Melchior
CHAILLOUX

Floryan
VERDONCK

05/06/2022

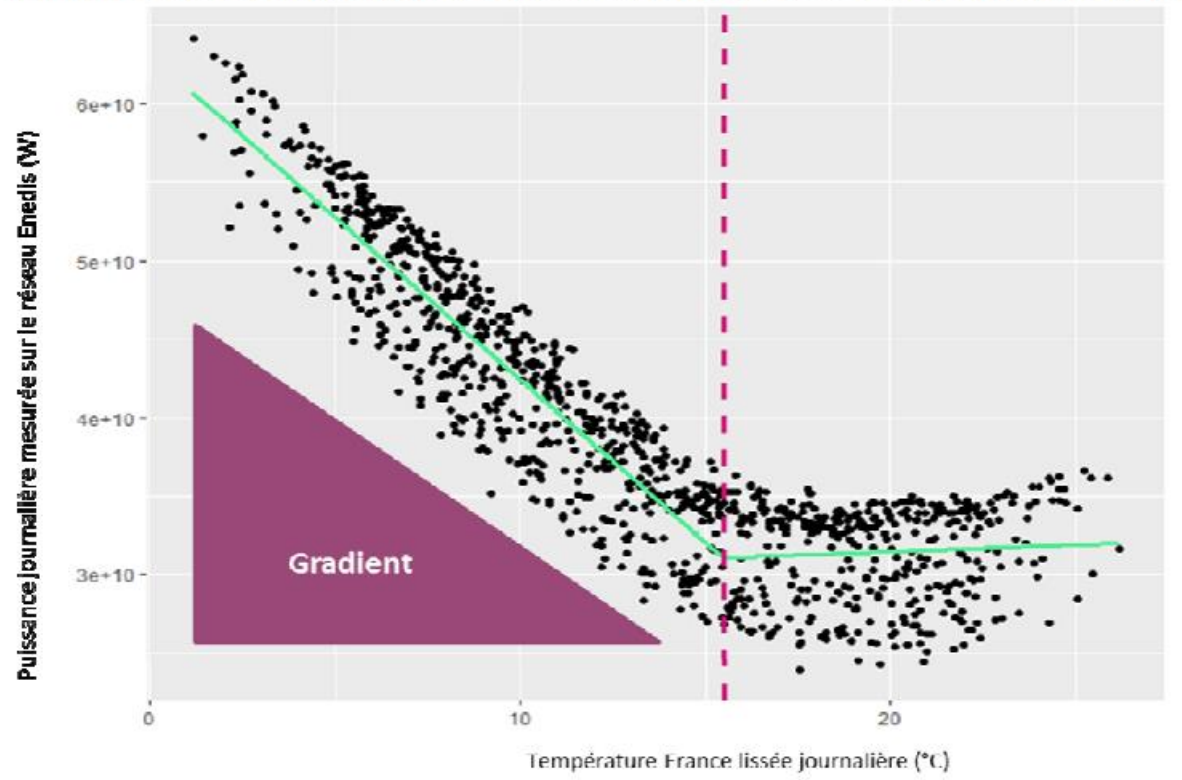
Gestion énergétique : rendu final de projet

Table des matières

Introduction

Depuis l'essor de l'informatique il y a une trentaine d'années, il s'est posé l'enjeu de la stabilisation de la fréquence des réseaux électriques pour éviter les micro-coupures en cas de saturation du réseau. Pour cela, il a fallu quantifier plus précisément et observer l'évolution annuelle de la consommation d'électricité des foyers afin d'anticiper les besoins en énergie électrique, éviter les saturations du réseau et prévenir les coupures.
On sait que la consommation d'énergie dépend de la température, comme le montre la source [1] : « L'observation des données de consommation et de température met en évidence qu'il existe une relation quasi linéaire entre la consommation et la température en dessous d'un certain seuil de température. Le graphique ci-dessous présente la puissance appelée journalière moyenne aux bornes du réseau ENDIS (en W) en fonction de la température France lissée journalière moyenne. ».

Graphe 1 : Puissance appelée en fonction de la température : exemple pour l'electricité

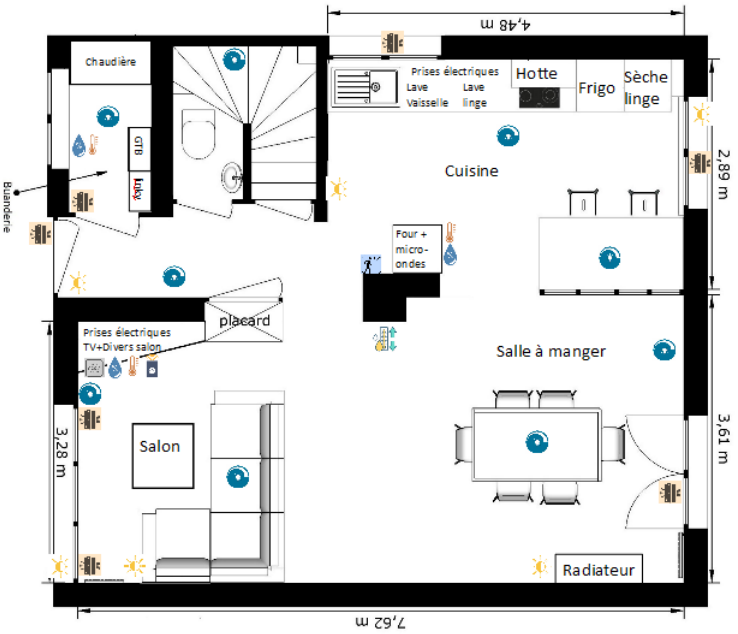


Nous avons exploité les données issues de l'expérience de Expe-Smarthouse. La maison étudiée est mitoyenne, de 3 étages avec un jardin et un garage. Elle comporte 3 chambres, 2 salles de bain, 2 toilettes, une cuisine, un salon et une salle à manger et est habitée par une famille de 5 personnes. Les données sont issues des relevés du compteur électrique et de la station météorologique, installée en extérieur.

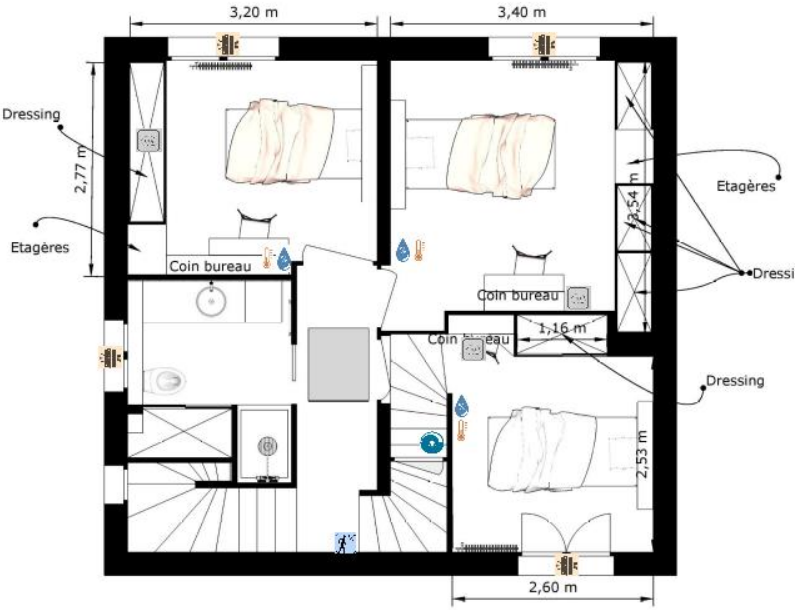
L'objectif de cette expérience est de permettre l'optimisation de la consommation énergétique des maisons de particuliers en estimant l'évolution annuelle de la consommation quotidienne en fonction des températures, de l'humidité, l'hydrométrie, du vent et de la pression atmosphérique.

Plan de la maison :

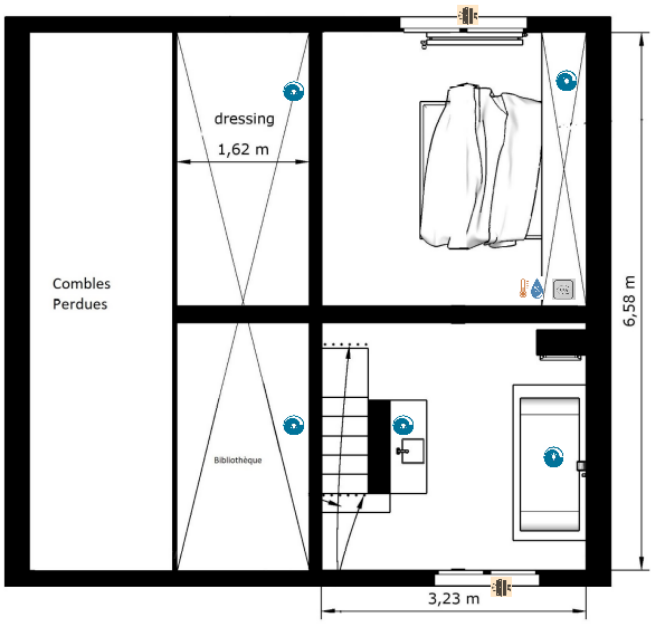
Rez de chaussée



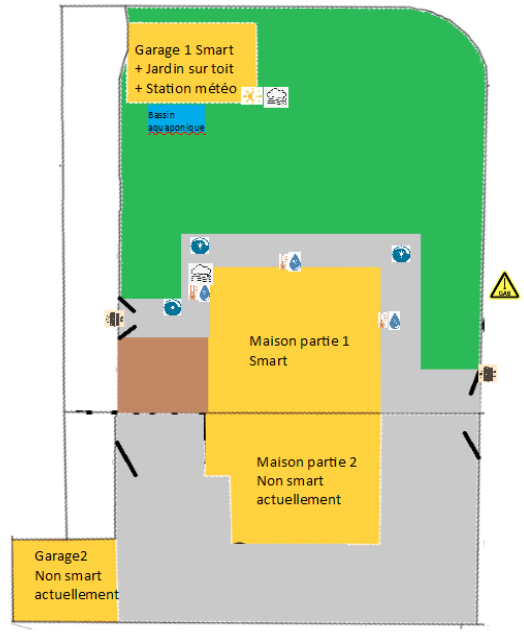
1er étage



2ème étage combles



Extérieur



Nomenclature

- Capteur CO2
- Capteur Ouverture porte/fenêtre
- Capteur Mouvement
- Capteur Luminosité
- Capteur Température
- Capteur Humidité
- Retour état lumière
- Retour consigne température
- Prises connectées équipement individuel
- Station météo
- Compteur gaz

Méthode d'analyse des données

Pour analyser les données, nous allons faire une extraction de données sur python grâce aux modules http.server, urllib, time, influxdb, sys, datetime et csv. Cela nous permet d'importer les données mais celles-ci ne sont alors pas au bon format, il faut donc traiter ces données pour les ré-échantillonner sur une période de temps unique. En effet, les capteurs météorologiques n'enregistrent que des valeurs lorsqu'il y a une variation significative de la variable observée. Pour ce faire, nous allons utiliser le module panda pour moyenner les résultats enregistrés sur 24h, et ainsi cette moyenne journalière obtenu est au bon format pour l'utilisation dans un script python,

En ce qui concerne la prédiction de la consommation d'électricité de la maison en fonction des conditions extérieures, je vais utiliser différentes méthodes de prédiction : l'arbre de décision de sklearn.tree, la forêt aléatoire de sklearn.ensemble et l'analyse par cluster de sklearn.cluster.

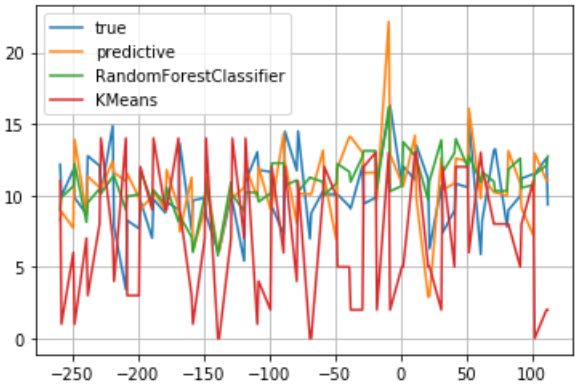


Figure 1 : exemple de prédiction
Profondeur max = 15, Clusters = 15

Définition des outils

Decision Tree (Arbre de décision) est un type d'apprentissage automatique supervisé (c'est-à-dire qu'il faut expliciter le contenu de l'entrée et de la sortie correspondante dans les données de formation) où les données sont divisées en continu selon un certain paramètre. L'arbre est formé de deux entités, à savoir les nœuds de décision et les feuilles. Les feuilles représentent les décisions ou les résultats finaux. Les nœuds de décision représentent les conditions qui divisent les données. [2]

Random Forest (Forêt aléatoire) est un algorithme d'apprentissage automatique supervisé, principalement utilisé pour résoudre des problèmes de classification et de régression. L'algorithme construit des arbres de décision sur différents échantillons et prend leur vote majoritaire pour la classification et la moyenne en cas de régression. L'une des caractéristiques intéressantes de l'algorithme de Random Forest est qu'il peut traiter des données contenant des variables continues comme dans le cas de la régression et des variables discrètes comme dans le cas de la classification. [3]

Le Cluster ou Analyse en cluster est une technique d'apprentissage automatique qui regroupe l'ensemble de données non étiquetées. Il peut être défini comme *"Une façon de regrouper les points de données en différents groupes, constitués de points de données similaires. Les objets avec les similitudes possibles restent dans un groupe qui a moins ou pas de similitudes avec un autre groupe."* [4]

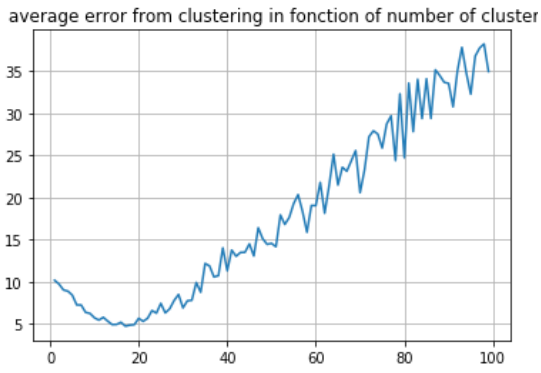
Problèmes rencontrés

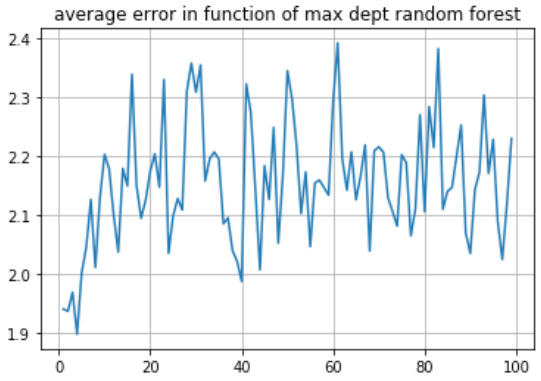
Je me suis posé des questions sur les données quand j'ai vu que certains jours la consommation dépassait les 100kWh alors que la consommation électrique moyenne d'un ménage français est de 12,5 kWh par jour. [5] En me penchant sur les données pour trouver d'où venait le problème, j'ai compris qu'il s'agissait du fait que le conteur électrique n'avait pas incrémenté les mesures pendant 9 jours de suite, du 30/07/2021 au 08/08/2021, et donc la consommation s'est accumulée avec la consommation du 10è jour.
Face à ces trous d'incrémentation des données, qui sont multiples, j'ai choisi de lisser les valeurs sur ces périodes en affectant pour chacune des journées la valeur moyenne ramenée sur 24h. Le lissage dans les cas où ce problème se présente facilite l'analyse des données.

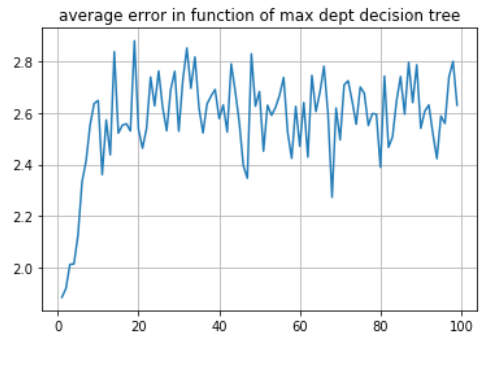
Au début de mon analyse j'ai envisagé prendre les premiers 70 % de la base de données pour entraîner les arbres de décisions puis utiliser les 30% restants pour les tester. Cela pose un problème pour la prédiction puisque la base de données couvre environ 1 an, et l'analyse des variables sur les 8 et demi premiers mois de l'année ne permet pas de prédire la consommation sur les mois restants car les conditions extérieures sont très différentes. J'ai donc décidé de séparer les variables en tranches de 10 jours, où les 7 premières valeurs servent à entraîner l'algorithme et les 3 suivantes pour le test.

Pour l'observation des résultats des arbres de décisions, nous allons étudier uniquement l'erreur moyenne entre la prédiction et les valeurs mesurées des jours test. En effet, mesurer la précision n'a pas d'intérêt dans ce cas de figure puisque les valeurs sont continues, donc la valeur exacte est presque impossible à trouver en pratique.

Analyse expérimentale :





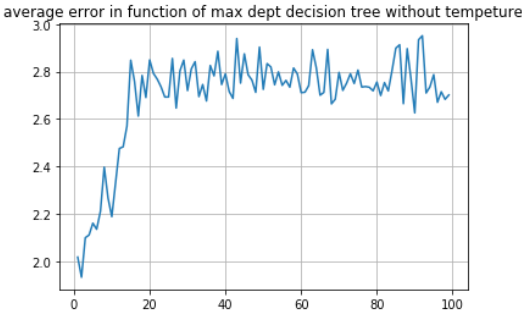


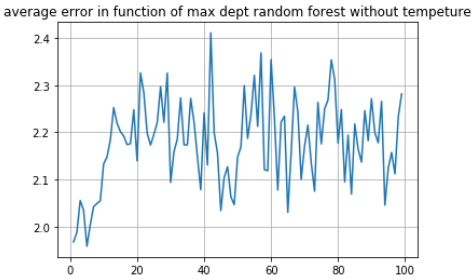
Ce qui saute aux yeux à la vue de ces résultats c'est la forte augmentation de l'erreur moyenne du clustering avec le nombre de cluster, ce qui n'est pas très efficace par rapport aux autres types de méthode. (cf. graphe 1a)
De plus, le clustering produit une erreur moyenne minimale de 5 kWh ce qui représente quand même environ 50 % de la consommation électrique moyenne journalière, de 10,6 kWh pour ce ménage.

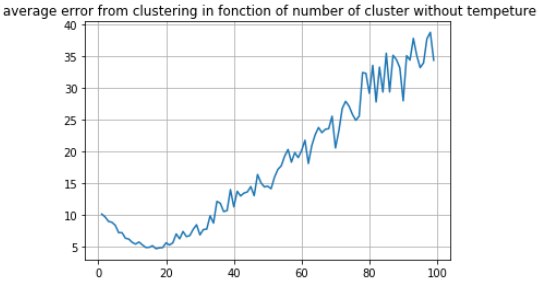
En revanche, avec la méthode de l'arbre de prédiction, l'erreur moyenne oscille autour des 26 % de la consommation moyenne (à partir de 15 de profondeur de l'arbre) et avec la forêt aléatoire on descend à environ 22% de la consommation moyenne à partir d'une profondeur de 10.

On en conclue dans ce cas que la forêt aléatoire est le meilleur type de prédiction. On peut néanmoins envisager que l'erreur moyenne augmente avec le nombre de variables prises en compte. Nous allons donc retirer une ou plusieurs variable(s) pour essayer d'obtenir des valeurs d'erreur moyenne plus faibles que celles observées sur les graphes ci-dessus.

1. Retrait de la variable température

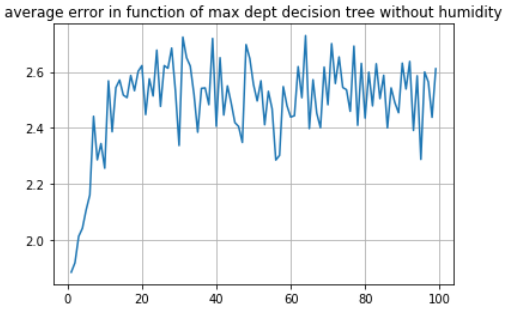


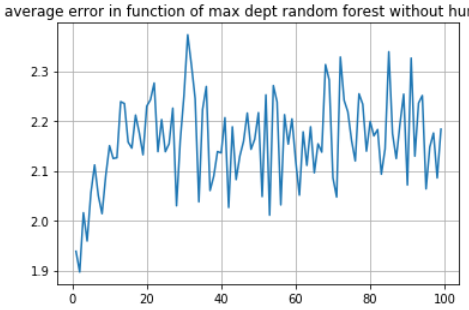


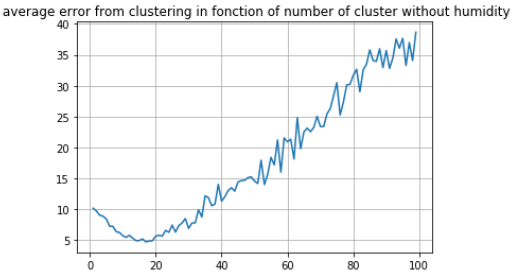


On peut remarquer que les résultats sont très proches des tests fait avec les valeurs de la température. Il y a une petite augmentation de l'erreur moyenne avec l'arbre de décision mais on a plus de précision avec la forêt aléatoire. En effet, les variations selon la profondeur sont moins importantes.
Pour le clustering l'allure est la même, le retrait de la variable de température n'a pas d'impact sur cette méthode.

2. Retrait de la variable humidité

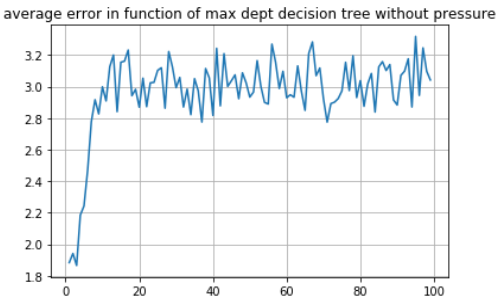


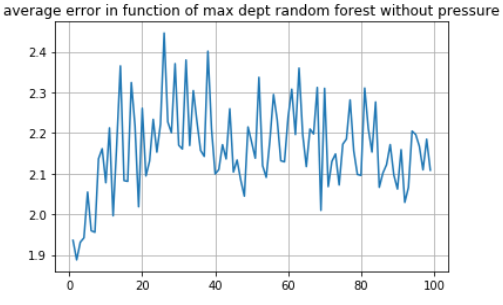


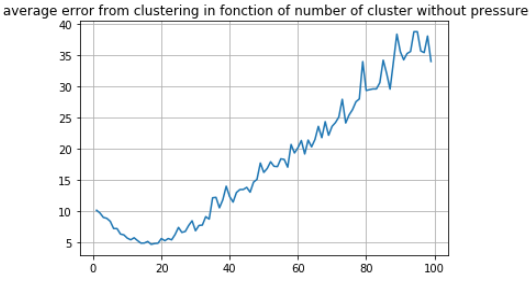


On voit avec l'arbre de décision que l'erreur moyenne représente 1% de la consommation électrique moyenne en moins par rapport à l'éducation avec toutes les variables.
Il en est de même pour la forêt aléatoire.
Pour le clustering l'allure est la même, le retrait de la variable d'humidité n'a pas d'impact sur cette méthode.

3. Retrait de la variable pression



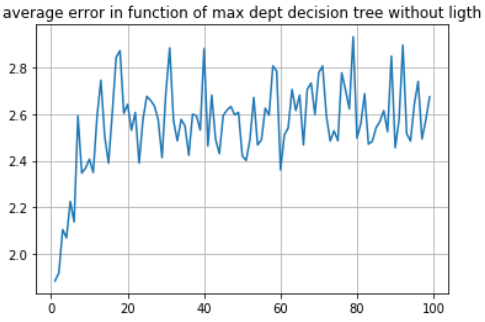


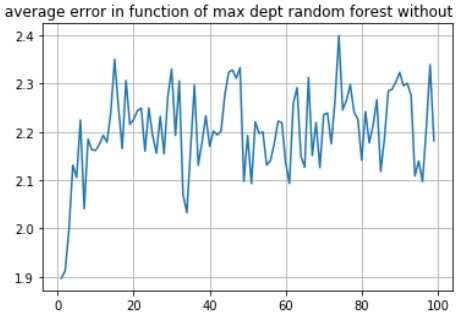


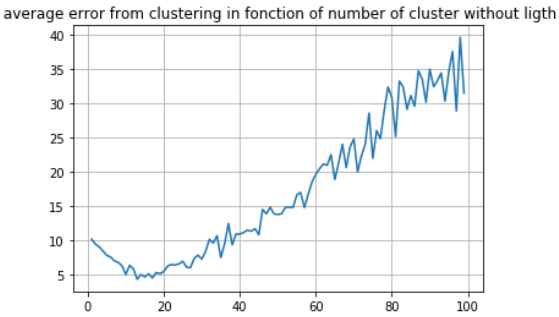
Pour l'arbre de décision, on observe une erreur moyenne supérieure à la valeur obtenue lors de la première expérience de 10% de la consommation moyenne.
En ce qui concerne la foret aléatoire, l'augmentation de l'erreur moyenne par rapport à la première valeur est seulement de l'ordre de 1,5 % de la consommation électrique moyenne.

Pour le clustering l'allure est la même, le retrait de la variable de pression n'a pas d'impact sur cette méthode.

4. Retrait de la variable luminosité

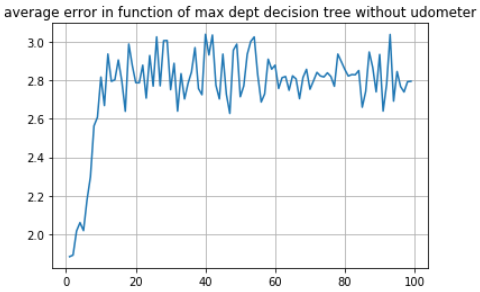


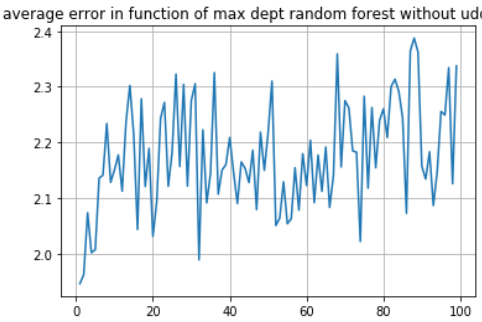


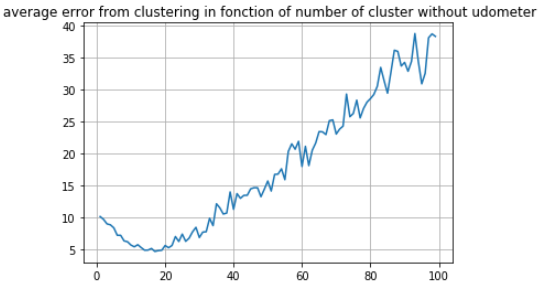


Le retrait de la variable de luminosité n'a aucun impact sur la consommation d'électricité.

5. Retrait de la variable pluviométrie

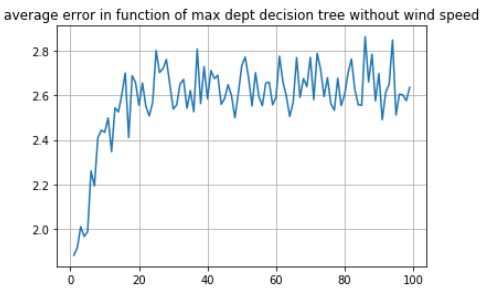


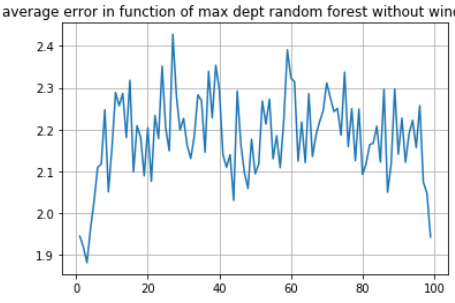


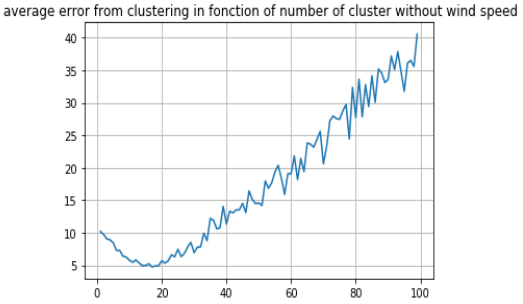


Le retrait de la variable pluviométrie diminue la précision de l'arbre de décision mais ne change pas les résultats des deux autres types de prédiction.

6. Retrait de la variable de vitesse du vent



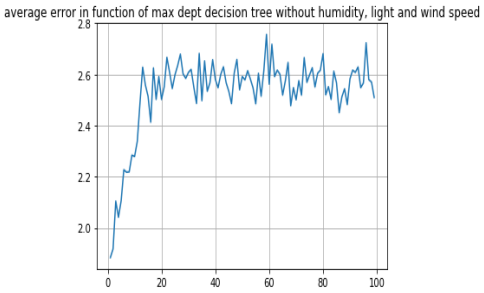


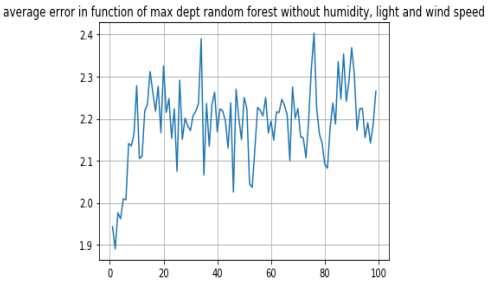


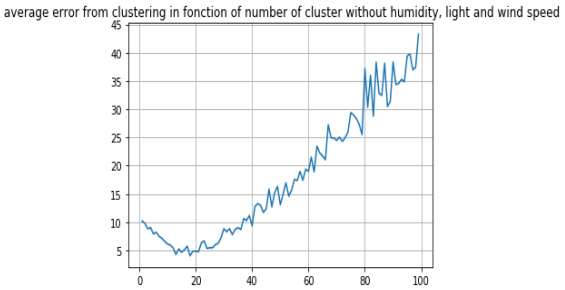
Le retrait de la variable vitesse du vent diminue très légèrement la précision de l'arbre de décision mais ne change pas les résultats des deux autres types de prédiction.

7. Retrait de plusieurs variables à la fois

Etudions les résultats du retrait des variables d'humidité, de pression et de vitesse du vent :







Nous voyons bien la diminution des erreurs moyennes comme avec le clustering qui passe sous les 5 kWh d'erreur moyenne minimale alors que dans les autres cas de figure, cela était plus vers les 6 kWh. Puis nous voyons aussi que l'arbre de décision gagne sur l'erreur moyenne : 25,5 % de la consommation moyenne. Mais la forêt aléatoire n'a pas une augmentation ni une diminution de l'erreur moyenne.

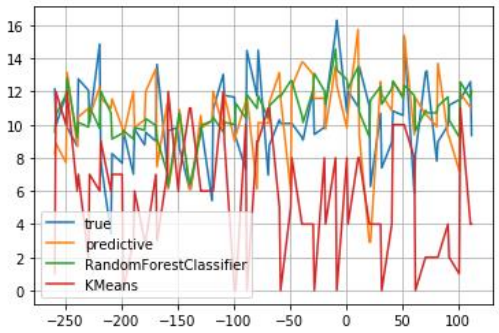
Conclusion : échec de l'expérience

On peut envisager plusieurs pistes pour expliquer l'échec de la non-réussite de cette expérience.

Une première explication vient de la base de données établie sur une durée trop courte avec un pas de temps trop long entre les valeurs exploitables. De plus, l'heure des relevés n'est pas prise en compte, or il y a une grande différence entre la consommation de jour et de nuit.
Une autre limitation des résultats s'explique par le fait que le chauffage pour la chaudière est au gaz dans cette maison, et n'est donc pas pris en compte ici, puisque l'on a que les relevés du compteur électrique.
Nous allons expliciter ces trois pistes d'amélioration.

Taille de la base de données

Pour commencer, la base de données à laquelle nous avons eu accès est trop restreinte.
On peut le constater dans le graphique suivant qui montre bien la difficulté à prédire les variations brusques de la consommation électrique.



En effet, les pics de consommation réelle (en bleu) sont grossièrement suivis par l'estimation de l'arbre de décision (en orange) et la forêt aléatoire (en vert) mais de manière imprécise.

Mais on pourrait envisager que ces changements brusques seraient mieux prédits avec une plus grande base de données, compilant les mesures sur plusieurs années par exemple.

Les conditions climatiques dépendent généralement de la période de l'année selon les saisons, il se pourrait que l'on observe des schémas d'évolutions de la consommation similaires d'une année sur l'autre, ce qui permettrait d'affiner les estimations de l'algorithme. Cela permettra aussi de mieux anticiper les habitudes énergétiques de la famille ainsi que d'anticiper certaines variations conditionnées par les conditions météorologiques de jours précédents. Par exemple, si la famille part tous les ans à la même période, ou encore, s'il pleut et fait très humide pendant quelques jours la famille va probablement repousser ses machines de linge pour quand les conditions seront plus adaptées au séchage du linge.

Avoir des données récoltées sur une période plus étendue permettrait ainsi de mieux éduquer les méthodes de décisions, même si l'on risque d'avoir une perte d'efficacité due au grand nombre de variables. Il faudrait alors tester une par une chaque variable ce qui serait exponentiellement plus long à effectuer.

Heure des relevés

Le pas de temps est trop grand pour bien estimer la consommation électrique de la maison.
En effet, l'heure des relevés à toute son importance pour prédire les variations de consommation. Par exemple, la pluie aura certainement plus d'impact le jour que la nuit sur les activités du ménage et donc sa consommation énergétique.

Absence de mesure de chauffage

Le fait de ne pas pouvoir prendre en compte le chauffage ne permet pas de traduire pleinement l'impact des conditions météorologiques sur a consommation énergétique, puisqu'il s'agit d'une dépense majeure, surtout l'hiver où le chauffage représente 62 % de la consommation énergétique d'un foyer [6]. Occulter le chauffage revient à fausser complètement nos estimations de l'utilisation d'énergie du foyer, puisque les variations observées dans nos résultats sont de fait beaucoup plus faibles qu'une maison qui se chauffe à l'énergie électrique en hiver.

Bilan

Nous pouvons finalement affirmer que cette expérience présente un fort potentiel mais les conditions d'exécution dans ce cadre ne permettent pas considérer notre expérience comme une réussite.
En effet, si nous approximons la consommation estimée par la moyenne de consommation électrique de la base d'entrainement, nous obtenons une erreur moyenne de 1,99 kWh, bien moins importante que celle obtenue avec le meilleur arbre de décision de 2,15 kWh !

Sources

[1] Consommations agrégées de gaz naturel et d'électricité fournies par les GRD aux personnes publiques estimation des données de thermosensibilité et de part thermosensible, fait par ENEDIS, GRDF et ORE : https://www.statistiques.developpement-durable.gouv.fr/sites/default/files/2019-06/note-methodologique-thermosensibilite.pdf 20/07/2018

[2] Decision Trees for Classification: A Machine Learning Algorithm : https://www.xoriant.com/blog/product-engineering/decision-trees-machine-learning-algorithm.html#:~:text=Introduction%20Decision%20Trees%20are%20a,namely%20decision%20nodes%20and%20leaves.

[3] Understanding Random Forest : https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/

[4] Clustering in Machine Learning : https://www.javatpoint.com/clustering-in-machine-learning

[5] Quelle est la consommation d'électricité moyenne par jour en France ?: https://www.lenergietoutcompris.fr/actualites-conseils/quelle-est-la-consommation-delectricite-moyenne-par-jour-en-france#:~:text=En%20France%2C%20la%20consommation%20%C3%A9lectrique%20moyenne%20d%E2%80%99un%20m%C3%A9nage,isolement%20ou%20encore%20le%20nombre%20d%E2%80%99%C3%A9quipements%20%C3%A9lectriques%20utilis%C3%A9s.

[6] Consommation électrique moyenne selon votre superficie : comment la calculer ? : https://particuliers.engie.fr/electricite/conseils-electricite/conseils-tarifs-electricite/consommation-electrique-moyenne-logement-par-superficie.html#:~:text='habitat...-,Quatre%20postes%20de%20consommation%20%C3%A9lectrique%20dans%20une%20maison,et%20la%20cuisson%20(8%20%25

Algorithmes

Extraction

```
from http.server import BaseHTTPRequestHandler, HTTPServer
import urllib
import time
from influxdb import InfluxDBClient
import sys
from datetime import datetime
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_ssl=False)

print(client.get_list_database())

client.switch_database('jeedom')

#### Timestamps in nano seconds ############
#print(client.query(query, bind_params=bind_params))
#print(now()
#print(client.query('SELECT "value" FROM "jeedom"."autogen"."5229" WHERE time >=
1641718672000000000 AND time < 1641805072000000000'))




#### Timestamps in seconds ###########"
timestamp_start= 1616612489  ## timestamp of the begin of period required (in second use
https://www.epochconverter.com/)
timestamp_end= 1653264000  ## timestamp of the end of period required (in second use
https://www.epochconverter.com/)

#### Convertion of timestamp to datetime
dt_object_start = datetime.fromtimestamp(timestamp_start)
print(dt_object_start)
dt_object_end = datetime.fromtimestamp(timestamp_end)
```

```
print(dt_object_end)

#### Preparation of the query
query = 'SELECT "value" FROM "jeedom"."autogen"."875" WHERE time >= $start_time  AND
time < $end_time '
bind_params = {'end_time': str(dt_object_end), 'start_time': str(dt_object_start)}

## query+Printof result

datasheet = client.query(query, bind_params=bind_params)

print(datasheet)
### Example from https://stackoverflow.com/questions/59155412/python-influx-query-inserting-
dates-into-query ###############
#query = "SELECT value FROM 'location/PRESSURE_SENSOR_1' where time >= $start_time
and time < $end_time"
#bind_params = {'end_time': '2019-10-03 00:00:00', 'start_time': '2019-10-02 00:00:00'}
#client.query(query, bind_params=bind_params))

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

with open("Capteur__Extérieur_Température.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])

query = 'SELECT "value" FROM "jeedom"."autogen"."876" WHERE time >= $start_time  AND
time < $end_time '
bind_params = {'end_time': str(dt_object_end), 'start_time': str(dt_object_start)}

## query+Printof result

datasheet = client.query(query, bind_params=bind_params)

print(datasheet)
### Example from https://stackoverflow.com/questions/59155412/python-influx-query-inserting-
dates-into-query ###############
#query = "SELECT value FROM 'location/PRESSURE_SENSOR_1' where time >= $start_time
and time < $end_time"
#bind_params = {'end_time': '2019-10-03 00:00:00', 'start_time': '2019-10-02 00:00:00'}
#client.query(query, bind_params=bind_params))
```

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

with open("Capteur_Extérieur_Humidite.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])

query = 'SELECT "value" FROM "jeedom"."autogen"."877" WHERE time >= $start_time  AND
time < $end_time '
bind_params = {'end_time': str(dt_object_end), 'start_time': str(dt_object_start)}

## query+Printof result

datasheet = client.query(query, bind_params=bind_params)

print(datasheet)
### Example from https://stackoverflow.com/questions/59155412/python-influx-query-inserting-
dates-into-query ###############
#query = "SELECT value FROM 'location/PRESSURE_SENSOR_1' where time >= $start_time
and time < $end_time"
#bind_params = {'end_time': '2019-10-03 00:00:00', 'start_time': '2019-10-02 00:00:00'}
#client.query(query, bind_params=bind_params))

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

with open("Capteur_Extérieur_Pression.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])

query = 'SELECT "value" FROM "jeedom"."autogen"."4412" WHERE time >= $start_time  AND
time < $end_time '
```

```
bind_params = {'end_time': str(dt_object_end), 'start_time': str(dt_object_start)}

## query+Printof result

datasheet = client.query(query, bind_params=bind_params)

print(datasheet)
### Example from https://stackoverflow.com/questions/59155412/python-influx-query-inserting-dates-into-query ###############
#query = "SELECT value FROM 'location/PRESSURE_SENSOR_1' where time >= $start_time and time < $end_time"
#bind_params = {'end_time': '2019-10-03 00:00:00', 'start_time': '2019-10-02 00:00:00'}
#client.query(query, bind_params=bind_params))

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

with open("Exterieur_BBX_Lumi_lux.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])

query = 'SELECT "value" FROM "jeedom"."autogen"."5439" WHERE time >= $start_time  AND time < $end_time '
bind_params = {'end_time': str(dt_object_end), 'start_time': str(dt_object_start)}

## query+Printof result

datasheet = client.query(query, bind_params=bind_params)

print(datasheet)
### Example from https://stackoverflow.com/questions/59155412/python-influx-query-inserting-dates-into-query ###############
#query = "SELECT value FROM 'location/PRESSURE_SENSOR_1' where time >= $start_time and time < $end_time"
#bind_params = {'end_time': '2019-10-03 00:00:00', 'start_time': '2019-10-02 00:00:00'}
#client.query(query, bind_params=bind_params))

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

with open("pluviomètre_Pluie_24H.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])

query = 'SELECT "value" FROM "jeedom"."autogen"."1802" WHERE time >= $start_time  AND
time < $end_time '
bind_params = {'end_time': str(dt_object_end), 'start_time': str(dt_object_start)}

## query+Printof result

datasheet = client.query(query, bind_params=bind_params)

print(datasheet)
### Example from https://stackoverflow.com/questions/59155412/python-influx-query-inserting-
dates-into-query ###############
#query = "SELECT value FROM 'location/PRESSURE_SENSOR_1' where time >= $start_time
and time < $end_time"
#bind_params = {'end_time': '2019-10-03 00:00:00', 'start_time': '2019-10-02 00:00:00'}
#client.query(query, bind_params=bind_params))

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

with open("WindSensor_Vitesse_vent.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])

query = 'SELECT "value" FROM "jeedom"."autogen"."6516" WHERE time >= $start_time  AND
time < $end_time '
bind_params = {'end_time': str(dt_object_end), 'start_time': str(dt_object_start)}

## query+Printof result

datasheet = client.query(query, bind_params=bind_params)
```

```
print(datasheet)
### Example from https://stackoverflow.com/questions/59155412/python-influx-query-inserting-dates-into-query ###############
#query = "SELECT value FROM 'location/PRESSURE_SENSOR_1' where time >= $start_time and time < $end_time"
#bind_params = {'end_time': '2019-10-03 00:00:00', 'start_time': '2019-10-02 00:00:00'}
#client.query(query, bind_params=bind_params))

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

with open("Compteur_électrique_BASE.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])
```

Mise au bon format

```
import pandas as pd



data = pd.read_csv('Capteur__Extérieur_Température.csv', sep=";")
data['time'] = pd.to_datetime(data['time'])
pd.to_datetime(data['time'], format='%Y-%m-%dT%H:%M:%S')
data.set_index('time', inplace=True)
data_new=data.resample('1440T').pad()
print(data_new)
newdata=pd.DataFrame(data_new)
newdata.to_csv('Capteur__Extérieur_Température1.csv')

data = pd.read_csv('Capteur_Extérieur_Humidite.csv', sep=";")
data['time'] = pd.to_datetime(data['time'])
pd.to_datetime(data['time'], format='%Y-%m-%dT%H:%M:%S')
data.set_index('time', inplace=True)
data_new=data.resample('1440T').pad()
print(data_new)
newdata=pd.DataFrame(data_new)
newdata.to_csv('Capteur_Extérieur_Humidite1.csv')

data = pd.read_csv('Capteur_Extérieur_Pression.csv', sep=";")
data['time'] = pd.to_datetime(data['time'])
```

```
pd.to_datetime(data['time'], format='%Y-%m-%dT%H:%M:%S')
data.set_index('time', inplace=True)
data_new=data.resample('1440T').pad()
print(data_new)
newdata=pd.DataFrame(data_new)
newdata.to_csv('Capteur_Extérieur_Pression1.csv')

data = pd.read_csv('Exterieur_BBX_Lumi_lux.csv', sep=";")
data['time'] = pd.to_datetime(data['time'])
pd.to_datetime(data['time'], format='%Y-%m-%dT%H:%M:%S')
data.set_index('time', inplace=True)
data_new=data.resample('1440T').mean()
print(data_new)
newdata=pd.DataFrame(data_new)
newdata.to_csv('Exterieur_BBX_Lumi_lux1.csv')

data = pd.read_csv('pluviomètre_Pluie_24H.csv', sep=";")
data['time'] = pd.to_datetime(data['time'])
pd.to_datetime(data['time'], format='%Y-%m-%dT%H:%M:%S')
data.set_index('time', inplace=True)
data_new=data.resample('1440T').pad()
print(data_new)
newdata=pd.DataFrame(data_new)
newdata.to_csv('pluviomètre_Pluie_24H1.csv')

data = pd.read_csv('WindSensor_Vitesse_vent.csv', sep=";")
data['time'] = pd.to_datetime(data['time'])
pd.to_datetime(data['time'], format='%Y-%m-%dT%H:%M:%S')
data.set_index('time', inplace=True)
data_new=data.resample('1440T').pad()
print(data_new)
newdata=pd.DataFrame(data_new)
newdata.to_csv('WindSensor_Vitesse_vent1.csv')

data = pd.read_csv('Compteur_électrique_BASE.csv', sep=";")
data['time'] = pd.to_datetime(data['time'])
pd.to_datetime(data['time'], format='%Y-%m-%dT%H:%M:%S')
data.set_index('time', inplace=True)
data_new=data.resample('1440T').pad()
print(data_new)
newdata=pd.DataFrame(data_new)
newdata.to_csv('Compteur_électrique_BASE1.csv')
```

Exploitation des données

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

```
temperature, humidite,
pression,luminosite,pluviometre,vitesse_vent,compteur_electrique=[],[],[],[],[],[],[]

file=open("Capteur__Extérieur_Température1.csv")
Title=file.readline()
for line in file:
    line_token=line.split(";")
    print(line_token[1])
    temperature.append(float(line_token[1]))
file.close()

file=open("Capteur_Extérieur_Humidite1.csv")
Title=file.readline()
for line in file:
    line_token=line.split(";")
    humidite.append(float(line_token[1]))
file.close()

file=open("Capteur_Extérieur_Pression1.csv")
Title=file.readline()
for line in file:
    line_token=line.split(";")
    pression.append(float(line_token[1]))
file.close()

file=open("Exterieur_BBX_Lumi_lux1.csv")
Title=file.readline()
for line in file:
    line_token=line.split(";")
    luminosite.append(float(line_token[1]))
file.close()

file=open("pluviomètre_Pluie_24H1.csv")
Title=file.readline()
for line in file:
    line_token=line.split(";")
    pluviometre.append(float(line_token[1]))
file.close()

file=open("WindSensor_Vitesse_vent1.csv")
Title=file.readline()
for line in file:
    line_token=line.split(";")
    vitesse_vent.append(float(line_token[1]))
file.close()

file=open("Compteur_électrique_BASE1.csv")
Title=file.readline()
for line in file:
    line_token=line.split(";")
    compteur_electrique.append(float(line_token[1]))
```

```
file.close()


X,Y=[],[]
Xt,Yt=[],[]
X1=[]
a=int(len(temperature)*0.7)
T=[]

for i in range(1,len(temperature)):
    test=i%10
    if test<=7:
        X1.append(temperature[i])
        X1.append(humidite[i])
        X1.append(pression[i])
        X1.append(luminosite[i])
        X1.append(pluviometre[i])
        X1.append(vitesse_vent[i])
        X.append(X1)
        X1=[]
        Y.append(compteur_electrique[i]-compteur_electrique[i-1])
    else:
        X1.append(temperature[i])
        X1.append(humidite[i])
        X1.append(pression[i])
        X1.append(luminosite[i])
        X1.append(pluviometre[i])
        X1.append(vitesse_vent[i])
        Xt.append(X1)
        X1=[]
        Yt.append(compteur_electrique[i]-compteur_electrique[i-1])
        T.append(i-a)

s=0

for i in range (len(Y)):
    s=s+Y[i]

s=s/len(Y)
k=0
for i in range (len(Yt)):
    k=k+(abs(Yt[i]-s))
k=k/len(Yt)

print(k)
print(s)

SA=[]
SAE=[]
ST=[]
EA=[]
EAE=[]
```

```
CA=[]
CE=[]

print (Y)

for j in range (1,100):
    clf = tree.DecisionTreeRegressor(max_depth=j,random_state=None)
    clf = clf.fit(X, Y)
    with open("tree.txt", 'w') as file:
        f = tree.export_graphviz(clf,out_file=file)

    S=clf.predict(Xt)

    treef=RandomForestRegressor(max_depth=j,random_state=None)
    treef=treef.fit(X,Y)
    E=treef.predict(Xt)

    cluster=KMeans(n_clusters=j, init='k-means++', n_init=20, max_iter=300, tol=0.0001,
verbose=0, random_state=0, copy_x=True, algorithm='auto')
    cluster=cluster.fit(X,Y)
    C=cluster.predict(Xt)

    s=0
    se=0
    ce=0
    for i in range(len(Xt)):
        s=s+abs(Yt[i]-S[i])
        se=se+abs(Yt[i]-E[i])
        ce=ce+abs(Yt[i]-C[i])
    ae=s/len(Xt)
    se=se/len(Xt)
    ce=ce/len(Xt)

    a=0
    e=0
    c=0
    for i in range(len(Xt)):
        if Yt[i]==S[i]:
            a=a+1
        if Yt[i]==E[i]:
            e=e+1
        if Yt[i]==C[i]:
            c=c+1
    a=a/len(Xt)
    e=e/len(Xt)
    c=c/len(Xt)

    SAE.append(ae)
    SA.append(a)
    EAE.append(se)
    EA.append(e)
    CE.append(ce)
```

```
    CA.append(c)
    ST.append(j)

    plt.plot(T,Yt,label="true")
    plt.plot(T,S,label="predictive")
    plt.plot(T,E,label="RandomForestClassifier")
    plt.plot(T,C,label="KMeans")
    plt.legend()
    plt.grid()
    plt.show()
    print(j)


plt.plot(ST,SAE)
plt.title("average error in function of max dept decision tree")
plt.grid()
plt.show()


plt.plot(ST,EAE)
plt.title("average error in function of max dept random forest")
plt.grid()
plt.show()


plt.plot(ST,CE)
plt.title(" average error from clustering in fonction of number of cluster")
plt.grid()
plt.show()
```