Novarina Paul & Poggio Maxime

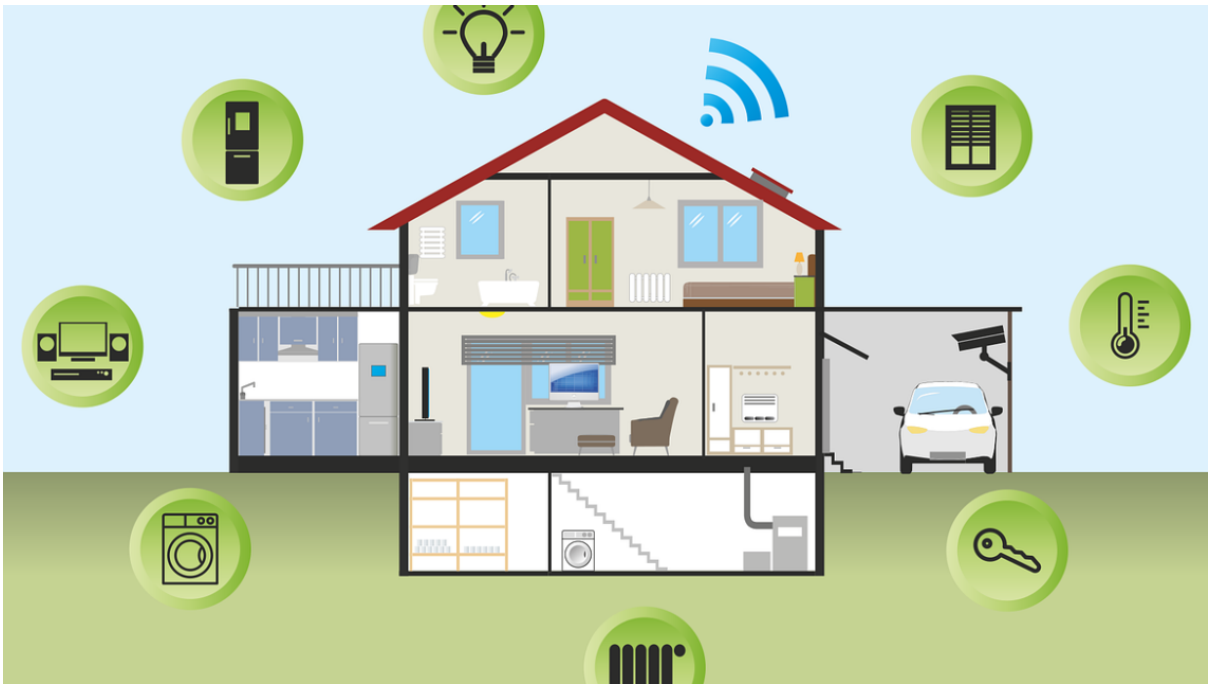
BE GESTION ÉNERGIE

SOMMAIRE :

Professeurs référents : **Manar Amayri, Ferrari Jérôme.**

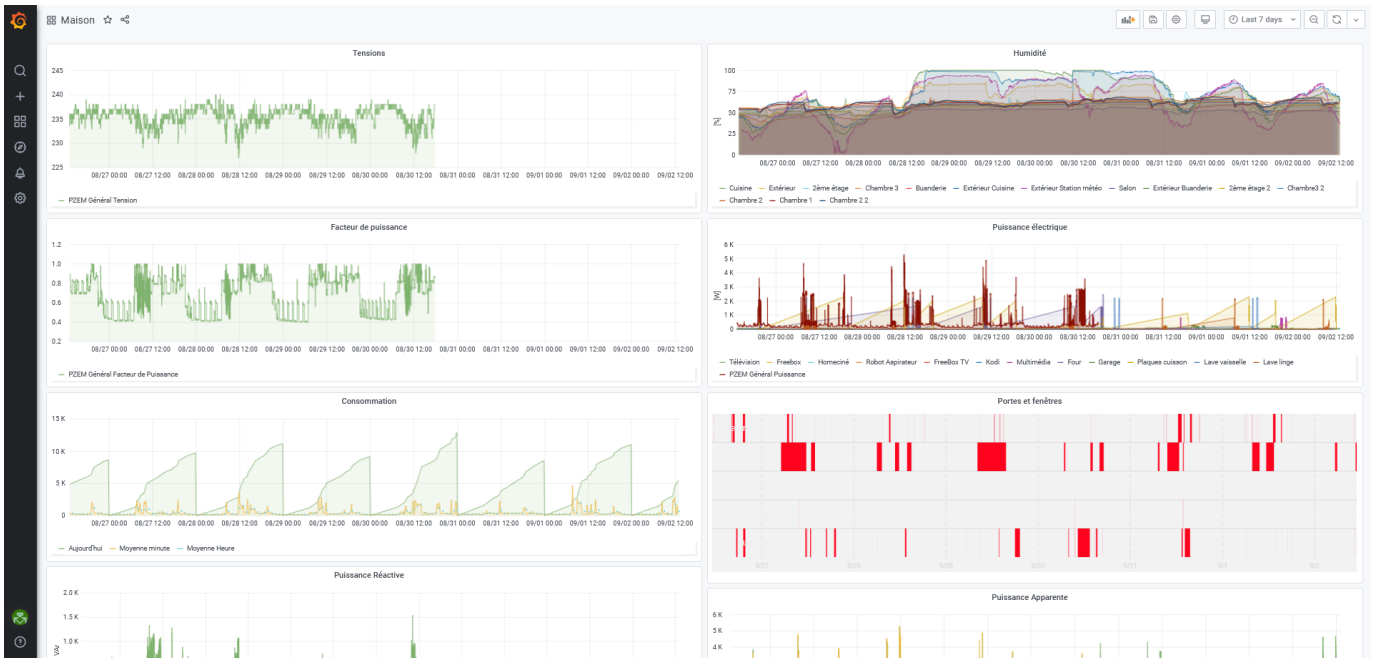
Introduction :

Durant notre projet Gestion d'Énergie nous avons tout d'abord appris à analyser des données. Les données d'une maison étaient mises à disposition et notre objectif a été d'évaluer à partir de ces données quel était le plus important des capteurs, la donnée la plus représentative de la réalité ou encore quel était le taux d'erreur de nos données. Nous avons ensuite prédit nos données à l'aide d'outils spécifiques et analysé cette prévision.
Nous avons ensuite réutilisé toutes ces méthodes afin d'analyser les données d'une maison sur plusieurs années. Nous étions libres de travailler sur le sujet que nous voulions en analysant les données que l'on souhaitait avec les techniques que nous trouvions les plus pertinentes.



Objectifs :

Depuis des dizaines d'années l'optimisation énergétique des habitats devient un enjeu majeur. Dans ce projet, nous avons la chance de pouvoir nous tester nous même à cette expérience. Nous travaillerons sur une maison datant de 2018, de 120 m^2où loge 5 personnes. Ce projet nous offre environ 340 mesures scientifiques accessibles sur un portail Grafana fourni par une base de données Influxdb sous cette forme :



Ici tout réside dans le choix des données que nous allons analyser et comment nous allons les analyser. Nous avons donc décidé d'utiliser :

- l'évolution de CO2
- l'évolution de température
- l'évolution de l'humidité
- l'ouverture des fenêtres au fil du temps
- la consommation de la pièce

Ainsi nous allons chercher à mettre en évidence un corollaire entre la consommation d'énergie et l'ouverture des fenêtres et de la population dans la chambre.

Extraction des données :

Nous avons donc initialement reçu une liste de données avec des identifiants sous ce format là :

```
'id': '734'   'name': 'Détecteur Ouverture Porte Entrée/Ouverture'
'id': '895'   'name': 'Détecteur Ouverture Fenêtre Salle à  Manger/Ouverture'
'id': '890'   'name': 'Détecteur Ouverture Fenêtre Cour/Ouverture'
'id': '885'   'name': 'Détecteur Ouverture Fenêtre Evier/Ouverture'
'id': '1109'  'name': 'Détecteur Ouverture Porte Buanderie/Ouverture'
'id': '4281'  'name': 'Détecteur Ouverture Baie Droite/Ouverture'
'id': '4286'  'name': 'Détecteur Ouverture Baie Gauche/Ouverture'
'id': '4291'  'name': 'Détecteur Ouverture Fenêtre Chambre 2/Ouverture'
'id': '4296'  'name': 'Détecteur Ouverture Fenêtre Chambre 1/Ouverture'
'id': '4346'  'name': 'Détecteur Ouverture fenêtre Salle de bains/Ouverture'
'id': '4351'  'name': 'Détecteur Ouverture Fenêtre chambre 3/Ouverture'
'id': '4356'  'name': 'Détecteur Ouverture Fenêtre 2eme etage Salle de bains/Ouverture'
'id': '4361'  'name': 'Détecteur Ouverture Fênetre 2eme étage chambre/Ouverture'
```

Ensuite l'objectif était d'obtenir les banques de données correspondant à ce que l'on cherchait, par exemple l'"ouverture fenêtre 2ème étage chambre".
Pour cela on utilise l'identifiant associé dans cette consigne :

```
datasheet = client.query('SELECT "value" FROM "jeedom"."autogen"."5229" WHERE time > now() - 4d')
```

qui permet de sélectionner les datas correspondant à à l'id "5229".

Et le code suivant nous permet d'afficher ces données et de les transformer en csv.

```
with open("output1.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])
```

Enfin il ne nous reste qu'à ouvrir nos fichiers à l'aide de l'instruction suivante :

```
with open ("Z:/Gestion d'énergie/donne_final/BEFERRARIDATAouverturefenetreCH2date_new.csv","r",) as fichier:
```

On aura ensuite un fichier au même nom sous format .txt

Code complet:

```
from http.server import BaseHTTPRequestHandler, HTTPServer
import urllib
import time
from influxdb import InfluxDBClient
import sys
import csv
##########################
#    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_s

print(client.get_list_database())

client.switch_database('jeedom')

datasheet = client.query('SELECT "value" FROM "jeedom"."autogen"."5229" WHERE time > now() - 4d')

print(datasheet)

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

with open("output1.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])
```

Méthode avec panda:

```
dataCO2 = pd.read_csv('/Users/poggi/Desktop/BEFEFE/newdata/DatacsvCO2.csv', sep=";")
dataHum = pd.read_csv('/Users/poggi/Desktop/BEFEFE/newdata/DatacsvHum.csv', sep=";")
dataOuvFen = pd.read_csv('/Users/poggi/Desktop/BEFEFE/newdata/DatacsvOuvFen.csv', sep=";")
dataTemp = pd.read_csv('/Users/poggi/Desktop/BEFEFE/newdata/DatacsvTemp.csv', sep=";")
dataconso = pd.read_csv('/Users/poggi/Desktop/BEFEFE/newdata/conso.csv', sep=";")
```

Mise en forme des données:

On a tout d'abord moduler les données par pas de 5 min:

```
dataCO2=dataCO2.resample('5T').sum()
```

Ensuite il y avait répartie d'une manière inconnue des NaN (Not a Number) dans nos data, on les a donc supprimés ,par le code:

```
dataCO2 = np.nan_to_num(dataCO2)
dataHum = np.nan_to_num(dataHum)
dataOuvFen = np.nan_to_num(dataOuvFen)
dataTemp = np.nan_to_num(dataTemp)
dataconso = np.nan_to_num(dataconso)
```

Afin d'implémenter un decision tree, on a créer label et concaténer les data:

```
AllData=[dataCO2, dataHum, dataOuvFen, dataTemp, dataconso]

result=pd.concat(AllData,axis=1,join='inner')
print (AllData)
print (result)
```

On a donc implémenter un decision tree en fonction de l'importance des variables:

```
### DECISION TREE###

x_train, x_test, y_train, y_test = train_test_split(result[["CO2","Hum","OuvFen","Temp","conso"]],result[['conso']],test_size=.1,random_

def featureimportant (x_train, y_train) :
    classifier = tree.DecisionTreeClassifier(random_state=0, criterion='entropy', max_depth=None)
    classifier.fit(x_train, y_train)
    important = classifier.feature_importances_
    return important

classifier = tree.DecisionTreeClassifier(random_state=0, criterion='entropy', max_depth=None)
variablelaplusimportante = featureimportant (x_train, y_train)

print("la variable la plus déterminante est:",variablelaplusimportante)

important = classifier.feature_importances_
numero = np.argsort(important)
plt.title('feature importance')
plt.barh(range(len(numero)), important[numero], align = 'center')
plt.yticks(range(len(numero)), [x_train.columns.values[i] for i in numero])
plt.xlabel('Relative importance')



def decision_tree(x_test,y_test,x_train,y_train) :
    classifier=tree.DecisionTreeClassifier(max_depth=None)
    classifier.fit(x_train,y_train)
    prediction=classifier.predict(x_test)
    accuracy = classifier.score(x_test, y_test)
    print("accuracy", accuracy)
    with open ("decisiontree.dot","w") as file :
        f=tree.export_graphviz(classifier,out_file=file)
    return(prediction, accuracy)
```

Enfin on a calculer le gaspillage effectué en comparant les résultats du decision tree et la consommation de la pièce:

```
n = 0
for i in result['conso']:
    if i < 300:
        result.loc[n, 'conso'] = 0
    else:
        result.loc[n, 'conso'] = 1
    n = n + 1

print ("le nombre dde minutes consommées pour rien vaut", n*5)
sommeconsogaspiller = 0
gaspillage=np.zeros(len(result))
for i in range (len(result)) :
    if (occupancy == 1 and dataOuvFen == 1 and dataCO2 < 700 and dataconso > 300) :
        gaspillage[i]=1
        sommeconsogaspiller += dataconso

plt.plot(result,gaspillage)
plt.xlabel("Time")
plt.ylabel("Wasted energy")

print (sommeconsogaspiller)
```

Analyse des résultats:

Lors de notre projet, nous avons rencontré un problème sur notre code:

En effet, lors de la concaténation de nos données en un tableau panda, on a eu une erreur de type que nous n'avons pas réussi à résoudre malgré internet (la méthode de concaténation a été elle-même obtenue grâce à internet). De ce fait nous n'avons pas pu exploiter les résultats mais nous avons écrit un code qui semble cohérent, affichant normalement la somme des consommations en Wh gaspillées ainsi qu'un graphique les représentant.
Afin d'écrire le code sans les résultats du decision tree, on a décidé de prendre des valeurs des data qui nous paraissait cohérente pour estimer la présence de personnes.

Conclusion et critique:

Pour revenir sur notre méthode, l'objectif de base était de connaître l'énergie dépensée par une pièce de la maison n'ayant pas d'utilité et donc étant gaspillée. Pour cela on a choisi d'étudier une chambre, ce qui permet de se concentrer principalement sur le chauffage pour l'énergie dépensée. On a alors décidé initialement de prendre en compte la température et l'humidité, afin de savoir sur la consommation du chauffage est utile ou pas, puis le taux de CO2, afin de connaître la présence de personnes dans la pièce, l'ouverture de la fenêtre afin de savoir lorsque la consommation du au chauffage est gaspillée et enfin la consommation.

En supposant que notre code fonctionne, il nous aurait théoriquement renvoyé la consommation gaspillée selon le decision tree permettant d'estimer la présence de personnes selon les variables les plus importantes obtenus via classifier.
Cependant, notre système de décision prendra uniquement en compte le gaspillage lorsque la personne n'y est pas, la consommation est haute et la fenêtre est ouverte. De ce fait, on ne prend pas en compte la consommation gaspillée à chauffer la pièce lorsque personne n'y est ou encore la température et l'humidité initiale de la pièce.

De plus, la consommation utilisée tirer de Grafana est dans notre cas celui de la maison entière(on s'en est rendu compte tardivement), sachant que les données étudiées sont ceux de la chambre 2, notre étude n'a pas de sens. Malgré ceci, on se place dans le cas où nous avions accès au chauffage de la chambre 2 comme ce qui était prévu et dans ce cas notre étude est tout de même viable.

Enfin pour conclure sur notre projet, nous aurions voulu pouvoir implémenter un décision tree pour obtenir à la fois la consommation gaspillée de manière exacte et également commandé le chauffage de la pièce en fonction des données à disposition. Cependant nous avons buté sur de nombreuses erreurs avec le code et nous n'avions pas été assez organisé dans nos démarches pour obtenir ce que l'on voulait. Enfin ce projet nous a fait réfléchir sur les nombreuses possibilités d'amélioration et d'utilisation de data open source pour l'ingénieur et nous pensons malgré notre échec être devenu plus attentif et plus doué pour la gestion de données.