Une image contenant capture d’écran, Graphique, conception

Description générée automatiquementUne image contenant capture d’écran, Graphique, graphisme, Caractère coloré

Description générée automatiquement=

Université Paris-Est Créteil

**Master :**

Méthodes Appliquées de la Statistique et de l’Econométrie pour la Recherche, l’Analyse et le Traitement de l’Information

**Module :**

MEMOIRE

**APPROCHE PROBABILISTE DE LA PREVISION DE LA CONSOMMATION D’ENERGIE A L’AIDE D’UN RESEAU BAYESIEN DYNAMIQUE**

**Présenté et soutenu par :**

Lajoie BENGONE AKOU

**Sous la supervision de :**

Sylvain Chareyron

Enseignant chercheur

**Année universitaire** :

2024-2025

REMERCIEMENTS

*Avant toute chose, je souhaiterai remercier très chaleureusement Monsieur Sylvain Chareyron qui m’a permis de réaliser cette étude. Malgré ses nombreuses occupations, il s’est toujours montré disponible lorsque j’avais besoin de son aide. Sa franchise et ses conseils ont fortement influencé l’orientation de ce projet. Je pense tout particulièrement aux suggestions par rapport à l’introduction et à l’idée de comparer les modèles de prévision.*

*Aussi, tout au long de ce travail, de nombreuses personnes ont manifesté de l’intérêt pour cette thématique. Je pense principalement à Warren, Léodia , Josué, Aurel, Gomez et Lévy avec lesquels j’ai entretenu des échanges qui m’ont permis de mieux appréhender les enjeux de ce mémoire.*

*Enfin, je remercie tous mes proches qui m’ont toujours soutenu et accompagné. Il s’agit de ma Famille, mes Amis, mes Camarades et mes Collègues de travail.*

TABLE DES MATIERES

[1 REMERCIEMENTS 2](#_Toc198107951)

[2 Introduction 5](#_Toc198107952)

[3 Revue de littérature 6](#_Toc198107953)

[3.1 Modèles conventionnels de régression et de séries temporelles 6](#_Toc198107954)

[3.2 Modèles d’apprentissage automatique 7](#_Toc198107955)

[3.3 Modèles d’apprentissage profond 7](#_Toc198107956)

[3.4 Modèles hybrides 8](#_Toc198107957)

[4 TABLEAU DE SYNTHESE 9](#_Toc198107958)

[5 Etude empirique 12](#_Toc198107959)

[5.1 Données 12](#_Toc198107960)

[5.1.1 Présentation du programme de recherche UKDALE 12](#_Toc198107961)

[5.1.2 Préparation de la base de données 13](#_Toc198107962)

[5.1.3 Détermination de la variable dépendante 15](#_Toc198107963)

[5.2 Analyses Exploratoires 15](#_Toc198107964)

[5.2.1 Statistiques descriptives 16](#_Toc198107965)

[5.2.2 Le diagramme de Tukey 18](#_Toc198107966)

[5.2.3 Séries chronologiques 19](#_Toc198107967)

[6 MODELES DE PREVISION 21](#_Toc198107968)

[6.1 Modèle ARIMA 21](#_Toc198107969)

[6.1.1 Etude de la stationnarité 22](#_Toc198107970)

[6.1.2 La différenciation 23](#_Toc198107971)

[6.1.3 Identification des modèles ARIMA 24](#_Toc198107972)

[6.1.4 Estimation des paramètres du modèles 24](#_Toc198107973)

[6.1.5 Résultats et Discussion 25](#_Toc198107974)

[6.2 Réseau Bayésien Dynamique 26](#_Toc198107975)

[6.2.1 Préparation de la base de données 26](#_Toc198107976)

[6.2.2 Apprentissage du Réseau Bayésien Dynamique 27](#_Toc198107977)

[6.2.3 Estimation des paramètres du modèles 27](#_Toc198107978)

[6.2.4 Résultats et Discussion 30](#_Toc198107979)

[7 CONCLUSION 31](#_Toc198107980)

[8 BIBLIOGRAPHIQUE 32](#_Toc198107981)

[9 ANNEXES 34](#_Toc198107982)

[9.1 Exploration des données : Graphiques supplémentaires 34](#_Toc198107983)

[9.2 PROGRAMME : Exploration des donnée 37](#_Toc198107984)

[9.3 PROGRAMME : MODELE DE SERIES TEMPORELLES : ARIMA 63](#_Toc198107985)

[9.4 PROGRAMME : RESEAUX BAYESIEN DYNAMIQUE LINEAIRE 82](#_Toc198107986)

[Figure 1: Système de collecte de données. Extrait (Kelly1 & Knottenbelt1, 2015) 12](#_Toc198110208)

[Figure 2 :Extrait de l'architecture des données de consommation du [réfrigérateur] du Foyer 1 14](#_Toc198110209)

[Figure 3 : Diagramme de Tukey representation la consommation d'électricité du foyer 1 par heure 18](#_Toc198110210)

[Figure 4 : Diagramme de Tukey representation la consommation d'électricité du foyer 2 par heure 19](#_Toc198110211)

[Figure 5 : Diagramme de Tukey representation la consommation d'électricité du foyer 2 par heure 19](#_Toc198110212)

[Figure 6 : Série chronologique de la consommation d'électricité du Foyer 1 par heure 20](#_Toc198110213)

[Figure 7 : Série chronologique de la consommation d'électricité du Foyer 2 par heure 20](#_Toc198110214)

[Figure 8 : Série chronologique de la consommation d'électricité du Foyer 5 par heure 21](#_Toc198110215)

[Figure 9 : : ACF et PACF : foyer 5 22](#_Toc198110216)

[Figure 10 : : ACF et PACF : foyer 2 22](#_Toc198110217)

[Figure 11 : : ACF et PACF : foyer 1 22](#_Toc198110218)

[Figure 12 : Série différenciée : Foyer 1 23](#_Toc198110219)

[Figure 13 : Série différenciée : Foyer 2 23](#_Toc198110220)

[Figure 14 : Série différenciée : Foyer 5 24](#_Toc198110221)

[Figure 15 : Architecture du Réseau Bayésien Dynamique 27](#_Toc198110222)

[Figure 16 : Schéma du nœud « apparent power » du foyer 5 30](#_Toc198110223)

[Figure 17 : Histogramme de la consommation d'électricité - Foyer 1 34](#_Toc198110224)

[Figure 18 : Histogramme de la consommation d'électricité - Foyer 2 34](#_Toc198110225)

[Figure 19 : Histogramme de la consommation d'électricité - Foyer 5 34](#_Toc198110226)

[Figure 20 : Consommation d'énergie du foyer 1 : filtré 35](#_Toc198110227)

[Figure 21 : Consommation d'énergie du foyer 2 : filtré 35](#_Toc198110228)

[Figure 22 : Consommation d'énergie du foyer 5 - filtré 35](#_Toc198110229)

[Figure 23 : Décomposition de la série - Foyer 1 36](#_Toc198110230)

[Figure 24 : Décomposition de la série - Foyer 2 36](#_Toc198110231)

[Figure 25 : Décomposition de la série - Foyer 5 36](#_Toc198110232)

Introduction

L’évolution de l’humanité a été permise par une succession de découvertes. Parmi elles figure la maîtrise des énergies. Ces énergies, qui se manifestent sous différentes formes, ont contribué de façon significative à l’amélioration des conditions de vie des individus et au dynamisme des sociétés actuelles.

En effet, de la gestion du patrimoine routier à l’interopérabilité du système bancaire, en passant par la maintenance des réseaux de télécommunication, le fonctionnement des infrastructures modernes est assuré en grande partie par l’intervention des énergies.

Cependant, la disponibilité de ces énergies tend de plus en plus à diminuer, menaçant ainsi les modes de production et de consommation actuels. Les causes apparentes de ce phénomène sont essentiellement d’ordre démographique et climatique. Parmi elles, on retrouve la croissance de la population mondiale et la raréfaction des ressources naturelles, qui créent des tensions sur le marché de l’énergie en matière d’approvisionnement, de distribution et de consommation. Ces tensions, si elles ne sont pas maîtrisées, peuvent entraîner des pertes financières aussi bien chez les consommateurs que chez les gestionnaires d’énergie.

Cela a notamment été le cas pour les consommateurs européens, qui, en 2022, ont été affectés par l’augmentation des coûts de l’électricité. Il en résulte une baisse du pouvoir d’achat pour les ménages et une perte de compétitivité pour les industriels (International Energy Agency, 2023).

Ainsi, face à cette crise énergétique, plusieurs travaux ont été menés dans le but d’utiliser de façon plus efficiente l’énergie disponible. Parmi eux, on retrouve le développement de modèles de prévision de la consommation d’énergie à partir des méthodes d’analyse de données. Ces modèles de prévision bénéficient aujourd’hui d’une grande attention de la part de la communauté scientifique, qui voit en eux un moyen d’estimer la demande future d’énergie. Cette demande estimée, si elle s’avère correcte et parfaitement ajustée à l’offre d’énergie, permettra à long terme de réaliser des économies et de préserver l’environnement.

Par conséquent, c’est au regard de ces différents enjeux que le présent ouvrage se concentre sur le développement de l’une de ces méthodes. En effet, il s’agira d’élaborer un modèle de prévision de la consommation d’énergie à l’aide d’un Réseau Bayésien Dynamique (RBD). Cette méthode, qui se base sur une approche probabiliste plutôt que fréquentiste, se veut prometteuse dans la mesure où elle permettrait de dépasser le problème de « boîte noire » rencontré par les modèles d’apprentissage automatique et profond.

Ainsi, c’est dans le but de développer ce modèle prédictif que lesdits travaux ont été structurés comme suit.

Dans un premier temps, il sera question d’effectuer une synthèse bibliographique de l’ensemble des méthodes d’analyse de données employées dans le cadre de la prévision de la consommation d’énergie. On retrouvera dans cette première partie les objectifs des études, les méthodologies employées et les résultats des modèles.

La deuxième partie, quant à elle, s’articulera autour de la présentation de la base de données UK-DALE. Elle comprendra une description des techniques de collecte de données impliquant l’utilisation des *smart meters* et un inventaire de la base de données (identifiants, variables, etc.).

Enfin, la dernière partie portera sur le déploiement de l’algorithme de prévision. Cette partie détaillera la méthodologie retenue dans la construction du modèle de prévision, les analyses exploratoires, les résultats du Réseau Bayésien et les perspectives des futurs travaux.

# Revue de littérature

La prévision de la consommation d’énergie est un sujet complexe, qui comporte notamment des enjeux économiques et climatiques. Cette thématique suscite un vif intérêt au sein de la communauté scientifique qui ne cesse de proposer des modèles de plus en plus sophistiqués afin d’atteindre l’optimalité en matière d’efficience énergétique. Cette synthèse bibliographique présente un aperçu non exhaustif des principales méthodes d’analyse de données utilisées à ce jour.

## Modèles conventionnels de régression et de séries temporelles

Les modèles conventionnels regroupent les modèles de régression linéaire, non linéaire, ainsi que ceux basés sur les séries temporelles. Ces approches font partie des premières méthodes d’analyse de données employées dans le cadre de cette thématique. Elles ont notamment été utilisées pour prédire la consommation d’énergie à l’échelle nationale, sectorielle et résidentielle.

En 1994, les premiers modèles de prévision de la consommation d’énergie apparaissaient déjà dans l’étude d’Al-Garni (1994). En effet, dans leurs travaux, Al-Garni *et al.* ont proposé un modèle de prévision de la consommation d’électricité basé sur une régression linéaire multiple, à partir de données collectées entre 1987 et 1992 en Arabie Saoudite. Leur modèle prenait pour variable à expliquer la consommation mensuelle d’électricité et pour variables explicatives : la température ambiante, la taille de la population, l’humidité, et les radiations solaires.

Plus tard, en 2006, Murat *et al.* ont poursuivi dans cette voie, en utilisant cette fois la consommation nette d’électricité comme variable cible. Les variables explicatives, quant à elles, se rapportaient au coût de l’électricité de quatre catégories de source d’énergie.

Cependant, ces méthodes de régression linéaire ont progressivement été délaissées au profit des modèles de régression non linéaire, jugés plus adaptés pour modéliser le caractère complexe et non linéaire de la consommation d’énergie.

C’est dans ce sens que s’inscrivent les travaux de Hao *et al.* (2015), qui ont développé trois modèles de régression intégrant une transformation logarithmique afin de prédire la consommation de charbon en 2020. Le présent ouvrage n’approfondit pas davantage cette catégorie de méthodes, dans la mesure où plusieurs études bibliographiques, telles que celles de Azhar *et al.* (2016), en proposent déjà une analyse plus détaillée.

Bien que les modèles de régression non linéaire aient permis une avancée significative dans la prévision de la consommation d’énergie, ils présentent, tout comme les modèles linéaires, une limite structurelle d’ordre mathématique : ils ne prennent pas en compte le caractère temporel du phénomène associé à la consommation d’énergie. En résumé, les modèles de régression ne parviennent pas à capturer la dépendance historique entre les observations.

Ainsi, afin de surmonter cette limitation, les modèles de séries temporelles ont été largement développés. En effet, grâce à l’intégration de notions telles que l’autocorrélation et l’autocovariance, ces modèles permettent de prendre en compte l’historicité des données de consommation d’énergie dans l’élaboration de modèles prédictifs. Très efficaces, ils ont été utilisés dans de nombreuses études.

Parmi elles, on peut citer celle de Chujai et Kerdprasop (2013), qui propose une comparaison entre les modèles ARMA et ARIMA pour la prévision de la consommation d’électricité à court, moyen et long termes. Les modèles ont été estimés à l’aide de la méthodologie de Box-Jenkins, et les résultats ont montré que le modèle ARIMA est préférable pour les prévisions mensuelles et semestrielles, tandis que le modèle ARMA convient mieux aux périodicités journalières et hebdomadaires. Les critères d’évaluation utilisés dans cette étude sont l’AIC et le RMSE.

En 2014, Yasmeen et Sharif ont développé et comparé quatre modèles de séries temporelles – ARIMA, SARIMA, ARCH et GARCH – pour la prévision de la consommation mensuelle d’électricité. Ces modèles, entraînés à partir de données collectées en Palestine entre janvier 1990 et décembre 2011, ont été estimés et évalués à l’aide de trois critères : l’AIC, le BIC et le MAPE. Les deux premiers ont permis de déterminer que les modèles les plus appropriés sont ARIMA(3,1,2), SARIMA(2,1,2), ARCH(2) et GARCH(1,1). Le MAPE, utilisé pour évaluer les performances prédictives, a révélé que le modèle ARIMA est le plus précis, avec un MAPE de 5,99, contre 10,51 pour SARIMA, 11,71 pour ARCH et 9,99 pour GARCH.

Enfin, les revues de littérature proposées par Mat Daut *et al.* (2017) et Deb *et al.* (2017) offrent des compléments utiles pour approfondir ces travaux.

## Modèles d’apprentissage automatique

Les hypothèses de normalité des résidus, de non-stationnarité, ainsi que la difficulté liée au caractère non linéaire des données constituent autant de faiblesses des modèles statistiques étudiés précédemment (Mat Daut *et al.*, 2017). Ces limites, combinées à l’émergence des données massives de consommation d’énergie, ont conduit les chercheurs à s’orienter vers des méthodes considérées comme plus modernes, telles que les approches d’apprentissage automatique.

Comme l’illustre la célèbre étude de Dong *et al.* (2005), certains modèles d’apprentissage automatique, notamment les machines à vecteurs de support (SVM), se distinguent par leur grande précision. Cela s’explique en partie par leur capacité à limiter les effets du surapprentissage et à modéliser des phénomènes non linéaires (Mat Daut *et al.*, 2017). Dans cette étude, le SVM, entraîné à partir d’observations mensuelles provenant de quatre bâtiments commerciaux, a obtenu un coefficient de variation (CV) inférieur à 3 % et une erreur de pourcentage inférieure à 4 %.

Compte tenu de cette efficacité, de nombreux autres travaux ont exploré l’usage de l’apprentissage automatique, notamment les méthodes dites *ensemblistes*.

En 2016, Wang et Srinivasan ont comparé les performances d’un arbre de régression (RT) à celles d’une forêt d’arbres décisionnels (EBT) pour la prévision de la consommation d’électricité. Les modèles, intégrant comme variables exogènes les données météorologiques et l’occupation des logements, ont été entraînés sur 89 % des observations, puis testés sur les 11 % restantes. Les résultats montrent que la forêt d’arbres décisionnels présente de meilleures performances que l’arbre de régression, comme en témoignent les valeurs du R² (0,90 contre 0,82), du MAPE (3,17 % contre 3,89 %) et du RMSE (0,88 contre 0,93).

À la lumière de ces résultats prometteurs, des revues de littérature plus approfondies ont été menées (Ahmad *et al.*, 2014 ; Deb *et al.*, 2017).

## Modèles d’apprentissage profond

L’efficacité des algorithmes d’apprentissage automatique est aujourd’hui indéniable. Cependant, ces derniers nécessitent des optimisations intensives et un temps de calcul relativement long (Ahmad et al., 2014, Amalou et al., 2022 ). Face à ces contraintes, d’autres méthodes d’apprentissage basées sur des architectures dites « profondes » ont été investiguées. Ils d’agit des modèles d’apprentissage profond à l’instar des Perceptrons Multicouches (MLP), des Réseaux de neurones Convolutifs (CNN) et des Réseaux de Neurones Récurrents (RNN).

Ainsi, dans leur étude, (Azadeh et al., 2008) ont développé un MLP pour prédire la consommation annuelle d’électricité dans les secteurs à forte intensité énergétique. Le réseau de neurone, caractérisé par une fonction d’activation sigmoïde a obtenu un MAPE de 0.0099 qui est largement inférieur à celui d’un modèle de régression non linéaire évalué à 0.075.

Enfin, (Amalou et al., 2022) ont conçu et comparé un modèle d’apprentissage profond et deux autres modèles hybrides. Il est question d’un réseau de neurone récurrent classique (RNN), d’un modèle de Long Short Term Memory (LSTM) et d’un modèle à Unité récurrente fermée (GRU). Les résultats indiquent que le GRU performe mieux que le LSTM et largement mieux que le RNN classique.

Bien que l’utilisation des réseaux de neurones pour la prévision de la consommation d’énergie soit un sujet de recherche largement documenté (Ahmad *et al*., 2014 ; Deb *et al*., 2017). il est important de noter que ces intelligences artificielles à l’instar des SVM souffrent d’un problème de « boîte noire » qui les prive de toute forme explicabilité (Foucquier et al., 2013)

## Modèles hybrides

Il en va de soi que chacune des méthodes énumérées ci-dessus revêt des avantages comme des inconvénients. De ce fait, c’est dans but de tirer parti des forces uniquement de ces méthodes qu’une approche par hybridation, bien que plus complexes, est de plus en plus privilégiée. Enfin, cette approche qui consiste à fusionner des modèles issues de différentes catégories dans le but dans de conserver uniquement leurs avantages est aujourd’hui considérée comme l’une plus efficace en matière de prévision et d’explicabilité.

Parmi ces modèles, on retrouve le modèles à machine à vecteurs de support des moindres carrés (LSSVM) qui offre un gain considérable de temps de calcul et une meilleure précision que les SVM. Il a notamment été utilisé dans l’étude de (Kaytez *et al.*, 2015) où il est comparé à une régression multiple et un réseau de neurones récurrent. Les résultats indiquent que les prévisions du LSSVM sont plus précises que celles des deux modèles au regard du MAPE et du RMSE.

Enfin, comme dernier modèle hybride proposé, on retrouve les réseaux bayésiens dynamiques qui ont été récemment utilisés pour prédire la consommation d’électricité (Singh et Yassine, 2018). Cependant, l’application de ces réseaux probabiliste est encore moins répandue bien que très prometteur (Aulia *et al.*, 2014).

En définitive, l’efficience énergétique est un enjeu majeur qui cristallise à la fois l’opinion publique et la recherche scientifique. L’une des approches privilégiée pour atteindre cet objectif reste la prévision de la consommation d’énergie. Bien que cette dernière semble être un phénomène complexe à modéliser en raison notamment de son caractère multifactoriel et non linéaire, les méthodes d’analyse de données de plus en plus sophistiquées sont développées, réduisant ainsi, au fil du temps, les erreurs de prévision et le temps de calcul.

# TABLEAU DE SYNTHESE

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Auteurs | Données | Périodicité | Variable  d’intérêt | Variables  Explicatives | Méthodes | Résultats |
| Al-Garni et al  1994 | Arabie Saoudite  (1987-1993) | Mois | Consommation d’électricité | Température, humidité, radiation solaire, population. | Régression Stepwise | L’influence de la variables associée à l’humidité est négligeable au sein du modèle de régression |
| Tunc et al  2006 | Turquie  (1980-2001) | Année | Consommation d’électricité | Coûts de production, coûts fixes et coûts variables lié à la construction | Régression linéaire Multiple | Au regard des prévision, les centrales hydroélectriques et nucléaires sont importantes pour la soutenabilité de la consommation d’énergie. |
| Hao et al  2015 | Chine  (1995-2012) | Année | Consommation de charbon par habitant | PIB par habitant, Taux d’urbanisation, Taux d’ouverture, valeur ajoutée de l’industrie secondaire | Régression avec une transformation logarithmique | Les prévisions faîtes par le modèle de régression indiquent une croissance de la consommation de charbon en chine jusqu’en 2020. |
| Chujai et al  2013 | France  (2006-2010) | Année,  Mois,  Semaine, Heure | Consommation d’électricité |  | ARMA, ARIMA | Les Résultats indiquent que le modèle ARMA est plus adapté pour les prévisions à long terme d’une part. D’autre part, le modèle ARIMA est plus efficace pour les prévisions à court terme. |
| Yasmeen and Sharif,  2014 | Palestine  (1990-2011) | Mois | Consommation d’électricité | Univarié | ARIMA, GARCH, SARIMA | ARIMA (3,1,2) model is the most appropriate model for forecasting electricity consumption of Pakistan. |
| Wang and Srinivasan  2016 | Etats-Unis  (2015.02.01-2015.02.28) | Heure | Consommation d’électricité | weather conditions, day types, and time of day | Ensemble Bagging Tree | Les capacités prédictives du modèle EBT sont supérieures à celles de l’arbre de régression (RT) avec une amélioration du MAPE de 18.5%. |
| Dong et al  2005 | Singapour  (1996-2001) | Mois | Consommation d’électricité | Univarié | SVM | Comparé aux autres modèles, le SVM présente l’un des coefficient de variation les plus faibles. Il est inférieur à 3%. |
| Ahmad et al  2014  (revue de littérature) | - | - | - | Univarié | - | Les méthodes d’apprentissage automatique ont comme principales limites : un besoin intensif d’optimisation et un temps de calcul relativement long |
| Azdeh et al  2008 | Iran  (1997-2003) | Année | Consommation d’électricité | Le prix de l’électricité, Le nombre de consommateur par secteur, intensité énergétique, valeur ajoutée par secteur | Réseaux de neurones ( Perceptron multicouche) | Les résultats montrent que la moyenne des prévisions effectuées par le modèle de MLP est statistiquement identique à celle issues des données réelles avec un intervalle de confiance à 99%. |
| Amalou et al  2022 | Australie  (2010-2014) | Heure | Consommation d’électricité | Univarié | Unité récurrente fermée | Le GRU a obtenu un meilleur RMSE que les modèles LSTM et RNN, 0.034 contre 0.039 et 0.051 respectivement. |
| Kaytez et al  2015 | Turquie  (1970-2009) | Année | Consommation net d’électricité | La capacité énergétique, la production brute d’électricité, La population, Le total des abonnements | LS-SVM | Le modèle LSSVM a obtenu des meilleurs résultats que les modèles MLR et ANN de 1,70 % et 0,88 % respectivement. |
| Aulia et al,  2024  (revue de littérature) | - | - | - |  | - | Les réseaux bayésiens dynamiques surpassent les modèles d’apprentissage automatique et profond en termes d’ |
| Zhang et al  2024 | Chine  (2020-2021) | Minute | Production d’énergie solaire | weather information, operation indicators, sensor data, and output AC power | Réseau Bayésien Dynamique | Le réseau bayésien présente une meilleure gestion de l’incertitude. |
| Singh et al  2018 | Royaume-Unis  (2012-2015)  Canada  (2012-2014) | Minute | Consommation d’électricité | hour of the day, period of the day, weekday, week, month and season of the year as well as appliance-appliance associations | Réseau Bayésien Dynamique | Le modèle de RBD présente des Résultats supérieurs à ceux du SVM et du MLP. |

Tableau : Tableau récapitulatif de l'ensemble des articles académique consultés

# Etude empirique

## Données

### Présentation du programme de recherche UKDALE

Au regard de la nécessité de concevoir des algorithmes de plus en plus sophistiqués, plusieurs programmes de recherche ont été mis en place dans le but de faciliter l’accès aux données de consommation d’énergie. Parmi ces initiatives, on retrouve la base de données UK-DALE (UK – Domestic Appliance-Level Electricity), développée par le département d’informatique de l’Imperial College London (Kelly & Knottenbelt, 2015).

Disponible en open source, UK-DALE est aujourd’hui considérée comme une référence parmi les bases de données de consommation d’énergie en raison de la richesse et de la précision des informations qu’elle fournit. Elle contient notamment des données de panel à haute fréquence : 16 kHz pour la consommation globale au niveau du foyer et 1 Hz ou 6 Hz pour la consommation par appareil. Ces données ont été collectées entre 2012 et 2015 auprès de cinq foyers, à l’aide de compteurs intelligents (« smart meters »).

Une image contenant diagramme, texte, croquis, Plan

Le contenu généré par l’IA peut être incorrect.Les compteurs intelligents sont des dispositifs électroniques conçus pour mesurer la consommation d’énergie en temps réel. Leur utilisation est de plus en plus répandue, notamment dans le contexte actuel marqué par la transition énergétique, l’essor de l’intelligence artificielle, et l’intégration croissante des nouvelles technologies de l’information et de la communication (NTIC) dans la vie quotidienne (smart city, smart home, smartphone).

Figure : Système de collecte de données. Extrait (Kelly1 & Knottenbelt1, 2015)

Au sein de UK-DALE, ces capteurs ont notamment permis de mesurer, collecter et stocker les données de consommation d’énergie qui sont organisées en dossiers, sous-dossiers et fichiers dans lesquels on retrouve :

* Les données de consommations d’électricité (puissance active, puissance apparente, tension... )
* Les données par type de bâtiments (Nombre d’occupants, année de construction, ...)
* Les données de contrôle ( date d’enregistrement, date d’installation des capteurs, date de la première mesure, date de la dernière mesure, ...)
* Les types d’appareils par résidence (réfrigérateur, Télévision, Lave-vaisselles, Micro-ondes, ...).

Toutes ces informations ont été résumées dans le tableau ci-dessous.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Foyer | 1 | 2 | 3 | 4 | 5 |
| Type de bâtiment | maison de bout de terrasse | maison de bout de terrasse |  | Maison en milieu de rangée | Immeuble résidentiele |
| Année de construction | 1905 | 1900 |  | 1935 | 2009 |
| Système de chauffage | gaz naturel | gaz naturel |  | gaz naturel | gaz naturel |
| Nombre d’occupants | 4 | 2 |  | 2 | 2 |
| Nombre de dispositifs de mesure par appareil | 54 | 20 | 5 | 6 | 26 |
| Nombre de dispositifs de mesure principal | 2 | 2 | 1 | 1 | 2 |
| Date de la première mesure | 2012-11-09 | 2013-02-17 | 2013-02-27 | 2013-03-09 | 2014-06-29 |
| Date de la dernière mesure | 2015-01-05 | 2013-10-10 | 2013-04-08 | 2013-10-01 | 2014-11-13 |
| Durée d’enregistrement (jour) | 786 | 234 | 39 | 205 | 137 |
| Durée d’enregistrement des mesures | 655 | 140 | 36 | 155 | 131 |
| Consommation moyenne d’électricité par jour (puissance active kWh) | 7.64 | 7.17 |  |  | 13.75 |
| Consommation moyenne d’électricité par jour (puissance apparente kVAh) | 8.90 | 8.00 | 12.35 | 10.24 | 17.56 |

Tableau : Données relatives à chaque Foyer. Extrait (Kelly1 & Knottenbelt1, 2015)

### Préparation de la base de données

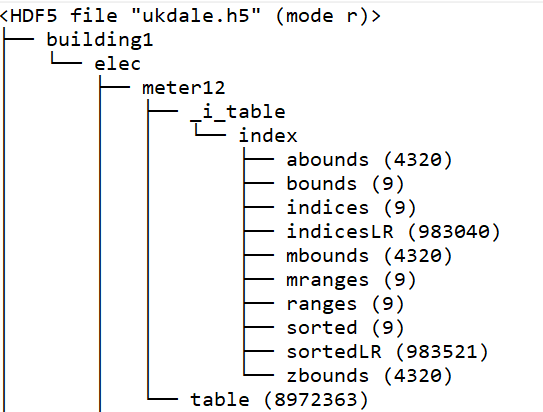
Les données massives de UK-DALE, disponibles au format HDF5, ont été exploitées à l’aide de divers outils tels que Python, Microsoft Azure, Databricks et SQL.

Tout d’abord, Python a été utilisé pour lire les fichiers, identifier l’architecture de la base de données (dont un aperçu est présenté ci-après) et convertir les fichiers au format CSV.

Ensuite, le langage SQL a été employé dans l’environnement cloud Microsoft Azure, via la plateforme Databricks, pour manipuler les données brutes. Cette étape a permis notamment le changement de type de certaines variables, la création de nouvelles variables, ainsi que l’inventaire des bases de données.

Toutes ces procédures ont été indispensables pour obtenir des données de consommations d’énergie de bonne qualité. Pour illustrer ce travail, un aperçu d’une table de données avant et après manipulation est présentés (ci-dessous).

Figure :Extrait de l'architecture des données de consommation du [réfrigérateur] du Foyer 1



Dans ces tables, on peut distinguer principalement 2 catégories de variable. La première catégorie indique la date à laquelle les données ont été enregistrées. Cette dernière englobe la variable unix et toutes les nouvelles variables temporelles présentent dans la base de données traitées (unix\_year, unix\_monthy, unix\_day, unix\_hour, unix\_minute, unix\_second).

La deuxième catégorie quant à elle représente les mesures associées à la consommation d’électricité. On y retrouve la puissance active (active\_power) exprimée en Watt (W), la puissance apparente (apparent\_power) exprimée en Volt-ampère (VAh) et la tension électrique (main\_rms\_voltage) exprimée en Volt (V).

|  |  |  |  |
| --- | --- | --- | --- |
| unix | active\_power | apparent\_power | main\_rms\_voltage |
| 1.363548e+09 | 337.88 | 431.04 | 240.15 |
| 1.363548e+09 | 339.43 | 427.94 | 240.56 |
| 1.363548e+09 | 340.63 | 429.66 | 241.07 |
| 1.363548e+09 | 338.80 | 426.99 | 240.44 |
| 1.363548e+09 | 340.88 | 429.13 | 241.01 |

Tableau : Extrait de données brutes de consommation d'énergie du Foyer 1

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| unix | unix\_year | unix\_month | unix\_day | unix\_hour | unix\_minute | unix\_second | active\_power | apparent\_power | main\_rms\_voltage |
| 2013-03-17 19:12:43.099999905 | 2013 | 3 | 17 | 19 | 12 | 43 | 337.88 | 431.04 | 240.15 |
| 2013-03-17 19:12:44.099999905 | 2013 | 3 | 17 | 19 | 12 | 44 | 339.43 | 427.94 | 240.56 |
| 2013-03-17 19:12:45.099999905 | 2013 | 3 | 17 | 19 | 12 | 45 | 340.63 | 429.66 | 241.07 |
| 2013-03-17 19:12:46.099999905 | 2013 | 3 | 17 | 19 | 12 | 46 | 338.80 | 426.99 | 240.44 |
| 2013-03-17 19:12:47.099999905 | 2013 | 3 | 17 | 19 | 12 | 47 | 340.88 | 429.13 | 241.01 |

Tableau : Extrait de données traitées de consommation d'énergie : Foyer 1

### Détermination de la variable dépendante

Après la préparation des bases de données, l’étape suivante consiste à choisir la variable indépendante qui est associée à la consommation d’énergie. Toutefois, il paraît important de rappeler que l’énergie peut revêtir plusieurs formes à l’instar de l’électricité, du gaz et de l’eau, etc. De ce fait, la consommation d’énergie se présente alors comme un phénomène polymorphe à modéliser. Néanmoins, comme dans beaucoup d’autres études, le présent ouvrage substitue la consommation finale d’énergie à la consommation finale d’électricité par souci de simplification. Ce choix repose sur le principe selon lequel l’électricité occupe la plus grande part dans la consommation finale d’énergie après le pétrole. En d’autres termes, il s’agira de modéliser la consommation d’énergie par la consommation d’électricité en utilisant comme variable dépendante la puissance active exprimée en Watt.

## Analyses Exploratoires

### Statistiques descriptives

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Critères | Variables | Foyer 1 | Foyer 2 | Foyer 5 |
| Max | Puissance active | 2829.45 | 1863.51 | 3530.69 |
| Puissance apparente | 2902.86 | 1877.75 | 3626.27 |
| Tension | 246.64 | 244.63 | 250.59 |
| Min | Puissance active | 87.51 | 0.00 | 226.11 |
| Puissance apparente | 125.26 | 145.60 | 399.48 |
| Tension | 237.77 | 236.62 | 238.76 |
| Moyenne | Puissance active | 318.55 | 298.63 | 571.98 |
| Puissance apparente | 371.27 | 333.24 | 730.42 |
| Tension | 243.18 | 241.15 | 246.12 |
| Médiane | Puissance active | 222.43 | 207.76 | 429.95 |
| Puissance apparente | 264.29 | 240.36 | 601.73 |
| Tension | 243.23 | 241.19 | 246.10 |
| Q1 | Puissance active | 155.27 | 143.33 | 316.88 |
| Puissance apparente | 196.04 | 174.12 | 488.35 |
| Tension | 242.41 | 240.42 | 244.96 |
| Q3 | Puissance active | 408.86 | 337.10 | 625.90 |
| Puissance apparente | 461.63 | 370.68 | 780.57 |
| Tension | 244.00 | 241.94 | 247.34 |
| Ecart-type | Puissance active | 242.06 | 260.33 | 396.12 |
| Puissance apparente | 261.11 | 261.29 | 377.18 |
| Tension | 1.17 | 1.13 | 1.68 |
| Variance | Puissance active | 58591.53 | 67772.01 | 156913.65 |
| Puissance apparente | 68179.43 | 68272.85 | 142265.37 |
| Tension | 1.36 | 1.27 | 2.81 |
| CV (%) | Puissance active | 75.99 | 87.18 | 69.26 |
| Puissance apparente | 70.33 | 78.41 | 51.64 |
| Tension | 0.48 | 0.47 | 0.68 |
| Skewness | Puissance active | 2.36 | 2.77 | 2.52 |
| Puissance apparente | 2.37 | 2.70 | 2.55 |
| Tension | -0.29 | -0.20 | -0.12 |
| Kurtosis | Puissance active | 8.40 | 8.84 | 8.04 |
| Puissance apparente | 7.96 | 8.32 | 8.43 |
| Tension | 0.07 | 0.13 | -0.17 |
| Valeurs nulles | Puissance active | 0.00 | 0.00 | 0.00 |
| Puissance apparente | 0.00 | 0.00 | 0.00 |
| Tension | 0.00 | 0.00 | 0.00 |

Tableau : Tableau de Statistique de la consommation d'énergie (kWh) en heure (H).

L’analyse exploratoire est une étape essentielle dans le développement des algorithmes de prévision. Elle permet d’extraire les principales caractéristiques de l’évolution de la consommation d’énergie en fonction des individus qui sont en l’occurrence les foyers et des périodicités qui peuvent être annuelles, mensuelles, journalières, horaires, en minute ou en seconde.  
Tout au long de cette étude, seuls trois individus parmi les cinq énoncés plus haut ont été retenus. Par ailleurs, l’heure (H) a été sélectionnée comme périodicité de référence. Ces précisions visent à encadrer l’étude en facilitant l’interprétation des résultats, la comparaison des modèles prédictifs et la reproduction de ces travaux.

Les paramètres de tendance centrale révèlent que le foyer 5 a une consommation horaire supérieure aux foyers 1 et 2 en moyenne et en médiane. En effet, la consommation horaire d’électricité du foyer 5 s’élève environ à 572W contre 319W et 299W qui correspondent respectivement aux foyers 1 et 2.

Dans le même ordre, il a été enregistré dans le foyer 5 un pic de consommation horaire d’électricité avoisinant les 3531W contre 2830W et 1864W qui correspondent aux foyers 1 et 2. Ces chiffres sont à mettre en opposition avec la consommation minimale d’électricité dont la plus petite mesure appartient au foyer 2 qui s’élève à 0W. Cela suggère que durant toute la période d’observation, au moins un appareil électrique est resté actif au sein des foyers 1 et 2. Ceci diffère du foyer 2, qui laisse entrevoir des périodes sans aucune mesure de consommation électrique.

S’agissant maintenant des paramètres de dispersion, on constate aisément que c’est au sein du foyer 5 que l’on mesure la plus grande variabilité en matière de consommation horaire d’électricité. En effet, l’écart-type de ce dernier s’élève à 396.12W contre 242.06W et 260.33W. Ces derniers chiffres se réfèrent respectivement aux foyers 1 et 2.

En définitive, en se basant sur ces paramètres, on peut conclure que le foyer 5 consomme en moyenne plus d’électricité que les foyers 1 et 2 par heure, d’une part. D’autre part, cette consommation est beaucoup plus volatile. De plus, on ne dénombre aucune valeur manquante. Ce type de constat est très caractéristique des bases de données cylindrées à l’instar de UKDALE qui ont une forte sélection en matière d’échantillon.

### Le diagramme de Tukey

La consommation de l’électricité est un phénomène temporel marqué fortement par un caractère saisonnier (cyclique). Comme les graphiques ci-dessous le suggèrent, ce phénomène revêt une courbe en forme de double cloche. Cette dernière a tendance à diminuer durant les heures de coucher (21h-5h) avant de fortement augmenter durant les heures d’activité (6h-9h) et (17h-19h).

Aussi, tout au long de la journée, la variance de celle-ci augmente avant d’atteindre un pic de consommation dans les alentours de (19h-20h). Ces différents comportements semblent être structurels.

Pour résumer, ce graphique a permis de mettre en évidence certaines caractéristiques de la consommation d’électricité à l’instar de la saisonnalité dans le cadre d’une périodicité par heure.

Les analyses effectuées sur cet échantillon semblent correspondrent aux observations réelles.

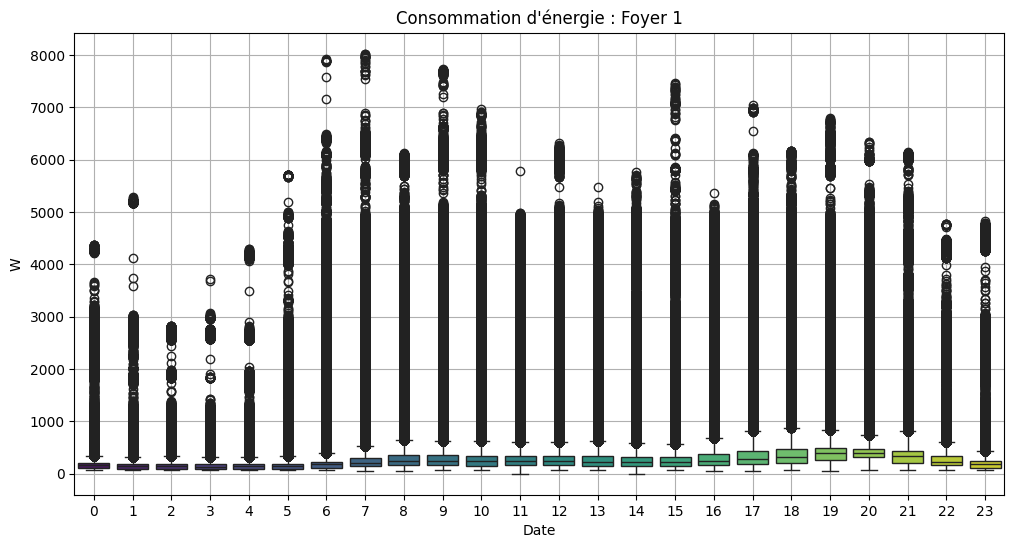


Figure : Diagramme de Tukey representation la consommation d'électricité du foyer 1 par heure

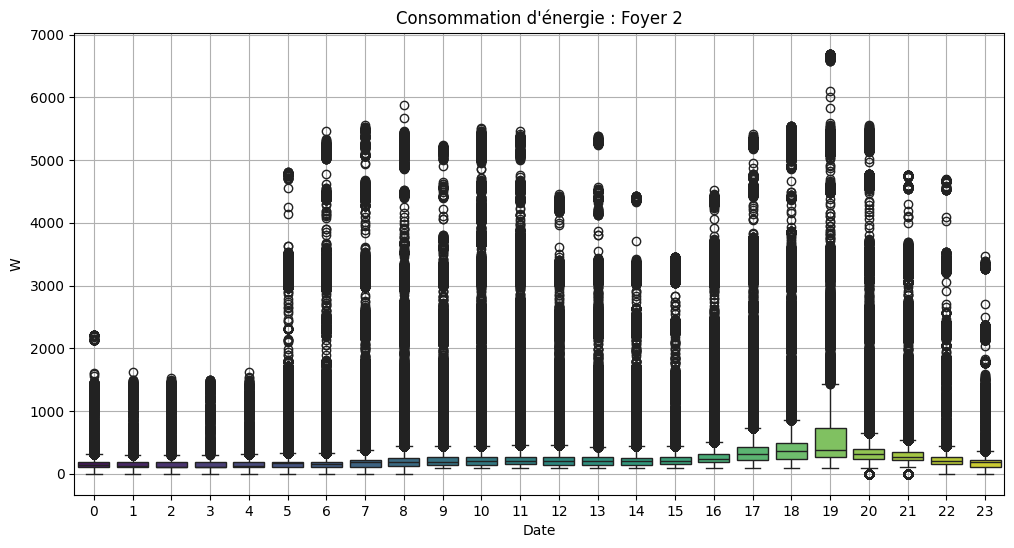


Figure : Diagramme de Tukey representation la consommation d'électricité du foyer 2 par heure

Une image contenant texte, diagramme, ligne, Parallèle

Le contenu généré par l’IA peut être incorrect.

Figure : Diagramme de Tukey representation la consommation d'électricité du foyer 2 par heure

### Séries chronologiques

Représenter visuellement la consommation d’électricité sous forme d’une série chronologique comporte plusieurs avantages tels que la possibilité d’observer des tendances, d’identifier des comportements périodiques et de détecter des anomalies.

Ainsi, tout comme les diagrammes de Tukey, l’analyse visuelle permet d’identifier la présence d’un comportement saisonnier. Bien que ce dernier ne soit plus visible en heures, on parvient à l’identifier maintenant en mois. Ainsi, on visualise aisément une hausse potentiellement structurelle de la consommation d’électricité dans la période d’octobre à janvier et une baisse de cette dernière entre les mois d’avril et d’août.

Ces variations de la consommation d’électricité semblent lier au climat et plus précisément au changement des saisons au Royaume-Uni. C’est en hiver, qu’on observerait des pics récurrents de la consommation d’électricité et c’est en automne que celles-ci seraient les plus basse.

Dans ces graphiques, l’information la plus importante semble être la continuité de la série. Les séries chronologiques des foyers 2 et 5 sont affectées par des niveaux de consommation d’électricité anormaux. Par exemple, on observe, au sein du foyer 2, entre fin août 2013 et fin septembre 2013, une augmentation linéaire du niveau de la puissance.

Concernant le foyer 5, cette anomalie est moins prononcée car elle dure moins d’un mois et à la différence de celle du foyer 2, elle présente une tendance linéaire vers le bas.  
Il est important de noter qu’aucune information n’a été donnée à ce sujet. L’une des hypothèses les plus probables serait des défauts d’enregistrement de la part des smart meters.

En définitive, toutes ces analyses ont servi à illustrer la complexité des données relatives à l’évolution de la consommation d’électricité dans le temps. Cette complexité est marquée, entre autres, par un caractère saisonnier qui dépend de la périodicité de référence, des anomalies qui peuvent être imputées aux erreurs de mesure, aux comportements des personnes au sein des foyers et à la présence de facteurs exogènes comme le climat et le nombre d’habitants.

Toutes ces contraintes doivent être prises en compte dans la construction des modèles de prévisions pour garantir des résultats fiables et de bonnes performances prédictives.

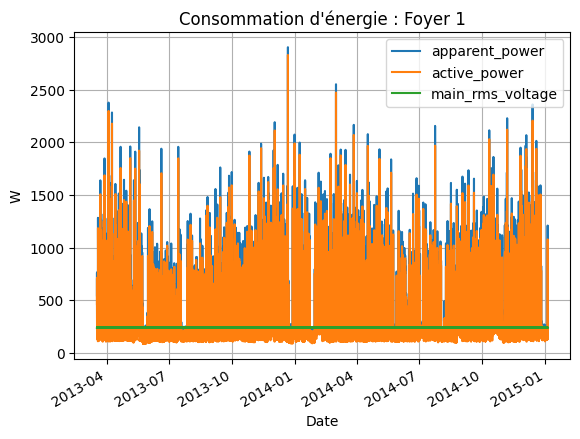


Figure : Série chronologique de la consommation d'électricité du Foyer 1 par heure

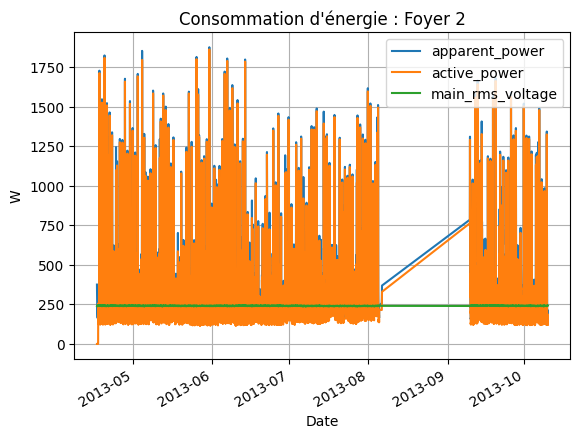


Figure : Série chronologique de la consommation d'électricité du Foyer 2 par heure

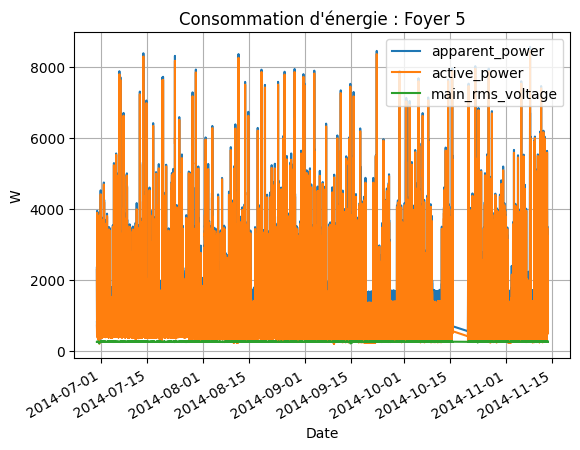


Figure : Série chronologique de la consommation d'électricité du Foyer 5 par heure

# MODELES DE PREVISION

Dans cette étude, deux modèles de prévision de la consommation d’électricité ont été développés. Il s’agit d’un modèle de séries temporelles ARIMA et d’un Réseaux Bayésien Dynamique. L’objectif est de comparer les performances du modèle graphique probabiliste à celles d’un modèle conventionnel d’analyse de données.

## Modèle ARIMA

Le modèle AutoRegressive Integrated Moving Average abrégé ARIMA est un modèle de séries temporelle univarié qui peut-être définit comme une généralisation du modèle ARMA. Il est composé de trois parties. La première partie appelée Autorégressive (AR) est une série chronologique dans laquelle les valeurs présentes sont modélisées par ses valeurs passées. La deuxième partie constitue la partie Moyenne Mobile (MA). Cette partie vise à modéliser la série chronologique par ses termes d’erreur passés. La combinaison de ces deux parties (AR) et (MA) forme un processus ARMA. Cependant, l’usage de ce dernier suppose une condition de stationnarité qui n’est pas toujours vérifiée.

Par conséquent, c’est dans le but de rendre le processus ARMA stationnaire qu’une troisième composante appelée Integrated représentée sous la forme d’un (I) est ajoutée. Cette dernière permet de rendre la série temporelle stationnaire en procédant à des calculs de différenciation.

Une fois la série différenciée et les hypothèses du modèle vérifiées, les paramètres du modèle peuvent être estimés et utilisés pour effectuer des prévisions.

Ainsi, c’est dans le but de réaliser des prévisions de la consommation d’électricité que le modèle ARIMA a été sollicité. Ce dernier a été développé en utilisant la méthodologie de Box-Jenkins qui se présente comme suit :

### Etude de la stationnarité

La première étape consiste à identifier si le phénomène associé à la consommation d’électricité au cours du temps est un processus stationnaire. Pour se faire, plusieurs méthodes ont été déployées. Parmi elles, on retrouve une analyse des autocorrélations (AFC) et des autocorrélations partielles (PAFC), une décomposition de la consommation d’électricité en tendance, saisonnalité et résidus et enfin des tests de Dickey-Fuller.

* Une image contenant texte, ligne, Tracé, nombre

  Le contenu généré par l’IA peut être incorrect.Une image contenant texte, ligne, nombre, Tracé

  Le contenu généré par l’IA peut être incorrect.Une image contenant texte, ligne, Tracé, nombre

  Le contenu généré par l’IA peut être incorrect.L’analyse des autocorrélations et des autocorrélations partielles révèle une stationnarité faible. En effet, les autocorrélations décroient progressivement avant de remonter tout légèrement après le 18ème retard. Quant aux autocorrélations partielles, elles chutent brusquement le plus souvent aux alentours du premier ordre avant de remonter légèrement vers le 18ème retard.

Figure : : ACF et PACF : foyer 5

Figure : : ACF et PACF : foyer 2

Figure : : ACF et PACF : foyer 1

* La décomposition de la série laisse transparaître une tendance légèrement descendante sur le long terme. Les composants saisonniers semblent bruités. Enfin, les résidus sont assez hétérogènes. On ne parvient pas à identifier des structures évidentes au sein des données.
* Au regard des pvaleurs des tests de Dickey-Fuller, on peut affirmer que la puissance active est un processus stationnaire même si des légères tendances ont été observées.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Foyer 1 | Foyer 2 | Foyer 5 |
| Valeur de test | -12.85 | -7.88 | -7.92 |
| Pvaleur | 5.49e-24 | 4.57-12 | 3.57-12 |
| Conclusion | Stationnaire | Stationnaire | Stationnaire |

Tableau : Tableau récapitulatif du test de Dickey-Fuller

### La différenciation

Une image contenant texte, capture d’écran, Tracé, diagramme

Le contenu généré par l’IA peut être incorrect.Bien que le processus peut sembler faiblement stationnaire au regard des éléments énoncés ci-dessus, des précautions ont tout de même été prises pour différencier dans certains cas la série. L’objectif étant d’obtenir les meilleurs estimations pour effectuer les meilleures prévisions, mêmes les cas de séries différenciées ont été étudiés. Cette étape a fait intervenir le graphique de différenciation.

Figure : Série différenciée : Foyer 1

Une image contenant texte, capture d’écran, Tracé, diagramme

Le contenu généré par l’IA peut être incorrect.

Figure : Série différenciée : Foyer 2

### Une image contenant texte, ligne, Tracé, capture d’écran Le contenu généré par l’IA peut être incorrect.Identification des modèles ARIMA

Figure : Série différenciée : Foyer 5

L’identification des modèles ARIMA s’est faite de façon itérative jusqu’à trouver les ordres qui minimisaient la perte d’information. Ces informations ont été synthétisées dans le tableau ci-dessous avec les paramètres estimés des modèles.

### Estimation des paramètres du modèles

|  |  |  |  |
| --- | --- | --- | --- |
|  | Foyer 1 | Foyer 2 | Foyer 5 |
| Model : | (1, 1, 1) | (5, 0, 3) | (0, 1, 1) |
| No Observations | 12596 | 2706 | 2539 |
| HBIQ | 168985.724 | 37153.268 | 36839.115 |
| Ljun-Box (Q) | 0.45 | 0.01 | 9.26 |
| Prob (Q) | 0.50 | 0.91 | 0.00 |
| Hétéroscédasticité (H) | 1.18 | 0.75 | 0.98 |
| Prob (H) | 0.00 | 0.00 | 0.76 |
| Jarque-Bera (JB) | 103769.31 | 15696.62 | 11679.90 |
| (JB) | 0.00 | 0.00 | 0.00 |

Tableau : Tableau récapitulatif des paramètres des Modèles ARIMA

Les modèles ARIMA des foyers 1, 2 et 5 ont été estimés à partir des échantillons de tailles respectives 125596, 27006 et 2539 observations. Ces observations sont en l’occurrence les mesures horaires de la consommation d’énergie durant 80% de la période d’enregistrement des données. Ces chiffres garantissent une fiabilité des résultats.

Le test de Ljung-Box est un test statistiques qui est utilisé pour vérifier si les résidus d’une série temporelle présentent une autocorrélation, c’est-à-dire une dépendance entre les observations t et t-1. C’est une hypothèse nécessaire pour obtenir des résidus aléatoires (bruits blancs) au sein d’un modèle ARIMA. Comme on peut le souligner les pvaleurs des foyers 1 et 2 sont inférieures à 5% donc les résidus de ces modèles sont des bruits blancs.

Le test d’Hétéroscédasticité sert quant à lui à vérifier si la variance d’une série temporelle est constante ou si elle dépend du temps. Cette hypothèse intervient dans la vérification de la stationnarité du processus. D’après le tableau, on peut conclure la présence d’une hétéroscédasticité au sein de processus de consommation d’électricité associé aux foyer 1 et 2. Néanmoins, on observe que la pvaleur du test d’hétéroscédasticité est supérieure à 5% donc, la variance de la consommation d’électricité est stable au sein du foyer 5.

Enfin, le test de Jarque-Bera vise à s’assurer que les résidus du modèle sont distribués selon une loi normale. Comme tous les foyers une pvaleur inférieure à 5%, on peut conclure que les résidus sont non normaux.

### Résultats et Discussion

Dans cette étude, une méthode de validation du modèle a été utilisée. Cette méthode consiste à séparer les jeux de données en deux parties. La première partie est constituée à partir 80% du jeu de données brute. Cette partie appelée base de données d’entraînement sert à entraîner les modèles de séries temporelles ARIMA. La deuxième partie quant à elle représente les 20% des données restantes. Elle est constituée à partir de la première observation après la dernière observation de la base de données d’entrainement jusqu’à la fin de la base de données.

A la suite de cela, une évaluation sera effectuée entre les valeurs prédites du modèles et les données réelles regroupées dans la base de données test. Ainsi, pour comparer les performances des modèles ARIMA, le choix s’est porté sur les critères de la log-vraisemblance, l’AIC, le BIC, le MAE, le MAPE et le RMSE.

|  |  |  |  |
| --- | --- | --- | --- |
| Métriques | Foyer 1 | Foyer 2 | Foyer 5 |
| Log Likelihood | -84486.127 | -18555.961 | -18415.439 |
| AIC | 168978.253 | 37131.923 | 36834.879 |
| BIC | 169000.576 | 37190.955 | 36846.557 |
| MAE | 193.888374 | 175.349217 | 306.666331 |
| MAPE | 0.594796 | 0.784164 | 0.596263 |
| RMSE | 276.896095 | 254.146390 | 435.986433 |

Tableau : Performances prédictives des modèles ARIMA

Avant d’interpréter des résultats des modèles, il parait important de rappeler définir les différentes métriques et de souligner leurs différences.

Log-Likelihood, l’AIC, et le BIC sont des critères d’information qui permettent de comparer la qualité des modèles. Plus la log-vraisemblance est élevée meilleur est l’ajustement. Les critères AIC et BIC servent à pénaliser la complexité du modèle. Le but étant d’obtenir des modèles avec les plus efficaces possibles avec le faible nombre de paramètres.

L’Erreur Absolue Moyenne traduis en anglais (MAE) est indicateur qui est utilisé pour évaluer les performances prédictives d’un modèle. Il se base sur la moyenne des erreurs absolues entre les valeurs observées et les valeurs prédites, sans tenir compte du signe de l'erreur. Dans le cadre des analyse de séries temporelles il est très utile pour obtenir une vision globale de la précision des prévisions.

Dans notre jeu de données, c’est le modèle développé à partir des observations du foyer 2 qui présente les meilleurs résultats. En effet, ce dernier, a obtenu un MAE de 175.35 contrairement aux foyers 1et 5 qui ont obtenu respectivement des MAE qui valent 306.67 et 193.89

Au regard maintenant du pourcentage d’erreur moyen (MAPE), c’est le foyer 1 qui a obtenu un meilleur score. Autrement dit, les prévisions du foyer 1 présentent en moyenne des erreurs de l’ordre de 60%.

Enfin, le RMSE, l’erreur quadratique moyenne est la mesure qui est la plus sensible aux grandes erreurs de prédiction. Concernant, cette métrique, c’est le modèle entrainé à partir des données du foyer 2 qui performe le mieux.

En définitive, le modèle ARIMA a obtenu des performances assez faibles et hétérogènes en matière de prévision de la consommation

## Réseau Bayésien Dynamique

Les Réseaux Bayésiens Dynamiques sont des variantes des Réseaux Bayésiens classiques. Un réseau bayésien est un modèle graphique probabiliste qui, à partir de variables aléatoires structurées en un graphe orienté acyclique, permet de calculer des probabilités conditionnelles liées à ces variables. Les réseaux bayésiens dynamiques étendent ce processus en prenant en compte l'évolution des variables aléatoires, généralement dans le temps. Ces graphes probabilistes sont composés :

* Des nœuds qui représentent des variables aléatoires qui peuvent être observables ou latentes.
* Des arcs qui sont associées aux dépendances temporelles d’une variable à l’instant t (intra-slice arcs) ou à l’instant t et t+1 (inter-slice arcs).

Dans cet ouvrage, chaque nœud au sein du réseau est modélisé par une variable aléatoire continue suivant une distribution normale. Il s’agit ainsi d’un Réseau Bayésien Dynamique Gaussien.

### Préparation de la base de données

Avant le déploiement du modèle probabiliste, l’une des étape consistait à préparer les données. Cette phase de préparation s’est déroulée en une succession d’opérations qui se présente comme suit : La transformation des variables périodiques discrètes en variables périodiques continues, La suppression de toutes variables discrètes (unix\_year, unix\_month, unix\_day, unix\_hour, unix\_minute et unix\_second), La normalisation des variables associées à la puissance apparente et à la tension. Et enfin, à division de la base de données brute en base de données d’entrainement (80%) et de base de données test (20%).

### Apprentissage du Réseau Bayésien Dynamique

Une image contenant lune, cercle, sphère, nuit

Le contenu généré par l’IA peut être incorrect.L’apprentissage du réseau bayésien a effectué en utilisant comme le Critère d’information Bayésien (BIC) comme critère de score. Ce critère permet d’obtenir une architecture parcimonieuse au sens où elle assure une efficacité en pénalisant les structures trop complexes. La structure retenue dans le cadre de cette étude est représentée ci-dessous. Cette dernière est composée de 9 nœuds qui constituent ici les variables explicatives. Ces variables sont composées des valeurs retards à l’ordre de la puissance active, de la puissance apparente, de la tension, du mois et de l’heure de la date d’enregistrement transformées en variables continues.

Figure : Architecture du Réseau Bayésien Dynamique

### Estimation des paramètres du modèles

|  |  |  |  |
| --- | --- | --- | --- |
| Nœud | Parents | Coefficients | Écart-type |
| Active\_power | – | -0.0518 | 0.9624 |
| Apparent\_power | Intercept Active\_power | 0.0015 0.9947 | 0.1223 |
| Main\_rms\_voltage | Intercept Active\_power Apparent\_power | 0.0093 -0.7883 0.5159 | 0.9514 |
| Active\_power\_lag1 | Intercept Active\_power Apparent\_power Main\_rms\_voltage | 312.2728 -255.1865 376.0574 -27.0074 | 192.0572 |
| Active\_power\_lag2 | Intercept Active\_power\_lag1 Hour\_sin | 161.7154 0.4717 -58.6096 | 194.7433 |
| Hour\_sin | Intercept Active\_power Apparent\_power Main\_rms\_voltage Active\_power\_lag1 Hour\_cos | 0.1755 -0.3697 0.3280 0.2217 -0.0006 -0.0570 | 0.6374 |
| Hour\_cos | Intercept Active\_power Apparent\_power Active\_power\_lag1 | 0.0263 0.2043 -0.3534 -0.0001 | 0.6884 |
| Month\_sin | Intercept Active\_power Main\_rms\_voltage Active\_power\_lag2 | -0.0322 0.0527 0.0549 0.0002 | 0.6872 |
| Month\_cos | Intercept Apparent\_power Main\_rms\_voltage Active\_power\_lag2 | -0.2876 0.0682 0.0454 0.0003 | 0.6827 |

Tableau : Estimation des paramètres du réseaux bayésien dynamique : Foyer 1

|  |  |  |  |
| --- | --- | --- | --- |
| Nœud | Parents | Coefficients | Écart-type |
| Active\_power | – | 0.0206 | 1.0060 |
| Apparent\_power | Intercept Active\_power Hour\_cos Month\_sin | -0.0027 0.9945 -0.0188 0.0262 | 0.0774 |
| Main\_rms\_voltage | Intercept Active\_power Hour\_sin Month\_sin | -0.1181 -0.3665 -0.2965 0.4394 | 0.8934 |
| Active\_power\_lag1 | Intercept Apparent\_power Hour\_cos | 301.7256 121.6126 19.6213 | 232.2229 |
| Active\_power\_lag2 | Intercept Main\_rms\_voltage Active\_power\_lag1 Hour\_sin Hour\_cos | 196.3457 21.0507 0.3615 -92.3134 36.6699 | 222.0166 |
| Hour\_sin | Intercept Active\_power Apparent\_power Active\_power\_lag1 | 0.2515 -0.7082 0.5232 -0.0008 | 0.6187 |
| Hour\_cos | Intercept Active\_power | 0.0050 -0.0811 | 0.7028 |
| Month\_sin | Intercept | 0.0529 | 0.5276 |
| Month\_cos | Intercept Apparent\_power Main\_rms\_voltage Active\_power\_lag2 Hour\_sin | -0.8365 0.0327 0.0385 0.0001 0.0280 | 0.1968 |

Tableau : Estimation des paramètres du réseaux bayésien dynamique : Foyer 2

|  |  |  |  |
| --- | --- | --- | --- |
| Nœud | Parents | Coefficients | Écart-type |
| Active\_power | Intercept Main\_rms\_voltage Active\_power\_lag1 Hour\_cos | -0.8509 -0.0939 0.0015 -0.0783 | 0.7866 |
| Apparent\_power | Intercept Active\_power Active\_power\_lag1 Hour\_sin Hour\_cos Month\_sin Month\_cos | -0.0179 0.9981 -0.00001 -0.0088 -0.0174 -0.0433 -0.0115 | 0.0488 |
| Main\_rms\_voltage | Intercept Hour\_cos Month\_sin Month\_cos | -0.4520 0.5276 -0.5743 0.4823 | 0.8327 |
| Active\_power\_lag1 | Intercept Month\_cos | 545.7480 -67.1559 | 384.0384 |
| Active\_power\_lag2 | Intercept Active\_power\_lag1 Hour\_sin Hour\_cos | 285.8123 0.4986 -98.8183 63.4435 | 305.8126 |
| Hour\_sin | Intercept Active\_power Main\_rms\_voltage Active\_power\_lag1 Hour\_cos | 0.2394 -0.1954 -0.1275 -0.0005 0.0532 | 0.6214 |
| Hour\_cos | Intercept | -0.0003 | 0.7077 |
| Month\_sin | Intercept | -0.7867 | 0.2153 |
| Month\_cos | Intercept Month\_sin | -1.6196 -1.6124 | 0.3019 |

Tableau : Estimation des paramètres du réseaux bayésien dynamique : Foyer 5

Les paramètres de ces réseaux bayésiens s’interprètent de façon très particulière. Ils permettent de mettre en évidence l’influence probabiliste entre les variables du système électrique, telles que la puissance active, la puissance apparente, la tension, ainsi que des composantes temporelles comme l’heure ou le mois.

Pour illustrer ces propos, on a utilisé le nœud « apparent\_power » du foyer 5 dont les arrêts sont reliés à 6 autres variables :

Une image contenant croquis, dessin, diagramme, cercle

Le contenu généré par l’IA peut être incorrect.

Figure : Schéma du nœud « apparent power » du foyer 5

Comme on peut le constater la coefficient associé à la puissance active vaut 0.9981 ce qui est très proche de 1. Par conséquent, on peut conclure la puissance apparent peut-être entièrement modélisée par la puissance active. Cette observation théorique est même physiquement avérée contenu de la relation en ces deux grandeurs.

### Résultats et Discussion

|  |  |  |  |
| --- | --- | --- | --- |
| Métrique | Foyer 1 | Foyer 2 | Foyer 5 |
| Log-Likehood | -245947 | -48271.58 | -44230.11 |
| AIC | 491974 | 96619.16 | 88538.22 |
| BIC | 492271.6 | 492271.6 | 88765.92 |
| MAE | 0.8011749 | 0.8011749 | 0.5263825 |
| MAPE | 211.6339 | 211.6339 | 128.8375 |
| RMSE | 1.144049 | 1.144049 | 0.891147 |

Tableau : Performances des réseaux bayésiens dynamiques

L’évaluation prédictive des réseaux bayésiens dynamique indiquent une meilleure performance du modèle appliqué au foyer 5, comparativement aux foyers 1 et 2. En effet, les indicateurs d’ajustement statistique tels que la log-vraisemblance, l’AIC et le BIC présentent des valeurs plus faibles pour le foyer 5, traduisant une meilleure adéquation du modèle aux données observées.

Sur le plan prédictif, les métriques d’erreur telles que le MAE, le MAPE et le RMSE confirment cette tendance, avec des erreurs significativement plus faibles pour le foyer 5, indiquant une précision accrue des prédictions. Ces résultats suggèrent que la structure de consommation du foyer 5 est plus régulière ou mieux captée par le modèle bayésien, ce qui facilite son apprentissage et améliore ses performances.

# CONCLUSION

En somme, la prévision de la consommation d’énergie est l’une des méthodes les plus privilégiées pour atteindre l’efficience énergétique. Cette étude a permis de montrer qu’en raison de son caractère non linéaire et de sa nature multifactorielle, plusieurs modèles mathématiques ont été développés. Parmi ces modèles les plus populaires sont les méthodes d’analyse de données qui regroupent les modèles de régression linéaire, les modèles de régression non linéaires, les séries temporelles, les méthodes d’apprentissage automatiques, les méthodes d’apprentissage profond et enfin les méthodes hybrides.

Ce travail s’est appuyé sur cette dernière catégorie de modèle en utilisant notamment les réseaux bayésiens dynamiques. Les résultats ont montré qu’en termes de log-vraisemblance, le modèle ARIMA doit être préféré à celui du réseau probabiliste. Cependant, en ce qui concerne les prévisions, les erreurs sont significativement plus faibles avec le modèle bayésien, notamment pour le foyer 5 où la précision des prévisions est notablement améliorée, comme en témoigne le MAPE réduit à 128,8375 contre 211,6339 pour ARIMA.

Ainsi, le modèle bayésien semble donc plus robuste et précis pour les prévisions de consommation d’énergie, en particulier pour les foyers avec des comportements de consommation plus réguliers et moins influencés par des facteurs externes complexes. En revanche, le modèle ARIMA, bien qu'efficace pour capturer les tendances à court terme, semble moins performant lorsqu’il s’agit de prédire précisément la consommation sur une plus longue période ou lorsque des effets non linéaires et des dépendances temporelles complexes entrent en jeu.

Pour améliorer les prévisions de la consommation d’énergie, plusieurs pistes peuvent être envisagées. On peut citer l’intégration de nouvelles variables exogènes, le changement du critère de score utilisé dans l’apprentissage du réseau et enfin le passage vers un réseau non gaussien.

# BIBLIOGRAPHIQUE

* **Papon, P. (2024).** World Energy Outlook 2023. Paris : Agence internationale de l’énergie (AIE). Futuribles, (2), 114–117.
* **Ritchie, H., Roser, M., & Rosado, P.** (2024). Energy production and consumption. Our World in Data.
* **Amasyali, K., & El-Gohary, N. M**. (2018). A review of data-driven building energy consumption prediction studies. Renewable and Sustainable Energy Reviews, 81, 1192–1205.
* **Al-Garni, A. Z., Zubair, S. M., & Nizami, J. S**. (1994). A regression model for electric energy-consumption forecasting in Eastern Saudi Arabia. Energy, 19(10), 1043–1049.
* **Tunç, M., Çamdali, Ü., & Parmaksizoğlu, C.** (2006). Comparison of Turkey's electrical energy consumption and production with some European countries and optimization of future electrical power supply investments in Turkey. Energy Policy, 34(1), 50–59.
* **Hao, Y., Zhang, Z. Y., Liao, H., & Wei, Y. M.** (2015). China’s farewell to coal: A forecast of coal consumption through 2020. Energy Policy, 86, 444–455.
* **Chujai, P., Kerdprasop, N., & Kerdprasop, K.** (2013, March). Time series analysis of household electric consumption with ARIMA and ARMA models. In Proceedings of the International Multiconference of Engineers and Computer Scientists (Vol. 1, pp. 295–300).
* **Yasmeen, F., & Sharif, M.** (2014). Forecasting electricity consumption for Pakistan. International Journal of Emerging Technology and Advanced Engineering, 4(4), 496–503.
* **Wang, Z., Srinivasan, R., & Wang, Y.** (2016). Homogeneous ensemble model for building energy prediction: A case study using ensemble regression tree. In Proceedings of the 2016 ACEEE Summer Study on Energy Efficiency in Buildings, Pacific Grove, CA, USA, 21–26.
* **Dong, B., Cao, C., & Lee, S. E.** (2005). Applying support vector machines to predict building energy consumption in tropical region. Energy and Buildings, 37(5), 545–553.
* **Ahmad, A. S., Hassan, M. Y., Abdullah, M. P., Rahman, H. A., Hussin, F., Abdullah, H., & Saidur, R.** (2014). A review on applications of ANN and SVM for building electrical energy consumption forecasting. Renewable and Sustainable Energy Reviews, 33, 102–109.
* **Azadeh, A., Ghaderi, S. F., & Sohrabkhani, S.** (2008). Annual electricity consumption forecasting by neural network in high energy consuming industrial sectors. Energy Conversion and Management, 49(8), 2272–2278.
* **Amalou, I., Mouhni, N., & Abdali, A**. (2022). Multivariate time series prediction by RNN architectures for energy consumption forecasting. Energy Reports, 8, 1084–1091.
* **Kaytez, F., Taplamacioglu, M. C., Cam, E., & Hardalac, F.** (2015). Forecasting electricity consumption: A comparison of regression analysis, neural networks and least squares support vector machines. International Journal of Electrical Power & Energy Systems, 67, 431–438.
* **Aulia, H., Syaharuddin, S., Mandailina, V., Gervas, H. E., & Ashraf, H.** (2024). Probabilistic forecasting of energy consumption using Bayesian dynamic linear models. Aceh International Journal of Science and Technology, 13(1), 68–78.
* **Zhang, Q., Yan, H., & Liu, Y.** (2024). Power generation forecasting for solar plants based on dynamic Bayesian networks by fusing multi-source information. Renewable and Sustainable Energy Reviews, 202, 114691.
* **Singh, S., & Yassine, A**. (2018). Big data mining of energy time series for behavioral analytics and energy consumption forecasting. Energies, 11(2), 452.
* **Kelly, J., & Knottenbelt, W.** (2015). The UK-DALE dataset, domestic appliance-level electricity demand and whole-house demand from five UK homes. *Scientific data*, *2*(1), 1-14.

# ANNEXES

## Exploration des données : Graphiques

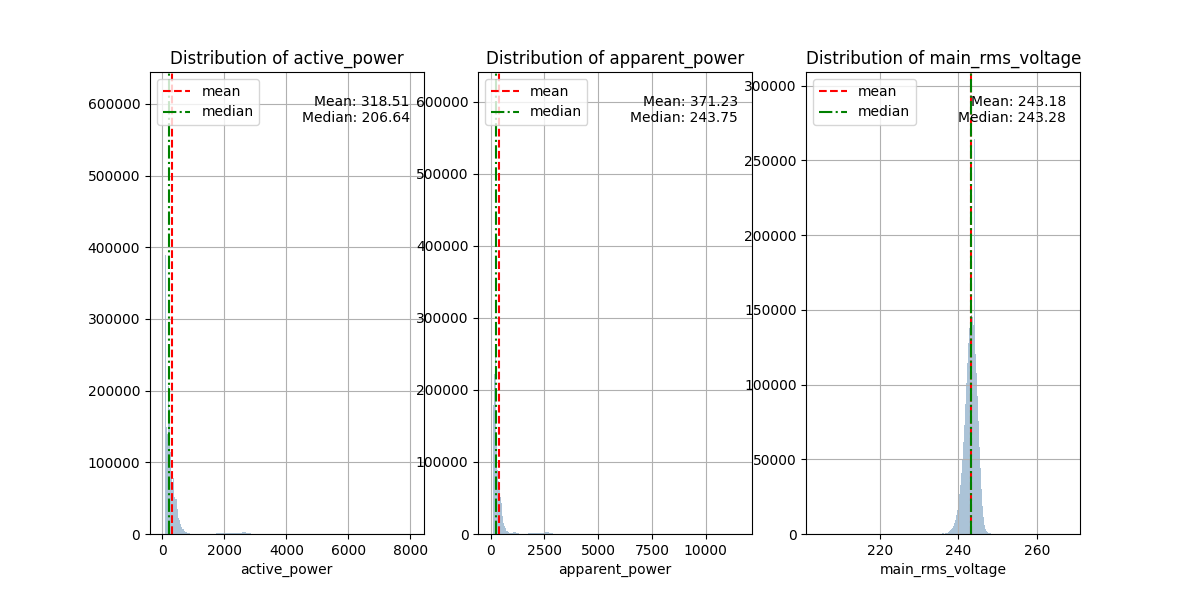


Figure  : Histogramme de la consommation d'électricité - Foyer 1

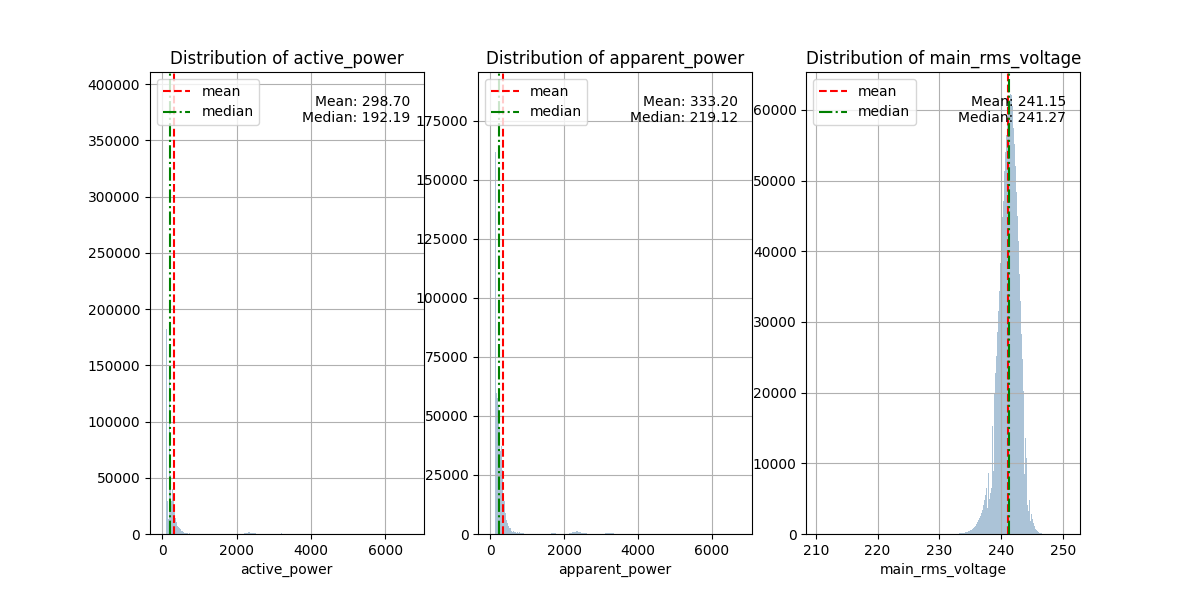


Figure : Histogramme de la consommation d'électricité - Foyer 2

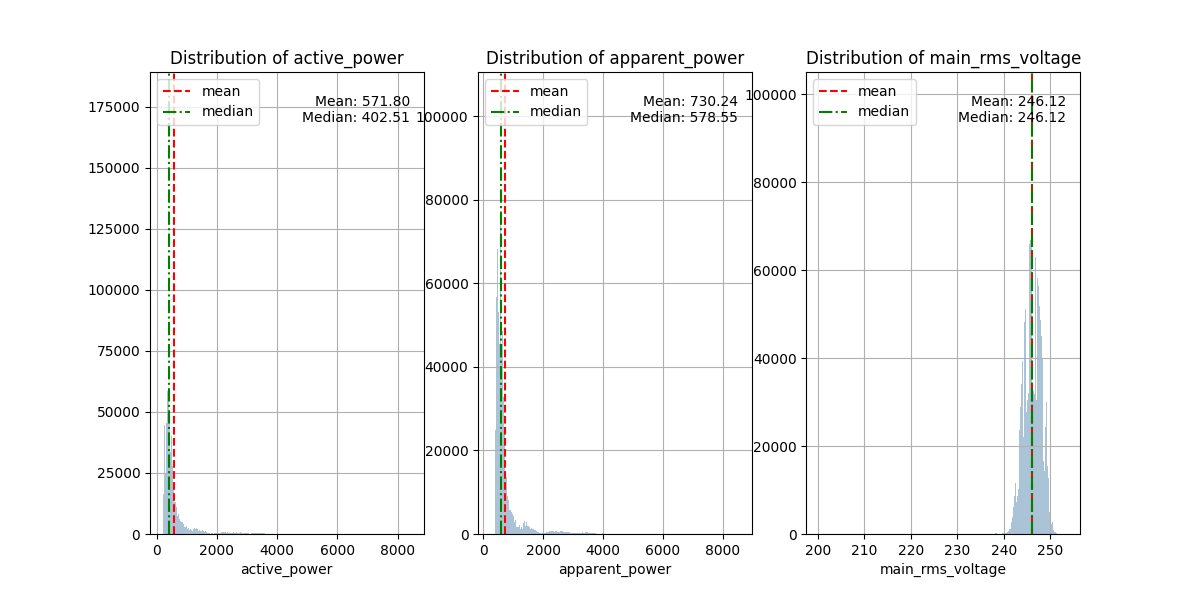


Figure : Histogramme de la consommation d'électricité - Foyer 5

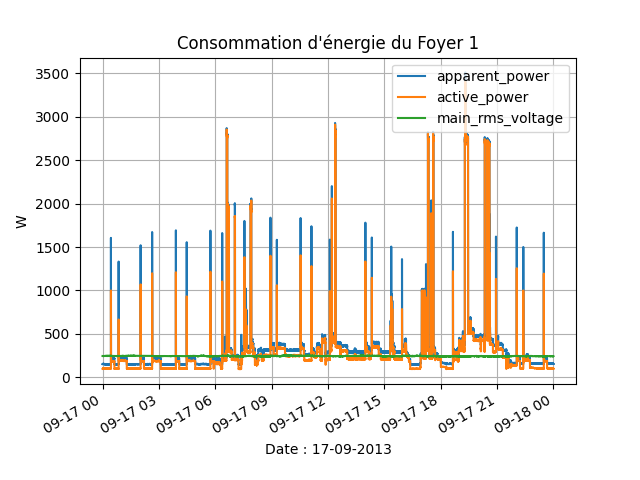


Figure : Consommation d'énergie du foyer 1 : filtré

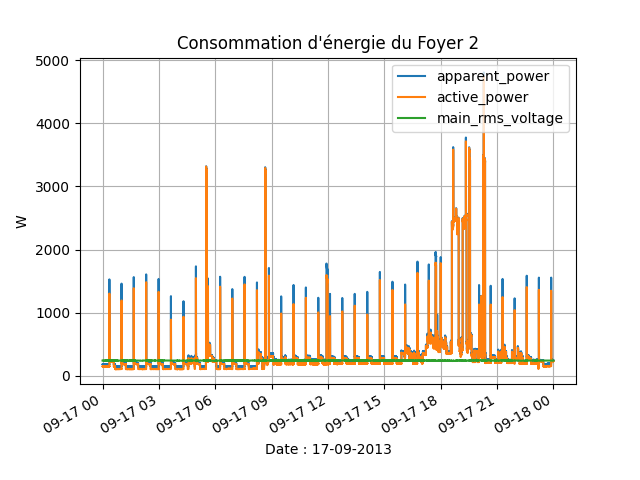


Figure : Consommation d'énergie du foyer 2 : filtré

Une image contenant texte, Tracé, diagramme, ligne

Le contenu généré par l’IA peut être incorrect.

Figure : Consommation d'énergie du foyer 5 - filtré

Une image contenant texte, capture d’écran, Police, Tracé

Le contenu généré par l’IA peut être incorrect.

Figure : Décomposition de la série - Foyer 1Une image contenant texte, capture d’écran, Police, Tracé

Le contenu généré par l’IA peut être incorrect.

Figure : Décomposition de la série - Foyer 2

Une image contenant texte, capture d’écran, diagramme, Tracé

Le contenu généré par l’IA peut être incorrect.

Figure : Décomposition de la série - Foyer 5

## PROGRAMME : Exploration des donnée

## \*\*I - Chemin vers le Google Drive\*\*

# Code pour accéder aux dossiers Google Drive

from google.colab import drive

drive.mount('/content/drive')

## \*\*II - Importation des librairies\*\*

# Importer les librairies

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

import os

## \*\*III - Paramétrages des librairies\*\*

# PARAMETRES DE PANDAS

pd.set\_option('display.max\_rows', 500)

pd.set\_option('display.max\_columns', 500)

pd.set\_option('display.width', 1000)

# PARAMETRES DE MATPLOTLIB

## Line

plt.rcParams["lines.linewidth"] = 1.5

plt.rcParams["lines.linestyle"] = "-" # https://matplotlib.org/stable/gallery/lines\_bars\_and\_markers/linestyles.html

plt.rcParams["lines.color"] = "C0"

plt.rcParams["lines.marker"] = "none"

plt.rcParams["lines.markeredgewidth"] = 1.0

plt.rcParams["lines.markersize"] = 6

plt.rcParams["lines.markerfacecolor"] = "auto"

plt.rcParams["lines.markeredgecolor"] = "auto"

## Boxplot

plt.rcParams["boxplot.notch"] = False

plt.rcParams["boxplot.vertical"] = True

plt.rcParams["boxplot.whiskers"] = 1.5

plt.rcParams["boxplot.bootstrap"] = None

plt.rcParams["boxplot.patchartist"] = False

plt.rcParams["boxplot.showmeans"] = True

plt.rcParams["boxplot.showcaps"] = True

plt.rcParams["boxplot.showbox"] = True

plt.rcParams["boxplot.showfliers"] = True

plt.rcParams["boxplot.meanline"] = True

plt.rcParams["boxplot.flierprops.color"] = "black"

plt.rcParams["boxplot.flierprops.marker"] ="o"

plt.rcParams["boxplot.flierprops.markerfacecolor"] = "none"

plt.rcParams["boxplot.flierprops.markeredgecolor"] = "black"

plt.rcParams["boxplot.flierprops.markeredgewidth"] = 1.0

plt.rcParams["boxplot.flierprops.markersize"] = 6

plt.rcParams["boxplot.flierprops.linestyle"] = "none"

plt.rcParams["boxplot.flierprops.linewidth"] = 1.0

plt.rcParams["boxplot.boxprops.color"] = "black"

plt.rcParams["boxplot.boxprops.linewidth"] = 1.0

plt.rcParams["boxplot.boxprops.linestyle"] = "-"

plt.rcParams["boxplot.whiskerprops.color"] = "black"

plt.rcParams["boxplot.whiskerprops.linewidth"] = 1.0

plt.rcParams["boxplot.whiskerprops.linestyle"] = "-"

plt.rcParams["boxplot.capprops.color"] = "black"

plt.rcParams["boxplot.capprops.linewidth"] = 1.0

plt.rcParams["boxplot.capprops.linestyle"] = "-"

plt.rcParams["boxplot.medianprops.color"] = "C1"

plt.rcParams["boxplot.medianprops.linewidth"] = 1.0

plt.rcParams["boxplot.medianprops.linestyle"] = "-"

plt.rcParams["boxplot.meanprops.color"] = "C2"

plt.rcParams["boxplot.meanprops.marker"] = "^"

plt.rcParams["boxplot.meanprops.markerfacecolor"]= "C2"

plt.rcParams["boxplot.meanprops.markeredgecolor"]= "C2"

plt.rcParams["boxplot.meanprops.markersize"]= 6

plt.rcParams["boxplot.meanprops.linestyle"]= "--"

plt.rcParams["boxplot.meanprops.linewidth"]= 1.0

## Axes

plt.rcParams["axes.facecolor"] = "white"

plt.rcParams["axes.edgecolor"] = "black"

plt.rcParams["axes.linewidth"] = 0.8

plt.rcParams["axes.grid"] = True

plt.rcParams["axes.grid.axis"] = "both"

plt.rcParams["axes.grid.which"] = "major"

plt.rcParams["axes.titlelocation"] = "center"

plt.rcParams["axes.titlesize"] ="large"

plt.rcParams["axes.titleweight"] = "normal"

plt.rcParams["axes.titlecolor"] = "auto"

plt.rcParams["axes.titley"] = None

plt.rcParams["axes.titlepad"] = 6.0

plt.rcParams["axes.labelsize"] = "medium"

plt.rcParams["axes.labelpad"] = 4.0

plt.rcParams["axes.labelweight"] = "normal"

plt.rcParams["axes.labelcolor"] = "black"

plt.rcParams["axes.axisbelow"] = "line"

## Grid

plt.rcParams["grid.color"] = "#b0b0b0"

plt.rcParams["grid.linestyle"] = "-"

plt.rcParams["grid.linewidth"] = 0.8

plt.rcParams["grid.alpha"] = 1.0

## Figure

plt.rcParams["figure.figsize"] = 6.4, 4.8

plt.rcParams["figure.dpi"] = 100

plt.rcParams["figure.facecolor"] = "white"  # Paramètre à modifier en gris

plt.rcParams["figure.edgecolor"] = "white"

## Legend

plt.rcParams["legend.loc"] = "upper right" # best

plt.rcParams["legend.frameon"] = True

plt.rcParams["legend.framealpha"] = 0.8

plt.rcParams["legend.facecolor"] = "inherit"

plt.rcParams["legend.fancybox"] = True

plt.rcParams["legend.shadow"] = False

plt.rcParams["legend.numpoints"] = 1

plt.rcParams["legend.scatterpoints"] = 1

plt.rcParams["legend.markerscale"] = 1.0

plt.rcParams["legend.fontsize"] = "medium"

## Image

plt.rcParams["image.cmap"] = "viridis" # # A colormap name (plasma, magma, etc.) https://matplotlib.org/stable/users/explain/colors/colormaps.html

## Saving figure

plt.rcParams["savefig.format"] = "png"  # {png, ps, pdf, svg}

plt.rcParams["savefig.directory"] = "~"

plt.rcParams["savefig.orientation"] = "portrait"

# PARAMETRES DE SEABORN

color\_palette = sns.color\_palette("crest", as\_cmap=True)

# PARAMETRES OS

print(os.getcwd())

os.makedirs("/content/drive/MyDrive/ColabNotebooks/1\_\_DataMining\_\_/data\_preprocessed", exist\_ok=True)

os.makedirs("/content/drive/MyDrive/ColabNotebooks/1\_\_DataMining\_\_/graphics", exist\_ok=True )

## \*\*IV - Chargemenet des bases de données\*\*

# arguments pour le chargement des données

sep\_df = "\s+" # <<\t>> dans d'autres cas.

col\_df = ["unix", "active\_power", "apparent\_power", "main\_rms\_voltage"]

# Charger de la base de données à l'aide de pandas

house\_1 = pd.read\_csv(r"/content/drive/MyDrive/ColabNotebooks/Dataset/house\_1/mains.dat", sep=sep\_df, names=col\_df)

house\_2 = pd.read\_csv(r"/content/drive/MyDrive/ColabNotebooks/Dataset/house\_2/mains.dat", sep=sep\_df, names=col\_df)

house\_5 = pd.read\_csv(r"/content/drive/MyDrive/ColabNotebooks/Dataset/house\_5/mains.dat", sep=sep\_df, names=col\_df)

# head.dataframe de house\_1\_brute : Foye 1

house\_1.head(5)

# head.dataframe de house\_2\_brute : Foyer 2

house\_2.head(5)

# head.dataframe de house\_3\_brute : Foyer 5

house\_5.head(5)

## \*\*V - Manipulation des bases de données\*\*

### A - Fonction pour créer les variables de type datetime(s)

# fonction pour créer des variables dates, Année, Mois, Jours et Séconde

def fct\_unix\_to\_date(dataframe, variable="unix", house\_i=1):

    """

    description

    ------------

    fonction qui permet de créer les variables "unix\_year", "unix\_month", "unix\_day","unix\_minute", "unix\_second", "unix\_microsecond"

    à partir de la variable unix et d'organiser les colonnes.

    arguments

    ----------

    dataframe : La base de données

    variable : La variable correspondante à l'unix stamp (1970-01-01 00:00:00 UTC)

    origin\_date : house\_1 : 2012-11-09 house\_2 2013-02-17 house\_3 2013-02-27 house\_4 : 2013-03-09 house\_5: 2014-06-29 ( format %year-%Month-%day  %hour:%minute:%second.microsecond")

    Optional Parameters

    --------------------

    #  Timestamps en fonction du premier enregistrement

     if house\_i == 1:

        origin\_date = "2012-11-09  00:00:00"

    elif house\_i ==2 :

        origin\_date = "2013-02-17  00:00:00"

    elif house\_i ==3:

        origin\_date = "2013-02-27  00:00:00"

    elif house\_i == 4:

        origin\_date = "2013-03-09  00:00:00 "

    else :

        origin\_date = "2014-06-29  00:00:00"

    # Convertir origin\_date en datetime

    origin\_datetime = pd.to\_datetime(origin\_date)

    """

    # Trier le DataFrame par la colonne UNIX

    dataframe = pd.DataFrame(dataframe)

    dataframe = dataframe.sort\_values(by=variable, ascending=True)

    New\_variable = ["unix\_year", "unix\_month", "unix\_day", "unix\_hour","unix\_minute", "unix\_second", "unix\_microsecond"]

    # Créer une variable unix au format date à partir de la variable unix

    dataframe.index = pd.to\_datetime(dataframe[variable], unit="s" ) # origin=pd.Timestamp(origin\_date) | format='%Y-%m-%d %H:%M:%S.%f

    # Créer des variables year, month, day et second

    dataframe[New\_variable[0]] = dataframe.index.year

    dataframe[New\_variable[1]] = dataframe.index.month

    dataframe[New\_variable[2]] = dataframe.index.day

    dataframe[New\_variable[3]] = dataframe.index.hour

    dataframe[New\_variable[4]] = dataframe.index.minute

    dataframe[New\_variable[5]] = dataframe.index.second

    dataframe[New\_variable[6]] = dataframe.index.microsecond

    # Réorganiser la base de données

    all\_new\_variable = ["unix", "unix\_year", "unix\_month", "unix\_day","unix\_hour","unix\_minute", "unix\_second", "unix\_microsecond", "active\_power", "apparent\_power", "main\_rms\_voltage"]

    dataframe\_ordered = dataframe[all\_new\_variable]

    return dataframe\_ordered

# Création des bases de données avec les variables unix\_date(s)

house\_1\_new = fct\_unix\_to\_date(house\_1, variable="unix", house\_i=1)

house\_2\_new = fct\_unix\_to\_date(house\_2, variable="unix", house\_i=2)

house\_5\_new = fct\_unix\_to\_date(house\_5, variable="unix", house\_i=5)

# head.dataframe de house\_1\_new : Foyer 1

print(house\_1\_new.info())

print("--------------")

house\_1\_new.head(5)

# head.dataframe de house\_2\_new : Foyer 2

print(house\_2\_new.info())

print("--------------")

house\_2\_new.head(5)

# head.dataframe de house\_5\_new : Foyer 5

print(house\_5\_new.info())

print("--------------")

house\_5\_new.head(5)

# Sauvegarde des fichiers house\_i\_new

house\_1\_new.to\_csv("/content/drive/MyDrive/ColabNotebooks/1\_\_DataMining\_\_/data\_preprocessed/house\_1\_new.csv", index=False)

house\_2\_new.to\_csv("/content/drive/MyDrive/ColabNotebooks/1\_\_DataMining\_\_/data\_preprocessed/house\_2\_new.csv", index=False)

house\_5\_new.to\_csv("/content/drive/MyDrive/ColabNotebooks/1\_\_DataMining\_\_/data\_preprocessed/house\_5\_new.csv", index=False)

### B - Traitement des valeurs nulles et des valeurs manquantes

# Vérifier si house\_1\_new contient des valeurs manquantes

print(house\_1\_new.isna().sum())

print("----------x------------")

# Vérifier si house\_1\_new contient des valeurs nulles

print(house\_1\_new.isnull().sum())

# Vérifier si house\_2\_new contient des valeurs manquantes

print(house\_2\_new.isna().sum())

print("----------x------------")

# Vérifier si house\_1\_new contient des valeurs nulles

print(house\_2\_new.isnull().sum())

# Vérifier si house\_5\_new contient des valeurs manquantes

print(house\_5\_new.isna().sum())

print("----------x------------")

# Vérifier si house\_1\_new contient des valeurs nulles

print(house\_5\_new.isnull().sum())

control\_drop = False

if control\_drop == True:

  # Suppression des valeurs manquantes

  house\_1\_new.dropna(inplace=True, axis=0)

  house\_2\_new.dropna(inplace=True, axis=0)

  house\_5\_new.dropna(inplace=True, axis=0)

### C - Fonction pour convetir les variables en float et int

# fonction pour convertir les variables en float

def fct\_convert\_to\_numeric(dataframe):

    """

    description

    ----------

    Cette fonction permet de convertir les variables unix\_type(s) en format numérique. Cela facilite la manipulation des données

    argument

    --------

    dataframe : La base de données avec les dates à jour

    """

    df = pd.DataFrame(dataframe)

    # Convertir les variables de la liste List\_col en variable numérique

    List\_col = ['unix\_year', 'unix\_month', 'unix\_day', "unix\_hour", 'unix\_minute', 'unix\_second', 'unix\_microsecond', 'active\_power', 'apparent\_power', 'main\_rms\_voltage']

    for j in List\_col:

        df[j] = pd.to\_numeric(df[j], errors="coerce", downcast="integer")

        if j in List\_col[:7]:

            df[j] = df[j].astype("int32")

            # Afficer le type de chaque variables

            print(f" \* La variable : {j} est doublement convertie en : \_\_{df[j].dtype}")

        else:

            print(f" - La variable : {j} est convertie uniquement en : \_\_{df[j].dtype}")

    return df

# Création des bases de données avec les variables numérique float et int

house\_1\_new\_numeric = fct\_convert\_to\_numeric(house\_1\_new)

print("-------------x-----------------")

house\_2\_new\_numeric = fct\_convert\_to\_numeric(house\_2\_new)

print("-------------x-----------------")

house\_5\_new\_numeric = fct\_convert\_to\_numeric(house\_5\_new)

# head.dataframe de house\_1\_new\_numeric : Foyer 1

house\_1\_new\_numeric.head(5)

# head.dataframe de house\_2\_new\_numeric : Foyer 2

house\_2\_new\_numeric.head(5)

# head.dataframe de house\_5\_new\_numeric : Foyer 5

house\_5\_new\_numeric.head(5)

# Sauvegarde des fichiers house\_i\_new\_numeric

house\_1\_new\_numeric.to\_csv("/content/drive/MyDrive/ColabNotebooks/1\_\_DataMining\_\_/data\_preprocessed/house\_1\_new\_numeric.csv", index=True)

house\_2\_new\_numeric.to\_csv("/content/drive/MyDrive/ColabNotebooks/1\_\_DataMining\_\_/data\_preprocessed/house\_2\_new\_numeric.csv", index=True)

house\_5\_new\_numeric.to\_csv("/content/drive/MyDrive/ColabNotebooks/1\_\_DataMining\_\_/data\_preprocessed/house\_5\_new\_numeric.csv", index=True)

# Fonction pour créer la variable unix\_id en fonction de la variable périodicité

def fct\_creation\_unix\_id\_var(dataframe, periodicite="unix\_day"):

    '''

    description:

    ------------

    Cette fonction sert à créer un dataframe en fonction de la fréquence choisie.

    argument:

    ---------

    dataframe : Le dataframe sélectionné.

    periodicite : La périodicité choisie pour la création du dataframe.

    option

    ------

    La fonction str\_format = '\_'.join(['{}'] \* size\_new\_list) remplace le code suivant :

    '''

    df = pd.DataFrame(dataframe)

    List\_col = df.columns.tolist()

    Index\_perdiocite = List\_col.index(periodicite)

    New\_List\_col = List\_col[1:Index\_perdiocite + 1]

    size\_new\_list = len(New\_List\_col)

    str\_format = '\_'.join(['{}'] \* size\_new\_list)

    df["unix\_id"] = df[New\_List\_col].apply(lambda row: str\_format.format(\*row), axis=1)

    return df

# Création des dataframes avec la variable unix\_id en fonction de la périodicité hour

house\_1\_new\_numeric\_preprocessed = fct\_creation\_unix\_id\_var(house\_1\_new\_numeric, periodicite="unix\_hour")

house\_2\_new\_numeric\_preprocessed = fct\_creation\_unix\_id\_var(house\_2\_new\_numeric, periodicite="unix\_hour")

house\_5\_new\_numeric\_preprocessed = fct\_creation\_unix\_id\_var(house\_5\_new\_numeric, periodicite="unix\_hour")

# head.dataframe de house\_1\_new\_numeric\_preprocessed : Foyer 1

house\_1\_new\_numeric\_preprocessed.head(5)

# head.dataframe de house\_2\_new\_numeric\_preprocessed : Foyer 2

house\_2\_new\_numeric\_preprocessed.head(5)

# head.dataframe de house\_5\_new\_numeric\_preprocessed : Foyer 5

house\_5\_new\_numeric\_preprocessed.head(5)

# Sauvegarde des fichiers house\_i\_n\_new\_numeric\_preprocesse

house\_1\_new\_numeric\_preprocessed.to\_csv("/content/drive/MyDrive/ColabNotebooks/1\_\_DataMining\_\_/data\_preprocessed/house\_1\_new\_numeric\_preprocessed.csv", index=True)

house\_2\_new\_numeric\_preprocessed.to\_csv("/content/drive/MyDrive/ColabNotebooks/1\_\_DataMining\_\_/data\_preprocessed/house\_2\_new\_numeric\_preprocessed.csv", index=True)

house\_5\_new\_numeric\_preprocessed.to\_csv("/content/drive/MyDrive/ColabNotebooks/1\_\_DataMining\_\_/data\_preprocessed/house\_5\_new\_numeric\_preprocessed.csv", index=True)

## \*\*VI - Statistiques descriptives\*\*

### A - Fonction pour dresser l'inventaire des bases de données

# L'inventaire doit se faire en utilisant la variable unix\_date et les opération pd.datetime

def fct\_inventaire\_database(dataframe, name):

    '''

    description

    -----------

    fonction pour dresser l'inventaire de la base de données. Elle permet d'obtenir un aperçu du nombre de variable, le nombre d'année étudiées...

    arguments

    ---------

    dataframe : La base de données

    name : Le nom de la base de données en format str

    '''

    df = pd.DataFrame(dataframe)

    List\_cols = ["unix\_year", "unix\_month", "unix\_day", "unix\_hour","unix\_minute","unix\_second", "unix\_microsecond"]

    start\_date = df.index.min()

    end\_date = df.index.max()

    delta = end\_date - start\_date

    days = delta.days

    seconds = delta.seconds

    years = days // 365

    months = (days % 365) // 30

    hours = seconds // 3600

    minutes = (seconds % 3600) // 60

    #   Création d'un dictionnaire contenant les informations clés

    Matrice = {

        'dataframe\_name\_': name,

        'rows\_no\_': df.shape[0],

        'cols\_no\_': df.shape[1],

        'start\_date\_': start\_date,

        'end\_date\_': end\_date,

        'years\_no\_': years,

        "mois\_no\_": months,

        'days\_no\_': days,

        'hours\_no\_': days\*24

    }

    return Matrice

# Liste contenant les dataframes et leurs noms

listes\_dataframes = [

    (house\_1\_new\_numeric\_preprocessed, 'house\_1\_new\_numeric\_preprocessed'),

    (house\_2\_new\_numeric\_preprocessed, 'house\_2\_new\_numeric\_preprocessed'),

    (house\_5\_new\_numeric\_preprocessed, 'house\_5\_new\_numeric\_preprocessed'),

]

# Création du dataframe qui contient l'inventaire des bases de données

resultats = []

for data, name in listes\_dataframes:

    resultats.append(fct\_inventaire\_database(data, name))

resultats\_df = pd.DataFrame(resultats)

resultats\_df

### B - Fonction pour résumer les statistiques descriptives

def fct\_matrice\_descriptive(dataframe, columns=["active\_power", "apparent\_power", "main\_rms\_voltage"]):

    """

    description

    -----------

    fonction pour calculer les statistiques descriptives en fonction du niveau de consommation

    arguments

    -----------

    dataframe : Base de données de type pandas

    """

    df = pd.DataFrame(dataframe)

    df\_filtred = df[columns]

    resultats = {}

    for i in df\_filtred.select\_dtypes(include=['float', 'integer']).columns:

        matrice = {

            'Max': round(df\_filtred[i].max(),2),

            'Min': round(df\_filtred[i].min(),2),

            'Moyenne': round(df\_filtred[i].mean(),2),

            'Mediane': round(df\_filtred[i].median(),2),

            'Q1': round(df\_filtred[i].quantile(0.25),2),

            'Q3': round(df\_filtred[i].quantile(0.75),2),

            'Ecart-type': round(df\_filtred[i].std(),2),

            'Variance': round(df\_filtred[i].var(),2),

            'Coefficient de variation': round(((df\_filtred[i].std()/df\_filtred[i].mean()) \* 100),2),

            'coef\_skewness': round(df\_filtred[i].skew(),2),

            'coef\_kurtosis': round(df\_filtred[i].kurtosis(),2),

            'Valeurs nulles': df\_filtred[i].isnull().sum()

        }

        resultats[i] = matrice

    resultats = pd.DataFrame(resultats)

    return resultats

# Création des dataframes pour afficher les statistiques descriptives

stats\_desc\_house\_1 = fct\_matrice\_descriptive(house\_1\_new\_numeric\_preprocessed)

stats\_desc\_house\_2 = fct\_matrice\_descriptive(house\_2\_new\_numeric\_preprocessed)

stats\_desc\_house\_5 = fct\_matrice\_descriptive(house\_5\_new\_numeric\_preprocessed)

# Statistiques descriptives de house\_1 : Foyer 1

stats\_desc\_house\_1

# Statistiques descriptives de house\_2 : Foyer 2

stats\_desc\_house\_2

# Sauvegarder les tables de statistiques descriptives

stats\_desc\_house\_1.to\_csv("/content/drive/MyDrive/ColabNotebooks/1\_\_DataMining\_\_/data\_preprocessed/house\_1\_stats\_des.csv", index=True)

stats\_desc\_house\_2.to\_csv("/content/drive/MyDrive/ColabNotebooks/1\_\_DataMining\_\_/data\_preprocessed/house\_2\_stats\_des.csv", index=True)

stats\_desc\_house\_5.to\_csv("/content/drive/MyDrive/ColabNotebooks/1\_\_DataMining\_\_/data\_preprocessed/house\_5\_stats\_des.csv", index=True)

## \*\*VII - Visualisation des données : Période Globale\*\*

### A – Plot

house\_1\_new\_numeric\_preprocessed[["active\_power"]].plot(title="Consommation d'énergie : Foyer 1")

plt.xlabel("Date")

plt.ylabel("W")

plt.legend()

plt.grid(True)

plt.savefig("/content/drive/MyDrive/ColabNotebooks/1\_\_DataMining\_\_/graphics/house\_1\_energy\_consumption\_global.png")

house\_2\_new\_numeric\_preprocessed[["active\_power"]].plot(title="Consommation d'énergie : Foyer 2")

plt.xlabel("Date")

plt.ylabel("W")

plt.legend()

plt.grid(True)

plt.savefig("/content/drive/MyDrive/ColabNotebooks/1\_\_DataMining\_\_/graphics/house\_2\_energy\_consumption\_global.png")

house\_5\_new\_numeric\_preprocessed[["active\_power"]].plot(title="Consommation d'énergie : Foyer 5")

plt.xlabel("Date")

plt.ylabel("W")

plt.legend()

plt.grid(True)

plt.savefig("/content/drive/MyDrive/ColabNotebooks/1\_\_DataMining\_\_/graphics/house\_5\_energy\_consumption\_global.png")

#### A.2 - Fonction pour regrouper les données en fonction de la périodicité par  : Heure

# Fonction pour filtrer les données à la date indiquée. ( Date de référence ~ 20-04-2014 )

def fct\_data\_groupby\_timeserie(dataframe, freq\_target="H"):

    '''

    description:

    ------------

    Cette fonction permet de grouper les données selon un critère de périodicité défini.

    argument:

    ---------

    dataframe : La base de données contenant les variables de type datetime(s) et variables explicatives.

    period\_target : Périodicité utilisée pour filtrer les données freq=["YE", "YS", "ME", "MS", "W", "D", "H", "min", "s"].

    '''

    # Création de la variable (floor) utilisée pour grouper les données

    df = pd.DataFrame(dataframe)

    df["unix\_id\_group"] = df.index.floor(freq\_target)

     # Création du dataframe groupé en fonction de la périodicité de filtre

    df\_groupby = df.groupby("unix\_id\_group").agg({

        "active\_power": "mean",

        "apparent\_power": "mean",

        "main\_rms\_voltage": "mean"

    }).reindex()

    df\_groupby.reindex(df["unix\_id\_group"].tolist())

    return df\_groupby

# Création des dataframe en utilisant la fonction pour regrouper les données en heure.

house\_1\_new\_numeric\_preprocessed\_H = fct\_data\_groupby\_timeserie(dataframe=house\_1\_new\_numeric\_preprocessed, freq\_target="H")

house\_2\_new\_numeric\_preprocessed\_H = fct\_data\_groupby\_timeserie(dataframe=house\_2\_new\_numeric\_preprocessed, freq\_target="H")

house\_5\_new\_numeric\_preprocessed\_H = fct\_data\_groupby\_timeserie(dataframe=house\_5\_new\_numeric\_preprocessed, freq\_target="H")

# head.dataframe par heure : Foyer 2

house\_1\_new\_numeric\_preprocessed\_H.head(5)

# head.dataframe par heure : Foyer 2

house\_2\_new\_numeric\_preprocessed\_H.head(5)

# head.dataframe par heure : Foyer 5

house\_5\_new\_numeric\_preprocessed\_H.head(5)

# Création des dataframes pour afficher les statistiques descriptives

stats\_desc\_house\_1\_H = fct\_matrice\_descriptive(house\_1\_new\_numeric\_preprocessed\_H)

stats\_desc\_house\_2\_H = fct\_matrice\_descriptive(house\_2\_new\_numeric\_preprocessed\_H)

stats\_desc\_house\_5\_H = fct\_matrice\_descriptive(house\_5\_new\_numeric\_preprocessed\_H)

# Statistiques descriptives en heure : Foyer 1

stats\_desc\_house\_1\_H

# Statistiques descriptives en heure : Foyer 2

stats\_desc\_house\_2\_H

# Statistiques descriptives en heure : Foyer 5

stats\_desc\_house\_5\_H

house\_1\_new\_numeric\_preprocessed\_H[["apparent\_power", "active\_power", "main\_rms\_voltage"]].plot(title="Consommation d'énergie : Foyer 1")

plt.xlabel("Date")

plt.ylabel("W")

plt.legend()

plt.grid(True)

plt.savefig("/content/drive/MyDrive/ColabNotebooks/1\_\_DataMining\_\_/graphics/house\_1\_energy\_consumption\_global\_H.png")

house\_2\_new\_numeric\_preprocessed\_H[["apparent\_power", "active\_power", "main\_rms\_voltage"]].plot(title="Consommation d'énergie : Foyer 2")

plt.xlabel("Date")

plt.ylabel("W")

plt.legend()

plt.grid(True)

plt.savefig("/content/drive/MyDrive/ColabNotebooks/1\_\_DataMining\_\_/graphics/house\_2\_energy\_consumption\_global\_H.png")

house\_5\_new\_numeric\_preprocessed[["apparent\_power", "active\_power", "main\_rms\_voltage"]].plot(title="Consommation d'énergie : Foyer 5")

plt.xlabel("Date")

plt.ylabel("W")

plt.legend()

plt.grid(True)

plt.savefig("/content/drive/MyDrive/ColabNotebooks/1\_\_DataMining\_\_/graphics/house\_5\_energy\_consumption\_global\_H.png")

### B - Boxplot

# Boxplot du Foyer 1 selon une périodicité par heure

plt.figure(figsize=(12,6))

sns.boxplot(data=house\_1\_new\_numeric\_preprocessed,

             x="unix\_hour",

             y="active\_power",

            palette="viridis")

plt.title(f"Consommation d'énergie : Foyer 1")

plt.xlabel("Date")

plt.ylabel("W")

plt.savefig("/content/drive/MyDrive/ColabNotebooks/1\_\_DataMining\_\_/graphics/house\_1\_boxplot\_global.png")

plt.grid(True)

plt.show()

# Boxplot du Foyer 2 selon une périodicité par heure

plt.figure(figsize=(12,6))

sns.boxplot(data=house\_2\_new\_numeric\_preprocessed,

             x="unix\_hour",

             y="active\_power",

             palette="viridis")

plt.title(f"Consommation d'énergie : Foyer 2")

plt.xlabel("Date")

plt.ylabel("W")

plt.savefig("/content/drive/MyDrive/ColabNotebooks/1\_\_DataMining\_\_/graphics/house\_2\_boxplot\_global.png")

plt.grid(True)

plt.show()

# Boxplot du Foyer 5 selon une périodicité par heure

plt.figure(figsize=(12,6))

sns.boxplot(data=house\_5\_new\_numeric\_preprocessed,

             x="unix\_hour",

             y="active\_power",

             palette="viridis")

plt.title(f"Consommation d'énergie : Foyer 5")

plt.xlabel("Date")

plt.ylabel("W")

plt.savefig("/content/drive/MyDrive/ColabNotebooks/1\_\_DataMining\_\_/graphics/house\_5\_boxplot\_global.png")

plt.grid(True)

plt.show()

# Histogramme de la consommation d'énergie du foyer 1

house\_1\_groupby\_viz\_filtred = house\_1\_new\_numeric\_preprocessed[["active\_power", "apparent\_power", "main\_rms\_voltage"]]

fig, axs = plt.subplots(1, 3, figsize=(12, 6))

axs = axs.ravel()

for i, colonne in enumerate(house\_1\_groupby\_viz\_filtred.select\_dtypes(include=["float", "integer"]).columns):

    sns.distplot(house\_1\_groupby\_viz\_filtred[colonne], ax=axs[i], bins='auto', hist=True, rug=False, kde=False,

                 color='#2E6A9D')

    axs[i].axvline(house\_1\_groupby\_viz\_filtred[colonne].mean(), color='red', linestyle='--', label='mean')

    axs[i].axvline(house\_1\_groupby\_viz\_filtred[colonne].median(), color='green', linestyle='-.', label='median')

    median = house\_1\_groupby\_viz\_filtred[colonne].median()

    mean = house\_1\_groupby\_viz\_filtred[colonne].mean()

    axs[i].text(0.95, 0.95, f'Mean: {mean:.2f}\nMedian: {median:.2f}', transform=axs[i].transAxes, fontsize=10,

                va='top', ha='right')

    axs[i].legend(loc='upper left')

    axs[i].set\_title(f"Distribution of {colonne}")

    axs[i].grid(True)

plt.savefig("/content/drive/MyDrive/ColabNotebooks/1\_\_DataMining\_\_/graphics/house\_1\_histplot\_global.png")

plt.tight\_layout()

plt.show()

# Histogramme de la consommation d'énergie du foyer 2

house\_2\_groupby\_viz\_filtred = house\_2\_new\_numeric\_preprocessed[["active\_power", "apparent\_power", "main\_rms\_voltage"]]

fig, axs = plt.subplots(1, 3, figsize=(12, 6))

axs = axs.ravel()

for i, colonne in enumerate(house\_2\_groupby\_viz\_filtred.select\_dtypes(include=["float", "integer"]).columns):

    sns.distplot(house\_2\_groupby\_viz\_filtred[colonne], ax=axs[i], bins='auto', hist=True, rug=False, kde=False,

                 color='#2E6A9D')

    axs[i].axvline(house\_2\_groupby\_viz\_filtred[colonne].mean(), color='red', linestyle='--', label='mean')

    axs[i].axvline(house\_2\_groupby\_viz\_filtred[colonne].median(), color='green', linestyle='-.', label='median')

    median = house\_2\_groupby\_viz\_filtred[colonne].median()

    mean = house\_2\_groupby\_viz\_filtred[colonne].mean()

    axs[i].text(0.95, 0.95, f'Mean: {mean:.2f}\nMedian: {median:.2f}', transform=axs[i].transAxes, fontsize=10,

                va='top', ha='right')

    axs[i].legend(loc='upper left')

    axs[i].set\_title(f"Distribution of {colonne}")

    axs[i].grid(True)

plt.savefig("/content/drive/MyDrive/ColabNotebooks/1\_\_DataMining\_\_/graphics/house\_2\_histplot\_global.png")

plt.tight\_layout()

plt.show()

# Histogramme de la consommation d'énergie du foyer 5

house\_5\_groupby\_viz\_filtred = house\_5\_new\_numeric\_preprocessed[["active\_power", "apparent\_power", "main\_rms\_voltage"]]

fig, axs = plt.subplots(1, 3, figsize=(12, 6))

axs = axs.ravel()

for i, colonne in enumerate(house\_5\_groupby\_viz\_filtred.select\_dtypes(include=["float", "integer"]).columns):

    sns.distplot(house\_5\_groupby\_viz\_filtred[colonne], ax=axs[i], bins='auto', hist=True, rug=False, kde=False,

                 color='#2E6A9D')

    axs[i].axvline(house\_5\_groupby\_viz\_filtred[colonne].mean(), color='red', linestyle='--', label='mean')

    axs[i].axvline(house\_5\_groupby\_viz\_filtred[colonne].median(), color='green', linestyle='-.', label='median')

    median = house\_5\_groupby\_viz\_filtred[colonne].median()

    mean = house\_5\_groupby\_viz\_filtred[colonne].mean()

    axs[i].text(0.95, 0.95, f'Mean: {mean:.2f}\nMedian: {median:.2f}', transform=axs[i].transAxes, fontsize=10,

                va='top', ha='right')

    axs[i].legend(loc='upper left')

    axs[i].set\_title(f"Distribution of {colonne}")

    axs[i].grid(True)

plt.savefig("/content/drive/MyDrive/ColabNotebooks/1\_\_DataMining\_\_/graphics/house\_5\_histplot\_global.png")

plt.tight\_layout()

plt.show()

### D - Fonction de densité

# Fonction de densité de la consommation d'énergie : Foyer 1

house\_1\_new\_numeric\_preprocessed["active\_power"].plot(title=" Densité de la consommation d'énergie : Foyer 1", kind="kde")

plt.savefig("/content/drive/MyDrive/ColabNotebooks/1\_\_DataMining\_\_/graphics/house\_1\_density\_global.png")

plt.legend()

plt.grid(True)

plt.show()

# Fonction de densité de la consommation d'énergie : Foyer 2

house\_2\_new\_numeric\_preprocessed["active\_power"].plot(title=" Densité de la consommation d'énergie : Foyer 2", kind="kde")

plt.savefig("/content/drive/MyDrive/ColabNotebooks/1\_\_DataMining\_\_/graphics/house\_2\_density\_global.png")

plt.legend()

plt.grid(True)

plt.show()

# Fonction de densité de la consommation d'énergie : Foyer 5

house\_5\_new\_numeric\_preprocessed["active\_power"].plot(title=" Densité de la consommation d'énergie : Foyer 5", kind="kde")

plt.savefig("/content/drive/MyDrive/ColabNotebooks/1\_\_DataMining\_\_/graphics/house\_5\_density\_global.png")

plt.legend()

plt.grid(True)

plt.show()

## \*\*VII - Visualisation des données : Période Spécifique\*\*

# Fonction pour filtrer les données à la date indiquée. ( Date de référence ~ 20-04-2014 )

def fct\_data\_groupby(dataframe, start\_date\_str, end\_date\_str, freq\_target="H"):

    '''

    description:

    ------------

    Cette fonction a deux fonctions. D'une part, elle permet de filtred les données selon une période de temps définie.

    D'autre part, elle sert à grouper les données selon un critère de périodicité défini.

    argument:

    ---------

    dataframe : La base de données contenant les variables de type datetime(s) et variables explicatives.

    start\_date\_str : La date du début de filtre  -- format='%Y-%m-%d %H:%M:%S.%f'.

    end\_date\_str : Date de fin du filtre -- format='%Y-%m-%d %H:%M:%S.%f'.

    period\_target : Périodicité utilisée pour filtrer les données freq=["YE", "YS", "ME", "MS", "W", "D", "H", "min", "s"].

    '''

    # Création de la liste destinée à filtrer le dataframe

    List\_freq = ["YE", "YS", "ME", "MS", "W", "D", "H", "min", "s"]

    if freq\_target in List\_freq[:2]:

        List\_col\_filtred = ["unix\_date\_filtred", "unix\_year", "active\_power", "apparent\_power", "main\_rms\_voltage"]

    elif freq\_target in List\_freq[2:4]:

        List\_col\_filtred = ["unix\_date\_filtred", "unix\_month", "active\_power", "apparent\_power", "main\_rms\_voltage"]

    elif freq\_target in List\_freq[4:6]:

        List\_col\_filtred = ["unix\_date\_filtred", "unix\_day", "active\_power", "apparent\_power", "main\_rms\_voltage"]

    elif freq\_target in List\_freq[6]:

        List\_col\_filtred = ["unix\_date\_filtred", "unix\_hour", "active\_power", "apparent\_power", "main\_rms\_voltage"]

    elif freq\_target in List\_freq[7]:

        List\_col\_filtred = ["unix\_date\_filtred", "unix\_second", "active\_power", "apparent\_power", "main\_rms\_voltage"]

     # creation du dataframe filtré en fonction de la période de filtre

    df = pd.DataFrame(dataframe)

    start\_date =  pd.to\_datetime(start\_date\_str)

    end\_date = pd.to\_datetime(end\_date\_str)

    df\_filtred = df[

        (df.index >= start\_date) &

        (df.index < end\_date)

    ]

    df\_filtred["unix\_date\_filtred"] = df\_filtred.index.floor(freq\_target)

    df\_filtred\_freq = df\_filtred[List\_col\_filtred]

     # Création du dataframe groupé en fonction de la périodicité de filtre

    df\_fitred\_groupby = df\_filtred\_freq.groupby("unix\_date\_filtred").agg({

        "active\_power": "mean",

        "apparent\_power": "mean",

        "main\_rms\_voltage": "mean"

    }).reindex() # set\_index("unix\_date\_filtred")

    #df\_filtred\_groupby\_reindex = df\_fitred\_groupby["unix\_date\_filtred"]

    return df\_filtred\_freq, df\_fitred\_groupby

# Création de dataframe pour la visualisation des données avec une période spécifique

house\_1\_filtred\_freq, house\_1\_groupby = fct\_data\_groupby(dataframe=house\_1\_new\_numeric\_preprocessed, start\_date\_str="2013-09-17 00:00:00", end\_date\_str="2013-09-18 00:00:00", freq\_target="H")

house\_2\_filtred\_freq, house\_2\_groupby = fct\_data\_groupby(dataframe=house\_2\_new\_numeric\_preprocessed, start\_date\_str="2013-09-17 00:00:00", end\_date\_str="2013-09-18 00:00:00", freq\_target="H")

house\_5\_filtred\_freq, house\_5\_groupby = fct\_data\_groupby(dataframe=house\_5\_new\_numeric\_preprocessed, start\_date\_str="2014-09-17 00:00:00", end\_date\_str="2014-09-18 00:00:00", freq\_target="H")

### A - Foyer 1

# Base de données Filtrée en fonction d'une journée : Foyer 1

house\_1\_filtred\_freq.head(5)

# Base de données Groupée en fonction d'une journée : Foyer 1

house\_1\_groupby.head(5)

### B - Foyer 2

# Base de données Filtrée en fonction d'une journée : Foyer 2

house\_2\_filtred\_freq.head(5)

# Base de données Groupée en fonction d'une journée : Foyer 2

house\_2\_groupby.head(5)

### C - Foyer 5

# Base de données Filtrée en fonction d'une journée : Foyer 5

house\_5\_filtred\_freq.head(5)

 # # Base de données Groupée en fonction d'une journée : Foyer 5

 house\_5\_groupby.head(5)

# Sauvegarder les dataset house\_i\_filtred\_freq

house\_1\_filtred\_freq.to\_csv("/content/drive/MyDrive/ColabNotebooks/1\_\_DataMining\_\_/data\_preprocessed/house\_1\_filtred\_freq.csv", index=True)

house\_2\_filtred\_freq.to\_csv("/content/drive/MyDrive/ColabNotebooks/1\_\_DataMining\_\_/data\_preprocessed/house\_2\_filtred\_freq.csv", index=True)

house\_5\_filtred\_freq.to\_csv("/content/drive/MyDrive/ColabNotebooks/1\_\_DataMining\_\_/data\_preprocessed/house\_5\_filtred\_freq.csv", index=True)

# Sauvegarder les dataset house\_i\_groupby

house\_1\_groupby.to\_csv("/content/drive/MyDrive/ColabNotebooks/1\_\_DataMining\_\_/data\_preprocessed/house\_1\_groupby.csv", index=True)

house\_2\_groupby.to\_csv("/content/drive/MyDrive/ColabNotebooks/1\_\_DataMining\_\_/data\_preprocessed/house\_2\_groupby.csv", index=True)

house\_5\_groupby.to\_csv("/content/drive/MyDrive/ColabNotebooks/1\_\_DataMining\_\_/data\_preprocessed/house\_5\_groupby.csv", index=True)

## \*\*VIII - Visualisation des données : Période Spécifique\*\*

### A - Plot

# Consommation d'énergie du 17-09-2013 : Foyer 1

house\_1\_filtred\_freq[["apparent\_power", "active\_power", "main\_rms\_voltage"]].plot(title="Consommation d'énergie du Foyer 1")

plt.xlabel("Date : 17-09-2013")

plt.ylabel("W")

plt.legend()

plt.grid(True)

plt.savefig("/content/drive/MyDrive/ColabNotebooks/1\_\_DataMining\_\_/graphics/house\_1\_energy\_consumption\_specific.png")

# Consommation d'énergie du 17-09-2013 : Foyer 2

house\_2\_filtred\_freq[["apparent\_power","active\_power", "main\_rms\_voltage"]].plot(title="Consommation d'énergie du Foyer 2")

plt.xlabel("Date : 17-09-2013")

plt.ylabel("W")

plt.legend()

plt.grid(True)

plt.savefig("/content/drive/MyDrive/ColabNotebooks/1\_\_DataMining\_\_/graphics/house\_2\_energy\_consumption\_specific.png")

# Consommation d'énergie du 17-09-2014 : Foyer 5

house\_2\_filtred\_freq[[ "apparent\_power", "active\_power", "main\_rms\_voltage"]].plot(title="Consommation d'énergie du Foyer 5")

plt.xlabel("Date : 17-09-2014")

plt.ylabel("W")

plt.legend()

plt.grid(True)

plt.savefig("/content/drive/MyDrive/ColabNotebooks/1\_\_DataMining\_\_/graphics/house\_5\_energy\_consumption\_specific.png")

### B - Boxplot

# Boxplot Consommation d'énergie du 17-09-2013 : Foyer 1

plt.figure(figsize=(12,6))

sns.boxplot(data=house\_1\_filtred\_freq,

             x="unix\_hour",

             y="active\_power")

plt.title(f"Consommation d'énergie : Foyer 1")

plt.xlabel("Date : 17-09-2013")

plt.ylabel("W")

plt.savefig("/content/drive/MyDrive/ColabNotebooks/1\_\_DataMining\_\_/graphics/house\_1\_boxplot\_specific.png")

plt.grid(True)

plt.show()

# Boxplot Consommation d'énergie du 17-09-2013 : Foyer 2

plt.figure(figsize=(12,6))

sns.boxplot(data=house\_2\_filtred\_freq,

             x="unix\_hour",

             y="active\_power")

plt.title(f"Consommation d'énergie : Foyer 2")

plt.xlabel("Date : 17-09-2013")

plt.ylabel("W")

plt.savefig("/content/drive/MyDrive/ColabNotebooks/1\_\_DataMining\_\_/graphics/house\_2\_boxplot\_specific.png")

plt.grid(True)

plt.show()

# Boxplot Consommation d'énergie du 17-09-2014 : Foyer 5

plt.figure(figsize=(12,6))

sns.boxplot(data=house\_5\_filtred\_freq,

             x="unix\_hour",

             y="active\_power")

plt.title(f"Consommation d'énergie : Foyer 5")

plt.xlabel("Date : 17-09-2014 ")

plt.ylabel("W")

plt.savefig("/content/drive/MyDrive/ColabNotebooks/1\_\_DataMining\_\_/graphics/house\_5\_boxplot\_specific.png")

plt.grid(True)

plt.show()

### C - Histogramme

# Histogramme de la consommation d'énergie du foyer 1

house\_1\_filtred\_hist\_specific = house\_1\_filtred\_freq[["active\_power", "apparent\_power", "main\_rms\_voltage"]]

fig, axs = plt.subplots(1, 3, figsize=(12, 6))

axs = axs.ravel()

for i, colonne in enumerate(house\_1\_filtred\_hist\_specific.select\_dtypes(include=["float", "integer"]).columns):

    sns.distplot(house\_1\_filtred\_hist\_specific[colonne], ax=axs[i], bins='auto', hist=True, rug=False, kde=False,

                 color='#2E6A9D')

    axs[i].axvline(house\_1\_filtred\_hist\_specific[colonne].mean(), color='red', linestyle='--', label='mean')

    axs[i].axvline(house\_1\_filtred\_hist\_specific[colonne].median(), color='green', linestyle='-.', label='median')

    median = house\_1\_filtred\_hist\_specific[colonne].median()

    mean = house\_1\_filtred\_hist\_specific[colonne].mean()

    axs[i].text(0.95, 0.95, f'Mean: {mean:.2f}\nMedian: {median:.2f}', transform=axs[i].transAxes, fontsize=10,

                va='top', ha='right')

    axs[i].legend(loc='upper left')

    axs[i].set\_title(f"Distribution of {colonne}")

    axs[i].grid(True)

plt.savefig("/content/drive/MyDrive/ColabNotebooks/1\_\_DataMining\_\_/graphics/house\_1\_histplot\_specific.png")

plt.tight\_layout()

plt.show()

# Histogramme de la consommation d'énergie du foyer 1

house\_2\_filtred\_hist\_specific = house\_2\_filtred\_freq[["active\_power", "apparent\_power", "main\_rms\_voltage"]]

fig, axs = plt.subplots(1, 3, figsize=(12, 6))

axs = axs.ravel()

for i, colonne in enumerate(house\_2\_filtred\_hist\_specific.select\_dtypes(include=["float", "integer"]).columns):

    sns.distplot(house\_2\_filtred\_hist\_specific[colonne], ax=axs[i], bins='auto', hist=True, rug=False, kde=False,

                 color='#2E6A9D')

    axs[i].axvline(house\_2\_filtred\_hist\_specific[colonne].mean(), color='red', linestyle='--', label='mean')

    axs[i].axvline(house\_2\_filtred\_hist\_specific[colonne].median(), color='green', linestyle='-.', label='median')

    median = house\_2\_filtred\_hist\_specific[colonne].median()

    mean = house\_2\_filtred\_hist\_specific[colonne].mean()

    axs[i].text(0.95, 0.95, f'Mean: {mean:.2f}\nMedian: {median:.2f}', transform=axs[i].transAxes, fontsize=10,

                va='top', ha='right')

    axs[i].legend(loc='upper left')

    axs[i].set\_title(f"Distribution of {colonne}")

    axs[i].grid(True)

plt.savefig("/content/drive/MyDrive/ColabNotebooks/1\_\_DataMining\_\_/graphics/house\_2\_histplot\_specific.png")

plt.tight\_layout()

plt.show()

# Histogramme de la consommation d'énergie du foyer 5

house\_5\_filtred\_hist\_specific = house\_5\_filtred\_freq[["active\_power", "apparent\_power", "main\_rms\_voltage"]]

fig, axs = plt.subplots(1, 3, figsize=(12, 6))

axs = axs.ravel()

for i, colonne in enumerate(house\_5\_filtred\_hist\_specific.select\_dtypes(include=["float", "integer"]).columns):

    sns.distplot(house\_5\_filtred\_hist\_specific[colonne], ax=axs[i], bins='auto', hist=True, rug=False, kde=False,

                 color='#2E6A9D')

    axs[i].axvline(house\_5\_filtred\_hist\_specific[colonne].mean(), color='red', linestyle='--', label='mean')

    axs[i].axvline(house\_5\_filtred\_hist\_specific[colonne].median(), color='green', linestyle='-.', label='median')

    median = house\_5\_filtred\_hist\_specific[colonne].median()

    mean = house\_5\_filtred\_hist\_specific[colonne].mean()

    axs[i].text(0.95, 0.95, f'Mean: {mean:.2f}\nMedian: {median:.2f}', transform=axs[i].transAxes, fontsize=10,

                va='top', ha='right')

    axs[i].legend(loc='upper left')

    axs[i].set\_title(f"Distribution of {colonne}")

    axs[i].grid(True)

plt.savefig("/content/drive/MyDrive/ColabNotebooks/1\_\_DataMining\_\_/graphics/house\_5\_histplot\_specific.png")

plt.tight\_layout()

plt.show()

### D - Fonction de densité

# Fonction de densité de la consommation d'énergie : Foyer 1

house\_1\_filtred\_freq["active\_power"].plot(title=" Densité de la consommation d'énergie : Foyer 1", kind="kde")

plt.savefig("/content/drive/MyDrive/ColabNotebooks/1\_\_DataMining\_\_/graphics/house\_1\_density\_global.png")

plt.legend()

plt.show()

# Fonction de densité de la consommation d'énergie : Foyer 2

house\_2\_filtred\_freq["active\_power"].plot(title=" Densité de la consommation d'énergie: Foyer 2", kind="kde")

plt.savefig("/content/drive/MyDrive/ColabNotebooks/1\_\_DataMining\_\_/graphics/house\_2\_density\_global.png")

plt.legend()

plt.show()

# Fonction de densité de la consommation d'énergie : Foyer 5

house\_5\_filtred\_freq["active\_power"].plot(title=" Densité de la consommation d'énergie: Foyer 5", kind="kde")

plt.savefig("/content/drive/MyDrive/ColabNotebooks/1\_\_DataMining\_\_/graphics/house\_5\_density\_global.png")

plt.legend()

plt.show()

### D - PCOLOR MESH IMAGE

### Création d'un dataframe destiné à groupé les house\_i\_grouby

var\_target = "active\_power"

data\_pcolor = pd.DataFrame({

    "unix\_index": house\_1\_groupby.index.hour,

    "house1": house\_1\_groupby[var\_target].tolist(),

    "house2": house\_2\_groupby[var\_target].tolist(),

    "house5": house\_5\_groupby[var\_target].tolist(),

})

data\_pcolor

# Sauvegarder le dataframe

data\_pcolor.to\_csv("/content/drive/MyDrive/ColabNotebooks/1\_\_DataMining\_\_/data\_preprocessed/data\_pcolor.csv", index=False)

from matplotlib.colors import LinearSegmentedColormap, Normalize

# Préparation des données pour pcolormesh

values = data\_pcolor[["house1", "house2", "house5"]].T.values  # Transpose pour avoir les maisons en ordonnées

hours = data\_pcolor["unix\_index"].values  # Heures

houses = ["House 1", "House 2", "House 5"]  # Noms des maisons

# Normalisation des valeurs

norm = Normalize(vmin=np.min(values), vmax=np.max(values))

# Création du graphique

fig, ax = plt.subplots(figsize=(16, 6))

c = ax.pcolormesh(hours, range(len(houses)), values, norm=norm, edgecolors='w', linewidth=0.5, cmap="Blues") # https://matplotlib.org/stable/users/explain/colors/colormaps.html

# Configuration des axes

ax.set\_xticks(hours + 0.5)  # Centrer les ticks

ax.set\_xticklabels(hours, rotation=0)

ax.set\_yticks(np.arange(len(houses)) + 0.5)  # Centrer les ticks

ax.set\_yticklabels(houses)

ax.set\_xlabel("Date")

ax.set\_ylabel("Maisons")

ax.set\_title("Consommation d'énergie par maison et par heure")

ax.set\_aspect('equal')

# Ajout de la barre de couleur avec ajustement

cbar = fig.colorbar(c, ax=ax, orientation='vertical', fraction=0.03, pad=0.04, shrink=0.8, label="Consommation (kWh)")

# Affichage

plt.tight\_layout()

plt.savefig("/content/drive/MyDrive/ColabNotebooks/1\_\_DataMining\_\_/graphics/pcolormesh\_specific.png")

plt.show()

## PROGRAMME : MODELE DE SERIES TEMPORELLES : ARIMA

## \*\*I - Chemin vers le Google Drive\*\*

# Code pour accéder aux dossiers Google Drive

from google.colab import drive

drive.mount('/content/drive')

## \*\*II - Importation des librairies\*\*

!pip uninstall -y numpy pmdarima

!pip install numpy==1.24.4  # Version stable compatible avec pmdarima

!pip install --no-cache-dir pmdarima

# Runtime > Restart runtime (Redémarrer le runtime)

import pmdarima as pm

# Importer les librairies

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

import os

import statsmodels

import tabulate

# Importation à partir de statsmodels

from statsmodels.tsa.stattools import adfuller, acf, pacf

from statsmodels.tsa.seasonal import seasonal\_decompose

from statsmodels.tsa.arima\_model import ARIMA

## \*\*III - Paramétrages des librairies\*\*

# Paramètres de Pandas

pd.set\_option("display.max.rows", 500)

pd.set\_option("display.max.columns", 500)

pd.set\_option("display.width", 1000)

# Paramètres de os

print(os.getcwd())

os.makedirs("/content/drive/MyDrive/ColabNotebooks/2\_\_TimeSeriesModel\_\_/TimeSeriesGraphics", exist\_ok=True)

os.makedirs("/content/drive/MyDrive/ColabNotebooks/2\_\_TimeSeriesModel\_\_/TimeSeriesTables", exist\_ok=True) # Version html ou Latex

## \*\*IV - Chargement des bases de données\*\*

# Charger de la base de données à l'aide de pandas

house\_1\_df = pd.read\_csv(r"/content/drive/MyDrive/ColabNotebooks/1\_\_DataMining\_\_/data\_preprocessed/house\_1\_new\_numeric\_preprocessed.csv")

house\_2\_df = pd.read\_csv(r"/content/drive/MyDrive/ColabNotebooks/1\_\_DataMining\_\_/data\_preprocessed/house\_2\_new\_numeric\_preprocessed.csv")

house\_5\_df = pd.read\_csv(r"/content/drive/MyDrive/ColabNotebooks/1\_\_DataMining\_\_/data\_preprocessed/house\_5\_new\_numeric\_preprocessed.csv")

# head.dataframe de house\_1\_new\_numeric\_preprocessed (house\_1\_df) : Foyer 1

house\_1\_df.head(5)

# head.dataframe de house\_2\_new\_numeric\_preprocessed (house\_2\_df) : Foyer 2

house\_2\_df.head(5)

# head.dataframe de house\_5\_new\_numeric\_preprocessed (house\_5\_df) : Foyer 5

house\_5\_df.head(5)

## \*\*V - Analyse des bases de données\*\*

### 1 - Vérifier le type des variables

# Vérifier le type de l'index house\_1\_df : Foyer 1

print(house\_1\_df.index.dtype)

print("-------------x---------------")

print(house\_1\_df.info())

# Vérifier le type de l'index house\_1\_df : Foyer 2

print(house\_2\_df.index.dtype)

print("-------------x---------------")

print(house\_2\_df.info())

# Vérifier le type de l'index house\_1\_df : Foyer 5

print(house\_5\_df.index.dtype)

print("------------- x---------------")

print(house\_5\_df.info())

### 2 - Convertir l'index en datetime

# Convertir le type de l'index en datetime

house\_1\_df.index = pd.to\_datetime(house\_1\_df["unix"])

house\_2\_df.index = pd.to\_datetime(house\_2\_df["unix"])

house\_5\_df.index = pd.to\_datetime(house\_5\_df["unix"])

# Vérifier à nouveau le type de l'index

print(house\_1\_df.index.dtype)

print("-------------x---------------")

print(house\_2\_df.index.dtype)

print("-------------x---------------")

print(house\_5\_df.index.dtype)

### 3 - Traitement des valeurs nulles et des valeurs \*\*manquantes\*\*

# Vérifier si house\_1\_new contient des valeurs manquantes : Foyer 1

print(house\_1\_df.isna().sum())

print("----------x------------")

# Vérifier si house\_1\_new contient des valeurs nulles : Foyer 1

print(house\_1\_df.isnull().sum())

# Vérifier si house\_1\_new contient des valeurs manquantes : Foyer 2

print(house\_2\_df.isna().sum())

print("----------x------------")

# Vérifier si house\_1\_new contient des valeurs nulles

print(house\_2\_df.isnull().sum())

# Vérifier si house\_1\_new contient des valeurs manquantes : Foyer 5

print(house\_5\_df.isna().sum())

print("----------x------------")

# Vérifier si house\_1\_new contient des valeurs nulles

print(house\_5\_df.isnull().sum())

# Supprimer les valeurs manquantes

control\_drop = False

if control\_drop == True :

    house\_1\_df.dropna(inplace=True)

    house\_2\_df.dropna(inplace=True)

    house\_5\_df.dropna(inplace=True)

#### 4 - Base de données groupées en Heure

# Fonction pour filtrer les données à la date indiquée. ( Date de référence ~ 20-04-2014 )

def fct\_data\_groupby\_timeserie(dataframe, freq\_target="H"):

    '''

    description:

    ------------

    Cette fonction permet de grouper les données selon un critère de périodicité défini.

    argument:

    ---------

    dataframe : La base de données contenant les variables de type datetime(s) et variables explicatives.

    period\_target : Périodicité utilisée pour filtrer les données freq=["YE", "YS", "ME", "MS", "W", "D", "H", "min", "s"].

    '''

    # Création de la variable (floor) utilisée pour grouper les données

    df = pd.DataFrame(dataframe)

    df["unix\_id\_group"] = df.index.floor(freq\_target)

     # Création du dataframe groupé en fonction de la périodicité de filtre

    df\_groupby = df.groupby("unix\_id\_group").agg({

        "active\_power": "mean",

        "apparent\_power": "mean",

        "main\_rms\_voltage": "mean"

    }).reindex()

    df\_groupby.reindex(df["unix\_id\_group"].tolist())

    return df\_groupby

# Création des dataframe house\_i\_df\_ts

house\_1\_df\_ts = fct\_data\_groupby\_timeserie(dataframe=house\_1\_df, freq\_target="H")

house\_2\_df\_ts = fct\_data\_groupby\_timeserie(dataframe=house\_2\_df, freq\_target="H")

house\_5\_df\_ts = fct\_data\_groupby\_timeserie(dataframe=house\_5\_df, freq\_target="H")

# head.dataframe pour la base de données house\_2\_df\_ts: Foyer 2

house\_1\_df\_ts.head(5)

# head.dataframe pour la base de données house\_1\_df\_ts: Foyer 1

house\_2\_df\_ts.head(5)

# head.dataframe pour la base de données house\_5\_df\_ts: Foyer 5

house\_5\_df\_ts.head(5)

# Sauvegarder les fichiers house\_i\_df\_ts

house\_1\_df\_ts.to\_csv("/content/drive/MyDrive/ColabNotebooks/2\_\_TimeSeriesModel\_\_/TimeSeriesTables/house\_1\_df\_ts.csv", index=True)

house\_2\_df\_ts.to\_csv("/content/drive/MyDrive/ColabNotebooks/2\_\_TimeSeriesModel\_\_/TimeSeriesTables/house\_2\_df\_ts.csv", index=True)

house\_5\_df\_ts.to\_csv("/content/drive/MyDrive/ColabNotebooks/2\_\_TimeSeriesModel\_\_/TimeSeriesTables/house\_5\_df\_ts.csv", index=True)

## \*\*VI - Méthodologie de BOX-JENKINS\*\*

#### 1.A - Graphique de la consommation d'énergie par foyer

# Graphique de la consommation d'énergie globale au sein des foyers

from IPython.display import Image, display

display(Image(filename=r"/content/drive/MyDrive/ColabNotebooks/1\_\_DataMining\_\_/graphics/house\_1\_energy\_consumption\_global.png"))

display(Image(filename=r"/content/drive/MyDrive/ColabNotebooks/1\_\_DataMining\_\_/graphics/house\_2\_energy\_consumption\_global.png"))

display(Image(filename=r"/content/drive/MyDrive/ColabNotebooks/1\_\_DataMining\_\_/graphics/house\_5\_energy\_consumption\_global.png"))

#### 1.B - Graphiques de la fonction d'autocorrelation et la fonction d'autocorrelation partielle

from statsmodels.graphics.tsaplots import plot\_acf, plot\_pacf

import numpy as np

# Fonction d'autocorrelation et Fonction d'autocorrelation partielle : Foyer 1

# Créer les subplots

fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(7, 6))

# Tracer l'ACF

plot\_acf(house\_1\_df\_ts['active\_power'], lags=30, zero=True, ax=ax1)

ax1.set\_title("ACF - Consommation d'énergie (W) : Foyer 1")

ax1.set\_xlabel('Lag')

ax1.set\_ylabel('Corrélation')

ax1.grid(True)

# Ajuster les graduations sur l'axe x pour l'ACF

ax1.set\_xticks(np.arange(0, 31, 1))

# Tracer le PACF

plot\_pacf(house\_1\_df\_ts['active\_power'], lags=30, zero=True, ax=ax2)

ax2.set\_title("PACF - Consommation d'énergie (W) : Foyer 1")

ax2.set\_xlabel('Lag')

ax2.set\_ylabel('Corrélation partielle')

ax2.grid(True)

# Ajuster les graduations sur l'axe x pour le PACF

ax2.set\_xticks(np.arange(0, 31, 1))

# Ajuster les subplots

plt.tight\_layout()

# Sauvegarder le fichier

#plt.savefig("/content/drive/MyDrive/ColabNotebooks/2\_\_TimeSeriesModel\_\_/TimeSeriesGraphics/house\_1\_caf\_pcaf.png")

# Afficher le graphique

plt.show()

# Fonction d'autocorrelation et Fonction d'autocorrelation partielle : Foyer 2

# Créer les subplots

fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(7, 6))

# Tracer l'ACF

plot\_acf(house\_2\_df\_ts['active\_power'], lags=30, zero=True, ax=ax1)

ax1.set\_title("ACF - Consommation d'énergie (W) : Foyer 2")

ax1.set\_xlabel('Lag')

ax1.set\_ylabel('Corrélation')

ax1.grid(True)

# Ajuster les graduations sur l'axe x pour l'ACF

ax1.set\_xticks(np.arange(0, 31, 1))

# Tracer le PACF

plot\_pacf(house\_2\_df\_ts['active\_power'], lags=30, zero=True, ax=ax2)

ax2.set\_title("PACF - Consommation d'énergie (kWh) : Foyer 2")

ax2.set\_xlabel('Lag')

ax2.set\_ylabel('Corrélation partielle')

ax2.grid(True)

# Ajuster les graduations sur l'axe x pour le PACF

ax2.set\_xticks(np.arange(0, 31, 1))

# Ajuster les subplots

plt.tight\_layout()

# Sauvegarder le fichier

#plt.savefig("/content/drive/MyDrive/ColabNotebooks/2\_\_TimeSeriesModel\_\_/TimeSeriesGraphics/house\_2\_caf\_pcaf.png")

# Afficher le graphique

plt.show()

# Fonction d'autocorrelation et Fonction d'autocorrelation partielle : Foyer 5

# Créer les subplots

fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(7, 6))

# Tracer l'ACF

plot\_acf(house\_5\_df\_ts['active\_power'], lags=30, zero=True, ax=ax1)

ax1.set\_title("ACF - Consommation d'énergie (W) : Foyer 5")

ax1.set\_xlabel('Lag')

ax1.set\_ylabel('Corrélation')

ax1.grid(True)

# Ajuster les graduations sur l'axe x pour l'ACF

ax1.set\_xticks(np.arange(0, 31, 1))

# Tracer le PACF

plot\_pacf(house\_5\_df\_ts['active\_power'], lags=30, zero=True, ax=ax2)

ax2.set\_title("PACF - Consommation d'énergie (kWh) : Foyer 5")

ax2.set\_xlabel('Lag')

ax2.set\_ylabel('Corrélation partielle')

ax2.grid(True)

# Ajuster les graduations sur l'axe x pour le PACF

ax2.set\_xticks(np.arange(0, 31, 1))

# Ajuster les subplots

plt.tight\_layout()

# Sauvegarder le fichier

#plt.savefig("/content/drive/MyDrive/ColabNotebooks/2\_\_TimeSeriesModel\_\_/TimeSeriesGraphics/house\_5\_caf\_pcaf.png")

# Afficher le graphique

plt.show()

from tabulate import tabulate

# Test de Dickey-Fuller : Foyer 1

house\_1\_adfuller = adfuller(house\_1\_df\_ts["active\_power"])

house\_1\_dickeyfuller = [

    ["Valeur de test", house\_1\_adfuller[0]],

    ["P-valeur", house\_1\_adfuller[1]],

    ["Conclusion", "La série est stationnare" if house\_1\_adfuller[1] < 0.05 else "La série est non stationnaire"]

]

# Afficher les résultats sous la forme d'un dataframe

house\_1\_dickeyfuller

# Test de Dickey-Fuller : Foyer 2

house\_2\_adfuller = adfuller(house\_2\_df\_ts["active\_power"])

house\_2\_dickeyfuller = [

    ["Valeur de test", house\_2\_adfuller[0]],

    ["P-valeur", house\_2\_adfuller[1]],

    ["Conclusion", "La série est stationnare" if house\_2\_adfuller[1] < 0.05 else "La série est non stationnaire"]

]

# Afficher les résultats sous la forme d'un dataframe

house\_2\_dickeyfuller

# Test de Dickey-Fuller : Foyer 1

house\_5\_adfuller = adfuller(house\_5\_df\_ts["active\_power"])

house\_5\_dickeyfuller = [

    ["Valeur de test", house\_5\_adfuller[0]],

    ["P-valeur", house\_5\_adfuller[1]],

    ["Conclusion", "La série est stationnare" if house\_5\_adfuller[1] < 0.05 else "La série est non stationnaire"]

]

# Afficher les résultats sous la forme d'un dataframe

house\_5\_dickeyfuller

#### 1.D - Graphique de Décomposition

# Effectuer la décomposition saisonnière : Foyer 1

decomposition = seasonal\_decompose(x=house\_1\_df\_ts['active\_power'], model='additive', period=12)

# Extraire les composantes de la décomposition

trend = decomposition.trend

seasonal = decomposition.seasonal

residual = decomposition.resid

# Afficher les composantes de la décomposition

plt.figure(figsize=(12, 8))

plt.subplot(411)

plt.plot(house\_1\_df\_ts['active\_power'], label='Série originale')

plt.legend(loc='best')

plt.subplot(412)

plt.plot(trend, label='Tendance')

plt.legend(loc='best')

plt.subplot(413)

plt.plot(seasonal, label='Saisonnalité')

plt.legend(loc='best')

plt.subplot(414)

plt.plot(residual, label='Résidus')

plt.legend(loc='best')

plt.savefig("/content/drive/MyDrive/ColabNotebooks/2\_\_TimeSeriesModel\_\_/TimeSeriesGraphics/house\_1\_decomposition.png")

plt.tight\_layout()

plt.show()

# Effectuer la décomposition saisonnière : Foyer 2

decomposition = seasonal\_decompose(x=house\_2\_df\_ts['active\_power'], model='additive', period=12)

# Extraire les composantes de la décomposition

trend = decomposition.trend

seasonal = decomposition.seasonal

residual = decomposition.resid

# Afficher les composantes de la décomposition

plt.figure(figsize=(12, 8))

plt.subplot(411)

plt.plot(house\_2\_df\_ts['active\_power'], label='Série originale')

plt.legend(loc='best')

plt.subplot(412)

plt.plot(trend, label='Tendance')

plt.legend(loc='best')

plt.subplot(413)

plt.plot(seasonal, label='Saisonnalité')

plt.legend(loc='best')

plt.subplot(414)

plt.plot(residual, label='Résidus')

plt.legend(loc='best')

plt.savefig("/content/drive/MyDrive/ColabNotebooks/2\_\_TimeSeriesModel\_\_/TimeSeriesGraphics/house\_2\_decomposition.png")

plt.tight\_layout()

plt.show()

# Effectuer la décomposition saisonnière : Foyer 5

decomposition = seasonal\_decompose(x=house\_5\_df\_ts['active\_power'], model='additive', period=12)

# Extraire les composantes de la décomposition

trend = decomposition.trend

seasonal = decomposition.seasonal

residual = decomposition.resid

# Afficher les composantes de la décomposition

plt.figure(figsize=(12, 8))

plt.subplot(411)

plt.plot(house\_5\_df\_ts['active\_power'], label='Série originale')

plt.legend(loc='best')

plt.subplot(412)

plt.plot(trend, label='Tendance')

plt.legend(loc='best')

plt.subplot(413)

plt.plot(seasonal, label='Saisonnalité')

plt.legend(loc='best')

plt.subplot(414)

plt.plot(residual, label='Résidus')

plt.legend(loc='best')

plt.savefig("/content/drive/MyDrive/ColabNotebooks/2\_\_TimeSeriesModel\_\_/TimeSeriesGraphics/house\_5\_decomposition.png")

plt.tight\_layout()

plt.show()

### 2 - Différenciation

# Différenciation pour rendre la série stationnaire : Foyer 1

differenced\_1 = house\_1\_df\_ts['active\_power'].diff().dropna()

# Afficher la série différienciée

plt.figure(figsize=(6, 4))

plt.plot(differenced\_1)

plt.title("Série temporelle différienciée (Consommation d'énergie) : Foyer 1")

plt.xlabel('Date')

plt.ylabel('Différence')

plt.grid(True)

#plt.savefig("/content/drive/MyDrive/ColabNotebooks/2\_\_TimeSeriesModel\_\_/TimeSeriesGraphics/house\_1\_differenciation.png")

plt.show()

# Différenciation pour rendre la série stationnaire : Foyer 2

differenced\_2 = house\_2\_df\_ts['active\_power'].diff().dropna()

# Afficher la série différienciée

plt.figure(figsize=(6,4 )) # 10,6

plt.plot(differenced\_2)

plt.title("Série temporelle différienciée (Consommation d'énergie) : Foyer 2")

plt.xlabel('Date')

plt.ylabel('Différence')

plt.grid(True)

#plt.savefig("/content/drive/MyDrive/ColabNotebooks/2\_\_TimeSeriesModel\_\_/TimeSeriesGraphics/house\_2\_differenciation.png")

plt.show()

# Différenciation pour rendre la série stationnaire : Foyer 5

differenced\_5 = house\_5\_df\_ts['active\_power'].diff().dropna()

# Afficher la série différienciée

plt.figure(figsize=(10, 4))

plt.plot(differenced\_5)

plt.title("Série temporelle différienciée (Consommation d'énergie) : Foyer 5")

plt.xlabel('Date')

plt.ylabel('Différence')

plt.grid(True)

#plt.savefig("/content/drive/MyDrive/ColabNotebooks/2\_\_TimeSeriesModel\_\_/TimeSeriesGraphics/house\_5\_differenciation.png")

plt.show()

### 3 - Identification des Modèles ARIMA(p,d,q)

identification = pd.DataFrame({

    "House": ["house\_1", "house\_2", "house\_5"],

    "p": [1, 2, 2],

    "d": [0, 1, 1],

    "q": [2, 1, 1]

})

identification

### 4 - Train Test Split ARIMA

#### 4.A - Création des bases de données d'entraînement et des bases de données test

# Séparer le jeu de données : house\_1 en 80% train et 20% test

house\_1\_train\_size = int(len(house\_1\_df\_ts) \* 0.8)

house\_1\_train = house\_1\_df\_ts.iloc[:house\_1\_train\_size]

house\_1\_test = house\_1\_df\_ts.iloc[house\_1\_train\_size:]

# Séparer le jeu de données : house\_1 en 80% train et 20% test

house\_2\_train\_size = int(len(house\_2\_df\_ts) \* 0.8)

house\_2\_train = house\_2\_df\_ts.iloc[:house\_2\_train\_size]

house\_2\_test = house\_2\_df\_ts.iloc[house\_2\_train\_size:]

# Séparer le jeu de données : house\_1 en 80% train et 20% test

house\_5\_train\_size = int(len(house\_5\_df\_ts) \* 0.8)

house\_5\_train = house\_5\_df\_ts.iloc[:house\_5\_train\_size]

house\_5\_test = house\_5\_df\_ts.iloc[house\_5\_train\_size:]

#### 4.B - Visualition des bases de données d'entraînement et des bases de données test

fig, ax = plt.subplots(figsize=(15, 5))

house\_1\_train["active\_power"].plot(ax=ax, label='Training Set', title='Data Train/Test Split : Foyer 1')

house\_1\_test["active\_power"].plot(ax=ax, label='Test Set')

ax.axvline(str(house\_1\_df\_ts.iloc[house\_1\_train\_size].name), color='black', ls='--')

ax.legend(['Training Set', 'Test Set'])

plt.savefig("/content/drive/MyDrive/ColabNotebooks/2\_\_TimeSeriesModel\_\_/TimeSeriesGraphics/house\_1\_splitdataset.png")

plt.grid(True)

plt.show()

fig, ax = plt.subplots(figsize=(15, 5))

house\_2\_train["active\_power"].plot(ax=ax, label='Training Set', title='Data Train/Test Split : Foyer 2')

house\_2\_test["active\_power"].plot(ax=ax, label='Test Set')

ax.axvline(str(house\_2\_df\_ts.iloc[house\_2\_train\_size].name), color='black', ls='--')

ax.legend(['Training Set', 'Test Set'])

plt.savefig("/content/drive/MyDrive/ColabNotebooks/2\_\_TimeSeriesModel\_\_/TimeSeriesGraphics/house\_2\_splitdataset.png")

plt.grid(True)

plt.show()

fig, ax = plt.subplots(figsize=(15, 5))

house\_5\_train["active\_power"].plot(ax=ax, label='Training Set', title='Data Train/Test Split : Foyer 5')

house\_5\_test["active\_power"].plot(ax=ax, label='Test Set')

ax.axvline(str(house\_5\_df\_ts.iloc[house\_5\_train\_size].name), color='black', ls='--')

ax.legend(['Training Set', 'Test Set'])

plt.savefig("/content/drive/MyDrive/ColabNotebooks/2\_\_TimeSeriesModel\_\_/TimeSeriesGraphics/house\_5\_splitdataset.png")

plt.grid(True)

plt.show()

### 5 - Estimation des modèles ARIMA(p,d,q)

#### 5.A - Tableau Statsmodels

# Importer la librairie ARIMA  à partir de statsmodels (Rappel)

from statsmodels.tsa.arima.model import ARIMA

house\_1\_model = ARIMA(house\_1\_train["active\_power"], order=identification.iloc[0,1:].tolist())

# Ajuster le modèle

house\_1\_model\_fit = house\_1\_model.fit()

# Afficher le résumé du modèle

print(house\_1\_model\_fit.summary())

house\_2\_model = ARIMA(house\_2\_train["active\_power"], order=identification.iloc[1, 1:].tolist())

# Ajuster le modèle

house\_2\_model\_fit = house\_2\_model.fit()

# Afficher le résumé du modèle

print(house\_2\_model\_fit.summary())

house\_5\_model = ARIMA(house\_5\_train["active\_power"], order=identification.iloc[2, 1:].tolist())

# Ajuster le modèle

house\_5\_model\_fit = house\_5\_model.fit()

# Afficher le résumé du modèle

print(house\_5\_model\_fit.summary())

### 6 - Prévision avec les modèles ARIMA(p,d,q)

#### 6.A - Foyer 1

# Prévision de la consommation d'énergie - Base de données d'entraînement

house\_1\_train\_prediction = house\_1\_model\_fit.predict(start=house\_1\_train.index[0], end=house\_1\_train.index[-1])

# Prévision de la consommation d'énergie - Base de données test

house\_1\_test\_prediction = house\_1\_model\_fit.forecast(steps=len(house\_1\_test))

# Mettre les données prédites de la base de données d'entraînement sous format dataframe

house\_1\_train\_prediction\_df = pd.DataFrame(data=house\_1\_train\_prediction.values , index=house\_1\_train.index, columns=["active\_power"])

# Mettre les données prédites de la base de données de test sous format dataframe

house\_1\_test\_prediction\_df = pd.DataFrame(data=house\_1\_test\_prediction.values , index=house\_1\_test.index, columns=["active\_power"])

# Sauvegarder des données prédites de la base de données d'entraînement sous format csv

house\_1\_train\_prediction\_df.to\_csv("/content/drive/MyDrive/ColabNotebooks/2\_\_TimeSeriesModel\_\_/TimeSeriesTables/house\_1\_train\_prediction\_df.csv", index=True)

# Sauvegarder des données prédites de la base de données de tes sous format csv

house\_1\_test\_prediction\_df.to\_csv("/content/drive/MyDrive/ColabNotebooks/2\_\_TimeSeriesModel\_\_/TimeSeriesTables/house\_1\_test\_predicted\_df.csv", index=True)

# Tracer les prédictions et les données réelles avec les courbes collées

plt.figure(figsize=(10, 6))

# Données d'entraînement en bleu

plt.plot(house\_1\_train["active\_power"].index, house\_1\_train["active\_power"].values, label="Ensemble d'entraînement", color="blue")

# Données de test en orange

plt.plot(house\_1\_test["active\_power"].index, house\_1\_test["active\_power"].values, label="Ensemble de test (réel)", color="orange", linestyle="--")

# Données prédites en rouge

plt.plot(house\_1\_test\_prediction\_df["active\_power"].index, house\_1\_test\_prediction.values, label="Prédictions (Test)", color="red")

# Ajouter les labels et le titre

plt.xlabel("Date")

plt.ylabel("Consommation d'énergie (hWh) : Foyer 1")

plt.title("Prédictions du modèle ARIMA(2, 1, 1) : Foyer 1 ")

# ax.axvline('2014-07-06  00:00:00', color='black', ls='--') ---------> Vérifier après.

plt.legend()

plt.grid(True)

plt.savefig("/content/drive/MyDrive/ColabNotebooks/2\_\_TimeSeriesModel\_\_/TimeSeriesGraphics/house\_1\_energy\_forecasting.png")

plt.show()

from sklearn.metrics import mean\_absolute\_error, mean\_absolute\_percentage\_error, root\_mean\_squared\_error

# Mesures de performance sur l'ensemble d'entraînement

house\_1\_train\_mae = mean\_absolute\_error(house\_1\_train["active\_power"], house\_1\_train\_prediction\_df["active\_power"])

house\_1\_train\_mape = mean\_absolute\_percentage\_error(house\_1\_train["active\_power"], house\_1\_train\_prediction\_df["active\_power"])

house\_1\_train\_rmse = root\_mean\_squared\_error(house\_1\_train["active\_power"], house\_1\_train\_prediction\_df["active\_power"])

# Mesures de performance sur l'ensemble de test

house\_1\_test\_mae = mean\_absolute\_error(house\_1\_test["active\_power"], house\_1\_test\_prediction\_df["active\_power"])

house\_1\_test\_mape = mean\_absolute\_percentage\_error(house\_1\_test["active\_power"], house\_1\_test\_prediction\_df["active\_power"])

house\_1\_test\_rmse = root\_mean\_squared\_error(house\_1\_test["active\_power"], house\_1\_test\_prediction\_df["active\_power"])

# Créer un DataFrame pour afficher les mesures de performance

house\_1\_performance\_df = pd.DataFrame({

    "Métrique": ['MAE', 'MAPE', 'RMSE'],

    "Train dataset": [house\_1\_train\_mae, house\_1\_train\_mape, house\_1\_train\_rmse],

    "Test dataset": [house\_1\_test\_mae, house\_1\_test\_mape, house\_1\_test\_rmse]

})

house\_1\_performance\_df

#### 6.B - Foyer 2

#  Prévision de la consommation d'énergie - Base de données d'entraînement

house\_2\_train\_prediction = house\_2\_model\_fit.predict(start=house\_2\_train.index[0], end=house\_2\_train.index[-1])

# Prévision de la consommation d'énergie - Base de données test

house\_2\_test\_prediction = house\_2\_model\_fit.forecast(steps=len(house\_2\_test))

# Mettre les données prédites de la base de données d'entraînement sous format dataframe

house\_2\_train\_prediction\_df = pd.DataFrame(data=house\_2\_train\_prediction.values , index=house\_2\_train.index, columns=["active\_power"])

# Mettre les données prédites de la base de données de test sous format dataframe

house\_2\_test\_prediction\_df = pd.DataFrame(data=house\_2\_test\_prediction.values , index=house\_2\_test.index, columns=["active\_power"])

# Sauvegarder des données prédites de la base de données d'entraînement sous format csv

house\_2\_train\_prediction\_df.to\_csv("/content/drive/MyDrive/ColabNotebooks/2\_\_TimeSeriesModel\_\_/TimeSeriesTables/house\_2\_train\_prediction\_df.csv", index=True)

# Sauvegarder des données prédites de la base de données de tes sous format csv

house\_2\_test\_prediction\_df.to\_csv("/content/drive/MyDrive/ColabNotebooks/2\_\_TimeSeriesModel\_\_/TimeSeriesTables/house\_2\_test\_prediction\_df.csv", index=True)

# Tracer les prédictions et les données réelles avec les courbes collées

plt.figure(figsize=(10, 6))

# Données d'entraînement en bleu

plt.plot(house\_2\_train["active\_power"].index, house\_2\_train["active\_power"].values, label="Ensemble d'entraînement", color="blue")

# Données de test en orange

plt.plot(house\_2\_test["active\_power"].index, house\_2\_test["active\_power"].values, label="Ensemble de test (réel)", color="orange", linestyle="--")

# Données prédites en rouge

plt.plot(house\_2\_test\_prediction\_df["active\_power"].index, house\_2\_test\_prediction.values, label="Prédictions (Test)", color="red")

# Ajouter les labels et le titre

plt.xlabel("Date")

plt.ylabel("Consommation d'énergie (hWh)")

plt.title("Prédictions du modèle ARIMA(2, 1, 1) : Foyer 2 ")

plt.legend()

plt.grid(True)

plt.savefig("/content/drive/MyDrive/ColabNotebooks/2\_\_TimeSeriesModel\_\_/TimeSeriesGraphics/house\_2\_energy\_forecasting.png")

plt.show()

from sklearn.metrics import mean\_absolute\_error, mean\_absolute\_percentage\_error, root\_mean\_squared\_error

# Mesures de performance sur l'ensemble d'entraînement

house\_2\_train\_mae = mean\_absolute\_error(house\_2\_train["active\_power"], house\_2\_train\_prediction\_df["active\_power"])

house\_2\_train\_mape = mean\_absolute\_percentage\_error(house\_2\_train["active\_power"], house\_2\_train\_prediction\_df["active\_power"])

house\_2\_train\_rmse = root\_mean\_squared\_error(house\_2\_train["active\_power"], house\_2\_train\_prediction\_df["active\_power"])

# Mesures de performance sur l'ensemble de test

house\_2\_test\_mae = mean\_absolute\_error(house\_2\_test["active\_power"], house\_2\_test\_prediction\_df["active\_power"])

house\_2\_test\_mape = mean\_absolute\_percentage\_error(house\_2\_test["active\_power"], house\_2\_test\_prediction\_df["active\_power"])

house\_2\_test\_rmse = root\_mean\_squared\_error(house\_2\_test["active\_power"], house\_2\_test\_prediction\_df["active\_power"])

# Créer un DataFrame pour afficher les mesures de performance

house\_2\_performance\_df = pd.DataFrame({

    "Métrique": ['MAE', 'MAPE', 'RMSE'],

    "Train dataset": [house\_2\_train\_mae, house\_2\_train\_mape, house\_2\_train\_rmse],

    "Test dataset": [house\_2\_test\_mae, house\_2\_test\_mape, house\_2\_test\_rmse]

})

house\_2\_performance\_df

#### 6.C - Foyer 5

#  Prévision de la consommation d'énergie - Base de données d'entraînement

house\_5\_train\_prediction = house\_5\_model\_fit.predict(start=house\_5\_train.index[0], end=house\_5\_train.index[-1])

# Prévision de la consommation d'énergie - Base de données test

house\_5\_test\_prediction = house\_5\_model\_fit.forecast(steps=len(house\_5\_test))

# Mettre les données prédites de la base de données d'entraînement sous format dataframe

house\_5\_train\_prediction\_df = pd.DataFrame(data=house\_5\_train\_prediction.values , index=house\_5\_train.index, columns=["active\_power"])

# Mettre les données prédites de la base de données de test sous format dataframe

house\_5\_test\_prediction\_df = pd.DataFrame(data=house\_5\_test\_prediction.values , index=house\_5\_test.index, columns=["active\_power"])

# Sauvegarder des données prédites de la base de données d'entraînement sous format csv

house\_5\_train\_prediction\_df.to\_csv("/content/drive/MyDrive/ColabNotebooks/2\_\_TimeSeriesModel\_\_/TimeSeriesTables/house\_5\_train\_prediction\_df.csv", index=True)

# Sauvegarder des données prédites de la base de données de tes sous format csv

house\_5\_test\_prediction\_df.to\_csv("/content/drive/MyDrive/ColabNotebooks/2\_\_TimeSeriesModel\_\_/TimeSeriesTables/house\_5\_test\_prediction\_df.csv", index=True)

# Tracer les prédictions et les données réelles avec les courbes collées

plt.figure(figsize=(10, 6))

# Données d'entraînement en bleu

plt.plot(house\_5\_train["active\_power"].index, house\_5\_train["active\_power"].values, label="Ensemble d'entraînement", color="blue")

# Données de test en orange

plt.plot(house\_5\_test["active\_power"].index, house\_5\_test["active\_power"].values, label="Ensemble de test (réel)", color="orange", linestyle="--")

# Données prédites en rouge

plt.plot(house\_5\_test\_prediction\_df["active\_power"].index, house\_5\_test\_prediction.values, label="Prédictions (Test)", color="red")

# Ajouter les labels et le titre

plt.xlabel("Date")

plt.ylabel("Consommation d'énergie (hWh) : Foyer 5")

plt.title("Prédictions du modèle ARIMA(2, 1, 1) : Foyer 5 ")

plt.legend()

plt.grid(True)

plt.savefig("/content/drive/MyDrive/ColabNotebooks/2\_\_TimeSeriesModel\_\_/TimeSeriesGraphics/house\_5\_energy\_forecasting.png")

plt.show()

from sklearn.metrics import mean\_absolute\_error, mean\_absolute\_percentage\_error, root\_mean\_squared\_error

# Mesures de performance sur l'ensemble d'entraînement

house\_5\_train\_mae = mean\_absolute\_error(house\_5\_train["active\_power"], house\_5\_train\_prediction\_df["active\_power"])

house\_5\_train\_mape = mean\_absolute\_percentage\_error(house\_5\_train["active\_power"], house\_5\_train\_prediction\_df["active\_power"])

house\_5\_train\_rmse = root\_mean\_squared\_error(house\_5\_train["active\_power"], house\_5\_train\_prediction\_df["active\_power"])

# Mesures de performance sur l'ensemble de test

house\_5\_test\_mae = mean\_absolute\_error(house\_5\_test["active\_power"], house\_5\_test\_prediction\_df["active\_power"])

house\_5\_test\_mape = mean\_absolute\_percentage\_error(house\_5\_test["active\_power"], house\_5\_test\_prediction\_df["active\_power"])

house\_5\_test\_rmse = root\_mean\_squared\_error(house\_5\_test["active\_power"], house\_5\_test\_prediction\_df["active\_power"])

# Créer un DataFrame pour afficher les mesures de performance

house\_5\_performance\_df = pd.DataFrame({

    "Métrique": ['MAE', 'MAPE', 'RMSE'],

    "Train dataset": [house\_5\_train\_mae, house\_5\_train\_mape, house\_5\_train\_rmse],

    "Test dataset": [house\_5\_test\_mae, house\_5\_test\_mape, house\_5\_test\_rmse]

})

house\_5\_performance\_df

#### 7 - ALGORITHME D'IDENTIFICATION DES ORDRES DU MODELE ARIMA

#### 7.A - Foyer 1

# Utiliser auto\_arima pour trouver le meilleur modèle ARIMA

house\_1\_pmdarima = pm.auto\_arima(house\_1\_train["active\_power"])

print(house\_1\_pmdarima.summary())

#### 7.B - Foyer 2

# Utiliser auto\_arima pour trouver le meilleur modèle ARIMA

house\_2\_pmdarima = pm.auto\_arima(house\_2\_train["active\_power"])

print(house\_2\_pmdarima.summary())

#### 7.C - Foyer 5

# Utiliser auto\_arima pour trouver le meilleur modèle ARIMA

house\_5\_pmdarima = pm.auto\_arima(house\_5\_train["active\_power"])

print(house\_5\_pmdarima.summary())

## PROGRAMME : RESEAUX BAYESIEN DYNAMIQUE LINEAIRE

### \*\*I - CHEMIN VERS LE GOOGLE DRIVE\*\*

# Paramètre d'installation de Google Drive à

if (file.exists("/usr/local/lib/python3.6/dist-packages/google/colab/\_ipython.py")) {

  install.packages("R.utils")

  library(R.utils)

  library(httr)

  my\_check <- function() {return(TRUE)}

  reassignInPackage("is\_interactive", pkgName = "httr", my\_check)

  options(rlang\_interactive=TRUE)

}

library(googledrive)

library(googlesheets4)

# call authentication forcing interactive login and save in cache.

drive\_auth(use\_oob = TRUE, cache = TRUE)

### \*\*II - DEFINIR L'ESPACE DE TRAVAIL\*\*

# Définir l'environnement de travail et créer les fichier

setwd

### \*\*III - INSTALLER LES LIBRAIRIES\*\*

install.packages("ggplot2")

install.packages("lubridate")

install.packages("shiny")

install.packages("bnlearn")

install.packages("data.table")

install.packages("dbnR")

install.packages("dplyr")

### \*\*IV - IMPORTER LES LIBRAIRIES\*\*

library(ggplot2)

library(lubridate)

library(shiny)

library(bnlearn)

library(data.table)

library(dbnR)

library(dplyr)

### \*\*V - CHARER LES FICHIERS\*\*

#### 5.1 - Charger les données

# Charger le fichier house\_1\_df\_ts : Foyer 1

file\_info\_h1 <- drive\_find(pattern = "house\_1\_df\_ts.csv")

file\_id\_h1 <- file\_info\_h1$id[1]

house\_1\_df\_dnb <- read.csv(drive\_download(file\_id\_h1, type = "csv", overwrite = TRUE)$local\_path)

# Charger le fichier house\_1\_df\_ts : Foyer 2

file\_info\_h2 <- drive\_find(pattern = "house\_2\_df\_ts.csv")

file\_id\_h2 <- file\_info\_h2$id[1]

house\_2\_df\_dnb <- read.csv(drive\_download(file\_id\_h2, type = "csv", overwrite = TRUE)$local\_path)

# Charger le fichier house\_1\_df\_ts : Foyer 5

file\_info\_h5 <- drive\_find(pattern = "house\_5\_df\_ts.csv")

file\_id\_h5 <- file\_info\_h5$id[1]

house\_5\_df\_dnb <- read.csv(drive\_download(file\_id\_h5, type = "csv", overwrite = TRUE)$local\_path)

#### 5.2 - Afficher les bases de données

head(house\_1\_df\_dnb, 5)

head(house\_2\_df\_dnb, 5)

head(house\_5\_df\_dnb, 5)

### \*\*VI - TRAITEMENT ET ANALYSE EXPLORATION DES DONNEES\*\*

sapply(house\_1\_df\_dnb, class)

sapply(house\_2\_df\_dnb, class)

sapply(house\_5\_df\_dnb, class)

#### 6.2 - Convertir la variable unix en variable date

# Convertir la variable unix en variable de type date : Foyer 1

house\_1\_df\_dnb$unix <- as.POSIXct(house\_1\_df\_dnb$unix)

head(house\_1\_df\_dnb, 5)

# Convertir la variable unix en variable de type date : Foyer 2

house\_2\_df\_dnb$unix <- as.POSIXct(house\_2\_df\_dnb$unix)

head(house\_2\_df\_dnb, 5)

# Convertir la variable unix en variable de type date : Foyer 5

house\_5\_df\_dnb$unix <- as.POSIXct(house\_5\_df\_dnb$unix)

head(house\_5\_df\_dnb, 5)

#### 6.3 - Création des variables Années, Mois, jour

# Création des variables explicatives temporelles : Foyer 1

house\_1\_df\_dnb <- house\_1\_df\_dnb %>%

  mutate(

    year = year(unix),

    month = month(unix),

    day = day(unix),

    hour = hour(unix)

  )

  head(house\_1\_df\_dnb, 5)

# Création des variables explicatives temporelles : Foyer 2

house\_2\_df\_dnb <- house\_2\_df\_dnb %>%

  mutate(

    year = year(unix),

    month = month(unix),

    day = day(unix),

    hour = hour(unix)

  )

  head(house\_2\_df\_dnb, 5)

# Création des variables explicatives temporelles : Foyer 5

house\_5\_df\_dnb <- house\_5\_df\_dnb %>%

  mutate(

    year = year(unix),

    month = month(unix),

    day = day(unix),

    hour = hour(unix)

  )

  head(house\_5\_df\_dnb, 5)

#### 6.3 - Encodage des variables

house\_1\_df\_dnb <- house\_1\_df\_dnb %>%

  arrange(unix) %>%

  mutate(

    active\_power\_lag1 = lag(active\_power, 1),

    active\_power\_lag2 = lag(active\_power, 2),

    hour\_sin = sin(2 \* pi \* hour / 24),

    hour\_cos = cos(2 \* pi \* hour / 24),

    month\_sin = sin(2 \* pi \* month / 12 ),

    month\_cos = cos(2 \* pi \* month / 12 )

  ) %>%

  na.omit()

head(house\_1\_df\_dnb, 5)

house\_2\_df\_dnb <- house\_2\_df\_dnb %>%

  arrange(unix) %>%

  mutate(

    active\_power\_lag1 = lag(active\_power, 1),

    active\_power\_lag2 = lag(active\_power, 2),

    hour\_sin = sin(2 \* pi \* hour / 24),

    hour\_cos = cos(2 \* pi \* hour / 24),

    month\_sin = sin(2 \* pi \* month / 12 ),

    month\_cos = cos(2 \* pi \* month / 12 )

  ) %>%

  na.omit()

head(house\_2\_df\_dnb, 5)

house\_5\_df\_dnb <- house\_5\_df\_dnb %>%

  arrange(unix) %>%

  mutate(

    active\_power\_lag1 = lag(active\_power, 1),

    active\_power\_lag2 = lag(active\_power, 2),

    hour\_sin = sin(2 \* pi \* hour / 24),

    hour\_cos = cos(2 \* pi \* hour / 24),

    month\_sin = sin(2 \* pi \* month / 12 ),

    month\_cos = cos(2 \* pi \* month / 12 )

  ) %>%

  na.omit()

head(house\_5\_df\_dnb, 5)

# Sauvegarde des fichiers

write.csv2(house\_1\_df\_dnb, file = "/content/RDataset/house\_1\_df\_dnb.csv", row.names = FALSE )

write.csv2(house\_2\_df\_dnb, file = "/content/RDataset/house\_2\_df\_dnb.csv", row.names = FALSE )

write.csv2(house\_5\_df\_dnb, file = "/content/RDataset/house\_5\_df\_dnb.csv", row.names = FALSE )

#### 6.4 - Normalisation des données

# Normalisation : Foyer 1

house\_1\_df\_dbn\_scaled <- house\_1\_df\_dnb %>%

  mutate(across(c(active\_power, apparent\_power, main\_rms\_voltage), scale))

write.csv2(house\_1\_df\_dbn\_scaled, file = "/content/RDataset/house\_1\_df\_dbn\_scaled.csv", row.names = FALSE )

# Normalisation :

house\_2\_df\_dbn\_scaled <- house\_2\_df\_dnb %>%

  mutate(across(c(active\_power, apparent\_power, main\_rms\_voltage), scale))

write.csv2(house\_2\_df\_dbn\_scaled, file = "/content/RDataset/house\_2\_df\_dbn\_scaled.csv", row.names = FALSE )

# Normalisation : Foyer 5

house\_5\_df\_dbn\_scaled <- house\_5\_df\_dnb %>%

  mutate(across(c(active\_power, apparent\_power, main\_rms\_voltage), scale))

write.csv2(house\_5\_df\_dbn\_scaled, file = "/content/RDataset/house\_5\_df\_dbn\_scaled.csv", row.names = FALSE )

### \*\*VII - TRAIN TEST SPLIT\*\*

#### 7.1 - Foyer 1

# Séparer les données en 80% des données d'entrainement et 20% des données de test

house\_1\_train\_size <- floor(0.80 \* nrow(house\_1\_df\_dbn\_scaled))

# Base de données d'entrainement : Foyer 1

house\_1\_train\_data <- house\_1\_df\_dbn\_scaled[1:house\_1\_train\_size, ]

write.csv2(house\_1\_train\_data, file = "/content/RDataset/house\_1\_train\_data.csv", row.names = FALSE )

# Base de données de test : Foyer 1

house\_1\_test\_data <- house\_1\_df\_dbn\_scaled[(house\_1\_train\_size + 1):nrow(house\_1\_df\_dbn\_scaled), ]

write.csv2(house\_1\_test\_data, file = "/content/RDataset/house\_1\_test\_data.csv", row.names = FALSE )

#### 7.2 - Foyer 2

# Séparer les données en 80% des données d'entrainement et 20% des données de test

house\_2\_train\_size <- floor(0.80 \* nrow(house\_2\_df\_dbn\_scaled))

# Base de données d'entrainement : Foyer 2

house\_2\_train\_data <- house\_2\_df\_dbn\_scaled[1:house\_2\_train\_size, ]

write.csv2(house\_2\_train\_data, file = "/content/RDataset/house\_2\_train\_data.csv", row.names = FALSE )

# Base de données de test : Foyer 2

house\_2\_test\_data <- house\_2\_df\_dbn\_scaled[(house\_2\_train\_size + 1):nrow(house\_2\_df\_dbn\_scaled), ]

write.csv2(house\_2\_test\_data, file = "/content/RDataset/house\_2\_test\_data.csv", row.names = FALSE )

#### 7.3 - Foyer 3

# Séparer les données en 80% des données d'entrainement et 20% des données de test

house\_5\_train\_size <- floor(0.80 \* nrow(house\_5\_df\_dbn\_scaled))

# Base de données d'entrainement : Foyer 5

house\_5\_train\_data <- house\_5\_df\_dbn\_scaled[1:house\_5\_train\_size, ]

#write.csv2(house\_5\_train\_data, file = "/content/RDataset/house\_5\_train\_data.csv", row.names = FALSE )

# Base de données de test : Foyer 5

house\_5\_test\_data <- house\_5\_df\_dbn\_scaled[(house\_5\_train\_size + 1):nrow(house\_5\_df\_dbn\_scaled), ]

write.csv2(house\_5\_test\_data, file = "/content/RDataset/house\_5\_test\_data.csv", row.names = FALSE )

### \*\*VIII - RESEAUX BAYESIENS DYNAMIQUES\*\*

#### 8.1 - Foyer 1

house\_1\_dbn\_learning <- house\_1\_train\_data %>%

  select(active\_power, apparent\_power, main\_rms\_voltage, active\_power\_lag1, active\_power\_lag2, hour\_sin, hour\_cos, month\_sin, month\_cos)

head(house\_1\_dbn\_learning, 5)

write.csv2(house\_1\_dbn\_learning, file = "/content/RDataset/house\_1\_dbn\_learning.csv", row.names = FALSE )

# Apprentissage de la structure avec Hill-Climbing

structure\_1 <- hc(house\_1\_dbn\_learning, score = "bic-g")

plot(structure\_1)  # Visualiser le graphe

ggsave("/content/RDataset/house\_1\_DynamicBayesianNetwork.png")

## Paramétrisation du Modèle

# Ajustement du DBN (modèle gaussien)

dbn\_model\_1 <- bn.fit(structure\_1, data = house\_1\_dbn\_learning, method = "mle-g")

dbn\_model\_1

# Résumé des paramètres

print(dbn\_model\_1$active\_power)

#### 8.2 - Foyer 2

house\_2\_dbn\_learning <- house\_2\_train\_data %>%

  select(active\_power, apparent\_power, main\_rms\_voltage, active\_power\_lag1, active\_power\_lag2, hour\_sin, hour\_cos, month\_sin, month\_cos)

head(house\_2\_dbn\_learning, 5)

write.csv2(house\_2\_dbn\_learning, file = "/content/RDataset/house\_2\_dbn\_learning.csv", row.names = FALSE )

# Apprentissage de la structure avec Hill-Climbing

structure\_2 <- hc(house\_2\_dbn\_learning, score = "bic-g")

plot(structure\_2)  # Visualiser le graphe

ggsave("/content/RDataset/house\_2\_DynamicBayesianNetwork.png")

## Paramétrisation du Modèle

# Ajustement du DBN (modèle gaussien)

dbn\_model\_2 <- bn.fit(structure\_2, data = house\_2\_dbn\_learning, method = "mle-g")

dbn\_model\_2

# Résumé des paramètres

print(dbn\_model\_2$active\_power)

#### 8.3 - Foyer 3

house\_5\_dbn\_learning <- house\_5\_train\_data %>%

  select(active\_power, apparent\_power, main\_rms\_voltage, active\_power\_lag1, active\_power\_lag2, hour\_sin, hour\_cos, month\_sin, month\_cos)

head(house\_5\_dbn\_learning, 5)

write.csv2(house\_5\_dbn\_learning, file = "/content/RDataset/house\_5\_dbn\_learning.csv", row.names = FALSE )

# Apprentissage de la structure avec Hill-Climbing

structure\_5 <- hc(house\_5\_dbn\_learning, score = "bic-g")

plot(structure\_5)  # Visualiser le graphe

ggsave("/content/RDataset/house\_5\_DynamicBayesianNetwork.png")

## Paramétrisation du Modèle

# Ajustement du DBN (modèle gaussien)

dbn\_model\_5 <- bn.fit(structure\_5, data = house\_5\_dbn\_learning, method = "mle-g")

dbn\_model\_5

### \*\*XIX - PREDICTION ET EVALUATION\*\*

#### 9.1 - Foyer 1 --------------------------------

# Préparation des données de test

house\_1\_dbn\_testing <- house\_1\_test\_data %>%

  select(active\_power, apparent\_power, main\_rms\_voltage, active\_power\_lag1, active\_power\_lag2, hour\_sin, hour\_cos, month\_sin, month\_cos)

head(house\_1\_dbn\_testing,5)

write.csv2(house\_1\_dbn\_testing, file = "/content/RDataset/house\_1\_dbn\_testing.csv", row.names = FALSE )

# Prédiction

house\_1\_dbn\_predictions <- predict(dbn\_model\_1, node = "active\_power", data = house\_1\_dbn\_testing)

write.csv2(house\_1\_dbn\_predictions, file = "/content/RDataset/house\_1\_dbn\_predictions.csv", row.names = FALSE )

head(house\_1\_dbn\_predictions, 5)

# Calcul des métriques : Foyer 1

house\_1\_dbn\_rmse <- sqrt(mean((house\_1\_dbn\_testing$active\_power - house\_1\_dbn\_predictions)^2))

house\_1\_dbn\_mae <- mean(abs(house\_1\_dbn\_testing$active\_power - house\_1\_dbn\_predictions))

house\_1\_dbn\_mape <- mean(abs((house\_1\_dbn\_testing$active\_power - house\_1\_dbn\_predictions) / house\_1\_dbn\_testing$active\_power)) \* 100 # S'assurer que les données ne contiennent pas des valeurs nulles ou manquantes

cat(

  "RMSE:",  house\_1\_dbn\_rmse,

  "MAE:", house\_1\_dbn\_mae,

  "MAPE:", house\_1\_dbn\_mape

)

# Affichage des résultats

house\_1\_dbn\_predictions\_dataframe <- data.frame(

  time = house\_1\_test\_data$unix,

  actual = house\_1\_dbn\_testing$active\_power,

  predicted = house\_1\_dbn\_predictions

)

head(house\_1\_dbn\_predictions\_dataframe, 5)

write.csv2(house\_1\_dbn\_predictions\_dataframe, file="/content/RDataset/house\_1\_dbn\_predictions\_dataframe.csv", row.names = FALSE)

# Visualisation

ggplot(house\_1\_dbn\_predictions\_dataframe, aes(x = time)) +

  geom\_line(aes(y = actual, color = "Réel")) +

  geom\_line(aes(y = predicted, color = "Prédit")) +

  labs(title = "Prédiction vs. Réel", x = "Temps", y = "Puissance Active (W)") +

  scale\_color\_manual(values = c("Réel" = "blue", "Prédit" = "red"))

ggsave("/content/RDataset/house\_1\_bdn\_actualvspredict.png")

# Log-likelihood

logLik\_value <- logLik(dbn\_model\_1, data = house\_1\_dbn\_learning)

# Nombre de paramètres (utile pour AIC/BIC)

num\_params <- nparams(dbn\_model\_1)

# Nombre d'observations

n\_obs <- nrow(house\_1\_dbn\_learning)

# AIC et BIC

aic\_value <- -2 \* logLik\_value + 2 \* num\_params

bic\_value <- -2 \* logLik\_value + log(n\_obs) \* num\_params

cat("Log-Likelihood:", logLik\_value, "/n")

cat("AIC:", aic\_value, "/n")

cat("BIC:", bic\_value, "/n")

#### 9.1 - Foyer 2

# Préparation des données de test

house\_2\_dbn\_testing <- house\_2\_test\_data %>%

  select(active\_power, apparent\_power, main\_rms\_voltage, active\_power\_lag1, active\_power\_lag2, hour\_sin, hour\_cos, month\_sin, month\_cos)

head(house\_2\_dbn\_testing,5)

write.csv2(house\_2\_dbn\_testing, file = "/content/RDataset/house\_2\_dbn\_testing.csv", row.names = FALSE )

# Prédiction

house\_2\_dbn\_predictions <- predict(dbn\_model\_2, node = "active\_power", data = house\_2\_dbn\_testing)

write.csv2(house\_2\_dbn\_predictions, file = "/content/RDataset/house\_2\_dbn\_predictions.csv", row.names = FALSE )

head(house\_2\_dbn\_predictions, 5)

# Calcul des métriques : Foyer 1

house\_2\_dbn\_rmse <- sqrt(mean((house\_2\_dbn\_testing$active\_power - house\_2\_dbn\_predictions)^2))

house\_2\_dbn\_mae <- mean(abs(house\_2\_dbn\_testing$active\_power - house\_2\_dbn\_predictions))

house\_2\_dbn\_mape <- mean(abs((house\_2\_dbn\_testing$active\_power - house\_2\_dbn\_predictions) / house\_2\_dbn\_testing$active\_power)) \* 100 # S'assurer que les données ne contiennent pas des valeurs nulles ou manquantes

cat(

  "RMSE:",  house\_2\_dbn\_rmse,

  "MAE:", house\_2\_dbn\_mae,

  "MAPE:", house\_2\_dbn\_mape

)

# Visualisation des résultats

house\_2\_dbn\_predictions\_dataframe <- data.frame(

  time = house\_2\_test\_data$unix,

  actual = house\_2\_dbn\_testing$active\_power,

  predicted = house\_2\_dbn\_predictions

)

head(house\_2\_dbn\_predictions\_dataframe, 5)

write.csv2(house\_2\_dbn\_predictions\_dataframe, file="/content/RDataset/house\_2\_dbn\_predictions\_dataframe.csv", row.names = FALSE)

# Visualisation

ggplot(house\_2\_dbn\_predictions\_dataframe, aes(x = time)) +

  geom\_line(aes(y = actual, color = "Réel")) +

  geom\_line(aes(y = predicted, color = "Prédit")) +

  labs(title = "Prédiction vs. Réel", x = "Temps", y = "Puissance Active (W)") +

  scale\_color\_manual(values = c("Réel" = "blue", "Prédit" = "red"))

ggsave("/content/RDataset/house\_2\_bdn\_actualvspredict.png")

# Log-likelihood

logLik\_value <- logLik(dbn\_model\_2, data = house\_2\_dbn\_learning)

# Nombre de paramètres (utile pour AIC/BIC)

num\_params <- nparams(dbn\_model\_2)

# Nombre d'observations

n\_obs <- nrow(house\_2\_dbn\_learning)

# AIC et BIC

aic\_value <- -2 \* logLik\_value + 2 \* num\_params

bic\_value <- -2 \* logLik\_value + log(n\_obs) \* num\_params

cat("Log-Likelihood:", logLik\_value, "/n")

cat("AIC:", aic\_value, "/n")

cat("BIC:", bic\_value, "/n")

#### 9.3 - Foyer 5

# Préparation des données de test

house\_5\_dbn\_testing <- house\_5\_test\_data %>%

  select(active\_power, apparent\_power, main\_rms\_voltage, active\_power\_lag1, active\_power\_lag2, hour\_sin, hour\_cos, month\_sin, month\_cos)

head(house\_5\_dbn\_testing,5)

write.csv2(house\_5\_dbn\_testing, file = "/content/RDataset/house\_5\_dbn\_testing.csv", row.names = FALSE )

# Prédiction

house\_5\_dbn\_predictions <- predict(dbn\_model\_5, node = "active\_power", data = house\_5\_dbn\_testing)

write.csv2(house\_5\_dbn\_predictions, file = "/content/RDataset/house\_5\_dbn\_predictions.csv", row.names = FALSE )

head(house\_5\_dbn\_predictions, 5)

# Calcul des métriques : Foyer 1

house\_5\_dbn\_rmse <- sqrt(mean((house\_5\_dbn\_testing$active\_power - house\_5\_dbn\_predictions)^2))

house\_5\_dbn\_mae <- mean(abs(house\_5\_dbn\_testing$active\_power - house\_5\_dbn\_predictions))

house\_5\_dbn\_mape <- mean(abs((house\_5\_dbn\_testing$active\_power - house\_5\_dbn\_predictions) / house\_5\_dbn\_testing$active\_power)) \* 100 # S'assurer que les données ne contiennent pas des valeurs nulles ou manquantes

cat(

  "RMSE:",  house\_5\_dbn\_rmse,

  "MAE:", house\_5\_dbn\_mae,

  "MAPE:", house\_5\_dbn\_mape

)

# Visualisation des résultats

house\_5\_dbn\_predictions\_dataframe <- data.frame(

  time = house\_5\_test\_data$unix,

  actual = house\_5\_dbn\_testing$active\_power,

  predicted = house\_5\_dbn\_predictions

)

head(house\_5\_dbn\_predictions\_dataframe, 5)

write.csv2(house\_5\_dbn\_predictions\_dataframe, file="/content/RDataset/house\_5\_dbn\_predictions\_dataframe.csv", row.names = FALSE)

# Visualisation

ggplot(house\_5\_dbn\_predictions\_dataframe, aes(x = time)) +

  geom\_line(aes(y = actual, color = "Réel")) +

  geom\_line(aes(y = predicted, color = "Prédit")) +

  labs(title = "Prédiction vs. Réel", x = "Temps", y = "Puissance Active (W)") +

  scale\_color\_manual(values = c("Réel" = "blue", "Prédit" = "red"))

ggsave("/content/RDataset/house\_5\_bdn\_actualvspredict.png")

# Log-likelihood

logLik\_value <- logLik(dbn\_model\_5, data = house\_5\_dbn\_learning)

# Nombre de paramètres (utile pour AIC/BIC)

num\_params <- nparams(dbn\_model\_5)

# Nombre d'observations

n\_obs <- nrow(house\_5\_dbn\_learning)

# AIC et BIC

aic\_value <- -2 \* logLik\_value + 2 \* num\_params

bic\_value <- -2 \* logLik\_value + log(n\_obs) \* num\_params

cat("Log-Likelihood:", logLik\_value, "/n")

cat("AIC:", aic\_value, "/n")

cat("BIC:", bic\_value, "/n")