Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Øystein Bjørkeng Haugen

Predicting students performance with the use of eye tracking and facial recognition to inform design principles for IDE intervention during collaborative debugging

Master's thesis in Master of Informatics
Supervisor: Kshitij Sharma
July 2023

NTNU
Norwegian University of
Science and Technology

Øystein Bjørkeng Haugen

Predicting students performance with the use of eye tracking and facial recognition to inform design principles for IDE intervention during collaborative debugging

Master's thesis in Master of Informatics
Supervisor: Kshitij Sharma
July 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

ABSTRACT

Programming and activities related to computer science studies require complex cognitive process, involving a number of emotive and mental dimensions. Employing multiple data capturing streams has shown to glean insight in these dimensions, and multimodal data is a way to visualise and quantify the complexities of these processes. This study has replicated and built upon a previous experiment, exposing pairs of students (20 pairs) to a a debugging task in a collaborative setting, while capturing data with eye-tracking, physiological and facial data. Features have been extracted from this captured data, and further analyzed to identify and decide which of these features bears the most importance in predicting students performance while debugging in a collaborative setting. These important features have been discussed to inform future work on how to design interventions in a collaborative setting, to enhance student learning and performance.

Oppsummering

Programmering og aktiviteter som inngår i informatikk-studier krever komplekse kognitive prosesser, og involverer flere følelsesmessge og mentale dimensjoner. Ved å bruke datannsamling fra flere kilder, kan vi få innsikt i disse dimensjonene, og multimodale data er en måte å visualisere og kvantifisere kompleksiteten i disse prosessene. Denne studien har reprodusert og bygd på et tidligere master-eksperiment, der par av studenter (20 par) har blitt eksponert for en debuggingoppgave i en samarbeidssetting, i mens deres blikkmønstre, fysiologiske og ansiktsdata har blitt innsamlet. Egenskaper (features) ved disse innsamlede dataene har blitt ekstrahert, og videre analysert, for identifiserte og vurdere hvilke av disse egenskapene som bærer mest viktighet for å kunne forutse måloppnåelse i en samarbeidssetting i programmering. Disse egenskapene har blitt diskutert, med mål for å informere fremtidig arbeid, om hvordan man kan bruke egenskapene til å designe feedback i en sammarbeidsetting, for å forbedre læring, og måloppnåelse.

PREFACE

Acknowledgment

I want to thank my supervisor Kshitij Sharma, who has provided guidance and feedback through this thesis. I also want to thank everyone that participated in this experiment. Lastly, I want to extend gratitude to all my friends and family for all the support throughout the years.

CONTENTS

Abstract **i**

Preface **ii**

Contents **iv**

List of Figures **iv**

List of Tables **v**

Abbreviations **vii**

1 Introduction **1**
1.1 Motivation 1
1.2 Problem statement 1
1.3 Thesis structure 2

2 Background and related work **3**
2.1 Cognitive load 3
2.2 Multimodal data for learning analytics 4
2.3 Collaborative programming 4
2.4 Interventions in IDEs 5
2.5 Intervention in collaborative setting 7

3 Methods **9**
3.1 Experimental design 9
3.1.1 Context 9
3.1.2 Participants 10
3.1.3 Experimental setup and procedure 10
3.2 Data collection 12
3.2.1 Pretest 12
3.2.2 Wristband Data 12
3.2.3 Video 12
3.2.4 Audio data 12
3.2.5 Gaze data 13
3.2.6 Code editor/webapp 13
3.3 Data processing 13

3.3.1 Timestamping 14
3.3.2 Trimming data 14
3.3.3 Sliding windows 14
3.3.4 Debug Score 14
3.3.5 Audio measures 15
3.3.6 Wristband measures 16
3.3.7 Gaze measures 18
3.3.8 Feature extraction 26
3.3.9 Data analysis 27

4 Results 29
4.1 Performance prediction accuracy using Multimodal data 29
4.2 Performance prediction accuracy using partial Multimodal data (Early prediction) 31
4.3 Categorical Success 32
4.4 Debugging Score 34
4.4.1 Performance prediction accuracy using individual data sources 36

5 Discussion 43
5.1 Future work 46

6 Conclusions 47

References 51

Appendices: 55

A - Pretest 56

B - Debug Task 62

C - Consent form 66

A D - Information letter 70

D - Information letter 70

LIST OF FIGURES

3.1.1 Environment of the Design Lab 11
3.3.1 Skewness of saccade velocity histogram 21
3.3.2 Skewness of fixation duration histogram 21

4.1.1 30 most relevant features for debugging score 30
4.1.2 30 most relevant features for categorical success 31
4.4.1 30 most important features for all data streams for categorical success 37
4.4.2 30 most important features for all data streams for categorical success 37
4.4.3 10 most important features for physiological data 38
4.4.4 10 most important features for gaze data 39
4.4.5 10 most important features for facial data 39
4.4.6 10 most important features for physiological data 40
4.4.7 10 most important features for gaze data 41
4.4.8 10 most important features for facial data 41

LIST OF TABLES

3.3.1 Fixation attributes 19
3.3.2 Saccade attributes 20

ABBREVIATIONS

List of all abbreviations in alphabetic order:

- **HCI** Human Computer Interaction
- **IDE** Integrated Development Environment
- **LA** Learning Analytics
- **NTNU** Norwegian University of Science and Technology
- **SLR** Systematic Literature Review
- **MMD** Multimodal data
- **RF** Random Forest

CHAPTER

ONE

INTRODUCTION

This section outlines the motivation of this thesis, describes how the thesis was conducted, and clarifies which research questions this thesis aims to answer.

1.1 Motivation

"Programming is a complex learning activity that involves coordination of cognitive processes and affective states." [1]. The field of Learning Analytics and Human Computer Interaction has come a long way in using both eye-tracking, facial recognition, facial and EEG data in its research. The bulk of the research has been focused on methodology, and investigating that these tools can be used to generate key insights in students learning process in the field of Computer Science. A key concept in Learning Analytics however, is the theory of the cycle. The cycle comprises of learners generating data, which is analyzed, and used to provide metrics of visualisation. The last step in the cycle, is providing this analyzed information as feedback to the learner. Figuring out how to effectively and appropriately use the information as interventions for learning is key. Research on closing the loop in with regards to using physiological data is still a new field. In a review and call-to-action on research on the use of IDEs in learning analytics, Hundhausen et. al. compares five generally used IDEs and the data they collect. They vary widely in this respect, and interestingly, none of them collect physiological data. This may be explained with the fact that IDEs plugins are not generally focused specifically on learners and learning processes, but rather on product usability and efficiency [2]. Still, this tells us that using augmented data, such as eye-tracking, physiological and facial data to inform interventions is a comparatively overlooked field of research, and should be investigated further.

1.2 Problem statement

This thesis aims to reproduce an experiment [3] to contribute to its findings. From this experiment, insights into which metrics could be turned into useful features, and which features best predict success. The analysis will both investigate how much data is needed to make a prediction, and discuss how to use the extracted most important features to aid designing interventions for students, to increase learning and performance.

The following research questions are defined:

- **RQ1**: Which metrics are the most important for predicting student expertise in collaborative coding?
- **RQ2**: How early can these features predict student success?
- **RQ3**: How can these predictive metrics be applied for designing IDE interventions to improve expertise?

1.3 Thesis structure

This thesis provides theory grounding on the use of multimodal sensor data in a collaborative setting and related work in chapter 2. Chapter 3 describes the design of this study, how the experiment was conducted and data collected and processed for analysis. Chapter 4 presents the results of the analysis of this data, and in chapter 5, the implications of these results are discussed. The following chapter details the conclusion of this thesis 6.

CHAPTER
TWO

BACKGROUND AND RELATED WORK

This section gives theoretical background to this thesis, and justifies the thesis in relation to the academic field of this subject. Large parts of this background is taken from the pre-project to this thesis "How eye-tracking and facial recognition can be used to give interventions to learners" [4], with modifications and additions.

2.1 Cognitive load

According to a seminal work on the subject, "Cognitive load can be defined as a multidimensional construct representing the load that performing a particular task imposes on the learner's cognitive system" [5]. This work is a development on how to use cognitive load measure for transfer of information, within a theory called Cognitive Load Theory. This theory assumes a cognitive architecture consisting of a limited working memory, partly independent processors for different information input, and an unlimited long time memory, working together. Furthermore, that within the process of learning, cognitive schematic building is an essential process to consider. According to Paas, "... cognitive load is not simply considered as a by-product of the learning process but as the major factor that determines the success of an instructional intervention." [5] Furthermore, the article argues that performance is affected by cognitive over-or underload. This lends to the argument that intervention in a learning context would greatly benefit by being modified by measured cognitive load. Cognitive Load Theory defines three types of cognitive load, *intrinsic*, *extraneous*, and *germane.*

Intrinsic: Equates to the inherent difficulty imposed by the subject matter or task. This type of cognitive load is difficult to impact without changing the subject matter or task itself.

Extraneous: Cognitive load that does not originate from the task, and is not directly related to learning, but from elements outside the activity of the task. This could be considered things that distracts from or facilitates learning, such as presentation of tasks.

Germane: The load that is associated with the subjects own learning process, the creation and modification of schema.

During the process of comprehending the purpose of code blocks and methods, and thus the functionality of a program, programmers must understand how the different parts of the program interacts and works, to get a mental model of the program.

2.2 Multimodal data for learning analytics

Sharafi et. al. did a systematic literature review in 2015, compiling all relevant studies on eye-tracking in software engineering [6]. One of the first recognized studies, examined the possibilities of using eye-movements as a proxy for cognitive processes in a field where verbal reports were the norm [7]. The exploratory study found that using eye-tracking, and inferring cognitive processes from the subjects visual attention allocation was viable. They found that this way of collecting information may solve some problems that verbal reports may pose, mainly the training required to properly formulate thought-processes, and the toll concurrent reports may take on the subjects ability to focus on the task at hand. The study also underlines limitations of the field at the time, and mentions that it is difficult to infer participants mental state and cognitive processes solely using their eye movements as data [6]. This advocated by Mangaroska et. al., who states that "Programming is a complex learning activity that involves coordination of cognitive processes and affective states." [1]. This study showed that the Learning Analytics and Learning Design would benefit from using multiple modes of data, considering the multitude of processes that must be consolidated during programming [1]. Using debugging performance as a dependent variable, the study compared prediction accuracy of eight models, each using a specific combination of pre-measured expertise, log data, eye-tracking, facial data, physiological data, or all of the above. The results show that the models combining data from other sources in addition to log data and performance indicators, performed better than the model using only these data streams [1]. The models using eye-tracking data and facial data performed well, but the model combining all data streams was clearly the most accurate. The findings of each of the types of data-streams underline the importance of combining information from several measurements, as to get the full picture of cognitive processes, learning strategies, and behavior that occurs during programming.

2.3 Collaborative programming

Collaborative programming, or pair programming, is a paradigm where two or more people do programming activities on the same task, with the same tools and often on the same display. As a learning strategy, it is implied to have several benefits over other paradigms of coding, including better code quality [8, 9], enables cognitive apprenticeship [2], and increased productivity. A disadvantage of this technique is that it requires communication skills to have learning outcomes for all participants. Another facet of collaborative programming, is the location. Studies has shown [10] that remote collaborative programming or pair programming may have the same benefits co-located programming gives. This might imply that the communication between programmers, and especially non-verbal communication has less significance for success. Within both university and computer science work, remote increased in popularity, especially recently due to the COVID pandemic. Therefore, researching and developing tools that are feasible both co-located and remote is useful. Data from multimodal streams could give a deeper understanding about collaboration in conjunction with individual mental processes, and might help mitigate the loss of both verbal and non-verbal communication.

2.4 Interventions in IDEs

For this study, we will use Hundhausen et. al.'s definition of intervention, which is "...an event in which some combination of information, guidance, and feedback is shared with learners for the purpose of positively influencing the learner's behavior, attitudes, or physiological state." [2].

Hundhausen et. al. also discusses the potential IDEs has to impact learning in computing education. In this review, they argue the importance of following a general learning strategy when designing and delivering interventions [2]. The study proposes a cyclical process model for IDE-based learning analytics, which consists of 1: collecting data, 2: analyzing data, 3: design intervention, and 4: deliver intervention. In our study, we know that we will collect gaze and facial data, and analyze this. In the design stage of the process, Hundhausen splits the intervention into content, presentation, and timing [2], in basic terms with *what* to intervene with, *how* to intervene, and *when* to intervene. Using CLT as a base learning strategy, designing and delivering intervention should focus on decreasing extraneous load, and facilitating the schema-building germane load. For decreasing extraneous load, Hundhausen recommends placing information physically close to the task that is to be solved [2]. This coincides with Van Gog and Halszkas findings on spatial layout of presentation [11]. This is the case for the next recommendation, which is to reduce redundancy of information. Presenting information that already has been presented increases germane load without actually contributing to schema-building. Another important concept of learning theory Hundhausen mentions is the *locus of control* [2]. This relates very much to the emotional state of the learner. Interventions try to move the locus of control from being *external*, where the learner perceives that the situation is happening to them, and altering its outcome is outside the learners control, to being *internal*, a state where the learner has control of how the situation plays out. Hundhausen calls this *attributional feedback*, which gives the learner encouragements that their work is worthwhile and progressive [2]. The Mirroring View in Mangarotska et. al. [12] is an example of this type of feedback, showing the learners its current progress and the evolution of their learning process. At a high level, this means creating interventions that helps learners identify learning goals, encouraging them on their work and effort, and push them to keep trying if they seem struck or disheartened.

According to Hundhausen et. al. "It is important to recognize that changes in response to interventions often happen gradually, not immediately. This is because the learning analytics process is inherently cyclical: learning data leads to interventions, which lead to new learning data and additional interventions. Hence, even if initial interventions lead to no detectable changes in learner behavior or attitudes, detectable changes may well occur over a longer period of time through an iterative process of progressive refinement." [2] A study that reflects this line of thinking was conducted by Trætteberg et. al. in 2018 [13]. In this case study, the researchers examined the effect visualizations on progress and feedback had on students habits, within the context of an Object-Oriented Programming (OOP) course. The research augmented the Eclipse IDE with two plugins for data collection, and used data from the use of these, eye-gaze data and self reports to measure changes in students attitudes and habits during the semester. The plugins collected log data, such as number of times the code was run, as well as tested, and displayed them real time, providing the students with a visualization of their progress. These plugins also informs instructors on how their students are working, so that they could see if the students are following guidance, and may give corrections based upon this information.

An important aspect when deciding to give a learner feedback, or provide intervention, is timing [2]. Eye-tracking and facial recognition shows promise for painting a complete picture on a learners cognitive load and emotional state. "...there are optimal opportunities to interrupt learners; if information about one's mental activity is available, interruptions can be used effectively to guide the user to pertinent information without hindering information retention." [14]

Motivation has an impact on effort of learners, and thus should be considered when discussing systems with the ability to adjust task difficulty. Effort and cognitive load are linked, and studies suggest that a learners motivation affects the cognitive load. Both too high and too low cognitive load inhibits performance [15].

Using expertise as a variable, and measuring differences in attention allocation, strategy and eye-movements between novices and experts is a common research theme within Learning Analytics. In an experiment utilizing the Eclipse IDE, Mangaroska et. al. [12] tried to make connections between visual attention, expertise and success using participants gaze patterns. The participants were tasked with removing a number of errors from a snippet of code. The environment had several views, each representing an AOI, one of these being a Mirroring Tool, which displayed several metrics about the participants coding behaviour, and information about the status of the code as well. The experiment measured the participants attention to AOI and attention switches between the AOI, as well as code reading patterns. The results show a correlation between expertise and reading patterns, as saccades in a vertical fashion would signify higher expertise. More interestingly, the experiment showed that the participants using the mirroring tool, and acted upon the feedback they got from it, seemed to have a greater amount of success in debugging. The study also found that difference in expertise amongst the participants was reflected in their gaze data, and how they allocated their attention to different AOIs, i.e gaze patterns. This gives motivation for developing tools that takes students expertise into account, and that personal adaptation of learning will help with learning outcomes across expertise.

Even though Hundhausen argues against systems that based intervention on expertise [2], if used correctly, this metric may still be of use when designing interventions for IDEs.

The duration of a fixation may indicate the amount of mental processing required for understanding the information gazed upon. The importance of the concept or the difficulty of understanding could explain higher gaze durations, and could be useful in designing systems for intervention. In the case of using experts gaze data to provide intervention for novice learners, the areas attended to longer, and with higher cognitive load could be used for interventions that hint to where a novice should aim their gaze. Conversely, using experts gaze patterns could help decide which level of cognitive load is appropriate for the given AOI or concept, and could merit intervention in the form of questions to the learner, if the system notices that the novice is not applying enough mental effort.

2.5 Intervention in collaborative setting

The study this thesis builds upon [3], presented as motivation the lack of an existing system that could collect synchronized data. There was then a research gap in collaborative systems that in realtime collected physiological data. This research gap extends to the motivation of this thesis, as there is little research done on providing intervention to students in a collaborative setting.

CHAPTER

THREE

METHODS

This section describes the methodical approach of the study, the system used for the experiment, how the system collects data, how this data is processed, and lastly how features for analysis were extracted.

3.1 Experimental design

This section describes the context and implementation of the experiment done in this study, which data were collected, and how the data was processed and analysed. It then describes how the research was designed and implemented.

3.1.1 Context

This study aims to analyze quantitative data to engineer features that can be used to inform the design of interventions for IDEs. Therefore, a formal experiment was set up to collect multimodal data from participants. This study is inspired by and builds on an earlier master thesis [3]. The thesis conducted an experiment capturing multimodal data from students participating in a collaborative programming task. The system described in this earlier master thesis was used as a tool to collect data for this current study. More details about the system and its use in the experiment can be found in section 3.2. From this point on in this text, the system implemented in this earlier master thesis will be referred to as *the system*, or in some specific cases, *the client*.

The research portion of this thesis will largely follow the same approach to collecting datasets.

A common disadvantage of laboratorial experiments are that they are not comparable to real-world settings[16]. In this case, sans the different sensors, the experiment is almost identical to a real world setting.

One of the objectives of this thesis is to supply more data points to further engineer features and attempt to improve upon the predictive results of the aforementioned master thesis. This means that the data collected from the previous master will be used for feature engineering in addition to new data collected. How the experiment was conducted, and the data collected are described as follows.

3.1.2 Participants

In total, the experiment had 40 participants, where 29 were male and 11 were female. This study's results and eventual findings were specifically aimed to be used by and for students. Generalizability was, therefore, not a consideration while choosing participants for the experiment. Given that representativity was not deemed an essential factor, participants were sampled by convenience. The participants were from the author's network and were contracted in person or via social media. Additional participants were recruited through interest forms published through message boards of organizations the author participated in, like the university, former employment, and volunteer organizations. At last, some participants were recruited through university staff working on similar research. The eligibility criteria of the population from where the sample was chosen was only the knowledge and skill that would be natural to assume a person had after completing an introductory Object Oriented programming course or comparable programming skill. No exclusion criteria were used. Most of the sample were students from NTNU (36), and four of the participants were professional software developers or scientists working for NTNU. The participants were rewarded with a gift card valued at Kr 200 for participating in the study. Additionally, two participants were rewarded with a gift card valued at Kr 500. The winners of these rewards were drawn by raffle.

3.1.3 Experimental setup and procedure

Due to limited time, and miscommunication, the experiment was conducted in two different lab settings, the UX lab and the Design lab. Firstly, the participants were welcomed to the lab setting; then, they were given a release form stating that they agreed to collect, store, store, and use their data for this master's and further research. The participants were then given an individual pretest. The pretest consisted of ten blocks of code written on it, given to the participants before the programming task, where the participants had 10 minutes to complete the test. The objective of the pretest was to gauge code comprehension, where the participants would answer what they thought the output of the code blocks would be or what the code blocks did, either through multiple choice or open answer. After the 10 minutes were up, the participants were guided to each computer. In the UX-lab setup, the computers faced one another, but in the Design-lab, the computers faced each other. In both instances, the setup was used to mimic remote collaboration. The different setups can be seen in 3.1.1.

When seated, the participants were given a run-through over how the system worked, where relevant artifacts in the code editor were placed, how to run the game, where to look for output, as well as the intricacies of the system. They were then presented with the problem they were about to solve, a preview of how the game should work, including specific functions. They were told that their task was to identify and rectify an unknown number of bugs to get the game to run as described. The pairs were also informed on how long they had to complete the task and that they could ask questions about the system, which would not lead to point reduction, and questions about the code in case they got stuck, which would lead to point reduction.

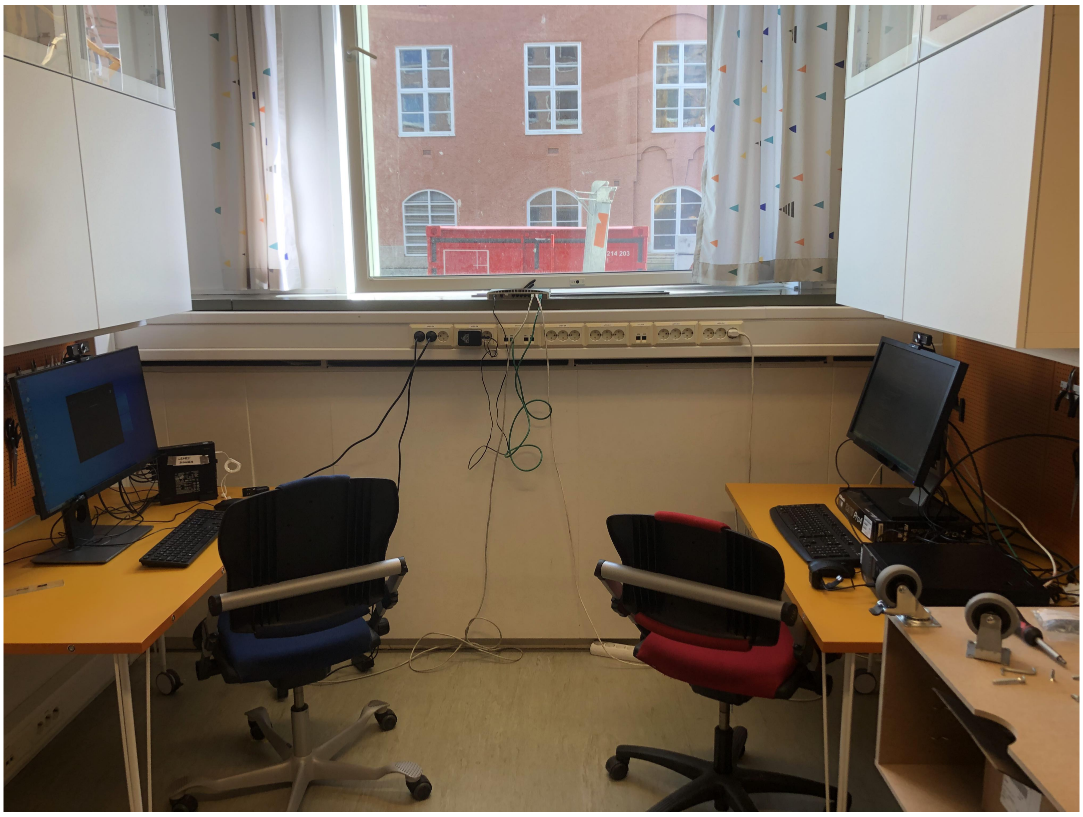


Figure 3.1.1: Environment of the Design Lab

After this, the participants were helped to put on their Empatica E4 wristbands [17], on their non-dominant hand, as per the wristbands user manual. The participants were helped with the fixation of the wristband to prevent the wristband being put in a position that might affect the data collection. As described in section 6, the usage of the wristbands had to be adjusted after the first three pairs. For the first pairs, after fixating the wristband, the wristbands were connected to the streaming server of the system. For the subsequent pairs, the wristbands were connected to their own phone. After this, the Tobii X3-120 [18] eye-tracker was calibrated for each participant, using a 7-point calibration with the Tobii Eye Tracker Manager [19]. The height and tilt of the monitor were adjusted according to each participant and the chair placement. The web camera was adjusted according to this tilting.

Finally, scripts starting the game server, eye-tracking server, and manually starting the wristband recording were commenced. The task was limited to 25 minutes, with a trigger-activated at the experiment's start, which automatically ended the test. Pairs that ended before were told to click on the button of the E4 wristband to mark the end of the session. The pairs spent an average of 24(SD = 2 minutes 2 seconds) minutes on the task.

The collaborative programming task will from here on out be referred to as *debug task*. The pre-test is listed in appendix A, the debug task is listed in appendix B.

3.2 Data collection

The data used in this study were collected quantitatively through the sensors described in the experiment setup 3.1. To sum up, data was collected through the pretest, biometric data from the wristbands and eye-trackers, log and click data from the server and code editor, facial and audio data from the server, and measurements from the completed code from the participants. The measurements associated with the collected data are a mix of pre and post-computed and are further explained in the subsequent paragraphs.

3.2.1 Pretest

The pre-test, found in Appendix A, was conducted on paper. The total score was the predetermined measurement. Each pre-test had a time limit of 10 minutes and consisted of 10 questions, each worth one point.

3.2.2 Wristband Data

As mentioned in 6, using the wristbands presented considerable difficulty. The original plan was to collect wristband sensor data through the client of the system described in the appendix. Instead, the wristband data was recorded to two cell phones, stored, and pushed to the Empatica developer client [20]. The wristbands collected six measurements. These measurements are BVP (blood volume pulse), sampled at 64 Hz, EDA (electrodermal activity), sampled at 4 Hz; TEMP (body temperature), sampled at 4 Hz; and IBI (interbeat interval), which was sampled at an irregular interval since it is based on measures from BVP. More on how the processing of this data differed from the planned processing will be in 3.3.6. All the mentioned measurements were associated with predefined measures and were used to compute post-processed measurements. These post-processed measurements include computed stress from body temperature, computed entertainment from heart rate, computed arousal, and computed engagement from electrodermal activity. These measurements are described further in section 3.3.6.

3.2.3 Video

The system used a JavaScript API face-api [21] to process the video in realtime. Therefore, the video was never recorded, only measures of facial landmarks with coordinates, and how much these facial landmarks correlated to predefined emotions, (1) neutral, (2) happy, (3) sad, (4) angry, (5) fearful, (6) disgusted, and (7) surprised. The video was sampled at 1hz, and the client processed the datastream with a timestamp per sample.

3.2.4 Audio data

Audio were recorded from each machines webcam. Relevant measures from this data point were all post-processed. The measurements needed were not what was being said but rather if and when something was said. Thus, the measurement was speech for each participant; speech overlaps between participants and silence.

3.2.5 Gaze data

Two Tobii X3-120 were used to collect gaze data. The data was collected at 120Hz, with a 7-point calibration pre-data capture. The data streams were captured through the client described in [3]. Gaze data collected were pupil diameter and coordinates of gaze points for both pupils. All measures used in the analysis were post-computed. These measurements include cognitive load, information processing index, perceived difficulty, and measured anticipation. In addition, aggregated measures were computed, including the number of saccades, number of fixations, ratio of saccades to fixations, mean durations and velocity, dispersion, and more. These measurements will be discussed further in section 3.3.7.

3.2.6 Code editor/webapp

As described in the [3], the client logged every event within the editor. This data of writing, removing, and replacing text, was logged in real-time to a Firebase real-time database. The client logged the event, a timestamp for the event, and the ID of the user performing the action. As the scope and objective of this study changed, as described in 3.1.1, a lot of the specific data collected by this stream ended up not being necessary for this study and ended up not being further used. The data logged in the real-time database from the editor has implications for the analysis. It is used as the basis for timestamping the experiment for each pair of participants. In addition, the data here contains the changes in the code during the experiment and the state of the code at the end of the experiment. Timestamps were used to trim data since only the data was captured during the 25-minute period the experiment was conducted, as well as for calculating the offset between devices. The server timestamping and synchronization usage is described further in section 3.3.1. The state of the code is essential to track, as it is used to measure performance for computing completion of debugging.

3.3 Data processing

This section describes how the data collected from the experiments were pre-processed, then processed to be ready for feature extraction and analysis. The data was processed using Python [22], with the packages pandas [23] for data analysis and manipulation, numpy [24] for calculating metrics, and matplotlib [25] for visualizing during processing and analysis. The same packages were used for post-processing, with tsfresh [26] added for feature extraction. All streams described in this section went through the same pipeline to be ready for analysis. Firstly, the data were inspected and pre-processed to check for missing values and evaluate which effect this would have on further analysis. Subsequently, the cleaned primitive data were transformed into more interpretable primary measures. Once the data was transformed into primary measures, these were used to compute features, which were used for analysis and further feature extraction using tsfresh.

3.3.1 Timestamping

In order to be able to interpret the data collected by the eye tracker in the context of something else, e.g., presented stimuli or data from other sources such as EEG, the data must be synchronized in time with whatever it is going to be analyzed in conjunction with. As the data collected were done via two clients and one server, the first processing step was generating reliable timestamps for each sensor.

3.3.2 Trimming data

Since the experiment involved several sensors, and the system used for the experiment was performed on two different computers, in which both sensors, client, and server were started manually, some data were collected both before and after the experiment started. To trim this superfluous data, start and end timestamps were used. The start timestamp is from the start of the experiment, that is, when the participants clicked start in the editor, and the end timestamp is set either automatically 25 minutes after the start timestamp, or if a pair completed the debug task early, the end timestamp is set to an event click on the wristbands.

3.3.3 Sliding windows

For all data streams, the sliding window technique was employed. The sliding window technique, often used in analyzing time-series data, is a highly beneficial method for understanding the nuances of temporal patterns and sequences. With this technique, a fixed-size window—30 seconds in this case—is "slid" across the time series, with each slide shifting the window by a specified interval, which for this analysis was 5 seconds. Each 30-second window overlapped with the previous window by 25 seconds, ensuring a significant level of data continuity and thereby aiding in the detection of trends, patterns, or anomalies that may exist within the data. This approach is beneficial for time-series data as it allows for the systematic examination of distinct, continuous chunks of data within a given timeframe, facilitating the extraction of meaningful local patterns and characteristics. The sliding window technique enables a more nuanced and dynamic analysis of time-dependent changes and relationships within the data by providing an ongoing snapshot of the dataset. The Pandas package has a `.rolling()` method that was useful for applying sliding windows of the time-series data. This method was only made for computing functions over a single column or set of values, so a custom rolling function was made for computations requiring several values and variables.

3.3.4 Debug Score

As the system logged every event of the code editor, the exact state of the code at the end of the debug task could be reproduced. The debug task had five bugs to be identified and fixed, each worth five points. The state of the code at the experiment was parsed manually, and each pair of participants were given a score according to the number of bugs corrected, from one to five. The processing of this dependent variable differs slightly from the one done in [3], as the time was not considered, and the number of bugs differed from six initially to five in this experiment. Two independent variables were extracted from the debug score, one continuous from zero to five, and one categorical, labeling success as low or high, with the cutoff at < 3 bugs corrected.

3.3.5 Audio measures

For processing and computing measures from the audio files, the Python library Pydub [27] was used. As a result of not using external microphones, the audio files were recorded in .webm format, a format often used to contain media for use on the web. As it is a container for both video and audio, these files are unsuitable for use with PyDub. Therefore, the files were converted into .wav format to make this data appropriate for processing. To do this, the Python module subprocesses for automation, to access FFmpeg, a software tool for handling audio, and more. The audio streams in the webm containers were decoded and converted into processable wav format. This conversion process decompresses the raw audio data, processes it, converts it, and decompresses it again. While compressed audio loses a lot of granularity and nuance, compressing is desirable to take less space. When opening audio files for processing using PyDub, decompression automatically takes place to load audio into memory before compressing again. Additional pre-processing was done to the audio files before computing measures. As mentioned in section 3.2.4, measures pertaining to times of communication were to be computed. These measures are for when one participant is speaking, both are speaking in an interval and when none are speaking. As described in section 6, a lot of background noise and audio artifacts could make the computation less accurate. Therefore, additional pre-processing was done to the audio files. To compute the measures, silence detection had to be used. Pydub has functionality for this, shown in code listing 3.1.

```
    silence.detect_silence(
        chunk,
        silence_thresh=-40,
        min_silence_len=1000,
        seek_step=100,
    )
```

Listing 3.1: pydub.silence

The parameter silence_thresh describes the threshold for audio to be considered silence and is measured in dBFS, decibels comparative to the signal level at max. The parameter min_silence_thresh describes how long a detected silence must last to be classified as silence. This parameter was set to 1000 milliseconds not to consider slow speech as silence. A high pass filter and noise reductions were applied to the files not to have to set the silence threshold too high and lose data from this filter. As applying such processing to audio decreases the fidelity and almost always results in data loss, the high pass filter was applied first, then the noise reduction. This was to keep as much of the original data as possible by first filtering out the low-frequency components of the keyboards and mouse and then reducing noise from other sources. The high-pass filter was implemented using signal processing methods from the Python package Scipy [28], cutting out frequencies below a certain threshold. This threshold was set at 250Hz, below the human speech frequency range of 300Hz. After removing unwanted low-frequency artifacts, noise reduction was applied to the files using the Python package noise reduction. This processing allowed the silent detection method to be less discriminatory at a value of -40dBFS. After the silence and speech detection, additional confirmation of the correctness of the values was done manually in Audacity. After this signal preprocessing, the measures that were extracted from this sensor were the number of silences, speech overlap, and single participant speaking during each interval of the sliding window.

3.3.6 Wristband measures

Since the measurements of the wristbands were recorded using Empatica proprietary software, the data was additionally processed before pre-processing. In the client described in [3], timestamps were automatically added for each sample point; this was not the case using the proprietary software. The data formatted by the wristbands came with the timestamp of the beginning of the recording, expressed in UNIX time in seconds, as well as the sample rate of each measurement. To get accurate timestamps for each sample in each measurement, the beginning timestamp was taken, and for each subsequent sample point, the start time plus the time delta was expressed by dividing the second by the frequency.

$$\Delta = 1s/Hz \tag{3.1}$$

After timestamping the samples, the following measures were computed:

- Arousal
- Engagement
- Stress
- Entertainment
- BVP change

Arousal

The arousal measure was computed from the EDA samples recorded by the wristband. A simple version of the EDA Positive Change algorithm, described by Leiner et al. [29], was implemented in Python to compute arousal. The computation measures the amplitude between skin conductance values samples and determines emotional arousal levels. Arousal as a measure can tell us how emotionally engaged a student is to the presented information.

Engagement

Engagement is a complex construct that can be measured in several ways and is beneficial for inferring the effectiveness of presented learning tasks. One of the ways engagement can be measured is through the conductance of the skin, EDA. By breaking down the EDA signal into its tonic and phasic components, we can calculate three inputs that can be combined into an accurate measure of engagement. Using the phasic part of the signal and defining an onset and offset threshold to quantify peaks of the signal, the measured amplitude was computed, by summing up the highest points of the signal, every time the signal curved above the onset threshold. In addition, the measure AUC (Area under the curve) was computed using the tonic of the signal to approximate the area under the curve of the signal between two points. AUC reflects the overall magnitude of the EDA response, and combining it with the amplitude and the number of peaks of the amplitude measurement is a valuable indication of emotional engagement.

Stress

A measure of stress was derived using the sampled skin temperature from the wristbands. Skin temperature is known to be influenced by physiological responses to stress, such as increased sweating and changes in blood flow. The recorded skin temperature data over time were analyzed to identify patterns and quantify the overall temperature change. This involved fitting a mathematical model to the temperature data points, which helped estimate the rate of temperature change. In this context, a significant negative change in temperature was considered indicative of stress. The assumption was that a more significant decrease in temperature corresponded to a higher stress level. This measure provided an objective assessment of stress levels based on the physiological response captured by the wristband without requiring subjective reporting or self-assessment.

Entertainment

The measure of entertainment could be calculated through a process that evaluated multiple dimensions of heart rate data sampled from the wristband. Firstly, the mean heart rate was calculated to understand each participant's physiological response during the experiment. In addition, the variance of the hr signal was computed to assess the degree of fluctuation in heart rates. The minimum and maximum hr values were obtained to provide a range of heart rates within which the participant's response varied. The difference between these minima and maxima was calculated.

The correlation coefficient between hr samples and timestamps was calculated to ascertain the linearity of the HR signal over time and thus provide valuable insight into how the participant's heart rate evolved throughout the experiment.

Additionally, the autocorrelation with a lag equal to 1 was computed. This process aimed to discern the level of non-randomness in the HR data, providing information about any consistent patterns or rhythms present in the heart rate.

Lastly, the approximate entropy of the signal was determined, which quantified the unpredictability of fluctuations in the HR time series [30]. This provided a measure of the complexity and irregularity of the HR, offering valuable insight into how much the participant's heart rate varied in an unpredictable manner during the experiment. Together, these nine measures give an understanding of the participant's entertainment level.

Blood volume pulse

For the blood volume pulse measurement from the wristband sensor, only descriptive statistics were generated.

not computed: -Emotional regulation(IBI)

For each of these measures, describe how they were computed, and why they were computed. Explain why IBI measures has not been computed, and what emotional regulation shows

Since IBI did not have a regular interval where it was sampled, this method could not be used to properly timestamp this measure.

3.3.7 Gaze measures

To obtain measures ready for feature extraction, the gaze data was processed in several steps. The system[3] used a Python language binding to the Tobii-research SDK to subscribe to and collect data from the eye trackers. The data captured from the eye trackers are formatted according to the model of the eye tracker. In this case, the raw data extracted from the eye tracker was a timestamp per sample, gaze point coordinates, gaze origin, eye position, pupil diameter, and a validation tag. Each of these data points was for both eyes. These primitive data were then sanitized only to include values for which at least pupil measurement was deemed valid. The same was done for pupil diameter. Then, the x and y values for both values of left and right pupils were concatenated to represent one point on the screen and order the screen into a grid.

Event classification

To generate aggregated attributes from these primitives, such as fixations and saccades, and derived attributes from these, processing that classified the primitives into events needed to be done. Fixations and saccades are the two primary states of gaze. Classification of these varies from study to study, but for this study, the definitions were any eye movement exceeding a velocity threshold. This threshold is described further down in this section. The first part of the processing was to capture collective aspects related to eye movements, including the duration, distance covered, velocity of the eye movements, as well as their direction. These oculomotor parameters were then used to decide if a sampled data point belonged to the same event, that is, if it belonged to a saccade or a fixation. There are many ways to make this classification. A commonly used technique for labeling saccades and fixations is to use a velocity threshold, the I-VT algorithm [31]. This algorithm was used in a modified fashion to classify. The classification process of this algorithm is simple: if the velocity value for a sample point is below the threshold, it is classified as a fixation; if it is above, it's a saccade. For each valid sample point, this threshold comparison is made, and each consecutive sample point that falls below this threshold is collapsed into a single event, a fixation. Thus one separates and labels fixations and saccades. The I-VT algorithm that Salvucci and Goldberg proposed changed the degree of visual angle to calculate velocity. As using degrees requires knowledge of the specific setup for the experiment, and the experiment allowed the participants to move both the screen and the chair they were sitting on, calculating the velocity based on visual angle degree was decided against simply because the viewing distances differed so much per participant. The solution was to divide the distance between each sample point coordinate with the time between sample points to get pixels per millisecond. A viewing distance estimate was used to get an accurate threshold from various experimental setups. Assuming a velocity threshold of 30°/s, a commonly used threshold, and a viewing distance of 45 to 65 cm (55 cm mean), the velocity threshold for the 24-inch monitor with a 1920x1080 resolution would be represented by:

For 45 cm viewing distance: 30°/s * 60 pixels/degree = 1800 pixels/s = 1.8 pixels/ms For 65 cm viewing distance: 30°/s * 40 pixels/degree = 1200 pixels/s = 1.2 pixels/ms

So a rough threshold of the mean of these values were chosen: saccade = 1.5 pixels/ms.

Once the sampled data points were concatenated and labeled according to I-VT algorithm, the attributes shown in table 3.3.1, 3.3.2 and () were extracted.

Attribute name	**Descriptive statistic**	**Description**
Number of fixations	total	The sum of all events labeled as fixations.
Fixation dispersion	mean, median, min, max, std	Maximum distance between any two gaze points within a potential fixation.
Fixation duration	mean, median, min, max, std	The duration of each event labeled as fixation.
Fixation to saccade ratio	total	The ratio of fixations to saccade within a given time period.

Table 3.3.1: Fixation attributes

In addition to these attributes pertaining to the movement of the gaze, the pupil diameter attribute descriptive statistics was calculated in the same manner.

With the presence of these attributes, several more complex measures could be computed, which are listed in the following sections.

Skewness of saccade velocity and fixation duration histograms

From the velocity of the aggregated saccade velocities, and likewise with the aggregated durations of fixations, skewness histograms for each of these measures were generated per participant. Skewness is a model that describes the distribution of a specified parameter.

For saccade velocity, high or positive skewness means the velocity tends to be high. High saccade velocity has been argued to indicate comprehensibility in the presented material [32]. High skewness is also related to anticipatory gaze, which could indicate familiarity with the material [33]. Neither of these arguments directly translates to expertise, as comprehensibility is an artifact of the material itself, and familiarity could be true for non-experts without directly impacting performance.

Attribute name	**Descriptive statistic**	**Description**
Number of saccades	total	The sum of all events labeled as saccades.
Saccade velocity	mean, median, min, max, std	The velocity of an event labeled as a saccade, in pixels/ms.
Saccade duration	mean, median, min, max, std	The duration of each event labeled as saccade.
Saccade direction	classification	The direction of the saccade, wherein an increasing x-coordinate value equals "forward", and a decreasing one equals "backwards".
Saccade to fixation ratio	total	The ratio of saccade to fixations within a given time period.
Scanpath velocity	ratio	The ratio of forward saccades to total saccades, calculated from saccade direction.
Duration to distance ratio	ratio	Attribute used to compute information processing index.

Table 3.3.2: Saccade attributes

For fixation duration, high or positive skewness means fixation duration tends to be long, and short mean duration fixation results in a low or negative skew. High fixation skew can point both to higher concentration.

Both these skewness measures could be used to indicate familiarity and comprehensibility.

Example histograms of these measures are shown in figure 3.3.1 and figure 3.3.2.

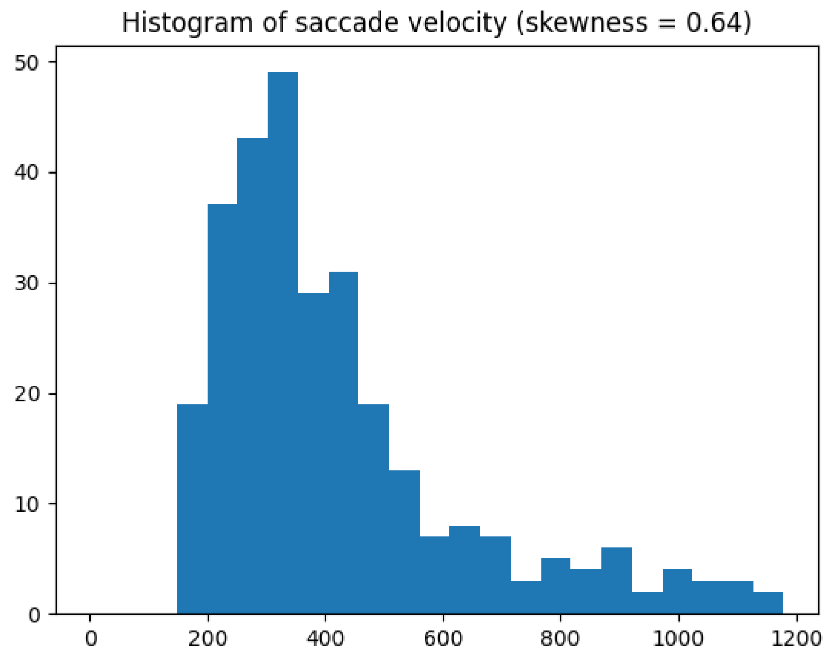


Figure 3.3.1: Skewness of saccade velocity histogram

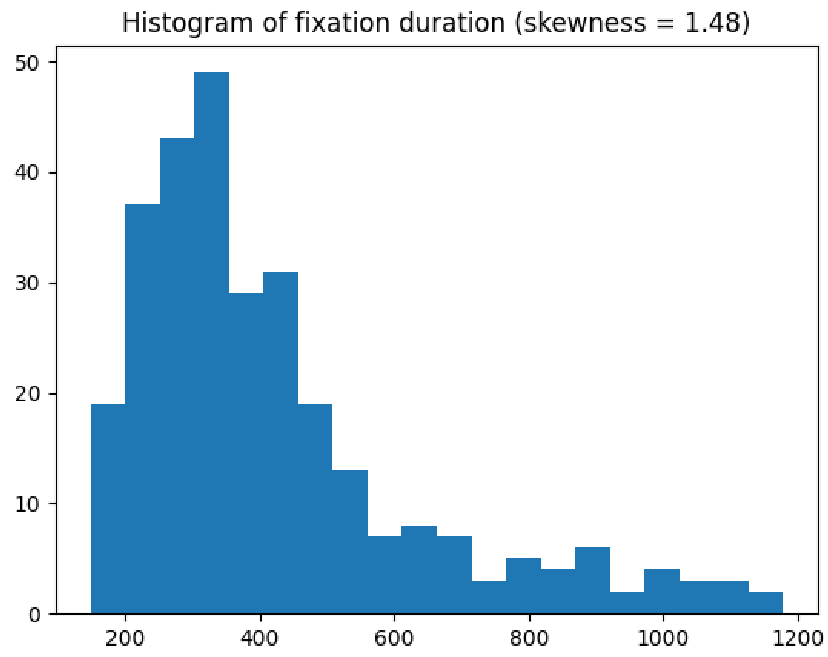


Figure 3.3.2: Skewness of fixation duration histogram

Cognitive load

Using the diameter of the pupil, a measure of cognitive load clould be derived. Like all continuous data streams, eye-tracking data contain noise in the form of head movements, off-screen eye flashes, and blinks. The first step in ensuring a valid measure for the cognitive load was to isolate and detect blinks, as no valid pupil diameter could be determined from when the eye was shut. As Hollander and Huette point out, parameters for blink extraction vary from study to study [34]. This study argues that consecutive sample loss of under 85ms is unlikely to be a blink. This lower threshold was also used by Faber et al. [35], with the same sampling rate 120Hz as this experiment. Therefore, all consecutive samples with a nonvalid gaze value that lasted at least 85ms were classified as blinks and filtered out of the sample set. To measure cognitive load, the LHIPA algorithm [36] was used, in addition to the ModMax algorithm created by many of the same authors [37]. The LHIPA algorithm was used to detect pupil diameter oscillation by analyzing the high and low frequencies of pupillary activity and outputs an approximate level value for the participant's cognitive load in the captured window. LHIPA was an open-source algorithm to another algorithm, Index of Cognitive Activity, which is proprietary, thus not open for use in research. LHIPA detects changes in cognitive load but is not shown to distinguish user-perceived difficulty. After the cognitive load was measured, aggregated attributes from pupil diameter, the mean, and the difference between the participant's cognitive load were used for further analysis.

Information processing index

From the length of the distance of the saccades and the duration of the fixations captured, tendencies in reading patterns or other types of visual scanning behavior can be inferred. Short fixations coupled with long saccades indicate skimming text, while the opposite, long fixations and short saccades, could signify deliberate focus. The metric information processing index uses both the spatial dimension of saccades and the temporal dimension of fixations to compute the ratio between the two and measure reading patterns [38]. The information processing index was calculated by finding the thresholds for what constitutes a short and long fixation and the same for a short and long saccade and then getting the ratio between long fixations and short saccades. How this code was implemented is shown in code listing 3.2.

```python
def compute_ipi_thresholds(ratios):

    short_threshold = np.percentile(np.asarray(ratios), 25)
    long_threshold = np.percentile(np.asarray(ratios), 75)

    return short_threshold, long_threshold


def compute_information_processing_index(ratios, short_threshold,
   long_threshold):

    number_of_long_f_short_s = max(np.sum(np.asarray(ratios) >
   long_threshold), 1)
    return np.sum(np.asarray(ratios) < short_threshold) /
   number_of_long_f_short_s
```

Listing 3.2: information processing index

Perceived difficulty

As mentioned in section 3.3.7, the LHIPA algorithm did not detect the participant's perceived difficulty. To provide some mitigation for this deficit, a measure of perceived difficulty was calculated. This measure was computed by averaging the length over the duration of each saccade over the total count of saccades for a given window, which gives a value of the perceived difficulty.

Anticipation

Another measure that could indicate a students familiarity with the task or problem, is anticipation. To compute this measure, the average velocities of the saccades of a given window were aggregated. Then, a measure of the difference from the average of saccade velocities were given, their variance. These two measures were then iterated through, and the cube of the differences between the velocity and the average velocity. The sign of this value is then used to give a categorical value for the classification of anticipation. Anticipation is a key factor for expertise [39], and could be used to gain insight in a students reading strategy.

An additional measure that could have been useful, is the angle of the pupils in relation to the screen. While this is a measure in it self, it could have been used to calculate other metrics. As mentioned, to calculate the velocity of a saccade, I-VT was used, with distance travelled as a variable. This computation could also be done with the change in degrees of the angle of the pupil. This measure could also have been used to measure saccade amplitude, a measure which can be used to compute several measures, most notably ratio of global to local saccades.

3.3.7.1 Final set of measures for feature extraction

Name	Sensor	Description
Silence	Audio	Number of intervals of minimum 1s where no participant were speaking per window.
One speaking	Audio	Number of intervals where one participant were speaking per window.
Both speaking	Audio	Number of intervals where both participants were speaking per window.
Arousal	Wristband	Emotional arousal levels from skin conductance, mean and difference between participants
Engagement amplitude	Wristband	Sum of peaks from phasic signal of skin conductance per window, mean and difference of participants

Engagement no. peaks	Wristband	Number of peaks from phasic signal of skin conductance per window, mean and difference between participants
Engagement AUC	Wristband	Overall magnitude of skin conductance per window, mean and difference between participants
Stress	Wristband	Indication of stress from change in skin temperature, mean and difference between participants
Entertainment	Wristband	Nine measures relating to heartrate signal dimensions, mean and difference between users
BVP	Wristband	Descriptive statistics of blood volume pulse per window, mean and difference between participants
No. fixations	Eye tracker	Number of fixations per window, mean and difference between participants
Fixation dispersion	Eye tracker	Descriptive statistics of spread of gaze points within fixations per window, mean and difference between participants
Fixation duration	Eye tracker	Descriptive statistics of fixation durations per window, mean and difference between users
Fixation to saccade ratio	Eye tracker	Ratio of fixations to saccades per window, mean and difference between participants
No. saccades	Eye tracker	Number of saccades per window, mean and difference between participants
Saccade velocity	Eye tracker	Descriptive statistics of saccade velocity per window, mean and difference between participants
Saccade duration	Eye tracker	Descriptive statistics of saccade durations per window, mean and difference between participants
Saccade distance	Eye tracker	Descriptive statistics of saccade distances per window, mean and difference between participants
Forward to total ratio	Eye tracker	Ratio of forward to total saccades per window, mean and difference between participants

Saccade to fixation ratio	Eye tracker	Ratio of saccades to fixations per window, mean and difference between participants
Anticipation	Eye tracker	Mode of measure anticipation per window, value per participant, and difference between participants
Perceived difficulty	Eye tracker	Descriptive statistics of the percieved difficulty value per window, mean and difference between participants
IPI value	Eye Tracker	Information processing index per window, mean and difference per participant
Cognitive load	Eye tracker	Computed mean cognitive load value per window, mean and difference between participants
Pupil diameter	Eye tracker	Descriptive statistics of pupil diameter per window, mean and difference between participants
Skewness of saccade velocity histogram	Eye tracker	Distribution of saccade velocity per window
Skewness of fixation duration	Eye tracker	Distribution of fixation duration per window
Neutral	FaceApi	Mean value of the neutral expression per window, mean of participants
Happy	FaceApi	Mean value of the happy expression per window, mean of participants
Sad	FaceApi	Mean value of the sad expression per window, mean of participants
Angry	FaceApi	Mean value of the angry expression per window, mean of participants
Fearful	FaceApi	Mean value of the fearful expression per window, mean of participants
Disguisted	FaceApi	Mean value of the disgusted expression per window, mean of participants
Surprised	FaceApi	Mean value of the surprised expression per window, mean of participants

The preprocessing and measure aggregation described in this section were applied to the data captured by the experiments conducted in [3] to get more data points for the analysis and increase accuracy.

Before going to the feature extraction step in the analysis, audio, and blood volume pulse measures were deemed superfluous to analysis and therefore cut out from the dataset.

3.3.8 Feature extraction

In this study, it was decided to incorporate feature extraction into the analytical pipeline after the initial data processing and computation stage. The capabilities of TSFRESH were leveraged. It is a Python library designed explicitly to extract relevant features from time-series data automatically. There were other options for extracting features from time series, like TSFEL or tsfeatures. However, there were several reasons to choose TSFRESH as a feature extraction library. This library can extract a more comprehensive list of features than the other mentioned libraries, which lends to capturing more complex patterns in the data. TSFRESH is also highly customizable, giving greater flexibility in dealing with diverse data. Lastly, the library comes with built-in mechanisms of relevance filtering, which makes feature space reduction easier, and helps avoid overfitting the model in the analysis. Despite already having a large set of variables before feature extraction, an argument for expanding upon this set was to ensure that the Random Forest model would be robust, provide accurate predictions, and leverage accurate feature importance levels. Another reason for using a library for extraction is that it saves time that would have been used on manual feature engineering, leaving more time and focus for analysis. In addition, this library is compatible with packages already used, like pandas, and easily integrates with scikitlearn [40], which was needed for the analysis.

In the context of this study, the processed sensor data from all the sources were merged into a single, stacked DataFrame. This unified dataset allows for comprehensive and integrated analysis, considering the interactions between different sensor data types. Each experiment with all its values was represented as one-time series.

As TSFRESH generated aggregate features from each value, the min, max, and median measures included from the final measure dataset were cut out to not double up and over-specify features and decrease computational time.

The specific operations are determined by the parameters supplied to the extraction function; in this case, 'MinimalFCParameters' was used. This preset configuration in TSFRESH specifies a subset of features that are relatively fast to calculate. This choice was made to balance the computational resources required for feature extraction and the richness of the resulting feature set. TSFRESH operates on the DataFrame on a window-by-window basis. Various features are based on the configured parameters for each rolling window in the time series. This rolling window approach is ideal for this dataset because it allows us to capture static and dynamic sensor readings' characteristics over time. The features extracted can include superficial statistical characteristics like mean, variance, and standard deviation, as well as more complex properties such as trends, seasonality, and autocorrelation. Once TSFRESH has extracted a broad set of features from the data, it proceeds to the relevance filtering stage. In this step, TSFRESH applies a hypothesis test to each feature to assess its significance in predicting the target vector. This is accomplished using the 'extract_relevant_features' function, with the target vector being the ability of the participants to debug the program.

This function works by evaluating the p-value of each feature; only features with a p-value below a certain threshold are retained, indicating that they have a statistically significant relationship with the target vector. This process results in a subset of the initially extracted features that are most likely to contribute to the predictive model.

3.3.9 Data analysis

For this analysis, it was decided to use RF as the predictive algorithm. Since there are continuous and categorical variables, combining them in one classifier other than RF was difficult. Moreover, RF offers easy extraction of features importance and has been found to be a top-performing algorithm in a large comparative study (Fernández-Delgado, Cernadas, Barro, & Amorim, 2014) [41]. Neural networks (NNs) were also looked into. However, since the outcomes cannot be explainable in terms of how the NN algorithm combines the features to produce the output, and it is impossible to calculate the importance of features, it was concluded that NNs were not the best approach. Finally, the support vector machine (SVM) approach was looked into, but since SVM can be biased based on the chosen support vectors, it was decided to avoid the risk of overfitting the models.

Random forest classifier

After the dataset was cleaned, measures were calculated, and relevant features were extracted from these measures; the final features were run through scikitlearns RandomForestClassifier. As one of the objectives of this thesis was to get an idea of which data and how much is needed to inform an intervention, the random forest classifier was run several times on different dimensions. One of these dimensions is time. The final dataset was run through the RF classifier, split on percentages of time series per experiment, firstly 100% of the time series, down to 10% of the time series. This was done to get an idea of *how much* sensor data to make an accurate prediction and thus deliver an intervention (early prediction). To gain insight into which of the sensors used in the experiment could give the most accurate prediction and therefore limit overfitting and excess computational usage, the RF classifier was run on datasets pertaining to each sensor individually, one for measures pertaining to the eye-tracker, one for the wristband sensor, and one for the facial measures. These runs were done without feature extraction, only using the computed measures. As mentioned, the experiment's target variable was measured categorically and as a continuous value. Therefore, all the aforementioned runs were done with continuous and categorical values as target vectors. In all, the random forest classifier was run 24 times.

CHAPTER

FOUR

RESULTS

4.1 Performance prediction accuracy using Multimodal data

When it comes to performance prediction in terms of debugging score, the random forest classifier, using the features extracted from the multimodal data provides an accuracy of 0.98. Similarly when it comes to performance prediction in terms of categorical success the random forest using the features extracted from multimodal data provides an accuracy of 0.98. When we analyzed the top 30 most important features from the multi modal data we observe that the top 26 features are derived from engagement and the two other features are there derived from pupil diameter 4.1.1. On the other hand when we analyze the top 30 most important features for predicting the categorical performance, we observe that 28 out of 30 most important features are derived from engagement and pupil diameter. In the case of predicting the performance categories we do not observe the top 26 features to be derived from engagement we observe that the features extracted from pupil diameter and engagement share the top 30 ranks 4.1.2.

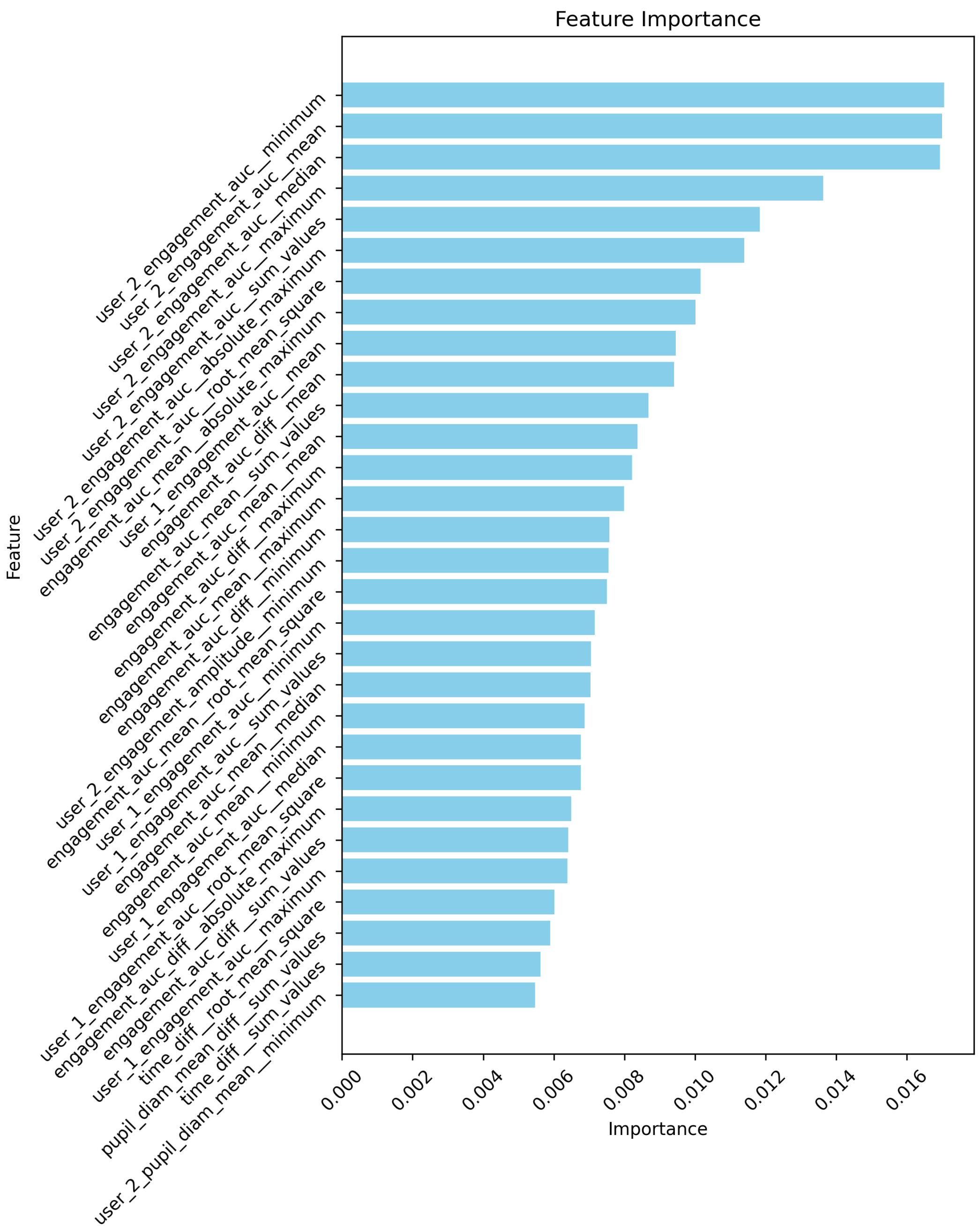


Figure 4.1.1: 30 most relevant features for debugging score

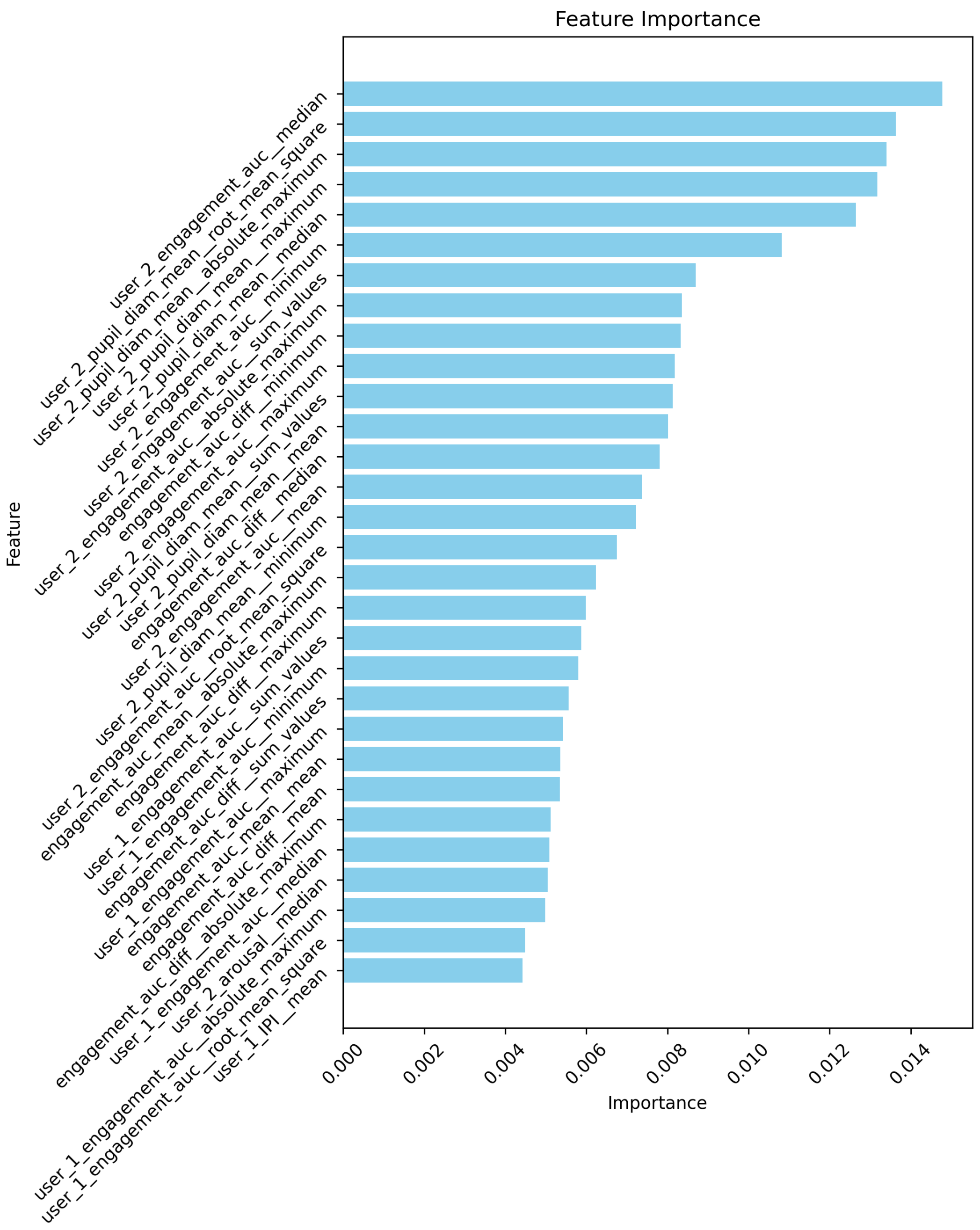


Figure 4.1.2: 30 most relevant features for categorical success

4.2 Performance prediction accuracy using partial Multimodal data (Early prediction)

We chose a threshold of 0.95 For both debugging score prediction and categorical success prediction when it comes to early prediction comparisons. We observe that there is no significant drop until we use 20% of the data in terms of time. This indicates that we can predict the performance of the diets with an accuracy of 0.96 and 0.97 for debugging score and categorical accuracy, respectively, just by using 30% of the total data. In terms of time, this corresponds to on an average 7.2 minutes of interaction time. In the remainder of this subsection we will analyze the top 30 most important features for the different data lengths separately for debug score accuracy and categorical success accuracy.

4.3 Categorical Success

When we use the 90% data to predict the categorical accuracy we notice that the top 30 most important features are derived from engagement (16 out of top 30 features), pupil diameter (six out of 30 top features), saccades (two out of top 30 features), perceived difficulty (two out of top 30 features), heart rate (two out of top 30 features), and information processing index (two out of top 30 features).

When we use the 80% data to predict the categorical accuracy we notice that the top 30 most important features are derived from engagement (17 out of top 30 features), pupil diameter (eight out of 30 top features), perceived difficulty (Two out of top 30 features), saccades (one out of top 30 features), heart rate (one out of top 30 features), and stress (one out of top 30 features).

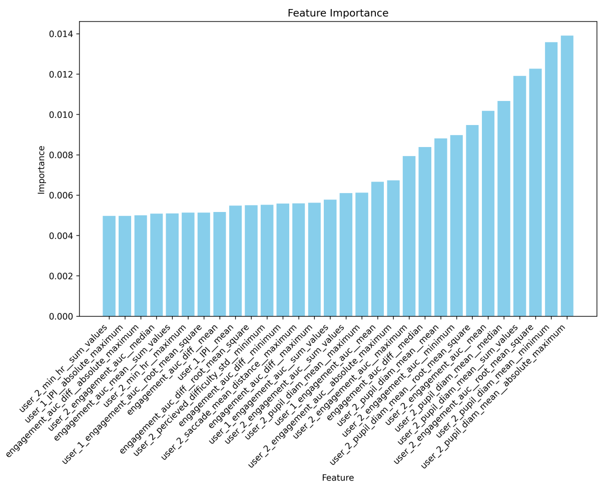
When we use the 70% data to predict the categorical accuracy we notice that the top 30 most important features are derived from engagement (13 out of top 30 features), pupil diameter (nine out of 30 top features), information processing index (three out of top 30 features), stress (three out of top 30 features), saccades (one out of top 30 features), and perceived difficulty (one out of top 30 features).

When we use the 60% data to predict the categorical accuracy we notice that the top 30 most important features are derived from engagement (16 out of top 30 features), pupil diameter (seven out of 30 top features), perceived difficulty (four out of top 30 features), information processing index (two out of top 30 features), and HR (one out of top 30 features).

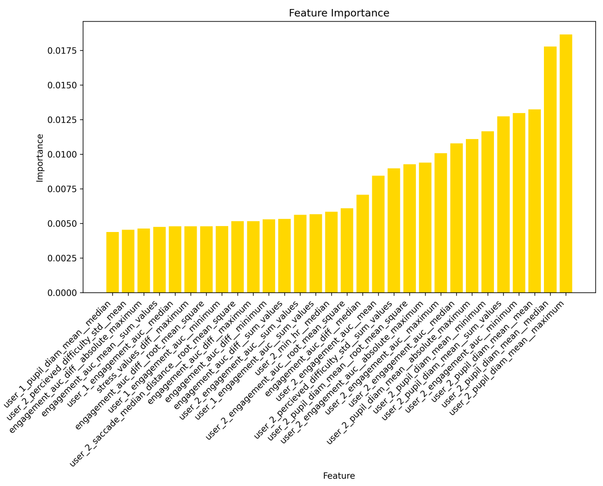
When we use the 50% data to predict the categorical accuracy we notice that the top 30 most important features are derived from engagement (13 out of top 30 features), pupil diameter (11 out of 30 top features), HR (four out of top 30 features), and saccade (two out of top 30 features).

When we use the 40% data to predict the categorical accuracy we notice that the top 30 most important features are derived from engagement (10 out of top 30 features), pupil diameter (10 out of 30 top features), saccade (five out of top 30 features) , perceived difficulty (four out of top 30 features), and HR (one out of top 30 features).

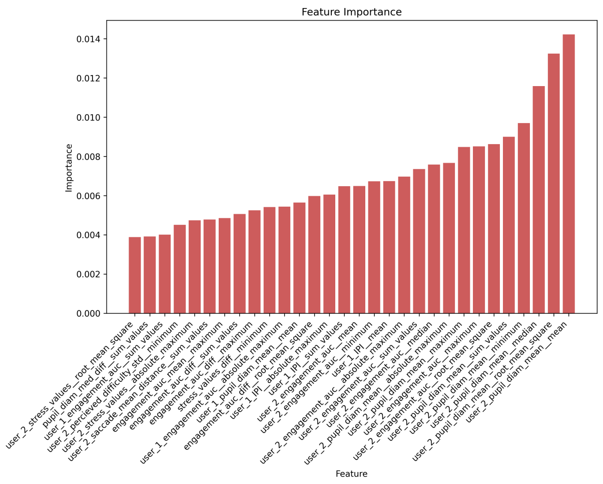
When we use the 30% data to predict the categorical accuracy we notice that the top 30 most important features are derived from pupil diameter (13 out of top 30 features), engagement (10 out of 30 top features), HR (six out of top 30 features), and saccade (one out of top 30 features).



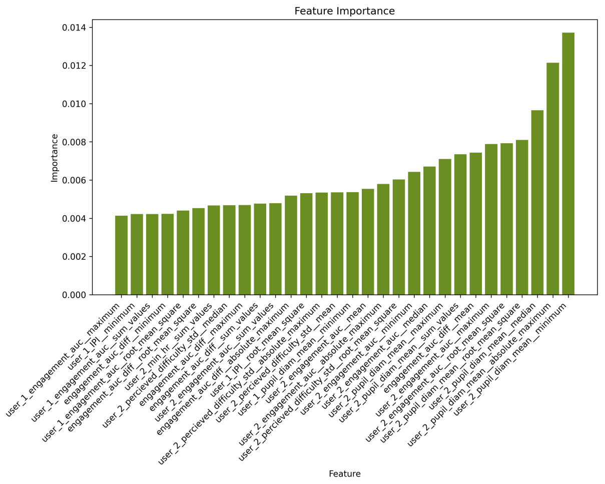
(a) 90% of data



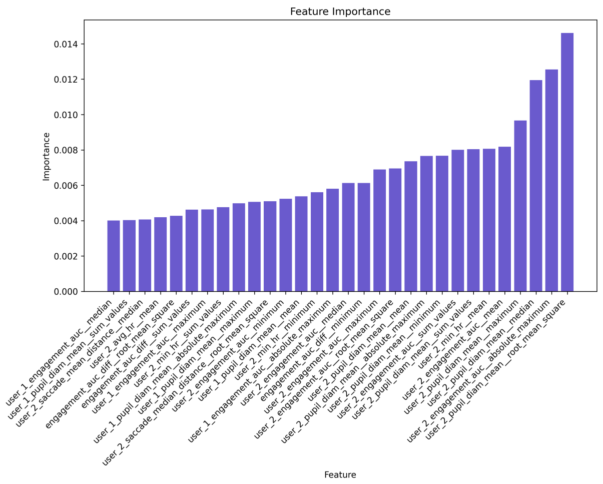
(b) 80% of data



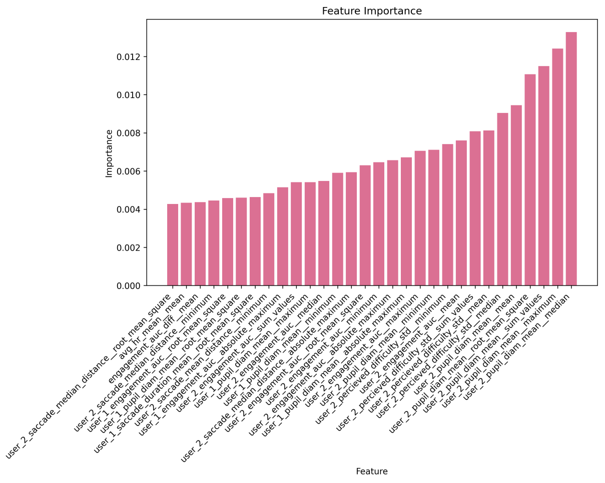
(c) 70% of data



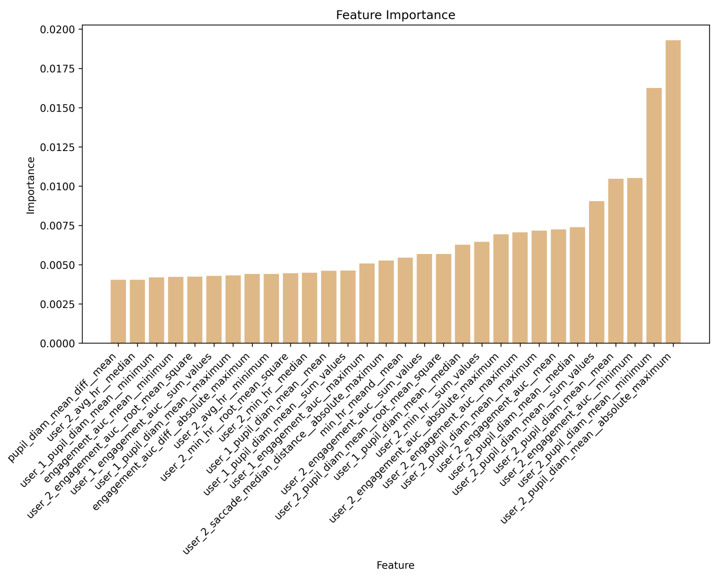
(d) 60% of data



(e) 50% of data



(f) 40% of data



(g) 30% of data

4.4 Debugging Score

When we use the 90% data to predict the debugging score we notice that the top 30 most important features are derived from engagement (25 out of top 30 features), pupil diameter (three out of 30 top features), and arousal (one out of 30 top features).

When we use the 80% data to predict the debugging score we notice that the top 30 most important features are derived from engagement (26 out of top 30 features), pupil diameter (one out of 30 top features), stress (one out of 30 top features), and arousal (one out of 30 top features).

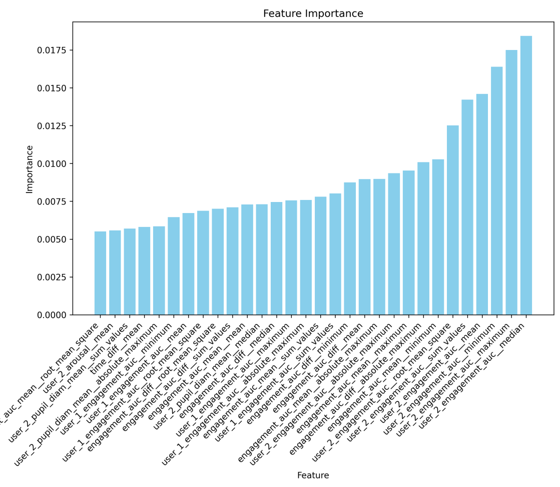
When we use the 70% data to predict the debugging score we notice that the top 30 most important features are derived from engagement (25 out of top 30 features) and pupil diameter (four out of 30 top features).

When we use the 60% data to predict the debugging score we notice that the top 30 most important features are derived from engagement (24 out of top 30 features) and pupil diameter (four out of 30 top features).

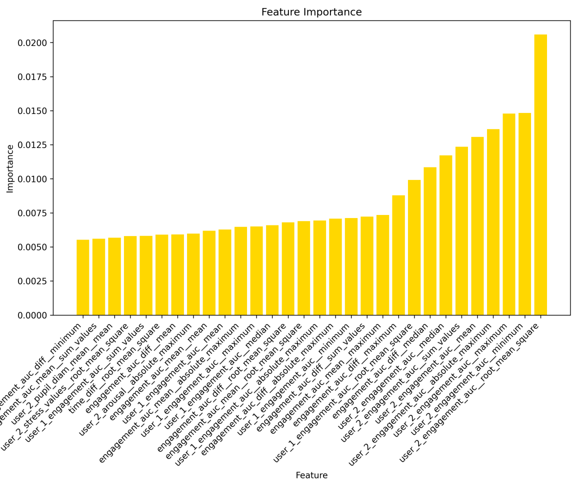
When we use the 50% data to predict the debugging score we notice that the top 30 most important features are derived from engagement (28 out of top 30 features) and pupil diameter (one out of 30 top features).

When we use the 40% data to predict the debugging score we notice that the top 30 most important features are derived from engagement (22 out of top 30 features), pupil diameter (four out of 30 top features), and arousal (two out of top 30 features).

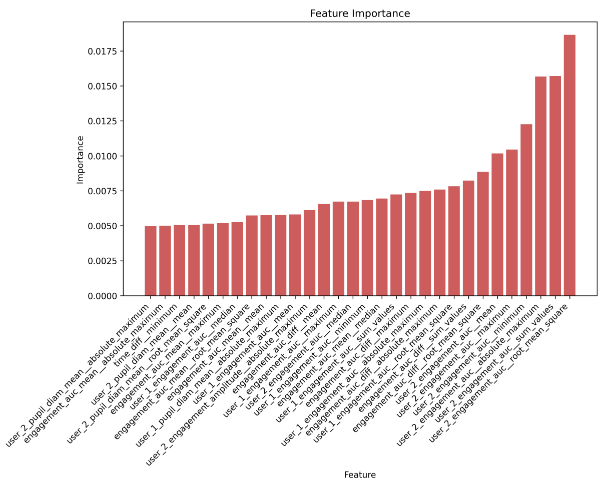
When we use the 30% data to predict the debugging score we notice that the top 30 most important features are derived from engagement (21 out of top 30 features), pupil diameter (7 out of 30 top features), and arousal (two out of top 30 features).



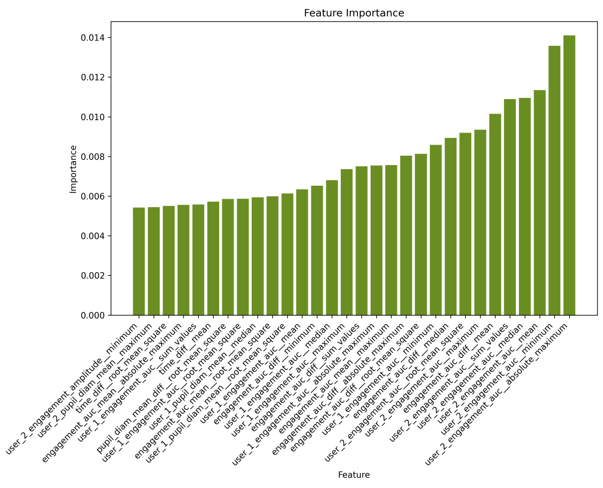
(a) 90% of data



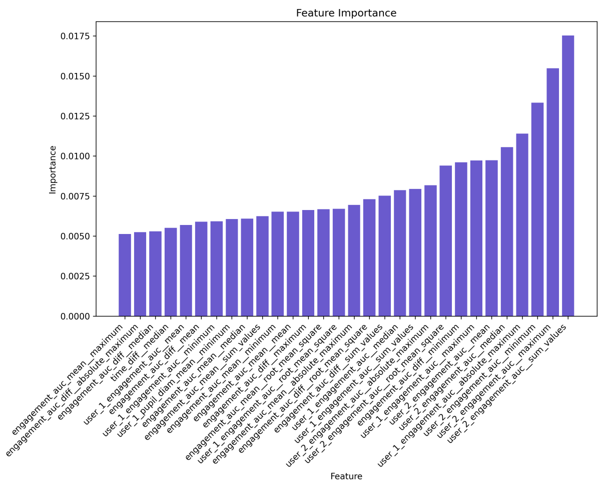
(b) 80% of data



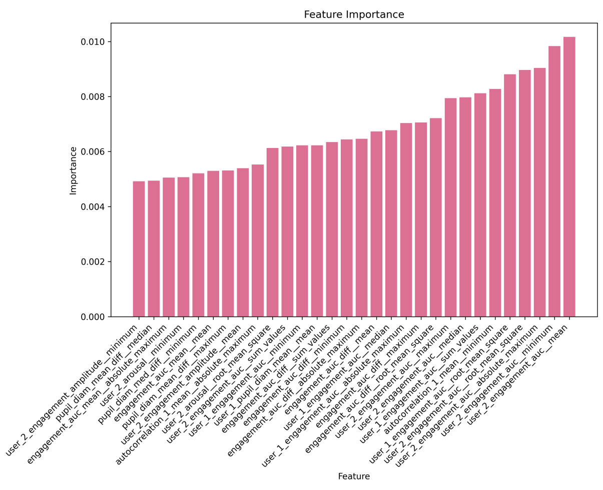
(c) 70% of data



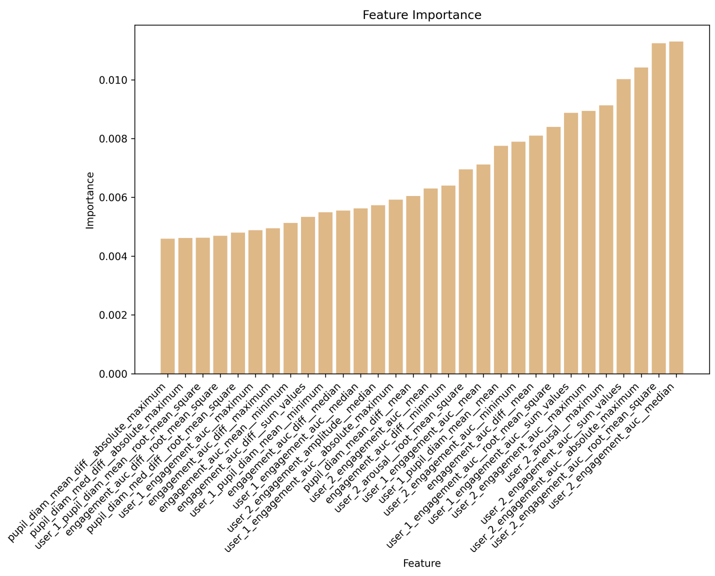
(d) 60% of data



(e) 50% of data



(f) 40% of data



(g) 30% of data

4.4.1 Performance prediction accuracy using individual data sources

We present the results of performance prediction using the individual data sources. We observe that the physiological data provides the highest performance prediction followed by gaze data, and the facial data provides the least accurate performance prediction. This is true for both debugging score and categorical success. We observed that while using only physiological data we get a performance prediction of 0.99 on the debugging score and a performance prediction of 0.99 for categorical success. When we use only eye tracking data to predict debugging score, the accuracy is 0.97 and that for categorical success is 0.96. Finally when we use only facial data to predict the debugging score and categorical success we get accuracies of 0.85 and 0.88 respectively.

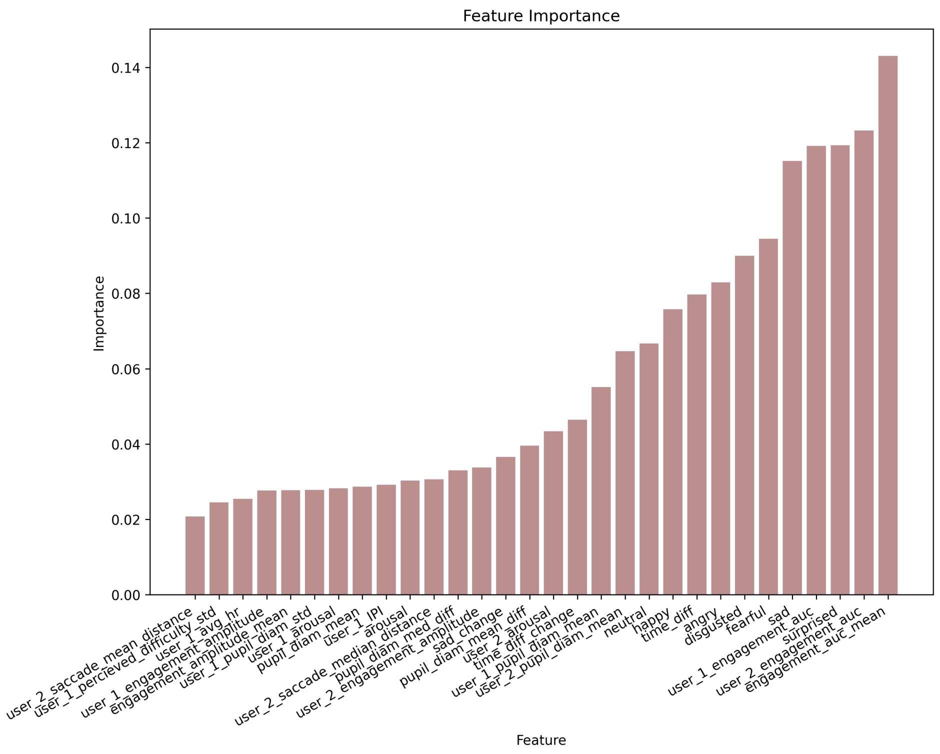


Figure 4.4.1: 30 most important features for all data streams for categorical success

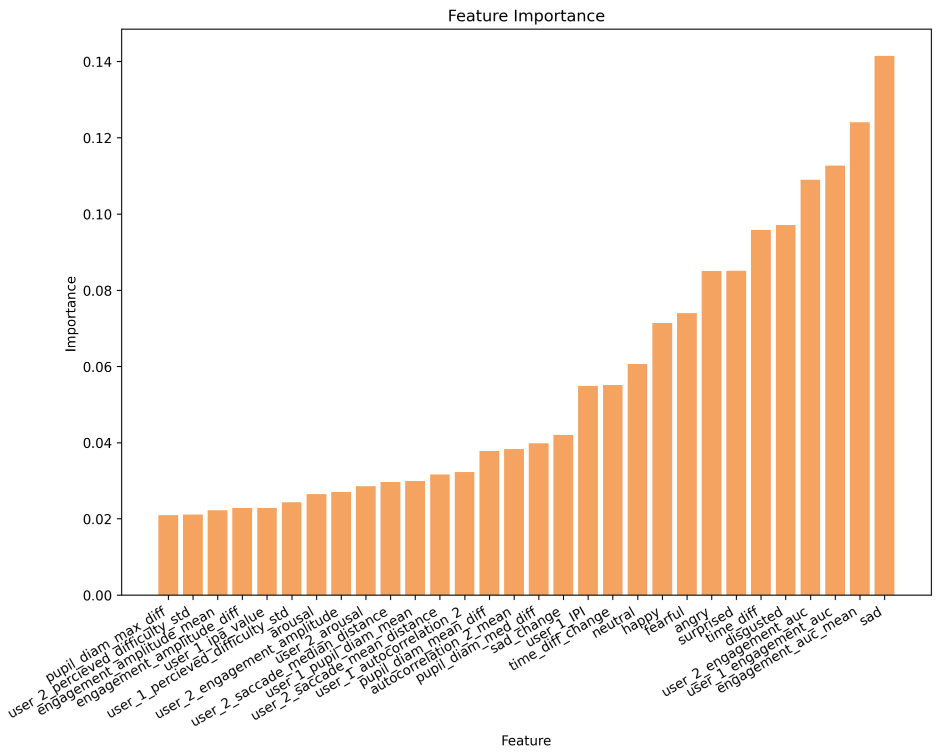


Figure 4.4.2: 30 most important features for all data streams for categorical success

Looking at the feature importance for predicting the performance, when it comes to the debugging score, with individual data sources, we observe the following:

For physiological data, six of top 10 most important features are derived from engagement (engagement_auc_mean, user_2_engagement_auc, user_1_engagement_auc, user_2_engagement_amplitude , user_1_engagement_amplitude, engagement_amplitude_mean). Other three in the top 10 most important features are derived from arousal (user_2_arousal, arousal, user_2_arousal). Other features derived from the HR and stress are ranked lower than the top 10 most important features 4.4.3.

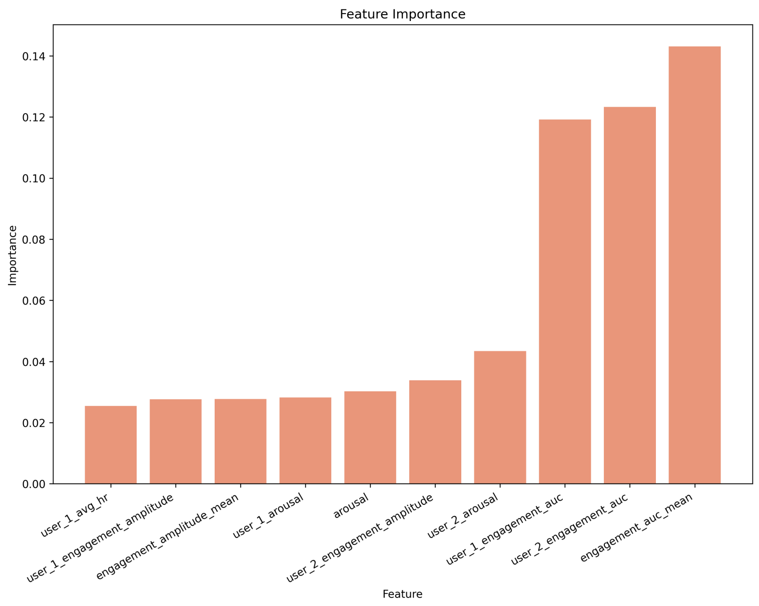


Figure 4.4.3: 10 most important features for physiological data

For gaze data, six of top 10 most important features are derived from pupil diameter (user_2_pupil_diam_mean, user_1_pupil_diam_mean, pupil_diam_mean_diff, pupil_diam_med_diff, pupil_diam_mean, user_1_pupil_diam_std). Other two in the top 10 most important features are derived from saccade (user_2_saccade_median_distance, user_2_saccade_median_distance). Other features derived from rest of the gaze-data measurements are ranked lower than the top 10 most important features 4.4.4.

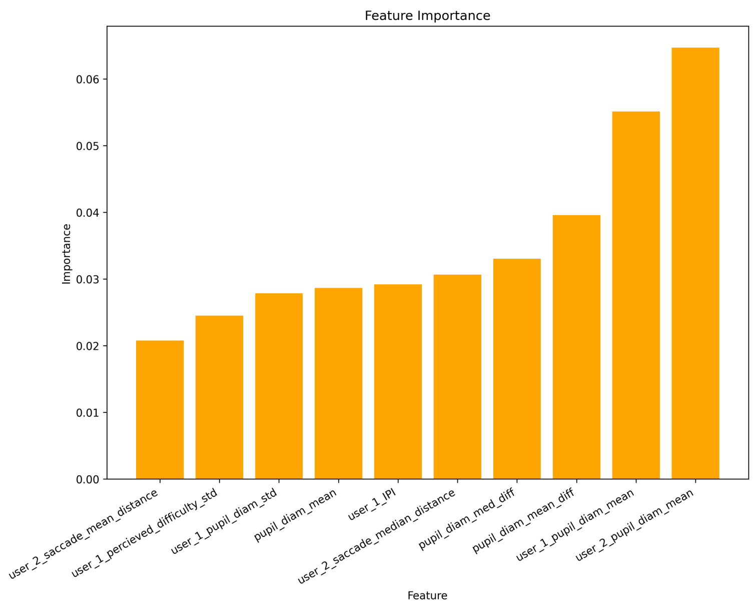


Figure 4.4.4: 10 most important features for gaze data

For facial data, we observe that the negative emotions are ranked higher in the terms of feature importance than the positive emotions. Moreover, we also observe that the emotions with high arousal, with the only exception of sadness, are ranked higher in terms of feature importance than the emotions with low arousal. This is also consistent with the observation that physiological arousal based features rank in top 10 most important features from the physiological data source 4.4.5.

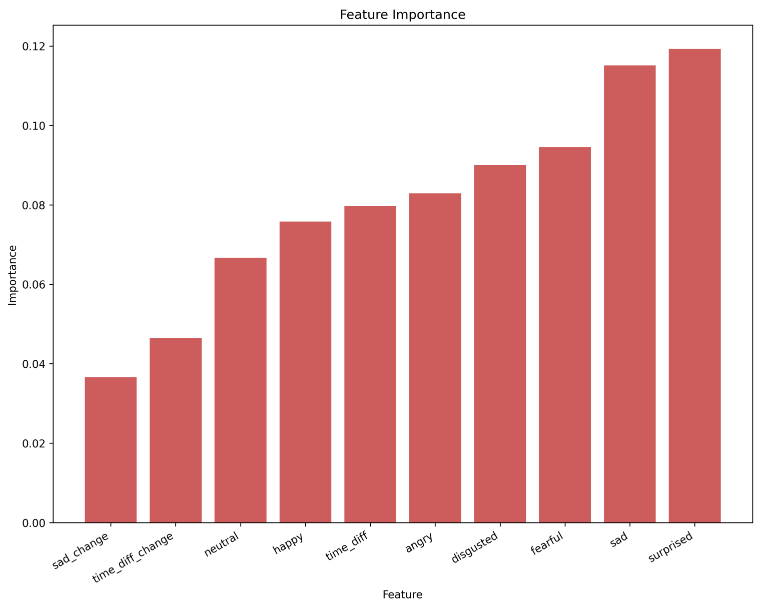


Figure 4.4.5: 10 most important features for facial data

Next looking at the feature importance for predicting the performance, when it comes to the categorical success we observed the following:

For physiological data, six of the top 10 most important features are derived from engagement (engagement_auc_mean, user_1_engagement_auc, user_2_engagement_au user_3_engagement_amplitude, engagement_amplitude_diff, engagement_amplitude_mean). Other two in top 10 most important features are derived from arousal (autocorrelation_2_mean, user_1_autocorrelation_2). Other features derived from HR and stress are ranked lower than the top 10 most important features 4.4.6.

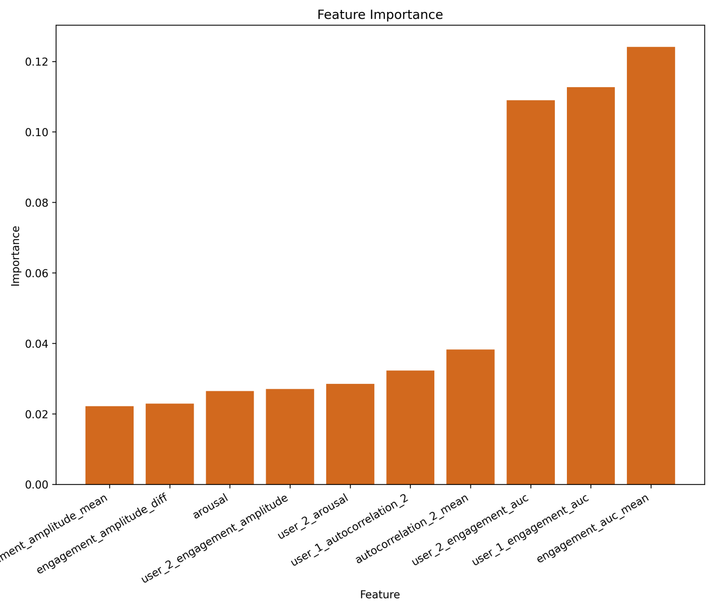


Figure 4.4.6: 10 most important features for physiological data

For gaze data, four of the top 10 most important features are derived from pupil diameter (user_2_IPI, pupil_diam_mean_diff, pupil_diam_median_diff, user_1_pupil_median). The other two of the top 10 most important features are derived from perceived difficulty (user_2_percieved_difficulty_std, user_1_percieved_difficulty_std). There are two more of the top 10 most important features that are derived from saccade distances (user_2_saccade_mean_distance, user_2_saccade_median_distance). Other features derived from rest of the gaze-data measurements are ranked lower than the top 10 most important features 4.4.7.

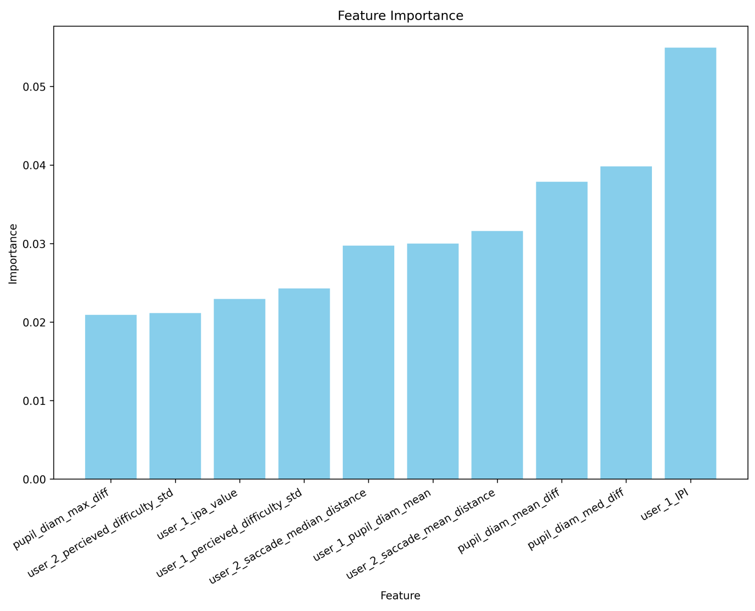


Figure 4.4.7: 10 most important features for gaze data

Finally for facial data we observe that once again the negative emotions are ranked higher in terms of feature importance than the positive emotions however while predicting the categorical success there is no clear indication of arousal based emotions in the terms of feature importance 4.4.8.

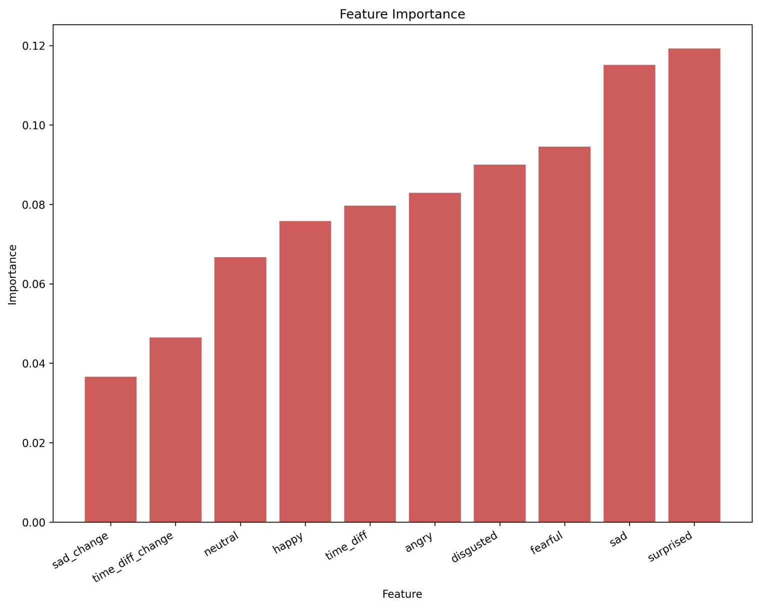


Figure 4.4.8: 10 most important features for facial data

CHAPTER
FIVE

DISCUSSION

This chapter provides a discussion of the results gleaned from the prediction capabilities of the data analysis.

The key findings of the analysis were as follows:

- Using data streams from several sensors leads to very accurate predictions of performance, both when testing for categorical and continuous defined performance.
- Using all data streams provides accurate prediction of performance with only using 30% of time series data.
- Results indicate prediction accuracy is highest for physiological data, then eye-tracking data, and lowest for facial emotion data.
- The most important features from all prediction cases are related to engagement, pupil diameter, HR, stress, arousals and saccades.

From the results we can clearly observe an emerging pattern that six multimodal measurements contribute to the most important features in a vast majority of the prediction cases. That is, with the complete or partial data. These measurements are engagement, pupil diameter, HR, stress, arousal and saccade. Therefore, on the basis of their nature, we can conclude that the most important measures emerging from the multimodal analysis are engagement, pupil diameter, saccades and stress. Due to the fact that stress is also a derived measure from HR and engagement has the similar properties as arousal, we can provide more informative feedback based on stress and engagement as compared to HR and arousal respectively. Finally, we know that pupil diameter and saccades are the basic constituent variables for cognitive load [14], we can provide more informative feedback based on cognitive load rather than pupil diameter and saccades. This answers our first research question, **RQ1**, the most important features that predicts a student performance in a collaborative debugging setting is **stress**, **engagement**, and **cognitive load**. We also see that this holds for early prediction, and that early prediction down to as little as 30% of the data yields accurate predictions, which answers **RQ2**. In the following we will discuss a few ways on how to provide feedback based on the most important factors stress, engagement, and cognitive load, which answers **RQ3**

Interventions for stress

As stress is a measure of affective state more so than measurable cognitive state, interventions catering to this measure may take the form of encouraging actions outside of the problem solving.

The purposes of these interventions would therefore be to encourage the student to step away from the problem solving. Suggestion on modifying the emotional states of the programmers [42] entails limiting negative emotions. With this in mind, we suggest interventions that remind students to take breaks when their measured stress levels reach a threshold.

Another factor to consider, is the nature of pair programming. Several studies has shown pair programming to reduce feeling of frustration[43, 44], an emotive state closely related to stress. With this quality of collaborative coding in mind, we suggest researching interventions that focus on the collaborative experience. This means, interventions that encourages communication between collaborators as a measure to modify stress levels. These interventions could be reminders to communicate to decompose the problem, change focus to another problem, or starting over.

Interventions for cognitive load

The results show that success can be predicted by analysing cognitive load. This is also true for the analysis of early prediction of success, implying that providing intervention is feasible in real time, to increase or decrease cognitive load to a value that increases learning, and thus success. To do this, the study proposes three types of intervention. For all the following suggestions, it should be mentioned that in designing interventions that aim to modify cognitive load, introducing visual elements during programming may indeed reach the goal of decreasing cognitive load but still not helping the pairs achievement or learning. If the interventions ends up decreasing cognitive load, but splitting attention from the task at hand, thus defeating the purpose of the intervention. The modification in cognitive load could also come from the load inferred by the intervention, only reflecting how much focus and attention the intervention requires, be it less or more focus. This argument may in many cases be not that important, as learning to program is a complex process, where the workflow requires attention splitting to several forms of information, to consolidate into learning schemas.

Content based help

A common workflow for solving problems for programmers, novices and experts alike, is searching for solutions, or example code on the web. This process splits the attention of the programmer, and especially for novices this may pose a problem, as they might not understand the code in the existing solutions, or the underlying concepts used in these examples. A workaround for this, could be to implement systems that generate tutorials or explanations for the concepts that generates higher cognitive load. As mentioned, this may overload the student with too much new information, and increase cognitive load. As a countermeasure, this content based help could be tailored to the participant in conjunction with the interventions discussed in the following section.

Worked examples

Several studies has shown that there is a positive correlation between employing worked examples, decrease in cognitive load and increase in performance [45, 46]. This however has been suggested to not be true for higher-prior knowledge learners, who benefit from completion-worked examples [47], that is, partially solved examples. The effect of using worked examples is studies, but a lot of these studies investigate them in the use in mathematics or other fields. Studying the use in programming is rare, something which also holds true in the context of collaborative programming. In both of the case of fully worked and partially worked examples, attention and therefore cognitive load can be directly directed at goal-inducing effort by interventions. Adaptive feedback could be helpful in a collaboration setting, especially if the students engaging in the collaboration have different levels of prior knowledge. This applies to both prior knowledge generally, and on a concept to concept level. Interventions that help towards leveling out the variance of knowledge or achievement would enhance the the technique.

Hints

Hints are a type of intervention that are meant to compensate for a non-sufficient level of prior knowledge. They can however have the opposite effect on learners with a sufficient base of prior knowledge, and act as distractors, decreasing cognitive load, but at a cost of diverting attention to objects not relating to higher performance.

Interventions for engagement

The duration of a fixation may indicate the amount of mental processing required for understanding the information gazed upon. The importance of the concept or the difficulty of understanding could explain higher gaze durations, and could be useful in designing systems for intervention. In the case of using experts gaze data to provide intervention for novice learners, the areas attended to longer, and with higher cognitive load could be used for interventions that hint to where a novice should aim their gaze. Conversely, using experts gaze patterns could help decide which level of cognitive load is appropriate for the given AOI or concept, and could merit intervention in the form of questions to the learner, if the system notices that the novice is not applying enough mental effort.

Modifying difficulty of problem

A factor that can affect students measured engagement the level of difficulty while solving a problem. Concepts that require a higher level of understanding might limit engagement, regardless of effort, and can be a demotivating factor for novices. Similarly for high achieving students, they might express lower engagement if the concept they are tackling is too understandable, or too novel for their level. A form of intervention that may tackle both of these issues, is an adaptive interventions that scale the difficulty of the learning concept based on the predicted success of the effort the pair is demonstrating.

Summative intervention

Another dimension of engagement apart from the cognitive, is the behavioural dimension of engagement. Sadler [48] employs a notion of "feedback gap", which for the students entails an understanding of the their own performance in relation to the goal of the task at hand. This can relate to the behavioural dimension of engagement, as this dimension could require summative instead of formative feedback. This means, to foster engagement for students, interventions that helps students understand their progress towards the learning goal, or visualizes their achievement may help modify their engagement in a way that increases learning and achievement.

5.1 Future work

As will be discussed in 6, the data collection suffered from issues with the system as a whole, and with individual sensors. Therefore, the first recommendation for future work would be to run the experiment described in this thesis again, for more data points for analysis. Another way of handling missing data would be to implement an imputation technique. Another option is to use Synthetic Minority Oversampling Technique (SMOTE) to create synthetic samples. Increasing sampling size would help underline which sensor features could be best suited for informing interventions. As we saw from the results, the predictions were very accurate, even with 30% of the data, purely because of the wealth of data samples, and the continuous nature of the data. SMOTE, or other statistical techniques could overcome overfitting by generating minority instances, or outliers through interpolation.

As the results of predicting based on individual data sensors imply, based on the accuracy of prediction, imply that some features may overlap in information. Short-term future research may focus on identifying which factors contain mutual information. In a future system that gives real time intervention, a possible goal should be to limit the amount of data is needed to make effective interventions. The idea of early prediction should be a focus for such a system, as collecting multimodal data is expensive, both in storage, computation and processing. As described in limitations 6, complex systems are time-consuming to manage, so doing more exploratory testing on which permutations of sensors gives sufficient accuracy in predicting success.

As described in section 6, the initial plan for this study was to actually design the interventions discussed. This plan involved a qualitative analysis of the extracted important features, and then a workshop with professors of HCI and design, for the design of these interventions. The next step was to make low-level designs of these actionable feedbacks. Therefore a long-term recommendation would be to implement designs for the actionable feedback, and then measure the effects of the feedback. As the effect of feedback on learning and academic success is something that is inherently cyclical, and takes more time than is permitted in a masters thesis, the effect of the implemented interventions should be in form of a case study ranging from a whole semester to a year, to gain novel insight, much like in [13].

CHAPTER
SIX

CONCLUSIONS

This thesis builds on and replicates an experiment using a web-based collaborative code editor, where multimodal data has been collected and analysed for use in design of interventions for students learning. The thesis presents the most important features or streams of data for predicting students performance while debugging, and how the datatreams collected could be used for early prediction of students success. The thesis provides a discussion about how these features could be used to provide interventions in the form of actionable feedback, and argues why this feedback could be used to improve student performance in a collaborative debug setting.

Limitations

This section describes the limitations of the thesis.

Improvement of system

A large workload of this study went into setting up the system to use for data collection during the experiments, as well as correcting issues during the experiments, which will be discussed further in the next section 6. The system as described, consisted of several sensors, protocols, languages, frameworks within these languages, packages for these languages, as well as proprietary software for the sensors. All of this had to run simultaneously on two different computers. Initially, the idea for the study and the experiments, were to use the system as delivered, plug and play. This was not the case, and such, around two months worth of workload was invested into making the system run. As one of the objectives of this thesis was to recreate the experiment done in [3], using the same system to test viability of collecting realtime multimodal data in a collaborative setting, the time spent debugging/improving the system was deemed worthwhile. The system, as mentioned consisted of several parts, with limited documentation and scale-wise a very large code base to get acquainted with. Disregarding getting to know the code-base, the problems that needed to be fixed could be summed up in three categories: language specific package dependencies issues, issues with individual sensors, and all their parts and proprietary software, and ensuring that all the parts worked consistently as a whole, for both participants in the experiment.

The issues with the first category were interconnected with the second one. The python packages used for parsing the data from both the eye-tracker and the wristband was pandas. During the time period from the implementation of the system, to the time it was employed for this thesis, this package had received updates. The system used several methods from pandas that were deprecated, but for several reasons, information about this deprecation was not relayed through the system. With little information to go from, the fix for this was to line by line debug the system for output. A similar issue, but smaller happened with programmatic connection to the eye-tracker. The eye-tracker used in the original experiment [3] and this experiment is no longer in production, and the documentation for both the eye-tracker and the SDK is scarce. The solution to this was to downgrade the python versioning, but this was difficult to figure out, as the SDK did not give error output specific for this to be a problem. This had to be figured out by brute force trying solutions. The biggest hurdle in getting the system to work, was to get consistent output from the wristbands. Firstly, the Empatica E4 is also a discontinued model, replaced by Empatica EmbracePlus. The Empatica E4 is connected with a BLED112 dongle through Bluetooth, and to get streaming data from this wristband, it is necessary to use E4 Streaming Server for Windows [49], documentation last updated in 2018. This proprietary server is connected to the system via a TCPClient. In short form, there are four layers of failure for getting streaming data from this sensor, and from the system, no error messages to inform on which step is causing the loss of information. Debugging included manual line by line debugging in python, updating drivers of the hardware, both the watch and the BLED112 dongle, consulting Empatica for advice, and Wiresharking to confirm for packet analysis. This was done on several machines to ensure that installation of all mentioned steps were done correctly. Wiresharking confirmed that the wristbands connected to each machine, and sometimes sent data, but not always. The proprietors of the wristbands recommended deleting the software and roaming data every time the wristbands were used, but this still didn't lead to consistent data transfer. A part of the problem was related to Windows Firewall, which may or may not block the signal, without giving notice as to when this was the case. One solution that seemed to work was to simply start the streaming server as Administrator. The proprietors of the wristbands were not aware of this issue.

Finally, this troubleshooting had to be done to both machines that would run the experiment.

Experimental limitations

Due to miscommunication, some of the experiments (10) were done in a different lab setting. The UX-lab was unavailable at the time. This led to half of the experiments taking place in different locations. The design lab was located in a room with very little ventilation. The only window in the room was adjacent to a construction site on campus, which at times made noise. As bad indoor air quality can lead to lapse in concentration, headaches, and generally bad performance, the assessment was made to keep the window open, as well as using a blade fan to further increase air quality. This decision was made knowing that the noise from the construction site and the fan might compromise the audio quality, and lead to the need of processing the audio in a way that constitutes loss of data.

At the first day scheduled for experiments, we discovered several errors with both the wristbands and one of the eye-tracker. The code used to parse the data collected from the wristbands, ended up with only collecting BVP. This proved that the issues with the wristbands persisted, and the long troubleshooting process had proved for naught. One of the eye-trackers disconnected at random, which gave no data collected from the eye-tracker. To tackle the issues with collecting data from the wristbands, the module that was used for streaming and parsing data from the wristband was discarded. Instead, a solution using Empaticas proprietary software was used. This app recorded the data directly to a database, and was processed at a later date, rather than directly with the system.

During the run of the tests, errors with the sensors continued. After getting consistent data from the wristbands, and solving this issue, the issues with the eye-trackers persisted, leading to incomplete data from the experiments.

Due to the technical difficulties of using one of the eyetracker (difficulties getting the eye-tracker to recognize the participant during calibration) the participants were allowed to move their chair. Therefore the viewing distance fluctuates, and may lead to less accuracy in the measurements. Saccade amplitudes were not calculated because of the fluctuation of the seating distances.

Considering the amount of time that was used to troubleshoot the system, the experiments had to be carried out, as it neared the end of the semester, and students were beginning their exam preparations, and were less willing to participate, there was no time to improve upon the system.

Missing computed data

The issues described above lead to few complete datasets from the experiments. 20 experiments were ran, but only 9 of the experiments data were complete, and could be used for analysis.

The practical implications the above mentioned troubleshooting had on the thesis, is twofold. Firstly, the amount of time spent on just setting up the experiment left little time on the analysis-part for the thesis. The original plan of the study was designing the interventions recommended in the discussion.

REFERENCES

[1] Katerina Mangaroska et al. "Multimodal Learning Analytics to Inform Learning Design: Lessons Learned from Computing Education". In: *Journal of Learning Analytics* 7.3 (Dec. 2020), pp. 79–97. DOI: 10.18608/jla.2020.73.7. URL: https://learning-analytics.info/index.php/JLA/article/view/6816.

[2] C. D. Hundhausen, D. M. Olivares, and A. S. Carter. "IDE-Based Learning Analytics for Computing Education: A Process Model, Critical Review, and Research Agenda". In: *ACM Trans. Comput. Educ.* 17.3 (Aug. 2017). DOI: 10.1145/3105759. URL: https://doi.org/10.1145/3105759.

[3] Fredrik Svendsen. "Collaborative code editing Tool: Design and Evaluation". MA thesis. NTNU, 2022.

[4] Øystein Bjørkeng Haugen. "How eye-tracking and facial recognition can be used to give interventions to learners". In: (2022).

[5] Fred Paas et al. "Cognitive Load Measurement as a Means to Advance Cognitive Load Theory". In: *Educational Psychologist - EDUC PSYCHOL* 38 (Mar. 2003), pp. 63–71. DOI: 10.1207/S15326985EP3801_8.

[6] Zohreh Sharafi, Zéphyrin Soh, and Yann-Gaël Guéhéneuc. "A systematic literature review on the usage of eye-tracking in software engineering". In: *Information and Software Technology* 67 (2015), pp. 79–107. ISSN: 0950-5849. DOI: https://doi.org/10.1016/j.infsof.2015.06.008. URL: https://www.sciencedirect.com/science/article/pii/S0950584915001196.

[7] Roman Bednarik and Markku Tukiainen. "An Eye-Tracking Methodology for Characterizing Program Comprehension Processes". In: *Proceedings of the 2006 Symposium on Eye Tracking Research &; Applications*. ETRA '06. San Diego, California: Association for Computing Machinery, 2006, pp. 125–132. ISBN: 1595933050. DOI: 10.1145/1117309.1117356. URL: https://doi.org/10.1145/1117309.1117356.

[8] Laurie A. Williams and Robert R. Kessler. "All I Really Need to Know about Pair Programming I Learned in Kindergarten". In: *Commun. ACM* 43.5 (2000), pp. 108–114. ISSN: 0001-0782. DOI: 10.1145/332833.332848. URL: https://doi.org/10.1145/332833.332848.

[9] John T. Nosek. "The Case for Collaborative Programming". In: *Commun. ACM* 41.3 (1998), pp. 105–108. ISSN: 0001-0782. DOI: 10.1145/272287.272333. URL: https://doi.org/10.1145/272287.272333.

[10] Prashant Baheti, Laurie Williams, and P. Stotts. “Exploring Pair Programming in Distributed Object-Oriented Team Projects”. In: (Jan. 2002).

[11] Tamara van Gog and Halszka Jarodzka. “Eye Tracking as a Tool to Study and Enhance Cognitive and Metacognitive Processes in Computer-Based Learning Environments”. In: *International Handbook of Metacognition and Learning Technologies*. New York, NY: Springer New York, 2013, pp. 143–156. ISBN: 978-1-4419-5546-3. DOI: 10.1007/978-1-4419-5546-3_10. URL: https://doi.org/10.1007/978-1-4419-5546-3_10.

[12] Katerina Mangaroska et al. “Gaze Insights into Debugging Behavior Using Learner-Centred Analysis”. In: LAK ’18. Sydney, New South Wales, Australia: Association for Computing Machinery, 2018, pp. 350–359. ISBN: 9781450364003. DOI: 10.1145/3170358.3170386. URL: https://doi.org/10.1145/3170358.3170386.

[13] Hallvard Trætteberg et al. “Utilizing Real-Time Descriptive Learning Analytics to Enhance Learning Programming”. In: Jan. 2018, pp. 1–22. ISBN: 978-3-319-17727-4. DOI: 10.1007/978-3-319-17727-4_117-1.

[14] Jonathan L. Rosch and Jennifer J. Vogel-Walcutt. “A review of eye-tracking applications as tools for training”. In: *Cognition, Technology & Work* 15 (2012), pp. 313–327.

[15] Fred Paas et al. “A motivational perspective on the relation between mental effort and performance: Optimizing learner involvement in instruction”. In: *Educational Technology Research and Development* 53 (2005), pp. 25–34.

[16] Briony J Oates. *Researching Information Systems and Computing*. Sage Publications Ltd., 2012. ISBN: 1412902231.

[17] .

[18] https://connect.tobii.com/s/x3-downloads?language=en$_US\&tabset-d8ee8 = 12b81$. Accesed: 12-07-2023.

[19] Tobii AB. *Tobii Pro Lab*. Computer software. bibcomputersoftware. Version 1.2xx. Danderyd, Stockholm, 2023. URL: https://developer.tobiipro.com/eyetrackermanager.html.

[20] https://developer.empatica.com/.

[21] https://justadudewhohacks.github.io/face-api.js/docs/index.html. Accessed: 03-04-2023.

[22] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009. ISBN: 1441412697.

[23] The pandas development team. *pandas-dev/pandas: Pandas*. Version latest. Feb. 2020. DOI: 10.5281/zenodo.3509134. URL: https://doi.org/10.5281/zenodo.3509134.

[24] J. D. Hunter. “Matplotlib: A 2D graphics environment”. In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55.

[25] Charles R. Harris et al. “Array programming with NumPy”. In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[26] https://tsfresh.readthedocs.io/en/latest/text/introduction.html. Accessed: 12-07-2023.

[27] James Robert, Marc Webbie, et al. *Pydub*. 2018. URL: http://pydub.com/.

[28] Pauli Virtanen et al. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[29] Dominik Leiner, Andreas Fahr, and Hannah Früh. “EDA Positive Change: A Simple Algorithm for Electrodermal Activity to Measure General Audience Arousal During Media Exposure”. In: *Communication Methods and Measures* 6 (Dec. 2012), pp. 237–250. DOI: 10.1080/19312458.2012.732627.

[30] Steve Pincus. “Approximate Entropy as a Measure of System Complexity”. In: *Proceedings of the National Academy of Sciences of the United States of America* 88 (Apr. 1991), pp. 2297–301. DOI: 10.1073/pnas.88.6.2297.

[31] Dario Salvucci and Joseph Goldberg. “Identifying fixations and saccades in eye-tracking protocols”. In: Jan. 2000, pp. 71–78. DOI: 10.1145/355017.355028.

[32] Frans van der Sluis and Egon L. van den Broek. “Feedback beyond accuracy: Using eye-tracking to detect comprehensibility and interest during reading”. In: *Journal of the Association for Information Science and Technology* 74.1 (2023), pp. 3–16. DOI: https://doi.org/10.1002/asi.24657. eprint: https://asistdl.onlinelibrary.wiley.com/doi/pdf/10.1002/asi.24657. URL: https://asistdl.onlinelibrary.wiley.com/doi/abs/10.1002/asi.24657.

[33] Ilias O Pappas et al. “Visual Aesthetics of E-Commerce Websites: An Eye-Tracking Approach”. In: ().

[34] John Hollander and Stephanie Huette. “Extracting blinks from continuous eye-tracking data in a mind wandering paradigm”. In: *Consciousness and Cognition* 100 (2022), p. 103303. ISSN: 1053-8100. DOI: https://doi.org/10.1016/j.concog.2022.103303. URL: https://www.sciencedirect.com/science/article/pii/S1053810022000356.

[35] Myrthe Faber, Robert Bixler, and Sidney D’Mello. “An automated behavioral measure of mind wandering during computerized reading”. In: *Behavior Research Methods* 50 (Feb. 2017). DOI: 10.3758/s13428-017-0857-y.

[36] Andrew T. Duchowski et al. “The Low/High Index of Pupillary Activity”. In: *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. CHI ’20. Honolulu, HI, USA: Association for Computing Machinery, 2020, pp. 1–12. ISBN: 9781450367080. DOI: 10.1145/3313831.3376394. URL: https://doi.org/10.1145/3313831.3376394.

[37] Andrew T. Duchowski et al. “The Index of Pupillary Activity: Measuring Cognitive Load Vis-à-Vis Task Difficulty with Pupil Oscillation”. In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. CHI ’18. Montreal QC, Canada: Association for Computing Machinery, 2018, pp. 1–13. ISBN: 9781450356206. DOI: 10.1145/3173574.3173856. URL: https://doi.org/10.1145/3173574.3173856.

[38] Kshitij Sharma, Serena Lee-Cultura, and Michail Giannakos. "Keep Calm and Do Not Carry-Forward: Toward Sensor-Data Driven AI Agent to Enhance Human Learning". In: *Frontiers in Artificial Intelligence* 4 (2022). ISSN: 2624-8212. DOI: 10.3389/frai.2021.713176. URL: https://www.frontiersin.org/articles/10.3389/frai.2021.713176.

[39] Michail Giannakos, Sofia Papavlasopoulou, and Kshitij Sharma. "Monitoring Children's Learning Through Wearable Eye-Tracking: The Case of a Making-Based Coding Activity". In: *IEEE Pervasive Computing* 19 (Jan. 2020). DOI: 10.1109/MPRV.2019.2941929.

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

[41] Manuel Fernández-Delgado et al. "Do We Need Hundreds of Classifiers to Solve Real World Classification Problems?" In: *J. Mach. Learn. Res.* 15.1 (2014), pp. 3133–3181. ISSN: 1532-4435.

[42] Daniela Girardi et al. "Recognizing Developers' Emotions While Programming". In: *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering.* ICSE '20. Seoul, South Korea: Association for Computing Machinery, 2020, pp. 666–677. ISBN: 9781450371216. URL: https://doi.org/10.1145/3377811.3380374.

[43] Alistair Cockburn and Laurie Williams. "The Costs and Benefits of Pair Programming". In: (Feb. 2000).

[44] Brian Hanks et al. "Pair programming in education: a literature review". In: *Computer Science Education* 21.2 (2011), pp. 135–173. DOI: 10.1080/08993408.2011.579808. eprint: https://doi.org/10.1080/08993408.2011.579808. URL: https://doi.org/10.1080/08993408.2011.579808.

[45] Fred Paas and Jeroen J. G. Van Merrienboer. "Variability of Worked Examples and Transfer of Geometrical Problem-Solving Skills: A Cognitive-Load Approach". In: *Journal of Educational Psychology* 86 (Mar. 1994), pp. 122–133. DOI: 10.1037/0022-0663.86.1.122.

[46] Ingrid A. E. Spanjers, Tamara van Gog, and Jeroen J. G. van Merriënboer. "Segmentation of Worked Examples: Effects on Cognitive Load and Learning". In: *Applied Cognitive Psychology* 26.3 (2012), pp. 352–358. DOI: https://doi.org/10.1002/acp.1832. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/acp.1832. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/acp.1832.

[47] U. Gupta and R. Z. Zheng. "Cognitive Load in Solving Mathematics Problems: Validating the Role of Motivation and the Interaction Among Prior Knowledge, Worked Examples, and Task Difficulty." In: *European Journal of STEM Education, 5(1), 05.* (2020). DOI: https://doi.org/10.20897/ejsteme/9252.

[48] D. R. Sadler. "Formative assessment and the design of instructional systems". In: *Instructional Science, 18(2), 119–144* (1989). DOI: https://doi.org/10.1007/BF00117714.

[49] https://developer.empatica.com/windows-streaming-server.html.

APPENDICES

A - PRETEST

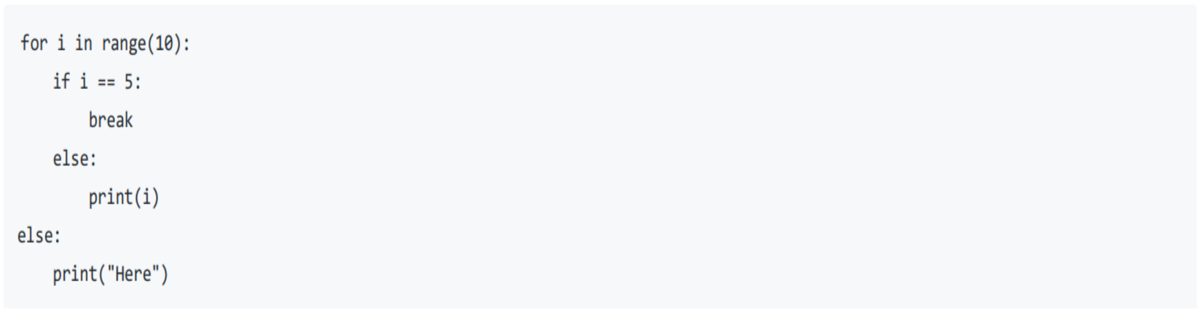
Pre-test: Debugging

Name:__________________

For each question you will be given a code snippet and you have to figure out what the output of the snippet is or what it does.

Question 1

What is the output of this program? Circle the correct answer.

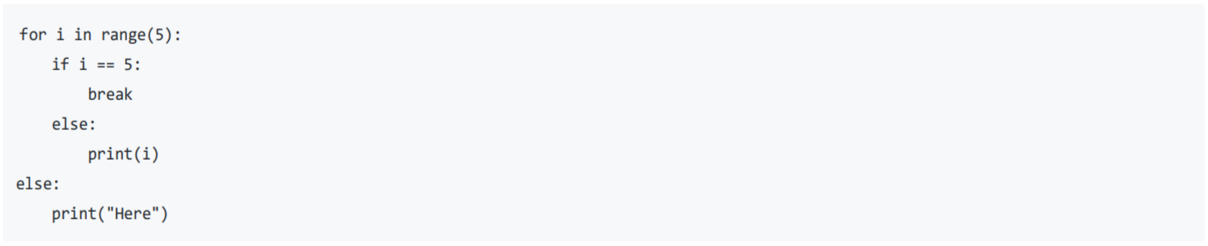


```
for i in range(10):
    if i == 5:
        break
    else:
        print(i)
else:
    print("Here")
```

A. 0 1 2 3 4 Here
B. 0 1 2 3 4 5 Here
C. 0 1 2 3 4
D. 1 2 3 4 5

Question 2

What is the output of this program? Circle the correct answer.



```
for i in range(5):
    if i == 5:
        break
    else:
        print(i)
else:
    print("Here")
```

A. 0 1 2 3 4 Here
B. 0 1 2 3 4 5 Here
C. 0 1 2 3 4
D. 1 2 3 4 5

Question 3
What is the output of this program? Circle the correct answer.

```
a = [0, 1, 2, 3]
i = -2
for i not in a:
    print(i)
    i += 1
```

A. -2 -1
B. 0
C. error
D. none of the mentioned

Question 4
What is the output of this program? Circle the correct answer.

```
class Demo:
    def __new__(self):
        self.__init__(self)
        print("Demo's __new__() invoked")

    def __init__(self):
        print("Demo's __init__() invoked")

class Derived_Demo(Demo):
    def __new__(self):
        print("Derived_Demo's __new__() invoked")

    def __init__(self):
        print("Derived_Demo's __init__() invoked")

def main():
    obj1 = Derived_Demo()
    obj2 = Demo()
main()
```

A.
1. Derived_Demo’s __init__() invoked
2. Derived_Demo's __new__() invoked
3. Demo's __init__() invoked
4. Demo's __new__() invoked

B.
1. Derived_Demo's __new__() invoked
2. Demo's __init__() invoked
3. Demo's __new__() invoked

C.
1. Derived_Demo's __new__() invoked
2. Demo's __new__() invoked

D.
1. Derived_Demo’s __init__() invoked
2. Demo's __init__() invoked

Question 5

What is the output of this program? Circle the correct answer.

```
class Test:
    def __init__(self):
        self.x = 0

class Derived_Test(Test):
    def __init__(self):
        self.y = 1
def main():
    b = Derived_Test()
    print(b.x,b.y)

main()
```

A. 0 1
B. 0 0
C. Error because class B inherits A but variable x isn't inherited
D. none of the mentioned

Question 6

What is the output of this program? Circle the correct answer.

```
count={}
count[(1,2,4)] = 5
count[(4,2,1)] = 7
count[(1,2)] = 6
count[(4,2,1)] = 2
tot = 0
for i in count:
    tot=tot+count[i]
print(len(count)+tot)
```

A. 25
B. 17
C. 16
D. Tuples can't be made keys of a dictionary

Question 7

What is the output of this program? Write down the output.

```
def fn(**kwargs):
    for emp, age in kwargs.items():
        print ("%s's age is %s." %(emp, age))

fn(John=25, Kalley=22, Tom=32)
```

Question 8

What is the output of this program? Write down the output.

```
class PC: # Base class
    processor = "Xeon" # Common attribute
    def set_processor(self, new_processor):
        processor = new_processor

class Desktop(PC): # Derived class
    os = "Mac OS High Sierra" # Personalized attribute
    ram = "32 GB"

class Laptop(PC): # Derived class
    os = "Windows 10 Pro 64" # Personalized attribute
    ram = "16 GB"

desk = Desktop()
print(desk.processor, desk.os, desk.ram)

lap = Laptop()
print(lap.processor, lap.os, lap.ram)
```

Question 9

What is the output of this program? Write down the output.

```
class PC: # Base class
    processor = "Xeon" # Common attribute
    def __init__(self, processor, ram):
        self.processor = processor
        self.ram = ram
    def set_processor(self, new_processor):
        processor = new_processor
    def get_PC(self):
        return "%s cpu & %s ram" % (self.processor, self.ram)

class Tablet():
    make = "Intel"
    def __init__(self, processor, ram, make):
        self.PC = PC(processor, ram) # Composition
        self.make = make

    def get_Tablet(self):
        return "Tablet with %s CPU & %s ram by %s" % (self.PC.processor, self.PC.ram, self.make)

if __name__ == "__main__":
    tab = Tablet("i7", "16 GB", "Intel")
    print(tab.get_Tablet())
```

Question 10

What is the output of this program? Write down the output.

```
def multiply_number(num):
    def product(number):
        'product() here is a closure'
        return num * number
    return product


num_2 = multiply_number(2)
print(num_2(11))
print(num_2(24))

num_6 = multiply_number(6)
print(num_6(1))
```

B - DEBUG TASK

```python
import math
import random
from os import environ

environ["PYGAME_HIDE_SUPPORT_PROMPT"] = "1"
import pygame
from pygame import mixer

# Intialize the pygame
pygame.init()

# create the screen
screen = pygame.display.set_mode((800, 600))

# Background
background = pygame.image.load("background.png")

# Sound
mixer.music.load("background.wav")
mixer.music.play(-1)

# Caption and Icon
pygame.display.set_caption("Space Invader")
icon = pygame.image.load("ufo.png")
pygame.display.set_icon(icon)

# Player
playerImg = pygame.image.load("player.png")
playerX = 370
playerY = 480
playerX_change = 0

# Enemy
enemyImg = []
enemyX = []
enemyY = []
enemyX_change = []
enemyY_change = []
num_of_enemies = 6

for i in range(num_of_enemies):
    enemyImg.append(pygame.image.load("enemy.png"))
    enemyX.append(random.randint(0, 736))
    enemyY.append(random.randint(50, 150))
    enemyX_change.append(4)
    enemyY_change.append(40)

# Bullet

# Ready - You can't see the bullet on the screen
# Fire - The bullet is currently moving

bulletImg = pygame.image.load("bullet.png")
bulletX = 0
bulletY = 480
bulletX_change = 0
bulletY_change = 10
bullet_state = "ready"

# Score

score_value = 0
font = pygame.font.Font("freesansbold.ttf", 32)

textX = 10
testY = 10

# Game Over
over_font = pygame.font.Font("freesansbold.ttf", 64)


def show_score(x, y):
    score = font.render("Score : " + str(score_value), True, (255,
    255, 255))
    screen.blit(score, (x, y))


def game_over_text():
    over_text = over_font.render("GAME OVER", True, (255, 255,
   255))
    screen.blit(over_text, (200, 250))


def player(x, y):
    screen.blit(playerImg, (x, y))


def enemy(x, y, i):
    screen.blit(enemyImg[i], (x, y))


def fire_bullet(x, y):
    global bullet_state
    bullet_state = "fire"
    screen.blit(bulletImg, (x + 16, y + 10))


def isCollision(enemyX, enemyY, bulletX, bulletY):
    distance = math.sqrt(
        math.pow(enemyX - bulletX, 2) + (math.pow(enemyY - bulletY
   , 2))
    )
    # Bug 1: Bullet-enemy collision
    #   Distance is only 0 if both objects are on the exact same (
   x,y) coordinate
    if distance < 0:
        return True
    else:
        return False


# Game Loop
running = True
while running:

    # RGB = Red, Green, Blue
    screen.fill((0, 0, 0))
    # Background Image
    screen.blit(background, (0, 0))
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

        # if keystroke is pressed check whether its right or left
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_LEFT:
                # Bug 2: Spaceship movement logic
                # Left arrow should set playerX_change to be a
   negative value
                playerX_change = 15
            # Bug 3: Player input
            if event.key == pygame.K_UP:  # Should be right arrow
   (pygame.K_RIGHT)
                playerX_change = 15
            if event.key == pygame.K_SPACE:
                if bullet_state == "ready":
                    bulletSound = mixer.Sound("laser.wav")
                    bulletSound.play()
                    # Get the current x cordinate of the spaceship
                    bulletX = playerX
                    fire_bullet(bulletX, bulletY)

        if event.type == pygame.KEYUP:
            if event.key == pygame.K_LEFT or event.key == pygame.
   K_RIGHT:
                playerX_change = 0

    # 5 = 5 + -0.1 -> 5 = 5 - 0.1
    # 5 = 5 + 0.1

    # Bug 4: Spaceship movement
    playerX = playerX_change  # Add playerX_change to playerX
    if playerX <= 0:
        playerX = 0
    elif playerX >= 736:
        playerX = 736

    # Enemy Movement
    for i in range(num_of_enemies):

```

```python
        # Game Over
        if enemyY[i] > 440:
            for j in range(num_of_enemies):
                enemyY[j] = 2000
            game_over_text()
            break

        enemyX[i] += enemyX_change[i]
        if enemyX[i] <= 0:
            enemyX_change[i] = 4
            enemyY[i] += enemyY_change[i]
        elif enemyX[i] >= 736:
            enemyX_change[i] = -4
            enemyY[i] += enemyY_change[i]

        # Collision
        collision = isCollision(enemyX[i], enemyY[i], bulletX, bulletY)
        if collision:
            explosionSound = mixer.Sound("explosion.wav")
            explosionSound.play()
            bulletY = 480
            bullet_state = "ready"
            # Bug 5: Increasing score
            score_value = 1  # Increment score
            enemyX[i] = random.randint(0, 736)
            enemyY[i] = random.randint(50, 150)

        enemy(enemyX[i], enemyY[i], i)

    # Bullet Movement
    if bulletY <= 0:
        bulletY = 480
        bullet_state = "ready"

    # Bug 6: Bullet movement
    if bullet_state == "fire":
        fire_bullet(bulletX, bulletY)
        bulletY = bulletY_change  # Subtract bulletY_change from bulletY (y starts from bottom)

    player(playerX, playerY)
    show_score(textX, testY)
    pygame.display.update()
```
Listing 1: Debug task

C - CONSENT FORM

Are you interested in taking part in the research project *”Collaborative Code Editing Tool”*?

This is an inquiry about participation in a research project where the main purpose is to research online pair programming from a multimodal data perspective. In this letter we will give you information about the purpose of the project and what your participation will involve.

Purpose of the project
The scientific purpose is to research collaborative programming in a real-time online environment by collecting multimodal data and finding key metrics for performance. Furthermore, the project will examine how a dashboard presenting students' progress and multimodal data may benefit teachers.

The project is part of a master’s thesis.

Who is responsible for the research project?
University: Norwegian University of Science and Technology
Faculty: Faculty of Information Technology and Electrical Engineering
Department of Computer Science is the institution responsible for the project.

Why are you being asked to participate?
The sample has been selected from the students private network and consists of computer science students and professional software developers. The sample consists of 20 pairs, i.e., 40 participants.

What does participation involve for you?
If you chose to take part in the project, this will involve that you take part in an online pair programming session along with one other participant. You will be able to communicate vocally with the other participant. It will take approximately 30 minutes.

Data will be collected from multiple sources during the programming session. The data sources and the data that will collected includes:

- Code editor:
  - Any change inside editor
- Wristband
  - Blood volume pulse
  - Galvanic skin response
  - Peripheral skin temperature
- Eye-tracker
  - Points of gaze
- Webcamera
  - Facial landmarks
- Microphone
  - Voice recordings

Participation is voluntary
Participation in the project is voluntary. If you chose to participate, you can withdraw your consent at any time without giving a reason. All information about you will then be permanently deleted. There

will be no negative consequences for you if you chose not to participate or later decide to withdraw. It will not affect your relationship with your school.

Your personal privacy – how we will store and use your personal data
We will only use your personal data for the purpose(s) specified in this information letter. We will process your personal data confidentially and in accordance with data protection legislation (the General Data Protection Regulation and Personal Data Act).

- The student (Øystein Haugen), in connection with the Department of Computer Science at NTNU, will have access to the personal data. Acces to the data will be transferred to the supervisor (Kshitij Sharma) of this project at 2023-06-11, along with the responsibilities of storage and security.
- I will replace your name and contact details with a code. The list of names, contact details and respective codes will be stored on a file speparately from the rest of the collected data.

Your age, gender and occupation may be included in a publication.

What will happen to your personal data at the end of the research project?
The project is scheduled to end 2023-06-10. At the end of the project, all data collected during the project will be transferred to the supervisor Kshitij Sharma, who will then be solely responsible for data storage and data protection. Kshitij Sharma will conduct further research on the collected data, and will keep the data until 2024-06-10 at which it will be deleted.

Your rights
So long as you can be identified in the collected data, you have the right to:

- access the personal data that is being processed about you
- request that your personal data is deleted
- request that incorrect personal data about you is corrected/rectified
- receive a copy of your personal data (data portability), and
- send a complaint to the Data Protection Officer or The Norwegian Data Protection Authority regarding the processing of your personal data

What gives us the right to process your personal data?
We will process your personal data based on your consent.

Based on an agreement with the Department of Computer Science at NTNU, NSD – The Norwegian Centre for Research Data AS has assessed that the processing of personal data in this project is in accordance with data protection legislation.

Where can I find out more?
If you have questions about the project, or want to exercise your rights, contact:

- NTNU Department of Computer Science via supervisor Kshitij Sharma (kshitij.sharma@ntnu.no) or student Øystein Haugen (oystebha@stud.ntnu.no).
- Our Data Protection Officer: Thomas Helgesen.
- NSD – The Norwegian Centre for Research Data AS, by email: (personverntjenester@nsd.no) or by telephone: +47 55 58 21 17.

Yours sincerely,

Kshitij Sharma	Øystein Haugen
Project Leader	Student
(Researcher/supervisor)	

Consent form

I have received and understood information about the project Collaborative Code Editing Tool and have been given the opportunity to ask questions. I give consent:

- ☐ to participate in a programming study
- ☐ for my personal data to be stored after the end of the project for follow-up studies (until 2023-06-10)
- ☐ for information about me/myself to be published in a way that I can be recognised (information is limited to age, gender and occupation)

I give consent for my personal data to be processed until the end date of the project, approx. 2023-06-10

(Signed by participant, date)

APPENDIX

A

D - INFORMATION LETTER

Sikt

Meldeskjema / Using Artificial Intelligence to Predict Student Performance using Eye-tracking Data an… / Eksport

Meldeskjema

Referansenummer
960234

Hvilke personopplysninger skal du behandle?

- Lydopptak av personer

Prosjektinformasjon

Tittel

Using Artificial Intelligence to Predict Student Performance using Eye-tracking Data and Facial Expressions

Sammendrag

The focus of the thesis is to develop artificial intelligence pipelines to enhance the prediction of student performance in individual learning tasks, such as video-based learning, programming, assessment. The challenge is to create pipelines that are better suited for a small number of students but with high frequency data (eye-tracking and facial expressions).

Begrunn hvorfor det er nødvendig å behandle personopplysningene

Data vil bli samlet inn og prosessert for å gjøre forskning på hvordan man kan forbedre og utvikle systemer som skal hjelpe studenter å lese og forstå kode.

Verktøy brukt for å samle inn data vil i hovedsak være gruppeintervju med frivillige studenter, for å få meninger og vurderinger rundt hvordan de opplever å lære seg å lese og forstå kode, samt meninger rundt et undervisningssystem som baserer seg på kodelesning.

Ekstern finansiering
Ikke utfyllt
Type prosjekt
Studentprosjekt, masterstudium

Kontaktinformasjon, student
Øystein Bjørkeng Haugen, oystein_bjorkeng@hotmail.com, tlf: 46412397

Behandlingsansvar

Behandlingsansvarlig institusjon
Norges teknisk-naturvitenskapelige universitet / Fakultet for informasjonsteknologi og elektroteknikk (IE) / Institutt for datateknologi og informatikk

Prosjektansvarlig (vitenskapelig ansatt/veileder eller stipendiat)
Kshitij Sharma, kshitij.sharma@ntnu.no, tlf: 41209147

Skal behandlingsansvaret deles med andre institusjoner (felles behandlingsansvarlige)?
Nei

Utvalg 1

Beskriv utvalget

Studenter ved NTNU, som studerer fag der kodelesning er en del av utdanningen deres.

Beskriv hvordan rekruttering eller trekking av utvalget skjer

Utvalget rekrutteres både selektiv, og via oppmelding på oppslag i offentlig fora.

Alder
18 - 29

Personopplysninger for utvalg 1
- Lydopptak av personer

Hvordan samler du inn data fra utvalg 1?

Gruppeintervju

Vedlegg

Intervjuguide.odt

Grunnlag for å behandle alminnelige kategorier av personopplysninger
Samtykke (Personvernforordningen art. 6 nr. 1 bokstav a)

Informasjon for utvalg 1

Informerer du utvalget om behandlingen av personopplysningene?
Ja

Hvordan?
Skriftlig informasjon (papir eller elektronisk)

Informasjonsskriv

information_letter.odt

Utvalg 2

Beskriv utvalget

Eksperter innenfor fagfeltet, fagstab ved instituttet

Beskriv hvordan rekruttering eller trekking av utvalget skjer

Selektiv rekruttert

Alder
25 - 69

Personopplysninger for utvalg 2
- Lydopptak av personer

Hvordan samler du inn data fra utvalg 2?

Gruppeintervju

Vedlegg

Intervjuguide.odt

Grunnlag for å behandle alminnelige kategorier av personopplysninger
Samtykke (Personvernforordningen art. 6 nr. 1 bokstav a)

Informasjon for utvalg 2

Informerer du utvalget om behandlingen av personopplysningene?
Ja

Hvordan?
Skriftlig informasjon (papir eller elektronisk)

Informasjonsskriv

information_letter.odt

Tredjepersoner

Skal du behandle personopplysninger om tredjepersoner?
Nei

Dokumentasjon

Hvordan dokumenteres samtykkene?
- Manuelt (papir)

Hvordan kan samtykket trekkes tilbake?

Samtykket kan trekkes tilbake når som helst, ved å informere prosjektansvarlige.

Hvordan kan de registrerte få innsyn, rettet eller slettet personopplysninger om seg selv?

Ved å kontakte de prosjektansvarlige, og disse vil være ansvarlige for å gi innsyn til opplysninger.

Totalt antall registrerte i prosjektet
1-99

Tillatelser

Skal du innhente følgende godkjenninger eller tillatelser for prosjektet?
Ikke utfyllt

Behandling

Hvor behandles personopplysningene?
- Maskinvare tilhørende behandlingsansvarlig institusjon
- Mobile enheter tilhørende behandlingsansvarlig institusjon

Hvem behandler/har tilgang til personopplysningene?
- Prosjektansvarlig
- Student (studentprosjekt)
- Interne medarbeidere

Tilgjengeliggjøres personopplysningene utenfor EU/EØS til en tredjestat eller internasjonal organisasjon?
Nei

Sikkerhet

Oppbevares personopplysningene atskilt fra øvrige data (koblingsnøkkel)?
Ja

Hvilke tekniske og fysiske tiltak sikrer personopplysningene?
- Adgangsbegrensning
- Personopplysningene anonymiseres fortløpende
- Andre sikkerhetstiltak

Hvilke

Datamaskinger er passordbeskyttet og mobile enheter har skjermlås og er passordbeskyttet.

Varighet

Prosjektperiode
01.01.2022 - 20.06.2023

Hva skjer med dataene ved prosjektslutt?
Data anonymiseres (sletter/omskriver personopplysningene)

Hvilke anonymiseringstiltak vil bli foretatt?
- Lyd- eller bildeopptak slettes

Vil de registrerte kunne identifiseres (direkte eller indirekte) i oppgave/avhandling/øvrige publikasjoner fra prosjektet?
Nei

Tilleggsopplysninger