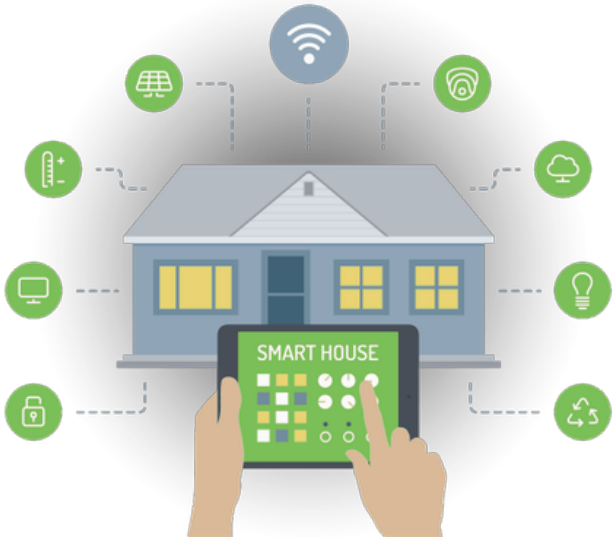
Data Analysis and Smart House



OTT Daniel JENFT Clara GAUBERT Ylann

abstract

In this article we will look at the processing of CSV data from the various sensors installed in a smart home. To do this we will first extract the data, format it according to what we are able to exploit and then gather it in a single file. Once this step is done, we will launch into the exploitation of this data. This will be done in three stages. Firstly, we sought to determine the occupancy in the room under consideration. Unfortunately we did not succeed in this step despite a lot of research and a strong involvement. Nevertheless, this allowed us to discover the K-means function of Python in order to carry out a clustering work.The second step consisted in determining, with a simple function, the energy losses when the window is opened to ventilate the room. This study allows a better understanding of the energy losses that one might think are marginal but that it is interesting to eliminate all the same. Finally, the last function, which was very different, was to determine when to switch off the box, so that it does not remain on standby when not in use. Indeed, as we will explain later, a box is highly energy consuming. The implementation of this function was also not successful because the function, using many loops and many transient data storage and processing, was very complex. Nevertheless, the manipulation of this function was also very instructive. We will therefore present these three functions in the following work and the conclusions we reached.

Summary

I - Background

As part of our formation at the ENSE3, we chose the project “Gestion d'Énergie" supervised by Jérôme Ferrari and Manar Aymari. In this course we worked on a mini-project which consisted in analysing data from real sensors installed on a real smart house. The Data used was drawn from expe smarthouse web site.
We used Grafana to visualise it and python to process.

Our aim was to develop a method to determine when the users have wasted energy during the last 40 days in order to set an automatic alert.

This rapport will first present and explain our program and then analyse, as deeply as we can, our results.

II - programme

In this report we will only present our own work, even if it is based on programs and functions written by Jérôme Ferrari. These last will be available in the annex.

The Goal of this project, as mentioned before, is to predict whether or not energy was wasted, and if so at what point in time. To this end, we want to compare the data on if windows were opened, the Co2 content of the air in the room, and the heating units activity. If the window was opened while the heater was on, and while the Co2 concentration did not exceed 1000ppm, our program would conclude that energy has been wasted. The following paragraphs describe the steps we took to achieve this.

Our program is divided into 3 parts :

- Reading data from csv files , filling gaps and resample it in order to get exploitable datas with as little loss of information as possible.
- Concatenate data from each sensor into only one dataframe.
- Finally, process data to determine whether users are wasting energy or not

1) From .csv files to exploitables datas

Datas on expe smarthouse are imported thanks to Jérôme Ferrari's program. We use IDs to select datas we want to analyse and import it one by one. Datetime strings in the .csv files are not written with the good syntaxe to be read by pandas functions. To modify it, we implemented a function which deletes the separators characters. And put in format 'datetime' from module datetime.

```
def stringdate_to_datetime(stringdatetime: str):
    if "." in stringdatetime:
        stringdatetime=stringdatetime.split(".")[0]

    else :
        stringdatetime = stringdatetime.split("Z")[0]

    return
datetime.datetime.fromtimestamp(time.mktime(time.strptime(stri
ngdatetime, '%Y-%m-%dT%H:%M:%S')))
```

Then, we defined a complete function `lire_csv(filename)` which:

- read one csv file, create a dataframe with the corresponding datas and apply the correct date-time format:

```
selected_attribute = pandas.read_csv(filename, sep=",")
selected_attribute["datetime"] =
selected_attribute["time"].apply(stringdate_to_datetime)
sa = selected_attribute[['value', 'datetime']]
sa.set_index('datetime', inplace=True)
```

- fill the gaps with the function (we were also experimenting with interpolation, but the differences were insignificant): `ffill()`

```
sa.ffill()
```

- resample each dataframe with the same frequency : We chose 40 minutes.

```
sa = sa.resample('40T').sum()
```

Finally, the function returns a dataframe with 1 column containing the filled end resampled data of one sensor.

```
return sa
```

We can now read data from each sensor :

```
Tv_power = lire_csv("TV_power.csv")
fenetre= lire_csv("fenetre.csv")
humidity = lire_csv("humidity.csv")
chauffage= lire_csv("chauffage.csv")
lumiere= lire_csv("lumiere.csv")
temperature=lire_csv("temperature.csv")
C02=lire_csv("CO2.csv")
```

To facilitate the processing, we can gather each dataframe into an only one:

```
result=pandas.concat([humidity,C02,fenetre,chauffage,temperatu
re,lumiere,Tv_power],axis=1,join='inner')
```

and add the index corresponding to each sensor:

```
result.columns=["humidity","C02","fenetre","chauffage","temper
ature","lumiere","Tv_power"]
```

2) Processing

We can now process the data.

a) determination of occupancy

The first part of our code had to consist of an efficient determination of the room occupancy. However, unlike the previous exercise, we did not have a label to train decisionTree algorithm. We therefore had to use an unsupervised learning method. For this, following a lot of research, we turned to the K-means function. Indeed, this Clustering technique seemed to us to be adapted to our problem. On the one hand, it allowed us to create groups. Our idea was therefore to create two groups: one corresponding to the occupied room, the other to the unoccupied room. On the other hand, this function seemed to be rather simple to use and to be able to manage lines with a lot of information. Namely: the date, as well as the set of sensors.
Nevertheless we did not manage to implement this function. Indeed, if we managed to draw groups for one sensor according to another, we did not manage to gather all the information. So we think that we didn't really grasp the subtleties of this function. Nevertheless, we spent many hours trying to figure it out. If this did not allow us to obtain a result, it allowed us to learn more about the different methods of unsupervised learning as well as about all the functions proposed by piton. Moreover, it was very interesting to explore this aspect of machine learning which seems obscure when we have not yet been confronted with it.
So if this part did not bear fruit, our attempts were not completely in vain.

b) waste of energy correlated with windows opening

We decide, to simplify, to define the waste condition with the following criteria :

- CO2 concentration over 1000 ppm
- open window (corresponding to a boolean equal to 1)
- Heating with power on (= 1)

Therefore, processing is quiet easy with a if function with 3 conditions:

```
def waste (data):
    w=np.zeros(len(data["C02"]))
    for n in range (len(data["C02"])):
        if data["C02"][n]>1000 and data["fenetre"][n]==1 and
data["chauffage"][n]>0 :
            w[n]=1
    return w
```

We can ad the waste label to our big dataframe :

```
Waste_of_energy=waste(result)

result['waste']=Waste_of_energy
```

c) box waste: a function to determine when the box should be turned off automatically.

An underestimated aspect of high energy expenditure is undoubtedly the power consumption of digital technology. This includes the huge data centres and the information transport network. But it also concerns the terminals we have at home. It is impressive to learn that a box consumes about as much power as a refrigerator. So, according to ADEM, all the boxes in French homes consume 1% of the electrical resources in France. This exorbitant figure should therefore be taken into consideration in our search for solutions to reduce our energy impact.
It is with this in mind that we have chosen to work on the box of this house.
Firstly, the analysis of the data allowed us to observe that the box consumes continuously about 50% of what it consumes when it is running at full speed. So we thought that we could optimise this by switching off the box when it was not in use. To do this, we had to create a function that takes the file corresponding to the box as input. Then we chose to normalise the data. That is, when the consumption was higher than 17.7 we added a row to the table equal 1. When the consumption was lower than 17.7, this value was set to zero.
Then we processed each day of each month to find for each day the hours of switching on and off of the time slots of use of the box. To do this we had to smooth the data as it was jagged. Then we stored them in a global list by couple: first value on time, second value off time. The idea was to group similar switch-on times together and then create an average between these values. In order to obtain several time slots during the day when the box was used. Then, it is enough to indicate that the box should be switched on half an hour before the beginning of the "classic use" range and then switched off half an hour after the end of this range. However, we did not manage to carry out these last steps (starting with the setting of the values, as the function was very complex for us). Nevertheless, we spent several more hours in order to succeed in this challenge.

III - Analysis

In this work, we were not able to extract much data for analysis. Indeed, while the program to determine the energy loss when opening the window gave us some results, the other two did not. Nevertheless, we could see that the users of this house lost very little energy when opening the window, because out of all the samples we had, there were only three moments when energy was lost according to our criteria. The other two functions, although they did not give us any results, allowed us to manipulate the various features of Python. This allowed us to realise that this tool was not very complicated to use in its basic functions but that it was nevertheless very powerful. In particular in the implementation of machine learning.

IV - Work methods

In the first part of the course we each worked alone on learning the basic principles we later applied to our project. On the basis of a dataset provided by Mme Aymari (which had already been preprocessed), we each wrote a program to estimate the occupancy of a room based on the aforementioned sensory data. To this end, we implemented a decision tree, which we later on experimented on, varying certain parameters and also implementing some cost optimization considerations.

In the second part of the course we worked as a team on the project, as described above. Originally, we were going to use git for version control, but it turned out to be to time intensive to initiate these measures during the course (in the future, this might be worth adding into the course, as it enables better teamwork and therefore would surely improve the learning experience as well as the results of this project).

During the project, we repeatedly came upon problems, beginning with the correct reading of the data, its preprocessing, and finally the actual focus, drawing valuable conclusions from the available data. We managed to resolve these problems, partially during the class under the supervision of the teachers.

We are adding to the project some more functionality after the end of classes, in order to achieve the original goal of actually implementing useful functionality,

by estimating the occupancy of a room, and deciding based on the opening of windows, the co2 concentration in the room and the level of heating currently applied, if energy is being wasted.

To conclude, we learned the basics of the aforementioned machine learning algorithms as well as working on reading in data into python, which certainly is a valuable skill set. We draw a positive verdict on our work in this project, and hope that you will find the results satisfactory.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import tree
from sklearn import metrics
import pandas
from sklearn.model_selection import train_test_split
import graphviz
import time
import datetime
from sklearn.cluster import KMeans

""""""""""""""""""" to read and put in form datas """""""""""""""""


def stringdate_to_datetime(stringdatetime: str):
    if "." in stringdatetime:
        stringdatetime = stringdatetime.split(".")[0]

    else:
        stringdatetime = stringdatetime.split("Z")[0]

    return
datetime.datetime.fromtimestamp(time.mktime(time.strptime(stringdatetime,
'%Y-%m-%dT%H:%M:%S')))


# lecture csv

def lire_csv(filename):
    # read csv and convert to data frame
    selected_attribute = pandas.read_csv(filename, sep=",")
    selected_attribute["datetime"] = selected_attribute["time"].apply(
        stringdate_to_datetime)
```

```
    sa = selected_attribute[['value', 'datetime']]
    sa.set_index('datetime', inplace=True)
    # fill gaps
    print("isnull", sa.isnull().any())
    sa.ffill()  # interpolate pour prendre une valeur moyenne entre la valeur prec
et suivante
    #  resample data
    sa = sa.resample('40T').sum()
    return sa


Tv_power = lire_csv("TV_power.csv")
fenetre = lire_csv("fenetre.csv")
humidity = lire_csv("humidity.csv")
chauffage = lire_csv("chauffage.csv")
lumiere = lire_csv("lumiere.csv")
temperature = lire_csv("temperature.csv")
C02 = lire_csv("CO2.csv")
box = lire_csv("box_tv.csv")
bruit = lire_csv("bruit.csv")

result = pandas.concat([humidity, C02, fenetre, chauffage,
                temperature, lumiere, Tv_power], axis=1, join='inner')
result.columns = ["humidity", "C02", "fenetre",
            "chauffage", "temperature", "lumiere", "Tv_power"]

# print("C02")
# print (result["C02"])

# print("\nfenetre")
# print (result["fenetre"])

# print("\nchauffage")
# print (result["chauffage"])


data = result.copy

print(data)


"""""""""""""""""""""""""""""""""""""""""""""""


""""""""""""""""""""""""""""determin occupancy whith KMeans"""""""""""""""""""""
```

```
# def nb_collums (data):
#     nb = len (data.axes[1])
#     return (nb)


# # def normalize(data) :

# #     for columns in nb_collums(data):
# #         title
# #         for line in len(data):
# #             x_norm =
((data[data.axes[1][columns]][line])-x_min)/(x_max-x_min)
# #             data_norm.append(x_norm)


# # def determine_occupation (data) :

# #     #distance Entre les différents points de données.
# #     #On utilisera la distance euclidienne sur les données normalisé

# #     #Normalisation des données

# #     for collums in nb_collums(data):#Nombre entre 0 et 9
# #         for line in len (data) :#Nombre entre 0 et 600 et quelques


# column_x = []
# for i in range (len(data)):
#     column_x.append(i)
# data["abcisse_time"]=column_x

# print(data)

# from pandas.plotting import scatter_matrix
# scatter_matrix(data,figsize=(nb_collums(data),nb_collums(data)))

# from sklearn import cluster
# kmeans = cluster.KMeans(n_clusters=2)
# # array_data = np.array(column_x).reshape((len(data_2), 1))
# kmeans.fit(data)
# #index triés des groupes
# idk = np.argsort(kmeans.labels_)
# #affichage des observations et leurs groupes
```

```
# print(pandas.DataFrame(data.index[idk],kmeans.labels_[idk]))
# #distances aux centres de classes des observations
# print(kmeans.transform(data))

""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""



"""""""""""""""""""""""""""""""""""""""""" to determin waste of energy with CO2, fenetre and
chauffage   first way   """""""""""""""


def waste(data):
    w = np.zeros(len(data["C02"]))
    for n in range(len(data["C02"])):
        if data["C02"][n] > 1000 and data["fenetre"][n] == 1 and data["chauffage"][n]
> 0:
            w[n] = 1
    return w


Waste_of_energy = waste(result)
# We create a new column to the dataframe containing the waste information

result['waste'] = Waste_of_energy

# print(result)

# We indicate the dates where there is waste of energy

for n in range(288):
    if (result["waste"][n] == 1):
        print("Gaspillage le :", result.index[n])


"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""" to determine when turn off the box in
order to save energy"""""""""""""""""""""""""


def box_waste(data):
    # Premièrement on discretise les valeurs
    newbox = []
    value = box['value']
    for n in range(len(value)):
        if (value[n] < 17.8):  # De manière peu rigoureuse, on a déterminerz Par
```

```
lecture graphique que sous ce seuil la boxe semblait en veille

            newbox.append(0)
        else:
            newbox.append(1)

    box['newbox'] = newbox
    len_box = len(box)

    #     /!\ Puissance en dents de scie cela risque de poser problème
    #     -> on Lisse la puissance

    for i in range(len_box-2):
        if newbox[i+2] == 1 and newbox[i] == 1:
            newbox[i+1] = 1

    print(box)

    # Création d'une liste contenant des plages horaires sur lesquelles la boxe
est utilisée
    liste_ALLUMAGE_globale = []
    sample = 1
    while sample <= (len_box-1):  # Pour pouvoir déterminer les heures
d'allumage et d'extinction de la boxe il est nécessaire de supprimer la première
et la dernière valeur
        time = box.index[sample]
        month = time.month
        day = time.day
        today = day
        the_month = month
        while the_month == month:  # 1 On récupère les mois, on fait une boucle
qui tourne dans un mois
            time = box.index[sample]
            the_month = time.month
            while today == day:
                # 2 Second boucle qui tourne pour travailler avec chaque jour
                #     /!\attention le nombre de jours de chaque mois n'est pas le
même,cette boucle devrait contourner le pb

                time = box.index[sample]
                today = time.day
                hour = time.hour
                minutes = time.minute
                # 3 Pour chaque jour on convertit le temps en seconde, plus facile à
```

```
traiter
            time_in_sec = minutes*60+hour*3600

            # 4 On récupère les heures d'allumage et l'extinction de la boxe
            # 5 On crée des tableau Avec une heure d'allumage et une heure
d'extinction
            tab_allumage = [0,0]
            if newbox[sample-1] == 0 and newbox[sample+1] == 1:
               tab_allumage[0] = time_in_sec
            if newbox[sample-1] == 1 and newbox[sample+1] == 0:
               tab_allumage[1] = time_in_sec

            if tab_allumage != []:
               # On stocke tous les couples allumage/extinction dans une même
liste
               liste_ALLUMAGE_globale.append(tab_allumage)
            sample += 1

         sample += 1

   # Traitement de la liste allumage globale
   # 6 On réalise des groupes avec les heures relevées.
# Ces groupes devant contenir les heures similaires
   print (liste_ALLUMAGE_globale)
   while liste_ALLUMAGE_globale != []:
      groupe = []
      groupe.append(liste_ALLUMAGE_globale[0])
      for i in range (len(liste_ALLUMAGE_globale)):
         if abs(liste_ALLUMAGE_globale[0] - groupe[0])<3600:#Si les valeurs
d'allumage sont similaires une heure près et les places dans une même liste
            groupe.append(liste_ALLUMAGE_globale[0])

      grp=[]
      grp.append(groupe) #GRP est une liste de liste, Chaque sous-groupe
contient les plages horaires similaires d'allumage

# Calculer les heures moyenne d'allumage aux differents moment de la
journée

   for i in range (len(grp)):
      moy=0
      for n in range (len(grp[i])):
         moy=moy+grp[i][n][0]
      moy =moy\(len(grp[i]))
```

box_waste(box)

6 On récupère la moyenne des valeurs de chaque groupe (centre ?) Dans la boucle

7 Place cette valeur dans un tableau
(première ou seconde place en fonction de si on est dans la boucle allumage ou extinction)

8 Première valeur (allumage) -1800 secondes : allumage de la boxe
Seconde valeur (Extinction) + 1800 secondes : extinction de la boxe