Classification of Heartbeats Using Machine Learning

Thomas Couloud, Johan Ghrenassia, Nadir Benmounah et Julien Catanese

Résumé
This study explores the automatic classification of heartbeats from electrocardiogram (ECG) signals using machine learning techniques. We compare the classification performance of different models on the Kaggle datasets, which were obtained from the MIT-BIH and PTB databases. We compare classical machine learning models with deep learning models. Our results show that the performances vary depending on the different models and their respective optimizations. We conducted two types of optimizations : one on the models and their parameters, and the other on the input data of the model. We highlight the limitations of certain models and their tendency to overfitting. Finally, we demonstrate that models based on neural network architectures yield better results for the classification of heartbeats from ECG signals.

Table des matières

1 Introduction

Context and overall objective

The aim of this report is to present the work carried out by our team during a project to detect anomalies in the heartbeats of healthy human subjects or those suffering from heart diseases. Our team is made up of 4 Data Scientist learners, and is supervised by the business expert Adrien Moreau. The technical challenge of the project will be to teach a machine to categorize heartbeats as Normal or Abnormal. For this, we have used Electrocardiograms (ECGs) collected and annotated by cardiologists. The data is freely available on the Kaggle database (https ://www.kaggle.com/shayanfazeli/heartbeat), which includes two collections of cardiac signals : MIT-BIH Arrhythmia Dataset and The PTB Diagnostic ECG Database. Based solely on these ECG data, we will use tools of artificial intelligence, to teach a machine to determine which are the healthy heartbeats and which are the abnormal heartbeats. To do this we will test traditional machine learning models such as logistic regression, KNN, and decision tree, as well as newer models like CATboost. Finally, we will enter into a deep learning modeling process by implementing artificial neural networks capable of classifying ECGs. In the end, we will compare the performance of these different models, and we will attempt to provide an explanation for the operation of each model. This project is primarily for educational purposes, but can have relevance from a scientific and medical point of view, as cardiac dysfunctions are today among the leading causes of mortality worldwide. Better understanding and detecting anomalies in ECG signals is therefore crucial to help healthcare professionals quickly diagnose heart problems and provide effective treatment, to avoid myocardial infarction for example. From an economic and social point of view, the automated classification of heartbeats can reduce healthcare costs by allowing early diagnosis while avoiding diagnostic errors. It should be noted that even a failure rate of 1 percent for such an automated classification system potentially applied to hundreds of thousands of people, would therefore have an impact on the lives of several thousand people. Active research on this subject therefore remains crucial to constantly increase the performance level of artificial intelligence.

2 Visualization, Understanding and Data Manipulation

exploration of the 3 Kaggle, MIT and PTB databases

The first objective of our modeling project is to prepare a usable data set. This begins with identifying a reliable and accessible ECG data source. Then to thoroughly analyze their content, to properly verify their cleanliness and their properties. Identify the target variables and explanatory variables. And finally reorganize the data set to obtain a training set, a test set, and a validation set (which will be isolated and preserved for a potential demonstration of the generalization of our models on data that have never been "seen" by the models).

2.1 Exploration of databases

We have identified 3 ECG databases : PTB, MIT and Kaggle. The first two databases contain "raw" ECG data, that is continuous electrophysiological traces obtained directly from the recording electrodes. These data are very reliable but not exploitable in their original format. Many pre-processing steps would be necessary to make them modelable ECG data.

2.1.1 raw data from MIT and PTB databases

Below is a fraction of the dataframe of the data set from PTB that we tried to extract several original files (.dat) and that we transformed into a Python dataframe :

	Unnamed: 0	0	1	2	3	4	5	6	7	8	...	120005	120006	120007	120008	120009	120010	120011	age	sex	diagnose
0	0	0.1755	0.1610	0.1690	0.1865	0.1970	0.2160	0.2200	0.2165	0.2135	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	87.0	female	Reason for admission: n/a
1	0	0.0655	0.0725	0.0655	0.0600	0.0535	0.0560	0.0790	0.0950	0.1010	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	87.0	female	Reason for admission: n/a
2	0	0.3860	0.4110	0.4200	0.4295	0.3900	0.3605	0.3385	0.3425	0.3445	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	85.0	female	Reason for admission: Myocardial infarction
3	0	-0.8430	-0.8395	-0.8405	-0.8455	-0.8515	-0.8490	-0.8515	-0.8460	-0.8480	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	80.0	female	Reason for admission: Myocardial infarction
4	0	-0.0310	-0.0305	-0.0255	-0.0205	-0.0345	-0.0385	-0.0430	-0.0570	-0.0725	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	80.0	female	Reason for admission: Myocardial infarction

Figure 1-VS : PTB Dataframe

Using all available files, we were able to recreate a dataframe with 120015 columns and 549 rows, with one row per patient (per .dat file). The first column is the patient's identifier. The last 3 columns give access to the patient's age, sex and diagnosis (retrieved from .hea files). All the other 120011 columns correspond to continuous recording points in time and are associated with a value in mV. However, as the recording times of each patient are different, the length of each varies and some lines therefore contain NANs. These data therefore need to be processed. We operated in the same way to explore the MIT database :

	Unnamed: 0	0	1	2	3	4	5	6	7	8	...	649993	649994	649995	649996	649997	649998	649999	age	sex	drug
0	0	0.260	0.260	0.260	0.260	0.260	0.260	0.260	0.260	0.260	...	-0.065	-0.125	-0.220	-0.305	-0.370	-0.380	0.00	73	F	Digoxin, Nitropaste, Pronestyl
1	0	-0.065	-0.065	-0.065	-0.065	-0.065	-0.065	-0.065	-0.080	...	-0.370	-0.510	-0.480	-0.410	-0.365	-0.335	0.00	69	M	Aldomet, Inderal
2	0	0.210	0.210	0.210	0.210	0.210	0.210	0.210	0.210	0.210	...	0.025	0.015	0.000	0.015	0.015	0.035	0.00	76	F	Aldomet, Inderal
3	0	-0.470	-0.470	-0.470	-0.470	-0.470	-0.470	-0.470	-0.470	-0.455	...	-2.165	-2.085	-1.985	-1.905	-1.825	-1.740	-1.28	51	F	Pronestyl
4	0	-0.515	-0.515	-0.515	-0.515	-0.515	-0.515	-0.515	-0.515	-0.485	...	-1.210	-1.470	-1.750	-1.960	-2.100	-2.140	-2.56	69	M	Digoxin, Norpace
5	0	0.135	0.135	0.135	0.135	0.135	0.135	0.135	0.135	0.135	...	0.080	0.080	0.075	0.065	0.075	0.100	0.00	89	M	None
6	0	0.135	0.135	0.135	0.135	0.135	0.135	0.135	0.135	0.135	...	0.085	0.065	0.070	0.070	0.075	0.080	0.00	56	F	None
7	0	0.035	0.035	0.035	0.035	0.035	0.035	0.035	0.035	0.035	...	-0.370	-0.515	-0.635	-0.745	-0.790	-0.795	-1.28	47	F	Digoxin, Lasix
8	0	0.050	0.050	0.050	0.050	0.050	0.050	0.050	0.050	0.050	...	0.070	0.090	0.100	0.095	0.080	0.075	0.00	83	M	Hydrochlorthiazide, Lasix

Figure 2-VS : MIT Dataframe

This dataframe is composed of more than 650,000 columns and 48 lines. The categorical data here are age, sex and prescribed medication (last 3 columns). Thus, we currently have 2 dataframes of different dimensions not containing the same information about the patients. The signals do not have the same duration either, nor the same sampling frequencies. Therefore, the extraction of the ECG signals from these dataframes for a possible fusion of the data in a compatible format would require enormous work of pre-processing and organization of the data to be exploitable for our models.

2.1.2 The Kaggle database is the most exploitable

The Kaggle database contains data from the two other PTB and MIT databases described above. The big advantage of Kaggle is that the data has already undergone the pre-processing phases necessary for the use of the data for modeling. PTB NORMAL

	0	1	2	3	4	5	6	7	8	9	...	178	179	180	181	182	183	184	185	186	187
0	1.000000	0.900324	0.358590	0.051459	0.046596	0.126823	0.133306	0.119125	0.110616	0.113047	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	1.000000	0.794681	0.375387	0.116883	0.000000	0.171923	0.283859	0.293754	0.325912	0.345083	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.909029	0.791482	0.423169	0.186712	0.000000	0.007836	0.063032	0.077002	0.074957	0.077342	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	1.000000	0.478893	0.056760	0.064176	0.081289	0.072732	0.055619	0.048774	0.054478	0.041643	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	1.000000	0.867238	0.201360	0.099349	0.141336	0.120934	0.108516	0.096393	0.093436	0.100828	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

4046 rows × 188 columns

PTB ABNORMAL

	0	1	2	3	4	5	6	7	8	9	...	178	179	180	181	182	183	184	185	186	187
0	0.932233	0.869679	0.886186	0.929626	0.908775	0.933970	0.801043	0.749783	0.687229	0.635100	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
1	1.000000	0.606941	0.384181	0.254237	0.223567	0.276836	0.253430	0.184826	0.153349	0.121872	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
2	1.000000	0.951613	0.923963	0.853303	0.791859	0.734255	0.672043	0.685100	0.670507	0.667435	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
3	0.977819	0.899261	0.230129	0.032348	0.142329	0.223660	0.328096	0.367837	0.381701	0.389094	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
4	0.935618	0.801661	0.805815	1.000000	0.722741	0.480789	0.454829	0.319834	0.266874	0.308411	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0

10506 rows × 188 columns

MIT TRAIN

	0	1	2	3	4	5	6	7	8	9	...	178	179	180	181	182	183	184	185	186	187
0	0.977941	0.926471	0.681373	0.245098	0.154412	0.191176	0.151961	0.085784	0.058824	0.049020	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.960114	0.863248	0.461538	0.196581	0.094017	0.125356	0.099715	0.088319	0.074074	0.082621	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	1.000000	0.659459	0.186486	0.070270	0.070270	0.059459	0.056757	0.043243	0.054054	0.045946	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.925414	0.665746	0.541436	0.276243	0.196133	0.077348	0.071823	0.060773	0.066298	0.058011	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.967136	1.000000	0.830986	0.586854	0.356808	0.248826	0.145540	0.089202	0.117371	0.150235	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

87554 rows × 188 columns

MIT TEST

	0	1	2	3	4	5	6	7	8	9	...	178	179	180	181	182	183	184	185	186	187
0	1.000000	0.758264	0.111570	0.000000	0.080579	0.078512	0.066116	0.049587	0.047521	0.035124	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.908425	0.783883	0.531136	0.362637	0.366300	0.344322	0.333333	0.307692	0.296703	0.300366	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.730088	0.212389	0.000000	0.119469	0.101770	0.101770	0.110619	0.123894	0.115044	0.132743	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	1.000000	0.910417	0.681250	0.472917	0.229167	0.068750	0.000000	0.004167	0.014583	0.054167	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.570470	0.399329	0.238255	0.147651	0.000000	0.003356	0.040268	0.080537	0.070470	0.090604	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

21892 rows × 188 columns

Figure 3-VS : The Kaggle set is divided into two folders, each containing two datasets. One PTB folder with one dataset containing abnormal signals and another normal and another MIT folder with one Train and Test dataset both composed of 5 types of signals (one for a normal ECG and 4 others for anomalies).

As shown in the tables of figure 2-VS, each of these dataframes consists of the same number of columns (188), with the labels of the target variable contained in the last column. Moreover, these dataframes have been homogenized to the same sampling frequency (125Hz), meaning that all samples have a duration of 1.5 seconds. None contain missing values (no NaNs), and signals that are too short are padded with zeros so that each signal has the same number of points ("zeros padding"). The mV values have been normalized by a minimax function (min=0 and max=1). The explanatory variable is therefore composed of a series of 187 points forming a pattern of electrical waves representing a heart beat cycle over time, which we will call here "ECG signal" or "ECG beat". The details of the pre-processing phase of the Kaggle data are described in the article by Kachuee et al., 2018 (ref[1]).

Following this initial exploration, the Kaggle database seems to us to be best suited for the project as the data are clean and organized in a way conducive to modeling.

2.2 Visualization and understanding of Kaggle data

2.2.1 Explanatory variables

Let's start by visualizing our explanatory variables, otherwise known as the successive ECG points forming a cardiac beat cycle.

We note that the signals form distinct patterns and are difficult to understand at first glance. We will use an explanatory diagram to better understand the signals (figure 5-VS)

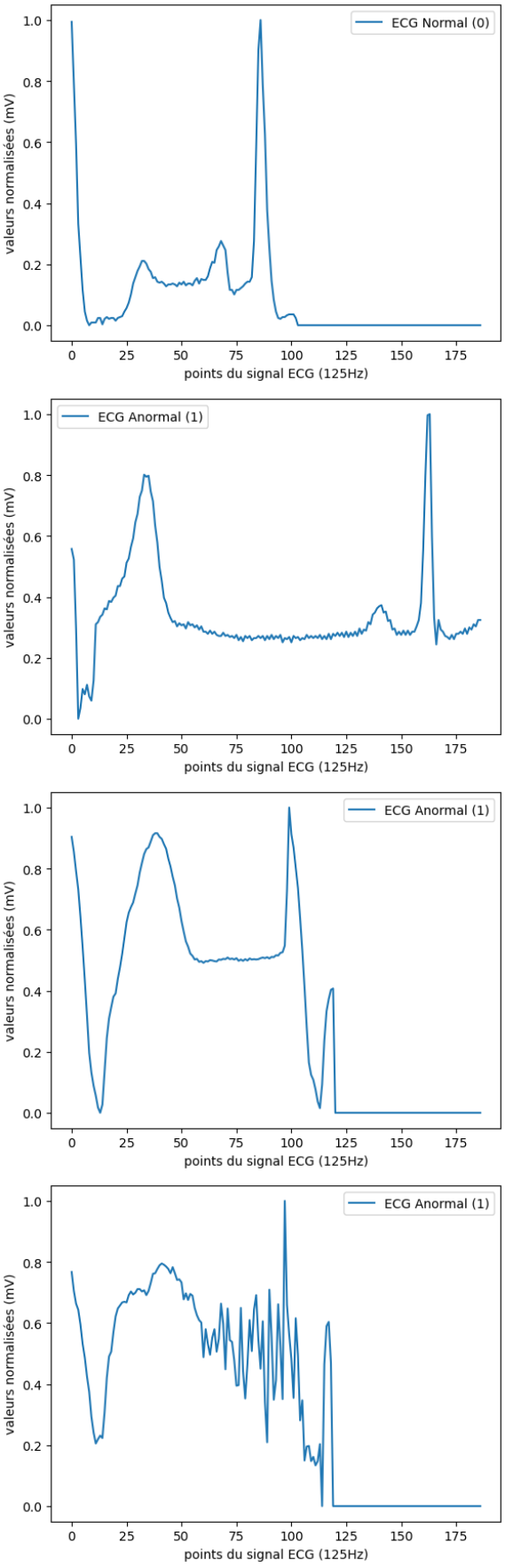


Figure 4-VS : Examples of ECG beats. The plots are obtained with the matplotlib library

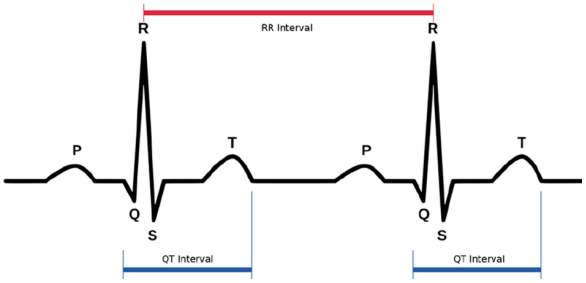


Figure 5-VS : Typical ECG

In this diagram we can understand that during a cycle (one beat) different P, Q, R, S, T waves follow each other on the recording. These waves represent the passage of a natural electrical signal through the heart, corresponding to an ordered series of muscle contractions/relaxations within the different parts of the heart. This normal cycle can be altered in certain pathologies.

The normal signal seen above (figure 4-VS) matches well with the descriptive drawing (figure 5-VS).

On the other hand, we notice that the abnormal signal differs at least in some points from the ECG. There are 5 major types of cardiac pathologies associated with known ECG patterns for cardiologists. see the following reference for more details[?].

2.2.2 Target variable

Our goal is to make a binary classification between two classes of signals. Our target variable must therefore be a binary label, indicating whether the shape of the ECG signal is Normal (= 0) or Abnormal (= 1). In our case, the data has already been labeled by cardiologists (last column of each dataframe). We will therefore extract this information from the Kaggle dataframe to obtain a label vector.

Then we want to visualize the distribution of classes within each dataframe (figure 6-VS).

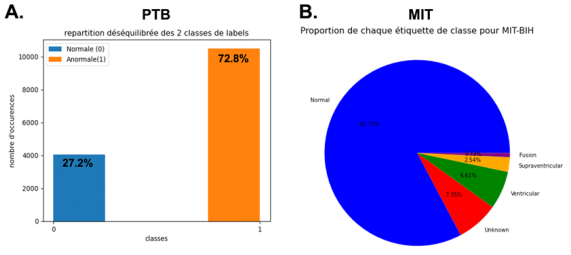


Figure 6-VS : Class distribution. A. for PTB database : 2 classes of unbalanced ECG ; These data are from 294 sick patients with 9 different heart diseases B. for MIT database : 5 classes of unbalanced labeled ECG ; These data are from 47 subjects chosen for their rare types of arrhythmia

The first important observation is that the data are unbalanced. The PTB dataframe contains more Abnormal ECG signals (72

The second important observation is that the MIT database contains 5 classes. This is due to the fact that there are several types of cardiac abnormalities. Therefore, we will need to group them into a single class of Abnormal ECG, in order to obtain two classes as for the PTB database.

2.3 Merging Kaggle sets to get our dataset

Our goal is to merge two dataframes, MIT and PTB, to obtain a single dataframe with even more clean data ready for modeling. We started by standardizing the labels of the target variables, grouping the MIT labels of different types of cardiac abnormalities (1,2,3,4) into a single label with a value of 1 (Abnormal ECG). We have previously seen that these dataframes are compatible with each other. We therefore decide to merge them.

Now, we will recalculate the new distribution of classes in our merged data (figure 7-VS)

Figure 7-VS : Distribution of classes. We get over 70% normal ECGs (class0), versus 30% abnormal ECGs (class1)

This graph provides us with two pieces of information. First, the normal class (0) is predominant over the abnormal class. Indeed, there are about 3 times more normal signals. This observation suggests that a class rebalancing may be used during model optimization. Methods of under or over sampling or even smote seem interesting to counter this potential bias.

2.4 Obtaining training, test and validation sets

At this stage, our data can be divided into several sets, necessary for the modeling steps. We thus create

a validation set (10.0

Now we will try to observe if our two classes present signals with systematic differences which would therefore be immediately visible through a visualization of the average trend. We have chosen here the median instead of the average in order to avoid potential biases due to outliers. The graph below describes the median of the different signals :

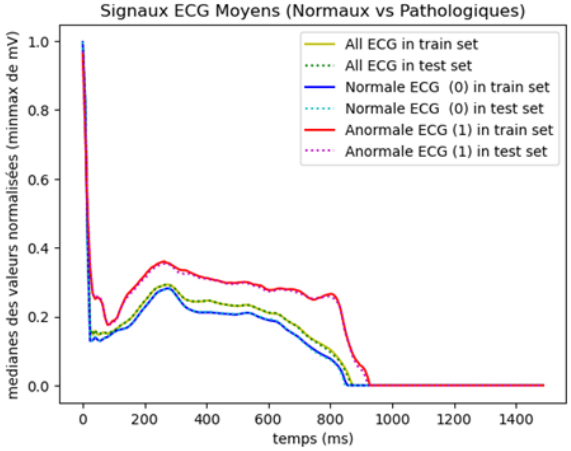


Figure 8-VS : Median signals as a function of classes.

Here we can see several things. First, the median of the abnormal and normal signals seems to follow a similar shape but with different amplitudes. The signals start to diverge at the end of the first R peak with a stronger amplitude for the abnormal ECGs. Moreover, the fact of using a median (or average) tends to remove the details for example we distinguish the T wave at 300ms but after that everything seems smoothed. We therefore lose fine temporal resolution when working with sets of curves. We will see if this can affect the models. Furthermore, we note that the test and training sets obtain identical median values which indicates to us that our data has been divided homogeneously.

2.5 Features engineering : the heart rate

After exploring our data, we decided to extract the R-R peak times in order to calculate the corresponding heart rate value for each ECG beat. We thus obtain a new column in our dataframe, corresponding to a new explanatory variable of quantitative type named "fcard" resulting from our feature engineering. These new variables could reinforce our models for their predictions by providing access to additional information. Let's check the distribution of heart rates thus obtained (figure 9-VS).

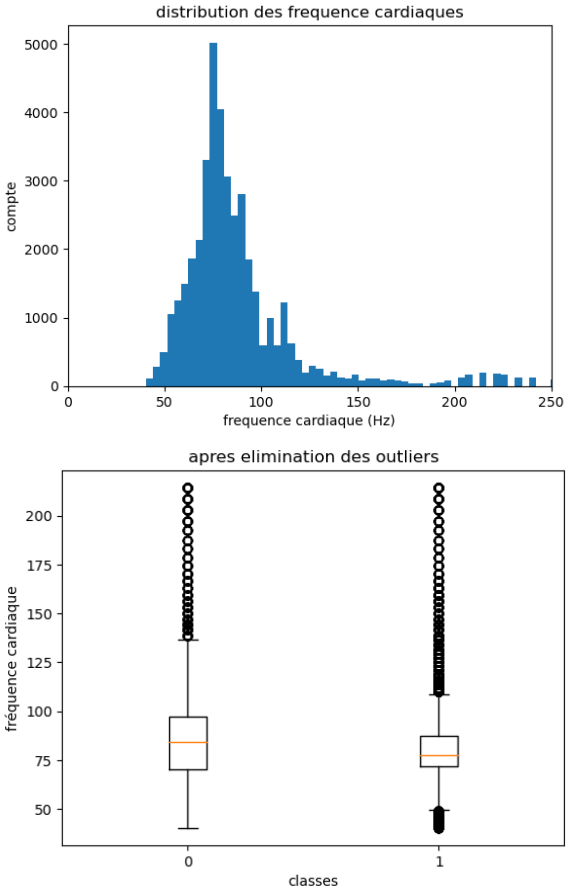


Figure 9-VS : Distribution of heart rates

The first graph describes the overall distribution of heart rates. The majority of values are between 40bpm (beats per minute) and 150bpm which corresponds to expected frequencies for the normal functioning of adult human hearts. The large interindividual variability is expected. We also get values above 220bpm which is physiologically impossible. These values are outliers. This is largely due to the fact that our calculation algorithm is based on Python's max and argmax functions, making it very sensitive to the quality of the R peaks. Some ECGs are noisy and of various shapes (figure 4-VS) which leads to errors in estimating fcard. To simplify, we decide to set a maximum of 220bpm.

Finally, the boxplot above (figure 9-VS) shows us that the medians are close but that abnormal ECGs seem to be associated with slower heart rates, with a longer time between R peaks (which also stands out in figure 8-VS). In addition, the distribution (quartiles) of abnormal heart rates is more restricted than that of normal ones. This seems to indicate less "cardiac flexibility". Our fcard variable is ready for modeling, we will try to use it during model optimization (see ANN part).

2.6 Summary of the data preparation phase

Before addressing the modeling phase, let's quickly review our dataset, we have :

187 columns made up of ECG recording points, our explanatory variables The "class" column containing our target variable, normal ECG (0) or abnormal ECG (1) Our classes are unbalanced so modifications will have to be considered No null or NaN values.

Our dataset is therefore ready to train and test our predictive models.

---

3 Classical Machine Learning Models

Logistic Regression - Decision Tree - K Nearest Neighbors

Here we enter the modeling phase of the project. This is a complex step with many choices to make and therefore requires in-depth analysis. It constitutes the fundamental core of a Data Scientist's work. We decided to start with the simplest classification models to interpret in order to get a general idea of the baseline scores against which to compare our more complex models.

3.1 Relevant Metrics

The **accuracy** is the proportion of correct predictions (True Positives "TP" and True Negatives "TN") among the total number of cases examined (including False Positives FP and Negatives FN). Accuracy is therefore calculated as follows (TP+TN)/(TP+TN+FP+FN). We must be careful with imbalanced data as is the case here where accuracy does not provide a reliable measure. For this reason here we will only use it to evaluate overfitting by comparing training and test data during the training iterations of the models. To quantify the overall score we prefer to use the F1 score.

The **precision** : The precision is also called Positive Predictive Value. It corresponds to the rate of correct predictions among the positive predictions, it is calculated as follows : TP/(TP+FP). It measures the model's ability not to make an error during a positive prediction.

The **recall** : this metric allows us to know the proportion at which the model detects signals as being in class 0 or 1 and is calculated as follows : TP/(TP+FN). This measure allows us to verify that possible biases do not influence the scores such as an imbalance of the data for example.

The **F1-score** : this last is defined as being the harmonic mean of precision and recall. This measure therefore combines the two previous ones. The higher this value, the more efficient the model will be. We will use it to assess the overall score of the models as it takes into account the recall (and is therefore not subject to the limitation of accuracy).

The **importance** : it is a value assigned to each explanatory variable, which will quantify the influence or impact of the variable in the result of the classification. For example, in the context of a decision tree, it is defined as the total decrease in impurity criterion between a node and the two following nodes.

The grouped study of these metrics is essential in order to avoid biases within the model that could arise notably due to class imbalance.

3.2 Logistic Regression

Logistic regression is widely used because it is extremely efficient and does not require huge amounts of computing resources. It can be easily interpreted and does not require scaling of input features. This model is also easy to implement, which makes it an excellent benchmark to help measure the performance of other complex algorithms. However, logistic regression is not the most powerful algorithm available. There are several alternatives that can create much better and more complex predictions.

Logistic regression is a statistical model that is often used for binary classification and we apply it to classify the data into two classes : normal and abnormal.

3.2.1 Logistic Regression without optimization

First, we created a raw logistic regression model without performing any optimization or parameter tuning. This raw model allows us to obtain initial results that give us an idea of the model's performance before any optimization. We then evaluated the performance of the raw model using the F1-score metric, which measures the harmonic between precision and recall.

	Classes	Précision	Rappel	F1-score
Modèle Logit	0	0,85	0,97	0,9
	1	0,8	0,45	0,57

Figure 1-LR : Score tables without optimizations

The raw model achieved the best f1-score for class 0, due to the imbalance of our classes, there are only 57

3.2.2 Logistic Regression with Optimization

We then sought to correct the imbalance of our classes by using an undersampling method for a first optimization of the logistic regression model. This method has allowed us to rebalance the recall between our classes, with a gain of 23

In order to improve precision while maintaining a high recall, we used a hyperparameter search technique called GridSearchCV. This allowed us to find the best hyperparameters for our logistic regression model.

#	Optimisation testée		Impact sur la précision	Impact sur le rappel	Impact sur le f1-score
1	RandomUndersampling du jeu de données	0	0,89	0,8	0,84
		1	0,52	0,68	0,59
2	Ajout des meilleurs paramètres avec GridSearchCV "C=1000" et "solver=liblinear"	0	0,89	0,8	0,84
		1	0,52	0,69	0,59

Figure 2-LR : Score tables with optimizations

However, despite the addition of the C and solver parameters, this technique did not correct the model's lack of precision. Although we observed a one-point improvement on the recall of the abnormal class, the scores obtained on this class remain too low or even random.

Ultimately, the logistic regression model does not provide satisfactory results for our cardiac anomaly detection project, as the scores on the abnormal class are not acceptable. We therefore cannot retain this model for the rest of our development and a deeper analysis is not necessary.

3.2.3 Analysis and Interpretation

We will now try to see how this regression uses our data in order to better understand its operation.

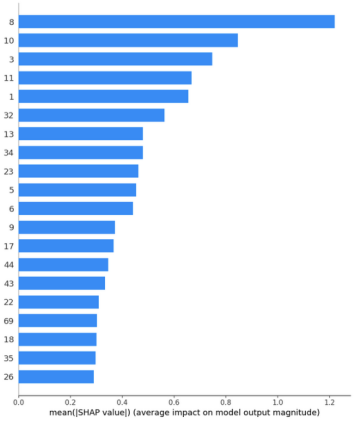


Figure 3-LR : Importance of variables

Overall, this graph shows us that the first points of the signal have a more significant impact on the model than the points located at the end. Indeed, points 8, 10, 3, 11, and 1 are in that order the most influential ones. Note that the larger these values, the more significant their impacts are.

The graph below shows us that large values located at the beginning induce the presence or absence of a peak. This characteristic being crucial in the differentiation between a normal or abnormal ECG, we can understand why these values have a significant importance for the model's choice. To further verify this hypothesis, we can place the important points on an ECG and see if they correspond to significant areas of the signal.

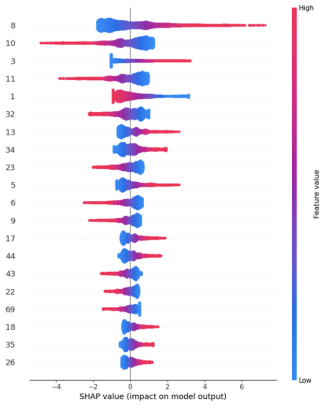


Figure 4 - LR : Importance of variables

Let's now take a signal but by placing the important points detected by Shap :

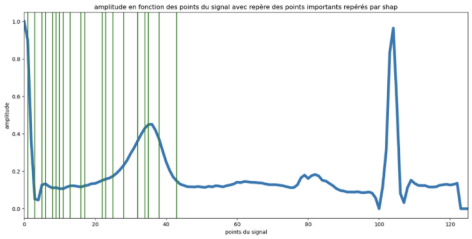


Figure 5 - LR : Important values placed on a class 0 ECG

We observe in this example that the model tries to discriminate the classes by the differences between the characteristics of normal and abnormal signals (presence or absence of peak for example). Below, other random signal examples allow us to reinforce this hypothesis.

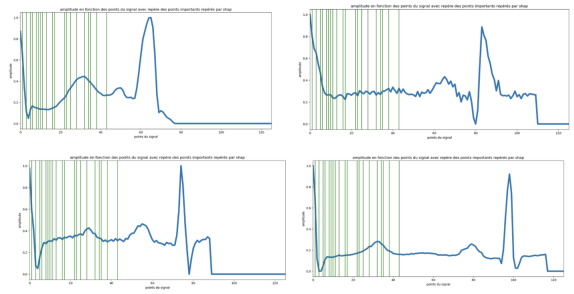


Figure 6 - LR : Examples of important values placed on ECGs

However, we can note the absence of important points located at the end of the ECG despite very variable characteristics between the signals at this location. This may explain the weaknesses of this model.

In conclusion, this model does not achieve very satisfactory results, indeed, the scores on class 1 remain too low or even random. This is not suitable in the context of a cardiac anomaly detection project. Logistic regression is not adapted to a large number of features. This algorithm cannot solve the non-linearity problem which requires the transformation of non-linear features.

3.2.4 Conclusion : advantages and limitations of the model

This logistic regression model is not very optimal for our goal, which is to predict heartbeats. Indeed, it has a significant lack of precision on the abnormal class, which is the one that matters most to us in this context. The recall on this class can be improved, but this is at the expense of precision, which is now comparable to that of a random prediction. In a cardiac anomaly detection project, it is crucial to have a model that can detect anomalies with high precision, as any prediction error can have serious consequences for the health of the patient to be predicted. Therefore, this logistic regression model needs to be replaced by a more sophisticated model.

3.3 Decision Tree

The Decision Tree is a supervised classification model. The Decision Tree can be described as a visual representation of a supervised learning algorithm that automatically selects discriminant variables from a very large database. The algorithm allows for extracting determinisms (logical cause-and-effect rules) in the data set called "nodes" or "decisions" following various quantitative criteria. Each node (a decision to split into two branches) is the result of a quantitative test. The final nodes form the terminal groups called leaves of the tree, each having one of the labels of the target variable (here 0 or 1 for Normal or Abnormal ECG respectively). During the training phase, the model's "decisions" are made from "optimal" tests based on thresholds calculated in relation to the training set. Generally, training stops according to a stopping criterion, homogeneity of groups, or maximum tree depth (maximum number of nodes for each branch). The advantage of the Decision Tree is that it allows obtaining easily interpretable results and provides a graphical representation of successive classification decisions, allowing understanding the logical rules that led to the predictions proposed by the model (see example figure 1). The hierarchical nature of a decision tree makes it easy to see which attributes are most important, which is not always clear with other algorithms, similar to neural networks.

3.3.1 Decision Tree without optimization

We began by training a simple tree of depth 2 (figure 1DT).

Classe	precision	rappel	f1
0	0.86	0.96	0.91
1	0.79	0.48	0.60

Figure 1 - DT : Tree of depth 2. The Gini indicates the node's impurity value.

The visualization of the tree makes it a very understandable model. For example, here we observe that the first two branches are created from a threshold value of 0.381 on explanatory variable number 4.

Now that our model is trained we will make a prediction on the labels of the test data set. The prediction gives us results summarized in table 1-DT.

	Classes	Précision	Rappel	F1-score	
baseline profondeur = 2	0	0, 86	0, 96	0, 91	
	1	0, 79	0, 48	0, 60	Rappel trop bas pour la classe1

	Optimisation testée		Impact sur la précision	Impact sur le rappel	Impact sur le f1-score	
1	profondeur = 8	0	0,91	0,98	0,94	
		1	0,91	0,69	0,78	
2	profondeur = 20	0	0,95	0,95	0,95	
		1	0,85	0,84	0,84	Surapprentissage

Table 1 - DT : "Baseline" scores obtained on a depth 2 tree on the test set.

These scores are good for class 0 but weak for class 1 (minority in the data). The recall of class 1 is less than 50

3.3.2 Decision Tree with optimization

The first step in optimization was to determine the optimal depth for the decision tree. A grid search indicates to us to use high depth values. We therefore test the model with a depth of 20. The scores obtained are better. However, the model becomes very long to produce a classification and the results become very difficult to interpret, as observed in figure2-DT. Such a tree seems inflexible and will probably struggle to adapt to a new data set.

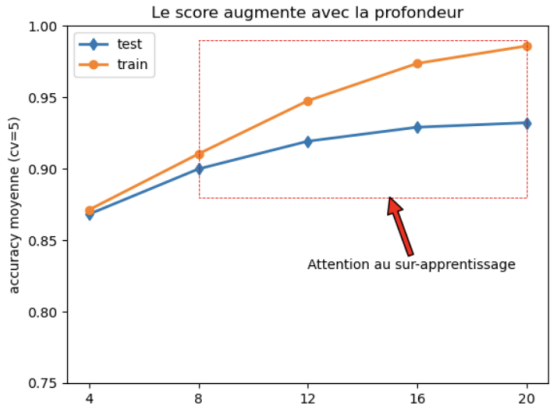


Figure 2 - DT : Tree of depth 20. The very deep tree has become difficult to understand.

We decided to represent the accuracy scores as a function of tree depth. And this during learning to visualize the difference between the training set and the test set (see figure 3-DT)

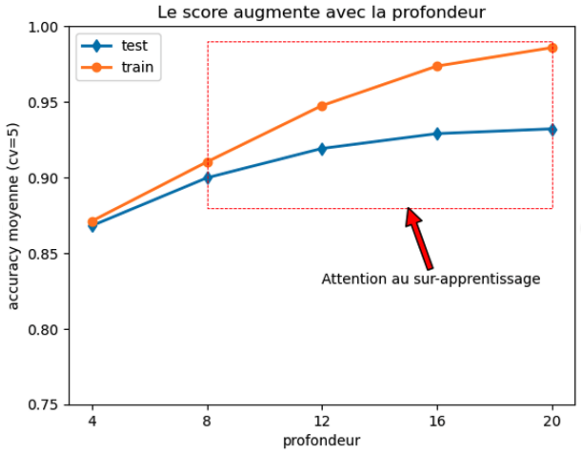


Figure 3 - DT : Accuracy as a function of tree depth.

The results of figure 3-DT indicate that model performance increases with depth, however, we clearly observe that the model achieves much better performance on the training set compared to the test set, indicating a lack of generalization, we talk about overfitting. From a depth of 8, we observe overfitting, we will fix the depth at 8 for optimization.

Other hyperparameters can be optimized to try to limit overfitting such as 1/ the minimum number of samples per leaf, 2/ the minimum decrease in impurity per node, and 3/ the maximum number of variables used per decision. By playing with these parameters, the model is constrained, the scores drop very slightly but this limits the overfitting phenomenon (data not shown).

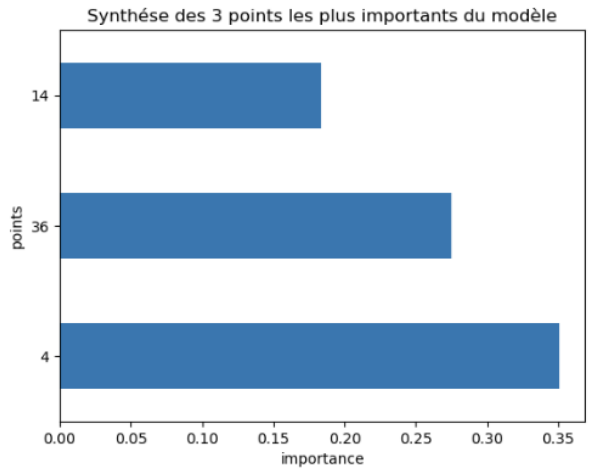


Table 2 - DT : Summary of the optimization

3.3.3 Analysis and Interpretation

The tree allows us to know the explanatory variables of the ECG with the most importance for classification. Initially, we could see that for the simple tree of depth equal to 2, the important points were 4, 14, and 36 (see figure 1-DT). We wondered if the depth could affect the model's choice of important points ? To answer this question we performed an analysis of the importance as a function of depth (figure 4-DT)

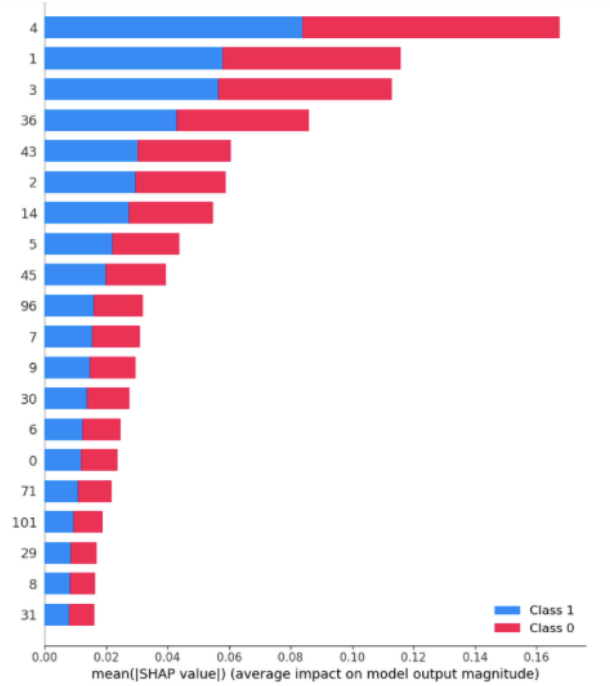


Figure 4 - DT : Importance as a function of depth We notice that the important points are the same regardless of the tree depth. Inset : Mean +/- standard deviation of all tested depths.

Finally, to visualize the important points for classification, we superimposed these points over time on the ECG signal (figure 5-DT).

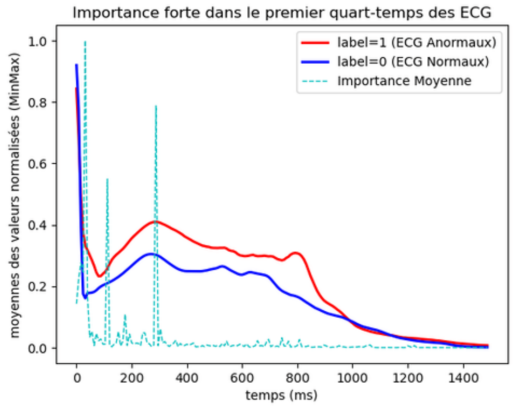


Figure 5 - DT : Importance overlaid with the average ECG signal. We observe that the peaks of importance correspond to the R1, S, and T waves. We observe that the R2 peak is drowned in the average and is therefore not visible.

This last analysis allows us to provide an interpretation of how our model works. In all cases, the model prefers to choose the first 5 points as well as points 14 and 36. The visualization in figure 5-DT shows us that these points correspond successively to the R (around 0 ms), S (around 100ms), and T (around 300ms) waves. The value taken by these points indicates whether or not the peak expected in the normal case has occurred. As these points are located close to point 0, the waves remain relatively well temporally aligned on these points, whereas the variations further from zero tend to desynchronize due to considerable inter-individual variability. This phenomenon is visible on our average signals where the R2 waves are almost invisible.

3.3.4 Conclusion : Advantages and Limitations of the Model

The model provides good explainability and is easy to handle. However, the model seems prone to overfitting as soon as the depth is too high. This prevents us from achieving high performance, especially for class 1, despite the additional optimizations carried out either by rebalancing the classes or by under-sampling the ECG (data not shown here). Therefore, we will need to explore other models or complicate the simple tree by using a random forest (not tested here) or even CATboost (see further).

3.4 K-Nearest Neighbors

KNN is a non-parametric classification method that looks for the k nearest neighbors of a new observation to predict its class. Unlike other algorithms, it does not make assumptions about the data, making it suited for complex and heterogeneous datasets. To get good results with this model, it is crucial to choose the optimal k value and the appropriate distance to measure similarity between data points. A too small k value may lead to overfitting, while a too large one may introduce significant bias in the classification. Therefore, we need to find the right balance for our model.

However, the KNN model can be computationally expensive, especially for large datasets or ones with many variables. The computation of distances between the new observations and the training set can be resource intensive. To solve this issue, it is possible to use dimensionality reduction techniques to decrease the complexity of the data, or to explore other classification algorithms that are more suited to large datasets.

3.4.1 KNN without Optimization

Here are the initial results obtained from the classification report and the confusion matrix of the KNeighborsClassifier model with default hyperparameters (k=5, weights='uniform') :

```
Modèle KNN avec k = 5

Temps d'entraînement: 0.03 secondes
Temps d'exécution: 59.48 secondes
              precision    recall  f1-score   support

         0.0       0.96      0.99      0.97     18990
         1.0       0.95      0.87      0.91      5810

    accuracy                           0.96     24800
   macro avg       0.96      0.93      0.94     24800
weighted avg       0.96      0.96      0.96     24800
```

Figure 1 - KNN : Score tables without optimization

The model shows promising results for the normal class (0) with high accuracy. However, it predicts the abnormal class (1) less well, which can be attributed to the class imbalance in the data. As a result, the recall for class 0 is 99

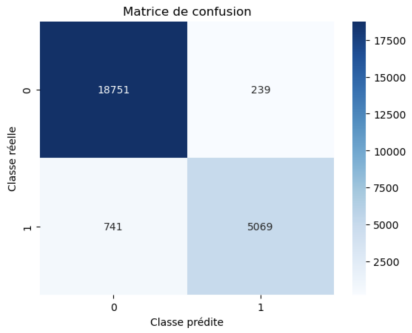


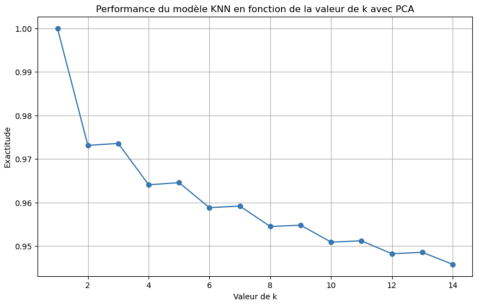
Figure 2 - KNN : Confusion matrix of the model without optimizations

To improve the model, it is necessary to apply optimization techniques and check if there is overfitting of the data. These measures aim to rebalance the model's performance between the classes and improve its ability to accurately predict the abnormal class.

3.4.2 KNN with Optimization

In this part, we used the PCA (Principal Component Analysis) dimensionality reduction technique to speed up the model. PCA helps reduce the complexity of data by decreasing their dimensionality, making it easier for the model to work and making the data more easily interpretable. This approach allowed us to determine the optimal value of the hyperparameter k by using a loop with k varying from 1 to 15. This process would have been costly in terms of time and computation for the machine, especially for voluminous data.

The graph below shows the k values in relation to the accuracy achieved after dimensionality reduction by PCA :




```
Meilleure valeur de k :  1
```

Figure 3 - KNN : K values in relation to the accuracy of the KNN model via a PCA.

The graph suggests that the optimal k value to maximize the model's accuracy is k = 1. However, it is important to critically examine this choice, as such a small k value is likely to lead to overfitting of the data.

Overfitting occurs when the model fits too tightly to the training data and loses its ability to accurately generalize on new data.

It is important to note that the reduced dimension to 2, represented in the graph, is used for visualization purposes and may not capture the full variance of the data. It is recommended to perform a more detailed analysis and consider other evaluation metrics to choose the best k value.

```
Modèle KNN avec k_best = 1

Temps d'entraînement: 0.02 secondes
Temps d'exécution: 81.85 secondes
              precision    recall  f1-score

         0.0       0.98      0.99      0.98
         1.0       0.95      0.92      0.94

    accuracy                           0.97
```

Figure 4 - KNN : Scores with optimization

By using k = 1 for the KNeighborsClassifier model, we observed a significant improvement in the results. The recall for class 1 increased from 87

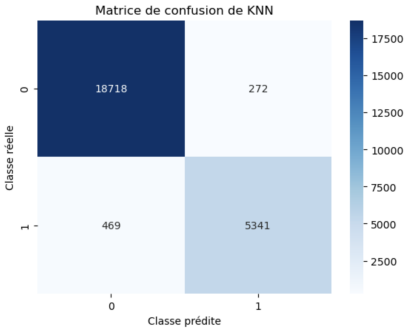


Figure 5 - KNN : Confusion matrix of the model with optimization

By examining the confusion matrix, we find that out of the 18990 samples of the normal class, 18718 are correctly predicted (true negatives), while out of the 5810 samples of the abnormal class, 5341 are correctly predicted (true positives). However, there are also 272 samples of the abnormal class that are wrongly classified as normal (false negatives) and 469 normal samples wrongly classified as abnormal (false positives).

3.4.3 Analysis and Interpretation

The learning curve is a graph that shows the evolution of a model's performance in relation to the amount of data used for training. It allows us to visualize whether the model is benefiting from an ample quantity of training data, and if the model is underfitting or overfitting.

It is important to strike a balance between accuracy and the model's ability to generalize to new data. In some cases, a higher value of k may be preferable to reduce the risk of overfitting and to allow for better generalization.

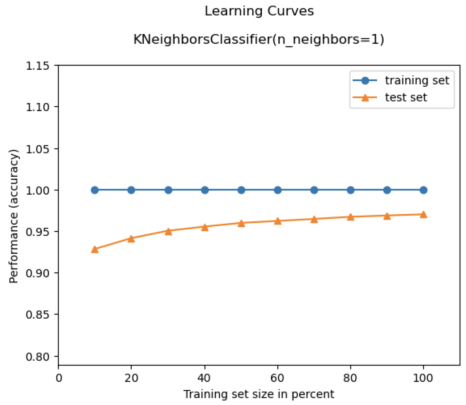


Figure 6 - KNN : Learning curve of the KNN model with k = 1.

The learning curve shown above for k = 1 indicates overfitting of the data, meaning that it has learned the training examples too precisely, to the point of not effectively generalizing to new data. As a result, it exhibits high accuracy on the training set, but less satisfactory performance on the test set. Therefore, it is important to take measures to avoid overfitting and improve the model's ability to generalize to new observations.

In this context of overfitting, the Ridge optimization algorithm applies regularization by penalizing extreme parameter values and favoring simpler models that provide similar results. This helps control the model's complexity and promotes better generalization to new data. By adjusting the regularization parameter, one can find the right balance between accuracy and model complexity.

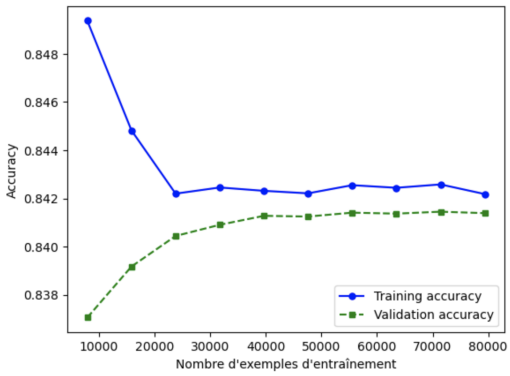


Figure 7 - KNN : Learning curve of the KNN model with Ridge regularization and k = 1.

The analysis of the learning curve obtained with Ridge-type regularization presents some encouraging aspects, but also elements that require deeper evaluation. On one hand, the convergence of the learning set and validation set curves towards similar performance suggests that regularization has helped reduce overfitting. This indicates that the model is correctly generalizing to new data, which is a positive result. On the other hand, the decline of the learning set curve may indicate that the model is not adequately capturing complex patterns in the training data. This could limit its performance and ability to adjust to real data. However, the growth of the test set curve as the training size increases is encouraging, suggesting that the model improves with more data.

Choosing a too small integer k, such as k = 1, in a classification problem can lead to ambiguities, particularly when the number of classes is even. It is generally recommended to choose an odd k, which avoids ties during voting. Furthermore, the choice of k should be large enough to allow a decision based on a significant number of neighbors.

By analyzing the graph in Figure 1, we can estimate a k that might be suited to our problem. By choosing k = 3, we have an odd number, which is large enough relative to the number of classes being studied, and which provides good model accuracy. It would therefore be more prudent to opt for this reasonable choice in the case of a binary classification problem.

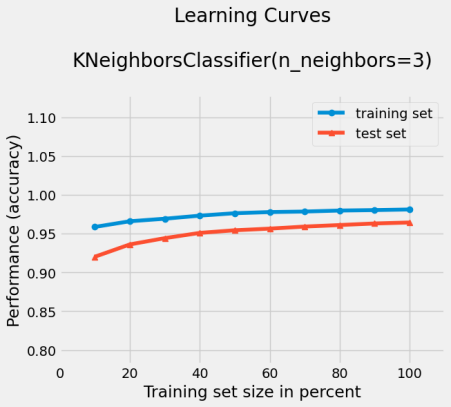


Figure 8 - KNN : Learning curve of the KNN model with k = 3.

The KNN model with k = 3 has experienced a slight reduction in recall for this minority class. However, this decrease is counterbalanced by an improvement in the model's ability to generalize and thus potentially reduce overfitting.

Additionally, the learning curves of the training and test sets display a simultaneous increase and convergence with a reasonable gap between the two. This indicates that the KNN model with k = 3 is effectively learning from the training data while maintaining good performance on new data.

3.4.4 Conclusion : Advantages and Limitations of the Model

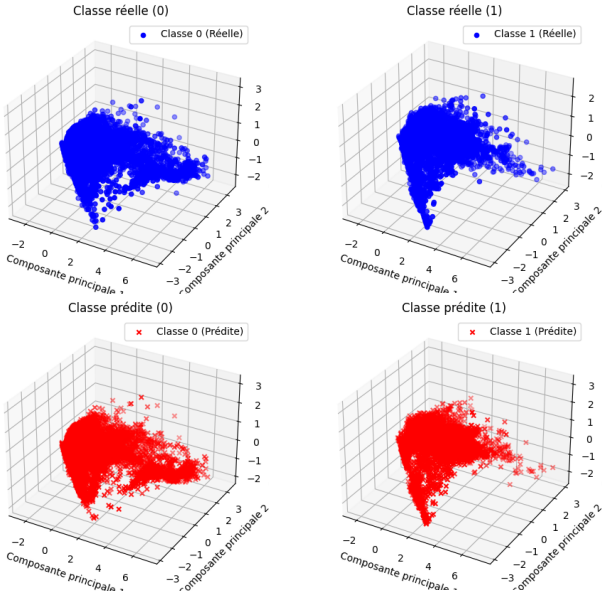


Figure 9 - KNN : 3D Plot of Real and Predicted Classes

By visually analyzing the 3D plots of the real and predicted classes below, we observe a similarity between the areas of points. This indicates that the KNN model with k = 3 has successfully captured the distinctive characteristics of normal and abnormal classes, suggesting some reliability in its ability to distinguish these two types of instances.

In conclusion, the KNN model is a simple and effective tool for classifying heartbeats. However, it is important to consider the limitations related to the choice of k, the quality of the data, and the specific constraints of the model, especially for large databases such as ECGs. A thorough evaluation and comparison with more advanced approaches are recommended for a better understanding of the model's performance.

4 Advanced Models and Deep Learning

CatBoost - Artificial Neural Network (ANN) - Recurrent Neural Network (RNN)

We are now tackling the training of more complex models of the "black box" type which are much more difficult to understand. Among these complex machine learning models such as CATboost, we will also test deep learning models such as artificial neural networks. Deep learning models present advancements compared to classical machine learning models, especially in terms of mathematical sophistications and usually lead to higher levels of predictions.

4.1 Gradient Boosting : CatBoost

CatBoost is a supervised Machine Learning algorithm used for classification and regression. It stands out from other decision tree-based methods by its ability to handle categorical data without preprocessing. It uses a method called ordered encoding to encode categorical entities, and symmetric trees to improve the speed of the algorithm. The CatBoost algorithm enhances the original gradient boosting method for faster implementation. It also retains some features of previous algorithms such as cross-validation, regularization, and missing values. Furthermore, it has overfitting detection parameters, but does not have any other boosters than trees available. CatBoost can work with small or large data.

4.1.1 CatBoost without optimization

Initially, we implemented the CatBoostClassifier model using the hyperparameters specified in the table below. The hyperparameters were selected by adjusting their value based on model performance to maximize metrics such as accuracy, recall, and the f1-score.

CatBoost modèle 1	
iterations	3200
learning_rate	0.08
depth	8

Figure 1 - CAT : Hyperparameters selection

The model undergoes 3200 iterations, which may seem substantial, but CatBoost handles large data well and maintains a very advantageous computational speed. Here are the results below :

	precision	recall	f1-score	support
0.0	0.98	0.99	0.99	18990
1.0	0.98	0.93	0.95	5810
accuracy			0.98	24800
macro avg	0.98	0.96	0.97	24800
weighted avg	0.98	0.98	0.98	24800

Figure 2 - CAT : Score table without optimization

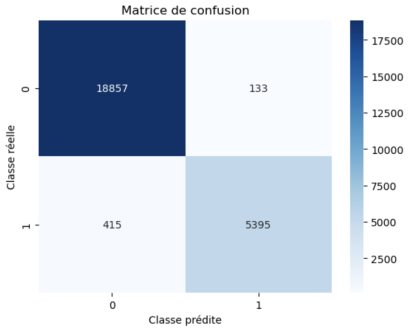


Figure 3 - CAT : Confusion matrix of the model without optimization

The code uses the CatBoostClassifier algorithm to classify data using custom hyperparameters such as the number of iterations, the learning rate, and the decision tree depth.

The model's performance is very high, with an accuracy of 99

It should be noted that the CatBoost algorithm is known for its intrinsic ability to handle class imbalance problems, which could explain the high performance without hyperparameter optimization.

4.1.2 CatBoost with optimization

We added hyperparameters to our CatBoostClassifier model to compare its performance with the initial model. By adding these hyperparameters, we aimed to refine and improve our model's ability to capture the data's discriminant features and to generalize more accurately on new examples.

CatBoost modèle 2	
iterations	3200
learning_rate	0.09
depth	8
l2_leaf_reg	3
border_count	64
random_seed	42

Figure 4-CAT : Hyperparameters Selection

Here are the results obtained with this list of hyperparameters :

```
Shrink model to first 3601 iterations.
              precision    recall  f1-score   support

         0.0       0.98      0.99      0.99     18990
         1.0       0.98      0.94      0.96      5810

    accuracy                           0.98     24800
   macro avg       0.98      0.96      0.97     24800
weighted avg       0.98      0.98      0.98     24800
```

Figure 5-CAT : Score table with optimization

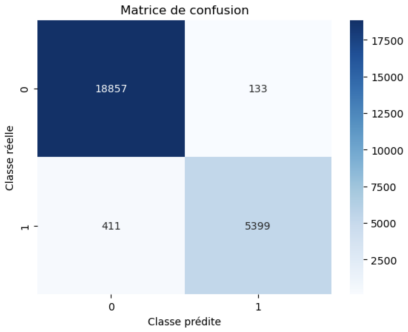


Figure 6-CAT : Confusion Matrix of the Model with Optimization

Although we have added several hyperparameters to the CatBoost model, the observed changes in terms of performance are not significant, with the exception of an improvement in recall and the F1 score for the class 1.0, which is potentially important. It is important to note that the influence of hyperparameters on the model could have been studied beforehand in order to adjust them in a more targeted way, for example using techniques such as grid search (GridSearch) or the Optuna algorithm, which allows for more quickly finding the best parameters for the model. This could have led to a more significant improvement in overall performance.

However, it is important to underline that adjusting hyperparameters does not always guarantee a substantial improvement in model performance. It is therefore essential to determine whether the changes made actually improve the model's performance. A more in-depth evaluation of these changes, particularly by comparing the results with other models or using additional metrics, may be necessary to fully assess the impact of hyperparameters on the overall performance of the model.

4.1.3 Analysis and Interpretation

To meet the performance of the model, we must take into account its learning curve in order to estimate whether our model is overfitting or underfitting. So, we obtained the following graph :

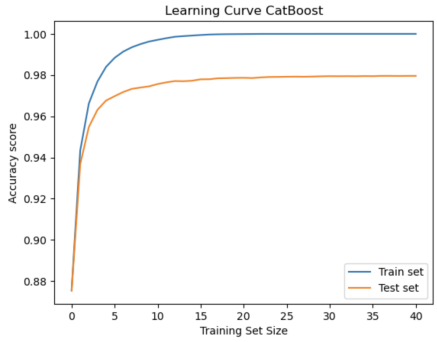


Figure 7-CAT : Learning Curve

The learning curve of the training set and the test set evolves similarly. This suggests that the model does not seem to be overfitted as it generalizes well on the test data. If the learning curve shows that the training set curve reaches a performance close to 1 and that the validation (test set) curve reaches a performance close to 0.98, this may indicate that the model is capable of predicting test data with very good accuracy.

We can rely on other graphs to analyze the robustness of the model, such as the calibration curve which is a very useful tool for evaluating the reliability of a model's predictions. It allows comparing prediction probabilities with their actual success rate. If the model is well calibrated, this means that the predictions are consistent with the predicted probabilities. On the other hand, if the calibration curve deviates from the perfect diagonal, this indicates that the prediction probabilities are not reliable and that the model or the training data need to be revised.

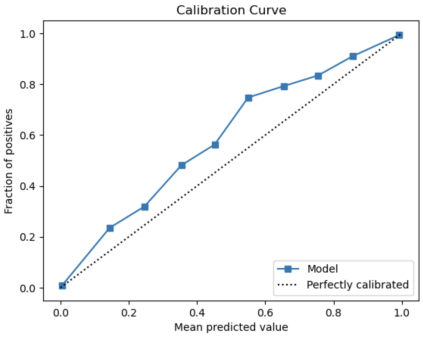


Figure 8-CAT : Calibration Curve

We notice that the model tends to approach the perfectly calibrated curve, then deviates more or less between 0.5 and 0.7 (mean predicted value). Otherwise, in general, the model does not deviate too much from the linear line, which allows adding a second argument about the validity of the CatBoost model, as well as its performance.

The feature importance graph allows seeing what is the relative contribution of each variable in the model's ability to predict the results. In the case of numerical ECG data, this means that we can observe the relative importance of each measure in the model's ability to predict a certain characteristic, such as the presence of an anomaly or the detection of a particular pathology.

In general, high feature importance indicates that the corresponding variable is important for the prediction and should be taken into account in the data analysis. Conversely, low feature importance indicates that the variable is not very useful for the prediction and can therefore be ignored or removed from the model to simplify its structure.

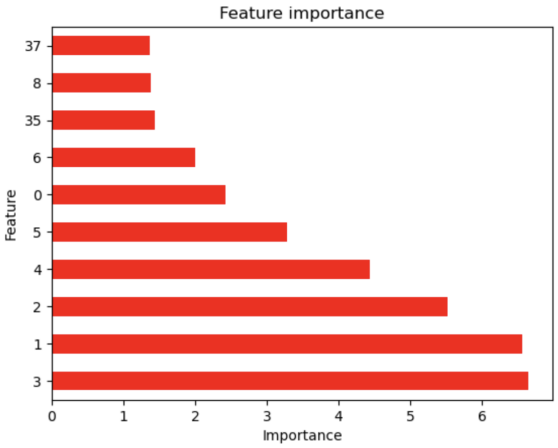


Figure 9-CAT : Feature Importance

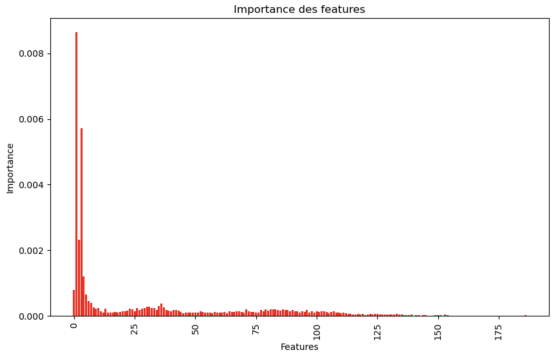


Figure 9-bis-CAT : Feature Importance

We can see that the important categorical variables are all in the first columns of the studied dataset, this can indicate that these data have a more important influence on the prediction of the class than the other columns. From variable 37, the importance of features begins to decrease little by little and beyond variable 100, it becomes almost non-existent in the model's influence. This could be due to several factors such as data quality, the relevance of features for the prediction task, etc...

It can be helpful to look more closely at the features of these columns to understand why they are so important and how this can help improve the performance of the model. Moreover, this may also indicate that some of the less important or useless columns can be removed to reduce the dimensionality of the feature space and facilitate the modeling process while improving the performance of the model.

4.1.4 Conclusion : Advantages and Limitations of the Model

The CatBoost model has significant advantages for binary classification of heartbeats from ECG data. It stands out for its ability to automatically handle categorical variables and effectively deal with missing values. In addition, it is capable of capturing nonlinear relationships between heartbeat characteristics and their class. The CatBoost model also offers good performance in terms of computing speed, making it an attractive option for real-time ECG data analysis. However, it is important to note that parameter optimization may be necessary to obtain the best results. In summary, the CatBoost model represents a promising approach for heartbeat classification, offering both accuracy and efficiency in processing ECG data.

4.2 Artificial Neural Network (ANN)

An artificial neural network is inspired by a biological neural network, which itself is composed of a stack of successive and connected layers of neurons. Thus, we can form several layers of algorithms with a complex architecture that will allow us to solve complex problems (in our case, a complex supervised classification problem). Our data enter through the input layer of the ANN, are then processed by the intermediate layer(s) (depending on the chosen architecture), and the output layer returns the result of the problem. Here, we have two output neurons, one for label 0 (Normal ECG) and the other for label 1 (Abnormal ECG).

4.2.1 ANN without optimization

We started by training a simple neural network with a single intermediate layer of 20 neurons (relu activation) and with an output layer of two neurons (softmax activation). We trained our model on our training set, with 100 samples per batch and a maximum epoch number of 1000. The model is equipped with an early-stopping callback function (with a patience of 20) which allows it to detect performance plateaus and stop there. We used the 'BinaryCrossentropy' loss function, the optimizer= 'adam' and followed the accuracy metric.

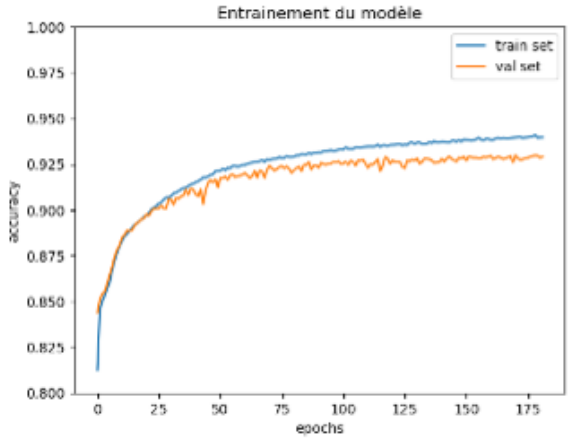


Figure 1-ANN : The learning curve of the training set reaches a plateau which stops the training after 175 epochs. We notice an overfitting that starts early (epoch 25) and continues to increase.

Following the accuracy scores during learning, and comparing this follow-up for the training and validation sets allows to observe an overfitting phenomenon (figure 1-ANN) Finally, the scores obtained on our test set are summarized in the following table :

Classe	Précision	Rappel	f1
0	0.93	0.98	0.95
1	0.92	0.75	0.82

Table 1-ANN : "Baseline" scores obtained on the test set with a 20-neuron ANN.

These scores are already well above those obtained via traditional machine learning models. However, the F1 score of class 1 remains at 0.82, which would need to be improved so that our model makes as few errors as possible. We will therefore try to optimize our model to increase the scores, especially those of class 1.

4.2.2 ANN optimization via architecture

The first test was to identify the optimal number of neurons for the ANN with a single intermediate layer. We therefore proceeded by trial and error, testing different numbers of neurons, all other things being equal. We obtained the following curve :

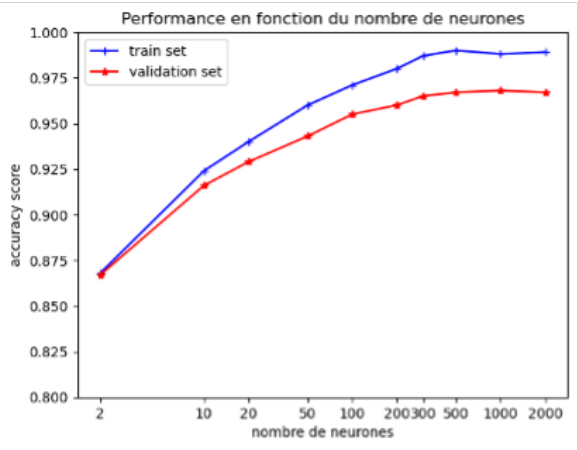


Figure 2-ANN : Performance increases with the number of neurons. However, overfitting also seems to increase with the number of neurons.

By increasing the number of neurons, we achieve better performance, but we increase overfitting. To significantly increase our score, we will choose a number of neurons of 300 but no more to limit overfitting.

We will now observe the effect of the number of layers on the performance of our 300-neuron model. To do this, we varied the number of layers while keeping all other things equal. Figure 3-ANN indicates that adding layers does not lead to a notable change in either accuracy or overfitting.

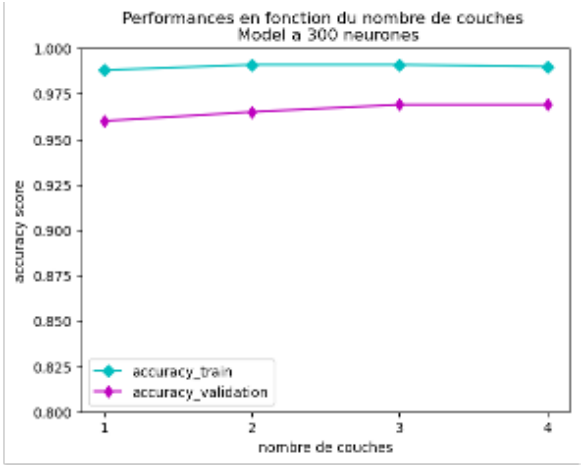


Figure 3-ANN : The performance of our model does not vary much depending on the number of layers. When evaluating the models via the test set, the best F1 score is obtained by the models with 3 and 4 layers with F1 = 0.93 for class 1.

However, the number of epochs required by the model to train effectively is reduced, as shown in Figure 4-ANN.

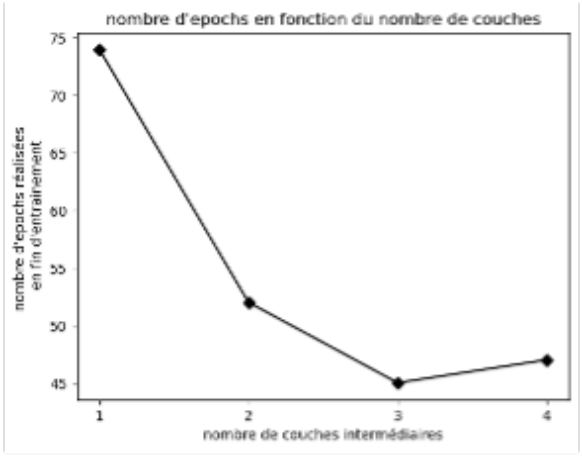


Figure 4-ANN : Here, 300 neurons are evenly distributed between the model's intermediate layers. Our model seems to converge more quickly to a solution when it contains a larger number of layers. 3 intermediate layers seem optimal in our case.

It seems that increasing the number of layers would allow the model to converge more quickly to a solution. We therefore opt for a model with 300 neurons and 3 intermediate layers (100 neurons per layer). The architecture and training of this new model are described in Figure 5-ANN. We note a rapid convergence towards high scores, but still with a presence of overfitting.

```
Layer (type)                 Output Shape
=================================================
inputs (InputLayer)          [(None, 187)]

dense_layer1_relu (Dense)    (None, 100)

dense_layer2_relu (Dense)    (None, 100)

dense_layer3_relu (Dense)    (None, 100)

dense_output_softmax (Dense  (None, 2)
```

New network architecture

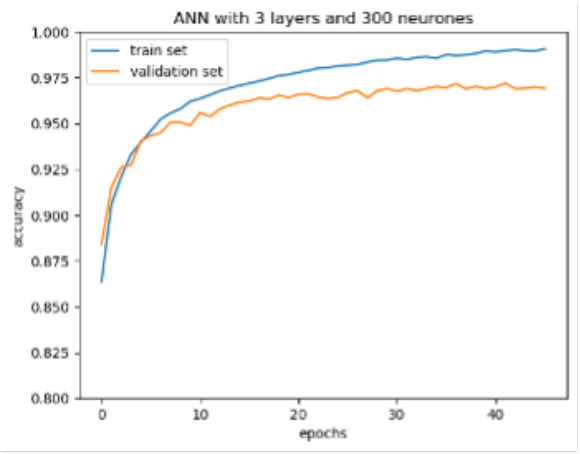


Figure 5-ANN : Our models after optimization. Compared to Figure 1-ANN, the performance is generally improved but the level of overfitting seems to persist.

Classe	Précision	Rappel	F1-score
0	0.97 (+0.04)	0.99 (+0.01)	0.98 (+0.03)
1	0.95 (+0.03)	0.91 **(+0.16)**	0.93 **(+0.11)**

Table 2-ANN : Optimized scores, obtained on the test set with an ANN with 3 intermediate layers of 100 neurons each. The parentheses indicate the difference with the "baseline model" from Figure 1-ANN. Notable improvements are indicated in red.

The Table 2 shows significant performance improvements after our first optimization step. Indeed, we have gained +16

Note that we tried to modify the network architecture using 150 neurons in layer1, 100 neurons in layer 2 and 50 neurons in layer 3. This modification did not improve the performances and was therefore not retained.

4.2.3 ANN optimization via inputs

After optimizing the model by modifying its own structural features, we decided to optimize it by modifying the model's inputs (data-driven optimization). For this step, we moved to a batch size of 32 and a patience of 10. We tried various modifications, particularly regarding the distribution of labels which in

our case was initially unbalanced. The best results were obtained with the "RandomOverSampler" function which allows to rebalance the classes by randomly oversampling data from the minority class.

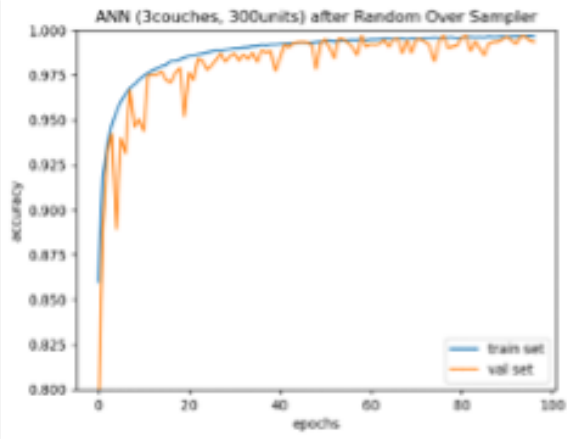


Figure 6-ANN : Learning curves obtained after resampling. High performances and no overfitting are observed.

Classe	Précision	Rappel	F1-score
0	0.98 (+0.01)	0.99 (+0.01)	0.98 (+0.00)
1	0.96 (+0.01)	0.93 **(+0.02)**	0.95 **(+0.02)**

Table 3-ANN : Optimized scores obtained on the test set after data rebalancing by random oversampling.

The Table 3 shows slight performance improvements after our second optimization step. Optimizing by modifying the model's input data allowed to gain 2 recall points which is not negligible considering our business problem. Moreover, our batch size and patience settings, combined with rebalancing, eliminated the overfitting phenomenon observed previously. Finally, we attempted to inject an additional explanatory variable, the heart rate (obtained previously by feature engineering), into our dataset. To do this, we normalized the heart rate between 0 and 1 (MinMax) ; the aberrant values higher than 200bpm were previously fixed at 200bpm, so the 1 represents all values equal to or greater than 200pbm. The "fcard" column was added to $X_train and X_test$.

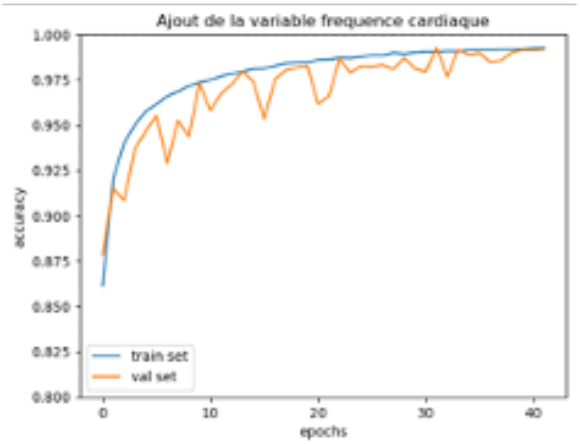


Figure 7-ANN : Learning curves obtained after adding fcard. Faster convergence is observed but the results are generally similar to those previously obtained.

Training the model with the addition of fcard in addition to the previous optimizations did not improve performance.

Classe	Précision	Rappel	F1-score
0	0.98 (+0.00)	0.99 (+0.00)	0.98 (+0.00)
1	0.96 **(-0.01)**	0.93 (+0.00)	0.94 **(-0.01)**

Table 4-ANN : Scores obtained on the test set after adding the heart rate.

As the results of Table 4 show, the addition of our heart rate variable did not improve the model's performance.

4.2.4 Interpretability and Analysis of the ANN

Now, we want to better understand how our model worked and what are the most important explanatory variables for the ECG classification.

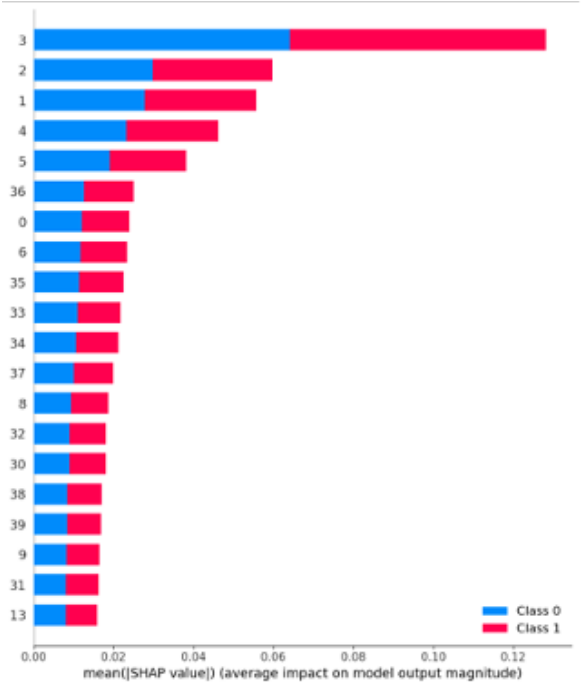


Figure 8-ANN : Interpretability with SHAP. We used DeepExplainer from Python's SHAP library to obtain global explainability on our neural network. The model analyzed here is that of Figure 7-ANN. The shap values were calculated on 6

The SHAP analysis of Figure 8-ANN allows us to give a global interpretation of how our model works. We note that the first 8 points of the ECG curves are those that have the most impact on the model's performance. These points correspond to the first R peak of the ECG. The second group of points with the most importance for the classification of our ANN are the points located between the 30th and 40th points of the ECG. These points correspond to the T wave of the ECG. It is interesting to note that, as with the decision tree seen previously (see Figure 5-DT), the waves close to the origin are the most relevant for the model. The 2 waves R1 and T seem to be enough for the model to solve most of the classification problem. Nothing here indicates that the model uses the R2 wave for classification, which seems too far from zero to maintain a stable temporal position between the samples. We hypothesize that this temporal instability does not allow the model to make comparisons of satisfactory statistical quality.

According to these results, we might think that only the first 40 points could generally suffice to classify satisfactorily. To test this idea, we performed a drastic reduction of the ECG dimensions by keeping only the first 50 points of the ECG for the model's training. The results (F1-score class0 = 0.98 and F1-score class1 = 0.93) are not very different from the results obtained on the 180 total points of the ECGs from the Kaggle database (see Table 3-ANN). However, the loss of 2

4.2.5 Conclusion : Advantages and Limitations of the Model

The significant advantage of the ANN is that it enables us to achieve very high scores while significantly limiting overfitting through fine-tuning. However, the ANN required a considerable amount of time for learning, optimization, and model interpretation compared to logistic regression or a decision tree. The complex optimization of the ANN necessitates extensive research and numerous empirical trials. Interpretability requires lengthy computation times and the use of complex software. Finally, the global interpretation of the results aligns with our previous analyses obtained on the decision tree and logistic regression, providing us confidence in our overall interpretation.

In conclusion, despite its limitations, our neural network has enabled us to surpass the performances of traditional machine learning models. Our classification performances are very good, with an error rate of less than 5

4.3 Recurrent Neural Networks

Recurrent Neural Networks (RNN) are powerful machine learning models used to analyze data sequences such as text, speech, or time series. These networks are capable of taking into account past information thanks to recurrent connections, thus allowing them to remember and make real-time decisions.

Compared to traditional neural network architectures, RNNs are particularly suited to processing sequential data. However, they can encounter difficulties when it comes to handling long-term dependencies, that is, relationships between distant events in the sequence.

Despite these challenges, RNNs remain valuable tools in many applications. Their ability to take into account contextual information and process sequences makes them excellent candidates for tasks such as automatic translation, text generation, and speech recognition.

4.3.1 RNN without optimization

To address the class imbalance, we performed oversampling by duplicating the observations of the minority class. Then, we evaluated different RNN model architectures by calculating recall, precision, and F1 metrics. We also plotted the evolution of these metrics over the epochs.

The deep learning model we used is an RNN with a GRU layer. We added a Flatten layer to prepare the data before adding a final layer with one neuron and a sigmoid activation function. At each epoch of learning, we recorded the value of the loss function, accuracy for the training dataset, and accuracy for the validation dataset.

Here are the results of the metrics of this first RNN model :

	Classes	Précision	Rappel	F1-score
RNN (GRU)	0	0.99	0.98	0.99
	1	0.94	0.96	0.95

Figure 1-RNN : Scoreboard without optimization

Then, here is the architecture of this basic RNN network that we used :

```
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 187)]             0
tf.expand_dims (TFOpLambda)  (None, 187, 1)            0
gru (GRU)                    (None, 187, 256)          198912
flatten (Flatten)            (None, 47872)             0
dense (Dense)                (None, 1)                 47873
=================================================================
Total params: 246,785
Trainable params: 246,785
Non-trainable params: 0
_________________________________________________________________
```

Figure 2-RNN : Model architecture without optimization

The evolution of accuracy for the training and testing data is presented in the following figure :

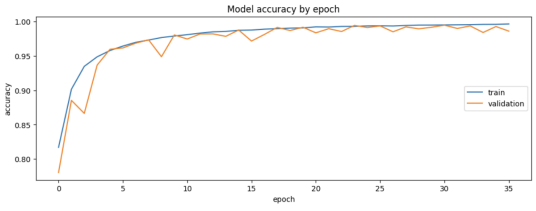


Figure 3-RNN : Evolution of accuracy for training and testing data

From the graph, we observe that the curves of the validation and training data overlap and are almost identical, leading us to deduce that there is no overfitting. Moreover, the obtained scores are, to our understanding, very good, with an f1-score of 99

4.3.2 RNN with optimization

In a second phase, and as the first optimization, we added a new dense layer of 100 neurons in the hope of improving the results.

Here is the score of the metrics obtained with this first optimization of the RNN network :

#	Optimisation testée		Impact sur la précision	Impact sur le rappel	Impact sur le f1-score
1	GRU avec couche 100 neurones	0	0.99	0.99	0.99
		1	0.97	0.97	0.97

Figure 4-RNN : Scoreboard with optimization

We can illustrate the architecture of the RNN network with the first optimization :

```
Layer (type)                 Output Shape              Param #
=================================================================
input_5 (InputLayer)         [(None, 187)]             0
tf.expand_dims_4 (TFOpLambd  (None, 187, 1)            0
a)
gru (GRU)                    (None, 187, 256)          198912
dense_7 (Dense)              (None, 187, 100)          25700
flatten_4 (Flatten)          (None, 18700)             0
dense_8 (Dense)              (None, 1)                 18701
=================================================================
Total params: 243,313
Trainable params: 243,313
Non-trainable params: 0
_________________________________________________________________
```

Figure 5-RNN : Model architecture with optimization

The obtained learning curve is presented in this graph :

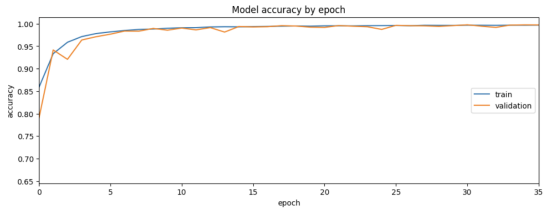


Figure 6-RNN : Learning curve

The curve shows that there is no overfitting where the validation score overlaps with the one given by the training data. The precision, recall, and F1 score are better than the neural network without the intermediate layer.

We were able to perform a second optimization where we tested the very famous LSTM model, very well known for capturing the essence of natural language but also adaptable to any type of time series.

This model obtains the following scores :

#	Optimisation testée		Impact sur la précision	Impact sur le rappel	Impact sur le f1-score
2	LSTM	0	0.98	0.99	0.99
		1	0.97	0.94	0.96

Figure 7-RNN : Score table with optimization

This model has the following architecture :

```
Layer (type)                 Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 187)]             0
 tf.expand_dims_1 (TFOpLambd  (None, 187, 1)           0
 a)
 lstm_1 (LSTM)               (None, 187, 256)          264192
 flatten_1 (Flatten)         (None, 47872)             0
 dense_2 (Dense)             (None, 1)                 47873
=================================================================
Total params: 312,065
Trainable params: 312,065
Non-trainable params: 0
_________________________________________________________________
```

Figure 8-RNN : Architecture of the model with optimization

The appearance of the learning curve is as follows :

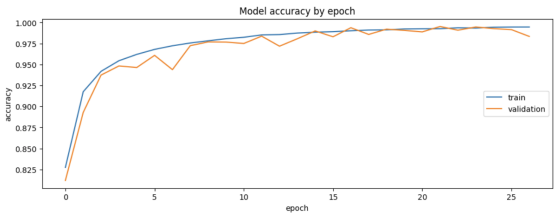


Figure 9-RNN : Learning curve

The performances of this model are roughly the same. It is possible to note that the F1 score has increased by 1

Then, we implemented another model with a third optimization, which we tested and is the same as the previous one but we added a dense layer of 100 neurons for which we obtained the scores :

#	Optimisation testée		Impact sur la précision	Impact sur le rappel	Impact sur le f1-score
3	LSTM avec couche 100 neurones	0	0.99	0.99	0.99
		1	0.96	0.97	0.96

Figure 10-RNN : Score table with optimization

We get the following architecture :

```
Layer (type)                 Output Shape              Param #
=================================================================
 input_3 (InputLayer)        [(None, 187)]             0
 tf.expand_dims_2 (TFOpLambd  (None, 187, 1)           0
 a)
 lstm (LSTM)                 (None, 187, 256)          264192
 dense_4 (Dense)             (None, 187, 100)          25700
 flatten_2 (Flatten)         (None, 18700)             0
 dense_5 (Dense)             (None, 1)                 18701
=================================================================
Total params: 308,593
Trainable params: 308,593
Non-trainable params: 0
_________________________________________________________________
```

Figure 11-RNN : Architecture of the model with optimization

The learning curve is given by the following figure :

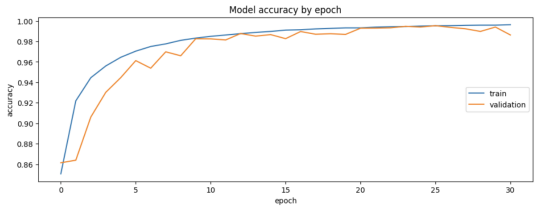


Figure 12-RNN : Learning curve

The model provides slightly higher scores than the model with LSTM without an intermediate dense layer. However, it is still a bit less efficient than the model with GRU and the same intermediate layer.

4.3.3 Conclusion : Advantages and limitations of the model

Regardless of the RNN architectures tested, scores can slightly improve, but not considerably, or deteriorate compared to what was previously presented. The results of the three models are summarized in the following table to highlight these findings.

It is important to remember that for all tested models, we applied an early stop with a patience of 5. This means that the model is stopped if the accuracy on the validation set does not improve for 5 consecutive epochs.

Moreover, it is worth highlighting that RNNs require a lot of computational time, which forced us to use graphic cards to speed up execution times.

As with the other models, the performances on class 1 are worse than for class 0. However, the RNNs obtain one of the best scores among the models already used, and slightly more accurate than the CatBoost model.

5 General Conclusion and Discussion

5.0.1 Comparison of all models

We have explored different models by level of complexity ranging from logistic regression to recurrent neural network. The following tables summarize our results.

Modèles Baselines	Classes	Précision	Rappel	F1-score
Régression logistique	0	0,85	0,97	0,9
	1	0,8	0,45	0,57
Arbre de décision	0	0,86	0,96	0,91
	1	0,79	0,48	0,6
KNN	0	0.96	0.99	0.97
	1	0.95	0.87	0.91
CATBOOST	0	0.98	0.99	0.99
	1	0.98	0.93	0.95
ANN	0	0,93	0,98	0,95
	1	0,92	0,75	0,82
RNN	0	0.99	0.99	0.99
	1	0.94	0.96	0.95

Figure 1-CC : Table of model scores without optimization

Meilleurs modèles	Classes	Précision	Rappel	F1-score
Régression logistique	0	0,85	0,97	0,9
	1	0,8	0,45	0,57
Arbre de décision	0	0,86	0,97	0,91
	1	0,91	0,69	0,78
KNN	0	0.98	0.99	0.98
	1	0.95	0.92	0.94
CATBOOST	0	0.98	0.99	0.99
	1	0.98	0.94	0.96
ANN	0	0,97	0,98	0,98
	1	0,95	0,93	0,95
RNN	0	0.99	0.99	0.99
	1	0.97	0.97	0.97

Figure 2-CC : Table of model scores after optimization (here we show the best scores obtained)

It is noted that in all cases the models obtain better scores for the majority class (class 0 representing normal ECGs). We saw earlier that even rebalancing through oversampling could not fully compensate for this bias. This indicates that the quality and quantity of data have a lot of impact on the performance of the models. However, optimization in some cases significantly improved the performance of the models. This is the case for the decision tree and the artificial neural network, particularly for class 1 (minority) where performances were hard to improve. The fact that the optimization of some models did not improve classification scores could be due to several factors. One of them could be that the model parameters were well adjusted by default, leading to a high baseline score that is difficult to improve. In general, we note that complex models such as CATboost and neural network models achieve high classification scores for both classes. However, these models have taken us a lot of time to train, optimize, and interpret. In some cases, the high-score models are subject to overfitting (KNN and CATBoost). This overfitting indicates that the model will struggle to generalize. We will therefore favor models for which overfitting is low or non-existent. Sometimes optimization has eliminated or significantly reduced overfitting (as in the case of the ANN). In the case of the tree, if we did not adjust the hyperparameters (keeping the default settings), we obtained a very deep tree with a very high score. This seemed very encouraging, but in reality the model was overtrained on our data and was unable to generalize. This would render it useless or at least not eligible for our business problem. It is noted that for the recurrent neural network, we could not find a way to achieve interpretability, neither with SHAP nor with SHAPASH. Although we are aware that it should be feasible with more time, the fact that we could not obtain interpretation results on the model's functioning reflects how difficult it is to understand a recurrent neural network compared to a simple model such as logistic regression or the decision tree.

5.0.2 Discussion and future directions

Although these simple models do not always achieve high scores, they have the advantage of allowing us to understand how the classification is performed. However, we were still able to obtain interpretability for the ANN. It is interesting to note that the parts of the ECG used by the ANN are the same as those used by the simpler models. Indeed, whether it is for logistic regression, the decision tree, CATboost, or the ANN, we always find that the points of the ECG corresponding to the R1, S, and T waves are the most important for classification. This once again indicates the importance of the initial data. The fact that the P, Q, and R2 waves, which are furthest from the origin, are hardly useful to the models indicates that the latter require robust and reliable data to make their prediction. However, in this case of classification, the distance from the zero point is synonymous with statistical instability because interindividual variabilities will prevent obtaining a synchronized ECG trace after 400-500ms. This observation is very interesting in itself

and provides us with avenues for possible improvement in our classification. For instance, we could place the zero of the curves just before the P wave and observe how the models would perform, having the P Q R2 wave as an anomaly indicator instead of R1, S, T. We could use transfer learning to retrain our pre-trained model on the Kaggle data and train it on the modified data with a zero before the P wave. This model could yield higher scores than those obtained here, which would confirm our analysis.

5.0.3 Overall assessment

Finally, for our business problem, we would recommend the model with the lowest error rate, the RNN.

6 References

[1] M. Kachuee, S. Fazeli, and M. Sarrafzadeh. *ECG Heartbeat Classification : A Deep Transferable Representation.* IEEE International Conference on Healthcare Informatics (ICHI), pp. 443-444, doi : 10.1109/ICHI.2018.00092, 2018.