2020

OPENCLASSROOMS

Parcours Data-Scientist – Projet 4

CentraleSupélec

ANTICIPEZ LES BESOINS EN CONSOMMATION ÉLECTRIQUE DE BÂTIMENTS

SOMMAIRE

- Problématique
- Présentation des données (structure / contenu du dataset)
- Nettoyage des données & Feature engineering
- Analyse univariée
- Analyse multivariée
- Algorithmes d'estimation
- Conclusion et perspectives

PROBLEMATIQUE

Seattle 2050

- **Historique :** relevés de consommation par des agents en 2015 et 2016
- **Problème :** ces relevés sont coûteux à obtenir !
- **Objectif :** ville neutre en émissions de carbone en 2050
  => estimer les émissions de CO_2 et la consommation totale d'énergie de bâtiments à partir de ceux déjà réalisés.
- **Méthodologie :**
  - Réaliser une courte analyse exploratoire.
  - Tester différents modèles de prédiction afin de répondre au mieux à la problématique.

PRESENTATION DES DONNEES

- Structure du dataset, d'où proviennent les données ?
- Contenu du dataset, taux de remplissage

STRUCTURE DU DATASET

- **Jeu de données :** https://www.kaggle.com/city-of-seattle/sea-building-energy-benchmarking#2015-building-energy-benchmarking.csv
- **Format de fichier :** .json (description), .csv (datasets)
- **Taille :** 3MB
- **Nombre de colonnes :** 42 (2015), 46 (2016)
- **Nombre de lignes :** 3340 (2015), 3376 (2016) => volume relativement faible

CONTENU DU DATASET

- Majorité de colonnes remplies
- ENERGYSTAR Score manquant dans plus de 20% des cas
- 7 colonnes sous les 50% de remplissage

NETTOYAGE DES DONNEES & FEATURE ENGINEERING

NETTOYAGE DES DONNEES

- Processus de nettoyage de données
- 5 étapes principales

NETTOYAGE DES DONNEES

- Renommage de colonnes
- Fusion des datasets
- Supression de colonnes et lignes peu remplies
- Supression de colonnes non utiles à l'analyse
- Supression de lignes aberrantes

```
# Nous renommons les 'GHGEmissionsIntensity' dans df1 et df2 (même définition d'après json) :
df1.rename(columns={"GHGEmissionsIntensity(kgCO2e/ft2)": "GHGEmissionsIntensity",
                    "GHGEmissions(MetricTonsCO2e)": "TotalGHGEmissions"}, inplace=True)
```

NETTOYAGE DES DONNEES

Renommage de colonnes

Fusion des datasets

Supression de colonnes et lignes peu remplies

Supression de colonnes non utiles à l'analyse

Supression de lignes aberrantes

```
# On fusionne les deux dataframes :
df = pd.concat([df1, df2], axis = 0, sort = False)

# Puis on trie pour que chaque batîment montre les relevés des 2 années consécutives :
df = df.sort_values(["OSEBuildingID","DataYear"], ascending = [True, True])
df.head(10)
```

NETTOYAGE DES DONNEES

Renommage de colonnes

Fusion des datasets

Supression de colonnes et lignes peu remplies

Supression de colonnes non utiles à l'analyse

Supression de lignes aberrantes

```
# On affiche les colonnes faiblement remplies :
remplissage.loc[(remplissage['Remplissage(%)'] <= 50)]
```

	Remplissage(%)	Colonnes
42	49.672424	SecondLargestPropertyUseType
43	49.672424	SecondLargestPropertyUseTypeGFA
44	17.703990	ThirdLargestPropertyUseTypeGFA
45	17.703990	ThirdLargestPropertyUseType
46	3.588446	YearsENERGYSTARCertified
47	2.814175	Outlier
48	0.372245	Comment

```
# Ajout d'une colonne ratio de données manquantes
df['nan_ratio'] = df.isnull().sum(axis=1) / len(df.columns) * 100

# Conservation des lignes dont le ratio de données manquantes est inférieur à 40% :
df = df[df['nan_ratio'] < 40]
```

```
# On supprime donc les colonnes dont les valeurs manquantes sont > 50% :

df = df.drop(['SecondLargestPropertyUseType', 'SecondLargestPropertyUseTypeGFA',
```

NETTOYAGE DES DONNEES

- Renommage de colonnes
- Fusion des datasets
- Supression de colonnes et lignes peu remplies
- Supression de colonnes non utiles à l'analyse
- Supression de lignes aberrantes

```
# ZipCode remplace 'Location', 'Address', 'City', 'State', 'Latitude', et 'Longitude' :
df = df.drop(['Location', 'Address', 'City', 'State', 'Latitude', 'Longitude'], axis='columns')
```

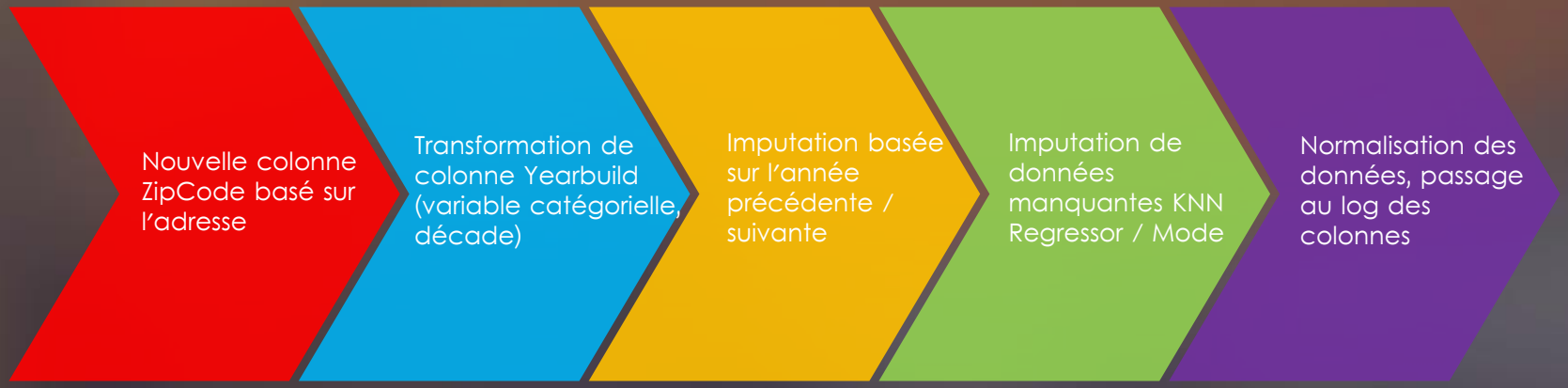
NETTOYAGE DES DONNEES

- Renommage de colonnes
- Fusion des datasets
- Supression de colonnes et lignes peu remplies
- Supression de colonnes non utiles à l'analyse
- Supression de lignes aberrantes

```
# Suppressions de toute valeur négative (sachant que les bâtiments ne produisent pas d'énergie) :

for i in nums:
    df[i].where((df[i]>0), np.nan, inplace=True)
```

FEATURE ENGINEERING



- Processus de Feature engineering
- 5 étapes principales

FEATURE ENGINEERING

- Nouvelle colonne ZipCode basé sur l'adresse
- Transformation de colonne Yearbuild (variable catégorielle, décade)
- Imputation basée sur l'année précédente / suivante
- Imputation de données manquantes KNN Regressor / Mode
- Normalisation des données, passage au log des colonnes

```
# On extrait le ZipCode en deux temps avec pandas.str :
df1['ZipCode'] = df1.Location.str[-9:]
df1['ZipCode'] = df1.ZipCode.str[:-4]
```

FEATURE ENGINEERING

- Nouvelle colonne ZipCode basé sur l'adresse
- Transformation de colonne Yearbuild (variable catégorielle, décade)
- Imputation basée sur l'année précédente / suivante
- Imputation de données manquantes KNN Regressor / Mode
- Normalisation des données, passage au log des colonnes

```
c = pd.cut(
    df1[['YearBuilt']].stack(),
    [1900, 1930, 1960, 1990, 2020],
    labels=['1900-1930', '1930-1960', '1960-1990', '1990-2020']
)
df1 = df1.join(c.unstack().add_suffix('_generation'))
```

```
c = pd.cut(
    df2[['YearBuilt']].stack(),
    [1900, 1930, 1960, 1990, 2020],
    labels=['1900-1930', '1930-1960', '1960-1990', '1990-2020']
)
df2 = df2.join(c.unstack().add_suffix('_generation'))
```

FEATURE ENGINEERING

- Nouvelle colonne ZipCode basé sur l'adresse
- Transformation de colonne Yearbuild (variable catégorielle, décade)
- Imputation basée sur l'année précédente / suivante
- Imputation de données manquantes KNN Regressor / Mode
- Normalisation des données, passage au log des colonnes

```
# Pour chaque Bâtiment, on rempli les valeurs nulles par l'année précédente/suivante :

for i in cols:
    df[i] = df.groupby(['OSEBuildingID'])[i].fillna(method='ffill')
    df[i] = df.groupby(['OSEBuildingID'])[i].fillna(method='bfill')
df.head(10)
```

FEATURE ENGINEERING

- Nouvelle colonne ZipCode basé sur l'adresse
- Transformation de colonne Yearbuild (variable catégorielle, décade)
- Imputation basée sur l'année précédente / suivante
- Imputation de données manquantes KNN Regressor / Mode
- Normalisation des données, passage au log des colonnes

```
# Imputer et réassigner les index/colonnes :
df[nums] = pd.DataFrame(imputer.fit_transform(df[nums]),columns = df[nums].columns)
```

```
# On impute par mode :
df[cat]=df[cat].fillna(df.mode().iloc[0])
```

FEATURE ENGINEERING

- Nouvelle colonne ZipCode basé sur l'adresse
- Transformation de colonne Yearbuild (variable catégorielle, décade)
- Imputation basée sur l'année précédente / suivante
- Imputation de données manquantes KNN Regressor / Mode
- Normalisation des données, passage au log des colonnes

```
# On initie, entraîne et fait la normalisation :
scaler=StandardScaler()

x = scaler.fit_transform(x)
```

```
# On affiche les données normalisées :
dfm = pd.DataFrame(data = x, columns = features)
```

```
# Passage en log des variables cibles :

import numpy as np

df['LOG_SiteEnergyUse(kBtu)'] = np.log10(df['SiteEnergyUse(kBtu)'])
df['LOG_TotalGHGEmissions'] = np.log10(df['TotalGHGEmissions'])
```

ANALYSE UNIVARIÉE

D’où proviennent les relevés ?

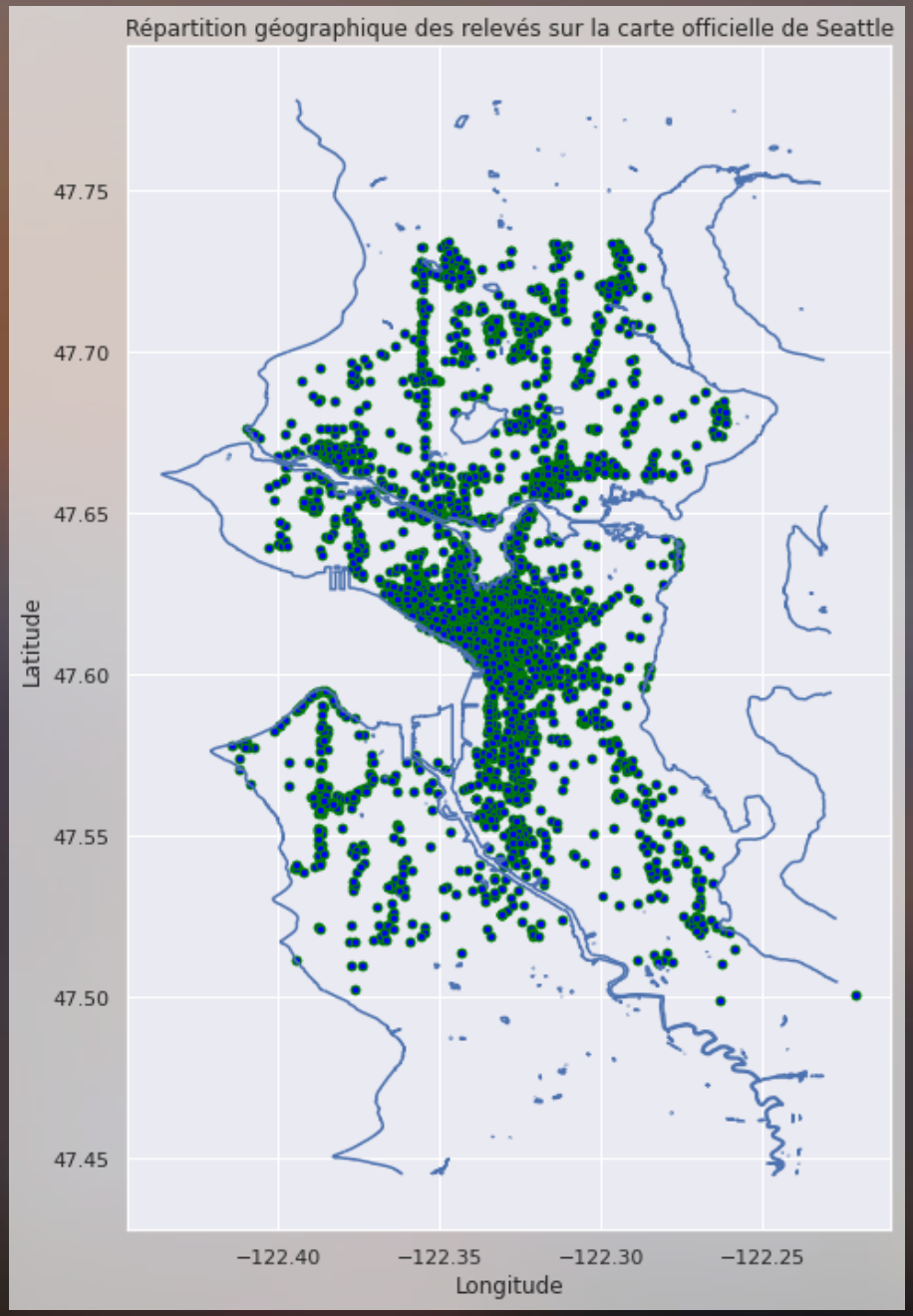
Quelques classements

Distribution des variables

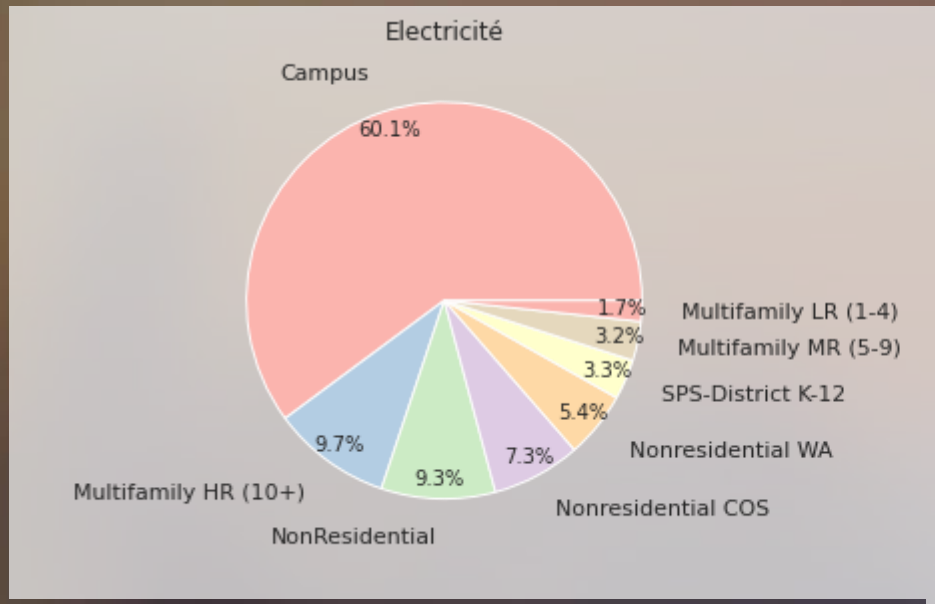
ANALYSE UNIVARIÉE

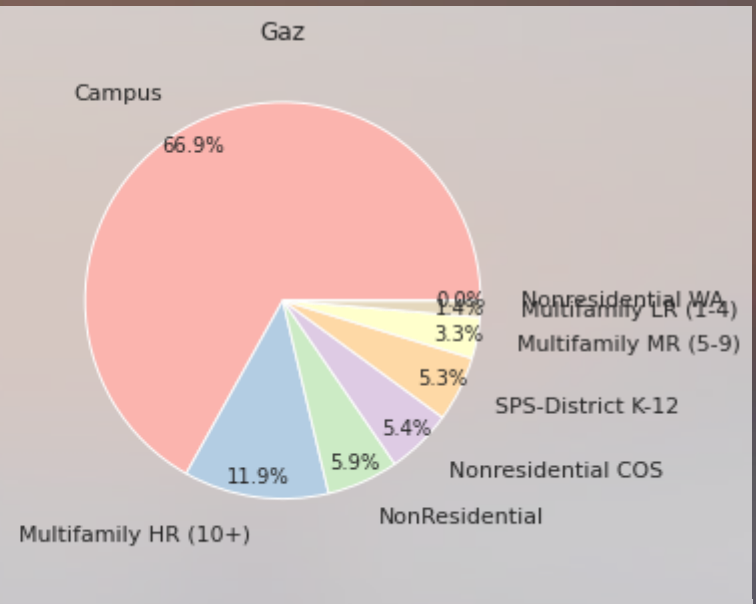
D'où proviennent les relevés ?

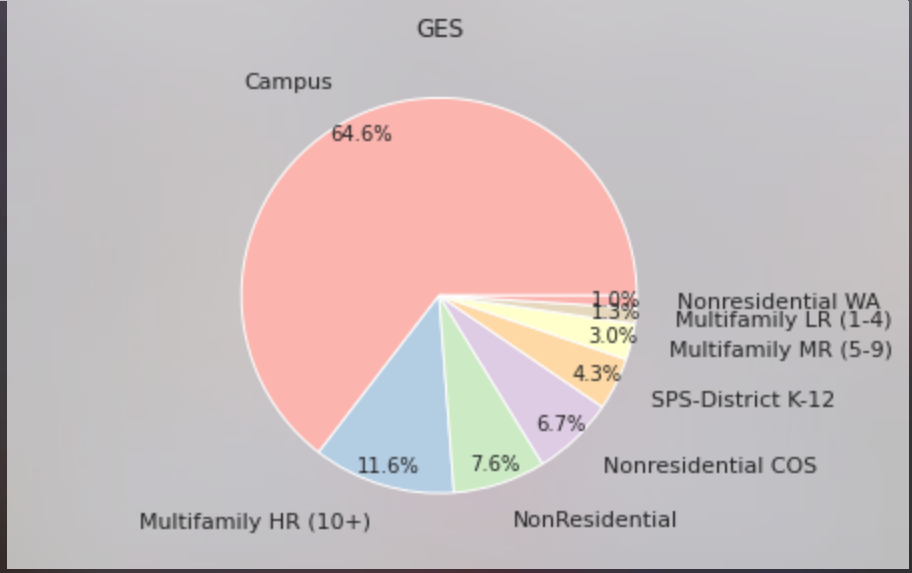
- Principalement centre-ville
- Beaucoup moins en périphérie
- Zones blanches dès qu'on s'éloigne des grands axes routiers



ANALYSE UNIVARIÉE







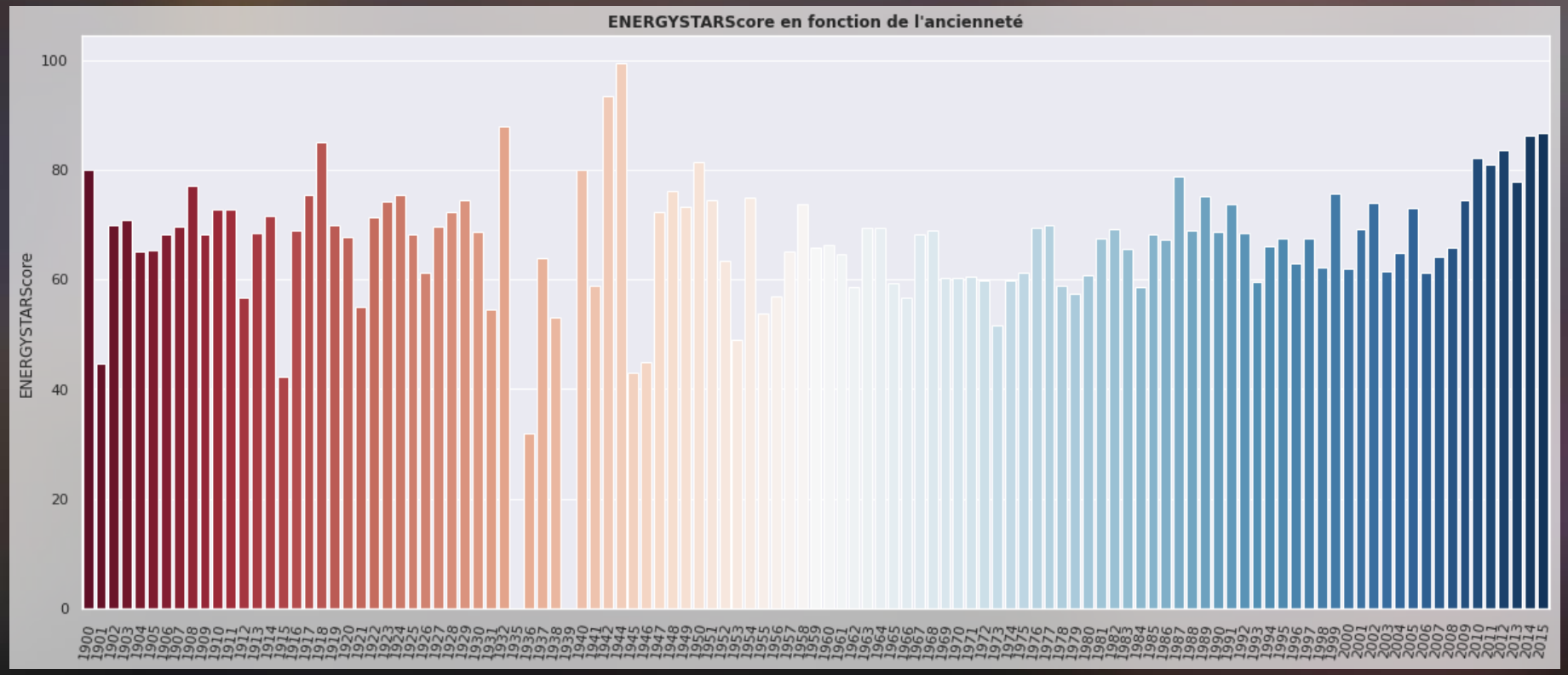
Répartition énergétique par type de bâtiment

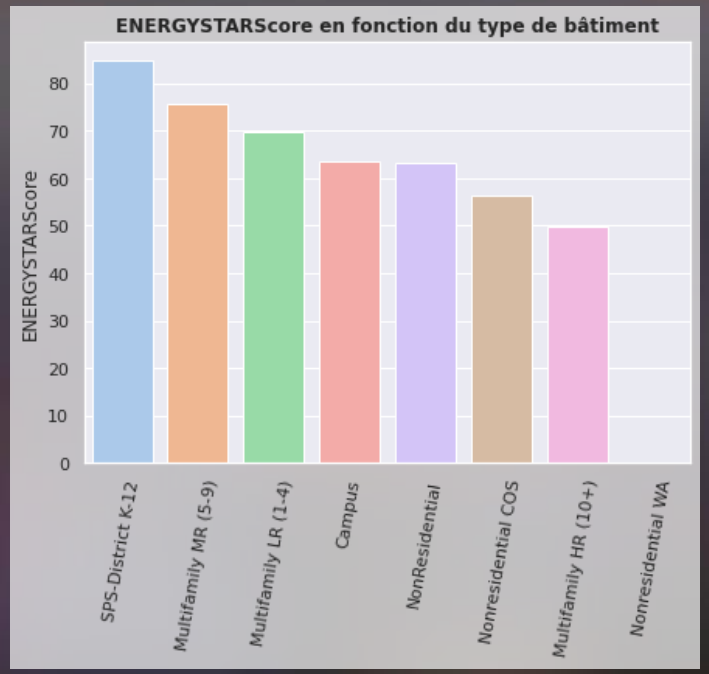
- Campus = gros consommateur
- Logements collectifs (Multifamily HR) loin derrière

ANALYSE UNIVARIÉE

- **Energy Star** = programme américain d'économie d'énergie (ici bâtiments)
- Score de 0 à 100, médiane à 50
- Score < 50 = moins performant que 50 % des bâtiments similaires dans le pays, score > 50 = plus performant que 50 % de ses pairs.
- **Score ≥ 75 = certification Energy Star**

- Les établissements scolaires pour enfants ont le meilleur score
- Pas d'Energy Star Score en 1939 (ou score à 0 ?)
- Scores moyens régulièrement compris entre 60 et 80
- Augmentation du score moyen entre 2009 et 2015 (bon)



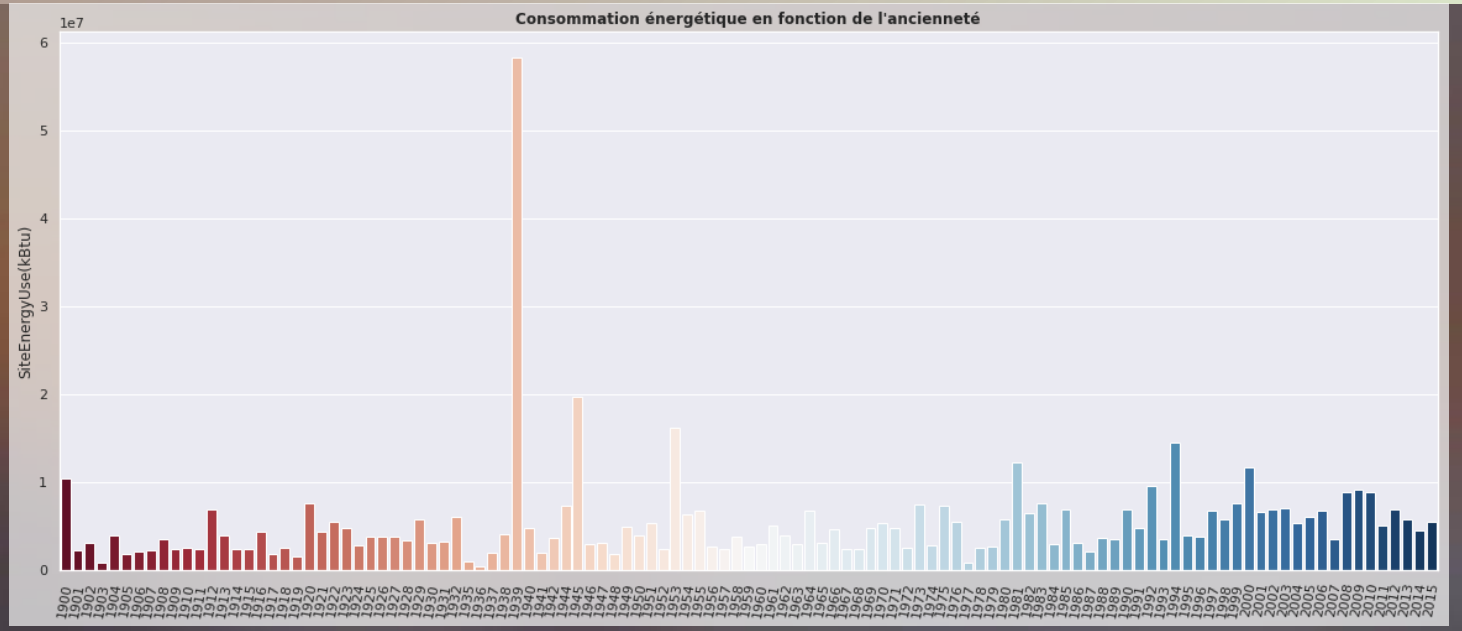


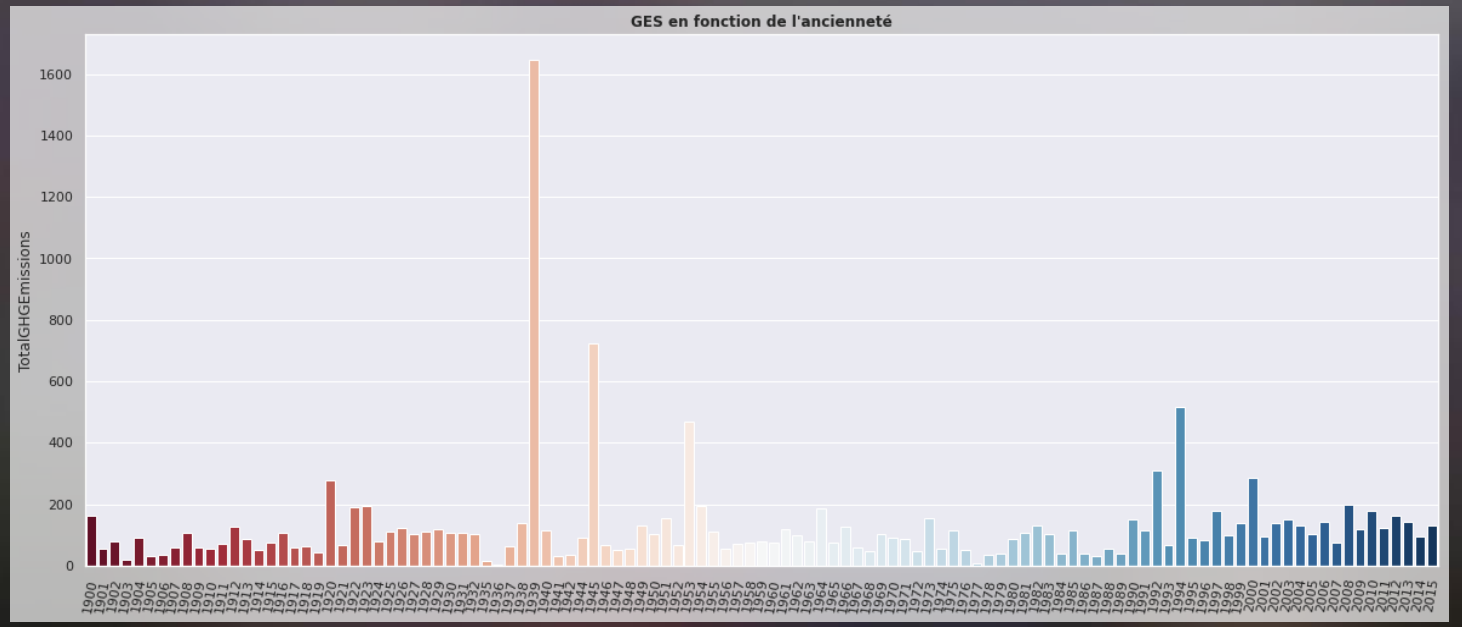
ANALYSE UNIVARIÉE

Consommation énergétique et Gaz à effet de serre par année de construction

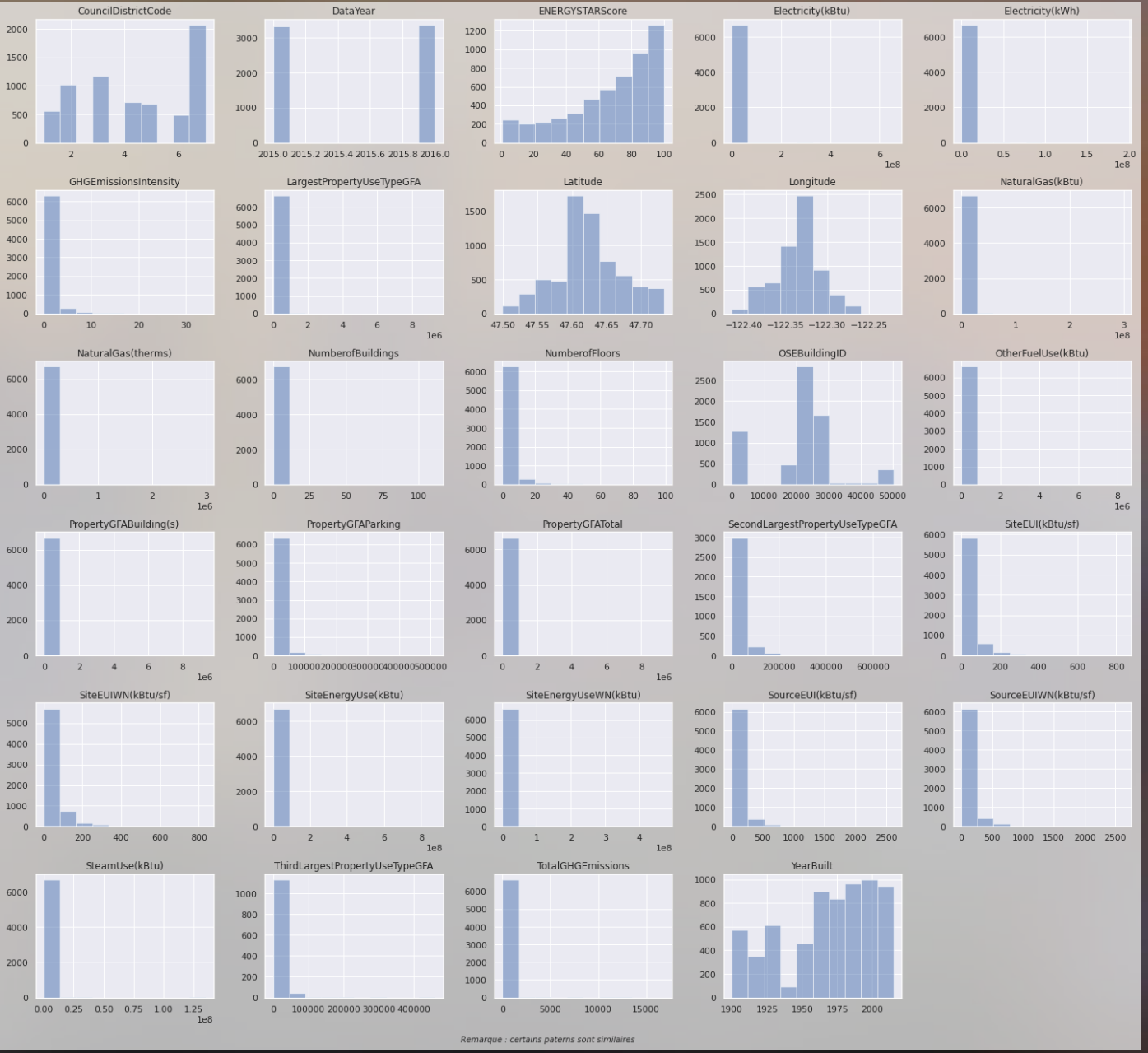
- Pic pour les bâtiments construits en 1939, corroboré par les 2 indicateurs (déplacement du complexe militaro-industriel de l'Atlantique à Seattle la même année ?) et 1945 = bâtiments de nature à consommer ?

- Autres pics visibles en 1953, et 1994

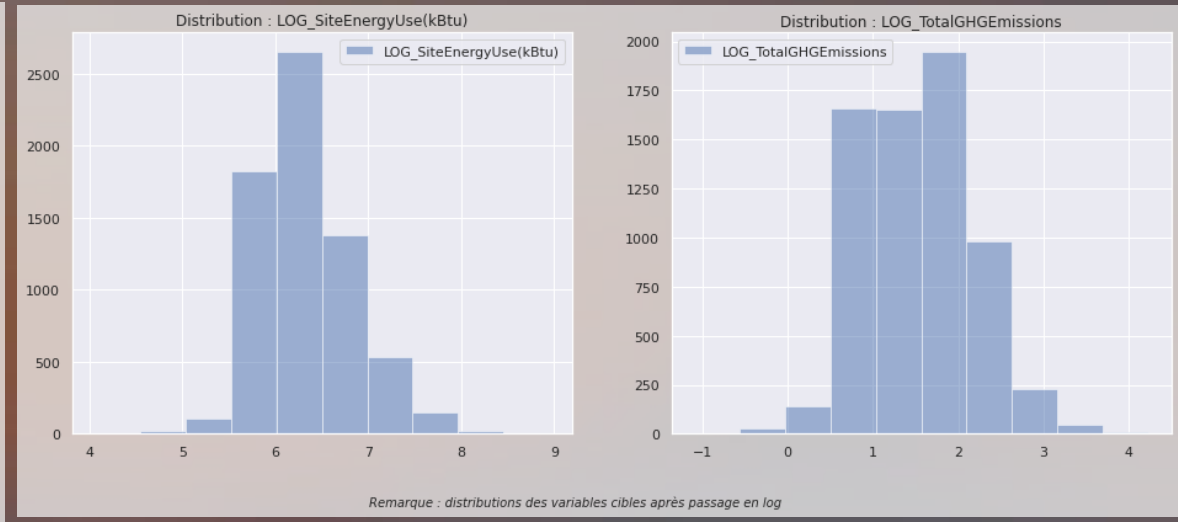




ANALYSE UNIVARIÉE



Remarque : certains paterns sont similaires



Remarque : distributions des variables cibles après passage en log

Distributions des variables

- Peu de formes gaussiennes
- Les variables cibles sont plus lisibles après passage à échelle logarithmique
- Leur distribution présentent une forme gaussienne

ANALYSE MULTIVARIÉE

ANALYSE MULTIVARIÉE

Analyse des Composantes Principales :

- 2 Dimensions (Composantes Principales) exprimant 62% de l'inertie totale du jeu de données
- Le Cercle des corrélations met en évidence quelques corrélations

PC1 : top 5 influence positive

	Dim 1
ENERGYSTARScore	0.292529
YearBuilt	-0.101972
NumberofBuildings	-0.178861
NumberofFloors	-0.362606
NaturalGas(kBtu)	-0.524545

PC1 : top 5 influence negative

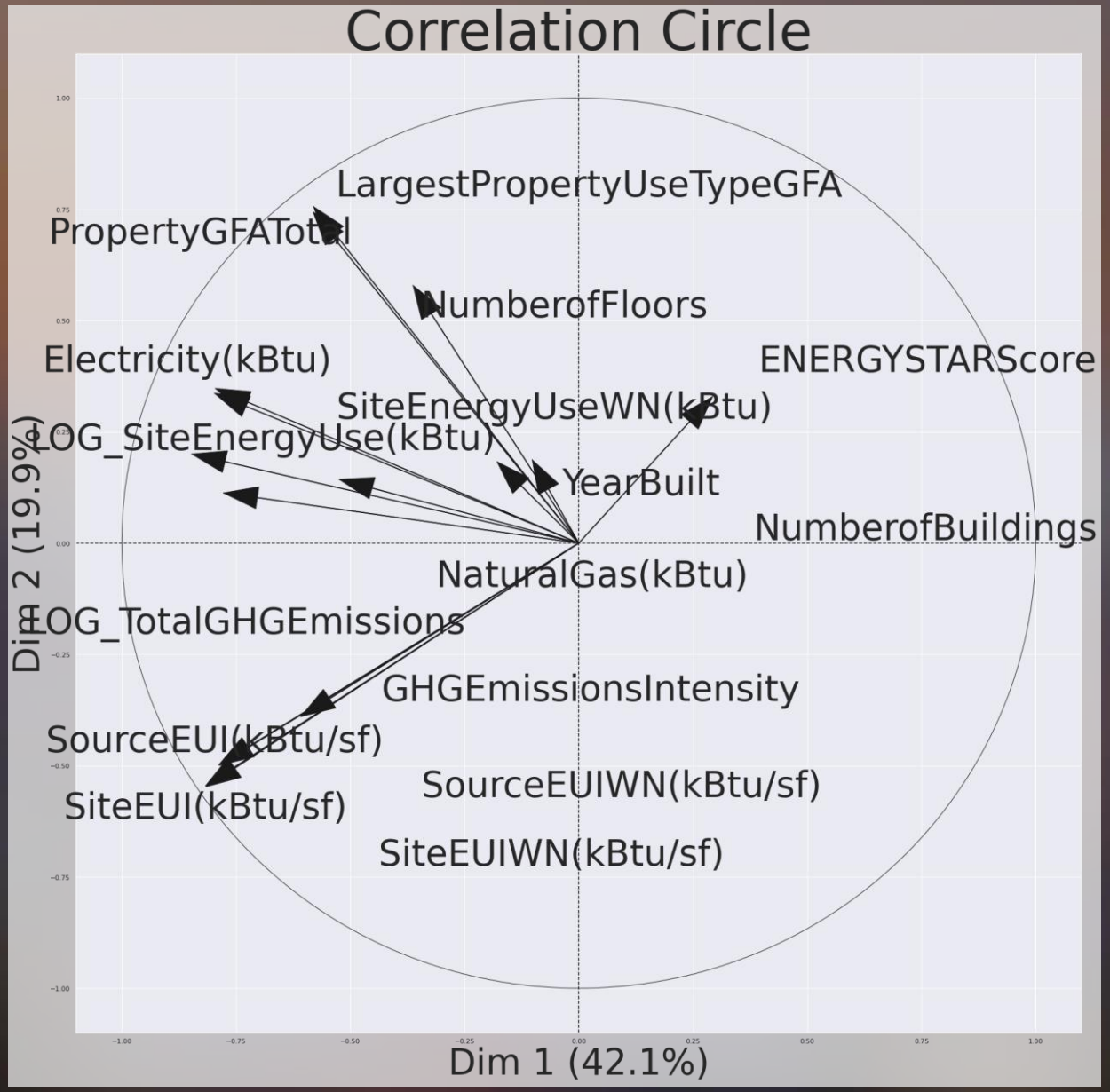
	Dim 1
LOG_SiteEnergyUse(kBtu)	-0.847105
SiteEUI(kBtu/sf)	-0.816696
SiteEUIWN(kBtu/sf)	-0.814872
SiteEnergyUseWN(kBtu)	-0.797046
Electricity(kBtu)	-0.795193

PC2 : top 5 influence positive

	Dim 2
LargestPropertyUseTypeGFA	0.755873
PropertyGFATotal	0.743808
NumberofFloors	0.578849
Electricity(kBtu)	0.34754
SiteEnergyUseWN(kBtu)	0.336393

PC2 : top 5 influence negative

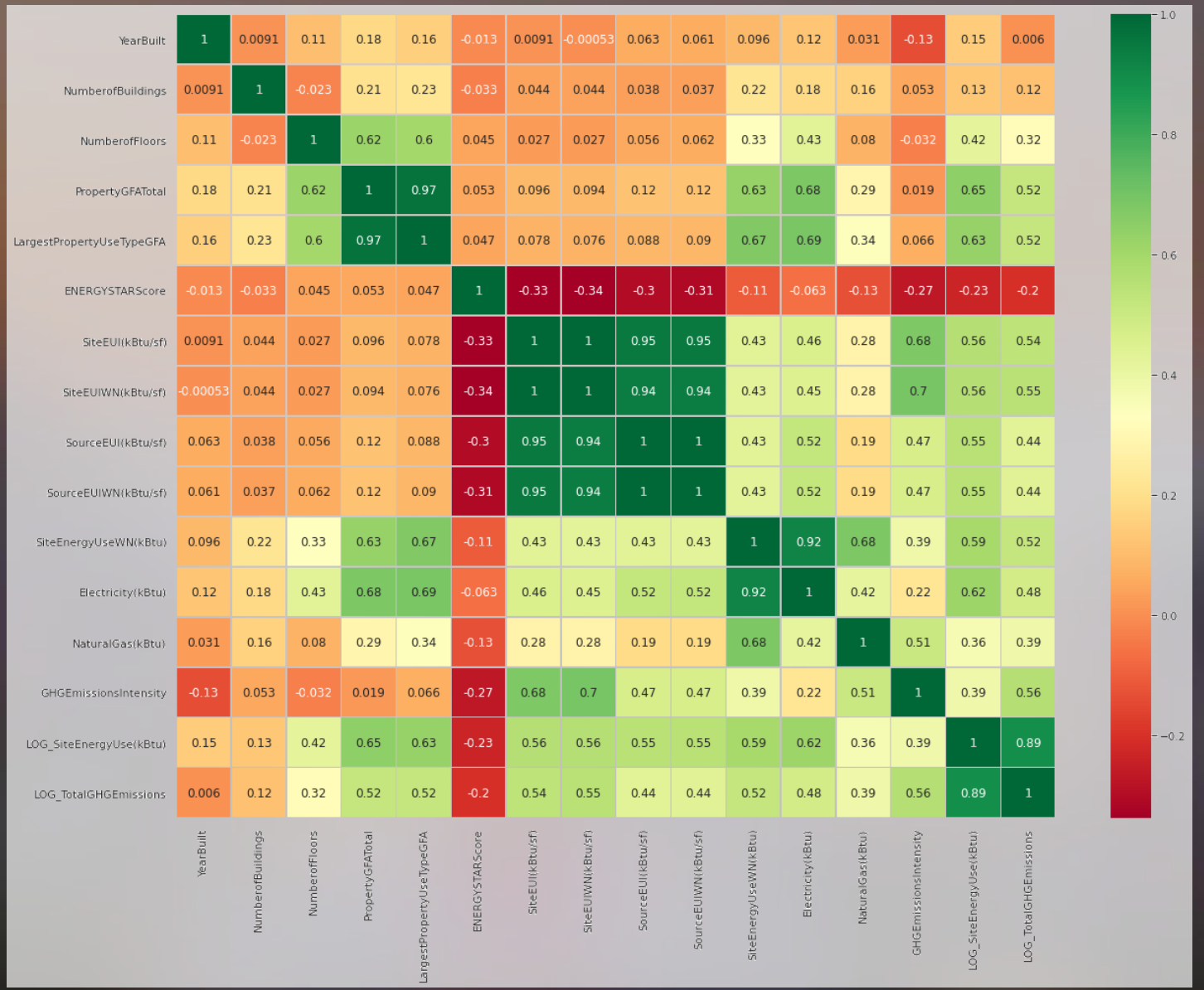
	Dim 2
SiteEUIWN(kBtu/sf)	-0.547233
SiteEUI(kBtu/sf)	-0.545312
SourceEUI(kBtu/sf)	-0.49907
SourceEUIWN(kBtu/sf)	-0.498155
GHGEmissionsIntensity	-0.389839



ANALYSE MULTIVARIÉE

Méthode de Pearson :

- La représentation graphique en heatmap semble confirmer les corrélations du cercle ACP.
- Mais aussi des colinéarités entre les variables
- Au vu de ces éléments **=> il est possible d'écrémer les variables utilisées pour la partie algorithmique**



ANALYSE MULTIVARIÉE

Choix des variables pour traitement algorithmique :

Variable cible Énergie :

- Suppression des variables **décorrélées** de la variable cible :
  - ~~NumberofBuildings,~~
  - ~~NumberofFloors,~~
  - ~~NaturalGas(kBtu),~~
  - ~~GHGEmissionsIntensity~~
- Suppression des variables **colinéaires** entre-elles (biais) :
  - ~~YearBuilt~~
  - ~~PrimaryPropertyType~~
  - ~~LargestPropertyUseTypeGFA~~
  - ~~SiteEUIWN(kBtu/sf)~~
  - ~~SourceEUI(kBtu/sf)~~
  - ~~SourceEUIWN(kBtu/sf)~~
- Suppression des variables **non nécessaires** à l'analyse (variante de la variable cible ou autre variable cible):
  - ~~ENERGYSTARScore~~
  - ~~SiteEnergyUseWN(kBtu)~~
  - ~~LOG_TotalGHGEmissions~~

Variable cible Gaz :

- Suppression des variables **décorrélées** de la variable cible :
  - ~~NumberofBuildings~~
  - ~~NumberofFloors~~
  - ~~SourceEUI(kBtu/sf)~~
  - ~~SourceEUIWN(kBtu/sf)~~
  - ~~NaturalGas(kBtu)~~
- Suppression des variables **colinéaires** entre-elles (biais) :
  - ~~YearBuilt~~
  - ~~PrimaryPropertyType~~
  - ~~LargestPropertyUseTypeGFA~~
  - ~~SiteEUIWN(kBtu/sf)~~
- Suppression des variables **non nécessaires** à l'analyse (variante de la variable cible ou autre variable cible):
  - ~~ENERGYSTARScore~~
  - ~~SiteEnergyUseWN(kBtu)~~
  - ~~LOG_SiteEnergyUse(kBtu)~~

Il reste donc :

	Numérique		Catégorielle	
Type de variable	**Énergie**	**Gaz**	**Énergie**	**Gaz**
Cible	LOG_SiteEnergyUse(kBtu)	LOG_TotalGHGEmissions		
Explicative	PropertyGFATotal SiteEUI(kBtu/sf) Electricity(kBtu)	PropertyGFATotal SiteEUI(kBtu/sf) Electricity(kBtu) GHGEmissionsIntensity	BuildingType ZipCode YearBuilt_generation	BuildingType ZipCode YearBuilt_generation

ALGORITHMES D'ESTIMATION

Quel meilleur algorithme ?

Quels sont les meilleurs hyperparamètres ?

ALGORITHMES D'ESTIMATION

Fonction automatique de préparation aux algorithmes:

On défini une fonction d'encodage automatique des données qui sépare le fichier csv en 2 tableaux :

```
8]:  # DEFINITION FONCTION ENCODAGE AUTOMATIQUE :

     def pipeline(FICHIER_VARIABLE):
         global dfe
         global dfg


         # LECTURE DU FICHIER CSV
         df = pd.read_csv(FICHIER_VARIABLE)
         list(df.columns)
```

On applique le pipeline à notre fichier csv

```
:  # Choix du fichier à encoder :
   pipeline('df.csv')

Processus d'encodage automatique du fichier réussi !
-------
  Le format de votre dataframe "dfe" est : 74 colonnes et 6712 lignes
-------
  Le format de votre dataframe "dfg" est : 75 colonnes et 6712 lignes
```

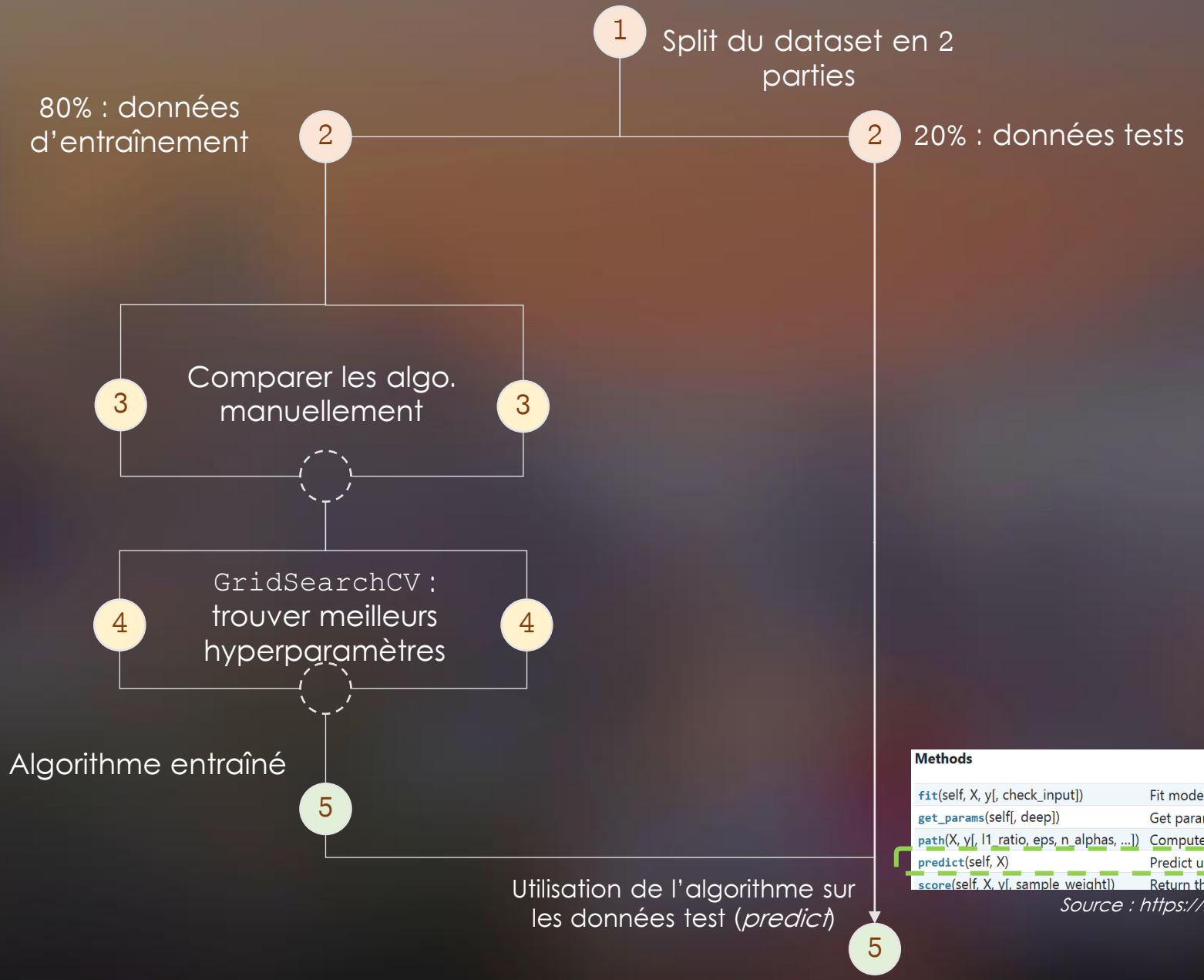
À partir d'un fichier csv, mise en forme automatique (encodage, nettoyage) de 2 tableaux :

- **dfe** (target = énergie)
- **dfg** (target = gaz à effet de serre)

Évaluation des performances de chaque algorithme

ALGORITHMES D'ESTIMATION

Méthodologie :



Methods

`fit`(self, X, y[, check_input])	Fit model with coordinate descent.
`get_params`(self[, deep])	Get parameters for this estimator.
`path`(X, y[, l1_ratio, eps, n_alphas, ...])	Compute elastic net path with coordinate descent.
`predict`(self, X)	Predict using the linear model.
`score`(self, X, y[, sample_weight])	Return the coefficient of determination R^2 of the prediction.

Source : https://scikit-learn.org

ALGORITHMES D'ESTIMATION

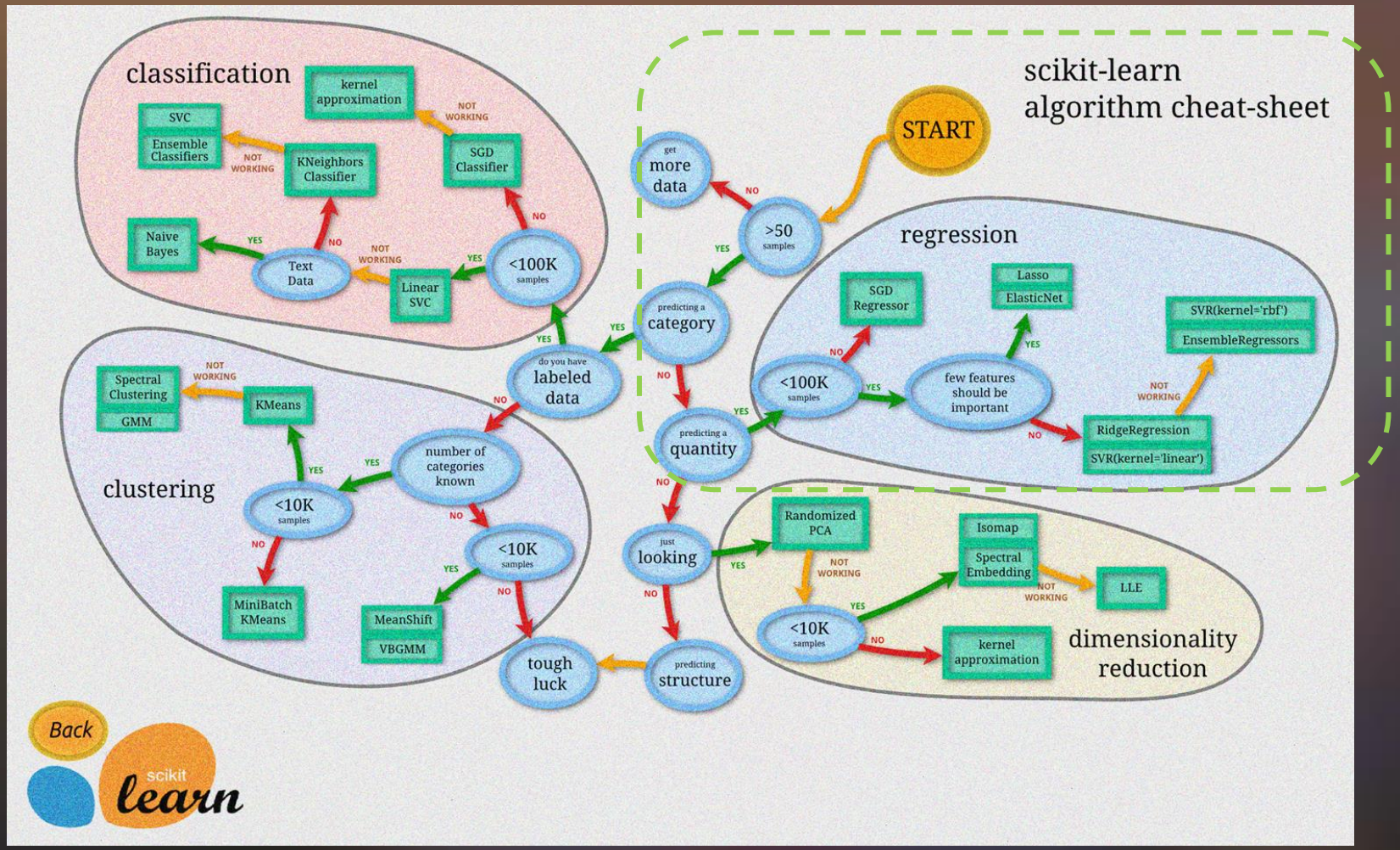
Choix des algorithmes :

3 Comparer les algo. manuellement 3

- Estimer le SiteEnergyUse(kBtu) ou le TotalGHGEmissions (quantités)
- Base de données de 6000 échantillons
- On ne connait pas exactement l'importance des features

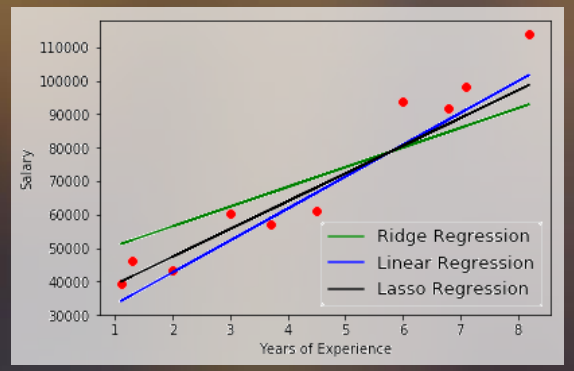
- => **Lasso** et **RidgeRegression**

- => Guide non exhaustif, nous y ajoutons **Linear regression** et **KNN Regression**



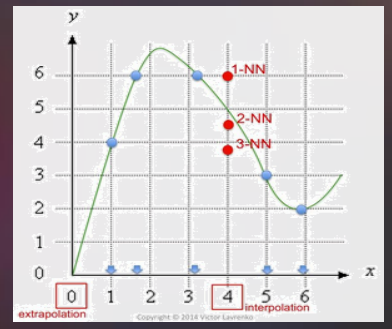
ALGORITHMES D'ESTIMATION

3 Comparer les algo. manuellement 3



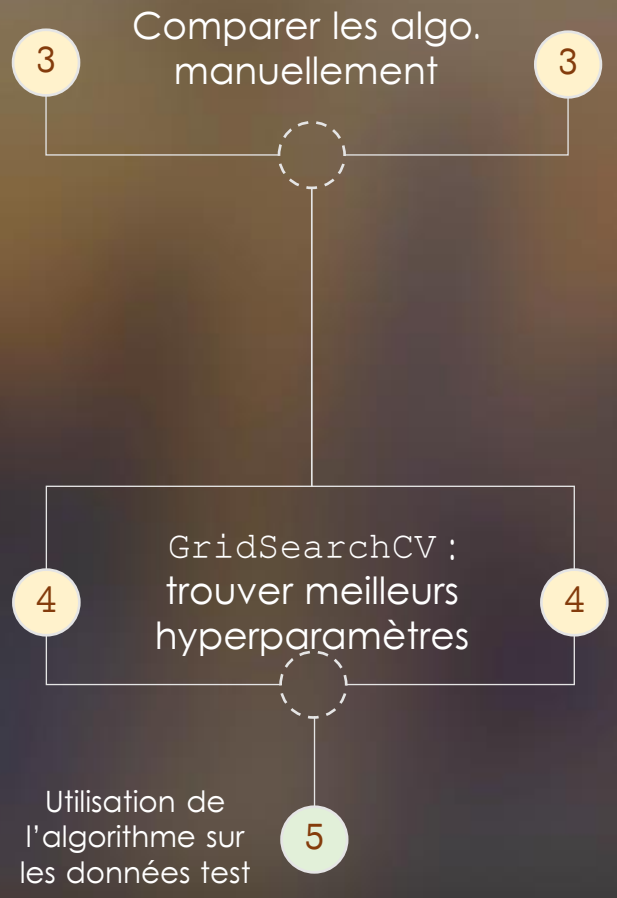
Source : Mubarak Bajwa – Ridge and Lasso a Simple Overview (medium.com)

- La **régression linéaire :**
  - Le modèle n'est pas pénalisé pour son choix de poids.
  - Pendant la phase d'entraînement, si le modèle estime qu'une variable est importante, il peut lui attribuer un poids important.
- Le **modèle Lasso :**
  - Modification de la régression linéaire, où le modèle est pénalisé pour la somme des valeurs absolues des poids.
  - Nouvel hyperparamètre, alpha = coefficient pour pénaliser les poids.
- Le **modèle Ridge :**
  - Va plus loin : il pénalise le modèle pour la somme des valeurs carrées des poids.
- **K plus proches voisins (KNN) :**
  - Stocke tous les cas disponibles et estime la cible numérique sur la base d'une mesure de similarité (par exemple, la distance).



Source : Victor Lavrenko – kNN.8 Nearest-neighbor regression example (youtube.com)

ALGORITHMES D'ESTIMATION



- Comparatif des algorithmes en mode « manuel » et via GridSearchCV
- GridSearchCV permet de trouver les hyperparamètres optimaux en subdivisant les données d'entraînement en autres données d'entraînement + validation
- Choix à faire entre la performance du score et le temps de calcul

		Hyperp. par défaut (aplha = 1, neighbors = 5)		**GridSearchCV**		
		Score	Temps (s)	Hyperp.	Score	Temps (s)
Variable cible Énergie	Régression Linéaire	0.691	0.021	–	–	–
	Régression de Lasso	0.670	0.012	alpha: 0.001	0.683	5.481
	Régression d'arête (Ridge)	0.692	0.019	alpha: 0.01	0.682	0.406
	Régression KNN	0.897	0.073	- metric: manhattan - n_neighbors: 21 - weights: distance	0.959	22.061
Variable cible GES	Régression Linéaire	0.608	0.063	–	–	–
	Régression de Lasso	0.539	0.02	alpha: 0.001	0.608	6.351
	Régression d'arête (Ridge)	0.609	0.017	alpha: 0.01	0.607	0.481
	Régression KNN	0.580	0.093	- metric: manhattan - n_neighbors : 45 - weights: distance	0.833	26.318

- **GridSearchCV** a globalement permis de trouver de meilleurs scores.
- **KNN Regressor** propose des scores très satisfaisants…
- … mais durée de calcul plus longue que ses concurrents (attention aux gros volumes de données !)
- Si l'on doit privilégier la durée de calcul à la précision, préférer **Ridge Regression**

CONCLUSION ET PERSPECTIVES

CONCLUSION ET PERSPECTIVES

Il a été effectué :

- Nettoyage d'un jeu de données et du Feature engineering
- Analyse univariée de ces données (origine des relevés, classements, distributions, etc.)
- Analyse multivariée de ces données (corrélations & colinéarité)
- Divers algorithmes de régression (KNN = meilleurs scores, Ridge = meilleurs temps) –> possibilité de faire des estimations sur la base de nos datasets.

Perspectives pour la ville de Seattle:

- Aller plus loin en testant d'autres algorithmes (ex: arbre de régression) et les comparer aux résultats
- Surveiller en temps réel toute anomalie de consommation / émission de GES par rapport aux données algorithmiques –> envoyer un agent vérifier l'anomalie si nécessaire
- Proposer un service / exiger l'installation d'énergies renouvelables en complément de l'électricité/gaz pour les bâtiments les plus consommateurs, etc.