UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

GRADO EN INGENIERÍA DE TECNOLOGÍAS Y SERVICIOS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

DEVELOPMENT OF AN OCCUPATIONAL STRESS PREDICTION SYSTEM EXPLOITING COMPUTER INTERACTION DATA THROUGH MACHINE LEARNING

SOFÍA PÉREZ CHAO

JULIO 2024

TRABAJO DE FIN DE GRADO

Título:	DESARROLLO DE UN SISTEMA DE PREDICCIÓN DEL ESTRÉS LABORAL QUE EXPLOTA DATOS DE LA INTERACCIÓN CON ORDENADORES MEDIANTE APRENDIZAJE AUTOMÁTICO
Título (inglés):	DEVELOPMENT OF AN OCCUPATIONAL STRESS PREDICTION SYSTEM EXPLOITING COMPUTER INTERACTION DATA THROUGH MACHINE LEARNING
Autor:	SOFÍA PÉREZ CHAO
Tutor:	SERGIO MUÑOZ LÓPEZ
Ponente:	PONENTE
Departamento:	Departamento de Ingeniería de Sistemas Telemáticos

MIEMBROS DEL TRIBUNAL CALIFICADOR

Presidente:	——
Vocal:	——
Secretario:	——
Suplente:	——

FECHA DE LECTURA:

CALIFICACIÓN:

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE
INGENIEROS DE TELECOMUNICACIÓN

Departamento de Ingeniería de Sistemas Telemáticos
Grupo de Sistemas Inteligentes

TRABAJO DE FIN DE GRADO

DEVELOPMENT OF AN OCCUPATIONAL STRESS PREDICTION SYSTEM EXPLOITING COMPUTER INTERACTION DATA THROUGH MACHINE LEARNING

JULIO 2024

Resumen

En la actualidad, el estrés laboral es un fenómeno ampliamente experimentado que se manifiesta en nuestras vidas de manera consciente o inconsciente. En los últimos años está aumentando su intensidad e incidencia debido en gran parte a las nuevas formas de organización del trabajo. La exposición prolongada al estrés mental contribuye a una mala experiencia laboral e incluso a problemas de salud graves.

La intensidad y prevalencia cambiantes del estrés laboral subrayan la necesidad de soluciones más accesibles y menos invasivas para su detección. Los avances en este campo han conducido al desarrollo de nuevos métodos basados en la explotación de datos biométricos. Sin embargo, estos métodos suelen plantear retos similares a los enfoques actuales, como la necesidad de sensores intrusivos. Gracias al aprendizaje automático, es posible realizar predicciones no invasivas del estrés, utilizando técnicas como el reconocimiento de la postura corporal, las expresiones faciales, el análisis de pulsaciones de teclas, el movimiento del ratón e incluso el uso de teléfonos móviles

En este proyecto se ha dado prioridad a la aplicación de técnicas de detección que no requieran hardware específico adicional. Un enfoque ampliamente adoptado en este ámbito consiste en analizar el comportamiento individual. Este proyecto pretende explorar cómo pueden explotarse las interacciones de las personas con sus ordenadores cotidianos para detectar sus niveles de estrés. Para ello, este proyecto evaluará el rendimiento predictivo del estrés de varios patrones asociados a las interacciones con el ordenador, como la dinámica del ratón y de las pulsaciones de teclas, la mirada, etcétera. Por lo tanto, el objetivo principal será desarrollar un sistema de detección de estrés basado en la interacción con el ordenador utilizando técnicas de aprendizaje automático.

Para lograr este propósito, se han identificado varias fases a seguir durante el desarrollo del proyecto: un estudio exhaustivo de diferentes técnicas de detección, una revisión y limpieza previa de los conjuntos de datos para eliminar elementos que puedan interferir en el proceso, y la aplicación de diferentes enfoques de aprendizaje automático para extraer conclusiones a partir de los datos recogidos. El desarrollo de este modelo se llevará a cabo utilizando el lenguaje de programación Python.

Palabras clave: Estrés laboral, Detección no invasiva, Datos Biométricos, Inteligencia artificial, Aprendizaje automático, Interacciones con el ordenador, Python.

Abstract

Nowadays, occupational stress is a widely experienced phenomenon that manifests itself in our lives consciously or unconsciously. It has always been present at work, however, in recent years it is increasing in intensity and incidence due in large part to new forms of work organization. Prolonged exposure to mental stress contributes to a poor work experience and even to serious health problems.

The changing intensity and prevalence of occupational stress underscore the need for more accessible and less invasive solutions for its detection. Significant advancements in the field have led to the development of new methods based on the exploitation of biometric data, whose reliability has been demonstrated. However, these methods often pose similar challenges to current approaches, such as the need for intrusive sensors, despite the increasing demand and interest in their accessibility and applicability in everyday life and work environments. Thanks to machine learning, non-invasive stress predictions are feasible, using techniques such as body posture recognition, facial expressions, keystroke analysis, mouse movement, and even the use of mobile phones.

In this project, priority has been given to the application of detection techniques that do not require additional specific hardware. A widely adopted approach in this domain involves analysing individual behaviour. Specifically, this project seeks to explore how people's interactions with their everyday computers can be exploited to detect their stress levels. To this aim, this project will evaluate the stress predictive performance of various patterns associated with computer interactions, including mouse and keystroke dynamics, eye gaze, and more. Therefore, the main objective will be to develop a stress detection system based on computer interaction using machine learning techniques.

To achieve this purpose, several phases have been identified to be followed during the project's development: an exhaustive study of different detection techniques, a previous review and cleaning of datasets to eliminate elements that may interfere in the process, and the application of different machine learning approaches in order to draw conclusions from the collected data. The development of this model will be carried out using the Python programming language.

Keywords: Occupational stress, Non-invasive detection, Biometric data, Artificial intelligence, Machine learning, Computer interactions, Python.

Agradecimientos

Al llegar al final de mis años como estudiante, no puedo evitar reflexionar sobre todas las experiencias vividas durante esta etapa que ha sido tan enriquecedora y desafiante para mí.

Quiero expresar mi profundo agradecimiento a todas las personas que me han apoyado y ayudado a lo largo de mis estudios en la Escuela Técnica Superior de Ingenieros de Telecomunicación (ETSIT).

En primer lugar, agradezco profundamente a mi tutor, Sergio Muñoz López, por su dedicación y orientación durante la realización de mi Trabajo de Fin de Grado. Gracias a él, pude explorar un tema interesante y actual, superando los desafíos que surgieron en el proceso y expandiendo mis horizontes académicos.

También quiero reconocer el apoyo incondicional de mis compañeros y amigos de la universidad. Durante estos años de carrera, su solidaridad y colaboración fueron fundamentales para superar los obstáculos y celebrar los éxitos juntos.

Aunque sin duda a quien más debo dar las gracias es a mi familia. Gracias por creer en mí incluso cuando dudaba de mí misma, por haber estado siempre dispuestos a ayudarme en todo lo posible y por animarme a no rendirme nunca.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

This chapter establishes the framework of the project, presenting the main theme and providing an overview of the objectives to be achieved. The relevance and importance of the project in its corresponding field is explored, providing the necessary context to understand the purpose and scope of the study.

1.1 Context

Stress is a universal phenomenon in modern life, affecting individuals of all ages and backgrounds. Its relevance lies in its significant impact on physical and mental health, as well as on people's performance and general well-being. In this context, occupational stress emerges as an increasingly prominent concern due to its prevalence and negative consequences in the work environment and beyond [8, 9].

In the midst of the fast pace of modern life, stress is a omnipresent force affecting people from all walks of life, profoundly impacting their physical and mental well-being, as well as their overall performance and quality of life [8]. Within this dynamic context, work-related stress has emerged as a significant concern due to its widespread prevalence and detrimental effects both inside and outside the workplace [9].

Work-related stress manifests itself when individuals face work demands and pressures that exceed their capabilities, thus challenging their coping mechanisms [10, 11]. This phenomenon occurs in a variety of work settings and tends to be exacerbated when employees perceive inadequate support from supervisors and colleagues, or have limited control over work processes [12].

The importance of addressing occupational stress lies in its adverse consequences for workers' health and well-being, as well as for organizational productivity and effectiveness. Prolonged exposure to occupational stress has been linked to a range of health issues, from psychological disorders such as anxiety and depression to physical ailments like cardiovascular diseases and musculoskeletal disorders. Furthermore, occupational stress can negatively impact morale, job satisfaction, and work quality, ultimately affecting job performance and employee retention [13, 14].

Current stress management techniques in organizations have focused on evaluating the state of employees to implement approaches to help them face the negative effects of stress. However, the use of human experts in these interventions can be costly and impractical for many organizations, which has led to the exploration of automated approaches [15]. In addition, advantages and limitations have been identified in traditional methods for measuring stress in the work context, where questionnaires offer inexpensive data collection but can be influenced by subjective perceptions, while physiological sensors provide accurate measurements but can generate discomfort and affect results due to the monitoring process itself.

Technological advances and medical research are driving a more precise approach to addressing stress, fatigue and emotions in the human body through physiological sensors such as cortisol, skin conductance, skin temperature, heart rate and heart rate variability. These indicators reveal mental states associated with stress and emotions, being useful for understanding the impact of stress on the human body. However, despite the accuracy of these approaches, they present challenges in work settings, such as individual discomfort and interference with established work routines. In addition, privacy and acceptability are major concerns when using these approaches, which may require significant changes in work practices or interfere with the individual's autonomy for effective implementation [16].

Smartphones are also playing an important role in stress management by taking advantage of the growing variety of built-in sensors, such as camera and microphones, to measure stress and provide feedback to users. In addition to wearable devices and smartphones, approaches based on computer vision, speech and other linguistic features, as well as the use of the computer mouse and keyboard are also being used to assess stress. These approaches offer non-intrusive alternatives for measuring and managing stress in different environments and situations, reflecting the diversity of options available to address this challenge in the workplace [15].

This work focuses on studying different classification techniques and comparing them to develop a model capable of accurately detecting a person's stress based on keyboard and mouse interaction patterns. For the analysis of these typing patterns and mouse movements, we will use the dataset generated for the SWELL project [7].

1.2 Project goals

The main objective of this project is to develop a stress detection system based on user interactions with the computer, taking advantage of advanced machine learning techniques to analyze keyboard patterns and mouse movements. This system will seek to provide an accurate and non-intrusive method to identify stress, which has important applications in work and personal contexts. To achieve this goal, the project is structured into several key tasks:

1. Review of the State of the Art to Analyze Current Solutions

The state-of-the-art review involves investigating the most recent and relevant stress detection techniques, analyzing the approaches and classification models used in previous studies to identify the most effective strategies, and evaluating the advantages, limitations and challenges of current solutions, focusing on non-intrusive methods.

2. Search and Acquisition of Datasets

The search and acquisition of datasets involves identifying and obtaining datasets that are representative for model training, selecting those related to the analysis of typing patterns and mouse movements.

3. Preprocess the Datasets

Prior to model training, the datasets will be preprocessed to remove any elements that may adversely affect the subsequent analysis. In addition, the labels of the datasets will be checked and balanced if they are found to be out of calibration.

4. Design of Machine Learning Models

The design involves applying different machine learning techniques to each dataset, allowing for the development of models that can effectively detect stress based on the available features.

5. Evaluation and Experimentation

This involves validating the models through experimentation and evaluation, facilitating the selection and validation of a stress detection model.

1.3 Structure of this document

In this section we provide a brief overview of the chapters included in this document. The structure is as follows:

***Chapter 1* Introduction**: Establishes the context of the project, presenting the main topic and providing an overview of the objectives to be achieved. Explores the context in which the project is being developed, highlighting its relevance and importance in the corresponding field. This chapter serves as a starting point for understanding the purpose and scope of the project, laying the basis for the development of the following chapters.

***Chapter 2* State of Art:** Conducts a comprehensive analysis of related work in the field of study to contextualize the project within the current research landscape. Reviews and compares different approaches, methodologies, and results obtained in similar or related projects. Provides a detailed description of the technologies and environments used to carry out the project, addressing aspects such as development tools, software libraries, and specific platforms used during the process. Offers an essential frame of reference for understanding the state of the art in the area of study and comprehending the technological decisions made in the development of the project.

***Chapter 3* Datasets and Model Development:** Describes the datasets used for training and evaluating the stress detection models, providing detailed information on their characteristics, sources, and relevance to the project. Explores the preprocessing steps required to prepare the data for model training, including cleaning, feature extraction, and normalization. Finally, presents the design of the machine learning models, explaining the architecture, algorithms, and classification techniques employed.

***Chapter 4* Evaluation:** Establishes the evaluation context of the classification models developed, presenting the main metrics and providing a detailed description of the results obtained. This chapter provides an understanding of the evaluative methods employed, laying the basis for the detailed discussion of the results of the experiments. It focuses on interpreting and assessing the performance of the models under several key metrics, showing a comprehensive evaluation.

Chapter 5 **Conclusion:** Establishes the closure of the project analysis, summarizing the key findings and highlighting the main contributions of the study. Explores the implications of the results obtained, emphasizing their importance in the context of developing stress detection tools and improving personal and occupational well-being. This chapter serves as a synthesis of the learnings and discoveries made, outlining the basis for the development of future research and applications in this field. In addition, directions for future research are proposed, highlighting opportunities to expand and deepen current findings.

CHAPTER 2

State of Art

This chapter conducts a comprehensive analysis of related work in the field of study, situating the project in the current research landscape. Different approaches, methodologies and results of similar or related projects are reviewed and compared. In addition, a detailed description of the technologies and environments used to carry out the project is provided, addressing aspects such as development tools, software libraries and specific platforms.

2.1 Related work

In this section, a review of previous work related to the detection of occupational stress through computer interaction is carried out. Due to the increasing importance of health and well-being in the work environment, recent research has explored different techniques and methodologies to identify and prevent stress in the workplace. Despite advances in wearable sensor technology and machine learning (ML) methods over the past two decades, as well as the considerable number of stress detection studies,there are still several challenges related to stress detection in real-world contexts.

Androutsou et al. [17], presented a discrete and multimodal system for automatic monitoring and detection of stress in office workers using a computer. In this study, physiological measurements recorded by the device are combined with behavioral parameters derived from the use of the computer keyboard and mouse to detect users' stress levels. Combined analysis of physiological and behavioral parameters through feature-level fusion resulted in models that demonstrated greater efficiency compared to those using single modalities. Their findings highlight the feasibility of using low-cost IoT devices and modules already integrated into the work routine to monitor the status and stress levels of workers.

This approach complements the findings of Koldjik et al. [7], who, in their research on automatic classifiers, investigated working conditions and mental states associated with stress using multimodal sensor data. Although their results suggest that computer interaction features do not provide the most meaningful information about stress, their study provides valuable insight into the possibilities and limitations of these technologies in the work environment. Furthermore, the dataset collected by Koldjik et al. (SWELL-KW) remains a reference resource for future research in this area.

Building on this foundation, Naegelin et al. [18] conducted a study simulating an office environment, where they assessed variability in keyboard, mouse and heart activity to identify stress automatically. Their results showed that keyboard and mouse dynamics are better indicators of work stress than heart rate variability.

Similarly, Pepa et al. [19] conducted a study where they collected experimental keyboard and mouse data from 62 volunteers in natural environments using a web application specifically designed to induce stress. This study applied multiple instance learning (MIL) to random forest (RF) classification to successfully distinguish three classes of stress levels from keyboard (76% accuracy) and mouse (63% accuracy) data. These results provide a new perspective on stress detection in real-world contexts and complement the findings of other studies [17, 7, 18], that have explored different approaches to automatic stress detection in the workplace.

In another approach, Sol et al. [20] investigated the possibility of identifying stress in users by analysing common computer mouse operations and constructing a model of the hand and muscles using a mass-spring-damper system.participants were asked to perform three different activities: pointing and clicking, steering, and dragging and dropping. Each of the tasks was performed in both a relaxed and stressed state. They concluded that stress assessment can be more accurate using metrics derived from mouse activities than by analysing electrocardiogram signals, especially when considering the within-subject model. They also showed that a small amount of data can be used to assess stress state with 70% accuracy.

Complementarily, Muaremi et al. [21] created a solution to assess people's stress level by using a smartphone and a wearable chest belt. During the workday, they collected audio, communication, and physical activity data. They also recorded heart rate variability (HRV) during the night. During the day, participants completed self-assessment surveys to describe their feelings. Thirty-five employees were studied over 4 months. They achieved 55% accuracy using only the smartphone features and 59% accuracy using only the HRV features. Combining these features, they achieved an accuracy of 61% for classifying stress into three different levels.

Finally, Hernandez et al. [22] examined the possibility of detecting stress in users using the keyboard and mouse. They focused on tactile pressure with a capacitive mouse and a pressure-sensitive keyboard. Twenty-four participants completed three different tasks: text input, expressive typing, and mouse clicking. The results indicated that stress had a significant impact on keyboard touch pressure in more than 83% of the users. Although the intensity of pressure did not change much with mouse clicking, the contact area increased during stress.

The reviewed studies demonstrate the diverse approaches and technologies used in detecting occupational stress through computer interaction. While some focus on physiological measurements and others on behavioral parameters, the combination of multiple modalities seems to offer the most promising results. However, we found challenges such as:

- Data Collection in Real-World Contexts: Many studies simulate office environments, but lack the complexity of real workplaces.
- Invasiveness: Wearable sensors can cause discomfort and interfere with established work routines.
- Generalization and Customization: Stress models often lack adaptability across individuals due to personal differences.

Following this analysis, our work aims to address these challenges by developing a non-invasive system to predict work-related stress using machine learning (ML). We will focus on analyzing keyboard and mouse dynamics based on the SWELL dataset to design and validate a robust stress detection model. By refining existing classification techniques and integrating them into an accessible application, our project will contribute to the advancement of non-intrusive stress detection in real-world work environments.

2.2 Enabling Technologies

In this section, the technologies and tools used in the development of this project will be presented. With a central focus on the application of Machine Learning techniques, the different libraries used to carry out the analysis and evaluation of data will be detailed. In addition, the tools and platforms used during the work will be included.

2.2.1 Machine Learning

Before detailing the specific technologies employed in the project, it is essential to understand the concept of Machine Learning(ML): is a field of computer science that studies algorithms and techniques for automating solutions to complex problems that are hard to program using conventional programming methods [23].

Machine Learning covers a wide range of applications, from speech recognition and natural language processing to medical diagnostics and autonomous vehicle driving. By exploiting large datasets and advanced algorithms, Machine Learning enables computer systems to learn patterns and make autonomous decisions, making it a critical component in the era of artificial intelligence and automation.

As shown in Figure 2.1, Machine Learning models are typically categorized into four essential categories [24].

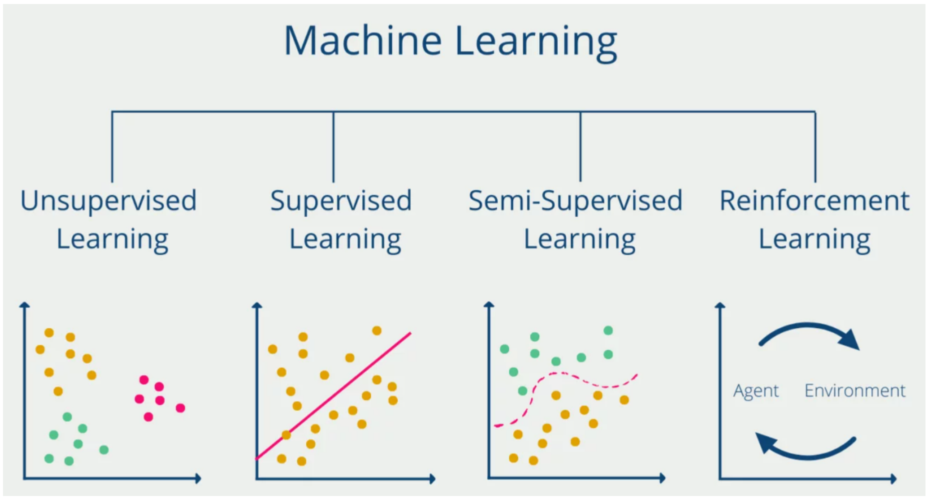

Figure 2.1: Types of Machine Learning [1].

- **Supervised Learning:** Supervised learning is an essential task in machine learning that involves learning a function that maps inputs to outputs based on pairs of sample data. This process uses labeled training data and a collection of training examples to derive the function. It is used when specific goals are identified to be achieved from a given set of inputs, following a task-oriented approach. The most common supervised tasks are "classification", which organizes data into categories, and "regression", which fits the data. An example would be predicting the class label or sentiment of a text, such as a tweet or product review.

- **Unsupervised Learning:** Unsupervised learning deals with the analysis of unlabeled datasets without human intervention, following a data-driven approach. This method is widely used to extract generative features, identify trends and meaningful structures, and to perform clustering and exploratory analysis. The most common tasks in unsupervised learning include clustering, density estimation, feature learning, dimensionality reduction, association rule search and anomaly detection.

- **Semi-supervised Learning:** Semi-supervised learning can be considered a combination of supervised and unsupervised methods, since it uses both labeled and unlabeled data. Therefore, it falls somewhere in between "unsupervised" and "supervised" learning. The main goal of a semi-supervised learning model is to achieve better prediction accuracy compared to using only labeled data. Some application areas of semi-supervised learning include machine translation, fraud detection, data labeling and text classification.

- **Reinforcement Learning:** Reinforcement learning is a machine learning approach that allows software agents and machines to automatically identify optimal behavior in a specific environment to improve their efficiency, using an environment-driven method. This type of learning is based on a system of rewards and penalties, the goal of which is to take advantage of information obtained from interactions with the environment to make decisions that increase rewards or reduce risks. Although it is a very effective tool for training artificial intelligence models, it is not ideal for solving simple or basic problems.

Given the nature of the datasets used in this project, supervised learning will be our focus.

2.2.1.1 Traditional Classifiers

Following our discussion on supervised learning, it's pertinent to delve deeper into traditional classifiers, which are pivotal in many supervised learning scenarios. These classifiers are designed to predict categorical outcomes by learning from labeled input data.

Traditional classifiers [25] in machine learning are algorithms that have a long history of use in classification tasks [23], where the goal is to assign labels to instances based on input features. These classifiers typically rely on simpler, well-understood mathematical models, and have been extensively studied and applied in a variety of fields.

Types of Traditional Classifiers [26]:

- **Logistic Regression:** A statistical model that predicts a binary outcome, determining whether an event happens or does not. It uses a logistic function to model the probability of a certain class or event existing. Logistic Regression is widely used in situations where the dependent variable is binary.
- **Decision Trees:** A supervised learning algorithm ideal for classification problems, capable of sorting classes at a precise level. Decision Trees split the data into subsets based on the value of input features, creating a tree-like structure. Each node represents a decision rule, and each leaf node represents an outcome. This method is particularly useful for its interpretability and simplicity.
- **Support Vector Machines (SVM):** An algorithm that trains and classifies data by degrees of polarity, extending beyond simple X/Y predictions. SVM finds the optimal hyperplane that maximizes the margin between different classes. It is highly effective in high-dimensional spaces.
- **Naive Bayes:** A probabilistic classifier that calculates the likelihood of a data point belonging to a particular category based on Bayes' theorem. It assumes independence between predictors, which simplifies the computation.
- **k-Nearest Neighbors (k-NN):** A pattern recognition algorithm that identifies the k closest data points in training datasets to classify new examples. The class of a new data point is determined by the majority class among its k nearest neighbors. k-NN is easy to implement and understand, and it performs well on smaller datasets where the relationships between data points are more straightforward.

2.2.2 Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics [27]. Python's built-in high-level data structures, along with its dynamic typing and linking, make it especially attractive for rapid application development. It is also useful as a scripting or integration language for connecting existing components. Python's simple, easy-to-learn syntax promotes readability, which reduces program maintenance costs.

Python supports modules and packages, thus encouraging modularity and code reuse. The Python interpreter and its extensive standard library are freely available in source or binary format for all major platforms, and can be freely distributed.

In summary, Python is a modern programming language that supports object-oriented, functional and imperative programming styles [28]. It is primarily a scripting language, but it can also be compiled into computer-readable binaries. The advantage of all this is that it allows programs to be written with fewer lines of code compared to equivalent programs in C/C++ or Java.

Having established a basic understanding of the concept, we will move on to explore the practical tools that make its implementation possible. Next, we will focus on some of the most important Python libraries that were used in this project, highlighting their fundamental role in the development and implementation of machine learning solutions.

- **Pandas:** Pandas is a Python library that provides fast, flexible and expressive data structures, making it easy to work with "relational" or "tagged" data. It is designed to be the fundamental high-level building block for performing practical, real-world data analysis in Python. In addition, it has the goal of becoming the most powerful and flexible open source data analysis and manipulation tool available in any language. It is currently making progress towards this goal [29].

  Built on top of NumPy, Pandas is intended to integrate seamlessly into a scientific computing environment, along with many other third-party libraries. Its two main data structures, Series (one-dimensional) and DataFrame (two-dimensional), cover most of the typical use cases in finance, statistics, social sciences and various areas of engineering.

  Pandas is an exceptional tool for data scientists, providing robust functionality that covers every stage of the data analysis process. This process generally involves several steps: data acquisition and cleaning, data analysis and modeling, and ultimately, structuring the analysis results in a manner that is suitable for visualization or tabular presentation.

- **Numpy:** NumPy, short for Numerical Python, is an open source Python library designed for scientific computing. It allows you to manipulate matrices and arrays without hassle and offers a comprehensive collection of mathematical functions, including tools for linear algebra, Fourier transform, and random number generation. NumPy effectively serves as a substitute for some features found in MATLAB and Mathematica, facilitating rapid interactive prototyping [30].

  In essence, NumPy is indispensable for numerical processing in Python because of its clean and efficient code, its high performance, and its ability to handle large datasets in a scalable way. Much of its code is written in C, which gives it a significant speed advantage over pure Python code. In addition, its C API allows it to extend its functionality.

- **Scikit-learn:** Scikit-learn is a free and open source software designed to solve supervised and unsupervised machine learning problems. It is based on Python and is based on high-quality libraries such as NumPy and SciPy [31]. The main advantage of scikit-learn's popularity lies in the ability to quickly implement most of the commonly used machine learning algorithms in a plug-and-play format, once the core pipeline is understood. Also, popular classification algorithms, such as logistic regression and support vector machines, are implemented in Cython, which provides C-like performance, making it possible to use scikit-learn in a highly efficient manner.

  Scikit-learn offers solutions for several types of machine learning problems [32]:

  - **Classification:** Determine the category to which an object belongs. Available algorithms include decision trees (ID3, C4.5, etc.), kNN, SVM, Random forest, Perceptron, and more.
  - **Clustering:** Automatic grouping of similar objects into sets. Algorithms such as k-Means and affinity propagation are available.
  - **Regression:** Prediction of a continuous-valued attribute associated with an object. Includes algorithms such as linear regression and logistic regression.
  - **Dimensionality reduction:** Reduce the number of random variables to consider, using algorithms such as SVD and PCA.

- **Tsfresh:** Tsfresh is a Python library designed for systematic feature engineering from time series and other sequential data. These data share the characteristic of being ordered by an independent variable, with time being the most common independent variable. To keep things simple, we refer to all the different types of sequential data as time series [33]. In Figure 2.2 the three steps of the TSFresh algorithm are illustrated: feature extraction (1), p-value calculation (2), and multiple testing processing (3) [2]. In particular, TSFresh provides 63 time series characterization methods, which compute a total of 794 time series features.

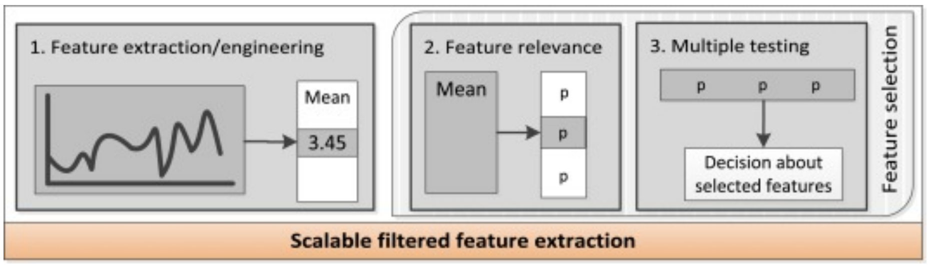
1. Feature extraction/engineering
Mean
3.45
2. Feature relevance
Mean
p
p
p
3. Multiple testing
p p p
Decision about selected features
Feature selection
Scalable filtered feature extraction

Figure 2.2: The three steps of the tsfresh algorithm [2].

The first stage of the TSFresh process is feature extraction, where various statistical and mathematical techniques are applied to extract relevant information from the time series, such as mean, variance, median, among others.

In the second stage, the significance of the extracted features is calculated by hypothesis testing, assigning p-values to each feature to determine its significance. This stage is crucial to filter out irrelevant features and reduce the dimensionality of the dataset.

The third and final stage is multiple testing processing, where p-values are adjusted using correction techniques, such as the Benjamini-Hochberg method, to control the false discovery rate. This step ensures that the selected features are statistically significant and relevant to the analysis.

The `feature_extraction` submodule contains both the collection of feature calculators and the logic to apply them efficiently to the time series data. The main public function of this submodule is `extract_features` [2].

CHAPTER 3

Datasets and Model Development

This chapter describes the datasets used for training and evaluation of the stress detection models, providing detailed information on their characteristics, sources, and relevance to the project. The preprocessing steps required to prepare the data, including cleaning, feature extraction, and normalization, are explored. Finally, the design of the machine learning models is presented, explaining their architecture, algorithms and classification techniques used.

3.1 Dataset

For the project, the most suitable dataset and thus chosen to carry out our work is the **SWELL-KW**. This dataset was specifically designed to investigate the effects of stressors on workers, collecting a wide range of both physiological and behavioral data. It provides detailed information on how time pressure and interruptions affect the subjective experience of workload, mental effort and emotions.

The SWELL-KW dataset was collected by Koldijk et al. with the participation of 25 individuals [7, 34]. Participants performed typical tasks such as report writing and presentations, under three working conditions:

- **Neutral (N):** Participants could work on the tasks as long as they needed to.
- **Time Pressure (T):** Time to complete the tasks was reduced to 2/3 of the time taken in the neutral condition, with a maximum limit of 30 minutes.
- **Interruptions (I):** Eight emails were sent during the task, some relevant and some irrelevant, to simulate interruptions.

Each experimental block began with an 8-minute relaxation phase. Then, participants received instructions on the tasks to be performed, writing reports and making presentations. Subsequently, they completed questionnaires about their subjective experience of stress, emotions and workload.

The following data were collected [34]:

- **Computer Interactions:** Detailed timestamped information about each computer event was recorded by the computer logging software. Examples of these events include mouse clicks, mouse scrolls, and application changes. Additionally, several relevant characteristics of mouse, keyboard, and application interactions were computed per minute.
- **Facial Expressions:** Extracted from the video for each timeframe using FaceReader software, the dataset includes characteristics such as quality, estimates on the orientation of the head, and some global features like looking direction and the activation of several facial action units.
- **Body Posture:** Data were extracted from 3D Kinect recordings using the Kinect SDK. By fitting the Kinect skeletal model, coordinates of all body joints per frame were obtained. Additionally, standard deviations were calculated for each minute to identify features indicating the amount of movement and changes in joint angles.

- **Physiology (ECG and Skin Conductance):** Raw and preprocessed ECG data are provided. Heart rate and heart rate variability were calculated. Additionally, raw skin conductance data were provided, and the average skin conductance level was determined by averaging the raw signal per minute.

In total, the dataset contains 149 features and 2688 instances, annotated according to the conditions under which they were collected. These features are summarized in Table 3.1.

Modality	**Feature type**
Computer interactions (18)	Mouse (7) Keyboard (9) Applications (2)
Facial expressions (40)	Head orientation (3) Facial movements (10) Action Units (19) Emotion (8)
Body postures (88)	Distance (1) Joint angles (10) Bone orientations (3x11) (as well as stdv of the above for amount of movement (44))
Physiology (3)	Heart rate (variability) (2) Skin conductance (1)

Table 3.1: Overview of SWELL-KW Dataset Features [7].

To conclude, in our project we have focused on computer interactions only. These data, captured through a logging tool, provide valuable information on keyboard and mouse usage patterns, allowing us to analyze correlations between typing and cursor movement behaviors, and stress levels experienced by participants.

3.2 Data Preprocessing

This section details the preprocessing process applied to the SWELL-KW ensemble data to ensure that they are ready for model training.

This dataset provides a wide range of features, but for this project, we only selected features related to computer interactions, 16 in total. These features were chosen based on their relevance to user behavior patterns that could indicate stress levels. The selected characteristics are listed in the Table 3.2.

Name	**Type**	**Description**
Mouse Activity	Mouse	Number of all MouseEvents
Left Clicks	Mouse	Number of left clicks
Right Clicks	Mouse	Number of right clicks
Double Clicks	Mouse	Number of double clicks
Mouse Wheel	Mouse	Number of mouse wheel actions
Dragged	Mouse	Number of dragged events
Mouse Distance	Mouse	Distance of mouse movements
KeyStrokes	Keyboard	Number of all KeyEvents
Shortcut Keys	Keyboard	Number of shortcut keys (Ctrl+c/x/v/z/s/a; Shift+Tab)
Direction Keys	Keyboard	Number of direction keys (arrow left/right/up/down)
Characters	Keyboard	Number of characters (a-z)
Characters Ratio	Keyboard	#characters divided by #keyStrokes
Error Keys	Keyboard	Number of error keys (Backspace, Delete, Ctrl+Z)
Error Key Ratio	Keyboard	#errorKeys divided by (#characters + #spaces)
Special Keys	Keyboard	Number of special keys
Spaces	Keyboard	Number of spaces

Table 3.2: Mouse and Keyboard Features.

Once the features were selected, duplicate values were eliminated. Rows containing null values (NaN) were also removed to maintain data integrity. This cleaning process ensured that the data fed into the model was of high quality and representative of actual user interactions.

A remarkable aspect of the data preprocessing was the transformation of the target variable, the stress level presented as a decimal number. To adapt this to a binary classification, we used the median value of the stress levels, which is 2.8. Other studies have applied similar binarization methods using the median [35]. However, in this case, we did not binarize per participant since we only had three different stress values, one for each condition. Instead, we chose to calculate the median of all the samples.

Therefore, we converted the stress measurements into a binary format where values greater than or equal to 2.8 were assigned a '1', indicating a stressed user, and values below 2.8 were assigned a '0', indicating a non-stressed user.

This median-based binarization makes the dataset balanced, providing a clear threshold that reflects the central tendency of stress levels within the dataset, ensuring that the classification process remains robust and meaningful.

The Figure 3.1 shows the number of stressed and unstressed users after binarization, clearly illustrating how this methodology has succeeded in balancing the classes within the dataset. In addition, Figure 3.2 classifies the stress values into intervals of 0-1, 1-2, so on up to 10, providing a detailed view of the original distribution of stress levels.

Comparing both figures, it can be seen that the original distribution of stress levels is varied, but the median-based binarization simplifies this variability into a clear and balanced binary classification. This balance is critical for the model to effectively learn to distinguish between the two conditions without preference for the more frequent class.

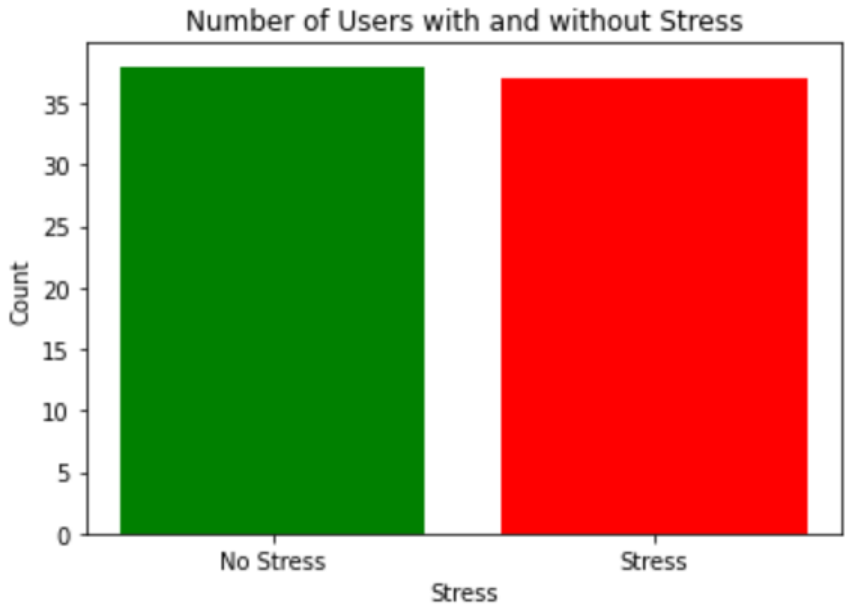

Figure 3.1: Users with and without Stress.

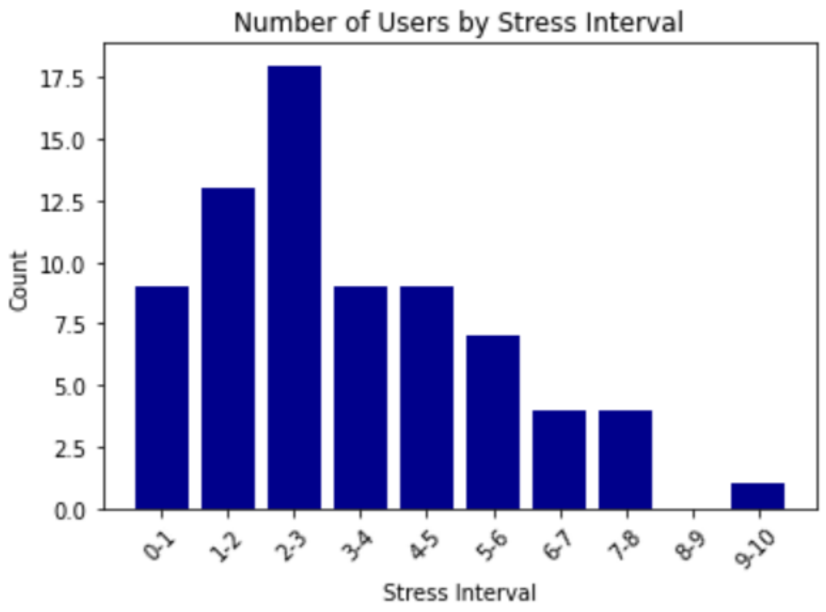

Figure 3.2: Number of Users by Stress Interval.

Continuing with the preprocessing, we performed two different approaches to prepare the SWELL ensemble data. In the first case, we worked directly with the original features extracted from the typing patterns and mouse movements without applying the Tsfresh library. This approach kept the original 2657 samples, allowing us to evaluate the impact of feature reduction on the model's performance.

In the second case, we used the Tsfresh library to summarize the 2657 samples of the dataset into 3 samples per user, corresponding to each of the experimental conditions: neutral, time pressure, and interruptions. This approach allowed us to consolidate the information into 75 rows, resulting in 25 users with one sample for each condition. As a result, 12432 features were generated, including statistics such as median, mean, variance, length, entropy, root mean square, standard deviation, minimum, and maximum for each feature of the original dataset. The use of Tsfresh has been essential to increase the number of statistical features derived from the original data, providing a more complete and detailed description of user behavior patterns. This process not only improves the model's ability to capture the complex patterns and nuances of user behavior, but also increases the quality of the data used to train the classification system. By extracting and summarizing a wide range of features, Tsfresh allowed us to more accurately and reliably identify user stress levels, significantly improving the accuracy and reliability of the stress detection model.

3.3 Architecture

We will explore the structure of a system specifically designed to analyze how user interactions with the computer can be indicative of stress levels. This model leverages data from keyboard and mouse usage, integrating advanced processing techniques to predict affective states. We will detail the fundamental components of this model, highlighting their functionality and how they collectively contribute to accurate stress detection.

The architecture of our model, Figure 3.3, unfolds through a series of clearly defined stages that turn everyday interactions with the computer into stress indicators. It begins with the collection of data on specific keyboard and mouse interactions. From this data, features categorized into four groups are extracted:

- **Keyboard interactions:** This group includes all features derived from the user's activity with the keyboard. The characteristics include total number of keystrokes (*SnKeyStrokes*), typed characters (*SnChars*), use of special keys (*SnSpecialKeys*), use of directional keys (*SnDirectionKeys*), erroneous or correction keystrokes (*SnErrorKeys*) and the use of keyboard shortcuts (*SnShortcutKeys*). These data provide valuable information about the user's typing frequency, intensity, and accuracy, which may be related to stress levels.

- **Mouse events:** This group covers user interactions with the mouse, such as total mouse activity (*SnMouseAct*), left clicks (*SnLeftClicked*), right clicks (*SnRightClicked*), double clicks (*SnDoubleClicked*), mouse wheel use (*SnWheel*), dragging (*SnDragged*) and mouse distance traveled (*SnMouseDistance*). These characteristics may indicate nervous or relaxed user behaviors during work sessions.

- **Keystrokes Combinations:** This group focuses on specific key combinations that the user employs. For example, the use of space keys (*SnSpaces*), application changes (*SnAppChange*), and tab focus changes (*SnTabfocusChange*). Analysis of these combinations may reveal patterns in keyboard usage that could correlate with stress.

- **Ratio Calculations:** This group includes the analysis of specific ratios derived from the above interactions. For example, the proportion of typed characters (*CharactersRatio*) and the proportion of erroneous keys (*ErrorKeyRatio*). These ratios help to better understand how different forms of interaction relate to each other and can provide more subtle indicators of user behavior under stress.

Depending on the approach, these features are either compiled directly into vectors or further processed using the Tsfresh library. As illustrated in Figure 3.3, these compiled vectors are used to feed a classification algorithm. The classifier evaluates the feature vectors and generates a prediction, determining whether the user is under stress based on the patterns identified during training.

This dual approach allows us to compare the effectiveness of raw versus extracted features in accurately predicting stress levels.

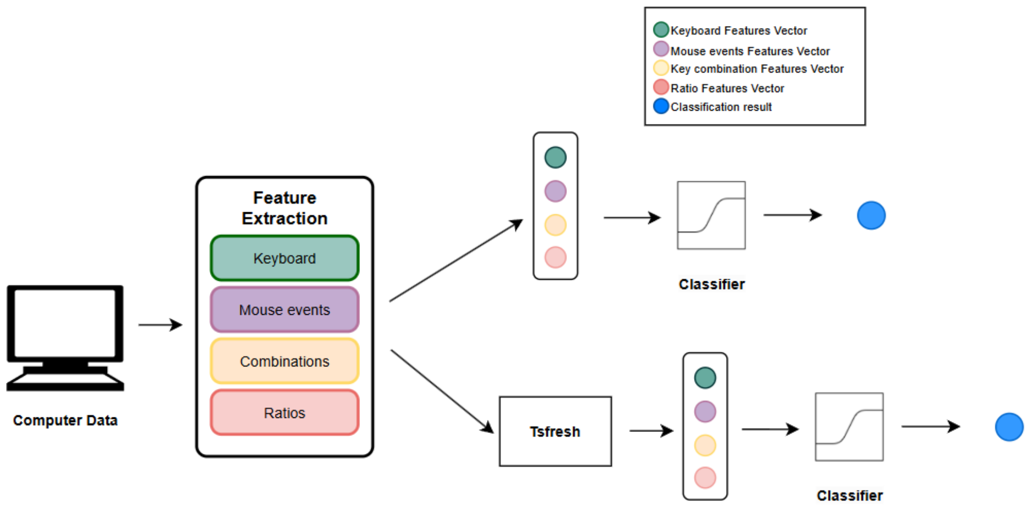

Figure 3.3: General architecture representation of the proposed method.

3.4 Algorithms

We will describe the selected machine learning algorithms for stress detection based on computer interactions. Each algorithm will be covered in detail in its own subsection, providing insights into both the theoretical foundations they are built upon and their practical implementation.

3.4.1 Logistic Regression

For the project, we have decided to focus on the use of Logistic Regression (LR) as the main algorithms. This type of statistical model (also known as a logit model) is a Machine Learning classification algorithm that is used to predict the probability of certain classes based on some dependent variables [36, 37]. In simple terms, the logistic regression algorithms computes a sum of the input features and calculates the logistic of the result.

The sigmoid function, central to logistic regression, transforms any real value into another within the range 0 to 1, which facilitates its interpretation as a probability. This function has an ”S” shape and is mathematically defined as:

$$f(x) = \frac{1}{1+e^{-x}} \tag{1}$$

where x is the weighted sum of the input features.

The logistic regression output always falls between 0 and 1, making it ideal for binary classification tasks. A higher output value indicates a higher probability that the given sample belongs to class=1, and a lower value suggests the opposite.

Continuing with the explanation of logistic regression, it is useful to compare it with linear regression to highlight their specific differences and applications. The main difference between the two models lies in the objective function they use and the type of output they produce. The following graph, Figure 3.4, visually clarifies these fundamental differences, especially in how each model handles the relationship between the independent variables and the dependent variable.

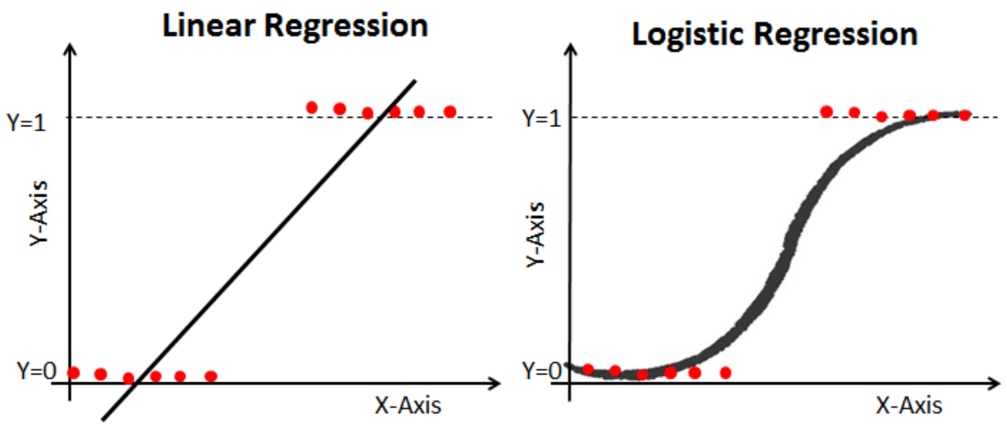

Figure 3.4: Linear Regression vs Logistic Regression [3].

3.4.2 XGBoost

XGboost [4], a scalable tree boosting system, was introduced by Chen et al. [38]. It is essentially an enhanced gradient boosting algorithm that uses the training set to predict future changes and trends in target variables. The core idea of XGBoost is to build multiple Classification and Regression Trees (CART). Each tree makes predictions independently, and the final prediction is obtained by combining the results of all trees. This model uses decision trees as base learners, constructing multiple weak learners and then training the model continuously along the gradient descent direction. The structure of the model is illustrated in Figure 3.5.

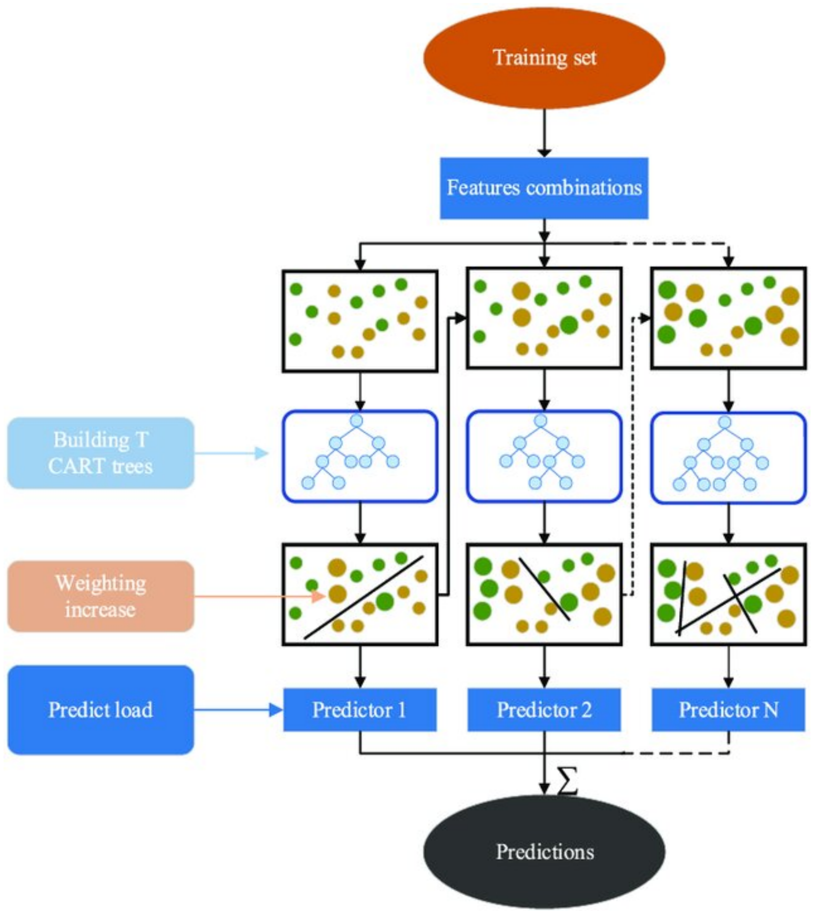

Figure 3.5: The structure of the XGBoost algorithm [4].

The mechanism of XGBoost involves adding and training new trees to fit the residual errors from the previous iteration. A predicted value is assigned to each instance by summing all corresponding leaf scores:

$$\hat{y}_i = \sum_{k=1}^{k} f_k\left(x_i\right), \quad f_k = F \tag{2}$$

where k represents a tree of the decision tree, fk is an independent function in the function space, and F is the function space defined by:

$$F = \left\{f\left(x\right) = w_{q(x)}\right\} \tag{3}$$

In this equation, q(x) indicates that the sample x is assigned to a leaf node, and w is the leaf node weight.

The objective function of the XGBoost algorithm consists of two parts: the training loss, which measures the difference between the predicted and actual values, and the regularization term, which helps prevent overfitting and controls the model complexity. The objective function is given by:

$$L(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k) \tag{4}$$

$$\Omega(f) = \Upsilon T + \frac{1}{2}\lambda \left|\left|\omega\right|\right|^2 \tag{5}$$

To minimize the objective function and set its derivative to zero, the weight of each leaf node is:

$$\omega_j^* = -\frac{\sum\limits_{i\epsilon I_j} g_i}{\sum_{i\epsilon I_j} h_i + \lambda} \tag{6}$$

Based on this, the steps to construct the XGBoost prediction model are as follows:

1. Initialize the model and construct a sub-prediction model in each iteration.
2. Before each iteration, calculate the first derivative gi and the second derivative hi of the loss function at each training sample point.
3. Generate a new decision tree and calculate the corresponding prediction value of each leaf node as shown in Figure 3.5
4. After each iteration, the newly generated model is added to the previous model. After several rounds of iteration, the final prediction model is obtained.

3.4.3 CatBoost

CatBoost is a gradient boosting library specifically designed to handle categorical data. This advanced machine learning method improves upon the traditional gradient boosting decision tree (GBDT) by addressing challenges such as noisy data, heterogeneous features, and complex dependencies [5]. CatBoost excels in managing categorical features effectively. Typically, the traditional GBDT algorithm replaces categorical features with their corresponding average label values, known as Greedy Target-based Statistics (Greedy TBS). The GTBS is defined as follows:

$$\frac{\sum_{j=1}^{p}[x_{j,k} = x_{i,k}]Y_i}{\sum_{j=1}^{p}[x_{j,k} = x_{i,k}]} \tag{7}$$

Features usually contain more information than labels. If average label value is forcefully used to represent features, it can lead to a conditional shift. CatBoost addresses this by adding an initial value to Greedy TBS. Given a dataset of observations D = Xi, Yi I = 1,

. . . , n, and a permutation = (1, . . . , n), xp,k is substituted with:

$$\frac{\sum_{j=1}^{p-1}[x_{\sigma(j,k)} = x_{\sigma(p,k)}]Y_i + aP}{\sum_{j=1}^{p-1}[x_{\sigma(j,k)} = x_{\sigma(p,k)}] + a} \tag{8}$$

where p is a prior value and a is the weight of the prior value. This method helps reduce noise from low-frequency categories. CatBoost integrates multiple categorical features by greedily combining all categorical features and their interactions in the current tree with all categorical features in the dataset, as shown in Figure 3.6. Additionally, CatBoost addresses gradient bias found in traditional GBDT by using a method called ordered boosting, which alters gradient estimation in the classic algorithm. This technique mitigates prediction shift caused by gradient bias and enhances the model's generalization ability.

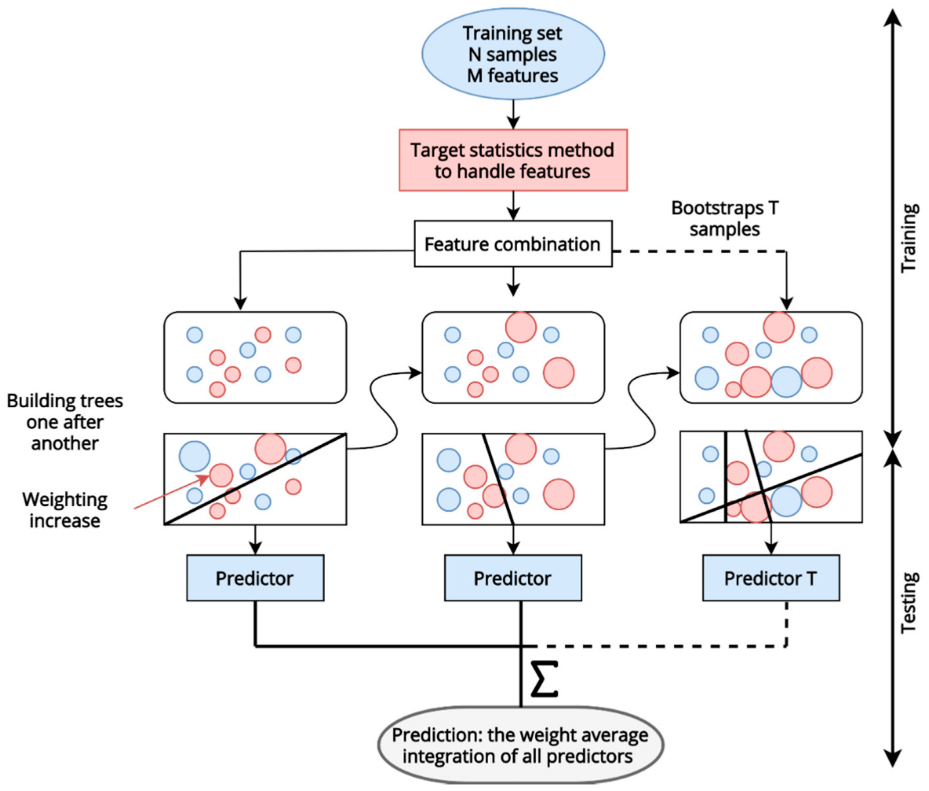
Training set
N samples
M features
Target statistics method to handle features
Feature combination
Bootstraps T samples
Building trees one after another
Weighting increase
Predictor
Predictor
Predictor T
Σ
Prediction: the weight average integration of all predictors
Training
Testing

Figure 3.6: The structure of the CatBoost algorithm [5].

CHAPTER 4

Evaluation

This chapter sets the context for the evaluation of the classification models developed, presenting the main metrics and describing in detail the results obtained. It focuses on the interpretation and assessment of the performance of the models under various key metrics, providing a comprehensive evaluation.

4.1 Metrics

In the metrics section of this chapter, we will discuss the key criteria used to evaluate the effectiveness of the classification models used in our project.

In our case, we are dealing with a binary classification problem to detect whether certain mouse and keyboard dynamics may be indicators of stress or have potential as predictors of stress level. We will use specific metrics, as our goal is to determine the presence or absence of stress in individuals, categorizing each instance into one of two possible conditions: stressed or not stressed.

Before detailing the specific metrics, it is essential to introduce the confusion matrix (Figure 4.1), a powerful tool for visualizing the performance of a classification model. The confusion matrix provides a summary of the correct and incorrect classifications made by the model compared to the actual observed results. This matrix not only allows us to see how many predictions were correct, but also to identify where and how the model is wrong. It is particularly useful for understanding the relationship between true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN), which are critical for calculating the various performance metrics.

To better understand these concepts, let's define each component of the confusion matrix:

- **True Positive (TP):** is predicted as True and is True in reality. This value represents the cases in which the model was correct in identifying a positive condition.

- **True Negative (TN):** is predicted as False and is False in reality. This value indicates the cases in which the model was correct in identifying a negative condition correctly.

- **False Positive (FP):** is predicted as True and False in reality. This value points out the errors of the model where it incorrectly identified a positive condition when it was actually negative.

- **False Negative (FN):** is predicted as False and is True in reality. This value shows the errors of the model where it failed to identify a positive condition, incorrectly classifying it as negative.

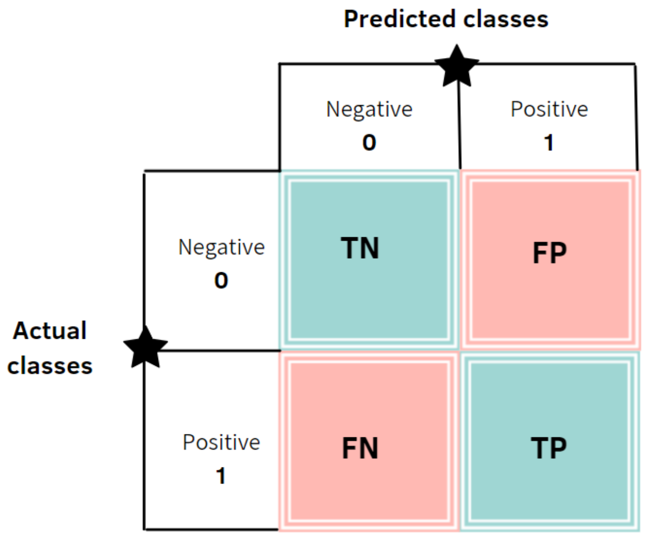

Figure 4.1: Confusion matrix [6].

We will detail each of the following metrics: precision, recall, F1-score, accuracy, macro average, and weighted average. These metrics are fundamental to understanding not only the overall efficiency of the models but also their ability to handle unbalanced classes and their performance in different aspects of classification. This detailed understanding of the metrics will allow us to select and adjust models that are not only accurate, but also fair and equitable in their predictions.

- **Precision:** measures the ratio of correct positive predictions to the total number of positive predictions made [39, 40].

$$Precision = \frac{TP}{TP + FP} \tag{9}$$

Where TP is the number of True Positives and FP is the number of False Positives.

- **Recall:** measures the ratio of correct positive predictions to the total actual positives [39, 40].

$$Recall = \frac{TP}{TP + FN} \tag{10}$$

Where TP is the number of True Positives and FN is the number of False Negatives.

- **F1-Score:** calculates the harmonic mean of precision and recall, providing a balance between the two [39, 40].

$$F1 - Score = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{11}$$

It is especially useful when seeking a balance between Precision and Recall.

- **Accuracy:** computes a global average F1 score by summing the True Positives (TP), False Negatives (FN), and False Positives (FP) [40].
- **Macro average:** computes the arithmetic mean (also known as unweighted mean) of all per-class F1 scores [40].
- **Weighted average:** calculates the mean of all per-class F1 scores, considering each class's support [40].

4.2 Results

In this section, we present the results obtained from the experiments conducted on the collected and preprocessed data. The experiments were conducted using various machine learning classification techniques to evaluate their effectiveness in detecting stress through user interactions with the computer. Features extracted from typing patterns and mouse movements were used to train and test the models. The results are analyzed against several performance metrics, including precision, recall, and F1 score, providing a comprehensive view of the models' ability to correctly identify stress levels. This critical evaluation allows for a better understanding of the strengths and limitations of each approach, as well as their applicability in real-world contexts.

We evaluated the XGBoost, CatBoost, and Logistic Regression models to analyze their ability to predict stress. The results are presented for two different cases: one without using the tsfresh library for feature extraction and the other using it. This comparison will allow us to understand the impact of feature extraction with tsfresh on the performance of the models and their ability to detect stress levels in users.

In all the experiments performed, we evaluated the three models using StandardScaler pipelines and cross-validation. To ensure consistency and robustness of the results, we implemented the stratified KFold cross-validation method (StratifiedKFold) with 10 splits (n_splits=10), a random seed of 42 (random_state=42) and with data shuffling (shuffle=True). This approach allows us to systematically and fairly evaluate model performance, ensuring that each validation iteration includes a balanced representation of stress and non-stress classes.

We have conducted several experiments to evaluate the two stress detection approaches. In the first case, we used the original features extracted directly from mouse typing patterns and movements, without applying additional feature extraction techniques. Experiments in this case included hyperparameter optimization using GridSearchCV and feature selection using RFECV to improve model performance.

In the second case, we applied the tsfresh library to increase the number of extracted features, generating a more detailed dataset. This approach was mainly evaluated in the last experiment, which combined hyperparameter optimization and feature selection, to observe performance improvements.

4.2.1 Case 1

4.2.1.1 Baseline

In this first experiment we used all the features of the SWELL dataset and the results are shown in Table 4.1.

Model	Precision	Recall	F1-Score
XGB	0.55	0.55	0.55
CAT	0.58	0.58	0.58
LR	0.55	0.55	0.55

Table 4.1: Summary of overall performance metrics by model (4.2.1.1)

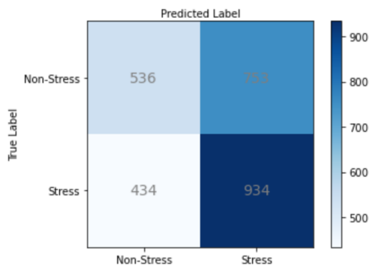

Figure 4.2: Confusion Matrix XGB

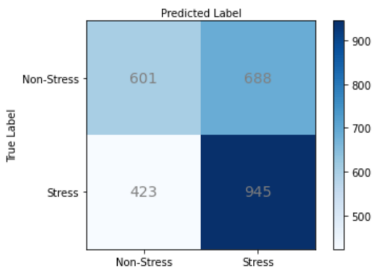

Figure 4.3: Confusion Matrix CAT

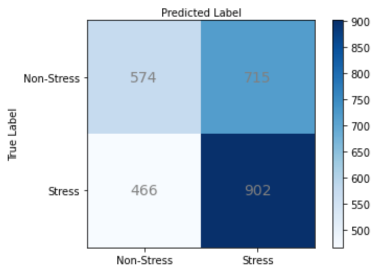

Figure 4.4: Confusion Matrix LR

In this experiment the performance metrics showed that CAT is the most effective model in terms of precision, recall and F1-Score. The confusion matrices (Figures 4.2, 4.3 and 4.4) reinforced these results, showing that CAT handles correct predictions better and reduces classification errors compared to XGB and LR.

4.2.1.2 Hyperparameters Optimization for Different Classification Models Using GridSearchCV

In this experiment, we optimized the hyperparameters of the different models using GridSearchCV to improve stress prediction. GridSearchCV identified the best hyperparameters. The results show that the optimization significantly improved the performance of the models in stress detection, as detailed in the corresponding table.

Model	Precision	Recall	F1-Score
XGB	0.57	0.57	0.57
CAT	0.59	0.59	0.59
LR	0.59	0.59	0.59

Table 4.2: Summary of overall performance metrics by model (4.2.1.2)

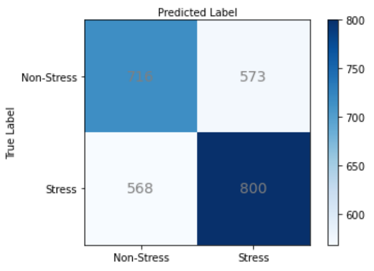

Figure 4.5: Confusion Matrix XGB

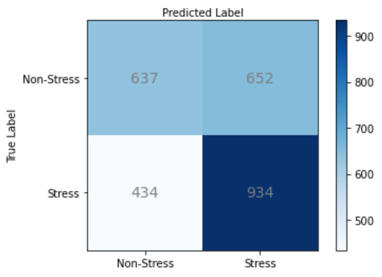

Figure 4.6: Confusion Matrix CAT

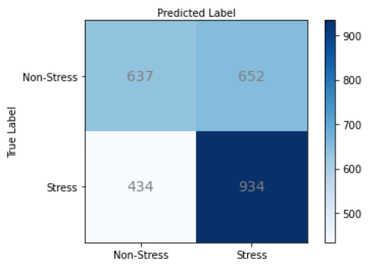

Figure 4.7: Confusion Matrix LR

In the second experiment, all models showed improvements in their performance metrics compared to the first experiment. CAT and LR emerged as the best performing models, each achieving values of 0.59 in precision, recall and F1-Score. XGB also demonstrated a marked improvement, with all metrics achieving 0.57. The confusion matrices (Figures 4.5, 4.6 and 4.7) support these results, showing that both CAT and LR significantly reduced false negatives and false positives. XGB, although improved, still exhibited more classification errors compared to CAT and LR. In conclusion, optimization of the models using GridSearchCV resulted in an overall improvement in performance.

4.2.1.3 Feature Selection and Model Evaluation Using RFECV

In this experiment, we apply feature reduction techniques to improve model performance and reduce training time. We use Recursive Feature Elimination with Cross Validation (RFECV) to iteratively select the most relevant features. We evaluated all three models, configuring different feature elimination steps for each model. The results detailed in the corresponding table.

Model	**Precision**	**Recall**	**F1-Score**
XGB	0.55	0.55	0.55
CAT	0.58	0.58	0.58
LR	0.55	0.55	0.55

Table 4.3: Summary of overall performance metrics by model (4.2.1.3)

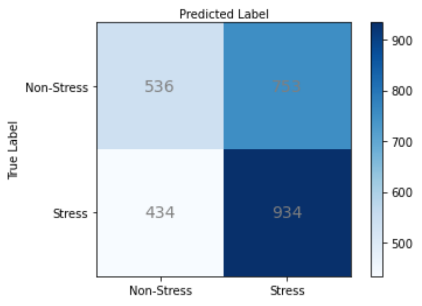

Figure 4.8: Confusion Matrix XGB

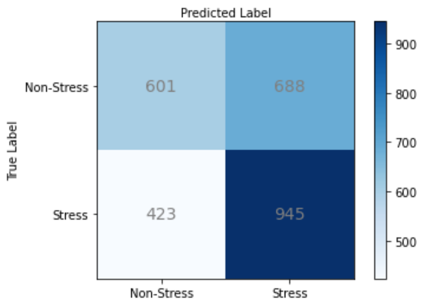

Figure 4.9: Confusion Matrix CAT

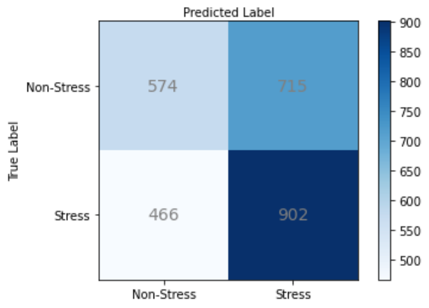

Figure 4.10: Confusion Matrix LR

Experiment 4.2.1.3, which involved feature selection by RFECV, showed that CAT stood out slightly above the others with values of 0.58 in precision, recall and F1-Score, while XGB and LR achieved values of 0.55 in all metrics. The confusion matrices (Figures 4.8, 4.9 and 4.10) confirm these results, indicating that CAT handles correct predictions better and reduces classification errors compared to XGB and LR. In summary, although all models showed good performance, CAT proved to be the most effective in this experiment.

4.2.1.4 Combined Hyperparameter Optimization and Feature Selection Using GridSearchCV and RFECV

In this experiment, we combine the features selected by RFECV in Experiment 4.2.1.3 with hyperparameter optimization using GridSearchCV for the three models. We accessed previously selected features and performed GridSearchCV to adjust the hyperparameters. This combination allowed us to find the best configuration for each model. The results indicate that this strategy significantly improved the performance of the models in predicting stress, as shown in the corresponding table.

Model	**Precision**	**Recall**	**F1-Score**
XGB	0.57	0.57	0.57
CAT	0.59	0.59	0.59
LR	0.55	0.55	0.55

Table 4.4: Summary of overall performance metrics by model (4.2.1.4)

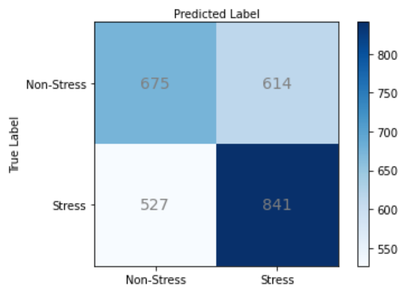

Figure 4.11: Confusion Matrix XGB

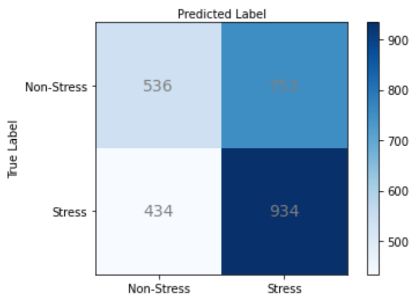

Figure 4.12: Confusion Matrix CAT

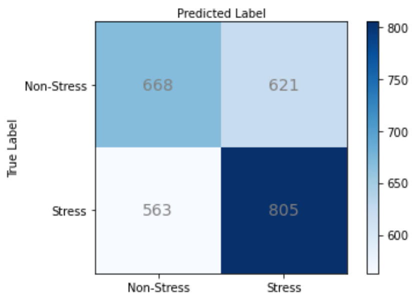

Figure 4.13: Confusion Matrix LR

The analysis of Experiment 4.2.1.4 reveals improvement in performance metrics. CAT continued to be the most effective model, achieving values of 0.59 in precision, recall and F1-Score. XGB showed moderate improvement with a value of 0.57 in all metrics, while LR achieved values of 0.55. The confusion matrices (Figures 4.14, 4.15 and 4.16) showed that CAT achieved a notable reduction in false positives and false negatives, showing a superior balance in classification compared to XGB and LR. In conclusion, the application of feature selection and hyperparameter optimization techniques resulted in an overall improvement in performance, consolidating CAT as the most robust model for stress detection.

To provide a complete comparison of the results obtained in the four experiments, the performance metrics for each model are summarized in the following table. This table highlights the precision, recall and F1-Score of XGBoost (XGB), CatBoost (CAT) and Logistic Regression (LR) across all experiments.

Experiment	**Algorithm**	**Precision**	**Recall**	**F1-Score**
Baseline	XGB	0.55	0.55	0.55
	CAT	0.58	0.58	0.58
	LR	0.55	0.55	0.55
Hyperparameter Optimization	XGB	0.57	0.57	0.57
	CAT	0.59	0.59	0.59
	LR	0.59	0.59	0.59
Feature Selection	XGB	0.55	0.55	0.55
	CAT	0.58	0.58	0.58
	LR	0.55	0.55	0.55
Combination	XGB	0.57	0.57	0.57
	CAT	0.59	0.59	0.59
	LR	0.55	0.55	0.55

Table 4.5: Comparison of Experimental Results

Throughout the four experiments, notable improvements in the performance of the stress detection models were observed. Initially, in the Baseline (4.2.1.1) experiment, CatBoost (CAT) proved to be the most effective model, outperforming both Logistic Regression (LR) and XGBoost (XGB). In the Hyperparameter Optimization (4.2.1.2) experiment, all models showed performance improvements, with CAT again leading the way. The Feature Selection (4.2.1.3) experiment, which incorporated feature selection with RFECV, maintained CAT's superiority with consistent values. Finally, in the Combination (4.2.1.4) experiment, CAT continued to excel with a performance score. In conclusion, throughout all experiments, CAT proved to be the most effective model in stress detection, benefiting particularly from hyperparameter optimization and feature selection techniques. This combination significantly improved the model's capability, consolidating CAT as the most robust model for stress prediction in our study.

4.2.2 Case 2

In this case, we have decided to apply the tsfresh library only in the last experiment. This decision is based on the observation that the improvement in results is more remarkable when advanced feature extraction techniques are used in later stages of the modeling process. By incorporating tsfresh in the last experiment, we were able to capture more complex patterns and trends in the data, resulting in a significant increase in model performance for stress detection.

4.2.2.1 Combined Hyperparameter Optimization and Feature Selection Using GridSearchCV and RFECV

In this experiment, we combine the features selected by RFECV in Experiment 4.2.1.3 with hyperparameter optimization using GridSearchCV for the three models. We used pipelines with StandardScaler and configured model-specific hyperparameter ranges. We accessed previously selected features and performed GridSearchCV with 10-fold cross-validation to adjust the hyperparameters. This combination allowed us to find the best configuration for each model. The results indicate that this strategy significantly improved the performance of the models in predicting stress, as shown in the corresponding table.

Model	**Precision**	**Recall**	**F1-Score**
XGB	0.92	0.92	0.92
CAT	0.82	0.81	0.81
LR	0.65	0.64	0.64

Table 4.6: Summary of overall performance metrics by model (4.2.2.1)

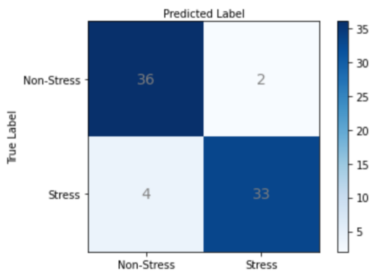

Figure 4.14: Confusion Matrix XGB

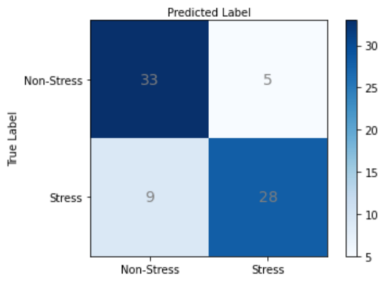

Figure 4.15: Confusion Matrix CAT

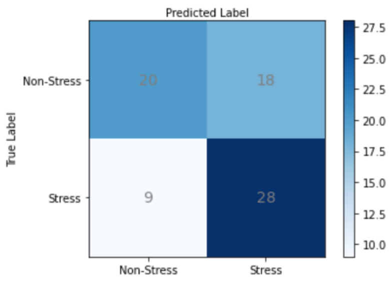

Figure 4.16: Confusion Matrix LR

The analysis of Experiment 4 reveals a significant improvement in performance metrics. XGBoost (XGB) showed outstanding performance with an precision, recall and F1-Score of 0.92, clearly standing out from the other models. CatBoost (CAT) also improved, reaching values of 0.82 in precision and 0.81 in recall and F1-Score. On the other hand, Logistic Regression (LR) had the lowest performance with values of 0.65 in precision and around 0.64 in recall and F1-Score. The confusion matrices (Figures 4.14, 4.15 and 4.16) highlighted that XGB managed to minimize classification errors, while CAT and LR presented more errors. In summary, the combination of feature selection and hyperparameter optimization significantly favored XGB, consolidating it as the most effective model for stress prediction in this experiment.

CHAPTER 5

Conclusions

This chapter closes the analysis of the project, summarizing the main findings and highlighting the most important contributions of the study. The implications of the results obtained are explored, emphasizing their relevance in the context of the development of stress detection tools and the improvement of personal and occupational well-being. In addition, the problems faced during the development of the project are discussed, providing a comprehensive overview of the challenges and how they were addressed.

5.1 Conclusion

In this work, we have explored and developed classification models for stress detection using user-computer interactions. Our main objective was to compare different classification techniques and optimize their performance in accurate stress detection using Machine Learning and User Behavior Analysis techniques.

In today's digital era, stress related to work and continuous use of electronic devices has increased significantly. Identifying and managing stress is crucial to improve the well-being and productivity of individuals. However, accurate and non-intrusive detection of stress remains a challenge. This project advances the creation of a system capable of predicting stress using data from typing patterns and mouse movements.

The process began with a comprehensive review of the state of the art to identify the most effective methodologies and most promising classification techniques in the field of stress detection. Then, we acquired and preprocessed the SWELL dataset, which provided a solid basis for training and evaluating our models.

For feature extraction, we employed two different approaches: one without the use of the tsfresh library and one using tsfresh to increase the number of features. The tsfresh library was used to increase the number of extracted features, providing a total of 12432 advanced features. This allowed a better representation of the data, focusing on the most significant aspects influencing stress perception.

Subsequently, experiments were carried out using various classification models, including XGBoost, CatBoost and Logistic Regression, evaluating their performances using a variety of key metrics. The models were optimized through feature selection and hyperparameter optimization techniques, using methods such as RFECV and GridSearchCV. These techniques allowed us to identify the most efficient configurations for each model.

The results showed significant performance improvements, especially when feature selection and hyperparameter optimization were combined. In the approach without tsfresh, the best model was CatBoost, which achieved an precision, recall and F1-Score of 0.59. On the other hand, in the approach using tsfresh, the best model was XGBoost, achieving an precision, recall and F1-Score of 0.92 .

In conclusion, this project has demonstrated the feasibility and effectiveness of using machine learning models for stress detection through computer interactions. The combination of feature selection and hyperparameter optimization techniques has significantly improved the performance of the models. The comparison of the two approaches has provided insight into the positive impact of feature extraction with tsfresh on the detection of stress levels in users, suggesting that the increase in the number of features can lead to an improvement in the precision and reliability of the classification models. The results obtained provide a solid foundation for future research and applications, with the potential to contribute significantly to the improvement of well-being and mental health in the work environment.

5.2 Achieved goals

This section presents the objectives achieved in the development of the project, which aimed to create a stress detection system based on the user's interactions with the computer.

- **State-of-the-Art Review to Analyze Current Solutions**

  A comprehensive review of the state of the art in stress detection techniques was conducted, investigating the most recent and relevant ones. Approaches and classification models used in previous studies were analyzed, identifying the most effective strategies and evaluating the advantages, limitations and challenges of current solutions. The review focused especially on non-intrusive methods, providing a solid foundation for the development of the project.

- **Preprocessing of the Datasets**

  The datasets were thoroughly preprocessed to remove elements that could adversely affect subsequent analysis. Duplicate and null values were removed, and the labels of the datasets were checked and balanced. In addition, the Tsfresh library was used to generate advanced features, consolidating the samples into a structure that improves computational efficiency and data quality.

- **Machine Learning Model Design**

  Several machine learning techniques were applied to the datasets, developing models capable of detecting stress based on the available features. Algorithms such as XGBoost, CatBoost and Logistic Regression were designed and evaluated, optimizing their hyperparameters and selecting the most relevant features using advanced techniques such as RFECV and GridSearchCV.

- **Evaluation and Experimentation**

  The performance of the models was thoroughly validated and evaluated through controlled experiments. The models were subjected to cross-validation processes and key performance metrics such as precision, recall and F1-score were used to evaluate their efficacy. The results showed significant improvements in performance, with the XGBoost model standing out especially after the combination of feature selection and hyperparameter optimization.

5.3 Problems Faced

In this section, we will discuss the main problems and challenges encountered during the development of the project. These problems range from technical issues related to data collection and preprocessing to difficulties in the optimization and evaluation of the machine learning models. Some of the most significant problems encountered are detailed below.

- **Availability of Combined Datasets**

  One of the most significant problems we faced was the lack of datasets that combined mouse and keyboard interactions. In many research articles, although studies on these interactions were mentioned, the datasets used were not provided publicly due to privacy concerns. This limitation made it difficult to obtain the data needed to develop and train our stress detection models. However, we managed to find the SWELL dataset, which includes both mouse and keyboard interactions, and decided to use it for this project. This dataset turned out to be fundamental to be able to carry out our experiments and validate our models effectively.

- **Balancing classes in the Dataset**

  Class balancing was another significant challenge during project development. The original dataset had an unbalanced distribution of stress classes, which could bias the machine learning models toward the majority class. To address this problem, we applied a binarization technique based on the median stress levels, which allowed balancing the classes in the dataset. Proper balancing of classes is crucial for models to learn in a fair and equitable manner, and to ensure that predictions are accurate and reliable for both stress classes.

- **Model Training Time**

  Another significant problem we encountered was the amount of time the models took to generate results. When using Tsfresh to extract features, we obtained 12432 features, which increased the computational load considerably. Training the models with so many features caused very long training times and, in some cases, even brought the Jupyter Notebook kernel to a halt. This situation forced us to consider dimensionality reduction and code optimization techniques to efficiently handle the large volume of data and ensure that training the models was more manageable and effective.

5.4 Future Work

This section focuses on possible improvements and extensions that can be made to the current work. Some relevant tasks that could be addressed in the future are presented below.

One important task would be the creation of new datasets. The lack of combined datasets that include both keyboard and mouse interactions has been a significant limitation in this study. Developing new datasets that collect these data in more detail and in a variety of work contexts would improve the generalization and robustness of stress detection models.

Another essential task would be to allow users to verify whether the model's prediction of their stress level is correct or not. This direct feedback would not only improve the precision of the model, but also increase the user's confidence in the system. In addition, this information could be used to adjust and customize models based on user feedback.

A possible extension of the current work could be the use of other non-intrusive techniques for stress detection. Techniques such as tone of voice evaluation, or passive monitoring through wearable devices could complement the data obtained from computer interactions and provide a more complete picture of the user's stress state.

In addition, it would be beneficial to improve existing models. Although the results obtained have been promising, increasing performance and obtaining better results is crucial to provide more effective tools. Experimenting with neural network architectures, such as recurrent neural networks (RNNs) and convolutional neural networks (CNNs), as well as implementing ensemble approaches, could lead to significant improvements and generalizability of models.

These directions for future work have the potential to significantly advance the precision and utility of stress detection systems, providing more effective and reliable tools for the well-being of users in occupational and academic settings.

APPENDIX A

Impact of the project

This section explores the multifaceted impact of our project on several domains, focusing specifically on the social, economic, and ethical aspects. Understanding these impacts is crucial to appreciating the broader implications of implementing stress detection models based on computer interactions. The analysis will provide a complete picture of how the project can influence society, generate economic benefits, and address ethical considerations.

A.1 Social impact

Given the growing concern for health and well-being in the workplace, this work offers a non-intrusive and effective tool to identify and prevent stress among workers.

The implementation of machine learning models that analyze typing patterns and mouse movements provides an innovative solution to improve the quality of life of employees. By facilitating early detection of stress, companies can take proactive measures to reduce workload, improve working conditions, and promote a healthier and more productive work environment.

In addition, the use of accessible and low-cost technologies allows this tool to be implemented in a wide variety of settings, from small businesses to large corporations, promoting equity and mental health care in the workplace.

A.2 Economic impact

This work not only improves the health and well-being of employees, but also has the potential to generate significant economic benefits for companies.

Implementing machine learning models that analyze computer interactions can help organizations identify and mitigate factors that contribute to workplace stress. By reducing stress levels, absenteeism rates can be reduced, productivity can be improved, and costs associated with stress-related health problems can be reduced.

In addition, the use of accessible and low-cost technologies allows even small and medium-sized companies to implement this tool without prohibitive costs. This promotes equity in mental health care in the workplace, allowing more companies to access advanced solutions for the well-being of their employees.

On the other hand, the investment in human and physical resources for the development and implementation of this project is justified by the long-term benefits to be gained. The initial cost of development, which includes the acquisition of datasets, computational infrastructure and training of personnel in machine learning techniques, will be offset by reduced operating costs and increased organizational efficiency.

A.3 Ethical impact

Our work focuses on stress detection through user interaction with the computer, which involves the handling of personal and potentially sensitive data. It is crucial to ensure the privacy and confidentiality of these data, adopting strict security and anonymization measures to protect the identity of the participants.

The use of advanced technologies such as artificial intelligence and machine learning in this context raises important ethical issues related to transparency and fairness. It is essential that the algorithms used are transparent and understandable, avoiding any bias that may discriminate against certain groups of users. Transparency in methodology and results ensures that the use of these technologies is fair and equitable.

In addition, it is essential that the tools are used responsibly, avoiding any misuse that may result in invasion of privacy or non-consensual monitoring of employees.

APPENDIX B

Economic budget

This section will detail the economic budget of the project, breaking down the physical and human resources used. The costs associated with the equipment and tools used will be analyzed, as well as the expenses derived from the participation of the personnel involved in the development of the project. Finally, the total economic budget will be presented, providing a clear and complete view of the financial resources required to carry out the project.

B.1 Physical resources

In order to carry out this project, various physical resources have been required that have been essential for the implementation and development of the stress detection models. These resources include both hardware and specialized software that have allowed the collection, storage, processing and analysis of huge volumes of data. The main physical resources used, as well as their respective costs, are detailed below.

- High-Performance Computer: 1500€ - 2500€
- Data Analysis Software such as Python or Anaconda [41, 42]: Free
- Cloud Services (AWS or Google Cloud) [43, 44]: 100€ - 500€ per month
- Data Input Devices (Quality Keyboard and Mouse): 50€ - 100€

B.2 Human resources

In addition to the physical and technological resources, a highly trained and specialized team is also required. The human resources involved in the development of the project are detailed below, highlighting the time and effort dedicated to achieve the objectives set. These professionals have been fundamental to ensure the quality and success of the project, contributing their experience and knowledge in several key areas.

- Personnel cost: 300 hours x 13,80€ (Price per hour [45]).

B.3 Total budget

Resource	**Quantity**	**Price per unit(€)**	**Total price(€)**
High-Performance Computer	1	2000	4000
Data Analysis Software	1	0	0
Cloud Services	1 year	3000	3000
Data Input Devices	2	100	200
Personnel cost	300 hours	15.38	4614
Total			**11814**

Table B.1: Total estimated budget

Bibliography

[1] DatabaseCamp. Reinforcement learning. Accessed: April 5, 2024.

[2] Maximilian Christ, Nils Braun, Julius Neuffer, and Andreas W. Kempa-Liehr. Time series feature extraction on basis of scalable hypothesis tests (tsfresh – a python package). *Neurocomputing*, 307:72–77, 2018.

[3] DataCamp. Understanding logistic regression in python, 2024.

[4] Xiaotong Yao, Xiaoli Fu, and Chaofei Zong. Short-term load forecasting method based on feature preference strategy and lightgbm-xgboost. *IEEE Access*, 10:75257–75268, 2022.

[5] Shihab Ahmad Shahriar, Imrul Kayes, Kamrul Hasan, Mahadi Hasan, Rashik Islam, Norrimi Rosaida Awang, Zulhazman Hamzah, Aweng Eh Rak, and Mohammed Abdus Salam. Potential of arima-ann, arima-svm, dt and catboost for atmospheric pm2. 5 forecasting in bangladesh. *Atmosphere*, 12(1):100, 2021.

[6] Towards AI. Deep understanding of confusion matrix. 2024.

[7] Saskia Koldijk, Mark A Neerincx, and Wessel Kraaij. Detecting work stress in offices by combining unobtrusive sensors. *IEEE Transactions on affective computing*, 9(2):227–239, 2016.

[8] American Psychological Association. Stress: Concepts, Definition, and History, Unknown. Accessed on: January 20, 2024.

[9] University of York. Occupational Stress: Definition, Types, Causes Management.

[10] World Health Organization (WHO). Occupational health: Stress at the workplace, 2020.

[11] Harvard Health Publishing. Understanding Chronic Stress.

[12] Simply Psychology. Causes of work stress.

[13] Sabine Sonnentag and Michael Frese. Stress in organizations. In Walter C. Borman, Daniel R. Ilgen, and Richard J. Klimoski, editors, *Handbook of Psychology: Industrial and Organizational Psychology*, volume 12, pages 453–491. John Wiley Sons, Hoboken, NJ, 2003.

[14] Daniel C. Ganster and Christopher C. Rosen. Work stress and employee health: A multidisciplinary review. *Journal of Management*, 39(5):1085–1122, 2013.

[15] Davide Carneiro, Paulo Novais, Juan Carlos Augusto, and Nicola Payne. New methods for stress assessment and monitoring at the workplace. *IEEE Transactions on Affective Computing*, 10(2):237–254, 2019.

[16] Yasin Sinan Can, Negin Chalabianloo, Duygu Ekiz, and Cem Ersoy. Continuous stress detection using wearable sensors in real life: Algorithmic programming contest case study. *Sensors*, 19(8):1849, 2019.

[17] Thelma Androutsou, Spyridon Angelopoulos, Evangelos Hristoforou, George K Matsopoulos, and Dimitrios D Koutsouris. Automated multimodal stress detection in computer office workspace. *Electronics*, 12(11):2528, 2023.

[18] Mara Naegelin, Raphael P Weibel, Jasmine I Kerr, Victor R Schinazi, Roberto La Marca, Florian von Wangenheim, Christoph Hoelscher, and Andrea Ferrario. An interpretable machine learning approach to multimodal stress detection in a simulated office environment. *Journal of Biomedical Informatics*, 139:104299, 2023.

[19] Lucia Pepa, Antonio Sabatelli, Lucio Ciabattoni, Andrea Monteriu, Fabrizio Lamberti, and Lia Morra. Stress detection in computer users from keyboard and mouse dynamics. *IEEE Transactions on Consumer Electronics*, 67(1):12–19, 2020.

[20] David Sun, Pablo Paredes, and John Canny. Moustress: detecting stress from mouse motion. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 61–70, 2014.

[21] Amir Muaremi, Bert Arnrich, and Gerhard Tröster. Towards measuring stress with smartphones and wearable devices during workday and sleep. *BioNanoScience*, 3:172–183, 2013.

[22] Javier Hernandez, Pablo Paredes, Asta Roseway, and Mary Czerwinski. Under pressure: sensing stress of computer users. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 51–60, 2014.

[23] Gopinath Rebala, Ajay Ravi, Sanjay Churiwala, Gopinath Rebala, Ajay Ravi, and Sanjay Churiwala. Machine learning definition and basics. *An introduction to machine learning*, pages 1–17, 2019.

[24] Iqbal H Sarker. Machine learning: Algorithms, real-world applications and research directions. *SN computer science*, 2(3):160, 2021.

[25] Ethem Alpaydin. *Introduction to machine learning.* MIT press, 2020.

[26] Rachel Wolff. 5 types of classification algorithms in machine learning, August 2020.

[27] Python Software Foundation. Python Essays.

[28] Sloan Kelly and Sloan Kelly. What is python? *Python, PyGame and Raspberry Pi Game Development*, pages 3–5, 2016.

[29] Wes McKinney and PD Team. Pandas-powerful python data analysis toolkit. *Pandas—Powerful Python Data Analysis Toolkit*, 1625, 2015.

[30] Ivan Idris. *NumPy: Beginner's Guide.* Packt Publishing Ltd, 2015.

[31] Kevin Jolly. *Machine learning with scikit-learn quick start guide: classification, regression, and clustering techniques in Python.* Packt Publishing Ltd, 2018.

[32] GSIC-EMIC, Universidad Politécnica de Madrid. Introduction to Scikit-Learn, 2022.

[33] TSFresh Developers. TSFresh Documentation: Introduction.

[34] Saskia Koldijk, Maya Sappelli, Suzan Verberne, Mark A Neerincx, and Wessel Kraaij. The swell knowledge work dataset for stress and user modeling research. In *Proceedings of the 16th international conference on multimodal interaction*, pages 291–298, 2014.

[35] Varun Mishra, Tian Hao, Si Sun, Kimberly N Walter, Marion J Ball, Ching-Hua Chen, and Xinxin Zhu. Investigating the role of context in perceived stress detection in the wild. In *Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers*, pages 1708–1716, 2018.

[36] IBM. Logistic regression, 2024.

[37] S. Jessica. How Does Logistic Regression Work? KDnuggets. https://www.kdnuggets.com/2022/07/logistic-regression-work.html.

[38] Tianqi Chen and Carlos Guestrin. *Xgboost: A scalable tree boosting system.* 2016.

[39] Hercules Dalianis and Hercules Dalianis. Evaluation metrics and evaluation. *Clinical Text Mining: secondary use of electronic patient records*, pages 45–53, 2018.

[40] Kenneth Leung. Micro, macro & weighted averages of f1 score, clearly explained. *Towards Data Science*, 4, 2022.

[41] Anaconda. Anaconda, 2023.

[42] Python Software Foundation. Python, 2023.

[43] Amazon. Amazon web services pricing, 2023.

[44] Google. Google cloud pricing, 2023.

[45] Talent.com. Salario de ingeniero en 2024, 2024. Accedido: 20-06-2024.

[46] N Schneiderman, G Ironson, and SD Siegel. Estrés y salud: determinantes psicológicos, conductuales y biológicos. *Annu Rev Clin Psychol*, 1:607–628, 2005.

[47] Govindasamy Shanmugasundaram, S Yazhini, E Hemapratha, and S Nithya. A comprehensive review on stress detection techniques. In *2019 IEEE International Conference on System, Computation, Automation and Networking (ICSCAN)*, pages 1–6. IEEE, 2019.

[48] Shawkat Ali and Kate A Smith. On learning algorithm selection for classification. *Applied Soft Computing*, 6(2):119–138, 2006.

[49] Bradley J Erickson and Felipe Kitamura. Magician's corner: 9. performance metrics for machine learning models, 2021.