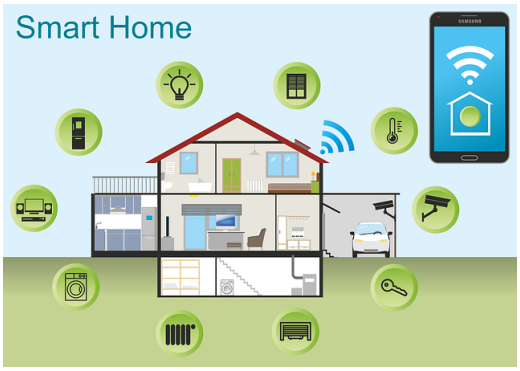
École Nationale Supérieur de l'Eau, Energie et Environnement

Rapport de projet 1A

GESTION D'ENERGIE



Elèves :
GRANDET Rémi
SERASSE Clément

Tuteurs : FERRARI Jérôme
AMAYR Manar

6 Juin 2022

Table des matières

1 Contexte **2**

2 Problématisation **2**

3 Gestion des données **3**

3.1 Récupération des données 3

3.2 Traitement des données 4

3.3 Import des données sur python 5

4 Résolution **6**

4.1 Utilisation des données 6

4.2 Résultats de notre prédiction 6

4.3 Méthode Clustering 7

5 Interprétation énergétique **7**

5.1 Pertes d'énergies 8

5.2 Pertes économiques 9

5.3 Bilan carbone 9

6 Conclusion **10**

7 Annexe **11**

8 Sources **16**

1 Contexte

L'augmentation de la population mondiale ainsi que le réchauffement climatique vont entrainer dans le futur une forte augmentation de la consommation d'énergie. Cependant, l'énergie est un domaine très sensible où nos principales sources d'énergie sont amenées à s'amoindrir. C'est pourquoi il est nécessaire d'utiliser de différents moyens de production mais surtout à mieux utiliser son énergie individuellement et collectivement. La gestion d'énergie est donc essentielle pour le futur.

2 Problématisation

Nous nous intéressons alors dans cet article à la gestion d'énergie dans une maison d'un particulier. En effet, nous avons eu accès aux données récoltées sur la maison projet ExpeSmarthouse du laboratoire G2Elab. C'est une maison dans laquelle vit 5 personnes dans 120 m^2. Elle est équipée de plus de 360 capteurs tels que des capteurs de température, de bruit, de CO_2, ou bien de la consommation énergétique. Ces capteurs nous serviront à déterminer de quelle manière nous pouvons économiser de l'énergie chez un particulier.

Avec les données que nous avons observé concernant la maison, nous avons remarqué qu'il y avait plusieurs périodes dans l'année et même dans les journées où l'on observait des minimums de consommation, en observant certaines données qui sont révélatrices d'une présence ou non. De plus durant cette période d'absence, de nombreux appareils continuent de consommer alors qu'ils ne sont pas indispensables.

A l'aide des différentes méthodes de prédiction que nous avions vues en cours, nous avons donc décidé d'axer notre travail sur la prédiction de la présence de personnes dans la maison. Nous déterminerons en conséquence les pertes énergétiques et l'impact environnemental de ces appareils.

Pour notre étude nous allons nous focaliser sur le salon de la maison, néanmoins notre approche du problème pourra être généralisé à la maison entière.

Pour déterminer si une personne était présente dans la maison, nous nous sommes appuyés sur les données du bruit, du capteur CO_2, de la consommation de la télévision ainsi que de l'état de la lumière dans le salon. Ces données nous paraissaient les données les plus pertinentes vis-à-vis de l'occupation du salon. D'autres capteurs étaient mis à notre disposition mais avec la restiction de temps, il était préférable de se limiter à quatres capteurs. Pour les pertes énergétiques entrainées par une consommation d'énergie d'un appareil quand personne n'est présent, nous avons choisi de regarder la consommation des appareils multimédia, de la freebox ainsi que du garage. Nous avons choisi ces appareils car ce sont des appareils pour lequel il n'est pas nécessaire qu'ils soient allumés quand personne n'est là, contrairement à un congélateur par exemple. Encore une fois, nous avons fait le choix de ces appareils mais nous aurions pu en considérer beaucoup plus, pour mieux expliciter la perte d'énergie.

Voici le panel d'objet que nous aurions pu utiliser avec les différentes puissances électriques de chaque appareil de la maison :

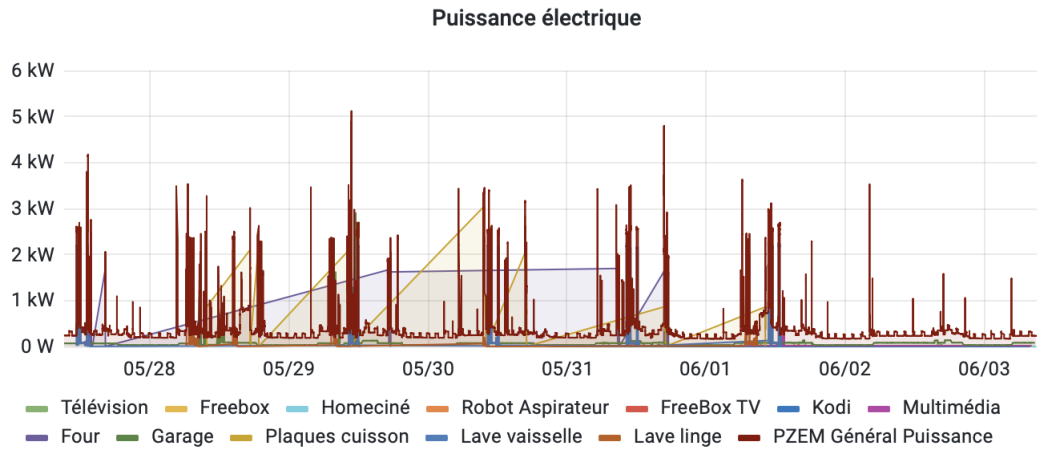


Fig.1 Données observées sur Expe-Smarthouse pour la puissance électrique consommées dans le salon

3 Gestion des données

3.1 Récupération des données

Pour répondre à notre problème, nous avons du dans un premier temps extraire les données des différents capteurs de notre futur modèle. Pour se faire, nous avions accès au site Grafana, où se trouve la base de donnée des 360 capteurs de la maison. Pour cela nous avions vu dans la première séance de notre Bureau Expérimental comment accéder à ces données à l'aide d'un raspberry. Cela traitait directement les données de capteurs que nous avions à disposition dans la salle de classe, puis nous avons mis en œuvre ce protocole sur python appliqué à notre maison. Nous devions alors rentrer différents identifiants sur python pour pouvoir accéder à nos données à l'aide de la database InfluxDB. On pouvait ainsi choisir à partir de quand et pendant combien de temps on voulait étudier ces données. On devait aussi identifier la donnée qui nous intéressait par un code fournit par l'enseignant. Cela nous donnait un code de la sorte :

```
PORT_NUMBER = 1234
# InfluxDB Server parameters
INLUXDB_SERVER_IP = '82.65.155.71'
INLUXDB_SERVER_PORT = 8086
INFLUXDB_USERNAME = 'eleves'
INFLUXDB_PASSWORD = 'SmarthouseG2Elab'
INFLUXDB_DB_NAME = 'jeedom'
############################

client = InfluxDBClient(INLUXDB_SERVER_IP, INLUXDB_SERVER_PORT, INFLUXDB_USERNAME, INFLUXDB_PASSWORD, INFLUXDB_DB_NAME,ssl=True,
verify_ssl=False)

print(client.get_list_database())

client.switch_database('jeedom')

#### Timestamps in seconds ##########"
timestamp_start= 1640995200  ## timestamp of the begin of period required (in second use https://www.epochconverter.com/)
timestamp_end= 1653177600  ## timestamp of the end of period required (in second use https://www.epochconverter.com/)

#### Convertion of timestamp to datetime
dt_object_start = datetime.fromtimestamp(timestamp_start)
print(dt_object_start)
dt_object_end = datetime.fromtimestamp(timestamp_end)
print(dt_object_end)

#### Preparation of the query
query = 'SELECT "value" FROM "jeedom"."autogen"."3888" WHERE time >= $start_time  AND time < $end_time '
bind_params = {'end_time': str(dt_object_end), 'start_time': str(dt_object_start)}
```

Fig.2 Accès aux données

Cette étape nous a permis d'accéder à nos données. Cependant, on ne peut toujours pas les utiliser car elles ne sont pas triées ni rangées sur python ou un autre type de fichier. Nous allons donc ranger ces données dans un fichier de type CSV pour pouvoir ensuite les utiliser sur python. Ainsi elles seront exploitables. Nous utilisons alors le script suivant qui nous a été fourni pour pouvoir mettre ces données dans un fichier CSV :

```
#### Preparation of the query
query = 'SELECT "value" FROM "jeedom"."autogen"."3888" WHERE time >= $start_time  AND time < $end_time '
bind_params = {'end_time': str(dt_object_end), 'start_time': str(dt_object_start)}

## query+Printof result

datasheet = client.query(query, bind_params=bind_params)

print(datasheet)

exported_data = list(datasheet.get_points())
header_list = list(exported_data[0].keys())

with open("bruit1.csv", "w", newline='') as fp:
    writer = csv.writer(fp, dialect='excel')
    print(header_list[1:])
    value_header = header_list[1]
    offset = sum(c.isalpha() for c in value_header)
    print(offset)
    #header_list[1:] = sorted(header_list[1:], key=lambda x: int(x[offset:]))
    header_list[1:] = ['value']
    # print(header_list)
    writer.writerow(header_list)
    for line in exported_data:
        # print(line)
        writer.writerow([line[kn] for kn in header_list])
```

Fig.3 Création des fichiers CSV

Ceci étant fait, nous nous retrouvions avec un fichier CSV brut avec la date et heure de prise de donnée du capteur ainsi que la valeur associée. Malheureusement, nous nous sommes rendus compte, une fois que nous avions les CSV pour chaque paramètre que l'on voulait étudier que nous ne pouvions toujours pas exploiter les données. En effet, ces différents fichier CSV n'étaient pas exploitables pour notre code python du fait des horaires différents de prise de valeur pour chaque capteur. Nous devons alors encore modifier ces CSV avant de les utiliser pour mener à bien notre projet. Nous avions des fichiers CSV de la forme suivante :

```
time,value
2022-04-22T02:00:28.057421Z,678.0
2022-04-22T02:05:41Z,732.0
2022-04-22T02:15:27.610918Z,693.0
2022-04-22T02:25:51Z,712.0
2022-04-22T02:30:29.081837Z,671.0
2022-04-22T02:35:56Z,708.0
2022-04-22T02:45:26.572575Z,672.0
2022-04-22T02:56:06Z,678.0
2022-04-22T03:00:40.099650Z,661.0
2022-04-22T03:06:12Z,693.0
2022-04-22T03:26:22Z,671.0
2022-04-22T03:28:56.150723Z,668.0
2022-04-22T03:30:37.038550Z,646.0
2022-04-22T03:36:27Z,672.0
2022-04-22T03:45:28.414588Z,651.0
2022-04-22T03:56:38Z,661.0
2022-04-22T04:00:33.772157Z,635.0
2022-04-22T04:06:43Z,668.0
2022-04-22T04:26:53Z,646.0
2022-04-22T04:30:34.686835Z,631.0
2022-04-22T04:36:58Z,651.0
2022-04-22T04:45:23.575802Z,639.0
2022-04-22T04:57:07Z,635.0
2022-04-22T05:00:34.895268Z,628.0
```

Fig.4 Exemple de fichier CSV pas encore exploitable

En effet, nous pouvons voir que en plus de notre problème de pas de temps, nous avons également un problème de séparation des paramètres. Il y a des 'T' entre la date et l'heure, et des 'Z,' entre l'heure et la valeur associée. Surtout à cause du séparateur 'Z,' nous avions des problèmes car notre code python pour utiliser par la suite ces données ne reconnaissait pas la valeur comme un nombre réel mais comme une chaîne de caractère. Nous devons ainsi traiter ces données pour pouvoir les rendre enfin utilisables.

3.2 Traitement des données

Le plus gros problème que nous avons rencontré était de rendre nos fichiers CSV exploitables aux yeux de notre programme. Il existe une bibliothèque python nommée "pandas" qui peut gérer ce genre de problème. Cependant, comparé aux années précédentes, nous n'avons pas été formé sur ce traitement de données, et donc nous avons perdu beaucoup de temps à chercher une solution. Arrivé un moment où notre professeur référent a pris les chose en main et nous a fourni un fichier CSV pour chaques capteurs partiellement triés.

Dans ce premier traitement de données, un problème majeur a été resolu. En effet, les différents capteurs n'ont pas la même période de prise de données, donc nous nous retrouvions avec des temps et un nombre de données différents pour chaques type de données. Ceci est très embettant pour la création de notre modèle de prédiction car celui-ci met en relation les données d'un même temps pour pouvoir déterminer une solution. Un pas de temps à donc été appliqué aux données, dans notre cas il est de 30 min entre chaque prise de données. Aussi, un intervale de temps à été mis en place, nous avons décidé d'étudier notre cas sur un mois entier.

Ensuite nous avons modifier le CSV directement sur Excel en remplaçant simplement les 'Z,' par des tabulations, ce qui a permis au programme python de ne plus confondre les chaîne de caractère et les réels. Enfin, nous avons réussi à mettre les données à la forme que nous voulions pour pouvoir les utiliser au mieux sur python. Nous avons ainsi obtenu ce type de fichier CSV :

```
time value
2022-04-22-02:00:00 703.75
2022-04-22-02:30:00 682.25
2022-04-22-03:00:00 673.25
2022-04-22-03:30:00 657.5
2022-04-22-04:00:00 649.66
2022-04-22-04:30:00 639.0
2022-04-22-05:00:00 631.25
2022-04-22-05:30:00 630.0
2022-04-22-06:00:00 634.75
2022-04-22-06:30:00 613.75
2022-04-22-07:00:00 520.25
2022-04-22-07:30:00 463.5
2022-04-22-08:00:00 413.75
2022-04-22-08:30:00 402.66
2022-04-22-09:00:00 396.0
2022-04-22-09:30:00 398.0
2022-04-22-10:00:00 392.2
2022-04-22-10:30:00 394.0
2022-04-22-11:00:00 389.0
2022-04-22-11:30:00 393.25
2022-04-22-12:00:00 392.5
2022-04-22-12:30:00 404.75
2022-04-22-13:00:00 400.0
2022-04-22-13:30:00 428.75
```

Fig.5 Exemple de fichier CSV que l'on utilise, ici pour le capteur de CO_2

3.3 Import des données sur python

Avec le type de fichier CSV que l'on a obtenu, il était devenu plus facile pour nous d'utiliser ces données sur python. En effet, nous avions deux colonnes séparées d'une tabulation et qui d'une part était composée d'une date et de l'autre d'un nombre réel qui correspondait à la valeur retournée par le capteur à un instant donné. Il nous restait alors plus qu'à répartir ces différents types de données dans des listes à l'aide du séparateur qu'est la tabulation puisque les listes sont facilement utilisables sur python.

Nous avons réparti les différentes données de la manière suivante :

```python
Co2, temps = [], []
file = open("CO22.csv")
Title = file.readline()
for line in file:
    line_token = line.split(" ")
    temps.append(str(line_token[0]))
    Co2.append(float(line_token[1]))
file.close()
```

Fig.6 Extraction des données du CO_2 sur python

Nous pouvons dorénavant utiliser ces données pour réaliser notre méthode de prédiction pour résoudre notre problème.

4 Résolution

4.1 Utilisation des données

Avec les données que nous venons de collecter sous les bons formats, nous allons pouvoir utiliser notre méthode de résolution pour prédire la présence de personnes dans le salon. Nous allons utiliser la méthode de résolution clustering. C'est une méthode de prédiction que l'on utilise sur python avec l'algorithme KMeans de la bibliothèque sklearn.cluster. Nous devons donc ainsi faire correspondre les paramètres de l'algorithme avec nos données et nous voyons que les listes que nous avons créés ne sont pas celles nécessaires à l'utilisation de l'algorithme. En effet, nous devions répartir dans une même liste nos 4 données. Ainsi, dans cette liste pour chaque temps nous avons créé une autre liste où il y a les valeurs associées de chaque paramètre.

L'algorithme KMeans nécessite aussi la définition du nombre de cluster que nous voulons. Comme nous n'avons que deux états finaux, à savoir si une personne est présente ou non, nous avons choisi de n'utiliser que deux clusters (Nous allons expliquer la méthode clustering dans une prochaine partie). Nous avons donc réalisé le code suivant :

```
21 from sklearn.cluster import KMeans
22
23 X, X1 = [], []
24
25 # construction features qui vont servir à creer notre modèle
26 for i in range(len(Tv)):
27
28     X1.append(Bruit[i])
29     X1.append(Co2[i])
30     X1.append(Tv[i])
31     X1.append(Lumiere[i])
32     X.append(X1)
33     X1 = []
34
35 clustering = KMeans(n_clusters=2, init='k-means++', n_init=10, max_iter=1000,
36                     tol=0.0001, verbose=0, random_state=1, copy_x=True, algorithm='auto')
37 clustering.fit(X)
38
39 clustering_predict = clustering.predict(X)
```

Fig.7 Code pour utiliser l'algorithme KMeans

Nous pouvons voir que tout d'abord, on importe bien l'algorithme, puis on créé notre liste de listes pour le clustering. On utilise ensuite cet algorithme.Enfin, les deux dernières lignes de ce code nous permettent d'affecter la méthode de clustering à nos données

4.2 Résultats de notre prédiction

Lorsque nous exécutons ce code et l'affichons en fonction du temps, nous obtenons les graphes suivant :

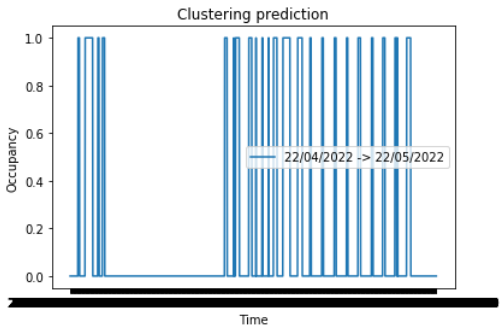


Fig.8 Prédictions de la méthode clustering

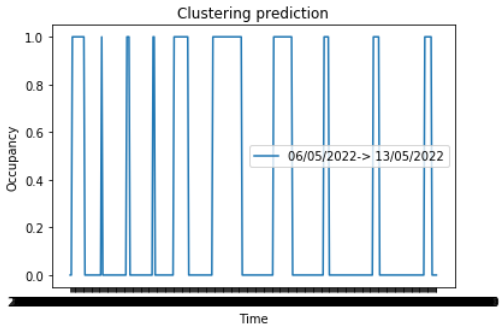


Fig.9 Prédictions de la méthode clustering pour une plus courte période

Ceci représente les prédictions que nous donnent l'algorithme en fonction du temps. Nous pouvons voir que les valeurs obtenues sont des successions de 0 et de 1 avec des paliers, et chaque valeur représente un état d'occupation. Lorsque nous regardons précisément à quelle date et quelle heure correspondent les valeurs, nous remarquons que la valeur 0 correspond à quand personne n'est présent et la valeur 1 quand le salon est occupé. On

remarque même que personne n'était présent dans la maison pendant l'espace d'une semaine. Cela correspond sûrement à une période de vacances.

Nous avons donc accès à l'occupation de la maison et nous pouvons ainsi déterminer les conséquences énergétiques d'une consommation d'énergie non nécessaire d'appareils lorsque personne n'est là.

4.3 Méthode Clustering

Avant d'expliciter les conséquences de l'inoccupation, nous allons expliquer ce qu'est le clustering et pourquoi nous avons choisi cette méthode de prédiction.

Tout d'abord, le clustering est une méthode de classification non supervisée qui permet d'analyser des données pour les regrouper selon leurs ressemblance, leur forme. Ainsi, deux données qui se ressemblent doivent être dans le même groupe et deux données qui sont éloignées dans leur forme doivent être dans deux groupes différents. Ces groupes sont appelés des clusters et correspondent à des états différents dans le sens de la prédiction. Dans notre cas d'occupation du salon, il y a deux états possibles soient si oui ou non une personne est présente, cela nous détermine ainsi notre nombre de clusters. On a alors deux clusters dans notre méthode de clustering.

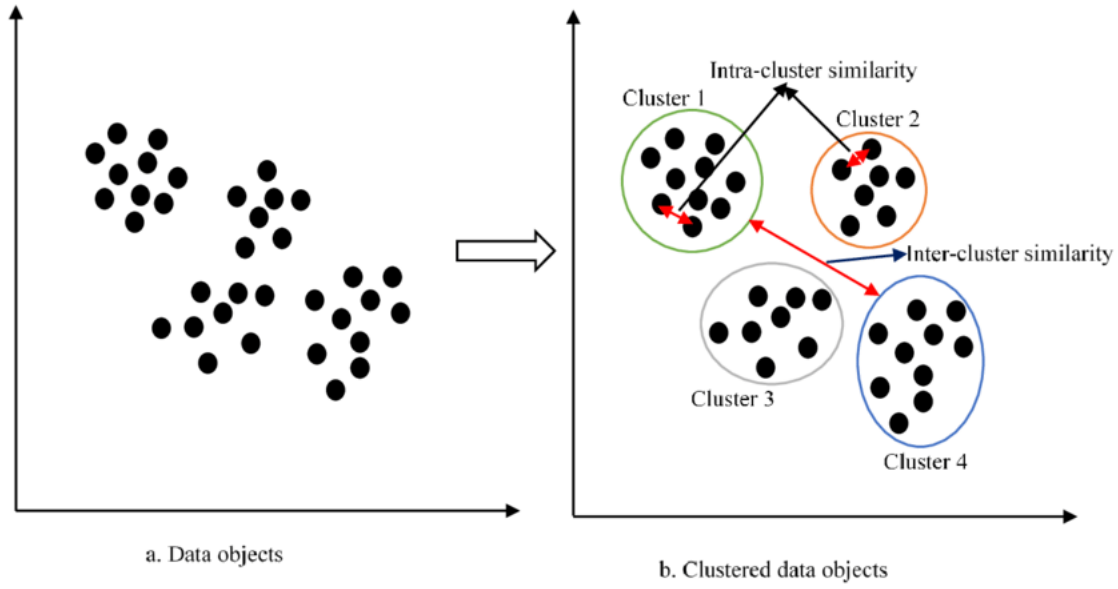


Fig.10 Schéma explicatif de la méthode clustering

Ensuite, nous allons expliquer ce qu'est l'algorithme KMeans, algorithme que nous avons utilisé. Selon la notice de sickit learn site sur lequel on a trouvé comment utiliser l'algorithme :

"L'algorithme KMeans regroupe les données en essayant de séparer les échantillons en n groupes de variance égale, en minimisant un critère connu sous le nom de somme des carrés d'inertie ou intra-groupe (voir ci-dessous). Cet algorithme nécessite que le nombre de groupes soit spécifié. Il s'adapte bien à un grand nombre d'échantillons et a été utilisé dans un large éventail d'applications dans de nombreux domaines différents."

Nous avons choisi la méthode de clustering plutôt que des méthodes de classification supervisées car contrairement au clustering, les méthodes de classification supervisées ont besoin pour prédire un état d'avoir déjà une base de données de cet état. Comme ici nous n'en avions pas, nous avons choisi la méthode de clustering.

5 Interprétation énergétique

Maintenant que nous avons notre système de prédiction, nous pouvons déterminer ce qui nous intéresse en gestion d'énergie.

5.1 Pertes d'énergies

Tout d'abord, nous pouvons déterminer la consommation d'énergie qui est inutile quand personne n'est présent. Comme expliqué dans la problématisation, nous avons regardé la consommation d'énergie de la Freebox TV, du garage et du multimédia. Nous avons donc cherché grâce à nos prédictions à savoir si quelqu'un était présent dans le salon et donc dans la maison. Dans ce cas, ces appareils peuvent être en fonctionnement ou avoir une certaine consommation d'énergie, dans le cas contraire, nous considérons que l'énergie consommée n'a pas lieu d'être, c'est de l'énergie perdue.

Cependant, il faut faire attention pour savoir à partir de quand cette énergie est réellement perdue car il peut n'y avoir personne dans le salon pendant deux heures sans que pour autant personne ne soit présent dans la maison et ait besoin de ces appareils dans un futur proche. Nous devons alors choisir à partir de combien de temps d'absence calculé avec notre prédiction nous considérons que personne n'est présent. Nous avons considéré plusieurs cas. Dans un premier temps, nous avons considéré un seuil à partir de six heures d'absences, cela prend donc en compte la nuit, où ces objets sont inutiles.

Ensuite, nous considérons un temps d'absence de vingt-quatre heures. Cela correspondrait à un temps durant lequel les résidents de la maison partent en week-end ou en vacances. Nous avons vu sur notre prédiction qu'il y avait en effet une période où les habitants étaient partis en vacances donc nous avons forcément des pertes dans cet espace de temps.

Nous obtenons donc les pertes suivantes :

Seuil choisi	Puissance perdue en kW	Rapport Puissance perdue/Puissance consommée
6 heures	67.39	0.76
24 heures	29.2	0.33

Fig.11 Tableau des pertes suivant le seuil d'absence

Nos résultats sur les pertes énergétiques nous montrent que les appareils que nous avons choisis représentent beaucoup de pertes par rapport à leur utilisation. En effet, lorsque l'on regarde le rapport Puissance perdue/Puissance consommée , on remarque que pour un seuil de temps de six heures, la puissance perdue représente un quart de la puissance totale. On en déduit que les appareils consomment énormément lorsqu'ils sont en veille. Avec le seuil de vingt-quatre heures, on remarque que même pendant des vacances, les appareils ne sont pas éteints et la puissance perdue représente un tiers de la puissance totale.

Il faudrait donc un système qui permette de débrancher ces appareils lorsque l'on prédit que personne ne sera là, la nuit par exemple, pour éviter ces pertes.

De plus nous pouvons calculer cette perte en kWh, pour cela il suffit juste d'utiliser la formule suivante : $(Nombred'heuresdefonctionnement)*(Nombredejoursdefonctionnement)*(Puissancedel'appareilenwatts)/1000$. Pour calculer la nombre d'heures de fonctionnement, nous avons déterminer le nombre de fois que la valeur relevée par le capteur est différente de 0, ceci durant 30 jours. Ensuite, une fois ce temps déterminer, nous divisons par la durée de notre modèle qui est de 30 jours pour avoir une moyenne journalière de fonctionnement des appareils. Nous avons déterminer de la même manière la puissance de l'appareil en Watt.

Nous obtenons le graphique suivant :

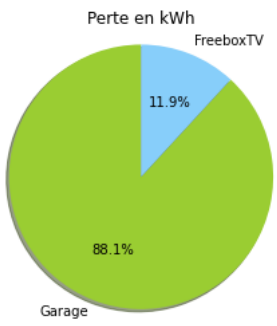


Fig.12 Graphique de la part de pertes des appareils

Appareils	Temps de fonctionnement inutile par jour (h)	Puissance moyenne (W)	Perte (kWh)
Garage	20.23	54	32.79
Multimedia	0.18	0.11	0.0006
FreeboxTV	20.63	7.12	4.41

Fig.13 Tableau de la perte énergétique en kWh avec un seuil de 24h

Nous pouvons remarquer une grosse dominance de perte de la part du garage comparer aux autres appareils. La part de perte du Multimédia étant inférieur à 1 %, nous avons décidé de ne pas la représenter. Cette valeur nous semble faussée car sur notre base de donnée ExpeSmartHouse, la puissance du multimédia était non négligeable, c'est pour cela que avions choisi ce capteur pour représenter notre modèle.
En regardant de plus près notre CSV "Multimédia", nous remarquons que beaucoup de valeurs sont nulles alors qu'elles ne devraient pas l'être. Ceci explique donc ce faible pourcentage de pertes, normalement elle devrait avoir une part beaucoup plus grande dans notre graphique.
Une autre remarque qui peut être faite est qu'on aurait pu prendre en compte la perte de la Freebox internet qui est constamment branchée sur le réseau avec une puissance de 3.5 W, ce qui représente sur un mois une puissance de 2.52 kWh. Malheureusement, nous n'avons pas pu avoir les données pour cette appareils.

Bien-sûr, d'autres capteurs pointant sur des appareils pourraient rentrer dans notre modèle, il suffit juste de généraliser notre méthode.

5.2 Pertes économiques

Lorsque l'on parle de perte énergétique, nous devons faire face à une perte économique au sein d'un foyer. Grâce au calcul des pertes en kWh, nous pouvons estimer cette somme.
En mai 2022, le prix du kWh d'électricité en France est de **0.1740€** (tarif réglementé - 6 kVA - Base). Avec un simple calcul nous arrivons au résultat de **6.47€** de perte économique sur notre durée de modèle (1 mois). Cela peut paraître peu mais si l'on se projette sur une année entière, cela représente plus de **70€**. Une somme non négligeable au vu des difficultés de certains foyers à faire face à l'augmentation de l'énergie en France.

5.3 Bilan carbone

Étant élève de l'Ecole Nationale Supérieur de l'Énergie, l'Eau et l'Environnement, nous sommes très sensibles à la situation climatique actuelle. C'est pour cela que nous avons modéliser les rejets de CO_2 en fonction de la source de production de l'énergie.

Temps	Moyen de production	Perte (kWh)	Rejet de CO_2 (kg)
30 jours	Nucléaire	37.19	4.46
	Gaz	37.19	15.62
	Charbon	37.19	30.5
1 an	Nucléaire	446.28	53.52
	Gaz	446.28	187.44
	Charbon	446.28	366

Fig.14 Tableau illustrant les rejet de CO_2 d'un foyer suivant le moyen de production

Nous pouvons remarquer qu'une masse conséquente de CO_2 peut être relâchée suivant le moyen de production. Ces chiffres nous permettent comparer ces rejets avec des éléments de la vie de tous les jours.
Nous avons fait le cas sur une maison de 5 personnes mais si nous élargissons le cas sur un pays entier, que nous faisons l'hypothèse que les maisons sont similaires à notre cas d'étude et qu'il y a 10 000 000 de foyers dans un pays quelconque. Nous arrivons sur des chiffres astronomiques de rejet de CO_2 de l'ordre de 0.535 Gkg pour les mêmes pertes sur 1 an en utilisant de l'énergie électrique issue du nucléaire.

On peut donc se rendre compte de l'impact environnemental de fuite d'énergie dans un foyer.

6 Conclusion

Ce projet nous a permis d'avoir une première approche du machine learning avec différentes méthodes comme Classification Tree, Random Forest, Clustering... Ce sont des outils très puissants qui sont et qui vont être très utiles dans l'avenir. Notamment pour la modélisation de modèles.

Durant ce projet, le plus compliqué a été de bien récupérer les données de la maison et de les mettre dans un format lisible sur python. La méthode de prédiction Clustering a été simple a implémenter puisque nous avions déjà travaillé sur ce sujet.

En ce qui concerne notre modèle, il y a plusieurs critiques qui peuvent être pointées du doigt. Dans un premier temps, nous aurions pu prendre beaucoup plus de données différentes pour le construire mais aussi prendre des données plus pertinentes, avec une meilleure corrélation pour avoir un modèle beaucoup plus précis. On remarque que le salon est très peu occupé, nous pensons donc que notre modèle n'est pas robuste.

Cependant, nous sommes quand même satisfait de ce projet en général puisque nous avons pu avoir une analyse profonde sur le problème de pertes énergétiques relevant les problèmes environnementaux actuels. Suite à cette analyse de notre modèle, nous avons pu voir l'importance de la gestion d'énergie d'un foyer avec les différentes valeurs que nous avons déterminées(économiques et rejets de CO_2). Ces valeurs sont sûrement évaluées à la hausse puisque notre modèle ressort beaucoup d'absence du salon.

Ce modèle pourra par la suite être généralisé à la maison entière en incluant d'autres capteurs qui donnent des informations sur les autres pièces de la maison. En faisant cela, le modèle sera beaucoup plus pertinent et précis, en conséquence les chiffres que nous avons déterminé seront beaucoup plus précis.

La gestion d'énergie sera dans le futur un domaine à approfondir pour gérer au mieux la transition énergétique.

7 Annexe

```python
# -*- coding: utf-8 -*-
"""
Created on Mon May 30 08:20:24 2022

@author: grandetr
"""

from http.server import BaseHTTPRequestHandler, HTTPServer
import urllib
import time
from influxdb import InfluxDBClient
import sys
from datetime import datetime
import csv
import pandas as pd
from sklearn import tree
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn import metrics
import numpy as np
from sklearn.cluster import KMeans


############################
#    SCRIPT SETTINGS
############################
# Set the port where you want the bridge service to run
PORT_NUMBER = 1234
# InfluxDB Server parameters
INLUXDB_SERVER_IP = '82.65.155.71'
INLUXDB_SERVER_PORT = 8086
INFLUXDB_USERNAME = 'eleves'
INFLUXDB_PASSWORD = 'SmarthouseG2Elab'
INFLUXDB_DB_NAME = 'jeedom'
############################

client = InfluxDBClient(INLUXDB_SERVER_IP, INLUXDB_SERVER_PORT,
    INFLUXDB_USERNAME, INFLUXDB_PASSWORD, INFLUXDB_DB_NAME,ssl=True,
    verify_ssl=False)

print(client.get_list_database())

client.switch_database('jeedom')


#### Timestamps in seconds ###########"
timestamp_start= 1640995200  ## timestamp of the begin of period required (
    in second use https://www.epochconverter.com/)
timestamp_end= 1653177600  ## timestamp of the end of period required (in
    second use https://www.epochconverter.com/)

#### Convertion of timestamp to datetime
dt_object_start = datetime.fromtimestamp(timestamp_start)
print(dt_object_start)
dt_object_end = datetime.fromtimestamp(timestamp_end)
print(dt_object_end)
```

```
53
54 #### Preparation of the query
55 query = 'SELECT "value" FROM "jeedom"."autogen"."3888" WHERE time >=
       $start_time  AND time < $end_time '
56 bind_params = {'end_time': str(dt_object_end), 'start_time': str(
       dt_object_start)}
57
58 ## query+Printof result
59
60 datasheet = client.query(query, bind_params=bind_params)
61
62 print(datasheet)
63
64 exported_data = list(datasheet.get_points())
65 header_list = list(exported_data[0].keys())
66
67 with open("output1.csv", "w", newline='') as fp: ##Ici output1.csv pour ne
       pas  craser  les fichiers existant
68     writer = csv.writer(fp, dialect='excel')
69     print(header_list[1:])
70     value_header = header_list[1]
71     offset = sum(c.isalpha() for c in value_header)
72     print(offset)
73     header_list[1:] = ['value']
74     writer.writerow(header_list)
75     for line in exported_data:
76         writer.writerow([line[kn] for kn in header_list])
77
78
79 ######## Cration des listes provenant des fichiers CSV ########
80 Co2, Bruit, Tv, temps,Freebox, Garage, Multimedia, Lumiere = [], [], [],
       [],[], [], [],[]
81 file = open("CO22.csv")
82 Title = file.readline()
83 for line in file:
84     line_token = line.split(" ")
85     temps.append(str(line_token[0]))
86     Co2.append(float(line_token[1]))
87 file.close()
88
89
90 file = open("bruit1.csv")
91 Title = file.readline()
92 for line in file:
93     line_token = line.split(" ")
94     Bruit.append(float(line_token[2]))
95 file.close()
96
97 file = open("Television_pad_new1.csv")
98 Title = file.readline()
99 for line in file:
100     line_token = line.split(" ")
101     Tv.append(float(line_token[2]))
102 file.close()
103
104 file = open("Freebox.csv")
105 Title = file.readline()
106 for line in file:
```

```
    line_token = line.split(" ")
    Freebox.append(float(line_token[2]))
file.close()

file = open("Lumiere.csv")
Title = file.readline()
for line in file:
    line_token = line.split(" ")
    Lumiere.append(float(line_token[2]))
file.close()

file = open("Garage.csv")
Title = file.readline()
for line in file:
    line_token = line.split(" ")
    Garage.append(float(line_token[2]))
file.close()

file = open("Multimedia.csv")
Title = file.readline()
for line in file:
    line_token = line.split(" ")
    Multimedia.append(float(line_token[2]))
file.close()

X, X1 = [], []

# construction features qui vont servir    creer notre mod le
for i in range(len(Tv)):

    X1.append(Bruit[i])
    X1.append(Co2[i])
    X1.append(Tv[i])
    X1.append(Lumiere[i])
    X.append(X1)
    X1 = []

clustering = KMeans(n_clusters=2, init='k-means++', n_init=10, max_iter
   =1000,
                    tol=0.0001, verbose=0, random_state=1, copy_x=True,
   algorithm='auto')
clustering.fit(X)

clustering_predict = clustering.predict(X)
######## d finition des fonctions utiles #######


##pour les pertes  nergtiques  ##
def PertesPrec (i, Multimedia,Freebox, Garage, compteur_t, abscence):
    indic=0
    pertes=0
    if compteur_t == abscence :
        indic = i-abscence
        while indic<=i:
            pertes += Multimedia [indic]
            pertes += Freebox [indic]
            pertes += Garage [indic]
            indic += 1
```

```
            print (pertes)
    return pertes

##Pour la consommation##
def consommation(ConsommationWatt, temps):
    consommation = temps * 30 * ConsommationWatt / 1000
    return consommation

##Cration d'un camembert pour la rpartition  nergtique ##
def camembert(consommationgarage,consommationmultimedia,consommationfreebox):
    labels = 'Garage', 'Multimedia', 'Freebox'
    sizes = [consommationgarage, consommationmultimedia,consommationfreebox]
    colors = ['yellowgreen', 'gold', 'lightskyblue']

    plt.pie(sizes, labels=labels, colors=colors,
            autopct='%1.1f%%', shadow=True, startangle=90)

    plt.axis('equal')
    plt.title("Perte en kWh")
    plt.legend()


    plt.show()

## Calcul du co t ##
def economie(consommationtotale):
    a=consommationtotale*0.1740
    print("Argent perdu :",a,"   " )
    return a

## Calcul du bilan carbone ##
def carbone(consommationtotale):
    gaz = (consommationtotale)*0.420
    nucleaire = consommationtotale*0.12
    print(gaz,"kg de CO2 provenant du gaz rejet s dans l'atmosph re pour rien\n")
    print(nucleaire,"kg de CO2 provenant du nucl aire rejet s dans l'atmosph re pour rien\n")

######## Dtermination des pertes  nergtiques  ########

abscence = 10 ## Correspond  notre seuil de temps. Ici 10 pour 5 heures car c'est compt  en demi heures
compteur_temps=0
compteur_energie=0
tempstot=0
compteurtot=0
rapport=0
a, b = [], []

for i in range(len(Tv)):
    compteurtot += Multimedia[i]
    compteurtot += Freebox[i]
    compteurtot += Garage[i]
    if(clustering_predict[i] == 0):
        compteur_temps += 1
```

```
        tempstot += 1
        print(compteur_temps)
        if(compteur_temps >= abscence):

            compteur_energie += Multimedia[i]
            compteur_energie += Freebox[i]
            compteur_energie += Garage[i]
            compteur_energie += PertesPrec(i, Multimedia,Freebox, Garage,
   compteur_temps, abscence)
            a.append(compteur_energie)
            b.append(a)
            a=[]



            print(compteur_energie)
    else:
        compteur_temps=0
        indic=0
rapport= compteur_energie/compteurtot

print ("Puissance totale perdue en W :\n",compteur_energie)
print ("Puissance totale consommee en 1 mois\n", compteurtot)
print ("Pourcentage de la puissance perdue par celle totale\n", rapport)
######## Determination heure de presence dans l'intervale de temps ########
c=0
for i in range(len(clustering_predict)):
    if clustering_predict[i]==0:
        c+=1
c = c/2
print("Heures d'abscence : ",c)
print("\n")

######## Determination heure de fonctionnement des appareils ########

tmultimedia = 0
tfreebox = 0
tgarage = 0
moyennegarage = 0
moyennemultimedia = 0
moyennefreebox = 0

for i in range(len(Garage)):
    if Garage[i] != 0:
        tgarage+=1
    if Freebox[i] != 0:
        tfreebox += 1
    if Multimedia[i] != 0:
        tmultimedia += 1

    moyennegarage += Garage[i]
    moyennefreebox += Freebox[i]
    moyennemultimedia += Multimedia[i]


######### Utilisation des fonctions d finis avant ########
tgarage = tgarage/60
tfreebox = tfreebox/60
```

```
tmultimedia = tmultimedia/60
moyennefreebox = moyennefreebox/len(Freebox)
moyennemultimedia = moyennemultimedia/len(Multimedia)
moyennegarage = moyennegarage/len(Garage)

consommationgarage = consommation(moyennegarage, tgarage)
consommationmultimedia = consommation(moyennemultimedia, tmultimedia)
consommationfreebox = consommation(moyennefreebox,tfreebox)
consommationtotale=consommationgarage+consommationfreebox+
    consommationmultimedia


camembert(consommationgarage,consommationmultimedia,consommationfreebox)

economie(consommationtotale)

carbone(consommationtotale)


######## Affichage de nos r sultats du clustering ########
intervalle_cluster, intervalle_temps = [],[]
for i in range(700,1050):
    intervalle_cluster.append(clustering_predict[i])
    intervalle_temps.append(temps[i])

plt.plot(intervalle_temps, intervalle_cluster, label = "06/05/2022->
    13/05/2022")
plt.title("Clustering prediction")
plt.ylabel("Occupancy")
plt.xlabel("Time")
plt.legend()
plt.show()
```

8 Sources

https ://www.researchgate.net/profile/Absalom-Ezugwu/publication/344590665/figure/fig1/AS :945789706702848@1602505246144/
example-with-intra-and-inter-clustering-illustrations.png

https ://tel.archives-ouvertes.fr/tel-00801555v2/document

https ://scikit-learn.org/stable/modules/clustering.htmloverview-of-clustering-methods

https ://particuliers.engie.fr/electricite/conseils-electricite/conseils-tarifs-electricite/comment-calculer-consommation-electrique.html

https ://youmatter.world/fr/co2-kwh-electricite-france-mix-electrique/