Rapport du TER : Traitement des flux de données horodatées



Master 2 Informatique SID

SEMESTRE 2

|  |  |  |
| --- | --- | --- |
| **Membres du Projets** | Khadija LAMNAOUR | khadija.lamnaour@etu.univ-amu.fr |
|  | Hadja Maimouna BALDE | Hadjamaimouna.BALDE@etu.univ-amu.fr |
|  |  |  |
|  |  |  |
| **Encadrant** | Omar BOUCELMA | Omar.boucelma@univ-amu.fr |

**Année 2021 - 2022**

**Résumé**

 L’analyse des données et la modélisation de l'évolution dans le temps sont plus intéressantes lorsqu’il s’agit du traitement des données horodatées ou Série Temporelle. Elles sont générées de manière continue (**en flux**) en très gros volumes et sont souvent de **qualité** variable. De plus, elles sont reçues avec un intervalle court afin d'être précis dans l’analyse. Nous avons nettoyé les données, effectué des pré traitements adaptés aux caractéristiques de chaque modèle, créé plusieurs modèles pour pouvoir tirer une conclusion déterminant laquelle des modèles à la meilleure prédiction et nous avons fini par traiter ces données à partir de deux systèmes qui sont kafka et influxDB. Pour terminer nous avons fait une architecture de chaque système et créé une application pour afficher le résultat des données. Une comparaison de ces deux systèmes a permis de préciser lequel d’entre eux est le meilleur. Nous précisons que ce projet a pour objectif de proposer une **architecture** et un **pipeline** « Data Science » pour le développement d’applications qui manipulent des données.

**Abstract**

Being scientists and future engineers in data science. The analysis of data and the modelling of the evolution over time is more interesting when it comes to the processing of time-stamped or Time Series data which are generated continuously (in streams) in very large volumes. They are often of variable quality and are received with a short interval, in order to be precise in the analysis.

To this end, under the supervision of Mr. Omar BOUCELMA, the objective of this project is to propose an architecture and a "Data Science" pipeline for the development of applications that handle time-stamped data.

As for the analysis of data types (dynamic data), several systems are in place. But for this project we will use **Apache Kafka** and **influxDB** which represent two approaches to manage time series data but do not necessarily integrate all the adequate analysis tools (anomaly detection, predictive analysis in real time, etc.).

Finally, from these two systems we will also draw a conclusion as to which one is better and why from the analyses we have carried out.

**Tableau des sigles**

Le tableau ci-dessous explique la signification de chaque sigle

Tableau 1 : Tableau des sigles

|  |  |
| --- | --- |
| **Abréviation** | **Signification** |
| SGBD | Système de gestion de base de données |
| LSTM | Long Short-Term Memory |
| AR | Autoregressive |
| STL | Seasonal and Trend decomposition using Loess |
| VAR | vectorial autorégressiv |
| ARIMA | Autoregressive-Integrated-Moving Average |

**Présentation**

Dans le cadre du Travail Encadré de Recherche (TER), dont l’objectif est de permettre aux étudiants de mettre en œuvre les connaissances acquises durant cette dernière année de la formation, nous avons réalisé un projet sur **le traitement des flux de données horodatées**.

L'horodatage est un mécanisme qui consiste à associer une date et une heure à un événement, une information ou une donnée informatique. Il a généralement pour but d'enregistrer l'instant auquel une opération a été effectuée (Ipside).

Ce projet dont l’objectif est de proposer une **architecture** et un **pipeline** « Data Science » pour le développement d’applications qui manipulent des données horodatées est encadré par Monsieur Omar BOUCELMA. Pour l’analyse des types de données (données dynamiques), plusieurs systèmes sont mis en place. Cependant, nous avons utilisé **Apache** **Kafka** et **influxDB** qui représentent deux approches servant à gérer des *séries de données temporelles* mais n’intègrent pas forcément tous les outils d’analyse adéquats (détection d’anomalies, analyse prédictive en temps réel, etc).

Pour mener à bien ce travail, nous allons d’abord effectuer l’état de l’art des projets type Apache et des systèmes de bases de données traitant des données horodatées et l'état de l’art des méthodes d’apprentissage (statistiques / machine Learning) adaptées aux séries temporelles en flux c’est-à-dire des méthodes de streaming (traitement en temps réel) qui nous permettrons de connaitre les différents types de traitement possible sur ces données. Ensuite nous proposerons des solutions pour traiter les données, puis une architecture avec Kafka et influxDB. Nous réaliserons ensuite des expérimentations sur les résultats obtenus. Et pour terminer nous développerons une application avec un jeu de données réel selon l’architecture proposée pour l’affichage des données.

1. **État de l’art**
   1. **État de l’art sur les systèmes (architectures data / logicielle) de traitement de données horodatées**

**2.1.1 Définition des données horodatées**

Le traitement de flux de données horodatées est conçu :

* Pour le traitement instantané des données (c à d des données qui sont encore en mouvement) et leurs analyses en temps réel.
* Pour enregistrer aussi l’instant auquel une opération a été effectuée.

L’objectif principal est de fournir des informations actualisées à la milliseconde sur les événements qui se produisent dans un système et de nous aider à réagir aux événements critiques dès qu’ils se produisent. De ce fait, ce traitement est très avantageux pour les entreprises, car ça leurs permettent d’avoir des informations en temps réel sur la santé d’un système.

Il existe différents domaines dans lesquels ces données sont utilisés telle que :

* La détection des fraudes
* Intelligence artificielle etc.

**2.1.2 Les solutions existantes pour traiter les données horodatées**

En général il y a deux types de systèmes qui permettent de traiter les données horodatées :

* Un autour de Kafka et,
* Un autre autour des bases de données time séries.

**2.1.3 Comparaisons entre les alternatives de Kafka**

Il existe plusieurs alternatives de Kafka qui permettent de recevoir les données en temps réel. Le tableau ci-dessous nous expliques et différencies les plus populaires.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nom** | **Apache Kafka** | **Apache Flink** | **Apache Étincelle** | **Tempête Apache** |
| Description | Apache Kafka est un logiciel sur site et basé sur le cloud conçu pour aider les utilisateurs d'Apache Kafka à gérer et à gérer la configuration des clusters afin de faciliter les processus de création et de développement d'applications.  Icône de validation par la communauté | Apache Flink est un framework et un moteur de traitement distribué pour les calculs avec état sur des flux de données illimités et limités. Flink a été conçu pour fonctionner dans tous les environnements de cluster courants et effectuer des calculs à une vitesse en mémoire et à n'importe quelle échelle. | Apache Spark est un moteur d'analyse unifié pour le traitement de données à grande échelle. Il se distingue par ses hautes performances pour les données par lots et en continu en utilisant un planificateur DAG, un optimiseur de requêtes et un moteur d'exécution physique | Apache Storm est un système de calcul en temps réel distribué gratuit et open-source. Apache Storm facilite le traitement fiable de flux de données illimités, faisant pour le traitement en temps réel ce que Hadoop a fait pour le traitement par lots. |
| Déploiement | Basé sur cloud | Basé sur cloud et sur site | Basé sur cloud et sur site |  |
| Première version | Janvier 2011 | 6 janvier 2015 (28 octobre 2022) |  |  |
| Version actuelle | 3.3.1 (29 septembre 2022) | 1.16.0 |  |  |

Les bases de données conçues pour traiter les séries temporaires sont multiples mais parmi les plus populaires on a KDB+ et influxDb.

**2.1.4 Comparaison entre influxDB, KDB+ et timescaleDB**

|  |  |  |  |
| --- | --- | --- | --- |
| **Informations éditoriales fournies par DB-Engines** | | | |
| Nom | InfluxDB | KDB+ | TimescaleDB |
| Description | SGBD pour stocker des séries chronologiques, des événements et des métriques. | SGBD de séries chronologiques hautes performances. | Un SGBD de séries chronologiques optimisé pour l’ingestion rapide et les requêtes complexes, basé sur PostgresQL. |
| Modèle de base de données principal | SGBD de séries temporelles | SGBD de séries temporelles | SGBD de séries temporelles |
| Modèle de base de données secondaire | SGBD spatial | SGBD relationnel | SGBD relationnel |
| Classement des moteurs DB | Score 30.25  Rang   #28 Globalement             #1 SGBD de séries temporelles | Score 7.23  Rang   #59 Globalement             #2 SGBD de séries temporelles | Score 4.76  Rang   #82 Globalement             #5 SGBD de séries temporelles |
| Première version | 2013 | 2000 | 2017 |
| Version actuelle | 2.5.1, novembre 2022 | 3.6, mai 2018 | 2.6.0, février 2022 |
| Licence | Open source | Commercial | Open source |
| Basé sur le cloud uniquement | Non | Non | Non |
| Caractéristique spécifiques | Influx Data est le créateur de influxDB, spécialement conçu pour gérer les volumes massifs de données horodatées produites par les appareils et capteurs IoT, Les applications en temps réel pour l’IoT, l’analyse et les services cloud natifs. | Base de données en colonnes intégrée et système de programmation pour le streaming, les données en temps réel et historiques.  Système d’exploitation et matériel standard. | Toutes les données sont des données chronologiques.  Elle est la seule base de données relationnelle conçue avec les performances, l'évolutivité et la fiabilité nécessaires pour stocker ces flux de données incessants. |
| Avantage concurrentiels | **Temps de valeur**  Disponible dans tous les langages et Framework populaires, permettant aux développeurs d’être opérationnels en quelques minutes, et non en quelques jours ou semaines. Avec des démarrages rapides et une documentation facile à utiliser, les développeurs peuvent créer des applications ouvertes, extensibles et faciles à déployer.  **Action réelle en temps réel**  InfluxDB offre une visibilité avec des analyses en temps réel afin que les développeurs puissent agir rapidement sur leurs données. | Offre une évolution transparente.  Fonctionne sur des plateformes de serveur standard de l'industrie.  S’adapte facilement à l’API pour la connectivité aux principaux systèmes et modules externes.  Nécessite moins de développeurs | Conçu pour mettre à l'échelle et gérer les charges de travail de données de séries chronologiques et est intentionnellement conçu comme une extension PostgreSQL |

Pour ce projet on utilisera Kafka pour recevoir les données en temps réel ou la base de données influxDB qui sont les plus populaire pour traiter ces série temporelles (données horodatée).

**2.1. 5 Architecture Kafka**

Kafka possède un ensemble de composant assurant son fonctionnement en tant que système de messagerie distribué :

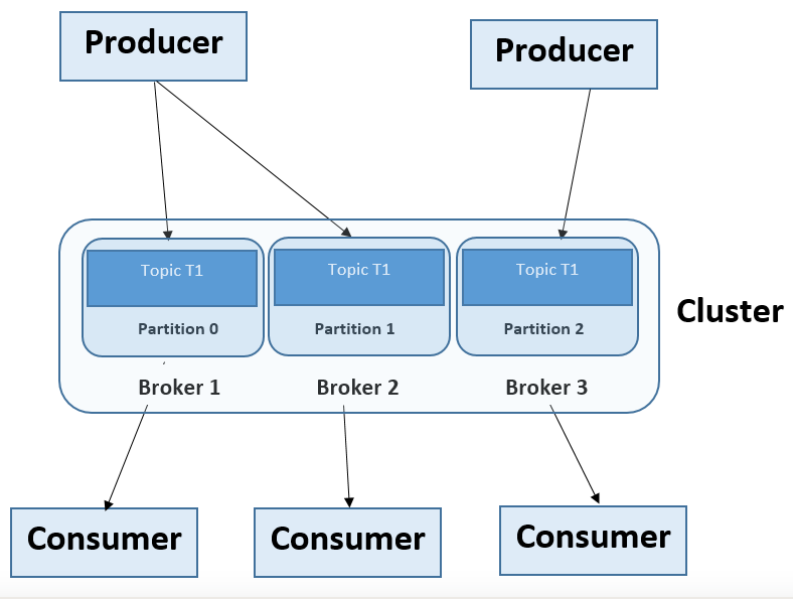


Figure 1 illustration du fonctionnement Kafka

En claire :

* Topic : les données dans Kafka sont stockées ou enregistrées ici.
* Producteur : c’est le système qui va produire les données dans Kafka et publié les messages à un topic
* Broker : est un ensemble de serveur ou nœud formant un cluster Kafka, permettant de stocker et gérer les messages publiés
* Consommateurs : peuvent s’abonner à un ou plusieurs topics et consomme les messages publiés à partir du cluster

Dans l’exemple de la figure 1, le topic T1 contenant N message est partitionné à travers 3 brokers du cluster Kafka. Les consommateurs consomment toujours les messages à partir d’une partition spécifique, de façon séquentielle. Cela joue un rôle non négligeable dans les bonnes performances des traitements effectués par Kafka. Avant toute consommation d’un message, chaque consommateur lance une requête asynchrone de type pull au broker afin que celui-ci lui alloue un tampon d’octets prêts à être consommé.

**2.1.6 Architecture influxDB**

InfluxDB est conçues pour stocker et analyser des données de capteurs ou des journaux horodatés sur une période donnée. Fournit un agent de serveur piloté par plug-in appelé Telegraf qui collecte et rapporte les métriques de toutes les sources prises en charge et alimente InfluxDB avec ces données ou vice versa.

Telegraf peut être démarré en tant que service sysvinit ou systemd sous linux, ou l'exécuter en tant que commande de terminal.

À partir de la figure ci-dessous on peut voir que telegraf a des plug-ins d’entrée et de sortie, qui doivent être configurés à l’avance avant de le démarrer. Par exemple, il dispose d’un plug-in d’entrée consommateur apache kafka qui permet de lire les messages d’un sujet kafka et d’un plug-in de sortie qui permet d'écrire toutes les données d’entrée dans influxDB.

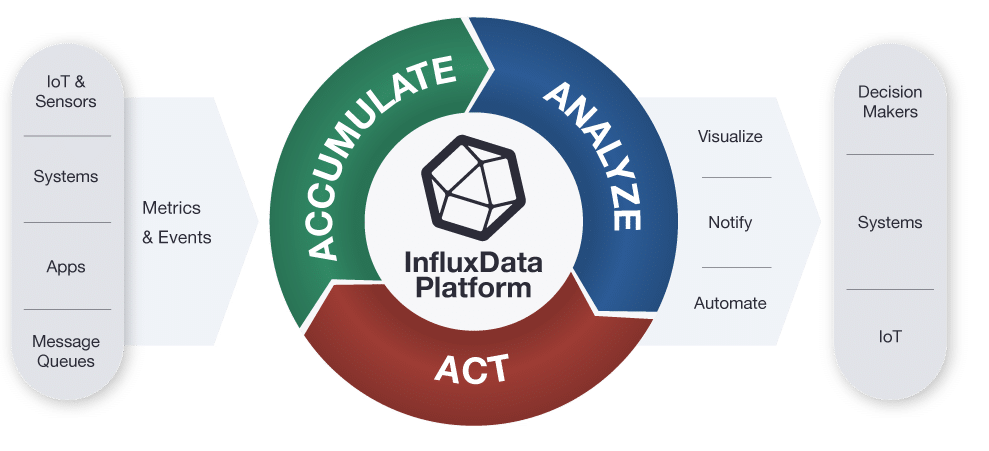


Figure 2 : illustration du fonctionnement InfluDB

**2.2 Etat de l’art sur les méthodes d’apprentissage (statistiques / machine learning) adaptées aux séries temporelles en flux**

Les principaux axes d’études d'apprentissage autour des séries temporelles qui ont été proposés dans la littérature sont les suivants :

**2.2.1 Prétraitement des données dynamiques**

**2.2.1.1 Données manquantes**

Pour traiter les données manquantes contrairement aux données statistiques on peut utiliser la **moyenne glissante**, la **moyenne mobile ou la moyenne de fenêtre**, qui consiste à prendre la moyenne des valeurs relatives à une fenêtre prédéfinie. Une autre manière de les traiter consiste à faire **l’interpolation** qui est aussi la méthode la plus utilisée pour déterminer les parties manquantes des données de séries temporelles. Cette dernière utilise la valeur présente avant et après le point manquant pour calculer la donnée manquante. Dans certains cas où de nombreuses données sont manquantes, on utilisé les **modèles autorégressifs (AR)**.

On peut aussi utiliser d’autres méthodes à base d’heuristiques et parmi ces méthodes on peut citer :

* Moyenne : Cette méthode consiste à remplacer les données manquantes d’une série temporelle par la moyenne des valeurs observées sur la séquence, par une moyenne sur les valeurs qui précédent la valeur manquante ou par une moyenne des valeurs observées dans une certaine fenêtre autour d’une valeur manquante.
* Dernière observation : L’idée est de remplacer une valeur manquante par la dernière valeur observée sur la séquence. Dans le cas où les valeurs manquantes sont peu nombreuses, cette méthode peut donner de très bons résultats
* Knn-substitution : utilisée dans la complétion de données manquantes dans le cas de séries temporelles multivariées, cette méthode consiste à rechercher la série la plus similaire (au sens des plus proches voisins) à une série donnée et de remplacer les valeurs manquantes de cette dernière par celles du plus proche voisin.
  + - 1. **Caractéristiques principales des séries temporaires**

Quant aux données de séries temporelles, elles ont obligatoirement des composantes de séries temporelles. L’extraction de ces caractéristiques peut s’effectuer via la décomposition STL (Seasonal and Trend decomposition using Loess) et certaines d’entre elles sont définies ci-dessous :

* **Saisonnalité :** La saisonnalité fait référence à une propriété des séries temporelles qui affiche des modèles périodiques se répétant à une fréquence constante.
* **Tendance :** Les données de séries chronologiques présentent une tendance lorsque leur valeur varie dans le temps. Une valeur croissante indique une tendance positive et une valeur décroissante, une tendance négative.
* **Reste :** Après avoir extrait la tendance et la saisonnalité des données, ce qui reste est ce que nous appelons le reste (erreur) ou le résidu. Cela permet de détecter les anomalies dans les séries chronologiques.
* **Cycle :** Les données de séries temporelles sont dites cycliques lorsqu'il existe des tendances sans répétitions fixes ou saisonnalité.
* **Stationnarité :** Les données de séries temporelles sont stationnaires lorsque leurs caractéristiques statistiques ne changent pas dans le temps, c'est-à-dire une moyenne et un écart-type constants.

Une fois extraites, ces composantes constituent la base pour l'analyse de l’évolution de la série temporelle afin de comprendre son comportement et de pouvoir choisir et adapter un modèle de série temporelle approprié.

Les données de séries temporelles peuvent avoir d'autres attributs que les caractéristiques temporelles. Si ces attributs sont temporels, la série temporelle résultante sera multivariée et si elle est statique, elle sera univariée avec des caractéristiques statiques. Les caractéristiques non temporelles peuvent utiliser des méthodes issues des techniques statiques de manière à ne pas nuire à l'intégrité des données.

**2.3 Approches algorithmiques pour la prévision des séries temporelles**

**2.3.1 Préparation des données dynamiques**

 Il y a deux choses à établir avant de concevoir un modèle de prévision :

* Les informations disponibles au moment où la prévision est faite (caractéristiques) ;
* La période pendant laquelle l’utilisateur a besoin de valeurs prévisionnelles (objectif).

 L'origine de la prévision est le moment auquel une prévision est effectuée. En pratique, l'origine de la prévision est le dernier moment pour lequel des données d'entraînement sont disponibles pour la période à prévoir. Tout ce qui va jusqu'à l'origine peut être utilisé pour créer des caractéristiques.  L'horizon de prévision, qui décrit l’objectif, est la période pour laquelle une prévision va être effectuée.

**2.3.2 Modèles de prédiction de séries temporelles multivariées**

Parmi les modèles existants, le modèle VAR (vectorial autoregressiv) est l’un des plus répandus et les plus utilisés. Il s’agit d’une extension naturelle au modèle autorégressif univarié AR. Il part du principe que la valeur d’une série temporelle dépend des valeurs précédentes de la série pendant une fenêtre temporelle donnée, mais également des valeurs des autres séries considérées pendant ce même intervalle de temps. Ce modèle du fait de sa simplicité a été extrêmement utilisé et il s’est montré particulièrement efficace dans certaines tâches prédictives. Cependant les dépendances linéaires entre séries restent une hypothèse forte et limitant le pouvoir d’expression de ces modèles ; de plus, la forte complexité en nombre de paramètres

**Autres modèles**

De nombreux autres modèles ont été proposés pour la prédiction de séries temporelles et il ne serait pas possible de faire ici un état de l’art exhaustif de ceux-ci. Plus précisément, un grand nombre de modèles existe dans des cadres restreints : on peut par exemple citer le modèle de :

* Croston particulièrement adapté dans le cas où les séries temporelles valent souvent zéro.
* ARIMA (Autoregressive-Integrated-Moving Average) : Il utilise la combinaison de moyennes autorégressives et mobiles pour prédire les valeurs futures, c’est un modèle univarié.
* Modèles de régression dynamique : cet algorithme prend également en compte d'autres informations diverses telles que les jours fériés, les modifications de la législation, etc.
* Prophet : qui a été publié par l'équipe Core Data Science de Facebook, est une bibliothèque open source développée par Facebook et conçue pour la prévision automatique de données de séries chronologiques univariées.

Dans ce projet nous nous limitons aux modèles les plus familièrement utilisés dans la communauté de l’apprentissage automatique pour les prévisions de séries chronologiques comme VAR et ARIMA.

**3. Solution proposé**

Pour ce projet, nous utiliserons Kafka pour recevoir les données en temps réel et la base de données influxDB qui sont les plus populaires pour traiter ces séries temporelles (données horodatées). Pour la partie Kafka nous nous servirons des modèles comme ARIMA, VAR qui sont les plus populaires pour la prévision des séries temporelle, dans le but de détecter les anomalies que contiennent les données qui arrivent dans le serveur Kafka, et un autre qui n’est pas très populaire mais qui possède les mêmes caractéristiques (Random Forest). Pour terminer nous utiliserons un web serveur django comme application pour afficher les données contenant des anomalies.

**3.1 Architecture de traitement avec apache kafka**

.

**Serveur web**

**Django**

**Producer**

Consomme les données

Produit les données

Affiche les données sur une page web



**ConsumerTopic**

**Anomalie**

**Page web**

**Broker**

Produit un message s’il y’a anomalie

Lu les données

**Consumer**

La figure ci-dessus met en exergue la manière dont nous avons effectué le traitement des données avec Kafka, tout d’abord, on remarque que le producer produit, procure les données dans le topic Incomming-data de ce dernier et le consommateur lu les données produites dans le topic. Ensuite quand le consommateur termine sa lecture, il produit aussi une nouvelle information vers un autre topic appelé Anomalie, donc le consommateur se comporte aussi comme un produceur. Par la suite nous avons un serveur web django, qui lui aussi va consommer les données qui ont été produites dans le topic anomalie et afficher les anomalies sur une page web.

**4. Expérimentations**

**4.1 Présentation des jeux de données**

Dans ce projet on a travaillé avec ce jeux de donnée

**4.1.1 Jeux de données météorologique**

**Titre des données :** Climat Météo Surface de Brésil - Horaire

**Description :** Ce jeu de données couvre des données climatiques par heure de 122 stations météorologiques inmet (Institut national brésilien de météorologie) du Brésil entre 2000 et 2021, à savoir qu’on a utilisé les données brutes de la région sud collectée sur plusieurs années et toutes les stations météorologiques n'ont pas commencé à fonctionner depuis 2000. Ce jeu de données a été publié sur le site de Kaggle.

**Thèmes :** Données météorologiques.

**Lieu :** Brésil

**Langue :** Portugais

**Propriétaire :** PROPPG/PPG em Informática - Doutorado e Mestrado.

**Fréquence de mise à jour :** Trimestriel

**Variables étudiées :** precip\_totl\_tp, prsn\_atmos\_max\_tp\_prcedt, prsn\_atmos\_min\_prmier\_tmp, rad\_gobal, tmp\_air, tmp\_pt\_rose, humid\_rlative\_air, vent\_vitess\_tmp, station.

**Villes étudiées :** GENERAL CARNEIRO, URUSSANGA, VACARIA, CAMPINA DA LAGOA, CURITIBANOS

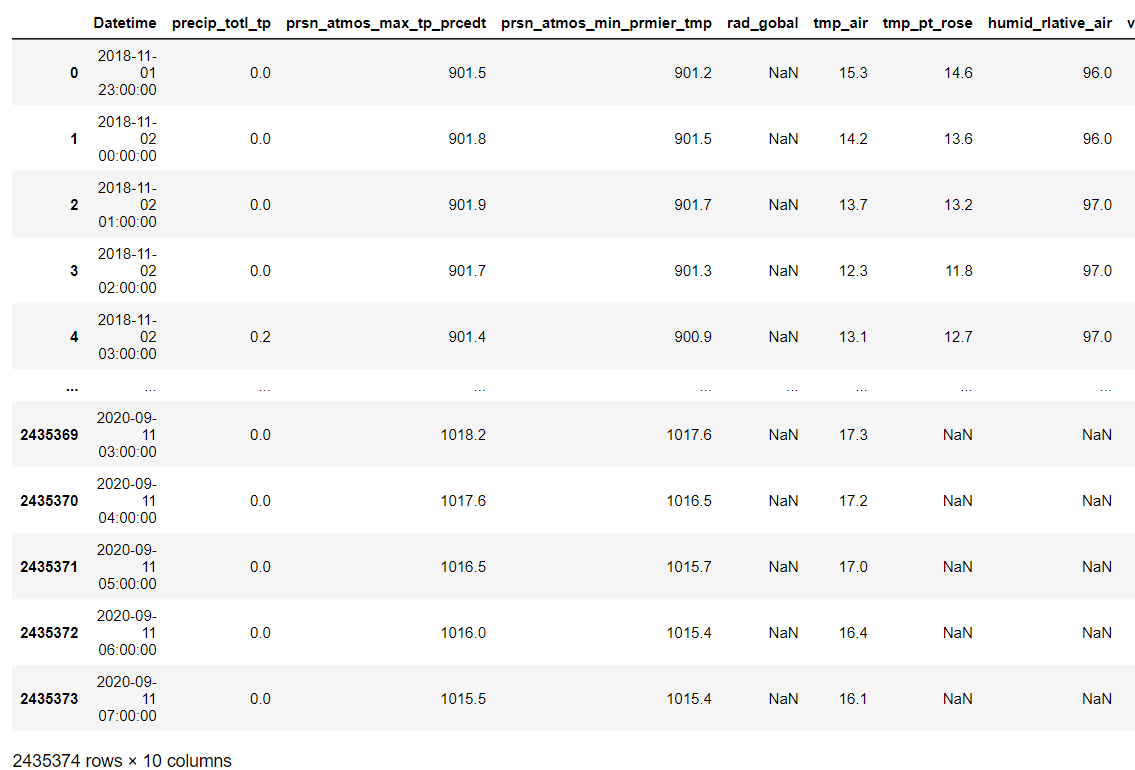


Tableau 2 : Visualisation du data frame de 2018 à 2020 avec la présence des données manquantes

Les colonnes de ce DataFrame ont été renommé en français car de base ils étaient en espagnole et comme le montre la Figure 2, on peut remarquer la présence de données manquantes comportant la valeur **NaN**.

**4.2 Analyse Des Données**

**4.2.1 Corrélation entre les capteurs**

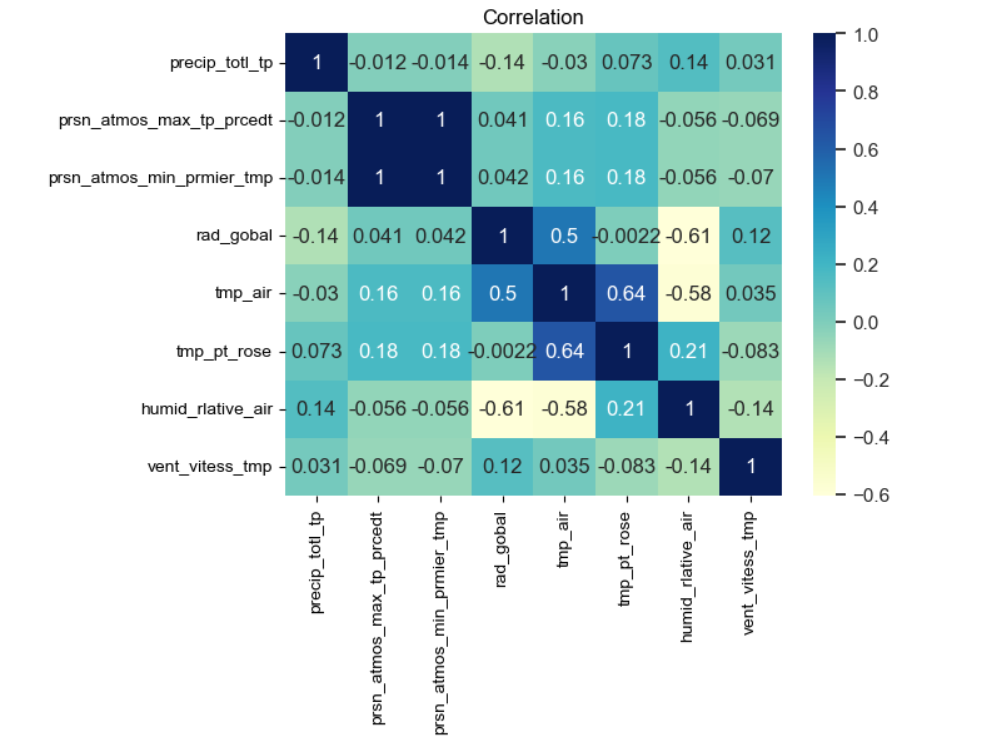


Tableau 3 : Figure 1 : Corrélation entre les capteurs du jeu de données météorologiques

La figure ci-dessus justifie qu’il y’a une forte relation entre les capteurs eux même et une faible relation entre les différents capteurs.

**4.2.2 Description des données**

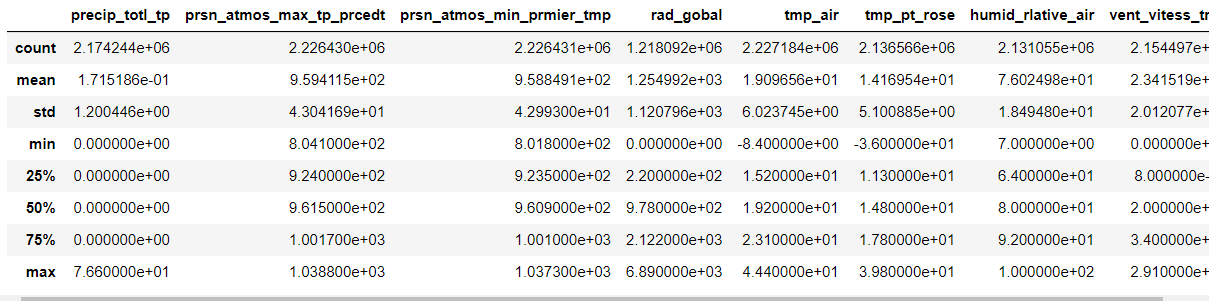


Tableau 4 : Tableau de la description des données

À partir de la figure 4 on peut remarquer que les données ne contiennent pas de valeur aberrante car une température peut être négative ou nulle tout dépend de l’état du climat.

**4.2.3 Traitement des valeurs manquantes**

Une image contenant texte, ordinateur, capture d’écran

Description générée automatiquement

Tableau 5 : Visualisation du dataframe sans la présence des données manquantes

À partir de la figure ci-dessus on peut remarquer que les données manquantes ont été traité car on a plus la présence des valeurs NAN.

**4.3 Models appliqués**

**4.3.1 Model ARIMA**

Vu que c’est un modèle univarié on a choisi la température de l'aire comme il s’agit des données météorologiques

Une image contenant table

Description générée automatiquement

Figure 1 : Visualisation du dataframe pour le model ARIMA

**4.3.1.1 Vérification des caractéristiques du model ARIMA**

**4.3.1.1.1 Vérification de la stationnarité**

Une image contenant texte

Description générée automatiquement

Figure 2 : stationnarité du model

À partir de la figure 2 on peut constater que le model n’est pas stationnaires car le p-value est supérieur à 0,05 or que le p-valeur doit être inferieur ou égal à 0,05 bien que ADF est proche des valeurs critiques.

**4.3.1.1.2 Rendre la série chronologique stationnaire**

Pour rendre la série stationnaire on doit lui différencier de 1.

Une image contenant texte

Description générée automatiquement

Figure 3 : Stationnarité du model

Pour qu’une série temporelle soit stationnaire son p-value doit être inferieur ou égal à 0,05, donc à partir de la figure ci-dessus on remarque maintenant que la série chronologique est stationnaire car le p-value est inférieur à 0,05.

**4.3.1.1.3 Comment choisir les paramètres de ARIMA**

ARIMA est définit par 3 Paramètres, à savoir p (nombre d’auto-régressions), d (nombre de différenciations) et q (nombre de termes de moyenne mobile). Pour cela on a utilisé la fonction auto\_arima qui nous a permis d’avoir le meilleur paramètre.

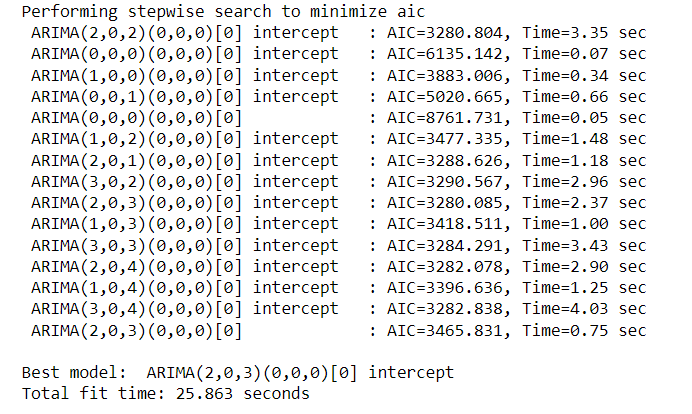


Figure 4 : Paramètres d'ARIMA

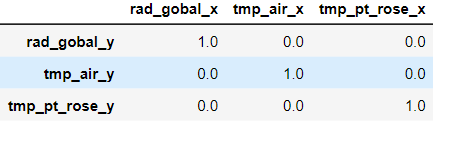
À partir de la figure ci-dessus on voit que le model à (2,0,3) comme meilleur paramètre

**4.3.2 Model VAR**

Ce modèle est un modèle multivarié et à partir de la matrice de corrélation (voir figure) ont peu voir que les colonnes qui sont proches de la colonne cibles tmp\_aire est tmp\_pt\_rose et rad\_global,donc donc on utilisera ces colonnes pour l’entrainement de ce model.

**Vérification des caractéristiques du model**

**Tester la causabilité**



**Tester la cointégration**

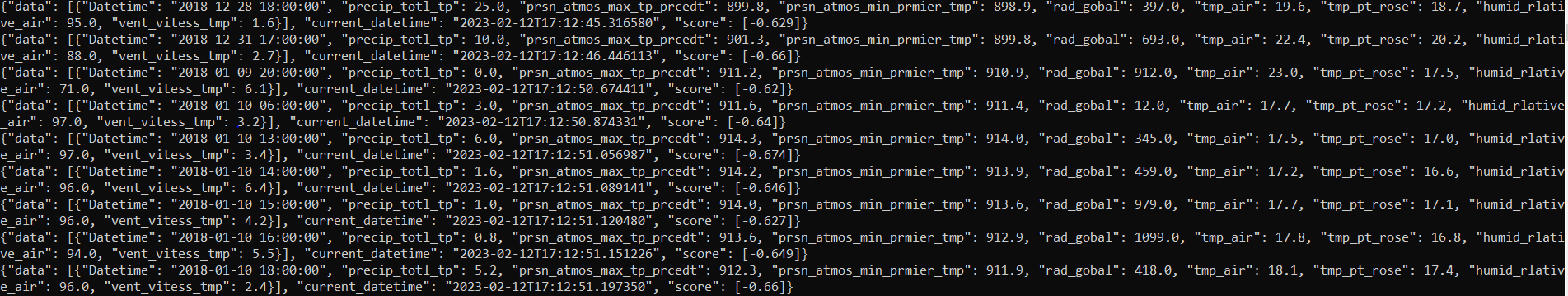
**Vérifier la stationnarité**

Pour chaque colonne vérifier la stationnarité comme effectué sur le model ARIMA et si ce n’est pas stationnaire lui rendre stationnaire. Pour ce model aussi.

Affichage des résultats

Partie kafka

Anomalie



Incomming data

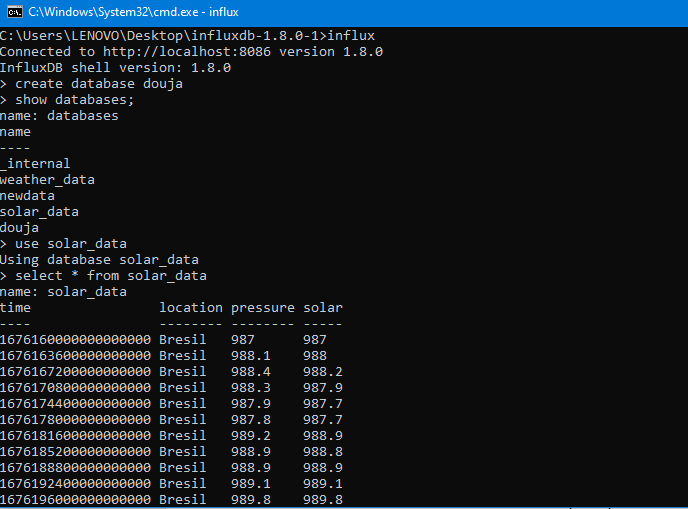
Une image contenant texte

Description générée automatiquement

**Obtention et stockage de données de séries chronologiques avec Python et InfluxDB.**

Tout d'abord, nous devons demander nos données. Nous utiliserons la bibliothèque de requêtes pour renvoyer des données météorologiques horaires à partir d'une longitude et d'une latitude spécifiées .

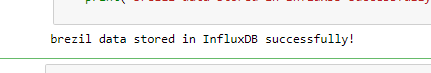
Ensuite, le lancement de influxdb et la création de base de donnée sous command line (create database solar\_data){on a utiliser la version 1.8 car la version 2.\* a un problème d'authentification(unauthorized)}



Et après initialisation d'un client influxdb pour l’importation du data de puis le fichier csv.on va utiliser seulement les 2000 premiers éléments puisque notre base de donnes est très grande , donc on va remplir un tableaux aves les donnes qui nous intéresse a méthode d'écriture InfluxDB nécessite que l'horodatage soit au format d'objet datetime. Nous utiliserons ensuite cette méthode pour écrire nos données dans InfluxDB. Nous supprimons également les colonnes que nous ne voulons pas écrire dans InfluxDB.

instanciez la bibliothèque client InfluxDB Python et écrivez le DataFrame dans InfluxDB. Nous spécifions un nom de mesure. Une mesure contient des données dans un compartiment. Vous pouvez le considérer comme le deuxième niveau hiérarchique le plus élevé d'organisation des données dans InfluxDB après un compartiment.

et finalment l’importation du data vers la base de données "solar\_data" de influxdb .



Maintenant que nous avons écrit nos données dans InfluxDB, nous pouvons utiliser l'interface utilisateur InfluxDB pour interroger nos données. Ou bien la visualisation on utilisant Grafana





****

**Bibliographie**

[1] Lien data

Source donnée météorologique : <https://www.kaggle.com/datasets/PROPPG-PPG/hourly-weather-surface-brazil-southeast-region?select=south.csv>

[2] État de l’art sur les projets type Apache et des systèmes de bases de données traitant des données horodatées :

<https://mcherif.wordpress.com/2015/08/07/big-data-kafka-concepts-generaux/>

https://www.journaldunet.fr/web-tech/guide-de-l-entreprise-digitale/1443846-influxdb-une-base-de-donnees-time-series-open-source-sur-optimisee/

<https://www.ionos.fr/digitalguide/hebergement/aspects-techniques/quest-ce-quinfluxdb/>

<https://www.influxdata.com/blog/getting-started-apache-kafka-influxdb/>

<https://db-engines.com/en/system/InfluxDB%3BKdb%2B%3BTimescaleDB>

<https://solutionsreview.com/data-integration/the-best-open-source-data-streaming-software-and-tools/>

[3] État de l’art sur les méthodes d’apprentissage (statistiques / machine Learning) adaptées aux séries temporelles en flux :

https://datascientest.com/le-modele-var

<https://eduscol.education.fr/sti/sites/eduscol.education.fr.sti/files/ressources/pedagogiques/14762/14762-series-temporelles-et-reseaux-de-neurones-recurrents-ensps.pdf>

<https://www.youtube.com/watch?v=8FCDpFhd1zk>

<https://www.machinelearningplus.com/time-series/arima-model-time-series-forecasting-python/>

<https://www.machinelearningplus.com/time-series/vector-autoregression-examples-python/>

[4] code partie kafka

[5] code partie influxDB

<https://www.conix.fr/les-time-series-databases/>

<https://stackoverflow.com/questions/3518778/how-do-i-read-csv-data-into-a-record-array-in-numpy>

ocs.influxdata.com/influxdb/v1.8/introduction/get-started/#:~:text=A%20fresh%20install%20of%20InfluxDB,database%20you%20wish%20to%20create.