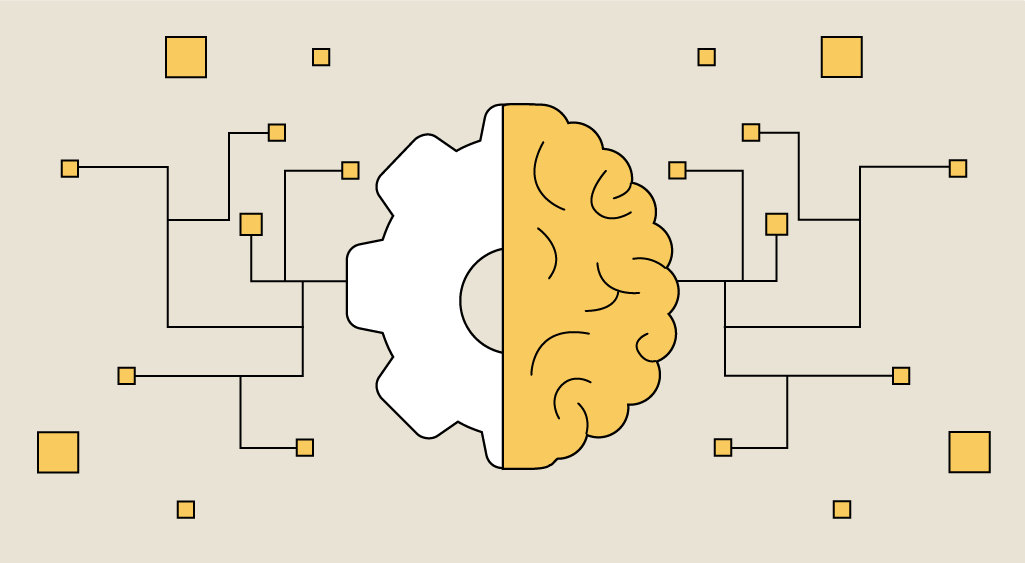
**Rapport**

**Projet Machine Learning**

*Explication du prix de l’électricité*



**Equipe:4**

SENECHAL-Morgan

SEBBANE-Ryan

SAVORY-Edwin

PIMENTA SILVA-Lionel

**Professeur :** CHAKCHOUK, Faten

**Table des matières**

[Introduction 4](#_Toc134465594)

[I. Description des données 5](#_Toc134465595)

[1) Importation des données 5](#_Toc134465596)

[2) Identification des données 5](#_Toc134465597)

[3) Importation des librairie python 7](#_Toc134465598)

[II. Préparation des données 7](#_Toc134465599)

[1) Chargement des données 7](#_Toc134465600)

[2) Valeurs manquantes 8](#_Toc134465601)

[3) Normalisation des données 9](#_Toc134465602)

[4) Comparaison des attributs 11](#_Toc134465603)

[III. Analyse exploratoire des données 12](#_Toc134465604)

[1) Identification de la variable cible 12](#_Toc134465605)

[2) Aperçus et éxamination des variables 12](#_Toc134465606)

[3) Les Histogramme 15](#_Toc134465607)

[4) Matrice des corrélations 20](#_Toc134465608)

[5) Interprétation des résultats 21](#_Toc134465609)

[IV. Modélisation des données 22](#_Toc134465610)

[V. Evaluation des modèles 22](#_Toc134465611)

[Conclusion 23](#_Toc134465612)

### Introduction

L'alimentation électrique est un produit de tous les jours dont le montant varie en fonction de nombreux facteurs. Divers événements peuvent affecter simultanément la production et la demande d'électricité (rivalités géopolitiques, fluctuations des coûts des matières premières, trafic commercial entre les pays européens...). La modélisation du prix de l'alimentation électrique en fonction de ces facteurs peut donc être difficile. Dans ce contexte, notre projet cherche à modéliser le prix de l'alimentation électrique en France et en Allemagne en utilisant des données météorologiques, énergétiques et commerciales. Ce n'est pas seulement un problème de prédiction, c'est plutôt un problème d'explication de prix par d'autres variables simultanées. L'objectif est de créer un modèle qui évalue le changement quotidien du prix des contrats à terme (également appelés "futures") sur l'alimentation électrique en France ou en Allemagne, à partir des conditions actuelles du marché. Les futures sont des instruments financiers qui donnent une estimation de la valeur de l'alimentation électrique au moment de la maturité du contrat à partir des conditions actuelles du marché. Les variables explicatives comprennent des mesures quotidiennes de données météorologiques, de fabrication énergétique et d'utilisation de l'alimentation électrique. Le projet sera conduit en utilisant la méthodologie CRISP-DM (Cross-Industry Standard Process for Data Mining), qui est une méthode flexible et itérative en 6 étapes, allant de la compréhension du problème commercial à la mise en service. Pour mener à bien ce projet nous allons voir les étapes d’implémentation et description des données puis la préparation des données pour pouvoir par la suite mener une analyse exploratoire sur c’est donné et d’ainsi les modélisé en créant des modèles que nous évaluerons.

# Description des données

## Importation des données

Dans un premier temps nous avons importé les 3 fichiers.csv contenant les données du projet :



Data\_X.csv : Les données d’entrée.

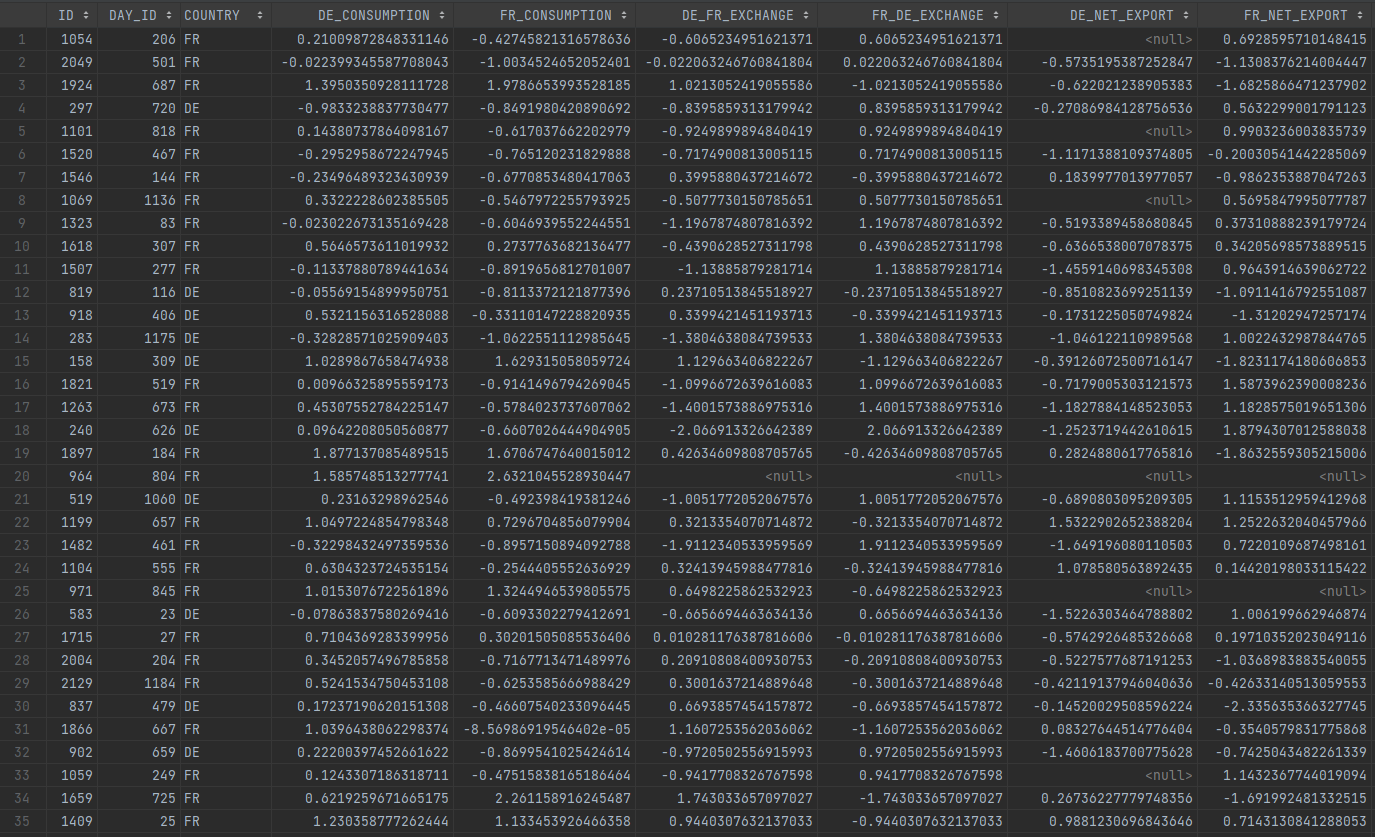
Data\_Y.csv : Les données en sortie.

DataNew\_X(1).csv : Nouvelles données en entrée non labélisées.

Les données en

L’affichage des données dans les fichiers se présente sous la forme ci-dessous:

**Data\_X.csv :**



## Identification des données

Les données d'entrée Data X et DataNew X correspondent aux mêmes variables explicatives, mais pour deux périodes distinctes.

Pour c’est deux fichiers, les données d’entrée possèdent 35 colonnes :

**Identifiant :**

-ID : identifiant associant jour (DAY\_ID) et pays (COUNTRY)

-DAY\_ID : Identifiant des jours

-COUNTRY : Identifiant des pays

**Variation journalières du prix des matières premières :**

-GAS RET : Gaz en Europe

-COAL RET : Charbon en Europe

-CARBON RET : Futures émissions de carbone

**Mesure météorologique :**

-X\_TEMP : Température

-X\_RAIN : Pluie

-X\_WIND : Vent

**Mesure de productions d’énergie journalière dans les pays :**

X\_GAS : Gaz naturel

X\_COAL : Charbon

X\_HYDRO : Hydraulique

X\_NUCLEAR : Nucléaire

X\_SOLAR: Photovoltaïque

X\_WINDPOW: Eolienne

X\_LIGNITE : Lignite

**Mesure d’utilisation électrique journalières dans les pays :**

-x\_CONSUMPTON : Électricité totale consommée

-x\_RESIDUAL LOAD : Électricité consommée après utilisation des d’énergies renouvelables

-x\_NET\_IMPORT : Électricité importée depuis l’Europe

-x\_NET\_EXPORT : Électricité exportée vers l’Europe

-DE\_FR\_EXCHANGE : Electricité ´échangée entre Allemagne et France

-FR\_DE\_EXCHANGE : Electricité échangée entre France et Allemagne.

Concernant les données en sortie Data\_Y, les données se composent en deux colonnes :

**Identifiant :**

-ID : identifiant identique aux données d’entrée

**Variation du prix de l’électricité :**

-TARGET : Variation journalière du prix de l’électricité dans le futur (maturité 24h)

## Importation des librairie python

Après avoir importé et identifié nos données, nous avons importé toutes les librairie python qui nous serons nécéssaire pour réalisé ce projet :

import pandas as pd  
import numpy as np  
import seaborn as sns  
import sklearn.cluster as skc  
import sklearn.preprocessing as skp  
import csv

**-pandas:** Manipuler et analyser des données de manière efficace et intuitive.

**-numpy:** Outils pour travailler avec des tableaux multidimensionnels (arrays) et effectuer des opérations mathématiques sur ces derniers.

**-seaborn:** Créer facilement des visualisations statistiques attractives et informatives à partir de données structurées.

**-sklearn.cluster:** Fournit des algorithmes de clustering pour regrouper des données similaires en fonction de leurs caractéristiques communes. Utile pour la segmentation de données et la détection de motifs.

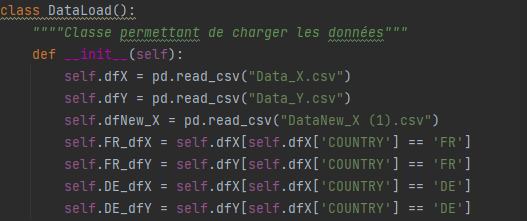
**-sklearn.preprocessing:** Outils pour prétraiter et transformer les données en vue de les utiliser dans des modèles d'apprentissage automatique, notamment pour la normalisation, la binarisation, l'encodage et l'imputation de données manquantes.

**-csv:** Fournit des fonctionnalités pour lire, écrire et manipuler des fichiers au format CSV

# Préparation des données

## 1) Chargement des données

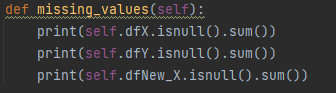
Pour commencer, nous avons créé une classe : DataLoad() nous permettant de charger les données à partir des fichiers csv :



Lorsque la classe est instanciée, les fichiers "Data\_X.csv", "Data\_Y.csv" et "DataNew\_X (1).csv" sont lus et les données sont stockées dans les attributs dfX, dfY et dfNew\_X respectivement. Quatre nouveaux DataFrame sont créés pour les données correspondantes aux pays 'FR' et 'DE' dans les DataFrames dfX et dfY, et stockés dans les attributs FR\_dfX, FR\_dfY, DE\_dfX et DE\_dfY.

## Valeurs manquantes

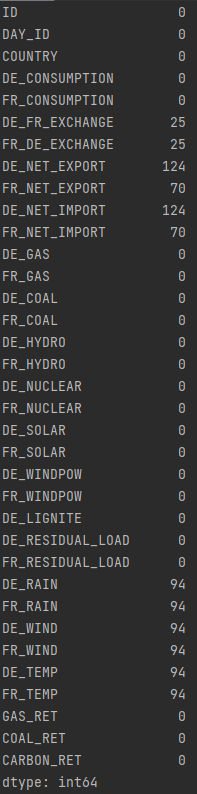
Une fois nos donnée afficher, nous avons remarqué qu’il y avait des données manquante.Nous avons donc crée une méthode missing\_values() pour afficher le nombre de valeurs manquantes dans cacune des DataFrame :



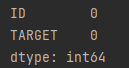
La méthode utilise la fonction isnull() de pandas pour déterminer les valeurs manquantes et la fonction sum() pour calculer la somme de ces valeurs manquantes. Les résultats sont affichés par un print() pour chaque DataFrame.

**Output de la méthode :**

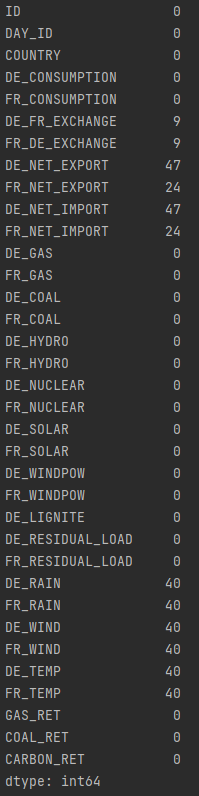
**Pour Data\_X :**



**Pour Data\_Y :**

****

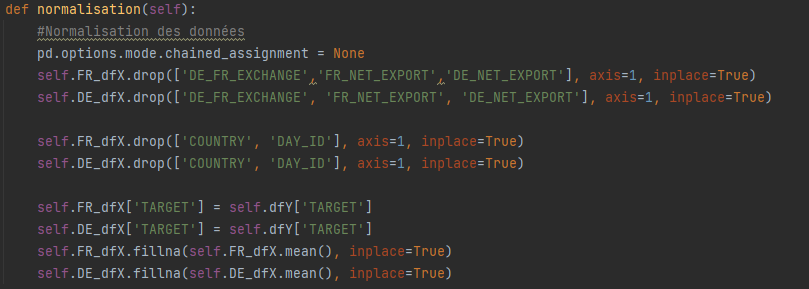
**Pour DataNew\_X :**

****

On peux voir que toutes les variable ou il y a des valeurs manquantes on une somme différentes de 0.

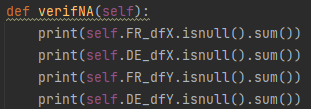
## Normalisation des données

Après avoir trouver le nombre de valeursm manquantes dans nos DataFrame, nous avons normalisé nos DataFrame à l’aide de la méthode normalisaion() :



Tout d'abord, les colonnes 'DE\_FR\_EXCHANGE', 'FR\_NET\_EXPORT' et 'DE\_NET\_EXPORT' sont supprimées des deux DataFrames. Ensuite, les colonnes 'COUNTRY' et 'DAY\_ID' sont également supprimées. Les colonnes 'TARGET' sont ensuite ajoutées aux deux DataFrames à partir du DataFrame dfY. Enfin, les valeurs manquantes de chaque DataFrame sont remplacées par la moyenne de la colonne correspondante. Le résultat final est deux DataFrames normalisés qui peuvent être utilisés pour entamer l'analyse de données.

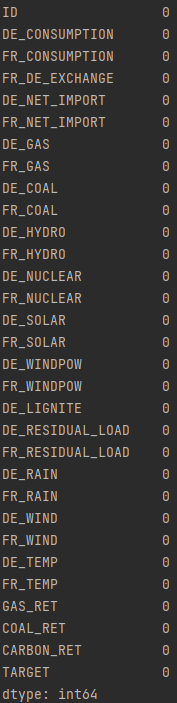
De plus, nous utilisons la méthode verifNA pour être sur qu’il y a aucune valeur manquante dans nos DataFrame :



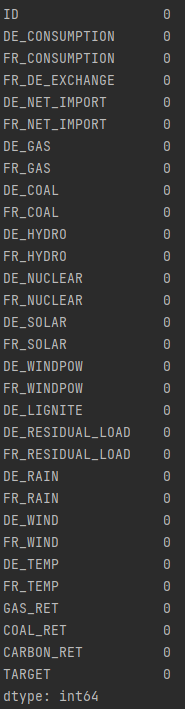
isnull() est utilisé pour détecter les valeurs manquantes et la fonction sum() pour calculer la somme de ces valeurs manquantes. Nous avons choisie d’utiliser cette méthode pour vérifier rapidement s'il y a des valeurs manquantes dans les données avant de poursuivre l'analyse des données.

**Output de la méthode :**

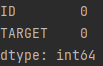
**FR\_dfX :**



**DE\_dfX :**



**FR\_dfY :**

****

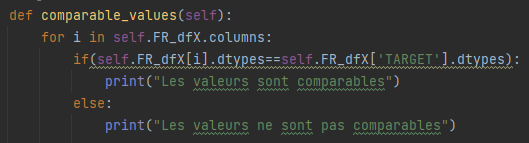
**DE\_dfY :**



On voit bien que toute nos variable on une somme de 0. Se qui se traduit pas l’absence de valeur null dans nos DataFrame.

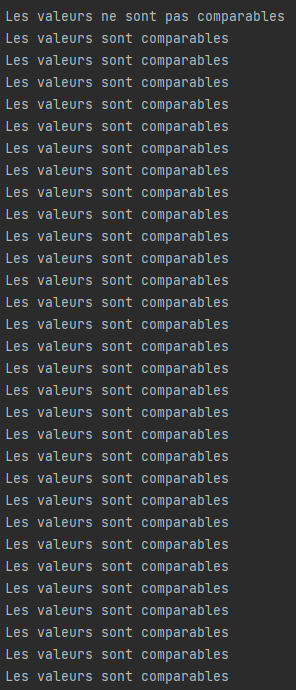
## Comparaison des attributs

Enfin, pour finir la préparation des données, nous allons vérifier si les valeurs des colonnes des données de nos DataFrame sont compparables avec la variable cible à étudié dans ce projet (TARGET) :



La méthode parcourt toutes les colonnes du DataFrame FR\_dfX en utilisant une boucle for et compare le type de données de chaque colonne avec celui de la colonne TARGET en utilisant la méthode dtypes() de pandas. Si les types de données sont les mêmes, la méthode affiche "Les valeurs sont comparables", sinon elle affiche "Les valeurs ne sont pas comparables". Cette méthode est utile pour vérifier si toutes les colonnes du DataFrame FR\_dfX ont des types de données appropriés pour l'analyse ou la modélisation de données.

**Output de la méthode :**

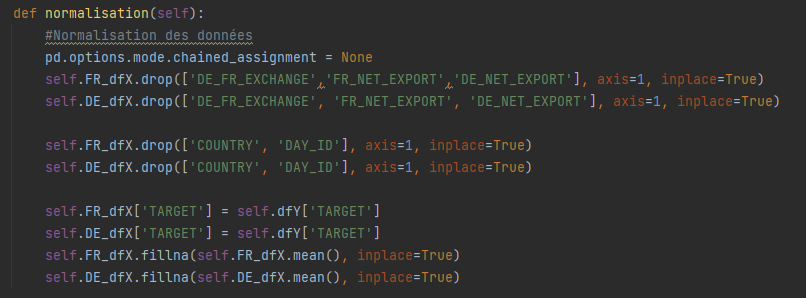
****

Nous pouvons voir que toutes les variables sont comparables avec TARGET sauf ID étant donnée que ID est de type INT et non de type FLOAT.

# Analyse exploratoire des données

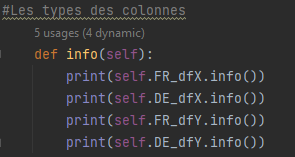
# 1) Identification de la variable cible

Comme dit précédemment, TARGET est la variable cible de ce projet. Etant donc la variable à étudier, nous avons donc décidé de l’implémenter dans les DataFrame X d’entrée pour FR et DE. Cela nous permet de faciliter l’études de Target avec les différentes DataFrame de FR et DE.



## Aperçus et éxamination des variables

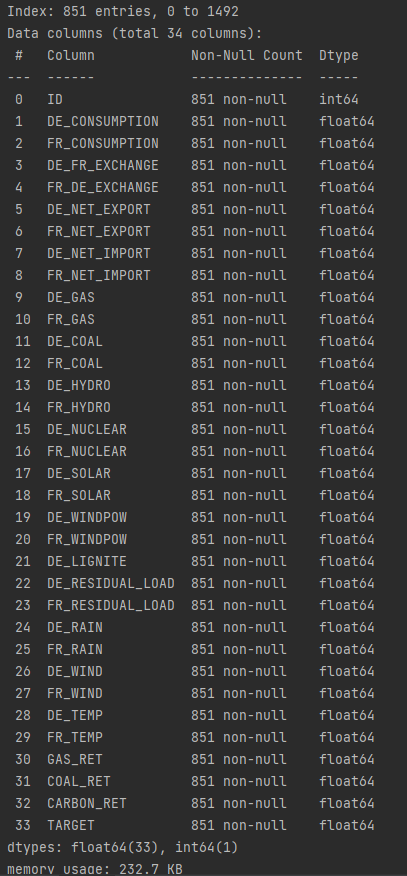
Pour commencer, nous avons utilisé la méthode info nous permettant d’obtenir les types des colonnes de nos DataFrame :



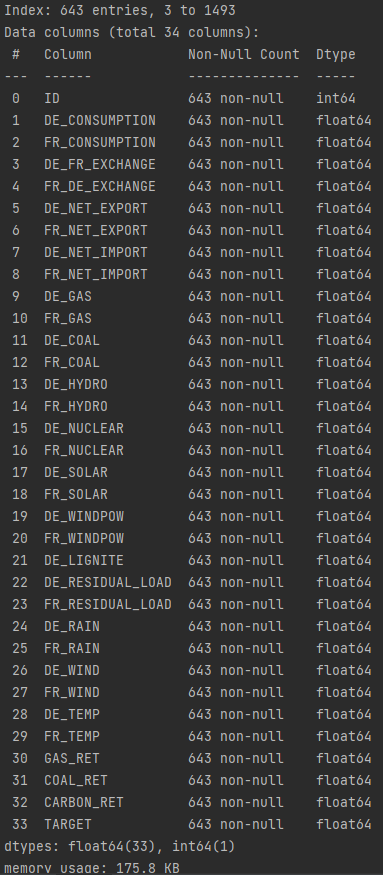
La méthode info() affiche des informations sur les colonnes de chaque DataFrame, telles que le nom de la colonne, le nombre de valeurs non nulles, le type de données et la mémoire utilisée. Cela nous permet de comprendre les données et à diagnostiquer les problèmes potentiels tels que les valeurs manquantes ou les types de données incorrects.

**Output de la méthode :**

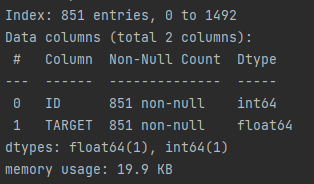
**FR\_dfX :**

****

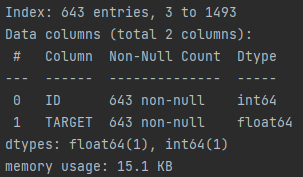
**DE\_dfX :**

****

**FR\_dfY :**

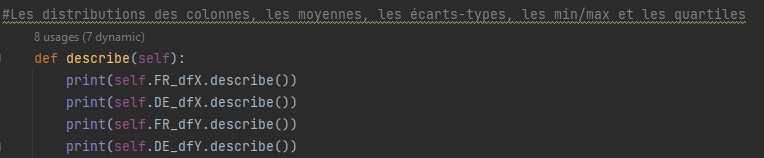
****

**DE\_dfY :**

****

Nous pouvons bien voir le type chaque variable dans les Data Frame étant tous des FLOAT mis à part ID qui est un INT.

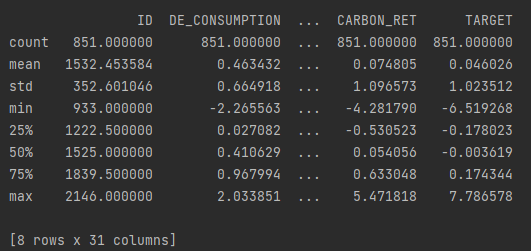
Ensuite nous affichons les distributions et les plages de valeurs des variables de nos DataFrame grâce à la méthode describe :



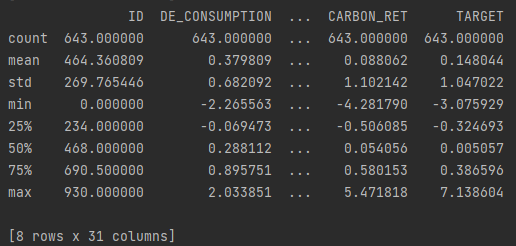
La méthode describe() calcule des statistiques de base pour chaque colonne de chaque DataFrame, telles que le nombre d'observations, la moyenne, l'écart type, le minimum et le maximum. Cela nous aide à comprendre la distribution des données et à identifier les valeurs aberrantes ou les erreurs potentielles.

**Output de la méthode :**

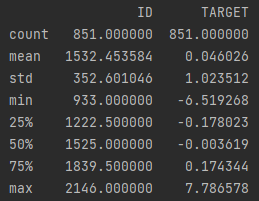
**FR\_dfX :**

****

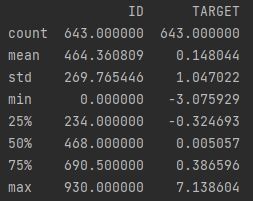
**DE\_dfX :**



**FR\_dfY :**

****

**DE\_dfY :**

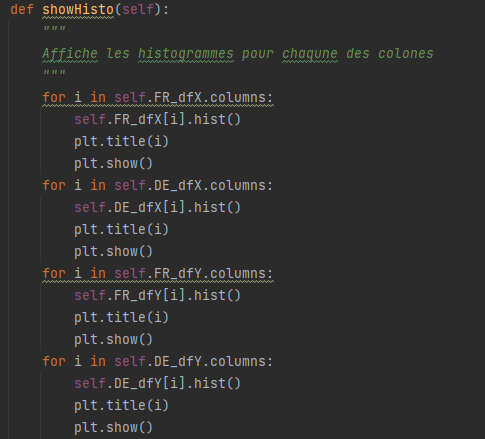


Nous pouvons voir ici les différentes informations statistique fournit :count(Nombres de valeurs), mean(Moyenne), std(écart type), min(minimum), 25%(1er quartile), 50%(2ème quartile ou médiane), 75%(3ème quartile), max(maximum).

On remarque que pour **FR\_dfX**et **DE\_dfX,** la taille des tableaux nous est donnée avec (…) étant donnée le terminal ne peux pas tout nous afficher en 1 seule fois.

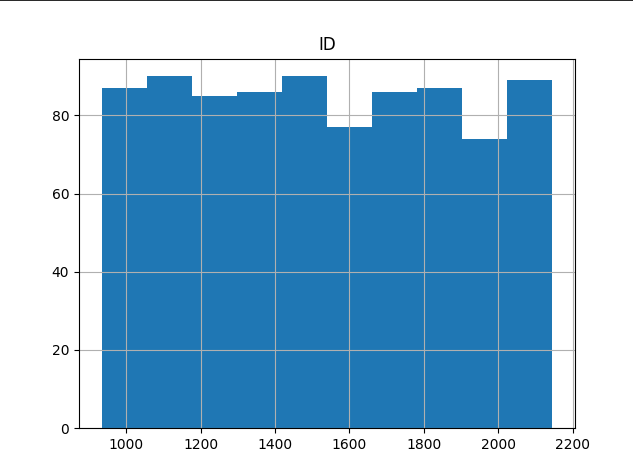
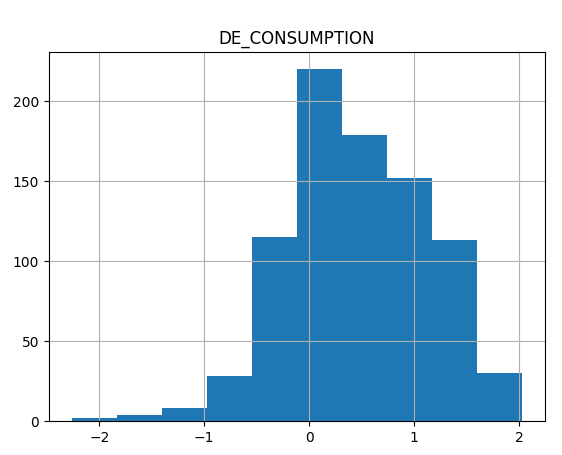
## Les Histogramme

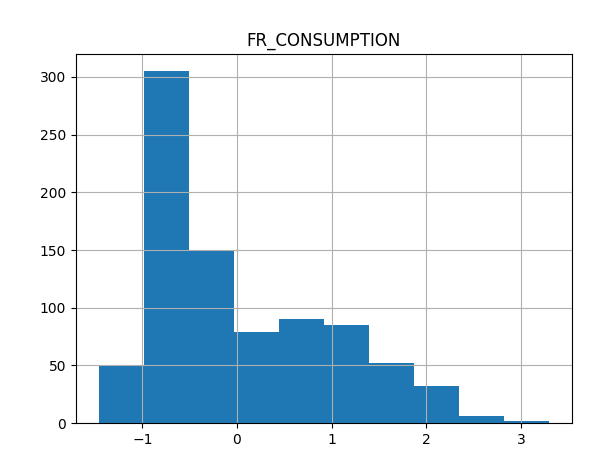
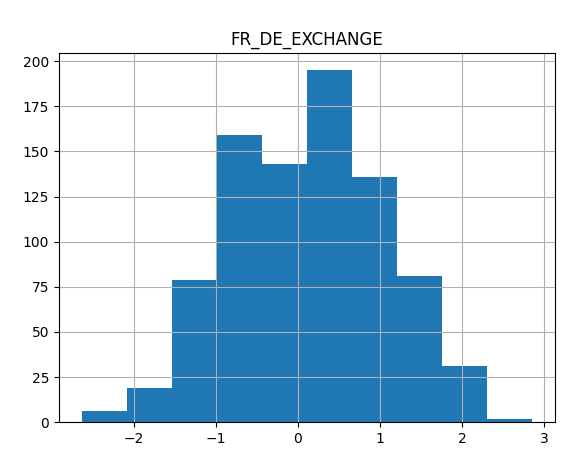
Maintenant, nous allons afficher les histogrammes de chaque variable deb nos DataFrame :

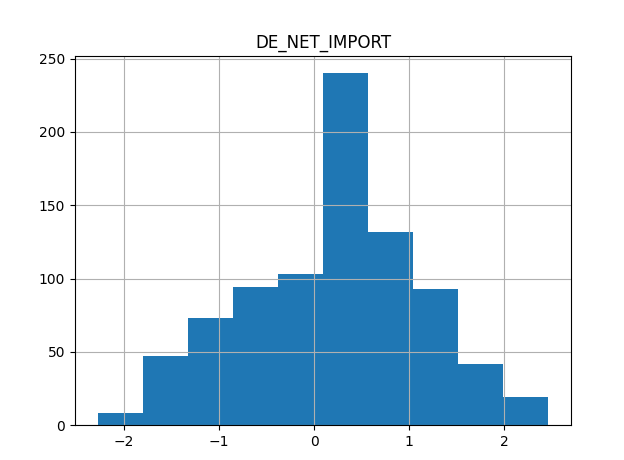
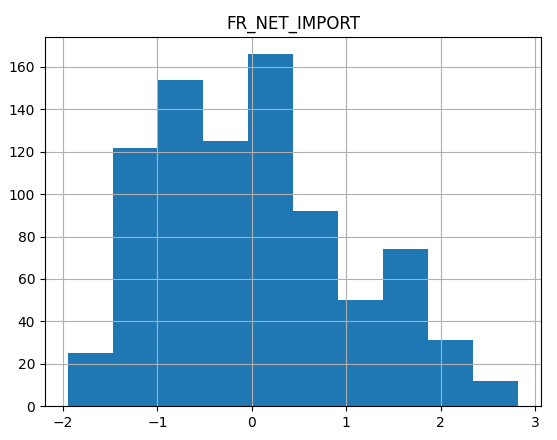
****

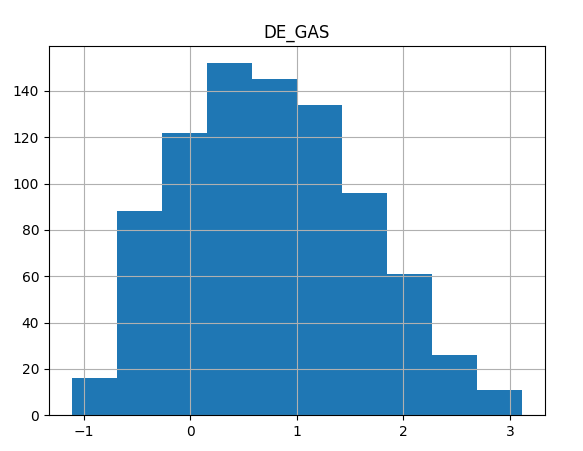
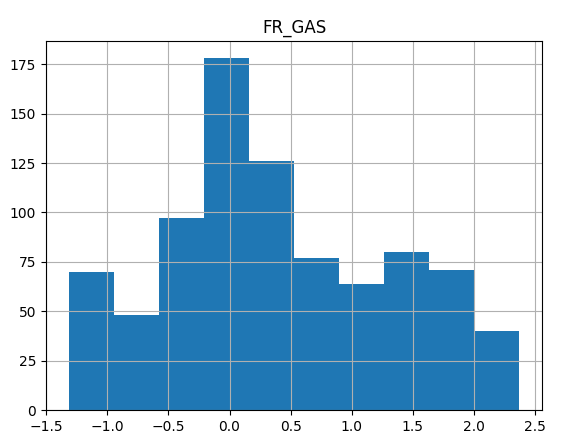
A l'aide de la fonction hist() de la bibliothèque matplotlib. La méthode itère à travers chaque colonne des DataFrames et crée un histogramme pour chaque colonne, puis affiche le titre de la colonne. show() est appelée à la fin de chaque boucle pour afficher chaque histogramme des DataFrame.

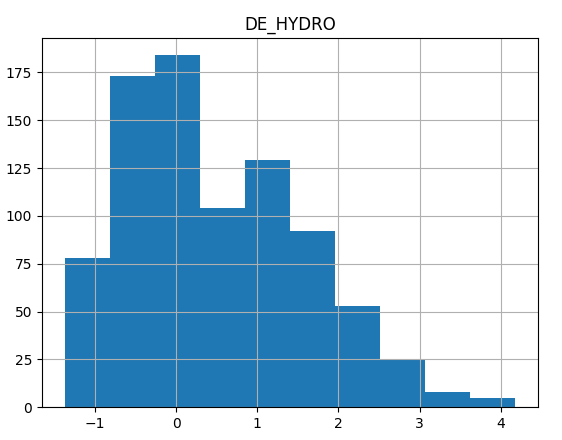
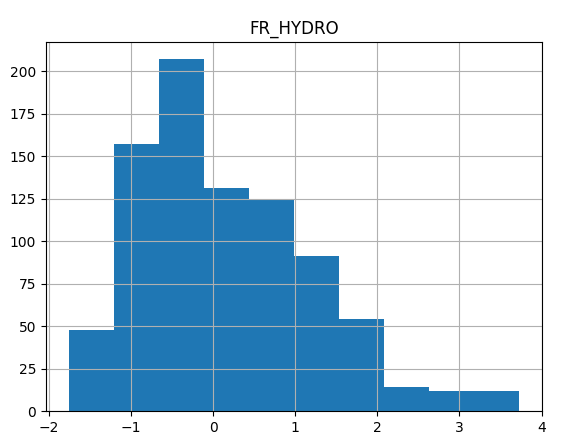
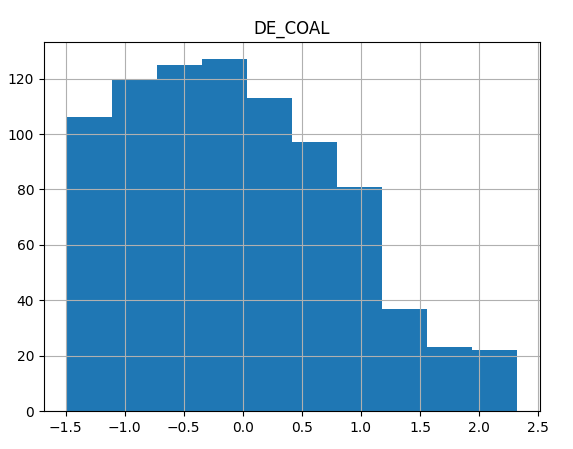
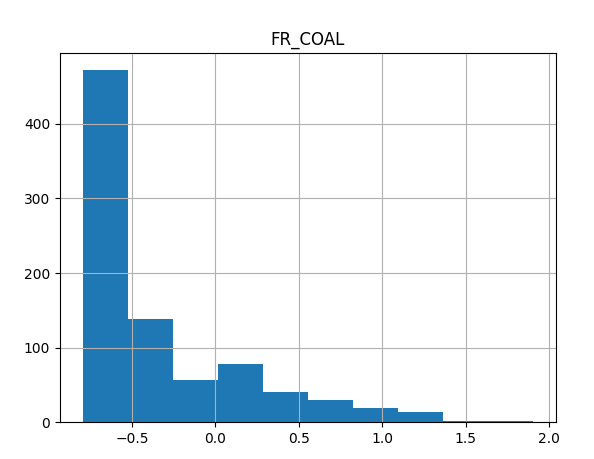
**Output de la méthode :**

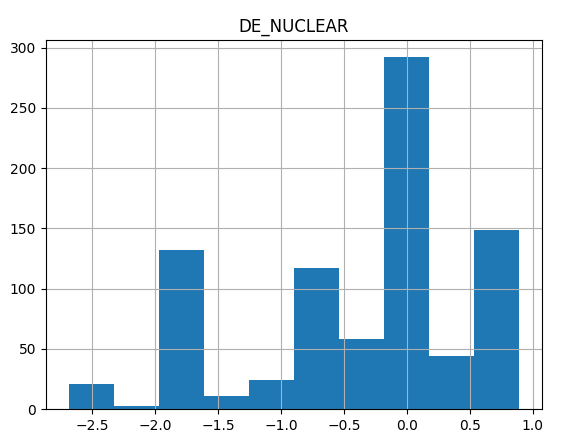
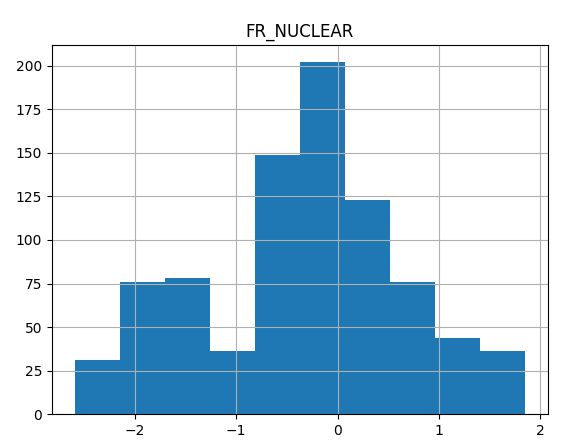


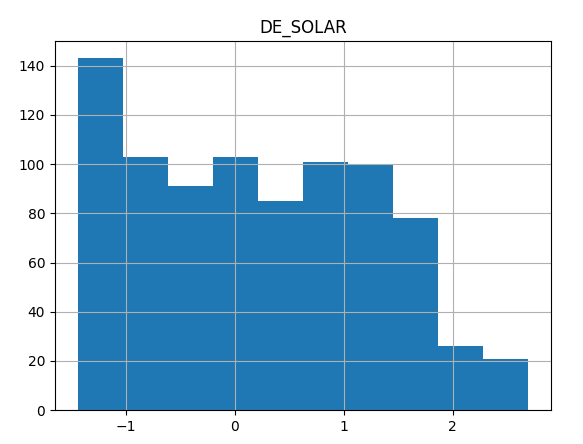
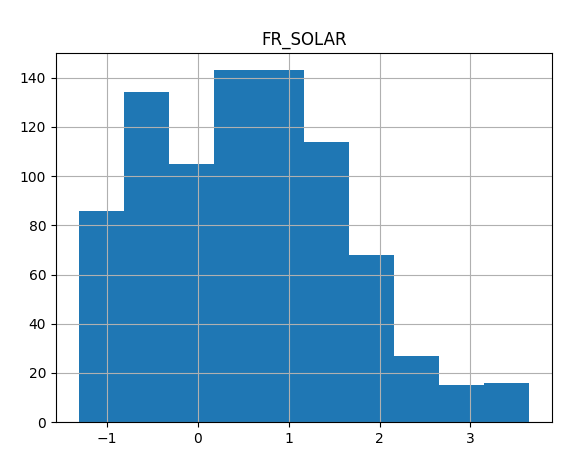
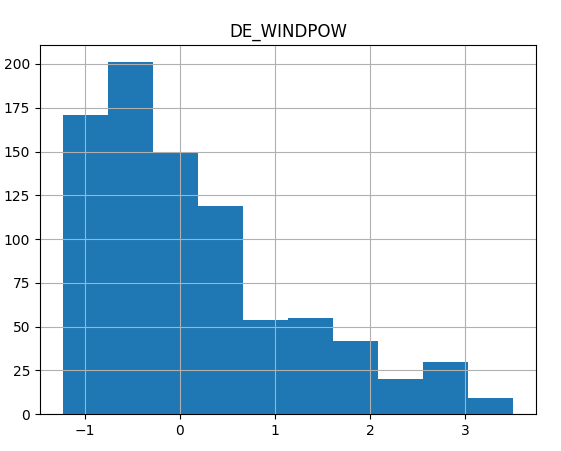
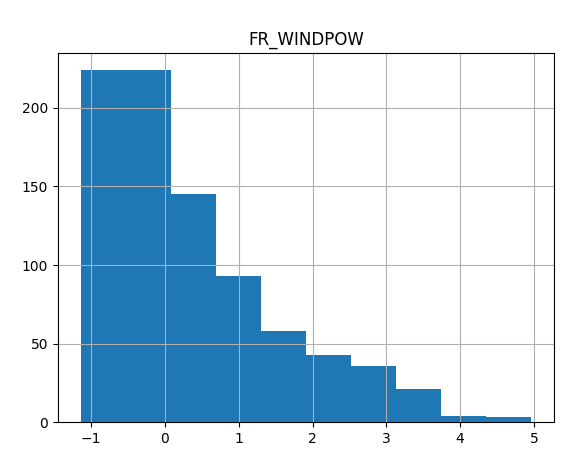
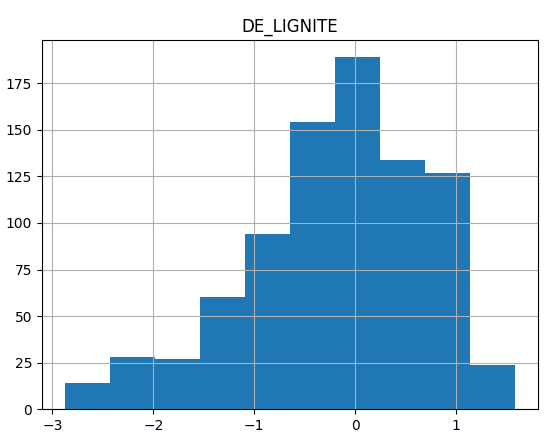
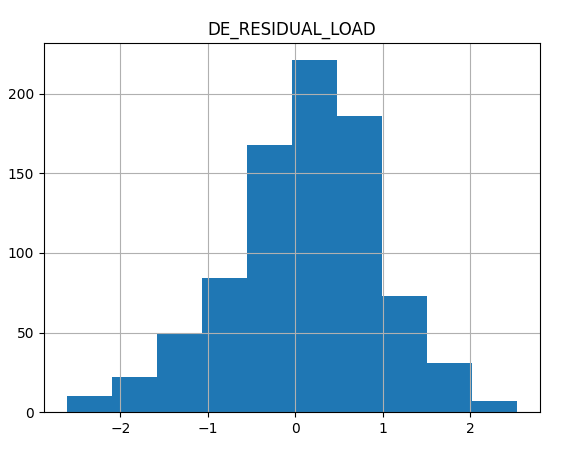


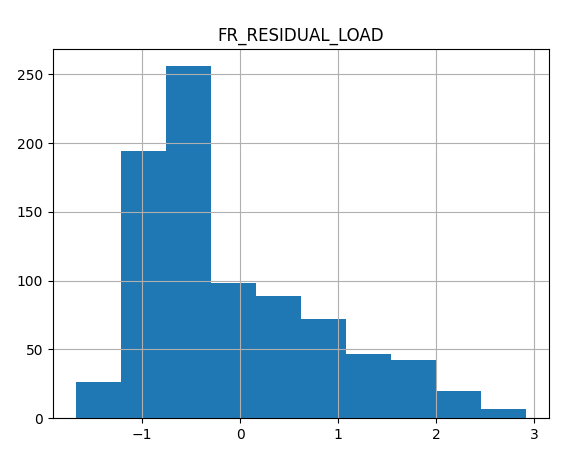
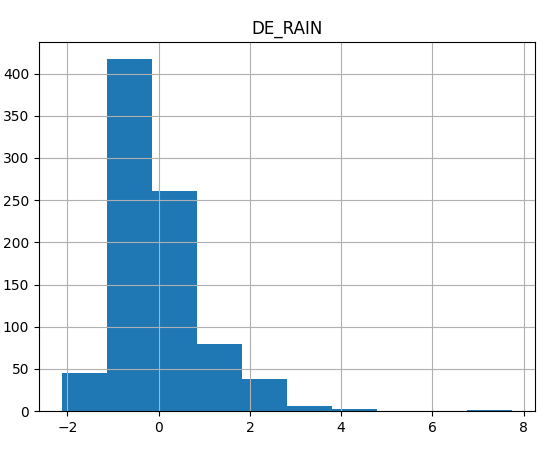


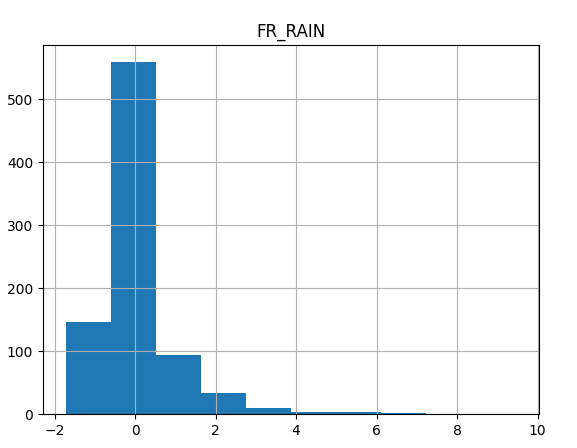
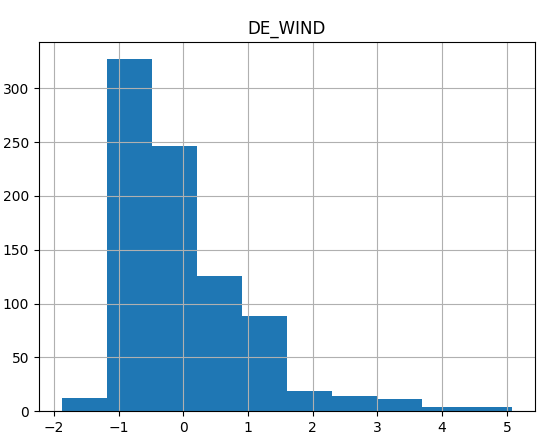


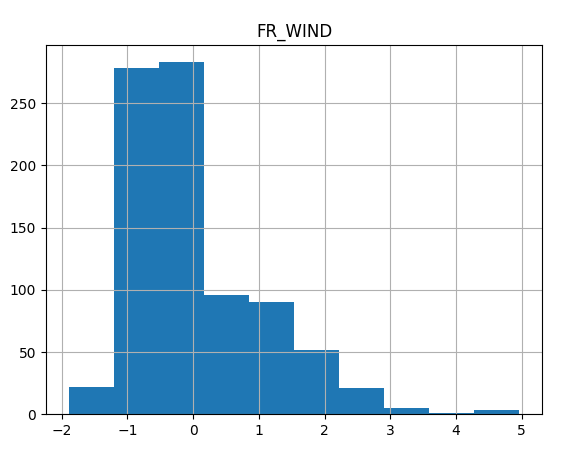
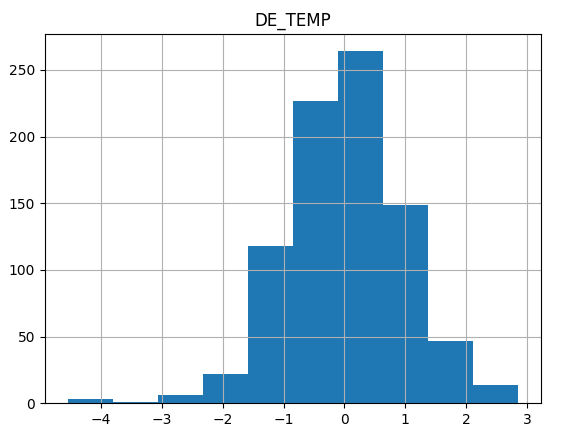
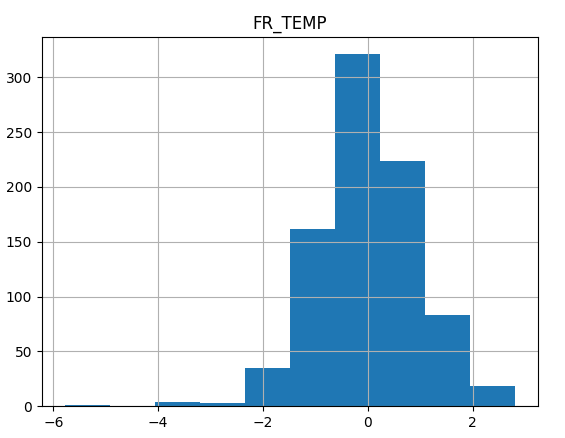
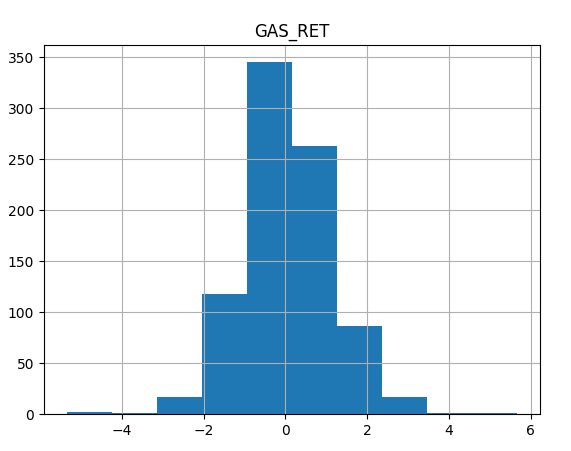
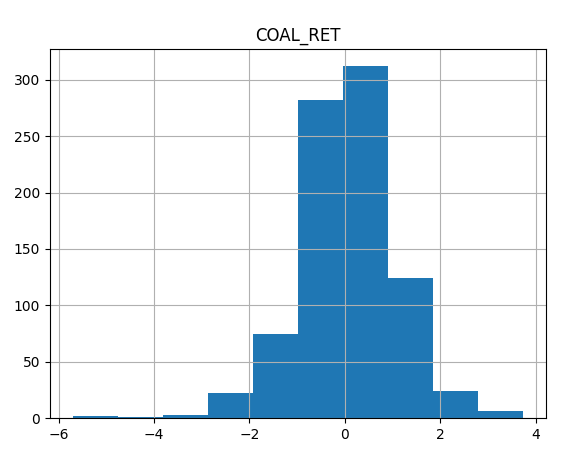
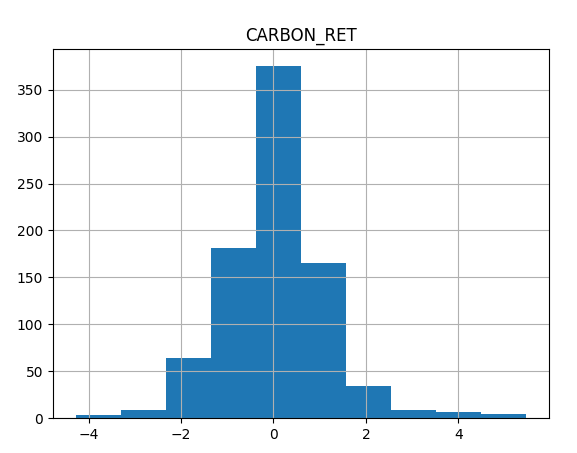






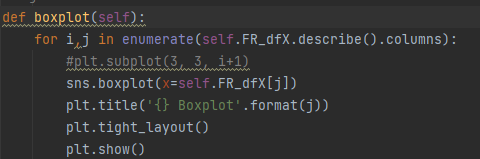






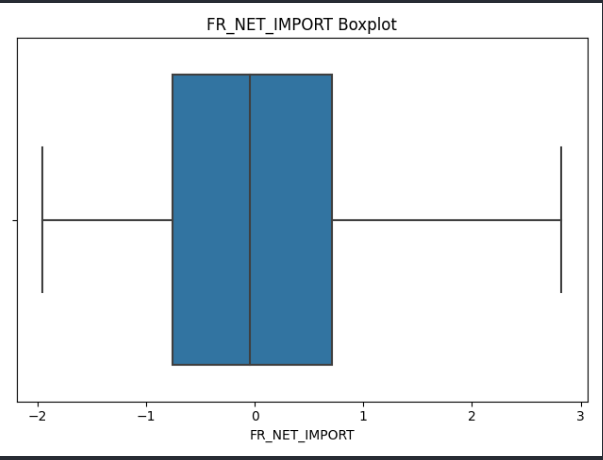
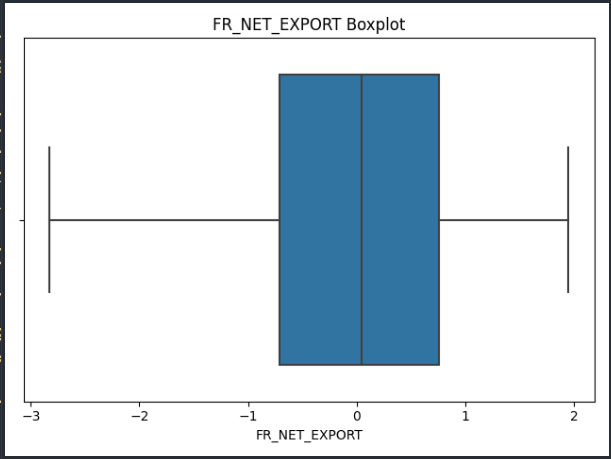
Une fois les histogramme afficher, on peut s’intéréssé a une autre manière de représenté les variables de nos DataFrame qui sont les Boxplot. Nous avons choisit de coder 2 méthode pour les Boxplot : Une pour afficher les DataFrame de FR et l’autre pour les DataFrame de DE.

**Pour FR :**

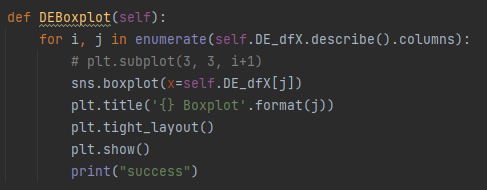


Cette méthode crée des boxplots pour chacune des colonnes du DataFrame **self.FR\_dfX** en utilisant la bibliothèque Seaborn. Les boxplots permettent de visualiser la distribution des données en affichant des quartiles, des valeurs aberrantes et la médiane.

**Output de la méthode :**

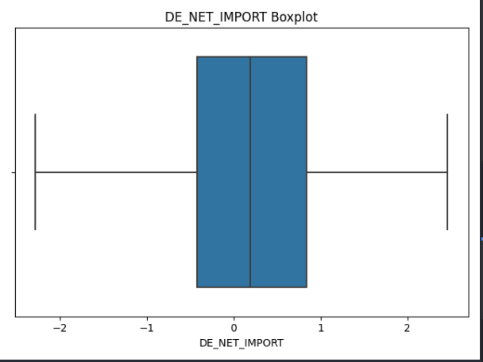
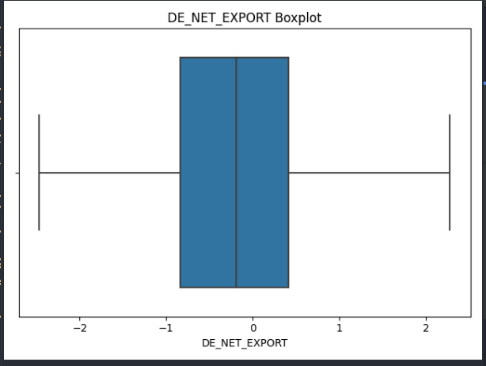


**Pour DE :**

****

Cette méthode et la même que boxplot, nous avons juste changer modifié le DataFrame pour l’appliqué cette fois-ci à DE.

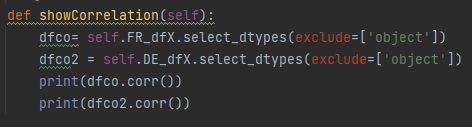
**Output de la méthode :**



Nous avons décidé de ne pas afficher tout les Boxplot pour FR\_dfX et DE\_dfX car il fournissent moins d’information nécéssaire que les histogrammes.

## Matrice des corrélations

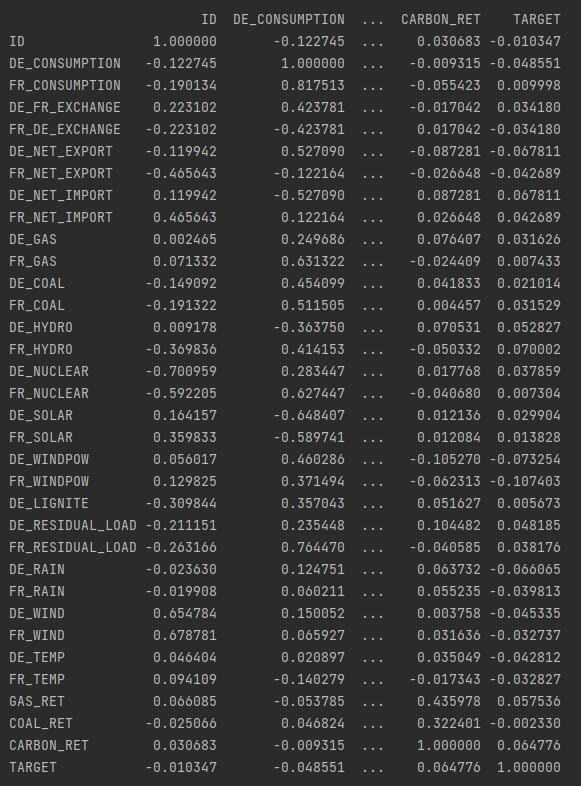
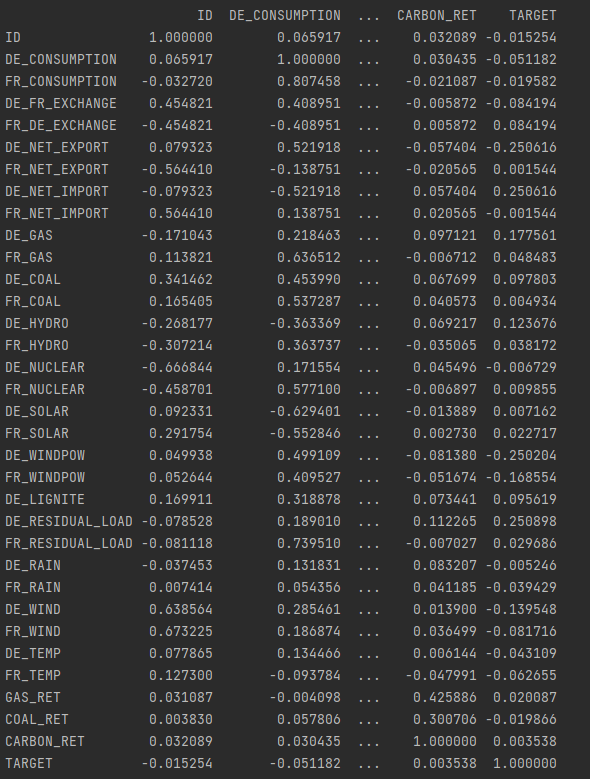
Ici, nous avons crée une matrice des corrélations afin de connêtre les corrélations entre TARGET et les autres variables des DataFrame FR\_dfX et DE\_dfX. Pour cela, nous utilisons la méthode showCorrelation() :



La méthode calcule la matrice de corrélation entre les variables numériques des DataFrames FR\_dfX et DE\_dfX. Il exclut les variables de type "object" (variables catégorielles) de ces deux DataFrames avant de calculer la matrice de corrélation pour éviter les erreurs de calcul. Les matrice sont affiché à l'aide du print().

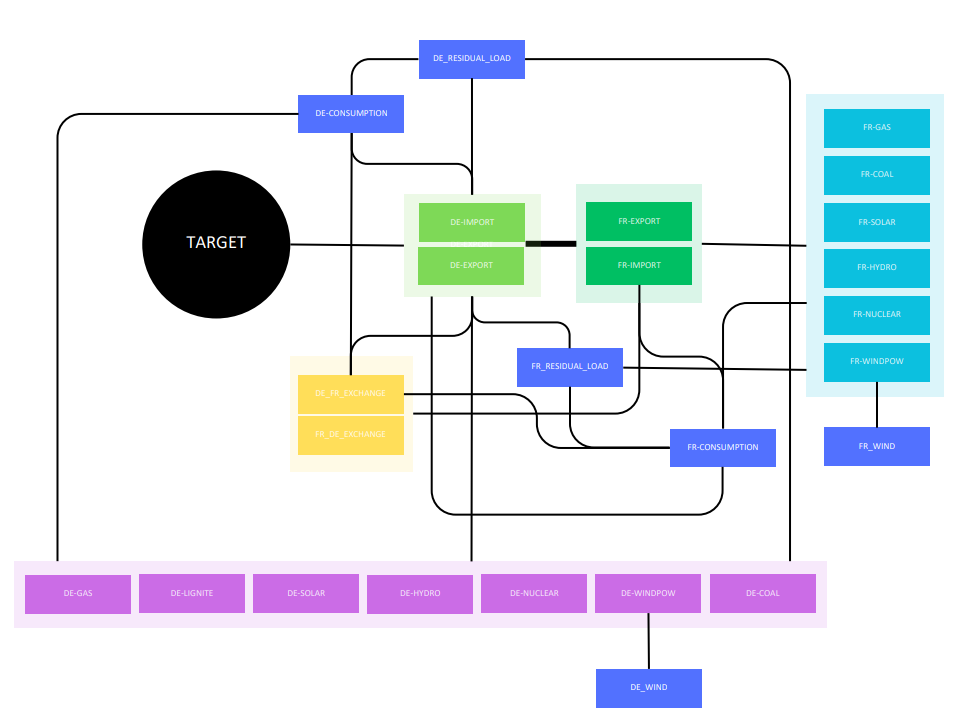
**Output de la méthode :**

**Matrice pour DE :**

**Matrice pour FR :**

Ci-dessous les corrélation des variables avec la variable cible TARGET pour les DataFrame FR\_dfX et DE\_dfX.

## Interprétation des résultats



Pour interpréter les résultats, nous avons choisie de représenté cela sous la forme d’un organigramme. Dans cette organigramme, **TARGET** et nôtre variable cible. En regardant, nos différentes corrélations avec **TARGET**, nous avons sélectionner les corrélations les plus importantes que nous représentons ci-dessous.

Comme nous pouvons le voir, **TARGET** est le plus corrélé avec **DE-IMPORT** et **DE-EXPORT**. De plus, nous pouvons remarquer d’autre corrélation secondaire qui reste corrélé avec TARGET comme FR-EXPORT, FR-IMPORT, DE-RESIDUAL\_LOAD, DE\_CONSUMPTION, FR-RESIDUAL\_LOAD, FR\_CONSUMPTION, DE\_FR\_EXCHANGE, FR\_DE\_EXCHANGE…

Les liaisons faites avec entre chaque variable représentent le niveau de corrélation entre elle.

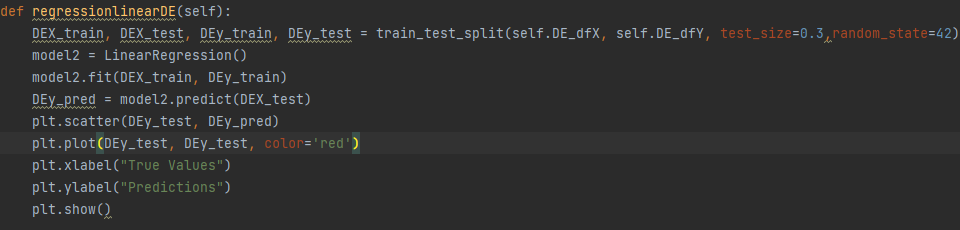
Par exemple si l’on prend **TARGET** et **DE\_IMPORT**, on peut voir que la liaison entre ces deux variables est directe ce qui montre une très forte corrélation. Tandis que si l’on prend **FR\_IMPORT**, avec TARGET, on peut voir que celui-ci n’est pas en lien directement avec notre variable cible. Ce qui nous montre que la corrélation entre c’est deux variable et plus faible que la précédente. Ce qui est logique car en effet, la France est l’un des pays les plus avancés en therme de nucléaire et donc n’a pas forcément de grand besoin d’importé de l’électricité contrairement à l’Allemagne qui utilise beaucoup moins le nucléaire et qui a donc besoin d’importer plus d’électricité.

# Modélisation des données

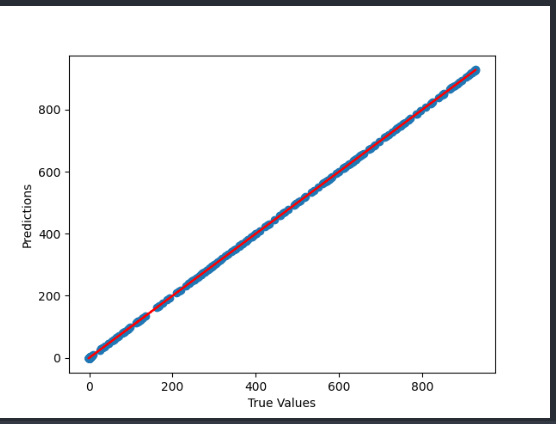
## 1) Régression linéaire simple

Nous allons effectué une régression linéaire sur les données de DE et FR. Pour cela, nous avons choisit de crée une fonction qui fait une régrétion linaire sur FR et une autre sur DE :

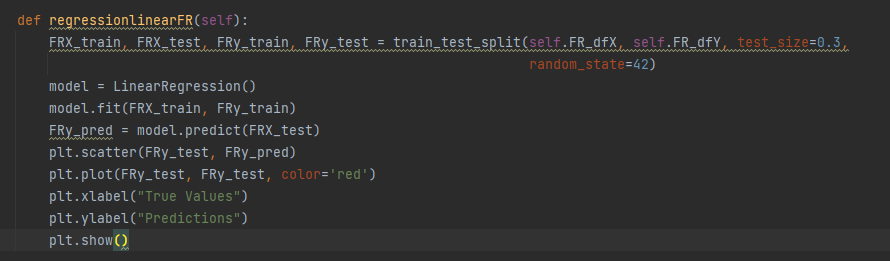
**Pour DE :**

Régression linéaire sur les données DE\_dfX et DE\_dfY. La fonction train\_test\_split de Scikit-learn pour diviser les données en ensemble d'entraînement et ensemble de test avec un ratio de 0,3. Ensuite, il crée une instance de l'algorithme de régression linéaire de Scikit-learn et l'entraîne sur l'ensemble d'entraînement. Il prédit ensuite les valeurs de la variable cible sur l'ensemble de test en utilisant la méthode predict de l'instance du modèle de régression linéaire. Enfin, il affiche un nuage de points pour comparer les valeurs prédites aux valeurs réelles, ainsi qu'une ligne rouge qui représente la relation linéaire idéale entre les deux.

**Output de la méthode :**

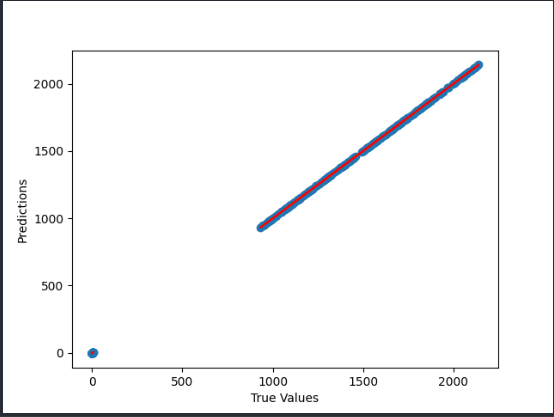


**Pour FR :**



Même fonctionnement que la méthode regressionlinearDE.

**Output de la méthode :**



**Puis on a réalisé les autres modèles de modélisation à part le KNN afin de pouvoir modéliser nos données.**

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

Description générée automatiquementUne image contenant texte, capture d’écran, Police

Description générée automatiquement

Une image contenant texte, capture d’écran

Description générée automatiquement

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

La régression ridge ajoute une pénalité L2 à la fonction de coût de la régression linéaire. Cette pénalité est égale à la somme des carrés des coefficients de la régression. Cela signifie que les coefficients sont régularisés en étant contraints à de plus petites valeurs, ce qui peut réduire le surajustement (overfitting) du modèle. La force de la régularisation est contrôlée par un hyperparamètre λ, qui doit être choisi à l'aide d'une validation croisée.

La régression Lasso ajoute une pénalité L1 à la fonction de coût de la régression linéaire. Cette pénalité est égale à la somme des valeurs absolues des coefficients de la régression. Cela signifie que certains coefficients peuvent être mis à zéro, ce qui peut réduire la complexité du modèle et améliorer sa généralisation. La force de la régularisation est contrôlée par l'hyperparamètre λ, qui doit être choisi à l'aide d'une validation croisée.

La méthode des forêts aléatoires (ou Random Forest en anglais) est un algorithme d'apprentissage automatique supervisé qui combine plusieurs arbres de décision pour améliorer la précision et la stabilité des prédictions. Elle est utilisée pour des tâches de classification ou de régression.

Le fonctionnement des forêts aléatoires consiste à créer un grand nombre d'arbres de décision indépendants, chacun étant entraîné sur un sous-ensemble aléatoire des données d'entraînement et des variables d'entrée. Cela permet de réduire les effets du surapprentissage (overfitting) et d'améliorer la capacité de généralisation du modèle.

Lors de la prédiction, chaque arbre de décision donne une prédiction pour la variable cible, et la prédiction finale est obtenue en moyennant (pour une tâche de régression) ou en effectuant un vote majoritaire (pour une tâche de classification) sur les prédictions de tous les arbres.

La méthode des forêts aléatoires présente plusieurs avantages : elle est robuste aux valeurs aberrantes, peut gérer de grandes quantités de données et de variables d'entrée, et fournit des informations sur l'importance des variables pour la prédiction.

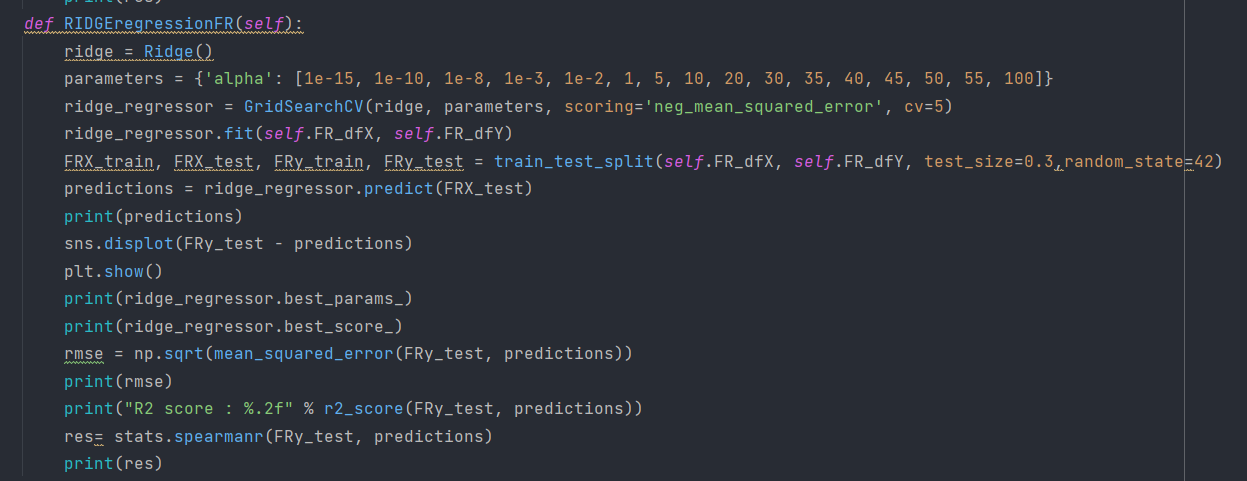
Cependant, elle peut être moins interprétable que d'autres modèles d'apprentissage automatique, et peut nécessiter une optimisation des hyperparamètres pour obtenir les meilleures performances.

# Evaluation des modèles

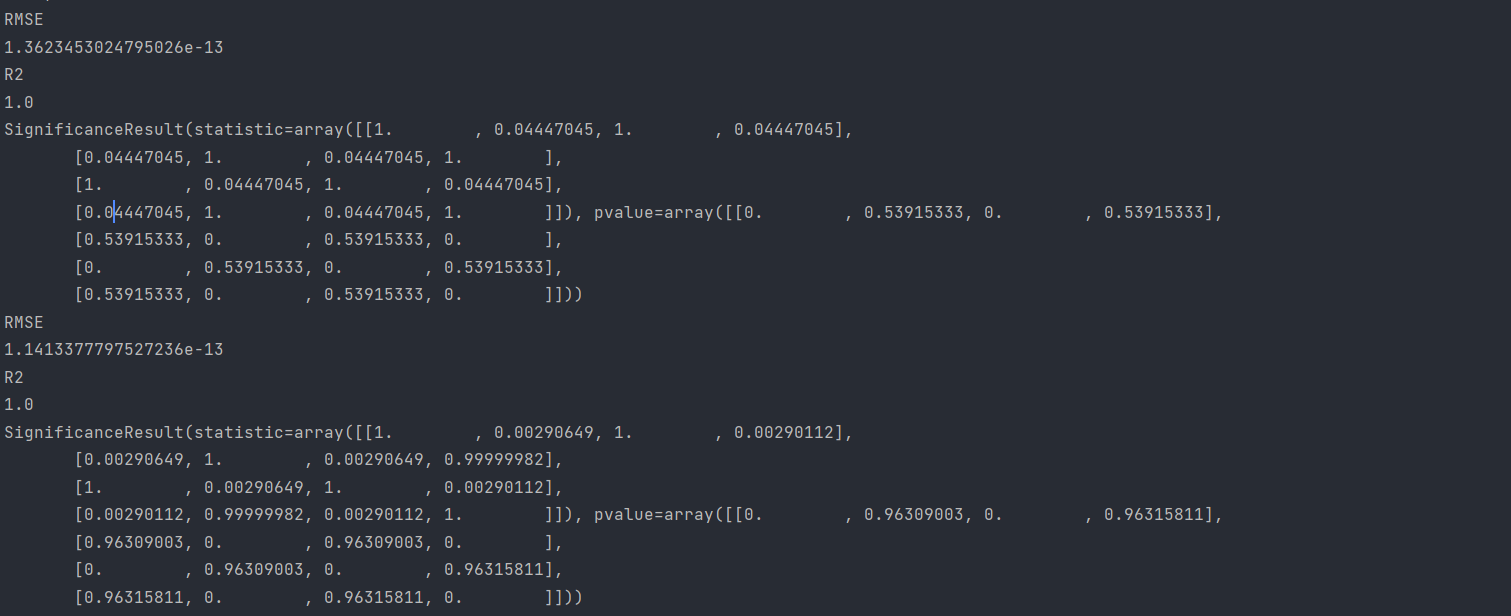
## Corrélation de Spearman, R2 et RMSE

Nous avons introduit la corrélation de Spearman (une mesure non paramétrique) pour savoir si on avait une monotonie entre nôtre Datasse de teste et notre prédiction.

Ensuite nous avons calculer le R2\_score qui nous permet de savoir si la droite de régression colle à l’ensemble des points du dataframe.

Enfin nous avons calculer le RMSE qui est l’erreur quadratique moyenne qui permet de fournir une indication sur la dispersion ou la variabilité de la qualité de la prédiction. Il est relié à la variance du modèle.

**Output de la méthode des méthodes avec Spearman :**



Ici on constate que dans notre modèle le RMSE est faible ce qui indique une faible variance des données. Notre R2 est égal à 1 ce qui veut dire que notre régression linéaire colle exactement avec l’ensemble des points donnés et que notre modèle est efficace.

On a ensuite calculé les meilleurs paramètres pour chaque modèle

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

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

Description générée automatiquement

On a ainsi pu obtenir les meilleurs paramètres pour notre modèle en ayant fait varier le alpha. GridSearchCV nous a permis de chercher automatiquement les meilleurs paramètres optimums pour notre modèle de modélisation des données afin d’obtenir le meilleur résultat possible ce qui explique notre R2 score de 1.

Une image contenant texte, Police, capture d’écran

Description générée automatiquement

Ainsi grâce au GridSearchCV on a trouvé le alpha optimal ainsi que le MSE optimal. Alors on a pu aussi déterminer une liste des modèles en termes de performance :

1-Forêts Aléatoires

2-Régression RIDGE

3-LASSO

4-Arbres de décision

5-Régression linéaire simple

6-K-NN

On a pu obtenir une très bonne performance en augmentant le n\_estimators (améliore la précision)

on a augmenté Maxdepth pour avoir une bonne performance (améliorer la précision) et on a augmenté la profondeur sans overfitter la modélisation.

Pour éviter l’overfitting on a augmenté le min\_samples\_split et le min\_samples\_leaf pour éviter l’overfitting. On a réduit max\_features pour réduire le surapprentissage.

### Conclusion

Notre projet de modélisation du prix de l'alimentation électrique en France et en Allemagne en utilisant des données météorologiques, énergétiques et commerciales a été mené à bien en respectant les différentes étapes de la méthodologie CRISP-DM. Nous avons réussi à créer un modèle qui évalue le changement quotidien du prix des contrats à terme sur l'alimentation électrique en se basant sur les conditions actuelles du marché et des variables explicatives telles que les mesures quotidiennes de données météorologiques, de fabrication énergétique et d'utilisation de l'alimentation électrique.

Malgré les difficultés rencontrées, ce projet nous a apporté une grande expérience en matière de traitement et de modélisation des données, ainsi que de gestion de projet. Nous avons également appris à travailler en équipe et à collaborer avec des experts dans différents domaines, ce qui a renforcé notre capacité à résoudre des problèmes complexes. En outre, nous avons pu acquérir une compréhension plus approfondie du marché de l'énergie et de ses facteurs d'influence, ainsi que de l'importance de l'analyse prédictive pour prendre des décisions éclairées.

Finalement, ce projet nous a permis de mettre en pratique nos connaissances en analyse de données et de développer des compétences utiles pour notre future carrière professionnelle. Notre travail nous rend ainsi plus confiant et prêts quant à nos futurs examens.