IMA205 Challenge Report

Nathan Rouillé

May 2025

1 Introduction

This report presents my work for a classification challenge on cardiac MRI data. The goal of the project is to automatically classify 150 subjects into five diagnostic categories based on features extracted from their cardiac MRI exams. These categories are:

- 0 – Healthy controls
- 1 – Myocardial infarction
- 2 – Dilated cardiomyopathy
- 3 – Hypertrophic cardiomyopathy
- 4 – Abnormal right ventricle

To do this, I extracted anatomical and functional features from 3D images at two different phases of the cardiac cycle: end-diastole and end-systole. I then trained and evaluated various machine learning models to predict the correct diagnosis based on these features. This report describes the data, the preprocessing steps, the feature extraction process, the models tested, and the final classification pipeline.

2 Data Description

The dataset provided for this challenge contains 150 subjects, each associated with cardiac MRI images and anatomical segmentations and basic metadata (height and weight). The data is split into a training-validation set (100 subjects) and a test set (50 subjects). Ground truth diagnostic labels are only available for the training-validation set.

For each subject, two 3D MRI volumes are available:

- One at **end-diastole (ED)** — the point of maximum ventricular filling
- One at **end-systole (ES)** — the point of maximal contraction

In the training set, each image is accompanied by a multi-label segmentation mask that identifies three key anatomical structures:

- Label 1 – Right ventricle cavity
- Label 2 – Myocardium
- Label 3 – Left ventricle cavity

Label 0 corresponds to the background. In the test set, the left ventricle cavity label is systematically missing (replaced by background), which require the development of an intermediate segmentation step for complete feature extraction.

All images and segmentations are stored in compressed NIfTI format (.nii.gz), which can be processed using libraries such as `TorchIO`. Visualization can be done either through 2D slice using `matplotlib` or via dedicated 3D software such as ParaView or 3D Slicer.

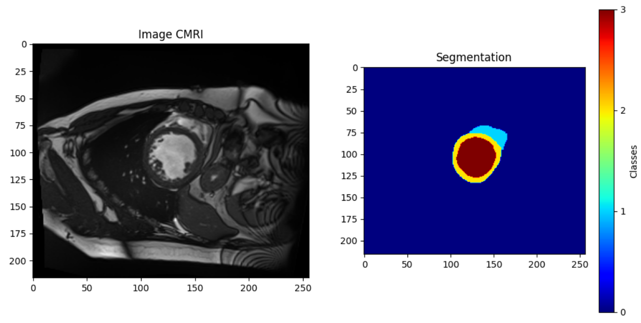


Figure 1: Example of a cardiac MRI slice for train patient.

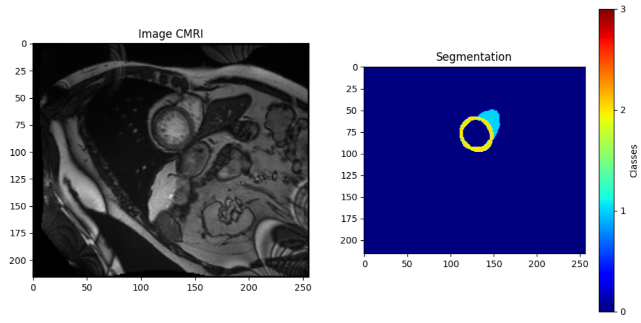


Figure 2: Example of a cardiac MRI slice for test patient.

3 Preprocessing

A crucial step in this project involved recovering the missing left ventricle cavity segmentations for the test subjects. While the provided training data included full segmentations of all relevant cardiac structures, the test data lacked label 3, which corresponds to the left ventricle cavity.

Upon visual inspection of the segmentations, I observed that the myocardium (label 2), which surrounds the left ventricle, was still present. This anatomical property allowed me to infer the location of the left ventricle cavity. Specifically, for each 2D slice of the segmentation, I applied the `scipy.ndimage.binary_fill_holes` function to the myocardium mask. This operation effectively filled the cavity enclosed by the myocardium, recovering a plausible left ventricle cavity segmentation.

This step enabled consistent feature extraction across both the training and test sets, including important left ventricle metrics such as volume and ejection fraction.

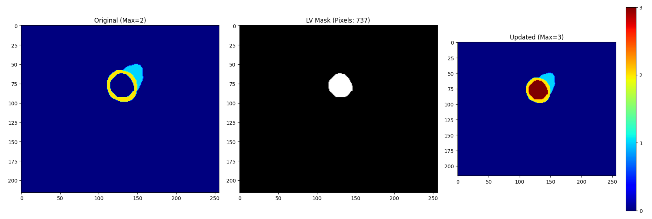


Figure 3: Example of left ventricle segmentation.

4 Feature Extraction

The choice of features was guided by insights from the medical literature as well as from previous studies addressing this specific classification challenge. I focused on features known to be clinically relevant and potentially discriminative between cardiac pathologies. Their consistency and discriminative power were later verified by analyzing their distributions across the different diagnostic classes.

To represent each subject in a compact and informative way, I extracted a total of 20 handcrafted features from the CMRI segmentations (+ weight and height). These features aim to capture both global and local anatomical and functional properties of the heart, derived from the segmentation maps at end-diastole (ED) and end-systole (ES) phases.

Volume-based Features

First, I computed the volumes of three key anatomical structures: the left ventricle cavity (LV), the right ventricle cavity (RV), and the myocardium (MYO).

For each structure, volumes were computed at both ED and ES phases using voxel counting and the image spacing metadata. From these raw volumes, I derived additional functional and anatomical ratios:

- **LV and RV ejection fractions (EF)**: estimated as the relative volume difference between ED and ES.
- **RV/LV volume ratios** at both ED and ES phases.
- **MYO/LV volume ratios** at both ED and ES phases.

In total, these operations yield 12 features: 6 raw volumes (LV, RV, MYO at ED and ES), 2 ejection fractions, and 4 volume ratios.

Myocardial Thickness Features

To further characterize cardiac morphology, I computed 8 features related to myocardial wall thickness. For each 2D axial slice in the segmentation volume, I extracted the inner (endocardial) and outer (epicardial) contours of the myocardium and calculated the minimal distances between them to estimate local wall thickness.

For both ED and ES phases, I aggregated these thickness profiles across slices by computing:

- The maximum mean thickness across all slices.
- The standard deviation of mean thicknesses (slice-wise variability).
- The mean of thickness standard deviations (intra-slice variability).
- The standard deviation of thickness standard deviations.

These descriptors capture both the global average thickness and its heterogeneity across the myocardium, which is relevant for detecting pathologies such as hypertrophy or dilation.

Final Feature Set

The final feature vector for each subject consists of:

- 12 volume-based features (including ejection fractions and ratios).
- 8 myocardial thickness-based features (4 from ED and 4 from ES).
- weight and height

These 22 features were extracted consistently for both training and test datasets and stored in CSV format for further processing.

5 Methodology

5.1 Overview

After performing the necessary preprocessing steps and extracting 22 features per subject, we aimed to train a robust classification model for diagnosing cardiac pathologies. To avoid bias and ensure the objectivity of our results, we decided to compare several families of classifiers: Support Vector Classifier (SVC), Logistic Regression, Random Forest, XGBoost, LightGBM, Multi-Layer Perceptron (MLP), and Naive Bayes. While one could select a model based on cross-validation performance with default parameters, this does not guarantee the optimal configuration, as the best model with default settings is not necessarily the best once optimized. For this reason, we employed a rigorous model selection procedure using nested cross-validation.

5.2 Nested Cross-Validation Setup

To ensure a reliable estimation of the generalization performance and fair comparison of models, we used nested cross-validation with two stratified loops:

- An **outer 5-fold stratified cross-validation**, used to compare the different classification algorithms.
- An **inner 3-fold stratified cross-validation**, used to optimize the hyperparameters of each model via grid search.

In each outer fold, the model is trained on 80% of the training data and validated on the remaining 20%. However, for every outer train/validation split, hyperparameter tuning is first performed using the 3-fold inner cross-validation on the 80% training portion:

- 66% of the outer training set (i.e., about 53% of the full training dataset) is used for training during inner CV.
- 33% (i.e., about 26% of the full training dataset) is used for validation.

This ensures that the outer validation set remains completely unseen during both training and hyperparameter optimization.

The choice of using 5 folds for the outer loop and 3 folds for the inner loop is motivated by:

- The size of the dataset (100 training samples): folds need to be large enough to ensure stable validation statistics.
- The computational complexity: comparing 7 different models, each trained and validated over 5 outer folds, and optimized over a 3-fold inner CV with a hyperparameter grid (ranging from dozens to thousands of combinations) results in a total number of training operations on the order of $7 \times 5 \times (1 + 3 \times \#paramcombinations)$, which can reach tens of thousands.

- The fact that the original training/test split (100/50) corresponds to a 2/3 training and 1/3 test division, making a 3-fold CV setup particularly suitable for evaluating the generalization capability of the models.

All cross-validation folds are **stratified**, ensuring balanced class distributions across training and validation sets. This is crucial given that each class contains the same number of subjects in the dataset.

5.3 Pipelines and Data Leakage Prevention

For models requiring standardized input features (SVC, Logistic Regression, MLP), we used Scikit-learn `Pipeline` objects that included a `StandardScaler` before the classifier. The scaler was fit only on the training portion of each fold:

- During inner cross-validation: the scaler is fit on the 66% inner training split and applied to the 33% validation split.
- During outer cross-validation: the scaler is fit on the 80% training split and used to transform the 20% validation split.

If we fit the scaler on the entire training set/entire training fold before outer/inner cross-validation, it would have leaked information from validation data into the model, violating the principles of nested CV and introducing bias.

5.4 Model Selection and Results

Given the large hyperparameter search space, we aimed to minimize the number of parameter combinations to keep computation time reasonable. For this, we used a **grid search** with only a few values per parameter, initially chosen to span a broad range. After each run, we manually analyzed the best-performing configurations and iteratively **refined and shifted the parameter ranges**, focusing the search around promising regions. This process allowed us to quickly explore the space and progressively hone in on optimal configurations while maintaining low computational cost. This approach was motivated by the fact that the final model would later undergo a more precise optimization, making this first stage a fast, coarse preselection.

Using this procedure within the nested cross-validation framework, we obtained the following average accuracies across outer folds:

- **Random Forest**: 96%
- **SVC**: 95%
- **Logistic Regression**: 94%

We selected the Random Forest classifier for further optimization due to its higher cross-validation performances.

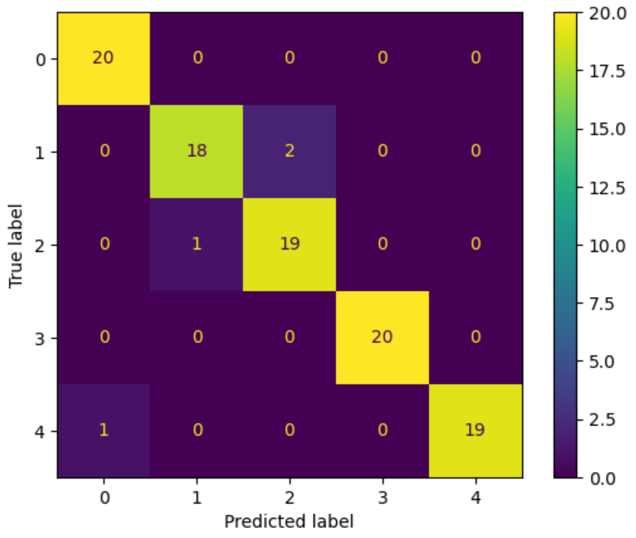


Figure 4: Confusion Matrix Random Forest.

5.5 Advanced Optimization and Feature Selection

We now aim to further improve the performance of our selected model by finding more precise hyperparameters and incorporating feature selection. For this, we continued to use a nested cross-validation scheme, but we modified the Random Forest pipeline by adding a feature selector step at the beginning. This selector chooses the most important features according to a Random Forest model trained with fixed hyperparameters (identical to those of the classifier being optimized), leveraging the native feature importance scores of Random Forest model.

The optimization included the following modifications:

- **New hyperparameters** were added to the search space (e.g., minimum number of samples per split, criterion).
- **Wider ranges** were explored for previously used hyperparameters to ensure coverage of more diverse configurations.
- **More precise search intervals** were used, with continuous search spaces where applicable and step sizes of 1 for discrete parameters.
- **Feature selector hyperparameters** were added: the `selector_threshold` (minimum importance required for a feature to be kept) and `selector_max_features` (maximum number of selected features).

Due to the enlarged and more complex search space, we switched to **Bayesian optimization** to efficiently explore promising regions without exhaustively evaluating all combinations.

Surprisingly, the optimized model achieved the same cross-validation accuracy as the simpler initial Random Forest. Both models made the same classification errors: one subject from class 4 misclassified as class 0, and some confusions between classes 1 and 2. Notably, no healthy subject (class 0) was predicted as class 4, suggesting that the misclassified class 4 subject may be an outlier.

Given the identical performance but increased complexity of the new model, I chose to continue using the original Random Forest configuration, which has fewer hyperparameters so a stronger regularization, and lower risk of overfitting the training data.

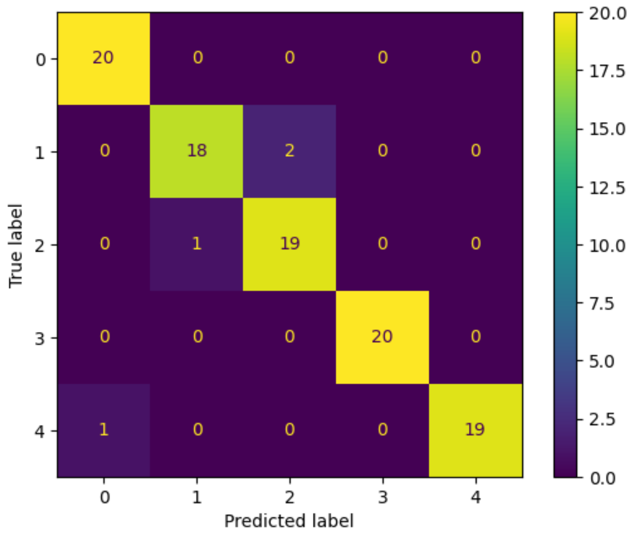


Figure 5: Confusion Matrix Random Forest after advanced optimization.

5.6 Two-Stage Classification Refinement

Given the frequent confusion between class 1 (myocardial infarction) and class 2 (dilated cardiomyopathy), we implemented a two-stage classification model to improve discrimination between these two specific classes:

1. A first-stage Random Forest classifier predicts among all five classes.
2. If the prediction is class 1 or 2, a second-stage model refines this decision using a classifier trained solely on samples from these two classes.

We tested several candidate models for the second stage using the same nested cross-validation procedure:

- **SVC** applying it in the full two-stage pipeline improved global accuracy by 2

- **MLP** gave slightly lower binary accuracy than SVC, but yielded same results when integrated into the overall two-stage pipeline.
- A **soft-voting meta-classifier** combining SVC and MLP did not outperform the other 2.

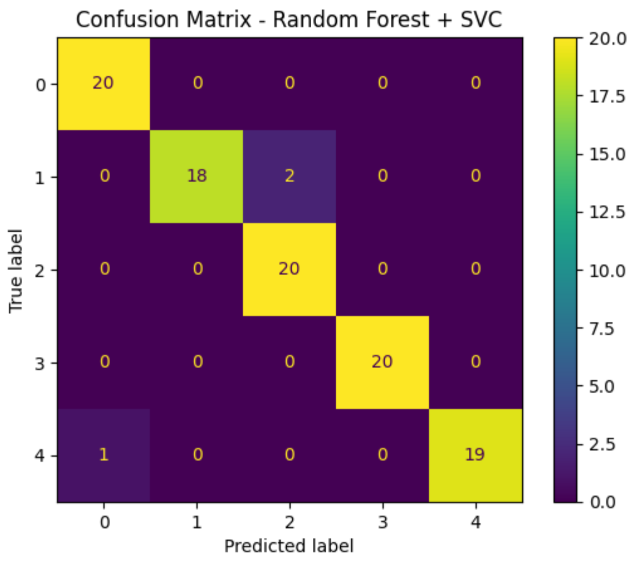


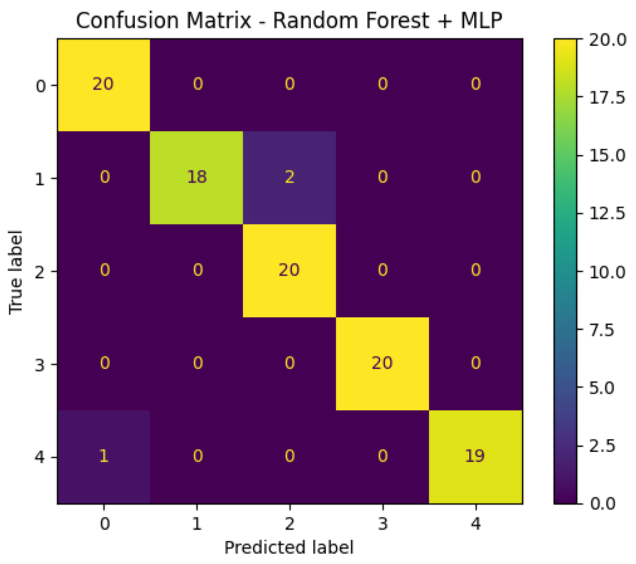
Figure 6: Confusion Matrix Random Forest + SVC.

We also investigated whether restricting the second-stage classifier to a smaller subset of features would improve performance. Based on medical literature and discussions with medical students, we selected features most relevant to distinguishing myocardial infarction from dilated cardiomyopathy, specifically those describing myocardial wall thickness at end-systole: `thickness_max_mean_ES`, `thickness_std_mean_ES`, `thickness_mean_std_ES`, and `thickness_std_std_ES`. However, classifiers trained only on these features consistently performed worse than those using the full feature set.

Therefore, the final model uses a second-stage SVC trained on all features, as it was the only configuration that improved cross-validation accuracy. This hierarchical approach preserved the strong overall performance of the Random Forest while enhancing the resolution between classes 1 and 2.

6 Conclusion

In this project, I developed a pipeline for the classification of cardiac pathologies from cardiac MRI (CMRI) data, using handcrafted features extracted from segmentation masks. By leveraging clinically meaningful anatomical and functional



True label \ Predicted label	0	1	2	3	4
0	20	0	0	0	0
1	0	18	2	0	0
2	0	0	20	0	0
3	0	0	0	20	0
4	1	0	0	0	19

Figure 7: Confusion Matrix Random Forest + MLP.

indicators—such as chamber volumes, ejection fractions, and myocardial thickness profiles—I was able to build compact and interpretable representations of each subject to train and evaluate machine learning models.

Despite the relatively limited size of the dataset and the handcrafted nature of the features, the results demonstrate promising discriminatory power and suggest that simple, well-chosen features can provide valuable insight into complex medical tasks.

Future Improvements. Several directions could be explored to further enhance performance:

- **Deep learning-based feature extraction:** Leveraging convolutional neural networks (CNNs) or transformers to learn latent features directly from the raw CMRI volumes could capture more subtle or complex patterns.
- **Temporal dynamics:** Incorporating the full cardiac cycle rather than just ED and ES phases could help model dynamic heart function more accurately.
- **Clinical metadata:** Integrating additional clinical data (e.g., age, gender, symptoms) could improve model generalization and interpretability.
- **Data augmentation and harmonization:** Especially for small datasets, improving robustness via augmentation or domain adaptation techniques may be critical.

These enhancements could help bridge the gap between classical machine learning approaches and fully automated, end-to-end diagnostic systems suitable for clinical deployment.