FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Developing an Automated Pipeline for Driver Behavior Analysis

Henrique Bastos Ribeiro

Mestrado em Engenharia e Ciência de Dados

Supervisor: Professor Daniel Castro Silva

Second Supervisor: Professor Sara Ferreira

Third Supervisor: Professor Rute Almeida

July 23, 2024

Developing an Automated Pipeline for Driver Behavior Analysis

Henrique Bastos Ribeiro

Mestrado em Engenharia e Ciência de Dados

July 23, 2024

Abstract

Driving simulators are a safe and effective method of collecting physiological and driving behavior data to develop studies focusing on safety and human-machine interaction. Detecting and predicting driver drowsiness is critical to preventing road accidents. This thesis presents a study of state-of-the-art (Machine Learning Operations) MLOps pipelines and machine learning models aimed at predicting and detecting the aforementioned behaviors. It also presents the development of a dynamic data pipeline specifically designed for driving simulator studies focusing on drowsiness and distraction experiments and subsequent implementation of Machine Learning (ML) models to predict these driver behaviors. The pipeline integrates diverse data types from multiple devices (driving simulator, eye tracker, and smartwatch) and allows users to incorporate additional files. The pipeline aims to be an asset in future experiments conducted in the Laboratório de Planeamento do Território e Transportes (LPTT) laboratory of FEUP, automating processes previously done manually. The pipeline is encapsulated within a command-line interface that facilitates seamless data fusion. This interface has been highly regarded by researchers in the field, achieving an 82.5% score based on the System Usability Scale (SUS). The development of machine learning models aims to predict drowsy driving, obtaining results comparable to those found in the state of the art. A data preparation step is implemented to deal with missing values and other inconsistencies in the various devices' data. Once all these inconsistencies were solved, a step encompassing feature engineering was taken. This step focuses on creating variables useful in the context of data analysis and machine learning. Before implementing the machine learning models, data fusion was performed to create a single unified file. Various machine learning models, namely XGBoost, Random Forest, Gradient Boosting, and SVM (Support Vector Machine) were applied to predict driver drowsiness. The SVM model emerged as the most effective, achieving a 70% accuracy on the test set.

Keywords: Driver Distraction, Driver Drowsiness, Data Fusion, MLOps Pipeline, Machine Learning

Acknowledgements

Without the support and collaboration of my loved ones, this thesis would have been a truly herculean task, if not impossible. I would therefore like to express my utmost appreciation and deepest gratitude to those who have been essential pillars in the development of this thesis.

I would especially like to thank my parents, the greatest foundations of this adventure, for believing in me and always supporting my decisions; without their support and help, these deeply formative years and my entire academic career would not have been possible.

It is also imperative to acknowledge and thank all the help and support given over many months by the guidance team, namely Professor Daniel Castro Silva, Professor Sara Ferreira, and Professor Rute Almeida.

To my girlfriend, who was by my side throughout this process and greatly helped me by being a shoulder to lean on, listening to my worries, and reminding me of my value and capabilities, always with the sweetest and kindest words.

To the friends I made during my Master's degree, for their infallible tips and help throughout the development of my thesis, and also for the unforgettable moments I've had over the last two years, which I'll treasure forever.

And finally, to my great childhood friends who, since I was largely absent for the last two years, always welcomed me enthusiastically when I returned home, helping me take my mind off all my worries and anxieties.

Henrique Bastos Ribeiro

"Ever tried. Ever failed. No matter. Try again. Fail again. Fail better."

Samuel Beckett

Contents

List of Figures

List of Tables

Abbreviations

1D-CNN	One-Dimensional Convolutional Neural Network
ACC	Accuracy
ANFIS	Adaptive Neuro-Fuzzy Inference System
ANN	Artificial Neural Network
AUC	Area Under ROC Curve
AWS	Amazon Web Service
BN	Bayesian Network
Bi-LSTM	Bidirectional Long-short Term Memory
CD	Continuous Delivery
CI	Continuous Integration
CNN	Convolutional Neural Network
CT	Continuous Training
CV	Cross Validation
DL	Deep Learning
DST	Dempster-Shafer Theory
DT	Decision Tree
ECG	Electrocardiogram
ED	Euclidean Distance
EEG	Electroencephalogram
EMD	Empirical Mode Decomposition
EOG	Electrooculogram
FL	Fuzzy Logic
GCS	Google Cloud System
LDA	Linear Discriminant Analysis
LOSO-CV	Leave-One-Subject-Out Cross-Validation
LPF	Linear Predictions Filter
LSTM	Long-short Term Memory
MED	Multivariate Euclidean Distance
ML	Machine Learning
NB	Naive Bayes
NREM	Non-Rapid Eye Movement
PDA	Partial Dependency Analysis
PERCLOS	Percentage of Eyelid Closure over the Pupil over Time
REM	Rapid Eye Movement
RF	Random Forest
RBF	Radial Basis Function
RRI	R-R Intervals
SFFS	Sequential Floating Forward Selection
SHAP	SHapley Additive exPlanations
SVM	Support Vector Machines

Chapter 1

Introduction

This introduction outlines the aim of the dissertation and provides context. Its objective is to address the significant road safety issue by studying driver drowsiness and distraction. Road safety is a critical global health and development challenge that affects millions of lives each year. Despite technological and infrastructural advancements as well as new safety regulations, road traffic deaths and injuries remain a significant concern across the globe. In 2021, an estimated 1.19 million people died due to road traffic incidents. Road traffic crashes are a leading cause of death worldwide, particularly among young people. In 2019, these crashes were the main cause of death for children and adolescents aged 5-29 years and the 12th leading cause of death for all age groups. The social and economic impact is profound, with two-thirds of deaths occurring among people of working age (18-59 years), resulting in significant health, social and economic consequences for societies (World Health Organization, 2023).

1.1 Problem Definition and Motivation

The primary factor contributing to road accidents is "user error," accounting for approximately 95% of all accidents (European Comission, 2003). Although several factors, such as alcohol consumption and exceeding the posted speed limit, contribute to human driving errors, detecting elements like distraction, fatigue, and drowsiness can be particularly challenging due to their subtle and subjective nature (Lenné and Jacobs, 2016). States such as distraction, fatigue, and drowsiness have prompted the development of detection systems integrated as standard features in certain vehicles or available as aftermarket devices. Although many detection systems are already very effective in assessing driver alertness, they often end up failing to distinguish and to

predict these events (Naujoks et al., 2016). The aforementioned existing systems face a challenge created by the common occurrence of false, unnecessary, distracting, and sometimes wrong alarms. The topic of this dissertation comes in the aftermath of the Brain Behavior Analysis using the most advanced Artificial Intelligence and Computer Vision (BBAI) project[1], which studied topics such as the ones mentioned above. Two types of experiments were implemented using a driving simulator: one focused on the drowsiness state and the other on distraction. More information about this project will be detailed in chapter 2.

Drowsiness is often associated with fatigue; however, these two terms do not have the same meaning. Fatigue can manifest itself both physically and mentally and may disappear after a period of rest; nonetheless, drowsiness can only be eliminated after sleep and manifests itself through a great difficulty in staying awake. These states can be influenced by a number of factors, such as physiological parameters, light settings, roadway geometry, landscape monotony, traffic conditions, or time of the journey (Soares et al., 2020a). Distraction, a different state that is often manifested when driving, is often caused by an activity performed by the driver that leads his attention away from the roadway and the driving itself, which may lead, such as drowsiness, to potentially dangerous driving (Aksjonov et al., 2019).

Driving simulators have been used since the 1970s to study driver drowsiness and distraction under a controlled, safe environment (Lenné and Jacobs, 2016). Using driving simulators in research offers a plentitude of advantages, namely in drastically reducing safety and ethical concerns by avoiding dangerous and unexpected events that could happen in a naturalistic setting. Apart from this, the simulated environment allows the manipulation of scenarios to induce drowsiness and distraction whilst ensuring methodological control, reproducibility, and standardization (Soares et al., 2020a). However, it is obvious that driving in a simulator is not the same as driving in a naturalistic setting. In the domain of the research being conducted on this project, it is crucial to collect different types of variables and data - both driving-related and physiologic characteristics. The variety of data and the large number of data sources means that a need arises - the need to synchronize the data with the aid of a processing pipeline. Automating synchronization is essential because manual data handling is prone to errors, time-consuming, and unsustainable, especially as data volume grows exponentially. This data has the potential to be analyzed using machine learning (ML) models, particularly for detecting and predicting drowsiness. In the case of distraction, which is a complex driver behavior since there are many types of distraction (e.g., cognitive versus visual), a distinct approach needs to be explored depending on the distraction context and study objectives.

[1][NORTE-01-0247-FEDER-069809]

1.2 Objectives

The primary objective of this dissertation is to design and develop a comprehensive pipeline that integrates data from multiple devices used in a driving simulator and related systems. The work developed in this project is primarily described as a pipeline throughout the document, emphasizing its sequential and modular processing steps. However, on a broader scale, it might also be defined as a framework because it encompasses a comprehensive set of tools that provide a standardized approach to building and deploying an application. The pipeline aims to overcome the existing time-consuming and error-prone processes related to the data extracted from the DriS simulator that take place in the Laboratório de Planeamento do Território e Transportes (LPTT) laboratory of FEUP (Soares et al., 2020b). Furthermore, using the data collected under the BBAI project, this dissertation will explore machine learning models and, based on the state of the art, implement these models to detect and predict drowsiness events specifically. With this in mind, to achieve the proposed objectives, it is necessary to answer the following research questions:

RQ1. What insights can machine learning models present regarding the study of drowsiness and distraction events?

RQ2. Using the data extracted from the BBAI experiments, is it possible to develop ML models for predicting drowsiness that achieve results comparable to those implemented in the state-of-the-art?

The initial aim is to automatically standardize the various files into a single file through a user-friendly interface. The initial various files are the result of the direct extraction from the devices after the driving experiments. After data preparation, feature engineering, and data fusion, these files will be transformed into refined, clean data. Automated pipelines ensure consistency, reduce the likelihood of human error, and significantly speed up data processing from collection to analysis. Additionally, automation enables the system to scale efficiently, handling more data sources and complexity without requiring additional resources. Finally, a guide, A, will be provided to support future applications and developments. Once this is done, machine learning methods will be explored, considering the data from the BBAI experiments. Considering the design and goals of the experiments, only the drowsiness data is selected for ML implementation with the objective of detecting driver drowsiness.

It is important to note that one of the key contributions of this project is its potential to assist future researchers of the LPTT laboratory of FEUP who may choose to pursue projects similar to BBAI. The pipeline has the capacity to help researchers, as it carries out processes that in past

studies were done manually. This will make it much easier to analyze data in the future and will save a lot of time, which can be devoted to other tasks. The main contribution of implementing the models is that they can predict drowsiness for the general public without the need for a period of training ML models personalized to each driver.

1.3 Dissertation Structure

In addition to this introduction, which explores the motivations for this work and defines the problem and the objectives, the initial part of this dissertation starts with chapter 2 (Contextualization), where the topics introduced in the Introduction are further explored. Here, the BBAI project is given a bigger focus, the data structure is presented, and some important concepts are defined.

Chapter 3 (Literature Review) and chapter 4 (Methodology) represent the research and the proposal. Chapter 3 (Literature Review) presents and analyzes research on state-of-the-art topics related to this dissertation. Firstly, there is a focus on MLOps pipelines, followed by a study on ML modeling related to detecting and predicting driver behaviors (drowsiness and distraction). In chapter 4 (Methodology), a work plan is proposed for the remainder of the work, referencing useful approaches to the remainder of the work.

The next two chapters, 5 (Pipeline Development) and 6 (ML Modelling), present what was implemented in the development of the work. At the end of each chapter, there is also a validation section (in the case of the pipeline) and a results section (in the case of the predictive models).

At last, chapter 7 (Concluding Remarks) presents a comprehensive summary of the findings, implications, and a further works section.

Chapter 2

Contextualization

In this chapter, the BBAI project and its experiments are explained in detail, as well as the data structure and sources. Additionally, this study introduces important concepts and the main processes of a machine learning project.

2.1 BBAI Project

The BBAI project was created with the aim of contributing to the detection and mitigation of deficient psychomotor efficiency in the context of driving vehicles, such as drowsiness and distraction. This project aimed at research and innovation, and its objective was to develop algorithms capable of detecting drowsiness and distraction and predicting the former using biometric data extracted during simulated driving experiments. The data collected for the BBAI project will be used to conduct the dissertation.

For the experiments, a customized Volvo 440 Turbo adapted for the purpose of a driving simulator was used. This driving simulator records various parameters related to driving and interaction with the vehicle. The simulator features multiple screens, including those integrated into the rear-view mirror, side-view mirrors, a touchscreen, and a 180º curved projection system that surrounds the driver and replicates a realistic road environment and its surroundings (Soares et al., 2021). The simulator collects several driving-related parameters and driver-vehicle interactions. It includes an In-Vehicle Information System (IVIS) consisting of a tactile screen, allowing the display of requests and information input. The simulator is also equipped with an eye tracker situated in the tableau of the car that extracts data regarding the participant's vision. A sound system with four speakers is positioned at the rear of the room to replicate various auditory cues, such as braking and acceleration sounds, the noise of virtual vehicles, and ambient environmental sounds.

A topic worthy of further analysis is the experiments conducted in the BBAI project. As stated before, one of the experiments focused on distraction, whereas the other focused on drowsiness.

Distraction Experiment: In the distraction-related experiment the driving session had a duration of approximately 25 minutes. Throughout that time, the drivers were prompted with four distinct instructions that demanded the execution of various tasks while driving, such as finding an object in the backseats. These instructions are conveyed through audio and on-screen indications via the IVIS system. These messages and tasks might have been associated with critical driving events (such as a spontaneous need for hard braking) that happened at the same time as the tasks referred. This way, the driver needed to react quickly in order to avoid a possible collision. This setup enabled the analysis of reaction times;

Drowsiness Experiment: In the drowsiness-related experiment, the participant drove for around 90 minutes in an extremely monotonous simulation specifically created with the purpose of creating driver drowsiness. Throughout this experiment the participant was asked to report his level of drowsiness through the IVIS system. The scale of drowsiness used for this experiment was the Karolinska Sleepiness Scale, a frequently used scale for analyzing subjective sleepiness that ranges between 1 and 9 (Åkerstedt and Gillberg, 1990), that was collected every five minutes during the experiment.

2.2 Data Structure

The variety of sources leads to the existence of data in various formats. The data was converted to the formats used in this project through a set of procedures done by the researchers involved in the BBAI project. There was a different procedure for each device. The files and their properties are presented in Table 2.3. This table presents the files extracted from each device, as well as their format, a summary of their features, the sampling frequency of the data in each file, the dependencies of the files in relation to each other, and a few other relevant notes.

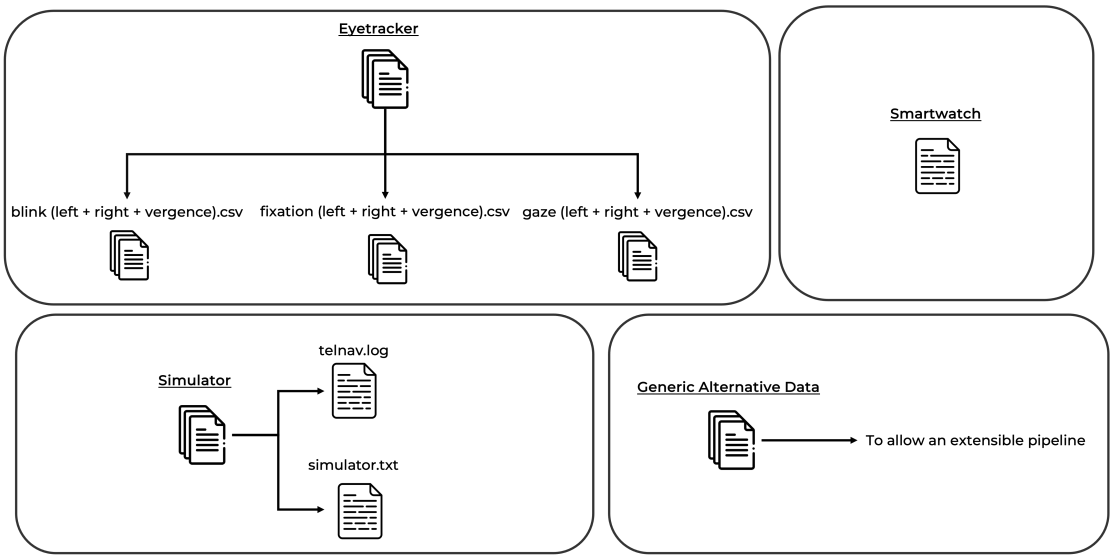

Figure 2.1: Multi-Source Data Structure for Integration in Processing Pipeline

The lack of documentation on the data structure poses a challenge for BBAI project researchers, requiring significant time and effort to comprehend the data format and determine effective utilization strategies by themselves. This process is time consuming and constituted a big constraint for the project.

Two data files are extracted from the simulator. One of them is related to driving behavior, which includes variables such as lane position, steering wheel angle, pedal activity, and speed. The full list is present in Table 2.1. The other file contains the identification number of the driver, the timestamps and the self-reported data.

There are a total of nine data files referent to the eye tracker data - three related to the blinking of the participant, three related to its gaze, and the last three related to the fixation of the participant's sight. The fixation files contain events defined as at least 6 successive samples in which the distances between pupil positions do not differ by more than 5 samples. The blink files represent time intervals where the eye is closed, in 3 or more consecutive samples. It's important to emphasize that the blink file doesn't necessarily contain data relating to the blink of an eye. The file in question was not assigned this name by the project team. It is an equipment output file for which there is no access to the theoretical basis of the threshold or the name. The variables of the eyetracker files are presented in Table 2.2.

The file originating from the smartwatch contains a multitude of variables. However, these variables are not within the scope of the project, as they relate to a "sports mode" of the smartwatch designed to capture data when the user is engaged in physical activity. In this case, as the

user is not engaged in this sort of activity, only the information related to the timestamp and the participant's heartbeat is considered from this file.

It is crucial to highlight the significance of the 'eixo offset(m)' feature, present in the .txt file, which refers to the data of the driving simulator, as it has a considerable impact on subsequent stages of feature engineering. This feature represents the distance to the center divider (left) in meters relative to the camera (positioned in the tableau, under the rearview mirror). The offset depicted in Fig.2.2 represents the measurements in the drowsiness experiment. It was not possible to obtain the measurements for the distraction experiment.

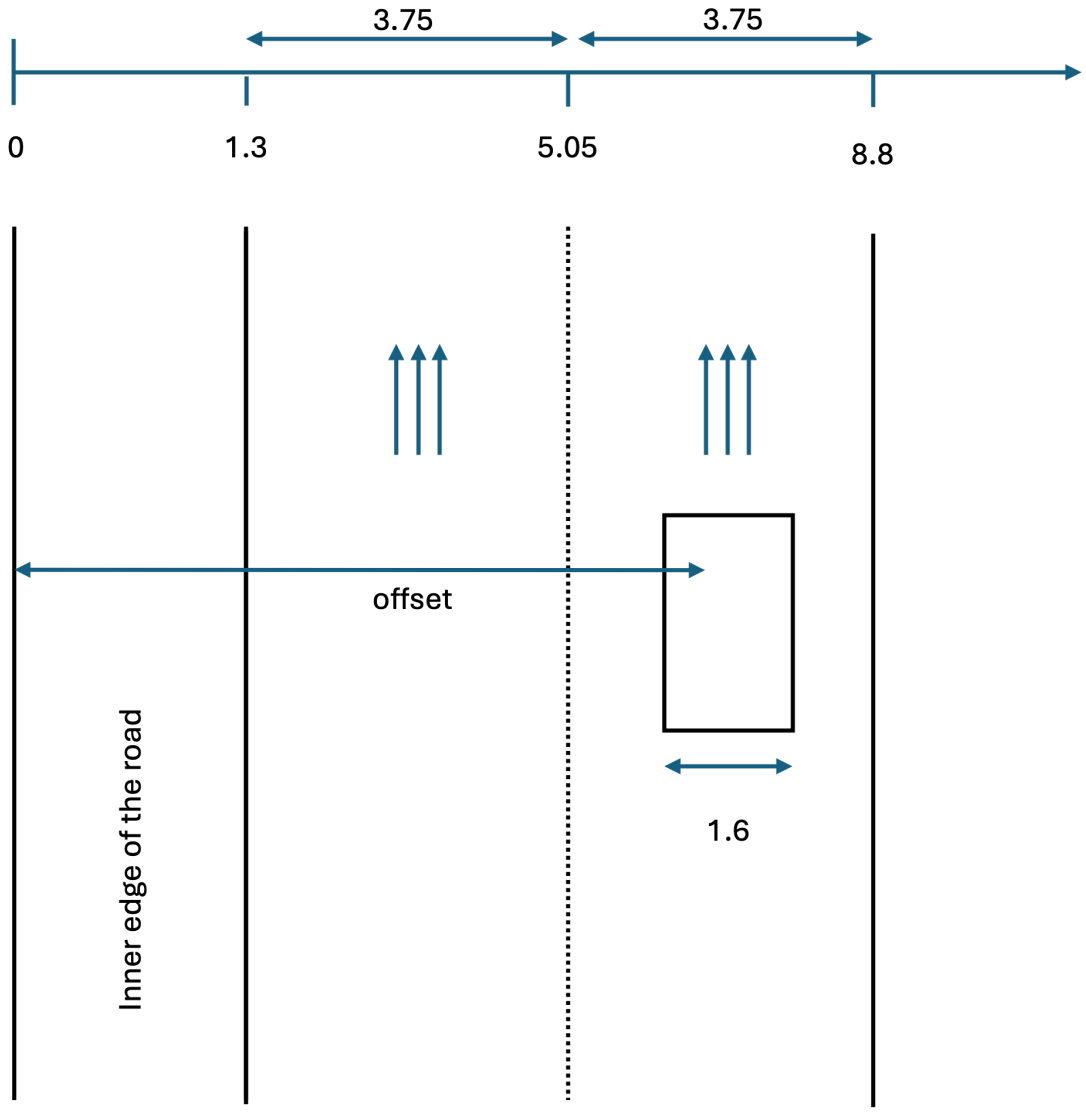

Figure 2.2: Diagram representing the offset (Drowsiness Experiment)

Table 2.1: Description of driving simulator variables

Variable	**Description**
time	instant of reading relative to the start of data recording, in seconds
X	absolute X coordinate (relative to the world)
Y	absolute Y coordinate (relative to the world)
Z	absolute Z coordinate (relative to the world)
IVIS	equals 1 (true) when an IVIS (In-Vehicle Information System) event is triggered
roaddist	distance from the driven vehicle to the start of the road, in meters
totaldistance	distance traveled by the driven vehicle in meters
gear	gear engaged in the manual transmission
blinkers	(2=Left, 4=Hazard, 5=Right)
speed	longitudinal speed in km/h
acc_state	Adaptive Cruise Control, 0-standby; 1-set; 2-set&following; 3-set&following&Alarm
acc_speed	Adaptive Cruise Control, Acc_Speed (km/h)
acc_dist	Adaptive Cruise Control, ACC_distance (s)
lane_offset	distance to the center divider (left) in meters, relative to the camera (driver's position)
wheel_angle	wheel angle relative to the car (+left, -right), in degrees
car_angle	absolute orientation angle of the driven vehicle (heading/yaw), in rad
car_road_angle	car orientation angle relative to the road, in rad
accelerator	accelerator pedal value, in percentage
brake	brake pedal value, in percentage
steering_wheel	steering wheel angle in degrees
steering_filtered	filtered steering angle (low pass butterworth second order filter, time domain) freq cut=0.6Hz
dist_front	distance to the vehicle in front (bumper to bumper) in meters
speed_front	longitudinal speed of the vehicle in front, in km/h
TTC (s)	Time-to-collision in seconds
TH (s)	Time headway to the vehicle in the front (seconds)
DH (m)	Distance Headway to the vehicle in the front (meters)
TTC (lim)	TTC limited to values below 15 seconds
TH (lim)	TH limited to values below 3 seconds
DH (lim)	DH limited to values below 50 meters
avg_speed	average longitudinal speed of the guided vehicle

Table 2.2: Eyetracker Files and Variables

Vars / Files	Gaze Left/Right	Gaze Vergence	Fixation Left/Right/Vergence	Blink L/R/Vergence
Start Time (secs)	Time reference			
Display	-1: pupil not visible, 1: pupil visible		1	1
XPos	Ordinate pupil position in the sample			
YPos	Abscissa of the pupil position in the sample			
Pupil Diameter	Visible pupil diameter	-1: pupil not visible. 0: pupil visible in at least one eye	N.A.	
Stop Time (secs)	N.A.		End time of the event	N.A.
Duration (secs)	N.A.		Event Duration	N.A.
Average Pupil Diameter	N.A.		Average visible pupil diameter	N.A.
Average Pupil Area	N.A.		Average visible pupil area	N.A.
Std. Dev. Pupil Diameter	N.A.		Standard deviation of visible pupil diameter	N.A.
Std. Dev. Pupil Area	N.A.		Standard deviation of visible pupil area	N.A.
Median Pupil Diameter	N.A.		Median visible pupil diameter	N.A.
Median Pupil Area	N.A.		Median visible pupil area	N.A.
Samples In Fixation	N.A.		Number of frames in the event	N.A.
ResumeTime_secs_	N.A.		N.A.	End time of the event
ZeroSampleCount	N.A.		N.A.	Number of frames with pupil not visible.

Table 2.3: Table describing the features and properties of the data. MV - Missing Values; KSS - Karolinska Sleeping Scale; ID - Identification Number

File	File Format	Features	Sampling Frequency (Hz)	Dependencies	Other Notes
Simulator - Vehicle	.txt	Timestamp, participant ID, velocity, acceleration, offset, etc.	23.33	None	None
Simulator - Telnav	.log	Timestamp, KSS (drowsiness), Event (distraction)	0.003	None	None
Eye Tracker - Gaze	.csv	Timestamp, Eye Coordinates (X and Y), Pupil Diameter, Display (0 if there are MV)	61.33	None	Is subdivided into three files: Left, Right and Vergence
Eye Tracker - Fixation	.csv	Timestamp, Eye Coordinates (X and Y), Pupil Diameter, Display, Stop and Start time, Pupil diameter statistics (std. dev., mean, etc.), samples in fixation	Variable	Extracted from the gaze files	Is subdivided into three files: Left, Right and Vergence
Eye Tracker - Blink	.csv	Timestamp, Eye Coordinates (X and Y), Display, Start Time, Resume Time, Duration, Zero Sample Count	Variable	Extracted from the gaze files	Is subdivided into three files: Left, Right and Vergence
Smartwatch - Vehicle	.csv	Sport-related features (Heart Rate, Slope, Watts, etc.)	0.96	None	None

As the pipeline is intended to be extensible to other types of data, it will also be possible to use alternative generic data, as represented in Fig.2.1.

2.3 Important ML Concepts

This section offers a concise introduction to key concepts, contextualization of machine learning projects, and their typical steps as potentially relevant to this work.

One of the concepts mentioned throughout this study is **DevOps**, short for Development and Operations. It emerged as a significant topic in software engineering in the late 2000's. It represents a movement aimed at harmonizing the efforts of development and operations teams through the integration of organizational structure, practices, and culture. This alignment enables efficient development processes and ensures scalable and reliable operations (Pang et al., 2020);

A **machine learning algorithm** is a software program that enhances its performance by learning from data or examples (Huang et al., 2023). It utilizes extensive data analysis to identify patterns within the data, allowing for the automatic creation of mappings between complex, nonlinear inputs and outputs (Chen and Zhao, 2023). The processes involved in an ML project can be described as follows:

1. **Data Extraction**: The process of transforming unstructured data, which is data that is not consistent and that may present multiple data types into a format suitable to be analyzed and further developed (Saha et al., 2017);

2. **Data Preprocessing**: Data preprocessing is crucial in preparing raw data for analysis and processing. This step is vital as the quality and format of the data can greatly impact the performance of predictive models and analytical algorithms. This process contains the following tasks: data cleaning, normalization, feature engineering, and imbalanced data management. Its primary goal is to make the data usable and interpretable by algorithms, thereby improving the accuracy and efficiency of subsequent analysis (Kang and Tian, 2018);

3. **Model Selection and Training**: After implementing several algorithms, the model selection process can be categorized as choosing the most appropriate model from a collection of multiple models based on its performance on the data. Evaluation through metrics such as accuracy, precision, and F1 score is used to perform the selection of the best model (Ding et al., 2018). Model training is the process by which a model uses data and extracts

patterns to be able to generalize to new data not presented in the training stage, where it is provided with correlations and prediction generation to accomplish the specified task [1];

4. **Model Evaluation**: The evaluation of machine learning model performance involves using a separate test data file that the model has not previously encountered during training. This way, a representation of how the model would perform in a real-life setting is depicted. The evaluation metrics depend on the type of problem, such as accuracy, precision, recall for classification problems, and mean squared error for regression problems (Varoquaux and Colliot, 2023);

5. **Model Deployment**: Integrating the trained model into the production environment allows it to make predictions or decisions based on new data. The deployment strategies can vary significantly depending on the application and operational requirements (Castro-Lopez and Vega-Lopez, 2019).

MLOps, short for Machine Learning Operations, refers to a collection of practices and tools created to deploy machine learning models in a production environment efficiently. It is a fusion of Machine Learning and DevOps principles. DevOps; however, comprises a set of practices aimed at shortening the software development cycle by bridging the gap between software development and operations teams, ultimately improving agility and productivity (Symeonidis et al., 2022). **Data Fusion** is the process of integrating data from different sources to produce more valuable, precise, and reliable information about the behavior of a system (Hassani et al., 2024).

[1] https://docs.aws.amazon.com/machine-learning/latest/dg/training-ml-models.html

Chapter 3

Literature Review

The literature review will be divided into two crucial parts to the development of the dissertation. The first focuses on the MLOps pipeline which will be implemented in order to automate the task of processing the data and the modelling phase. Afterwards, a study of the ML algorithms applied in state-of-the-art literature in the domain of drowsiness and distraction in driving simulators will be conducted, as well as in naturalistic scenarios.

3.1 MLOps

Created based on Agile methodologies, DevOps emerged in 2008/2009 to deal with challenges related to slow software rollout and aims to automate and streamline software development, testing, and software deployment[1]. DevOps' practices allow software development (dev) and operations (ops) teams to hasten delivery by leveraging automation, improving collaboration, smoothing feedback[2], and promoting iterative improvement, thus solving the challenges that troubled the software development field (Subramanya et al., 2022).

Throughout the years, ML started to gain popularity and with it a new set of challenges, such as the need to accelerate the deployment stage of ML projects, was starting to rise. MLOps was created around 2015 to deal with these challenges by combining the practices of DevOps, ML and data engineering (Bodor et al., 2023), as shown in Fig. 3.1.

[1]More information on: https://everythingdevops.dev/a-brief-//history-of-devops-and-its-impact-on-software-development/

[2]More information on: https://about.gitlab.com/topics/devops/

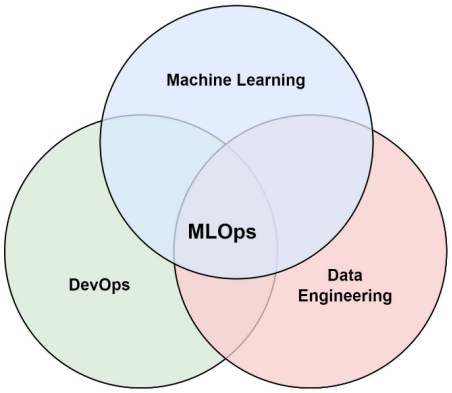

Figure 3.1: MLOps' Influences (Testi et al., 2022)

MLOps aims to automate and monitor all stages of ML system development, which is an iterative process of pushing the latest and best ML models into production (Testi et al., 2022). While continuous integration (CI) and continuous delivery (CD) are practices present in DevOps, its adaptation to the field of ML is one of the principal components of MLOps (Bodor et al., 2023). Continuous integration is the practice of code integration written at frequent intervals, while continuous delivery is the delivery of software (also in frequent intervals). Continuous training (CT) is a phase - whose importance must also be emphasized - that does not exist in DevOps projects, and its aim is to retrain the ML model when needed. Continuous delivery is the process of deploying the application to different deployment instances, such as test or production. The key difference is in the way the code is shipped - in continuous delivery, code is often shipped manually to production or test instances, while in the other process, the deployment of code to instances is automated. In both cases, the code is kept ready for deployment at any time (Subramanya et al., 2022). Within MLOps, its integration into a classic software engineering project brings a new set of requirements and fresh features, therefore there is a necessity of a comprehensive view regarding ML projects for more effective implementation and management. A crucial part of MLOps is also the focus on the communication flow and knowledge sharing between human resources - such as data scientists, data engineers, analysts, and so on - involved in an ML project (Bodor et al., 2023).

A concept often used within MLOps is the so-called Maturity Levels - an MLOps system can be classified at a corresponding level depending on its level of automation. A level of automation refers to the degree to which tasks, operations, and workflows are performed by systems, software, or machines without human intervention. Although there is no universal model for maturity levels, Google and Microsoft have proposed their own models. After examining Google's proposed maturity model, it is evident that it comprises three distinct levels. It describes three

levels: level 0, which refers to manual, script-driven, and interactive processes; level 1, which aims to continuously train the model by automating the ML pipeline and includes characteristics such as allowing rapid experimentation, continuous training of the model in production, continuous delivery of models, and pipeline deployment that automatically and recurrently runs to serve the trained model as the prediction service; and level 2, which focuses on the integration of CI/CD and includes components such as source control, test/build services, deployment services, and model registry [3]. Microsoft proposes a model with five levels of MLOps implementation. Level 0, 'No MLOps,' is characterized by a lack of automation, with the entire process being carried out manually. Level 1, 'DevOps but no MLOps', involves automated builds but limited feedback on model performance and difficulty reproducing results. Level 2, 'Automated Training', includes automated model training and management, but manual releases. At level 3, 'Automated Model Deployment', releases are already automated. At the fifth and final level, 'Full MLOps Automated Operations', the entire system is automated and easily monitored. The production systems provide information on how to improve and, in some cases, automatically improve with new models [4]. Level 2 of the Google maturity levels pyramid, ML pipeline automation (Symeonidis et al., 2022), seems to be the most aligned with this project's goal. Regarding the project, there is a focus on the development and automation of a data pipeline and a modelling pipeline.

John et al. (2021) establishes a framework that identifies the key activities in the adoption of MLOps and is divided into three distinct but interrelated pipelines (a data pipeline, a modelling pipeline and a release pipeline), two of which overlap with the objective of this project, the data and modelling pipelines. This framework aims to collect relevant data from various data sources, preprocess the data and perform feature extraction in the data pipeline. Once a suitable ML model is selected, after an experimentation and optimization phase in the modelling pipeline, this model is evaluated and packaged for production deployment. In case the performance worsens, the model is trained once again by initiating a new data feedback loop.

3.1.1 Data Pipeline

Data processing pipelines in ML can be defined as a sequence of data processing steps designed to transform raw data into a format suitable for training and deploying ML models. These pipelines are instrumental in the entire data lifecycle, from data collection to the deployment of ML models. A data processing pipeline should be consistent (i.e., reduce the probability of

[3]More information on: https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning

[4]More information on: https://learn.microsoft.com/en-us/azure/architecture/ai-ml/guide/mlops-maturity-model

errors that may occur during manual processing), time efficient, and reproducible, and it should take resource optimization into account. The quintessential techniques used in efficient data processing pipelines can be divided into 3 steps that lead to the modelling pipeline: a) Data Collection and Ingestion, b) Data Cleaning and Preprocessing, c) Feature Engineering (Blessing and Klaus, 2023).

Mumuni and Mumuni (2024) discuss automated data pipelines by referencing the variation in automation provided by different approaches, initially focusing on the automation of data processing in ML pipelines, followed by an analysis of general approaches for automating data preprocessing tasks. When applied to deep learning (DL) pipelines, data processing tasks tend to be implemented within the framework in order to be trained end-to-end. It is possible to extract different types of features at different semantic levels through the hierarchical layers of ANNs commonly present in deep learning pipelines (i.e., in computer vision problems), that is, when in the learning process, the model itself performs feature selection and low-level data processing in response to performance results. The automation aspect of data processing and feature engineering varies in complexity, flexibility, and performance - while the lower level of the spectrum relies on a hand-crafted approach, more advanced methods use mechanisms, such as automated ML (AutoML), to perform the processing of the data. Several of these methods are able to perform data processing automatically without the intervention of human operators. A typical AutoML pipeline, shown in Fig. 3.2) is constituted by an initial phase concerned with data processing tasks followed by tasks related to the modelling itself.

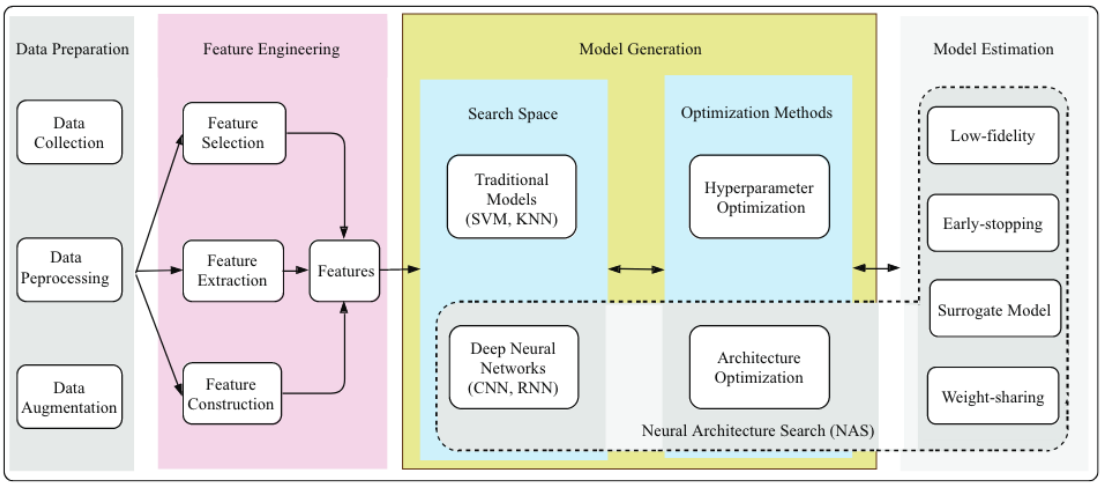

Figure 3.2: AutoML Pipeline Framework Representation (Mumuni and Mumuni, 2024)

Although the automation level may be variable, Mumuni and Mumuni (2024) states that all automated preprocessing approaches use ML algorithms to maximize performance. Table 3.1

summarises the general approaches for automating data preprocessing tasks.

Table 3.1: General Approaches for Automating Data Processing Tasks (Mumuni and Mumuni, 2024)

Method	**Description**
Basic	Standalone preprocessing, usually manual or with the aid of generic data processing tools (e.g., image processing libraries for manipulating images).
Analytical	Automated processing within deep learning pipelines using explicitly formulated analytical relations.
DL	Deeply learned processing modules within ML pipelines.
AutoML	End-to-end processing, model selection, and hyperparameter tuning using AutoML.

As previously highlighted, a critical part of this project lies in the data pipeline, where there is a particular focus on the automation of data fusion from a variety of heterogeneous sources that provide data for this project.

Multimodal fusion is the concept of integrating information from multiple modalities/data sources with the goal of predicting an outcome measure through ML models. Baltrusaitis et al. (2019) classify multimodal fusion into two separate categories: model-agnostic and model-based. Whilst the model-agnostic approach is not dependent on a specific ML model, in model-based approaches, data fusion is not only an additional step but a fundamental part of the model's architecture. Model-agnostic multimodal data fusion can be split into three levels: early (i.e., feature-based), late(i.e., decision-based), and hybrid (Atrey et al., 2010).

Early (or feature-based) fusion is a technique often described as an early type of multimodal representation learning that aims at the integration of features right after extraction, typically by concatenating their representations, and only requires the training of a single model. The challenges faced include ensuring time synchronization among multimodal features extracted at different times, standardizing data into a uniform format before fusion, and managing the increasing complexity of learning cross-correlations with more modalities involved. To overcome these hurdles, careful preprocessing and model design is necessary to effectively harness the full potential of feature-level data fusion (Atrey et al., 2010). Numerous scholars have embraced the early fusion method in fields such as multimedia analysis. For instance, Nefian et al. (2002) used early fusion to combine audio and visual attributes to improve speech recognition.

Late fusion (or decision-based) integrates decision values from each modality after they have made independent decisions, while providing more flexibility modelling-wise since it allows for

the training of different models for each modality. This type of data fusion is often used after separately cleaning and preprocessing data from each one of the modalities/data sources. Decision-based fusion has several advantages, including the ease of combining decisions due to their uniform representation across modalities, such as audio and video, unlike the varied representations in feature-level fusion. It allows for scalability and flexibility in the fusion process, enabling the use of modality-specific analysis methods. However, it cannot utilize correlations between modalities at the feature level, which can complicate and prolong the learning process for individual classifiers (Atrey et al., 2010). Iyengar et al. (2003) used late-fusion by combining the decisions from a face detection system and a speech recognition tool, along with their synchrony score, using two methods: a linear weighted sum and a linear weighted product. Hybrid fusion is an attempt at combining aspects of both previously mentioned approaches, which has been successfully implemented in various domains and is able to utilize the advantages of both early and late fusion strategies (Baltrusaitis et al., 2019).

Model-based multimodal data fusion is often based on a set of mathematical models that can be defined as: a) probability-based, b) AI-based, and c) methods based on theory of evidence. Probability-based is one of the most widely used techniques in data fusion. This approach uses probability theory, and its most common approaches are Bayesian theory-based, copula theory-based approaches, and Markov chain-based techniques. Zhang et al. (2019) introduces techniques based on a statistical tool, regular vine copulas, for recognizing activities using data from various sensors. AI-based data fusion integrates data from various sources, such as Internet of Things (IoT) devices, sensors, and brain signals, using deep learning and fuzzy logic. This enhances decision-making and efficiency. Examples include emotion recognition, smart learning environments, efficient resource management, and emergency response. AI-driven fusion systems combine different data types to unlock new insights and functionalities, improving productivity and understanding in both everyday and critical applications. This form of data fusion has been applied across diverse domains. For example, in Zheng et al. (2019), the fusion of brain wave signals and eye movement was utilized for emotion recognition. Additionally, in Mehmood et al. (2017), a personalized teaching and learning framework was introduced, integrating signals from different modalities. Furthermore, in Pires et al. (2016), signals from various sensors in the phone were combined to comprehend a user's daily activities. The theory of belief, initiated by Arthur P. Dempster, introduces a structured approach to managing uncertainty. Advanced fusion methodologies, such as integrating ontological data for enriched knowledge bases (Bellenger and Gatepaille, 2011) and combining LiDAR with multi-spectral imagery for precise building detection (Rottensteiner et al., 2005), have been enabled through the utilization of Dempster-Shafer Theory (DST). These applications illustrate DST's ability to

enhance data analysis and decision-making by effectively integrating various sources of information under uncertain conditions (Gaonkar et al., 2021).

In Zhang et al. (2018), data fusion is focused on multi-source heterogeneous data, which is similar to multi-modal data but considers more diversity in data types. As stated in the article, Zheng (2015) proposed a division for heterogeneous data fusion methods that consisted of: a) stage-based, b) feature-level, and c) semantic meaning-based. Semantic meaning-based data fusion is then subcategorized into: 1) multi-view-based methods, 2) similarity-based methods, 3) probabilistic-dependency-based methods, and 4) transfer learning-based methods. Stage-based data fusion utilizes different data at various stages in data mining analysis, allowing the fusion of multi-source heterogeneous data throughout the process. The stage-based initial intention aimed at using multi-source data to reach a single goal. This type of data fusion is often used together with other data fusion methods, as seen in the study conducted by Yuan et al. (2015). The authors initially utilize data from road networks and taxi paths to create a map of an area. They then combined details from points of interest (POIs) with the map using diagrams. Finally, they integrated the information further using a probability-based method and diagrams in a stage-based approach. Feature-based, also described as early fusion, was already mentioned and explained above. The semantic meaning-based approach comprehends the insights of each dataset and the relations between features across different datasets. As previously mentioned, this approach is divided into four subcategories. Multi-view-based data fusion integrates data from multiple perspectives or 'views' of the same object to achieve a comprehensive and accurate representation. Each view, like different feature sets of data from sensors or various biometric data of a person, offers unique and complementary insights. The fusion process in multi-view learning leverages these diverse datasets, finding consensus among them and exploiting their distinctiveness. Similarity-Based Data Fusion presents the concept that different objects may show similarities across various metrics. When one object lacks data, information from a similar object (i.e., subject of analysis) can be utilized. With multiple datasets for each object, multiple similarities are calculated and can reinforce each other, enhancing the overall correlation. This approach is particularly useful in augmenting sparse datasets with information from denser ones. Combining multiple datasets of two objects often leads to more accurate similarity estimations. Probabilistic Dependency-based fusion relies on probabilistic graphical models to represent conditional dependencies between variables. Thanks to the properties of these graphical models, this method results in a better understanding of the probabilistic relationships inherent in the data. Lastly, Transfer Learning-based fusion involves applying knowledge extracted from one domain to a different but related domain, where feature spaces or data distributions might differ. Zheng (2015) states that feature-based data fusion requires a large number of labeled instances as training data, whereas semantic meaning-based methods can be applied to a dataset with a small

number of labeled instances. Fixed data sources allow for the straightforward application of multi-view and similarity-based fusion methods due to their consistency. However, they face challenges when dealing with variability in data acquisition. It is concluded that, when given the same amount of training data, semantic meaning-based approaches are more effective than straightforward feature-based methods due to the dependencies and correlations between features. An overview of the techniques described so far, with a brief description, can be found in table 3.2.

In a similar context to the project this dissertation is focused on, Wu et al. (2023) conduct a comparative study of multi-modality feature fusion and selection in the domain of physiological measurements for driving drowsiness. The study compares 5 distinct modalities, namely forehead EEG (Electroencephalogram), forehead EEG without ocular artifacts, EOG (Electrooculogram), RRI (R-R intervals), and breath, under the requirements of portable or wearable in-vehicle systems. In order to prevent overfitting and computation load, the aforementioned study implemented Sequential Forward Floating Selection (SFFS), a feature selection method that combines sequential backward elimination and sequential forward selection and aims to maximize a criterion function to exclude or include features. A correlation coefficient was used to measure the relationship between features and drowsiness level, which was then used to calculate the correlation between the estimated drowsiness curve and an ideal sigmoid drowsiness curve that was used as the criterion function for selecting the optimal feature subset. Apart from the feature selection approach, an intra-modality and inter-modality comparison was also conducted. While intra-modality refers to data/features within the same modality, inter-modality integrates data/features from different modalities. While the inter-modality approach resulted in positive results for a specific combination of features (RRI and EEGraw2), namely a 0.773 correlation coefficient, feature selection's results are superior - 0.785±0.011 correlation coefficient for the training test and 0.758 ± 0.159 on the testing set - for the modality fusion of RRI, breath, EEGraw1, and EEGraw2.

Table 3.2: Summary of Data Fusion Techniques

Family	**Technique**	**Summary**
Early Fusion	Early Fusion	Integration of features right after extraction, typically by concatenating their representations. Requires the training of a single model (Atrey et al., 2010).
Late Fusion	Late Fusion	Integrates decision values from each modality after independent decisions. Allows for the training of different models for each modality(Atrey et al., 2010).
Hybrid Fusion	Hybrid Fusion	Combines aspects of both early and late fusion approaches (Baltrusaitis et al., 2019).
Model-based	Probability-Based	Uses probability theory, including Bayesian theory-based, copula theory-based approaches, and Markov chain-based techniques (Gaonkar et al., 2021).
	AI-Based	Utilizes advancements in IoT and AI, notably deep learning (Gaonkar et al., 2021).
	Theory of Evidence/Belief-Based	Focuses on understanding uncertainty (Gaonkar et al., 2021).
Stage-Based	Stage-Based	Utilizes different data at various stages in data mining analysis, allowing fusion of multi-source heterogeneous data throughout the process (Zhang et al., 2018).
Semantic Meaning	Multi-View-Based	Integrates data from multiple perspectives of the same object, leveraging diverse datasets to find consensus and exploit distinctiveness (Zheng, 2015).
	Similarity-Based	Utilizes similarities across various metrics to augment sparse datasets with information from denser ones, enhancing overall correlation (Zheng, 2015).
	Probabilistic-Dependency-Based	Relies on probabilistic graphical models to represent conditional dependencies, improving understanding of probabilistic relationships in the data (Zheng, 2015).
	Transfer Learning-Based	Involves applying knowledge from one domain to a different but related domain, where feature spaces or data distributions might differ (Zheng, 2015).

3.1.2 Modelling Pipeline

The modelling pipeline is interconnected with the data pipeline, as the trend toward full automation of the entire ML pipeline from data acquisition to model deployment suggests. However, we aim to look at this pipeline from both a standalone context and within the context of its connection to the data pipeline (Mumuni and Mumuni, 2024).

An automated ML workflow has a number of steps that start with data extraction once the pipeline is triggered, followed by automated data preparation and validation, and then the automated final model training on the new data. After the retraining of the model, it is then evaluated and deployed (Kreuzberger et al., 2022).

Kreuzberger et al. (2022) names versioning as an essential principle of MLOps pipelines due to the necessity of maintaining the versions of data, models, and code. While it serves to facilitate reproducibility, it also ensures traceability. Continuous ML training and testing is also stated as a principal component - when new feature data is present, a retraining of the ML model must trigger, followed by an evaluation run. This typically helps prevent a decrease in the model's performance. Continuous monitoring is also stated as a principal component of MLOps modelling pipelines that has the objective of detecting potential errors and assessing the performance of the model. As stated before, experimentation with different models, parameters, or even training data (hence its relation to the data pipeline) is a key part of the modelling area in an MLOps framework, so there is a need to use experimentation tracking tools to save and compare the different results. These tools are a crucial part of the automation of ML training and testing due to their ability to meet the need to keep track of all experiments and maintain reproducibility while maximizing code reusability by storing data and features selected, as well as model parameters used and performance metrics. In addition, these tools are essential in MLOps pipelines as they allow the automation, management, and streamlining of the various stages of the ML lifecycle. In Table 3.3, a list of these tools is presented and their features, respectively: 1) Cloud Agnostic, 2) Open Source, 3) User Interface (UI), 4) CI/CD, 5) Library Agnostic, 6) Model Tracking, 7) Performance Monitoring, 8) Automation and 9) Quality of Supporting Documentation. Being cloud agnostic ensures consistent performance across different deployment platforms and meets this criterion by demonstrating deployability on Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP). Open source tools have their software code publicly available for use, modification, and distribution. The presence of a UI means that the tool presents a user interface or a dashboard. Being library agnostic means that all (top) ML frameworks and libraries are supported. Tools that allow model tracking allow intermediate ML model performance tracking and logging to maintain reproducibility and gain insight. Model-tracking tools allow you to track and log the intermediate performance of ML

models to maintain reproducibility and gain insight. Performance monitoring means that the predictive performance of the model is monitored to potentially trigger a new iteration in the ML process. When a tool enables automation, it means that the model management process can be executed automatically in production based on a schedule or in response to a trigger (Testi et al., 2022). The features of the MLOps tools presented have been extracted from their documentation by the Testi et al. (2022) and were updated in preparation for this dissertation. A new feature related to the quality of the documentation was also added.

Table 3.3: Comparison of MLOps Tools and Features [Adapted from Recupito et al. (2022)] C.A - Cloud Agnostic; O.S - Open Source; UI - User Interface; CI/CD - Continuous Integration/-Continuous Deployment; L.A - Library Agnostic; M.T - Model Tracking; P.M - Performance Monitoring; Auto. - Automation; Q.S.D - Quality of Supporting Documentation

Tool	**C.A.**	**O.S.**	**UI**	**CI/CD**	**L.A.**	**M.T.**	**P.M.**	**Auto.**	**Q.S.D.**
TensorFlow Extended (TFX)[5]	Yes	Yes	No	No	No	Yes	Yes	Yes	High
Apache Airflow[6]	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	High
MLflow[7]	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	High
Kubeflow[8]	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Med
Azure ML[9]	No	No	Yes	Yes	Yes	Yes	Yes	Yes	High
AWS SageMaker[10]	No	No	No	Yes	Yes	Yes	Yes	Yes	High
GitLab[11]	Yes	No	Yes	Yes	No	Yes	Yes	Yes	High
Jenkins[12]	Yes	Yes	Yes	Yes	No	No	No	Yes	High
Vertex AI[13]	No	No	Yes	Yes	Yes	Yes	Yes	Yes	High

After analyzing the tools shown in Table 3.3, we can conclude that some tools, such as TensorFlow Extended and Kubeflow, are more specialized for certain aspects of the ML lifecycle or certain technologies. Others, such as MLflow and Jenkins, offer more generalized solutions that can fit into a wider range of workflows and technology stacks. Specialization can be seen in tools designed for specific frameworks or environments, such as TensorFlow Extended (TFX), which focuses on TensorFlow. Generalization is seen in tools such as MLflow, which offer broad compatibility and feature sets across different ML libraries and workflows.

[5]More information on: https://www.tensorflow.org/tfx
[6]More information on: https://airflow.apache.org
[7]More information on: https://mlflow.org
[8]More information on: https://www.kubeflow.org
[9]More information on: https://azure.microsoft.com/en-gb/products/machine-learning/
[10]More information on: https://aws.amazon.com/sagemaker/
[11]More information on: https://about.gitlab.com
[12]More information on: https://www.jenkins.io
[13]More information on: https://cloud.google.com/vertex-ai?hl=en

3.1.3 Summary

This research revealed that data pipelines are very useful processes that convert raw data into formats ready for ML model development. These pipelines help establish consistency and efficiency, which would be harder to maintain otherwise, in processes like data extraction, preparation, and feature engineering. The degree of automation in these pipelines varies from manual to fully automated processes, with techniques such as automated machine learning (AutoML) employed to independently conduct the processes in the pipeline. Multiple variations of data fusion are used in the literature. Methods range from simpler ones, such as early, late, and hybrid fusion techniques, to more specialized ones, like model-based, AI-based, and stage-based fusions. Also connected to the data pipelines are the modelling pipelines, which aim to take an extra step into the complete automation of all processes, including model deployment. The principle components of this type of pipeline are versioning, continuous integration, and deployment. There are several tools that allow tracking experiments, managing the lifecycle of the models, and other processes inherent to the MLOps pipelines.

3.2 ML Modelling

This section is a literature review of the implementations and results of ML models in the domain of driver drowsiness and distraction detection/prediction. The main focus of this study will be on experiments conducted with data extracted from driving simulators, but experiments in naturalistic settings will also be considered to allow comparison between the two and to gain new interesting insights. There will also be a focus on state-of-the-art ML approaches to problems with similar data whilst focusing on a different domain. It will be split into research on detection models and prediction models.

3.2.1 Detection Models

The objective of the detection models is to accurately identify the state of the driver, such as distraction and drowsiness that is related to poor driving performance. Misra et al. (2023) utilized a variety of ML techniques to perform driver cognitive distraction classification via the analysis of data collected from a driving simulator. The experiment was divided into several scenarios, namely: a) Work-zone scenario, b) Curve scenario, c) Stop-controlled intersection scenario, d) Pedestrian crossing scenario, e) School zone scenario, and f) Parked vehicles scenario. The data of these experiments was extracted through a number of devices: an eye tracker, a wristband, and a simulator composed of a chassis consisting of a seat, pedals, and steering console. Within the data collected from the simulator arose the necessity to reduce the frequency of data points and to

condense the information. The wristband ensured the extraction of physiological data related to heart rate, electrodermal activity, temperature, and accelerometer. The eye tracker extracted the gaze point of the participant, fixation information, and saccades, which are fast movements of the eyes that change the point of fixation. A secondary set of features were created from the features initially extracted from the hardware. The chosen algorithms for this study were: a) Naive Bayes (NB), b) Support Vector Machines (SVM) with linear and Radial Basis Function (RBF) kernels, c) k-Nearest Neighbours (k-NN), d) Decision Tree (DT), e) Random Forests (RF). These models were trained using Python utilizing the scikit-learn package and were trained with data from each scenario separately and merged. Across all scenarios, the model that achieved the best results was RF (achieving 100% accuracy in the curved scenario when using a subset of data containing only the physiological data), while NB was the one with the poorest results. Results were superior when considering specialized scenarios instead of concatenating the data of each one, which led to the conclusion that driving context can lead to an improved driver distraction monitoring system. Misra et al. (2023) also concludes that eye-tracking and physiological data provide the best insights for distraction classification. A Bayesian Network (BN) classifier was implemented by Ahangari et al. (2020) also to detect driver distraction. This experiment, much like the one lead by Misra et al. (2023), was divided into scenarios: a) handsfree calling, b) hand-held calling, c) voice command, d) texting, e) clothing, and f) eating/drinking. In this experiment, data was collected only using a driving simulator that provided data such as speed, acceleration, throttle, lane changing, brake, collision, and offset from the lane center. As stated previously, the chosen classifier was a BN classifier implemented in a Weka 3.8 software package that was evaluated using the following metrics: a) prediction accuracy (ACC), b) sensitivity (true positive rate), c) precision, d) Matthews Correlation Coefficient (MCC), and e) Area under the ROC curve (AUC). In the independent test set, the prediction accuracy achieved was 67.8%, while the achieved sensitivity was 62.6% and 75.1% AUC. Using 10-fold cross-validation (CV), the prediction accuracy achieved was 70.8%. Nakano and Chakraborty (2022) aim to develop a technique able to detect driving distraction in real-time while using restricted computational resources. To do this, the collected data is either labelled as ‘normal’ or ‘distracted’ in order to have more data per class for better training of the classifier instead of differentiating between varying levels of cognitive load, which was a distinction made in their earlier research that showed higher detection accuracy for distractions involving higher cognitive loads. The data for this experiment was collected using a driving simulator that ran a mountain scenario and a city scenario, and each sample in the collected data represents several time series. Each sample may be represented by a multidimensional time series, each dimension can be considered as a feature which can be used for developing classification models for distraction detection. Sliding windows with a variable overlap were used to generate the training and testing samples for each

time series of the driving sample. The features from the simulator data are common among similar experiments - steering angle, steering torque, acceleration stroke, car speed, etc.. Nakano and Chakraborty (2022) propose two classification models using raw time series to preserve the dynamic characteristics of the time series. The first proposed model was a k-NN Classifier, and the second was a One Dimensional Conditional Neural Network (1D-CNN) Classifier - while conventional CNNs are quite popular in image classification problems, research indicates that using 1D-CNNs directly on raw time-series data results in high classification accuracy for a variety of problems (Nakano and Chakraborty, 2022). It must also be noted that the experiment was conducted with window sizes from 15 to 70 seconds in 5-second intervals and that ten-fold CV was used for training. Overall, the 1D-CNN led to better results, and despite the classifier, the accuracy increased with the increase of the overlap size. For the user with the best results, the 1D-CNN reached an accuracy of 0.847. Nakano and Chakraborty (2022) concludes that 1D-CNN models may be used for real-time detection of driving distraction as it performs quite well when applied with a smaller window size, which signifies the possibility of earlier detection - cognitive distraction can be successfully identified using a 1D-CNN with the past 15 seconds of driving data. Aksjonov et al. (2019) states that, despite all ML algorithms implemented in the domain of driving distraction, all of them use boolean binary classification (distracted/not distracted) and proposes a method that detects but also evaluates the driving distraction level of each individual driver, allowing the method to evaluate the influence of the distraction on safe driving performance. The apparatus used for this experiment was a simulator with a steering wheel and two pedals, and the used attributes were only performance-based - the data was obtained using the simulator. To detect distraction, a non-linear regression based on Euclidean distance is applied, while fuzzy logic (FL) is used to determine the percentage of driver distraction by fusing vehicle performance data. Predictors in this study are used to comprehend the typical driving patterns exhibited by the drivers when they're not distracted. This facilitates the establishment of a standard for expected driving behavior. The primary objective of detecting distraction is then achieved by comparing the ongoing driving behavior to these established norms. If a driver deviates significantly from these predicted norms, it suggests that they may be distracted. In essence, by predicting normal behavior, the study can more effectively and accurately detect when distractions occur. Regarding predictors for driver performance, the study focuses on the Euclidean Distance (ED) predictor and the Adaptive Neuro-Fuzzy Inference System (ANFIS), which was implemented to study the degree of accuracy of the ED prediction method. The ED predictor revealed slightly more accurate prediction capabilities (RMSE for the speed: 1.9992, RMSE for car the position: 0.14) and a faster training term when compared to ANFIS. While this approach brings a new level of innovation in the study of driving distraction, it is hard to compare its results to the remaining studies to its focus being on evaluating

the level of distraction instead of only detecting it, which is the focus in the great majority of the literature in this domain. Wang et al. (2022) proposes a driver distraction detection method based on vehicle dynamics using naturalistic driving data. The data extraction involved five vehicles equipped with data acquisition systems to collect driving data, including vehicle dynamics, environmental conditions, and driver behavior. Performance attributes included speed, longitudinal acceleration, lateral acceleration, lane offset, and steering wheel rate. The study focused on detecting phone-induced distractions through video recordings categorizing different types of phone use and comparing distracted driving periods with focused driving baselines. The ML model chosen for this purpose was a Long-Short Term Memory (LSTM) with some modifications - a bidirectional layer and attention mechanism were added to the model. The Bidirectional Long Short-Term Memory (Bi-LSTM) model achieved an accuracy of 91.2% in 5-fold CV, outperforming traditional ML methods such as recurrent neural networks (RNN), support vector machines, k-nearest neighbors, and adaptive boosting. This advanced Bi-LSTM model, enhanced with an attention mechanism, shows great promise for integration into driving assistant systems to alert drivers and mitigate distractions caused by mobile phone use.

McDonald et al. (2014) focuses on the detection of drowsiness-related lane departures through the use of ensemble learning using steering wheel angle data, namely an RF classifier. To conduct this experiment, a driving simulator and an eye tracker were used to extract data, and the model's performance was assessed relative to the PERCLOS algorithm. McDonald et al. (2014)'s main hypothesis is that an ensemble classifier will detect drowsiness more efficiently when compared with other ML models; however, when comparing the results of the RF with other fitted models (DT, ANN, k-NN, NB, SVM, and boosted tree), the RF shows the best results (0.70 AUC) but only outperforms significantly the NB approach. Nonetheless, the ROC curve was above all the other six curves. When compared to a PERCLOS-based algorithm, RF achieved similar results in detecting drowsiness-related lane departures. de Naurois et al. (2018) studies both detection and prediction of driving drowsiness using a driving simulator. The predictive aspect of the experiment will be analyzed in the next section. In this experiment, the level of drowsiness was evaluated by two raters on a scale that ranges from 0 (alert state) to 4 (extremely drowsy). Apart from data related to driving performance, data related to eyelid and head movements, as well as physiological data, was also collected. To detect driving drowsiness, a feedforward neural network (FNN) with one hidden layer optimized with the Levenberg-Marquardt algorithm was used. An innovative subject-specific performance assessment of the so-called Adaptive ANN was conducted in this study, where the model was trained with data from 19 participants, then adapted with the data of the 20th participant, followed by the testing of the model with the data of a 21st participant - in each iteration, the participants changed. However, experiments without this approach were also conducted. Regarding the re-

sults obtained for the detection of driving drowsiness, the model obtained the best results when all sources of information (simulator, physiological and behavioral, as well as personal information and driving time) were combined and after an adaptation of data-related parameters. The aforementioned innovative study revealed a significant improvement in the model's performance regarding the participant whose data was not initially in the ANN training set - the performance of the model improved by about 40%. In the study, after using the previously stated adaptation to a new participant, the average RMSE is 0.93. The necessity of being able to explain the decisions taken by the ML models is a major preoccupation of Hasan et al. (2024) when detecting driving drowsiness. In this study, EEG, EOG, and electrocardiogram (ECG) signals were extracted from the participants while these reported their level of drowsiness through the Karolinska Sleepiness Scale. Data related to the level of drowsiness was also extracted through psychomotor vigilance tasks, where participants must respond to a stimulus as quickly as possible via a keystroke. Hasan et al. (2024) implements two explainable ML methods, namely SHapley Additive exPlanation (SHAP) analysis and partial dependency analysis (PDA), and three supervised ML techniques were also implemented (k-NN, SVM, and RF). The interpretable methods show the most important physiological features for detecting drowsiness, demonstrating a distinct threshold in the decision-making boundary. The 'leave one participant out' CV technique revealed the best results with a sensitivity of 70.3%, specificity of 82.2%, and an accuracy of 80.1% using the RF classifier.

A brief study on literature that does not focus on the detection of drowsiness and/or distraction is also conducted here by mentioning a few studies that revolve around multivariate source event detection. Although the studies presented above may not be directly related to the theme of driver distraction and drowsiness, particularly Liu et al. (2015) and Fatichah et al. (2020), they do work with similar data to that in this dissertation, which is multi-source heterogeneous data. Therefore, these studies may provide valuable insights into handling and analyzing complex data structures. The conducted approaches may shed light on new techniques for data processing, methods for feature extraction, and particularly, modelling strategies that are applicable to your field. Liu et al. (2015) introduces a detection method for the identification of water contamination events using collected data from 8 different sensors that capture data such as temperature, pH, and conductivity. In this study, a main detection model was implemented, namely a Pearson correlation Euclidean distance-based method (PE), as well as two baseline detection methods used for benchmarking: a multivariate Euclidean distance (MED) method and linear predictions filter (LPF) method. Initially, the main method showed a 93% probability of detection, however using optimal parameter values the results increase to 97%. This model revealed better results when compared to the baselines - the MED method incorrectly classified 22% of the events, and the LPF wrongly labelled 38% of the events. Due to its lower false alarm rate

and higher detection potential, the PE method is considered by its authors as a more promising approach for use in early warning systems in the domain. Another interesting study that deals with multimodal data was conducted by Fatichah et al. (2020). The main aim of this study is to detect emergency incidents using multimodal data, such as text data and image data, extracted from social media platforms. The authors use multiple deep learning approaches such as: Convolutional Neural Network (CNN), LSTM, and C-LSTM - a type of LSTM that utilises CNN to extract a sequence of higher-level phrase representations (Zhou et al., 2015). The CNN architectures used are AlexNet, VGG16, VGG19, and SqueezeNet. Regarding text data, the C-LSTM revealed the best results - 99.09% accuracy with a lower average loss of 0.087. When focusing on the image data, SqueezeNet performed the best when data augmentation was not present (90.66% accuracy, 0.42 average loss), and when data augmentation was present, VGG16 had the best results (VGG16: 99.08% accuracy, 0.08 average loss). Due to the very good results, the authors conclude that these approaches should be applied in real applications more regularly. Rodrigues et al. (2021) proposes a strategy to enhance road safety by creating a non-intrusive wrist-worn device. The device should be able to detect and predict driver drowsiness, as well as continuously identify signs of chronic sleep deprivation or sleep disorders. Using a publicly available dataset for sleep analysis, this study utilizes heart rate variability (HRV) data derived from ECG signals in order to perform sleep staging. Sleep staging is the process of categorizing the periods of sleep into different stages: Wake, REM (Rapid Eye Movement), and various levels of NREM (Non-Rapid Eye Movement) sleep. Multiple traditional ML models are employed to obtain results that will serve as a starting point for analyzing future wearable data. The preliminary findings of this study establish a basis for future analysis of data gathered from wearable devices. The dataset used consists of recordings of 8 hours from 22 subjects aged between 18 and 63 years. The approach considered employs two types of validation: stratified 10-fold CV and Leave-One-Subject-Out CV (LOSO-CV). In each iteration, the training data was first normalized for each attribute, with a mean of zero and a standard deviation of one. Then, the best subset of features was selected using Pearson's correlation coefficient with a threshold of 0.9. The data was then oversampled using the SVM-Smote technique. The metrics used for evaluation in these studies were sensitivity and accuracy. Four classification algorithms were tested: SVM, linear discriminant analysis (LDA), k-NN with 15 neighbors, and RF with a maximum depth of 20. In the first validation (10-fold CV), it was observed that larger data segments improved accuracy for all algorithms except for LDA. RF was found to be the best algorithm for distinguishing sleep stages in a 4.5-minute window - sensibility to Wake, REM, and NREM of 81%, 71%, and 93%, respectively. However, the test that is dependent on the subject revealed that RF's sensitivity to REM decreased significantly due to variations in sleep stage distribution among individuals. This indicates that a larger dataset is required to handle this variability

effectively.

3.2.2 Prediction Models

While there is extensive research regarding the detection of both drowsiness and distraction, predictive models often focus on drowsiness due to the difficulty in predicting distraction caused by the inherently spontaneous nature of this event. The study conducted by de Naurois et al. (2018) was already mentioned in the last section regarding the detection of drowsiness. Here, there will be a focus on the results relative to the prediction of the time of occurrence of drowsiness. Just like its detection counterpart, the results greatly improved after the adaptation of data-related parameters. An 80% improvement in performance in the AdANN-validation dataset after varying the amount of data used. The results of the subject-specific performance of the ANN adapted to one driver and tested with another (technique explained in the last chapter) showed significantly lower RMSE and standard deviation values for participant A (i.e., the 20th participant). However, the values regarding Participant B (i.e., the 21st participant) are higher. Hallvig et al. (2014)'s study aimed to predict drowsiness in night shift workers during morning commutes using ML models. 16 participants drove a real vehicle on a closed track after their night shift. In this study, drowsiness was defined by performance degradation and electroencephalogram-characterized sleep episodes. Two logistic regression predictive models were built: one for performance degradation and the other focused on the EEG signals. When a model was applied to the data of an individual driver instead of the general data, both the AUC and accuracy improved. According to Hallvig et al. (2014), eyelid measures improved the results related to lane-crossing models. The overall performance of the lane-crossing model revealed an AUC of 0.82 and an accuracy of 88%, while the microsleep model's results were 0.82 AUC and 90% accuracy. Even though the predictions had low false alarms, there were a considerable number of failures in the detection of drowsiness events. A suggested solution to mitigate this is to use SVM. Chui et al. (2021) implements an LSTM neural network with the objective of predicting driver drowsiness and driver stress, where the data source chosen for this experiment is ECG signals. The data was extracted from two public data sources, which were extracted through experiments in naturalistic settings. A 5-fold CV was implemented in the study. The network implemented for this study was based on empirical mode decomposition (EMD), a technique used to decompose a signal into physically relevant components. The LSTM network's results were compared for different advance predictions, namely 1 to 5-second windows. The LSTM network results seem to worsen if the time window for prediction is larger - when the advance prediction was 1 second, it achieved an accuracy of 81.5%, whereas when the advance prediction was 5%, it decreased to 72.2%. The sensitivity and specificity results for each time window

were, respectively, 71.0–81.1%, 72.9–81.9%.

3.2.3 Summary

The summarized results, presented in Tables 3.4 and **??**, cover a wide range of problem types, data sources, ML techniques used, results obtained, and key findings of each study. The purpose of this summary is to highlight the significant progress that has been made in improving driver safety. It shows how various ML methods have been used to detect and predict driver drowsiness and distraction with remarkable effectiveness. Looking at Tables 3.4 and **??**, we can see that ensemble models such as RFs give particularly satisfactory results and that a wide range of neural networks, such as CNNs, FNNs, RNNs or LSTMs, are also widely used and provide good results. It can be concluded that RFs and Neural Networks, especially LSTMs, are the most widely used ML methods and generally present the best results. Cross Validation is the most commonly used validation method. It can also be concluded that there is no noticeable difference when comparing results from experiments carried out in simulated and naturalistic scenarios.

Table 3.4: Summary of Research on ML Models for Detection of Driver Drowsiness and Distraction

Paper	Problem Type	Data Type	Intrusion	ML Techniques	Validation	Results	Main Achievement	Open Source
Misra et al. (2023)	Distraction (Detection) - Simulated Setting	Simulator data, Eye-tracking, Physiological	No	NB, SVM, k-NN, DT, RF	10-fold CV	RF: 100% in curve scenario	Context-specific distraction detection improves performance	No
Ahangari et al. (2020)	Distraction (Detection) - Simulated Setting	Simulator data	No	BN	10-fold CV	Acc. 67.8%, Sensitivity 62.6%, AUC 75.1%	Validation of BN classifier for distraction detection	No
Nakano and Chakraborty (2022)	Distraction (Detection) - Simulated Setting	Simulator data	No	k-NN, 1D-CNN	10-fold CV	Acc. up to 84.7%	CNN effective for real-time distraction detection	No
Aksjonov et al. (2019)	Distraction (Detection) - Simulated Setting	Simulator performance data	No	Euclidean Distance, ANFIS	Train-Test Split	ED: RMSE for the speed: 1.9992, RMSE for car the position: 0.14	Evaluates level of distraction for safety assessment	No
Wang et al. (2022)	Distraction (Detection) - Naturalistic Setting	Vehicle data and driver behaviour	No	BiLSTM with attention mechanism	5-fold cross-validation	91.2% accuracy	Demonstrated the efficacy of an advanced Bi-LSTM model in detecting phone-induced driver distractions, with potential for driving assistant system integration.	No
McDonald et al. (2014)	Drowsiness (Detection) - Simulated Setting	Simulator data, Eye-tracking	No	RF	10-fold CV	AUC 0.70, Best compared to NB	RF effective in detecting drowsiness-related lane departures	No
de Naurois et al. (2018)	Drowsiness (Detection) - Simulated Setting	Simulator, Eyelid/Head movements, Physiological	No	Feedforward NN with Levenberg-Marquardt	5-fold CV	Improvement after data adaptation (RMSE <1)	Innovative subject-specific detection of drowsiness	No
Hasan et al. (2024)	Drowsiness (Detection) - Naturalistic Setting	EEG, EOG, ECG, Psychomotor vigilance tasks	No	SHAP, PDA, k-NN, SVM, RF	10-fold CV; Stratified 10-fold CV	Sensitivity 70.3%, Specificity 82.2%, Accuracy 80.1%	Explainable ML models for drowsiness detection	No

Table 3.5: Summary of Research on ML Models for Driver Distraction and Multi-source Problems

Paper	Problem Type	Data Type	Intrusion	ML Techniques	Validation	Results	Main Achievement	Open Source
Liu et al. (2015)	Water contamination detection	Sensor data (e.g., temperature, pH, conductivity)	-	Pearson correlation Euclidean distance-based method (PE), MED, LPF	-	93% detection, increasing to 97% with optimal parameters	More accurate detection with lower false alarm rate, promising for early warning systems	No
Fatichah et al. (2020)	Emergency incident detection	Social media data (text and images)	-	CNN, LSTM, C-LSTM, AlexNet, VGG16, VGG19, SqueezeNet	-	C-LSTM: 99.09% accuracy; SqueezeNet: 90.66% accuracy; VGG16: 99.08% accuracy with data augmentation	Effective use of deep learning for incident detection in social media data	No
Rodrigues et al. (2021)	Sleep Staging	HRV data from ECG signals	Yes	SVM, LDA, KNN, RF	Stratified 10-fold CV, LOSO-CV	RF best algorithm with sensibility to Wake, REM, and NREM of 81%, 71%, and 93%, respectively.	Demonstrated the potential of a wrist-worn device to detect/predict driver drowsiness.	Yes
de Naurois et al. (2018)	Drowsiness (Prediction) - Simulated Setting	Physio., Simulator, Behavioral	No	Adapt.ANN	5-fold CV	80% improvement after adaptation (RMSE < 6)	Improved drowsiness detection	No
Hallvig et al. (2014)	Drowsiness (Prediction) - Naturalistic Setting	EEG signals	Yes	Log. Regression	-	AUC 0.82, Acc. 88%-90%	Predicting drowsiness in workers	No
Chui et al. (2021)	Drowsiness (Prediction) - Naturalistic Setting	ECG signals	Yes	EMD-based LSTM	5-fold CV	Acc. 72.2%-81.5%	Predicting drowsiness and stress	No

Chapter 4

Methodology

This chapter aims to present a comprehensive look at the work developed. The requirements and specifications of the work, proposed solutions, and technological choices are presented. Potential challenges, highlighting how these factors can influence the development of the work, are also present.

4.1 Requirements and Specifications

This work is subject to a few requirements. It is essential that the developed pipeline is able to process the data collected from the BBAI experiments. Nonetheless, it should also be flexible enough to work with data in different formats or provide options to transform its formatting into a desired one. This is important because it is not certain that data resulting from future experiments will have the desired format. It should function with data presenting different formats and be dynamic enough to work with data from different devices than those used in the BBAI experiment or without some of the devices previously used. The pipeline must be integrated into an interface, which must be clear and simple to use so that everyone can easily use it. It is also desirable that the ML models perform well using the data extracted from the BBAI experiments.

4.2 Solution Proposal

Taking into account the state of the art and, on the other hand, the existing experience of the data, the methodology proposed for this project will follow the order shown in the flowchart in Fig. 4.1.

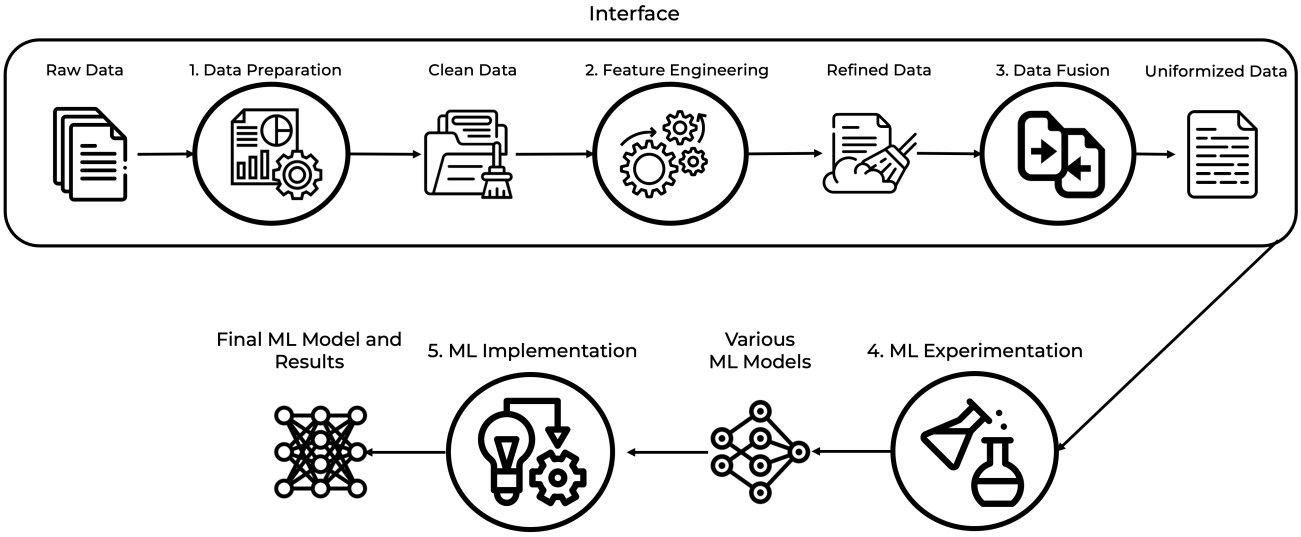

Figure 4.1: Proposed Methodology Fluxogram

The pipeline development process begins with data extraction and data preparation. This data preparation consists, for example, of imputing null values and changing the format of the files to data frames that enable fusion. The imputation of missing values is before the fusion as it was simpler to process the data before having it joined, and because having null values could be a problem when fusing it. The missing data, present in the data related to the interactions of the participant with the IVIS system and related to the eyetracker, in this project, cannot be characterized as Missing Completely at Random (MCAR) because the missingness is not independent of any data. Similarly, they do not fit the Missing at Random (MAR) definition, as the probability of missingness cannot be explained by other observed variables, indicating dependency on unobserved factors. Even though the missing data is related to unobserved data, it might not be related to itself, so we can't say with full certainty that it is Missing Not at Random (MNAR) either. There is a need to conduct a stage of feature engineering to create features that are useful for analysis and for the implementation of ML algorithms, which is represented in the second step. The third step in the development of this dissertation aims to solve one of the biggest challenges of this project. Due to the multi-modal nature of the data collected, the data fusion process is highly important. Data fusion's objective in the scope of this project is to create a uniform dataset that will facilitate the data analysis, in future experiments, and the implementation of ML algorithms. Due to the fact that the fusion is done after the data of each modality has been cleaned and preprocessed individually, late fusion is the technique that best resembles what has been developed. The interface comprises the steps up to this point. It is a command-line interface.

Once the data is ready to be used in predictive/detection ML models phase referred as "ML Experimentation" in Fig. 4.1 is conducted. This is an experimental phase where different algorithms are implemented to see which ones perform the best. Based on the conclusions drawn from the analysis of the relevant literature, as presented in section 3, the algorithms chosen are RF Classifier, Gradient Boosting Classifier, XGBoost and SVM. RF performs historically well in drowsiness detection, as seen in chapter 3. Nevertheless, SVMs are also often used on ML experiments in this domain, presenting good results. Gradient Boosting and XGBoost are under experimented within the literature, and, as RFs perform so well and these algorithms are part of the same "family" as the RF, it was thought that it would be interesting to test these. While ANNs, specifically LSTMs present good results in detecting and predicting driver behaviors, these algorithms demand a lot of data, so they are not developed in this work. Once the experimentation phase is over, the results of the models are compared, and the best ones are chosen for further development. Regarding the comparison, the metrics that will be used for this step are accuracy, AUC, and F1-Score, as most studies in this domain use them. This is useful for comparing the results of the experiments with state-of-the-art results.

4.3 Technological Choices

The processes described before must be automated, so tools like the ones described in Table 3.3 were considered. The criteria for choosing the tools was: a) being able to work locally/ not use cloud storage and b) being library agnostic. It is important that the tool does not use cloud storage due to privacy concerns. Library-agnostic software can be easily used alongside various other libraries without compatibility issues, which is also a concern. With both these criteria in mind, the most suitable tools are Apache Airflow and Jenkins. Both these tools can work with local data, are library agnostic, and are open source, making them free of charge. Nevertheless, it was established that managing each stage of the process through manual handling and development was more aligned with this work as it permitted immediate adjustments and streamlined troubleshooting, eliminating the need for the complexities of setting up and managing automation tools. Python is the chosen programming language.

4.4 Possible Challenges

It is important to note that this project carries risks, specifically related to data quality and quantity, as well as time and resource management. Data quality issues can be addressed through data cleansing, while data quantity issues can be mitigated by implementing data augmentation

techniques. Efficient resource management and algorithm optimization are necessary to avoid time management problems that might arise from long code execution times.

Chapter 5

Pipeline Development

This section aims to document and discuss the pipeline implementation that allows the transformation of raw data from various devices into a single refined file. It also aims to present the interface. This project was initiated by extracting and manipulating the various files, converting them into dataframes to facilitate their integration. As stated before, the files extracted from the devices related to the experience were large in number, some coming from the simulator itself, others from the eyetracker device, and others from the smartwatch. It's also worth mentioning that the names of the files extracted from the current version of the simulator (used in the BBAI experiments) and other devices currently do not have standardized names, which led to a necessity to allow various file patterns for the possibility of the extraction.

5.1 Data Extraction and Preparation

It was crucial to have an initial data preparation step in order to deal with data quality problems, such as missing values and outliers. Feature engineering followed this step. These files come in different formats (all of them text-based, such as .csv, .log, or .txt) and have different sampling rates. These are some of the challenges that need to be addressed before feature engineering and fusion can be performed. The following section presents the aforementioned challenges, which have been divided into sections according to the different data sources.

5.1.1 Simulator

The simulator data had two different types of files: one related to the data of the driving performance, such as velocity and acceleration, as presented in Table 2.1, and another related to the KSS (1-9 scale) of the participant in the drowsiness-related experiment, and to the events of the distraction-related experiment - the telnav file. The former file comes in the txt format, which

led to a straightforward extraction - apart from some general cleaning, namely removing some unwanted information present in the header and at the end of the file, the extraction of these files and subsequent transformation into dataframes did not necessitate any additional treatment. However, the files related to the driver interaction with the IVIS system (telnav files), corresponding to the self-reported KSS, required more specialized processing. The first challenge with this file was related to the timestamp. The timestamp has an error that makes it wrong - the first digits of the timestamp are not in the correct position. For example, 438195000289,00 should actually be 00,289438195000. This bug occurs in all the data related to both experiments. The second challenge is associated with missing KSS values.

When missing values follow high levels of KSS, the participant might have fallen asleep; missing values at the beginning of the experiment may have been because the participant didn't understand that he/she had to report his/her sleep level, and the missing data at the end of the experiment may have occurred because the participant had already left the experiment but the simulator still prompted him/her to report his/her sleep level.

The imputation of missing values in the 'KSS' column is described below:

1. Missing values in the first rows of the dataframe:
    - One/two rows have missing values - filled with the first available non-missing value
    - Two or more rows have missing values - removed
2. Missing values in the middle of the dataframe:
    - Imputed with 10, if the last non-null value is > 5
    - Imputed with 0, if the last non-null value is < 5
3. Missing values in the final rows of the dataframe:
    - Removed

It is also important to mention the reason why these missing values are imputed with 10 or 0 since these values are not on the KSS scale. The reason for this is to distinguish genuine KSS values from synthetic ones. A feature called 'Fiablity' is also created for this same purpose - to record which of these values are synthetic and which ones are real. This feature 'Fiability' takes the value '1' if the value of the KSS is genuine and '0' otherwise. This approach ensures that all missing values in the 'KSS' column are systematically managed based on their context within the data.

Although not all of these steps were essential at this stage of the work, the smaller size of the dataframes, when compared to the sizes after the fusion of the files, allowed for easier analysis

and debugging. The consistency in the data quality across all dataframes to be fused was also a concern. It made sense to have the dataframes prepared before they were combined to prevent the propagation of errors and inconsistencies.

The files regarding the distraction-related experiment did not receive such treatment because they had no missing values. In this experiment, when the user does not interact with the IVIS, it is also considered an event and is registered as such, hence the non-existence of missing values.

5.1.2 Eyetracker

The focus regarding the data extraction and preparation of the eyetracker data revolves around the data preparation. The data extraction was simple as it was saved in a simple format - csv. Data preparation, which involves processes such as missing value imputation, deserves a more significant focus. Among all the files, the gaze-related ones were the emphasis. This is because they are the original files from which the blink and fixation files are derived.

5.1.2.1 Missing Value Imputation

The gaze files (regarding the left and right eyes and their vergence) contain missing values caused when the eye was not visible. Figures 5.1, 5.2, 5.3 show the percentage of missing values for each participant in, respectively, the 'gaze right,' 'gaze left' and 'gaze vergence' files.

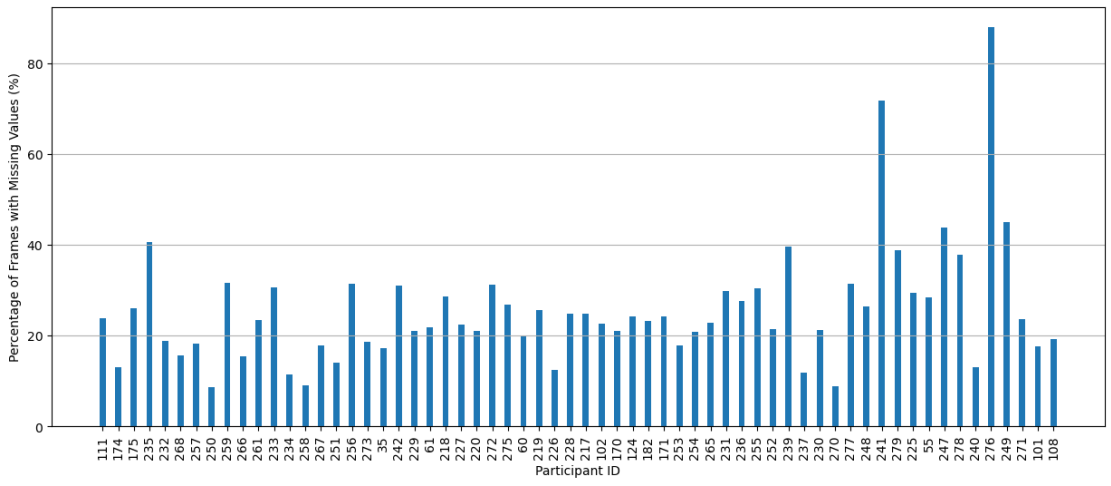

Figure 5.1: Percentage of frames containing MV for each participant in the file 'Gaze Right'

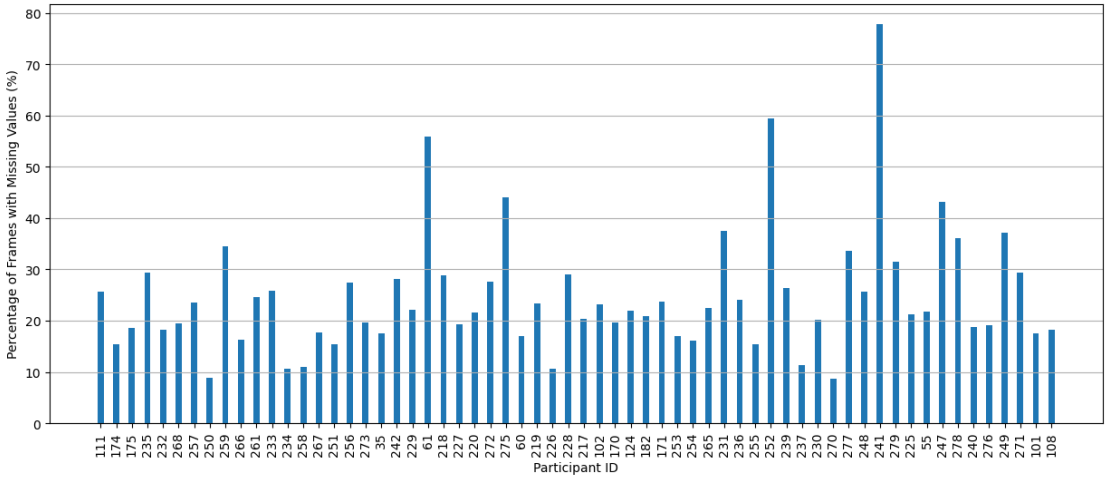

Figure 5.2: Percentage of frames containing MV for each participant in the file 'Gaze Left'

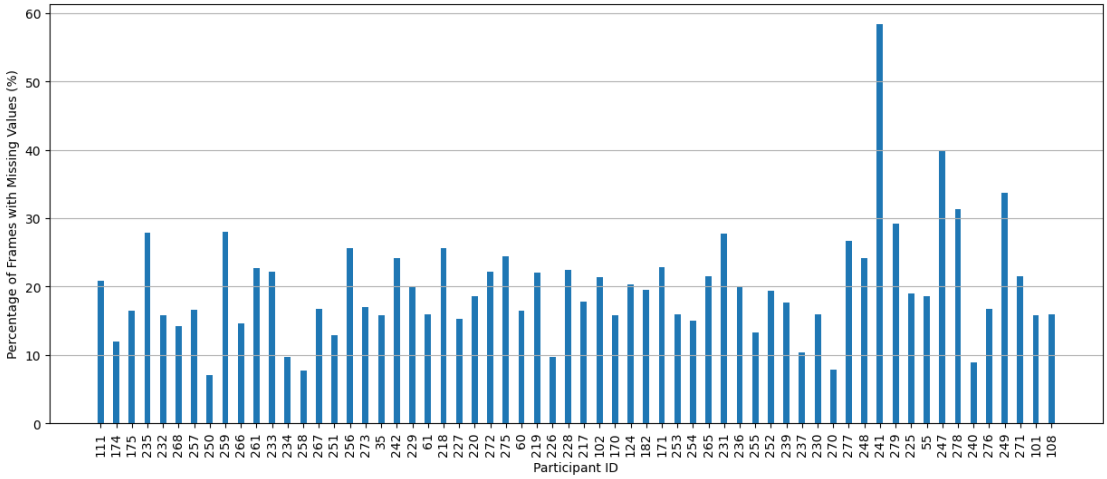

Figure 5.3: Percentage of frames containing MV for each participant in the file 'Gaze Vergence'

The fact that the eyetracker could not detect the eye, resulting in missing values, could be due to a myriad of reasons that we could not assess with certainty. Some possible explanations, other than that the participant might have been sleeping or blinking, are:

- Eye movement when looking at rear-view mirrors;
- Frames of the glasses obstructing the camera angle to the eye;
- Participant moving;

- Technological limitations of the eyetracker device.

The frames in our data have a temporality of around 150 microseconds. This means that three frames would amount to 450 microseconds. Normally, blinks have an approximate duration of 150 to 400 ms. So, when three or fewer consecutive frames had missing values, we considered these to be blinks. The missing values caused by these blinks were imputed by replacement. The missing pupil diameter values related to the pupil diameter when a blink had happened were not imputed with zeros because it was determined that zeros would represent extended periods of a non-visible pupil. Including zeros or interpolating for both blinks and for extended periods of time could be prejudicial and confusing because it could make analysis and subsequent drowsiness detection harder. The replacement was done by linearly interpolating the coordinates' and pupil diameter values. If three or more frames were missing, the data related to these frames was imputed with zeros. It was also decided to interpolate the values of the coordinates when the interval was shorter; however, for cases where it extended beyond three or more frames, this method could be misleading due to the unpredictable nature of visual perception, so they were imputed with zeros. As these instances could also represent micro-sleeps, it wouldn't make sense to replace the pupil coordinate (and the pupil diameter) values with anything other than 0.

In the context of handling missing data for intervals shorter than four frames, interpolation was used to replace the missing values, as aforementioned. For each gap, the index immediately preceding the gap and the index immediately following the gap was determined, ensuring that these indices remain within the bounds (start and finish) of the dataframe. The values at these indices were retrieved, and their mean was calculated. This calculated mean then filled the missing values within the gap, interpolating the data based on the adjacent known values. This interpolation approach secures the continuity of pupil diameter data, as well as the vehicle coordinates, ensuring a more accurate reconstruction of small missing segments.

5.1.2.2 Outlier Treatment

It's important to note that there are certain anomalies in the data, particularly in relation to pupil diameter. These anomalies are often related to the subject's movements, particularly as they move closer to the eyetracker, which can result in an increased pupil diameter being recorded. The distribution of these outliers can be seen in Figures 5.4 and 5.5. The data related to the pupil diameter received treatment to deal with these outliers - a threshold-based approach based on the interquartile range (IQR) was chosen. The IQR approach is effective in the detection of extreme values, and due to the physiological nature of the data, where outliers can represent errors or natural anomalies, using the IQR method provides a balanced approach to ensure the integrity of

the analysis. The k value used to define the upper threshold was 3 since we wanted to preserve as much useful data as possible (using lower values of k removed several values that were not considered outliers).

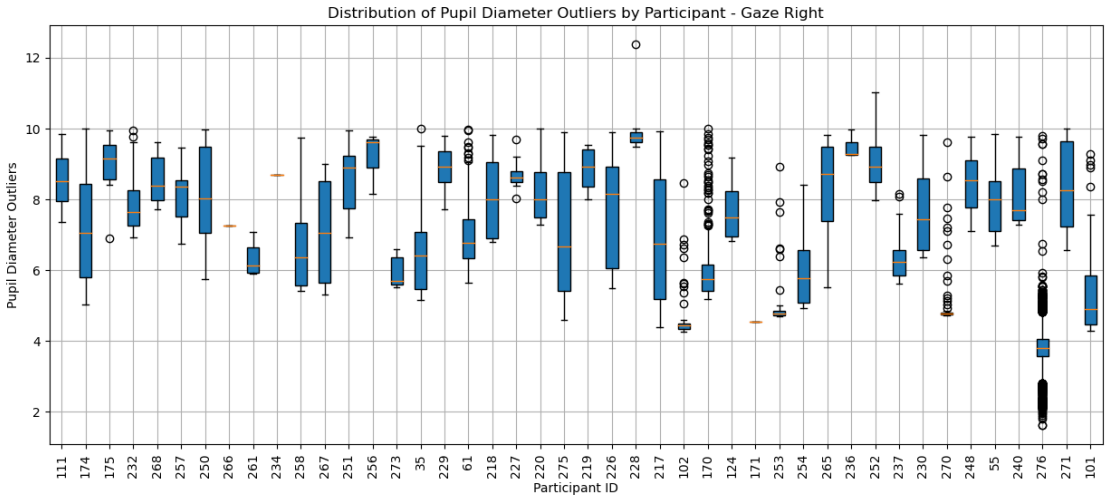

Figure 5.4: Outlier distribution of the Pupil Diameter for file 'Gaze Right'

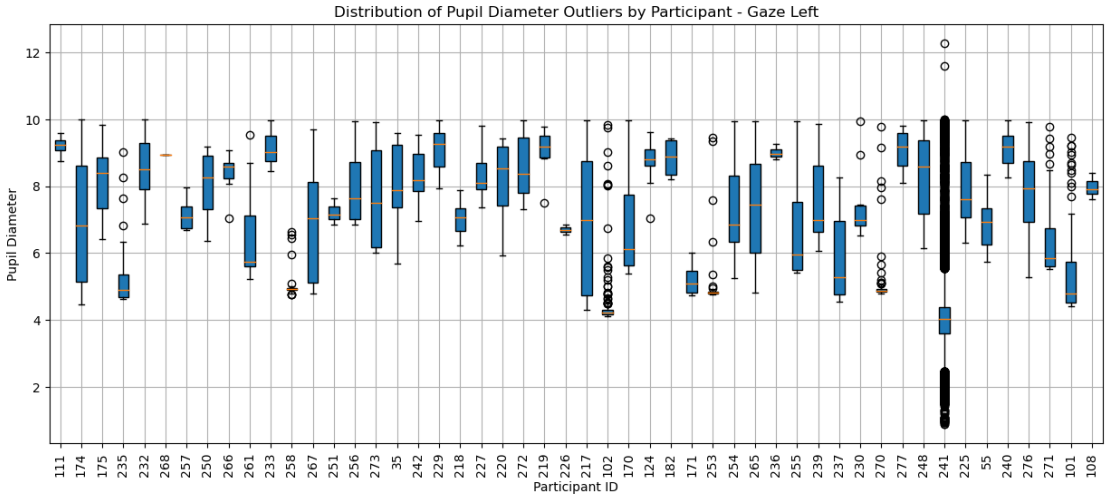

Figure 5.5: Outlier distribution of the Pupil Diameter for file 'Gaze Left'

The chosen approach regarding the outliers was to replace their values with a synthetic value of 999. The approach of replacing instead of simply removing these samples was chosen to maintain temporality. The approach of replacing instead of removing was chosen to maintain temporality. Removing outliers could potentially skew the temporal distribution of the data,

leading to gaps or irregularities in the time-series analysis. By replacing outliers with a standardized value like 999, we ensure that each data point retains its temporal context while indicating a clear deviation from the typical range of values.

5.1.2.3 PERCLOS Calculation

Once the outliers were handled, a normalized feature representing the percentage of the visible pupil was calculated- the formula used is:

$$\left(\frac{PupilDiameter - min\_value}{max\_value - min\_value}\right) \times 100 \tag{5.1}$$

Here, min_value and max_value are the minimum and maximum 'Pupil Diameter' values from the non-outlier data. This formula converts the pupil diameter values to a percentage representing the degree of eyelid closure. A new binary feature called 'eye open' was created after calculating these values. If the value of a certain frame was above 25% (Abe, 2023), then it was considered that the participant had his/her eye open. Afterward, with the information on which frames the eyes were open or not, PERCLOS was calculated. PERCLOS, representing the percentage of eyelid closure over the pupil over time, is a measure normally utilized to determine the degree of drowsiness, quantifying the proportion of time that a person's eyelids are partially or fully closed throughout a specified period. The PERCLOS values were calculated at 60-second intervals using a sliding window. Using a sliding window approach captures temporal or sequential dependencies in data and, by incorporating temporal dependency, can smooth the distribution of values over time. The 60-second interval was chosen because it aligns with the standard duration used for measuring heartbeats (bpm - beats per minute), allowing for parallelism with data captured by the smartwatch.

5.1.3 Smartwatch

Regarding the files related to the smartwatch, the extraction process was straightforward, as the data itself was already well prepared for extraction. It was extracted as a .csv file, allowing simple extraction using the pandas library. The smartwatch used was set on 'Sport Mode,' which is aimed at being used in activities like running, cycling, or other outdoor sports. The data from these files was reduced to the 'Heart Rate' data since it was the only information necessary for this study. Examples of other features captured by the smartwatch are cadence, power output (watts), headwind, and slope.

5.2 Feature Engineering

Before the data was merged, some feature engineering was conducted. This process permitted the generation and extraction of novel features from existing ones, thereby facilitating more effective data analysis and potentially offering utility for future ML model implementations. The newly created features are as follows:

- **Lane Change and Line Steppings** - Two features related to the position of the vehicle were created based on the feature 'eixo offset (m),' which is described in greater detail in Fig. 2.2. The information provided by this feature allows for the determination of the instances in which the simulated vehicle contacts the center line and when it shifts into the adjacent lane or into the curb. In order for the vehicle to change lanes, i.e., the entirety of the vehicle in the adjacent lane, the offset value must fall within the range of 2.1 to 4.25 meters. Conversely, in order to assess whether the vehicle has touched the line that separates lanes or the curb line, the offset must be between 4.25 and 5.85 meters (left/right axis of the wheels touching the middle line of the road) or equal to 8 meters (right axis of the wheels touching the left line). These calculations are conducted at discrete, non-overlapping 60-second intervals, during which the cumulative count of lane changes and line steppings is calculated. The function initially sets a cumulative counter and an interval start time using the first timestamp in the dataset. The lane changes, or line stepping, are assessed and increase the counter's value. As it iterates through the data, it checks if the current timestamp has passed the 60-second mark from the start interval, resetting the counter and updating the start time if necessary;

- **Cumulative Interactions with the Vehicle** - A new set of features is added by a function that calculates the cumulative interactions of the driver with the vehicle within 60-second intervals. This includes counting changes in several variables that indicate driver interaction, namely braking, acceleration, gear changes, and steering adjustments - the threshold for steering wheel changes to be considered is set at a difference of more than 6 degrees (Soares et al., 2021). The function initially sets a cumulative counter that assesses changes in braking, acceleration, gear, and steering in the same way used in the calculation of lane changes and line steppings. This feature is critical for analyzing driver responsiveness and interaction patterns, which can be critical in studies focused on driving habits, safety analysis, and automated driving systems by enhancing the dataset with insights into driver behavior over discrete periods of time;

- **Steering Wheel Features** - Steering wheel movement is often associated with drowsiness. The patterns of steering wheel movement when a participant falls asleep or is about to are

typically different than when awake and alert. Thus, a group of features that enrich a driving data set with calculations related to steering dynamics were also created. This processing enables a comprehensive analysis of steering behavior. It includes a feature for steering angle in radians, the difference in steering angle between successive data points, the time interval between these points, and angular velocity, which is the rate of change of steering angle over time. The function processes steering angle measurements, converting them from degrees to radians, then calculating the steering angle differences and time intervals to calculate the angular velocity;

- **Statistical Features (Standard Deviation)** - It is also important to mention that a feature was created relating to the standard deviation of velocity, offset, acceleration, and angular velocity. This feature was calculated using moving averages. Calculating moving averages smooths the data, reducing noise and highlighting longer-term trends or cycles. This is critical for data analysis, as it helps to better understand underlying patterns and behavior over time that may be obscured by short-term fluctuations. When implementing ML algorithms, this can improve model accuracy and robustness by providing a more generalized data representation. Models trained on such features are often better at predicting trends and making decisions based on more stable, less noisy input data, which is particularly valuable in dynamic environments such as driving, where conditions change rapidly;

- **Label - Sleepy** - One of the created features was the feature 'sleepy.' This feature is binary and takes the value 1 if the KSS of the user is 6 or above, and 0 otherwise (Rudin-Brown and Filtness, 2023). This feature was used as the label for this project's machine learning algorithms.

This process was performed prior to merging the data to increase efficiency and optimize resource usage.

5.3 Data Fusion

The data fusion technique employed was inspired by Late Fusion, primarily because it was implemented after each data modality had been prepared, cleaned, and feature-engineered. This approach was chosen to facilitate direct control over the integration of features from different modalities, ensuring that each source's unique information was accurately combined. By integrating features extracted from each modality after their respective data extraction, preparation,

and feature engineering stages but before applying ML algorithms, we maintained precise control over the final dataset. This method allowed us to merge features in a hands-on manner, enabling swift identification and resolution of any inconsistencies or heterogeneities in the data.

The potential use of Apache Airflow and Jenkins for developing the data pipeline processes was discussed. This was due to their ability to work locally, library-agnostic nature, and open-source licensing. Despite the advantages of these tools, the decision was made against employing them for several reasons. Firstly, the complexity of setting up and maintaining these tools for the project's scope did not justify the benefits. The learning curve and the time required to integrate these systems could divert focus from other critical aspects of the research. Furthermore, the scale and specific requirements of the project made manual management a more practical and efficient approach. By maintaining direct control over each step of the pipeline processes through manual handling and development, it was possible to implement immediate adjustments and simplify troubleshooting without the overhead of configuring and managing an automation tool. Furthermore, this approach provided invaluable insights into the operational aspects of the pipeline, enhancing understanding and offering direct empirical experience, which was crucial for the exploratory nature of the dissertation.

5.3.1 Data Resampling

After the initial extraction, data preparation, and feature engineering, the data fusion/merging of the different files occurred. The pipeline allows for the extraction and unification of all possible combinations of devices with the data from the simulator (simulator data by itself, simulator data with smartwatch, simulator data with eyetracker (gaze), and simulator data with both), and even the addition of different types of data for devices that might be useful in experiments that might take place in the future. The joining was done in such a way as to maintain the temporality of the data extracted by the simulator since this was considered the main modality of the experiments. If the sampling frequency of the devices to be added to the simulator data is greater than the sampling frequency of the simulator, then downsampling occurs. This downsampling is done by merging the files on the closest timestamp of the simulator dataframe and discarding the remaining samples. Downsampling was only necessary when merging the simulator files (driver behavior) with the eyetracker gaze files, as the latter has a higher sampling frequency. Otherwise, upsampling is used. This method warrants that samples with the closest timestamps are aligned. Since the file being added has a lower sampling frequency, its samples are duplicated to match the number of samples in the higher-frequency file. Linear interpolation was considered unnecessary because the intervals between samples are so short that it wouldn't make much difference. Additionally, performing linear interpolation on such a large number of samples would

be computationally expensive. Apart from the eyetracker files, every device used this technique when joined with simulator data.

5.3.2 Synchronization

When joining the dataframes previously extracted from the BBAI experiments, it was also made sure that the data was synchronized - due to technical constraints, the devices were not synchronized when the experiments started, which led to the data start time of the different devices not being the same. Through a previous file that documented the start time (hours, minutes, and seconds) of each device for each participant, this synchronization was done by adding the time difference between the start time of each device in comparison to the start time of the simulator, which was the reference. When merging the telnav files that contain the KSS values or the events, depending on the experiment, it was also important to make sure that the value of the KSS or event is upsampled backward, using backward filling, not forward. This ensures that each low-frequency value is propagated to cover all preceding high-frequency timestamps until the next low-frequency timestamp, maintaining data integrity and continuity for analysis. This is specifically necessary for the drowsiness-related experiments because the participant reports the drowsiness level that he/she had been experiencing for the last 5 minutes. It is important to mention that the merging so far referenced is not done by the index number of the columns of the different dataframes, but instead by their denomination.

5.3.3 Metadata

With the help of external data collected at the time of the previous experiment, it was also possible to add the metadata of each participant (such as gender, age, health problems, medication, etc.) to the main dataframe. This fusion is summarised in the addition of this data as features in the dataframes of each participant, without the need for the downsampling/upsampling techniques described above. Like the data from the other devices, apart from the simulator data, this is not strictly necessary for the pipeline to work.

5.4 Pipeline and Interface Functionalities

Once the principal types of files had been merged, the interface that implemented the pipeline, allowing for the extraction, preparation, and merging of the files, was developed. The interface containing the pipeline comprises three principal options, depicted in Fig. 5.6:

- Extracting and fusing files from the BBAI experiments;

- Mapping features of datasets from other experiments;
- Extracting and fusing files from other experiments.

The first option is the implementation of the previously developed code that joins the diverse data from the simulator experiments while also allowing other types of data to be joined. The alternative files, however, must be in a .csv format with a ';' separator and must have a column named 'tempo (s)' in order to properly join the data. This option is useful in case there are added devices in future experiments while still using the same driving simulator or any other device used in the BBAI experiment. The following option allows the user to map features to their necessary names so that the data fusion can be successful. This option allows the mapping to be done via a preconfigured .json file or to be done manually, where it prompts the user to map each feature. If there is any feature the user does not want to map, i.e., there is not an equivalent feature in its dataset, he/she may skip it. It also allows the user to add additional features that are not necessary. The last mentioned option allows the user to select the datasets that were once mapped in order to proceed with the fusion of the data. The main difference between this option and the first is that the first allows the user to input a folder with all the data from all the devices in the exact format and with the exact path files used for the BBAI experiment while the last version prompts the user to select the data from each device (file or folder) individually. The program runs on the command line and uses a series of commands and menus to allow the user to select what they want as easily and simply as possible, and when the user is prompted to select or save files/folders, it opens a file explorer window for the user to do so.

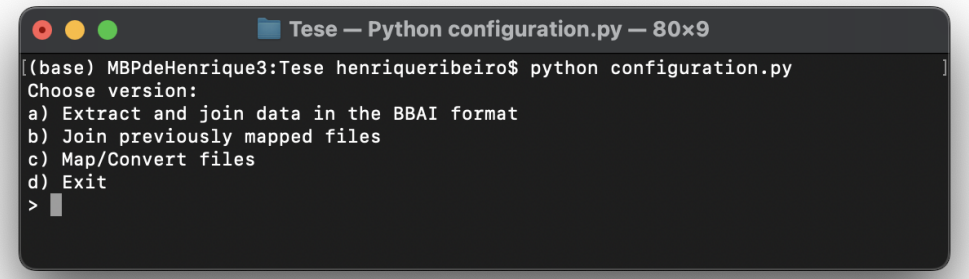

Figure 5.6: Inital Menu of the Program

A guide written to aid future researchers that might use this tool is presented on A.

5.5 Interface Validation

In order to validate the performance of the interface containing the automated pipeline, the program was tested by three investigators of the Universidade do Porto, who were familiar with the driving simulator and participated in previous research using it. The guide for this validation is presented in appendix B. The testing consisted of the realization of five tasks. These tasks mainly consist of using all the main functionalities of the interface: fusing data from the BBAI experiments, fusing data from the BBAI experiments with additional files from other devices, mapping features from synthetic data to the necessary denominations (both manually and through preconfigured files), fusing data strictly from other devices/with other formats. During these five tasks, the number of errors made by the participants and the number of times they needed help was recorded. The duration was also recorded. The System Usability Scale (SUS) is a survey that aims to assess the usability of various products, such as websites, TV applications, and others (Bangor et al., 2009). The evaluation was measured by a scale system based on the SUS, where the participants answered a few questions on a scale of 1 to 5. The responses are then converted to a range of 0 to 100, indicating the level of usability. The questions of the SUS-adapted questionnaire were the following:

1. I think the system is useful in the context of the simulator.
2. I found the system unnecessarily complex.
3. I found the system easy to use.
4. I think I would need the support of a technical person to be able to use this system.
5. I found the various functions of this system to be well integrated.
6. I think there is too much inconsistency in this system.
7. I imagine that most people would learn to use this system very quickly.
8. I needed to learn a lot of things before I started using this system.

The questionnaire results yielded scores of 82.5%, 80%, and 85% for each participant, respectively, with an average score of 82.5%. Since only 8 questions were used instead of the usual 10 in the SUS, adjustments were made to adapt the SUS scoring to this modified format.

Table 5.1: Tasks' Results

Participant	**Duration (m)**	**Errors**	**Help**
Participant 1	12:42	1	4
Participant 2	8:54	1	1
Participant 3	13:39	0	1
Author	5:37	0	-

The participants showed ease when using the interface, as can be seen by the results shown in Table 5.1. When analyzing the results, it can be seen that the participants did not require much help throughout the experiment and committed a minimal number of errors. It was expected that the participants would take more time doing the tasks than the author; however, the difference was not very discrepant, considering that the participants were not accustomed to the interface and spent some time understanding the tasks at hand.

Some feedback was also collected at the end of this experiment, where the participants made a few suggestions, namely:

- Some menus should have a better description;
- Change the sequences of some of the options in the menus to allow easier reading;
- Change the sequences of the prompts to the user; for example, when mapping features, always prompt the user for the necessary configurations before asking for the directory to save the file;
- In the option that leads to the fusion of data from the BBAI format allow the user to select the columns he/she wants to extract via a configuration file instead of always extracting all the features;
- Rather than having the user manually copy and paste directories, always open a file explorer window to facilitate this process.

The aforementioned suggestions stated by the participant at the end of the experiment were posteriorly integrated. Participants emphasized that the tool has the potential to speed up processes relating to data processing in future experiments, which is in line with the objectives of developing this tool. The validation proved to be successful, with the participants being satisfied with its performance, as proved by the aforementioned results.

Chapter 6

Machine Learning Modelling

This chapter discusses the various steps and approaches that were discussed and implemented. Once these have been made clear, a second section focusing on the results of the models and methods that received a greater focus is presented.

6.1 ML Experimentation

Before delving into ML models, approaches regarding their objective were reflected upon. It is crucial to note that the algorithms and methodologies employed were designed specifically to detect drowsiness. Three different approaches were considered:

- **Personalized Approach:** Creating a personalized individual model for each participant could provide good results. This individual approach would require every individual to set up his own devices and self-report his level of drowsiness so that the model could be trained and adapted to himself. While theoretically feasible, this method diverges from the intended purpose as it necessitates a user-level training phase, which is undesirable. Although the individual approach was initially considered in the early stages of the project, it was subsequently abandoned;
- **Clustering Approach:** The initial idea/approach to be taken was to create several predictive models based on user profiles, which would be created through clustering; however, due to the limited amount of data available this proved to be too challenging - it led to unreliable or overly simplistic profiles. This approach used KMeans clustering, and the number of clusters was determined using the elbow method. KMeans was the chosen clustering approach due to its computationally efficiency and interpretable results, shown in Fig. 6.1. The chosen number of clusters (k) was 4 because it is on this value that there

is a significant decrease of the Within-Cluster Sum of Squares (WCSS) - a measure that determines the variance of each cluster. This can be seen in Fig. 6.2;

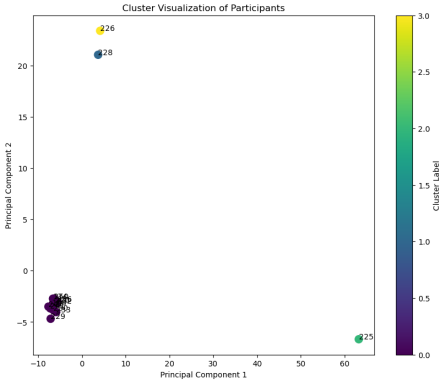

Figure 6.1: Clustering using KMeans

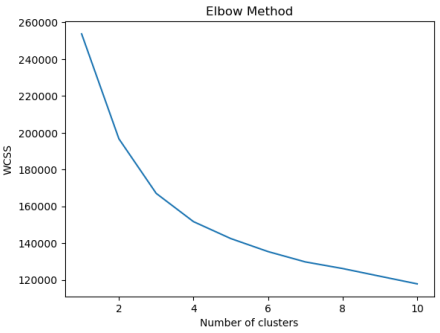

Figure 6.2: Elbow Curve

- **Generalized Approach:** In light of the issues identified in the aforementioned approaches, the selected methodology involved developing a model capable of generalizing drowsiness predictions for all participants. This way, all the participants' data is used to develop a single model. While this approach may not be the most optimal, it was the most logical given that data was insufficient to produce the clustering approach and because the personalized approach goes against the objective of this work. The process of implementation and specifics of this approach are detailed below.

6.1.1 Data Selection

To begin the process of experimenting with ML models in order to predict sleep in the context of driving, it was necessary to select the relevant data. Since the data has been preprocessed by removing outliers and null values, it is now well-prepared for use in ML models. However, it was essential to include participants with relevant data selectively. Those who had not reported being drowsy during the experiment were not considered because training ML models with datasets containing only one label would be ineffective. For model development, data from 30 of the 66 participants were selected. These participants' data contained both labels, 0 and 1.

6.1.2 Feature Selection

The dataset underwent a phase of feature selection, during which the variables that were highly correlated with the label were identified and removed to prevent data leakage. Although various combinations of features have been tried in this experimentation phase, the ones that have always

been excluded were: ‘X’, ‘Y’, ‘roaddist (m)’, ‘timestamp’ and ‘totaldistance (m)’. Due to the experiment’s design, it was expected that these variables would have a high correlation with the label. This is because the design artificially amplifies the effect of the features. Longer driving distances and longer driving times were associated with increased levels of drowsiness. This is presented in Fig. 6.3. The ’KSS’ variable was also removed because the label was extracted from this feature. These variables were excluded due to their high correlation with participants’ level of drowsiness.

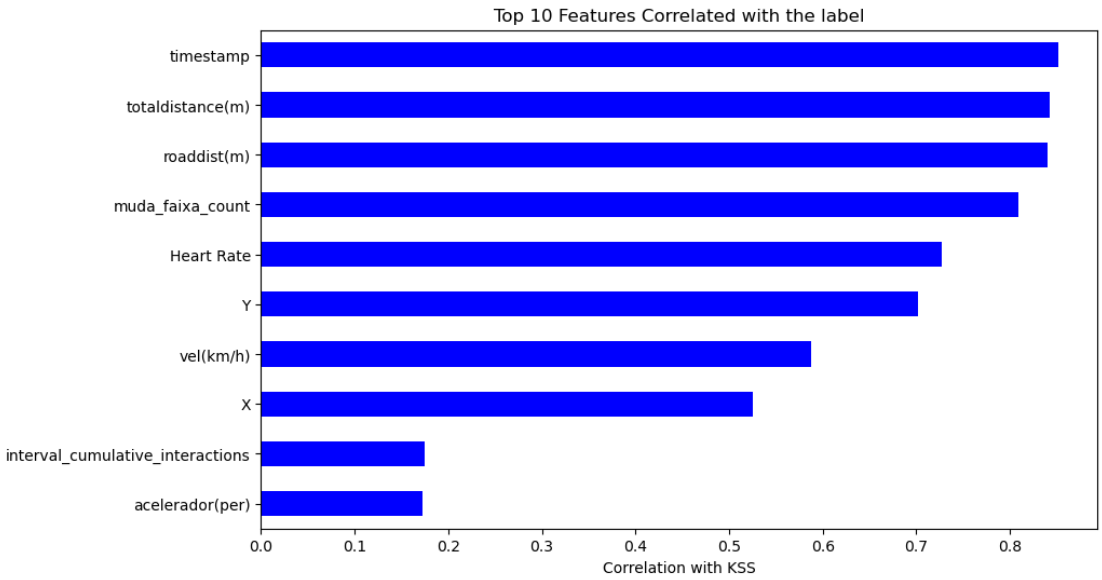

Figure 6.3: Top 10 features correlated with the Label

6.1.3 Data Interval Selection

A significant challenge encountered was preserving the temporal sequence of the data. This task was complicated by the observation that participants generally exhibited increasing levels of sleepiness over time. Consequently, time was found to be strongly correlated with sleep levels. As a result, the data labeling exhibited a pattern where it commenced with 0 (label for non-drowsiness) at the start of the experiment and transitioned to 1 (label for drowsiness) towards the end, without any crossover between these states. To try to get around this problem, the approach chosen was to transform the data into 60-second blocks, i.e., to transform all the data captured during 60 seconds into a record that captures statistical information (mean, mode, standard deviation, and median) about the data during this interval. The interval duration was chosen to be 60 seconds to match the conventional measurement period for heart rate, which

is typically expressed in beats per minute (bpm). This standard interval ensures consistency with conventional heart rate metrics. In addition, maintaining a 60-second interval maintains integrity and consistency with other previously implemented systems that also use a 60-second interval. Note that a sliding window was used. This way, it was possible to create more samples. The sliding window permitted an overlap of 25 seconds between samples. A higher value could result in higher redundancy and similarity between samples, leading to overfitting. Through this approach, it was also possible to shuffle the data while maintaining the temporality associated with the intervals and dealing with the changes in the drowsiness levels.

6.1.4 Data Splitting

The dataset was divided using adaptive k-fold cross-validation to ensure robust model evaluation, which is particularly crucial for our dataset's imbalanced class distribution. This technique involves partitioning the data into multiple subsets, each containing an approximately equal proportion of the classes present in the entire dataset, thereby preventing training bias towards the majority class. Each subset is used sequentially as a validation set, while the remaining subsets are used for training, allowing for a comprehensive assessment of the model's performance across different data segments. To further enhance the evaluation's robustness, each class is represented by a fixed number of samples, which are systematically incremented in the training set across sequential folds while maintaining consistent representation in the validation set. Random shuffling of class indices before dataset segregation introduces necessary randomness, minimizing order-based biases.

6.1.5 ML Models and Metrics

Various machine learning algorithms were explored when developing a predictive model for driver sleepiness, and their effectiveness was compared with the specific dataset. This experimentation phase included algorithms such as ensemble and boosting methods, which were evaluated using various metrics. The models with the best results were selected, and, at a later stage, further refinements were focused on optimizing these models (presented in section 6.2), setting the stage for advanced developments in enhancing road safety. The algorithms implemented were a Random Forest Classifier, Gradient Boosting Classifier, XGBoost, and Support Vector Machines. Random Forest Classifiers and Support Vector Machines are often used when detecting or predicting driver behaviours; however, XGBoost and Gradient Boosting are not as much. Observing how these algorithms would behave in this domain was thought to be interesting. The performance metrics such as accuracy, F1 score, and AUC are utilized to assess the models's efficacy.

6.2 ML Implementation and Results

The development of the models described in the previous section generally resulted in poor performances, with the occurrence of severe overfitting. At this stage, an attempt was made to improve the most promising models, namely the SVM and Gradient Boosting models. Despite making adjustments to the models, the results remained subpar. This being said, SVM was the model that best avoided overfitting. While the SVM remained able not to overfit (in fact, it was slightly underfitted), the Gradient Boosting maintained the overfitting. In Table 6.1, it is possible to check the results for the generalized approach, i.e., a model for all the participants. The results displayed by the SVM and Gradient Boosting were the ones that received an attempt at improvement through the tuning of hyperparameters in this section. This was accomplished through a grid search or, in instances where it was computationally infeasible, through manual experimentation with different combinations of hyperparameters individually. The grid search used for parametrization had the following parameters: 'C' values of [0.1, 1, 10, 100] and 'kernel' types of ['linear', 'rbf']. Testing additional possibilities was not feasible due to limited computational resources.

Table 6.1: ML models' Results

Model	**Train. Acc.**	**Train. AUC**	**Train. F1**	**Val. Acc.**	**Val. AUC**	**Val. F1**	**Test Acc.**	**Test AUC**	**Test F1**
XGBoost	0.69	0.75	0.66	0.50	0.57	0.50	0.41	0.58	0.46
R. F.	0.81	0.84	0.80	0.47	0.50	0.47	0.36	0.50	0.47
G. Boost.	0.87	0.88	0.87	0.53	0.59	0.54	0.63	0.68	0.65
SVM	0.61	0.73	0.52	0.53	0.65	0.53	0.70	0.66	0.69

As seen in table 6.1, the model with the best results was SVM, followed by Gradient Boosting. However, SVM was not as overfitted as the latter; in fact, it was slightly underfitted. After using grid search, the model was set on a linear kernel, and the C hyper-parameter was set to 1. The SVM model with a linear kernel only achieved 70% accuracy because it struggled to distinguish clear patterns between both classes. The data might not be linearly separable, limiting the linear kernel's effectiveness; however, as the RBF kernel presented worse results, it can also be concluded that separating the data through a hyperplane was also ineffective. The difficulty in distinguishing between the classes is likely due to issues with the data. Participants inaccurately self-reporting their drowsiness levels can lead to the models making incorrect predictions. Additionally, data collected at different times of day might reflect varying levels of alertness, affecting consistency. Furthermore, participants might have reported their current drowsiness level instead of how drowsy they had been feeling for the last 5-minute interval, leading to inconsistent data. Out of the data selected from 30 participants, the data from 24 were used for

training/validation, and the remaining 6 were used for testing. In the initial fold, the training set included a subset of 200 samples taken from the full dataset. The training set was expanded for each subsequent fold by adding 20 samples. The size of the validation set remained consistent at 80 samples across all folds.

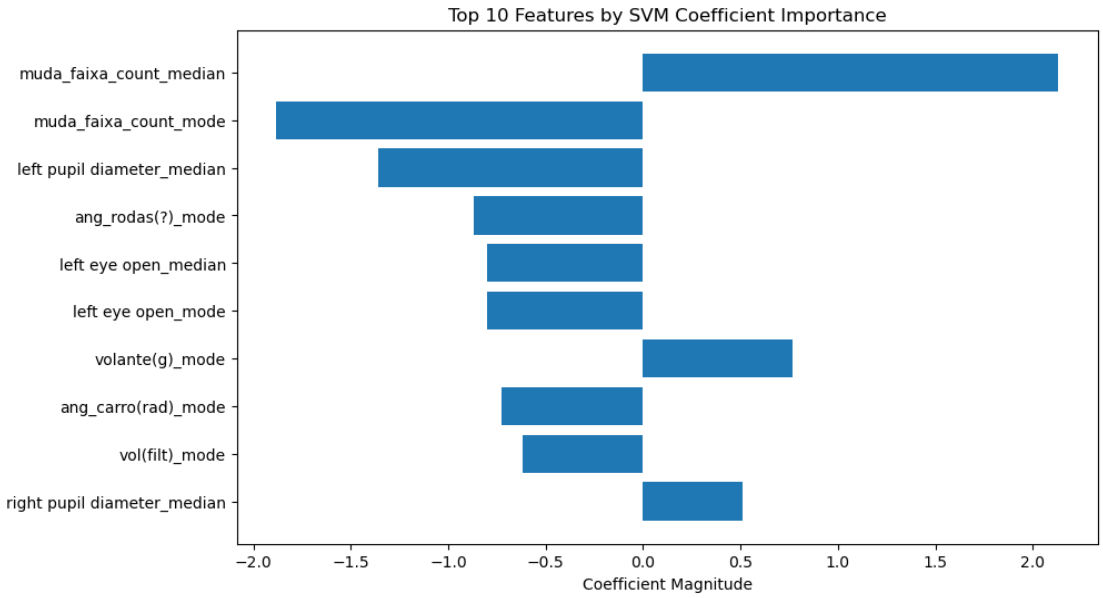

Figure 6.4: SVM - Feature Importance

Figure 6.4 displays the top 10 features ranked by their coefficient importance in the SVM model. This ranking highlights the most influential features in the model's decision-making process. The feature 'muda_faixa_count_median' holds the highest positive importance, indicating a strong positive correlation with the target variable, while 'muda_faixa_count_mode' has a substantial negative coefficient, suggesting an inverse relationship. These values can be interpreted by considering that a higher median value means a greater number of lane changes, which usually happens when the driver is drowsy, causing them to lose control of the car momentarily. In the case of the mode, these values can be interpreted as a lower level of activity, where the driver has the car under control and drives in their normal lane. Other significant features include 'left pupil diameter_median', 'ang_rodas(?)_mode', and 'left eye open_median'. These features reflect various eye measurements and aspects of vehicle dynamics, underscoring their relevance in the model. The differences in the coefficient magnitudes suggest varied impacts on the model's predictions, with some features like 'volante(g)_mode' and 'ang_carro(rad)_mode' having a moderate positive influence, while others like 'vol(filt)_mode' show lesser importance.

This analysis provides valuable insights into which features are most critical for the SVM's performance, aiding in further refinement and interpretation of the model. In Fig. 6.5, the matrix confusion is displayed, where it is possible to see that the model is better at predicting when the label is 0 (i.e., participant without drowsiness) than when the label is 1 (i.e., drowsy participant). The model predicts label 0 well 75.4% of the time, while it predicts label 1 correctly 63.5% of the time.

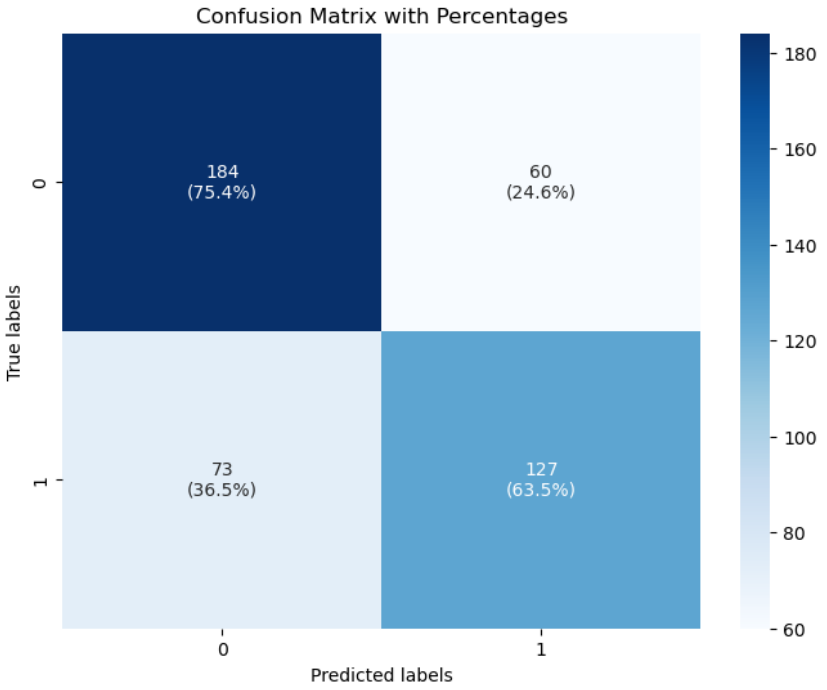

Figure 6.5: SVM - Confusion Matrix

Although the results of these models were not optimal, they demonstrated that they are promising. With further refinement, more data, and the subsequent incorporation of the clustering approach, it would be interesting to see how these models would behave. While comparing the results of this work to those presented in section 3.2 is not an easy task, some remarks can be made. The experiment conducted in McDonald et al. (2014) is perhaps the experiment most similar to the one used on the BBAI project, using data from a driving simulator and an eye-tracking device. By implementing an RF model, McDonald et al. (2014) manages to reach 70% AUC. While the results of the RF model implemented in this thesis are not as good as these, the SVM model reached 66% AUC and the Gradient Boosting model reached 68%. However, other studies, such as the one conducted by Hasan et al. (2024), can achieve an accuracy of 80.1%. However, this study was conducted in a naturalistic scenario, which makes the comparison more

unrealistic. Once again, it is important to state the difficulty in comparing the results of state-of-the-art literature with our implementation because these experiments' designs, devices, and data have great variability.

Chapter 7

Concluding Remarks

This study aimed to develop a tool that aids researchers in processing and analyzing driver behavior data in future experiments and a robust ML model that predicts driver drowsiness using data from a multitude of devices used in the BBAI experiment.

The interface presents an automatic and multifunctional pipeline able to transform raw data into refined data, which can be analyzed by researchers or used in ML models such as the one also developed in this project. It is also quite versatile due to the possibility of joining files from devices that were not included in the BBAI experiments. It can be concluded that, within the scope of this work, it was possible to automate the processes until data fusion, which includes data extraction, preparation, feature engineering, and fusion. It can also be concluded that data from devices other than those present in the BBAI experiment can be fully incorporated as long as it has a feature related to a timestamp and the identification of the participant. Analyzing RQ1, it can be concluded that machine learning models are useful tools that provide insightful evidence into the nature of drowsiness and distraction, as seen by the literature review provided. These models are often able to find patterns otherwise difficult to detect. As seen by the implemented SVM model in this work, it could be concluded that lane changes are a good indicator of drowsiness. Implementing the ML models proved challenging; however, the SVM model emerged as the most suitable, achieving 70% accuracy, a 66% AUC score, and a 69% F1 score. Additionally, it effectively prevented overfitting, which was a significant issue during the implementation of other ML models. Looking at the research question RQ2, we can understand that it was not possible to arrive at state-of-the-art values, even if we come close to the results of the studies that are more similar to ours, namely McDonald et al. (2014). However, it should be emphasized that our data is not always comparable to that used in the state of the art due to how the experiments are designed. The design of these experiments varies in the equipment used, driving scenarios, values, and variables recorded, making comparison difficult.

It is also important to mention that this work is fully aligned with Goal 3 from the United Nations Sustainable Development Goals (SDG) – 2030 Agenda, which aims to ensure healthy lives and promote well-being for all at all ages. Under this SDG, the objective was to help reduce the number of global deaths and injuries from road traffic accidents. Current estimates suggest that fatigue effects account for 20% of all fatal and severe crashes. Road accidents are the first cause of death in youngsters (www.opj.ics.ulisboa.pt) in Portugal and European Union. This work focuses on the study of driver fatigue. Road safety is also related to SDG 11, which is about making cities and human settlements inclusive, safe, resilient, and sustainable. Driver fatigue affects all kinds of traveling environments, including urban roads.

7.1 Limitations

During the development of the project, several limitations posed significant challenges. The lack of data was a main concern because it made the approach of creating different driver profiles not feasible. Data consistency issues were also a concern; the design of the experiment might have caused some data consistency issues that were reflected in the results of the ML models. Existing computing capacity was also an obstacle to developing the models, especially parameterizing them. Due to the lack of a reference to identify when the participant is distracted, it was impossible to develop ML models to detect this behavior. It would be interesting to design future experiments with this in mind. In addition, features related to the offset axis created with the drowsiness experiment in mind could not have been calculated for the distraction data. This is because we didn't have access to the simulator's configuration files containing information relating to the measurements of the distraction experiment scenario. The distraction experiment was not designed to detect distraction but to study the driving performance under distraction events when a critical event occurs.

7.2 Future Works

There are also a few possibilities for further exploration and enhancement of the proposed pipeline in further works.

1. **Development of Graphical User Interface (GUI)**: Transforming the current command-line interface into a user-friendly graphical interface represents an important advancement. Developing a GUI would enhance accessibility and usability for non-technical users, allowing intuitive interaction with the pipeline and its outputs. Incorporating visualizations

and interactive controls within the GUI would streamline, for example, data input and result interpretation, thereby making the tool more accessible to its users. The GUI was not implemented due to time constraints;

2. **Enhancement of Driver Profile Clustering**: Creation of a viable driver clustering technique could be pursued to improve the accuracy and granularity of driver profile segmentation. The utilization of advanced clustering algorithms, such as hierarchical clustering or density-based clustering methods, could lead to a more nuanced understanding of distinct driver groups based on behavioral and demographic attributes. Implementing these techniques effectively would necessitate a larger and more diverse dataset. This expanded dataset would provide the necessary depth and variability to discern subtle distinctions among driver profiles, improving the robustness of the clustering results. Collecting and integrating such data would be necessary in advancing the reliability of driver profiling methodologies;

3. **Bias Analysis**: It would be beneficial to investigate where bias is most introduced in the pipeline. This could be achieved by performing the processes in different orders and comparing the results of the ML models. For example, it would be insightful to see the effects of imputing missing values at a different stage, later in the pipeline. Understanding the impact of each step on bias could help in refining the pipeline for better accuracy and impartiality.

4. **Model Experimentation**: Future work should include more extensive experimentation with the ML models. Specifically, it would be valuable to test the models using data from various combinations of devices or even individual devices, rather than the fusion of all of them. This approach could reveal how each device affects the model's performance, leading to better-targeted data sources and potentially more accurate outcomes.

5. **Integration of ML Models in the Pipeline**: Incorporating the ML models into the interface would be an interesting next step. It would provide future researchers with a simplified way of using the ML models to predict drowsiness, where the input of unfused data could be easily transformed into the results of an ML model. When integrating ML models into the interface, it would also be interesting to take the opportunity to apply some MLOps concepts, such as CI/CD, which have been mentioned throughout the work but have not been implemented as there was no opportunity to do so. Although this integration has not been implemented yet due to time constraints and the necessity for further refinement of the models to achieve better results, the ability to achieve this, combined

with a transformation of the command line interface into a GUI, would be a powerful tool for future research;

6. **Predictive Model**: As mentioned above, the models implemented were developed to detect drowsiness. Predicting drowsiness is difficult and is a topic with a complicated definition that does not have a single defined approach. It would, therefore, be interesting to evolve the models that have been implemented in this work and try to implement or adapt them in the context of prediction.

In summary, future developments in driver profiling should focus on creating clustering techniques to segment driver profiles accurately. Transitioning from a command-line interface to a graphical user interface (GUI) will improve accessibility, allowing users to interact more intuitively with the program. Analyzing bias in the results and experimenting with different combinations of data sources for the ML models are also areas to explore in future work. Integrating advanced ML models directly into the GUI will further enable robust prediction of driver behaviors, facilitating sophisticated analysis and decision-making in driver behavior research. It would also be interesting to try to adapt the models to the prediction context and see how they perform in future work. These advancements aim to bolster the framework's reliability and usability, advancing applications in transportation safety and driver assistance systems.

Bibliography

Abe, T. (2023). PERCLOS-based technologies for detecting drowsiness: current evidence and future directions. *SLEEP Advances*, 4(1):zpad006.

Ahangari, S., Jeihani, M., and Dehzangi, A. (2020). A machine learning distracted driving prediction model. In *Proceedings of the 3rd International Conference on Vision, Image and Signal Processing*, ICVISP 2019, New York, NY, USA. Association for Computing Machinery.

Aksjonov, A., Nedoma, P., Vodovozov, V., Petlenkov, E., and Herrmann, M. (2019). Detection and Evaluation of Driver Distraction Using Machine Learning and Fuzzy Logic. *IEEE Transactions on Intelligent Transportation Systems*, 20(6):2048–2059.

Atrey, P. K., Hossain, M. A., El Saddik, A., and Kankanhalli, M. S. (2010). Multimodal fusion for multimedia analysis: a survey. *Multimedia Systems*, 16(6):345–379.

Baltrusaitis, T., Ahuja, C., and Morency, L.-P. (2019). Multimodal Machine Learning: A Survey and Taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(2):423–443.

Bangor, A., Kortum, P., and Miller, J. (2009). Determining what individual sus scores mean: Adding an adjective rating scale. *Journal of usability studies*, 4(3):114–123.

Bellenger, A. and Gatepaille, S. (2011). Uncertainty in ontologies: Dempster-shafer theory for data fusion applications. *CoRR*, abs/1106.3876.

Blessing, E. and Klaus, H. (2023). Pipeline Automation: Techniques to create efficient and automated data processing pipelines for ML models.

Bodor, A., Hnida, M., and Najima, D. (2023). Mlops: Overview of current state and future directions. In Ben Ahmed, M., Boudhir, A. A., Santos, D., Dionisio, R., and Benaya, N., editors, *Innovations in Smart Cities Applications Volume 6*, pages 156–165, Cham. Springer International Publishing.

Castro-Lopez, O. and Vega-Lopez, I. F. (2019). Multi-target compiler for the deployment of machine learning models. In *2019 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*, pages 280–281.

Chen, J.-X. and Zhao, X.-Y. (2023). A general integrated machine learning pipeline: Its concept, main steps and application in shear strength prediction of RC beams strengthened with FRCM. *Engineering Structures*, 281:115749.

Chui, K. T., Zhao, M., and Gupta, B. B. (2021). Long Short-Term Memory Networks for Driver Drowsiness and Stress Prediction. In Vasant, P., Zelinka, I., and Weber, G.-W., editors, *Intelligent Computing and Optimization*, Advances in Intelligent Systems and Computing, pages 670–680, Cham. Springer International Publishing.

de Naurois, C. J., Bourdin, C., Stratulat, A., Diaz, E., and Vercher, J.-L. (2018). Adapting artificial neural networks to a specific driver enhances detection and prediction of drowsiness. *Accident Analysis & Prevention*, 121:118–128.

Ding, J., Tarokh, V., and Yang, Y. (2018). Model selection techniques: An overview. *IEEE Signal Processing Magazine*, 35(6):16–34.

European Comission (2003). *Final report of the eSafety Working Group on road safety*. Publications Office. Commission, European and Society, Directorate-General for the Information and Media.

Fatichah, C., Sammy Wiyadi, P. D., Adni Navastara, D., Suciati, N., and Munif, A. (2020). Incident Detection based on Multimodal data from Social Media using Deep Learning Methods. In *2020 International Conference on ICT for Smart Society (ICISS)*, pages 1–6, Bandung, Indonesia. IEEE.

Gaonkar, A., Chukkapalli, Y., Raman, P. J., Srikanth, S., and Gurugopinath, S. (2021). A comprehensive survey on multimodal data representation and information fusion algorithms. In *2021 International Conference on Intelligent Technologies (CONIT)*, pages 1–8.

Hallvig, D., Anund, A., Fors, C., Kecklund, G., and Åkerstedt, T. (2014). Real driving at night – Predicting lane departures from physiological and subjective sleepiness. *Biological Psychology*, 101:18–23.

Hasan, M. M., Watling, C. N., and Larue, G. S. (2024). Validation and interpretation of a multimodal drowsiness detection system using explainable machine learning. *Computer Methods and Programs in Biomedicine*, 243:107925.

Hassani, S., Dackermann, U., Mousavi, M., and Li, J. (2024). A systematic review of data fusion techniques for optimized structural health monitoring. *Information Fusion*, 103:102136.

Huang, X., Jin, G., and Ruan, W. (2023). *Machine Learning Safety*. Artificial Intelligence: Foundations, Theory, and Algorithms. Springer Nature Singapore, Singapore.

Iyengar, G., Nock, H., and Neti, C. (2003). Audio-visual synchrony for detection of monologues in video archives. In *2003 International Conference on Multimedia and Expo. ICME '03. Proceedings (Cat. No.03TH8698)*, pages I–329, Baltimore, MD, USA. IEEE.

John, M. M., Olsson, H. H., and Bosch, J. (2021). Towards MLOps: A Framework and Maturity Model. In *2021 47th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 1–8, Palermo, Italy. IEEE.

Kang, M. and Tian, J. (2018). *Machine Learning: Data Pre-processing*, chapter 5, pages 111–130. John Wiley Sons, Ltd.

Kreuzberger, D., Kühl, N., and Hirschl, S. (2022). Machine Learning Operations (MLOps): Overview, Definition, and Architecture. arXiv:2205.02302 [cs].

Lenné, M. G. and Jacobs, E. E. (2016). Predicting drowsiness-related driving events: a review of recent research methods and future opportunities. *Theoretical Issues in Ergonomics Science*, 17(5-6):533–553.

Liu, S., Smith, K., and Che, H. (2015). A multivariate based event detection method and performance comparison with two baseline methods. *Water Research*, 80:109–118.

McDonald, A. D., Lee, J. D., Schwarz, C., and Brown, T. L. (2014). Steering in a Random Forest: Ensemble Learning for Detecting Drowsiness-Related Lane Departures. *Human Factors*, 56(5):986–998. Publisher: SAGE Publications Inc.

Mehmood, R., Alam, F., Albogami, N. N., Katib, I., Albeshri, A., and Altowaijri, S. M. (2017). UTiLearn: A Personalised Ubiquitous Teaching and Learning System for Smart Societies. *IEEE Access*, 5:2615–2635. Conference Name: IEEE Access.

Misra, A., Samuel, S., Cao, S., and Shariatmadari, K. (2023). Detection of driver cognitive distraction using machine learning methods. *IEEE Access*, 11:18000–18012.

Mumuni, A. and Mumuni, F. (2024). Automated data processing and feature engineering for deep learning and big data applications: a survey. *Journal of Information and Intelligence*, page S2949715924000027.

Nakano, K. and Chakraborty, B. (2022). Real-Time Distraction Detection from Driving Data Based Personal Driving Model Using Deep Learning. *International Journal of Intelligent Transportation Systems Research*, 20(1):238–251.

Naujoks, F., Kiesel, A., and Neukum, A. (2016). Cooperative warning systems: The impact of false and unnecessary alarms on drivers' compliance. *Accident Analysis & Prevention*, 97:162–175.

Nefian, A. V., Liang, L., Pi, X., Liu, X., and Murphy, K. (2002). Dynamic Bayesian Networks for Audio-Visual Speech Recognition. *EURASIP Journal on Advances in Signal Processing*, 2002(11):783042.

Pang, C., Hindle, A., and Barbosa, D. (2020). Understanding devops education with grounded theory. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering Education and Training*, pages 107–118, Seoul South Korea. ACM.

Pires, I., Garcia, N., Pombo, N., and Flórez-Revuelta, F. (2016). From Data Acquisition to Data Fusion: A Comprehensive Review and a Roadmap for the Identification of Activities of Daily Living Using Mobile Devices. *Sensors*, 16(2):184.

Recupito, G., Pecorelli, F., Catolino, G., Moreschini, S., Nucci, D. D., Palomba, F., and Tamburri, D. A. (2022). A Multivocal Literature Review of MLOps Tools and Features. In *2022 48th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 84–91, Gran Canaria, Spain. IEEE.

Rodrigues, C., Faria, B. M., and Reis, L. P. (2021). Detecting, Predicting, and Preventing Driver Drowsiness with Wrist-Wearable Devices. In Marreiros, G., Melo, F. S., Lau, N., Lopes Cardoso, H., and Reis, L. P., editors, *Progress in Artificial Intelligence*, pages 109–120, Cham. Springer International Publishing.

Rottensteiner, F., Trinder, J., Clode, S., and Kubik, K. (2005). Using the Dempster–Shafer method for the fusion of LIDAR data and multi-spectral images for building detection. *Information Fusion*, 6(4):283–300.

Rudin-Brown, C. M. and Filtness, A. J. (2023). *The Handbook of Fatigue Management in Transportation: Waking Up to the Challenge*. CRC Press, New York, 1 edition.

Saha, A. K., Mridha, M. F., Rafiq, J. I., and Das, J. K. (2017). Data extraction from natural language using universal networking language. In *2017 International Conference on Current*

Trends in Computer, Electrical, Electronics and Communication (CTCEEC), pages 1228–1232.

Soares, S., Campos, C., Leitão, J. M., Lobo, A., Couto, A., and Ferreira, S. (2021). Distractive Tasks and the Influence of Driver Attributes. *Sustainability*, 13(9):5094.

Soares, S., Ferreira, S., and Couto, A. (2020a). Driving simulator experiments to study drowsiness: A systematic review. *Traffic Injury Prevention*, 21(1):29–37. Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/15389588.2019.1706088.

Soares, S., Monteiro, T., Lobo, A., Couto, A., Cunha, L., and Ferreira, S. (2020b). Analyzing driver drowsiness: From causes to effects. *Sustainability*, 12(5).

Subramanya, R., Sierla, S., and Vyatkin, V. (2022). From DevOps to MLOps: Overview and Application to Electricity Market Forecasting. *Applied Sciences*, 12(19):9851.

Symeonidis, G., Nerantzis, E., Kazakis, A., and Papakostas, G. A. (2022). MLOps - Definitions, Tools and Challenges. In *2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 0453–0460.

Testi, M., Ballabio, M., Frontoni, E., Iannello, G., Moccia, S., Soda, P., and Vessio, G. (2022). MLOps: A Taxonomy and a Methodology. *IEEE Access*, 10:63606–63618.

Varoquaux, G. and Colliot, O. (2023). *Evaluating Machine Learning Models and Their Diagnostic Value*, pages 601–630. Springer US, New York, NY.

Wang, X., Xu, R., Zhang, S., Zhuang, Y., and Wang, Y. (2022). Driver distraction detection based on vehicle dynamics using naturalistic driving data. *Transportation Research Part C: Emerging Technologies*, 136:103561.

World Health Organization (2023). Global status report on road safety 2023. *Geneva: World Health Organization*.

Wu, Y., Jiang, X., Guo, Y., Zhu, H., Dai, C., and Chen, W. (2023). Physiological measurements for driving drowsiness: A comparative study of multi-modality feature fusion and selection. *Computers in Biology and Medicine*, 167:107590.

Yuan, N. J., Zheng, Y., Xie, X., Wang, Y., Zheng, K., and Xiong, H. (2015). Discovering Urban Functional Zones Using Latent Activity Trajectories. *IEEE Transactions on Knowledge and Data Engineering*, 27(3):712–725.

Zhang, L., Xie, Y., Xidao, L., and Zhang, X. (2018). Multi-source heterogeneous data fusion. In *2018 International Conference on Artificial Intelligence and Big Data (ICAIBD)*, pages 47–51, Chengdu. IEEE.

Zhang, S., Theagarajan, L. N., Choi, S., and Varshney, P. K. (2019). Fusion of Correlated Decisions Using Regular Vine Copulas. *IEEE Transactions on Signal Processing*, 67(8):2066–2079. arXiv:1803.09350 [eess].

Zheng, W.-L., Liu, W., Lu, Y., Lu, B.-L., and Cichocki, A. (2019). EmotionMeter: A Multimodal Framework for Recognizing Human Emotions. *IEEE Transactions on Cybernetics*, 49(3):1110–1122.

Zheng, Y. (2015). Methodologies for Cross-Domain Data Fusion: An Overview. *IEEE Transactions on Big Data*, 1(1):16–34.

Zhou, C., Sun, C., Liu, Z., and Lau, F. C. M. (2015). A C-LSTM Neural Network for Text Classification. arXiv:1511.08630 [cs].

Åkerstedt, T. and Gillberg, M. (1990). Subjective and Objective Sleepiness in the Active Individual. *International Journal of Neuroscience*, 52(1-2):29–37.

Appendix A

Pipeline User Guide

This guide intends to provide guidance to the users of this pipeline. This tool provides a command-line interface allowing users to join data from multiple devices and different formats, focusing on data from driving simulators. The tool containing the pipeline is launched in the command line by running the script through the command 'python configuration.py'. To perform this action, it is necessary to be in the directory that contains the Python file. The program has three main options:

a) Extracting and fusing files from the BBAI experiments;

b) Extracting and fusing files from other experiments;

c) Mapping features of datasets from other experiments.

A.1 Option A

The first option automatically fuses and preprocesses the data exactly as it is extracted from the devices used in the BBAI experiment. When selecting this option, it is asked to indicate the experiment context: drowsiness or distraction. Once this choice has been made, a list of options containing the possible files to extract is presented, as seen in figure A.1. After the user chooses the desired combination of files, it is presented with a file explorer window in order to select the folder that contains to be extracted and fused.

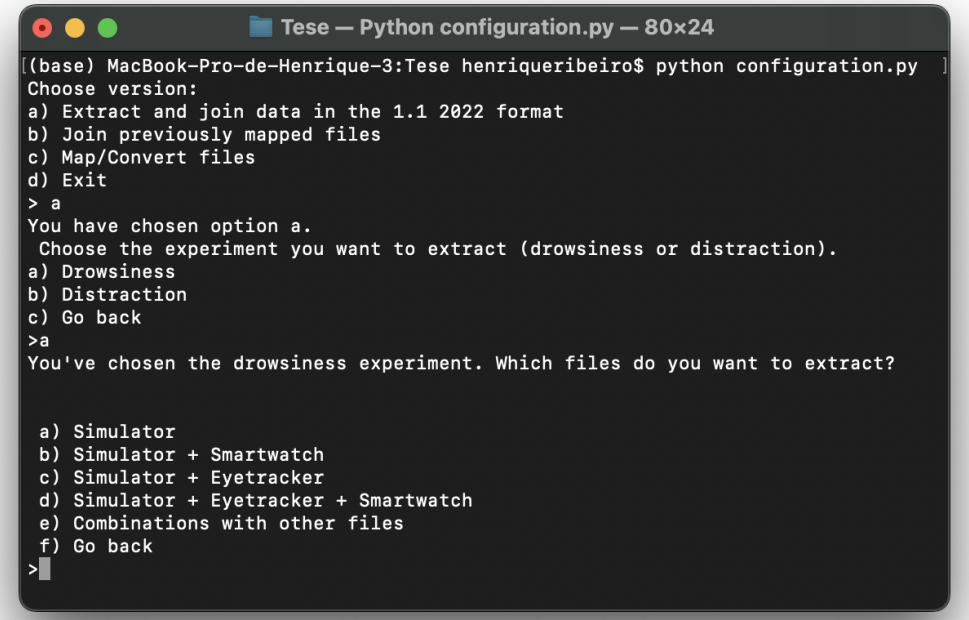

Figure A.1: Option A - Possible combinations of devices

Since the names and directories of the files are not always the same, the directories where the files are located must be in one (or more) of the formats listed below. In the directory formats listed below, the variable *base_path* represents the directory or folder chosen by the user, *i* denotes the participant's ID, and *type* indicates the type of gaze files, which may be 'Left', 'Right', or 'Vergence'. The asterisk (*) is used as a wildcard character to denote any sequence of characters that may appear in that part of the directory structure. The variable *exp* represents the experiment and can be either 'dist' for distraction data or 'sono' for drowsiness data.

Eyetracker - Possible directories:

- `{base_path}/{i}/{exp}_{i}_*_*_*/{exp}_{i}_eyetracker_*_*_*/*/*/* Gaze-{type}.csv`
- `{base_path}/{i}/{exp}_{i}_*_*_*/{exp}_{i}_eyetracker_*_*_*/*/* Gaze-{type}.csv`
- `{base_path}/{i}/{exp}_{i}_*_*_*/{exp}_{i}_eyetracker_*_*_*/*/Outputs/* Gaze-{type}.csv`
- `{base_path}/{i}/{exp}_{i}_*_*_*/{exp}_{i}_eyetracker_*_*_*/* Gaze-{type}.csv`

Smartwatch - Possible directories:

- `{base_path}/{i}/{exp}_{i}_*_*_*/{exp}_{i}_smartwatch_*_*_*/*.csv`

Simulator .log Files - Possible directories:

- `{base_path}/{i}/{exp_}{i}_*_*_*/{exp_}{i}_simulador/{exp_}{i}_simulador_telnav.log`
- `{base_path}/{i}/{exp_}{i}_*_*_*/{exp_}{i}_simulador/{exp_}{i}_simulador.log`

Simulator .txt Files - Possible directories:

- `{base_path}/{i}/{exp_}{i}_*_*_*/{exp_}{i}_simulador/{exp_}{i}_simulador_txt.txt`
- `{base_path}/{i}/{exp_}{i}_*_*_*/{exp_}{i}_simulador/{exp_}{i}_simulador.txt`

Once the folder containing the files has been chosen, the program fuses the files and asks for the folder where the files shall be saved.

If the user, when prompted to select the combination of desired files chooses the last option ("e) Combinations with other files") after the process described, above a new step is added. Even with this option, the only strictly necessary files are the ones related to the simulator. The user is prompted if he would like to select a single file or a folder containing multiple files. The program then opens a file explorer window that lets the user select the file/folder containing generic data to the already existing fused dataframe. However, this is only possible if the newly added files contain a feature called 'tempo (s)', representative of the timestamp, and other called 'id' , which is the identification number of the participant. It is essential to note that in instances where the user opts to select a folder containing generic data, only the initial file corresponding to a repeating participant identifier (ID) will be integrated. The selection of the folder is designed to amalgamate multiple files from the same device across different participants, rather than to compile files from multiple devices for a single user. In this option, the algorithm performs several enhancements to the data, such as, missing value imputation, outliers treatment and feature engineering.

A.2 Option B

This option allows the user to merge multiple, much like the first option does, however, this does not follow the structure of files present in the BBAI version. In this regard, instead of selecting only a folder and the algorithm "finding" where the relevant files are, the user needs to select the files or folder containing them regarding each device. After selecting the desired combination of devices, the user is then prompted to select each file/folder individually, as aforementioned. After selecting, for example, the combination of Simulator and Smartwatch, the user is asked whether he would like to select a file or a folder, and once he has done this a window opens with a prompt asking the user to select the file/folder containing the data related to the vehicle simulator (velocity, etc.) (as seen on figure A.2, followed by a prompt asking for the data related

to the participant and interaction with vehicle IVIS system (ID, KSS, etc.) and finally the data related to the smartwatch. To successfully merge files using this method, the features within the files must follow certain naming conventions. These naming conventions are presented before prompting the user with the files that have this requirement. To solve this problem, option c) is provided in the program. It is also necessary that the files containing this data are in .csv format, using a semicolon (';') as a separator.

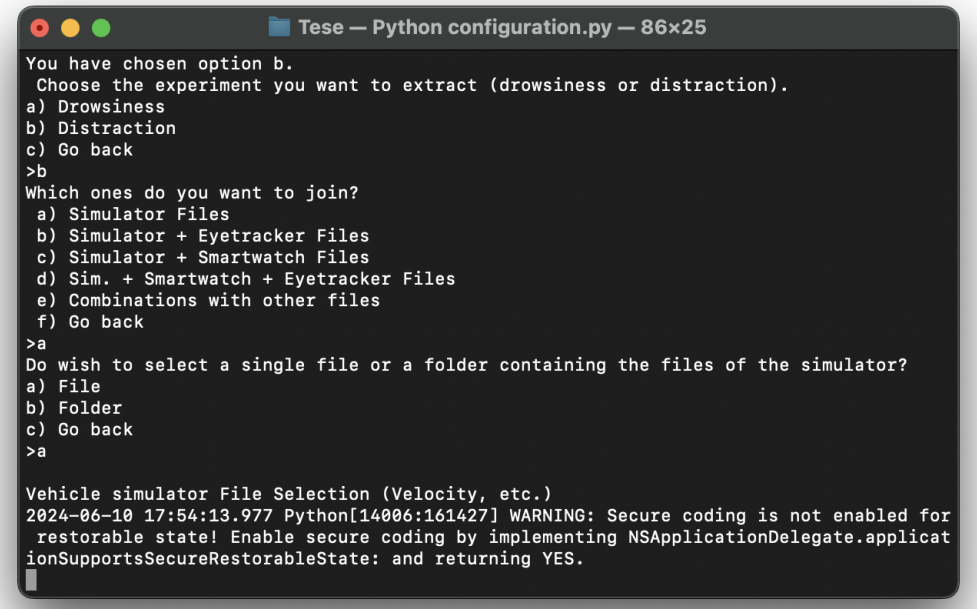

Figure A.2: Option B - File choice

A.3 Option C

This option allows the user to map certain features to the denominations needed to merge the data in option B. This option, once again, prompts the user to select the experiment, the device whose features he wants to map, and, if he wants to select a file or a folder containing a larger number of files. Once the user selects his preferences and the directory of the files/folder, he is asked whether he wants to perform the mapping manually or through a preconfigured file. If the manual option is chosen, the user is asked to map the features manually, as seen on figure A.3. Here, the participant must input the names of the features of the original files that correspond to

the ones that appear on the interface. He may skip the features that are not useful in relation to their input data by writing 'skip' or pressing enter. Nonetheless, some features are mandatory, and when trying to skip, the program will force the user to do so. After manually mapping the features, a prompt appears asking the user whether he may want to add additional features contained in the input files to the output file. By writing 'yes' the user can write the names of the desired features and subsequently they will be added to the output files. If the user opts to map the features using a pre-configured file, they will be prompted to select the appropriate file. This file must be in .json format, similar to the one depicted in the listing A.1, with mappings located on the right side of the file. Upon uploading the .json file to the program, the mappings will be executed automatically. Following this, the user will receive a prompt to save the newly configured files.

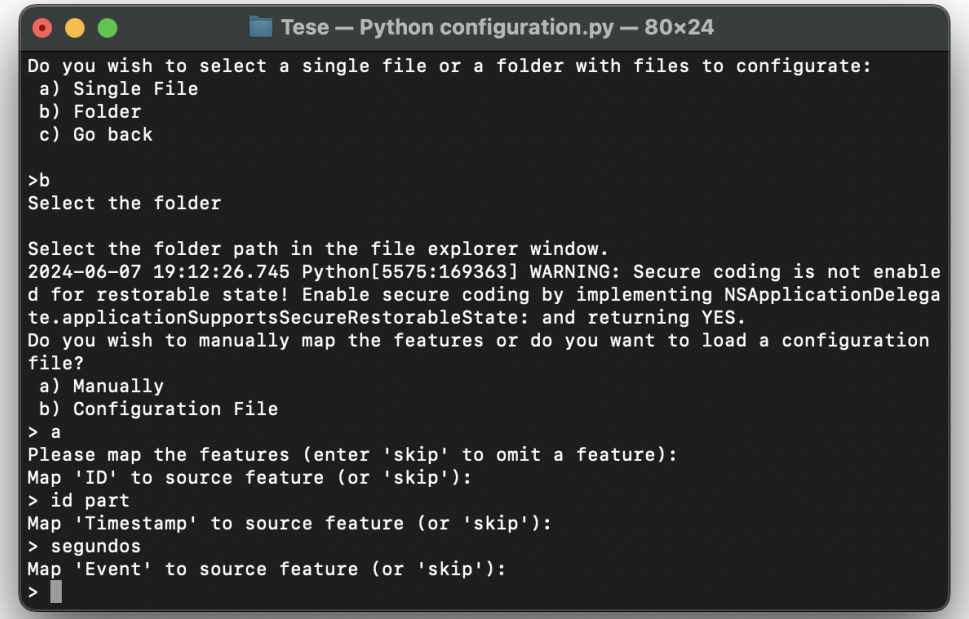

Figure A.3: Option C - Feature mapping

```
1 {
2 "column_mapping": {
3 "ID": {"mandatory": true},
```

```
4 "Timestamp": {"mandatory": true},
5 "KSS": {"mandatory": true}
6 }
```

Listing A.1: Pre-configured JSON file

We hope this guide has provided you with all the necessary information to utilize our tool effectively. For further assistance contact henriquebribeiro00@gmail.com.

Appendix B

Validation Tasks

Henrique Ribeiro - Master's in Engineering and Data Science

The following tasks are included in the Master's thesis 'Developing an Automated Pipeline for Driver Behavior Analysis'. The purpose of the program is to unify the different files extracted from the different pieces of equipment used in the driving simulator experiments. This program had to be developed because of the large number of files (and file formats) that are extracted from the devices. The diagram B.1 shows the different files that are extracted as part of these experiments. In addition to collecting the different types of files that are extracted during the experiment, the program developed also collects other types of generic files and map files using configuration files (manual and automatic). The program uses a sequence of commands and menus with the aim of structuring the synchronised information for subsequent analysis, including extraction, combining the different data sources, mapping variables (manually or through pre-configured configuration files) and other steps necessary for the purpose. The first main option presented in the program relates to previously conducted experiments, namely the BBAI experiment (on sleep and distraction), and allows the data from the different sources to be extracted and combined. The second option allows you to join data that has been previously mapped and the third option allows you to map data so that it can then be unified.

While the tasks are being carried out, data will be recorded, including their duration, the number of errors made (i.e., the number of times the task had to be restarted and the number of times it had to be retraced), and the number of questions asked of the master's student. At the end of the tasks, feedback and suggestions on the program will be asked for.

1. **1st Task** - Extracting data from the BBAI version of the simulator:
   (a) Extract data from the 'Distraction' experiment - Simulator + Smartwatch
   (b) Data contained in the 'Task 1' folder

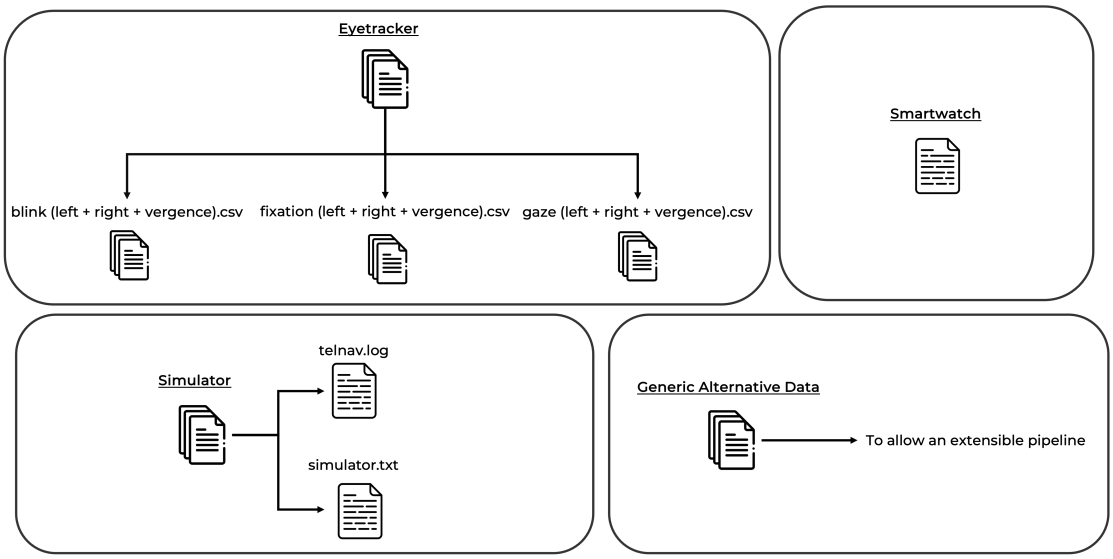

Figure B.1: Multi-Source Data Structure for Integration in Processing Pipeline

(c) Save extracted files in the /Users/henriqueribeiro/Desktop/Task 1 folder - Completed

2. **2nd Task** - Extracting data from the BBAI version of the simulator with additional files:

   (a) Extract data from the 'Distraction' experiment - another combination (Simulator + Smartwatch + Other file)

   (b) Main data contained in the 'Task 2/Main Files' folder

   (c) Additional file contained in the location /Users/henriqueribeiro/Desktop/Task 2/additional_file.csv

   (d) Save extracted files in the /Users/henriqueribeiro/Desktop/Task 2 folder - Completed

3. **3rd Task** - Manual mapping of variables:

   (a) Map variables to the 'Sleep' experiment

   (b) Type of device to map: Simulator - Participant data

   (c) Location of the data to be mapped: /Users/henriqueribeiro/Desktop/Task 3/file_to_map.csv

   (d) Save mapped files in the /Users/henriqueribeiro/Desktop/Task 3 folder – Completed

4. **4th Task** - Automatic variable mapping:

   (a) Map variables to the 'Sleep' experiment

   (b) Type of device to map: Smartwatch

(c) Location of the data to be mapped: /Users/henriqueribeiro/Desktop/Task 4/to_config_t4.csv

(d) Save extracted files in the /Users/henriqueribeiro/Desktop/Task 4 folder - Done.

(e) Use the pre-configured file in the folder /Users/henriqueribeiro/Desktop/Task 4/pre-config_log.json

5. **5th Task** - Extraction of previously mapped files:

   (a) Extract data from the 'Sleep' experiment - Simulator + Smartwatch

   (b) Data relating to the simulator (speed, acceleration...) in the location /Users/henriqueribeiro/Desktop/Task 5/car_data.csv

   (c) User data (ID, KSS) in the location /Users/henriqueribeiro/Desktop/Task 5/participant_kss.csv

   (d) Smartwatch data in location /Users/henriqueribeiro/Desktop/Task 5/smartwatch.csv

   (e) Save files in the /Users/henriqueribeiro/Desktop/Task 5 folder - Done

Answer the questions from 1 to 5:

1. I think the system is useful in the context of the simulator.
2. I found the system unnecessarily complex.
3. I found the system easy to use.
4. I think I would need the support of a technical person to be able to use this system.
5. I found the various functions of this system to be well integrated.
6. I think there is too much inconsistency in this system.
7. I imagine that most people would learn to use this system very quickly.
8. I needed to learn a lot of things before I started using this system.