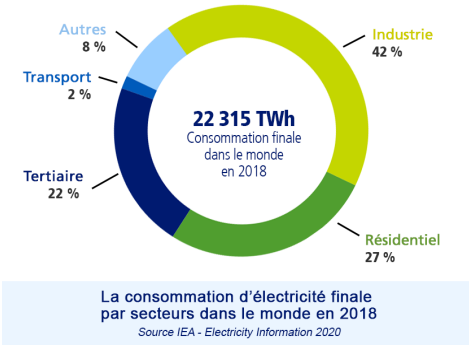
Charlotte Jacqueline & Adrien Claverie Mai 2022

Gestion d'énergie : Smart house



La consommation d'électricité finale par secteurs dans le monde en 2018

Source IEA - Electricity Information 2020

Constat :

D'après EDF, le secteur résidentiel représente **27%** de la consommation énergétique mondiale. Chercher à minimiser cet impact est donc un enjeu primordial pour demain. A l'ère de l'éveil des consciences sur la précarité énergétique qui accentue les inégalités, œuvrer pour l'isolation et une gestion responsable de l'énergie est une priorité.

Objectif :

On souhaite s'intéresser à l'étude de l'isolation d'une maison connectée en comparant la température extérieure et celle intérieure puis de mettre en parallèle ce gap de température avec la consommation énergétique due au chauffage.

On cherchera à montrer qu'une bonne isolation est clé pour réduire son impact énergétique et s'approcher de la sobriété énergétique.

Comment allons-nous procéder?

Nous apprenons dans ce bureau d'étude à récupérer des données et à les exploiter. Le but est d'optimiser l'utilisation des données pour gérer de manière efficace l'énergie d'une habitation.

A l'aide du projet Open Source Expé SmartHouse et de Spyder (Python), nous allons exploiter les données réelles d'une habitation (maison de 120 m^2 équipée d'une centaine de capteurs).

Grâce aux différences de température observées entre le rez-de-chaussée et l'extérieur et l'utilisation du système de chauffage, on pourra déduire la qualité de l'isolation thermique mais aussi de savoir si le chauffage est utilisé à bon escient ou non et comment améliorer cette consommation grâce à l'analyse des données.

I. Récupération des données

A l'aide d'un script Python nous convertissons des données brutes en fichier .csv que nous allons pouvoir exploiter. Pour éviter une perte d'informations par discrétisation des données, on utilise la fonction timestamp de Python en convertissant nos dates de début et de fin d'étude en **timestamp** (Epoch converter). Il est important de bien réaliser cette étape. En effet, si les dates de début et de fin de mesure ne sont pas identiques pour toutes les données, on ne pourra pas les comparer. Nous choisissons d'étudier les données du 01/02 au 28/02. En effet, au mois de février le chauffage est utilisé et on observe sur l'ensemble des données une semaine de vacances dans le mois qui nous permettra d'étudier la maison dans un contexte différent.

II. Traitement des données

Maintenant que nos données sont en format .csv nous faisons face à un autre souci: nous travaillons avec des données réelles qui présentent des défauts, des irrégularités, des trous dans les données etcetera. Pour pouvoir exploiter ces données il faut donc les traiter pour rendre la comparaison cohérente. Pour cela, nous avons dû, par exemple, ajouter des données à des endroits où il n'y en avait pas ou encore ajuster l'échelle des temps pour les différents capteurs pour avoir une base de temps commune pour chaque capteur.

Obtenir une base de temps identique pour les trois fichiers de données (Bibliothèque Panda de Python) :

Les capteurs n'effectuent pas les mesures à la même fréquence. Pour pouvoir exploiter et comparer nos différentes données, il faut les échantillonner. L'objectif est d'obtenir des données sur une base de temps identique pour chaque capteur. Le code suivant (ceci est un exemple) permet ainsi d'échantillonner nos données toutes les 30 min par temps croissant. Il faut vérifier que l'étape précédente a bien permis d'avoir une période de mesure identique pour les trois données.

```
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
data = pd.read_csv('C02Salon.csv', sep=",")
data['time'] = pd.to_datetime(data['time'])
pd.to_datetime(data['time'], format='%Y-%m-%dT%H:%M:%S')
data.set_index('time', inplace=True)
data_new=data.resample('30T').sum()
```

Nous allons réaliser cette méthode de panda pour chacune de nos bases de données:

1) Température extérieure
2) Température rez-de-chaussée
3) Consommation énergétique du chauffage

Après cette étape, nous aurons donc trois fichiers ".csv" échantillonnés de la même manière.

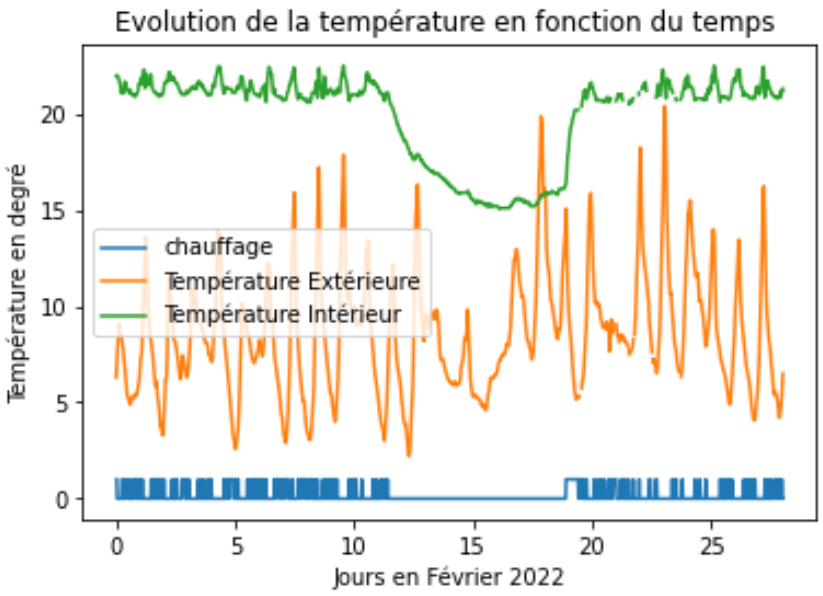
Nous avons un problème car les données de températures que nous obtenons ne correspondent pas du tout aux valeurs réelles. En effet la fonction sum additionne les valeurs sur la plage de temps ce qui explique que nous obtenions des valeurs trop élevées pour certaines plages de mesure alors que d'autres valent 0. Nous avons essayé de régler ce problème en utilisant les fonctions mean, pad et bfill de panda. En utilisant pad pour le chauffage (qui prend les valeurs binaires 0 ou 1/ éteint ou allumé) et mean pour les températures intérieures et extérieures, les données après sampling sont logiques et exploitables.

Regrouper les nouvelles données bien formatées dans des listes

Nous avons donc trois listes : Donnee_chau, Donnee_Text, Donnee_Trdc qui contiennent respectivement les valeurs de la consommation du chauffage, de la température extérieure et de la température du RDC toutes les 30 min tout au long du mois de février. Nous pouvons maintenant utiliser les différents outils mis à disposition par Python pour les lire.

Analyse graphique des données

Nous traçons les courbes du chauffage et des températures en fonction du temps. Nous pouvons discerner deux périodes.



<u>Période de vacances du 12 au 19 février</u> :

Sur cette période le chauffage est éteint donc nous allons pouvoir étudier directement le lien entre la température intérieure et celle extérieure sans influence de système de chauffage. Cela revient à étudier l'isolation.

➔ On observe entre le 11 et le 16 février une décroissance de la température intérieure suite à la fermeture du chauffage. On observe qu'il faut un peu plus d'un jour sans chauffage pour que la maison perde la chaleur qu'elle avait. Cela témoigne de la présence de pertes thermiques (mauvaise isolation au niveau des fenêtres, des portes, des murs ?).

➔ Après cette chute, la température intérieure se stabilise autour de 15 degrés malgré les variations de température extérieures entre le jour et la nuit. L'isolation est donc efficace sur ce point là car elle devient indépendante de la température extérieure.

➔ On observe que vers le 18 février, la température extérieure a un pic qui atteint 20°C alors que la température intérieure est inférieure à cette valeur. A la suite de cette singularité nous n'observons plus d'augmentation significative de la température intérieure. Il y a donc plusieurs possibilités :

- Soit le capteur extérieur a eu un problème et la température réelle n'était pas la température mesurée.

- Si cela n'est pas une erreur du capteur, cela signifie que la maison ne récupère aucune chaleur de l'extérieur. Cette singularité de comportement de la maison suggère que celle-ci cède uniquement de la chaleur à l'extérieur sans jamais en récupérer. En dehors d'une période de vacances, nous pourrions conseiller aux propriétaires d'ouvrir leurs fenêtres quand il fait beau dehors. Mais dans le contexte des vacances, les propriétaires ont probablement fermé les volets ce qui explique cette absence de récupération d'énergie.

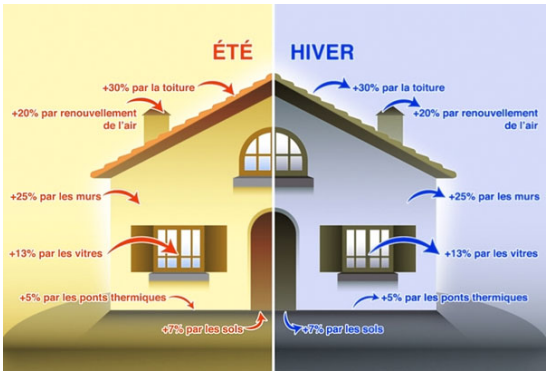
➔ Quand le chauffage est de nouveau allumé le 19 février, la température intérieure augmente rapidement. Elle atteint la température désirée en seulement quelques à la différence des 30 heures que la maison a mis pour céder toute son énergie donc le système de chauffage paraît donc efficace car la chaleur produite n'est pas perdue.

<u>En dehors de cette période</u> :

➔ La température intérieure varie peu malgré les fortes variations de la température extérieure, ce qui témoigne d'une bonne gestion du chauffage et d'une isolation satisfaisante.

Conclusion :

On peut donc déduire de ces premières observations que la maison cède de l'énergie à l'extérieur (perte de chaleur) sans jamais en récupérer. En effet, les températures extérieures hivernales ne permettent pas de gain de chaleur venant de l'extérieur. Cependant l'isolation paraît satisfaisante car la température dans la maison n'est pas trop influencée par les variations de la température extérieure.



Il aurait été intéressant d'étudier le comportement de la maison sur un mois estival où la température extérieure dépasse celle intérieure pour conclure sur la capacité de la maison à garder le frais afin d'éviter l'usage d'une climatisation.

Adaptation du code Python des premières séances pour faire de la prédiction

Nous allons maintenant commencer l'exploitation de nos données afin d'obtenir des Decision Tree dont nous déduirons des If/No rules. Pour fabriquer ces decision Tree, on utilise le site graphitz qui à partir d'un code obtenu avec les données, trace le decision Tree. Ces If/No rules permettront d'établir des conseils d'exploitation de la maison. Elles permettent de savoir la valeur de certains paramètres en fonction de la valeur des autres paramètres, par exemple dans notre cas on peut déterminer la température intérieure en fonction de la température extérieure et si le chauffage est allumé ou éteint.

Echec de l'implémentation d'un Decision Tree classifier :

Les données que nous avons sont continues et non discrètes comme dans la première partie du BE. Donc le Decision Tree classifier ne fonctionne pas avec ces données. Nous allons donc passer par la méthode de régression qui elle fonctionne avec des données linéaires.

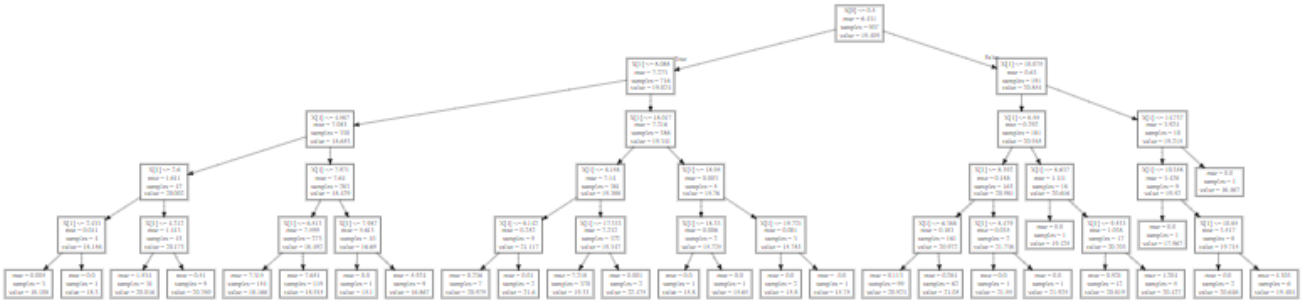
Fonction Regressor :

La fonction Regressor ne fonctionnait pas au début avec notre programme car les données que nous lui fournissons en entrée contiennent des valeurs "Nan" (pas de valeurs).

Nous avons donc remplacé les valeurs manquantes à l'aide de la fonction fillna() de pandas pour chacune des trois données (en remplaçant nan par une valeur moyenne adaptée à chaque donnée).

Analyse des arbres :

On commence avec une profondeur de 5 qui donne un arbre difficile à lire et à analyser. De plus, avoir autant de critères n'a pas beaucoup d'utilité car nous avons seulement trois bases de données à étudier (températures intérieure et extérieure et état du chauffage).



On réduit la profondeur de l'arbre à 3 pour plus de visibilité. Nous allons réaliser nos analyses sur cet arbre.

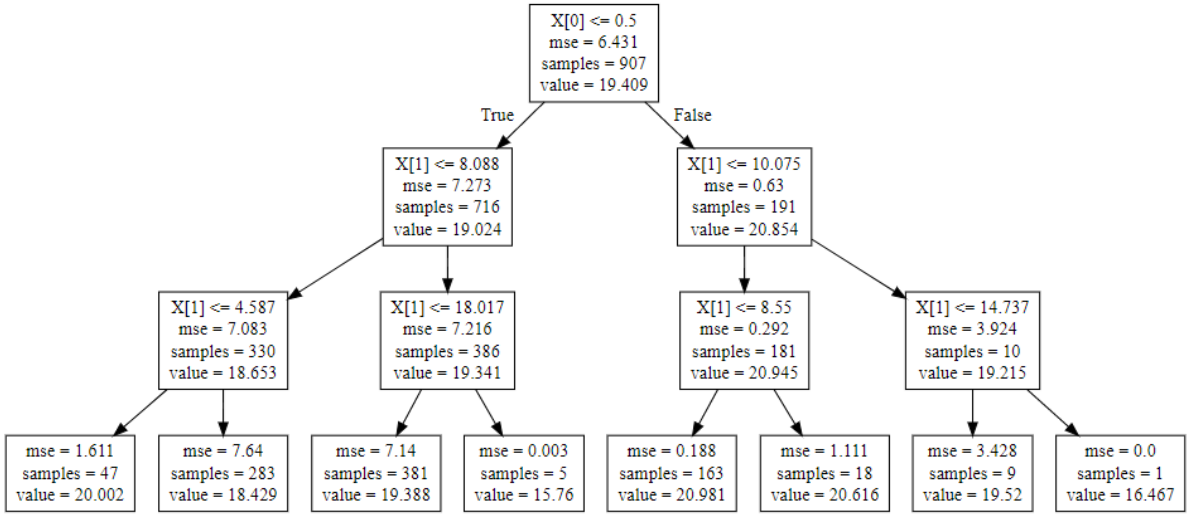


Tableau de correspondance :

X[0]	Etat du chauffage : - 0 si il est éteint - 1 si il est allumé
X[1]	Température extérieure
value	Température intérieure que l'on cherche à prédire
Samples	nombre d'échantillons appartenant au cas étudié sur cette période de temps

Observations :

La partie gauche de l'arbre représente le comportement de la maison quand le chauffage est éteint et celle de droite quand il est allumé.

Nous observons que la température intérieure varie entre 15,8 °C et 21°C mais que la majorité des échantillons sont situés entre 18,4°C et 21°C. Donc la température intérieure est assez stable et nous pouvons donc en déduire que le chauffage doit être assez bien réglé.

De plus, en observant la répartition des échantillons nous remarquons que le chauffage est assez peu utilisé (en prenant en compte qu'il y a une semaine de vacances).

Tableau d'analyse de l'arbre :

Observations/Conditions	Interprétation /Résultat
Chauffage éteint + 8°C<Text<18°C	Probabilité élevée d'avoir Tint = 19,4°C
Text < 8°C → Etat du chauffage =1	Chauffage bien utilisé en cas de besoin. Optimisation : réaliser un programme activant le chauffage en dessous d'une certaine valeur de la température extérieure. Cela permettrait d'éviter d'utiliser le chauffage les 10% du temps où il n'était pas utilisé à bon escient.

Conclusion :

Optimiser la gestion de l'énergie permet de réduire la consommation énergétique et donc de diminuer son empreinte environnementale. Cela permet aussi d'améliorer le confort des habitations.

Nous cherchions à étudier l'isolation d'une maison connectée et l'efficacité du système de chauffage à l'aide de l'exploitation de trois bases de données : températures intérieures et extérieures et état du chauffage.

De notre étude nous avons appris à traiter des données pour en tirer des conclusions sur l'isolation et la gestion du chauffage de l'habitation. Nous avons également appris à rebondir face à la confrontation à de nombreuses difficultés techniques de la récupération des données jusqu'à leur analyse en passant par l'implémentation d'un code Python.

Pour approfondir notre étude, nous pourrions réaliser une étude similaire sur un mois estival pour étudier la maison dans des conditions inverses. Etudier l'influence des ouvertures de portes et des fenêtres pourrait également être une prochaine étape d'étude.

Annexe : Code final

```python
# -*- coding: utf-8 -*-
"""
Created on Mon May 30 08:25:18 2022

@author: claverad
"""

import graphviz
import numpy as np
import random as rd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
import time
from sklearn.tree import DecisionTreeRegressor
import pandas as pd
from sklearn import tree
from sklearn.model_selection import train_test_split
from sklearn import metrics


data = pd.read_csv(
    'Z:/GESTION ENERGIE/S3_Python_Database/Chauffage.csv', sep=",")
print("jfjkjjkl",data.isnull().sum())
data.ffill()
data['time'] = pd.to_datetime(data['time'])
pd.to_datetime(data['time'], format='%Y-%m-%dT%H:%M:%S')
data.set_index('time', inplace=True)
data_new = data.resample('30T').pad()
data_new['value']=data_new['value'].fillna(0)

#Temps_chauffage= [data_new[:,0]]
# val_chauffage= [data_new[:,1]]


print((data_new))
Donnee_chau = data_new["value"]



data1 = pd.read_csv('Z:/GESTION ENERGIE/S3_Python_Database/Text1.csv', sep=",")
data1['time'] = pd.to_datetime(data1['time'])
pd.to_datetime(data1['time'], format='%Y-%m-%dT%H:%M:%S')
data1.set_index('time', inplace=True)
data_new1 = data1.resample('30T').mean()
data_new1['value']=data_new1['value'].fillna(7)
print((data_new1))
# Temps_chauffage= [data_new[:,0]]
# val_chauffage= [data_new[:,1]]

Donnee_Text = data_new1["value"]


data2 = pd.read_csv('Z:/GESTION ENERGIE/S3_Python_Database/Trdc1.csv', sep=",")
data2['time'] = pd.to_datetime(data2['time'])
pd.to_datetime(data2['time'], format='%Y-%m-%dT%H:%M:%S')
data2.set_index('time', inplace=True)
data_new2 = data2.resample('30T').mean()
```

```
58  data_new2['value']=data_new2['value'].fillna(21)
59  print((data_new2))
60  Donnee_Trdc = data_new2["value"]
61
62  temps=np.linspace(0,28,1296)
63  plt.plot(temps,Donnee_chau,label='chauffage')
64  plt.plot(temps,data_new1,label='Température Extérieure')
65  plt.plot(temps,data_new2,label='Température Intérieur')
66
67  plt.title('Evolution de la température en fonction du temps ')
68  plt.xlabel('Jours en Février 2022')
69  plt.ylabel('Température en degré')
70  plt.legend()
71  plt.show()
72  # Temps_chauffage= [data_new[:,0]]
73  # val_chauffage= [data_new[:,1]]
74
75
76
77  # for i in range(len(Donnee_Text)):
78  #     print(Donnee_Text[i])
79
80  feature = []
81
82  taille_training = int(0.7*len(Donnee_Trdc))
83
84  X = []
85  Y = []
86
87  for i in range(taille_training):
88      feature = [Donnee_chau[i], Donnee_Text[i]]
89      X.append(feature)
90      Y.append(Donnee_Trdc[i])
91
92  taille_tempRDC = len(Donnee_Trdc)
93  # print (taille_training)
94  # print (taille_tempRDC)
95  # Xnew=[]
96  # Xcluster=[]
97  # for i in range(taille_training, taille_tempRDC-1):
98  #     feature = [Donnee_chau[i], Donnee_Text[i]]
99  #     Xcluster.append((feature[1],feature[0]))
00  #     Xnew.append(feature)
01
02
03  # Ymod = classifier.predict(Xnew)
04
05  # Yref = []
06
07  # for i in range(taille_training, taille_tempRDC):
08  #     Yref.append(Donnee_Trdc[i])
09
10
11  regr = DecisionTreeRegressor(max_depth=3)
12  regr.fit(X, Y)
13  y = regr.predict(X)
14
```

```
    Yref.append(Donnee_Trdc[i])


regr = DecisionTreeRegressor(max_depth=3)
regr.fit(X, Y)
y = regr.predict(X)

with open("tree23.dot", 'w') as file:
    f = tree.export_graphviz(regr, out_file=file)

plt.plot(y, label="max_depth=5")
plt.xlabel("features")
plt.ylabel("tempRDC")
plt.show()

m = len(Yref)-1
vrai = 0
faux = 0


for i in range(m):
    if Yref[i]==Ymod[i]:
        vrai += 1
    else:
        faux+= 1
print ("Le modele donne le nb exact à :", ((vrai/m)*100,"%"))

error=[]
somme=0
erreur_moy=0

for i in range(m):
    if Yref[i]!=0:
        error_i=abs((Yref[i]-Ymod[i])/Yref[i])
    else:
        if Ymod[i]==0:
            error_i=0
        else:
            error_i=Ymod[i]

    error.append(error_i)
    somme+=error[i]
    erreur_moy=somme/m


print("L'erreur moyenne vaut: \n")
print(erreur_moy)

importance= classifier.feature_importances_
for i,v in enumerate(importance):
    print('Feature: %0d, Score: %.5f' % (i,v))

plt.title("Most important feature")
plt.bar([x for x in range(len(importance))], importance)
#plt.show()
```

```
154    print("L'erreur moyenne vaut: \n")
155    print(erreur_moy)
156
157    importance= classifier.feature_importances_
158    for i,v in enumerate(importance):
159        print('Feature: %0d, Score: %.5f' % (i,v))
160
161    plt.title("Most important feature")
162    plt.bar([x for x in range(len(importance))], importance)
163    #plt.show()
164
165
166    kmeans = KMeans(n_clusters=5, random_state=0).fit(X)
167
168    print(kmeans.labels_)
169    print(kmeans.cluster_centers_)
170
171    print(Xnew)
172    k=[]
173    error_ie=0
174    for i in range(0,216):
175        k.append(kmeans.predict([X[i]])[0])
176    if Yref[i]!=0:
177        error_ie = abs((Yref[i] - k[i]) / Yref[i])
178        if k[i]==0:
179            error_ie=0
180        else:
181            error_ie=k[i]
182    vrai2=0
183    faux2=0
184
185    for i in range(0,216):
186        if Yref[i]==k[i]:
187            vrai2 += 1
188        else:
189            faux2+= 1
190        error.append(error_ie)
191        somme += error[i]
192    erreur_moy = somme / m
193    print ("Le modele donne le nb exact à :", ((vrai2/216)*100,"%"))
194    print(k)
195
196    print(erreur_moy)
197
```