I

Energy Consumption Prediction

Mars 20, 2022

Sommaire

III

Description du Projet

Mettre en place un modèle prédictif sur la consommation d'energie

Objectifs :

1. Réalisez une analyse univariée et bivariée
2. Réalisez une ACP puis une matrice de corrélation
3. Identifier les variables importantes
4. Identifier les variables nulles, les outliers, etc
5. Tester différents modèles de prédiction afin de répondre au mieux à la problématique.
6. Compare ces algorithmes, puis optimiser les hyperparamètres du modèle sélectionné

IV

Let's BEGIN

V

Import des librairies utiles et brêve description du dataset

Describe the dataset

	OSEBuildingID	Latitude	Longitude	YearBuilt	NumberofBuildings	NumberofFloors	PropertyGFATotal	PropertyGFAParkin
count	3376.000000	3376.000000	3376.000000	3376.000000	3368.000000	3376.000000	3.376000e+03	3376.00000
mean	21208.991114	47.624033	-122.334795	1968.573164	1.106888	4.709123	9.483354e+04	8001.52606
std	12223.757015	0.047758	0.027203	33.088156	2.108402	5.494465	2.188376e+05	32326.72392
min	1.000000	47.499170	-122.414250	1900.000000	0.000000	0.000000	1.128500e+04	0.00000
25%	19990.750000	47.599860	-122.350662	1948.000000	1.000000	2.000000	2.848700e+04	0.00000
50%	23112.000000	47.618675	-122.332495	1975.000000	1.000000	4.000000	4.417500e+04	0.00000
75%	25994.250000	47.657115	-122.319407	1997.000000	1.000000	5.000000	9.099200e+04	0.00000
max	50226.000000	47.733870	-122.220966	2015.000000	111.000000	99.000000	9.320156e+06	512608.00000

Le dataset est composés de 3376 ligne avec 17 features.

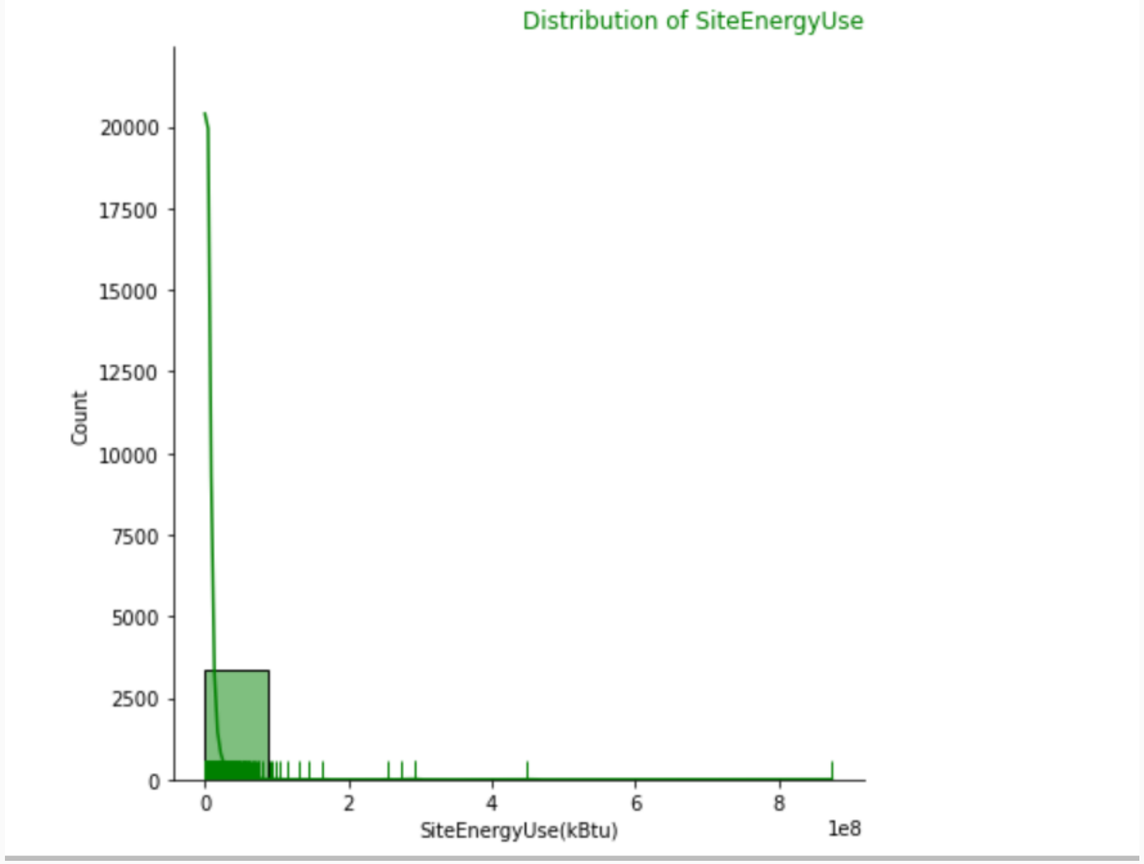
VI

Data Exploration

Univariate Analysis

La **Data Exploration** est la première étape de l'analyse des données. Elle permet d'explorer et de visualiser les données afin de découvrir des informations dès le départ ou d'identifier des zones ou des modèles à approfondir.

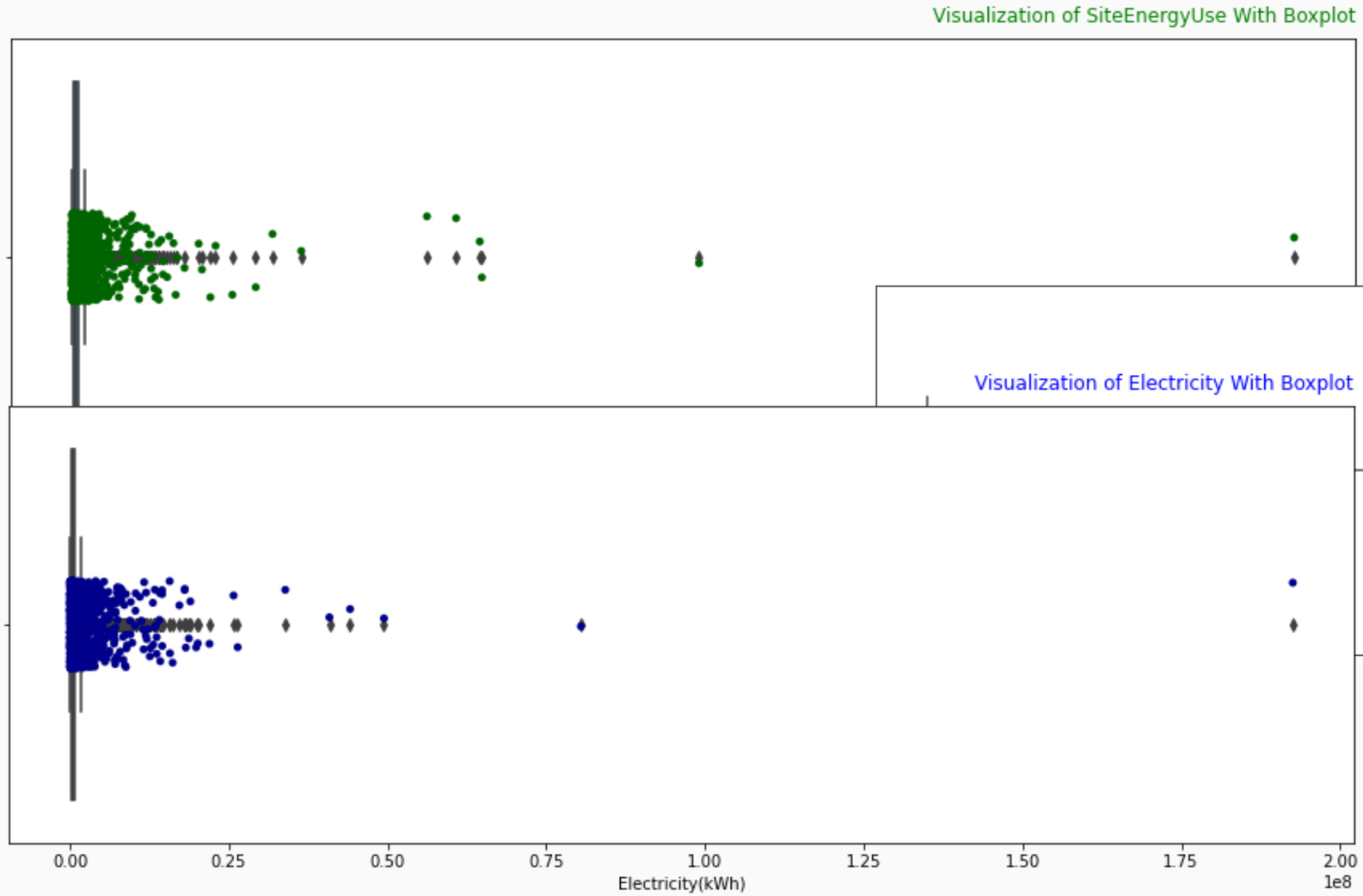
Cette figure représente la distribution de la variable SiteEnergyUse (la variable à prédire).

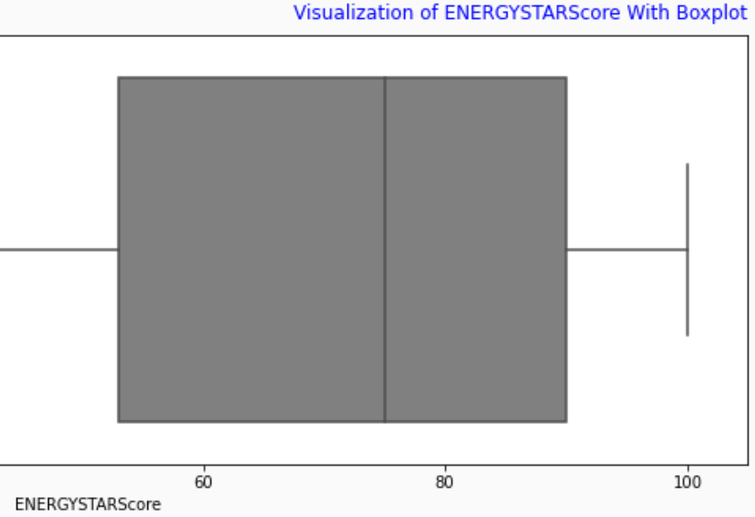


VII

Data Exploration

Univariate Analysis

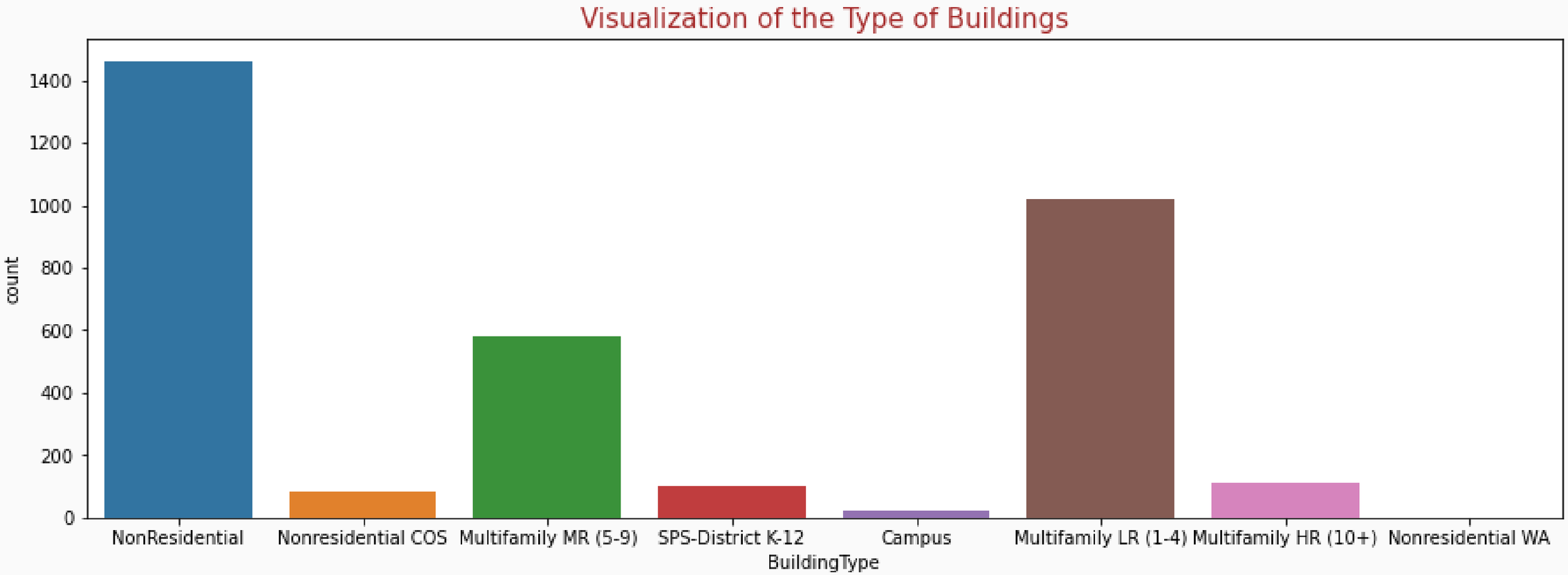




VIII

Data Exploration

Univariate Analysis

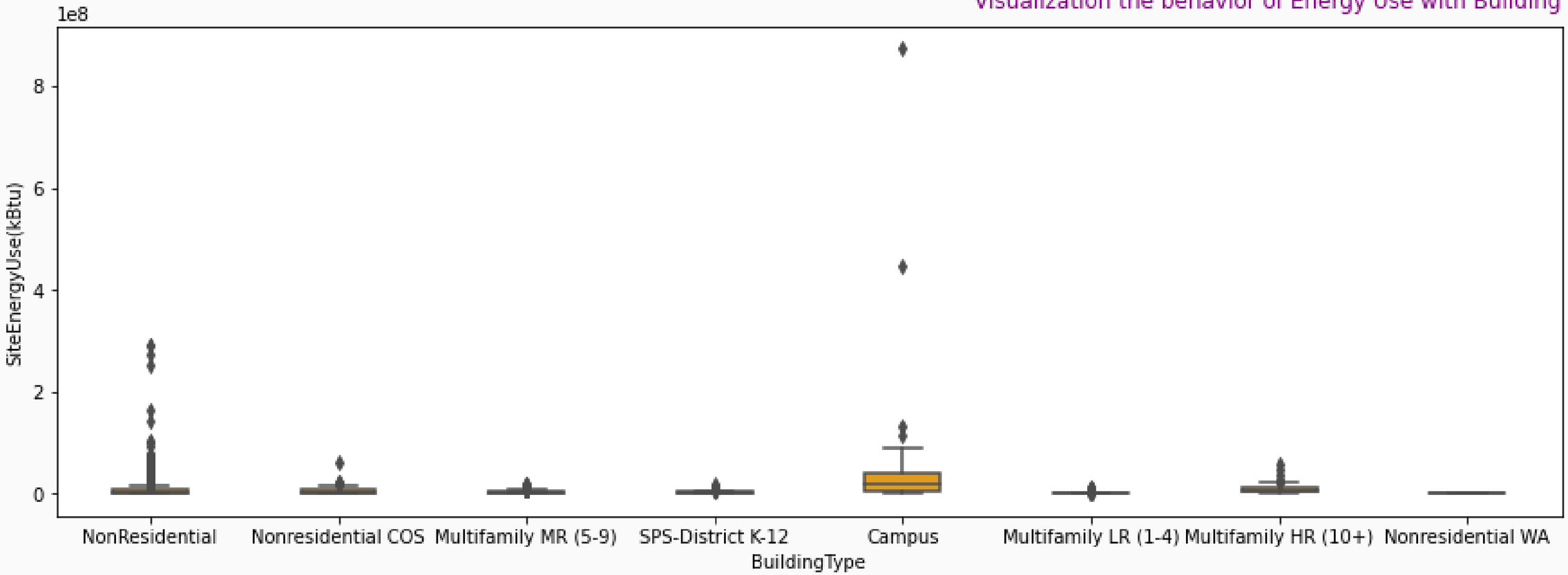


X

Data Exploration

Bivariate Analysis

Visualization the behavior of Energy Use with Building

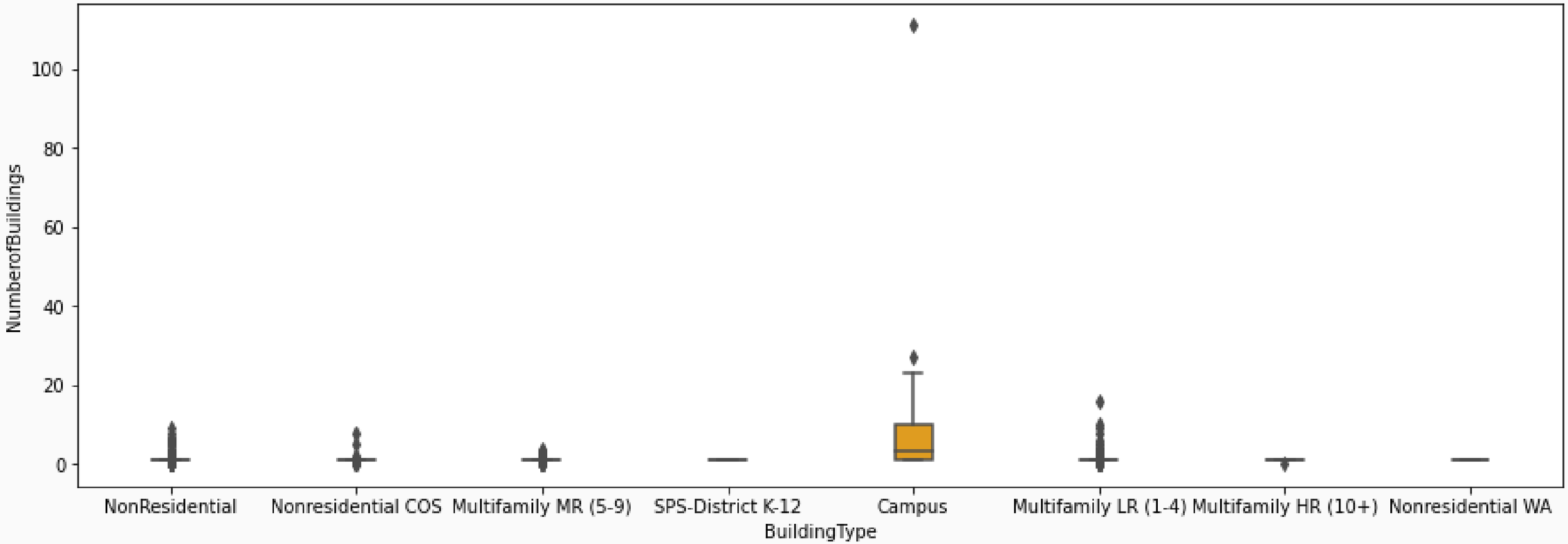


XI

Data Exploration

Bivariate Analysis

Visualization the behavior of Number of Buildings by Type

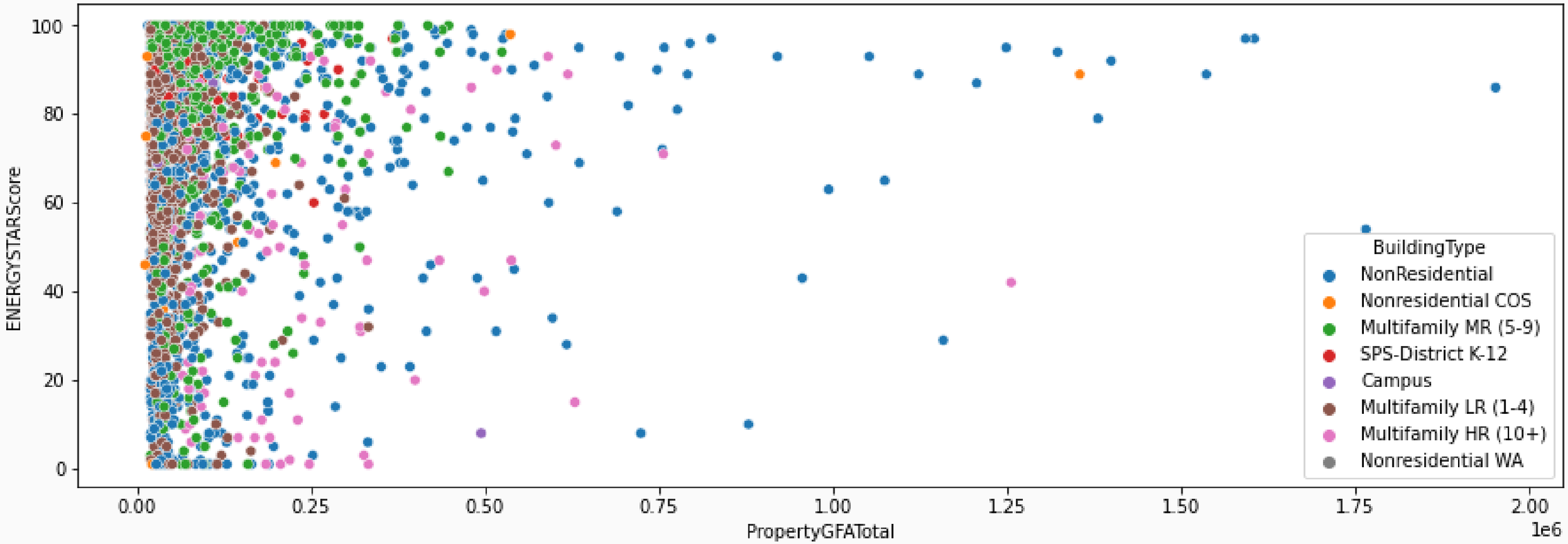


XII

Data Exploration

Bivariate Analysis

Visualization the behavior of PropertyGFATotal with ENERGYSTARScore by Building

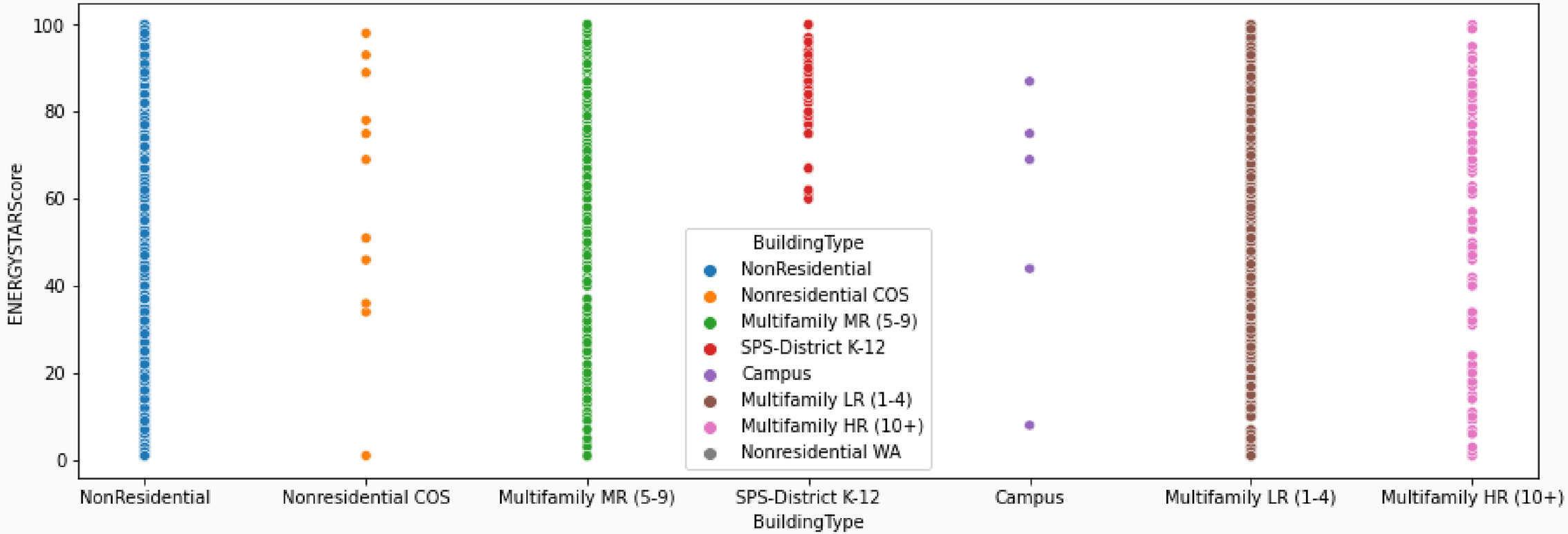


XIII

Data Exploration

Bivariate Analysis

Visualization the behavior of BuildingType with ENERGYSTARScore by Building

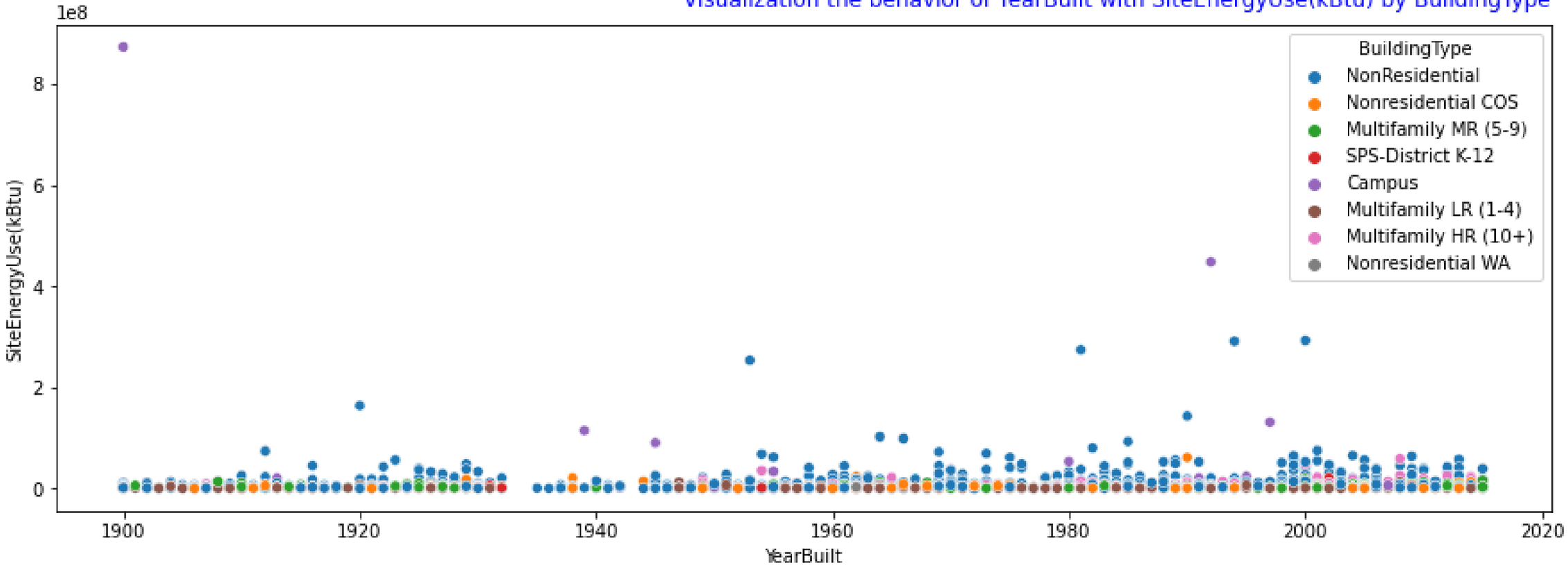


XIV

Data Exploration

Bivariate Analysis

Visualization the behavior of YearBuilt with SiteEnergyUse(kBtu) by BuildingType

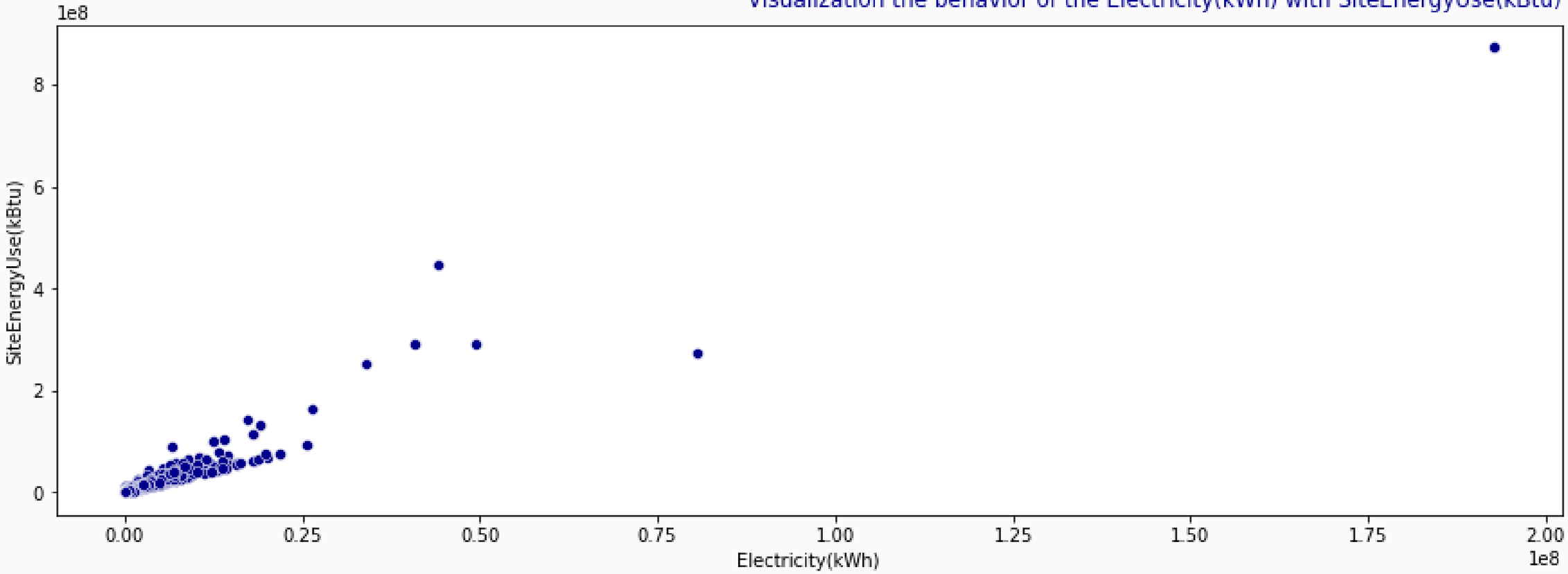


XV

Data Exploration

Bivariate Analysis

Visualization the behavior of the Electricity(kWh) with SiteEnergyUse(kBtu)

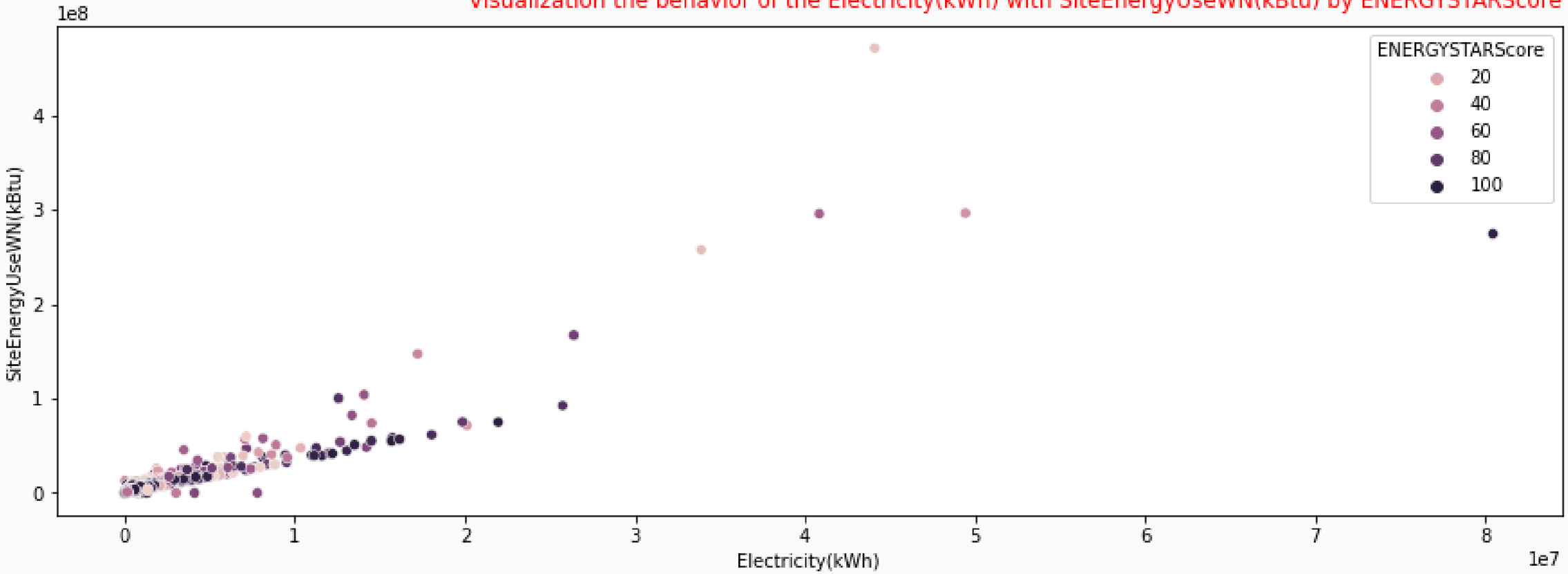


XVI

Data Exploration

Bivariate Analysis

Visualization the behavior of the Electricity(kWh) with SiteEnergyUseWN(kBtu) by ENERGYSTARScore

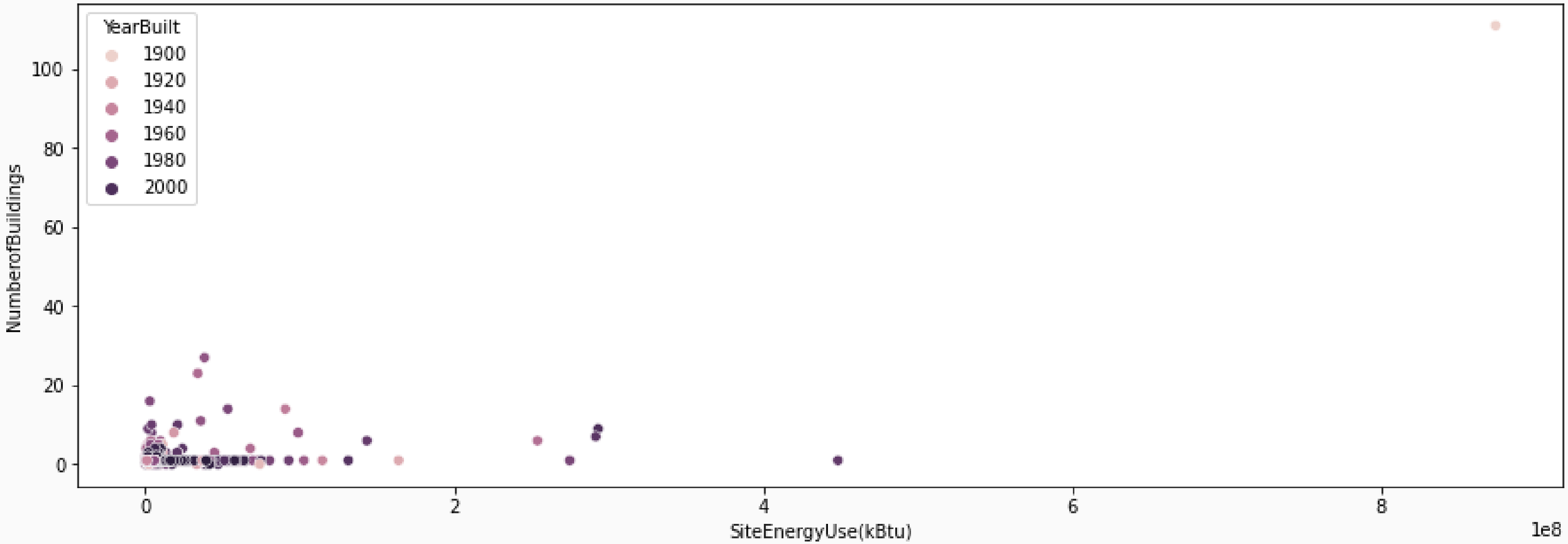


XVII

Data Exploration

Bivariate Analysis

Visualization the behavior of NumberofBuildings with SiteEnergyUse(kBtu) by YearBuilt

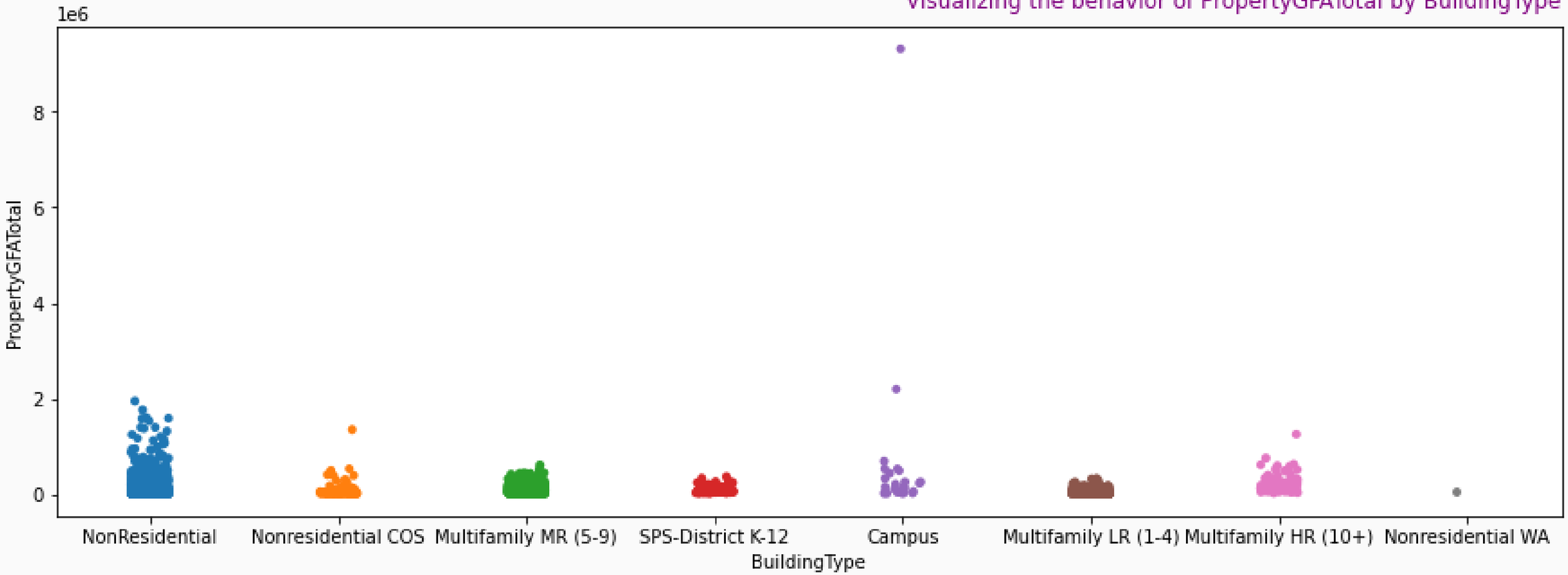


XVIII

Data Exploration

Bivariate Analysis

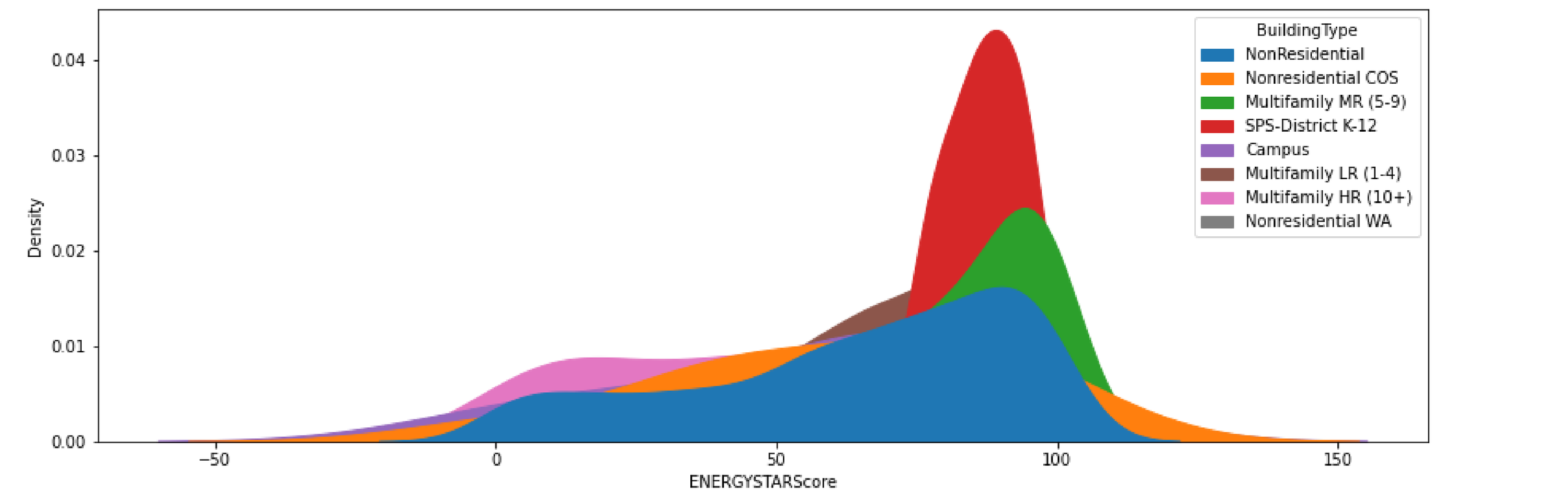
Visualizing the behavior of PropertyGFATotal by BuildingType



XIX

Data Exploration

Bivariate Analysis

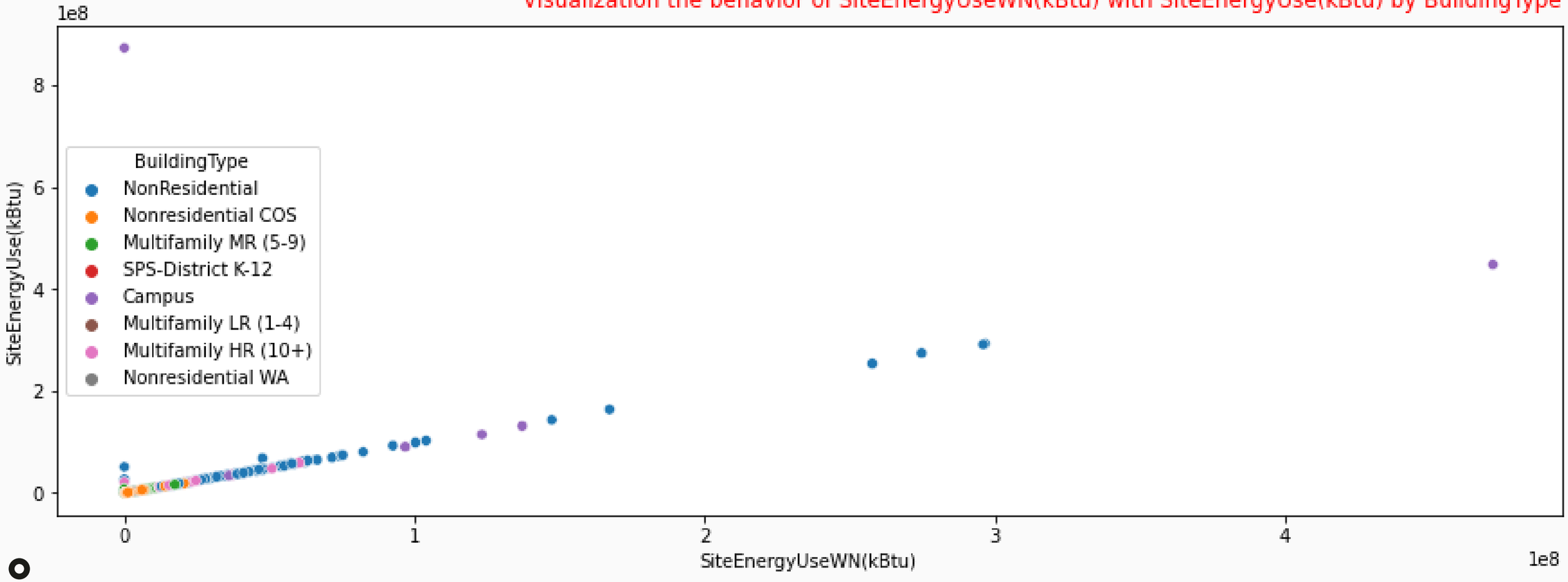


XX

Data Exploration

Bivariate Analysis

Visualization the behavior of SiteEnergyUseWN(kBtu) with SiteEnergyUse(kBtu) by BuildingType



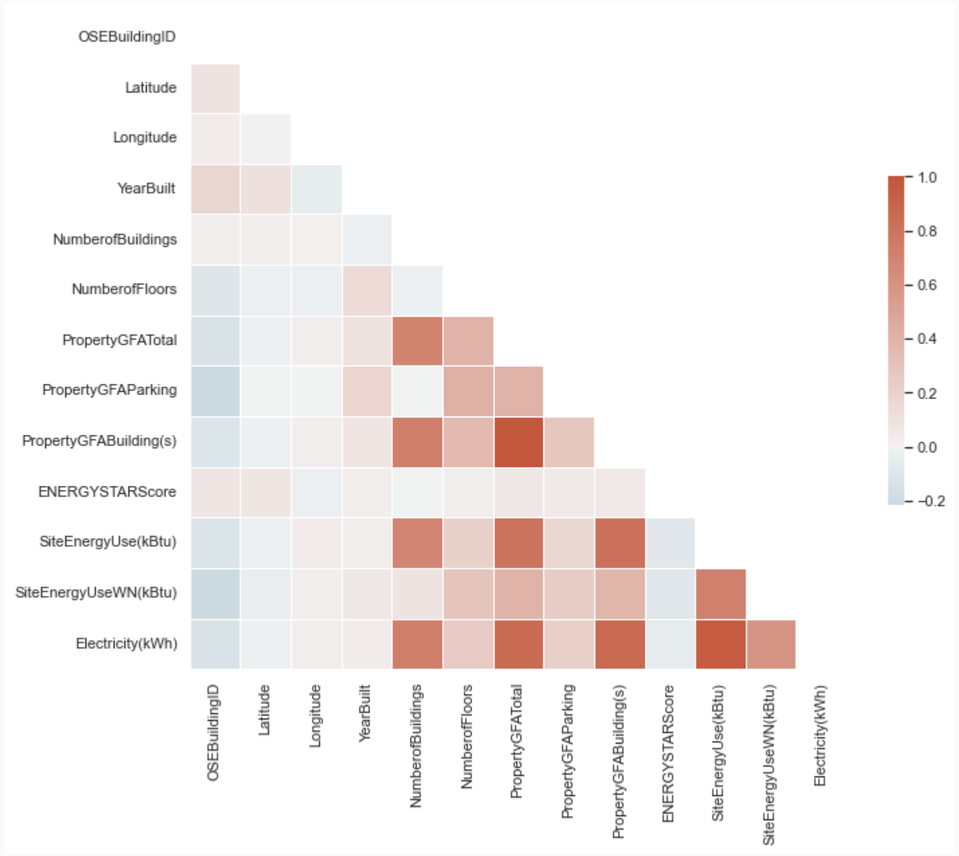
IV

Data PreProcessing

Data PreProcessing

Lors de cette étape, nous nettoyons notre jeu de données. Elle peut consister à gérer les valeurs nulles ou manquantes, gérer les outliers et/ou les doublons, faire du feature reduction, etc.

ACP & Correlation Matrix



XXIII

Data PreProcessing

Most Important Features

- Identifing the most important variables

```
# Removing useless features
newdata = dataset.drop(columns=['Electricity(kWh)', 'Longitude', 'Latitude', 'SiteEnergyUseWN(kBtu)'])

#Printing the shape
print(newdata.shape)
```

```
(3376, 13)
```

XXIV

Data PreProcessing

Missing and Null Values

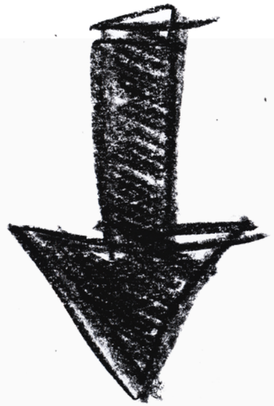
OSEBuildingID	0
BuildingType	0
PrimaryPropertyType	0
PropertyName	0
YearBuilt	0
NumberofBuildings	8
NumberofFloors	0
PropertyGFATotal	0
PropertyGFAParking	0
PropertyGFABuilding(s)	0
LargestPropertyUseType	20
ENERGYSTARScore	843
SiteEnergyUse(kBtu)	5

OSEBuildingID	0
BuildingType	0
PrimaryPropertyType	0
PropertyName	0
YearBuilt	0
NumberofBuildings	0
NumberofFloors	0
PropertyGFATotal	0
PropertyGFAParking	0
PropertyGFABuilding(s)	0
LargestPropertyUseType	0
ENERGYSTARScore	0
SiteEnergyUse(kBtu)	0

Data PreProcessing

- Outliers

```
(array([   2,   35,   49,  124,  167,  170,  231,  309,  340,  477,  558,
         559,  618, 1494, 1672, 1690, 2325, 3264, 3274],
       dtype='int64'),
```



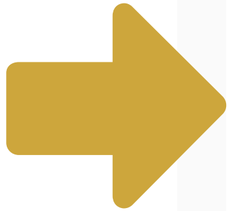
```
(3357, 13)
```

XXVI

Data PreProcessing

✓ LOG

```
count    3.376000e+03
mean     5.408019e+06
std      2.159304e+07
min      1.340900e+04
25%      9.386493e+05
50%      1.813404e+06
75%      4.218227e+06
max      8.739237e+08
Name: SiteEnergyUse(kBtu), dtype: float64
```



```
count    3376.000000
mean       14.582695
std         1.155866
min         9.503681
25%        13.752197
50%        14.410716
75%        15.254925
max        20.588504
Name: SiteEnergyUse(kBtu), dtype: float64
```

IV

Model Building

Model Building

Lors de cette étape, nous nettoyons notre jeu de données. Elle peut consister à gérer les valeurs nulles ou manquantes, gérer les outliers et/ou les doublons, faire du feature reduction, etc.

La première chose à faire est le Label Encoder pour gérer les valeurs catégorielles.

✓ Splitting (Train & Test)

```
(2685, 12)
(2685,)
(672, 12)
(672,)
```

XXIX

Model Building

Feature Scaling

	0	1	2	3	4	5	6	7	8	9	10	11
0	0.791171	-0.181140	-0.227791	1.302887	-1.265148	-0.129224	0.494204	0.310194	-0.361823	0.367131	-0.328804	1.254732
1	0.735443	0.661774	0.982308	0.530952	-1.140636	-0.129224	-0.744767	0.051780	-0.361823	0.097141	2.268370	0.526994
2	1.072907	3.190517	-1.092147	-0.347707	0.011106	-0.129224	-0.744767	-0.103486	-0.361823	-0.054184	-1.028043	0.569802
3	-1.281084	0.661774	-0.746405	0.454277	-2.074481	-0.129224	-0.125281	1.334079	-0.361823	1.402456	-0.029130	0.398570
4	-1.639188	3.190517	-1.092147	1.637565	0.664798	-0.129224	-0.538272	0.331910	-0.361823	0.391082	-1.028043	0.997884

XXX

Model Building

- Linear Regression
- XgBoost Regressor
- Random Forest Regressor
- Decision Tree Regression
- Lasso Regression

	Model	Score	MAE	RMSE	EX_TIME	CROSS
1	XgBoost Regressor	0.776	333.197	210902.684	0.777	76.00
2	Random Forest Regression	0.760	335.674	225893.854	0.946	74.80
0	Linear Regression	0.680	414.537	301666.575	0.006	66.32
4	Lasso Regression	0.680	414.636	301511.304	0.019	66.32
3	Decision Tree Regressor	0.491	465.664	479830.762	0.042	51.02

IV

Optimize Our Model

Grid Search

C’est une méthode d’optimisation (hyperparameter optimization) qui va nous permettre de tester une série de paramètres et de comparer les performances pour en déduire le meilleur paramétrage.
Il existe plusieurs manières de tester les paramètres d’un modèle et le Grid Search est une des méthodes les plus simples. Pour chaque paramètre, on détermine un ensemble de valeurs que l’on souhaite tester.

XXXIII

Grid Search

```python
# Importing GridSearch
from sklearn.model_selection import GridSearchCV

import warnings
warnings.filterwarnings('ignore')

#setting the hyperparameters
param_grid = [
    {'n_estimators': [30, 50, 80, 100], 'max_features': [2, 4, 6, 8, 10, 12]},
]

grid_search = GridSearchCV(model_rf, param_grid, cv=5, verbose=2,
                           scoring='neg_mean_squared_error',
                           return_train_score=True)

grid_search.fit(x_train, y_train)
```

```
Fitting 5 folds for each of 24 candidates, totalling 120 fits
[CV] END ...................max_features=2, n_estimators=30; total time=   0.0s
[CV] END ...................max_features=2, n_estimators=30; total time=   0.0s
[CV] END ...................max_features=2, n_estimators=30; total time=   0.0s
[CV] END ...................max_features=2, n_estimators=30; total time=   0.0s
[CV] END ...................max_features=2, n_estimators=30; total time=   0.0s
[CV] END ...................max_features=2, n_estimators=50; total time=   0.1s
[CV] END ...................max_features=2, n_estimators=50; total time=   0.1s
[CV] END ...................max_features=2, n_estimators=50; total time=   0.1s
[CV] END ...................max_features=2, n_estimators=50; total time=   0.1s
[CV] END ...................max_features=2, n_estimators=50; total time=   0.1s
[CV] END ...................max_features=2, n_estimators=80; total time=   0.2s
```

XXXIV

Grid Search

```
# displaying the best parameters
grid_search.best_params_
```

```
{'max_features': 6, 'n_estimators': 100}
```

```
# displaying the results
cvres = grid_search.cv_results_
for mean_score, params in zip(cvres["mean_test_score"], cvres["params"]):
    print(np.sqrt(-mean_score), params)
```

```
487.6469767228032 {'max_features': 2, 'n_estimators': 30}
483.8540301570484 {'max_features': 2, 'n_estimators': 50}
483.1065420618409 {'max_features': 2, 'n_estimators': 80}
479.912573735219 {'max_features': 2, 'n_estimators': 100}
488.08688558619684 {'max_features': 4, 'n_estimators': 30}
481.5209126692242 {'max_features': 4, 'n_estimators': 50}
477.61900623993677 {'max_features': 4, 'n_estimators': 80}
480.31594185416657 {'max_features': 4, 'n_estimators': 100}
484.38049070509504 {'max_features': 6, 'n_estimators': 30}
482.3028721979103 {'max_features': 6, 'n_estimators': 50}
480.04951718741876 {'max_features': 6, 'n_estimators': 80}
476.8866447005832 {'max_features': 6, 'n_estimators': 100}
485.42758226221486 {'max_features': 8, 'n_estimators': 30}
480.1019805511762 {'max_features': 8, 'n_estimators': 50}
```


thank
you