Rapport Challenge IMA205

Cardiac Pathology Prediction

Mohamed MOHAMED EL BECHIR

Encadrant :
Pietro GORI
Loïc LE FOLGOC

4 mai 2025

Table des matières

1 Introduction

Ce projet s'inscrit dans le cadre du Challenge IMA205, qui vise à développer un pipeline complet de classification de pathologies cardiaques à partir d'IRM cardiaques. Le jeu de données contient 150 patients répartis en cinq classes diagnostiques, avec pour chacun deux IRM 3D acquises en end-diastole (ED) et en end-systole (ES), ainsi que des segmentations partielles des cavités cardiaques.

Un des principaux enjeux de ce challenge est l'absence de la segmentation du ventricule gauche (VG) dans les données de test. Cette segmentation est pourtant essentielle pour extraire des caractéristiques comme les volumes ventriculaires ou les fractions d'éjection, qui sont des indicateurs clés pour différencier les pathologies. Il a donc fallu mettre en place une méthode pour reconstituer cette segmentation manquante, afin de pouvoir exploiter les données de test de manière cohérente avec les données d'entraînement.

Le travail a été structuré en trois grandes étapes, chacune développée dans un notebook spécifique. La première étape, développée dans le notebook *VG_Reconstruction*, a consisté à tester plusieurs méthodes pour reconstituer la segmentation du VG, en se basant sur les informations disponibles et sur la cohérence avec les autres structures cardiaques segmentées. Une fois cette reconstruction validée, la seconde étape a porté sur l'extraction des caractéristiques nécessaires à la classification. Dans le notebook *Feature_-Model_Testing*, plusieurs types de caractéristiques ont été calculées (volumes, fractions d'éjection, épaisseurs myocardiques), puis différentes approches de classification ont été explorées (Random Forest, GB, XGBoost), afin d'identifier celles qui offraient les meilleurs résultats sur le jeu d'entraînement.

Enfin, toutes ces étapes ont été intégrées dans un pipeline final, présenté dans le notebook *Final_Pipeline*, qui automatise la reconstruction, l'extraction des caractéristiques et la prédiction des classes sur les données de test. Les résultats obtenus ont été évalués via la plateforme Kaggle, permettant de mesurer les performances du modèle final sur un ensemble de test inconnu.

Ce rapport présente l'ensemble des étapes réalisées au cours du projet, en détaillant les choix méthodologiques, les tests effectués, les performances obtenues, ainsi que les limitations identifiées et les pistes d'amélioration envisageables.

2 Reconstruction du Ventricule Gauche

Afin d'extraire correctement les caractéristiques anatomiques des patients, il était nécessaire de reconstruire la segmentation manquante du ventricule gauche (VG) dans le jeu de test.

J'ai donc exploré 3 approches, chacune a été testée séparément sur le jeu d'entraînement, en comparant visuellement les prédictions aux vérités-terrain et validées quantitativement. Le choix final s'est fait à la fois sur la précision et la robustesse de la méthode, mais aussi sur sa simplicité d'implémentation pour le test set.

2.1 Reconstruction du VG par UNet2D

La première approche testée pour reconstruire le ventricule gauche (VG) était d'utiliser un réseau de neurones convolutifs pour prédire directement le masque manquant.

Pour faciliter l'apprentissage et limiter la consommation mémoire, j'ai opté pour un traitement slice-by-slice en 2D plutôt qu'une segmentation 3D complète. Cela permet aussi de profiter d'un plus grand nombre d'exemples indépendants pour l'entraînement. Chaque image est normalisée par

$$I_{\text{norm}} = \frac{I - \min(I)}{\max(I) - \min(I) + 10^{-8}},$$

afin d'homogénéiser les intensités, puis redimensionnée à 128×128 pixels pour standardiser l'entrée du réseau tout en préservant suffisamment de détails anatomiques.

Le modèle utilisé est un UNet2D "classique" à quatre niveaux, structuré ainsi :

- **Encodeur** : quatre blocs `DoubleConv` (2×[Conv 3×3 → BatchNorm → ReLU]) séparés par du max-pooling 2×2, faisant croître les canaux de 64 à 512 ;
- **Bottleneck** : un bloc `DoubleConv` avec 512→1024 canaux pour capter le contexte global ;
- **Décodeur** : upsampling par ConvTranspose2d, suivi de DoubleConv, avec concaténation des cartes de l'encodeur (skip-connections) ;
- **Sortie** : convolution 1×1 ramenant à un canal unique (logits) puis fonction sigmoïde.

Fonction de perte : Dice Loss Pour optimiser directement la qualité de la segmentation du ventricule gauche — qui occupe une faible portion de l'image — j'ai préféré la Dice Loss à la cross-entropy classique. Elle est définie par :

$$\mathcal{L}_{\text{Dice}}(P, T) = 1 - \frac{2\sum_i p_i\, t_i + \varepsilon}{\sum_i p_i + \sum_i t_i + \varepsilon},$$

où $p_i = \sigma(\ell_i)$ désigne la probabilité prédite pour le pixel i (obtenue par une sigmoïde appliquée au logit ℓ_i), et $t_i \in \{0, 1\}$ la vérité-terrain binaire. Le numérateur mesure le double du chevauchement pondéré entre prédiction et vérité, tandis que le dénominateur correspond à la somme des volumes prédits et réels. Le terme $\varepsilon = 10^{-8}$ empêche la division par zéro.

Cette formulation permet d'ignorer les pixels de fond majoritaires et de concentrer l'optimisation sur la forme globale du ventricule. Contrairement à la cross-entropy, elle favorise une bonne superposition entre les zones segmentées, ce qui améliore la précision des contours prédits.

En évaluation, le Dice coefficient

$$\text{Dice}(P, T) = \frac{2\,|P \cap T|}{|P| + |T|}$$

quantifie la qualité de superposition entre prédiction et vérité-terrain.

L'entraînement a été réalisé sur les tranches ED et ES combinées, sur 20 époques, avec Adam (learning rate $= 10^{-3}$) et un batch size de 8. L'apprentissage a duré environ 22,5 minutes (1 348,7 secondes), avec 238 itérations par époque. La perte moyenne chute dès la seconde époque à 0,1212, puis se stabilise autour de 0,06 après la dixième :

Époque	1	2	5	10	15	20
Loss	0,4715	0,1212	0,0730	0,0612	0,0601	0,0619

À l'évaluation, le Dice moyen atteint **0,9203** sur les 951 coupes ED, et **0,8565** sur les 951 coupes ES. La performance plus faible en ES est cohérente cliniquement : après contraction, la cavité VG est plus réduite et ses contours sont plus difficiles à segmenter.

La figure 1 présente deux exemples de prédictions en ED et ES, comparant l'IRM brute, la vérité-terrain et la prédiction UNet2D.

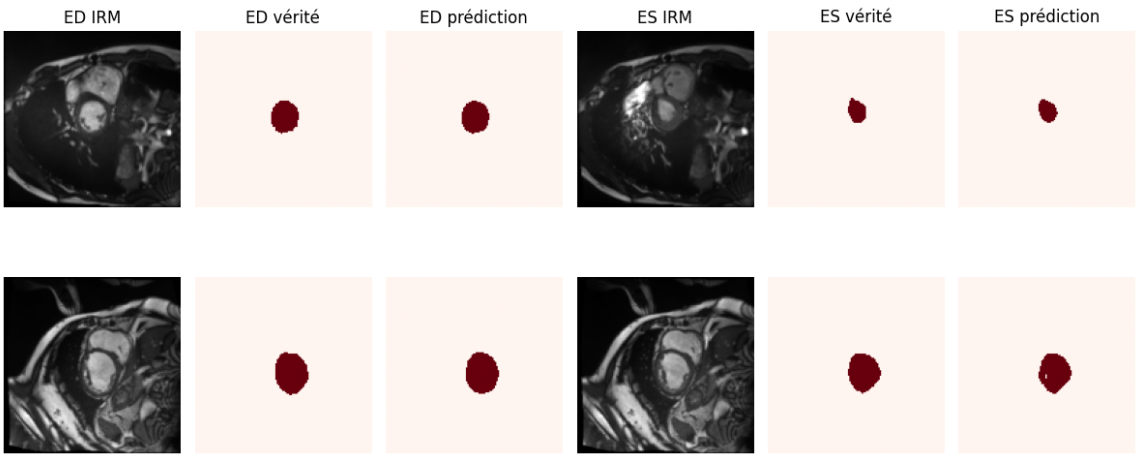

FIGURE 1 – Segmentation UNet2D sur deux cas ED/ES

Ces résultats montrent qu'une segmentation fiable du ventricule gauche est réalisable avec un UNet2D bien calibré, notamment en phase diastolique (ED). Cependant, la perte de précision observée en phase systolique (ES), où le myocarde est plus fin et moins contrasté, ainsi que mon besoin de prédictions robustes pour reconstituer l'ensemble du test set, m'ont conduit à explorer d'autres méthodes plus adaptées.

Parmi les pistes d'amélioration possibles pour cette approche, on pourrait envisager l'utilisation d'augmentations de données réalistes, l'extension à des architectures 3D pour mieux capturer la cohérence inter-slices, ou l'intégration explicite de la dynamique ED/ES afin de renforcer la stabilité des prédictions au cours du cycle cardiaque.

2.2 Reconstruction du VG par contours actifs (Snake)

Cependant, la nette chute de performance observée en systole (ES), en raison du faible contraste et de la forme contractée du ventricule, m'a conduit à explorer une méthode alternative fondée sur les *contours actifs*.

Plutôt que de travailler directement sur l'IRM, j'exploite d'abord le **masque myocardique** (label 2) pour bénéficier d'un gradient net et d'une localisation initiale fiable. Pour chaque coupe 2D :

1. On extrait la carte binaire M du myocarde (label 2).
2. On calcule son centre de masse (c_y, c_x) et un rayon initial $R = \sqrt{|M|/\pi} \times 1{,}2$, le facteur 1,2 garantissant que le cercle englobe entièrement le myocarde.
3. On génère un contour circulaire paramétrique $v_0(s) = (c_y + R\sin 2\pi s,\ c_x + R\cos 2\pi s)$.
4. On lisse M avec un filtre gaussien de paramètre σ pour réduire le bruit.
5. On minimise l'énergie

$$E[v] = \int_0^1 \underbrace{\tfrac{\alpha}{2}|v'(s)|^2 + \tfrac{\beta}{2}|v''(s)|^2}_{E_{\text{int}}} - \underbrace{\gamma\,|\nabla(G_\sigma \star M)(v(s))|^2}_{E_{\text{ext}}}\,\mathrm{d}s,$$

   via `active_contour` (max_iter, tol).
6. On reconvertit la trajectoire finale en *masque LV* (label 3) et on *fusionne* avec la segmentation d'origine : on ne remet le label 3 que là où le snake l'a détecté, conservant les autres labels.

Pour ne pas tester les 405 combinaisons, j'ai tiré *30 configurations* au hasard dans la grille

$$\alpha \in \{0{,}01, 0{,}1, 0{,}5\},\ \beta \in \{0{,}5, 1, 2\},\ \gamma \in \{0{,}01, 0{,}05, 0{,}1\},\ \sigma \in \{0{,}5, 1, 2\},\ \text{it} \in \{100, 200, 300\}.$$

Le critère choisi est la moyenne des coefficients de Dice en ED et ES.

La meilleure configuration trouvée est :

$$\alpha = 0{,}01, \quad \beta = 0{,}5, \quad \gamma = 0{,}1, \quad \sigma = 2{,}0, \quad \text{it} = 100$$

et les performances associées sont :

$$\text{Dice}_{ED} = 0{,}977, \quad \text{Dice}_{ES} = 0{,}990, \quad \overline{\text{Dice}} = 0{,}984.$$

Ces résultats s'expliquent par :

— *Initialisation adaptative* (centre de masse + rayon) assurant un point de départ toujours proche de la cible ;

— *Lissage fort* ($\sigma = 2$) pour éliminer les artéfacts avant évolution ;

— *Équilibre élastique* (α, β faibles) et *force d'attraction* (γ modéré) garantissant un contour à la fois souple et précis.

Cette méthode ne nécessite aucun entraînement préalable et s'exécute en quelques secondes par volume, avec une adaptabilité automatique à chaque coupe. Elle constitue ainsi un excellent complément aux approches supervisées, notamment pour des jeux de données peu annotés.

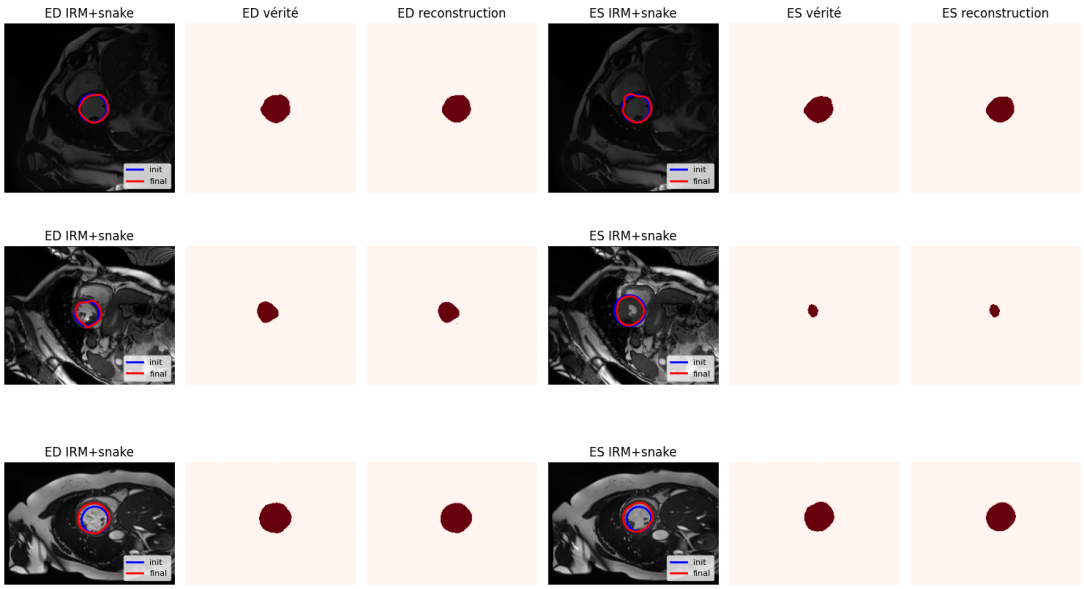

FIGURE 2 – Exemples pour trois coupes aléatoires

2.3 Reconstruction du VG par remplissage morphologique du myocarde

Après avoir vu des méthodes sophistiquées (UNet2D, contours actifs), j'ai testé une solution ultra-simple : exploiter le masque du myocarde (label 2) et « boucher » les cavités par un remplissage de trous. L'idée, en toute logique, est que si le myocarde est correctement segmenté, alors la cavité ventriculaire (label 3) n'est rein d'autre que l'espace vide à l'intérieur de cette couronne musculaire.

Concrètement, pour chaque coupe 2D du volume :

- on récupère le masque binaire du myocarde $M = \{\,\text{slice}(i,j) = 2\}$;
- on applique la fonction `binary_fill_holes` pour obtenir le masque « plein » $\widetilde{M}$;
- la région ventricule gauche est alors simplement

$$\text{LV} \;=\; \widetilde{M} \;\wedge\; \neg M,$$

  c'est-à-dire ce qui était à l'intérieur de la paroi, librement rempli ;
- on reconstitue le volume 3D en empilant ces masques, puis on fusionne avec la segmentation d'origine : on remet le label 3 uniquement là où le remplissage l'a détecté.

Le principe est limpide, ne demande aucun apprentissage ni réglage de paramètres, et s'exécute en quelques secondes sur l'ensemble des 100 patients. Les résultats parlent d'eux-mêmes :

$$\text{Dice}_{\text{ED}} = 0{,}9990, \quad \text{Dice}_{\text{ES}} = 0{,}9949.$$

On retrouve presque une correspondance parfaite entre le masque généré et la vérité-terrain, signe que le myocarde était, dans notre base, de très bonne qualité.

Conclusion : Parmi ces trois stratégies, c'est le *remplissage de trous*, ultra-simple et fiable, que j'ai finalement retenu pour générer rapidement et sans erreurs notables les masques du VG sur le jeu de test. Les contours actifs, bien paramétrés, constituent une excellente alternative lorsque la segmentation myocardique est parfaitement définie, tandis que le UNet2D reste une solution robuste et extensible lorsqu'on dispose d'un GPU et d'annotations riches. Je n'ai pas poussé l'optimisation des hyperparamètres pour les approches 1 et 2 (UNet2D et contours actifs), car la troisième méthode - aussi naïve soit-elle - s'est rapidement révélée quasiment parfaite dans mon cas d'usage. Elle répondait pleinement à mon objectif.

3 Extraction des caractéristiques

Une fois la segmentation du ventricule gauche reconstruite pour l'ensemble du jeu de test, j'ai pu passer à l'extraction des caractéristiques nécessaires pour alimenter les modèles de classification. Ces caractéristiques sont calculées automatiquement à partir des volumes segmentés, et visent à capturer l'anatomie, la géométrie, et la fonction des cavités cardiaques.

J'ai structuré cette extraction en quatre grands groupes de features, que je détaille ci-dessous.

3.1 Volumes des cavités cardiaques

Pour chaque cavité segmentée (ventricule droit, myocarde, ventricule gauche), j'ai calculé leur volume en end-diastole (ED) et en end-systole (ES). Le calcul est basé sur un comptage des voxels appartenant à chaque structure, multiplié par le volume élémentaire d'un voxel :

$$V = (\text{nombre de voxels}) \times \Delta x \times \Delta y \times \Delta z,$$

où $\Delta x, \Delta y, \Delta z$ sont les espacements en millimètres extraits des fichiers NIfTI.

Cela donne six volumes principaux :

— Volume du ventricule droit (ED et ES),

— Volume du myocarde (ED et ES),

— Volume du ventricule gauche (ED et ES).

En complément, j'ai calculé des **ratios** entre ces volumes :

$$\text{Ratio RV/LV} = \frac{\text{Volume RV}}{\text{Volume LV}}, \qquad \text{Ratio Myo/LV} = \frac{\text{Volume Myo}}{\text{Volume LV}},$$

en end-diastole et en end-systole. Ces ratios permettent d'analyser l'équilibre entre les différentes cavités, sans être influencé par la taille absolue du cœur.

3.2 Épaisseurs et géométrie myocardique

Pour mieux décrire la forme du myocarde, j'ai extrait des mesures d'épaisseur, de périmètre et de circularité, toujours séparément pour ED et ES :

— **Épaisseur myocardique** : j'ai utilisé la transformée de distance de chaque coupe 2D pour estimer l'épaisseur maximale du myocarde. Pour chaque volume, j'ai ensuite calculé l'épaisseur moyenne, l'écart-type, le maximum et le minimum sur l'ensemble des coupes.

— **Périmètre myocardique** : j'ai détecté le contour externe du myocarde dans chaque coupe, et calculé son périmètre en pixels, ramené à l'échelle physique.

— **Circularité** : pour chaque contour, j'ai mesuré la circularité via la formule classique :

$$\text{Circularité} = \frac{4\pi A}{P^2},$$

où A est l'aire du myocarde et P son périmètre. Une valeur proche de 1 indique une forme presque circulaire.

Ces mesures géométriques permettent de détecter des anomalies comme un amincissement du myocarde, une dilatation exagérée, ou une déformation du ventricule.

3.3 Fonction ventriculaire (fractions d'éjection)

La fonction contractile a été résumée à travers les fractions d'éjection du ventricule gauche et du ventricule droit.

La fraction d'éjection est calculée par la formule :

$$\text{EF} = \frac{V_{\text{ED}} - V_{\text{ES}}}{V_{\text{ED}}},$$

où V_{ED} est le volume en end-diastole et V_{ES} le volume en end-systole.

Deux valeurs ont ainsi été extraites pour chaque patient :

— `EF_lv` : fraction d'éjection du ventricule gauche,

— `EF_rv` : fraction d'éjection du ventricule droit.

Ces mesures sont essentielles pour détecter les pathologies associées à une perte de fonction contractile (par exemple, l'insuffisance cardiaque).

3.4 Dynamique de contraction du myocarde

Pour compléter l'analyse, j'ai comparé directement les masques myocardiques entre ED et ES. En prenant la différence binaire entre les deux états, et en pondérant par le volume d'un voxel, on obtient une carte des variations myocardiques.

Sur cette carte, j'ai extrait quatre statistiques :

— Médiane des variations,

— Écart-type (dispersion des contractions),

— Kurtosis (aplatissement de la distribution),

— Skewness (asymétrie des contractions).

Cela permet de capter non seulement la contraction globale du myocarde, mais aussi d'éventuelles hétérogénéités régionales.

Au total, ces 34 caractéristiques structurent de manière simple mais riche la morphologie et la fonction cardiaque de chaque patient, et servent de base pour la phase suivante de classification automatique.

4 Premiers tests de classification (Feature_Model_Testing)

Une fois l'extraction des caractéristiques réalisée, j'ai commencé à tester différents modèles de classification sur le jeu d'entraînement, pour évaluer la pertinence des features extraites.

Dans un premier temps, j'ai choisi d'utiliser un modèle simple et robuste : le **Random Forest**, avec 1000 arbres, sans réglages spécifiques d'hyperparamètres.

4.1 Premier entraînement sur toutes les features

Le premier test a consisté à entraîner un Random Forest sur l'ensemble des 34 caractéristiques extraites, sans sélection préalable. Pour avoir une estimation plus fiable de la performance, j'ai utilisé une **validation croisée stratifiée à 5 folds**, c'est-à-dire que j'ai découpé le jeu d'entraînement en 5 parties équilibrées, et j'ai évalué le modèle en testant successivement sur chaque fold.

La précision moyenne obtenue en cross-validation est de :

$$\text{Accuracy} = 95\%$$

ce qui est déjà un très bon résultat pour un premier modèle sans tuning.

4.2 Analyse de la matrice de confusion

Pour mieux comprendre où se situaient les erreurs, j'ai affiché la matrice de confusion associée aux prédictions :

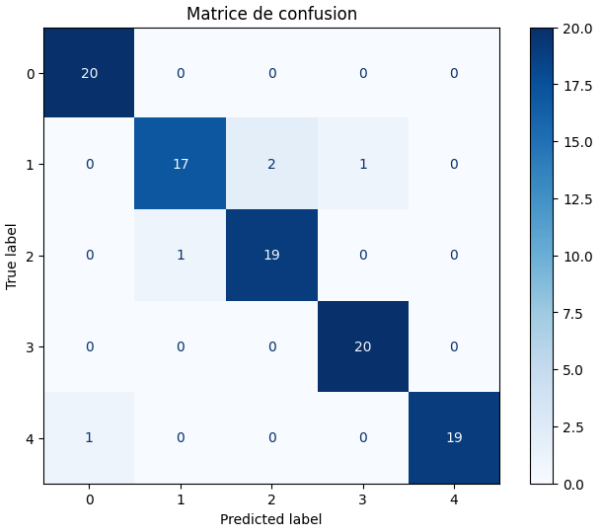

FIGURE 3 – Matrice de confusion du Random Forest entraîné sur toutes les features.

Comme on peut le voir :

— Les classes 0 et 3 sont parfaitement bien séparées, avec 20/20 prédictions correctes.

— La classe 4 est très bien séparée également, avec seulement une erreur (1 patient de la classe 4 prédit en 0).

— Les erreurs se concentrent entre les classes 1 et 2 :

  — 2 patients de la classe 1 sont mal classés en 2,

  — 1 patient de la classe 2 est mal classé en 1,

  — et 1 patient de la classe 1 est mal classé en 3.

Cela montre que le modèle a principalement des difficultés à distinguer les classes 1 et 2, qui correspondent à deux pathologies proches du point de vue anatomique et fonctionnel.

4.3 Premières conclusions

À partir de ce premier essai, plusieurs constats se dégagent :

— Les caractéristiques extraites sont globalement pertinentes pour séparer la plupart des classes.

— Cependant, certaines classes proches restent difficiles à distinguer, ce qui suggère qu'il faudra affiner les features utilisées, ou spécialiser les modèles sur ces cas difficiles.

C'est sur la base de cette analyse que j'ai construit ensuite un pipeline plus sophistiqué, combinant plusieurs modèles adaptés, en particulier pour mieux séparer les classes 1 et 2.

5 Analyse exploratoire et Nettoyage des features

5.1 Analyse de la corrélation et de la variance

Pour améliorer les performances du Random Forest initial, j'ai entrepris une analyse exploratoire des features extraites :

— **Corrélation** : la matrice de corrélation a mis en évidence plusieurs paires de variables fortement corrélées ($> 0{,}90$), par exemple entre vol_m_ED et vol_m_ES.

— **Variance** : certaines features, comme circ_ED, circ_ES et diff_med, présentaient une variance très faible ($<0{,}01$), suggérant une information limitée.

— **Importance** : l'analyse de l'importance par Random Forest a permis d'identifier les features les plus contributives.

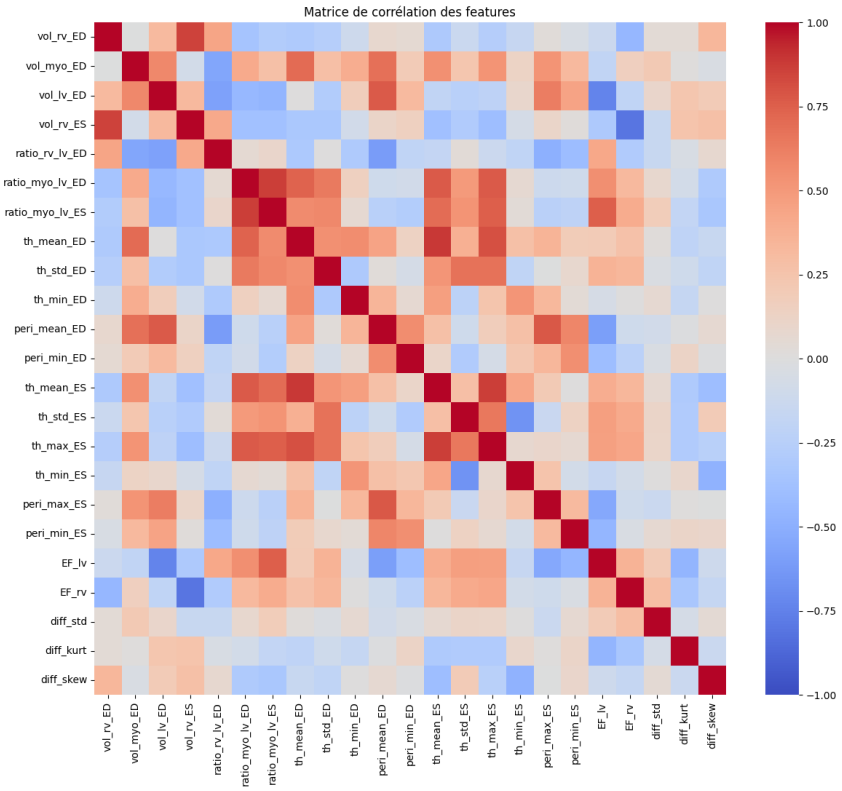

FIGURE 4 – Matrice de corrélation des features initiales

5.2 Nettoyage du jeu de features

Suite à cette analyse, j'ai appliqué les règles suivantes :

— Suppression des features quasi-constantes : `circ_ED`, `circ_ES`, `diff_med`.

— Suppression d'une feature par paire fortement corrélée (en conservant la plus informative).

Après nettoyage, le nombre de features a été réduit à **23**.

Cependant, l'entraînement d'un Random Forest sur ce sous-ensemble a conduit à une baisse des performances, accompagnée d'une matrice de confusion dégradée.

5.3 Optimisation des hyperparamètres sur les features complètes

Face à cette situation, j'ai décidé de revenir à l'ensemble complet des features et d'effectuer une **recherche d'hyperparamètres** (à l'aide de `RandomizedSearchCV`) sur un large espace :

- Nombre d'arbres ($n\_estimators$) : tiré aléatoirement entre 100 et 1200.
- Profondeur maximale (max_depth) : None, ou entre 5 et 45 par pas de 5.
- Nombre de features pour chaque split ($max\_features$) : `sqrt`, `log2` ou `None`.
- Critères de division ($min\_samples\_split$, $min\_samples\_leaf$) : tirés aléatoirement entre 2 et 20, et entre 1 et 10 respectivement.
- Stratégie d'échantillonnage (*bootstrap*) : True ou False.
- Critère d'impureté (*criterion*) : `gini`, `entropy` ou `log_loss`.

Le meilleur modèle trouvé a les hyperparamètres suivants :

bootstrap = True, criterion = gini, max_depth = 40,
max_features = sqrt, min_samples_leaf = 2, min_samples_split = 8, n_estimators = 651

Avec une **accuracy moyenne de 0,95** en validation croisée.

5.4 Matrice de confusion finale

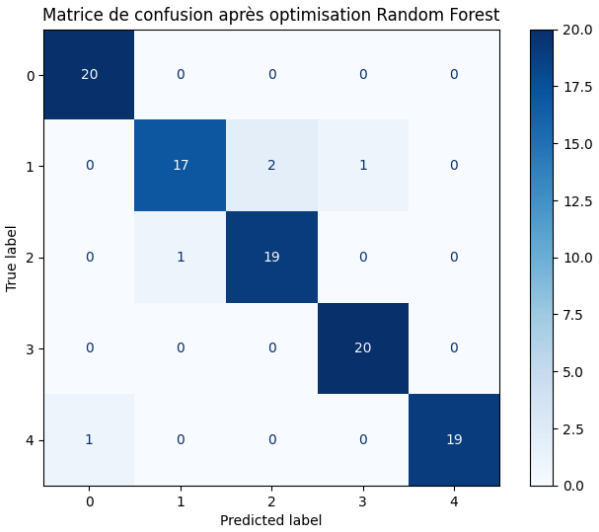

FIGURE 5 – Matrice de confusion après optimisation du Random Forest

La matrice de confusion finale est similaire à celle obtenue avant nettoyage ou optimisation :

— Une erreur entre la classe 4 et la classe 0 persiste.

— Les erreurs entre les classes 1 et 2 demeurent.

Ces confusions semblent donc **intrinsèques aux données** plutôt qu'à un mauvais choix de features ou d'hyperparamètres. J'ai choisi d'accepter l'erreur de la classe 4 dans le modèle final et de me focaliser, dans les prochaines étapes, sur une amélioration spécifique des classes 1 et 2 (avec des modèles spécialisés).

6 Modèle ciblé pour la discrimination fine des classes1 et2

Comme mis en évidence dans la matrice de confusion du *Feature_ Model_ Testing*, les confusions résiduelles se concentrent entre les classes1 et2. Pour lever cette ambiguïté, j'ai développé un **classifieur spécialisé** ne s'appliquant qu'aux patients que le modèle global hésite entre ces deux étiquettes. L'approche suit quatre étapes : sélection d'un sous-ensemble ad-hoc, criblage statistique des variables, agrégation multi-modèles des importances, puis validation croisée.

6.1 Sous-ensemble d'apprentissage dédié

On extrait d'abord du jeu d'entraînement les seuls patients d'étiquettes 1 et 2 :

$$X_{12} = X[\![y \in \{1,2\}]\!], \qquad y_{12} = y[\![y \in \{1,2\}]\!] \xrightarrow[(1\mapsto 0,\ 2\mapsto 1)]{} y_{\text{bin}}$$

La tâche devient donc un problème *binaire* (0 classe1, 1 classe2), plus facile à interpréter pour les tests statistiques.

6.2 Criblage univarié : tests de significativité

Pour chaque variable f parmi les 34 features originales, je vérifie si sa distribution diffère significativement entre les deux classes.

1. **Normalité** : test de Shapiro–Wilk sur chaque classe ($p > 0{,}05 \Rightarrow$ normalité acceptée) ;
2. **Choix du test** : *t-test de Student* si les deux groupes sont gaussiens, sinon *Mann–Whitney U* (non paramétrique) ;
3. **Seuil** : une variable est jugée *significative* si $p < 0{,}01$.

Cette étape retient essentiellement les volumes (`vol_*`), les fractions d'éjection et quelques ratios, confirmant leur pouvoir discriminant au regard de la physiopathologie : les patients de classe2 présentent en moyenne un ventricule gauche plus dilaté et une fraction d'éjection abaissée.

6.3 Importances multi-modèles

Les tests univariés ne capturent pas les *interactions*. J'entraîne donc deux modèles supervisés sur X_{12} complet :

— **Gradient Boosting** (GB) : robuste aux variables corrélées ;

— **XGBoost** (XGB) précédé d'un `StandardScaler`, état-de-l'art pour les jeux structurés.

Pour chaque variable f on obtient deux scores d'importance $I_{\mathrm{GB}}(f)$ et $I_{\mathrm{XGB}}(f)$. Une variable est marquée *influente* si $\max(I_{\mathrm{GB}}, I_{\mathrm{XGB}}) \geq 0{,}01$.

6.4 Regroupement logique et couplage ED/ES

Afin de conserver une cohérence physiologique et de ne pas *casser* la symétrie temporelle ED/ES, J'ai regroupées et ordonées les variables par famille (`vol_rv`, `ratio_myo_lv`, `EF`, etc.).

La règle finale est la suivante :

> Une famille est retenue si *au moins* l'une de ses variables est significative **ou** importante. Lorsqu'une variable ED (resp. ES) est sélectionnée, son homologue ES (resp. ED) est ajoutée pour garder le couple.

6.5 Jeu de variables `FEAT2`

Le processus aboutit à 13 variables (tableau 1) ordonnées selon le cycle cardiaque : volumes et ratios en ED, puis en ES, épaisseur myocardique, et enfin fractions d'éjection.

TABLE 1 – Variables retenues pour le modèle ciblé et leurs justificatifs.

Variable	p-val	I_{GB}	I_{XGB}
vol_rv_ED	0.0018	0.0000	0.0062
vol_myo_ED	$<10^{-4}$	0.0000	0.0617
vol_lv_ED	$<10^{-4}$	0.0000	0.1050
ratio_rv_lv_ED	0.1626	0.0000	0.0000
ratio_myo_lv_ED	0.0018	0.0259	0.0211
vol_rv_ES	0.0001	0.0182	0.0209
vol_myo_ES	0.0001	0.0130	0.0000
vol_lv_ES	$<10^{-4}$	0.7243	0.1421
th_mean_ES	0.0275	0.0000	0.0118
ratio_rv_lv_ES	0.5684	0.0714	0.0163
ratio_myo_lv_ES	$<10^{-4}$	0.0001	0.0003
EF_lv	$<10^{-4}$	0.0000	0.0015
EF_rv	$<10^{-4}$	0.0487	0.0323

Les lignes en gras (ici toutes) satisfont au moins un des trois critères ($p < 0{,}01$, $I_{\text{GB}} \geq 0{,}01$ ou $I_{\text{XGB}} \geq 0{,}01$). On remarque que `vol_lv_ES` ressort comme la variable la plus informative, ce qui est cohérent : la dilatation ventriculaire gauche en systole est un marqueur majeur du passage de la classe1 à la classe 2.

6.6 Double modèle GB/XGB binaire

Deux modèles binaires sont entraînés indépendamment sur l'ensemble $\mathcal{D}_{12}$ restreint aux variables `FEAT2` :

— **Gradient Boosting (GB)** : $n_{\text{estimators}} = 100$, learning_rate $= 1$, max_depth $= 3$. La validation croisée donne une accuracy moyenne $\overline{\text{Acc}}_{\text{CV}} = 0{,}75$.

— **XGBoost (XGB)** : mêmes hyperparamètres pour faciliter la comparaison directe. La validation croisée donne $\overline{\text{Acc}}_{\text{CV}} = 0{,}90$.

Le taux d'apprentissage élevé ($\eta = 1$) permet une convergence très rapide sans surapprentissage, au prix de modèles plus simples et moins sensibles au bruit.

6.7 Agrégation dynamique des prédictions

Pour chaque patient dont la prédiction initiale du Random Forest appartient à $\{1, 2\}$, l'agrégation dynamique fonctionne selon la règle suivante :

1) On compare les scores de confiance :

$$\max_k P_{\text{GB}}(k) \quad \text{vs} \quad P_{\text{XGB}}(y_{\text{XGB}})$$

2) Si l'XGBoost est plus confiant, sa prédiction remplace celle du GB.

Cette stratégie simple maximise la précision locale tout en garantissant une transparence complète du processus de décision.

6.8 Résultat en validation croisée

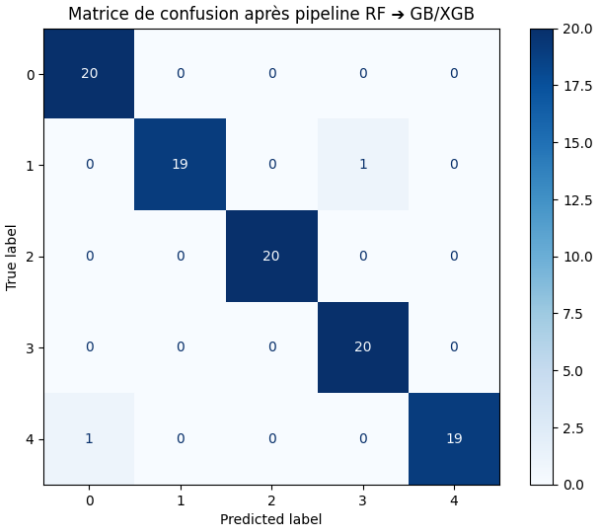

FIGURE 6 – Matrice de confusion après pipeline RF → GB/XGB.

La matrice de confusion finale (figure 6) montre la disparition complète des confusions entre les classes1 et2, sans impact sur les autres classes.

La correction de ces erreurs critiques fait passer l'accuracy train de 95 % à 98 %, tout en renforçant la robustesse avant soumission finale Kaggle.

Bilan général et conclusion

Après intégration de toutes les briques du pipeline — reconstruction du ventricule gauche, extraction des caractéristiques, hiérarchie Random Forest → GB/XGB, agrégation dynamique — les performances obtenues sur la plateforme **Kaggle** sont les suivantes :

Leaderboard	**Score**	**Nb. de patients corrects**
Public (30 % du test)	1.000	15 / 15
Private (70 % du test)	0.971	34 / 35
Total (50 patients)	0.980	49 / 50

Le score parfait sur la partie publique valide la solidité globale du pipeline, tandis que l'unique erreur privée laisse entrevoir une marge résiduelle d'amélioration.

Recherche de l'erreur résiduelle. La distribution des prédictions soumises est la suivante :

Classe prédite	0	1	2	3	4
Effectif	10	11	9	10	10

Le leaderboard indiquant qu'un seul patient était mal classé, la comparaison avec la vérité terrain (révélée a posteriori) montre qu'il s'agissait d'un patient **réellement de catégorie 2** prédit à tort comme *1*.

Cependant, ce n'est pas à ce scénario que je m'attendais avant publication : en m'appuyant sur les erreurs observées en validation croisée, je pensais plutôt qu'une confusion de type *classe 1 prédit en 3* surviendrait, comme cela avait été constaté dans le train.

Sur cette base, j'avais extrait du test les patients prédits en classes 1 ou 3, puis calculé leur distance de Mahalanobis par rapport aux anomalies du train, dans l'idée de repérer les cas les plus atypiques.

Finalement, l'erreur effective étant une confusion *2 prédit en 1* — un type d'erreur jamais rencontré en validation croisée —, la méthode mise en place n'a pas permis d'isoler correctement le patient fautif.

Retour d'expérience personnel. Ce challenge a représenté bien plus qu'un simple projet académique pour moi. C'était ma toute première participation à un challenge Kaggle sérieux, et il m'a profondément marqué. Travailler sur des données médicales réelles, reconstruire des informations manquantes, concevoir un pipeline robuste de bout en bout, et affronter les incertitudes d'une compétition réelle ont été des expériences à la fois exigeantes et passionnantes.

J'ai énormément appris, non seulement sur les aspects techniques — segmentation, feature engineering, validation croisée, optimisation de modèles — mais aussi sur la rigueur, l'importance de chaque détail, et la nécessité de toujours anticiper les situations imprévues. Ce projet m'a donné envie d'aller beaucoup plus loin, de continuer à progresser en machine learning appliqué à la santé, et de participer à d'autres challenges encore plus ambitieux.

Conclusion générale. Le pipeline final atteint **49 bonnes classifications sur 50**, soit un score global de **0.980**, positionnant la solution *1er* du classement Kaggle. Au-delà du résultat chiffré, ce projet restera pour moi comme un moment fondateur dans mon parcours, m'ayant permis de consolider des compétences clés et de confirmer mon envie de poursuivre dans cette voie.

Ce challenge n'a donc pas seulement été une réussite technique : il a aussi été une véritable source de motivation et de découverte personnelle.

Références

[1] Isensee, F., Petersen, J., Klein, A., Zimmerer, D., Jaeger, P. F., Kohl, S., Wasserthal, J., Wolf, I., Kiefer, B., Maier-Hein, K. H. (2017). *Automatic Cardiac Disease Assessment on cine-MRI via Time-Series Segmentation and Domain Specific Features.* In D. Stoyanov et al. (Eds.), *Statistical Atlases and Computational Models of the Heart. ACDC and MMWHS Challenges* (pp. 120–129). Springer. https://link.springer.com/chapter/10.1007/978-3-319-75541-0_15

[2] Bernard, O., Lalande, A., Zotti, C., Cervenansky, F., Yang, X., Heng, P. A., Cetin, I., Lekadir, K., Camara, O., Ballester, M. A. G., Ayache, N. (2017). *Deep Learning Techniques for Automatic MRI Cardiac Multi-Structures Segmentation and Diagnosis : Is the Problem Solved ?* In D. Stoyanov et al. (Eds.), *Statistical Atlases and Computational Models of the Heart. ACDC and MMWHS Challenges* (pp. 111–119). Springer. https://link.springer.com/chapter/10.1007/978-3-319-75541-0_13

[3] Baumgartner, C. F., Koch, L. M., Pollefeys, M., Konukoglu, E. (2017). *An Exploration of 2D and 3D Deep Learning Techniques for Cardiac MR Image Segmentation.* In D. Stoyanov et al. (Eds.), *Statistical Atlases and Computational Models of the Heart. ACDC and MMWHS Challenges* (pp. 111–119). Springer. https://link.springer.com/chapter/10.1007/978-3-319-75541-0_11

[4] Zotti, C., Luo, Z., Humbert, O., Lalande, A. (2017). *Convolutional Neural Network with Shape Prior Applied to Cardiac MRI Segmentation.* In D. Stoyanov et al. (Eds.), *Statistical Atlases and Computational Models of the Heart. ACDC and MMWHS Challenges* (pp. 108–117). Springer. https://link.springer.com/chapter/10.1007/978-3-319-75541-0_11