ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΩΝ

ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

“ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ & ΥΠΗΡΕΣΙΕΣ”

Predicting Stress and Burnout Levels of Remote Employees Using Machine Learning and a Wearable Sensor Case Study

by

Dimitra Samara

Submitted

in partial fulfilment of the requirements for the degree of

Master of Information Systems and Services

Big data and analytics

at the

UNIVERSITY OF PIRAEUS

Piraeus, December 2025

Author: Dimitra Samara

MSc “Information Systems and Services” Piraeus, December 2025

Approved by the examination committee

. . . . . Ilias Maglogiannis (Professor)

. . . . . Andreas Menychtas (Assistant Professor)

. . . . . Konstantinos Delimpasis (Professor)

Table of Contents

Περίληψη 3
Abstract 4
1. Introduction 5
1.2 Objectives 5
1.3 Structure overview 6
2. Literature review 6
3. Methodology 9
4. Exploratory data analysis (EDA) 18
7.1 Data preprocessing 26
7.2 Metrics 27
7.3 Predictive models 28
7.4 High stress levels 31
7.5 Burnout 48
7.5.1 The most impactful features of burnout 53
8. Wearable Stress and Affect Detection (WESAD) dataset: a special case 55
8.1 Support Vector Machines 61
9. Conclusion 64
References 65

Table of Figures

Figure 1: Dialogflow's training phrases and intent handling 12
Figure 2: Dialogflow's basic flow for intent matching and responding to the end-user 12
Figure 3: Chatbot's welcome message that initiates the conversation 13
Figure 4: Chatbot UI to collect user information 14
Figure 5: Chatbot UI to gather user input about work related information 15
Figure 6: Chatbot questions about work related issues 16
Figure 7: Histogram with distribution of age 18
Figure 8: Distribution of stress level 19
Figure 9: Distribution of hours worked per week 20
Figure 10: Hours worked vs Stress levels 20
Figure 11: Satisfaction with work by years of experience 21
Figure 12: Correlation matrix 22
Figure 13: Average age by satisfaction level 23
Figure 14: Mental Health Condition bar plot 23
Figure 15: Work life balance rating chart 24
Figure 16: Industry distribution 25
Figure 17: Job role distribution 25
Figure 18: Code for one-hot encoding for Stress_level variable 27
Figure 19: Code for one-hot encoding for Mental_health_Condition variable 27
Figure 20: Random Forest Classifier overview 29
Figure 21: Random Forest Classifier working process 29
Figure 22: XGBoost overview 30

Figure 23: XGBoost algorithm detailed description 30
Figure 24: Code for Random Forest Classifier for stress prediction 32
Figure 25: Confusion matrix for Random Forest for stress prediction 33
Figure 26: Confusion matrix after SMOTE for stress levels prediction 34
Figure 27 Code for Random Forest Classifier for stress level prediction using SMOTE 35
Figure 28: Grid Search overview 36
Figure 29: Confusion matrix with GridSearchCV for stress level prediction 38
Figure 30: Code for Random Forest Classifier with grid search for stress level prediction 39
Figure 31: XGBoost confusion matrix for stress level prediction 41
Figure 32: Code for XGBoost for stress level prediction 42
Figure 33: Code for XGBoost for stress level prediction (part 2) 42
Figure 34: Random Forest and XGBoost accuracy comparison for stress level prediction 43
Figure 35: Accuracy comparison across all models for stress prediction 44
Figure 36: Precision comparison for low/medium stress level 45
Figure 37: Recall comparison for low/medium stress level 45
Figure 38: F1-score comparison for low/medium stress level 46
Figure 39: Precision comparison for high stress level 46
Figure 40: Recall comparison for high stress level 47
Figure 41: F1-score comparison for high stress level 47
Figure 42: Burnout prediction – Random Forest Confusion Matrix 49
Figure 43: Splitting into test and train set for burnout prediction 49
Figure 44: Random Forest Classifier code for burnout prediction 50
Figure 45: XGBoost Confusion Matrix for burnout prediction 51
Figure 46: Code implementation for XGBoost for burnout prediction (part 1) 52
Figure 47: Code implementation for XGBoost for burnout prediction (part 2) 52
Figure 48: Metrics comparison for burnout prediction 53
Figure 49: Features affecting burnout based on Random Forest 54
Figure 50: Features affecting burnout based on XGBoost 55
Figure 51: Window size and step initialization 57
Figure 52: Load and align signals 58
Figure 53: Feature extraction (part 1) 58
Figure 54: Feature extraction (part 2) 59
Figure 55: Code implementation for loso cross validation 59
Figure 56: Code implementation for classifiers and reports 60
Figure 57 Preprocessing pipeline 60
Figure 58: SVM overview 61
Figure 59: SVM algorithm description 61
Figure 60: Model Comparison— Accuracy 63
Figure 61: Model Comparison —F1-score 64

Περίληψη

Η παρούσα μελέτη στοχεύει στην παρουσίαση μιας λεπτομερούς ανάλυσης σχετικά με την ανίχνευση στρες χρησιμοποιώντας δεδομένα που συλλέγονται από chatbot καθώς και δεδομένα που συλλέγονται από wearables. Ο κύριος στόχος είναι η κατασκευή ενός chatbot για τη συλλογή δεδομένων που σχετίζονται με το στρες από τους εργαζομένους. Στη συνέχεια, χρησιμοποιούνται προγνωστικά μοντέλα όπως το Random Forest Classifier και το XGBoost για την ταξινόμηση των κλάσεων του στρες καθώς και για την πρόβλεψη της επαγγελματικής εξουθένωσης (burnout). Και οι δύο περιπτώσεις μετατρέπονται σε δυαδικά μοντέλα ταξινόμησης χρησιμοποιώντας τις κατάλληλες μεθόδους. Αυτά τα μοντέλα αξιολογούνται χρησιμοποιώντας τις καταλληλες μετρικές όπως accuracy, precision, F1-score και confusion matrix. Επιπλέον, χρησιμοποιείται ο αλγόριθμος SMOTE, ο οποίος είναι ένας αλγόριθμος για την παραγωγή συνθετικών δεδομένων, για την επίτευξη ισορροπίας κλάσεων και χρησιμοποιούνται μέθοδοι αναζήτησης για την απόκτηση των καλύτερων παραμέτρων για τα προγνωστικά μοντέλα. Επιπλέον, παρουσιάζονται λεπτομερώς τα πιο σημαντικά χαρακτηριστικά που είναι υπεύθυνα για την παρουσία επαγγελματικής εξουθένωσης. Όσον αφορά τα ψυχολογικά σήματα που συλλέγονται από wearables, χρησιμοποιούνται επίσηςγια την ταξινόμηση του στρες. Σε αυτήν την περίπτωση, τα δύο προγνωστικά μοντέλα που χρησιμοποιούνται είναι το Random Forest Classifier και το Support Vector Machines. Αυτά τα μοντέλα αξιολογούνται επίσης χρησιμοποιώντας τις ίδιες μετρικές που έχουν ήδη αναφερθεί.

Abstract

The current study aims to present a detailed analysis about stress detection using data gathered from chatbot as well as data collected from wearables. The main target is to build a chatbot in order to collect stress related data from employees. Then predictive models such as Random Forest Classifier and XGBoost are used for stress classification as well as for burnout prediction. Both cases are converted to binary classification models using suitable methods. These models are evaluated using robust metrics such as accuracy, precision, F1-score and confusion matrix. Moreover, SMOTE algorithm, which is an algorithm for producing synthetic data, is used in order to achieve class balance and Grid search parameters are used to get the best parameters for the predictive models. In addition to this, the most impactful features that are responsible for the presence of burnout are presented in detail. Regarding the psychological signals collected from wearables, they are used also to build a pipeline for stress classification. In this case, the two predictive models that are used are Random Forest Classifier and Support Vector Machines. These models are also evaluated using the same metrics already mentioned.

1. Introduction

Stress at workplace has become an increasingly significant concern in modern society. Undoubtedly, today's fast pace has affected people's psychology and specifically employees. Workplace stress has impact on employees' productivity, mental health, and overall job satisfaction. Chatbots have become a useful tool, to help collect data about employees' stress level at workplace, analyze and evaluate them. As artificial intelligence (AI) becomes more popular and powerful and chatbot technology is improving constantly, chatbots gives us the chance to collect real time data, analyze stress's effects and gain actionable and clear insights about employees' stress level , allowing us to make predictions about workplace future and take well-founded decisions. Moreover, it is beneficial to gain insights into workplace stress to recognize the symptoms and detect coping mechanisms. People who experience stress at their workplace might observe changes in their work quality, in their working methods and in their mood. This thesis investigates the potential of analyzing data collected by chatbots to predict stress patterns in the workplace.

While there is already data about workplace stress, they rely primarily on traditional means like surveys and manual data. These existing methods lack real-time and actionable data, meaning that measuring the stress level and making better predictions is still a challenge. This study aims to develop a chatbot to collect clear and real-time data that will contribute to understanding and making better predictions about workplace stress related trends.

In this study, there is also a brief presentation about stress data that was collected from wearables. Data collected from wearables can provide clear insights into people's stress levels as well as help to discover multimodal physiological stress patterns. Combining more than one way to gather useful data can undoubtedly lead to enhanced knowledge about what causes stress conditions to people in order to conclude to better decisions for prevention and treatment.

1.2 Objectives

The primary objective of this study is to create a predictive model for stress related conditions utilizing data gathered from a chatbot and wearables. The following research questions are specifically addressed in this study:

- Which model provides better classification about stress levels?
- Which model can make better predictions for employers who have experienced burnout during their work life?
- Which are the most important features that can lead on burnout at work?
- Which is the most suitable model for classification that is related to multimodal physiological stress patterns?

1.3 Structure overview

The meet the aforementioned aims, the study structure is the following:

- Chapter 1 Introduction
- Chapter 2 Literature review
- Chapter 3 Methodology
- Chapter 4 Exploratory data analysis (EDA)
- Chapter 5 Machine Learning Approaches to Predict Employee Stress and Burnout
- Chapter 6 Wearable Stress and Affect Detection (WESAD) dataset: a special case
- Chapter 7 Conclusion

2. Literature review

This chapter reviews the existing research on the use of chatbots in healthcare highlighting their applications, benefits and difficulties.

The first case is a chatbot called Viki, which has application on workplace mental health [1].The aim of this chatbot is to engage employees in a scalable, anonymous, and interactive manner, incorporating gamification elements to increase participation.

As far as the deployment process is concerned a platform, constructed using Ruby and JavaScript, has been selected. The UX team has designed a visually engaging interface. The UX has been designed by insights from focus groups and interviews to ensure consistency with user expectations. The checkup assessment is rule-based and the next steps determined by user responses. During the design phase, decision trees were established, and all potential user pathways were mapped and analyzed. Decision trees are suitable to provide the right responses and information based on the user's inputs.

The Viki chatbot uses gamification features (the use of game design elements in nongame contexts) such as levels, challenges, points, progress, feedback, story, and reward to keep a high engagement level among the participants. Furthermore, questionnaires are used as the Viki chatbot to communicate with the users through standardized questions so as to provide encouraging messages to sustain engagement. The duration of the assessment is about 15 minutes total. The responses from the users are stored automatically in a secure database. After the completion of the assessment, the participants receive personalized feedback and recommendations.

79% of the employees who participate in this survey complete the task of using the chatbot. This high response rate is better than traditional face-to-face methods and

exceeds typical online survey participation rates, highlighting how efficient this approach is.

Also, it is important to emphasize that the majority of the participants were male. Many employees reported that they have low scores for mental health symptoms, but there was also a significant proportion indicated experiencing mild to moderate levels of anxiety, depression, stress, and insomnia. Significantly, there was a high prevalence of work-related stress and the risk of burnout. Many employees mentioned having little control over high job demands, which is known to be a risk factor for burnout.

Higher prevalence rates of anxiety and depression symptoms were found among study participants when compared with general population data in Brazil, which may have been influenced by workplace-specific stressors. The study emphasized the role of departmental managers in promoting the assessment, which contributed to its high participation rate and positive reception among employees.

Overall, the study demonstrates the feasibility and potential benefits of utilizing chatbots for workplace mental health assessments. It provides scalability, anonymity, and high engagement rates compared to traditional face-to-face methods. This makes chatbot a useful tool for monitoring employers' well-being.

Another case is a chatbot named WisMoBot designed for stress management using Sense of Coherence (SOC) as a measure [2].

The framework is implemented across two robot platforms (NAO robot at home, Double 2 robot at the workplace) and a chatbot for use anywhere. A Belief-Desire-Intention (BDI) architecture is used to represent the internal state of the buddy persona within the framework. The user enters as input speech and text. Then the system interprets these inputs to make assumptions about the users' states. The objective of the framework is to enhance an individual's self-efficacy in managing stress, using the Sense of Coherence (SOC) model (Comprehensibility, Manageability, Meaningfulness).

From a technical aspect the chatbot is developed for mobiles using the LINE chatbot platform. The chatbot provides the users with the ability to interact with it via free text or selecting fixed response options. Data and information are stored in PostgreSQL.

The conversational model has a 5-phase scheme. The phases are the following:

1. Phase 1 begins a simple conversation with a greeting so as to engage the user.
2. Phase 2 of the conversational model involves measuring the Sense of Community (SOC) based on individual responses.
3. In phase 3 the SOC informs the Peer Support model, which decides the type of peer support.

4. In phase 4 the framework will offer assistance if the person agrees to the peer selection process.
5. In Phase 5, the user evaluates the efficacy of the support provided.

In the experiment have participated 12 adults, 2 males and 10 females, who interacted daily with the chatbot. The results show that even though half of the participants thought that the chatbot could detect stress effectively when they asked if they would like to use them in the future, the percentage of those who would like to use them was lower. Generally, most of the participants believed that communication and interaction with the chatbot is very simple. However, only roughly half of the users found the support suggestions helpful, indicating moderate levels of satisfaction. The reason for this is that the chatbot asked the same questions to the users every time, as some users mentioned. This caused users to get bored with the interaction. It was supposed to ask the same questions to those who prefer the same questions and ask different questions to those who don't, but it asked the same questions to everyone. Also, the chatbot's purpose is to measure stress-coping ability and while it gives advice to the users, they do not improve their ability. As a result, the individual's capacity to handle stress remains the same. Another reason for the refusal to use the chatbot in the future is that the experiment only provided questions and answers, so it didn't feel like a chatbot. Lack of personalities is also a problem. If the chatbots have personalities it would be easier for the users to have a conversation with them as well as this will improve their interactivity with the chatbot.

In conclusion, this study contributes to improving digital health interventions by providing a chatbot-driven solution for stress assessment and management based on SOC theory. It provides solutions to improve and support mental health in the workplace. Researchers and practitioners are encouraged to further investigate and enhance chatbot contributions so that the chatbot effectively supports stress management practices.

Lastly Vickybot is another case where a chatbot is used to detect anxiety and depressive symptoms [3]. The objective of this paper is to present the development, feasibility, and potential effectiveness of Vickybot, a chatbot designed for screening, monitoring, and reducing anxiety-depressive symptoms and work-related burnout, as well as detecting suicide risk among primary care patients and healthcare workers. It is developed within the broader PRESTO project.

The main functions of the Vickybot are the following:

1. The primary functions of Vickybot include self-administered scales for assessing anxiety, depressive symptoms, and work-related burnout. Users were prompted to complete all self-assessments every 2 weeks.

2. There were 12 short, sequential, and customizable psychological modules: 10 modules for anxiety and depressive symptoms (1: depressed mood, 2: anxiety, 3: apathy-anhedonia, 4: depressive cognitions, 5: suicidal thoughts, 6: restlessness, 7: decreased concentration, 8: overthinking, 9: irritability, and 10: sleep disturbance), and 2 modules for the management of the work-related stress and burnout. They mostly include cognitive behavioral therapy (CBT) techniques, mindfulness, dialectical behavioral therapy and acceptance and commitment therapy.
3. The chatbot system facilitated user-friendly navigation through the intervention, enabling access to various functions according to user scenarios and answering questions. Also, it can detect emergency situations, such as suicidal thoughts.
4. Vickybot also includes reminder functions. The chatbot identifies suicidal expressions through natural language processing and prompts an alert to the research team. The user is advised to immediately visit the emergency department and is provided with emergency resources.
5. Reminder for the “weekly objective” function from the modules: Users can select an hour of the day to perform the directed activity and have the option to create an automatic notification.
6. Audio register

The research involved primary care patients and healthcare workers, having a wide spectrum of backgrounds and occupational roles. Participants experienced varying severity levels of anxiety and depression, with a notable proportion experiencing severe symptoms. The study findings indicated that Vickybot was effective in screening anxiety, depressive symptoms, and work-related burnout among participants. Although follow-up assessments did not reveal statistically significant reductions in anxiety or depressive symptoms, a notable decrease in work-related burnout was observed, which affected over 60% of the sample. This underscores the potential of digital interventions such as Vickybot in mitigating occupational stressors and burnout. Chatbot also has the ability to accurately detect emergency situations such as suicidal thoughts. However, engagement metrics have low adherence and completion rates, indicating that improvement is necessary.

3. Methodology

To meet study's aim, quantitative research was conducted to examine and forecast workplace stress using data collected by the chatbot. The research contains the development of a chatbot, the collection and storage of users' data and the application of predictive analytics to recognize patterns and forecast trends. Specifically, the study consists of the following stages: (a) chatbot deployment and data collection (b) preprocessing of data and exploratory analysis, and (c) predictive modeling and evaluation.

As far as the chatbot implementation is concerned, it was developed using Python programming language and Google Dialogflow framework. Precisely, for the backend implementation of the chatbot, FastAPI, a modern, fast (high-performance), web framework for building APIs with Python was used. Moreover, Dialogflow is a natural language understanding platform that facilitates the design and integration of a conversational user interface within a web-based chatbot.

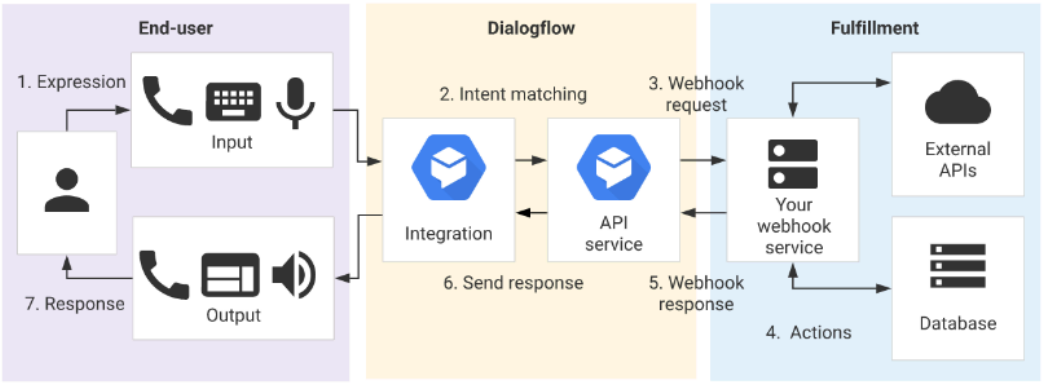

Figure 1: Dialogflow's architecture overview

To implement this chatbot, using Google Dialogflow platform, a virtual agent was created, who handles conversations with the users. Dialogflow can understand users' input and translate it to structured data, so that applications can interpret them to useful data and employ them accordingly. Every agent has many intents, that contributes to categorizing end user's inputs to handle the conversation efficiently. These intents constitute complete conversation. In order the chatbot to handle intents, it is necessary to define training phrases. Training phrases are example phrases of what users might enter as input. When user enters a phrase, which is the same or similar to the training phrases, the chatbot can extract the useful data and match them to a defined intent and display to the user the proper answer so as to continue the conversation. Dialogflow has built-in machine learning, which expands on the list, that is already defined, with other, similar phrases.

Figure 1: Dialogflow's training phrases and intent handling

When the user's input matches with an intent, Dialogflow extracts values from the end-user expression and structure them as parameters. The type of each parameter, known as the entity type, determines precisely how the data is extracted. In contrast to unprocessed end-user input, parameters represent structured data that can be readily utilized for logical operations or response generation. The responses that are returned to the users may be an answer, a prompt to give more information or a phrase that just terminates the conversation.

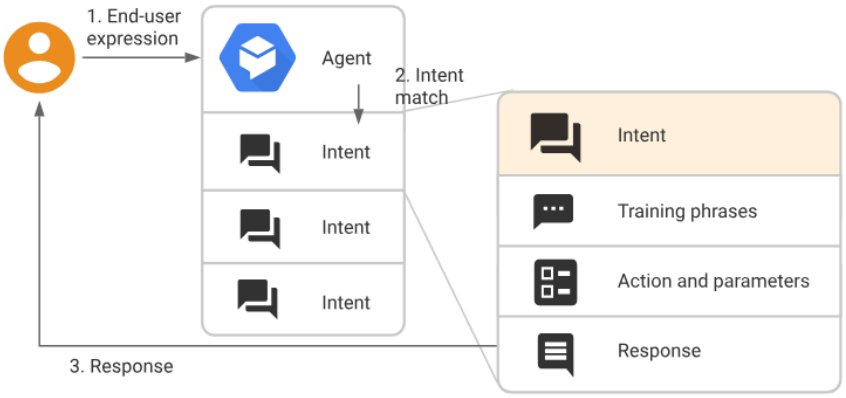

Figure 2: Dialogflow's basic flow for intent matching and responding to the end-user

The chatbot is web-based and can been accessed through a browser. The user interface (UI) has been implemented with HTML and the Google Dialogflow agent has been integrated inside the page.

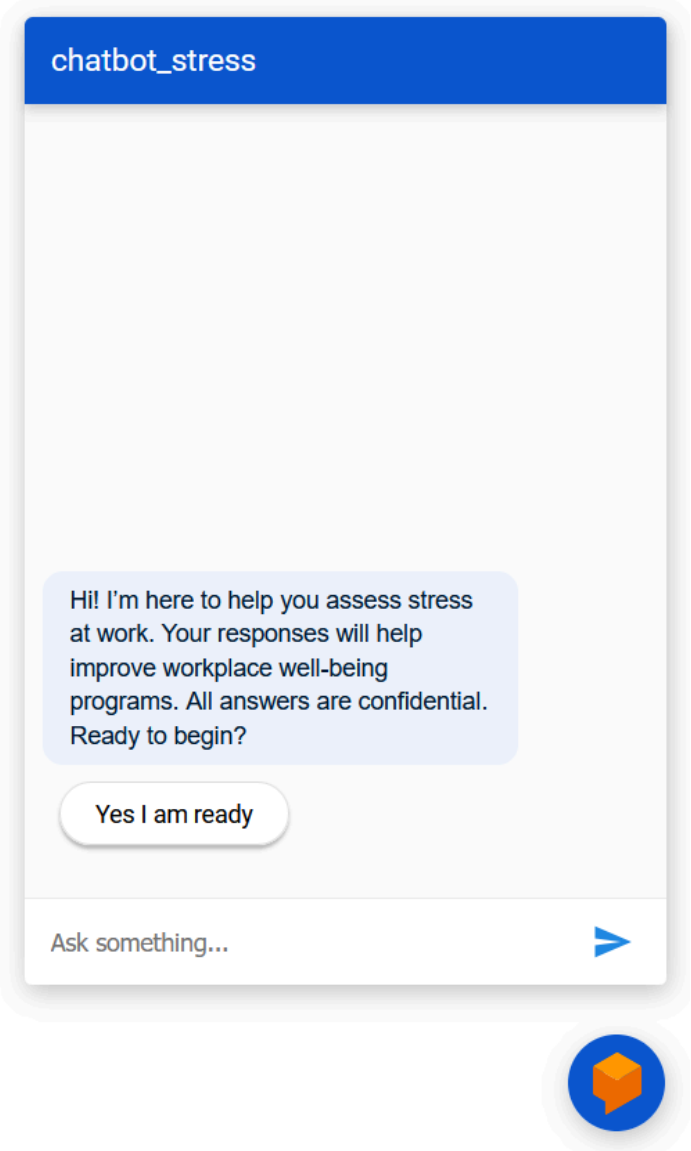

Figure 3: Chatbot's welcome message that initiates the conversation

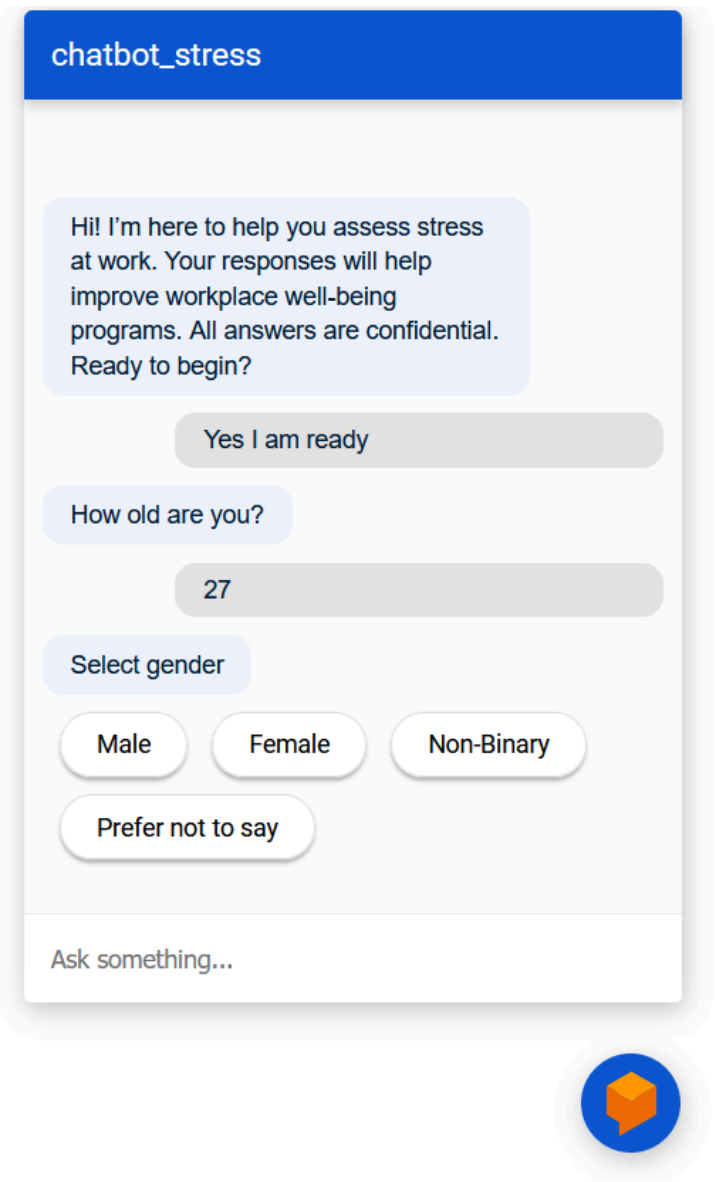

Figure 4: Chatbot UI to collect user information

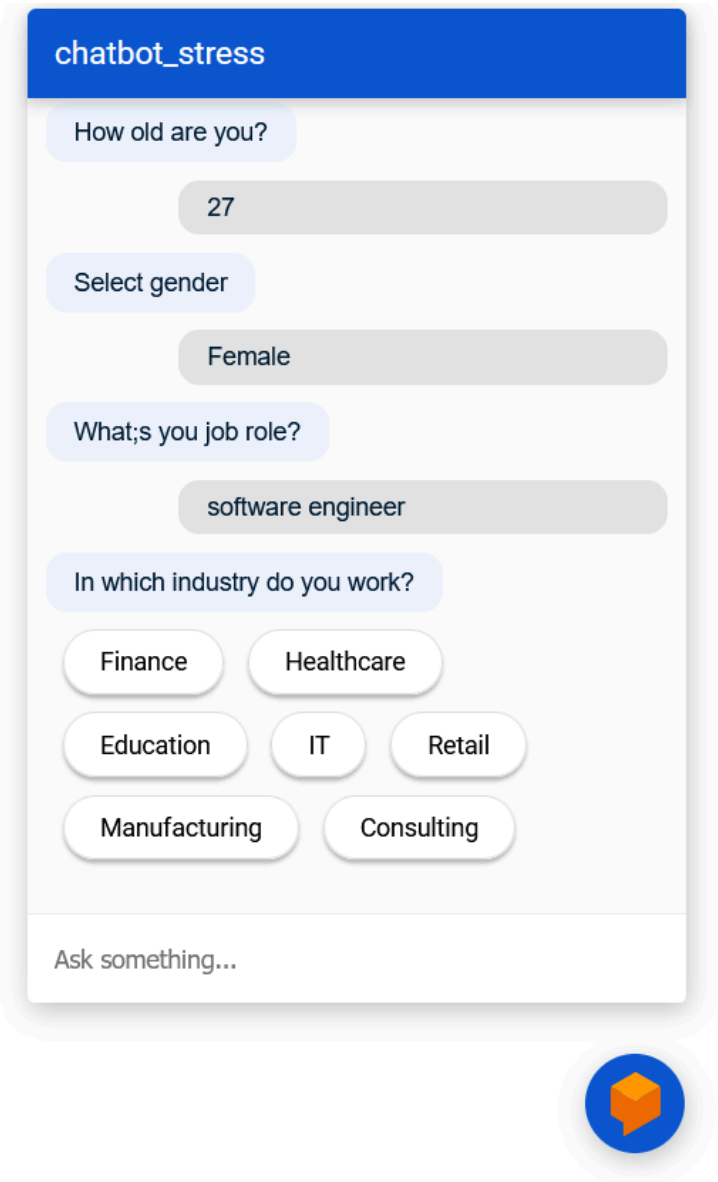

Figure 5: Chatbot UI to gather user input about work related information

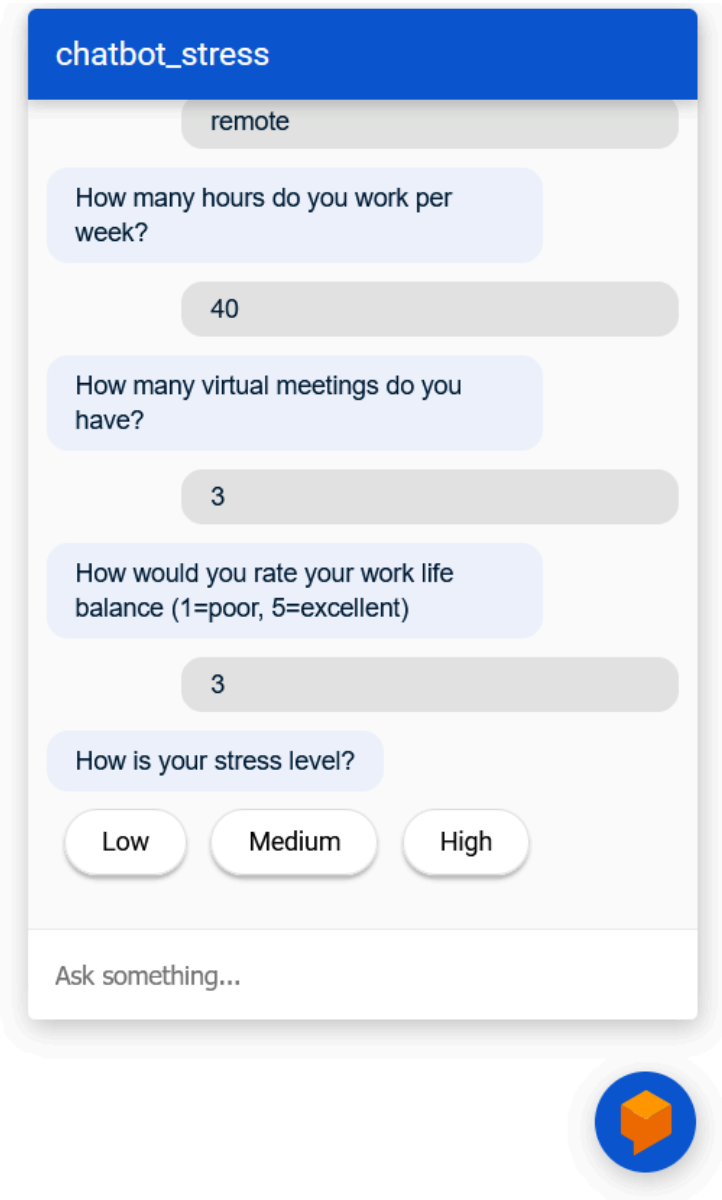

Figure 6: Chatbot questions about work related issues

The users' responses are stored in a PostgreSQL database, which is an open-source object-relational database management system (ORDBMS). All the collected data is stored anonymously. In addition to this, a NGINX has been used as a reverse proxy, to achieve communication between the client, namely the web browser and the FastAPI backend. NGINX is responsible for receiving the requests from the client and forward them to the FastAPI app and after the request processing it sends it back to the client. It enhances security since it doesn't exposure the backend server directly to the internet.

The software component of the application is installed inside Docker containers, for better scalability and easier management. A docker-compose tool is used to contribute to handling multiple containers, achieving better communication between them and configuring them easier.

The data include demographic information and stress related data. More precisely the data that I collected consists of the following information:

- age
- gender
- job role
- industry
- years of experience
- work location
- hours worked per week
- number of virtual meetings
- work life balance rating
- stress level
- mental health condition
- access to mental health resources
- productivity
- social isolation rating
- work satisfaction
- company support
- physical activity
- sleep quality

The above data are divided into two categories: numerical features and categorical features. Specifically, numerical features are the following:

- age
- years of experience
- hours worked per week
- number of virtual meetings
- work life balance rating
- social isolation rating
- company support

The rest are categorical data, namely:

- gender (male, female, non-binary, prefer not to say)
- job role
- industry (Finance, IT, Education, Healthcare, Manufacturing, Retail, Consulting)
- work location (Remote, Hybrid, Onsite
- stress level (Low, Medium, High)
- mental health condition (Depression, Burnout, Anxiety)
- access to mental health resources (Yes, No)
- productivity (Yes, No)
- work satisfaction (Unsatisfied, Satisfied, Neutral)
- physical activity (Daily, Weekly, None)

- sleep quality (Poor, Average, Good)

Before beginning the data exploration, so as to gain useful insights and discover trends data preprocessing and cleaning is necessary. The preprocessing steps included handling missing values, normalization, categorical encoding, text preprocessing, so as to ensure data is clean and ready to use them in the exploratory data analysis (EDA).

4. Exploratory data analysis (EDA)

Exploratory data analysis, or EDA, is the process of comprehending datasets by highlighting their key features, frequently through visual representation. This step is of high importance, especially when it comes to modeling the data for the application of machine learning techniques. Visualizations utilized in EDA include histograms, box plots, scatter plots, among others. Throughout the EDA process, it is essential to request to define the problem statement or definition on our dataset which is very significant. Moreover, it contributes to identifying hidden patterns and relationships between the data. Furthermore, errors or unusual data points (outliers) that could have an impact in our outcomes can easily be spotted. Insights gained from EDA can help us understand which features are most crucial for building models, selecting the best modeling techniques and how to set them up to enhance performance.

The dataset consists of 5000 records, namely it consists of chatbot interactions, as well as some additional data is used to enhance the gathered information. In the following paragraphs, a detailed explanation the exploratory data analysis's outcomes will be presented.

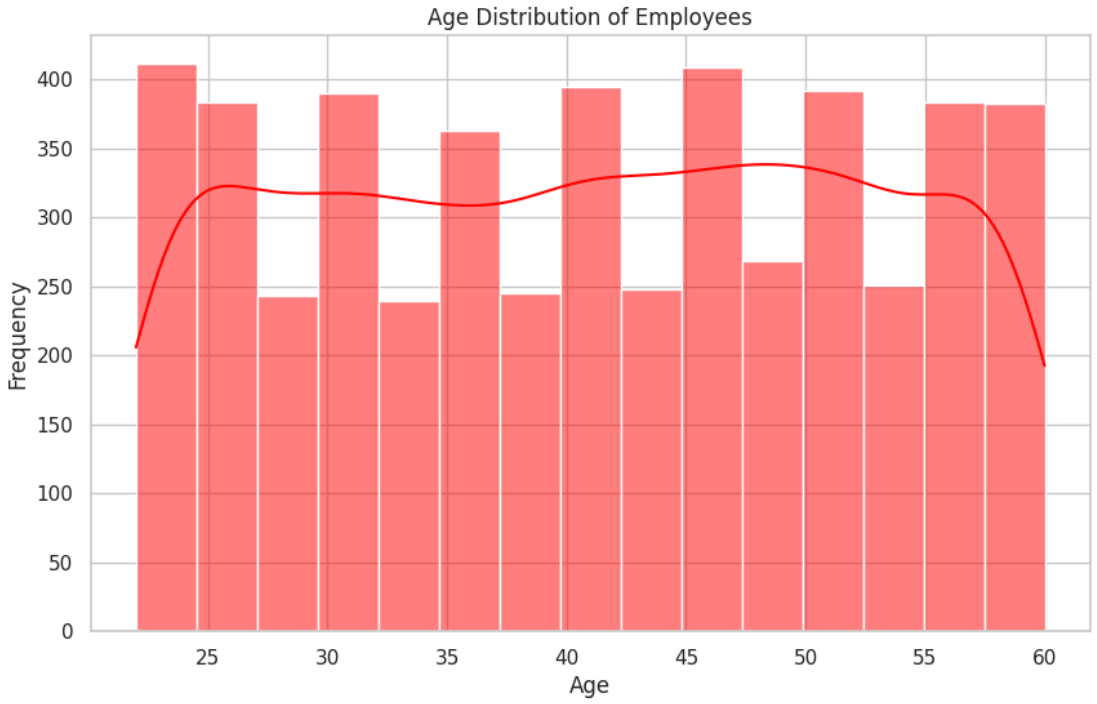

Figure 7: Histogram with distribution of age

In Figure 7 a histogram that visualizes the distribution of age in the dataset is presented. It is a histogram with a kernel density estimate (KDE) curve. On the x-axis, we can observe the age. The employees are evenly distributed across the ages 22 to 60. The frequency (vertical axis) indicates how many employees fall within each age group. The red line on the chart, namely the kernel density estimation (KDE) curve, indicates the estimated probability density function of the age. Given a set of data, Kernel density estimation is a technique that generates a smooth curve. As a sort of continuous substitute for the discrete histogram, this can be functional if you only want to see the "shape" of the distribution without relying only on the histogram. It is noticeable that the dataset includes a wide variety of ages. This ensures that a wide range of perspectives in the workplace are covered in this analysis.

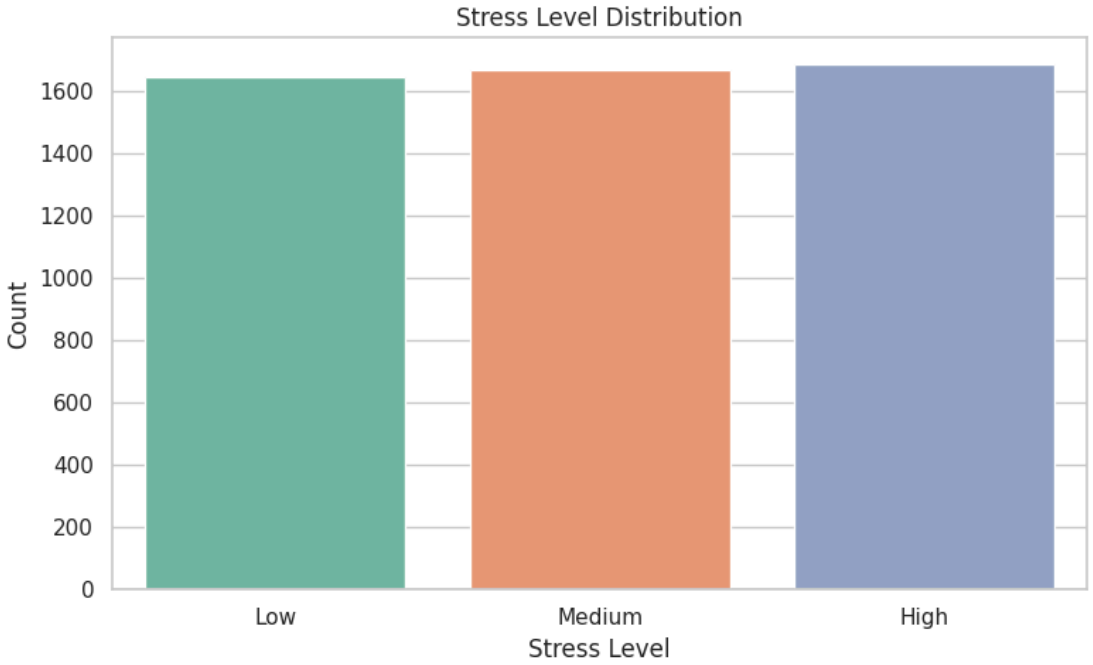

Figure 8: Distribution of stress level

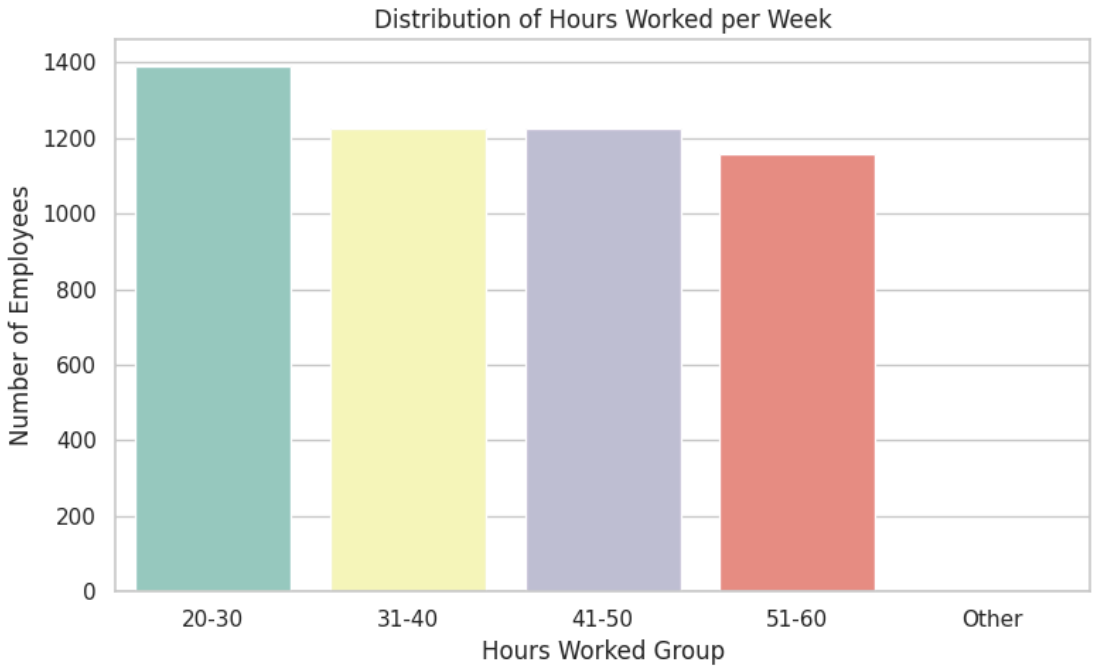

Figure 9: Distribution of hours worked per week

Figure 8 displays the distribution of stress level. It's obvious that there is an equal distribution between the three stress levels, namely low, medium, and high. Figure 9 illustrates the hours worked per week. The x-axis represents the hours worked per week and the y-axis indicates how many employees are in every hour worked group.

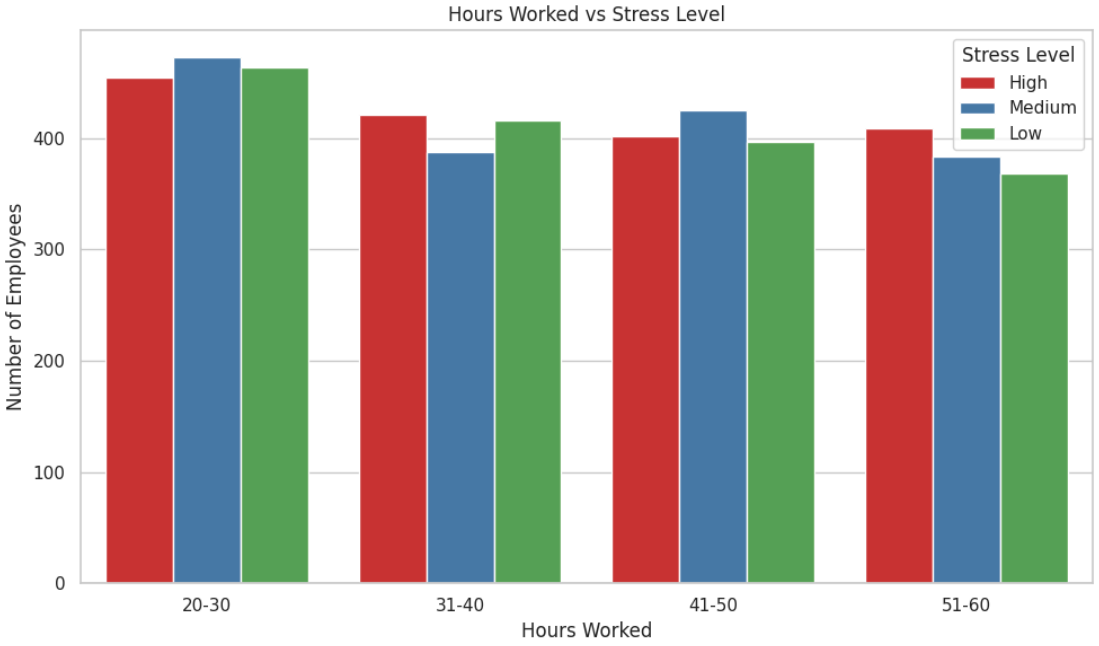

Figure 10: Hours worked vs Stress levels

Figure 10 is a grouped bar plot and presents how employees reported their stress levels, which fall into three categories, as mentioned above, low, medium and high.

The hours groups are 20-30, 31-40, 41-50 and 51-60 hours per week. In group 20-30 hours per week each stress level is reported almost the same with medium stress level to be slightly higher. In group 31-40 the most participants report high and low stress level. Medium stress increases in the 41–50 hour group. The total number of employees who work 51–60 hours decline, while high stress stays constant while medium and low stress—particularly low stress—decline. Generally, the results indicate that while high stress levels remain stable across all hours worked, medium stress is most frequently reported by participants who work 20–30 and 41–50 hours per week. Low stress decreases as hours worked increase, especially in the group 51-60 hours worked per week. These results indicate that longer work hours are strongly related to high stress levels. Considering the above, it is significant to monitor the workload in order to promote employees' mental health.

Figure 11: Satisfaction with work by years of experience

Figure 11 is a set of horizontal boxplots that compares employees' satisfaction with remote work (categorized as Unsatisfied, Satisfied, and Neutral) across different levels of professional experience. Each boxplot represents the distribution of years of experience for employees within each satisfaction category (Unsatisfied, Satisfied, Neutral). According to the boxplots, the range and median experience are basically the same for all the groups. This pattern indicates that the years of experience have little impact on how satisfied the employees with remote work are. This insight is essential for companies thinking about implementing remote work policies because it implies that employees of all levels may react to such policies in a similar way.

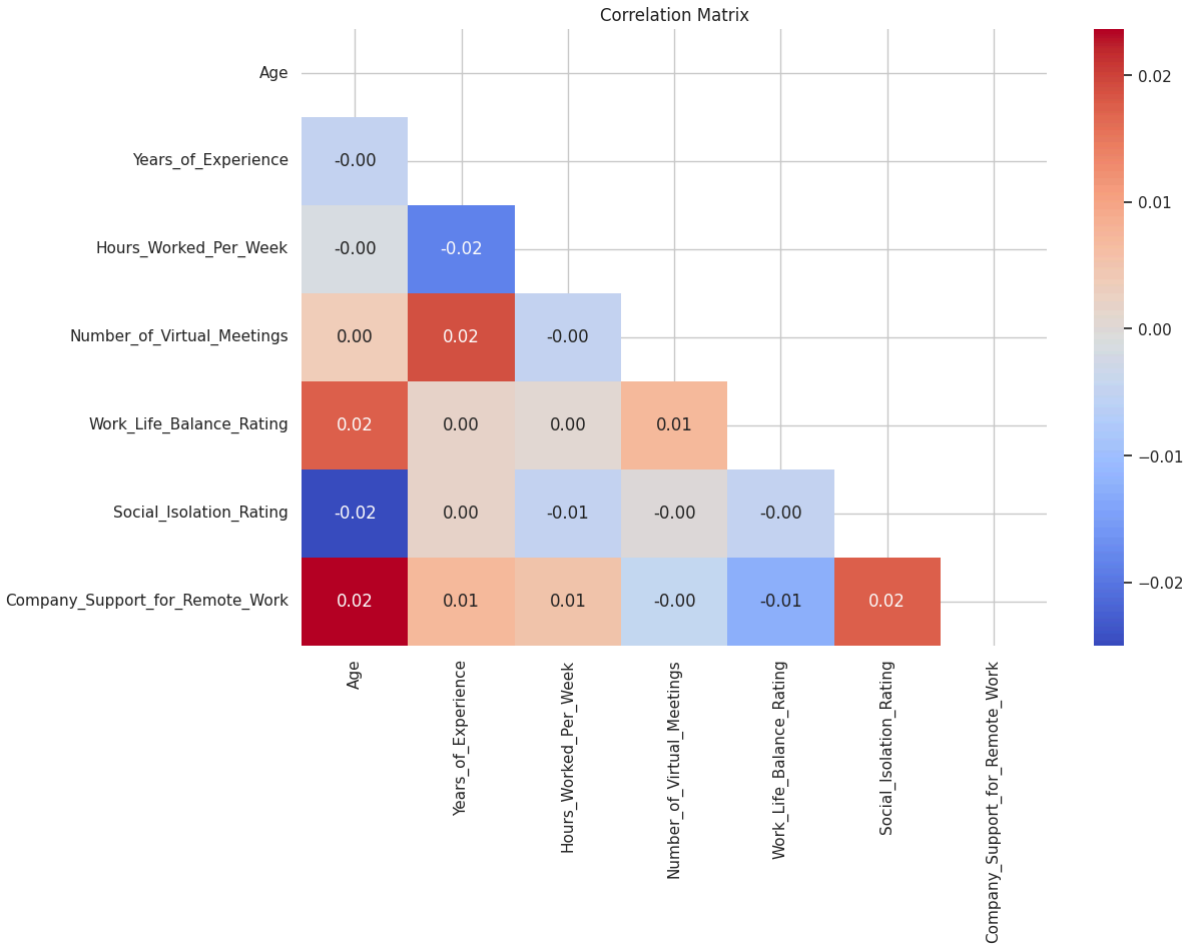

Figure 12: Correlation matrix

The correlation matrix heatmap in Figure 12 represents the correlations between variables in the dataset including Age, Years of Experience, Hours Worked Per Week, Number of Virtual Meetings, Work-Life Balance Rating, Social Isolation Rating, and Company Support for Remote Work. The range of correlation coefficients is from -0.02 to 0.02 The correlation coefficients indicate the direction and strength of linear relationships between the variables. As is observed from the above correlation matrix there is no important linear relationship between any pair of these variables since all the correlations are very close to zero. This implies that the above variables are independent of each other. Consequently, changes in one variable don't have a significant impact on another. This insight is very valuable for the study of the dataset, since it highlights the complexity of the relationships between the dataset's variables.

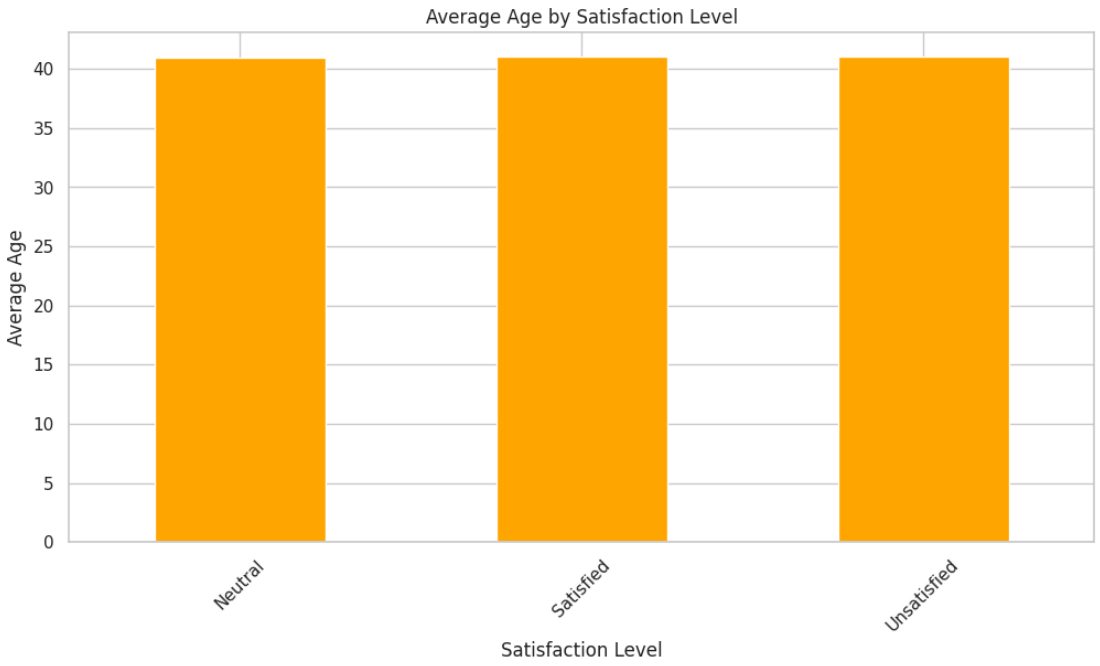

Figure 13: Average age by satisfaction level

Figure 14: Mental Health Condition bar plot

Figure 15: Work life balance rating chart

At Figure 13 is displayed a chart that compares the average age of employees across three satisfaction categories: Neutral, Satisfied, and Unsatisfied. The average age is very similar across all satisfaction levels, with each group's average age being slightly over 40. This means that age has no significant impact on participant's satisfaction. The average age of workers who are neutral, satisfied, or dissatisfied with their jobs is almost the same.

Figure 14 presents the prevalence of three key mental health conditions—depression, anxiety, and burnout—among employees. Anxiety and burnout have slightly higher incidences than depression, but the counts for these conditions are almost equal.

The distribution of employees' self-reported work-life balance ratings is presented in Figure 15. The work life balance rating is measured on a scale from 1 to 5, where 1 represents the lower rating and 5 represents the higher rating. Rating 2 has the lowest count, and the neutral rating (3) has the highest count. Similar counts are also found for ratings 1, 4, and 5, suggesting that there is no significant bias towards either high or low work-life balance. There is no indication of extreme satisfaction or dissatisfaction with work life balance. A broad spectrum of experiences within the participants may be reflected in this (probably related to the company and the department that each one works), as only a small percentage of workers report extremely high or extremely low levels of satisfaction, while the majority rate their work-life balance as moderate.

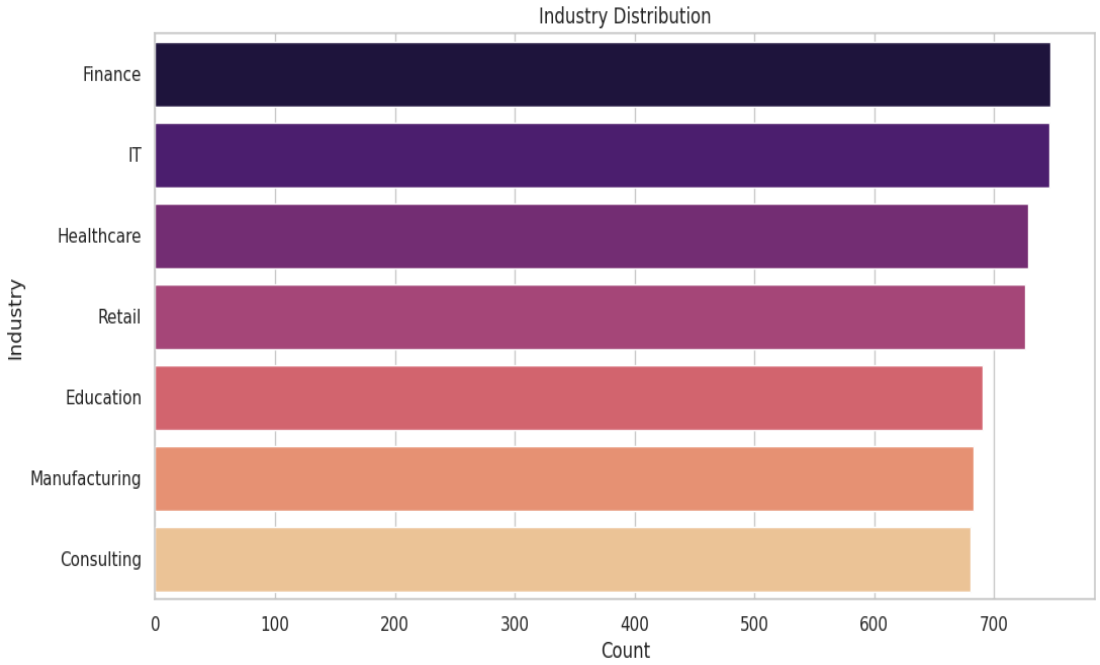

Figure 16: Industry distribution

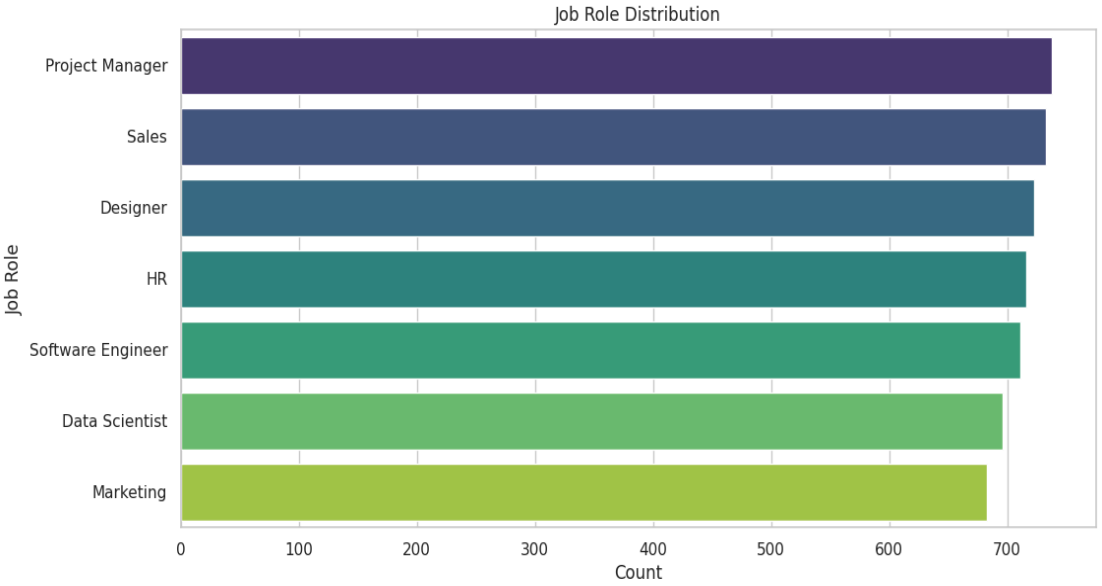

Figure 17: Job role distribution

Figure 16 and Figure 17 are presenting an overview of the industries and the job roles that are included in the dataset.

7. Machine Learning Approaches to Predict Employee Stress and Burnout

In the following paragraphs the study will focus on predicting the high stress levels among the participants as well as people who have experienced burnout during their work life. The ground truth for predicting high stress levels is the variable Stress_level and for predicting burnout is the variable Mental_Health_Condition.

7.1 Data preprocessing

In order to perform predictive models, firstly data preprocessing to transform the data in the suitable type is needed to perform.

- The Employee_ID column were eliminated from the dataset because it is not useful for the predictive analysis and might introduce bias or data leakage.
- The missing values were removed to ensure the integrity of the training data. In the dataset there are the following missing values:

Missing values	
Mental_Health_Condition	1196
Physical_Activity	1629

Table 1: Missing values in the dataset

- One-hot encoding for categorical values: avoids forcing ordinal relationships on non-ordinal data and generates binary columns for each category, was used to encode all categorical variables. The categorical variables that were encoded are the following: 'Gender', 'Job_Role', 'Industry', 'Work_Location', 'Company_Support_for_Remote_Work', 'Mental_Health_Condition', 'Productivity_Change','Access_to_Mental_Health_Resources', 'Satisfaction_with_Remote_Work','Physical_Activity','Sleep_Quality'.
- The target variable, which is Stress_Level and Mental_Heath_condition were transformed into a binary classification target in order to use classification models for the prediction. The participants that reported High stress level are labeled are 1 while all the other participants, meaning those who reported Medium and Low stress levels are labeled as 0. This method focuses on differentiating between people who are under a lot of stress and those who are not. Respectively, regarding the variable Mental_Heath_condition people, who reported that had burnout are labeled as 1 and the other mental health conditions are lebeled as 0. This case is concerned to distinct people that have experienced burnout and people that have experienced any other mental health disorder. Then these variables are dropped out of the dataset in order to perform the predictive models.

```python
# Load data
df = pd.read_csv('Impact_of_Remote_Work_on_Mental_Health.csv')

# Drop unnecessary columns
df = df.drop(['Region', 'Employee_ID'], axis=1)

# Handle missing values (drop rows with any nulls)
df = df.dropna()


# Convert Stress_Level to binary: High=1, else=0
df['Stress_Level_Binary'] = df['Stress_Level'].apply(lambda x: 1 if x == 'High' else 0)

# Drop original Stress_Level column
df = df.drop('Stress_Level', axis=1)

# Encode categorical variables using one-hot encoding
categorical_cols = df.select_dtypes(include=['object']).columns.tolist()
df = pd.get_dummies(df, columns=categorical_cols, drop_first=True)

# Split features and target
X = df.drop('Stress_Level_Binary', axis=1)
y = df['Stress_Level_Binary']
```

Figure 18: Code for one-hot encoding for Stress_level variable

```python
# Only keep rows where Mental_Health_Condition is not null
df = df[df['Mental_Health_Condition'].notnull()]

# Create binary target: Burnout = 1, else = 0
df['Burnout_Binary'] = df['Mental_Health_Condition'].apply(lambda x: 1 if x == 'Burnout' else 0)

# Drop original column
df = df.drop('Mental_Health_Condition', axis=1)
```

```python
#drop  missing values
df = df.dropna()
```

```python
categorical_cols = df.select_dtypes(include=['object']).columns.tolist()
df = pd.get_dummies(df, columns=categorical_cols, drop_first=True)
```

```python
X = df.drop('Burnout_Binary', axis=1)
y = df['Burnout_Binary']
```

Figure 19: Code for one-hot encoding for Mental_health_Condition variable

7.2 Metrics

The metrics that will be used to evaluate the predictive models will be presented and detailed explained in the following paragraph.

- Accuracy: the proportion of accurate predictions to all the model's predictions. Formally, accuracy has the following definition:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

where TP, TN, FP, FN correspond to True Positives, True Negatives, False Positives and False Negatives accordingly. Accuracy is intuitive and provides a

quick overview of the model's performance. Although this metric can be misleading, especially when dataset has imbalanced classes.

- Precision: the proportion of cases that are actually positive out of those that were predicted to be so. Actually, it attempts to answer the following question: How many predictions are actually correct out of all the positive predictions? The definition of precision is the following:

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

  Precision is important when the cost of false positives is high.
- Recall: the ability of the model to identify all actual positives. It answers the question: Out of all true positives, how many did the model correctly identify? The definition of recall is the following:

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

- F1-score: combines precision and recall using the harmonic mean. The formula for F1-score is the following: $F1 = 2 \times \frac{Precision \times Recall}{Precision+Recall}$. F1- score is high if both precision and recall is high. If one of these declines significantly, the F1 score declines also. It is useful for imbalanced datasets where a balance between precision and recall is required.
- Support: the number of actual occurrences of each class in the dataset (i.e., the number of true instances for each class). It is not a metric itself, but it provides context to the precision, recall, and F1-score metrics.
- Confusion Matrix: a table used to describe the performance of a classification model by comparing actual vs. predicted labels. This matrix is the basis for calculating accuracy, precision, recall, and F1-score. For a binary classification, it is a 2x2 table of:
  - True Positives (TP)
  - True Negatives (TN)
  - False Positives (FP)
  - False Negatives (FN)

7.3 Predictive models

In this case two different machine learning models are used, Random Forest Classifier and XGBoost Classifier. In the following Chapters a detailed description about these models will be provided.

7.3.1 Random Forest Classifier

Random Forest Classifier is an ensemble learning algorithm that creates multiple decision trees during training and combines their predictions to produce the final output. It is used for both classification (which is our case) and regression tasks.

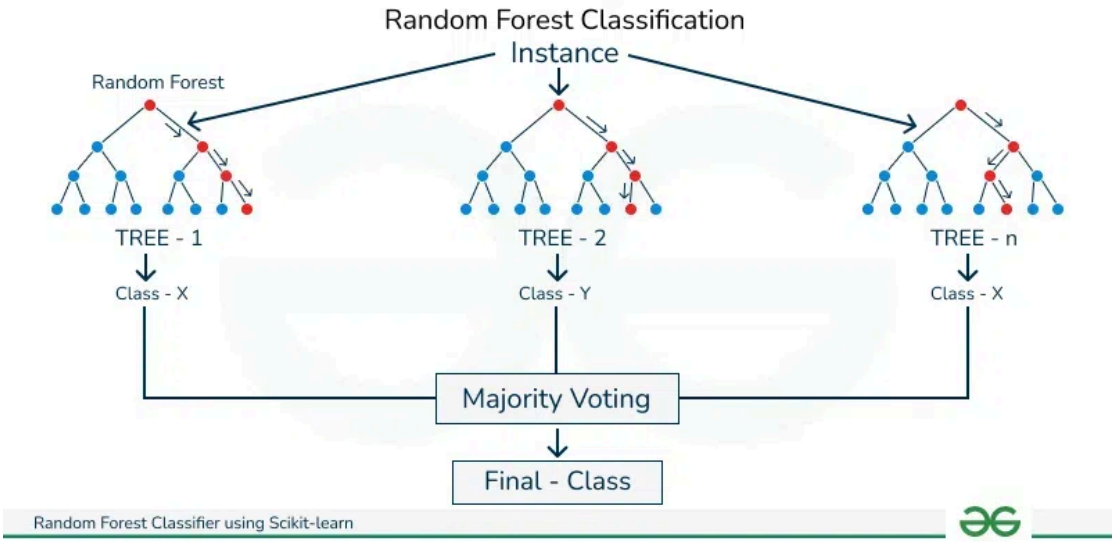

Figure 20: Random Forest Classifier overview

Regarding the classification tasks, Random Forest Classifier aims to predict the categorical result considering the input data. In order to achieve this, it utilizes multiple decision trees and outputs the label with the most votes out of all the predictions made by each individual tree. Each tree is trained using a random subset of the features and data in order to avoid overfitting. The process of Random Forest Classifier is presented explicitly below:

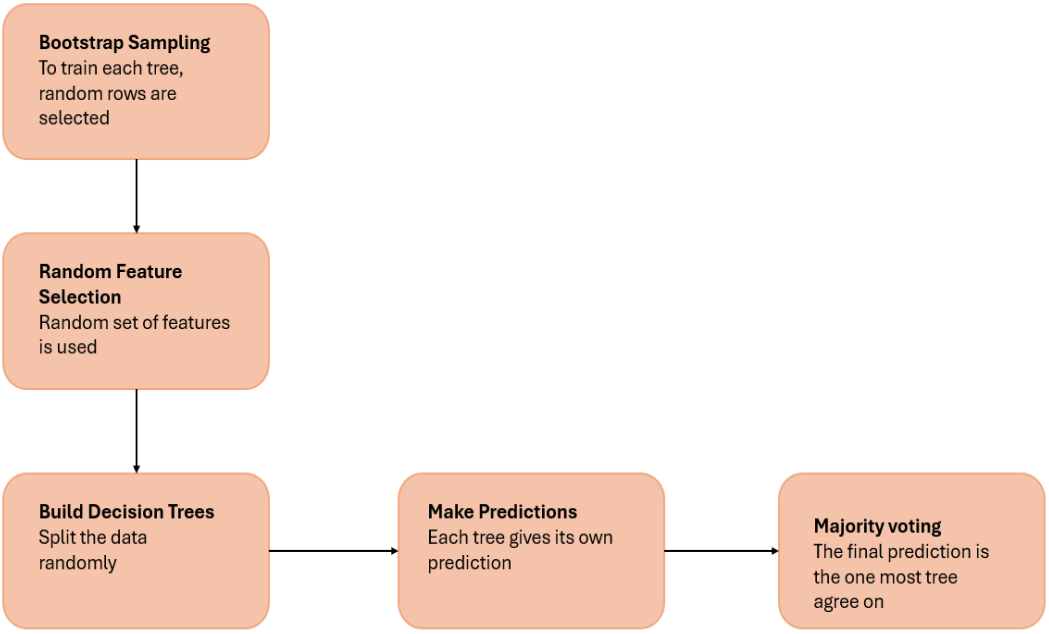

Figure 21: Random Forest Classifier working process

Some advantages of Random Forest Classifier are the following:

- It is suitable for large datasets and high-dimensional data.

- Compared to a single decision tree, it has lower overfitting risk because of its ability to combine multiple trees.
- It is resilient to noisy data and it performs well with categorical data.

7.3.2 XGBoost

XGBoost (eXtreme Gradient Boosting) is also an ensemble method. More specifically, it is an advanced machine learning algorithm which aims to efficiency, speed and high performance.

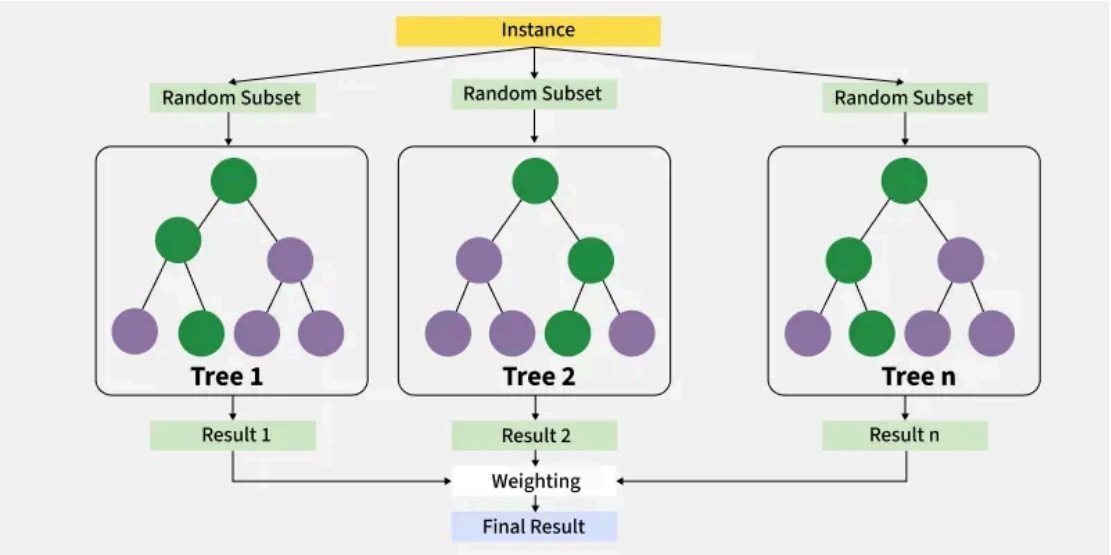

Figure 22: XGBoost overview

In contrast to Random Forest Classifier, which builds trees independently, XGBoost builds trees sequentially and each new tree is trained to fix the mistakes made by the one before it. This process is called boosting. Specifically, the whole algorithm of XGBoost is presented below:

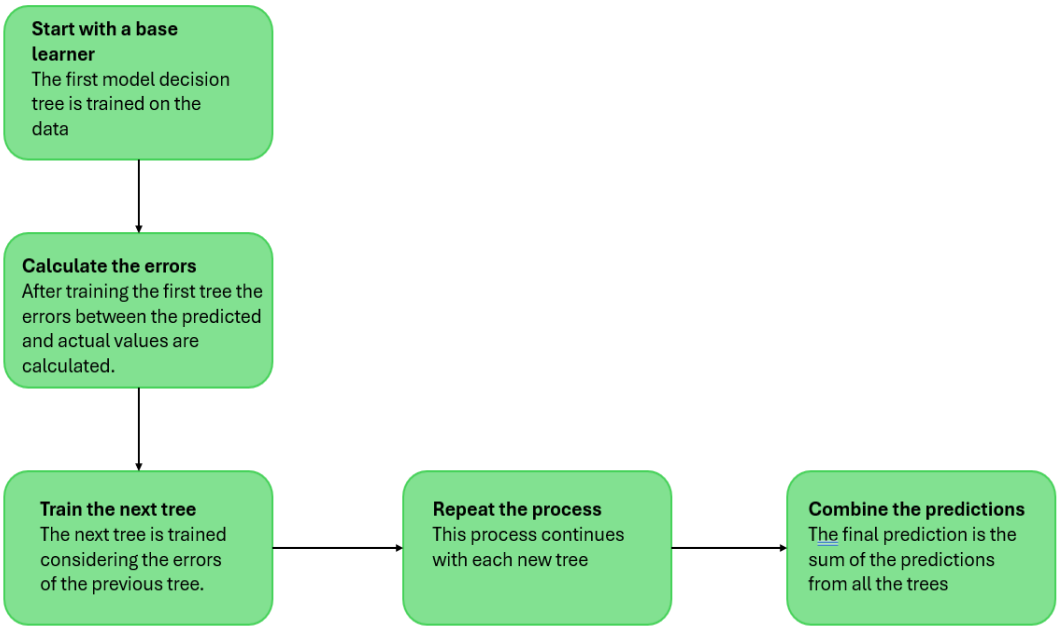

Figure 23: XGBoost algorithm detailed description

A major benefit of XGBoost is that it can handle missing values automatically during training. Moreover, it can effectively reduce overfitting since it includes regularization. Also, XGBoost builds trees level-wise or breadth-first. This means that the tree grows one level at a time by adding nodes for each feature at a specific depth before going on to the next level.

7.4 High stress levels

In the following paragraphs a detailed methodology for the machine learning models that are used for the prediction of high stress levels will be described.

The first predictive model that is used in order to predict high stress levels among employees is the **Random Forest Classifier**. The dataset was split into features and the binary target variable (which are converted to binary variable). An 80/20 split was used to separate the data into training and testing sets, and stratification was used to guarantee that both classes were fairly represented in the target variable. This guarantees that the model is tested on a representative sample and helps avoid bias in model evaluation. Then a Random Forest Classifier was trained on the training set. The results from the metrics are presented below:

Classification Report:

	precision	recall	F1-score	support
0(low and medium)	0.66	0.97	0.79	342
1(high)	0.40	0.03	0.06	174
accuracy			**0.66**	516

Table 2: Classification report for Random Forest for stress levels prediction

```
# Split features and target
X = df.drop('Stress_Level_Binary', axis=1)
y = df['Stress_Level_Binary']

# Split into train and test sets
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

# Train RandomForestClassifier
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)

# Predict on test set
y_pred = model.predict(X_test)

# Evaluate model
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))

# Confusion matrix
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

Figure 24: Code for Random Forest Classifier for stress prediction

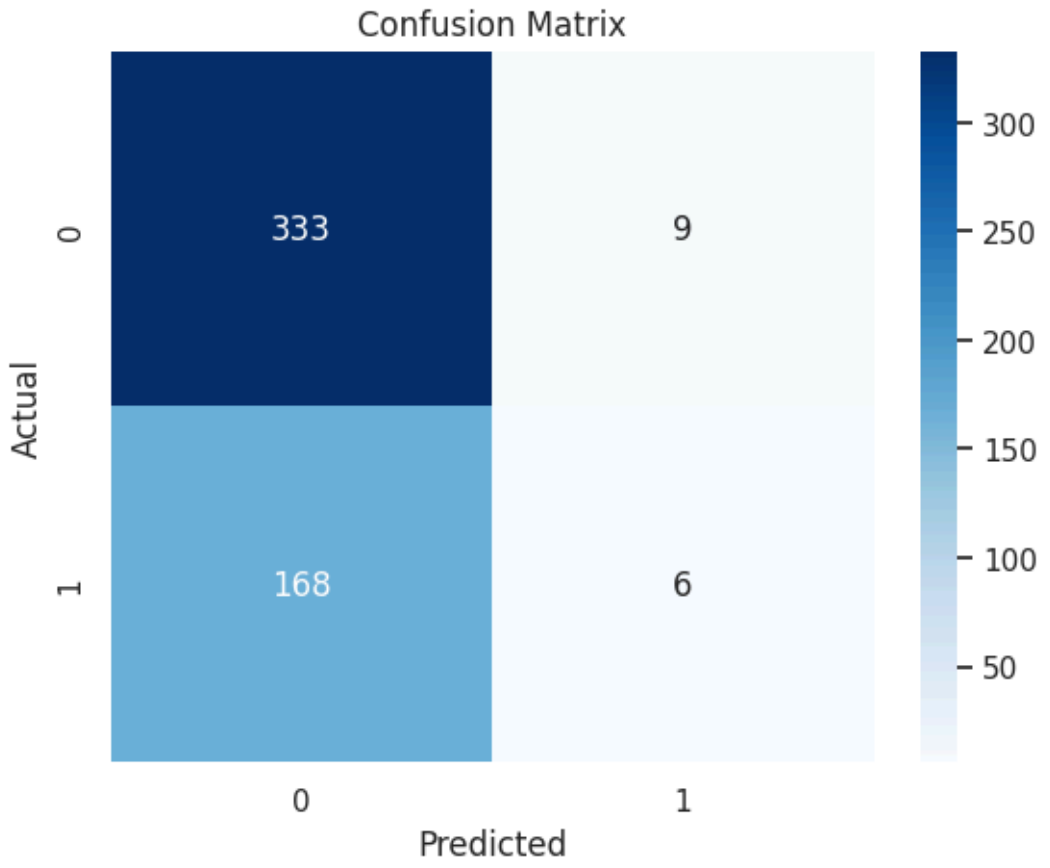

Figure 25: Confusion matrix for Random Forest for stress prediction

The Random Forest classifier achieved an overall accuracy of 66%. However, with a recall of 0.97, the model only performed well for the majority class, which is not high stress. Also, with a recall of only 0.03 and an F1-score of 0.06 for the class 1(high stress levels), it was unable to accurately identify high-stress employees. This problem is further highlighted by the confusion matrix, which shows that only 6 out of 174 high-stress cases were correctly classified. These results indicate that the model is not appropriate for identifying high-stress workers with accuracy. This happens most likely because of the dataset's class imbalance.

Since a significant class imbalance was revealed from the first model evaluation improvements should be done. The minority class, which is employees with high stress levels, were underrepresented and in order to change this Synthetic Minority Over-sampling Technique (SMOTE) were used. SMOTE is a common method for balancing unbalanced datasets by creating synthetic samples of the minority class, to lessen this problem. After using the SMOTE method, the new evaluation metrics are summarized below:

Classification Report:

	precision	recall	F1-score	support
0(low and medium)	0.69	0.85	0.76	342

1(high)	0.80	0.61	0.71	342
accuracy			**0.73**	648

Table 3: Classification report for Random Forest Classifier after SMOTE for stress levels prediction

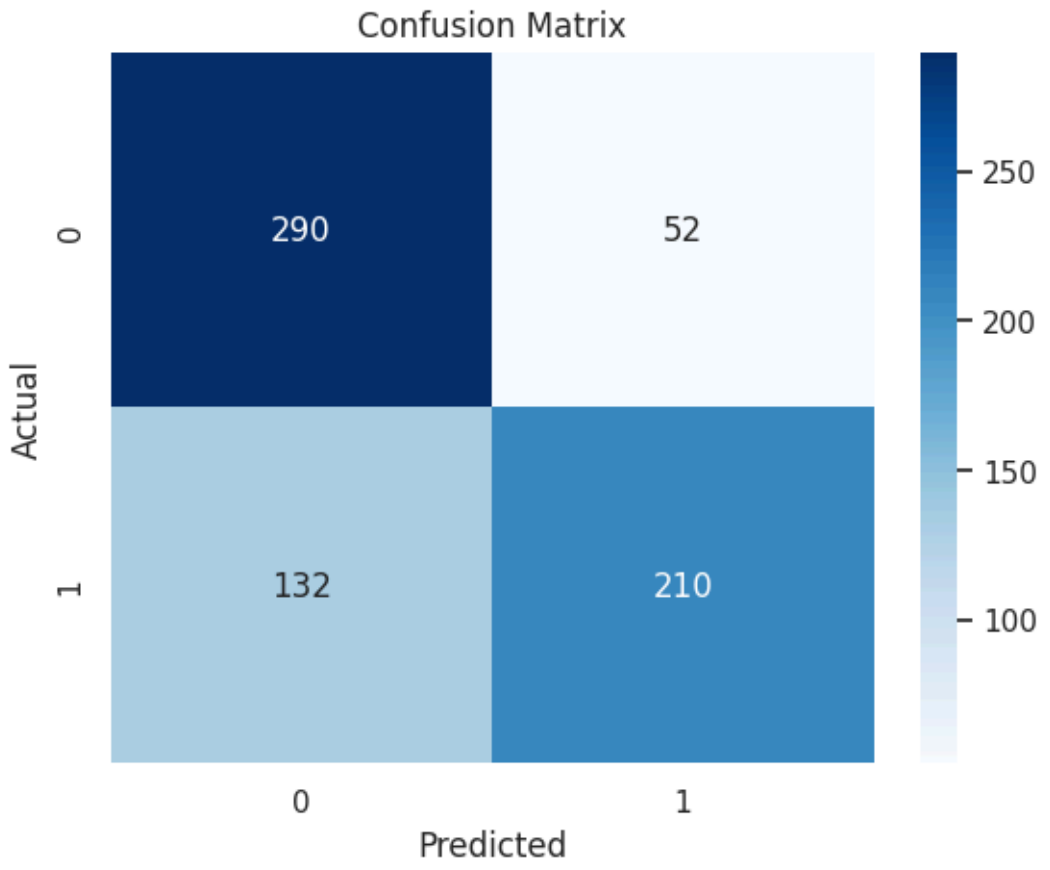

Figure 26: Confusion matrix after SMOTE for stress levels prediction

The new overall accuracy is 73% meaning that a significant improvement was made after SMOTE were applied. In comparison to earlier attempts, the new accuracy shows a better overall ability to accurately classify employees' stress levels. It is obvious that the precision and the recall values indicate a more balanced performance. With a precision of 0.69, which means that roughly 69% of predicted negatives are accurate, and a good recall (0.85), which means it correctly identifies 85% of employees without high stress, the model performs well for the not high stress group (class 0). Precision for the high stress group (class 1) is higher at 0.80, indicating that 80% of predicted high stress cases are accurate, while recall is 0.61, indicating that 61% of true high stress cases are identified by the model. The precision and recall are well-balanced, as indicated by the F1-score of 0.70. These new results indicate that the model can effectively identify employees that are going to be under high stress, while also accurately classifying employees who are not under stress.

```
smote = SMOTE(random_state=42)
X_res, y_res = smote.fit_resample(X, y)

# Now split the resampled data
X_train, X_test, y_train, y_test = train_test_split(
    X_res, y_res, test_size=0.2, random_state=42, stratify=y_res
)

# Train and evaluate as before
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))

cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

Figure 27 Code for Random Forest Classifier for stress level prediction using SMOTE

To investigate if further improvement is possible for the model's performance a grid search hyperparameter tuning using GridSearchCV is utilized. This method methodically investigates combinations of important hyperparameters in order to determine the optimal model configuration based on cross-validated F1-macro scores that balance precision and recall across classes.

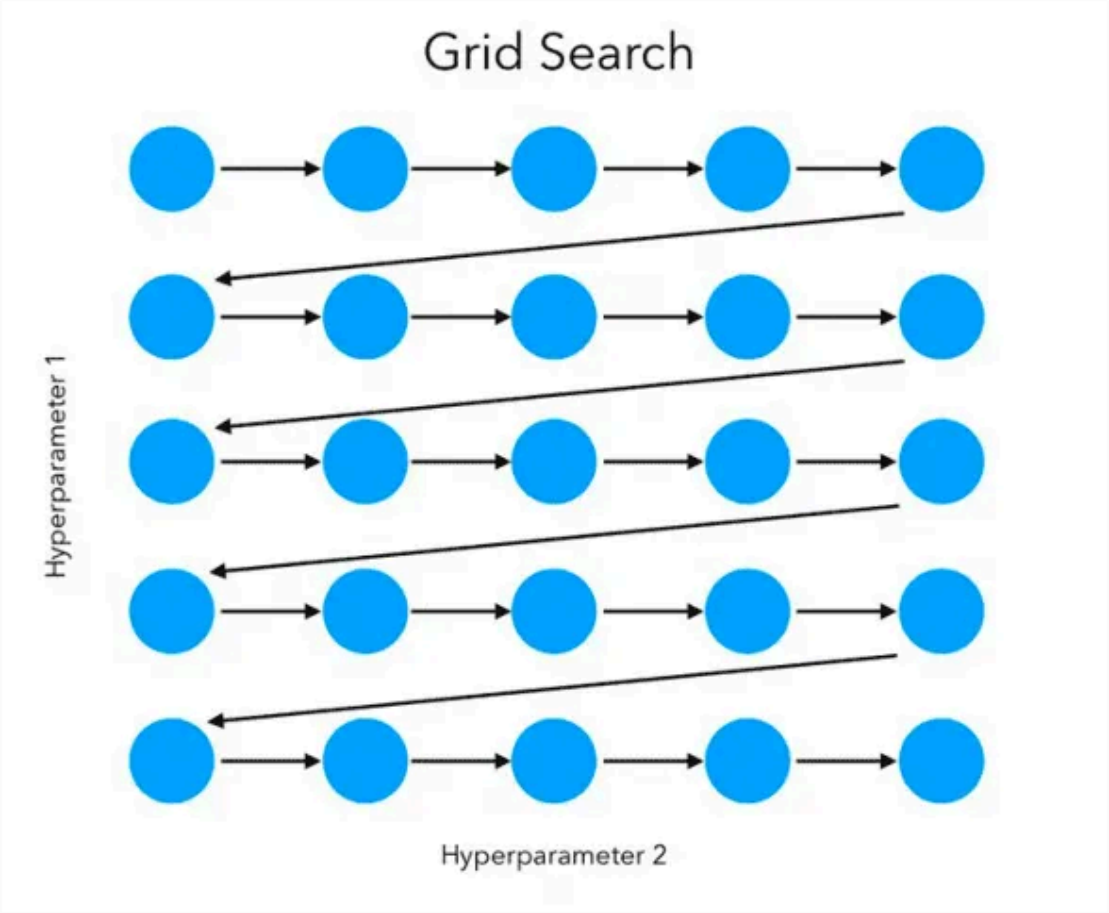

Figure 28: Grid Search overview

Below there is a description of the steps of GridSearchCV:

1. The function is initialized with an estimator, and a parameter grid.
2. Data is split into training and test sets.
3. The function iterates over the parameter grid and the estimator is trained on the training set for each combination of hyperparameters.
4. Then it evaluates the estimator's performance on the test set for each combination of hyperparameters.
5. The function returns the combination of hyperparameters that resulted in the best performance

A random search over a predefined hyperparameter space was carried out. The hyperparameters that are used are the following:

- n_estimators: [100, 200, 300],
- max_depth: [None, 10, 20, 30],
- min_samples_split: [2, 5, 10],
- min_samples_leaf: [1, 2, 4],
- bootstrap: [True, False]

where:

- **n_estimators**: The number of decision trees in the forest.
- **max_depth**: The maximum depth of each tree; None indicates that nodes will expand until all leaves are pure or contain fewer than min_samples_split samples.
- **min_samples_split**: The minimum number of samples required to split an internal node.
- **min_samples_leaf**: The minimum number of samples required to be at a leaf node.
- **bootstrap**: Whether bootstrap samples are used when building trees.

The grid search evaluated 216 parameter combinations over 3-fold cross-validation, resulting in 648 fits and resulted in the following parameters:

- n_estimators: 300,
- min_samples_split: 2,
- min_samples_leaf: 1,
- max_depth: 20,
- bootstrap: True

The findings of this method are displayed in the following table:

Classification Report:

	precision	recall	F1-score	support
0(low and medium)	0.68	0.83	0.75	342
1(high)	0.78	0.61	0.69	342
accuracy			**0.72**	648

Table 4: Classification report with GridSearchCV for stress level prediction

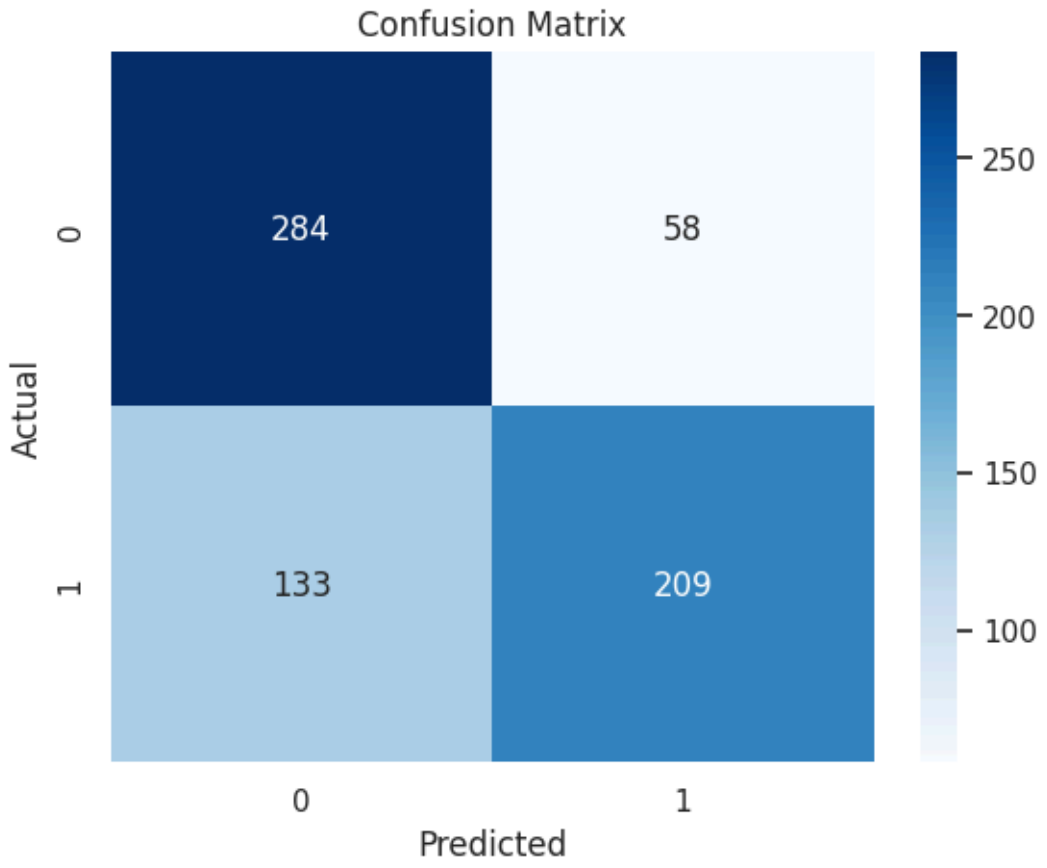

Figure 29: Confusion matrix with GridSearchCV for stress level prediction

An optimal Random Forest configuration with 300 trees and a maximum depth of 20 was found through hyperparameter tuning using grid search. The accuracy of 72% and balanced precision-recall metrics of the final model, however, only indicated a slight improvement over the default model.

```
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))

cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV

# Define parameter grid
param_grid = {
    'n_estimators': [100, 200, 300],        # number of trees
    'max_depth': [None, 10, 20, 30],        # max depth of each tree
    'min_samples_split': [2, 5, 10],        # min samples to split a node
    'min_samples_leaf': [1, 2, 4],          # min samples at a leaf node
    'bootstrap': [True, False]               # bootstrap samples or not
}

# Initialize RandomForest
rf = RandomForestClassifier(random_state=42)

# GridSearch with 3-fold CV
grid_search = GridSearchCV(estimator=rf, param_grid=param_grid,
                           cv=3, scoring='f1_macro', n_jobs=-1, verbose=2)

# Fit on training data
grid_search.fit(X_train, y_train)
```

Figure 30: Code for Random Forest Classifier with grid search for stress level prediction

After this, XGBoost (Extreme Gradient Boosting) model is performed to make the same predictions as previously. Firstly, the XGBoost classifier is initialized, and a parameter grid is used to explore different combinations of hyperparameters for the best model performance. The hyperparameters that are used are the following:

- estimator=xgb_model,
- param_grid=param_grid,
- scoring='f1_macro',
- cv=3,
- verbose=2,
- n_jobs=-1

where:

- estimator: the machine learning model that will be utilized
- param_grid: a dictionary specifying the hyperparameters and the values to test
- scoring: metric used to evaluate the model performance during the search
- cv: number of cross-validation folds.It creates CV segments from the training data. While the remainder is used for training, each component is used once as a validation set. This offers better model performance since it minimizes the overfitting.
- verbose: the amount of information printed during execution
- n_jobs: number of CPU cores to use for computation. -1 means that all available cores will be used

The grid search evaluated 216 parameter combinations over 3-fold cross-validation, and resulted in the following parameters:

- colsample_bytree: 0.8
- learning_rate: 0.01
- max_depth: 10
- n_estimators: 200
- subsample: 0.8

The results of the above machine learning model are presented in the following classification report:

Classification Report:

	precision	recall	F1-score	support
0(low and medium)	0.67	0.80	0.73	342
1(high)	0.76	0.61	0.68	342
accuracy			**0.71**	648

Table 5: Classification report for XGBoost with hyperparameters for stress level prediction

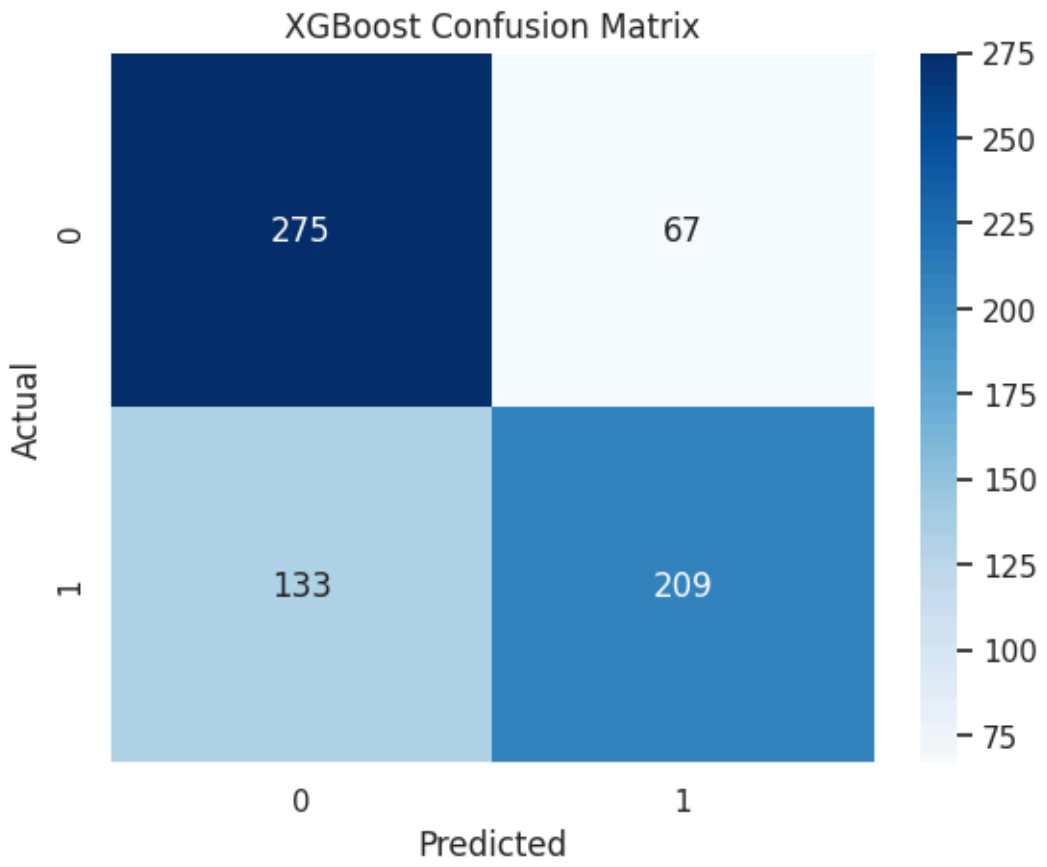

Figure 31: XGBoost confusion matrix for stress level prediction

```
# 1. Train the XGBoost Classifier
xgb_model = xgb.XGBClassifier(
    objective='binary:logistic',
    eval_metric='logloss',
    use_label_encoder=False,
    random_state=42
)

# Optional: Tune hyperparameters
param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [3, 6, 10],
    'learning_rate': [0.01, 0.1, 0.2],
    'subsample': [0.8, 1.0],
    'colsample_bytree': [0.8, 1.0]
}

grid_search = GridSearchCV(estimator=xgb_model,
                           param_grid=param_grid,
                           scoring='f1_macro',
                           cv=3,
                           verbose=2,
                           n_jobs=-1)

# Fit the model on training data
grid_search.fit(X_train, y_train)
```

Figure 32: Code for XGBoost for stress level prediction

```
# 2. Best estimator
best_xgb = grid_search.best_estimator_
print("Best Parameters:", grid_search.best_params_)

# 3. Make predictions
y_pred = best_xgb.predict(X_test)

# 4. Evaluate the model
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))

# 5. Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('XGBoost Confusion Matrix')
plt.show()
```

Figure 33: Code for XGBoost for stress level prediction (part 2)

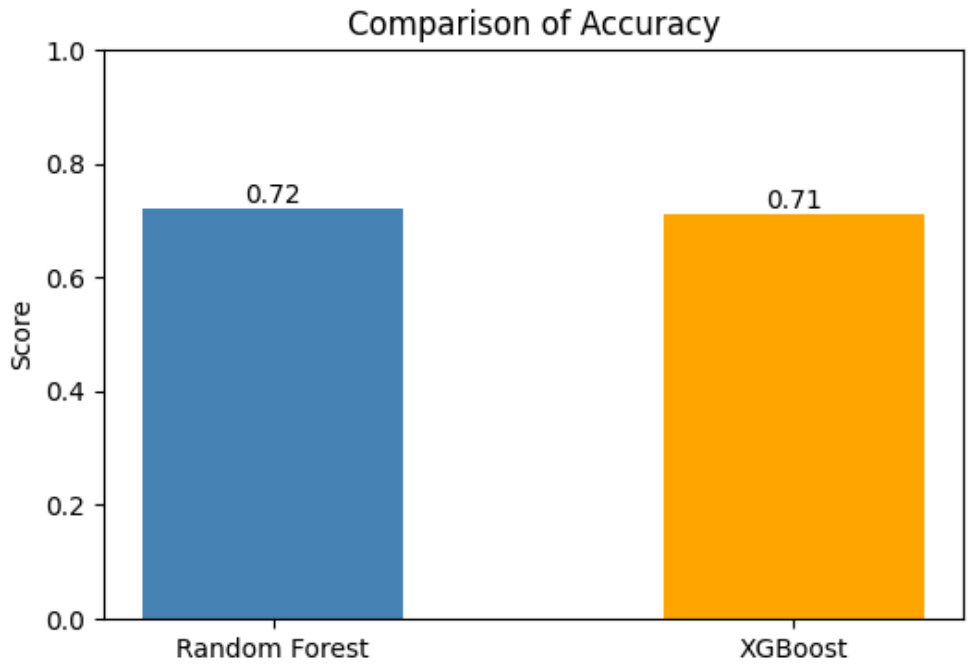

Figure 34: Random Forest and XGBoost accuracy comparison for stress level prediction

After performing grid search with hyperparameters for both models the results produced by the models, Random Forest and XGBoost, are almost the same with overall accuracy of 72% and 71%, respectively. Although SMOTE algorithm is used in order to handle class imbalance, the models still have lower recall for the high-stress class. This indicates that models struggle to identify with accuracy people who are under a lot of stress. Another limitation that is observed is that the size of the dataset may not be sufficient to make better predictions. The lack of diversity, as well as the slightly small number of records may be an obstacle, especially for XGBoost, which is best utilized for larger datasets.

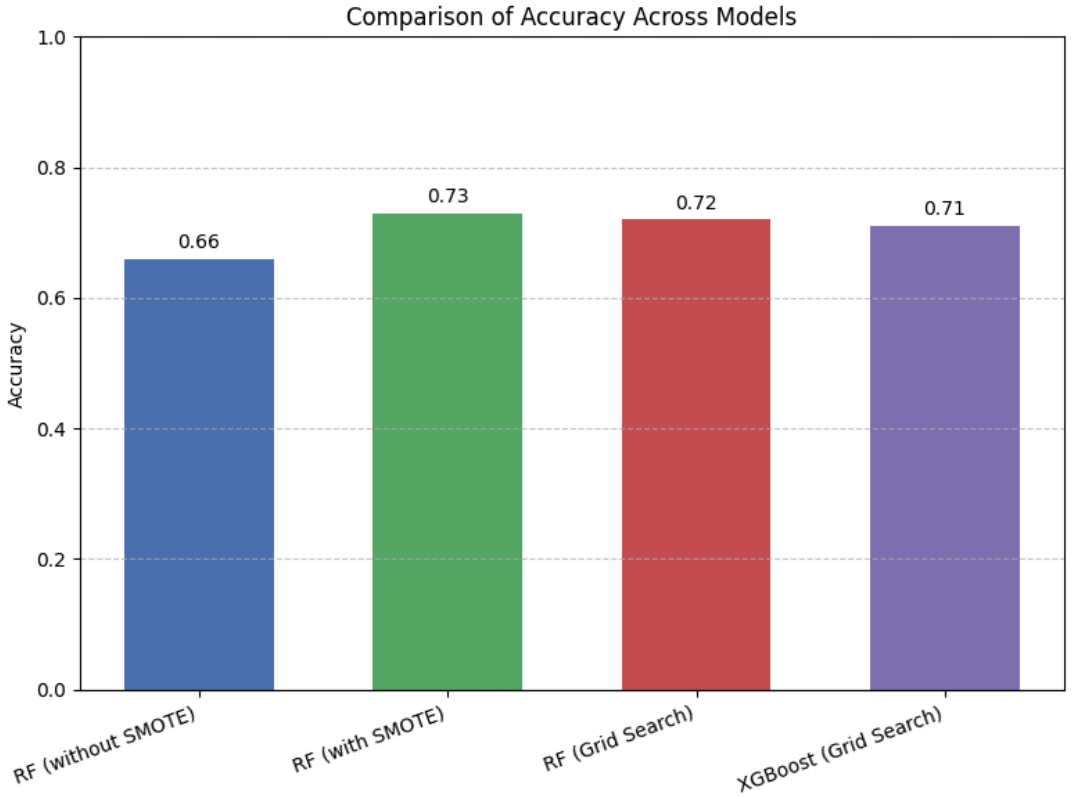

Figure 35: Accuracy comparison across all models for stress prediction

Figure 35 presents the accuracy across all models. The lowest accuracy is achieved by Random Forest without using the SMOTE technique (66%). Then SMOTE technique is applied and as a result the Random Forest achieved higher accuracy (73%). Hyperparameter tuning with Grid Search provided a slight improvement over the baseline Random Forest, achieving 72%. XGBoost with Grid Search obtained an accuracy of 71%, comparable to Random Forest. These findings indicate that class balancing, namely using the SMOTE technique, had a bigger impact on accuracy.

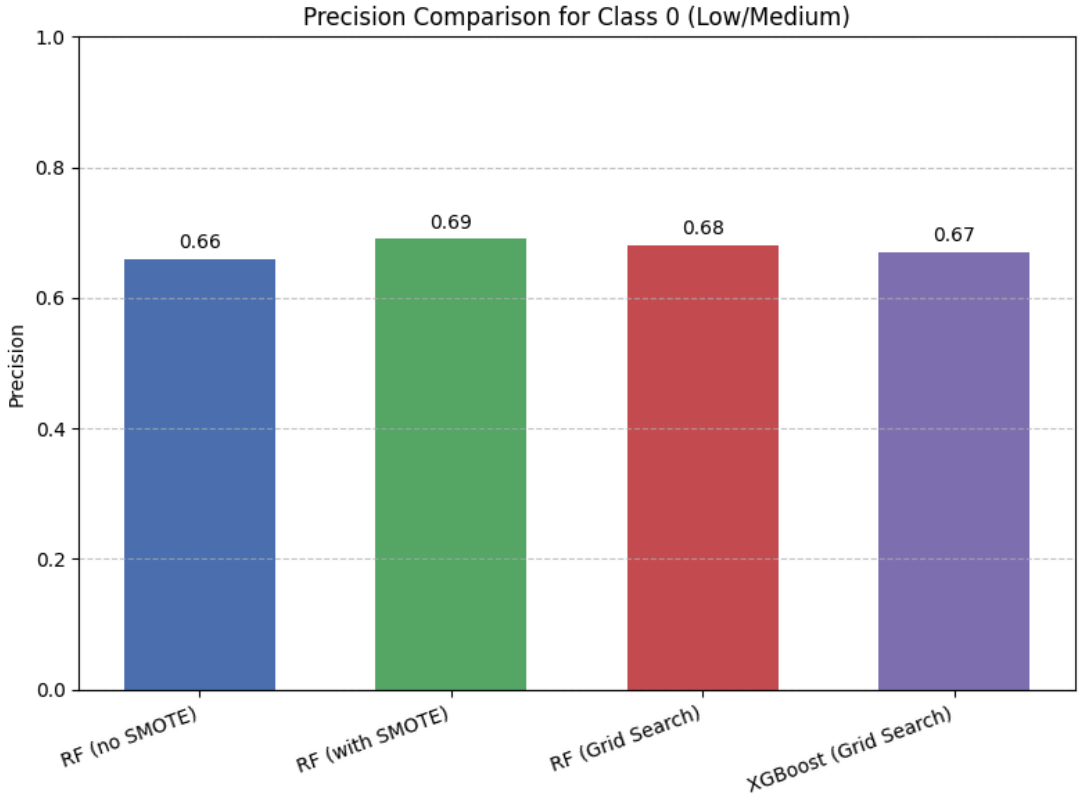

Figure 36: Precision comparison for low/medium stress level

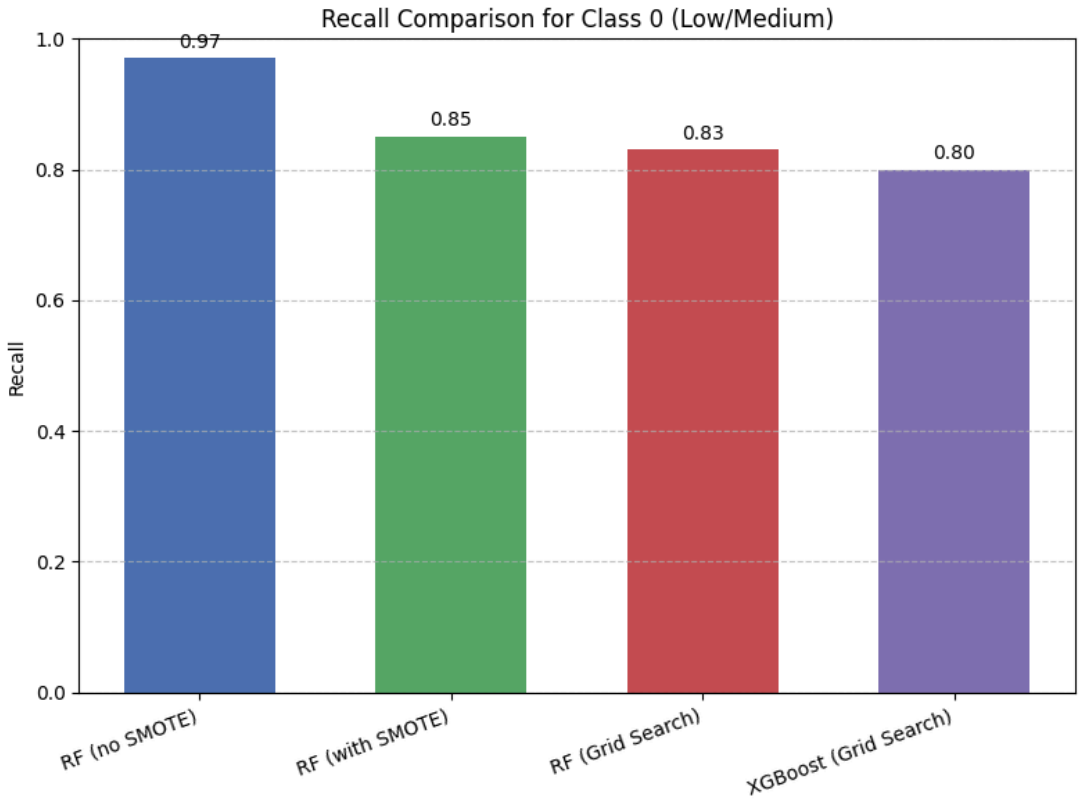

Figure 37: Recall comparison for low/medium stress level

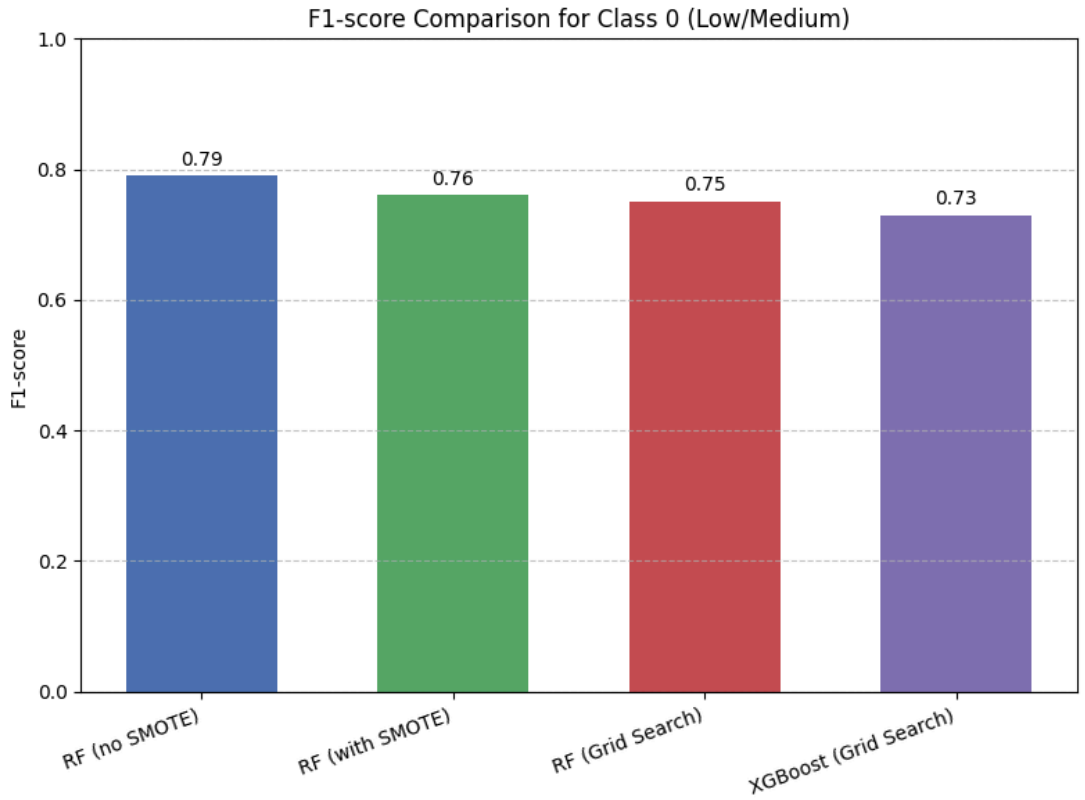

Figure 38: F1-score comparison for low/medium stress level

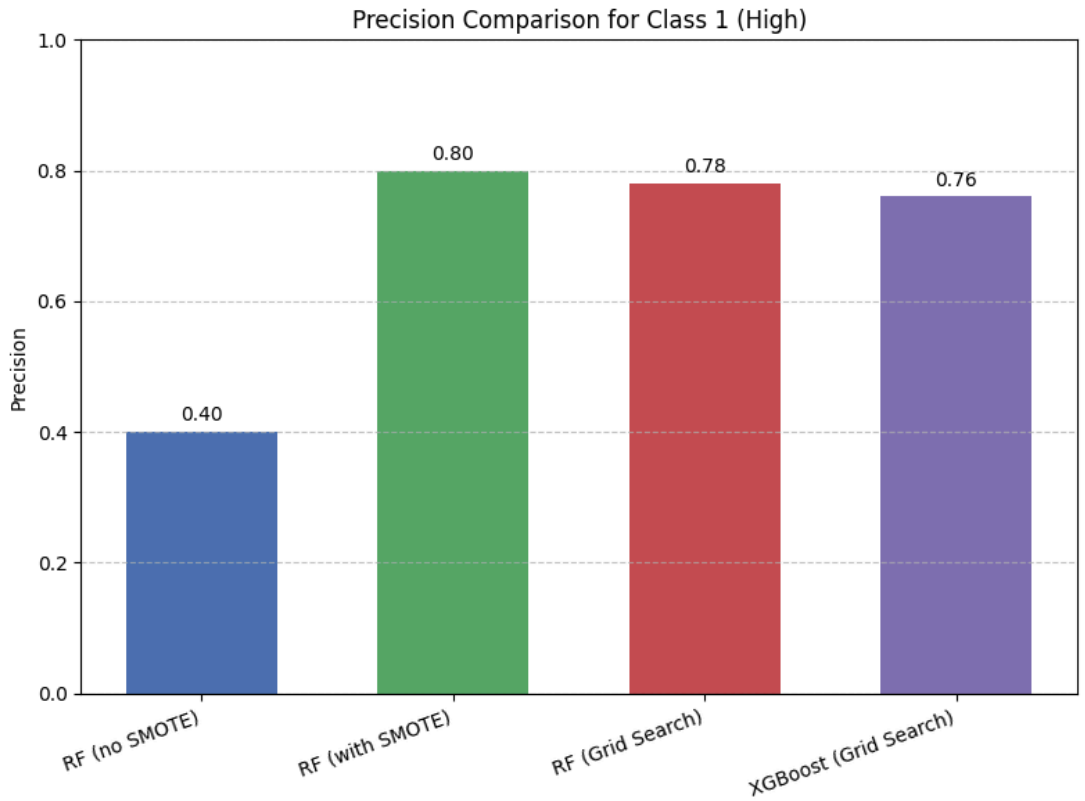

Figure 39: Precision comparison for high stress level

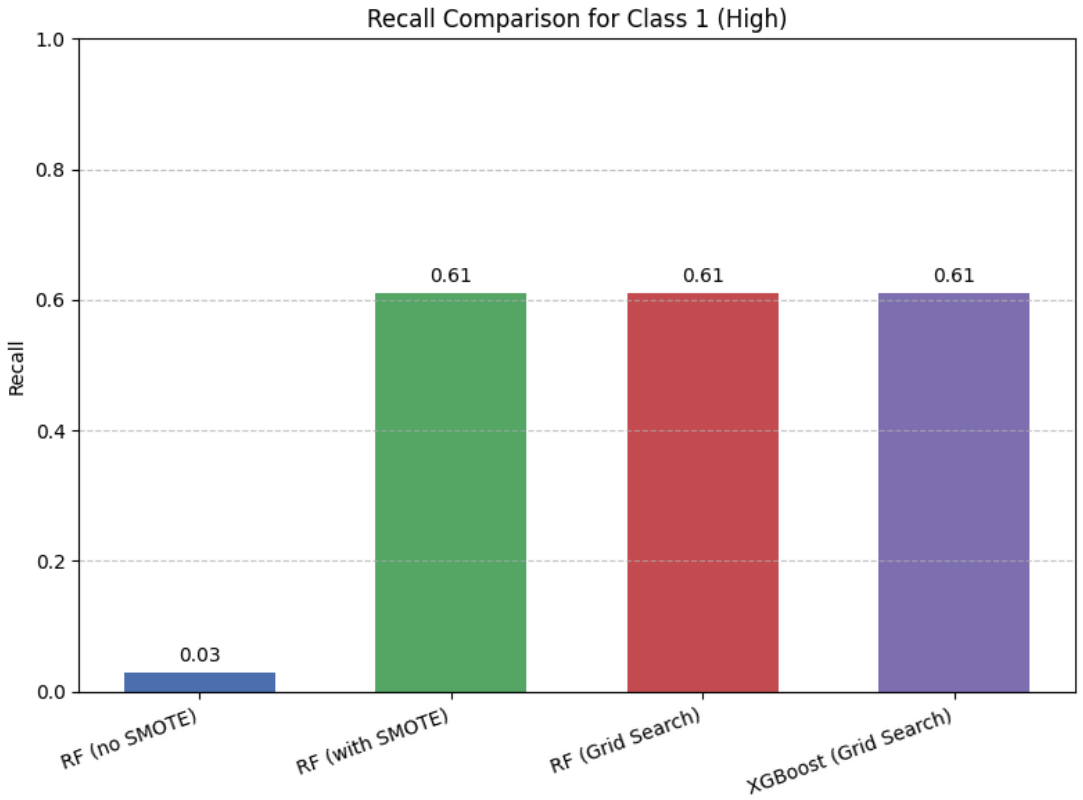

Figure 40: Recall comparison for high stress level

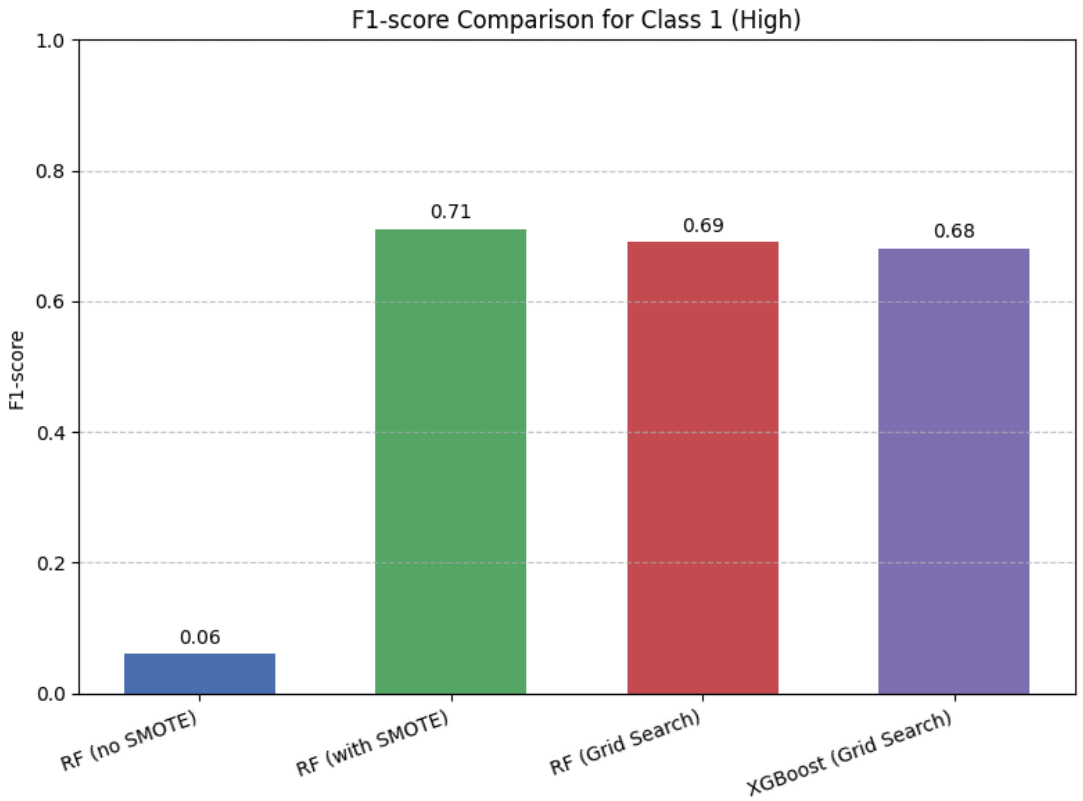

Figure 41: F1-score comparison for high stress level

Figure 36-Figure 41 illustrate a comparison between precision, recall and F1-score across all the models for both classes. Random Forest before applying SMOTE calculated a high recall (97%) for class 0 (low/medium stress class), but the recall and F1-score for class 1(high stress class) was very low (recall = 3%, F1 = 6%). Then SMOTE was used and the performance had a dramatical increase, achieving 61% recall and 71% F1-score. Random Forest with Grid Search and XGBoost with Grid Search had almost the same performance for both classes with balanced precision-recall metrics. These outcomes indicate that class balancing has the most significant impact on the models' performance.

7.5 Burnout

Burnout is undoubtedly a major problem among a high majority of employees in modern society. Predicting burnout contributes to a better work environment and improves well-being. In the following paragraphs a detailed methodology for the machine learning models that are used for the prediction of burnout will be described. Moreover, the most impactful features that lead to burnout will be presented in order to gain actionable insights.

For the predictions of burnout the same models as described previously — Random Forest and XGBoost — will be used. In order to apply the algorithms and implement the predictions, the target variable, which is the Mental_Health_Condition variable, is converted to binary. Namely, cases where people have reported burnout are marked with 1 otherwise are marked with 0. Moreover, the target variable is removed from the dataset. The SMOTE algorithm is also used in this cases in order to eliminate the class imbalance. The results are presented below:

Classification Report (Random Forest):

	precision	recall	F1-score	support
0(no burnout)	0.69	0.86	0.77	345
1(burnout)	0.81	0.61	0.70	345
accuracy			**0.73**	690

Table 6: Burnout classification report - Random Forest

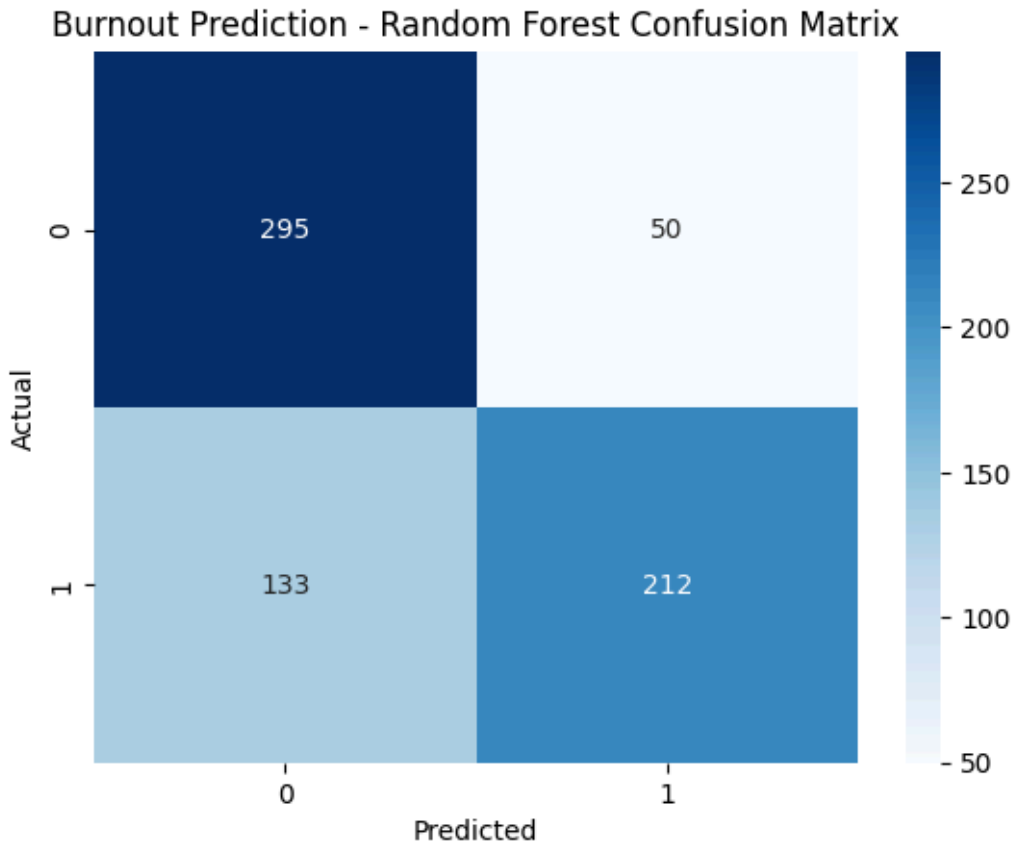

Figure 42: Burnout prediction – Random Forest Confusion Matrix

```
from imblearn.over_sampling import SMOTE

smote = SMOTE(random_state=42)
X_res, y_res = smote.fit_resample(X, y)
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X_res, y_res, test_size=0.2, random_state=42, stratify=y_res
)
```

Figure 43: Splitting into test and train set for burnout prediction

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train, y_train)

y_pred = rf_model.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))

cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Burnout Prediction - Random Forest Confusion Matrix')
plt.show()
```

Figure 44: Random Forest Classifier code for burnout prediction

Classification Report (XGBoost):

	precision	recall	F1-score	support
0(no burnout)	0.70	0.81	0.75	345
1(burnout)	0.78	0.65	0.71	345
accuracy			**0.73**	690

Table 7: XGBoost classification report for burnout prediction

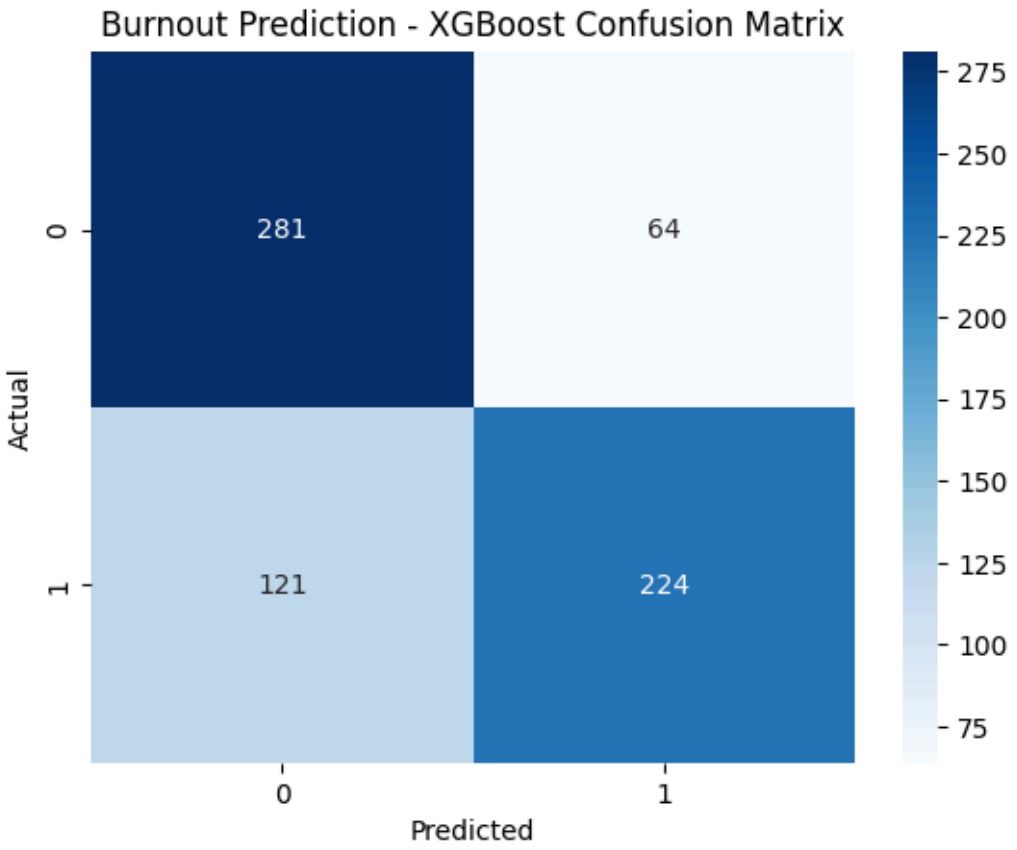

Figure 45: XGBoost Confusion Matrix for burnout prediction

```python
import xgboost as xgb
from sklearn.model_selection import GridSearchCV

xgb_model = xgb.XGBClassifier(
    objective='binary:logistic',
    eval_metric='logloss',
    use_label_encoder=False,
    random_state=42
)

param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [3, 6],
    'learning_rate': [0.01, 0.1],
    'subsample': [0.8, 1.0],
    'colsample_bytree': [0.8, 1.0]
}

grid_search = GridSearchCV(
    estimator=xgb_model,
    param_grid=param_grid,
    scoring='f1_macro',
    cv=3,
    verbose=2,
    n_jobs=-1
)
```

Figure 46: Code implementation for XGBoost for burnout prediction (part 1)

```python
grid_search.fit(X_train, y_train)
best_xgb = grid_search.best_estimator_
y_pred = best_xgb.predict(X_test)

print("Best Parameters:", grid_search.best_params_)
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

Figure 47: Code implementation for XGBoost for burnout prediction (part 2)

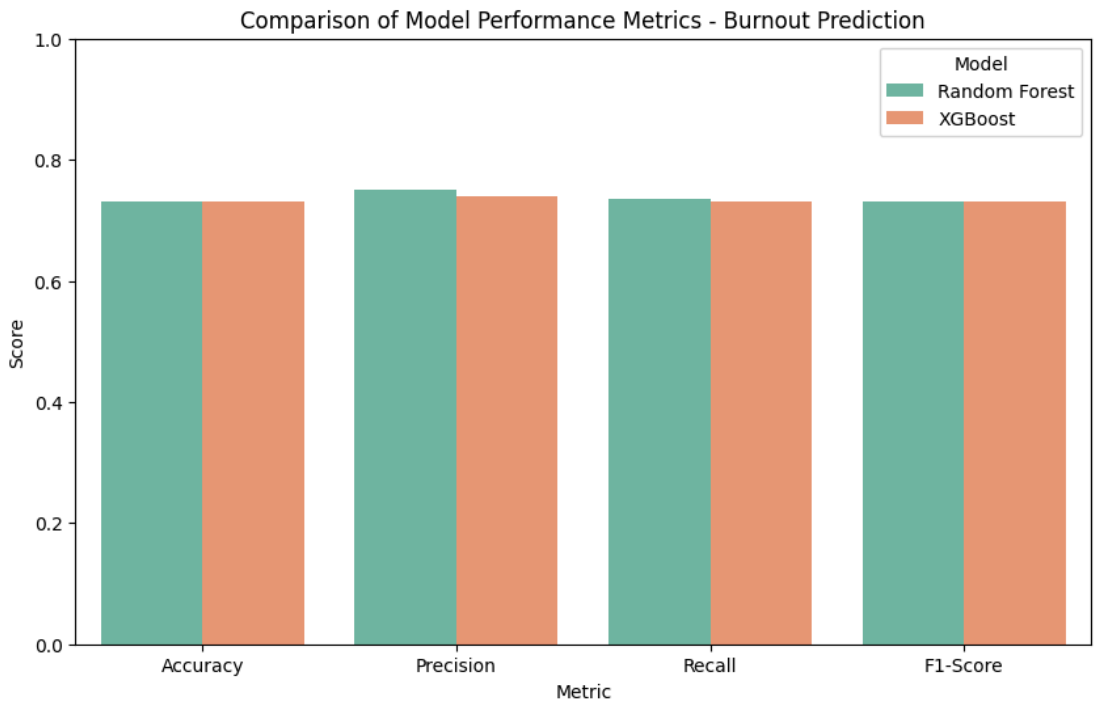

Figure 48: Metrics comparison for burnout prediction

The classification reports described in Table 6 and Table 7 conclude that both models can predict the burnout variable with the same accuracy, namely 73%. With a deep and detailed look in Figure 48, it seems that both models succeed in predicting the burnout variable with almost the same metrics.

7.5.1 The most impactful features of burnout

Having a clear overview of which features can lead to this stressful and exhausting situation is of high importance. This could conclude to some very useful and actionable insights regarding the overall work satisfaction and quality of life. The two models use different ways to calculate the most affecting features, therefore the results are different across them. One important difference between these two algorithms that lead to different important features is the fact that Random Forest use Gini index in order to calculate feature importance, while XGBoost is based on gain.

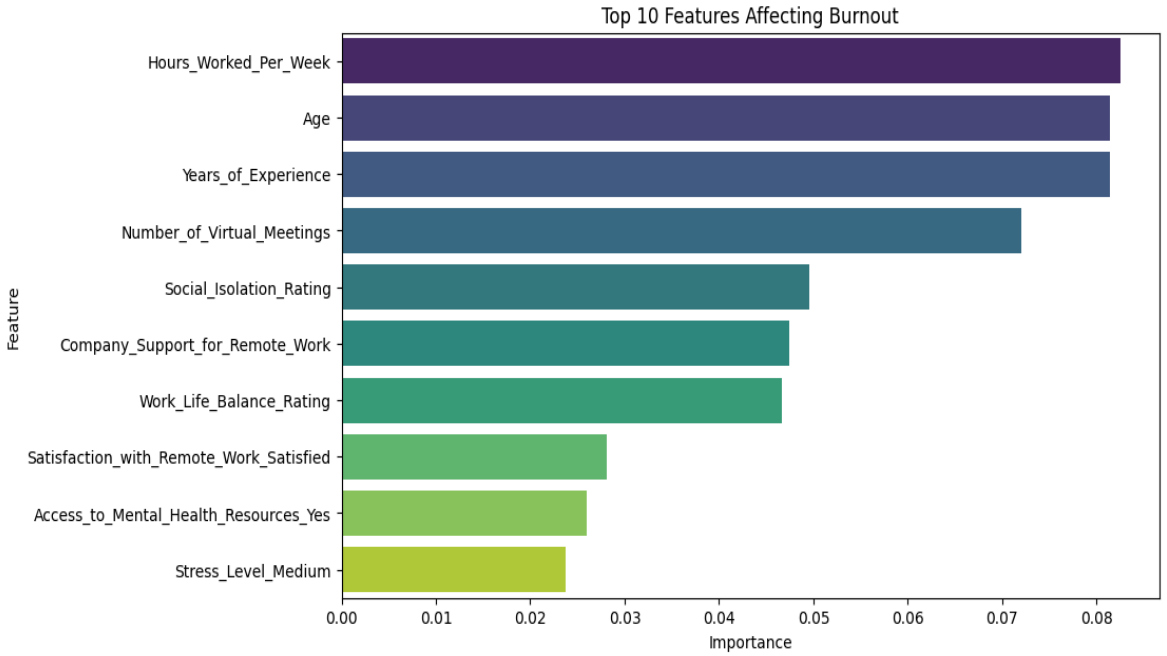

Figure 49: Features affecting burnout based on Random Forest

In Figure 49 are visualized the most significant features that can lead to burnout based on Random Forest algorithm. As represented in Figure how many hours people work per week, as well as, the age, the years of experience and the number of virtual meetings can lead people to feel exhausted and affect their overall quality of life. In respect with the above, people that tend to work more than expected is more likely to end up experiencing a burnout during their work life. This is obviously also related to the age, years of experience and the number of meetings that they attend. Employees with more work experience and — inevitably these are the older employees — tend to be more valuable in their company, meaning they have more workload and more responsibilities. This situation can lead to a very stressful occasion, where employees can hit their limits and end up with a burnout. In contrast, observing the figure, it is obvious that people with access to mental health resources and medium stress level are less likely to experience burnout.

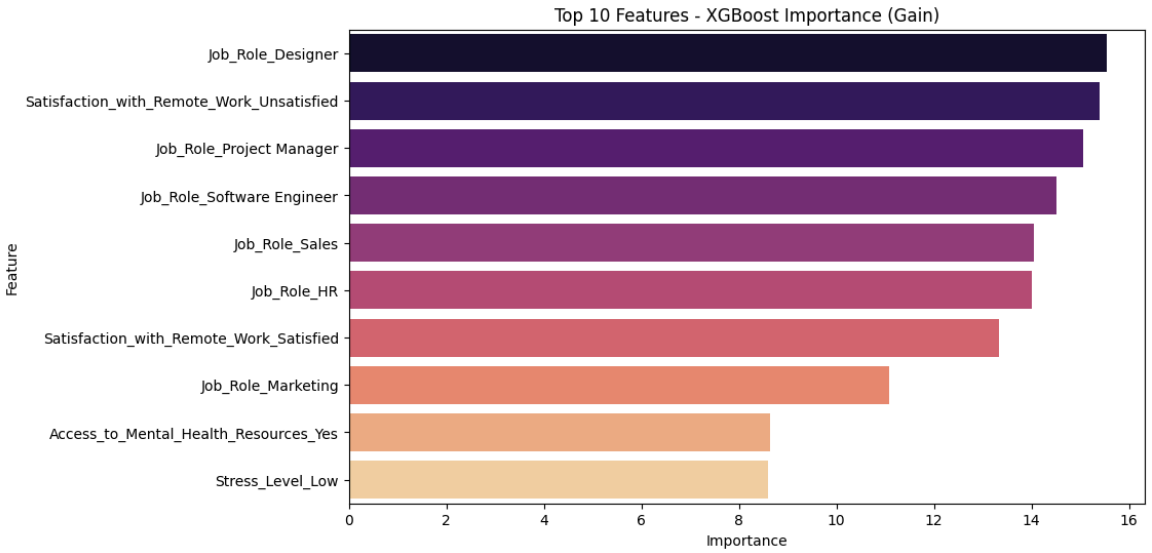

Figure 50: Features affecting burnout based on XGBoost

Based on XGBoost model people that are working in the fields of Designing, Project management, software engineering as well as people that are unsatisfied with their remote work are more prone to having burnout symptoms. As it is presented in Figure 49 and Figure 50 there are some common features that are of low importance for burnout in both models. Participants who feel satisfied with their remote work and have access to mental health resources are likely to have stressless work experience. In addition to this, low stress levels contribute to a healthier and more balanced life.

8. Wearable Stress and Affect Detection (WESAD) dataset: a special case

The Wearable Stress and Affect Detection (WESAD) dataset is a multimodal dataset that is openly accessible and intended for use in affective computing and stress detection research. Using wearable technology worn on the wrist and chest, it was gathered from 15 participants in a controlled laboratory study. The dataset is especially well-suited for research on stress and affect detection because it includes self-reported emotional states in addition to synchronized physiological and motion signals. The participants underwent several conditions: **baseline**, **stress (Trier Social Stress Test)**, **amusement**, and **meditation**. Two wearables were used simultaneously. Their description is provided below:

1. **RespiBAN Professional** (chest-worn) sampled at 700 Hz, providing:
    - 3-axis **accelerometer** (ACC)
    - **Electrocardiogram** (ECG)
    - **Electrodermal activity** (EDA)

- **Electromyogram** (EMG)
- **Respiration** (RESP)
- **Temperature** (TEMP)

2. **Empatica E4** (wrist-worn) with modality-specific sampling rates:
   - 3-axis **accelerometer** (32 Hz)
   - **Blood volume pulse (BVP)** (64 Hz)
   - **Electrodermal activity** (4 Hz)
   - **Temperature** (4 Hz)

Multimodal Physiological Signals:

- **Electrodermal Activity (EDA)** – measures skin conductance, indicative of sympathetic nervous system activity.
- **Blood Volume Pulse (BVP)** – derived from photoplethysmography to monitor heart rate.
- **Electrocardiogram (ECG)** – captures cardiac electrical activity for heart rate variability analysis.
- **Electromyogram (EMG)** – measures muscle activity.
- **Respiration (RESP)** – monitors breathing patterns.
- **Body Temperature (TEMP)** – captures changes in skin temperature.
- **Three-axis Acceleration (ACC)** – wrist and chest motion data.

The raw data from the two devices was manually synchronized using a double-tap signal pattern. The result is provided in subject-specific synchronized files (SX.pkl). Each .pkl file contains:

- **subject**: the subject identifier
- **signal**: a dictionary with two keys
  - **chest**: chest-worn RespiBAN signals (ACC, ECG, EDA, EMG, RESP, TEMP)
  - **wrist**: wrist-worn Empatica E4 signals (ACC, BVP, EDA, TEMP)
- **label**: protocol condition labels sampled at 700 Hz
  - 0 = not defined / transient (ignored)
  - 1 = baseline
  - 2 = stress

- 3 = amusement
- 4 = meditation
- 5/6/7 = other conditions (not used)

For this report the synchronized data that are included in .pkl files were used. These data were manually synchronizing the two devices' raw data. The synchronized data is appropriate for multimodal feature extraction and classification since it guarantees that features derived from the wrist and the chest correspond to the same temporal segments.

All signals were synchronized and stored in subject-specific .pkl files, allowing for multimodal analysis. In order to perform the machine learning models on these data, sliding-window feature extraction with a window size of 128 samples and a step of 64 samples was conducted. Only windows that had a dominant label that matched baseline or stress conditions are used and covered at least 80% of the window.

```
DATASET_PATH = r"C:\Users\demy2\Desktop\python projects\nurses\dataset"
SUBJECTS = ["S2", "S3", "S4", "S5", "S6", "S7", "S8", "S9", "S10", "S11",
            "S13", "S14", "S15", "S16", "S17"]
WINDOW_SIZE = 128
STEP = 64
LABELS_TO_USE = [1, 2]  # baseline=1, stress=2
```

Figure 51: Window size and step initialization

- **Wrist:** mean, standard deviation, minimum, and maximum for ACC (x, y, z), EDA, BVP, and TEMP.
- **Chest:** mean, standard deviation, minimum, and maximum for ECG, EMG, and respiration.
- **Derived HR & IBI features:** mean, std, min, max of heart rate and inter-beat intervals (when available).

```python
# ---------------- FEATURE EXTRACTION ----------------
def extract_features_from_subject(subject_path):
    with open(subject_path, "rb") as f:
        data = pickle.load(f, encoding="latin1")

    wrist = data["signal"]["wrist"]
    chest = data["signal"]["chest"]
    labels = data["label"]

    # Wrist signals
    eda = wrist["EDA"]
    temp = wrist["TEMP"]
    bvp = wrist["BVP"]
    acc = wrist["ACC"]  # Nx3

    # Chest signals
    ecg = chest["ECG"]  # HRV features
    hr = chest["Resp"]  # respiration
    emg = chest["EMG"]

    # Optional derived HR/IBI
    ibi = chest.get("IBI", None)
    hr_sig = chest.get("HR", None)

    # Downsample labels to match EDA
    downsample_factor = len(labels) // len(eda)
    label_ds = labels[::downsample_factor]
```

Figure 52: Load and align signals

```python
for i in range(0, min_len - WINDOW_SIZE, STEP):
    window_label = label_ds[i:i+WINDOW_SIZE]
    counts = Counter(window_label)
    dominant_label, dominant_count = counts.most_common(1)[0]

    if dominant_count < 0.8 * WINDOW_SIZE:  # purity threshold
        continue
    if dominant_label not in LABELS_TO_USE:
        continue

    # Wrist windows
    eda_win = eda[i:i+WINDOW_SIZE]
    temp_win = temp[i:i+WINDOW_SIZE]
    bvp_win = bvp[i:i+WINDOW_SIZE]
    acc_win = acc[i:i+WINDOW_SIZE]

    # Chest windows
    ecg_win = ecg[i:i+WINDOW_SIZE, 0] if len(ecg) >= i+WINDOW_SIZE else np.zeros(WINDOW_SIZE)
    emg_win = emg[i:i+WINDOW_SIZE, 0] if len(emg) >= i+WINDOW_SIZE else np.zeros(WINDOW_SIZE)
    hr_win = hr[i:i+WINDOW_SIZE, 0] if len(hr) >= i+WINDOW_SIZE else np.zeros(WINDOW_SIZE)

    # HR & IBI features
    hr_features = [np.mean(hr_sig), np.std(hr_sig), np.max(hr_sig), np.min(hr_sig)] if hr_sig is not None else [0]*4
    ibi_features = [np.mean(ibi[:,1]), np.std(ibi[:,1]), np.max(ibi[:,1]), np.min(ibi[:,1])] if ibi is not None and len(ibi) > 0 else [0
```

Figure 53: Feature extraction (part 1)

```
    # Feature vector
    features = [
        np.mean(eda_win), np.std(eda_win), np.max(eda_win), np.min(eda_win),
        np.mean(temp_win), np.std(temp_win), np.max(temp_win), np.min(temp_win),
        np.mean(bvp_win), np.std(bvp_win), np.max(bvp_win), np.min(bvp_win),
        np.mean(acc_win[:,0]), np.std(acc_win[:,0]), np.max(acc_win[:,0]), np.min(acc_win[:,0]),
        np.mean(acc_win[:,1]), np.std(acc_win[:,1]), np.max(acc_win[:,1]), np.min(acc_win[:,1]),
        np.mean(acc_win[:,2]), np.std(acc_win[:,2]), np.max(acc_win[:,2]), np.min(acc_win[:,2]),
        np.mean(ecg_win), np.std(ecg_win), np.max(ecg_win), np.min(ecg_win),
        np.mean(emg_win), np.std(emg_win), np.max(emg_win), np.min(emg_win),
        np.mean(hr_win), np.std(hr_win), np.max(hr_win), np.min(hr_win),
    ] + hr_features + ibi_features

    X.append(features)
    y.append(1 if dominant_label == 2 else 0)  # stress=1, baseline=0

return np.array(X), np.array(y)
```

Figure 54: Feature extraction (part 2)

The next step is to carry out the classification. In order to perform the classification a **Leave-One-Subject-Out (LOSO) cross-validation** scheme is utilized. One test set was used for each subject, and the remaining subjects were used for training. **LOSO** is used in order to ensure that evaluation is independent of the subject. Moreover, this ensures the model's ability to generalize better to new people's stress detection and avoids overfitting to individual patterns. Then two machine learning models were performed : **Random Forest** and **Support Vector Machine (SVM).** First metrics for each subject were calculated and at the end the average metrics (classification report) across all subject were calculated to provide a reliable indicator of each model's performance.

```
# ---------------- LOSO CROSS-VALIDATION ----------------
all_y_test_rf, all_y_pred_rf = [], []
all_y_test_svm, all_y_pred_svm = [], []

for test_subj in SUBJECTS:
    X_train, y_train, X_test, y_test = [], [], [], []

    for subj in SUBJECTS:
        subj_path = os.path.join(DATASET_PATH, subj, f"{subj}.pkl")
        if not os.path.exists(subj_path):
            continue
        X, y = extract_features_from_subject(subj_path)
        print(f"{subj}: {X.shape[0]} windows, stress={np.sum(y)}, baseline={np.sum(y==0)}")

        if subj == test_subj:
            X_test.append(X)
            y_test.append(y)
        else:
            X_train.append(X)
            y_train.append(y)

    if len(X_test) == 0 or len(X_train) == 0:
        continue

    X_train = np.vstack(X_train)
    y_train = np.concatenate(y_train)
    X_test = np.vstack(X_test)
    y_test = np.concatenate(y_test)
```

Figure 55: Code implementation for loso cross validation

```
    # Random Forest
    rf_clf = RandomForestClassifier(n_estimators=200, random_state=42)
    rf_clf.fit(X_train, y_train)
    y_pred_rf = rf_clf.predict(X_test)
    all_y_test_rf.extend(y_test)
    all_y_pred_rf.extend(y_pred_rf)

    # SVM
    scaler = StandardScaler()
    X_train_scaled = scaler.fit_transform(X_train)
    X_test_scaled = scaler.transform(X_test)
    svm_clf = SVC(kernel='rbf', C=1.0, gamma='scale')
    svm_clf.fit(X_train_scaled, y_train)
    y_pred_svm = svm_clf.predict(X_test_scaled)
    all_y_test_svm.extend(y_test)
    all_y_pred_svm.extend(y_pred_svm)

    # Per-subject accuracy
    acc_rf = accuracy_score(y_test, y_pred_rf)
    acc_svm = accuracy_score(y_test, y_pred_svm)
    print(f"{test_subj} test accuracy -> RF: {acc_rf:.3f}, SVM: {acc_svm:.3f}\n")

# ---------------- AVERAGE REPORT ----------------
print("\n✅ Average Random Forest Classification Report:")
print(classification_report(all_y_test_rf, all_y_pred_rf, target_names=['baseline','stress']))

print("\n✅ Average SVM Classification Report:")
print(classification_report(all_y_test_svm, all_y_pred_svm, target_names=['baseline','stress']))
```

Figure 56: Code implementation for classifiers and reports

An overview of the pipeline that is followed is provided in the diagram Figure 57 below:

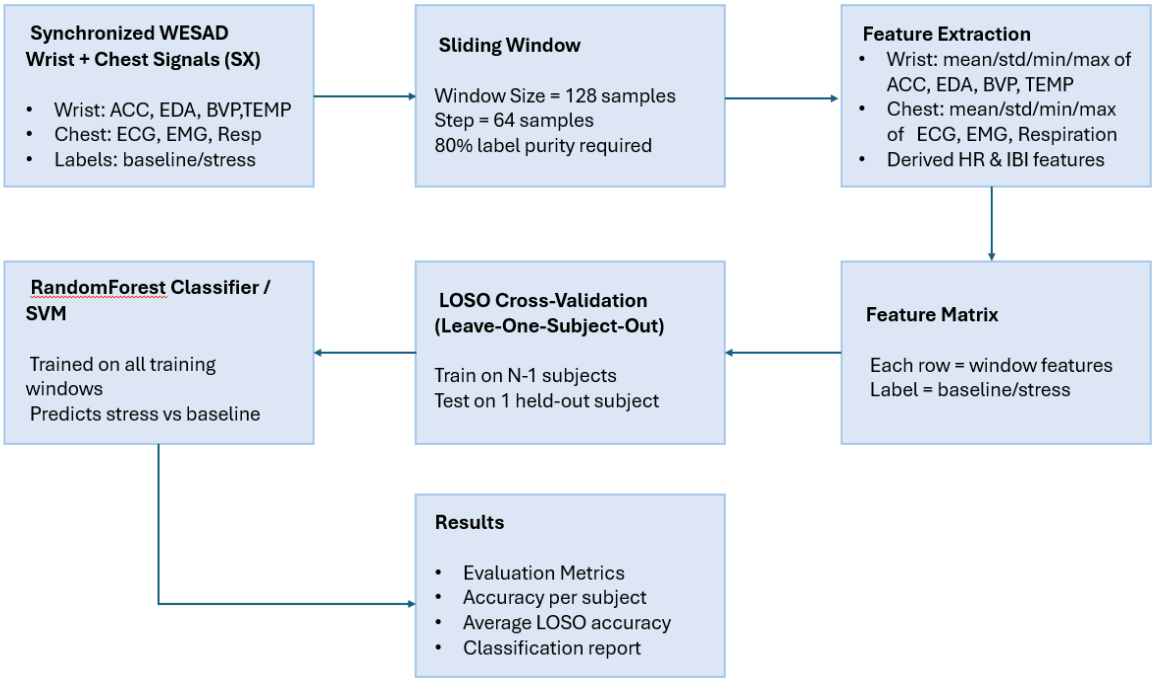

Figure 57 Preprocessing pipeline

The predictive models that were used are Random Forest and Support Vector Machine (SVM). Random Forest Classifier is already described in 7.3.1 Random Forest Classifier Chapter.

8.1 Support Vector Machines

Support Vector Machines (SVM) are a supervised learning algorithm used for classification and regression tasks. SVM aims to find a hyperplane that best separates data points of different classes in a high-dimensional feature space. The optimal hyperplane maximizes the margin. The larger the margin the better the model performs on new and unseen data.

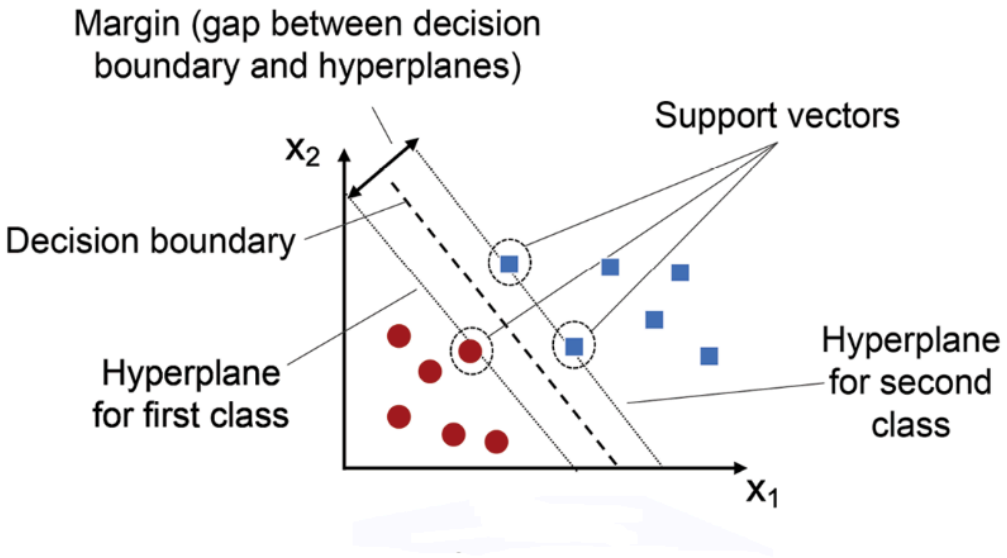

Figure 58: SVM overview

Below in Figure 59 the steps of SVM algorithm are demonstrated.

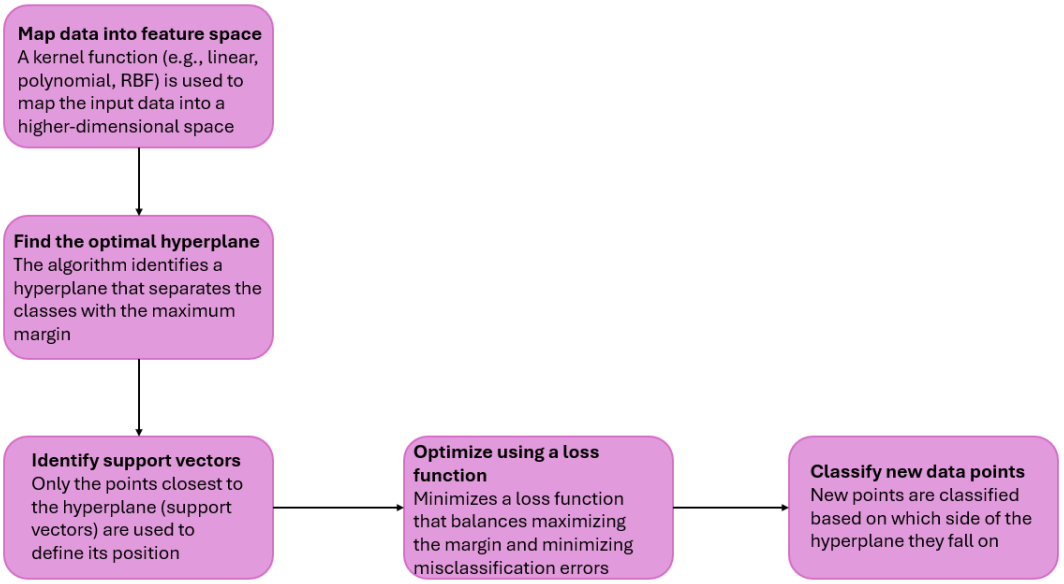

Figure 59: SVM algorithm description

A significant advantage of SVM is that it can ignore easily the outliners which results in greater robustness in anomaly detection. Moreover, it is suitable for both binary and

multiclass classification. In addition to this, it uses kernel functions like RBF and polynomial. This means that SVM can effectively handle both linear and nonlinear relationships. Finally, it is more memory efficient than other algorithms.

Although, it should be taken into account that SVM is sensitive to the scale of input features. To overcome this obstacle and ensure that all features are normalized, meaning all the features have zero mean and unit variance, for our case the StandardScaler was used. Using StandardScaler guarantees the performance stability of the algorithm. Also, the SVM model was initialized with the RBF (Radial Basis Function) kernel, C=1.0, and gamma='scale'. The C parameter aims to control the trade-off between maximizing the margin and minimizing classification errors. Meanwhile, gamma parameters used the value scale and describe the influence of individual training points.

In the following tables(Table 8,Table 9,Table 10,Table 11) the number of sliding windows per subject , as well as the test accuracy for each subject and the classification report with average metrics for each model are provided.

Subject	Total windows	Baseline (label=0)	Stress (label=1)
S2	108	37	71
S3	109	39	70
S4	109	38	71
S5	112	39	73
S6	112	39	73
S7	111	38	73
S8	113	41	72
S9	111	39	72
S10	116	44	72
S11	115	42	73
S13	112	40	72
S14	113	41	72
S15	113	41	72
S16	114	41	73
S17	116	44	72

Table 8: Number of Sliding Windows per Subject and Class Distribution

Subject	Random Forest	Support Vector Machine
S2	0.796	0.685
S3	0.688	0.560
S4	1.000	0.688
S5	0.938	0.857
S6	0.768	0.786
S7	0.829	0.586
S8	0.708	0.628
S9	0.910	0.685
S10	0.974	0.966

Subject	Random Forest	Support Vector Machine
S11	0.974	0.957
S13	0.893	0.777
S14	0.637	0.363
S15	1.000	0.867
S16	1.000	0.886
S17	0.578	0.629

Table 9: Test accuracy for each subject

Average Random Forest Classification Report:

	precision	recall	F1-score	support
baseline	0.87	0.89	0.88	1081
stress	0.79	0.77	0.78	603
accuracy			**0.85**	1684

Table 10: Classification report for Random Forest

Average SVM Classification Report:

	precision	recall	F1-score	support
baseline	0.76	0.85	0.80	1081
stress	0.65	0.52	0.58	603
accuracy			**0.73**	1684

Table 11: Classification report for SVM

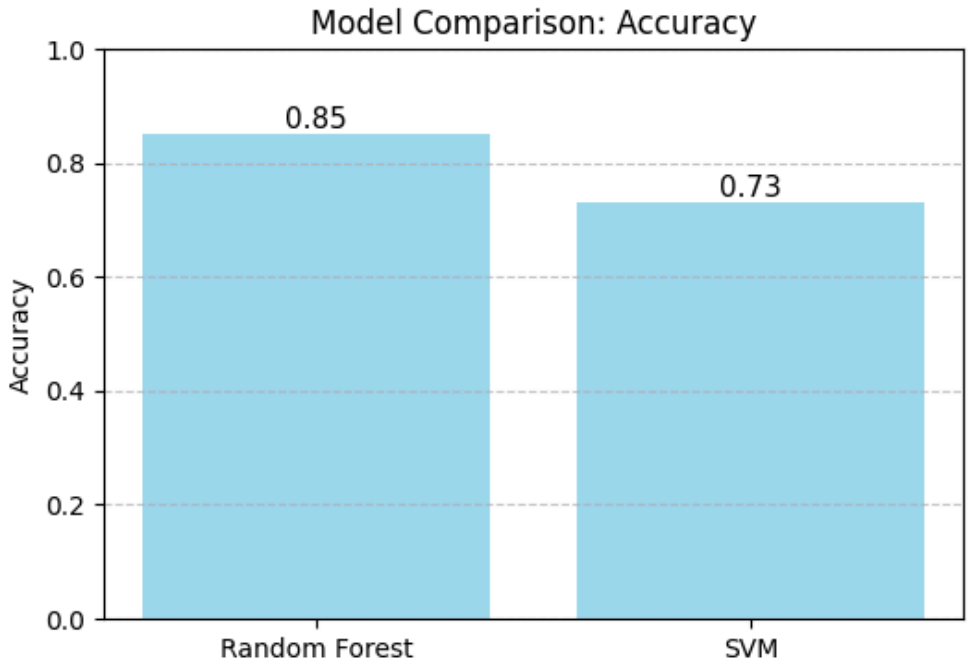

Figure 60: Model Comparison— Accuracy

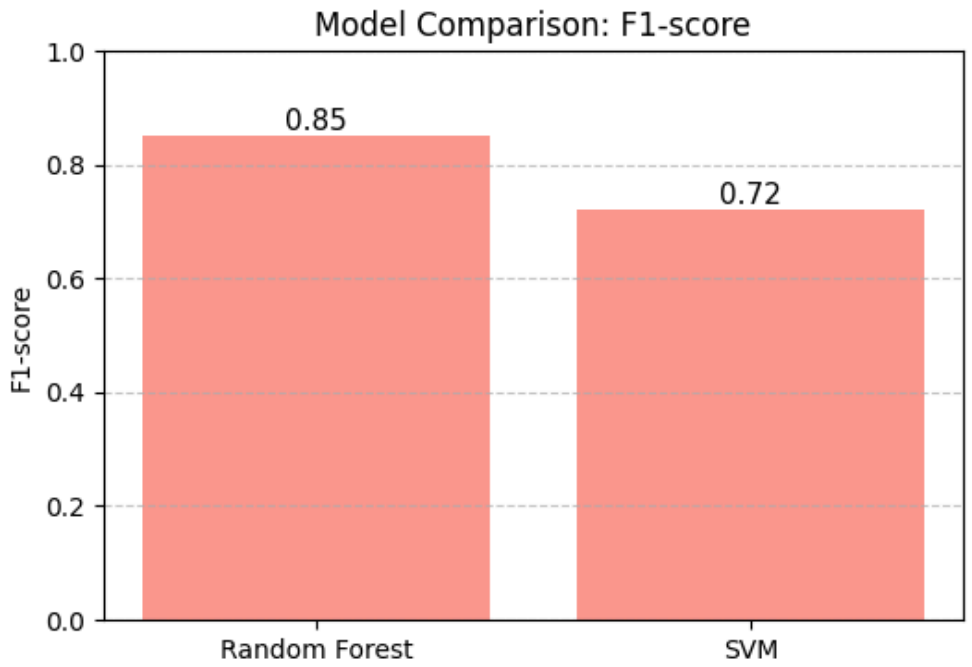

Figure 61: Model Comparison —F1-score

From the above tables, it is clear that there is a slightly higher number of baseline windows, which implies that there is a mild class imbalance in the dataset. Comparing the results from Table 10 and Table 11, Random Forest Classifier achieves better overall accuracy(85%)and more balanced F1-scores between the baseline and stress classes. SVM achieves lower accuracy (73%). This may be cause of SVM's sensitivity to class imbalance and high-dimensional feature space, especially during stress windows. All in all, Random Forest because of its ensemble nature can effectively handle complex multimodal features, which makes it appropriate for physiological stress detection.

9. Conclusion

In modern society a constantly increasing percentage of people suffer from various stressful conditions in their work life. High stress levels and the risk of experiencing burnout during adult's life due to pressuring work conditions is a serious concern that should be taken into consideration in order to improve employees' life quality and provide them work life balance. The machine learning models that are used in the current study aim to predict effectively high stress levels and burnout. Both models that are used, Random forest and XGBoost, provide a quite satisfied accuracy. There are some limitations due to noisy data, heterogeneous and small sample and

imbalanced classes. In addition to this predicting psychological data like stress level and mental health condition is complex and unpredictable. Many external factors impact human behavior and their emotional condition, that makes it difficult for the models to achieve high accuracy. Around 70% accuracy and precision are considered a reasonable result for this kind of data. The above predictions can provide meaningful insights and a good overview for the companies to predict high risk employees that may suffer from some stressful condition in the workplace, but in any case, they should be considered just as some good indicators. Taken into consideration the most significant features that impact on people's mental health should assist as indicator to improving company's culture and have satisfied employees.

As far as stress detection using data from wearables, it is obvious that Random Forest provides a significant better prediction for stress classification compared to Support Vector Machines. Worth mentioning is the fact that both Random Forest and Support Vector Machines can classify the baseline case better than the stress case. This happens because, as mentioned also previously, stress cases are more complex and in most cases with higher variability. Namely, the fact that Random Forest can provide better classification indicates that multimodal physiological stress patterns can be detected better by non-linear models.

References

[1] I. Hungerbuehler, K. Daley, K. Cavanagh, H. G. Claro, and M. Kapps, *"Chatbot-Based Assessment of Employees' Mental Health: Design Process and Pilot Implementation,"* **JMIR Formative Research**, vol. 5, no. 4, p. e21678, Apr. 2021, doi: 10.2196/21678.

[2] A. Yorita, S. Egerton, C. Chan and N. Kubota, *"Chatbots and robots: a framework for the self-management of occupational stress,"* **ROBOMECH Journal**, vol. 10, art. no. 24, Oct. 27, 2023, doi: 10.1186/s40648-023-00261-z.

[3] G. Anmella *et al.*, **"Vickybot, a Chatbot for Anxiety-Depressive Symptoms and Work-Related Burnout in Primary Care and Health Care Professionals: Development, Feasibility, and Potential Effectiveness Studies,"** *Journal of Medical Internet Research*, vol. 25, p. e43293, Apr. 3, 2023, doi: 10.2196/43293.

[4] J. Rangondi, "Impact of Remote Work on Mental Health," *Kaggle Notebook*, Jun. 2024. [Online]. Available: https://www.kaggle.com/code/judyrangondi/impact-of-remote-work-on-mental-health

[5] Google, "Dialogflow ES Documentation," Google Cloud, 2025. [Online]. Available: https://cloud.google.com/dialogflow/docs

[6] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “SMOTE: synthetic minority over-sampling technique,” *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[7] “FastAPI documentation.” [Online]. Available: https://fastapi.tiangolo.com/

[8] E. Lazarou and T. P. Exarchos, “Predicting stress levels using physiological data: Real-time stress prediction models utilizing wearable devices,” *AIMS Neuroscience*, vol. 11, no. 2, pp. 76–102, Apr. 2024, doi: 10.3934/Neuroscience.2024006.

[9] “Scikit-learn RandomForestClassifier,” Scikit-learn.org. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[10] M. Schonlau and R. Y. Zou, “The random forest algorithm for statistical learning,” *The Stata Journal*, vol. 20, no. 1, pp. 3–29, 2020.

[11] F. Amato, S. Marrone, V. Moscato, G. Piantadosi, A. Picariello, and C. Sansone, “Chatbots meet eHealth: automatizing healthcare,” in *Proceedings of WA-IAH@AI*IA*, 2017.

[12] World Health Organization, *Mental Health at Work*, Geneva: WHO, 2022. [Online]. Available: https://www.who.int/teams/mental-health-and-substance-use/mental-health-at-work

[13] O. Orvile, “WESAD—Wearable Stress and Affect Detection Dataset,” *Kaggle Dataset*, May 2025. [Online]. Available: https://www.kaggle.com/datasets/orvile/wesad-wearable-stress-affect-detection-dataset :contentReference[oaicite:0]{index=0}

[14] P. Schmidt, A. Reiss, R. Duerichen, C. Marberger, and K. Van Laerhoven, “Introducing WESAD, a multimodal dataset for wearable stress and affect detection,” in *Proc. 20th ACM Int. Conf. Multimodal Interaction (ICMI ’18)*, New York, NY, USA, 2018, pp. 400–408, doi: 10.1145/3242969.3242985.

[15] J.-Z. Xiang, Q.-Y. Wang, Z.-B. Fang, J. A. Esquivel, and Z.-X. Su, “A multi-modal deep learning approach for stress detection using physiological signals: integrating time and frequency domain features,” *Frontiers in Physiology*, vol. 16, art. 1584299, Apr. 2025, doi: 10.3389/fphys.2025.1584299. :contentReference[oaicite:0]{index=0}