

**Prediction ‘Time Series’ sur la consommation d’energie dans la région gérer par la société PJM**

**PROJET**

|  |  |
| --- | --- |
|  |  |
|  |  |
| * + Version: | 1.0 |
| * + Date: | 05/03/2021 |
|  |  |
|  |  |

|  |  |
| --- | --- |
|  |  |

* TABLE DES MATIERES

[1 Contexte / Problématique 4](#_Toc65789693)

[1.1 Contexte 4](#_Toc65789694)

[1.2 Problématiques rencontrées par les RTO 5](#_Toc65789695)

[1.3 Périmètre de l’étude 5](#_Toc65789696)

[2 Identification des données 6](#_Toc65789697)

[2.1 Sources 6](#_Toc65789698)

[2.1.1 Energy Consumption (PJM’s website) 6](#_Toc65789699)

[2.1.2 Weather forecast (PJM’s website) 7](#_Toc65789700)

[2.2 Sélection des données 7](#_Toc65789701)

[2.2.1 Chaîne de traitement des données 7](#_Toc65789702)

[3 Méthodologie 8](#_Toc65789703)

[4 Préparation des données 9](#_Toc65789704)

[4.1 Gérer la composante temporelle 9](#_Toc65789705)

[4.2 Enlever les mauvaises données 9](#_Toc65789706)

[4.3 Remplacer les mauvaises données 9](#_Toc65789707)

[5 Exploration des données 10](#_Toc65789708)

[5.1 Visualition de la série temporelle 10](#_Toc65789709)

[5.1.1 Distribution de la consumation par compagnie 10](#_Toc65789710)

[5.1.2 Evolution annuelle de la consommation par compagnie 11](#_Toc65789711)

[5.1.3 Evolution de la consommation totale 11](#_Toc65789712)

[5.1.4 Corrélation de la série temporelle (Lag Scatter plot) 13](#_Toc65789713)

[5.1.5 Saisonnalité (Autocorrélation plots) 13](#_Toc65789714)

[5.2 Stationnarité 14](#_Toc65789715)

[6 Evaluation des modèles prédictifs 16](#_Toc65789716)

[6.1 Persistance 16](#_Toc65789717)

[6.2 Modèle kNN Regressor 17](#_Toc65789718)

[6.2.1 Définition 17](#_Toc65789719)

[6.2.1 Etude 18](#_Toc65789720)

[6.3 Modèle XGBoost 20](#_Toc65789721)

[6.3.1 Definition: XGBoost 20](#_Toc65789722)

[6.3.2 Etude 21](#_Toc65789723)

[6.4 Modèle ARIMA 28](#_Toc65789724)

[6.4.1 Recherche des meilleurs Hyper Paramètres 28](#_Toc65789725)

[ Méthode d’autocorrélation et corrélation partielle 28](#_Toc65789726)

[ Grid Search: 29](#_Toc65789727)

[6.5 Modèle SARIMA 30](#_Toc65789728)

[Le modèle SARIMA est une version alternative du modèle ARIMA conçue spécialement pour prendre en considération les effets de saisonnalité. 30](#_Toc65789729)

[ Résultats : 31](#_Toc65789730)

[6.6 Modèle LSTM (Long Short-Term Memory) 31](#_Toc65789731)

[Dans cette partie nous allons construire un reseau de Nerones. 31](#_Toc65789732)

[Un réseau de neurones classique permet de gérer les cas ou les expériences sont independentes les unes des autres. 31](#_Toc65789733)

[Lorsque les expériences sont sous la forme de séquences temporelles, une nouvelle structure a été inventée : les réseaux de neurones récurrents. 31](#_Toc65789734)

[Cette nouvelle structure introduit un mécanisme de mémoire des entrées précédentes qui persiste dans les états internes du réseau et peut ainsi impacter toutes ses sorties futures. Le réseau de neurones Long Short-Term Memory (LSTM) est l’un des plus connu. 31](#_Toc65789735)

[7 Résultats 33](#_Toc65789736)

[8 Déploiement de la solution 34](#_Toc65789737)

[9 Annexe – Procédure d’installation de l’environnement de développement 36](#_Toc65789738)

[9.1 Description de l’environnement 36](#_Toc65789739)

[9.2 Installation Anaconda en local sur la VM 36](#_Toc65789740)

[9.2.1 Mise à jour de la gestion locale des packages 36](#_Toc65789741)

[9.2.2 Téléchargement de la dernière version d'Anaconda 36](#_Toc65789742)

[9.2.3 Installation d'Anaconda 36](#_Toc65789743)

[9.3 Installation des container docker HADOOP, Jupyter/SPARK et CASSANDRA 37](#_Toc65789744)

[9.3.1 Copie des sources Docker 37](#_Toc65789745)

[9.3.2 Création des scripts de démarrage et d'arrêt des container 38](#_Toc65789746)

[9.3.3 Modification des droits des scripts de démarrage et d'arrêt des containers pour autoriser l'exécution 38](#_Toc65789747)

[9.3.4 Démarrage des container docker 38](#_Toc65789748)

[9.3.5 Arrêt des container docker 38](#_Toc65789749)

[9.4 Résumé des containers 39](#_Toc65789750)

[9.5 Microservices 39](#_Toc65789751)

# Contexte / Problématique

## Contexte

Le secteur électrique des Etats-Unis est en grande partie aux mains des « utilities », entreprises publiques ou privées historiquement responsables de l'approvisionnement en électricité sur le territoire d'un État.

Afin de faciliter l'accès libre au transport d’électricité et de favoriser la concurrence pour la production d'électricité, les Etats-Unis ont rendu les ordonnances 888 et 889 en 1996 pour promouvoir la création d’opérateurs indépendants, les ISO (Independant System Operator), responsables de l’exploitation du système de transport électrique. Plusieurs groupes de propriétaires de transport ont formé des ISO, certains à partir de pools énergétiques existants.

Dans l'ordonnance n ° 2000, la Commission a encouragé les services publics à rejoindre les organisations régionales de transport (RTO) qui, comme une ISO, exploiteraient les systèmes de transport et développeraient des procédures innovantes pour gérer le transport de manière équitable.



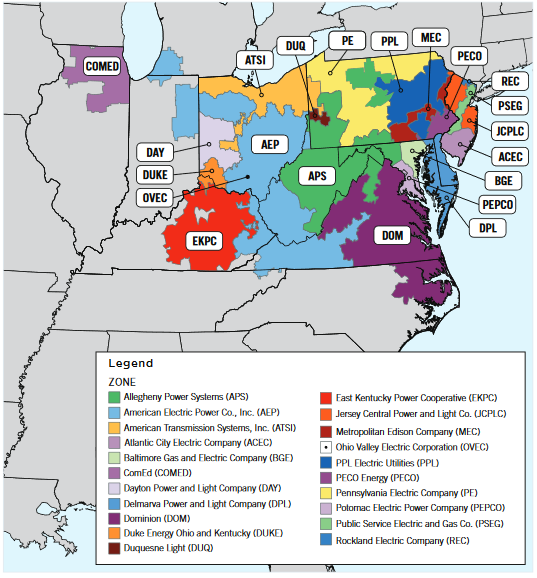
Chacun des ISO et RTO actuels ont des marchés de l'énergie et des services auxiliaires sur lesquels acheteurs et vendeurs peuvent soumissionner ou proposer de la production. Les ISO et les RTO utilisent des marchés basés sur des offres pour déterminer la répartition économique. La FERC (Federal Energy Regulatory Commission) impose un ensemble de règles aux ISO/RTO afin d’assurer une fourniture continue d’électricité répondant à la demande des consommateurs pour éviter les blackouts survenus par le passé.

Les ISO/RTO se sont réunis pour former une collaboration inter-entreprise nommée IRC (ISO/RTO Council) et se sont engagés à :

* Partager des idées,
* Favoriser l'innovation,
* Travailler ensemble pour faire progresser le développement du réseau électrique nord-américain,
* Améliorer la fiabilité et l'efficacité, tout en soutenant la durabilité continue de la fourniture, non-discriminatoire, d'énergie électrique aux consommateurs.

PJM Interconnection LLC (PJM) est un pool énergétique créé en 1927, devenu un ISO du réseau d’Interconnexion Est en 1996 et désigné RTO en 2001.

Le réseau de PJM est actuellement divisé en 24 zones. Les zones sont généralement utilisées par les sociétés de distribution d'électricité (EDC).



Le réseau PJM est composé de 12 hubs qui sont un regroupement de « bus » individuels dans une zone de tarification régional dont l’objectif est de produire un signal de prix stable sur le marché de l'énergie et d'autres systèmes et marchés divers au sein de PJM. Les consommations électriques sont mesurées/estimées au niveau des Hubs.

Un « bus » est un point d'interconnexion au système où l'énergie produite devient disponible pour le transport. Il s’agit aussi d’un conducteur électrique qui sert de connexion commune pour deux ou plusieurs circuits électriques.

## Problématiques rencontrées par les RTO

Les RTO ont besoin de :

* + - Prévoir la production d’électricité nécessaire sur les années à venir (3 ans) afin de planifier les opérations de maintenance sur les installations existantes et les investissements pour des installations supplémentaires pour faire face à la demande croissante d’électricité,
    - D’analyser les conditions météorologiques afin d’investir dans les énergies renouvelables (parcs solaires et éoliens) afin de réduire les émissions de CO2 (objectif imposé par la FERC),
    - D’évaluer différents paramètres (météo, disponibilité des générateurs d’électricité, pannes des lignes électriques) en temps réel pour répondre à la demande des consommateurs sur une base quotidienne, car l’électricité ne peut pas être stockée, tout en sélectionnant les ressources énergétiques à moindre de coût, et de manière optimale.

## Périmètre de l’étude

L’objectif de l’étude est d’analyser la consommation d’électricité depuis 2001 et de réaliser un modèle prédictif de la consommation d’électricité sur les années à venir.

# Identification des données

## Sources

### Energy Consumption (PJM’s website)

Le dataset contient la consommation de puissance électrique heure par heure en mégawatts (MW) par compagnie (hubs). Les compagnies évoluent au fil des années, par conséquent les données ne sont pas renseignées pour certaines dates.

|  |  |
| --- | --- |
| **Source** | PJM website : <https://www.pjm.com/>  PJM Data Miner IHM : <https://dataminer2.pjm.com/feed/load_frcstd_hist> |
| **Caractéristiques** | Estimation des consommations d’énergie par heure en MegaWatts avec une colonne par compagnie d’électricité :   |  |  | | --- | --- | | **Attribut** | **Désignation** *compagnie (secteur – état)* | | AEP | American Electric Power (Western - OHIO) | | ComEd | Commonwealth Edison (Western - Illinois) | | DAYTON | Dayton Power and Light Company (Western - OHIO) | | DEOK | Duke Energy Ohio/Kentucky (Western - OHIO) | | DOM | Dominion Virginia Power (South – Virginia) | | DUQ | Duquesne Light Co. (Western - OHIO) | | EKPC | East Kentucky Power Cooperative (South - Kentucki) | | FE | FirstEnergy (East - Pennsylvania) | | NI | Northern Illinois Hub (North - Illinois) | | PJME | PJM East Region: 2001-2018 (East) | | PJMW | PJM West Region: 2001-2018 (West) | | PJM\_Load | PJM Load Combined: 1998-2001 | |
| **Fréquence** | Par heure, tous les jours calendaires de l’année |
| **Période** | 2001 à 2018 |
| **Technologie** | Fichier initial parquet  API sur le site « https://www.eia.gov/opendata » |

### Weather forecast (PJM’s website)

[Le](https://www.kaggle.com/selfishgene/historical-hourly-weather-data)s données météorologiques peuvent être utiles dans les prédictions de consommation électriques (température, humidité) mais ne sont pas traitées dans le cadre de cette étude.

## Sélection des données

La prédiction de consommation d’énergie électrique va être basée sur la consommation totale d’énergie électrique gérée par la société PJM (somme des consommations par heure de toutes les compagnies).

### Chaîne de traitement des données

VM DEV

PJM website

HDFS

CASSANDRA

<< Micro service >>

Collecte des données sources

<< Micro service >>

Préparation des données

<< Micro service >>

Evaluation des modèles

Modèles .h5

CSV brutes

<< Micro service >>

Prédiction données courantes

# Méthodologie

Analyse

Préparation des données

Exploration des données

Evaluation des modèles

Sélection du modèle

Déploiement

Extraction des informations et statistiques descriptives

Nettoyage des données

Imputation des valeurs manquantes

Visualisation des données

Contrôle des caractéristiques des données

Sélection des features

Choix des modèles

Entraînement des modèles

Sélection du meilleur modèle

Déploiement du modèle pour l’aide à la décision

# Préparation des données

La préparation des données est une partie critique du traitement car toutes les étapes suivantes s’appuieront sur l’ensemble de données produit à ce niveau.

Ici on s’intéressera aux étapes clefs pour préparer correctement ses séries: gérer la composante temporelle, détecter les mauvaises valeurs et les remplacer.

## Gérer la composante temporelle

Une bonne technique pour éviter les casses têtes au niveau des formats temporels est de stocker toutes les composantes temps dans un format unique numérique de type ‘timestamp’ dans le même fuseau horaire (typiquement UTC). Ainsi toutes les dates et temps ont un sens clair et non ambigu.

Notre dataset étant sur un seul fuseau horaire, nous n’appliquons pas cette solution de timestamp.

Finalement, on conseille très fortement de régulariser sa série temporelle, c’est-à-dire de rendre tous les intervalles de temps constants.

Après analyse du dataset (119064 lignes), nous avons rajouté 24 lignes pour des heures manquantes.

## Enlever les mauvaises données

Nous avons supprimé 4 doublons au niveau des heures.

Les séries temporelles : préparation et exploration des données

https://www.actuia.com/actualite/les-series-temporelles-preparation-et-exploration-des-donnees/

## Remplacer les mauvaises données

Aucune mauvaise données identifiées

# Exploration des données

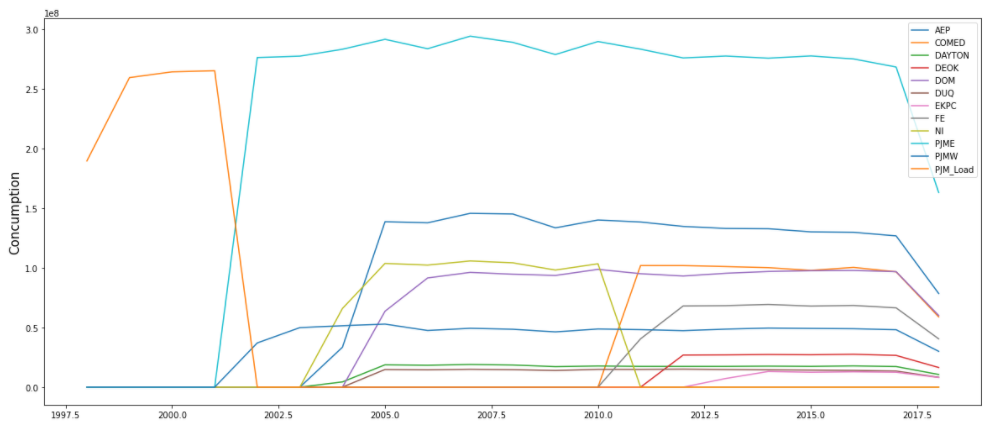
L’exploration de données temporelles « Time Series » est un processus que l’on peut décomposer en plusieurs étapes, pour cette étude, on commencera par visualiser la distribution et l’évolution de notre série temporelle. Ensuite on présentera la stationnarité, concept essentiel en analyse temporelle, puis nous passerons à la décomposition des séries temporelles et finalement étudier des dépendances pour les analyses multivariées.

## Visualition de la série temporelle

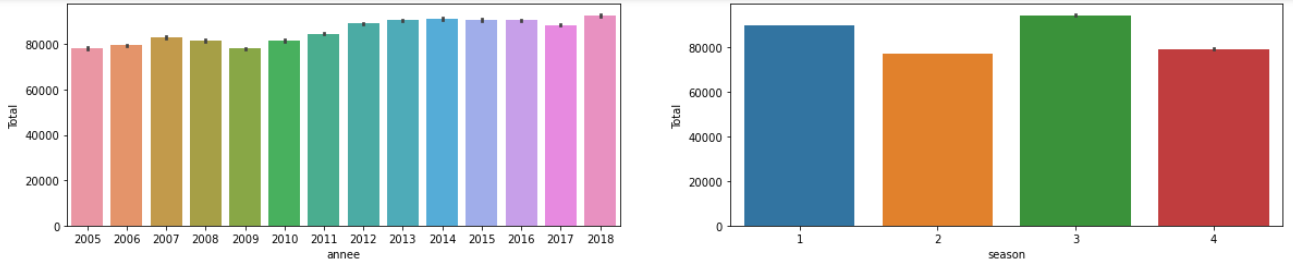
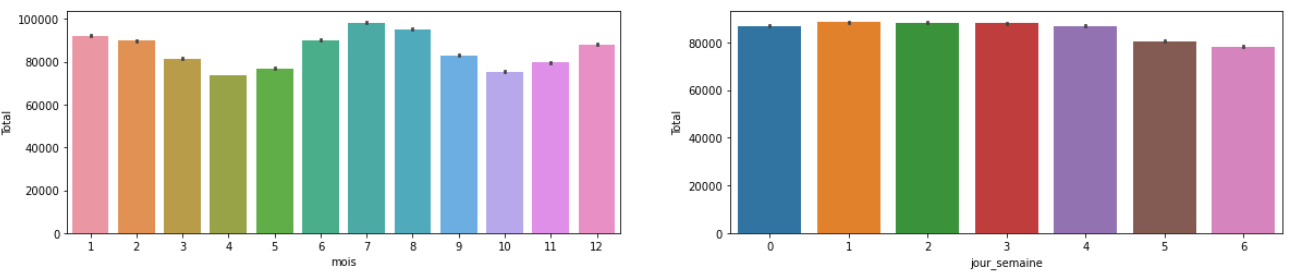
### Distribution de la consumation par compagnie

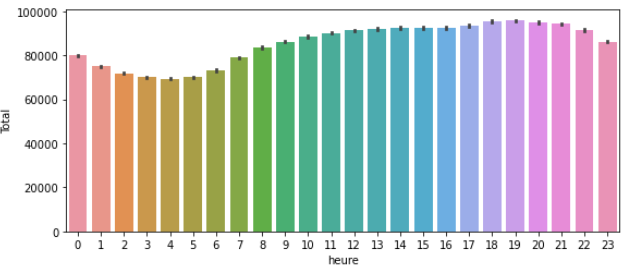


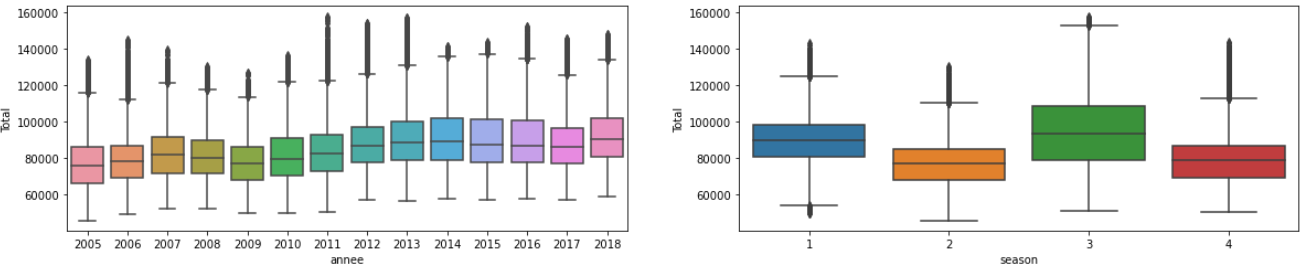
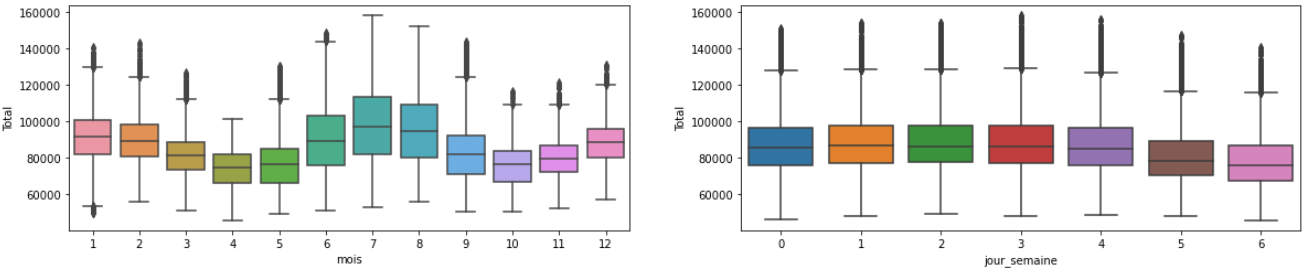
### Evolution annuelle de la consommation par compagnie

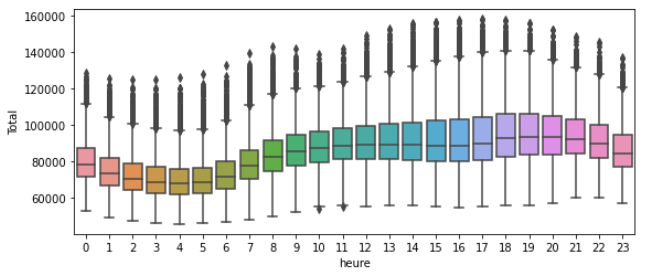


### Evolution de la consommation totale





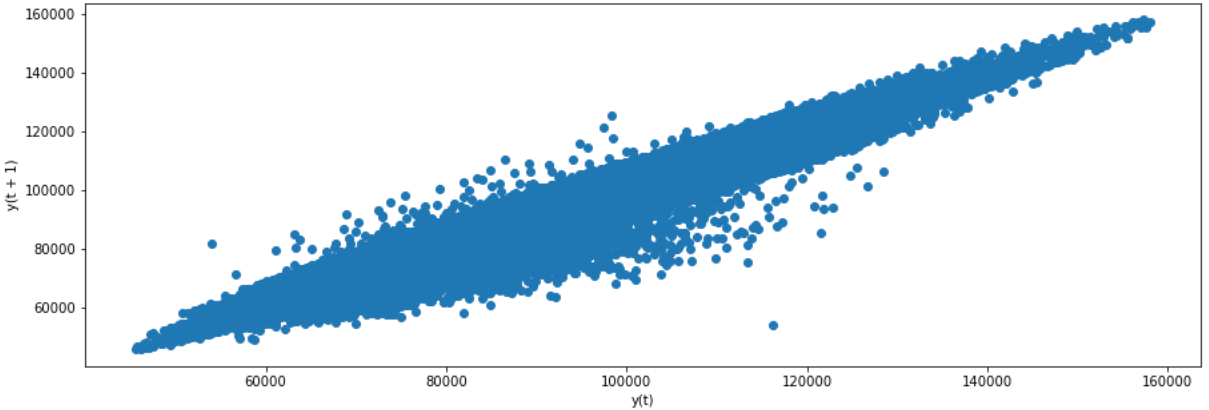
NOTE :

1. La consommation est en augmentation permanente depuis 2005
2. On note une chute importante de consommation en 2008, probablement liée a la crise bancaire du printemps 2008.
3. La consommation est plus élevée en été (juillet) et plus faible eu printemps
4. La consommation est plus faible le week-end
5. La consommation augmente après 5h du matin pour attendre un maximum à 18h et commence a baisser à partir de 18h.

### Corrélation de la série temporelle (Lag Scatter plot)

La modélisation de séries chronologiques suppose une relation entre une observation et l'observation précédente.

Pour étudier la corrélation de nos données, nous traçons la consommation totale a l’instant (t) sur l’axe des abscisses et la consommation totale a l’instant (t+1) sur l’axe des ordonnées.



Les points se regroupent le long d’une diagonale allant du bas gauche vers le haut à droite, cela suggère une relation de corrélation positive.

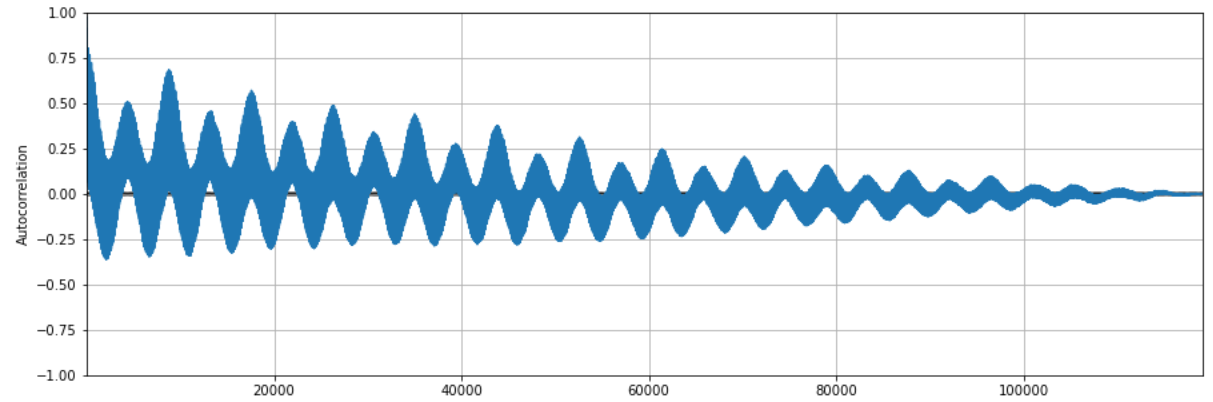
### Saisonnalité (Autocorrélation plots)

L’autocorrélation est la moyenne temporelle du produit du signal par lui-même décalé d'un temps τ.

C’est un outil très utilisé pour l’analyse de saisonnalité de séries temporelles.

Un graphique de L’autocorrélation peut être créé pour aider à mieux comprendre comment la relation entre deux intervalles de temps change dans le temps.

L’observation utilisée pour cette étape est le total de consommation par heure.



Le signale périodique montré ci-dessus, est un signe fort de la saisonnalité de l’ensemble de notre série temporelle.

## Stationnarité

La stationnarité signifie que les statistiques de la série temporelle ne dépendent pas du temps. En particulier cela veut dire qu’il n’y a pas de tendance générale et que les variations apparaissent comme d’amplitude constante. Cela signifie que la variance et moyenne ne change pas au cours du temps.

Il y a 3 procédés pour identifier si le dataset est stationnaire :

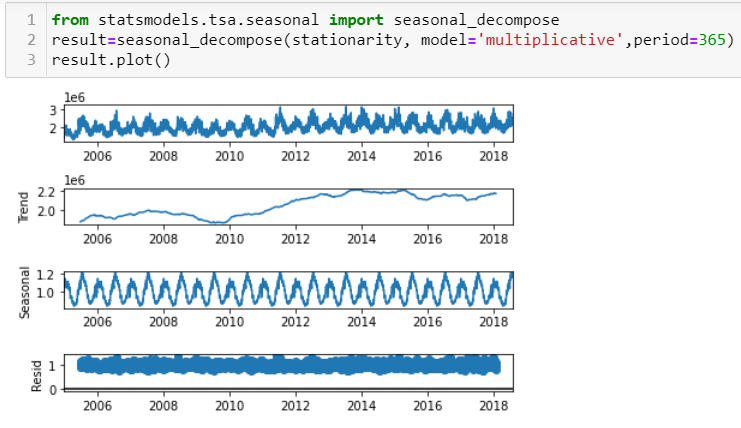
* Visuel: graphique
* Statistique de base: comparant la moyenne et la variance
* Test Statistique: Augmented Dickey Fuller test

Dans notre étude, nous utilisons le visuelle et le Test Statistique.

**Visuel:**

Un outil de decomposition de series temporelles permet de visusaliser en 3 graphiques si il se dégage:

* Une tendance
* Une variance (seasonal)
* Un bruit



NB: A première vue, nous pourrions penser que nous avons une tendance et une variance mais l’échelle du graphique nous induit en erreur.

Pour valider, que le dataset est stationaire nous avons utilisé le Test Statistique

**Statistique Test: ADF**

**Augmented Dickey-Fuller (ADF)** est un type de test pour identifier la non-stationnarité de notre dataset.

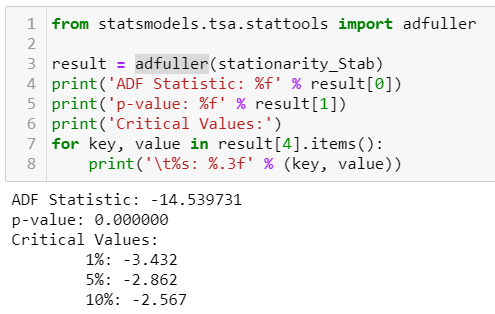
* + - Null Hypothesis (H0): Time series non stationaire.
    - Alternate Hypothesis (H1): Time series stationaire

**Si H0 peut être rejeté, nous concluerons que le time series est stationnaire.**

Il y a deux chemins pour rejeter le H0:

* if p-value < 5%
  + - p-value > 0.05: dataset est non-stationnaire.
    - p-value <= 0.05: dataset est stationnaire.
* if t-test < valeur critique
  + - ADF statistique > valeur critique: dataset est non-stationnaire
    - ADF statistique < valeur critique: dataset est stationnaire.

Ci-dessous le resultat mentionnant notre dataset comme stationnaire (p-value=0):



# Evaluation des modèles prédictifs

Les modèles sélectionnés suivants sont adaptés aux analyses prédiction de séries temporelles et triés par ordre de complexité :

* + - KNN Regressor : k-Nearest Neighbors pour analyse de régression
    - XGBoost : eXtreme Gradient Boosting
    - ARIMA : AutoRegressive Integrated Moving Average
    - SARIMA : Seasonnal AutoRegressive Integrated Moving Average
    - LSTM: Long short-term memory (recurrent neural network)

Tous les modèles utilisent le même dataset nettoyer et préparer lors des étapes précédentes.



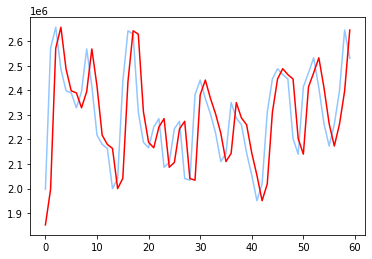
Modélisation et évaluation des séries temporelles

<https://www.actuia.com/actualite/modelisation-et-evaluation-des-series-temporelles/>

## Persistance

La première étape avant de passer à l'analyse et la modélisation des données est d'établir une référence de performance. Cela fournira à la fois un modèle pour évaluer les modèles en utilisant les "harnais'" de test et une mesure de la performance par laquelle tous les modèles prédictifs plus élaborés peuvent être comparés.

La prédiction de base pour la prévision de séries chronologiques est appelée prévision naïve, ou persistance. C'est là que l'observation du pas de temps précédent est utilisée comme prédiction pour l'observation au prochain pas de temps. (valeur prédite = valeur du timestep précédent)



Le RMSE du test de persistance est de 147466[¶](http://localhost:8890/notebooks/Documents/Data%20Scientist/Projet/Energy_consumption_prediction_notebook_v1kamel.ipynb#Le-RMSE-du-test-de-persistence-est-de-147466)

Donc en moyenne la valeur prédite a une erreur de 147466 MW

Cette mesure métrique RMSE sera notre référence pour le choix final de notre modèle.

## Modèle kNN Regressor

### Définition

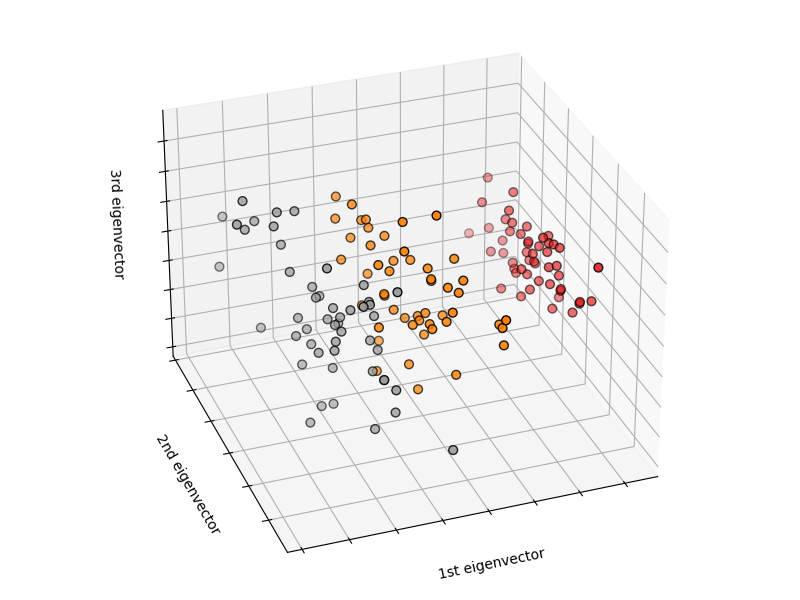
L’algorithme K-NN (K-nearest neighbors) est une méthode d’apprentissage supervisé. Il peut être utilisé aussi bien pour la régression que pour la classification. Son fonctionnement peut être assimilé à l’analogie suivante *“dis moi qui sont tes voisins, je te dirais qui tu es…”*.

Pour effectuer une prédiction, l’algorithme K-NN va se baser sur le jeu de données en entier. En effet, pour une observation, qui ne fait pas partie du jeu de données, qu’on souhaite prédire, l’algorithme va chercher les K instances du jeu de données les plus proches de notre observation.

Ensuite pour ces **k** voisins, l’algorithme se basera sur leurs variables de sortie *(target) y pour calculer la valeur de la variable* ***y*** *de l’observation qu’on souhaite prédire.*

Par ailleurs :

* Si K-NN est utilisé pour la régression, c’est la [moyenne (ou la médiane)](https://mrmint.fr/exploration-donnee-python) des variables **y** des **K** plus proches observations qui servira pour la prédiction
* Si K-NN est utilisé pour la classification, c’est le [mode](https://fr.wikipedia.org/wiki/Mode_(statistiques)) des variables des plus proches observations qui servira pour la prédiction



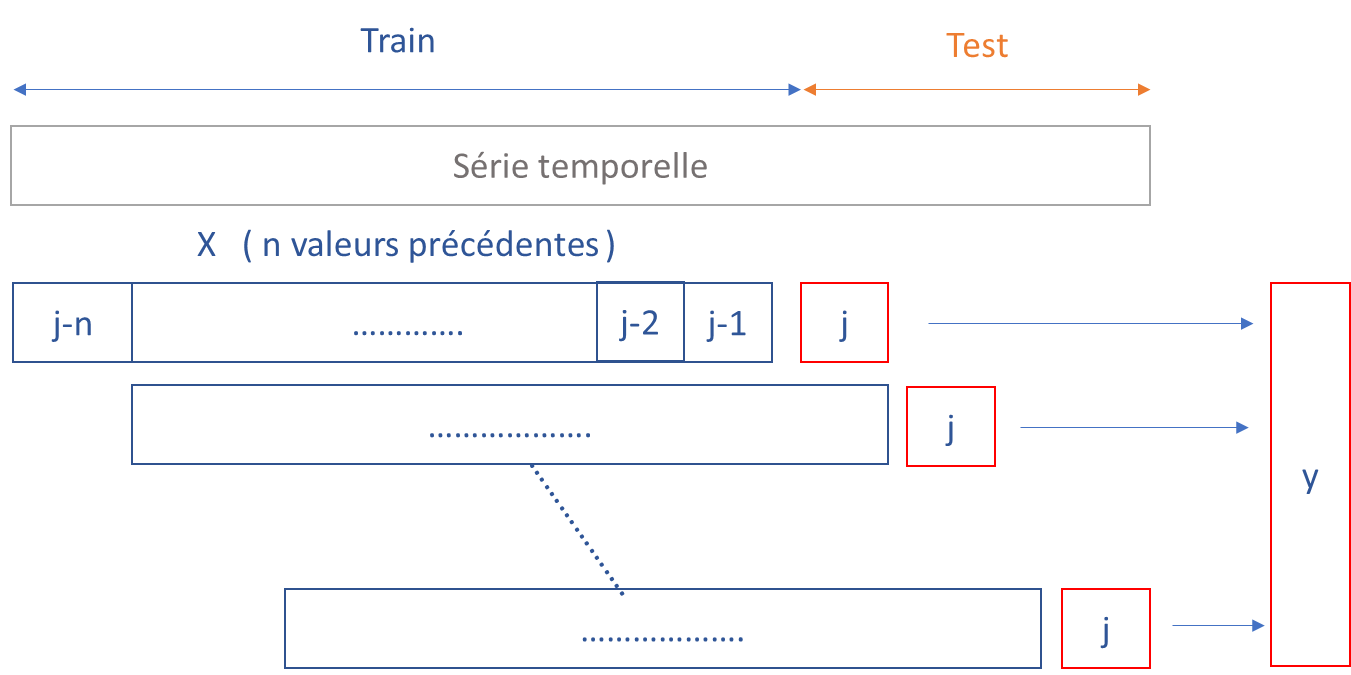
### Etude

L’algorithme utilisé pour l’évaluation est le modèle KNeighborsRegressor de SKLearn.neighbors.

#### Features et hyperparamètres

L’évaluation de ce modèle a été réalisée en découpant la série temporelle d’entrée en échantillons de n jours pour les features et le n+1eme jour pour la target (valeur à prédire). Les échantillons sont créés en réalisant un décalage d’un step à chaque fois.

Les évaluations ont été faites pour plusieurs valeurs de n, qui correspondent à différentes périodicités : 1,2,3,4,5,6,7 (semaine) ,15 ,30 (mois) ,90 (trimestre) ,180 (semestre),365 (année)



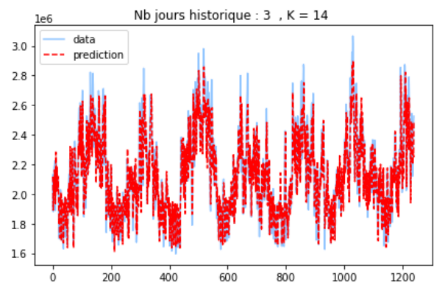
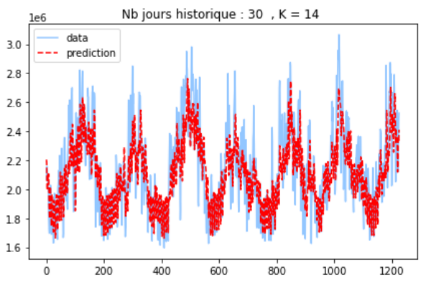
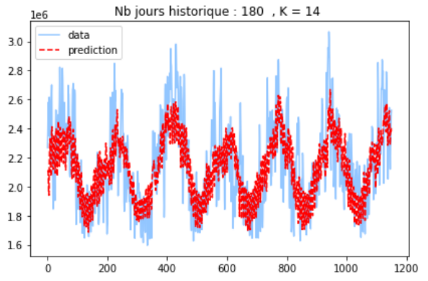
La recherche de la meilleure performance de ce modèle a été réalisée en effectuant un GridSearch pour différents paramètres de K et 1 à 14 et pour chaque « période » n des échantillons de X, afin de trouver le meilleur K pour chaque « période ».

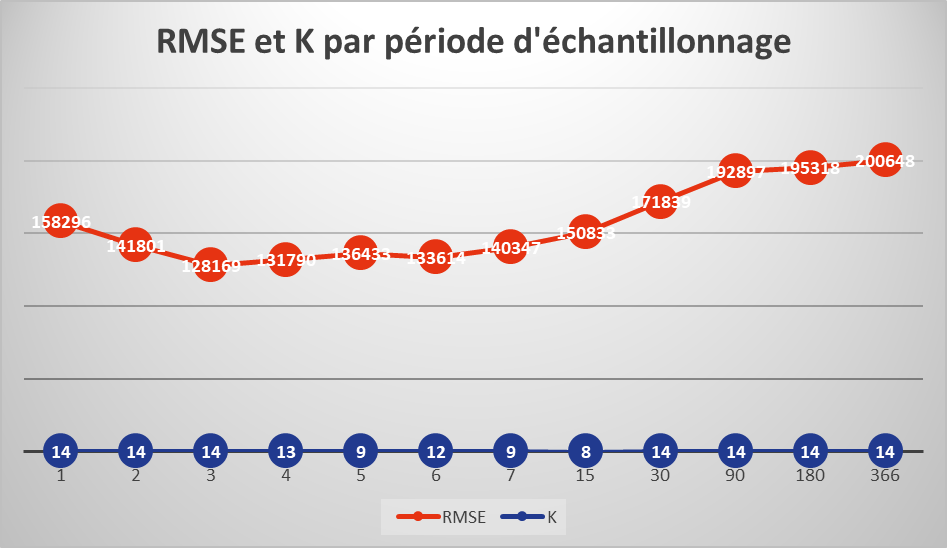
Ensuite, une prédiction sur l’échantillon de test est réalisée en prenant comme paramètre le meilleur K de chaque « période » n. Puis le RMSE est calculé en comparant les données prédites et les données réelles.

#### Résultats

Les meilleurs résultats sont obtenus avec des échantillons sur des périodes courtes : 3 à 6 jours.

Ce résultat s’explique par le fait que plus les données sont nombreuses plus la valeur prédite se concentre vers la moyenne de l’échantillon, ce qui augmente les erreurs.



Le meilleur **RMSE** pour ce modèle est de **128169**.

Il est obtenu avec des échantillons de périodes de 3 jours.

## Modèle XGBoost

### Definition: XGBoost

XGBoost signifie eXtreme Gradient Boosting.

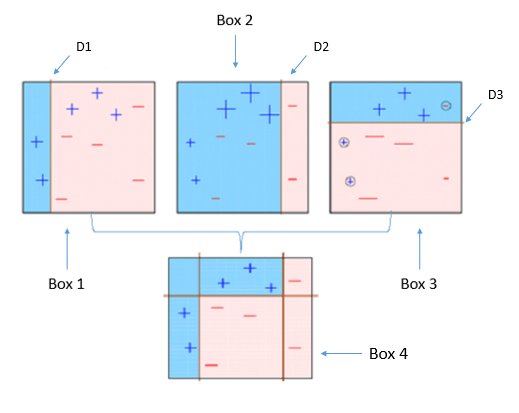
Il est l’un des plus populaire algorithme de Machine learning (en regression ou classification). C’est une implémentation open source optimisée de l’algorithme d’arbres de boosting de gradient.

Le Boosting de Gradient est un algorithme d’apprentissage supervisé dont le principe et de combiner les résultats d’un ensemble de modèles plus simple et plus faibles afin de fournir une meilleur prédiction.

On parle d’ailleurs de méthode d’agrégation de modèles. L’idée est donc simple : au lieu d’utiliser un seul modèle, l’algorithme va en utiliser plusieurs qui serons ensuite combinés pour obtenir un seul résultat.

C’est avant tout une approche pragmatique qui permet donc de gérer des problèmes de régression comme de classification.

Illustration simple pour comprendre le boosting:



4 classifiers (dans 4 boxes) ci-dessus, essayant de classifier plus ou moins de classes de la facon la plus homogène.

La box 4 est une combinaison de poids des box classifiés 1,2,3. Comme on peut le voir, il fait un bon travaille de classification en séparant les points correctement.

Dans le cadre de ce projet avec Time Series, nous allons utilisé le XGBoost pour résoudre ce probleme de régression.

### Etude

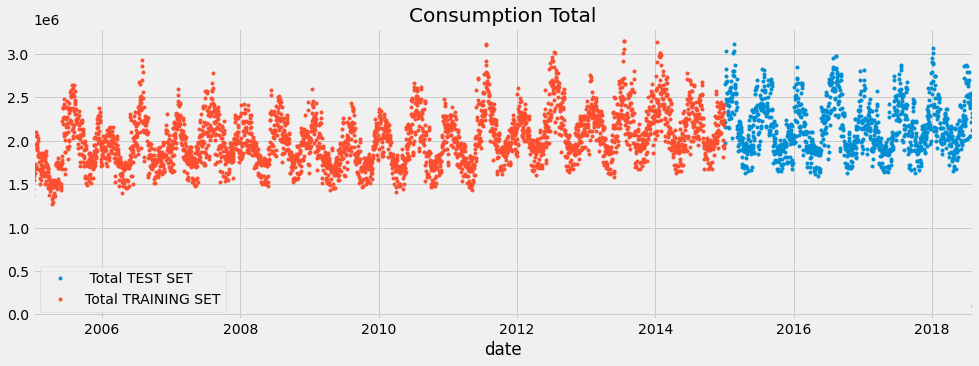
Deux études d’application du XGBoost vont être faite avec des datasets différents:

* Dataset à granularité fine jusqu’à l’heure
* Dataset à granularité macro jusqu’à la date

#### Dataset macro (sans les heures) avec les dates

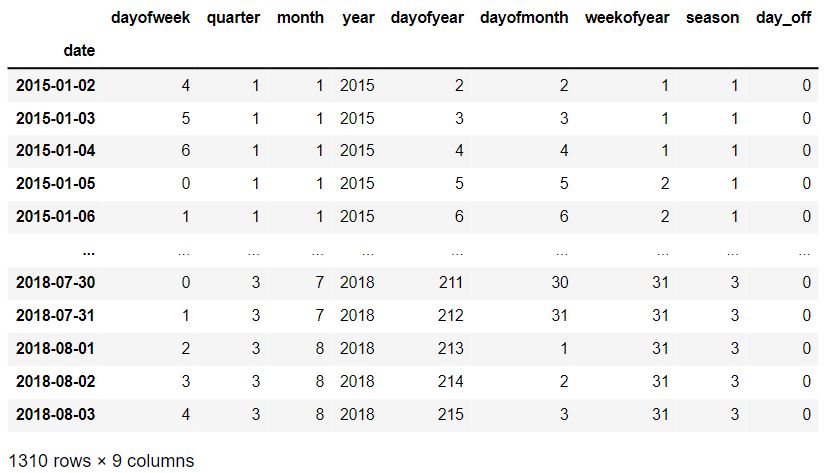
##### Train/Test Split

On va couper les données après 2015 afin de l’utiliser comme notre jeu de validation.



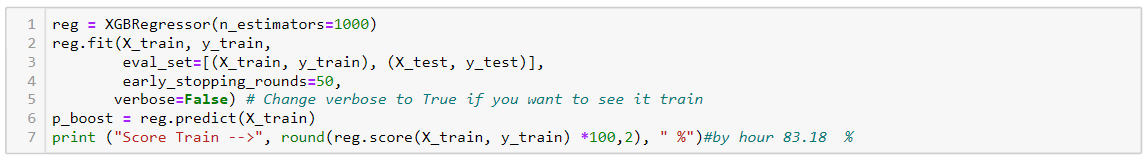
##### Creation des features pour notre modèle

On va créer des critères de période qui aideront notre modèle de machine learning: Date, jour de semaine, mois, trimestre, année, jour de l’année, jour du mois, semaine de l’année.



On relance un Train/Test Split avec les nouvelles features.

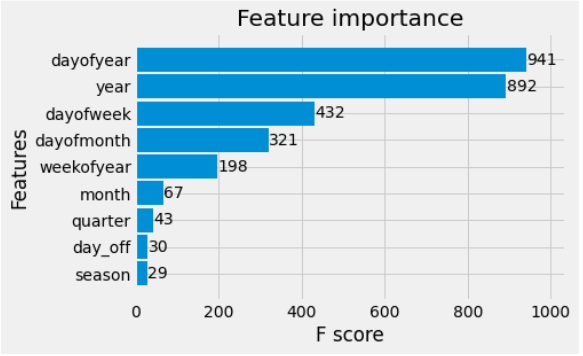
##### Creation du modèle XGBoost



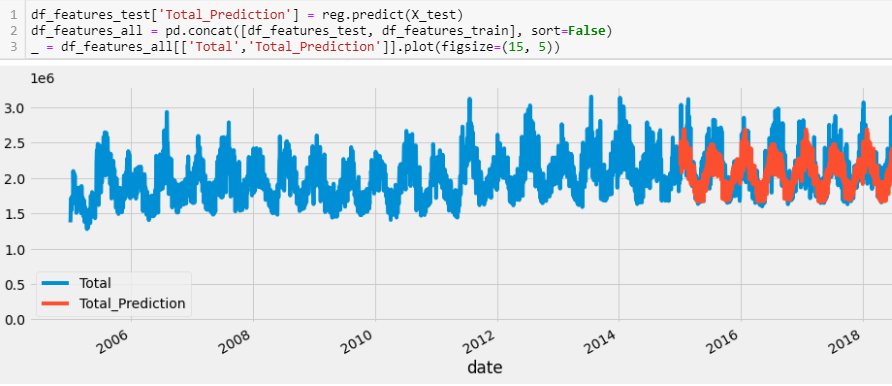
##### Evaluations

XGBoost founit quelques métriques intéressantes.

Graphique présentant les champs par degré d’importance : dayofyear,year,hour,dayofweek



##### Prediction sur le jeu de Test

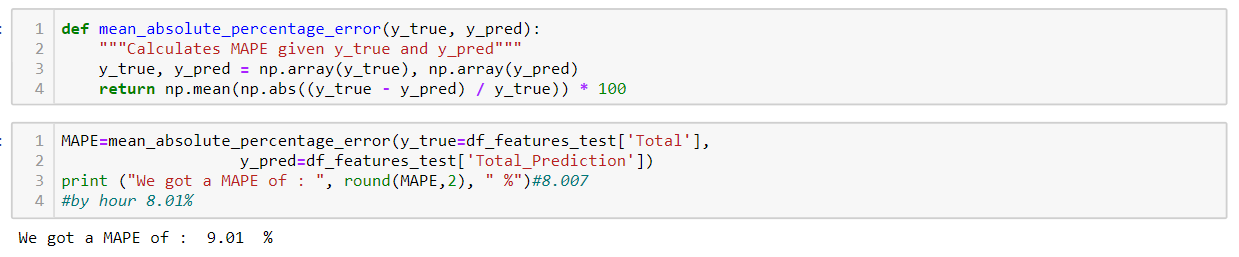


##### Erreur Metriques sur jeu de Test

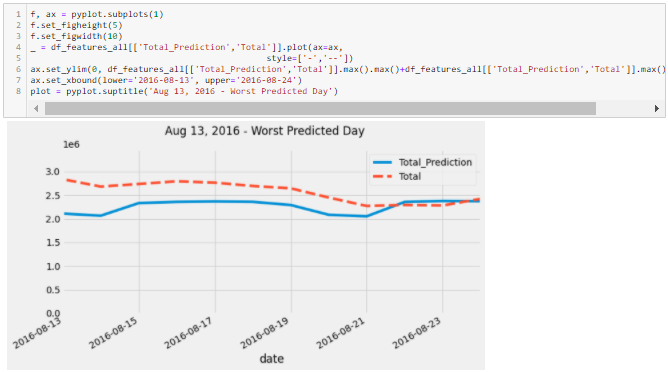
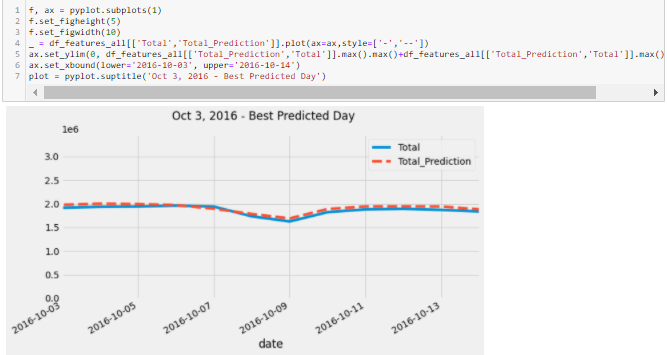
MAPE: Mean Absolute Percentage Error permet d’interpréter avec une valeur en pourcentage l’écart de prédiction.

MAPE=> 9.01%

NB: MAPE n’est pas inclus dans sklearn donc une fonction doit être créée



##### Exploration de la meilleur et moins bonne prédiction

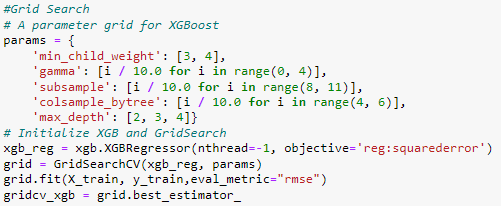


##### Optimisation du modèle XGBoost

Le XGBoost contient comme tous machine learning un lot d’hyperparamètres (plus de détail ici <https://xgboost.readthedocs.io/en/latest/parameter.html>), un exemple ci-dessous:

* params['booster'] = 'gbtree'
* params['objective'] = 'binary:logistic'
* params["eval\_metric"] = "error"
* params['eta'] = 0.3
* params['gamma'] = 0
* params['max\_depth'] = 6
* params['min\_child\_weight']=1
* params['max\_delta\_step'] = 0
* params['subsample']= 1
* params['colsample\_bytree']=1
* params['silent'] = 1
* params['seed'] = 0
* params['base\_score'] = 0.5

L’outil GridSearch va permettre d’optimiser ces hyperparamètres et nous fournir les valeurs à assigner pour chacun de ces hyperparmètres



Réultat du best\_estimator\_:

{'colsample\_bytree': 0.4,

'gamma': 0.0,

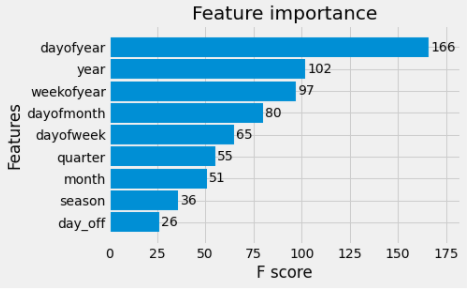
'max\_depth': 3,

'min\_child\_weight': 3,

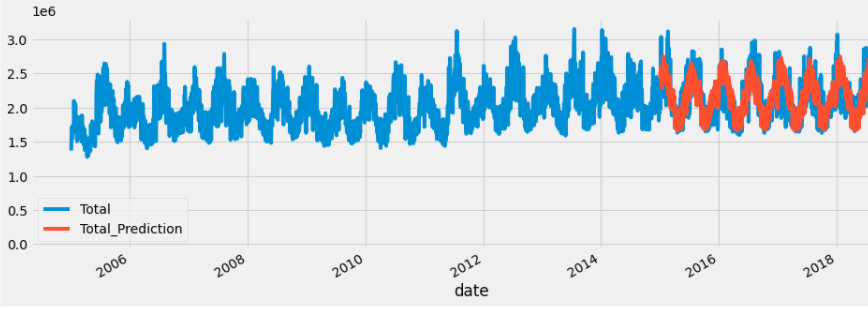
'subsample': 0.8}

##### Evaluation du modèle XGBoost optimisé

On peut s’apercevoir que l’ordre d’importance des critères n’est plus le même.



La prédiction n’est pas bien meilleur que la précédante



Erreur Métriques

* RMSE : 231 414 KW (bien au dessus de notre RMSE de reference du modele persistent)
* MAPE: est équivalente au premier modèle environ de 9%

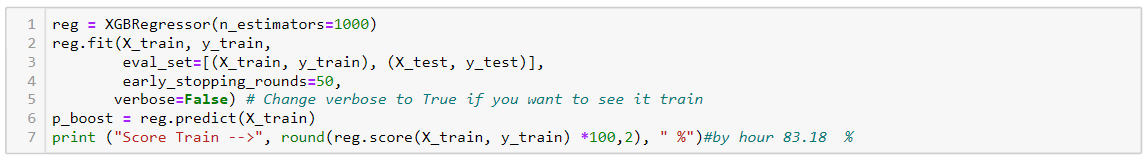
#### Dataset micro avec les heures dans les features

##### Creation des features pour notre modèle

On remarquera ici le feature ‘Hour’

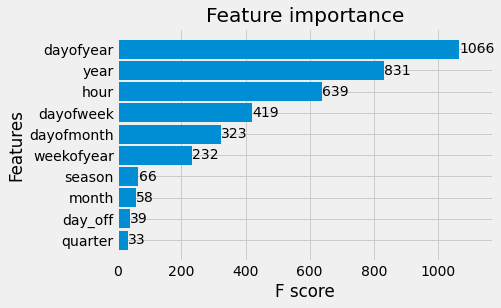


##### Creation du modèle XGBoost



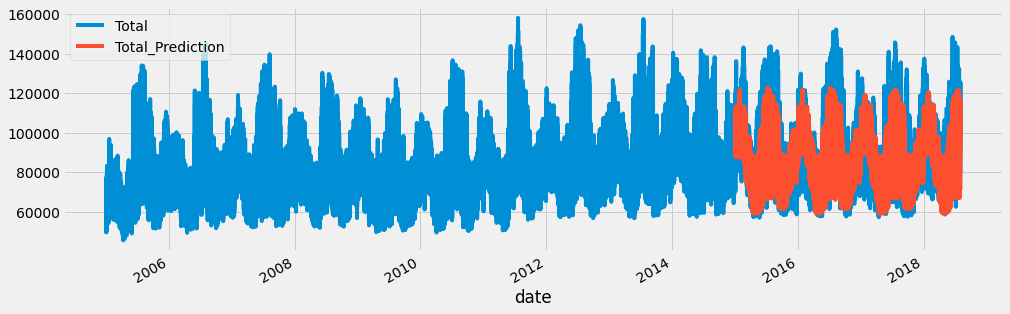
##### Evaluations

On perçoit ici l’importance de la feature ‘Hour’ dans notre modèle



##### Prediction sur le jeu de Test

On percoit dejà une meilleur prédiction au dataset sans les heures (cf. Graphique 5.1.2.1.5)



##### Erreur Metriques sur jeu de Test

MAPE: Mean Absolute Percentage Error permet d’interpréter avec une valeur en pourcentage l’écart de prédiction.

MAPE=> 8.01%

Le pourcentage s’est amélioré de 1% au dataset sans les heures

##### Optimisation du modèle XGBoost

L’outil GridSearch va permettre d’optimiser ces hyperparamètres et nous fournir les valeurs à assigner pour chacun de ces hyperparmètres

Réultat du best\_estimator\_:

{'colsample\_bytree': 0.4,

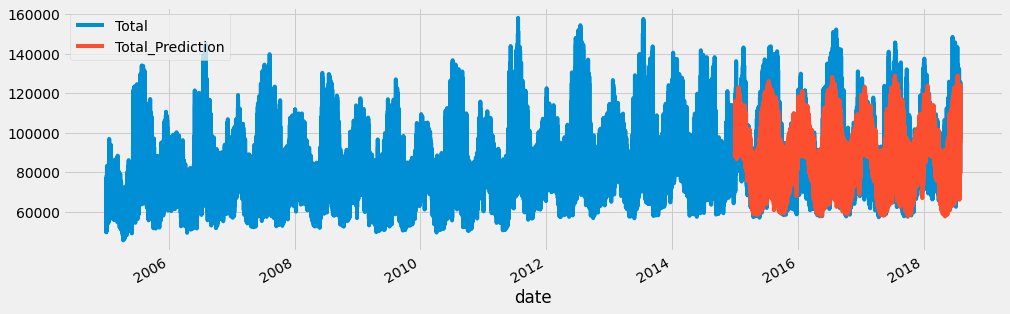
'gamma': 0.0,

'max\_depth': 3,

'min\_child\_weight': 3,

'subsample': 0.8}

##### Evaluation & Mesure du modèle XGBoost optimisé



Erreur Métriques MAPE: est équivalente au premier modèle environ de 8%.

Nous ne chercherons pas à optimiser les hyperparamètres de ce machine learning au vue du temps restant et de la belle performance des autres Machines Learning ci-dessous

## Modèle ARIMA

Les modèles de la famille ARIMA (Auto Regressive integrated Moving Average) permettent de représenter sous une forme succincte certains phénomènes variant avec le temps (Time Séries), et de faire des prévisions pour les valeurs futures du phénomène, avec un intervalle de confiance autour des prévisions.

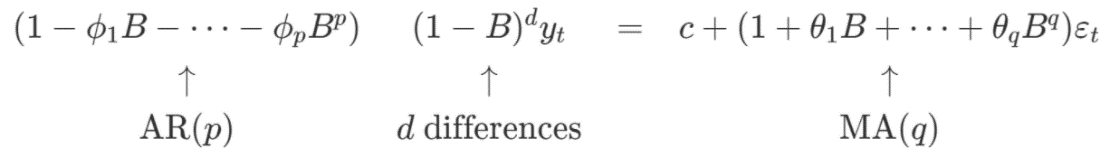
L'analyse des données avec le modèle ARIMA suppose que la série temporelle stationnaire.

Le modèle ARIMA nécessite trois hyperparamètres (p, d, q):

**p :**  termes autorégressif: Nombre de points nécessaires pour décrire l’instant t

**d :** Nombre de différenciations pour arriver à un état stationnaire

**q:** est le nombre de moyennes mobiles.



### Recherche des meilleurs Hyper Paramètres

Avant de lancer l’entrainement du modèle ARIMA, on doit chercher les Hyperparamètres qui donnent un meilleur résultat.

Pour cela on utilise deux méthodes différentes :

* Fonctions d’autocorrélation et Autocorrélation partielle
* Grid Search

### Méthode d’autocorrélation et corrélation partielle

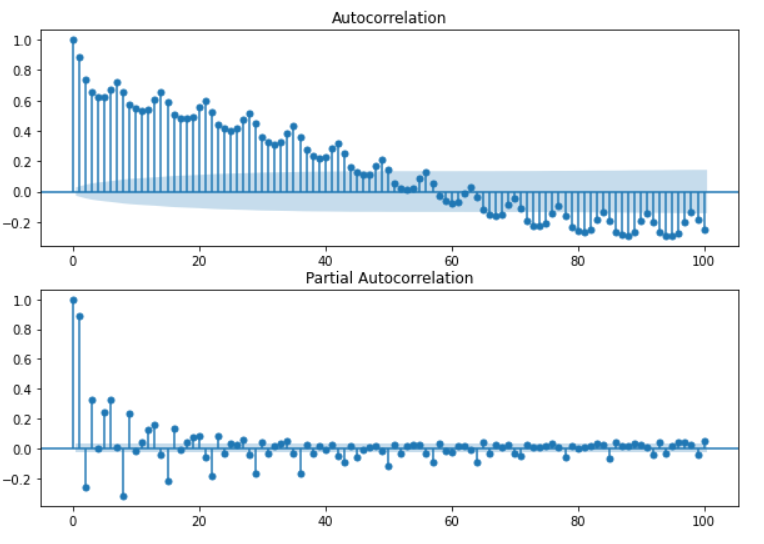
L'autocorrélation et l'autocorrélation partielle sont des mesures de l'association entre des valeurs de séries actuelles et passées ; elles indiquent les valeurs de séries passées les plus utiles à la prévision de valeurs futures. Avec ces données, vous pouvez déterminer l'ordre des processus d'un modèle ARIMA. De façon plus spécifique :

* **Fonction d'autocorrélation (ACF). Au décalage k:** il s'agit de la corrélation entre les valeurs de séries séparées par k intervalles.
* **Fonction d'autocorrélation partielle (PACF). Au décalage k:**  il s'agit de la corrélation entre les valeurs de séries séparées par k intervalles, compte tenu des valeurs des intervalles intermédiaires.

L'axe x du tracé ACF indique le décalage auquel l'autocorrélation est calculée ; l'axe y indique la valeur de la corrélation (entre 1 et 1). Par exemple, une pointe au décalage 1 dans un tracé ACF indique une forte corrélation entre chaque valeur de série et la valeur précédente ; une pointe au décalage 2 indique une forte corrélation entre chaque valeur et la valeur apparaissant deux points auparavant, etc.

* Une corrélation positive indique que des valeurs actuelles élevées correspondent à des valeurs élevées au niveau du décalage spécifié ; une corrélation négative indique que des valeurs actuelles élevées correspondent à des valeurs faibles au niveau du décalage spécifié.
* La valeur absolue d'une corrélation est une mesure de la force de l'association, des valeurs absolues élevées indiquant des relations plus fortes.

La figure ci-dessous représente l'évolution de la corrélation de notre série temporelle en fonction du décalage journalier (lag).



Les deux graphes ci-dessus nous montre que les meilleurs paramètres (p,d,q) à prendre pour entrainer le modèle ARIMA sont (44,1,4).

Le modèle ARIMA (44,1,4) est très gourmand en termes de ressources de calcul, l’entrainement du modèle avec les ressources dont on dispose prendra beaucoup de temps, pour cette raison on a utilisé une autre méthode de recherche d’hyperparamètres (Grid Search) en limitant l’intervalle de ces derniers.

### Grid Search:

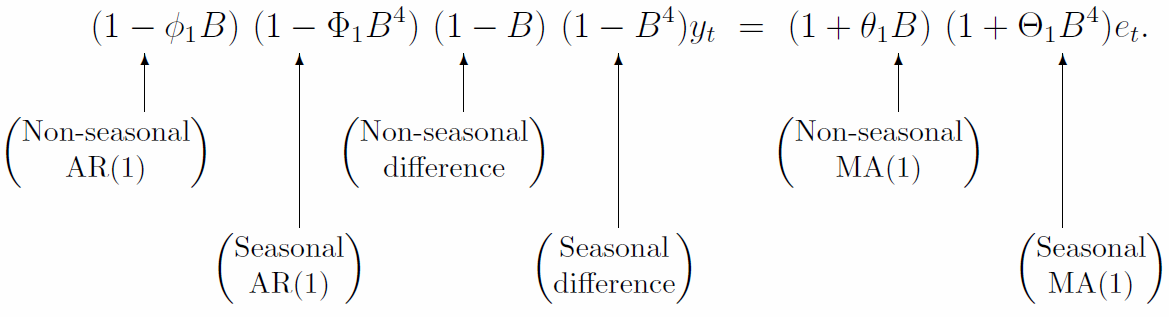
Le Grid search consiste évaluer le modèle ARIMA sur différents sur différents hyperparamètres pour en tirer les meilleurs.

En utilisant cette méthode avec la liste d’hyperparamètres suivant p=[0,7], d=[0,0], q[0,7], le meilleur modèle calculé hyperparametres calculés sont P=7, d=0, q=5 avec un **RMSE (Root Mean Squar Error) de 100944.**

## Modèle SARIMA

### Le modèle SARIMA est une version alternative du modèle ARIMA conçue spécialement pour prendre en considération les effets de saisonnalité.

La saisonnalité est une autre forme de non stationnarité qu’il faut prendre en considération dans le modèle afin de définir les poids optimums du modèle.



Le modèle SARIMA introduit 4 (P, D, Q, M) nouveaux paramètres par rapport au modèle ARIMA qui sont les suivants :

P : partie autorégressive de la saisonnalité

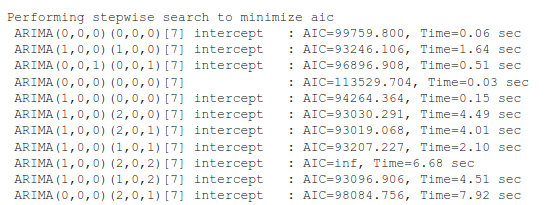
D : Nombre de différenciations nécessaire à enlever la saisonnalité

Q : partie moyenne amovible de la saisonnalité

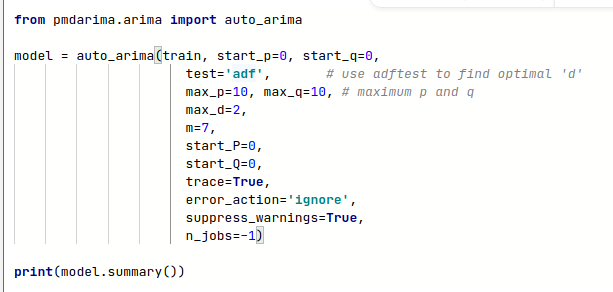
M : saisonnalité, typiquement pour un relevé quotidien elle est de 7 pour un relevé annuel : 12.

Dans notre cas, on a un relevé journalier, on fixe donc M à 7.

Le reste des paramètres est défini en lançant une optimisation automatique avec la libraire **“pmdarima”** et son module auto\_arima.



Le module auto\_arima teste différents paramètres et cherche à minimiser le paramètre AIC qui calcule la perte d’information.



Afin de définir le paramètre d, la fonction effectue un test adf ou kpss de stationnarité.

Deux modèles sont retenus suivant le test statistique effectué.

SARIMA (5,1,2,2,0,0,7) avec un test kpss

SARIMA (3.,0,3,2,0,1,7) avec un test adf

### Résultats :

On compare les 3 modèles sur une période de 90 jours :

Le modèle SARIMA (5,1,2,2,0,0,7) se montre le plus performant avec une valeur RMSE de 80900 contre une valeur de 93800 pour le modèle ARIMA (7,5,0)

## Modèle LSTM (Long Short-Term Memory)

### Dans cette partie nous allons construire un reseau de Nerones.

### Un réseau de neurones classique permet de gérer les cas ou les expériences sont independentes les unes des autres.

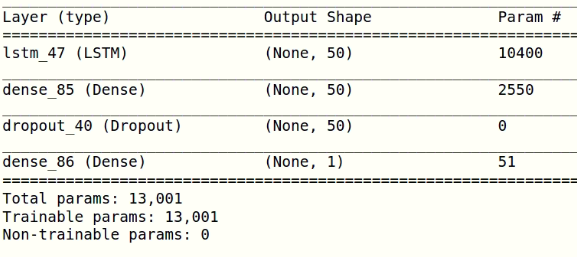
### Lorsque les expériences sont sous la forme de séquences temporelles, une nouvelle structure a été inventée : les réseaux de neurones récurrents.

### Cette nouvelle structure introduit un mécanisme de mémoire des entrées précédentes qui persiste dans les états internes du réseau et peut ainsi impacter toutes ses sorties futures. Le réseau de neurones Long Short-Term Memory (LSTM) est l’un des plus connu.

Comment pour les modèles précédents, plusieurs modèles LSTM ont été testés, on retient ici le meilleur :

Le modèle retenu est constitué d’une couche LSTM de 50 neurones, une couche dense de 50 neurones, une couche dropout pour éviter un sur apprentissage et à la sortie un réseau dense d'un neurone pour recevoir une valeur de prédite.

Le



Le même modèle a été entrainé de deux manières différentes.

Sur des plages de 90 et 120 jours pour prédire le jour suivant.

Les résultats obtenus pour ces deux modèles sont

**90 jours : RMSE 129027,742**

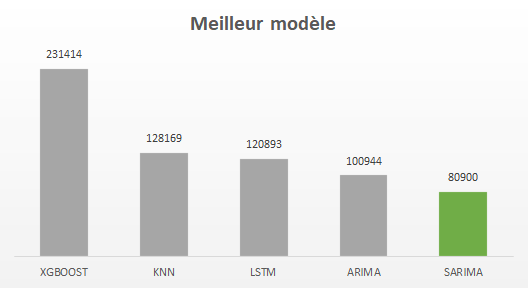
**120 jours : RMSE 120893,289**

Le modèle entrainé sur une plage de 120 jours prends plus de temps à entrainer mais à une meilleure performance car il capte une information plus complète sur la saisonnalité de la time série.

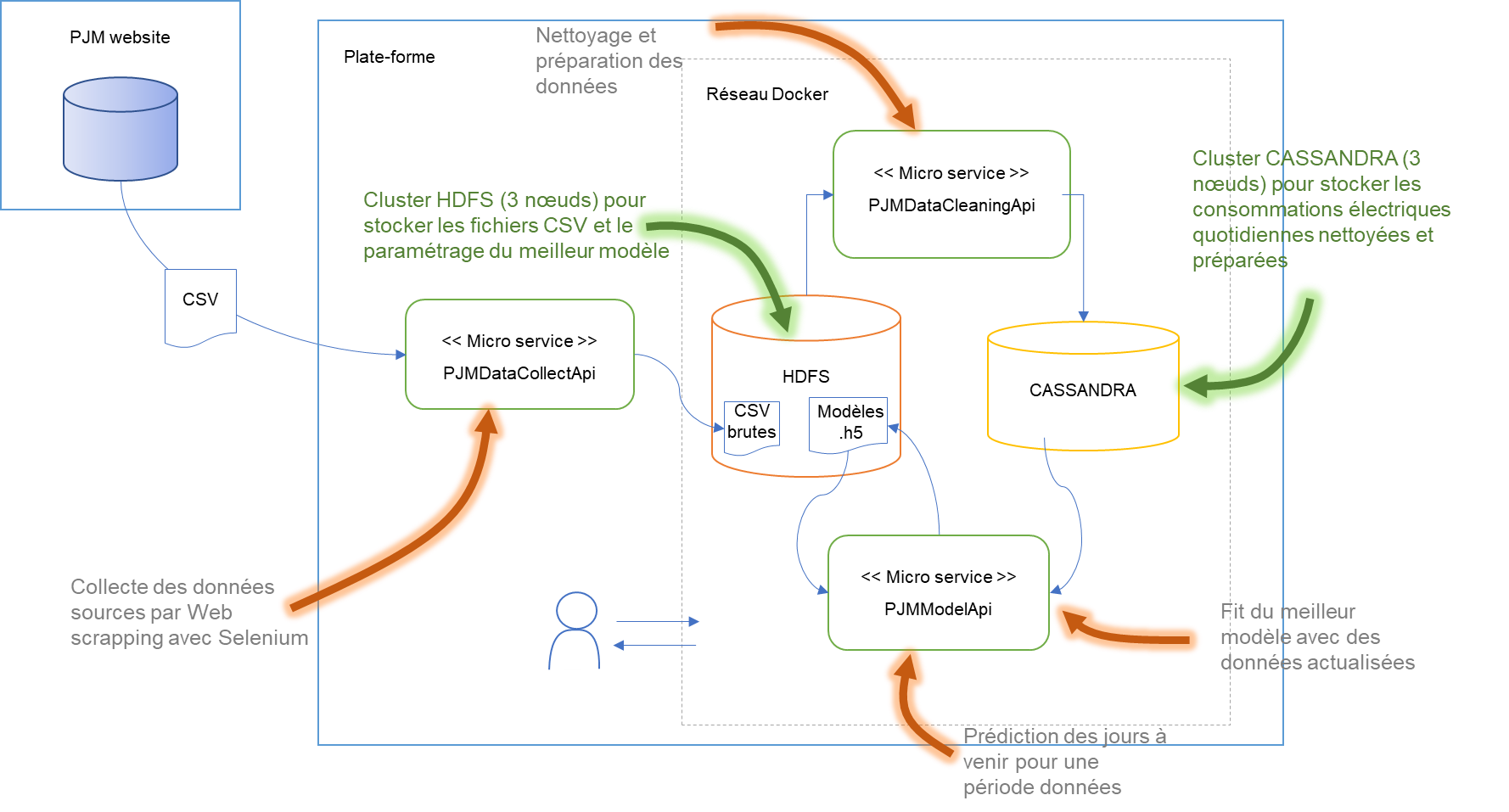
# Résultats

Après optimisation de tous les modèles présentés dans cette étude (XGBoost, KNN, ARIMA, LSTM), le modèle SARIMA reste celui qui présente les meilleures performances avec un RMSE **80900**.

La figure ci-dessous représente les valeurs du RMSE pour chaque modèle



# Déploiement de la solution



**Composants techniques** :

* Un cluster HADOOP (1 namenode et 2 datanode) pour stocker les données OPENDATA récupérées sur le site Web du RTO PJM,
* Un cluster CASSANDRA pour stocker les données et les mettre à disposition des services d’évaluation et de prédiction des modèles.

**Composants logiciels** :

* Microservice « **PJMDataCollectApi** » :

* + Récupère les données de consommation électrique par heure sur le site de PJM par Web scrapping en manipulant le navigateur Chrome avec la bibliothèque Selenium et les stocke sur le cluster HDFS.
  + URL d’accès : <http://localhost:5000>
  + Technologies :Flask, Python , Selenium, HDFSCli
* Microservice « **PJMDataCleaningApi** » :

* + Récupère les fichiers CSV à traiter sur le cluster HDFS pour en traiter les données, les nettoyer et les préparer pour être utilisables par le modèle de prédiction et les enregistrer dans CASSANDRA.
  + URL d’accès : <http://localhost:5001/start> | /stop
  + Technologies :Flask, Python, Pandas, Numpy, Cassandra-driver, HDFSCli
* Microservice « **PJMModelApi** » :
  + Entraine le modèle de prédiction à partir de données réactualisées
  + URL d’accès :
    - <http://localhost:5002/evaluate> (entrainement du modèle)
    - [http://localhost:5002/predict/<int](http://localhost:5002/predict/%3cint): period> (prediction pour les « period » prochains jours)
  + Technologies :Flask, Python, Pandas, Numpy Statsmodel, cassandra-driver

Les microservices (sauf PJMDataCollectApi ) sont interconnectés sur un même réseau Docker.

# Annexe – Procédure d’installation de l’environnement de développement

## Description de l’environnement

### 

|  |  |
| --- | --- |
| **OS :** | VM UBUNTU 20.04 |
| **Stockage de fichiers « DataLake »** | Hadoop HDFS |
| **Base de données** | CASSANDRA |
| **Framework Data analytics** | Python / Pandas / Numpy / Tensorflow / Keras |

## Installation Anaconda en local sur la VM

### Mise à jour de la gestion locale des packages

sudo apt-get update

sudo apt-get install curl

### Téléchargement de la dernière version d'Anaconda

Vérification de la version de Python

python --version

==> 3.7.9

Téléchargement de la version Anaconda correspondant à la version de Python

cd ~/Downloads

wget <https://repo.anaconda.com/archive/Anaconda3-2020.02-Linux-x86_64.sh>

### Installation d'Anaconda

sudo bash ~/Downloads/Anaconda3-2020.02-Linux-x86\_64.sh

>> Saisir '**yes**' pour accepter les conditions

>> Saisir '**/home/formation/anaconda3**' comme chemin d'installation

conda update --all --yes

#Création de l'environnement dev\_env (ne pas travailler sur l'environnement par défaut "base")

conda create --name dev\_env python=3.7.9

# Activation de l'environnement dev\_env

conda activate dev\_env

# Lancement de la mise à jour (peut-être pas nécessaire)

conda update --all --yes

# Installation des bibliothèques manquantes (PySpark, Tensorflow, Keras)

conda install pyspark

conda install -c conda-forge tensorflow

conda install keras

## Installation des container docker HADOOP, Jupyter/SPARK et CASSANDRA

### Copie des sources Docker

* Copier le package docker-cluster.zip dans le dossier "~/Desktop" de la VM et le décompresser

*(Le fichier* *docker-cluster.zip est présent aussi dans la note « Installation environnement »  dans Teams)*

Fichier à récupérer sur GIT (<https://github.com/LARROUY-Bruno/GitM2iDS2Gr1_Project> dans le dossier 03-sources)

* Mettre à jour le fichier ~/Desktop/ docker-cluster/docker-compose.yml si nécessaire

Fichier à récupérer sur GIT (<https://github.com/LARROUY-Bruno/GitM2iDS2Gr1_Project/> dans le dossier 03-sources)

### Création des scripts de démarrage et d'arrêt des container

nano ~/Desktop/start\_hadoop.sh

*#!/bin/bash*

*cd ~/Desktop/git/docker-hadoop-master*

*docker-compose up -d*

nano ~/Desktop/stop\_hadoop.sh

*#!/bin/bash*

*cd ~/Desktop/git/docker-hadoop-master*

*docker-compose down*

### Modification des droits des scripts de démarrage et d'arrêt des containers pour autoriser l'exécution

sudo chmod 774 start\_hadoop.sh

sudo chmod 774 stop\_hadoop.sh

### Démarrage des container docker

~/start\_hadoop.sh

### Arrêt des container docker

~/stop\_hadoop.sh

Récupération du token pour Jupyter du container jupyter/pyspark-notebook

docker logs spark\_jupyter

## Résumé des containers

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nom du container** | **Type** | **IP** | **Ports ouverts (exposés en local)** | **Port localhost** |
| resourcemanager | Hadoop resource manager (YARN) | 172.23.0.3 |  |  |
| nodemanager | Hadoop node manager | 172.23.0.11 |  |  |
| hystoryserver | Hadoop Hystory Server | 172.23.0.12 |  |  |
| namenode | Hadoop Namenode | 172.23.0.20 | 9870 : interface Web de gestion  9000 : Hadoop IPC port | 9870  9000 |
| datanode | Hadoop Datanode | 172.23.0.21 | 9864 : interface web (9866) | 9864 |
| datanode2 | Hadoop Datanode | 172.23.0.22 | 9864 : interface web (9866) | 9865 |
| datanode3 | Hadoop Datanode | 172.23.0.23 | 9864 : interface web (9866) | 9866 |
| cassandra | Base de données CASSANDRA | 172.23.0.31 | 9042 : service cassandra | 9042 |
| Cassandra2 | Base de données CASSANDRA | 172.23.0.32 | 9042 : service cassandra | 9043 |
| Cassandra3 | Base de données CASSANDRA | 172.23.0.33 | 9042 : service cassandra | 9044 |
| Spark\_jupyter | SPARK + Jupyter notebook | 172.23.0.40 | 7077 : service SPARK master  8888 : jupyter notebook | 7077  9888 |

Réseau

Driver : Bridge

Subnet : 172.23.0.0/16

Gateway : 172.23.0.1

## Microservices

* Copier le package app.zip dans le dossier "~/Desktop" de la VM et le décompresser

Fichiers à récupérer sur GIT (<https://github.com/LARROUYBruno/GitM2iDS2Gr1_Project/tree/dev/03_sources> )

* Pour démarrer un microservice, aller dans le dossier "~/Desktop/apps/*microservice*/", et exécuter la commande « sudo ./start.sh »

****