International Journal of Combinatorial Optimization Problems and Informatics, 14(3), Sep-Dec 2023, 3-15. ISSN: 2007-1558. Doi:10.61467/2007.1558.2023.v14i3.389

www.editada.org

Classification of Cardiac Arrhythmias

Miguel Ángel Ruiz Jaimes [1], *Jorge Alarcón Álvarez* [1], *Juan José Flore Sedano*[2], *Jorge A. Ruiz-Vanoye*[3], *Yadira Toledo-Navarro*[2], *Eric Simancas-Acevedo* [3], *Francisco Rafael Trejo-Macotela* [3], *Marco Antonio Vera-Jiménez* [3]

[1] Universidad Politécnica del Estado de Morelos, México.
[2] Centro Nacional de Investigación y Desarrollo Tecnológico, México.
[3] Universidad Politécnica de Pachuca, México.
E-mails: mruiz@upemor.edu.mx

Abstract. Cardiovascular diseases (CVD) are one of the leading causes of mortality worldwide. In this context, the design of accurate and efficient methods for the automated classification of electrocardiogram (ECG) signals is essential for clinical diagnosis, particularly in the detection of cardiac arrhythmias. The main objective of this project is to develop an automated classification system for cardiac arrhythmias using electrocardiogram signals. To carry out this project, the iterative cascade methodology was used, which allows modifications and improvements to be made in each development iteration. The interview technique was used to collect the project requirements, which allowed obtaining detailed and precise information on the needs and expectations of the client.
Keywords; electrocardiogram, Cardiovascular diseases.

Article Info
Received 23 Aug 2023
Accepted 23 Sep 2023

1 Introduction

Cardiovascular diseases (CVD) are one of the leading causes of mortality worldwide. In this context, the design of accurate and efficient methods for the automated classification of electrocardiogram (ECG) signals is essential for clinical diagnosis, particularly in the detection of cardiac arrhythmias. The present investigation addresses this challenge with the purpose of contributing to the improvement of medical procedures in the early identification of cardiac pathologies. According to the World Health Organization (WHO), CVDs accounted for approximately 30% of deaths worldwide in 2012. Within this context, more accessible and rapid diagnostic pathways are being explored that can benefit both low-income countries low and medium as well as those with greater resources (WHO, n.d.; Pressman, 2002; Bass, Clements, & Kazman, 2003). In this sense, the automatic classification of cardiac arrhythmias from ECG signals emerges as a potential solution. The main objective of this project is to develop an automated classification system for cardiac arrhythmias using electrocardiogram signals. Cardiovascular diseases represent one of the main causes of death worldwide, so having fast and accurate diagnostic methods is essential. To carry out this project, the iterative cascade methodology was used, which allows modifications and improvements to be made in each development iteration. The interview technique was used to collect the project requirements, which allowed obtaining detailed and precise information on the needs and expectations of the client.

The requirements analysis was carried out through structured interviews, formulating specific questions to obtain the necessary information (Clínica Universidad de Navarra, n.d.; Bass, Clements, & Kazman, 2003). Functional, non-functional, interface, quality, evolution, project, support and design restrictions were established. In addition, support requirements were established, such as having adequate computer equipment and writing the system. The system was developed using the MIT-BIH Arrhythmia Database and processing the signals through Wavelet transforms. It was implemented using the SciKit-Learn library for the classifiers, following the established design restrictions. Functionality tests were carried out and the library was published in the Python Package Index (PyPI) (The PyData Development Team, n.d.; NumPy Project, n.d.; SciPy Developers, n.d.). The project was carried out in a period of 4 months, complying with the established deadlines. It was possible to develop a library that allows the classification of cardiac arrhythmias from electrocardiogram signals, with a web graphic interface and the possibility of extracting signals from the MIT-BIH Arrhythmia Database. It is suggested to carry out additional experiments with different characteristics and databases in future works.

2 Requirements

The requirements analysis was carried out through structured interviews with the client. Specific questions were asked to obtain the necessary information and detailed and accurate information was collected on the needs and expectations of the client. In addition, different types of requirements were established, such as functional, non-functional, interface, quality, evolution, project and support. These requirements were identified and documented to ensure that all project needs were met.

The interview as a requirements gathering technique was selected due to its ability to collect information in a structured way and create an environment of trust with the client. The result of the requirements analysis was the identification and documentation of the project requirements, which were divided into different categories, such as:
Functional requirements

- FR.1: Signal extraction from the MIT-BIH Arrhythmia Database.
- FR.2: Signal preprocessing using wavelet transforms.
- FR.3: Classification of cardiac arrhythmias using different machine learning algorithms.
- FR.4: Evaluation of the performance measures of the classifiers.

Interface requirements:

- IR.1:System user interface design.
- IR.2: Detailed documentation of the implemented procedures.

Quality requirements:

- QR.1: Compliance with minimum software specifications.
- QR.2: Installation of the necessary dependencies for the execution of the software.

Design requirements:

- DR.1: Exclusive use of the classifiers defined in the SciKit-Learn library.
- DR.2: Exclusive use of the wavelet families defined in the PyWavelets library.
- DR.3: Processing of specific signals of the electrocardiogram.
- DR.4: Processing of existing electrocardiogram signal records in the database.

Detailed and accurate information on the needs and expectations of the client was collected, using the technique of structured interviews.

3 Design

The system design was done conceptually using UML diagrams. The chosen architecture was described. The user interfaces of the system were also designed and the data manipulated through data dictionaries and entity-relationship diagrams were explained. Software architecture, a fundamental guide for the organization and operation of computer systems, transcends the mere structure of the source code. According to the definition of IEEE 1471, this architecture is made up of the intrinsic organization of the system and its components, their interactions and the principles that guide their design and evolution. This translates into a broader concept than a simple description, since it covers the fundamental and essential aspects of the system.

The Pipe and Filter architectural pattern was used for information processing, where the data passes through different filters until reaching the final result. This design was chosen due to the nature of the project, which involves electrocardiogram signal processing and classification of cardiac arrhythmias.

In this work, data will be extracted from a database, Daubechies wavelet transforms will be applied to preprocess the signals, and various classifiers will be used to analyze arrhythmias. The structure of this process resembles the Pipe and Filter pattern, where information flows through processing stages until reaching a final result. The choice of this architectural pattern underlines its suitability for this project. Figure 1 shows the Pipe and Filter architecture.

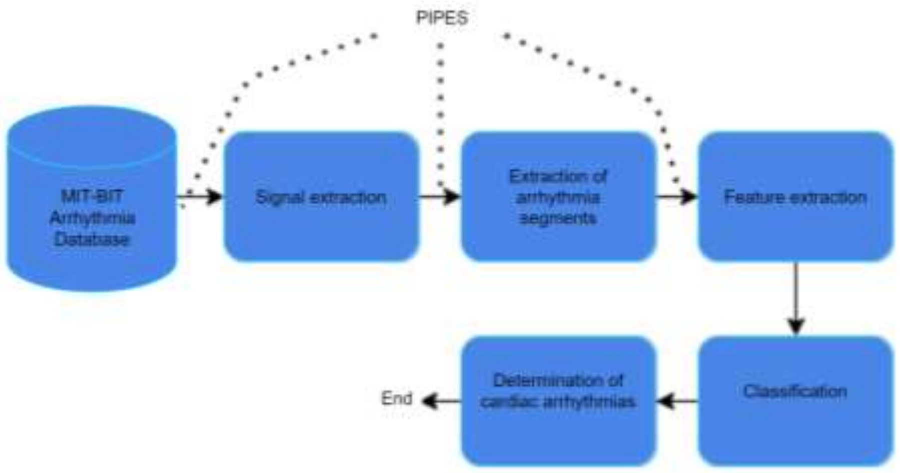
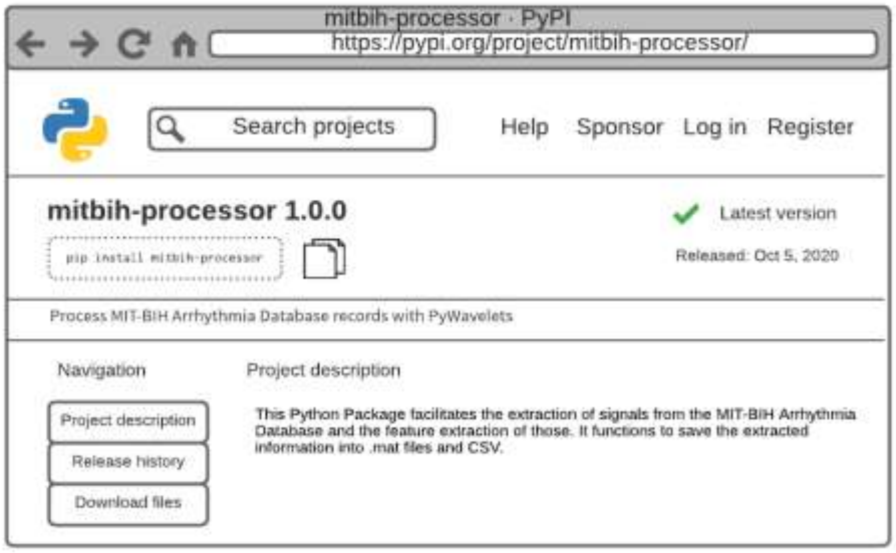

PIPES
MIT-BIT Arrhythmia Database
Signal extraction
Extraction of arrhythmia segments
Feature extraction
Classification
Determination of cardiac arrhythmias
End

Figure 1. Architecture Pipe Filter of the project

The system design was done conceptually using UML diagrams. The chosen architecture was described. The user interfaces of the system were also designed and the data manipulated through data dictionaries and entity-relationship diagrams were explained. Software architecture, a fundamental guide for the organization and operation of computer systems, transcends the mere structure of the source code. According to the definition of IEEE 1471, this architecture is made up of the intrinsic organization of the system and its components, their interactions and the principles that guide their design and evolution. This translates into a broader concept than a simple description, since it covers the fundamental and essential aspects of the system.

The Pipe and Filter architectural pattern was used for information processing, where the data passes through different filters until reaching the final result. This design was chosen due to the nature of the project, which involves electrocardiogram signal processing and classification of cardiac arrhythmias. In this work, data will be extracted from a database, Daubechies wavelet transforms will be applied to preprocess the signals, and various classifiers will be used to analyze arrhythmias. The structure of this process resembles the Pipe and Filter pattern, where information flows through processing stages until reaching a final result. The choice of this architectural pattern underlines its suitability for this project. Figure 2 shows the Pipe and Filter architecture.



mitbih-processor 1.0.0

```
pip install mitbih-processor
```

Latest version

Released: Oct 5, 2020

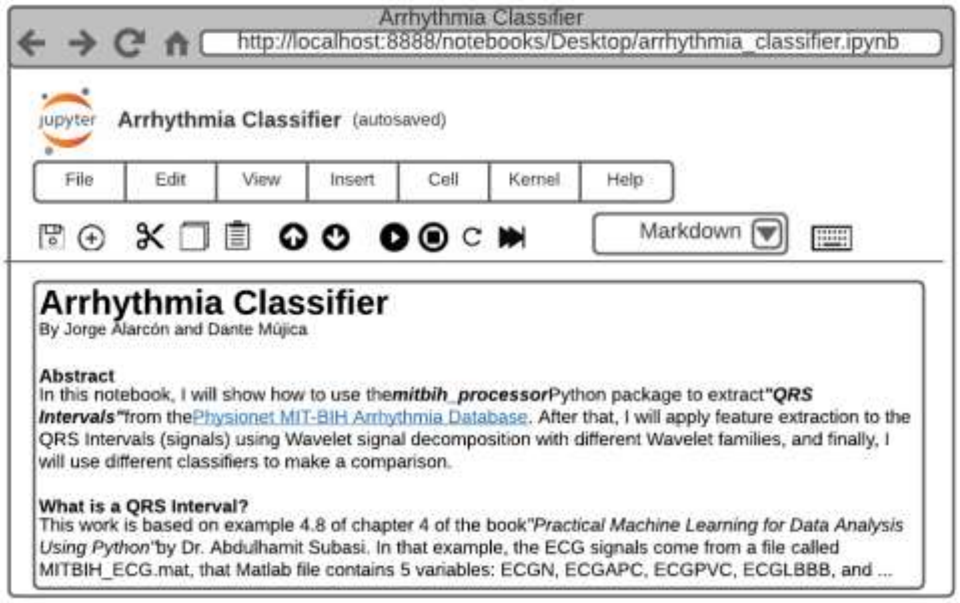
Process MIT-BIH Arrhythmia Database records with PyWavelets

Project description

This Python Package facilitates the extraction of signals from the MIT-BIH Arrhythmia Database and the feature extraction of those. It functions to save the extracted information into .mat files and CSV.

Figure 2. mitbih-processor library page interface

The signal processing was used in conjunction with SciKit-Learn and Jupyter Notebook to obtain the classifiers, in other words Jupyter contains the parts of extraction of cardiac arrhythmia signals from the database, storage of the extracted signals, application of a algorithm for feature extraction, storage of extracted features, training of classification algorithms, testing of classifiers and comparison between different classification algorithms. All the above procedures can be done with mitbih-processor, except for the classification part. This structure can be seen in figure 3.


Arrhythmia Classifier
http://localhost:8888/notebooks/Desktop/arrhythmia_classifier.ipynb
jupyter
Arrhythmia Classifier (autosaved)
File
Edit
View
Insert
Cell
Kernel
Help
Markdown

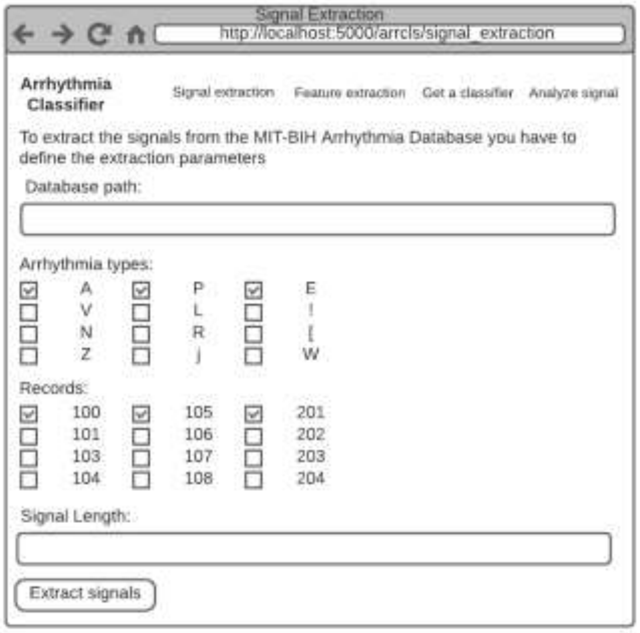
Arrhythmia Classifier
By Jorge Alarcón and Dante Mújica

Abstract
In this notebook, I will show how to use the***mitbih_processor***Python package to extract***"QRS Intervals"***from thePhysionet MIT-BIH Arrhythmia Database. After that, I will apply feature extraction to the QRS Intervals (signals) using Wavelet signal decomposition with different Wavelet families, and finally, I will use different classifiers to make a comparison.

What is a QRS Interval?
This work is based on example 4.8 of chapter 4 of the book*"Practical Machine Learning for Data Analysis Using Python"*by Dr. Abdulhamit Subasi. In that example, the ECG signals come from a file called MITBIH_ECG.mat, that Matlab file contains 5 variables: ECGN, ECGAPC, ECGPVC, ECGLBBB, and ...

Figure 3. Jupyter Notebook interface where the cardiac arrhythmia classifier will be developed.

The library's web client will have four windows, one for each main functionality, which are extraction of signals from the database, extraction of signal characteristics, training and obtaining a classifier, and signal analyzer. In the design of figure 4 you can see the diagram corresponding to the extraction of signals.



To extract the signals from the MIT-BIH Arrhythmia Database you have to define the extraction parameters

Database path:

Arrhythmia types:

✓	A	✓	P	✓	E
☐	V	☐	L	☐	!
☐	N	☐	R	☐	[
☐	Z	☐	j	☐	W

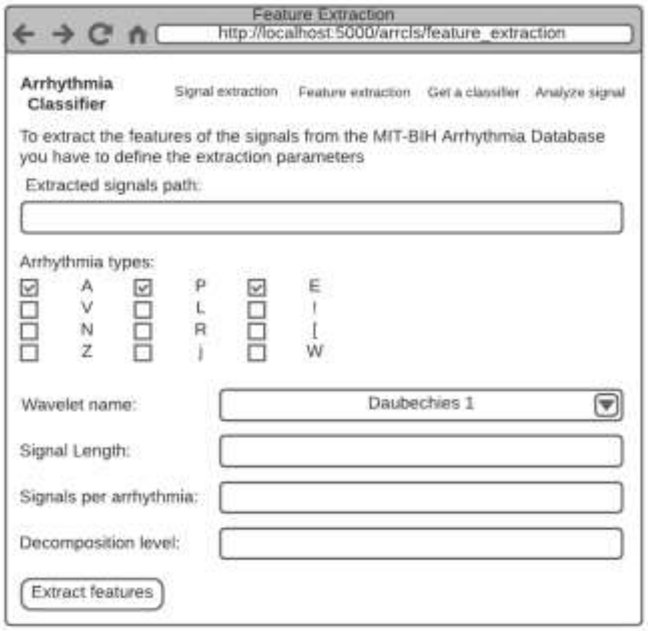
Records:

✓	100	✓	105	✓	201
☐	101	☐	106	☐	202
☐	103	☐	107	☐	203
☐	104	☐	108	☐	204

Signal Length:

Figure 4. Design of the signal extraction window of the web client.

In the design of Figure 5 you can see the diagram that corresponds to the signal characteristics extraction window.


Feature Extraction

http://localhost:5000/arrcls/feature_extraction

Arrhythmia Classifier

Signal extraction | Feature extraction | Get a classifier | Analyze signal

To extract the features of the signals from the MIT-BIH Arrhythmia Database you have to define the extraction parameters

Extracted signals path:

Arrhythmia types:

✓	A	✓	P	✓	E
☐	V	☐	L	☐	!
☐	N	☐	R	☐	[
☐	Z	☐	j	☐	W

Wavelet name: Daubechies 1

Signal Length:

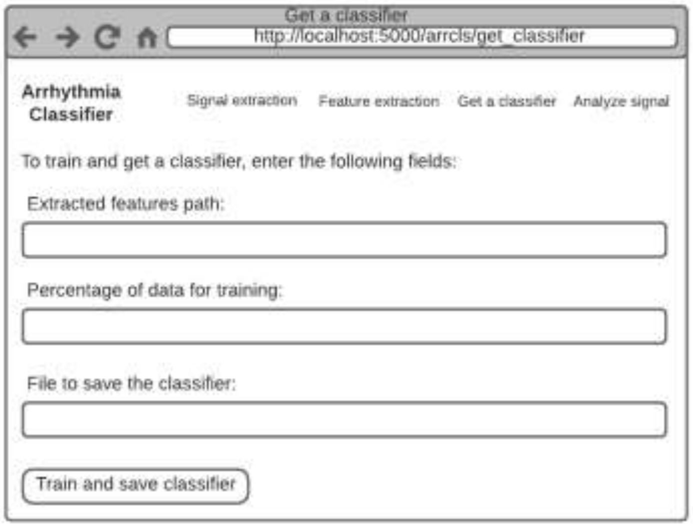
Signals per arrhythmia:

Decomposition level:

Extract features

Figure 5. Design of the feature extraction window of the web client signals.

In the design of Figure 6 you can see the window that corresponds to the training and obtaining of a classifier.


Get a classifier

http://localhost:5000/arrcls/get_classifier

Arrhythmia Classifier

Signal extraction | Feature extraction | Get a classifier | Analyze signal

To train and get a classifier, enter the following fields:

Extracted features path:

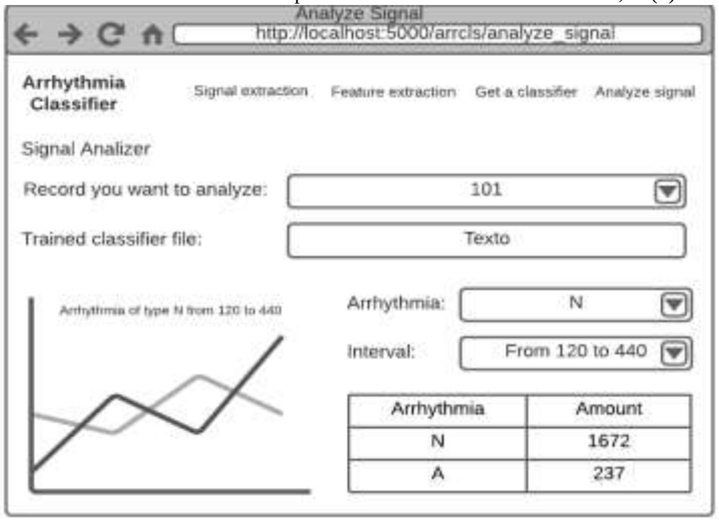
Percentage of data for training:

File to save the classifier:

Train and save classifier

Figure 6. Design of the training window of the web client classifier.

The design of Figure 7 shows the window corresponding to the electrocardiogram signal analyzer.



Arrhythmia	Amount
N	1672
A	237

Figure 7. Design of the signal analyzer window of the web client.

Two databases will be used in this project: one will be Physionet's MIT-BIH Arrhythmia Database and the other will be the result of the information extracted from the first. The MIT-BIH Arrhythmia Database contains 48 records, each of these records has the age of the patient, the name of the record, the age of the patient, and the electrocardiogram signals. A deliverable of this project corresponds to the data extracted from the aforementioned database. These extracted data will be stored in a Matlab binary file, in these files values are usually stored by key and value, in this case the key would be the type of cardiac arrhythmia and the value would be a matrix containing electrocardiogram signals for each of its rows. If you wanted to represent the MIT-BIH Arrhythmia Database and the extracted signals in an entity relationship diagram, it would look something like Figure 8.

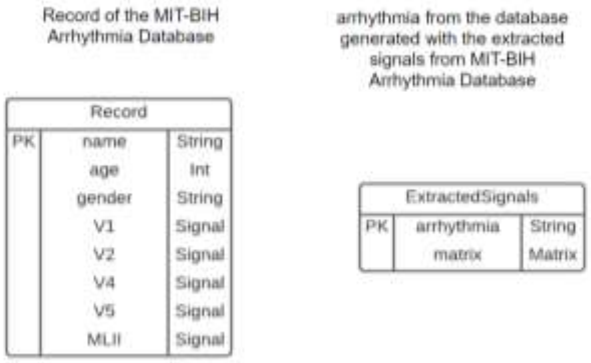

Record of the MIT-BIH Arrhythmia Database
arrhythmia from the database generated with the extracted signals from MIT-BIH Arrhythmia Database
Record
PK
name String
age Int
gender String
V1 Signal
V2 Signal
V4 Signal
V5 Signal
MLII Signal
ExtractedSignals
PK
arrhythmia String
matrix Matrix

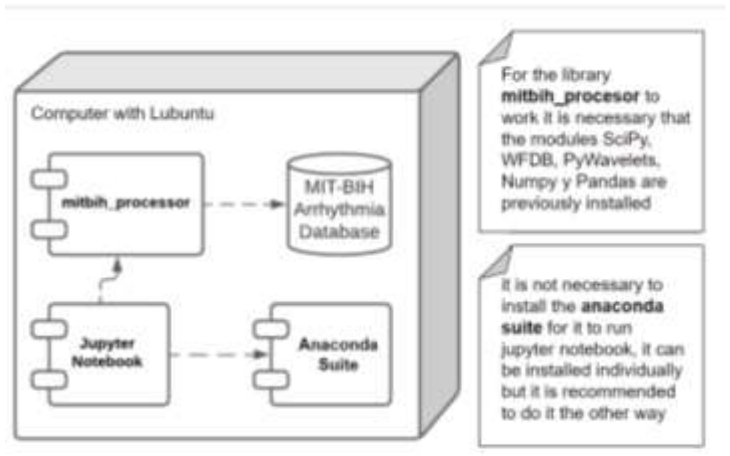
Figure 8. Representation in entity relationship diagram of the project databases.

Ultimately, this research explores the synergy between software architecture and cardiac signal analysis, showing how the Pipe and Filter structure finds application in the sequence of processes that culminate in the identification and classification of arrhythmias. This approach not only highlights the relevance of architecture in medical technology, but also highlights how architectural patterns can shape and optimize computing solutions for crucial medical challenges.

4 Implementatión

The implementation of the system was carried out following an agile methodology, using the Python programming language and various libraries and tools. The Jupyter Notebook development environment was used to write and execute the code, and the Numpy, Pandas, WFDB, SciPy and PyWavelets libraries were used for signal processing and feature extraction.

The system was implemented following a modular architecture, where each component was in charge of a specific task. Modules were developed for the extraction of signals from the MIT-BIH Arrhythmia Database, the preprocessing of the signals using wavelet transforms, the classification of cardiac arrhythmias using different machine learning algorithms, and the evaluation of performance measures. of the classifiers. In Figure 9 you can see the deployment diagram of the developed system.



For the library **mitbih_procesor** to work it is necessary that the modules SciPy, WFDB, PyWavelets, Numpy y Pandas are previously installed

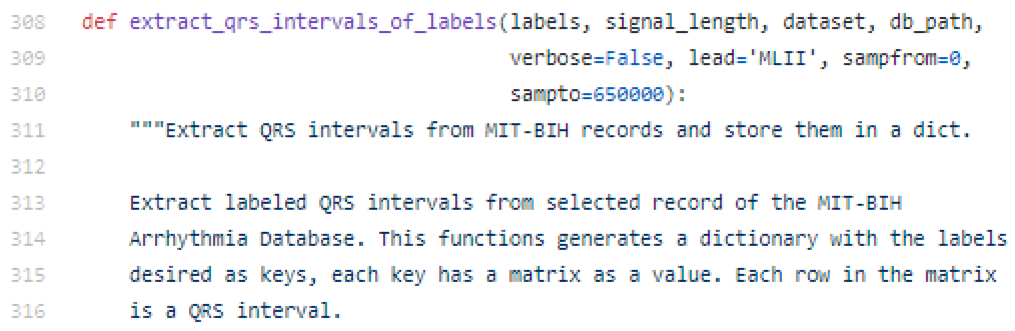
It is not necessary to install the **anaconda suite** for it to run jupyter notebook, it can be installed individually but it is recommended to do it the other way

Figure 9. Deployment diagram of the developed system.

Unit tests were carried out to verify the correct functioning of each component of the system individually. Specific test cases were written for each implemented function or method, with the aim of evaluating their behavior and ensuring that they met the established requirements. During unit tests, it was verified that the signal extraction, preprocessing, classification and performance evaluation functions will work correctly. Specific test data was used for each function and the obtained results were compared with the expected results.

In the nominal functional requirement 1 it is indicated that the electrocardiogram signals will be extracted from the MIT-BIH Arrhythmia Database, this process is carried out in several functions, however, the function presented in Figure IV.6 involves all of these parts. Nominal Functional Requirement 2 states that all information retrieved from the database will be stored in a Matlab file. This functionality can be seen in the X-image function, where the retrieved electrocardiogram signals are stored in a Matlab file.

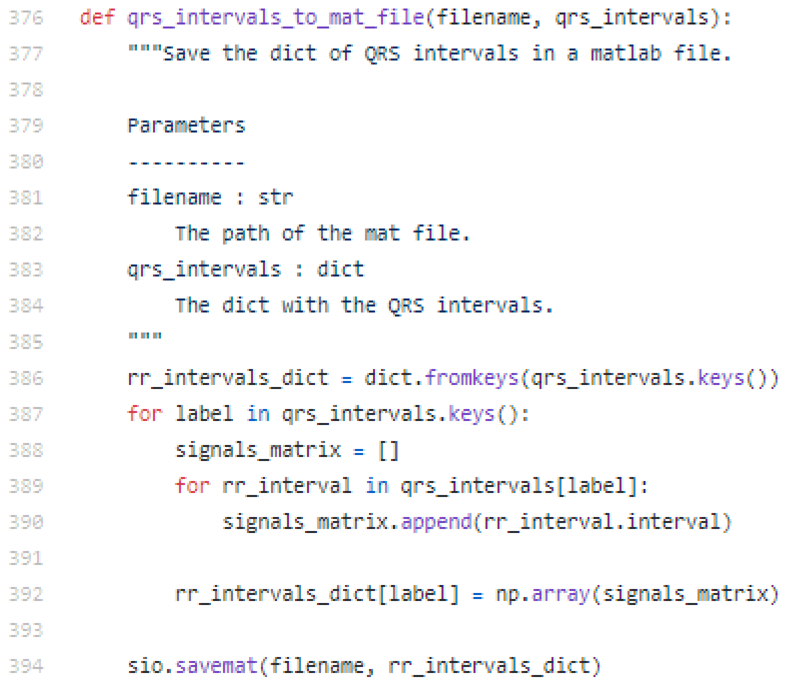
The nominal functional requirement 3 mentions that the signals will be processed with wavelet transforms for the extraction of features, all this processing is carried out in the image function 10, where these features are extracted and stored in a matrix.



```python
def extract_qrs_intervals_of_labels(labels, signal_length, dataset, db_path,
                                    verbose=False, lead='MLII', sampfrom=0,
                                    sampto=650000):
    """Extract QRS intervals from MIT-BIH records and store them in a dict.

    Extract labeled QRS intervals from selected record of the MIT-BIH
    Arrhythmia Database. This functions generates a dictionary with the labels
    desired as keys, each key has a matrix as a value. Each row in the matrix
    is a QRS interval.
```

Figure 10. Segment of the "extract_qrs_intervals_of_labels" function, where the electrocardiogram signals are extracted from the database.

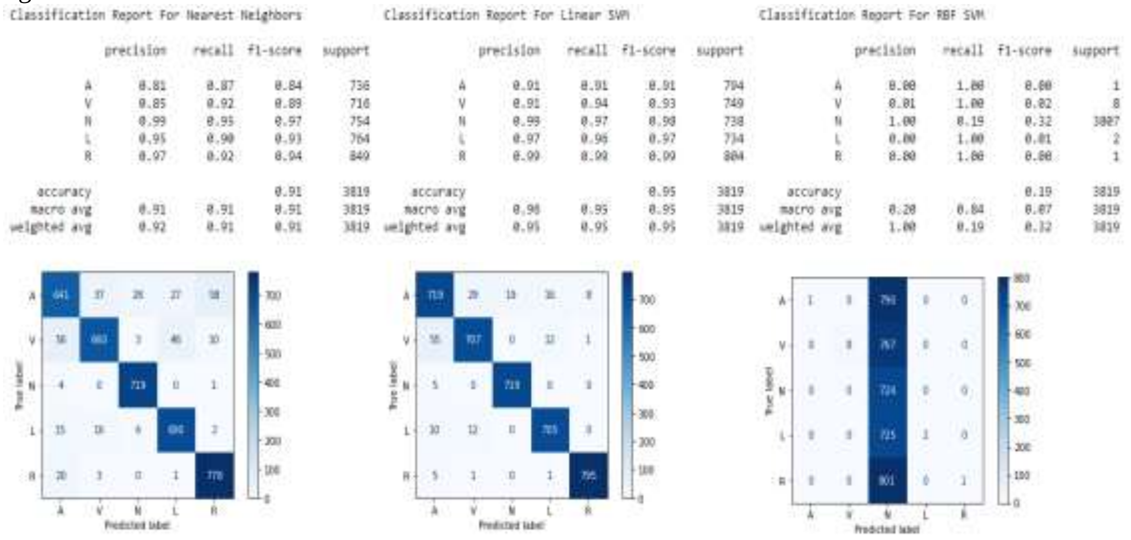
The classification of cardiac arrhythmias is a process that is carried out in the nominal functional requirement 4, in the program there are several sections that are in charge of the classification, one of them is the one in image 11, this function receives a complete electrocardiogram signal, a previously trained classifier, etc. It then uses the classifier to process segments of the signal, determining whether they are normal heartbeats or some type of arrhythmia.



```python
def qrs_intervals_to_mat_file(filename, qrs_intervals):
    """Save the dict of QRS intervals in a matlab file.

    Parameters
    ----------
    filename : str
        The path of the mat file.
    qrs_intervals : dict
        The dict with the QRS intervals.
    """
    rr_intervals_dict = dict.fromkeys(qrs_intervals.keys())
    for label in qrs_intervals.keys():
        signals_matrix = []
        for rr_interval in qrs_intervals[label]:
            signals_matrix.append(rr_interval.interval)

        rr_intervals_dict[label] = np.array(signals_matrix)

    sio.savemat(filename, rr_intervals_dict)
```

Figure 11. Segment of the "qrs_intervals_to_mat_file" function, where the recovered signals are stored.

In nominal functional requirement 5 it is mentioned that a comparison will be made between the different classification algorithms, this comparison is available in the Jupyter Notebook of the project repository, it shows the confusion matrices of each classifier, and some of the others. classification metrics mentioned in that requirement. In image X you can see a fragment of the Notebook corresponding to what was mentioned



```
Classification Report For Nearest Neighbors

              precision    recall  f1-score   support

           A       0.81      0.87      0.84       736
           V       0.85      0.92      0.89       716
           N       0.99      0.95      0.97       754
           L       0.95      0.90      0.93       764
           R       0.97      0.92      0.94       849

    accuracy                           0.91      3819
   macro avg       0.91      0.91      0.91      3819
weighted avg       0.92      0.91      0.91      3819
```

```
Classification Report For Linear SVM

              precision    recall  f1-score   support

           A       0.91      0.91      0.91       794
           V       0.91      0.94      0.93       749
           N       0.99      0.97      0.98       738
           L       0.97      0.96      0.97       734
           R       0.99      0.98      0.99       804

    accuracy                           0.95      3819
   macro avg       0.96      0.95      0.95      3819
weighted avg       0.95      0.95      0.95      3819
```

```
Classification Report For RBF SVM

              precision    recall  f1-score   support

           A       0.00      1.00      0.00         1
           V       0.01      1.00      0.02         8
           N       1.00      0.19      0.32      3807
           L       0.00      1.00      0.01         2
           R       0.00      1.00      0.00         1

    accuracy                           0.19      3819
   macro avg       0.20      0.84      0.07      3819
weighted avg       1.00      0.19      0.32      3819
```

Figure 12. Fragment of the Jupyter Notebook of the project, where the confusion matrices of 3 classifiers used and their metrics are presented.

5 Software testing

This section focuses on describing the testing process carried out in the project. This chapter is essential to guarantee the quality and proper functioning of the developed system. Through the application of a test plan that consists of defining an environment where the system will be tested, defining the database with which the tests will be executed and executing the planned tests under the established conditions. The following describes the conditions of software, hardware, network and number of users in which the system will be tested:

Software
The system was executed on a computer with Windows 10 Home Single Language, the Python interpreter that will be used will be that of Anaconda, more specifically in version 3.8, said interpreter will be used for the execution of the code.

Hardware
The system ran on a Dell Inspiron 15 5000 series computer, its hardware specifications are as follows:

- Intel(R) Core(TM) Processor i7-7500U CPU @ 2.70 GHz 2.90 GHz
- 64-bit operating system, x64 processor
- 8.00 GB RAM memory
- 1 TB HDD, Kingston Brand

The database used (MIT-BIH Arrhythmia Database) is not relational but a file directory. Table 5.1 lists the records that were used for training the cardiac arrhythmia classifier.

Table 1 Technologies used in the development of the project.

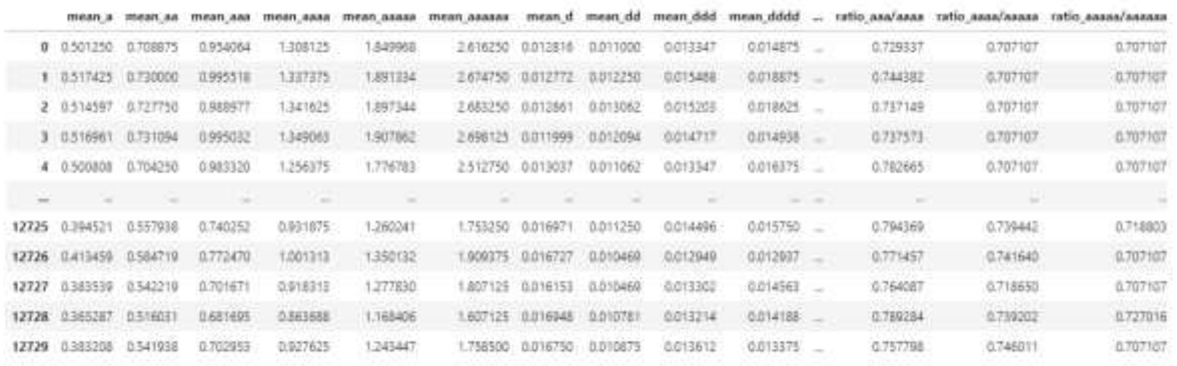
record name	Canal	Start sample number	End sample number	Length of the extracted signals	Arrhythmias that will be retrieved from said record
100	MLII	0	650,000	320	Normal beats, right and left bundle branch blocks, premature atrial beats, and premature ventricular contractions
101	MLII	0	650,000	320	Normal beats, right and left bundle branch blocks, premature atrial beats, and premature ventricular contractions
103	MLII	0	650,000	320	Normal beats, right and left bundle branch blocks, premature atrial beats, and premature ventricular contractions
105	MLII	0	650,000	320	Normal beats, right and left bundle branch blocks, premature atrial beats, and premature ventricular contractions
106	MLII	0	650,000	320	Normal beats, right and left bundle branch blocks, premature atrial beats, and premature ventricular contractions
108	MLII	0	650,000	320	Normal beats, right and left bundle branch blocks, premature atrial beats, and premature ventricular contractions
109	MLII	0	650,000	320	Normal beats, right and left bundle branch blocks, premature atrial beats, and premature ventricular contractions

111	MLII	0	650,000	320	Normal beats, right and left bundle branch blocks, premature atrial beats, and premature ventricular contractions
112	MLII	0	650,000	320	Normal beats, right and left bundle branch blocks, premature atrial beats, and premature ventricular contractions
113	MLII	0	650,000	320	Normal beats, right and left bundle branch blocks, premature atrial beats, and premature ventricular contractions
114	MLII	0	650,000	320	Normal beats, right and left bundle branch blocks, premature atrial beats, and premature ventricular contractions
115	MLII	0	650,000	320	Normal beats, right and left bundle branch blocks, premature atrial beats, and premature ventricular contractions
116	MLII	0	650,000	320	Normal beats, right and left bundle branch blocks, premature atrial beats, and premature ventricular contractions
117	MLII	0	650,000	320	Normal beats, right and left bundle branch blocks, premature atrial beats, and premature ventricular contractions
118	MLII	0	650,000	320	Normal beats, right and left bundle branch blocks, premature atrial beats, and premature ventricular contractions

The execution of the test plan was carried out according to the formats established for each nominal functional requirement. Specific tests were performed for each of the requirements, such as the extraction of signals from the MIT-BIH Arrhythmia Database, the analysis of an electrocardiogram signal, the classification of the data using various types of machine learning classifiers, among others. others. Each test was executed by the designated manager, who recorded the date and time of execution, as well as the results obtained. It was evaluated if the test was carried out satisfactorily and corrective actions were taken if necessary. In general, a systematic approach was followed and the general data of each test, the requirement to be tested, the performer, the date and time of execution, and the evaluator were documented. This allowed a detailed record of the execution of the test plan to be kept.

GENERAL DATA OF THE TEST
Requirement to try: FN1. Extraction of information from electrocardiogram signals from the MIT-BIH Arrhythmia Database
Executor: Jorge Alarcón Álvarez
Date and Time: 10/30/2020, 08:30 pm
Evaluator: Dr. Miguel Ángel Ruíz Jaimes
ACTIONS DONE
1. The MIT-BIH Arrhythmia Database path was entered.
2. The type of signal to be extracted (MLII) was selected.
3. The range of samples to be extracted was established (from 0 to 650,000).
4. The number of signals to be extracted (320) was specified.
RESULTS OBTAINED
The system showed the message that the signals were extracted correctly and the file with the signals was also created.

EVALUATION OF THE TEST
The test was carried out satisfactorily. Corrective actions are not required.
Requirement to try: FN2. Storing the extracted signals in a Matlab file.
Executor: Jorge Alarcón Álvarez
Date and Time: 10/30/2020, 08:50 pm
Evaluator: Dr. Miguel Ángel Ruíz Jaimes
ACTIONS DONE
1. The file path where the extracted signals were stored was entered.
2. Verified that the file exists and is in the correct format.
3. The file was opened and it was verified that the signals were correctly stored
RESULTS OBTAINED
The system displayed the table with the extraction results and also created the file in which the extracted signals were stored.

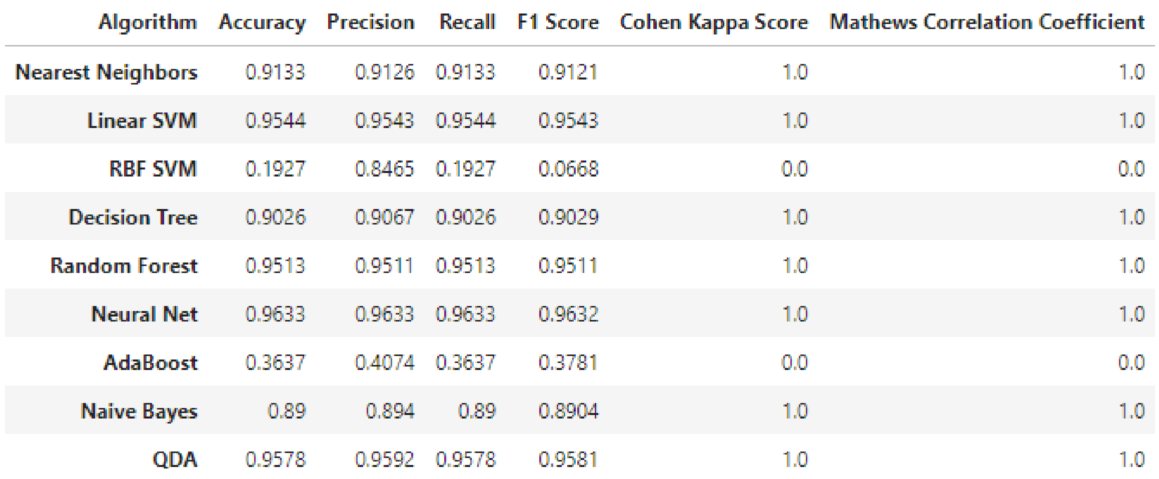


	mean_a	mean_aa	mean_aaa	mean_aaaa	mean_aaaaa	mean_aaaaaa	mean_d	mean_dd	mean_ddd	mean_dddd	...	ratio_aaa/aaaa	ratio_aaaa/aaaaa	ratio_aaaaa/aaaaaa
0	0.501250	0.708875	0.954064	1.308125	1.849968	2.616250	0.012816	0.011000	0.013347	0.014875	...	0.729337	0.707107	0.707107
1	0.517425	0.730000	0.995518	1.337375	1.891334	2.674750	0.012772	0.012250	0.015488	0.018875	...	0.744382	0.707107	0.707107
2	0.514597	0.727750	0.988977	1.341625	1.897344	2.683250	0.012861	0.013062	0.015203	0.018625	...	0.737149	0.707107	0.707107
3	0.516961	0.731094	0.995032	1.349063	1.907862	2.698125	0.011999	0.012094	0.014717	0.014938	...	0.737573	0.707107	0.707107
4	0.500808	0.704250	0.983320	1.256375	1.776783	2.512750	0.013037	0.011062	0.013347	0.016375	...	0.782665	0.707107	0.707107
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
12725	0.394521	0.557938	0.740252	0.931875	1.260241	1.753250	0.016971	0.011250	0.014496	0.015750	...	0.794369	0.739442	0.718803
12726	0.413459	0.584719	0.772470	1.001313	1.350132	1.909375	0.016727	0.010469	0.012949	0.012937	...	0.771457	0.741640	0.707107
12727	0.383539	0.542219	0.701671	0.918313	1.277830	1.807125	0.016153	0.010469	0.013302	0.014563	...	0.764087	0.718650	0.707107
12728	0.365287	0.516031	0.681695	0.863688	1.168406	1.607125	0.016948	0.010781	0.013214	0.014188	...	0.789284	0.739202	0.727016
12729	0.383208	0.541938	0.702953	0.927625	1.243447	1.758500	0.016750	0.010875	0.013612	0.013375	...	0.757798	0.746011	0.707107

Figura 13. Resultados de la extracción de señales

EVALUATION OF THE TEST
The test was carried out satisfactorily. Corrective actions are not required.
Requirement to try: FN3. Extraction of features of electrocardiogram signals by signal decomposition using Wavelets.
Executor: Jorge Alarcón Álvarez
Date and Time: 10/30/2020, 09:00 pm
Evaluator: Dr. Miguel Ángel Ruíz Jaimes
ACTIONS DONE
1. The electrocardiogram signal to be analyzed was entered.
2. The type of signal decomposition using Wavelets was selected.
3. The decomposition level has been established.
4. The feature extraction algorithm was executed.
RESULTS OBTAINED
The system displayed the characteristics extracted from the electrocardiogram signal, including decay coefficients and energy values.

EVALUATION OF THE TEST
The test was carried out satisfactorily. It was verified that the extracted characteristics were consistent and relevant.
Requirement to try: FN4. Data classification using various types of machine learning classifiers.
Executor: Jorge Alarcón Álvarez
Date and Time: 10/30/2020, 09:11 pm
Evaluator: Dr. Miguel Ángel Ruíz Jaimes
ACTIONS DONE
1. Training and test data have been entered.
2. Several types of machine learning classifiers were selected.
3. The classifiers were trained with the training data.
4. The classifiers were evaluated with the test data.
RESULTS OBTAINED
After the analysis of the different algorithms, it was determined that the multilayer perceptron classifier was the one that could best classify cardiac arrhythmias.



Algorithm	Accuracy	Precision	Recall	F1 Score	Cohen Kappa Score	Mathews Correlation Coefficient
Nearest Neighbors	0.9133	0.9126	0.9133	0.9121	1.0	1.0
Linear SVM	0.9544	0.9543	0.9544	0.9543	1.0	1.0
RBF SVM	0.1927	0.8465	0.1927	0.0668	0.0	0.0
Decision Tree	0.9026	0.9067	0.9026	0.9029	1.0	1.0
Random Forest	0.9513	0.9511	0.9513	0.9511	1.0	1.0
Neural Net	0.9633	0.9633	0.9633	0.9632	1.0	1.0
AdaBoost	0.3637	0.4074	0.3637	0.3781	0.0	0.0
Naive Bayes	0.89	0.894	0.89	0.8904	1.0	1.0
QDA	0.9578	0.9592	0.9578	0.9581	1.0	1.0

Figura 14. Resultados de los algoritmos.

EVALUATION OF THE TEST
The test was carried out satisfactorily. Corrective actions are not required.
Requirement to try: FN5. Comparison of different classifiers.
Executor: Jorge Alarcón Álvarez
Date and Time: 10/30/2020, 09:30 pm
Evaluator: Dr. Miguel Ángel Ruíz Jaimes
ACTIONS DONE
1. Several machine learning classifiers were selected.
2. Training and test data have been entered.
3. The classifiers were trained with the training data.
4. The classifiers were evaluated with the test data.
5. Classifier performance metrics were compared.
RESULTS OBTAINED
After the analysis of the different algorithms, it was determined that the Random Forest classifier obtained the best precision and recall in the classification of cardiac arrhythmias.
EVALUATION OF THE TEST
The test was carried out satisfactorily. Corrective actions are not required.

6 Conclusions

The main objective of the project was to develop a cardiac arrhythmia classifier model that could distinguish between normal heartbeats and arrhythmias from electrocardiogram (ECG) signals. This sought to provide a fast and non-invasive method of diagnosing cardiovascular diseases. The objective was fully achieved, achieving the implementation of a library with a programming interface (API) that can be used by both programmers and people with knowledge in ECG signal analysis. The library allows extracting signals from the MIT-BIH Arrhythmia Database, as well as characteristics of these signals using different types of wavelets. It also allows training various classifiers and analyzing signals using a previously trained classifier to identify arrhythmias, locate affected intervals, and graphically display them.

This library is easily accessible, as it is available in the official Python library repository, PyPI. Information collected during implementation was used to write a more detailed scientific paper on signal processing in connection with this project.
Regarding future work, it is suggested to experiment with the characteristics extracted from the signals to determine their impact on the classification. In addition, it is proposed to explore other previously uninvestigated features that could improve the ranking metrics of the algorithms used. The possibility of testing different wavelet families and signal decomposition levels to optimize the classification results is mentioned. It is also recommended to evaluate the methods developed in databases of ECG signals different from the one used in the project, considering possible changes in cardiac patterns due to environmental adaptations and changes in the population over time. These changes could be implemented through modifications to existing functions, taking advantage of the modularity and good practices implemented in the original project.

References

Bass, L., Clements, P., & Kazman, R. (2003). *Software Architecture in Practice* (Second Edition ed.). Addison-Wesley. https://doi.org/10.5555/773239

Clínica Universidad de Navarra. (n.d.). Closed interview. CUN. Retrieved October 9, 2020, from https://www.cun.es/diccionario-medico/terminos/entrevista-cerrada

Clínica Universidad de Navarra. (n.d.). Opened interview. CUN. Retrieved October 9, 2020, from https://www.cun.es/diccionario-medico/terminos/entrevista-abierta

NumPy Project. (n.d.)). What is Numpy. Read The Docs. Retrieved October 19, 2020, from https://numpy.readthedocs.io/en/stable/user/whatisnumpy.html

Pressman, R. S. (2002). *Ingeniería del Software. Un enfoque Práctico* (Quinta edición). Concepción Fernández Madrid.
SciKit-Learn. (2020, 3 de octubre). GitHub. https://github.com/scikit-learn/scikit-learn

SciPy Developers. (n.d.)). About SciPy. SciPy. Retrieved October 22, 2020, from, de https://www.scipy.org/about.html

The PyData Development Team. (n.d.). Python Data Analysis Library. PyData. Retrieved October 19, 2020, from https://pandas.pydata.org/

WHO. (n.d.). ¿Qué son las enfermedades cardiovasculares? https://www.who.int/cardiovascular_diseases/about_cvd/es/