Time Series Recognition Through Machine Learning

Biometric Systems project

Michele Damato, damato.1847565@studenti.uniroma1.it
Luigi Sarcina, sarcina.1873458@studenti.uniroma1.it

MSc in Cybersecurity
Sapienza Università di Roma

Contents

1 Introduction

Biometric systems are able to identify a person by the traits that characterize him. These biometrical features are widely used as they uniquely identify a subject and therefore allow automatic systems to be able to do what we humans do instinctively through the brain. Starting from this concept, several implementations have been carried out on different biometric traits. The most common traits used include fingerprints, iris scanning and facial recognition.

This report presents a preliminary study for a biometric recognition system based on time series of different possible nature. Although this system is applicable to different types of time series, what interests us in the field of biometric systems is the study of a person's walk, however not recording the images of the walk but starting from the data of sensors applied to the body such as the accelerometer of a mobile phone.

As a consequence, for the purposes of this study, we based ourselves on a pre-existing dataset thus providing a possible case study for an *uncontrolled closed set identification system* based on data recorded by an accelerometer while the subjects walked.

2 Dataset selection

For the purpose of this study, we considered the *Gait Dataset* created by Jordan Frank at McGill University [3]. This dataset is made up of forty *".csv"* files each containing the raw data collected by different sensors of a mobile phone in the pocket of 20 individuals, performing two separate 15 minute walks on two different days.

These files are organized as follows: each line is associated with a timestamp represented in milliseconds, consequently the increase of the row index corresponds to the passage of time; as regards the columns, there are multiple, each of which indicates a specific data collected by the sensors. The structure of the file can be summarized as follows: the first column represents the timestamps while the other columns represent functions of time or the evolution of a certain type of data over time. The three columns (*accel_x, accel_y, and accel_z*) give the values collected by the triaxial accelerometer, and the units associated to them is m/s^2.

In order to increase the volume of data contained in the dataset to make it more detailed and therefore more useful for an analysis by the machine learning algorithm, we decided to further fragment the initial file data to create a much more complete data frame but based on the same twenty initial subjects. This choice was made, in addition to the increase in the data to be analyzed, to overcome the computational problem that the feature extraction algorithm had faced us. In fact, the algorithm found it difficult to analyze the almost twenty-five thousand lines on average that each file contained and to extract the most relevant characteristics from them.

Starting from the records provided, the code present in the reading script of the dataset, specific for this case, performs an operation of extraction from each *".csv"* file of 10 subsets composed by 250 consecutive lines each, starting from a randomly selected line. In consideration of this case study the only columns that are taken into consideration during the extraction are the timestamp, accel_x, accel_y and accel_z columns.

At the end of the execution of the code we will obtain a new dataset consisting of 20 records (10 per day) for each subject in the dataset, for a total of 400 records. Please also note that during this operation the column named "timestamp" is renamed in "time" for the purpose of standardizing the interface between the script responsible for reading the dataset and the rest of the code. In addition, two columns have been added for the correct functioning of the code and the feature extraction and machine learning algorithms:

- *ts_id*: containing the time series identifier;
- *subj_id*: containing the subject identifier.

The data contained in these two columns will be used to populate the target vector, which represents the *time series - subject* association.

2.1 Several possibilities

As mentioned in section 1, the construction of this system took place with the idea in mind of being able to reuse it with different types of time series. This means that different datasets can be used and analyzed through this code and consequently, as also reported in the documentation itself, it is necessary to provide an ad hoc script with a common interface in order to read the specific dataset.

All this makes this system flexible and versatile with different classification configurations, for example predictions of price range variations over time, predictions of malfunctions in factories based on the analysis of the behavior of instruments over time or prediction of the type of human activity based on temporal analysis of the activities themselves.
In general, any type of time series classification can be performed.

3 System Execution

The whole system is written in the Python3 language. After consulting the results obtained by [4] we decided to use the Random Forest algorithm since according to the study cited it is the algorithm that obtained the best score in terms of accuracy.

As for the system architecture, the code and the Python modules that compose it, we refer you to the project folder from where you can access the complete documentation and the code is also present in easily accessible HTML format.

In order to execute the code, several libraries are required which are contained in the file requirements.txt. To install them all at once launch, form inside the folder, this command from terminal:

```
$ cat requirements.txt | xargs sudo pip3 install
```

Then we can move on to the execution of the main program by typing inside the terminal:

```
$ python3 bio_ts_identifier.py
```

Once the program has been launched, a menu will appear showing all the functions it makes available to the user in the order presented:

- **A**: Train Model
- **B**: Automatic Evaluation
- **C**: Manual Test
- **R**: Reset Instances
- **Q**: Exit

3.1 Model Training

As a first fundamental operation to carry out there is the training of the model based on the data present in the dataset files, which will be automatically analyzed and from which the dataset will be created as mentioned in the section 2. To do this, select option "A":

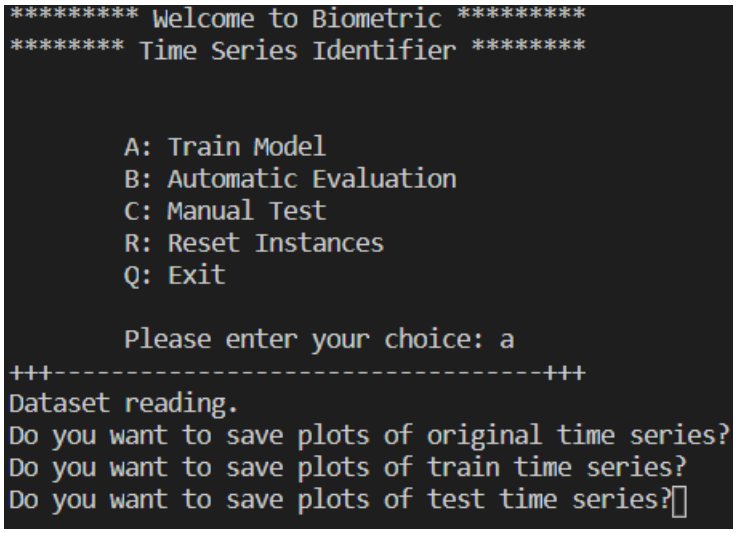

Figure 1: The program, before performing the training operations, asks if you want to save the plots related to the Time Series.

During the reading, analysis and subsequent creation of the dataset, you will be asked if you want to generate and save the plots relating to the different time series. If yes, type *"Y"* or *"y"*, otherwise type any other key. If you have chosen to save them they will be saved in the *"plots"* folder.

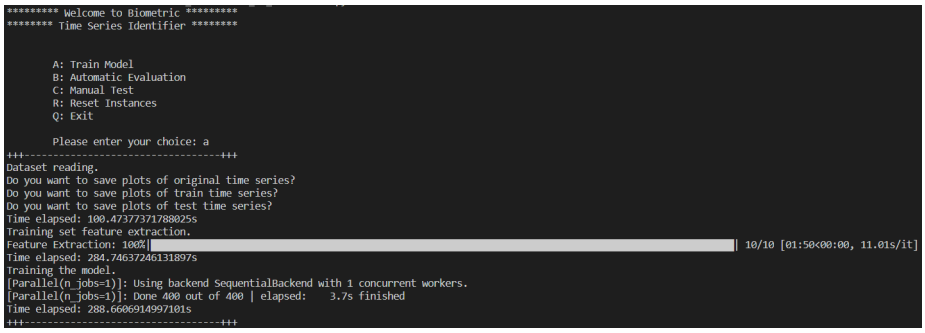

Figure 2: Execution of model training

The model training activity is preceded by two fundamental operations:

1. Dataset reading;
2. Feature extraction;

To carry out the features extraction and training phase of the model, we used two Python libraries that provide fundamental functions for our purposes. The *scikit-learn* [5] library provides a multitude of functions suitable for the use of machine learning techniques. The *tsfresh* [2] library instead provides useful functions to perform complete feature extraction also based on the target vector and therefore allowing the selection of the most important and useful features, it is also optimized for coexistence with *scikit-learn*.

3.2 Manual Tests

At the end of the training phase, it will be possible to carry out tests manually ("C" option). Will be asked if we want to generate test files for each subject in the dataset.

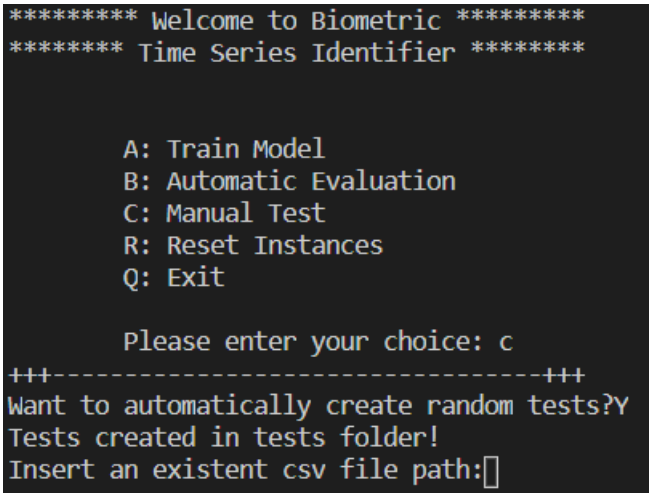

Figure 3: Creation of tests file

Let's proceed with the creation of these by typing "Y" (Yes) and we will get the following result as in the figure:

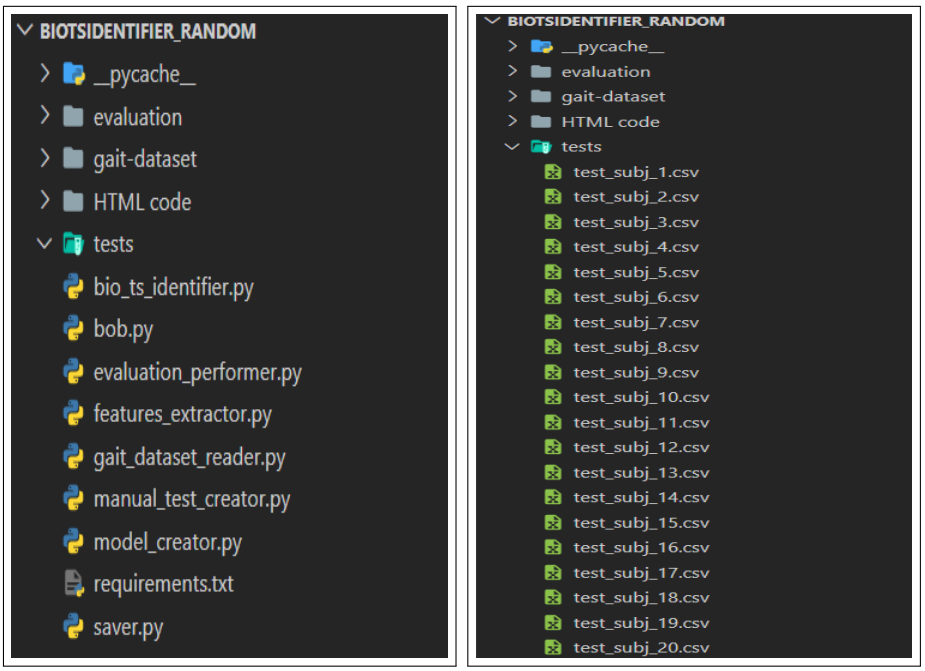

Figure 4: "Tests" folder before vs after the creation of tests file for each subject

After creating the test files, we select the path to pass to the program as argument. Next the operation of feature extraction and prediction will be handled, and wait for the result to be shown:

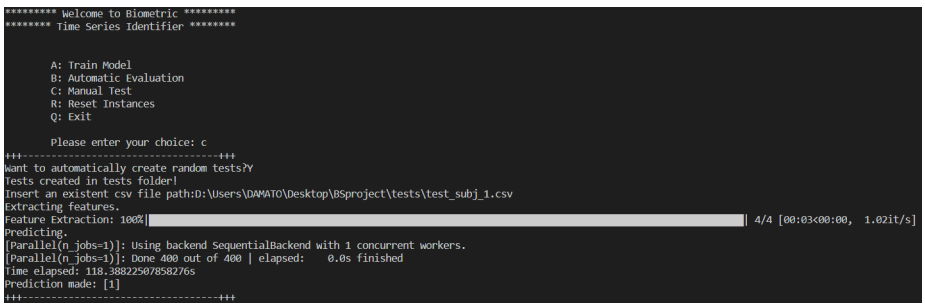

Figure 5: Manual test execution on a random file test choose from the "Tests" folder

4 System Evaluation

Before proceeding with the evaluation, some basic concepts useful for understanding this activity should be kept clear:

- **CMS (Cumulative Match Score)**: The probability of identification at rank k, or even the ratio between the number of individuals which are correctly recognized among the first k and the total number of individuals in the test set (probe).
- **CMC (Cumulative Match Characteristic)**: CMC curve shows the CMS value for a certain number of ranks (clearly, each implying the following ones). It therefore reports the probability that the correct identity is returned at the first place in the ordered list (CMS at rank 1), or at the first or second place (CMS at rank 2), or in general among the first k places (CMS at rank k). If the number n of ranks in the curve equals the size of the gallery, we will surely have a probability value of 1 at point n.

  For the calculation of the metrics above we used a function taken from the bob library [1].
- **RR - Recognition Rate**: CMS at rank 1 is also defined as Recognition Rate.
- **Confusion Matrix**: a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one (in unsupervised learning it is usually called a matching matrix). Each row of the matrix represents the instances in a predicted class while each column represents the instances in an actual class (or vice versa).
- **Precision**: is the ratio between the true positives (TP) number and the false positives (FP) number. The precision is intuitively the ability of the classifier not to label as positive a sample that is negative.

$$\text{precision} = \frac{TP}{(TP + FP)}$$

- **Recall**: is the ratio between the true positives (TP) number and the false negatives (FN) number. The recall is intuitively the ability of the classifier to find all the positive samples.

$$\text{recall} = \frac{TP}{(TP + FN)}$$

- **F1 Score**: weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0. The relative contribution of precision and recall to the F1 score are equal.The formula for the F1 score is:

$$\text{F1} = \frac{2 * (precision * recall)}{(precision + recall)}$$

For the evaluation phase we decided to carry out a small experiment able to make us understand which approach was the best between the two programs

proposed. In fact in order to carry out this experiment, we have created another system almost identical to the original one described in the previous sections. The small difference lies in the way of creating the so-called "final datasets" which will then be used respectively as training dataset and testing dataset by the program and the Random Forest algorithm. In fact, our original system called *"BioTSIdentifier_random"* creates, starting from the original *".csv"* files, the final dataset by randomly choosing multiple subsets which are then assigned an identifier (*ts_id*) and which are then associated with a specific subject (*subj_id*). The second system called *"BioTSIdentifier_static"* instead chooses the subsets consecutively therefore without any intrinsic randomness. It follows that at each execution of this code the final dataset will always be identical. In order to have sufficiently precise results, we decided to run the code for both systems several times to be precise 10 times each. At the end of the executions we calculated an average of the obtained accuracies and compared them.

The final objective of the experiment is to verify what we had hypothesized, that the accuracy obtained from the random system is higher than the one obtained from the static system. Before showing the result of this experiment it is necessary to show what the automatic evaluation function that we have implemented for this system provides the user by saving various significant metrics in the *evaluation* folder.

Below is an example of one of the ten evaluations performed on the program in the *"randomized"* version:

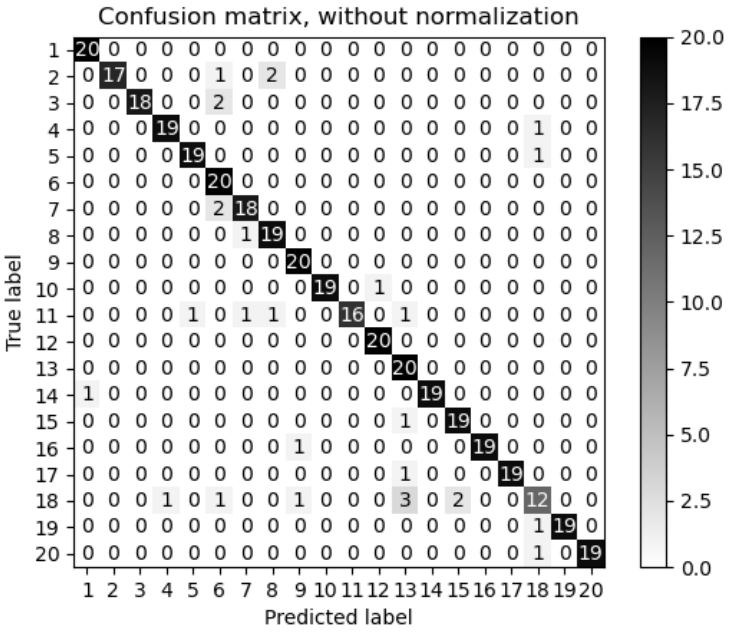

Figure 6: Confusion matrix without normalization of a random system

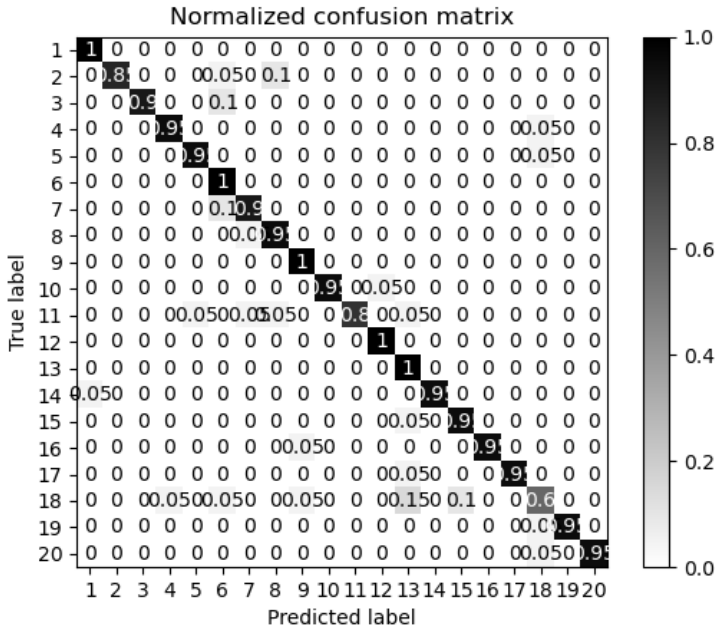

Figure 7: Normalized confusion matrix of a random system

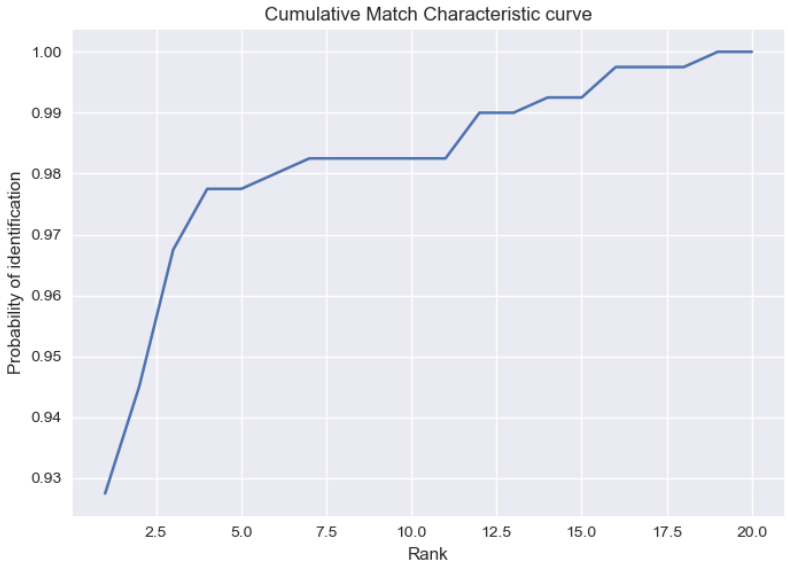

Figure 8: Cumulative Match Characteristic curve of a random system

Classification report				
id	precision	recall	f1-score	support
1	0.95	1.00	0.98	20
2	1.00	0.85	0.92	20
3	1.00	0.90	0.95	20
4	0.95	0.95	0.95	20
5	0.95	0.95	0.95	20
6	0.77	1.00	0.87	20
7	0.90	0.90	0.90	20
8	0.86	0.95	0.90	20
9	0.91	1.00	0.95	20
10	1.00	0.95	0.97	20
11	1.00	0.80	0.89	20
12	0.95	1.00	0.98	20
13	0.77	1.00	0.87	20
14	1.00	0.95	0.97	20
15	0.90	0.95	0.93	20
16	1.00	0.95	0.97	20
17	1.00	0.95	0.97	20
18	0.75	0.60	0.67	20
19	1.00	0.95	0.97	20
20	1.00	0.95	0.97	20

accuracy	0.93	400		
macro avg	0.93	0.93	0.93	400
weighted avg	0.93	0.93	0.93	400

Accuracy score (Recognition Rate): 0.9275

Cumulative Match Score as the rank increases:	
rank	scores
1	0.9275
2	0.945
3	0.9675
4	0.9775
5	0.9775
6	0.98
7	0.9825
8	0.9825
9	0.9825
10	0.9825
11	0.9825
12	0.99
13	0.99
14	0.9925

Cumulative Match Score as the rank increases:	
15	0.9925
16	0.9975
17	0.9975
18	0.9975
19	1.0
20	1.0

Below, instead, we have an example of one of the ten evaluations made on the program in the *"static"* version:

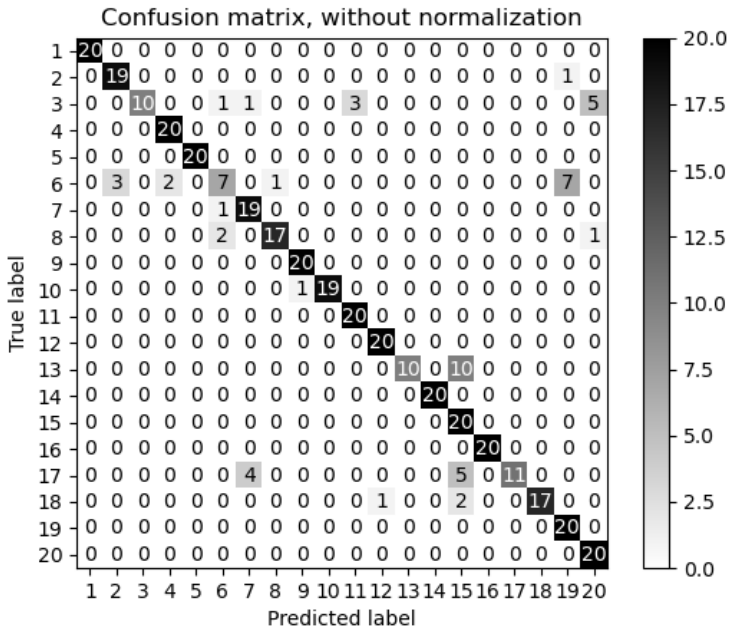

Figure 9: Confusion matrix without normalization for a static system

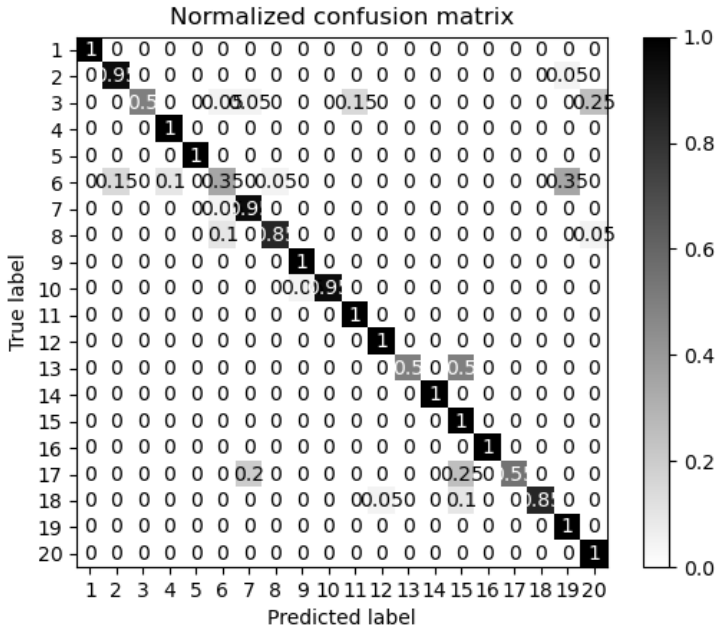

Figure 10: Normalized confusion matrix for a static system

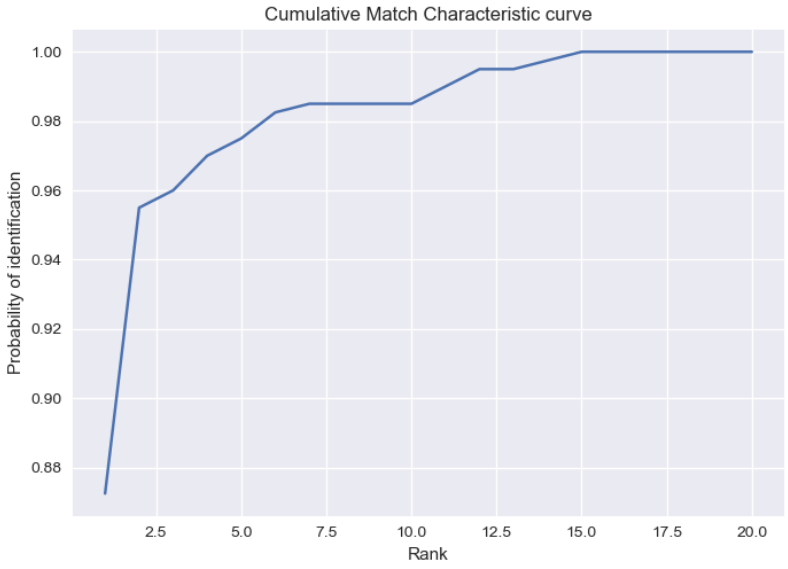

Figure 11: Cumulative Match Characteristic curve for a static system

Classification report				
id	precision	recall	f1-score	support
1	1.00	1.00	1.00	20
2	0.86	0.95	0.90	20
3	1.00	0.50	0.67	20
4	0.91	1.00	0.95	20
5	1.00	1.00	1.00	20
6	0.64	0.35	0.45	20
7	0.79	0.95	0.86	20
8	0.94	0.85	0.89	20
9	0.95	1.00	0.98	20
10	1.00	0.95	0.97	20
11	0.87	1.00	0.93	20
12	0.95	1.00	0.98	20
13	1.00	0.50	0.67	20
14	1.00	1.00	1.00	20
15	0.54	1.00	0.70	20
16	1.00	1.00	1.00	20
17	1.00	0.55	0.71	20
18	1.00	0.85	0.92	20
19	0.71	1.00	0.83	20
20	0.77	1.00	0.87	20

accuracy	0.87	400		
macro avg	0.90	0.87	0.86	400
weighted avg	0.90	0.87	0.86	400

Accuracy score (Recognition Rate): 0.8725

Cumulative Match Score as the rank increases:	
rank	scores
1	0.8725
2	0.955
3	0.96
4	0.97
5	0.975
6	0.9825
7	0.985
8	0.985
9	0.985
10	0.985
11	0.99
12	0.995
13	0.995
14	0.9975

Cumulative Match Score as the rank increases:	
15	1.0
16	1.0
17	1.0
18	1.0
19	1.0
20	1.0

After analyzing the system evaluation method we can draw conclusions about our hypotheses. The results below reflect what we initially thought:

Accuracy calculated on the two systems		
# of test	Random System	Static System
1	0.91	0.8725
2	0.89	0.8825
3	0.9275	0.8725
4	0.8725	0.8775
5	0.9225	0.88
6	0.925	0.875
7	0.9225	0.8775
8	0.895	0.875
9	0.895	0.875
10	0.9125	0.8775
Mean Accuracy	**0.90725**	**0.8765**

With an average accuracy of 0.8765 the static version system was less efficient than the random version system, which obtained an average accuracy of 0.90725.

5 Future improvements

Remaining in the field of biometric recognition, a possible future implementation of the system shown could be the recognition through smartwatch of the features characterizing the way to perform different types of movement. For example it would be possible to use it for the construction of an authentication system through gestures such as:

- writing something (e.g. a signature) on a sheet of paper;
- performing a predefined movement in the air.

We can therefore imagine how such a system could be organized and how the data can be used starting from the collection by the smartwatch sensors passing through the application on the smartphone up to sending them to the server where they will be processed and from which a response will be returned by checking the gallery on the server itself. So the main elements that could potentially be part of such a system are:

- a smartwatch, with which to collect data;

- a smartphone, with which communication between smartwatch and server is possible;
- a running server, with which all data operations are carried out.

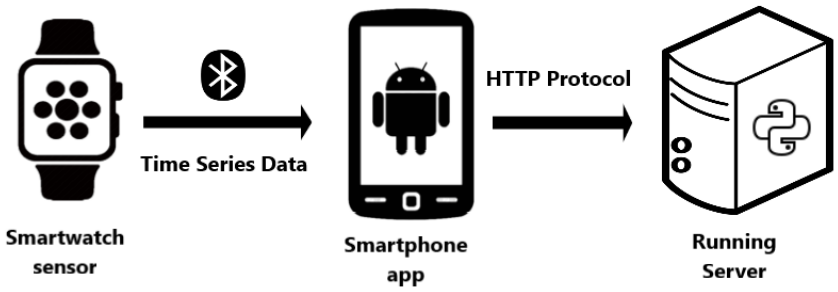

Figure 12: possible implementation

Figure 12 summarizes a possible operation of the scheme:

1. the smartwatch collects data both during the enrollment which will form the gallery and during the recognition of a user thus providing the initial structure of a probe which will then be analyzed and send everything to the smartphone via Bluetooth;
2. the smartphone, through an app, organizes all phases providing visual feedback to the user and acts as an intermediary between the smartwatch and the server by forwarding the data received to the server via the HTTP protocol;
3. once the data has been received, the server proceeds according to whether it is an enrollment or a recognition to carry out the operations relating to the respective procedure. In case of recognition, it will respond to the request with a binary response (true or false) in case of verification or with the user id in case of identification depending on the configuration that will be set to the system.

6 Conclusions

The study conducted through the development of this project results innovative from the point of view of the use of time series as biometric features suitable for the recognition of individuals, in fact this type of use is not yet widespread. Although the use of time series derived from sensors applied to the human body in combination with the use of machine learning algorithms are widely used to classify human activities this type of approach is still very limited in the field of security. Consequently, this system acts as a starting point for future studies regarding cybersecurity and predictive data analysis. Some proposals could be:

1. *Human identification/verification*: in the field of security, the possibility of identifying an individual starting from a pre-existing gallery or the

possibility of verifying that the identity provided by an individual is the right one is highly requested. This is a problem of biometric systems related to the field of cybersecurity. A possible solution could be the one proposed in section 5. The application scenarios can be manifold, for example:

- the use of bracelets with accelerometer-like sensors capable of identifying in real time the workers of a company who have guaranteed access within the structure, shortening control times;
- the verification using as a biometric trait the way to run to unlock the smartphone during exercise without further encumbrances or obstacles, as is currently done by fingerprint.

2. *Classification of criminal activities*: in the field of predictive data analysis the use of this system, in combination with algorithms for extrapolating data from multimedia elements such as camera recordings, could bring major improvements in the recognition of human activities. Therefore, not focusing on the classification of the classic types of activities such as *running, walking, etc.*, but training the system to recognize suspicious movements and possible offences that are carried out within the perimeter of large environments (whether public or private).

The work carried out shows how the implementation of this type of system is comparable as a level of confidentiality to the classic facial recognition systems, broadly guaranteeing the same implications as regards privacy. Different the matter regarding the intrusiveness and consequently the acceptability of the measurement since for this system it is necessary to use at least a bracelet.

Another positive result derived from the project work is the possibility of adaptability and potential scalability of this system. In fact (despite the lack of large quantities of data and of subjects to be able to analyze and on which to be able to train the algorithm) the final evaluation can be considered satisfactory. Furthermore, even if no real simulation has been carried out, we are confident that the accuracy of the results obtained can be replicated even with large amounts of data while ensuring a good performance in terms of processing time.

References

[1] Switzerland Biometrics group at the Idiap Research Institute. *bob: free signal-processing and machine learning toolbox*. URL: https://www.idiap.ch/software/bob/.

[2] Maximilian Christ, Nils Braun, and Julius Neuffer. *tsfresh: automatic extraction of relevant features from time series*. URL: https://github.com/blue-yonder/tsfresh.

[3] Jordan Frank. *Gait Dataset*. URL: https://www.cs.mcgill.ca/~jfrank8/data/gait-dataset.html.

[4] Aybuke Kececi et al. "Implementation of machine learning algorithms for gait recognition". In: *Engineering Science and Technology, an International Journal* (2020). ISSN: 2215-0986. DOI: https://doi.org/10.1016/j.jestch.2020.01.005. URL: http://www.sciencedirect.com/science/article/pii/S2215098619306214.

[5] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.