# Heartbeat Classification: Final Report

**Aleksei Liksman, Simon Dommer and Hakan Dogan**

---

## Table of Contents

# Introduction

- ***Context of the project's integration into your business.***

  The objective of this project is to leverage heartbeat signals from the MIT-BIH and ECG Database to develop a machine learning model aimed at automating the classification of normal cardiac rhythms versus various arrhythmias and myocardial infarctions. As part of both software and hardware for ECG machines, to enhance the capabilities of these products ensure their continued improvement in the field.

- ***From a technical point of view.***

  From a technical perspective, the manual analysis of ECG signals poses challenges due to the intricacies involved in detecting and categorizing various waveforms and morphologies within the signal. This is where machine learning techniques offer quantitative precision that qualitative assessment might overlook.

  Machine learning aids in automating this process by employing preprocessing techniques, handcrafted feature extraction, and abnormality detection at scale. Deep learning frameworks have demonstrated potential in acquiring more scalable and accurate representations of ECG signals, potentially surpassing human cardiologists in signal analysis. However, deep learning approaches require extensive datasets for training and may entail a considerable number of variables, necessitating efficient strategies such as transfer learning to address data scarcity issues

- ***From an economic point of view.***

  Individual heartbeat analysis assessments can be time-consuming and prone to diagnostic errors. Currently, there is a significant global burden of cardiovascular diseases. Consequently, from an economic perspective, accurate and cost-effective diagnosis of arrhythmic heartbeats is highly desirable. By automating ECG signal analysis through machine learning, healthcare systems have the potential to reduce the time and resources allocated to manual analysis while enhancing diagnostic accuracy. Transfer learning offers a cost-effective approach by leveraging pre-trained models and transferring knowledge between tasks, thereby optimizing the utilization of available data and computational resources.

  Once the algorithm is finalized and proven successful in providing accurate diagnostic assessments, it can be deployed at scale. Implementing a self-learning system and hyperparameter auto-tuning can ensure the robustness of the model. Additionally, if the implemented algorithm is transferable across different data sources using deep neural networks, it could lead to further market expansion by facilitating knowledge transfer across diverse data sources.

- ***From a scientific point of view.***

  From a scientific perspective, the application of machine learning techniques in ECG analysis holds the promise of significant advancements in healthcare informatics. Research in this area aims to develop robust models capable of accurately detecting anomalies in ECG signals, thereby facilitating early diagnosis and treatment of cardiovascular diseases. The proposed framework outlined in the original paper introduces a novel approach that leverages deep neural network architectures and transfer learning principles to enhance the representation and analysis of ECG signals, highlighting the potential for interdisciplinary collaboration between machine learning and healthcare research fields.

  There are several benefits associated with machine learning ventures in the medical field:

  - Demonstrating that machine learning can operate effectively with limited domain knowledge.
  - Demonstrating that machine learning can accurately predict complex diagnoses, provided it has access to the necessary data.
  - Assessing the feasibility of transfer learning across cardiovascular disease assessments.

## Objectives of the project

---

- **What are the main objectives to be achieved? Describe in a few lines.**

  1. Automating ECG Signal Analysis: The primary objective is to automate the analysis of ECG signals to address the challenges associated with manual review of the cardiac cycle, which can be time-consuming and may lead to incorrect diagnoses.

  2. Improving Diagnostic Accuracy: By employing machine learning techniques, the goal is to enhance the accuracy of diagnosing cardiovascular diseases, including arrhythmic heartbeats, which remain a leading cause of death globally.

  3. Advancing Deep Learning Methods: This study aims to advance deep learning techniques in healthcare informatics, particularly in the analysis of biomedical signals such as ECG, by developing more scalable and accurate models.

  4. Ensuring Performance Consistency: Given the sensitivity of the medical domain, it is crucial to ensure that the model performs consistently well across the test set, in addition to improving diagnostic accuracy.

- **For each member of the group, specify the level of expertise around the problem addressed?**

  Simon: Lacks expertise in the medical field, with limited experience in data science (has not extensively worked with data science).

  Hakan: Lacks expertise in the medical field, possesses knowledge in signal processing, has some initial work experience in data science using models such as support vector machine and random forest classifier.

  Alex: Lacks expertise in the medical field, holds an M.Sc. in Data Analytics, but lacks significant work experience in data science or modeling.

- **Have you contacted business experts to refine the problem and the underlying models? If yes, detail the contribution of these interactions.**

  The problem does fall within typical classification problems within the medical domain. As it currently stands, we are in preliminary phases of model selection.

- ***(Are you aware of a similar project within your company, or in your entourage? What is its progress? How has it helped you in the realization of your project? How does your project contribute to improving it?).***

  As none of us possess medical expertise, encountering such a project within our company is unlikely. Nevertheless, we are aware of resources that provide comparative analyses of various endeavors in this field.

## Understanding and manipulation of data

---

- **Which set(s) of data(s) did you use to achieve the objectives of your project?**

**Dataset 1: MIT- BIH Dataset**

- ECG recordings from 47 different people
- Sampling rate of 360 Hz
- Each beat was annotated by at least two independent cardiologists
- Annotations were used in this paper to identify 5 categories:
  - N: Normal, left-/right bundle branch block, atrial escape, nodal escape
  - S: Atrial premature, aberrant atrial premature, nodal premature, supra-venticular premature
  - V: Premature ventricular contraction, ventricular escape
  - F: Fusion of ventricular and normal
  - Q: Paced, Fusion, of paced and normal, unclassifiable

**Dataset 2: PD Diagnostic Dataset**

- Recordings from 290 subjects: 148 with myocardial infect (MI), 52 healthy controls, 7 with different disease
- 12 leads sampled at 1000Hz
- Paper uses only ECG lead 2 and using MI and healthy control groups

- **Are these data freely available? If not, who owns the data?**

  This data is freely available. Please see KaggleWebsite for more information. This data is owned by authors of the original paper.

- **Describe the volume of your dataset?**

| | Dataset | RangeIndex | Memory Usage (MB) | Number of 0-values | Number of NaN-values | Data Types |
|---|---|---|---|---|---|---|
| 0 | mitbih_test | 21891 | 31.4 | 0 | 0 | float64 (188) |
| 1 | mitbih_train | 87553 | 125.6 | 0 | 0 | float64 (188) |
| 2 | ptbdb_abnormal | 10505 | 15.1 | 0 | 0 | float64 (188) |
| 3 | ptbdb_normal | 4045 | 5.8 | 0 | 0 | float64 (188) |

**4) Relevance**

- **Which variables seem most relevant to you with regard to your objectives?**

  The datasets contain variables pertaining to electrocardiogram readings, representing time series data of heartbeats. Certain variables exhibit high correlations with both target and other variables. Notably, there are regions of heightened significance regarding prediction or correlation with the target variable. For further details, please refer to the "Visualization and Statistics" section.

- **What is the target variable?**

  **MIT-BIH Dataset** corresponds to target variable consisting of 5 types of heartbeats:

  1. N: Normal, left-/right bundle branch block, atrial escape, nodal escape
  2. S: Atrial premature, aberrant atrial premature, nodal premature, supra-venticular premature
  3. V: premature ventricular contraction, ventricular escape
  4. F: Fusion of ventricular and normal
  5. Q: Paced, Fusion of paced and normal, Unclassifiable

  **PD Dataset** corresponds to target variable consisting of 2 types of output:

  1. 1: Abnormal (MI Incidence)
  2. 0: Normal (No MI Incidence)

- **What features of your dataset can you highlight?**

  The following potential biases are identified:

  - Annotations were performed by only two individuals: The error rate of these annotators remains unknown.
  - The criteria utilized by the annotators to differentiate heartbeats into classes are undisclosed. This information would have been crucial for further data classification or for selecting specific parameters or constraints for subsequent models.
  - Classifications comprise various annotations from different individuals. While the correct class may be predicted, ambiguity arises regarding the specific entity within the class. Thus, human interpretation remains necessary. Even with 100% accuracy, determining the correct class may only provide insight into various interpretation possibilities.

- **Are you limited by some of your data?**

  1. An Imbalanced dataset can present challenges in terms of machine learning adaptability to test sets.

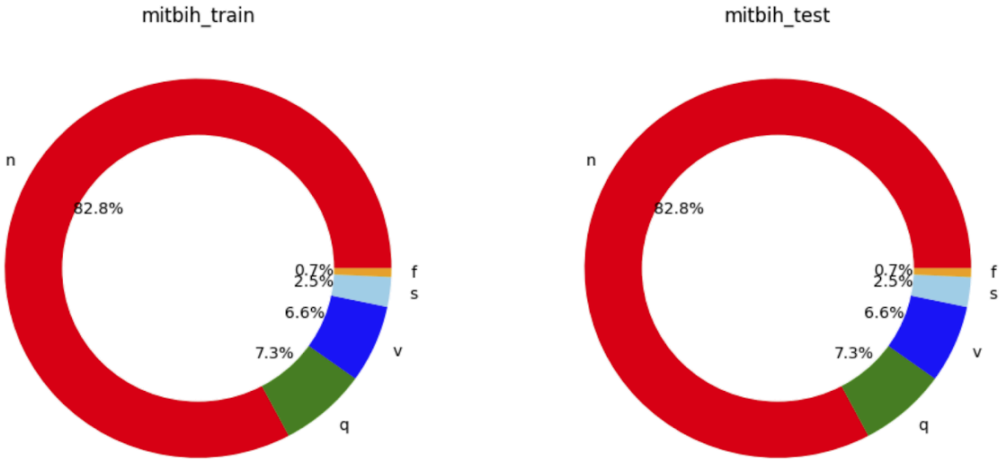



Fig. 1: Pie chart for the distribution of classes (mitbih shown as example, valid for all data)

  2. Column names are numeric values and cannot be used for plotting. As such the following assumption is present: Each column represents 1/1496 of a ms in an ECG as time series.
  3. Because of the data imbalance between classes, certain modeling algorithms might be needed, i.e. similar to those employed in Anomaly Detection type of problems.

**5) Pre-processing and feature engineering**

- **Did you have to clean and process the data? If yes, describe your treatment process.**

  The dataset contained no missing values, and categorical variables had already been converted to numerical format. Regarding the feature variables, preprocessing had been previously conducted by the original authors. Below is a summary of the preprocessing steps that were undertaken.

  Extraction of single ECG Beats in an R-R Interval (interval between two heartbeats, varies slightly for each heartbeat)

  Steps to reproduce:
  - Continuous ECG-Signal to 10s windows
  - Normalizing amplitude values to one and zero
  - Find all local maximums with zero crossings of deviation
  - Find RR-Peak candidates with threshold 0.9 for local maximums
  - Find median RR-time as nominal heartbeat time T
  - Stretch signaling part to 1.2 * T for each R-Peak (overlaps)
  - Pad each selected part with zeros, so they have all the same length (and starting point of R-peak)

  ***Further preprocessing may still take place in order to use the data for model fitting.

- **Did you have to carry out normalization/standardization type transformations of your data? If yes, why?**
  No normalization/standardization was needed as the data was already normalized to values between 0 and 1 for the amplitude of the ECG Signal in lead 2 (no. 2 of the 12 electrodes used in ECG-recording).

- **Are you considering dimension reduction techniques in the modeling part? If yes, why?**

  No. As our data represents important information on medical issues with the heart, we cannot afford to remove valuable data points which could point to abnormal heartbeats. We therefore do not use dimension reductions and take a longer time to train the model(s) in return.

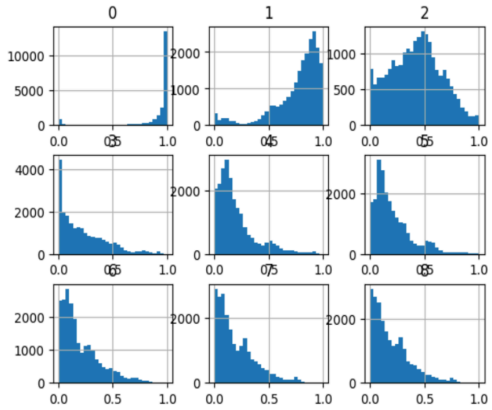
**6) Visualizations and Statistics**

- **Have you identified relationships between different variables? Between explanatory variables? and between your explanatory variables and the target(s)?**

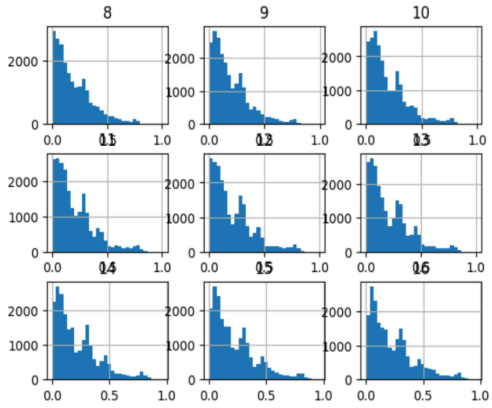
  We present the tests conducted in the order of their appearance in our data analysis notebook. The objective is to initially assess the underlying assumptions of statistical tests, followed by the exploration of correlation patterns in the data.

## Distributions of variables in Mit-Bih Test Set

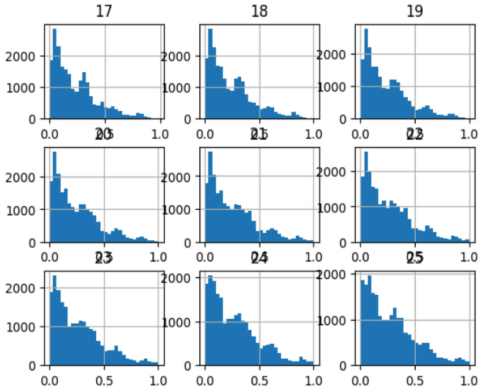
Distribution Plot for column [0:8] for Mit Bih-Test



Distribution Plot for column [8:16] for Mit Bih-Test



Distribution Plot for column [17:25] for Mit Bih-Test



Distribution Plot for column [27:35] for Mit Bih-Test

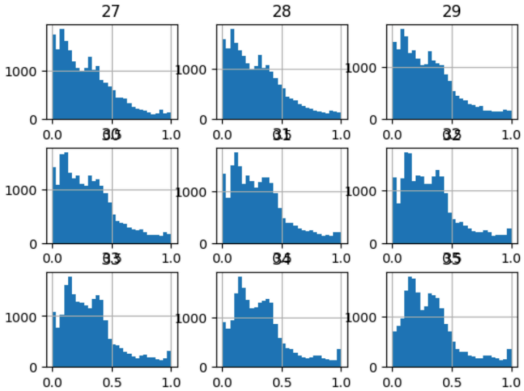


Fig. 2: Distributions of the independent variables (column-wise values) of the test data.

Upon reviewing distribution plots for numerical variables for Mit-Bih Test, the data does not appear to be normally distributed. Majority of the plots seem skewed. In order to confirm this, a Shapiro-Wilk test was performed. In the following diagram (Fig. 3), Shapiro-Wilk p-value results for all the explanatory columns have been displayed. The obtained low p-values of the Shapiro-Wilk test (below the significance level of 0.05) confirm that the Null hypothesis (that the variables are normally distributed) should be rejected. This shapiro wilk test was carried out for all available data with similar results.

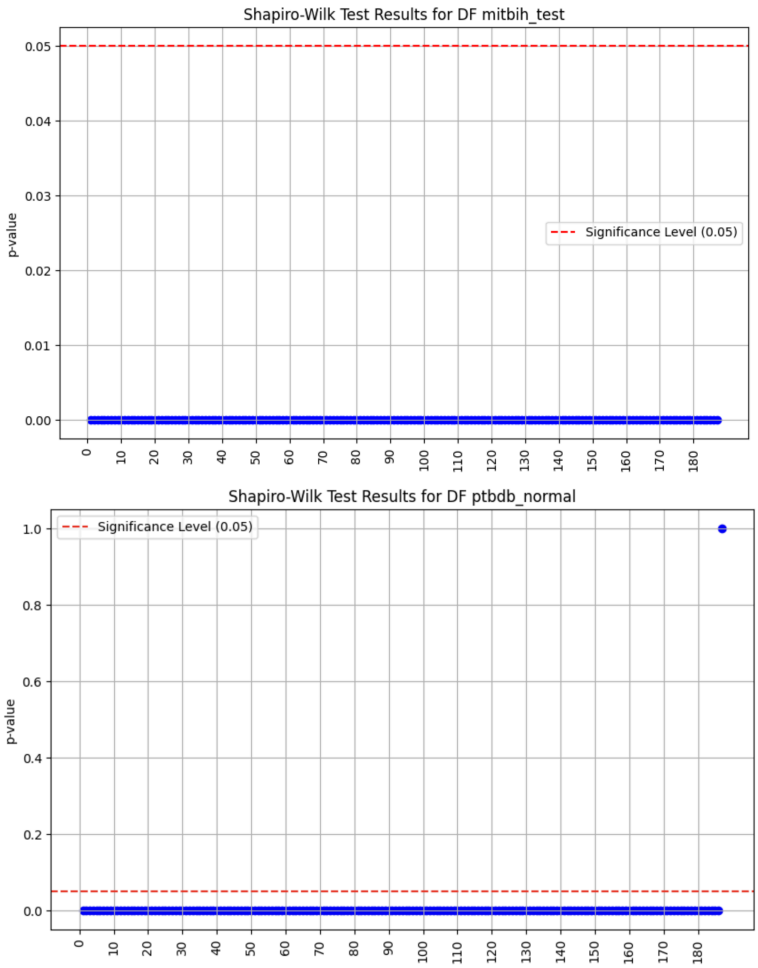
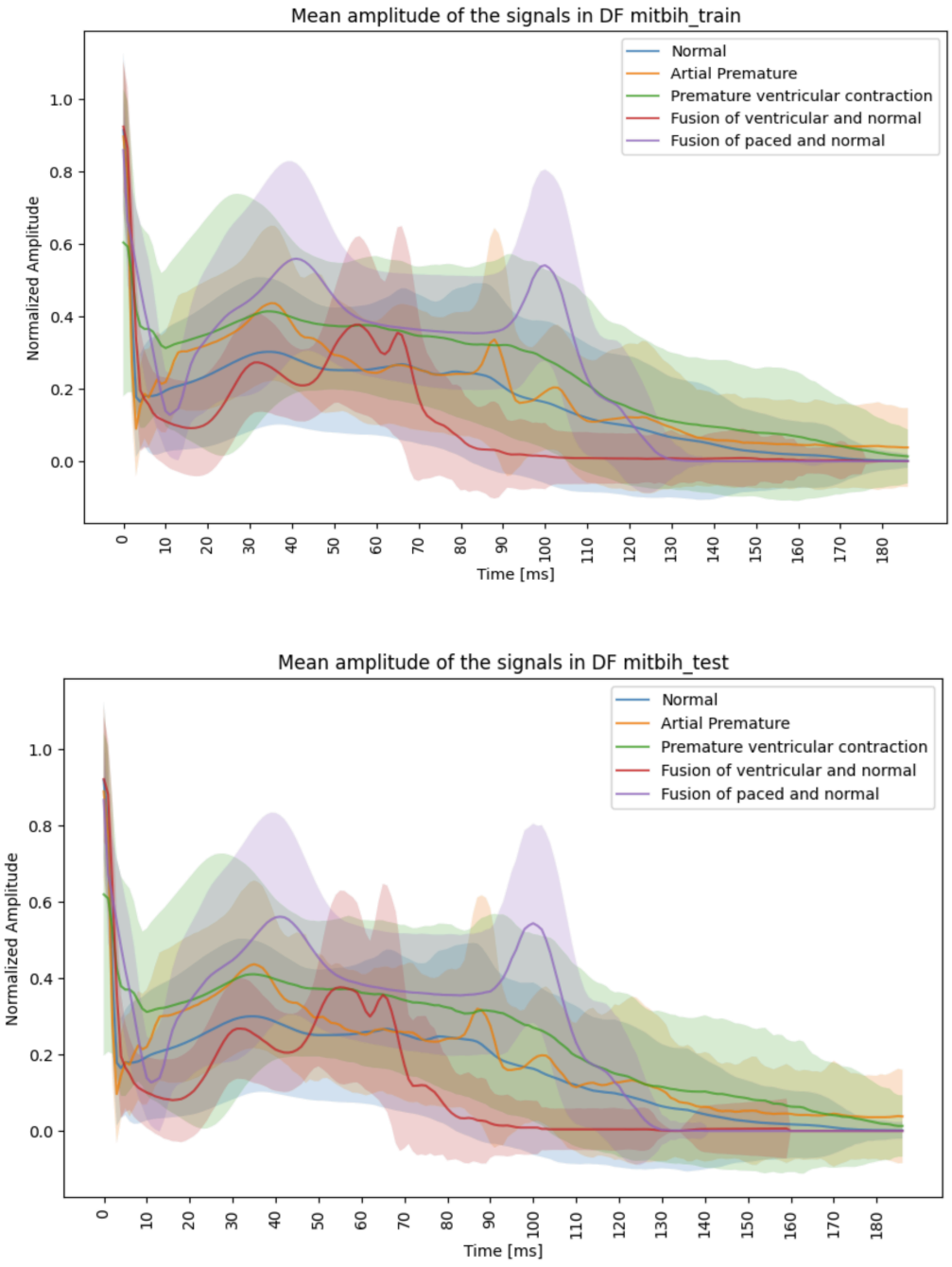



Fig. 3: The p-value results of the Shapiro-Wilk test for normal distribution

## Exploratory Analysis




Fg. 4: The mean amplitude of the signals per each class in train and test set

Upon observing the mean amplitude of the signals per each class in Fig. 4, it is easy to observe the overall data exhibits distinct characteristics. While individual heartbeats may vary slightly, the overarching patterns remain consistent. This observation suggests that perhaps the initial

milliseconds and mid-sections of data already characterize the temporal pattern of the signal. This hypothesis is further verified in the next sub-section through correlation analysis. These properties of the data allows one perhaps to implement highly accurate classification algorithms, for example, based on recurrent neural networks or LSTMs.

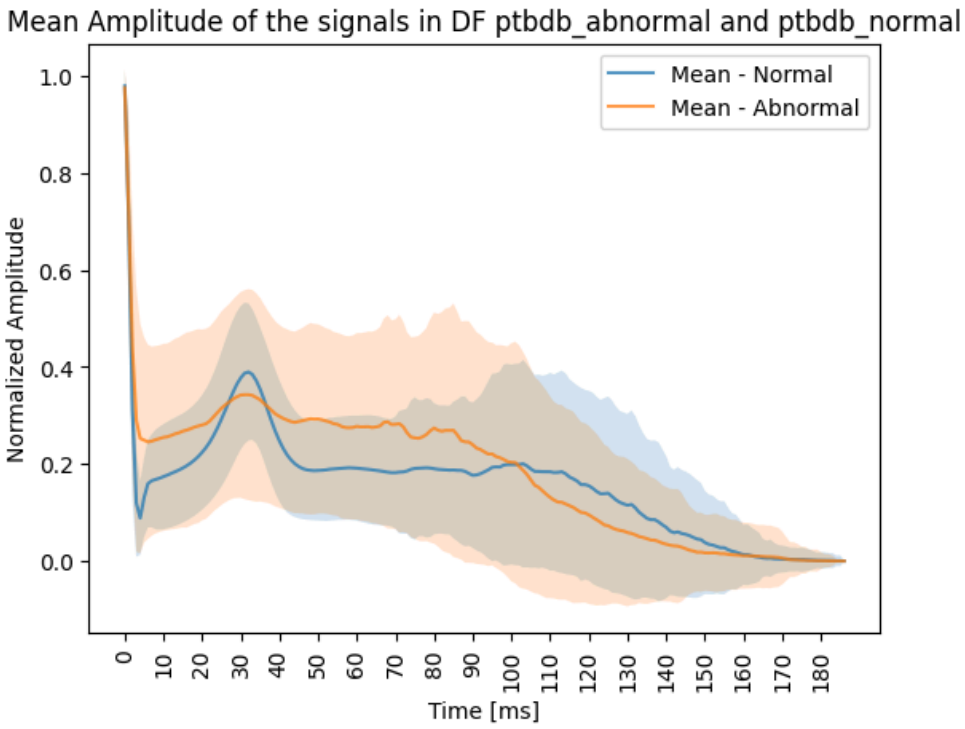



Fig. 5: The mean amplitude of the signals plotted for both PTB datasets.

Regarding the PTB dataset, the evaluation of normal heartbeats highlights significant differences from the mean and standard deviation of normal values. Although certain features, such as the first steep decline and rise, appear similar across heartbeats, heart rate variability becomes apparent. Conversely, abnormal data lacks the characteristic rise and fall seen in normal heartbeats, emphasizing the importance of considering the entire heartbeat rather than specific values in individual columns. This underscores the necessity of assessing the overall shape of the heartbeat for accurate classification.

### *Independence of Observation*

We integrate the search for correlations with the requisite statistical tests to ensure the independence of observations. We present a correlation matrix for all columns in each dataframe. In Fig. 6 such a correlation matrix is presented for the data frame mitbih_test. Herein, we exclusively present the visualizations for the MIT-BIH test dataset, while the interpretation text encompasses all datasets.

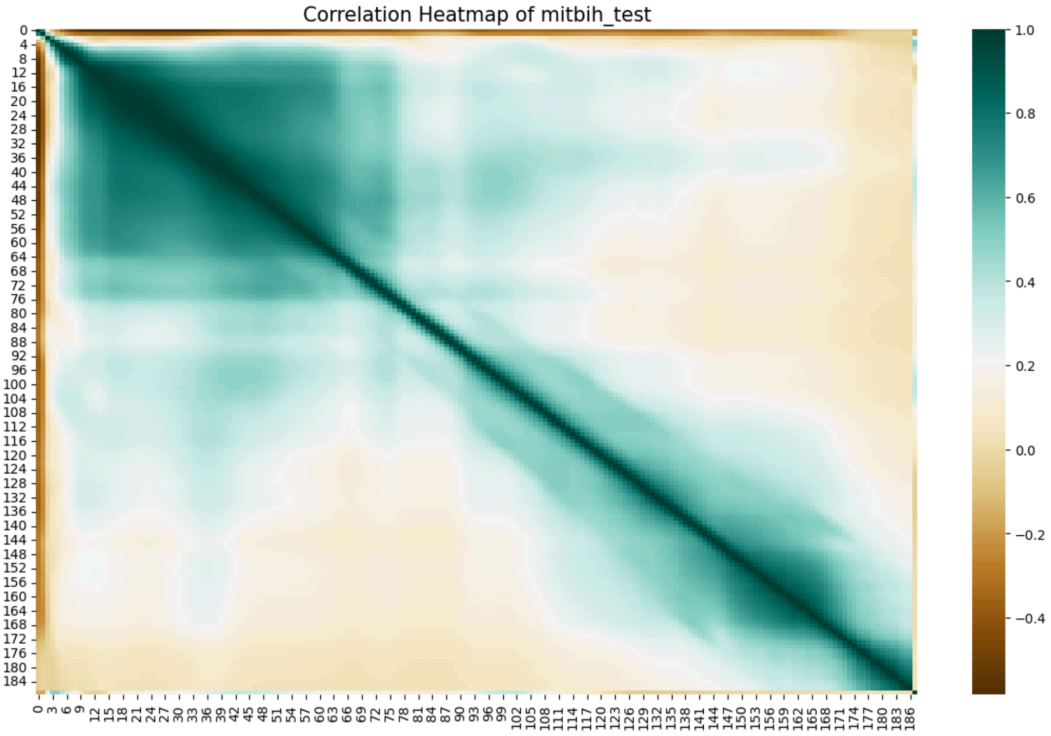



Fig. 6: Cross-correlation of all variables in the MIT-BIH test set

The interpretation of the correlation matrix yields valuable insights into the interrelationship among various columns within the datasets. The correlation heatmap provides a comprehensive overview of the correlations between different sets of values. Notably, both the MIT-BIH test and train data sets exhibit similar patterns, particularly in the initial milliseconds of data, suggesting a significant influence of the initial data on the overall behavior of the respective heartbeat.

In summary, the correlation matrix analysis underscores the importance of specific events in heartbeats and highlights potential differences in outlier influence between test and training datasets. Additionally, it emphasizes the necessity for careful consideration when evaluating datasets with preselected data, suggesting avenues for further investigation to enhance understanding and interpretation.

### *Outlier detection*

The following diagrams (Figures 7-9) present the Boxplot results for the 187 numerical variables which are further subdivided into heartbeat classification classes. There appears to be a presence of significant outlier values. Majority of outliers fall between column ~140 and 188. We can see no visible boxplot for these columns, as opposed to columns 1-140. This seems to be related to the original author's preprocessing steps where values were padded with zeros. Normal heartbeat type seems to be affected by outliers more than any other heartbeat class.

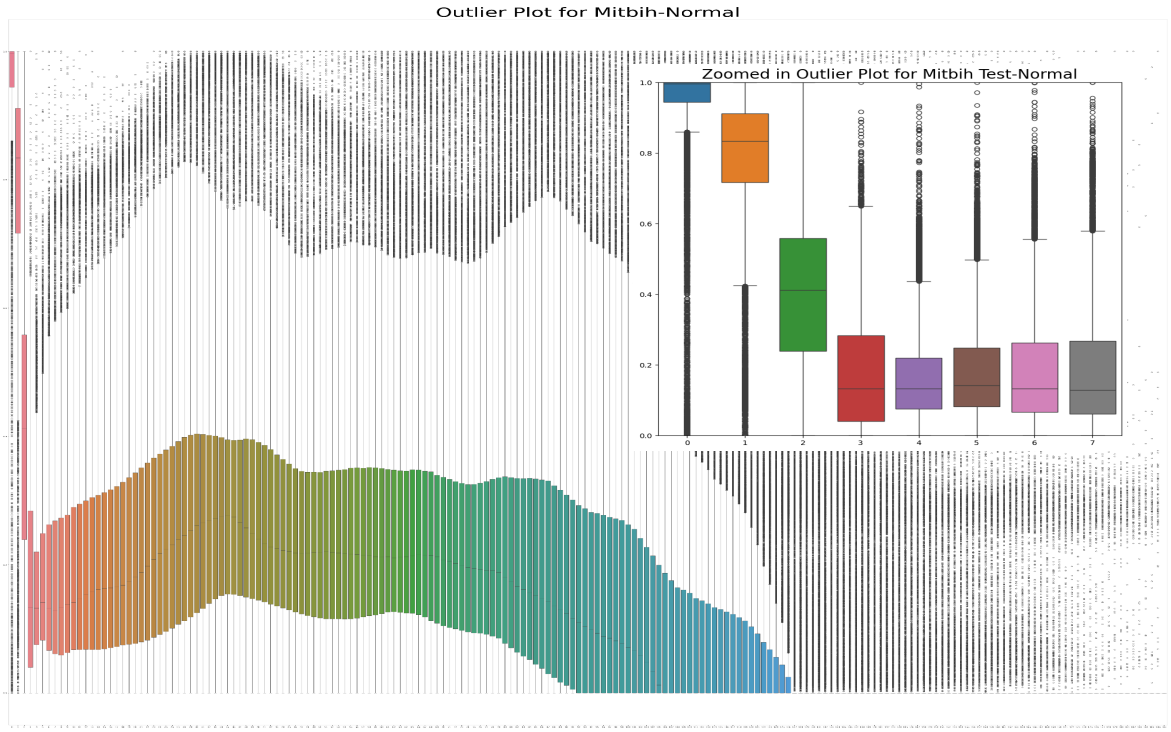



Fig. 7: Plots for the outlier distributions for MIT-BIH dataset- Normal class

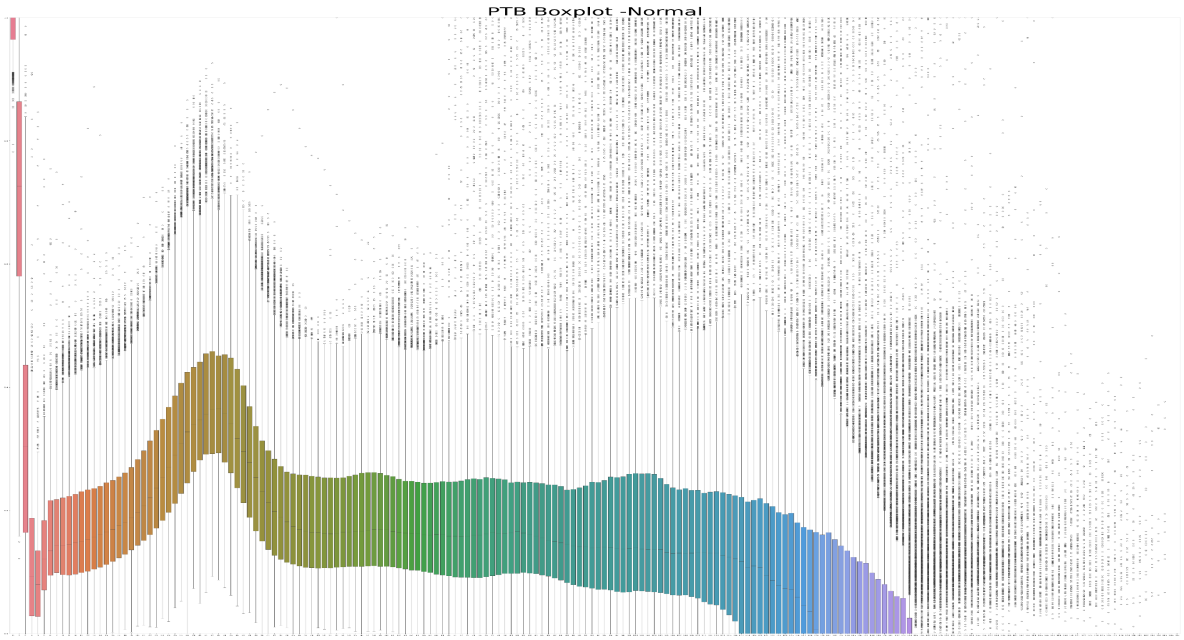



Fig. 8: Plots for the outlier distributions for PTB- Normal class

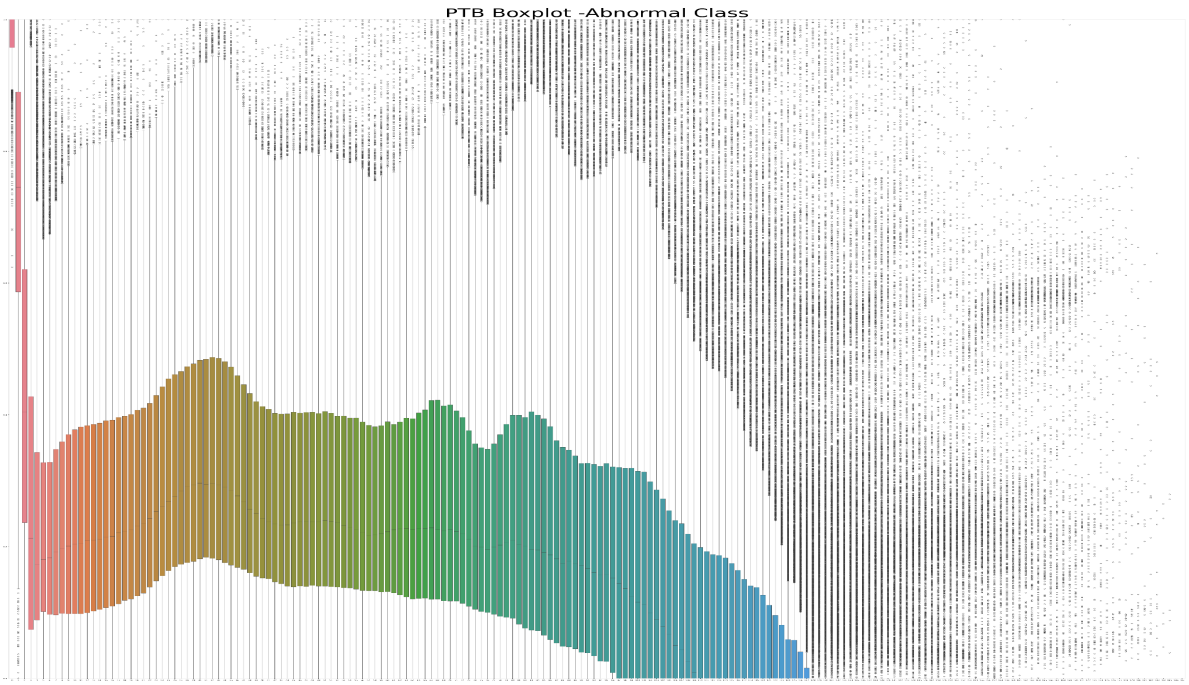



Fig. 9: Plots for the outlier distributions for PTB- Abnormal class

The test and train data sets exhibit similar outlier distributions, with the highest values, approximately 6 percent, observed at the significant onset and less significant termination of the datasets. Interestingly, outliers are sparse in areas corresponding to the first and second rises, suggesting these points as the most significant evaluation points.

Regarding the PTBDB dataset:

Although the shape differs from MIT-BIH, both abnormal and normal datasets display a similar pattern. Outliers increase until column 140 before declining again. Notably, discernible differences between normal and abnormal datasets suggest the absence of the second rise in abnormal subjects.

The following diagram displays comparison of full mitbih_train data vs data with outlier values removed. Mean across each column have been computed and compared. As padding resulted in a lot of zero values added at the end of the dataset, the actual values that are of importance could now be treated as outlier values because of this.

The original authors already pre-processed numerical variables to be between [0,1]. In order to verify if outliers should be removed, we would need to undo pre-processing. After that we would

need to seek guidance from a medical professional who is in a position to interpret the ECG + heartbeat charts and confirm if the outliers are a part of system error or intended abnormal values. As we are dealing with the medical domain, together with padding added at the bottom, we are running the risk of removing important data which can negatively affect future diagnosis. As such, it was determined to leave outlier data as is and include it for machine learning.

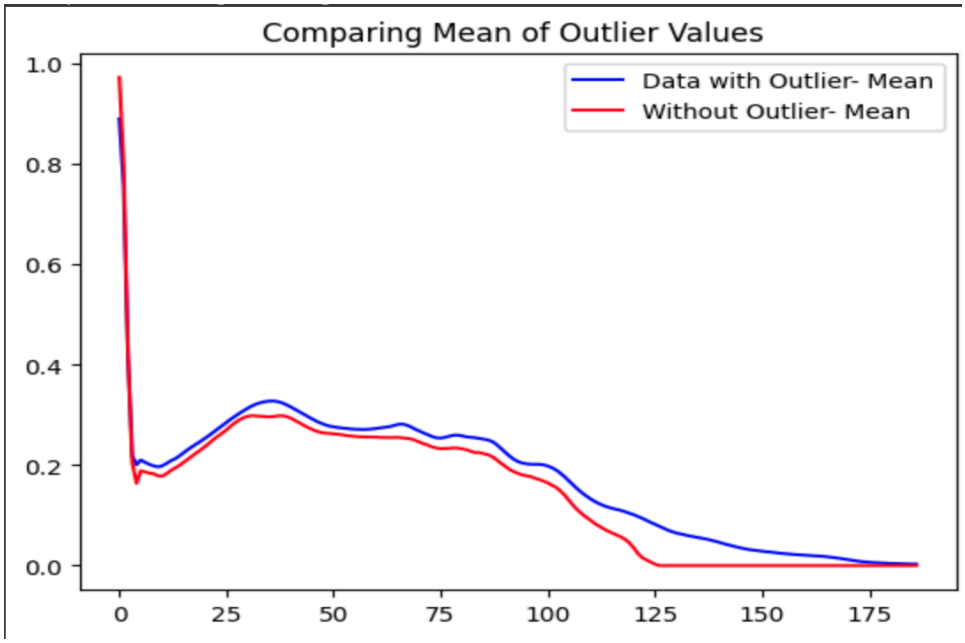



Fig. 10: Effect of the outliers on the mean amplitude of the signals

## Data Preprocessing

---

From the data exploration, we can conclude that we have imbalanced datasets in both cases. Also, normal distribution of values in the columns is not present. Since all other preprocessing steps (standardization, filling in with zeros) have been done by the providers of the dataset, we produce our Test and Train datasets, including splitting in feature (X) and target (y) dataframes, in three categories:

A) We do not resample the dataframes, but where necessary combine and shuffle them.
B) We resample the dataframes using the RandomOverSampler from imblearn.
C) We resample the dataframes using the RandomUnderSampler from imblearn.

With the balancing / resampling, we aim to produce better results for the targets that were underrepresented. We can check this later in the modeling step by using different datasets (from the Categories A, B or C). One can also compare the computational time and memory requirements of these datasets for different classification models.

Step A)

MITBIH:

- The values in the Test and Train datasets are splitted into features and target variables. The original value count of the targets is given in Fig. 1 as percentage values.

PTBDB:

- We generate a combined dataframe out of the provided data frames with abnormal and normal values. For more randomness, we shuffle this combined dataframe.
- From the combined dataframe, we generate our feature and target dataframes.
- From the feature and target data frames we generate our “original” train and test dataset using the test_train_split function.

Step B and C)

MITBIH:

- An oversampled training set is generated for both feature and target.
- The different value counts for the targets and percentages can be compared in Fig. 11: With RandomOverSampling, the amount of the data is considerably increased. Therefore, the computational time requirements need to be monitored for this case. With RandomUnderSampling, we reduce the overall data size quite drastically, but do not add random data. This would speed up the training process; however, the different classification models can be compared for prediction accuracy and/or precision.

PTBDB:

- The same oversampling/undersampling methods have been employed also for the PTB dataset, as for MIT-BIH. The data imbalance for the current case is less pronounced. However, the effects of balancing strategies can likewise be observed during the modeling stage.

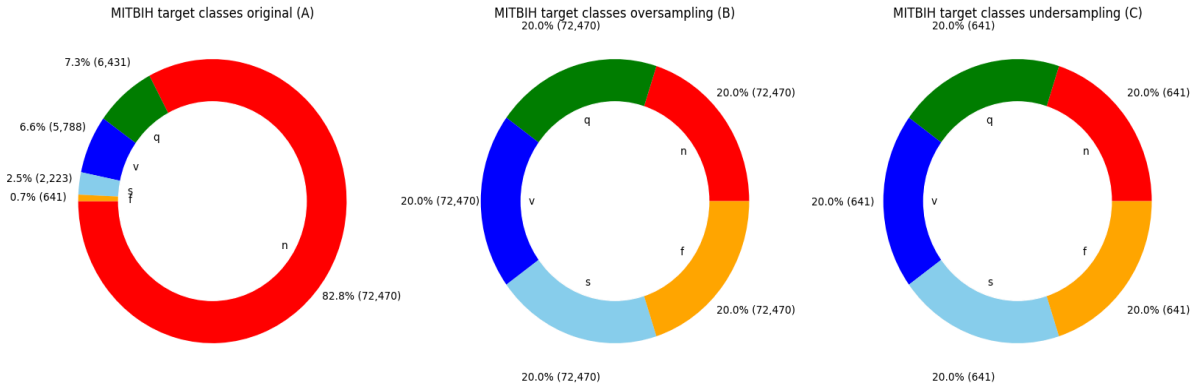



Fig. 11: Pie chart for the distribution of data after oversampling and undersampling.

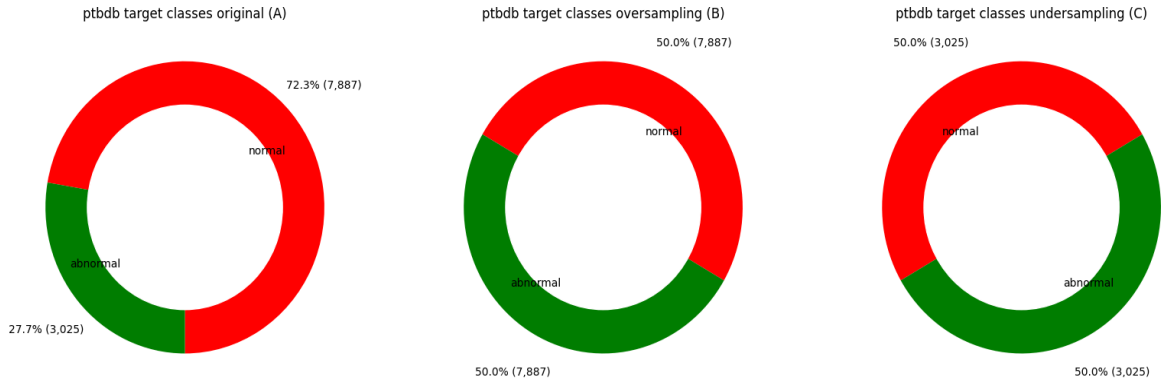



Fig. 12: Pie chart for the distribution of data after oversampling and undersampling for PTB dataset

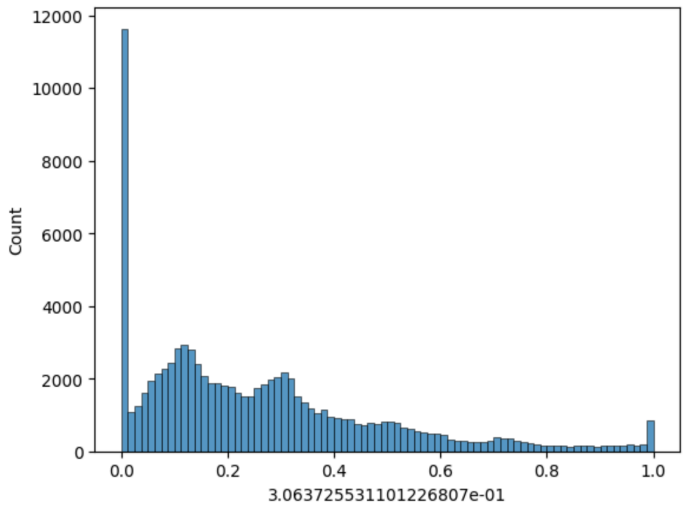
Conclusion: We did provide three different categories for test and training sets with which we can extensively test the upcoming models. By over or undersampling the datasets, we have provided a more balanced approach to identify the abnormal values of heartbeats. While the original authors of the dataset did not use such techniques, we aim to produce better training and predicting results by optimizing the target variable distribution in our training sets.

### ***Transformation approach***

As part of feature engineering, one can perform certain transformations in order to obtain a distribution that is closer to normal distribution.

In Figure 13, the mapping of the data frames to normal distribution with PowerTransformer (method: *yeo-johnson*) is presented. Despite the visual improvement towards a normal distribution, the calculation of the Shapiro-Wilk statistics on the whole transformed Data Frame still disproves a normal distribution, i.e. the p-value is calculated as 0. This is mainly due to the large amount of zero-crossings in the time series as seen in Fig. 13. Nevertheless, such variable transformations can be applied in the modeling phase and their effects on the classification results can be investigated.

Before transformation:




After transformation:

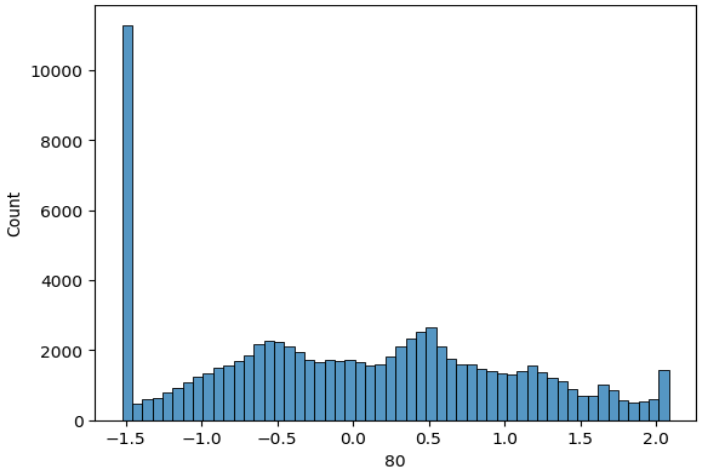



Fig. 13: Transformation of a feature variable (Column 80) in the MIT-BIH train set

## Difficulties encountered during the project

---

- **What was the main scientific obstacle encountered during this project?**

  There were no major obstacles identified during the project. All steps were accomplished in a timely manner, using the knowledge and practice gained through the bootcamp materials. We were able to perform a detailed data analysis on the dataset and implement a total of eight models: five ML models and three DL models for the given classification task. Additionally, in order to improve model performance, model optimization through parameter tuning has been conducted, and the final results were compared to the original model.

- **For each of the following points, if you encountered difficulties, detail how they slowed you down in setting up your project.**
- **Forecast: tasks that took longer than expected, etc.**

  As a general observation, no specific task took much longer than expected. Despite this, we can highlight that the Deep Learning section of the project required a lot of time to fine-tune the model. This project could have benefited from further experiments with deep learning to strengthen our experience and expertise with deep-learning models. For example, it would be interesting to experiment with adapting LSTM or RNN layers into the models to see if they would contribute to a positive shift in accuracy metrics.

- **Datasets: acquisition, volumetry, processing, aggregation, etc.**

  No issues were encountered during the data acquisition process. The MIT-BIH and ECG Database were extracted from [Ref 1]. Regarding data processing, since the data had already been pre-processed, no additional complexities were encountered. The only aggregation required was to merge the Normal Data from the ECG Database with the Abnormal data. No metric aggregation was deemed necessary for this project.

- **Technical/theoretical skills: timing of skill acquisition, skill not offered in training, etc.**

  One of the challenges encountered during the project execution revolved around the need for continuous learning at the same time as project execution. Certain advanced machine learning techniques necessary for this project did not coincide with the timing of the module curriculum learning. As such, in order to expedite skill development, we had to engage in ongoing learning of current materials together with future materials that fall outside of our assigned curriculum. However, despite this challenge, the resources available, including Kaggle notebooks and course materials, provided valuable support,

facilitating the acquisition of essential theoretical and practical skills required to successfully complete the project.

- **Relevance: of the approach, model, data, etc.**

  Our approach was centered around classification-type problems. By reviewing materials and approaches from other Kaggle authors tackling the same problem, we refined our approach with focus on enhancing the accuracy of the model prediction. We conducted experiments with various parameters and models to gain deeper insights into the essential steps of model selection and optimization. Data extraction was straightforward and led to minimal transformation. A significant advantage of this project was the opportunity to address real-life data challenges, providing exposure to important issues within the medical domain.

- **IT: storage power, computational power, etc.**

  After conducting the initial experiments with the classification models, we noticed that the computational capabilities of our personal devices could potentially affect the speed of our progress. To address this concern, we largely mitigated the issue by conducting computationally intensive simulations on the Kaggle platform. As a result, the structure of our notebooks was designed to run seamlessly on Kaggle. Nonetheless, we have rewritten the final notebooks to be executable independent of Kaggle.

## Classification of the problem

---

- ***What kind of machine learning problem is your project like? (classification, regression, clustering, etc)***

  This project is dealing with classification type problems. Our objective is to classify five types of heartbeats in the MITBIH dataset and determine whether the heartbeat is normal or not (PITBDB). The data was pre-labeled and split into training and testing sets before the project, and is publicly accessible at [1].

- ***What task does your project relate to? (fraud detection, facial recognition, sentiment analysis, etc)?***

  The task is in the medical field and we have to correctly predict the kind of heartbeat class for each ECG data. The task can be described as a typical classification problem.

- ***What is the main performance metric used to compare your models? Why this one?***

  As the main performance metric to compare the models, we use the **Accuracy Score**, which is a suitable performance indicator for machine learning performance. By definition, it is the number of correct predictions decided by the total predictions.

  Our primary project goal is to automate the evaluation of heartbeat ECGs through machine learning, with the aim of replacing the need for manual assessment. Given our engagement in the biomedical field, the occurrence of False Negatives in abnormal type classifications can potentially result in serious health consequences, thus rendering it of great importance for this project. The cost of mis-classified medical diagnosis is very high and can result in projects not being allowed to be deployed due to risk of incorrect diagnosis. Therefore, monitoring quantities such as true positives (TP), false positives (FP), false negatives (FN), etc., becomes crucial. Precision and Recall metrics encompass these measures and serve as significant performance indicators. As such, we employ these metrics for model comparison. The explicit formulas for Precision and Recall are given below.

  - **Precision**: True Positives compared to total number of positive predictions.

$$P = \frac{TP}{TP+FP}$$

  - **Recall:** Recall is the ratio of true positives to all REAL positives in ground truth.

$$R = \frac{TP}{TP+FN}$$

  - **F1-Score**: The F1-Score is the harmonic average of the Precision and Recall values. Therefore, it can be used as a main metric as well, in cases one would like to maximize the average of the afore-mentioned metrics. For a specific (trained) model, one can compromise between the recall and precision value, based on the threshold value used in the calculation of the metrics. Determining the maximum value of the F1-Score might be an optimal choice in such a case.

  To conclude, we aggregate and compare the accuracy score as the primary metric. Alongside Precision and Recall, we are able to evaluate the performance of the models

- ***Did you use other qualitative or quantitative performance metrics? If yes, detail it.***
  We use(d) plots of the loss-function and validation accuracy over the epochs and designed our experiments with the qualitative assessment of those. In the appendix some plots are shown for the MITBIH and PTBDB Dataset, but no further explanations are given.

# Model choice and optimization

---

- ***What algorithms have you tried?***

  We have tried two groups of algorithms: traditional machine learning methods and two types of neural networks, i.e. one (simple) artificial neural network and one (simple) convolutional neural network. In terms of the classical ML methods, we have tried five algorithms: Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Decision Tree, Random Forest and Extreme Gradient Boosting (XGBoost).

  ***Traditional Machine Learning Methodology***

  As mentioned in detail in the first report, we have two datasets, namely the MITBIH dataset and the PTBDB dataset. One important note is that the MITBIH dataset has been previously split into a training and testing set, whereas the PTBDB dataset is provided as two separate datasets consisting of two classes: “normal” and the “abnormal” data [Ref. 1]. Therefore, for the latter, we concatenate the normal and abnormal data classes and apply a unique train-test split on our own, using the usual 20% test partition option. Moreover, we apply oversampling and undersampling strategies onto training datasets to balance the datasets with respect to the occurrence of classes. As a result, including the resampled datasets, we have six datasets where we applied the algorithms. These are:

  - MITBIH_A (original)
  - MITBIH_B (oversampled using SMOTE)
  - MITBIH_C (undersampled)
  - PTBDB_A (original)
  - PTBDB_B (oversampled using SMOTE) and
  - PTBDB_C (undersampled)

  After performing machine learning, further optimization using GridSearch has taken place to further optimize performance of our models.

  The results of the ML algorithms on all datasets will be presented in a separate section below.

  *Deep Learning*

  After traditional machine learning, deep learning was introduced on the same datasets. In relation to undersampled datasets, due to performance disparity, those datasets were dropped in favor of original/oversampled. The following datasets were used for Artificial Neural Network and Convoluted Neural Network :

  - MITBIH_A (original)
  - MITBIH_B (oversampled using SMOTE)
  - PTBDB_A (original)
  - PTBDB_B (oversampled using SMOTE)

In addition to building ANN/CNN models, we have also conducted parameter tuning experiments on deep learning models in an effort to determine if this would lead to improvements in the accuracy of predictions.

- **Describe which one(s) you selected and why?**

  As this project is a classification-type problem, we have utilized the content of the classification modules of the course as a reference point for our models selection. We have attempted to include various methods in our modeling in an effort to compare and weed out the best result for this project.

  It is worth noting that boosting and bagging-based methods might perform better than other geometrical fitting models, such as SVM and KNN. Similar outcomes have been observed in our results, which will be detailed in the coming sections.

- ***Did you use parameter optimization techniques such as Grid Search and Cross Validation?***

  We have utilized the GridSearchCV method from scikit-learn extensively for all ML algorithms. The results are presented in the Results section.

- ***Have you tested advanced models? Bagging, Boosting, Deep Learning… Why?***

  We have tested Boosting (XGBoost) and Deep Learning. For the former, we have also applied GridSearch as mentioned above. Regarding Deep Learning methods, we conducted extensive experiments and hyperparameter tuning to enhance model scores. Bagging was inadvertently omitted from the project from the get go.

## Modeling Results

---

### *Traditional Machine Learning Results*

In this section, the results for different algorithms are presented in detail. First, we present the model results for traditional machine learning algorithms for both datasets. Table 1 consists of performance results of the model without any hyper-parameter tuning. The metrics were calculated using the classification_report function in the Sklearn.metrics module. The results are displayed for the weighted averages of the metrics. As previously mentioned, the models were trained using different resampling techniques but tested on the unaltered test datasets. Table 1 shows the results for the MITBIH dataset for five different ML algorithms used in our modeling. The best-performing model for each dataset A, B, and C is presented in bold font. The XGBoost algorithm provides the highest score for all metrics for the Mitbih_A dataset. For the

oversampled dataset (using SMOTE), the RandomForest algorithm gives the best scores for all metrics.

Table 1: Results for the MITBIH dataset, for all ML models

| Method | Precision | Recall | F1 | Accuracy | DATASET |
|---|---|---|---|---|---|
| SVM | 0.9676 | 0.968 | 0.9657 | 0.968 | Mitbih A |
| KNN | 0.9727 | 0.9736 | 0.9725 | 0.9736 | Mitbih A |
| Decision Tree | 0.9548 | 0.9548 | 0.9548 | 0.9548 | Mitbih A |
| Random Forest | 0.9745 | 0.9746 | 0.9729 | 0.9746 | Mitbih A |
| **XGB** | **0.9804** | **0.9808** | **0.9799** | **0.9808** | **Mitbih A** |
| SVM | 0.9629 | 0.9366 | 0.9462 | 0.9366 | Mitbih B |
| KNN | 0.966 | 0.9551 | 0.959 | 0.9551 | Mitbih B |
| Decision Tree | 0.9499 | 0.9402 | 0.9439 | 0.9402 | Mitbih B |
| **Random Forest** | **0.9806** | **0.9811** | **0.9808** | **0.9811** | **Mitbih B** |
| XGB | 0.9727 | 0.9684 | 0.97 | 0.9684 | Mitbih B |
| SVM | 0.9278 | 0.8344 | 0.8715 | 0.8344 | Mitbih C |
| KNN | 0.9195 | 0.7671 | 0.8204 | 0.7671 | Mitbih C |
| Decision Tree | 0.8975 | 0.7072 | 0.7673 | 0.7072 | Mitbih C |
| Random Forest | 0.9421 | **0.8743** | **0.8998** | **0.8743** | **Mitbih C** |
| **XGB** | **0.943** | 0.8716 | 0.8978 | 0.8716 | Mitbih C |

After conducting machine learning analyses on various datasets, we proceeded to perform hyperparameter tuning on the MITBIH Original Dataset (A). Due to performance issues observed with the undersampled dataset, it was decided to exclude the undersampling technique from future optimization efforts, instead focusing extensively on grid search for the high-performing dataset. Although oversampling techniques yielded positive outcomes, the improvements compared to the original dataset were considered insignificant. Despite minor changes in performance, the use of an oversampled dataset significantly prolonged machine learning training. Consequently, it was also discarded in favor of the original dataset. The results for the best estimator for each model are presented in the table below. We calculated the difference between the optimized and non-optimized models in order to showcase the effect of hyperparameter tuning on machine learning performance.

Fig. 14 showcases a comparison of machine learning performance between optimized and unoptimized hyperparameter tuning. Only slight changes are observed with extensive GridSearch. In terms of the best-performing models, RF and XGB remain in the lead. Despite this, the improvements in performance with parameter optimization are quite minor. The objective of GridSearch is clearly twofold:

I. the improvement of scores through hyperparameter tuning and

II. the prevention of overfitting.

On its own, unoptimized machine learning results for the original dataset already deliver solid results. With balanced train-test splits (also provided in the original data), the overall scores are quite satisfactory for our classification problem, as seen in Table 1. Therefore, the small changes in Fig. 14, although minor, are still favorable. Furthermore, the application of a cross-validation strategy facilitates the prevention of overfitting. Although not directly observed by comparing Table 1 and Fig. 14, once again, due to the high scores, this was also the case in our experiments. For instance, the train accuracy obtained with the RF method with default parameters was 1.0, whereas the train accuracy for the RF method with cross-validation was obtained as 0.979

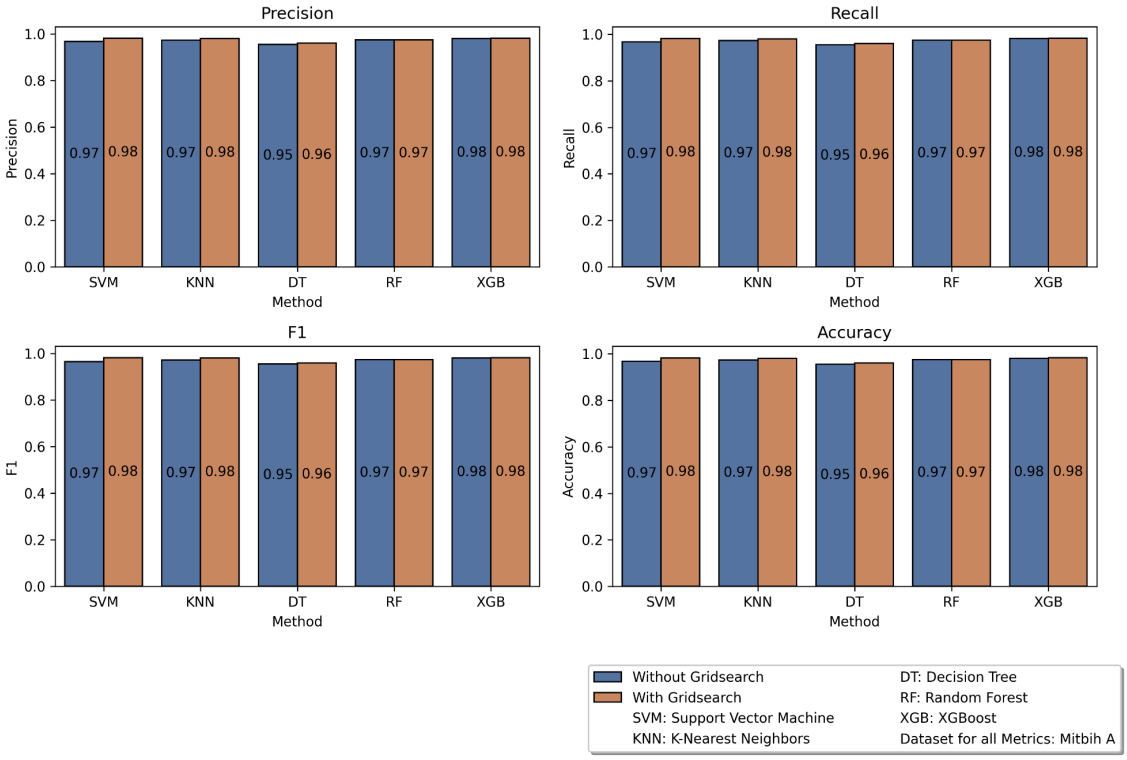



Fig.14: Metrics (weighted averages) for simple ML Models on MITBIH A. Differentiated for default and (best) gridsearch values.

As previously mentioned, the focus on predicting abnormal classes is of high importance when dealing with the medical domain and medical diagnostics. In Table 2, the metrics of the Random Forest Classifier were selected for comparison of class metrics with and without gridsearch. Despite minor differences, we can clearly see how parameter hypertuning can lead to improvements in predicting both normal and abnormal classes.

Table 2: Comparison result for MITBIH Random Forest machine learning with hypertuning vs without

| Method | Metric | Class-Normal | Class Premature | Class-Ventricular | Class V and N fused | Class Paced |
|---|---|---|---|---|---|---|
| RFC | precision | 97.29% | 97.95% | 98.23% | 87.72% | 99.35% |
| RFC | recall | 99.91% | 60.07% | 88.40% | 61.73% | 94.40% |
| RFC | f1-score | 98.58% | 74.47% | 93.06% | 72.46% | 96.81% |
| RFC-GridSearch | precision | 97.31% | 98.54% | 98.53% | 88.89% | 99.48% |
| RFC-GridSearch | recall | 99.92% | 60.79% | 88.19% | 64.20% | 94.84% |
| RFC-GridSearch | f1-score | 98.60% | 75.19% | 93.08% | 74.55% | 97.10% |

The results for the PTBDB dataset, obtained using the ML models, are shown in Table 3. Similar to the MITBIH data, the Random Forest and XGB models exhibit very comparable results, which are considerably better than those obtained with SVM, KNN, and Decision Tree. The difference in scores between the RF and XGB methods, obtained for all datasets (A, B, and C), is minor, with the overall best accuracy obtained with XGB for the original data (PTBDB_A).

Table 3. Results for PTBDB dataset, for all ML models.

| Method | Precision | Recall | F1 | Accuracy | DATASET |
|---|---|---|---|---|---|
| SVM | 0.9126 | 0.9138 | 0.9126 | 0.9138 | Ptbdb A |
| KNN | 0.9339 | 0.932 | 0.9326 | 0.932 | Ptbdb A |
| Decision Tree | 0.9283 | 0.9285 | 0.9284 | 0.9285 | Ptbdb A |
| Random Forest | 0.974 | 0.9739 | 0.9736 | 0.9739 | Ptbdb A |
| **XGB** | **0.9797** | **0.9797** | **0.9797** | **0.9797** | Ptbdb A |
| SVM | 0.9159 | 0.9045 | 0.907 | 0.9045 | Ptbdb B |
| KNN | 0.9322 | 0.9206 | 0.9228 | 0.9206 | Ptbdb B |
| Decision Tree | 0.9215 | 0.92 | 0.9205 | 0.92 | Ptbdb B |
| **Random Forest** | 0.9759 | 0.976 | 0.9759 | 0.976 | Ptbdb B |
| **XGB** | **0.9763** | **0.9763** | **0.9763** | **0.9763** | Ptbdb B |
| SVM | 0.8941 | 0.8674 | 0.8725 | 0.8674 | Ptbdb C |
| KNN | 0.8987 | 0.8681 | 0.8735 | 0.8681 | Ptbdb C |
| Decision Tree | 0.914 | 0.9076 | 0.9093 | 0.9076 | Ptbdb C |
| Random Forest | 0.9573 | 0.9543 | 0.955 | 0.9543 | Ptbdb C |
| **XGB** | **0.9663** | **0.9646** | **0.965** | **0.9646** | Ptbdb C |

***Conclusion for ML models:***

- We obtain very good results for each model regarding overall accuracy. With the usage of non-optimized standard models, we achieve accuracy values that surpass those of the original authors [2].

- Among all traditional machine learning models, the best-performing models tend to be Random Forest and XGB. This could be attributed to their nature of fitting the data almost exclusively with sufficient depth.

- Through the usage of GridsearchCV, we could achieve minor improvements to each model, as shown in Fig 14 above. However, we could not produce significantly better results overall, such that one specific model outperforms the rest.

- We have witnessed the possibility of making our class predictions favor the sick classes (higher recall for those). This would lead to a more “conservative” model, which is favourable in our medicinal project. However, the further implementation of such conservative models was not possible due to the following reasons: The *scoring* parameter of the GridSearchCV function allows us to input the metrics defined in *sklearn* library. A grid-search with respect to the scores of a certain class could be only possible by defining metric functions of our own. As a result, it would be very time-consuming to perform all the numerical experiments for each model, abnormal class, and internal parameters of each model.

- Due to SearchGrid, improved scores for class-wise results, specifically for classifying abnormal classes correctly, have been displayed in Table 2.

***Deep Learning***

The next stage of our modeling phase was to attempt Deep Learning Models. We applied both Artificial Neural Networks (ANN) and Convolutional Neural Networks (CNN) to our dataset to explore whether performance could be further improved.

As a starting point, we utilized the Artificial Neural Network provided in the GitHub repository [3] for the PTBDB dataset. This model primarily consisted of dense layers. The ANN model from the referenced source was initially employed for the PTBDB dataset, and subsequently, it was further adapted and applied to the MITBIH dataset in our current project. We made slight modifications to this algorithm by adjusting the output layer size and the loss function for the MITBIH dataset.

*Fig. 15* shows the model architecture that is modified in order to train and test for the MITBIH dataset. In building the ANN model, the LeakyReLu function was used as the activation function, with a negative slope (alpha) value of 0.001. The optimizer is chosen as Adam, and the loss function is set as Sparse Categorical Cross-Entropy. The activation layer of the output layer is the softmax function.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 60)                11280

 dense_1 (Dense)             (None, 20)                1220

 dense_2 (Dense)             (None, 20)                420

 dense_3 (Dense)             (None, 20)                420

 dense_4 (Dense)             (None, 5)                 105

=================================================================
Total params: 13445 (52.52 KB)
Trainable params: 13445 (52.52 KB)
Non-trainable params: 0 (0.00 Byte)
```

Fig. 15**.** Model structure of the ANN model taken from Ref. [3] and modified for MITBIH dataset

It is well-known that convolutional neural networks (CNN) can perform very well for specific classification problems. To explore the advantages inherent in convolutional filters, we modified the algorithm from Fig. 15 by adding a convolutional layer. Given that we are dealing with time-series-like data, the use of the Conv1D layer is appropriate. The final model architecture for the MITBIH dataset is depicted in Fig. 16. Similarly, this CNN algorithm was also employed for the PTBDB dataset, with different output dimensions. Similar to ANN, the CNN models were constructed and compiled with identical activation, learning, and loss parameters. We configured the Conv1D layer with a filter size of 3 and a total of 32 filters.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 185, 32)           128

 flatten (Flatten)           (None, 5920)              0

 dense (Dense)               (None, 60)                355260

 dense_1 (Dense)             (None, 20)                1220

 dense_2 (Dense)             (None, 20)                420

 dense_3 (Dense)             (None, 5)                 105

=================================================================
Total params: 357133 (1.36 MB)
Trainable params: 357133 (1.36 MB)
Non-trainable params: 0 (0.00 Byte)
```

Fig 16. Model structure of the Simple CNN algorithm

**Have you changed the model since the last iteration? If yes, provide details.**

In our second report, we deployed 5 machine learning (ML) models and 2 deep learning (DL) models. Subsequently, we have introduced modifications to the convolutional neural network (CNN) model structure. The updated version of the CNN model yields superior performance. Fig. 17 illustrates the revised architecture of the CNN model. The primary distinctions between the prior and current configurations of the CNN encompass: 1) the incorporation of a max_pooling1d layer, 2) an additional convolution1d layer, 3) another max_pooling1d layer, and 4) the inclusion of a dropout layer.

| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv1d (Conv1D) | (None, 185, 32) | 128 |
| max_pooling1d (MaxPooling1D) | (None, 92, 32) | 0 |
| conv1d_1 (Conv1D) | (None, 90, 16) | 1552 |
| max_pooling1d_1 (MaxPooling1D) | (None, 44, 16) | 0 |
| flatten (Flatten) | (None, 704) | 0 |
| dropout (Dropout) | (None, 704) | 0 |
| dense (Dense) | (None, 120) | 84600 |
| dense_1 (Dense) | (None, 60) | 7260 |
| dense_2 (Dense) | (None, 20) | 1220 |
| dense_3 (Dense) | (None, 5) | 105 |

Total params: 94865 (370.57 KB)
Trainable params: 94865 (370.57 KB)
Non-trainable params: 0 (0.00 Byte)

Fig. 17: Model structure of the new CNN version

***Experiment design for Deep Learning models:***

After the basic structures of the models were fixed, the following parameter tuning was applied and compared. In order to automatically log the results and further influence the preservation of the models, we implemented three callback functions:
1) Early Stop Callback: simple early stopping function from tf.keras.callbacks. The validation accuracy is monitored. Patience is configured for each experiment (see table 4).
2) Model Checkpoint Callback: simple model checkpoint function from tf.keras.callbacks. Only the best results regarding the validation accuracy are saved. Furthermore only weights are saved. This callback is not changed in any way during experiments, so for each experiment the best model is available.

3) Learning Rate Scheduler callback: simple learning rate scheduler function from tf.keras.callbacks. The adaptation of learning rate is fixed for the value of 10 epochs. The learning rate reduction rate is adjustable, but was fixed to 0.5 for all experiments, where the learning rate scheduler callback was used. With this configuration, the (initial) learning rate was halved each 10 epochs.

Table 4 provides a summary of experiments performed and hyperparameter adjustments.

Table 4: Experiment Design for Deep Learning Models

| **Experiment Design** | **Model Type** | **Changes Made** |
|---|---|---|
| Experiment 1 | Simple ANN<br>Simple / Advanced CNN | Epochs: 70<br>batch_size = 10<br>patience = 10<br>Initial_Learning_Rate = 0.001 |
| Experiment 2 | Simple ANN<br>Simple / Advanced CNN | Epochs: 70<br>batch_size = 10<br>patience = 10<br>Initial_Learning_Rate: 0.0001 |
| Experiment 3 | Simple ANN<br>Simple / Advanced CNN | Epochs: 70<br>batch_size = 10<br>patience = 70<br>Initial_Learning_Rate: 0.0001<br>Early Stop Callback is essentially deactivated. |
| Experiment 4 | Simple ANN<br>Simple / Advanced CNN | Epochs: 70<br>batch_size: 10<br>patience: 70<br>Initial_Learning_Rate: 0.001<br>lr_reduction_rate: 0,5 @10 Epochs<br>Early Stop Callback is essentially deactivated. Learning rate is halved each 10 epochs, initial lr is used. |

***Deep Learning Results***

MITBIH:

Table 1 displays model results for all experiments conducted for Simple ANN/CNN and Advanced CNN of MITBIH Dataset. The best scoring modes are highlighted in bold font based on the **Accuracy score** of each experiment.

Table 5: MITBIH results with DL models

| Method | Precision | Recall | F1 | Accuracy | DATASET |
|---|---|---|---|---|---|
| SIMPLE ANN - 1 | 0.977 | 0.9784 | 0.9778 | 0.9784 | Mitbih A |
| SIMPLE ANN - 2 | 0.9759 | 0.9767 | 0.9761 | 0.9767 | Mitbih A |
| SIMPLE ANN - 3 | 0.9738 | 0.9746 | 0.973 | 0.9746 | Mitbih A |
| **SIMPLE ANN - 4** | **0.977** | **0.9785** | **0.9779** | **0.9785** | Mitbih A |
| SIMPLE ANN - 1 | 0.965 | 0.9482 | 0.9544 | 0.9482 | Mitbih B |
| SIMPLE ANN - 2 | 0.9702 | 0.9671 | 0.9683 | 0.9671 | Mitbih B |
| SIMPLE ANN - 3 | 0.9698 | 0.9684 | 0.969 | 0.9684 | Mitbih B |
| **SIMPLE ANN - 4** | **0.9716** | **0.9704** | **0.9709** | **0.9704** | Mitbih B |
| SIMPLE CNN - 1 | 0.9787 | 0.9794 | 0.9787 | 0.9794 | Mitbih A |
| SIMPLE CNN - 2 | 0.978 | 0.9788 | 0.9782 | 0.9788 | Mitbih A |
| SIMPLE CNN - 3 | 0.9771 | 0.9778 | 0.9773 | 0.9778 | Mitbih A |
| **SIMPLE CNN - 4** | **0.9788** | **0.9794** | **0.9789** | **0.9794** | Mitbih A |
| SIMPLE CNN - 1 | **0.9773** | **0.9774** | **0.9773** | **0.9774** | Mitbih B |
| SIMPLE CNN - 2 | 0.9746 | 0.9729 | 0.9736 | 0.9729 | Mitbih B |
| SIMPLE CNN - 3 | 0.9761 | 0.9761 | 0.9761 | 0.9761 | Mitbih B |
| SIMPLE CNN - 4 | 0.9754 | 0.9754 | 0.9754 | 0.9754 | Mitbih B |
| ADVANCED CNN - 1 | 0.981 | 0.9815 | 0.9808 | 0.9815 | Mitbih A |
| ADVANCED CNN - 2 | 0.9846 | 0.9849 | 0.9845 | 0.9849 | Mitbih A |
| ADVANCED CNN - 3 | 0.9848 | 0.9852 | 0.9847 | 0.9852 | Mitbih A |
| **ADVANCED CNN -** 4 | **0.9853** | **0.9857** | **0.9853** | **0.9857** | Mitbih A |
| ADVANCED CNN - 1 | 0.9796 | 0.9788 | 0.9791 | 0.9788 | Mitbih B |
| ADVANCED CNN - 2 | 0.9761 | 0.9725 | 0.9739 | 0.9725 | Mitbih B |
| ADVANCED CNN - 3 | 0.9792 | 0.9773 | 0.978 | 0.9773 | Mitbih B |
| **ADVANCED CNN - 4** | **0.9814** | **0.981** | **0.9812** | **0.981** | Mitbih B |

When it came to performance, according to the original author's metrics, their deep residual CNN achieved an average accuracy of 93.5% on the MIT-BIH dataset. Our advanced CNN-4 model resulted in an accuracy of 98.57% on the original MIT-BIH dataset, while the advanced CNN-4 model trained on the oversampled dataset attained an accuracy of 98.1%.

The most favorable performance for the MIT-BIH dataset was achieved by the Advanced CNN model utilizing the parameters outlined in Experiment 4.

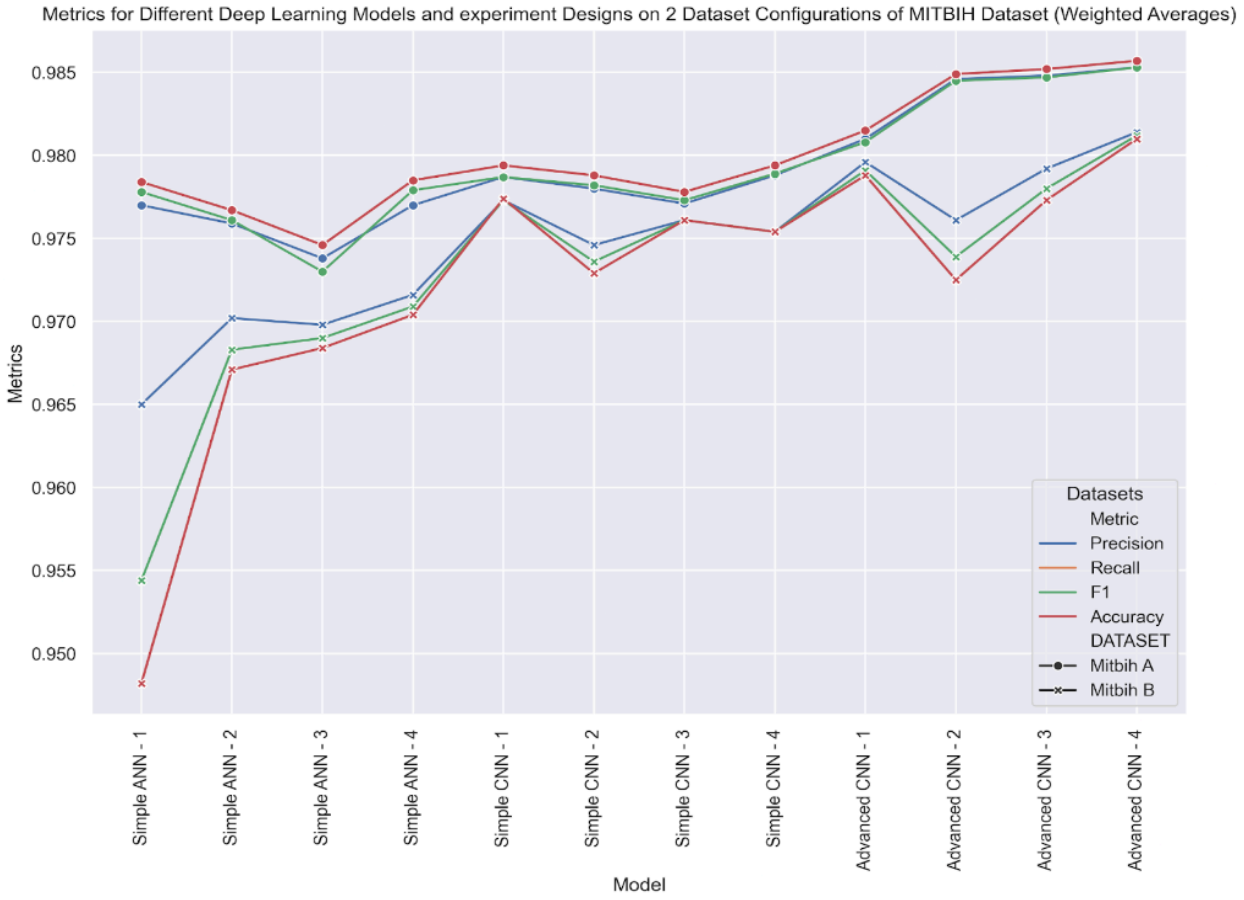



Fig.18: DL Models results for MITBIH

On the left-hand side of Fig. 18, you can see ANN performance metrics, while on the right side, CNN metrics are displayed. Generally speaking, CNN produces better results. From the 1st iteration to the 3rd, the initial learning rate decreases and results generally improve. In the 4th iteration, ANN performs the best as the learning rate is halved every 10 seconds. However, in iteration-4, performance is not consistently the best. We found that if the default iteration experiment finds the model early (i.e., within 30 of 70 possible epochs), further decreasing the learning rate is not efficient and may even worsen the initial results. Conversely, if the model is

found later in the default experiment, decreasing the learning rate is efficient, and we generally observe the best results with iteration-4, where the learning rate was halved after 10 epochs.

In terms of ANN, an examination of the original dataset reveals a performance decline during the second and third experiment, followed by performance stabilization in the fourth experiment. Conversely, the oversampled dataset demonstrates a consistent surge in performance accuracy across all experiments, before reaching the highest performance metric in the final experiment.

Regarding CNN, experimentation appears to result in a slight performance decline for both the original and oversampled datasets. Although the metric stabilizes in the fourth experiment, there is no significant improvement from the first experiment.

Advanced CNN exhibits leading metric performance from the first experiment, with further improvements observed in the second and third experiments, ultimately reaching peak accuracy performance. In summary, among all deep-learning models, advanced CNN demonstrates the highest performance metrics.

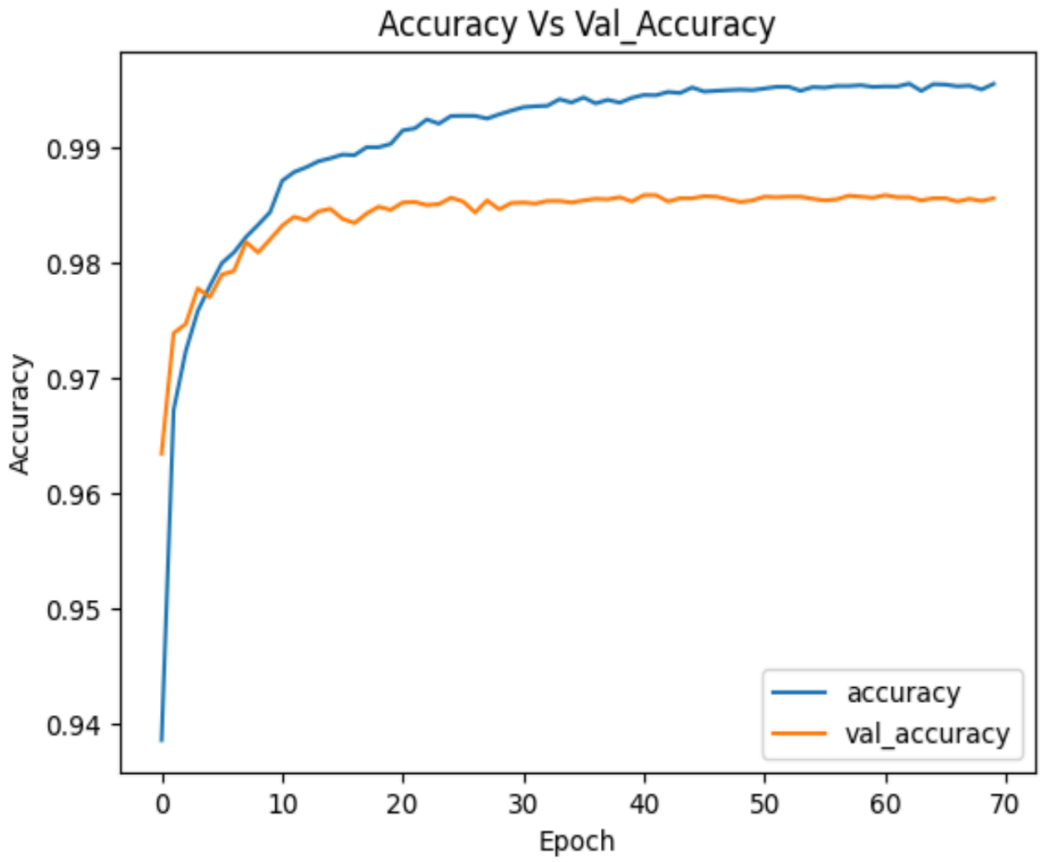



Fig. 19. Train accuracy vs. validation accuracy for the Advanced CNN model(as the best performing model on MITBIH test set), obtained with experiment 4 and MITBIH A dataset.

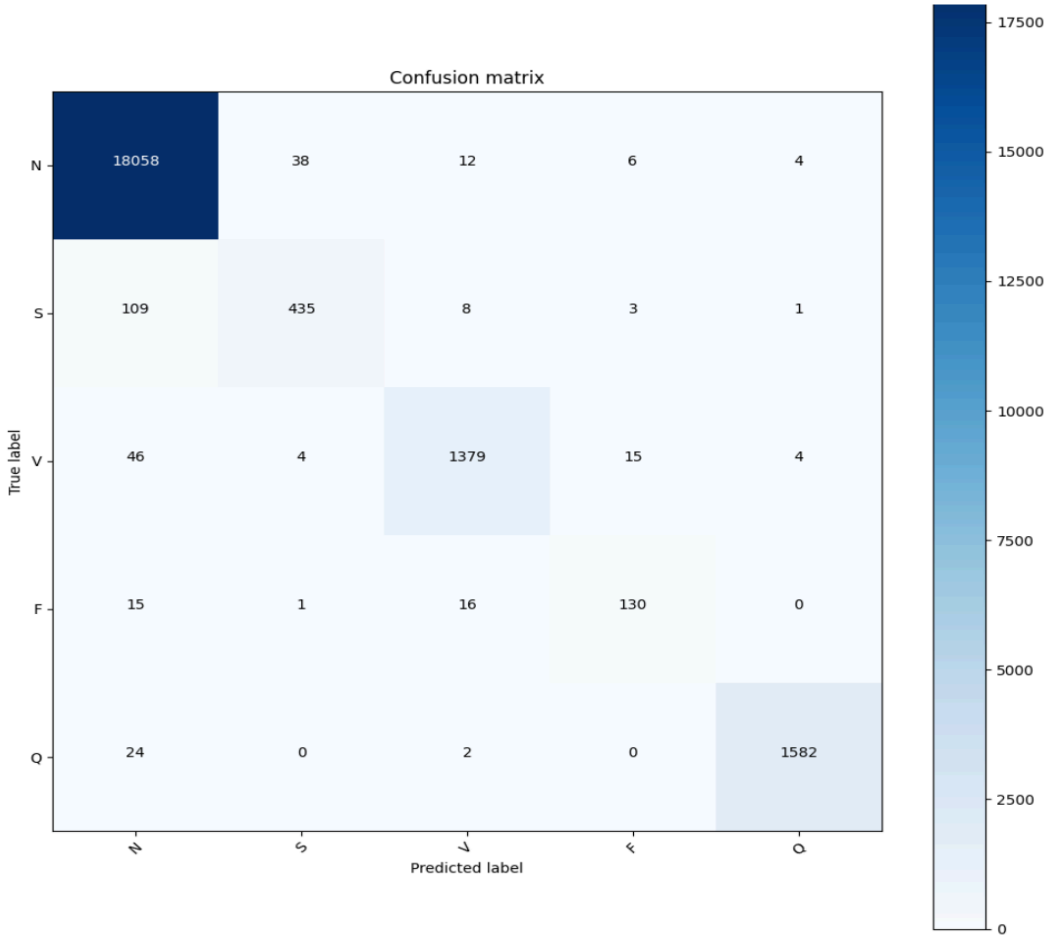



Fig. 20: Confusion matrix for MITBIH test set, obtained with Advanced-CNN model (exp. 4)

In addition to the author's performance comparison, our findings in machine learning (ML) versus deep learning (DL) reveal more trends. XGBoost emerged as the leading performer in ML, achieving an accuracy of 98.20% on the original dataset, while Random Forest outperformed other ML models with an accuracy of 98.11% on the oversampled MIT-BIH dataset. Among all traditional machine learning models, the best-performing models tend to be Random Forest and XGB. This could be attributed to their nature of fitting the data almost exclusively with sufficient depth.

Both ML and DL models demonstrate remarkable efficacy. It is worth noting that even with a simple model, the obtained accuracy remains relatively high. Consequently, the comparison of ML versus DL performance metrics becomes increasingly hard to compare due 1) high performance metric across all models and 2) slight changes in accuracy improvement for fine-tunes models.

PTBDB:

Table 6 displays model results for all experiments conducted for Simple ANN/CNN and Advanced CNN of PTBDB Dataset. The best scoring modes are highlighted in bold font based on the **Accuracy score** of each experiment.

Table 6: PTBDB results with DL models

| Method | Precision | Recall | F1 | Accuracy | DATASET |
|---|---|---|---|---|---|
| **SIMPLE ANN - 1** | 0.9654 | 0.9643 | 0.9646 | 0.9643 | PTBDB A |
| **SIMPLE ANN - 2** | 0.9445 | 0.9437 | 0.944 | 0.9437 | PTBDB A |
| **SIMPLE ANN - 3** | 0.9593 | 0.9588 | 0.959 | 0.9588 | PTBDB A |
| **SIMPLE ANN - 4** | 0.9667 | 0.9667 | 0.9667 | 0.9667 | PTBDB A |
| **SIMPLE ANN - 1** | 0.9725 | 0.9725 | 0.9725 | 0.9725 | PTBDB B |
| **SIMPLE ANN - 2** | 0.9566 | 0.9564 | 0.9565 | 0.9564 | PTBDB B |
| **SIMPLE ANN - 3** | 0.9573 | 0.9574 | 0.9574 | 0.9574 | PTBDB B |
| **SIMPLE ANN - 4** | 0.9663 | 0.966 | 0.9661 | 0.966 | PTBDB B |
| **SIMPLE CNN - 1** | 0.9775 | 0.9773 | 0.9774 | 0.9773 | PTBDB A |
| **SIMPLE CNN - 2** | 0.9804 | 0.9804 | 0.9804 | 0.9804 | PTBDB A |
| **SIMPLE CNN - 3** | 0.9783 | 0.9784 | 0.9783 | 0.9784 | PTBDB A |
| **SIMPLE CNN - 4** | **0.9842** | **0.9842** | **0.9842** | **0.9842** | PTBDB A |
| **SIMPLE CNN - 1** | 0.9853 | 0.9852 | 0.9852 | 0.9852 | PTBDB B |
| **SIMPLE CNN - 2** | 0.9785 | 0.9784 | 0.9784 | 0.9784 | PTBDB B |
| **SIMPLE CNN - 3** | 0.9832 | 0.9832 | 0.9832 | 0.9832 | PTBDB B |
| **SIMPLE CNN - 4** | **0.9856** | **0.9856** | **0.9856** | **0.9856** | PTBDB B |
| **ADVANCED CNN - 1** | 0.9904 | 0.9904 | 0.9904 | 0.9904 | PTBDB A |
| **ADVANCED CNN - 2** | 0.9845 | 0.9842 | 0.9841 | 0.9842 | PTBDB A |
| **ADVANCED CNN - 3** | 0.9884 | 0.9883 | 0.9883 | 0.9883 | PTBDB A |
| **ADVANCED CNN - 4** | **0.9938** | **0.9938** | **0.9938** | **0.9938** | PTBDB A |
| **ADVANCED CNN - 1** | 0.9921 | 0.9921 | 0.9921 | 0.9921 | PTBDB B |
| **ADVANCED CNN - 2** | 0.9873 | 0.9873 | 0.9873 | 0.9873 | PTBDB B |
| **ADVANCED CNN - 3** | 0.9907 | 0.9907 | 0.9907 | 0.9907 | PTBDB B |
| **ADVANCED CNN - 4** | **0.9959** | **0.9959** | **0.9959** | **0.9959** | PTBDB B |


Fig. 21: DL results for PTBDB dataset.

On the left-hand side of Fig. 21, ANN performance metrics are depicted, while CNN metrics are displayed on the right side. CNN exhibits far better performance compared to ANN. In Experiment-1, better performance is evident compared to the 2nd and 3rd iterations. However, reducing the initial learning rate and running epochs does not significantly contribute to success in smaller datasets.

In Experiment 4, we observe a halving of the learning rate every 10 epochs, resulting in a resurgence in performance akin to that seen in the original Experiment 1. It is probable that further tuning and adaptation could potentially exceed default metrics. When early stopback is invoked later in the fitting stage, adaptation (reduction) of the learning rate appears to enhance performance, as evidenced in Experiment 4.

During the first experiment, Advanced CNN delivers phenomenal improvement; superseding performance on both, simple CNN and simple ANN. Accuracy of 99.38% is achieved for Experiment 4-Original dataset, whilst accuracy of 99.59% is achieved for oversampled dataset. We can conclude that advanced CNN has resulted in best performance across PTBDB and MITBIH dataset. Fig. 22 shows history of the model train- and validation accuracy during training.

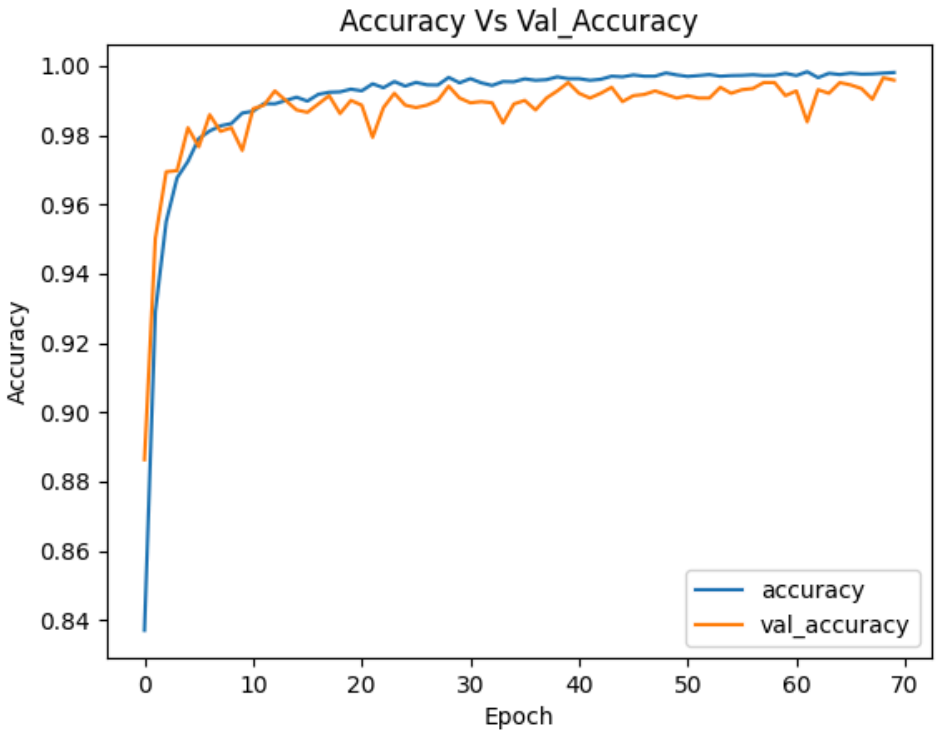



Fig. 22: Train accuracy vs. validation accuracy for the Advanced CNN model (as the best performing model on PTBDB test set), obtained with experiment 4 and PTBDB B dataset.

Comparing XGB performance with CNN/ANN, we can see that both, simple and advanced CNN has outperformed XGB. XGB accuracy for the original dataset stood at 97.9%, whereas advanced CNN-4 is at 99.38%. Similarly, XGB for an oversampled dataset stood at 97.63%, whereas advanced CNN-4 is at 99.59%. The most favorable performance for the PTBDB dataset was achieved by the Advanced CNN model utilizing the parameters outlined in Experiment 4. Advanced CNN model produced phenomenal results on both original and oversampled PTBDB dataset.

**Compare the results obtained to the benchmark**

Comparing the original author's paper results with our own, the author's deep residual CNN achieved an average accuracy of 93.5% on the MIT-BIH dataset. Our advanced CNN-4 model resulted in an accuracy of 98.57% on the original MIT-BIH dataset, while the advanced CNN-4 model trained on the oversampled dataset attained an accuracy of 98.1%.

Regarding PTDBD, we are unable to compare performance due to differences in our approach. The original authors [ref. 2] used output activations of the very last convolution layer of MIT-BIH as a representation of input for heartbeats. In contrast, we applied a deep-learning model separately on PTBD and MITBIH. Despite this, our machine learning results for PTBD reached

an accuracy of 99.38% for Experiment 4-Original dataset, while an accuracy of 99.59% was achieved for the oversampled dataset.

Table. 7: Comparisons of the results with the benchmark methods from the literature

| Algorithm Type | Source | Accuracy Score | Testset |
|---|---|---|---|
| ANN | Ref. [3] | 0.94 | MITBIH |
| XGB | Ref. [4] | 0.96 | MITBIH |
| ANN | Ref. [5] | 0.982 | MITBIH |
| CNN-LSTM | Ref. [6] | 0.9734 | MITBIH |
| CNN | Ref. [7] | 0.94 | MITBIH |
| TRANSFORMER | Ref. [7] | 0.72 | MITBIH |
| CNN | Ref. [8] | 0.72 | MITBIH |
| CNN | Ref. [9] | 0.769 | PTBDB |
| CNN-Autoencoder | Ref. [10] | 0.95 | MITBIH |
| CNN-ATTENTION | Ref. [11] | 0.9925 | MITBIH |
| CNN | Current Work | 0.9857 | MITBIH |
| CNN | Current Work | 0.996 | PTBDB |

# Conclusion

**For each of the project's goals, detail how they were achieved or not.**

Our primary objective was to transition from manual labeling of heartbeat types to an automated process. Throughout this project, we showcased the capacity of machine learning for task automation within the medical domain.

Our secondary objective was focused around diagnostic accuracy. The Results Section illustrates the outcomes achieved through deep learning machine learning techniques. Furthermore, we conducted comparative analyses against other authors' methodologies to demonstrate the competitive performance of our model. In conclusion, our deep-learning model attained an accuracy rate of 99%.

Our third focus was on utilization of deep learning techniques. We conducted experiments employing ANN and CNN models for this project, showcasing exceptional performance results in deep learning methodologies.

Our final objective centered on performance consistency. When utilizing machine learning models, we employed GridSearch with cross-validation. Cross-validation allowed us to measure performance consistency by repeating the process and averaging results. This approach helped us address the overfitting problem. In terms of deep learning, we utilized a validation set for each epoch. By observing the validation performance over time, we were able to monitor the model's generalization ability.

**If they have been reached, in which process(es) can your model fit? Detail.**

This project can be integrated into an automated classification system for heartbeat arrhythmia. This would involve creating a model in an API that could be utilized within medical workflows or even embedded in medical devices. Given the potential for application across the entire medical domain, maintaining precision and recall metrics is crucial to ensuring the consistency of machine learning. This can be accomplished through random sampling and manual review of machine learning classifications across different classes.

**What avenues for improvement do you suggest to increase the performance of your model?**

To increase the performance of our model, we suggest the following avenues for improvement:

1. Identifying parameters that contribute to success: It has been observed that incorporating more convolutional layers leads to better results.
2. Sharing the model for use on different datasets to assert performance consistency.
3. While we did not implement preprocessing steps, it is essential to thoroughly examine the preprocessing procedures required.
4. Utilizing special data augmentation techniques as mentioned in reference 11.
5. Notably, an initial trend indicates that with improved CNN models, not only does accuracy increase, but there is also an improvement in recall for the diseased classes, which is a desirable outcome. Despite this, further research is necessary to understand and propagate this model's behavior

**How has your project contributed to an increase in scientific knowledge?**

Due to the presence of good data,including but not limited to large sample size and prior-pre-processing, performance was already very good regardless of the model. In real life, it is challenging to get such "perfect" data with such solid results. As a result of high precision metrics across all the models, it was difficult to pinpoint significant performance improvements that would signify a major advancement in scientific knowledge.

We also experienced lower precision for class 3 and 5. This was related to the small sample size of these classes. If more data were available for these classes, our dataset would be more reliable and could potentially lead to an increase in performance across all classes. Specifically, acquiring more data for abnormal classes (rather than normal heartbeats) would be beneficial.

As non-professionals in the medical domain, our understanding of the success of this model may be limited to the machine learning domain. Despite good ML performance, medical professionals may still choose to review the ECG diagrams themselves. There might be many reasons why medical professionals may deem medical automation techniques as concerning. Some reasons include:

1) Possible concerns around the algorithms' trustworthiness or consistency.
2) Abnormal classes not reaching 100% accuracy might lead to legal and life threatening risks
3) Specific legal requirements established for achieving a certain metric performance as a minimal threshold requirement. These might fall outside of currently established machine model performance.

# Bibliography

---

**References**


1. S. Fazeli, ECG Heartbeat Categorization Dataset, https://www.kaggle.com/datasets/shayanfazeli/heartbeat. Last accessed on 16.03.2024.
2. M. Kachuee, S. Fazeli, and M. Sarrafzadeh, ECG Heartbeat Classification: A Deep Transferable Representation, https://arxiv.org/pdf/1805.00794.pdf. Last accessed on 16.03.2024.
3. PhysioNet PTBDB ANN, https://github.com/anandprems/physionet_ptbdb_ann. Last accessed on 15.03.2024.
4. Mahfuj Hossain, Kaggle user. XGB classification algorithm. https://www.kaggle.com/code/mahfujhossain/ecg-xgb-unbalanced
5. Mahfuj Hossain, Kaggle user. DNN classification algorithm. https://www.kaggle.com/code/mahfujhossain/ecg-imbalanced-dataset-mlp
6. Kanhaiyachatla, Kaggle user. CNN - LSTM classification algorithm. https://www.kaggle.com/code/kanhaiyachatla/ecg-classification-using-cnn-lstm
7. Erfan Saeedi, Kaggle user. CNN and Transformer classification algorithms. https://www.kaggle.com/code/erfansaeedi/cnn-and-transformer#CNN
8. Gregoire DC, Kaggle user. CNN algorithm. https://www.kaggle.com/code/gregoiredc/arrhythmia-on-ecg-classification-using-cnn
9. Zakari, Kaggle user. CNN Autoencoder algorithm. https://www.kaggle.com/code/redpen12/anomaly-detection-with-cnn-autoencoders#Building-CNN-Autoencoder-Model
10. Nicolas MINE, Kaggle user. CNN algorithm with data augmentation. https://www.kaggle.com/code/coni57/model-from-arxiv-1805-00794
11. Marco Polo, Kaggle user. CNN - LSTM - Attention algorithm. https://www.kaggle.com/code/polomarco/ecg-classification-cnn-lstm-attention-mechanism

# Appendices

**Gantt diagram.**


**Description of code files.**

- Git links
- Ran on kaggle, adapted to be run independently on github (see readme.md in github and comments in the notebooks)