Capstone Project

Robson Rocha
October 9th, 2020.

Machine Learning Engineer Nanodegree

1 Definition

1.1 Project overview

This project consists in predict if an Electrocardiogram (ECG) presents some disease or not. To reach this goal, it will be compared two algorithms' answers and it will be check if it is possible to detect a healthy (or not) signal, just understanding its measures.

The electrocardiogram is a very important resource for cardiac medicine. With the graph drawn through an appropriate device, it is possible to view important information about the conditions of the patient's cardiovascular system.

The heartbeat is controlled by an electrical activity, that is, in order for the heart muscle to contract, electrical currents must pass through it. This electrical action in which the heart is inserted is closely linked to the question of being immersed in an electrically conductive saline solution. The principle of electrocardiography lies in the fact that the electrical currents generated in the heartbeat are likely to be registered on the body surface.

The heart has its electrical stimulus originated in a structure called the Sinus or Sinoatrial Node. This stimulus does not contract the heart in a uniform way, because if it were, there would be no blood pumping to the body.

Each contraction stage generates a type of wave that can be recorded on the ECG, this work consists in interpret and transformed in measures these waves and pass them to those algorithms classify them.

1.2 Graphic representation

1.2.1 A normal ECG

Figure 1 shows the stages of a heartbeat. The blue part of the graph represents the P wave, which reflects the result of the stimulus from its origin and shows the state of the indicated area. The QRS complex, in red, represents the stimulus propagated to the center and lower part of the heart. The T and U waves are the result of relaxing the heart muscle after contraction.

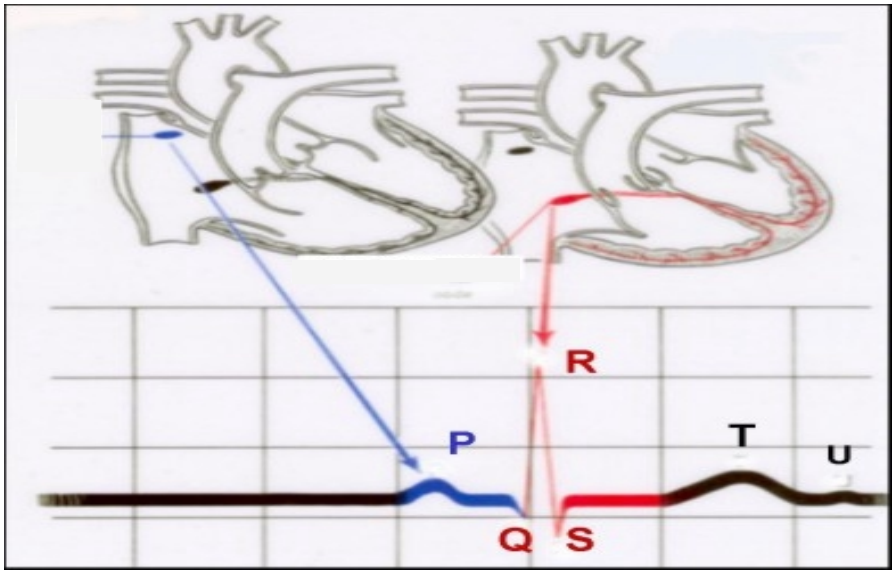

Figure 1. A normal ECG.

1.2.2 Pathologies

Any injury or arrhythmia in any part of the heart distorts one or more waves displayed graphically on an ECG, giving the doctor the notion that something is wrong [1]. The shape of each wave reflects a kind of pathology and / or a stage of evolution, as can be seen in the examples below. Figure 2 shows the case of ischemia, which is the blockage of some vein that irrigates the heart. In it, the T wave can be seen in an inversion process, which characterizes ischemia. After this stage, you can get a heart attack.

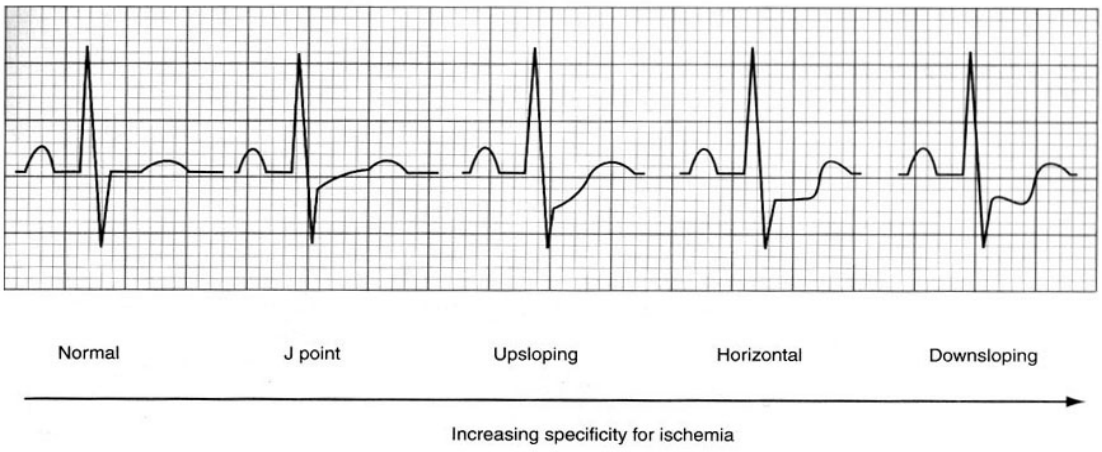

Figure 2. Stages of evolution of ischemia.

Figure 3 shows how the heart pulse can be presented in the case of a heart attack, and Figure 4 shows the arrhythmia process that occurs in some parts of the heart.

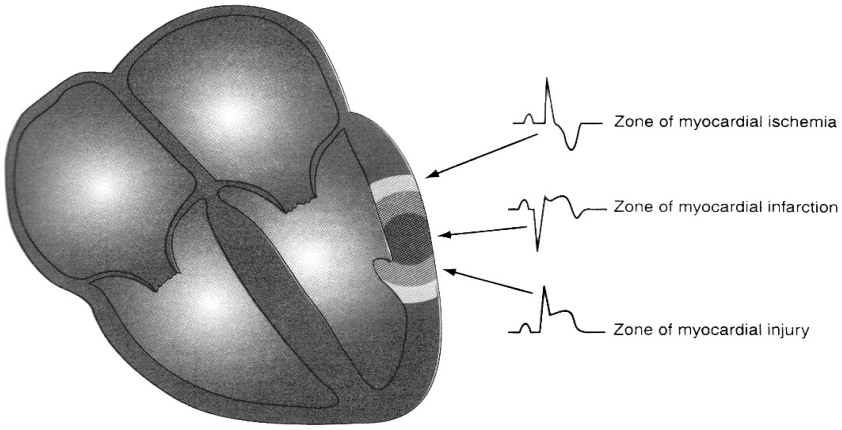

Figure 3. Myocardial infarction after ischemia.

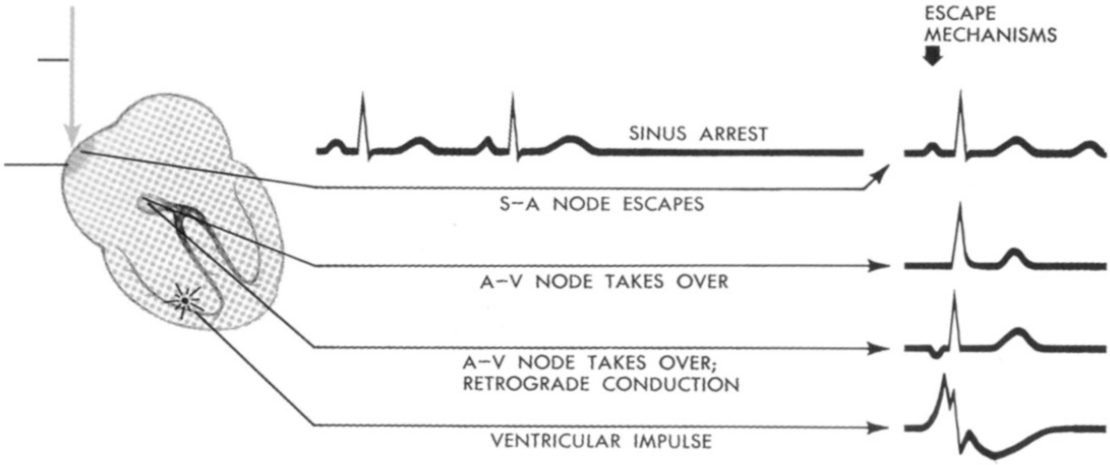

Figure 4. Arrhythmia in different points of the heart.

1.3 Problem Statement

Analyse a graph is a complex work, it requires a good filter and algorithms that can read a long array of floats like a unique feature. The idea here is transform this kind of array in a set of features and based on it to classify the ECG in normal or sick.

To do that it was used a toolkit called heartpy[2] that provides a lot of functions that handle with ECG very well and it extracts the measures that will be used to predict.

This process is summarized in figure 5.

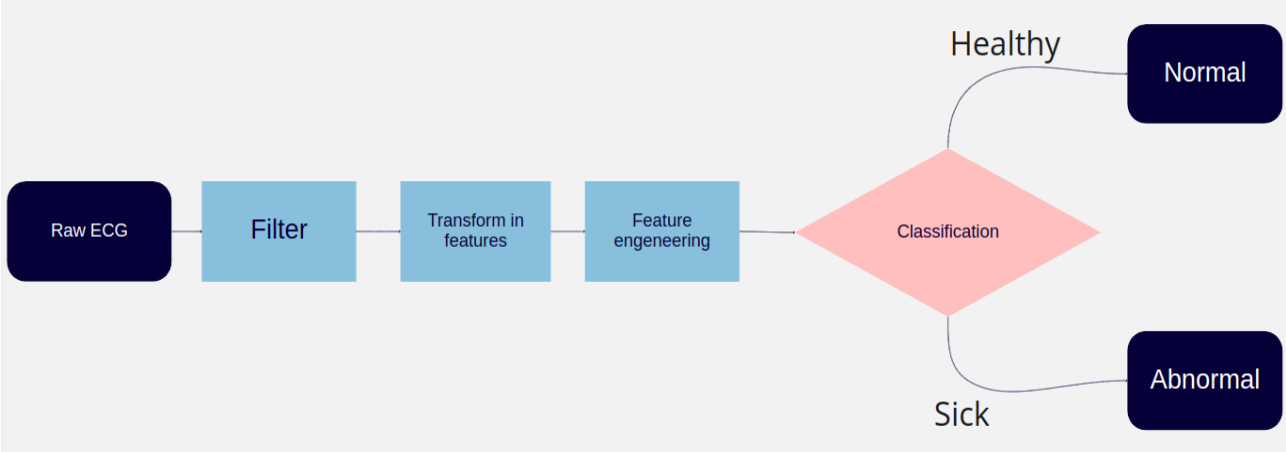

Figure 5. The process.

The raw data is acquired by download at Kaggle[3]. This dataset is divided in 4 files, 2 sets from the MIT-BIH Arrhythmia Dataset[4] and 2 from the PTB Diagnostic Database[5]. As PTB is divided in Normal and Abnormal classification, it will be the option. The MIT database is divided in a lot classes that is not in the plan of this work.

The heartpy has a function that filter noise from this dataset, as it can see in figures 6 and 7 (before and after filter, respectively). And, after that, it is served as input to other function that transform it in a set of features that interest to this project, these features are discribed below:

- BPM - heart rate (BPM), is calculated as the average beat-beat interval across the entire analysed signal (segment).
- IBI - interbeat interval.
- SDNN - standard deviation of RR intervals.
- SDSD - standard deviation of successive differences. See figure 8.
- RMSSD - root mean square of successive differences. See figure 8.
- PNN20 - proportion of successive differences above 20ms.
- PNN50 - proportion of successive differences above 50ms.
- HR_MAD - median absolute deviation of RR intervals.
- SD1 – standard deviation perpendicular to identity line (Poincaré parameters[7]).
- SD2 – standard deviation a long identtiy line.
- S – area of ellipse described by SD1 and SD2.
- SD1/SD2 – ratio.
- BREATHING RATE – that is the frequency with which the heart beats is strongly influenced by.

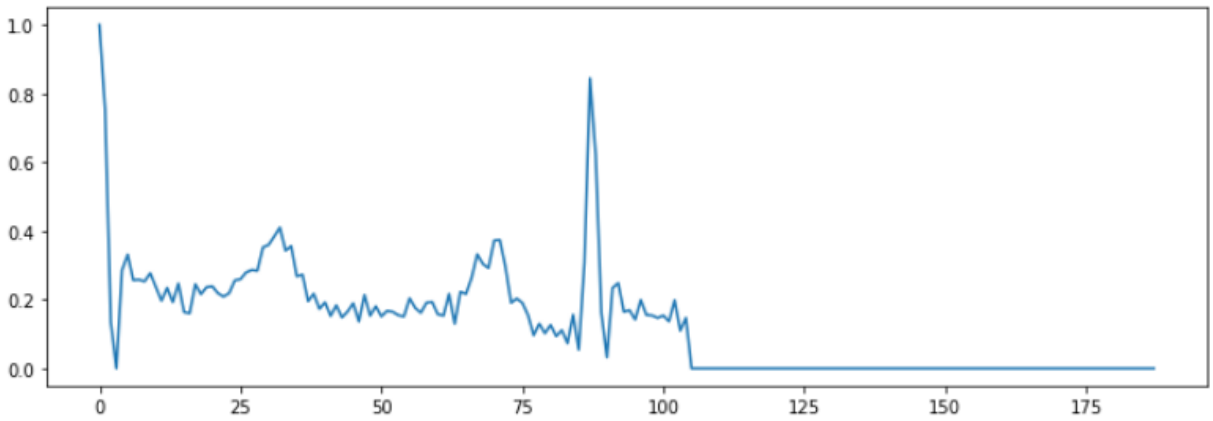

Figure 6. The graph with noise.

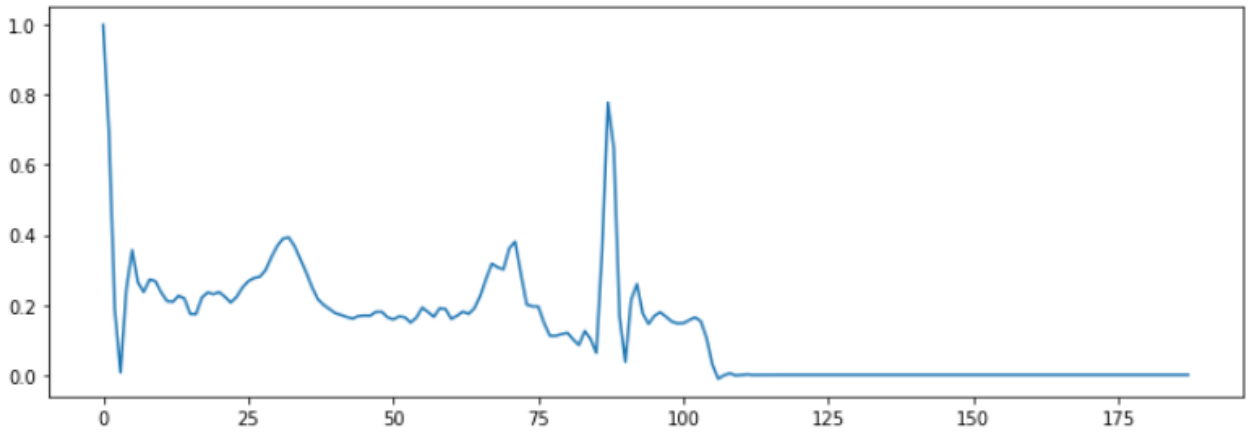

Figure 7. The graph after a soft filter.

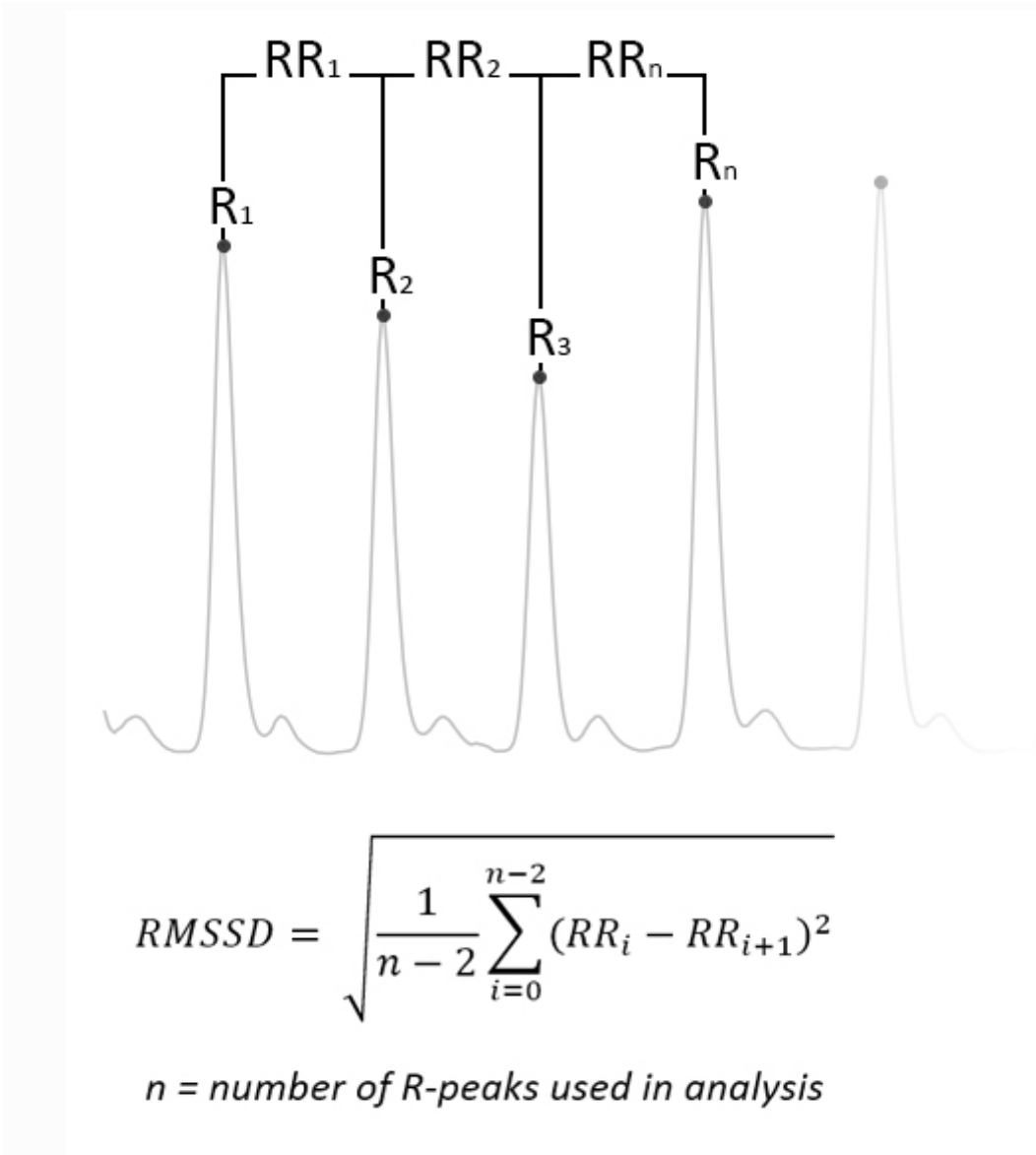

Figure 8. Image displaying the desired peak detection result, as well as the calculation of the RMSSD measure. The SDSD measure is the standard deviation between successive differences[6]

The figures 9 and 10 show how are the dataframe before and after transform, respectively. Even after the raw data pass through the filter, there are lines that were droped, because the heartpy could not converted them.

	0	1	2	3	4	5	6	7	8	9	...	178	179	180	181	182	183	184	185	186
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	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
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
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
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
5	0.925265	0.433352	0.073620	0.079197	0.136643	0.182934	0.182934	0.182376	0.196877	0.203569	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6	0.999144	0.774829	0.522260	0.250856	0.000000	0.066781	0.093322	0.144692	0.154110	0.152397	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
7	1.000000	0.871452	0.678353	0.357262	0.051196	0.000000	0.090707	0.142460	0.138564	0.185309	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8	1.000000	0.729630	0.583333	0.350926	0.279630	0.321296	0.267593	0.229630	0.218519	0.203704	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
9	0.932234	0.810440	0.627595	0.422161	0.263126	0.145910	0.109280	0.088523	0.079976	0.073565	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Figure 9. Raw data before filter, each line is an ECG.

	bpm	ibi	sdnn	sdsd	rmssd	pnn20	pnn50	hr_mad	sd1	sd2	s	sd1/sd2	breathingrate
0	322.285714	186.170213	180.851064	0	361.702128	1.0	1.0	180.851064	0.0	0.0	0.0	0.0	0.0
0	278.518519	215.425532	210.106383	0	420.212766	1.0	1.0	210.106383	0.0	0.0	0.0	0.0	0.0
0	5640.000000	10.638298	0.000000	--	0.000000	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
0	118.736842	505.319149	0.000000	--	0.000000	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
0	223.366337	268.617021	242.021277	0	484.042553	1.0	1.0	242.021277	0.0	0.0	0.0	0.0	0.0

Figure 10. Each ECG transformed in features.

The feature engeneering is applyed as next step. All these parameters are normalized by z-score represented by the formula:

$$y_t = \frac{y_t - \mu(y)}{\delta(y)}$$

The z-score garatees a mean iguals zero and standard deviation iguals 1, and it makes easy for the algorithms the comparison between features.

Following, the dataset is spllited into training, testing and validation sets it will be used in two algorithms. The choice of these algorithms will be explained at session Algorithms and Techniques.

2 Metrics

The accurancy of models will be measure using a confunsion matrix[8].

Confusion matrix returns the true positives(TP), true negatives(TN), false positives(FP) and false negatives(FN), with these measures, it is possible to extract a lot of others metrics that work like a list of rates of performance that allows to create a ranking and to compare algorithms more thoroughly.

Those other metrics are:

- Accuracy = (TP+TN)/Total
- AUC = Area under curve
- Precision = TP/(TP + FP)
- Recall = TP/(TP+FN)
- F1-Score = 2*Precision*Recall/(Precision+Recall)

The final benchmark reached was considered satisfatory, because all algorithms passed 80% of accurancy.

3 Analysis

1.3 Data exploration

The Kaggle dataset includes 14552 samples of ECG from Physionet's PTB Diagnostic Database with 2 categories in csv file.

After transform it, 1660 samples was dropped, because heartpy did not could convert them, so the rest of 12892 sample was used.

First, this dataset was splited in train and test 50-50% and the test dataset result was splited in 85-15% for test and validation, respectively.

The dataset is unbalanced as it can see in figure 11.

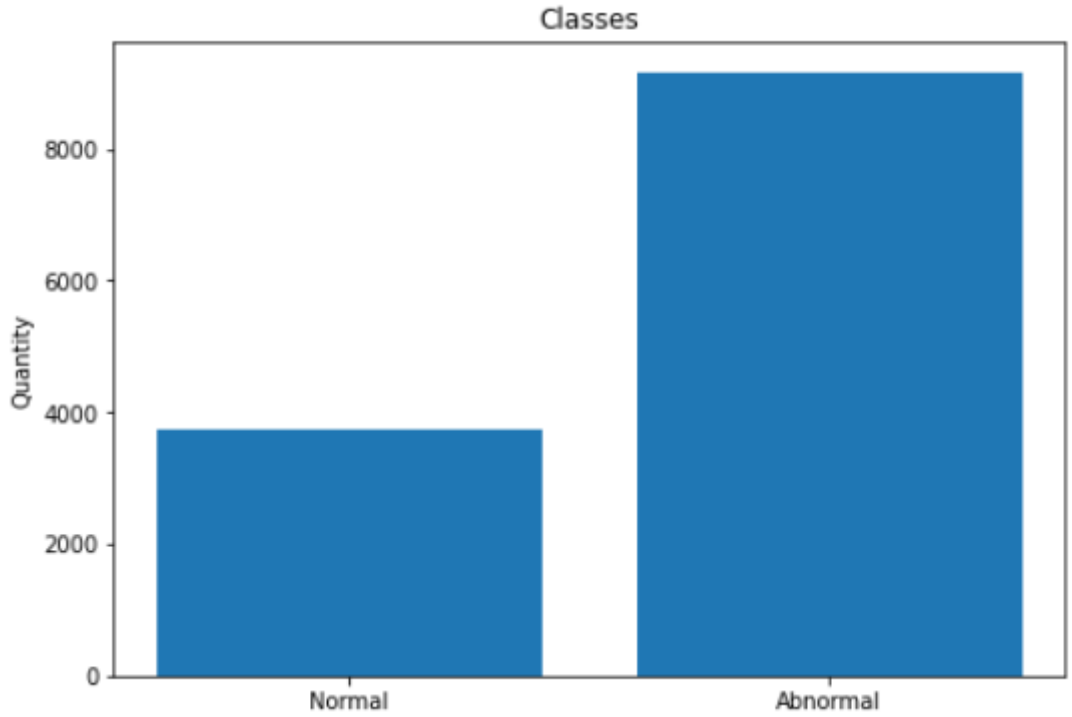

Figure 11.

Figure 12 bellow compares the graph of BPM and IBI features splited between normal and abnormal. It is possible to see that although the data is complicated, the algorithms can predict and find a pattern.

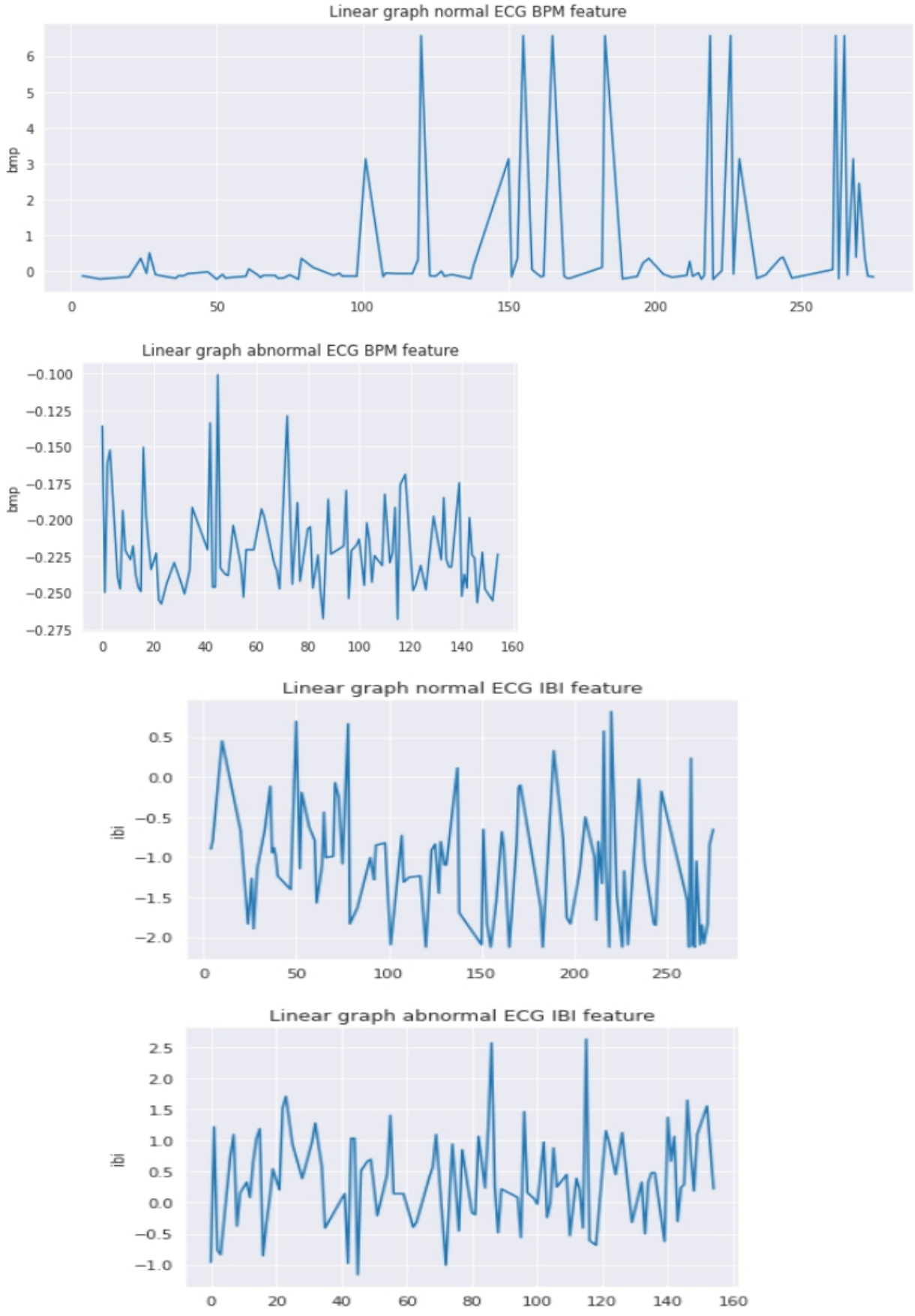
Linear graph normal ECG BPM feature
bmp
Linear graph abnormal ECG BPM feature
bmp
Linear graph normal ECG IBI feature
ibi
Linear graph abnormal ECG IBI feature
ibi

Figure 12.

4 Algorithms and techniques

4.1 Data preprocessing

It was possible to see in session above, that the dataset was filtered by heartpy function and after that it was extract features that will passed to the algorithms as input to training, testing and validation as intent to allow the models classify between normal and abnormal ECG.

The set of features was normalized using z-score to facilitate the classification of algorithms.

4.2 Implementation

Like the dataset is very unbalanced, it was chosen XGBoost[9] as first algorithm to classify this kind of data, because it can handle very well with this issue.

It was using PyCaret[10] to choose the better algorithm that matches with this kind of data and this algorithm was the CatBoost Classifier[11].

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
0	CatBoost Classifier	0.8852	0.9392	0.9385	0.9037	0.9207	0.7132	0.7153	1.9184
1	Gradient Boosting Classifier	0.8825	0.9372	0.9375	0.9012	0.9189	0.7060	0.7083	0.2467
2	Light Gradient Boosting Machine	0.8821	0.9391	0.9419	0.8975	0.9190	0.7027	0.7063	0.0835
3	Extreme Gradient Boosting	0.8819	0.9376	0.9403	0.8982	0.9187	0.7031	0.7060	0.2337
4	Ada Boost Classifier	0.8768	0.9273	0.9410	0.8918	0.9156	0.6879	0.6922	0.1134
5	Random Forest Classifier	0.8757	0.9212	0.9335	0.8960	0.9142	0.6886	0.6911	0.0229
6	Decision Tree Classifier	0.8743	0.8672	0.9294	0.8974	0.9130	0.6868	0.6886	0.0087
7	Extra Trees Classifier	0.8741	0.8995	0.9329	0.8945	0.9132	0.6844	0.6869	0.1093
8	Logistic Regression	0.8726	0.9252	0.9360	0.8904	0.9125	0.6784	0.6817	0.0674
9	K Neighbors Classifier	0.8726	0.9097	0.9219	0.9011	0.9113	0.6850	0.6860	0.0112
10	Linear Discriminant Analysis	0.8648	0.9235	0.9450	0.8749	0.9085	0.6509	0.6585	0.0223
11	Ridge Classifier	0.8630	0.0000	0.9522	0.8681	0.9081	0.6414	0.6523	0.0145

Figure 13. CatBoost Classifier was choosed by Pycaret.

"PyCaret is an open source low-code machine learning library in Python that aims to reduce the hypothesis to insights cycle time in a ML experiment. It enables data scientists to perform end-to-end experiments quickly and efficiently. In comparison with the other open source machine learning libraries, PyCaret is an alternate low-code library that can be used to perform complex machine learning tasks with only few lines of code. PyCaret is simple and easy to use. All the operations performed in PyCaret are automatically stored in a custom Pipeline that is fully orchestrated for deployment. PyCaret is essentially a Python wrapper around several machine learning libraries and frameworks such as scikit-learn, XGBoost, Microsoft LightGBM, spaCy and many more."[10]

CatBoost is an implementation of gradient boosting, which uses binary decision trees as base predictor. It is very good with categorical features and handle very well with unbalanced dataset and noisy data.

The software requirement for the implementation is as followed:

- Python >= 3.6
- numpy>=1.19.0
- pandas>=1.0.5
- pycaret>=2.1.2
- heartpy>=1.2.6
- sklearn>=0.19.1
- xgboost>=1.1.1

5 Refinement

5.1 Hyperparameters tuning

It was used the sagemaker's hyperparameters tuning to find the better ajust to XGBoost. The input for start the process was that parameters:

- max_depth: Maximum depth of a tree. Range between 3 and 12
- eta: Controls the learning rate. Range between 0.05 and 0.5
- min_child_weight: Minimum sum of weights of all observations required in a child node . Used to avoid overfitting as higher values prevent model to learn relations specific to the dataset. Range between 2 and 8.
- subsample: Corresponds to the fraction of observations (the rows) to subsample at each step. Range between 0.5 and 0.9
- gamma: Minimum loss reduction required to make a further partition on a leaf node of the tree. The larger gamma is, the more conservative the algorithm will be. Range between 0 and 10.
- colsample_bytree: corresponds to the fraction of features (the columns) to use. Range between 0.1 and 0.6.

The objective was configured to minimize the rmse(root mean square error) and after this process of tuning the better ajust found was:

- colsample_bytree = 0.5942246874135562
- eta = 0.1570389955204472
- gamma = 0
- max_depth = 11
- min_child_weight = 2
- subsample = 0.787644075752256

As the result, the confusion matrix presented was:

True Positive: 3821
False Positive: 608
False Negative 106
True Negative: 945

Metrics

Accuracy: 0.8697080291970803
AUC: 0.7907535314072945
Precision: 0.8627229622939715
Recall: 0.9730073847720907
F1-Score: 0.9145524174246051

5.1.1 XGBoost vs CatBoost Classifer

After that, CatBoost Classifier was implemented and the same dataset was given to it as input. That's the result:

True Positive: 3601
False Positive: 393
False Negative 246
True Negative: 1240

Metrics

Accuracy: 0.8833941605839416
AUC: 0.8476963543219511
Precision: 0.9016024036054081
Recall: 0.9360540681050169
F1-Score: 0.9185052926922587

XGBoost	CatBoost Classifier
True Positive: 3821	**True Positive: 3601**
False Positive: 608	False Positive: 393
False Negative 106	False Negative 246
True Negative: 945	**True Negative: 1240**
Accuracy: 0.8697080291970803	**Accuracy: 0.8833941605839416**
AUC: 0.7907535314072945	**AUC: 0.8476963543219511**
Precision: 0.8627229622939715	**Precision: 0.9016024036054081**
Recall: 0.9730073847720907	**Recall: 0.9360540681050169**
F1-Score: 0.9145524174246051	**F1-Score: 0.9185052926922587**

It is possible to see that CatBoost Classifier had a better performance because it missed less. But XGBoost still had an excellent performance.

5.2 Incremental training

To see what happen if only the features that is in top 5 of correlation do with the response of each algorithm, it was used a random forest[12] to detect which are those features and reprocess all steps did above.

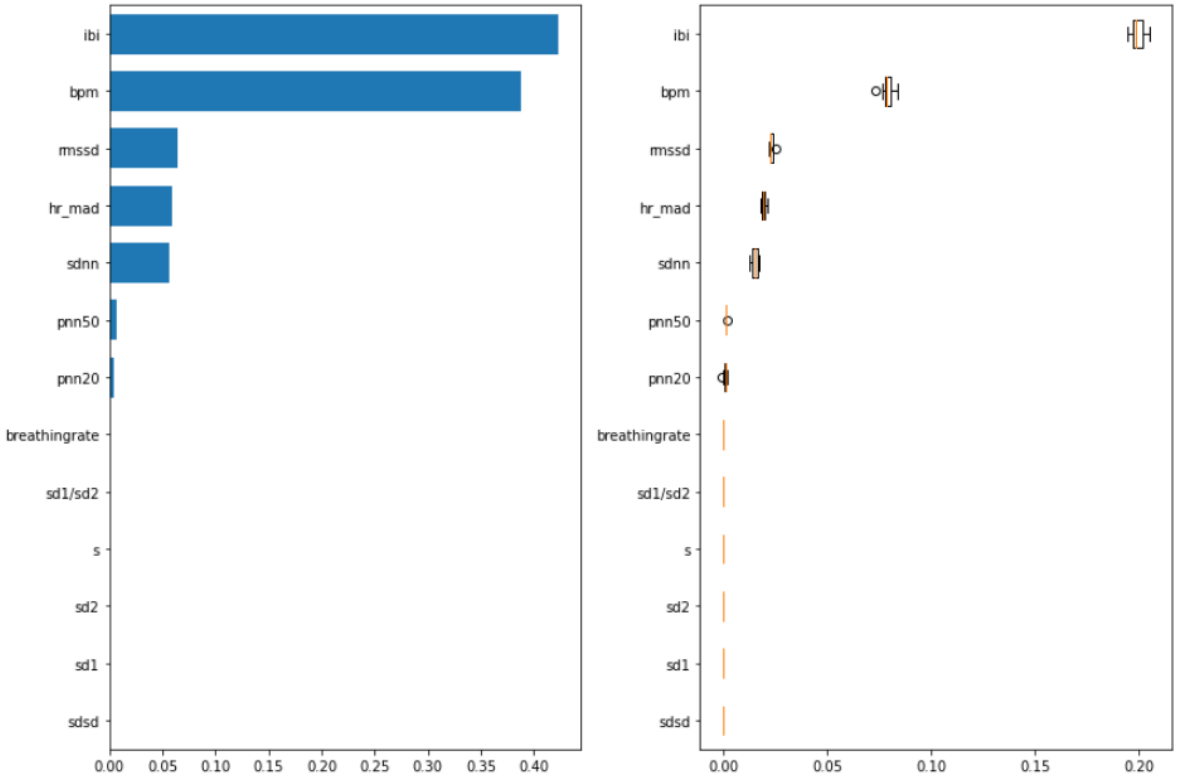

Figure 14. Most important features.

Figure 14 shows that the 5 most important featues are IBI, BPM, RMSSD, HR_MAD and SDNN.

The dataset was reprocessed with only this 5 columns and the XGBoost' s hyperparameters were processed:

- colsample_bytree = 0.5692556432570872
- eta = 0.26126842500652403
- gamma = 1
- max_depth = 7
- min_child_weight = 2
- subsample = 0.6747585047611748

The XGBoost's result:

True Positive: 3749
False Positive: 664

False Negative 98
True Negative: 969

Metrics

Accuracy: 0.8609489051094891
AUC: 0.7839560048779471
Precision: 0.8495354634035803
Recall: 0.9745256043670393
F1-Score: 0.9077481840193704

And CatBoost Classifier's result:

True Positive: 3605
False Positive: 399
False Negative 242
True Negative: 1234

Metrics

Accuracy: 0.8830291970802919
AUC: 0.8463791303329067
Precision: 0.9003496503496503
Recall: 0.9370938393553419
F1-Score: 0.9183543497643614

5.2.1 XGBoost v1 vs XGBoost v2 vs CatBoost v1 vs CatBoost v2

XGBoost V1	CatBoost V1	XGBoost V2	CatBoost V2
True Positive: 3821	**True Positive: 3601**	**True Positive: 3749**	**True Positive: 3605**
False Positive: 608	False Positive: 393	False Positive: 664	False Positive: 399
False Negative 106	False Negative 246	False Negative 98	False Negative 242
True Negative: 945	**True Negative: 1240**	**True Negative: 969**	**True Negative: 1234**
Accuracy: 0.8697080291970803	**Accuracy: 0.8833941605839416**	**Accuracy: 0.8609489051094891**	**Accuracy: 0.8830291970802919**
AUC: 0.7907535314072945	**AUC: 0.8476963543219511**	**AUC: 0.7839560048779471**	**AUC: 0.8463791303329067**
Precision: 0.8627229622939715	**Precision: 0.9016024036054081**	**Precision: 0.8495354634035803**	**Precision: 0.9003496503496503**
Recall: 0.9730073847720907	**Recall: 0.9360540681050169**	**Recall: 0.9745256043670393**	**Recall: 0.9370938393553419**
F1-Score: 0.9145524174246051	**F1-Score: 0.9185052926922587**	**F1-Score: 0.9077481840193704**	**F1-Score: 0.9183543497643614**

No algorithm increased significantly performance after the procedure of drop less relevant

features, so the information of all of them is important.

6 Conclusion

This project shows that is possible to classify ECGs by the features extract from the signal (since the begining this was a great doubt). As future works, it is possible to try other algoritms like Light Gradient Boosting Machine and/or unsupervised ones.

CatBoost Classifier presents a little bit better performance than XGBoost, but if possible, it's recommended deploy both in prodution to compares benchmarks.

Also it is also possible to conclude that not always removing less important features makes the algorithms perform better.

7 References

[1] Curso de eletrocardiografia básica. **Centro de Ciência da Saúde – CCS**. Universidade Estadual de Londrina. Disponível em: <http://www.ccs.uel.br/medicina/pbl/cardio/capitulo1.asp>. Acesso em: 9 de jul. 2005.

[2] HeartPy – Python Heart Rate Analysis Toolkit's documentation - https://python-heart-rate-analysis-toolkit.readthedocs.io/en/latest/

[3] ECG Heartbeat Cathegorization Dataset - https://www.kaggle.com/shayanfazeli/heartbeat

[4] MIT-BIH Arrhythmia Database - https://www.physionet.org/content/mitdb/1.0.0/

[5] PTB Diagnostic ECG Database - https://www.physionet.org/content/ptbdb/1.0.0/

[6] Heart Rate Analyse by heartpy - https://python-heart-rate-analysis-toolkit.readthedocs.io/en/latest/heartrateanalysis.html

[7] Shaffer, F., Ginsberg, J.P. (2017), An Overview of Heart Rate Variability Metrics and Norms.

[8] Simple guide to confusion matrix terminology - https://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/

[9] XGBoost: A Scalable Tree Boosting System - http://dmlc.cs.washington.edu/data/pdf/XGBoostArxiv.pdf

[10] PyCaret - https://pycaret.org/

[11] CatBoost: unbiased boosting with categorical features - https://arxiv.org/pdf/1706.09516.pdf

[12] Random forest - https://link.springer.com/content/pdf/10.1023/A:1010933404324.pdf