UNIVERSITE LIBANAISE

FACULTE DE TECHNOLOGY

Génie de Réseaux Informatiques
Et Télécommunications

Projet fin d'étude

Préparé par

Mohamed Zahed BAHLAK

Entreprise D'accueil

Université Libanaise

Encadré par

Dr.CHANTAF Samer

Date de soumission: 15 Juillet 2022

Remerciements

Tout d’abord j’adresse mes remerciements au doyen de la faculté de technologie de l’université Libanaise **Dr Mohamad Hajjar**, au directeur de la faculté à Saida **Dr Mahmoud Abbas**, au directeur du département GRIT **Dr Mohamad Alwan** et à tous les professeurs de la faculté de technologie qui m’ont aidé à avoir une bonne formation.

Je tiens tout particulièrement à remercier mon Professeur **Dr. Samer Chantaf**, directeur de ce projet pour le temps qu’il m’a consacré, pour le partage de son expertise au quotidien et pour les aides précieux qu’il m’a apporté dans ce projet.

Je tiens également à remercier les membres du jury qui me font l’honneur de juger mon travail.

Table de matière

Table des figures

Liste des tableaux

Abréviations

CSV: Valeurs séparés par une virgule.
E/S: Entrée/Sortie.
Re: Expressions régulières.
NLTK: La boite à outils en langage naturel.

Introduction générale

Le domaine de l'apprentissage automatique, qui peut être brièvement défini comme permettant aux ordinateurs de faire des prédictions réussies à l'aide d'expériences passées, a récemment connu un développement impressionnant grâce à l'augmentation rapide de la capacité de stockage et de la puissance de traitement des ordinateurs. Avec de nombreuses autres disciplines, les méthodes d'apprentissage automatique ont été largement utilisées en bio-informatique. Les difficultés et le coût des analyses biologiques ont conduit au développement d'approches sophistiquées d'apprentissage automatique pour ce domaine d'application. Dans ce mémoire, nous passons d'abord en revue les concepts fondamentaux de l'apprentissage automatique tels que l'évaluation des fonctionnalités, l'apprentissage non supervisé par rapport à l'apprentissage supervisé et les types de classification. Ensuite, nous soulignons les principaux enjeux de la conception d'expériences d'apprentissage automatique et de leur évaluation des performances. Enfin, nous introduisons quelques méthodes d'apprentissage supervisé.

Une série temporelle est une succession de données classées chronologiquement et espacées à intervalles égaux ou inégaux. Le processus de prévision consiste à prédire la valeur futur d’une série

temporelle, soit en modélisant la série uniquement sur la base de son comportement passé (autorégressif), soit en utilisant d'autres variables externes.

Lorsque vous travaillez avec des séries chronologiques, il est rarement nécessaire de prédire uniquement l'élément suivant de la série (t+1). Au lieu de cela, l'objectif le plus courant est de prédire un intervalle futur entier ((t+1), ..., (t + n)) ou un point éloigné dans le temps (t + n). Plusieurs stratégies permettent de générer ce type de prédiction, skforecast a implémenté ce qui suit :

- Prévision multi-étapes récursive : puisque la valeur t_{n-1} est requise pour prédire t_n, et que t_{n-1} est inconnue, il est nécessaire de faire des prédictions récursives dans lesquelles chaque nouvelle prédiction est basée sur la précédente.
- Prévision directe en plusieurs étapes : cette méthode implique la formation d'un modèle différent pour chaque étape.

Ce projet montre un exemple d'utilisation des méthodes de prévision pour prévoir la demande horaire d'électricité. Plus précisément, il introduit skforecast, une bibliothèque simple qui contient les classes et les fonctions nécessaires pour adapter tout modèle de régression scikit-learn aux problèmes de prévision.

Chapitre 1 : Langage Python

1.1 Introduction

Dans ce chapitre, nous allons présenter le langage de programmation Python, langage utilisé dans le projet et les différentes bibliothèques standards associées.

1.2 Langage Python

Python est un langage de programmation interprété, orienté objet et de haut niveau avec une sémantique dynamique. Ses structures de données intégrées de haut niveau, combinées au typage dynamique et à la liaison dynamique, le rendent très attrayant pour le développement rapide d'applications, ainsi que pour une utilisation en tant que langage de script ou de collage pour connecter des composants existants entre eux. La syntaxe simple et facile à apprendre de Python met l'accent sur la lisibilité et réduit donc le coût de maintenance du programme. Python prend en charge les modules et les packages, ce qui encourage la modularité du programme et la réutilisation du code. L'interpréteur

Python et la vaste bibliothèque standard sont disponibles gratuitement sous forme source ou binaire pour toutes les principales plates-formes et peuvent être librement distribués.

1.3 Bibliothèques

Plusieurs bibliothèques standard sont disponibles gratuitement. Nous allons les décrire brièvement.

1.3.1 Pandas

Pandas est une puissante boîte à outils d'analyse de données Python. Lorsque nous travaillons avec des données tabulaires, telles que des données stockées dans des feuilles de calcul ou des bases de données, pandas est l'outil qu'il nous faut.

Panda nous aidera à explorer, nettoyer et traiter nos données. Elle prend en charge l'intégration avec de nombreux formats de fichiers ou sources de données prêts à l'emploi (csv, Excel, SQL, json, parquet, …). L'importation de données à partir de chacune de ces sources de données est fournie par la fonction avec le préfixe read_*. De même, les méthodes to_* sont utilisées pour stocker des données [1].

Pandas permet de tracer vos données prêtes à l'emploi, en utilisant la puissance de Matplotlib. Vous pouvez choisir le type de tracé (scatter, bar, boxplot, …) correspondant à vos données.

1.3.2 NumPy

NumPy est le package fondamental pour le calcul scientifique en Python. Il s'agit d'une bibliothèque Python qui fournit un objet tableau multidimensionnel, divers objets dérivés (tels que des tableaux masqués et des matrices) et un assortiment de routines pour des opérations rapides sur des tableaux, y compris mathématiques, logiques, manipulation de forme, tri, sélection, E/S, transformées de Fourier discrètes, algèbre linéaire de base, opérations statistiques de base, simulation aléatoire et bien plus encore [2].

1.3.3 matplotlib.pyplot

Matplotlib.pyplot est une collection de fonctions qui font fonctionner matplotlib comme MATLAB. Chaque fonction pyplot apporte des modifications à une figure. Par exemple, crée une figure, crée une zone de traçage dans une figure, trace certaines lignes dans une zone de traçage, décore le tracé avec des étiquettes, etc… dans matplotlib [3].

1.3.4 Seaborn

Seaborn est une bibliothèque Python de visualisation de données basée sur matplotlib. Il fournit une interface de haut niveau pour dessiner des graphiques statistiques attrayants et informatifs [4].

1.3.5 Re

Une expression régulière (ou Re) spécifie un ensemble de chaînes qui lui correspond; les fonctions de ce module nous permettent de vérifier si une chaîne particulière correspond à une expression régulière donnée (ou si une expression régulière donnée correspond à une chaîne particulière, ce qui revient au même) [5].

1.3.6 NLTK

Le Natural Language Toolkit (NLTK) est une plate-forme utilisée pour créer des programmes Python qui fonctionnent avec des données de langage humain pour une application dans le traitement statistique du langage naturel (TLN). Il contient des bibliothèques de traitement de texte pour la tokenisation, l'analyse, la classification, la radicalisation, le balisage et le raisonnement sémantique [6].

1.3.7 Warnings

Les messages d'avertissement sont généralement émis dans des situations où il est utile d'alerter l'utilisateur d'une condition dans un

programme, où cette condition (normalement) ne justifie pas de déclencher une exception et de terminer le programme. Par exemple, on peut souhaiter émettre un avertissement lorsqu'un programme utilise un module obsolète [7].

1.3.8 Sklearn.model_selection: import train_test_split

C’est la bibliothèque la plus puissante et la plus robuste pour la machine learning en Python. Elle fournit une sélection d’outils efficaces pour l’apprentissage automatique et la modélisation statistique, notamment la classification, la régression et le clustering via une interface cohérente en Python. Cette bibliothèque, qui est en grande partie écrite en Python, s’appuie sur NumPy, SciPy et Matplotlib [8].

1.3.9 Sklearn.ensemble: import RandomForestClassifier

C’est un estimateur qui adapte un certain nombre de classificateurs d’arbre de décision sur divers sous-échantillons de l’ensemble de données et utilise la moyenne pour améliorer la précision prédictive et contrôler le sur ajustement. La taille du sous-échantillon est contrôlée avec le paramètre max_samples si bootstrap=True (par défaut), si non l’ensemble de données entier est utilisé pour construire chaque arbre [9].

1.4 Conclusion

Dans ce chapitre, nous avons présenté notre interpréteur Python et

les différentes bibliothèques associées.

Chapitre 2 : Etude théorique et implantation pratique du projet

2.1 Introduction

Dans le chapitre précèdent, nous avons décrit les outils utilisés pour réaliser notre projet. Dans ce chapitre, nous allons présenter ce projet, les librairies utilisées ainsi que les données

2.2 But de notre projet

Disons que nous avons une maison qui peut avoir beaucoup de matériel électrique, pour que nous sachions combien la facture sera à l'avenir, certaines personnes avaient fait des programmes et partagé avec nous afin que nous puissions l'utiliser, cela peut nous aider au processus de prévision de notre facture le lendemain peut-être, ou le mois suivant, même après une année complète.

Notre objectif est de prédire la quantité d'électricité dépensée en utilisant les données disponibles dans le fichier Mon_Studio_Data.csv. Appliquer l'ingénierie des fonctionnalités et le réglage du modèle pour obtenir 80 % à 95 % de la précision. Nous allons réaliser ce programme en utilisant le langage python avec l'apprentissage automatique.

2.3 Les bibliothèques

Pour réaliser notre projet, nous allons utiliser plusieurs librairies qui sont :

```
# Data manipulation
# ========================================
========================================

Import pandas as pd
Import numpy as np

# Plots
# ========================================
========================================
import matplotlib.pyplot as plt
import seaborn as sns
import re
import nltk

# Modeling and Forecasting

# ========================================
========================================
```

```
From sklearn.model_selection import train_test_split
From sklearn.ensemble import RandomForestClassifier

# Warnings configuration

# ============================================================
============================================

import warnings
warnings.filterwarnings('ignore')
```

2.4 Les données

Les données utilisées dans ce projet ont été faites par moi-même. L'ensemble de mes données contient 28 colonnes, et 2254 enregistrements complets de 2017 à 2022. Étant donné le nom de la variable, le type de variable, l'unité de mesure et une brève description. La consommation d'électricité est le problème de régression. Les informations dans chaque colonne sont :

- Day--Day of the date--Input Variable
- Month--Month of the date--Input Variable
- Year--Year of the date--Input Variable
- FridgeConsVal--one-hour Fridge consumption--Input Variable
- FridgeConsTime--Time Fridge consumption--Input Variable
- FridgeConsTotal--Total Fridge consumption--Input Variable
- AC_ConsVal--one-hour AC consumption--Input Variable

- AC_ConsTime--Time AC consumption--Input Variable
- AC_ConsTotal--Total AC consumption--Input Variable
- WashingMachineConsVal-one-hour
  WashingMachine consumption--Input Variable
- WashingMachineConsTime--
  Time WashingMachine consumption--Input Variable
- WashingMachineConsTotal--
  Total WashingMachine consumption--Input Variable
- MicrowaveConsVal--one-hour Microwave consumption--
  Input Variable
- MicrowaveConsTime--Time Microwave consumption--
  Input Variable
- MicrowaveConsTotal--Total Microwave consumption--
  Input Variable
- TVConsVal--one-hour TV consumption--Input Variable
- TVConsTime--Time TV consumption--Input Variable
- TVConsTotal--Total TV consumption--Input Variable
- SpeakerConsVal--one-hour Speaker consumption--
  Input Variable
- SpeakerConsTime--Time Speaker consumption--
  Input Variable
- SpeakerConsTotal--Total Speaker consumption--
  Input Variable

- TotalCons--forecasted Consumption--Input Variable
- RealTotalCons--The actual ammount of electricity consumed
- (labels or values to be predicted)--Output Variable

2.5 Connexion

Pour se connecter sur Google Drive :

```
# Connecting to Google Drive

#=========================================
================

from google.colab import drive
drive.mount('/content/drive')

# Data download

#=========================================
================

file_path = '/content/drive/My Drive/Colab Notebooks/project/Mon_Studio_Data.csv'
Dataset = pd.read_csv(file_path)
```

2.6 Affichage des données

Dans le tableau 2.1, nous présentons un exemple des données :

```
dataset.head(10)
```

	Day	Month	Year	FridgeConsVal	FridgeConsTime	FridgeConsTotal	AC_ConsTime	AC_ConsVal	AC_ConsTotal	WashingMachineConsVal	...	MicrowaveConsTime	MicrowaveConsTotal	TVConsVal	TVConsTime	TVConsTotal	SpeakerConsV
0	1	1	2017	200	14.246704	2849.340761	5.911297	1000	5911.296738	500	...	0.718102	933.532525	20	0.733620	14.672397	
1	2	1	2017	200	21.147261	4229.452130	7.997217	1000	7997.217397	500	...	0.589766	766.695757	20	2.603107	52.062147	
2	3	1	2017	200	23.370561	4674.112249	2.079248	1000	2079.247666	500	...	0.453708	589.820504	20	2.039337	40.786731	
3	4	1	2017	200	3.734855	746.970990	3.635435	1000	3635.435169	500	...	0.571774	743.305838	20	2.208055	44.161100	
4	5	1	2017	200	12.881658	2576.331680	4.008363	1000	4008.363377	500	...	0.697911	907.284080	20	1.815397	36.307949	
5	6	1	2017	200	12.680021	2536.004156	1.278179	1000	1278.179476	500	...	0.629091	817.817777	20	1.197364	23.947286	
6	7	1	2017	200	21.723009	4344.601811	3.224340	1000	3224.339857	500	...	0.174796	227.234440	20	1.817031	36.340617	
7	8	1	2017	200	13.869803	2773.960640	2.296657	1000	2296.656789	500	...	0.071213	92.576390	20	1.013179	20.263590	
8	9	1	2017	200	17.211568	3442.313667	1.347652	1000	1347.651976	500	...	0.000610	0.793171	20	2.115872	42.317437	
9	10	1	2017	200	7.087084	1417.416823	2.964972	1000	2964.971876	500	...	0.560552	728.718018	20	2.916899	58.337975	

10 rows × 29 columns

Tableau 2.1: Tableau des données.

2.7 Affichage des informations

Dans le tableau 2.2, nous affichons les informations concernant nos variables.

dataset.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2252 entries, 0 to 2251
Data columns (total 29 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Day                       2252 non-null   int64
 1   Month                     2252 non-null   int64
 2   Year                      2252 non-null   int64
 3   FridgeConsVal             2252 non-null   int64
 4   FridgeConsTime            2252 non-null   float64
 5   FridgeConsTotal           2252 non-null   float64
 6   AC_ConsTime               2252 non-null   float64
 7   AC_ConsVal                2252 non-null   int64
 8   AC_ConsTotal              2252 non-null   float64
 9   WashingMachineConsVal     2252 non-null   int64
 10  WashingMachineConsTime    2252 non-null   int64
 11  WashingMachineConsTotal   2252 non-null   int64
 12  DesktopConsVal            2252 non-null   int64
 13  DesktopConsTime           2252 non-null   float64
 14  DesktopConsTotal          2252 non-null   float64
 15  LampConsVal               2252 non-null   int64
 16  LampConsTime              2252 non-null   float64
 17  LampConsTotal             2252 non-null   float64
 18  MicrowaveConsVal          2252 non-null   int64
 19  MicrowaveConsTime         2252 non-null   float64
 20  MicrowaveConsTotal        2252 non-null   float64
 21  TVConsVal                 2252 non-null   int64
 22  TVConsTime                2252 non-null   float64
 23  TVConsTotal               2252 non-null   float64
 24  SpeakerConsVal            2252 non-null   int64
 25  SpeakerConsTime           2252 non-null   float64
 26  SpeakerConsTotal          2252 non-null   float64
 27  TotalCons                 2252 non-null   float64
 28  RealTotalCons             2252 non-null   float64
dtypes: float64(16), int64(13)
memory usage: 510.3 KB
```

Tableau 2.2 : Tableau des informations.

Nous remarquons que toutes les fonctionnalités avec des valeurs numériques sont bonnes à utiliser (float64, int64) dans l'ensemble de données. Nous allons vérifier si une ou plusieurs variables prennent la valeur zéro.

dataset.isnull().sum()

```
Day                        0
Month                      0
Year                       0
FridgeConsVal              0
FridgeConsTime             0
FridgeConsTotal            0
AC_ConsTime                0
AC_ConsVal                 0
AC_ConsTotal               0
WashingMachineConsVal      0
WashingMachineConsTime     0
WashingMachineConsTotal    0
DesktopConsVal             0
DesktopConsTime            0
DesktopConsTotal           0
LampConsVal                0
LampConsTime               0
LampConsTotal              0
MicrowaveConsVal           0
MicrowaveConsTime          0
MicrowaveConsTotal         0
TVConsVal                  0
TVConsTime                 0
TVConsTotal                0
SpeakerConsVal             0
SpeakerConsTime            0
SpeakerConsTotal           0
TotalCons                  0
RealTotalCons              0
dtype: int64
```

Tableau 2.3 : Tableau de vérification des valeurs nulles.

Nous constatons que toutes nos variables sont différentes de la valeur nulle.

2.8 Exploration graphique

Lorsqu'il est nécessaire de générer un modèle de prévision, le traçage des valeurs de la série chronologique peut être utile. Cela permet d'identifier des modèles tels que les tendances et la saisonnalité.

Maintenant, nous allons utiliser le boxplot pour montrer l'existence de valeurs aberrantes.

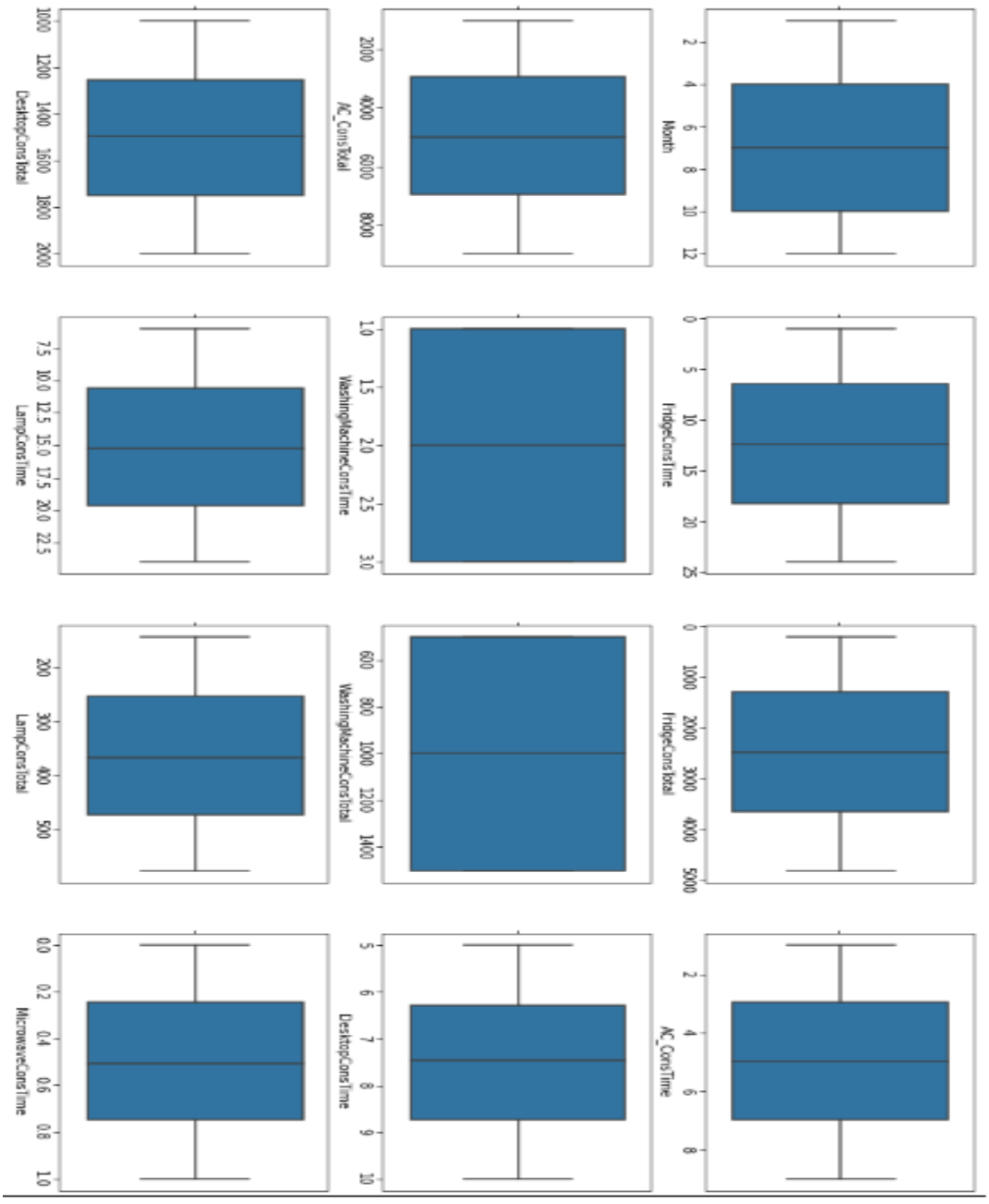

Figure 2.1: Recherche des valeurs aberrantes.

```
datset_new = dataset. iloc[:, [0,1,4,5,6,8,10,11,13,14,16,17,19]]
plt.figure(figsize=(20,15))
pos = 1
for i in datset_new.drop(columns = 'Day').columns:
    plt.subplot(4, 3, pos)
    sns.boxplot(datset_new[i])
    pos += 1
```

D’après la figure 2.1, nous constatons le non existence des valeurs aberrantes.

2.9 Analyse bivariée

Nous allons examinons maintenant la corrélation entre toutes les colonnes de l'ensemble de données (tableau 2.3):

```
correlations = dataset[["Day", "Month", "FridgeConsTime",
"AC_ConsTime","WashingMachineConsTime","DesktopConsTime","
LampConsTime","MicrowaveConsTime","TVConsTime",
"SpeakerConsTime","TotalCons","RealTotalCons"]].corr(method='pea
rson')
plt.figure(figsize=(16, 12))
sns.heatmap(correlations, cmap="YlGnBu", annot=True)
plt.show()
```

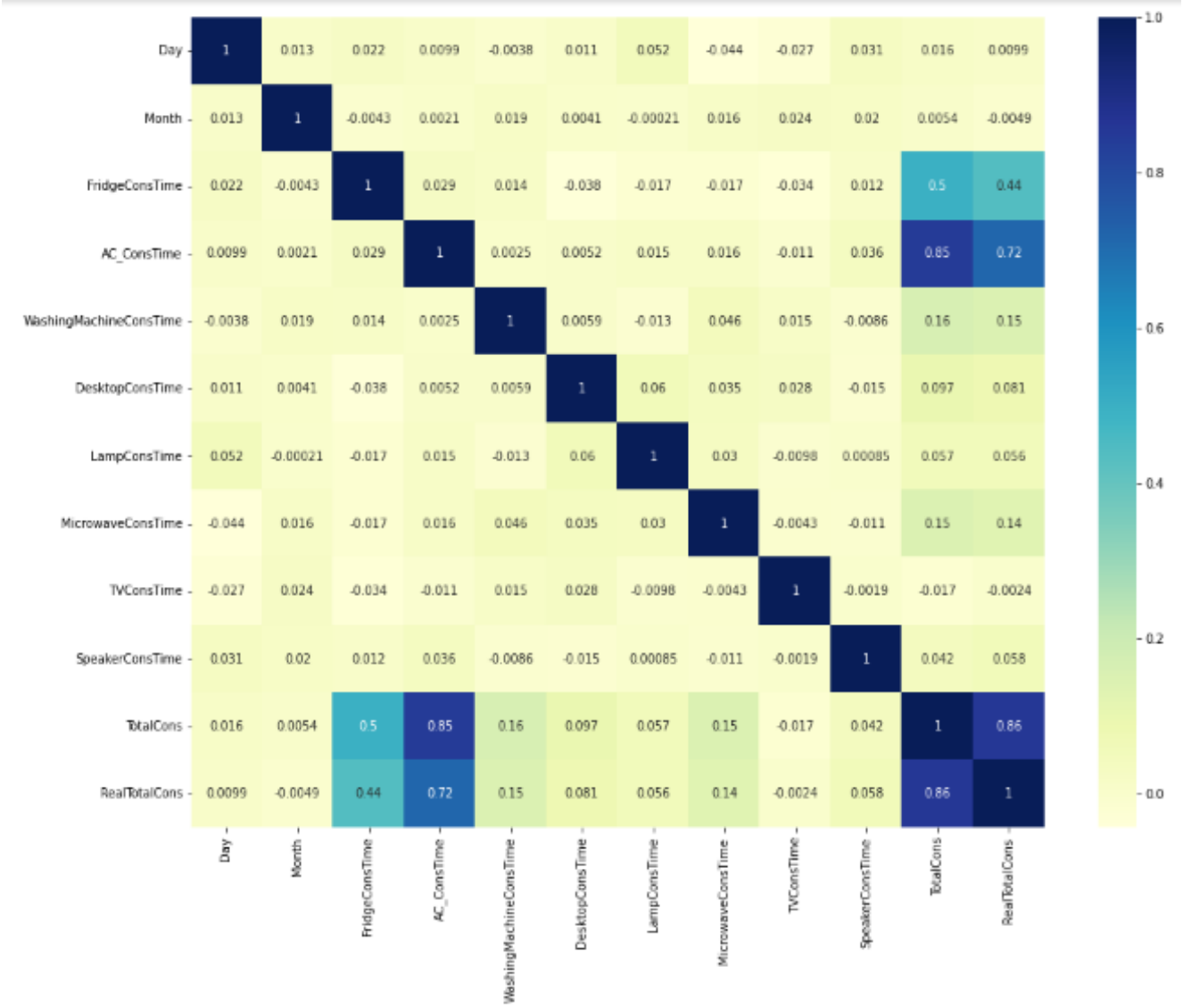

Tableau 2.4 : Tableau de corrélation.

2.10 Étape d'apprentissage automatique

Nous allons passer à la tâche de former un modèle de prévision des prix de l'électricité. Ici, nous allons tout d'abord associer toutes les

fonctionnalités importantes à la variable x et la colonne cible à y, puis nous diviserons les données en ensembles d'apprentissage et de test:

```
x = dataset[["Day", "Month", "FridgeConsTime"
              ,"AC_ConsTime","WashingMachineConsTime"
              ,"DesktopConsTime","LampConsTime"
              ,"MicrowaveConsTime","TVConsTime"
              ,"SpeakerConsTime","TotalCons"]]
y = dataset["RealTotalCons"]
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(x, y,
                                                test_size=0.2,
                                               random_state=42)
print(xtrain.shape)
print(ytrain.shape)
print(xtest.shape)
print(ytest.shape)
```

Le tableau 2.4 présente nos résultats.

```
(1801, 11)
(1801,)
(451, 11)
(451,)
```

Tableau 2.5: Tableau des résultats

Comme il s'agit d'un problème de régression, nous choisirons l'algorithme de régression Random Forest pour entraîner le modèle de prédiction du prix de l'électricité. Notons que d'autres méthodes d'apprentissage automatique ne sont pas utilisables pour notre état de

travail car elles fonctionnent sur des valeurs logiques et des types préticulaires, peu importe que ce type soit le seul qui puisse nous être utile dans ce projet, c'est pourquoi nous n'utiliserons que ce préticulaire méthode en notant que c'est une méthode très précise.

```
from sklearn.ensemble import RandomForestRegressor
model1 = RandomForestRegressor()
model1.fit(xtrain, ytrain)
```

Nous introduisons nos valeurs dans le tableau d'entrée et la machine nous calcule la valeur prédite de la consommation dans la journée demandée. Comme exemple, dans la journée 1 janvier, la consommation prédite est aux alentours de 10392.88076223 watts.

```
features = np.array([[1, 1,17 ,4,1,5,20,1,6,4,10848]])
model1.predict(features)
```

```
array([10392.88076223])
```

2.11 Conclusion

Dans ce chapitre, nous avons décrit notre projet et les différents outils utilisés. Nous avons estimé la consommation de l'électricité le 1 janvier à 10392.88 watts.

Conclusion générale

Dans ce projet, nous avons prédit la facture d’électricité à n’importe quelle date de l’année et pour n’importe qu’elle délai. Ceci en utilisant le domaine de l'apprentissage automatique. Dans ce mémoire, nous avons passé d'abord en revue les concepts fondamentaux de cet apprentissage. Puis, nous avons créé une série temporelle est une succession de données classées chronologiquement et espacées à intervalles égaux ou inégaux. Et ensuite, nous avons appliqué nos méthodes de prévision pour prévoir la demande horaire d'électricité.

Références

[1]- https://datascientest.com/pandas-python-data-science

[2]- https://courspython.com/apprendre-numpy.html

[3]- https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.html

[4]- https://seaborn.pydata.org/

[5]- https://docs.python.org/3/library/re.html

[6]- https://www.nltk.org/

[7]- https://docs.python.org/3/library/warnings.html

[8]- https://realpython.com/train-test-split-python-data/

[9]- https://scikit-learn.org/stable/modules/ensemble.html