Machine learning appliqué au domaine de l'énergie

Clustering de la consommation annuelle

d'électricité des bâtiments et

Clustering de courbes de charge journalières

selon la consommation journalière

d'électricité d'un bâtiment

Cristina Ghinda

3WAcademy

2023

Table des matières

Description du document

Ce document est un document explicatif du projet de clustering réalisé pour la validation de l'option « Construire et développer des modèles de big data », dans le cadre de la formation « Expert en informatique et systèmes d'information » proposée par la 3WAcademy.
Le code fait pour ce projet peut être trouvé à l'adresse :https://github.com/Cristina-MariaG/Big_data_clustering_electricity_consumption.git

Le code est exécutable dans un environnement virtuel python, mais le fait que la data n'est pas en libre accès fait qu'elles ne puissent pas être partagé, ce qui fait que l'exécution du code en dehors de la présentation orale du projet ne soient pas possibles.

Ce document est structuré conformément à la séquence suivante : une introduction, huit chapitres, et une conclusion.

L'introduction vise à présenter le champ d'étude choisi, centré sur l'électricité. Nous explorerons le lien entre l'entreprise Efficacity et la réduction de la consommation d'énergie, mettant en avant l'importance cruciale de l'intelligence artificielle dans le domaine de l'électricité.

Le premier chapitre, intitulé « Cadrage du problème et vue d'ensemble » fait une analyse de la problématique que nous étudierons.

Le deuxième chapitre, « Construction et récupération de la donnée » montrera les étapes dans la création de la data et comment les données utilisées dans cette étude ont été récupérées.

Le troisième chapitre, nommé « Récupération et exploration de la data » montrent les procédures que nous utiliserons pour importer la data ainsi que son exploration.

Le quatrième chapitre représente les résultats d'une étude sur le clustering de la consommation totale d'électricité par jour de la semaine, sur une durée d'une année pour tous les bâtiments du secteur résidentiel. Nous allons parcourir les étapes les plus importantes qui ont dû être suivi dans la réalisation de cette étude, ainsi qu'une interprétation des résultats obtenus.

Le cinquième chapitre « Création des profils journalières » décris en détail tout le processus et les problèmes rencontrés ainsi que leur résolution dans le cadre de la construction des courbes de consommation d'électricité journalière.

Dans **le sixième chapitre,** une analyse approfondie de la consommation d'électricité d'un bâtiment est présentée. Cette section met en lumière la création détaillée des profils journaliers, englobant les étapes essentielles du processus ainsi que les algorithmes employés pour le clustering. Elle explore également les méthodes de visualisation des résultats, offrant une interprétation éclairée des conclusions tirées de ces analyses.

Le septième chapitre liste les démarches effectues dans le cadre des deux études présentées pour optimiser l'algorithme de machine learning.

Le huitième chapitre met en lumière les pistes d'amélioration envisageables pour le projet actuel. Nous suggérons l'intégration d'une base de données, le développement d'un backend utilisant une API, et la création d'un frontend qui consommera cette API. Nous détaillons dans ce chapitre les technologies recommandées ainsi que les étapes concrètes pour atteindre ces objectifs.

Introduction dans le projet

Le changement climatique est une réalité incontestable et il est plus urgent que jamais de réduire les émissions de gaz à effet de serre. Pour atteindre cet objectif, il est essentiel de maîtriser, optimiser et réduire notre consommation d'énergie.

C'est dans cette optique que l'entreprise Efficacity a été créée. Elle propose des solutions innovantes pour construire la ville de demain : une ville énergétiquement efficace et massivement décarbonée.

Efficacity

Efficacity est un institut de Recherche & Développement pour la transition énergétique de la ville et des milieux urbains, créés en 2014 par un consortium de 30 grands acteurs de l'industrie, de l'ingénierie et de la recherche.
Soutenu par l'État français à travers le programme d'investissement d'avenir, il a permis le rassemblement de plus de 100 chercheurs et experts de tous les horizons travaillant ensemble. Le but étant de développer et mettre en œuvre des solutions innovantes pour construire la ville de demain : une ville efficiente énergétiquement et massivement décarbonée visant à accélérer la transition énergétique des villes.

L'objectif d'Efficacity est de faire évoluer les pratiques et les compétences des acteurs à la fois publics et privés sur les villes durables et ainsi répondre à une politique publique prioritaire de l'État.
L'activité principale est alors de développer des outils et d'accompagner les acteurs de la ville pour permettre

- De concevoir et piloter des quartiers à très haute performance énergétique et environnementale.
- D'innover et expérimenter de façon beaucoup plus efficace.
- Sécuriser et accélérer les investissements au travers de contrats de performance énergétique à l'échelle urbaine.

L'IA dans le domaine de l'électricité

En examinant les données de consommation d'électricité pour des périodes similaires dans le passé, il est possible d'identifier des tendances saisonnières ou des événements spécifiques qui peuvent influencer la consommation d'électricité.

Il est possible de faire une analyse de données traditionnelle, mais l'utilisation de l'intelligence artificielle (IA) pour faire des prédictions a plusieurs avantages par rapport à l'analyse de données traditionnelle :

1. Capacité à identifier des modèles complexes : Les modèles d'IA peuvent identifier des relations et des modèles dans les données qui seraient difficiles, voire impossibles, à trouver avec des méthodes traditionnelles. L'IA peut ainsi améliorer considérablement la précision des prévisions.

2. Capacité d'adaptation : Les modèles d'IA peuvent s'adapter en temps réel à mesure que de nouvelles données sont collectées, ce qui signifie que les prévisions peuvent être mises à jour en permanence pour tenir compte des changements dans l'environnement.
3. Rapidité et efficacité : Les modèles d'IA peuvent analyser de grandes quantités de données en un temps record, ce qui signifie que les prévisions peuvent être générées rapidement et avec une grande efficacité.
4. Réduction de l'erreur humaine : L'IA peut réduire l'erreur humaine en automatisant le processus de prédiction, ce qui signifie qu'il y a moins de risques d'erreurs dues à des biais ou des erreurs humaines.

L’exploitation de la puissance de l'IA peut donc jouer un rôle important grâce à sa capacité à prédire et anticiper les événements. En utilisant l’IA, les compagnies intéressées par la consommation d’électricité peuvent obtenir de meilleures prévisions en matière de production et de consommation d'énergie, ce qui leur permet de gérer leurs réseaux plus efficacement et de programmer leur maintenance de manière plus efficiente. L'IA peut donc aider à réduire les pertes d'énergie et à optimiser la production, ce qui permet de diminuer les émissions de gaz à effet de serre et ainsi peut jouer un rôle crucial dans la lutte contre le changement climatique en aidant à optimiser l'utilisation de l'énergie et à réduire les émissions de gaz à effet de serre.

1. Cadrage du problème et vue d’ensemble

L'analyse de la consommation d'électricité d'un bâtiment peut être un élément clé pour comprendre et optimiser l'efficacité énergétique. La création de clusters, également connue sous le nom de regroupement, peut apporter plusieurs avantages :

1. Identification des profils de consommation : En regroupant les données de consommation électrique, on peut découvrir des schémas de consommation similaires ou des profils de charge communs. Cela permet de distinguer différents modes de fonctionnement du bâtiment, comme les heures de pointe, les périodes de faible consommation ou les moments de consommation anormalement élevée. Cette information est précieuse pour prendre des décisions éclairées en matière de gestion de l'énergie.
2. Détection d'anomalies : Les clusters peuvent aider à détecter les écarts et les comportements inhabituels dans la consommation d'électricité. En comparant les clusters ou les profils de consommation, il devient plus facile de repérer des variations ou des déviations anormales. Cela peut indiquer des problèmes techniques, des équipements défectueux ou des gaspillages d'énergie, permettant ainsi une intervention rapide pour résoudre les problèmes et améliorer l'efficacité.
3. Planification et optimisation énergétique : Les clusters peuvent servir de base pour développer des stratégies d'optimisation énergétique spécifiques. Par exemple, en identifiant les périodes de consommation élevée, on peut mettre en place des mesures pour réduire la demande pendant ces moments critiques, comme l'utilisation de systèmes de gestion de l'énergie, de contrôles automatiques ou de tarifications différenciées. Cela permet de réduire les coûts énergétiques, d'optimiser l'utilisation des ressources et de minimiser l'empreinte environnementale.
4. Comparaison et benchmarking : En regroupant les données de consommation d'électricité de différents bâtiments ou sites, il devient possible de comparer et de réaliser un benchmarking de leur performance énergétique. Cela peut aider à identifier les bâtiments les plus efficaces et à repérer ceux qui ont des opportunités d'amélioration. Ces informations peuvent être utilisées pour définir des objectifs de réduction de consommation, établir des normes de référence et partager les meilleures pratiques entre différents sites.

La création de clusters sur la consommation d'électricité d'un bâtiment permet de mieux comprendre les schémas de consommation, de détecter les anomalies, d'optimiser l'efficacité énergétique et de comparer les performances. Cela contribue à une gestion plus intelligente de l'énergie, à des économies de coûts et à une réduction de l'impact environnemental.

L’institut Efficacity est structuré en 3 pôles couvrant 3 domaines complémentaires, qui sont les suivants :

- Quartier bas carbone (QBC1)
- Stratégies urbaines durables
- Innovations urbaines

L’objectif de l’axe thématique « Quartiers Bas Carbone » (QBC) est de produire des méthodes et outils logiciel permettant d’accompagner les différents acteurs dans le volet énergie de leurs projets d’aménagement urbain (neuf ou rénovation). Les principaux enjeux

énergétiques d'un projet d'aménagement urbain à l'échelle d'un quartier peuvent être formulés de la façon suivante :

- Participer à la décarbonation des activités humaines avec pour objectif global la neutralité carbone ;
- Améliorer l'efficacité énergétique (consommations des bâtiments et performances des réseaux énergétiques en particulier) ;
- Contribuer au confort de vie (été comme hiver) dans les bâtiments et espaces extérieurs.

Pour répondre à ces enjeux, une chaîne d'outils est développée par Efficacity en collaboration avec ses membres. La figure 2 présente cette chaîne et les fonctionnalités des logiciels.

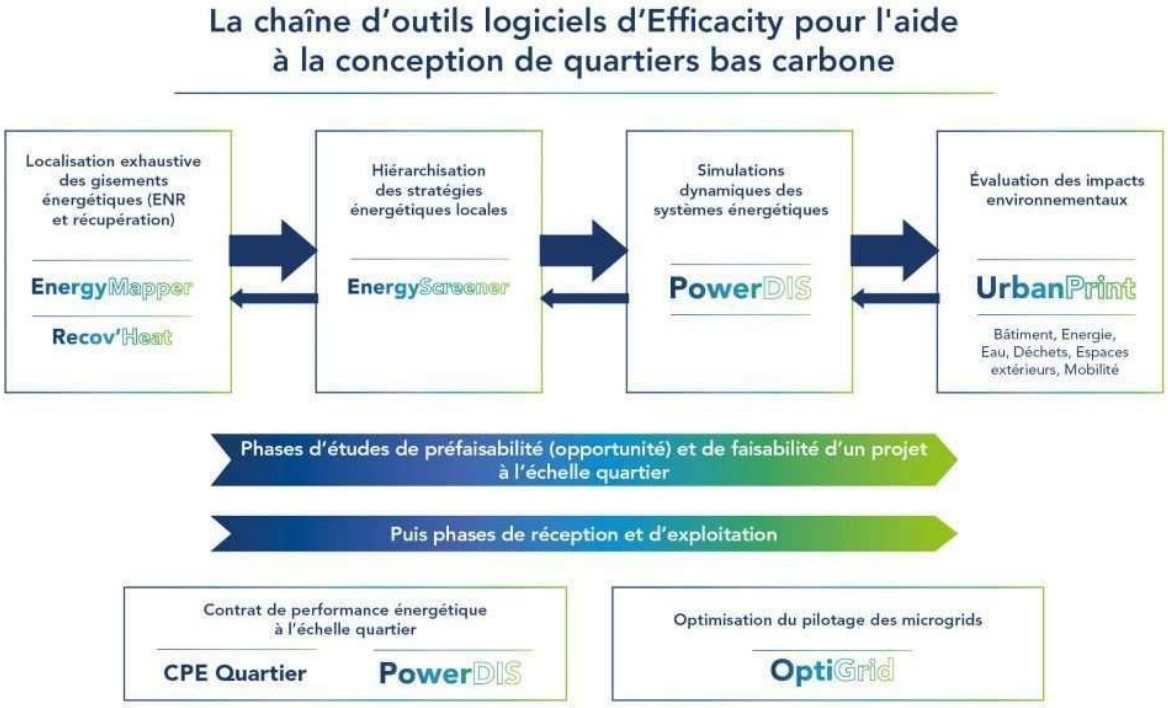


Figure 1 Suite logicielle d'Efficacity

En outre, cet axe accompagne également les collectivités dans leurs projets urbains et opérations pilotes pour qu'ils participent efficacement à la stratégie énergétique définie par exemple dans le PCAET.

Dans le cadre de cette entreprise, un outil qui permettra la création de clusters à partir de la consommation électrique d'un bâtiment est très pertinent pour les applications du lot QBC1, car il permet une caractérisation rapide des besoins et consommations énergétiques sans analyse détaillée et modélisation physique des bâtiments.

Cette solution pourrait être utilisée dans le développement des applications en termes d'analyse de données pour la création de modèles énergétiques de bâtiments.

La performance qui dans ce cas est représenté par la fiabilité des résultats est mesurable en validant les résultats de classification sur des bâtiments connus et pré-étudiés pour obtenir le bon nombre de groupes de profils et la bonne répartition des profils dans ces groupes.

Pour l'instant on dispose d'une expertise ingénieur métier bâtiments et efficacité énergétique pour analyser les résultats du clustering et les valider en amont de la modélisation.

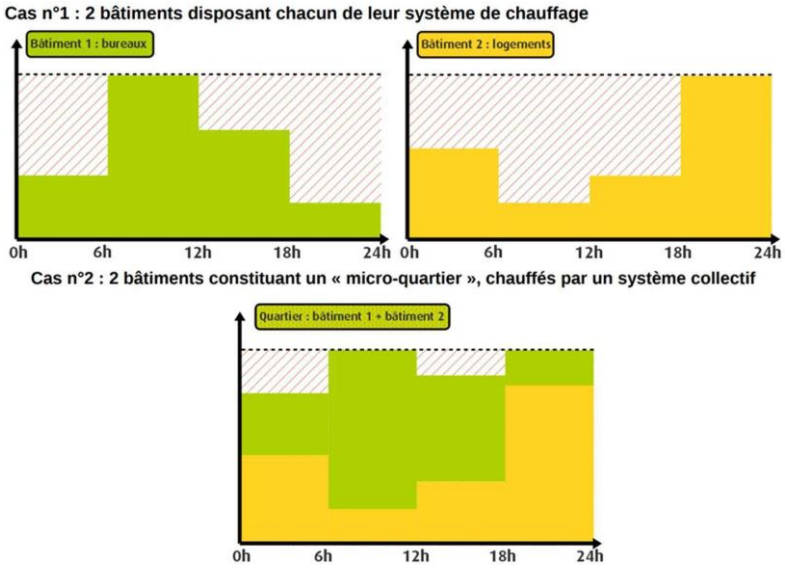
Pour la résolution manuellement du problème, il y a une exploration manuelle et graphique des données pour l'indentification des groupes de profils (difficulté de représentation et de mise en avant des groupes de profils), analyse de jours « types » -- jour ouvré, weekend, jour de vacances, etc. – avec découpage manuel de ces jours « types » (choix des jours biaisé, impossibilité de mettre en avant d'autres typologies).

2. Construction et récupération de la donnée

Dans le contexte général de la transition énergétique des villes, plusieurs logiciels de simulation sont développés par Efficacity pour l'aide à la décision dans les projets d'aménagement urbain (valorisation des énergies renouvelables et de récupération, choix des systèmes énergétiques en fonction des besoins des bâtiments).

Sachant que ces logiciels cherchent à déterminer, en première approche, les besoins énergétiques des bâtiments, une base de données de profils de besoins énergétiques de bâtiments, qui prend en compte du foisonnement de besoins, a été créé et est essentielle pour leurs usages à l'échelle d'un quartier. Cette BDD a été développée dans le lot « QBC 1.1 — Données bâtimentaires pour la modélisation multi-énergie et est enrichie et mise à jour régulièrement.

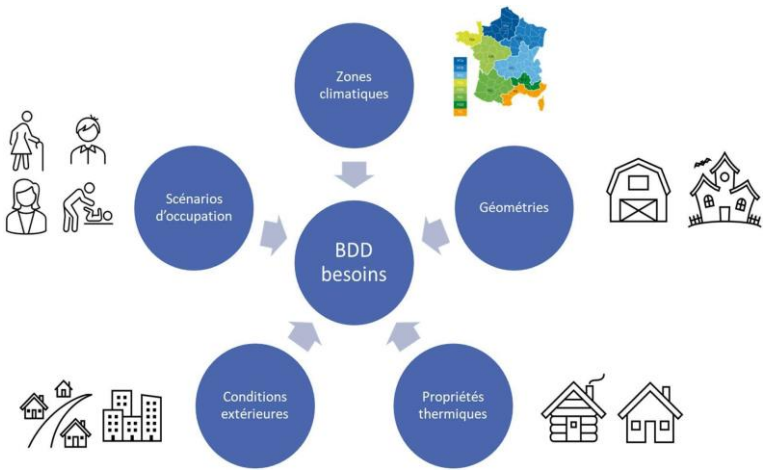
L'effet de foisonnement se caractérise par une diminution de l'intermittence du puisage d'énergie et un lissage des pics de besoins. Un lissage des pics permet d'éviter un surcoût d'investissement comme beaucoup de systèmes énergétiques sont dimensionnés pour répondre au pic le plus élevé. La figure suivante montre un exemple simplifié de foisonnement.



Dans le contexte d'Efficacity, il est essentiel de prendre en compte l'effet de foisonnement dans les outils qui sont utilisés en phase de préfaisabilité, car il permet de donner une première estimation ajustée du dimensionnement des installations énergétiques. **C'est pourquoi cette base de données de profils de besoins foisonnés est un élément important aux logiciels utilisés en phase de préfaisabilité.**

Echantillon et profils de bâtiments

Des échantillons représentatifs pour chaque typologie de bâtiment ont dû être crées. Plus précisément, les différents aspects retrouvés dans la figure suivante doivent être couverts :



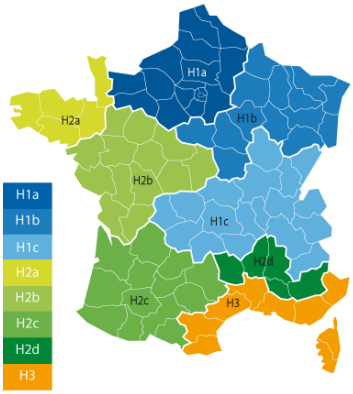
L'ensemble des bâtiments est décrit avec différents attributs comme les zones géographiques, les typologies (logements, bureaux, commerces, etc.), les années de construction, les géométries, les matériaux, les profils d'occupation, etc. Ces attributs sont en fait demandés par le modèle physique du logiciel de simulation énergétique dynamique et ils sont soit renseignés directement par une BDD géolocalisée, soit par une étape d'enrichissement des données.

Calcul des besoins énergétiques de bâtiments

Une fois que tous les bâtiments sont complètement caractérisés, pour déterminer leurs besoins énergétiques (chaud, froid, eau chaude sanitaire, électricité spécifique), nous simulons ces bâtiments avec un logiciel qui permettra d'établir des profils de besoins au pas de temps horaire sur une année.

Ce logiciel s'appelle DIMOSIM (Dlstrict MOdeller and SIMulator) et est développé par le CSTB (Centre Scientifique et Technique du Bâtiment). C'est un outil qui permet d'effectuer des simulations énergétiques à l'échelle d'un quartier.

Dans le cadre de ce projet on va utiliser une partie de cette base de données, et plus précisément, des données qui caractérisent des bâtiments du secteur tertiaire (bureaux) qui font partie de la région H2b (cf. à l'image suivante).(Ile de France)



Pour pouvoir avoir accès à ces données il y a eu besoin d'une *autorisation d'utilisation.* Pourtant il n'y a pas d'accès aux codes permis de simuler les données et pas d'autorisation de diffusion des données utilisées pour la classification.

La data est récupérée sous forme de fichiers .csv et il n'y a pas *de données sensibles*.

Pour cette étude nous avons accès à des données de caractérisation pour 389 bâtiments du secteur tertiaire, des bureaux, ainsi que et des données de besoin d'électricité et le besoin d'électricité pour le chauffage pour chaque bâtiment pour chaque heure, pour une période d'une année.

Les fluctuations constatées dans les données de consommation d'énergie d'un bâtiment du secteur résidentiel ou du secteur tertiaire peuvent être expliquées par divers facteurs, dont les différences d'occupation de l'espace au fil du temps. Les variations d'utilisation des locaux, telles que des périodes de pic d'activité ou des périodes de faible utilisation, peuvent influencer la consommation d'énergie. De plus, l'enrichissement stochastique des données, effectué à l'aide de réseaux bayésiens, pourrait également contribuer à ces variations. Les réseaux bayésiens, en modélisant les relations probabilistes entre différentes variables, peuvent introduire une composante aléatoire dans les données, ce qui peut se refléter dans les fluctuations observées de la consommation énergétique du bâtiment résidentiel. Ainsi, ces deux aspects, les différences d'occupation et l'enrichissement stochastique par des réseaux bayésiens, peuvent jouer un rôle crucial dans la compréhension des variations de la consommation d'énergie au fil du temps.

Tous ces données sont retrouvées dans des fichiers csv. A partir de ces fichiers csv, nous avons construit des fichiers .xlsx et le contenu de ces fichiers est le suivant :

- Fichier 1 : liste les bâtiments du secteur tertiaire, bureaux, et pour chaque bâtiment des données de caractérisation d'un bâtiment, par exemple : la surface au sol (area), le périmètre (length), l'année de construction, la surface habitable, la hauteur du bâtiment, la surface par occupant (m^2 /occ), le nombre des bureaux etc.
- Fichier 2 : liste le besoin d'électricité pour chaque bâtiment bureau, en décours d'une année
- Fichier 3 : liste le besoin d'électricité pour le chauffage, pour chaque bâtiment bureau, en décours d'une année

  Pour ce projet, nous allons utiliser des données des fichier 2. En sachant que le fichier 2 est construit à partir du fichier 1, et que dans les étude nous faisons appel aux données du fichier 1 nous allons présenter aussi ce fichier, par la suite.

3. Récupération et exploration de la data

Comme mentionné précédemment, nous disposons de six fichiers au format .csv qui ont été convertis en fichiers .xlsx. Dans cette section, nous entreprendrons une analyse détaillée des données contenues dans ces fichiers. Notre objectif est d'extraire des informations significatives et de tirer des conclusions pertinentes à partir de ces ensembles de données.

Préparation de l'environnement de travail

Pour pouvoir travailler ces données un environnement python de travail a été créé.

Pour la création de l'environnement nous avons besoin que python et pip soient installé.

Pour commencer le projet nous allons créer un venv avec la commande suivante :

python -m venv nomenv.

Pour activer ensuite ce venv nous utilisons la commande *source nomenv/bin/activate.*

Nous installerons par la suite tous les packages avec *pip install nomdupackage.*

Un des packets installés dans ce projet est *scikit-learn*. Pour l'installer il faut exécuter *pip install scikit-learn,* dans l'environnement active.

Une bibliotheque utilisée dans ce projet est tslearn. Pour l'installer il faut exécuter *pip install tslearn,* dans l'environnement active.

Fichier BU_all_buildings_IDF.xlsx

Ce fichier contient les données de caractérisation pour 396 bâtiments. Pour ces 396 bâtiments nous avons aussi des données sur les besoins d'électricité, et les besoins en électricité pour le chauffage, pour chaque bâtiment et chaque heure, sur une période d'une année (2019).

Pour pouvoir ouvrir et charger les données des fichiers .xlsx, a été construite la fonction `get_data_for_consommation_type` qui prend comme paramètre consommation_type comme dans le code suivant :

```python
def get_excel_file_data(file_name):
    script_directory = os.path.dirname(os.path.realpath(__file__))
    data_directory = os.path.join(
        script_directory, "..", "data"
    )  # Path to the directory containing the data files

    # Selection of the file based on the consumption type
    absolute_path = os.path.join(
        data_directory, file_name
    )  # Absolute path to the data file

    data = pd.read_excel(absolute_path)  # Reading data from the Excel file
```

```
    return data
```

Pour charger les données du fichier, nous allons appeler cette fonction en lui passant ‘buildings’ comme attribut :

```
df= get_excel_file_data()
```

Pour explorer les données contenues dans le DataFrame, nous allons utiliser les fonctions `head()` et `describe()`.

```
df = data_excel.copy()
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 396 entries, 0 to 395
Data columns (total 42 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   ID                         396 non-null    object
 1   bou                        396 non-null    int64
 2   caf                        396 non-null    int64
 3   chauf                      396 non-null    int64
 4   ecr                        396 non-null    int64
 5   mfd                        396 non-null    int64
 6   pcf2e                      396 non-null    int64
 7   port1e                     396 non-null    int64
 8   nb_equip                   396 non-null    int64
 9   nb_equip_bureau            396 non-null    int64
 10  nb_equip_autre             396 non-null    int64
 11  height                     396 non-null    float64
 12  floor_count                396 non-null    int64
 13  ExteriorWall_U_value       396 non-null    float64
 14  ExteriorRoof_U_value       396 non-null    float64
 15  ExteriorFloor_U_value      396 non-null    float64
 16  ExteriorWall_window_share  396 non-null    float64
 17  ExteriorWall_window_type   396 non-null    object
 18  comfort_heating_set_point  396 non-null    float64
 19  comfort_cooling_set_point  396 non-null    float64
...
 40  bd_surface_par_occ         396 non-null    float64
 41  hbs_surface_par_occ        396 non-null    float64
dtypes: float64(28), int64(11), object(3)
memory usage: 130.1+ KB
```

```
df.head()
```

	ID	bou	caf	chauf	ecr	mfd	pcf2e	port1e	nb_equip	nb_equip_bureau	...	Pmax_cl_par_m2	annuel_el_par_m2	Pmax_el_par_m2	annee_construction	effectif	nb_equip_par_m2	nb
0	BATIMENT0000000000449256	20	18	0	15	8	11	67	139	101	...	48.964595	16.444503	5.469096	1986.0	74.0	0.077554	
1	BATIMENT0000000000449819	120	134	0	404	82	275	251	1266	1012	...	53.930219	15.809512	4.906805	1993.0	502.0	0.102556	
2	BATIMENT0000000240289285	147	164	0	494	101	337	307	1550	1239	...	61.992223	62.169510	24.472787	1990.0	582.0	1.018106	
3	BATIMENT0000000240289639	11	13	0	38	8	26	24	120	96	...	44.438954	16.732151	5.850676	2014.0	43.0	0.093402	
4	BATIMENT0000000240290871	20	18	0	16	8	11	68	141	103	...	54.231608	17.286522	5.909568	1995.0	75.0	0.084678	

5 rows × 42 columns

```
df.describe()
```

	bou	caf	chauf	ecr	mfd	pcf2e	port1e	nb_equip	nb_equip_bureau	nb_equip_autre	...	Pmax_cl_par_m2	annuel_el_par_m2	Pmax_el_par_m2	annee_construc
count	396.000000	396.000000	396.0	396.000000	396.000000	396.000000	396.000000	396.000000	396.000000	396.000000	...	396.000000	396.000000	396.000000	396.00
mean	80.734848	81.994949	0.0	150.888889	41.535354	103.638889	217.747475	676.540404	513.810606	162.729798	...	51.427282	20.217850	7.564638	1990.51
std	152.068449	149.257640	0.0	238.026895	69.672916	163.324445	464.201153	1157.882815	862.879065	300.834957	...	9.595772	17.275369	6.775197	25.75
min	6.000000	6.000000	0.0	7.000000	4.000000	5.000000	12.000000	60.000000	46.000000	12.000000	...	19.674044	10.820489	3.258104	1828.00
25%	11.000000	11.000000	0.0	30.000000	6.000000	21.000000	29.500000	96.000000	77.000000	21.750000	...	45.694673	13.914004	4.870908	1986.00
50%	26.500000	26.000000	0.0	66.000000	15.000000	45.000000	67.500000	235.500000	184.500000	52.000000	...	51.369623	15.912472	6.323297	1997.00
75%	74.000000	73.250000	0.0	146.750000	40.000000	100.500000	212.500000	633.000000	499.000000	147.000000	...	56.060131	19.696159	7.616647	2006.00
max	1583.000000	1466.000000	0.0	1822.000000	668.000000	1245.000000	5203.000000	11438.000000	8389.000000	3049.000000	...	100.037116	216.061064	84.547891	2016.00

8 rows × 39 columns

En executant

```
print(len(data_df.index))
```

Nous obtenons : 396 ce qui signifie que nous avons les données pour 396 bâtiments.

En ce qui concerne les données manquantes, pour ce fichier on a 0 comme résultat à

```
data_df.isnull().sum().sum()
```

donc 0 données manquantes.

En exécutant, après, le code :

```
# iterating the columns
for col in data_df.columns:
    print(col)
```

Nous obtiendrons :

```
'ID',
 'bou',
 'caf',
 'chauf',
 'ecr',
 'mfd',
 'pcf2e',
 'port1e',
 'nb_equip',
 'nb_equip_bureau',
 'nb_equip_autre',
 'height',
 'floor_count',
 'ExteriorWall_U_value',
 'ExteriorRoof_U_value',
 'ExteriorFloor_U_value',
 'ExteriorWall_window_share',
 'ExteriorWall_window_type',
 'comfort_heating_set_point',
 'comfort_cooling_set_point',
 'geometry',
 'area',
 'length',
 'surface_utile',
 'hsp_moyenne',
 'annuel_ve_ch_par_m2',
 'annuel_ve_cl_par_m2',
 'Pmax_ve_ch_par_m2',
 'Pmax_ve_cl_par_m2',
 'annuel_ch_par_m2',
 'Pmax_ch_par_m2',
 'annuel_cl_par_m2',
 'Pmax_cl_par_m2',
 'annuel_el_par_m2',
 'Pmax_el_par_m2',
 'annee_construction',
 'effectif',
 'nb_equip_par_m2',
 'nb_equip_bureau_par_m2',
```

Nous allons voir par la suite ce que chaque donnée représente et les types de données :

Voici chaque attribut et sa signification :

- ID': Identifiant unique du bâtiment.
- 'bou': Nombre de bouilloires
- 'caf': nombre des machines à café
- 'chauf': nombre de chauffage d’appoint
- 'ecr': nombre d'écrans ou de moniteurs dans le bâtiment.
- 'mfd': nombre d imprimantes multifonctionnel dans le bâtiment.
- 'pcf2e': nombre des pc fixes avec 2 ecrans fixes
- 'port1e': nombre des pc portables avec 1 ecran fixes
- 'nb_equip': Nombre total d'équipements dans le bâtiment.
- 'nb_equip_bureau': Nombre d'équipements bureautiques
- 'nb_equip_autre': Nombre d'équipements non-bureautiques
- 'height': Hauteur du bâtiment.
- ‘area’: emprise au sol
- 'floor_count': Nombre d'étages dans le bâtiment.

- 'ExteriorWall_U_value': Valeur U des murs extérieurs (indicateur de l'efficacité énergétique).
- 'ExteriorRoof_U_value': Valeur U du toit extérieur.
- 'ExteriorFloor_U_value': Valeur U du sol extérieur.
- 'ExteriorWall_window_share': Proportion de fenêtres par rapport aux murs extérieurs.
- 'ExteriorWall_window_type': Type de fenêtres sur les murs extérieurs.
- 'comfort_heating_set_point': Point de consigne de chauffage confortable.
- 'comfort_cooling_set_point': Point de consigne de refroidissement confortable.
- 'geometry': La forme géométrique du bâtiment.
- 'area': Superficie totale du bâtiment.
- 'length': périmètre de la géométrie d'emprise
- 'surface_utile': Surface habitable totale
- 'hsp_moyenne': Hauteur moyenne sous plafond.
- 'annuel_ve_ch_par_m2': Besoins énergétiques annuels pour le chauffage par mètre carré.
- 'annuel_ve_cl_par_m2': Besoins énergétiques annuels pour le refroidissement par mètre carré.
- 'Pmax_ve_ch_par_m2': Puissance maximale nécessaire pour le chauffage par mètre carré.
- 'Pmax_ve_cl_par_m2': Puissance maximale nécessaire pour le refroidissement par mètre carré.
- 'annuel_ch_par_m2': Consommation annuelle d'énergie pour le chauffage par mètre carré.
- 'Pmax_ch_par_m2': Puissance maximale nécessaire pour le chauffage par mètre carré.
- 'annuel_cl_par_m2': Consommation annuelle d'énergie pour le refroidissement par mètre carré.
- 'Pmax_cl_par_m2': Puissance maximale nécessaire pour le refroidissement par mètre carré.
- 'annuel_el_par_m2': Consommation annuelle d'énergie électrique par mètre carré.
- 'Pmax_el_par_m2': Puissance maximale nécessaire en électricité par mètre carré.
- 'annee_construction': Année de construction du bâtiment.
- 'effectif': Nombre d'employés dans le bâtiment.
- 'nb_equip_par_m2': Nombre d'équipements par mètre carré.
- 'nb_equip_bureau_par_m2': Nombre d'équipements dans les bureaux par mètre carré.
- 'nb_equip_autre_par_m2': Nombre d'équipements dans d'autres parties du bâtiment par mètre carré.
- 'bd_surface_par_occ': Surface de bureau par occupant.
- 'hbs_surface_par_occ': Surface de bureaux imosable par occupant.

Toutes les données récupérées du fichier buidings.xlsx sont **de type float, sauf l'ID qui est de type string.**

Fichiers bureau_electricity.xlsx et heating_electricity.xlsx

Ces fichiers ont le même format des données.

Chaque de ces fichiers aura une colonne nommée *Datetime* et des colonnes correspondant aux IDs de chaque bâtiment conf. à la colonne ID du fichier buildings.xlsx.

- Le fichier bureau_electricity.py liste le besoin d'électricité pour chaque bâtiment, en décours d'une année.

- Le fichier heating_electricity.xlsx liste le besoin d'électricité pour le chauffage pour chaque bâtiment, pour chaque heure, en décours d'une année

Les colonnes *Datetime* contiennent des données en série temporelle s'étalant sur une année (2019), avec un pas de temps d'une heure. En ce qui concerne les données de consommation d'électricité, elles sont de type float et représente la consommation d'électricité en kWh pour chaque heure de l'année 2019.

Nous allons explorer par la suite les données du fichier bureau_electricity.py, en sachant que les autres fichiers ont le même format.

```
# The function takes a consumption type as an argument and returns the
corresponding data
def get_excel_data(consommation_type):
    # Mapping of consumption types to numerical values for ease of processing

    consommation_types = {
        "bureau_electricity": 1,
        "heating_bureau": 2
    }

    # Checking if the consumption type is valid
    if consommation_type not in consommation_types:
        raise ValueError("Invalid consommation_type value.")

    consommation_type = consommation_types[
        consommation_type
    ]  # Using the mapping to get the numerical value

    script_directory = os.path.dirname(os.path.realpath(__file__))
    data_directory = os.path.join(
        script_directory, "..", "data"
    )  # Path to the directory containing the data files

    # Selection of the file based on the consumption type
    file_names = {
        1: "bureau_electricity.xlsx",
        2:"BU_heating_demand_IDF.xlsx"
    }
    file_name = file_names[
        consommation_type
    ]  # Selecting the appropriate file name based on the consumption type

    absolute_path = os.path.join(
        data_directory, file_name
    )  # Absolute path to the data file

    data = pd.read_excel(absolute_path)  # Reading data from the Excel file

    return data


data_excel = get_excel_data("bureau_electricity")

df = data_excel.copy()
df.info()
```

Après avoir récupéré les données du fichier avec la fonction déjà présenté en début de ce chapitre, nous allons explorer le dataframe ainsi obtenu :

Les collonnes du dataframe aurons le type suivant :

```
data_df.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8760 entries, 0 to 8759
Columns: 397 entries, Datetime to BATIMENT0000002207357911
dtypes: float64(396), object(1)
memory usage: 26.5+ MB
```

	Datetime	BATIMENT0000000000449256	BATIMENT0000000000449819	BATIMENT0000000240289285	BATIMENT0000000240289639	BATIMENT0000000240290871	BATIMENT0000000240291408	B
0	2019-01-01 00:00:00+01:00	0.087154	0.048175	0.433317	0.057333	0.119203	0.046176	
1	2019-01-01 01:00:00+01:00	0.086219	0.045624	0.404004	0.057292	0.101625	0.047186	
2	2019-01-01 02:00:00+01:00	0.085622	0.045578	0.429938	0.056963	0.100764	0.046862	
3	2019-01-01 03:00:00+01:00	0.085868	0.045494	0.450665	0.056998	0.100790	0.046652	
4	2019-01-01 04:00:00+01:00	0.088139	0.054144	0.476290	0.058606	0.100598	0.058250	

5 rows × 397 columns

Nous pouvons voir qu'on a 397 collons donc les données de consommation d'électricité pour 396 bâtiments.

```
data_df.describe()
```

En exécutant

```
print(len(data_df.index))
```

Nous obtenons : 8760 ce qui correspond à chaque heure pour les 365 jours de l'année 2019.

Après avoir exécuté :

```
data_df.isnull().sum().sum()
```

nous obtenons 0, donc 0 données manquantes dans le fichier.

4. Etude de la consommation totale d’électricité, comportement de la consommation d’électricité par jours de la semaine des bâtiments du secteur tertiaire

Dans ce chapitre, nous examinerons la consommation d'électricité de 396 bâtiments au cours de l'année 2019, puis nous procéderons à une analyse de regroupement basée sur cette consommation d'électricité.

Cette étude est utile pour plusieurs raisons :

1. **Comportement de Consommation Différent** : Les clusters peuvent mettre en évidence des groupes de jours avec des comportements de consommation d'énergie similaires, ce qui peut être utile pour comprendre les tendances et les modèles.
2. **Catégorisation des différents "types" de bâtiments** présents dans la base de données des bureaux d'Efficacity. Cette classification offre une vision structurée des caractéristiques des bâtiments, facilitant ainsi une compréhension approfondie de la diversité présente.
3. **Evaluation de la diversité des niveaux de consommation d'électricité annuelle parmi les différents types de bâtiments**. En analysant les profils de consommation, elle fournit des insights précieux sur les disparités énergétiques entre les différentes catégories de structures. Cette double approche contribue à une meilleure compréhension des patterns de consommation et permet d'identifier des opportunités d'optimisation énergétique spécifiques à chaque type de bâtiment.
4. **Détecter des Anomalies** : Les clusters pourraient mettre en évidence des consommations atypiques ou des anomalies dans le comportement de la consommation d'énergie, ce qui peut être crucial pour la détection de problèmes potentiels ou d'opportunités d'optimisation
5. **Analyse Jour de la Semaine** : En clusterisant en fonction des la consommation pour chaque jour de la semaine, nous pouvons identifier des groupes de jours où la consommation d'énergie est particulièrement élevée ou basse. Cela peut être utile pour la planification de la gestion de l'énergie.
6. **Discrimination entre Jours de Semaine et Week-ends** : Les clusters pourraient aider à différencier les jours de semaine des week-ends en termes de comportement de consommation, ce qui peut être crucial pour les entreprises ou les infrastructures qui ont des schémas de consommation distincts pendant ces périodes.
7. **Identification de Tendances Hebdomadaires** : En analysant les clusters, on peut identifier des tendances hebdomadaires, par exemple, des jours spécifiques de la semaine où la consommation d'énergie est régulièrement plus élevée ou plus basse.
8. **Optimisation de la Gestion de l'Énergie** : En regroupant les jours avec des caractéristiques de consommation similaires, vous pourriez développer des stratégies de gestion de l'énergie plus efficaces et ciblées pour chaque groupe.
9. **Adaptation aux Profils de Consommation** : La clusterisation peut aider à adapter les stratégies de marketing ou d'approvisionnement en énergie en fonction des différents profils de consommation identifiés.

Clusteriser donc ces données peut fournir des informations précieuses pour mieux comprendre, optimiser et adapter la gestion de la consommation d'énergie.

Comme nous l'avons précisé et montré dans un chapitre précédent, nous disposons des données de consommation d'électricité pour chaque heure de l'année et pour chaque bâtiment. C'est pourquoi nous devrons effectuer quelques opérations préliminaires avant de passer à l'étape de regroupement.

Récupération, visualisation et préparation de la data

Pour récupérer la data la fonction présentée dans le chapitre 3 est utilisé :

```
data_excel = get_excel_data("bureau_electricity")
df = data_excel.copy()
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8760 entries, 0 to 8759
Columns: 397 entries, Datetime to BATIMENT0000002207357911
dtypes: float64(396), object(1)
memory usage: 26.5+ MB
```

```
df.describe()
```

	BATIMENT0000000000449256	BATIMENT0000000000449819	BATIMENT0000000240289285	BATIMENT0000000240289639	BATIMENT0000000240290871	BATIMENT0000000240291
count	8760.000000	8760.000000	8760.000000	8760.000000	8760.000000	8760.000
mean	1.877080	1.804739	7.096976	1.910063	1.973347	2.137
std	1.742185	1.689174	6.384155	1.998046	1.915161	2.011
min	0.000000	0.005537	0.018853	0.000000	0.000000	0.000
25%	0.355420	0.311975	2.955516	0.190111	0.332014	0.409
50%	0.836308	1.223467	4.396004	0.534240	0.784247	0.969
75%	3.859576	3.680434	12.866793	4.292988	4.190443	4.402
max	5.469096	4.906805	24.472787	5.850676	5.909568	6.157

8 rows × 396 columns

Nous n’avons pas des valeurs manquantes, pas des NaN.

Après la préparation de la data, nous obtiendrons la structure suivante :

Datetime	BATIMENT0000000000449256	BATIMENT0000000000449819	BATIMENT0000000240289285	BATIMENT0000000240289639	BATIMENT0000000240290871	E
2019-01-01 00:00:00	0.087154	0.048175	0.433317	0.057333	0.119203	
2019-01-01 01:00:00	0.086219	0.045624	0.404004	0.057292	0.101625	
2019-01-01 02:00:00	0.085622	0.045578	0.429938	0.056963	0.100764	
2019-01-01 03:00:00	0.085868	0.045494	0.450665	0.056998	0.100790	
2019-01-01 04:00:00	0.088139	0.054144	0.476290	0.058606	0.100598	
...	...	...	...	...	...	
2019-12-31 19:00:00	3.859109	3.478534	9.354594	4.384926	4.103083	
2019-12-31 20:00:00	3.813450	3.407296	8.741954	0.586865	4.011770	

Création et visualisation de la consommation journalière d’électricité pour tous les bâtiments

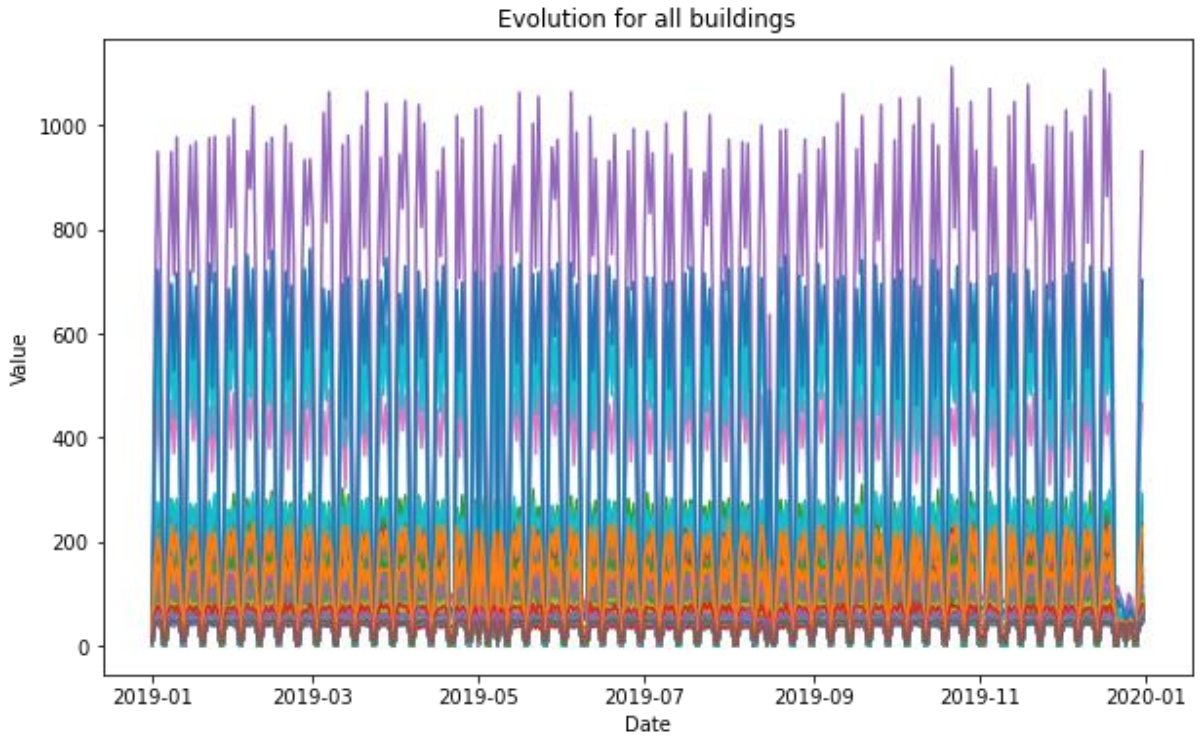


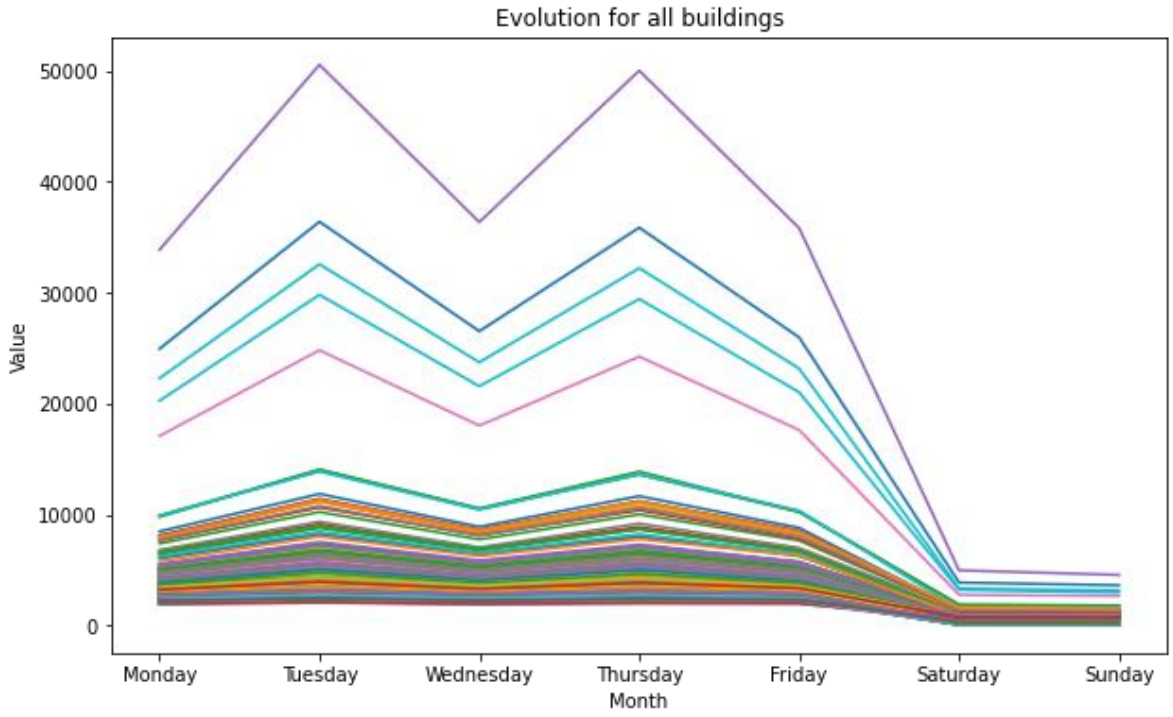
Figure 2 consommation journalière d'électricité par jour pour tous les bâtiments

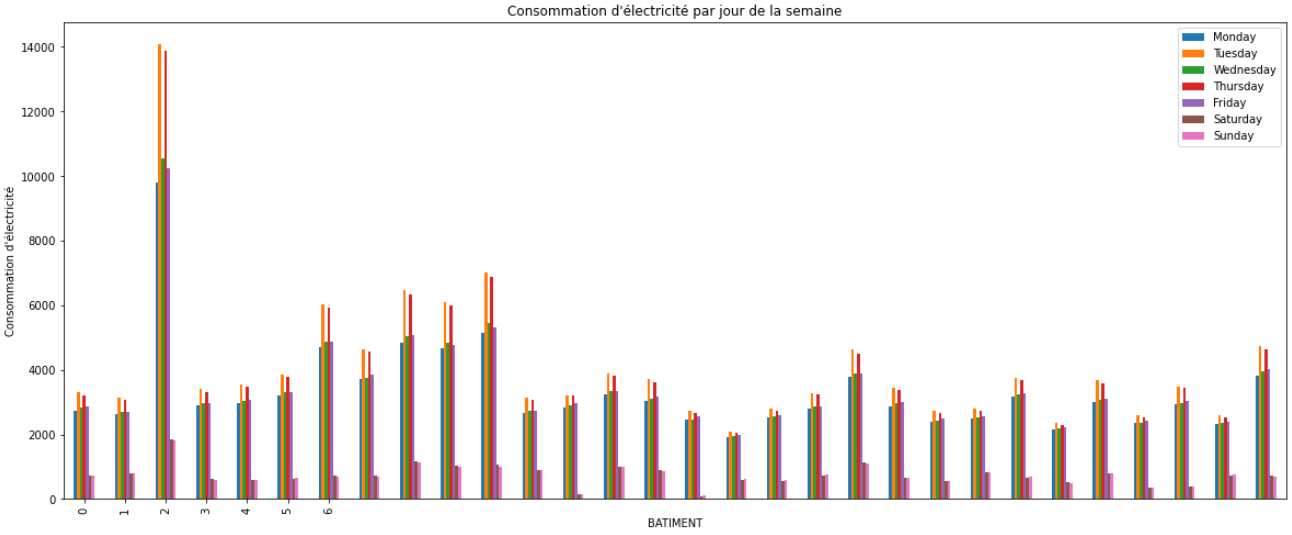
Nous allons construire **un nouveau dataframe avec la consommation totale d'électricité par jour de la semaine en décours d'une année**

Voici à quoi la nouvelle data se ressemble :

	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
BATIMENT0000000000449256	2745.164665	3317.550386	2837.298716	3214.670154	2866.090459	732.624896	729.818248
BATIMENT0000000000449819	2637.351379	3130.642186	2694.933973	3069.500057	2713.407684	780.364788	783.311629
BATIMENT0000000240289285	9788.947804	14062.767320	10549.564201	13863.965026	10243.492885	1858.653824	1802.118735
BATIMENT0000000240289639	2891.531604	3400.563904	2972.996880	3302.210520	2967.421782	608.870476	588.556295
BATIMENT0000000240290871	2969.454481	3549.454224	3037.276966	3480.455181	3057.576025	591.119080	601.186526

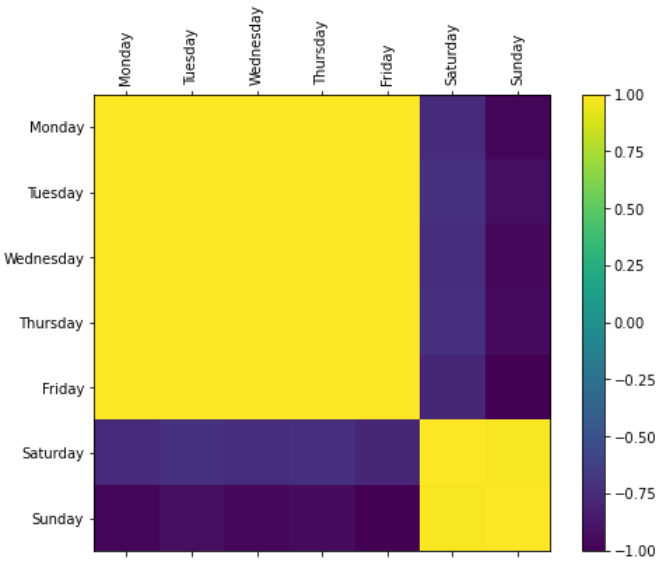
Représentation graphique





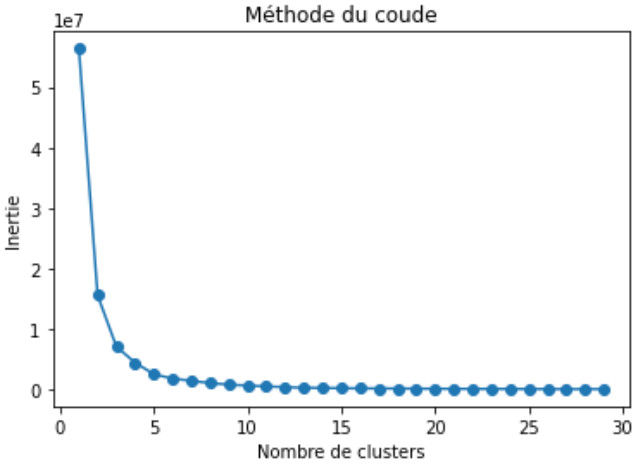
Correlation

Nous allons calculer la matrice de corrélation entre les variables de l'ensemble de données et ensuite nous allons créer un graphique de type heatmap pour visualiser cette matrice de corrélation. Chaque cellule de la heatmap représente le degré de corrélation entre deux variables, avec une échelle de couleurs allant de -1 à 1. Les étiquettes des axes sont ajustées pour correspondre aux noms des variables, et une barre de couleur est ajoutée sur le côté pour indiquer les valeurs de corrélation. Ce graphique aide à identifier les relations linéaires entre les variables du jeu de données.

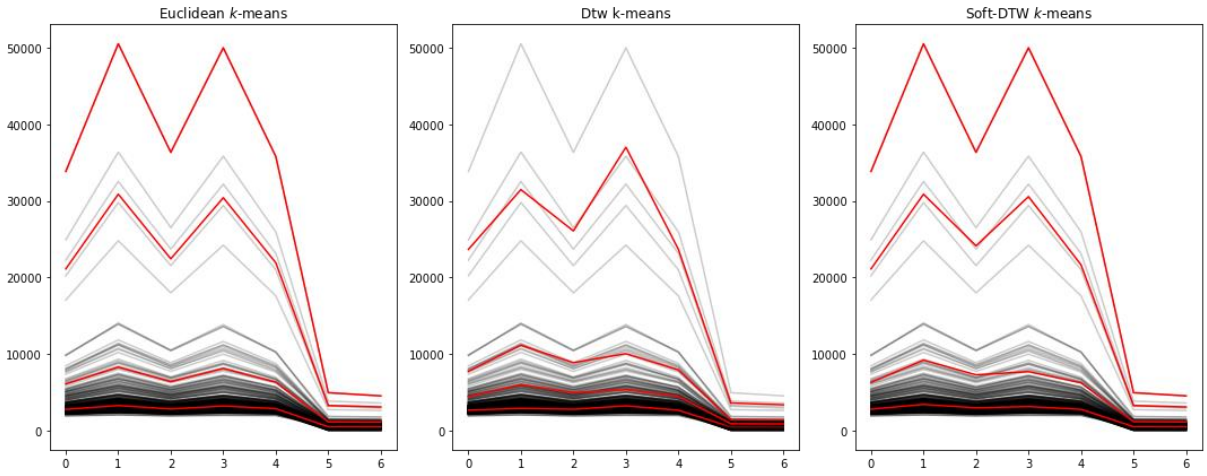


Comme on pouvait le supposer les jours « ouvré (de travail)» de la semaine présente une forte corrélation positive entre elles. De plus, "%Saturday" et "Sunday" ont également une forte corrélation positive entre elles.

Calcul du nombre optimal des clusters



Résultats de la clusterisation en utilisant des métriques différentes



Il y a des différences dans les résultats, mais nous pouvons voir de maintenant que le choix de la métrique sera faite entre la métrique euclidien et la métrique soft-dtw.

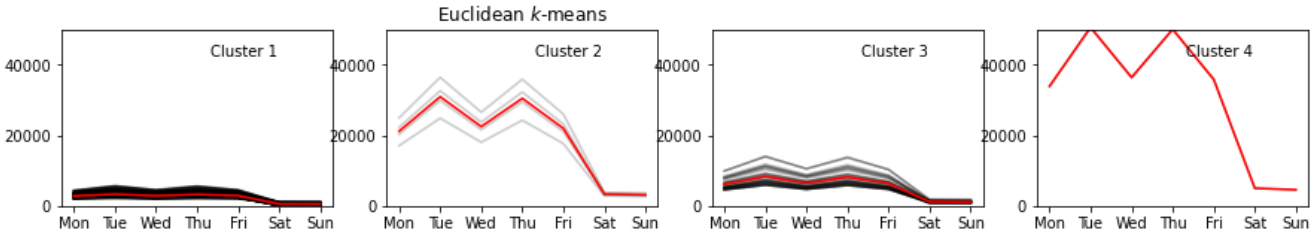
Choix de la métrique optimale

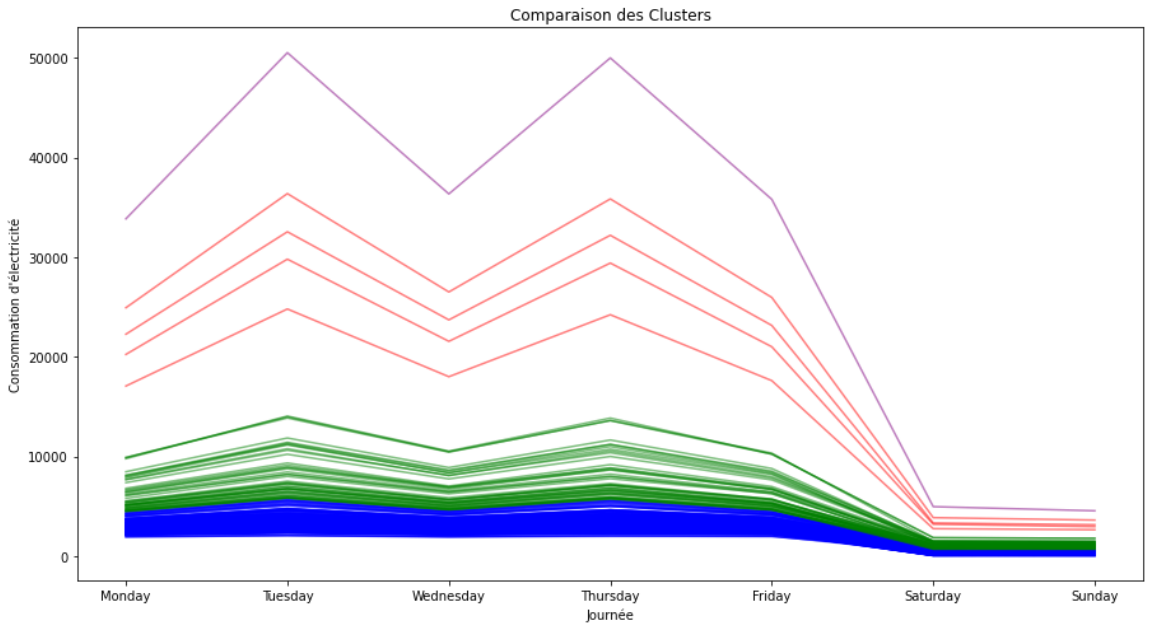
Nous allons utiliser l'inertie pour faire un choix entre les trois métriques utilisées.

Pour ce cas précis, la plus petite inertie est l'inertie pour la métrique euclidean, et par la suite nous allons étudier ces résultats de clusterisation.

```
Inertia euclidean metric :  4439521.956284885
Inertia dtw metric :  4600830.825043915
Inertia sdtw metric :  155054617533107.9
Inertie preferable : euclidean
```

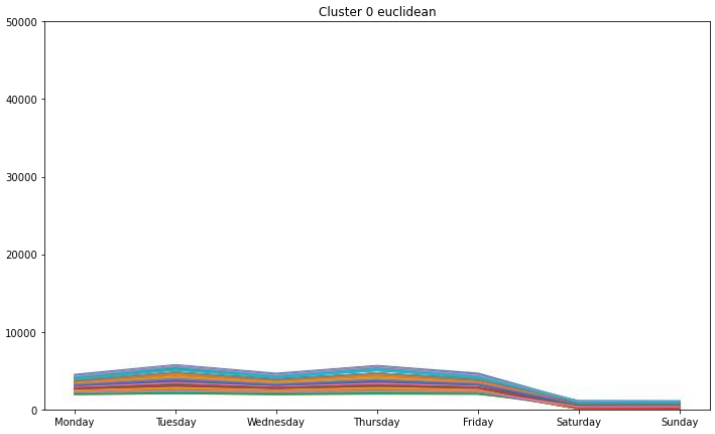
Distribution des bâtiments pour chaque cluster

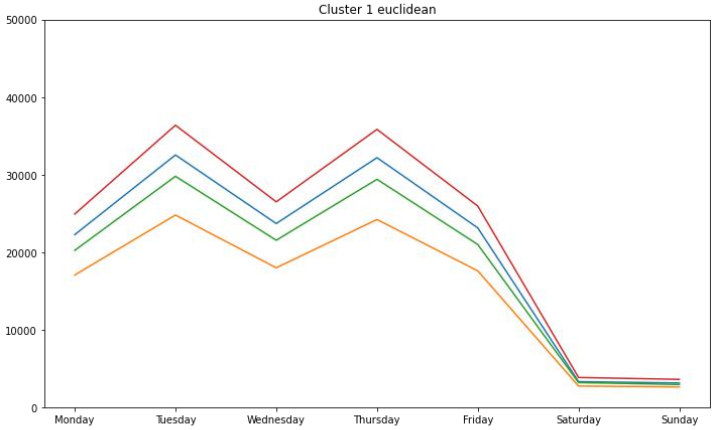


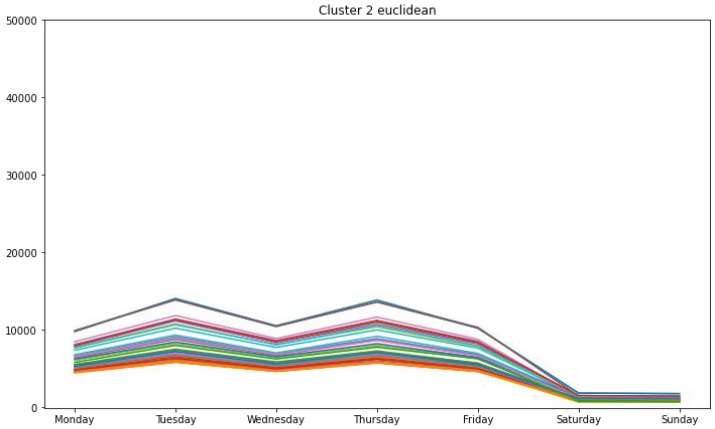


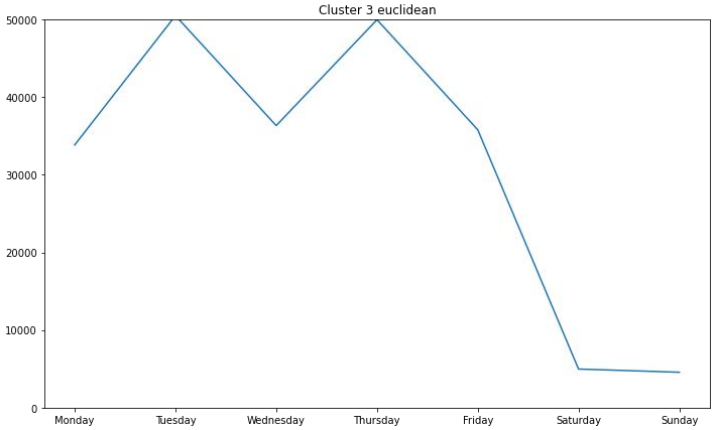
Dans cette image, chaque consommation pour un bâtiment appartenant à un cluster est représentée avec une couleur différente.

Nous allons voir dans les graphiques suivant, **la distribution des bâtiments pour chaque cluster :**

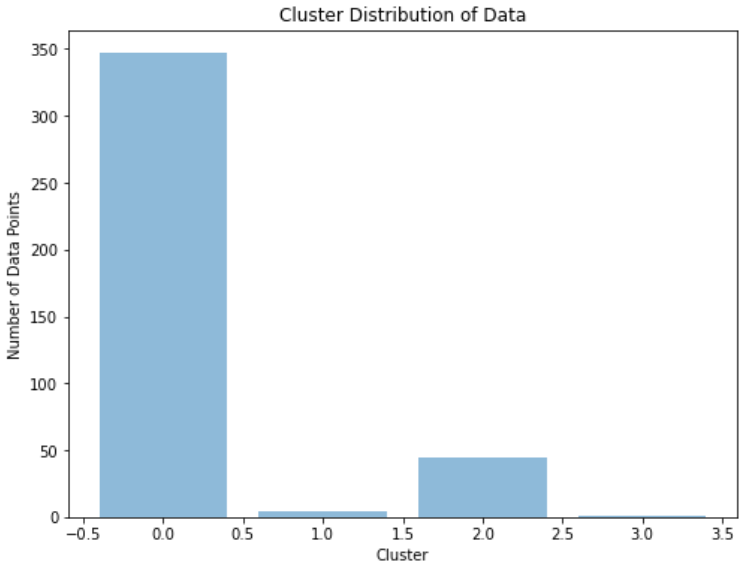

Cluster 0 euclidean
50000
40000
30000
20000
10000
0
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday


Cluster 1 euclidean
50000
40000
30000
20000
10000
0
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday


Cluster 2 euclidean
50000
40000
30000
20000
10000
0
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday


Cluster 3 euclidean
50000
40000
30000
20000
10000
0
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday

Distribution de la data



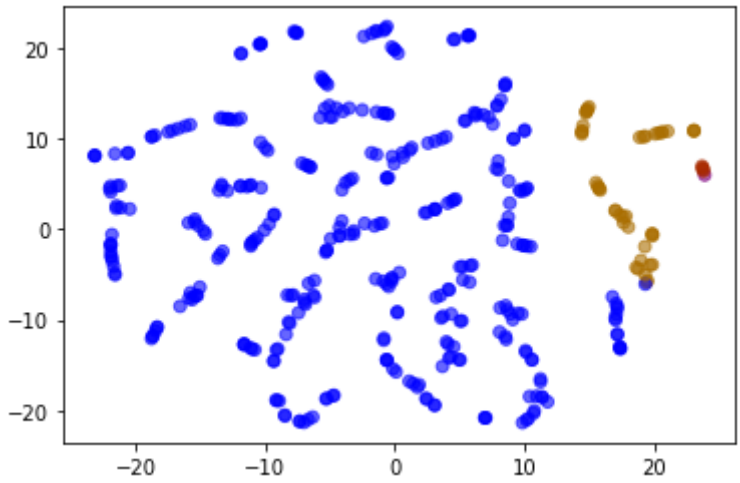
Il semble que le cluster 0 regroupe la majorité des bâtiments, suggérant une tendance générale à une consommation d'électricité similaire tout au long de la semaine. Cette observation suggère une stabilité dans les habitudes de consommation électrique, indiquant peut-être une efficacité énergétique globalement constante parmi ces bâtiments. Il est possible que les bâtiments de ce groupe partagent des caractéristiques similaires, contribuant à cette homogénéité dans les profils de consommation d'électricité*

Reduction de dimensions et visualisation de la data

L'évaluation correcte de la performance d'un clustering est toujours un problème ouvert, et pas seulement pour le clustering de séries temporelles. Comme il s'agit d'un problème non supervisé, nous n'avons pas de label pour établir des mesures de performance. La définition d'un bon clustering dépend du problème et est souvent subjective. Par exemple, le nombre de clusters, la taille des clusters, la définition des valeurs aberrantes, et la définition de la similarité entre les séries temporelles d'un problème sont tous des concepts qui dépendent de la tâche à accomplir et doivent être déclarés subjectivement.

Le résultat peut être évalué à l'aide de certaines mesures. La visualisation et les mesures scalaires sont les principales techniques d'évaluation de la qualité du clustering.

Reduction de dimensions avec t-SNE et visualisation de la data par cluster en 2 dimensions :



Les données semblent distribuées bien en fonction de leurs clusters.

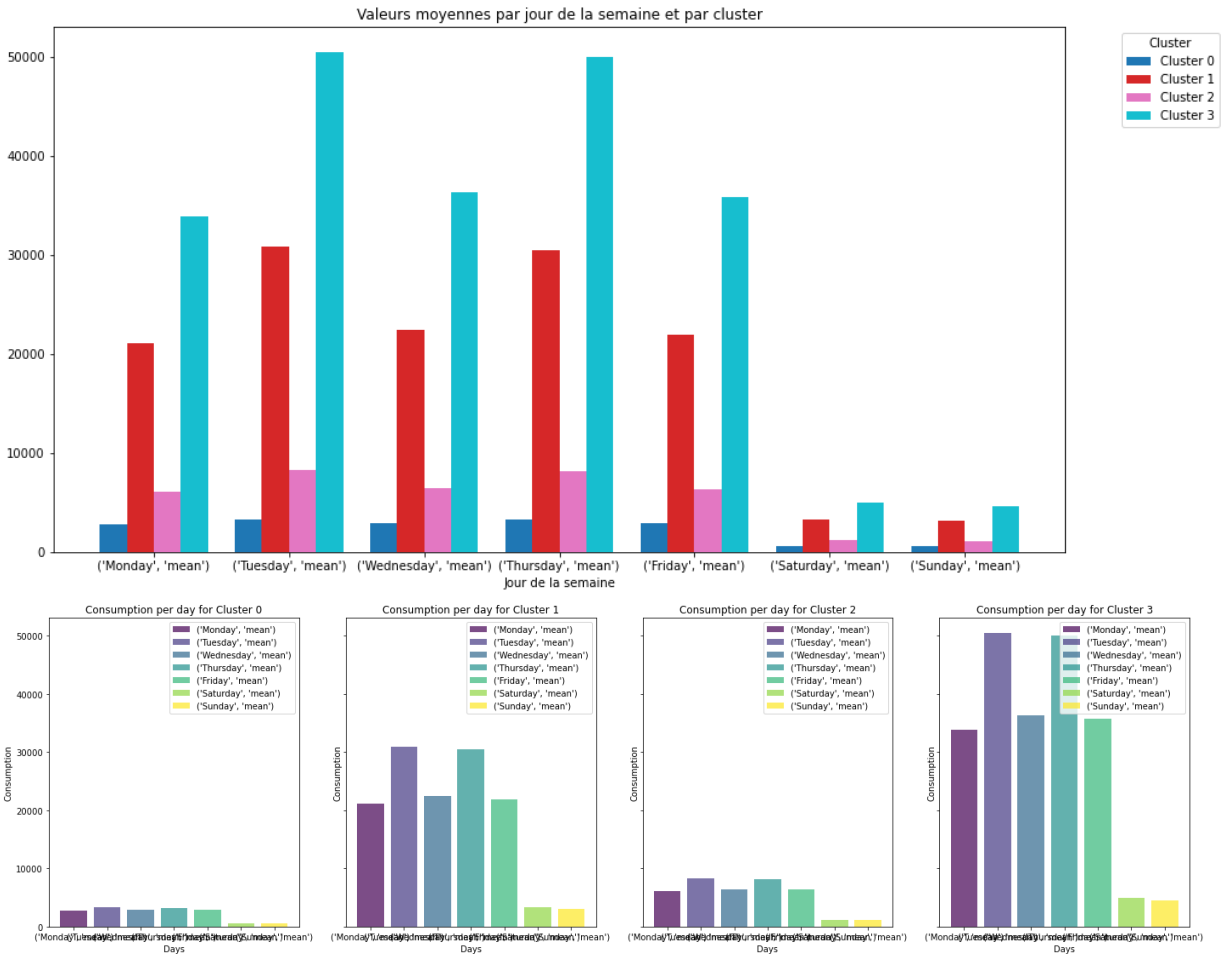
Valeurs moyennes par cluster

Lors de la validation des résultats des clusters, l'utilisation des moyennes par cluster peut être un moyen informatif de comprendre les caractéristiques moyennes des groupes formés par l'algorithme de clustering.

Voici les moyennes par cluster pour notre data :

	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
	mean	mean	mean	mean	mean	mean	mean
clusters							
0	2801.071896	3297.803572	2848.943250	3236.695914	2892.474700	533.762658	529.652572
1	21134.396093	30886.766741	22449.540855	30424.356504	21938.226099	3297.134671	3104.814899
2	6118.769816	8321.283163	6422.051785	8151.430072	6347.468759	1160.886080	1126.862240
3	33854.149677	50521.648846	36349.864490	49994.258638	35807.386858	4973.729322	4560.026112

Voici une représentation graphique des moyennes des consommation d’électricité par jour pour chaque cluster.



Analyse des résultats

Je souhaite préciser que cette étude a été aussi réalisé avec une normalisation préalable à la clusterisation des données et que les résultats obtenus ont été similaires.

En considérant que l'étude est réalisée pour la consommation d'électricité dans le domaine tertiaire (bureaux), les résultats de clustering semblent cohérents avec les tendances générales de consommation d'électricité dans ce secteur.

Les résultats du clustering sont représentés par quatre clusters.

L'observation générale pour chaque cluster suggère que la consommation d'électricité reste relativement stable tout au long de la semaine pour les jours de travails, avec des variations modérées.

Il est également notable que les pics de consommation sont significativement plus bas le samedi et le dimanche. Cette tendance est cohérente avec l'idée que les bureaux, représentés par ces clusters, sont principalement utilisés en semaine. Par conséquent, une consommation d'électricité plus élevée pendant les jours ouvrés et une diminution pendant le week-end reflètent une utilisation attendue des installations de bureau.

clusters	Mean_Workdays	Mean_Weekend
0	3015.397866	531.707615
1	25366.657258	3200.974785
2	7072.200719	1143.874160
3	41305.461702	4766.877717

Cluster 0 :

- Au sein de ce cluster, on observe que les bâtiments présentent une consommation d'électricité relativement modeste par rapport aux autres groupes.
- Les moyennes de consommation pour les jours de la semaine varient entre 2801 kWh et 3297 kWh, tandis que le week-end affiche une moyenne de 531 kWh.
- Cette tendance suggère que ce cluster englobe principalement des entreprises de petite taille, dont les besoins énergétiques sont plus limités en comparaison avec d'autres segments.

Cluster 1 :

- Ce cluster englobe les bâtiments affichant une consommation d'électricité élevée tout au long de la semaine, suggérant une correspondance avec des entreprises de plus grande envergure, caractérisées par un nombre plus élevé de bureaux par rapport aux autres édifices de l'étude. Cette tendance à la consommation d'électricité accrue est en cohérence avec la nature et l'envergure de ces entreprises.
- Les bâtiments de ce cluster présentent une consommation plus élevée que celle du cluster 0 et 3.

Cluster 2 :

- Au sein du deuxième cluster, nous observons une consommation d'électricité modérée par rapport aux autres groupes.
- Ces bâtiments présentent une consommation moyenne de 7032 kWh par jour ouvré et de 1143 kWh en weekend.
- Ces chiffres indiquent une correspondance avec le profil énergétique généralement associé aux entreprises de taille moyenne.
- La consommation modérée, tant en semaine qu'en fin de semaine, suggère une utilisation équilibrée de l'électricité, typique des entreprises de cette catégorie.
- Cette caractéristique peut être le reflet d'activités professionnelles plus constantes, avec une différence de consommation relativement limitée entre les jours ouvrés et le weekend.

Cluster 3 :

- Le troisième cluster représente un seul bâtiment caractérisé par une consommation d'électricité exceptionnellement élevée pendant les jours ouvrés.
- Ce bâtiment affiche une moyenne de 41,305 kWh par jour ouvré et de 4,766 kWh par jour de fermeture (weekend).

- Les résultats relatifs à ce bâtiment nécessitent une analyse approfondie pour comprendre les facteurs sous-jacents à cette consommation extraordinaire. Cette démarche est cruciale pour établir une conclusion éclairée, que celle-ci confirme la validité des résultats ou révèle d'éventuels problèmes dans le processus de simulation, indiquant ainsi une divergence significative entre les données simulées et la réalité.

Dans le cadre du processus de validation de la consommation d'électricité de chaque bâtiment, une analyse approfondie des caractéristiques propres à chaque bâtiment est essentielle.

C'est pourquoi la récupération des éléments caractéristiques pour chaque bâtiment et pour chaque cluster est effectuée dans le cadre de cette étude.

Cette démarche permet d'obtenir une compréhension approfondie des facteurs spécifiques influençant la consommation énergétique de chaque bâtiment, facilitant ainsi le processus de validation, assurant la précision des résultats obtenus mais ça permettra aussi de trouver les valeurs aberrantes de consommation d'électricité en fonction des caractéristiques de chaque bâtiment appartenant à un cluster donné.

Cluster 0 :

- L'analyse du cluster avec le plus grand nombre de bâtiments révèle une tendance générale de consommation d'électricité plus faible par rapport aux autres clusters. Cette observation peut être attribuée au fait que les bâtiments de ce cluster ont en moyenne moins d'équipements consommateurs d'électricité par rapport aux bâtiments des autres clusters.

- Cependant, il est crucial de noter quelques exceptions, notamment le bâtiment avec l'ID BATIMENT0000000000449819. Ce bâtiment se distingue par un nombre exceptionnellement élevé d'équipements, soit 1266, comparé aux valeurs prédominantes qui se situent en dessous de 200 équipements par bâtiment dans ce cluster. Cette disparité pourrait indiquer une configuration ou une utilisation particulière de ce bâtiment, nécessitant une attention particulière de la part des experts métier.

- L'identification de telles exceptions souligne l'importance de mener une analyse approfondie sur des cas particuliers pour comprendre les raisons des profils de consommation d'électricité divergents et valider ou détecter des anomalies dans le processus de simulation des données. Cela peut être crucial pour optimiser l'efficacité des algorithmes utilisées pour générer la consommation d'électricité et pour identifier des opportunités d'amélioration pour des cas spécifiques.

Cluster 2 :

- Les bâtiments repartis dans ce cluster suivent comme consommation annuelle les bâtiments du cluster 0.
- Ce cluster contient 44 bâtiments avec une consommation moyenne d'électricité en décours d'une année et ces bâtiments ont un nombre d'équipements plus élevés que les

bâtiments appartenant aux premiers clusters ce qui peut expliquer la consommation d'électricité plus élevés et aussi la répartition dans ce cluster.

Cluster 1 :

- Ce cluster n'a que 4 bâtiments.
- Tous ces bâtiments ont un nombre relativement élevé des consommateurs d'électricité ce qui montre que la consommation d'électricité de ces bâtiments est normale, et que la répartition dans ce cluster de ces bâtiments est bonne.

Cluster 3 :

- Le bâtiment correspondant à ce cluster a une consommation beaucoup plus élevée d'électricité que les autres.
- En analysant les données de caractérisation de ce bâtiment et les données des autres bâtiments appartenant aux autres clusters qui correspond à une consommation d'électricité moins élevé, nous pouvons conclure que ce bâtiment et ces résultats de consommation d'électricité nécessite une analyse plus détaillée par les experts métiers.
- Les raisons possibles de ces données seront :
    - Une erreur de la part des personnes qui ont généré ces données
    - Une erreur de la part du logiciel de simulation DIMOSIM.

5. Création des profils journalières

Nous cherchons à créer des clusters pour la consommation quotidienne d'électricité (parmi les deux types existants, choisis en fonction des préférences de l'utilisateur) pour l'un des 396 bâtiments de la liste pour le secteur tertiaire. Avant de passer à la phase de création de clusters, nous devons traiter les données extraites des fichiers .xlsx et créer un script qui :

1. Demande à l'utilisateur de fournir l'identifiant du bâtiment souhaité.
2. Demande à l'utilisateur de spécifier le type d'électricité à regrouper en clusters.
3. Récupère les données de consommation d'électricité pour ce bâtiment.
4. Travaille les données récupérées pour créer 365 lignes, chacune correspondant à une journée de consommation d'électricité avec un pas de temps d'une heure.

1.Pour répondre à la première question, la fonction suivante a été créé :

```
def getUserBuilding()
    buildings = get_excel_file_data("BU_all_buildings_IDF.xlsx")

    building_ids = buildings["ID"].tolist()

    while True:
        building_id = input("What building? (the building id)")

        # Check if the building id is valid
        if building_id in building_ids:
            break
        else:
            print("Invalid building id. Please try again.")

    return building_id
```

Avec cette fonction nous récupérons un id de bâtiment à l'utilisateur, et nous redemanderons tant que cet id n'est pas valide.

2.Pour la deuxième question nous utiliserons :

```
def get_consommation_type_wanted():
    file_names = {
        1: "bureau_electricity",
         2: "heating_bureau"
    }    while True:
        consommation_type = input(
            "What type of consommation we want to cluster? 1:bureau electricity :
2 : heating bureau"
        )

        if consommation_type in "1", "2"]:
            break
        else:
            print("Invalid choice. Please enter 1 or 2")

    return file_names[int(consommation_type)]
```

Cette fonction nous permet de récupérer le type d'électricité qu'on souhaite clustériser.

3. Pour le troisième point, nous allons appeler en passant comme paramètres l'id du bâtiment et la consommation d'électricité souhaité, la fonction suivante :

```
def get_Df_for_building(building_id, consommation_type):
    data = get_excel_data(consommation_type)
    copy_data = data.copy()

    consommation_for_building_wanted =
make_df_with_time_and_building_consommation(
        copy_data, building_id
    )

    return consommation_for_building_wanted
```

Nous avons déjà vu la fonction `get_excel_data` dans le chapitre précèdent. Elle récupérera sous fourme de dataframe les données d'un fichier excel souhaité.

Voici la fonction `make_df_with_time_and_building_cosommation(copy_data, building_id)` qui retournera un dataframe avec une colonne nomme Datetime et une colonne nommé électricité qui contiendra la consommation d'électricité pour le bâtiment avec l'id souhaité.

```
def make_df_with_time_and_building_consommation(data, building_column_name):
    data["Datetime"] = pd.to_datetime(data["Datetime"])

    df_building = data[["Datetime", building_column_name]]
    df_building.rename(columns={building_column_name: "electricity"}, inplace=True)

    return df_building
```

4. En ce qui concerne le quatrième point, répondre à cette question implique de faire face à un problème crucial lié au changement d'heure qui se produit en fin mars et en fin octobre.

En raison de ce changement d'heure, nous observons que le 31 mars comporte 23 heures et que le 27 octobre, 25 heures (2 fois l'heure 2). Par conséquent, deux scénarios se présentent :

1. **Un jour avec une valeur manquante.**

```
2136 2019-03-31 00:00:00     0.359621
2137 2019-03-31 01:00:00     0.333796
```

2. **Un jour avec une valeur en excès**.

```
7177 2019-10-27 02:00:00     0.187455
7178 2019-10-27 02:00:00     0.187455
```

Pour ce qui est des valeurs manquantes, deux approches sont envisageables :

1. **Reconstruire les données manquantes.**
2. **Supprimer le profil contenant des données manquantes.**

Étant donné la faible variation horaire dans notre jeu de données et le fait que nous ne reconstruirons qu'un seul point sur l'ensemble de l'année, nous pouvons considérer que toute incertitude introduite par la reconstruction est totalement négligeable. Une autre raison de reconstruire le profil manquant est le risque de supprimer un profil essentiel, notamment un profil spécifique notablement différent des autres, ce qui pourrait affecter les résultats de la clusterisation.

Pour la reconstruction, nous allons utiliser la méthode consistant à prendre la moyenne des valeurs précédant et suivant le point manquant. Nous allons également vérifier plusieurs cas pour évaluer si la valeur de cette heure pour les jours précédents de la même semaine diffère considérablement. Nous examinerons également un échantillon d'autres profils similaires pour garantir qu'il n'y a pas de variations significatives des valeurs au point de données que nous souhaitons reconstruire (à la même heure sur d'autres jours).

Cela nous permettra de nous assurer que les données reconstruites seront relativement fiables et que toute erreur introduite dans l'ensemble de données sera négligeable.

Pour trouver les situations où nous avons des données manquantes et pour ajouter une ligne avec la moyenne des voisins nous allons utiliser la fonction suivante :

```
def manage_missing_day_data(df):
    # Convert the datetime column to a specific format ("%Y-%m-%d %H:%M") and then
to datetime object
    df["Datetime"] = df["Datetime"].apply(lambda x: x.strftime("%Y-%m-%d %H:%M"))
    df["Datetime"] = pd.to_datetime(df["Datetime"])

    # Compute the time differences between consecutive rows in the Datetime column
    hour_changes = df["Datetime"].diff().fillna(pd.Timedelta(0))

    # Find the indices where the hour changes by 2 hours
    indices_hour_change = hour_changes[hour_changes ==
pd.Timedelta(hours=2)].index

    # Print the indices where the hour changes
    print("Indices of hour changes:")
    print(indices_hour_change)

    # Iterate through the indices of the hour changes
    for idx in indices_hour_change:
        # Compute the average consumption between the two existing values
        average_consumption = (df.loc[idx - 1, "electricity"] + df.loc[idx + 1,
"electricity"]) / 2
        # Append a new row with the adjusted datetime and the average consumption
        df = df.append(
            {
                "Datetime": df.loc[idx, "Datetime"] - pd.Timedelta(hours=1),
                "electricity": average_consumption,
            },
            ignore_index=True,
        )
    # Sort the dataframe by datetime and reset the index
    df = df.sort_values("Datetime").reset_index(drop=True)
    return df
```

Voici la liste des index correspondant aux données manquantes trouvés dans le dtaframe :

```
Indices des changements d'heure :
Int64Index([2138], dtype='int64')
```

Après la reconstruction nous aurons :

```
2137 2019-03-31 01:00:00     0.333796
2138 2019-03-31 02:00:00     0.260626
2139 2019-03-31 03:00:00     0.187455
```

Pour comparer cette valeur nouvelle introduite, nous allons regarder d'autre valeurs correspondant à la même heure dans les jours voisines :

```
                Datetime  electricity
1922 2019-03-22 02:00:00     0.285016
1946 2019-03-23 02:00:00     0.187455
1970 2019-03-24 02:00:00     0.187455
1994 2019-03-25 02:00:00     0.187455
2018 2019-03-26 02:00:00     0.325188
2042 2019-03-27 02:00:00     0.187455
2066 2019-03-28 02:00:00     0.187455
2090 2019-03-29 02:00:00     0.187455
2114 2019-03-30 02:00:00     0.187455
2138 2019-03-31 02:00:00     0.260626
2162 2019-04-01 02:00:00     0.187455
2186 2019-04-02 02:00:00     0.187455
2210 2019-04-03 02:00:00     0.187455
2234 2019-04-04 02:00:00     0.359621
2258 2019-04-05 02:00:00     0.187455
2282 2019-04-06 02:00:00     0.156213
2306 2019-04-07 02:00:00     0.187455
2330 2019-04-08 02:00:00     3.165714
2354 2019-04-09 02:00:00     0.187455
2378 2019-04-10 02:00:00     0.187455
```

Il n'y a pas de différence majeure entre la donnée qu'on vient d'ajouter et les autres voisins, donc nous allons considérer que les données reconstruites seront relativement fiables et que toute erreur introduite dans l'ensemble de données sera négligeable.

Pour le jour comportant une heure supplémentaire, nous avons le choix entre deux approches :

1. **Supprimer une heure et conserver 24 heures, ce qui équivaut à une journée complète.**
2. **Éliminer le profil contenant des données en plus.**

Nous avons opté pour la première méthode en supprimant une des deux valeurs correspondant à la même date et à la même heure. Cette décision a été motivée par la nécessité d'éviter tout risque de suppression d'un profil essentiel, notamment d'un profil spécifique distinct des autres, susceptible d'influer sur les résultats de la classification.

Étant donné que notre ensemble de données ne présente qu'un seul cas de ce type, la suppression d'une seule valeur, bien que pouvant introduire une certaine incertitude, reste négligeable au regard de l'ensemble des données.

Pour trouver et retirer les lignes dupliqué nous allons utiliser la fonction suivante :

```python
def manage_duplicates(df):
    # Check for duplicated entries with the same hour and date
```

```
    same_hour_and_date_check = df.duplicated(subset="Datetime", keep=False)

    # If there are duplicated entries with the same hour and date
    if len(df[same_hour_and_date_check]) > 0:
        # Create a new dataframe containing the duplicated entries
        result_df = df[same_hour_and_date_check]

        # Identify the index of the first duplicated entry and drop it from the
original dataframe
        index_to_drop = result_df.index[0]
        df = df.drop(index_to_drop)

    # Return the updated dataframe
    return df
```

Après avoir utilisé cette fonction pour notre dataframe et avoir retiré la ligne correspondant à l'heure en plus, nous allons avoir :

```
  7177 2019-10-27 01:00:00     0.187455
  7179 2019-10-27 02:00:00     0.187455
  7180 2019-10-27 03:00:00     0.187455
```

Nous allons passer après à la création des profils journaliers.

Pour ça, nous allons utiliser la fonction :

```
def make_day_profil (df):
    # Créer un nouveau DataFrame avec l'index souhaité
    new_df = pd.DataFrame(columns=range(24))

    # Remplir le nouveau DataFrame avec les valeurs appropriées
    for index, row in df.iterrows():
        date = row["Datetime"].strftime("%Y-%m-%d-%A")
        hour = row["Datetime"].hour
        new_df.loc[date, hour] = row["electricity"]

    return new_df
```

Une autre chose à noter est que nous devons également supprimer la dernière ligne du DataFrame, qui correspond à l'heure 00:00 du 2020-01-01. Cette valeur créerait sinon un profil journalier avec une seule valeur pour la première heure de la journée, avec des valeurs NaN pour le reste, ce qui n'est pas pertinent dans notre étude.

Voici le dataframe ainsi obtenu :

	0	1	2	3	4	\
2019-01-01-Tuesday	0.187455	0.187455	0.187455	0.187455	3.025112	
2019-01-02-Wednesday	0.187455	0.187455	0.187455	0.187455	0.239105	
2019-01-03-Thursday	0.187455	0.187455	0.187455	0.187455	0.262061	
2019-01-04-Friday	0.187455	0.187455	0.187455	0.187455	0.287885	
2019-01-05-Saturday	2.06201	0.187455	0.187455	0.187455	0.310841	

	5	6	7	8	9	...	\
2019-01-01-Tuesday	3.171453	0.359621	0.21328	0.187455	0.187455	...	
2019-01-02-Wednesday	0.359621	0.359621	0.187455	0.0	0.0	...	
2019-01-03-Thursday	0.359621	0.336666	0.187455	0.187455	0.187455	...	
2019-01-04-Friday	0.359621	0.310841	0.187455	0.421775	0.187455	...	
2019-01-05-Saturday	0.359621	0.285016	0.187455	0.187455	0.187455	...	

	14	15	16	17	18	\
2019-01-01-Tuesday	2.811832	0.937277	2.849135	0.172166	2.983998	
2019-01-02-Wednesday	0.0	0.0	0.063127	0.172166	0.160688	
2019-01-03-Thursday	0.0	0.0	0.086083	0.172166	0.137733	
2019-01-04-Friday	0.0	0.0	0.117647	0.176852	0.117647	
2019-01-05-Saturday	0.004686	0.008608	0.172166	0.172166	0.040172	

	19	20	21	22	23
2019-01-01-Tuesday	2.826179	2.811832	2.811832	2.874318	2.999288
2019-01-02-Wednesday	0.0	0.0	0.004686	0.0	0.12497
2019-01-03-Thursday	0.014059	0.0	2.811832	2.343193	0.0
2019-01-04-Friday	0.0	0.0	2.811832	2.811832	2.811832
2019-01-05-Saturday	0.032805	2.343193	2.811832	13.346831	4.295854

[5 rows x 24 columns]

Voici le fichier principal jusqu'ici :

```
building_id = getUserBuilding()
consommation_type = get_consommation_type_wanted()

data_consommation_hour_for_building = get_Df_for_building(
    building_id, consommation_type
)

data_consommation_hour_for_building_with_hours_changes_managed = manage_hour_changes(
    data_consommation_hour_for_building
)

df_profil_day = make_day_profil
(data_consommation_hour_for_building_with_hours_changes_managed)

show_graphic_for_building_day_profil(df_profil_day, buildind_id)
```

La function `show_graphic_for_building_day_profil(df_profil_day, buildind_id)` fais l'affichage des profils journaliers pour les 365 jours de l'année :

```
def show_graphic_for_building_day_profil(df, buildind_id):
    # def plot_daily_electricity_profile(df):
    # Extraire les jours de la semaine
    days = df.index

    # Extraire les valeurs de consommation d'électricité pour chaque heure
    values = df.values

    # Créer un graphique
    plt.figure(figsize=(10, 6))
    for i in range(len(days)):
        plt.plot(range(24), values[i], label=days[i])

    # Ajouter des titres et des légendes
    plt.title("Profil journalier de consommation d'électricité")
    plt.xlabel("Heure")
    plt.ylabel("Consommation d'électricité pour la construction {}".format(building_id))
    plt.xticks(range(24))
    plt.show()
```

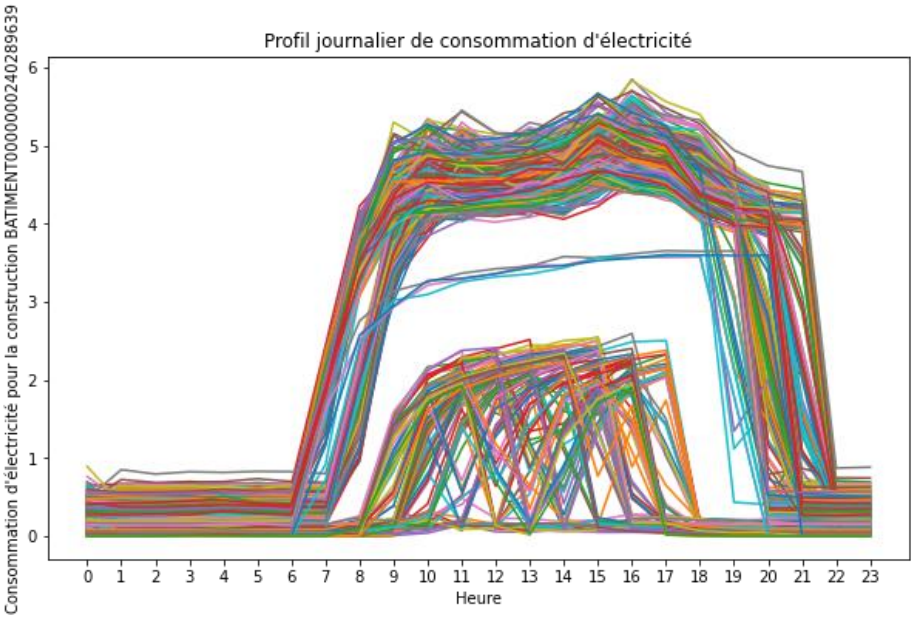


Figure 3, Profil journalier, bâtiment du secteur tertiaire (bureau id BATIMENT0000000240289639)

Les fluctuations constatées dans les données de consommation d'énergie d'un bâtiment du secteur peuvent être expliquées par divers facteurs, dont les **différences d'occupation de l'espace** au fil du temps.

Les variations d'utilisation des locaux, telles que des périodes de pic d'activité ou des périodes de faible utilisation, peuvent influencer la consommation d'énergie.

De plus, l'enrichissement stochastique des données, effectué à l'aide de réseaux bayésiens, pourrait également contribuer à ces variations.

Les réseaux bayésiens, en modélisant les relations probabilistes entre différentes variables, peuvent introduire une composante aléatoire dans les données, ce qui peut se refléter dans les fluctuations observées de la consommation énergétique du bâtiment résidentiel.

Une autre raison des résultats divergents peut s'expliquer par le fait que les données sont générées à partir de scénarios d'utilisation des divers appareils présents dans le bâtiment de bureaux. Pour les bureaux**, les résultats ELECSPE font appel aux scénarios d'utilisation des appareils via QIRIEL, un enrichisseur du CSTB**. Cela explique probablement les variations observées, notamment les baisses abruptes les samedis/dimanches. Je tiens à souligner que les profils ELECSPE examinés dans ce travail sont élaborés à partir de scénarios d'utilisation des différents appareils présents dans le bâtiment de bureaux, avec des journées types (week-ends, jours ouvrés classiques, jours de vacances...) et des semaines types constituées à partir de ces jours types.

Ainsi, ces trois aspects, **les différences d'occupation, l'utilisation des scenarios d'utilisation via QIRIEL** et **l'enrichissement stochastique par des réseaux bayésiens**, peuvent jouer un rôle crucial dans la compréhension des variations de la consommation d'énergie au fil du temps.

6. Etude Bâtiment 1, Secteur bureaux, consommation d’électricité, étude sur le clustering de la consommation électrique d'un bâtiment du secteur ternaire, bureaux

ID du bâtiment étudié : `BATIMENT0000000240777665`

Clustériser les profils journaliers de la consommation d'électricité peut être motivé par divers objectifs et apporter plusieurs avantages :

1. **Identification de Profils de Consommation** : Le clustering permet de découvrir des groupes de profils de consommation similaires, révélant ainsi des tendances et des comportements spécifiques dans la consommation d'électricité.

2. **Segmentation du Marché** : Les clusters peuvent être utilisés pour segmenter les utilisateurs en fonction de leurs profils de consommation, ce qui est précieux pour les entreprises énergétiques dans la personnalisation des offres et services.

3. **Optimisation de la Production et de la Distribution** : Comprendre les différents profils de consommation permet une meilleure planification de la production et de la distribution d'électricité, adaptée aux besoins spécifiques de chaque groupe.

4. **Gestion de la Demande** : L'identification de profils de consommation peut faciliter la mise en œuvre de stratégies de gestion de la demande, telles que la tarification dynamique, pour inciter à une consommation plus efficiente.

5. **Détection d'Anomalies** : Les clusters peuvent mettre en évidence des profils de consommation inhabituels, permettant la détection rapide d'anomalies ou de problèmes potentiels dans le réseau électrique.

6. **Planification de l'Infrastructure** : La connaissance des différents profils de consommation peut aider à planifier les investissements dans l'infrastructure électrique pour répondre aux besoins spécifiques de chaque segment.

7. **Réduction des Coûts** : En comprenant les variations de consommation, les entreprises peuvent mettre en place des initiatives visant à réduire les coûts, notamment en optimisant les périodes de pic de demande.

8. **Gestion de l'Énergie dans les Bâtiments** : Pour les installations individuelles, la clusterisation des profils de consommation peut aider à optimiser la gestion de l'énergie, en ajustant les systèmes en fonction des tendances spécifiques de chaque cluster.

9. **Développement de Politiques Énergétiques** : Les clusters peuvent fournir des informations pour le développement de politiques énergétiques, en mettant en lumière les caractéristiques dominantes des différents segments de consommateurs.

10. **Analyse Temporelle** : La clusterisation permet une analyse temporelle approfondie, révélant les variations horaires spécifiques dans les habitudes de consommation d'électricité.

En somme, le clustering des profils journaliers de la consommation d'électricité offre une approche puissante pour comprendre les schémas de consommation, améliorer la planification énergétique et mettre en place des stratégies ciblées pour une utilisation plus efficace de l'électricité.

Contexte

Dans le cadre de la transition énergétique des villes, Efficacity, un institut de recherche et développement dédié à la transition énergétique urbaine, développe plusieurs logiciels de simulation. Ces outils sont conçus pour aider à la prise de décision dans les projets d'aménagement urbain en prenant en compte la valorisation des énergies renouvelables et de récupération, ainsi que le choix des systèmes énergétiques en fonction des besoins spécifiques des bâtiments.

Un aspect crucial de ces logiciels est la détermination des besoins énergétiques des bâtiments. À cette fin, une base de données de profils de besoins énergétiques a été créée. Cette base de données est essentielle pour les simulations à l'échelle d'un quartier. Initialement, un échantillon représentatif des bâtiments français a été créé en classifiant les bâtiments par typologie. Les bâtiments sélectionnés ont ensuite été simulés à l'aide du logiciel DIMOSIM, spécialisé dans la simulation énergétique urbaine, afin de calculer leurs profils de besoins énergétiques. Cette base de données est constamment enrichie et mise à jour.

Pour cette étude, nous allons etudier la consommation d'éléctricité pour un des 396 bâtiments du secteur tertiare (bureau).

Récupération de la data et conversion la data dans un format que nous pouvons manipuler facilement (sans changer la data elle-même)

```
from helpers.get_data_consumation import get_Df_for_building

building_id = "BATIMENT0000000240777665"

consommation_type = "bureau_electricity"

data_consommation_hour_for_building = get_Df_for_building(
    building_id, consommation_type
)
data_consommation_hour_for_building
```

```
                     Datetime  electricity
0     2019-01-01 00:00:00+01:00     0.051205
1     2019-01-01 01:00:00+01:00     0.047232
2     2019-01-01 02:00:00+01:00     0.047482
3     2019-01-01 03:00:00+01:00     0.047517
4     2019-01-01 04:00:00+01:00     0.050697
...                         ...          ...
8755  2019-12-31 19:00:00+01:00     3.440564
8756  2019-12-31 20:00:00+01:00     3.389798
8757  2019-12-31 21:00:00+01:00     0.513952
8758  2019-12-31 22:00:00+01:00     0.504356
8759  2019-12-31 23:00:00+01:00     0.505625

[8760 rows x 2 columns]
```

Vérification de la taille et le type de la data

```
data_consommation_hour_for_building.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8760 entries, 0 to 8759
Data columns (total 2 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Datetime     8760 non-null   object 
 1   electricity  8760 non-null   float64
dtypes: float64(1), object(1)
memory usage: 137.0+ KB
```

```
data_consommation_hour_for_building.describe()
```

```
       electricity
count  8760.000000
mean      1.821979
std       1.681863
min       0.007467
25%       0.327235
50%       1.465605
75%       3.677617
max       4.835091
```

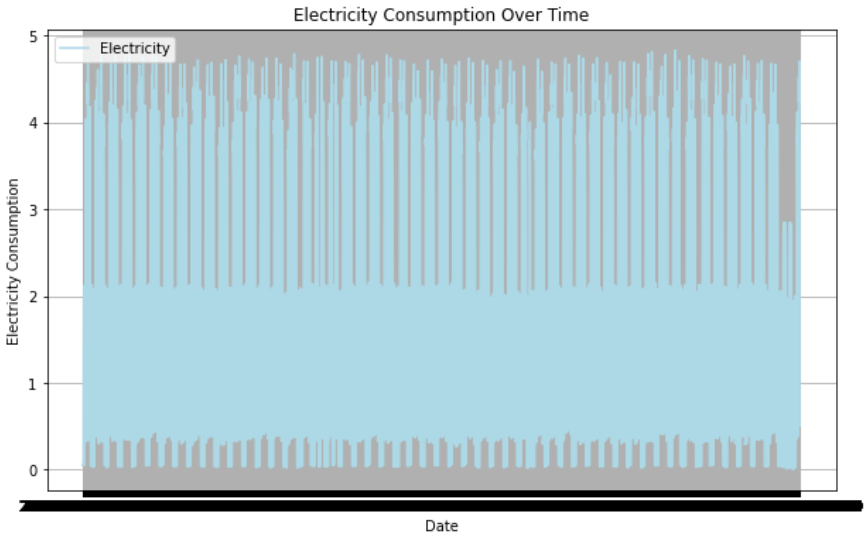
Conversion de la collonne "Datetime" en format Timestamp et setter comme index cette colonne

```
data = data_consommation_hour_for_building.copy()

data["Datetime"] = data["Datetime"].apply(lambda x: x.strftime("%Y-%m-%d %H:%M"))

# Set the "Datetime" column as the index of the DataFrame
df_to_show = data.set_index("Datetime")

df_to_show
```

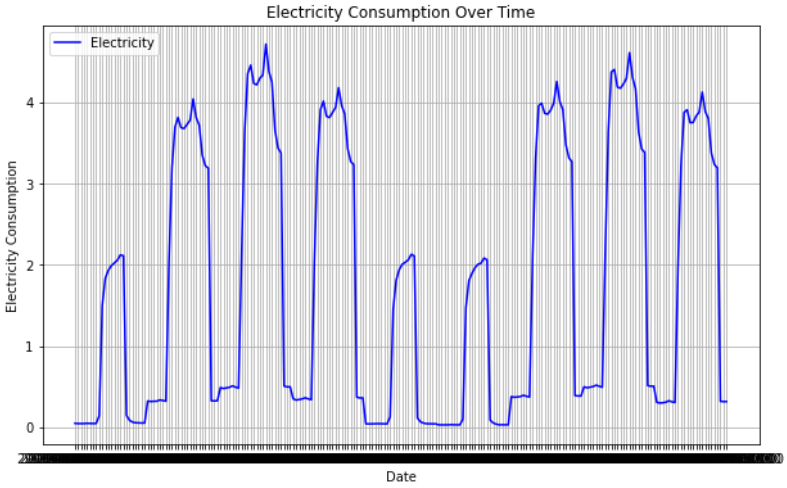
```
                  electricity
Datetime                     
2019-01-01 00:00     0.051205
2019-01-01 01:00     0.047232
2019-01-01 02:00     0.047482
2019-01-01 03:00     0.047517
2019-01-01 04:00     0.050697
...                       ...
2019-12-31 19:00     3.440564
2019-12-31 20:00     3.389798
2019-12-31 21:00     0.513952
2019-12-31 22:00     0.504356
2019-12-31 23:00     0.505625

[8760 rows x 1 columns]
```

Visualisation de la data

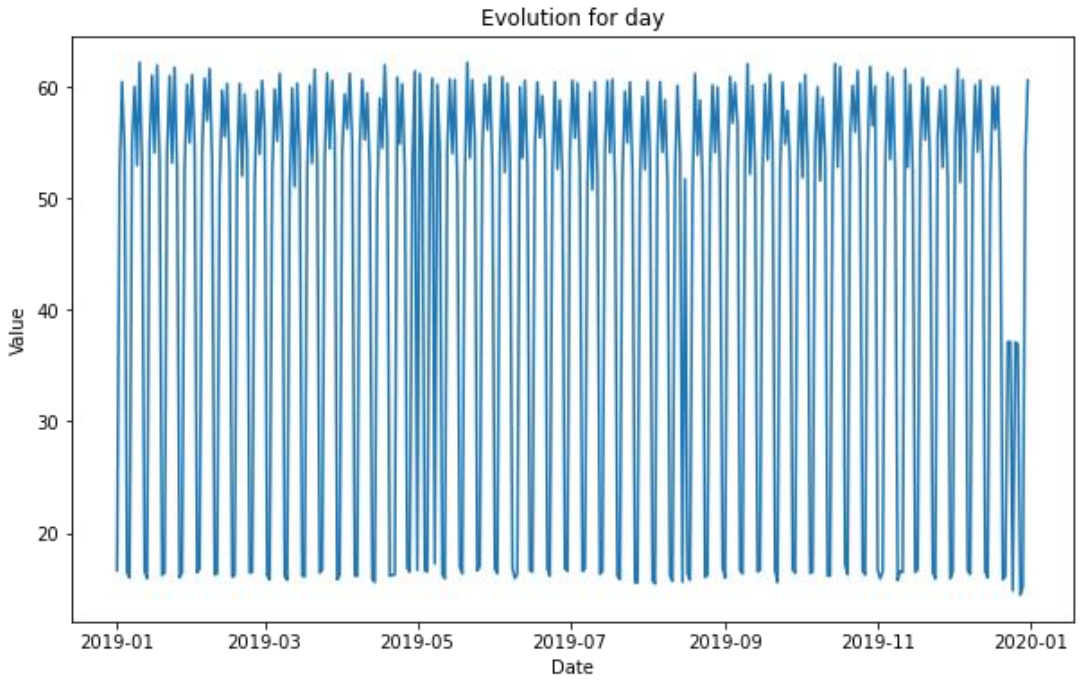
Visualisation de la consommation d'éléctricité sur toute l'année (chaque heure sur une année)



Visualisation de la consommation d'électricité sur une période de l'année (chaque heure sur une période spécifié)



Visualisation de la consommation d'électricité sur toute l'année (chaque jour sur une année)



Parcourir la data pour récupérer les nulls or les NaN

```
nan_count = data_consommation_hour_for_building.isna().sum().sum()
print("Nombre total de NaN dans le DataFrame : {}".format(nan_count))

null_count = data_consommation_hour_for_building.isnull().sum().sum()
print("Nombre total de null dans le DataFrame : {}".format(null_count))
```

```
Nombre total de NaN dans le DataFrame : 0
Nombre total de null dans le DataFrame : 0
```

Récupérer la data de caractérisation pour le bâtiment étudie

```
def get_data_for_building(id):
    buildings = get_excel_file_data("BU_all_buildings_IDF.xlsx")

    # Soustraire la ligne avec l'ID spécifié
    row_to_subtract = buildings.index[buildings["ID"] == id]
    print("row_to_subtract", row_to_subtract)

    # Soustraire la ligne de 'row_to_subtract' du DataFrame original 'df'
    df_subtracted = buildings.loc[row_to_subtract]

    # Afficher le DataFrame après la suppression
    return df_subtracted


building_data = get_data_for_building(building_id)

print(building_data.head())
```

```
row_to_subtract Int64Index([42], dtype='int64')
                        ID  bou  caf  chauf   ecr  mfd  pcf2e  port1e  \
42  BATIMENT0000000240777665  433  480      0  1429  294    980     917

    nb_equip  nb_equip_bureau  ...  Pmax_cl_par_m2  annuel_el_par_m2  \
```

```
42      4533             3620  ...       55.381831          15.96054

    Pmax_el_par_m2  annee_construction  effectif  nb_equip_par_m2  \
42        4.835091              2010.0    1784.0         0.103478

    nb_equip_bureau_par_m2 nb_equip_autre_par_m2  bd_surface_par_occ  \
42                0.082637              0.020842           24.555078

    hbs_surface_par_occ
42            18.891256

[1 rows x 42 columns]
```

La description du bâtiment avec l'ID "BATIMENT0000000240777665" est la suivante :

Identifiant unique (ID) : BATIMENT0000000240777665

Usage des équipements par catégorie :

Bouilloires (bou) : 433

Machines à café (caf) : 480

Chauffages d'appoint (chauf) : 0

Écrans (ecr) : 1429

Imprimantes multifonctionnelles (mfd) : 294

PC fixes avec 2 écrans fixes (pcf2e) : 980

PC portables avec 1 écran fixe (port1e) : 917

Nombre total d'équipements :

Total : 4533

Nombre d'équipements par catégorie :

Bureautiques : 3620

Non-bureautiques : 913

Caractéristiques du bâtiment :

Hauteur : 18.8

Nombre d'étages : 5

Caractéristiques énergétiques :

Valeur U des murs extérieurs : 0.5447418746709977

Valeur U du toit extérieur : 0.37263793862262495

Valeur U du sol extérieur : 0.3860091116262472

Proportion de fenêtres par rapport aux murs extérieurs : 0.7306956855379408

Type de fenêtres sur les murs extérieurs : double_glazing

Point de consigne de chauffage confortable : 20.31304233306252

Point de consigne de refroidissement confortable : 24.68006957347139

Informations sur la géométrie :

Forme géométrique du bâtiment : POLYGON Z (... géométrie détaillée ...)

Superficie totale du bâtiment : 10252.92

Périmètre de la géométrie d'emprise : 604.53

Surface habitable totale : 43806.26

Caractéristiques additionnelles :

Hauteur moyenne sous plafond : 3.61

Année de construction : 2010

Effectif : 1784

Densité d'équipements par mètre carré :

Nombre total d'équipements par mètre carré : 0.1035

Nombre d'équipements de bureau par mètre carré : 0.0826

Nombre d'équipements dans d'autres parties du bâtiment par mètre carré : 0.0208

Surfaces par occupant :

Surface de bureau par occupant : 24.56

Surface de bureaux imposables par occupant : 18.89

Profils journaliers

Gérer le changement d'heure (fin mars et fin octobre)

Fin mars : un jour avec une valeur manquante

Deux approches sont envisageables :

- Reconstruire les données manquantes.
- Supprimer le profil contenant des données manquantes.

Étant donné la faible variation horaire dans notre jeu de données et le fait que nous ne reconstruirons qu'un seul point sur l'ensemble de l'année, nous pouvons considérer que toute incertitude introduite par la reconstruction est totalement négligeable. Une autre raison de reconstruire le profil manquant est le risque de supprimer un profil essentiel, notamment un profil spécifique notablement différent des autres, ce qui pourrait affecter les résultats de la classification.

Pour la reconstruction, nous allons utiliser la méthode consistant à prendre la moyenne des valeurs précédant et suivant le point manquant.

Nous allons également vérifier plusieurs cas pour évaluer si la valeur de cette heure pour les jours précédents de la même semaine diffère considérablement. Nous examinerons également un échantillon d'autres profils similaires pour garantir qu'il n'y a pas de variations significatives des valeurs au point de données que nous souhaitons reconstruire (à la même heure sur d'autres jours).

Cela nous permettra de nous assurer que les données reconstruites seront relativement fiables et que toute erreur introduite dans l'ensemble de données sera négligeable.

Fin octobre : un jour avec une heure supplémentaire

Nous avons le choix entre deux approches :

- Supprimer une heure et conserver 24 heures, ce qui équivaut à une journée complète.
- Éliminer le profil contenant des données manquantes

Nous avons opté pour la première méthode en supprimant une des deux valeurs correspondant à la même date et à la même heure. Cette décision a été motivée par la nécessité d'éviter tout risque de suppression d'un profil essentiel, notamment d'un profil spécifique distinct des autres, susceptible d'influer sur les résultats de la classification. Étant donné que notre ensemble de données ne présente qu'un seul cas de ce type, la suppression d'une seule valeur, bien que pouvant introduire une certaine incertitude, reste négligeable au regard de l'ensemble des données.

```
from helpers.manage_hour_changes import manage_hour_changes

data_consommation_hour_for_building_with_hours_changes_managed = manage_hour_changes(
    data_consommation_hour_for_building
)
print(data_consommation_hour_for_building_with_hours_changes_managed)
```

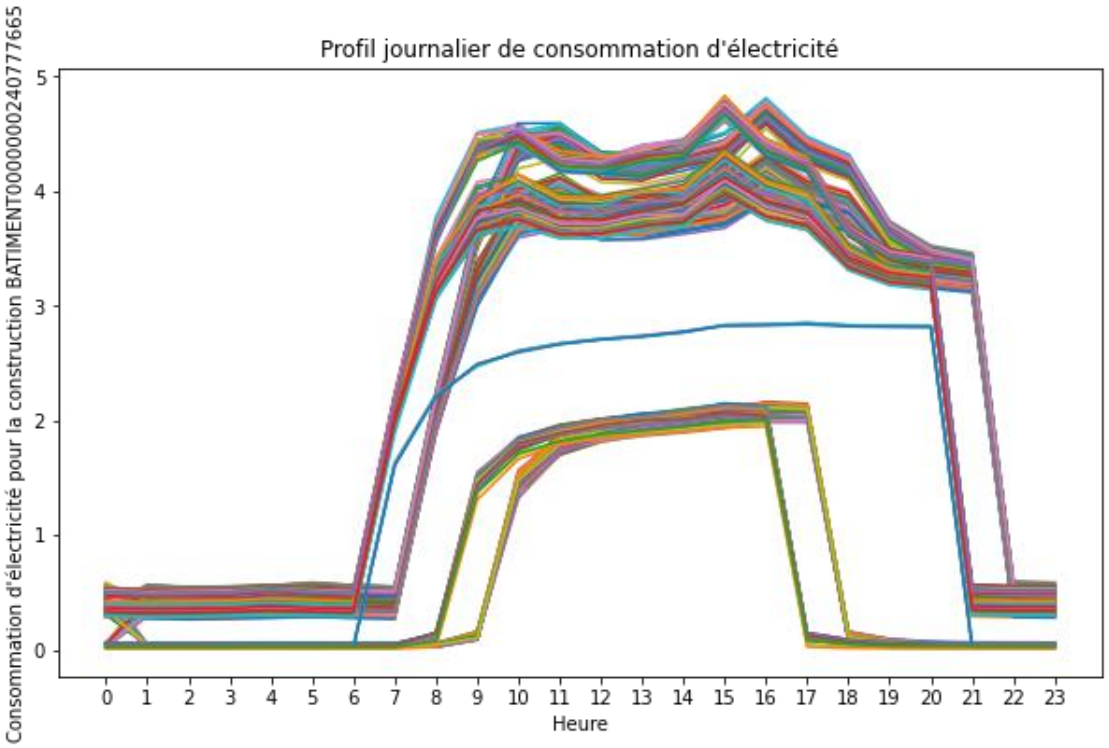
```
Indices of hour changes:
Int64Index([2138], dtype='int64')
                Datetime  electricity
2133 2019-03-30 21:00:00     0.030327
2134 2019-03-30 22:00:00     0.029766
2135 2019-03-30 23:00:00     0.030067
2136 2019-03-31 00:00:00     0.033938
2137 2019-03-31 01:00:00     0.034140
2138 2019-03-31 02:00:00     0.034286
2139 2019-03-31 03:00:00     0.034052
2140 2019-03-31 04:00:00     0.034431
2141 2019-03-31 05:00:00     0.035847
2142 2019-03-31 06:00:00     0.034877
Les lignes qui sont duppliquées :
                Datetime  electricity
7178 2019-10-27 02:00:00     0.034399
7179 2019-10-27 02:00:00     0.035444
                Datetime  electricity
0    2019-01-01 00:00:00     0.051205
1    2019-01-01 01:00:00     0.047232
2    2019-01-01 02:00:00     0.047482
3    2019-01-01 03:00:00     0.047517
4    2019-01-01 04:00:00     0.050697
...                  ...          ...
8756 2019-12-31 19:00:00     3.440564
8757 2019-12-31 20:00:00     3.389798
8758 2019-12-31 21:00:00     0.513952
8759 2019-12-31 22:00:00     0.504356
8760 2019-12-31 23:00:00     0.505625

[8760 rows x 2 columns]
```

La fonction suivante (make_day_profil) retourne un nouveau DataFrame qui représente les **profils de consommation d'électricité pour chaque jour de l'année, organisés par heures**.

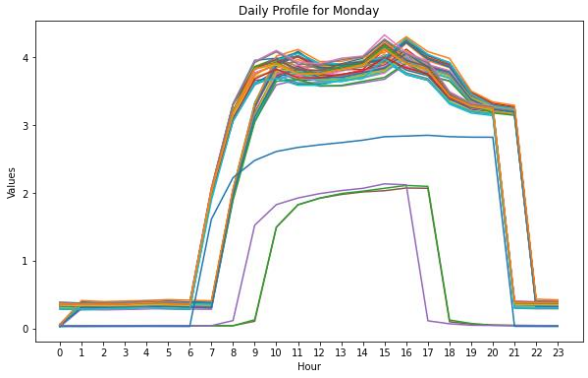
```
from helpers.make_day_profil import make_day_profil

df_profil_day = make_day_profil(
    data_consommation_hour_for_building_with_hours_changes_managed
)
print(df_profil_day)

df_profil_day_original=df_profil_day.copy()
```

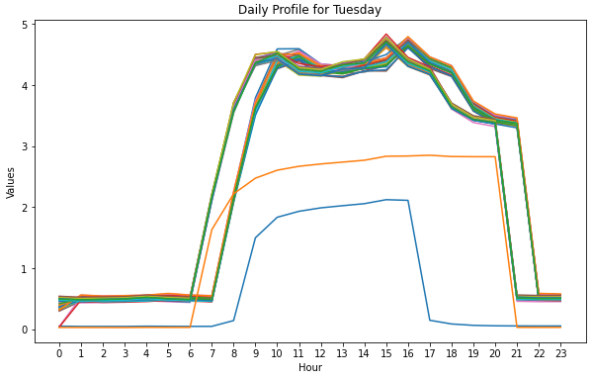
```
                           0         1         2         3         4  … 23  \
2019-01-01-Tuesday    0.051205  0.047232  0.047482  0.047517  0.050697
2019-01-02-Wednesday  0.324805  0.319113  0.321784  0.322277  0.338286
2019-01-03-Thursday    0.49037  0.478599  0.485637  0.494218  0.511541
2019-01-04-Friday     0.352068  0.337039  0.342835  0.351214   0.36492
2019-01-05-Saturday   0.044422  0.043594  0.044735  0.045023  0.046484
...                        ...       ...       ...       ...       ...

[365 rows x 24 columns]
```

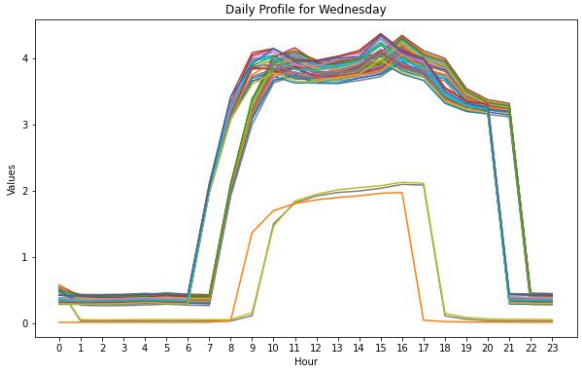
Représentation graphique des profils journaliers pour le bureau étudié

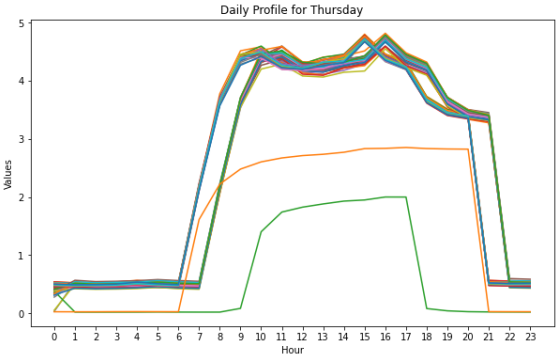


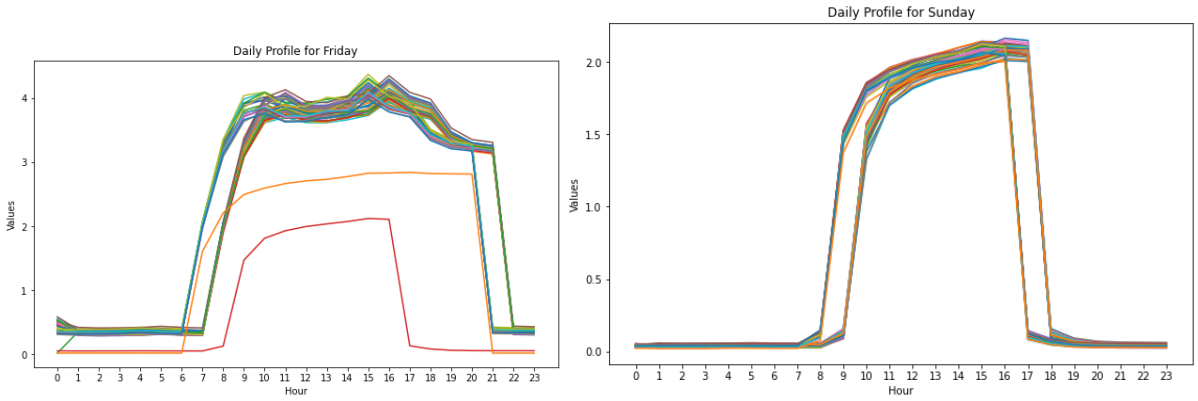
Représentation graphique des profils journaliers pour le bureau étudié pour chaque jour de la semaine sur une année











Après l'étude des performances du clustering avec des différents algorithmes et des différents paramètres pour chaque algorithme, nous déduisons que les algorithmes TimeseriesKmeans ont des meilleures performances que DbScan or MeanShift, par exemple, et s'est pour ça que, par la suite nous allons voir un peu plus en détail cet algorithme.

tslearn et TimeseriesKMeans

"tslearn" est une bibliothèque Python dédiée à l'apprentissage automatique sur des séries temporelles (time series). Elle offre des outils et des modèles spécifiquement conçus pour traiter les données séquentielles, ce qui est particulièrement utile dans des domaines tels que la reconnaissance de formes temporelles, la classification de séries temporelles, la prédiction de séries temporelles, etc.

Quelques caractéristiques clés de tslearn incluent :

- Compatibilité avec scikit-learn : tslearn est intégrée avec l'écosystème scikit-learn, ce qui signifie que vous pouvez l'utiliser de manière transparente avec d'autres bibliothèques populaires d'apprentissage automatique en Python.

- Modèles spécifiques pour les séries temporelles : Elle propose des modèles spécifiques aux séries temporelles tels que les réseaux de neurones récurrents (RNN), les k-plus proches voisins pour séries temporelles, les forêts aléatoires pour séries temporelles, etc.

- La méthode TimeSeriesKMeans de tslearn est spécifiquement conçue pour le clustering de séries temporelles. Voici quelques avantages de l'utiliser pour le clustering de données temporelles

- Considération de la nature séquentielle : Contrairement à des méthodes de clustering traditionnelles, TimeSeriesKMeans prend en compte la structure séquentielle des données temporelles. Elle mesure les similarités entre les séquences temporelles en prenant en considération l'ordre et les motifs temporels.

- Invariance par rapport à la translation et la dilatation temporelle : La méthode est conçue pour être invariante par rapport aux translations temporelles et aux dilatations, ce qui signifie qu'elle peut identifier des motifs similaires même si les séquences temporelles ont des décalages ou des changements d'échelle.

- Utilisation de métriques spécifiques aux séries temporelles : TimeSeriesKMeans utilise des métriques de similarité spécifiques aux séries temporelles pour mesurer la distance entre les séquences. Cela permet une meilleure adéquation aux caractéristiques temporelles des données.

- Intégration avec tslearn : Comme elle est intégrée avec tslearn, elle peut être utilisée de manière cohérente avec d'autres fonctionnalités de la bibliothèque, facilitant ainsi le prétraitement, l'évaluation et la visualisation des résultats.

- Optimisation pour les données temporelles : TimeSeriesKMeans est conçue pour tirer parti des spécificités des données temporelles, ce qui peut conduire à des résultats plus pertinents pour ce type de données par rapport à des méthodes de clustering génériques.

- Gestion des problèmes de dimension : Elle offre des méthodes pour gérer efficacement la dimension temporelle, qui peut varier d'une série à l'autre, en utilisant des méthodes de resampling.

- Flexibilité dans le choix des métriques et des mécanismes de clustering : La méthode permet de spécifier différentes métriques de distance et de choisir entre différents mécanismes de clustering, ce qui offre une certaine flexibilité pour adapter l'algorithme aux caractéristiques spécifiques de vos données.

Après avoir analysé la méthode du coude, il est déterminé que l**e nombre optimal de clusters est 3.**

Par la suite, nous appliquerons l'algorithme TimeSeriesKMeans en utilisant trois métriques différentes.

- metric="euclidean"
- metric="dtw"
- metric="softdtw"

Ces trois métriques "euclidean", "dtw" (Dynamic Time Warping), et "softdtw" (Soft Dynamic Time Warping), sont utilisées dans le contexte de la mesure de la similarité ou de la distance entre des séries temporelles.

Explications :

Euclidean (euclidienne) :

Description : C'est la métrique euclidienne classique qui mesure la distance directe entre deux points dans un espace euclidien. Application : Elle est souvent utilisée lorsque la forme globale des courbes est importante et que les séries temporelles ont des valeurs comparables.

DTW (Dynamic Time Warping) :

Description : Le DTW est une méthode qui mesure la dissimilarité entre deux séries temporelles en trouvant la meilleure correspondance entre leurs points, même si elles ont des échelles temporelles différentes ou des déformations dans le temps.

Application : Utile lorsque la synchronisation temporelle n'est pas constante entre les séries temporelles, ce qui est fréquent dans les domaines tels que la reconnaissance de la parole ou l'analyse de séquences temporelles.

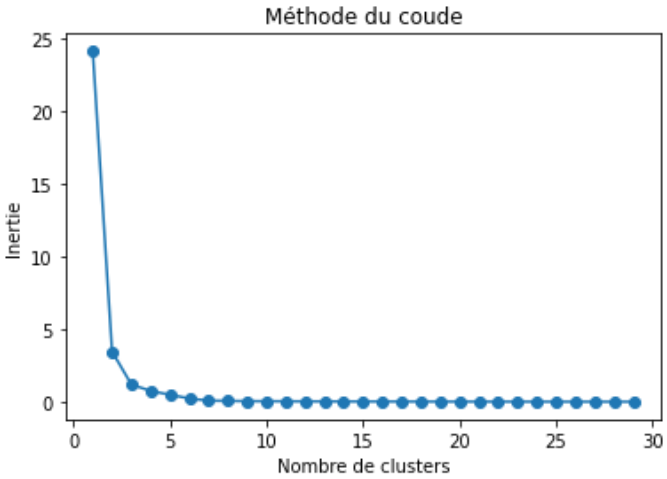
SoftDTW (Soft Dynamic Time Warping) :

Description : Une version "adoucie" du DTW qui introduit une pénalité continue pour les déformations temporelles. Cela rend la métrique différentiable, ce qui la rend plus adaptée à des applications où la différenciation est importante.

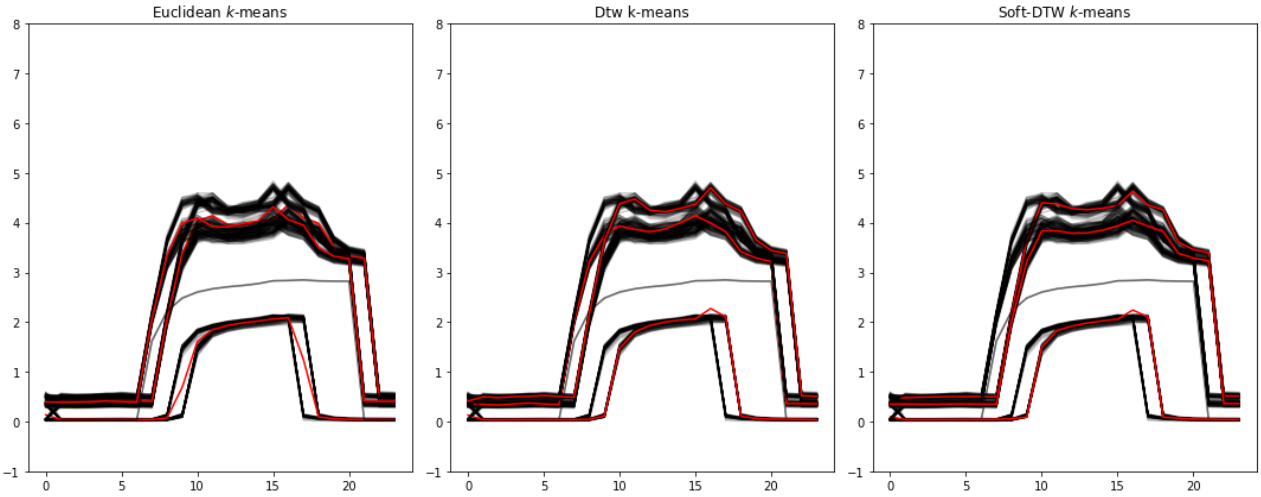
Application : Souvent utilisée dans des contextes d'apprentissage automatique, en particulier pour les tâches nécessitant des gradients tels que l'optimisation de modèles.

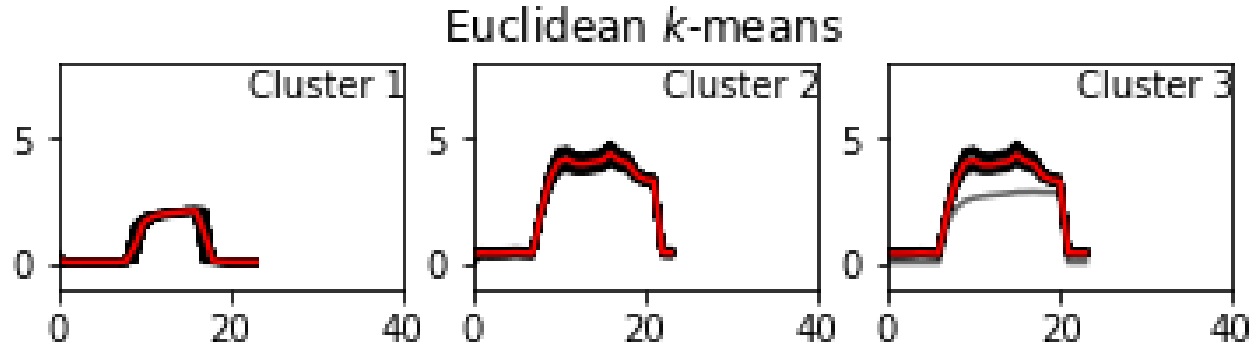
Calcul du nombre des clusters

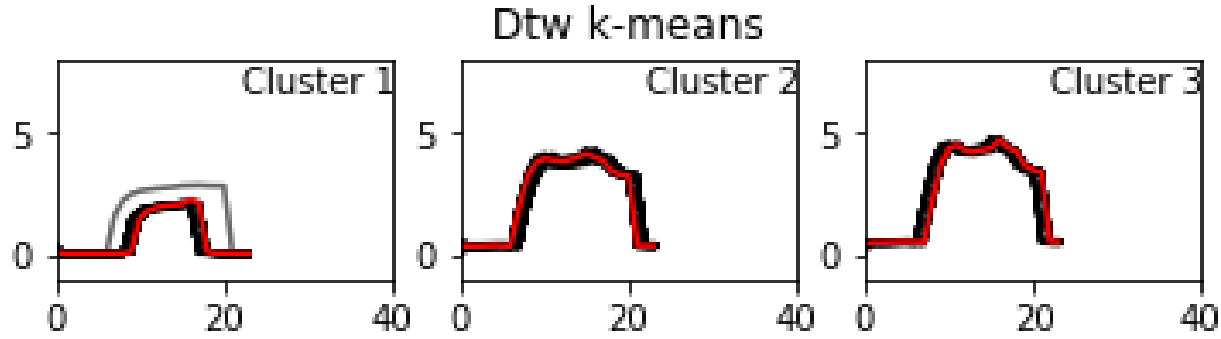
Pour le calcul du nombre des clusters nous allons utiliser La méthode du coude (elbow method), qui appliqué pour l'algorithme TimeseriesKmeans donné comme résultat :

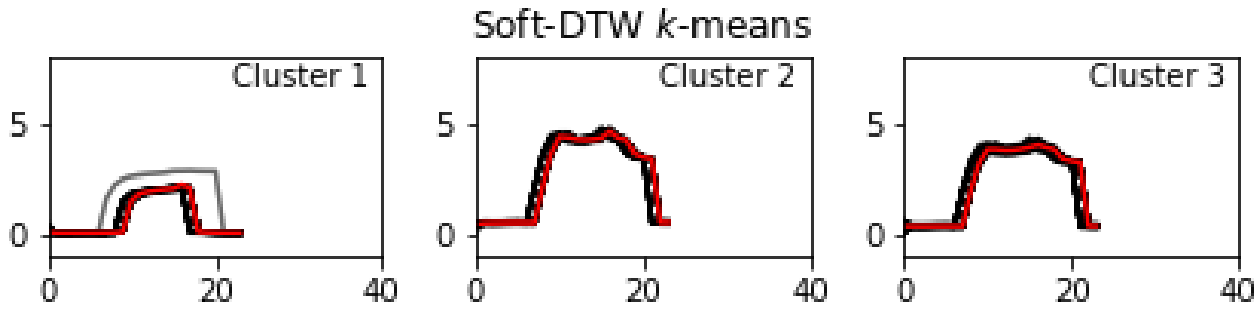


Affichage des resultats de clustering









L'inertia et le score de la silhouette

L'inertie (inertia en anglais) est une mesure utilisée pour évaluer la performance d'un algorithme de clustering, en particulier dans le contexte des algorithmes de type K-means. Elle mesure la somme des carrés des distances entre chaque point de données et le centroïde du cluster auquel il est assigné. En d'autres termes, l'inertie mesure la cohésion des points à l'intérieur d'un cluster.

L'objectif du clustering est de minimiser cette inertie. Plus l'inertie est faible, plus les points à l'intérieur de chaque cluster sont proches les uns des autres, indiquant une meilleure cohésion.

Score de la Silhouette

Cette méthode permet d'évaluer la qualité des clusters créés grâce aux algorithmes de clustering. Compris entre [-1,1], le score silhouette est parfois utilisé pour trouver la valeur optimale du nombre de clusters « k ». Pour ce faire, on considère la valeur de « k » ayant le score de silhouette le plus proche de 1.

```
from tslearn.clustering import silhouette_score


silhouette_scores = {}

silhouette_scores["euclidean"] = silhouette_score(
    df_profil_day, model.labels_, metric="euclidean"
)
silhouette_scores["dtw"] = silhouette_score(
    df_profil_day, model_dtw.labels_, metric="dtw"
)
silhouette_scores["sdtw"] = silhouette_score(
    df_profil_day, model_sdtw.labels_, metric="softdtw"
)

best_metric = max(silhouette_scores, key=silhouette_scores.get)

print(f"The metric with the highest silhouette score is: {best_metric}")
print(f"Silhouette Score: {silhouette_scores[best_metric]}")
print(
    "Silhouette Coefficient Euclidean: (between -1 and 1, better bigger) ",
    silhouette_scores["euclidean"],
)
print(
    "Silhouette Coefficient dtw: (between -1 and 1, better bigger) ",
    silhouette_scores["dtw"],
)

print(
    "Silhouette Coefficient sdtw : (between -1 and 1, better bigger) ",
    silhouette_scores["sdtw"],
```

```
)
inertia_scores = {}

inertia_scores["euclidean"] = model.inertia_
inertia_scores["dtw"] = model_dtw.inertia_
inertia_scores["sdtw"] = model_sdtw.inertia_

best_metric_inertia = min(inertia_scores, key=inertia_scores.get)
print( "Inertia euclidean metric : ", inertia_scores["euclidean"])
print( "Inertia dtw metric : ", inertia_scores["dtw"])
print( "Inertia sdtw metric : ", inertia_scores["sdtw"])
print(f"Inertie preferable : {best_metric_inertia}")
```

Création de DataFrames contenant les données originels et les clusters correspondants aux métriques avec la plus petite inertie, "dtw", dans ce cas précis

```
df_profil_day_cluster = df_profil_day_original.copy()


if best_metric_inertia == "euclidean":
    df_profil_day_cluster.insert(0, "clusters", y_pred)
    model_optimal = model
    y_predict = y_pred
elif best_metric_inertia == "dtw":
    df_profil_day_cluster.insert(0, "clusters", y_pred_dtw)
    model_optimal = model_dtw
    y_predict = y_pred_dtw
elif best_metric_inertia == "sdtw":
    df_profil_day_cluster.insert(0, "clusters", y_pred_sdtw)
    model_optimal = model_sdtw
    y_predict = y_pred_sdtw

df_profil_day_cluster
```

	clusters	0	1	2	3	4	\
2019-01-01	0	0.051205	0.047232	0.047482	0.047517	0.050697	
2019-01-02	2	0.324805	0.319113	0.321784	0.322277	0.338286	
2019-01-03	2	0.49037	0.478599	0.485637	0.494218	0.511541	
2019-01-04	2	0.352068	0.337039	0.342835	0.351214	0.36492	
2019-01-05	0	0.044422	0.043594	0.044735	0.045023	0.046484	
...	...	...	...	...	...	...	
2019-12-27	2	0.022067	0.022174	0.022222	0.022779	0.022894	
2019-12-28	0	0.007929	0.007467	0.00773	0.007926	0.008416	
2019-12-29	0	0.023599	0.021457	0.021423	0.021216	0.023871	
2019-12-30	2	0.364904	0.353678	0.356368	0.360222	0.380445	
2019-12-31	2	0.495741	0.484452	0.490274	0.497408	0.523991	

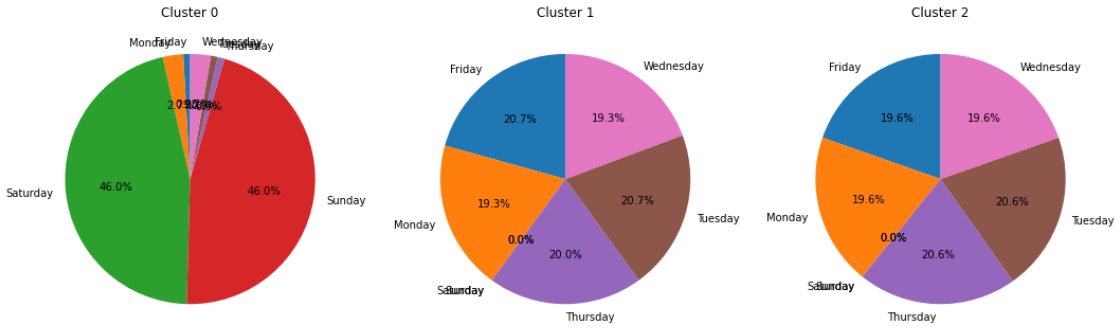
	5	6	7	8	...	14	15	\
2019-01-01	0.048953	0.047959	0.048649	0.143458	...	2.057848	2.123015	
2019-01-02	0.331128	0.322116	2.010638	3.190806	...	3.780604	4.041302	

[365 rows x 25 columns]

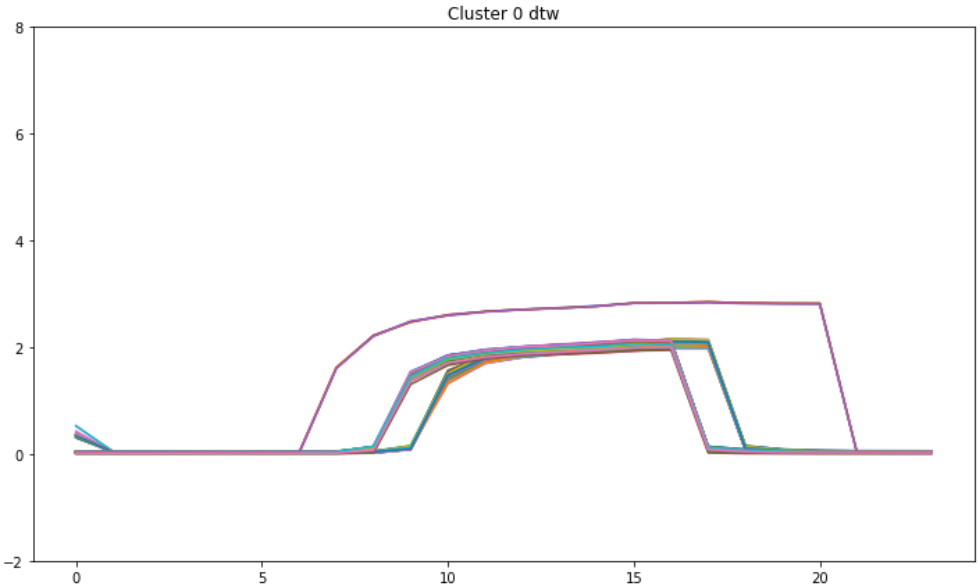
Distribution des profiles journalières par cluster

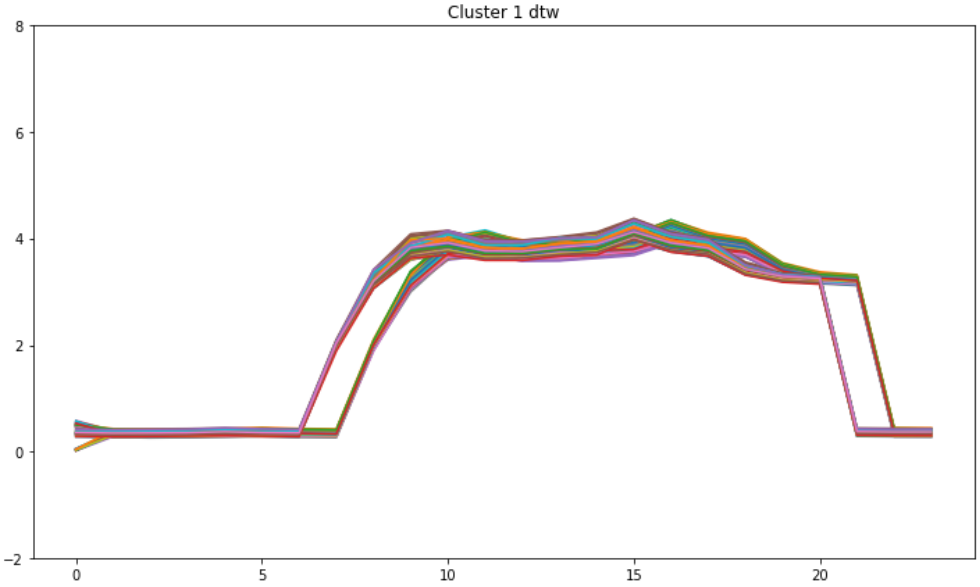
Distribution des jours appartenant au même cluster selon le jour de la semaine dtw

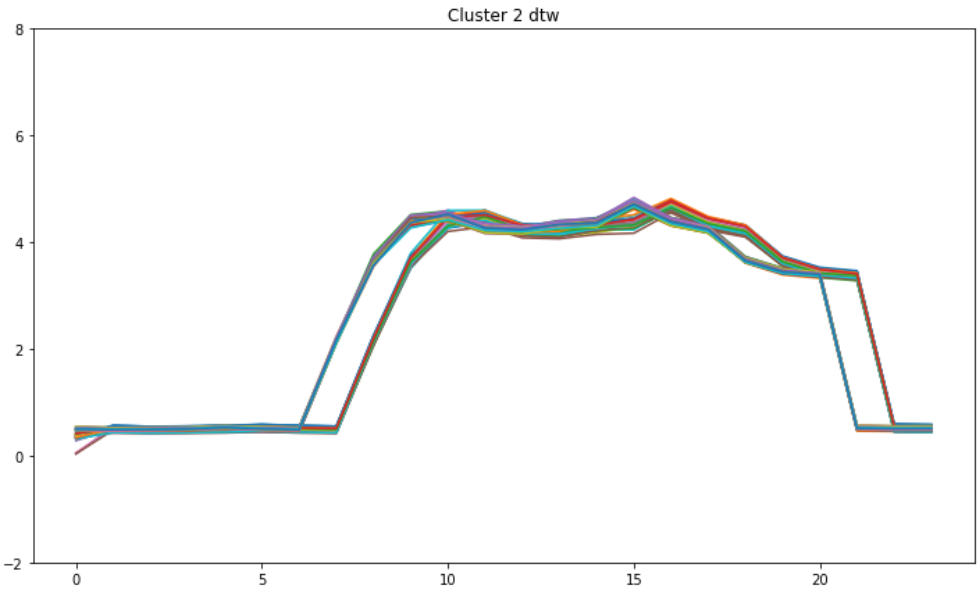
	Friday	Monday	Saturday	Sunday	Thursday	Tuesday	Wednesday
0	2.0	4.0	52.0	52.0	2.0	2.0	3.0
1	50.0	48.0	0.0	0.0	0.0	0.0	49.0
2	0.0	0.0	0.0	0.0	50.0	51.0	0.0



Représentation des données appartenant à chaque cluster


Cluster 0 dtw
8
6
4
2
0
−2
0
5
10
15
20


Cluster 1 dtw
8
6
4
2
0
−2
0
5
10
15
20


Cluster 2 dtw
8
6
4
2
0
−2
0
5
10
15
20

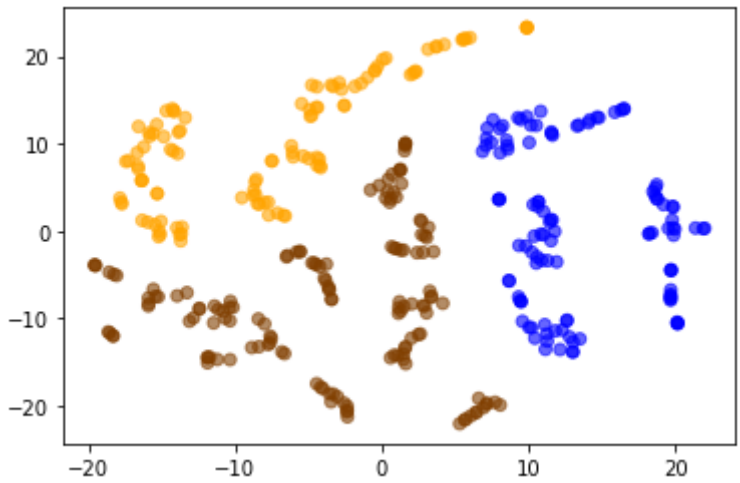
Réduction de dimension et représentation graphique

L'algorithme t-SNE (t-distributed stochastic neighbor embedding) est une technique de réduction de dimension pour la visualisation de données développée par Geoffrey Hinton et Laurens van der Maaten et publiée en 2008.

Il s'agit d'une méthode non linéaire permettant de représenter un ensemble de points d'un espace à grande dimension dans un espace de deux ou trois dimensions. Les données peuvent ensuite être visualisées sous la forme d'un nuage de points. L'algorithme t-SNE tente de trouver une configuration optimale selon un critère de théorie de l'information afin de conserver la proximité entre les points pendant la transformation : deux points qui sont proches (resp. éloignés) dans l'espace d'origine doivent être proches (resp. Éloignés) dans l'espace de faible dimension.

L'algorithme t-SNE se base sur une interprétation probabiliste des proximités. Une distribution de probabilité est définie sur les paires de points de l'espace d'origine de telle sorte que des points proches l'un de l'autre ont une forte probabilité d'être choisis tandis que des points éloignés ont une faible probabilité d'être sélectionnés. Une distribution de probabilité est également définie de la même manière pour l'espace de visualisation. L'algorithme t-SNE consiste à faire concorder les deux densités de probabilité, en minimisant la divergence de Kullback-Leibler entre les deux distributions par rapport à l'emplacement des points sur la carte.

L'algorithme t-SNE a été utilisée pour de nombreuses applications : analyse de la musique, recherche sur le cancer, bioinformatique, et le traitement de signaux biomédicaux. Cette méthode est souvent utilisée pour la visualisation de représentations de haut-niveau apprises par un réseau de neurones artificiel.




```
grouped_clusters = df_profil_day_cluster.groupby("clusters")
dfs_cluster = {cluster: group for cluster, group in grouped_clusters}

index_clusters = {}
for cluster, df_cluster in dfs_cluster.items():
    print(f"Index of DataFrame for Cluster {cluster}: {df_cluster.index.tolist()}")
    index_clusters[cluster] = df_cluster.index.tolist()
```

```
Index of DataFrame for Cluster 0: [Timestamp('2019-01-01 00:00:00'), Timestamp('2019-01-05
00:00:00'), Timestamp('2019-01-06 00:00:00'), Timestamp('2019-01-12 00:00:00'), Timestamp('
2019-01-13 00:00:00'), Timestamp('2019-01-19 00:00:00'), Timestamp('2019-01-20 00:00:00'),
…
```

```python
import pandas as pd
def convert_to_days_of_week(timestamps):
    # Convertir les Timestamps en objets Pandas Timestamp
    timestamps_pd = pd.to_datetime(timestamps)

    # Extraire les noms des jours de la semaine
    days_of_week = timestamps_pd.day_name()

    return days_of_week.tolist()
result = {}
for cluster, index_list_for_clust in index_clusters.items():
    result[cluster] = convert_to_days_of_week(index_clusters[cluster])
    print(f"Cluster {cluster}: {result[cluster]}")
```

```
Cluster 0: ['Tuesday', 'Saturday', 'Sunday', 'Saturday', 'Sunday', 'Saturday', 'Sunday', 'S
aturday', 'Sunday', 'Saturday', 'Sunday', 'Saturday', 'Sunday', 'Saturday', 'Sunday', …

Cluster 1: ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Monday', 'Tuesday', 'W
ednesday', 'Thursday', 'Friday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', …

Cluster 2: ['Wednesday', 'Thursday', 'Friday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday'
, 'Friday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Monday', 'Tuesday', 'W
ednesday', 'Thursday', 'Friday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'M
onday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Monday', 'Tuesday', 'Wednesday',…
```

Interprétation des résultats pour le clustering avec la métrique optimale, « dtw » dans notre cas

Cluster 0 :

- Ce cluster est caractérisé par une forte concentration de jours de consommation le samedi et le dimanche.

- Dans le tableau, on observe que pour le premier ensemble de données (cluster 0), il y 52 occurrences le samedi, 52 occurrences le dimanche, et une occurrence le vendredi, le lundi, le mardi, le mercredi et le jeudi.

Cluster 1 :

- Ce cluster présente une distribution plus uniforme des jours de consommation, avec une prédominance le vendredi, le lundi et le mardi.

- Pour le deuxième ensemble de données (cluster 1), on observe 30 occurrences le vendredi, 28 occurrences le lundi, le mardi, et le mercredi, et 29 occurrences le jeudi.

Cluster 2 :

- Ce cluster est également caractérisé par une concentration de jours de consommation le vendredi, le lundi, le mardi, et le mercredi.
  Pour le troisième ensemble de données (cluster 2), on observe 21 occurrences le vendredi, le lundi, le mardi et le mercredi, et 22 occurrences le jeudi.
- Ces résultats suggèrent que les clusters ont été formés en regroupant les jours de consommation d'électricité en fonction de leurs similitudes. Chaque cluster représente un motif distinct de consommation sur les jours de la semaine, avec des concentrations différentes selon le cluster.

Cluster 0 - Forte concentration le samedi et le dimanche :

- Absence d'Activités : Le week-end, les bureaux sont généralement fermés, ce qui entraîne une baisse significative de la consommation d'électricité. Les équipements non essentiels sont éteints, contribuant à une faible consommation.

Cluster 1 et cluster 2 : Jours Ouvrables Classiques :

- La répartition relativement uniforme en semaine suggère une utilisation régulière des équipements de bureau pendant les heures de travail normales.
- La répartition peut être influencée par des pratiques de télétravail fréquentes ce jour-là, où les employés travaillant à domicile maintiennent une consommation d'énergie similaire à celle du bureau.
- Ces résultats peuvent être attribuée à une combinaison d'activités au bureau et de télétravail. Certains employés peuvent choisir de travailler à domicile le vendredi, influençant ainsi la consommation électrique à la fois au bureau et chez eux.

Facteurs Généraux :

- Flexibilité du Télétravail : Les politiques de télétravail flexibles peuvent entraîner des variations dans les motifs de consommation, avec des jours de consommation plus élevée au bureau et à domicile.

  Politiques d'Économie d'Énergie : Des politiques d'économie d'énergie, comme l'extinction automatique des lumières et des équipements non essentiels en dehors des heures de travail, peuvent influencer les motifs de consommation.

- Équipements de Bureau : Certains équipements de bureau, tels que les ordinateurs, les imprimantes, et l'éclairage, peuvent contribuer de manière significative à la consommation d'électricité. Des cycles réguliers d'utilisation de ces équipements peuvent influencer les motifs.

7. Optimisation de l’algorithme machine learning

Pour répondre à cette problématique nous avons fait les démarches suivantes :

- **Choix du Nombre de Clusters :** Utilisation de la méthode du coude pour déterminer le nombre optimal de clusters. Expérimentation avec différentes valeurs de k et évaluation des performances du modèle en utilisant l’inertie.

- **Évaluation des Performances :** Utilisation des métriques d'évaluation (la silhouette et l’inertie) pour évaluer la qualité des clusters obtenus.

- **Paramètres de l'Algorithme :** Expérimentation avec les paramètres spécifiques à l'algorithme (exemple : la métrique, la valeur du paramètre n_init, la valeur du nombre des clusters), la validation des résultats se faisant en calculant le score de la silhouette et l’inertie pour chaque cas à la fois.

- **Interprétation des Résultats :** Les résultats du clustering sont interprété. Une analyse du sens dans le contexte de chaque problème est réalisée en chaque fin d’étude.

- **Algorithme Alternatif :** Des essayes d'autres algorithmes de clustering, tels que DBSCAN or MeanShift ont été également utilisé, mais les performances obtenues (calculé avec le score de la silhouette ou l’inertie) n’ont pas égalé la performance des algorithmes présentés dans cette étude (TimeseriesKmeans).

- **Reduction de dimensionalité et visualisation des resultats :** L'évaluation correcte de la performance d'un clustering est toujours un problème ouvert, et pas seulement pour le clustering de séries temporelles. Comme la visualisation et les mesures scalaires sont les principales techniques d'évaluation de la qualité du clustering dans notre cas (24 or 7 dimensions), pour pouvoir avoir une visualisation des résultats facilement interprétable nous utilisons une méthode de réduction de dimensionalité (t-SNE) et l’affichage graphique des résultats ainsi obtenus.

8. Améliorations possibles et recommandations

Plusieurs améliorations seront possibles pour rendre ce projet plus facile à utiliser et plus performant :

- **La mise en place d'une base de données et d'un backend API**

L'intégration d'une base de données ainsi que l'ajout d'un backend api dans le projet présentent plusieurs avantages, notamment un stockage efficace et structuré des données, un accès rapide aux informations, une évolutivité pour gérer l'expansion des données, une intégration aisée avec d'autres systèmes, une sécurité renforcée des données, et une optimisation de la constante de temps.

La décision de ne pas avoir intégré actuellement cette partie est motivée par la constante de temps mais aussi par la nécessité de pouvoir appliquer ces algorithmes sur des données générées en tant que résultats de certains logiciels d'Efficacity. Cette approche permet d'utiliser l'algorithme sur des données autres que celles provenant d'une base de données structurée. Ainsi, la mise en place d'une base de données n'est pas indispensable pour tous les utilisateurs, dépendamment de l'objectif visé.

La flexibilité offerte par cette approche permet l'application des algorithmes de clustering sur des ensembles de données divers, qu'ils proviennent de logiciels tiers, de fichiers CSV, ou d'autres formats. L'accent est mis sur la préparation des données de manière à ce qu'elles soient adaptées à l'algorithme de clustering choisi, sans nécessiter une infrastructure de base de données formelle.

- **La création d'un frontend convivial pour les utilisateurs**

L'introduction d'une interface utilisateur conviviale (frontend) améliorerait significativement l'interaction avec le logiciel, rendant les résultats plus accessibles. Une interface intuitive, des outils de visualisation graphique, des fonctionnalités interactives et une documentation intégrée contribueraient à faciliter la compréhension des profils de consommation d'électricité, offrant aux utilisateurs une expérience plus efficace et adaptable.

- **Automatiser le choix du nombre optimal des clusters et des paramètres optimales des algorithmes de clusterisation**

Comme nous savons déjà, les algorithmes tels que K-means nécessitent une précision du nombre optimal des clusters qu'ils doivent réaliser ainsi que la précision des valeurs optimales pour les paramètres nécessaires pour la clusterisation et comme nous savons aussi, ces choix ne sont pas toujours des choix faciles à faire.

Pour l'instant nous disposons comme méthode, l'interprétation de la méthode du coude et l'interprétation des divers résultats de calcul des inerties or des scores de silhouettes avec des paramètres différents. Ça peut prendre du temps mais aussi, l'interprétation peut être

erronée où les données obtenues peuvent ne pas donner une solution précise (par exemple, la méthode du coude n'est pas toujours interprétable). C'est pourquoi l'automatisation de cette sélection peut apporter des avantages majeurs. Elle économise du temps, garantit l'objectivité, la répétabilité et l'adaptabilité du processus, éliminant ainsi le biais subjectif. Une approche automatisée pourra optimiser les performances des clusters.

Parfois la méthode du Coude peut ne pas être interprétable, et dans ce cas, explorer d'autres pistes et mettre en place d'autres méthodes supplémentaires est une excellente idée.
Par exemple, l'utilisation d'autre métriques pour réaliser ce choix, comme le score de la silhouette ou le **Davies-Bouldin Index** qui mesure la compacité des clusters et leur séparation (un indice plus bas indique une meilleure partition des données)

Base de données, backend en mode API et frontend consommateur de notre backend API

Actuellement, tout fonctionne dans un Jupiter Notebook pour une première approche d'étude, mais il y a beaucoup d'avantages de faire évoluer cela et de transformer cette étude dans une application plus complexe, avec une bdd, un backend et un frontend consommateur de ce backend.

L'ajout d'une base de données (BDD) à notre projet de clustering peut apporter plusieurs avantages significatifs. Voici quelques raisons pour lesquelles l'intégration d'une BDD et d'un backend sont envisageable dans ce projet de clustering :

1. **Stockage des Données :** Une base de données permettra de stocker de manière efficace les données nécessaires au clustering. Les algorithmes de clustering nécessitent un accès rapide et efficace aux données pour fonctionner correctement, et une base de données bien conçue peut fournir cela.

2. **Gestion des Données en Temps Réel :** Un backend peut être utilisé pour récupérer, traiter et mettre à jour les données en temps réel. Cela est essentiel si les données du projet de clustering évoluent constamment, et nous avons besoin de mettre à jour les résultats du clustering en conséquence.

3. **Sécurité des Données :** Un backend peut jouer un rôle crucial dans la sécurisation des données, en mettant en place des mécanismes d'authentification, d'autorisation et de chiffrement. Cela est particulièrement important car dans notre cas les données de clustering contiennent des informations qui ne sont pas en libre accès.

4. **Évolutivité :** Un backend bien conçu peut rendre le projet plus scalable. Il peut gérer de manière efficace un grand nombre de requêtes, ce qui est crucial si un projet de clustering qui est amené à traiter des volumes massifs de données.

5. **Facilitation du Développement :** Un backend offre une interface entre le frontend (interface utilisateur) et la base de données. Cela peut simplifier le développement en permettant une séparation claire entre la logique métier et la gestion des données, facilitant ainsi la maintenance et l'évolution du code.

6. **Réutilisation du Code :** En intégrant un backend, la logique pourrait être encapsulé pour la rendre plus modulaire et facilement réutilisable dans d'autres parties du projet ou dans de futurs projets

7. **Communication avec le Frontend :** Avec un backend, la communication avec le frontend serait plus propre et flexible. Nous allons pouvoir définir des API pour transmettre les résultats du clustering à l'interface utilisateur, facilitant ainsi la visualisation et l'interaction.

En résumé, l'ajout d'un backend et d'une BDD à notre projet de clustering peut apporter des avantages considérables en termes de gestion des données, de sécurité, de scalabilité et de facilitation du développement, contribuant ainsi à la robustesse et à la flexibilité de cette solution.

Pas à suivre dans la création d'une base de données et l'ajout d'un backend et d'un frontend

Pour optimiser la gestion de données dans le cadre de cette étude, l'utilisation d'une base de données **NoSQL** se révèle appropriée. Le choix du NoSQL est représenté par le besoin de la gestion de grandes quantités de données.

Pour implémenter cette base de données, une approche consiste à générer des fichiers .json à partir des fichiers .xlsx existants, lesquels seront ensuite utilisés pour alimenter la base de données.

En termes de construction de la base de données, une solution envisageable serait de construire un **backend** avec **Django**, un framework Python, et l'exploiter pour mettre en place une base de donnée**s ArangoDB** comme système de gestion de base de données NoSQL.

Un script pourrait être développé pour se connecter à ArangoDB, créer la base de données et les collections requises, puis ajouter les données issues des fichiers Excel, lors du premier démarrage de l'application backend.

Dans le cadre de la récupération des données nécessaires en fonction des objectifs spécifiques de l'étude, mais aussi en prévision de l'ajout d'un frontend, la conception de routes, chacune correspondant à une demande en cohérence avec le frontend qui sera conçu sera nécessaire.

La mise en place des fonctions réutilisable sera aussi un aspect important de notre backend et permettra de réutiliser le code.

Pour chaque route, les algorithmes spécifiques présentés déjà dans les diverses études de ce projet seront intégrés, et les résultats seront renvoyés en réponse aux requêtes.

C'est pour ce but que l'automatisation du nombre des clusters et des paramètres optimales du cluster est un élément important.

Ça permettra la réalisation d'une étude entière, sans avoir une interaction avec l'utilisateur après le démarrage d'une étude.

En ce qui concerne le choix d'un **frontend** consommateur de notre backend en mode API, nous pouvons opter pour un frontend réalisé avec le framework **Vue.js et Chart.js** pour générer les affichages graphiques.

Pages et fonctionnalités possibles :

Page de Connexion :

- La première page de l'interface sera dédiée à la connexion, comme les données ne doivent pas être accessible que par des utilisateurs avisés. L'utilisateur devra avoir un compte et fournir ses identifiants pour accéder au système.

Page de Choix d'Options :

- Après la connexion réussie, l'utilisateur sera dirigé vers une page de choix d'options.
- Cette page affichera deux options principales :
    1. **Création d'une Nouvelle Étude :**
        - Si l'utilisateur choisit cette option, il sera redirigé vers une page dédiée à la création d'une nouvelle étude.
    2. **Visualisation des Résultats des Études Réalisées Avant :**
        - Si l'utilisateur choisit cette option, il aura accès aux résultats des études précédemment réalisées.

Page de Création d'une Nouvelle Étude :

- Sur cette page, l'utilisateur aura la possibilité de choisir entre deux types d'études :
    1. **Étude de Consommation des Profils Journaliers pour un Bâtiment :**
        - L'utilisateur devra fournir l'ID du bâtiment qu'il souhaite étudier.
        - Il pourra spécifier le type de consommation d'électricité (par exemple, domestique, commercial, industriel).

        - Après ça il pourra être dirigé vers la page des résultats où il pourra consulter les résultats proposés par le clustering de ces données.
    2. **Étude de Clustering pour Tous les Bâtiments :**
        - L'utilisateur pourra renseigner des informations spécifiques liées aux données qu'on souhaite clusteriser, par exemple, le secteur et la région pour lesquels nous souhaitons réaliser cette étude
    3. Pour pouvoir valider la demande d'utilisation du logiciel pour des études sur d'autre résultats en plus que ce qui se trouvent dans la bdd, des resultats des simulations réalisées avec des logiciels d'Efficacity, nous pouvons ajouter le choix : **Réaliser une étude avec ma propre donné** qui va supposer à uploader une courbe de charge qui sera étudié.

Page de Visualisation des Résultats des Études :

Une fois l'utilisateur choisi l'option de visualisation des résultats des études, la page affichera les résultats en fonction du type d'étude réalisé.

1. **Étude de Consommation Totale sur une Année pour Tous les Bâtiments :**
    - Affiche les résultats agrégés de la consommation totale sur une année pour l'ensemble des bâtiments inclus dans l'étude.
    - Graphiques et tableaux dynamiques illustrant les tendances de consommation, les pics, etc.
    - Fournit des filtres pour permettre à l'utilisateur de segmenter les résultats par type de bâtiment, par région, ou d'autres paramètres pertinents.
2. **Résultats du Clustering des Profils Journaliers d'un Bâtiment Particulier :**
    - Présente les résultats détaillés du clustering pour un bâtiment spécifique.
    - Affiche visuellement les clusters formés avec des graphiques représentant les profils journaliers de consommation.
    - Propose des métriques d'évaluation du clustering (le cas échéant), fournissant une compréhension approfondie de la qualité du modèle.
    - Intègre des fonctionnalités interactives pour permettre à l'utilisateur d'explorer les clusters et les caractéristiques de chaque groupe.
3. **Etude réalisés sur des données uploadées**
    - Présente les résultats détaillés du clustering pour les données uploadées.

Enregistrement de la Version de la BDD et Suivi des Mises à Jour :

Pour assurer la traçabilité des résultats, il est essentiel d'enregistrer la version de la base de données utilisée pour chaque étude, car les bases de données sont enrichies et mise à jour régulièrement.

- Pour chaque étude des deux premiers groupes nous allons enregistrer la version de bdd utilisé et au niveau de l'affichage nous allons préciser clairement dans les résultats la version de la base de données qui a été utilisée pour générer ces résultats (des informations telles que la date de la dernière mise à jour de la base de données) et une notification ou un avertissement si la version de la base de données utilisée pour des certain études est obsolète par rapport à la dernière version disponible.

Cela garantit la transparence quant à l'origine des résultats et permet à l'utilisateur de prendre des décisions éclairées en tenant compte de l'évolution constante des données dans le contexte d'Efficacity.

D’autre améliorations possibles :

Feedback en Temps Réel : Donne un feedback en temps réel à l'utilisateur pour l'informer sur la progression de son processus de création d'étude (car on sait que quelquefois une analyse de ce genre peut prendre du temps).

L’ajout du docker pourrait être envisagé pour garantir un déploiement rapide et fiable de l’application, en garantissant la portabilité et la cohérence de l’environnements de travail.

Conclusion

Le présent travail rend compte des bénéfices que la clusterisation des profils de charge électrique quotidiens des bâtiments, dans ce cas précis, des bâtiments non résidentiels peut apporter.

Les conclusions tirées pour chaque étude démontrent l'importance de ce projet et mettent en lumière la pertinence et l'impact significatif de cette initiative, soulignant son rôle crucial dans la compréhension approfondie des données et dans la prise de décisions éclairées en ce qui concerne la validation des bases de données d'Efficacity, réalisée avec des étapes d'enrichissement stochastique et à l'aide des réseaux bayésiennes, mais aussi par l'utilisation des outils de simulations des entreprises partenaires.

Personnellement, ce projet représente le premier projet big data que je réalise. Ce projet a constitué donc une opportunité d'apprentissage enrichissante, m'offrant une compréhension approfondie du domaine de la big data, des diverses méthodes qui peuvent être employés pour le traitement de la big data, et de l'intelligence artificielle, notamment en ce qui concerne le clustering appliqué dans ce cas à la consommation d'électricité. J'ai acquis des connaissances substantielles et stimulantes qui ont non seulement élargi mes compétences, mais ont également suscité un intérêt accru pour les applications innovantes de l'IA dans le secteur de l'énergie.