CE-ARR - Machine Learning Project on ECG Classification for Arrhythmia Detection

Group 30: Victor PAUL-DUBOIS-TAINE 6062245 — Tony RAFFOUL 6065856

TU Delft - Quarter 1 - 2023

Summary

In this machine learning project, we were requested to develop a model for the detection of cardiac arrhythmias diseases using ElectroCardioGram (ECG) data. We began by preprocessing the ECG data, where we converted the data into a suitable format and extracted essential features representing heartbeats. The data was split into training, validation, and test sets to train and evaluate our models. Then, we utilized several machine learning models, including deep neural networks (MLP, KNN, SVM, Logistic Regression), to classify the preprocessed ECG data into AAMI classes. Based on our results, we were able to compare and select the most suited suited for this use. We can say that we successfully developped strategies for detection of abnormal hearbeats and utilized the knowledge acquired from previous CodeLabs and Lectures given throughout this quarter.

ML Pipeline

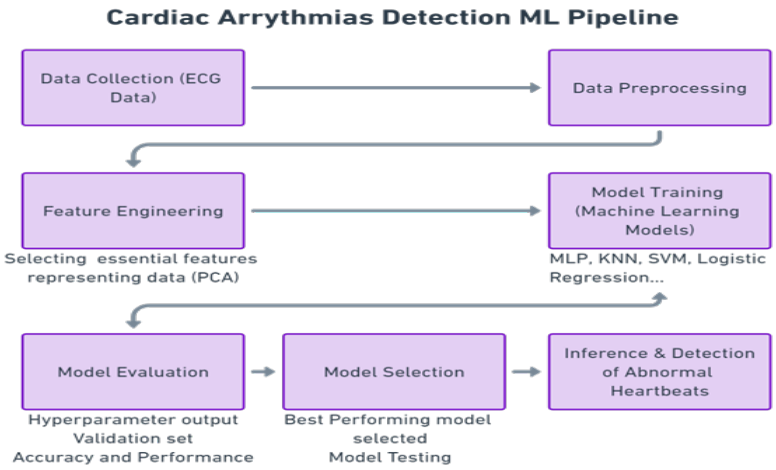


Figure 1: ML Pipeline Flowchart

The process of detecting cardiac arrhythmias using ECG data is a multi-step procedure, as visualized in our flowchart we chose this typical workflow:

1. **Data Collection**: The initial step involves gathering ECG data, which serves as the primary input for our analysis.
2. **Data Preprocessing**: Before any analysis can be conducted, the raw ECG data is subjected to preprocessing. This step ensures that the data is in a suitable format, free from noise or any inconsistencies. This step was already done, we data that was given was preprocessed and splitted in three sub-sets : Training / Validation / Testing. We adopted the same preprocessed data and splits for sake of simplicity
3. **Feature Engineering**: At this stage, essential features that represent heartbeats are extracted. These features play a crucial role in the accuracy of the subsequent classification process. We applied PCA (Principal Component Analysis) and scalling. We selected the first 10 dimensions resulting from the PCA transformation. PCA was trained on the training set and applied to all the different sets.
4. **Model Training**: We trained multiple machine learning models, including MLP, KNN, SVM, and Logistic Regression. The models are trained on the training dataset. Each model aims to classify the ECG data into AAMI classes effectively.
5. **Model Evaluation**: Post-training, each model's performance is assessed using the validation set. This step helps in determining the accuracy and overall effectiveness of each model in detecting arrhythmias. Based on results obtained from this step we were able to fine tune our hyperparameter selection and train the model again with optimized hyperparameters.
6. **Model Selection**: Based on the evaluation results, the best-performing model is chosen. This model is expected to provide the highest accuracy in real-world scenarios.
7. **Inference**: The final step involves using the selected model on the test set. This provides a final evaluation and aids in the detection of abnormal heartbeats.

In conclusion, this pipeline provides a systematic approach to cardiac arrhythmias detection using ECG data, ensuring that the model developed is both accurate and reliable.

Task 1: Data Pre-processing

As mentioned in the document `CE_ARR_Additional_information.pdf`, the pre-processing and splits were provided in the files `train_data.csv`, `val_data.csv`, and `test_data.csv`.

Upon receiving the ECG data, our first step was to import and visualize it. The data was loaded from the CSV file `train_data.csv`, and was visualized to understand its structure and characteristics.

Key steps in data preprocessing:

1. **Data Import:** We loaded the ECG data using the `numpy` library into a numpy array and visualized the first heartbeat using `matplotlib`.

```
train_data = np.loadtxt("train_data.csv", delimiter=",", dtype=float)
plt.figure(figsize = (12, 6))
plt.plot(train_data[0, :250])
plt.title("First Heartbeat of the data")
```

2. **Separate data/label:** As labels corresponded to final column, we had to separate them from the rest of the data.

3. **Data normalization:** We utilized the `StandardScaler` from `sklearn` to normalize the data, ensuring that our models can converge faster and perform better. We also utilized PCA on the training data. We trained each of our model with these 3 types of data: Raw data - Scaled Data - PCA Data We chose the first 10 dimensions of PCA: they cover more than 95% of all the explained variance. Both scaler and PCA have been trained with the training data. For instance, the standard scaler needs two parameters, the mean and the standard deviation to scale new data. Here, the mean and std used are those of the training data. The same principle applies to the PCA transformations.

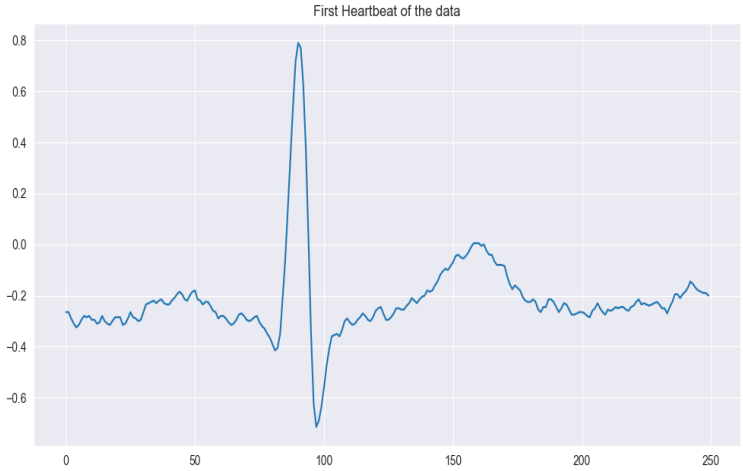


Figure 2: First Heartbeat of Training set

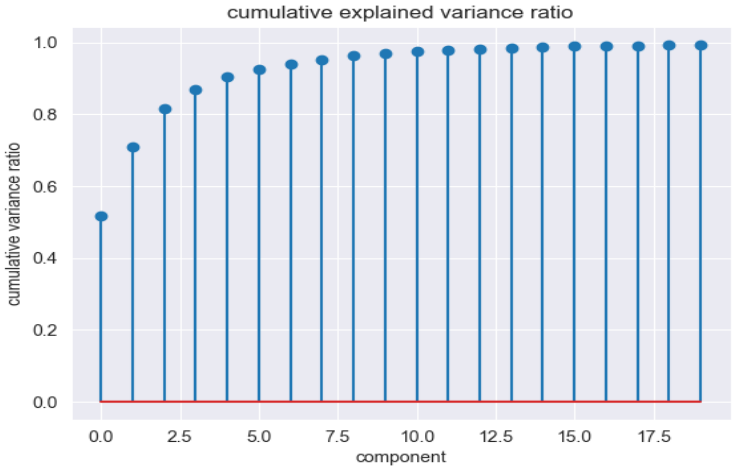


Figure 3: Cumulative Explained Variance in function of number of PCA Components

In practice

In practice in industry, the pre-processing data isn't received that easily, data from the ECG need to go through a lot of steps, these steps are imperative for further analysis, ensuring the accuracy and reliability of the information obtained from the ECG signals. Here are some of the principals steps[1]:

1. **Noise Reduction:** ECG signals can be contaminated by various types of noise like muscle activity, electrical interference... Used techniques include bandpass filters and notch filters
2. **Segmentation:** ECG signals are divided into segment, each segment typically contains one heart beat
3. **Normalization:** The newly segmented data is normalized to ensure that the ECG signals are within a standard rand, which is crucial for comparison and analysis
4. **Baseline Correction:** Baseline removal is performed in order to remove linear trend of ECG signal
5. **Peak Detection:** Peak detection is crucial to identify the most significant points within an ECG signal, used for further analysis
6. **Filtering:** A moving average filter is used to smooth out the signal further removing the low-frequency noise present in the ECG signal [2]

Another step in the preprocessing process worth mentioning is the **Personalization**. In ECG signal preprocessing can enhance the accuracy of analysis by tailoring steps to individual characteristics. For instance, personalized noise reduction algorithms can filter out noise based on unique characteristics of an individual's ECG signals. Moreover, machine learning algorithms like personalized LSTM models or federated learning methods can be employed to create models that adapt to individualized features of a patient's ECG signals, improving diagnostic accuracy and monitoring efficacy. Through such personalized fine-tuning, a more accurate and actionable interpretation of ECG data can be achieved. [3]

Task 2: Classification

For the classification task, we employed various machine learning models, with an emphasis on deep neural networks.

Data Split

As mentioned before, the data was provided in 3 sets, following a 60%-20%-20% split respectively training/validation/test.

Model Training

Our approach towards detecting cardiac arrhythmia encompassed a variety of machine learning methods, each with its own set of hyper-parameters and configurations. The intention behind using multiple methods was to explore a diverse solution space and ensure the best possible model is identified for the given problem.

- **Logistic Regression:** As a starting point, we employed Logistic Regression using 6 distinct parameter configurations. Despite its simplicity and efficiency, the results were not as promising as we had hoped for our specific data-set.

- **Support Vector Machines (SVM):** Transitioning to a more complex model, SVMs were trained with an extensive hyper-parameter tuning process. We fine-tuned and evaluated 12 different SVM models, each with its own set of configurations, to identify the most optimal one for our data.
- **K-Nearest Neighbors (KNN):** The non-parametric nature of KNN made it an interesting choice. We tested 9 different KNN models, each differing in terms of the number of neighbors and the processing of the data used (normal, scaled, or PCA).
- **Multi-Layer Perceptron (MLP):** Diving deep into neural networks, we utilized the MLP architecture. With 54 different parameter combinations, our experiments were exhaustive. This model, among all, yielded the highest performance metrics given its Deep Neural Network (DNN) architecture. We first selected the activation (logistic, tanh or relu) that worked the best with one to five layers. We realized that the best models used relu activation for 1 and 4 layers, and a tanh activation for 3 layers. We then for each of the three models tuned the sizes of the layers. For the 1 layered MLP, we tried 5 different sizes. For the 3 layered MLP, we did a grid search with 3 different sizes : 50, 100 and 150. We tested each combination of those three sizes. Then, for the 4 layered MLP, we also did a grid search, but with 2 different sizes, since more sizes would have taken too long to compute. We used the layer sizes 50 and 150. All of the above tuning has been done with the normal data. We then selected the

Expected Performance:

Model	Strengths	Weaknesses / Considerations
Logistic Regression	- Effective for datasets with clear linear boundaries between arrhythmia types	- Might underperform if arrhythmia types have complex, non-linear boundaries in feature space
SVM	- Versatile for different arrhythmia patterns due to the kernel - Effective in high-dimensional ECG feature spaces	- Requires careful selection of kernel for specific ECG patterns - Training might be slow if using high-resolution ECG segments
KNN	- Can capture localized patterns in ECG data, useful for rare arrhythmias	- Real-time classification might be slow, - Performance can degrade if irrelevant ECG features are included
MLP	- Deep Neural Network - Capable of modeling complicated ECG patterns and relationships - Flexibility in architecture allows for modeling of different arrhythmia complexities	- Risk of overfitting to noise in ECG recordings without proper regularization - Training requires substantial data and can be computationally intensive

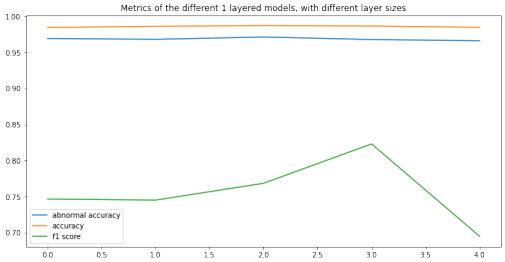
Table 1: Performance Expectations of Models in ECG Arrhythmia Classification Context

Model Evaluation

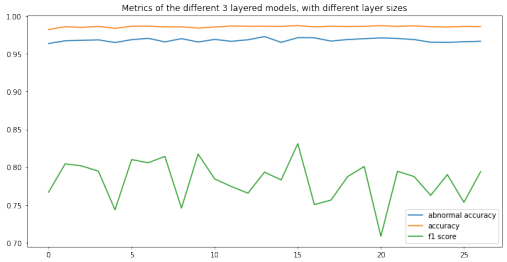
During the Model Evaluation phase, our main objective was to compare the performances of the various machine learning models that we trained. Using the training set (60% of the data) we trained MLP, KNN, SVM and Logistic Regression models. We evaluated the performance of these models using the validation set. This allowed us to fine-tune hyperparamters and select the best-performing for each method.

Performance on Validation Set

The graphs below illustrate the performance metrics of the different models on the validation set, each number on the X axis represent a model with different hyperparameter.

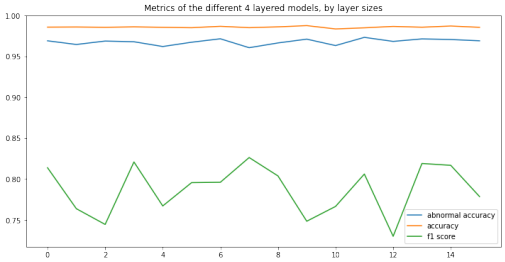


(i) MLP_metrics_1layer

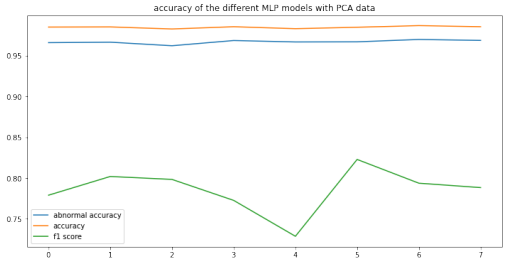


(ii) MLP_metrics_3layer

Figure 4: MLP Model

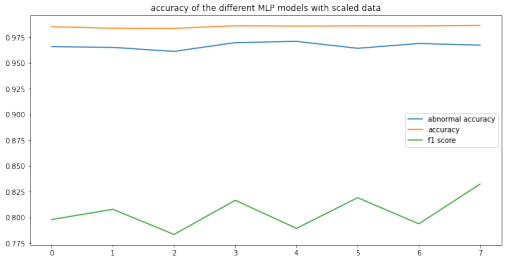


(iii) MLP_metrics_4layer

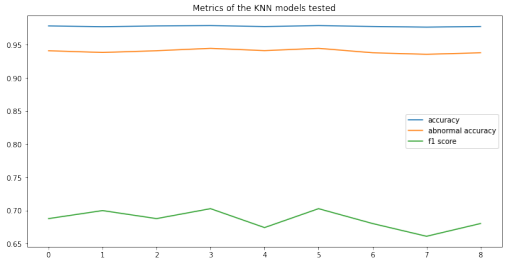


(iv) MLP_metrics_PCA

Figure 5: MLP Model



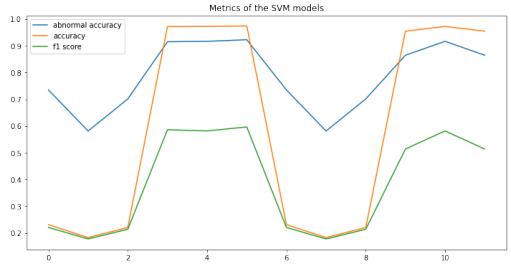
(v) MLP_metrics_scaled



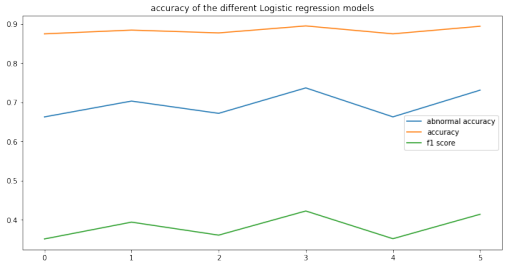
(vi) KNN_metrics

Figure 6: MLP Model & KNN Model

For each model type, we observed the F1 score, accuracy score, and abnormal accuracy. The abnormal accuracy is a metric we implemented ourselves. The aim is to exclude the normal heartbeats, which represents 68 % of the data. The abnormal accuracy is thus the sum of all the correctly predicted abnormal labels, divided by the total number of abnormal labels. While most models exhibited high accuracy, our decision to



(vii) SVM_metrics



(viii) LR_metrics

Figure 7: SVM Model & LR Model

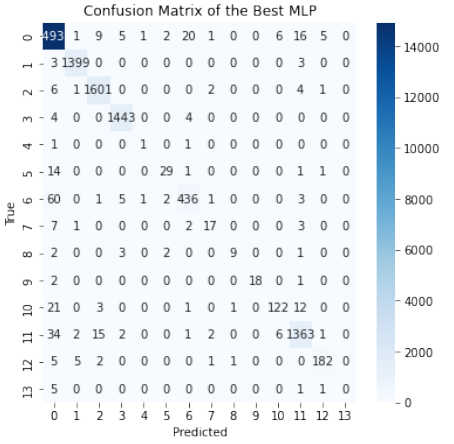
select the best model was primarily influenced by the F1 score, the metric that fluctuated the most. The F1 score, being the harmonic mean of precision and recall, provided a more balanced measure, especially crucial for medical diagnosis tasks where both false positives and false negatives can have significant implications.

Reflection of F1 Score

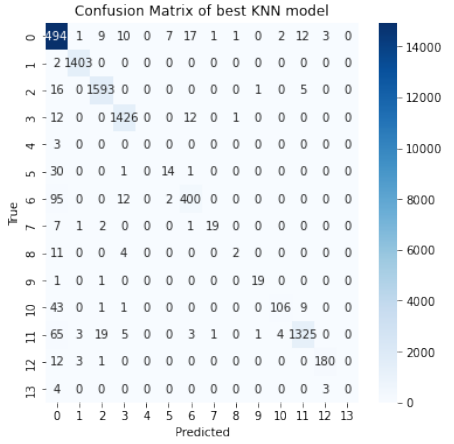
In our experiments, the datasets used for training, testing, and validation isn't evenly distributed. A significant portion of the data (approximately two-thirds) consists of regular heartbeats, while irregular ones are less frequent.For example class 4, appears 10 times in our training set and only 3 times during testing. Given the critical nature of medical scenarios, it's crucial for our algorithm to correctly identify all irregular heartbeats because each misclassification could have serious implications. Hence, we opted for the macro-average f1 score as our metric. This approach calculates the f1 score for every class and then determines an unweighted average for these scores. The result is that each category equally influences the final f1 score, allowing us to reduce the weight of regular heartbeats. However, a side effect is that it overemphasizes the rarer classes. This emphasis might be a reason our f1 score isn't as high as we'd like; ideally, we're aiming for a score close to 0.9.

Performance on Test Set

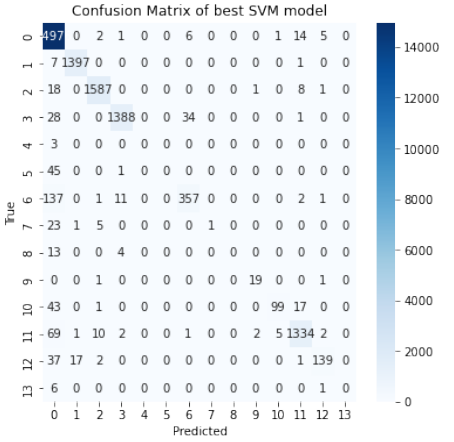
After identifying the top performing model for each method using the validation set. We wanted to evaluate these different models using the Test set. You can find below the confusion matrices for the each method best performing model.



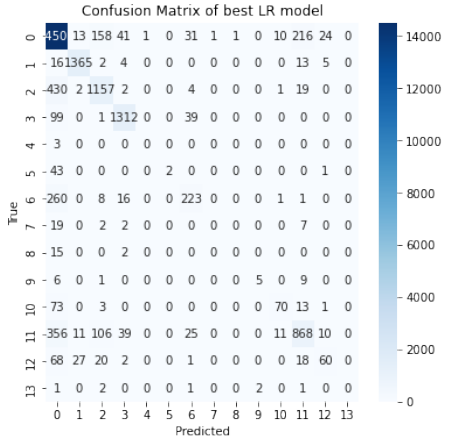
(i) MLP Confusion Matrix



(ii) KNN Confusion Matrix



(iii) SVM Confusion Matrix



(iv) LR Confusion Matrix

Figure 8: Confusion Matrices of top performing models

Furthermore, to provide a clear comparison of the F1 scores of the best-performing models, the bar chart below provides the results:

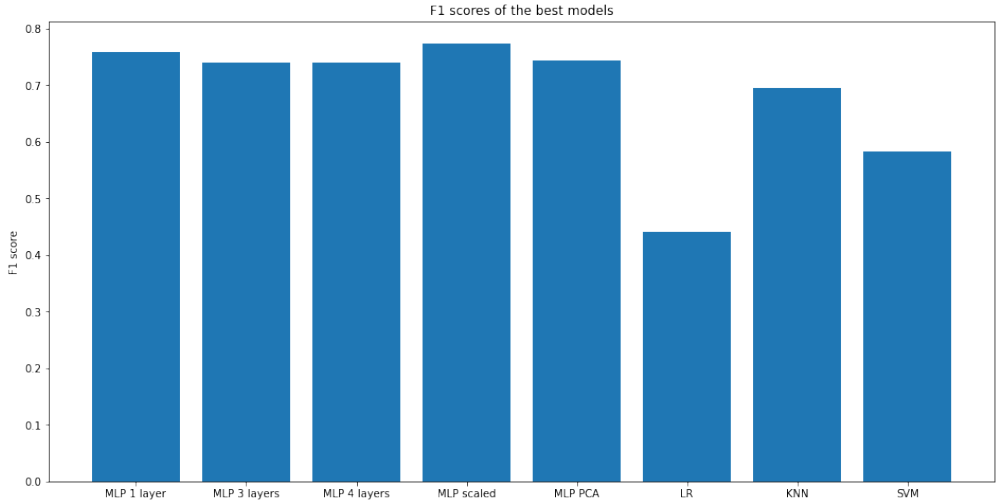


Figure 9: Bar Plot - F1 scores of top performing models

Following our chart comparison, we can select the two models `MLP_1_layer` and `MLP_scaled`. They were very close in terms of performance, with close F1 scores. But to pick the final model we felt the need to look into different factors. We decided to compare other metrics.
Here's a comparative table

Metric	`MLP_1Layer`	`MLP_Scaled`
F1 Score	0.758	0.774
Accuracy (%)	98.47	98.48
Hidden Layer Structure	(200)	(150 x 150 x 50 x 150)
Number of Parameters	53215	70865
Training Time	1min 23sec	2min 9sec

Table 2: Comparison between MLP_1 Layer and MLP_Scaled

From the table it is clear the `MLP_Scaled` has a slightly higher F1 score and accuracy, but `MLP_1Layer` has fewer parameters and faster training time. We can make a case of a trade-off, while having a higher F1 score and accuracy, the efficiency of `MLP_1Layer` in terms of running time and complexity cannot be ignored.

Delving deeper into the Trade-offs

In the world of Machine Learning, and especially in medical diagnosis, in application that have real-world consequences there is not really a one-size-fits-all answers. When comparing `MLP_1layer` and `MLP_scaled` it is not only about which model has the highest F1 score or accuracy but also about scalability, efficiency and practicality. [4]

1. **F1 score & Accuracy:** While F1 score of `MLP_Scaled` are higher, the difference is pretty minimal. In real world application, this small difference wouldn't impact the outcomes that much.
2. **Model Complexity:** Defined by its of parameters, a model's complexity can affect various aspects of its deployment. A model with less parameters typically requires less memory. This can be a significant

advantage when deploying in memory-constrained environments.

3. **Training Time:** When models need to be retrained frequently or time sensitive decisions are required, training time plays a crucial role. `MLP_1Layer` has clear advantage training about 35% faster than `MLP_Scaled`

Motivation behind final choice

- **Scalability:** As the dataset grows or as the model is implemented in real-time scenarios. The efficiency of `MLP_scaled` will become more important. Faster training time will lead to quicker iterations, allowing for more flexible responses to changes in the environment or new data
- **Efficiency in Deployment:** With fewer parameters, `MLP_1Layer`, might offer a better performance in deployment, especially where computer resources are limited.
- **Balanced Performance:** While F1 score and Accuracy are very important metrics, it is always important to consider the 'cost' of this accuracy. If a simpler model offer nearly same accuracy with better efficiency, it makes sense to go with the 'not-so-best' option.
- **Safety Precautions:** In medical application like ECG arrhythmia classification, false negatives can have severe consequences. The model `MLP_scaled` is more robust in its predictions, minimizing the chances of classifications. It is crucial to prioritize patient safety over slight improvements in performance metrics. When examining the wrongful classification of normal cases as false negatives, `MLP_scaled` recorded 164 false negatives out of 15,005 data points, in contrast to the 169 false negatives observed with `MLP_1Layer`. This the difference is very slight, only a difference of 0.03% in accuracy. Both model are perfectly capable of differentiate normal to abnormal ECG

Final choice

In conclusion, while `MLP_Scaled` performed best, the overall benefits of `MLP_1Layer` make it the model of our choice. It represent a balanced trade-off between performance and efficiency, a crucial trait for real-world application where both factors are equally important.

Conclusions

In our goal to classify ECG data for arrhythmia detection, we looked deep into the complexities of data preprocessing, feature extraction and model selection. We trained and test a variety of models from Logistic Regression, SVM and KNN to Deep Neural Networks MLP.
We compared each models performance and fine-tuned hyper-parameters using the Validation set, using the Test set we finally were able to select the top performing model. The close contest between `MLP_1layer` and `MLP_scaled` highlighted the need to balance between performance and efficiency - practical factors play pivotal roles. Our final choice `MLP_1Layer` encapsulate this learning.
As we conclude this project has been a nourishing opportunity to put into practice our learning from the "EE4C12 : Machine Learning for Electrical Engineering" course

Appendix

References

[1] A. C. B. Garcia, M. Ferro, and J. C. R. Ribón. A statistical designing approach to matlab based functions for the ecg. 2023.

[2] T. Anbalagan, M. K. Nath, D. Vijayalakshmi, and A. Anbalagan. Analysis of various techniques for ecg signal in healthcare, past, present, and future. *Biomedical Engineering Advances*, 6(100089):100089, 2023.

[3] P. Shyam Kumar, M. Ramasamy, K. R. Kallur, P. Rai, and V. K. Varadan. Personalized lstm models for ecg lead transformations led to fewer diagnostic errors than generalized models: Deriving 12-lead ecg from lead ii, v2, and v6. *Sensors (Basel, Switzerland)*, 23(3):1389, 2023.

[4] Minjung Ryu, Hong-Linh Truong, and Matti Kannala. Understanding quality of analytics trade-offs in an end-to-end machine learning-based classification system for building information modeling. *Journal of Big Data*, 8, 02 2021.

Source code

https://github.com/VictorPDT/EE4C12_CE_ARR

https://www.overleaf.com/read/njmmtcphdymd#00165e

Contact

Victor Paul-Dubois-Taine V.G.Paul-Dubois-Taine@student.tudelft.nl

Tony Raffoul t.raffoul@student.tudelft.nl