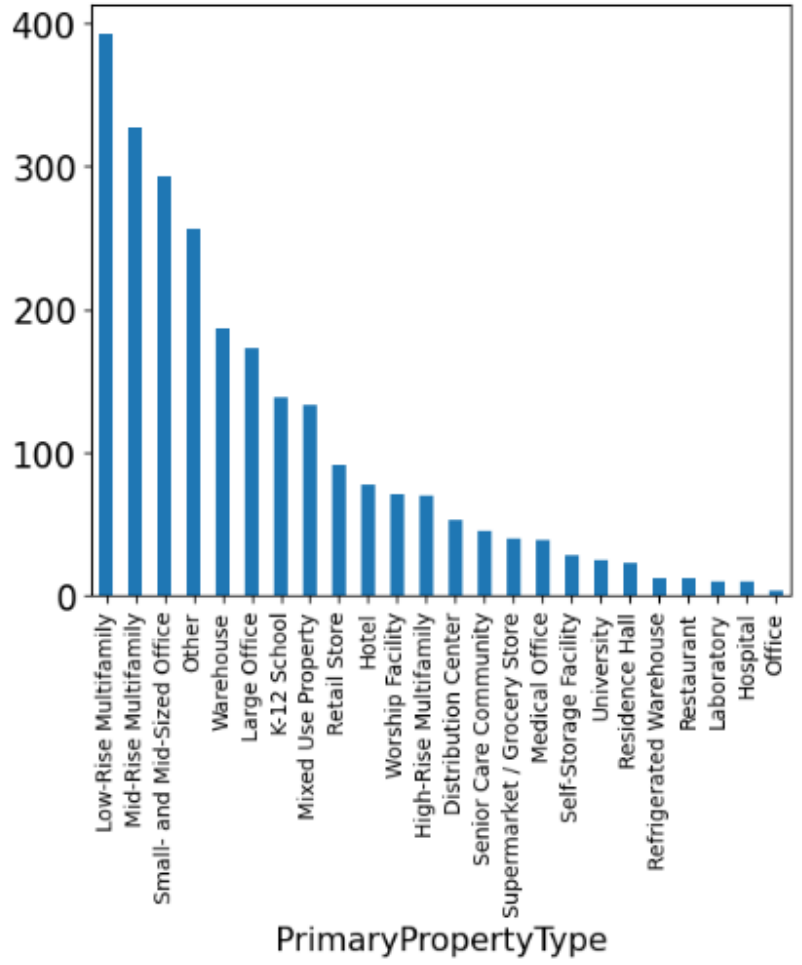
Les variables numériques

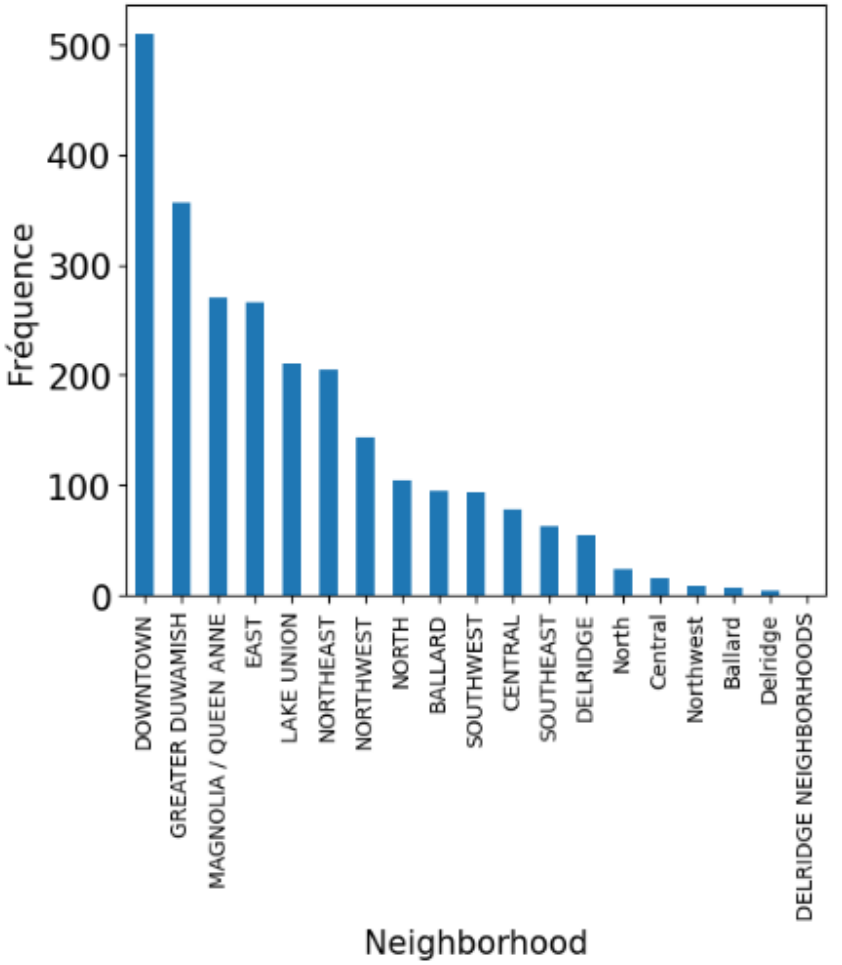


Les variables catégorielles



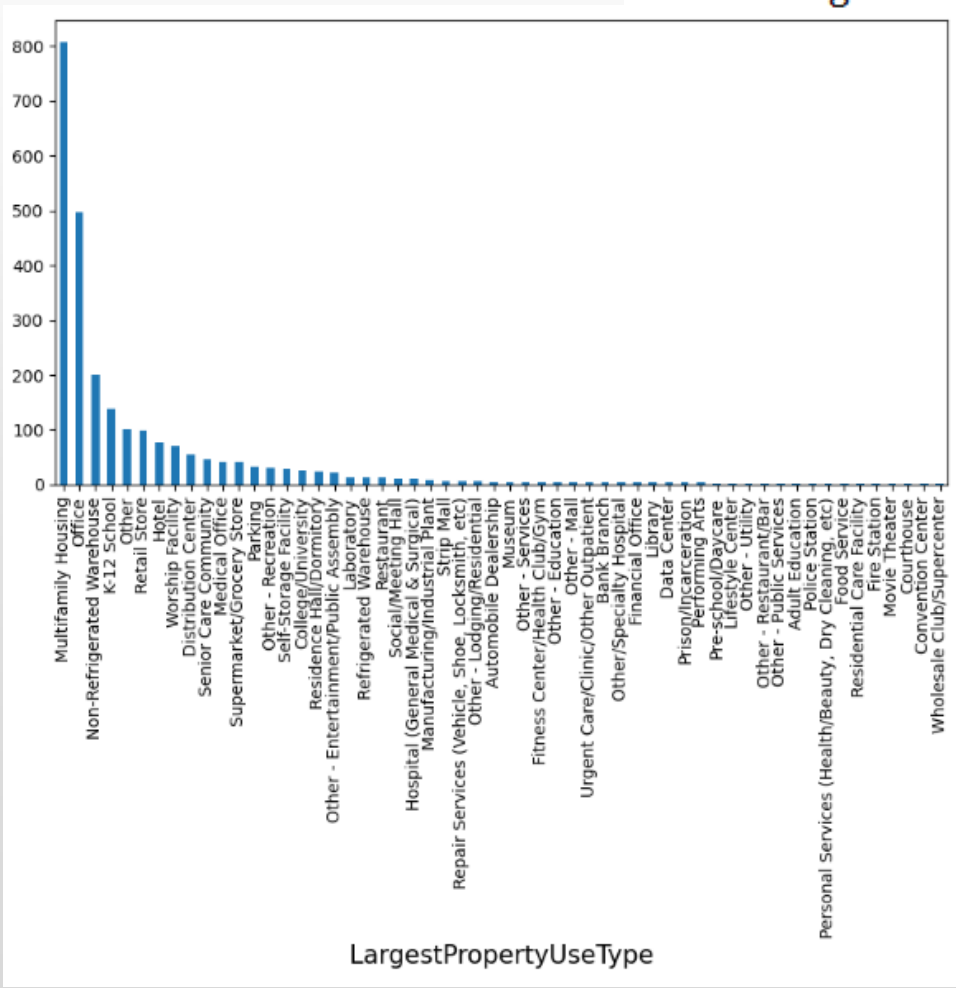
BuildingType	8
PrimaryPropertyType	24
Neighborhood	19

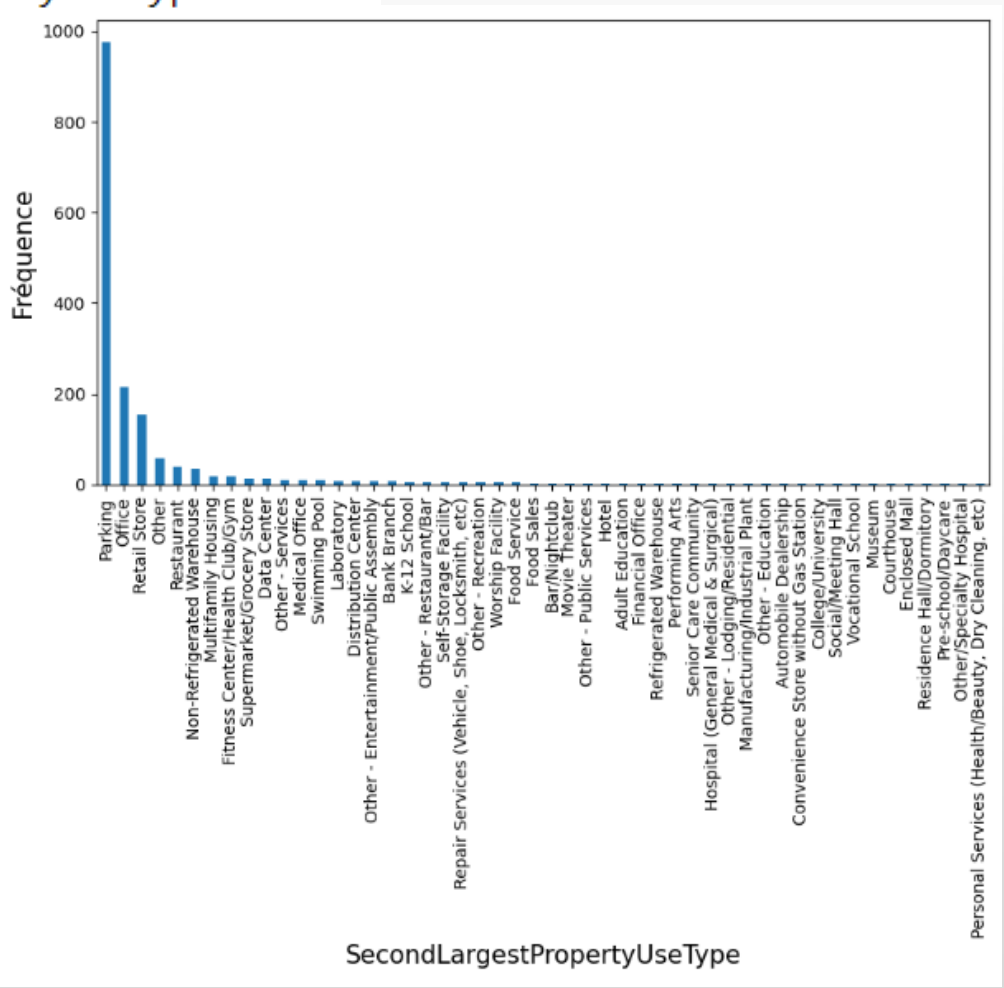




Les variables catégorielles

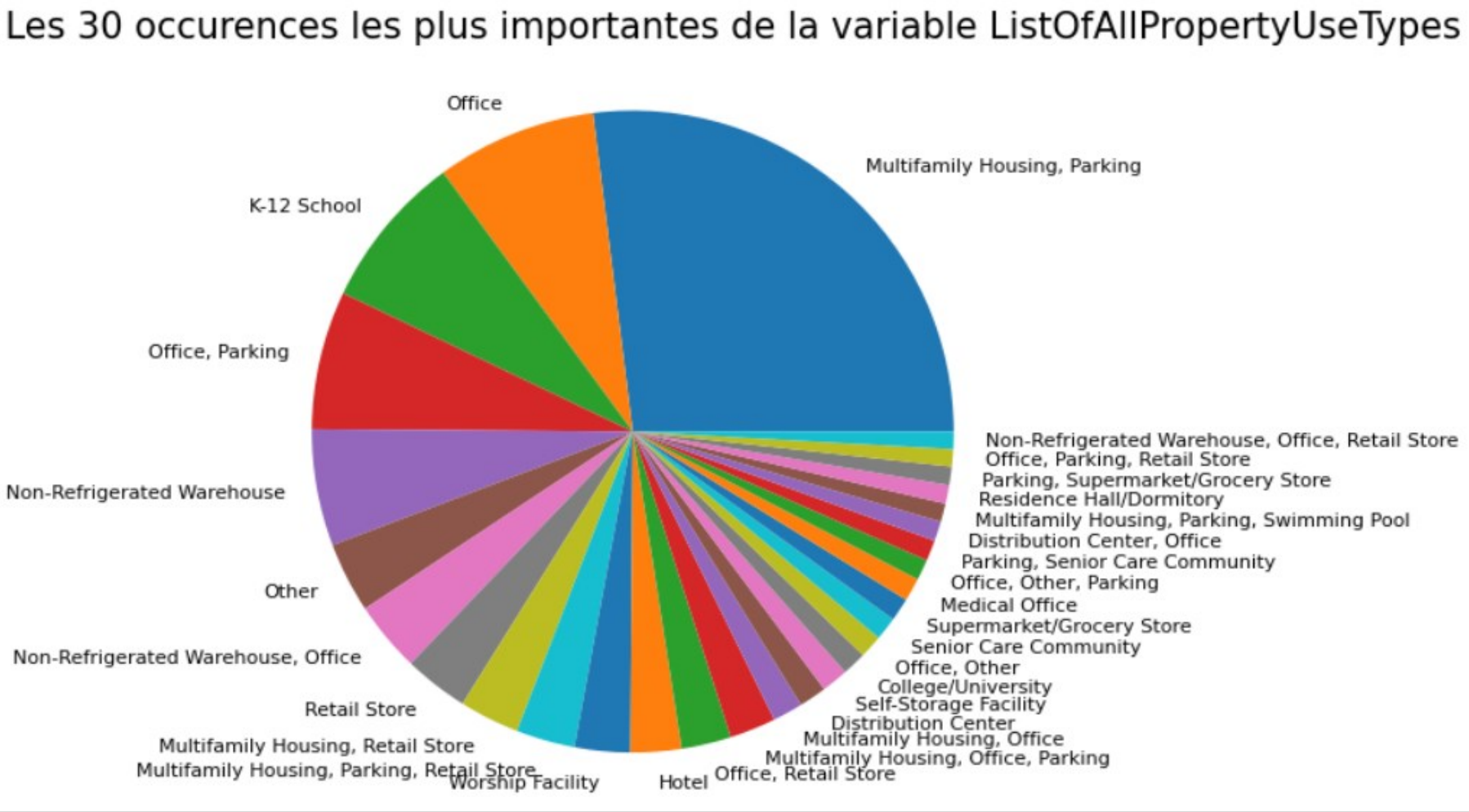
```
LargestPropertyUseType          56
SecondLargestPropertyUseType    50
```





Les variables catégorielles

ListOfAllPropertyUseTypes	465



Préparation des données

Retrait des observation incohérantes

```
print("nombre de lignes effacees dont la valeur number of building = 0",20*'_',df[df['NumberofBuildings'] == 0.0].shape[0])
df = df[df['NumberofBuildings'] != 0.0]
print("nombre de lignes effacees dont la valeur NumberofFloors = 0",20*'_',df[df['NumberofFloors']==0].shape[0])
df = df[df['NumberofFloors'] != 0.0]
print("nombre de lignes effacees dont la valeur TotalGHGEmissions < 0",20*'_',df[df['TotalGHGEmissions']<0].shape[0])
df = df[df['TotalGHGEmissions']>=0]
```

```
nombre de lignes effacees dont la valeur number of building = 0 ____________________ 71
nombre de lignes effacees dont la valeur NumberofFloors = 0 ____________________ 15
nombre de lignes effacees dont la valeur TotalGHGEmissions < 0 ____________________ 1
```

Création de df1

```
df = df.dropna(subset=['NumberofBuildings','TotalGHGEmissions','SiteEnergyUse(kBtu)'],how='any')
df1 = df.copy()
df = df.dropna(subset=['LargestPropertyUseTypeGFA','LargestPropertyUseType'],how='any')
df1 = df1.drop(['LargestPropertyUseType','LargestPropertyUseTypeGFA'],axis=1).copy()
```

Préparation des données et création de 4 datasets

Comptage des valeurs uniques par variable categoriel

```
list_col_cat = df.select_dtypes(exclude=['int64','float64']).columns.tolist()
for column in df[list_col_cat]:
    print(column,len(df[column].unique()))
```

```
BuildingType 8
PrimaryPropertyType 24
Neighborhood 19
ListOfAllPropertyUseTypes 445
LargestPropertyUseType 54
SecondLargestPropertyUseType 51
```

Création de df3 (1595,54) par encodage des 3 variables catégoriels de df1.

Création de df4 (1582,108) par encodage des 4 variables catégoriels de df.

Création de df_log3 (1059,54) par passage en log des variables non encodées de df3 (excepté ENERGYSTARScore).

Création de df_log4 (1054,108) par passage en log des variables non encodées de df4 (excepté ENERGYSTARScore).

Les variables des 4 datasets

dataset	variables numériques	variables catégorielles encodées
df3 (1595,54)	'NumberofBuildings' 'NumberofFloors' 'PropertyGFATotal' 'PropertyGFAParking' 'PropertyGFABuilding(s)' 'age' 'ENERGYSTARScore' 'SiteEnergyUse(kBtu)' 'TotalGHGEmissions'	'BuildingType' 'PrimaryPropertyType' 'Neighborhood'
df_log3 (1059,54) **Passage en log des variables numériques (sauf ENERGYSTARScore)**		
df4 (1582,108)	'NumberofBuildings' 'NumberofFloors' 'PropertyGFATotal' 'PropertyGFAParking' 'PropertyGFABuilding(s)' 'LargestPropertyUseTypeGFA' 'age' 'ENERGYSTARScore' 'SiteEnergyUse(kBtu)' 'TotalGHGEmissions'	'BuildingType' 'PrimaryPropertyType' 'Neighborhood' 'LargestPropertyUseType'
df_log4 (1054,108) **Passage en log des variables numériques (sauf ENERGYSTARScore)**		

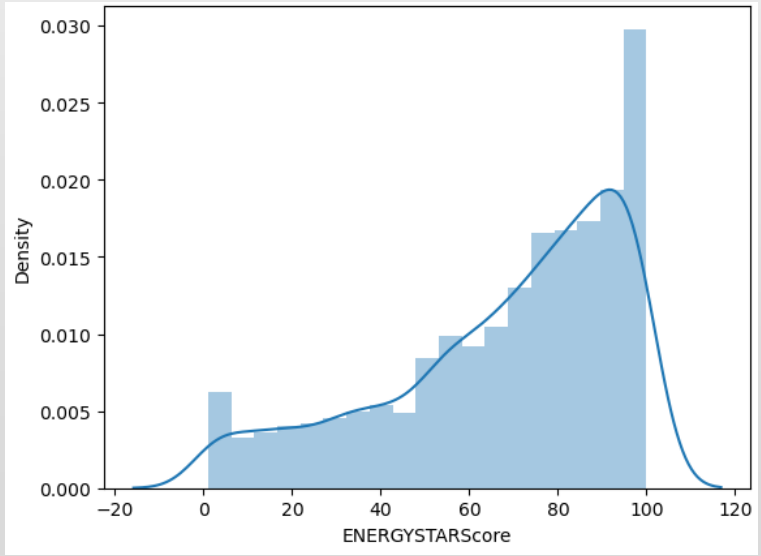
Variables en entrée: structurelles + localisation + âge

Variable en entrée dont l'intérêt est a évaluer

Variables en sortie

Variables explicatives et variables cibles

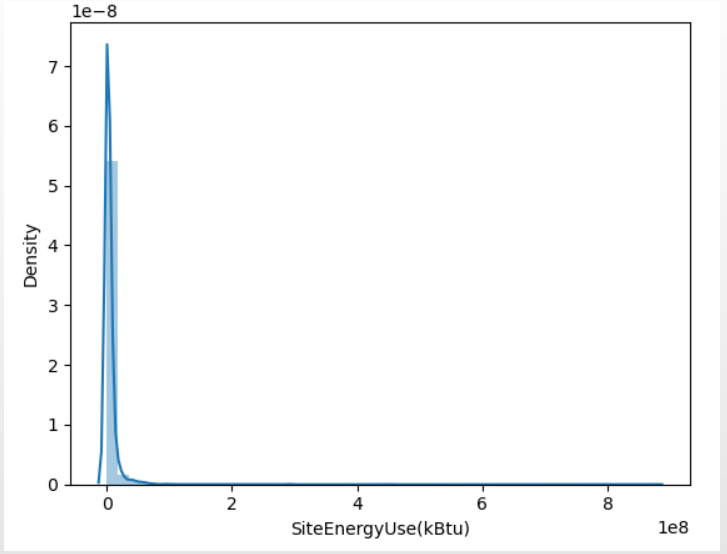
Variables en entrée :
structurelles
+ localisation
+ âge



Variable en entrée dont l'intérêt est a évaluer

Variables à prédire

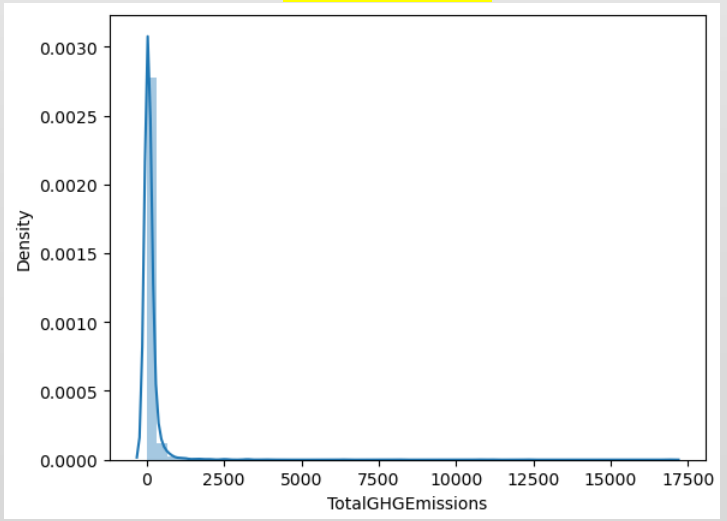
1



Variable en sortie

2

3



Choix des algorithmes

Variables explicatives

4 datasets

Variables structurelles
+ localisation
+ âge

Variables structurelles
+ localisation
+ âge

Variables structurelles
+ localisation
+ âge
+ENERGYSTARScore
(Variable en entrée dont l'intérêt est a évaluer)

Entrainement des 4 algorithmes avec GridSearchCV

- **Régression linéaire**
- **SVM**
- **Arbre de décision**
- **Adaboost**

Variables cibles

SiteEnergyUse(kBtu) 1

TotalGHGEmissions 2

TotalGHGEmissions 3

Les modèles de prédiction
Régression linéaire

```
from sklearn.linear_model import LinearRegression
model1 = LinearRegression()
params1 ={"fit_intercept":[True]}
```

```
results1 = bestmod(model1,params1)
```

```
RESULTATS GRID SEARCH :
LinearRegression

temps:_________________ 0:00:05.303484
```

```
results1
```

	dataset_name	dataset_shape	target	best_sc	r2_test	r2_train	best_pa	model_name
0	dataframe_3encoded	(1595, 54)	TotalGHGEmissions	-3.154019e+23	-1.070067e+25	0.480	{'fit_intercept': True}	LinearRegression
1	dataframe_3encoded	(1059, 54)	TotalGHGEm(+in ES)	-1.388625e+24	6.200000e-02	0.638	{'fit_intercept': True}	LinearRegression
2	dataframe_3encoded	(1595, 54)	SiteEnergyUse(kBtu)	-1.108494e+23	-2.322071e+25	0.552	{'fit_intercept': True}	LinearRegression
3	dataframe_4encoded	(1582, 108)	TotalGHGEmissions	-2.565869e+27	-4.628387e+24	0.452	{'fit_intercept': True}	LinearRegression
4	dataframe_4encoded	(1054, 108)	TotalGHGEm(+in ES)	-4.100871e+26	-8.557255e+25	0.771	{'fit_intercept': True}	LinearRegression
5	dataframe_4encoded	(1582, 108)	SiteEnergyUse(kBtu)	-1.186818e+27	-5.476758e+23	0.581	{'fit_intercept': True}	LinearRegression
6	dataframe_3encoded_log	(1059, 54)	TotalGHGEmissions	-2.950671e+25	5.680000e-01	0.610	{'fit_intercept': True}	LinearRegression
7	dataframe_3encoded_log	(1059, 54)	TotalGHGEm(+in ES)	-1.007857e+24	6.280000e-01	0.663	{'fit_intercept': True}	LinearRegression
8	dataframe_3encoded_log	(1059, 54)	SiteEnergyUse(kBtu)	-1.417615e+25	7.710000e-01	0.694	{'fit_intercept': True}	LinearRegression
9	dataframe_4encoded_log	(1054, 108)	TotalGHGEmissions	-6.680353e+26	-2.322651e+26	0.590	{'fit_intercept': True}	LinearRegression
10	dataframe_4encoded_log	(1054, 108)	TotalGHGEm(+in ES)	-2.317223e+25	-5.843835e+25	0.660	{'fit_intercept': True}	LinearRegression
11	dataframe_4encoded_log	(1054, 108)	SiteEnergyUse(kBtu)	-8.878054e+25	-3.819376e+25	0.685	{'fit_intercept': True}	LinearRegression

Les modèles de prédiction
SVM

```
from sklearn.svm import SVR
model2 = SVR()
params2 = {'C':np.arange(10,85, 5),
           'kernel':['linear','rbf','poly'],
            'gamma':['scale','auto'],
            'degree':[1,2,3],
           'coef0':np.arange(0.1,0.9, 0.1)}
```

```
results2 = bestmod(model2,params2)
```

```
RESULTATS GRID SEARCH :
SVR

temps:_________________ 2:21:13.675436
```

```
results2
```

	dataset_name	dataset_shape	target	best_sc	r2_test	r2_train	best_pa	model_name
0	dataframe_3encoded	(1595, 54)	TotalGHGEmissions	0.465	0.711	0.538	{'C': 80, 'coef0': 0.8, 'degree': 3, 'gamma': 'auto', 'kernel': 'poly'}	SVR
1	dataframe_3encoded	(1059, 54)	TotalGHGEm(+in ES)	0.595	0.494	0.983	{'C': 80, 'coef0': 0.8, 'degree': 3, 'gamma': 'scale', 'kernel': 'poly'}	SVR
2	dataframe_3encoded	(1595, 54)	SiteEnergyUse(kBtu)	-0.080	-0.067	-0.046	{'C': 80, 'coef0': 0.1, 'degree': 1, 'gamma': 'scale', 'kernel': 'linear'}	SVR
3	dataframe_4encoded	(1582, 108)	TotalGHGEmissions	0.359	0.677	0.313	{'C': 80, 'coef0': 0.8, 'degree': 2, 'gamma': 'scale', 'kernel': 'poly'}	SVR
4	dataframe_4encoded	(1054, 108)	TotalGHGEm(+in ES)	0.424	0.703	0.552	{'C': 15, 'coef0': 0.5, 'degree': 3, 'gamma': 'auto', 'kernel': 'poly'}	SVR
5	dataframe_4encoded	(1582, 108)	SiteEnergyUse(kBtu)	-0.073	-0.061	-0.045	{'C': 80, 'coef0': 0.1, 'degree': 1, 'gamma': 'scale', 'kernel': 'linear'}	SVR
6	dataframe_3encoded_log	(1059, 54)	TotalGHGEmissions	0.526	0.536	0.585	{'C': 10, 'coef0': 0.5, 'degree': 1, 'gamma': 'auto', 'kernel': 'poly'}	SVR
7	dataframe_3encoded_log	(1059, 54)	TotalGHGEm(+in ES)	0.584	0.607	0.642	{'C': 10, 'coef0': 0.2, 'degree': 1, 'gamma': 'auto', 'kernel': 'poly'}	SVR
8	dataframe_3encoded_log	(1059, 54)	SiteEnergyUse(kBtu)	0.681	0.771	0.675	{'C': 10, 'coef0': 0.5, 'degree': 1, 'gamma': 'scale', 'kernel': 'poly'}	SVR
9	dataframe_4encoded_log	(1054, 108)	TotalGHGEmissions	0.531	0.571	0.595	{'C': 10, 'coef0': 0.4, 'degree': 1, 'gamma': 'auto', 'kernel': 'poly'}	SVR
10	dataframe_4encoded_log	(1054, 108)	TotalGHGEm(+in ES)	0.598	0.635	0.661	{'C': 10, 'coef0': 0.5, 'degree': 1, 'gamma': 'auto', 'kernel': 'poly'}	SVR
11	dataframe_4encoded_log	(1054, 108)	SiteEnergyUse(kBtu)	0.678	0.804	0.676	{'C': 10, 'coef0': 0.1, 'degree': 1, 'gamma': 'auto', 'kernel': 'poly'}	SVR

Les modèles de prédiction
L'arbre de décision

```
from sklearn.tree import DecisionTreeRegressor
model3 = DecisionTreeRegressor()
params3 = {
    'criterion':['absolute_error'],
    'splitter':['best'],
    'max_leaf_nodes': np.arange(1,15, 1),
    'ccp_alpha': np.arange(0.1,0.6, 0.1),
    'max_depth' : np.arange(2,22, 2)
    }
```

```
results3 = bestmod(model3,params3)
```

```
results3
```

	dataset_name	dataset_shape	target	best_sc	r2_test	r2_train	best_pa	model_name
0	dataframe_3encoded	(1595, 54)	TotalGHGEmissions	0.346	0.323	0.476	{'ccp_alpha': 0.4, 'criterion': 'absolute_error', 'max_depth': 4, 'max_leaf_nodes': 12, 'splitter': 'best'}	DecisionTreeRegressor
1	dataframe_3encoded	(1059, 54)	TotalGHGEm(+inES)	0.232	0.392	0.530	{'ccp_alpha': 0.30000000000000004, 'criterion': 'absolute_error', 'max_depth': 16, 'max_leaf_nodes': 13, 'splitter': 'best'}	DecisionTreeRegressor
2	dataframe_3encoded	(1595, 54)	SiteEnergyUse(kBtu)	0.483	0.549	0.466	{'ccp_alpha': 0.1, 'criterion': 'absolute_error', 'max_depth': 4, 'max_leaf_nodes': 4, 'splitter': 'best'}	DecisionTreeRegressor
3	dataframe_4encoded	(1582, 108)	TotalGHGEmissions	0.219	0.479	0.169	{'ccp_alpha': 0.1, 'criterion': 'absolute_error', 'max_depth': 2, 'max_leaf_nodes': 6, 'splitter': 'best'}	DecisionTreeRegressor
4	dataframe_4encoded	(1054, 108)	TotalGHGEm(+inES)	0.342	-0.230	0.412	{'ccp_alpha': 0.1, 'criterion': 'absolute_error', 'max_depth': 2, 'max_leaf_nodes': 2, 'splitter': 'best'}	DecisionTreeRegressor
5	dataframe_4encoded	(1582, 108)	SiteEnergyUse(kBtu)	0.403	0.314	0.312	{'ccp_alpha': 0.1, 'criterion': 'absolute_error', 'max_depth': 2, 'max_leaf_nodes': 4, 'splitter': 'best'}	DecisionTreeRegressor
6	dataframe_3encoded_log	(1059, 54)	TotalGHGEmissions	0.252	0.357	0.272	{'ccp_alpha': 0.1, 'criterion': 'absolute_error', 'max_depth': 2, 'max_leaf_nodes': 2, 'splitter': 'best'}	DecisionTreeRegressor
7	dataframe_3encoded_log	(1059, 54)	TotalGHGEm(+inES)	0.252	0.357	0.272	{'ccp_alpha': 0.1, 'criterion': 'absolute_error', 'max_depth': 2, 'max_leaf_nodes': 2, 'splitter': 'best'}	DecisionTreeRegressor
8	dataframe_3encoded_log	(1059, 54)	SiteEnergyUse(kBtu)	0.398	0.493	0.377	{'ccp_alpha': 0.1, 'criterion': 'absolute_error', 'max_depth': 2, 'max_leaf_nodes': 3, 'splitter': 'best'}	DecisionTreeRegressor
9	dataframe_4encoded_log	(1054, 108)	TotalGHGEmissions	0.260	0.301	0.283	{'ccp_alpha': 0.1, 'criterion': 'absolute_error', 'max_depth': 2, 'max_leaf_nodes': 2, 'splitter': 'best'}	DecisionTreeRegressor
10	dataframe_4encoded_log	(1054, 108)	TotalGHGEm(+inES)	0.260	0.301	0.283	{'ccp_alpha': 0.1, 'criterion': 'absolute_error', 'max_depth': 2, 'max_leaf_nodes': 2, 'splitter': 'best'}	DecisionTreeRegressor
11	dataframe_4encoded_log	(1054, 108)	SiteEnergyUse(kBtu)	0.407	0.467	0.392	{'ccp_alpha': 0.1, 'criterion': 'absolute_error', 'max_depth': 2, 'max_leaf_nodes': 2, 'splitter': 'best'}	DecisionTreeRegressor

Les modèles de prédiction
Adaboost

```
from sklearn.ensemble import AdaBoostRegressor
from sklearn.svm import SVR
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LinearRegression

model4 = AdaBoostRegressor()
params4 = {'base_estimator': [RandomForestRegressor(), LinearRegression(), SVR()],
           'learning_rate': [0.01,0.05,0.1,0.3,0.5,0.7, 1],
           'n_estimators': np.arange(10,300,10)}
```

```
results4 = bestmod(model4,params4)
```

```
RESULTATS GRID SEARCH :
AdaBoostRegressor
temps:_________________ 19:55:26.190882
```

```
results4
```

	dataset_name	dataset_shape	target	best_sc	r2_test	r2_train	best_pa	model_name
0	dataframe_3encoded	(1595, 54)	TotalGHGEmissions	0.274	0.676	0.899	{'base_estimator': RandomForestRegressor(), 'learning_rate': 0.01, 'n_estimators': 10}	AdaBoostRegressor
1	dataframe_3encoded	(1059, 54)	TotalGHGEm(+inES)	0.410	0.753	0.998	{'base_estimator': RandomForestRegressor(), 'learning_rate': 0.7, 'n_estimators': 10}	AdaBoostRegressor
2	dataframe_3encoded	(1595, 54)	SiteEnergyUse(kBtu)	0.374	0.620	0.915	{'base_estimator': RandomForestRegressor(), 'learning_rate': 0.01, 'n_estimators': 10}	AdaBoostRegressor
3	dataframe_4encoded	(1582, 108)	TotalGHGEmissions	0.137	0.669	0.543	{'base_estimator': RandomForestRegressor(), 'learning_rate': 0.01, 'n_estimators': 10}	AdaBoostRegressor
4	dataframe_4encoded	(1054, 108)	TotalGHGEm(+inES)	-0.012	-0.029	-0.011	{'base_estimator': SVR(), 'learning_rate': 0.3, 'n_estimators': 130}	AdaBoostRegressor
5	dataframe_4encoded	(1582, 108)	SiteEnergyUse(kBtu)	0.467	0.586	0.690	{'base_estimator': RandomForestRegressor(), 'learning_rate': 0.01, 'n_estimators': 10}	AdaBoostRegressor
6	dataframe_3encoded_log	(1059, 54)	TotalGHGEmissions	0.524	0.577	0.936	{'base_estimator': RandomForestRegressor(), 'learning_rate': 0.01, 'n_estimators': 90}	AdaBoostRegressor
7	dataframe_3encoded_log	(1059, 54)	TotalGHGEm(+inES)	0.602	0.647	0.944	{'base_estimator': RandomForestRegressor(), 'learning_rate': 0.01, 'n_estimators': 80}	AdaBoostRegressor
8	dataframe_3encoded_log	(1059, 54)	SiteEnergyUse(kBtu)	0.626	0.771	0.960	{'base_estimator': RandomForestRegressor(), 'learning_rate': 0.01, 'n_estimators': 60}	AdaBoostRegressor
9	dataframe_4encoded_log	(1054, 108)	TotalGHGEmissions	0.558	0.557	0.949	{'base_estimator': RandomForestRegressor(), 'learning_rate': 0.05, 'n_estimators': 50}	AdaBoostRegressor
10	dataframe_4encoded_log	(1054, 108)	TotalGHGEm(+inES)	0.605	0.659	0.966	{'base_estimator': RandomForestRegressor(), 'learning_rate': 0.1, 'n_estimators': 80}	AdaBoostRegressor
11	dataframe_4encoded_log	(1054, 108)	SiteEnergyUse(kBtu)	0.671	0.763	0.977	{'base_estimator': RandomForestRegressor(), 'learning_rate': 1, 'n_estimators': 10}	AdaBoostRegressor

Traitement des résultats

Filtrage des rersultats pour lesquels r2_train est superieur au r2_test (creation du dataframe equilib)

```
equilib = results_all34[results_all34.r2_test < results_all34.r2_train]
```

Fonction qui permet d'afficher les meilleurs resultats par target

```
def best3(dataframe):
    res1 = []
    for target in dataframe.target.unique():
        subset = dataframe[dataframe.target == target]
        res = subset[subset.r2_test == subset.r2_test.max()]
        res1.append(res)
    final = pd.concat(res1)
    return final
```

Meilleurs résultats

	dataset_name	dataset_shape	target	best_sc	r2_test	r2_train	best_pa	model_name
36	dataframe_3encoded	(1595, 54)	TotalGHGEmissions	0.274	0.676	0.899	{'base_estimator': RandomForestRegressor(), 'learning_rate': 0.01, 'n_estimators': 10}	AdaBoostRegressor
37	dataframe_3encoded	(1059, 54)	TotalGHGEm(+in ES)	0.410	0.753	0.998	{'base_estimator': RandomForestRegressor(), 'learning_rate': 0.7, 'n_estimators': 10}	AdaBoostRegressor
44	dataframe_3encoded_log	(1059, 54)	SiteEnergyUse(kBtu)	0.626	0.771	0.960	{'base_estimator': RandomForestRegressor(), 'learning_rate': 0.01, 'n_estimators': 60}	AdaBoostRegressor

Conclusion

Les algorithmes ne permettent pas d'obtenir une évaluation exacte de la consommation énergétique ou d'émission de gaz a effet de serre à partir de données structurelles, cependant ils permettent d'obtenir un ordre de grandeur non négligeable pour cette évaluation.

L'énergie score peut s'avérer utile pour améliorer la prédiction de gaz a effet de serre.