Bures Kyllian
Masson Pierre-Nicolas

Rapport de projet gestion énergie

Sommaire :

Introduction :

Lors de ce projet notre objectif est de réduire la consommation énergétique d'un foyer à l'aide d'informations sur divers capteurs. Pour cela nous souhaitons éteindre les appareils fonctionnant inutilement. Dans le but de réaliser cet objectif nous souhaitons construire un modèle qui permettra de déterminer la présence ou non d'individus Le nombre de ces personnes n'est pas essentiel contrairement à leur position dans la maison. Dans notre démarche nous commencerons par estimer si la maison est occupé en essayant de savoir où sont les gens. Par la suite, si nous avons le temps, nous voudrions déterminer les appareils fonctionnant inutilement pour pouvoir les éteindre si nécessaire.

Raisonnement :

La première étape était de décider quelles données nous allions utiliser. Notre objectif est de réduire la consommation en fonction de la présence ou non d'occupant. Pour cela nous ne pouvions pas utiliser les données liées à la consommation énergétique pour construire notre modèle de prédiction. De plus, les informations provenant de l'extérieur de la maison, tel que la vitesse du vent, ne peuvent nous aider à déterminer si la maison est occupée. Il nous a donc fallu des informations qui sont, le plus possible, due à l'occupation. Les informations correspondantes sont la quantité de CO2 et le bruit. Nous avons pensé à utiliser l'humidité, cependant son évolution et l'occupation semblent décorrélées. Pour utiliser la concentration de CO2 il nous faut des informations supplémentaires sur la position des portes et des fenêtres. En effet, si une fenêtre est ouverte la concentration ne sera plus exploitable, si une porte est ouverte la concentration des deux pièces tendra vers la même valeur. Enfin, le niveau de bruit peut être un bon indicateur de présence, notre seule information sur le bruit provient du salon. Néanmoins cette information peut vite être inexploitable. Par exemple une personne peut allumer la télé et partir sans l'éteindre. Il faudrait donc faire attention à la présence de source de bruits parasites quand nous utiliserons l'information du bruit.

Réduction de la consommation énergétique :

En connaissant la présence ou non d'une personne dans les différentes pièces nous pouvons agir pour réduire le gaspillage énergétique. Nos actions doivent être réalisées dans les pièces non occupées. Nous souhaitons laisser quelques instants entre le moment où nous ne détectons plus personne et notre première action de réduction de gaspillage. Dans un cas pratique si une personne s'absente d'une pièce pour un bref instant (pas plus de 5minutes) elle ne souhaite pas devoir à rallumer ses différents appareils à son retour. De plus notre réflexion se basant principalement sur le CO2 nous avons besoin d'un certain temps pour que l'on observe une variation de concentration. Nos actions de réduction de gaspillage nécessitent de contrôler les appareils en fonctionnement dans les pièces non occupés. Nous pourrions contrôler des éléments basiques tels que l'éclairage de la pièce étudiée. Dans une réflexion plus large nous pourrions éteindre les plaques de cuissons si nous ne détectons personne dans la maison. Notre modèle pourrait aussi servir de dispositif de sécurité dans le cas d'une maison vide où l'on détecte une porte ouverte mais nous nous éloignons de notre
but de base.

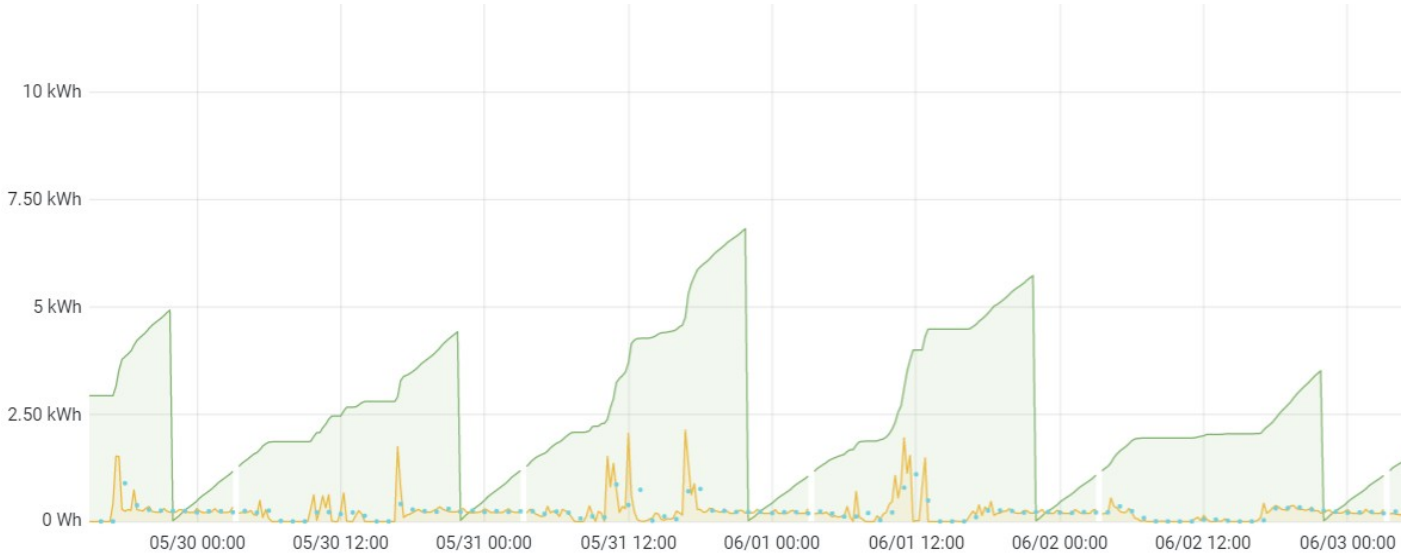

Figure 1 : Exemple de graphique de consommation d'énergie

Mise en place du modèle :

Pour réaliser notre modèle nous avons défini un ensemble de données qui nous servent de référence pour établir notre modèle. Malheureusement nous ne savions pas le nombre de personnes associées aux différents échantillons en notre possession. Nous avons donc arbitrairement défini que la présence de l'arrivé ou le départ d'occupant entraînait une variation importante du niveau de CO2. Ainsi nous avons analysé manuellement nos données d'entraînement pour leur associées une présence d'occupant ou non. Puisque le nombre d'occupant ne nous intéresse pas nous affectons uniquement des 0 et des 1.

CO2_salon (1).csv - Libre...

	A	B
1	time	Valeur
2	2022-05-01T00:14:33Z	467
3	2022-05-01T00:24:38Z	447
4	2022-05-01T00:44:48Z	447
5	2022-05-01T00:54:53Z	447
6	2022-05-01T01:15:05Z	438

Bruit_salon (1).csv - LibreOffice Calc

	A	B	C
1	time	value	
2	2022-05-01T00:14:33Z	37	
3	2022-05-01T00:24:38Z	37	
4	2022-05-01T00:44:48Z	37	
5	2022-05-01T00:54:53Z	37	
6	2022-05-01T01:15:05Z	37	

Figure 2 : Tableurs des données étudiées

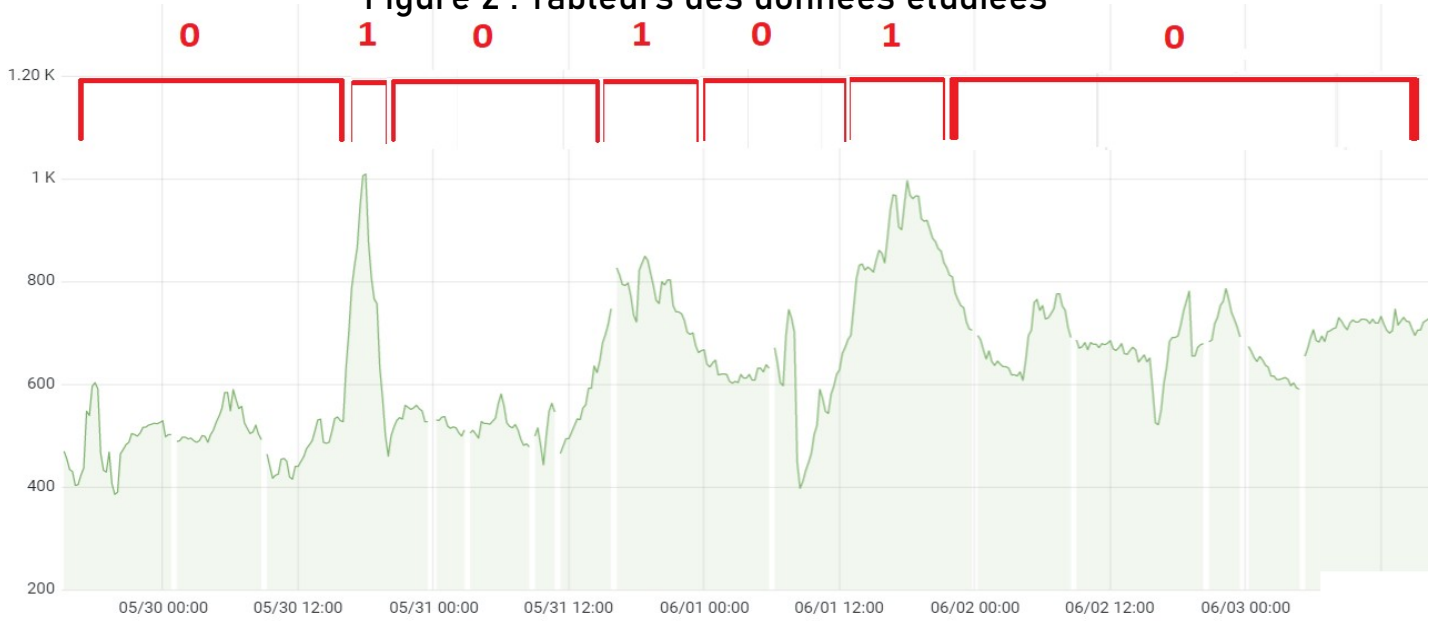

Figure 3 : Estimation de l'occupation réelle à partir du graphe du CO_2

Nous avons ensuite utilisé ces résultats pour configurer un arbre de décision à l'aide de la fonction RandomForestClassifier qui est une méthode par classification du module sklearn. Cela nous a permis d'obtenir l'arbre suivant :

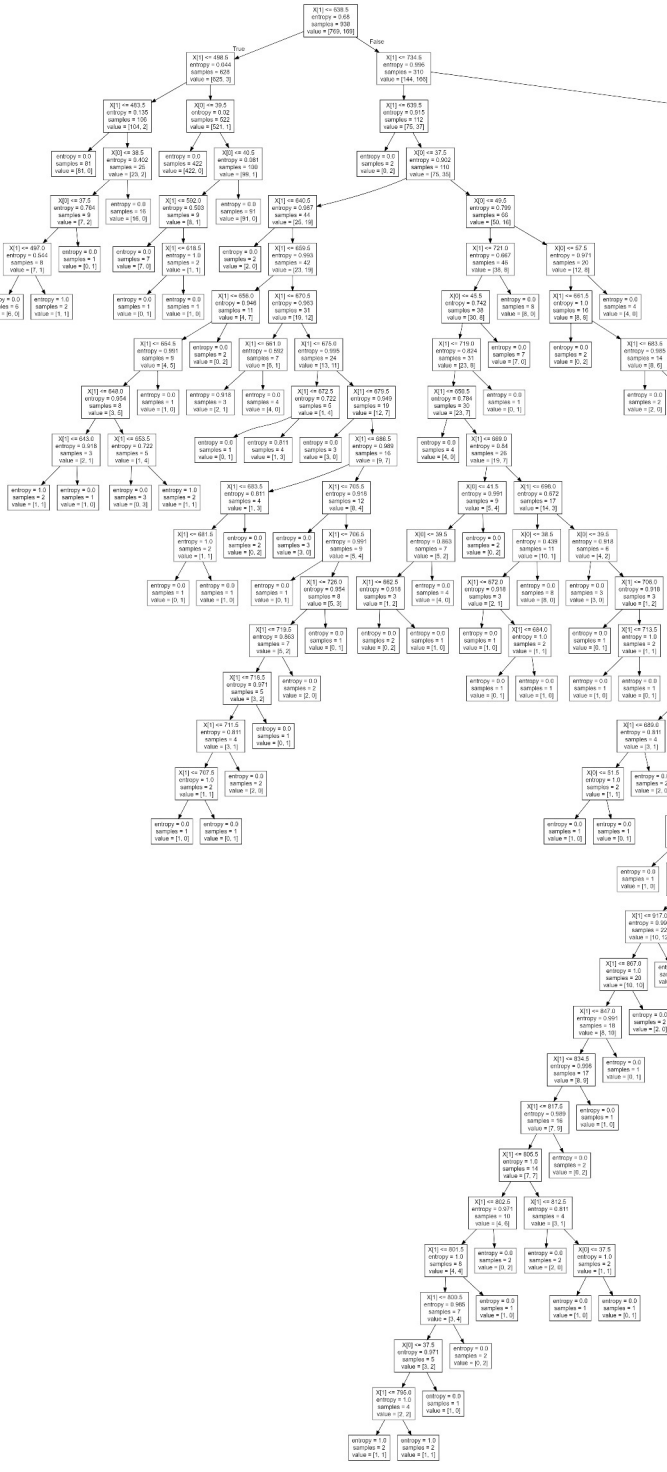

Figure 4 : Arbre binaire décisionnel issue de la fonction RandomForestClassifier

Résultats :

Les résultats d'estimation d'occupation que nous obtenons sont proche de ce que nous aurions choisi manuellement cependant il y a certaines différences (efficacité de 85 %) . En effet, nous pouvons constater de brèves présences lorsqu'il n y a personne. Ces fluctuations rapides ne posent que peu de problème car au préalable personne n'a été détecté donc les appareils inutiles sont éteints. Détecter une présence alors qu'il n'y en a aucune ne pose plus de problème car personne n'est là pour allumer un objet que l'on doit contrôler. Enfin notre modèle revient rapidement à une détection nulle.
Le réel problème est le nombre de fluctuations rapides quand nous avons des occupants. En effet, cela a pour conséquence d'éteindre certains appareils qui pourraient être utilisés. Nous avons réfléchis à des solutions pouvant pallier à ce problème. Notre première idée était de faire une moyenne cependant cette piste nous semble complexe voire impossible. Une autre possibilité est de bloquer un état haut si nous avons des fluctuations nombreuses

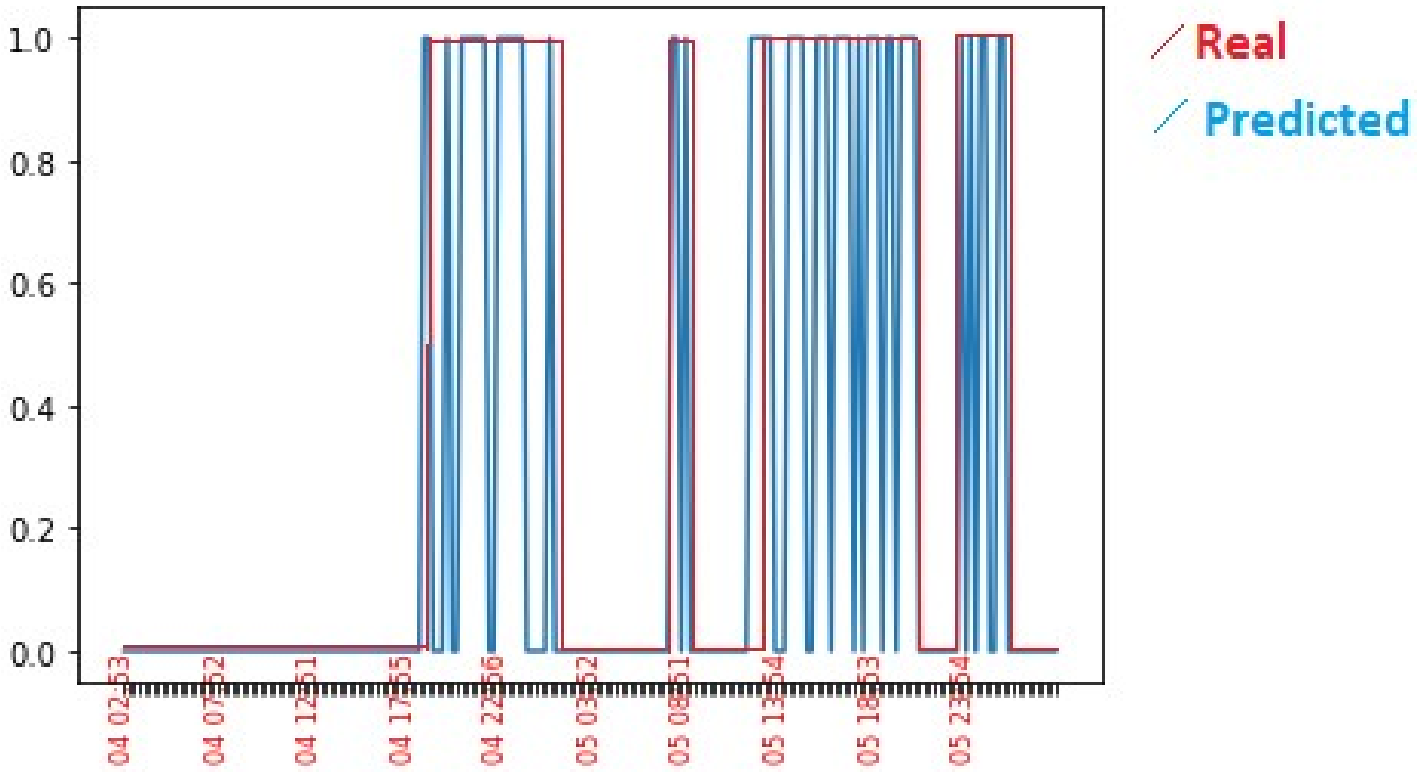

Figure 5 : Différence entre les prédictions du modèle et la réalité sur un échantillon

Conclusion :

Notre première approche du problème avec notre modèle n'est pas parfaite mais satisfait certaines de nos attentes. Nous pouvons estimer la présence d'occupant, malheureusement nous manquons de précision. Une façon de nous améliorer serait de prendre une période plus importante pour nos données d'entraînement Nous pourrions aussi changer notre méthode pour estimer manuellement s'il y a des occupants. Actuellement notre modèle ne comporte pas encore les informations des portes et des fenêtres, nous pourrions donc améliorer notre programme en incorporant ces données. Enfin notre études c'est limitées au salon, il serait donc possible d'essayer d'adapter notre modèle pour les différentes pièces. Pour continuer ce projet la priorité est d'améliorer la précision puis nous pourrons l'étendre au reste de la maison.

ANNEXES code Python

```
from sklearn import tree
import matplotlib.pyplot as plt
import numpy as np
from sklearn.cluster import KMeans

from sklearn import tree
import matplotlib.pyplot as plt
import numpy as np
from sklearn.ensemble import RandomForestClassifier

result_bruit=[] #Liste des valeurs de bruit pour la construction du modèle
result_co2=[] #Liste des valeurs de la concentration de CO2 pour la construction du modèle
result=[] #Liste des valeurs croisés de bruit et de CO2 pour la construction du modèle
label=[] #Resultats réels pour la construction du modèle
result2_bruit=[] #Liste des valeurs de bruit pour les tests
result2_co2=[] #Liste des valeurs de la concentration de CO2 pour les tests
result2=[] #Liste des valeurs croisés de bruit et de CO2 pour les tests
value=[] #Liste des résultats des tests
date=[] #Liste des dates correspondants aux échantillons de test
absc=[] #Liste d'affichage de l'axe des abscisses

#Remplissage de la liste result_bruit
with open("bruit_salon.csv", 'r') as fichier:
    lignes = fichier.readlines()[1155:]
    for ligne in lignes:
        ligne = ligne.split("\n")[0]
        ligne_token=ligne.split(";")
        result_bruit.append(ligne_token[1])

#Remplissage de la liste result_co2
with open("Co2_salonv.csv", 'r') as fichier:
    lignes = fichier.readlines()[1155:]
    for ligne in lignes:
        ligne = ligne.split("\n")[0]
        ligne_token=ligne.split(";")
        result_co2.append(ligne_token[1])

#Remplissage de la liste result
for i in range(len(result_bruit)):
   result.append([result_bruit[i],result_co2[i]])

#Remplissage de la liste label
for i in range(len(result_bruit)):
   i+=1155
   if i<1544:
      label.append(0)
   elif i<1550:
      label.append(1)
   elif i<1583:
      label.append(0)
   elif i<1607:
      label.append(1)
   elif i<1642:
      label.append(0)
   elif i<1644:
      label.append(1)
   elif i<1653:
      label.append(0)
   elif i<1655:
      label.append(1)
elif i<1677:
      label.append(0)
```

```
    elif i<1741:
        label.append(1)
    elif i<1764:
        label.append(0)
    elif i<1806:
        label.append(1)
    elif i<1854:
        label.append(0)
    elif i<1857:
        label.append(1)
    elif i<2063:
        label.append(0)
    elif i<2080:
        label.append(1)
    elif i<2084:
        label.append(0)
    else:
        label.append(1)

# Construction de l'arbre décisionel
classifier = RandomForestClassifier(n_estimators=40,random_state=0, criterion='entropy',max_depth=None)
classifier.fit(result, label)

# Définition de l'intervalle de test
valeur_test_debut=300
valeur_test_fin=500

#Remplissage de la liste result2_bruit
with open("bruit_salon.csv", 'r') as fichier:
    lignes = fichier.readlines()[valeur_test_debut:valeur_test_fin]
    for ligne in lignes:
        ligne = ligne.split("\n")[0]
        ligne_token=ligne.split(";")
        result2_bruit.append(ligne_token[1])

#Remplissage de la liste result2_co2
with open("Co2_salonv.csv", 'r') as fichier:
    lignes = fichier.readlines()[valeur_test_debut:valeur_test_fin]
    for ligne in lignes:
        ligne = ligne.split("\n")[0]
        ligne_token=ligne.split(";")
        result2_co2.append(ligne_token[1])
        #Remplissage de la liste date
        date.append(ligne.split("T")[0][-2:]+' '+ligne.split("\n")[0].split("T")[1][:5])

#Remplissage de la liste result2
for i in range(len(result2_bruit)):
    result2.append([result2_bruit[i],result2_co2[i]])

#Remplissage de la liste value
for i in range(len(result2_bruit)):
    value.append(float(classifier.predict([result2[i]])))

#Remplissage de la liste absc
for i in range(len(date)):
    if i%(int((valeur_test_fin-valeur_test_debut)/10))==0:
        absc.append(date[i])
    else:
        absc.append(' ')

plt.plot(date,value)
#formatage de l'abscisse du graphe
plt.gca().xaxis.axes.xaxis.set_ticklabels(absc, rotation = 90, color = 'red', fontsize = 8, verticalalignment = 'center')
```