SYSTEME DE SURVEILLANCE DES DONNÉES DE PRESSION ARTERIELLE EN TEMPS REEL

Kafka - Elasticsearch - Kibana - Machine Learning

CHIAB Meriem et KACHOUR Amara

Contexte et objectif

01 Contexte

- Surveillance médicale continue
- Données de pression artérielle en temps réel
- Importance de la détection rapide des anomalies

02 Objectifs

- Mettre en place un pipeline temps réel
- Détecter automatiquement les anomalies
- Visualiser et alerter via Kibana

SOMMAIRE

01 Architecture global du système

Producer → Kafka → Consumer → Elasticsearch → Kibana

↳ Cas normaux stockés en local

02 Génération et streaming des données (Kafka)

```python
import json
import time
from datetime import datetime, timezone
from faker import Faker
from kafka import KafkaProducer

# Générateur de valeurs réalistes
fake = Faker()

# Kafka
KAFKA_BROKER = "localhost:9092"
TOPIC = "blood_pressure_fhir"

# 5 patients fixes
PATIENT_IDS = ["PAT-001", "PAT-002", "PAT-003", "PAT-004", "PAT-005"]

# 2 practitioners fixes (soignants)
PRACTITIONER_IDS = ["PRAC-01", "PRAC-02"]

# Producteur Kafka : envoie des dict Python encodés en JSON
producer = KafkaProducer(
    bootstrap_servers=KAFKA_BROKER,
    value_serializer=lambda v: json.dumps(v).encode("utf-8"),
)

def now_iso():
    """Renvoie la date/heure actuelle en ISO 8601 (UTC)."""
    return datetime.now(timezone.utc).isoformat()
```

```python
def build_fhir_observation(patient_id: str, practitioner_id: str) -> dict:
    """
    Crée une Observation FHIR (format simple).
    On inclut :
    - subject : Patient/...
    - performer : Practitioner/...
    - component : systolic + diastolic
    """
    systolic = fake.random_int(min=80, max=190)
    diastolic = fake.random_int(min=50, max=120)

    return {
        "resourceType": "Observation",
        "status": "final",
        "subject": {"reference": f"Patient/{patient_id}"},
        "performer": [{"reference": f"Practitioner/{practitioner_id}"}],
        "effectiveDateTime": now_iso(),
        "component": [
            {
                "code": {"text": "Systolic Blood Pressure"},
                "valueQuantity": {"value": systolic, "unit": "mmHg"},
            },
            {
                "code": {"text": "Diastolic Blood Pressure"},
                "valueQuantity": {"value": diastolic, "unit": "mmHg"},
            },
        ],
    }
```

```python
if __name__ == "__main__":
    print("Producer started (5 patients / 2 practitioners / every 5 seconds)")
    idx = 0

    # Boucle infinie : toutes les 5 secondes, on envoie 5 messages (1 par patient)
    while True:
        for patient_id in PATIENT_IDS:
            # On alterne PRAC-01 / PRAC-02
            practitioner_id = PRACTITIONER_IDS[idx % len(PRACTITIONER_IDS)]
            idx += 1

            obs = build_fhir_observation(patient_id, practitioner_id)

            # Envoi vers Kafka
            producer.send(TOPIC, obs)
            print(f"Sent -> patient={patient_id} practitioner={practitioner_id}")

        # On force l'envoi immédiat du batch
        producer.flush()

        # Pause de 5 secondes
        time.sleep(5)
```

03 Traitement des données et détection des anomalies

```python
# 2) OUTILS : EXTRACTION
#

def extract_fields(obs: dict):
    """
    Extrait les champs utiles depuis une Observation FHIR (ton format producer).
    - patient_id : subject.reference = "Patient/<id>"
    - practitioner_id : performer[0].reference = "Practitioner/<id>" (si présent)
    - systolic / diastolic : component[0] / component[1]
    - timestamp : effectiveDateTime
    """
    # patient
    patient_ref = obs.get("subject", {}).get("reference", "Patient/unknown")
    patient_id = patient_ref.split("/")[-1]

    # practitioner
    performer = obs.get("performer", [])
    if performer and isinstance(performer, list):
        prac_ref = performer[0].get("reference", "Practitioner/unknown")
        practitioner_id = prac_ref.split("/")[-1]
    else:
        practitioner_id = "unknown"

    # pression
    components = obs.get("component", [])
    systolic = int(components[0]["valueQuantity"]["value"])
    diastolic = int(components[1]["valueQuantity"]["value"])

    # temps
    ts = obs.get("effectiveDateTime")

    return patient_id, practitioner_id, systolic, diastolic, ts
```

```python
# 3) OUTILS : RÈGLES + ML
#

def classify_rules(sys_val: int, dia_val: int):
    """
    Règles simples (seuils du sujet):
    - systolique > 140 ou < 90
    - diastolique > 90 ou < 60

    Retour:
    - is_anomaly : bool
    - anomaly_type : "hypertension" / "hypotension" / "normal"
    - details : liste (tags précis)
    """
    details = []

    if sys_val > 140:
        details.append("hypertension_systolic")
    if sys_val < 90:
        details.append("hypotension_systolic")
    if dia_val > 90:
        details.append("hypertension_diastolic")
    if dia_val < 60:
        details.append("hypotension_diastolic")

    is_anomaly = len(details) > 0

    if not is_anomaly:
        anomaly_type = "normal"
    else:
        anomaly_type = "hypertension" if any("hyper" in d for d in details) else "hypotension"

    return is_anomaly, anomaly_type, details
```

```python
def predict_risk_ml(sys_val: int, dia_val: int):
    """
    Prédiction ML très simple :
    - pred : 0 (normal) ou 1 (anormal)
    - risk : probabilité d'anomalie (entre 0 et 1)
    """
    X = [[sys_val, dia_val]]
    pred = int(model.predict(X)[0])
    risk = float(model.predict_proba(X)[0][1])
    return pred, risk
```

04 Intégration du modèle de Machine Learning

```python
# train_ml.py
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
import joblib

np.random.seed(42)

# 1) dataset simple : sys + dia
N = 5000
sys = np.random.normal(loc=120, scale=20, size=N)
dia = np.random.normal(loc=80, scale=15, size=N)

# 2) label = anomalie selon seuils cliniques
y = ((sys >= 140) | (sys <= 90) | (dia >= 90) | (dia <= 60)).astype(int)

# 3) X = features
X = np.column_stack([sys, dia])

# 4) split train/test
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)
```

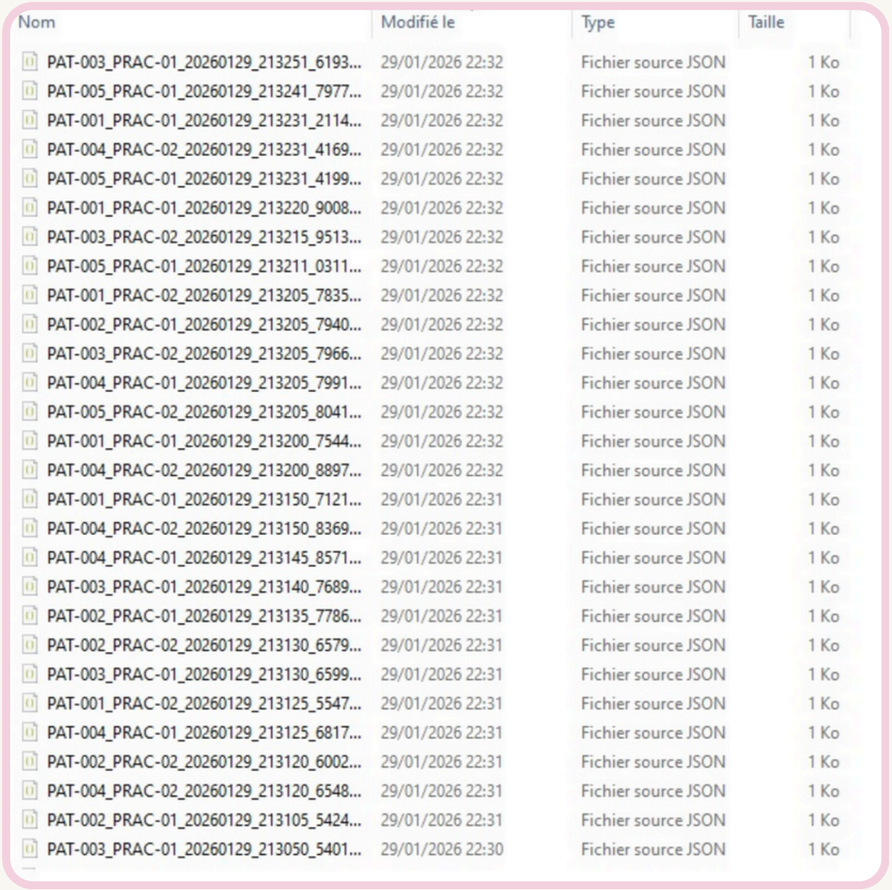
```python
# 5) modèle simple
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)

# 6) évaluation rapide
pred = model.predict(X_test)
print(classification_report(y_test, pred))

# 7) sauvegarde
joblib.dump(model, "bp_logreg.joblib")
print(" Modèle sauvegardé : bp_logreg.joblib")
```

05 Stockage et indexation des données

Stockage différencié :

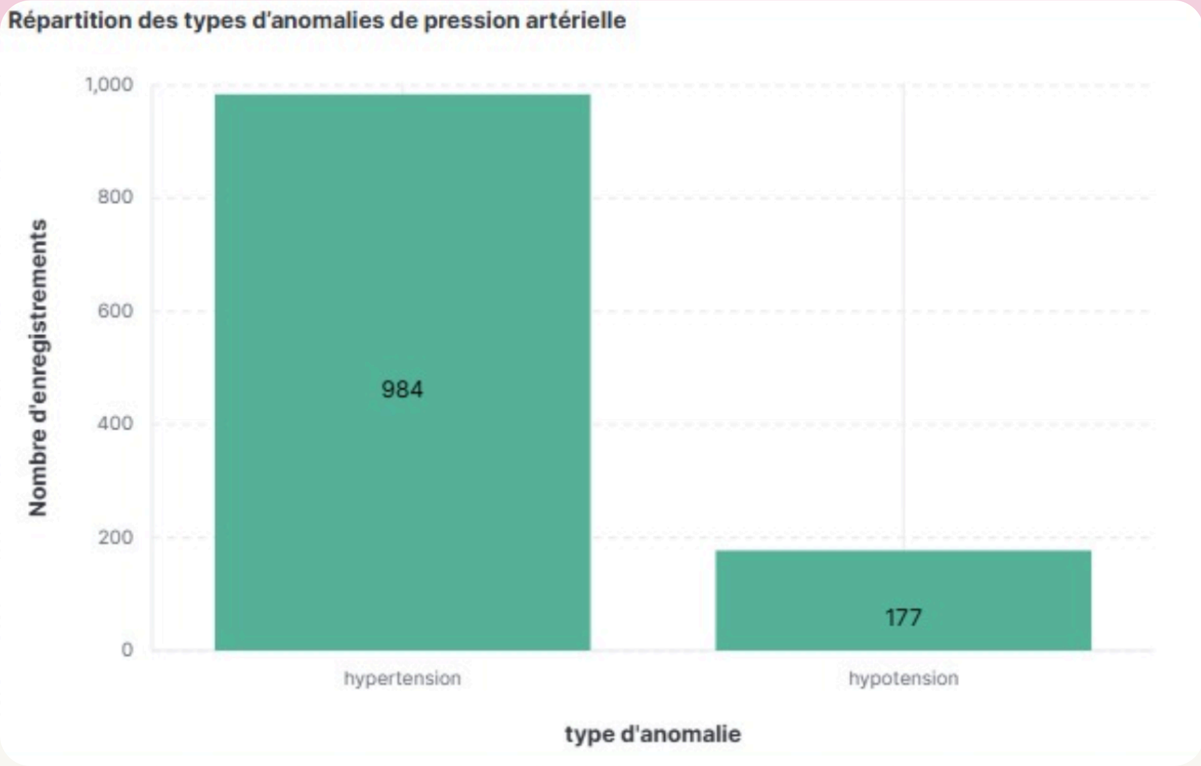
Anomalies → Elasticsearch

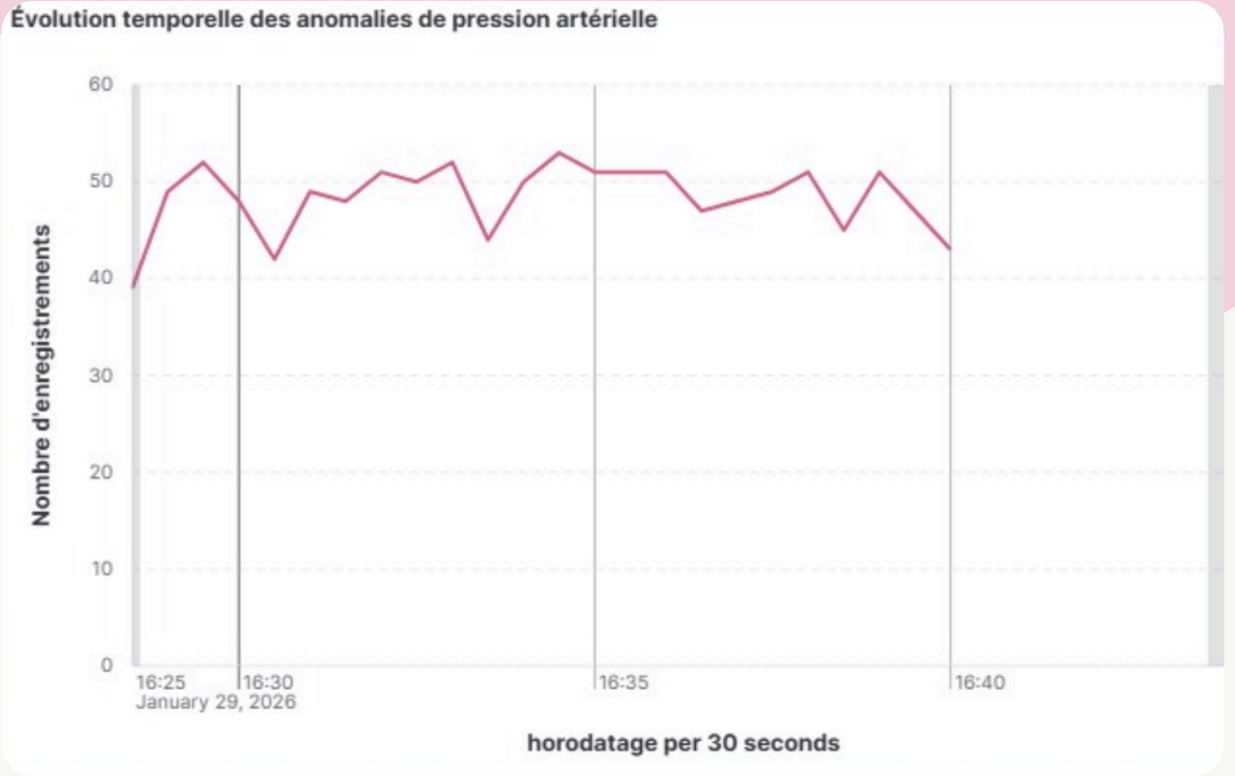
Données normales → fichiers JSON locaux



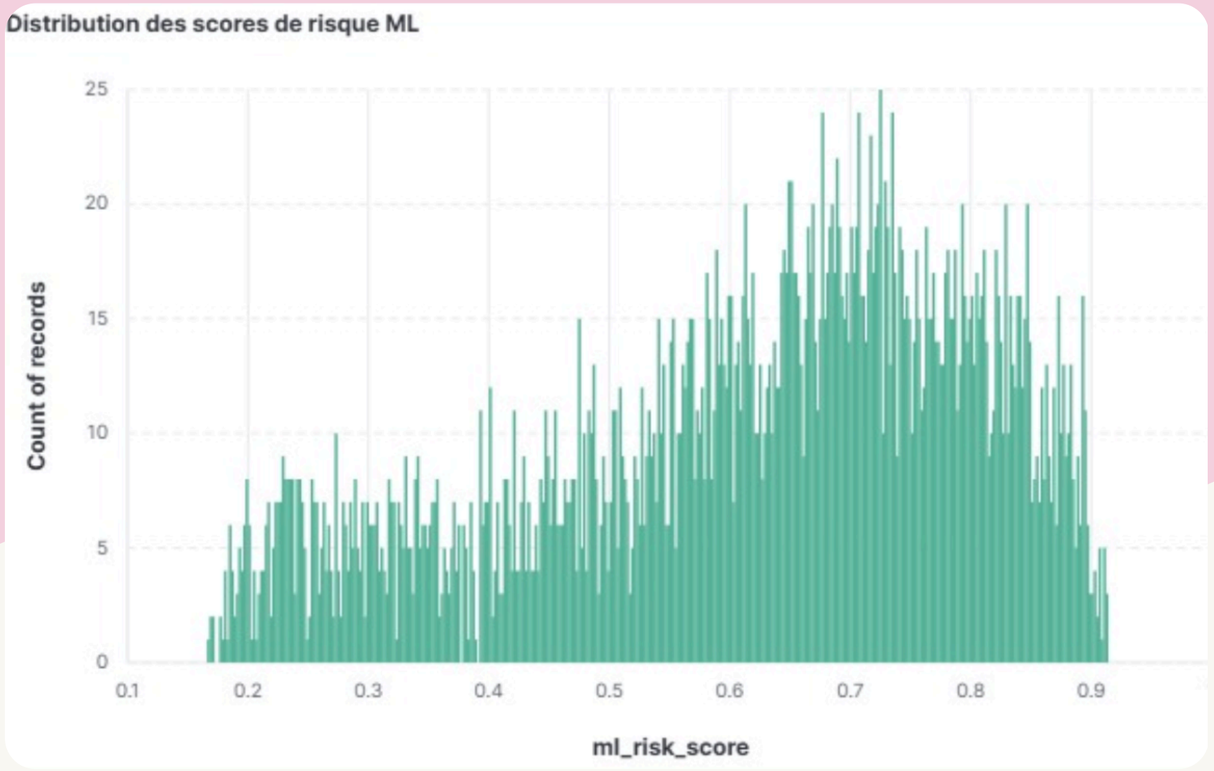
Nom	Modifié le	Type	Taille
PAT-003_PRAC-01_20260129_213251_6193...	29/01/2026 22:32	Fichier source JSON	1 Ko
PAT-005_PRAC-01_20260129_213241_7977...	29/01/2026 22:32	Fichier source JSON	1 Ko
PAT-001_PRAC-01_20260129_213231_2114...	29/01/2026 22:32	Fichier source JSON	1 Ko
PAT-004_PRAC-02_20260129_213231_4169...	29/01/2026 22:32	Fichier source JSON	1 Ko
PAT-005_PRAC-01_20260129_213231_4199...	29/01/2026 22:32	Fichier source JSON	1 Ko
PAT-001_PRAC-01_20260129_213220_9008...	29/01/2026 22:32	Fichier source JSON	1 Ko
PAT-003_PRAC-02_20260129_213215_9513...	29/01/2026 22:32	Fichier source JSON	1 Ko
PAT-005_PRAC-01_20260129_213211_0311...	29/01/2026 22:32	Fichier source JSON	1 Ko
PAT-001_PRAC-02_20260129_213205_7835...	29/01/2026 22:32	Fichier source JSON	1 Ko
PAT-002_PRAC-01_20260129_213205_7940...	29/01/2026 22:32	Fichier source JSON	1 Ko
PAT-003_PRAC-02_20260129_213205_7966...	29/01/2026 22:32	Fichier source JSON	1 Ko
PAT-004_PRAC-01_20260129_213205_7991...	29/01/2026 22:32	Fichier source JSON	1 Ko
PAT-005_PRAC-02_20260129_213205_8041...	29/01/2026 22:32	Fichier source JSON	1 Ko
PAT-001_PRAC-01_20260129_213200_7544...	29/01/2026 22:32	Fichier source JSON	1 Ko
PAT-004_PRAC-02_20260129_213200_8897...	29/01/2026 22:32	Fichier source JSON	1 Ko
PAT-001_PRAC-01_20260129_213150_7121...	29/01/2026 22:31	Fichier source JSON	1 Ko
PAT-004_PRAC-02_20260129_213150_8369...	29/01/2026 22:31	Fichier source JSON	1 Ko
PAT-004_PRAC-01_20260129_213145_8571...	29/01/2026 22:31	Fichier source JSON	1 Ko
PAT-003_PRAC-01_20260129_213140_7689...	29/01/2026 22:31	Fichier source JSON	1 Ko
PAT-002_PRAC-01_20260129_213135_7786...	29/01/2026 22:31	Fichier source JSON	1 Ko
PAT-002_PRAC-02_20260129_213130_6579...	29/01/2026 22:31	Fichier source JSON	1 Ko
PAT-003_PRAC-01_20260129_213130_6599...	29/01/2026 22:31	Fichier source JSON	1 Ko
PAT-001_PRAC-02_20260129_213125_5547...	29/01/2026 22:31	Fichier source JSON	1 Ko
PAT-004_PRAC-01_20260129_213125_6817...	29/01/2026 22:31	Fichier source JSON	1 Ko
PAT-002_PRAC-02_20260129_213120_6002...	29/01/2026 22:31	Fichier source JSON	1 Ko
PAT-004_PRAC-02_20260129_213120_6548...	29/01/2026 22:31	Fichier source JSON	1 Ko
PAT-002_PRAC-01_20260129_213105_5424...	29/01/2026 22:31	Fichier source JSON	1 Ko
PAT-003_PRAC-01_20260129_213050_5401...	29/01/2026 22:30	Fichier source JSON	1 Ko

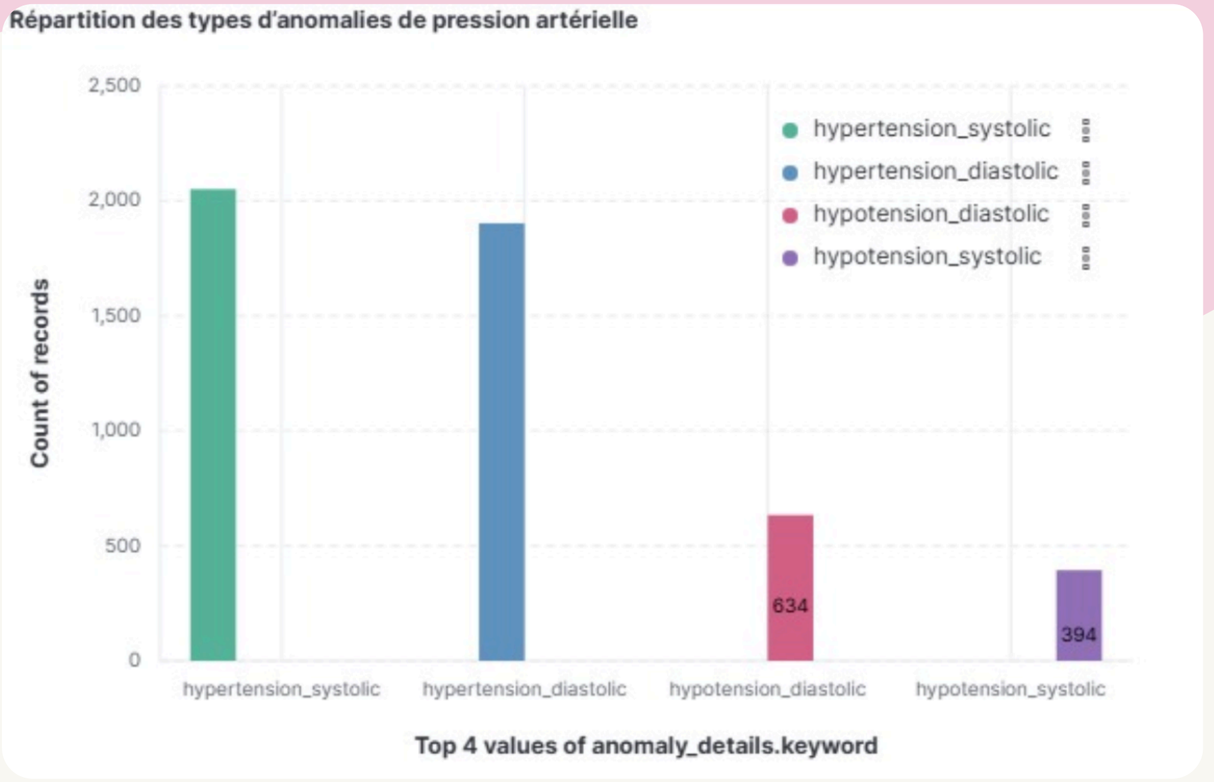
06 Visualisation et dashboards Kibana



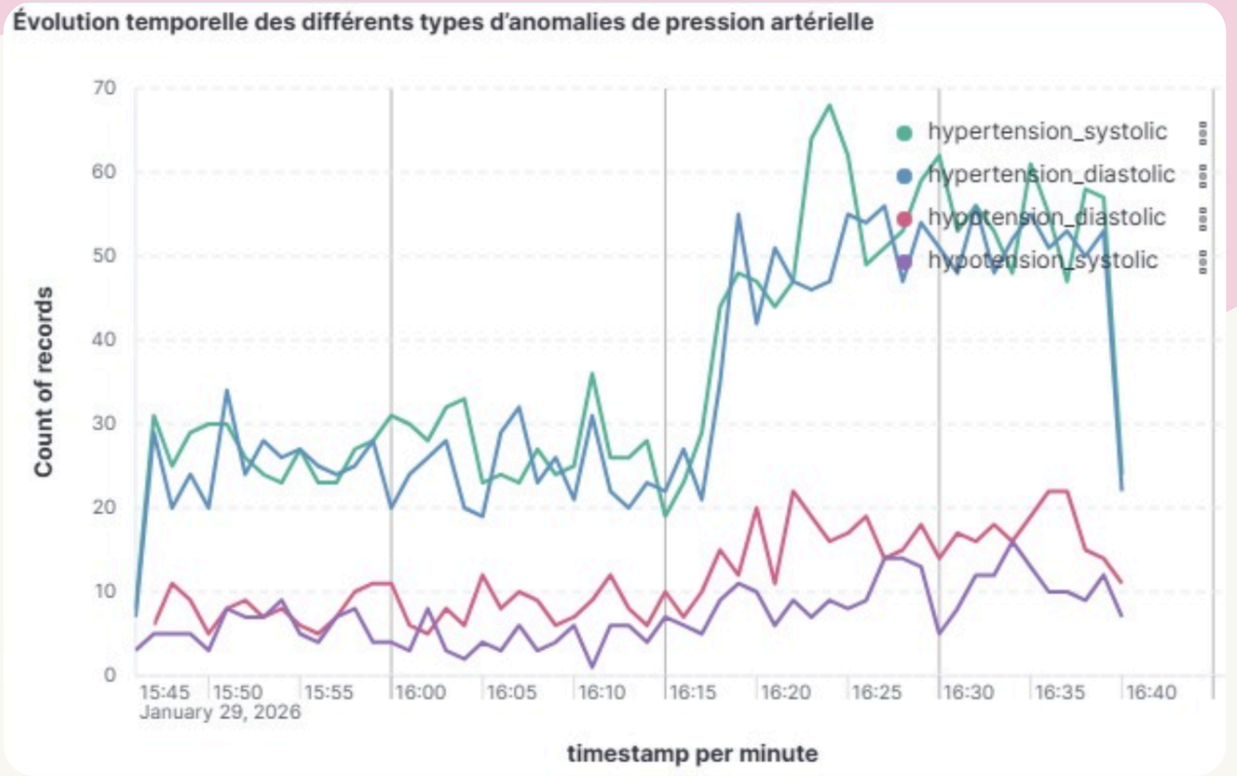


06 Visualisation et dashboards Kibana





06 Visualisation et dashboards Kibana



06 Visualisation et dashboards Kibana

Liste des patients présentant des anomalies de pression artérielle

patient_id.keyw	systolic_pressu	diastolic_pressu	anomaly_details	ml_risk_score
PAT-004	89	70	hypotension_systc	0.192
PAT-004	89	70	hypotension_systc	0.194
PAT-004	89	70	hypotension_systc	0.196
PAT-004	89	70	hypotension_systc	0.198
PAT-004	89	70	hypotension_systc	0.2
PAT-004	89	70	hypotension_systc	0.202
PAT-004	89	70	hypotension_systc	0.204
PAT-004	89	70	hypotension_systc	0.206
PAT-004	89	70	hypotension_systc	0.208

06 Visualisation et dashboards Kibana

Cas Critiques - alertes BP

14 documents | Columns 7 | Sort fields 1

timestamp	patient_id	systolic_pressure	diastolic_pressure	anomaly_details
Jan 29, 2026 @ 22:43:43.451	PAT-004	150	117	[hypertension_systolic , …
Jan 29, 2026 @ 22:43:43.450	PAT-003	152	116	[hypertension_systolic , …
Jan 29, 2026 @ 22:43:43.449	PAT-002	149	118	[hypertension_systolic , …
Jan 29, 2026 @ 22:43:43.448	PAT-001	176	107	[hypertension_systolic , …

Cas critiques récents de pression artérielle

timestamp pe	patient_id.keyw	systolic_pressu	diastolic_pressu	anomaly_details
22:45:00	PAT-002	108	99	hypertension_dias
22:45:00	PAT-002	111	108	hypertension_dias
22:45:00	PAT-002	152	76	hypertension_syst
22:44:30	PAT-003	80	94	hypertension_dias
22:44:30	PAT-003	157	74	hypertension_syst
22:44:30	PAT-003	167	118	hypertension_dias
22:44:30	PAT-003	171	70	hypertension_syst
22:44:30	PAT-003	180	70	hypertension_syst
22:44:00	PAT-005	82	62	hypotension_syst

07 Système d’alertes en temps réel

4 rules

Columns 10

Name	Last run	Notify	In...	Duration	P50	Success ra	Last response	State
Alerte - Pic anomalies BP Index threshold	Jan 29, 2026 23:02:53pm 41 seconds ago		1 min	00:00	00:01	100%	Succeeded	Enabled
Alerte BP — Cas critique (systolique + diastolique) Elasticsearch query	Jan 29, 2026 23:02:53pm 41 seconds ago		1 min	00:00	00:01	100%	Succeeded	Enabled
Alerte BP — Hypertension diastolique Elasticsearch query	Jan 29, 2026 23:02:53pm 41 seconds ago		1 min	00:00	00:01	100%	Succeeded	Enabled
Alerte – Hypertension systolique Elasticsearch query	Jan 29, 2026 23:02:53pm 41 seconds ago		1 min	00:00	00:01	100%	Succeeded	Enabled

Alerte - Pic anomalies BP

View in app Refresh Edit

Type: Index threshold

Rule is Enabled

283 executions in the last 24 hr

Last response a minute ago

Succeeded

Notify when alerts generated

Definition

Rule type Index threshold

Description Alert when an aggregated query meets the threshold.

Runs every 1 min

Conditions 0 conditions

Actions

server-log-bp
On status changes

server-log-bp
On status changes

08 Limites et perspectives

Limites

- Données simulées (pas de données cliniques réelles)
- Modèle ML simple (régression logistique)
- Règles cliniques statiques (seuils fixes)
- Pas de gestion de l’historique patient
- Scalabilité limitée (Kafka & consumer simples)

Perspectives

- Utilisation de données réelles anonymisées
- Modèles ML plus avancés (Random Forest, XGBoost, LSTM)
- Analyse temporelle par patient
- Alertes personnalisées selon le profil patient
- Déploiement cloud & monitoring avancé

09 Conclusion

- Pipeline temps réel complet de bout en bout
- Intégration Data Engineering + Data Science
- Détection d'anomalies par règles cliniques + Machine Learning
- Visualisation et alertes via Kibana
- Projet fonctionnel, modulaire et évolutif