Rapport du Travail d’étude et de Recherche

Traitement des flux de données horodatées



Master 2 Informatique SID

SEMESTRE 2

**Année 2021 - 2022**

|  |  |  |
| --- | --- | --- |
| **Membres du Projets** | Khadija LAMNAOUR | [khadija.lamnaour@etu.univ-amu.fr](mailto:khadija.lamnaour@etu.univ-amu.fr) |
|  | Hadja Maimouna BALDE | Hadjamaimouna.BALDE@etu.univ-amu.fr |
| **Encadrant** | Omar BOUCELMA | [Omar.boucelma@univ-amu.fr](mailto:Omar.boucelma@univ-amu.fr) |
| **Version** | Version 1.1 |  |
| **Date** | 16/02/2023 |  |

**Sommaire**

[I. État de l’art 4](#_Toc127972664)

[I.1. État de l’art sur les systèmes (architectures data / logicielle) de traitement de données horodatées 4](#_Toc127972665)

[I.1.1. Définition des données horodatées 4](#_Toc127972666)

[I.1.3. Les solutions existantes pour traiter les données horodatées 4](#_Toc127972667)

[I.1.4. Comparaisons entre les alternatives de Kafka 4](#_Toc127972668)

[I.1.5. Architecture d’Apache Kafka 5](#_Toc127972669)

[I.1.6. Comparaison entre influxDB, KDB+ et timescaleDB 6](#_Toc127972670)

[I.1.7. Architecture influxDB 7](#_Toc127972671)

[I.2. État de l’art sur les méthodes d’apprentissage (statistiques / machine Learning) adaptées aux séries temporelles en flux 8](#_Toc127972672)

[I.2.1 Prétraitement des données dynamiques 8](#_Toc127972673)

[I.3. Approches algorithmiques pour la prévision des séries temporelles en flux 9](#_Toc127972674)

[I.3.1. Préparation des données dynamiques 9](#_Toc127972675)

[I.3.2. Modèles de prédiction de séries temporelles 9](#_Toc127972676)

[II. Solutions proposées pour réaliser ce projet 9](#_Toc127972677)

[II.1. Architecture des solutions proposées (l’Application) 10](#_Toc127972678)

[III.1. Présentation du jeu de données 11](#_Toc127972679)

[III.1.1. Jeux de données météorologique 11](#_Toc127972680)

[III.1.2 Description des données 12](#_Toc127972681)

[III.2 Modèles appliqués pour les prévisions de séries temporelle en flux 12](#_Toc127972682)

[III.2.1 Model ARIMA 12](#_Toc127972683)

[III.2.2 Model VAR 14](#_Toc127972684)

[III.3.3 Model Isolation Forest 16](#_Toc127972685)

[III.3.4. Comparaison des modèles : 17](#_Toc127972686)

[IV.1. Réception des données dans le topic Incoming-data 17](#_Toc127972687)

[IV.2. Réception des données dans le topic Anomalie 17](#_Toc127972688)

[IV.3 Anomalies détectés après la prédiction 17](#_Toc127972689)

[IV.4 Obtention et stockage de données des séries chronologiques avec Python et InfluxDB 18](#_Toc127972690)

**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 en flux. 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** et nous avons fini par traiter les 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**

Data analysis and time series modelling are most interesting when dealing with time-stamped or time series data in streams. They are generated continuously (in flow) in very large volumes and are often of variable quality. Furthermore, they are received with a short interval in order to be accurate in the analysis. We cleaned the data, performed pre-processing adapted to the characteristics of each model, created several models, and ended up processing the data using two systems: kafka and influxDB. Finally, we made an architecture of each system and created an application to display the result of the data. A comparison of these two systems allowed us to specify which of them is the best. The objective of this project is to propose a data science architecture and pipeline for the development of applications that manipulate data.

**Tableau des sigles**

Le tableau ci-dessous explique la signification de chaque sigle.

|  |  |
| --- | --- |
| **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 | Vectoriel autorégressive |
| ARIMA | Autoregressive-Integrated-Moving Average |

Tableau 1 : Tableau des sigles

**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**. Les données horodatées sont des données qui sont associées à une date et une heure précise, indiquant ainsi le moment où elles ont été créées, enregistrées ou mises à jour. Ces données sont souvent utilisées pour suivre l'évolution d'un événement au fil du temps ou pour effectuer des analyses temporelles.

Ce projet a pour objectif de proposer une **architecture** et un **pipeline** « Data Science » pour le développement d’applications qui manipulent des données horodatées en flux. 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* en flux 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. Et pour terminer nous développerons une application avec un jeu de données réel selon l’architecture proposée pour l’affichage des données.

# I. État de l’art

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

### I.1.1. Définition des données horodatées

Le traitement de flux de données horodatées est conçu :

* Pour le traitement instantané des données et leurs analyses en temps réel,
* Pour enregistrer aussi l’instant où 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, anomalies,
* la surveillance des performances etc.

I.1.2. Différence entre les séries temporelles en flux et séries temporelles en statique

Les séries temporelles en flux (streaming data en anglais) et les séries temporelles statiques (time séries en anglais) sont deux types de données liées au temps, mais ils présentent des différences importantes.

En résumé, la principale différence entre les séries temporelle en flux et les séries temporelles statiques réside dans la façon dont elles sont collectées et utilisées. Les séries temporelles en flux sont des données en constante évolution, générées en temps réel, nécessitant une analyse en temps réel, tandis que les séries temporelles statiques sont des données collectées à des intervalles réguliers dans le passé et sont utilisées pour étudier l'évolution d'un phénomène au fil du temps.

### I.1.3. 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.

### I.1.4. Comparaisons entre les alternatives de Kafka

Il existe plusieurs alternatives de Kafka qui permettent de gérer les séries temporelles en flux. Le tableau ci-dessous les plus populaires.

|  |  |  |
| --- | --- | --- |
| **Nom** | **Apache Kafka** | **Apache Flink** |
| 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. |
| Déploiement | Basé sur cloud | 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 |

Tableau 2 : Tableau de comparaison des alternatives de Kafka.

### I.1.5. Architecture d’Apache Kafka

À partir de la figure 1 ci-dessous on peut constater que Kafka possède un ensemble de composants assurant son fonctionnement en tant que système de messagerie distribué :

Les composants principaux sont :

* Topic : là où les données dans Kafka sont stockées ou enregistrées.
* Producer : c’est le système qui va produire les données dans Kafka et publierai les messages à un topic
* Broker : est un ensemble de serveurs ou nœuds formant un cluster Kafka, permettant de stocker et gérer les messages publiés
* Consumer : s’abonne 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 messages 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.

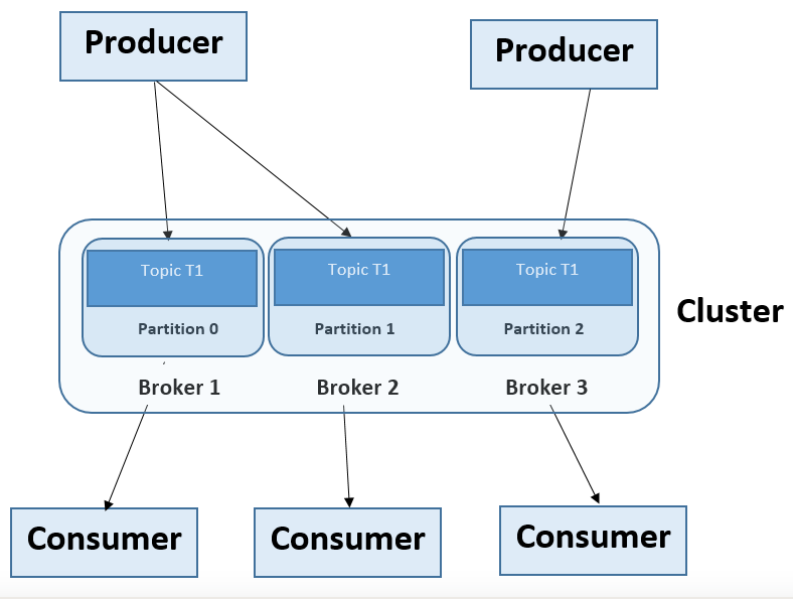


Figure 1 : schéma global du fonctionnement de Kafka.

### I.1.6. Comparaison entre influxDB, KDB+ et timescaleDB

Les bases de données conçues pour traiter les séries temporelles en flux sont multiples mais parmi les plus populaires on a KDB+ et influxDB.

|  |  |  |  |
| --- | --- | --- | --- |
| **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. | 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. | 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 |

Tableau 3 : Tableau de comparaison des alternatives d'influxDB.

### I.1.7. Architecture influxDB

InfluxDB est conçue 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.

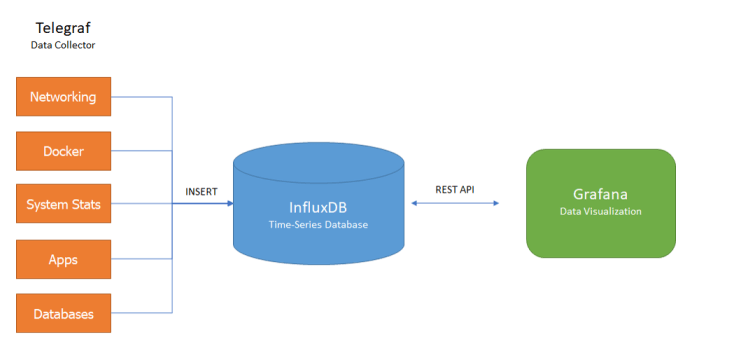


Figure 2 : illustration du fonctionnement InfluDB.

## I.2. État 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 statiques qui ont été proposés dans la littérature sont les suivants :

### I.2.1 Prétraitement des données dynamiques

#### I.2.1.1. Données manquantes

Pour traiter les données manquantes 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.

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 : utilisé 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.

#### I.2.1.2. Caractéristiques principales des séries temporaires en flux

Quant aux données de séries temporelles en flux, elles ont obligatoirement des composantes de séries temporelles en flux. L’extraction de ces caractéristiques peut s’effectuer via la décomposition STL (Seasonal and Trend decomposition using Loess).

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.

## I.3. Approches algorithmiques pour la prévision des séries temporelles en flux

### I.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.

### I.3.2. Modèles de prédiction de séries temporelles

Parmi les modèles existants, le modèle VAR 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.

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.
* LSTM : Long Short-Term Memory (LSTM) est un type de réseau neuronal récurrent qui peut apprendre la dépendance d'ordre entre les éléments d'une séquence. Il est souvent utilisé pour résoudre des problèmes de prévision de séries chronologiques
* 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.

# II. Solutions proposées pour réaliser ce projet

Pour ce projet, nous utiliserons Kafka et la base de données influxDB pour recevoir les données en temps réel qui sont les plus populaires pour traiter ces séries temporelles en flux (données horodatées).

Dans ce projet, pour la partie Kafka après avoir fait les prétraitement de ces données, nous nous servirons des modèles les plus familièrement utilisés dans la communauté de l’apprentissage automatique pour les prévisions de séries temporelle en flux comme VAR, ARIMA, 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 (Isolation Forest) mais on peut l’utiliser en prenant en considération les caractéristiques temporelles des données. Ensuite nous utiliserons une base de données SQLite pour stocker les données qui ne contiendront pas d’anomalies et pour l’affichage nous utiliserons un web serveur Django comme application pour afficher les données contenant des anomalies sur une page web.

## II.1. Architecture des solutions proposées (l’Application)

II.1.1 Avec Kafka

Nous avons utilisé Apache Kafka dans notre architecture décrite dans la figure ci-dessous, tout d’abord, on remarque que le **Producer** produit les données en entrée dans le topic incomming data, sur lesquelles on doit faire la **prédiction** pour vérifier s’il y’a des anomalies ou pas. Ensuiteun consommateur consomme les données reçues sur ce topic et analyse ces données pour vérifier s’il y’a des anomalies en utilisant un modèle de détection pré entrainer. Dans le cas où ils ne contiennent pas d’anomalie nous les stockons dans une base de données SQLite (MeteoDB) et s’il y’a des anomalies il les diffuse sur le **topic Anomalie**. Après nous avons un serveur Django qui écoute le topic anomalie, qui à chaque fois que les données arrivent sur ce topic il récupère ces informations et les affiches sur la page web.

.

**Producer**

**Serveur web**

**Django**

Consomme les données

Produit les données

Affiche les données sur une page web

**Topic**

**Anomalie**

**Topic**

**Incoming-data**

**Page web**

**Broker**

Produit un message s’il y’a anomalie

Lu les données

SQLITE

meteoDB

**Consumer**

Stocke les données qui ne contient pas d’anomalie dans la base meteoDB

Figure 3 : Architecture de la solution proposée avec kafka

II.1.2 Avec InfluxDB

III. Analyse Des Données

## III.1. Présentation du jeu de données

### III.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



Figure 4 : Visualisation des données de 2018 à 2020 avec la présence des données manquantes.

Les colonnes de ces données ont été renommé en français car de base elles étaient en espagnole et comme le montre la Figure 2, on peut remarquer la présence de données manquantes comportant la valeur **NaN**, que nous allons traiter.

### III.1.2 Description des données

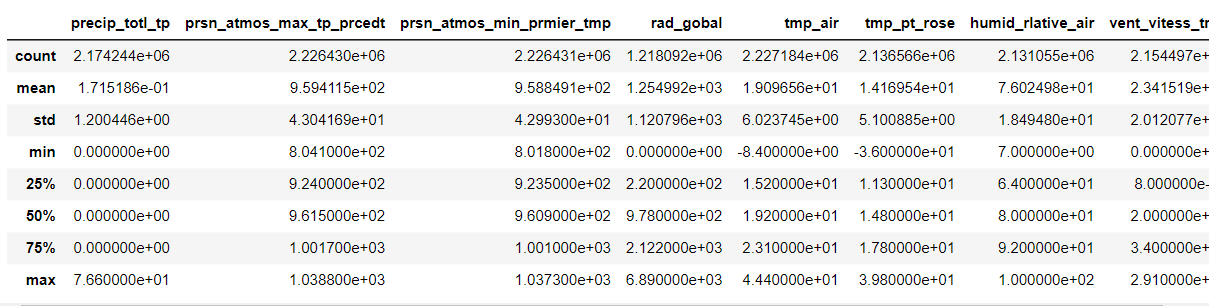


Figure 5 : visualisation de la description des données.

À partir de la figure 5 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.

III.1.3 Traitement des valeurs manquantes

Une image contenant texte

Description générée automatiquement

Figure 6 : Visualisation des capteurs 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 leurs présences.

## III.2 Modèles appliqués pour les prévisions de séries temporelle en flux

### III.2.1 Model ARIMA

Vu que c’est un modèle univarié nous avons choisi la date et la température de l'aire pour l’étudier comme il s’agit des données météorologiques.

#### III.2.1.1 Vérification des caractéristiques du model ARIMA

##### III.2.1.1.1 Vérification de la stationnarité de la série temporelle en flux

Pour vérifier si la série temporelle en flux est stationnaire nous avons utilisé le test statistique le plus couramment utilisé (Dickey-Fuller (DF)) pour analyser le stationnaire de la série*.* Il vérifie si une série chronologique est stationnaire autour de sa moyenne, ce qui est le cas si la valeur p est inférieur à un certain seuil critique (0,05) qui est le seuil le plus utilisé, sinon elle n’est pas stationnaire nous devons la différencier pour le rendre stationnaire.

Une image contenant texte

Description générée automatiquement

Figure 7 : stationnarité du model.

À partir de la figure 9 on peut constater que le modèle n’est pas stationnaire car le p-value obtenue est supérieur à 0,05 bien que la statistique ADF est proche des valeurs critiques.

##### III.2.1.1.2 Rendre la série chronologique stationnaire

Pour rendre la série stationnaire nous devons la différencier de 1, cette différenciation consiste à soustraire une observation antérieure de la série à partir de l’observation actuelle pour rendre la série plus stable et peut être répétée plusieurs fois pour obtenir une série chronologique plus stationnaire.

En général, la différentiation est utilisée pour éliminer la tendance et la saisonnalité d’une série chronologique, ce qui peut améliorer la qualité des prévisions et des modèles.

Une image contenant texte

Description générée automatiquement

Figure 8 : Stationnarité du model

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.

##### III.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). Nous avons utilisé la fonction auto\_arima qui nous a généré le meilleur paramètre.

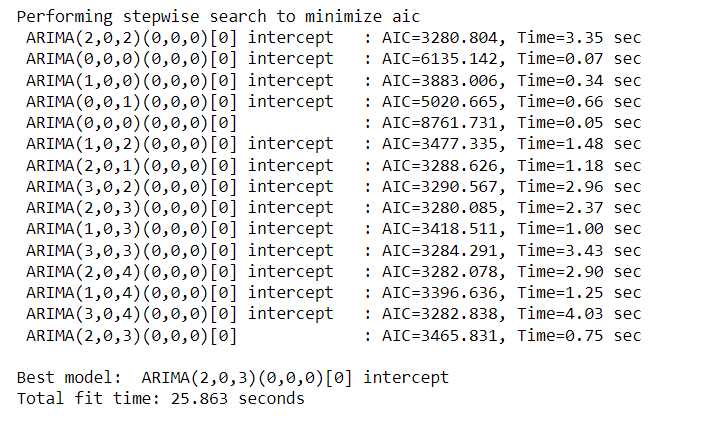


Figure 9 : Résultat de la sélection des paramètres d'ARIMA

À partir de la figure ci-dessus on voit que le model à (2,0,3) comme meilleur paramètre et comme nous avons déterminé les paramètres du model, nous pouvons maintenant utiliser le model pour faire des prévisions pour les valeurs futurs de la série Chronologiques.

### III.2.2 Model VAR

Le model VAR est un algo de prévision multivarié, notre choix de variable est basé sur la (tmp\_pt\_rose et rad\_global) pour l’entrainement de ce modèle, car à partir de la figure de la matrice de corrélation ci-dessous nous pouvons constater qu’ils ont une forte corrélation avec notre variable cible qui est la tmp\_air.

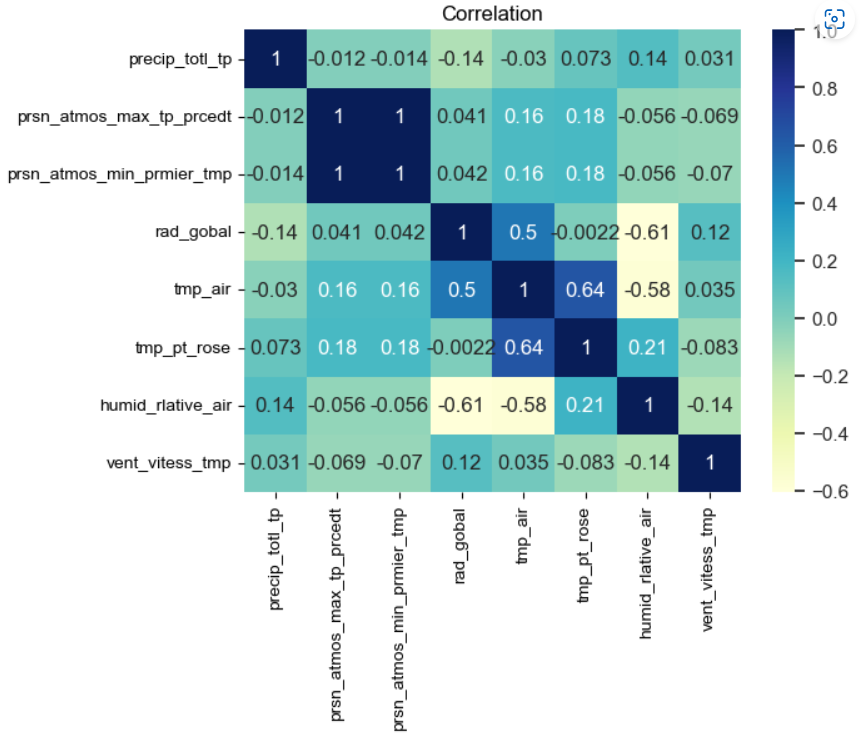


Figure 1*0* : Corrélation entre les *différents* capteurs du jeu de données météorologiques.

#### III.2.2.1 Vérification des caractéristiques du model

##### III.2.2.1.1 Vérification de la stationnarité des séries temporelles en flux

Comme expliqué au niveau du model ARIMA nous avons tester la stationnarité de toutes les séries de ce model aussi avec le test de Dickey-Fuller (DF)). Nous avons trouvé qu’une série n’était pas stationnaire et on a différencié toutes les séries car la différenciation d’une série réduit une cellule en moins, donc pour qu’elles aient la même taille à chaque différentiation on doit appliquer sur toutes les séries jusqu’à obtenir des séries stables.

Une image contenant texte

Description générée automatiquement

Figure 11 : résultat de la stationnarité

À partir de la figure ci-dessus on remarque que toutes les séries sont stationnaires car le p-value est inférieur à 0,05.

##### III.2.2.1.2. Tester la cointégration

##### Pour faire ce test nous avons utilisé la cointégration de Soren Johansen, qui est une méthode pour déterminer si plusieurs séries chronologiques sont cointégrées, c'est-à-dire si elles ont une relation de long terme stable. Les résultats sont interprétés en examinant les coefficients estimés pour les relations cointégrées.

Une image contenant texte

Description générée automatiquement

Figure 12 : Résultat de la cointégration.

À partir de figure ci-dessus on remarque que toutes les cellules sont cointégrées sauf la cellule tmp\_pt\_rose.

##### III.2.2.1.3 Comment sélectionner l'ordre (P) du modèle VAR

Une image contenant table

Description générée automatiquement

Figure 13 : Résultat de la sélection de l’ordre de p.

À partir de la figure ci-dessus on voit que le model à 12 comme meilleur ordre comme nous nous sommes basés sur le BIC, qui est le plus utilisé et vu que nous avons déterminé les paramètres du model, nous pouvons maintenant utiliser le model pour faire des prévisions pour les valeurs futurs de la série Chronologiques.

### III.3.3 Model Isolation Forest

C’est un algorithme non supervisé qui fonctionne avec tout type de données, il permet de détecter des anomalies que contienne des données, en utilisant un arbre de décision pour isoler les données anormales. Pour l’entrainer nous l’avons appliqué les mêmes caractéristiques que le model ARIMA, excepté le choix des paramètres, car c’est ce qui fait la particularité de ARIMA. Après l’entrainement nous l’avons dumper sur un fichier et on lui a utiliser pour la prédiction.

## III.3.4. Comparaison des modèles :

Nous avons comparé ces 3 modèles théoriquement mais pour la prédiction nous avons utilisé le model Isolation Forest car il est plus efficace dans le cadre de la détection d’anomalie que les deux autres algorithmes.

IV. Expérimentations

## IV.1. Visualisation de la réception des données dans le topic Incoming-data

Une image contenant texte

Description générée automatiquement

Figure 14 : Visualisation du topic incoming-data.

La figure ci-dessus nous montre les informations reçues dans le topic incoming-data qui ont été produite sur ce topic.

## IV.2. Visualisation de la Réception des données dans le topic Anomalie

Une image contenant texte

Description générée automatiquement

Figure 15*: Visualisation du t*opic incoming-data.

La figure ci-dessus nous montre les informations reçues dans le topic Anomalie qui ont été produite sur ce topic.

## IV.3 Visualisation des anomalies détectés après la prédiction

À partir de la figure 16 on peut voir les données qui contiennent des anomalies à partir de la page web du serveur Django.

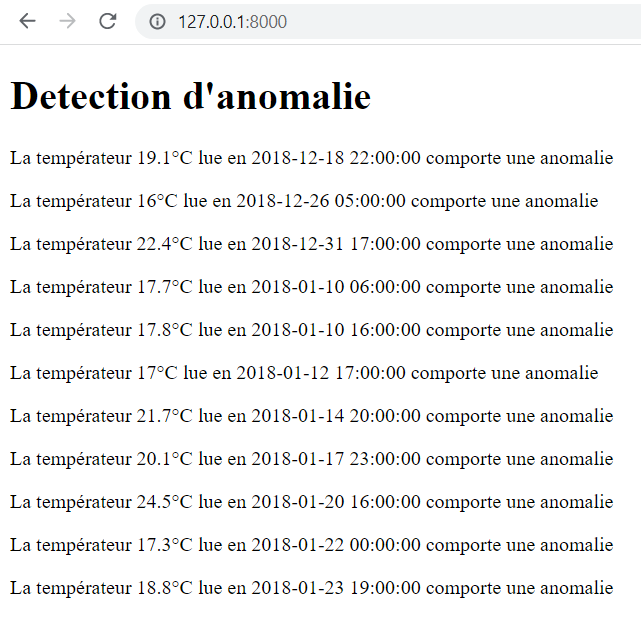


Figure 16 : Visualisation des anomalies détectées à partir de la page web.

## IV.4 Obtention et stockage de données des 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 utilisé la version 1.8 car la version 2.\* a un problème d'authentification(unauthorized)}

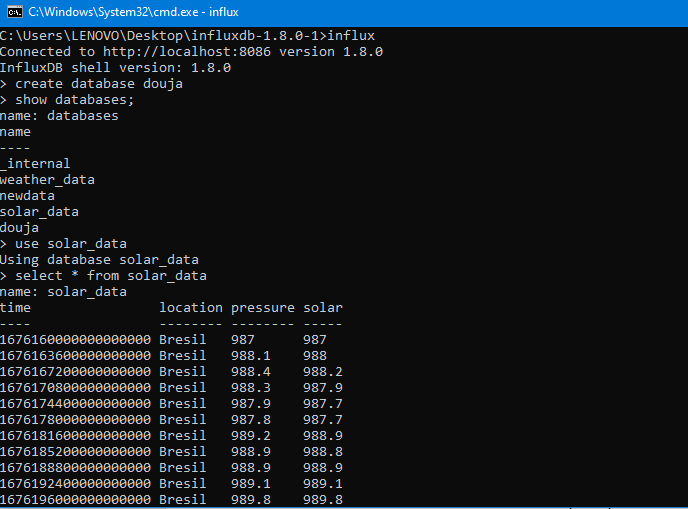


Figure 17 : Confirmation de la création de la base de données

Après initialisation d'un client influxdb pour l’importation des 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 intéressent 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, supprimons également les colonnes que nous ne voulons pas écrire dans InfluxDB. Ensuite instancié la bibliothèque client InfluxDB Python et écrire le DataFrame dans InfluxDB, spécifions un nom de mesure. À rappeler qu’une mesure contienne des données dans un compartiment, on peut le considérer comme le deuxième niveau hiérarchique le plus élevé d'organisation des données dans InfluxDB après un compartiment. Finalment l’importation du data vers la base de données "solar\_data" de influxdb .

Une image contenant texte

Description générée automatiquement

Figure 18 : Confirmation du stockage de la base de donnée

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



Figure 19 : Visualisation des données solar en fonction du temps.

La fifure ce dessus nous montre



Figure 20 : illustration de l’affichage en point

****

Figure 21 : illustration de l’affichage en ligne

Conclusion

Ce travail d’étude et de recherche, nous a permis d’avoir une ouverture d’esprit sur le monde de la recherche et de la curiosité à savoir comment les nouvelles technologies apparaissent.

Les principales difficultés rencontrées dans ce TER sont : savoir comment connecter python à ces deux systèmes (Apache Kafka et influxDB), la diffusion des données dans apache kafka, ensuite comment faire la prédiction après avoir consommé les données.

Globalement ce travail nous a fait découvrir le fonctionnement des données horodatées et comment les manipuler. Il nous a aussi permis d’avoir une expérience solide sur le traitement de ces données avec deux systèmes différents (Apache Kafka et influxDB), qui nous a permis de produire les affichages présentés précédemment.

Malgré toutes ces difficultés rencontrées nous avons réussi à atteindre l’objectif avec Apache Kafka et le travail a été très riche en connaissance et manipulation dans le traitement de ces données, un domaine qui passionne chaque membre de l’équipe.

**Bibliographie**

[1] Lien data

Source donnée météorologique

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

[2] Référence 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] Référence état de l’art sur les méthodes d’apprentissage (statistiques / machine Learning) adaptées aux séries temporelles en flux

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

[4] Référence code Kafka

<https://towardsdatascience.com/real-time-anomaly-detection-with-apache-kafka-and-python-3a40281c01c9>

https://kafka.apache.org/30/documentation.html

https://kafka.apache.org/intro

<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/>

<https://kafka.apache.org/>

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

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

[5] Référence code 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.

<https://www.influxdata.com/>

[6] Django

<https://developer.mozilla.org/fr/docs/Learn/Server-side/Django/skeleton_website>

[7] Référence Chercheur

[7.1] Silvia Nittel: Real-time sensor data streams, ACM SIGSPATIAL Special, 7(2):22-28, 2015.

[7.2] Sandra Garcia-Rodriguez, Mohammad Alshaer, Cedric Gouy-Pailler: STREAMER: A Powerful Framework for Continuous Learning in Data Streams. CIKM '20: Proceedings of the 29th ACM International Conference on Information & Knowledge Management.

[7.3] Albert Bifet, João Gama: IoT data stream analytics. Ann. des Télécommunications 75(9-10): 491-492 (2020)

[7.4] Debnath P Mukherjee, Suman Datta: Incremental time series algorithms for IoT analytics: an example from autoregression, ICDCN '16: Proceedings of the 17th International Conference on Distributed Computing and Networking.

[7.5] Mateusz Fedoryszak, Brent Frederic, Vijay Rajaram, Changtao Zhong: Real-time Event Detection on Social Data Streams. KDD '19: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.

[7.6] Seungjun Kim, Dan Tasse, and Anind K. Dey. 2017.   
Making Machine-Learning Applications for Time-Series Sensor Data Graphical and Interactive. ACM Trans. Interact. Intell. Syst. 7, 2, Article 8 (July 2017), 30 pages. DOI:https://doi.org/10.1145/2983924