# Heart Disease Prediction from heart beat audio signals using Machine Learning and Network Analysis

Ligari D. • Alberti A.[1]

[1] *Department of Computer Engineering, Data Science, University of Pavia, Italy*
*Course of Advanced Biomedical Machine Learning*
Github page: https://github.com/DavideLigari01/advanced-biomedical-project

Date: June 30, 2024

**Abstract** — Heart disease remains one of the leading causes of mortality worldwide, making early diagnosis crucial. This study aims to predict heart diseases by analyzing heartbeat audio signals using machine learning and network analysis. We utilized a dataset from the PASCAL Classifying Heart Sounds Challenge 2011, which includes normal heart sounds, murmurs, extra heart sounds, extra systoles, and artifacts. Various preprocessing techniques such as noise reduction, resampling, and segmentation were applied to ensure data quality. Features were extracted using methods like Mel-Frequency Cepstral Coefficients (MFCC), Chroma, RMS, ZCR, and spectral features. Multiple machine learning models including LightGBM, XGBoost, CatBoost, Random Forest, and Multilayer Perceptron were trained and evaluated. The best performing model achieved high accuracy in distinguishing between different heart sound categories. This research highlights the potential of machine learning in cardiac diagnostics and provides a foundation for future advancements in the field.

**Keywords** ——TO BE DEFINED—

## Contents

## 1. Introduction

Heart disease remains a leading cause of mortality worldwide. Early diagnosis is critical for effective treatment and management. Traditional methods of diagnosis often involve invasive procedures and expensive equipment. Recent advancements in machine learning have opened new avenues for non-invasive diagnosis using heart sound recordings. This paper explores the use of machine learning and network analysis to predict heart disease from heart beat audio signals. Despite significant progress, gaps remain in accurately classifying heart sounds due to data imbalance and the presence of noise in recordings. This study aims to address these gaps by employing advanced data preprocessing techniques and robust machine learning models. The research question guiding this study is: How can machine learning models be optimized to improve the accuracy of heart disease prediction from heart sound recordings?

### 1.1. Problem Domain

### 1.2. Research Question

### 1.3. Previous Research

## 2. Methods

### 2.1. Source of Data

The dataset for this project was obtained from a Kaggle repository titled *Dangerous Heartbeat Dataset (DHD)* [1], which in turn sources its data from the PASCAL Classifying Heart Sounds Challenge 2011 (CHSC2011) [2]. This dataset comprises audio recordings of heartbeats, categorized into different types of heart sounds. Specifically, the dataset consists of 5 types of recordings: Normal Heart Sounds, Murmur Sounds, Extra Heart Sounds, Extrasystole Sounds, and Artifacts. Data has been gathered from the general public via the iStethoscope Pro iPhone app and from a clinic trial in hospitals using the digital stethoscope DigiScope.

#### *Type of Sources*

The dataset comprises audio recordings collected from three distinct sources:

**Type A:** This subset includes recordings contributed by the general public through the iStethoscope Pro iPhone app. Users from diverse backgrounds and locations have submitted these recordings, providing a wide range of heart sounds in various conditions.

**Type B:** This subset consists of recordings obtained from clinical trials conducted in hospitals using the DigiScope digital stethoscope. These recordings are collected in controlled environments, contributing to a high-quality dataset for clinical applications.

**Type C:** This subset is a mixed collection that includes recordings from both the iStethoscope Pro app and the DigiScope digital stethoscope. Additionally, this subset incorporates heart sound recordings sourced from various publicly available datasets on the internet. This mixed dataset is valuable for its diversity and comprehensiveness, covering a broad spectrum of heart sound variations and abnormalities.
These diverse sources ensure a robust dataset that supports comprehensive analysis and improves the generalizability of the heartbeat audio classification model.

#### *Classes*

Heart sounds can be categorized into different classes based on their acoustic characteristics and clinical significance. Accurate classification of these sounds is essential for diagnosing and treating a variety of cardiac conditions. The primary categories include Normal heart sounds, Murmurs, Extra Heart Sounds, Artifacts, and Extra Systoles. Understanding the distinct features and clinical implications of each class is a crucial step before building a machine learning model to classify heartbeats. This phase is particularly important for the identification of patterns that are characteristic of specific classes, which in turn guides the selection of features to extract from the audio. This knowledge aids in identifying specific patterns and anomalies within the heart sounds, leading to more precise and reliable model predictions.

**Normal** The Normal category includes recordings of typical, healthy heart sounds. These sounds exhibit the characteristic “lub-dub, lub-dub” pattern, where “lub” (S1) represents the closing of the atrioventricular valves and “dub” (S2) signifies the closing of the semilunar valves. In a normal heart, the time interval between “lub” and “dub” is shorter than the interval from “dub” to the next “lub,” especially when the heart rate is below 140 beats per minute. Most normal heart rates at rest fall between 60 and 100 beats per minute, though rates can vary from 40 to 140 beats per minute based on factors such as age and activity level. Recordings may include background noises like traffic or radio sounds and may capture incidental noises such as breathing or microphone contact with clothing or skin. It contains both clean and noisy normal recordings, the latter featuring significant background noise or distortion, which simulates real-world conditions.
Figure 1 shows a sample of a normal heart beat audio. The characteristic “lub-dub, lub-dub” pattern can be observed, where the peaks represent the “lub” (S1) and “dub” (S2) sounds of a healthy heart.

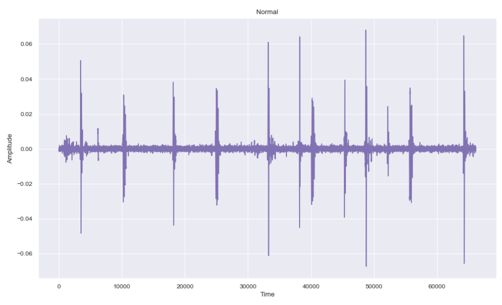

**Fig. 1:** Sample of normal heart beat audio.

**Murmur** Heart murmurs are abnormal sounds during the heartbeat cycle, such as a “whooshing, roaring, rumbling, or turbulent fluid” noise, heard between the “lub” and “dub” (systolic murmur) or between “dub” and “lub” (diastolic murmur). These murmurs are typically indicative of turbulent blood flow in the heart and can signal various heart conditions, some of which may be serious. It is crucial to distinguish murmurs from the normal “lub-dub” sounds since they occur between the primary heart sounds and not concurrently with them. It also includes noisy murmur data, which mimics real-world recording scenarios by incorporating significant background noise and distortions.
Figure 2 shows a sample of a murmur heart beat audio. The presence of additional sounds between the “lub” and “dub” peaks can be observed, indicating the characteristic “whooshing, roaring, rumbling, or turbulent fluid” noise

typical of heart murmurs.

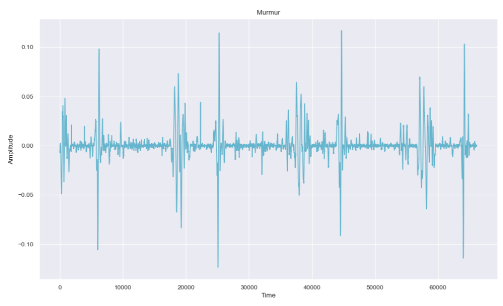

**Fig. 2:** Sample of murmur heart beat audio.

**Extra Heart Sound** Extra heart sounds are characterized by an additional sound in the cardiac cycle, producing patterns such as “lub-lub dub” or “lub dub-dub”. These sounds can arise from physiological or pathological conditions. For example, a third heart sound (S3) may indicate heart failure or volume overload, while a fourth heart sound (S4) can be associated with a stiff or hypertrophic ventricle. Detecting these extra sounds is important for identifying potential heart diseases early, allowing for timely intervention and management. Figure 3 shows a sample of an extra heart sound audio. The presence of additional peaks within the normal “lub-dub” pattern indicates extra heart sounds, which can be critical for diagnosing various heart conditions.

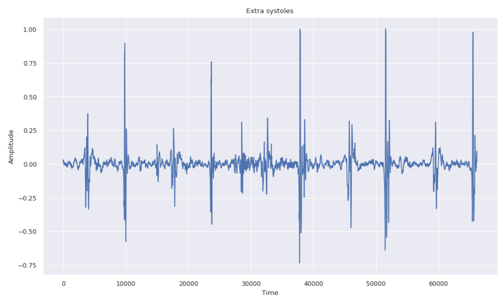

**Fig. 3:** Sample of extra heart sound audio.

**Artifact** The Artifact category consists of recordings with non-cardiac sounds, including feedback squeals, echoes, speech, music, and various types of noise. These recordings generally lack discernible heart sounds and do not exhibit the temporal periodicity typical of heartbeats at frequencies below 195 Hz. Accurately identifying artifacts is essential to avoid misinterpreting non-cardiac sounds as pathological heart sounds, ensuring that data collection efforts focus on genuine heart sounds. Figure 4 shows a sample of an artifact heart beat audio, there can be observed that there is not a clear pattern in the audio.

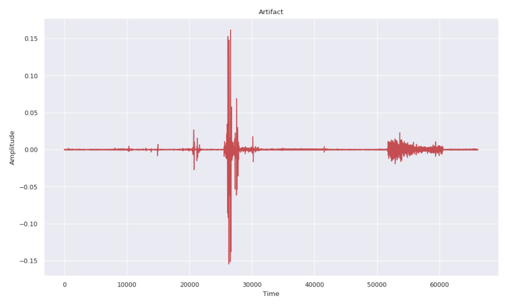

**Fig. 4:** Sample of artifact heart beat audio

**Extra systoles** Extra systoles refers to extra or skipped heartbeats, resulting in irregular patterns such as “lub-lub dub” or “lub dub-dub”. Unlike the regular extra heart sounds, extra systoles are sporadic and do not follow a consistent rhythm. These premature beats can occur in healthy individuals, particularly children, but they may also be associated with various heart diseases. Identifying extra systoles is crucial as they can be early indicators of cardiac conditions that might require medical attention if they occur frequently or in certain patterns.
In the audio signal depicted in Figure 5, irregularities within the normal “lub-dub” pattern are evident. These irregularities manifest as additional peaks or skipped beats, indicating extra systoles.

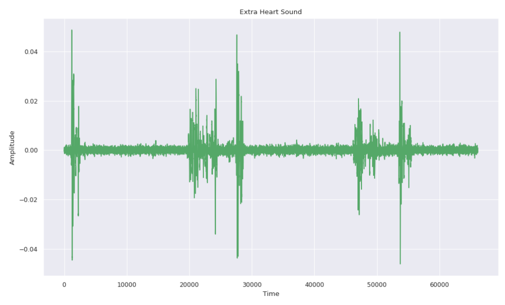

**Fig. 5:** Sample of extra systoles heart beat audio

**Comparison of Heart Sounds** In Figure 6, a comparison of the different classes of heart sounds can be observed.
As we can see, the “Artifact” signal appears erratic with no consistent pattern, likely representing noise or interference rather than true heart sounds.
The “Murmurs” signal shows irregular fluctuations in amplitude, which could indicate turbulent blood flow typically associated with murmurs.
The signal for “Extra Heart Beat Sound” has occasional spikes in amplitude that stand out from the baseline.
The “Normal” signal appears more uniform and regular compared to the others, reflecting the expected rhythm of a healthy heartbeat. FInally, the signal for “Extra Systoles” shows extra spikes at irregular intervals, indicating unexpected contractions of the heart muscle (systoles) occurring outside the normal rhythm.

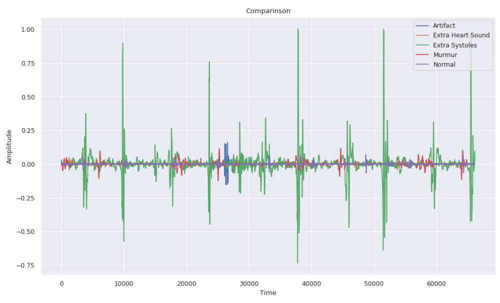

**Fig. 6:** Comparison of the different classes of heart sounds.

### *Data Distribution*

Figure 7, illustrates the significant class imbalance present in the dataset, particularly for the 'Extrastole' and 'Extrahls' classes, which have far fewer samples compared to other classes. This imbalance poses a challenge for the classification task, as the model may struggle to learn and accurately predict the underrepresented classes due to the insufficient number of training examples. To mitigate this issue, several strategies are employed.

Data augmentation techniques are applied to artificially increase the size of the dataset by creating modified versions of the existing audio files through methods such as pitch shifting, time stretching, and adding noise. Additionally, the original audio recordings are segmented into smaller clips, which not only increases the number of samples available for training but also provides the model with more varied examples of heart sounds, enhancing its ability to generalize across different heart sound variations.

Furthermore, the effectiveness of oversampling and undersampling techniques is tested. Oversampling involves duplicating samples from the minority classes to increase their representation in the training set, while undersampling involves reducing the number of samples from the majority classes to balance the dataset. The data is split into training and testing sets with an 80% - 20% ratio, respectively.

A validation set is omitted due to the low number of samples available, ensuring that the maximum amount of data is used for training and testing the model.

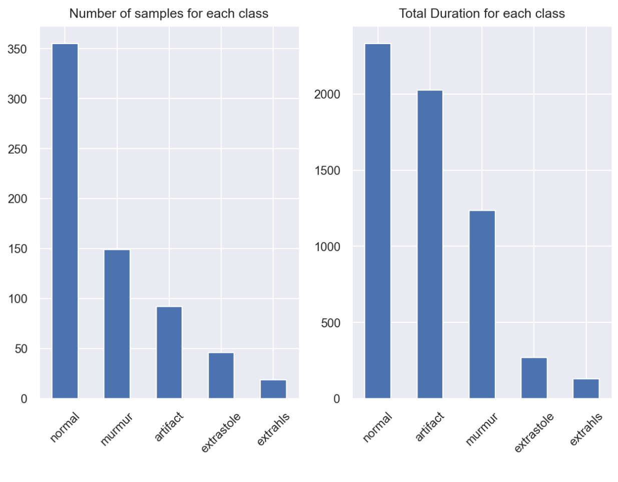

**Fig. 7:** Number of samples per duration.

## 2.2. Data Preprocessing

To prepare the data several preprocessing operations were performed:

**Noise Reduction:** the audio data was already provided in a clipped format to minimize noise and irrelevant information.

**Normalization**: the audio are loaded using the *torchaudio.load()* function, which normalized the audio signals in the range [-1, 1].

**Removal of Corrupted Files:** corrupted files were identified and removed from the dataset to ensure data quality.

**Outlier Detection and Removal:** we investigated the average duration of each class and found the 'artifact' class to have a significantly larger average duration. This was due to a few long lasting audio recordings (see Figure 8). A large number of samples from the same audio might not be as informative, thereby we used IQR to detect and remove outliers.

**Resampling:** we evaluated two sampling rates to determine the optimal rate for heartbeat sounds and all audio files were resampled to a common frequency of 4000 Hz (see Section 2.3).

**Segmentation:** the audio data was segmented into 1-second intervals, identified as the optimal extraction interval (see Section 2.3), as it offered both good performance and dataset size increasing.

**Hop and Window Size**: the hop size determines the number of samples between successive windows, while the window size determines the number of samples considered. Each feature was extracted using the same window length and hop length facilitating a fair assessment of each feature's contribution to the classification task.

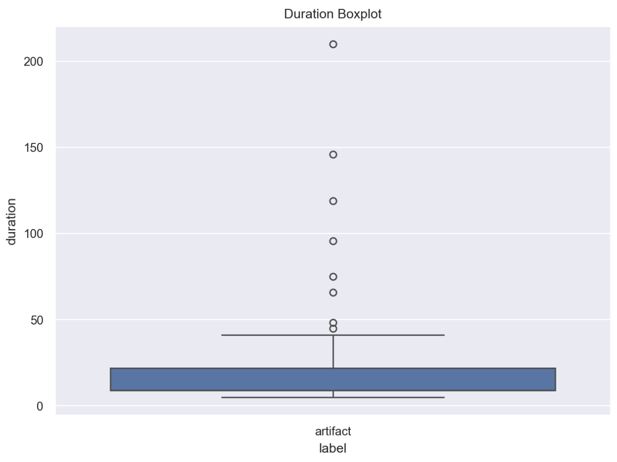

**Fig. 8:** Outliers in the Artifacts class.

## 2.3. Feature Extraction

s demonstrated by [4] and [3], MFCCs are highly effective features for heartbeat classification. In addition to MFCCs, we incorporated other features to capture various characteristics of heart sounds, enhancing the classification accuracy. The features used are explained in the following section.

## *Features Type*

**MFCC**
Mel-Frequency Cepstral Coefficients (MFCCs) are representations of the short-term power spectrum of sound. They are derived by taking the Fourier transform of a signal, mapping the powers of the spectrum onto the mel scale, taking the logarithm, and then performing a discrete cosine transform. MFCCs are effective in capturing the timbral texture of audio and are widely used in speech and audio processing due to their ability to represent the envelope of the time power spectrum. In heartbeat classification, MFFCs can reflect the different perceived quality of heart sounds, such as the presence of murmurs or other anomalies.

**Chroma STFT**
Chroma features represent the 12 different pitch classes of music (e.g., C, C#, D, etc.). They are particularly useful for capturing harmonic and melodic characteristics in music. By mapping audio signals onto the chroma scale, these features can identify pitches regardless of the octave, making them useful for analyzing harmonic content in heart sounds.

**RMS**
Root Mean Square (RMS) measures the magnitude of varying quantities, in this case, the amplitude of an audio signal. It is a straightforward way to compute the energy of the signal over a given time frame. RMS is useful in audio analysis for detecting volume changes and can help identify different types of heartbeats based on their energy levels. For example, in a given timeframe the RMS may be altered by the presence of a murmur with respect to a normal heart sound.

**ZCR**
Zero-Crossing Rate (ZCR) is the rate at which a signal changes sign, indicating how often the signal crosses the zero amplitude line. It is particularly useful for detecting the noisiness and the temporal structure of the signal. In heartbeat classification, ZCR can help differentiate between normal and abnormal sounds by highlighting changes in signal periodicity.

**CQT**
Constant-Q Transform (CQT) is a time-frequency representation with a logarithmic frequency scale, making it suitable for musical applications. Since it captures more detail at lower frequencies, it may be useful for analyzing the low-frequency components of heart sounds.

**Spectral Centroid**
The spectral centroid indicates the center of mass of the spectrum and is often perceived as the brightness of a sound. It is calculated as the weighted mean of the frequencies present in the signal, with their magnitudes as weights. In heart sound analysis, a higher spectral centroid can indicate sharper, more pronounced sounds, while a lower centroid suggests smoother sounds.

**Spectral Bandwidth**
Spectral bandwidth measures the width of the spectrum around the centroid, providing an indication of the range of frequencies present. It is calculated as the square root of the variance of the spectrum. This feature helps in understanding the spread of the frequency components in the heart sounds, which can be indicative of different heart conditions.

**Spectral Roll-off** Spectral roll-off is the frequency below which a certain percentage of the total spectral energy lies. It is typically set at 85% and helps distinguish between harmonic and non-harmonic content. In heartbeat classification, spectral roll-off can be used to differentiate between sounds with a concentrated energy distribution and those with more dispersed energy.

## *Sampling Rate Selection*

The sampling rate of the data were heterogeneous, ranging from 4000 Hz to 44100 Hz, with a majority of the data being sampled at 4000 Hz. To assess the impact of the sampling rate on the classification performance, we trained different models on different features, extracted at different sampling rates and from various intervals. Each model is then evaluated using different metrics, taking into account the class imbalance issue. We also considered a possible dependency between the sampling rate and the extraction interval, as shown in Algorithm 1.

**Algorithm 1** Sampling rate and Interval choice

```
1: Input:
2: features = [mfcc30 & 120, cqt30 & 70, chroma12]
3: sampling_rates = [mix, 4000]
4: extraction_intervals = [0.5, 1, 2, 3]
5: models = [rf, svm-rbf, lr]
6: metrics = [macrof1, mcc]
7: for sr in sampling_rates do
8:     for interval in extraction_intervals do
9:         for feature in features do
10:            extract feature with interval at sr
11:            for model in models do
12:                train model with extracted feature
13:                for metric in metrics do
14:                    evaluate model with metric
15: Output:
16: Given all the results, group by model and average the values
    of a specific metric across features
```

The results, reported in Figure 9 showed no evident advantage to using a mix of sampling frequencies over a fixed resampled sample rate. Moreover, employing a fixed sample rate of 4000 Hz reduces the risk of introducing bias, enhances efficiency, and permits the use of a broader range of features and models.

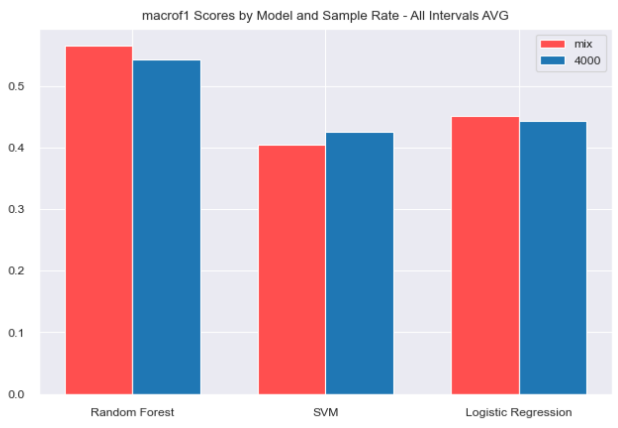

**Fig. 9:** Comparison of the macro F1 score for different sampling rates.

### *Extraction Interval Selection*

The extraction interval refers to the duration of the audio segment from which the features are extracted. Using algorithm 1, we evaluated the performance of 0.5, 1, 2, and 3-second intervals on the classification task. It is important to note that the interval choice affects the number of samples available for training and evaluation, so in case of a limited number of samples, this choice should no be based solely on the performance of the model. The results, showed that a 2-second interval yielded the best performance, however it also reduced the number of samples, impeding a correct training and evaluation of the models. As a consequence, we picked a 1-second interval as a compromise.

### *Number of Features per Type*

Now the focus, the focus is on identifying the optimal number of features for different types of audio features (MFCC, Chroma, CQT, etc.). Each type of feature can consist of a varying number of individual features, and it is crucial to determine the optimal number to maximize the model's performance. Proper feature selection is vital because it directly impacts the efficiency, accuracy, and generalizability of the machine learning model. Including too many features can lead to overfitting, increased computational costs, and degraded performance due to the curse of dimensionality. On the other hand, using too few features can result in underfitting, where the model fails to capture the necessary patterns in the data. Therefore, the goal of this study is to identify the optimal number of features for each type to ensure the model is both robust and efficient.

**Determining the Optimal Number of Features** To maximize the model's performance, it is crucial to determine the optimal number of features for each type. This was achieved using a 'One Model per Feature' approach, which involved the following steps:

**Algorithm 2** Feature Optimization Process

```
1: Step 1: Extract feature sets with varying sizes: 12, 20, 30, 40, 60, 70, 90, and 120 features.
2: Step 2: Train a separate model for each feature type and feature set size.
3: Step 3: For each feature set, train three different models using three classifiers: SVM, Random Forest, and Logistic Regression.
4: Step 4: Evaluate the performance of each model.
```

Figure 10 shows the results obtained for each type of feature with the Random Forest model, which significantly outperformed the other classifiers. From the figure, we can observe the following trends for each feature type:

- **MFCC:** The MFCC features consistently achieved the highest F1 scores, peaking around 0.7. This indicates their effectiveness for the classification task. Interestingly, the number of MFCC features (ranging from 30 to 120) did not drastically affect performance, suggesting that even a smaller set of MFCC features can be highly informative.
- **CQT:** The CQT features showed moderate performance, with F1 scores around 0.4 to 0.5. The optimal number of features was around 70, beyond which there was no significant improvement.
- **RMS:** RMS features exhibited F1 scores ranging from 0.4 to 0.5, with optimal performance achieved with around 70 features.
- **ZCR, Spectral Centroid (SC), Spectral Bandwidth (SB), and Spectral Roll-off (SR):** The F1 scores for these features generally stabilized around 0.4 to 0.5. Increasing the number of features beyond 40 did not result in significant performance gains and could even degrade the model's performance. This suggests that adding too many features, especially those without strong predictive power, can confuse the model and degrade its performance.

**Optimal Number of Features** Based on the results, the optimal number of features for each type is shown in Table 1.

| Feature Type | Optimal Number of Features |
|---|---|
| MFCC | 30 |
| Chroma | 12 |
| CQT | 70 |
| RMS | 40 |
| ZCR | 40 |
| Spectral Centroid | 40 |
| Spectral Bandwidth | 60 |
| Spectral Roll-off | 40 |

**Table 1:** Optimal number of features for each type

## 2.4. Feature Selection

Given the large number of features (338 in total), it was necessary to identify and remove features that are poorly correlated with the target variable as well as those that are highly correlated with each other. Due to the high number of features, a visual approach, such as a correlation matrix, was not feasible. Instead, two filters were applied to select the most relevant features using the Spearman correlation

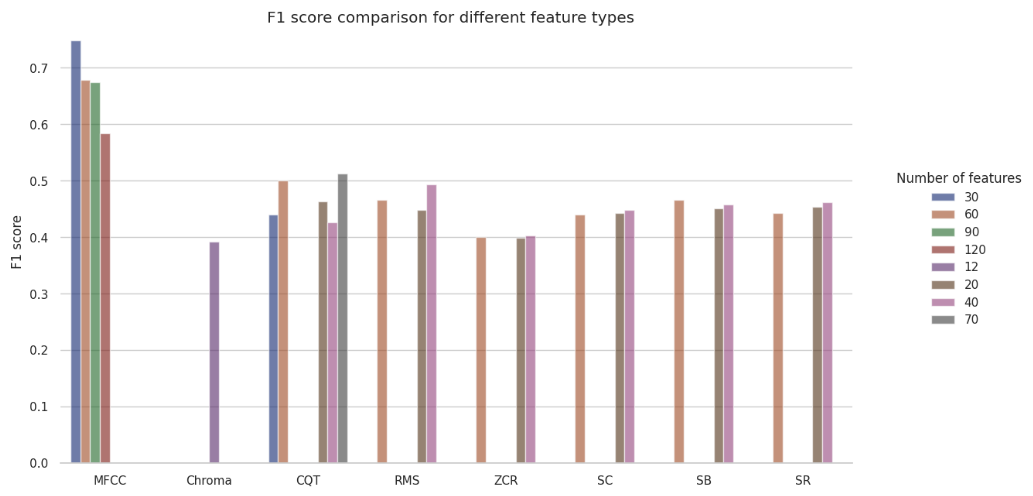

**Fig. 10:** F1 score per number of features

coefficient, as the normality test failed.
The first filter is based on the correlation between the features and the target variable. Features with a correlation below a certain threshold with the target variable are removed.
The second filter focuses on the correlation among the features themselves. It counts, for each feature, the number of other features with which it has a correlation above a certain threshold. Features with a number of correlations above a specified threshold are then removed.

**Algorithm 3** Feature Selection Process

```
1: Step 1: Compute the normal test (D'Agostino Pearson).
2: Step 2: Compute the Spearman correlation coefficient for each feature with the target variable.
3: Step 3: Apply the first filter to remove features with a correlation below a certain threshold with the target variable.
4: Step 4: Compute the correlation matrix among all features.
5: Step 5: Apply the second filter to remove features that have a high number of correlations (above a certain threshold) with other features.
6: Step 6: Choose threshold values empirically and apply the filters using various combinations of these thresholds.
7: Step 7: Train Random Forest models on the filtered data to evaluate performance and select the best combination of thresholds.
```

### *Threshold Selection and Model Evaluation*

Threshold values were chosen empirically and the filters were applied using the combinations shown in Table 2. Using the filtered data, Random Forest models were trained and evaluated, as Random Forest was found to be the best performing model. The optimal combination of thresholds was found to be: threshold 1 = 0, threshold 2 = 0.6 and number of features = 30, resulting in 30 features.

| Threshold | Values |
|---|---|
| THRESHOLD 1 | 0 - 0.1 - 0.2 - 0.3 - 0.4 - 0.5 |
| THRESHOLD 2 | 0.6 - 0.7 - 0.8 - 0.9 - 1 |
| N° FEATURES | 5 - 10 - 15 - 20 - 25 - 30 - 40 |

**Table 2:** threshold values

With threshold 1 = 0, the filter on the correlation between the features and the target variable was effectively bypassed. However, with threshold 2 = 0.6, a stringent filter was applied on the correlation among the features themselves, removing features that had a correlation above 0.6 with at least 30 other features. This indicates that having features highly correlated with each other is more detrimental to the model than having features poorly correlated with the target variable.
Figure 11 shows the results obtained with the model trained on filtered features compared to the model trained on all features. As demonstrated, the model trained on filtered features performs significantly better.

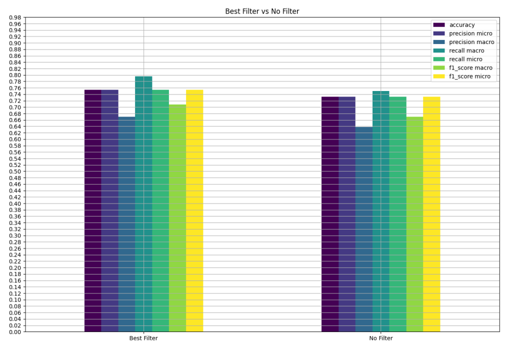

**Fig. 11:** Comparison of different metrics between the model on all features and the model on the filtered ones

### *Selected Features and Correlation Matrix*

From this analysis, 41 features remained: 28 MFCC, 12 Chroma, and 1 ZCR. The correlation matrix of the filtered features is shown in Figure 12. This matrix illustrates the pairwise correlation between the selected features, with the color intensity indicating the strength and direction of the correlation. Dark red cells represent high positive correlations, while dark blue cells indicate high negative correlations.

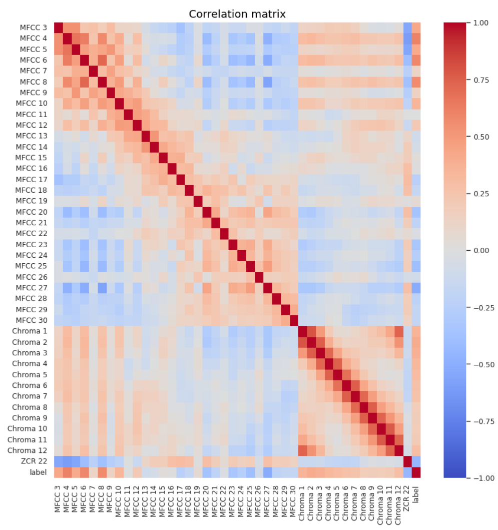

**Fig. 12:** Correlation matrix of the filtered features

The matrix demonstrates that the remaining features have low correlations with each other, as evidenced by the predominantly light colors away from the diagonal. This implies that the features are relatively uncorrelated, preventing multicollinearity issues and enhancing the robustness of the model. The high diagonal values indicate that each feature is perfectly correlated with itself, which is expected. However, the off-diagonal values being close to zero for most feature pairs confirm that the filtering process was effective in selecting features that do not exhibit high intercorrelations.

## 2.5. Models

### *Metrics*

In this project, several metrics are used to evaluate the performance of the heartbeat audio classification model. These metrics provide insights into various aspects of model performance, particularly in the context of multiclass classification with highly imbalanced classes.

**Accuracy** The first metric considered is *accuracy*, which is defined as the ratio of the number of correct predictions to the total number of predictions. Mathematically, it is expressed as:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

Accuracy provides a straightforward measure of overall model performance and is easy to understand and compute. However, in the presence of class imbalance, accuracy can be misleading because it does not account for the distribution of different classes, potentially giving a false sense of high performance when the model is biased towards the majority class.

**Balanced Accuracy** Balanced accuracy tries to address the limitations of accuracy in the case of imbalanced classes. It is the average of recall obtained on each class. It is calculated as:

$$\text{Balanced Accuracy} = \frac{1}{N}\sum_{i=1}^{N}\frac{TP_i}{TP_i + FN_i}$$

where $N$ is the number of classes, $TP_i$ is the true positives for class $i$, and $FN_i$ is the false negatives for class $i$. Balanced accuracy handles class imbalance better by giving equal weight to each class, providing a more nuanced measure of performance across all classes.

**Matthews Correlation Coefficient (MCC)** The Matthews Correlation Coefficient (MCC) is a robust metric used to evaluate the performance of classification models, especially useful in the presence of imbalanced classes. For multiclass classification problems, the MCC provides a balanced measure that takes into account the correct and incorrect predictions across all classes. The MCC for multiclass problems is calculated as:

$$\text{MCC} = \frac{\sum_k \sum_l \sum_m C_{kk}C_{lm} - C_{kl}C_{mk}}{\sqrt{\sum_k \left(\sum_l C_{kl}\right)\left(\sum_{k'\neq k}\sum_{l'} C_{k'l'}\right)}\sqrt{\sum_k \left(\sum_l C_{lk}\right)\left(\sum_{k'\neq k}\sum_{l'} C_{l'k'}\right)}}$$

This formula can be more easily understood by defining intermediate variables:

- $t_k = \sum_i C_{ik}$: the number of times class $k$ truly occurred,
- $p_k = \sum_i C_{ki}$: the number of times class $k$ was predicted,
- $c = \sum_k C_{kk}$: the total number of samples correctly predicted,
- $s = \sum_i \sum_j C_{ij}$: the total number of samples.

This allows the formula to be expressed as:

$$\text{MCC} = \frac{cs - \vec{t}\cdot\vec{p}}{\sqrt{s^2 - \vec{p}\cdot\vec{p}}\sqrt{s^2 - \vec{t}\cdot\vec{t}}}$$

The MCC ranges from -1 to 1:

- An MCC of 1 indicates perfect prediction, where the model correctly identifies all instances across all classes.
- An MCC of 0 indicates no better performance than random chance.
- An MCC of -1 indicates total disagreement between predictions and actual outcomes.

The multiclass MCC is especially valuable because it considers the distribution of errors across all classes, providing a more comprehensive and balanced measure of performance than simpler metrics like accuracy, which can be misleading in imbalanced datasets. By taking into account the correct and incorrect predictions for each class, the MCC ensures that performance is evaluated equitably across all classes, making it a reliable indicator of overall model quality in multiclass classification tasks.

**Precision and Recall** Precision and recall are fundamental metrics used to evaluate the performance of classification models, particularly in scenarios with imbalanced class distributions. They provide insights into different aspects of how well a model distinguishes between classes. Precision measures the accuracy of positive predictions. It is calculated as:

$$\text{Precision} = \frac{TP}{TP + FP}$$

where $TP$ (True Positives) is the number of correctly predicted positive instances, and $FP$ (False Positives) is the number of incorrectly predicted positive instances. Precision answers the question: 'Out of all instances predicted as positive, how many are actually positive' A high precision indicates that when the model predicts a positive result, it is likely to be correct.

Recall, also known as sensitivity or true positive rate, measures the proportion of actual positives that are correctly identified by the model. It is calculated as:

$$\text{Recall} = \frac{TP}{TP + FN}$$

where $FN$ (False Negatives) is the number of incorrectly predicted negative instances. Recall answers the question: 'Out of all actual positive instances, how many did the model correctly predict as positive' A high recall indicates that the model is effectively capturing all positive instances, minimizing false negatives.
In the context of imbalanced datasets, where one class may significantly outnumber another, both precision and recall become important.
The micro variant aggregates these metrics across all classes, providing an overall measure that considers the total number of true positives, false positives, and false negatives across the entire dataset.
Conversely, the macro variant computes precision and recall for each class independently and then averages them. This method is valuable for understanding the performance of the model on individual classes, particularly when class distribution varies significantly. It ensures that the evaluation does not disproportionately favor the majority class, thereby providing a balanced view of model performance across all classes.

**F1 Score** The F1 score is a metric that combines both precision and recall into a single value, offering a balanced assessment of a model's performance. It is defined as:

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Precision measures the accuracy of positive predictions, while recall measures the coverage of actual positives by the model. The F1 score effectively balances these two metrics, making it particularly useful in scenarios where both precision and recall are equally important.
The F1 score can be computed in two ways: *micro* and *macro*. The micro F1 score aggregates the contributions of all classes to compute a single F1 score. On the other hand, the macro F1 score computes the F1 score for each class independently and then averages these scores. This approach is crucial in the context of class imbalance because it ensures that the performance on minority classes receives equal weight and is not overshadowed by the majority class.

**Receiver Operating Characteristic (ROC) Curve and Area Under the Curve (AUC)** For binary classification scenarios, the *ROC curve* and the *AUC* (Area Under the Curve) are used. The ROC curve is a plot of the true positive rate (recall) against the false positive rate, showing the trade-off between sensitivity and specificity. The AUC provides a single value summarizing the overall ability of the model to discriminate between positive and negative classes, with a higher AUC indicating better performance. These metrics are particularly useful for binary classification tasks, providing a visual and quantitative assessment of model performance.

**Risk Score** The risk score is a custom metric designed to evaluate the model's performance in predicting heart diseases, particularly focusing on avoiding the misclassification of heart disease patients as normal. Misclassifying heart disease as normal can lead to incorrect diagnoses and inadequate treatment, posing serious health risks.
The metric measures the proportion of heart disease patients misclassified as normal.
It is intended to increase with the number of false positives (cases of heart disease classified as normal) and also as the number of samples predicted as normal decreases. A lower risk score indicates better model performance in minimizing the misclassification of heart disease as normal. The risk score is calculated using the formula:

$$\text{Risk score} = \frac{FP}{FP + TP}$$

where $FP$ (False Positives) is the number of heart disease samples misclassified as normal, and $TP$ (True Positives) is the number of normal samples correctly classified.

Additionally, the risk score can be defined for each specific heart disease to provide a more detailed understanding of the model's performance for different types of heart conditions. The disease-specific risk score is calculated as:

$$\text{Risk score}_i = \frac{FP_i}{FP + TP}$$

where $FP_i$ is the number of samples of heart disease $i$ misclassified as normal, and $FP + TP$ is the total number of samples classified as normal.

### *Prevention Model*

The goal of the prevention model is to provide an accessible tool for the early diagnosis of heart diseases, potentially usable by non-experts. Therefore, it is crucial to develop a model that minimizes the number of false normal predictions to accurately indicate the presence or not of disease or identify artifacts in the provided data.

To achieve this, different heart diseases were grouped together, transforming the problem into a 3-class classification task: normal, disease, and artifact. Grouping the diseases not only simplified the classification but also balanced the class distribution. The data was divided into training and testing sets in an 80-20 ratio, and various models were evaluated, as shown in Table 3.

The primary metrics for evaluating the models were the ROC-AUC score, false positive rate (FPR), and true positive rate (TPR), with F1-score and accuracy as secondary metrics. To adapt binary metrics for multi-class classification, the one-vs-rest strategy was employed. Specifically, we focused on the normal-vs-rest case to minimize false normal predictions.

In summary, each model was trained on the 3-class classification problem but was evaluated based on its binary classification performance (normal-vs-rest). The best model was selected based on its ROC-AUC score and performance at specific FPR levels (1%, 5%, 10%, and 20%). The objective was to minimize false normal predictions while maximizing true normals. A model predicting no cases as normal to achieve a 0% FPR would be ineffective.

### ***Support Model***

### ***2.5.1. Experimented Architectures***

| **Name** | **Architecture (hidden layers)** |
|---|---|
| Random Forest | - |
| XGBoost | - |
| CatBoost | - |
| LightGBM | - |
| MLP_Basic | (128, 64, 32) |
| MLP_Ultra | (512, 256, 128, 64, 32) |
| MLP_Large | (256, 128, 64, 32) |
| MLP_Small | (64, 32) |
| MLP_Tiny | (32, 16) |
| MLP_Reverse | (32, 64, 128, 256, 512, 256, 128, 64, 32) |
| MLP_Bottleneck | (512, 64, 32) |
| MLP_Rollercoaster | (512, 128, 256, 128, 256, 64, 32) |
| MLP_Hourglass | (512, 256, 128, 64, 32, 64, 128, 256, 512) |
| MLP_Pyramid | (1024, 512, 256, 128, 128, 128, 64, 32) |
| MLP_Wide | (1024, 1024) |
| MLP_WideUltra | (1024, 1024, 128, 32) |
| MLP_Sparse | (32, 16, 8) |
| MLP_Dropout | (128, 64, 32) |
| MLP_Ensemble1 | MLP_Basic, Large, Ultra |
| MLP_Ensemble2 | RandomForest, MLP_Ultra |
| MLP_Ensemble3 | MLP_Rollercoaster, Large |
| MLP_Ensemble4 | MLP_Rollercoaster, Large, Ultra |
| MLP_Ensemble5 | RandomForest, MLP_Ultra, Rollercoaster |
| MLP_Ensemble6 | MLP_Rollercoaster, Large, Ultra, Wide |
| ALL_Ensemble | All models majority vote |
| CB_ALL_Ensemble | All models CatBoost |

**Table 3:** Models names and architectures.

The architectures of the models used in the experiments are detailed in Table 3. Special attention is given to the ensemble models, which combine predictions from multiple models to enhance overall performance.
All MLP_Ensemble models consist of the individual models listed in their architecture name. These models' predictions are combined using a soft voting strategy, where the final prediction is determined by the argmax of the sum of the predicted probabilities from each model. This approach is effective when the models are well-calibrated and exhibit complementary strengths and weaknesses.
The ALL_Ensemble model aggregates the predictions of all individual models using a majority vote strategy. In contrast, the CB_ALL_Ensemble model also considers all individual models but uses a CatBoost model to aggregate the predictions. This allows for a more flexible voting strategy, potentially leading to improved performance.

## 2.6. Tools and Software

This study utilized several powerful libraries and tools for data processing, model training, and evaluation:

- **Scikit-learn**: Used for MLP, RF, and metrics such as F1, Balanced Accuracy, Accuracy, MCC, ROC, AUC, permutation importance, train-test split, confusion matrix and voting classifiers.
- **Torchaudio**: Used to load the audio, for MFCC extraction and audio resampling.
- **Librosa**: Used for other features extraction and audio processing and augmentation.
- **Imblearn**: Applied for handling imbalanced datasets with techniques such as undersampling, oversampling, and SMOTEN.
- **Numpy**: Essential for numerical computations and array manipulations.
- **Pandas**: Crucial for data manipulation and analysis.
- **Matplotlib**: Employed for creating visualizations.
- **Seaborn**: Used for statistical data visualization.
- **Scipy**: Utilized for scientific and technical computing.
- **XGBoost**: Implemented for gradient boosting models.
- **CatBoost**: Applied for gradient boosting on decision trees.
- **PyTorch**: Used for developing and training CNN models, specifically VGG16_bn.
- **TensorFlow**: Used for building and training deep learning models, including CNNs.
- **Keras**: High-level API for building and training neural networks on TensorFlow.
- **Shap**: Utilized for model interpretability.
- **Other Utility Libraries**: Includes `joblib` for model serialization, and `os`, `sys` for system operations and file handling.

## 3. Results

### 3.1. Prevention Model

The initial evaluation is presented using the ROC curves of the normal class versus the others for selected models. According to Figure 13, the MLP models outperformed the other models, as indicated by the higher AUC values. Specifically, *MLP_Ensemble5* achieved the highest AUC value of 0.96, followed by *MLP_Ultra*, *MLP_Rollercoaster*, *MLP_Ensemble2*, and *MLP_Ensemble4*, all with an AUC of 0.95.

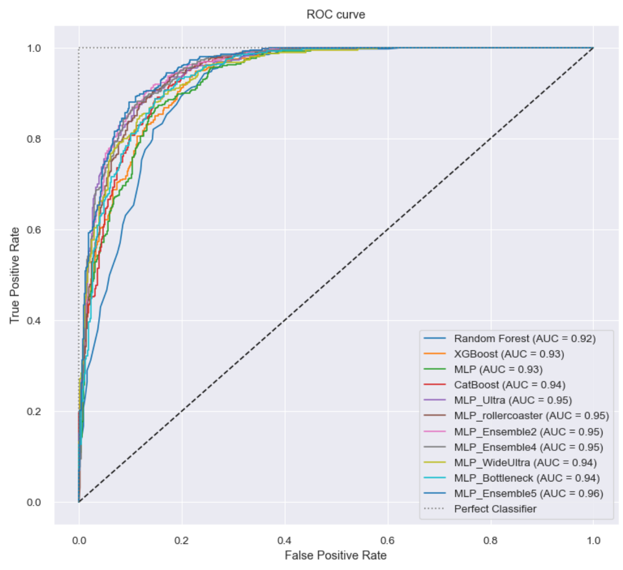

**Fig. 13:** ROC curves for the normal class against the rest of the classes across all models.

To further analyze model performance, we selected four significative FPR levels (1%, 5%, 10%, 20%) and calculated the corresponding TPR. The consolidated results are shown in Figure 14.

At each FPR level, *MLP_Ensemble5* outperformed the other models, achieving TPRs of 43.4%, 74.3%, 86.6%, and 95.8% at the 1%, 5%, 10%, and 20% FPR levels, respectively. Excluding *MLP_Ensemble5*, the best-performing model varied by FPR level: *MLP_WideUltra* at 1%, *MLP_Ultra* at 5%, *MLP_Ensemble2* at 10%, and *MLP_Ensemble4* at 20%.

These outcomes highlight the task's challenges in creating a model that performs well across all FPR levels and demonstrate the efficacy of a well-built ensemble model, which leverages the strengths of different models to achieve optimal performance.

#### *Best Model Analysis*

To understand the performance of the ensemble model (detailed architecture depicted in Figure 17), we compared its confusion matrix against the confusion matrices of the individual models that compose it (Figure 15).

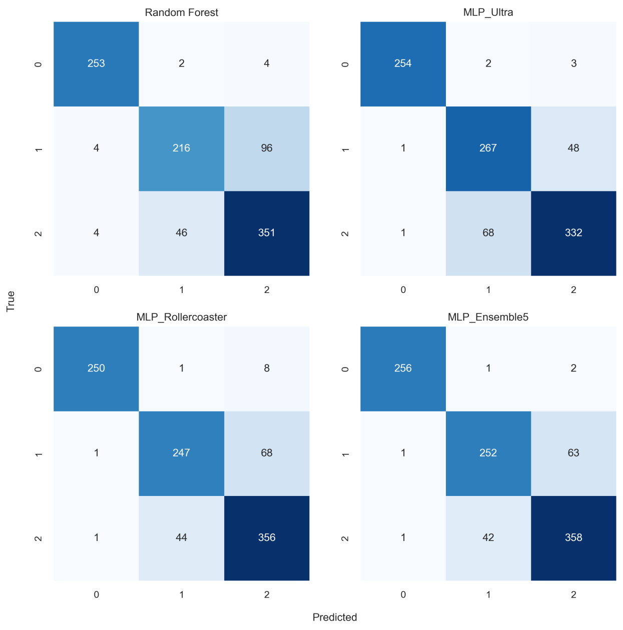

**Fig. 15:** Confusion matrices of the individual models and the ensemble model.

In the confusion matrix, class 2 represents normal heartbeats, class 1 represents abnormal heartbeats, and class 0 represents artifacts. The Random Forest and MLP_Ultra models exhibit complementary strengths: the Random Forest works better in recognizing normal samples, while the MLP_Ultra model performs better in identifying abnormal samples. The ensemble model effectively combines these strengths.

The contribution of the MLP_Rollercoaster model to the ensemble's performance is less apparent but has been experimentally demonstrated. This may be due to its ability to correctly classify some samples that are misclassified by the other models.

Notably, the MLP_Ultra model classifies fewer abnormal heartbeats as normal compared to the MLP_Ensemble5 model. However, this is because the MLP_Ultra model simply classifies fewer samples as normal. Minimizing the false positive rate (FPR) for the normal class by classifying all samples as abnormal would result in a very low true positive rate (TPR) for the normal class. This highlights the importance of analyzing FPR and TPR together.

In conclusion, the ensemble model is the most effective for classifying normal heartbeats, abnormal heartbeats, and artifacts, achieving an optimal trade-off between FPR and TPR.

### 3.2. Support Model

#### *Explainability*

### 3.3. Other Experiments

To further explore the classification problem, we conducted additional experiments involving CNN-based feature extraction, data augmentation, and a novel approach using a tiered ensemble model.

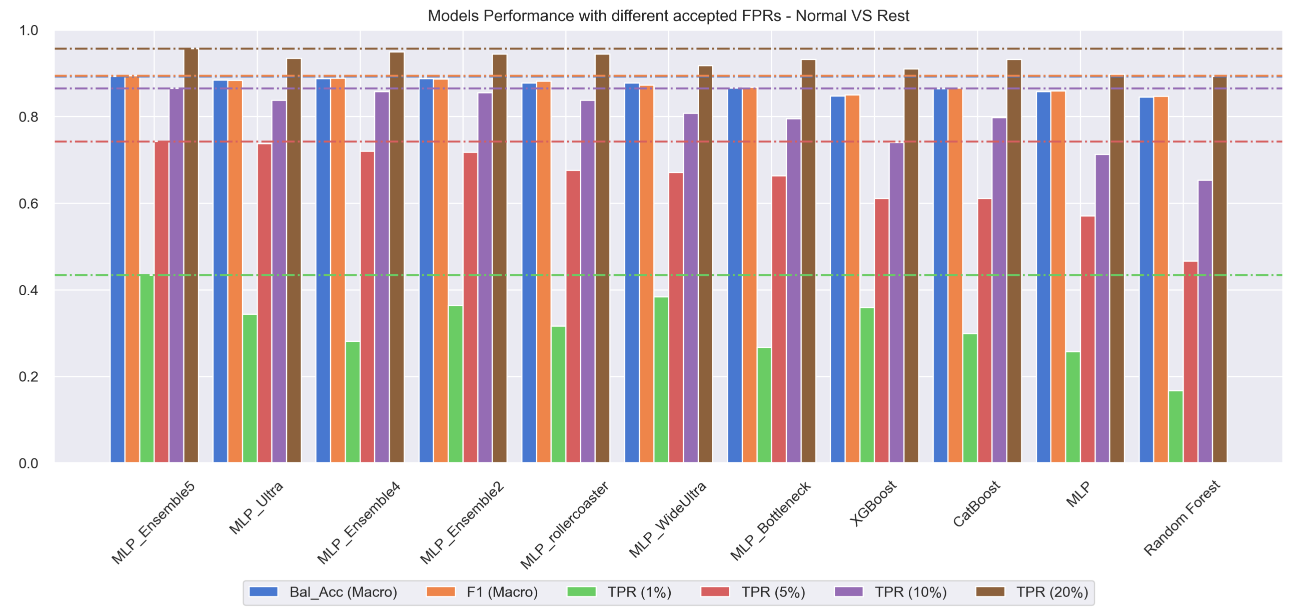

**Fig. 14:** TPR at different FPR levels for all models.

### *3.3.1. CNN-Based Experiments*

We conducted a series of experiments using Convolutional Neural Networks (CNNs) to explore their effectiveness as feature extractors for the 5-class classification problem. The CNN was used with ImageNet weights and was not fine-tuned.

- **VGG16 with Spectral Features**: VGG16 CNN was employed as a feature extractor with spectral features (MFCC, CQT, Chroma STFT, among others) used as input images. This approach yielded a Macro F1 score of approximately 65%, which is lower than the performance achieved in the primary work.

- **VGG16 with Raw Waveform Images**: VGG16 was also used to extract features from raw waveform images. Features were taken from the 5th, 4th, and 3rd convolutional layers after pooling, and various classifiers (RF, SVM, MLP) were tested on these features. This method resulted in a performance of around 67%.

### *3.3.2. Data Augmentation*

We explored data augmentation techniques to address the limited and imbalanced dataset and improve the model's generalization capabilities.

- **Noise Addition and Speed/Pitch Alteration**: We augmented the data by adding random noise (factor 0.05) and altering speed and pitch. However, the improvement in performance was not significant. This may be due to the limited size and inherent imbalance of the dataset, as data augmentation did not alter the class distribution.

- **Synthetic Data Generation with SMOTEN**: Synthetic data generation was applied to the less represented classes using SMOTEN. Although there was an initial spike in performance metrics, this was identified as bias. The model could easily distinguish the synthetically generated data from the original data. The underlying issue was the limited size of the original dataset, which did not provide sufficient variability for the synthetic generation algorithm to produce realistic and diverse samples.

### *3.3.3. Tiered Ensemble Model*

We attempted to decompose the classification problem into two sub-problems, according to Figure 16.

- **Sub-problem 1: Artifact, Normal, and Abnormal Classification**: Various models were tested for distinguishing between artifact, normal, and abnormal audio. The best model (MLP_Ensemble5) achieved a balanced accuracy of approximately 89.2%.

- **Sub-problem 2: Disease Classification**: Different models were also applied to distinguish between different diseases, achieving a balanced accuracy of more than 90.3% with MLP_Ensemble2.

- **Final Ensemble Model**: A third model (CatBoost) was used to integrate the predictions from the above sub-problems and make the final classification among the five classes. This ensemble approach resulted in a balanced accuracy of 80.5%.

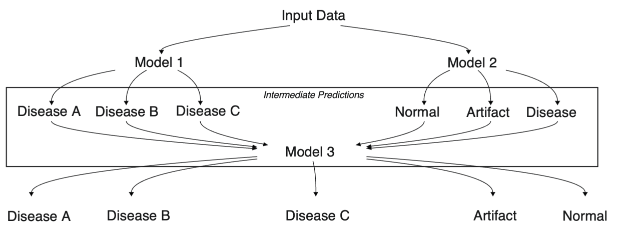

**Fig. 16:** Tiered Ensemble Model

Despite the promising results, the tiered ensemble model did not outperform the primary models.

## 4. DISCUSSION

### 4.1. Positioning in Existing Research

### 4.2. Limitations

## 5. CONCLUSION

### 5.1. Overall Impression

### 5.2. Future Work

## 6. APPENDIX

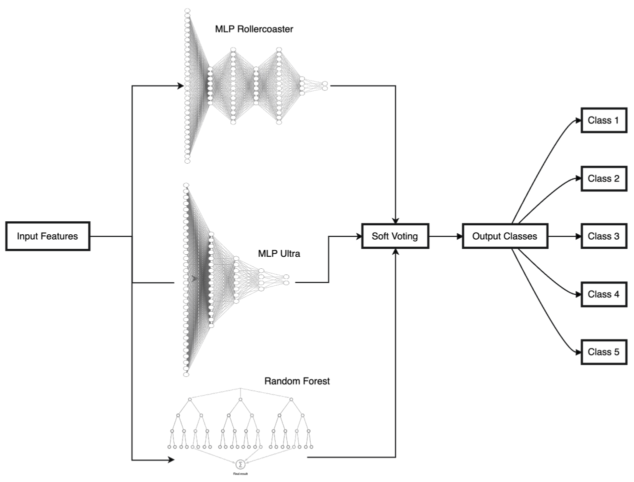

**Fig. 17:** MLP Ensemble5 Architecture

## References

[1] en. URL: https://www.kaggle.com/datasets/mersico/dangerous-heartbeat-dataset-dhd.

[2] P. Bentley et al. *The PASCAL Classifying Heart Sounds Challenge 2011 (CHSC2011) Results*. URL: http://www.peterjbentley.com/heartchallenge/index.html.

[3] Wei Chen et al. "Deep Learning Methods for Heart Sounds Classification: A Systematic Review". In: *Entropy* 23.6 (May 2021), p. 667. ISSN: 1099-4300. DOI: 10.3390/e23060667.

[4] Ali Raza et al. "Heartbeat Sound Signal Classification Using Deep Learning". en. In: *Sensors* 19.2121 (Jan. 2019), p. 4819. ISSN: 1424-8220. DOI: 10.3390/s19214819.