UNIVERSITY OF CATANIA

MASTER'S DEGREE COURSE IN DATA SCIENCE FOR MANAGEMENT
CURRICULUM DATA-DRIVEN AND APPLICATION FOR IoT

Alessia Simone

SMART HEALTHCARE: CPS FOR STRESS DETECTION

Supervisor:
Prof. Maurizio Palesi

Co-Supervisor:
Prof. Sebastiano Battiato

Academic Year 2022-2023

"To your paws, forever etched within my heart."

Contents

Abstract

Smart wearable devices play an essential role in the fourth industrial revolution, thanks to the prominence of cyber-physical systems, which have enabled the introduction of Artificial Intelligence within embedded devices. The application of such devices in the medical field is growing due to the vital support they can provide during treatment and therapy.

This thesis aims to develop a low-power wearable device with a stable and easy-to-deploy algorithm that can detect stress peaks by analyzing the biometric values of each individual. In particular, the project presents a comparison between two essential branches of Artificial Intelligence: on the one hand, the neural network, belonging to the technique of deep learning, which can provide largely accurate results due to the growing research in this area, and on the other hand, the use of supervised machine learning algorithms that are mainly based on the analysis of features extracted from original signals and appears to be extremely fast and easy to implement.

Although the experimental part has provided acceptable results, it seems necessary to evaluate further implementations and improvements due to the importance of such mental health support within the medical field, such as contextual and physiological data.

Keywords: #smart, #stress, #cps, #machinelearning, #deeplearning

Acknowledgements

I want to express my deepest gratitude and appreciation to the following individuals and organizations, without whom this thesis would not have been possible:

To my supervisor, to him go my deepest thanks for believing in me and my work until the end, expressing deep trust, empathy, kindness and endless support.

To my co-supervisor, I appreciate the multitasking skills and the crucial support to part of my work, without which my thesis would not have been so detailed.

To the entire teaching staff for instilling a passion for this field of study in me in the best possible way and for their endless kindness and helpfulness throughout my studies.

To my family, the most important pillar of my life goes the most important thanks for believing in me more than anyone else, spending moments of joy with me, and cheering me up in times of despair.

To my other half, for always listening to me very carefully without ever giving up, for taking care of my mental and physical state during this journey, for always providing me with important considerations on all my projects and for always supporting them without ever letting me give up.

Finally, to myself, to my strength and determination to carry out this project, which I hope will be helpful to many of those people who experience high levels of problems even in the smallest actions of daily life, may you find the strength to overcome them.

List of Tables

List of Figures

Acronyms

2D: 2 Dimentional; 14

ACC: Accelerometer; 11

AI: Artificial Intelligence; 3; 4; 7

AUC: Area Under the Curve; 25; 26; 27; 28

BLAST: Bi-Lateral Alternating Stimulation-Tactile; 5

BPM: Beats Per Minute; 12; 13; 14; 29; 48

BVP: Blood Volume Pulse; 11; 40; 52

CAM: Class Activation Map; 36; 37; 38

CNN: Convolutional Neural Network; 33; 34; 36

COVID-19: Corona Virus 2019; 4

CPS: Cyber Physical System; 1; 3;

CPU: Central Process Unit; 30

DSM-5: Diagnostic and Statistical Manual of mental disorders 5th edition; 2

ECG: Electrocardiogram; 8

EDA: ElectroDermal Activity; 11

EEG: Electroencephalogram; 4; 8

FC: Fully Connected; 33

FPR: False Positive Rate; 27; 28

GASF: Gramian Angular Summation Field; 31

GSR: Galvanic Skin Response; 8

HR: Heart Rate; 6; 8; 12; 45; 46; 52

HRV: Heart Rate Variability; 5; 6; 52

Hz: Hertz; 12

IBI: Inter Beat Interval; 52

ID: Identifier; 12; 18

IoT: Internet of Things; 3; 30; 39

Lbfgs: Limited Broyden–Fletcher–Goldfarb–Shanno; 19

LED: Light Emitting Diode; 41

LOGO: Leave One Group Out; 18; 20

LSAS-SR: Liebowitz Social Anxiety Scale Self Reported; 45

MIoT: Medical Internet of Things; 1

ML: Machine Learning; 7; 15; 29; 38

PPG: Photoplethysmograph; 4; PhotoPlethysmoGraph; 40

PTSD: Post Traumatic Stress Disorder; 2

RGB: Red Green Blue; 33

ROC: Receiver Operator Characteristic; 25

Sag: Stochastic Average Gradient; 19

Saga: Stochastic Average Gradient Accelerated; 19; 20

SNR: Signal-to-Noise Ratio; 47

ST: Skin Temperature; 45; 46

SVM: Support Vector Machine; 4

TEMP: Temperature; 11; 12

TPR: True Positive Rate; 27; 28

TSST: Tier Stress Social Test; 12; 16

WESAD: Wearable Stress and Affect Detection; 10

XGBoost: Extreme Gradient Boosting; 22; 23; 24; 25

Chapter 1 – Introduction

The evolution of artificial intelligence through the industrial revolutions until today has allowed the automation and simplification of decision-making processes in numerous fields of application. In particular, the use of innovative technologies in the healthcare field has allowed its rapid evolution, where the introduction of the MIoT leads to significant opportunities during patient monitoring and simplifying decision-making processes. Among all, the identification of stress represents one of the main objectives of research and evolution.

Stress is growing today and is a ubiquitous feature in each of us daily, bringing more negative aspects and difficulties, together with the growing need for monitoring and research of adjuvant solutions to this disorder. Stress disorder is linked to an increasing risk of cardiovascular pathologies, depression and anxiety, and, if identified and treated with a sudden and punctual intervention, it can reduce these risks.

Embedded wearable devices represent one of the emerging tools in this field due to the many positive features and contributions: real-time processing of vital parameters, the compact form that makes them portable and wearable devices, reduced energy consumption, the privacy of confidential data, personalization and adaptability to different contexts within healthcare.

This introductory chapter laid the groundwork for the investigation into developing an embedded device that can detect peak stress. Subsequent chapters will delve into the technical aspects of device design, data acquisition, signal processing and validation: chapter 2 will examine the comparison between the two main branches of machine learning in order to identify the specific optimal approach to classify biometric values, chapter 3 will examine in detail the CPS in terms of hardware and software, in chapter 4 the device containing the pipeline is tested in order to examine the results in real life, in chapter 5 conclusions are provided to the present thesis, shedding light on further considerations for future improvements.

1.1 What is meant by Stress

«Stress is a state of mental or emotional strain or tension resulting from adverse or demanding circumstances.»

Oxford Languages

Analyzing emotions through the physiological responses of each individual has been a widely studied topic in the scientific field because of the increase in positive support that the multitude of methods has been capable of providing to improve individuals' psychophysical state.

Significantly, stress is our body's physiological response to external stimuli and internal cognitive processes (Ahmad & Khan, 2022). Nevertheless, how everyone reacts is entirely subjective. Some individuals are better able to handle stress than others, and prolonged exposure can cause numerous physical and psychological damages[1]:

- cardiovascular disease,
- depression and anxiety,
- headache,
- stomach upset,
- increasing usage of alcohol, tobacco and other substances[2]

Stress disorders were officially recognized as a mental disorder in ’90 by the American Psychiatric Association (American Psychiatric Association, 2013) and are categorized in the DSM-5 as “Trauma and Stressor-Related Disorders” in different shades: PTSD, Acute Stress Disorder, Adjustment Disorder, Reactive Attachment Disorder, Disinhibited Social Engagement Disorder, other specified Trauma and Stressor-Related Disorders. In particular, Adjustment Disorder is a consequence of stress-inducing events rather than traumatic events which characterize the other types. This type of disorder is comorbid, as it can manifest in

[1] Stress Symptoms, by Hedy Marks
[2] Stress | WHO

various symptoms like depression and anxiety[3] , and it is directly tied to the experience of stressors in daily life, as a relatively common reaction to such stressors; thus, it can be seen as one of the most common stress-related disorders resulting from daily life stimuli.

In this regard, wearable smart devices are the protagonists of the fourth industrial revolution and have been considered reliable for identifying stress by analyzing physiological inputs (Quadrini et al., 2022).

Also, Ghosh et al. (2022) point out that wearable sensors are an optimal solution for obtaining precautionary treatment in those cases where individuals are under stress conditions for a prolonged time.

1.2 Cyber-Physical Systems

The fourth industrial revolution plays an essential role in this domain, particularly the leading players of this era, such as AI, which makes it possible to automate and facilitate learning processes, and the IoT, which makes it possible to accelerate the exchange of information (Ali et al., 2022), enable the development of new wearable devices capable of simplifying the detection and treatment of symptomatic patients (Sarfraz et al., 2021). The development of such tools in synergy with human intervention has led to various discussions on the entry into the fifth industrial revolution, where human-technology collaboration is the prominent feature, especially in medical fields where the intervention of specialists is strictly required. (Noble et al., 2022).

CPS represent the prominent leader of this era, representing an evolution of embedded devices coming from the previous industrial revolution:

- Cyber represents the evolution of the software part, which involves the usage of novelty algorithms capable of processing signals coming from sensors,

[3] Officially recognized with code F43.23

- Physical represents the hardware part, composed of sensors, actuators and mighty boards related to the real-world aspect,
- Systems represent the interconnection of virtual and real-world aspects to generate intelligent and real-time decisions.

In particular, the spread usage of wearable devices appears increasingly attractive in this field of application for their power of prescreening anxiety and depression symptoms throw some daily and non-invasive characteristics, such as biometric measures, daily activity, movements and sleeping habits, majorly throw algorithm like Random Forest or SVM. (Ahmed et al., 2023).

1.3 State-Of-Art

The demand for easily accessible digital healthcare solutions has increased significantly, particularly in light of the COVID-19 pandemic. Wearable devices now incorporate sensors that were once exclusive to hospitals and medical professionals.

Various wearable devices are available in the market, serving as data collectors or medical aids during treatments to reduce anxiety peaks and panic attacks in a non-invasive manner that does not require a medical prescription. Outlined below are a few examples of such devices:

- Muse S (Gen 2): This brain-sensing headband utilizes Bluetooth connectivity to monitor brain activity, focus levels during meditation, and the duration spent in a focused state. It employs an array of sensors, including an accelerometer, gyroscope, PPG, EEG, pulse oximeter, and smart-fade technology to monitor sleep, brain activity, body movement, breath, and heart function [4] to monitor sleep, brain-body, breath, and heart. (*Muse - the Brain Sensing Headband*, 2021)

[4] It's an algorithm that can detect the moment during which the user wakes up from a deepsleep state and activate a series of actions to help the user to fall back asleep.

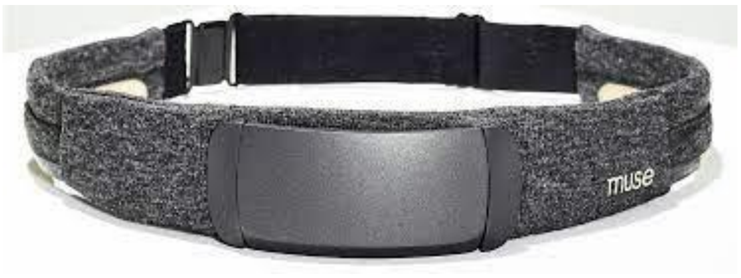


Figure 1 - Muse S Headband (https://choosemuse.my.site.com/)

- Sensate: Positioned on the chest, the Sensate is a device that stimulates the nerves responsible for stress responses. It emits infrasound frequencies that propagate to the vagus nerves, toning and stimulating them to induce relaxation. This approach aims to enhance stress resilience and increase HRV, a well-known biomarker of health and longevity (*Sensate - Meet Sensate, Your Key to Calm*, 2021)

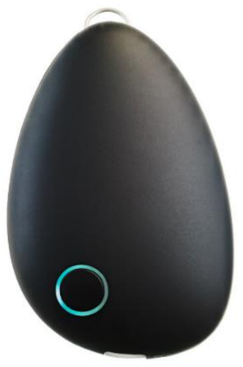


Figure 2 - Sensate (sensateinvest.com)

- Touchpoints: This pair of devices utilizes BLAST technology[5] to alleviate stress reactions through vibrations. The devices are worn on opposing sides of the body, and the vibrations can effectively reduce anxiety peaks within seconds (*Touchpoints Solutions*, 2023)

[5] The BLAST is a technology that, thanks to its placebo effect, can reduce the stress-related cortisol.

Figure 3 - Touchpoints (thetouchpointsolution.com)

- Apollo: With a wearable wristband, Apollo allows users to adjust the vibration intensity for managing sleep, social interactions, performances, and stress. Vibration therapy targets the sympathetic nervous system, reducing stress hormones like cortisol, slowing breathing, and increasing HRV (*Apollo Neuro*, 2023)

Figure 4 - Apollo Neuro (sleepfoundation.org)

- Oura: Designed as a ring, Oura utilizes precise HR and HRV metrics from the palm to connect with a smartphone application for monitoring sleep, fitness activity, and stress levels (*Oura Ring*, n.d.)

Figure 5 - Oura Ring (ouraring.com)

1.4 Related Works

Many researchers utilize existing wearable Smart Watches, such as FitBit or Empatica E4 (Garbarino et al., 2014), to collect biometric data from various sensors and employ ML models for analysis and prediction. These devices are valuable tools for gathering continuous and real-time physiological data, enabling researchers to study and understand various aspects of human health and behaviour.

However, the development of AI algorithms within wearable devices is still in its early stages. While many existing devices in the marketplace focus on monitoring body parameters, such as heart rate, sleep patterns, or physical activity levels, they often require user activation to initiate specific actions to reduce stress peaks (Ahmed et al., 2023). These devices are designed to provide feedback or reminders to users based on predetermined thresholds or user-defined goals.

Despite these limitations, ongoing advancements in wearable technology and AI pave the way for more sophisticated and intelligent devices. The integration of AI algorithms within wearable devices has the potential to enhance their functionality and provide personalized interventions for managing various aspects of health and well-being.

For example, researchers are exploring using AI algorithms to detect specific events or patterns in biometric data collected from wearable devices. These algorithms can analyze the data in real-time, identify moments when the user wakes up from a deep sleep state, and initiate a series of actions to help the user fall asleep. This demonstrates the potential for AI algorithms to proactively assist individuals in achieving better sleep quality and managing sleep-related issues.

In addition, developing wearable devices with advanced AI capabilities can go beyond simple monitoring and provide more active interventions. For instance, specific devices may utilize bilateral alternating somatosensory stimulation, a technology known for its placebo effect, to help reduce stress-related cortisol levels. These devices could autonomously deliver specific stimulation patterns based on AI algorithms to alleviate stress and promote relaxation (Ahmed et al., 2023).

As the field of wearable technology continues to advance, incorporating AI algorithms into these devices holds great promise. Future developments may enable wearable devices to proactively monitor and manage various health conditions, detect anomalies or potential risks, and provide real-time personalized interventions or recommendations. These advancements can potentially revolutionize how individuals monitor and manage their health, promoting proactive and preventive approaches to well-being.

Table 1 - Related works about emotion and anxiety detection

Authors	Biometrics	Subjects	Model	Accuracy
(Kanjo et al., 2018)	HR, electrodermal activity, body temperature, hand acceleration, air pressure, light UV	40	Stacking models	86%
(McGinnis et al., 2018)	Acceleration, angular velocity	59	Logistic regression	80%
(Kanjo et al., 2019)	HR, GSR, body temperature, accelerometer, gyroscope, noise, light UV, air pressure, location	40	C-LSTM	95%
(Agrafioti et al., 2012)	ECG	44	LDA	76%
(Ramzan et al., 2016)	EEG, ECG	8	KNN	74%
(Kim et al., 2004)	ECG, skin temperature, electrodermal activity	50	SVM	78%
(P. Schmidt et al., 2018)	Electrodermal activity	12	rSVM	98%

Chapter 2 – Methodology

Selecting a high-quality dataset is critical to creating a stable model that generalises to never-before-seen data.

Furthermore, while the tools provided to date by deep-learning produce excellent results due to the feature extraction methods within the neural network itself (such as convolution or the attention mechanism), spectral feature extraction provided to machine learning models still proves to be an excellent approach in contexts such as low-power devices and can overcome the weaknesses of deep-learning, such as the inability to learn from a low number of samples or the trade-off between efficiency and speed.

In the following sections, both methods will be compared to understand the optimal approach for this specific type of problem.

2.1 Dataset Description

The WESAD dataset is a valuable resource for researchers and developers engaged in affective computing, stress detection, and physiological signal analysis.

Researchers specifically designed it to facilitate the study of stress and affect recognition using wearable sensors (Schmidt et al., 2018). The dataset can be utilized for various applications, including but not limited to:

- Stress detection: Developing algorithms to detect stress levels based on physiological signals.
- Affect recognition: Investigating methods to recognize affective states, such as amusement or relaxation, from wearable sensor data.
- Human-computer interaction: Designing intelligent systems that realistically adapt to users' emotional and stress levels.
- Health monitoring: Exploring the relationship between physiological signals and stress-related disorders to improve health assessment and management.

The WESAD dataset was collected from 15 participants in a controlled laboratory environment. During the data collection process, the participants were asked to engage in various activities while wearing multiple physiological and motion sensors. The measurements encompassed different modalities: the RESP signals were captured using the Respiban device on the participants' chests[6]. In contrast, the EMPATICA E4 wristband was used to collect the TEMP, EDA, BVP, and ACC data positioned on the wrist (Garbarino et al., 2014).

Table 2 - Participants' details

Participant ID	Gender	Age
2	Male	27
3	Male	27
4	Male	25
5	Male	35
6	Male	27
7	Male	28
8	Female	27
9	Male	26
10	Male	28
11	Female	26
13	Male	28
14	Male	27
15	Male	28
16	Male	24
17	Female	24

The activities performed by the participants include baseline, stress, amusement, meditation, and watching a relaxing video clip. Each activity was designed to elicit specific physiological responses and affective states in the participants, allowing for the investigation of stress and affect recognition algorithms in various scenarios.

For this purpose, the heart rate (computed from the BVP signals with a peak detection algorithm) and the skin temperature from the wrist device have been considered and preprocessed with the following steps:

[6] https://www.pluxbiosignals.com/collections/wearables

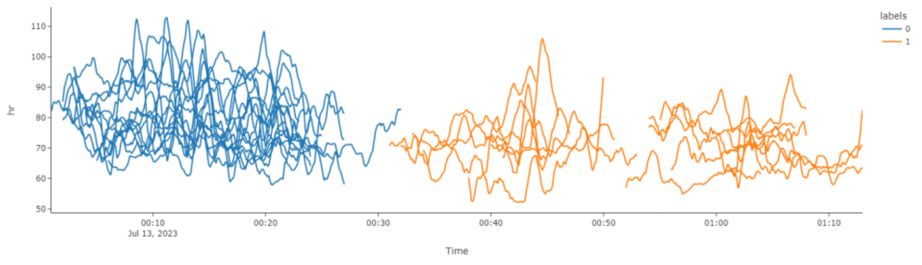
- Downsampling of the Skin Temperature signal from 4 Hz to 1 Hz to match the Heart Rate signal.
- Smoothing signals with a quadratic smoothing approach and rolling average window of size 3 seconds.
- Generation of target variable: transition time referring to moments when individuals were filling the questionnaire has been highlighted with a placeholder" 999"; then, labels have been added according to the additional" .txt" file, which contains information about the study protocol procedure and range of minutes.

The resultant dataframe, encompassing the selected variables, consists of 106,934 rows and five columns, namely 'ID, 'Time', 'HR', 'TEMP', and 'labels'.

Table 3 - Dataframe Overview

	ID	Time	hr	temp	labels
0	0	2023-07-13 00:00:00	100.943333	155.060433	999
1	0	2023-07-13 00:00:01	100.943333	155.060433	999
2	0	2023-07-13 00:00:02	100.943333	155.060433	999
3	0	2023-07-13 00:00:03	97.360000	82.106448	999
4	0	2023-07-13 00:00:04	91.260000	22.577358	999

The 'ID' column is an integer comprising 16 unique values corresponding to individual participants. The 'Time' column expresses biometric signals collected at 1Hz, organized in groups for each individual. The 'HR' and 'TEMP' variables are continuous floating-point values, where 'HR' denotes the heart rate in BPM, and 'TEMP' signifies the peripheral skin temperature in degrees Celsius. The 'labels' column contains six classes: ' placeholder', 'medi1', 'medi2', 'baseline', 'TSST', and 'baseline'.



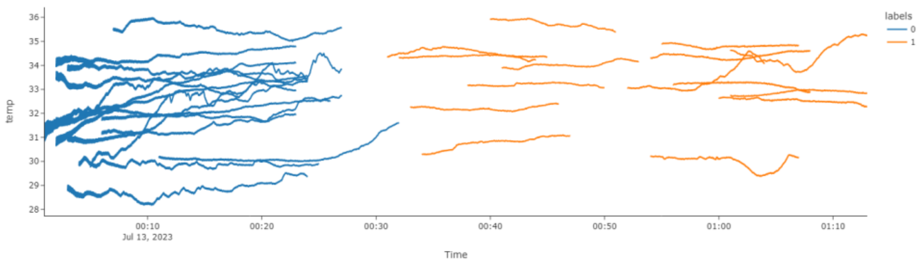


Figure 6 - Lineplots of biometric features for each individual: HR (top), Temp (bottom)

The line plots in Figure 7 show the trend of biometric features for baseline and stress class only for each individual. Here, it is possible to understand a distinct behaviour between the two classes on the BPM signals rather than the temperature signal, with slightly different durations for each individual.

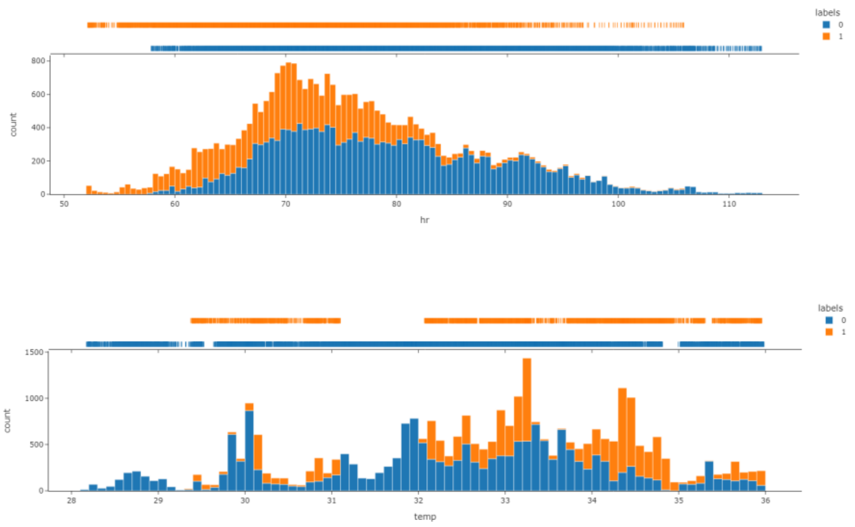


Figure 7 - Histogram of biometric signals per class: HR (top), Temp (bottom)

The plots in Figure 8 represent the histogram of both biometric signals coloured by labels. The current plot suggests a lack of clear distinction between the 'baseline' and 'stress' labels, making it difficult to learn and predict anxiety from raw data.

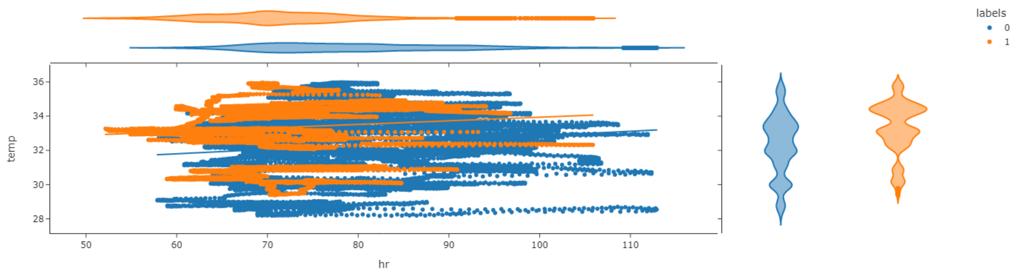


Figure 8 - Scatterplot of biometric features

The current plot shows the observations with BPM on the x-axis and temperature on the y-axis to understand the behaviour of samples in a 2D plane. In addition, the two distinct lines representing the linear correlation between the features and the plot outside the margin are the violin plot, which helps understand the distribution and outliers for each feature class.

The observations do not seem very well clustered in labels and are not linearly separable; this leads to the assumption that a linear algorithm on the raw data will not produce satisfactory results. The R^2 of the linear correlation in terms of the baseline class is 0.02; for the positive class, the amount is 0.01. This result allows declining any particular correlation between features, called multicollinearity, which can lead the model to negative behaviour.

Finally, from the violin plots, it is possible to detect some outliers before the minimum point for the positive temperature feature class and for both BPM classes after the maximum value. However, for this purpose, outliers are supposed to be representative behaviour of the experiment and have not been removed due to lack of knowledge base.

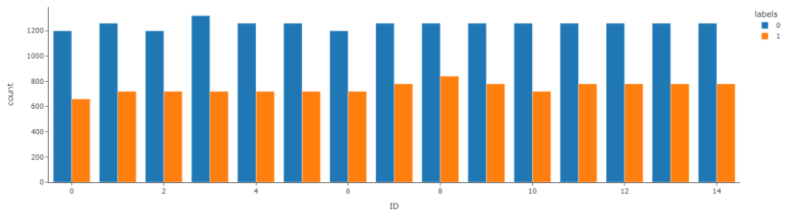


Figure 9 - Distribution of labels for each individual

Considering the experimental protocol's varying durations for each ground truth, the distribution of the target variables is imbalanced, as evidenced by the bar plot in Figure 10[7].

Based on the exploratory data analysis findings, it has been concluded that preprocessing the data to extract meaningful features, rather than utilizing the raw signals themselves, would be more beneficial in feeding the machine learning algorithm.

2.2 Supervised Machine Learning

ML consists of four crucial branches: supervised, unsupervised, semi-supervised and reinforcement learning. The most widespread of these methods is supervised machine learning due to its high learning potential and accuracy, especially applicable in real life. It is based on a learning process that takes into account specific inputs in order to generate outputs that come as close as possible to the ground truth while at the same time trying to make the model as generalizable as possible rather than perfecting it on a single task (Singh, 2019).

This project evaluates five distinct types of supervised machine learning algorithms: Logistic Regression as a linear model, Random Forest as an ensemble model, Gaussian Naive Bayes as a probabilistic model and Extreme Gradient Boosting as a gradient optimization model. Each of these algorithms has strengths and weaknesses, and understanding those models' strengths and weaknesses can help choose the most appropriate algorithm for this task, leading to more accurate and reliable predictions and classifications.

The models mentioned above are the most commonly used in related work (see Chapter 1) and are suitable for various scenarios due to the optimizable parameters that make them very generalizable, an essential feature for any learning algorithm.

[7] The Exploratory Data Analysis part has been developed as an interactive Tableau Dashboard (*Tableau*, n.d.) version 2023.2, published in Tableau Public

For this purpose, the objective is to find a model with the optimal set of parameters that contains the fewest misclassified observations in the "stress" class and is light enough to fit inside a low-power device.

2.2.1 Features Extraction

As mentioned in the paragraph above, a binary classification has been implemented. Thus, the ”Baseline” and ”TSST” protocol states only have been selected from the merged dataframe.

Then, eight features for each signal were extracted in the grouped dataset, applying a moving window of size 60:

- Minimum: the lowest value of the window
- Maximum: maximum value of the window
- Absolute minimum: lowest absolute value of the window
- Absolute maximum: highest absolute value of the window
- Mean: average value of the window

$$\frac{1}{n}\sum_{i=i}^{n} x_i$$

- Absolute energy: average of the sum over the squared values of the window
- Mean absolute change: average over the first difference of the window

$$\frac{1}{n-1}\sum_{i\,=\,1,\ldots,n-1} |x_{i+1} - x_i|$$

- CID-CE (Batista et al., n.d.): It calculates the values of

$$\sqrt{\sum_{i=1}^{n-1}(x_i - x_{i-1})^2}$$

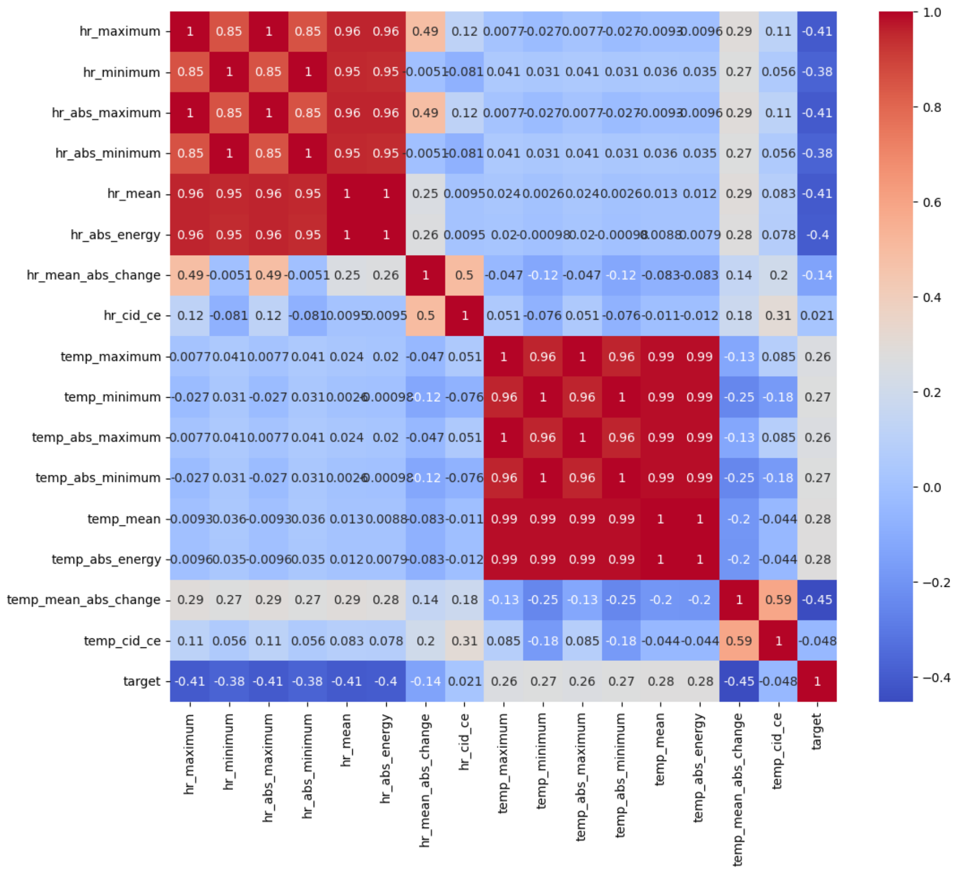


Figure 10 - Correlation Matrix of lag features

From the correlation matrix in Figure 2, it is possible to detect that all the features instead of mean absolute change and cid ce are positively correlated.

In addition, it is possible to detect that those features contribute the most to predicting the labels; in particular, features related to heartbeat are negatively correlated to the target variable and features related to skin temperature are positively correlated to the target variable.

A min-max scaler has been applied to avoid the different measurement scales and allows comparison between the groups of features by fixing a range of [0, 1] for each feature.

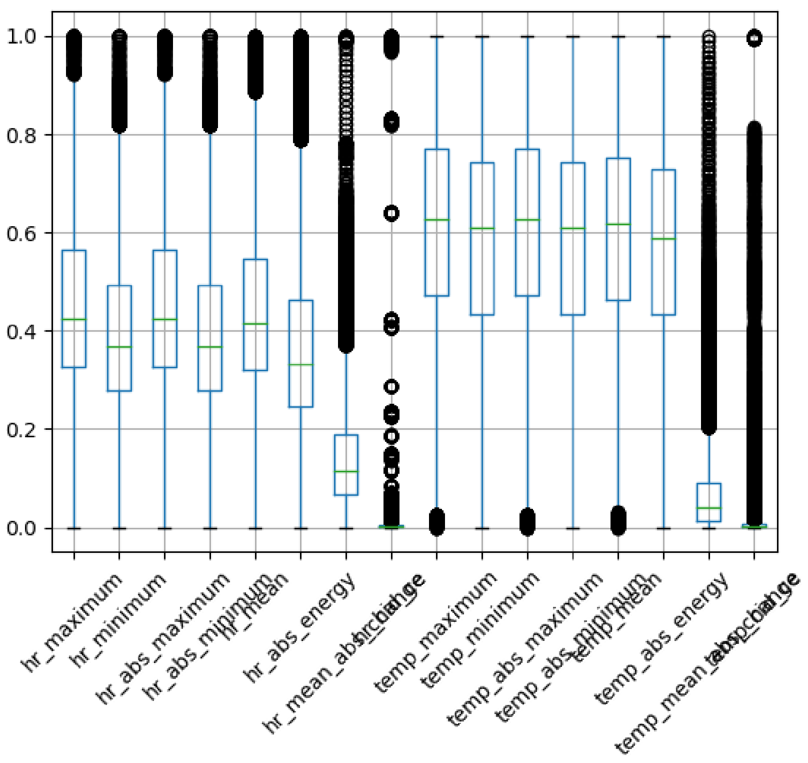


Figure 11 - Boxplot of scaled lag features

The boxplot in Figure 3 shows more outliers on the less correlated features. However, assuming that those outliers are not anomalies but are proper behaviour of the biometric measures, those have not been removed.

2.2.2 Hyperparameter Optimisation

The models have been optimized with a Leave-One-Group-Out Cross-Validation and Halving Grid Search, considering the 'ID' column as a grouping variable. The LOGO k-fold process allows us to consider each individual's behaviour by taking apart a subsample in terms of target iteratively. Then, the Halving Grid Search initially allows consideration of a subsample of observations and iteratively increases the number of samples for the candidates who received a higher score.

In addition, a random seed has been set to allow the exact reproduction of the random part, and the class weights have been selected to assign different weights to each class as an imbalanced dataset. The metric to inspect the model's behaviour was the weighted F1 score to find a perfect balance between Type-I and Type-II errors.

2.2.2.1 Logistic Regression

The Logistic Regression model, also known as logit, maximum-entropy classification, or log-linear classifier, is implemented as a regression analysis for binary systems and was first published by (Cox, 1958).

It is a parametric model where the prediction is based on the probability of being positive or negative through a sigmoid function. The primary assumption is that the dependent variables are supposed to be linearly composed to predict the probability for the independent variable by finding the optimal set of parameters.

Table 4 - Logistic Regression Hyperparameters

Penalty	L1, L2, Elastic-Net
Solver	Lbfgs, Liblinear, Newton-CG, Newton-Cholesky, Sag, Saga,
C	10 numbers [10^{e-3}, 1]

For this purpose, 180 candidates have been tried with the following hyperparameters: penalty refers to the regularisation methods, solver and regularisation strength.

The L1 regularisation adds an absolute magnitude value as a penalty term to the loss function; on the L2 regularisation, the magnitude value is squared. Both techniques avoid overfitting by selecting a subset of features for the former and applying weight decay for the latter.

The LibLinear solver (Fan et al., 2008) applies a Large Linear Classification with a coordinate descent algorithm by moving toward the minimum in one direction at a time; the L-BFGS (Richard B et al., 1994) is an optimization algorithm that approximates the Limited-memory Broyden–Fletcher–Goldfarb–Shanno algorithm, which belongs to quasi-Newton methods and approximate the second derivative matrix updates with gradient evaluation, Sag (M. Schmidt et al., 2017) and Saga (Defazio et al., 2014) solvers perform a Stochastic Average Gradient Descent algorithm which allows to faster the process by using a random sample of previous gradient values, and Newton-Cholesky which is a Newton method which uses an exact Hessian matrix (Pedregosa et al., 2011).

After the Halving Grid Search performed with LOGO Cross Validation, the optimal set of hyperparameters is the Lib-Linear algorithm with L1 regularisation of strength 1, giving a weighted F1 score of 93.88%.

Table 5 - Classification Report of Logistic Regression

	Precision	**Recall**	**F1-score**	**Support**
Baseline	0.85	0.73	0.79	18780
Stress	0.64	0.79	0.71	11220
Accuracy			0.75	30000
Macro avg	0.75	0.76	0.75	30000
Weighted avg	0.77	0.75	0.76	30000

Table 5 shows the most important metrics related to the final logistic regression model with the cross-validation method explained in the previous paragraph. It can detect 79% of recall regarding the stress class and 64% of precision with an F1 score of 71% on 11220 observations. The final accuracy is 75% for 30,000 observations.

2.2.2.2 Random Forest

The Random Forest algorithm is an ensemble method that creates a diverse classifier by introducing randomness in the classifier construction and was first published by (Breiman, 2001). The prediction of the ensemble is given as the averaged prediction of the individual classifiers.

Each tree performs various splitting considering a subset of independent samples and features to generate a pure node for correct prediction.

Table 6 - Random forest Hyperparameters

Number of estimators	5, 10, 20, 30
Maximum depth	5, 10, 20, 30
Minimum Samples Split	2, 5, 10

For this purpose, a restricted number of trees have been tried to ensure that the model will fix the available space inside the device. Then, to avoid overfitting, the proper maximum depth of each tree has been set up, and the minimum subsample of observations is considered to perform a split.

The optimal set of hyperparameters is a Random Forest model composed of 5 decision trees, a maximum depth of 10 sub-nodes and two minimum samples to perform each split, giving 74% of the weighted F1 score.

Table 7 - Classification Report of Random Forest

	Precision	**Recall**	**F1-score**	**Support**
Baseline	0.82	0.79	0.80	18780
Stress	0.67	0.70	0.68	11220
Accuracy			0.76	30000
Macro avg	0.74	0.75	0.74	30000
Weighted avg	0.76	0.76	0.76	30000

Table 7 shows the most important metrics related to the final random forest model with the cross-validation method explained in the previous paragraph. It is possible to detect 70% recall regarding the stress class and 67% precision with an F1 score of 68% on 11220 observations. The final accuracy is 76% for 30000 observations.

2.2.2.3 Gaussian Naïve Bayes

Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of features given the value of the class variable. They are appreciated in real-life problems for being fast and accurate (Pedregosa et al., 2011).

For this purpose, the algorithm requires the variance smoothing hyperparameter as the portion of the most significant variance of all features added to variances for calculation stability. One hundred values from 10^{e-9} to 10^{e9} were tried out, and the optimal value was 2.85, giving a weighted F1 score of 69%.

Table 8 - Classification Report of Gaussian Naive Bayes

	Precision	**Recall**	**F1-Score**	**Support**
Baseline	0.80	0.75	0.77	18780
Stress	0.62	0.69	0.65	11220
Accuracy			0.73	30000
Macro avg	0.71	0.72	0.71	30000
Weighted avg	0.73	0.73	0.73	30000

Table 8 shows the most important metrics related to the final Gaussian Naïve Bayes model with the cross-validation method. It is possible to detect 69% of recall regarding the stress class and 62% of precision with an F1 score of 66% on 11220 observations. The final accuracy is 73% for 30,000 observations.

2.2.2.4 Extreme Gradient Boosting

The extreme gradient boosting model, also called XGBoost, is a scalable tree-boosting system widely used by data scientists and provides state-of-the-art results on many problems. The model's authors proposed a novel sparsity-aware algorithm for handling sparse data and a theoretically justified weighted quantile sketch for approximate learning. Cache access patterns, data compression and sharing are essential for building a scalable end-to-end system for tree boosting. These can be applied to other machine learning systems as well. XGBoost can solve real-world scale problems using minimal resources (Chen & Guestrin, 2016).

The optimal set of hyperparameters for this model was found in two stages: in the first stage, the optimal solver was the tree algorithm with a 73.3% weighted F1 score among random forest, decision tree and linear algorithm; in the second stage, learning rate, eta and lambda regularisation on the XGBoost with decision tree as algorithm was found to be 0.03, 0.001 and 0.001 respectively to increase the weighted F1 score of +0.3%. Also, in this case, the model was trained by assigning the proper weight to the minority class, with a scale positive weight of 1.67 (number of negative labels/number of positive labels).

Table 9 - Classification Report of Extreme Gradient Boosting

	Precision	**Recall**	**F1-Score**	**Support**
Baseline	0.85	0.74	0.79	18780
Stress	0.64	0.78	0.70	11220
Accuracy			0.75	30000
Macro avg	0.74	0.76	0.75	30000
Weighted avg	0.77	0.75	0.76	30000

Table 9 shows the most important metrics related to the final Extreme Gradient Boosting model with the cross-validation method. It is possible to detect 78% of recall regarding the stress class with an F1 score of 70% on 11220 observations. The final accuracy is 75% for 30,000 observations.

2.2.3 Model Selection

The medical field of application usually refers to metrics other than the classical accuracy percentage, like Type-II errors. Those errors refer to true positive labels, which have been predicted as negative; this behaviour is most dangerous in real-life applications rather than the contrary.

In addition, especially when dealing with an imbalanced dataset where generating synthetic data for balance has not been approached, one of the essential tools to inspect Type-I and Type-II errors is the confusion matrices: it represents a table that collects the ground truth and the predicted labels. In this way, it is possible to determine the amount of misclassified observations and their grade of importance in the recall.

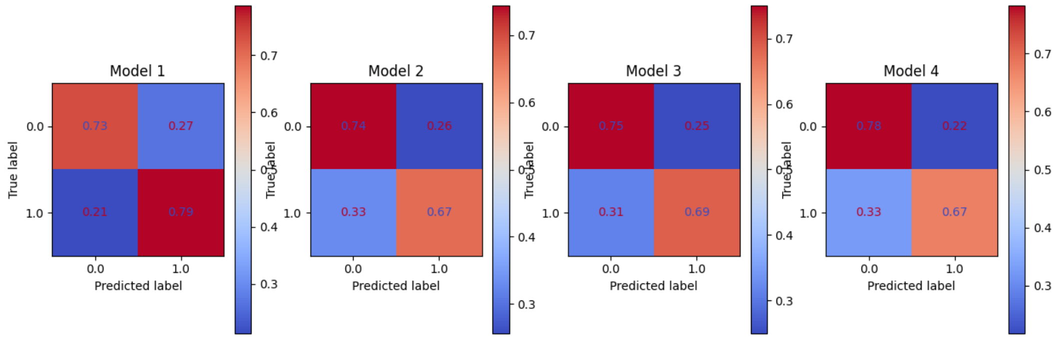


Figure 12 - Confusion Matrices of models

From left to right, the confusion matrices in Figure 10 refer to Logistic Regression, Random Forest, Gaussian Naive Bayes and Extreme Gradient Boosting. The matrices express normalized values regarding ground truth; thus, 100% is reached for each row.

In this case, Logistic regression has an accuracy of 79% on the anxiety class with 21% of misclassified observations as false negative; Random Forest has an accuracy of 67% on the stress class with 33% of misclassified observations as false negative; Gaussian Naive Bayes have an accuracy of 69% on the stress class with 31% of misclassified observations as Type-II errors and XGBoost has an accuracy of 67% on the positive class with 33% of Type-II error.

At this first stage, the optimal model appears to be the Logistic Regression classifier. However, another critical aspect is to inspect the stability of the model in order to understand if those metrics, despite the cross-validation method, have a high standard deviation or if the model can be considered a robust model and find a trade-off between those characteristics.

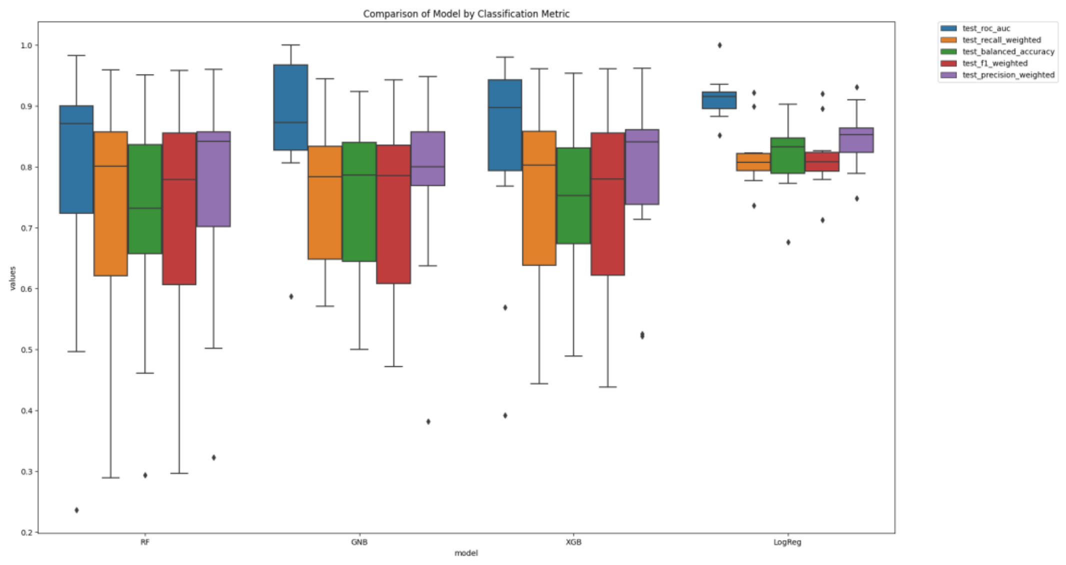


Figure 13 - Comparison of Models Metrics

The plot in Figure 11 represents a series of boxplots divided by different metrics: weighted F1, weighted precision, weighted recall, weighted ROC AUC, and weighted balanced accuracy.

Generally, the boxplot extends from the minimum to the maximum values, and the points represent the outliers. In addition, the box represents the interquartile range, such as observations between 25% and 75% of the total amount. The line inside the box represents the mean value of the whole observation.

In this case, we can see that the acceptable behaviour of the Logistic Regression Model inspected in the previous part of this work is confirmed in the plot above as it is the model with the lowest standard deviation, leading to a more stable prediction. Visually, the Random Forest classifier has the highest standard deviation, followed by XGBoost and Gaussian Naive Bayes.

The final model was selected regarding a trade-off between metrics and stability from the results of a bootstrap method: in this case, ten observations were selected to iteratively test the model with the cross-validation method by reinserting the samples each time.

Table 10 - Metrics Comparison with Bootstrap Method

	ROC AUC		**Recall**		**Accuracy**		**F1**		**Precision**	
	std	**mean**	**Std**	**mean**	**std**	**mean**	**std**	**mean**	**std**	**mean**
Model										
GNB	0.12	0.87	0.13	0.76	0.15	0.74	0.17	0.73	0.16	0.77
LogReg	0.04	0.91	0.05	0.82	0.06	0.82	0.06	0.81	0.05	0.85
RF	0.23	0.77	0.21	0.72	0.20	0.70	0.21	0.72	0.20	0.75
XGB	0.19	0.82	0.17	0.75	0.15	0.73	0.17	0.74	0.15	0.78

As we can see from the results above, the optimal model in terms of standard deviation and mean of the overall metrics are confirmed by the Logistic Regression Classifier with the following results: 91% of AUC (± 0.04), 81% of F1 score (± 0.06), 82% of recall (± 0.05), 82% of accuracy (± 0.06), 85% of precision (± 0.05).

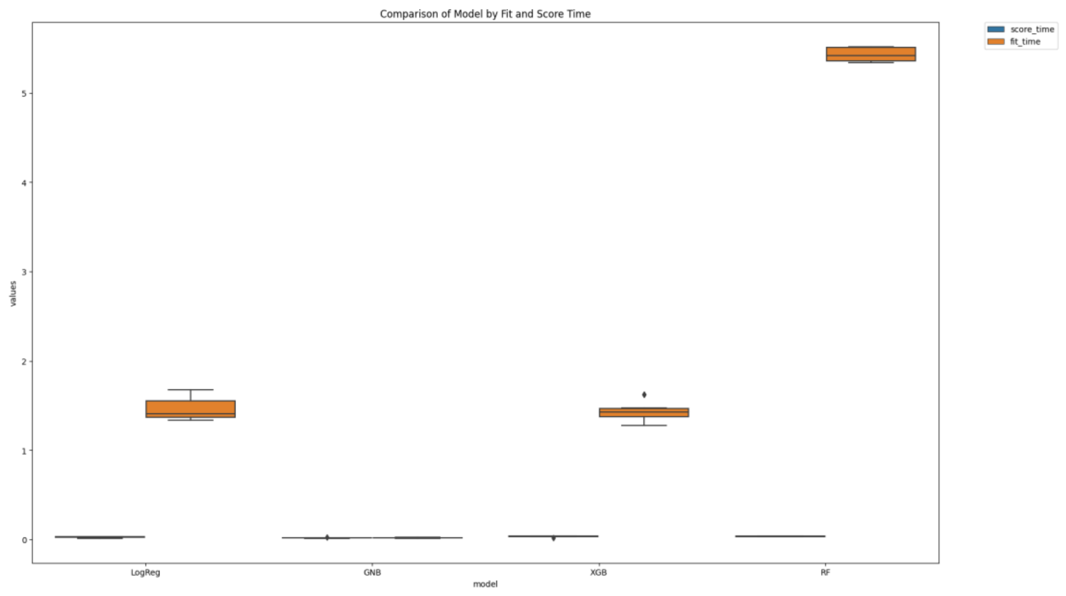


Figure 14 - Fit and Score Time

Inspecting the time of fitting and scoring is an important aspect, as the final model will be deployed on an embedded device. Given the same bootstrap results as above, obtained with the cross-validation method by Leave-One-Group-Out, the Gaussian Naive Bayes appears to be the fastest model in terms of fitting and scoring, followed by Extreme Gradient Boosting, Logistic Regression and Random Forest, which is the slowest model.

As a final test, the Mc Nemar's test (McNemar, 1947) for paired nominal data has been applied to compare the behaviour of the Logistic Regression and Gaussian Naive Bayes models regarding significant changes between the two models.

Table 11 - Mc Nemar's table (chi2: 116.04 - p-value: 4.65^{e-27})

	Gnb correct	**Gnb wrong**
LogReg correct	19284	3338
LogReg wrong	2513	4865

From Table 11, we can see the contingency table between predictions, produced through cross-validation technique, from Gaussian Nayve Bayes and Logistic Regression Classifier: as we can see, Logistic Regression made 2513 wrong misclassified predictions that the Gaussian Naive Bayes made correct; instead, the Gaussian Naive Bayes made 3338 wrong misclassified predictions that the Logistic Regression made correct. Given this behaviour, the final optimal model selected for this purpose was the Logistic Regression Classifier.

This behaviour is confirmed by the chi2 with a p-value significantly lower than 0.05, which leads to accepting the null hypothesis that there is a significant difference between the two models.

2.2.4 Model Tuning

The Receiver Operator Curve expresses the ratio between the FPR and TPR to maximize the AUC. In addition, the ROC plot is a valuable method of model tuning that allows adjusting the threshold in binary classification tasks, considering the threshold nearest to the top-left margin of the plot as a value that can best maximize the AUC value.

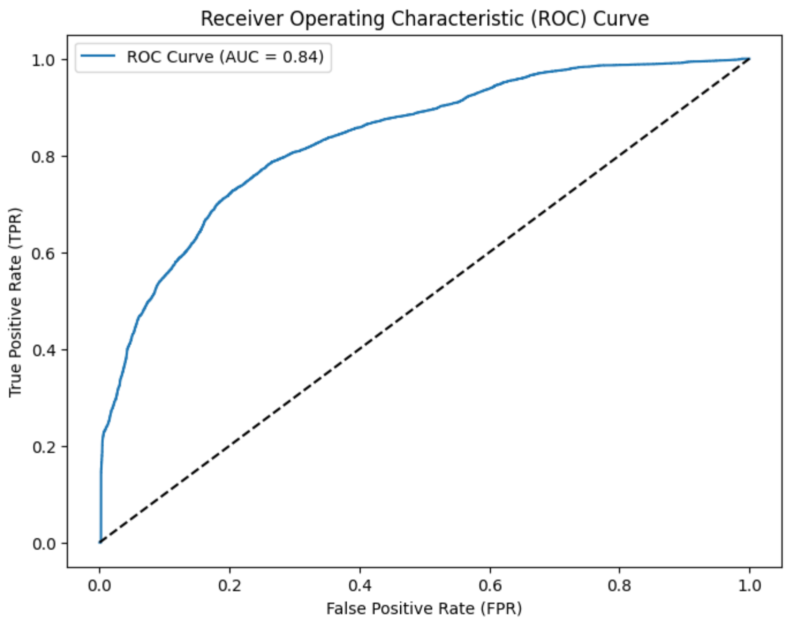


Figure 15 - ROC Curve (AUC = 0.84)

The plot in Figure 16 depicts the ratio between TPR and FPR with an AUC value of 0.84 over a maximum of 1. From this plot, the threshold that allows maximising the TPR while keeping FPR low is 0.59; this means that probability lower than 59% should be considered as baseline predictions, anxiety state prediction instead.

Table 12 - Classification Report of the tuned optimal model

	Precision	**Recall**	**F1-Score**	**Support**
Baseline	0.83	0.79	0.81	18780
Anxiety	0.68	0.73	0.70	11220
Accuracy			0.77	30000
Macro avg	0.75	0.76	0.76	30000
Weighted avg	0.77	0.77	0.77	30000

Although the tuned model increased the accuracy by +2% and the weighted F1 score of +1%, we can see an inverse behaviour on precision and recall, leading to the increase of Type-II errors. For this reason, the final model has been left as it is without tuning the threshold.

2.2.5 Model Interpretability

Given the Logistic Regression as the optimal model for this purpose in terms of metrics, misclassified observations and timing, inspecting the optimal model can lead to further essential considerations about the model behaviour, thanks to several techniques that can improve the interpretability of ML models.

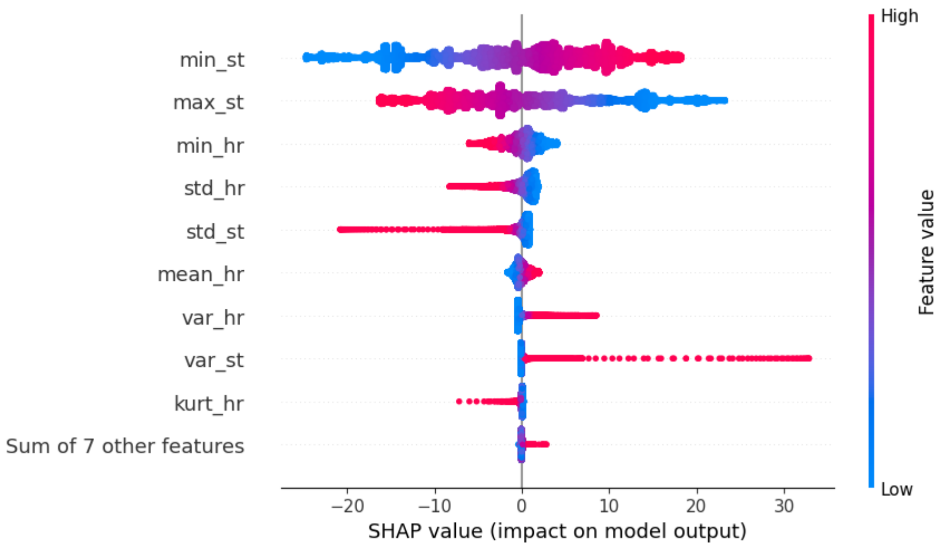


Figure 16 - Shapley Values of Logistic Regression Model

The plot in Figure 16 represents all the Shap values grouped by features on the y-axis: the features are ordered from the most important to the least important, and redder points represent higher values of each feature. The plot above shows that minimum and maximum skin temperature values are the two most important features, followed by BPM's minimum and standard deviation. In addition, higher values of minimum skin temperature lead to a higher impact on stress detection; obviously, the maximum skin temperature values have the opposite behaviour. Finally, higher values of minimum and standard deviation BPM lead to a negative impact on stress detection. Values near 0 mean that that feature does not contribute very well to detecting the stress state.

2.3 Deep Learning

Deep learning is a branch of machine learning that has developed over the past few years due to the rise of Big Data and the development of IoT technologies capable of collecting an ever-increasing amount of data.

The main component of Deep Learning is the neural network, a parametric model inspired by the brain's biological structure (see Figure 17): receiving various inputs from dendrites, the human brain can process these inputs inside the nucleus and recreate a logical and comprehensible output from the axon and axon terminal, providing answers and connections via a synapse. Similarly, the neural network receives input, consisting of video, images or time signals, and processes it thanks to the CPU to provide an output that flows to other neurons.

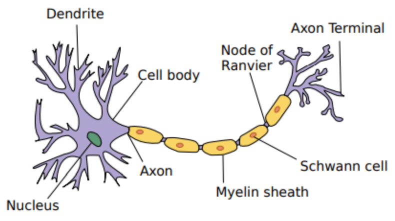


Figure 17 - The real neuron (Zhang et al., 2023)

The first genuinely neural network model was the multilayer perceptron, also known as the dense network: this model processes input through one or more layers, composed of one or more neurons, that are fully connected; this complexity is increased by the use of activation functions in order to insert a non-linearity to the problem that allows it to simulate real-life problems.

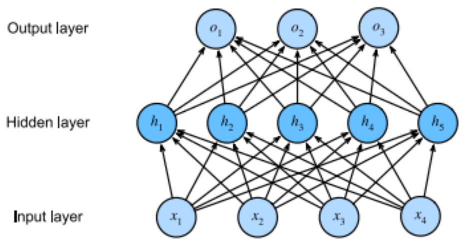


Figure 18 - The Multilayer Perceptron (MLP) with one hidden layer and five hidden units (Zhang et al., 2023)

To date, there are various models of neural networks; among the most famous and used in the field of research are convolutional neural networks, fully-connected networks, recurrent neural networks, Generative Adversarial networks, and transformers.

This project uses a pre-trained convolutional neural network on Gramian Angular Summation Field images generated from the time series to check the powerful instrument of the convolutional activation map.

2.3.1 Data Preprocessing

For this part of the analysis, a method for encoding time series has been used to use the powerful tools of computer vision to inspect them and visually understand hidden structures and patterns.

In particular, the GASF method (Wang & Oates, 2015) is capable of producing images from time series by representing them in polar coordinates instead of the typical cartesian coordinates in such a way as to identify the temporal dependencies within different time intervals.

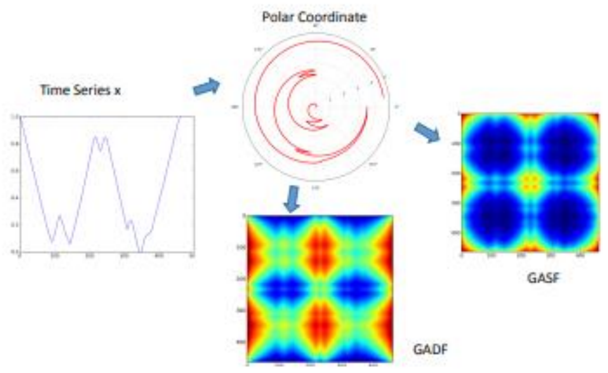


Figure 19 - Gramian Angular Summation and Differentiation Field Pipeline (Wang & Oates, 2015)

In this case, the time series has been split into two parts for each individual: 15 images were produced for the baseline class, and 15 images were produced for the stress class.

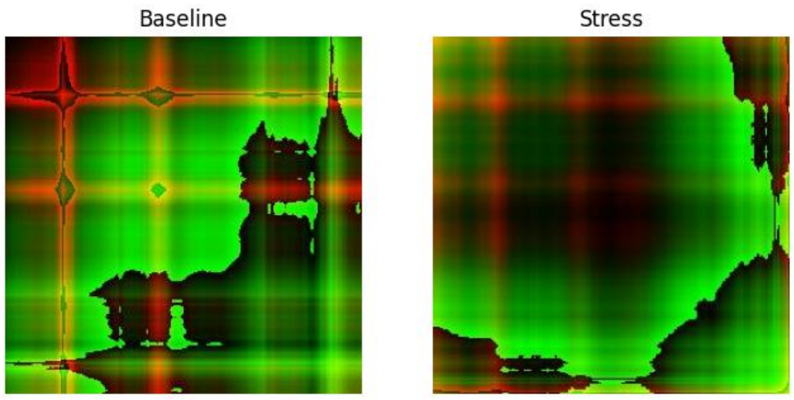


Figure 20 - Gramian Angular Summation Field Images

The resulting images represent the square Garmian matrix of a predefined size of 250x250px; the first channel represents the heart rate sequence, the second channel represents the skin temperature sequence, and the third channel is black, added to fit the standard requirements of network and display programmes.

Finally, to apply some diversity in every (3, 250, 250) image, a data augmentation technique has been applied:

- Random horizontal flip with 100% probability,
- Random vertical flip with 100% probability,
- Random rotation of 30°,
- Image normalization

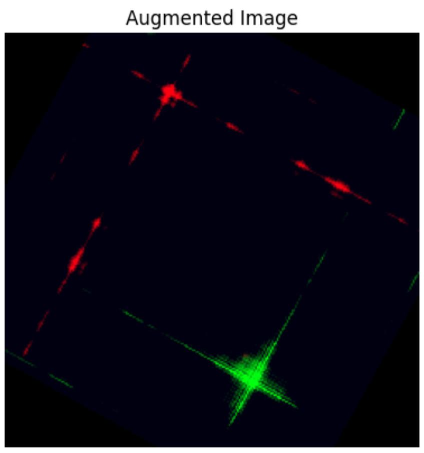


Figure 21 - Augmented GASF Image

2.3.3 Model Architecture

The Convolutional Neural Network is a state-of-the-art tool of computer vision: the CNN architectures are compelling and optimized to process images; the leading player among the CNN architectures is the convolutional operation, which allows extracting the most critical features from an input image by lowering the number of parameters, compared to an FC network (Bengio, 1994).

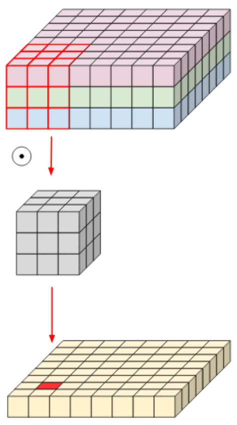


Figure 22 - Convolutional Kernel on an RGB Image (Eli Bendersky's website)

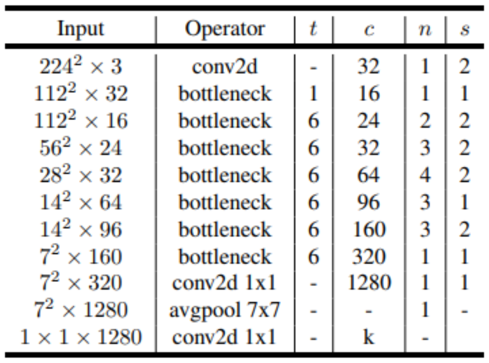
The basic idea of the convolution is the application of a sliding window, also expressed as “filter” or “kernel”, along the input image, performing a dot product between the filter and the input image to output a cross-correlation between pixels. Considering a multichannel image, called RGB image (see Figure 22), a single filter is applied along all the channels to produce, through the dot product, a single output for each step[8]. The hyperparameters selected by the user are the stride, which represents the step along which the filter roll throws the input image, and the padding, which represents the additional border in order to perform the same output dimension and to process the corner part of the image, and the dilation, which is additional space within the filter, to capture a larger receptive field on the input image.

[8] Eli Bendersky’s Website | Depthwise Separable Convolution for Machine Learning

Thus, the convolution is based on three main ideas: local receptive field, shared weights and spatial subsampling. Each layer unit receives inputs from a set of units located in a small neighbourhood to capture the correlation among pixels, forcing those pixels to share the same weights. Finally, as the goal of the convolution is to increase the resolution of the image by applying several filters, resulting in a multiple-channel feature map, each convolutional layer is usually followed by a layer that performs a local averaging or subsampling called the MaxPooling layer (Bengio, 1994), intending to reduce the spatial dimensions.

As CNN often requires a significant amount of data to produce optimal results, using a pre-trained network was a fundamental choice, as the model cannot capture high-level features from a low amount of images.

The MobileNetV2 (Sandler et al., 2018) has been selected as an optimal trade-off between efficiency and velocity through the depthwise separable convolution, inverted residual blocks and linear bottleneck techniques, resulting in about 3M trainable parameters.



Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

Figure 23 - MobileNetV2 Architecture

The main reason for such low latency is the usage of depth-wise separable convolution (Chollet, 2016), first introduced into the deep-learning with the Xception network: computations are reduced due to the application of a kernel of size 3 for each channel of the input, followed by a point-wise convolution applied on the concatenated input resulted from the depth-wise kernel. In this way, the correlation between pixels and spatial features reduction is performed on a previous stage and depth is managed by a kernel of size one on a lower dimensional feature map (Simone, 2023).

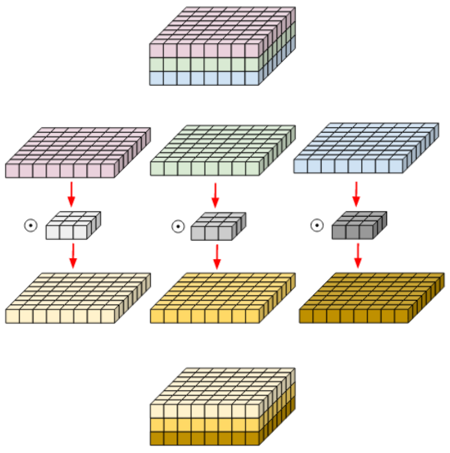


Figure 24 - Depthwise Separable Convolution (Eli Bendersky's Website)

Then, the concept of residual connection allows us to avoid the vanishing gradient problem as a consequence of iteratively multiplying very low weights. In the residual connection blocks, first implemented in the ResNet architecture (He et al., 2015), the previous wide convolutional layer is connected to the following wide convolutional layer. On the contrary, the inverted residual block (see Figure 25) implemented in the MobileNet architecture allows concatenating the previous narrow block to the following narrow block, resulting in fewer parameters.

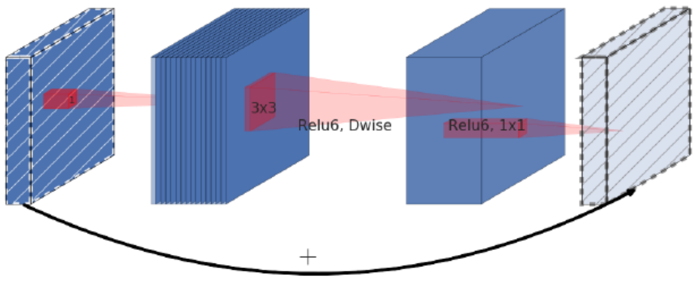


Figure 25 - Inverted Residual Block

Finally, a linear layer is inserted before the last convolutional layer of each block to avoid the loss of information from a non-linear activation function.

The model has been trained on the resulting 30 images with a K-Fold Cross-Validation technique using the following hyperparameters: Stochastic Gradient Descent optimization algorithm with a learning rate 1e^{-4}, allowing a small change step on the updating weights.

The matrix in Figure 26 represents the comparison between ground truth and predicted responses on the train set images, producing an accuracy of 56% on the test set.

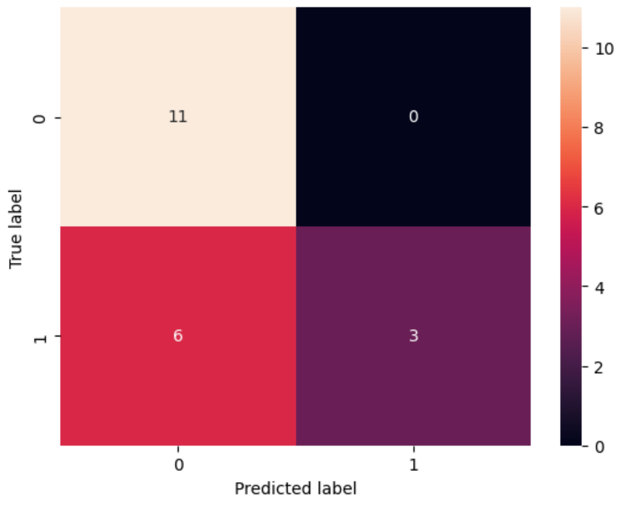


Figure 26 - Confusion Matrix of MobileNetV2

The accuracy metric mainly regards the baseline class, which does not contain misclassified images. Instead, the model could predict three images belonging to the stress class out of 9, with six Type-II errors.

2.3.4 XAI

During the last few years, the evolution of computer vision has grown together with the need to transform algorithms into an understandable form. CAM is a technique of visual explanation of the CNN decisions; in particular, Gradient-weighted CAM (Selvaraju et al., 2016) uses the gradients of the target of a specific convolutional layer to produce a localization map highlighting the most activated zone.

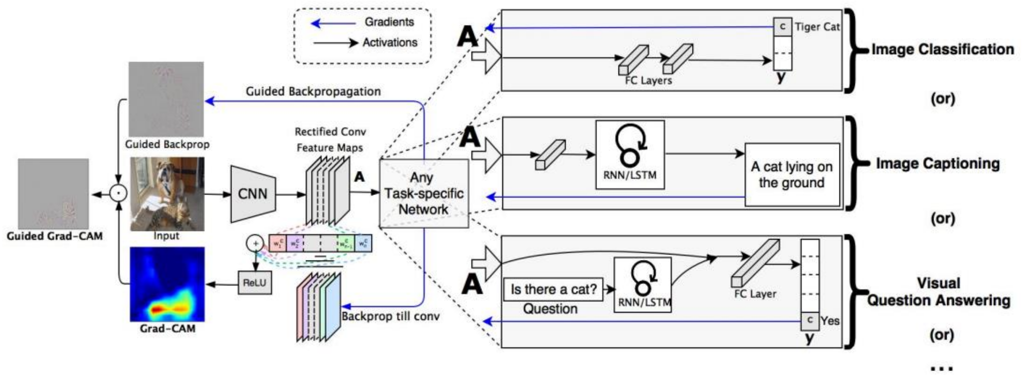


Figure 27 - Grad-Cam Overview (Selvaraju et al., 2016)

For this purpose, a Grad-Cam has been applied to the input image belonging to the stress class of individual 14, on which the last convolutional layer of the MobileNetV2 has been activated to understand why the model is leading to predict the stress class.

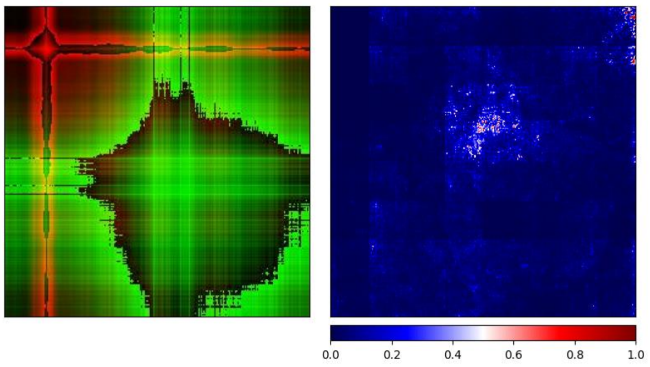


Figure 28 - Guided Grad-Cam of Stress class

In particular, a Noise Tunnel with SmoothGrad technique (Smilkov et al., 2017) has been applied to the Guided Grad-Cam activation, providing ten samples with a standard deviation of 0.3. This technique allows the addition of a Gaussian noise to each input to check for sanity. Redder pixels represent the images' most activated parts; in Figure 22, we can compare the original image and the CAM image: the model concentrates on the central and upper parts to predict the stress class on the subject's Gramian Angular Summation Field image.

2.4 Conclusions

In the previous chapter, the powerful convolution tool has been explored and tried out, trying to maximize the results on the time series. Thanks to the high amount of research in computer vision, sequences coming from signals or video can be processed as images while maintaining the time relation within data.

However, one of the essential requirements of DL is a high amount of good-quality data to properly feed the network and let it learn the most hidden features that humans sometimes cannot detect.

The dataset used for this purpose contains information from 15 individuals, and, despite the try to improve results, quality and time-related conditions, the images were too low to be processed optimally by a convolutional network, resulting in poor accuracy and low-level features detected from the CAM.

Conversely, techniques like statistical, spectral or time-spatial related features extraction from a sequence allow the improvement of ML algorithms while being easy to deploy for low-power embedded devices.

In this case, the Logistic Regression algorithm appears to be the optimal model among the ML and DL models, which have been tried out for the trade-off between Type-II errors decreasing, stability and velocity.

Chapter 3 - Deployment

Once the optimal model has been identified, the next step is the realization of the device's hardware and software parts, consisting of two sensors that allow the collection of biometric parameters and, after the preprocessing pipeline, use the algorithm to produce an appropriate response for the recorded parameters.

To obtain measurements similar to the dataset in which the algorithm was trained, sensors with similar characteristics to the device used in the reference research project were selected.

In addition, the algorithm identified as optimal and the preprocessing phase in the previous chapter were translated into C++ language and incorporated as a library to ensure similar behaviour.

3.1 Hardware

3.1.1 Development Board

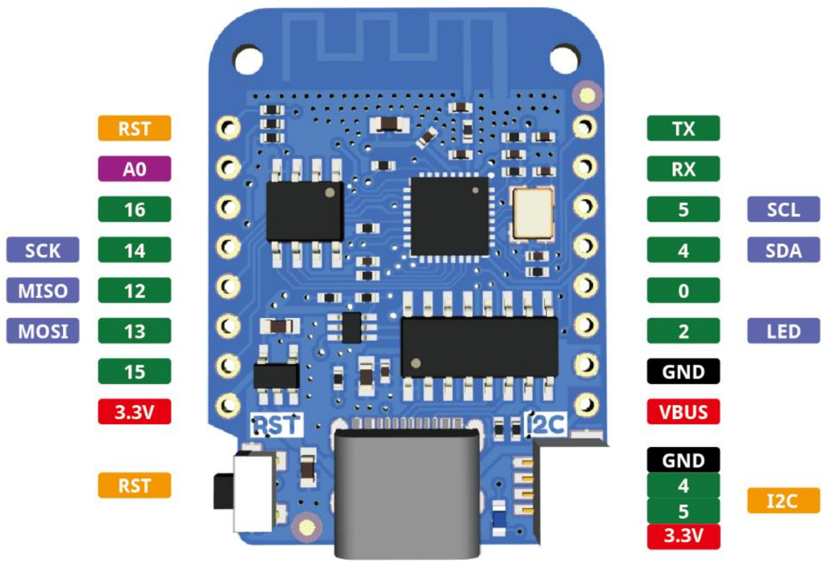


Figure 29 - Development Board Overview

The device selected for this project was the Wemos D1 Mini, which contains an ESP8266EX microprocessor. It is a classic IoT development board designed and produced by Wemos[9] with 3.3/5 operating voltage, 4MB of flash RAM, 11 GPIO, one analogue input pin and one I2C bus.

3.1.2 Photoplethysmograph Sensor

The sensor used to collect heart data was a PPG, a low-cost, non-invasive optical method to detect changes in BVP that can be easily integrated into wearable sensors and used in a wide range of medical devices to measure oxygen saturation and blood pressure.

PPG collects reliable information from organs under the control of the Autonomous Nervous System, which is widely affected by emotional states (Rinella et al., 2022).

The produced waveform comprises a pulsatile physiological waveform attributed to each heartbeat and a slowly varying baseline attributed to respiration, sympathetic nervous system activity and thermoregulation.

Recent years have seen a growing usage of this technique due to the demand for low-cost, simple and portable technology for primary care and clinical communities and the advancement of computer-based pulse wave analysis techniques (Allen, 2007).

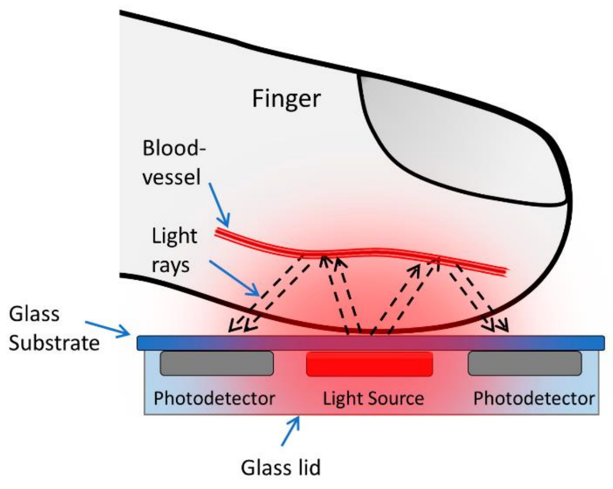


Figure 30 - Acquisition of the PPG from the reflection method (Allen, 2007)

[9] Wemos.cc

The sensor used for this purpose was the MAX30101 from Pimoroni with a microchip from Maxim Integrated[10], which has a photodetector and three types of LED: InfraRed, red and green LED. The sensors work as explained in Figure 25: the sensor positioned on the wrist is set up to emit a certain amount of green light, which is absorbed by the blood; thus, at every blood pulse, the photodetector captures a peak of the signal which corresponds to reflected light (Artz, 2020).

Then, the Peripheral Beat Amplitude algorithm, implemented by Maxim Integrated[11], is applied to the collected PPG values to compute the Beats Per Minute.

3.1.3 Peripheral Skin Temperature Sensor

Peripheral skin temperature is another important measure for monitoring an individual's emotional state: it is linked to the sympathetic nervous system and reflects blood flow in the veins. When the sympathetic nervous system is activated, triggering the so-called "fight-or-flight" feeling, blood flow increases to vital organs, increasing heart and core body temperature and decreasing peripheral skin temperature.

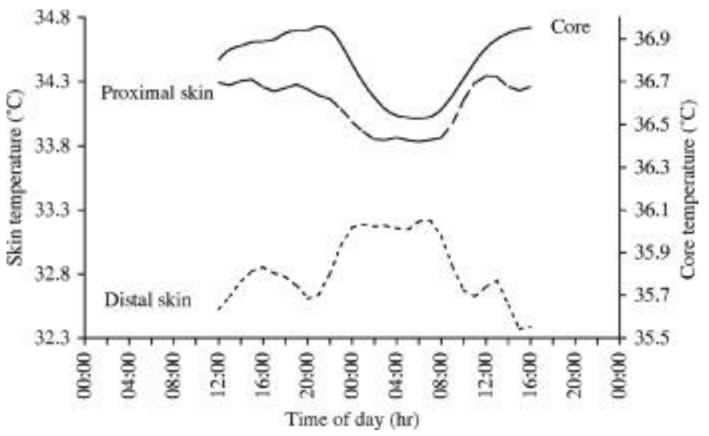


Figure 31 - Human Temperature Curve (Te Lindert & Van Someren, 2018)

[10] MAX30101 | Max Integrated

[11] Peripheral Beat Amplitude Algorithm

As it is possible to see in Figure 31, peripheral skin temperature increases during sleep and relaxing tasks while core temperature decreases.

The sensor used for this purpose is an infrared contactless thermometer with a sensitive thermophile detector with a resolution of ± 0.02 °C and a precision of ± 0.2 °C on the temperature in the range [36, 38].

3.2 Software

The main objective of the software development is to replicate as much as possible the selected methods of the previous chapter.

The first part of the code involves the preprocessing of the collected signals: firstly, the blood volume pulse is collected from the reflected green light of the wrist, and the signal is then processed with the Peripheral Beat Amplitude to detect peaks, which refers to BPM; secondly, the peripheral skin temperature is collected by measuring throw the IR sensor on the wrist. Then, the signals are smoothed with an average window of size 3 to remove noise.

The second part of the code involves the application of the pipeline for feature extraction and standardization: the two signals are collected inside a vector in which the pipeline is applied, applying a window of size 60, which allows the extraction of the selected lag features from both signals generating a total of 16 features which will be standardized with a min-max scaler.

Finally, the features will feed the classifier, generating a response according to the biometric measures.

It is essential to notice that the overall process requires a few seconds to show optimal measures and response due to the size of the rolling windows.

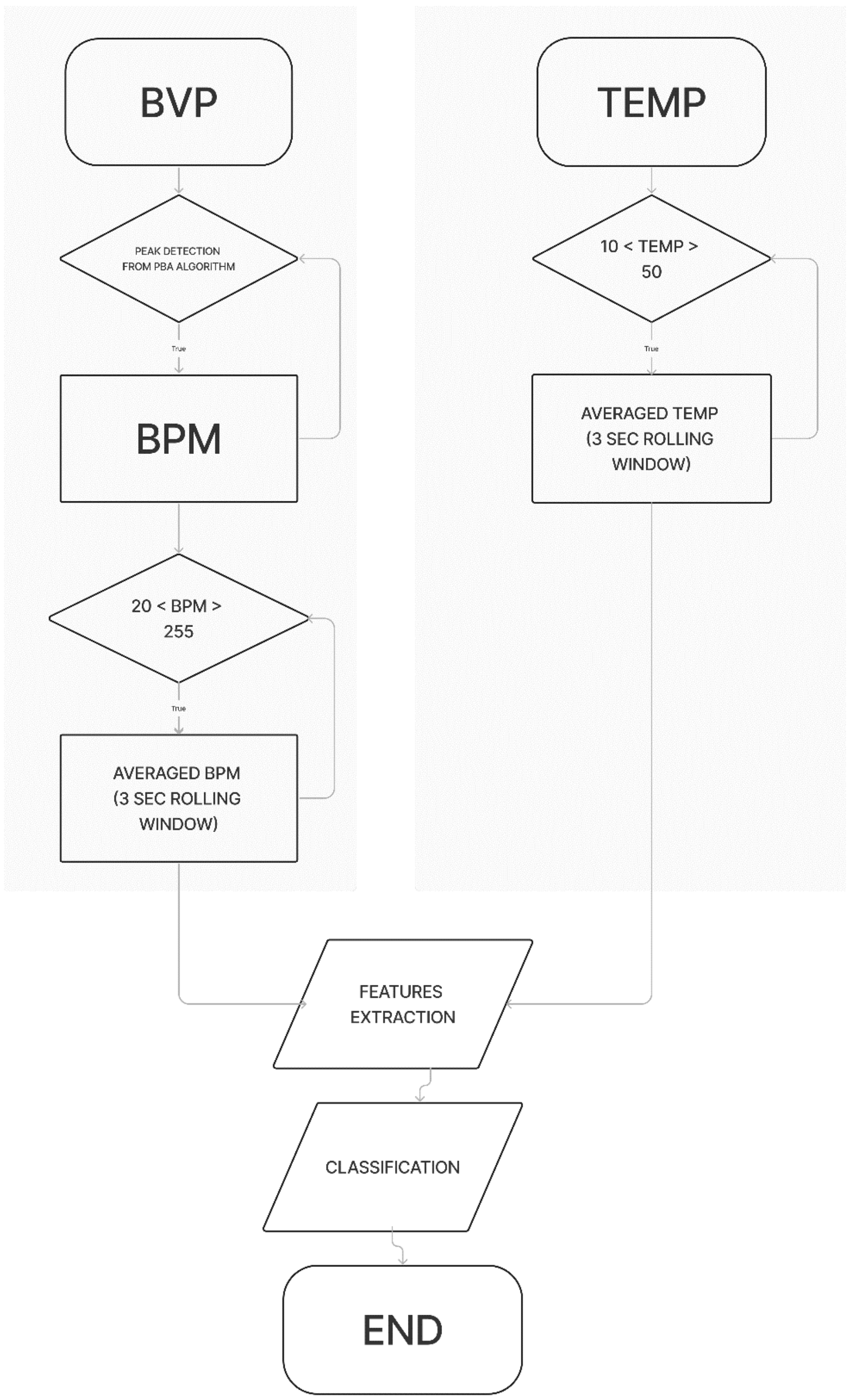
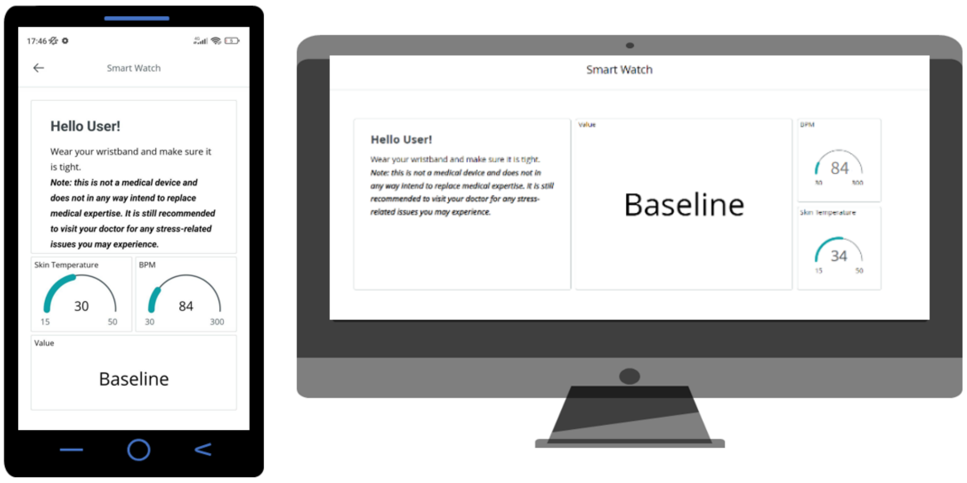

BVP
PEAK DETECTION FROM PBA ALGORITHM
True
BPM
20 < BPM > 255
True
AVERAGED BPM (3 SEC ROLLING WINDOW)
TEMP
10 < TEMP > 50
True
AVERAGED TEMP (3 SEC ROLLING WINDOW)
FEATURES EXTRACTION
CLASSIFICATION
END

Figure 32 - Code Diagram Flow

The most crucial part of making the device an IoT device is the WiFi connection: the signal will be sent to the Arduino Cloud, which will show the results in a dashboard, visualizable from both the web and smartphone. Figure 33 shows the responsive preview for web and smartphone dashboard visualization: the values are displayed in integer form inside the gauges, the string represents the response given by the algorithm, and the note underly the fact that the device will not in any way replace a domain expert advice and user alone should not use the device for medical purposes.



Hello User!

Wear your wristband and make sure it is tight.

Note: this is not a medical device and does not in any way intend to replace medical expertise. It is still recommended to visit your doctor for any stress-related issues you may experience.

Hello User!

Wear your wristband and make sure it is tight.

Note: this is not a medical device and does not in any way intend to replace medical expertise. It is still recommended to visit your doctor for any stress-related issues you may experience.

Figure 33 - Responsive Dashboard Visualization

Chapter 4 - Experimental Design

The experiment investigates participants' physiological responses and subjective stress levels during a relaxing and cognitive math quiz task. The investigation seeks to collect heart rate and skin temperature data using the embedded device described above and administer the LSAS-SR questionnaire.

Four volunteers (3 female, 1 male) have been recruited for the study by sharing the manifest in Figure 34 through community advertisements, online platforms, or university student populations. The inclusion criteria for participants will involve individuals aged 20-37 years with no known cardiovascular or respiratory conditions and who have not made any sports activity and did not drink alcohol or coffee at least one hour before the experiment.

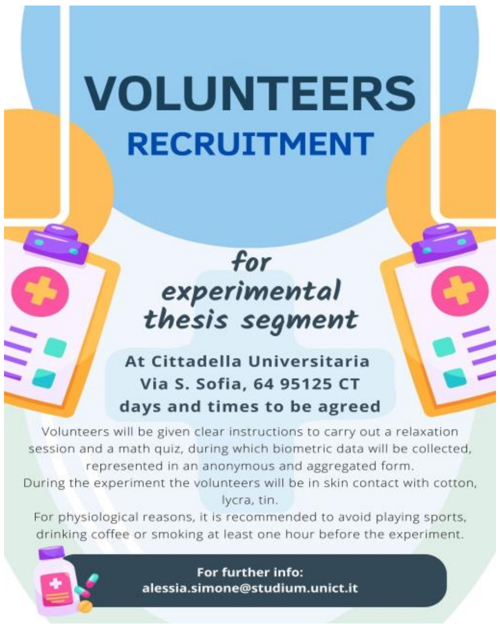

VOLUNTEERS
RECRUITMENT
for
experimental
thesis segment
At Cittadella Universitaria
Via S. Sofia, 64 95125 CT
days and times to be agreed
Volunteers will be given clear instructions to carry out a relaxation session and a math quiz, during which biometric data will be collected, represented in an anonymous and aggregated form.
During the experiment the volunteers will be in skin contact with cotton, lycra, tin.
For physiological reasons, it is recommended to avoid playing sports, drinking coffee or smoking at least one hour before the experiment.
For further info:
alessia.simone@studium.unict.it

Figure 34 – Manifest

The experiment adheres to ethical guidelines and regulations regarding participant confidentiality, informed consent, and data handling, during which each participant has been informed of their right to withdraw from the study at any time without consequences. Any potential risks to participants' well-being have been minimized, and appropriate measures were in place to address any distress or discomfort experienced during the experiment. It followed a precise protocol which has been announced to every subject, and followed the following steps:

1. Informed Consent: Participants will receive an informed consent form outlining the study's purpose, procedures, and potential risks and benefits. They will be able to ask questions and provide written consent before participation.
2. Pre-Experiment Instructions: Participants will receive detailed instructions on the tasks and how the embedded device will operate for data collection. Before the experiment, they will be instructed to avoid consuming stimulants (e.g., caffeine and alcohol) and to play sports activities to minimize potential confounding factors. In addition, they have been informed about skin contact materials for allergy reasons.
3. Relaxing Task: Participants will be instructed to relax and listen to nature sounds for 5 minutes (Akitan, 2010). HR and ST data will be collected using the embedded device and related sensors during this time, in addition to the time stamp and predicted labels for each timestep from the model inside the device.
4. Math Quiz Task: Participants will be given a set of math quizzes to solve for 5 minutes[12]. The examinations will be of moderate difficulty in inducing cognitive engagement. HR and ST data will be collected during this task as well.
5. LSAS-SR Questionnaire (Liebowitz, 1987): Participants will receive the LSAS-SR questionnaire to assess their personal anxiety levels after completing the tasks. The questionnaire will be self-administered and include 24 items related to social anxiety symptoms and situations: in particular, there are 12 questions for both avoidance and anxiety situations where response from 0 to 3 means never for the former and severe for the latter. The score range can be computed by summing up all the responses

[12] https://www.math-only-math.com/math-quiz-b.html

and can be interpreted as follow: < 55 low social phobia, 55 – 65 moderate social phobia, 65 – 80 marked social phobia, 80 – 95 severe social phobia, > 95 very severe social phobia.

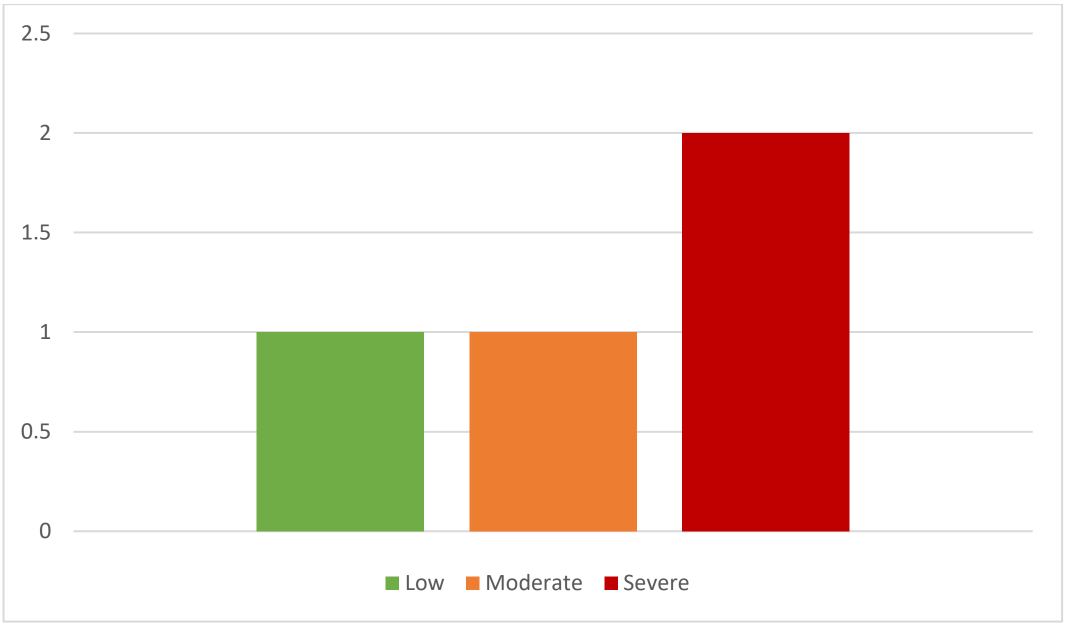
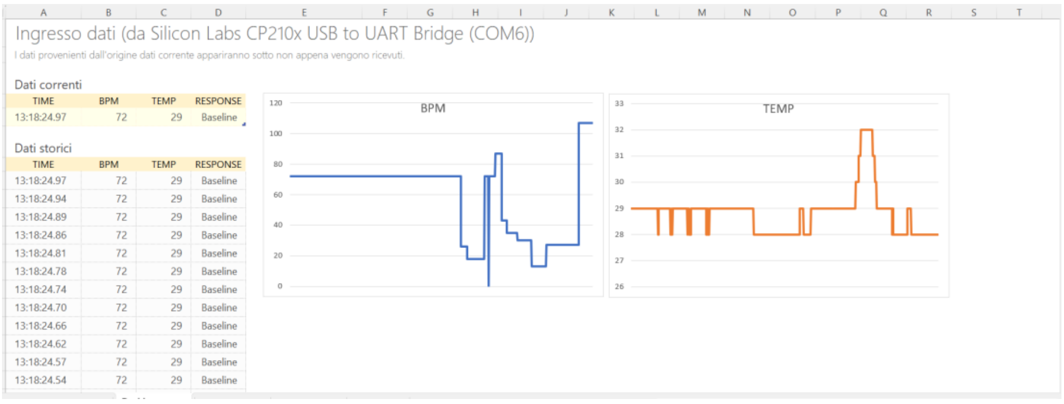


Figure 35 - LSAS-SR Experimental Score Range

6. Debriefing and Data Collection: Participants can ask questions and provide feedback on their experience. The collected heart rate and skin temperature data will be stored securely for further analysis.

The sensor data has been collected with the DataStreamer tool of Excel[13], a powerful tool that enables the connection from the real world to the IoT.

[13] DataStreamer tool | Microsoft Excel


Ingresso dati (da Silicon Labs CP210x USB to UART Bridge (COM6))

I dati provenienti dall'origine dati corrente appariranno sotto non appena vengono ricevuti.

Dati correnti

TIME	BPM	TEMP	RESPONSE
13:18:24.97	72	29	Baseline

Dati storici

TIME	BPM	TEMP	RESPONSE
13:18:24.97	72	29	Baseline
13:18:24.94	72	29	Baseline
13:18:24.89	72	29	Baseline
13:18:24.86	72	29	Baseline
13:18:24.81	72	29	Baseline
13:18:24.78	72	29	Baseline
13:18:24.74	72	29	Baseline
13:18:24.70	72	29	Baseline
13:18:24.66	72	29	Baseline
13:18:24.62	72	29	Baseline
13:18:24.57	72	29	Baseline
13:18:24.54	72	29	Baseline

Figure 36 - Excel DataStreamer Dashboard

As Figure 36 shows, a two-way data transfer streams live data from a microcontroller to an Excel sheet and sends back data to the microcontroller. This tool allows the collection of up to 10 channels with different frequencies and can register the screen and save the collected data into a ".csv" file.

Data from each subject has been collected and merged to produce a single dataframe, ready to explore. Before starting the exploratory data analysis, the signal-to-noise ratio (Yuan et al., 2019) was computed to compare the desired information with the undesired background information, called noise. As the SNR is higher than 0, it means that the signal information is relatively higher than the noise: in this case, the average heart rate SNR was 21.28 dB (±3.12), and the average skin temperature SNR was 39.54 (±4.54), which represent a good quality of signal giving enough flow of information.

Table 13 - Summary Statistics of Experiment Dataframe

	count	mean	std	min	25%	50%	75%	max
BPM	2815.0	70.610302	8.387631	50.00	66.00	72.00	77.00	87.00
ST	2815.0	33.581208	0.588427	32.18	33.18	33.51	33.78	35.76
Predicted	2815.0	0.039787	0.195493	0.00	0.00	0.00	0.00	1.00
True	2815.0	0.426288	0.494625	0.00	0.00	0.00	1.00	1.00
ID	2815.0	2.522202	1.123244	1.00	2.00	3.00	4.00	4.00

The average beats per minute were about 70 (±8.4), ranging from 50 to 87, and the average peripheral skin temperature was 33 (±0.6), ranging from 32 to 35.

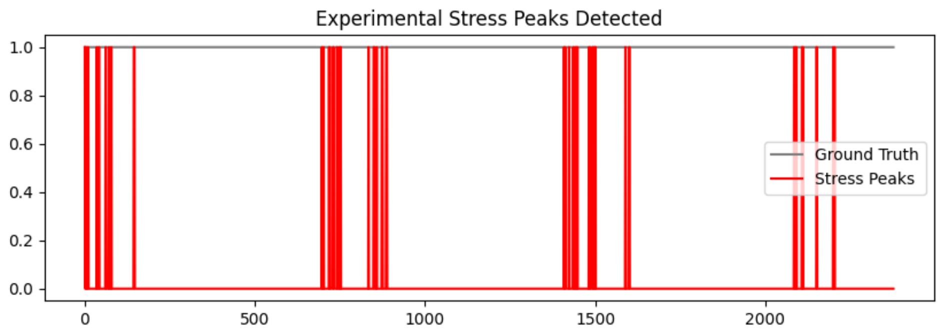


Figure 37 – Experimental Stress Peaks Detected

Figure 37 shows the stress peaks detected during the 5 minutes of stress task: the red line corresponds to the predicted classes and the gray line corresponds to the ground truth, selected as described above. It's important to underline that some stress peaks were detected during the math quiz only, while the relaxing task didn't detected any stress peaks, as the ground truth represents a point of referrment and the subjects isn't expected to be stressed for the whole stress task. For this reason, it's possible to confirm that the experiment produced satisfactory results, allowing to confute the thesis from this state on.

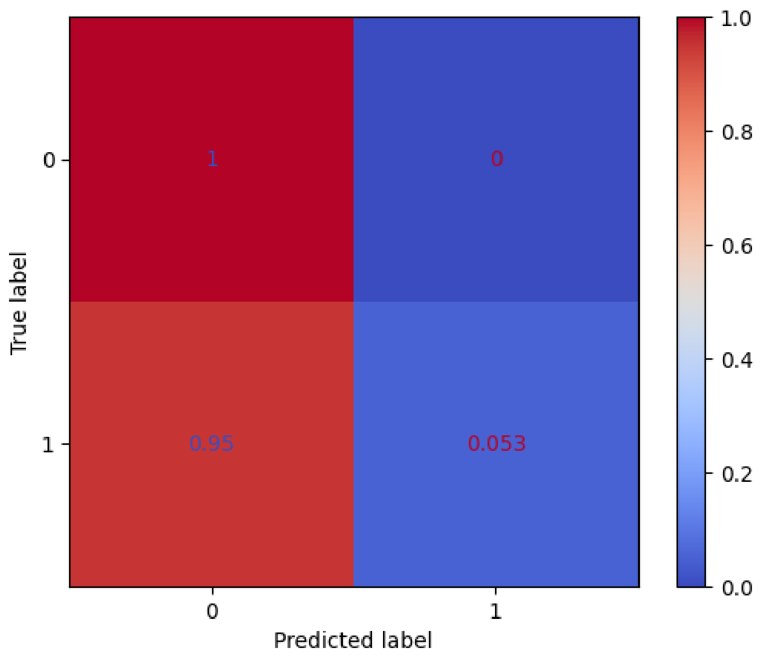


Figure 38 - Confusion Matrix of experiment results

The plot in Figure 38 shows the confusion matrix, which compares ground truth and predicted classes: the ground truth has been constructed by placing 1 for the entire first 5 minutes of the experiment and 0 for the entire relaxing task. The baseline was perfectly predicted for the whole experiment duration, while 5% of stress peaks were predicted from the 5 minutes stress task.

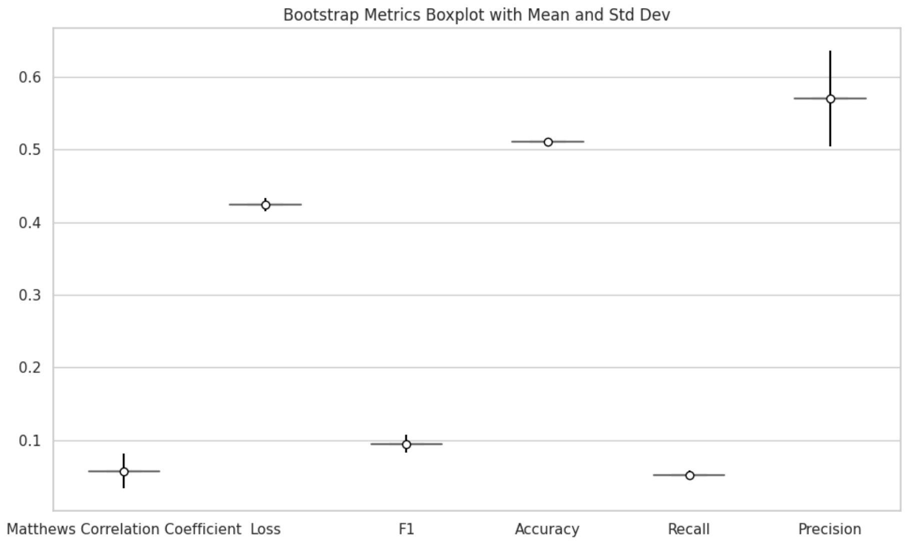


Figure 39 - Evaluation Metrics of experimental results (mean: white circle, standard deviation: vertical line)

The image in Figure 39 shows the chosen evaluation metrics, produced with the bootstrap technique: subsamples of size ten are evaluated iteratively with replacement; the graph shows the resulting mean with the white circle and the standard deviation with the vertical line. As we can see, the balanced accuracy is usually not so informative, especially in the medical field, as the recall is 5% and F1 score is 10%.

Chapter 5 – Discussion and Future Works

The machine learning models applied in this work produced significantly more significant results than the convolutional neural network model since extracting spectral features from a low number of temporal sequences presents itself as a better approach than deep learning.

Indeed, the experimental method used to induce stress or other emotional states attributable to the state of stress is a crucial moment to improve the results. In this case, the attempt is to replicate the experiment that allowed for the collection of the data on which the models were trained: the method of performing a large number of mathematically motivated responses under a set period is one of the most widely used methods of inducing anxiety and other comorbid disorders (Almazrouei et al., 2022).

However, the absence of researchers or subject matter experts during the experiment may cause a slight modification in the experimental data collection. This modified factor, together with the absence of a meditation phase between tasks, individual differences, timing and duration and other psychological and physiological factors, could cause the results obtained.

Despite the many measures of positive class quality, it is essential to note that each individual reacts differently to external stimuli. The fact that, during the stress experiment, only a few spikes were detected on the whole positive vector does not constitute a negative result since the individual may not feel stressed for the entire experiment duration but only during specific moments of difficulties. Instead, this experiment constitutes a starting point toward refining the quality and quantity of the sensors and, consequently, the learning algorithm.

Finally, the need to use a small number of biometric measurements due to the capabilities of the board is a significant limitation: the use of a more powerful device that can support the presence of more sensors, along with the extraction of more detailed features, is also a cue for improving the results obtained.

Future research can explore several vital directions to enhance the abovementioned aspects, utilizing further biometric measures in addition to heart rate and skin temperature data collected with wearable embedded devices.

Firstly, investigating the extraction of additional features from HR and ST data could improve detection accuracy. This exploration may involve using advanced signal processing techniques, such as time-frequency analysis, statistical analysis or non-linear dynamics analysis, to extract more informative features from the physiological signals and statistical parts. Furthermore, incorporating multimodal data, such as facial expressions, voice patterns or accelerometer, in conjunction with supplemental features from HR, such as IBI, BVP itself and HRV, could provide a more comprehensive understanding of social anxiety and enhance the performance of the algorithm to exclude confusing situations like training, walking and running.

To broaden the applicability of stress detection models, future research should consider conducting longitudinal studies that assess changes in biometric measure patterns over time in individuals with stress. Long-term monitoring using wearable devices would allow for investigating how such biometric profiles fluctuate across different social contexts and the impact of interventions or treatments on these physiological responses. Additionally, incorporating ecological validity by studying individuals in real-world social situations could provide valuable insights into the practical utility and generalizability of the random forest algorithm for stress detection.

Moreover, as has been seen in previous chapters about the limitations of applied models, the quality and diversity of the data is a crucial aspect of optimal model results. Collecting more biometric measures related to more individuals with different stress shades could be an additional and essential starting point for refining the algorithms tested in this work.

By pursuing these research directions, advancements can be made in stress detection through biometric data collected with wearable embedded devices. This intervention will improve detection accuracy, broader applicability, and greater interpretability of the algorithm, ultimately facilitating more effective assessment and intervention strategies for individuals with stress disorder.

Appendix

All the materials, such as codes, datasets, classifiers and pipelines used for this work, can be found in the GitHub repository[14], except for the original dataset, which requires approval before downloading.

The first part of the code has been developed in Python (Van Rossum & Fred, 2009), version 3.10.8, on both Visual Studio Code and Google Colab notebooks.

The second part of the code has been developed in C++ programming language (ISO/IEC, 2014) using the Arduino IDE, version 2.1.1.

[14] Master Thesis Repository | Github

Appendix A

```
import os
import pandas as pd
import numpy as np
from scipy import signal
from datetime import datetime, timedelta
def process_data(folder_path):
    individuals = ['S2', 'S3', 'S4', 'S5', 'S6', 'S7', 'S8', 'S9',
'S10', 'S11', 'S13', 'S14', 'S15', 'S16', 'S17']  # List of
individual folder names
    sampling_rate_original = 4  # Original sampling rate
    sampling_rate_target = 1  # Target sampling rate
    dfs = []

    for idx, individual in enumerate(individuals):
        individual_path = os.path.join(folder_path, individual)
        if not os.path.isdir(individual_path):
            continue  # Skip if the folder doesn't exist

        hr_path = os.path.join(individual_path, 'E4_Data', 'HR.csv')
        temp_path = os.path.join(individual_path, 'E4_Data',
'TEMP.csv')

        if not (os.path.isfile(hr_path) and
os.path.isfile(temp_path)):
            continue

        # HR signal
        hr_df = pd.read_csv(hr_path, names=['hr'], skiprows=2)
        hr_signal = hr_df['hr'].values
        hr_smoothed = pd.Series(hr_signal).rolling(3).mean().values

        # Temp signal
        temp_df = pd.read_csv(temp_path, names=['temp'], skiprows=2)
        temp_signal = temp_df['temp'].values
        temp_resampled = signal.resample(temp_signal,
int(len(temp_signal) * sampling_rate_target /
sampling_rate_original))
        temp_smoothed =
pd.Series(temp_resampled).rolling(3).mean().values

        # Time
        start_time = pd.to_datetime('00:00', format='%M:%S')
        time_hr = pd.Series([start_time + timedelta(seconds=(i /
sampling_rate_target)) for i in range(len(hr_smoothed))])
        time_temp = pd.Series([start_time + timedelta(seconds=(i /
sampling_rate_target)) for i in range(len(temp_smoothed))])

        # Create dataframes for 'bvp' and 'temp' signals with time
vectors and ID column
        hr_df_processed = pd.DataFrame({'ID': [idx] *
len(hr_smoothed), 'Time': time_hr, 'hr': hr_smoothed})
        temp_df_processed = pd.DataFrame({'ID': [idx] *
len(temp_smoothed), 'Time': time_temp, 'temp': temp_smoothed})

        # Merge the dataframes based on the 'ID' and 'Time' columns
        final_df = pd.merge(hr_df_processed, temp_df_processed,
on=['ID', 'Time'], how='outer')

        # Handle missing values, if any
        final_df = final_df.ffill().bfill()  # Forward-fill and
backward-fill missing values

        # Convert hours to minutes
        final_df['Time'] = final_df['Time'].dt.hour * 60 +
final_df['Time'].dt.minute

        # Fill with labels
        final_df['labels'] = 999 # Fill with placeholder for
transition
        quest = pd.read_csv(os.path.join(individual_path,
'quest.csv'), delimiter= ';')
        labels = quest.columns

        final_df.loc[(final_df['Time'] >= quest.iloc[0, 1]) &
(final_df['Time'] <= quest.iloc[1, 1]), 'labels'] = labels[1]
        final_df.loc[(final_df['Time'] >= quest.iloc[0, 2]) &
(final_df['Time'] <= quest.iloc[1, 2]), 'labels'] = labels[2]
        final_df.loc[(final_df['Time'] >= quest.iloc[0, 3]) &
(final_df['Time'] <= quest.iloc[1, 3]), 'labels'] = labels[3]
        final_df.loc[(final_df['Time'] >= quest.iloc[0, 4]) &
(final_df['Time'] <= quest.iloc[1, 4]), 'labels'] = labels[4]
        final_df.loc[(final_df['Time'] >= quest.iloc[0, 5]) &
(final_df['Time'] <= quest.iloc[1, 5]), 'labels'] = labels[5]

        start_time = pd.to_datetime('2023-07-13 00:00', format='%Y-
%m-%d %M:%S')
```

```
        final_df['Time'] = pd.Series([start_time + 
timedelta(seconds=(i / sampling_rate_target)) for i in 
range(len(final_df))])

        dfs.append(final_df)

    # Save the merged dataframe to a CSV file
    merged_df = pd.concat(dfs, ignore_index=True)
    output_file_path = os.path.join(folder_path, 'final_df.csv')
    merged_df.to_csv(output_file_path, index=False)
    print(f"Merged dataset saved as {output_file_path}")
process_data(folder_path)
```

Appendix B

```
import pandas as pd
import numpy as np
import plotly.express as px
import matplotlib.pyplot as plt
df = pd.read_csv(path)
protocol_states = ['Base', 'TSST']
df = df[df['labels'].isin(protocol_states)]
mapping_dict = {'Base': 0,
                'TSST': 1}
df['labels'] = df['labels'].map(mapping_dict)
df['labels'] = pd.Categorical(df['labels'])
df.head()
fig = px.line(df, x="Time", y="hr", color="labels", line_group="ID",
hover_name="ID", line_shape="spline", render_mode="svg",
template='simple_white')
fig.show()
fig = px.line(df, x="Time", y="temp", color="labels",
line_group="ID", hover_name="ID", line_shape="spline",
render_mode="svg", template='simple_white')
fig.show()
fig = px.histogram(df, x="hr", color="labels", marginal="rug",
hover_data=df.columns, template='simple_white')
fig.show()
fig = px.histogram(df, x="temp", color="labels", marginal="rug",
hover_data=df.columns, template='simple_white')
fig.show()
fig = px.scatter(df, x="hr", y="temp", color="labels",
marginal_y="violin",
           marginal_x="violin", trendline="ols",
template="simple_white")
fig.show()
fig = px.histogram(df, x="ID", color='labels', barmode='group',
template = 'simple_white')
fig.show()
```

Appendix C

```
import pandas as pd
import glob
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from everywhereml.data import Dataset
from everywhereml.preprocessing import Pipeline, MinMaxScaler,
Window, SpectralFeatures (Salerno Simone, 2023)
df = pd.read_csv(path)
protocol_states = ['Base', 'TSST']
df = df[df['labels'].isin(protocol_states)]
mapping_dict = {'Base': 0,
                'TSST': 1}
df['labels'] = df['labels'].map(mapping_dict)
ID = df['ID'].reset_index(drop=True) #store id column for later
usage
df.drop(['ID', 'Time'], inplace=True, axis = 1)
df.head()
df.info()
pipeline = Pipeline(name='Pipeline', steps=[
    Window(length=60, shift=1),
    SpectralFeatures(order=1),
    MinMaxScaler()
])
data = Dataset.from_pandas(df = df, name = 'sequence_dataset',
target_column = 'labels', target_name_column = 'labels')
data.apply(pipeline)
data.df.to_csv('C:/Users/aless/OneDrive - Università degli Studi di
Catania/tesi/dataset/lag_data.csv', index=False)
data = pd.read_csv('C:/Users/aless/OneDrive - Università degli Studi
di Catania/tesi/dataset/lag_data.csv')
ID = ID[30:-29] #removing initial and final values from the
sequence, because of the moving window
data['ID'] = ID.reset_index(drop=True) #insert id column to later
perform LOSO cross-validation
data.to_csv('C:/Users/aless/OneDrive - Università degli Studi di
Catania/tesi/dataset/lag_data.csv', index=False)
data.describe()
data.head()
corr = data.iloc[:,:-2].corr()
plt.figure(figsize=(12, 10))
sns.heatmap(corr, annot=True, cmap='coolwarm')
```

```
plt.show()
data.iloc[:,:-3].boxplot()
plt.xticks(rotation=45)
plt.show()
pipeline.to_arduino_file('C:/Users/aless/OneDrive - Università degli
Studi di Catania/tesi/codes/arduinocode/Final_code/Scaler.h',
instance_name='processor')
```

Appendix D

```
import os
import warnings
import pickle
warnings.filterwarnings('ignore')
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.experimental import enable_halving_search_cv
from sklearn.model_selection import cross_validate,
LeaveOneGroupOut, HalvingGridSearchCV, cross_val_predict
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from xgboost import XGBClassifier
from mlxtend.evaluate import mcnemar_table, mcnemar
from sklearn.metrics import ConfusionMatrixDisplay,
classification_report
import shap
from micromlgen import port
df = pd.read_csv(path)
df.drop('target_name', inplace=True, axis=1)
df.head()
X = df.drop(['target'], axis=1, inplace=False)
y = df.target
group = df.ID
false = (y == 0).sum()
true = (y == 1).sum()
print(false)
print(true)
parameters_logreg = {'penalty':('l1', 'l2', 'elasticnet'),
                     'solver':('lbfgs', 'liblinear', 'newton-cg',
'newton-cholesky', 'sag', 'saga'),
                     'C': np.logspace(-3, 0, 10)}
logreg = LogisticRegression(random_state=46, class_weight =
'balanced')
cv = LeaveOneGroupOut()
logreg_hyperparams = HalvingGridSearchCV(logreg, parameters_logreg,
cv = cv, scoring = 'f1_weighted', random_state=46, verbose=3)
logreg_hyperparams.fit(X, y, groups = group)
print(logreg_hyperparams.best_params_)
print(logreg_hyperparams.best_score_)
```

```
logreg = logreg_hyperparams.best_estimator_
parameters_rf = {'n_estimators': [5, 10, 20, 30],
               'max_depth': [5, 10, 20, 30],
               'min_samples_split': [2, 5, 10]}
rf = RandomForestClassifier(random_state=46, class_weight=
'balanced')
rf_hyperparams = HalvingGridSearchCV(rf, parameters_rf, cv = cv,
scoring = 'f1_weighted', random_state=46, verbose=3)
rf_hyperparams.fit(X, y, group)
print(rf_hyperparams.best_params_)
print(rf_hyperparams.best_score_)
rf = rf_hyperparams.best_estimator_
parameters_gnb = {'var_smoothing': np.logspace(-9, 9, 100)}
gnb = GaussianNB()
gnb_hyperparams = HalvingGridSearchCV(gnb, parameters_gnb, cv = cv,
scoring = 'f1_weighted', random_state=46, verbose=3)
gnb_hyperparams.fit(X, y, groups = group)
print(gnb_hyperparams.best_params_)
print(gnb_hyperparams.best_score_)
gnb = gnb_hyperparams.best_estimator_
parameters_xgb = {'booster': ['gbtree', 'gblinear', 'dart'],
                  'lambda': np.logspace(-3, 0, 3),
                  'alpha': np.logspace(-3, 0, 3),
                  'eta': np.logspace(-3, 0, 3)}
xgb = XGBClassifier(scale_pos_weight = false/true, objective =
'binary:logistic')
xgb_hyperparams = HalvingGridSearchCV(xgb, parameters_xgb, cv = cv,
scoring = 'f1_weighted', random_state=46, verbose=3)
xgb_hyperparams.fit(X, y, groups = group)
print(xgb_hyperparams.best_params_)
print(xgb_hyperparams.best_score_)
xgb = xgb_hyperparams.best_estimator_
def run_exps[15](X: pd.DataFrame , y: pd.DataFrame, ID: pd.DataFrame) -
> pd.DataFrame:
        '''
        Lightweight script to test many models and find winners
        :param X: features
        :param y: target
        :param ID: grouping variable for LOSO cross-validation
        :return: DataFrame of predictions
        '''

        dfs = []
```

[15] Quickly Compare Multiple Models | Cole on towardsdatascience.com

```
        models = [
                ('LogReg', logreg_hyperparams.best_estimator_),
                ('RF', rf_hyperparams.best_estimator_),
                ('GNB', gnb_hyperparams.best_estimator_),
                ('XGB', xgb_hyperparams.best_estimator_)
                ]
        results = []
        names = []
        scoring = ['balanced_accuracy', 'precision_weighted',
'recall_weighted', 'f1_weighted', 'roc_auc']
        target_names = ['no-anxiety', 'anxiety']
        for name, model in models:
                cv_results = cross_validate(model, X, y, cv=cv,
scoring=scoring, groups = ID)
                y_pred = cross_val_predict(model, X, y, cv=cv,
groups = ID)
                print(name)
                print(classification_report(y, y_pred,
target_names=target_names))
                results.append(cv_results)
                names.append(name)
                this_df = pd.DataFrame(cv_results)
                this_df['model'] = name
                dfs.append(this_df)
        final = pd.concat(dfs, ignore_index=True)
        return final
final = run_exps(X=X, y=y, ID=group)
bootstraps = []
for model in list(set(final.model.values)):
    model_df = final.loc[final.model == model]
    bootstrap = model_df.sample(n=10, replace=True, random_state=46)
    bootstraps.append(bootstrap)

bootstrap_df = pd.concat(bootstraps, ignore_index=True)
results_long =
pd.melt(bootstrap_df,id_vars=['model'],var_name='metrics',
value_name='values')
time_metrics = ['fit_time','score_time'] # fit time metrics
results_long_nofit =
results_long.loc[~results_long['metrics'].isin(time_metrics)] # get
df without fit data
results_long_nofit = results_long_nofit.sort_values(by='values')
results_long_fit =
results_long.loc[results_long['metrics'].isin(time_metrics)] # df
with fit data
```

```
results_long_fit = results_long_fit.sort_values(by='values')
plt.figure(figsize=(20, 12))
g = sns.boxplot(x="model", y="values", hue="metrics",
data=results_long_nofit)
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.title('Comparison of Model by Classification Metric')
plt.show()
plt.figure(figsize=(20, 12))
g = sns.boxplot(x="model", y="values", hue="metrics",
data=results_long_fit)
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.title('Comparison of Model by Fit and Score Time')
plt.show()
metrics = list(set(results_long_nofit.metrics.values))
bootstrap_df.groupby(['model'])[metrics].agg([np.std, np.mean])
time_metrics = list(set(results_long_fit.metrics.values))
bootstrap_df.groupby(['model'])[time_metrics].agg([np.std, np.mean])
fig, axes = plt.subplots(1, 4, figsize=(15, 5))
for i, clf in enumerate([logreg_hyperparams.best_estimator_,
rf_hyperparams.best_estimator_, gnb_hyperparams.best_estimator_,
xgb_hyperparams.best_estimator_]):
    prediction = cross_val_predict(clf, X, y, cv=cv, groups=group)
    disp = ConfusionMatrixDisplay.from_predictions(y, prediction,
display_labels=np.unique(y), normalize='true')
    disp.plot(ax=axes[i], colorbar=True, cmap='coolwarm')
    disp.ax_.set_title(f"Model {i+1}")

plt.tight_layout()
#plt.show()
logreg_pred = cross_val_predict(logreg, X, y, cv=cv, groups=group)
xgb_pred = cross_val_predict(xgb, X, y, cv=cv, groups=group)
tb = mcnemar_table(y_target=y,
                   y_model1=logreg_pred,
                   y_model2=xgb_pred )

print(tb)
chi2, p = mcnemar(ary=tb, corrected=True)
print('chi-squared:', chi2)
print('p-value:', p)
explainer = shap.Explainer(logreg, X)
shap_values = explainer(X)
shap.plots.beeswarm(shap_values)
if __name__ == '__main__':
    classmap = {
        0: 'Baseline',
        1: 'Stress'
    }
    c_code = port(logreg_hyperparams.best_estimator_,
classmap=classmap)

    with open('C:/Users/aless/OneDrive - Università degli Studi di
Catania/tesi/codes/arduinocode/Final_code/Classifier.h', 'w') as
file:
        file.write(c_code)
filename = 'C:/Users/aless/OneDrive - Università degli Studi di
Catania/tesi/codes/ML and DL - Python/ML/classifier.sav'
pickle.dump(logreg_hyperparams.best_estimator_, open(filename,
'wb'))
model = pickle.load(open(filename, 'rb'))
```

Appendix E

```
import os
import warnings
import pickle
warnings.filterwarnings('ignore')
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
path = 'C:/Users/aless/OneDrive - Università degli Studi di
Catania/tesi/dataset/scaled_dataset.csv'
from sklearn.model_selection import cross_validate,
StratifiedGroupKFold, cross_val_predict
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_curve, roc_auc_score
cv = StratifiedGroupKFold(n_splits=15, shuffle=True,
random_state=46)
df = pd.read_csv(path, index_col=0)
df.head()
X = df.drop(['ID', 'target'], axis=1, inplace=False)
y = df.target
groups = df.ID
# Create and train the logistic regression model
model = LogisticRegression(C = 1.0, penalty = 'l1', solver = 'saga',
random_state=46, class_weight = 'balanced')
model.fit(X,y)
# Predict probabilities of the positive class (class 1)
y_pred_prob = cross_val_predict(model, X, y, cv=cv, groups = groups,
method='predict_proba')[:, 1]
fpr, tpr, thresholds = roc_curve(y, y_pred_prob)
roc_auc = roc_auc_score(y, y_pred_prob)
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label=f'ROC Curve (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], 'k--')  # Diagonal line representing random
guessing
plt.xlabel('False Positive Rate (FPR)')
plt.ylabel('True Positive Rate (TPR)')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend()
plt.show()
top_left_idx = np.argmax(tpr - fpr)
optimal_threshold = thresholds[top_left_idx]
print(f"Optimal Threshold: {round(optimal_threshold, 2)}")
```

```
from sklearn.metrics import ConfusionMatrixDisplay
optimal_threshold = 0.59
y_prob = cross_val_predict(model, X, y, cv=cv, groups = groups,
method='predict_proba')
y_pred = (y_prob[:, 1] >= optimal_threshold).astype(int)
ConfusionMatrixDisplay.from_predictions(y, y_pred, normalize='true',
cmap='coolwarm')
from sklearn.metrics import classification_report
print(classification_report(y, y_pred))
```

Appendix F

```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from pyts.image import GramianAngularField
from sklearn.preprocessing import MinMaxScaler
np.random.seed(46)
path = '/content/final_df.csv'
df = pd.read_csv(path)
protocol_states = ['Base', 'TSST']
df = df[df['labels'].isin(protocol_states)]
mapping_dict = {'Base': 0,
                'TSST': 1}
df['labels'] = df['labels'].map(mapping_dict)
ID = df['ID'].reset_index(drop=True) #store id column for later
usage
df.drop(['Time'], inplace=True, axis = 1)
df.head()
scaler = MinMaxScaler()
columns_to_scale = ['hr', 'temp']
scaled_df = df.copy(deep=True)
scaled_df[['hr', 'temp']] = scaler.fit_transform(scaled_df[['hr',
'temp']])
scaled_df.head()
output_folder = '/content/gaf_imgs'
if not os.path.exists(output_folder):
    os.makedirs(output_folder)

def apply_gaf_transformation(data_array, filename):
    transformer = GramianAngularField(image_size=250,
sample_range=(-1, 1))
    img = transformer.fit_transform(data_array)

    black = np.zeros((1, 250, 250), dtype=np.float32)
    gaf = np.concatenate((img, black), axis=0)

    scaled = (gaf - gaf.min()) / (gaf.max() - gaf.min())
    scaled = np.transpose(scaled, (1, 2, 0))

    plt.imsave(filename, scaled)
for i in scaled_df['ID'].unique():
    for label in [0, 1]:
        df = scaled_df[(scaled_df['ID'] == i) & (scaled_df['labels']
== label)]
        array = df[['hr', 'temp']].values
        array = np.transpose(array)

        label_folder = os.path.join(output_folder, f'label_{label}')
        if not os.path.exists(label_folder):
            os.makedirs(label_folder)

        filename = os.path.join(label_folder, f'img_{i}.png')

        apply_gaf_transformation(array, filename)
import shutil
folder_to_download = "/content/gaf_imgs"
shutil.make_archive("/content/gaf_imgs", 'zip', folder_to_download)
from google.colab import files
files.download('/content/gaf_imgs.zip')
```

Appendix G

```
import os, sys, itertools, glob, torch, torchvision, cv2, natsort
import matplotlib.pyplot as plt
from PIL import Image
import numpy as np
from numpy import asarray
import pandas as pd
from matplotlib import pyplot as plt
from matplotlib import cm
from PIL import Image
from matplotlib.colors import LinearSegmentedColormap
import seaborn as sns
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import KFold
from torchvision import datasets, transforms, models
from torch.utils.data import DataLoader, random_split, Dataset,
TensorDataset, SubsetRandomSampler, ConcatDataset
import torch.optim as optim
import torchvision.transforms as transforms
torch.manual_seed(46)
from torch import nn
import torchvision
from torchview import draw_graph
import torchinfo
import torch.nn.functional as F
from captum.attr import visualization as viz
from captum.attr import IntegratedGradients, LayerGradCam,
LayerAttribution, NoiseTunnel, GuidedGradCam
print('Folders inside the main directory: ',
os.listdir("/content/drive/MyDrive/gaf_imgs"))
classes = ['label_0', 'label_1']
sum_images = []
for i in classes:
    img_base_path = "/content/drive/MyDrive/gaf_imgs/" + str(i)
    images = os.listdir(img_base_path)
    sum_images.append(len(images))
plt.bar(classes, sum_images)
plt.title('Distribution of classes')
plt.show()
images = []
paths = []

fig,ax = plt.subplots(1,2,figsize = (8,5))
```

```
ax = ax.ravel()

for idx, i in enumerate(classes):
    img_base_path = "/content/drive/MyDrive/gaf_imgs/" + str(i)
#link of the subfolders
    timages = os.listdir(img_base_path) #list of images inside each
subfolders
    images_path_to_display = os.path.join(img_base_path,
str(timages[0]))
    paths.append(images_path_to_display)
    img = Image.open(images_path_to_display)
    images.append(img)
    image = images[idx]
    ax[idx].axis('off')
    ax[idx].imshow(image,cmap = 'gray')
    ax[idx].set_title(str(i))

plt.suptitle('Image Example')
plt.show()
data = datasets.ImageFolder('/content/drive/MyDrive/gaf_imgs')
transform = transforms.Compose([transforms.RandomHorizontalFlip(1),
                                transforms.RandomVerticalFlip(1),
                                transforms.RandomRotation(30),
                                transforms.ToTensor(),
                                transforms.Normalize(mean=[0.5227,
0.4927, 0.4786], std=[0.3068, 0.3044, 0.3066])])
class loadData(Dataset):
    def __init__(self, dataset, transform = None):
        self.dataset = dataset
        self.transform = transform

    def __getitem__(self, index):
        if self.transform:
            x = self.transform(self.dataset[index][0])

        y = self.dataset[index][1]
        return x, y

    def __len__(self):
        return len(self.dataset)
dataset = loadData(data, transform)
print(dataset[0][0].shape)
plt.imshow(dataset[3][0].permute(1, 2, 0))
plt.axis('off')
plt.title('Augmented Image')
plt.show()
model = models.mobilenet_v2(pretrained=True)
model.classifier[1] = nn.Linear(1280, 2)
device = torch.device('cuda' if torch.cuda.is_available() else
'cpu')
model = model.to(device)
torchinfo.summary(model, (3, 250, 250), batch_dim = 0, col_names =
('input_size', 'output_size', 'num_params'), verbose = 0)
model_graph = draw_graph(model, input_size=(1,3,250,250), roll=True,
expand_nested=False, directory='/content/drive/MyDrive')
model_graph.visual_graph
optimizer = optim.SGD(model.parameters(), lr = 0.0001)
criterion = nn.CrossEntropyLoss()
num_epochs=30
batch_size=1
k=5
splits=KFold(n_splits=k,shuffle=True,random_state=42)
foldperf={}
def train_epoch(model,device,dataloader,loss_fn,optimizer):
    train_loss,train_correct=0.0,0
    model.train()
    for images, labels in dataloader:

        images,labels = images.to(device),labels.to(device)
        optimizer.zero_grad()
        output = model(images)
        loss = loss_fn(output,labels)
        loss.backward()
        optimizer.step()
        train_loss += loss.item() * images.size(0)
        scores, predictions = torch.max(output.data, 1)
        train_correct += (predictions == labels).sum().item()

    return train_loss,train_correct

def valid_epoch(model,device,dataloader,loss_fn):
    valid_loss, val_correct = 0.0, 0
    model.eval()
    with torch.no_grad():
      for images, labels in dataloader:

          images,labels = images.to(device),labels.to(device)
          output = model(images)
          loss=loss_fn(output,labels)
          valid_loss+=loss.item()*images.size(0)
          scores, predictions = torch.max(output.data,1)
          val_correct+=(predictions == labels).sum().item()

    return valid_loss,val_correct
history = {'train_loss': [], 'test_loss':
[],'train_acc':[],'test_acc':[]}

for fold, (train_idx,val_idx) in
enumerate(splits.split(np.arange(len(dataset)))):

    print('Fold {}'.format(fold + 1))

    train_sampler = SubsetRandomSampler(train_idx)
    test_sampler = SubsetRandomSampler(val_idx)
    train_loader = DataLoader(dataset, batch_size=batch_size,
sampler=train_sampler)
    test_loader = DataLoader(dataset, batch_size=batch_size,
sampler=test_sampler)

    model.to(device)

    for epoch in range(num_epochs):
        train_loss,
train_correct=train_epoch(model,device,train_loader,criterion,optimi
zer)
        test_loss,
test_correct=valid_epoch(model,device,test_loader,criterion)

        train_loss = train_loss / len(train_loader.sampler)
        train_acc = train_correct / len(train_loader.sampler) * 100
        test_loss = test_loss / len(test_loader.sampler)
        test_acc = test_correct / len(test_loader.sampler) * 100

        print("Epoch:{}/{} AVG Training Loss:{:.3f} AVG Test
Loss:{:.3f} AVG Training Acc {:.2f} % AVG Test Acc {:.2f}
%".format(epoch + 1,

                                                  num_epochs,

                                                  train_loss,

                                                  test_loss,

                                                  train_acc,
                                  test_acc))
        history['train_loss'].append(train_loss)
        history['test_loss'].append(test_loss)
        history['train_acc'].append(train_acc)
        history['test_acc'].append(test_acc)
avg_train_loss = np.mean(history['train_loss'])
avg_test_loss = np.mean(history['test_loss'])
avg_train_acc = np.mean(history['train_acc'])
avg_test_acc = np.mean(history['test_acc'])

print('Performance of {} fold cross validation'.format(k))
print("Average Training Loss: {:.4f} \t Average Test Loss: {:.4f} \t
Average Training Acc: {:.3f} \t Average Test Acc:
{:.3f}".format(avg_train_loss,avg_test_loss,avg_train_acc,avg_test_a
cc))
import seaborn as sns
confusion_matrix = np.zeros((2, 2))
with torch.no_grad():
    for i, (inputs, classes) in enumerate(train_loader):
        inputs = inputs.to(device)
        classes = classes.to(device)
        outputs = model(inputs)
        _, preds = torch.max(outputs, 1)
        for t, p in zip(classes.view(-1), preds.view(-1)):
                confusion_matrix[t.long(), p.long()] += 1

plt.figure()
df_cm = pd.DataFrame(confusion_matrix).astype(int)
heatmap = sns.heatmap(df_cm, annot=True, fmt="d")
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.show()
test_img = Image.open('/content/img_14.jpeg')
test_img_data = np.asarray(test_img)
plt.imshow(test_img_data)
plt.title('Stress - Individual 14')
plt.axis('off')
plt.show()
test_transform = transforms.Compose([
 transforms.ToTensor()])

transform_normalize = transforms.Normalize(
     mean=[0.5227, 0.4927, 0.4786], std=[0.3068, 0.3044, 0.3066])
```

```
transformed_img = test_transform(test_img)
norm_img = transform_normalize(transformed_img)
input_img = norm_img.unsqueeze(0)
input_img = input_img.to(device)
output = model(input_img)
output = F.softmax(output, dim=1)
prediction_score, pred_label_idx = torch.topk(output, 1)
pred_label_idx.squeeze_()
gg = GuidedGradCam(model, model.features[18]) #Guided GradCam on
last block
noise_tunnel = NoiseTunnel(gg) #noise tunnel on Guided GradCam
attributions_ig_nt = noise_tunnel.attribute(input_img,
nt_samples=20, nt_type='smoothgrad_sq', target=pred_label_idx,
stdevs = 0.2)
#visualize
_ =
viz.visualize_image_attr_multiple(np.transpose(attributions_ig_nt.sq
ueeze().cpu().detach().numpy(), (1,2,0)), #noise tunneled Guided
GradCam
                                      np.transpose(transformed_img.s
queeze().cpu().detach().numpy(), (1,2,0)), #unormalized img
                                      ["original_image",
"heat_map"],
                                      ["all", "positive"],
                                      cmap=cm.seismic,
                                      show_colorbar=True)
```

Appendix H

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.metrics import classification_report,
matthews_corrcoef, PrecisionRecallDisplay, RocCurveDisplay,
ConfusionMatrixDisplay, zero_one_loss, f1_score, precision_score,
balanced_accuracy_score, recall_score
import plotly.express as px
data = pd.read_csv("/content/round1.csv")
data.head()
data.info()
data.describe().T
fig = px.density_contour(data, x="BPM", y="ST", color="Predicted",
marginal_x = 'box', marginal_y = 'box', template ='simple_white')
fig.show()
data['Predicted'] = pd.Categorical(data['Predicted'])
fig = px.histogram(data, x='Predicted', color='Predicted',
barmode='group', template ='simple_white')
fig.show()
def signaltonoise_dB(a, axis=0, ddof=0):
    a = np.asanyarray(a)
    m = a.mean(axis)
    sd = a.std(axis=axis, ddof=ddof)
    return 20*np.log10(abs(np.where(sd == 0, 0, m/sd)))
grouped_data = data.groupby('ID')['BPM'].apply(signaltonoise_dB)

# Calculate the mean and standard deviation of the metrics.
mean_metric = grouped_data.mean()
std_metric = grouped_data.std()

# Convert the metrics to a list if needed.
metrics_list = grouped_data.tolist()

print("Average BPM SNR:", round(mean_metric,2), '(+-',
round(std_metric,2), ')')
grouped_data = data.groupby('ID')['ST'].apply(signaltonoise_dB)

# Calculate the mean and standard deviation of the metrics.
mean_metric = grouped_data.mean()
std_metric = grouped_data.std()
```

```
# Convert the metrics to a list if needed.
metrics_list = grouped_data.tolist()

print("Average BPM SNR:", round(mean_metric,2), '(+-',
round(std_metric,2), ')')
y_true = data['True']
y_pred = data['Predicted']
n_bootstraps = 10
matthews_scores = []
zero_one_loss_scores = []
f1_scores = []
precision_scores = []
recall_scores = []
accuracy_scores = []

for _ in range(n_bootstraps):
  indices = np.random.choice(len(y_true), size=len(y_true),
replace=True)
  bootstrap1 = y_true[indices]
  bootstrap2 = y_pred[indices]

  mcc = matthews_corrcoef(bootstrap1, bootstrap2)
  matthews_scores.append(mcc)

  zero_one = zero_one_loss(bootstrap1, bootstrap2, normalize=True)
  zero_one_loss_scores.append(zero_one)

  f1 = f1_score(bootstrap1, bootstrap2)
  f1_scores.append(f1)

  accuracy = balanced_accuracy_score(bootstrap1, bootstrap2)
  accuracy_scores.append(accuracy)

  precision = precision_score(bootstrap1, bootstrap2)
  precision_scores.append(precision)

  recall = recall_score(bootstrap1, bootstrap2)
  recall_scores.append(recall)

bootstrap_mean_mcc = np.mean(matthews_scores)
bootstrap_std_mcc = np.std(matthews_scores)

bootstrap_mean_loss = np.mean(zero_one_loss_scores)
bootstrap_std_loss = np.std(zero_one_loss_scores)
```

```
bootstrap_mean_f1 = np.mean(f1_scores)
bootstrap_std_f1 = np.std(f1_scores)

bootstrap_mean_accuracy = np.mean(accuracy_scores)
bootstrap_std_accuracy = np.std(accuracy_scores)

bootstrap_mean_recall = np.mean(recall_scores)
bootstrap_std_recall = np.std(recall_scores)

bootstrap_mean_precision = np.mean(precision_scores)
bootstrap_std_precision = np.std(precision_scores)

print(f"Bootstrap Mean Matthews Correlation Coefficient:
{bootstrap_mean_mcc:.2f}")
print(f"Bootstrap Std Matthews Correlation Coefficient:
{bootstrap_std_mcc:.2f}")

print(f"Bootstrap Mean Loss: {bootstrap_mean_loss:.2f}")
print(f"Bootstrap Std Loss: {bootstrap_std_loss:.2f}")

print(f"Bootstrap Mean F1: {bootstrap_mean_f1:.2f}")
print(f"Bootstrap Std F1: {bootstrap_std_f1:.2f}")

print(f"Bootstrap Mean Accuracy: {bootstrap_mean_accuracy:.2f}")
print(f"Bootstrap Std Accuracy: {bootstrap_std_accuracy:.2f}")

print(f"Bootstrap Mean Recall: {bootstrap_mean_recall:.2f}")
print(f"Bootstrap Std Recall: {bootstrap_std_recall:.2f}")

print(f"Bootstrap Mean Precision: {bootstrap_mean_precision:.2f}")
print(f"Bootstrap Std Precision: {bootstrap_std_precision:.2f}")
print(classification_report(y_true, y_pred))
ConfusionMatrixDisplay.from_predictions(y_true, y_pred,
cmap='coolwarm', normalize='true')
plt.show()
RocCurveDisplay.from_predictions(y_true, y_pred,
plot_chance_level=True)
plt.show()
PrecisionRecallDisplay.from_predictions(y_true, y_pred,
drop_intermediate=True, plot_chance_level=True)
plt.show()
```

Appendix I

```
#include "arduino_secrets.h"
// SparkFun MAX3010x Pulse and Proximity Sensor Library - Version:
Latest
#include <MAX30105.h>[16] // --> Sparkfun library for MAX30101 sensor
#include <heartRate.h>[17] // --> Optical Heart Rate Detection (PBA
Algorithm)
#include <DFRobot_MLX90614.h> // --> Melexis temperature sensor
library
#include "Scaler.h"[18] // --> Feature extraction library
#include "Classifier.h"[19] // --> Classification library containing
Logistic Regression algorithm

/*
  Sketch generated by the Arduino IoT Cloud Thing "Untitled"
  https://create.arduino.cc/cloud/things/3f6c33c9-c58d-43fe-816c-
a85d5290d3a4

  Arduino IoT Cloud Variables description

  The following variables are automatically generated and updated
when changes are made to the Thing

  String response;
  int beatAvg;
  int st;

  Variables which are marked as READ/WRITE in the Cloud Thing will
also have functions
  which are called when their values are changed from the Dashboard.
  These functions are generated with the Thing and added at the end
of this sketch.
*/

#include "thingProperties.h"
//#include <Wire.h> // --> I2C connection
MAX30105 particleSensor; // --> defining particle sensor object used
for BPM
```

[16] Sparkfun library Github repository
[17] Peripheral Beat Amplitude Algorithm
[18] Everywhereml | Eloquent Arduino
[19] Micromlgen | Eloquent Arduino

```
Eloquent::ML::Port::LogisticRegression clf; // --> declare
classifier library
DFRobot_MLX90614_I2C tempSensor; // --> defining temperature sensor
object used for st

const byte RATE_SIZE = 3; //Averaging
byte rates[RATE_SIZE]; //Array of heart rates
byte rateSpot = 0;
long lastBeat = 0; //Time at which the last beat occurred
int beatsPerMinute; // --> raw BPM
//int beatAvg; // --> averaged BPM
//st
byte st_values[RATE_SIZE];
byte st_spot = 0;
int degree; // --> raw peripheral skin temperature in °C
//int st; // averaged peripheral skin temperature in °C

//const char* response;

void setup() {
  // Initialize serial and wait for port to open:
  //Serial.begin(9600);
  //Wire.begin(4, 5, 100000);
  particleSensor.begin(Wire, I2C_SPEED_STANDARD, 0x57);
  tempSensor.begin();
  byte ledBrightness = 0x1F;
  byte sampleAverage = 1;
  byte ledMode = 3;
  byte sampleRate = 50;
  int pulseWidth = 411;
  int adcRange = 4096;
  particleSensor.setup(ledBrightness, sampleAverage, ledMode,
sampleRate, pulseWidth, adcRange);
  particleSensor.setPulseAmplitudeRed(0); //0 mA
  particleSensor.setPulseAmplitudeIR(0);  // 0 mA
  particleSensor.setPulseAmplitudeGreen(0x19); // 1 mA

  // Defined in thingProperties.h
  initProperties();

  // Connect to Arduino IoT Cloud
  ArduinoCloud.begin(ArduinoIoTPreferredConnection);

  /*
     The following function allows you to obtain more information
     related to the state of network and IoT Cloud connection and
errors
     the higher number the more granular information you’ll get.
     The default is 0 (only errors).
     Maximum is 4
 */
  setDebugMessageLevel(2);
  ArduinoCloud.printDebugInfo();
}

void hrCollection() {
/**
* The particle sensor capture the green led emission from the inner
wrist. When a blood pulse happen, the photodiode capture a pulse of
light which corresponds to a heart beat.
* In order to be able to capture peaks throw the wrist, the green
led has been accurately set up in the setup() section, following the
instructions published by the authors
* on page 22 of the paper[20].
* The peaks, corresponding to the BPM, are computed throw a PBA
algorithm, then, they are averaged with a moving window of size 3.
*/
  long irValue = particleSensor.getGreen();

  if (checkForBeat(irValue) == true)
  {
    //We sensed a beat!

    long delta = millis() - lastBeat;
    lastBeat = millis();

    beatsPerMinute = 60 / (delta / 1000.0);

    // removing signal error
    if (beatsPerMinute < 255 && beatsPerMinute > 20)
    {
      rates[rateSpot++] = (byte)beatsPerMinute; //Store this reading
in the array
      rateSpot %= RATE_SIZE; //Wrap variable

      //computing average of 3 samples
      beatAvg = 0;
      for (byte j = 0 ; j < RATE_SIZE ; j++)
```

[20] Wrist positioning | MAX30101

```
        beatAvg += rates[j];
      beatAvg /= RATE_SIZE;
    }
  }

}

void stCollection() {
/**
* Peripheral skin Temperature has been measured in degree Celsius
thanks to an InfraRed thermophile that capture the emissivity of the
human skin.
*/

  degree = tempSensor.getObjectTempCelsius();

  if (degree > 10 && degree < 50) // removing signal error
  {
    //computing average of 3 samples
    st_values[st_spot++] = (byte)degree;
    st_spot %= RATE_SIZE;
    st = 0;
    for (byte z = 0 ; z < RATE_SIZE ; z++)
      st += st_values[z];
    st /= RATE_SIZE;

  }

}

void loop() {
  ArduinoCloud.update();
  hrCollection();
  stCollection();

  int features[] = {beatAvg, st};

  if (!processor.transform(features))
  return;

  response = clf.predictLabel(processor.X);


  //Serial.print(beatAvg); Serial.print(","); Serial.print(st);
Serial.print(","); Serial.println(response);}
```

Bibliography

Agrafioti, F., Hatzinakos, D., & Anderson, A. K. (2012). ECG pattern analysis for emotion detection. *IEEE Transactions on Affective Computing*, *3*(1), 102–115. https://doi.org/10.1109/T-AFFC.2011.28

Ahmad, Z., & Khan, N. (2022). *A Survey on Physiological Signal Based Emotion Recognition*. http://arxiv.org/abs/2205.10466

Ahmed, A., Aziz, S., Alzubaidi, M., Schneider, J., Irshaidat, S., Serhan, H. A., Abd-alrazaq, A. A., Solaiman, B., & Househ, M. (2023). Wearable devices for Anxiety and Depression: A scoping review. *Computer Methods and Programs in Biomedicine Update*, 100095. https://doi.org/10.1016/j.cmpbup.2023.100095

Akitan. (2010). *Esraj-Brook-Birds*. Akitan-Music.

Ali, S. H., Al-Sultan, H. A., & Al Rubaie, M. T. (2022). Fifth Industrial Revolution. *International Journal of Business, Management and Economics*, *3*(3), 196–212. https://doi.org/10.47747/ijbme.v3i3.694

Allen, J. (2007). Photoplethysmography and its application in clinical physiological measurement. *Physiological Measurement*, *28*(3), R1–R39. https://doi.org/10.1088/0967-3334/28/3/R01

Almazrouei, M. A., Morgan, R. M., & Dror, I. E. (2022). A method to induce stress in human subjects in online research environments. *Behavior Research Methods*. https://doi.org/10.3758/s13428-022-01915-3

American Psychiatric Association. (2013). *Diagnostic and statistical manual of mental disorders : DSM-5.* (Vol. 5). American Psychiatric Association.

Apollo Neuro. (2023). https://apolloneuro.com/pages/science

Artz, D. (2020). *Information Technology Support for the Arterial Thoracic Outlet Syndrome*.

Batista, G. E. A. P. A., Wang, X., & Keogh, E. J. (n.d.). *A Complexity-Invariant Distance Measure for Time Series*.

Bengio, Y. (1994). *Convolutional Networks for Images, Speech, and Time-Series*. https://www.researchgate.net/publication/216792820

Breiman, L. (2001). *Random Forests*.

Chen, T., & Guestrin, C. (2016). *XGBoost: A Scalable Tree Boosting System*. https://doi.org/10.1145/2939672.2939785

Chollet, F. (2016). *Xception: Deep Learning with Depthwise Separable Convolutions*. http://arxiv.org/abs/1610.02357

Cox, D. R. (1958). The Regression Analysis of Binary Sequences. *Journal of the Royal Statistical Society. Series B (Methodological)*, *20*(2), 215–242. http://www.jstor.org/stable/2983890

Defazio, A., Bach, F., & Lacoste-Julien, S. (2014). *SAGA: A Fast Incremental Gradient Method With Support for Non-Strongly Convex Composite Objectives*. http://arxiv.org/abs/1407.0202

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., & Lin, C.-J. (2008). LIBLINEAR: A Library for Large Linear Classification. In *Journal of Machine Learning Research* (Vol. 9). http://www.csie.ntu.edu.tw/

Garbarino, M., Lai, M., Bender, D., Picard, R. W., & Tognetti, S. (2014). Empatica E3 - A wearable wireless multi-sensor device for real-time computerized biofeedback and data acquisition. *2014 EAI 4th International Conference on Wireless Mobile Communication and Healthcare (Mobihealth)*, 39–42. https://doi.org/10.1109/MOBIHEALTH.2014.7015904

Ghosh, S., Kim, S. K., Ijaz, M. F., Singh, P. K., & Mahmud, M. (2022). Classification of Mental Stress from Wearable Physiological Sensors Using Image-Encoding-Based Deep Neural Network. *Biosensors*, *12*(12). https://doi.org/10.3390/bios12121153

He, K., Zhang, X., Ren, S., & Sun, J. (2015). *Deep Residual Learning for Image Recognition*. http://arxiv.org/abs/1512.03385

ISO/IEC. (2014). *ISO International Standard ISO/IEC 14882:2014(E) – Programming Language C++*.

Kanjo, E., Younis, E. M. G., & Ang, C. S. (2019). Deep learning analysis of mobile physiological, environmental and location sensor data for emotion detection. *Information Fusion*, *49*, 46–56. https://doi.org/10.1016/j.inffus.2018.09.001

Kanjo, E., Younis, E. M. G., & Sherkat, N. (2018). Towards unravelling the relationship between on-body, environmental and emotion data using sensor information fusion approach. *Information Fusion*, *40*, 18–31. https://doi.org/10.1016/j.inffus.2017.05.005

Kim, \ K H, Bang, I. S. W., & Kim, S. R. (2004). Emotion recognition system using short-term monitoring of physiological signals. In *Med. Biol. Eng. Comput* (Vol. 42, pp. 419–427).

Liebowitz, M. R. (1987). *Social Phobia* (pp. 141–173). https://doi.org/10.1159/000414022

McGinnis, R. S., McGinnis, E. W., Hruschak, J., Lopez-Duran, N. L., Fitzgerald, K., Rosenblum, K. L., & Muzik, M. (2018). Rapid Anxiety and Depression Diagnosis in Young Children Enabled by Wearable Sensors and Machine Learning. *40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 3983–3986. https://doi.org/10.1109/EMBC.2018.8513327

McNemar, Q. (1947). Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, *12*(2), 153–157. https://doi.org/10.1007/BF02295996

Muse - the brain sensing headband. (2021). choosemuse.com

Noble, S. M., Mende, M., Grewal, D., & Parasuraman, A. (2022). The Fifth Industrial Revolution: How Harmonious Human–Machine Collaboration is Triggering a Retail and Service [R]evolution. *Journal of Retailing*, *98*(2), 199–208. https://doi.org/10.1016/j.jretai.2022.04.003

Oura Ring. (n.d.). Retrieved February 4, 2023, from https://ouraring.com/it

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel V. and Thirion, B., Grisel, O., Blondel, M., Prettenhofer P. and Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Quadrini, M., Daberdaku, S., Blanda, A., Capuccio, A., Bellanova, L., & Gerard, G. (2022). Stress Detection from Wearable Sensor Data Using Gramian

Angular Fields and CNN. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *13601 LNAI*, 173–183. https://doi.org/10.1007/978-3-031-18840-4_13

Ramzan, N., Palke, S., Cuntz, T., Gibson, R., & Amira, A. (2016). Emotion recognition by physiological signals. *Human Vision and Electronic Imaging 2016, HVEI 2016*, 231–236. https://doi.org/10.2352/ISSN.2470-1173.2016.16HVEI-129

Richard B, by H., Lu, P., Nocedal, J., & Zhu, C. (1994). *A LIMITED MEMORY ALGORITHM FOR BOUND CONSTRAINED OPTIMIZATION*.

Rinella, S., Massimino, S., Fallica, P. G., Giacobbe, A., Donato, N., Coco, M., Neri, G., Parenti, R., Perciavalle, V., & Conoci, S. (2022). Emotion Recognition: Photoplethysmography and Electrocardiography in Comparison. *Biosensors*, *12*(10), 811. https://doi.org/10.3390/bios12100811

Salerno Simone. (n.d.). *Eloquent Arduino* . Retrieved March 1, 2023, from eloquentarduino.com

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018). *MobileNetV2: Inverted Residuals and Linear Bottlenecks*. http://arxiv.org/abs/1801.04381

Sarfraz, Z., Sarfraz, A., Iftikar, H. M., & Akhund, R. (2021). Is covid-19 pushing us to the fifth industrial revolution (Society 5.0)? *Pakistan Journal of Medical Sciences*, *37*(2), 1–4. https://doi.org/10.12669/pjms.37.2.3387

Schmidt, M., Le Roux, N., & Bach, F. (2017). Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, *162*(1–2), 83–112. https://doi.org/10.1007/s10107-016-1030-6

Schmidt, P., Reiss, A., Duerichen, R., Marberger, C., & Van Laerhoven, K. (2018). Introducing WESAD, a Multimodal Dataset for Wearable Stress and Affect Detection. *Proceedings of the 20th ACM International Conference on Multimodal Interaction*, 400–408. https://doi.org/10.1145/3242969.3242985

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2016). *Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization*. https://doi.org/10.1007/s11263-019-01228-7

Sensate - Meet sensate, your key to calm. (2021). getsensate.com

Simone, A. (2023). Least Squared Wasserstein GAN with Gradient Penalty for MRI Tumour Brain generation. In *Proceedings of Knowledge Discovery Course-UNICT*. https://www.kaggle.com/datasets/abhranta/brain-tumor-detection-mri

Singh, P. (2019). Supervised Machine Learning. In *Learn PySpark: Build Python-based Machine Learning and Deep Learning Models* (pp. 117–159). Apress. https://doi.org/10.1007/978-1-4842-4961-1_6

Smilkov, D., Thorat, N., Kim, B., Viégas, F., & Wattenberg, M. (2017). *SmoothGrad: removing noise by adding noise*. http://arxiv.org/abs/1706.03825

Tableau. (n.d.).

Touchpoints Solutions. (2023). https://thetouchpointsolution.com/pages/research

Van Rossum, G. & D., & Fred, L. (2009). *Python 3 Reference Manual*. CreateSpace.

Wang, Z., & Oates, T. (2015). *Imaging Time-Series to Improve Classification and Imputation*. http://arxiv.org/abs/1506.00327

Yuan, T., Deng, W., Tang, J., Tang, Y., & Chen, B. (2019). *Signal-to-Noise Ratio: A Robust Distance Metric for Deep Metric Learning*. http://arxiv.org/abs/1904.02616

Zhang, A., Lipton, Z. C., Li, M. U., & Smola, A. J. (2023). *Dive into Deep Learning*.