Smart Systems Project

Etude sur la rentabilisation économique et énergétique en utilisant des capteurs en éclairage

Prof. Jérôme Ferrari

SEM-3A

Presented By:

Thomas Guillot Goguet

Wiam Razi

Table des matières

1. Introduction

L'éclairage automatique est un système permettant d'allumer une pièce ou une zone sans intervention humaine. Il est devenu automatique dans certaines pièces car il offre commodité et sécurité en éclairant les zones qui seraient autrement sombres et donc peu sûres.

La révolution annoncée des IOT (Internet of Things) c'est-à-dire du développement d'objets connectés entre eux va entraîner une augmentation majeure dans l'utilisation de nouveaux capteurs. Cette augmentation du nombre de ces capteurs pose aussi le coût écologique associé. Certes, ces capteurs sont conçus pour réduire notre consommation électrique et nous simplifier la vie mais est-ce qu'en prenant en compte l'alimentation, le coût de fabrication, et la fin de vie de ces appareils, nous avons toujours un coût environnemental plus favorable que si nous n'avions pas mis en place ces dispositifs ?

2. Méthodologie et Raisonnement

Notre projet consiste à étudier la rentabilité énergétique et économique de l'utilisation de l'éclairage qui s'allume avec des détecteurs et qui s'éteint automatiquement.

Le projet de l'Expe-smarthouse, qui couvre une surface de 120m² et où vit une famille de 5 personnes, a été initié en 2018 afin de fournir les données relatives au foyer.

Pour simplifier notre travail, nous nous sommes focalisées sur l'étude de la buanderie puisque c'est la seule pièce où se trouve l'éclairage automatique. Nous avons ensuite décidé ensuite de faire l'étude uniquement sur deux mois(décembre 2022 et janvier 2023).

Nous avons exploité les données sur python grâce aux modules http.server, urllib, influxdb et datetime issues de l'expérience de la Smart House. Nous avons eu accès aux capteurs suivants pour la réalisation de notre étude :

- Détecteur de mouvement (mouvement, luminosité, durée ouverture, batterie)
- Détecteur de l'ouverture de la porte (ouverture, batterie)
- Détecteur de luminosité

Nous avons ensuite fait l'extraction de ces données sous un fichier csv.

Ces données n'étaient pas utilisables en l'état, il a fallu les formater pour les transformer en un élément utilisable. Ces opérations ont été réalisées en utilisant le module pandas ce qui nous a permis de les échantillonner et de les traiter.

En effet, ces données n'étaient pas au même format, les mesures étaient prises à des instants différents et le nombre de mesures n'était pas la même pour chaque capteur. Nous avons donc décidé dans un premier temps de rassembler les données dans un même fichier, et de ne considérer qu'il ne peut y avoir au maximum qu'une seule opération par seconde. Ensuite, il a fallu s'occuper des valeurs manquantes. En effet, comme les capteurs ne visualisent pas les données au même instant, il arrive qu'il n'y ait pas de valeur de certains capteurs à un certain moment alors que pour d'autres c'est le cas.

Maintenant que les données sont prêtes à être analysées, nous allons utiliser de construire un modèle de XAI en utilisant la méthode de kmeans pour prédire notre label et ensuite utiliser "Decision Tree" comprise dans le module Scikit Learn pour déterminer les capteurs les plus importants pour l'éclairage. Nous déterminerons ainsi comment la pièce est utilisée par les résidents. Ensuite, nous allons évaluer la durée pendant laquelle les lumières sont allumées avec le détecteur de mouvement.

Nous élaborerons ensuite des hypothèses sur l'utilisation de la pièce pour avoir une courbe de consommation de l'électricité de l'éclairage à partir de la puissance de la lumière utilisée et de sa durée d'utilisation.

Par la suite, on précisera la durée pendant laquelle les lumières seraient allumées sans les détecteurs et on comparera cette durée à celle avec les détecteurs. On calcule ensuite la différence entre la consommation d'énergie actuelle et la consommation d'énergie potentielle. Et pour déterminer les économies d'énergie potentielles en monnaie, on multiplie la différence de consommation d'énergie par le coût de l'énergie électrique qui est de 0,2062 € TTC par kWh selon EDF.

Ce raisonnement nous a permis d'obtenir une estimation des économies d'énergie potentielles que nous avons réalisées en utilisant des détecteurs de mouvement et d'ouverture de porte pour l'éclairage automatisé. Il est important de noter que ces estimations peuvent varier en fonction de la fréquence réelle des mouvements et des ouvertures de porte dans la pièce.

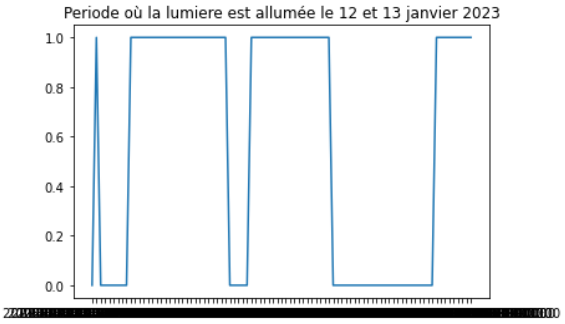
3. Analyse des données

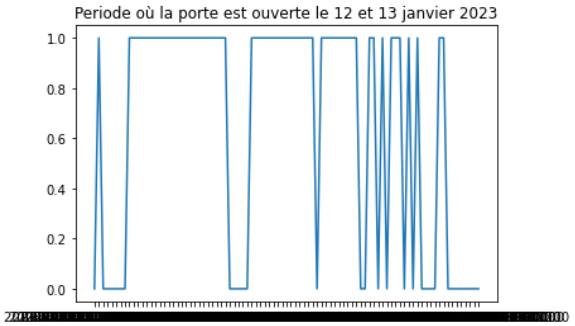
Notre expérience s'étalant sur une période réduite, nous n'allons pas considérer l'importance de la mesure de la tension au borne de la pile du capteur. Celà aurait pu être utile pour calculer le nombre de pile nécessaire à l'utilisation du capteur sur une longue période et ainsi être plus précis sur les coûts économiques et écologiques de l'utilisation d'un tel capteur. Cependant, comme notre expérience se déroule sur deux semaines, il semble pertinent de considérer qu'il n'y aura pas besoin de changer l'alimentation des capteurs sur cette période.

Pour analyser de manière efficace nos données il a fallu faire l'extrapolation de la valeur de certaines données lorsqu'elles n'ont pas été mesurées. Pour les données de type booléen (0 ou 1) comme le capteur pour savoir si la lumière est allumée, s'il y a du mouvement dans la pièce ou si la porte est ouverte, nous avons répété la dernière valeur jusqu'à la prochaine valeur différente.

Il a été assez difficile de trouver la logique entre les différentes valeurs mesurées de la luminosité. Il a a donc été décidé de ne pas ajouter de nouvelles valeurs.

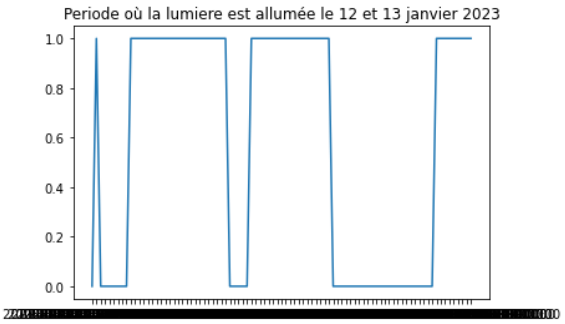
Les graphes suivants représentent les différents capteurs étudiés lors de cette étude:

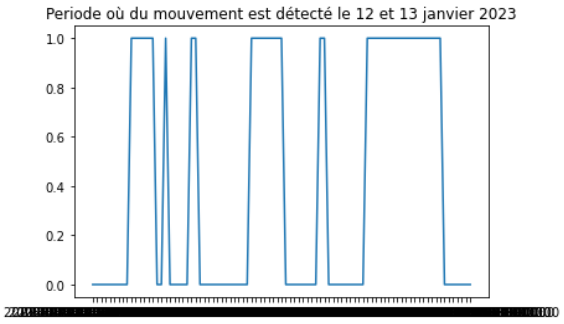




Il est important de garder en tête que l'échelle des abscisses dans cette représentation n'est pas uniforme. Un nouveau nouveau point est créé à chaque fois qu' un des capteur enregistre une information. Cette représentation n'est valide que pour comparer différents capteurs entre eux. A titre d'information, la lumière n'est restée allumée que 1024 minutes et la porte ouverte pendant 925 minutes sur ces deux jours. A première vue, ces résultats nous semblent trop importants pour une simple buanderie. Il aurait été intéressant d'avoir plus d'informations sur l'utilisation de cette pièce.

Effectuons maintenant la même manipulation mais en comparant la luminosité avec le capteur de mouvement.



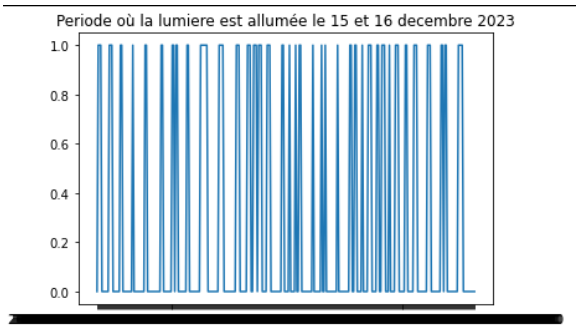


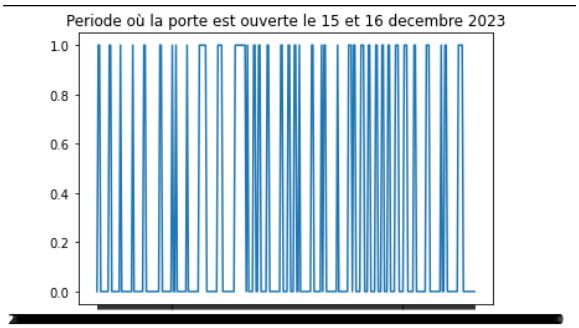
Le mouvement est détecté pendant 804 minutes sur ces deux jours.
Il semble à première vue avoir moins de rapport entre la période où le mouvement est détecté et où la lumière est réellement allumée qu'entre celui où la lumière est ouverte de porte. C'est pourquoi nous allons chercher à vérifier par la suite ça en mettant en place une

méthode de machine learning pour comprendre ce qu'apporte chaque capteur et si les deux sont nécessaires pour réaliser une telle prédiction.

Les résultats obtenus ici sont très surprenants et contredisent les données que nous avons grâce aux détecteurs d'absence de mouvement et d'absence de fermeture de porte.

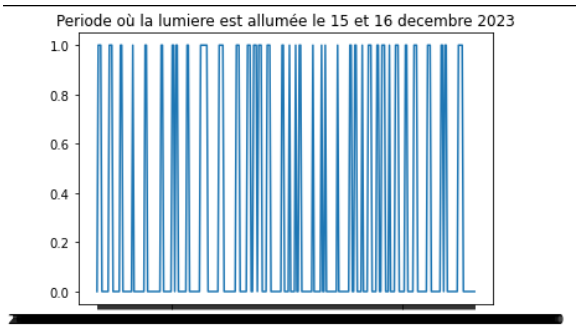
Nous avons donc réitéré cette expérience en choisissant deux autres jours pour vérifier s'il n'y avait pas eu un problème lors de la captation des données ou lors de notre manipulation préliminaires de nos données.

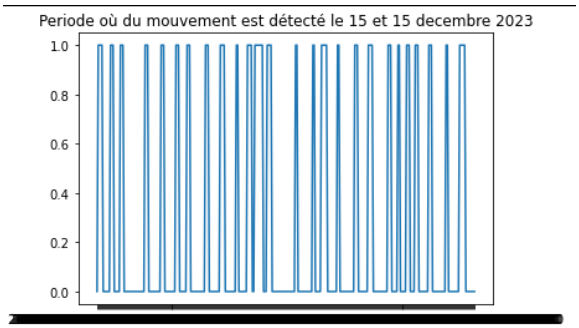




Les résultats sont complètement différents par rapport à la première étude. Celà traduit cependant toujours une utilisation très importante de la pièce, plus que ce que nous attendions. En effet, la lumière s'est allumée 76 fois pendant ces deux jours.
Au sujet de la durée où la lumière est allumée, on passe à 17 minutes et 49 secondes. Pour la porte, elle est restée ouverte pendant 40 minutes et 8 secondes et s'est ouverte 76 fois aussi.

Regardons maintenant ce qu'il en est pour le mouvement.





Une nouvelle fois les résultats nous semblent plus cohérents. Le mouvement est détecté 52 fois pour une durée de 56 minutes. Ces résultats sont compatibles avec les capteurs d'absence de mouvement ou de fermeture ce qui nous rassure sur la fiabilité de la méthode employée.

La question se pose de savoir pourquoi on a obtenu de tels résultats dans un premier temps, est-ce qu'il y a eu un défaut de fonctionnement avec les capteurs ces jours-là ? Est-ce qu'un événement exceptionnel a fait que cette pièce a été surutilisée ?

Nous ne pouvons pas établir de réponses certaines. Cependant, avant passer à la prédiction, il semble pertinent de continuer à analyser nos données pour voir quels sont les jours exploitables et si cette défaillance supposée des capteurs est un phénomène rare ou non. Nous ferons cette étude sur les données disponibles du 15 décembre au 15 janvier.

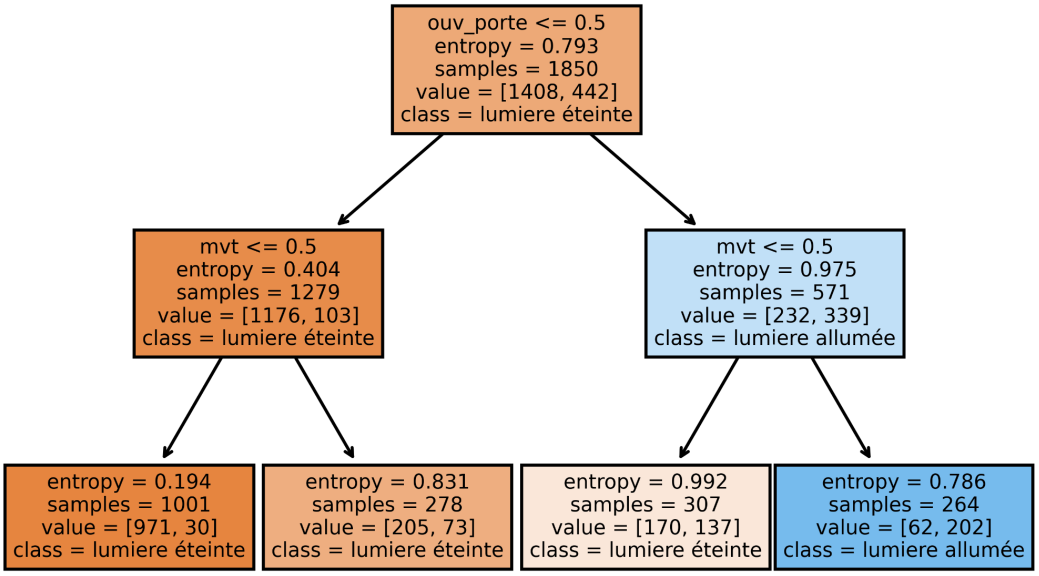
Nous avons une période sans aucune données du 26 décembre au 8 janvier, nous ne le prendrons donc pas en compte dans cette étude. Les données ont été séparées en trois parties et on a regardé seulement le temps où la lampe restait allumée.

Période	15 au 22 décembre (7 jours)	23 au 26 décembre (4 jours)	9 au 14 janvier (6 jours)
Durée lumière allumée	4 jours et 10 heures	2 jours	2 jours et 15 heures

La lampe reste allumée bien trop longtemps pour une utilisation normale de la buanderie. Le cas étudié lors du premier test est plus commun que le cas étudié dans le second. Nous procéderons donc à la détermination du meilleur capteur en gardant en tête ce problème d'interprétation des données mais sans modification supplémentaire.

4. Résultats

Nous avons utilisé dans un premier temps une méthode de classification pour ce premier test. La taille de nos données d'entraînement est fixée à 80% du total des données pour éviter une problème de sous ou de sur-ajustement. Nous n'avons pas pu prendre en compte les données présentées par la capteur de luminosité car l'algorithme ne supporte pas les données de type Nan. En utilisant donc le capteur de mouvement et celui de l'ouverture de la porte nous obtenons l'arbre de décision suivant.

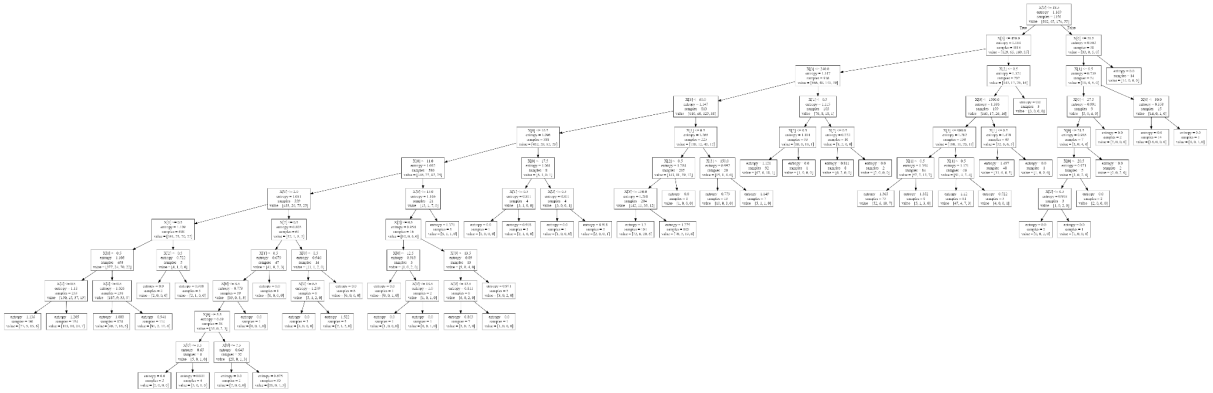


Par la position de l'ouverture de la porte en premier critère, il semble être plus important pour les décisions et donc le critère à favoriser. L'algorithme mis en place par l'ordinateur consiste à considérer la lumière allumée seulement si on détecte simultanément du mouvement et que la porte est ouverte.
Faisons la même expérience mais avec seulement en retirant l'un des capteurs. Les résultats sont présentés dans le tableau ci-dessous.

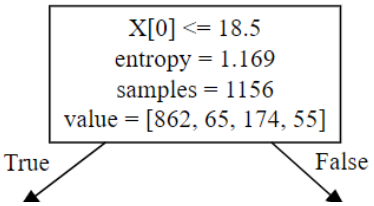
	Avec les deux capteurs	Avec détecteur de mouvement seulement	Avec détecteur d'ouverture seulement
Précision	0.844	0.816	0.816
Erreur	0.156	0.184	0.184
F-score	0.771	0.766	0.778

Les résultats montrent que le capteur le plus fiable est l'utilisation combinée des deux capteurs au niveau de la précision et de l'erreur. Cependant, le détecteur de mouvement seul est le meilleur en ce qui consiste à ne pas s'allumer lors des faux positifs (la lumière s'allume alors qu'il ne faut pas). L'erreur n'est pas nulle sur le code, nous n'avons pas réussi avec ces deux capteurs à déterminer précisément quels sont les éléments qui déclenchent effectivement la lampe. Probablement que l'intégration de la luminosité à ces différents paramètres permettra d'obtenir des résultats plus proches de la réalité. La variation de ces résultats reste cependant très faible et ne semble pas montrer l'utilité d'utiliser deux types de capteurs pour seulement le système de lumière automatique.

Dans un second temps, nous avons essayé de résoudre le problème autrement pour en utilisant une méthode XAI (eXplainable Artificial Intelligence qui consiste à utiliser une méthode de partitionnement puis de classification sur les résultats obtenus par le partitionnement). Nous avons aussi gérer les valeurs non existantes (les valeurs Nan qui signifie Not a number) avec des zéros pour simplifier la tâche au modèle. Enfin nous prendrons plus de paramètres d'entrée à savoir : luminosité, ouverture de porte, mouvement et absence du mouvement.
La prédiction a été réalisée avec la fonction K-Means qui est une méthode de partitionnement qui sert à regrouper les données dans différentes classes. Ensuite, nous obtenons l'arbre de décision suivant :



On s'intéresse au critère le plus important c'est à dire celui qui se trouve tout en haut de l'arbre de décision.



Sur le fichier CSV, on a les capteurs classés comme suit :

```
1  time;lumino;ouv_porte;mvt;abs_mvt;label
```

Donc en regardant le maximum et la liste des valeurs value=[862,65,174,55], on déduit que le capteur le plus important est celui de luminosité et ensuite vient celui de mouvement selon le classement de colonnes qu'on a dans le fichier csv.
Notre compréhension initiale du capteur de luminosité est qu'il permettait de déterminer s' il y avait déjà assez de lumière dans la pièce de voie naturelle (par la lumière du soleil qui passe par une fenêtre). S'il y avait déjà assez de lumière, la lumière ne s'allumait pas.

En fin, nous nous attendions bien d avoir ce résultat qui montre que le capteur de luminosité est l'essentiel pour décider d'allumer la lumière ce qui permet de faire des économies d'énergie dans un 1er temps et d'argent dans un second temps surtout quand la luminosité est assez et on a pas besoin d'allumer l'éclairage malgré un mouvement détecté

Nous nous sommes aussi posé la question de si le capteur de luminosité captait la lumière de la lumière directement ce qui expliquerait une corrélation aussi importante. Or cette théorie n'est pas supportée par le fait que la valeur ne soit jamais identique à ce qu'on pourrait raisonnablement penser sur une lumière artificielle.
En utilisant ces modèles et ces capteurs, on obtient une erreur qui est plus petite que celle précédente ce qui est plus précis et proche de la réalité de 0.0069 :

```
accuracy:1.0,recall:0.99654 ,fscore:0.99827:
mean absolute error is : 0.006920415224913495
```

5. Conclusion

L'analyse préliminaire des différentes données a soulevé de nombreuses incompréhensions sur le fonctionnement de la buanderie et de comment fonctionnent les capteurs.

En construisant notre modèle, on comprend que l'élément le plus important pour l'éclairage est le capteur de luminosité et il est donc indispensable pour nos questions d'économie économique ou de sobriété énergétique.

Dans le projet initial, nous voulions comparer deux scénarios pour allumer la lumière : un avec détecteur et un autre sans. Cependant, nous manquons encore beaucoup d'informations pour pouvoir continuer ce travail. Nous avons besoin d'étude statistique pour savoir la fréquence et la durée d'un oubli d'éteindre la lampe puisque la lampe est dans la buanderie, une pièce où on suppose que les enfants s'y rendent rarement. Nous avons aussi besoin des références précises des capteurs et ainsi calculer leur cycle de vie. Nous gardons en tête que le mix énergétique français est pour une grande part décarboné donc l'impact sera faible sur les émissions de CO_2 mais il aura peut être plus d'influence sur l'écotoxicité et l'utilisation de nos matières premières.

6. Sources

[1] Référence pour le prix de électricité
https://www.fournisseur-energie.com/prix-kwh/

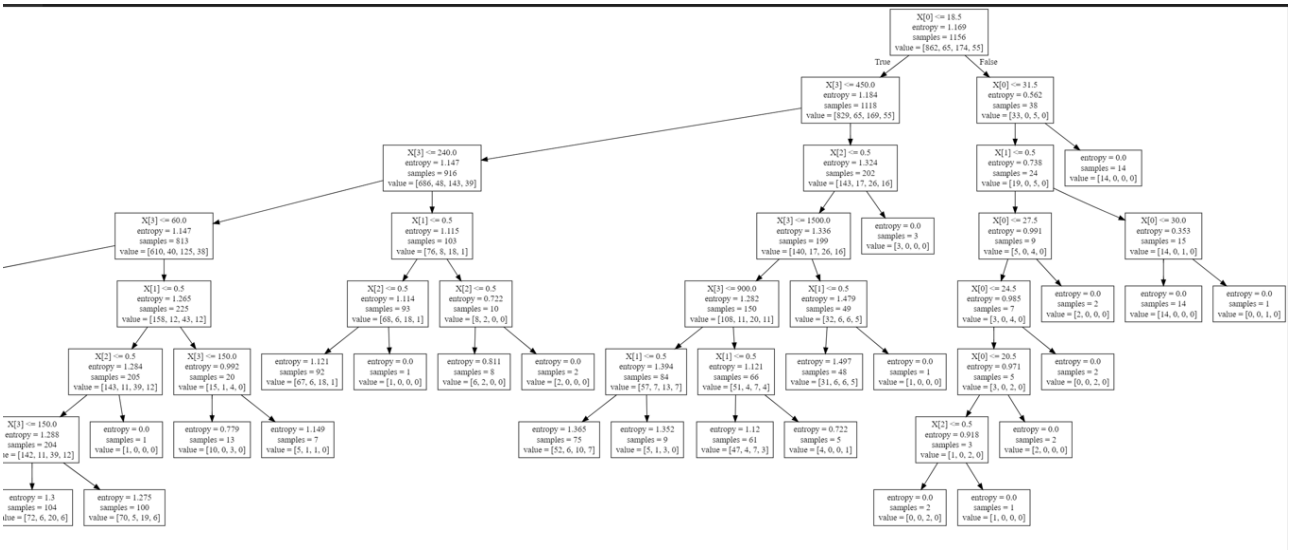
[2] La détection de mouvement associée à l'éclairage LED : sécurité et maîtrise de la consommation d'énergie. Siècle Digital. Septembre 2019

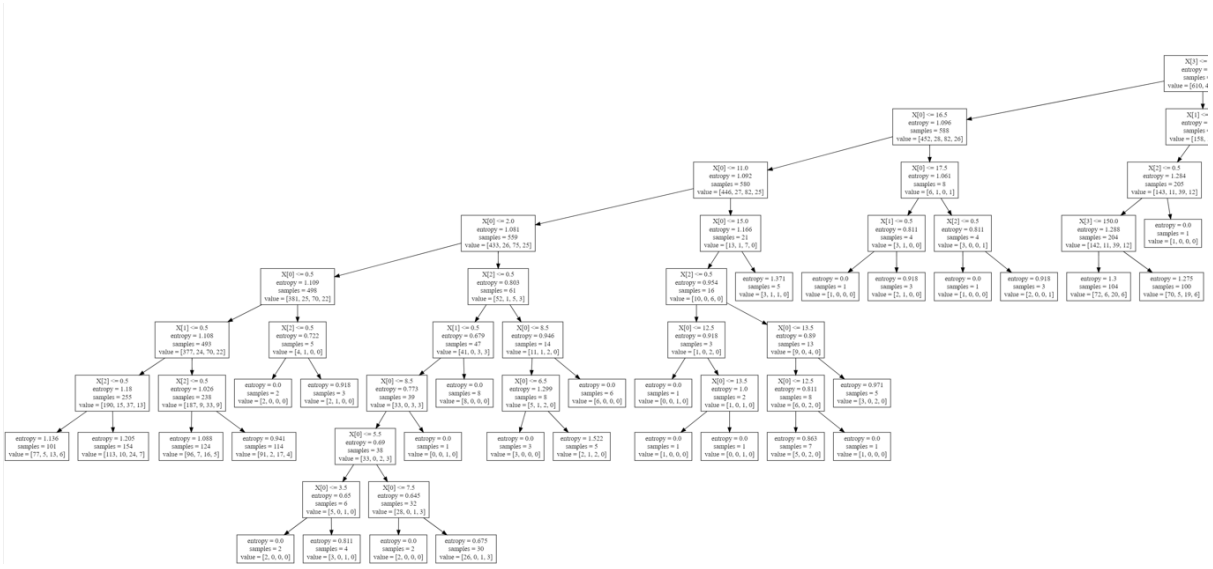
https://siecledigital.fr/2019/09/13/la-detection-de-mouvement-associee-a-leclairage-led-securite-et-maitrise-de-la-consommation-denergie/

[3] "À quelle fréquence éteignez-vous la lumière en quittant une pièce ?". Statistica. Mars 2015.
https://fr.statista.com/statistiques/607963/frequence-eteindre-lumiere-quitter-piece-france/

[4] Exemple des econnomies d'energie
https://theecoguide.org/benefits-using-motion-sensor-light-switches

7. Annexes





ID des capteurs ayant été récupéré pour réalisé l'étude

Capteur porte

'id':'1109' ***'name': 'Détecteur Ouverture Porte Buanderie/Ouverture'***

'id':'1110' ***'name': 'Détecteur Ouverture Porte Buanderie/Absence de fermeture'***

'id':'1111' 'name': 'Détecteur Ouverture Porte Buanderie/Batterie'

'id':'1112' 'name': 'Détecteur Ouverture Porte Buanderie/Pile Voltage'

Capteur mouvement

'id':'1956 'name': 'Détecteur Mouvement Buanderie/Mouvement'

'id':'1957 'name': 'Détecteur Mouvement Buanderie/Absence de mouvement'

'id':'1958 'name': 'Détecteur Mouvement Buanderie/Capteur Luminosité'

'id':'1959 'name': 'Détecteur Mouvement Buanderie/Batterie'

Lampe etat

'id':'1123' 'name': 'SonOff Buanderie Lumiere Relais/Etat'

Code ayant servi à faire l'extraction des données

Ce code a été utilisé en changeant à changer fois l'ID et le nom du fichier de sortie. Ce bout de code tel qu'il est présenté permet de comprendre ce qui a été fait mais a besoin de modification pour récupérer tous les fichiers nécessaires pour reproduire nos résultats.

```
from http.server import BaseHTTPRequestHandler, HTTPServer
import urllib
import time
from influxdb import InfluxDBClient
import sys
import csv
import pandas as pd
import matplotlib.pyplot as plt

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

client = InfluxDBClient(INFLUXDB_SERVER_IP, INFLUXDB_SERVER_PORT, INFLUXDB_USERNAME,
INFLUXDB_PASSWORD, INFLUXDB_DB_NAME,ssl=True, verify_ssl=False)
```

```
print(client.get_list_database())

client.switch_database('jeedom')


# Capteur mouvement

#  'id': '1958'      'name': 'Détecteur Mouvement Buanderie/Capteur Luminosité'
#  'id': '1959'      'name': 'Détecteur Mouvement Buanderie/Batterie'

detecteur_battery = client.query('SELECT "value" FROM "jeedom"."autogen"."1959" WHERE time > now() -
365d')
#760 correspond au capteur voulu qui a été récupéré



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

with open("C:/Users/tguil/Desktop/Dernier cours de l'E3/detecteur_battery.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])

df = pd.read_csv("C:/Users/tguil/Desktop/Dernier cours de l'E3/detecteur_battery.csv")
#df = pd.read_csv("C:/Users/tguil/Desktop/Dernier cours de l'E3/capteur_lumiere.csv")

df['time'] =pd.to_datetime(df['time'])
df['time'] = df['time'].dt.floor('s')
df.drop_duplicates(inplace=True)


x = df['time']
#print(x)
y = df['value']
#print(y)

print(df)

plt.plot(x,y)

# beautify the x-labels
plt.gcf().autofmt_xdate()

plt.show()

df.to_csv("C:/Users/tguil/Desktop/Dernier cours de l'E3/datadoubleremoved_detecteur_battery.csv",index=False)
```

Code ayant servi à faire le traitement des données

```
from http.server import BaseHTTPRequestHandler, HTTPServer

import urllib

import time

from influxdb import InfluxDBClient

import sys

import csv

import pandas as pd

import matplotlib.pyplot as plt

import math


ouv_porte=pd.read_csv('ouverture_porte_ouverte.csv')

pile_volt=pd.read_csv('ouverture_pile_volt.csv')

mvt=pd.read_csv('detecteur_mvt.csv')

abs_fermeture=pd.read_csv('ouverture_abs_fermeture.csv')

abs_mvt=pd.read_csv('detecteur_abs_mvt.csv')

lumino=pd.read_csv('datadoubleremoved_detecteur_luminosite.csv')

lumiere=pd.read_csv('datadoubleremoved_lumiere.csv')


#removing duplicates

ouv_porte['time'] =pd.to_datetime(ouv_porte['time'])

ouv_porte['time'] = ouv_porte['time'].dt.floor('s')

ouv_porte.drop_duplicates(inplace=True)


pile_volt['time'] =pd.to_datetime(pile_volt['time'])

pile_volt['time'] = pile_volt['time'].dt.floor('s')

pile_volt.drop_duplicates(inplace=True)
```

```
mvt['time'] =pd.to_datetime(mvt['time'])
mvt['time'] = mvt['time'].dt.floor('s')
mvt.drop_duplicates(inplace=True)

abs_fermeture['time'] =pd.to_datetime(abs_fermeture['time'])
abs_fermeture['time'] = abs_fermeture['time'].dt.floor('s')
abs_fermeture.drop_duplicates(inplace=True)

abs_mvt['time'] =pd.to_datetime(abs_mvt['time'])
abs_mvt['time'] = abs_mvt['time'].dt.floor('s')
abs_mvt.drop_duplicates(inplace=True)

lumino['time'] =pd.to_datetime(lumino['time'])
lumino['time'] = lumino['time'].dt.floor('s')
lumino.drop_duplicates(inplace=True)

lumiere['time'] =pd.to_datetime(lumiere['time'])
lumiere['time'] = lumiere['time'].dt.floor('s')
lumiere.drop_duplicates(inplace=True)

# Having same lenght
start, end = '2022-12-15 02:49:01','2023-01-15 00:00:00'

interv = (ouv_porte['time'] > start) & (ouv_porte['time'] <= end)
ouv_porte = ouv_porte.loc[interv]

interv = (pile_volt['time'] > start) & (pile_volt['time'] <= end)
pile_volt = pile_volt.loc[interv]
```

```
interv = (mvt['time'] > start) & (mvt['time'] <= end)

mvt = mvt.loc[interv]


interv = (abs_fermeture['time'] > start) & (abs_fermeture['time'] <= end)

abs_fermeture = abs_fermeture.loc[interv]


interv = (abs_mvt['time'] > start) & (abs_mvt['time'] <= end)

abs_mvt = abs_mvt.loc[interv]


interv = (lumino['time'] > start) & (lumino['time'] <= end)

lumino = lumino.loc[interv]


interv = (lumiere['time'] > start) & (lumiere['time'] <= end)

lumiere = lumiere.loc[interv]


ouv_porte.rename(columns={'value':'ouv_porte'}, inplace=True)



#On veut regrouper toutes les données dans un seul gros fichier


pile_volt.rename(columns = {'value':'pile_volt'}, inplace = True)

abs_fermeture.rename(columns = {'value':'abs_fermeture'}, inplace = True)

abs_mvt.rename(columns = {'value':'abs_mvt'}, inplace = True)

mvt.rename(columns = {'value':'mvt'}, inplace = True)

mvt.rename(columns = {'value':'mvt'}, inplace = True)

lumiere.rename(columns = {'value':'lumiere'}, inplace = True)

lumino.rename(columns = {'value':'lumino'}, inplace = True)
```

```
df1 = lumiere

df2 = lumino

df3=df1.merge(df2, how='right', on='time')

df4=df1.merge(df2, how='left', on='time')

df1=pd.concat([df3,df4])

df1.drop_duplicates(inplace=True)

df1=df1.sort_values(by='time')


labels=[ouv_porte,pile_volt,mvt,abs_fermeture,abs_mvt]


for i in range (len(labels)) :

        df2=labels[i]

        df3=df1.merge(df2, how='right', on='time')

        df4=df1.merge(df2, how='left', on='time')

        df1=pd.concat([df3,df4])

        df1.drop_duplicates(inplace=True)

        df1=df1.sort_values(by='time')


df1.rename(columns = {'Temps basé sur luminosité':'Temps'}, inplace = True)


#Initialisation

error = df1.iloc[0,1]

df1.iloc[0,1] = 0 #Lumiere

df1.iloc[0,3] = 0 #Ouverture_porte

df1.iloc[0,5] = 0 #Mouvement


for i in range (1,len(df1['time'])) :

        if math.isnan(df1.iloc[i,1]): #Vérifie si l'élément est un nombre ou pas

        df1.iloc[i,1] = df1.iloc[i-1,1]
```

```
        if math.isnan(df1.iloc[i,3]):
        df1.iloc[i,3] = df1.iloc[i-1,3]

        if math.isnan(df1.iloc[i,5]):
        df1.iloc[i,5] = df1.iloc[i-1,5]

df1.to_csv("C:/Users/tguil/Desktop/Dernier cours de l'E3/donnes_expérience_1an.csv",index=False)
```

Code ayant servi à faire l’étude de l'analyse des données et la méthode de classification

```
from sklearn import tree
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import datetime

from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, mean_squared_error, f1_score

data=pd.read_csv("C:/Users/tguil/Desktop/Dernier cours de l'E3/donnes_expérience_1an.csv", sep=',')

plt.plot(data.iloc[2127:2216,0], data.iloc[2127:2216,1])
plt.title('Periode où la lumiere est allumée le 12 et 13 janvier 2023')
#plt.show()

plt.plot(data.iloc[2127:2216,0], data.iloc[2127:2216,3])
plt.title('Periode où la porte est ouverte le 12 et 13 janvier 2023')
```

```
#plt.show()


plt.plot(data.iloc[2127:2216,0], data.iloc[2127:2216,5])
plt.title('Periode où du mouvement est détecté le 12 et 13 janvier 2023')
#plt.show()
#Travail sur deux autres jours
data_reduit=data.iloc[0:308,:]


plt.plot(data_reduit.iloc[:,0], data_reduit.iloc[:,1])
plt.title('Periode où la lumiere est allumée le 15 et 16 decembre 2023')
#plt.show()


plt.plot(data_reduit.iloc[:,0], data_reduit.iloc[:,3])
plt.title('Periode où la porte est ouverte le 15 et 16 decembre 2023')
#plt.show()


plt.plot(data_reduit.iloc[:,0], data_reduit.iloc[:,5])
plt.title('Periode où du mouvement est détecté le 15 et 15 decembre 2023')
#plt.show()


#Combien de fois la lumière s'allume par jour ?
test=0
compte_switch = 0
Tps_switch=[]
for i in range (1,len(data_reduit['time'])) :
        if data_reduit.iloc[i,1] != test :
        test=data_reduit.iloc[i,1]
        compte_switch += 1
        Tps_switch.append([data_reduit.iloc[i,0],data_reduit.iloc[i,1]])
```

```
#print(compte_switch)


#Et pour combien de temps ?
t1=pd.to_datetime(Tps_switch[0][0],format= "%Y-%m-%d %H:%M:%S", exact=False)
t2=pd.to_datetime(Tps_switch[1][0],format= "%Y-%m-%d %H:%M:%S", exact=False)
tps = t2-t1

for i in range (1,len(Tps_switch)//2) :
        t1=pd.to_datetime(Tps_switch[2*i][0],format= "%Y-%m-%d %H:%M:%S", exact=False)
        t2=pd.to_datetime(Tps_switch[2*i+1][0],format= "%Y-%m-%d %H:%M:%S", exact=False)
        tps += t2-t1

#print(tps)

#Meme chose mais avec l'ouverture de la porte
tests = [0,0]
comptes = [0,0]
Tpss = [[],[]]
indice = [3,5]

for i in range (1,len(data_reduit['time'])) :
        for j in range (2) :
        if data_reduit.iloc[i,indice[j]] != tests[j] :
        tests[j]=data_reduit.iloc[i,indice[j]]
        comptes[j] += 1
        Tpss[j].append([data_reduit.iloc[i,0],data_reduit.iloc[i,indice[j]]])
#print(compte_switch)
```

```
#Et pour combien de temps ?
t1=pd.to_datetime(Tpss[0][0][0],format= "%Y-%m-%d %H:%M:%S", exact=False)
t2=pd.to_datetime(Tpss[0][1][0],format= "%Y-%m-%d %H:%M:%S", exact=False)

duree_ouv_porte =  t2-t1
t1=pd.to_datetime(Tpss[1][0][0],format= "%Y-%m-%d %H:%M:%S", exact=False)
t2=pd.to_datetime(Tpss[1][1][0],format= "%Y-%m-%d %H:%M:%S", exact=False)
duree_mvt = t2-t1

for i in range (1,len(Tpss[0])//2) :
        t1=pd.to_datetime(Tpss[0][2*i][0],format= "%Y-%m-%d %H:%M:%S", exact=False)
        t2=pd.to_datetime(Tpss[0][2*i+1][0],format= "%Y-%m-%d %H:%M:%S", exact=False)
        duree_ouv_porte += t2-t1

for i in range (1,len(Tpss[1])//2) :
        t1=pd.to_datetime(Tpss[1][2*i][0],format= "%Y-%m-%d %H:%M:%S", exact=False)
        t2=pd.to_datetime(Tpss[1][2*i+1][0],format= "%Y-%m-%d %H:%M:%S", exact=False)
        duree_mvt += t2-t1

#Analyse durée d'allumage différents jour

semaine1=[i for i in range (1124)] #7 jours
semaine2=[i for i in range (1124,1744)] #pas de données du 25 décembre au 9 janvier #4 jours
semaine3=[i for i in range (1744,2311)] #6 jours


t1=pd.to_datetime(data.iloc[0][0],format= "%Y-%m-%d %H:%M:%S", exact=False)
t2=pd.to_datetime(data.iloc[1][0],format= "%Y-%m-%d %H:%M:%S", exact=False)
tps1 = t2-t1
```

```
for i in range (1,len(semaine1)//2) :
        t1=pd.to_datetime(data.iloc[2*i][0],format= "%Y-%m-%d %H:%M:%S", exact=False)
        t2=pd.to_datetime(data.iloc[2*i+1][0],format= "%Y-%m-%d %H:%M:%S", exact=False)
        tps1 += t2-t1

t1=pd.to_datetime(data.iloc[semaine2[0]][0],format= "%Y-%m-%d %H:%M:%S", exact=False)
t2=pd.to_datetime(data.iloc[semaine2[0]+1][0],format= "%Y-%m-%d %H:%M:%S", exact=False)
tps2 = t2-t1

for i in range (1,len(semaine2)//2) :
        t1=pd.to_datetime(data.iloc[2*i+semaine2[0]][0],format= "%Y-%m-%d %H:%M:%S", exact=False)
        t2=pd.to_datetime(data.iloc[2*i+1+semaine2[0]][0],format= "%Y-%m-%d %H:%M:%S", exact=False)
        tps2 += t2-t1

t1=pd.to_datetime(data.iloc[semaine3[0]][0],format= "%Y-%m-%d %H:%M:%S", exact=False)
t2=pd.to_datetime(data.iloc[semaine3[0]+1][0],format= "%Y-%m-%d %H:%M:%S", exact=False)
tps3 = t2-t1

for i in range (1,len(semaine3)//2) :
        t1=pd.to_datetime(data.iloc[2*i+semaine3[0]][0],format= "%Y-%m-%d %H:%M:%S", exact=False)
        t2=pd.to_datetime(data.iloc[2*i+1+semaine3[0]][0],format= "%Y-%m-%d %H:%M:%S", exact=False)
        tps3 += t2-t1

#Etude de machine learning
size=int(len(data['time'])*0.2)
x_train, x_test, y_train, y_test = train_test_split(data[["mvt","ouv_porte"]], data[["lumiere"]], test_size=size,
random_state=0)

feature=x_train
```

```
clf = tree.DecisionTreeClassifier(random_state=0, criterion='entropy', max_depth=None)

clf.fit(feature, y_train)



plt.figure(dpi=700)

plot_tree(clf, max_depth=5, feature_names= ["mvt","ouv_porte"], class_names=["lumiere éteinte","lumiere
allumée"],filled=True)



y_pred = clf.predict(x_test)

print("\nThe accuracy is", accuracy_score(y_test, y_pred))

print("The error is", mean_squared_error(y_test, y_pred))

print("The f-score is", f1_score(y_test, y_pred, average='macro'),"\n")



#Scenario 2 avec seulement le capteur de mouvement



size=int(len(data['time'])*0.2)

x_train, x_test, y_train, y_test = train_test_split(data[["mvt"]], data[["lumiere"]], test_size=size, random_state=0)



feature=x_train

clf = tree.DecisionTreeClassifier(random_state=0, criterion='entropy', max_depth=None)

clf.fit(feature, y_train)



plt.figure(dpi=700)

plot_tree(clf, max_depth=5, feature_names= ["mvt"], class_names=["lumiere éteinte","lumiere
allumée"],filled=True)



y_pred = clf.predict(x_test)

print("\nThe accuracy is", accuracy_score(y_test, y_pred))

print("The error is", mean_squared_error(y_test, y_pred))

print("The f-score is", f1_score(y_test, y_pred, average='macro'),"\n")
```

```
#Scenario 3 avec seulement l'ouverture de la porte


size=int(len(data['time'])*0.2)

x_train, x_test, y_train, y_test = train_test_split(data[["ouv_porte"]], data[["lumiere"]], test_size=size,
random_state=0)


feature=x_train

clf = tree.DecisionTreeClassifier(random_state=0, criterion='entropy', max_depth=None)

clf.fit(feature, y_train)


plt.figure(dpi=700)

plot_tree(clf, max_depth=5, feature_names= ["ouv_porte"], class_names=["lumiere éteinte","lumiere
allumée"],filled=True)


y_pred = clf.predict(x_test)

print("\nThe accuracy is", accuracy_score(y_test, y_pred))

print("The error is", mean_squared_error(y_test, y_pred))

print("The f-score is", f1_score(y_test, y_pred, average='macro'),"\n")
```

Code pour Maching learning XAI sur jupiter :

```
from math import *

from numpy import *

import numpy as np

from sklearn.model_selection import train_test_split

from sklearn import tree

from sklearn.cluster import KMeans

import matplotlib.pyplot as plt

import xlrd

import pandas as pd
```

```
data = pd.read_csv('datadivi.csv', sep =';',index_col='time')

print(data.columns)

data.fillna(0)

x_train, x_test, y_train, y_test = train_test_split(data[["lumino","ouv_porte","mvt","abs_mvt"]],
data[["label"]], test_size=0.5, random_state=0)

from sklearn.cluster import KMeans

kmeans_model = KMeans(n_clusters=4, random_state=0).fit(x_train)

labelpredict = kmeans_model.predict(x_test)

print(len(labelpredict))

clf = tree.DecisionTreeClassifier(criterion='entropy')

clf = clf.fit(x_train,labelpredict)

tree.plot_tree(clf)

y_predict=clf.predict(x_test)


with open("tree.dot", 'w') as file:

 f = tree.export_graphviz(clf, out_file=file)

from IPython.display import Image

Image(url= "capteurimportant.png", width=2000, height=2000)

from sklearn.metrics import precision_recall_fscore_support

accuracy_xai=precision_recall_fscore_support(y_test,y_predict , average='weighted')

print("accuracy:{a:.4},recall:{b:.5} ,fscore:{c:.5}:".format(a=accuracy_xai[0],
b=accuracy_xai[1],c=accuracy_xai[2]))


from sklearn.metrics import mean_absolute_error

error_xai=mean_absolute_error(y_test,y_predict)

print("mean absolute error is :",error_xai)
```