

# Rapport Projet Fil-Rouge

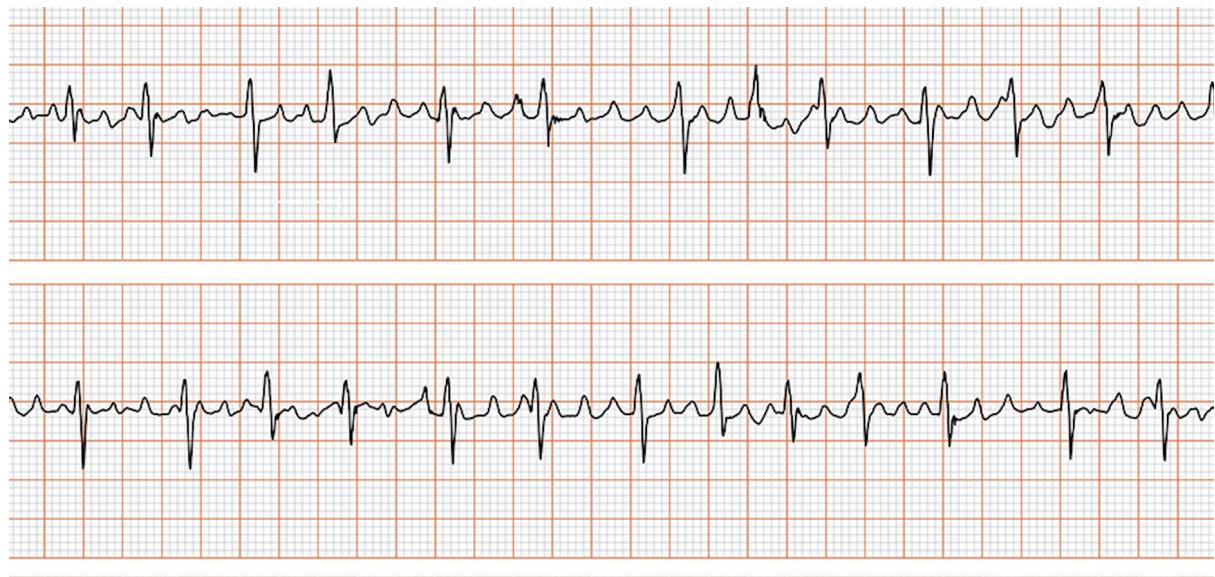
## Heart beat

### I. Introduction

#### 1. Présentation générale

Notre projet HeartBeat est un projet de data science appliquée au domaine de la santé. Il repose sur la récupération et la structuration automatique de métadonnées issues des jeux de données MIT-BIH Arrhythmia Database et PTB, en faisant principalement appel à des compétences en Text Mining des fichiers présents dans cette base.

Il demande aussi l'utilisation de bibliothèques spécialisées comme wfdb, pandas et re et des principes d'automatisation de pipeline de données (exploration récursive, nettoyage, consolidation, export CSV).



Sur le plan scientifique, il s'agit d'un projet de recherche sur l'analyse des signaux physiologiques, plus précisément de l'étude des arythmies cardiaques. Cela englobe la validation de modèles d'apprentissage automatique pour la classification de signaux ECG, et le développement de méthodes de détection.

L'objectif du projet MIT-BIH est de développer un modèle de classification automatique des signaux électrocardiographiques de la base de données MITBIH.

Plus précisément, ce projet vise à entraîner des algorithmes de machine learning capables d'identifier et de distinguer différents types de battements cardiaques (normaux, ventriculaires, supraventriculaires) à partir des signaux bruts enregistrés.

## 2. MITBIH

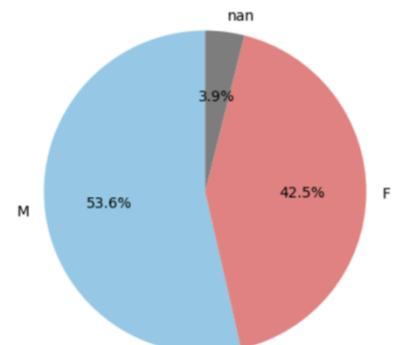
### 1. Présentation:

Le MIT-BIH Arrhythmia Database a été créé à la fin des années 1970 et publié pour la première fois en 1980 par le MIT (Massachusetts Institute of Technology) et le Beth Israel Hospital de Boston.

Il s'agit d'un jeu de données public, référence en électrocardiographie largement utilisé pour la détection automatique des arythmies cardiaques et l'évaluation d'algorithmes d'analyse du rythme cardiaque.

Le MIT-BIH contient 48 enregistrements d'une demi heure d'ECG sur deux canaux de 47 sujets (23 enregistrements au hasard, 25 enregistrements sélectionnés car cliniquement significants).

Parmi les sujets, on dénombre 25 hommes âgés de 25 à 89 ans, et 22 femmes âgées de 23 à 89 ans.



## 2. Technique d'enregistrement:

A l'époque, les résultats ont été enregistrés puis relus analogiquement avec les caractéristiques suivantes: 360 samples/secondes/canal, avec une sensibilité de 10 mV.

Puis ils ont été digitalisés au format Signal 5: format 212, c'est-à-dire sur 12 bits.

Ces différentes étapes ont impliqué la présence de différents artéfacts bien documentés (wobble, changement de bande, bruit...)

## 3. Labellisation:

Après de multiples corrections par différents cardiologues, chaque pic a été labellisé avec une annotation. Ce label constitue notre variable cible.

Label	Correspondance	Description
0	N	Normal beat
1	S	Supraventricular ectopic beat
2	V	Ventricular ectopic beat
3	F	Fusion beat
4	Q	Unknown beat
-1	/	Other

## 4. Contenu:

La base de données MITBIH est constituée d'un dossier contenant des fichiers de différents formats (.hea, .dat et .atr), et d'un sous dossier x\_mitbih contenant lui aussi des fichiers .hea, .dat et .atr.

Extension	Contenu	Rôle
-----------	---------	------

.hea	Texte	Contient les métadonnées du signal (format, fréquence d'échantillonnage, nombre de signaux, durée, etc.)
.dat	Données binaires	Contient les signaux ECG bruts (valeurs analogiques enregistrées par les électrodes)
.atr	Annotations	Contient les annotations des battements (instants des battements + labels : N, V, S, F, Q, etc.)

## 5. Remarques:

La polarité de l'enregistrement 114 a été inversée.

La vitesse des enregistrements 112, 115 -> 124, 205, 220, 223, 230 -> 234 est le double de la vitesse réelle. (Après vérification, la vitesse des ces enregistrements a été réajustée dans notre version du jeu de données)

## 3. PTB

### 1. Présentation

La PTB Diagnostic ECG Database a été publiée par le Physikalisch-Technische Bundesanstalt (PTB), l'institut national de métrologie allemand.

Elle est disponible publiquement via la plateforme PhysioNet et sert de référence pour l'étude de l'ischémie myocardique, de l'infarctus du myocarde et d'autres pathologies cardiaques.

Cette base se distingue de MIT-BIH par son caractère fortement clinique :

Elle contient, en plus des signaux ECG, de nombreuses informations médicales et hémodynamiques (diagnostics, localisation d'infarctus, médications, mesures invasives, etc.) au niveau patient / visite.

La base PTB regroupe des enregistrements ECG de patients malades (principalement infarctus du myocarde) et de sujets sains ("healthy controls").

Un même patient peut avoir plusieurs enregistrements (visites ou examens différents).

### 2. Technique d'enregistrement

Les enregistrements ECG de PTB sont réalisés dans un cadre hospitalier, avec :

- une acquisition multicanale, généralement les 12 dérivations standards (I, II, III, aVR, aVL, aVF, V1–V6),
- une fréquence d'échantillonnage élevée, de l'ordre de 1000 Hz,
- une amplitude exprimée en millivolts (mV).

Dans notre travail, ces signaux sont ensuite :

- resamplés à 360 Hz pour être compatibles avec MIT-BIH,
- fenêtrés autour des R-peaks (fenêtres de 187 points) pour extraire les battements,
- normalisés (correction de la ligne de base, centrage-réduction) avant d'être passés dans les modèles.

### 3. Labellisation

Contrairement à MIT-BIH, la base PTB ne fournit pas de labels au niveau du battement. Les informations de diagnostic sont :

- disponibles au niveau patient / visite (infarctus aigu, infarctus ancien, autres pathologies cardiaques, healthy controls, etc.),
- mais il n'existe pas d'annotations "beat-level" indiquant si chaque battement est normal ou anormal.

Dans ce projet, la stratégie de labellisation est donc indirecte :

- au niveau battement, des modèles entraînés sur MIT-BIH (XGBoost, réseau de neurones) sont appliqués sur PTB pour générer des pseudo-labels normal / anormal,
- au niveau patient, on distingue :
  - les sujets "Healthy control" → patients sains,
  - les autres diagnostics (infarctus, pathologies cardiaques, etc.) → patients malades.

### 4. Contenu

La base PTB se présente, comme MIT-BIH, sous forme de fichiers de différents formats. On retrouve notamment :

Extension	Contenu	Rôle
-----------	---------	------

.hea	Texte	<b>Métadonnées</b> de l'enregistrement : format, fréquence d'échantillonnage, nombre de signaux, noms des dérivations, durée, unités, etc.
.dat	Données binaires	<b>Signaux ECG bruts</b> mesurés par les électrodes (valeurs analogiques numérisées).
.atr	Annotations	<b>Vectorcardiographie*</b> : représentation 3D du vecteur électrique cardiaque, dérivée des 12 dérivations.

\*La vectocardiographie (VCG), est une méthode d'exploration du cœur qui enregistre non pas juste la tension électrique comme l'ECG classique, mais le vecteur du champ électrique cardiaque dans l'espace.

## 5. Remarques

- La base PTB est très riche en métadonnées, mais avec une complétude très variable d'une variable à l'autre (certaines mesures invasives ou paramètres d'effort étant souvent manquants).
- Aucun fichier d'annotations de type .atr n'est fourni : la labellisation beat-level doit donc être construite via des méthodes de transfert de modèle (MIT → PTB).
- Le fait qu'un même patient puisse avoir plusieurs enregistrements impose d'être attentif aux fuites de données lors de la construction des jeux d'entraînement / test (groupes par patient ou par visite).

# II. Extraction et exploration des données

## 1. MIT-BIH

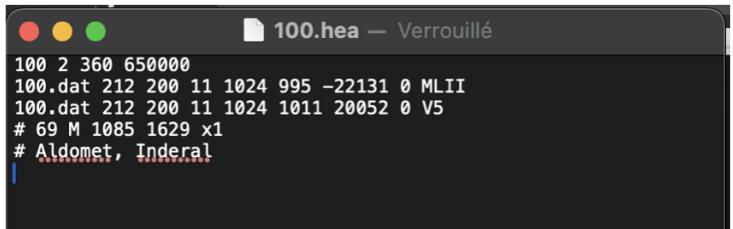
### 1. Extraction des données-Text Mining

1. Première étape: récupérer les métadonnées avec .hea et .atr:

#### a. Objectif global:

On veut rassembler toutes les métadonnées des enregistrements du jeu de données MIT-BIH Arrhythmia Database en un seul fichier CSV. Chaque enregistrement correspond à un battement de cœur). On récupère ainsi:

- la fréquence d'échantillonnage (fs),
- la durée,
- le nombre de signaux,
- le sexe, l'âge du patient,
- le nombre de battements détectés,
- la fréquence cardiaque moyenne (bpm\_moyen).



```
100 2 360 650000
100.dat 212 200 11 1024 995 -22131 0 MLII
100.dat 212 200 11 1024 1011 20052 0 V5
# 69 M 1085 1629 x1
# Aldomet, Inderal
```

Pour parcourir les fichiers, manipuler des textes, stocker les données et gérer les options, nous utiliserons os, re, pandas, et argparse.

Pour manipuler les fichiers ECG, nous utiliserons WFDB (WaveForm DataBase).

wfdb.rdsamp('XXX') pour lire les métadonnées .hea  
wfdb.rdrecord('XXX') pour lire les enregistrements .dat

### c. Recherche et lecture des fichiers .hea

On parcourt le dossier racine (ROOT\_DEFAULT) et on cherche tous les fichiers qui se terminent par .hea.

Chaque fichier .hea contient un en-tête décrivant un enregistrement ECG (nom, fréquence, nombre d'échantillons, etc.).

### d. Extraction des informations utiles

Pour chaque fichier, la lecture de la première ligne nous donne:

- le nom du record (ex: 121 ou x\_121),
- le nombre de signaux,
- la fréquence d'échantillonnage (fs),
- le nombre d'échantillons (nsamp).

La lecture des lignes suivantes nous donne des informations sur les canaux utilisés (ex: MLII, V5...). L'âge et le sexe du patient, les médicaments ou notes se trouvent après le #.

On obtient le nombre de battements (nb\_beats) via le fichier d'annotations .atr correspondant.

#### e. Calculs dérivés

Durée totale d'enregistrement:	$duree\_sec = nb\_echantillons / fs$
Fréquence cardiaque moyenne:	$bpm\_moyen = (nb\_beats * 60) / duree\_sec$

#### f. Consolidation (fusion des doublons)

Certains enregistrements existent sous deux noms (ex: 108 et x\_108).

La fonction consolidate() regroupe tout par record\_num, garde les valeurs pertinentes (max, première non nulle, etc.), crée une table propre (1 ligne par enregistrement).

#### g. Harmonisation “2x vitesse”

Après vérification, la base de données sur PhysioNet a déjà procédé à l'harmonisation des enregistrements.

#### h. Export final au format csv.

2. Deuxième étape: récupérer les points des battements et annotation correspondante avec .dat et .atr:

#### a. Objectif

Créer un CSV de segments de battements à partir des enregistrements MIT-BIH. Chaque ligne correspond à un battement, avec :

- l'ID du record,
- l'index d'échantillon,
- le temps en secondes,
- le lead,
- 187 points du signal autour du battement (fenêtre centrée),
- le symbole d'annotation MIT-BIH
- un label (de 0 à 4, ou -1 hors mapping).

#### **b. Entrées / Sortie**

Le code va parcourir le dossier de base BASE\_DIR (fichiers .dat, .hea, .atr), et va scanner les .dat.

#### **c. Normalisation des records**

On convertit 121 → x\_121 et on extrait record\_num, de manière à harmoniser le format dans le csv et faciliter la manipulation (fusion d'entrées ou merge de csv)

#### **d. Lecture du signal & annotations**

On charge le record avec wfdb (rdrecord), on récupère fs et les noms de canaux.

Pour le choix du canal, on choisit MLII de préférence, sinon canal 0.  
Enfin on récupère les positions d'échantillons (ann.sample) et symboles (ann.symbol) avec rdann.

#### **e. Fenêtrage autour de chaque battement**

On veut une fenêtre fixe 187 échantillons. On fixe à 90 les points avant le R-peak PRE=90, et 96 ceux après POST=96.

Si la fenêtre sort du début et de la fin du signal, on ignore le battement.

Pour chaque annotation à l'échantillon s : calcule le temps t\_sec = s / fs.

#### f. Harmonisation “2x vitesse”:

Après vérification, la base de données sur PhysioNet a déjà procédé à l'harmonisation des enregistrements.

#### g. Mapping des labels

On mappe les symboles MIT-BIH vers classes :

N=0, S=1, V=2, F=3, Q=4.

Les autres symboles sont conservés mais label = -1.

#### h. On exporte le csv.

3. Analyse des csv avant merge:

##### a. Le **MITBIH\_metadata\_all.csv**:

Colonnes:	record, record_num, age, sexe, leads, fs, nb_signaux, nb_echantillons, duree_sec, duree_min, nb_beats, bpm_moyen, medications, notes
Dimensions:	48 lignes, 14 colonnes
Présence de NaN:	2 NaN dans les colonnes Age et Sexe 21 NaN dans la colonne Medication.

	record	record_num	age	sexe	leads	fs	nb_signaux	nb_echantillons	duree_sec	duree_min	nb_beats	bpm_moyen	medications	notes
0	x_100	100	69.0	M	MLII,V5	360.0	2	650000	1805.555556	30.09	2274	75.566769	Aldomet, Inderal	69 M 1085 1629 x1   Aldomet   Inderal
1	x_101	101	75.0	F	MLII,V1	360.0	2	650000	1805.555556	30.09	1874	62.274462	NaN	75 F 1011 654 x1   Diapres
2	x_102	102	84.0	F	V2,V5	360.0	2	650000	1805.555556	30.09	2192	72.841846	NaN	84 F 1525 167 x1   Digoxin   The rhythm is pac...
3	x_103	103	NaN	NaN	MLII,V2	360.0	2	650000	1805.555556	30.09	2091	69.485538	Diapres, Xyloprim	-1 M 742 654 x1   Diapres   Xyloprim
4	x_104	104	66.0	F	V2,V5	360.0	2	650000	1805.555556	30.09	2311	76.796308	Digoxin, Pronestyl	66 F 1567 694 x1   Digoxin   Pronestyl   The r...

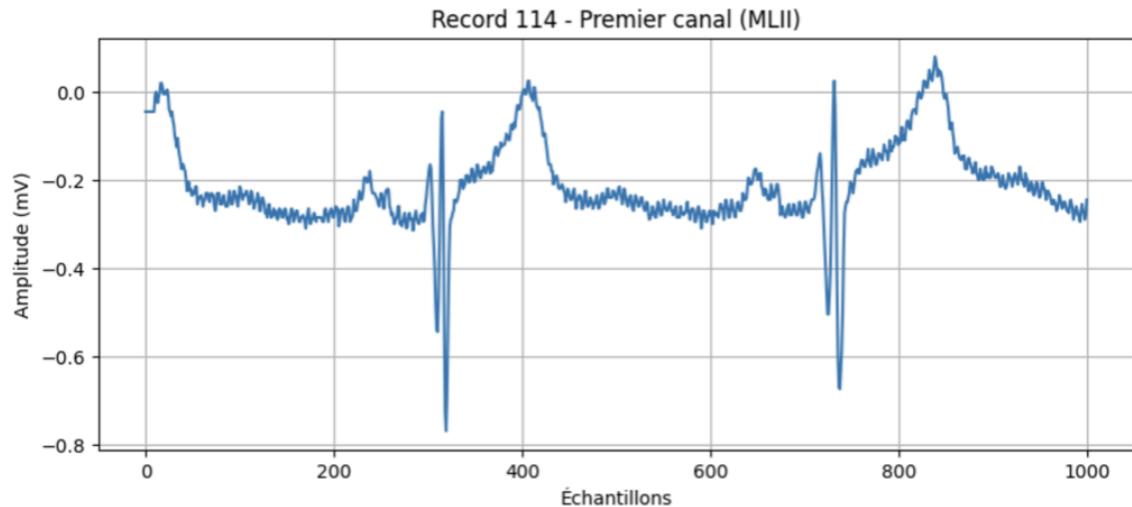
##### b. Le **MITBIH\_segments.csv**:

Colonnes:	record, record_num, sample, t_sec, lead, 0, 1, ..., 186, symbol, label Les colonnes 0, 1,..., 186 correspondent aux points du signal.
Dimensions:	450308 lignes 194 colonnes
Présence de NaN:	0

	record	record_num	sample	t_sec	lead	0	1	2	3	4	...	179	180	181	182	183	184	185	186	symbol	label
0	x_100	100	370	1.027778	MLII	-0.305	-0.310	-0.300	-0.305	-0.290	...	-0.430	-0.450	-0.455	-0.450	-0.435	-0.445	-0.450	-0.455	N	0
1	x_100	100	662	1.838889	MLII	-0.330	-0.345	-0.345	-0.355	-0.350	...	-0.415	-0.415	-0.410	-0.435	-0.430	-0.435	-0.420	-0.420	N	0
2	x_100	100	946	2.627778	MLII	-0.375	-0.385	-0.355	-0.360	-0.355	...	-0.380	-0.390	-0.375	-0.375	-0.365	-0.370	-0.385	-0.395	N	0
3	x_100	100	1231	3.419444	MLII	-0.345	-0.345	-0.350	-0.355	-0.345	...	-0.435	-0.445	-0.470	-0.455	-0.460	-0.440	-0.435	-0.425	N	0
4	x_100	100	1515	4.208333	MLII	-0.340	-0.340	-0.340	-0.325	-0.325	...	-0.420	-0.430	-0.445	-0.440	-0.435	-0.430	-0.430	-0.450	N	0

#### 4. Inversion du signal 114:

En utilisant wfdb.rdrecord sur le signal 114, on peut voir que le signal est encore inversé (R-peak vers le bas). Il nous faut donc l'inverser avant de fusionner les bases de données.



#### 5. Merge

On fait une jointure à gauche des deux csv après les avoir importés dans deux DataFrames distincts.

Caractéristiques du csv fusionné:

Colonnes	record_x, record_num, sample, t_sec, lead, 0, 1, 2, ..., leads, fs, nb_signaux, nb_echantillons, duree_sec, duree_min, nb_beats, bpm_moyen, medications, notes
Dimensions:	450308 lignes 207 colonnes.

Présence de NaN:

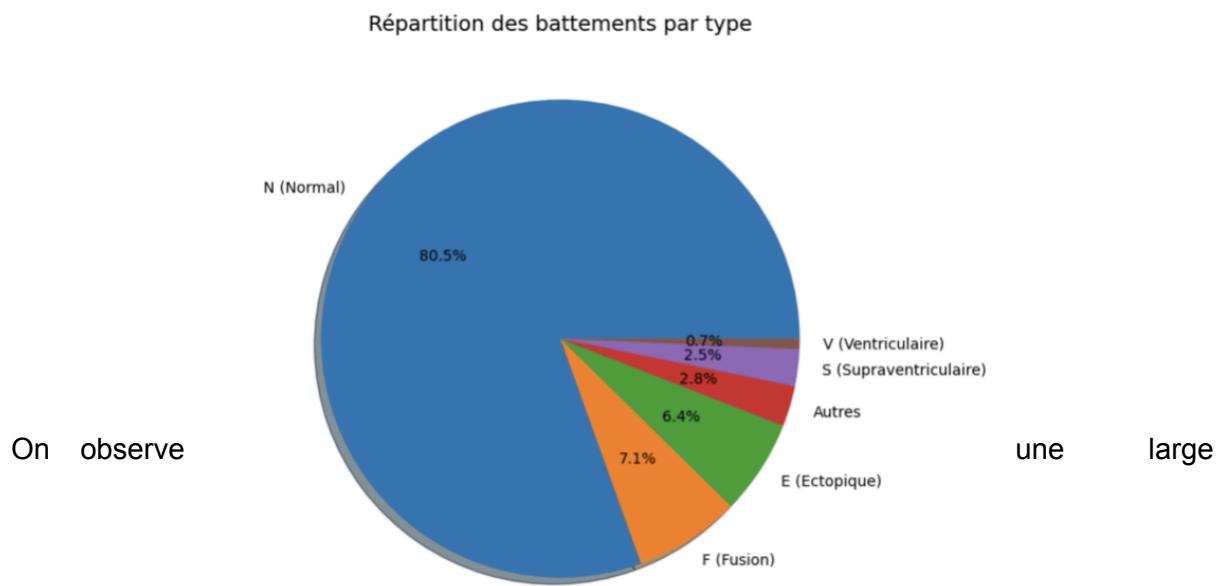
On observe la présence de 4% de NaN pour les colonnes Âge et Genre. Nous verrons comment les traiter après une exploration des variables plus approfondies. Un des axes de réflexion serait de voir si l'on peut imputer le sexe et le genre avec des fréquences cardiaques identiques.

On observe aussi plus de 20 % de NaN dans la colonne Medications. Nous verrons alors l'opportunité de travailler ou non avec cette variable.

Il y a aussi des colonnes avec un label -1 qui ne correspond à aucun type de battement.

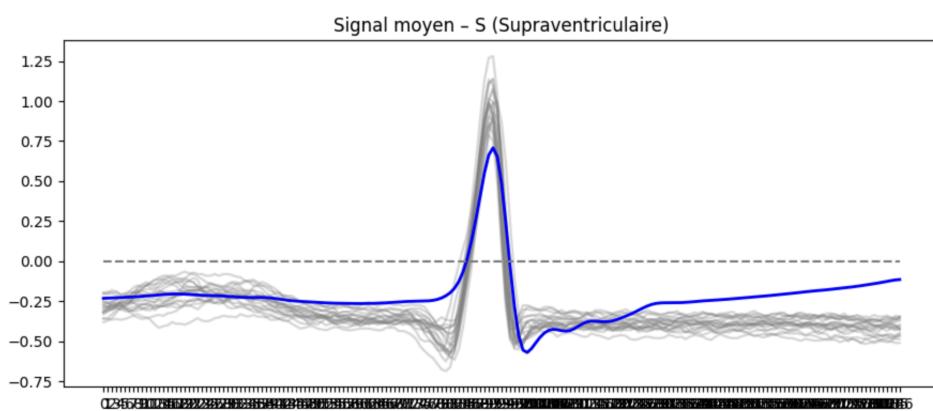
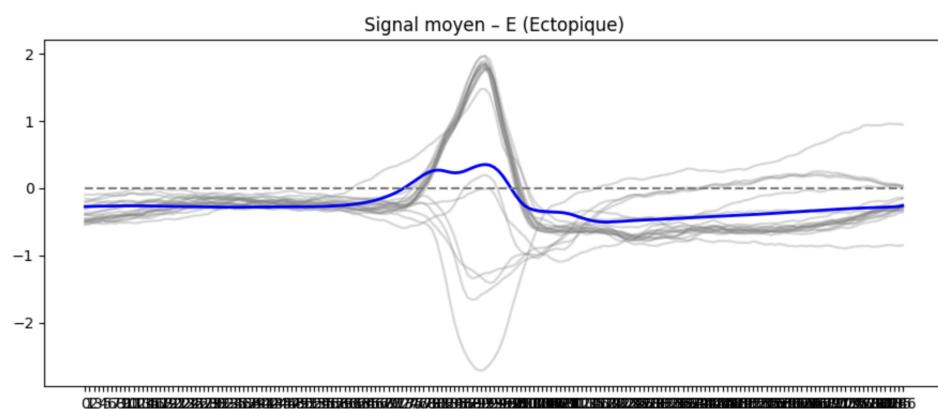
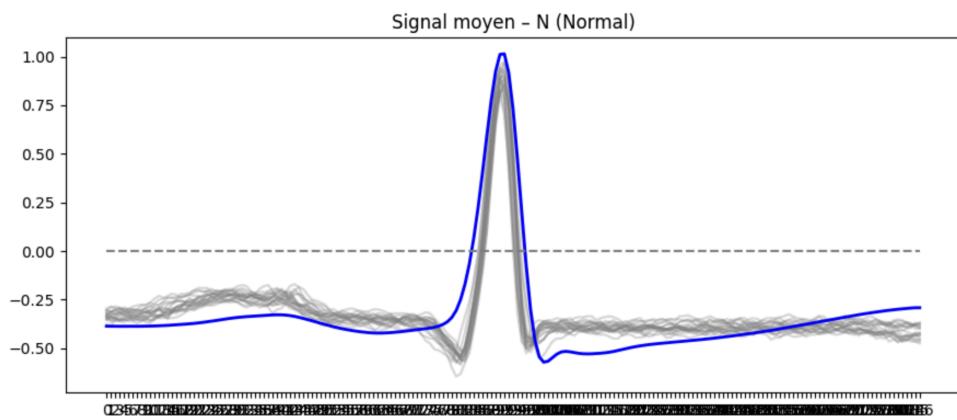
## 2. Exploration des données-Datavisualisation

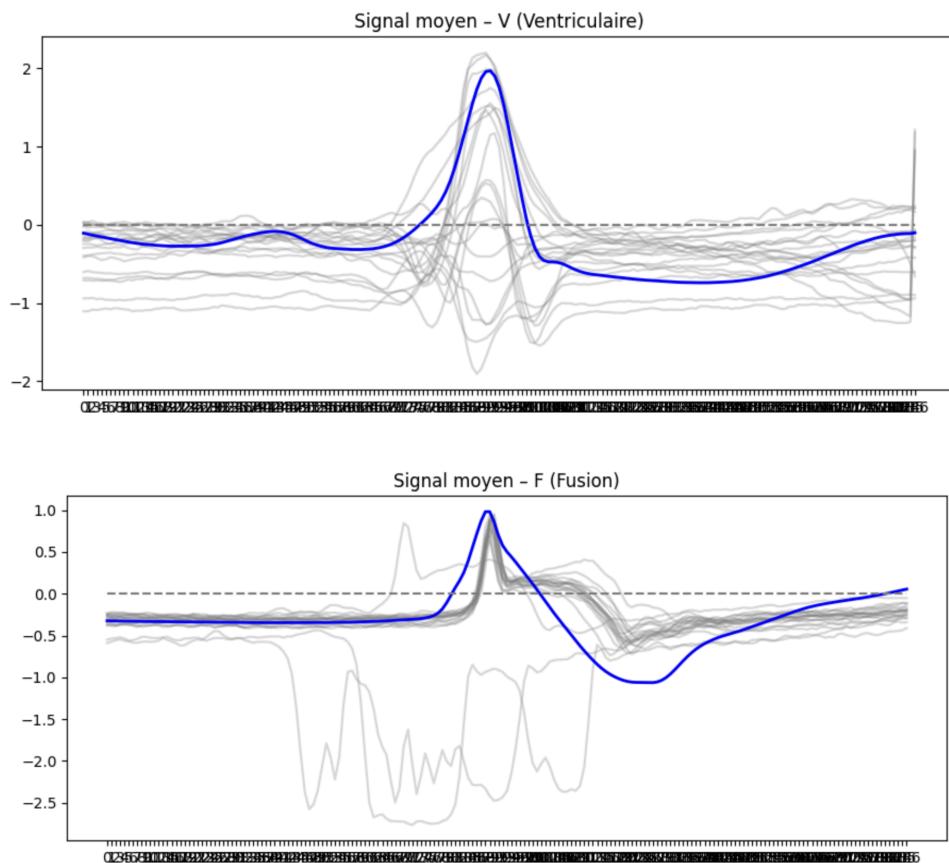
1. Proportion des types de battements dans la base de données MITBIH.



surreprésentation des battements normaux dans la base de données MITBIH. Cela aura un impact sur la stratégie de pré-processing des data comme nous le verrons dans la partie Stratégie d'undersampling.

2. Tracé de courbe moyenne pour chaque type de battements (ainsi que quelques courbes appartenant à ce type):



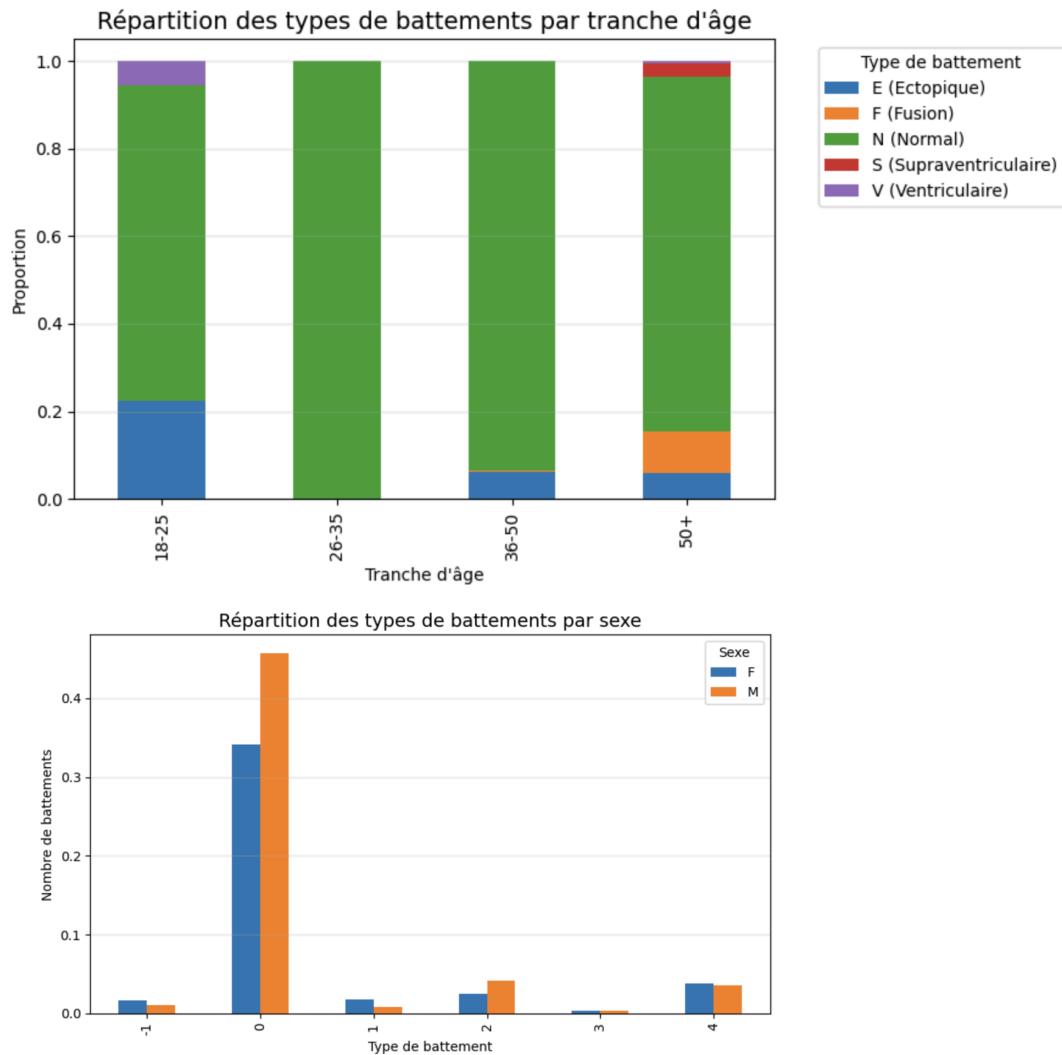


On observe que mise à part pour les signaux Normal (N) et Supraventriculaire (S), on observe une dispersion assez grande des courbes de battement.

Il semblerait que le signal ectopique (E) change de polarité, ce qui pourrait expliquer le signal moyen quasi nul.

Le signal fusion (F) est relativement reconnaissable par sa forme caractéristique, le signal le plus compliqué à classifier visuellement serait le signal ventriculaire (V) qui présente le plus de dispersion.

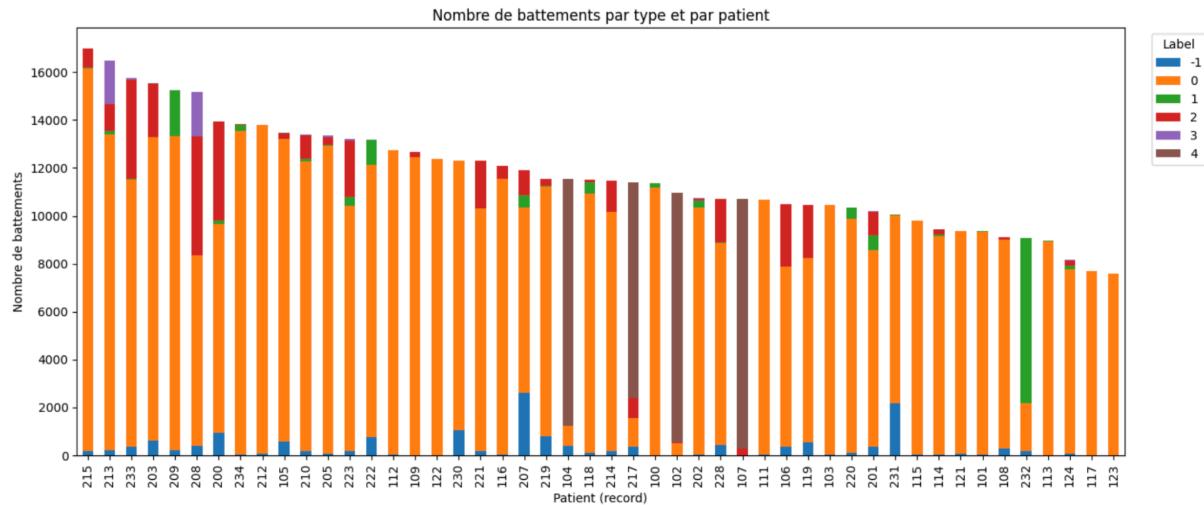
### 3. Répartition des types de battements par tranche d'âge et sexe:



Si l'on analyse la répartition des types de battements par genre, en considérant la proportion homme/femme de la base de donnée, on remarque que les femmes sont plus représentées pour les label 1 et 4 (Supraventricular ectopic beat, et la catégorie Unknown).

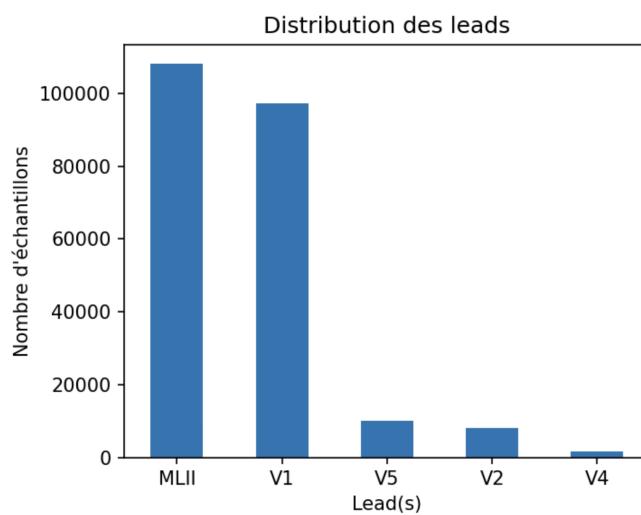
La répartition des types de battements par tranche d'âge montre que la catégorie des 26-35 ans de notre base de données ne présente que des battements normaux, les battements anormaux étant le plus présent en proportion pour les catégories 18-25 ans et plus de 50 ans.

#### 4. Répartition des battements par type et par enregistrement:



On observe une grande disparité de types de battements parmi les patients.

#### 5. Répartition des leads dans la base MIT-BIH:

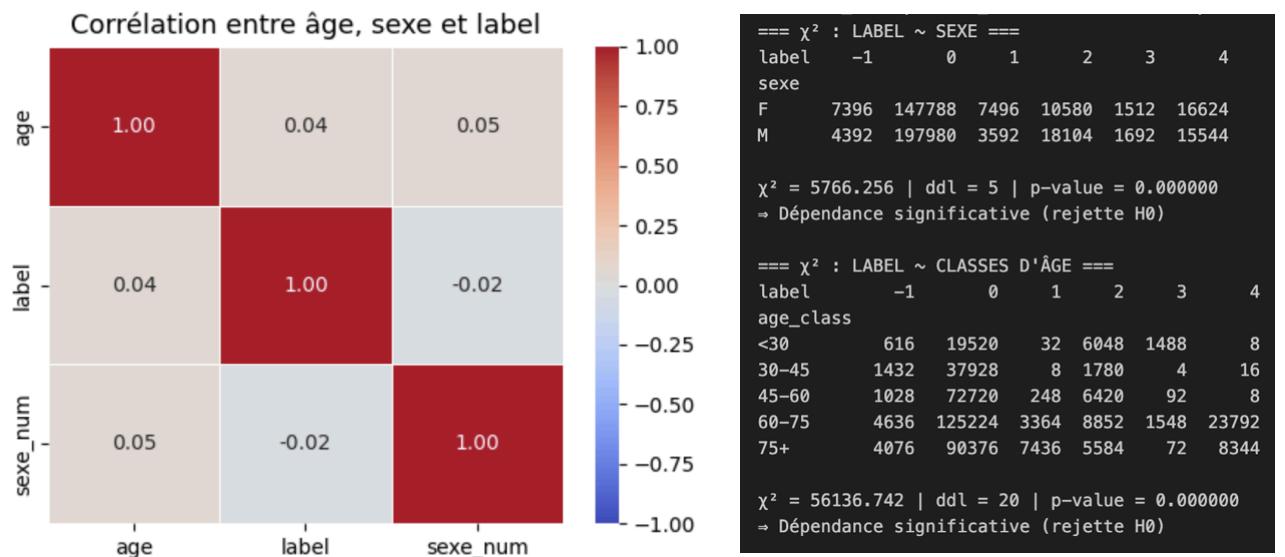


On observe que les signaux proviennent de deux leads différents, le MLII et le V5.

Ces leads correspondent à des emplacements de dérivation différents. Le V5 étant plus proche du ventricule, il est plus sensible aux signaux ventriculaires. En revanche, il n'a pas

la même amplitude ni la même morphologie. Pour garder une cohérence dans nos données, il faut donc supprimer ce lead, malgré le fait qu'il contienne beaucoup de classes anormales.

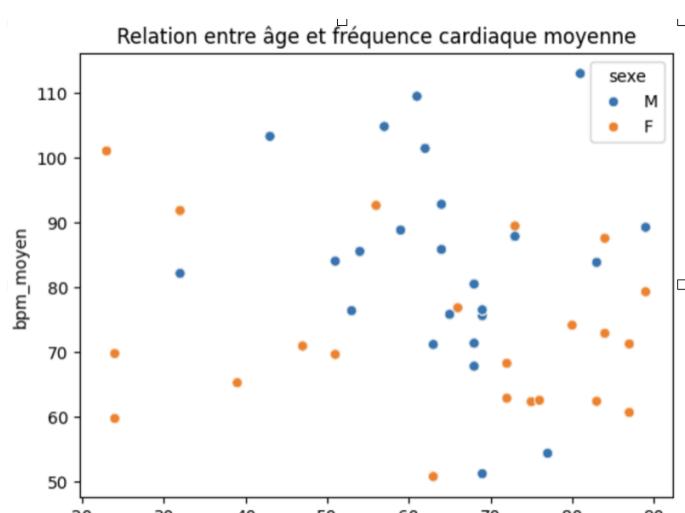
## 6. Etude de la corrélation entre les variables âge, sexe et le label:



Un test de Chi2 sur le sexe et l'âge des patients montre qu'il y a bien une relation entre le label et le sexe et l'âge, cependant la heatmap montre que cette relation est relativement faible ( abs(coefficient de Pearson) < 0,05 )

## 7. Etude de la corrélation entre fréquence cardiaque et âge/genre:

Intuitivement, on imagine une relation entre l'âge et la fréquence cardiaque, ainsi qu'une corrélation entre la fréquence cardiaque et le genre. Si la corrélation est assez forte, cela nous permettra d'imputer les 'âge' et 'genre' manquants dans notre base de données.



Il n'apparaît aucune corrélation claire entre l'âge, le genre et la fréquence cardiaque. Il faudra trouver une autre solution pour traiter les NaN.

On pourrait remplacer les NaN par la médiane pour la variable 'âge', et le mode pour la variable 'genre', mais étant donné qu'il ne s'agit que de 4% des valeurs du dataset, on pourra se permettre de supprimer les lignes contenant ces NaN.

## 8. Conclusion:

Compte tenu des observations précédentes, nous allons préparer notre base de donnée de la manière suivante:

- Suppression des lignes avec des NaN pour la variable Genre et la variable Age.
- Suppression des lignes correspondant aux enregistrements 207 et 231.
- Suppression des lignes correspondant à lead = 'V5'

Une fois notre base de données nettoyée, il ne nous restera plus qu'à adopter une stratégie pour 'choisir' les battements de manière judicieuse.

## 2. PTB

### 1 . Extraction des données – Text Mining

#### 1. Première étape : récupérer les métadonnées via les fichiers .hea

#### **Objectif global**

Comme pour MIT-BIH, on veut rassembler toutes les métadonnées de la PTB Diagnostic ECG Database dans un fichier CSV unique (ptb\_metadata\_full.csv).

Ici, une ligne = un enregistrement patient/visite.

On récupère notamment :

- record\_name, patient\_id
- fréquence d'échantillonnage (fs), nombre de signaux, nombre d'échantillons

- liste des dérivations (sig\_name) et unités
- âge, sexe
- diagnostics et localisation d'infarctus
- dates (admission, infarctus, cathétérisme, etc.)
- paramètres hémodynamiques (pressions, index cardiaque, etc.)
- médications et remarques cliniques

## Modules utilisés

- os, re, pathlib, pandas – pour parcourir les dossiers, parser les textes, stocker les données
- wfdb – pour lire les en-têtes ECG

La logique de parcours des fichiers et de lecture des .hea est la même que pour MIT (*comme vu pour MIT*).

## Recherche et lecture des fichiers .hea (comme vu pour MIT)

- Parcours récursif du répertoire PTB pour trouver tous les fichiers se terminant par .hea.
- Chaque .hea contient une ligne d'en-tête principale (record, n\_signals, fs, n\_samples) puis une ligne par signal, suivies de lignes de commentaires commençant par #.

## Extraction des informations utiles

- Première ligne → informations techniques : record\_name, n\_signals, fs, n\_samples.
- Lignes de signaux → noms de leads, gain, unités.
- Lignes # → infos cliniques : âge, sexe, diagnostics, localisation(s) d'infarctus, médications, dates, mesures hémodynamiques, etc.

On calcule ensuite quelques **variables dérivées** :

- duree\_sec = n\_samples / fs
- indicateurs de complétude (taux de NaN par colonne).

On obtient un CSV **1 ligne / enregistrement** : ptb\_metadata\_full.csv.

2. Deuxième étape : extraire des segments de battements (comme vu pour MIT)

## Objectif

Créer un CSV de segments de battements PTB harmonisés au format MIT-BIH.  
Comme pour MIT, chaque ligne correspond à un battement avec :

- patient\_id, record\_name
- index d'échantillon du R-peak, temps en secondes
- lead utilisé
- 187 points du signal (fenêtre centrée sur le R-peak)

### Entrées / sortie

- Entrée : fichiers .dat + .hea de PTB
- Sortie : ptb\_beats\_187pts.csv

### Lecture du signal & choix du lead (méthode similaire à MIT)

- Lecture du signal avec wfdb.rdrecord(record\_name) pour récupérer :
  - le tableau de signaux,
  - fs,
  - la liste des dérivations.
- Choix d'un lead principal :
  - les modèles MIT étant entraînés sur MLII, on choisit dans PTB le lead II comme lead de référence (lead le plus proche).

### Resampling & détection des R-peaks

- Les enregistrements PTB sont à ~1000 Hz → on resample à 360 Hz, comme pour MIT (*comme vu pour MIT*).
- On applique un algorithme de détection de R-peaks sur le lead choisi (Pan-Tompkins ou équivalent via WFDB).
- On conserve uniquement les R-peaks dont la fenêtre de 187 points reste entièrement à l'intérieur du signal.

### Fenêtrage autour de chaque battement (comme vu pour MIT)

- On utilise la même fenêtre que pour MIT :
  - PRE = 90 points avant R,
  - POST = 96 points après R → 187 points au total.
- Pour chaque R-peak à l'échantillon s :
  - on extrait les valeurs  $x[s-PRE : s+POST+1]$ ,
  - on calcule  $t\_sec = s / fs$ .

Les fenêtres invalides (début/fin de signal) sont ignorées.

On obtient `ptb_beats_187pts.csv`, contenant pour chaque battement : identifiants, temps, lead, 187 points.

### 3. Analyse rapide des CSV PTB (avant fusion / modélisation)

#### `ptb_metadata_full.csv`

- Colonnes typiques : `patient_id`, `record_name`, `age`, `sex`, `fs`, `n_signals`, `n_samples`, `duree_sec`, `diagnostics`, localisation d'infarctus, paramètres hémodynamiques, médications, dates...
- Dimensions : une ligne par enregistrement.
- **NaN** :
  - très peu dans les champs techniques (`fs`, `n_samples`, `leads...`),
  - beaucoup dans certaines variables cliniques fines (mesures invasives, effort, médications détaillées).

#### `ptb_beats_187pts.csv`

- Colonnes : identifiants, `sample`, `t_sec`, `lead`, `x0...x186`, `label_xgb`, `label_nn`, etc.
- **NaN** quasiment absents (fenêtres invalides filtrées en amont, comme pour MIT).

### 5. Conclusion : préparation de la base PTB

En résumé, en reprenant la **même philosophie que pour MIT** (*comme vu pour MIT*) :

1. On parse les métadonnées PTB (.hea) → `ptb_metadata_full.csv`.
2. On extrait des segments de battements PTB au format MIT (187 points, 360 Hz, lead II) → `ptb_beats_187pts.csv`.

On obtient ainsi une base PTB :

- **alignée techniquement** avec MIT-BIH,
- **enrichie en métadonnées cliniques**,
- prête pour la suite : **exploration, extrapolation des modèles et analyse patients sains/malades**.

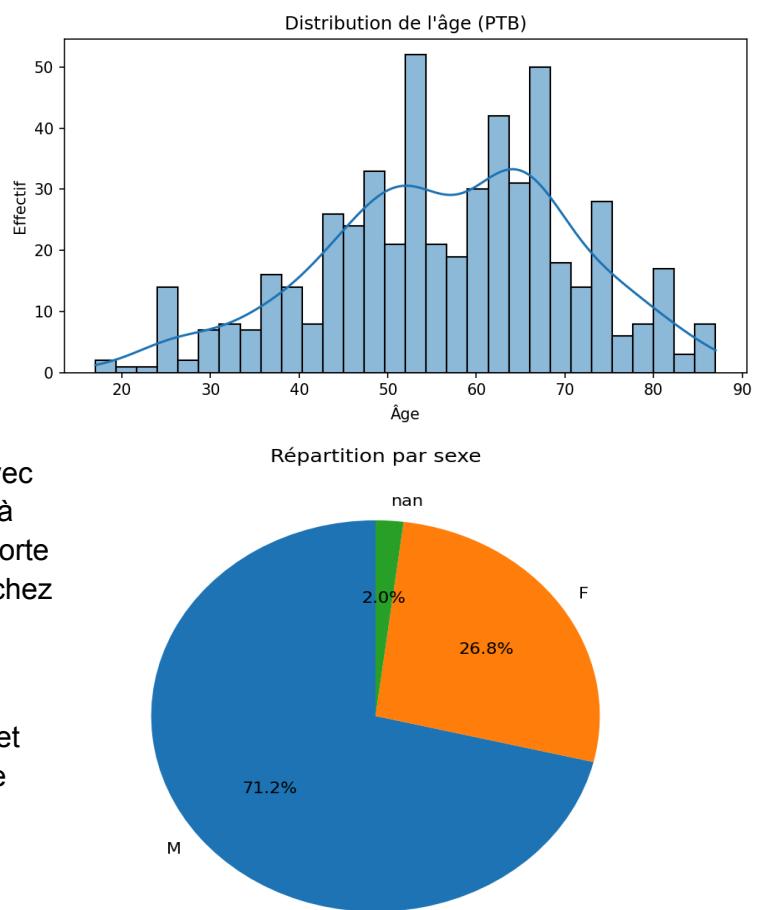
## 2. Exploration des données – Datavisualisation (PTB)

### 1. Profil démographique des patients

Comme pour MIT-BIH (*comme vu pour MIT*), on commence par regarder l'**âge** et le **sexé** des patients.

- L'histogramme de l'âge montre une distribution centrée autour de 60–70 ans, avec très peu de patients jeunes. La base PTB reflète donc une population majoritairement âgée, cohérente avec l'épidémiologie de l'infarctus du myocarde.
- Le camembert de répartition par sexe met en évidence une forte prédominance masculine : environ 70 % d'hommes pour 30 % de femmes, avec très peu de valeurs non renseignées. Là encore, cela est cohérent avec la plus forte incidence des syndromes coronariens chez l'homme.

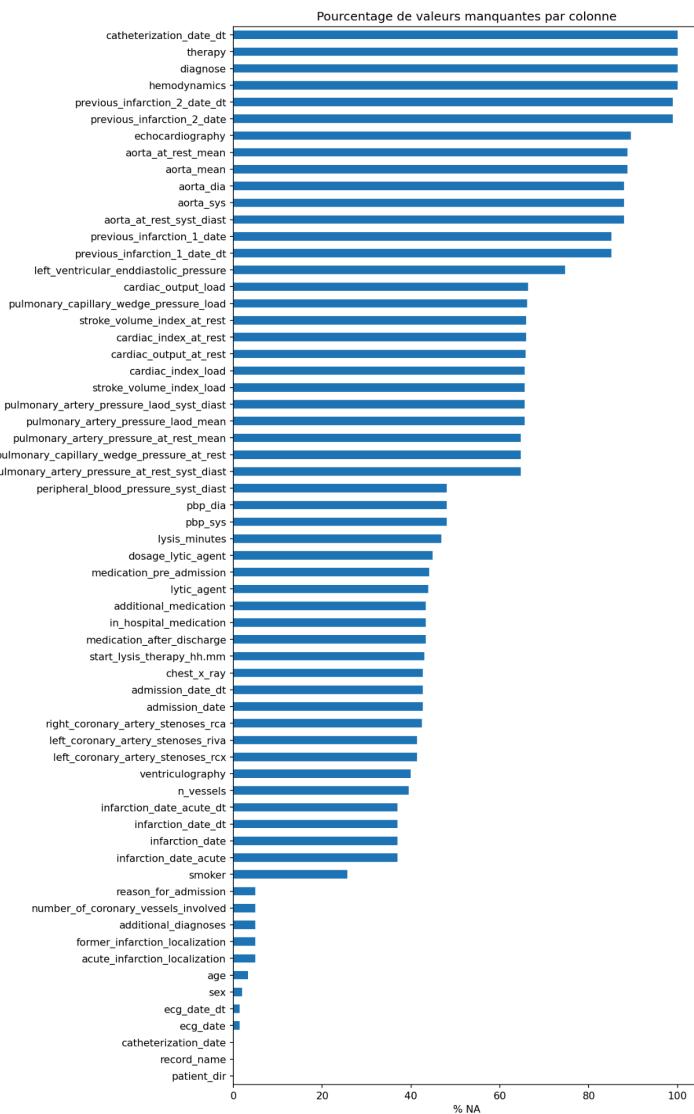
Ces éléments confirment le caractère clinique et spécialisé de la base PTB, très différente d'une population générale.



## 2. Analyse des valeurs manquantes dans les métadonnées

Comme pour MIT (*même approche de barplot des NA*), on a étudié le pourcentage de valeurs manquantes par colonne dans `ptb_metadata_full.csv`.

- Certaines variables techniques (fréquence d'échantillonnage, nombre de signaux, nombre d'échantillons, noms des dérivations...) présentent quasi 0 % de NA. Elles sont donc fiables et peuvent être utilisées directement.
- À l'inverse, plusieurs variables cliniques fines présentent des taux de NA très élevés (parfois > 70–80 %) : certaines dates précises, certains paramètres hémodynamiques avancés, ou encore des champs textuels très spécifiques.
- Les variables “médication détaillée” et certains champs d’effort sont également relativement peu complets.



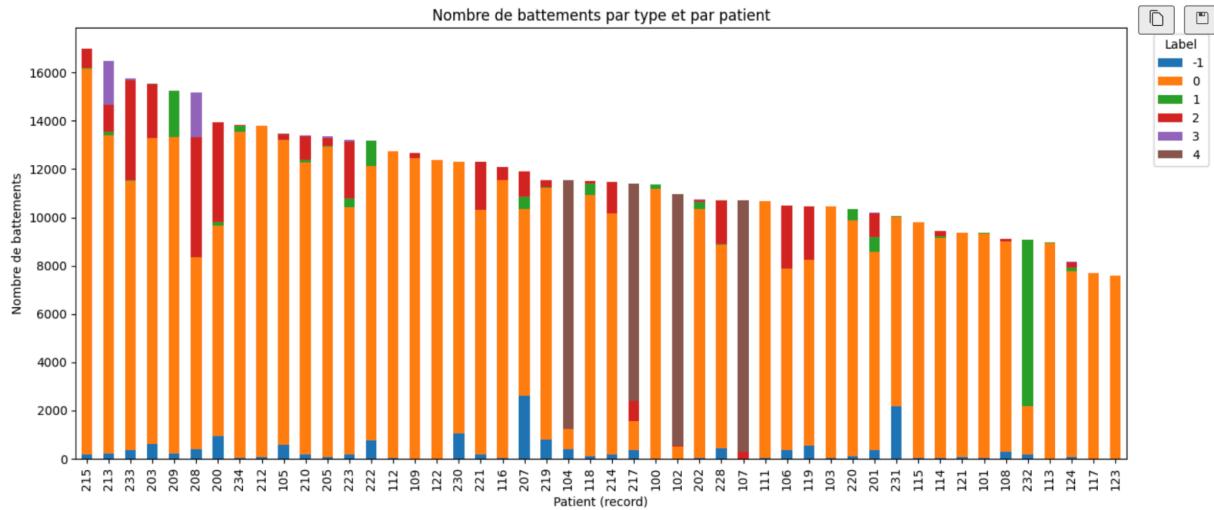
Conclusion : préparation de la base PTB pour la modélisation

À l'issue de cette phase exploratoire, plusieurs décisions se dégagent pour la préparation des données PTB :

- Conserver les variables techniques (fs, durée, nombre de signaux, dérivations...) et les variables cliniques bien renseignées (âge, sexe, diagnostics principaux, nombre de vaisseaux atteints...).
- Utiliser avec prudence les variables fortement incomplètes (certains paramètres hémodynamiques d’effort, dates secondaires, médications détaillées), plutôt à des fins descriptives.
- Travailler principalement sur un label binaire patient sain / patient malade,
- Garder à l'esprit le déséquilibre des classes (peu de patients sains, beaucoup de patients malades), qui influence le choix des métriques et des stratégies de validation.

## II. Preprocessing sur MITBIH

### 1. Première approche



Nous avons une base de données très déséquilibrée, la classe normale (0) étant largement majoritaire. Par ailleurs, nous avons une grosse variance de proportions de types de battements chez les patients.

Notre première approche est donc de réduire la classe majoritaire (0) pour rééquilibrer les classes, tout en essayant de garder la représentativité de chaque patient.

#### 1. Méthode:

a. On définit deux limites (par patient et par type de battement, pour éviter qu'un seul patient domine une classe, et par classe globale, pour garder un dataset final équilibré entre labels)

b. On compte le nombre de battements de chaque patient pour chaque type de battement.

```
group_counts = df.groupby(['record_num', 'label']).size().reset_index(name='count')
```

c. Grâce à la fonction sample et notre limite, on prend jusqu'à 300 battements maximum.

```
df_stratified = (df.groupby(['record_num', 'label'], group_keys=False).apply(sample_patient_label)
.reset_index(drop=True))
```

d. On limite chaque classe globale à notre deuxième limite

```
df_balanced = (df_stratified.groupby('label', group_keys=False)
               .apply(lambda x: x.sample(min(len(x), MAX_PER_LABEL_GLOBAL), random_state=42))
               .reset_index(drop=True))
```

e. On vérifie la répartition des labels:

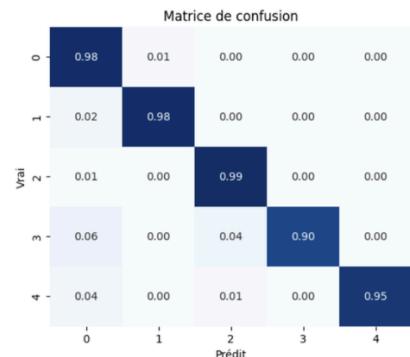
```
Répartition finale par label :  
label  
0    10000  
2     6585  
1     3595  
4     1275  
3     935
```

## 2. Résultats:

On a chacun travaillé sur ce jeu de données, en essayant d'optimiser une régression logistique, un SVC et un RandomForest..

Les résultats de ce dernier, sans trop d'efforts d'optimisation, nous ont fait supposer une fuite de données. En effet, nous obtenons un f1\_score de quasiment 1.

```
Meilleurs hyperparamètres : {'learning_rate': 0.1, 'max_depth': 8, 'n_estimators': 400}  
Score CV: 0.9650958558985566  
precision    recall   f1-score   support  
  
      0    0.98    0.98    0.98    2000  
      1    0.96    0.98    0.97    719  
      2    0.99    0.99    0.99    1311  
      3    0.98    0.90    0.94    187  
      4    1.00    0.95    0.97    135  
  
accuracy          0.98    4352  
macro avg       0.98    0.96    0.97    4352  
weighted avg     0.98    0.98    0.98    4352  
  
F1-score global: 0.9801717286742324
```



## 2. Deuxième approche

## 1. Introduction:

Au vu des résultats de notre précédent modèle, nous avons conclu à une fuite de données quelque part. Après réflexion, il se trouve que nous n'avons pas fait un split par patient avant toute chose, ce qui ne garantit pas la présence d'un même patient dans le jeu de données d'entraînement, et de test.

En effet, un patient a plusieurs lignes dans la base de données, le modèle ne reconnaît pas alors le signal, mais le patient, ce qui le favorise grandement, et explique les résultats de notre modèle.

## 2. Stratégies:

Nous allons utiliser le même modèle pour toutes les stratégies, afin de pouvoir décider de la meilleure stratégie à adopter.

Le modèle de base que nous allons utiliser est un XGBoost. C'est un modèle rapide et performant pour les grosses bases de données.

```
#####
## APPLICATION A XGBOOST SIMPLE ##

model = xgb.XGBClassifier(eval_metric="mlogloss")

model.fit(X_train, y_train)

y_pred_under_N = model.predict(X_test)
| 

print(classification_report(y_test, y_pred_under_N))
print("F1-score global:", f1_score(y_test, y_pred_under_N, average="weighted"))
```

La métrique que nous utilisons sera la f1\_macro, étant donné que nos classes sont très déséquilibrées.

Les stratégies que nous allons explorer sont:

- undersampling en respectant les proportions du dataset de base,
- undersampling en équilibrant toutes les classes sur la classe minoritaire,
- undersampling sur la classe majoritaire N seulement,

- une répartition par patients semblables (normaux, quasi normaux, un peu anormaux, et anormaux majoritairement),
- une stratégie où l'on respecte le standard AAMI-3 en fusionnant les classes 3 et 4 avec 0
- une stratégie où on classifie en binaire (normal/anormal)

### 3. Résultats:

La différence avec la première approche réside dans la séparation dès le début selon le patient. C'est à dire que l'on ne pourra pas, au hasard de la sélection `train_test_split`, retrouver une ligne d'un patient `x_102` par exemple dans le jeu de données `train`, et une autre ligne de ce même patient `x_102` dans le `test`.

Pour ce faire, nous allons récupérer le `record_num`, et faire un `train_test_split` sur le `record_num`, avant de récupérer ensuite les lignes correspondantes à `train_pat`, et `test_pat` pour séparer les datasets.

```
# Patients uniques:

patients = df["record_num"].unique()

# Split des patients avant tout pour éviter le data-leakage:

train_pat, test_pat = train_test_split(
    patients,
    test_size=0.2,
    random_state=42
)

# Récupération des bases de données train et test:

df_train = df[df["record_num"].isin(train_pat)].copy()
df_test = df[df["record_num"].isin(test_pat)].copy()
```

#### a. Undersampling en respectant les proportions:

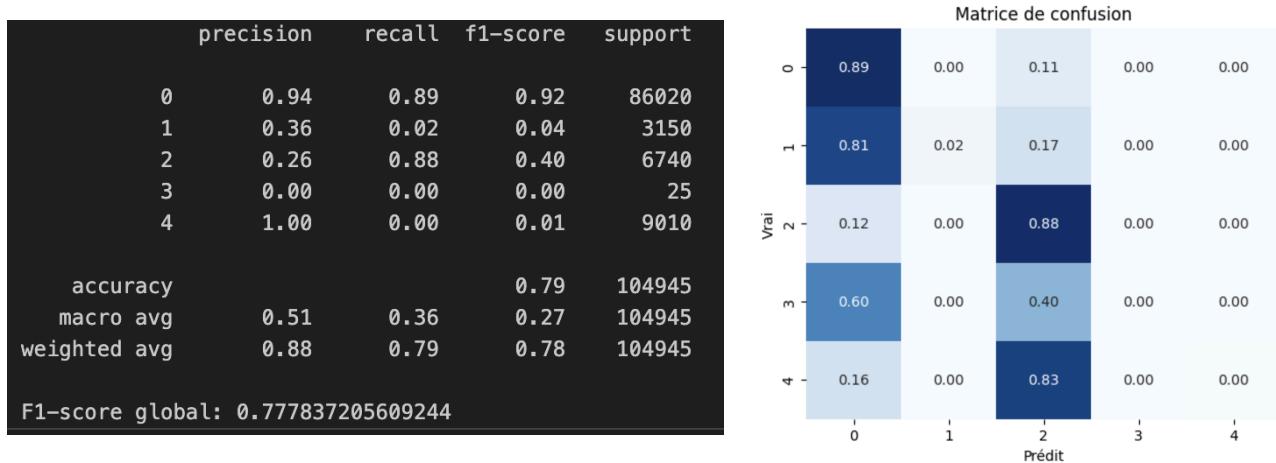
L'idée de cet undersampling est de sampler en respectant les proportions  $f$  de chaque label.

```

N = 20000
f = N / len(df_train)

train_under = (
    df_train.groupby("label", group_keys=False).sample(frac=f, random_state=42)
    .reset_index(drop=True)
)

```



## Résultats:

Classe 0	Il détecte 0,89 % des battements normaux, avec une très bonne précision!
Classe 1	Quasiment tous les battements classe 1 sont prédits en 0
Classe 2	Il détecte 88% des battements classe 2, mais en invente presque 3/4
Classe 3	Prédit à 60% en 1, à 40% en 2
Classe 4	Prédit en majorité en classe 2
f1_score	Bon

## b. Undersampling en équilibrant sur la classe minoritaire:

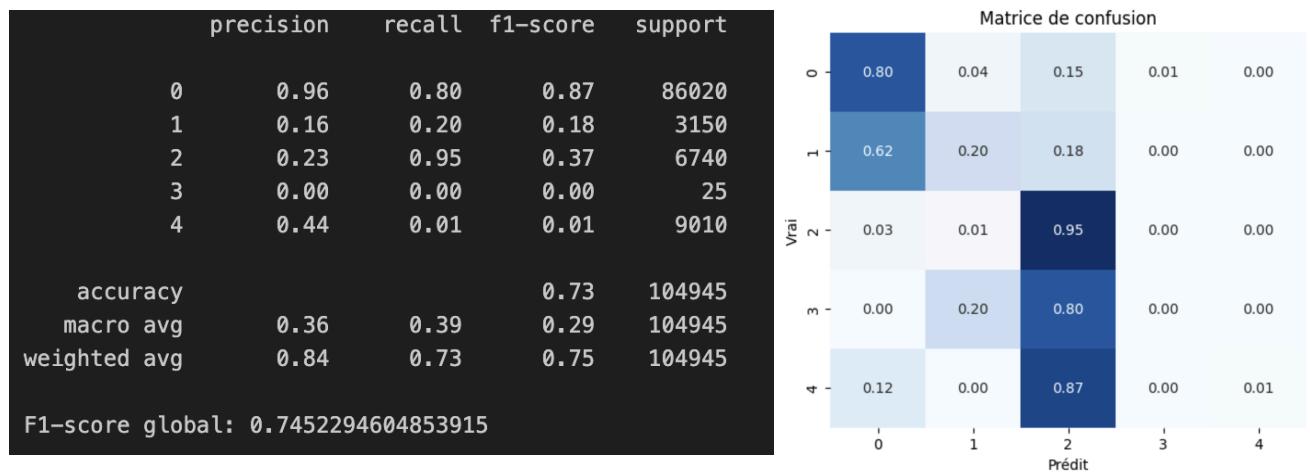
```

mini = df_train["label"].value_counts().min()

train_bal = (
    df_train.groupby("label", group_keys=False).sample(n=mini, random_state=42)
        .reset_index(drop=True))

```

L'idée de cet undersampling est de ramener toutes les proportions de label au nombre de lignes de la classe minoritaire (3695)



Classe 0	Il détecte 0,80 % des battements normaux, avec une très bonne précision!
Classe 1	Il commence à détecter la classe 1, mais avec une mauvaise précision.
Classe 2	Il détecte 95% des battements classe 2, mais en invente presque 3/4
Classe 3	Majoritairement prédict en 2
Classe 4	Majoritairement prédict en 2
f1_score	Bon

### c. Undersampling sur la classe majoritaire N seulement:

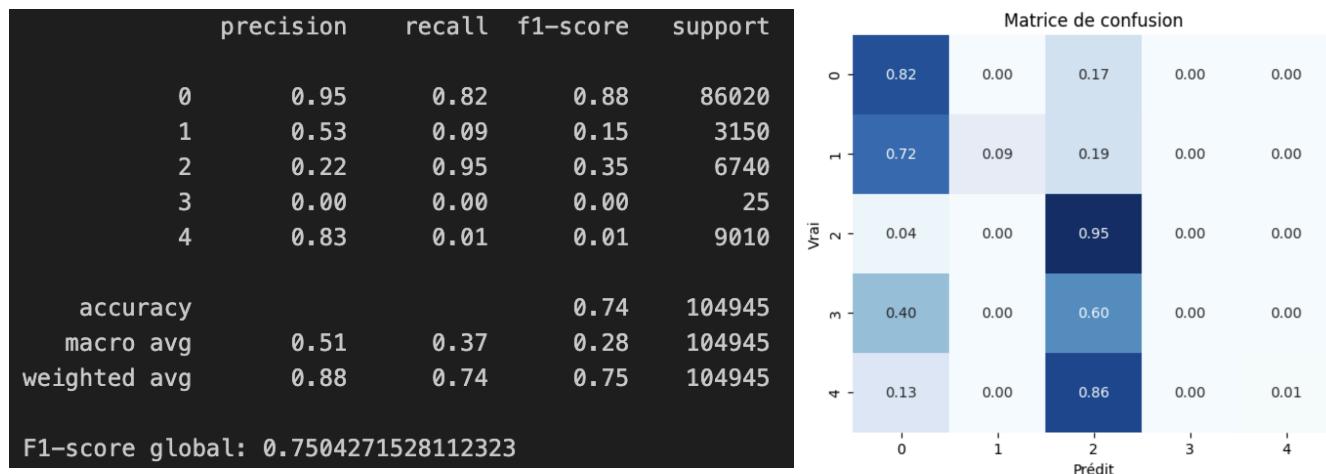
```

# Séparation N et les autres
df_N = df_train[df_train["label"] == MAJ_LABEL]
df_others = df_train[df_train["label"] != MAJ_LABEL]

# Échantillonnage 30 000 N aléatoirement
df_N_sampled = df_N.sample(n=30_000, random_state=42)

```

Après séparation des lignes label == 'N' et label !='N', on sample seulement les lignes ayant le label N, avant concaténer le df\_N\_sampled avec le df\_others (les autres labels).



Classe 0	Il détecte 0,82 % des battements normaux, avec une très bonne précision!
Classe 1	On perd la classe 1 qui fond dans la classe 0.
Classe 2	Il détecte 95% des battements classe 2, mais en invente presque 3/4
Classe 3	Majoritairement prédite en 2.
Classe 4	Majoritairement prédite en 2.
f1_score	Bon

#### d. Répartition par bin de patients semblables:

```

tab = df.groupby(['record_num', 'label']).size().unstack(fill_value=0)
tab['total'] = tab[[0,1,2,3,4]].sum(axis=1)

# ratios normal / anormal

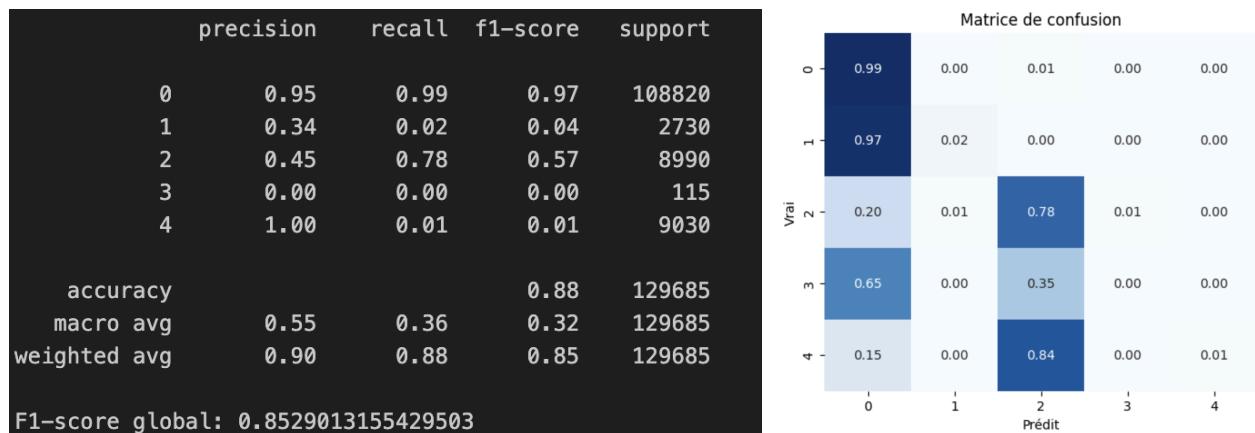
tab['N_ratio'] = tab[0] / tab['total'].replace(0, np.nan)
tab['abnormal_ratio'] = 1 - tab['N_ratio']

# type de patient

bins = [0, 0.01, 0.05, 0.2, 1.0]
labels = ['quasi_normaux', 'un_peu_anormaux', 'mixtes', 'tres_anormaux']
tab['type'] = pd.cut(tab['abnormal_ratio'], bins=bins, labels=labels)

```

L'idée de cette stratégie est de compartimenter les patients par caractéristique similaire, afin de ne pas se retrouver avec des patients quasi normaux dans le train, et que des cas anormaux dans le test, ou inversement. En effet, si le train ne « voit » que des patients sains, et que dans le test il y a des cas anormaux, il n'y a aucune chance que le modèle sache classifier ces signaux.

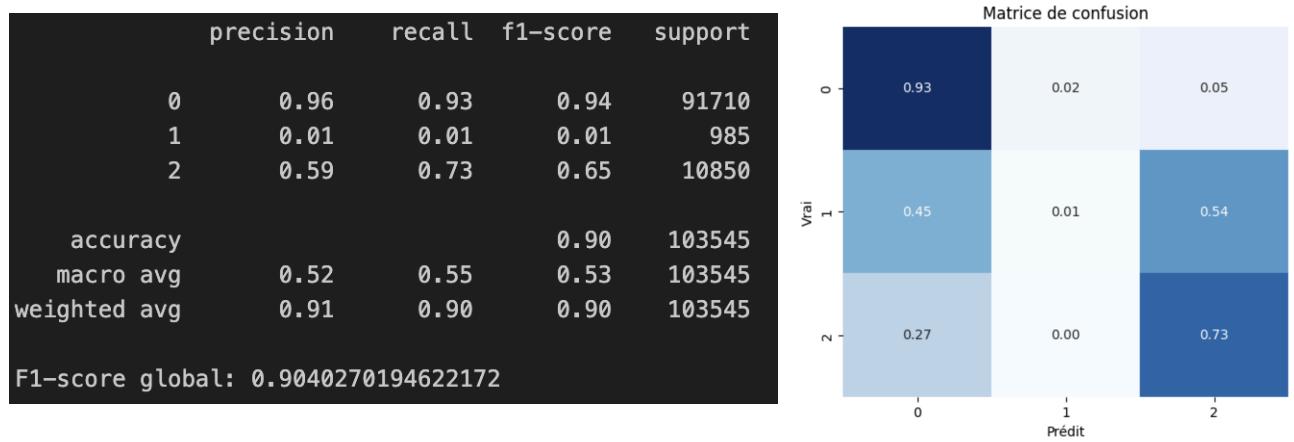


Classe 0	Il détecte 0,99 % des battements normaux, avec une très bonne précision!
Classe 1	Il perd la classe 1 qui est prédite en 0.
Classe 2	Il détecte un peu moins de battements classe 2, mais plus précis
Classe 3	La classe 3 est détectée enfin
Classe 4	Pas détectée.
f1_score	Bien meilleur

### e. Stratégie standard AAMI-3 en fusionnant les classes 3 et 4 avec 0:

L'idée est de respecter le standard AAMI-3 (Association for the Advancement of Medical Instrumentation), et de fusionner les classes 3 et 4 dans le label 0.

```
merge_map = {0: 0, 1: 1, 2: 2, 3: 0, 4: 0}
df_aami["label"] = df_aami["label"].map(merge_map)
```



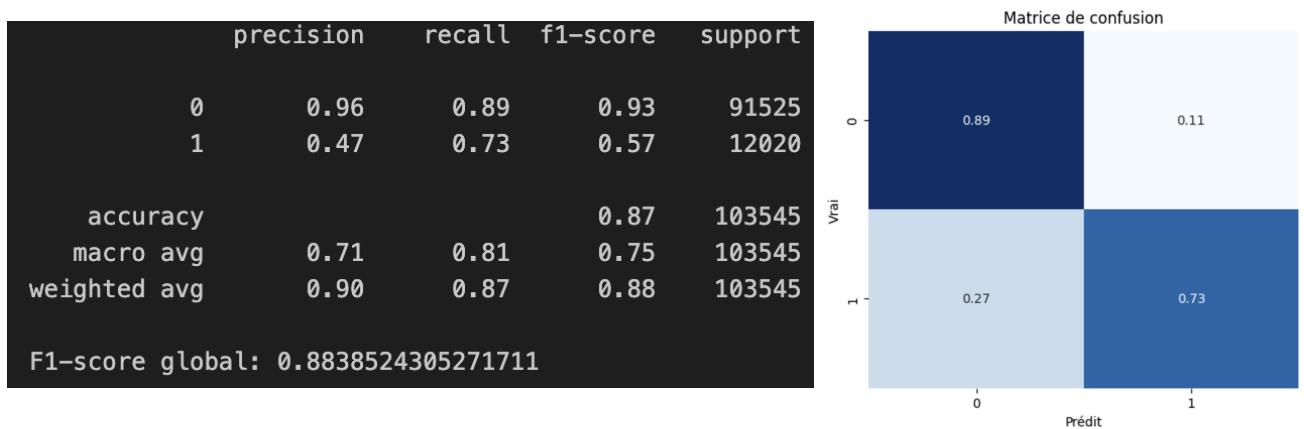
Classe 0	Il détecte 0,93 % des battements normaux, avec une très bonne précision!
Classe 1	La classe 1 disparaît dans la classe normale
Classe 2	Il détecte 73% de battements classe 2, et seulement 27% de faux négatifs.
f1_score	Bien meilleur

#### f. Stratégie de classification binaire (normal/anormal):

L'idée de cette stratégie, c'est de réduire notre problématique au strict minimum, à savoir de savoir si notre patient est malade ou pas. On cherchera la granularité dans un second temps.

```
df_binary["binary"] = (df_binary["label"] != 0).astype(int)
```

Ici, notre `y_train` ou `y_test` = 0 si le label == 0, et égal à 1 pour tous les autres labels.



Classe 0	Il détecte 0,89 % des battements normaux, avec une très bonne précision!
Classe 1	Il détecte 3 /4 des battements anormaux, avec 27% de faux négatifs.
f1_score	Bien meilleur

\*Dans le contexte médical, le fait qu'il invente beaucoup de faux positifs (classe 1 au lieu de classe 0) n'est pas gênant, on a un modèle sûr.

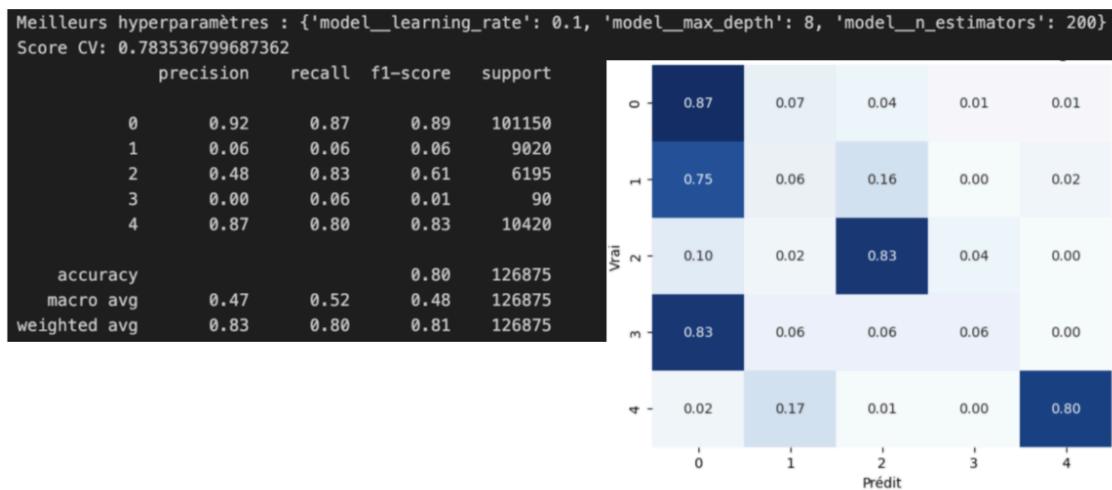
### 3. Conclusion

Nous avons donc testé plusieurs stratégies de construction de jeu de données (patient-wise, classification binaire, AAMI, classification par bin).

En multiclass, la meilleure stratégie est la classification par bin de patients semblables. Les meilleurs résultats sont obtenus avec un XGBoost + SMOTE sur cette stratégie.

Cependant, pour la suite de notre projet, nous avons décidé de réduire notre modélisation à une classification binaire (normal/anormal) pour pouvoir appliquer notre modèle sur le PTB.

Résultat multiclass XGBoost + SMOTE:



## III. Modélisation MITBIH binaire:

### 1. Préambule

Dans notre partie précédente concernant le pré-processing, nous sommes tombés sur la problématique du split patient-wise. En effet, un patient ayant plusieurs enregistrements (lignes), on risque de retrouver le même patient dans le train et dans le test en faisant juste appel à la fonction train-test-split.

Il en va de même pour l'optimisation de nos modèles en GridSearchCV. Si nous ne faisons pas attention à cette problématique, on risque de retrouver le même patient dans le train/test interne du cross-validation.

> Il faut donc utiliser GroupKFold pour s'assurer que la CV ne prenne pas le même patient dans le train et test interne.

Par ailleurs, étant donné le contexte médical, nous allons chercher à optimiser le recall sur la classe positive.

> On va donc utiliser le scoring ‘recall’ dans nos GridSearchCV.

Dans cette seconde partie, l’objectif est d’évaluer plusieurs approches de modélisation afin d’identifier celle qui offre les meilleures performances pour la classification des battements cardiaques à partir du dataset MIT-BIH.

Après avoir parcouru les données dans leur ensemble dans le grand I, nous mettons en œuvre plusieurs modèles issus aussi bien des méthodes statistiques classiques que de l’apprentissage automatique moderne.

L’ensemble des algorithmes testés couvre un spectre large :

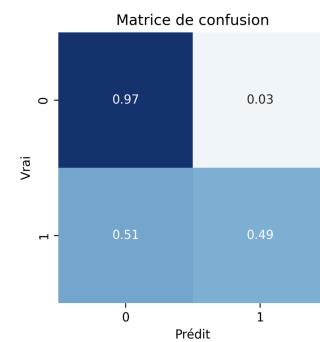
- **Régression logistique** (baseline)
- **XGBoost** (boosting gradientiel)
- **Random Forest** (modèle d’ensemble)
- **SVM** (max-margin classifier)
- **CNN** (modèle profond adapté aux signaux)
- **Réseau de neurones dense**

Chaque modèle est entraîné sur les données prétraitées, puis évalué selon des métriques standards (accuracy, F1-score, matrice de confusion), afin de comparer leur capacité à distinguer les différentes classes de battements cardiaques.

## 2. Régression Logistique

### 1. Baseline:

		precision	recall	f1-score	support
	0	0.94	0.97	0.96	20831
	1	0.67	0.49	0.56	2450
accuracy				0.92	23281
macro avg		0.80	0.73	0.76	23281
weighted avg		0.91	0.92	0.91	23281

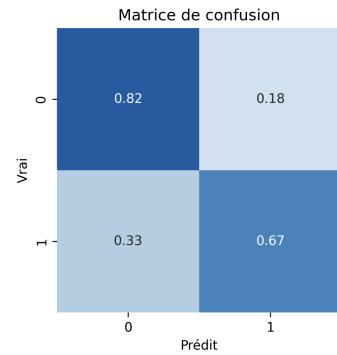


Notre modèle de régression logistique en baseline présente une bonne détection des battements normaux, mais présente une grande quantité de faux négatifs, ce qui est problématique dans notre contexte médical.

## 2. Modèle optimisé:

Nous avons réalisé plusieurs modèles de régression logistique, avec différents solvers (LBFGS, SAGA, ELASTICNET), et ils sont équivalents en termes de résultat. Ici le résultat d'une GridSearchCV sur un modèle de régression logistique Solver SAGA avec correction ElasticNet. Les hyperparamètres utilisés sont le coefficient de régularisation C, et le L1\_ratio de régularisation Ridge et Lasso.

	precision	recall	f1-score	support
0	0.96	0.82	0.88	20831
1	0.30	0.67	0.42	2450
accuracy			0.80	23281
macro avg	0.63	0.75	0.65	23281
weighted avg	0.89	0.80	0.83	23281

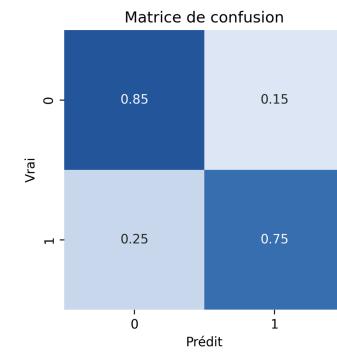


On observe que le modèle gagne en précision sur la classe positive 1, il y a seulement 33% de faux négatifs. Comme on pouvait s'y attendre, le modèle de régression logistique n'est pas très efficace sur notre projet.

## 3. XGBoost

### 1. Baseline:

	precision	recall	f1-score	support
0	0.97	0.85	0.91	20831
1	0.37	0.75	0.50	2450
accuracy			0.84	23281
macro avg	0.67	0.80	0.70	23281
weighted avg	0.90	0.84	0.86	23281



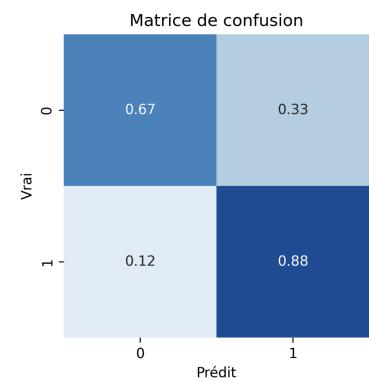
On observe que notre modèle baseline XGBoost détecte très bien les classes normales. Le taux de faux négatifs descend à 0,25%, ce qui est une bonne base de travail pour l'optimisation.

## 2. Modèle optimisé:

Nous avons réalisé plusieurs GridSearchCV sur le modèle XGBoost. Les hyperparamètres utilisés sont le learning\_rate, le max\_depth et le n\_estimators. Nous utilisons l'objective binary:logistic puisqu'il s'agit d'une classification binaire. On a aussi réalisé des modélisations avec oversampling SMOTE et ADASYN.

Le meilleur résultat a été obtenu avec ADASYN.

	precision	recall	f1-score	support
0	0.98	0.67	0.80	20831
1	0.24	0.88	0.38	2450
accuracy			0.69	23281
macro avg	0.61	0.78	0.59	23281
weighted avg	0.90	0.69	0.75	23281



On observe une chute spectaculaire du taux de faux positif, au dépend de la classe 0. Dans notre contexte, c'est le résultat le plus intéressant.

## 4. Random Forest

### 1. Baseline :

RandomForest — Classification report

	precision	recall	f1-score	support
0	0.95	0.92	0.94	20831
1	0.47	0.59	0.53	2450
accuracy			0.89	23281
macro avg	0.71	0.76	0.73	23281
weighted avg	0.90	0.89	0.89	23281

#### Résumé

Seuil : 0.50

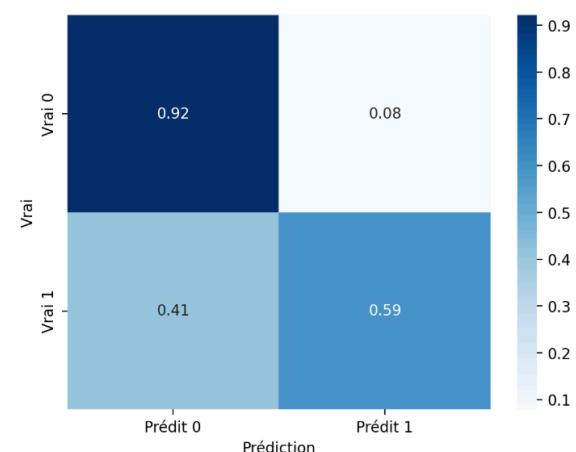
Accuracy globale : 0.887

Precision classe 1 : 0.471

Recall classe 1 : 0.593

F1 classe 1 : 0.525

RandomForest — Matrice de confusion



Notre modèle Random Forest en baseline présente une bonne détection des battements normaux avec une précision élevée 0.95, Le modèle produit peu de faux négatifs sur la classe 1 (ce qui est positif), mais génère encore un nombre important de faux positifs..

## 2. Modèle optimisé :

Pour améliorer les performances du modèle Random Forest, nous avons effectué une **RandomizedSearchCV**, permettant d'explorer un large espace d'hyperparamètres (`n_estimators`, `max_depth`, `min_samples_split`, `bootstrap`...) sans tester exhaustivement toutes les combinaisons. L'évaluation est réalisée avec un **GroupKFold** à 3 splits afin d'éviter toute fuite de données entre les patients.

Baseline RandomForest

Baseline RandomForest — Classification report

	precision	recall	f1-score	support
0	0.97	0.89	0.93	20831
1	0.45	0.75	0.56	2450
accuracy			0.88	23281
macro avg	0.71	0.82	0.75	23281
weighted avg	0.91	0.88	0.89	23281

### Résumé

Seuil : 0.50

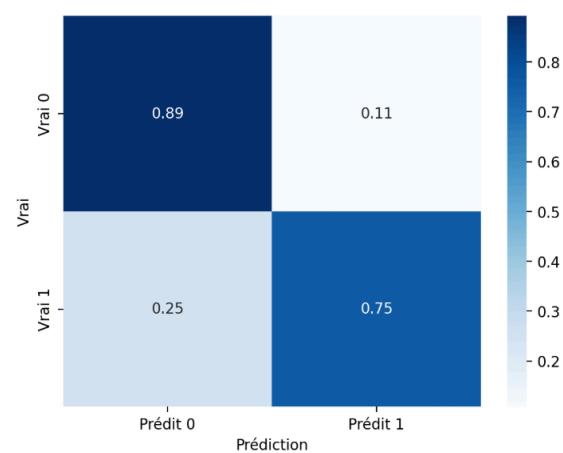
Accuracy globale : 0.878

Precision classe 1 : 0.451

Recall classe 1 : 0.753

F1 classe 1 : 0.564

Baseline RandomForest — Matrice de confusion



Le modèle optimisé montre une amélioration significative du **recall** sur la classe 1 0.75 vs 0.59 avant, ce qui signifie qu'il détecte beaucoup plus efficacement les battements anormaux.

En contrepartie, la performance sur la classe 0 diminue, entraînant davantage de faux positifs.

## 5. SVM

Dans notre projet, la modélisation SVM correspond à l'utilisation d'un Support Vector Machine pour apprendre, à partir des signaux ECG, une frontière de décision qui distingue au mieux les battements normaux des battements anormaux, puis à l'évaluer grâce aux métriques (accuracy, recall, F1, matrices de confusion) et à l'utiliser pour prédire la classe de nouveaux battements.

Pour le SVM, on a d'abord construit un pipeline simple : normalisation des 187 points avec StandardScaler, puis un SVC à noyau RBF.

Les hyperparamètres à régler sont le C et le γ (gamma).

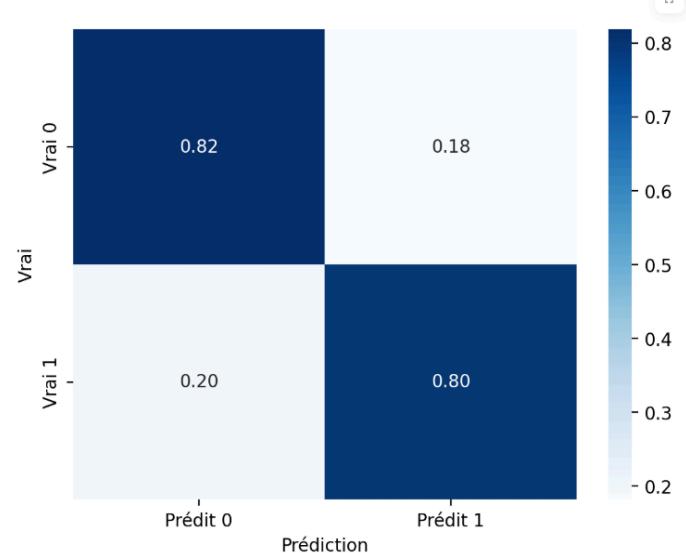
Pour ne pas mélanger les patients entre apprentissage et validation, on a utilisé une validation croisée groupée avec `GroupKFold(n_splits=3)` en groupant par `record_x`. À chaque fold, certains patients servent au train, d'autres au validation, mais jamais les mêmes.

Sur cette base, on a lancé un `GridSearchCV` qui teste toutes les combinaisons de `C` et `y` et choisit celle qui maximise le score F1 de la classe 1 (anormal). Le meilleur pipeline est ensuite ré-entraîné sur tout le train avant d'être évalué sur le test.

### **Modèle optimisé:**

	precision	recall	f1-score	support
0	0.97	0.82	0.89	20831
1	0.34	0.80	0.48	2450
accuracy			0.82	23281
macro avg	0.66	0.81	0.68	23281
weighted avg	0.91	0.82	0.85	23281

SVM RBF — Matrice de confusion



Sur la figure, le classification report montre :

- une accuracy globale d'environ 0,82 ;
- pour la classe normale (0) : bonne précision (0,97) et bon rappel (0,82) ;
- pour la classe anormale (1) : rappel élevé (0,80, le modèle détecte bien les anormaux) mais précision faible (0,34, beaucoup de faux positifs).

La matrice de confusion normalisée confirme :

82 % des battements vraiment normaux sont bien classés, 18 % sont pris à tort pour anormaux ;  
80 % des battements vraiment anormaux sont bien détectés, 20 % sont ratés.

En résumé, le SVM est configuré pour ne pas rater les battements anormaux, au prix d'un surplus d'alertes fausses.

## 6. CNN

### 1. Baseline :

CNN Keras 1D (retrained) — Classification report

	precision	recall	f1-score	support
0	0.96	0.85	0.90	20831
1	0.36	0.73	0.49	2450
accuracy			0.84	23281
macro avg	0.66	0.79	0.70	23281
weighted avg	0.90	0.84	0.86	23281

#### Résumé

Seuil : 0.50

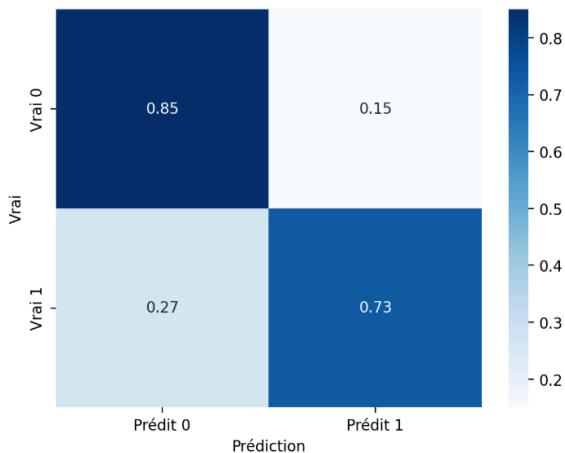
Accuracy globale : 0.838

Precision classe 1 : 0.365

Recall classe 1 : 0.733

F1 classe 1 : 0.487

CNN Keras 1D (retrained) — Matrice de confusion



Le modèle CNN baseline constitue notre première architecture convolutionnelle appliquée aux signaux ECG.

Malgré une structure relativement simple (quelques blocs convolution + pooling), le modèle parvient déjà à bien distinguer les battements normaux des battements anormaux.

On observe un très bon score sur la classe 0, avec une précision élevée, mais un recall encore limité sur la classe 1 (0.73). Cela signifie que le modèle identifie correctement la majorité des battements normaux, mais manque encore certaines anomalies.

## 2. Modèle optimisé :

Le modèle optimisé repose sur un **CNN 1D** composé de trois blocs convolutionnels (32, 64 et 128 filtres), chacun suivi d'une **Batch Normalization** et de couches de réduction dimensionnelle (*MaxPooling* puis *GlobalAveragePooling*).

La partie dense inclut une couche fully connected de 64 neurones avec **Dropout (0.3)** avant la sortie sigmoïde.

Le modèle est entraîné avec l'optimiseur **Adam (1e-3)** et la perte **binary\_crossentropy**, en suivant les métriques **accuracy** et **AUC**.

Baseline CNN Keras 1D — Classification report

	precision	recall	f1-score	support
0	0.96	0.85	0.90	20831
1	0.36	0.73	0.49	2450
accuracy			0.84	23281
macro avg	0.66	0.79	0.69	23281
weighted avg	0.90	0.84	0.86	23281

#### Résumé

Seuil : 0.50

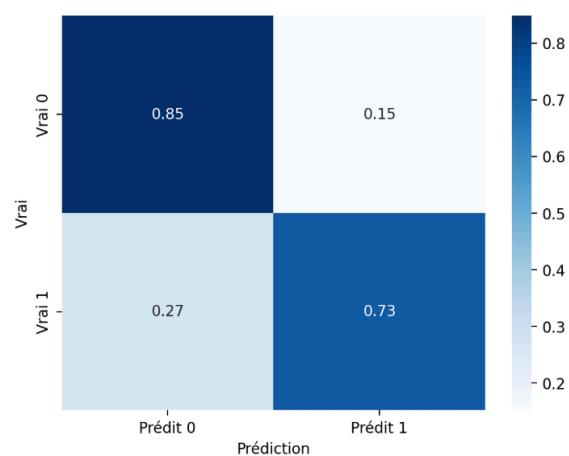
Accuracy globale : 0.836

Precision classe 1 : 0.363

Recall classe 1 : 0.734

F1 classe 1 : 0.485

Baseline CNN Keras 1D — Matrice de confusion



On peut voir que même après optimisation, les métriques restent relativement faibles par rapport aux autres modèles et surtout par rapport au modèle suivant de NN.

## 7. NN

### 1. Baseline :

Le modèle NN baseline repose sur une architecture simple entièrement connectée, composée de plusieurs couches denses activées en ReLU et d'une sortie sigmoïde. Cette première version parvient à capter les tendances globales des signaux ECG, mais reste limitée dans la détection fine des battements anormaux, avec un recall encore insuffisant sur la classe 1.

Elle constitue néanmoins une bonne base pour mesurer le gain obtenu avec une architecture plus profonde et régularisée.

Baseline NN Keras dense — Classification report

	precision	recall	f1-score	support
0	0.99	0.81	0.89	20831
1	0.35	0.90	0.51	2450
accuracy			0.82	23281
macro avg	0.67	0.85	0.70	23281
weighted avg	0.92	0.82	0.85	23281

#### Résumé

Seuil : 0.50

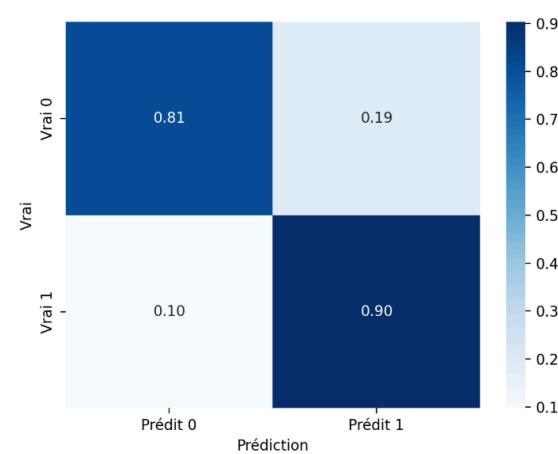
Accuracy globale : 0.817

Precision classe 1 : 0.354

Recall classe 1 : 0.903

F1 classe 1 : 0.509

Baseline NN Keras dense — Matrice de confusion



### 2. Modèle optimisé :

Le modèle optimisé repose sur un réseau de neurones entièrement connecté composé de deux couches denses (128 puis 64 neurones), chacune suivie d'une Batch Normalization et d'un Dropout (0.3) pour limiter l'overfitting.

La sortie sigmoïde permet la classification binaire.

Le modèle est entraîné avec Adam (1e-3) et la perte binary\_crossentropy, en suivant les métriques **accuracy** et **AUC**.

NN Keras dense (retrained) — Classification report

	precision	recall	f1-score	support
0	0.98	0.84	0.90	20831
1	0.39	0.89	0.54	2450
accuracy			0.84	23281
macro avg	0.69	0.86	0.72	23281
weighted avg	0.92	0.84	0.87	23281

#### Résumé

Seuil : 0.50

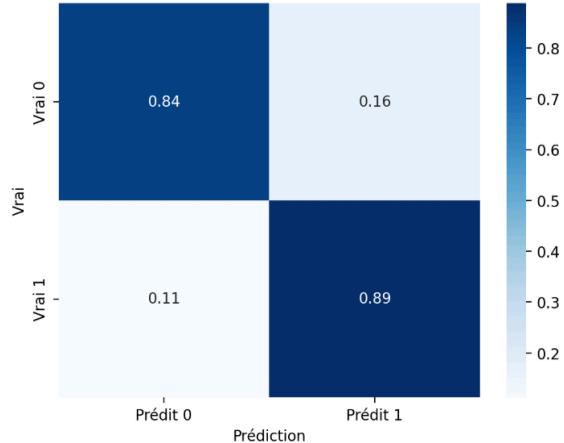
Accuracy globale : 0.842

Precision classe 1 : 0.390

Recall classe 1 : 0.888

F1 classe 1 : 0.542

NN Keras dense (retrained) — Matrice de confusion



Après optimisation, les performances sont nettement améliorées, notamment sur la **classe malade (1)** :

Accuracy globale : ~0.84%

Recall classe 1 : ~0.89

Ces résultats montrent que le réseau parvient à détecter efficacement les battements anormaux, avec un recall élevé, ce qui réduit fortement le risque de faux négatifs.

Même si la précision reste perfectible (quelques faux positifs), le compromis est très satisfaisant pour une tâche clinique où la priorité est d'identifier les anomalies.

Grâce à son excellente capacité de détection des malades, sa stabilité et sa rapidité d'entraînement, ce NN fait partie des modèles retenus pour la suite de l'étude, aux côtés du CNN.

## 8. Conclusion

Pour l'extrapolation des modèles entraînés sur MIT-BIH vers PTB, on s'est concentré sur la capacité à détecter les battements anormaux, donc sur le recall de la classe 1. Dans notre contexte de dépistage cardiaque, rater un patient malade (faux négatif) est beaucoup plus grave que générer des fausses alertes. C'est pourquoi le recall (sensibilité) est notre critère principal, l'accuracy ou la précision venant seulement en second.

Scores triés par recall									
	model_id	model_file	framework	type	n_test	accuracy	f1	precision	recall
3	xgb_mit_binary	XGB_mit_binary.pkl	sklearn	classique	49028	0.4481	0.473	0.3106	0.9906
4	nn_mit_binary	nn_mit_mlII_binary.keras	keras_dense	deep	49028	0.981	0.9625	0.9506	0.9746
5	cnn_mit_binary	cnn_mit_mlII_binary_best.keras	keras_cnn	deep	49028	0.9724	0.9449	0.9443	0.9454
0	logreg_mit_binary	LogReg_mit_binary.pkl	sklearn	classique	49028	0.6941	0.5969	0.445	0.9059
1	svm_mit_binary	svm_mit_mlII_binary.pkl	sklearn	classique	49028	0.6422	0.5554	0.4029	0.8939
2	rf_mit_binary	rf_mit_mlII_binary.pkl	sklearn	classique	49028	0.7414	0.6173	0.4899	0.8342

En regardant le tableau de synthèse, les modèles (LogReg, SVM, Random Forest) ont un recall autour de **0,83–0,91**, ce qui est correct mais pas exceptionnel pour un problème médical. Les deux modèles qui se détachent sont :

- **XGBoost** : recall ≈ **0,99**, mais avec une précision faible (~0,31) et une accuracy moyenne (~0,45).
- **NN dense** : recall ≈ **0,97**, avec d'excellents scores globaux (accuracy ≈ **0,98**, F1 ≈ **0,96**, précision ≈ **0,95**).

On choisit donc de tester **XGBoost** et le **NN dense** sur PTB parce qu'ils maximisent la détection des battements anormaux (critère vital dans notre use-case médical), tout en couvrant deux approches complémentaires : arbre de décision boosté et deep learning.

### III Extrapolation(PTB)

Dans une seconde étape, nous avons cherché à extrapoler les modèles entraînés sur MIT-BIH vers une base externe plus réaliste : la base PTB Diagnostic ECG Database. L'idée est de vérifier si un modèle appris sur des battements annotés au niveau « rythme / arythmie » (MIT-BIH) reste pertinent lorsqu'on le transpose à un contexte clinique cardiaque différent, avec d'autres patients, d'autres pathologies et un protocole d'enregistrement plus complexe. Cette partie décrit d'abord la stratégie de labellisation des patients PTB, puis la façon dont les modèles MIT (XGBoost et réseau de neurones dense) sont appliqués aux battements PTB et agrégés au niveau patient/visite.

#### 1) Labellisation des patients

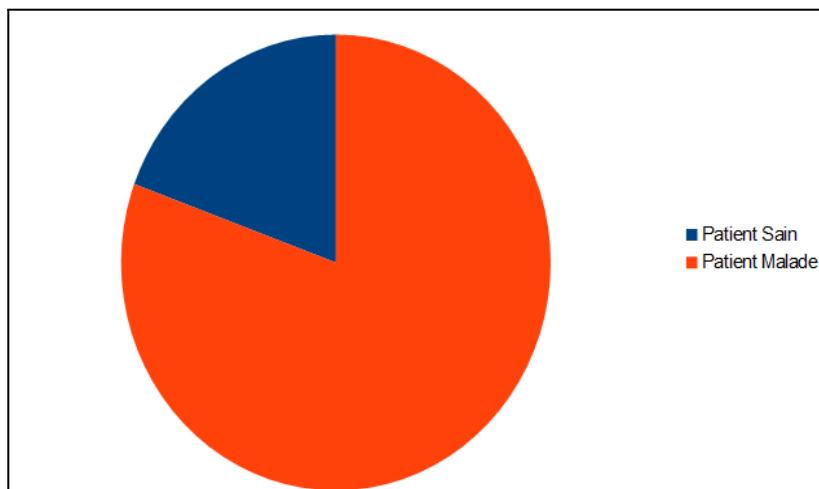
Le jeu de données PTB ne fournit pas directement un label binaire « sain / malade » au niveau des battements. Nous avons donc commencé par labelliser

Diagnostic class	Number of subjects
Myocardial infarction	148
Cardiomyopathy/Heart failure	18
Bundle branch block	15
Dysrhythmia	14
Myocardial hypertrophy	7
Valvular heart disease	6
Myocarditis	4
Miscellaneous	4
Healthy controls	52

les patients à partir des métadonnées cliniques (diagnostics, antécédents d'infarctus, raisons d'admission, etc.).

- Les patients décrits comme *healthy control* ou sans pathologie cardiaque documentée sont labellisés sains (*healthy\_label* = 1).
- Les patients présentant un infarctus du myocarde (aigu ou ancien) ou une autre pathologie cardiaque significative sont labellisés malades (*healthy\_label* = 0).

Ce label patient est ensuite propagé à l'ensemble des battements extraits pour PTB, ce qui nous permet d'obtenir un fichier de battements PTB annotés



(`ptb_beats_all_with_healthy_label.csv`) compatible avec les modèles entraînés sur MIT-BIH.

## 2) Application du modèle

À partir de ce fichier de battements PTB labellisés, nous appliquons les modèles sélectionnés sur MIT-BIH (XGBoost et réseau de neurones dense) en plusieurs étapes :

### 1. Préparation des features

On récupère les 187 points du battement (b0...b186), on remplace les valeurs manquantes et on applique la **même normalisation** que sur MIT-BIH car le scaler a été sauvegardé

### 2. Prédiction au niveau battement

Une fois les battements PTB normalisés au même format que MIT-BIH (vecteurs de 187 points), ils sont passés un par un dans le modèle choisi :

- avec XGBoost, on utilise `predict_proba` pour obtenir, pour chaque battement, une probabilité d'être « anormal / malade » ;
- avec le réseau de neurones dense, on utilise `model.predict`, qui renvoie également une probabilité comprise entre 0 et 1.

On obtient ainsi, pour chaque battement PTB, un score continu  $p(\text{malade})$  qui sert de base à toutes les décisions ultérieures.

## Cas particulier du NN : calibration + choix de seuil

Pour le réseau de neurones, on ajoute deux briques spécifiques dans un script dédié :

### 1. Calibration isotone des probabilités

- On sélectionne environ 20 % des patients PTB (échantillon de calibration).
- On ajuste un modèle d'Isotonic Regression qui apprend une fonction monotone  $f$  telle que  $p(\text{calibre})=f(p(\text{brut}))$
- Cette étape corrige le fait que les probabilités brutes du NN peuvent être trop optimistes ou pessimistes : après calibration, un battement annoncé à 0,8 a réellement ~80 % de chances d'être malade sur les données de calibration.

### 2. Choix d'un seuil orienté précision (logique anti-faux-positifs)

- On balaye un ensemble de seuils possibles (par exemple de 0,01 à 0,99).
- Pour chaque seuil, on binarise la probabilité en prédiction 0/1 et on calcule précision, recall et F1 sur les battements labellisés.
- On retient le seuil le plus élevé qui :
  - respecte une précision minimale cible (par exemple  $\geq 0,90$ , donc très peu de faux positifs),
  - tout en gardant un score F1 correct pour ne pas sacrifier totalement le rappel.
- Si aucun seuil n'atteint la précision cible, on prend en secours le seuil qui maximise simplement le F1.

Cette logique « précision d'abord » permet de limiter autant que possible les faux positifs au niveau battement. C'est important pour la suite : lors de l'agrégation au niveau patient/visite, quelques battements faussement malades pourraient sinon suffire à classer un patient sain comme malade.

### 3. Prédiction au niveau patient

Une fois que chaque battement PTB est étiqueté par le modèle (XGBoost ou NN), on passe à l'échelle clinique qui nous intéresse vraiment : le patient (ou la visite patient-record).

#### Agrégation des battements

1. On regroupe les lignes du fichier de battements par :
  - patient + record pour une prédiction par visite,

- ou uniquement patient si l'on veut une prédiction globale par individu.
2. Pour chaque groupe, on calcule :
    - le nombre total de battements `n_beats`,
    - le nombre de battements prédis malades `n_pred_malade`,
    - le label de référence patient `healthy_label` (1 = sain, 0 = malade) hérité des métadonnées PTB.
  3. On en déduit la vérité terrain au niveau patient :  
(0 = patient sain, 1 = patient malade).

### Règle de décision patient

Pour transformer ces informations en un diagnostic patient, on applique une règle simple mais contrôlable :

- on choisit un seuil sur le nombre de battements anormaux prédis par le modèle ;
- dans nos scripts, la règle de base est :  
 $\text{pred\_malade\_patient}=1(\text{n\_pred\_malade} \geq 1)$   
 Autrement dit, un patient est prédi malade dès qu'au moins un battement est classé anormal.

Cette règle est cohérente avec notre objectif de ne pas rater de patients réellement malades: dès qu'un battement est flippé comme suspect par le modèle, la visite bascule du côté « malade ». Le seuil peut être augmenté (par exemple  $\geq 3$  battements anormaux) si l'on veut durcir la décision et limiter les faux positifs.

### Évaluation et fichiers générés

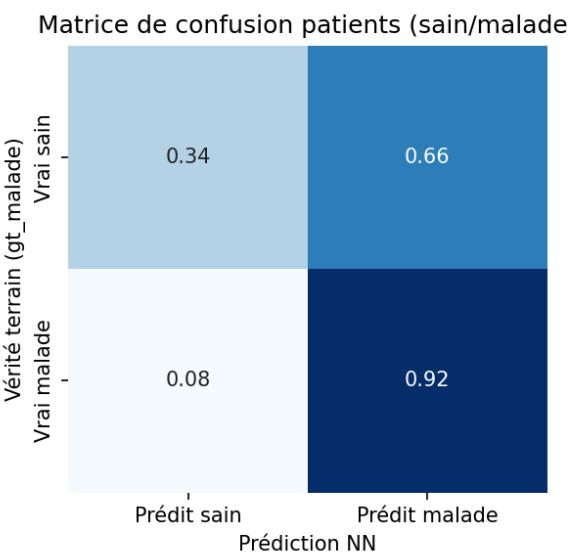
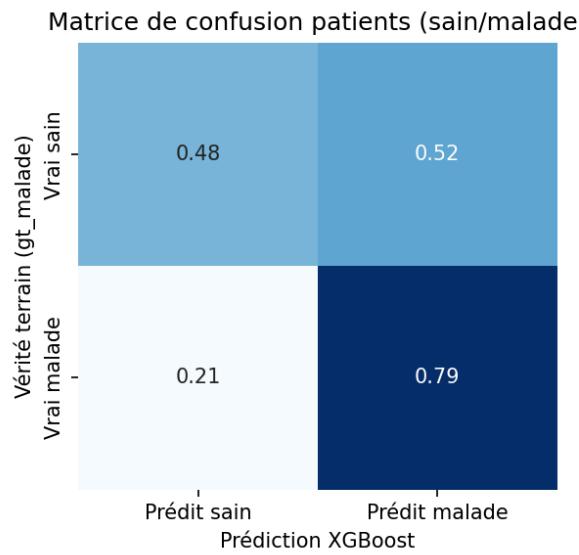
Pour chaque modèle (XGBoost et NN), on produit alors :

- un CSV patient (ou visite) contenant, pour chaque ligne :
  - `patient`, `record`, `n_beats`, `n_pred_malade`,
  - `healthy_label`, `gt_malade`, `pred_malade`,
  - et les versions inversées `gt_healthy`, `pred_healthy` pour lecture clinique;
- une **matrice de confusion patient** (`gt_malade` vs `pred_malade`) qui donne les taux de :
  - vrais malades détectés,
  - patients sains correctement laissés « sains »,
  - faux négatifs (malades non détectés)
  - et faux positifs (sains classés malades).

Cette étape de prédiction au niveau patient est donc l'aboutissement de l'extrapolation MIT → PTB : les modèles entraînés sur les battements de MIT-BIH sont utilisés pour proposer un

**diagnostic binaire sain/malade par patient PTB**, à partir du comportement global de leurs battements.

## 4. Résultats au niveau patient



Les matrices de confusion montrent que les deux modèles détectent bien les patients malades, mais avec un coût en faux positifs.

- XGBoost (à gauche, niveau patient)
  - Environ 79 % des patients malades sont correctement identifiés (case « Vrai malade / Prédit malade »).
  - En revanche, 52 % des patients sains sont tout de même classés « malades ».  
→ Le modèle est sensible, mais il déclenche beaucoup d'alertes à vérifier.
- NN (à droite, niveau visite)
  - La détection des malades est encore meilleure : autour de 92 % des patients malades sont correctement repérées.
  - Le prix à payer est un taux très élevé de faux positifs : 66 % des visites saines sont signalées comme malades.

Globalement, les modèles réussissent à ne pas rater les patients pathologiques (bon recall), ce qui correspond à notre priorité clinique. En revanche, ils doivent être vus comme des outils de présélection : ils identifient une population « à risque » qui nécessite ensuite une relecture et une confirmation par le cardiologue.

# Conclusion

Dans ce projet HeartBeat, nous avons mis en place une chaîne complète de data science appliquée aux signaux ECG, depuis le text mining des fichiers bruts MIT-BIH et PTB jusqu'à la modélisation et l'extrapolation clinique. La première étape a consisté à structurer des bases hétérogènes en construisant des jeux de données propres et harmonisés : extraction automatisée des métadonnées (.hea), génération de segments de battements fenêtrés à 187 points, normalisation des signaux et alignement des fréquences d'échantillonnage. Ce travail de préparation a permis d'obtenir des CSV exploitables, à la fois au niveau battement (MIT-BIH et PTB) et au niveau patient/visite (PTB), tout en maîtrisant les problèmes classiques de données manquantes, d'artefacts de mesure et de déséquilibre des classes.

Sur MIT-BIH, nous avons étudié en détail la distribution des types de battements, l'impact de l'âge, du sexe et des leads, puis conçu plusieurs stratégies de préprocessing pour limiter les biais : suppression de certains enregistrements problématiques, exclusion du lead V5, split patient-wise, et différentes formes d'undersampling. Ces jeux de données ont servi de base à une comparaison systématique de modèles : régression logistique, Random Forest, XGBoost, SVM, réseau de neurones dense et CNN. En classification binaire normal/anormal, les meilleurs compromis entre sensibilité et performance globale sont obtenus par XGBoost et le réseau de neurones dense, qui maximisent le rappel sur la classe malade tout en restant suffisamment robustes pour une utilisation de dépistage.

Dans un second temps, nous avons extrapolé ces modèles vers la base PTB, beaucoup plus clinique et réaliste. Après labellisation des patients (sains vs malades) à partir des diagnostics et métadonnées, les modèles entraînés sur MIT-BIH ont été appliqués aux battements PTB pour générer des scores de probabilité, ensuite agrégés au niveau patient/visite. Les résultats montrent que XGBoost et le NN dense détectent une grande majorité de patients malades, au prix d'un nombre non négligeable de faux positifs. Dans cette optique, les modèles doivent être vus comme des outils d'aide au tri et à la priorisation : ils permettent d'identifier une population « à risque » qui nécessitera une relecture experte, plutôt que comme des systèmes de décision autonomes.

Enfin, nous avons volontairement travaillé avec des modèles de machine learning classiques (XGBoost, SVM, Random Forest, NN dense, CNN simple), offrant un bon compromis entre performance, lisibilité du pipeline et contraintes matérielles. L'état de l'art sur les bases MIT-BIH et surtout PTB-XL montre toutefois que les meilleures performances sont aujourd'hui obtenues avec des architectures profondes plus avancées (CNN 1D de type ResNet/Inception, modèles hybrides CNN–Transformer, modèles pré-entraînés en self-supervised sur de très grandes bases ECG). Ces approches dépassent probablement nos résultats, mais au prix d'une complexité algorithmique et d'un coût de calcul bien

supérieurs. Notre travail fournit ainsi une base solide, reproductible et bien documentée, sur laquelle il sera naturel, dans de futurs développements, d'explorer ces architectures de pointe, d'améliorer la calibration des probabilités et de renforcer le lien avec la pratique clinique réelle.