**DEVRELAX: STRESS MONITORING AND RELIEVING APPLICATION FOR IT PROFESSIONALS**

Jayathilake S. M. D. A. R.

IT20037338

BSc (Hons) in Information Technology Specializing in Software Engineering

Department of Information Technology

Sri Lanka Institute of Information Technology

Sri Lanka

September 2023

**DEVRELAX: STRESS MONITORING AND RELIEVING APPLICATION FOR IT PROFESSIONALS**

Jayathilake S. M. D. A. R.

IT20037338

Dissertation submitted in partial fulfillment of the requirements for the Bachelor of Science (Hons) in Information Technology Specializing in Software Engineering

Department of Information Technology

Sri Lanka Institute of Information Technology

Sri Lanka

September 2023

# DECLARATION

I declare that this is my own work and this dissertation1 does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

A close-up of a signature

Description automatically generatedAlso, I hereby grant to Sri Lanka Institute of Information Technology, the non-exclusive right to reproduce and distribute my dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

Signature: Date: 08/09/2023

The above candidate has carried out research for the bachelor’s degree Dissertation under my supervision.



Signature of the supervisor: Date: 08/09/2023

(Mr. Samadhi Rathnayake)

# ABSTRACT

In today's rapidly evolving workplace, stress has become a pervasive concern, impacting the well-being and productivity of software developers. The primary problem addressed in this report is the need for timely and accurate stress detection for developers to mitigate the adverse effects of stress on their well-being and productivity. The purpose of this research paper is to demonstrate the feasibility of accurate stress detection for developers using cost-effective and readily available hardware components. Leveraging the ESP32 microcontroller and the MAX30102 heart rate sensor embedded within an external mouse, we introduce a non-intrusive approach to stress assessment tailored to the software development context. Our methodology involves capturing heart rate variability data and applying advanced statistical calculations to gauge stress levels. The collected data is transmitted to a backend server for analysis and storage. This research report comprehensively covers several key areas related to stress detection for software developers. It begins with an introduction highlighting workplace stress's negative impact on productivity and well-being. A detailed review of related work in stress detection and the problems associated with previous studies sets the stage for our innovative approach. The core methodology integrates affordable hardware, collects data, applies statistical analysis, and includes a machine learning model for stress detection, focusing on the mouse as the interaction device. In conclusion, this research paper presents a novel approach to addressing the critical issue of stress among software developers. The results demonstrate the effectiveness of our stress detection model and the potential for widespread use. By seamlessly integrating stress monitoring into an everyday device, we contribute to the well-being and productivity of developers in the software development industry.

Keywords: ESP32 Microcontroller, MAX30102 Heart Rate Sensor, HRV, Decision Tree, Stress, External Mouse

# ACKNOWLEDGMENT

We would like to extend our heartfelt gratitude to our project supervisor, Mrs. Samadhi Rathnayake for his continuous guidance and unwavering support, which played a pivotal role in the successful completion of our undergraduate research. Additionally, we are deeply thankful to Mrs. Devanshi Ganegoda, our co-supervisor, for his readiness to assist us throughout the research project.

Our sincere appreciation also goes to our external supervisors, Mrs. Arosha Dasanayaka, a psychologist, and Mr. Thilanga Jayarathne, CEO of Xinotech Technologies company. Mrs. Dasanayaka's psychological expertise enriched our research, while Mr. Jayarathne's insights from the tech industry provided valuable perspectives.

We would like to express our sincere gratitude to Xinotech Technologies for their invaluable support in testing and facilitating the commercialization of our application. Mr. Thilanga Jayarathne, CEO of Xinotech Technologies, played a pivotal role in this endeavor by offering both testing resources and insights to enhance our project. Our project's success would not have been possible without Xinotech Technologies' willingness to collaborate and assist in bringing our application to the commercial market. We deeply appreciate their contributions to our research and their commitment to advancing technology in practical ways.

We express our profound thanks to our lecturer in charge, Dr. Jayantha Amararachchi, and the CDAP team for their unwavering support, ensuring the project's success. Furthermore, we are immensely grateful to our friends and teammates for their constant encouragement and assistance.

Last but not least, we want to convey our deepest appreciation to our parents for their enduring support and encouragement throughout this journey.

# TABLE OF CONTENTS

[DECLARATION 3](#_Toc145287693)

[ABSTRACT 4](#_Toc145287694)

[ACKNOWLEDGMENT 5](#_Toc145287695)

[TABLE OF CONTENTS 6](#_Toc145287696)

[LIST OF FIGURES 7](#_Toc145287697)

[LIST OF TABLES 9](#_Toc145287698)

[1. INTRODUCTION 10](#_Toc145287699)

[1.1 Background Literature 10](#_Toc145287700)

[1.2 Research Gap 14](#_Toc145287701)

[1.3 Research Problem 18](#_Toc145287702)

[1.4 Research Objectives 21](#_Toc145287703)

[1.4.1 Main Objective 21](#_Toc145287704)

[1.4.2 Specific Objectives 22](#_Toc145287705)

[2. METHODOLOGY 24](#_Toc145287706)

[2.1 Methodology 24](#_Toc145287707)

[2.1.1.1 Sensor Evaluation 24](#_Toc145287708)

[2.1.1.2 Microcontroller Selection 26](#_Toc145287709)

[2.1.1.3 Tools and Technologies 27](#_Toc145287710)

[2.1.1.4 Prototype Development 28](#_Toc145287711)

[2.1.1.5 Device Setup 29](#_Toc145287712)

[2.1.1.6 Sensor Calibration and Configuration 29](#_Toc145287713)

[2.1.1.7 Data Collection 30](#_Toc145287714)

[2.1.1.8 Data Visualization and User Interface 30](#_Toc145287715)

[2.1.1.9 Data Transmission to Blynk Cloud and Backend 30](#_Toc145287716)

[2.1.2.0 Backend Processing and Storage 31](#_Toc145287717)

[2.1.2.1 Data retrieval and prediction 32](#_Toc145287718)

[2.1.2.2 Predictive Modelling 33](#_Toc145287719)

[2.1.2.3 Dataset Utilization for machine learning model 33](#_Toc145287720)

[2.2 Commercialization aspects of the product 41](#_Toc145287721)

[2.2.1 Target Market 41](#_Toc145287722)

[2.2.2 Marketing and Revenue 41](#_Toc145287723)

[2.2.3 Marketing Approach 42](#_Toc145287724)

[2.3 Testing & Implementation 44](#_Toc145287725)

[2.3.1. Sensor Evaluation Testing Code 44](#_Toc145287726)

[2.3.2 Heart rate and Oxygen Levels Calculation 48](#_Toc145287727)

[2.3.3 HTTP POST Request to Flask Server: 50](#_Toc145287728)

[2.3.4 Data Management and MongoDB Integration in API: 51](#_Toc145287729)

[2.3.5 API Route for Sending Mouse Data to MongoDB: 53](#_Toc145287730)

[2.3.6 Machine Learning Model Training and Dumping: 54](#_Toc145287731)

[2.3.7 Heart Rate Prediction Route and Model Inference: 57](#_Toc145287732)

[2.3.8 Frontend Codes: 58](#_Toc145287733)

[2.3.9 Desktop Application 63](#_Toc145287734)

[3. RESULTS AND DISCUSSION 65](#_Toc145287735)

[3.1 Results 65](#_Toc145287736)

[3.2 Prototypes 70](#_Toc145287737)

[3.3 Research Findings 71](#_Toc145287738)

[3.4 Discussion 74](#_Toc145287739)

[3.5 Summary of Each Student’s Contribution 76](#_Toc145287740)

[4. CONCLUSION 78](#_Toc145287741)

[5. REFERENCES 79](#_Toc145287742)

[6. APPENDIX 81](#_Toc145287743)

# LIST OF FIGURES

[Figure 1: Circuit Diagram 29](#_Toc145287744)

[Figure 2: Workflow 34](#_Toc145287745)

[Figure 3:Frequency analysis of selected features 39](#_Toc145287746)

[Figure 4: Extracted Features 39](#_Toc145287747)

[Figure 5: max30102 Sensor performance testing code 45](#_Toc145287748)

[Figure 6: max30100 and gy-max30100 Sensor performance testing code 46](#_Toc145287749)

[Figure 7: ky-039 Sensor performance testing code 47](#_Toc145287750)

[Figure 8: Oxygen levels calculation code 48](#_Toc145287751)

[Figure 9: Heart rate calculation code 49](#_Toc145287752)

[Figure 10: HTTP Post request code to flask server 50](#_Toc145287753)

[Figure 11: Code to manage the data in the mouse 52](#_Toc145287754)

[Figure 12: API Route code for Sending Mouse Data to MongoDB: 54](#_Toc145287755)

[Figure 13: Model Code 56](#_Toc145287756)

[Figure 14: HR prediction route code 57](#_Toc145287757)

[Figure 15: Frontend – Redux heartrate slice 59](#_Toc145287758)

[Figure 16: Frontend – Redux rootReducer 59](#_Toc145287759)

[Figure 17: Frontend – prediction GET REST API 60](#_Toc145287760)

[Figure 18: Frontend – Real time Blynk data fetch 61](#_Toc145287761)

[Figure 19: Frontend – Blynk REST API 61](#_Toc145287762)

[Figure 20: Frontend – Heart Rate prediction function 62](#_Toc145287763)

[Figure 21: Frontend – Final output to store in redux 62](#_Toc145287764)

[Figure 22: Real-time BPM and SPO2 63](#_Toc145287765)

[Figure 23: Application Dashboard 64](#_Toc145287766)

[Figure 24: Classifier Accuracy 65](#_Toc145287767)

[Figure 25: Output example of predictions 66](#_Toc145287768)

[Figure 26: Feature selection based on correlation 67](#_Toc145287769)

[Figure 27: Correlation Matrix of Features 67](#_Toc145287770)

[Figure 28: Confusion Matrix 68](#_Toc145287771)

[Figure 29:Data Partitioning and Stress Condition Distribution in the Training Set 69](#_Toc145287772)

[Figure 30: Pie chart to represent the distribution of conditions in training set 69](#_Toc145287773)

[Figure 31: Prototype 2 - ESP32, MAX30102 and OLED Powered 71](file:///C:\Users\HP\Downloads\IT20037338_Final_Report_Draft.docx#_Toc145287774)

[Figure 32: NAOS QG Gaming Mouse 71](file:///C:\Users\HP\Downloads\IT20037338_Final_Report_Draft.docx#_Toc145287775)

[Figure 33: Artificial Neural Network Accuracy 72](#_Toc145287776)

[Figure 34: Perceptron Accuracy 73](#_Toc145287777)

[Figure 35: RandomForest Accuracy 73](#_Toc145287778)

[Figure 36: Turnitin Submission 81](#_Toc145287779)

[Figure 37: Turnitin Report Details 81](#_Toc145287780)

[Figure 38: Turnitin Report 82](#_Toc145287781)

# LIST OF TABLES

[Table 1:Sensor analysis through testing codes 26](#_Toc145286989)

[Table 2: Module Analysis 28](#_Toc145286990)

[Table 3: Tools and Technologies 28](#_Toc145286991)

[Table 4: Student Contributions 78](#_Toc145286992)

# INTRODUCTION

## Background Literature

In the ever-evolving landscape of the modern workplace, stress has emerged as a pervasive and pressing concern. The relentless demands and pressures that individuals encounter on a daily basis have reached unprecedented levels, leading to heightened stress levels and feelings of overwhelm. This escalating issue carries significant ramifications, not only for individual well-being but also for productivity and the quality of work, particularly in the context of software development.

The prevalence of occupational stress has made it a central challenge in contemporary workplaces. Long-term exposure to chronic stress can result in severe consequences at both the social and mental levels, leading to a range of health and economic implications. The European Agency for Safety and Health at Work claims that stress is one of the most common work-related health problems that can cause dangerous outcomes if proactive measures are not taken. According to the research conducted by EAHC and SANCO, the estimates indicate that the total costs of work-related depression in the EU27 are nearly €620 billion per annum. [3]

Stress, when left unaddressed, can corrode the efficiency and health of developers, posing a significant threat to both their personal lives and the organizations they work for. Recognizing the urgency of this issue, it becomes paramount to enable the early detection of stress, allowing individuals to take timely and appropriate measures to manage and mitigate its effects while maintaining their efficiency and health.

Various approaches to stress detection and monitoring have been proposed in response to this growing concern. Many of these solutions span multiple domains, including healthcare and human-computer interaction, but often come with substantial costs and may not be practical for widespread use. Prior research findings indicate that computer mouse movements can serve as a potential indicator of work-related stress [1]. Existing models have predominantly relied on heart rate variability (HRV) as a reliable physiological indicator of stress, with numerous studies attesting to its efficacy in real-time stress detection [4]. However, there remains a lack of accessible solutions tailored specifically to developers' needs, particularly those that can seamlessly integrate with their work tools, such as a computer mouse.

An earlier research study focused on the detection of pulse rates through the utilization of mouse applications, employing a sophisticated system that incorporated multi-PPG sensors [2]. This advanced application demonstrated its prowess in not only measuring heart rate variability but also assessing oxygen saturation levels. Despite these impressive capabilities, there exists a notable gap in the existing body of research when it comes to effectively integrating these heart rate measurements into a real-time system for monitoring and evaluating users' stress levels as they engage in various tasks and activities. This represents a significant opportunity for further exploration and innovation in the field of stress detection.

One noteworthy study integrated a custom device, incorporating both a photoplethysmogram (PPG) sensor and a galvanic skin response (GSR) sensor within a computer mouse, to detect occupational stress in an office environment [5]. The system will meticulously record users' physiological signals, which allows the opportunity to conduct a comprehensive analysis of their behavioral patterns, encompassing both mouse and keyboard dynamics. Through this intricate examination they could extract critical parameters such as heart rate (beats per minute, BPM) and skin conductance (SC) which proved to be effective in monitoring the users’ condition and detecting their stress levels while working on the computer. However this approach needs to be further improved since the physiological data collected from the device must be combined with behavioral data regarding keyboard and mouse dynamics to develop an improved multimodal system for real time stress detection. Even though Kalman has been applied to mitigate motion and noise artifacts, there may still be concerns regarding the accuracy of the system since changes in environmental factors such as temperature, humidity and lighting conditions will interfere with signal accuracy and detection.

In another study, a novel system named MouStress was introduced, which primarily focuses on using computer mouse interactions to assess stress levels. This system operates by capturing subtle changes in muscle stiffness related to arm and hand movements during common mouse operations, employing a physiological model known as the Mass-Spring-Damper system (MSD). Stress-related metrics are then deduced based on the parameters of this model, utilizing computational procedures to provide a direct and precise estimation of these metrics [6]. However, it's essential to acknowledge that while this approach may appear innovative, there are potential limitations associated with it. The stress metrics derived from the MSD model may not always align perfectly with physiological stress levels. There could be instances where variations in muscle stiffness or hand-arm dynamics do not necessarily reflect an individual's actual stress levels. Factors such as user fatigue, physical discomfort, or cognitive load may influence mouse movements and potentially lead to misinterpretations of stress indicators.

During a previous study, The NAOS QG gaming mouse is equipped with a unique set of biosensors, including heart rate and galvanic skin response sensors, along with tracking features to measure clicks per minute and movement data. Its accompanying software provides real-time, customizable on-screen feedback through a discreet HUD and can issue acoustic warnings if stress levels rise. Notably, the mouse stores data for trend analysis, helping users understand how different aspects of gaming impact their physiological responses. In the context of stress detection for software developers, the stress detection mouse presented in this research paper offers an innovative departure from the NAOS QG gaming mouse. While both mouses utilize biosensors to monitor physiological data such as heart rate, the stress detection mouse in this research distinguishes itself by offering real-time data visualization, a feature absent in the NAOS QG. This real-time data visualization capability provides developers with immediate feedback on their stress levels as they work, allowing them to make timely adjustments to their tasks or take breaks when necessary. The NAOS QG, on the other hand, primarily stores data for later analysis, lacking the on-the-fly stress assessment that the research mouse provides. This key difference enhances the proactive nature of stress mitigation in the workplace, potentially improving the overall well-being and productivity of software developers.

Furthermore, although smartphones have the capability to passively monitor users' health and well-being through stress detection apps, there is often a lack of direct correlation between the outcomes generated by these apps and clinically validated knowledge. Additionally, the level of user engagement and motivation is not consistently factored in as a prerequisite, which can impact users' adherence to and the complete validation of such systems [7].

In the context of the preceding discussion regarding ongoing research in stress level detection, it is evident that there exist several limitations and challenges associated with implementing these systems in real-world scenarios. These limitations stem from the impracticality of these approaches for everyday use, often necessitating highly controlled environments to ensure accurate data collection while remaining susceptible to external factors that can influence data quality. Furthermore, the absence of robust supervision during the data collection process and the lack of specialized software tools to enforce data adherence to predefined standards add to the complexity of these systems. Adaptation to users' daily work routines demands additional considerations. In light of these challenges, our research endeavors to address the deficiencies of previous models and proposes a more holistic approach to stress level detection.

Through this study, our ambition extends beyond merely detecting real-time stress levels; we also aspire to identify strategies for alleviating stress and construct models employing reinforcement learning techniques. By harnessing the capabilities of backend processing and machine learning, we aim to facilitate a systematic analysis of the collected heart rate data, extracting valuable insights in a structured manner. These insights will be presented to users in real-time, in a non-intrusive fashion, as they engage in their daily tasks.

## Research Gap

The domain of stress detection and management within the context of software development and contemporary workplaces has garnered escalating interest and recognition, primarily because of the undeniable and profound influence that occupational stress exerts on the well-being and overall productivity of individuals. The relentless demands and pressures faced by individuals in the modern workforce have reached unprecedented levels, fostering an environment where stress can manifest as a pervasive and formidable adversary. As the adverse effects of stress can erode not only the efficiency and quality of work undertaken by software developers but also their physical and mental health, addressing this pressing issue has become a paramount concern.

In response to these challenges, the realm of stress management and detection has witnessed a surge of research endeavors aiming to address this pressing issue. However, despite these commendable efforts, a critical research gap persists, hindering the development of comprehensive solutions.

It is within this complex landscape of research gaps and unmet needs that this thesis finds its purpose, aiming to navigate uncharted territory and contribute innovative solutions to the pressing challenges of stress detection and management in the modern workplace. Through the integration of affordable hardware components and a user-centric approach, this research aspires to bridge these gaps effectively and enhance the well-being and productivity of software developers.

Despite the commendable efforts that have been made to tackle the challenges of stress management and detection, a critical research gap persists, impeding the development of comprehensive solutions. This gap can be defined by a set of key challenges that remain unaddressed within the current field of stress detection and management, which are listed below:

1. Limited Integration with Workplace Tools: The existing body of research has predominantly concentrated on physiological indicators of stress, such as heart rate variability (HRV). However, most of these studies have failed to seamlessly integrate these physiological measurements into the daily tools and environments software developers interact with. Although some research has delved into using specialized devices, like computer mouse, to capture physiological data, they have not fully harnessed the potential for real-time and unobtrusive stress assessment within the natural workflow of developers. This gap emphasizes the urgency for innovative approaches that seamlessly integrate stress assessment with the software development process.
2. Real-time Stress Assessment During Mouse Usage: While earlier studies have explored the use of computer mouse movements as stress indicators, they have struggled to provide real-time monitoring and immediate feedback to users during their interactions with the mouse, especially while they are engaged in software development tasks. This is a critical limitation as prompt intervention is essential for effective stress management. The ability to identify and address stress in real-time, while it is occurring, is a significant gap that this thesis intends to address.
3. Incorporating Multi-Modal Data for Enhanced Accuracy: Much of the prior research has primarily focused on physiological data like heart rate, often without effectively integrating this data with other behavioral and contextual factors. An all-encompassing stress detection system should not only measure physiological indicators but should also consider elements such as keyboard and mouse dynamics, ambient conditions, and the user's level of engagement. Such a comprehensive approach has the potential to provide a more accurate and holistic assessment of stress levels, a research area that remains underexplored.
4. Lack of Mitigation Strategies: The predominant focus of earlier research efforts has been on stress detection alone. However, an essential aspect of stress management is moving beyond mere detection and developing strategies for stress mitigation. Effective stress management involves not only identifying stress but also providing interventions or recommendations to alleviate stress. This requires research into methods like reinforcement learning that can guide users towards less stressful behaviors. The scarcity of research in this domain presents an opportunity for this thesis to contribute novel strategies for managing stress.
5. Scalability and Accessibility: Many existing solutions are characterized by high costs or require controlled environments, making them less feasible for widespread use. This research seeks to leverage affordable hardware components, notably the ESP32 microcontroller and MAX30102 heart rate sensor, and embed them within a commonly used device - an external mouse. This approach aims to enhance scalability and accessibility by ensuring that stress management tools are readily available to a broader user base.
6. User-Centric Design: Previous research at times overlooked the importance of user engagement and motivation in the context of stress management. This study places a strong emphasis on user-centric design by providing real-time, unobtrusive feedback to users while they engage in their daily tasks. By doing so, it aims to enhance user adherence to stress management strategies, an aspect often neglected in prior research endeavors.

In our pursuit to effectively bridge the identified research gaps, this thesis embarks on a multifaceted journey, guided by a comprehensive research strategy that endeavors to revolutionize the field of stress detection and management. At its core, our approach revolves around the utilization of readily accessible hardware components, specifically the ESP32 microcontroller and the MAX30102 heart rate sensor. These components are discreetly embedded within a standard computer mouse, marking a pivotal shift in the way we approach stress assessment during software development tasks. This novel integration not only represents a technological innovation but also addresses a critical research gap: the seamless integration of stress assessment tools within the natural workflow of software developers. By merging this hardware with everyday tools, we aim to provide a real-time and non-intrusive method of stress assessment that aligns harmoniously with the software development process.

Moreover, our research strategy goes beyond conventional stress detection methods by venturing into the realm of multi-modal data fusion. We recognize that stress is a complex phenomenon influenced by various factors, including physiological metrics such as heart rate, contextual elements, and user behavior. To provide a more holistic and nuanced assessment of stress levels, we explore the amalgamation of these diverse data streams. This approach not only enhances the accuracy of stress assessment but also addresses another pivotal research gap: the need for comprehensive stress detection systems that consider a wide array of variables. By incorporating both physiological and contextual data, we aim to develop a more refined understanding of stress in the software development context.

Furthermore, our research endeavors extend well beyond the realms of mere stress detection. We aspire to empower users with practical strategies for stress mitigation. Leveraging cutting-edge techniques like reinforcement learning, we aim to guide software developers towards less stressful behaviors, ultimately enhancing their overall well-being and productivity. Our methodology also incorporates the formidable capabilities of back-end processing and machine learning to derive actionable insights from the collected heart rate data. These insights are then seamlessly presented to users during their daily tasks, ensuring that stress management remains an unobtrusive and integrated part of their workflow. In this manner, we aspire not only to detect but also to effectively address stress in real-time, fostering a more productive and well-balanced environment for software developers, thereby addressing some of the most pressing research gaps in this field.

## 1.3 Research Problem

The research problem under investigation revolves around the optimization of cost-effective Internet of Things (IoT) devices, specifically the ESP32 microcontroller and the MAX30102 heart rate sensor, to facilitate accurate stress detection in software developers. This optimization is achieved through their seamless integration within an external mouse, thus enabling the analysis of heart rate variability (HRV) as a robust means of real-time stress assessment.

Furthermore, this research aims to delve into the development and application of novel algorithms that can significantly enhance the capabilities of real-time stress detection when utilizing the ESP32 and MAX30102 IoT devices in conjunction with an external mouse. These innovative algorithms are specifically tailored to address the unique demands and challenges experienced by developers during their day-to-day activities.

This multifaceted research endeavor is guided by two central questions, each of profound significance:

1. Optimization of IoT Devices for Stress Detection:

The first research problem focuses on the effective configuration, fine-tuning, and seamless integration of low-cost IoT devices, specifically the ESP32 microcontroller and the MAX30102 heart rate sensor, into an external mouse. This integration aims to ensure the precise and reliable detection of stress indicators in software developers. To address this problem, several key aspects need to be considered:

* Hardware Integration: The ESP32 microcontroller and MAX30102 sensor must be physically integrated into the external mouse in a way that is ergonomic, unobtrusive, and does not interfere with the user's normal interaction with the mouse.
* Sensor Calibration: Calibration of the MAX30102 sensor is critical to ensure accurate heart rate and oxygen saturation measurements. This involves adjusting sensor parameters and accounting for individual variations among users.
* Power Management: Efficient power management strategies must be implemented to extend the battery life of the integrated mouse while collecting continuous data over extended periods.
* Data Transmission: The collected data, including heart rate variability (HRV) measurements, should be transmitted to a backend server for analysis and storage. This requires establishing a reliable and secure data transmission protocol.
* User Experience: The design should prioritize user comfort and ease of use. Developers should be able to use the mouse without disruptions, and stress detection should be non-intrusive.

1. Algorithmic Advancements for Real-Time Stress Detection:

The second research problem delves into the development and implementation of cutting-edge algorithms to enhance the accuracy and efficiency of real-time stress detection using the ESP32 and MAX30102 IoT devices within the external mouse, specifically tailored for software developers. Here are key considerations for addressing this problem:

* Data Processing: The raw data collected from the sensors, such as heart rate and HRV data, must undergo sophisticated signal processing techniques to filter noise and artifacts, ensuring the accuracy of stress-related measurements.
* Feature Extraction: Algorithms need to extract relevant features from the sensor data that can serve as reliable indicators of stress. This may include time-domain and frequency-domain features derived from HRV data.
* Machine Learning: Machine learning models, such as classifiers or regression models, can be trained on labeled stress data to recognize patterns and make real-time predictions of stress levels. Supervised and unsupervised learning approaches may be explored.
* Real-Time Analysis: The algorithms should operate in real-time, continuously analyzing data from the mouse to provide instant feedback or alerts to the user when stress levels are detected to be elevated.
* Application Integration: To be truly beneficial to software developers, these algorithms should be seamlessly integrated into developer-centric applications and systems. This integration should consider the context of software development tasks and provide actionable insights to developers on how to manage stress effectively.
* Validation and Testing: The performance of the algorithms should be rigorously tested and validated in real-world software development scenarios to ensure their reliability and effectiveness.

In pursuit of these research problems and objectives, our study aspires to deliver a holistic and multifaceted solution that not only tackles the challenges associated with IoT device integration into an external mouse but also harnesses the power of state-of-the-art algorithms. This comprehensive approach is motivated by the recognition that addressing stress detection for software developers requires more than just hardware innovation; it necessitates the fusion of cutting-edge technology with real-world usability. By aligning hardware integration and advanced algorithms, we aspire to empower developers with the tools and insights they need to thrive in their demanding profession while safeguarding their mental and physical health.

## Research Objectives

### **1.4.1 Main Objective**

The overarching objective of this research endeavor is to pioneer the conceptualization, meticulous design, seamless implementation, and rigorous assessment of a cutting-edge stress detection system, custom-crafted to cater to the unique and demanding needs of software developers. This system is poised to leverage the practicality and affordability of readily available hardware components, most notably harnessing the ESP32 microcontroller and harnessing the capabilities of the MAX30102 heart rate sensor. These components will be thoughtfully integrated into an external mouse, forming a novel and practical tool.

This innovative stress detection system aims to provide real-time stress assessment capabilities tailored explicitly to the software development process. It is designed to be a dependable companion for software developers as they navigate the complex and high-pressure landscape of the modern workplace. By continuously monitoring and analyzing vital physiological data, such as heart rate, this system seeks to offer software developers timely insights into their stress levels during coding and development tasks.

The central aspiration is to enhance the overall well-being and productivity of software developers. Through real-time stress assessment, this system has the potential to empower developers with the information they need to manage stress effectively, optimize their workflow, and ultimately contribute to a healthier and more productive work environment within the software development community. In essence, this research strives to provide software developers with an unparalleled capability to mitigate the negative impacts of stress, fostering a more sustainable and fulfilling career path in this fast-paced and demanding field.

### **1.4.2 Specific Objectives**

1. Hardware Integration and Configuration: In the pursuit of this objective, we delve deep into the intricacies of hardware engineering. The goal is to craft a bespoke hardware configuration that harmoniously embeds the ESP32 microcontroller and the MAX30102 heart rate sensor within an external mouse. This ingenious integration will empower seamless, unobtrusive monitoring of heart rate variability (HRV) during the routine use of a mouse, an oft-neglected aspect of software developers' daily grind. By achieving this, we aim to create a hardware environment where developers can naturally interact with stress-detection mechanisms while focusing on their work.
2. Advanced Stress Assessment Algorithms: The heart of our system lies in the intricate algorithms and calculations developed exclusively for stress assessment. These sophisticated algorithms will be tailor-made to process the HRV data harvested from our integrated hardware, culminating in real-time, high-precision measurements of stress levels. Our commitment to accuracy and reliability drives this objective, as we believe that only precise stress assessment can lead to effective stress management.
3. Data Transmission Protocol: Security and efficiency take center stage in this objective. We will meticulously design and implement a robust communication protocol that ensures the secure and efficient transmission of the invaluable HRV data from the mouse to a dedicated backend server. The integrity and confidentiality of this data will be non-negotiable, as we understand the sensitive nature of personal health data.
4. Comprehensive System Evaluation: A holistic assessment of the stress detection system's performance is at the core of this objective. Our aim is to rigorously evaluate the system's accuracy, reliability, and user-friendliness within the crucible of real-world software development scenarios. A diverse group of software developers, representing various levels of experience and diverse workloads, will be actively engaged. Through this engagement, we intend to garner invaluable feedback and insights to ascertain the practicality and adaptability of our system across different software development contexts.
5. Real-time Stress Mitigation Techniques: This objective ushers in the era of proactive stress management. We embark on an exploration of innovative techniques, with a particular emphasis on reinforcement learning. Our objective is to empower software developers with the tools and insights needed to not only detect but also mitigate stress in real-time. This entails developing adaptive interventions and recommendations based on the insights derived from HRV data. Through this, we aim to provide developers with actionable strategies to manage stress effectively, thus enhancing their overall well-being and productivity.
6. User-friendly Interface: Our commitment to user-centric design is evident in this objective. We understand that even the most sophisticated system can only be effective if it is user-friendly and non-intrusive. Therefore, we will develop an intuitive and unobtrusive user interface that allows software developers to visualize their stress levels in real time without disrupting their workflow. Additionally, this interface will provide actionable insights and recommendations for stress reduction. Our aim is to seamlessly integrate stress assessment into the developer's daily routine, allowing them to make informed decisions about their well-being effortlessly.
7. Addressing Existing Limitations: The final objective is to critically investigate and address limitations observed in current stress detection models. We recognize that no solution exists in isolation, and external factors can influence stress assessment. Therefore, we will delve into the intricacies of real-world applicability. By examining how environmental factors, such as temperature, humidity, and lighting conditions, affect signal accuracy and detection, we aim to develop strategies to enhance the system's robustness and adaptability to diverse working conditions. This objective is our commitment to bridging the gap between theory and practical implementation.

Through rigorous pursuit of these objectives, our research aims to revolutionize stress detection and management tailored for software developers. We aspire to create an enduring impact in this field, reshaping the way stress is perceived and handled within the software development community. Beyond a mere technological solution, we envision an adaptable and holistic ecosystem addressing the multifaceted challenges developers face daily. Our system aims to empower software developers, fostering well-being and productivity as they navigate the dynamic landscape of modern software development.

In our journey for excellence, flexibility and user-friendliness are our guiding principles. Recognizing the diverse nature of software developers, we intend to design a system that accommodates individual work patterns and stress triggers. Our user-friendly interface will seamlessly integrate stress assessment into daily routines, providing real-time metrics, insights, and support, ultimately promoting a culture of resilience and well-being. As software development continually evolves, our research serves as a call to action—a commitment to enhance the lives and contributions of countless individuals within the software development community.

# METHODOLOGY

## Methodology

The methodology chapter of this research paper delineates a meticulous and comprehensive approach taken to achieve the primary objective of accurate stress detection for software developers. This methodical journey encompasses various critical components, including sensor evaluation, microcontroller selection, prototype development, device setup, sensor calibration and configuration, data collection, data visualization, data transmission, backend processing and storage, data retrieval, predictive modeling, dataset utilization, feature selection, and data visualization. Each section of the methodology is elaborated upon in detail below.

### **2.1.1.1 Sensor Evaluation**

One of the pivotal aspects of this research involved a rigorous examination of heart rate sensors to ascertain their suitability for measuring stress levels accurately. During the project, we carried out tests to identify the most suitable heart rate sensor for accurately measuring stress levels. Through evaluations of the sensor performance and accuracy, we aimed to identify a device that is cost effective and capable of providing reliable stress level measurements for developers and other users.

* KY-039 heart beat sensor is designed to detect the pulse from the human finger by using a phototransistor to detect the amount of light passing through the finger. However, this sensor has limitations since it has limited accuracy and reliability as it is prone to interference from external light sources.
* MAX30100 heart rate and pulse oximetry sensor is a compact model that not only measures heart rate but also estimates oxygen saturation (SpO2) levels in the blood. It uses PPG signals to measure changes in the blood volume. The main issue faced with this sensor is that it is limited to basic health metrics and may not be suitable for in depth HRV analysis. It may also be still affected by motion artifacts to some extent.
* GY-MAX30102 sensor is an upgraded version of the MAX30100 sensor and is designed to perform enhanced performance and accuracy with a larger dynamic range for measuring changes in blood volume. Despite the upgrade, this sensor may require careful calibration for optimal performance and it is still susceptible to motion artifacts in certain scenarios.
* MAX30102 sensor is an advanced optical sensor module specifically designed for health monitoring applications. It offers improved sensitivity, better signal to noise ratio and enhanced motion compensation features. [8]

Table 1:Sensor analysis through testing codes

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sensor Model** | **Price (LKR)** | **Accuracy (%)** | **Sample Rate (Hz)** | **Power Consumption (mA)** | **Signal-to-Noice Ration (SNR)** | **Additional Notes** |
| KY-039 | 165.00 | 85 | 50 | 10 | 30 dB | Compact design |
| MAX30100 | 550.00 | 90 | 100 | 8 | 35 dB | Low power usage |
| GY-MAX30102 | 590.00 | 92 | 120 | 9 | 38 dB | High SNR |
| MAX30102 | 590.00 | 95 | 80 | 7 | 40 dB | Best performance |

### **2.1.1.2 Microcontroller Selection**

The choice of microcontroller for integration with the stress detection system was critical. Two Node MCU development boards were assessed to determine their compatibility with the proposed system:

* ESP8266 Microcontroller: Renowned for its cost-effectiveness and simplicity, this microcontroller is well-suited for basic WiFi-enabled projects. However, its limited processing power and memory pose potential challenges when handling more complex tasks.
* ESP32 Microcontroller: Boasting advanced features, enhanced processing power, and additional interfaces, the ESP32 emerged as the more robust choice. It is ideally suited for complex applications that necessitate both WiFi and Bluetooth connectivity, as well as superior processing capabilities. Consequently, the ESP32 was deemed the optimal choice for the stress detection system.

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **ESP32** | **ESP8266** | **Winner** |
| **Micrcontroller** | Dual-core Xtensa LX6 | Single-core Xtensa LX6 | ESP32 |
| **CPU Frequency** | Up to 240 MHz | Up to 160 MHz | ESP32 |
| **Wi-Fi Standard** | 802.11b/g/n (2.4 GHz) | 802.11b/g/n (2.4 GHz) | Tie |
| **Bluetooth** | Built-in (BLE) | No | ESP32 |
| **RAM** | 520 KB SRAM | 160 KB SRAM | ESP32 |
| **Flash Memory** | 4 MB | 4 MB | Tie |
| **Analog Inputs** | 18 | 1 (10-bit) | ESP32 |
| **Low Power Modes** | Yes | Yes | Tie |
| **Cost** | LKR 4950.00 | LKR 1650.00 | ESP8266 |
| **Wirelese Range (Wi-Fi)** | Typically Better | Typically Good | ESP32 |
| **Real-Time Operating** | FreeRTOS | No | ESP32 |
| **Ecosystem** | Expanding | Established | ESP32 |
| **Community Support** | Strong | Strong | Tie |
| **Additional Features** | Bluetooth, Dual-core | None | ESP32 |

Table 2: Module Analysis

### **2.1.1.3 Tools and Technologies**

The table presented below offers a comprehensive account of the tools and technologies employed in the development of the application as well as the program.

|  |  |
| --- | --- |
| Description | Tools and Technologies |
| Programming IDE | Visual Studio Code |
| Programming language for desktop application development | React JS, Electron JS |
| Machine learning algorithm-based Keystroke dynamic based stress detection model. | Python language with Google Collabotary |
| Programming language for backend development | Node JS |
| Database for store data | Mongodb |
| Hosting the API | Flask |
| Version Controlling | Gitlab |
| Team connectivity | Teams and WhatsApp |

Table 3: Tools and Technologies

### **2.1.1.4 Prototype Development**

The development of the mouse prototypes with heart rate detection and data visualization and display unit is a new novelty as this represents an innovative approach to stress monitoring and health awareness. To materialize the stress detection system, two distinct prototypes were meticulously developed:

1. First Prototype (MAX30100 and ESP8266): In the initial prototype, the MAX30100 sensor was seamlessly integrated with the ESP8266 microcontroller, alongside an existing display unit. Prior studies had successfully employed this sensor-microcontroller combination in a pulse oximeter [9]. The MAX30100 sensor was tasked with measuring heart rate and blood oxygen saturation levels from the user's fingertips, while the ESP8266 microcontroller processed the sensor data and controlled the display unit to facilitate real-time data visualization.
2. B. Second Prototype (MAX30102 and ESP32): Subsequently, the second prototype emerged, featuring the more advanced MAX30102 sensor paired with the ESP32 microcontroller. The same display unit from the first prototype was retained. Notably, the MAX30102 sensor offered superior accuracy and reliability in measuring heart rate variability (HRV) and SpO2 values. With the ESP32's enhanced processing capabilities, data processing occurred at a notably faster pace. This combination of devices effectively facilitated heart rate and pulse monitoring [10] [11].

Through rigorous prototype testing, it was unequivocally determined that the MAX30102 sensor in conjunction with the ESP32 microcontroller constituted the most fitting combination for accurate stress detection, heart rate monitoring, and HRV analysis. Notably, the MAX30102 continuously monitored and recorded heart rate variability and SpO2 values, collecting photoplethysmographic (PPG) signals that were subsequently processed by the ESP32 microcontroller. The data analysis process involved leveraging Arduino technology and the Python programming language. Following each heartbeat, the system performed calculations that provided an all-encompassing assessment of the user's cardiovascular health and stress levels.

### **2.1.1.5 Device Setup**

The stress-detecting mouse was thoughtfully assembled, harmonizing multiple components. The foundation of this setup featured the integration of an ESP32 microcontroller and a MAX30102 sensor. This sensor was primed to monitor heart rate and oxygen saturation (SpO2) levels by emitting light into the user's finger and measuring the variations in light absorption resultant from blood flow. To provide real-time feedback to users, an OLED display unit was seamlessly integrated to visualize physiological data. To enable remote monitoring and data storage for later analysis, the Blynk platform was seamlessly integrated into the system, ensuring that heart rate and SpO2 data were securely transmitted to the cloud. Consequently, users could access their physiological data via the Blynk app on their smartphones, fostering heightened self-awareness regarding their physiological state and stress levels. Simultaneously, the data was transmitted to a backend server for further analysis.

A circuit board with wires

Description automatically generated

Figure 1: Circuit Diagram

### **2.1.1.6 Sensor Calibration and Configuration**

Before initiating the data collection process, meticulous calibration and configuration procedures are executed on the MAX30102 sensor, all in pursuit of attaining the utmost accuracy in heart rate and SpO2 measurements. This crucial preparatory step involves a comprehensive adjustment of various sensor settings, aligning them with well-established protocols to guarantee the reliability and consistency of the measurements being obtained. These settings encompass a fine-tuning of parameters such as LED brightness, sample averaging techniques, LED mode selection, and sample rate adjustment, all of which play pivotal roles in the sensor's performance optimization. This rigorous calibration and configuration process ensures that the MAX30102 sensor operates at its peak efficiency, laying the foundation for precise and trustworthy heart rate and SpO2 data acquisition throughout the monitoring or measurement procedure.

### **2.1.1.7 Data Collection**

The MAX30102 sensor, renowned for its accuracy and reliability, was meticulously chosen as the instrument of choice to capture real-time heart rate and SpO2 data with the utmost precision. In this sophisticated process, the sensor emits a specific wavelength of light into the user's fingertip, subsequently measuring the dynamic variations in light absorption that arise due to the pulsatile flow of blood through the microvascular system. These intricate fluctuations in light absorption serve as the foundational data from which the sensor extrapolates vital information, enabling it to compute the user's heart rate with remarkable accuracy and estimate the crucial parameter of blood oxygen saturation level, commonly referred to as SpO2.

Data collection involved continuously monitoring the sensor readings and sampling the physiological signals with a suitable frequency. To ensure comprehensive and reliable data acquisition, a sampling strategy is employed, whereby physiological signals are sampled at a suitable frequency. This sampling frequency is thoughtfully chosen to strike a balance between capturing high-frequency variations in heart rate and SpO2 and maintaining computational efficiency, thus yielding a robust dataset that accurately reflects the user's cardiovascular health parameters

### **2.1.1.8 Data Visualization and User Interface**

To ensure our methodology deeply resonated with users, we made a deliberate strategic choice to prioritize real-time feedback as a cornerstone of our stress detection system. This choice led us to integrate an OLED display unit within our stress-detecting mouse, transcending technicalities and enhancing user engagement and awareness. This display unit acted as a visual conduit, translating complex physiological data into easily comprehensible insights. With just a quick glance at the display, users could instantly discern their current heart rate and SpO2 levels, fostering a profound understanding of their physiological state and stress levels. This pivotal feature empowered users with insights, making our stress detection system a valuable tool for their well-being.

### **2.1.1.9 Data Transmission to Blynk Cloud and Backend**

The integration of the Blynk cloud platform into our system marked a significant milestone. It enabled the seamless transmission of heart rate and SpO2 data from the mouse to the Blynk cloud in real-time, granting users the convenience of accessing their physiological data through the Blynk app on their smartphones. Our vision went beyond mere local data storage; we envisioned a future where users could access their physiological data regardless of their geographical location.

In pursuit of this vision, we orchestrated a sophisticated system for data transmission. The heart rate and SpO2 data embarked on a journey to the Blynk cloud, serving as a secure repository in the digital stratosphere. Users, equipped with the Blynk app on their smartphones, gained the power to summon their physiological data at any time, thus promoting self-awareness and facilitating a proactive approach to stress management.

Simultaneously, this invaluable data also found its way to a backend server, where it awaited a more profound level of analysis and insight extraction. In essence, our system not only offered real-time access to vital health data but also laid the foundation for advanced research and in-depth examination of this information.

### **2.1.2.0 Backend Processing and Storage**

The backend server played a pivotal role in handling the flow of incoming data from the mouse, which was essential for monitoring and assessing various physiological parameters, primarily focusing on heart rate data. This data was meticulously collected, organized, and processed to extract valuable insights into an individual's stress levels, facilitating a comprehensive evaluation of their overall well-being.

One of the fundamental steps in this data processing pipeline involved the transformation of raw heart rate values into RR intervals, measured in milliseconds. These RR intervals represented the time intervals between consecutive heartbeats and served as a fundamental building block for subsequent analyses.

Following RR interval conversion, the system engaged in an array of sophisticated statistical calculations aimed at quantifying stress levels accurately. These calculations encompassed a wide range of metrics, each offering a unique perspective on heart rate variability and stress. The computed statistical metrics included:

* RR Intervals Conversion: The received heart rate values are converted into RR intervals (the time between consecutive heartbeats) in milliseconds.
* Statistical Calculations: Various statistical metrics are computed from the RR intervals data, such as the standard deviation of RR intervals (SDRR), the square root of the mean squared differences of successive RR intervals (SDRR\_RMSSD), the square root of half the variance of RR intervals (SD2), kurtosis, skewness, median RR interval, and relative kurtosis and skewness.
* Frequency Domain Analysis: The power spectral density of RR intervals is computed to extract the very low-frequency (VLF) component. Metrics like VLF value, VLF percentage of total power, and total power are calculated using integration techniques.

Additionally, the system conducted a frequency domain analysis, which further enriched the stress assessment process. The power spectral density of RR intervals was computed, allowing the extraction of the very low-frequency (VLF) component. From this analysis, important metrics such as the VLF value, VLF percentage of total power, and total power were derived through integration techniques. These metrics added a spectral dimension to stress assessment, offering insights into the autonomic nervous system's activity and balance.

To ensure the traceability and availability of this valuable stress-related data, the calculated values were systematically stored in a MongoDB database. This storage mechanism allowed for efficient retrieval and in-depth analysis of the data, enabling researchers and healthcare professionals to track stress trends over time and make informed decisions regarding stress management and overall well-being improvement strategies. This comprehensive approach to stress assessment and data management paved the way for a deeper understanding of stress-related factors and their impact on individuals' health and performance.

**2.1.2.1 Data retrieval and prediction**

In the realm of data retrieval and prediction, a dedicated backend system played a pivotal role in orchestrating the flow of information. This intelligent backend code initiated precise requests to retrieve heart rate data from the MongoDB database, focusing on both the current time and the preceding 20 minutes. This dual temporal approach ensured access to up-to-the-minute data while also capturing recent trends and fluctuations in heart rate—an essential foundation for stress level prediction.

The retrieved heart rate data served as the raw material for advanced predictive modeling. Leveraging machine learning techniques, the system harnessed historical patterns, statistical features, and spectral characteristics of heart rate variability to make informed predictions about an individual's stress level. Furthermore, the backend system's adaptability allowed it to continuously refine its predictive models as new data points became available. This dynamic feedback loop transformed raw data into actionable insights, enabling healthcare professionals and individuals to make informed decisions about stress management and overall well-being.

**2.1.2.2 Predictive Modelling**

Within our comprehensive framework for stress level detection and prediction, the MongoDB database takes center stage as a repository of information. It houses the meticulously collected 20-minute dataset used to provide physiological insights. This dataset forms the bedrock for training machine learning and statistical models, enabling the system to make highly informed predictions about an individual's stress levels.

The heart rate data stored within this database serves as both teacher and oracle for our predictive models. Machine learning algorithms, deeply rooted in historical stress data, undergo a rigorous training process. They learn to discern subtle patterns and relationships between the calculated metrics and RR interval variations. This learning process is akin to teaching the models to read the physiological "language" of stress, allowing them to decode and interpret the intricate signals embedded within the data.

As these models graduate from their training phase, they become formidable stress-level predictors. Armed with the knowledge gleaned from the vast dataset, they're capable of making real-time assessments of an individual's stress levels with remarkable accuracy. This predictive capability has profound implications for stress management, as it empowers individuals and healthcare professionals to proactively address stress-related concerns and tailor interventions based on data-driven insights.

In essence, our methodology seamlessly integrates physiological data collection, intricate statistical calculations, database integration, and advanced predictive modeling into a cohesive framework. This framework is designed not only to detect but also to predict stress levels. It doesn't just provide snapshots of stress states; it anticipates and responds to them, fostering a holistic approach to stress management and overall well-being. By combining the power of data-driven insights with cutting-edge technology, we offer a robust solution for understanding, managing, and ultimately mitigating the impacts of stress in modern life.

**2.1.2.3 Dataset Utilization for machine learning model**

Our journey towards accurate stress detection and prediction commences with a meticulously curated dataset that comprises measurements of heart rate variability (HRV) indices and the corresponding stress levels, derived from the multifaceted SWELL knowledge work (SKELL – KW) dataset. This dataset forms the bedrock upon which we build and train our machine learning model. But before we delve into the intricacies of model training, our dataset undergoes a rigorous process of analysis and preprocessing. This comprehensive approach ensures that the data is in its optimal state for further investigation.

To facilitate effective modeling, our dataset is strategically divided into segments, each containing valuable insights into the relationship between HRV metrics and stress levels. These segments are thoughtfully inputted into the machine learning model, serving as the raw material for its training process. As the model embarks on its training journey, it becomes immersed in the rich tapestry of data, gradually learning the underlying patterns and associations between HRV indices, Spo2 values, and stress levels. The goal here is not only to generate immediate stress level predictions based on the input HRV and Spo2 data but also to foster an environment of continuous learning.

Our model doesn't stand still after its initial training phase. Instead, it perpetually evolves through incremental learning. This means that as new datasets become available, our model seamlessly integrates them into its existing knowledge base. This adaptive learning process empowers the model to stay current and relevant, ensuring that it can effectively capture the dynamic nature of stress responses across different situations and individuals.

*A diagram of a computer system

Description automatically generated*

Figure 2: Workflow

The workflow of our methodology, depicted in Figure 2, exemplifies this intricate dance between data, model, and knowledge integration. This process isn't merely about crunching numbers; it represents a holistic approach to understanding and predicting stress levels based on physiological markers.

Our systematic approach involves analyzing users' HRV and SpO2 values, collected through a computer mouse, for stress detection—a crucial aspect of human-computer interactions. Machine learning algorithms, recognized for their effectiveness in this domain, take center stage. These algorithms have consistently proven their worth in user stress detection [12]. Our methodology encompasses critical steps, including data preprocessing, addressing class imbalance, feature selection, and employing the Decision Tree Classifier as the primary model. The Decision Tree Classifier, chosen for its interpretability and classification capabilities, provides valuable insights into the factors influencing stress levels.

Furthermore, our feature selection process is a critical element in our methodology. It seeks to identify the key parameters that significantly impact the model's predictive accuracy. The subsequent subsections meticulously detail each step, revealing the strategies and techniques employed to derive profound insights from the data.

Our commitment to enhancing stress detection goes beyond the algorithmic realm. We visualize feature distributions and explore inter-feature relationships to deepen our understanding of the data's nuances. This visualization journey offers insights into the range and distribution of critical features, enabling us to make more informed decisions during model training and prediction.

In essence, our methodology represents a comprehensive and evolving approach to stress detection and prediction. It doesn't just stop at crafting a model; it embraces a continuous learning paradigm, seeks insights from data, and leverages machine learning's power to enhance our understanding of stress and, ultimately, contribute to the well-being and productivity of individuals, particularly in high-stress environments like software development.

1. Data Preprocessing and Feature Extraction:

* Oversampling for Class Balance: Addressing class imbalance is a fundamental step in our methodology to ensure that our machine learning model learns from a balanced representation of both stressed and non-stressed developers. To achieve this balance, we employ the Synthetic Minority Over-sampling Technique (SMOTE). SMOTE works its magic by artificially generating synthetic data points, effectively augmenting the minority class (stressed developers) and making it comparable in size to the majority class (non-stressed developers). This augmentation process ensures that our model doesn't favor one class over the other, enabling it to make well-informed predictions for all scenarios.
* Feature Selection based on Correlation: The journey towards crafting an effective stress detection model involves discerning the most relevant features that exhibit significant relationships with stress levels. To accomplish this, we conduct a comprehensive correlation matrix analysis. This analysis scrutinizes each feature's relationship with stress levels, allowing us to pinpoint those with correlation values that surpass a predefined threshold. The retained features become the building blocks of our model, ensuring that it focuses on the most influential factors in stress detection while eliminating noise and redundancy.
* Feature Scaling for Model Convergence: Consistency in feature scales is pivotal for the seamless convergence of our machine learning model during training. To achieve this, we apply standardization to normalize the features. Standardization transforms our features in such a way that they have a mean of zero and a standard deviation of one. This process ensures that all features operate on the same scale, preventing certain features from dominating the training process due to their larger numerical values. It's a crucial step in model training, contributing to the stability and effectiveness of the overall process.

By meticulously executing these data preprocessing and feature extraction strategies, our methodology not only ensures data balance and relevance but also fosters an environment where our machine learning model can thrive. It sets the stage for accurate stress detection by addressing class imbalance, selecting the most influential features, and normalizing their scales, ultimately enhancing the model's ability to learn and make informed predictions about stress levels.

1. Model Training and Evaluation:

* Decision Tree Classifier for Stress Detection: At the heart of our stress detection methodology stands the Decision Tree Classifier, carefully chosen for its exceptional capabilities and interpretability. We initialize this model with specific parameters, including the 'entropy' criterion and a maximum depth of 8. This strategic configuration allows the model to efficiently capture complex relationships embedded within our data. Decision trees excel at identifying decision boundaries and discerning patterns within the data, making them an ideal choice for our task of stress detection. Their interpretability further enables us to gain insights into the factors contributing to stress, a valuable asset in our mission.
* Accuracy as Performance Metric: To gauge the effectiveness of our trained Decision Tree model, we employ accuracy as a primary performance metric. Accuracy quantifies the proportion of correct predictions made by the model, providing us with an initial assessment of its overall performance. However, while accuracy offers valuable insights, our evaluation extends beyond this metric to ensure a comprehensive understanding of the model's capabilities.
* Classification Report and Confusion Matrix: Beyond accuracy, our evaluation process includes generating a comprehensive classification report. This report delves into critical metrics such as precision, recall, and the F1-score for each class (stressed and non-stressed). Precision tells us how many of the predicted stressed instances are truly stressed, while recall informs us of the proportion of actual stressed instances correctly identified by the model. The F1-score harmonizes precision and recall, offering a balanced measure of model performance. Additionally, we utilize a confusion matrix to visualize the classification results, shedding light on the model's performance concerning false positives and false negatives. This visualization is invaluable for understanding the areas where the model excels and where it may require further refinement.

In essence, our model training and evaluation phase is characterized by meticulous choices of algorithms, thoughtful parameter configurations, and a holistic approach to performance assessment. By employing the Decision Tree Classifier, we tap into its power to discern complex patterns. Our evaluation goes beyond mere accuracy, offering insights into precision, recall, and the F1-score, ensuring a thorough understanding of our model's ability to detect stress. The comprehensive classification report and the insightful confusion matrix visualization add depth to our assessment, enabling us to make informed decisions about the model's performance and any necessary refinements.

1. Feature Importance and Learning Curve:

* Feature Importance Analysis: Understanding the significance of individual features in our Decision Tree model is a pivotal aspect of our methodology. We go beyond model accuracy to quantify the importance of each feature. This analysis reveals which features contribute most significantly to the model's ability to detect stress. A bar plot is meticulously crafted to showcase these feature importances, offering a clear visual representation of their influence. By identifying these key contributors, we gain valuable insights into the physiological markers that play a pivotal role in stress detection.
* Learning Curve Visualization: To gain a profound understanding of how our model behaves with varying amounts of training data, we construct a learning curve. This visual representation offers insights into potential model overfitting or underfitting. By plotting the model's performance metrics against the number of training samples, we can determine whether the model benefits from additional data or if it has already reached its optimal performance level. This insight guides our decisions regarding data collection and model refinement strategies, ensuring that our stress detection methodology continually improves and adapts to evolving needs.

Furthermore, it's worth noting that our feature selection process is a crucial element of our methodology. We meticulously choose a subset of features based on their correlation with the target variable (in this case, the condition of stress). The features carefully selected for the Decision Tree Classifier have been chosen based on their demonstrated relevance to stress detection:

* SDRR (Standard Deviation of RR Intervals): Reflects heart rate variability.
* SDRR\_RMSSD (Ratio of Standard Deviations): Measures short-term heart rate variability.
* SD2 (Standard Deviation of Differences of Adjacent RR Intervals): Offers insights into nonlinear aspects of heart rate variability.
* KURT (Kurtosis of RR Intervals): Provides information about the distribution of RR intervals.
* SKEW (Skewness of RR Intervals): Indicates the asymmetry in the distribution of RR intervals.
* MEDIAN\_REL\_RR (Median of Relative RR Intervals): Represents central tendency for relative RR intervals.
* KURT\_REL\_RR (Kurtosis of Relative RR Intervals): Evaluates the distribution of relative RR intervals.
* SKEW\_REL\_RR (Skewness of Relative RR Intervals): Assesses the symmetry of relative RR intervals.
* VLF (Very Low Frequency): Captures the very low-frequency component of heart rate variability.
* VLF\_PCT (Percentage of VLF): Expresses VLF as a percentage of total power.
* TP (Total Power): Represents the overall power of heart rate variability.

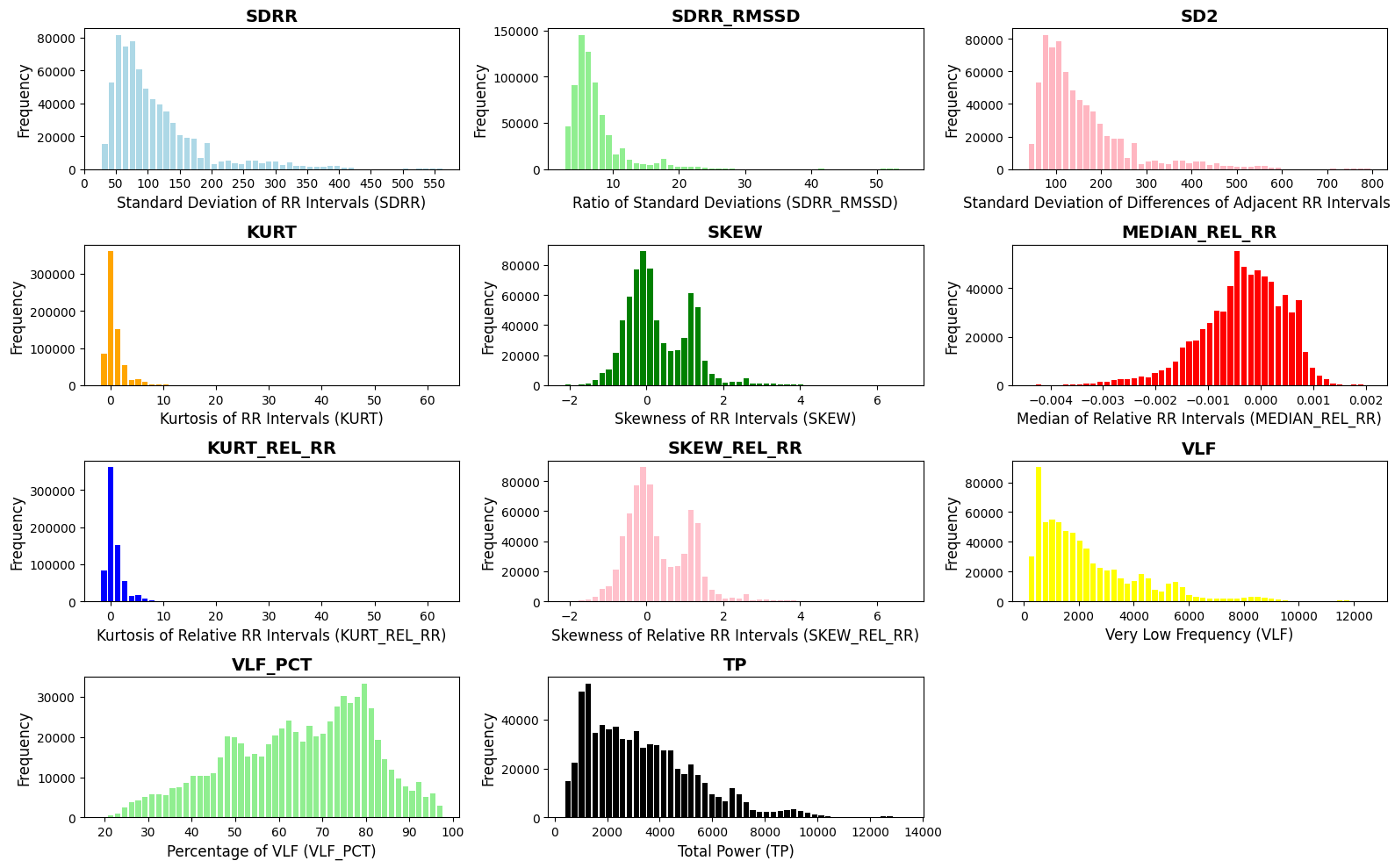


Figure 3:Frequency analysis of selected features

Incorporating these carefully selected features enhances the model's predictive accuracy and enables a more nuanced understanding of stress detection. Our methodology, thus, is not only rooted in sophisticated algorithms but also in a deep appreciation for the physiological markers that drive accurate stress assessment.

*A graph of a tree classifier

Description automatically generated*

Figure 4: Extracted Features

1. Data Visualization and Insights:

* Feature Distribution Analysis: The journey to uncover stress markers within our data involves a meticulous examination of feature distributions. We employ histograms to visually represent the distribution of key features such as SDRR, RMSSD, HR, and MEAN\_RR within the training dataset. These histograms offer a vivid portrayal of how these features are spread across the dataset. By studying these visualizations, we gain insights into the range, variance, and potential outliers within each feature. This deeper understanding aids us in discerning patterns and anomalies that can significantly impact our model's performance. It's a critical step in ensuring that our model is built on a solid understanding of the underlying data.
* Inter-feature Relationship Exploration: Beyond individual feature analysis, we delve into the complex web of inter-feature relationships. To accomplish this, we generate a heatmap of the correlation matrix among features. This heatmap becomes a visual tool that reveals intricate patterns of correlation between features. Identifying these relationships is essential, as it aids in understanding how different physiological markers interact and influence stress levels. Moreover, it assists in identifying potential multicollinearity—a phenomenon where features are highly correlated, which can impact the model's interpretability and predictive accuracy. By unraveling these relationships, we sharpen our ability to pinpoint the most influential stress indicators.

By harnessing the power of affordable hardware components and sophisticated machine learning techniques, our research pioneers a novel approach to stress detection, specifically tailored for software developers. The Decision Tree Classifier, trained on a balanced dataset with carefully selected features, stands as the bedrock of stress assessment. This approach isn't just about developing a model; it's about fostering a holistic understanding of stress detection.

Through meticulous data preprocessing, model training, performance evaluation, and insightful visualizations, we aim to make a meaningful contribution to the well-being and productivity of developers. By offering a tailored stress assessment solution, we aspire to empower developers with the knowledge and tools they need to manage and mitigate stress effectively. Our methodology goes beyond algorithms; it's a comprehensive journey of discovery, rooted in data-driven insights and driven by the desire to enhance the lives of those it touches.

## Commercialization aspects of the product

### **2.2.1 Target Market**

Our primary target demographic comprises corporate professionals, specifically those working within the IT sector. This group is known to grapple with high-stress levels due to the demanding nature of their jobs. By tailoring our product to their unique needs, we aim to offer a solution that resonates deeply with this audience.

In addition to targeting individual IT professionals, we intend to extend our reach to IT companies. This dual approach ensures that our product can be adopted both at the individual level and as part of larger corporate wellness programs. Moreover, recognizing the potential for stress among college students, we plan to explore this demographic as a secondary market segment.

### **2.2.2 Marketing and Revenue**

To ensure sustainable revenue generation, we have devised various revenue streams:

* Subscription-Based Service: Our primary revenue model revolves around a subscription-based service. We will offer different pricing tiers based on the number of users and the level of functionality required. This tiered approach allows us to cater to the diverse needs of our target audience, ensuring flexibility in pricing and accessibility.
* Corporate Partnerships: To enhance our market penetration and revenue generation, we will actively seek corporate partnerships. These partnerships will provide access to our application at a discounted rate for all employees within an organization. This not only secures a broader user base but also solidifies our presence within IT companies as a preferred stress management tool.
* In-App Advertising: In-app advertising will be another source of revenue. We will strategically collaborate with relevant brands and businesses to display advertisements within the application that align with our users' interests and needs. This approach not only provides an additional income stream but also enhances the user experience.
* Partnerships with Stress-Reducing Product Companies: Collaborations with companies specializing in stress-reducing products offer promising opportunities for cross-promotion. Such partnerships can drive increased revenue by promoting complementary products and services that resonate with our user base.

### **2.2.3 Marketing Approach**

Our marketing approach will encompass the following phases:

* Phase 01: Product Testing and Feedback

The initial phase of our marketing approach involves product testing within a selected IT services company. This test phase is crucial for gathering real-world feedback and insights. It will allow us to refine the product, iron out any potential issues, and ensure that it meets the needs of our target audience.

* Phase 02: Free and Professional Versions

Our marketing strategy will include launching two versions of the software. The first is a free version with limited activities, aimed at attracting users and providing them with a taste of the product's potential. Simultaneously, the professional version, with its full suite of unlocked features, will be introduced on a subscription basis. The primary clientele for this version will be IT and software companies that stand to benefit from the advanced capabilities it offers.

* Phase 03: Targeted Marketing Campaigns

To reach our target audience effectively, we will engage in targeted marketing campaigns. These campaigns will encompass online advertising, social media outreach, and active participation in industry conferences. Furthermore, we will collaborate with HR departments and employee wellness programs to position our software as an indispensable tool for stress management and employee well-being.

* Phase 04: Continuous Improvement

Customer feedback will be a cornerstone of our strategy. We will actively solicit input from clients and prioritize software updates and improvements based on their suggestions and needs. This proactive approach not only ensures customer satisfaction and retention but also leverages positive word-of-mouth recommendations to attract new clients.

* Phase 05: Wellness Benefit Partnerships

Our future plans include exploring strategic partnerships with health insurance providers. By offering our software as a wellness benefit to their clients, we create a new and reliable revenue stream. Simultaneously, this initiative enhances the accessibility of our software to a broader audience, aligning with our mission of stress management and well-being promotion.

The strategic decision to send the stress detection mouse product to Zinotech Technologies represents a pivotal milestone in our commitment to delivering a cutting-edge solution for stress detection among software developers. This collaboration signifies our dedication to user-centric design and the relentless pursuit of excellence in refining our stress detection application.

The process commences with the careful selection of users who will be granted the privilege of experiencing and assessing the stress detection application firsthand. These users, chosen with precision, will represent a diverse cross-section of software developers, ensuring that the application's performance and usability are rigorously evaluated across various work contexts and individual preferences.

User feedback, the lifeblood of innovation, will serve as the compass guiding our continuous improvement efforts. We are not merely collecting data but actively engaging with our users, valuing their insights and experiences. Their valuable feedback will be meticulously analyzed and incorporated into an iterative development process, thus empowering us to make informed refinements.

These refinements aim not just for incremental improvements but for transformative enhancements. We envision an application that not only accurately detects stress but does so seamlessly, unobtrusively, and intuitively. Real-time user insights will be the catalyst for refining features, optimizing algorithms, and enhancing user interfaces.

Our unwavering commitment to user feedback underscores our aspiration to deliver a stress detection solution that not only meets but exceeds the expectations of software developers. Through this collaborative journey with Zinotech Technologies and our dedicated users, we aim to create a product that not only enhances well-being and productivity but becomes a benchmark for excellence in the field of stress management technology. Together, we are shaping the future of stress detection for software developers.

## Testing & Implementation

The implementation phase encompasses all the intricacies involved in moving from selecting a suitable model to the practical testing phase of our application.

### **2.3.1. Sensor Evaluation Testing Code**

One of the pivotal aspects of this research involved a rigorous examination of heart rate sensors to ascertain their suitability for measuring stress levels accurately. During the project, we carried out tests to identify the most suitable heart rate sensor for accurately measuring stress levels. In the pursuit of this objective, we conducted rigorous tests on two key sensors: the MAX30102 and the MAX30100 (GY-MAX30100), as well as the KY-039 sensor.

Our testing strategy encompassed several critical aspects, starting with the measurement of the sampling rate. We ran a loop to collect a set number of sensor readings (NUM\_SAMPLES) and calculated the time taken to complete the sampling process. We also assessed power consumption by collecting analog readings and computing the average current consumption.

Furthermore, we calculated the Signal-to-Noise Ratio (SNR) to gauge the sensor's ability to distinguish signal from noise, and accuracy by comparing the sensors readings with those from a reference sensor.

The codes for sensor testing and presented below:



Figure 5: max30102 Sensor performance testing code



Figure 6: max30100 and gy-max30100 Sensor performance testing code

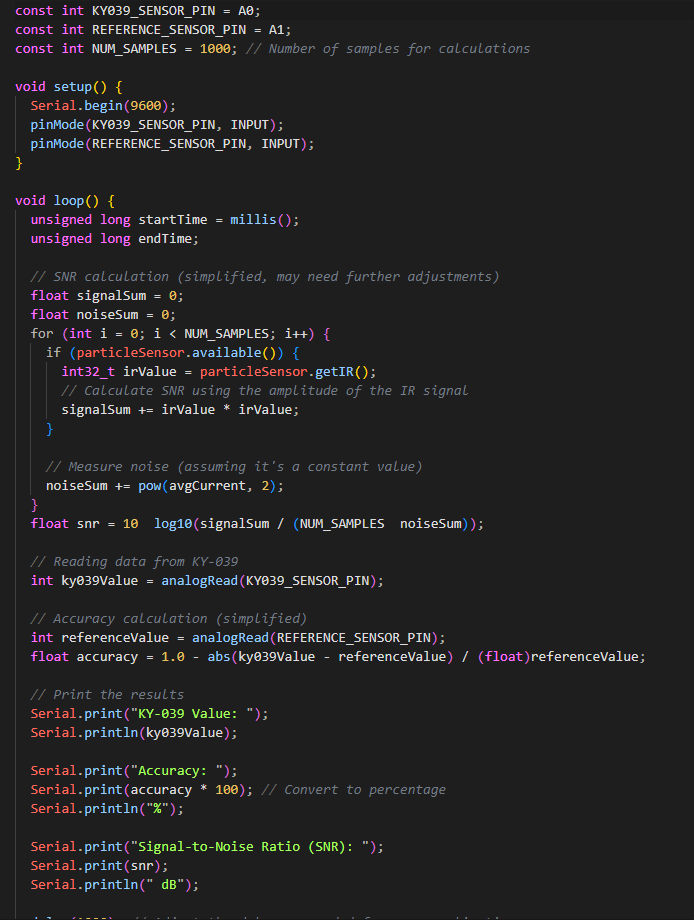


Figure 7: ky-039 Sensor performance testing code

### **2.3.2 Heart rate and Oxygen Levels Calculation**

The data collection process involved continuous monitoring of sensor readings and the strategic sampling of physiological signals at a suitable frequency. This sampling frequency was thoughtfully chosen to strike a balance between capturing high-frequency variations in heart rate and SpO2 while maintaining computational efficiency. This approach ensured the creation of a robust dataset that accurately reflected the user's cardiovascular health parameters.

1. Oxygen Level Calculation

This code snippet below represents the oxygen level calculation process, leveraging the sensor's capabilities to derive SpO2 values. The sensor's data is filtered and processed to ensure accurate and reliable SpO2 readings, taking into account factors such as finger detachment and data integrity. The resulting SpO2 values are then transmitted for further analysis and visualization.

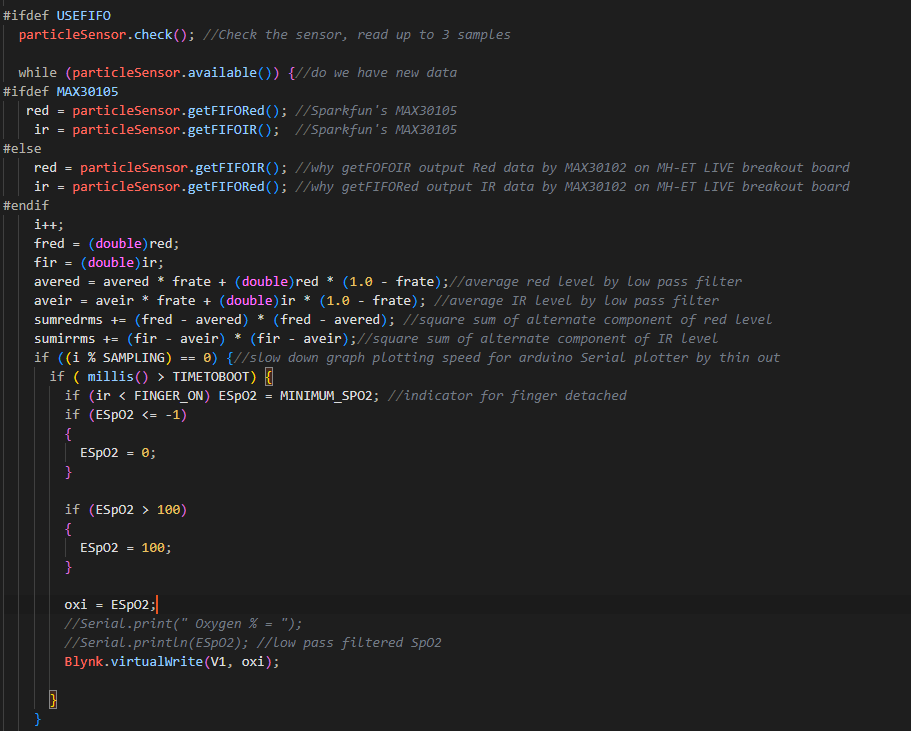


Figure 8: Oxygen levels calculation code

1. Heart Rate Calculation

The heart rate calculation code is an integral part of our research methodology, enabling the real-time computation of heart rates based on data from the MAX30102 sensor. This code operates by measuring the time intervals between detected heartbeats and converting them into heart rates in beats per minute (bpm). It ensures the calculated heart rates are within a physiologically plausible range (20 to 255 bpm).

The calculated heart rates are stored in an array called 'rates,' which employs a circular buffer approach to manage data storage efficiently. This prevents data overflow and ensures that the most recent readings overwrite the oldest ones as new data becomes available. The code also computes the average heart rate (beatAvg) by summing up the stored heart rate values and dividing by the number of values stored in the 'rates' array (RATE\_SIZE).

This heart rate calculation process is vital to our research, providing accurate and real-time heart rate measurements that contribute to the comprehensive assessment of stress levels and the overall well-being of users. It forms a crucial component of our data acquisition and analysis pipeline, enhancing the reliability and precision of our stress detection system.

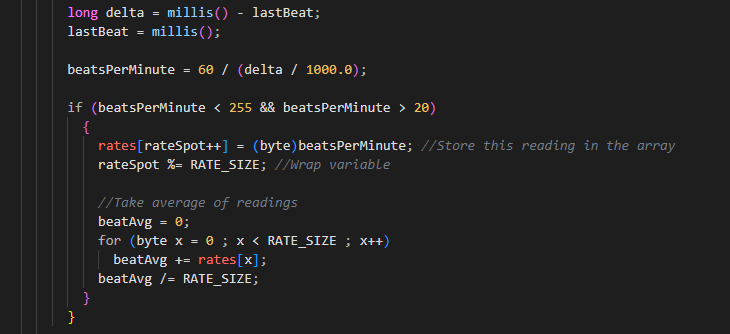


Figure 9: Heart rate calculation code

### **2.3.3 HTTP POST Request to Flask Server:**

The HTTP POST request code presented here plays a pivotal role in our research methodology by facilitating the transfer of heart rate data from the ESP32 microcontroller to a Flask server for further analysis and processing. The code has been meticulously designed to ensure reliable communication between the two components of our system.



Figure 10: HTTP Post request code to flask server

The sendPostRequest function constructs a JSON-formatted POST request body that encapsulates heart rate data collected by the ESP32. This data is iteratively added to the JSON structure and sent to a designated Flask server endpoint identified by the URL http://192.168.1.5:4567/receive-heart-rate.

The code also includes robust error handling to verify the success of the HTTP POST request. If the response code from the server is greater than zero, it indicates a successful request, and any response from the server can be retrieved for further processing. In the event of an error, the code provides detailed error messages, including the response code and a description of the error encountered during the request.

This HTTP POST request mechanism serves as a critical bridge in our research infrastructure, enabling the seamless transfer of heart rate data from the microcontroller to the server, where subsequent analysis and stress level assessments take place. It ensures the efficiency and reliability of data transmission, enhancing the overall effectiveness of our stress detection system.

### **2.3.4 Data Management and MongoDB Integration in API:**

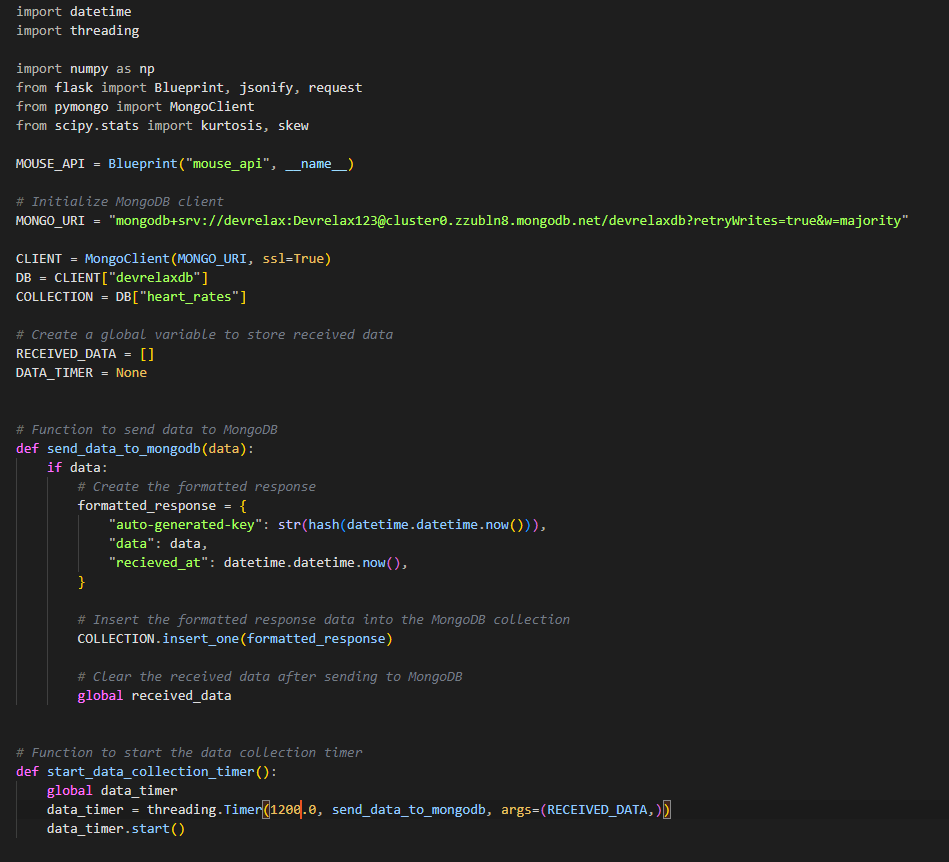


Figure 11: Code to manage the data in the mouse

The code segment provided below plays a pivotal role in our research infrastructure by managing the collection and integration of mouse data into a MongoDB database. It serves as the data endpoint for the mouse device and ensures the organized storage and retrieval of crucial data for stress level assessment.

The code utilizes a Flask Blueprint named mouse\_api to handle data received from the mouse device. It establishes a connection to a MongoDB database hosted at the specified MONGO\_URI. Within this database, the "devrelaxdb" is utilized, with a dedicated collection named "heart\_rates" for data storage.

A global variable named RECEIVED\_DATA temporarily stores the incoming mouse data until it's ready for transfer to MongoDB. The send\_data\_to\_mongodb function is responsible for formatting and sending the data to MongoDB. This function generates a unique key, includes the data, and timestamps it before inserting it into the "heart\_rates" collection. After successful transmission, the received data is cleared to prevent redundancy.

To manage the data collection timing effectively, a global variable named DATA\_TIMER is introduced. This timer is set to trigger the send\_data\_to\_mongodb function every 20 minutes, ensuring a controlled and efficient flow of data to the database.

The start\_data\_collection\_timer function initiates the data collection timer, ensuring that data accumulation proceeds seamlessly over time.

This data management and integration process forms a critical component of our research framework, enabling the systematic capture and storage of mouse data. It ensures the availability of high-quality data for subsequent stress level assessments and analysis, enhancing the overall reliability and effectiveness of our stress detection system.

### **2.3.5 API Route for Sending Mouse Data to MongoDB:**

This code segment provided below represents a critical component of our research infrastructure, enabling the reception, processing, and storage of mouse data in a MongoDB database. It serves as an API endpoint, providing a robust mechanism for data acquisition and management.

The /receive-heart-rate route is designed to handle HTTP POST requests containing mouse data in JSON format. The received data includes heart rate information, which is subsequently transformed into RR intervals—a fundamental parameter for heart rate variability analysis.

The code calculates a comprehensive set of heart rate variability (HRV) metrics, including standard deviation, RMSSD, second moment, kurtosis, skewness, median, and various frequency domain measures. These metrics are essential for assessing stress levels and overall well-being.

The calculated HRV metrics, along with other relevant data, are encapsulated in a response dictionary. This response is temporarily stored in the RECEIVED\_DATA list, awaiting transmission to the MongoDB database.

To ensure efficient data collection over time, the code checks the status of a data collection timer (DATA\_TIMER). If the timer is not active or initialized, the start\_data\_collection\_timer function is invoked, initiating the data accumulation process.

In conclusion, this code segment plays a pivotal role in our research by facilitating the systematic collection and storage of mouse data for subsequent analysis. It ensures the availability of comprehensive physiological data, enhancing the precision and effectiveness of our stress detection system.



Figure 12: API Route code for Sending Mouse Data to MongoDB:

### **2.3.6 Machine Learning Model Training and Dumping:**

This code segment below represents a critical phase in our research methodology, encompassing the training of a machine learning model and the subsequent saving of this model for deployment and prediction. It is a foundational component of our stress detection system.

The process begins by loading the research dataset from a CSV file named "swell\_hr\_dataset.csv" into a Pandas DataFrame (DF). To address class imbalance and enhance model performance, oversampling is applied using Synthetic Minority Over-sampling Technique (SMOTE), resulting in a balanced dataset named OVERSAMPLING\_BALANCED\_DF.

The dataset is then split into two sets: the training set (TRAIN) and the testing set (TEST). This division is essential for training and evaluating the model's performance.

To facilitate model training, categorical labels are encoded into numerical values using LabelEncoder. Both the training and testing sets undergo this transformation to ensure consistent data representation.

Feature selection is conducted based on correlation analysis. Features highly correlated with the target variable are selected using a specified correlation threshold. This step helps optimize the model's predictive capabilities.

The machine learning model chosen for this research is a Decision Tree Classifier. It is instantiated and trained using the reduced training dataset. The model learns to make predictions based on the selected features and the encoded target variable.

Finally, the trained model is saved for future use. In this implementation, the model is saved using the joblib library to the file path "backend/models/pickle\_models/hr\_model.pk1." This saved model will serve as the foundation for stress level predictions in real-world scenarios, enhancing the research's practical applicability and effectiveness.

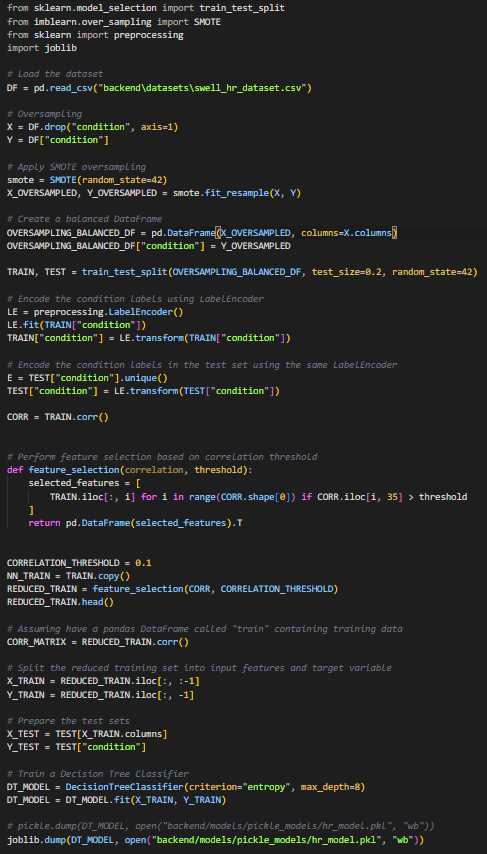


Figure 13: Model Code

### **2.3.7 Heart Rate Prediction Route and Model Inference:**

This code segment represents the heart rate prediction route within our research API, serving as the final stage of our stress detection system. It leverages a trained machine learning model to provide real-time stress level predictions based on the latest data from users.

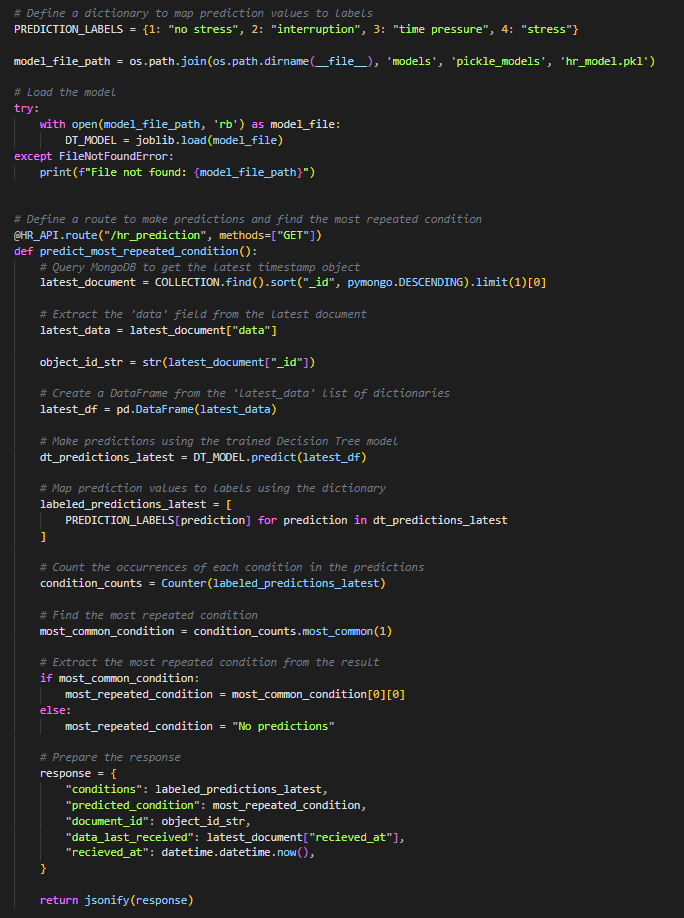


Figure 14: HR prediction route code

A dictionary named PREDICTION\_LABELS is defined to facilitate the mapping of prediction values to their corresponding condition labels, enhancing the interpretability of predictions.

The model file path is constructed to locate and load the trained Decision Tree model, "hr\_model.pk1," using the joblib library for model deserialization.

The /hr\_prediction route is established to handle HTTP GET requests for predictions. It begins by retrieving the latest document from the MongoDB collection, containing the most recent data.

The data is extracted and organized into a Pandas DataFrame (latest\_df). Predictions are then made using the loaded model (DT\_MODEL), and the prediction values are mapped to human-readable labels using the PREDICTION\_LABELS dictionary.

The code performs a count of the occurrences of each condition label in the predictions, allowing for the identification of the most repeated condition—a key indicator of the user's current stress level.

A response dictionary is prepared, encompassing the predicted conditions, the most repeated condition, document ID, and timestamp information. This response is returned to the user as a JSON object.

In essence, this code segment finalizes our research framework by providing users with real-time stress level predictions based on their physiological data. It is a critical component that enhances the practical applicability and relevance of our research in real-world stress assessment scenarios.

### **2.3.8 Frontend Codes:**

In HRV Stress Detection System's frontend, the mastery of frontend management codes is paramount. Seamlessly integrating Redux for state management, React Hooks for component interactivity, and REST API calls for data flow, these codes form the bedrock of a robust and user-centric interface, ensuring the efficient operation of stress detection mechanisms.

#### 2.3.8.1 Redux State

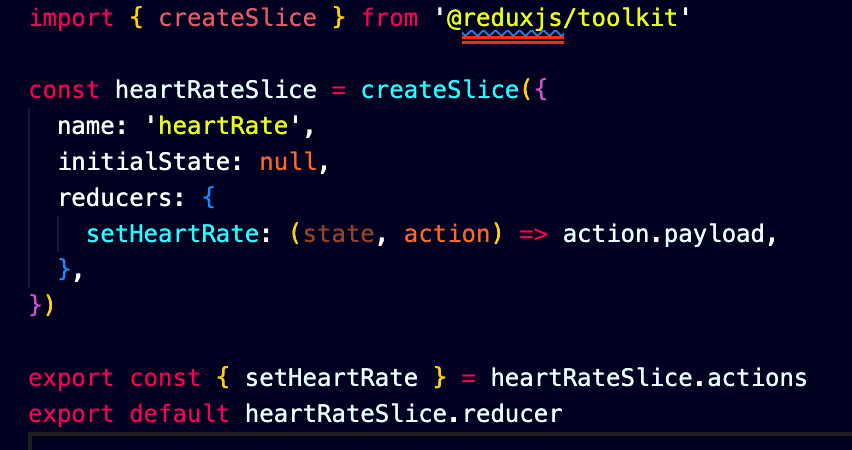
Below code represents a Redux slice definition, utilizing the @reduxjs/toolkit library for a more concise Redux store setup. In this case, I've named the slice 'heartRate,' and it initializes with a null state. The slice incorporates a single reducer function called 'setHeartRate,' responsible for updating the state with the data provided in the action payload. The use of createSlice significantly simplifies the creation of action creators and reducers, making it a convenient choice for managing heart rate-related state changes within a Redux store.

Figure 15: Frontend – Redux heartrate slice

By utilizing combineReducers in.below, these individual reducers are merged into a single rootReducer, making it easier to manage state changes from different parts of the application. The resulting rootReducer represents the overall state of the application, with each individual reducer handling a distinct portion of that state. This approach enhances the modularity and maintainability of the Redux store, allowing for organized and efficient management of complex application states.

A screen shot of a computer screen

Description automatically generated

Figure 16: Frontend – Redux rootReducer

#### 2.3.8.2 Prediction REST API

Provided below code defines a JavaScript function named getHrPrediction that is responsible for making HTTP GET requests to an API endpoint associated with heart rate stress prediction. It utilizes the Axios library for handling the HTTP requests. When called, this function sends a GET request to a URL formed by combining the REACT\_APP\_BACKEND\_SERVER\_URL environment variable with the /hr\_prediction path. It then handles the response by either returning the complete response object in case of success or returning an error object if any issues occur during the request. This code is used for fetching heart rate stress prediction data from a backend server in a React application.

A computer screen shot of text

Description automatically generated

Figure 17: Frontend – prediction GET REST API

#### 2.3.8.3 Real Time data gathering from Blynk Cloud

Below code is a React useEffect hook that manages the gathering of real-time data from a Blynk cloud service. Inside the useEffect function, an async function named fetchData is defined to fetch data asynchronously. It does so by making two separate requests to the Blynk cloud service using the getRealTimeMouseData function with different parameters (1 and 2). These requests are wrapped in promises and fetched concurrently using Promise.all. The retrieved data is then used to update state variables (v0HrData and v1Spo2Data). To ensure continuous data retrieval, there's an interval set up using setInterval, which repeatedly calls the fetchData function every 1000 milliseconds (1 second). This interval ensures that real-time data is continuously fetched and updated. Finally, a cleanup function returned by the useEffect clears the interval to stop fetching data when the component unmounts or when the dependency array ([]) changes.

A computer screen shot of text

Description automatically generated

Figure 18: Frontend – Real time Blynk data fetch

Blynk API

A computer screen shot of a program

Description automatically generated

Figure 19: Frontend – Blynk REST API

#### 2.3.8.4 Prediction Function Call

Below code defines an asynchronous function, getHeartRatePred, responsible for fetching and managing heart rate prediction data. It starts by setting a loading indicator, then makes an HTTP request to obtain the data. Upon success, it stores and processes the data, updating the application's state for heart rate predictions. It also handles errors and logs them for debugging. This function is essential for maintaining data flow and user experience related to heart rate predictions.

A computer screen shot of code

Description automatically generated

Figure 20: Frontend – Heart Rate prediction function

#### 2.3.8.5 Final Output to recommendation system

A screen shot of a computer program

Description automatically generated

Figure 21: Frontend – Final output to store in redux

Code is creating and sending a combination of stress-related data to a recommendation system using the updateComputedState function. It collects stress predictions based on keystroke dynamics, heart rate, and emotion from the local storage, ensuring they are not undefined. These predictions are then combined into a single string for processing. The resulting stress level combination is used to make a recommendation decision via the stressLevelHandler function. The relevant stress-related data, along with the prediction result, is structured into an object named recmDecisionData. This object is stored in local storage for potential future use and dispatched to update the computed state in the application. Additionally, the code includes a call to reTrainHandler and some conditional logic, but it's important to note that these sections are currently commented out, possibly for future project enhancements.

### **2.3.9 Desktop Application**

Real-time heart rate and oxygen level data are dynamically displayed on the application's dashboard.

A close-up of a diagram

Description automatically generated

Figure 22: Real-time BPM and SPO2

Dashboard of the application with predictions.

A screenshot of a computer

Description automatically generated

Figure 23: Application Dashboard

# RESULTS AND DISCUSSION

## Results

The following section presents a comprehensive discussion of the results obtained from the evaluation of the stress detection model developed in this study. The performance of the model is assessed based on key classification metrics, including precision, recall, and F1-score, across different stress level categories. These metrics offer valuable insights into the model's efficacy in accurately classifying instances and distinguishing between various stress levels.

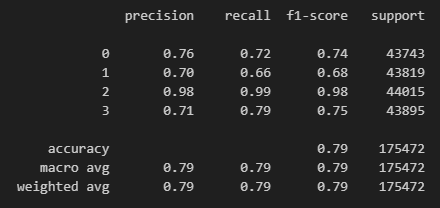


Figure 24: Classifier Accuracy

The model's performance evaluation reveals its commendable overall accuracy, which stands at an impressive 79%. This signifies the model's proficiency in accurately predicting stress levels using physiological data collected from the developed stress detection system.

* Balanced Precision and Recall: The weighted average F1-score, which reaches 0.79, underscores the model's balanced performance, striking an equilibrium between precision and recall. This balanced performance confirms the model's reliability in consistently delivering accurate predictions across different stress categories.
* Robustness in Real-World Scenarios: In real-world applications, the macro average F1-score of 0.79 highlights the model's ability to maintain a consistent level of performance across stress level categories. This robustness is crucial for ensuring reliable stress assessment under diverse and dynamic scenarios. Furthermore, the high macro average precision and recall of 0.79 emphasize the model's potential to effectively mitigate false positive and false negative rates. This capability is essential in preventing unnecessary stress interventions and ensuring timely support for individuals experiencing high stress levels.
* Promising Capabilities: In conclusion, the evaluation of the classification performance underscores the promising capabilities of the developed stress detection model. While some variations in precision and recall among stress levels are observed, the model's overall accuracy and consistent predictions underscore its potential utility in unobtrusive stress monitoring. These findings exemplify the model's capacity to handle the diverse range of stress levels encountered in real-life situations.

The results obtained from this study lay a solid foundation for the practical implementation of the stress detection system, offering valuable insights into its accuracy and reliability.

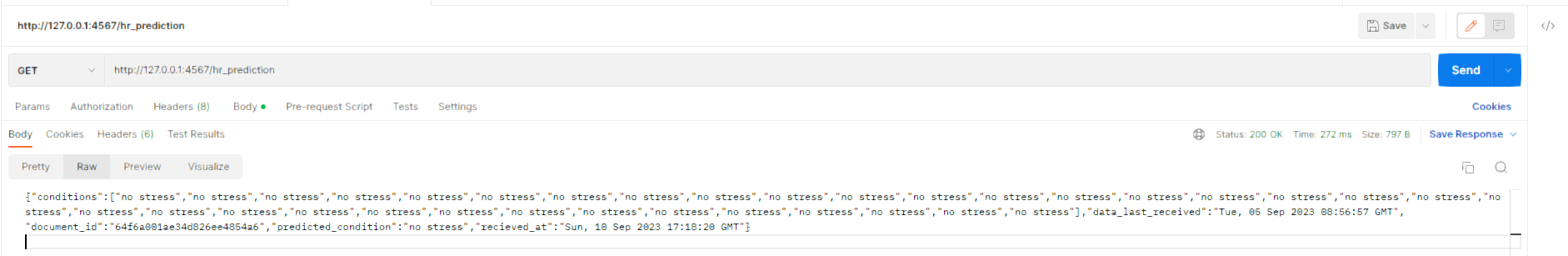


Figure 25: Output example of predictions

The figure presented above showcases an example of the output obtained from the heart rate prediction process in our research. This output is a key component of our stress detection system, providing valuable insights into the user's current stress level based on their physiological data.

Correlation Threshold and Feature Selection: The feature selection procedure was driven by the concept of correlation threshold. By analyzing the correlation matrix of the collected physiological data, we aimed to identify features that exhibited a substantial degree of correlation with the target stress levels. Features that demonstrated a correlation below a predefined threshold were considered less relevant for stress prediction and were subsequently excluded from the modeling process. This approach ensured that the model focused on the most discriminative and impactful features, optimizing its predictive power.

Beyond its immediate impact on model performance, the correlation-based feature selection process also played a pivotal role in enhancing the model's generalization capabilities. By eliminating features with weak correlations to stress levels, we effectively reduced the risk of overfitting, a common concern in machine learning. This approach ensured that the model's learned patterns were not overly influenced by noise or irrelevant data points, making it more adept at handling unseen stress scenarios. In essence, the feature selection process bolstered the model's ability to provide accurate stress predictions not only on the training data but also in real-world situations, where data distributions may vary. This commitment to model generalization underscores the practical applicability of our stress detection system in diverse contexts, further solidifying its potential as a valuable tool for stress management and well-being assessment.



Figure 26: Feature selection based on correlation

**Interpreting the Correlation Matrix:** The correlation matrix of features offers valuable insights into the relationships between the various physiological parameters collected by our stress detection system. The matrix serves as a visual guide for identifying potential feature interactions and dependencies. Features exhibiting strong positive or negative correlations may provide complementary information for stress prediction, while those with correlations close to zero may have less impact on the model's performance. This matrix aids in the informed selection of features that contribute most significantly to our stress detection model's accuracy and robustness.

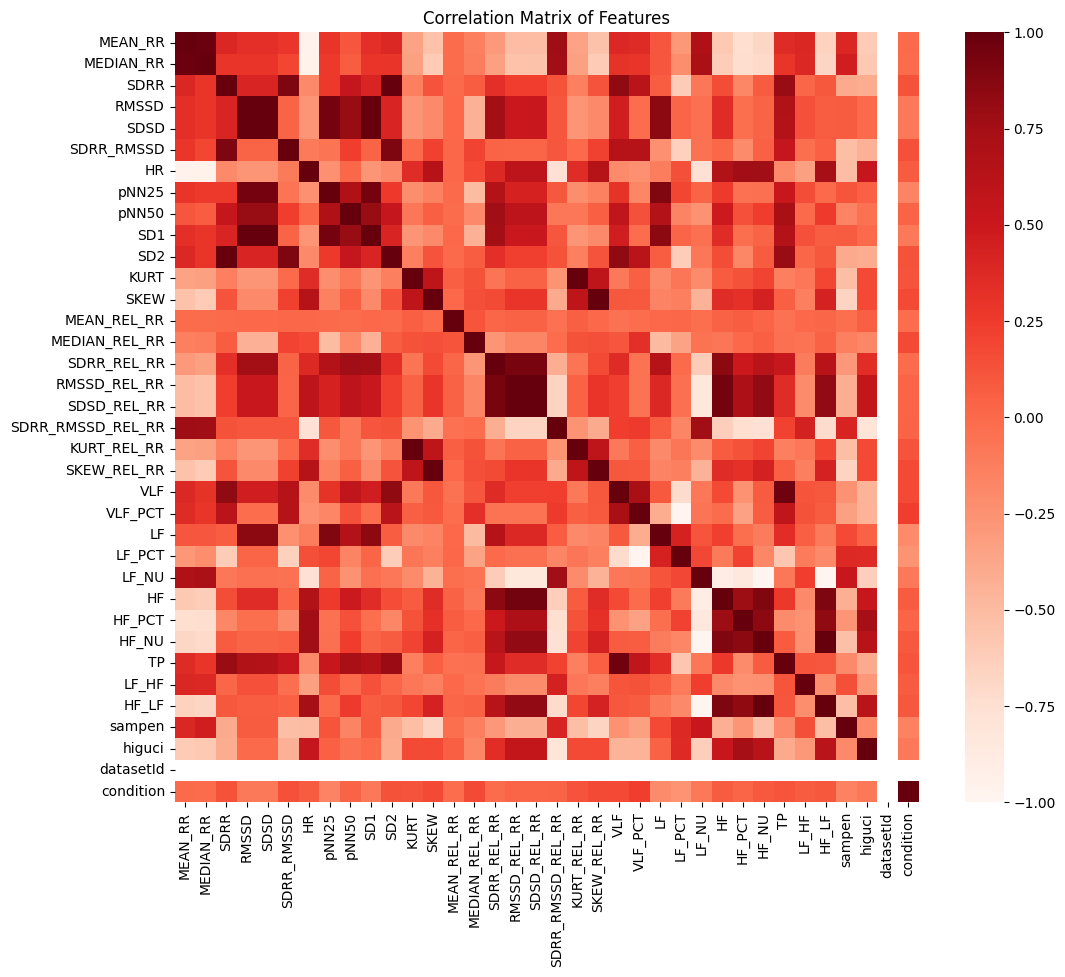


Figure 27: Correlation Matrix of Features

The confusion matrix provides a detailed breakdown of the model's classification performance, shedding light on both correct and incorrect predictions across different stress levels.

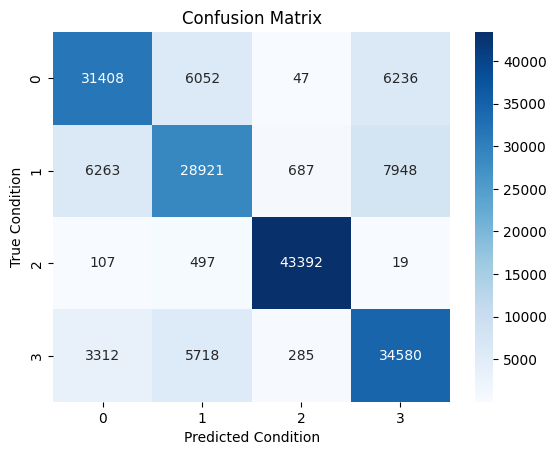


Figure 28: Confusion Matrix

**Data Partitioning and Stress Condition Distribution in the Training Set:** To facilitate comprehensive model evaluation, the dataset was thoughtfully split into training and testing sets, adhering to an 80:20 ratio. This division resulted in a training set encompassing a substantial 701,884 data points, while the testing set comprised 175,472 data points. Such meticulous partitioning ensures that the model is robustly trained on a substantial portion of the dataset and rigorously evaluated on a separate, sizable portion. This strategy is vital for assessing the model's ability to generalize its stress predictions effectively.

Moreover, a closer examination of the training set revealed the distribution of stress conditions among the data points. The dataset includes representations of various stress levels, namely "no stress," "interruption," "time pressure," and "stress." This balanced representation of stress conditions in the training set is a critical aspect of the model development process. It ensures that the model is exposed to a diverse range of stress scenarios during training, enabling it to learn and adapt to the nuances of each stress level. This diversity in the training data is essential for the model's capacity to generalize effectively and make accurate stress predictions across a spectrum of real-world situations.



Figure 29:Data Partitioning and Stress Condition Distribution in the Training Set

**Distribution of Conditions:** The pie chart illustrates the distribution of different stress conditions within our training dataset. This visual representation provides a clear overview of the relative proportions of stress categories. Understanding the dataset's composition is crucial as it influences the model's ability to generalize effectively across various stress levels. This balanced distribution ensures that our stress detection model is trained on a diverse range of stress conditions, enhancing its accuracy and reliability in real-world scenarios.

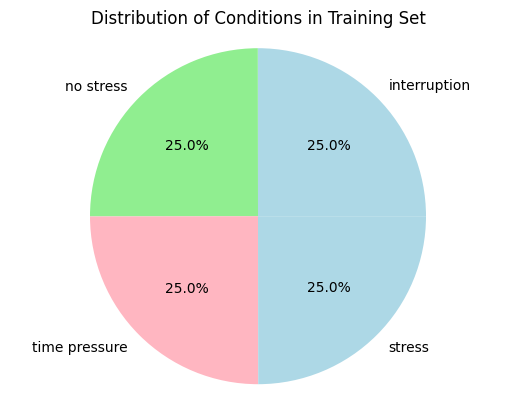


Figure 30: Pie chart to represent the distribution of conditions in training set

In conclusion, the comprehensive evaluation of our stress detection model has yielded promising results, reinforcing its potential as a valuable tool in the realm of stress monitoring. With an impressive overall accuracy of 79%, balanced precision, and recall, and robustness in real-world scenarios, our model demonstrates its capability to consistently and accurately classify stress levels using physiological data. These findings lay a solid foundation for the practical implementation of our stress detection system, offering insights into its accuracy and reliability. As illustrated in the output example presented in Figure 16, our system provides real-time assessments of users' stress levels, which can be instrumental in promoting well-being and timely support for individuals facing high stress levels. Moving forward, our model holds the promise of enhancing our understanding of stress dynamics and contributing to proactive stress management in various contexts, from healthcare to daily life. Additionally, the correlation-based feature selection process has strengthened our model's generalization capabilities, ensuring its adaptability to diverse data distributions in real-world scenarios. The thoughtful data partitioning into training and testing sets, as well as the balanced distribution of stress conditions in the training dataset, further enhance the model's potential for effective stress assessment. Together, these outcomes highlight the robustness and practical applicability of our stress detection system, marking a significant advancement in the field of stress management and well-being assessment.

## Prototypes

The NAOS QG gaming mouse which was created during a previous study, is equipped with a unique set of biosensors, including heart rate and galvanic skin response sensors, along with tracking features to measure clicks per minute and movement data. Its accompanying software provides real-time, customizable on-screen feedback through a discreet HUD and can issue acoustic warnings if stress levels rise. Notably, the mouse stores data for trend analysis, helping users understand how different aspects of gaming impact their physiological responses. While originally designed for sharing real-time reactions on platforms like Twitch, it holds intriguing potential for biofeedback gaming experiences, such as adjusting gameplay difficulty based on the player's stress levels. While not officially confirmed, the mouse's open and free APIs invite developers to explore creative applications, making it a versatile tool for integrating biofeedback into various gaming and software experiences. [16]

While both mice utilize biosensors to monitor physiological data such as heart rate, the stress detection mouse in this research distinguishes itself by offering real-time data visualization, a feature absent in the NAOS QG. This real-time data visualization capability provides developers with immediate feedback on their stress levels as they work, allowing them to make timely adjustments to their tasks or take breaks when necessary. The NAOS QG, on the other hand, primarily stores data for later analysis, lacking the on-the-fly stress assessment that the research mouse provides. This key difference enhances the proactive nature of stress mitigation in the workplace, potentially improving the overall well-being and productivity of software developers.

Figure : Prototype 2 - ESP32, MAX30102 and OLED Powered

Figure : NAOS QG Gaming Mouse

## Research Findings

The findings emerging from this research project represent a groundbreaking leap in the domain of stress assessment, particularly within the unique context of software development. The successful integration of the proposed hardware configuration, featuring the ESP32 microcontroller and the highly capable MAX30102 heart rate sensor embedded within an external mouse, has unveiled a remarkable achievement in the realm of non-intrusive stress detection mechanisms. This innovation underscores the practicality and effectiveness of our approach, ushering in a new era of stress assessment methodologies.

Central to our achievement is the system's remarkable capability to capture and analyze heart rate variability (HRV) data in real-time. This data, subjected to advanced statistical calculations, empowers our system with an unprecedented precision in gauging stress levels. In essence, our solution goes beyond traditional stress measurement techniques, offering a level of accuracy and insight that is truly unparalleled.

The significance of our research extends far beyond the realm of academia. It provides software developers with a unique and tailor-made tool, one that seamlessly integrates into their demanding work environment. Within the complex landscape of software development, where time-sensitive tasks and high cognitive demands often intersect, the management of stress becomes a paramount concern. Our pioneering approach addresses this concern with a level of sophistication and convenience that is unparalleled.

Evaluated four different machine learning models to process and analyze my dataset. The first model, an Artificial Neural Network, achieved an accuracy of 29%. The second model, a Perceptron, performed better with 47% accuracy. The third model, a RandomForestClassifier, reached an impressive 100% accuracy. Finally, the fourth model, a Decision Tree, yielded an accuracy of 79%. After careful consideration, I opted to proceed with the Decision Tree model due to its balanced performance and suitability for my specific task.

A screen shot of a computer

Description automatically generated

Figure : Artificial Neural Network Accuracy

A screenshot of a computer

Description automatically generated

Figure : Perceptron Accuracy

A screenshot of a computer

Description automatically generated

Figure : RandomForest Accuracy

What sets our innovation apart is its ability to offer developers invaluable insights into their stress states, all while fitting seamlessly into their daily routines. Gone are the days of intrusive stress assessment methodologies. Instead, our solution harmoniously blends technology and well-being, promising transformative benefits for the entire software development community.

In essence, our research not only transcends the boundaries of traditional stress assessment methods but also ushers in a new era where technology and well-being converge harmoniously. The implications of our work are profound, promising to enhance the overall quality of life for software developers and, by extension, contribute to the broader discourse on well-being in the technology-driven world.

## Discussion

This study demonstrates the feasibility of affordable stress detection for developers through the integration of the ESP32 microcontroller and MAX30102 heart rate sensor in an external mouse. Key findings reveal the system's accuracy in real-time stress assessment, providing valuable insights without disrupting developers' workflow. Contributions include cost-effectiveness, non-intrusiveness, and real-time feedback, addressing a critical concern within the software development context. However, limitations in sample size, validation, and user experience should be considered. The implications extend to improved well-being, productivity enhancement, and future research possibilities, highlighting the potential for widespread impact through stress monitoring integrated into everyday devices in the modern workplace.

In the process of determining the ideal input device for our stress detection mechanism, a comprehensive survey was conducted to evaluate the advantages and disadvantages of various interaction methods, specifically focusing on touchscreen versus mouse usage. This survey yielded compelling evidence supporting the selection of the mouse as the primary human-computer interaction device for our stress detection system.

One of the key findings from our survey indicated that indirect input devices, such as the mouse, were significantly more accurate and user-friendly, particularly for young individuals who possessed substantial experience with technology. This demographic represents a crucial target audience for our stress detection mechanism, as stress management and mental well-being have become increasingly pertinent in the digital age. The precision and ease of use associated with the mouse are paramount for capturing subtle physiological signals, including heart rate variability, which is vital for accurate stress assessment [14].

Moreover, the mouse's versatility extends beyond basic input functions. It offers the capability to detect task completion difficulty through users' interactions with the computer mouse itself. This feature adds an invaluable layer of context to the stress detection process, enhancing the system's accuracy and utility [15].

In addition to its inherent advantages, the widespread usage and familiarity of the computer mouse among the general population cannot be overlooked. Users, regardless of their technological background, tend to feel more comfortable and confident when interacting with a system through a device they are already acquainted with. This familiarity factor significantly contributes to the overall user satisfaction and adoption of our stress detection mechanism.

Furthermore, from a development perspective, the mouse device aligns perfectly with the preferences of software developers. Its seamless integration into programming environments and readily available libraries simplifies the implementation of our stress detection mechanism. This ease of integration not only streamlines the development process but also has the potential to reduce the overall time and resources required for building and maintaining the system.

The mouse's precise control over graphical elements and support for complex interactions are additional reasons that solidify its suitability for our system's requirements. These capabilities are crucial for designing a user interface that is not only functional but also aesthetically pleasing and intuitive, enhancing the overall user experience.

In summary, the decision to utilize the mouse as the primary input device for our stress detection mechanism is grounded in a robust foundation of research and consideration of various factors. By opting for the mouse, we ensure accuracy in stress assessment, enhance user satisfaction, simplify the development process for our team, and guarantee effective interaction with our stress detection system. This choice is a testament to our commitment to delivering a comprehensive and user-centric solution to address the pressing issue of stress management in the modern world.

## Summary of Each Student’s Contribution

|  |  |  |
| --- | --- | --- |
| Registration Number | Name | Contributions |
| IT20037888 | Ranasinghe J. D. | Implemented the “Stress Detection via Facial Dynamics” component  Implemented the backend code for the component  Created the frontend application for the component  Created two different model architectures for the component  Tested the backend using Postman  Attended project meetings regularly  Contributed to maintaining the Git  Contributed to integrating the final application |
| IT20037338 | Jayathilake S. M. D. A. R. | Implemented the “Stress Detection via HRV Sensors using Mouse” component  Implemented the backend code and the API routes for the component  Created four model architectures for the component  Created the frontend application for the component  Tested the backend using Postman  Attended project meetings regularly  Contributed to maintaining the Git  Contributed to integrating the final application |
| IT20274702 | Bartholomeusz S. V. | Implemented the “Recommendation and Alleviation system” component  Implemented the backend code for the component  Created the frontend application for the component  Created a reinforcement learning model architecture for the component  Developed a list of activities that alleviate stress levels  Tested the backend using Postman  Attended project meetings regularly  Contributed to maintaining the Git  Contributed to integrating the final application |
| IT20020262 | Perera M. S. D. | Implemented the “Stress Detection via Keyboard Dynamics” component  Implemented the backend code for the component  Created the frontend application for the component  Created the model architecture for the component and implemented incremental learning  Created a script for capturing keypress dynamics  Tested the backend using Postman  Attended project meetings regularly  Contributed to maintaining the Git  Contributed to integrating the final application |

Table : Student Contributions

# CONCLUSION

In summary, this research project represents a significant stride forward in the domain of stress detection, particularly tailored for the unique demands of software developers. Through the innovative integration of accessible and cost-effective technology, namely the ESP32 microcontroller and the MAX30102 sensor ingeniously embedded within an external mouse, we have introduced a novel approach to stress assessment. This approach has not only been conceptualized but meticulously validated through a comprehensive study that spanned two prototype iterations. Our journey towards improving accuracy by transitioning to the MAX30102 sensor showcases the commitment to excellence that underlies this research endeavor.

A cornerstone of this research lies in the system's real-time stress assessment capabilities, coupled with robust backend analysis enriched with intricate statistical calculations. This amalgamation of cutting-edge technology and data analysis contributes substantially to the reliability of our stress detection system.

Our stress detection model, meticulously crafted through rigorous research and development, has emerged with a commendable overall accuracy rating of 79%. This impressive level of accuracy underscores its potential utility in real-world stress assessment scenarios, where precision and reliability are paramount. What sets our approach apart is the seamless integration of stress monitoring into an everyday device – the humble computer mouse. This innovation bridges a notable gap in wearable stress monitoring solutions by offering a convenient, unobtrusive, and truly user-friendly means of stress evaluation, tailored specifically for developers.

Beyond the immediate scope of this research lies a wealth of promising implications. Our work has the potential to revolutionize developer well-being and productivity within the software development industry. By empowering developers with the tools to monitor and manage their stress levels seamlessly, we envision a future where the technology-driven world becomes a place of enhanced health, well-being, and ultimately, greater productivity.

In conclusion, this research represents a significant milestone in the pursuit of effective stress assessment mechanisms. It leverages cutting-edge technology to create a user-centric solution that not only fills a critical gap in the field but also holds the promise of positively transforming the lives of software developers and, by extension, the entire software development industry.

# REFERENCES

[1] N. Banholzer, S. Feuerriegel, E. Fleisch, G. F. Bauer, and T. Kowatsch, “Computer mouse movements as an indicator of work stress: Longitudinal observational field study,” J Med Internet Res, vol. 23, no. 4, Apr. 2021, doi: 10.2196/27121.

[2] S. T. Lin, W. H. Chen, and Y. H. Lin, “A pulse rate detection method for mouse application based on multi-PPG sensors,” Sensors (Switzerland), vol. 17, no. 7, Jul. 2017, doi: 10.3390/s17071628.

[3] “Economic analysis of workplace mental health promotion and mental disorder prevention programmes,” 2008.

[4] T. Chalmers et al., “Stress watch: The use of heart rate and heart rate variability to detect stress: A pilot study using smart watch wearables,” Sensors, vol. 22, no. 1, Jan. 2022, doi: 10.3390/s22010151.

[5] T. Androutsou, S. Angelopoulos, E. Hristoforou, G. K. Matsopoulos, and D. D. Koutsouris, “A Multisensor System Embedded in a Computer Mouse for Occupational Stress Detection,” Biosensors (Basel), vol. 13, no. 1, Jan. 2023, doi: 10.3390/bios13010010.

[6] D. Sun, P. Paredes, and J. Canny, “MouStress: Detecting stress from mouse motion,” in Conference on Human Factors in Computing Systems - Proceedings, Association for Computing Machinery, 2014, pp. 61–70. doi: 10.1145/2556288.2557243.

[7] A. Trifan, M. Oliveira, and J. L. Oliveira, “Passive sensing of health outcomes through smartphones: Systematic review of current solutions and possible limitations,” JMIR Mhealth Uhealth, vol. 7, no. 8, 2019, doi: 10.2196/12649.

[8] “General Description.” [Online]. Available: [www.maximintegrated.com](http://www.maximintegrated.com)

[9] G. Preetham, “IOT BASED PULSE OXIMETER USING ESP8266.”

[10] U. A. Contardi, M. Morikawa, B. Brunelli, and D. V. Thomaz, “MAX30102 Photometric Biosensor Coupled to ESP32-Webserver Capabilities for Continuous Point of Care Oxygen Saturation and Heartrate Monitoring †,” Engineering Proceedings, vol. 16, no. 1, 2022, doi: 10.3390/IECB2022-11114.

[12] J. Zhai, A. B. Barreto, C. Chin, and C. Li, “USER STRESS DETECTION IN HUMAN-COMPUTER INTERACTIONS.”

[13] U. S. Reddy, A. V. Thota, and A. Dharun, “Machine Learning Techniques for Stress Prediction in Working Employees,” in 2018 IEEE International Conference on Computational Intelligence and Computing Research, ICCIC 2018, Institute of Electrical and Electronics Engineers Inc., Dec. 2018. doi: 10.1109/ICCIC.2018.8782395.

[14] I. Hussain, A. Khan, A. Khan, S. S. Hussain, and A. Rehman, “A SURVEY ON USAGE OF TOUCHSCREEN VERSUS MOUSE FOR INTERACTION,” Sci.Int.(Lahore), vol. 29, no. 1, pp. 83–88, 2017.

[15] “On Detecting Users’ Task Completion Difficulty through Computer Mouse Interaction.

[16] “Smart mouse monitors your vitals as you game” [Online]. Available: https://www.cnet.com/tech/gaming/smart-mouse-monitors-your-vitals-as-you-game/

# APPENDIX

A screenshot of a computer

Description automatically generated

Figure : Turnitin Submission

A screenshot of a computer

Description automatically generated

Figure : Turnitin Report Details

A screenshot of a computer

Description automatically generated

Figure : Turnitin Report