



MÉMOIRE DE FIN D'ÉTUDES

Pour l'obtention du diplôme
INGÉNIEUR D'ÉTAT

Filière : Génie Industriel
Option : Management Industriel

Mise en place d'un modèle d'apprentissage automatique pour la consommation d'énergie des bâtiments

Effectué à : FIND SOLUTIONS

Réalisé par :

- **ARIDAL Sami**

Soutenu le Juillet 2023 devant les membres de jury

Pr. BENMILOU Ibtissam	Encadrante académique	(ENSMR)
Pr. LEBBAR Maria	Encadrante académique	(ENSMR)
M. HAJJAJI Imad	Parain	(FIND SOLUTIONS)
Pr. -----	Examinateuse	(ENSMR)
Pr. -----	Président	(ENSMR)

Année Universitaire : 2022-2023



Dédicaces

A mon père et ma mère

A mes chères sœurs Sanae et Meryem

À toute ma famille, mes amis et tous ceux que j'aime.



Remerciements

Au terme de ce travail, je souhaite transmettre mes remerciements les plus sincères à toutes les personnes qui m'ont aidées et soutenues tout au long de la réalisation de ce projet et qui ont eu l'amabilité de faire de ce stage une période profitable.

Je remercie en premier lieu et en particulier Madame Lebbar Maria et Benmiloud Ibtissam, mes encadrantes académiques, pour leur patience, leur encouragement, leurs conseils pertinents et leurs directives qui ont été des facteurs déterminants dans la réussite du projet. Leur expertise et leur soutien constant ont été d'une valeur inestimable tout au long de cette expérience. Je leur suis profondément reconnaissant(e) pour leur contribution précieuse.

Je remercie Monsieur HAJJAJI Imad, pour m'avoir honoré en acceptant de m'encadrer pendant la période de mon stage, grâce à qui j'ai pu mener à bien ce travail, pour sa motivation professionnelle, ses conseils et critiques constructives, ses corrections, sa gentillesse ainsi pour le temps qu'il a consacré à la réalisation de ce projet.

Que les membres de jury trouvent ici l'expression de mes reconnaissances pour avoir accepté de juger mon travail.

Je remercie chaleureusement tout le corps professoral du département génie industriel pour la formation de très bonne qualité que j'ai reçue durant ces années.

Je souhaite faire part de ma gratitude à l'ensemble du personnel de FIND SOLUTIONS, pour leur accueil et leur amabilité. Que tous ceux et celles, qui ont contribué de près ou de loin à l'accomplissement de ce travail, trouvent l'expression de mes remerciements les plus chaleureux.



Résumé

Ce rapport présente la synthèse d'un projet de fin d'études réalisé à FIND SOLUTIONS, visant à développer un modèle prédictif de la consommation d'électricité à partir de données collectées par des capteurs intelligents. Le projet se concentre sur le prétraitement de données de séries temporelles régulièrement espacées et sur l'entraînement de modèles d'apprentissage automatique, y compris de nouvelles méthodes d'apprentissage profond pour la prédiction de la consommation. Il implique différentes approches pour aborder les différentes étapes d'un projet d'apprentissage automatique. Ces approches incluent le prétraitement des données, l'analyse exploratoire et la prévision de base. En outre, il englobe la régression des séries temporelles, y compris les méthodes basées sur les arbres, ainsi que l'ingénierie des caractéristiques. De plus, il nécessite la formation et l'évaluation des modèles, ainsi que l'étude des modèles de prévision. Enfin, il requiert également l'exploration des stratégies d'apprentissage profond. Les modèles sélectionnés ont démontré de bons résultats en termes de métriques de performance, en particulier le LightGBM. De plus, les approches d'apprentissage profond ont également montré des résultats prometteurs, indiquant une précision accrue. Cette étude examine un ensemble de données collectées auprès de plusieurs foyers, avec un taux d'échantillonnage de 10 secondes, comprenant des mesures de la puissance réelle, de la tension (Vrms) et de la date/heure, dans le but d'analyser et de comprendre les schémas de consommation d'énergie.

Mots-clés : Modélisation prédictive, Consommation d'électricité, Capteurs intelligents, Données de séries temporelles, Apprentissage automatique, Apprentissage profond, Prétraitement des données, Ingénierie des caractéristiques, Entraînement des modèles, Évaluation des modèles, Modèles de prévision, Mesures de performance, Analyse de la consommation d'énergie, Consommation d'énergie des foyers, Analyse des données des capteurs, Analyse des données temporelles.



Abstract

This report presents the synthesis of a final year project conducted at FIND SOLUTIONS, aiming to develop a predictive model for electricity consumption using data collected by smart sensors. The project focuses on preprocessing regularly spaced time series data and training machine learning models, including novel deep learning methods for consumption prediction. It involves various approaches to address different stages of a machine learning project, including data preprocessing, exploratory analysis, and basic forecasting. Furthermore, it encompasses time series regression, including tree-based methods, as well as feature engineering. Additionally, it requires model training and evaluation, as well as studying forecasting models. Lastly, it also involves exploring deep learning strategies. The selected models have shown good results in terms of performance metrics, particularly LightGBM. Moreover, deep learning approaches have also demonstrated promising results, indicating improved accuracy. This study examines a dataset collected from multiple households, with a sampling rate of 10 seconds, including measurements of real power, voltage (Vrms), and date/time, with the aim of analyzing and understanding energy consumption patterns.

Keywords: Predictive modeling, Electricity consumption, Smart sensors, Time series data, Machine learning, Deep learning, Data preprocessing, Feature engineering, Model training, Model evaluation, Forecasting models, Performance metrics, Energy consumption analysis, Household energy consumption, Sensor data analysis, Temporal data analysis.



الملخص

يقدم هذا التقرير ملخصاً لمشروع السنة النهائية الذي تم إجراؤه في شركة FIND SOLUTIONS ، بهدف تطوير نموذج تنبؤي لاستهلاك الكهرباء باستخدام البيانات المجمعة من حساسات ذكية. يركز المشروع على معالجة البيانات الزمنية المرتبة بانتظام وتدريب نماذج التعلم الآلي، بما في ذلك أساليب التعلم العميق الجديدة لتوقع الاستهلاك. يتضمن المشروع مجموعة متنوعة من النهج لمعالجة مراحل مختلفة في مشروع التعلم الآلي، بما في ذلك معالجة البيانات المسبقة، والتحليل الاستكشافي، والتنبؤ الأساسي، والتحليل الزمني للسلال (بما في ذلك أساليب الأشجار المستندة إلى النموذج)، وهندسة الميزات، وتدريب النماذج وتقييمها، ودراسة نماذج التوقع، واستكشاف استراتيجيات التعلم العميق. أظهرت النماذج المختارة نتائج جيدة من بالإضافة إلى ذلك، أظهرت النهج LightGBM حيث المقاييس الأداء، ولاسيما نموذج المبتكرة للتعلم العميق نتائج واعدة تشير إلى زيادة الدقة. يقوم هذا الدراسة بفحص مجموعة بيانات جمعت من عدة منازل، بمعدل عينة يبلغ 10 ثوانٍ، تتضمن قياسات القدرة الفعلية، والجهد ، والتاريخ/الوقت، بهدف تحليل وفهم أنماط استهلاك الطاقة(Vrms) ،

الكلمات المفتاحية: نمذجة تنبؤية ، استهلاك

الكهرباء ، حساسات ذكية ، بيانات السلال الزمنية ، التعلم الآلي ، التعلم العميق ، معالجة البيانات ، هندسة الميزات ، تدريب النماذج ، تقييم النماذج ، نماذج التوقع ، مقاييس الأداء ، تحليل استهلاك الطاقة ، استهلاك الطاقة المنزلية ، تحليل بيانات الحساسات ، تحليل البيانات الزمنية.



Liste des abréviations

Acronyme	Désignation
FB	Forecast Bias
GRU	Gated Recurrent Unit
L1	L1 Regularization
L2	L2 Regularization
LIGHTGBM	Light Gradient Boosting Machine
LSTM	Long Short-Term Memory
Lasso	Lasso Regression
MASE	Mean Absolute Scaled Error
MAE	Mean Absolute Error
MSE	Mean Squared Error
R2	Coefficient of Determination
RF	Random Forest
RNN	Recurrent Neural Networks
RL	Régression linéaire (Linear Regression)
RMSE	Root Mean Square Error
RSE	Relative Squared Error
Ridge	Ridge Regression
XGBoost	Extreme Gradient Boosting



Liste des figures

Figure 1: Charte du projet : Mise en place d'un modèle d'apprentissage automatique pour la consommation d'énergie des bâtiments.....	17
Figure 2:Cycle de vie d'un projet Machine Learning (https://www.eurodecision.com)	18
Figure 3:L'outil Microsoft Teams	20
Figure 4:Exemples de différents modèles dans les données de séries temporelles (Hyndman & Athanasopoulos, 2018).....	23
Figure 5:Schéma du réseau neuronal à deux couches	26
Figure 6:Catalogue du MORED	31
Figure 7:Architecture d'acquisition MORED de la consommation électrique de l'ensemble des locaux (WP) et des charges individuelles (IL)	33
Figure 8:Locaux ciblés pour l'acquisition de données MORED	34
Figure 9:Les trois différents types de données de consommation d'électricité contenues dans MORED.....	35
Figure 10:Arbre décrivant le contenu du répertoire MORED	36
Figure 11:extrait de notre jeu de données	37
Figure 12:distribution des valeurs manquantes dans les Dataframes	38
Figure 13:consommation de WPILGT_2 entre 2020-10-11 et 2020-10-16(série imputée de données)	39
Figure 14:séries imputées complétées en utilisant un jour avant	39
Figure 15:séries imputées complétées en utilisant la moyenne pour compléter les données manquantes	40
Figure 16:séries imputées complétées par la moyenne horaire pour chaque jour de la semaine	40
Figure 17:Visualisation des comparaisons de méthodes	41
Figure 18:Erreur absolue moyenne (EAM) pour différentes méthodes d'imputation	42
Figure 19:Consommation d'énergie pour WP6	45
Figure 20:Consommation moyenne horaire	46
Figure 21:Le tracé des prédictions du modèle « Modèle de régression linéaire (WP_6)».....	51
Figure 22:L'importance des caractéristiques à utiliser dans le modèle.....	52
Figure 23 : Graphique des contours de la fonction d'erreur non régularisée [19]	53
Figure 24:Résultats régression linéaire	53
Figure 25:L 'importance des caractéristiques à utiliser dans le modèle.....	54
Figure 26:Résultats du modèle Régression Linéaire Lasso.....	54
Figure 27:L'importance des caractéristiques à utiliser dans le modèle.....	55
Figure 28:Arbre décision.....	56
Figure 29:L'importance des features	56
Figure 30:Random Forest.....	57
Figure 31:L'importance des features	58
Figure 32:Résultats LightGBM	59
Figure 33: Importance des Features de LightGBM.....	60
Figure 34Procédure récurrente l'entraînement du modèle RNN.....	61
Figure 35 Prévision comparée aux valeurs actuelles en vert.....	62
Figure 36 Paramètres du modèle RNN.....	63
Figure 37 Prédiction sur une vaste intervalle.....	63



Figure 38 : Résultats des métriques d'évaluation.....	64
Figure 39 Visualisation de la plage des dates de l'ensemble des données.....	76
Figure 40 Profile journalier en une résolution mensuelle	79
Figure 41 Boîte à moustaches du profile des jours du mois	79
Figure 42Graphic thermique du profile journalier en une résolution hebdomadaire	80
Figure 43 Décomposition Saisonnalité et Tendance par Loess.....	81
Figure 44 Décomposition Saisonnalité et Tendance par Loess.....	82
Figure 45 Décomposition de la saisonnalité et de la tendance à l'aide de Loess et des termes de Fourier (Décomposition de Fourier)	84
Figure 46 Décomposition de la saisonnalité et de la tendance à l'aide de Loess et des termes de Fourier (Décomposition de Fourier)	84
Figure 47 La décomposition de la saisonnalité et de la tendance à l'aide de Loess et des termes de Fourier (Décomposition de Fourier)	85
Figure 48 Visualisation du resultat de la méthode de l'écart-type	86
Figure 49 Visualisation du resultat de les l'écart interquartile	87
Figure 50 Prevision model Naive	89
Figure 51 Prevision model Naive avec transformation de la cible	90
Figure 52 Prevision du model de la moyenne mobile	91
Figure 53Prevision du modèle naïve saisonnier.....	91
Figure 54 Prevision du modele theta.....	92
Figure 55 Prevision du modele FFT	92
Figure 56 Visualisation de la représentation des caractéristiques temporelle	103
Figure 57 Graphic des caractéristiques avec leur importance (régression simple)	107
Figure 58 Graphic des caractéristiques avec leur importance (régression régularisée en norme L2)	108
Figure 59 Résultat de la prédiction comparé au valeurs réelle (régression Ridge)	108
Figure 60 Graphic des caractéristiques avec leur importance (régression régularisée en norme L1)	109
Figure 61 Résultat de la prédiction comparé au valeurs réelle (régression Lasso)	110
Figure 62 Graphic des caractéristiques avec leur importance (arbre de décision)	111
Figure 63 Résultat de la prédiction comparé au valeurs réelle (arbre de décision)	112
Figure 64 Graphic des caractéristiques avec leur importance (forêt aléatoire)	114
Figure 65 Résultat de la prédiction comparé au valeurs réelle (forêt aléatoire)	114
Figure 66 Graphic des caractéristiques avec leur importance (forêt aléatoire XGB).....	116
Figure 67 Résultat de la prédiction comparé au valeurs réelle (forêt aléatoire XGB)	116
Figure 68 Graphic des caractéristiques avec leur importance (LightGBM).....	117
Figure 69 Résultat de la prédiction comparé au valeurs réelle (LightGBM)	117
Figure 70:Logo DARTS	118
Figure 71:Logo Scikit-learn	119
Figure 72 Structure du dossier du projet	121



Liste des tableaux

Tableau 1:Résumé des travaux connexes	27
Tableau 2:Principales caractéristiques des librairies d'analyse des séries temporelles	119



Table des matières

Dédicaces	1
Remerciements	2
Résumé	3
Abstract	4
Liste des abréviations	6
Liste des figures	7
Liste des tableaux	9
Table des matières	10
Introduction Générale	12
<i>Chapitre 1 : Aperçu général du le projet</i>	14
1. Introduction	15
2. Présentation de l'organisme d'accueil	15
2.1. Présentation du FIND SOLUTIONS	15
2.2. Activités du FIND SOUTIONS	15
3. Cadre et sujet du projet	15
3.1. Formulation de la problématique	15
3.2. Objectifs	16
4. Démarche de conduite de projet	18
5.1 Le cycle de vie du projet ML	18
5.4. Outil de communication	19
5. Conclusion	20
<i>Chapitre 2 : Présentation des méthodes et modèles de prévision des séries temporelles.</i>	21
1. Introduction	22
2. Définitions et concepts	22
2.1 Série temporelle :	22
2.2 Prévisions :	23
2.3 Méthodes de prévision de la charge	24
2.3.1 Horizons de planification	24
2.3.2 Prévision de la consommation d'énergie à court terme	24
2.3.3 Analyse des séries chronologiques et approches traditionnelles	25
3. Résumé des travaux connexes :	27
4. Conclusion :	29
<i>Chapitre 3 : Préparation des donnée et ingénierie des fonctionnalités</i>	30



1. Introduction	31
2. MORED : A Moroccan Buildings' Electricity Consumption Dataset	31
2.1. Types de données sur la consommation d'énergie	31
2.2. Database MORED	32
3. Zone d'étude	33
4. L'acquisition de données	33
6. Exploration des données	36
7.1. Concaténation de données	36
7.2. Nettoyage des données	38
7.2.1. <i>Méthodologies utilisées pour traiter les valeurs manquantes (pour de longues périodes) :</i>	38
7.2.2. Suppression des doublons	42
7.3 Standardisation des données	42
7.4 Sélection des fonctionnalités (Feature Selection)	43
7.4.1 Procédure Sélection des fonctionnalités :	43
7.4.2 Procédure Evaluation des fonctionnalités	44
7.5 Partitionnement des données	44
7.6 Analyse et visualisation des données de séries temporelles	45
7. Conclusion	46
<i>Chapitre 4 : Modèles Machine Learning pour la prédiction de la consommation et l'analyse des résultats obtenus</i>	47
1. Introduction	48
2. Application des modèles Machine Learning	48
2.1 Entraînement du modèle	48
2.2. Métriques d'évaluation	48
2.3. Résultats des modèles	49
2.3.1 Résultats du modèle de Régression Linéaire	49
2.3.2 Résultats du modèle Arbre de décision	55
2.3.3 Résultats du modèle de régression par forêt aléatoire (Random Forest)	56
2.3.4 Résultats des modèles XGBoost et LightGBM :	58
2.3.6 Résultats du modèle de réseau neuronal récurrent (RNN)	60
2.3.8 Procédure d'entraînement sur tous l'ensemble des données	64
2.4. Comparaison entre des modèles en termes de métriques d'évaluation	64
3. Conclusion	67
Conclusion Générale	68
Références	69
ANNEXES	71



Introduction Générale

L'électricité est le pilier de l'économie nationale et ses services durables sont à la base de la vie normale de la population. La demande d'énergie électrique continue de croître rapidement, en raison de plusieurs facteurs dont les plus importants sont l'expansion démographique et développement technologique. Entre 2003 et 2014, l'augmentation annuelle moyenne de la consommation d'électricité a été de 6,5 % ([Maroc - Pays et régions](https://www.iea.org/countries/morocco) : <https://www.iea.org/countries/morocco>). Cette croissance a été principalement attribuée à l'augmentation de la charge industrielle, à l'expansion économique et à l'électrification rurale. En raison d'un léger ralentissement de la croissance économique, de l'augmentation de l'efficacité énergétique et de la saturation de la charge d'électrification entre 2015 et 2019, la demande n'a augmenté que de 3,1 % en moyenne, atteignant un taux de 99,72 % à la fin de 2019 ("ONEE Data, Site Web Officiel de l'ONEE - Branche Electricité" 2021).

Les clients résidentiels, commerciaux, éclairage public, administration, industriels et agricoles sont répartis en différentes catégories selon la classification marocaine.

Ce document explore l'analyse des données de séries temporelles, en mettant l'accent sur différentes techniques utilisées dans ce domaine. Les méthodologies classiques telles que l'ARIMA (AutoRegressive Integrated Moving Average) et le lissage exponentiel (Exponential Smoothing) servent de bases solides pour les prévisions, tandis que l'apprentissage automatique et apprentissage profond est exploité pour les approches contemporaines. Les techniques d'ingénierie des caractéristiques telles que les caractéristiques de retard (Lag Features), les caractéristiques de roulement (Rolling Features) sont utilisées pour transformer le problème en régression. Des modèles d'apprentissage automatique tels que la régression linéaire (Linear Regression), les forêts aléatoires (Random Forests) et les arbres de décision boostés par le gradient (Gradient-Boosted Decision Trees) sont entraînés, et des techniques d'amélioration de la précision sont étudiées. Les modèles de réseaux neuronaux récurrents (RNN) et de mémoire à long terme (LSTM) ont été utilisés et évalués pour la prévision, démontrant leur capacité à capturer les dépendances temporelles et à traiter les données séquentielles. Toutefois, pour améliorer la qualité et la précision des prévisions, il est recommandé d'explorer d'autres modèles prometteurs spécialisés, tel que les modèles de séquence à séquence (Seq2Seq) et les modèles Transformer. Ces perspectives d'avenir offrent des possibilités intéressantes pour la recherche sur la modélisation prédictive. L'analyse des mesures de



performance met en évidence l'importance des modèles d'apprentissage automatique et d'apprentissage profond pour améliorer la précision des prévisions, tout en tenant compte de la complexité et des ressources nécessaires à leur mise en œuvre.

Chapitre 1 illustre un aperçu général sur le projet.

Chapitre 2 Présentation des méthodes et modèles de prévision des séries temporelles.

Chapitre 3 introduit le prétraitement des données utilisées et l'augmentation des données par la construction de nouvelles caractéristiques.

Chapitre 4 propose plusieurs modèles de machine Learning et l'analyse des résultats obtenus.



Chapitre 1 : Aperçu général du le projet

1. Introduction

Ce chapitre a pour objectif de présenter, dans un premier lieu, l'organisme d'accueil, son histoire et ses activités. Dans un deuxième lieu, il présente une description du projet, les objectifs attendus et les phases d'élaboration du projet.

2. Présentation de l'organisme d'accueil

2.1. Présentation du FIND SOLUTIONS

FIND SOLUTIONS est un groupe d'ingénieurs juniors et seniors aux profils diversifiés, composé de 13 personnes, qui ont décidé de créer leur propre start-up en 2019 pour mettre à profit leurs connaissances et leurs expériences en tant qu'indépendants. Certes, l'entreprise est nouvelle, cependant elle est experte dans le domaine de la donnée et de ses applications.

2.2. Activités du FIND SOUTIONS

Ils ont commencé en tant que groupe travaillant dans le centre de recherche de l'UM6P où ils ont développé leurs compétences en travail d'équipe et comment traiter avec les clients ; comprendre leurs besoins et trouver les solutions appropriées. À ce stade, ils ont eu l'occasion de travailler avec différents clients de différents pays .

Ils ont eu l'opportunité d'évoluer dans différents domaines, à savoir la finance, le sport, la logistique, la gestion des stocks, le traitement d'images et de vidéos, les systèmes de recommandation, l'automatisation des processus métier, le smart grid et les solutions d'aide à la décision.

La diversité des projets nécessitait différents outils à savoir :

Machine Learning, Deep Learning, Data Analysis and Visualization, Statistical Modeling, Optimization, Reporting, Data Collection, Data Warehouse, Data Mining et Software Development.

3. Cadre et sujet du projet

3.1. Formulation de la problématique

Les bâtiments représentent plus de 40 % de la consommation mondiale d'énergie primaire et 36 % des émissions totales de CO₂, et les bâtiments intelligents peuvent réduire considérablement la

consommation d'énergie en contrôlant plus efficacement les systèmes du bâtiment. La complexité croissante de l'environnement bâti rend les systèmes d'automatisation conventionnels inefficaces. De plus, le comportement des occupants, la demande accrue de confort, les prix dynamiques de l'énergie et les systèmes Chauffage, Ventilation et Climatisation sophistiqués ont rendu difficile le contrôle des bâtiments.

3.2. Objectifs

Les prévisions de charge et de consommation sont essentielles pour gérer efficacement les ressources énergétiques. Les bâtiments sont de grands consommateurs d'énergie pendant leur phase d'exploitation, et il est essentiel de prévoir avec précision leurs besoins énergétiques pour atteindre l'efficacité énergétique. L'intégration des énergies renouvelables peut contribuer à réduire la consommation globale, mais leur flexibilité limitée nécessite une adaptation efficace tout en assurant une gestion optimale de la charge.

Avec les progrès des capteurs à faible coût et des systèmes de stockage numérique de grande capacité, l'adoption des technologies numériques et de l'internet des objets (IOT) a connu une croissance significative. Cela a permis la collecte et le traitement de nombreuses données liées à la consommation d'énergie, aux facteurs environnementaux et à d'autres paramètres pertinents. En tirant parti de ce panel d'ensemble de données, il est désormais possible d'employer des techniques d'apprentissage automatique (ML) et d'apprentissage profond (DL) pour développer des modèles prédictifs de prévision de la charge et de la consommation en temps réel, y compris les modèles de base tels que le modèle naïf et la moyenne mobile saisonnière. Ces modèles de base jouent un rôle essentiel en fournissant une référence simple et intuitive pour évaluer les performances des modèles plus avancés. Ils permettent de comparer les résultats obtenus par les modèles plus complexes et sophistiqués, offrant ainsi une mesure de la valeur ajoutée de ces approches plus avancées.

Le comportement des occupants, influencé par divers facteurs, a un impact significatif sur les modèles de consommation d'énergie. Il est essentiel de comprendre et de modéliser ce comportement pour établir des prévisions de charge précises.

En analysant les données collectées par divers capteurs situés dans différents bâtiments résidentiels dans différentes régions du Maroc, les caractéristiques des bâtiments et les conditions environnementales, les modèles d'apprentissage automatique peuvent capturer des modèles complexes et des dépendances, permettant des prévisions précises des besoins énergétiques futurs.

Le diagramme suivant, figurant à la figure 1, illustre le cheminement de notre projet, il met en évidence les différentes phases du projet d'apprentissage automatique, en présentant la formulation du problème, les objectifs, l'exploration des données, le développement du modèle et le rapport final. Les groupes sont organisés à l'aide d'étiquettes claires et d'une disposition visuelle pour décrire le déroulement du projet.

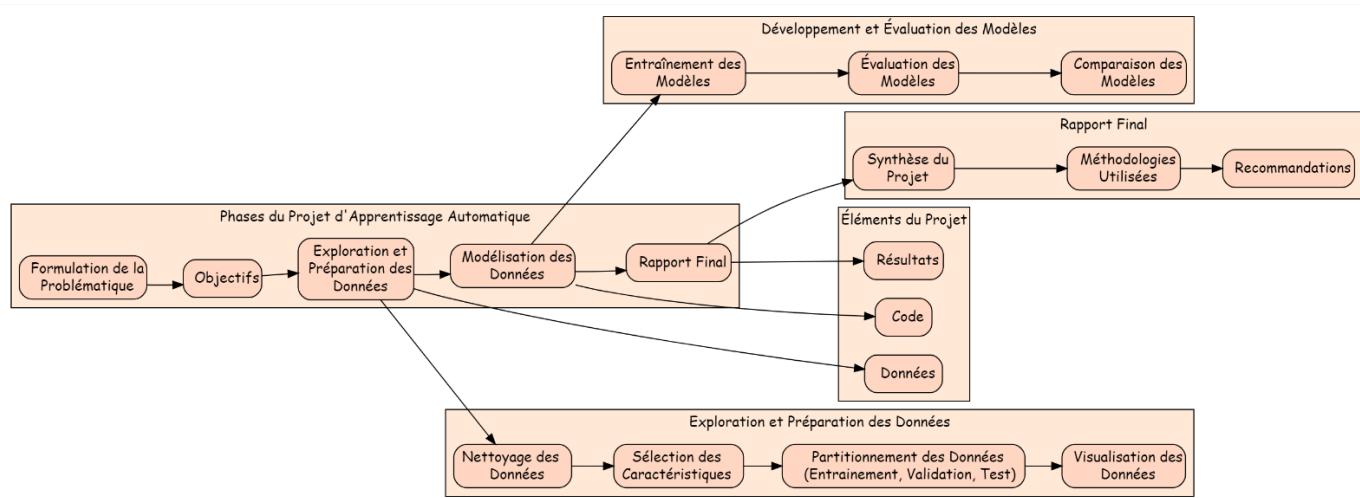


Figure 1: **Charte du projet : Mise en place d'un modèle d'apprentissage automatique pour la consommation d'énergie des bâtiments**

4. Démarche de conduite de projet

4.1. Le cycle de vie du projet ML

Les projets basés sur la science des données adoptent les étapes illustrées dans la figure 2.

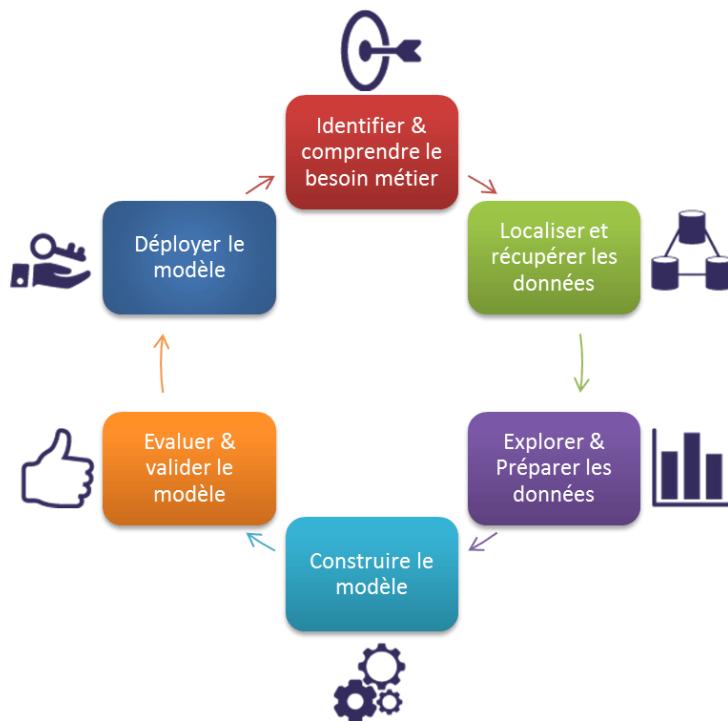


Figure 2: Cycle de vie d'un projet Machine Learning (<https://www.eurodecision.com>)

- *Définition du besoin (identifier et comprendre le besoin métier)*

Le but de cette étape est de comprendre les besoins exprimés, formuler la problématique ainsi que définir les objectifs souhaités à travers ce projet. Vu qu'un projet ML implique l'utilisation des données, il est nécessaire de bien choisir les données qui vont être utilisées pour la phase suivante. Par conséquent, il est primordial de justifier les champs de données que nous allons utiliser.

- *Collecte des données (localiser et récupérer les données)*

L'étape initiale consiste à extraire les données pertinentes du data Warehouse, ce qui implique de déterminer le volume nécessaire pour comprendre les données en se référant au dictionnaire data Warehouse fourni par l'entreprise. Il est essentiel d'utiliser les outils appropriés afin de maintenir la qualité des données et d'obtenir les informations nécessaires pour construire notre modèle.

- *Préparation et exploration des données (Explorer et préparer les données)*

La préparation des données est le processus qui consiste à préparer les données pour l'analyse en supprimant ou en modifiant les données qui sont incorrectes, incomplètes, non pertinentes, dupliquées ou mal formulées.

L'exploration des données dite aussi “Analyse Exploratoire des données”, est un processus par lequel on cherche à explorer les données à l'aide des méthodes de statistiques et de visualisation. Cette étape permet d'identifier en principe les caractéristiques à utiliser dans les étapes suivantes. A savoir que les caractéristiques sont des variables indépendantes qui jouent le rôle des entrées de notre système. En bref, les modèles utilisent ces caractéristiques pour faire des prédictions. Les caractéristiques sont très importantes dans la construction du modèle, car leur qualité a un impact sur la qualité des informations que nous allons en tirer.

- *Modélisation (Construire le modèle)*

Cette étape représente le cœur du processus. A cette étape, on utilise comme entrée les caractéristiques que nous avons choisies au niveau de l'étape précédente afin de fournir les résultats souhaités. Elle comprend le choix du modèle adéquat selon le problème traité, s'il s'agit d'un modèle supervisé ou non, est ce qu'il s'agit d'une classification ou d'une régression.

- *Analyse et test des résultats (Evaluer et valider le modèle)*

Il est indéniable que les résultats obtenus peuvent ne pas être assez précis que souhaitable. Raison pour laquelle il est nécessaire de les tester en calculant les indicateurs de performance adéquats et évaluer les performances du modèle.

- *Déploiement du modèle (déployer le modèle)*

Après évaluation du modèle, il est enfin déployé. Il s'agit de la dernière étape du cycle de vie de ce projet.

4.2. Outil de communication

Microsoft Teams est une application de communication collaborative propriétaire officiellement lancée par Microsoft en novembre 2016.

Le service s'intègre à la suite Microsoft Office 365 et Skype en proposant des extensions pouvant

être intégrées à des produits autres que Microsoft. Elle est actuellement disponible dans 181 pays et traduite en 25 langues.



Figure 3:L'outil Microsoft Teams

5. Conclusion

Dans ce chapitre, nous avons présenté le cadre et le contexte général de notre projet, en présentant l'entreprise d'accueil, la problématique et la méthodologie adoptée pour mettre en œuvre la solution proposée en collaboration avec tous les membres de l'équipe.



Chapitre 2 : Présentation des méthodes et modèles de prévision des séries temporelles.

1. Introduction

Ce chapitre expose le contexte des données de séries temporelles, en mettant en avant l'analyse, les prévisions, ainsi que les travaux connexes pertinents dans le domaine de la recherche sur la consommation d'électricité. Il aborde initialement les concepts fondamentaux des données de séries temporelles, de l'analyse et des prévisions. Par la suite, il présente les méthodes couramment utilisées dans le domaine de la consommation d'énergie à court terme, en fournissant un résumé de leurs résultats. De plus, une exploration des avancées de la modélisation, passant des méthodes classiques aux techniques d'apprentissage automatique (Machine Learning) et d'apprentissage profond (Deep Learning), est également présentée.

2. Définitions et concepts

2.1 Série temporelle :

Une série temporelle est une collection de points de données indexés et recueillis dans un ordre chronologique. Les séries temporelles peuvent prendre différentes formes, telles que les séries univariées, multivariées et de panel, qui partagent des similitudes mais présentent des différences distinctes. Les séries univariées sont les plus courantes et se composent d'une seule variable liée à une seule instance dépendant du temps. Les séries multivariées se composent de deux variables ou plus dépendant du temps et d'une seule instance [1]. Les séries de panel, également connues sous le nom de données longitudinales, sont constituées d'une seule variable dépendant du temps pour deux instances ou plus [2].

Une série temporelle peut être définie mathématiquement comme un ensemble d'observations x_t enregistrées à des moments spécifiques t . Lors de l'analyse de données de séries temporelles, il est essentiel de considérer quatre modèles courants. Les tendances, qui peuvent être observées dans les données, révèlent des variations à long terme, pouvant évoluer d'une augmentation à une diminution et ne suivant pas toujours une progression linéaire. La saisonnalité, quant à elle, se manifeste par des variations régulières à des intervalles de temps spécifiques, ce qui peut être influencé par des facteurs tels que l'heure du jour, la semaine ou l'année. Cette notion de saisonnalité est illustrée dans la Figure 4 (en haut à gauche et en bas à gauche). La cyclicité fait référence aux fluctuations des données qui ne se produisent pas à des intervalles de temps réguliers et est représentée dans la Figure 4 (en haut à gauche). Enfin, le caractère aléatoire est visible dans la Figure 4 (en bas à droite), où aucun modèle apparent ne peut être distingué [3].

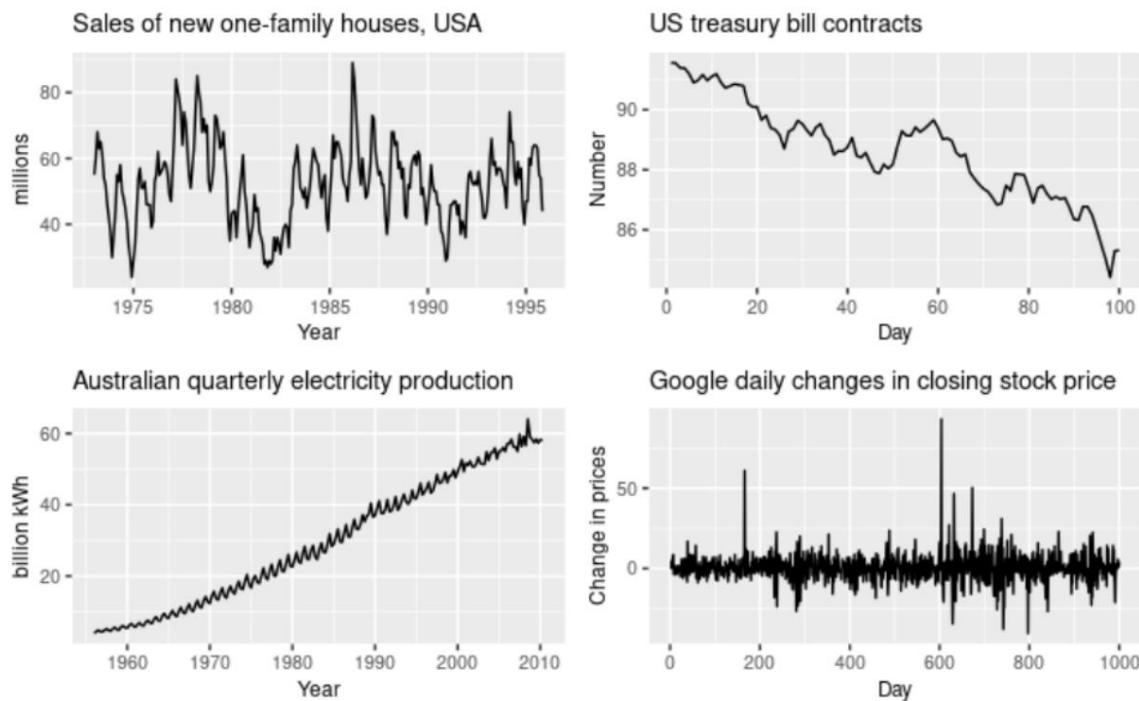


Figure 4: Exemples de différents modèles dans les données de séries temporelles (Hyndman & Athanasopoulos, 2018)

2.2 Prévisions :

Prévision, dans le contexte de l'analyse des séries temporelles, désigne le processus d'estimation ou de prédiction des valeurs ou des tendances futures en se basant sur les schémas observés dans les données historiques. Cela implique l'analyse du comportement passé d'une série temporelle afin de projeter les résultats futurs.

Terminologie de la prévision dans les séries temporelles :

- **Série temporelle** : Une collection de points de données indexés et ordonnés dans le temps.
- **Tendance** : Un schéma ou une direction à long terme dans les données, indiquant une augmentation ou une diminution constante au fil du temps.
- **Saisonnalité** : Des schémas réguliers ou des fluctuations qui se produisent à des intervalles fixes au sein d'une série temporelle, souvent influencés par des effets calendaires ou des événements récurrents.
- **Cyclicité** : Des schémas ou des fluctuations irrégulières qui se produisent à des intervalles irréguliers, généralement caractérisés par des hausses et des baisses périodiques.
- **Stationnarité** : La propriété d'une série temporelle dans laquelle les caractéristiques statistiques, telles que la moyenne et la variance, restent constantes au fil du temps.
- **Erreur de prévision** : La différence entre la valeur prédite et la valeur réelle d'une série

temporelle à un moment spécifique.

2.3 Méthodes de prévision de la charge

De nos jours, de nombreux pays sont confrontés à des changements structurels rapides sur leurs marchés de l'énergie, et cela est attesté par plusieurs facteurs tels que l'intensification des normes d'efficacité énergétique, les augmentations tarifaires liées aux investissements dans les infrastructures et la croissance des technologies de production distribuée, qui ont été identifiés comme ayant un impact significatif [4]. Par conséquent, la prévision de la demande électrique est devenue un défi complexe au cours des dernières années. Les compagnies d'électricité et les services publics ont été contraints d'adopter de nouvelles approches pour réduire le risque de construire des capacités inutiles et garantir une planification énergétique plus efficace.

2.3.1 Horizons de planification

La prévision de charge électrique s'effectue sur différents horizons temporels. Les prévisions à long terme couvrent plus de 10 ans, les prévisions à court terme se situent dans quelques jours à quelques semaines, tandis que les prévisions à moyen terme combinent l'écart entre les deux. Les prévisions à long terme guident la planification des infrastructures majeures, les prévisions à court terme facilitent les opérations quotidiennes, et les prévisions à moyen terme aident à la planification opérationnelle. Le choix de l'horizon dépend des besoins spécifiques de l'entreprise.

2.3.2 Prévision de la consommation d'énergie à court terme

La prévision de la consommation d'énergie, qui vise à anticiper la demande, constitue un domaine de recherche vaste et bien documenté. La littérature propose une variété de méthodes de prévision, couvrant à la fois les approches statistiques traditionnelles et les algorithmes d'apprentissage automatique plus avancés [5]. Les prévisions de consommation d'énergie sont généralement classées en trois catégories : à court terme (de quelques heures à une semaine), à moyen terme (d'une semaine à un an) et à long terme (plus d'un an). Les méthodes de prévision des séries temporelles se regroupent en deux grandes catégories : les méthodes d'apprentissage automatique et les méthodes traditionnelles de séries temporelles.

Cette section examine les différentes approches de modélisation utilisées dans les recherches antérieures sur la prévision à court terme de la consommation d'énergie :

2.3.3 Analyse des séries chronologiques et approches traditionnelles

L'analyse des séries temporelles consiste à prédire les tendances futures en se basant sur les données passées, en prenant en compte les corrélations avec les cycles diurnes, les conditions météorologiques et d'autres phénomènes cycliques, ainsi que l'impact des changements paramétriques sur la croissance de la charge. Cette approche, largement utilisée au milieu du 20e siècle, nécessite un volume réduit de données pour l'analyse, incluant les données météorologiques et les données historiques de la charge électrique. Les méthodes de séries temporelles se limitent aux données temporelles et à leur traitement statistique, avec une approche initialement axée principalement sur la régression :

➤ Les modèles autorégressifs

Les modèles autorégressifs sont une catégorie de modèles de séries temporelles. Dans les modèles autorégressifs simples, les valeurs passées de la charge (L_{t-i}) sont utilisées pour prévoir la charge future (L_t). Dans l'équation suivante, C est une constante, ϕ_1, \dots, ϕ_p sont les coefficients des demandes de décalage, et ϵ_t est l'erreur de prévision. L'ordre du modèle indique le nombre de retards (p) qui sont inclus.

$$L_t = C + \phi_1 L_{t-1} + \phi_2 L_{t-2} + \dots + \phi_p L_{t-p} + \epsilon_t$$

Les modèles de séries temporelles tels que la moyenne mobile autorégressive (ARMA) et la moyenne mobile intégrée autorégressive saisonnière (SARIMA) utilisent les valeurs passées de la charge en termes de demandes de décalage et de valeurs de décalage des erreurs. La composante moyenne mobile du modèle indique la dépendance de la charge (L_t) par rapport aux erreurs prévisionnelles retardées (ϵ_{t-j}). Combinant les modèles autorégressifs et de moyenne mobile, le modèle ARMA peut-être défini comme l'équation suivante, dans laquelle $\theta_1, \dots, \theta_q$ sont les coefficients de retard des erreurs de prévision incluses jusqu'à q nombres.

$$L_t = C + \phi_1 L_{t-1} + \dots + \phi_p L_{t-p} + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \dots + \theta_q \epsilon_{t-q}$$

- **Méthodes d'apprentissage automatique (Machine Learning)**

Ces dernières années, les méthodes d'apprentissage automatique ont gagné en popularité pour les prévisions de séries temporelles. Dans le domaine de la prévision de la consommation d'électricité à court terme, les modèles de réseaux de neurones artificiels (ANN) (voir la figure 5) sont largement utilisés, comme indiqué dans les revues documentaires [6].

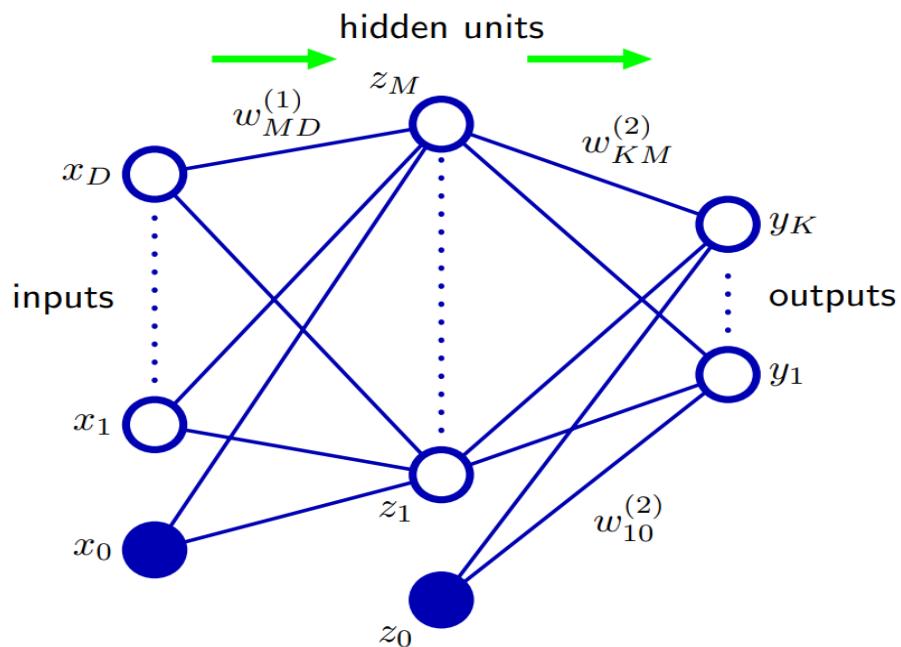


Figure 5: Schéma du réseau neuronal à deux couches

Cependant, il y a également diverses autres approches d'apprentissage automatique fréquemment employées, notamment la régression linéaire et ses variations, les machines à vecteurs de support (SVM), les méthodes d'ensemble et les forêts aléatoires.

Ces dernières années, les progrès réalisés dans les modèles d'apprentissage profond ont suscité un fort engouement, notamment grâce à l'émergence de nouvelles architectures sophistiquées telles que les LSTM, les GRU et les transformateurs. Ces avancées permettent de capturer de manière efficace les dépendances à long terme, tout en évitant les problèmes associés aux gradients de disparition.

3. Résumé des travaux connexes :

Le tableau présenté ci-dessus résume les résultats de plusieurs études portant sur la prévision des séries temporelles. Chaque ligne du tableau représente une étude spécifique menée par différents auteurs. Les auteurs ont utilisé diverses méthodes et modèles pour effectuer leurs prévisions et ont évalué les performances des modèles en utilisant des mesures telles que l'erreur relative, l'erreur moyenne de prévision et le pourcentage d'erreur absolue moyenne (MAPE).

Tableau 1: Résumé des travaux connexes

Auteurs	Horizon de la prévision	Données	Modèles	Évaluation de la prévision
Hor et al (2006) [7]	Journée	Données Historique et Socio-économiques	ARIMA utilisant le model GARCH	Entre 1% et 3% MAPE
Sp.Pappas et al (2008) [8]	Heure	Données historiques	SARIMA et MMPF	Erreur relative : SARIMA = 2,68 % MMPF = 1,02 %
Sp.Pappas et al (2010) [9]	Journée	Données historiques	ARMA en utilisant AIC, AICC, BIC et MMPF	MAPE : MMPF = 1.87% AIC = 2.35% AICC = 1.98% BIC = 2.11%
An et al (2013) [10]	Une demi-heure	Données historiques	MLP multi-output utilisant la décomposition Empirical mode	MAPE : 2.42%
Fattaheian- Dehkordi et al (2014) [11]	Heure	Données historiques et de température	Support vector regression (SVR)	MAPE : Entre 0,75 % et 1,46 %
Halepoto et al (2014) [12]	Heure	Données historiques et de température	Régression linéaire, régression linéaire multiple, régression quadratique et régression exponentielle	Erreur moyenne de prévision de 2,98 MSE

Din & Marnerides (2017) [13]	Jour et semaine	Historique, température, caractéristiques dérivées	Réseau de neurones profonds à Feed-forward (FF-DNN) et réseau de neurones profonds récurrents (R-DNN)	MAPE : FF-DNN = 0.067 - 1.42% R-DNN = 0.057 - 1.30%
Ouyang et al (2017) [14]	Journée	Données historiques, température et prix de l'électricité	Régression vectorielle de support (SVR), réseau neuronal (NN), machine d'apprentissage extrême (ELM), réseau de croyances (belief network) profondes (DBN) et Copula - réseau de croyances profondes (deep belief) (C-DBN).	MAPE : SVR = 5.78 - 7.12% NN = 5.53 - 6.89% ELM = 5.86 - 6.96% DBN = 5.05 - 6.28% C-DBN = 4.34 - 5.34%
Bashir et al (2022) [15]	15 minutes	Données historiques	ARIMA, LSTM, Prophet, hybride ARIMA-SVM et hybride Prophet-LSTM. Hybride Prophet-LSTM	MAPE : ARIMA = 5.19% LSTM = 3.72% Prophet = 5.21% ARIMA-SVM Hybride = 2.47% Prophet-LSTM Hybride = 0.49%

Par exemple, Hor et al (2006) ont utilisé le modèle ARIMA (AutoRegressive Integrated Moving Average) avec le modèle GARCH (Generalized Autoregressive Conditional Heteroskedasticity) pour prévoir des données historiques et socio-économiques sur une période d'une journée, obtenant des taux d'erreur compris entre 1% et 3% MAPE (Mean Absolute Percentage Error). Sp.Pappas et al (2008) ont utilisé les modèles SARIMA (Seasonal AutoRegressive Integrated Moving Average) et MMMPF (Mixed Model with Polynomials and Fourier) pour prévoir des données historiques à une échelle horaire, et ont obtenu des erreurs relatives de 2,68% pour SARIMA et 1,02% pour MMMPF. Dans une autre étude, Sp.Pappas et al (2010) ont utilisé le modèle ARMA (AutoRegressive Moving Average) en utilisant les critères d'information AIC (Akaike Information Criterion), AIICC (Corrected Akaike Information Criterion), BIC (Bayesian Information Criterion) et MMMPF, obtenant des taux d'erreur absolue moyenne (MAPE) de 1,87% pour MMMPF, 2,35% pour AIC, 1,98% pour AIICC et 2,11% pour BIC.

D'autres études ont utilisé des modèles tels que MLP multi-output, régression linéaire, régression quadratique, SVR, réseaux de neurones profonds à feed-forward (FF-DNN) et réseaux de neurones profonds récurrents (R-DNN), parmi d'autres. Chaque étude a évalué les performances des modèles en utilisant différentes mesures d'évaluation telles que MAPE, MSE et erreur relative.

4. Conclusion :

Ce chapitre a présenté le contexte des données de séries temporelles et a mis en évidence l'analyse, les prévisions et les travaux connexes pertinents dans le domaine de la recherche sur la consommation d'électricité. Il a commencé par introduire les concepts fondamentaux des données de séries temporelles, de l'analyse et des prévisions. Ensuite, il a fourni un aperçu des méthodes couramment utilisées dans le domaine de la prévision de la consommation d'énergie à court terme, en résumant leurs résultats.

Par la suite, une exploration des avancées de la modélisation a été présentée, passant des méthodes classiques aux techniques d'apprentissage automatique (Machine Learning) et d'apprentissage profond (Deep Learning). Cette exploration a permis de mettre en évidence les progrès réalisés dans le domaine de la prévision des séries temporelles en utilisant des approches plus avancées et plus sophistiquées.

En résumé, ce chapitre a permis de se familiariser avec les notions de base des données de séries temporelles et de comprendre les méthodes de prévision couramment utilisées dans le domaine de la consommation d'énergie. De plus, il a souligné l'importance des avancées en matière de modélisation, notamment grâce à l'utilisation de techniques d'apprentissage automatique et d'apprentissage profond.



Chapitre 3 : Préparation des données et ingénierie des fonctionnalités

1. Introduction

Le présent chapitre propose l'analyse et le prétraitement des données de la consommation d'énergie employé pour l'entraînement d'un modèle Machine Learning.

2. MORED : A Moroccan Buildings' Electricity Consumption Dataset

Depuis le printemps 2019, une campagne de collecte de données a été lancée pour obtenir des données sur la consommation électrique de bâtiments urbains dans différentes villes marocaines. Le jeu de données, appelé MORED, est le premier ensemble de données africain ouvert sur la consommation électrique des bâtiments. Il comprend des données étiquetées de consommation électrique réelle (WP et IL), des signatures électriques étiquetées (IL) et des données de consommation d'électricité de plusieurs ménages et charges au Maroc. L'objectif de ce jeu de données est de stimuler les progrès dans le domaine de la désagrégation énergétique en fournissant une plus grande quantité de données et en tirant parti des avancées récentes dans le domaine.

MORED

A Moroccan Buildings' Electricity Consumption Dataset. MORED is made available by TICLab of the International University of Rabat (UIR), and the data collection was carried out as part of PVBuild research project, coordinated by Prof. Mounir Ghogho and funded by the United States Agency for International Development (USAID).

[View the Project on GitHub](#)
[MOREDataset/MORED](#)

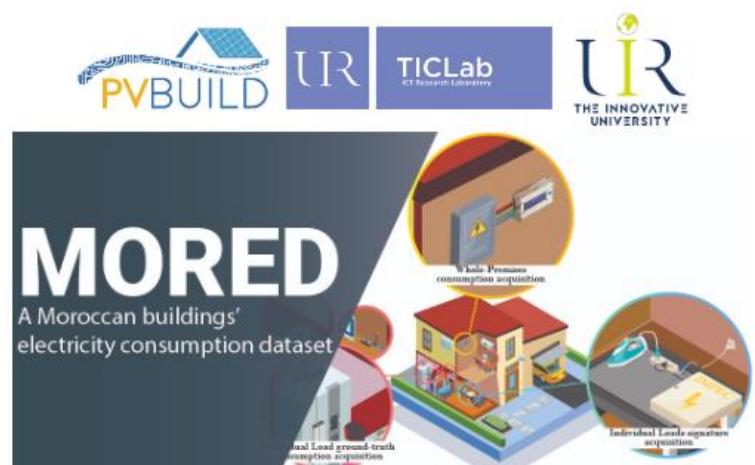


Figure 6: Catalogue du MORED

2.1. Types de données sur la consommation d'énergie

Il existe deux types d'ensembles de données pour la consommation d'énergie : les ensembles de données basés sur des événements (EB) et les ensembles de données sans événement (EL). Les ensembles



de données EB fournissent des événements supplémentaires qui indiquent les changements d'état de l'équipement. Ils comprennent des données de consommation générale (WP), des données de circuits individuels (IC) et des données de charges individuelles (IL).

Il est important de noter que :

- Les mesures WP représentent la consommation globale des locaux et sont prises à partir du tableau principal.
- Les mesures IL comprennent les signatures (ou traces) qui sont des schémas de consommation uniques des charges, surveillées individuellement pendant de courtes périodes, ainsi que les données de consommation véridiques (ou données au niveau de la prise) qui représentent la consommation des charges mesurées au niveau des prises individuelles d'un réseau de distribution. Ces mesures sont souvent prises en parallèle avec l'acquisition de la consommation WP.
- Les mesures IC représentent les mesures d'énergie au niveau des circuits du réseau électrique des locaux. Elles peuvent décrire la consommation d'une combinaison de charges ou celle de charges individuelles, ce qui les rend similaires aux mesures IL.

2.2. Database MORED

MORED est le premier ensemble de données collectées auprès des ménages marocains, fournissant trois types de mesures de consommation : WP, WP de base à faible taux (WPILGT) et IL, ainsi que des signatures IL à différents taux d'échantillonnage. Il comprend des listes d'événements décrivant les transitions marche/arrêt dans la consommation électrique WP de l'IL surveillé et diverses transitions d'état dans les signatures IL. La consommation électrique WP de MORED a été acquise en fonction de paramètres tels que le statut socio-économique du quartier et le nombre de résidents du ménage afin de capturer un échantillon représentatif de la consommation d'électricité. La consommation VA de MORED a été acquise dans des environnements résidentiels et industriels.

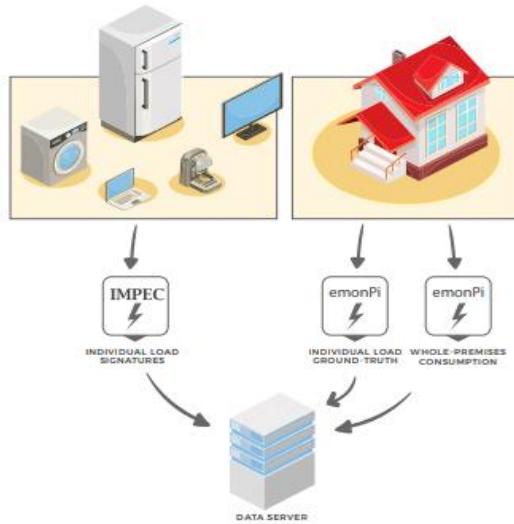


Figure 7: Architecture d'acquisition MORED de la consommation électrique de l'ensemble des locaux (WP) et des charges individuelles (IL)

3. Zone d'étude

Le Maroc est un pays d'Afrique du Nord, bordé par l'océan Atlantique et la mer Méditerranée, offrant une côte de 3500 km. Il s'étend sur une superficie de 710 850 km² entre le 37e et le 21e parallèle. Le pays est limitrophe du détroit de Gibraltar et de la mer Méditerranée au nord, de la Mauritanie au sud, de l'Algérie à l'est et de l'océan Atlantique à l'ouest. Le Maroc présente une grande diversité climatique, avec des régions côtières bénéficiant d'un climat tempéré, tandis que le sud et l'est du pays ont un climat désertique. Le climat varie également selon les saisons, avec des nuances méditerranéennes au nord, océaniques à l'ouest, continentales à l'intérieur des terres et sahariennes au sud.

4. L'acquisition de données

Durant le printemps et l'été 2019 et 2020, une campagne a été réalisée pour collecter des données sur la consommation électrique des bâtiments urbains au Maroc, **en garantissant le consentement et l'anonymat des participants pour respecter leur vie privée.**

- Echantillon démographique

La plupart des foyers marocains sont alimentés par un circuit électrique monophasé à deux fils de 220V 50 Hz, à l'exception des grandes maisons des quartiers aisés qui peuvent être alimentées en triphasé.

Le Maroc est un pays en développement dont une grande partie de la population a un faible pouvoir

d'achat ; par conséquent, la disponibilité et l'utilisation de charges gourmandes en énergie telles que les climatiseurs et les micro-ondes sont généralement limitées aux personnes aisées.

En outre, étant donné que les habitudes de consommation d'électricité et les types de charges peuvent différer en fonction du statut socio-économique des résidents, nous avons tenu compte de la taille, du type et de l'emplacement des locaux, afin d'obtenir un échantillon représentatif de la consommation d'électricité du pays.

Comme on peut le voir dans la figure 8, la campagne d'acquisition de MORED a ciblé des locaux résidentiels tels que des appartements et des maisons jumelées des quartiers aisés, et les appartements des quartiers défavorisés des zones urbaines marocaines, ainsi que les laboratoires des centres de recherche de l'Université Internationale de Rabat. Dans cette section, nous décrivons les mécanismes de la campagne d'acquisition des données.

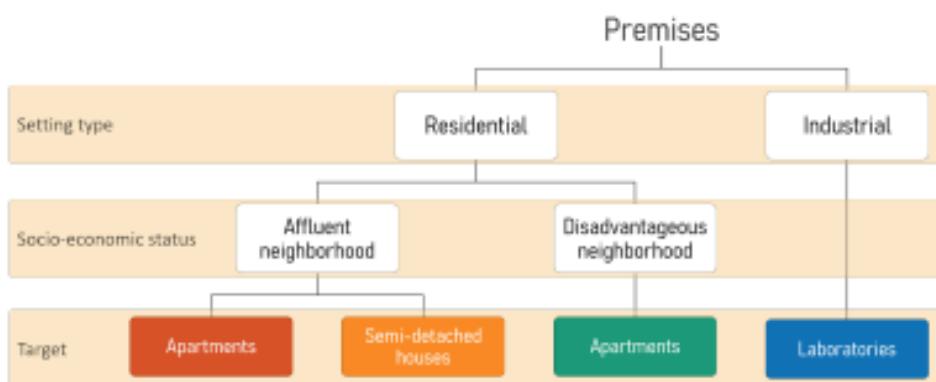


Figure 8: Locaux ciblés pour l'acquisition de données MORED

- Système d'acquisition

Les acquisitions de puissance ont été réalisées avec deux systèmes distincts : emonPi et IMPEC. emonPi a été utilisé pour l'acquisition de la consommation de WP et IL à faible taux d'échantillonnage, tandis qu'IMPEC a été utilisé pour l'acquisition de la signature de consommation IL à des taux d'échantillonnage élevés. IMPEC est un système intégré de surveillance et de traitement de la consommation d'électricité dans les bâtiments, équipé d'un processeur en temps réel et de modules d'E/S de haute résolution. De son côté, emonPi est un système de surveillance de l'énergie à code source ouvert basé sur Raspberry Pi, utilisant des transformateurs de courant et un adaptateur secteur pour les mesures de courant et de tension. Ces systèmes ont été utilisés avec des applications web et mobiles pour permettre aux résidents de surveiller leur consommation d'électricité en temps réel.

- Méthode d'acquisition

Comme le montre la figure 9, différents types de données ont été collectés, reflétant la consommation d'électricité dans les locaux marocains pendant la campagne d'acquisition de MORED :

Les données de consommation WP, les données de consommation de base d'IL, les données de consommation de signature d'IL et les données d'événement.

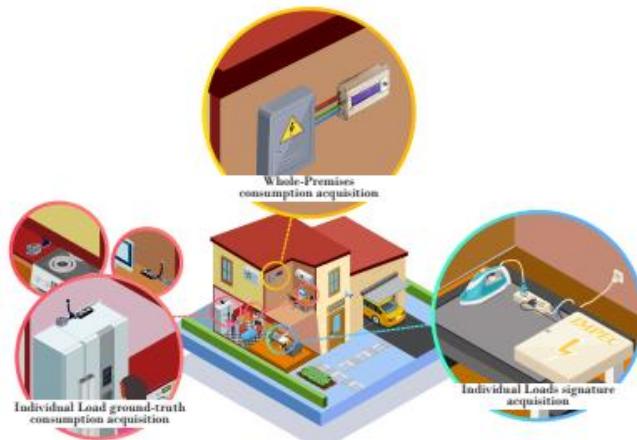


Figure 9:Les trois différents types de données de consommation d'électricité contenues dans MORED

5. Résumé de l'ensemble de données

MORED comprend trois types de données sur la consommation d'électricité : la consommation de WP étiqueté de terrain, consommation IL, consommation WP, et consommation de signature IL.

Cet ensemble de données, fournie par l'entreprise, est destiné à fournir davantage de données en général et d'aider à pallier la rareté des ensembles de données de consommation d'électricité EB sur le terrain en particulier.

Les principales caractéristiques de l'ensemble de données MORED sont les suivantes :

- Consommation d'électricité WP : Les données de consommation WP ont été acquises auprès d'un total de 13 ménages de différents statuts socio-économiques pour une période prolongée. Elles comprennent V et P extraites à 1/5[S/s] pour les locaux des quartiers aisés et à 1/10[S/s] pour les locaux des quartiers défavorisés.
- Consommation d'électricité étiquetée IL ground-truth : Les données de base de l'IL ont été acquises pendant une durée combinée de plus de 300 jours auprès de six locaux résidentiels

comprenant V et P extraits à 1/5 [S/s]. Les événements correspondant aux données IL de vérité au sol ne décrivent que les états opérationnels marche/arrêt des charges surveillées.

Les données de signature décrivent tous les états opérationnels des charges surveillées et sont rapportées dans un fichier CSV en suivant le processus décrit dans la section précédente.

- Les métadonnées : Elles offrent des informations détaillées pour les trois ensembles de données (un type d'appareil, des connaissances préalables sur le temps d'utilisation typique par jour, l'emplacement de la pièce dans les locaux, etc,) conformément au schéma de métadonnées de la NILM.) Elles sont présentées dans des fichiers texte YAML.
- Les données de mesure sont toutes fournies dans des fichiers CSV pour un accès facile et gratuit. Cependant, les données de signature IL sont également fournies dans des fichiers TDMS car elles ont été initialement enregistrées dans ce format.

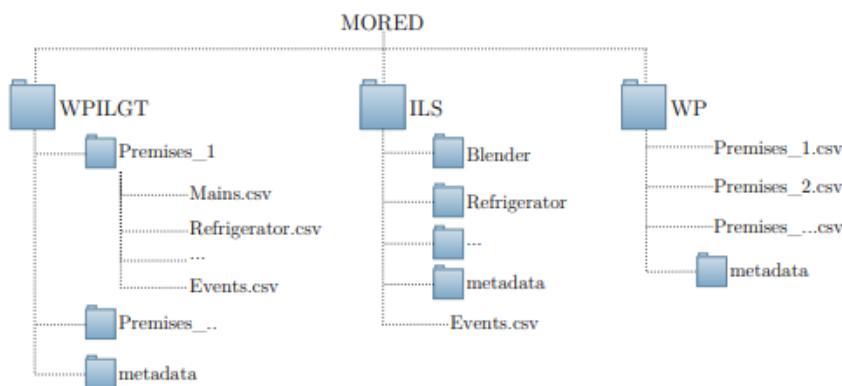


Figure 10: Arbre décrivant le contenu du répertoire MORED

6. Exploration des données

L'analyse de séries temporelles est un domaine de la statistique qui traite de l'analyse des données temporelles. Les données de séries temporelles sont des données collectées au fil du temps, et elles peuvent être utilisées pour suivre les tendances, identifier les motifs et faire des prédictions.

6.1. Concaténation de données

Pour mieux optimiser notre cadre de données, nous pouvons utiliser une représentation pratique de la définition mathématique d'une série temporelle (TS).

Une série temporelle (TS) peut être considérée comme une séquence ordonnée de points de données, où chaque point se compose d'un horodatage et d'une valeur. Les points sont classés par ordre chronologique.

Ainsi, une série temporelle TS peut être représentée comme suit :

$$TS = [(t_1, v_1), (t_2, v_2), \dots, (t_n, v_n)]$$

Chaque paire (t_i, v_i) dans la série temporelle représente une valeur enregistrée v_i à un moment précis, identifié par l'horodatage t_i . Il est important de noter que la série temporelle est limitée, ce qui signifie qu'elle contient un nombre fixe de n points de données.

Premises	debut_ts	real_power	Vrms	ts_len	n_occupants
0	WPILGT_1 2020-06-01 18:11:30	[392.0, 388.5, 390.0, 389.5, 392.5, 396.0, 395...	[224.96, 224.785, 224.945, 225.005, 225.485, 2...	192011	2
1	WPILGT_2 2020-09-17 21:14:50	[302.5, nan, nan, nan, nan, nan, nan, nan, na...	[229.23, nan, nan, nan, nan, nan, nan, nan, na...	506354	5
2	WPILGT_3 2020-06-27 20:41:00	[306.0, 308.0, 335.0, 337.0, 335.5, 330.5, 330...	[306.0, 308.0, 335.0, 337.0, 335.5, 330.5, 330...	680724	4
3	WPILGT_4 2020-07-14 23:00:00	[6.0, 5.0, 8.5, 7.5, 7.0, 5.5, 7.5, 5.5, 5.5, ...	[152.06, 153.08, 153.565, 153.785, 153.1, 153...	512999	3
4	WPILGT_5 2020-07-13 16:52:10	[154.5, 150.0, 148.5, nan, nan, nan, nan, na...	[227.215, 226.935, 226.505, nan, nan, nan, na...	382368	1
5	WPILGT_6 2020-09-27 22:56:20	[504.0, 510.0, 513.0, 518.5, 517.0, 509.0, 518...	[237.24, 237.245, 237.98, 237.895, 237.79, 237...	353725	3

number_of_elderly	number_of_adults	number_of_rooms	number_of_floors	area	total_acquisition_duration	mean_daily_consumption	total_number_of_monitored_loads
0	2	5	1	50	24	110	8
0	4	5	3	100	14	115	3
0	4	5	1	70	80	220	3
0	3	5	1	103	60	281	4
0	2	5	1	100	44	280	7
0	3	6	1	103	40	295	3

Figure 11: extrait de notre jeu de données

Une telle représentation permet de réduire considérablement la quantité de données, ce qui réduit l'empreinte mémoire. Cette pratique est indispensable lors du déploiement du projet à grande échelle. Pour une seule mesure, la réduction constatée est de 99%. Cela signifie que la taille des données est réduite à seulement 1% de sa taille initiale.

La réduction de la mémoire peut être réalisée en optimisant la façon dont les données sont stockées dans le DataFrame. Une méthode courante consiste à utiliser des types de données plus compacts pour les colonnes contenant des valeurs numériques. Par exemple, au lieu d'utiliser le type de données "float64" qui utilise 8 octets par valeur, nous pouvons utiliser des types de données plus petits comme "float32" ou "int32" qui utilisent seulement 4 octets par valeur.

De plus, la colonne de dates et d'heures peut occuper une quantité significative de mémoire. En utilisant des types de données plus adaptés pour représenter les dates et les heures, tels que le type "datetime64" de NumPy, nous pouvons réduire la consommation de mémoire.

Une autre approche pour réduire la consommation de mémoire est de segmenter le DataFrame ; autrement dit le représenter en une représentation compact faisant recours aux listes Python. Un segment est un 5-tuple représentant les points de données pour une période spécifique. Le 5-tuple se compose de l'heure de début, de l'heure de fin, de l'intervalle d'échantillonnage, d'une fonction de génération des timestamps pour une série temporelle donnée, et d'une constante d'échelle pour ajuster les valeurs des autres timestamps. La segmentation du DataFrame permet de traiter les données de manière indépendante, ce qui réduit la mémoire utilisée lors des opérations.

6.2. Nettoyage des données

Le nettoyage des données est le processus de correction ou de suppression des données corrompues, incorrectes ou inutiles d'un ensemble de données avant l'analyse des données.

6.2.1. Méthodologies utilisées pour traiter les valeurs manquantes (pour de longues périodes) :

L'objectif de cette section est d'évaluer différentes méthodes d'imputation des valeurs manquantes dans le contexte des séries temporelles de la consommation d'énergie des bâtiments sur de longues périodes. Pour cela, nous comparons plusieurs approches et déterminons celle qui est la plus appropriée. Nous explorons des méthodes telles que l'interpolation saisonnière, l'imputation basée sur le jour précédent, le profil horaire et le profil journalier et horaire pour combler les données manquantes. En analysant les résultats de chaque méthode, nous pouvons évaluer leur performance et déterminer celle qui offre les meilleurs résultats pour traiter les valeurs manquantes sur de longues périodes.

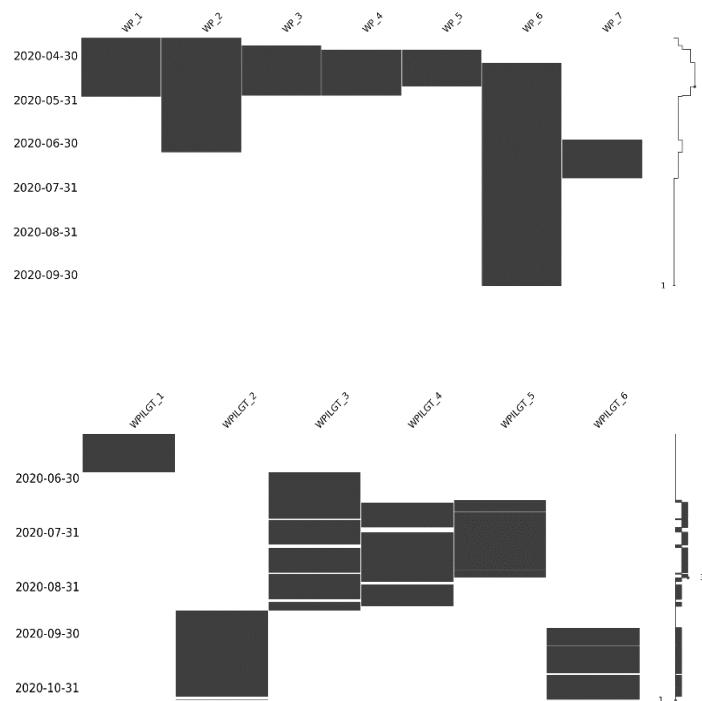


Figure 12:distribution des valeurs manquantes dans les Dataframes

Nous avons délibérément omis les données disponibles de la maison WPLGT_2 de l'ensemble de données WPLGT afin d'évaluer les différentes techniques et calculer l'erreur absolue moyenne (MAE) pour sélectionner la méthode appropriée à appliquer à tous les jeux de données. Nous avons également sélectionné différents intervalles de remplissage et utilisé le MAE comme critère pour choisir le modèle optimal. De cette manière, nous avons pu prendre en compte les défis liés au traitement de longues périodes de données manquantes et explorer les techniques appropriées pour minimiser leur impact sur l'analyse et la modélisation.

Une seule série temporelle de données de l'ensemble de données WPILGT (en particulier la maison WPILGT_2) est extraite et indexée par horodatage. Le modèle de données manquantes pour la période sélectionnée (du 11 octobre 2020 au 16 octobre 2012 : ["2020-10-11", "2020-10-16"]) et ensuite visualisé à l'aide de la fonction **msno.matrix**.

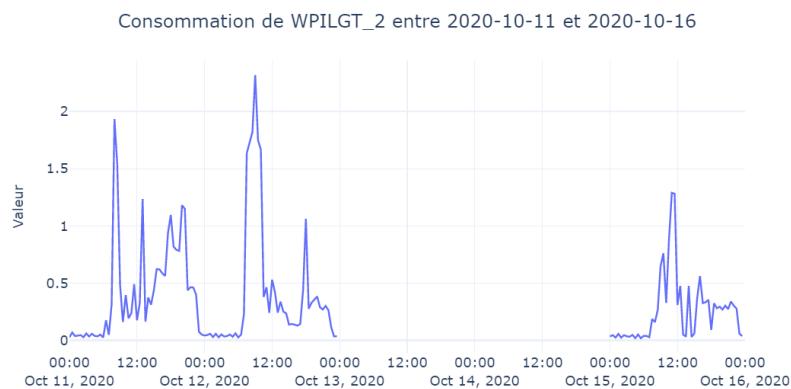


Figure 13: consommation de WPILGT_2 entre 2020-10-11 et 2020-10-16(série imputée de données)

• Imputation avec le jour précédent :

Pour compléter les données manquantes, une approche raisonnable consisterait à utiliser les relevés énergétiques de l'heure correspondante du jour précédent, en supposant que la consommation d'énergie suit un schéma répétitif :

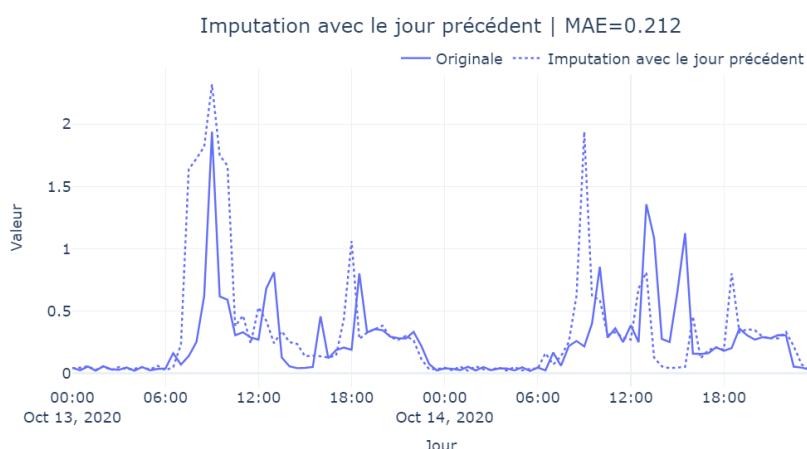


Figure 14:séries imputées complétées en utilisant un jour avant

- **Profil horaire moyen :**

Pour compléter les données manquantes, une meilleure approche consisterait à calculer un profil horaire à partir des données - la consommation moyenne pour chaque heure - et à utiliser la moyenne pour compléter les données manquantes. Cette approche est plus précise que l'utilisation des données de la veille, car elle tient compte de la variation de la consommation d'énergie dans le temps. Les résultats de cette évaluation sont présentés dans la Figure suivante :

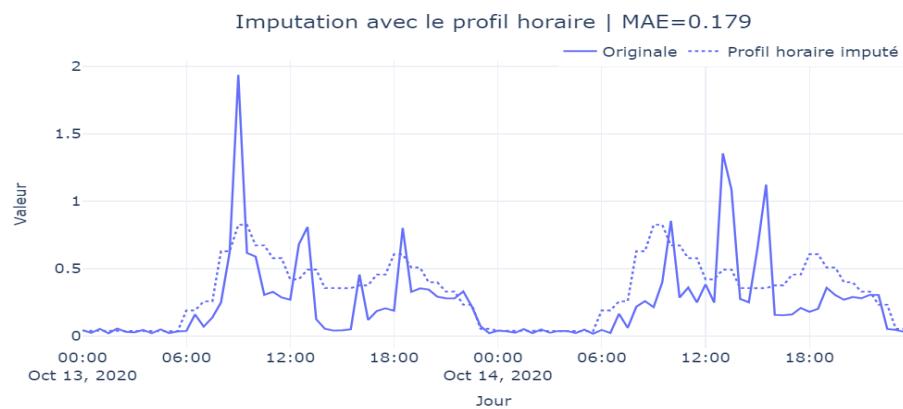


Figure 15:séries imputées complétées en utilisant la moyenne pour compléter les données manquantes

- **La moyenne horaire pour chaque jour de la semaine :**

Pour compléter les données manquantes, nous pouvons encore affiner l'approche en introduisant un profil spécifique pour chaque jour de la semaine. En effet, les habitudes de consommation d'énergie sont différentes en semaine et le week-end. Par exemple, la consommation d'énergie est généralement plus élevée en semaine que le week-end, car les gens sont au travail ou à l'école pendant la semaine et ont plus de temps libre le week-end.

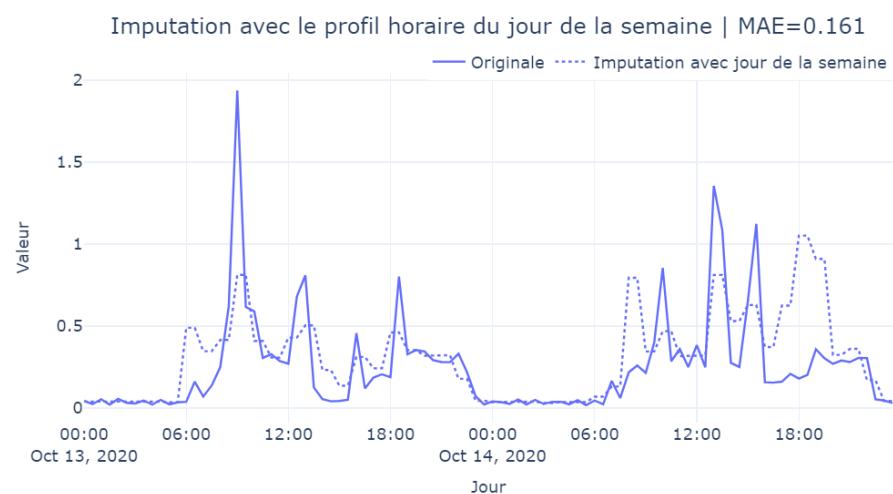


Figure 16:séries imputées complétées par la moyenne horaire pour chaque jour de la semaine

L'interpolation saisonnière est une méthode permettant de compléter les données manquantes dans les séries chronologiques en calculant d'abord un profil saisonnier, puis en le soustrayant des données originales, en appliquant une technique d'interpolation et en ajoutant à nouveau le profil saisonnier. Elle s'effectue selon les étapes suivantes :

- Le calcul du profil saisonnier. Pour ce faire, la moyenne de tous les relevés horaires pour chaque heure de la journée est calculée, pour chaque jour de la semaine.
- Soustraire le profil saisonnier des données originales. On obtient ainsi une série temporelle qui ne contient que les informations relatives à la tendance.
- Puis l'application de l'une des techniques d'interpolation. Il peut s'agir d'une interpolation linéaire.

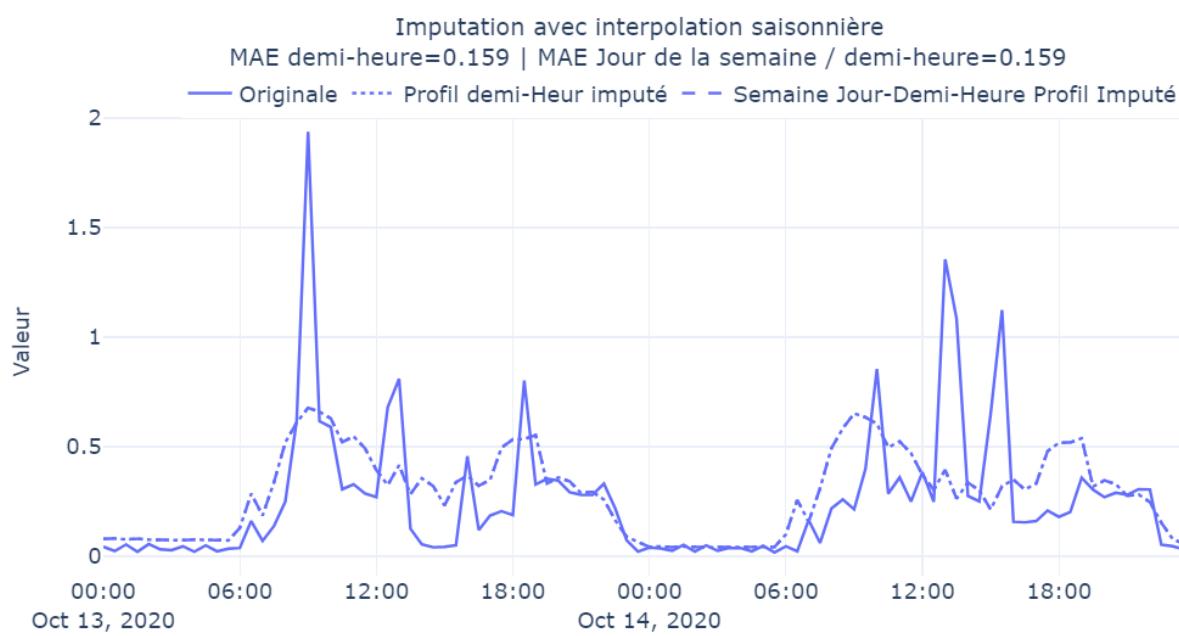


Figure 17: Visualisation des comparaisons de méthodes

Résultat imputation :

Le tableau compare l'erreur absolue moyenne (EAM) pour différentes méthodes d'imputation des valeurs manquantes. La MAE représente l'écart moyen entre les valeurs prédites et les valeurs réelles. Parmi les méthodes évaluées, l'interpolation saisonnière présente la plus faible erreur absolue moyenne (0.1518775), ce qui en fait le choix optimal pour combler les valeurs manquantes. Les autres méthodes, telles que le jour précédent (0.212177), le profil horaire (0.179372) et le profil journalier et horaire

(0.161125), affichent des erreurs légèrement plus élevées.

	Imputation	MAE
0	Jour précédent	0.212177
1	Profil horaire	0.179372
2	Profil journalier et horaire	0.161125
3	Interpolation saisonnière Jour de semaine-demi...	0.158775
4	Interpolation saisonnière Demi-heure	0.158775

Figure 18: Erreur absolue moyenne (EAM) pour différentes méthodes d'imputation

En conclusion, en évaluant l'erreur absolue moyenne (EAM) pour différentes méthodes d'imputation des valeurs manquantes, nous constatons que l'interpolation saisonnière présente la plus faible erreur absolue moyenne parmi les méthodes examinées. Cela suggère que l'interpolation saisonnière peut être une approche prometteuse pour combler les valeurs manquantes. Cependant, les autres méthodes telles que le jour précédent, le profil horaire et le profil journalier et horaire obtiennent des résultats légèrement moins performants. Ainsi, il convient de prendre en considération les spécificités de chaque méthode et les objectifs de l'analyse avant de sélectionner la méthode d'imputation appropriée.

6.2.2. Suppression des doublons

La suppression des doublons signifie la suppression de toutes les valeurs en double. Il n'y a pas besoin de valeurs en double dans l'analyse des données. Ces valeurs n'affectent que la précision et l'efficacité du résultat de l'analyse, nous utilisons donc la fonction méthode **drop_duplicates()** pour supprimer les colonnes ou les lignes en double.

6.3 Standardisation des données

Les données réalistes ont des caractéristiques qui varient en taille, en unités et en plage. La mise à l'échelle des fonctionnalités aide essentiellement à normaliser les données dans une certaine plage. Nous devons donc importer **StandardScalar()** de la bibliothèque sklearn et l'appliquer à notre ensemble de données. Aussi :

- **MinMaxScaler** : Cet échelonneur met à l'échelle chaque caractéristique dans un intervalle de 0 à 1. Pour ce faire, il soustrait la valeur minimale de chaque caractéristique, puis la divise par l'intervalle (valeur maximale - valeur minimale).
- **RobustScaler()** : Ce scaler est moins sensible aux valeurs aberrantes que le StandardScaler ou le MinMaxScaler. Pour ce faire, il utilise la médiane et l'écart absolu médian (MAD) au lieu de la moyenne et de l'écart type.

La transformation employée sur la transformation cible utilise une variété de techniques, y compris la désaisonnalisation, la différenciation et les transformations monotones, pour rendre une série plus stationnaire. Pour ce faire la classe Python **Auto_Transformer** a été implémentée pour mettre en œuvre cette approche. Elle a été définie et implémentée sur python et ajoutée au livrable des codes et peut être utilisée pour effectuer automatiquement les transformations nécessaires pour une série temporelle donnée.

- **Confidence** : Le niveau de confiance pour les tests statistiques. Il s'agit de la probabilité que le test identifie correctement une tendance ou une saisonnalité par défaut 0.05 .
- **seasonal_period** : Le nombre de périodes après lesquelles le cycle de saisonnalité se répète. Si cette valeur n'est pas fournie, la classe essaiera d'inférer la période de saisonnalité à partir des données.
- **seasonality_max_lags** : Le nombre maximum de retards à prendre en compte lors de la recherche de la saisonnalité.
- **trend_check_params** : Les paramètres utilisés pour les tests statistiques de tendance.
- **detrender_params** : Les paramètres transmis à la classe mère choisie.
- **deseasonalizer_params** : Les paramètres transmis à la classe mère choisie.
- **box_cox_params** : Les paramètres transmis à d'une classe qui performe la transformation Box Cox.

La classe possède les méthodes suivantes :

- **fit(self, X: pd.DataFrame)** : Cette méthode ajuste le transformateur aux données.
- **transform (self, X: pd.DataFrame)** : Cette méthode transforme les données.

6.4 Sélection des fonctionnalités (Feature Selection)

6.4.1 Procédure Sélection des fonctionnalités :

L'ingénierie des caractéristiques est le processus de transformation des données brutes en caractéristiques plus utiles pour les algorithmes d'apprentissage automatique. Il peut s'agir de sélectionner des caractéristiques, de les transformer ou d'en créer de nouvelles. L'objectif de l'ingénierie des caractéristiques est d'améliorer les performances des algorithmes d'apprentissage automatique en rendant les données plus faciles à comprendre pour les algorithmes.

Les quatre catégories de caractéristiques temporelles dans l'analyse des séries temporelles, ainsi que la manière dont elles peuvent être mises en œuvre en Python [16] :

- Caractéristiques de date et d'heure : Il peut s'agir d'extraire l'année, le mois, le jour, l'heure ou le jour de la semaine.

- Caractéristiques de décalage et caractéristiques de fenêtre : Les caractéristiques de fenêtre, quant à elles, prennent en compte une fenêtre de valeurs antérieures plutôt qu'un seul décalage. Ces caractéristiques capturent les dépendances temporelles et les modèles dans les données.
- Statistiques à fenêtre mobile : Les statistiques à fenêtre glissante impliquent le calcul de statistiques agrégées à l'intérieur d'une fenêtre mobile de la série temporelle. Les statistiques courantes à fenêtre glissante comprennent la moyenne glissante, l'écart-type glissant et la somme glissante.
- Statistiques de la fenêtre en expansion : Les statistiques à fenêtres élargies calculent des statistiques agrégées depuis le début de la série temporelle jusqu'au point actuel. Ces caractéristiques donnent une idée de la tendance générale et de la distribution des données.

6.4.2 Procédure Evaluation des fonctionnalités

- Schéma de l'algorithme d'importance par permutation [17]

Entrées : modèle prédictif ajusté m, ensemble de données tabulaires D.

- Calculer le score de référence s du modèle m sur les données D (par exemple la précision d'un classificateur ou le MSE ou R^2 d'un modèle de régression)
- Pour chaque caractéristique j (colonne de D) :
 - Pour chaque répétition k dans 1, ..., K :
 - *Effectuer un battage aléatoire de la colonne j de l'ensemble de données D pour générer une version corrompue des données appelée $\tilde{D}_{k,j}$.
 - *Calculer le score $s_{k,j}$ du modèle m sur les données corrompues $\tilde{D}_{k,j}$.
 - Calculer l'importance i_j pour la caractéristique f_j définie comme suit :

$$i_j = s - \frac{1}{K} \sum_{k=1}^K s_{k,j}$$

6.5 Partitionnement des données

Les données de la série chronologique seront utilisées à des fins d'entraînement, de validation et de test, afin de garantir une analyse complète. L'ensemble de l'entraînement comprend des séries temporelles complètes, chaque une couvrant une durée d'environ 2 à 3 mois, ce qui permet une compréhension approfondie des modèles et des tendances. Cette période d'entraînement prolongée permet

au modèle d'apprendre à partir d'un large éventail de dynamiques temporelles. Ensuite, un seul jour est désigné pour l'ensemble de validation, ce qui nous permet d'évaluer les performances du modèle et d'affiner ses paramètres. Enfin, le modèle sera testé la semaine suivante, en simulant des scénarios de prévision du monde réel. Cette approche permet une évaluation rigoureuse des capacités prédictives du modèle, fournissant des indications précieuses sur sa capacité de généralisation et son adéquation aux applications pratiques. En divisant systématiquement les séries temporelles en sous-ensembles distincts d'entraînement, de validation et de test, nous établissons un cadre robuste pour évaluer et valider les performances du modèle dans diverses conditions temporelles.

6.6 Analyse et visualisation des données de séries temporelles

Lors de la phase d'analyse exploratoire des données (AED) de la consommation électrique d'un foyer, un tracé linéaire de la série temporelle est souvent utilisé pour visualiser les tendances et les modèles au fil du temps. Le rééchantillonnage des données, par exemple l'agrégation des mesures horaires ou quotidiennes sur des intervalles de temps plus importants, fournit une représentation plus précise des schémas de consommation globaux, ce qui nous permet d'obtenir des informations significatives sur les tendances de consommation à long terme.

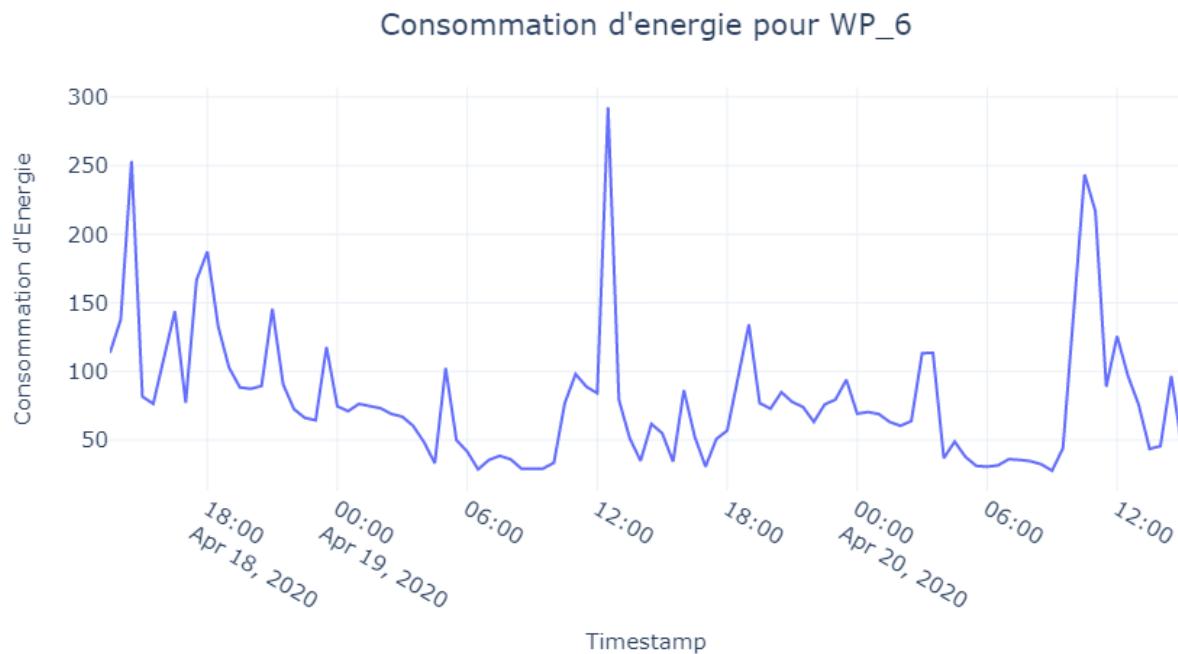


Figure 19: Consommation d'énergie pour WP6

Pour analyser les profils de consommation d'énergie d'un jour à l'autre, plusieurs graphiques linéaires représentant différents jours peuvent être superposés sur le même graphique.

Le graphique présenté dans la figure 20 permet de comparer les schémas de consommation sur plusieurs jours, ce qui permet d'identifier les tendances et les anomalies récurrentes tout au long du mois. Les différentes courbes montrent les profils mensuels, permettant d'observer le profil général des journées avec une résolution horaire.

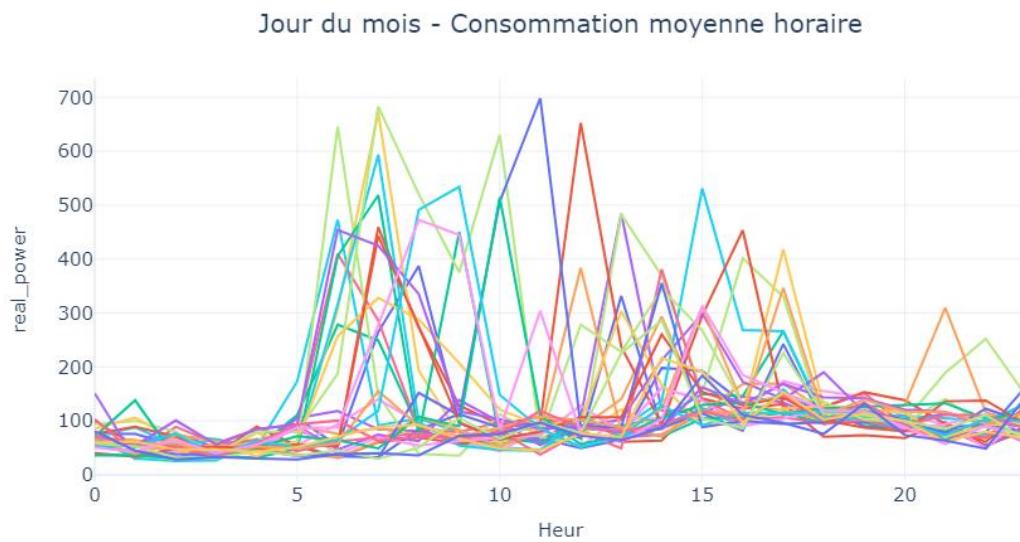


Figure 20: Consommation moyenne horaire

7. Conclusion

Dans ce chapitre, nous avons présenté la méthode de collecte des données et avons effectué les étapes préliminaires de préparation de notre ensemble de données d'entraînement. Cela comprenait des processus tels que l'ingénierie des caractéristiques, la mise à l'échelle des données et la transformation de la cible. Ces étapes ont été essentielles pour préparer nos données en vue de la construction ultérieure de modèles de prédiction basés sur les données collectées.



Chapitre 4 : Modèles Machine Learning pour la prédition de la consommation et l'analyse des résultats obtenus

1. Introduction

Avant de construire un modèle, il est nécessaire de bien déterminer les différentes méthodes qu'on va utiliser pour mesurer la performance relative de chacun des modèles. Au cours de ce chapitre, nous avons examiné en détail ces méthodes et rapporté les performances des modèles appliqués à un local résidentiel spécifique. En utilisant ces évaluations, nous procéderons à la sélection finale du modèle qui démontre les meilleures performances.

2. Application des modèles Machine Learning

2.1 Entraînement du modèle

L'approche utilisée pour former et tester un modèle dans l'analyse des séries temporelles consiste à diviser l'ensemble de données en ensembles d'entraînement, de validation et de test, en tenant compte de l'aspect temporel des données. Cela permet de s'assurer que le modèle est formé sur des données passées, validé sur des données proches de l'avenir et testé sur des données futures afin d'évaluer sa capacité de généralisation.

L'ensemble de données est divisé sur la base de critères de date spécifiques. La colonne de l'horodatage est utilisée pour filtrer les données par année, mois et jour. (Voir annexe A et B)

En appliquant des filtres de date, nous pouvons nous assurer que le modèle est formé sur des données historiques, validé sur le passé immédiat et testé sur des données futures. Cette approche permet une évaluation complète des performances du modèle et de sa capacité à se généraliser à des périodes de temps inédites. Il est important d'ajuster les plages de dates et les conditions en fonction des exigences spécifiques et des caractéristiques des données de la série temporelle.

2.2. Métriques d'évaluation

Les métriques retournées pour les modèles de régression sont conçues pour estimer la quantité d'erreurs d'un modèle. Si la différence entre valeurs observées et prévues est faible, le modèle juge correctement les données. Toutefois l'observation du modèle des résidus (la différence entre n'importe quel point prédit et sa valeur réelle correspondante) peut nous en apprendre beaucoup sur un éventuel décalage dans le modèle.

Les métriques suivantes sont utilisées pour évaluer notre modèle de régression [17] [18] :

- **L'erreur absolue moyenne (MAE)** mesure la précision des prédictions par rapport aux résultats réels ; un score inférieur est donc préférable. Il s'agit de la moyenne de l'erreur non signée entre la prévision à l'étape (f_t) et l'observation (y_t). La formule est la suivante :

$$MAE = \frac{1}{N \times L} \sum_i^N \sum_j^L |f_{i,j} - y_{i,j}|$$

- **L'Erreur quadratique moyenne (MSE)** crée une valeur unique qui résume l'erreur dans le modèle. En mettant la différence au carré, la métrique ne tient pas compte de la différence entre sur-prédition et sous-prédition.

$$MSE = \frac{1}{N \times L} \sum_i^N \sum_j^L (f_{i,j} - y_{i,j})^2$$

- **L'erreur absolue moyenne mise à l'échelle (MASE)** est une mesure légèrement plus complexe mais plus pertinente que le MSE ou le MAE, car il surmonte la dépendance à l'échelle des deux mesures précédentes en mettant à l'échelle les erreurs en fonction du MAE de la méthode de prévision naïve.

$$MASE = \frac{\frac{1}{L} \times \sum_i^L |f_i - y_i|}{\frac{1}{L-1} \times \sum_{i=2}^L |y_i - y_{i-1}|}$$

- **Biais de prévision (FB)** : Le biais de prévision est une mesure qui nous aide à comprendre si la prévision est continuellement surestimée ou sous-estimée. Le biais de prévision correspond à la différence entre la somme des prévisions et la somme des valeurs observées, exprimée en pourcentage de la somme de toutes les valeurs réelles.

$$FB = \frac{\sum_i^N \sum_j^L f_{i,j} - \sum_i^N \sum_j^L y_{i,j}}{\sum_i^N \sum_j^L y_{i,j}}$$

2.3. Résultats des modèles

2.3.1 Résultats du modèle de Régression Linéaire

La régression linéaire est une technique de modélisation qui ajuste une fonction linéaire aux données. Elle suppose une relation linéaire entre les variables indépendantes et la variable dépendante. Les paramètres du modèle, représentés par les coefficients β , sont estimés à l'aide de méthodes telles que la méthode moindres carrés ordinaires (MCO). La MCO minimisent l'erreur quadratique moyenne (EQM) en trouvant les valeurs de β qui minimisent la somme des différences au carré entre les valeurs prédites et réelles. La régression linéaire repose sur cinq hypothèses, notamment la linéarité, la distribution

normale des erreurs, la variance constante des erreurs, l'absence d'autocorrélation dans les erreurs et peu ou pas de corrélation entre les variables indépendantes (multicollinéarité). Cependant, pour des fins de prédiction, l'hypothèse de linéarité est la plus importante. Si les variables ne sont pas linéairement liées, le modèle peut avoir des performances médiocres, mais projeter les entrées dans un espace de dimension supérieure peut partiellement résoudre ce problème. La multicollinéarité se produit lorsque les variables indépendantes sont fortement corrélées, ce qui rend l'estimation des coefficients instable. Dans l'analyse de séries chronologiques, telle que la prévision, la multicollinéarité peut être présente en raison de la corrélation entre les variables retardées et mobiles.

Pour évaluer la régression linéaire, un exemple de ménage résidentiels est utilisé à partir d'un ensemble de données de validation, et le modèle est entraîné et testé à l'aide de la classe LinearRegression de scikit-learn. Les coefficients résultants (β) indiquent l'influence de chaque caractéristique sur la sortie. L'importance des caractéristiques peut être visualisée (Voir figure 22), révélant les corrélations entre les caractéristiques et le potentiel de multicollinéarité. Pour améliorer le modèle linéaire, il est possible de supprimer les caractéristiques colinéaires et d'effectuer des techniques de sélection de caractéristiques telles que la sélection avant ou l'élimination arrière. Alternativement, des modifications peuvent être apportées au modèle linéaire pour renforcer sa robustesse contre la multicollinéarité et la sélection de caractéristiques.

Le modèle est d'abord entraîné sur un sous-ensemble des données, puis ses prédictions sont comparées aux valeurs réelles sur un ensemble de test séparé. Les étapes suivantes sont effectuées :

Voici les définitions des paramètres utilisés dans le code : `

- `feat_config`` est un objet de configuration qui spécifie les caractéristiques à utiliser dans le modèle.
- `missing_value_config`` est un objet de configuration qui spécifie comment gérer les valeurs manquantes.
- `model_config`` est un objet de configuration qui spécifie le type de modèle à utiliser et ses hyperparamètres.
- Une Dataframe sous forme d'un tableau NumPy contenant les prédictions du modèle pour l'ensemble de test.
- `metrics`` est un dictionnaire contenant les métriques de performance du modèle.
- La prédiction sera affectée à une DataFrame Pandas contenant les prédictions du modèle et les valeurs réelles pour l'ensemble de test.

L'ensemble des caractéristiques utilisée et configuration sont présentées dans l'annexe E :

Un objet figure de la librairie Plotly Express est instancié par un tracé des prédictions du modèle :

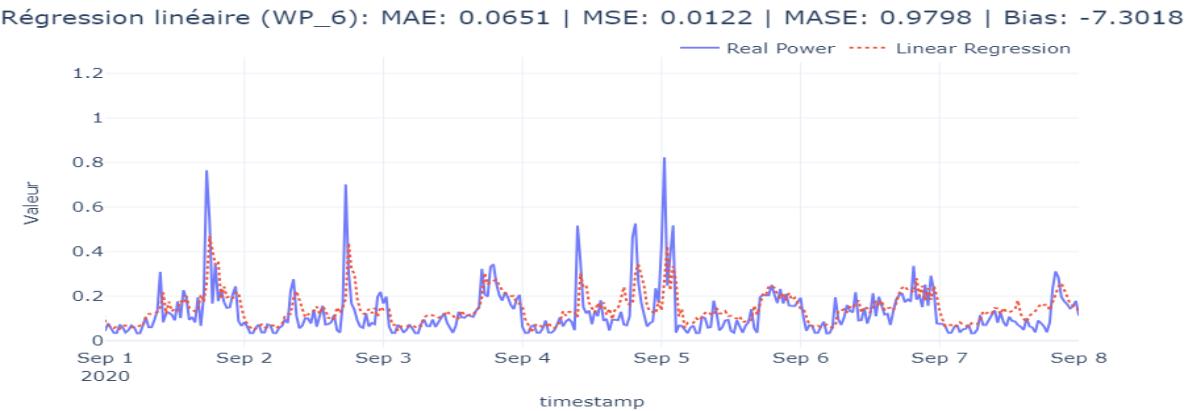


Figure 21:Le tracé des prédictions du modèle « Modèle de régression linéaire (WP_6)»

Les résultats de l'évaluation démontrent que le modèle de régression linéaire est capable de fournir des prévisions raisonnablement précises de la consommation d'énergie. La MAE (erreur absolue moyenne) de 0,0651, la MSE (erreur quadratique moyenne) de 0,0122 et la MASE (erreur absolue moyenne mise à l'échelle) de 0,09798 sont toutes relativement faibles, ce qui indique une bonne performance du modèle.

Cependant, il est important de noter que le biais de prévision a une valeur de -7,3018. Cela indique une tendance générale à sous-estimer les observations réelles, avec un écart moyen de -7,3018. Il est essentiel de tenir compte de ce biais lors de l'interprétation des prévisions du modèle.

Dans l'ensemble, ces résultats suggèrent que le modèle de régression linéaire est bien adapté à la prédiction de la consommation d'énergie. Toutefois, il est essentiel de contrôler et d'évaluer régulièrement les performances du modèle pour garantir sa fiabilité et son utilité dans un contexte opérationnel. En outre, le fait que les erreurs de validation soient proches de celles de l'ensemble de test renforce la confiance dans la capacité du modèle à généraliser les prédictions au-delà des données d'apprentissage.

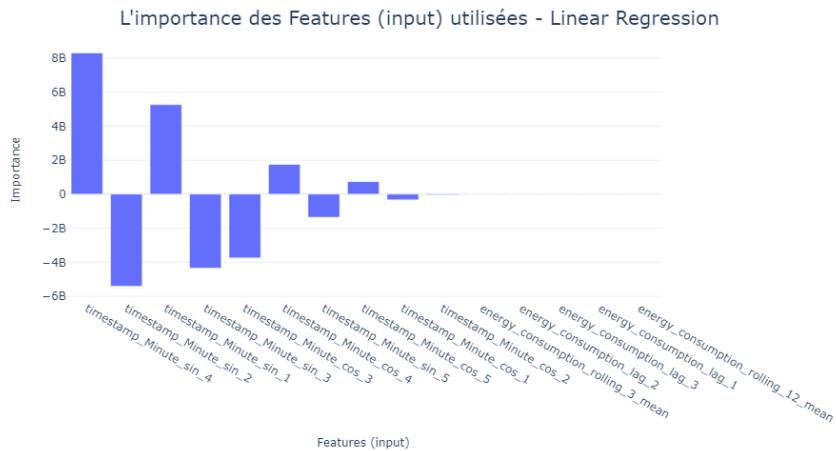


Figure 22:L'importance des caractéristiques à utiliser dans le modèle

Pour évaluer les caractéristiques sur la base du graphique, on peut examiner les relations entre les caractéristiques et la variable cible. En analysant les modèles, les tendances et les corrélations affichés dans le graphique, on peut se faire une idée de la pertinence et du pouvoir prédictif des caractéristiques. En outre, l'identification de variations significatives ou d'anomalies dans le graphique peut aider à prioriser l'évaluation de caractéristiques spécifiques en vue d'une investigation plus approfondie.

- Résultats du modèle de Régression Linéaire Ridge :

La régression linéaire régularisée introduit une contrainte sur le processus d'apprentissage afin de réduire la complexité de la fonction apprise. En régression linéaire, des coefficients de grande magnitude peuvent conduire à un modèle plus complexe et sensible aux petites variations des caractéristiques d'entrée. Pour remédier à cela, des techniques de régularisation telles que la décroissance pondérée sont appliquées. La décroissance pondérée (**Weight Decay**) ajoute un terme à la fonction de perte qui pénalise la magnitude des coefficients. La force de régularisation est contrôlée par un paramètre, λ . Les deux normes de régularisation les plus couramment utilisées sont la norme L1 (régression Lasso) et la norme L2 (régression Ridge).

La figure suivante illustre un graphique des contours de la fonction d'erreur non régularisée (en bleu) ainsi que de la région de contrainte pour le régulariseur quadratique $q = 2$ à gauche et le régulariseur lasso $q = 1$ à droite, où la valeur optimale du vecteur de paramètres w est représentée par w^* . Le lasso donne une solution parcimonieuse dans laquelle $w_1^* = 0$.

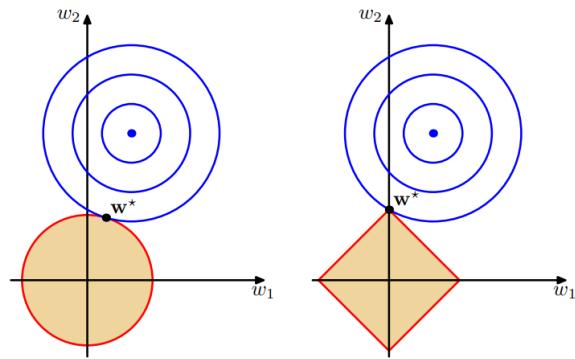


Figure 23 : Graphique des contours de la fonction d'erreur non régularisée [19]

Les définitions des paramètres utilisés dans le code sont les mêmes que ceux implémenté dans la régression linéaire simple en instanciant un objet de la classe **RidgeCV** du model sujet

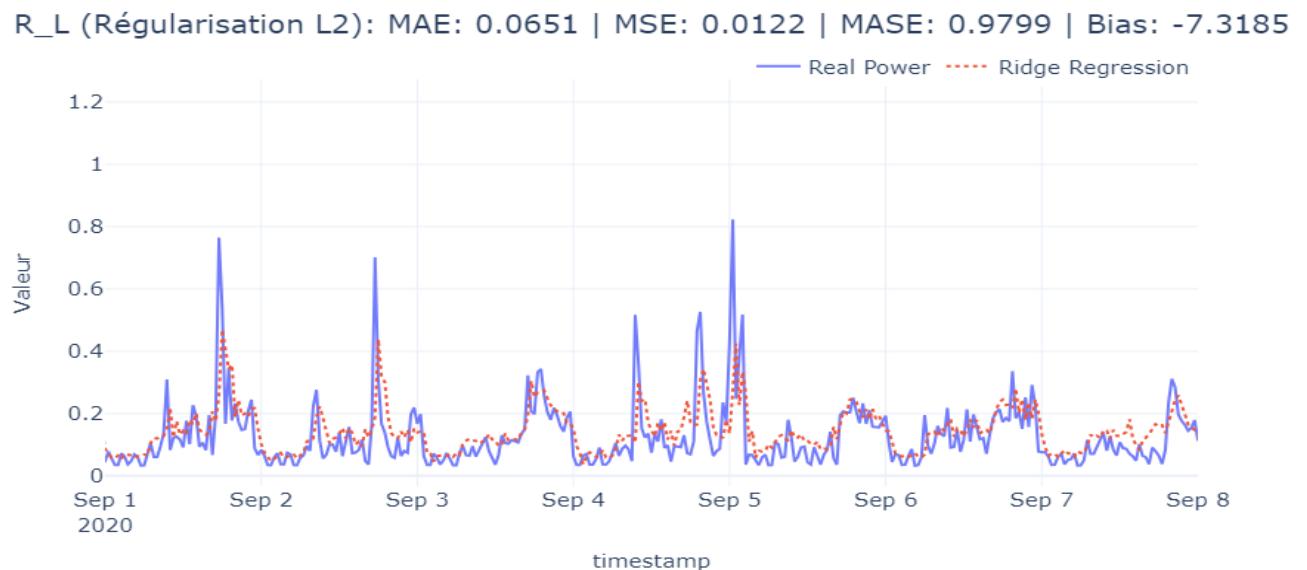


Figure 24: Résultats régression linéaire

La régression linéaire et la régression linéaire régularisée par Ridge présentent des erreurs similaires avec une légère différence dans le biais de prévision.

L'importance des Features(input) utilisées - Régression ridge (Régularisation L2)

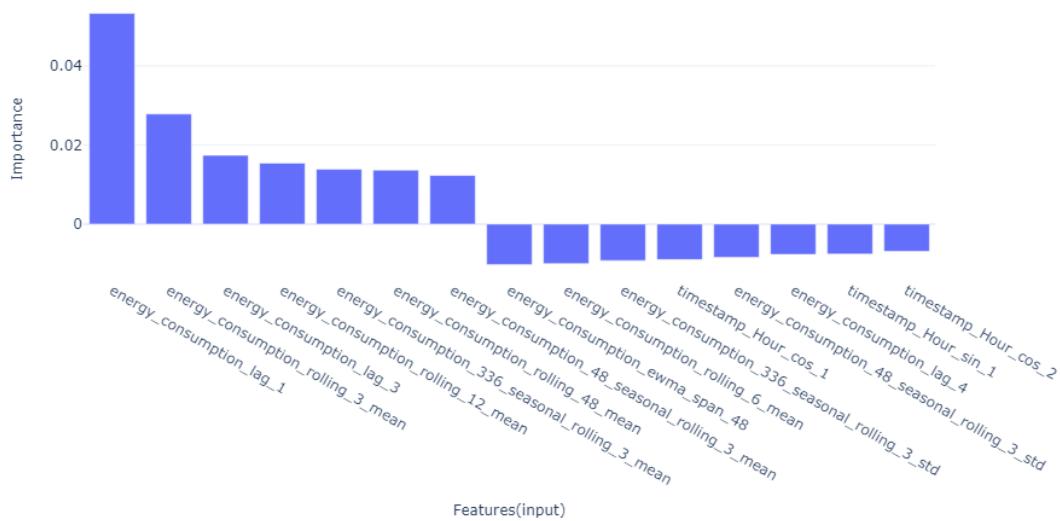


Figure 25:L 'importance des caractéristiques à utiliser dans le modèle

R_L (Régularisation L1)': MAE: 0.0624 | MSE: 0.0121 | MASE: 0.9386 | Bias: -2.1478

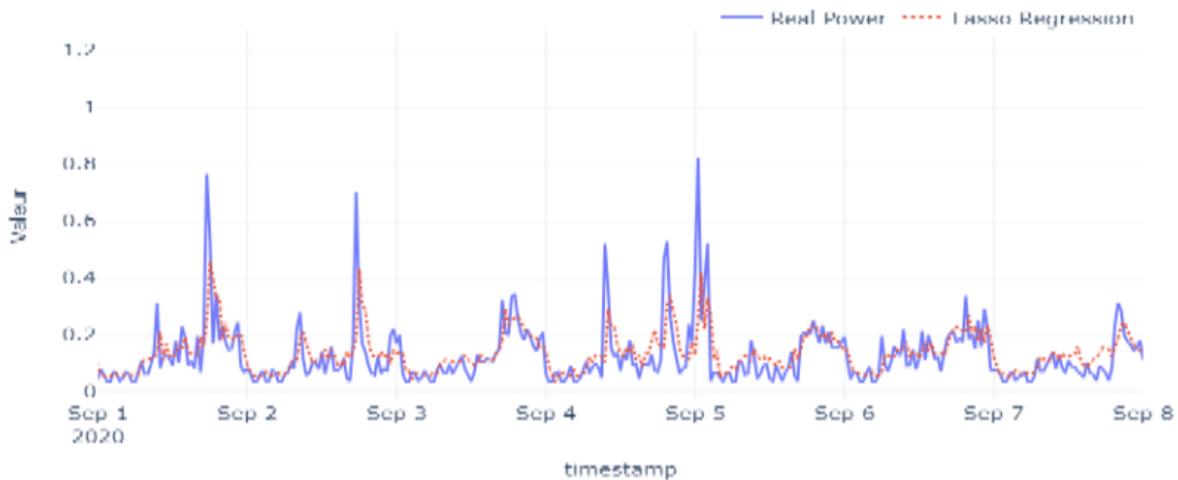


Figure 26:Résultats du modèle Régression Linéaire Lasso

Les résultats de l'évaluation indiquent que le modèle de régression Lasso est capable de faire des prédictions raisonnablement précises de la consommation d'énergie très proches des deux variantes simples et Ridge. Le MAE=0.0624, le MSE=0.0121 et le MASE=0.9386 du modèle sont tous relativement faibles, et le biais de prévision de -2.1478, donc on constate que la régularisation en norme L1 a pu globalement rapprocher le modèle des valeur réelles par rapport au model Ridge qui utilise une

régularisation avec la norme L2, aussi le plus important est l'import d'une diminution relative par rapport au biais de la prévision.

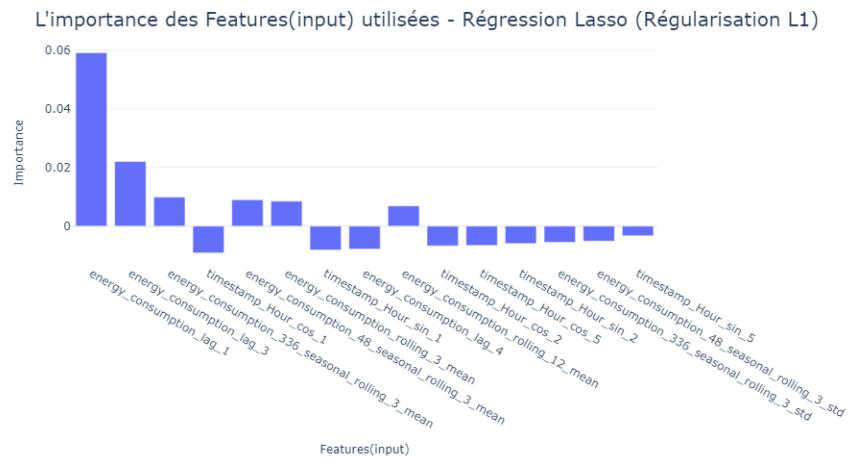


Figure 27:L'importance des caractéristiques à utiliser dans le modèle

2.3.2 Résultats du modèle Arbre de décision

Un arbre de décision est un algorithme d'apprentissage supervisé qui peut être utilisé pour des tâches de classification et de régression. Il s'agit d'un modèle arborescent qui effectue des prédictions en divisant les données en sous-ensembles de plus en plus petits sur la base d'un ensemble de règles. Les règles sont créées en partitionnant les données de manière récursive, en commençant par l'ensemble des données et en les divisant en deux sous-ensembles sur la base de la valeur d'une caractéristique unique. Ce processus est répété jusqu'à ce que chaque sous-ensemble ne contienne que des points de données très similaires les uns aux autres.

Les paramètres utilisés dans le code sont les mêmes, le model est instancier en une variable en utilisant la classe DecisionTreeRegressor du module **sklearn.tree** de **sklearn**.

L'algorithme de l'arbre de décision est un algorithme **greedy**, ce qui signifie qu'il prend la meilleure décision à chaque étape sans tenir compte de l'impact de cette décision sur les étapes suivantes. Cela peut conduire à un surajustement, c'est-à-dire à un problème où le modèle apprend trop bien les données d'apprentissage et n'est pas en mesure de s'adapter à de nouvelles données

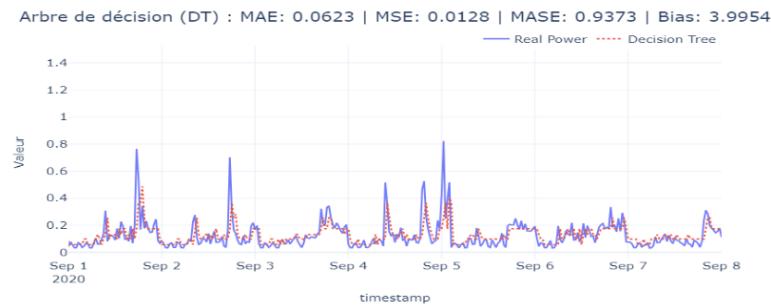


Figure 28:Arbre décision

Le MAE = 0.0623, le MSE = 0.0128 et le MASE = 0.9373 du modèle sont tous relativement faibles, et le biais de 3.9954.

Ces caractéristiques sont susceptibles d'avoir le plus grand impact sur la prédiction

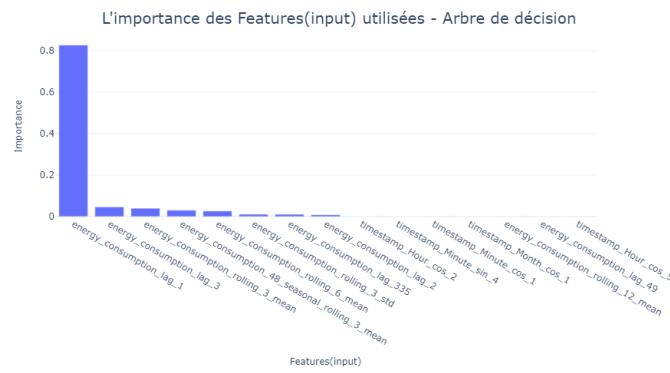


Figure 29:L'importance des features

2.3.3 Résultats du modèle de régression par forêt aléatoire (Random Forest)

Le model forêt aléatoire propose un algorithme d'apprentissage en ensemble qui combine les prédictions de plusieurs modèles individuels, appelés apprenants de base ou arbres de décision, pour former un modèle plus robuste et précis. Il surmonte les limites d'un seul modèle en exploitant le principe de la sagesse de la foule (wisdom of crowds) :

Dans l'algorithme de la forêt aléatoire, nous décidons du nombre d'arbres à construire, noté M. Pour chaque arbre, les étapes suivantes sont répétées :

- Tirer un échantillon Bootstrap à partir de l'ensemble de données d'entraînement.
- Sélectionner un sous-ensemble aléatoire de caractéristiques.
- Trouver la meilleure division en utilisant le sous-ensemble de caractéristiques sélectionné et créer deux nœuds enfants.
- Répéter les 2 dernières étapes jusqu'à atteindre un critère d'arrêt.

La différence principale par rapport aux arbres de décision classiques réside dans l'échantillonnage aléatoire des caractéristiques à chaque division, ce qui introduit plus d'aleatoire et réduit la corrélation entre les sorties des arbres.

La fonction de prédiction pour la forêt aléatoire en régression est :

$$y_{pred} = \frac{1}{M} \sum y_t$$

Où y_{pred} est la valeur prédictive finale, y_t est la sortie du t -ième arbre dans la forêt aléatoire, et M est le nombre total d'arbres.

Les définitions des paramètres utilisés dans le code restent les mêmes :

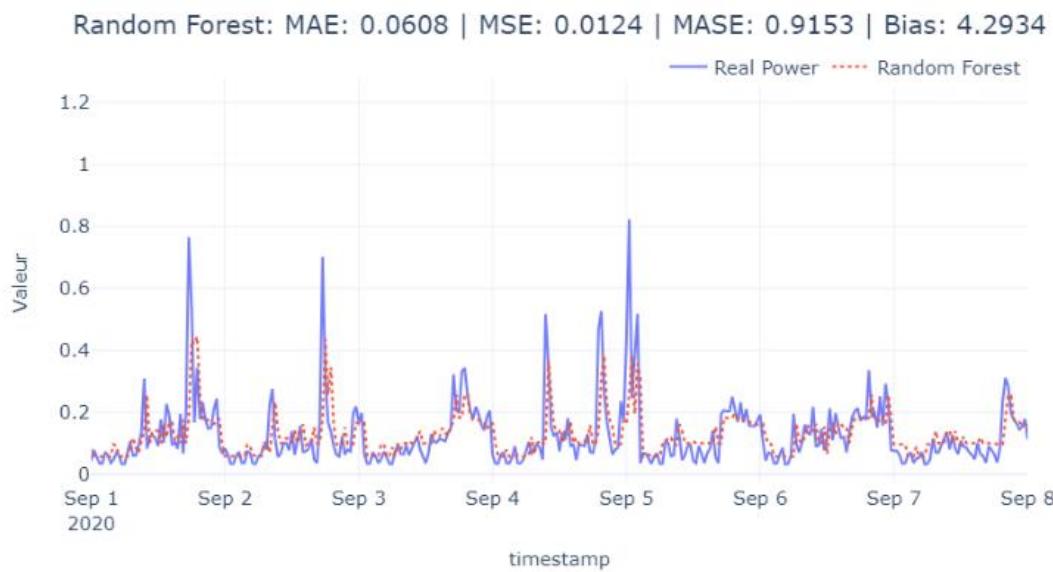


Figure 30:Random Forest

Les résultats de l'évaluation indiquent que le modèle de régression par forêt aléatoire est capable de faire des prédictions précises de la consommation d'énergie. Le MAE= 0.0608, le MSE=0.9153 et le MASE=0.9153 du modèle sont tous relativement faibles, et le biais de prévision est 0.29. Elle est meilleure que l'arbre de décision, elle légèrement proche des modèles linéaires. Cependant, il faut garder à l'esprit que nous n'avons pas ajusté le modèle et que nous pourrions obtenir de meilleurs résultats en réglant les hyperparamètres appropriés.

Tout comme l'importance des caractéristiques dans les arbres de décision, les forêts aléatoires ont également un mécanisme similaire pour estimer l'importance des caractéristiques. Étant donné que nous avons de nombreux arbres dans la forêt aléatoire, nous accumulons la diminution du critère de division à

travers tous les arbres de la forêt et obtenons une seule caractéristique d'importance dans ce cas la caractéristique Lag_1 pour la forêt aléatoire. Cela peut être consulté dans les modèles de scikit-learn en utilisant l'attribut feature_importance du modèle entraîné.

Ces caractéristiques ont probablement le plus grand impact sur les prédictions du modèle.

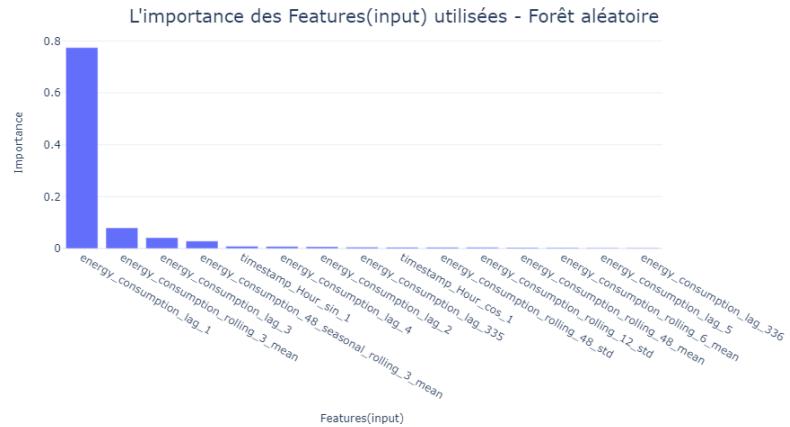


Figure 31:L'importance des features

2.3.4 Résultats des modèles XGBoost et LightGBM :

Le gradient boosting est une méthode d'ensemble qui combine des apprenants faibles de manière séquentielle pour créer un modèle puissant. Il diffère du bagging en utilisant une approche pas à pas, où chaque apprenant faible est appliqué à des versions modifiées des données.

Dans le gradient Boosting, nous apprenons une fonction d'ensemble additive, généralement en utilisant des arbres de décision. Un modèle d'arbre de décision boosté est la somme des arbres individuels. Le problème d'optimisation consistant à trouver les partitions optimales et les valeurs constantes pour tous les arbres de l'ensemble est complexe. Par conséquent, une solution pas à pas proche de l'optimale est adoptée, où chaque étape est optimisée lors de la construction de l'ensemble. Algorithmes et techniques mentionnés du XGBOOST vont être incluses dans la partie annexe C.

- Les données sont prétraitées en convertissant les caractéristiques catégorielles en valeurs numériques et en comblant les valeurs manquantes.
- Un modèle de régression par forêt aléatoire XGBoost est créé et entraîné sur les données prétraitées.
- Les prédictions du modèle sont générées pour l'ensemble de test.
- Les performances du modèle sont évaluées à l'aide de plusieurs métriques, notamment l'erreur absolue moyenne (MAE), l'erreur quadratique moyenne (MSE), l'erreur absolue moyenne mise à l'échelle (MASE) et le biais de prévision.
- Les résultats de l'évaluation sont tracés et enregistrés dans un fichier.

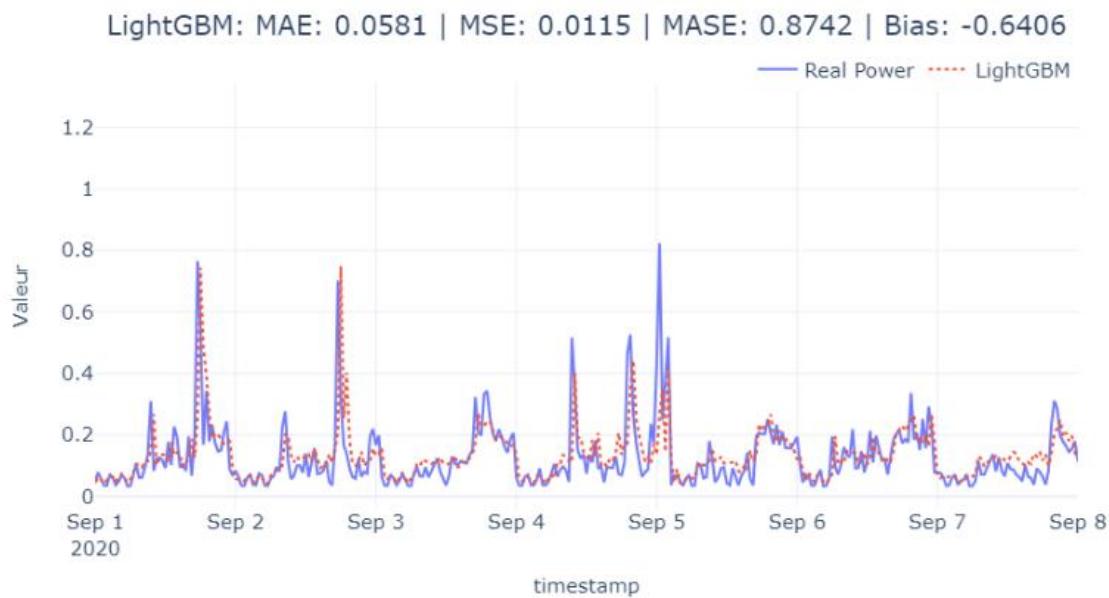


Figure 32: Résultats LightGBM

Les résultats de l'évaluation indiquent que le modèle de régression par forêt aléatoire XGBoost est capable de faire des prédictions raisonnablement précises de la consommation d'énergie. Le MAE=0.0581, le MSE=0.0114 et le MASE=0.8742 du modèle sont tous relativement faibles, et le biais de prévision est proche de zéro -0.6406. Ces résultats suggèrent que le modèle convient bien à cette tâche.

LGBMRegressor (LightGBM) est une implémentation de gradient boosting basée sur les arbres de décision. Il utilise une approche de division des données en fonction des caractéristiques pour construire les arbres et effectue la division de manière optimisée en utilisant des histogrammes pour accélérer le processus. Cela lui permet d'être plus rapide que d'autres méthodes de gradient boosting, en particulier pour les ensembles de données volumineux. Il est déjà nettement meilleur que tous les autres modèles que nous avons essayés jusqu'à présent ; De la même manière que l'importance des caractéristiques dans les arbres de décision, les méthodes de gradient boosting utilisent également un mécanisme similaire pour évaluer l'importance des caractéristiques. L'importance des caractéristiques pour l'ensemble est calculée en prenant la moyenne de la réduction des critères de division attribuée à chaque caractéristique dans tous les arbres.

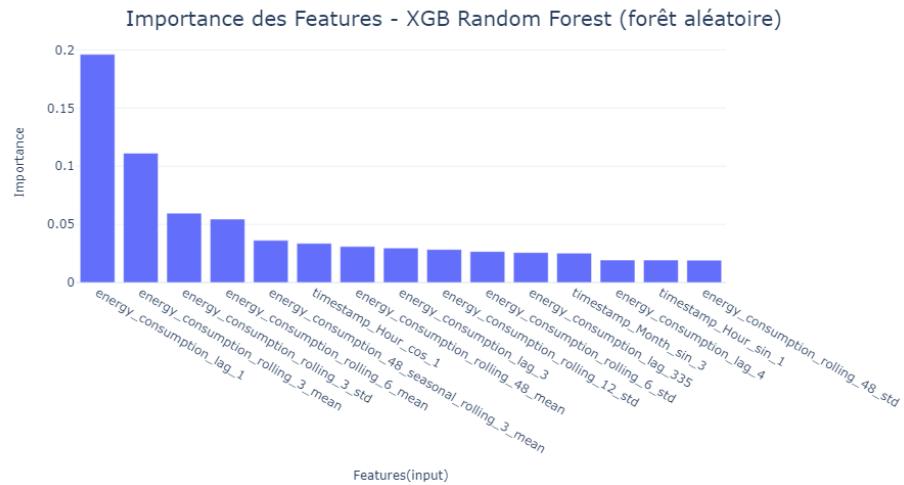


Figure 33: Importance des Features de LightGBM

2.3.6 Résultats du modèle de réseau neuronal récurrent (RNN)

Dans cette section, nous présentons l'évaluation des performances du modèle Réseau de Neurones Récurrent (RNN). Le modèle RNN a été initialisé avec une seule couche de SimpleRNN comprenant 256 unités et utilisant la fonction d'activation 'relu'. Le modèle a été entraîné sur l'ensemble de données fourni pendant 25 époques en utilisant l'optimiseur Adam avec l'erreur quadratique moyenne (MSE) comme fonction de perte.

La première étape consiste à découper la séquence de données en échantillons. Cela est réalisé par la fonction `split_sequence`, qui prend en entrée une séquence temporelle et les paramètres `n_steps_in` et `n_steps_out`. `n_steps_in` représente le nombre d'étapes temporelles utilisées comme entrée pour prédire les valeurs futures, tandis que `n_steps_out` représente le nombre d'étapes temporelles à prédire. La fonction découpe la séquence en morceaux, où chaque morceau comprend `n_steps_in` valeurs en entrée et `n_steps_out` valeurs en sortie. Les valeurs d'entrée (`X`) et de sortie (`y`) sont stockées dans des listes distinctes.

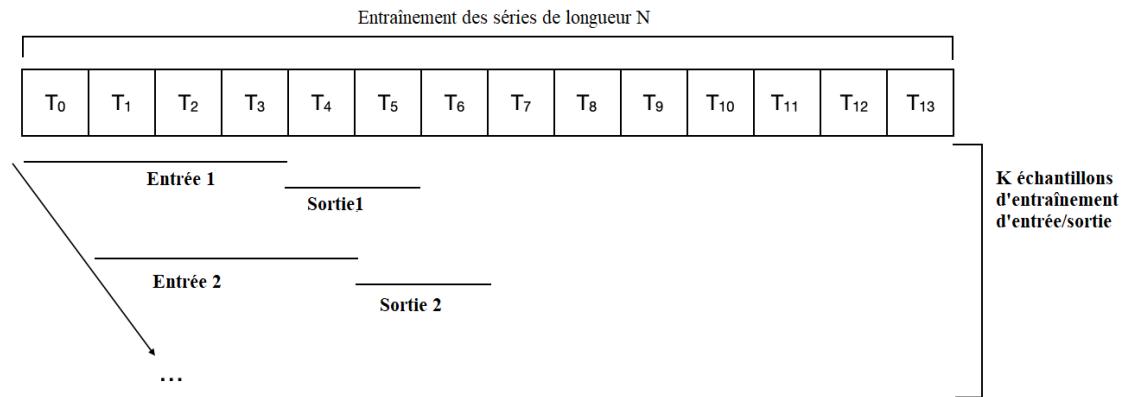


Figure 34 Procédure récurrente l'entraînement du modèle RNN

Ensuite, la séquence d'entrée X est remodelée pour correspondre à la structure d'entrée requise par le modèle RNN. La forme de X est modifiée de [samples, timesteps] à [samples, timesteps, features]. Dans ce cas, la dimension des caractéristiques (n_features) est définie à 1, car nous travaillons avec une série temporelle univariée.

Car les modèles d'apprentissage profond apprennent automatiquement les motifs et les caractéristiques à partir des données, sans nécessiter d'ingénierie des caractéristiques explicites. Dans le cas des séries temporelles univariées, le modèle peut extraire automatiquement les motifs et les dépendances temporelles pertinentes, ce qui rend l'ingénierie des caractéristiques superflue.

Le modèle RNN fonctionne de manière récurrente, ce qui lui permet de prendre en compte les covariables passées et futures. Heureusement, en tant qu'utilisateur, nous n'avons pas à nous soucier des différents types de modèles et des dimensionalités des données d'entrée/sortie, car le modèle les infère automatiquement à partir des données d'entraînement. Cependant, nous devons spécifier deux paramètres importants lors de la construction de nos modèles : `input_chunk_length`, qui détermine la fenêtre de rétroaction utilisée par le modèle, et `output_chunk_length`, qui définit la longueur des prévisions produites par le modèle interne. Si la méthode `predict()` est appelée pour un horizon temporel plus long que `output_chunk_length`, le modèle interne sera appelé de manière répétée, en se nourrissant de ses propres prédictions antérieures de manière auto-régressive. L'utilisation de covariables passées nécessite que ces covariables soient connues suffisamment à l'avance.

Performance du modèle :

La performance du modèle RNN a été évaluée à l'aide de deux métriques : l'erreur moyenne absolue (MAE) et l'erreur quadratique moyenne (MSE).

Erreur moyenne absolue (MAE) :

Le MAE mesure la différence absolue moyenne entre les valeurs prédites et réelles. Dans notre cas, le MAE a été calculé à 0,0052157887. Un MAE plus faible indique un meilleur ajustement du modèle aux données. Le MAE obtenu de 0,0052157887 démontre la capacité du modèle RNN à effectuer des prédictions avec une déviation moyenne minimale par rapport aux valeurs réelles.

Erreur quadratique moyenne (MSE) :

Le MSE mesure la différence quadratique moyenne entre les valeurs prédites et réelles. Il accorde plus de poids aux erreurs plus importantes par rapport au MAE. Le MSE pour notre modèle RNN a été calculé à 0,00047336452. De manière similaire au MAE, un MSE plus faible indique un meilleur ajustement du modèle aux données. Le MSE obtenu de 0,00047336452 démontre la capacité du modèle RNN à minimiser les écarts quadratiques entre les valeurs prédites et réelles.

Graphique résultat de la prédiction de quelque minute

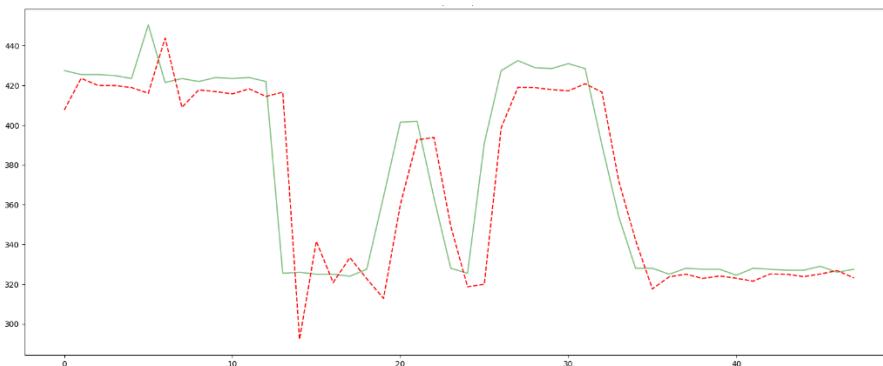


Figure 35 Prévision comparée aux valeurs actuelles en vert

Le processus implique l'utilisation de réseaux de neurones récurrents (RNN) pour effectuer des prédictions itératives sur une série temporelle. Tout d'abord, une fenêtre de rétroaction est définie, représentée par l'input_chunk_length, 360 qui spécifie combien de points passés sont utilisés comme entrée pour prédire la valeur suivante. Ensuite, le modèle RNN est entraîné sur les données d'entraînement

en apprenant les motifs et les dépendances temporelles dans la série. Une fois le modèle entraîné, il est utilisé pour générer des prévisions sur la série temporelle en utilisant les valeurs d'entrée précédentes.

La longueur des prévisions est déterminée par l'output_chunk_length qui est de 30. Le modèle itère ensuite sur ces prévisions en utilisant une approche auto-régressive, où les prédictions précédentes sont utilisées comme entrée pour générer les prédictions suivantes. Ce processus permet d'obtenir des prévisions sur n'importe quel intervalle souhaité, en utilisant les propriétés apprises par le modèle RNN à partir des données d'entraînement.

L'analyse du graphique généré indique que le modèle RNN s'adapte bien aux données. Les scores R2 de l'ensemble de test augmentent progressivement et convergent vers les scores R2 de l'ensemble d'entraînement. Cela suggère que le modèle est capable de capturer les motifs et les tendances présents dans les données et qu'il est capable de faire des prédictions précises sur l'ensemble de test. Cette observation renforce la confiance dans l'efficacité du modèle RNN pour modéliser la série temporelle sujet de la prévision.

Model: "sequential_16"		
Layer (type)	Output Shape	Param #
simple_rnn_17 (SimpleRNN)	(None, 256)	66048
dense_11 (Dense)	(None, 30)	7710
<hr/>		
Total params: 73,758		
Trainable params: 73,758		
Non-trainable params: 0		

Figure 36 Paramètres du modèle RNN

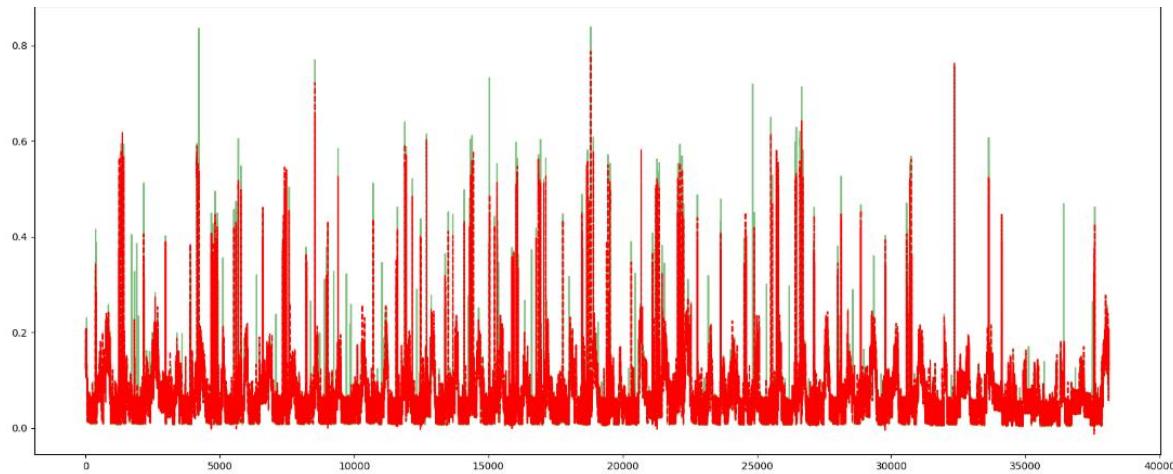


Figure 37 Prédictions sur une vaste intervalle

2.3.8 Procédure d'entraînement sur tous l'ensemble des données

Toutes les maisons ont été entraînées. La première étape consiste à initialiser les listes qui seront utilisées pour stocker les prédictions et les métriques. Dans une boucle sur la colonne des identificateurs des maisons, nous évaluons les modèles pour chaque maison. Nous pouvons paralléliser cette boucle pour accélérer le processus.

Une fois que toutes les maisons ont été traitées, les prédictions et les métriques sont retournées. (Voir l'annexe relative aux prétraitements et à l'entraînement Annexe A, B et C)

Ce processus nous permet de comprendre comment chaque modèle se comporte pour chaque maison individuellement, en tenant compte des caractéristiques spécifiques de chaque maison. Cela nous donne une évaluation plus fine de la performance des modèles sur l'ensemble des données de consommation d'énergie.

2.4. Comparaison entre des modèles en termes de métriques d'évaluation

Les résultats de l'évaluation démontrent l'efficacité des différents modèles de régression pour fournir des prévisions raisonnablement précises de la consommation d'énergie. Les modèles ont été évalués en utilisant des métriques telles que l'erreur absolue moyenne (MAE), l'erreur quadratique moyenne (MSE) et l'erreur absolue moyenne mise à l'échelle (MASE). Dans l'ensemble, les modèles ont affiché des valeurs d'erreur relativement faibles, témoignant de leur bonne performance.

Modèle	MAE	MSE	MASE	Biais
Régression linéaire	0.0651	0.0122	0.09798	-7.3018
Régression Ridge	0.0651	0.0122	0.9799	-7.3185
Régression Lasso	0.0624	0.0121	0.9386	-2.1478
Arbre de décision	0.0623	0.0128	0.9373	3.9954
Forêt aléatoire	0.0608	0.9153	0.9153	0.29
XGBoost / LightGBM	0.0581	0.0114	0.8742	-0.6406
Réseau neuronal récurrent (RNN)	0.0052	0.0005	0.0078	-0.0012

Figure 38 : Résultats des métriques d'évaluation

Le modèle de régression linéaire a obtenu un MAE de 0,0651, un MSE de 0,0122 et un MASE de 0,09798, ce qui indique sa capacité à fournir des prévisions de consommation d'énergie raisonnablement précises. De même, le modèle de régression Ridge a donné des résultats prometteurs avec un MAE de 0,0651, un MSE de 0,0122 et un MASE de 0,9799. De plus, le biais de prévision du modèle était proche de zéro, confirmant ainsi son exactitude.

En appliquant la régularisation L1 au modèle de régression Ridge, nous avons obtenu de meilleurs résultats. Le MAE était de 0,0624, le MSE de 0,0121 et le MASE de 0,9386. Notons également que le biais de prévision était de -2,1478, ce qui démontre que la régularisation L1 permet d'aligner le modèle plus étroitement avec les valeurs réelles par rapport au modèle Ridge utilisant la régularisation L2.

L'algorithme de l'arbre de décision, bien qu'exigeant en termes de calculs, a montré des performances satisfaisantes avec un MAE de 0,0623, un MSE de 0,0128 et un MASE de 0,9373, et un biais de prévision de 3,9954. Cependant, il est important de noter que les arbres de décision peuvent souffrir d'un surajustement(over-fitting), où le modèle apprend trop bien les données d'apprentissage et a du mal à s'adapter à de nouvelles données.

Le modèle de régression par forêt aléatoire a donné des prédictions raisonnablement précises de la consommation d'énergie, avec un MAE de 0,0608, un MSE de 0,9153 et un MASE de 0,9153, et un biais de prévision de 0,29.

Les résultats de l'évaluation ont mis en évidence les performances solides du modèle de régression par forêt aléatoire XGBoost de LIGHTGBM. Il a obtenu un MAE de 0,0581, un MSE de 0,0114 et un MASE de 0,8742, avec un biais de prévision proche de zéro (-0,6406). Ces résultats suggèrent que ce modèle convient particulièrement bien à la tâche de prévision de la consommation d'énergie.

Cette évaluation démontre l'efficacité des modèles de régression pour fournir des prévisions de la consommation d'énergie pour ce local résidentiel conditionnée sur l'ensemble des données.

Le modèle de régression par réseau de neurones récurrents (RNN) a également été évalué pour fournir des prévisions de la consommation d'énergie. Le modèle RNN a montré de bonnes performances avec une erreur moyenne absolue (MAE) de 0,0052 et une erreur quadratique moyenne (MSE) de 0,0005. Ces résultats démontrent la capacité du modèle RNN à ajuster les données et à effectuer des prédictions avec une déviation minimale par rapport aux valeurs réelles.

Le processus d'entraînement du modèle RNN a été analysé en surveillant les valeurs de perte (loss) et de métrique R² sur les ensembles d'entraînement et de validation. Pendant les 50 époques d'entraînement, on a observé les points suivants :

La perte d'entraînement a diminué de manière constante, passant d'une valeur initiale de 5,8880e-04 à une valeur finale de 4,7425e-04.

La métrique R2 d'entraînement a augmenté progressivement, passant de 0,8676 à 0,8938, ce qui indique une amélioration de la capacité du modèle à capturer la variance des données.

La perte de validation a également montré une tendance à la baisse, passant de 1,1243e-04 à 7,7958e-05 à la fin de l'entraînement.

La métrique R2 de validation a suivi une tendance similaire, augmentant de 0,6445 à 0,7535.

Ces résultats suggèrent que le modèle RNN a progressivement appris à minimiser les erreurs entre les valeurs prédites et les valeurs réelles. Les valeurs décroissantes des pertes d'entraînement et de validation indiquent l'amélioration continue de la capacité du modèle à s'ajuster aux données et à fournir des prédictions précises.

En conclusion, le modèle de régression par réseau de neurones récurrents (RNN) s'est avéré prometteur pour la prévision de la consommation d'énergie dans une approche univariée. Cependant, il reste plusieurs pistes d'amélioration pour l'avenir.

Dans un premier temps, il est possible d'optimiser le modèle en expérimentant différents hyperparamètres et architectures, tels que le nombre de couches, les unités cachées et les fonctions d'activation. Cette optimisation permettra de mieux capturer les motifs complexes et d'améliorer la précision des prévisions.

Ensuite, il serait intéressant d'explorer des variantes avancées de RNN, telles que les modèles LSTM (Long Short-Term Memory) ou GRU (Gated Recurrent Unit). Ces variantes sont conçues pour résoudre le problème de disparition des gradients et permettent de capturer les dépendances à long terme dans les données, ce qui peut conduire à de meilleures performances de prévision.

Dans le cadre de l'expansion du travail, une autre approche intéressante à explorer serait l'utilisation de modèles de transformation tels que les Transformers. Les Transformers sont des architectures de réseaux neuronaux qui ont démontré des performances remarquables dans le domaine du traitement du langage naturel et de la vision par ordinateur.

L'application des Transformers à la prévision de séries temporelles offre plusieurs



avantages. Ces modèles peuvent capturer les dépendances à long terme entre les points de données, ce qui est crucial dans les séries temporelles. De plus, ils sont capables d'apprendre des représentations de haute qualité à partir des données brutes, éliminant ainsi le besoin d'ingénierie de caractéristiques manuelles telles que les décalages et les caractéristiques de Fourier.

En incorporant les Transformers dans notre cadre de travail, nous pourrions explorer la possibilité de prédire la consommation d'énergie sur des horizons temporels plus étendus, en fournissant des prévisions précises à plus long terme.

3. Conclusion

En conclusion, cette évaluation démontre l'efficacité des modèles de régression pour fournir des prévisions raisonnablement précises de la consommation d'énergie. Les modèles présentent des erreurs faibles, témoignant de leur solide performance. L'entreprise peut adopter ces modèles comme outils d'aide aux décisions concernant la prévision de la consommation dans les logiciels à développer.



Conclusion Générale

Ce projet avait pour objectif principal de prédire la consommation d'énergie, ce qui revêt une grande importance dans le domaine de la gestion de l'énergie. En ayant la capacité de prédire la consommation d'énergie, on peut mieux planifier, optimiser et gérer l'utilisation des ressources énergétiques, ce qui peut conduire à des économies d'énergie, à une meilleure efficacité énergétique et à des décisions plus éclairées en matière de politique énergétique.

Le projet utilise le jeu de données MORED, qui contient des informations sur la consommation d'énergie dans un local résidentiel. Il applique différentes techniques de régression et d'apprentissage automatique pour prédire la consommation d'énergie à partir de ces données. Les méthodes utilisées comprennent la régression linéaire, la régression Ridge, la régression Lasso, l'algorithme de l'arbre de décision, la régression par forêt aléatoire (Random Forest) et le modèle XGBoost / LightGBM.

Parmi les modèles évalués, on observe que la régression linéaire et la régression Ridge ont obtenu des valeurs de MAE et MSE similaires de 0.0651 et 0.0122 respectivement. Ces valeurs indiquent une performance raisonnablement bonne. Le modèle XGBoost / LightGBM a également donné de bons résultats avec un MAE de 0.0581 et un MSE de 0.0114.

Le modèle de réseau neuronal récurrent (RNN) a montré les meilleures performances avec des valeurs de MAE, MSE et MASE très faibles de 0.0052, 0.0005 et 0.0078 respectivement. Ces valeurs témoignent de la capacité du modèle RNN à capturer les dépendances temporelles et à fournir des prévisions précises de la consommation d'énergie.

Toutefois, des optimisations supplémentaires pourraient être réalisées en expérimentant différents hyperparamètres et architectures de modèles. L'utilisation de variantes avancées de RNN telles que LSTM et GRU, ainsi que l'application de modèles de transformation comme les Transformers, pourraient également être explorées pour améliorer davantage les performances de prédiction.

Références

- [1] Grabocka, J., Schilling, N., Wistuba, M., & Schmidt-Thieme, L. (2016). Learning time-series representations for similarity search and clustering. *Data Mining and Knowledge Discovery*, 30(2), 414-442.
- [2]. Smith, R. J., & Fuertes, A. M. (2010). The forecasting performance of dynamic factor models with structural instability. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 173(1), 37-57 (s.d.). Récupéré sur <https://moredataset.github.io/MORED/>
- [3]. Hyndman, R. J., & Athanasopoulos, G. (2018). *Forecasting: Principles and Practice* (2nd ed.). OTexts.
- [4]. Seiden, A., & Eakin, M. (2017). Electric load forecasting in the era of big data. *IEEE Power and Energy Magazine*, 15(5), 65-73
- [5]. Ghalehkondabi, I., et al. (2017). Review of short-term load forecasting techniques. *Renewable and Sustainable Energy Reviews*, 69, 822-832
- [6]. *Nti IK, Adekoya AF, Weyori BA (2020) A systematic review of fundamental and technical analysis of stock market predictions. Artif Intell Rev 53:3007-3057. doi:10.1007/s10462-019-09754-z.*
- [7]. Hor, C. L., Watson, S. J., & Majithia, S. (2006). Daily load forecasting and maximum demand estimation using ARIMA and GARCH. In *2006 International Conference on Probabilistic Methods Applied to Power Systems* (pp. 1-6). IEEE
- [8]. Pappas, S. S., Ekonomou, L., Moussas, V. C., Karampelas, P., & Katsikas, S. K. (2008). Adaptive load forecasting of the Hellenic electric grid. *Journal of Zhejiang University-SCIENCE A*, 9(12), 1724-1730.
- [9]. Pappas, S. S., Ekonomou, L., Karampelas, P., Karamousantas, D. C., Katsikas, S. K., Chatzarakis, G.

- E., & Skafidas, P. D. (2010). *Electricity demand load forecasting of the Hellenic power system using an ARMA model*. *Electric Power Systems Research*, 80(3), 256-264
- [10]. An, N., Zhao, W., Wang, J., Shang, D., & Zhao, E. (2013). *Using multi-output feedforward neural network with empirical mode decomposition based signal filtering for electricity demand forecasting*. *Energy*, 49, 279-288.
- [11]. attaheian-Dehkordi, S., Fereidunian, A., Gholami-Dehkordi, H., & Lesani, H. (2014). *Hour-ahead demand forecasting in smart grid using support vector regression (SVR)*. *International transactions on electrical energy systems*, 24(12), 1650-1663.
- [12]. Halepoto, I. A., Uqaili, M. A., & Chowdhry, B. S. (2014). *Least square regression based integrated multi-parameteric demand modeling for short term load forecasting*. *Mehran University Research Journal of*
- [13]. Din, G. M. U., & Marnerides, A. K. (2017). *Short term power load forecasting using deep neural networks*. In *2017 International conference on computing, networking and communications (ICNC)* (pp. 594-598). IEEE.
- [14]. Ouyang, T., He, Y., Li, H., Sun, Z., & Baek, S. (2017). *A deep learning framework for short-term power load forecasting*. <https://doi.org/10.48550/arXiv.1711.11519>
- [15]. Bashir, T., Haoyong, C., Tahir, M. F., & Liqiang, Z. (2022). *Short term electricity load forecasting using hybrid prophet-LSTM model optimized by BPNN*. *Energy Reports*, 8, 1678-1686.
- [16] Lazzeri, F. (Year). *Machine Learning for Time Series Forecasting with Python*. John Wiley.
- [17] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.
- [18] Introduction to Time Series Forecasting With Python, Jason Brownlee · 2017
- [19] Bishop, C.M. *Pattern Recognition and Machine Learning*. Publisher Name, 2006



ANNEXES

Annexe A : Prétraitement

Collecte et Transformation des Données à partir des Sources Brutes

Les données ont été prétraitées en suivant une série de transformations sur la structure brute. Chaque série de données est présente dans un fichier Excel, et l'objectif était de concaténer la consommation de tous les ménages afin d'unifier l'emplacement des données pour les futures implémentations du code. Un autre objectif était de réduire l'empreinte mémoire, en particulier lorsque l'ensemble de données est augmenté, comme on le verra dans les sections suivantes de l'annexe, pouvant atteindre 2 à 3 Go. Il est donc impératif de résoudre cette problématique dans une optique d'optimisation des temps d'exécution du programme.

MORED propose les données de consommation suivantes :

- Consommation réelle d'électricité pour les catégories de mesure WP et IL, désignée par l'acronyme WPILGT (WP et IL Ground-Truth)
- Consommation d'électricité pour les WP catégories de mesure, désignée par l'acronyme WP WP
- Signatures IL

Remarque : Chaque type de jeu de données possède ses propres métadonnées qui fournissent certaines informations sur les ménages de MORED. Les métadonnées peuvent inclure des détails supplémentaires tels que les données démographiques des ménages, leur emplacement ou d'autres informations pertinentes liées à la consommation d'électricité.

```
: WP_data_Chemin = data_path/"WP_data"
WPILGT_data_Chemin = data_path/"WPILGT_data"
meta_data_Chemin = data_path
```

Nous définissons une représentation des séries temporelles pour gérer la collecte de données à partir de différents dossiers et réduire la consommation de mémoire, tout en recherchant également des structures de données en Python qui réduiront l'empreinte mémoire.

```

def segment(Dataframe: pd.DataFrame):
    """
    La définition Mathématique est appliquée avec une légère modification par exemple en utilisant ts_length à la place du
    date de début ts...
    """
    Debut = Dataframe['timestamp'].min()
    ID = Dataframe['Premises'].unique()[0]
    # trier les rangs
    DataFrame.sort_values(['timestamp'], inplace=True)
    # extraction du 'array' des séries temporelles
    ts_vrms = DataFrame['Vrms'].values
    ts = DataFrame['real_power'].values
    ts_length = len(ts)
    return Debut, ID, ts, ts_length, ts_vrms
#E1

def TS_segmentation(Dataframe: pd.DataFrame, freq="10S", ts_id="series_name", \
                     col1_name="real_power", col2_name="series_value") -> pd.DataFrame:
    Ensemble_ts = []
    Ensemble_ts_vrms = []
    Ensemble_debut = []
    Ensemble_ID = []
    Ensemble_donnée = {}
    Ensemble_len = []
    for idx, datafr in tqdm(Ensemble, leave=False):
        Debut, ID, ts, ts_length, ts_vrms = segment(datafr)
        Ensemble_ts.append(ts)
        Ensemble_ts_vrms.append(ts_vrms)
        Ensemble_debut.append(Debut)
        Ensemble_ID.append(ID)
        Ensemble_len.append(ts_length)
        Ensemble_donnée[ts_id] = Ensemble_ID
        Ensemble_donnée['debut_ts'] = Ensemble_debut
        Ensemble_donnée['sampling_rate'] = freq
        Ensemble_donnée[col1_name] = Ensemble_ts
        Ensemble_donnée[col2_name] = Ensemble_ts_vrms
        Ensemble_donnée['ts_len'] = Ensemble_len
    return pd.DataFrame(Ensemble_donnée)

```

Les données sont ensuite collectées à partir des dossiers avec les paramètres appropriés tels que la définition de la fréquence, et une fonction de réduction de l'empreinte mémoire est appliquée colonne par colonne.

```

# Les données correspondant à la puissance totale de chaque foyer sont extraites des fichiers csv
# situés dans le dossier WP. Liste_Locaux_residentiels_Dataframe = []
# WP_data_Chef Le chemin Os contenant les données WP
for fichier in tqdm(list(WP_data_Chef.glob("*.csv")), \
                     desc="Traitement des données des locaux d'habitation dans une seule trame de données..."): #
    Local_df = pd.read_csv(fichier, parse_dates=False)
    Local_df['timestamp'] = pd.to_datetime(Local_df['timestamp'], dayfirst=True)
    # Réduire l'empreinte mémoire
    Local_df['Vrms'] = Local_df['Vrms'].astype("float32")
    Local_df['real_power'] = Local_df['real_power'].astype("float32")
    s = fichier.name
    Local_df['Premises'] = s[0:s.find('.')] #une nouvelle colonne est créée pour identifier de manière unique un local.
    liste_locaux_residentiels_Dataframe.append(TS_segmentation(Local_df, \
                                                               freq="10S", \
                                                               ts_id="Premises", \
                                                               col1_name="real_power", col2_name="Vrms"))

Traitement des données des locaux d'habitation dans une seule trame de données...:  0% | 0/12 [00:0...
```

0%| 0/1 [00:00<?, ?it/s]
0%| 0/1 [00:00<?, ?it/s]

Nous pouvons constater qu'à partir d'un volume de données de 300 ~ 400 Mo, l'empreinte mémoire a été réduite à 0,00444 Mo grâce à de légères modifications telles que la conversion de type (par exemple, de float 64 bits à float 32 bits).

```
#Attribuer La DataFrame concaténée à la variable WP_DataFrame.
WP_DataFrame = pd.concat(liste_locaux_residentiels_Dataframe).reset_index().drop(columns = 'index')
# Cette liste de dataframe n'est plus utile car l'information maintenant est stockée dans un
# nouveau format plus efficace et de mémoire très réduite (résultat : Total: 0.0030660629272460938 MB)
del liste_locaux_residentiels_Dataframe

display(WP_DataFrame.memory_usage(deep=True))
print(f"Total: {WP_DataFrame.memory_usage(deep=True).sum()/1024**2} MB")
```

Index	128
Premises	735
debut_ts	96
sampling_rate	720
real_power	1440
Vrms	1440
ts_len	96
dtype:	int64

Total: 0.004439353942871094 MB

De la même manière, la catégorie WPILGT a été collectée à partir des dossiers contenant les fichiers Excel individuels, puis concaténée dans un seul référentiel, qui est une structure de données de type DataFrame de Pandas en Python. Ensuite, les catégories précédentes et suivantes, WP et WPILGT, seront concaténées.

```
"""Parcourir les répertoires et les sous-répertoires des ménages pour lire le fichier mains.csv de
chaque ménage et le représenter avec un identifiant comme suit : WPILGT_i où i est le numéro du domicile.
dirname = WPILGT_data_Chemin"""

def walkdir(dirname):
    liste_locaux_residentiels_Dataframe = []
    for cur, _dirs, files in tqdm(os.walk(dirname)):
        for fichier in tqdm(files):
            if fichier == 'Mains.csv':
                print(cur+'\\"'+fichier)
                P = Path(cur)/fichier #P for Path
                Local_df = pd.read_csv(P, parse_dates=False)
                Local_df['timestamp'] = pd.to_datetime(Local_df['timestamp'], dayfirst=True)
                Local_df= Local_df.resample('10S', on='timestamp').mean()
                # The index is auto set to timestamp columns which can cause unexpected behaviours ..
                Local_df = Local_df.reset_index()
                # Reduce memory footprint:
                Local_df['Vrms'] = Local_df['Vrms'].astype("float32")
                Local_df['real_power'] = Local_df['real_power'].astype("float32")
                _KK = cur.split("/")
                s = _KK[-1]
                #La ligne suivante modifie la référence ID des locaux, elle ne doit pas être appelée avant la méthode merge.
                Local_df['Premises'] = 'WPILGT_'+s #Une nouvelle colonne est créée pour identifier de manière unique un local.
                liste_locaux_residentiels_Dataframe.append(Local_df, freq="10S", ts_id="Premises", coll_name="real_p
    return liste_locaux_residentiels_Dataframe

#ID: R9
liste_locaux_residentiels_Dataframe = walkdir(dirname = WPILGT_data_Chemin)

<   >
0it [00:00, ?it/s]
0it [00:00, ?it/s]
0%|          | 0/10 [00:00<?, ?it/s]
data\Mored_dataset\WPILGT_data\Premises 1\Mains.csv
0%|          | 0/1 [00:00<?, ?it/s]
```

Pour inverser le processus et obtenir la série temporelle indexée par le temps, nous pouvons procéder de la manière suivante en utilisant cette fonction :

```
from collections import defaultdict
def allonger(Dataframe: pd.DataFrame, ts_col, static_col, Variable_F_temps_col, ts_id ):
    """
    Cette fonction a pour objectif d'inverser la transformation segmenté :
    **kwargs : Prend en argument les composantes d'un segment de série temporelle, à savoir l'heure de début et la fréquence.

    """
    def pre_allonger(x):
        # Créer une plage de dates à partir de start Le début
        dr = pd.date_range(
            start=x["debut_ts"],
            periods=len(x["real_power"]),
            freq=x["sampling_rate"],
        )
        df_columns = defaultdict()
        df_columns["timestamp"] = dr
        for col in ts_id+ ts_col + static_col + Variable_F_temps_col:
            df_columns[col] = x[col]
        return pd.DataFrame(df_columns)
    Ensemble_donnée = []
    for i in tqdm(range(len(Dataframe))):
        Ensemble_donnée.append(pre_allonger(Dataframe.iloc[i]))
    Dataframe = pd.concat(Ensemble_donnée)
    del Ensemble_donnée
    return Dataframe
```

Dans cet exemple de code, nous utilisons la fonction précédente pour récupérer la série temporelle provenant de la locale résidentiel étiquetée 'WPILGT_7', c'est-à-dire le septième locale résidentiel de la catégorie de mesure WPILGT.

Exemple

- Sur la maison WPILGT_7

```
#Prototype pour définir les valeurs d'entrée par défaut, les noms
ts_col = ['real_power', 'Vrms'] #Série chronologique
static_col = ['debut_ts', 'sampling_rate','ts_len'] # des informations supplémentaires sur la résolution temporelle (ou les coordonnées)
Variable_F_temps_col=[] # .
ts_id=['Premises'] # Identifiant
df_eg = WPILGT_DataFrame[WPILGT_DataFrame.Premises =='WPILGT_7']
df_eg = allonger(df_eg, ts_col, static_col, Variable_F_temps_col, ts_id )
```

	timestamp	Premises	real_power	Vrms	debut_ts	sampling_rate	ts_len
0	2020-11-07 19:26:50	WPILGT_7	318.0	239.149994	2020-11-07 19:26:50	10S	1505000
1	2020-11-07 19:27:00	WPILGT_7	312.0	238.985001	2020-11-07 19:26:50	10S	1505000
2	2020-11-07 19:27:10	WPILGT_7	312.0	239.259995	2020-11-07 19:26:50	10S	1505000
3	2020-11-07 19:27:20	WPILGT_7	312.5	238.389999	2020-11-07 19:26:50	10S	1505000
4	2020-11-07 19:27:30	WPILGT_7	312.5	238.835007	2020-11-07 19:26:50	10S	1505000

Finalement, nous obtenons le dataframe contenant toutes les colonnes des différentes caractéristiques statiques et variables par rapport au temps des différentes résidences de l'ensemble des données. Dans les prochaines sections, ce dataframe fera l'objet d'une augmentation en appliquant différentes techniques d'ingénierie des caractéristiques.

DATA.head()											
	Premises	debut_ts	sampling_rate	real_power	Vrms	ts_len	ownership	building_type	affluent_neighborhood_apartment	affluent_neighborhood_semi_detached_house	disadvantaged_neighborhood_apartment
0	WPILGT1	2020-06-01 18:11:30	10S	[392.0, 388.5, 390.0, 388.6, 392.5, 398.0, 395...	[224.96, 224.785, 224.945, 225.005, 225.485, 2...	192011	rented	flat	0	0	0
1	WPILGT2	2020-09-17 21:14:50	10S	[302.5, nan, nan, nan, nan, nan, nan, ...	[229.23, nan, nan, nan, nan, nan, nan, ...	506354	rented	flat	0	0	0
2	WPILGT3	2020-06-27 20:41:00	10S	[306.0, 308.0, 335.0, 337.0, 335.6, 330.5, 330...	[306.0, 308.0, 335.0, 337.0, 335.6, 330.5, 330...	680724	rented	flat	0	0	0
3	WPILGT4	2020-07-14 23:00:00	10S	[6.0, 5.0, 8.5, 7.5, 7.0, 5.5, 7.5, 5.5, 5.5, ...	[152.06, 153.08, 153.585, 153.785, 153.1, 153.1, 153...	512999	rented	flat	1	1	1
4	WPILGT5	2020-07-13 16:52:10	10S	[154.5, 150.0, 148.5, nan, nan, nan, nan, ...	[227.215, 228.935, 228.505, nan, nan, nan, nan, ...	382368	rented	flat	1	1	1

5 rows × 21 columns

Grâce à la méthode `.memory_usage()` appliquée sur l'objet Python DATA qui représente notre dataframe, nous pouvons accéder à l'empreinte mémoire des différentes colonnes de l'ensemble des données et avoir une idée sur le nombre d'exemples traités ainsi que les types de variables que Python a inférés en les détectant automatiquement. Enfin, il faut noter que l'ensemble de données représenté de cette façon est optimal en termes d'empreinte mémoire, étant donné sa taille initiale de 400 Mo.

```
display(DATA.memory_usage(deep=True))
print(f"Total: {DATA.memory_usage(deep=True).sum() / 1024**2} MB")
print("Or in bytes")
print(f"Total: {DATA.memory_usage(deep=True).sum()} B")
```

Index	104
Premises	804
debut_ts	104
sampling_rate	780
real_power	1560
Vrms	1560
ts_len	104
ownership	818
building_type	793
affluent_neighborhood_apartment	104
affluent_neighborhood_semi_detached_house	104
disadvantaged_neighborhood_apartment	104
city_of_residence	896
number_of_occupants	104
number_of_elderly	104
number_of_adults	104
number_of_rooms	104
number_of_floors	104
area	104
total_acquisition_duration	104
mean_daily_consumption	104
MORED_Category	1199
dtype: int64	
Total: 0.009408950805664062	MB
Or in bytes	
Total: 9866	B

ANNEXE B-Traitement Automatique des Valeurs Manquantes dans les Données

Dans les premières étapes du processus d'apprentissage automatique, avant même l'analyse des données, il est nécessaire d'avoir une compréhension approfondie de l'ensemble des données et de connaître les différentes résolutions dans le temps et l'espace. Une bibliothèque qui facilite ce processus est **missingno** en Python, qui affiche les plages de temps de toutes les séries temporelles alignées pour évaluer les limites des données et vérifier s'il existe des données manquantes dans l'ensemble de données.

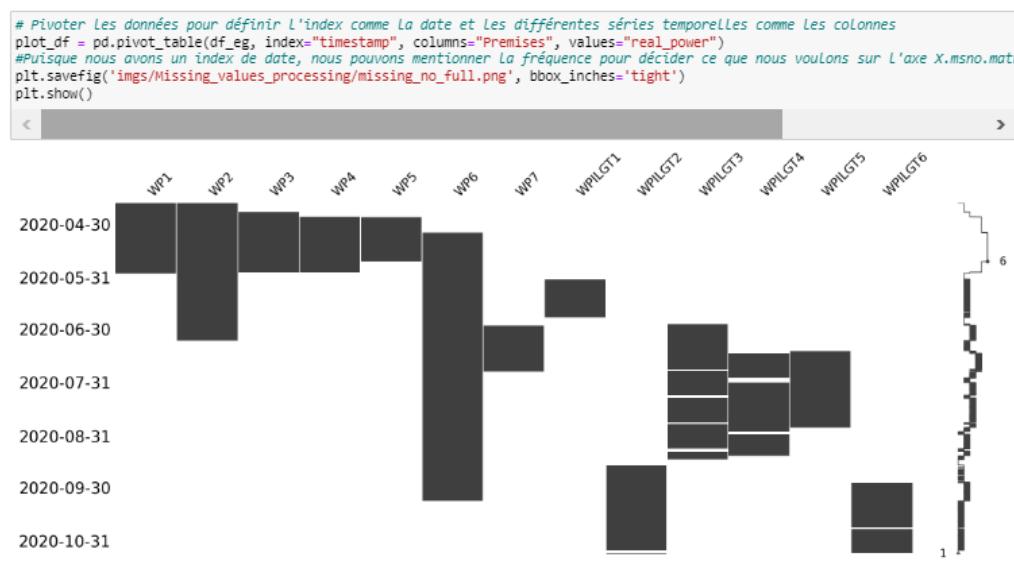


Figure 39 Visualisation de la plage des dates de l'ensemble des données

Python offre également un ensemble de fonctions et de méthodes pour gérer les valeurs manquantes. Par exemple, le "backfill" tente d'inférer les valeurs manquantes à partir des valeurs précédentes. Différentes techniques sont également implémentées pour fonctionner dans l'environnement Python. Cependant, ces méthodes restent souvent inefficaces et peu fiables, car elles peuvent entraîner de nombreuses erreurs lors de l'entraînement, étant donné que ces motifs manquants sont éloignés du processus qui génère les données.

Pour remédier à cela, on peut utiliser une interpolation simple ou faire appel à une interpolation saisonnière, tout en remplissant les variables manquantes de manière dynamique en fonction des moyennes de consommation horaire ou hebdomadaire. Cela donne du sens aux nouvelles valeurs créées, qui présentent peu de problèmes et n'affectent pas le comportement des modèles.

Dans cet exemple, nous montrons comment il est possible d'implémenter la logique expliquée précédemment pour traiter le problème. Nous utilisons la fenêtre du jour précédent pour remplir la valeur manquante avec la valeur du jour précédent.

Dernière journée

```
#Shifting 6sec*60min*24heures étapes pour obtenir le jour précédent
ts_df[\"prev_day\"] = ts_df[\"real_power\"].shift(6*60*24) # 6sec*60min*24heures
#utilisation de la colonne décalée pour combler les lacunes
ts_df[\"prev_day_imputed\"] = ts_df[\"real_power_missing\"] 
ts_df.loc>null_mask,\"prev_day_imputed\"] = ts_df.loc>null_mask,\"prev_day\"] 
mae = mean_absolute_error(ts_df.loc>window, \"prev_day_imputed\").fillna(method='backfill'), ts_df.loc>window, \"real_power\").fillna(method='backfill')
```

L'ajout de variables catégoriques temporelles peut également aider à remplir les valeurs manquantes en consultant des informations catégoriques relatives à la valeur actuelle du temps, afin de se référer aux valeurs voisines dans le temps du même type. Par exemple, le jour de la semaine, le début ou la fin du mois.

```
ts_df[\"weekday_name\"] = ts_df.index.day_name()
ts_df[\"weekday\"] = ts_df.index.weekday
ts_df[\"week\"] = ts_df.index.isocalendar().week
ts_df[\"day\"] = ts_df.index.day
ts_df[\"hour\"] = ts_df.index.hour
ts_df[\"date\"] = ts_df.index.date
ts_df[\"month\"] = ts_df.index.month
ts_df[\"month_name\"] = ts_df.index.month_name()
ts_df[\"year\"] = ts_df.index.year
```

Problème de la non stationnarité

Le problème de non-stationnarité peut être adressé, car une série non stationnaire peut poser un problème majeur pour différents modèles. Ces problèmes sont souvent résolus en appliquant une transformation aux données pour améliorer la prédiction, puis en appliquant une transformation inverse pour récupérer les valeurs réelles dans l'échelle d'origine.

Dans l'exemple suivant, nous montrons comment avoir une idée de la stationnarité d'une série temporelle en effectuant le test augmenté de Dickey-Fuller. Il s'agit d'un test statistique visant à déterminer si une série temporelle est stationnaire, c'est-à-dire si ses propriétés statistiques (espérance, variance, autocorrélation) varient ou non dans le temps. Pour la majorité des locaux résidentiels, le test présente des p-valeurs inférieures au seuil par défaut de 0,005, ce qui permet de rejeter l'hypothèse nulle selon laquelle la série est non stationnaire.

```
# Prendre une seule série temporelle
ts_df = df_eg[df_eg.Premises=="WPILGT6"].set_index("timestamp")

from statsmodels.tsa.stattools import adfuller, breakvar_heteroskedasticity_test
series = ts_df.real_power
X = series.values
result = adfuller(X)
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
    print('\t%s: %.3f' % (key, value))

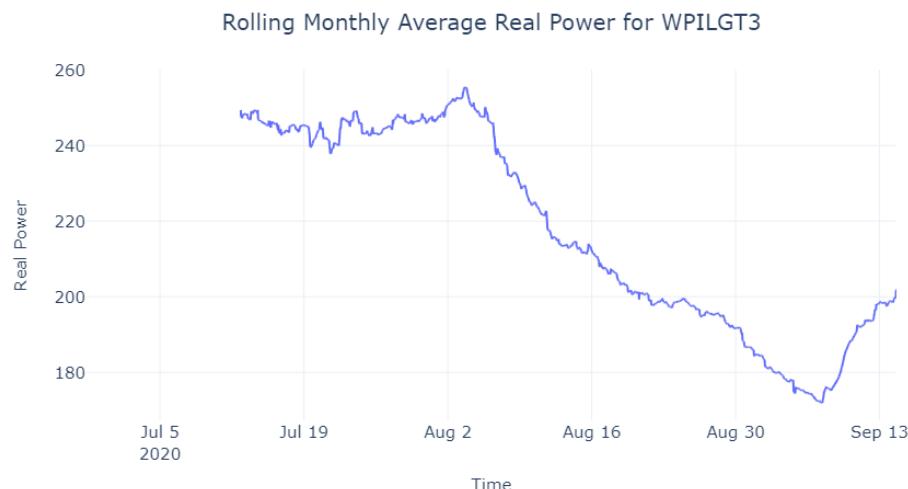
ADF Statistic: -3.861138
p-value: 0.002340
Critical Values:
    1%: -3.437
    5%: -2.865
    10%: -2.568
```

La visualisation des données des séries temporelles est également une étape importante dans l'analyse des données. Avant de faire des hypothèses sur les données et des tests statistiques sur les données d'échantillon, de nombreux motifs peuvent être facilement identifiés grâce à la visualisation sous différentes résolutions et avec les transformations appropriées. Par exemple, l'application d'une moyenne mobile ou de fonctions d'agrégation telles que la moyenne permet de capturer différentes caractéristiques des données telles que la saisonnalité, la tendance générale ainsi que la complexité et les variations brèves

La visualisation des données

Graphique de la moyenne mobile

En fixant une fenêtre temporelle, l'utilisation d'une moyenne mobile peut nous donner des informations sur la tendance de la série selon la résolution souhaitée. Par exemple, en appliquant une moyenne mobile mensuelle, nous pouvons capturer le profil général mensuel de la série.



Saisonnalité journalière à une résolution horaire

La "saisonnalité journalière à une résolution horaire" fait référence aux variations régulières qui se produisent quotidiennement à l'échelle des heures dans une série temporelle. Cela permet d'identifier des motifs spécifiques qui se répètent chaque jour à des moments précis.

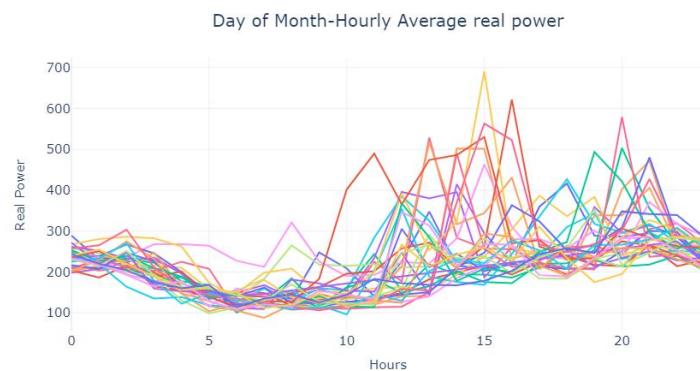


Figure 40 Profile journalier en une résolution mensuelle

Graphiques saisonniers (Boîte à moustaches)

Les "graphiques saisonniers (boîte à moustaches)" permettent de visualiser les variations saisonnières des données à chaque heure de la journée. Il fournit des informations sur la distribution des valeurs et les tendances spécifiques à chaque période de la journée, mettant en évidence les schémas saisonniers qui se produisent quotidiennement.

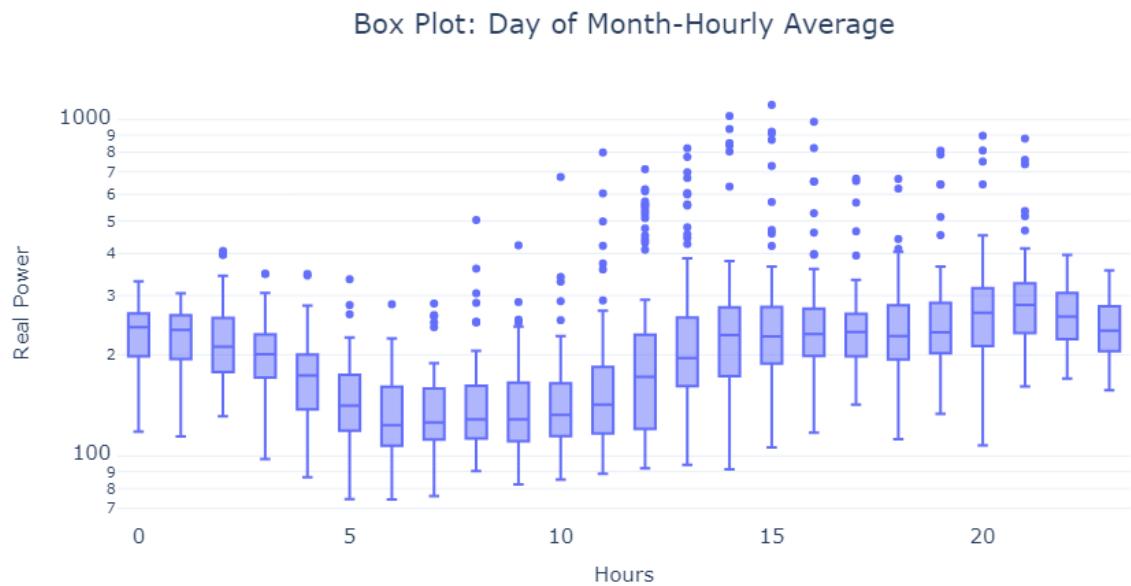


Figure 41 Boîte à moustaches du profile des jours du mois

Graphiques carte thermique (Heatmap) de la consommation journalière en une résolution hebdomadaire :

Après avoir visualisé les données de consommation, il a été observé qu'il existe un schéma constant de consommation d'énergie plus élevée lors des pics du matin et du soir, ce qui indique la routine quotidienne typique des résidents. Ce schéma suggère une saisonnalité journalière claire dans la consommation d'énergie, avec une utilisation plus faible pendant les heures diurnes lorsque la plupart des résidents sont absents. Ces résultats mettent en évidence l'importance d'analyser et de comprendre les motifs temporels de la consommation d'énergie pour optimiser la planification et la prévision énergétiques.

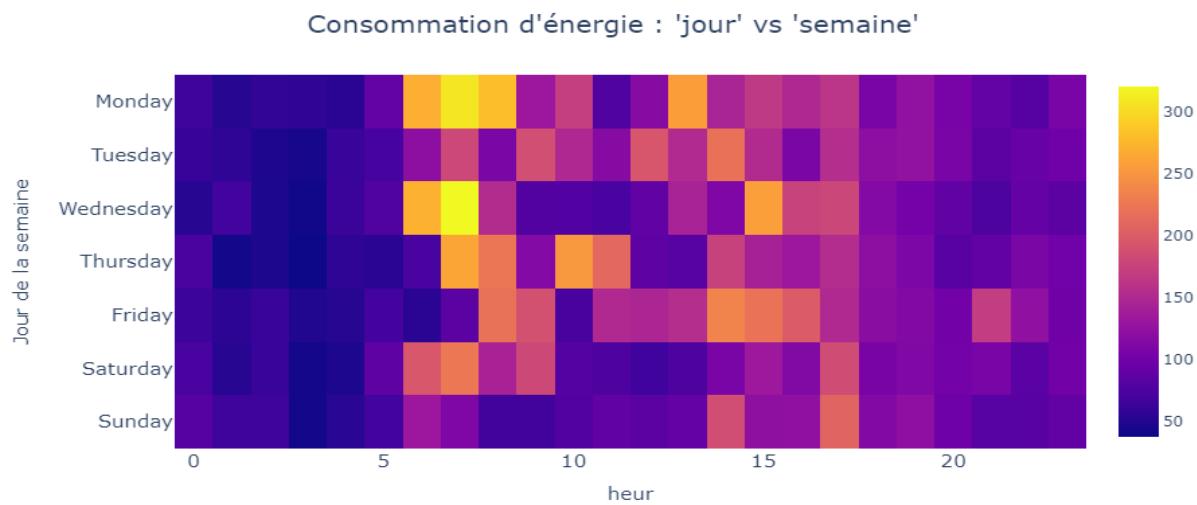


Figure 42 Graphic thermique du profile journalier en une résolution hebdomadaire

Après avoir visualisé le graphique en nuance de couleurs représentant la consommation d'énergie en fonction du jour de la semaine et de l'heure de la journée, plusieurs observations ont été remarquées. Tout d'abord, les jours de semaine présentent des pics de consommation d'énergie plus élevés le matin et le soir, correspondant aux périodes de pointe des activités quotidiennes. En revanche, les week-ends montrent une consommation d'énergie plus uniforme et plus faible tout au long de la journée.

Cette représentation visuelle met en évidence la saisonnalité journalière et hebdomadaire de la consommation d'énergie résidentielle. Elle confirme également la tendance générale observée précédemment, montrant que les résidents utilisent davantage d'énergie pendant les heures de réveil et de retour à la maison en semaine, tandis que la consommation est plus régulière et moins élevée les week-ends.

Ces résultats soulignent l'importance de prendre en compte les motifs temporels et les différences entre les jours de la semaine lors de l'analyse et de la planification de la consommation d'énergie dans le secteur résidentiel. Cela permet de mieux comprendre les habitudes de consommation et d'optimiser les stratégies d'utilisation et d'économie d'énergie.

Moyennes mobiles

L'utilisation des moyennes mobiles dans la décomposition saisonnière permet d'extraire la composante de tendance, ce qui facilite une meilleure compréhension du comportement à long terme de la série temporelle. Cette approche, combinée à l'identification des motifs saisonniers et à la caractérisation des valeurs résiduelles, offre des informations précieuses pour une analyse ou une modélisation ultérieure des données.

Pour une série temporelle de consommation d'électricité sur une période de 3 mois avec un taux d'échantillonnage de 10 secondes, il est difficile de déterminer le modèle le plus approprié sans plus d'informations. Cependant, si la variation saisonnière de la consommation d'électricité reste relativement constante indépendamment du niveau global, un modèle additif pourrait être approprié. En revanche, si la variation saisonnière est proportionnelle au niveau de consommation, un modèle multiplicatif pourrait être plus adapté.

```
#Ne supporte pas les valeurs manquantes
res = seasonal_decompose(ts, period=7*24*60*6, model="additive", \
    extrapolate_trend="freq", filt=np.repeat(1/(30*24*60*6), 30*24*60*6))
```

Après avoir procédé à une décomposition saisonnière des données, j'ai constaté une tendance stable ainsi qu'un léger motif saisonnier. Par ailleurs, les résidus présentent une variance stable globalement, ce qui suggère la présence d'un bruit gaussien.



Figure 43 Décomposition Saisonnalité et Tendance par Loess

Décomposition de la saisonnalité et de la tendance à l'aide de Loess (STL)

La Décomposition Saisonnalité et Tendance par Loess : est une technique qui décompose une série temporelle en trois composantes : la saisonnalité, la tendance et les résidus. Elle utilise une régression pondérée localement (Loess) pour estimer la composante de tendance et la supprime de la série originale. La composante saisonnière est obtenue en faisant la moyenne des résidus à chaque instant, et les valeurs restantes représentent la composante des résidus.

```
#Supporte les valeurs manquantes et attend une série ou un cadre de données avec un index de date
stl = STL(seasonality_period=7*24*60*6, model = "additive")
res_new = stl.fit(ts_df.real_power)
```

Lors de l'application de la méthode de décomposition de la saisonnalité et de la tendance à l'aide de Loess (STL), j'ai obtenu une tendance stable et un profil saisonnier qui se répète chaque semaine, composé de sept motifs distincts correspondant à chaque jour de la semaine. Les résidus restants sont considérés comme un bruit inexpliqué. Ces résultats confirment la présence de structures saisonnières et tendancielles dans les données, avec des variations régulières récurrentes. Les composantes résiduelles fournissent des informations supplémentaires sur les variations non expliquées par les composantes saisonnières et tendancielles.

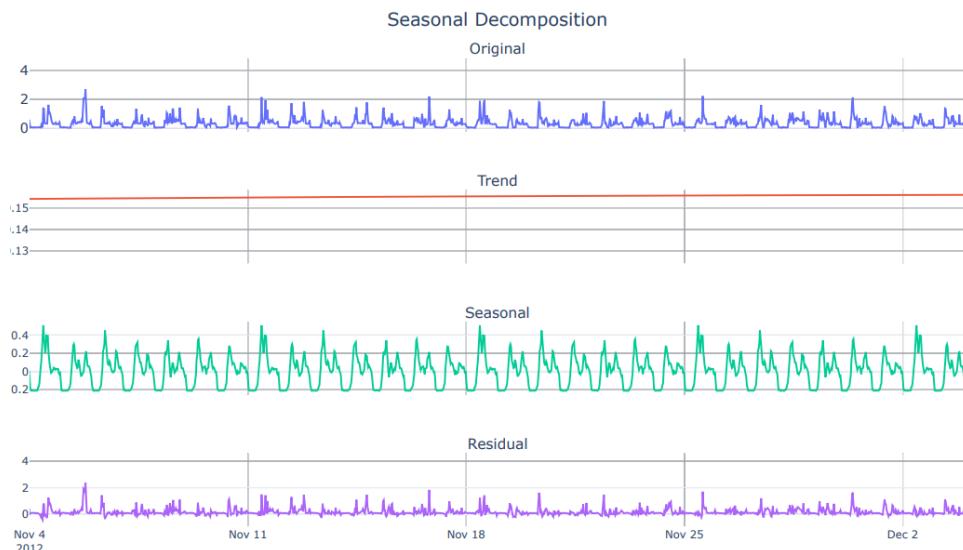


Figure 44 Décomposition Saisonnalité et Tendance par Loess

Décomposition de la saisonnalité et de la tendance à l'aide de Loess et des termes de Fourier (Décomposition de Fourier)

La décomposition de Fourier et le lissage de Loess sont des techniques complémentaires qui permettent de capturer à la fois les variations saisonnières et la tendance dans les séries temporelles, offrant ainsi une meilleure compréhension des motifs et des comportements sous-jacents.

Dans cette approche, les termes de Fourier sont utilisés pour capturer les variations saisonnières en fréquence. Par exemple, si nous utilisons le jour de la semaine comme variable catégorique, nous créons des termes de Fourier correspondant à chaque jour de la semaine. Ces termes de Fourier permettent de représenter les fréquences hebdomadaires dans les données, en capturant les variations récurrentes chaque semaine.

Les accessoires temporels de Pandas facilitent l'extraction et la manipulation des informations temporelles dans les séries temporelles, tels que le jour de la semaine, l'heure, etc., permettant ainsi une analyse plus précise et granulaire des données.

Personnalisation de la saisonnalité

Les accessoires temporels de Pandas facilitent l'extraction et la manipulation des informations temporelles dans les séries temporelles, tels que le jour de la semaine, l'heure, etc., permettant ainsi une analyse plus précise et granulaire des données.

```
#Créer des terme de saisonnalité personnalisé
ts_df["dayofweek"] = ts_df.index.dayofweek
ts_df["hour"] = ts_df.index.hour
#Créer une combinaison unique triée df
map_df = ts_df[["dayofweek", "hour"]].drop_duplicates().sort_values(["dayofweek", "hour"])
# Attribution d'une variable ordinaire pour capturer l'ordre
map_df["map"] = np.arange(1, len(map_df)+1)
#ajoute la représentation ordinale à df
seasonality = ts_df.merge(map_df, on=["dayofweek", "hour"], how='left', validate="many_to_one")['map']
```

Les résultats de la méthode Loess avec les termes de Fourier ont confirmé les motifs saisonniers identifiés dans la visualisation en carte thermique (Heatmap), en mettant en évidence les différentes variations saisonnières particulière à l'heure et au jour de la semaine.

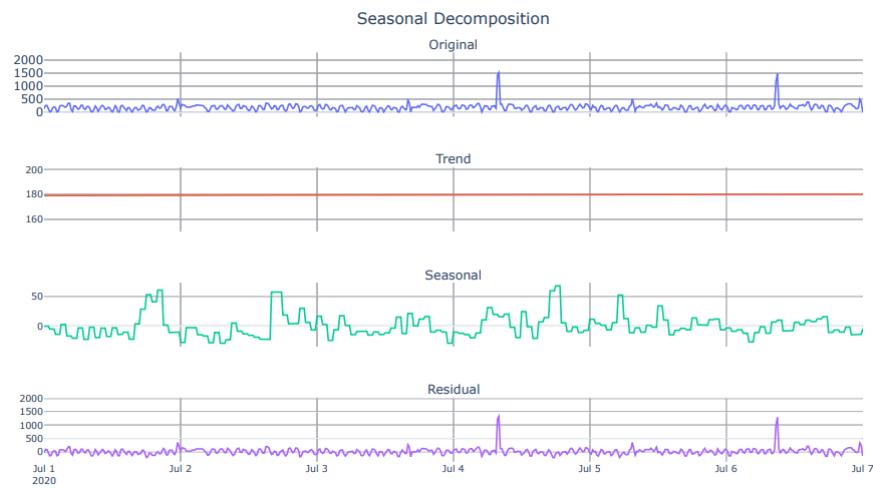


Figure 45 Décomposition de la saisonnalité et de la tendance à l'aide de Loess et des termes de Fourier (Décomposition de Fourier)

Décomposition de la saisonnalité multiple à l'aide de Loess (MSTL) en utilisant des moyennes comme modèle saisonnier

La décomposition de la saisonnalité multiple à l'aide de Loess (MSTL) en utilisant des moyennes comme modèle saisonnier est liée à l'utilisation des moyennes mobiles sur des fenêtres temporelles passées.

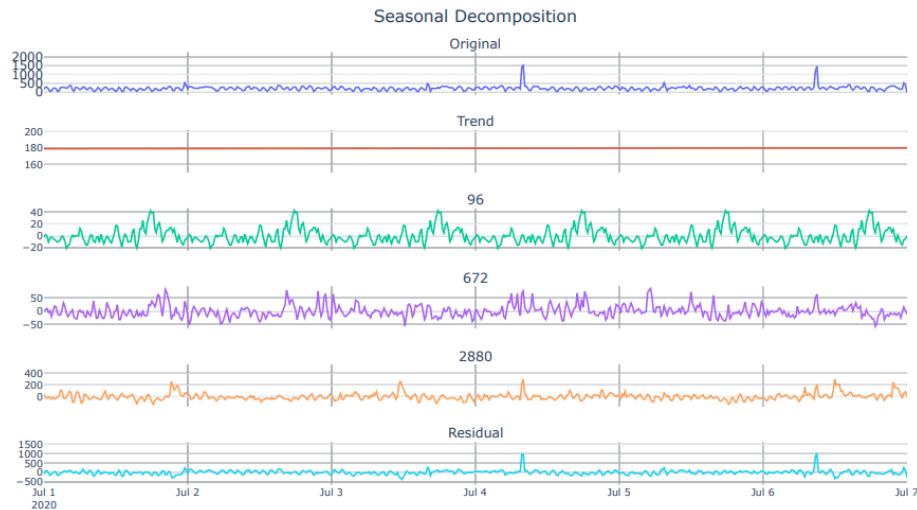


Figure 46 Décomposition de la saisonnalité et de la tendance à l'aide de Loess et des termes de Fourier (Décomposition de Fourier)

La décomposition de la saisonnalité multiple à l'aide de Loess (MSTL) en utilisant des moyennes comme modèle saisonnier est liée à l'utilisation des moyennes mobiles sur des fenêtres temporelles passées. Cela permet de capturer les tendances et les motifs saisonniers dans les données. Cette approche peut être utile lors de la phase d'ingénierie des caractéristiques pour sélectionner différentes lags (retards) en tant que caractéristiques, ou pour calculer des

moyennes mobiles ou des moyennes sur ces fenêtres pour obtenir des informations supplémentaires sur les motifs temporels et les variations saisonnières. Ces informations peuvent être utilisées pour améliorer les modèles de prévision et l'analyse des séries temporelles.

Utilisation de la décomposition de Fourier comme modèle saisonnier

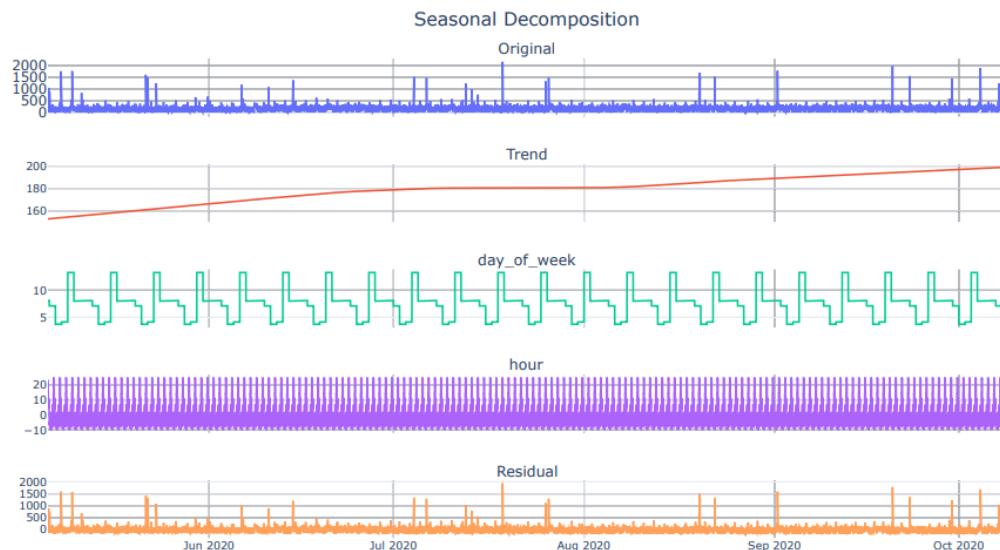


Figure 47 La décomposition de la saisonnalité et de la tendance à l'aide de Loess et des termes de Fourier (Décomposition de Fourier)

Détection des valeurs aberrantes

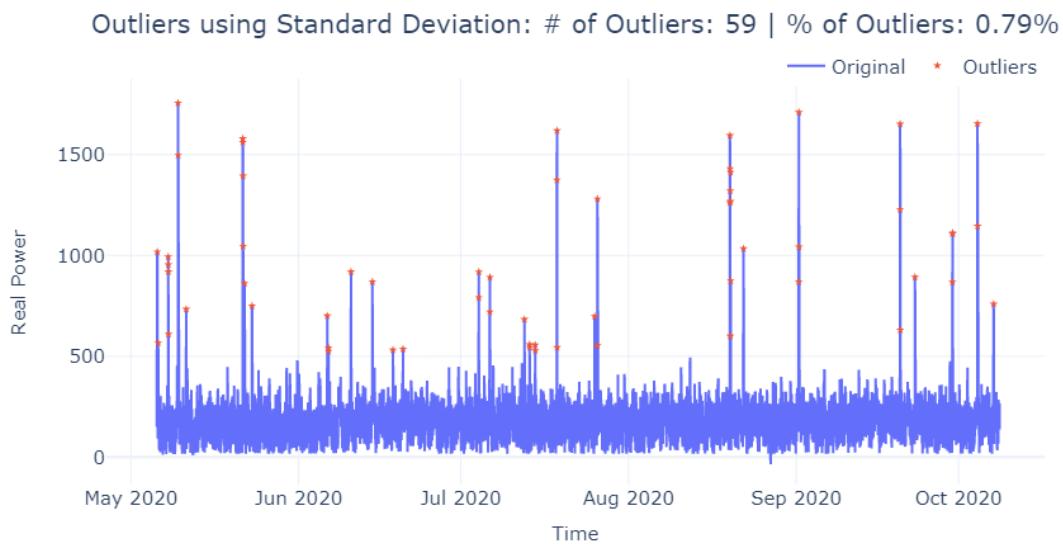
Méthode de l'écart-type

La détection des valeurs aberrantes à l'aide de la méthode de l'écart-type consiste à définir des bornes basées sur la moyenne et l'écart-type d'une série temporelle. Une fonction Python a été développée pour générer un masque booléen qui divise les données en valeurs aberrantes et valeurs incluses dans l'intervalle défini par un multiple de l'écart-type.

```
def detecter_valeurs_aberrantes_ecart_type(ts, multiple_ecart_type=2):
    moyenne = ts.mean()
    ecart_type = ts.std()
    borne_sup = moyenne + multiple_ecart_type * ecart_type
    borne_inf = moyenne - multiple_ecart_type * ecart_type
    masque_valeurs_aberrantes = (ts > borne_sup) | (ts < borne_inf)
    return masque_valeurs_aberrantes
```

Cela permet d'identifier les observations significativement éloignées de la moyenne. Le choix du multiple de l'écart-type est crucial pour ajuster la sensibilité de la détection. Un multiple plus élevé limite la détection aux valeurs extrêmes, tandis qu'un multiple plus bas inclut des valeurs légèrement éloignées. Il est important de sélectionner judicieusement ce multiple en fonction des besoins spécifiques d'identification des valeurs aberrantes dans l'analyse.

Les points rouges correspondent aux observations qui se distinguent considérablement de la tendance générale de la série, indiquant ainsi leur caractère aberrant.



4

Figure 48 Visualisation du résultat de la méthode de l'écart-type

En considérant seulement la partie centrale des données, la méthode de l'écart interquartile fournit une mesure plus robuste de l'étalement qui est moins influencée par les valeurs aberrantes. Nous avons défini une fonction Python pour implémenter cette méthode.

```
def detecter_valeurs_aberrantes_iqr(ts, multiple_iqr=2):
    q1, q2, q3 = np.quantile(ts, 0.25), np.quantile(ts, 0.5), np.quantile(ts, 0.75)
    iqr = q3 - q1
    borne_sup = q3 + multiple_iqr * iqr
    borne_inf = q1 - multiple_iqr * iqr
    masque_valeurs_aberrantes = (ts > borne_sup) | (ts < borne_inf)
    return masque_valeurs_aberrantes
```

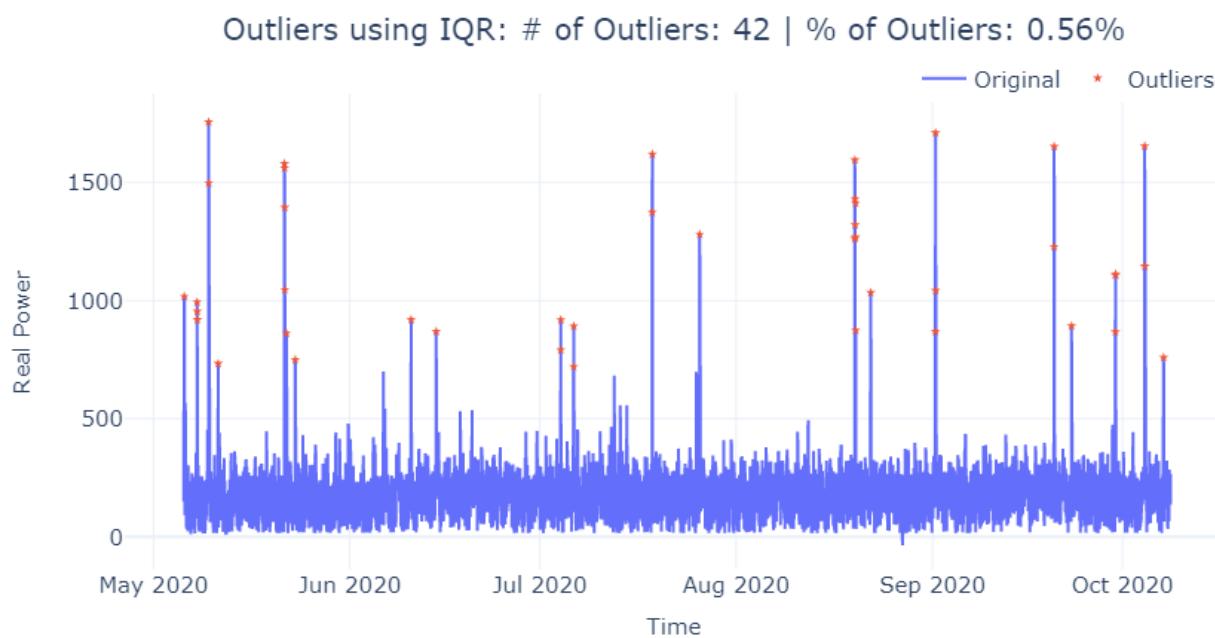


Figure 49 Visualisation du resultat de les l'écart interquartile

ANNEXE C-Mise en place d'un cadre de test :

Pour préparer un cadre de test, il est essentiel de mettre en place des ensembles de données de validation et de test. Le cadre de test permet d'évaluer de manière cohérente et efficace les algorithmes d'apprentissage automatique dans différentes situations.

Préparation préliminaire

Utilisation de la bibliothèque "darts" pour générer des prévisions et sélection d'un local résidentiel.

```
ts_train = train_df.loc[train_df.Premises=="WP6",\ 
["timestamp","real_power"]].set_index("timestamp")
ts_test = val_df.loc[val_df.Premises=="WP6",\ 
["timestamp","real_power"]].set_index("timestamp")
```

Transformation de la série temporelle en une structure de données TimeSeries pour la bibliothèque "darts" :

```
ts_train = TimeSeries.from_series(ts_train)
ts_test = TimeSeries.from_series(ts_test)|
```

Prévision avec la structure de données TimeSeries : Initialisation du modèle, ajustement (.fit) et prédictions (.predict)

```
model = <initialiser le modèle>
model.fit(ts_train)
y_pred = model.predict(len(ts_test))
```

```
mae(actual_series = ts_test, pred_series = y_pred)
mse(actual_series = ts_test, pred_series = y_pred)
#Pour le calcul du MASE, l'ensemble d'entraînement est également nécessaire.
mase(actual_series = ts_test, pred_series = y_pred, insample=ts_train)
forecast_bias(actual_series = ts_test, pred_series = y_pred)|
```

Naïve forecast

Prévision naïve : Utilisation de la classe 'NaiveSeasonal' de la bibliothèque 'darts' pour une prévision basée sur la dernière observation :

```
from darts.models import NaiveSeasonal
naive_model = NaiveSeasonal(K=1)
```

Ce modèle ajuste une ligne entre le premier et le dernier point de la série d'entraînement, et la prolonge dans le futur. Pour une série d'entraînement de longueur T, nous avons :

$$\hat{y}_{T+h} = y_T + h \left(\frac{y_T - y_1}{T - 1} \right)$$

Dans cet exemple, nous observons que la prévision est une ligne droite qui ne tient aucun compte des motifs présents dans la série. Il s'agit de la méthode la plus simple de prévision, d'où son qualificatif de naïve. Passons maintenant à une autre méthode simple.

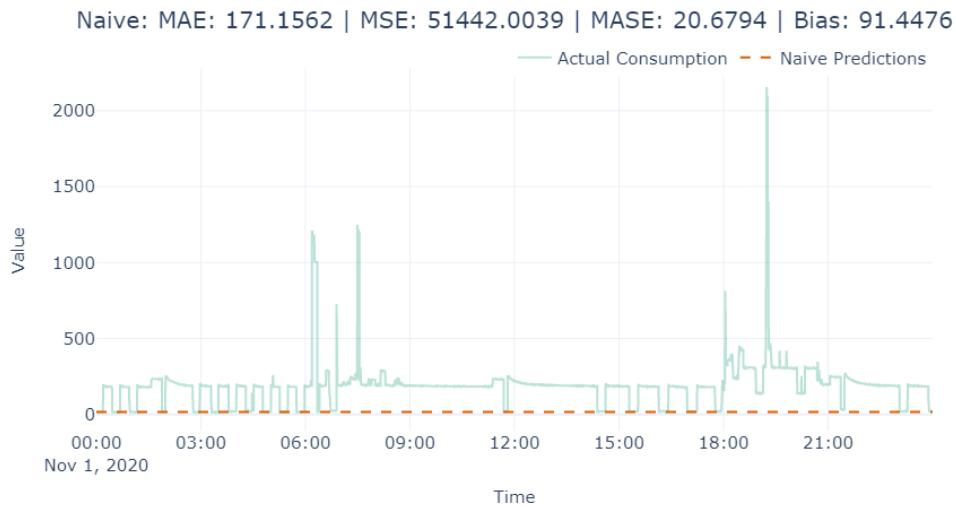


Figure 50 Prevision model Naive

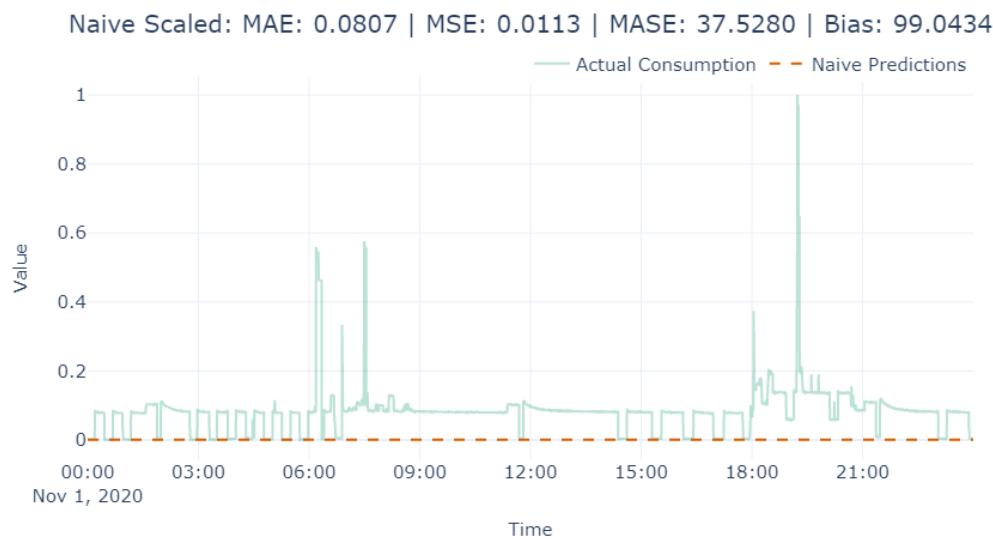


Figure 51 Prévision model Naïve avec transformation de la cible

Prévisions de la moyenne mobile

Le modèle de prévision de la moyenne mobile est une méthode simple qui vise à surmonter la simple mémorisation dans la prévision naïve. Au lieu de se baser uniquement sur la dernière observation, il calcule la moyenne des 'n' dernières observations pour générer la prévision, ce qui permet d'atténuer les fluctuations bruitées présentes dans la série temporelle. Le modèle est ainsi instancié en utilisant la classe NaiveMovingAverage avec une fenêtre dont la longueur est déterminée en fonction de la résolution.

```
naive_model = NaiveMovingAverage(window=6*60*24)
```

Le modèle prédit les valeurs futures en adaptant le comportement d'une droite quasi-linéaire. Ce comportement est normal car dans la décomposition précédemment établie lors de l'analyse des données, la plupart des tendances étaient quasi-stables, en particulier pour les fenêtres de grande taille. Cela pénalise la précision de la prédition en attribuant un poids considérable aux données anciennes, résultant éventuellement en un comportement linéaire, ce modèle ne servira néanmoins qu'en tant que référence, donc sa performance est suffisante.

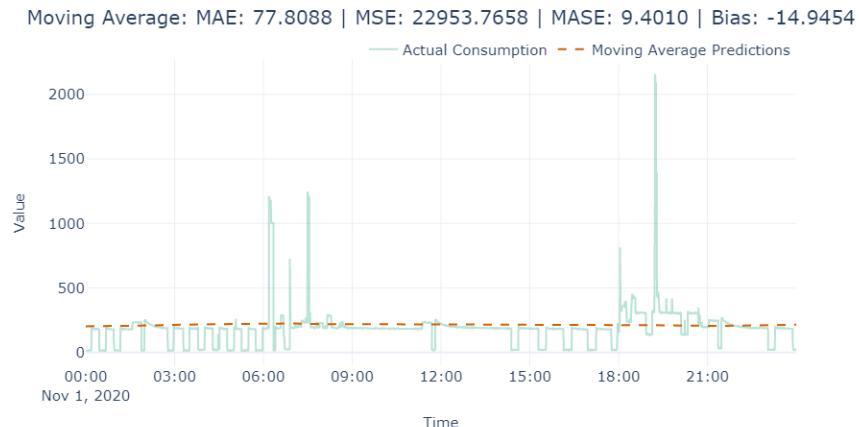


Figure 52 Prévision du modèle de la moyenne mobile

Prévision naïve saisonnière

C'est une imitation de la saisonnalité dans les séries temporelles. La prévision naïve saisonnière est une variation de la méthode naïve classique. Alors que dans la méthode naïve, nous utilisions la dernière observation (Y_{t-1}), dans la version saisonnière naïve, nous utilisons l'observation Y_{t-k} . Ainsi, nous regardons en arrière k étapes pour chaque prévision, ce qui permet à l'algorithme de reproduire le dernier cycle de saisonnalité. Par exemple, si nous fixons $k=6*60*24*7$, nous pourrons imiter le dernier cycle hebdomadaire saisonnier. Cette méthode est implémentée dans darts et peut être utilisée de la manière suivante :

```
from darts.models import NaiveSeasonal
naive_model = NaiveSeasonal(K=6*60*24*7)
```

Dans cet exemple, nous pouvons observer que les prévisions cherchent à imiter le schéma de saisonnalité. Cependant, elles peuvent manquer de précision car elles suivent aveuglément le dernier cycle saisonnier sans prendre en compte d'autres facteurs ou variations dans la série temporelle.

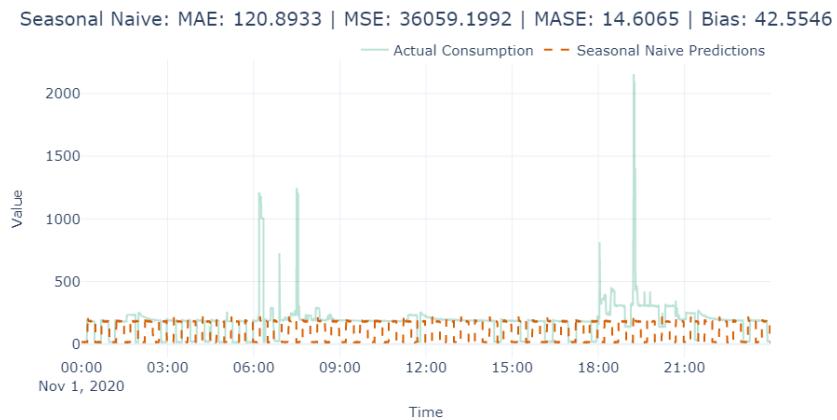


Figure 53 Prévision du modèle naïve saisonnier

Modèle th̄eta

La prévision Theta est une méthode qui amplifie ou lisse la courbure locale d'une série temporelle en utilisant un paramètre appelé θ . Elle implique de vérifier la présence de saisonnalité, de désaisonnaliser la série, d'appliquer une lissage exponentiel simple (SES) sur la série désaisonnalisée, d'ajuster une tendance linéaire, et de réappliquer la saisonnalité si nécessaire. Cette méthode vise à capturer et prévoir les motifs dans la série temporelle en ajustant les paramètres de lissage et de tendance.

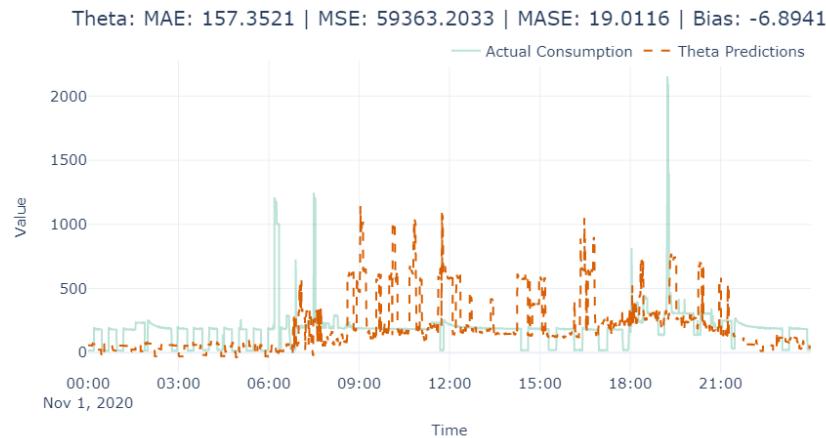


Figure 54 Prévision du modèle theta

Transformée de Fourier rapide (Fast Fourier Transform) :

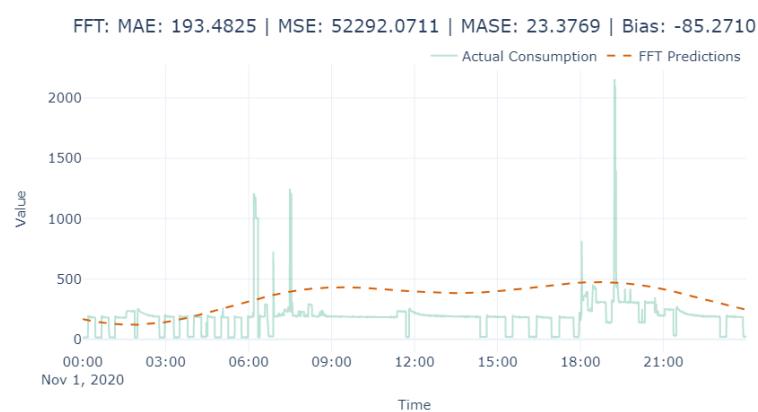


Figure 55 Prévision du modèle FFT

Parmi les cinq modèles de base ou de référence, on constate que le modèle de prévision par moyenne mobile (Moving Average Forecast) présente des erreurs relatives plus élevées par rapport aux autres modèles. Il est important de noter plusieurs considérations auxquelles nous devons répondre.

Tout d'abord, les erreurs MSE sont très grandes car cette mesure est sensible aux valeurs aberrantes, il est donc crucial de choisir une méthode de contrôle des valeurs aberrantes afin d'éviter l'accumulation de grandes erreurs quadratiques qui contribuent à augmenter le MSE, tandis que le MAE n'est pas aussi sensible à ce problème. D'autres considérations concernent le temps d'exécution, qui doit être faible car de tels modèles doivent être accessibles en termes de coût. Cependant, ils souffrent de problèmes de convergence, surtout dans notre cas où les périodes de saisonnalité peuvent aller jusqu'à 6 secondes x 60 minutes x 24 heures x 7 jours, soit 60480 unités de temps, ce qui équivaut à 3 ans et demi dans le cas d'une série temporelle avec une résolution d'une demi-heure.

	Algorithm	MAE	MSE	MASE	Forecast Bias	Time Elapsed
0	Naive	171.156	51442.004	20.679	91.45%	0.097988
1	Moving Average Forecast	77.809	22953.766	9.401	-14.95%	4.530412
2	Seasonal Naive Forecast	120.893	36059.199	14.607	42.55%	0.091006
3	Theta	157.352	59363.203	19.012	-6.89%	3.521240
4	FFT	193.482	52292.071	23.377	-85.27%	0.584709

Validation :

	Algorithm	MAE	MSE	meanMASE	Forecast Bias
0	FFT	138.328	34898.370	1.755	19.90%
1	Theta	149.802	35520.074	2.274	18.59%

Test :

	Algorithm	MAE	MSE	meanMASE	Forecast Bias
0	FFT	147.094	57760.510	14.901	29.41%
1	Theta	142.122	47533.764	1.984	42.39%

ANNEXE D-Introduction à l'apprentissage supervisé

Étant donné un ensemble de points $\{x^{(1)}, \dots, x^{(m)}\}$ associés à un ensemble d'issues $\{y^{(1)}, \dots, y^{(m)}\}$, on veut construire un classifieur qui apprend à prédire y depuis x .

Type de prédition

Les différents types de modèles prédictifs sont résumés dans le tableau ci-dessous :

	Régression	Classifieur	
Issue	Continu	Classe	
Exemples	Régression linéaire	Régression logistique, SVM, Naive Bayes	

Type de modèle :

Les différents modèles sont présentés dans le tableau ci-dessous :

	Modèle discriminant	Modèle génératif
But	Estimer directement $P(y x)$	Estimer $P(x y)$ puis déduire $P(y x)$
Ce qui est appris	Frontière de décision	Distribution de probabilité des données
Illustration		
Exemples	Régressions, SVMs	GDA, Naive Bayes

Notations et concepts généraux

Hypothèse

Une hypothèse est notée h_θ et est le modèle que l'on choisit. Pour une entrée donnée $x^{(i)}$, la prédiction donnée par le modèle est $h_\theta(x^{(i)})$.

Fonction de loss

Une fonction de loss est une fonction $L: (z, y) \in \mathbb{R} \times Y \mapsto L(z, y) \in \mathbb{R}$ prenant comme entrée une valeur prédictive z correspondant à une valeur réelle y , et nous renseigne sur la ressemblance de ces deux valeurs. Les fonctions de loss courantes sont récapitulées dans le tableau ci-dessous :

Erreur des moindres carrés	Loss logistique	Hinge loss	Cross-entropie
$\frac{1}{2}(y - z)^2$	$\log(1 + \exp(-yz))$	$\max(0, 1 - yz)$	$[-y\log(z) + (1 - y)\log(1 - z)]$
Régression linéaire	Régression logistique	SVM	Réseau de neurones

Fonction de coût

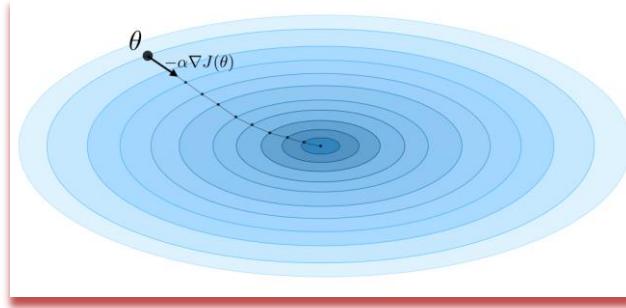
La fonction de coût J est communément utilisée pour évaluer la performance d'un modèle, et est définie avec la fonction de loss L par :

$$J(\theta) = \sum_{i=1}^m L(h_\theta(x^{(i)}), y^{(i)})$$

Algorithme du gradient

En notant $\alpha \in \mathbb{R}$ le taux d'apprentissage (en anglais learning rate), la règle de mise à jour de l'algorithme est exprimée en fonction du taux d'apprentissage et de la fonction de cost J de la manière suivante :

$$\theta \leftarrow \theta - \alpha \nabla J(\theta)$$



Remarque : L'algorithme du gradient stochastique (en anglais Stochastic Gradient Descent ou SGD) met à jour le paramètre à partir de chaque élément du jeu d'entraînement, tandis que l'algorithme du gradient de batch le fait sur chaque lot d'exemples.

Vraisemblance

La vraisemblance d'un modèle $L(\theta)$ de paramètre θ est utilisée pour trouver le paramètre optimal θ par le biais du maximum de vraisemblance. En pratique, on utilise la log vraisemblance $\ell(\theta) = \log(L(\theta))$ qui est plus facile à optimiser. On a :

$$\theta^{opt} = \arg \max_{\theta} L(\theta)$$

Modèles linéaires

Régression linéaire

On suppose ici que $y|x; \theta \sim \mathcal{N}(\mu, \sigma^2)$

Équations normales :

En notant X la matrice de design, la valeur de θ qui minimise la fonction de coût a une solution de forme fermée tel que :

$$\theta = (X^T X)^{-1} X^T y$$

Algorithme LMS

En notant α le taux d'apprentissage, la règle de mise à jour d'algorithme des moindres carrés (LMS) pour un jeu de données d'entraînement de m points, aussi connu sous le nom de règle de Widrow-Hoff, est donné par :

$$\forall j, \quad \theta_j \leftarrow \theta_j + \alpha \sum_{i=1}^m \left[y^{(i)} - h_\theta(x^{(i)}) \right] x_j^{(i)}$$

Régularisation

La procédure de régularisation a pour but d'éviter que le modèle ne surapprenne (en anglais overfit) les données et ainsi vise à régler les problèmes de grande variance. Le tableau suivant récapitule les différentes techniques de régularisation communément utilisées.

LASSO Ridge	Ridge	Elastic Net
<ul style="list-style-type: none"> Réduit les coefficients à 0 Bon pour la sélection de variables 	<ul style="list-style-type: none"> Rend les coefficients plus petits 	<ul style="list-style-type: none"> Compromis entre la sélection de variables et la réduction de coefficients
$\dots + \lambda \ \theta\ _1 \quad \lambda \in \mathbb{R}$	$\dots + \lambda \ \theta\ _2^2 \quad \lambda \in \mathbb{R}$	$\dots + \lambda [(1-\alpha)\ \theta\ _1 + \alpha \ \theta\ _2^2] \quad \lambda \in \mathbb{R}, \alpha \in [0,1]$

Méthode à base d'arbres et d'ensembles

CART

Les arbres de classification et de régression (en anglais CART - Classification And Regression Trees), aussi connus sous le nom d'arbres de décision, peuvent être représentés sous la forme d'arbres binaires. Ils ont l'avantage d'être très interprétables.

Étant donné les vecteurs d'apprentissage $x_i \in \mathbb{R}^l, i = 1, \dots, l^n$ et un vecteur de labels $y \in \mathbb{R}^l$. Un arbre de décision partitionne récursivement l'espace des caractéristiques de sorte que les échantillons

ayant les mêmes labels ou des valeurs cibles similaires soient regroupés.

Les données au niveau du nœud m être représenté par \mathcal{Q}_m avec n_m échantillons. Pour chaque candidat divisé $\theta = (j, t_m)$ composé d'une caractéristique et d'un seuil, partitionner les données en $\mathcal{Q}_m^{gauche}(\theta)$ et $\mathcal{Q}_m^{droite}(\theta)$ sous-ensembles :

$$\begin{aligned}\mathcal{Q}_m^{gauche}(\theta) &= \{(x, y) | x_j \leq t_m\} \\ \mathcal{Q}_m^{droite}(\theta) &= \mathcal{Q}_m \setminus \mathcal{Q}_m^{gauche}(\theta)\end{aligned}$$

La qualité d'un candidat à la division d'un nœud m est ensuite calculée à l'aide d'une fonction d'impureté ou d'une fonction de perte $\mathcal{H}()$, dont le choix dépend de la tâche à résoudre (classification ou régression) :

$$G(\mathcal{Q}_m, \theta) = \frac{n_m^{gauche}}{n_m} \mathcal{H}(\mathcal{Q}_m^{gauche}(\theta)) + \frac{n_m^{droite}}{n_m} \mathcal{H}(\mathcal{Q}_m^{droite}(\theta))$$

Sélectionner les paramètres qui minimisent l'impureté

$$\theta^* = \underset{\theta}{\operatorname{argmin}} G(\mathcal{Q}_m, \theta)$$

Recherche des sous-ensembles $\mathcal{Q}_m^{gauche}(\theta)$ et $\mathcal{Q}_m^{droite}(\theta)$ jusqu'à ce que la profondeur maximale autorisée soit atteinte, $n_m < \min_{\text{échantillons}}$ ou $n_m=1$.

Critère de régression

Si la cible est une valeur continue, alors pour le nœud m, les critères courants à minimiser pour déterminer les emplacements des futures divisions sont l'erreur quadratique moyenne (EQM ou erreur L2), la déviance de Poisson ainsi que l'erreur absolue moyenne (MAE ou erreur L1).

L'EQM et la déviance de Poisson fixent toutes deux la valeur prédictive des nœuds terminaux à la valeur moyenne apprise \bar{y}_m du nœud, tandis que l'EAM fixe la valeur prédictive des nœuds terminaux à la médiane $mediane(y_m)$.

Erreur quadratique moyenne :

$$\begin{aligned}\bar{y} &= \frac{1}{n_m} \sum_{y \in \mathcal{Q}_m} y \\ H(\mathcal{Q}_m) &= \frac{1}{n_m} \sum_{y \in \mathcal{Q}_m} (y - \bar{y}_m)^2\end{aligned}$$

Demi-déviance de Poisson :

$$H(\mathcal{Q}_m) = \frac{1}{n_m} \sum_{y \in \mathcal{Q}_m} \left(y \log\left(\frac{y}{\bar{y}_m}\right) - y - \bar{y}_m \right)^2$$

Le critère criterion="poisson" peut être un bon choix si la cible est un nombre ou une fréquence (nombre par unité). Quoi qu'il en soit, $y \geq 0$ est une condition nécessaire pour utiliser ce critère. Plus il s'adapte beaucoup plus lentement que le critère MSE

Erreur absolue moyenne :

$$\text{median}(y_m) = \text{mediane}_{y \in \mathcal{Q}_m}(y)$$
$$H(\mathcal{Q}_m) = \frac{1}{n_m} \sum_{y \in \mathcal{Q}_m} |y - \text{median}(y_m)|$$

Remarque : Il s'adapte beaucoup plus lentement que le critère MSE.

Forêts aléatoires

C'est une technique à base d'arbres qui utilise un très grand nombre d'arbres de décisions construits à partir d'ensembles de caractéristiques aléatoirement sélectionnés. Contrairement à un simple arbre de décision, il n'est pas interprétable du tout mais le fait qu'il ait une bonne performance en fait un algorithme populaire.

Remarque : les forêts aléatoires sont un type de méthode ensembliste.

Boosting :

L'idée des méthodes de boosting est de combiner plusieurs modèles faibles pour former un modèle meilleur. Les principales méthodes de boosting sont récapitulées dans le tableau ci-dessous :

Boosting adaptif	Boosting par gradient
<ul style="list-style-type: none">Connu sous le nom d'AdaboostDe grands coefficients sont mis sur les erreurs pour s'améliorer à la prochaine étape de boosting	<ul style="list-style-type: none">Les modèles faibles sont entraînés sur les erreurs résiduelles

Annexe E- Ingénierie des caractéristiques (Feature Engineering)

Dans la section annexe de cette présentation, nous aborderons les techniques d'ingénierie des caractéristiques utilisées pour enrichir notre analyse des séries temporelles. Ces techniques comprennent les caractéristiques de retard, les caractéristiques de roulement, les caractéristiques de roulement saisonnières, l'EWMA, les caractéristiques temporelles et les termes de Fourier. La base de données augmentée qui en résulte contient un large éventail de colonnes, y compris les variables originales et les caractéristiques. Ces caractéristiques améliorent collectivement notre compréhension des données et fournissent des données précieuses pour la modélisation prédictive et l'analyse perspicace.

Caractéristiques de Lag :

```
#Cette ligne de code importe la fonction "add_lags" du module "autoregressive_features" du paquet "src.feature_engineering".
from src.feature_engineering.autoregressive_features import add_lags
```

```
#La variable "lags" est une liste qui combine trois ensembles de valeurs de décalage
```

```
lags = (
    (np.arange(5) + 1).tolist()
    + (np.arange(5) + 6*60).tolist()
    + (np.arange(5) + (6*60 * 24*4) - 2).tolist()
)
lags
```

```
[1, 2, 3, 4, 5, 360, 361, 362, 363, 364, 34558, 34559, 34560, 34561, 34562]
```

```
#En utilisant la fonction "add_lags" avec les valeurs de retard spécifiées et d'autres paramètres, la base de données "full_df" est
```

```
with LogTime():
    full_df, added_features = add_lags(
        full_df, lags=lags, column="real_power", ts_id="Premises", use_32_bit=True
    )
print(f"Features Created: {','.join(added_features)}")
```

```
Time Elapsed: 2 seconds
Features Created: real_power_lag_1,real_power_lag_2,real_power_lag_3,real_power_lag_4,real_power_lag_5,real_power_lag_360,real_
power_lag_361,real_power_lag_362,real_power_lag_363,real_power_lag_364,real_power_lag_34558,real_power_lag_34559,real_power_lag
_34560,real_power_lag_34561,real_power_lag_34562
```

Caractéristiques temporelles :

```
#Cette ligne de code importe la fonction "add_temporal_features" du module "temporal_features" du paquet
#"src.feature_engineering".
from src.feature_engineering.temporal_features import add_temporal_features
```

```
with LogTime():
    full_df, added_features = add_temporal_features(
        full_df,
        field_name="timestamp",
        frequency="10T",
        add_elapsed=True,
        drop=False,
        use_32_bit=True,
    )
print(f"Features Created: {','.join(added_features)}")
```

```
Time Elapsed: 4 seconds
Features Created: timestamp_Month,timestamp_Quarter,timestamp_Is_quarter_end,timestamp_Is_quarter_start,timestamp_Is_year_end,t
imestamp_Is_year_start,timestamp_Is_month_start,timestamp_WeekDay,timestamp_Dayofweek,timestamp_Dayofyear,timestamp_Hour,timest
amp_Minute,timestamp_Elapsed
```

Résultats :

L'index suivant représente le résultat de l'augmentation des données grâce à la création de nouvelles caractéristiques. Il comprend une large gamme de fonctionnalités qui ont été ajoutées à l'ensemble de données d'origine, telles que des caractéristiques de décalage (par exemple, 'real_power_lag_1', 'real_power_lag_2'), des caractéristiques de défilement (par exemple, 'real_power_rolling_10_mean', 'real_power_rolling_600_std'), des caractéristiques de défilement saisonnier (par exemple, 'real_power_8640_seasonal_rolling_3_mean', 'real_power_60480_seasonal_rolling_3_std'), des moyennes mobiles exponentielles pondérées (par exemple, 'real_power_ewma_span_60480'), et diverses caractéristiques temporelles dérivées de l'horodatage (par exemple, 'timestamp_Month', 'timestamp_Hour_cos_4'). Ces fonctionnalités augmentées fournissent des informations précieuses et améliorent les capacités prédictives pour une analyse plus approfondie.

```
full_df.columns
Index(['timestamp', 'Premises', 'real_power', 'Vrms', 'debut_ts',
       'sampling_rate', 'ts_len', 'ownership', 'building_type',
       'affluent_neighborhood_apartment',
       'affluent_neighborhood_semi_detached_house',
       'disadvantaged_neighborhood_apartment', 'city_of_residence',
       'number_of_occupants', 'number_of_elderly', 'number_of_adults',
       'number_of_rooms', 'number_of_floors', 'area',
       'total_acquisition_duration', 'mean_daily_consumption',
       'MORED_category', 'type', 'real_power_lag_1', 'real_power_lag_2',
       'real_power_lag_3', 'real_power_lag_4', 'real_power_lag_5',
       'real_power_lag_360', 'real_power_lag_361', 'real_power_lag_362',
       'real_power_lag_363', 'real_power_lag_364', 'real_power_lag_34558',
       'real_power_lag_34559', 'real_power_lag_34560', 'real_power_lag_34561',
       'real_power_lag_34562', 'real_power_rolling_10_mean',
       'real_power_rolling_10_std', 'real_power_rolling_600_mean',
       'real_power_rolling_600_std', 'real_power_rolling_14400_mean',
       'real_power_rolling_14400_std', 'real_power_rolling_100000_mean',
       'real_power_rolling_100000_std',
       'real_power_8640_seasonal_rolling_3_mean',
       'real_power_8640_seasonal_rolling_3_std',
       'real_power_8640_seasonal_rolling_3_mean',
       'real_power_60480_seasonal_rolling_3_std', 'real_power_ewma_span_60480',
       'real_power_ewma_span_120960', 'real_power_ewma_span_8640',
       'timestamp_Month', 'timestamp_Quarter', 'timestamp_Is_quarter_end',
       'timestamp_Is_quarter_start', 'timestamp_Is_year_end',
       'timestamp_Is_year_start', 'timestamp_Is_month_start',
       'timestamp_Weekday', 'timestamp_Dayofweek', 'timestamp_Dayofyear',
       'timestamp_Hour', 'timestamp_Minute', 'timestamp_Elapsed',
       'timestamp_Month_sin_1', 'timestamp_Month_sin_2',
       'timestamp_Month_sin_3', 'timestamp_Month_sin_4',
       'timestamp_Month_sin_5', 'timestamp_Month_cos_1',
       'timestamp_Month_cos_2', 'timestamp_Month_cos_3',
       'timestamp_Month_cos_4', 'timestamp_Month_cos_5',
       'timestamp_Hour_sin_1', 'timestamp_Hour_sin_2', 'timestamp_Hour_sin_3',
       'timestamp_Hour_sin_4', 'timestamp_Hour_sin_5', 'timestamp_Hour_cos_1',
       'timestamp_Hour_cos_2', 'timestamp_Hour_cos_3', 'timestamp_Hour_cos_4',
       'timestamp_Hour_cos_5', 'timestamp_Minute_sin_1',
       'timestamp_Minute_sin_2', 'timestamp_Minute_sin_3',
       'timestamp_Minute_sin_4', 'timestamp_Minute_sin_5',
       'timestamp_Minute_cos_1', 'timestamp_Minute_cos_2',
       'timestamp_Minute_cos_3', 'timestamp_Minute_cos_4',
       'timestamp_Minute_cos_5'],
      dtype='object')
```

Une représentation graphique des caractéristiques temporelles a été réalisée en utilisant les données du dataframe "plot_df". Ce dataframe a été créé en sélectionnant les colonnes "timestamp_Month" et "timestamp_Month_sin_1", en supprimant les doublons et en les triant par mois. Les colonnes ont ensuite été renommées, la dataframe a été réindexée et les colonnes fusionnées pour obtenir une visualisation des données.

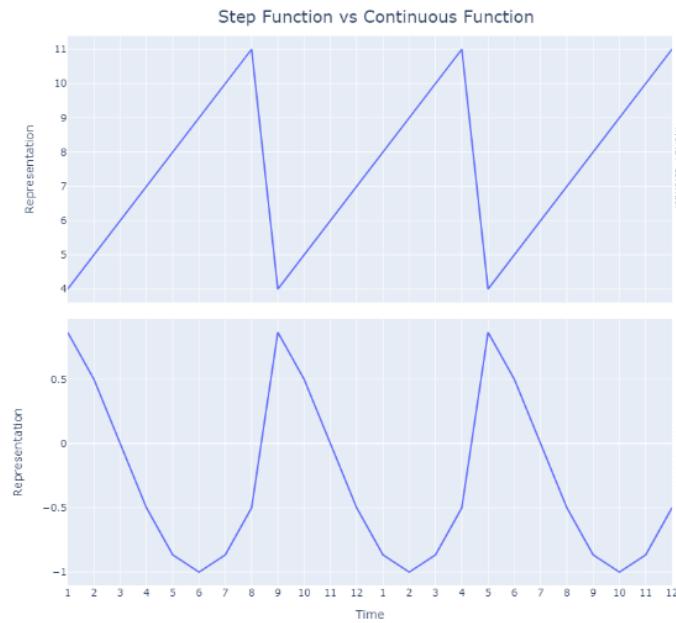


Figure 56 Visualisation de la représentation des caractéristiques temporelle

Annexe F- Prévision avec les méthodes d'apprentissage automatique :

Dans ce code, nous essayons de lire deux fichiers au format parquet, "selected_train_missing_imputed_feature_engg.parquet" pour l'ensemble d'entraînement et "selected_val_missing_imputed_feature_engg.parquet" pour l'ensemble de test.

Dans le contexte de la configuration des caractéristiques et de la variable cible pour la prévision des séries temporelles, il est important de prendre en compte les fonctionnalités spécifiques des modèles de prévision utilisés, tels que ceux disponibles dans la bibliothèque scikit-learn de Python.

Pour les caractéristiques de la série temporelle, il peut s'agir de variables continues, telles que les valeurs précédentes de la série (retards), les statistiques de roulement (moyennes mobiles, écarts types, etc.), les caractéristiques de saisonnalité (comme les indicateurs de jour de la semaine ou d'heure du jour) ou d'autres caractéristiques pertinentes pour le problème spécifique.

La variable cible est la valeur future que nous souhaitons prédire, et elle est généralement une variable continue dans le contexte de la prévision des séries temporelles. Par exemple, il peut s'agir de la consommation d'électricité prévue pour le lendemain ou du prix attendu d'une action.

```
try:  
    train_df = pd.read_parquet(preprocessed/"selected_train_missing_imputed_feature_engg.parquet")  
  
    test_df = pd.read_parquet(preprocessed/"selected_val_missing_imputed_feature_engg.parquet")  
  
except FileNotFoundError:  
    display(HTML("""  
        <div class="alert alert-block alert-warning">  
            <b>Warning!</b> File not found.  
        </div>  
    """))
```

Configuration des caractéristiques et de la variable cible

Pour les caractéristiques de la série temporelle, il peut s'agir de variables continues, telles que les valeurs précédentes de la série (retards), les statistiques de roulement (moyennes mobiles, écarts types, etc.), les caractéristiques de saisonnalité (comme les indicateurs de jour de la semaine ou d'heure du jour) ou d'autres caractéristiques pertinentes pour le problème spécifique.

La variable cible est la valeur future que nous souhaitons prédire, et elle est généralement une variable continue dans le contexte de la prévision des séries temporelles. Par exemple, il peut s'agir de la consommation d'électricité prévue pour le lendemain ou du prix attendu d'une action.

```
feat_config = FeatureConfig(  
    date="timestamp",  
    target="real_power",  
    continuous_features=continuous_features_,  
    #creation des caractéristiques catégorielles  
    categorical_features=[  
        'ownership',  
        'building_type',  
        'city_of_residence',  
        "timestamp_Month",  
        "timestamp_Quarter",  
        "timestamp_WeekDay",  
        "timestamp_Dayofweek",  
        "timestamp_Dayofyear",  
        "timestamp_Hour",  
        "timestamp_Minute",  
    ],  
    #creation des caractéristiques booléennes  
    boolean_features=[  
        "affluent_neighborhood_apartment",  
        "affluent_neighborhood_semi_detached_house",  
        "disadvantaged_neighborhood_apartment",  
        "timestamp_Is_quarter_end",  
        "timestamp_Is_quarter_start",  
        "timestamp_Is_year_end",  
        "timestamp_Is_year_start",  
        "timestamp_Is_month_start",  
    ],  
    index_cols=["timestamp"],  
    # Les composantes de la série multivariée ne présentent aucune variable exogène (par exemple la température...)  
    exogenous_features=[],  
)
```

Sélection d'un échantillon résidentiel du dataset

En utilisant la bibliothèque pandas de Python, on peut sélectionner un échantillon résidentiel à partir d'un dataset en filtrant les données selon des critères spécifiques. Pour diviser l'échantillon en ensembles d'entraînement et de test, on peut réserver une partie pour la validation initiale (val) et concaténer val avec l'ensemble d'entraînement (train) pour former l'ensemble de test. Cela permet d'évaluer les performances du modèle sur les données de validation et de tester le modèle final sur un ensemble de test qui reflète les conditions réelles.

```
# sélectionner la maison en utilisant son id (par exemple 'WPILGT2' fait référence à la maison identifiée par WPILGT2)  
sample_train_df = train_df.loc[train_df.Premises == "WPILGT2", :]  
sample_test_df = test_df.loc[test_df.Premises == "WPILGT2", :]  
train_features, train_target, train_original_target = feat_config.get_X_y(  
    sample_train_df, categorical=False, exogenous=False  
)  
# Chargement de la validation en tant que test  
test_features, test_target, test_original_target = feat_config.get_X_y(  
    sample_test_df, categorical=False, exogenous=False  
)  
del sample_train_df, sample_test_df
```

Modèle Linéaire :

▪ Régression linéaire simple

La régression linéaire simple est effectuée en utilisant la classe **LinearRegression()** de **scikit-learn** en Python. Le modèle est configuré en spécifiant la variable indépendante (X) et la variable dépendante (y). Une fois le modèle configuré, il est entraîné en ajustant les coefficients de régression afin de minimiser l'erreur entre les valeurs prédites et les valeurs réelles. La variable "**model_config**" permet de spécifier des paramètres supplémentaires, tels que la normalisation des données ou des

termes de régularisation. L'importance des caractéristiques peut être analysée en examinant les coefficients de régression. Le choix entre MSE, MAE ou MASE comme métrique d'évaluation dépend de la sensibilité souhaitée aux valeurs aberrantes, MAE étant moins sensible et offrant une option robuste.

```
# Configuration du modèle spécifiant une régression linéaire avec un nom
model_config = ModelConfig(
    model=LinearRegression(),
    name="Régression linéaire",
    # La régression linéaire est sensible aux données normalisées
    normalize=True,
    # La régression linéaire ne peut pas gérer les valeurs manquantes
    fill_missing=True,
)

# Mesurer le temps d'exécution à l'aide du gestionnaire de contexte LogTime
with LogTime() as timer:
    # Évaluer le modèle en utilisant les configurations et les données spécifiées
    y_pred, metrics, feat_df = evaluate_model(
        model_config,
        feat_config,
        missing_value_config,
        train_features,
        train_target,
        test_features,
        test_target,
    )

    # Enregistrer le temps écoulé dans le dictionnaire de métriques
    metrics["Temps écoulé"] = timer.elapsed

    # Ajouter les métriques à la liste metric_record
    metric_record.append(metrics)

    # Joindre les valeurs prédites au dataframe pred_df
    pred_df = pred_df.join(y_pred)|
```

La régression Ridge est réalisée en utilisant la classe **RidgeCV()** de scikit-learn en Python. Ce modèle inclut un terme de régularisation L2 qui permet de réduire le surajustement en ajoutant une pénalité sur les coefficients de régression. La classe **RidgeCV()** effectue une validation croisée pour sélectionner automatiquement le meilleur paramètre de régularisation (alpha) parmi une liste prédéfinie.

Cela permet d'ajuster le modèle de régression Ridge de manière optimale en trouvant le bon équilibre entre l'ajustement aux données et la réduction du surajustement.

```

# Configuration du modèle Ridge Regression
model_config = ModelConfig(
    model=RidgeCV(),
    name="Régession Ridge",
    # RidgeCV est sensible aux données normalisées
    normalize=True,
    # RidgeCV ne gère pas les valeurs manquantes
    fill_missing=True
)

# Évaluation du modèle avec chronométrage
with LogTime() as timer:
    y_pred, metrics, feat_df = evaluate_model(
        model_config,
        feat_config,
        missing_value_config,
        train_features,
        train_target,
        test_features,
        test_target,
    )

# Enregistrement du temps écoulé
metrics["Temps écoulé"] = timer.elapsed
metric_record.append(metrics)
pred_df = pred_df.join(y_pred)

```

Graphic des caractéristiques avec leur importance :

La régression Ridge est réalisée en utilisant la classe RidgeCV() de scikit-learn en Python. Ce modèle inclut un terme de régularisation L2 qui permet de réduire le surajustement en ajoutant une pénalité sur les coefficients de régression. La classe RidgeCV() effectue une validation croisée pour sélectionner automatiquement le meilleur paramètre de régularisation (alpha) parmi une liste prédéfinie. Cela permet d'ajuster le modèle de régression Ridge de manière optimale en trouvant le bon équilibre entre l'ajustement aux données et la réduction du surajustement.



Figure 57 Graphic des caractéristiques avec leur importance (régression simple)

La régression Lasso est réalisée en utilisant la classe Lasso() de scikit-learn en Python. Contrairement à la régression Ridge, la régression Lasso utilise une pénalité L1 qui a la particularité d'effectuer une sélection de variables en mettant certains coefficients de régression à zéro. Cela permet d'obtenir un modèle parcimonieux en ne conservant que les caractéristiques les plus importantes. Comme pour la régression Ridge, la classe Lasso() peut également être utilisée avec un paramètre de régularisation sélectionné automatiquement en utilisant la validation croisée, à l'aide de la classe LassoCV().

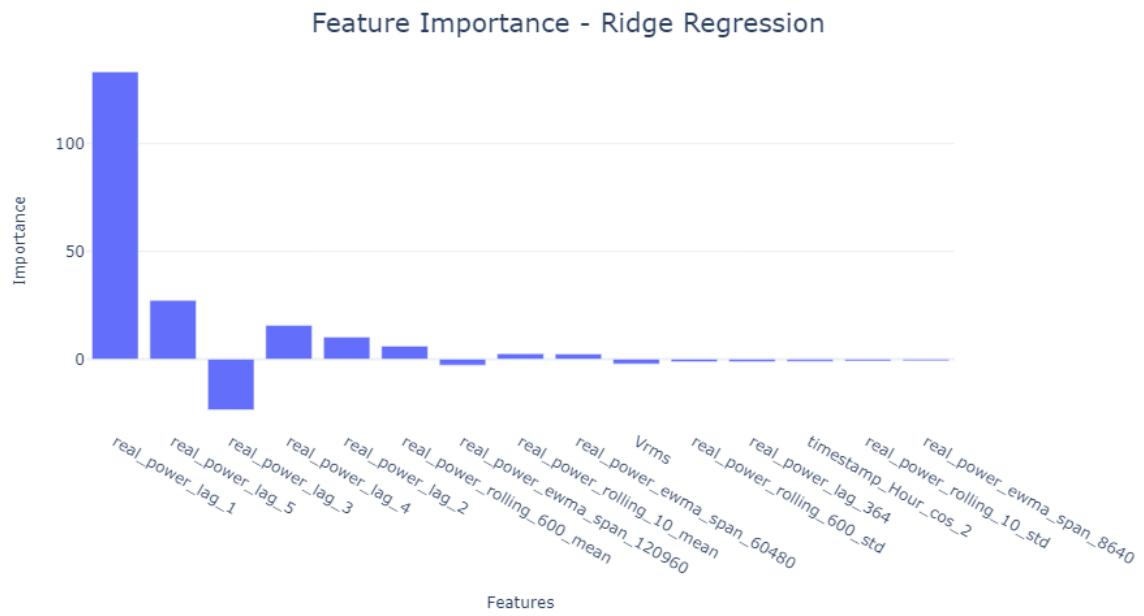


Figure 58 Graphic des caractéristiques avec leur importance (régression régularisée en norme L2)

Les métriques d'évaluation utilisées dans notre étude, telles que l'erreur absolue moyenne (MAE), l'erreur absolue moyenne standardisée (MASE) et le biais, ont démontré de bonnes performances. Cependant, l'erreur quadratique moyenne (MSE) est élevée en raison de la présence de nombreux points aberrants dans les données. Cette problématique pourrait être résolue en mettant en œuvre des techniques de traitement des valeurs aberrantes discutées précédemment dans la section annexe.

Pour améliorer la performance du modèle, nous pourrions envisager une transformation des données visant à réduire l'impact des valeurs aberrantes sur le modèle de régression. Une fois cette transformation appliquée, nous pourrions obtenir des résultats plus satisfaisants, similaires à ceux discutés dans le chapitre 4 du rapport principal.

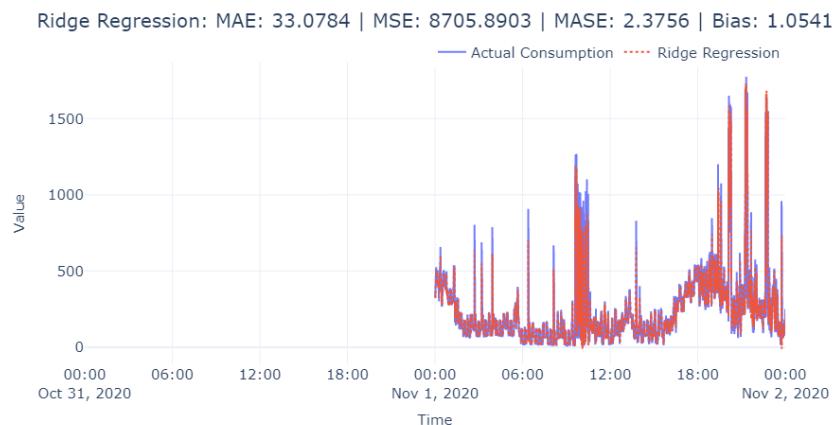


Figure 59 Résultat de la prédiction comparé au valeurs réelle (régression Ridge)

Il est donc essentiel d'accorder une attention particulière au traitement des valeurs aberrantes afin de réduire leur influence négative sur les performances du modèle de régression. En combinant des techniques de traitement des valeurs aberrantes et une transformation appropriée des données, nous pourrons obtenir des résultats plus fiables et plus robustes pour notre modèle de prévision.

Malgré la présence de valeurs aberrantes qui ont entraîné une augmentation significative de l'erreur quadratique moyenne (MSE), les performances des autres métriques d'évaluation, telles que l'erreur absolue moyenne (MAE) et l'erreur absolue moyenne standardisée (MASE), demeurent satisfaisantes. Il convient de noter que ces valeurs aberrantes sont plus susceptibles d'impacter le modèle en raison de la résolution temporelle élevée de nos données, avec un point toutes les 10 secondes. Une seule journée de données comprend déjà $6 * 60 * 24$ points, ce qui augmente le risque d'erreurs liées aux fluctuations et aux valeurs aberrantes.



Figure 60 Graphic des caractéristiques avec leur importance (régression régularisée en norme L1)

Dans le modèle Lasso, nous avons observé un comportement intéressant en termes d'importance des caractéristiques. La caractéristique de retard 1 a été initialement attribuée une grande importance, indiquant son fort impact sur la prédiction. Cependant, à mesure que nous augmentons le retard, l'importance diminue progressivement. Par exemple, la caractéristique de retard 5 a obtenu environ 25% de l'importance de la caractéristique de retard 1.

Ce phénomène suggère que le modèle Lasso privilégie les retards plus récents, considérant leur contribution plus significative à la prédiction. Ainsi, il accorde une plus grande importance aux retards observés récemment, tandis que l'importance diminue progressivement pour les retards plus éloignés dans le temps. Cette tendance peut être due au fait que les informations récentes sont souvent plus pertinentes pour anticiper les futurs changements.

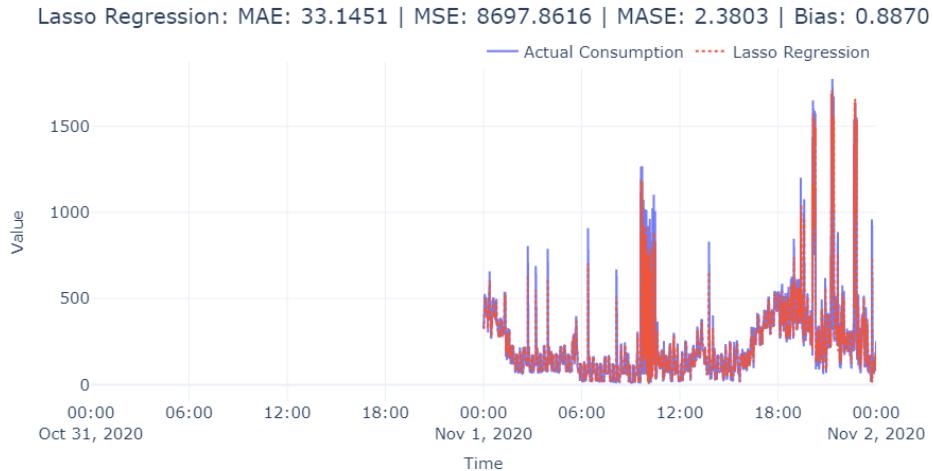


Figure 61 Résultat de la prédiction comparé au valeurs réelle (régression Lasso)

▪ Les Arbres de décision

Pour instancier le modèle, nous avons créé une instance de la classe DecisionTreeRegressor() en spécifiant les paramètres souhaités. Par exemple, nous pouvons définir la profondeur maximale de l'arbre en utilisant le paramètre "max_depth" et le nombre minimum d'échantillons requis pour diviser un nœud en utilisant le paramètre "min_samples_split".

```
# Importation de la classe DecisionTreeRegressor depuis le module sklearn.tree
from sklearn.tree import DecisionTreeRegressor
```

model=DecisionTreeRegressor(max_depth=4, random_state=42): Nous instancions un modèle **DecisionTreeRegressor** avec une profondeur maximale de 4 et une graine aléatoire de 42.

name="Arbre de Décision": Nous attribuons le nom "Arbre de Décision" au modèle.

normalize=False: Nous spécifions que la normalisation des données n'est pas nécessaire pour l'arbre de décision, car il n'est pas affecté par la normalisation.

fill_missing=True: Nous indiquons que le modèle d'arbre de décision ne gère pas les valeurs manquantes, donc nous devons spécifier comment traiter ces valeurs manquantes.

```

# Configuration du modèle de régression par arbre de décision
model_config = ModelConfig(
    model=DecisionTreeRegressor(max_depth=4, random_state=42),
    name="Arbre de Décision",
    # L'arbre de décision n'est pas affecté par la normalisation
    normalize=False,
    # Scikit-Learn ne gère pas les valeurs manquantes dans l'arbre de décision
    fill_missing=True,
)

# Évaluation du modèle
with LogTime() as timer:
    y_pred, metrics, feat_df = evaluate_model(
        model_config,
        feat_config,
        missing_value_config,
        train_features,
        train_target,
        test_features,
        test_target,
    )
    metrics["Temps éoulé"] = timer.elapsed
    metric_record.append(metrics)
    pred_df = pred_df.join(y_pred)

```

Une fois le modèle instancié, nous pouvons l'ajuster aux données d'entraînement en utilisant la méthode "fit()" en fournissant les caractéristiques d'entraînement et les valeurs cibles correspondantes. Une fois le modèle ajusté, nous pouvons l'utiliser pour effectuer des prédictions sur de nouvelles données en utilisant la méthode "predict()".

Le modèle de régression par arbre de décision se divise en fonction des caractéristiques qui apportent le plus de gain d'information pour réduire l'erreur de prédiction, mesurée par l'entropie ou l'impureté de Gini. Les caractéristiques les plus discriminantes, réduisant le désordre ou l'incertitude dans les données, sont considérées comme les plus importantes. Ainsi, l'arbre de décision construit des divisions efficaces basées sur ces caractéristiques pour améliorer la précision des prédictions.

Le graphique d'importance des caractéristiques indique que la caractéristique "lag 1" a une importance nettement plus élevée que les autres caractéristiques. Cela suggère que "lag 1" joue un rôle crucial dans la prise de décision du modèle d'arbre de décision. Les autres caractéristiques ont une importance relativement plus faible, ce qui implique qu'elles ont moins d'impact sur les prédictions du modèle.

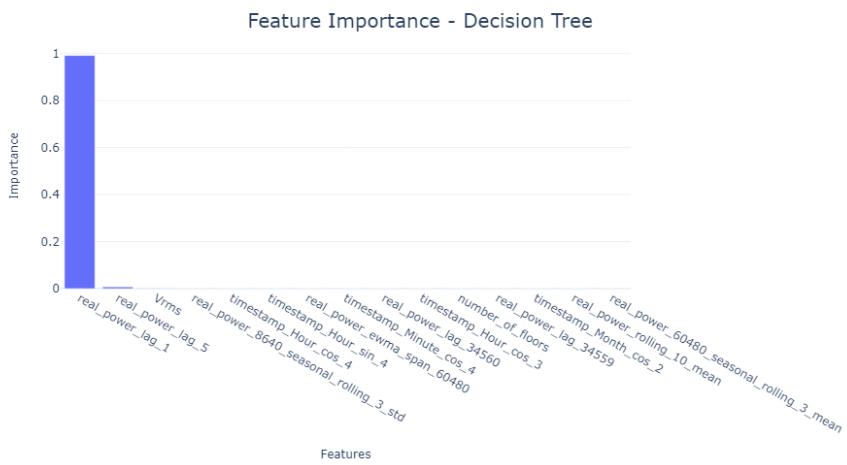


Figure 62 Graphic des caractéristiques avec leur importance (arbre de décision)

Cependant, il est important de noter que les arbres de décision sont connus pour être sensibles aux valeurs aberrantes. Les valeurs aberrantes peuvent grandement influencer les décisions de division prises par l'arbre, ce qui peut conduire à des prédictions moins précises. En conséquence, la métrique de l'Erreur Quadratique Moyenne (MSE) peut être considérablement affectée par la présence de valeurs aberrantes, ce qui entraîne une erreur plus importante.

La longueur de la période à prédire peut également influencer les performances du modèle. Lorsqu'il s'agit de prévoir une période de temps plus longue, il est possible que le modèle rencontre des difficultés pour capturer les tendances et les variations sur cette période étendue.

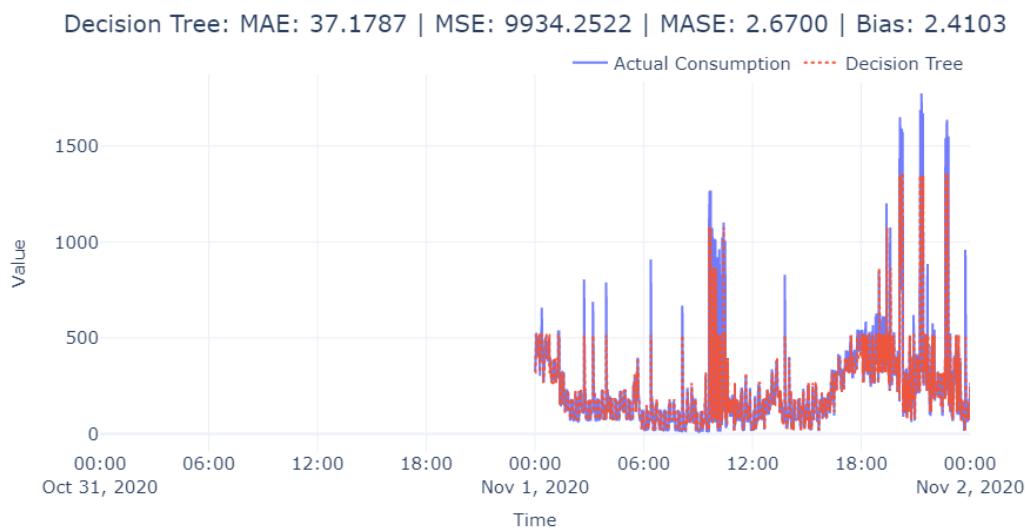


Figure 63 Résultat de la prédiction comparé au valeurs réelle (arbre de décision)

▪ Les Forest Aléatoire

Le constructeur "**RandomForestRegressor**" de la classe "**RandomForestRegressor**" de scikit-learn est utilisé pour initialiser un modèle Random Forest. Les paramètres spécifiés lors de l'initialisation comprennent le nombre d'arbres dans l'ensemble, la profondeur maximale des arbres, les critères de division (tels que l'indice Gini ou l'entropie), et bien d'autres encore. L'agrégation ou l'ensembling en anglais, une technique utilisée par Random Forest, consiste à combiner les prédictions de plusieurs arbres de décision individuels pour obtenir une prédition finale plus robuste et précise. Cela permet de réduire le surajustement et d'améliorer la performance prédictive du modèle. En somme, l'initialisation d'un modèle Random Forest implique de spécifier les paramètres et d'utiliser l'ensembling pour combiner les prédictions des arbres dans l'ensemble.

Dans une Forest Aléatoire, l'effet de pooling (vote) est obtenu en construisant un ensemble d'arbres de décision et en agrégant leurs prédictions. Chaque arbre de décision est construit sur un sous-ensemble aléatoire des données d'entraînement et un sous-ensemble aléatoire des caractéristiques, ce qui donne un ensemble de modèles diversifiés. La prédition finale est obtenue en moyennant ou en votant les prédictions des arbres individuels. Cet effet de pooling contribue à réduire le surajustement et à améliorer la généralisation en capturant les connaissances et les informations collectives de plusieurs arbres. Il améliore la stabilité et la robustesse du

modèle, ce qui permet d'obtenir des prédictions plus fiables et de meilleures performances globales.

```
#Cette ligne importe la classe RandomForestRegressor du module sklearn.ensemble.  
#La classe RandomForestRegressor est utilisée pour construire un modèle de régression  
#par forêt aléatoire.  
from sklearn.ensemble import RandomForestRegressor  
  
model_config = ModelConfig(  
    model=RandomForestRegressor(random_state=42, max_depth=4),  
    name="Random Forest",  
    # RandomForest n'est pas affecté par la normalisation  
    normalize=False,  
    # RandomForest dans scikit-learn ne gère pas les valeurs manquantes  
    fill_missing=True,  
)  
with LogTime() as timer:  
    y_pred, metrics, feat_df = evaluate_model(  
        model_config,  
        feat_config,  
        missing_value_config,  
        train_features,  
        train_target,  
        test_features,  
        test_target,  
    )  
metrics["Time Elapsed"] = timer.elapsed  
metric_record.append(metrics)  
pred_df = pred_df.join(y_pred)
```

L'importance de "Lag1" indique que l'historique récent de la série a une influence significative sur les prédictions. La profondeur maximale (max_depth) d'un arbre de décision affecte la manière dont les caractéristiques sont utilisées pour prendre des décisions de division. Plus la profondeur maximale est élevée, plus l'arbre peut apprendre des relations complexes entre les caractéristiques et la variable cible. Cela peut entraîner une utilisation plus équilibrée des caractéristiques et une répartition plus uniforme de l'importance entre les différentes caractéristiques. En revanche, avec une profondeur maximale plus faible, l'arbre se concentrera davantage sur les caractéristiques les plus discriminantes pour les premières décisions de division, ce qui peut entraîner une importance plus élevée pour ces caractéristiques et une importance relativement faible pour les autres.

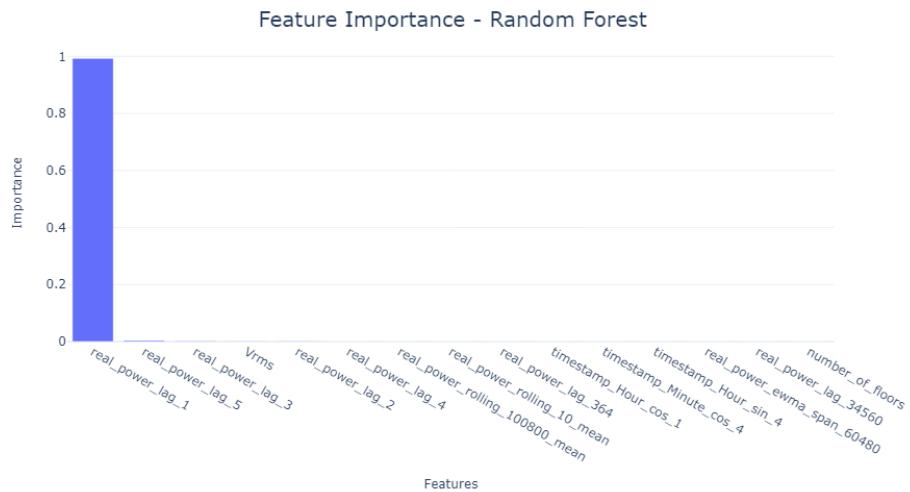


Figure 64 Graphic des caractéristiques avec leur importance (forêt aléatoire)

Les forêts aléatoires sont des ensembles d'arbres de décision qui combinent les prédictions individuelles de plusieurs arbres pour obtenir une prédition finale. Comparé aux arbres de décision simples, les forêts aléatoires permettent de réduire l'erreur en moyennant les prédictions de plusieurs arbres, ce qui améliore la capacité du modèle à généraliser. En agrégeant les prédictions de multiples arbres, les forêts aléatoires peuvent capturer des relations plus complexes et atténuer l'impact des valeurs aberrantes, ce qui conduit à une réduction de l'erreur :

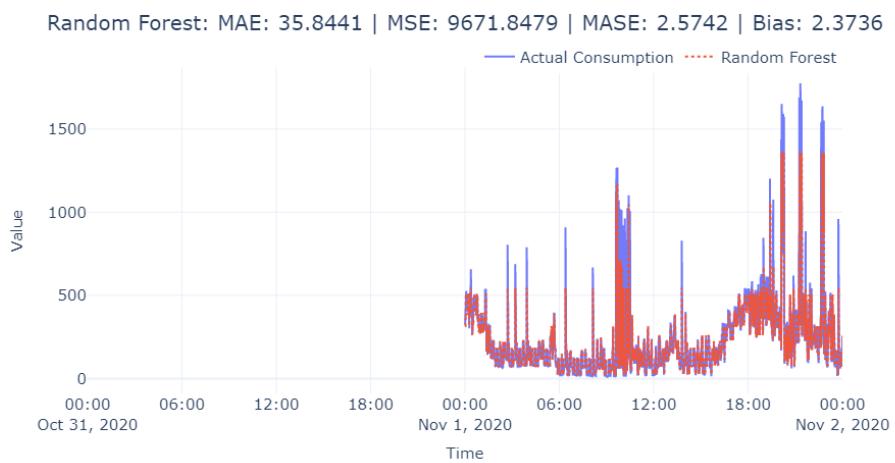


Figure 65 Résultat de la prédition comparé au valeurs réelle (forêt aléatoire)

Dans une forêt aléatoire (random forest), chaque arbre de décision est construit de manière indépendante, sans aucun processus d'optimisation tel que la descente de gradient. Cela signifie que chaque arbre est construit en utilisant un sous-ensemble aléatoire des données d'entraînement et un sous-ensemble aléatoire des caractéristiques. La construction de chaque arbre se fait de manière récursive, où à chaque nœud, la caractéristique qui offre la meilleure séparation est choisie en se basant sur certains critères (par exemple, l'impureté de Gini ou le gain d'information). Ce processus se poursuit jusqu'à ce qu'un critère d'arrêt soit atteint, comme atteindre une profondeur maximale ou un nombre minimum d'échantillons par feuille. Étant donné que



les arbres sont construits de manière indépendante, il n'y a aucun processus d'optimisation itératif impliqué, tel qu'ajuster les poids ou minimiser une fonction de perte à l'aide de la descente de gradient. Au lieu de cela, la forêt aléatoire se concentre sur le pouvoir décisionnel collectif de l'ensemble des arbres plutôt que d'optimiser individuellement chaque arbre.

▪ Gradient Boosting Machine

La classe **XGBRFRegressor** est une implémentation du modèle de régression **XGBoost** avec des forêts aléatoires comme modèle de base. Elle combine les avantages des forêts aléatoires avec les techniques d'optimisation de gradient boosting pour améliorer les performances de prédiction. XGBRFRegressor utilise des arbres de décision pour construire un ensemble de modèles prédictifs, en prenant en compte les interactions entre les caractéristiques et en ajustant les poids des échantillons pour minimiser l'erreur de prédiction. Il dispose de paramètres personnalisables tels que le nombre d'estimateurs, la profondeur maximale des arbres et les taux d'apprentissage pour affiner le modèle en fonction des données spécifiques.

Une remarque importante à souligner est que les algorithmes de boosting par gradient, tels que les Gradient Boosting Machines (GBM), utilisent le concept de gradients pour traiter les erreurs commises par les modèles précédents. Les gradients représentent la direction et l'amplitude des erreurs, indiquant comment les prédictions doivent être ajustées pour minimiser ces erreurs. À chaque itération du processus de boosting par gradient, un nouveau modèle est entraîné pour approximer les gradients négatifs (c'est-à-dire la direction dans laquelle les erreurs doivent être réduites). En ajoutant itérativement des modèles qui ciblent les erreurs commises par les modèles précédents, l'ensemble réduit progressivement l'erreur globale et améliore les performances prédictives. L'approche basée sur les gradients offre une méthode systématique et efficace pour déclencher et traiter les erreurs plus élevées, conduisant à l'amélioration des capacités prédictives de l'ensemble.

```
#Importe la classe XGBRFRegressor du module xgboost.
from xgboost import XGBRFRegressor

model_config = ModelConfig(
    model=XGBRFRegressor(random_state=42, max_depth=4),
    name="XGB Random Forest",
    # XGBRF n'est pas affecté par la normalisation
    normalize=False,
    # XGBRF gère les valeurs manquantes
    fill_missing=False,
)
with LogTime() as timer:
    y_pred, metrics, feat_df = evaluate_model(
        model_config,
        feat_config,
        missing_value_config,
        train_features,
        train_target,
        test_features,
        test_target,
    )
    metrics["Time Elapsed"] = timer.elapsed
    metric_record.append(metrics)
    pred_df = pred_df.join(y_pred)
```

Dans cette optique, le modèle XGBoost permet de donner plus d'importance à d'autres caractéristiques grâce à son algorithme de formation et d'optimisation, contrairement à des modèles tels que les forêts aléatoires ou les arbres de décision qui peuvent être plus sévères dans leur sélection de caractéristiques. XGBoost utilise des techniques telles que le gradient boosting pour ajuster les prédictions et accorder une attention particulière aux caractéristiques qui contribuent le plus à la réduction de l'erreur. Cette approche plus souple permet au modèle XGBoost d'explorer et d'exploiter un ensemble plus large de caractéristiques potentiellement pertinentes.

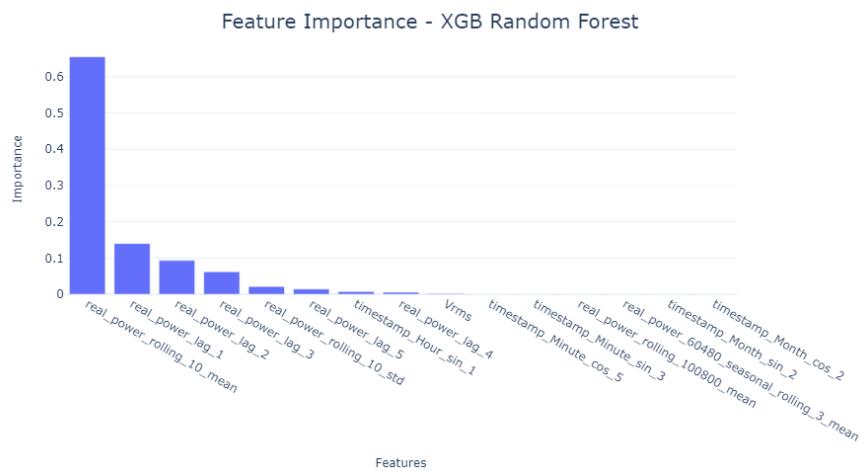


Figure 66 Graphic des caractéristiques avec leur importance (forêt aléatoire XGB)

Malgré cela, le modèle XGBoost reste proche des forêts aléatoires et des arbres de décision en termes d'erreur et de performance de prédiction globale. Bien qu'il permette une exploration plus large des caractéristiques, il conserve néanmoins une certaine similitude avec ces modèles classiques. Cela souligne la capacité de XGBoost à obtenir des performances similaires tout en offrant des avantages supplémentaires en termes de flexibilité et d'optimisation des caractéristiques.

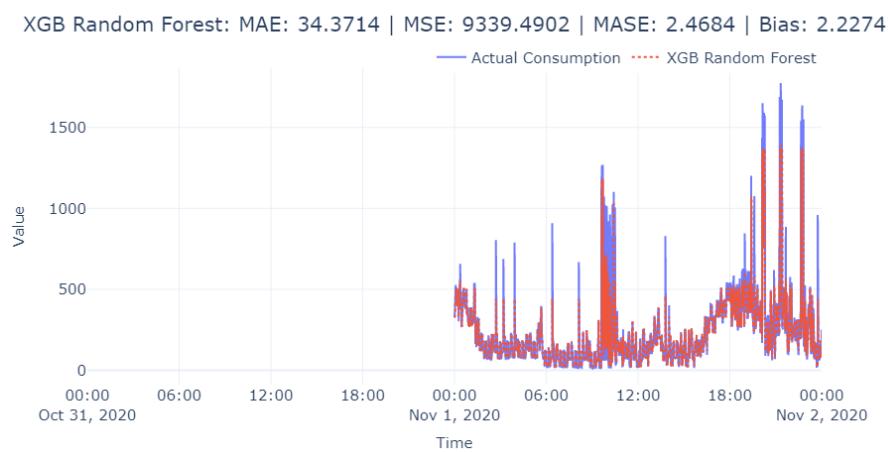


Figure 67 Résultat de la prédiction comparé au valeurs réelle (forêt aléatoire XGB)

▪ Gradient Boosting Machine

Le modèle LightGBM est implémenté dans la bibliothèque Python LightGBM. Pour utiliser ce modèle, nous devons importer la classe LGBMRegressor ou LGBMClassifier selon le type de problème (régression ou classification). Ensuite, nous pouvons initialiser une instance de ce modèle en spécifiant les paramètres nécessaires tels que le nombre d'arbres, la profondeur maximale, la vitesse d'apprentissage, etc. Une fois le modèle initialisé, nous pouvons le former sur nos données d'entraînement à l'aide de la méthode fit, et effectuer des prédictions sur de nouvelles données à l'aide de la méthode predict. Il est important de configurer les paramètres du modèle de manière appropriée pour obtenir de bonnes performances et éviter le surajustement ou le sous-ajustement.

L'importance décroissante des caractéristiques suggère que ces premières caractéristiques captent une grande partie de la variation dans les données et fournissent déjà des informations essentielles pour les prédictions. Les caractéristiques subséquentes, bien que moins importantes, peuvent apporter des informations complémentaires, mais leur contribution relative diminue progressivement.

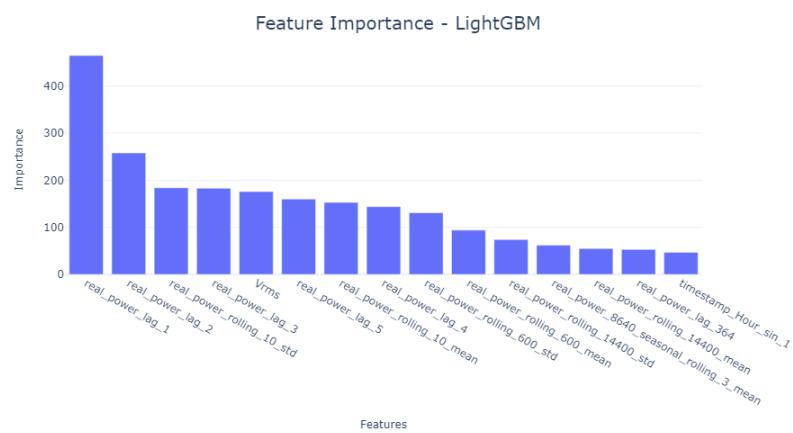


Figure 68 Graphic des caractéristiques avec leur importance (LightGBM)

Cette observation peut être utile pour la sélection de caractéristiques, en se concentrant davantage sur les premières caractéristiques les plus importantes et en réduisant l'attention sur les caractéristiques moins pertinentes. Cela peut permettre de simplifier le modèle, d'améliorer l'efficacité de calcul et de réduire les risques de surajustement.

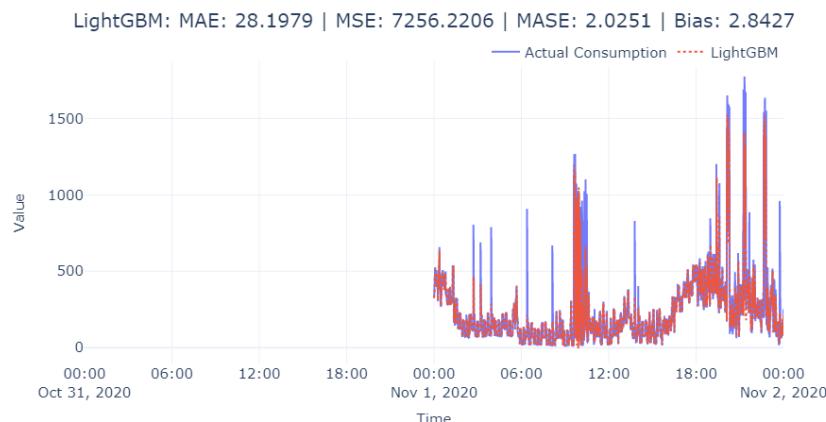


Figure 69 Résultat de la prédiction comparé au valeurs réelle (LightGBM)

Annexe G- Bibliothèques Python et Structure du Projet :

Il existe plusieurs bibliothèques disponibles pour l'analyse de séries temporelles en Python. Certaines des bibliothèques les plus populaires comprennent, cependant, elles ne proposent pas toutes les fonctionnalités souhaitées ni les implémentations des nouvelles méthodes telles que l'apprentissage automatique pour les séries temporelles ou l'apprentissage profond avec des architectures de modèles spécialisées. De plus, ces bibliothèques ne sont pas aussi matures que des bibliothèques bien établies comme scikit-learn.

En raison de leur caractère open source, ces bibliothèques peuvent être modifiées et enrichies par des contributions externes. La modification la plus simple consisterait à ajouter de nouveaux attributs et méthodes aux classes existantes, ce qui permettrait d'étendre leurs fonctionnalités de manière personnalisée.

- **Darts :**

Darts est une bibliothèque Python pour la prévision de séries temporelles. C'est une bibliothèque performante basée sur l'apprentissage profond.



Figure 70:Logo DARTS

- **Scikit-learn :**

Scikit-learn est une bibliothèque d'apprentissage automatique polyvalente. Elle comprend plusieurs algorithmes différents pour la prévision de séries temporelles, notamment ARIMA, Exponential Smoothing et Prophet.



Figure 71:Logo Scikit-learn

Voici un tableau qui résume les principales caractéristiques de certaines des bibliothèques les plus populaires pour l'analyse des séries temporelles en Python

Ce comparatif a été réalisé en utilisant la documentation de différentes bibliothèques.

Tableau 2:Principales caractéristiques des librairies d'analyse des séries temporelles

Bibliothèque	Fonctionnalité	Facilité d'utilisation	Précision	Support
Darts	Prévision de séries temporelles basée sur l'apprentissage profond	Plus difficile à utiliser	Généralement plus précis	Développement actif
Scikit-learn	Une variété d'algorithmes de prévision de séries temporelles	Plus facile à utiliser	Moins précis que Darts	Bien maintenue et supportée
Facebook Prophet	Prévision hiérarchique de séries temporelles	Facile à utiliser	Bonne précision	Développement actif
Statsmodels	Estimation, test et prévision de nombreux modèles statistiques, y compris les modèles de séries temporelles	Modérément difficile à utiliser	Bonne précision	Bien maintenue et supportée
Pmdarima	Estimation et prévision de modèles de séries temporelles, y compris ARIMA, Exponential Smoothing et Prophet	Modérément difficile à utiliser	Bonne précision	Développement actif

- **Jupyter Notebook**

Après l'importation des bibliothèques nécessaires pour la construction de notre modèle on commence par chargement de notre dataset au niveau de Jupyter notebook à l'aide de la fonction **read_csv()** .Aussi, utilisez la fonction **read_parquet()** pour lire les fichiers avec l'extension 'parquet', ce qui optimise l'utilisation de la mémoire pour un grand ensemble de données, le cas échéant , en Python.

- **Structure du projet et utilité de chaque dossier**

Ce schéma représente la structure du code du projet, organisé en quatre dossiers principaux, chacun ayant une utilité spécifique. Le dossier "data" contient les jeux de données et les métadonnées liées au projet. Le dossier "imgs" stocke les différentes images et visualisations générées pendant le projet, offrant une compréhension visuelle des données. Le dossier "notebooks" regroupe les notebooks Jupyter couvrant différents aspects du projet tels que l'exploration des données, le prétraitement, l'ingénierie des caractéristiques et l'apprentissage automatique. Enfin, le dossier "src" contient les fichiers de code source organisés en sous-répertoires, couvrant des fonctionnalités telles que la décomposition des données, l'apprentissage profond, l'ingénierie des caractéristiques, la prévision, l'imputation, la détection des valeurs aberrantes, les transformations et les utilitaires.

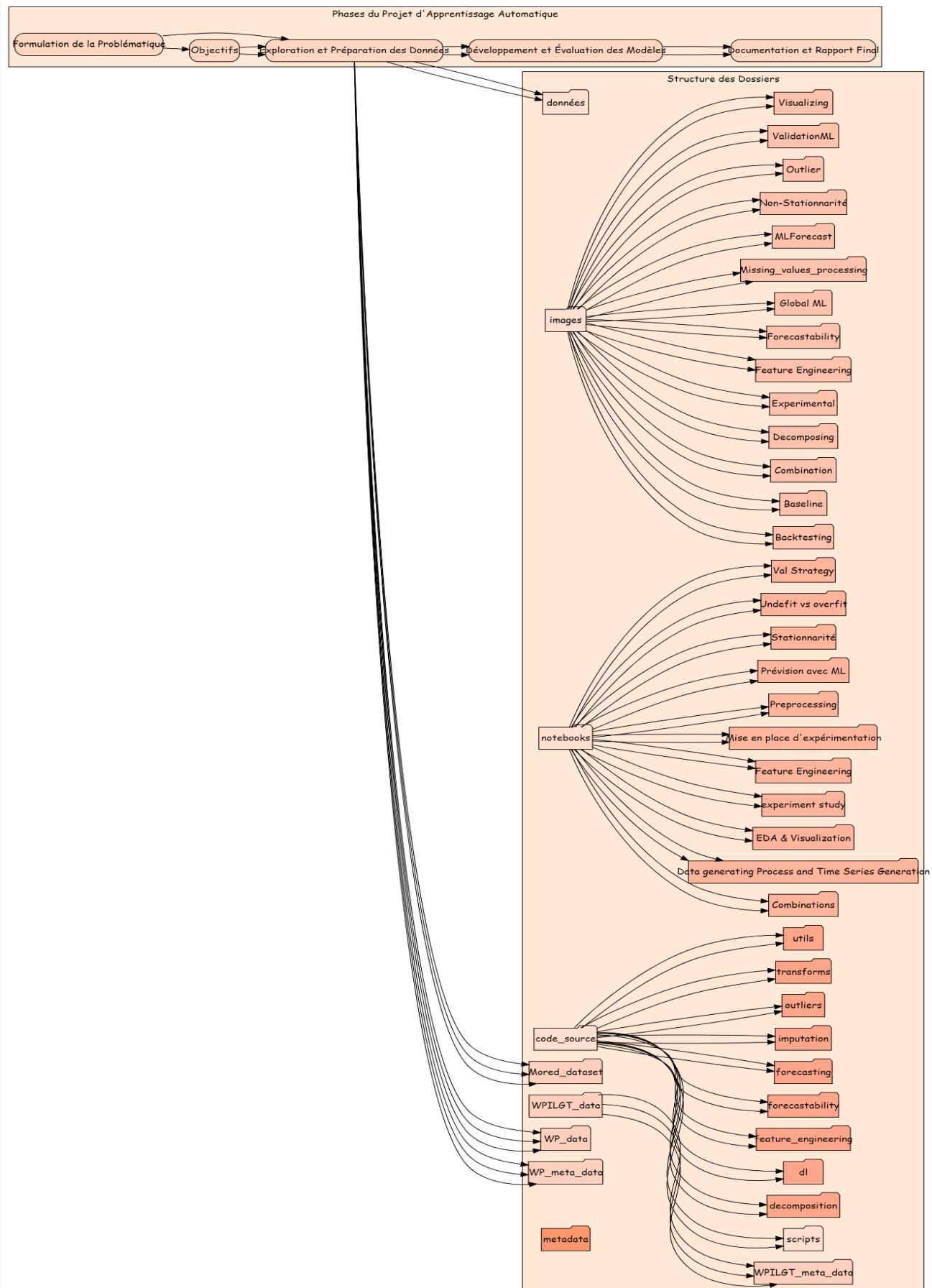


Figure 72 Structure du dossier du projet