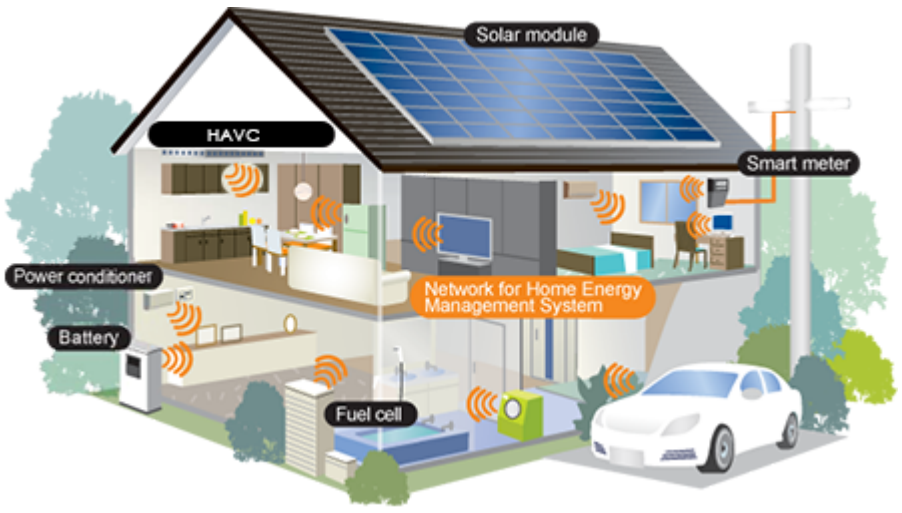
Thibaud PELOPS
Tom LAMBERT
Céleste CHAMBON

Projet 11: Gestion d'énergie

Encadrants :
Manar AMAYRI
Jérôme FERRARI

Thibaud PELOPS
Tom LAMBERT
Céleste CHAMBON

SOMMAIRE



I. Introduction

L'habitat et le secteur du bâtiment en général, est fortement consommateur d'énergie. La consommation d'énergie dans les habitations représente 43 % de la consommation totale d'énergie en France et 25 % des émissions de gaz à effet de serre. L'énergie est destinée au chauffage, à l'éclairage, à la climatisation, à l'utilisation de l'eau chaude pour la toilette et la cuisine à l'utilisation des appareils électriques (réfrigérateur, télévision, four, ordinateur etc.)

II. Problématique

Le projet Expe-smart house a été initié en 2018 afin de fournir les données d'un foyer de 120 m² où vit une famille de 5 personnes. Ce projet donne accès à environ 340 points de mesure pour les scientifiques, accessibles en temps réel via un portail Grafana alimenté avec une base de données Influxdb. Cette maison intelligente est développée sur la base de matériel et de logiciels Open Source. Le système est également flexible en ce qui concerne l'ajout de nouvelles technologies ou de nouveaux capteurs. Les mesures remontées sont :

- Consommation d'électricité, de gaz et d'eau de chaque appareil
- Température, humidité et luminosité de chaque pièce
- Position d'ouverture de chaque porte et fenêtre
- Détecteurs de mouvement
- État de l'éclairage
- Analyse de l'air de chaque pièce
- Conditions météorologiques extérieures

Le problème que nous nous sommes mis en tête est de résoudre l'optimisation d'une pièce de cette maison en fonction de la météo à l'aide de différentes données. Voici un plan de la maison sur laquelle nous allons travailler.

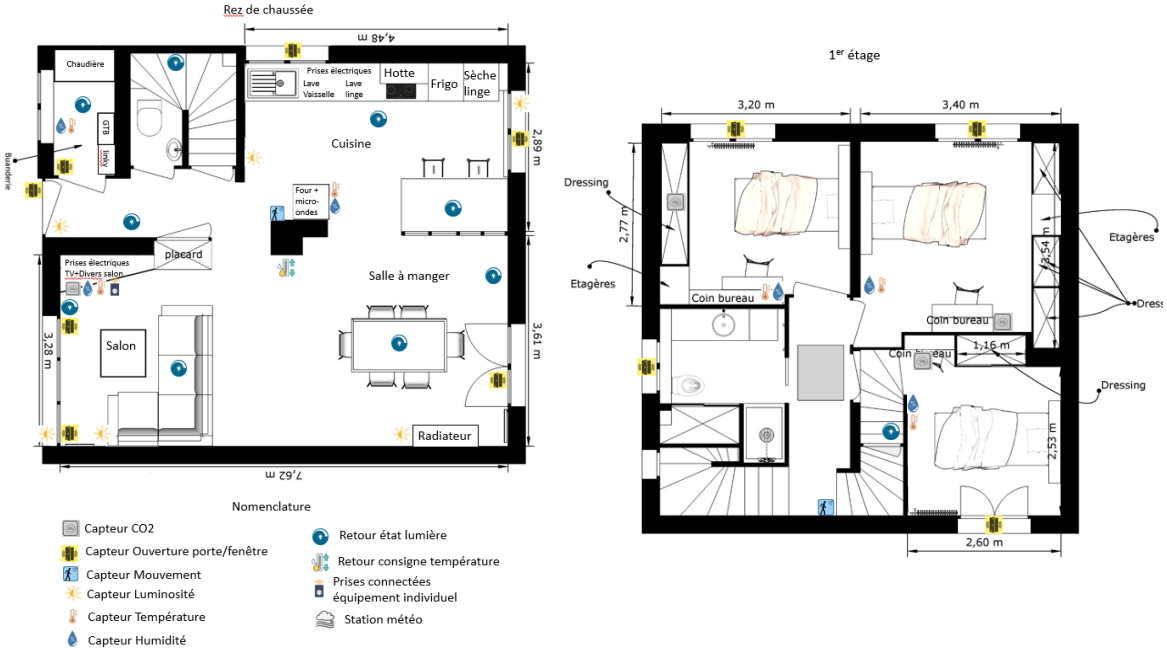


Figure 1: plan de la maison

III. Notre méthode : Machine Learning

Dans un premier temps, il a fallu apprendre le machine learning. On nous a donc appris à utiliser différentes bibliothèques sur python comme scikit-learn qui permet la création d'un arbre de décision. Tout notre apprentissage s'est effectué sur un dossier de données fictif qui était fictif. De ce fait, les données que nous avions n'étaient pas continues.
Dans notre cas réel de maison cela va être différent car elles seront réelles continues.
Par le biais des capteurs répartis dans la maison nous recevons toutes les données dont nous avons besoin pour créer notre algorithme. Il y en a une certaine multiplicité : température,CO2,luminosité,humidité,mouvement,ouverture porte/fenêtre etc. Celles qui vont nous intéresser sont les capteurs :

- de température extérieur
- pour la pluie
- de luminosité
- pour la fenêtre
- température intérieur (qui ici nous servira de "label")

En fonction des saisons la météo sera changeante et la température et le taux d'humidité le seront aussi en conséquence. Notre étude repose sur quel capteur permet au mieux de prédire la température de la chambre afin de limiter l'utilisation multiple de capteurs et pour voir quel élément extérieur influe le plus sur la température intérieure. Nous allons donc utiliser tout ce que nous avons appris et notamment la fonction "feature importance". Par la suite, on

pourrait vouloir réguler la température et le taux de CO2 de la chambre grâce aux fenêtres mais pour ce faire il faudrait qu'elles soient automatisées.

1. Extraction des données.

En voulant extraire nos données, nous nous sommes rendus compte que nous étions trop ambitieux. Nous voulions étudier toute la maison ce qui représentait plus d'une vingtaine de capteurs avec des données manquantes, des plages horaires et des durées différentes. Nous avons décidé de nous concentrer sur l'étude que d'une seule chambre, afin de voir les possibilités d'étude qui s'offriraient à nous. Si nous arrivons à mener à bien notre projet, il sera plus simple ensuite de généraliser à plusieurs chambres voir plusieurs pièces différentes.

La première difficulté que nous avons rencontré est le fait que les données n'étaient pas sur les même plage temporelles et le même pas. Il nous était donc impossible de les utiliser de la sorte car toute comparaison n'aurait pas de sens. Nos encadrants ont réussi à régler ce problème. Nous avons ensuite choisi de travailler sur toute la plage de données disponible (c'est-à-dire toute la plage ou nous avions l'entièreté des données du 01/01/2022 8h15 au 20/05/2022 7h05).

Pour commencer notre étude, nous avons placé toutes nos données sur le même excel avec le même pas de temps puis convertir ce fichier en fichier texte pour enfin extraire nos données avec python comme appris en cours:

```
###ouvrir le fichier et avoir les données
Text , label , fenetre , pluie , Lumext =[],[],[],[],[]
fichier=open('Z:/projet Gestion Energie/mini projet/donneefinal.txt','r')
titre=fichier.readline()
for line in fichier:
    ligne_choisie=line.split("\t") #\t=tabulation
    pluie.append(float(ligne_choisie[3]))
    Text.append(float(ligne_choisie[5][0:3]))
    label.append(float(ligne_choisie[4]))
    fenetre.append(float(ligne_choisie[1]))
    Lumext.append(float(ligne_choisie[2]))
fichier.close()
```

Figure 2: Code nécessaire pour extraire les données

2. Création d'un premier arbre de décisions.

Nous avons voulu ensuite commencer à analyser ces données, le principe était simple: quel(s) capteur(s) entre la pluie, la luminosité, la température extérieure, ou l'ouverture de la fenêtre pouvait influencer le plus la température intérieure prise ici en tant que label. Pour ce faire, nous voulions utiliser Decision Tree Classifier afin de faire un arbre de décision pour commencer notre étude. Cependant, nous n'avons pas pu utiliser cette technique car nous utilisons comme "label" des valeurs continues et non discrètes. Nous avons été obligés d'utiliser **Decision Tree Regressor** qui fonctionne sur le même principe que classifier mais pour les données continues. Ensuite nous avons rencontré un problème pour afficher l'arbre de décision car la plage de données choisie était trop importante. Nous avons décidé de nous

concentrer uniquement sur le mois d'avril qui présentait à la fois des températures froides et chaudes.

L'arbre ne peut pas être créé car il n'y a pas assez de mémoire. Il est alors nécessaire de diminuer la profondeur de l'arbre à 3. Nous obtenons finalement le code suivant:

```
###creer arbre de decision
N=int(len(label)*0.7) #au debut on prend 70% pour creer l'arbre et 30% pour le tester
X=[]
Y=[]
for i in range(N):
    X =X + [[Text[i],fenetre[i],pluie[i],Lumext[i]]]
    Y =Y + [[label[i]]]
classifier = tree.DecisionTreeRegressor(random_state=0,max_depth=(3))
classifier.fit(X, Y)
fichier2=open('resultats.txt','w')
f = tree.export_graphviz(classifier, out_file=fichier2)
fichier2.close()

###prend les parametres et on cherche la temp interieur avec notre modele.
M=len(label)
Xnew=[]
for i in range(N,M):
    Xnew=Xnew + [[Text[i],fenetre[i],pluie[i],Lumext[i]]]
Ymod=classifier.predict(Xnew)
Yref=[]
for i in range(N,M):#echantillon 30% des données de Y pour comparer
    Yref =Yref + [[label[i]]] # labelling data
```

Figure 3:code utile pour créer l'arbre de décision

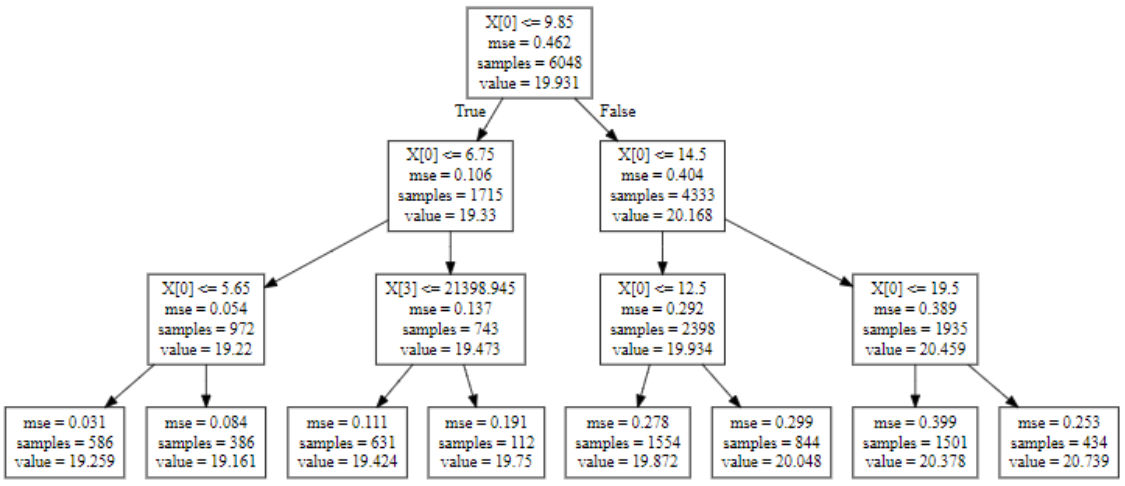


Figure 4:1e arbre de décisions que nous obtenons

Règles de l'arbre :

(1) - Text <= 5.65° ⇒ Tint = 19.259°
(2) - 5.65°<Text<= 6.75° ⇒ Tint = 19.161°
(3) - 6.75°<Text<= 9.85° et Lumext <= 21398.945 lux ⇒ Tint = 19.424°
(4) - 6.75°<Text<= 9.85° et Lumext > 21398.945 lux ⇒ Tint = 19.75°
(5) - 9.85°<Text<= 12.5° ⇒ Tint = 19.872°
(6) - 12.5°<Text<= 14.5° ⇒ Tint = 20.048°

(7) - 14.5°<Text<= 19.5° ⇒ Tint = 20.378°
(8) - Text > 19.5° ⇒ Tint = 20.739°

3. Test de différentes proportions pour le "build model".

On cherche ensuite à déterminer comment séparer les données qui serviront à construire l'arbre et les données à prédire pour tester notre modèle. Pour ça on calcule l'accuracy et l'average error pour chaque proportions possible. Mais cette fois nous sommes encore en présence de valeurs continues (température intérieur) donc nous ne pouvons pas exactement faire ce que nous faisions avec des valeurs continues. Pour analyser l'efficacité, nous effectuons la méthode suivante : on prend la température intérieure réelle et celle que nous avons prédit grâce au modèle puis on les compare sur un graphe pour analyser la proportion de "testing data" et "training data".
On a donc affiché chaque graphes pour différentes proportions. On s'est rendu compte qu'on ne pouvait pas réellement utiliser ses données car la plage de temps est trop longue. On aimerait donc se ramener à une température moyenne par jour. Une itération correspondant à 5 min.

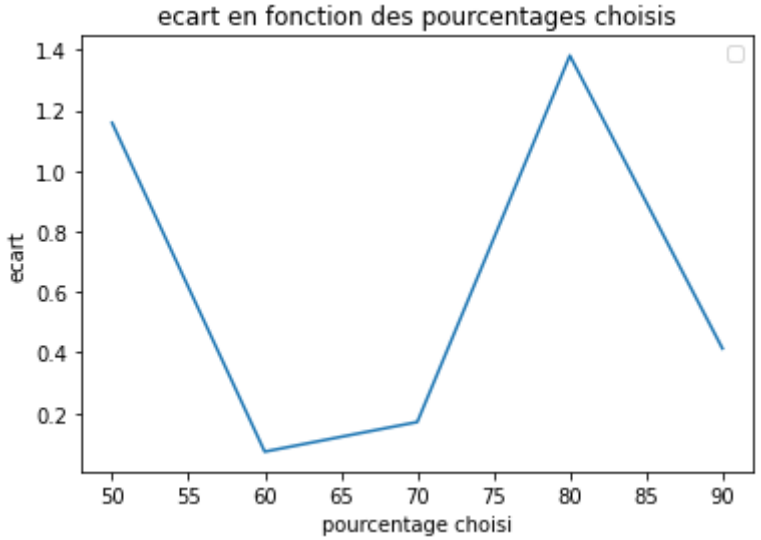


Figure 5:Graphe des écarts pour chaque %

L'écart est le plus faible pour **une proportion de 60%** pour les training data nous allons donc travailler avec cette proportion.

4. Utilisation de "feature importance".

On utilise actuellement quatre capteurs pour prédire la température intérieure de la chambre, pour cela on essaie de savoir l'importance des capteurs. On a donc utilisé la fonction "features importance" de la bibliothèque sklearn.tree pour savoir les capteurs à utiliser dans une optique d'économie.

```
#feature importances:
N=int(len(label)*0.6)
X=[]
Y=[]
for i in range(N):
    X =X + [[Text[i],fenetre[i],pluie[i],Lumext[i]]]
    Y =Y + [[label[i]]]
classifier = tree.DecisionTreeRegressor(random_state=0)
classifier.fit(X, Y)
importance = classifier.feature_importances_
importance=np.sort(importance)
# summarize feature importance
for i,v in enumerate(importance):
    print('Feature: {}, Score: {}'.format(i,v))
# plot feature importance
sorted_indices = np.argsort(importance)[::-1]
plt.bar([x for x in range(len(importance))], importance)
plt.title('importance des capteurs')
noms = ['Text','fenetre','pluie','Lumext']
Xv = pd.DataFrame(X, columns=noms)
a=np.shape(Xv)
plt.xticks(range(a[1]), Xv.columns[sorted_indices], rotation=90)
plt.show()
```

Figure 6 : Code permettant de déterminer les capteurs les plus importants

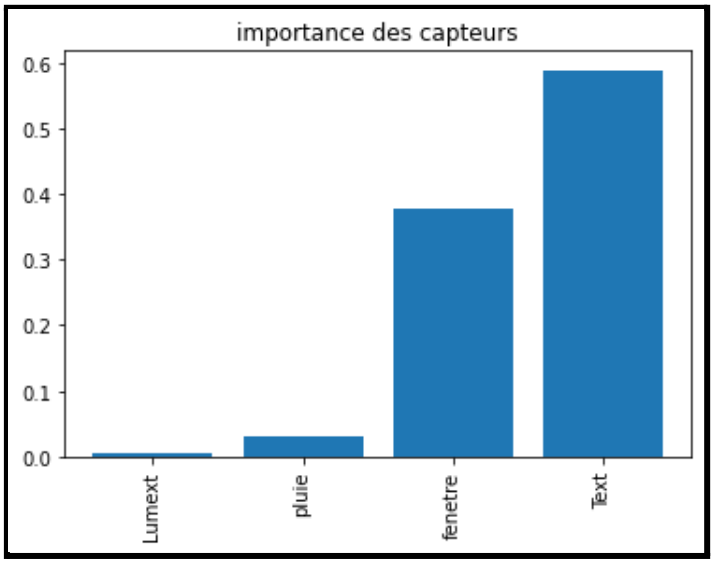


Figure 7: Importance de chaque capteurs

On voit donc que c'est le capteur de température extérieur qui est le plus important (environ 60%) on voit aussi que le capteur de la fenêtre l'est (environ 40%).En effet ce sont ceux qui sont vraiment utiles, les deux autres étant négligeables (moins de 5%).

On essaie d'avoir la profondeur de l'arbre (nombre de rangs) la plus optimale en fonction du nombre de capteur choisi. Pour cela nous avons dissocié cette étude en trois cas :

- Avec tous les capteurs (comme précédemment sans utiliser l'importance des capteurs)
- Avec les deux capteurs les plus important (Text et fenêtre)
- Seulement le capteur Text (étant le capteur avec le plus d'importance d'après le code utilisant features importance).

```
#changement de profondeur (analyse):
    ecart=[]
    prof=[]
    for m in range (1,12):
        N=int(len(label)*0.6)
        X=[]
        Y=[]
        for i in range(N):
            X =X + [[Text[i]]] #a changer pour avoir les autres profodeurs qu'on a mis dans le CR
            Y =Y + [[label[i]]]
        classifier = tree.DecisionTreeRegressor(random_state=0,max_depth=(m))
        classifier.fit(X, Y)
        M=len(label)
        Xnew=[]
        for i in range(N,M):
            Xnew=Xnew + [[Text[i]]] #a changer pour avoir les autres profodeurs qu'on a mis dans le CR
        Ymod=classifier.predict(Xnew)
        Yref=[]
        for i in range(N,M):
            Yref =Yref + [[label[i]]]
        ##on va faire une moyenne sur la journée pour mieux visualiser nos données:
        Tref=[]
        Tmod=[]
        somme=0
        J=288
        for j in range(31):
            sommeref=0
            sommemod=0
            for i in range(j,J+j):#nombre de données en une journée
                sommeref=sommeref + Yref[i][0]
                sommemod=sommemod + Ymod[i]
            Tref= Tref +[[sommeref/J]]
            Tmod= Tmod +[[sommemod/J]]
            somme+= abs((sommeref/J)-(sommemod/J))
        ecart = ecart + [[somme/30]]
        prof+= [m]

    plt.figure( )
    plt.plot(prof,ecart)
    plt.xlabel("profondeur choisie")
    plt.ylabel("ecart reel vs predit")
    plt.title("evolution de l'ecart en fonction de la profondeur")
    plt.show()
```

Figure 8 : Code permettant d'évaluer la profondeur

5. Optimisation de la profondeur

a. *Avec tous les capteurs:*

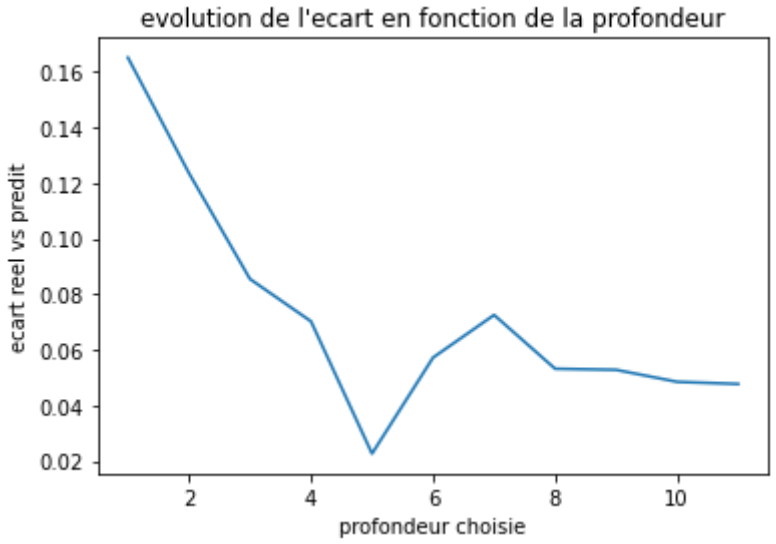


Figure 9 : graphique de l'écart entre la température réelle et estimée pour une profondeur d'arbre différente (prise avec tous les capteurs)

On relève sur ce graphique que la profondeur optimale avec utilisation de tous les capteurs est 5.

b. Avec les capteurs Text et fenêtre:

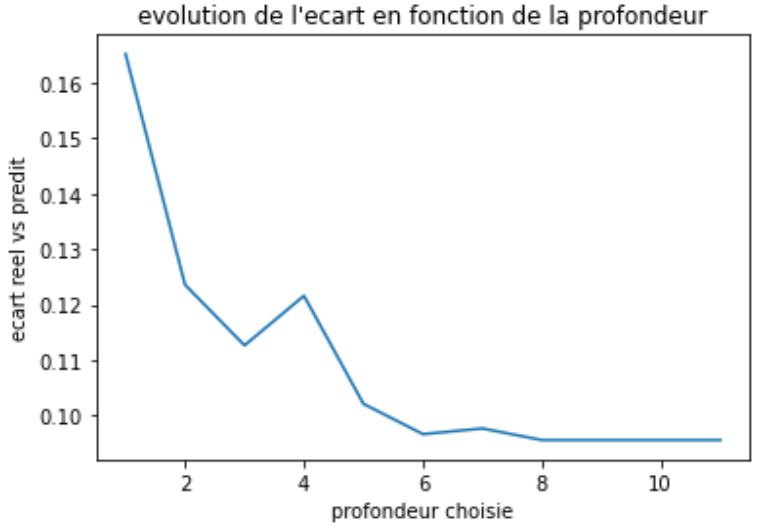


Figure 10 : graphique de l'écart entre la température réelle et estimée pour une profondeur d'arbre différente (prise avec Text et fenetre)

On relève sur ce graphique que la profondeur optimale avec utilisation des capteurs Text et fenêtre est 6.

c. Avec le capteur Text:

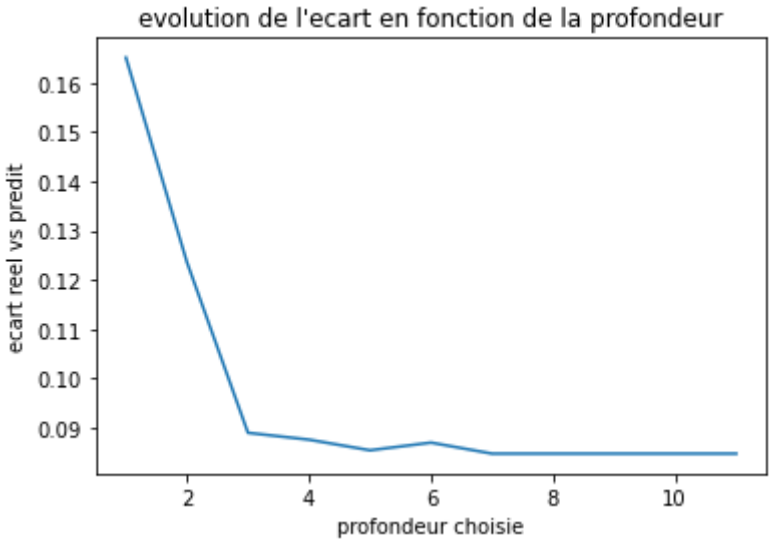


Figure 11 : graphique de l'écart entre la température réelle et estimée pour une profondeur d'arbre différente (prise avec Text)

On relève sur ce graphes que la profondeur optimale avec utilisation du capteur Text seulement est 3.

Nous avons donc réalisé des graphes pour déduire la profondeur à choisir pour la suite du graphe. Le but étant d'avoir une profondeur minimale, le mieux serait de garder seulement le

capteur Text (profondeur égale à 3). Pour la suite de cette analyse nous utiliserons donc le capteur de température extérieur avec une profondeur de 3.

On commence par faire de nouveau un arbre de décision, on a en effet rechercher la profondeur pour faire un arbre avec plus de précision.

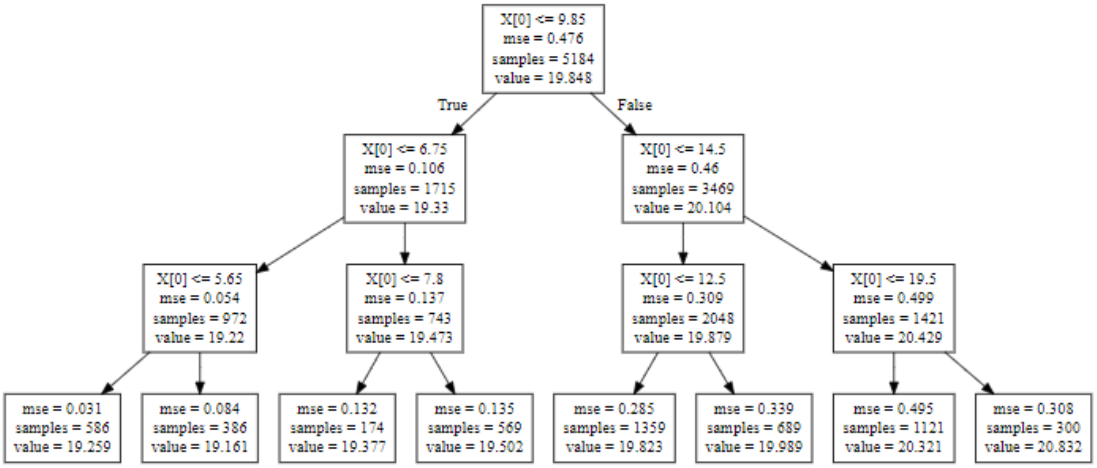


Figure 12 : arbre de décision avec notre système amélioré

Règles de l'arbre :

(1) - Text <= 5.65° ⇒ Tint = 19.259°
(2) - 5.65°<Text<= 6.75° ⇒ Tint = 19.161°
(3) - 6.75°<Text<= 7.8° ⇒ Tint = 19.377°
(4) - 7.8°<Text<= 9.85° ⇒ Tint = 19.502°
(5) - 9.85°<Text<= 12.5° ⇒ Tint = 19.823°
(6) - 12.5°<Text<= 14.5° ⇒ Tint = 19.989°
(7) - 14.5°<Text<= 19.5° ⇒ Tint = 20.321°
(8) - Text > 19.5° ⇒ Tint = 20.832°

On va maintenant comparer l'erreur (température intérieur réelle de la chambre et température intérieur estimée avec les capteurs dans la chambre) que l'on avait au début de cette étude en utilisant tous les capteurs et une profondeur choisie aléatoirement (au moins pour qu'on puisse afficher l'arbre) avec l'erreur que l'on obtient maintenant avec utilisation du capteur Text et une profondeur de 3.

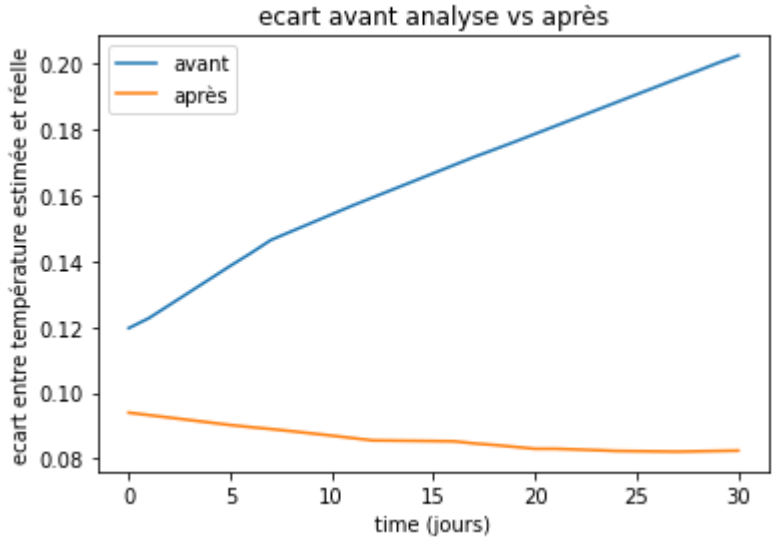


Figure 13 : graphe comparant l'arbre de décision du début de l'étude a celui de la fin

Le choix que nous avons fait de choisir seulement le capteur de température extérieure et une profondeur de 3 est pertinent, en effet on remarque bien que sur le graphe précédent l'écart est plus faible.

Conclusion:

Le but de ce projet était de réaliser une étude sur une maison automatisée, intelligente. Dans le cadre de notre cours de gestion d'énergie nous avons appliqué les connaissances apprises sur l'exploitation de données utilisées dans ces maisons. Nous nous sommes concentrés sur une pièce d'une maison pendant une période d'un mois. Nous avons eu de bons résultats et nous en avons déduit que l'utilisation de tous les capteurs n'était pas forcément nécessaire. Dans une optique de généraliser ce processus,pour rendre accessible au grand public cette méthode d'automatisation, il faudrait que le coût global de l'installation soit moins important et pour ce faire que choisir quel capteur est important ou non nous paraît cohérent.

Cependant en portant un regard critique sur ce projet nous avons réduit l'étude à une seule pièce située à l'étage pendant un mois et il y a beaucoup de variables qui peuvent changer en fonction des mois et des saisons, de l'orientation et la localisation de la maison. En effet, certains mois montreront peut être l'utilité des autres capteurs car il y aurait plus de pluie, il ferait plus froid, plus de vent, plus de luminosité...

V.ANNEXE: le code

```
from sklearn import tree
from sklearn.tree import DecisionTreeRegressor
import matplotlib.pyplot as plt
from math import *
import numpy as np
import pandas as pd
from matplotlib import pyplot


###utiliser les differentes partie du code:
k=4 #k=0: on a l'arbe de base
   #k=1: combien on va prendre de données pour tester et pour creer l'arbre
   #k=2: importance des capteurs dans la prédiction de la température
   #k=3: choisir la profondeur de notre arbre
   #k=4:arbre final et comparaison finale

###ouvrir le fichier et avoir les données
Text , label , fenetre , pluie , Lumext =[],[],[],[],[]
fichier=open('Z:/projet Gestion Energie/mini projet/donneefinal.txt','r')
titre=fichier.readline()
for line in fichier:
   ligne_choisie=line.split("\t") #\t=tabulation
   pluie.append(float(ligne_choisie[3]))
   Text.append(float(ligne_choisie[5][0:3]))
   label.append(float(ligne_choisie[4]))
   fenetre.append(float(ligne_choisie[1]))
   Lumext.append(float(ligne_choisie[2]))
fichier.close()



if k==0:
   ###creer arbre de decision
   N=int(len(label)*0.7) #au debut on prend 70% pour creer l'arbre et 30% pour le tester
   X=[]
   Y=[]
   for i in range(N):
      X =X + [[Text[i],fenetre[i],pluie[i],Lumext[i]]]
      Y =Y + [[label[i]]]
   classifier = tree.DecisionTreeRegressor(random_state=0,max_depth=(3))
   classifier.fit(X, Y)
   fichier2=open('resultats.txt','w')
   f = tree.export_graphviz(classifier, out_file=fichier2)
   fichier2.close()

   ###prend les parametres et on cherche la temp interieur avec notre modele.
```

```
    M=len(label)
    Xnew=[]
    for i in range(N,M):
        Xnew=Xnew + [[Text[i],fenetre[i],pluie[i],Lumext[i]]]
    Ymod=classifier.predict(Xnew)
    Yref=[]
    for i in range(N,M):#echantillon 30% des données de Y pour comparer
        Yref =Yref + [[label[i]]] # labelling data

if k==1:
    H=[]
    ecart=[]
    for h in range(50,100,10):
        H= H+[[h]]
        N=int(len(label)*(h/100))
        X=[]
        Y=[]
        for i in range(N):
            X =X + [[Text[i],fenetre[i],pluie[i],Lumext[i]]]
            Y =Y + [[label[i]]]
        classifier = tree.DecisionTreeRegressor(random_state=0)
        classifier.fit(X, Y)
        M=len(label)
        Xnew=[]
        for i in range(N,M):
            Xnew=Xnew + [[Text[i],fenetre[i],pluie[i],Lumext[i]]]
        Ymod=classifier.predict(Xnew)
        Yref=[]
        for i in range(N,M):
            Yref =Yref + [[label[i]]]
        ##on va faire une moyenne sur la journée pour mieux visualiser nos données:
        Tref=[]
        Tmod=[]
        somme=0
        J=288
        for j in range(31):
            sommeref=0
            sommemod=0
            for i in range(j,J+j):#nombre de données en une journée
                sommeref=sommeref + Yref[i][0]
                sommemod=sommemod + Ymod[i]
            Tref= Tref +[[sommeref/J]]
            Tmod= Tmod +[[sommemod/J]]
            somme+= abs((sommeref/J)-(sommemod/J))
        ecart = ecart + [[somme/30]]
    plt.figure()
    plt.plot(H,ecart)
    plt.xlabel("pourcentage choisi")
    plt.ylabel("ecart")
```

```
    plt.title("ecart en fonction des pourcentages choisis")
    plt.show()
    plt.figure()
    ##avec cette etude on en deduis qu'il faut prendre 60% pour tester afin d'avoir les meilleurs
resultats


if k==2:
    #feature importances:
    N=int(len(label)*0.6)
    X=[]
    Y=[]
    for i in range(N):
        X =X + [[Text[i],fenetre[i],pluie[i],Lumext[i]]]
        Y =Y + [[label[i]]]
    classifier = tree.DecisionTreeRegressor(random_state=0)
    classifier.fit(X, Y)
    importance = classifier.feature_importances_
    importance=np.sort(importance)
    # summarize feature importance
    for i,v in enumerate(importance):
        print('Feature: {}, Score: {}'.format(i,v))
    # plot feature importance
    sorted_indices = np.argsort(importance)[::-1]
    plt.bar([x for x in range(len(importance))], importance)
    plt.title('importance des capteurs')
    noms = ['Text','fenetre','pluie','Lumext']
    Xv = pd.DataFrame(X, columns=noms)
    a=np.shape(Xv)
    plt.xticks(range(a[1]), Xv.columns[sorted_indices], rotation=90)
    plt.show()

if k==3:
#changement de profondeur (analyse):
    ecart=[]
    prof=[]
    for m in range (1,12):
        N=int(len(label)*0.6)
        X=[]
        Y=[]
        for i in range(N):
            X =X + [[Text[i]]] #a changer pour avoir les autres profodeurs qu'on a mis dans le CR
            Y =Y + [[label[i]]]
        classifier = tree.DecisionTreeRegressor(random_state=0,max_depth=(m))
        classifier.fit(X, Y)
        M=len(label)
        Xnew=[]
        for i in range(N,M):
```

```
        Xnew=Xnew + [[Text[i]]] #a changer pour avoir les autres profodeurs qu'on a mis
dans le CR
      Ymod=classifier.predict(Xnew)
      Yref=[]
      for i in range(N,M):
        Yref =Yref + [[label[i]]]
      ##on va faire une moyenne sur la journée pour mieux visualiser nos données:
      Tref=[]
      Tmod=[]
      somme=0
      J=288
      for j in range(31):
        sommeref=0
        sommemod=0
        for i in range(j,J+j):#nombre de données en une journée
          sommeref=sommeref + Yref[i][0]
          sommemod=sommemod + Ymod[i]
        Tref= Tref +[[sommeref/J]]
        Tmod= Tmod +[[sommemod/J]]
        somme+= abs((sommeref/J)-(sommemod/J))
      ecart = ecart + [[somme/30]]
      prof+= [m]

  plt.figure( )
  plt.plot(prof,ecart)
  plt.xlabel("profondeur choisie")
  plt.ylabel("ecart reel vs predit")
  plt.title("evolution de l'ecart en fonction de la profondeur")
  plt.show()

if k==4:
#arbre de decision final
  N2=int(len(label)*0.6)
  X2=[]
  Y2=[]
  Tref2=[]
  Tmod2=[]
  ecart2=[]
  jours=[]
  for i in range(N2):
    X2 =X2 + [[Text[i]]]
    Y2 =Y2 + [[label[i]]]
  classifier = tree.DecisionTreeRegressor(random_state=0,max_depth=(3))
  classifier.fit(X2, Y2)
  fichier2=open('resultats.txt','w')
  f = tree.export_graphviz(classifier, out_file=fichier2)
  fichier2.close()
  M2=len(label)
  Xnew=[]
```

```
    for i in range(N2,M2):
        Xnew=Xnew + [[Text[i]]]
    Ymod=classifier.predict(Xnew)
    Yref=[]
    for i in range(N2,M2):
        Yref =Yref + [[label[i]]]
    J=288
    for j in range(31):
        sommeref=0
        sommemod=0
        jours=jours+[j]
        for i in range(j,J+j):#nombre de données en une journée
            sommeref=sommeref + Yref[i][0]
            sommemod=sommemod + Ymod[i]
        Tref2= Tref2 +[sommeref/J]
        Tmod2= Tmod2 +[sommemod/J]
        ecart2=ecart2+[Tref2[j]-Tmod2[j]]

    ##au début:
    N=int(len(label)*0.7)
    X=[]
    Y=[]
    ecart=[]
    for i in range(N):
        X =X + [[Text[i],fenetre[i],pluie[i],Lumext[i]]]
        Y =Y + [[label[i]]]
    classifier = tree.DecisionTreeRegressor(random_state=0)
    classifier.fit(X, Y)
    M=len(label)
    Xnew=[]
    for i in range(N,M):
        Xnew=Xnew + [[Text[i],fenetre[i],pluie[i],Lumext[i]]]
    Ymod=classifier.predict(Xnew)
    Yref=[]
    for i in range(N,M):#echantillon 30% des données de Y pour comparer
        Yref =Yref + [[label[i]]] # labelling data
    Tref=[]
    Tmod=[]
    J=288
    for j in range(31):
        sommeref=0
        sommemod=0
        for i in range(j,J+j):#nombre de données en une journée
            sommeref=sommeref + Yref[i][0]
            sommemod=sommemod + Ymod[i]
        Tref= Tref +[sommeref/J]
        Tmod= Tmod +[sommemod/J]
        ecart = ecart + [abs(Tref[j]-Tmod[j])]
```

```
plt.figure(1)
plt.plot(jours,ecart,label="avant")
plt.plot(jours,ecart2,label="après")
plt.xlabel("time (jours)")
plt.ylabel("ecart entre température estimée et réelle")
plt.title("ecart avant analyse vs après")
plt.legend()
plt.show(1)
```