DataScientest

Detecting Heartbeat Normality and Abnormality Using Machine and Deep Learning Techniques

Team members: Ping Yuan[1], Dr. Sepideh Tabrik[2], Isabell Gurstein[3]

[1] ping.yuan.paris@gmail.com

[2] sepideh.tabrik@gmail.com

[3] i.gurstein@gmail.com

Mentor: Francesco Madrisotti

GitHub repository: https://github.com/Ping-YUAN/heartbeat-analysis-ai

Abstract:

In this report, we use advanced machine learning and deep learning techniques to detect whether a heartbeat is normal or abnormal. It aims to improve the accuracy of Heartbeat detection and classification within general supervision by using both traditional machine learning techniques, as well as where more advanced Deep Learning models are powerful. We provide a comprehensive overview of the methodologies employed, including data exploration, data preprocessing, feature engineering, model selection, and evaluation metrics. Overall, our results show promising efficacy of these techniques to improve diagnostic accuracy and possibly help in early diagnosis of cardiac anomalies. Supported by expert mentorship, our group work has resulted in detailed discussions and encouraging outcomes that could serve as groundwork for further research on medical diagnosis.

Introduction:

Electrocardiogram (ECG) signals are important diagnostic tools for monitoring a patient's heart health. These signals reflect the electric activity of the heart and can provide vital information about both normal and pathological heart functioning. Normal ECG signals show a particular pattern that represents every phase of a cardiac cycle, but deviations from this pattern may mean something is not right or there exists a possible abnormality in the heart. Abnormal ECG signals present as waveform anomalies, absence or extra beats, or irregular rhythms among others. Such abnormalities could indicate heart failure, arrhythmias, coronary artery disease, and other conditions.

We aimed to develop an intelligent model to classify these ECGs into two groups-normal and abnormal. By doing so, we might be able to develop a tool that might be useful in earlier detection of cardiovascular diseases, hence improving patients' outcomes.

To begin with, we explored and collated our dataset before any necessary pre-processing was done on it. Afterward, we evaluated our model's performance following its training using various machine learning algorithms and deep learning techniques. ECG signal manual analysis can take a long time

and demands a high level of skill. This is where deep learning (DL) and machine learning (ML) come in handy.

ML and DL models are the best suited for assessing ECG signals because they can be trained to identify patterns within large datasets. For numerous benefits, develop a model that differentiates normal from abnormal ECG signals. With the evaluation of ECG signals by ML and DL models rather than humans, much data is treated quickly. Comparatively, machine learning (ML) and deep learning (DL) algorithms outperform manual analysis when equipped with proper training data. Continuous monitoring of ECG signals using ML and DL tools helps detect cardiovascular problems before they turn into serious health issues. Also, the development of these models on different platforms like smartphones and wearables will make cardiac monitoring more accessible.

Conclusively, there is much promise for using ML and DL in ECG signal processing for heart health monitoring as well as disease diagnosis purposes. It is also a milestone towards identification and intervention for various medical cases.

The ongoing Heartbeat Analysis project is an international collaboration that transcends business environments and national borders. Our team consists of Ping Yuan from France, Dr. Sepideh Tabrik, and Isabell Gurstein from Germany, each with their own set of viewpoints and skills.

Objective:

The main objectives of the project are to design, train, and optimize deep learning models for analysing and classifying heartbeat sounds. Additionally, the project aims to provide team members with the opportunity to learn and apply deep learning techniques, equipping them with the skills to implement these technologies in their own professional fields. Furthermore, the project seeks to facilitate international collaboration, leveraging diverse expertise to enhance the overall outcomes.

Ping Yuan works at Datategy, a startup that builds daily tools for data scientists. With about 8 years of development experience, he has rich expertise in implementing features in code. He is interested in data science topics. His goal is to enhance his knowledge of mathematics to improve his understanding and application of data science concepts in daily life. He hopes to have the opportunity to switch his career path to data science after these classes. He has built a front-end to visualize heartbeats to assist in medical analysis. However, there are no AI or Machine Learning tools to help with predictions. He hopes that with this project, we can build a Docker image containing the model we built, along with a server and UI, so that medical professionals can load data through the UI and receive diagnostic assistance.

Dr. Sepideh Tabrik, throughout her academic journey, has acquired broad knowledge in two distinct fields of study. She focused on developing methods for breast cancer detection using mammography during her master's degree studies. In her Ph.D. study, she concentrated on the neuroscience of neuroimaging, specifically how the brain perceives objects across different sensory modalities and exhibits plasticity in its response to these stimuli. She used functional magnetic resonance imaging (fMRI) data to explore how the brain processes and classifies stimuli from two sensory system modalities. Her fMRI dataset allowed her to acquire expertise in fMRI data acquisition, preprocessing,

and analysis and an understanding of cognitive and sensory processing within the brain. She is currently on maternity leave but expects to gather new knowledge about data science that will enable her transition into industrial employment in Germany. Since she is from a medical background, she would like to be part of the current project. This position would allow her to apply what she knows and build her skills, especially deep learning abilities.

Isabell Gurstein works as an SAP BI/Data Analyst at an SAP consulting company in Germany, specializing in the retail sector. Her goal is to transition into a data scientist role and eventually a machine learning engineer position within her current company. Although her current work primarily revolves around the retail industry, this project aligns with her original background in healthcare. She chose this project because it offers a unique opportunity to apply deep learning models to the analysis of sounds, allowing her to gain complementary skills that she can later implement in her company. With several years of experience in medical invoice auditing, two years as a research associate in geronto-psychiatric clinical trials, and an additional two years in IT project management at a health insurance company in Germany, Isabell has developed a strong expertise in the healthcare field and statistical analysis. She has also completed several courses in Machine Learning as part of her Master of Science degree, positioning her as a late beginner to early intermediate in machine learning proficiency. In her current role within the AI team, Isabell carries out machine learning algorithms and provides the chatbot with Business Intelligence (BI) data from the SAP context, enriching the bot's answers and enhancing its functionality. Despite her involvement in this advanced work, she has not yet consulted the machine learning expert on her team regarding this project. Through this initiative, Isabell aims to broaden her company's spectrum of clients and methodologies by integrating advanced data science techniques. She is excited about the potential to apply what she learns in this healthcare-focused project to innovate and expand the services offered by her company in the retail sector and beyond.

Rendering 1: exploration, data visualization, and data pre-processing report

These procedures are critical for converting raw data into useful insights and actionable information. This study will provide a detailed review of the techniques and methodologies used to explore datasets, illustrate data trends, and prepare data for further research.

Framework:

In this experiment, we utilize the freely available Heartbeat dataset from Kaggle. This dataset, curated by Shayan Fazeli, has a significant number of ECG signals classified into various classifications depending on their properties. The major classifications are 'Normal' and many types of 'Abnormal' heartbeats.

We simultaneously worked on two ECG Heartbeat Categorization Datasets obtained and cleaned beforehand for analysis. The first dataset originated from the MIT-BIH Arrhythmia Database (mitbih), while the second was based on the PTB Diagnostic ECG Database (ptbdb). Both the original and

cleaned datasets are freely accessible, facilitating easy training and testing of our model on the prepared datasets and validation on the raw ones.

MIT-BIH Arrhythmia Database includes 48 half-hour ECG recordings from 47 individuals, recorded between 1975-1979. 23 recordings were randomly selected from 4000, representing a mix of hospital patients and visitors. The other 25 recordings were chosen for their rare but important heart conditions. Each recording was digitized at a rate of 360 samples per second with 11-bit resolution within a 10 mV range. Over 110,000 annotations were made by cardiologists, providing a reference for each heartbeat. Two databases are available at https://www.kaggle.com/datasets/shayanfazeli/heartbeat, mitbih_train.csv, and mitbih_test.csv.

The mitbih_train.csv file is a crucial part of the ECG Heartbeat Categorization Dataset used for training models to recognize different types of heartbeats. Here are the detailed aspects of this file:

- Size and Structure: The file is quite large, with a size of 392 MB. It contains 87,554 rows, each representing a single heartbeat.
- Columns: There are 188 columns in the file. The first 187 columns represent the signal values at different time points, and the last column is the target label for the heartbeat class.
- Classes: The target column categorizes heartbeats into five classes:
  - 0: Normal heartbeats ("Normal")
  - 1: Supraventricular premature beats ("Supraventricular")
  - 2: Ventricular escape beats ("Ventricular")
  - 3: Fusion of ventricular and normal beats ("Fusion")
  - 4: Unclassified beats ("Unclassifiable")
- Class Imbalance: The dataset is known to be imbalanced, with a significant majority of the heartbeats labelled as normal (class "0"). This imbalance needs to be addressed during the model training process to prevent bias.

The mitbih_test.csv file is the testing counterpart to the mitbih_train.csv file in the ECG Heartbeat Categorization Dataset. Here are the details:

- Size and Structure: The file is approximately 98.1 MB in size. It contains 21,892 rows, with each row representing a single heartbeat.
- Columns: Similar to the training file, there are 188 columns. The first 187 columns represent the individual time points of the ECG signal, and the final column is the class label[2].
- Classes: The class labels are the same as in the training set, with five categories ranging from normal heartbeats to various types of arrhythmias.
- Usage: This file is used to evaluate the performance of the model trained on the mitbih_train.csv data. It helps in assessing how well the model generalizes to new, unseen data.

For PTB Diagnostic ECG Database a specialized recorder with 16 input channels was used, including 14 for ECGs, one for respiration, and one for line voltage. The recorder could handle an input voltage of ±16 mV, with a high resolution of 16 bits and a bandwidth of 0-1 kHz. It contains 549 records from 290 subjects, with ages ranging from 17 to 87. Signal Measurement: Each record has 15 signals,

including the standard 12 ECG leads plus 3 Frank lead ECGs, all digitized at 1000 samples per second.

The ptbdb_normal.csv and ptbdb_abnormal.csv files are part of the PTB Diagnostic ECG Database, which includes ECG recordings from both healthy subjects and patients with various heart conditions. Here's a detailed explanation of these files:

ptbdb_normal.csv: This file contains ECG recordings from healthy subjects. The data is structured similarly to the MITBIH files, with each row representing a single heartbeat and columns representing the signal values at different time points. The last column typically contains the label, which in this case would be 0 indicating a normal heartbeat.

ptbdb_abnormal.csv: This file includes ECG recordings from patients with various heart conditions. The structure is the same as the ptbdb_normal.csv file, with the last column containing the label 1 to indicate an abnormal heartbeat. Given the two distinct types of datasets, we will describe each one separately to prevent any confusion.

Relevance:

Both datasets comprise 188 features, where features 0-186 constitute the data points for a single heartbeat, and the last feature represents a label. After data exploration, we have decided to consider all classes from 1 to 3 as abnormal heartbeats. However, the 'Unclassifiable' beat adds complexity without providing additional value for classification. Consequently, we have opted to exclude this class from the modelling process.

This dataset is a classification problem that uses supervised learning algorithms on the above-mentioned variables. The major aim of this classification is to differentiate between healthy and unhealthy heartbeats in order to develop models.

The major attributes of the dataset are as follows:

- ECG Data Points (Variables 0-186): These represent the amplitudes of ECG signals for one heartbeat. It is primarily characterized by the analysis and classification of heartbeats based on these parameters. Since all explanatory variables are sequential and represent specific segments of the heartbeat sequence for each case, they are uniformly important in predicting the target variable. As a result, no individual feature can be identified as particularly significant.
- Labels (Variable 187): This indicates how the heartbeats are classified thereby being necessary for supervised learning tasks (0: Normal, 1:Abnormal).

Data Limitations:

However, there are some limitations with this clean and well-structured dataset such as: for instance, this dataset lacks frequency components that can yield more information about ECG signals. Also lacking in patient metadata include age, medical history, etc.; all these details could be helpful in

making personalized assessments. Furthermore, there is a data imbalance between normal and abnormal heartbeats, which results in bias when implementing classification models.

1 - Pre-processing and feature engineering of MIT datasets:

Given that the initial data was already clean and all features were on a consistent scale, we carried out minimal feature engineering, specifically, converting the target variable type from float to numeric and adding labels, as the data was initially unlabelled. This allowed us to swiftly advance to the preprocessing stage. As the initial step in data visualization, we plotted the raw ECG signals for each distinct class—Normal, Unclassifiable, Ventricular, Supraventricular, and Fusion—for both the training dataset (Fig. 1) and the test dataset (Fig. 2).. This visualization aids in understanding the variations and patterns in heartbeat data, which are crucial for developing accurate machine learning models for heartbeat classification.

To further refine and finalize the previous step, when running histograms over a section of data points in the mitbih training dataset by each class, we realized that the shapes of heart beat classes do indeed differ from each other, as shown in the example of normal and supraventricular heart beats (Fig3a-b).

The distribution of different heartbeat categories in the training data and the test data is depicted in Figure 4 and Figure 5. This figure highlights the prevalence of each category, which is crucial for understanding the dataset's composition. This analysis reveals that the majority of the data consists of normal heartbeats, while the other categories are significantly less frequent. Understanding this distribution is essential for developing and training machine learning models, as it impacts the model's ability to accurately classify and detect abnormalities in heartbeats.

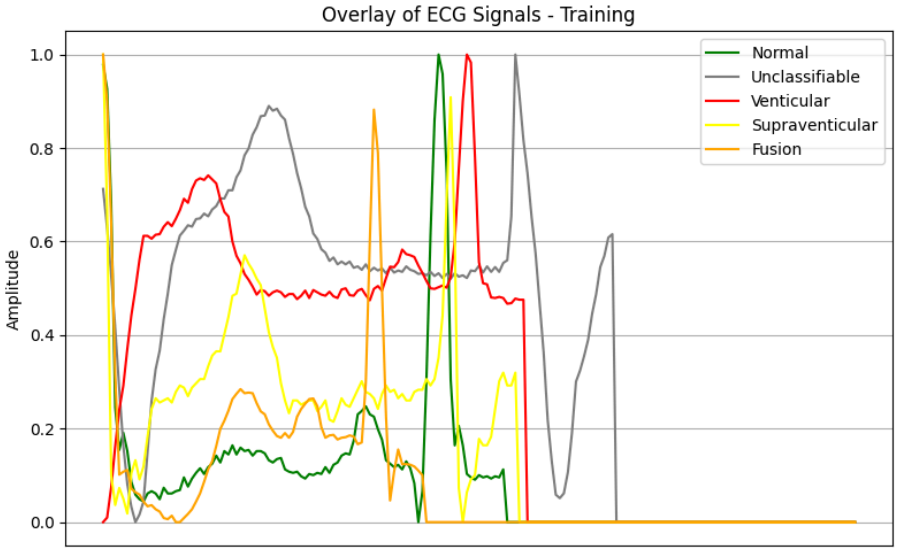

Fig1 - This graph illustrates the variations in ECG signals across different heartbeat categories during the training phase. The categories include normal (green), unclassifiable (grey), ventricular (red), supraventricular (yellow), and fusion (orange). Each line represents the unique electrical activity associated with each type of heartbeat, providing a visual comparison crucial for analysing and understanding heart rhythms.

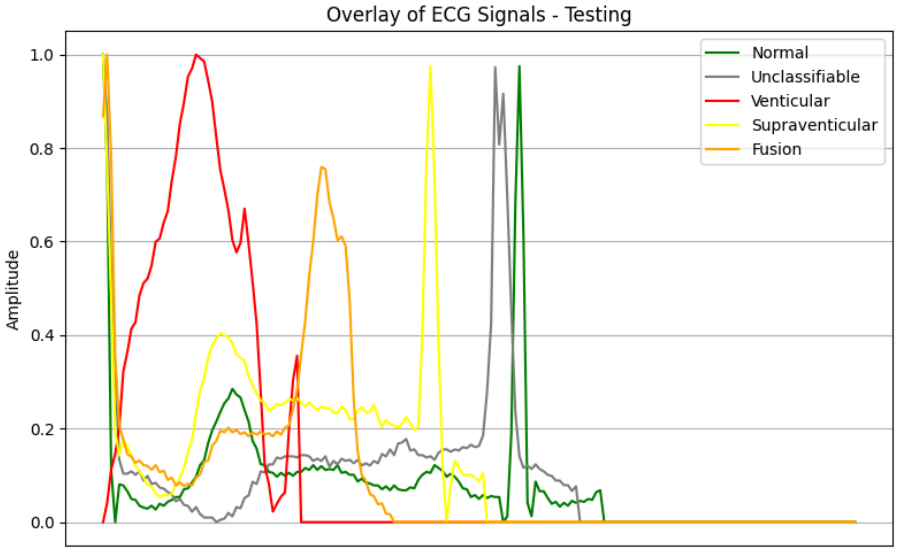

Fig2 - This graph illustrates the variations in ECG signals across different heartbeat categories during the test phase. The categories include normal (green), unclassifiable (grey), ventricular (red), supraventricular (yellow), and fusion (orange). Each line represents the unique electrical activity associated with each type of heartbeat, providing a visual comparison crucial for analysing and understanding heart rhythms.

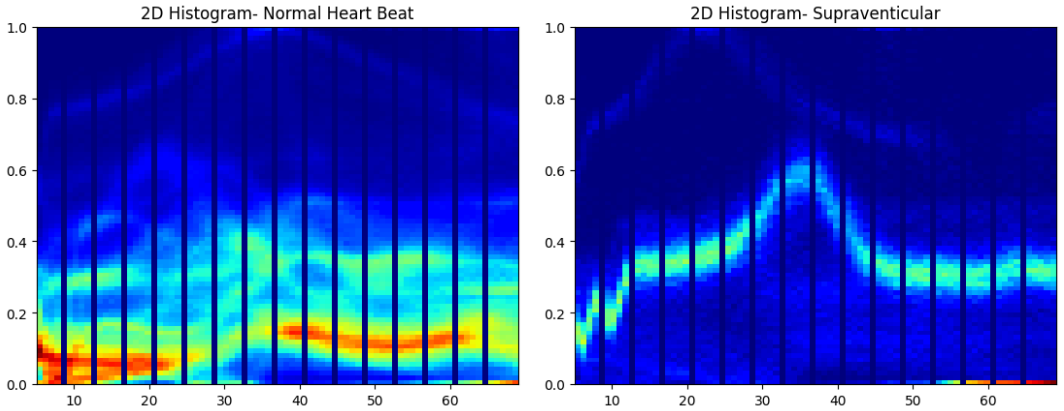

Fig3 - Two-dimensional histograms over a section of data points [5-70] in the training dataset for (left) Normal class and (right) Supraventricular class.

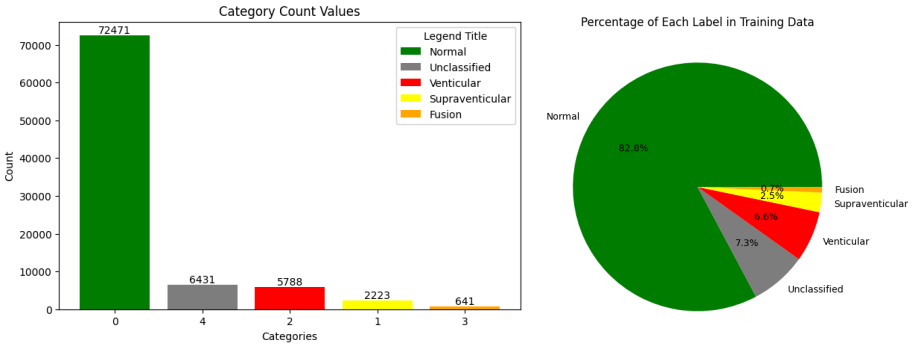

Fig4 - These bar and pie graphs display the count of values for different heartbeat categories in the training data. The categories include Normal (green) with 72,471 counts, Unclassified (grey) with 6,431 counts, Ventricular (red) with 5,788 counts, Supraventricular (yellow) with 2,223 counts, and Fusion (orange) with 641 counts.

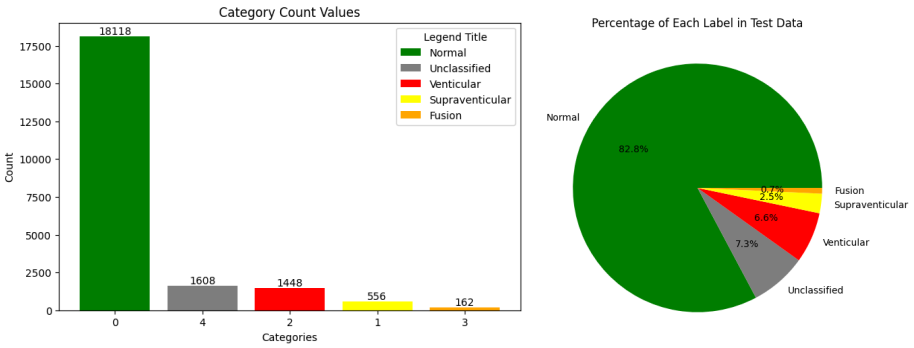

Fig5 - These bar and pie graphs display the count of values for different heartbeat categories in the training data. The categories include Normal (green) with 18118 counts, Unclassified (grey) with 1608 counts, Ventricular (red) with 1448 counts, Supraventricular (yellow) with 556 counts, and Fusion (orange) with 162 counts.

PCA (Principal Component Analysis) is a powerful technique in the field of statistical analysis, often used to make sense out of complex data and present it more simply. This is done, for example using PCA (Principal Component Analysis), and in the field of interpreting heartbeat rate material helps visualization to distinguish between normal behaviour and that of failing abnormalities.

PCA processes the original high-dimensional data to a lower-dimensional space, enabling us to project our data into principal components revealing the highest significant structures. This simplification makes it easier to find clusters and outliers that represent different types of heartbeats (e.g., normal, supraventricular, ventricular beats) including fusions. In our case, we used PCA to a dataset including heartbeat rates, and the dimensions were reduced up to two principal components. This reduction enabled us to create a scatter plot that visually represents the distribution of different heartbeat types.

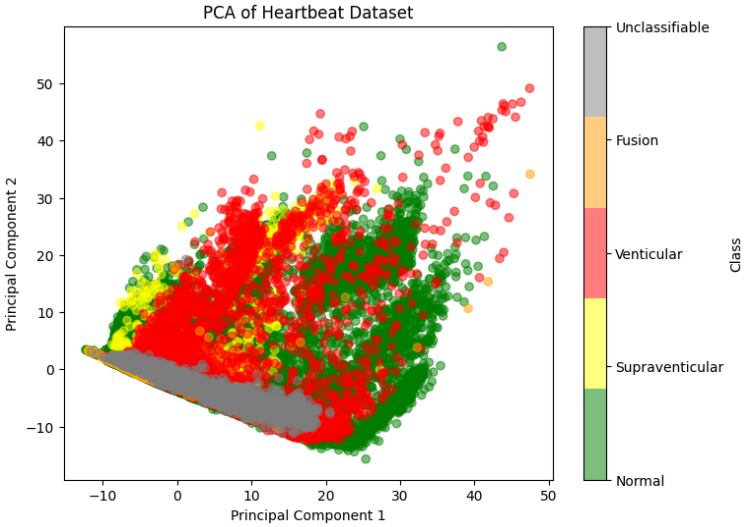

Fig6 - Scatter plot illustrating the distribution of heartbeat data projected onto the first two principal components after PCA dimension reduction. Each point represents an individual heartbeat, color-coded by class: Normal (green), Supraventricular (yellow), Ventricular (red) Fusion (orange), and Unclassifiable (grey). This visualization aids in distinguishing between normal and abnormal heartbeats.

The plot in Figure 6 shows five distinct clusters for each heartbeat class. A higher density indicates more consistency within each class. For example, the Supraventricular heartbeats (yellow) form a dense cluster, suggesting consistent patterns in this heart activity. Additionally, we can see a significant overlap between the 'Normal' (green) and all other abnormal heartbeats, indicating that these classes share similar characteristics, which might make them harder to distinguish. This exploratory data analysis revealed that the classes of the target variable were not distinctly separated. Specifically, 'Fusion' (depicted in orange) represents a mixed category of 'Supraventricular' (yellow) and 'Ventricular' (red), while 'Unclassifiable' (gray) may potentially encompass elements from all other classes. Therefore, we should find a robust model to accurately detect normal and abnormal heartbeats. It is worth noting that points that are far from any cluster, such as some 'Ventricular' (red) heartbeats, can be considered outliers. These outliers might represent rare or unusual heartbeat patterns that could be clinically significant. As we have no information about the data frequency, we will keep all of them in the first step.

Reducing our data to two dimensions helps in visualizing the structure more clearly. To determine the optimal number of dimensions for representing our data, a scree plot is employed. A scree plot is a useful tool in Principal Component Analysis (PCA) for determining the number of principal components to retain. The plot displays the variance explained by each principal component, helping to identify the point where adding more components contributes little additional variance. In the scree plot (Fig 7), the x-axis represents the principal components, while the y-axis shows the percentage of variance explained by each component. The plot typically exhibits a steep decline initially, followed by a leveling off. The "elbow" point, where the slope changes most dramatically, indicates the optimal number of components to retain. From the scree plot, we observe a noticeable elbow after the second principal component. This suggests that the first two components capture the majority of the variance in the data, making them sufficient for further analysis. Retaining more components beyond this point would contribute minimally to explaining additional variance.

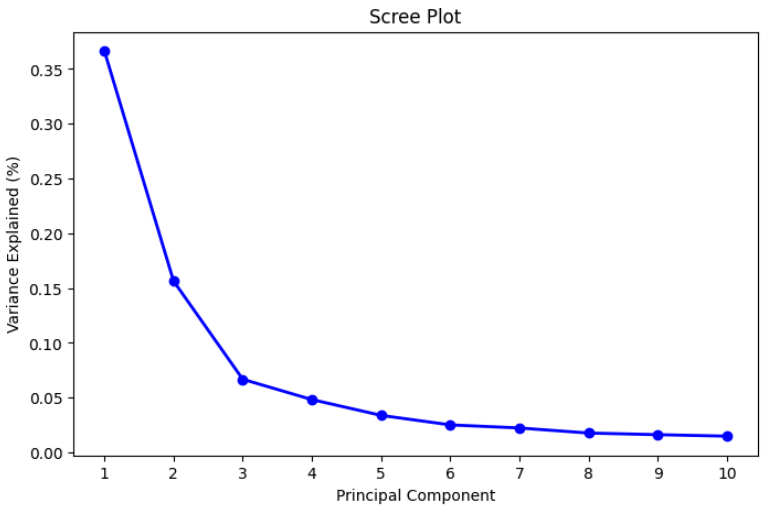

Fig6 - Scree plot illustrating the variance explained by each principal component. The elbow point after the second component suggests that retaining the first two principal components is optimal for capturing the significant variance in the dataset.

Although the elbow in the scree plot suggests three dimensions, we chose to present our data in two dimensions. This decision enhances the clarity of our data representation in the report.

In addition, we have used a t-SNE technique to visualize the heartbeat classes. This map helps us see patterns in normal and abnormal heartbeats by grouping similar ones together. In other words, t-SNE is particularly useful for visualizing high-dimensional data in 2D or 3D, making it easier to see the natural groupings in the data. Each color on the map indicates a particular type of heartbeat, making it easier to determine which are distinct groups and which overlap, making model prediction more challenging.

In our context, clusters represent different classes of heartbeats (normal, ventricular, supraventricular, fusion, and unclassifiable). The distance between clusters indicates how distinct different types of heartbeats are from each other. Well-separated clusters suggest clear differences between the types. As it has been shown in Figure 7, there is separation and clustering between the abnormal classes, but they are not dense and spread, and somehow they overlap in some data points. As we can see, there is also a huge data set from normal heartbeats that leads to overlap. The separation and clustering of these points can help identify specific patterns and anomalies in the heartbeats, but the overlapping between them might lead to a challenging work on this project.

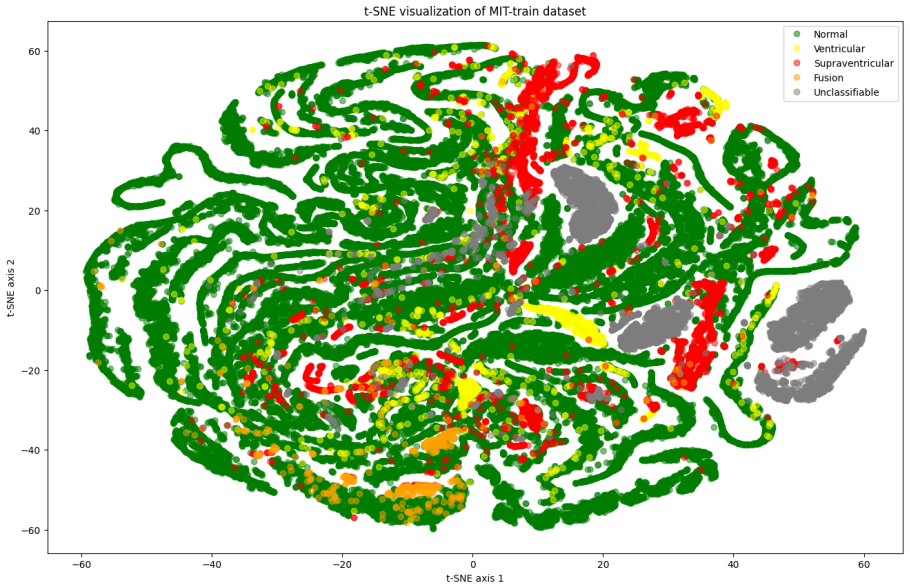

Fig 7 - t-SNE visualization of heartbeat classes from the MITtrain dataset. Green signifies normal heartbeats, red highlights Supraventricular abnormalities, yellow captures ventricular abnormalities, orange marks fusion beats, and grey indicates unclassifiable beats.

By using UMAP, we can gain valuable insights into the patterns of heartbeats, helping to identify and understand normal and abnormal conditions more effectively. As we can see in Figure 8, we have again 5 different classes, but they are not dense and the data points of each class are somehow spread, and clusters also overlap.

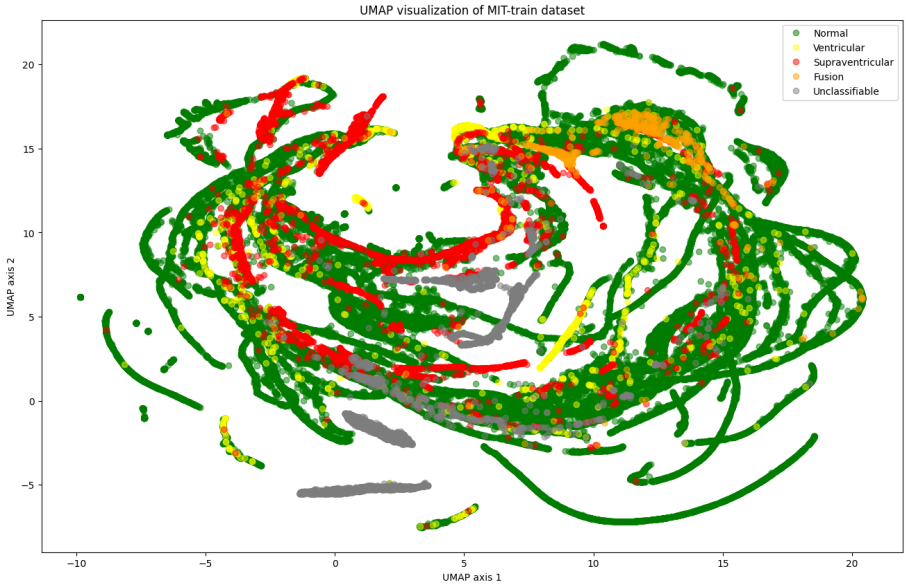

Fig 8 - UMAP visualization showcasing the clustering of heartbeat types within the MIT-train dataset. Each cluster corresponds to a different heartbeat classification, illustrating both the diversity and similarity within heartbeat clusters.

To address the complex and highly unbalanced nature of the class distribution, we opted to proceed with preprocessing and modelling using a dichotomized class variable. In this approach, all classes other than "Normal" are aggregated into a single "Abnormal" class. This simplification reduces the classification task to a binary problem, enabling the model to focus on distinguishing normal cases from all types of abnormalities. This strategy facilitates more effective learning and potentially improves model performance by reducing the complexity of class differentiation. This dichotomized approach not only simplifies the classification task by consolidating all non-normal cases into a single "Abnormal" class but also enables easier comparison with the PTB dataset. By aligning the class structure between datasets, we can more effectively evaluate and benchmark model performance across different sources, enhancing the robustness and interpretability of our results.

Data rescaling (Normalization):

Rescaling, also referred to as normalization or standardization, is a common practice in machine learning preprocessing. It ensures that all features contribute equally to the model, preventing features with larger ranges from dominating the learning process. Hence, scaling can improve model performance, reduce sensitivity, and enhance training stability. In our analysis, by cross-validation, we aimed to evaluate the effectiveness of different rescaling methods to determine the best suitable scaler. The scalers that we considered were: StandardScaler, MinMaxScaler, and RobustScaler.

To address the performance of the mentioned scalers we conducted experiments on the dataset with each scaler and evaluated it by the average F1-score with different machine learning models. We applied multiple models —Logistic Regression (LR), Decision Tree (DT), and K-Nearest Neighbor (KNN)— using the differently scaled test data. We also tried to apply Support Vector Machine (SVM) but could not perform the GridSearch with the additional model, because it was computationally too expensive.

Table 1 - Mean F1-Scores for different combinations of rescaling methods and models

	StandardScaler	MinMaxScaler	RobustScaler	None
LogisticRegression	0.408	0.408	0.409	0.408
Decision Tree	0.781	0.781	0.781	0.781
KNN	0.847	0.855	0.846	0.854

The results indicate all scaling strategies yield comparable, and in some cases identical, mean F1 scores across the models. However, the MinMax scaler demonstrates the highest performance with the KNN model on the mit dataset Therefore, we will employ the MinMaxScaler for MIT data preprocessing in future model training to optimize performance.

Data resampling:

Given the intrinsic imbalance in the dataset, which predominantly favors the normal heartbeat class, we addressed this challenge by employing several resampling techniques. The key methods included Synthetic Minority Over-sampling Technique (SMOTE), Random Oversampling andRandom UndersamplingSMOTE enhances the representation of the minority class by generating synthetic samples, while Random Oversampling duplicates existing minority class samples to balance the class distribution. Conversely, Random Undersampling mitigates imbalance by reducing the number of majority class samples. Given the distribution, in our dataset of heartbeats with normal heartbeats being more common than ones it's crucial to rebalance our data. This way we prevent our model from favoring the group. It's also necessary to modify our loss function to address this discrepancy enabling learning from the class.

We evaluated these resampling methods in conjunction with a set of the previously mentioned base machine learning models: LR, DT andKNN. To identify the optimal combination of model and resampling method, we employed Stratified K-Fold Cross-Validation. This technique ensures that each fold maintains the same class distribution as the original dataset, providing a reliable assessment of model performance. The models were evaluated using the mean F1-score, a metric well-suited for imbalanced datasets.The mean F1-Score calculated through K-Fold Cross-Validation with GridSearch is achieved by the model across all cross-validation folds. This score provides an overall measure of the model’s ability to balance precision and recall across multiple data splits, offering a reliable estimate of its generalizability. By averaging the F1-Scores from each fold, this metric captures the model’s performance consistency and robustness, especially important for selecting the optimal model and resampling method in imbalanced datasets. The mean F1 scores obtained for each sampling method and model are as follows:

Table 2 - Mean F1-Scores for different combinations of resampling methods and models.

	SMOTE	OverSampling	UnderSampling	None
LogisticRegression	0.408	0.408	0.407	0.408
Decision Tree	0.760	0.788	0.626	0.781
KNN	0.824	0.835	0.752	0.835

Among the tested methods, LR consistently yielded relatively low scores across all resampling techniques, with the highest mean F1-score of 0.408 achieved using SMOTE and Random OverSampling. In contrast, OverSampling in particular emerged as the most effective resampling technique, especially when used in conjunction with KNN, which achieved a superior mean F1-score of 0.835. The DT model also performed best with OverSampling, achieving a mean F1-score of 0.788.

Overall OverSampling, especially when paired with KNN, demonstrated the most substantial improvement in model performance, highlighting its effectiveness in addressing class imbalance in the dataset. Hence, we decided to further optimize the model with hyperparameter tuning.

Conclusion:

In our first report, we prepared for the application of machine learning and deep learning techniques to detect and classify heartbeat normality and abnormality using ECG signals. Our study focused on data preparation, including exploration, preprocessing, feature engineering, and model setup for future applications.

We utilized two primary datasets—the MIT-BIH Arrhythmia Database and the PTB Diagnostic ECG Database—which provided a solid foundation for our preparations. Our preprocessing steps ensure data consistency and facilitate effective model training. Exploratory data analysis, including visualizations and dimensionality reduction techniques such as PCA, t-SNE, and UMAP, revealed distinct patterns in ECG signals that enhanced our understanding of heartbeat classifications. These techniques helped us visualize the data structure and relationships between variables.

To benefit from the rescaling, we applied several rescaling methods. Finally, we concluded that MinMaxScaler is more suitable for our case. It can help provide a better F1-score compared to other scaler or no-scaler cases. It will help improve model performance, ensure fair feature contribution, and enhance the stability and speed of the training process.

To address the significant class imbalance, we applied various resampling techniques, with OverSampling demonstrating notable effectiveness. Which, when combined with models like KNN, is expected to achieve the highest mean F1-score. This approach is crucial for improving diagnostic accuracy and enabling early detection of cardiac abnormalities.

2 - Pre-processing and feature engineering of PTB datasets:

Although the MIT dataset and PTB dataset look in a similar structure, their origins are different. Strictly speaking, we can't directly put them together for preprocessing or modelling without proof. That's why we separate into two distinct preprocess steps for MIT and PTB. The pre-processing for the PTB dataset will be under a similar concept to the MIT dataset with little differences as PTB data only contains two classes and the dataset is not that imbalanced.

For the first inspection of the raw data, we plotted the raw ECG signals for reach class–Normal, and abnormal for both the training dataset (Fig.9) and test dataset (Fig. 10) to understand the variation and possible patterns in heartbeat data.

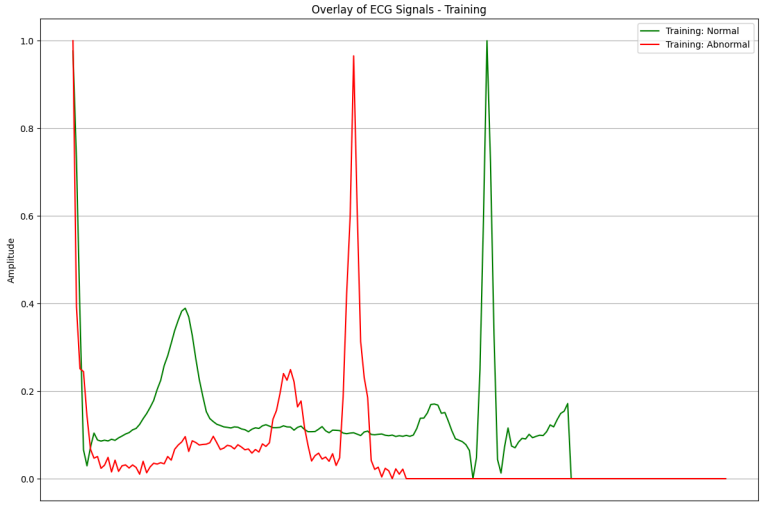

Fig9—The graph shows that in the ECG signal, the normal heartbeat is quite different from the abnormal class.

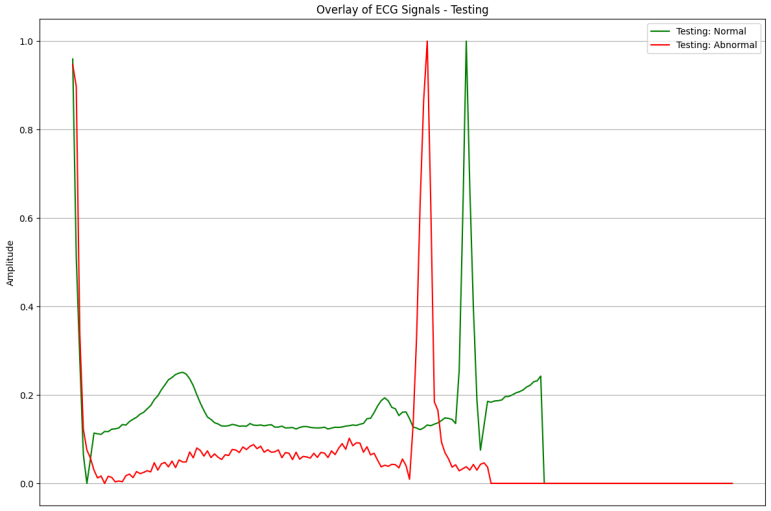

Fig10 - ECG signal on the test set.

To visualize the heartbeat pattern of normal and abnormal, we plot a histogram by class normal and abnormal (Fig. 11). We realize that the shapes of heartbeat classes do indeed differ from each other.

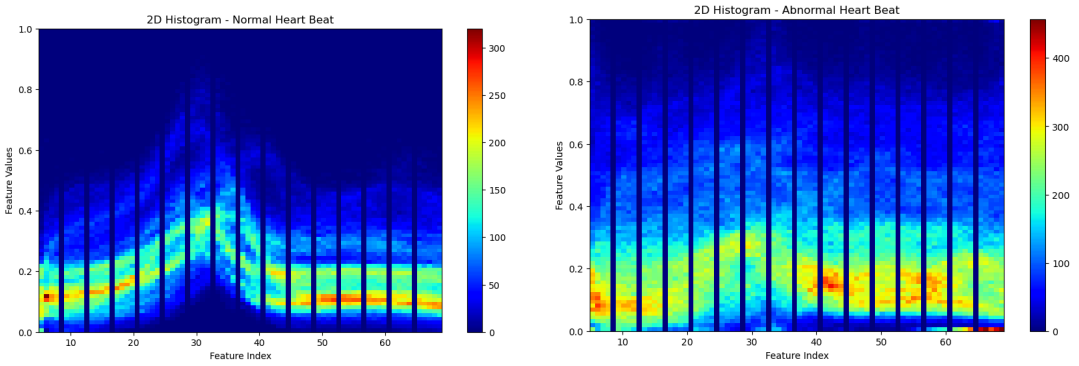

Fig 11 - Two-dimensional histograms over a section of data in the training dataset for (left) Normal class and (right) abnormal class.

The distribution of the normal and abnormal by class is depicted in Figure 12 and Figure 13. This figure highlights the ratio for normal vs abnormal is about 3: 1. We can conclude that this dataset is not collected by nature. In reality, we don't have one-third of people who have heart abnormalities. This is important for us to understand and intercept the result of our model.

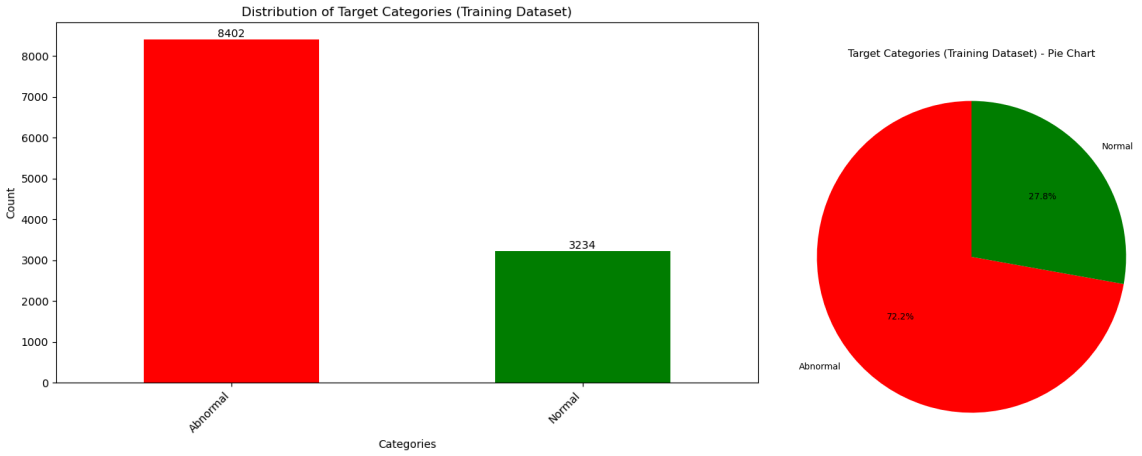

Fig 12 - These charts show the value count for normal and abnormal over the train set.

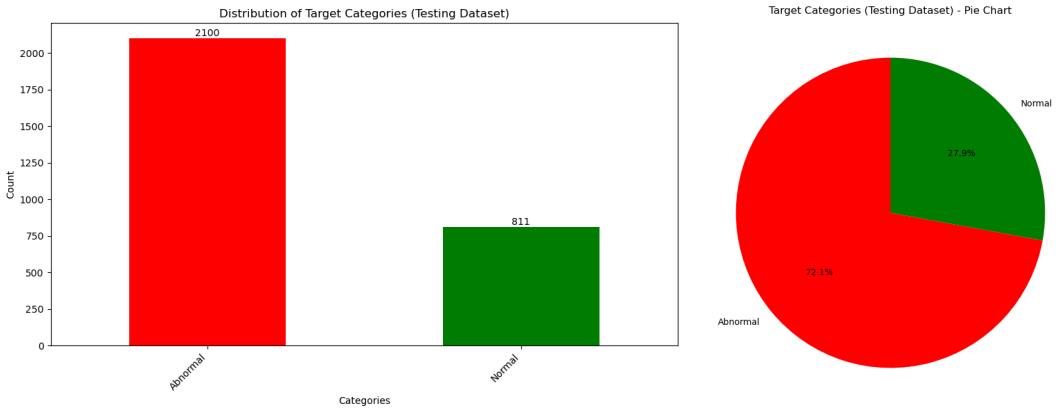

Fig 13 - These charts show the value count for normal and abnormal over the test set.

To further reduce the dataset's complexity and aid in visualization, we performed Principal Component Analysis (PCA). As is well known, PCA retains the majority of variation while converting the initial high-dimensional data into a lower-dimensional space. main Components 1 and 2 are represented by the x- and y-axes, respectively, in Figure 14, which shows the first two main components. "Normal" (green) and "Abnormal" (red) heartbeats can be distinguished from one another by color-coding the data points.

Since most "Normal" heartbeats are centered in the center-left area of the graph, it appears that these data points have comparable features and are therefore simpler to categorize. However, 'Abnormal' heartbeats are more dispersed but also cluster in the same region as 'Normal' ones, suggesting greater variability in their features and thus making them more difficult to categorize. Classification models may face difficulties because of the overlap between "normal" and "abnormal" heartbeats, which indicates that the two classes share some characteristics. The distribution and relationship between "Normal" and "Abnormal" heartbeats are generally better understood thanks to the PCA visualization, which also offers insightful information for more research and model improvement.

Fig 14 - PCA of Heartbeat Dataset (2D) - Training Data. The scatter plot shows the distribution of 'Normal' (green) and 'Abnormal' (red) heartbeats based on the first two principal components. The visualization highlights the clustering of 'Normal' heartbeats and the spread of 'Abnormal' heartbeats, providing insights into their characteristics and potential challenges for classification.

In this analysis, we applied t-Distributed Stochastic Neighbor Embedding (t-SNE) to the PTB train dataset to reduce its dimensionality and facilitate visualization. t-SNE is a machine learning algorithm particularly well-suited for visualizing high-dimensional datasets. It converts similarities between data points into joint probabilities and tries to minimize the Kullback-Leibler divergence between the joint probabilities of the low-dimensional embedding and the high-dimensional data. Figure 15 illustrates the data points in two dimensions.

The data points are color-coded to distinguish between two classes: 'Normal' (green) and 'Abnormal' (red) heartbeats. The visualization shows clusters of points that suggest patterns or groupings within the data. The clustering of 'Normal' heartbeats indicates that these data points share similar characteristics, making them easier to classify. On the other hand, the spread of 'Abnormal' heartbeats suggests more variability in their characteristics, which might make them harder to classify. The overlap between 'Normal' and 'Abnormal' heartbeats indicates some similarities between the two classes, which could pose a challenge for classification models.

Overall, the t-SNE visualization helps in understanding the distribution and relationship between 'Normal' and 'Abnormal' heartbeats, providing valuable insights for further analysis and model development. Such insights can be particularly useful for medical diagnostics and research, where understanding these patterns can lead to better prediction models or insights into heart health.

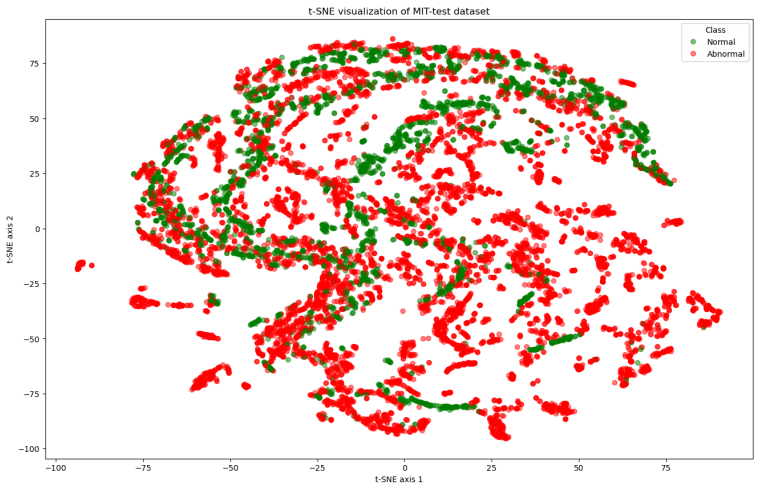

Fig 15 - t-SNE visualization of PTB-train dataset. The scatter plot shows the distribution of 'Normal' (green) and 'Abnormal' (red) heartbeats based on t-SNE Axis 1 and t-SNE Axis 2. The visualization highlights the clustering of 'Normal' heartbeats and the spread of 'Abnormal' heartbeats, providing insights into their characteristics and potential challenges for classification.

Data rescaling(Normalization):

As scaling can improve model performance, we want to find the best rescaling method for the PTB dataset. The scalers that we considered were: StandardScaler, MinMaxScaler, and RobustScaler.

To measure the performance of the mentioned scalers we conducted an experiment on the dataset with each scaler and evaluated it by the average F1-score with different machine learning models. The

models used are Logistic Regression (LR), Decision Tree (DT), Support Vector Machine (SVM), and the k-nearest neighbors (KNN). We employed Stratified K-Fold cross-validation. The models were evaluated using the mean F1-score.

Table 3 - Mean F1-scores for different combinations of rescaling methods and models

	StandardScaler	MinMaxScaler	RobustScaler	None
Logistic Regression	0.837	0.832	0.836	0.83
Decision Tree	0.939	0.939	0.939	0.939
SVM	0.910	0.897	0.905	0.898
KNN	0.947	0.951	0.940	0.949

The results indicate that even when MinMaxScaler is highest with KNN, StandardScaler more often provides better performance than MinMaxScaler and RobustScalerTherefore, we will adopt StandardScaler as a preprocessing step for future model training, given its slight edge in performance.

Data resampling:

Although the PTB dataset is not as imbalanced (3:1) as the MIT dataset, there is still some imbalance present. Therefore, we also test if resampling can help to improve the performance. We test the resampling methods SMOTE, OverSampling and UnderSamping with the same machine learning base models: Logistic Regression, Decision Tree, Support vector Machine, and the K-Nearest Neighbors. To identify the optimal combination of the model and resampling method, we employed Stratified K-Fold Cross-validation. The models were evaluated using the mean F1-score.

Table 4 - Mean F1-scores for different combinations of resampling methods and models

	SMOTE	OverSampling	UnderSampling	None
Logistic Regression	0.838	0.837	0.833	0.837
Decision Tree	0.933	0.941	0.916	0.936
SVM	0.922	0.928	0.891	0.928
KNN	0.925	0.929	0.886	0.945

Among all the tests, most models with the OverSampling strategy or None resampling achieved the best F1-score, despite the dataset's lower imbalance compared to the MIT dataset. To enhance diagnostic accuracy and facilitate early detection of cardiac abnormalities, we applied OverSampling as our primary resampling strategy.

After conducting a grid search using the StandardScaler, OverSampling, and the given base models, it was found that the Decision Tree outperformed the other models under identical conditions, achieving an F1-Score of 0.941. Hence, we decided to further optimize the model with hyperparameter tuning.

Conclusion:

In our first inspection of PTB data, we checked the statistical chart and the shape of the ECG for normal and abnormal classes. Exploratory data analysis, including visualizations and dimensionality reduction techniques such as PCA, and t-SNE, revealed distinct patterns in ECG signals.

To benefit from rescaling, we applied several rescaling methods. Finally, we concluded that StandardScaler is more suitable for PTB data. As for the resampling, even though it is not a clear imbalance dataset, we also applied several resampling methods to find if it's a good idea to perform resampling.

Rendering 2 : Modelling Report

After identifying the optimal resampling, rescaling, and base models for both datasets, we proceeded with hyperparameter tuning of the KNN model for the MIT dataset with dichotomized target variable and the Decision Tree for the PTB dataset. Additionally, we applied grid search on ensemble models to compare their performance with the baseline models. Finally, we implemented a Dense Neural Network, LSTM, and Transformer as deep learning techniques to determine the optimal model based on the average F1-Scores and the confusion matrix of misclassified cases across all models. The following sections present our findings.

MIT_baseline_Models:

In the initial phase of the project, we set out to classify heartbeats using the MIT-BIH Arrhythmia Dataset by exploring baseline models. We used preprocessed data and applied oversampling methods to address class imbalance. Additionally, we utilized the MinMax scaler as a rescaling method to normalize the data. These steps were crucial in improving the model's performance and ensuring that the classification results were accurate and reliable. The models selected for this task were Logistic Regression, Decision Tree, and K-Nearest Neighbors (KNN), chosen for their simplicity and effectiveness in handling classification task**s.**

Logistic Regression: This model served as a solid starting point, delivering moderate accuracy. However, it struggled with the complexity inherent in ECG signals, as its linear nature limited its ability to capture the more intricate patterns within the data. The Logistic Regression model achieved an accuracy of approximately 80.68%. It performed well in identifying normal heartbeats with a precision of 0.97 and a recall of 0.81. However, it struggled with abnormal heartbeats, showing a low precision of 0.33 but a higher recall of 0.77. This indicates that while the model is good at identifying normal heartbeats, it has a high rate of false positives for abnormal heartbeats. The results highlight the need for further optimization to improve the model's ability to accurately detect abnormal heartbeats.

Decision Tree: The Decision Tree model offered better interpretability, making its decision-making process more transparent. Despite this advantage, the high overall accuracy combined with the disparity in class performance (very high for normal, lower for abnormal) and the inherent complexity of the model suggest it might be overfitting. The Decision Tree model achieved an impressive accuracy of approximately 94.13%. It demonstrated excellent performance in identifying normal heartbeats, with a precision of 0.98 and a recall of 0.95, resulting in an F1-score of 0.97. For abnormal heartbeats, the model showed a precision of 0.69 and a recall of 0.83, with an F1-score of 0.75. These results indicate that the Tuned Decision Tree model is highly effective, particularly in detecting normal heartbeats, while still performing reasonably well with abnormal heartbeats. The balanced macro and weighted averages further emphasize its robustness across both classes.

K-Nearest Neighbors (KNN): This model demonstrated strong performance, effectively leveraging local patterns in the ECG signals. It achieved higher accuracy and a better balance between precision and recall when compared to the other baseline models. The K-Nearest Neighbors (KNN) model achieved excellent performance, with an overall accuracy of 97%. For normal heartbeats (Class 0), it showed high precision (0.99) and recall (0.98), resulting in an F1-score of 0.98. For abnormal heartbeats (Class 1), the model had a precision of 0.84 and recall of 0.93, with an F1-score of 0.88. These results indicate the KNN model is highly effective in classifying both normal and abnormal heartbeats, maintaining strong performance across both classes.

Based on the results, the K-Nearest Neighbors (KNN) model is the best choice among the baseline models for this task. It achieved an impressive overall accuracy of 98%, with balanced performance in detecting both normal and abnormal heartbeats. The KNN model showed high precision and recall, making it highly effective and reliable for this application.

To further enhance the performance of the KNN model, we employed the Grid Search method. Grid Search systematically explores different hyperparameter combinations to identify the optimal settings for the model. This tuning process ensures that the KNN model operates at its best, providing even better classification results.

Following grid search optimization on data preprocessed with MinMax scaling and oversampling, we observed a significant improvement in performance, reaching an average F1-Score of 0.94. This result was achieved by selecting the optimal configuration of five neighbors with distance-based weighting and employing the auto algorithm for nearest neighbor computation.

The model demonstrates robust classification accuracy, achieving a precision of 0.99 and recall of 0.98 for normal cases, and a precision of 0.88 and recall of 0.92 for abnormal cases, resulting in F1-Scores of 0.99 and 0.90, respectively. The overall accuracy is 0.98, supported by a macro-average F1-Score of 0.94 and a weighted F1-Score of 0.98.

While the model performs exceptionally well, the difference in F1-Scores between classes highlights some room for improvement, as 455 cases were misclassified across both classes in the MIT dataset. This indicates potential areas for further optimization and refinement. This strong performance makes the KNN model a highly suitable choice for heartbeat classification tasks, providing reliable and accurate results.

MIT Advanced models: Bagging and Boosting Techniques

In the advanced phase of our modelling process, we incorporated bagging and boosting techniques to enhance the performance and robustness of our models. These ensemble methods are particularly effective in improving the accuracy and reducing the variance of our predictions, thereby providing a more reliable classification of heartbeats. Then in addition to base models, using grid search with K-Fold Cross Validation, we optimized the Balanced Random Forest (BRF), Gradient Boosting (GB), Gradient Extreme Boosting (XGB), and LightGBM models. Given that these models are all tree-based, no scaling was applied. Similarly, we avoided resampling methods, as ensemble models possess intrinsic weighting mechanisms to mitigate the effects of dataset imbalance.

Bagging (Bootstrap Aggregating): Bagging involves training multiple versions of a model on different subsets of the data, and then averaging their predictions. In our project, we utilized the Balanced Random Forest algorithm, a popular bagging method. Random Forest creates multiple decision trees using bootstrapped samples of the dataset and averages their predictions. This approach helps mitigate the risk of overfitting. The Balanced Random Forest model achieved an accuracy of 0.91035, with precision and recall both at 0.91035, and an F1-Score of 0.91429. The detailed classification report indicates strong performance in detecting normal heartbeats (Class 0) with a precision of 0.9348 and recall of 0.9476, while showing room for improvement in detecting abnormal heartbeats (Class 1) with a precision of 0.5042 and recall of 0.4462.

Boosting: Boosting sequentially builds models that correct the errors of their predecessors. Each new model focuses on the mistakes made by previous models, resulting in a strong final model. We employed Gradient Boosting and XGBoost algorithms in our project.

Gradient Boosting works by adding weak learners, typically decision trees, to the model iteratively. The Gradient Boosting model achieved an accuracy of 0.94794, with a precision of 0.94610 and recall of 0.94794, resulting in an F1-Score of 0.93801. It excelled in detecting normal heartbeats (Class 0) with a precision of 0.9048 and recall of 0.9965, but had lower performance for abnormal heartbeats (Class 1) with a precision of 0.8053 and recall of 0.1220.

XGBoost, an advanced implementation of Gradient Boosting, optimizes the boosting process by introducing regularization techniques to prevent overfitting and enhance performance. The XGBoost model demonstrated exceptional performance with an accuracy of 0.96147, precision of 0.96067, recall of 0.96147, and an F1-Score of 0.95755. It showed high precision (0.9326) and recall (0.9950) for normal heartbeats (Class 0), and reasonable performance for abnormal heartbeats (Class 1) with a precision of 0.9041 and recall of 0.3977.

Additionally, we implemented LightGBM, a boosting algorithm known for its efficiency and accuracy. LightGBM achieved an accuracy of 0.91647, with precision at 0.92968, recall at 0.91647, and an F1-Score of 0.92088. It performed well for normal heartbeats (Class 0) with a precision of 0.9422 and recall of 0.9303, and showed moderate performance for abnormal heartbeats (Class 1) with a precision of 0.4718 and recall of 0.5214.

Among the different ensemble techniques applied, XGBoost made the most significant improvement to our modelling. Contrary to our expectations, the ensemble models exhibited markedly low

performance with the MIT dataset compared to KNN. On the MIT dataset, the ensemble models demonstrated relatively poor performance overall. Although XGBoost remained the best-performing model with an average F1-Score of 0.779, the ensemble models, in general, underperformed compared to the base models with the MIT dataset. However, if the highest overall accuracy and precision are paramount, and recall can be managed through additional means (e.g., further data preprocessing or model tuning), XGBoost is a strong contender despite its lower recall. Nevertheless, we aimed to further optimize the XGBoost model through hyperparameter tuning to evaluate potential performance improvements. Yet, following grid search optimization on un-preprocessed data with a binary target variable, we observed a modest improvement in performance.

After tuning the XGBoost model using a grid search with specified parameters, the performance metrics showed significant improvements. The tuned XGBoost model achieved an impressive overall accuracy of 0.97. For normal heartbeats (Class 0), the precision was 0.97, recall was 1.00, and F1-Score was 0.98. For abnormal heartbeats (Class 1), the precision improved to 0.96, with an F1-Score of 0.81, although the recall was 0.70. In comparison, the untuned XGBoost model had an accuracy of 0.96147, precision of 0.96067, recall of 0.96147, and an F1-Score of 0.95755. Specifically, the untuned model's precision and recall for normal heartbeats were 0.9326 and 0.9950 respectively, while for abnormal heartbeats, the precision was 0.9041 and the recall was lower at 0.3977. The tuning process enhanced the precision for abnormal heartbeats but still highlighted the need for further improvement in recall, balancing the critical aspects of both precision and recall in medical applications

This result was achieved by selecting the optimal configuration of XGBoost hyperparameters, tailored to balance complexity and performance in the model:

The optimal configuration for the XGBoost model includes several key hyperparameters designed to balance model complexity and performance. The parameter colsample_bytree is set to 0.8, meaning that for each tree built, only 80% of the features are randomly selected. By using a subset of features per tree, the model reduces the risk of overfitting while introducing diversity among trees, which can improve its generalization to new data.

The learning rate was set to 0.1, which controls the step size during model optimization. This moderate learning rate provides a balance between rapid learning and stable convergence. Lower learning rates allow the model to make smaller, more gradual adjustments, often leading to better generalization and performance.

The max depth parameter was set to 3, limiting each tree to a maximum depth of three levels. By restricting the depth, the model's complexity is controlled, which helps prevent overfitting. This setting reduces the tendency of trees to memorize specific patterns in the training data, making shallower trees more resilient in datasets where noise might mislead deeper models.

With 100 estimators, the model includes a sufficient number of trees to capture patterns in the data without becoming overly complex. While adding more estimators can enhance performance, it also increases computation time; thus, 100 trees represent a balanced choice that provides adequate learning capacity.

Finally, the subsample rate is set to 0.8, indicating that each tree is trained on 80% of the available training data, sampled randomly for each iteration. This sampling introduces randomness and reduces the likelihood of overfitting by ensuring that each tree sees slightly different data, thereby enhancing the model's robustness. Together, these hyperparameters create a model that is well-tuned for effective learning and generalization.

We also tried to apply the XGBoost weighting parameter as a follow-up step, but realized that the model performed not significantly different with the one without the weighting parameter. Despite this optimization, the XGBoost model without class weighting achieved a maximum overall F1-Score of 0.814, which misclassified 708 cases and is still falling short of the base KNN model.

MIT Advanced models: Deep Learning techniques

In the final stage of our modelling process, we employed a Deep Neural Network (DNN) to further enhance the performance of our heartbeat classification model. A DNN utilizes multiple fully connected layers to model complex patterns and relationships within the data. For this task, we constructed a sequential DNN with several hidden layers, each followed by batch normalization and dropout to prevent overfitting and ensure robust performance. By leveraging the depth and hierarchical structure of the DNN, we aimed to improve the detection of both normal and abnormal heartbeats, leading to a reliable and efficient classification system. This deep learning approach represents a significant advancement in our effort to develop an effective heartbeat classification model, capable of handling the variability inherent in medical data.

We implemented deep learning methods due to the sequential nature of the data. Specifically, we evaluated a Dense Neural Network (DNN), a Convolutional Neural Network(CNN) a Long Short-Term Memory (LSTM) network, and a Transformer model to assess their effectiveness in capturing temporal dependencies within the dataset.

The selected deep learning models are particularly suited for sequential data due to their distinct capabilities. The DNN serves as a flexible baseline model, capturing general patterns within fixed-length representations of the data. The CNN excel at identifying local patterns in data and automatic extract features with high dimension data. The LSTM is specifically designed for sequence processing, using gated mechanisms to retain information over time, making it ideal for detecting short- and medium-term dependencies. The Transformer model, with its attention mechanism, excels in learning complex, long-range relationships without the constraints of sequential memory, often outperforming on tasks requiring nuanced pattern recognition across extended sequences. Together, these models provide a robust framework for sequential data analysis.

However, the results of the Transformer models are inconclusive, as they have demonstrated suspiciously poor performance (~ 0,50 average F1-Score). This suggests that the DNN model, with its simpler structure, may be better suited to the specific characteristics of the dataset. After implementing CNN and LSTM with MIT, we also got a convincing result compared to DNN. We will present those two models later on. Whereas the DNN outperformed base and ensemble models on the MIT dataset. The deep learning model architecture contains 187 neurons in the input layer, aligning with the dataset's 187 features. A ReLU activation function was used to introduce non-linearity, enabling the

model to learn intricate feature relationships. The hidden layers include three fully connected (Dense) layers with 64, 32, and 8 neurons, respectively, each using ReLU activation as well. Batch Normalization and Dropout accompany these layers to enhance performance and prevent overfitting. Batch Normalization normalizes activations within each mini-batch, promoting faster and more stable training. Dropout, set to randomly disable 30% of neurons during training, discourages overreliance on specific neurons and fosters more generalizable representations. The output layer, designed for binary classification, contains a single neuron with a sigmoid activation function, producing class probabilities. This setup enables the model to estimate the likelihood for each class accurately. The model uses the Adamax optimizer, a stable variant of Adam, with a learning rate of 0.002. Adamax is particularly effective for models with sparse gradients, providing robust learning across datasets. The chosen learning rate promotes gradual, stable learning, minimizing the risk of bypassing optimal solutions. To optimize classification, the binary cross entropy loss function is applied, as it measures the divergence between predicted and actual labels, essential for binary tasks. Accuracy is monitored throughout training to track model performance. To further enhance training, two callbacks are employed: EarlyStopping monitors validation loss, halting training if no improvement is detected after 10 epochs, while restoring the best weights to mitigate overfitting. ReduceLROnPlateau reduces the learning rate by half when validation loss plateaus for 5 epochs, with a minimum threshold of 0.00001.

Table 5 : Performance Metrics for DNN Techniques on MIT Data

	precision	recall	F1_score	support
0 (Normal)	0.99	0.99	0.99	18118
1(abnormal)	0.94	0.90	0.92	2166
accuracy			0.98	20284
Macro avg	0.96	0.95	0.95	20284
weighted avg	0.98	0.98	0.98	20284

The table shows that the model performs very well in classifying cases. For normal cases, the model achieves a precision and recall of 0.99, giving it an F1-Score of 0.99. For abnormal cases, it reaches a precision of 0.92 and recall of 0.91, resulting in an F1-Score of 0.92. Overall, the model is 98% accurate, with a macro-average F1-Score of 0.95 and a weighted F1-Score of 0.98, showing it performs well across both types of cases. The F1-Score of 0.918 reflects balanced effectiveness in classifying both case types, with particularly high accuracy for normal cases. The model misclassified only 353 instances in the test dataset, representing the lowest error rate among models tested on the MIT dataset.

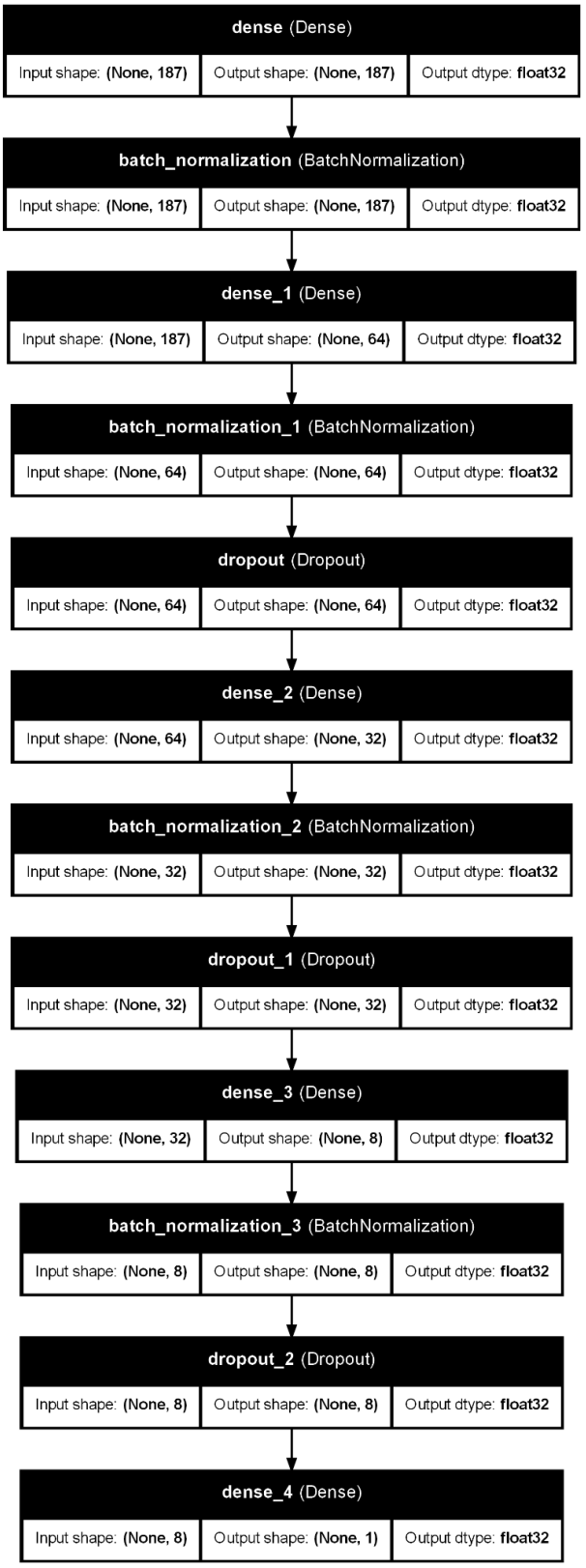

Fig 16 - *layout of dense neural network* created for MIT dataset.

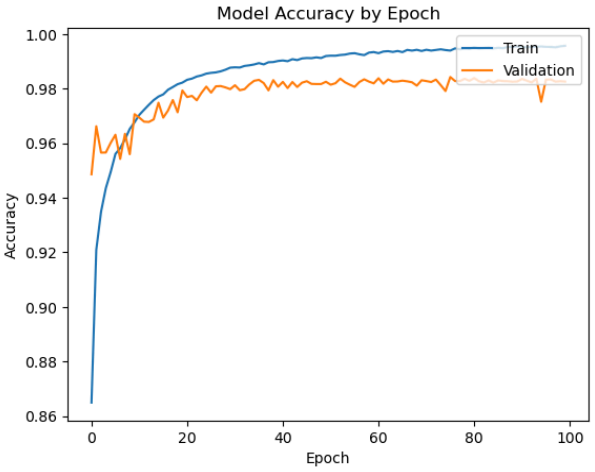

Fig 17 : *Model Accuracy by Epoch.* The graph shows the training and validation accuracy of the DNN model over 100 epochs. The blue line represents the training accuracy, which steadily increases and plateaus around 99%. The orange line represents the validation accuracy, which fluctuates around 97%. This indicates that the model performs well on unseen data but shows slight overfitting, as evidenced by the training accuracy being higher than the validation accuracy. The figure is essential for evaluating the performance and generalization capability of the DNN model.

The training accuracy (blue line, fig 17) shows a steady increase, reaching near 100% by the final epochs, indicating that the model is highly fitted to the training data. The validation accuracy (orange line) follows a similar upward trend initially, stabilizing around 97-98% after approximately 20 epochs. This plateau in validation accuracy suggests that the model achieves high performance on the validation set, but further training does not yield significant improvements. The visible gap between training and validation accuracy indicates mild overfitting, as the model generalizes slightly less effectively on the validation data compared to the training set.

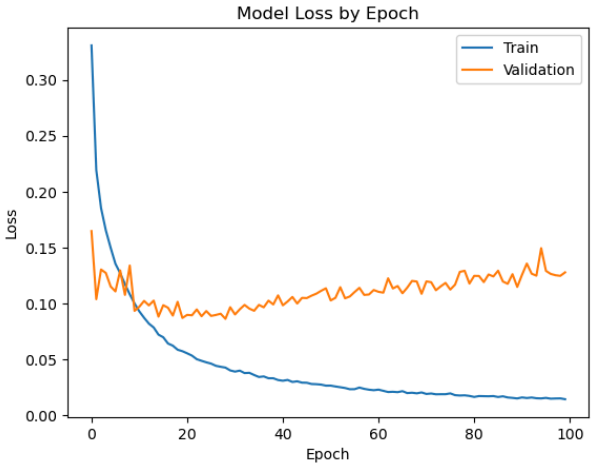

Fig 18 : *Model Loss by Epoch:*This figure illustrates the training and validation loss of the DNN model over 100 epochs. The x-axis represents the number of epochs, ranging from 0 to 100, while the y-axis shows the loss values. The blue line indicates the training loss, which consistently decreases, demonstrating the model's learning process. The orange line represents the validation loss, which initially decreases but starts to increase after a certain point, suggesting potential overfitting as the model's performance on unseen data begins to degrade.

The training loss (blue line) decreases continuously, approaching zero by the end of training, which is consistent with effective learning on the training data. In contrast, the validation loss (orange line) decreases initially but begins to fluctuate and slightly increase after approximately 20 epochs, showing signs of stabilization. This behavior is again a characteristic of overfitting, where the model becomes increasingly specialized to the training data at the expense of generalization. The divergence between training and validation performance suggests that the model needs further regularization strategies or early stopping to mitigate overfitting.

MIT Advanced models: CNN

Similar to DNN, CNN also outperformed based and ensemble models on the MIT dataset. The convolutional neural network model architecture contains 187 neurons in the input layer which is (Conv1D layer), with ReLU activation function was used to introduce non-linearity, enabling the model to learn intricate feature relationships. After the first Conv1D layer, we also have a second Conv1D layer with 64 neurons to reduce the dimension. It uses ReLU activation function as well. Batch normalization and Dropout accompany these layers to enhance performance and prevent overfitting. We choose different dropout rate for the two Conv1D layer. First layer, we use 0.3, second layer we use 0.4, discourages overreliance on specific neurons and fosters more generalizable representations. The output layer, designed for binary classification, contains a signal neuron with a sigmoid activation function, producing class probabilities.

For the model, we use adam as an optimizer, which can adapt the learning rate for each parameter dynamically. We use binary_crossentropy as the loss function, as we use binary classification. Accuracy matrix will be used to evaluate the performance of the model.

We introduced two callbacks to help us have better control with the training process. One of the callback is EarlyStopping, it can stop training if we don't get any improvement after 10 epochs. ReduceLROnPlateau reduce the learning rate by waiting 3 epochs to reduce the learning rate, the new learning rate will be reduced to one tenth.

Table 6 : Performance Metrics for CNN Techniques on MIT Data

	precision	recall	F1_score	support
0 (Normal)	1	0.97	0.98	18118
1 (abnormal)	0.8	0.96	0.87	2166
accuracy			0.97	20284
Macro avg	0.9	0.97	0.93	20284
weighted avg	0.97	0.97	0.97	20284

Fig 19 - *layout for convolutional neural network*

The table shows that the model performs well in classifying cases. For normal cases, the CNN model achieves a precision of 1 and recall of 0.97, giving the F1_score of 0.98. For abnormal cases, it reaches a precision of 0.8 and recall of 0.96, resulting f1_score of 0.87. Overall, the CNN model is 97% accuracy, with a macro-average F1-score of 0.93 and weighted f1-score of 0.98, showing it

performs well across both types of cases. Also, the f1_score of all classes looks good. However, as we are more care about the abnormal case, so compare to DNN, the CNN model doesn't perform well on the abnormal class.

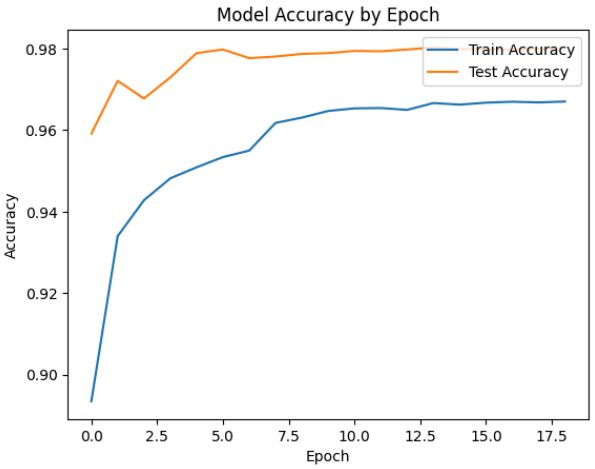

Fig 20 - *Model Accuracy by Epoch* for CNN

We noticed that the accuracy of CNN model converge very quicker than DNN. After about 18 epochs it arrives in the condition of early stop stage. But the problem is the accuracy of validation always looks better than train set. There might be an underfit issue. As we applied the scaler to a dataset before training, This might suppress the model's ability to perfectly memorize the training data which reduce the training accuracy. We may need to try to increase the training epochs to check if the problem continues to persist.

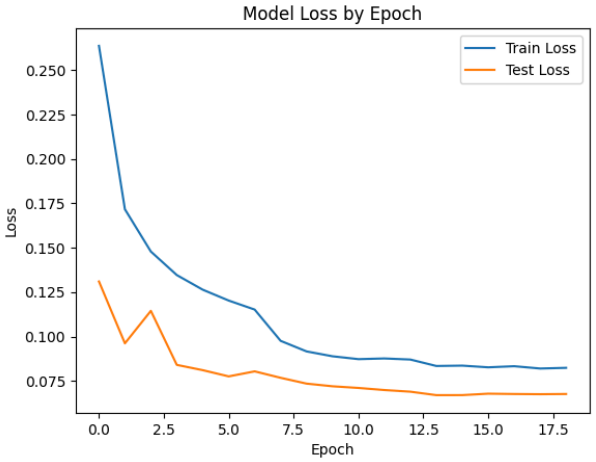

Fig 21 : *Model Loss by Epoch* for CNN

The training loss decreases continuously, approaching 0.1. Even it is not close to zero, but after several epochs it doesn't decrease, we can conclude that we arrive the optimized loss result no more epoch

needed. The loss of validation set keep almost the same trend with train set, it indicate that the model works well with good generalization due to regularization.

LSTMAs for LSTM model, we also outperformed based and ensemble models on the MIT dataset. The Long short-term memory neural network model architecture contains 187 neurons in the input layer which with tanh as activation function to introduce non-linearity. The hidden layer we applied another LSTM model with 16 neurons same activiation function tanh, following by a dropout layer with dropout rate of 0.3. The output layer, designed for binary classification, contains a singal neuron with a sigmoid activation function, producing class probabilities.

During the training process, we use Adam optimizer to adapt the learning rate for each parameter dynamically. Binary_crossentropy as the loss function as we use binary classification. Accuracy matrix will be used to evalue the performance of the whole model.

We introduced two callbacks to help us have better control with the training process. One of the callback is EarlyStopping, it can stop training if we don't get any improvement after 10 epochs. ReduceLROnPlateau reduce the learning rate by waiting 5 epochs to reduce the learning rate, the new learning rate will be reduced by half.

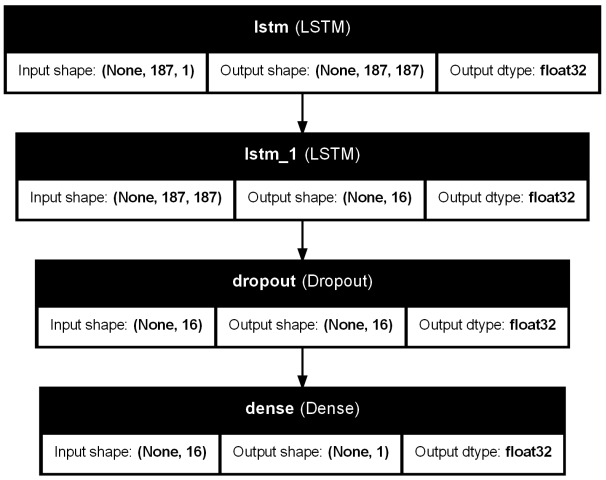

Fig 22 - layout for long short-term memory model

Table 7 : Performance Metrics for LSTM Techniques on MIT Data

	precision	recall	F1_score	support
0 (Normal)	0.95	0.94	0.95	72471
1(abnormal)	0.94	0.95	0.95	72471
accuracy			0.95	144942
Macro avg	0.95	0.95	0.95	144942
weighted avg	0.95	0.95	0.95	144942

The table shows the LSTM model performs well on both normal and abnormal classes. The model achieves almost balanced precision recall and f1_score of about 0.94-0.95. Overall, the model is 95% accuracy for both Marco avg and weighted avg.

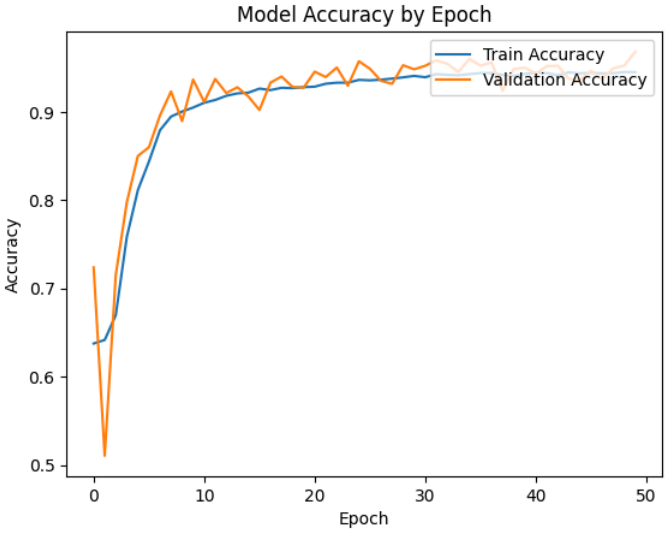

Fig 23 - *Model Accuracy by Epoch* for LSTM

The train accuracy shows a steady increase, reaching near 95% by the final epochs, indicating that the model is highly fitted to the training data. The validation accuracy follows a similar upward trend, stabilizing around 95% after 20 epochs. This plateau in validation accuracy suggests that the model achieves high performance on the validation set, but further training does not make significant improvements.

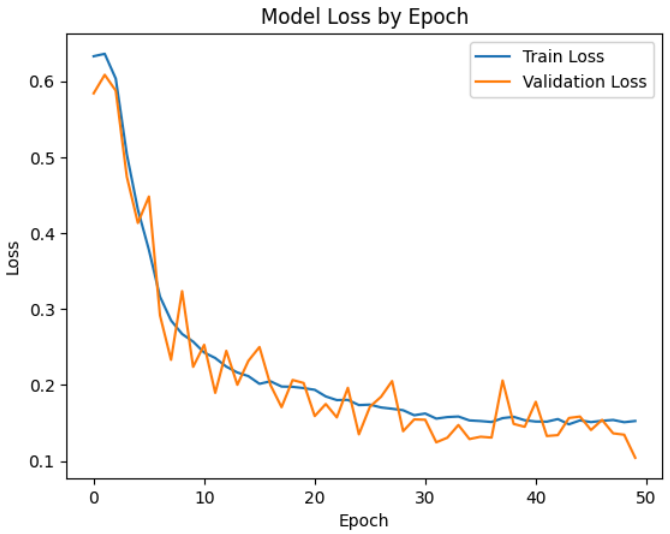

Fig 24 : *Model Loss by Epoch* for LSTM

The training loss decreases continuously, approaching 0.2. Even it is not close to zero, but after several epochs it doesn't decrease. We can conclude that we arrive in the optimized loss result, no more epoch needed. The loss of validation keeps almost the same trend with training set with some noise. The gap with the train set is also closed. Basically, it indicates that the model has converged. It's good for us to stop there.

PTB_Baseline_Models:

As grid search identified the Decision Tree model as the optimal base model and all included ensemble models are tree-based as well, the preprocessing for these models does not include scaling. Additionally, only the Decision Tree model is preprocessed with OverSampling, as ensemble methods inherently incorporate mechanisms to handle unbalanced datasets. The optimal configuration for the Decision Tree model was determined through grid search, resulting in the following hyperparameters (Table 8):

The criterion is set to entropy, meaning the model uses information gain based on entropy to determine the best way to split nodes. This approach prioritizes splits that maximize the separation of classes within each node. The max depth parameter is set to None, allowing the tree to expand fully without depth restrictions, so each branch continues growing until all leaves are pure or contain fewer than the minimum specified samples. The min. samples leaf` parameter is set to 1, meaning that each leaf node can contain as few as one sample, enabling the tree to capture fine-grained distinctions in the data. Finally, the min. samples split parameter is set to 2, meaning that any node with at least two samples can be split, promoting a flexible structure that adapts to the data's patterns. This configuration achieved an mean F1-Score of 0.9508, reflecting the model's effective balance between the two classes.

Table 8 : Performance Metrics for Decision Tree model on PTB Data

	precision	recall	F1_score	support
0 (Normal)	0.95	0.95	0.95	834
1(abnormal)	0.88	0.87	0.88	2077
accuracy			0.93	2911
Macro avg	0.92	0.91	0.91	2911
weighted avg	0.93	0.93	0.93	2911

Furthermore, the model shows strong classification performance, achieving an overall accuracy of 0.93, which means that 93% of all cases in the test dataset were correctly classified. For abnormal cases, the model achieved a precision of 0.88 and a recall of 0.87, resulting in an F1-Score of 0.88. This indicates that the base model is already reasonably effective at identifying abnormal cases, though it occasionally misclassifies them as normal, leading to slightly lower precision and recall compared to the other class. For normal cases, the model achieved both a precision and recall of 0.95, yielding an F1-Score of 0.95. This high performance indicates that the model accurately identifies

normal cases with few false positives or false negatives, demonstrating a strong capacity to correctly classify normal instances.

The macro average F1-Score of 0.91 provides a balanced view of performance across both classes. Meanwhile, the weighted average F1 score of 0.93 reflects the model's strong overall performance, accounting for the larger number of normal cases in the dataset. Overall, the model performs well in distinguishing between normal and abnormal cases, with slightly higher accuracy in classifying normal cases, likely due to their greater representation in the dataset.

The overall F1 score of 0.951 indicates a strong fit of the fine-tuned Decision Tree model to the PTB dataset, highlighting its effectiveness in distinguishing between normal and abnormal cases. This high F1 score is supported by the confusion matrix, which shows only 205 misclassified cases within the PTB dataset.

PTB Advanced models: Bagging and Boosting Techniques

In addition to base models, using grid search, we optimized the Balanced Random Forest (BRF), Gradient Boosting (GB), Gradient Extreme Boosting (XGB), and LightGBM models for the PTB dataset as well, in which no scaling or sampling was applied. All ensemble models performed well on the PTB dataset, with GB achieving the lowest overall F1-Score of 0.952, XGBoost outperformed the other ensemble models, achieving an F1-Score of 0.984 (Table 9).

Table 9 : F1-score Mean for some advanced models on PTB Data

Model	F1-score	Model	F1-score
Balaced Random Forest	0.978	XGBoost	0.984
Gradient Boosting	0.952	LightGBM	0.979

The optimal hyperparameters identified for the XGBoost model through grid search are carefully chosen to enhance performance while balancing complexity and generalization. First, the colsample_bytree parameter is set to 0.8, meaning that 80% of the features are used to construct each tree. By subsampling the features for each tree, this parameter introduces variability among the trees, which helps prevent overfitting and promotes better generalization on new data.

The learning rate is set at 0.2, which allows the model to converge more quickly, providing faster learning. Although this rate is relatively high, it is well-balanced with other hyperparameters, reducing the risk of overshooting optimal solutions and ensuring that the model learns efficiently without compromising accuracy.

For tree depth, the max_depth parameter is set to 5, restricting each tree to a moderate depth. This depth enables the model to capture moderately complex patterns within the data. While deeper trees can represent more intricate relationships, a depth of 5 strikes a good balance, controlling overfitting and supporting generalization.

Finally, the model's total number of trees, or n_estimators, is set to 300. This number of estimators provides sufficient capacity for the model to learn detailed patterns, which can improve accuracy. However, this choice also maintains computational efficiency and avoids excessive complexity, particularly in light of the subsampling and depth constraints. Together, these hyperparameters create a model that is both powerful and efficient, optimizing performance while managing overfitting and ensuring scalability.

Table 10 : Performance Metrics for XGBoost model on PTB Data

	precision	recall	F1_score	support
0 (Normal)	0.98	0.98	0.98	2077
1(abnormal)	0.98	0.96	0.97	834
accuracy			0.98	2911
Macro avg	0.98	0.98	0.98	2911
weighted avg	0.98	0.98	0.98	2911

According to the classification report (Table 10) the model performs exceptionally well in distinguishing between abnormal and normal cases, achieving an overall accuracy of 0.98, which means it correctly classified 98% of all cases in the test dataset. For abnormal cases, the model achieved a precision of 0.98, indicating that 98% of cases predicted as abnormal were actually abnormal. The recall for abnormal cases is 0.96, meaning that 96% of all true abnormal cases were correctly identified by the model. This results in an F1 score of 0.97, showing the model's strong ability to balance precision and recall in detecting abnormal cases, with few misclassifications. For normal cases, the model also attained a high precision of 0.98, meaning that 98% of predicted normal cases were indeed normal. The recall for normal cases is 0.99, indicating that the model accurately identified 99% of all true normal cases. This leads to an F1 score of 0.99, highlighting the model's effectiveness in correctly identifying normal instances with minimal errors. The macro-average F1 score is 0.98, which averages the F1 scores for both classes equally, showing balanced performance across abnormal and normal cases. Additionally, the weighted average F1 score is also 0.98, which takes the support (number of instances) for each class into account. This weighted score closely aligns with the overall accuracy, reflecting consistent model performance across both classes.

The gridsearch configurations maximized the model to a strong predictive performance to an average F1-Score of 0.988, in which only 49 cases have been misclassified.

PTB Advanced models: Deep Learning techniques

Due to the similar structure of both datasets, the DNN maintains the same architecture as was used with the MIT dataset. This consistency allows the model to leverage the same feature representation and processing layers, ensuring a uniform approach across datasets and facilitating direct comparison of performance results (187 neurons in input layer with ReLU activation function, three hidden layers with progressively fewer neurons (64, 32, and 8), each followed by Batch Normalization and Dropout layers, a single neuron with a sigmoid activation function for binary classification tasks and Adamax

optimizer with a learning rate of 0.002). They layout of DNN for PTB is defined the same as MIT dataset. You can check the details in Figure 16.

Table 11 : Performance Metrics for DNN model on PTB Data

	precision	recall	F1_score	support
0 (Normal)	0.99	0.98	0.99	2116
1(abnormal)	0.98	0.99	0.99	2087
accuracy			0.99	4203
Macro avg	0.99	0.99	0.99	4203
weighted avg	0.99	0.99	0.99	4203

The classification report (Table 11) reveals that the model demonstrates near-perfect performance in distinguishing between abnormal and normal cases, achieving an overall accuracy of 0.99. This high accuracy indicates that 99% of cases in the test dataset were correctly classified, reflecting the model's strong generalization capability. For abnormal cases, the model attained a precision of 0.98, meaning that 98% of instances predicted as abnormal were indeed abnormal. The recall for abnormal cases is 0.99, indicating that 99% of actual abnormal cases were correctly identified by the model. This results in an F1 score of 0.99, highlighting the model's balanced effectiveness in detecting abnormal instances with very few misclassifications. In the case of normal cases, the model achieved a precision of 0.99, demonstrating that 99% of instances predicted as normal were truly normal. The recall for normal cases is 0.98, meaning that 98% of true normal cases were accurately classified. This yields an F1 score of 0.99, underscoring the model's high accuracy in identifying normal cases. The macro-average F1-Score is 0.99, indicating that the model performs uniformly well across both classes, regardless of class distribution. The weighted average F1 score, also 0.99, accounts for the support of each class and aligns closely with the overall accuracy, confirming the model's robust performance across the dataset. The overall F1-Score of 0.989 highlights its reliability and accuracy for this binary classification task as only 48 cases are misclassified.

In the loss graph (Fig 25), the training loss (blue line) continuously decreases, approaching near-zero levels by the end of training. This downward trend reflects the model's ability to minimize error on the training data effectively. The validation loss (orange line) follows a similar trajectory, decreasing initially and then stabilizing with slight fluctuations around a low value after approximately 20 epochs. The low, stable validation loss further depicts that the model is not overfitting, as it maintains a good balance between the training and validation data, which suggests that the model is well-calibrated.

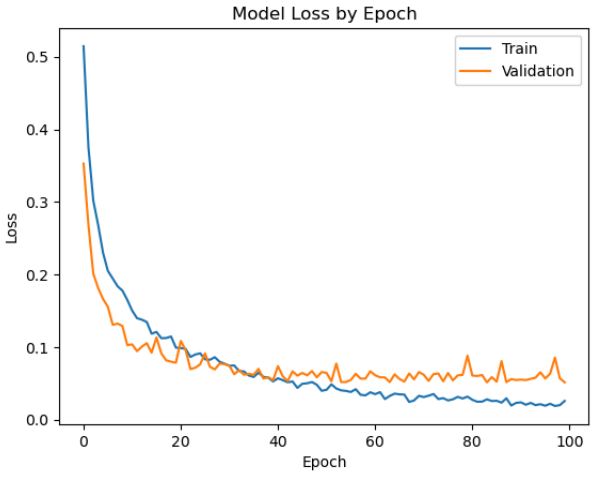

Fig 25 : *Model Loss by Epoch:*This figure illustrates the training and validation loss of the DNN model over 100 epochs. The x-axis represents the number of epochs, ranging from 0 to 100, while the y-axis shows the loss values.

In the accuracy graph (Fig 25), the training accuracy, represented by the blue line, steadily increases, approaching nearly 100% by the final epochs. This upward movement emphasizes that the model is successfully learning from the training data and improving its accuracy as training progresses. The validation accuracy, shown in orange, also rises rapidly at the beginning, stabilizing around 98-99% after the initial epochs. This high and stable validation accuracy suggests that the model generalizes well to unseen data, without significant overfitting. The close alignment between training and validation accuracy curves further supports this observation, as it indicates that the model is performing consistently well on both the training and validation sets.

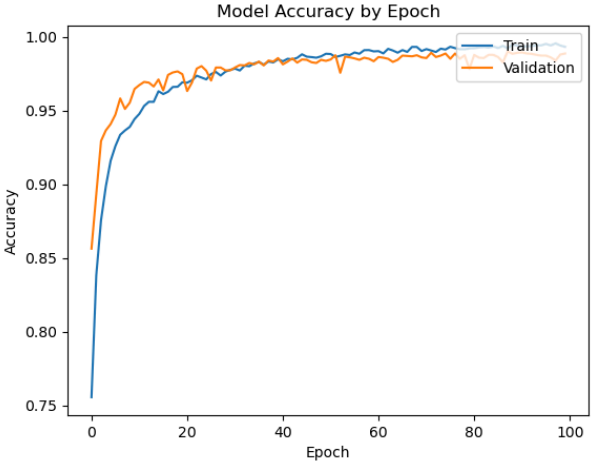

Fig 26 : *Model Accuracy by Epoch.* The graph shows the training and validation accuracy of the DNN model over 100 epochs.

CNN

The CNN model we defined for PTB data is the same with MIT data, as they have similar data structure and output layer. The layout of CNN is the same as MIT dataset, check the Figure 19.

Table 12 : Performance Metrics for CNN model on PTB Data

	precision	recall	F1_score	support
0 (Normal)	0.98	0.99	0.99	803
1(abnormal)	1.00	0.99	0.99	2108
accuracy			0.99	2911
Macro avg	0.99	0.99	0.99	2911
weighted avg	0.99	0.99	0.99	2911

The table shows the model performing very well in classifying cases. For both normal and abnormal cases, the precision, recall, f1_score are all close to 1 around 0.99. The CNN model for PTB data is 99% accuracy, also for Marco avg and weighted avg. Compared to DNN model, CNN perform a bit better on abnormal cases, which is more important for our target.

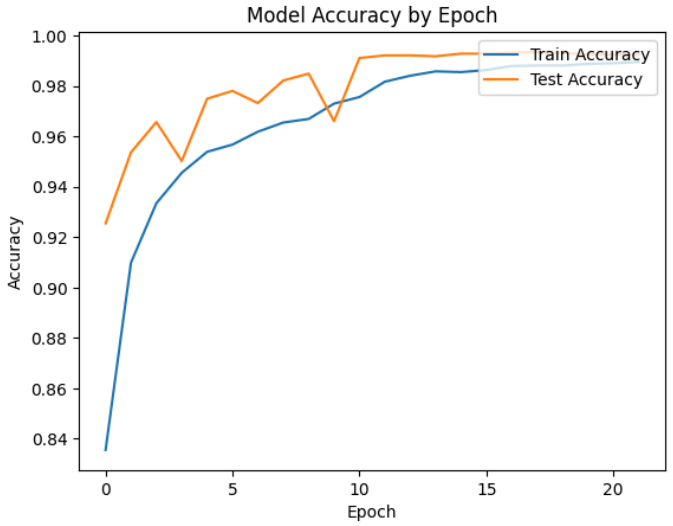

Fig 27 : Model Accuracy by Epoch for CNN of PTB

We noticed that the accuracy of CNN model converged quickly and arrived in the condition of early stop stage after just 20 epochs. This indicates that the CNN model is highly fitted to the training data. The validation accuracy in general follows the same trend with the training set. We can conclude that the model works well both on the train set and validation set.

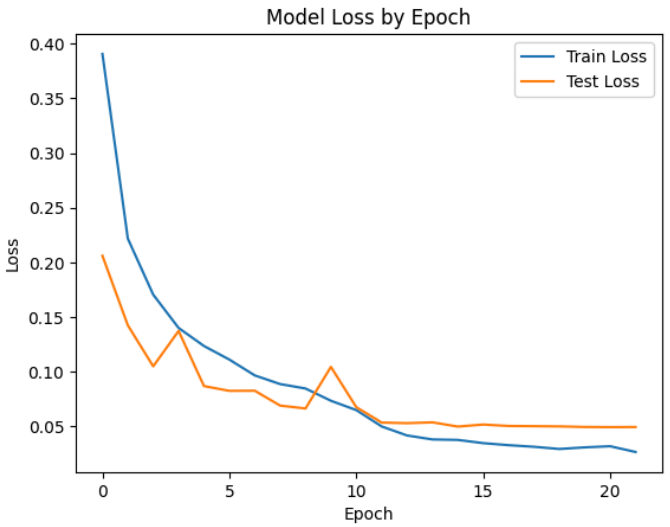

Fig 28 : Model Loss by Epoch for CNN of PTB

The training loss decreases continuously, approaching 0.025. Even though it is not close to zero, after 5 epochs it doesn't continue to decrease. We conclude that we arrive at the optimized loss result no more epoch needed. The loss of validation set keeps almost the same trend with train set but the converged loss is a lot bigger than training set, which indicate it may not work as well as on train set. In general, the model works well.

LSTM

The LSTM model we defined for PTB data is the same with MIT data, as they have similar data structure and output layer. The layout of LSTM for PTB is the same as MIT, for more details please check fig 22.

Table 13 : Performance Metrics for CNN model on PTB Data

	precision	recall	F1_score	support
0 (Normal)	0.93	0.97	0.95	8398
1(abnormal)	0.97	0.93	0.95	8398
accuracy			0.95	16796
Macro avg	0.95	0.95	0.95	16796
weighted avg	0.95	0.95	0.95	16796

Compared to CNN and DNN, the result of the LSTM model for PTB data doesn't work that well. It only achieves for normal class precision 0.93, recall 0.97, f1_score 0.95, for abnormal class precision 0.97, recall 0.93, f1_score 0.95. And the overall accuracy is 95% the same accuracy for Marco average and weighted avg.

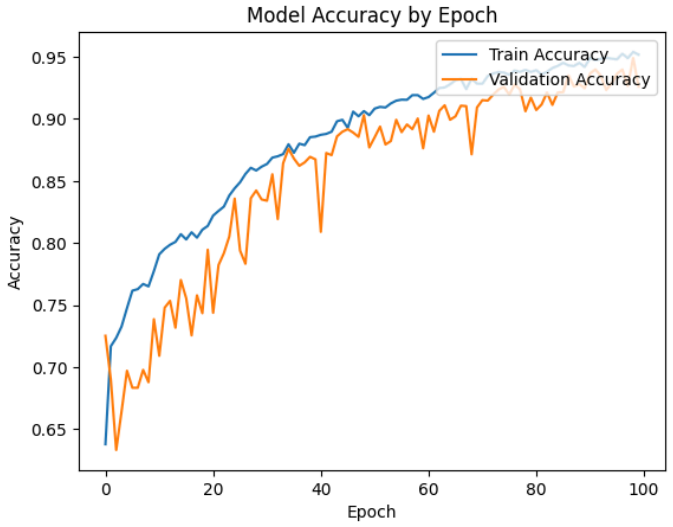

Fig 29 : Model Accuracy by Epoch for LSTM of PTB

The blue line represents the training accuracy, which steadily increases and plateaus around 95%. This upward movement emphasizes that the model is successfully learning from the training data and improving its accuracy as a training process. Validation accuracy trend similar with train trend but with noise, stabilizing around 0.95. This validation accuracy suggests that the model generalize well to unseen data without significant overfitting. The close alignment between training and validation accuracy curves further supports this observation, as it indicates that the model is performing consistently well on both the training and validation sets.

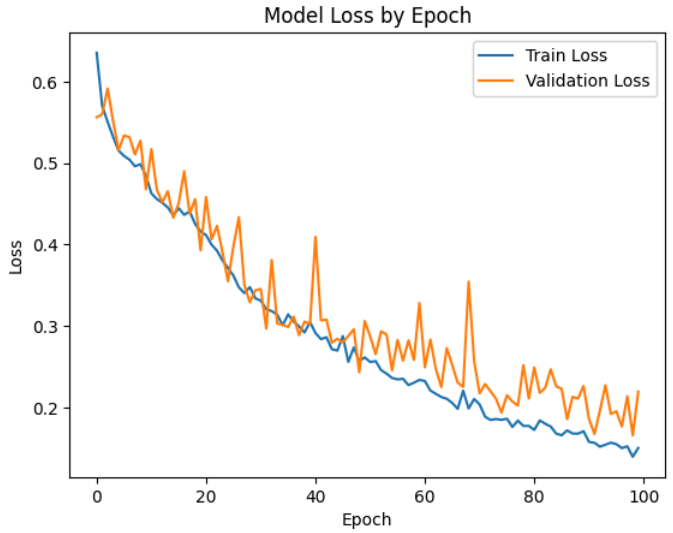

Fig 30 : Model Loss by Epoch for LSTM of PTB

In the loss graph, the training loss continuously decrease. From the graph, we didn't see any significant converge for the loss function. This indicates, we may need to add more epochs so that we can get a curve that converge somewhere for training set. As for the validation set, the loss graph

decrease as the same trend as train set, with some noises. The loss graph further depicts that the model can still be improved with more epoches.

Explanation of the DNN for MIT and PTB:

In the following step we carried out a global and local interpretability analyses for the base models (KNN and DT), the XGBoost models, and the DNN models using Feature Importance (where applicable), SHapley Additive exPlanations (SHAP) and Local Interpretable Model-Agnostic Explanations (LIME).

Feature Importance ranks features based on their contribution to model predictions, primarily useful for tree-based models like Decision Trees and XGBoost. It shows which features are most influential in improving prediction accuracy, aiding in model interpretation and feature selection. However, it is less effective for complex models, like DNN.

SHAP uses cooperative game theory to assign each feature a SHAP value, representing its contribution to a prediction. It provides both global and local insights, explaining overall model behavior and individual predictions by quantifying how much each feature influences the model's output. SHAP is model-agnostic, making it suitable for all types of models.

LIME explains individual predictions by approximating the model's behavior around a specific instance with an interpretable linear model. It identifies which features are most impactful locally, offering a clear explanation for a specific prediction. LIME is also model-agnostic, working with any type of model to provide localized interpretability.

Given that the DNN achieved the highest performance on both datasets, we will focus on presenting its interpretability results in the report by SHAP and LIME. However, explanations for the other models are also available in our GitHub repository for reference.

DNN Explanation for MIT:

In the DNN the SHAP values provide a clear ranking of feature importance, emphasizing that a few key features dominate the prediction process within the DNN model. The feature c_0 demonstrates the highest importance with a mean absolute SHAP value of 0.0488, suggesting that it plays a significant role in the model's decision-making process (Fig 31). Following c_0, features c_3 and c_2 are also highly influential, with mean absolute SHAP values of 0.0372 and 0.0276, respectively. These values indicate that the DNN model places considerable weight on these features when making predictions. Other features in the top 10, such as c_13, c_39, c_26, and c_6, show lower, yet still substantial, SHAP values around 0.019, indicating their moderate but meaningful impact on the model's decisions. Finally, features c_54, c_30, and c_31 round out the top 10, with SHAP values close to 0.016.

In other words, the SHAP analysis indicates that specific parts of the heartbeat sequence—particularly the early segments (features 0, 2, and 3), as well as points around the 13th, 30th, and 54th positions—are highly relevant for classifying the heartbeat. These features, all positioned toward the beginning of the heartbeat frequency, play a significant role in the model's classification decisions. This suggests that early patterns within the heartbeat signal contain critical information that the model

uses to distinguish between different heartbeat types, underscoring the importance of early-cycle features in the classification process.

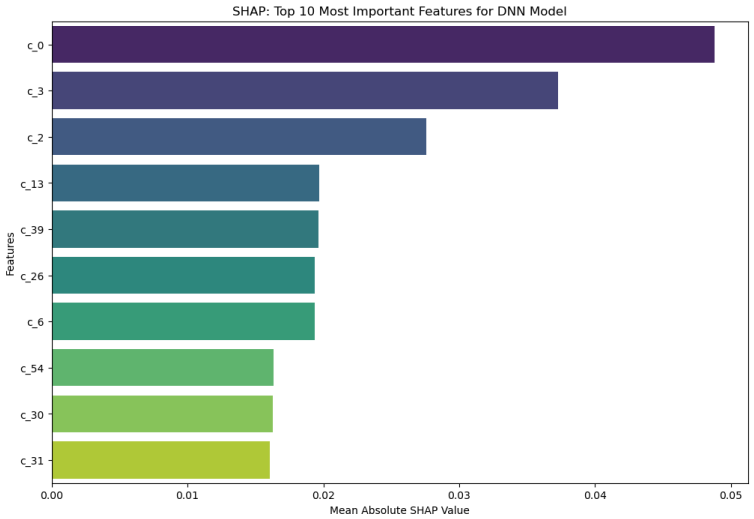

Fig 31: *SHAP values* show that feature c_0 is the most important in the DNN model with a mean absolute value of 0.0488, followed by features c_3 (0.0372) and c_2 (0.0276). Early-cycle features, especially around positions 13, 30, and 54, play a critical role in heartbeat classification.

In the LIME explanation for a randomly chosen Instance 200 (True Label: 0 = Normal), the top contributing features are c_0, c_39, c_26, c_155, c_183, and others (Fig 32). c_0, c_39, and c_26 are among the top-ranked features in both LIME and SHAP, indicating a level of agreement between the methods. This alignment suggests that these features consistently play an important role in predicting the class for this instance and potentially across other cases as well.

However, LIME also highlights additional features like c_155 and c_183 for this specific prediction, which may not have as high importance globally (as indicated by SHAP). This difference reflects LIME's focus on local explanations, emphasizing features most relevant to this particular instance rather than the model's behavior across the entire dataset.

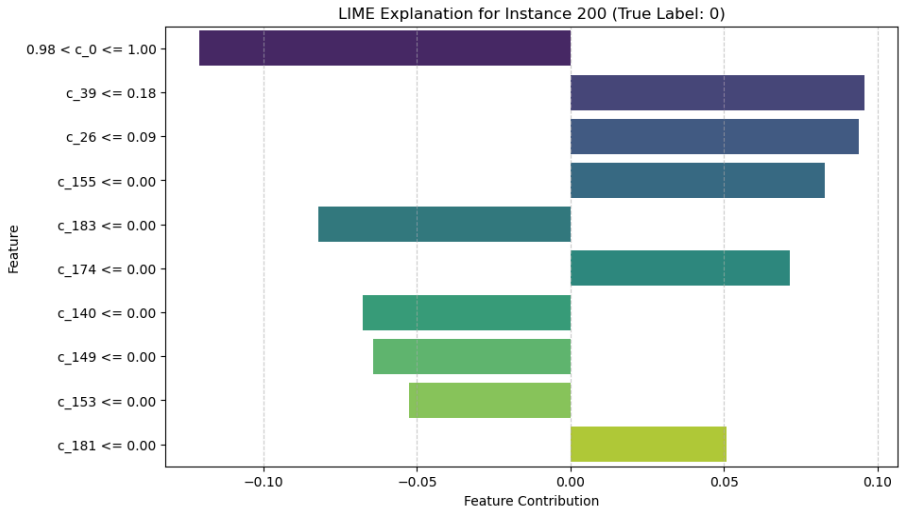

Fig 32: LIME reveals features c_0, c_39, and c_26 as top predictors for Instance 200, with additional features c_155 and c_183 specific to this case, showing local nuances.

DNN Explanation for PTB:

For the PTB dataset, the feature c_2 has the highest mean SHAP value, suggesting that it is the most critical feature in driving the model's predictions (Fig 33). Following closely are features c_7, c_5, and c_32, each contributing significantly to the classification decisions.

The remaining features in the top 10, such as c_1, c_6, and c_34, show slightly lower mean SHAP values but still have substantial contributions to the model's predictions. This distribution of feature importance indicates that the DNN model relies on a combination of specific features to make accurate predictions, with a focus on early and distinct signals within the dataset.

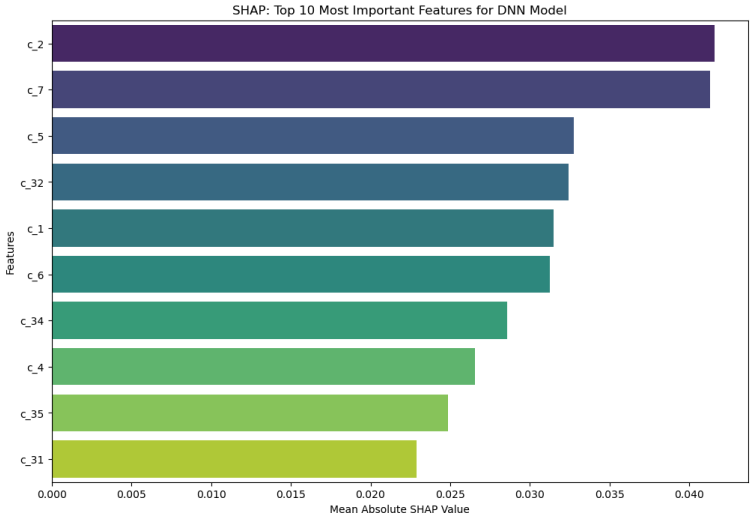

Fig 33: Feature c_2 dominates with the highest mean SHAP value in the PTB dataset, making it crucial for model predictions. Significant contributions come from c_7, c_5, and c_32, while c_1, c_6, and c_34 still play important roles.

The LIME plot shown here provides a local explanation for randomly chosen Instance 200 from the PTB dataset for an as Normal labeled case. In this instance, LIME indicates that the features c_181, c_180, c_182, and c_175, hence very late features, play a dominant role in prediction. Feature c_181 has the largest positive contribution, meaning it pushes the prediction towards the normal class (Class 1). This indicates that the model interprets the value of c_181 as a strong signal that this instance is normal. Meanwhile, features such as c_180, c_182, and c_175 have negative contributions, meaning they push the prediction toward the abnormal class (Class 0). These features signal characteristics that the model typically associated with abnormal cases.

Despite some features (like c_180 and c_182) pulling toward abnormal, the strong positive influence of c_181 and other positively contributing features results in the model's final prediction as normal (Class 1), consistent with the true label of this instance.

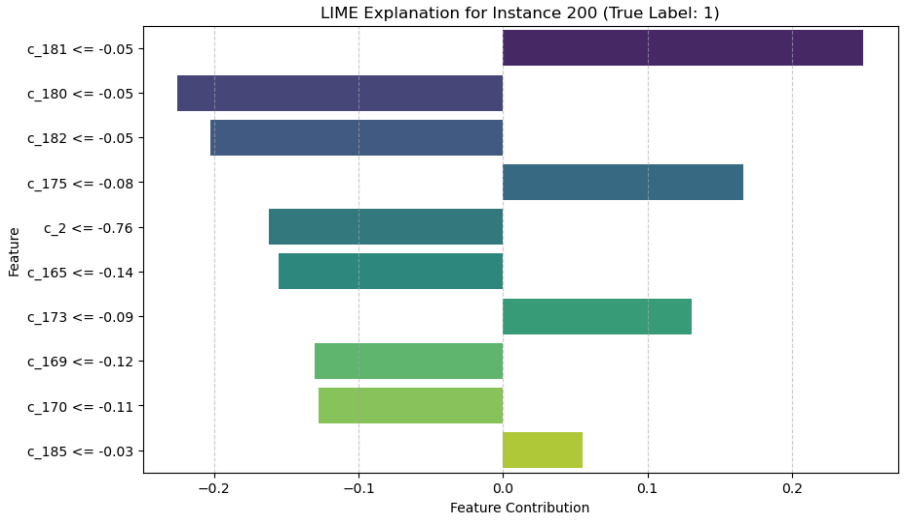

Fig 34: The LIME plot illustrates that for Instance 200, features c_181, c_180, c_182, and c_175 significantly impact the model's prediction. Feature c_181 has the strongest positive influence, pushing towards the normal class, while c_180, c_182, and c_175 have negative contributions, indicating abnormal traits. Despite these, the dominant positive influence of c_181 ensures a final normal prediction.

Even though the exact features that are critical for prediction vary, the pattern in the PTB dataset suggests even stronger than in the MIT dataset that early segments of the sequence play a crucial role in model predictions, because they capture essential information or signal patterns relevant to classifying heartbeat data.

However, the differences in the pattern reflects differences in the signal characteristics or distribution across the datasets, with each dataset emphasizing slightly different parts of the heartbeat signal. For this reason, we have decided to merge both datasets and conduct a combined analysis in the final step, which will be presented in the final report. By integrating the MIT and PTB datasets, we aim to leverage the strengths of each dataset and capture a more comprehensive representation of the data. This approach will allow us to examine shared patterns and differences across datasets, potentially enhancing the model's generalizability and robustness. The combined analysis will also provide a

unified framework for evaluating feature importance and interpretability, offering more reliable insights into the model's behavior across diverse heartbeat signals.

Conclusion:

In conclusion, our modeling efforts demonstrated strong classification performance across both the MIT and PTB datasets, with the DNN model achieving the highest accuracy. Through systematic hyperparameter tuning, we optimized the KNN, Decision Tree, and XGBoost models, with XGBoost achieving solid performance on the PTB dataset, but less so on MIT. Our interpretability analysis, using SHAP and LIME, highlighted critical features influencing predictions, particularly early-cycle features in the heartbeat sequence that proved significant in the DNN's decision-making. These insights suggest that key segments of the heartbeat carry essential signals for classification, which the models leverage effectively. The local interpretability provided by LIME offered additional context for individual predictions, aligning well with global feature importance in many cases. While the DNN's architecture proved effective for both datasets, further tuning is needed for LSTM and Transformer models due to their unexpectedly low performance. Given differences in feature importance across datasets, merging the MIT and PTB datasets for a combined analysis may enhance model generalizability and robustness. This integration will support a more comprehensive interpretability framework, enabling a deeper understanding of feature contributions across diverse heartbeat signals.

Final thoughts:

During our analysis of heart rate signals, we realized that global explanations derived through SHAP offered limited utility due to substantial variation in individual heart rates. Consequently, we decided to reanalyze the data using the given models after aligning the heart rate signals based on their R-peaks and standardizing their lengths through padding. Shifting the signals ensures better alignment and comparability of the data. Heart rate signals can vary due to different physiological conditions or external disturbances. Aligning the signals minimizes such variations and ensures that differences between classes are more likely attributed to the target variable rather than random shifts in the signal.

Bibliography:

1. Mark RG, Schluter PS, Moody GB, Devlin, PH, Chernoff, D. An annotated ECG database for evaluating arrhythmia detectors. *IEEE Transactions on Biomedical Engineering* 29(8):600 (1982).
2. Moody GB, Mark RG. The MIT-BIH Arrhythmia Database on CD-ROM and software for use with it. *Computers in Cardiology* 17:185-188 (1990).
3. Moody GB, Mark RG. The impact of the MIT-BIH Arrhythmia Database. *IEEE Eng in Med and Biol* 20(3):45-50 (May-June 2001). (PMID: 11446209)
4. Goldberger, A., Amaral, L., Glass, L., Hausdorff, J., Ivanov, P. C., Mark, R., ... & Stanley, H. E. (2000). PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. *Circulation [Online]*. 101 (23), pp. e215–e220.