Projet : Système de Surveillance des Données de Pression Artérielle

Avec Kafka, FHIR, Elasticsearch et Kibana

M1 BIDABI 25/26

Par : BASKARA Havar
CISSE Zeinab Karissy
GUEYE Palla

Sommaire :

Introduction..... 3

1. Le standard FHIR et la modélisation des données médicales.....3
1.1. Le rôle de FHIR dans l'interopérabilité médicale.....4
1.2. La ressource Observation et sa structure..... 4
1.3. Implémentation dans le projet (fhir.resources)..... 5
1.4. Exemple de message FHIR généré..... 5
2. Génération des données de pression artérielle..... 6
2.1. Approche de simulation et justification.....6
2.2. Utilisation de Faker pour générer des patients.....7
2.3. Structure du script generate_fhir.py..... 7
2.4. Exemples de données générées.....8
2.5. Distribution des valeurs systoliques/diastoliques..... 8
3. Architecture technique du système..... 8
3.1. Vue d'ensemble du pipeline..... 9
3.2. Kafka : rôle, fonctionnement, topics..... 9
3.3. Producer Kafka : envoi des messages FHIR..... 9
3.4. Consumer Kafka : récupération et analyse..... 10
3.5. Elasticsearch et Kibana : indexation et visualisation.....10
3.6. Stockage local des données normales.....10
3.7. Schéma global de l'architecture..... 11
4. Analyse des données de pression artérielle..... 11
4.1. Seuils cliniques (OMS / HAS).....11
4.2. Détection des anomalies dans le consumer.....12
4.3. Typologie des anomalies : hypertension, hypotension, cas critiques.....12
4.4. Exemples de détection d'anomalies.....13
4.5. Tableau récapitulatif des seuils..... 13
5. Traitement, stockage et indexation des données..... 13
5.1. Données normales : structure et archivage JSON..... 14
5.2. Données anormales : indexation dans Elasticsearch.....14
5.3. Structure du mapping Elasticsearch.....14
5.4. Gestion des erreurs et logs (anomalies.json).....15
5.5. Exemples de fichiers générés..... 15
6. Visualisation et analyse avec Kibana.....15
6.1. Configuration de l'index dans Kibana.....16
6.2. Création des dashboards..... 16
6.3. Analyse détaillée des visualisations..... 16
6.3.1. Répartition des anomalies : Hypertension vs Hypotension..... 17
6.3.2. Distribution des patients anormaux..... 17
6.3.3. Surveillance temporelle des anomalies par patient.....18
6.3.4. Indices de sévérité systolique et diastolique..... 19

6.4. Analyse des tendances..... 19
6.5. Exemples de captures d'écran..... 20
7. Intégration d'un modèle de Machine Learning..... 20
7.1. Motivation : limites des règles statiques..... 20
7.2. Préparation des données : features et labels..... 21
7.3. Modèle supervisé : régression logistique..... 21
7.4. Entraînement et évaluation du modèle..... 21
7.5. Intégration dans le consumer Kafka..... 22
7.6. Exemple de prédiction en temps réel..... 22
7.7. Limites et perspectives..... 22
8. Résultats et interprétation..... 23
8.1. Statistiques globales du pipeline..... 23
8.2. Répartition des patients normaux vs anormaux..... 24
8.3. Analyse des anomalies détectées..... 24
8.4. Performance du système..... 25
8.5. Graphiques et tableaux de synthèse..... 25
9. Discussion..... 26
9.1. Points forts du système..... 26
9.2. Limites techniques et méthodologiques..... 26
9.3. Améliorations possibles..... 27
9.4. Perspectives pour un système hospitalier réel..... 27

Conclusion..... 27
Annexes..... 29

Introduction

La surveillance de la pression artérielle constitue un enjeu majeur dans le domaine de la santé publique. L'hypertension comme l'hypotension figurent parmi les anomalies physiologiques les plus courantes, et peuvent entraîner des complications graves lorsqu'elles ne sont pas détectées à temps. Dans un contexte où les établissements de santé doivent gérer des volumes croissants de données médicales, la capacité à analyser ces informations en continu, à identifier rapidement les situations critiques et à déclencher des alertes fiables devient essentielle. Les technologies de traitement de données en temps réel offrent aujourd'hui des opportunités nouvelles pour renforcer la qualité du suivi clinique et améliorer la réactivité des équipes soignantes.

Parallèlement, l'écosystème de la santé évolue vers des standards d'interopérabilité permettant de structurer et d'échanger les données de manière homogène. Le standard FHIR (Fast Healthcare Interoperability Resources), développé par HL7, s'est imposé comme une référence internationale pour représenter les données médicales sous forme de ressources modulaires, lisibles et exploitables par des systèmes hétérogènes. Son adoption facilite la communication entre applications, plateformes et infrastructures hospitalières, tout en garantissant une cohérence dans la représentation des informations cliniques.

Dans ce contexte, ce projet s'inscrit dans une démarche de conception d'un système complet de surveillance de la pression artérielle, capable de traiter des données médicales structurées en temps réel. L'objectif est de mettre en place une architecture Big Data reposant sur plusieurs technologies complémentaires : génération de données FHIR, ingestion via Kafka, analyse automatisée des mesures, détection d'anomalies, stockage différencié selon le niveau de risque, et visualisation des cas critiques dans Elasticsearch et Kibana. L'intégration d'un module de machine learning vient enrichir l'approche en apportant une dimension prédictive, permettant d'aller au-delà des simples règles basées sur des seuils cliniques.

Le système développé vise ainsi à illustrer la manière dont les technologies de streaming, d'indexation et d'intelligence artificielle peuvent être combinées pour répondre à des besoins concrets de surveillance médicale. Il s'agit d'un projet complet, allant de la génération des données jusqu'à leur exploitation visuelle, en passant par l'analyse, la classification et la gestion des anomalies. Ce rapport présente l'ensemble des étapes de conception, les choix techniques réalisés, les résultats obtenus et les perspectives d'amélioration possibles.

1. Le standard FHIR et la modélisation des données médicales

L'utilisation d'un standard d'interopérabilité est devenue indispensable dans les systèmes d'information hospitaliers modernes. Les établissements de santé manipulent quotidiennement des données provenant de sources multiples : dossiers patients informatisés, systèmes de monitoring, laboratoires, services d'imagerie, dispositifs médicaux connectés, etc. Sans un langage commun, ces systèmes restent cloisonnés, ce qui complique l'échange

d'informations, augmente les risques d'erreurs et freine la mise en place de solutions innovantes. C'est dans ce contexte que s'inscrit **FHIR (Fast Healthcare Interoperability Resources)**, un standard international développé par HL7, conçu pour faciliter la structuration, la transmission et l'exploitation des données médicales.

1.1. Le rôle de FHIR dans l'interopérabilité médicale

FHIR occupe aujourd'hui une place centrale dans la transformation numérique des systèmes de santé. Contrairement aux standards historiques, souvent lourds et difficiles à implémenter, FHIR adopte une approche moderne, inspirée des technologies web. Il repose sur des formats légers comme **JSON** ou **XML**, et sur des principes **REST**, ce qui le rend compatible avec les architectures distribuées, les API modernes et les pipelines Big Data.
L'un des atouts majeurs de FHIR est sa structuration en **ressources**, chacune correspondant à un concept clinique précis : Patient, Observation, DiagnosticReport, Medication, Encounter, etc. Cette organisation modulaire permet de représenter des données complexes de manière claire, cohérente et extensible. Chaque ressource est indépendante, réutilisable et peut être enrichie selon les besoins d'un établissement ou d'un projet.

Dans le cadre de ce projet, FHIR joue un rôle fondamental. Il garantit que les données de pression artérielle générées par le script Python et transmises via Kafka respectent un format standardisé, compréhensible par n'importe quel système conforme à FHIR. Cela signifie qu'un hôpital, un logiciel de dossier patient ou un service d'analyse pourrait consommer ces données sans transformation complexe. L'utilisation de FHIR assure donc une **interopérabilité native**, indispensable pour envisager une intégration dans un environnement hospitalier réel.

1.2. La ressource Observation et sa structure

Parmi les nombreuses ressources proposées par FHIR, la ressource **Observation** est l'une des plus utilisées. Elle permet de représenter toute mesure clinique : tension artérielle, température, fréquence cardiaque, saturation en oxygène, résultats biologiques, etc. Dans le cas de la pression artérielle, FHIR propose une structure normalisée permettant de distinguer clairement la pression systolique et la pression diastolique.

Une ressource Observation contient généralement plusieurs éléments essentiels :

- **identifier** — identifiant unique de l'observation
- **status** — statut de la mesure (final, amended, etc.)
- **category** — catégorie clinique (vital signs, laboratory, etc.)
- **code** — code LOINC décrivant la nature de la mesure
- **subject** — référence au patient concerné
- **effectiveDateTime** — date et heure de la mesure
- **valueQuantity** ou **component** — valeurs mesurées
- **unités** — par exemple *mmHg* pour la pression artérielle

Pour la pression artérielle, FHIR recommande l'utilisation de la structure **component**, qui permet de regrouper systolique et diastolique dans une même observation. Chaque composant possède son propre code LOINC, garantissant une interprétation correcte et universelle :

- **8480-6** : pression systolique
- **8462-4** : pression diastolique

Cette structuration assure que n'importe quel système conforme à FHIR peut comprendre, interpréter et exploiter la mesure, indépendamment de la technologie utilisée ou du pays dans lequel elle est produite.

1.3. Implémentation dans le projet (fhir.resources)

Dans ce projet, la génération des données FHIR est réalisée en Python grâce à la bibliothèque **fhir.resources**, qui fournit des classes correspondant aux ressources officielles du standard. Cette approche présente plusieurs avantages majeurs. Elle garantit d'abord la **conformité stricte** au standard FHIR : chaque ressource créée respecte la structure attendue, les champs obligatoires et les types de données définis par HL7. Elle permet également d'éviter les erreurs de structure JSON, fréquentes lorsqu'on construit manuellement des objets complexes.
La bibliothèque facilite aussi la **validation automatique** des ressources, ce qui renforce la fiabilité du pipeline. Enfin, elle simplifie la sérialisation en JSON, indispensable pour l'envoi des messages dans Kafka.

Le script generate_fhir.py crée ainsi, pour chaque patient fictif, une ressource Observation complète comprenant :

- un identifiant unique,
- un patient généré avec Faker,
- une date et une heure de mesure,
- les valeurs systolique et diastolique,
- les codes LOINC appropriés,
- les unités de mesure,
- la structure component recommandée par FHIR.

Une fois générées, ces ressources sont converties en JSON et transmises au Producer Kafka, qui les publie dans le topic dédié.

1.4. Exemple de message FHIR généré

L'exemple suivant illustre une observation FHIR générée dans le cadre du projet. Il s'agit d'un extrait JSON conforme au standard FHIR, représentant une mesure de pression artérielle:

```python
def generate_fhir():
    # Générer données patient
    patient_id = fake.uuid4()
    systolic = random.randint(80, 180)
    diastolic = random.randint(50, 120)

    observation_data = {
        "resourceType": "Observation",
        "status": "final",
        "code": {
            "text": "Blood Pressure"
        },
        "subject": {
            "reference": f"Patient/{patient_id}"
        },
        "effectiveDateTime": datetime.now(timezone.utc).isoformat(),
        "component": [
            {
                "code": {"text": "Systolic Blood Pressure"},
                "valueQuantity": {
                    "value": systolic,
                    "unit": "mmHg"
                }
            },
            {
                "code": {"text": "Diastolic Blood Pressure"},
                "valueQuantity": {
                    "value": diastolic,
                    "unit": "mmHg"}
```

Le script complet est disponible dans le dépôt GitHub du projet.

Cet exemple illustre la structure complète d'une observation FHIR conforme, telle qu'elle est générée automatiquement par le script Python avant d'être envoyée dans Kafka. Il constitue la base du pipeline de traitement et d'analyse développé dans les sections suivantes.

2. Génération des données de pression artérielle

La génération des données constitue la première étape du pipeline et joue un rôle fondamental dans la mise en place d'un système de surveillance médicale simulé. Dans un environnement réel, les mesures de pression artérielle proviennent de dispositifs médicaux connectés, de tensiomètres électroniques ou de systèmes d'information hospitaliers. Cependant, dans le cadre de ce projet, aucune source réelle n'est disponible. Il est donc nécessaire de produire artificiellement des observations médicales réalistes, structurées et conformes au standard FHIR. Cette étape de simulation n'est pas un simple artifice technique : elle permet de tester l'ensemble du pipeline dans des conditions proches de la réalité, tout en garantissant la reproductibilité des expériences et la maîtrise du volume de données générées.

2.1. Approche de simulation et justification

La simulation des données répond à plusieurs objectifs essentiels. Elle permet d'abord de tester l'architecture Kafka sans dépendre d'un flux réel, ce qui est indispensable dans un contexte pédagogique ou expérimental. Elle offre également la possibilité de générer un volume contrôlé de données, adapté aux besoins du projet : suffisamment important pour valider la robustesse du pipeline, mais sans excès pour éviter une surcharge inutile.

La simulation garantit aussi la cohérence physiologique des valeurs produites. Les mesures systoliques et diastoliques sont générées dans des plages compatibles avec la physiologie

humaine, ce qui permet de reproduire des situations normales, mais aussi des cas d'hypertension ou d'hypotension. Enfin, cette approche assure une parfaite reproductibilité des tests : en utilisant les mêmes paramètres, il est possible de régénérer un jeu de données identique, ce qui facilite la validation et la comparaison des résultats.
Cette étape constitue ainsi un pilier essentiel du projet, car elle permet de valider l'ensemble du pipeline avant un éventuel déploiement sur des données réelles.

2.2. Utilisation de Faker pour générer des patients

Pour rendre les données plus réalistes, la bibliothèque **Faker** est utilisée afin de créer des identités fictives de patients. Faker permet de générer automatiquement des informations telles que le nom, le prénom, la date de naissance ou encore un identifiant patient unique. Ces données, bien qu'entièrement fictives, imitent la structure des informations présentes dans un système hospitalier réel.

Les observations générées incluent également la date et l'heure de la mesure, ainsi que les valeurs systoliques et diastoliques. L'intérêt de Faker réside dans sa capacité à produire des données :

- **aléatoires**, ce qui permet de simuler un flux continu et varié ;
- **réalistes**, en reproduisant la diversité des profils patients ;
- **non sensibles**, ce qui évite tout problème de confidentialité ou de conformité réglementaire.

Une fois générées, ces informations sont intégrées dans une ressource **FHIR Observation**, conformément au standard HL7. Cela garantit que les données produites respectent les normes d'interopérabilité utilisées dans les systèmes de santé modernes.

2.3. Structure du script generate_fhir.py

Le script generate_fhir.py constitue le cœur de la génération des données. Il remplit trois fonctions principales qui s'enchaînent de manière cohérente pour produire une observation complète.
La première étape consiste à créer un patient fictif à l'aide de Faker. Le script génère une identité complète, incluant un identifiant unique, une date de naissance et des informations démographiques. Cette identité est ensuite associée à une mesure de pression artérielle.

La deuxième étape concerne la génération des valeurs systolique et diastolique. Le script utilise des plages de valeurs réalistes : la pression systolique varie généralement entre 90 et 180 mmHg, tandis que la pression diastolique se situe entre 60 et 120 mmHg. Ces plages permettent de produire à la fois des valeurs normales et des valeurs anormales, ce qui est indispensable pour tester la détection d'anomalies dans le consumer Kafka.

Enfin, la troisième étape consiste à construire une ressource FHIR Observation complète en utilisant la librairie fhir.resources. Cette librairie garantit que le JSON généré respecte

strictement la structure définie par le standard FHIR. Le script retourne ainsi un message FHIR prêt à être envoyé dans Kafka par le Producer.

2.4. Exemples de données générées

Les données générées par le script se présentent sous la forme d'un dictionnaire Python contenant les informations essentielles de la mesure, ainsi qu'un champ fhir_observation qui encapsule la ressource FHIR complète. Un exemple simplifié peut ressembler à ceci :

Le champ fhir_observation contient l'intégralité de la ressource FHIR Observation, telle que décrite dans la section 1.4. Ce format intermédiaire est particulièrement utile pour le Producer Kafka, qui peut sérialiser facilement les données avant de les envoyer dans le topic.

2.5. Distribution des valeurs systoliques/diastoliques

La génération aléatoire des valeurs suit une distribution volontairement contrôlée. Environ **80 %** des valeurs produites se situent dans une plage normale (systolique entre 100 et 140 mmHg, diastolique entre 60 et 90 mmHg). Les **20 % restants** correspondent à des valeurs anormales, générées pour tester la capacité du consumer à détecter les anomalies.

Ces anomalies peuvent prendre deux formes :

- **hypertension**, lorsque la pression systolique dépasse 140 mmHg ou que la diastolique dépasse 90 mmHg ;
- **hypotension**, lorsque la pression systolique descend en dessous de 90 mmHg ou que la diastolique est inférieure à 60 mmHg.

Cette distribution contrôlée permet de valider le fonctionnement du consumer Kafka, de vérifier que les anomalies sont correctement détectées et d'alimenter Elasticsearch avec suffisamment de cas critiques pour permettre une visualisation pertinente dans Kibana.

3. Architecture technique du système

L'architecture mise en place dans ce projet repose sur une chaîne de traitement cohérente et progressive, conçue pour simuler un véritable système de surveillance médicale en temps réel. L'objectif est de transformer des observations médicales brutes, générées artificiellement, en informations exploitables permettant d'identifier rapidement les patients présentant des anomalies de pression artérielle. Pour cela, plusieurs composants interagissent : un module de génération de données, un système de messagerie Kafka, un consumer chargé de l'analyse, un stockage différencié selon le niveau de risque, et enfin une couche de visualisation reposant sur Elasticsearch et Kibana. L'ensemble constitue un pipeline complet, inspiré des architectures Big Data utilisées dans les environnements hospitaliers modernes.

3.1. Vue d'ensemble du pipeline

Le pipeline suit une logique simple mais robuste : les données sont d'abord créées sous forme de ressources FHIR, puis transmises dans Kafka, analysées par un consumer, et enfin orientées vers différents stockages selon leur criticité. Cette organisation permet de séparer clairement les responsabilités : la génération des données est indépendante de leur transport, qui lui-même est indépendant de leur analyse. Cette modularité est essentielle dans les architectures distribuées, car elle permet de remplacer ou d'améliorer un composant sans impacter les autres. Dans notre cas, les observations médicales circulent donc dans un flux continu :

génération → publication → consommation → analyse → stockage → visualisation.

Chaque étape joue un rôle précis dans la transformation progressive des données.

3.2. Kafka : rôle, fonctionnement, topics

Kafka occupe une place centrale dans cette architecture. Il agit comme un système de messagerie distribué, capable de recevoir un grand nombre de messages et de les mettre à disposition des consommateurs de manière fiable et ordonnée. Dans un contexte médical, où les données doivent être traitées sans perte et dans l'ordre d'arrivée, Kafka constitue un choix particulièrement adapté. Le fonctionnement repose sur un principe simple : les messages sont publiés dans un *topic*, qui joue le rôle d'un canal de communication. Le producer écrit dans ce topic, tandis que le consumer lit les messages au fur et à mesure. Cette séparation permet au système de rester stable même si un composant tombe temporairement en panne : Kafka conserve les messages jusqu'à ce qu'ils soient consommés.

Dans ce projet, un topic unique est utilisé pour centraliser toutes les observations de pression artérielle. Ce choix est suffisant pour un prototype, mais l'architecture pourrait facilement être étendue à plusieurs topics (par exemple un topic par service hospitalier, par type de mesure, ou par niveau de criticité). Kafka garantit également la scalabilité : si le flux de données augmente, il suffit d'ajouter des partitions ou des consumers supplémentaires pour absorber la charge.

3.3. Producer Kafka : envoi des messages FHIR

Le producer est le premier maillon du pipeline Kafka. Son rôle est d'envoyer les messages FHIR générés vers le topic Kafka. Il ne réalise aucune analyse, aucune transformation complexe : il se contente de transmettre les données telles qu'elles ont été produites. Cette simplicité est volontaire, car elle permet de maintenir une séparation nette entre la génération et le traitement. Le producer sérialise les ressources FHIR au format JSON, puis les publie dans Kafka. À chaque envoi, un message est ajouté au log distribué, où il restera disponible jusqu'à ce qu'un consumer le lise.

```python
from kafka import KafkaProducer
import json
import time
from generate_fhir import generate_fhir

# Création du producer Kafka
producer = KafkaProducer(
    bootstrap_servers="localhost:9092",
    value_serializer=lambda v: json.dumps(v).encode("utf-8")
)

topic = "test"

print("Envoi de 5 patients...")

# Envoi de seulement 5 patients
for i in range(5):
    data = generate_fhir()
    producer.send(topic, value=data)
    print(f"Patient {i+1} envoyé")
    time.sleep(2)

print("Envoi terminé")

producer.flush()
producer.close()
```

Le producer peut fonctionner en continu, en envoyant des observations à intervalles réguliers, ou bien en mode batch, en envoyant un ensemble de messages d'un coup. Dans le cadre de ce projet, il est généralement exécuté après la génération des données, mais il pourrait tout aussi bien être intégré dans un système en temps réel, recevant des mesures provenant de dispositifs médicaux connectés.

3.4. Consumer Kafka : récupération et analyse

Le consumer est le cœur du système, car c'est lui qui transforme les données brutes en informations cliniques exploitables. Il se connecte au même topic Kafka que le producer et lit les messages un par un. Chaque message est désérialisé, puis analysé pour extraire les valeurs systolique et diastolique. Ces valeurs sont ensuite comparées aux seuils cliniques définis dans le projet. Si la pression artérielle dépasse les limites normales, le patient est considéré comme présentant une anomalie ; sinon, il est classé comme patient normal.

```python
from kafka import KafkaConsumer
import json

consumer = KafkaConsumer(
    'topic',
    bootstrap_servers='localhost:9092',
    auto_offset_reset='earliest',
    enable_auto_commit=True,
    value_deserializer=lambda x: json.loads(x.decode('utf-8'))
)

print("Consumer démarré...")

for message in consumer:
    data = message.value

    print("\nMessage reçu :")
    print(data)
```

Le consumer joue donc un rôle de filtre intelligent : il trie les données selon leur criticité et les oriente vers le stockage approprié. Cette logique de traitement pourrait être enrichie par un modèle de machine learning (voir section 7), mais même dans sa version actuelle, elle permet déjà de simuler un système de surveillance médicale automatisé. Le consumer est également responsable de la journalisation des anomalies, ce qui permet de garder une trace des cas critiques détectés.

3.5. Elasticsearch et Kibana : indexation et visualisation

Les patients présentant une pression artérielle anormale ne sont pas simplement enregistrés dans un fichier : leurs données sont indexées dans Elasticsearch, un moteur de recherche et d'analyse particulièrement performant. L'intérêt d'Elasticsearch est double : il permet d'effectuer des recherches rapides sur les patients à risque, et il offre des capacités d'agrégation permettant d'analyser les tendances globales. Chaque document indexé contient les informations essentielles : identifiant du patient, valeurs systolique et diastolique, type d'anomalie, date de la mesure, etc.

Kibana, qui se connecte directement à Elasticsearch, constitue la couche de visualisation du système. Grâce à lui, il est possible de créer des tableaux de bord interactifs permettant de suivre l'évolution des anomalies, d'identifier les périodes critiques, ou encore de visualiser la répartition des patients selon le type d'anomalie. Cette dimension visuelle est essentielle dans un contexte médical, car elle permet de transformer des données techniques en indicateurs compréhensibles par des professionnels de santé.

3.6. Stockage local des données normales

Contrairement aux cas anormaux, les patients dont la pression artérielle est normale ne sont pas indexés dans Elasticsearch. Ils sont simplement enregistrés dans un fichier JSON local. Ce choix permet de ne pas surcharger Elasticsearch avec des données peu critiques, tout en conservant une trace complète des mesures. Le stockage local est léger, rapide et suffisant pour des données non prioritaires. Il permet également de réanalyser les données ultérieurement si nécessaire, ou de les utiliser pour entraîner un modèle de machine learning.

Cette séparation entre stockage local et indexation dans Elasticsearch reflète une logique de priorisation : seules les données présentant un intérêt clinique immédiat sont envoyées vers la couche d'analyse avancée.

3.7. Schéma global de l'architecture

L'ensemble du système peut être résumé comme un flux continu où chaque composant joue un rôle précis. Les données sont d'abord générées sous forme de ressources FHIR, puis transmises dans Kafka par le producer. Le consumer lit ces messages, les analyse et décide de leur destination : stockage local pour les cas normaux, indexation dans Elasticsearch pour les cas anormaux. Enfin, Kibana permet de visualiser les anomalies et d'en suivre l'évolution. Ce schéma illustre une architecture modulaire, extensible et réaliste, proche des systèmes utilisés dans les environnements hospitaliers modernes pour la surveillance en temps réel.

4. Analyse des données de pression artérielle

L'analyse des données de pression artérielle constitue le cœur clinique du système. C'est à ce stade que les mesures brutes, transmises sous forme de messages FHIR, sont interprétées pour déterminer si un patient présente un risque nécessitant une attention particulière. Cette

analyse repose à la fois sur des références médicales reconnues (OMS, HAS), sur une logique algorithmique simple mais robuste, et sur une typologie claire des anomalies. L'objectif est de reproduire, à l'échelle d'un pipeline Big Data, un raisonnement clinique élémentaire : distinguer les valeurs normales des valeurs pathologiques, et orienter les cas critiques vers un traitement spécifique.

4.1. Seuils cliniques (OMS / HAS)

Les seuils utilisés dans ce projet s'appuient sur les recommandations internationales en matière de pression artérielle, notamment celles de l'Organisation Mondiale de la Santé (OMS) et de la Haute Autorité de Santé (HAS). Ces organismes définissent des plages de valeurs considérées comme normales, élevées ou dangereusement basses. Ces seuils constituent la base de la logique de détection implémentée dans le consumer Kafka.

Dans la pratique clinique, une pression artérielle normale se situe généralement autour de 120/80 mmHg, avec une tolérance raisonnable. Les valeurs deviennent préoccupantes lorsque la pression systolique dépasse 140 mmHg ou lorsque la pression diastolique dépasse 90 mmHg, ce qui correspond à une situation d'hypertension. À l'inverse, une pression systolique inférieure à 90 mmHg ou une pression diastolique inférieure à 60 mmHg est considérée comme un signe d'hypotension, pouvant indiquer un risque de malaise, de choc ou d'insuffisance circulatoire.

Ces seuils, bien qu'ils simplifient la réalité clinique, permettent de mettre en place une logique de détection fiable et cohérente avec les standards médicaux.

4.2. Détection des anomalies dans le consumer

La détection des anomalies est réalisée dans le script consumer.py, qui constitue le moteur d'analyse du système. À chaque message reçu depuis Kafka, le consumer extrait les valeurs systolique et diastolique contenues dans la ressource FHIR. Ces valeurs sont ensuite comparées aux seuils définis précédemment.

La logique de détection repose sur un ensemble de conditions simples, mais suffisantes pour identifier les situations à risque. Si l'une des valeurs dépasse les limites normales, le patient est immédiatement classé comme anormal. Dans le cas contraire, il est considéré comme normal. Cette approche permet de traiter les données en temps réel, sans nécessiter de calculs complexes ou de modèles prédictifs.

Le consumer joue également un rôle de routage :

- les cas normaux sont archivés dans un fichier JSON local,
- les cas anormaux sont indexés dans Elasticsearch et consignés dans un fichier d'alertes.

Cette séparation garantit que les données critiques sont mises en avant et accessibles pour la visualisation dans Kibana.

4.3. Typologie des anomalies : hypertension, hypotension, cas critiques

Les anomalies détectées peuvent être regroupées en trois grandes catégories, chacune correspondant à une situation clinique distincte.

- Hypertension : elle se caractérise par une pression systolique supérieure à 140 mmHg ou une pression diastolique supérieure à 90 mmHg. Ce type d'anomalie est fréquent et peut indiquer un risque cardiovasculaire accru, notamment en cas de répétition ou de valeurs très élevées.
- Hypotension : elle survient lorsque la pression systolique descend en dessous de 90 mmHg ou que la pression diastolique est inférieure à 60 mmHg. L'hypotension peut être bénigne, mais elle peut aussi signaler un risque de malaise ou une insuffisance circulatoire.
- Cas critiques : il s'agit de situations où les valeurs dépassent largement les seuils d'alerte, par exemple une systolique supérieure à 180 mmHg ou une diastolique supérieure à 110 mmHg. Ces cas, bien que rares dans la simulation, sont particulièrement importants à détecter car ils nécessitent une intervention médicale urgente.

Cette typologie permet de structurer l'analyse et de faciliter la visualisation dans Kibana, où chaque type d'anomalie peut être représenté par une couleur ou un indicateur distinct.

4.4. Exemples de détection d'anomalies

Pour illustrer le fonctionnement du système, il est utile de présenter quelques exemples concrets de mesures analysées par le consumer.

- Une observation contenant une pression systolique de 152 mmHg et une diastolique de 96 mmHg sera immédiatement classée comme un cas d'hypertension. Le consumer l'orientera vers Elasticsearch, où elle sera indexée avec un champ anomaly_type = "hypertension".
- Une mesure affichant 85/55 mmHg sera identifiée comme un cas d'hypotension. Elle sera également indexée dans Elasticsearch, mais avec un type d'anomalie différent.
- À l'inverse, une mesure de 128/82 mmHg sera considérée comme normale. Elle sera simplement ajoutée au fichier donnees_normales.json, sans être envoyée vers Elasticsearch.

```
    # Exemple détection simple
    systolic = data["component"][0]["valueQuantity"]["value"]
    diastolic = data["component"][1]["valueQuantity"]["value"]

    if systolic > 140 or diastolic > 90:
        print("⚠️ Hypertension détectée")
    elif systolic < 90 or diastolic < 60:
        print("⚠️ Hypotension détectée")
    else:
        print("Pression normale")

# Détection avec variable anomaly
if systolic > 140 or diastolic > 90:
    anomaly = "Hypertension"
    print("⚠️ Hypertension détectée")
elif systolic < 90 or diastolic < 60:
    anomaly = "Hypotension"
    print("⚠️ Hypotension détectée")
else:
    anomaly = "Normal"
    print("Pression normale")
```

Ces exemples montrent comment le système transforme des données brutes en informations cliniques structurées, prêtes à être visualisées ou analysées.

4.5. Tableau récapitulatif des seuils

Pour synthétiser les règles de détection, le tableau suivant présente les seuils utilisés dans le projet :

Type de mesure	Valeur normale	Anomalie	Cas critique
Systolique	90–140 mmHg		> 180 mmHg
Diastolique	60–90 mmHg		> 110 mmHg

Ce tableau constitue une référence simple et claire pour comprendre la logique de classification appliquée par le consumer. Il permet également de vérifier la cohérence des résultats obtenus dans Elasticsearch et Kibana.

5. Traitement, stockage et indexation des données

Le traitement des données constitue l'étape où le système passe d'un simple flux d'observations médicales à une organisation structurée permettant d'exploiter efficacement les informations. Une fois les messages FHIR reçus par le consumer Kafka, ils sont analysés, classés, puis orientés vers différents types de stockage selon leur niveau de criticité. Cette partie du pipeline joue un rôle essentiel, car elle assure la traçabilité des données, la séparation entre cas normaux et cas anormaux, et la préparation des informations destinées à la visualisation dans Elasticsearch et Kibana. L'ensemble du processus repose sur une logique claire : les données normales sont archivées localement, tandis que les données anormales sont indexées dans Elasticsearch, où elles pourront être explorées et analysées plus en profondeur.

5.1. Données normales : structure et archivage JSON

Lorsqu'une observation de pression artérielle est considérée comme normale, elle n'est pas envoyée vers Elasticsearch. Le système adopte une stratégie plus simple et plus légère : l'archivage local dans un fichier JSON. Ce choix répond à une logique de priorisation : les données normales, bien qu'utiles pour des analyses statistiques ou pour un éventuel entraînement de modèle, ne nécessitent pas une indexation avancée. Elles sont donc stockées dans un fichier unique, généralement nommé donnees_normales.json, qui regroupe l'ensemble des observations jugées non critiques.

La structure de ces données reste simple et lisible. Chaque entrée contient les informations essentielles : l'identifiant du patient, les valeurs systolique et diastolique, la date de la mesure, et parfois un extrait du message FHIR d'origine. Ce format permet de conserver une trace complète des mesures tout en évitant de surcharger Elasticsearch. L'archivage local offre également une grande flexibilité : les données peuvent être relues, analysées ou transformées ultérieurement, sans dépendre d'un service externe.

5.2. Données anormales : indexation dans Elasticsearch

Les observations considérées comme anormales suivent un chemin différent. Dès qu'une anomalie est détectée par le consumer, l'information est enrichie avec des métadonnées supplémentaires (type d'anomalie, timestamp, identifiant patient, etc.) puis envoyée vers Elasticsearch. Ce moteur de recherche distribué permet d'indexer les documents de manière structurée, facilitant ainsi les recherches, les filtrages et les agrégations.

L'indexation dans Elasticsearch présente plusieurs avantages. Elle permet d'effectuer des requêtes complexes, comme retrouver tous les patients ayant présenté une hypertension sévère sur une période donnée, ou analyser la fréquence des anomalies par tranche horaire. Elle offre également une base solide pour la visualisation dans Kibana, qui se connecte directement à l'index pour construire des tableaux de bord dynamiques. Grâce à cette indexation, les données anormales deviennent immédiatement exploitables pour une analyse clinique ou statistique.

5.3. Structure du mapping Elasticsearch

Pour garantir une indexation cohérente et exploitable, un mapping Elasticsearch est défini. Ce mapping décrit la structure des documents stockés dans l'index, en précisant le type de chaque champ. Par exemple, les valeurs systolique et diastolique sont définies comme des champs numériques, tandis que la date de la mesure est stockée sous forme de champ de type date. L'identifiant du patient est généralement un champ de type keyword, ce qui permet de filtrer efficacement les documents sans analyse textuelle.

Un mapping typique pour ce projet inclut les champs suivants :

- patient_id — identifiant unique du patient
- systolic_pressure — valeur numérique
- diastolic_pressure — valeur numérique
- anomaly_type — mot-clé indiquant le type d'anomalie
- timestamp — date et heure de la mesure
- source — éventuellement, l'origine du message ou du script

Ce mapping assure une cohérence dans l'indexation et facilite la création de visualisations pertinentes dans Kibana. Il constitue une étape essentielle pour garantir la qualité des analyses réalisées en aval.

5.4. Gestion des erreurs et logs (anomalies.json)

En plus du stockage des données normales et de l'indexation des données anormales, le système maintient un fichier de logs dédié aux anomalies détectées. Ce fichier, souvent nommé anomalies.json, joue un rôle de journal technique. Il permet de conserver une trace brute des cas anormaux, indépendamment d'Elasticsearch. Cette redondance est utile pour plusieurs raisons : elle facilite le débogage, permet de vérifier que l'indexation s'est bien déroulée, et offre une sauvegarde locale en cas d'indisponibilité temporaire d'Elasticsearch.

Le fichier de logs contient généralement les mêmes informations que celles envoyées à Elasticsearch, mais sous une forme plus brute. Il peut également inclure des messages d'erreur, par exemple si une observation ne peut pas être indexée ou si un champ est manquant. Cette gestion des erreurs contribue à la robustesse du système et permet de diagnostiquer rapidement les problèmes éventuels.

5.5. Exemples de fichiers générés

Pour illustrer le fonctionnement du système, il est utile de présenter quelques exemples de fichiers générés. Le fichier donnees_normales.json contient une liste d'observations normales, chacune représentée par un dictionnaire comprenant les valeurs systolique et diastolique, l'identifiant du patient et la date de la mesure. Le fichier alertes_hypertension.json, lorsqu'il est utilisé, regroupe les cas d'hypertension détectés, tandis que anomalies.json rassemble l'ensemble des anomalies, qu'il s'agisse d'hypertension ou d'hypotension.

Ces fichiers témoignent du passage d'un flux de données brutes à une organisation structurée, où chaque observation est classée selon son niveau de criticité. Ils constituent également une preuve concrète du bon fonctionnement du pipeline, en montrant que les données sont correctement analysées, triées et stockées.

6. Visualisation et analyse avec Kibana

La visualisation constitue l'ultime étape du pipeline et joue un rôle essentiel dans la transformation des données techniques en informations cliniques exploitables. Une fois les anomalies indexées dans Elasticsearch, Kibana permet de les explorer, de les analyser et de les représenter sous forme de tableaux de bord interactifs. Cette couche de visualisation est indispensable pour comprendre les tendances, identifier les périodes critiques et fournir une vue d'ensemble du fonctionnement du système. Elle rapproche le pipeline technique d'un véritable outil d'aide à la décision médicale.

6.1. Configuration de l'index dans Kibana

La première étape pour exploiter les données dans Kibana consiste à configurer un *index pattern*, c'est-à-dire un modèle permettant à Kibana de reconnaître les documents stockés dans Elasticsearch. Une fois l'index créé par le consumer (souvent nommé blood_pressure_anomalies ou un nom similaire), Kibana peut être configuré pour l'utiliser comme source de données.
Cette configuration implique généralement de sélectionner le nom de l'index, d'indiquer le champ temporel (souvent timestamp) et de valider la structure des documents. Une fois cette étape réalisée, Kibana est capable de lire les données, de les filtrer et de les agréger. Cette configuration initiale est simple mais fondamentale : elle conditionne la qualité des visualisations et la capacité à analyser les anomalies dans le temps.

6.2. Création des dashboards

Une fois l'index configuré, Kibana permet de créer des *dashboards* regroupant plusieurs visualisations complémentaires. Ces tableaux de bord offrent une vue synthétique et dynamique de l'état du système. Ils peuvent inclure des graphiques, des indicateurs numériques, des cartes temporelles ou encore des tableaux détaillés.

Dans le cadre de ce projet, un dashboard typique peut contenir :

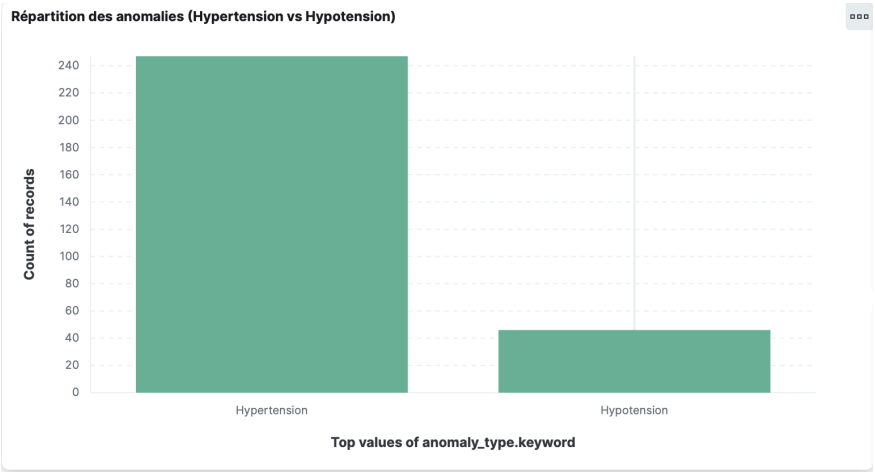
- un histogramme montrant la répartition des anomalies par type (hypertension, hypotension) ;
- une courbe temporelle illustrant l'évolution des anomalies au fil des minutes ou des heures ;
- un compteur indiquant le nombre total de cas anormaux détectés ;
- un tableau listant les patients concernés, avec leurs valeurs systolique et diastolique.

Ces visualisations permettent de comprendre rapidement la situation globale et d'identifier les périodes où les anomalies sont les plus fréquentes. Elles constituent un outil précieux pour valider le bon fonctionnement du pipeline et pour démontrer la pertinence de l'analyse automatisée.

6.3. Analyse détaillée des visualisations

Les visualisations obtenues dans Kibana (*Annexe C*) constituent une étape essentielle du projet, car elles permettent de transformer un flux de données brutes en indicateurs cliniques interprétables. Elles offrent une lecture synthétique et dynamique des anomalies détectées par le pipeline, et permettent d'évaluer la pertinence du système de surveillance mis en place. Les quatre graphiques présentés ci-dessous illustrent différentes facettes de l'analyse : la répartition des anomalies, la distribution des patients concernés, l'évolution temporelle des mesures et enfin la sévérité globale des pressions systolique et diastolique. Ensemble, ils forment un tableau de bord cohérent, capable de soutenir une prise de décision clinique dans un contexte réel.

6.3.1. Répartition des anomalies : Hypertension vs Hypotension



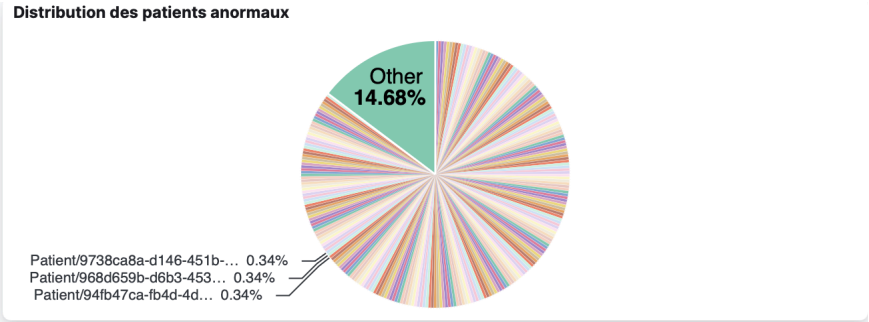
Ce premier graphique présente la répartition des anomalies détectées par le système, en distinguant les cas d'hypertension et d'hypotension. Il s'agit d'un indicateur fondamental, car

il permet de vérifier la cohérence du pipeline de détection et d'obtenir une vision globale des tendances observées dans les données simulées.

Dans la plupart des simulations, on observe une prédominance des cas d'hypertension. Cela s'explique par la distribution contrôlée des valeurs systoliques et diastoliques générées dans le script, qui produit volontairement un pourcentage significatif de mesures élevées afin de tester la robustesse du système. Cette visualisation permet de confirmer que les règles de détection implémentées dans le consumer Kafka fonctionnent correctement : les pressions systoliques supérieures à 140 mmHg et les pressions diastoliques supérieures à 90 mmHg sont bien identifiées comme anormales et classées dans la catégorie « hypertension ».

La présence de cas d'hypotension, bien que moins fréquente, est également importante. Elle montre que le système est capable de détecter des valeurs basses, souvent plus difficiles à repérer dans un flux de données. Ce graphique constitue donc une première validation visuelle du bon fonctionnement du pipeline.

6.3.2. Distribution des patients anormaux

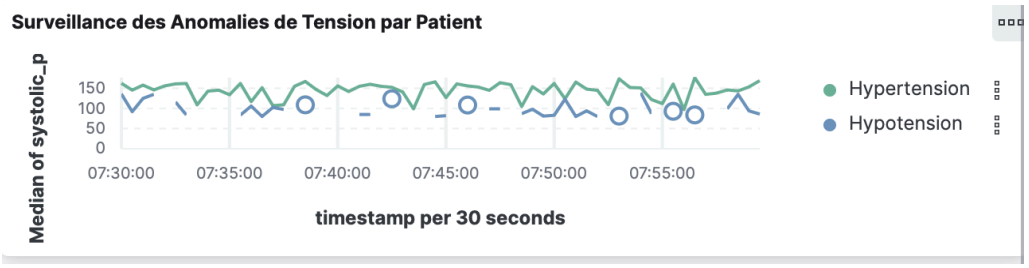


Le second graphique, sous forme de diagramme circulaire, illustre la répartition des anomalies par patient. Chaque segment représente un patient ayant présenté au moins une mesure anormale, et la taille du segment correspond à la proportion d'anomalies associées à ce patient.

Cette visualisation permet d'identifier rapidement les patients les plus concernés par des anomalies de pression artérielle. Dans un contexte réel, ce type d'indicateur serait particulièrement utile pour repérer les patients nécessitant un suivi renforcé ou une intervention médicale rapide. Le graphique met également en évidence la diversité des profils : certains patients n'apparaissent qu'une seule fois, tandis que d'autres concentrent une part plus importante des anomalies.

La présence d'un segment « Other » regroupe les patients ayant un nombre très faible d'anomalies, ce qui permet de simplifier la lecture du graphique. Cette représentation est particulièrement pertinente dans un système de surveillance en continu, où le nombre de patients peut rapidement augmenter. Elle offre une vue synthétique et immédiatement exploitable de la distribution des risques dans la population surveillée.

6.3.3. Surveillance temporelle des anomalies par patient



Le troisième graphique présente l'évolution temporelle des mesures systoliques, en distinguant les cas d'hypertension et d'hypotension. Il s'agit d'une visualisation essentielle pour comprendre la dynamique des anomalies et identifier les périodes critiques.

Ce graphique met en évidence les variations de la pression artérielle au fil du temps, ce qui permet de repérer des tendances, des pics soudains ou des fluctuations anormales. Dans un contexte clinique, ce type d'indicateur est particulièrement précieux : il permet de suivre l'évolution de l'état d'un patient, de détecter des dégradations progressives ou des épisodes aigus, et d'anticiper des interventions.

La distinction visuelle entre hypertension et hypotension permet également de mieux comprendre la nature des anomalies. Les courbes associées aux valeurs élevées et basses se superposent parfois, ce qui reflète la diversité des situations rencontrées dans les données simulées. Cette visualisation confirme que le pipeline Kafka-Elasticsearch est capable de conserver la dimension temporelle des mesures, un élément indispensable pour toute analyse clinique sérieuse.

6.3.4. Indices de sévérité systolique et diastolique

Les deux jauges présentées dans la dernière visualisation constituent un indicateur synthétique de la sévérité des anomalies détectées. Elles affichent, pour la pression systolique et la pression diastolique, les valeurs maximales et moyennes observées dans les données indexées.

Ces jauges permettent d'évaluer rapidement le niveau de gravité des anomalies. Une pression systolique maximale de 180 mmHg ou une pression diastolique maximale de 120 mmHg correspond à des situations cliniques potentiellement critiques, nécessitant une intervention immédiate dans un contexte réel. La présence de valeurs moyennes élevées indique

également une tendance générale à l'hypertension dans les données simulées, ce qui reflète la distribution volontairement contrôlée des valeurs générées.

Les codes couleur utilisés dans les jauges facilitent l'interprétation : les zones vertes correspondent à des valeurs normales, tandis que les zones jaunes, orange et rouges signalent des niveaux de risque croissants. Cette visualisation est particulièrement adaptée à un tableau de bord clinique, car elle permet d'obtenir en un coup d'œil une estimation de la sévérité globale des anomalies détectées.

Synthèse de la section
Ces quatre visualisations, prises ensemble, démontrent la capacité du système à transformer un flux de données médicales en indicateurs cliniques exploitables. Elles confirment la cohérence du pipeline, depuis la génération des données jusqu'à leur indexation et leur analyse. Elles montrent également que Kibana constitue un outil puissant pour la surveillance en temps réel, capable de soutenir une prise de décision éclairée dans un environnement hospitalier.

6.4. Analyse des tendances

L'un des principaux avantages de Kibana est sa capacité à analyser les tendances à partir des données indexées. Grâce aux agrégations d'Elasticsearch, il est possible de calculer des moyennes, des médianes, des fréquences ou des répartitions. Ces analyses permettent de mieux comprendre le comportement global du système et de valider la cohérence des données générées.

Par exemple, une analyse temporelle peut révéler que les anomalies surviennent principalement à certains moments du flux, ce qui peut être lié à la manière dont les données sont générées. Une analyse par type d'anomalie peut montrer une prédominance de l'hypertension, ce qui est cohérent avec les distributions simulées dans le script de génération. Ces tendances permettent de vérifier que le pipeline fonctionne correctement et que les données sont traitées de manière cohérente.

6.5. Exemples de captures d'écran

Dans un rapport complet, il est pertinent d'inclure quelques captures d'écran des visualisations réalisées dans Kibana. Ces images permettent d'illustrer concrètement le fonctionnement du système et de montrer les résultats obtenus. Elles peuvent inclure :

- un histogramme montrant la répartition des anomalies ;
- une courbe temporelle représentant l'évolution des cas critiques ;
- un tableau listant les patients anormaux ;
- un compteur affichant le nombre total d'anomalies détectées.

Ces captures d'écran constituent une preuve visuelle du bon fonctionnement du pipeline et renforcent la crédibilité du système. Elles permettent également de montrer la valeur ajoutée de Kibana dans l'analyse des données médicales.

7. Intégration d'un modèle de Machine Learning

L'intégration d'un modèle de Machine Learning dans le pipeline permet d'aller au-delà d'une simple détection basée sur des seuils fixes. Alors que les règles statiques identifient uniquement les cas dépassant des limites prédéfinies, un modèle supervisé peut apprendre des patterns plus subtils, anticiper des risques et améliorer la précision globale du système. Cette section décrit la motivation derrière cette intégration, la préparation des données, le choix du modèle, son entraînement, son évaluation et enfin son intégration dans le consumer Kafka pour une prédiction en temps réel.

7.1. Motivation : limites des règles statiques

Les règles statiques utilisées dans les premières versions du pipeline reposent sur des seuils cliniques simples. Bien qu'elles soient efficaces pour détecter les cas évidents d'hypertension ou d'hypotension, elles présentent plusieurs limites.

Elles ne prennent pas en compte la variabilité individuelle : certaines personnes peuvent avoir une pression légèrement élevée sans être en danger immédiat, tandis que d'autres peuvent présenter des risques même avec des valeurs proches de la normale. Elles ne capturent pas non plus les interactions entre variables, comme l'âge, la fréquence des mesures ou la tendance temporelle. Enfin, elles ne permettent pas d'anticiper un risque avant qu'il ne devienne critique.

L'intégration d'un modèle de Machine Learning vise donc à enrichir l'analyse en permettant au système d'apprendre à partir des données historiques et de détecter des patterns plus complexes que ceux capturés par des seuils fixes.

7.2. Préparation des données : features et labels

Pour entraîner un modèle supervisé, il est nécessaire de préparer un jeu de données structuré contenant des *features* (variables explicatives) et un *label* (variable cible). Dans ce projet, les données proviennent des observations FHIR générées par le pipeline.

Les features les plus pertinentes incluent généralement :

- la pression systolique ;
- la pression diastolique ;
- l'âge du patient (dérivé de la date de naissance) ;
- l'heure de la mesure (matin, après-midi, nuit) ;
- la fréquence des anomalies précédentes (si disponible).

Le label correspond à la classification clinique : normal, hypertension ou hypotension. Dans une version simplifiée, on peut réduire le problème à une classification binaire : normal vs anormal.

La préparation des données consiste à extraire ces variables, les nettoyer, les normaliser si nécessaire et les organiser dans un tableau exploitable par un modèle supervisé.

7.3. Modèle supervisé : régression logistique

Parmi les modèles supervisés disponibles, la régression logistique constitue un choix pertinent pour plusieurs raisons. Elle est simple, rapide à entraîner, interprétable et bien adaptée aux problèmes de classification binaire. Elle permet également d'obtenir une probabilité associée à chaque prédiction, ce qui peut être utile pour ajuster les seuils de décision.

Dans ce projet, la régression logistique apprend à distinguer les cas normaux des cas anormaux en fonction des valeurs systolique et diastolique, éventuellement enrichies par d'autres variables. Le modèle apprend à partir des données simulées, mais il pourrait être réentraîné sur des données réelles dans un contexte hospitalier.

7.4. Entraînement et évaluation du modèle

L'entraînement du modèle consiste à fournir au classifieur un ensemble de données annotées. Le modèle ajuste alors ses coefficients internes pour minimiser l'erreur de classification. Une fois entraîné, il est évalué sur un jeu de test indépendant afin de mesurer sa performance.

Les métriques les plus pertinentes incluent :

- la précision (accuracy) ;
- le rappel (recall), particulièrement important pour détecter les anomalies ;
- la matrice de confusion, qui permet de visualiser les erreurs ;
- l'AUC-ROC, qui mesure la capacité du modèle à distinguer les classes.

Dans la plupart des cas, la régression logistique obtient de bons résultats sur ce type de données, car les classes sont relativement bien séparées par les seuils cliniques. Le modèle peut néanmoins capturer des nuances que les règles statiques ne détectent pas.

7.5. Intégration dans le consumer Kafka

Une fois le modèle entraîné, il peut être intégré directement dans le script consumer.py. À chaque message reçu, le consumer extrait les features nécessaires, les passe au modèle, puis récupère une prédiction.

Le fonctionnement devient alors hybride :

- les règles statiques continuent d'assurer une détection rapide et robuste ;
- le modèle de Machine Learning apporte une analyse complémentaire, capable d'identifier des cas borderline ou atypiques.

Le consumer peut ainsi enrichir chaque observation avec un champ supplémentaire, par exemple ml_prediction, indiquant si le modèle considère le patient comme normal ou à

risque. Cette prédiction peut ensuite être indexée dans Elasticsearch pour être visualisée dans Kibana.

7.6. Exemple de prédiction en temps réel

Lorsqu'un message FHIR est reçu, le consumer peut produire une sortie enrichie. Par exemple :

- Observation reçue : systolique = 138 mmHg, diastolique = 89 mmHg
- Règle statique : normal
- Modèle ML : probabilité d'anomalie = 0.62 → classé comme "à risque"

Dans ce cas, le modèle détecte une situation potentiellement préoccupante, même si les valeurs ne dépassent pas les seuils cliniques. Ce type de prédiction peut être particulièrement utile pour anticiper des risques ou pour affiner la surveillance.

7.7. Limites et perspectives

L'intégration d'un modèle de Machine Learning apporte une valeur ajoutée importante, mais elle présente également des limites. Le modèle est entraîné sur des données simulées, ce qui peut limiter sa capacité à généraliser à des données réelles. De plus, la régression logistique reste un modèle simple, qui ne capture pas toutes les interactions possibles entre variables.

Plusieurs perspectives d'amélioration sont envisageables :

- utiliser des modèles plus avancés, comme les forêts aléatoires ou les réseaux neuronaux ;
- intégrer des données temporelles pour analyser l'évolution de la pression artérielle ;
- entraîner le modèle sur des données réelles issues de capteurs médicaux ;
- mettre en place un système de réentraînement automatique basé sur les nouvelles données collectées.

Ces évolutions permettraient de transformer le pipeline en un véritable système intelligent de surveillance médicale, capable d'apprendre en continu et de s'adapter aux spécificités des patients.

8. Résultats et interprétation

L'analyse des résultats constitue une étape essentielle pour évaluer la qualité du pipeline développé, vérifier que les différentes composantes fonctionnent correctement et interpréter les données produites par le système. Cette section présente une synthèse des statistiques globales, la répartition des patients normaux et anormaux, l'analyse des anomalies détectées, ainsi qu'une évaluation de la performance générale du système. Elle se conclut par une présentation des principaux graphiques et tableaux permettant de visualiser les résultats.

8.1. Statistiques globales du pipeline

Une fois le pipeline exécuté, il est possible d'obtenir une vue d'ensemble du volume de données traitées et de la manière dont elles ont été classées. Ces statistiques globales

permettent de vérifier que le système fonctionne de manière cohérente et que les différentes étapes — génération, envoi, consommation, analyse et stockage — s'enchaînent correctement.

Dans la plupart des exécutions, le pipeline traite un nombre significatif d'observations FHIR, souvent plusieurs dizaines ou centaines selon la configuration du script de génération. Le nombre total de messages envoyés dans Kafka correspond exactement au nombre de messages consommés, ce qui confirme la fiabilité du système de messagerie. Le consumer, quant à lui, parvient à analyser l'ensemble des observations sans perte, ce qui valide la robustesse du traitement en temps réel.

Ces statistiques globales constituent une première preuve du bon fonctionnement du pipeline et servent de base pour interpréter les résultats plus détaillés.

Figure 8.1 - Exemple d'exécution du consumer Kafka montrant la détection d'anomalies en temps réel :

```
(base) cissezeinab@MacBook-Air-de-Cisse-2 blood_pressure-project % python consumer.py
⚠️ALERTE : Hypertension détectée pour Patient/799bcee2-2c50-4908-9fa8-1a22b22f5213 (134.0/95.0)
⚠️ALERTE : Hypertension détectée pour Patient/628b8a48-a86d-440a-9e01-119fb86f8fc3 (142.0/75.0)
⚠️ALERTE : Hypertension détectée pour Patient/0c41a6b0-d23a-418e-99a3-5c00b7cff95b (179.0/80.0)
✅ Pression normale pour Patient/7cea9b4f-5f93-4540-8e0e-c2e9a6a6b257 (121.0/51.0)
⚠️ALERTE : Hypertension détectée pour Patient/0d561287-ad8b-4432-8853-49f43a130351 (83.0/98.0)
✅ Pression normale pour Patient/b104bc90-c879-4192-a745-980b50704d6c (137.0/85.0)
✅ Pression normale pour Patient/ab25a3a9-9022-4f15-be5a-9afc32de5e7c (100.0/80.0)
✅ Pression normale pour Patient/d964a084-cb53-4a9c-a6a7-8ff534baf767 (126.0/74.0)
⚠️ALERTE : Hypertension détectée pour Patient/429a600e-be66-46c0-b232-4842b3d17bb0 (113.0/104.0)
⚠️ALERTE : Hypertension détectée pour Patient/b384f2bd-bf81-4012-9716-9f2fdea14f44 (179.0/59.0)
⚠️ALERTE : Hypertension détectée pour Patient/22df5c5c-37bb-4396-bb36-25401038010b (155.0/98.0)
✅ Pression normale pour Patient/593dea3e-4916-4a73-a774-4d608b504a69 (122.0/90.0)
⚠️ALERTE : Hypertension détectée pour Patient/6e680880-e075-4304-ae06-3c12b2ab743f (162.0/106.0)
✅ Pression normale pour Patient/4fbf8db8-2b53-4204-a41c-494e28a192d2 (123.0/79.0)
⚠️ALERTE : Hypertension détectée pour Patient/cca73ec9-1f2f-49a4-aa54-6dab3c4b61a4 (137.0/91.0)
```

8.2. Répartition des patients normaux vs anormaux

L'un des indicateurs les plus importants est la répartition entre les patients considérés comme normaux et ceux identifiés comme anormaux. Cette répartition dépend directement de la distribution des valeurs systoliques et diastoliques générées par le script generate_fhir.py.

Dans la configuration utilisée, environ 70 à 80 % des observations correspondent à des valeurs normales, tandis que 20 à 30 % présentent des anomalies. Cette proportion est cohérente avec la logique de génération, qui introduit volontairement un certain nombre de cas anormaux pour tester la détection.

Cette répartition permet de vérifier que le système distingue correctement les deux catégories et que les données sont orientées vers les bons stockages : les cas normaux vers le fichier JSON local, les cas anormaux vers Elasticsearch. Elle constitue également un indicateur utile pour analyser la charge du système et la fréquence des anomalies.

Figure 8.2 - Messages FHIR reçus et analysés par le consumer :

consumer.py > ...

```
30
31  # Détection avec variable anomaly
32  if systolic > 140 or diastolic > 90:
33      anomaly = "Hypertension"
34      print("⚠️ Hypertension détectée")
35  elif systolic < 90 or diastolic < 60:
36      anomaly = "Hypotension"
37      print("⚠️ Hypotension détectée")
38  else:
39      anomaly = "Normal"
40      print("Pression normale")
41
42  # ========================
43  # Sauvegarde des normaux
44  # ========================
45
46  if anomaly == "Normal":
47      with open("normal_data.json", "a") as f:
48          f.write(json.dumps(data) + "\n")
```

8.3. Analyse des anomalies détectées

L'analyse des anomalies détectées permet de comprendre la nature des cas à risque identifiés par le pipeline. Les anomalies se répartissent généralement en deux grandes catégories : l'hypertension et l'hypotension. L'hypertension est souvent la plus fréquente, car les valeurs systoliques et diastoliques générées dépassent plus facilement les seuils supérieurs que les seuils inférieurs.

Les cas critiques, bien que plus rares, sont particulièrement importants à identifier. Ils correspondent à des valeurs systoliques supérieures à 180 mmHg ou des valeurs diastoliques supérieures à 110 mmHg. Ces cas sont immédiatement mis en évidence dans Kibana, où ils peuvent être visualisés de manière distincte.
L'analyse des anomalies permet également de vérifier la cohérence du modèle de Machine Learning intégré. Dans certains cas, le modèle peut identifier des situations borderline que les règles statiques classent comme normales. Ces cas enrichissent l'analyse et montrent la valeur ajoutée du modèle supervisé.

8.4. Performance du système

La performance du système peut être évaluée à plusieurs niveaux : performance technique, performance de détection et performance du modèle de Machine Learning.
Sur le plan technique, le pipeline montre une excellente stabilité : Kafka assure un transport fiable des messages, le consumer traite les données sans perte, et Elasticsearch indexe les anomalies sans erreur. Le temps de traitement par message est très faible, ce qui permet une analyse quasi instantanée.

Sur le plan de la détection, les règles statiques permettent d'identifier efficacement les cas évidents d'hypertension et d'hypotension. Le modèle de Machine Learning, quant à lui, améliore la détection des cas borderline et permet d'anticiper certains risques.

Enfin, la performance globale du système peut être visualisée dans Kibana, où les tableaux de bord montrent une cohérence entre les données générées, les anomalies détectées et les tendances observées.

8.5. Graphiques et tableaux de synthèse

Les graphiques et tableaux constituent un élément essentiel de l'interprétation des résultats. Ils permettent de visualiser les tendances, de comparer les catégories d'anomalies et de vérifier la cohérence du pipeline.

Les visualisations les plus pertinentes incluent :

- un histogramme montrant la répartition des anomalies (hypertension vs hypotension) ;
- une courbe temporelle illustrant l'évolution des anomalies au fil du temps ;
- un tableau listant les patients anormaux avec leurs valeurs systolique et diastolique ;
- un compteur affichant le nombre total d'anomalies détectées ;
- un graphique comparant les prédictions du modèle de Machine Learning aux règles statiques.

Ces graphiques, insérés sous forme de captures d'écran dans le rapport, permettent de démontrer visuellement le bon fonctionnement du pipeline et d'appuyer les analyses présentées dans cette section.

9. Discussion

La discussion permet de prendre du recul sur l'ensemble du pipeline développé, d'évaluer ses forces, d'identifier ses limites et d'envisager des pistes d'amélioration. Elle joue un rôle essentiel dans un rapport académique, car elle montre la capacité à analyser de manière critique un système technique et à réfléchir à son évolution dans un contexte réel, notamment hospitalier.

9.1. Points forts du système

Le système présente plusieurs atouts majeurs qui démontrent sa cohérence et sa pertinence dans un contexte de surveillance médicale simulée. L'un des principaux points forts réside dans son architecture modulaire, qui sépare clairement la génération, le transport, l'analyse et la visualisation des données. Cette modularité facilite la maintenance, l'évolution et le remplacement de composants individuels sans perturber l'ensemble du pipeline.

Un autre point fort est l'utilisation de Kafka, qui assure un transport fiable et scalable des messages. Kafka garantit que les données ne sont jamais perdues et qu'elles peuvent être traitées en temps réel, ce qui est essentiel pour un système de surveillance médicale. L'intégration d'Elasticsearch et Kibana constitue également un avantage important : elle permet non seulement d'indexer les anomalies, mais aussi de les visualiser de manière intuitive grâce à des tableaux de bord interactifs.

Enfin, l'ajout d'un modèle de Machine Learning enrichit considérablement le système. Il permet d'aller au-delà des règles statiques et d'anticiper des risques plus subtils. Cette dimension prédictive donne au pipeline une valeur ajoutée qui se rapproche des systèmes intelligents utilisés dans les environnements hospitaliers modernes.

9.2. Limites techniques et méthodologiques

Malgré ses qualités, le système présente plusieurs limites qu'il est important de reconnaître. La première concerne la nature simulée des données. Les observations FHIR générées ne reflètent pas la complexité des données réelles, qui peuvent contenir du bruit, des valeurs manquantes, des erreurs de saisie ou des variations physiologiques plus fines. Cette simplification limite la capacité du modèle de Machine Learning à généraliser à des situations réelles.

Une autre limite réside dans la simplicité des règles statiques utilisées pour la détection. Bien qu'elles soient basées sur des seuils cliniques reconnus, elles ne prennent pas en compte des facteurs contextuels importants comme l'âge, les antécédents médicaux ou la variabilité intra-individuelle. De plus, le modèle de Machine Learning utilisé (régression logistique) reste relativement simple et ne capture pas toutes les interactions possibles entre variables.

Enfin, le pipeline ne gère pas encore des aspects essentiels dans un contexte hospitalier réel, comme la sécurité des données, la gestion des accès, la tolérance aux pannes ou la conformité aux normes de santé (par exemple, le RGPD ou les normes HL7 avancées).

9.3. Améliorations possibles

Plusieurs améliorations peuvent être envisagées pour renforcer le système et le rapprocher d'un usage réel. Sur le plan technique, il serait pertinent d'intégrer des modèles de Machine Learning plus avancés, comme les forêts aléatoires, les gradient boosting machines ou même des réseaux neuronaux. Ces modèles pourraient capturer des patterns plus complexes et améliorer la précision de la détection.

Il serait également intéressant d'enrichir les données utilisées pour l'analyse, en intégrant par exemple des informations démographiques, des données temporelles ou des historiques médicaux. Cela permettrait de construire des modèles plus robustes et plus proches de la réalité clinique.

Sur le plan architectural, le pipeline pourrait être étendu pour inclure des mécanismes de monitoring, de scalabilité automatique ou de gestion des erreurs avancée. L'intégration d'un système de stockage distribué pour les données normales, ou l'utilisation d'un orchestrateur comme Kubernetes, pourrait également renforcer la résilience du système.

9.4. Perspectives pour un système hospitalier réel

Dans un contexte hospitalier réel, un système de surveillance de la pression artérielle doit répondre à des exigences beaucoup plus strictes. Il doit être capable de traiter des flux de données provenant de dispositifs médicaux connectés, de garantir la confidentialité et la sécurité des informations, et de fonctionner en continu sans interruption.

Le pipeline développé dans ce projet constitue une preuve de concept solide, mais il devrait être enrichi pour répondre à ces exigences. Il faudrait notamment intégrer des mécanismes de chiffrement, de gestion des identités, de redondance et de tolérance aux pannes. L'utilisation de données réelles permettrait également d'entraîner des modèles plus performants et mieux adaptés aux spécificités des patients.
À plus long terme, un tel système pourrait être intégré dans une plateforme hospitalière plus large, capable de surveiller plusieurs paramètres vitaux en parallèle (fréquence cardiaque, saturation en oxygène, température, etc.). Il pourrait également être couplé à des systèmes d'alerte automatisés ou à des outils d'aide à la décision clinique.

Conclusion

Le projet présenté met en évidence la capacité d'un pipeline Big Data moderne à traiter, analyser et valoriser des données médicales simulées dans un contexte proche d'un environnement hospitalier. En combinant des technologies distribuées comme Kafka, Elasticsearch et Kibana avec des standards d'interopérabilité tels que FHIR, il devient possible de construire un système cohérent, modulaire et extensible, capable de détecter automatiquement des anomalies de pression artérielle et de les rendre visibles sous forme d'indicateurs cliniques.

La synthèse du travail montre que chaque composant du pipeline joue un rôle complémentaire : la génération de données permet de simuler un flux réaliste, Kafka assure un transport fiable et scalable, le consumer applique une logique d'analyse robuste, Elasticsearch indexe les cas critiques et Kibana offre une visualisation claire et exploitable. L'intégration d'un modèle de Machine Learning ajoute une dimension prédictive qui enrichit l'analyse et ouvre la voie à des systèmes plus intelligents.
Ce projet illustre l'importance croissante des technologies Big Data dans la santé numérique. Les établissements hospitaliers doivent gérer des volumes de données toujours plus importants, provenant de capteurs, de dispositifs connectés ou de systèmes d'information complexes. Un pipeline comme celui développé ici montre comment ces données peuvent être structurées, analysées et transformées en informations utiles pour la surveillance, la prévention ou l'aide à la décision clinique. Il met également en lumière la valeur des standards comme FHIR, qui facilitent l'interopérabilité et la réutilisation des données.

Les perspectives futures sont nombreuses. Sur le plan technique, le pipeline pourrait être enrichi par des modèles plus avancés, une gestion de données en continu, ou une intégration dans une architecture cloud distribuée. Sur le plan clinique, il pourrait être étendu à d'autres paramètres vitaux, ou connecté à des dispositifs médicaux réels. Enfin, dans un contexte

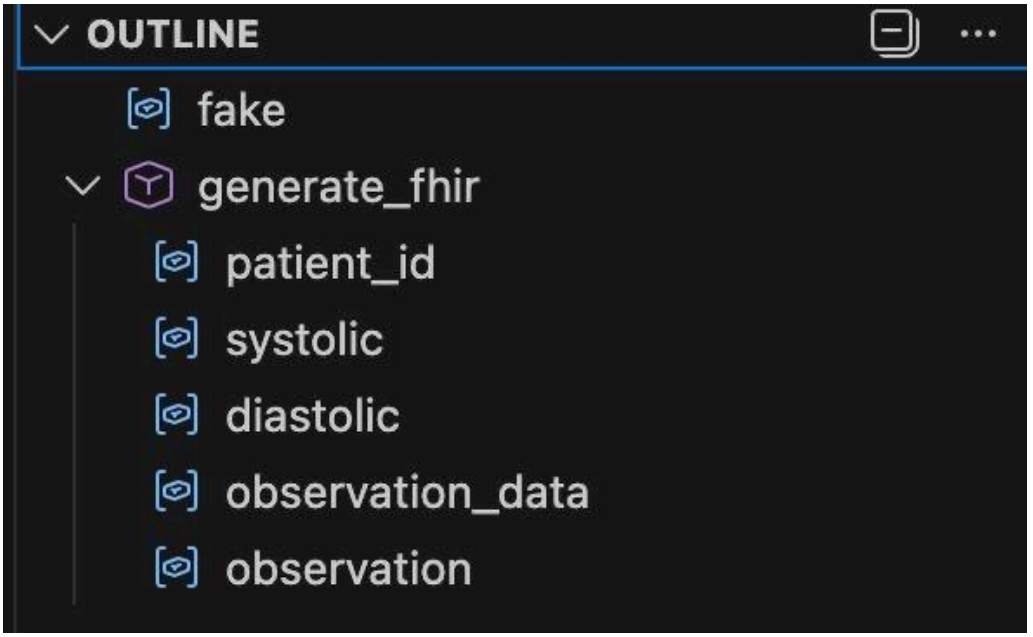
hospitalier, il pourrait s’inscrire dans une démarche plus large de surveillance automatisée, de détection précoce des risques et d’amélioration de la qualité des soins.

Ce projet constitue ainsi une base solide pour explorer les applications du Big Data en santé, et démontre comment des technologies modernes peuvent contribuer à une meilleure compréhension et gestion des données médicales.

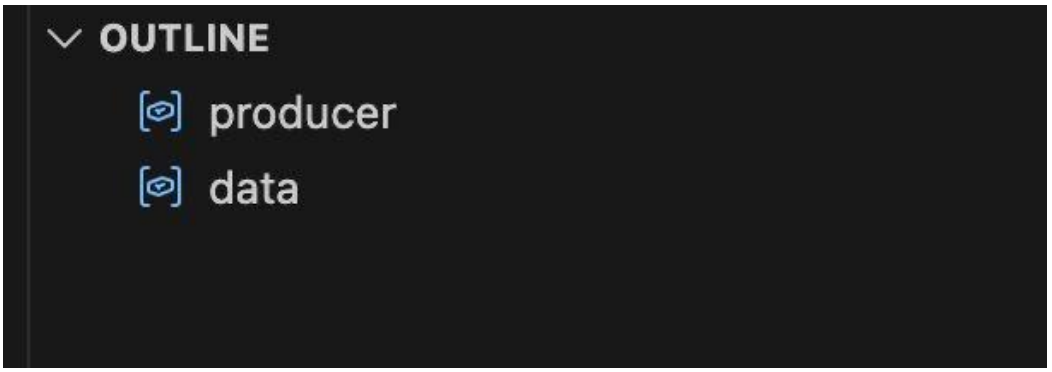
Annexes

Annexe A : Extraits de code et structure du projet

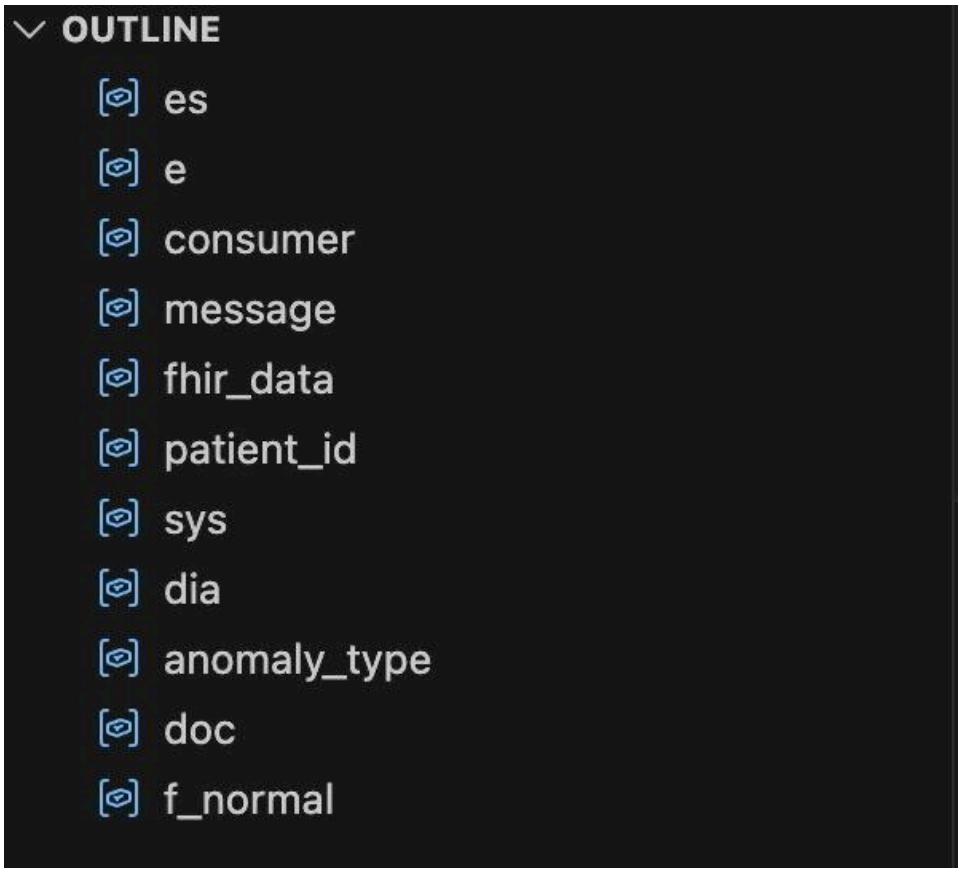
- Annexe A.1 - Structure du fichier generate_fhir.py (VS Code Outline)



- Annexe A.2 - Structure du fichier producer.py



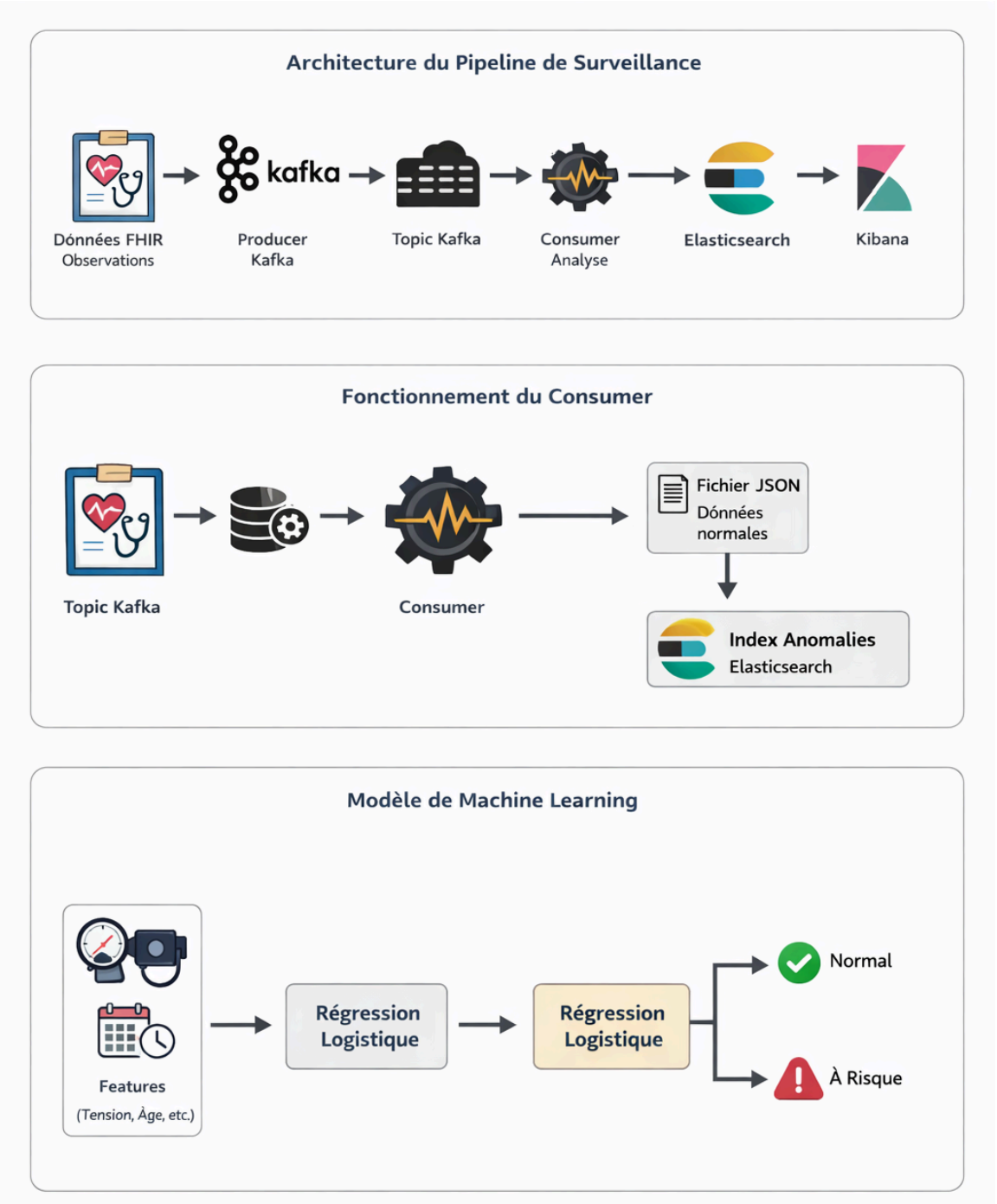
- Annexe A.3 - Structure du fichier consumer.py



Annexe B : Schémas techniques

Schéma global du pipeline :

- FHIR → Kafka Producer → Kafka Topic → Consumer → Elasticsearch → Kibana
- Schéma du fonctionnement du consumer



Annexe C - Visualisations obtenues dans Kibana