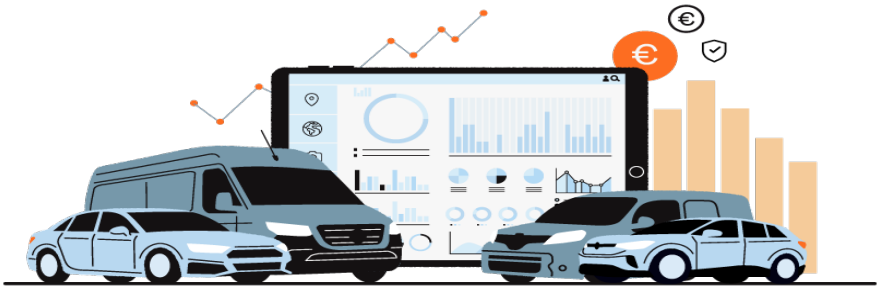
INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Departamento de Engenharia Eletrónica e de Telecomunicações e Computadores



Driver Profile Classification

Luís Miguel Pinto Loureiro

(Bachelor's degree)

Dissertação para obtenção do Grau de Mestre
em Engenharia Informática e de Computadores

Orientadores : Prof. Doutor André Ribeiro Lourenço
Prof. Doutor Artur Jorge Ferreira

Júri:

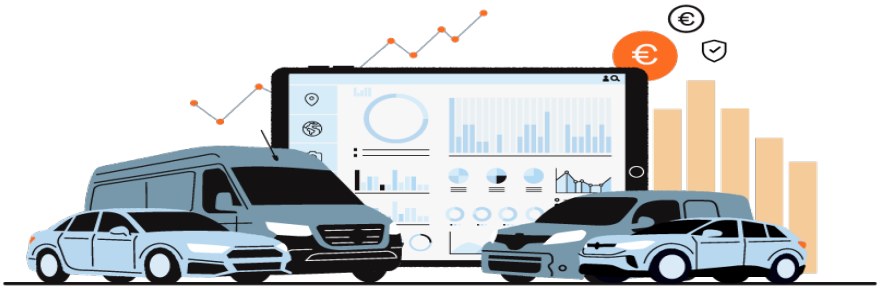
Presidente: Prof. Doutor Tiago Miguel Braga da Silva Dias

Vogais: Prof. Doutor Paulo Manuel Trigo Cândido da Silva
Prof. Doutor André Ribeiro Lourenço

July, 2023

INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Departamento de Engenharia Eletrónica e de Telecomunicações e Computadores



Driver Profile Classification

Luís Miguel Pinto Loureiro

(Bachelor's degree)

Dissertação para obtenção do Grau de Mestre
em Engenharia Informática e de Computadores

Orientadores : Prof. Doutor André Ribeiro Lourenço
Prof. Doutor Artur Jorge Ferreira

Júri:

Presidente: Prof. Doutor Tiago Miguel Braga da Silva Dias

Vogais: Prof. Doutor Paulo Manuel Trigo Cândido da Silva
Prof. Doutor André Ribeiro Lourenço

July, 2023

To my mom, dad, sisters and grandmothers.

Acknowledgments

To my thesis supervisors, Prof. Doutor André Ribeiro Lourenço and Prof. Doutor Artur Jorge Ferreira, for their support, incentive, guidance, permanent availability, and the opportunity that they provided me to realize this thesis.

To CardioID, in particular to Carlos Carreiras, for sharing the drivers' data and assisting with its analysis.

To ISEL, for all the knowledge transmitted over the years.

To my great friend, Diogo Guerreiro, for his support, his tremendous willingness to help perform tests, and his unmatched energy.

To my sister, Matilde, for her willingness to review the thesis.

To my family, for their support, patience, the time I took away from them, understanding, and constant concern.

My deepest gratitude to each of you,

Luís Loureiro

Abstract

Nowadays, we have billions of vehicles around the world. In most countries, insurance against civil liability for vehicles is mandatory. Usually, insurance companies define vehicle insurance rates according to static known time-invariant variables or with a slight adjustment over time. The insurance rates calculation relies mostly on factors such as the age of the driver, the number of years one holds a driving license, and the driving history. These variables are not in close connection with the everyday behavior of the driver on the road, thus being not fair for young drivers, for instance.

In this thesis, we follow a pay-as-you-behave approach, in which we describe the research and development work to devise a driver profile classification solution, based on driver behavior data. From data records with the trips from different drivers, we build a dataset, targeting a driver profile classification task. We explore and evaluate the use of machine learning techniques over the dataset, starting with an unsupervised approach and moving to a supervised one, where we save the final model. We also develop a REST API to serve as proof of concept to classify drivers and fleets, that uses the best model to determine the driver profile.

The experimental results show that unsupervised machine learning techniques can cluster and identify distinct trip profiles, and supervised learning techniques are capable of classifying different trips based on the profiles found, which are then used to assign the driver profile. Many drivers revealed to have inconsistent driver behavior over time. The majority of the drivers has a non-aggressive driving style, some are aggressive, and a few are risky style drivers.

Keywords: i-DREAMS project, driver profile, pay-as-you-behave, driving dataset, feature engineering, machine learning, unsupervised learning, supervised learning, clustering.

Resumo

Hoje em dia, existem milhares de milhões de veículos em todo o mundo. Na maioria dos países, seguro de veículo é obrigatório. Normalmente, as companhias de seguro definem as taxas de seguro de veículos de acordo com variáveis estáticas, ou que têm um ligeiro ajustamento ao longo tempo, como a idade do condutor, o número de anos encartado, e o histórico de condução. Estas variáveis não estão directamente relacionadas com o comportamento diário do condutor na estrada, não sendo por isso justas para os jovens condutores.

Nesta tese adoptamos uma abordagem pay-as-you-behave, na qual analisamos a literatura e descrevemos o trabalho desenvolvido para conceber uma solução de classificação do perfil de condução. A partir de registos de dados de viagens com diferentes condutores, desenvolvemos um dataset organizado por viagens. Exploramos e avaliamos o uso de técnicas de aprendizagem automática sobre o conjunto de dados, começando com uma aborgadem não supervisionada, e de seguida, supervisionada. Foi também desenvolvida uma REST API que serve como exemplo de uma abordagem para classificar condutores e frotas de veículos.

Os resultados experimentais mostram que as técnicas não supervisionadas são capazes de realizar agrupamento dos dados, e identificar diferentes perfis de condução. As técnicas supervisionadas conseguem classificar diferentes viagens com base nos perfis encontrados, que são depois utilizados para calcular o perfil de condução. No conjunto de dados utilizado, a maior parte dos condutores revelaram ter um comportamento inconsistente ao longo do tempo.

Palavras-chave: projeto i-DREAMS, perfil de condução, pay-as-you-behave, conjunto de dados de condução, extração de características, aprendizagem automática, aprendizagem não supervisionada, aprendizagem supervisionada, agrupamento.

Contents

List of Figures

List of Tables

List of Acronyms

ADASYN Adaptive Synthetic Sampling. 44, 62
API Application Programming Interface. 3, 4, 6, 51
AUC Area Under the Curve. 26
AWS Advanced Warning System. 30

BIC Bayesian Information Criterion. 41, 55
BLE Bluetooth Low Energy. 3

CAN Controller Area Network. 3, 4
CART Classification and Regression Trees. 23
CH Calinski-Harabasz. 24
CPU Central Processing Unit. 51

DB Davies-Bouldin. 24
DBMS Database Management System. 49
DBSCAN Density-Based Spatial Clustering of Applications with Noise. 19, 20
DT Decision Trees. 21, 22, 23, 42, 43, 57, 58, 60, 62, 66

EAC Evidence Accumulation Clustering. 21, 41, 56
EM Expectation-Maximization. 20
EU European Union. 2
EW-AHP Entropy weight Analytical hierarchy process. 15

FCW Forward Collision Warning. 32

FNR False Negative Rate. 26
FPR False Positive Rate. 25
FR Feature Reduction. 40
FS Feature Selection. 39, 40

GMM Gaussian Mixture Models. 20, 41, 55
GNSS Global Navigation Satellite System. 33
GPS Global Positioning System. 3, 11, 12, 15

HMW Headway Monitoring & Warning. 32
HTTP Hypertext Transfer Protocol. 6

IoT Internet of Things. 4, 9
IT Information Technology. 5

KSS Karolinska Sleepiness Scale. 31

LDW Lane Departure Warning. 32
LSTM Long Short-Term Memory. 17

ML Machine Learning. 4
MSE Mean Squared Error. 23

OBD On-Board Diagnostics. 11
ORM Object-Relational Mapping. 49

PAM Partition Around Medoids. 16
PCA Principal Component Analysis. 39, 40, 52, 53

RAM Random Access Memory. 51
REST Representational State Transfer Architectural Style. 6, 46
RF Random Forest. 23, 43, 57, 58, 60, 66
RNN Recurrent Neural Networks. 16, 17
ROC Receiver Operating Characteristics. 26

SMOTE Synthetic Minority Oversampling Technique. 44
SPC Statistical Process Control. 15

SQL Structured Query Language. 49
STD Standard Deviation. 43
STZ Safety Tolerance Zone. 3, 29, 33, 34, 35
SVD Singular Value Decomposition. 40, 52
SVM Support Vector Machine. 17, 21

UBI Usage-Based-Insurances. 5
URI Uniform Resource Identifier. 47

List of Symbols

c	number of classes in the dataset
d	number of features in the dataset
n	number of patterns in the dataset

1

Introduction

This chapter introduces the context in which this project falls in, as well as the motivations for its realization. The key objectives established to develop a system that allows the classification of driving profiles are presented.

Section 1.1 presents the current context regarding car insurance companies and also describes in part the reason for developing this project based on the i-DREAMS project [30]. Section 1.2 exposes the motivations for the development of the project. Section 1.3 describes the objectives of this project through the proposed driver profiling system. Finally, Section 1.5 is presented, which details the organization and structure of this document.

1.1 Context

According to «Our World in Data» 2014 statistics [46], there are billions of vehicles in the entire world, and in most countries, car insurance is mandatory to have legal permission to drive on public roads.

The owner or driver of a vehicle is responsible for any damage it may cause and may have to pay out large amounts of money in case of an accident. To protect the interests of injured parties, who are entitled to have their losses paid for, regardless of whether or not the person responsible for the accident is financially able to do so, insurance against civil liability for motor vehicles and their trailers is mandatory for most countries.

As reported in 2023 by the European Road Safety Observatory [31], from 2010 to 2019 there were approximately 1.000.000 crashes and 1.250.000 injuries per year in the European Union (EU). The number of crashes, injuries, and fatalities in the EU has shown a decreasing trend, in the last few years. Figure 1.1 and 1.2 show these results. An interesting outcome is a decrease in the number of fatalities from 2019 to 2020, mostly because of the COVID-19 pandemic restrictions.

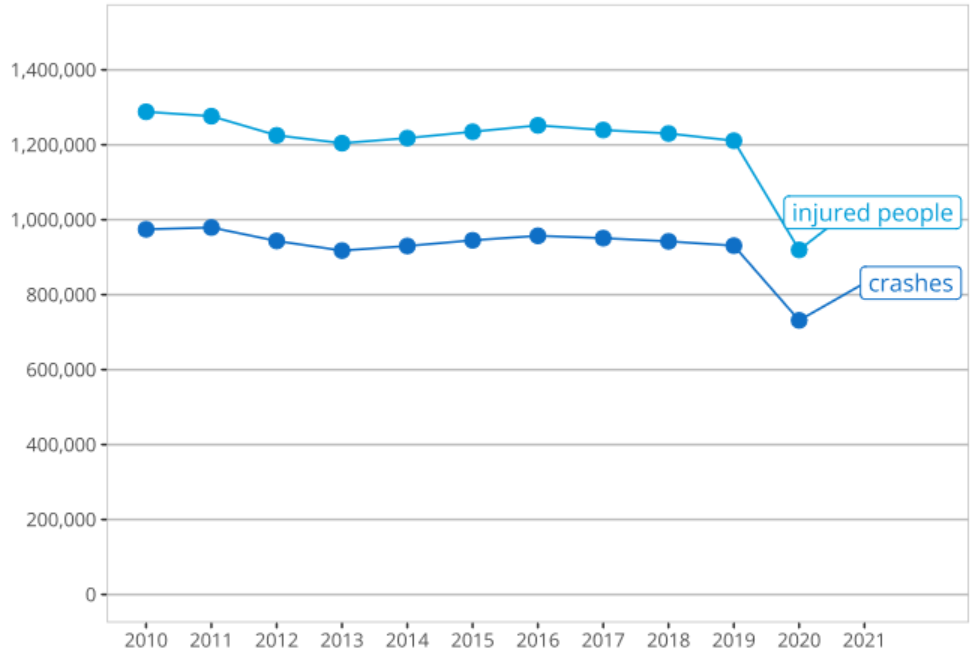


Figure 1.1: Road crashes and injured people in the European Union [31]

Another way to decrease these numbers is to develop systems that are able to determine driver behavior, and can be used both on a personal level and on a corporate level with fleets of vehicles. These decreases can also indicate an increase in profit for vehicle insurance companies. Insurance companies usually set vehicle insurance rates according to pre-defined variables that tend to be deterministic, meaning that their values are known and do not change with time, thus being considered static, or they change in a controlled manner. Furthermore, the insurance rates calculation relies most of the time on demographic information, such as the age of the driver, the number of years he or she has a driving license, and the driving history [23]. Although the idea has a statistical rationale and is being applied in most European countries, it ends up excessively penalizing not only young drivers (less than 25 years old) but also old ones (over 65 years old). Moreover, it turns to be ineffective to yield accurate insurance pricing over time resulting in profit decline for insurance companies and being unfair to drivers [28].

This project also arises in the context of the European Horizon2020 i-DREAMS

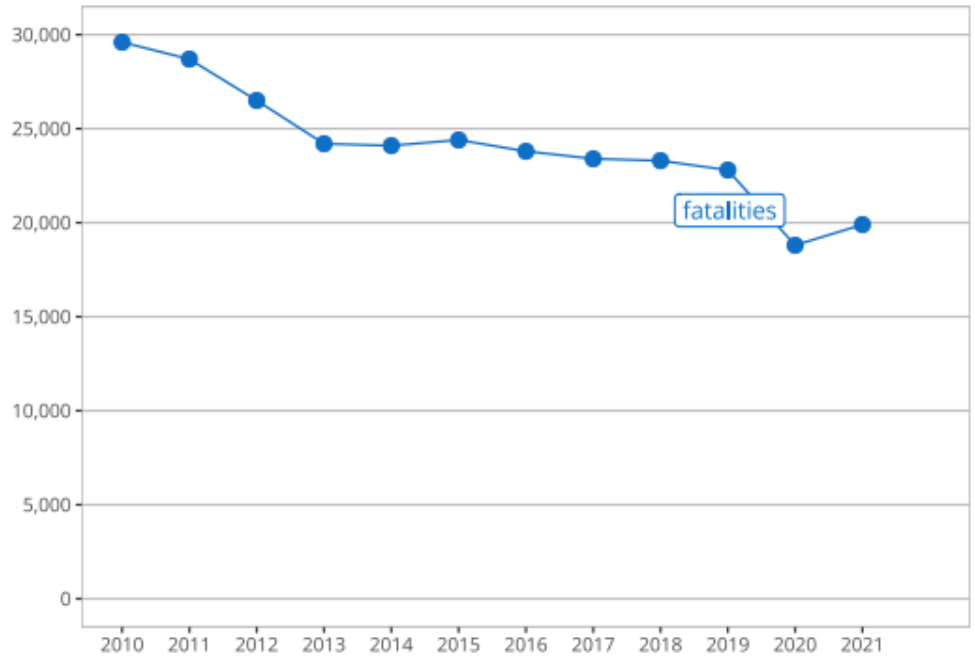


Figure 1.2: Fatalities in the European Union [31]

project [30] developed by the CardioID [11] company and associates. This project is based on a set of devices installed in vehicles that gather data about the driver's status, as well as a variety of other data related to driving behavior. In this regard, the i-DREAMS project aims at calculating the Safety Tolerance Zone (STZ) and interventions for driver-vehicle-environment interactions under challenging conditions. Figure 1.3 provides an overview of the devices installed in the vehicles for the i-DREAMS project.

The key components are the CardioWheel system (installed on the steering wheel), the Mobileye [43] device (installed on the windshield), the Gateway (GW in Figure 1.3), and a smartphone application. The i-DREAMS project also uses other components, which are not addressed in this project.

The Gateway corresponds to the central element of the system, which collects and centralizes the information coming from the other components and handles data connectivity and transmission. At the same time, it is an edge computing device that calculates the STZ, allows the triggering of alarms, and real-time communication with an Application Programming Interface (API) or storage for post-trip synchronization.

Currently, it supports connection via Ethernet, WiFi, and mobile data (3G/4G). This element also has two independent Controller Area Network (CAN) interfaces, Bluetooth Low Energy (BLE), a Global Positioning System (GPS) module, and the ability to connect to a dashcam. This dashcam is activated by events triggered from the

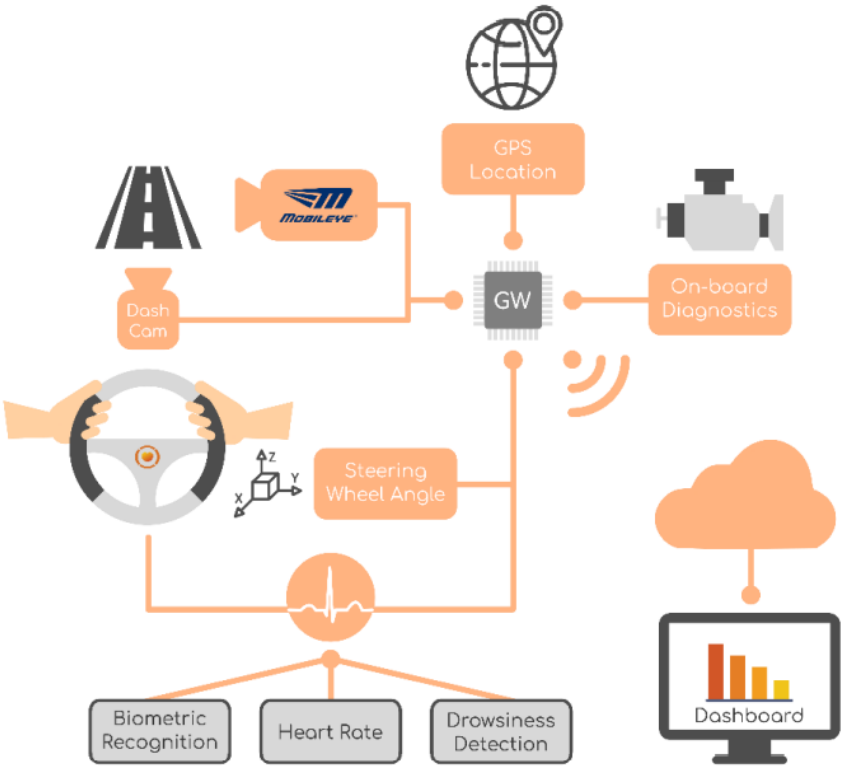


Figure 1.3: Global overview of the i-DREAMS on-vehicle systems

Mobileye component, and its goal is to perform a post-trip analysis of the cause of the event. However, the information obtained from this camera is not used in the context of this project. The gateway system also includes an inertial unit composed of an accelerometer and a gyroscope for driver behavior monitoring. The unit performs a calibration routine to find the correct orientation of the system under the vehicle axis. Accelerometer and gyroscope data are processed to obtain events such as harsh-braking, harsh-acceleration, and harsh-cornering and then fused with the speed obtained via CAN bus. The i-DREAMS project provides access to data through an API. The API defines that each data collection system is associated with a vehicle and not with a specific driver. Data is acquired during trips, which are defined from the moment the vehicle's engine is activated until the moment it is deactivated. However, there is a grace period of 5 minutes during which a quick turn-off and back-on are considered the same trip. Figure 1.4 shows some of the sensors installed in vehicles.

1.2 Motivation

The recent rise of the Internet of Things (IoT) [58] paradigm generates an immense flow of data that can be exploited with Machine Learning (ML) [44] and data mining

(a) Mobileye and dashcam seen from the inside of the vehicle

(b) Mobileye and dashcam seen from the outside of the vehicle

Figure 1.4: i-DREAMS on-vehicle sensors

techniques to automate or support business decision-making. A particular kind of data that has greatly benefited from this evolution is automotive telematics data. Telematics is a disruptive automotive technology that utilizes Information Technology (IT) and communication protocols to send, receive and store information about remote vehicles and their drivers. This type of data can be used in various areas and applications:

- Vehicle maintenance and improvement to reduce costs of overhauling vehicle equipment.
- Safety tracking to monitor vehicle speed and location, as well as harsh driving events and seat belt use.
- Insurance risk assessment, normally used by insurance companies to monitor driver behavior, allowing them to adjust insurance fees accordingly.
- Autonomous driving.

This project explores the value that telematics data (vehicle sensing data) can have to devise a driver profile identification solution and help companies supervise drivers' behavior. This perspective can help generate business opportunities for insurance companies to automate insurance price calculations based on the driver profile, attract customers, and then maximize their profit. Nowadays, companies use Usage-Based-Insurances (UBI) methods that calculate auto insurance premiums. These methods follow a pay-as-you-behave approach where the insurance fee is calculated based on the driving competencies such as over speed, hard acceleration, hard braking, hard cornering, and others [6].

In addition, this project also tries to show that this kind of data could be useful for fleet management companies, where the main objective is to oversee all fleet performance (fleet profile) and fleet maintenance to increase productivity and help a business run as smoothly as possible. For instance, the best drivers can be assigned to most difficult routes and services.

1.3 Objectives

The main goal of this project is to develop a system that takes advantage of machine learning techniques to devise a driver profile identification based on trip data obtained in a non-intrusive way. In order to accomplish this, we use the following approach:

- Devise a generic driver profile classification solution, based on driver behavior data.
- Analysis and processing of i-DREAMS data obtained from driver trips to construct a dataset organized by trips.
- Evaluate the use of unsupervised machine learning techniques over the dataset, to identify a driver profile for each trip.
- Label the dataset to get a ground-truth data, build a classification model with supervised machine learning techniques, and evaluate the model's performance.
- Create a method to establish a driver profile from the trip's profile, that can evolve/change over time.
- Design and development of a Representational State Transfer Architectural Style (REST) API that uses the classification model to profile drivers and fleets.

The data used for this project was provided by the CardioID company [11] in the context of the i-DREAMS project. The input data is obtained from the i-DREAMS devices with the i-DREAMS API. A detailed analysis of each device data is provided in Section 3.1. To access the data we use Hypertext Transfer Protocol (HTTP) request interaction with the i-DREAMS API. The data of each involved system is grouped in tables, and for each table, we extract trip features. Thus, the constructed dataset is organized by trips. Figure 1.5 summarizes the proposed solution to profile trips. This figure shows an example of some data types obtained by interacting with the i-DREAMS API. From that data, we create features for the dataset. Then, we apply feature engineering and

unsupervised learning to determine what the trips mean and assign them the class they belong to. After that, we apply supervised learning, train different models and evaluate them. Finally, we save the model with the best scores. After deploying the

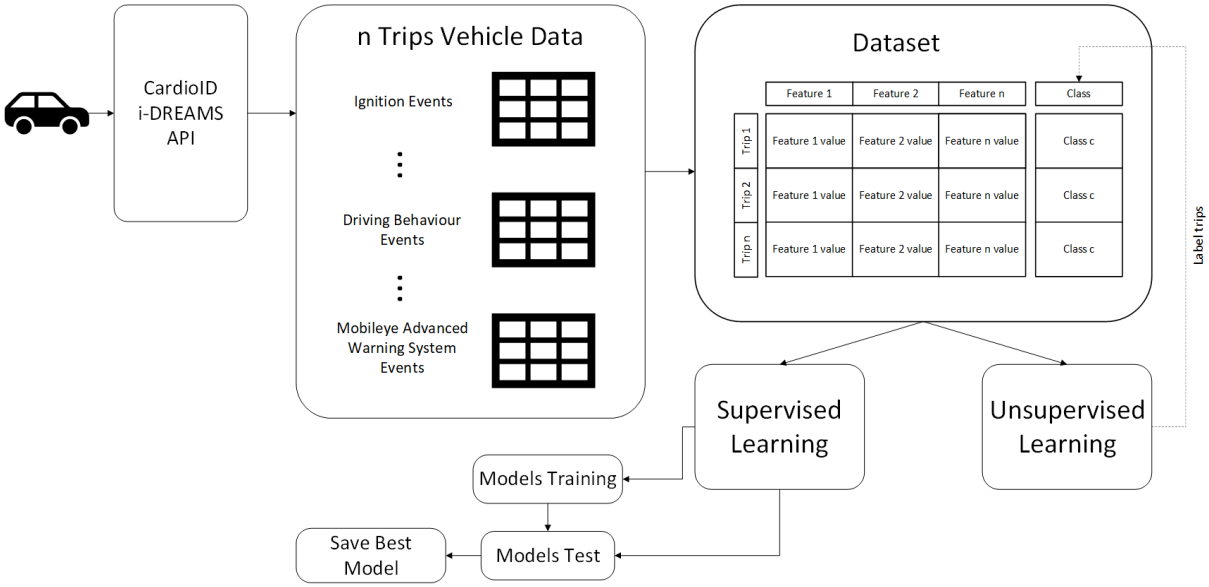


Figure 1.5: System to evaluate and save the best model

best model, we use it to predict the trip profile of each driver with the driver profile developed API. Figure 1.6 shows this process. Initially, an user uploads trips to the API. Then, the best model predicts the trip's profile. Finally, the user can obtain the global driver profile.

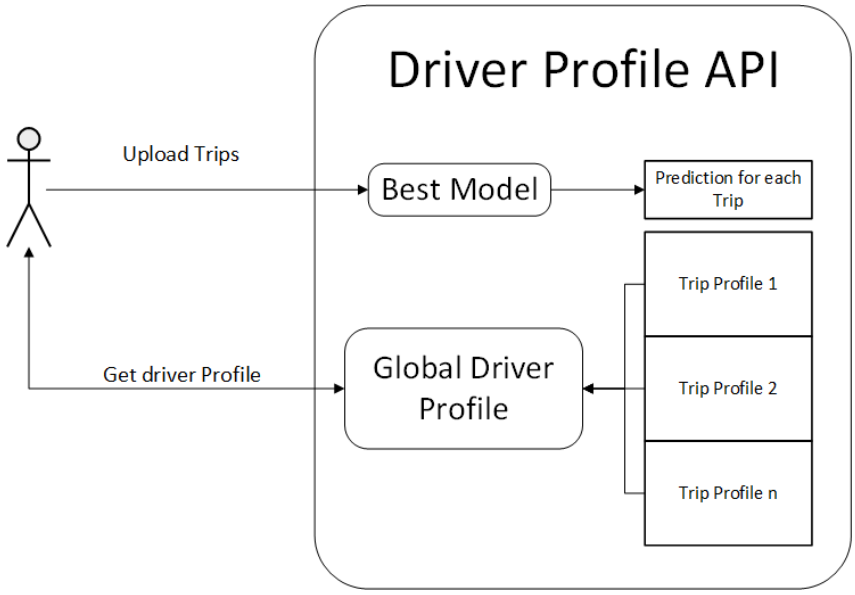


Figure 1.6: Global Driver Profile

1.4 Contributions

As a result of this project, the articles [5] and [3] were published in the RECPAD 2022 Portuguese conference on pattern recognition [54] and the DATA 2023 international conference on data science, technology and applications [17] respectively. Additionally, the article [4] was submitted to the ISEL Academic Journal [32].

1.5 Document Organization

This chapter highlighted not only the context and motivation of this project but the objectives as well. The remainder of this document is organized as follows.

In Chapter 2, we present the state of the art research techniques on driver profiles, parameters used for driver profiling, machine learning methods, developed frameworks and studies according to the literature. Chapter 3 gives a detailed analysis of the data collection sensors, describes the events obtained from these sensors via i-DREAMS API, and reports on how the dataset was constructed. Chapter 4 describes the methodology used to implement a driver profile. In Chapter 5, we present the experimental evaluation of the developed techniques. Chapter 6 presents the conclusions and directions of future work.

In Appendix A, we describe all the features of the constructed dataset. Appendix B provides the data distribution for some features. Finally, Appendix C shows the grids used for parameter tuning.

2

Background and State Of The Art

As mentioned in Section 1.2, the evolution of the IoT paradigm has led to an enormous increase in the use of telematics data in vehicles. With this data, it is possible to develop solutions to determine driver profiles. This chapter aims to explore the concepts and techniques used in the literature on driver profiles.

Section 2.1 introduces the evolution of the different types of driver profiles established in various studies, which according to the literature consists of those that are most often applied for this type of problem. Section 2.2 explores the various parameters used according to the literature in datasets to achieve an adequate and optimized driver profile. Section 2.3 presents the characteristics of some chosen studies to show a systematic analysis of the machine learning methods used, devices, and applications proposed. To conclude, Section 2.4 gives a general explanation of how the most common machine learning methods for driver profile classification problems work.

2.1 Driver Profiles

In the research literature, driving styles are operationalized at different levels of specification, ranging from single indicators, such as speeding or hard acceleration, to very general concepts, for example aggressive driving or risky driving, which may be based on a combination of a variety of more specific behavioral indicators.

As stated in [71], a study concluded that there are eight different driving styles regarding as how an individual usually drives. They also found that specific driver

variables were significantly related to those driving styles. These styles were organized into two main categories: safe driving styles and unsafe driving styles. Table 2.1 presents each style and the category it belongs to. Within the safe style category, there are three styles. The distress-reduction driving style is related to muscle relaxation techniques while driving. The patient driving style is associated to calm and safe behaviors based on the motto "better safe than sorry". The careful driving style is referred to cautious behaviors and a state of mind to always react well and quickly to unexpected actions of other drivers. On the other hand, within the unsafe style category, there are five styles. The dissociative driving style is based on unaware and misjudging behaviors. The anxious driving style is related to nervous and worrying behavior while driving. The risky driving style is associated with drivers that like to take risks while driving. The angry driving style usually means that the driver will make aggressive reactions to other drivers actions. The high-velocity driving style is related to an impulsive attitude towards driving faster than allowed by the present road conditions or feeling impatient when traffic slows down. Recently, it was pro-

Table 2.1: Driving Styles [71]

Safe	**Unsafe**
Distress-reduction	Dissociative
Patient	Anxious
Careful	Risky
-	Angry
-	High-velocity

posed a conceptual framework in which driving styles are seen in terms of driving habits established as a result of individual dispositions, as well as social norms and cultural values [59]. This framework suggests the use of a global driving style which is made up of a set of specific driving styles. A specific driving style refers to a specific behavior commonly used when driving. Table 2.2 presents a connection between driving behavior and various driving parameters [66].

2.2 Parameters for Driver Profiling

A review of the literature reveals some of the parameters adopted by researchers for profiling drivers, such as their vehicle speed, acceleration, braking, mileage, fuel consumption, among others [66]. The studies presented in this section assume that driver behavior was profiled under the naturalistic driving environment and based on actual

crash data obtained from self-reported surveys. Table 2.3 presents a collection of different parameters used in the studies for profiling drivers into different categories based on driving behavior.

All the collected parameters are analyzed and discussed below based on conducted studies. The following subsections describe the key parameters on these studies.

2.2.1 Speed

As referenced in the literature, speed is probably the most used parameter for classifying a driver profile. Excessive speed is one of the main causes of accidents according to a study on car accidents involving pedestrians, motorcycles, and large vehicles [72]. During a crash, the outcome is directly interrelated to the amount of kinetic energy (associated with car movement) generated due to vehicle velocity (V) and mass (M),

$$KE = \frac{1}{2}(M.V^2) \tag{2.1}$$

The amount of kinetic energy is directly proportional to the velocity and mass. The larger the amount of kinetic energy, the more destructive a crash will be. Speed, therefore, directly influences crash and injury severity [22]. This is expected, since the speed of a vehicle directly influences the safety of a driver, by giving them less time to respond to a safety–critical situation and there are smaller margins of error at higher speed. In addition to driver safety, vehicle speed also influences fuel consumption and vehicular emissions.

2.2.2 Acceleration, Braking, and Jerks

On every trip, different maneuvers are performed that can influence the occurrence of accidents. Part of these maneuvers are accelerations and sudden braking. During braking, vehicles are subjected to negative longitudinal velocity events. On the other hand, during accelerations the longitudinal velocity is positive. Aggressive acceleration and deceleration events happen when the driver applies more force than the usual to the pedals.

In 2006, one study examined speed, acceleration, and braking patterns to predict accident situations [35]. The used data was obtained from a GPS system built into vehicles and On-Board Diagnostics (OBD).
The ultimate goal was to identify behaviors and their relationship to crash risk. One of the essential conclusions they reached was that the same behaviors were not observed

in drivers with the same demographic characteristics, but in general, speed, harsh acceleration, and hard braking patterns were associated with vehicle crash involvement. Recently, a study used GPS devices to profile drivers based on their speeding, aggressive acceleration, and braking behavior [21]. Each event was assigned a unique timestamp, a spatial indicator based on time of the day, speed limit, intersection type, weather condition, and school zones. This study, allowed the GPS devices to collect driver speed, and determine aggressive acceleration and braking events. The experimental results were similar to those of the previous study. Aggressive acceleration and braking events are associated with higher accident risk, fuel consumption, and emissions.

2.2.3 Mileage

Annual mileage is one of the parameters that best predict the possibility of having an accident [50]. However, it has been shown that drivers with high mileage have a lower risk of accidents per mile driven. The explanation for this has been suggested by [33], where drivers with a few miles driven mainly drive on busy streets with multiple vehicle categories and roads with two-way traffic. On the other hand, drivers with several miles driven, usually do it on highways with limited access and with separate lanes. Another explanation given for this is that drivers who drive more miles have better driving skills, as opposed to drivers with fewer miles. The higher the mileage, the greater the amount and emission of gases and fuel consumption.

2.2.4 Lateral Maneuver

Lateral maneuvers such as swerving, lane changing, and sharp turns are associated with aggressive driving behavior, which increases the possibility of a crash [66]. Swerving is defined as an abrupt movement maneuver in any direction that causes the wheels to deflect at a high rate. The event can be detected when a driver tries to move away from an obstacle on the road or suddenly detects the risk of an involuntary lane change [26]. Cornering events are characterized as maneuvers made at high speed during a turn, and can lead to car rollovers, skidding, and total loss of vehicle control [38]. The level of aggressiveness of this maneuver can be calculated using the horizontal force (centrifugal force), which acts on the vehicle generating lateral acceleration measured in G's. Several researchers have used lateral vehicle maneuvers as a parameter for creating a driving profile based on accident probability. The study "Driving analytics using smartphones: Algorithms, comparisons, and challenges" [77] concluded that

smartphones accurately detect braking, acceleration, left cornering, and right cornering events. The smartphone sensor-based data collection approach can be considered reliable and accurate, as it can identify specific driving patterns with high accuracy. This study also proposed a threshold value of 0.24g and -0.24g to categorize left and right harsh cornering events, respectively. Similarly to other parameters, lateral maneuvers yield increased fuel consumption and emissions.

2.2.5 Temporal and Spatial Variation

The risk of a car accident can vary depending on temporal factors such as time of day, day of the week, and month. In addition, spatial factors such as traffic flow, road type, and characteristics can also influence crash risk. During the night, the probability of an accident occurring is higher than during the day, since there is less visibility. The study "Driver behavior profiles for road safety analysis" [21] concluded that driver behavior is different during weekdays and weekends. Weekdays are associated with a lower risk of an accident due to the congestion that exists during these days. Contrarily, on weekends the probability of an accident occurring is higher. An interesting observation made by [12] determined that during weekends there were more accidents where only one vehicle was involved, and on Fridays we have accidents with multiple vehicles. When it comes to spatial factors, the study [39] concluded that with an increase in the number of vehicles, the number of interactions between vehicles increases, raising the probability of a crash. The study [51] noted that the probability of a crash decreases with an increase in the number of lanes and road width. Also, higher occupancy in junctions increases the number of harsh accelerations and harsh braking events.

2.2.6 Distraction

It is indisputable that distractions while driving significantly increase the likelihood of an accident. Distracted driving is one of the behaviors that has caused more accidents, and is associated with an aggressive driving style. It has been observed that using a cell phone while driving increases reaction time, meaning that the driver may not react optimally to adverse conditions [15].

The study [78] noted that when the driver is distracted, vehicle speed directly influences the driving quality. However, the study concluded that vehicle speed has a similar effect on driving quality regardless the driver is distracted or not. Other studies, such as [52], have suggested that using a cell phone while driving increases the likelihood of driving errors to occur.

Recently, the study [49] investigated drivers behavior while driving distracted (talking or texting on their cell phones). The conclusion reached was that when cell phone use lasts a short time, it is less likely to affect driving quality. On the other hand, when the distraction time is long, it is more plausible that the driver will perform dangerous and aggressive maneuvers.

Table 2.3: Parameters for driver profiling

Study	**Parameters**								
	Speed	Acceleration		Braking/ Deceleration	Jerk	Sudden lane change	Sudden manoeuver	Cornering	Idling
		Longitudinal	Lateral						
[21]	✓	✓	-	✓	-	-	-	-	-
[16]	✓	-	-	-	-	-	-	-	✓
[79]	✓	✓	✓	-	-	-	-	-	-
[38]	✓	✓	-	✓	-	-	✓	-	-
[73]	✓	✓	✓	✓	-	✓	✓	✓	-
[37]	-	-	-	-	✓	-	-	-	-
[13]	✓	✓	-	✓	-	-	-	-	-
[60]	✓	✓	-	✓	-	✓	✓	-	-
[42]	✓	-	-	-	✓	-	-	-	-
[40]	✓	✓	-	✓	-	-	-	-	-
[61]	✓	✓	✓	-	-	-	-	-	-
[14]	✓	✓	✓	✓	-	-	-	-	-
[19]	-	✓	-	✓	✓	-	-	-	-
[7]	✓	✓	✓	✓	-	✓	✓	-	-
[76]	✓	✓	✓	✓	-	✓	-	-	-

Table 2.3: Parameters for driver profiling (*Continuation*)

Study	**Parameters**								
	Engine Speed	Throttle position	Mileage	Traffic flow	Traffic violation	Spatial Variation	Temporal Variation	Distraction	Time Headway
[21]	-	-	-	-	-	✓	✓	-	-
[16]	-	-	-	-	-	-	-	-	-
[79]	-	-	-	-	-	✓	-	-	-
[38]	-	-	✓	✓	✓	-	-	-	-
[73]	-	-	-	-	-	✓	✓	-	-
[37]	-	-	-	-	-	-	-	-	✓
[13]	-	-	-	-	-	✓	-	-	-
[60]	-	-	-	-	-	-	-	-	-
[42]	✓	✓	-	-	-	✓	-	-	-
[40]	-	-	-	-	-	-	-	✓	-
[61]	-	-	-	✓	-	✓	-	-	-
[14]	-	-	-	-	-	-	-	-	-
[19]	-	-	-	-	-	-	-	-	-
[7]	-	-	-	-	-	-	-	-	✓
[76]	✓	✓	-	-	-	-	✓	-	-

2.3 Methods for Driver Profiling

This section describes the machine learning methods applied, devices used, and the proposed applications for the set of studies presented in Table 2.3 . This information is reported in Table 2.4 [66] which provides a systematic review of the state of the art.

Table 2.4: Methods for driver profiling

Study	Methods	Devices used	Driver behavior profiles	Proposed application
[21]	-	GPS	Scoring	Risk profilling of driver
[16]	SPC tool	GPS	Scoring	Performance appraisal of fleet driver
[79]	K-means clustering	Smartphone sensores	Aggressive, Conservative and Conservative drivers making aggressive turns	Classification of drivers into various groups
[38]	EW-AHP	OBD sensor	Scoring	Usage based Insurance Schemes
[73]	Pattern Matching approach	In-vehicle data recorder	Scoring	Fleet drivers monitoring
[37]	Statistical analysis	In-vehicle data recorder	Aggressive, Normal	Identification of aggressive drivers
[13]	Fuzzy inference system	Smartphone sensors	Normal, Moderate, Aggressive	Usage based Insurance Schemes
[60]	-	Smartphone sensors	Very Safe, Safe, Aggressive, Very aggressive	Identification of blackspot location
[42]	Gamification approach	OBD sensor, Smartphone sensors	Saver, Typical, Careless	Eco-driving
[40]	K-means clustering	Smartphone sensors	Safe, Aggressive, Risky, Distracted, Aggressive/risk, Aggressive/distracted	Identification of aggressive and dangerous driving profiles
[61]	LSTM (Long-Short term memory) Recurrent Neural Network architecture	Smartphone sensors	Normal, Aggressive, Drowsy	Classification of drivers into various groups
[14]	PAM (Partition Around Medoids) cluster analysis	In-vehicle data recorders	-	Identification of driver style
[19]	Fuzzy inference system	GPS, OBD, and Inertial Navigation	Good, Normal, Aggressive	Usage based Insurance schemes
[7]	-	Smartphone sensors	Scoring, Aggressive, Calm, Drowsy	Classification of drivers into various groups
[76]	SVM (Support Vector Machine), K-means clustering	In-vehicle data sensors,	-	Driver Profile Classification

As shown in Table 2.4, cluster analysis is one of the most commonly used methods to solve problems of identification of driving profiles into several groups. In this regard, the [14, 40, 79] studies use clustering to group different objects that are similar.

The first study [79] uses clustering to group drivers into different categories based on data acquired by smartphone sensors. Data was collected for about 300 drivers on speed, location, and acceleration. For each road segment, the standard deviation and mean of speed and acceleration were calculated. Next, k-means clustering [27] was applied to the dataset to identify different types of roads. Then, the authors calculated the average driver deviation from the established normal behavior for each road segment. For each driver and road, the difference between the road mean and the driver's mean on this road is calculated and normalized by standard deviation. This gives a measure of how many standard deviations the driver's mean deviates from the road's mean. The normalized values were combined into a vector, and finally, the clustering algorithm was applied to the vectors to identify groups of drivers.

In the case of the study [40], k-means clustering was performed in two phases to identify dangerous driving profiles. The first phase of clustering was intended to organize drivers into two groups, aggressive and non-aggressive, according to the number of harsh acceleration and deceleration events. The second phase aimed to detect unsafe driving behaviors such as speeding events and cell phone use for both previous driving profiles (aggressive and non-aggressive).

The last of the three studies, [14] uses Partition Around Medoids (PAM) to organize drivers into different types of profiles according to their driving style. Based on PAM cluster analysis, the study identified three driving styles with four parameters, speed, lateral acceleration, longitudinal acceleration, and braking. The first profile identifies drivers who perform maneuvers at high speed and low acceleration (longitudinal and lateral) and deceleration. The second profile identifies drivers who execute maneuvers at high speed but with moderate acceleration (longitudinal and lateral) and deceleration. Finally, the last profile identifies drivers who execute maneuvers at low speed but with high acceleration (longitudinal and lateral) and deceleration.

As technology and machine learning methods advance, neural network techniques have recently been applied in various applications, as is in [61]. This study proposed a methodology based on stacked Recurrent Neural Networks (RNN), on which nine different smartphone sensors were able to capture data from naturalistic driving sessions. The proposal was formulated as a time-series classification problem of driving behaviors. Drivers were classified into three different categories: normal, aggressive,

and drowsy. To support time-series classification, the proposed model used a RNN architecture called Long Short-Term Memory (LSTM) network. The model was tested on the naturalistic driving behavior classification dataset, "UAH-DriveSet" [74]. The results obtained when compared to other state-of-the-art methods revealed an increase of about 10% in the F1-measure metric. It was important to explore studies using these neural network techniques to give acknowledgment that more and more use of these techniques is currently being made. However, for this thesis, we chose not to use these methods as they prove to be difficult to understand and explain how the neural network works. They have high predictive power but low interoperability (act as a black box). It is essential for driver profile classification problems to understand how the algorithms work to better understand the results. Another reason for us to not use neural networks is that it requires much more data than traditional machine learning algorithms, making the neural network computationally expensive.

According to the literature, Support Vector Machine (SVM) [2, 62] is another method used for driving profile classification problems. The study [76] uses inertial sensors placed inside the vehicle used to obtain data. The acceleration, turning, and braking data, were used to extract feature vectors. Feature vectors were acquired by histogramming the time-series signals (for example, the accelerometer) into five bins [61]. Other features such as the *mean*, *variance*, *min*, and *max* were also included in the feature vectors. These vectors were then used in the SVM algorithm to obtain the driver's classification. The results achieved show that features associated with acceleration events were not the best at differentiating drivers. On the other hand, features related with braking and turning events showed great potential to differentiate drivers. This study also applied k-means clustering to the same dataset, but the results were somewhat inferior to SVM in terms of accuracy.

2.4 Machine Learning Techniques

This section seeks to briefly introduce the most commonly used algorithms for driver profile classification and also the ones that we used for this thesis. Here we described, some advantages and disadvantages of the algorithms and how they work. We also describe some of the evaluation metrics used.

2.4.1 Unsupervised Algorithms

2.4.1.1 K-Means

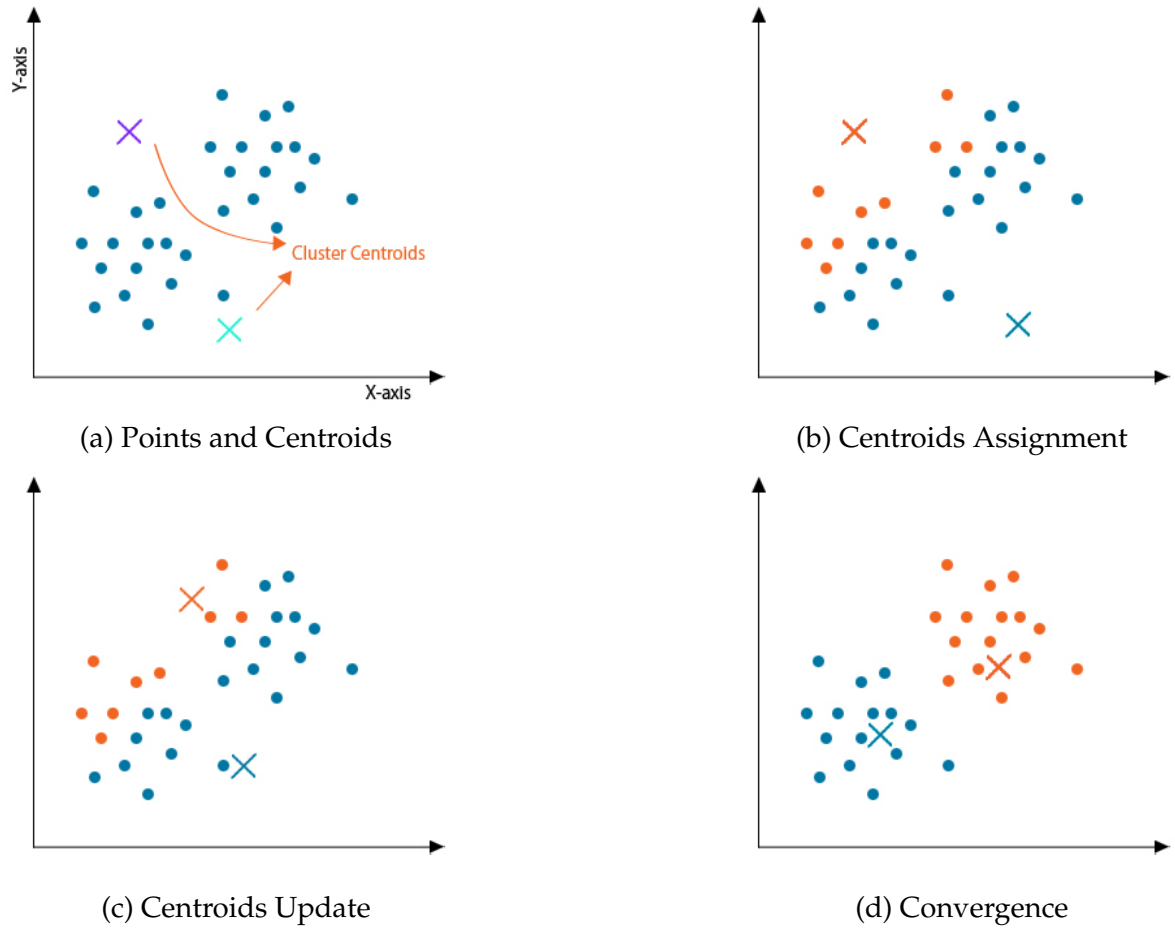
The k-means algorithm [27] aims to group data in such a way that data belonging to the same group (cluster) is more similar to each other (in some way) than to data belonging to other groups. It is an unsupervised learning algorithm because it is in the category of clustering methods. This means that this algorithm does not assume the existence of labeled data. The motivations for using this method are the following:

- It is one of the main ways of exploring and representing structures found in data.
- Clusters contain meaningful information.
- It is simple to implement and to understand how it works.
- It is easy to scale by implementing parallel k-means.
- Efficient (and higher-level) description of the data.
- It provides easy detection of outliers.

In terms of functionality, the general algorithm takes into account the following steps:

1. Firstly, it is necessary to define the value of "k", that is, the number of clusters.
2. Randomly define a centroid for each cluster.
3. For each data point, calculate the centroid of the shortest distance. Each point will belong to the nearest centroid.
4. Relocate the centroid. The new centroid position should be the average of the position of all the points in the cluster.
5. The last two steps are repeated, iteratively, until we get the optimal position (until the position does not change) of the centroids.

Figure 2.1 illustrates how the algorithm works. Even though the algorithm can be parallelized, it is possible to increase its performance (reduce the number of iterations) by improving the initialization process. It is also important to note that different initializations yield different results. On the other hand, defining a "k" value is also a delicate task. The "k" value is a hyperparameter for the algorithm, which is required to tell the



(a) Points and Centroids

(b) Centroids Assignment

(c) Centroids Update

(d) Convergence

Figure 2.1: K-means Algorithm with two centroids [36]

system how many clusters will be created. There are, however, some methods to find the best or optimal "k" value.

Although these techniques improve the algorithm's performance, its solutions tend to be locally optimal, which may be far from the global optimum.

2.4.1.2 DBSCAN

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [24] is an unsupervised algorithm that is able to find arbitrary shaped clusters and clusters with noise. The idea behind DBSCAN is that a point belongs to a cluster if it is close to many points from that cluster. The two main parameters used for this algorithm are: eps and minPts. The eps parameter indicates the distance that specifies the neighborhoods. Two points are neighbors if the distance between them is less than or equal to the eps value. The minPts parameter specifies the minimum number of data points to define a cluster. Thus, data points can be classified as:

- Core Point - If there are at least minPts number of points (including the point itself) in its surrounding area with radius eps.
- Border Point - If it is reachable from a core point and there are less than minPts number of points within its surrounding area.
- Outliers - If it is not a core point and not reachable from any core points.

A cluster is composed by core points neighbors and all the border points of these core points.

The main pros of using DBSCAN are:

- Does not need to specify the number of clusters.
- Performs well with arbitrary shapes of data points distribution.
- Is able to detect outliers.

The cons are that sometimes is very difficult to determine the best value for the eps parameter.

2.4.1.3 Gaussian Mixtures

Gaussian Mixture Models (GMM) [57] assume that there are a certain number of Gaussian distributions, and each of these distributions represents a cluster. Contrary to K-Means, GMM perform soft classification, meaning that they provide the probabilities that a given data point belongs to each of the possible clusters. They work on an algorithm called Expectation-Maximization (EM) [45], that tries to estimate the parameters of the Gaussian distributions in two steps:

1. **E-step** - Data points are assigned to a Gaussian cluster based on probabilities that they belong to that cluster.
2. **M-step** - The mean, covariance, and density are calculated for clusters based on the data points in the E-step.

After these steps, the algorithm repeats the process until convergence is reached.

Some pros of using GMM are that they can handle a greater variety of shapes when compared to K-Means that is only good at spherical shapes. Another positive aspect is the use of soft classification. On the other hand, a con is that they might take longer to run because they have slower convergence.

2.4.1.4 Evidence Accumulation Clustering with K-Means

K-Means is one of the most used algorithms for clustering, because it is simple to use and understand. With an ensemble approach, it tries to find better and more robust clustering solutions. The objective of Evidence Accumulation Clustering (EAC) [47] is to combine the different results of the ensemble into a single data partition, by viewing each clustering result as an independent evidence of data organization. A K-Means clustering algorithm, l, by organizing the n patterns into clusters according to the partition P^l, expresses relationships between objects in the same cluster; these are mapped into a binary $n \times n$ co-association matrix, $C^l(i, j)$, where non-zero pairwise relations, $C^l(i, j) = 1$, express co-existence of patterns i and j in the same cluster of P^l [47]. In order to identify natural clusters (retrieve the combined data partition), hierarchical agglomerative algorithms are used.

2.4.2 Supervised Algorithms

2.4.2.1 Support Vector Machines

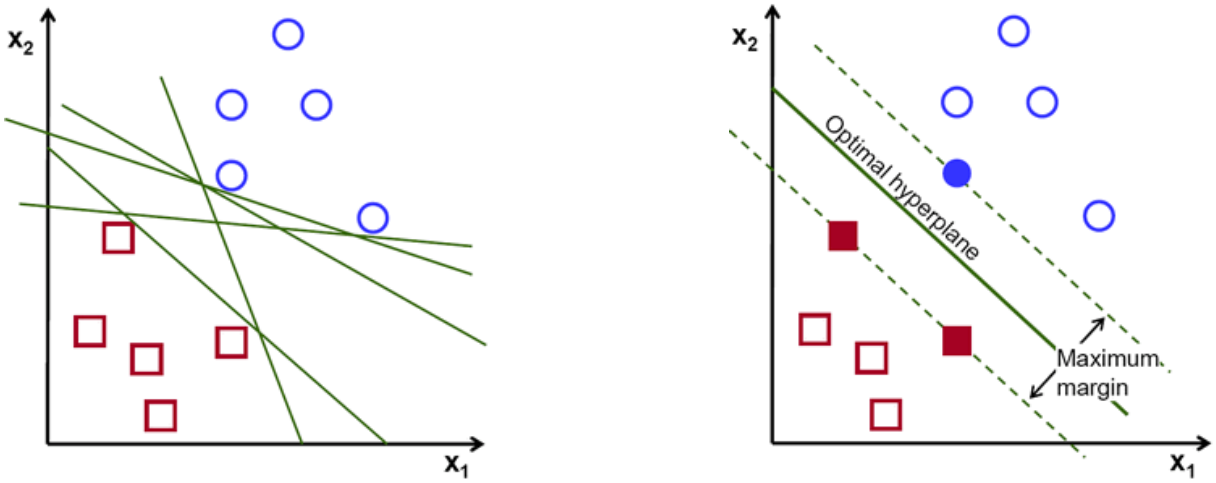
Support Vector Machines (SVM) [2, 62] are supervised machine learning algorithms applicable to classification and regression problems. They were initially developed for binary classification problems. These algorithms build a set of support vectors that separate classes with hyperplanes, which can take linear (two features) and planar (three features) forms. It becomes difficult to imagine when the number of features exceeds three.

One of the main goals of this algorithm is to maximize the margin of separation between existing classes, i.e., to find the hyperplane that maximizes the distance between two points of distinct classes. The points closest to the separation margin correspond to the support vectors, which influence the position and orientation of the hyperplane. In other words, the support vectors define the margin of separation of the classes and are the most difficult examples to classify.

Figure 2.2 shows an example of how the optimal hyperplane is found.

2.4.2.2 Decision Trees

Decision Trees (DT) [2, 62] are a supervised machine learning algorithm used for classification and regression problems. They have a hierarchical tree structure, which consists of a root node, branches, internal nodes and leaf nodes. The most relevant attribute is placed at the top of the tree (root node). Then through a decision rule over



(a) Possible Hyperplanes (b) Optimal Hyperplane

Figure 2.2: SVM Example [70]

that attribute the root node is splited in two more nodes that can be an internal node or leaf node (end node with the predicted class value). These steps are repeated until there are leaf nodes in all the branches of the tree.

The main advantages of DT are:

- Simple to understand and to interpret.
- Does not require data normalization.
- Can handle both numeric and categorical variables.
- Handles multi-class classification problems.
- Useful to attribute meaning to clusters.

Some of the disadvantages are:

- If the tree is too complex, the learned model is not well generalized, leading to overfitting problems.
- Trees can be unstable due to small variations in the data, resulting in different trees.
- Decision tree learners create biased trees if some classes dominate.

2.4.2.3 Random Forest

Random Forest (RF) [2, 62] is a supervised learning algorithm used for classification and is an upgrade over DT. It consists of a large number of individual DT that work as an ensemble. Each tree in the ensemble predicts a class, and the class with the majority of the votes becomes the model's final prediction. When splitting each node during the construction of a tree, the best split is found either from all input features or a random subset. The purpose of this randomness is to decrease the variance of the estimator and to prevent overfitting problems that are related to single DT.

2.4.2.4 XGBoost

XGBoost [2, 62] stands for eXtreme Gradient Boosting and is a supervised learning algorithm for classification and regression problems. It is based on decision tree Classification and Regression Trees (CART) algorithms. As opposed to RF, it uses a type of ensemble learning called boosting, which uses the previous model's result as an input to the next model. Instead of training models separately, boosting trains models sequentially, with each new model being trained to correct the errors of the previous ones. At last, the term gradient boosting refers to a boosting method where the error is minimized using a gradient descent algorithm. In other words, gradient descent is an iterative optimization algorithm to minimize a loss function, typically the Mean Squared Error (MSE).

2.4.3 Evaluation Metrics

This section describes evaluation metrics used for unsupervised and supervised learning.

2.4.3.1 Unsupervised Metrics

Firstly, we have the Silhouette Coefficient [65]. This metric uses the model itself to obtain the score, and the higher the score, the better the clusters are defined. The score is bounded between -1 for incorrect clustering and +1 for highly dense clustering. Scores around zero indicate overlapping clusters. The Silhouette Coefficient s_{sc} for a single sample is given as:

$$s_{sc} = \frac{b - a}{max(a, b)}, \tag{2.2}$$

where a is the mean distance between a sample and all other points in the same cluster and b the mean distance between a sample and all other points in the next nearest cluster.

Another unsupervised metric is the Calinski-Harabasz (CH) [10] Index also known as the Variance Ratio Criterion. The score is higher when clusters are dense and well separated, which relates to a standard concept of a cluster. The score is the ratio of the sum of between-clusters dispersion and of within-cluster dispersion for all clusters, where dispersion is defined as the sum of distances squared. For a set of data E of size n_E which has been clustered into k clusters, the CH score s_{CH} is defined as:

$$s_{CH} = \frac{tr(B_k)}{tr(W_k)} \times \frac{n_E - k}{k - 1}, \tag{2.3}$$

where $tr(B_k)$ is the trace of the between group dispersion matrix and $tr(W_k)$ is the trace of the within-cluster dispersion matrix defined by:

$$B_k = \sum_{q=1}^{k} n_q (c_q - c_E)(c_q - c_E)^T \tag{2.4}$$

$$W_k = \sum_{q=1}^{k} \sum_{x \in C_q} (x - c_q)(x - c_q)^T, \tag{2.5}$$

where C_q is the set of points in cluster q, c_q the center of cluster q, c_E the center of E, and n_q the number of points in cluster q.

Another evaluation metric used for unsupervised learning, is the Davies-Bouldin (DB) index [18], where a lower value relates to a model with better separation between the clusters, being zero the lowest possible score. The score is defined as the average similarity between each cluster C_i for $i = 1, ..., k$ and its most similar one C_j. The similarity is defined as a measure R_{ij}:

$$R_{ij} = \frac{s_i + s_j}{d_{ij}}, \tag{2.6}$$

where s_i is the average distance between each point of the cluster and the centroid of that cluster – also known as cluster diameter, and d_{ij} the distance between cluster centroids i and j. The final DB index, DB, is defined as:

$$DB = \frac{1}{k} \sum_{i=1}^{k} \max_{i \neq j} R_{ij}. \tag{2.7}$$

2.4.3.2 Supervised Metrics

A confusion matrix [62], manages to outline for each class the model's performance. Figure 2.3 shows a confusion matrix for a binary problem (two classes). For simplicity of explanation we assume two driver profiles: Non-Agressive (negative) and Agressive (positive).

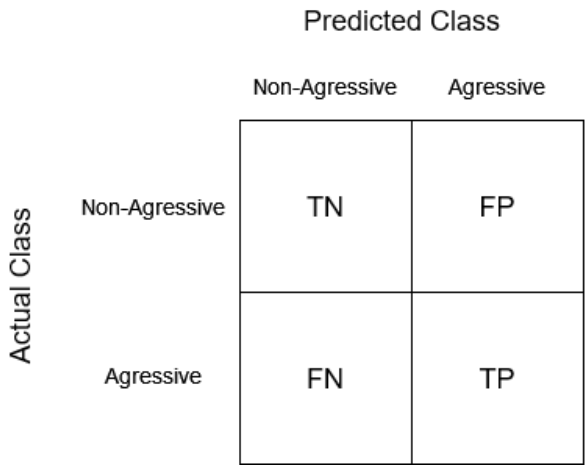


Figure 2.3: Confusion Matrix example

From the confusion matrix it is possible to compute the model's accuracy, which corresponds to the ratio of correctly classified points (prediction) to the total number of predictions, given by

$$Accuracy = \frac{TN + TP}{TN + TP + FN + FP}. \tag{2.8}$$

On the other hand, the model's error is the ratio of incorrectly classified points (prediction) to the total number of predictions,

$$ErrorRate = 1 - accuracy. \tag{2.9}$$

The False Positive Rate (FPR) corresponds to the portion of actual negative instances which are incorrectly identified as positive,

$$FPR = \frac{FP}{FP + TN}. \tag{2.10}$$

The False Negative Rate (FNR) corresponds to the portion of actual positive instances which are incorrectly identified as negative,

$$FNR = \frac{FN}{FN + TP}. \tag{2.11}$$

The precision metric corresponds to the portion of points classified as positive well classified or vice versa,

$$Precision_{Non-Agressive} = \frac{TN}{TN + FN}, \tag{2.12}$$

$$Precision_{Agressive} = \frac{TP}{TP + FP}. \tag{2.13}$$

The recall metric corresponds to the portion of positives well classified or vice versa,

$$Recall_{Non-Agressive} = \frac{TN}{TN + FP}, \tag{2.14}$$

$$Recall_{Agressive} = \frac{TP}{TP + FN}. \tag{2.15}$$

Only valid classifiers achieve high performance in both metrics. To take this into account both precision and recall are combined in a single metric that corresponds to their harmonic mean, called F-Score or F1 score.

$$F1_{Non-Agressive} = 2 \times \frac{Precision_{Non-Agressive} \times Recall_{Non-Agressive}}{Precision_{Non-Agressive} + Recall_{Non-Agressive}}, \tag{2.16}$$

$$F1_{Agressive} = 2 \times \frac{Precision_{Agressive} \times Recall_{Agressive}}{Precision_{Agressive} + Recall_{Agressive}}. \tag{2.17}$$

To conclude, we also took in consideration Receiver Operating Characteristics (ROC) curves, computing the Area Under the Curve (AUC), and Precision vs Recall curves, which are graphs that are meant to illustrate and compare the performance of one or more classifiers.

Table 2.2: Connection between different driving styles and driving parameters

Global driving style	Specific driving style	Driving parameters	Reference
Aggressive	Over-speeding	Speed	[79]
	Rapid acceleration & decelaration	Longitudinal acceleration	
	Sharp turning	Lateral acceleration	
	Jerky driving	Jerks	[37]
	Tailgating	Time headway	
	Over-speeding	Speed	[60]
	Sudden braking & acceleration	Longitudinal acceleration	
	Sudden lane change Sudden turn	Lateral acceleration	
	Harsh acceleration	Acceleration	[40]
	Hash braking	Deceleration	
	Smoothness	Acceleration and speed distribution characteristics	
	Jerky driving	Jerks	[19]
	Sharp acceleration Sharp decelaration	Longitudinal acceleration	
	Hard acceleration & braking	Longitudinal acceleration	[7]
	Aggressive steering	Speed	
		Time headway	
	Weaving		
	Drifting		
	Over-speeding		
	Short car following		
Distracted	Mobile phone usage	% of mobile phone usage while driving	[40]
Risky	Speeding	Speed	[21, 40]
	Harsh braking	Longitudinal acceleration	
	Harsh acceleration		
Conservative/Calm	Driving Slower	Speed	[79]
	Maintaining greater Headway	Headway	
	Avoid sharp turning	Lateral acceleration	
Safe	Driving below speed limit	Speed	[40]
	Less usage of mobile phone	% of mobile phone usage	
Safer (based on fuel consumption)	Less jerky driving	Speed	[42]
	Driving below speed limit	Jerks	
	Pressing and releasing accelerator pedal gently	Throttle position	
	Ideal gear shifting based on engine RPM	RPM	

3

Events and Data

This chapter aims to give an in-depth analysis and explanation of the acquired data. We also describe the transformations required to create the set of features and construct the dataset.

Section 3.1 deals with the whole process of analysis of the raw data obtained through the i-DREAMS API. In Section 3.2, we transform the data to generate a set of features representing the dataset organized by trips.

3.1 Raw Data Analysis

As mentioned in section 1.1, the i-DREAMS project consists of several systems. Each of these systems gathers different types of data that will be used to classify a trip profile. The gateway is the central component of this project that collects the data from different devices and algorithms (STZ) and makes them available through an API. Table 3.1 describes the types of data that these systems can provide when interacting with the i-DREAMS API.

The data initially obtained does not necessarily come in the optimal format for solving machine learning problems and, in this case determining the driver profile. That's why this section describes the data obtained when interacting with the i-Dreams API and then construct the dataset organized by trips.

While driving a vehicle, several events that retain information about the trip in an

Table 3.1: Systems, devices, and data description

System/Device	Data Identifier	Description
Gateway	Ignition	Ignition on/off events
CardioWheel	LOD_Event	Hands on wheel detection events
Drowsiness Detector	Drowsiness	Drowsiness detection event (KSS level)
Gateway Motion Sensor	DrivingEvents	Harsh driving detection events
Mobileye	ME_AWS	Decoded Mobileye AWS message
Mobileye	ME_TSR	Decoded Mobileye TSR message
Mobileye	ME_Car	Decoded Mobileye Car message
GNSS	GPS	GPS location message
Driver App	Distraction	Use of mobile phone events
STZ	iDreams_Fatigue	Driver fatigue intervention warning
STZ	iDreams_Headway	Headway monitoring intervention warning
STZ	iDreams_Overtaking	Illegal overtaking intervention warning
STZ	iDreams_Speeding	Speeding intervention warning

instant of time and a location obtained through satellite-based geolocation data can be generated. The set of possible events for each system in Table 3.1, is described in the following sections.

3.1.1 Hands-On Detection Events

The CardioWheel system produces Hands-On Detection (LOD_Event) events that indicate the detection of the driver's hands on the steering wheel. Each event consists of a timestamp and the Hands-On state. This state has four possible values:

- 0 - No hands on the steering wheel.
- 1 - Left hand on the steering wheel.
- 2 - Right hand on the steering wheel.
- 3 - Both hands on the steering wheel.

3.1.2 Drowsiness Events

The Drowsiness Detector system is able to detect events of the driver drowsiness level based on the Karolinska Sleepiness Scale (KSS) in a certain timestamp. According to KSS, there are 10 levels of drowsiness:

1. Extreme alert.
2. Very alert.
3. Alert.
4. Rather alert.
5. Neither alert nor sleepy.
6. Some signs of sleepiness.
7. Sleepy, but no effort to keep awake.
8. Sleepy, but some effort to keep awake.
9. Very sleepy, great effort to keep awake, fighting sleep.
10. Extremely sleepy, can´t keep awake.

The level of drowsiness is a useful indicator to determine the responsibility and state of the driver.

3.1.3 Driving Behavior Events

The Gateway Motion Sensor system generates driving behavior (DrivingEvents) data, which indicates the occurrence of harsh acceleration, harsh braking, and harsh cornering behaviors. Each event stores the timestamp and the type of event (harsh acceleration, braking, and cornering). At the same time, the maximum acceleration (in g-force) during an event is registered with the total duration of the event in seconds, and the event severity level is measured with one of 3 values:

- L - Low.
- M - Medium.
- H - High.

This indicator is useful to calculate the number of each type of event that occurred during a trip. As stated in Section 2.2, these types of events are widely used in the literature for driver profile classification applications and are mostly related with aggressive driving.

3.1.4 Mobileye Advanced Warning System

The Mobileye Advanced Warning System (ME_AWS) gives information about the safety and warning state of the Mobileye system in a certain timestamp. A large part of the information provided refers to the system usage status. For this reason, only the relevant parameters are exposed below:

- fcw - If True, forward collision warning is active.
- hw_level - Headway monitoring level (0 – no car detected, 1 – green car, 2 – red car, warning).
- ldw - If True, lane departure warning is active (left or right).
- ldw_left - If True, left lane departure warning is active.
- ldw_right - If True, right lane departure warning is active.
- pcw - If True, pedestrian collision warning is active.
- pedestrian_dz - If True, pedestrian detected in danger zone.
- time_indicator - Indicates lighting conditions: day, dusk, or night.
- tsr_level - Traffic sign recognition level. Level of warning according to units over speed limit in (kph or mph).
- zero_speed - If True, vehicle is stopped.

The *fcw* parameter corresponds to Forward Collision Warning (FCW) that activates when the system detects an imminent collision danger with cars ahead and will trigger visible and audible warnings. The *hw* related parameters correspond to Headway Monitoring & Warning (HMW) that activates when the vehicle is too close to the vehicle ahead and will trigger visible and audible warnings. The *ldw* related parameters correspond to Lane Departure Warning (LDW) that activates when the vehicle departs from the current lane without turning signals and will trigger visible and audible warnings.

3.1.5 Mobileye Car Information

The Mobileye system also provides information about the car parameters (ME_Car). Each data sample provides the current speed of the vehicle (kilometers per hour) and a set of parameters described below:

- High_beam - If the high beam is on or off.
- Low_beam - If the low beam is on or off.
- Wipers - If wipers are on or off.
- Signal_right - If right turn signal is on or off.
- Signal_left - If left turn signal is on or off.
- Brakes - If brakes are on or off.

3.1.6 Global Navigation Satellite System

The Global Navigation Satellite System (GNSS) provides satellite-based geolocation data, with a rate of about one sample per second.

3.1.7 Ignition

The Gateway system is able to provide events that tell if the ignition is on or off.

3.1.8 Distraction Events

The i-DREAMS driver app provides real-time mobile phone use events, which are an indicator of distraction. Each sample is mapped by the type of event, mobile usage start, or mobile usage end. These events are used to map any distractions while driving.

3.1.9 Fatigue Intervention

The STZ system gives real-time fatigue intervention (iDreams_Fatigue) events with different levels. The levels are defined as follows:

- Level 0 - No warning (normal driving).
- Level 1 - Visual warning (dangerous driving).
- Level 2 - Visual and auditory warning (dangerous driving).
- Level 3 - Frequent warnings (avoidable accident).

3.1.10 Headway Intervention

The STZ system generates real-time headway intervention (iDreams_Headway) events with different levels. The levels are defined as follows:

- Level -1 - No vehicle detected (normal driving).
- Level 0 - Vehicle detected, but headway >= 2.5 (normal driving).
- Level 1 - Vehicle detected, headway < 2.5, but above warning threshold (normal driving).
- Level 2 - First warning stage (dangerous driving).
- Level 3 - Second warning stage (avoidable accident).

3.1.11 Overtaking Intervention

The STZ system produces real-time overtaking intervention (iDreams_Overtaking) events with different levels. The levels are defined as follows:

- Level 0 - No warning (normal driving).
- Level 1 - Visual warning (normal driving).
- Level 2 - Visual and auditory warning (dangerous driving).
- Level 3 - Frequent warnings (avoidable accident).

3.1.12 Speeding Intervention

The STZ system produces real-time speeding intervention (iDreams_Speeding) events with different levels. The levels are defined as follows:

- Level 0 - No warning (normal driving).
- Level 1 - Visual indication (normal driving).
- Level 2 - Visual speeding warning (dangerous driving).
- Level 3 - Visual and auditory warning (avoidable accident).

3.2 Dataset Construction

After the complete analysis of the events provided, the next step was to build a dataset through these events. To do this, for each possible event we designed several features, which based on Table 2.2, are the most appropriate to determine the style of driving on a given trip. One important aspect to take into consideration is that the events provided vary from trip to trip. This is because we are dealing with a naturalistic driving environment, which means that cars are equipped with devices that, over a longer period, continuously monitor various aspects of their driving behavior, the vehicle, and the surroundings in an unobtrusive way and without the presence of a test supervisor [75]. This includes aspects of vehicle movement, driver behavior, and the direct environment.

As a result, there may be sensor failures on a given trip, resulting in some features, such as the number of harsh accelerations, not being detected, or having wrong values. Therefore, the built dataset may contain missing values such that are processed in a second phase.

Although some events may not have assigned values, all trips are represented by a mandatory set of features as follows:

- *Trip_start* - Date and time trip started. This feature is only relevant to input missing values for other features such as the trip lighting condition.
- *Trip_end* - Date and time trip ended. This feature is only relevant to input missing values for other features such as the trip lighting condition.
- *Distance* - Trip distance in kilometers.

- *Duration* - Trip duration in seconds.

The start and end features are not used in the supervised and unsupervised learning phases. The distance and duration features are only applied in the normalization tasks.

As a first step, all the available data was collected, and the features were created. After a deeper analysis of the data, only the most significant features were selected according to the literature reviewed and the results obtained in the following feature extraction phase. Through interaction with the i-DREAMS API, it was possible to obtain trips from April 1, 2021, to July 20, 2022, which were 17138 trips in total.

The resulting features are organized according to the type of system that originated them. According to the four mandatory features and the tables presented in Appendix A, for each of the 17138 trips, 75 features should be generated. However, for this collection of trips, the Hands-On and Drowsiness Systems were not available, so only 67 features were produced. At the end of this phase the dataset was composed of $n = 17138$ trips and $d = 67$ features.

4

Driver Profile Implementation

This chapter aims to describe all the steps involved in profiling a driver with the previously constructed dataset.

Section 4.1 describes the approach to filter noisy data and how to deal with missing values. Section 4.2 details how we normalized the data. In Section 4.3, we present all the techniques used to reduce the dataset dimensionality. Both Section 4.4 and 4.5 aim to describe all the steps involved in the modeling phase where a semi-supervised approach is applied. In the first, we discuss the approach taken on clustering with all the unsupervised techniques presented in Section 2.4. The second involves all the processes of applying the supervised algorithms and metrics discussed in Section 2.4 to the clusters found in the unsupervised stage. In Section 4.6, we define a way to establish a driver profile from the classified trips. Section 4.7 presents the structure of the developed API. Finally, Section 4.8 provides the thesis repositories with the code.

4.1 Data Pre-processing and Missing Values

The data pre-processing phase was crucial because it involved removing certain trips that acted as noise and did not provide sufficient information for the data. Most of the time these kinds of trips, were a result of a malfunction of the i-DREAMS devices or trips that were very short in distance or duration. Therefore, we applied the following steps to remove these trips:

1. Remove trips with all values missing.
2. Remove trips with all values missing, except the four mandatory features.
3. Remove trips with less than one minute.
4. Remove trips with less than 1.5 km traveled.

After this, the number of trips was reduced from 17138 to 15002, and the imputation of missing values was performed.

As mentioned in Section 3.2 the dataset may include missing values. Therefore, it was necessary to find a way to deal with them because some algorithms cannot work with missing data, and filling them in will impact the quality of the data.

Initially, we needed to check the number of missing values for each feature except the four mandatory ones. In Appendix A, Table A.13 provides the number of missing values for each feature and their percentage. We can see that only the distraction_time and n_distractions features have a large percentage of missing values. After this analysis, it was necessary to find a strategy to fill in the missing values. In this regard, we decided to fill the values with the mean, median, or mode if the percentage of missing values was less or equal to 10%. If the percentage is greater or equal to 50%, the feature is removed. With this strategy only the distraction_time and n_distractions features were removed, since they have too many missing values, leaving the dataset with 65 features.

The next step was to decide which imputation technique to use to assign the most accurate value for the missing values. The goal was to find out which one of the techniques is a better measure of the central tendency of data and use that value for replacing missing values appropriately. If the data is skewed, there are several or large numbers of data points that act as outliers, and because of that, the mean is not appropriate. On the other hand, when the data is skewed, it is adequate to consider the median or mode value to replace the missing values. The figures provided in Appendix B show that for the majority of the features, there are outliers and for that reason, the median was used to replace missing values.

Although the median strategy was useful for most features, for the light_mode we created an algorithm to impute the missing values using the trip start date (trip_start), end date (trip_end), and the geolocation coordinates features. The algorithm we developed calculates whether the trip took place, predominantly during the day, dusk or night.

4.2 Feature Normalization

Feature normalization is a very important task often applied as part of data preparation for machine learning. The idea of normalization is to change the values of numeric columns in the dataset to use a common scale, without distorting differences in the ranges of values or losing information. Normalization is also essential for many machine learning algorithms, especially distance-based ones like clustering algorithms. We should also normalize features if you need to apply any dimensionality reduction techniques such as Principal Component Analysis (PCA) [34] because it seeks to maximize the variance for each component.

The other aspect that emphasizes the importance of normalization is that we can compare trips by their feature values. For example, if we normalize each trip by its total distance, we can now compare each trip by the number of harsh braking events per km traveled.

The majority of the reviewed literature takes into account two approaches to solve the normalization problem. Studies such as [7, 40, 60] normalize the parameters/attributes of their dataset by the total distance traveled. On the other hand, studies such as [73] normalize by the trip's duration.

The trip's distance is used more often as a normalization metric, since the duration might not be the best choice to normalize. During a trip, the vehicle can stop multiple times, which is not taken into account in the final duration value. Although everything suggests that distance is the best metric out of the two, we opted to evaluate both metrics in the modeling phase and choose the one that gives the best results.

4.3 Dimensionality Reduction

One of the purposes of using dimensionality reduction is to reduce the number of features that the computer must process to perform its function (fewer features lead to less computation time) while trying to preserve most of the relevant information in the data, and avoid the curse of dimensionality [8]. Another reason why dimensionality reduction is useful is that it helps to visualize the data when reduced to very low dimensions such as 2D or 3D. Dimensionality reduction also helps to avoid the curse of dimensionality by removing irrelevant and redundant features from the data.

The are two approaches for dimensionality reduction:

- Feature Selection (FS) - Select/choose adequate subsets of the existing features.

- Feature Reduction (FR) - Select/generate/extract a new smaller subset of features, computed from the original features.

We decided to choose FR over FS techniques because most of the features are numeric and also because FR techniques are better for visualization and validation of the clustering algorithms used. Another reason is that FR is more adequate than FS, when all the features seem to exhibit the same predictive power. Since the dataset does not have pre-labeled data we had to choose only unsupervised FR techniques, such as PCA and Singular Value Decomposition (SVD) [69].

Although the FR process seemed to be sufficient, we decided to apply FS before the normalization and FR to remove some features that don't relate to the objective of this thesis (find driver behavior to determine a driver profile). For that reason, the features light_mode, zero_speed_time, n_zero_speed, n_ignition_on, n_ignition_off, n_high_beam, n_low_beam, n_wipers, n_signal_right, n_signal_left were removed since they don't provide any meaningful information for driver behavior, leaving the dataset with 53 features. This decision, was based on the analysis of previous clustering results, where features such as the light_mode revealed to be the most important and kept most of the data variance. The correlation matrix before and after reduction is presented in Section 5.2, to prove that the dimensionality reduction phase produces better results.

4.4 Clustering Analysis

The unsupervised phase combined all the algorithms previously described and tested them with different normalization and dimensionality reduction techniques. Initially, we apply a first stage clustering and then a second stage from the results of the first stage.

4.4.1 First Stage Clustering

Each clustering algorithm has a set of mandatory parameters that were computed beforehand to maximize the final scores. In the case of K-Means, the silhouette [65] and elbow methods were performed to find the best k value. The elbow method computes the sum of squared distances from each point to its assigned center and plots it in a graph. The "elbow" corresponds to the k value where the average distance falls suddenly. Although it is a good indicator, sometimes it is unable to define the value of k

based on its graph [20]. For that reason, the silhouette method shows to give a better estimation of the k value.

As for the DBSCAN algorithm, the *minPts* parameter was set accordingly to [63], which suggests setting it to twice the dataset dimensionality ($2 \times d$) for more than two-dimensional data. The same study, states that we find a suitable value for eps by calculating the distance to the nearest $(2 \times d) - 1$ points for each point, sorting and plotting the results. The optimal value for eps is at the point of maximum curvature.

The number of components for the GMM algorithm was obtained via Bayesian Information Criterion (BIC). This criterion gives us an estimation of how good the GMM is in terms of predicting the data we have. The lower the BIC, the better the model to predict the data we have.

Finally, for EAC with K-Means, the study [47] suggests using a value for k that *"should be greater than the "true" (unknown) number of clusters, leading in general to highly fragmented data partitions, but preventing the formation of artificial clusters gathering samples belonging to distinct "natural groupings" of the data."*. We choose 10 as the *min k* value and 30 the *max*. For the number of ensembles, the same study recommends a high value, we opted for 250.

The results in Section 5.3.1 show the best parameter values for all algorithms with all combinations of dimensionality reduction and normalization techniques as described in Algorithm 1. After determining the best set of parameters, we applied the algorithm

Algorithm 1 Best parametrs

```
for normalization = distance, duration do
    for reduction = pca, svd, no reduction do
        for algorithm = kmeans, dbscan, gaussian mixture do
            Compute best parameters
        end for
    end for
end for
```

2 that performs the clustering itself for each algorithm. We provide the final results in Section 5.3.2, where we found two clusters, one representing aggressive trips and the other holds non-aggressive trips.

4.4.2 Second Stage Clustering

After clustering trips into aggressive and non-aggressive, we decided to apply a second-stage clustering. The purpose of this, was to find if there were more groups inside the

Algorithm 2 First Stage Clustering

```
for normalization = distance, duration do
    for reduction = pca, svd, no reduction do
        for algorithm = kmeans, dbscan, gaussian mixture, EAC kmeans do
            Apply clustering
            Compute Calinski-Harabasz, Davies-Bouldin and Silhouette scores
            Compute the number of instances assigned to each cluster
        end for
    end for
end for
```

two initial clusters. We applied the procedure described in Algorithm 3. It was only possible to cluster the aggressive trips because the Silhouette score was too low for the non-aggressive trips. The final clustering organized the aggressive trips into aggressive and risky trips. More detailed results are provided in Section 5.3.1 and 5.3.2.

Algorithm 3 Second Stage Clustering

```
Organize dataset in Aggressive and Non-Aggressive trips
for trips = aggressive, non − aggressive do
    Normalize by distance
    Apply PCA
    Find best k value
    Apply K-Means clustering
end for
```

4.4.3 Clusters Meaning

After clustering the data, it was imperative to interpret the clusters in both stages. Understanding the meaning of clusters can be more important than making clusters for this kind of problem. We took three different approaches to interpreting the meaning of the clusters:

- PCA analysis.
- Machine learning analysis with DT.
- Statistical analysis.

The objective of the PCA analysis was to find the most important features for the first component, which preserves the largest variance/information. After that, we used

the DT and RF algorithms to convert the problem of determining the meaning of a cluster into a classification problem. In other words, we attempted to find the relation between input (features) and output (cluster). Due to the structure of the DT, it was possible to obtain the most important features, which are the ones at the top of the tree. At last, we used statistical analysis to observe the different feature values between clusters, using the minimum, maximum, mean, and Standard Deviation (STD) values. Section 5.3.2 provides the results.

4.5 Classification

After clustering the trips, we took a supervised approach. The main purpose of this phase is to automatically classify a trip given the labels produced in the clustering phase. Here we applied the supervised algorithms and evaluation metrics detailed in Section 2.4.2 and Subsection 2.4.3.2. We decided to develop a machine learning pipeline to simplify the process of determining the trip profile. Instead of applying all the necessary phases (e.g. data normalization and reduction) one by one, we place them on a pipeline and then save the whole process. The devised pipeline is presented in Figure 4.1. For the normalization and dimensionality reduction phases, we used

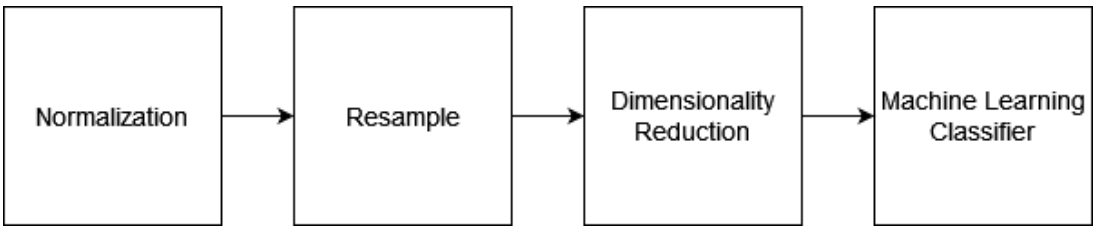


Figure 4.1: Machine Learning Pipeline

the same approach applied in the clustering analysis. All the trips were normalized by distance, and we applied PCA to reduce the dataset dimensionality. Section 4.5.1 details the approach used for the resample phase. Finally, Section 4.5.2 provides the methodology followed for model selection, including cross-validation and parameter tuning.

4.5.1 Imbalanced Learning

The generated clusters in the clustering phase are very imbalanced. This means that it was a must to perform resampling to create a more balanced distribution. In the literature, there are two approaches for resampling a dataset:

- Undersampling - Removes examples from the majority class to balance the proportion of the two classes.
- Oversampling - Replicates examples in the minority class to balance the proportion of the two classes.

We opted to use an oversampling approach because reducing the number of majority class examples removes potentially helpful information to train the models. A handful of oversampling techniques were taken into consideration: Random oversample; Synthetic Minority Oversampling Technique (SMOTE) [67]; Borderline SMOTE [9] and Adaptive Synthetic Sampling (ADASYN) [1]. To choose the best technique, we applied stratified k-fold cross-validation with ten folds and also took into consideration the reviewed literature. The article [29] used ADASYN to balance data for the i-DREAMS project because it gave the best results and also because it is an improved version of SMOTE and is considered the most suitable for handling imbalanced datasets and avoiding overfitting. We evaluated the model with the accuracy, weighted precision, weighted recall, weighted F1 score, and the area under the ROC curve (AUC) with a one versus one approach. Algorithm 4 shows the steps performed, and the results that confirm that ADASYN was the best technique are provided in Section 5.4.1.

Algorithm 4 Oversampling

```
for alg = decision tree, random forest, xgboost, svm do
    for resample = random oversample, smote, borderline smote, adasyn do
        Create pipeline
        Apply stratified 10-fold cross-validation
        Save mean test scores
    end for
end for
```

4.5.2 Model Selection

This section describes the methodology used for model selection, which involves cross-validation techniques and parameter tuning.

In typical cross-validation, the data is split into train and test subsets. Then, the training set is split into k smaller sets. For each k fold, the model is trained using $k-1$ of the folds as training data and tested in the remaining fold. The performance measure reported by k-fold cross-validation is the average of the values computed in the loop. With this method, initially, the data is organized into training and test subsets, but we

only use cross-validation on the training set, and for that reason, the results on the test set may be unstable. We applied nested cross-validation as a direct consequence.

Nested cross-validation can evaluate the model's performance on the entire dataset. This approach, is composed of two loops, the inner loop and the outer. The inner loop is responsible for selecting the best model (including parameters). The outer loop, is used for estimating the quality of the models trained on the inner loop. With nested cross-validation, we are not assessing just the quality of the model but the quality of the procedure for model selection. In other words, for each iteration of the outer loop, we select one (and only one) inner model (the best one), and this model will be evaluated on the test set for this outer fold. After we vary the outer test set, we will have k estimates, and we can average them to better assess the model's quality. For the inner and the outer loop, we applied a stratified cross-validation strategy, which returns stratified folds: each set contains approximately the same percentage of samples of each target class as the complete set. The reason we picked this strategy was that it deals well with unbalanced data, which is our case. For the inner loop, we used ten folds, and for the outer loop, we used only two because of hardware and time constraints. Algorithms such as SVM with a one versus one approach, increase in time and complexity when the number of classes increases. This is the reason why we could not increase the number of folds in the outer loop. The results that evaluate the model's performance are reported in Section 5.4.2.

At the end of nested cross-validation, we do not obtain the final best estimator because it is not the purpose. To obtain it, we split the dataset in a stratified fashion into train and test subsets, giving 80% of the data to the training set. Before the split, we shuffled the data. After, we performed parameter tuning (with cross-validation) again but only on the training set. In this case, we used ten stratified folds. In both nested-cross-validation and parameter tuning we used the grids shown in Appendix C, for the whole pipeline (includes classifier and ADASYN parameters).

Afterwards, we apply the pipeline on the test data. The results are provided in Section 5.4.2.

4.6 Driver Profile

The central purpose of this thesis is to establish a driver profile from trip data. After supervised learning, we classified trips into three categories: Non-Aggressive, Aggressive, and Risky. The final step was to devise a driver profile from the driver's trips. The approach we took was to find the most common category from the trips. In case there

is more than one common category, we prefer the category that appears more times consecutively. If no category appears more times consecutively, we choose the most recent category. This approach is somewhat naive and fails if the driver's behavior is inconsistent over time because it is impossible to assign a specific driving profile. To overcome this problem, we decided to use a similar method as in [40]. We applied a function that computes the gain/loss ratio per trip because it is crucial to observe how the driver shifts between the identified categories and whether one shows stability in his driving behavior. The function corresponds to the gain or loss respectively, a driver receives according to the way one shifts between successive trips, and is defined as follows:

$$r_{t,i} = ln\left(\frac{S_{t,i}}{S_{t-1,i}}\right), \tag{4.1}$$

where $S_{t,i}$ corresponds to the codified category predicted in the supervised phase, on the trip number t and driver i. $S_{t-1,i}$ corresponds to the codified category of the previous trip. This indicator has a positive value when the driver improves his behavior over the previous trip, whereas negative values indicate that the driver has moved toward less safe behavior. If the value is zero, the driver maintained his behavior. To detect the stability of the way the driver shifts between the different driving categories we measured the driver's behavior volatility, which consists of the gain/loss ratio standard deviation:

$$Driver's\ behavior\ volatility = \sqrt{\frac{\sum_{t=1}^{t}(r_{t,i} - \bar{r}_i)^2}{t-1}}, \tag{4.2}$$

where $r_{t,i}$ corresponds to the gain/loss per trip of driver i, $\bar{r}_i$ is the average of gain/loss ratio and t the number of trips.

Based on this mechanism, we tested if the i-DREAMS drivers had consistent driver behavior or not. The results are provided in Section 5.5.

4.7 Driver Profile API

The final main objective of this thesis was to develop a REST API with the purpose of integration with an external application. The purpose of this API is for others to use and serves as an example of an approach to classify drivers and fleets. Because CardioID provided the data for this thesis, this API is better suited for incorporation into CardioID-related products. The API, however, was developed so that anyone who required a similar system may use it in addition to CardioID. We took into consideration the following REST principles when developing the API:

- Uniform interface - All API requests for the same resource should look the same, no matter where the request comes from. The developed API uses JSON as a data format in all requests. It also prioritizes nouns over verbs in Uniform Resource Identifier (URI) and uses the HTTP methods to differentiate different request types to the same resource.
- Client-Server decoupling - To make sure the API can be used by an external client application, the client and server applications are independent of each other. The only thing the client application knows is the URI of the requested resource.
- Statelessness - Each request needs to include all the information necessary for processing it. Some requests do not use JSON data because it is not necessary.
- Layered system architecture - The API is structured in a three-layered system where each layer/subsystem has a distinct task to perform. The first layer is the presentation subsystem and is responsible for accepting requests from the user and displaying data in a user-friendly format (JSON in this case). The second layer is the domain subsystem and is responsible for the business logic. The last layer is the data access subsystem, responsible to interact with persistent data stored in the database, or external services.

The overall architecture solution is presented in Figure 4.2, using a subsystem diagram. We made sure that each layer only interacts with the subsequent layer to reduce coupling between layers. This means that the presentation layer only interacts with the domain layer and the domain with the data access layer. The models package inside the data access subsystem is responsible to create the data models. The main requirement for this system is to have drivers that can perform trips and be associated with a client (company). The clients may have fleets that can be part of the trip. Finally, a trip must always be performed by a driver and, in the end, the trip profile is established. Given these requirements, Figure 4.3 shows the conceptual model (entity relationship model). In other words, it describes the relationships between entities and their attributes.

Considering the entity relationship model it was possible to compose the logical model in a relational model described below.

Client(id, uuid, name, created)
Candidate Keys = {id}

Driver(id, uuid, name, created, client_id)
Candidate Keys = {id}
Foreign Keys = {client_id}

Fleet(id, uuid, name, created, client_id)
Candidate Keys = {id, client_id}
Foreign Keys = {client_id}

Trip(id, uuid, created, distance, duration, start, end, profile, driver_id, fleet_id)
Candidate Keys = {id, driver_id}
Foreign Keys = {driver_id, fleet_id}

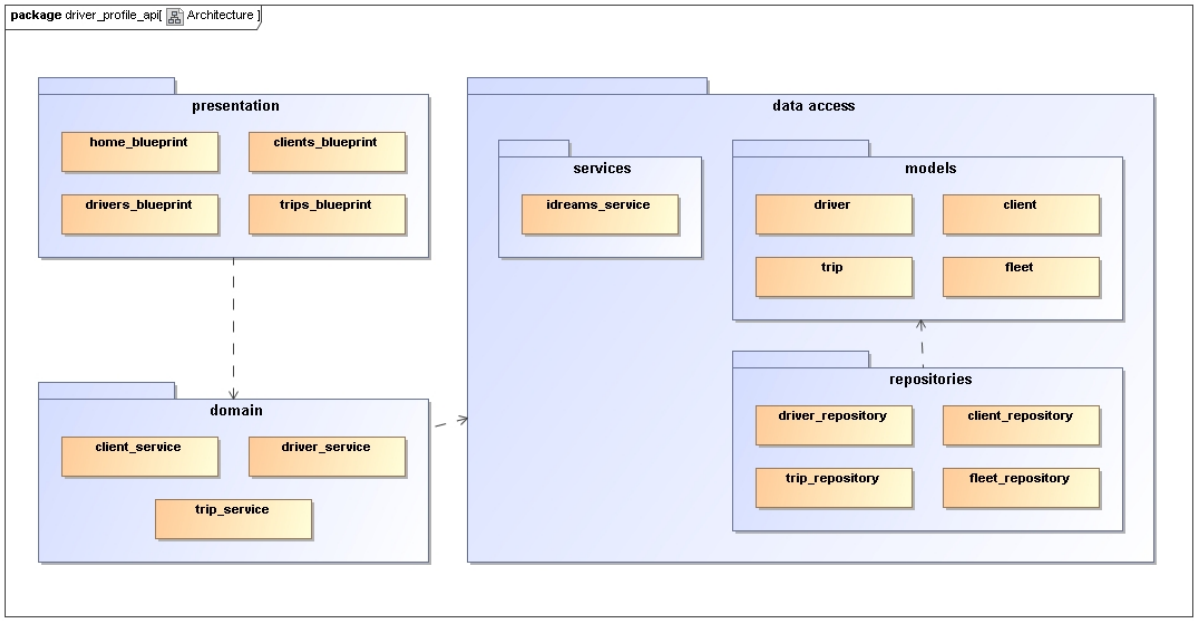


Figure 4.2: Overall Architecture Solution

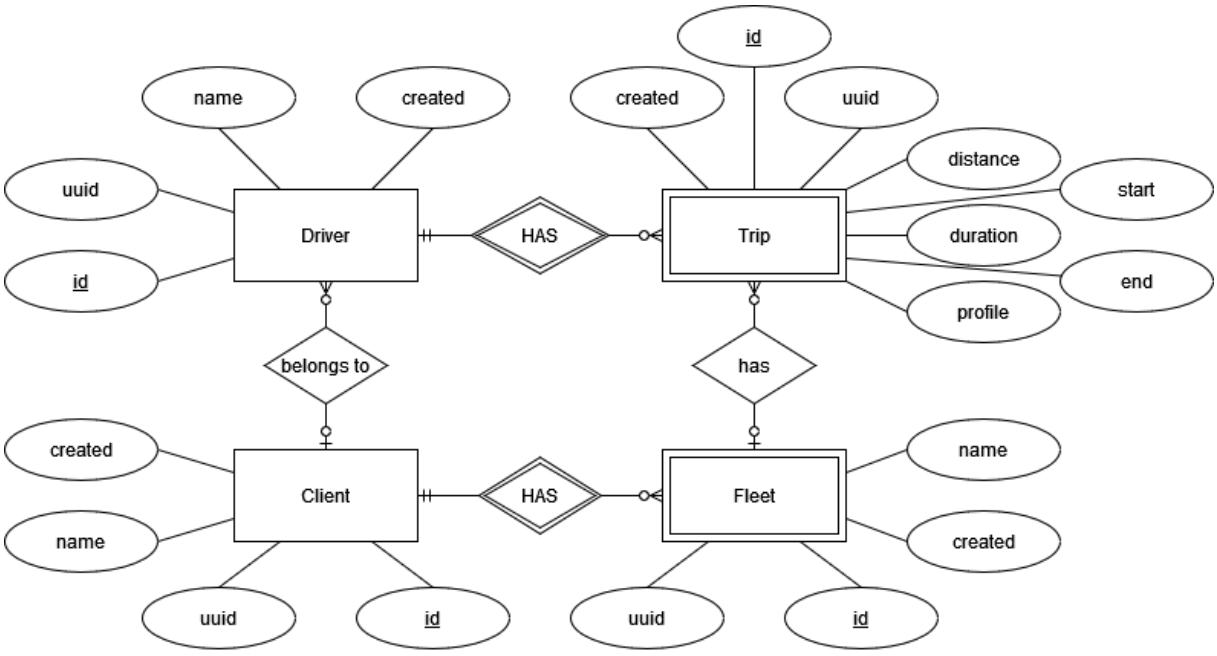


Figure 4.3: Entity Relationship Model

To convert the logical model into a physical model, we used Object-Relational Mapping (ORM), where objects are used to connect the programming language to the database systems, with the facility to work with Structured Query Language (SQL) and object-oriented programming concepts. We opted to use SQLAlchemy [68] Python package because it is the most used Python SQL toolkit and Object Relational Mapper. Additionally, we used the MariaDB Database Management System (DBMS) [41] to create the data models.

In terms of endpoints, we have developed the following:

- Create driver - /drivers [POST]
- Create client - /clients [POST]
- Create trip - /trips [POST]
- Get clients - /clients [GET]
- Get drivers - /drivers [GET]
- Get driver trips - /drivers/<uuid>/trips [GET]
- Get fleet trips - /clients/<uuid>/fleets/<uuid>/trips [GET]
- Get driver profile - /drivers/<uuid>/profile [GET]
- Get fleet profile - /clients/<uuid>/fleets/<uuid>/profile [GET]
- Add client drivers - /clients/<uuid>/drivers [PATCH]
- Add client fleets - /clients/<uuid>/fleets [PATCH]

A detailed documentation of each endpoint is presented in the API repository.

4.8 Thesis Repositories

The source code of this thesis is available in two different repositories. The first repository [55] has the code implemented for the machine learning phase. This repository also stores all the resulting images of the modeling and pre-processing steps. The code was developed in Python [53] using the Scikit-learn [64] library for machine learning tasks and the Pandas [48] library to handle data analysis and manipulation. Additional libraries used are provided in the repository.

The second repository [56] provides the source code of the implemented REST API. It also includes unit tests of the endpoints and documentation. We used Python as a programming language with the Flask [25] package as a framework for API development.

5

Experimental Evaluation

This chapter presents and evaluates the experimental results obtained from the data pre-processing phase to the driver profile estimation phase.

Section 5.1 provides the testing execution environment details. Section 5.2 describes the results obtained from the feature reduction phase via correlation matrix and the number of features/components. Section 5.3 reports the experimental results of the unsupervised learning phase. After this, Section 5.4 shows the results obtained in the supervised learning phase. Finally, Section 5.5 concludes this chapter with the results obtained from the i-DREAMS driver's profile test.

5.1 Testing Environment

This section aims to define the hardware, software, Central Processing Unit (CPU), and Random Access Memory (RAM) that compose the test execution environment for the machine learning and API development. The testing environment was the following:

- Machine - Lenovo Legion Y520-15IKBM
- Processor - Intel(R) Core(TM) i7-7700HQ, CPU 2.80GHz with 4 cores and 8 logical processors
- Installed RAM - 16,0 GB (15,9 GB usable)

- System type - 64-bit operating system, x64-based processor
- Operating system - Windows 10 Home
- Operating system version - 21H1

For supervised learning, we used all the 8 processors to perform nested cross validation and parameter tuning.

5.2 Feature Reduction

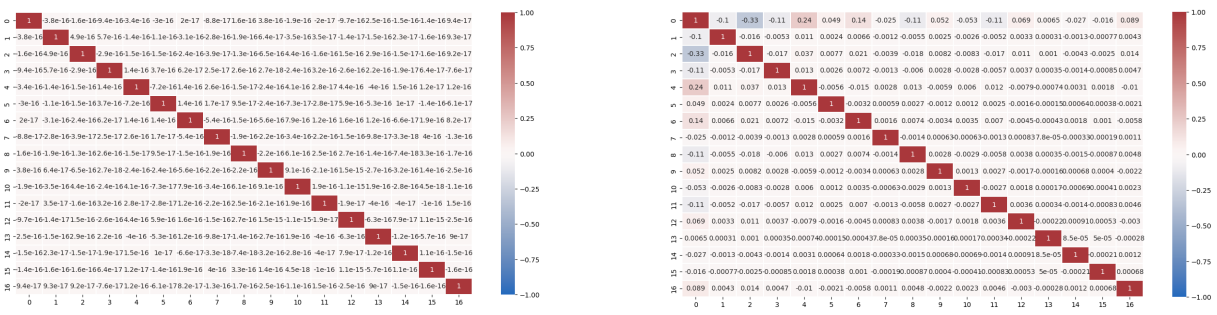
In this section, we present and evaluate the results obtained from the feature reduction task before unsupervised learning.

Table 5.1 displays the number of components obtained via PCA and SVD not only for the distance and duration normalization but also without applying any normalization. Analyzing the results, it is clear to see that, without normalization, the feature reduction was excessively high. The dataset went from 53 dimensions to only 1. For that reason, we opted to only consider PCA and SVD techniques with normalized data.

Table 5.1: Feature Reduction Components preserving 99% of variance

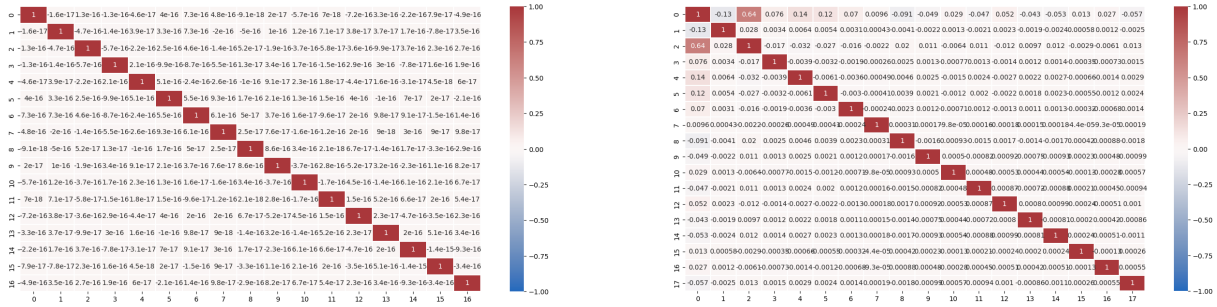
Distance		**Duration**		**No Norm**	
PCA	SVD	PCA	SVD	PCA	SVD
17	17	17	18	1	1

Given the results from Table 5.1, the correlation matrices from the distance and duration normalization techniques are presented in Figure 5.1 and 5.2, respectively.



(a) PCA reduction (b) SVD Reduction

Figure 5.1: Distance normalization correlation matrices



(a) PCA reduction (b) SVD Reduction

Figure 5.2: Duration normalization correlation matrices

5.3 Unsupervised Learning

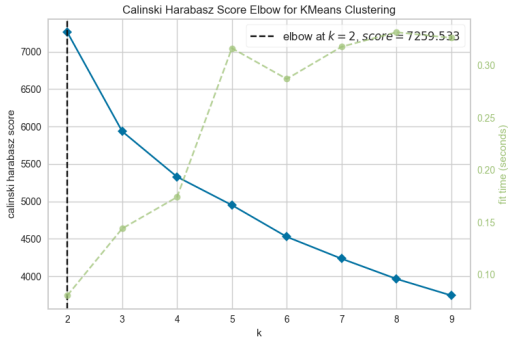
5.3.1 Best Parameters

For each algorithm, we determined the best set of parameters. Firstly, K-Means was the only algorithm applied in both first and second-stage clustering. For the first stage, Table 5.2 shows that $k = 2$ is the best value for all combinations of normalization and reduction techniques. Additionally, we provide Figure 5.3 for the distance normalization and PCA reduction (because they gave the best clustering results) that confirm the results shown in the table. For the second stage clustering, we determined the best

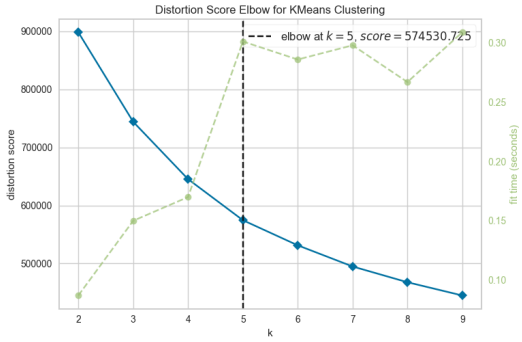
Table 5.2: K-Means best k value

Distance			**Duration**		
PCA	SVD	No reduction	PCA	SVD	No reduction
2	2	2	2	2	2

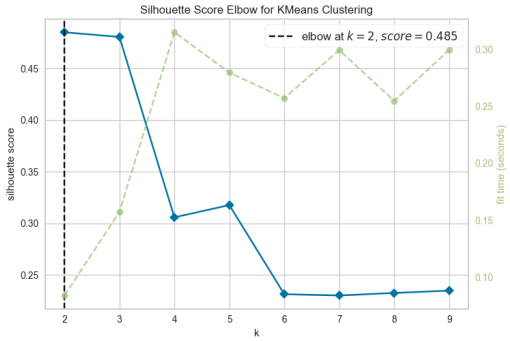
k value for the aggressive and non-aggressive trips as shown in Table 5.3. The results obtained for the aggressive trips reveal that $k = 2$ is the best value with great scores. For the non-aggressive trips, the Silhouette score is very low and close to zero, which corresponds to overlapping clusters with samples very close to the decision boundary of the neighboring clusters. For that reason, we decided to only cluster the aggressive trips.



(a) Calinski Harabasz score



(b) Sum of Squared Distances score



(c) Silhouette score

Figure 5.3: K-Means best k value for distance normalization and PCA

Table 5.3: Second Stage Clustering best k value. Elbow method with Calinski Harabasz (CH), Sum of Squared Distances (SSD) and Silhouette scores

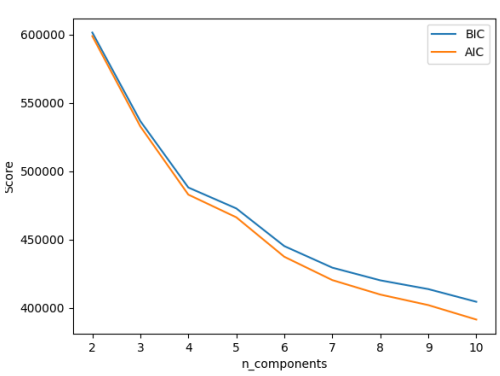
	Aggressive			**Non-Aggressive**		
	CH ↑	SSD ↓	Silhouette ↑	CH ↑	SSD ↓	Silhouette ↑
Score	1144.656	273108.864	0.735	3486.637	231625.608	0.267
k	2	4	2	3	3	3

For the DBSCAN algorithm, as mentioned in Section 5.3.2, we calculated the average distance between each point and its $(2 \times d) - 1$ nearest neighbors to determine the best eps value. Table 5.4 provides the optimal eps value (point of maximum curvature).

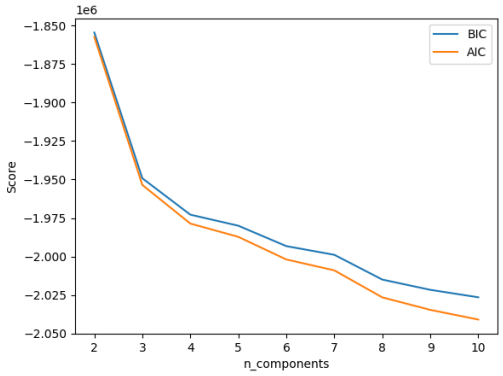
Table 5.4: DBSCAN best eps value

Distance			**Duration**		
PCA	SVD	No reduction	PCA	SVD	No reduction
7	7	7	0.065	0.065	0.065

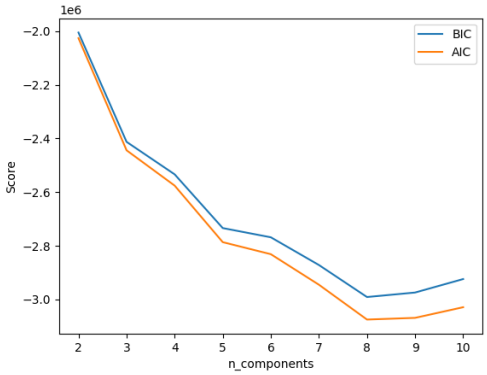
Finally, for GMM we used the BIC criterion. For an optimal value, we should choose the number of components where the BIC score is the lowest or where it starts to level off. When compared to the K-Means, the results obtained revealed that there was not an ideal value for the number of components, because the BIC score never really starts to level off, or when it does, the number of components is high resulting in poor clustering. Figure 5.4 highlights this point of view for the distance normalization.



(a) PCA



(b) SVD



(c) Without Reduction

Figure 5.4: Gaussian Mixture best number of components for distance metric

5.3.2 Clustering

In this section, we address the clustering results. Initially, we provide the best results for each algorithm according to the scores and the number of instances assigned to each cluster, like in Algorithm 2. Afterward, we combine the best results of each algorithm and choose the one with the best scores.

Tables 5.5, 5.6, 5.7, and 5.8 provide the best results for each algorithm highlighted in bold.

In K-Means results, we can see that, for all cases, the cluster distribution is fairly the same. However, with distance metric and PCA reduction, we obtain the best results for the Davies-Bouldin and Silhouette scores. We used different distance metrics, but the euclidean and euclidean square provided the best results.

DBSCAN results were not as easy to compare. The majority of data was labeled as -1 for "noise", which indicates that the *eps* parameter was probably too small. However, when we increased this value the entire dataset was assigned to a single cluster. To conclude, it was very difficult to find the best configuration for this algorithm due to the *eps* value. However, we choose the combination of distance and SVD because it has a better score overall and better cluster balance.

As for Gaussian Mixture, the results indicate that the combination of distance and PCA resulted in the best scores. As a reminder, it was not possible to find the best number of components. For that reason, we used two components as in K-Means.

Finally, for EAC with K-Means, the results were inconsistent due to the natural diversity of the algorithm. The number of instances assigned to each cluster changes a lot when running the algorithm multiple times, nonetheless, the results shown were the most consistent. For the duration normalization, the Silhouette score gave the best results. However, this is because with imbalanced clusters (6 instances in cluster 0 and 14996 in cluster 1) the Silhouette score increases. On the other hand, the distance normalization was able to place more instances in cluster 0, which better defines the data. We opted to choose the combination of distance and PCA because it provides more balanced clusters.

Table 5.5: K-Means results

Scores	Distance			Duration		
	PCA	SVD	No reduction	PCA	SVD	No reduction
Calinski-Harabasz ↑	7258.919	**7259.569**	7162.864	4503.604	4614.778	4563.278
Davies-Bouldin ↓	**1.164**	1.167	1.177	1.165	1.314	1.303
Silhouette ↑	**0.488**	0.485	0.483	0.443	0.438	0.435
Instances cluster 0	11916	11852	11852	12367	12358	12416
Instances cluster 1	3086	3150	3150	2635	2644	2586

For the final evaluation, we combined the best results to choose the best algorithm. Table 5.9 presents the best scores for each algorithm. The EAC K-Means obtained better

Table 5.6: DBSCAN results

Scores	Distance			Duration		
	PCA	SVD	No reduction	PCA	SVD	No reduction
Calinski-Harabasz ↑	2365.483	2365.621	**2913.508**	1844.437	1918.241	2149.874
Davies-Bouldin ↓	**1.372**	**1.372**	1.407	1.488	1.486	1.583
Silhouette ↑	0.650	0.650	0.621	**0.715**	0.712	0.669
Instances cluster -1	452	452	680	277	293	469
Instances cluster 0	14550	14550	14322	14725	14709	14533

Table 5.7: Gaussian Mixture results

Scores	Distance			Duration		
	PCA	SVD	No reduction	PCA	SVD	No reduction
Calinski-Harabasz ↑	**5602.887**	5596.921	5465.703	3585.217	3575.389	3517.701
Davies-Bouldin ↓	**1.377**	**1.377**	1.391	1.698	1.701	1.715
Silhouette ↑	**0.382**	0.381	0.373	0.366	0.365	0.355
Instances cluster 0	5287	5301	5414	10594	10581	10419
Instances cluster 1	9715	9701	9588	4408	4421	4583

Davies-Bouldin and Silhouette scores due to high imbalanced clusters. The disadvantage of this algorithm was that the results were inconsistent. Sometimes the distribution of instances in each cluster was higher, other times lower. When the distribution was approximately the same as the other algorithms, K-Means ended up with better scores. On the other hand, the K-Means algorithm produced a better cluster balance with decent results. The K-Means outperforms the other algorithms in the Calinski-Harabasz score and comes in second (after EAC K-Means) with the lowest Davies-Bouldin score. Consequently, we decided to use the K-Means algorithm with distance normalization and PCA reduction.

To attribute meaning to the clusters obtained with K-Means, we verified which features were the most important for the first PCA component. The most important features in order were: *speed*, *n_tsr_level*, *n_brakes*, *n_hc*, and *n_ha*. Figure 5.5 compares the *speed* with *n_tsr_level* and *n_brakes* features, and reveal that both features are able to distinguish the clusters. From this figure, we can say that cluster 0 represents trips with low-speed values per km and low speeding events per km. Cluster 1 represents trips with low to medium speed/km and low to medium speeding events per km. In terms of the number of braking events per km, cluster 1 has more events.

After that, we applied DT and RF algorithms to determine the top three features:

Table 5.8: EAC with K-Means results

Scores	Distance			Duration		
	PCA	SVD	No reduction	PCA	SVD	No reduction
Calinski-Harabasz ↑	**1965.691**	1261.184	1248.881	1263.116	1260.945	1250.079
Davies-Bouldin ↓	0.482	**0.274**	0.275	0.305	0.305	0.306
Silhouette ↑	0.807	0.867	0.866	**0.919**	**0.919**	**0.919**
Instances cluster 0	54	16	16	6	6	6
Instances cluster 1	14948	14986	14986	14996	14996	14996

Table 5.9: Best Algorithm

Scores	K-Means	DBSCAN	Gaussian Mixture	EAC K-Means
Calinski-Harabasz ↑	**7258.919**	2365.621	5602.887	1965.691
Davies-Bouldin ↓	1.164	1.372	1.377	**0.482**
Silhouette ↑	0.488	0.650	0.382	**0.807**
Instances cluster 0	11916	14550	5287	54
Instances cluster 1	3086	-	9715	14948
Instances cluster -1	-	452	-	-

- Decision Tree - *speed*, *n_brakes*, *n_tsr_level*
- Random Forest - *speed*, *n_fatigue_0*, *n_overtaking_0*

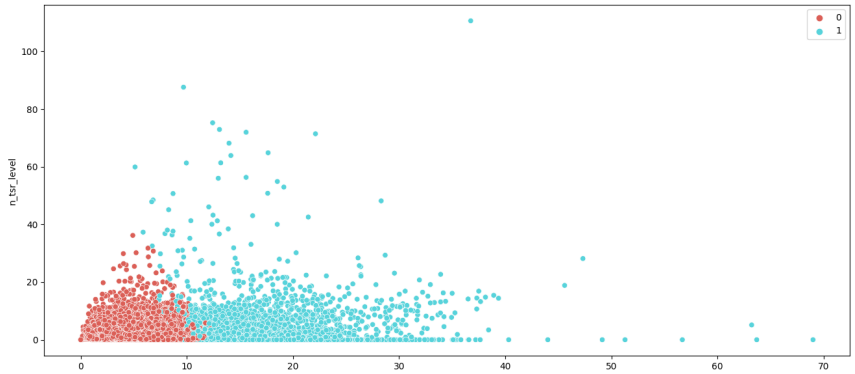
As expected, the most important features from the DT are the same as in the PCA first component but in a different order. For the RF algorithm only the speed feature remains the most important. The tree resulting from the DT classifier is shown partly in Figure 5.6.

Finally, we compared the *min*, *max*, *mean*, and *standard deviation* values of each feature in the different clusters. Table 5.10 presents these values for the most important features. When summarizing all the analysis results, we can say that for almost all the

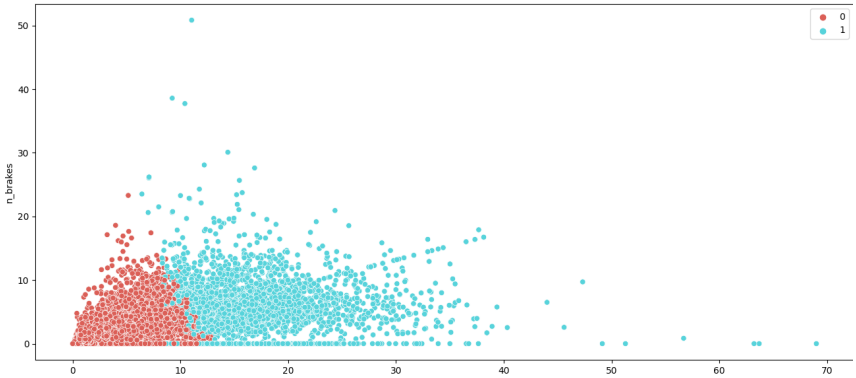
Table 5.10: Cluster Statistical Analysis

Features	Cluster 0				Cluster 1			
	MIN	MAX	MEAN	STD	MIN	MAX	MEAN	STD
speed	0.0	13.31	4.50	2.66	5.12	69.00	16.78	5.90
n_tsr_level	0.0	36.19	2.82	2.75	0.0	110.60	4.32	7.65
n_brakes	0.0	23.30	2.38	1.92	0.0	50.85	5.72	4.21

features cluster 1 has more events/km traveled. Cluster 1 corresponds to more aggressive trips when compared to cluster 0. However, there is no way to say that you must



(a) speed versus n_tsr_level



(b) speed versus n_brakes

Figure 5.5: First Stage Clustering Analysis

have, for example, a certain number of speeding events to consider the trip aggressive or non-aggressive. For instance, the trips belonging to cluster 0 may be considered aggressive for other datasets depending on the problem context. Therefore, we made this selection according to the characteristics of the dataset at hand. In conclusion, we considered cluster 1 trips as aggressive when compared to cluster 0 trips, which were described as non-aggressive.

A second stage clustering was applied for the aggressive trips with K-means using $k = 2$. Table 5.11 shows the K-Means scores when combined with distance normalization and PCA reduction.

Table 5.11: Second stage K-Means results

Calinski-Harabasz ↑	Davies-Bouldin ↓	Silhouette ↑	Instances cluster 0	Instances cluster 1
1144.656	0.548	0.735	3034	53

The scores obtained seem to cluster the aggressive trips very well. An interesting result was that the number of instances that were assigned to cluster 1 is almost the same as in the EAC with K-Means used in the first stage of clustering. Here we get 53

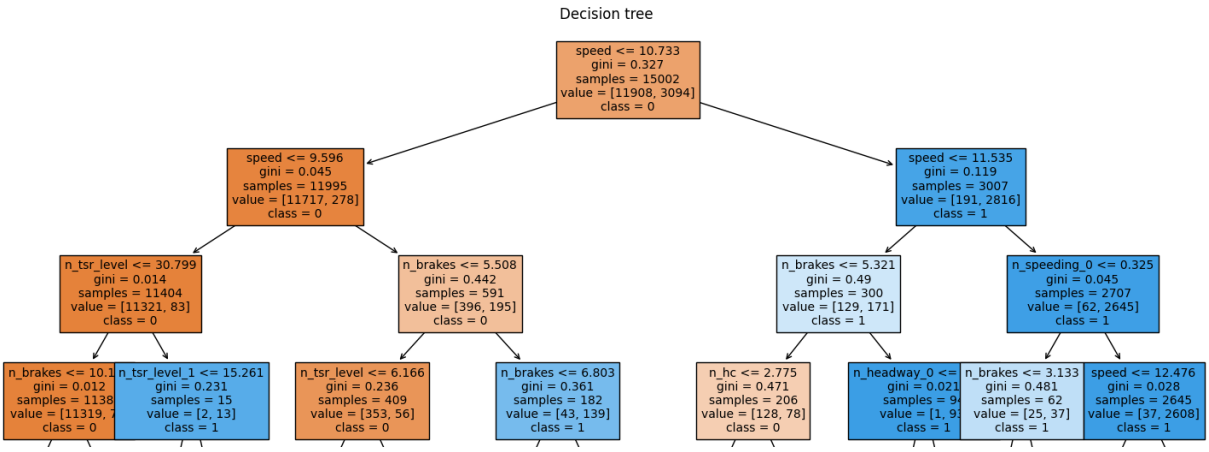


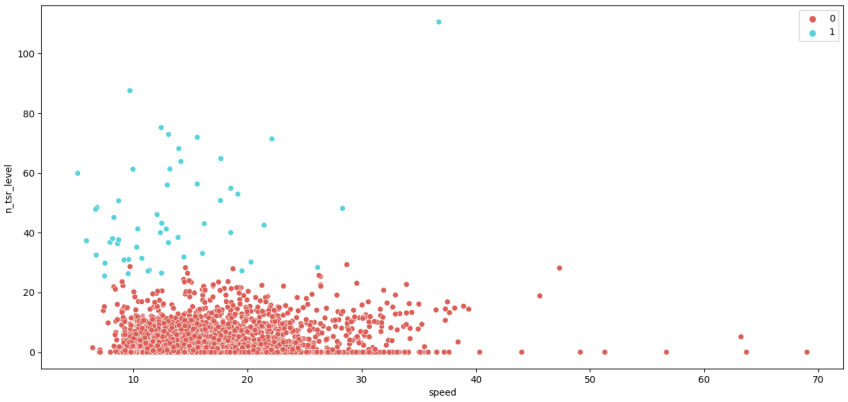
Figure 5.6: Decision Tree

trips for cluster 1, and in the EAC case, we get 54. The main point to take from this is that the EAC with K-Means seems to cluster all the data in the same cluster except for outliers.

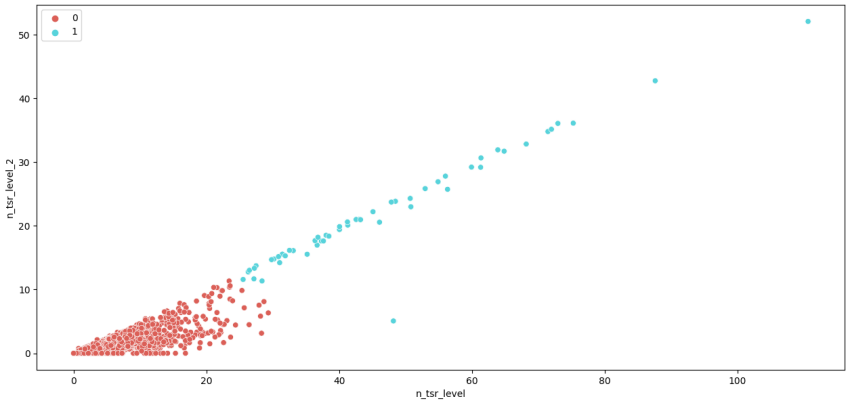
After this, we took the same approach applied in the first stage of clustering to attribute meaning to the clusters. The most important features for the first PCA component were the same: *speed*, *n_tsr_level*, *n_brakes*, *n_hc* and *n_ha*. For the DT analysis the most important features were: *n_tsr_level_2*, *n_tsr_level_5*, and *n_speeding_3*. For the RF: *n_tsr_level_2*, *n_tsr_level_1*, and *n_tsr_level*. With these results, we can say that the number of events exceeding the speed limit was the most important feature at differentiating both clusters. Figure 5.7 compares the *speed* feature with the *n_tsr_level* and the *n_tsr_level* with the *n_tsr_level_2*.

We can see that the *n_tsr_level* feature produces more events/km in cluster 1. The *n_tsr_level_2* feature also has a lot more events in cluster 1. That means that cluster 1 has more speeding events where the driver was 5-10 units over the speed limit than cluster 0. To better characterize cluster 1, we observed that the study [40] classified aggressive trips into aggressive, distracted, and risky. The risky trips involved more speeding events than the aggressive trips. Consequently, we decided to assign cluster 1 instances as risky trips when compared to cluster 0. The cluster 0 trips remained as aggressive trips.

To conclude the clustering phase, the final clusters and labels are presented in Table 5.12 and Figure 5.8. The dataset was labeled accordingly, and before proceeding to supervised learning the dataset was composed by $n = 15002$ trips, $d = 43$ features, and $c = 3$ classes.



(a) speed vs n_tsr_level



(b) n_tsr_level vs n_tsr_level_2

Figure 5.7: Second Stage Clustering Analysis

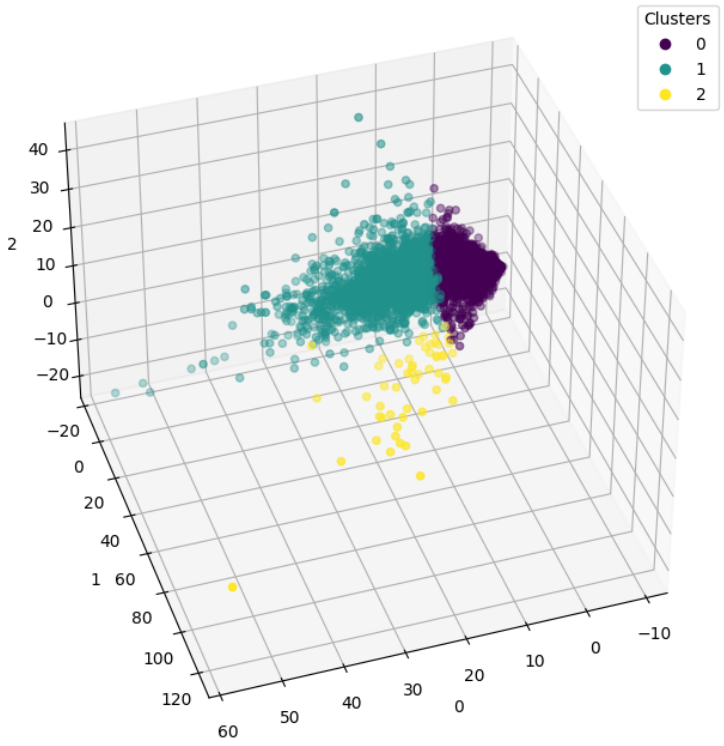


Figure 5.8: Clustering Output

Table 5.12: Final Clustering Results

Label	Description	Number of instances
0	Non-Aggressive trips	11915
1	Aggressive trips	3034
2	Risky trips	53

5.4 Supervised Learning

5.4.1 Oversampling Results

In this section we provide the experimental results obtained for all the oversampling techniques and algorithms used. Table 5.13 confirms that the ADASYN was the resampling technique that obtained the best score among all the algorithms except for the DT classifier. For this reason, in the resample phase of the proposed pipeline we opted to use ADASYN.

Table 5.13: Oversampling Mean Test Results with Accuracy (ACC), Precision (PREC), Recall (REC), and AUC scores

Algorithm	Random Oversampling					SMOTE				
	ACC	PREC	REC	F1	AUC	ACC	PREC	REC	F1	AUC
Decision Tree	0.9936	0.9936	0.9936	0.9936	0.9853	0.9936	0.9936	0.9936	0.9936	0.9819
Random Forest	0.9939	0.9939	0.9939	0.9939	0.9998	0.9945	0.9946	0.9945	0.9945	**0.9999**
XGBoost	0.9961	0.9962	0.9961	0.9961	**0.9999**	**0.9964**	**0.9965**	**0.9964**	**0.9964**	**0.9999**
SVM	**0.9967**	**0.9968**	**0.9967**	**0.9967**	**0.9999**	0.9963	0.9964	0.9963	0.9962	**0.9999**

Table 5.13: Oversampling Mean Test Results (*Continuation*)

Algorithm	Borderline SMOTE					ADASYN				
	ACC	PREC	REC	F1	AUC	ACC	PREC	REC	F1	AUC
Decision Tree	0.9913	0.9915	0.9913	0.9914	0.9845	0.9881	0.9882	0.9881	0.9881	0.9654
Random Forest	0.9948	0.9949	0.9948	0.9948	**0.9999**	0.9953	0.9954	0.9953	0.9954	0.9956
XGBoost	0.9938	0.9939	0.9938	0.9938	0.9997	**0.9971**	0.9971	**0.9971**	**0.9971**	**0.9999**
SVM	**0.9970**	**0.9971**	**0.9970**	**0.9970**	**0.9999**	**0.9971**	**0.9972**	**0.9971**	**0.9971**	**0.9999**

5.4.2 Classification Results

This Section presents the classification results of the test data for all the supervised algorithms used and also the overall model's performance with nested cross-validation.

Starting with nested cross validation results, Table 5.14 presents the mean values for each score.

Table 5.14: Nested Cross-Validation Results

Algorithm	Accuracy	Precision	Recall	F1	AUC
Decision Tree	0.9568	0.9639	0.9568	0.9583	0.9866
Random Forest	0.9783	0.9801	0.9783	0.9787	0.9989
XGBoost	0.9940	0.9941	0.9940	0.9939	0.9989
SVM	**0.9986**	**0.9986**	**0.9986**	**0.9986**	**0.9999**

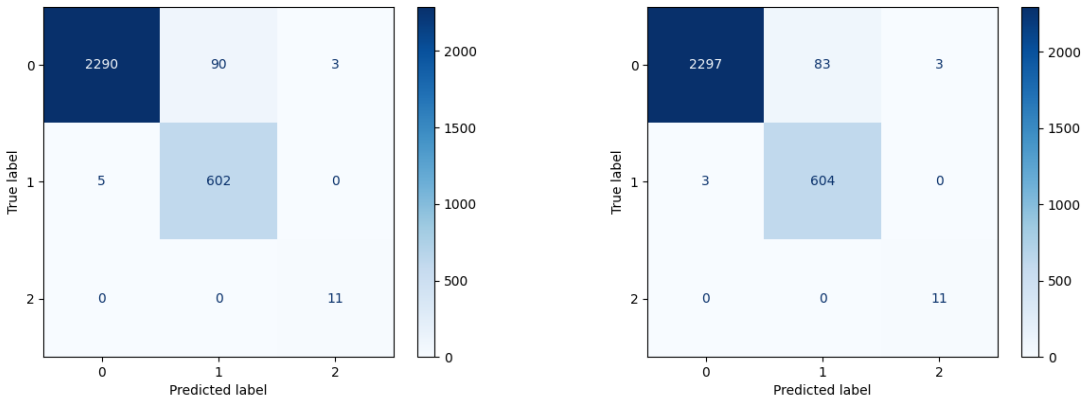
Figure 5.9 illustrates the confusion matrices obtained for each algorithm. From the confusion matrices it was possible to retrieve the information shown in Table 5.15 and 5.16. We also plotted for each algorithm the Precision-Recall and the ROC curves, shown in Figures 5.10, 5.11, 5.12 and 5.13.

Table 5.15: Classification Test Report (Decision Tree and Random Forest)

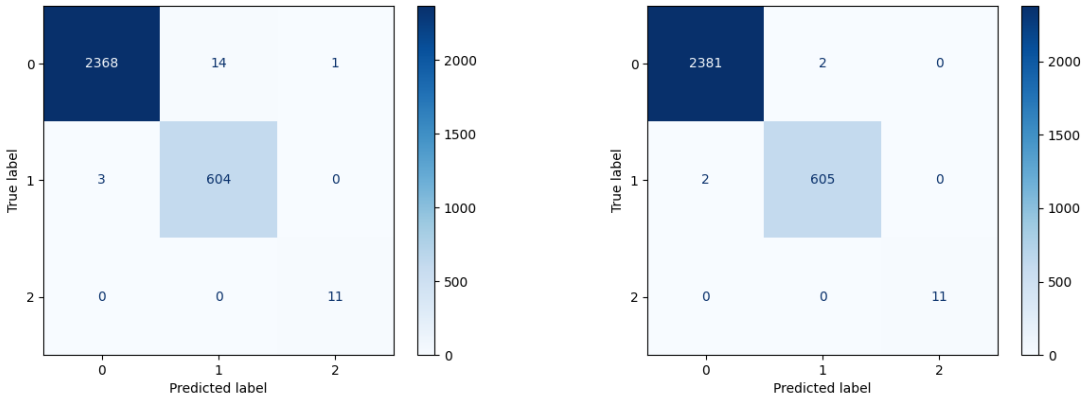
Class	Decision Tree				Random Forest			
	Precision	Recall	F1	Support	Precision	Recall	F1	Support
0	0.998	0.961	0.979	2383	0.999	0.964	0.981	2383
1	0.870	0.992	0.927	607	0.879	0.995	0.934	607
2	0.786	1.000	0.880	11	0.786	1.000	0.880	11
accuracy			0.967	3001			0.970	3001
macro avg	0.884	0.984	0.929	3001	0.888	0.986	0.932	3001
weighted avg	0.971	0.967	0.968	3001	0.974	0.970	0.971	3001

Table 5.16: Classification Test Report (XGBoost and SVM)

Class	XGBoost				SVM			
	Precision	Recall	F1	Support	Precision	Recall	F1	Support
0	0.999	0.994	0.996	2383	0.999	0.999	0.999	2383
1	0.977	0.995	0.986	607	0.997	0.997	0.997	607
2	0.917	1.000	0.957	11	1.000	1.000	1.000	11
accuracy			0.994	3001			0.999	3001
macro avg	0.964	0.996	0.980	3001	0.999	0.999	0.999	3001
weighted avg	0.994	0.994	0.994	3001	0.999	0.999	0.999	3001



(a) Decision Tree and Random Forest



(b) XGBoost and SVM

Figure 5.9: Confusion Matrices

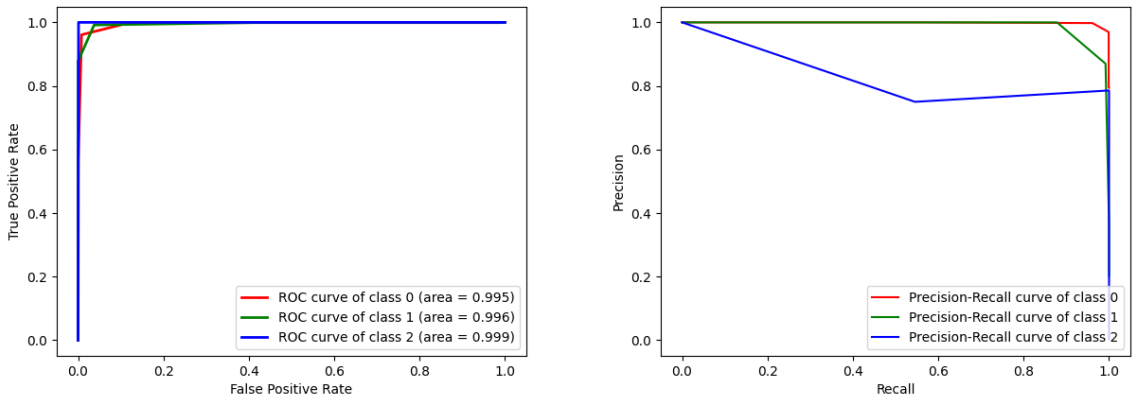


Figure 5.10: Decision Tree ROC and Precision-Recall curves

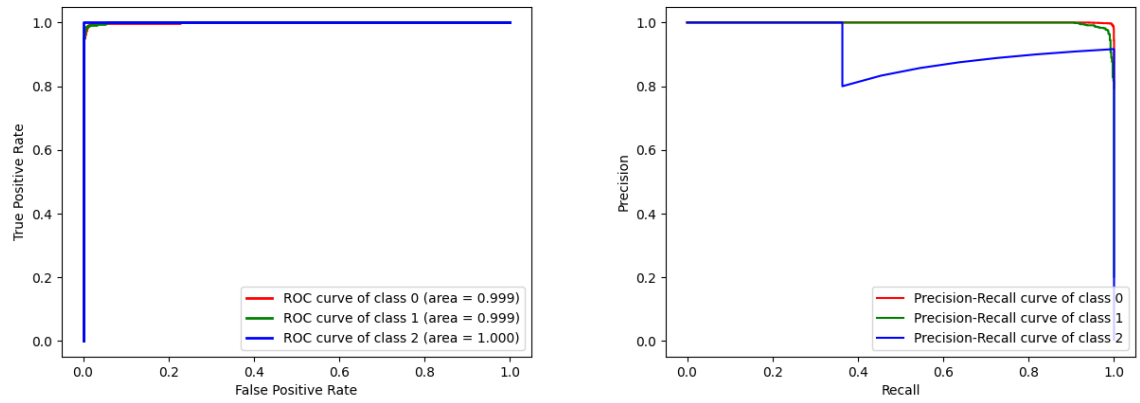


Figure 5.11: Random Forest ROC and Precision-Recall curves

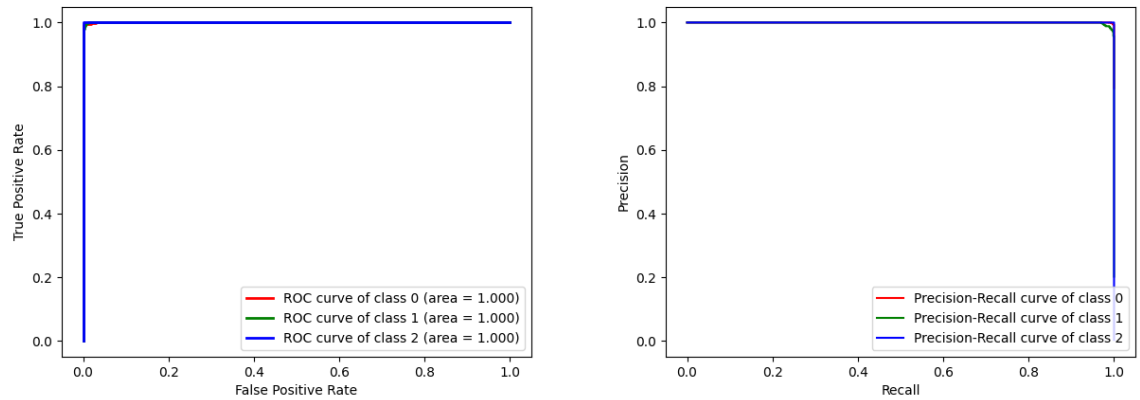


Figure 5.12: XGBoost ROC and Precision-Recall curves

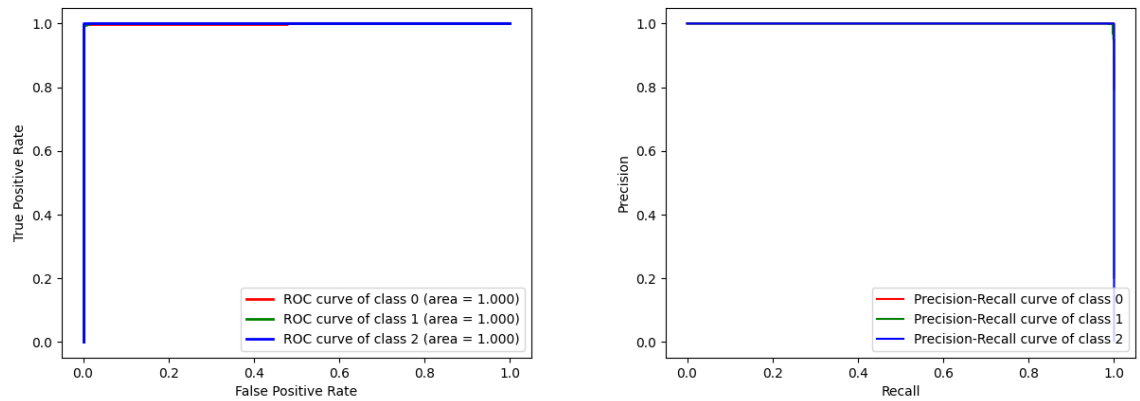


Figure 5.13: SVM ROC and Precision-Recall curves

The results state that the SVM algorithm was the best at predicting trips profile

with excellent scores. Each classifier failed at predicting examples of classes 0 and 1, but most frequently on class 0, except the SVM classifier that exhibits the same error (recall) in both classes. For the DT classifier, the error on class 0 was about 3.9% and on class 1 around 0.82%. For the RF classifier, the error on class 0 was about 3.6%, and on class 1 around 0.49%. For the XGBoost classifier, the error on class 0 was about 0.63%, and on class 1 around 0.49%. Finally, for the SVM classifier, the error on class 0 was about 0.08%, and on class 1 around 0.33%. Each classifier except SVM obtained the worst results on the precision metric for classes 1 and 2 mostly because they have fewer examples than class 0. The F-Score results confirm that the precision and recall metrics are well balanced. To conclude, the precision scores were worst for classes 1 and 2, and the recall scores were worst for classes 0 and 1.

5.5 i-DREAMS Drivers Profile

This section details the experimental results for i-DREAMS drivers. We tested twenty different drivers with more than twenty trips to verify if their driver behavior was consistent over time. Table 5.17 shows the results, and Figure 5.14 provides the volatility histogram for a better understanding of the results.

The results reveal that the i-DREAMS drivers have a Non-Aggressive profile most of the time. From this sample, we can observe that most i-DREAMS drivers do not perform risky trips. Risky trips occur only in specific drivers that are an exception to the rest, like driver 18. To have a better understanding of the range values of the volatility referenced in 4.2, we calculated the maximum volatility value and found that it tends to stabilize around 1.09 regardless of the number of trips. Most of the volatility numbers lay between 0 and 0.2, and those values suggest that the driver's behavior is consistent over time. When the volatility value starts to increase beyond 0.2, we begin to identify inconsistent drivers. Drivers with a volatility value close to the maximum are considered very inconsistent. Drivers with volatility close to the 0.2 threshold are the most difficult to predict consistency. One may say, for example, that drivers 5 and 17 are consistent, and others say inconsistent. For these drivers, the number of trips is too short to be able to make assumptions. For this reason, we decided to use the following thresholds to tell if the driver has consistent or inconsistent behavior:

- If volatility <= 0.2, the driver is consistent
- If 0.2 < volatility <= 0.4, the driver is inconsistent
- If volatility > 0.4, the driver is very inconsistent

Table 5.17: i-DREAMS Drivers Profile, Volatility, number of Non-aggressive (NA), Aggressive (A), and Risky (R) trips

Driver	Profile	Volatility	NA	A	R
Test i-DREAMS driver 1	Non-Aggressive	0.209	78	28	0
Test i-DREAMS driver 2	Non-Aggressive	0.000	32	0	0
Test i-DREAMS driver 3	Non-Aggressive	0.142	82	9	0
Test i-DREAMS driver 4	Non-Aggressive	0.151	27	3	0
Test i-DREAMS driver 5	Non-Aggressive	0.224	28	9	0
Test i-DREAMS driver 6	Non-Aggressive	0.241	46	23	0
Test i-DREAMS driver 7	Non-Aggressive	0.211	51	13	0
Test i-DREAMS driver 8	Non-Aggressive	0.154	115	10	0
Test i-DREAMS driver 9	Non-Aggressive	0.225	21	6	0
Test i-DREAMS driver 10	Non-Aggressive	0.166	22	3	0
Test i-DREAMS driver 11	Non-Aggressive	0.116	94	4	0
Test i-DREAMS driver 12	Non-Aggressive	0.062	85	1	0
Test i-DREAMS driver 13	Non-Aggressive	0.098	34	1	0
Test i-DREAMS driver 14	Non-Aggressive	0.181	27	4	0
Test i-DREAMS driver 15	Non-Aggressive	0.148	105	9	0
Test i-DREAMS driver 16	Non-Aggressive	0.287	18	11	0
Test i-DREAMS driver 17	Non-Aggressive	0.208	33	6	0
Test i-DREAMS driver 18	Non-Aggressive	0.494	65	16	12
Test i-DREAMS driver 19	Non-Aggressive	0.065	77	1	0
Test i-DREAMS driver 20	Non-Aggressive	0.236	68	33	0

The thresholds when applied to this sample, cause eleven drivers to be consistent and the other nine inconsistent. To conclude, for the i-DREAMS drivers, the consistency calculation is mandatory because almost 50% of drivers (in this sample) have inconsistent behavior, making the final driver profile value not trustworthy in those cases.

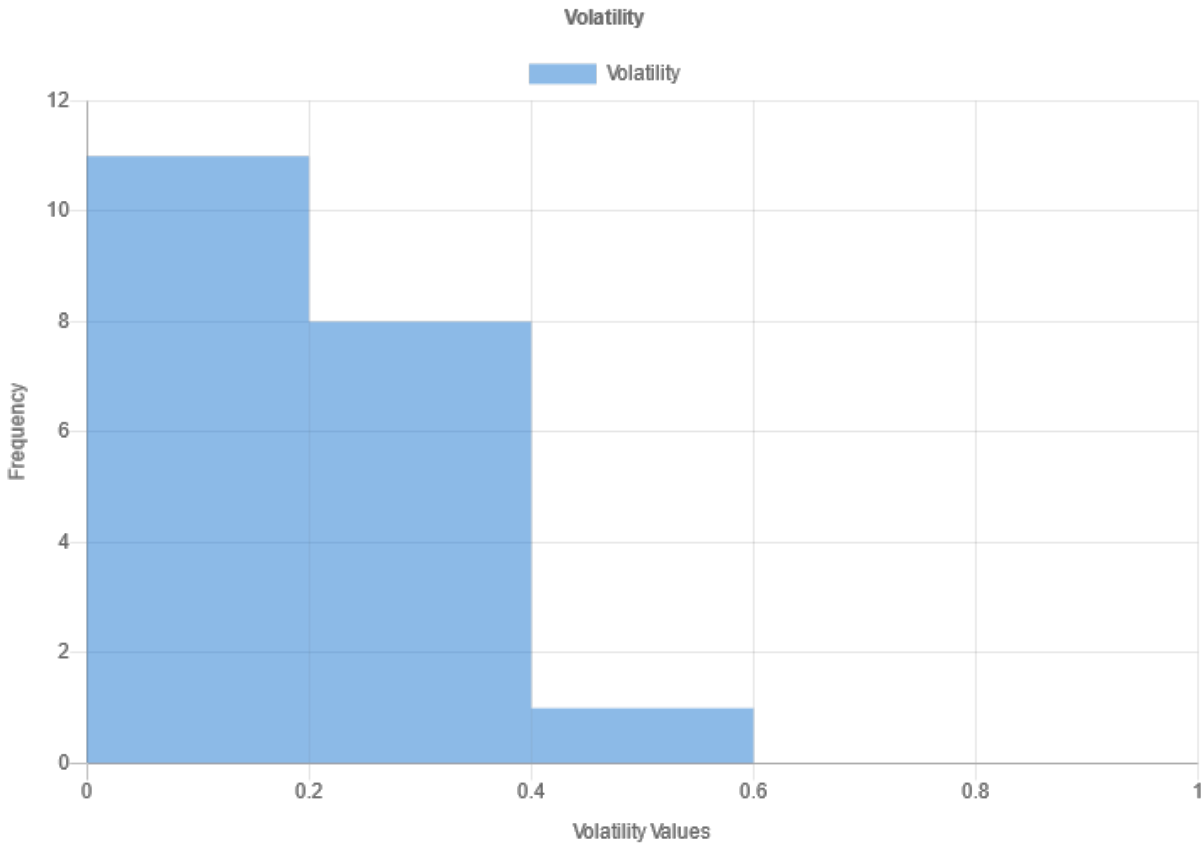


Figure 5.14: Volatility Histogram

6

Conclusions

This Chapter concludes this thesis. In Section 6.1, we detail the summary of the work accomplished, the difficulties, and final conclusions. Section 6.2 addresses the future work based on the conclusions.

6.1 Summary

This thesis aimed to provide an effective approach to driver profiling based on their behavior while driving and develop an API capable of establishing a driving profile for different drivers and fleets based on the i-DREAMS project data.

By analyzing different studies about driver profiling, we proposed an approach combining unsupervised learning and supervised learning techniques. The approach follows a typical machine learning problem, where initially we acquire data, then apply it to pre-process methods, and after, perform feature extraction. After these three steps, we apply unsupervised and supervised learning, finishing with evaluation.

The dataset construction, cleaning, and normalization phases were the ones that most influenced the clustering results. One of the problems that occurred the most while developing the thesis was that the normalization technique or feature set used revealed poor clustering, so we had to revise those phases several times. By normalizing the trips by distance and applying PCA we achieved the best results for unsupervised learning. We applied a two-stage clustering strategy, in which the K-Means algorithm

revealed to be the one with the best and most consistent results. The most challenging part of the clustering phase was to attribute meaning to the clusters, but the techniques used revealed to be sufficient. For supervised learning, we created a machine learning pipeline that performed normalization, oversampling, dimensionality reduction, and classification tasks. In the classification part, we applied different algorithms, and the SVM with linear kernel obtained the best scores (with a short difference) for all the distinct evaluation metrics.

Since the classification task only classified trips, we had to find a method to profile drivers based on the trip's profile. The method proposed for the i-DREAMS drivers fails to establish an accurate driver profile when the driver's behavior is inconsistent and succeeds when it is consistent.

Finally, we merged all the previous work in an API that is well structured and provides the most crucial functionalities for a driver and fleet profile system.

6.2 Future Work

To better understand the implications of these results, future work could address different clustering algorithms and take a deeper understanding and analysis of EAC with more data. With the use of more data, the models built in the supervised learning phase can be further tested to get a more robust evaluation. We should also take into consideration exploring different methods to determine the driver profile based on the trips profile. Additionally, it will be interesting to compare the experimental results of feature selection with those of feature reduction. The use of feature selection methods would provide the advantage to identify the most relevant original features. As for the API, new endpoints can be devised as needed.

References

[1] Haibo He, Yang Bai, Edwardo A. Garcia, and Shutao Li, «Adasyn: Adaptive synthetic sampling approach for imbalanced learning», in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, 2008, pages 1322–1328. DOI: 10.1109/IJCNN.2008.4633969.

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

[3] Loureiro, Luís and Ferreira, Artur and Lourenço, André, «Building a Dataset for Trip Style Assessment Based on Real Trip Data», *12th International Conference on Data Science, Technology and Applications, Rome, Italy*, Jul. 2023.

[4] ——, «On the development of a driver profile classification system», *i-ETC : ISEL Academic Journal of Electronics Telecommunications and Computers*, Oct. 2022.

[5] ——, «Finding Driver Styles on Driver Behavior Data with Unsupervised Learning», *Portuguese Conference on Pattern Recognition (RECPAD), Leiria, Portugal*, Oct. 2022.

[6] Subramanian Arumugam and R Bhargavi, «A survey on driving behavior analysis in usage based insurance using big data», *Journal of Big Data*, vol. 6, no. 1, pages 1–21, 2019.

[7] Luis M Bergasa, Javier Araluce, Eduardo Romera, Rafael Barea, Elena López-Guilén, Javier del Egido, and Carlos A Hernanz-Mayoral, «Naturalistic driving study for older drivers based on the drivesafe app», in *IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pages 1574–1579.

[8] Christopher M Bishop and Nasser M Nasrabadi, *Pattern recognition and machine learning*, 4. Springer, 2006, vol. 4.

[9] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao, «Borderline-smote: A new over-sampling method in imbalanced data sets learning», in *Advances in Intelligent Computing*, De-Shuang Huang, Xiao-Ping Zhang, and Guang-Bin Huang, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pages 878–887, ISBN: 978-3-540-31902-3.

[10] Tadeusz Caliński and Harabasz JA, «A dendrite method for cluster analysis», *Communications in Statistics - Theory and Methods*, vol. 3, pages 1–27, Jan. 1974. DOI: 10.1080/03610927408827101.

[11] CardioID. «Every heart has a beat, but the way we use it is unique.» (2021), [Online]. Available: https://www.cardio-id.com/. Accessed on 10 December 2021.

[12] Cher Carney, Dan McGehee, Karisa Harland, Madonna Weiss, and Mireille Raby, «Using naturalistic driving data to examine teen driver behaviors present in motor vehicle crashes, 2007-2015», *AAA Foundation for Traffic Safety*, Jun. 2016.

[13] German Castignani, Raphaël Frank, and Thomas Engel, «Driver behavior profiling using smartphones», in *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, pages 552–557.

[14] Kuan-Ting Chen and Huei-Yen Winnie Chen, «Driving style clustering using naturalistic driving data», *Transportation research record*, vol. 2673, no. 6, pages 176–188, 2019.

[15] Pushpa Choudhary and Nagendra R Velaga, «Effects of texting on accident risk during a sudden hazardous event: Analysis of predetection and postdetection phases», *Traffic injury prevention*, vol. 19, no. 8, pages 806–811, 2018.

[16] Barış Şimşek, Fatma Pakdil, Berna Dengiz, and Murat Caner Testik, «Driver performance appraisal using GPS terminal measurements: A conceptual framework», *Transportation research part C: emerging technologies*, vol. 26, pages 49–60, 2013.

[17] DATA. «12th international conference on data science, technology and applications». (2023), [Online]. Available: https://data.scitevents.org/. Accessed on 1 June 2023.

[18] David L. Davies and Donald W. Bouldin, «A cluster separation measure», *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, no. 2, pages 224–227, 1979. DOI: 10.1109/TPAMI.1979.4766909.

[19] Oussama Derbel and Rene Jr Landry, «Driving style assessment based on the GPS data and fuzzy inference systems», in *IEEE 12th International Multi-Conference on Systems, Signals & Devices (SSD15)*, 2015, pages 1–8.

[20] Danny Matthew SAPUTRA, Daniel SAPUTRA, and Liniyanti D. OSWARI, «Effect of distance metrics in determining k-value in k-means clustering using elbow and silhouette method», in *Proceedings of the Sriwijaya International Conference on Information Technology and Its Applications (SICONIAN 2019)*, Atlantis Press, 2020, pages 341–346, ISBN: 978-94-6252-963-2. DOI: https://doi.org/10.2991/aisr.k.200424.051. [Online]. Available: https://doi.org/10.2991/aisr.k.200424.051.

[21] Adrian B Ellison, Stephen P Greaves, and Michiel CJ Bliemer, «Driver behaviour profiles for road safety analysis», *Accident Analysis & Prevention*, vol. 76, pages 118–132, 2015.

[22] Rune Elvik, «Speed limits, enforcement, and health consequences», *Annual review of public health*, vol. 33, pages 225–238, 2012.

[23] Daniel Hutson. «How car insurance premiums are calculated». (Sep. 2021), [Online]. Available: https://www.comparethemarket.com/car-insurance/content/what-impacts-upon-your-car-insurance/. Accessed on 20 December 2022.

[24] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, *et al.*, «A density-based algorithm for discovering clusters in large spatial databases with noise.», in *Knowledge Discovery and Data Mining*, vol. 96, 1996, pages 226–231.

[25] «Flask». (Jun. 2022), [Online]. Available: https://flask.palletsprojects.com/en/2.2.x/. Accessed on 1 June 2022.

[26] Peter Handel, Isaac Skog, Johan Wahlstrom, Farid Bonawiede, Richard Welch, Jens Ohlsson, and Martin Ohlsson, «Insurance telematics: Opportunities and challenges with the smartphone solution», *IEEE Intelligent Transportation Systems Magazine*, vol. 6, no. 4, pages 57–70, 2014.

[27] John A Hartigan and Manchek A Wong, «Algorithm as 136: A k-means clustering algorithm», *Journal of the Royal Statistical Society. series c (applied statistics)*, vol. 28, no. 1, pages 100–108, 1979.

[28] Bing He, Dian Zhang, Siyuan Liu, Hao Liu, Dawei Han, and Lionel M Ni, «Profiling driver behavior for personalized insurance pricing and maximal profit», in *2018 IEEE International Conference on Big Data (Big Data)*, pages 1387–1396.

[29] Thodoris Garefalakis, Christos Katrakazas, and George Yannis, «Data-driven estimation of a driving safety tolerance zone using imbalanced machine learning», *Sensors*, vol. 22, no. 14, 2022, ISSN: 1424-8220. DOI: 10.3390/s22145309. [Online]. Available: https://www.mdpi.com/1424-8220/22/14/5309.

[30] i DREAMS Team. «A smart driver and road environment assessment and monitoring system». (2021), [Online]. Available: https://idreamsproject.eu/wp/. Accessed on 10 December 2022.

[31] Europen Comission. «Road safety». (Nov. 2021), [Online]. Available: https://ec.europa.eu/transport/road_safety/statistics-and-analysis/data-and-analysis/annual-statistical-report_en. Accessed on 20 December 2022.

[32] ISEL. «I-etc: Isel academic journal of electronics telecommunications and computers». (2022), [Online]. Available: http://www.journals.isel.pt/index.php/i-ETC. Accessed on 1 June 2022.

[33] Mary K Janke, «Accidents, mileage, and the exaggeration of risk», *Accident Analysis & Prevention*, vol. 23, no. 2-3, pages 183–188, 1991.

[34] Ian T Jolliffe, *Principal component analysis for special types of data*. Springer, 2002.

[35] Jungwook Jun, «Potential crash measures based on gps-observed driving behavior activity metrics», Ph.D. dissertation, Georgia Institute of Technology, 2006.

[36] Priya Pedamkar. «K- means clustering algorithm». (2022), [Online]. Available: https://www.educba.com/k-means-clustering-algorithm/. Accessed on 12 February 2022.

[37] Jordanka Kovaceva, Irene Isaksson-Hellman, and Nikolce Murgovski, «Identification of aggressive driving from naturalistic data in car-following situations», *Journal of Safety Research*, vol. 73, pages 225–234, 2020.

[38] Zhishuo Liu, Qianhui Shen, and Jingmiao Ma, «A driving behavior model evaluation for UBI», *International Journal of Crowd Science*, vol. 1, no. 3, pages 223–236, 2017.

[39] David Llopis-Castelló, Francisco Javier Camacho-Torregrosa, and Alfredo Garcia, «Analysis of the influence of geometric design consistency on vehicle CO_2 emissions», *Transportation Research Part D: Transport and Environment*, vol. 69, pages 40–50, 2019.

[40] Eleni G Mantouka, Emmanouil N Barmpounakis, and Eleni I Vlahogianni, «Identification of driving safety profiles from smartphone data using machine learning techniques», *Safety Science*, vol. 119, 2019.

[41] «Mariadb foundation». (Jun. 2022), [Online]. Available: https://mariadb.org/. Accessed on 1 June 2022.

[42] Rana Massoud, Francesco Bellotti, Riccardo Berta, Alessandro De Gloria, and Stefan Poslad, «Eco-driving profiling and behavioral shifts using IoT vehicular sensors combined with serious games», in *IEEE Conference on Games (CoG)*, 2019, pages 1–8.

[43] Mobileye. «Autonomous driving & ADAS (Advanced Driver Assistance Systems)». (2021), [Online]. Available: https://www.mobileye.com/. Accessed on 10 December 2022.

[44] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar, *Foundations of machine learning*. MIT press, 2018.

[45] Todd K Moon, «The expectation-maximization algorithm», *IEEE Signal processing magazine*, vol. 13, no. 6, pages 47–60, 1996.

[46] NationMaster. «Motor vehicle ownership, per 1000 inhabitants». (2014), [Online]. Available: https://ourworldindata.org/grapher/motor-vehicle-ownership-per-1000-inhabitants. Accessed on 20 December 2022.

[47] Oleg Okun, *Supervised and Unsupervised ensemble methods and their applications*. Springer, 2008, vol. 126.

[48] «Pandas - python dada analysis library». (Jan. 2022), [Online]. Available: https://pandas.pydata.org/. Accessed on 1 January 2022.

[49] Eleonora Papadimitriou, Anastasia Argyropoulou, Dimitrios I. Tselentis, and George Yannis, «Analysis of driver behaviour through smartphone data: The case of mobile phone use while driving», *Safety Science*, vol. 119, pages 91–97, 2019.

[50] Raymond C Peck and Jensen Kuan, «A statistical model of individual accident risk prediction using driver record, territory and other biographical factors», *Accident Analysis & Prevention*, vol. 15, no. 5, pages 371–393, 1983.

[51] Virginia Petraki, Apostolos Ziakopoulos, and George Yannis, «Combined impact of road and traffic characteristic on driver behavior using smartphone sensor data», *Accident Analysis & Prevention*, vol. 144, page 105 657, 2020.

[52] Lisa Precht, Andreas Keinath, and Josef F Krems, «Identifying effects of driving and secondary task demands, passenger presence, and driver characteristics on driving errors and traffic violations–using naturalistic driving data segments preceding both safety critical events and matched baselines», *Transportation research part F: traffic psychology and behaviour*, vol. 51, pages 103–144, 2017.

[53] «Python programming language». (Jan. 2022), [Online]. Available: https://www.python.org/. Accessed on 1 January 2022.

[54] RECPAD. «28th portuguese conference on pattern recognition». (2022), [Online]. Available: https://recpad2022.ipleiria.pt/. Accessed on 1 March 2023.

[55] Loureiro Luís. «Driver-profile-classification». (Jan. 2022), [Online]. Available: https://github.com/luisl12/Driver-Profile-Classification. Accessed on 1 January 2022.

[56] ——, «Driver-profile-api». (Jun. 2022), [Online]. Available: https://github.com/luisl12/Driver-Profile-API. Accessed on 1 June 2022.

[57] Douglas A Reynolds, «Gaussian mixture models.», *Encyclopedia of biometrics*, vol. 741, no. 659-663, 2009.

[58] Karen Rose, Scott Eldridge, and Lyman Chapin, «The internet of things: An overview», *The internet society (ISOC)*, vol. 80, pages 1–50, 2015.

[59] Fridulv Sagberg, Selpi, Giulio Francesco Bianchi Piccinini, and Johan Engström, «A review of research on driving styles and road safety», *Human factors*, vol. 57, no. 7, pages 1248–1275, 2015.

[60] Chalermpol Saiprasert, Suttipong Thajchayapong, Thunyasit Pholprasit, and Chularat Tanprasert, «Driver behaviour profiling using smartphone sensory data in a V2I environment», in *International Conference on Connected Vehicles and Expo (ICCVE)*, 2014, pages 552–557.

[61] Khaled Saleh, Mohammed Hossny, and Saeid Nahavandi, «Driving behavior classification based on sensor data fusion using LSTM recurrent neural networks», in *IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, 2017, pages 1–6.

[62] Claude Sammut and Geoffrey I Webb, *Encyclopedia of machine learning*. Springer Science & Business Media, 2011.

[63] Jörg Sander, Martin Ester, Hans-Peter Kriegel, and Xiaowei Xu, «Density-based clustering in spatial databases: The algorithm GDBSCAN and its applications», *Data mining and knowledge discovery*, vol. 2, no. 2, pages 169–194, 1998.

[64] «Scikit-learn - machine learning in python». (Jan. 2022), [Online]. Available: https://scikit-learn.org/stable/. Accessed on 1 January 2022.

[65] Peter J. Rousseeuw, «Silhouettes: A graphical aid to the interpretation and validation of cluster analysis», *Journal of Computational and Applied Mathematics*, vol. 20, pages 53–65, 1987, ISSN: 0377-0427. DOI: https://doi.org/10.1016/0377-0427(87)90125-7. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0377042787901257.

[66] Harpreet Singh and Ankit Kathuria, «Profiling drivers to assess safe and eco-driving behavior–a systematic review of naturalistic driving studies», *Accident Analysis & Prevention*, vol. 161, page 106 349, 2021.

[67] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer, «Smote: Synthetic minority over-sampling technique», *J. Artif. Int. Res.*, vol. 16, no. 1, 321–357, Jul. 2002, ISSN: 1076-9757.

[68] «Sqlalchemy - the database toolkit for python». (Jun. 2022), [Online]. Available: https://www.sqlalchemy.org/. Accessed on 1 June 2022.

[69] Gilbert Strang and Kai Borre, *Linear algebra, geodesy, and GPS*. Wellesley-Cambridge Press, 1997.

[70] Rohith Gandhi. «Support vector machine — introduction to machine learning algorithms». (Jul. 2018), [Online]. Available: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47. Accessed on 5 February 2022.

[71] Orit Taubman-Ben-Ari, Mario Mikulincer, and Omri Gillath, «The multidimensional driving style inventory—scale construct and validation», *Accident Analysis & Prevention*, vol. 36, no. 3, pages 323–332, 2004.

[72] Pete Thomas, Andrew Morris, Rachel Talbot, and Helen Fagerlind, «Identifying the causes of road crashes in Europe», *Annals of advances in automotive medicine*, vol. 57, page 13, 2013.

[73] Tomer Toledo, Oren Musicant, and Tsippy Lotan, «In-vehicle data recorders for monitoring and feedback on drivers' behavior», *Transportation Research Part C: Emerging Technologies*, vol. 16, no. 3, pages 320–331, 2008.

[74] «The uah-driveset». (Jan. 2022), [Online]. Available: https://www.robesafe.uah.es/personal/eduardo.romera/uah-driveset/#dataset. Accessed on 1 January 2022.

[75] Ingrid Van Schagen, Ruth Welsh, Agatha Backer-Grondahl, Marika Hoedemaeker, Tsippy Lotan, Andrew Morris, Fridulv Sagberg, and Martin Winkelbauer, «Towards a large scale european naturalistic driving study: Final report of prologue: Deliverable d4. 2», 2011.

[76] Minh Van Ly, Sujitha Martin, and Mohan M Trivedi, «Driver classification and driving style recognition using inertial sensors», in *IEEE Intelligent Vehicles Symposium (IV)*, 2013, pages 1040–1045.

[77] Eleni I Vlahogianni and Emmanouil N Barmpounakis, «Driving analytics using smartphones: Algorithms, comparisons and challenges», *Transportation Research Part C: Emerging Technologies*, vol. 79, pages 196–206, 2017.

[78] Chang Wang, Zhen Li, Rui Fu, Yingshi Guo, and Wei Yuan, «What is the difference in driver's lateral control ability during naturalistic distracted driving and normal driving? a case study on a real highway», *Accident Analysis & Prevention*, vol. 125, pages 98–105, 2019.

[79] Josh Warren, Jeff Lipkowitz, and Vadim Sokolov, «Clusters of driving behavior from observational smartphone data», *IEEE Intelligent Transportation Systems Magazine*, vol. 11, no. 3, pages 171–180, 2019.

Dataset Description

This Appendix describes and analyzes the features generated for the dataset.

Tables A.1, A.2, A.3, A.4, A.5, A.6, A.7, A.8, A.9, A.10, A.11, and A.12 indicate for each type of system built in the i-DREAMS project the features that were possible to generate.

Table A.13 provides the number of missing values and also the percentage of missing values for each feature.

Table A.1: Hands-On related features

Feature	Type	Description
n_lod_0	int	Number of events with no hands on the steering wheel
n_lod_1	int	Number of events with only the left hand on the steering wheel
n_lod_2	int	Number of events with only the right hand on the steering wheel
n_lod_3	int	Number of events with both hands on the steering wheel

Table A.2: Drowsiness related features

Feature	Type	Description
n_drowsiness_0	int	Number of drowsiness events level 0
n_drowsiness_1	int	Number of drowsiness events level 1
n_drowsiness_2	int	Number of drowsiness events level 2
n_drowsiness_3	int	Number of drowsiness events level 3

Table A.3: Driving Behavior related features

Feature	Type	Description
n_ha	int	Number of harsh acceleration events
n_ha_l	int	Number of harsh acceleration events with low severity
n_ha_m	int	Number of harsh acceleration events with medium severity
n_ha_h	int	Number of harsh acceleration events with high severity
n_hb	int	Number of harsh braking events
n_hb_l	int	Number of harsh braking events with low severity
n_hb_m	int	Number of harsh braking events with medium severity
n_hb_h	int	Number of harsh braking events high severity
n_hc	int	Number of harsh cornering events
n_hc_l	int	Number of harsh cornering events with low severity
n_hc_m	int	Number of harsh cornering events with medium severity
n_hc_h	int	Number of harsh cornering events with high severity

Table A.4: Distraction related features

Feature	Type	Description
distraction_time	float	Time spent distracted (seconds)
n_distractions	int	Number of distraction events

Table A.5: Ignition related features

Feature	Type	Description
n_ignition_on	int	Number of events where ignition was turned on
n_ignition_off	int	Number of events where ignition was turned off

Table A.6: Mobileye Advanced Warning System related features

Feature	Type	Description
fcw_time	float	Amount of time forward collision warning was active (seconds)
hmw_time	float	Amount of time headway monitoring was active (seconds)
ldw_time	float	Amount of time lane departure warning was active (seconds)
pcw_time	float	Amount of time pedestrian collision warning was active (seconds)
n_pedestrian_dz	int	Number times a pedestrian was detected in danger zone
light_mode	categorical	Trip Lighting condition (most frequent)
n_tsr_level	int	Number of times the speed limit was exceeded
n_tsr_level_0	int	Number of times the speed limit was not exceeded
n_tsr_level_1	int	Number of times 0-5 units over speed limit
n_tsr_level_2	int	Number of times 5-10 units over speed limit
n_tsr_level_3	int	Number of times 10-15 units over speed limit
n_tsr_level_4	int	Number of times 15-20 units over speed limit
n_tsr_level_5	int	Number of times 20-25 units over speed limit
n_tsr_level_6	int	Number of times 25-30 units over speed limit
n_tsr_level_7	int	Number of times 30+ units over speed limit
zero_speed_time	float	Amount of time the vehicle was stopped (seconds)
n_zero_speed	int	Number of times the vehicle stopped

Table A.7: Mobileye Car related features

Feature	Type	Description
n_high_beam	int	Number of times high beam is ON
n_low_beam	int	Number of times low beam is ON
n_wipers	int	Number of times wipers are ON
n_signal_right	int	Number of times right turn signal is ON
n_signal_left	int	Number of times left turn signal is ON
n_brakes	int	Number of times breaks are ON
speed	float	Mean Speed (km/h)

Table A.8: FCW, HMW, LDW, and PCW related features

Feature	Type	Description
n_fcw	int	Number of FCW events
n_hmw	int	Number of HMW events
n_ldw	int	Number of LDW events
n_ldw_left	int	Number of times right turn signal is ON
n_ldw_right	int	Number of times left turn signal is ON
n_pcw	int	Number of times breaks are ON

Table A.9: Fatigue related features

Feature	Type	Description
n_fatigue_0	int	Number of fatigue level 0 events, no warning
n_fatigue_1	int	Number of fatigue level 1 events, visual warning
n_fatigue_2	int	Number of fatigue level 2 events, visual and auditory warning
n_fatigue_3	int	Number of fatigue level 3 events, visual and auditory warning

Table A.10: Headway related features

Feature	Type	Description
n_headway__1	int	Number of headway level -1 events, no vehicle detected
n_headway_0	int	Number of headway level 0 events, vehicle detected
n_headway_1	int	Number of headway level 1 events, vehicle detected
n_headway_2	int	Number of headway level 2 events, first warning stage
n_headway_3	int	Number of headway level 3 events, second warning stage

Table A.11: Overtaking related features

Feature	Type	Description
n_overtaking_0	int	Number of overtaking level 0 events, no warning
n_overtaking_1	int	Number of overtaking level 1 events, visual warning
n_overtaking_2	int	Number of overtaking level 2 events, visual and auditory warning
n_overtaking_3	int	Number of overtaking level 3 events, frequent warning

Table A.12: Speeding related features

Feature	Type	Description
n_speeding_0	int	Number of overtaking level 0 events, no warning
n_speeding_1	int	Number of overtaking level 1 events, visual warning
n_speeding_2	int	Number of overtaking level 2 events, visual and auditory warning
n_speeding_3	int	Number of overtaking level 3 events, frequent warning

Table A.13: Missing Values

Feature	Number	Percentage
n_ha	1301	7.59%
n_ha_l	1301	7.59%
n_ha_m	1301	7.59%
n_ha_h	1301	7.59%
n_hb	1301	7.59%
n_hb_l	1301	7.59%
n_hb_m	1301	7.59%
n_hb_h	1301	7.59%
n_hc	1301	7.59%
n_hc_l	1301	7.59%
n_hc_m	1301	7.59%
n_hc_h	1301	7.59%
n_ignition_on	485	2.83%
n_ignition_off	485	2.83%
fcw_time	635	3.71%
hmw_time	635	3.71%
ldw_time	635	3.71%
pcw_time	635	3.71%
n_pedestrian_dz	635	3.71%
light_mode	635	3.71%
n_tsr_level	635	3.71%
n_tsr_level_0	635	3.71%
n_tsr_level_1	635	3.71%
n_tsr_level_2	635	3.71%
n_tsr_level_3	635	3.71%
n_tsr_level_4	635	3.71%
n_tsr_level_5	635	3.71%
n_tsr_level_6	635	3.71%
n_tsr_level_7	635	3.71%
zero_speed_time	635	3.71%
n_zero_speed	635	3.71%

Feature	Number	Percentage
n_high_beam	635	3.71%
n_low_beam	635	3.71%
n_wipers	635	3.71%
n_signal_right	635	3.71%
n_signal_left	635	3.71%
n_brakes	635	3.71%
speed	635	3.71%
n_fcw	1057	6.20%
n_hmw	1057	6.20%
n_ldw	1057	6.20%
n_ldw_left	1057	6.20%
n_ldw_right	1057	6.20%
n_pcw	1057	6.20%
n_fatigue_0	654	3.82%
n_fatigue_1	654	3.82%
n_fatigue_2	654	3.82%
n_fatigue_3	654	3.82%
n_headway__1	639	3.73%
n_headway_0	639	3.73%
n_headway_1	639	3.73%
n_headway_2	639	3.73%
n_headway_3	639	3.73%
n_overtaking_0	639	3.73%
n_overtaking_1	639	3.73%
n_overtaking_2	639	3.73%
n_overtaking_3	639	3.73%
n_speeding_0	645	3.76%
n_speeding_1	645	3.76%
n_speeding_2	645	3.76%
n_speeding_3	645	3.76%
distraction_time	16966	98.99%
n_distractions	16966	98.99%

B

Feature Distribution

This Appendix provides figures that prove that the features data is skewed. We use box plots and distribution plots to analyze the data because they give a good indication of how the values in the data are spread out, and help find outliers. Figures B.1, B.2, B.3, B.4, B.5, and B.6 show that there are several or large numbers of data points that act as outliers, and also indicate that the data is right-skewed for most features except the speed where the data has almost a symmetric distribution. Outlier's data points will have a significant impact on the mean, and hence, in such cases, it is not recommended to use the mean for replacing the missing values, that is why we used the median.

Additional images for other features are provided in the driver profile classification repository [55].

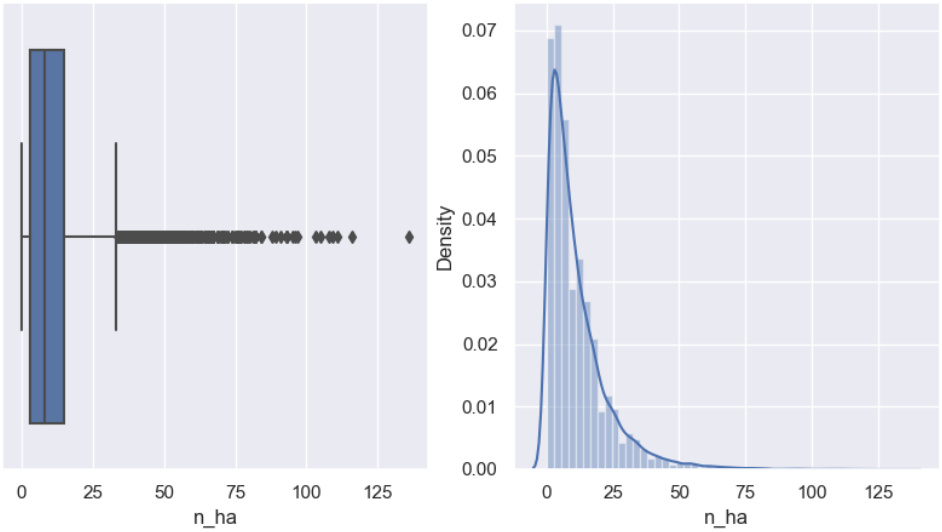


Figure B.1: Number of harsh accelerations distribution

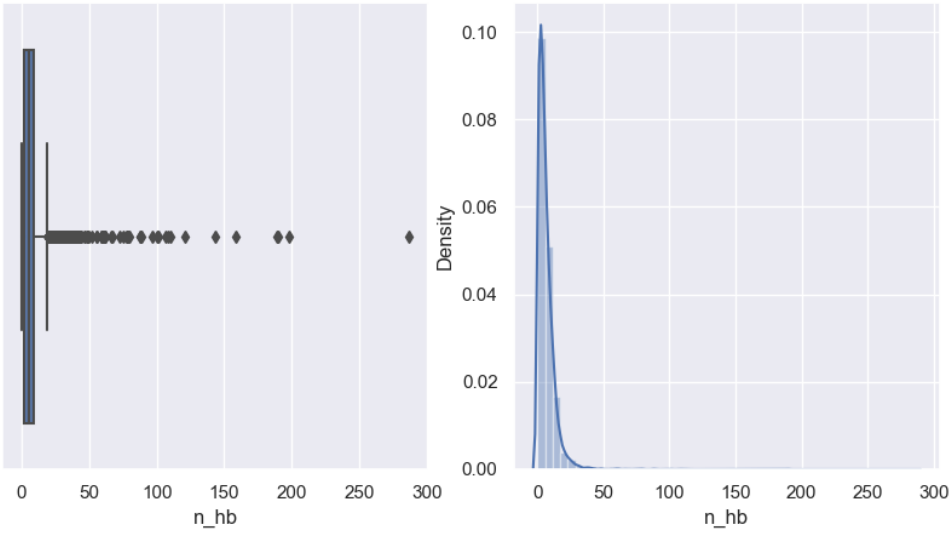


Figure B.2: Number of harsh breaking distribution

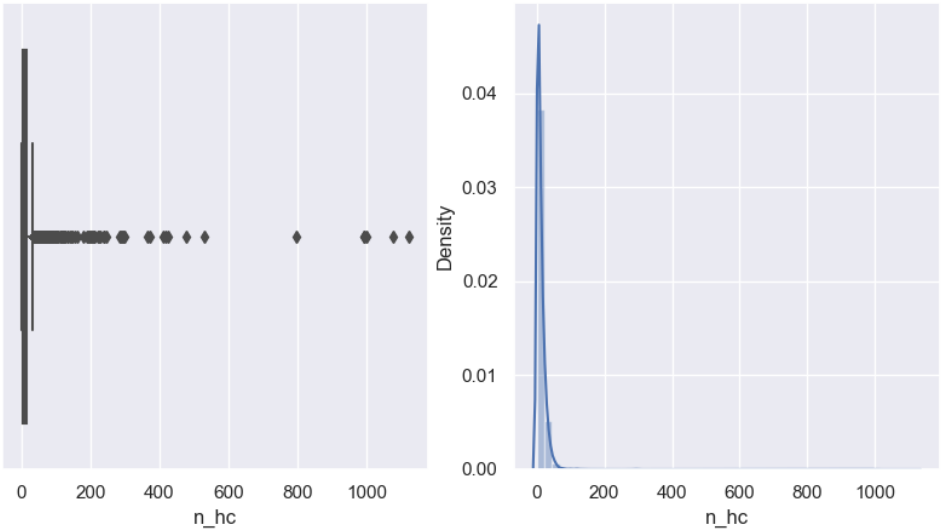


Figure B.3: Number of harsh cornering distribution

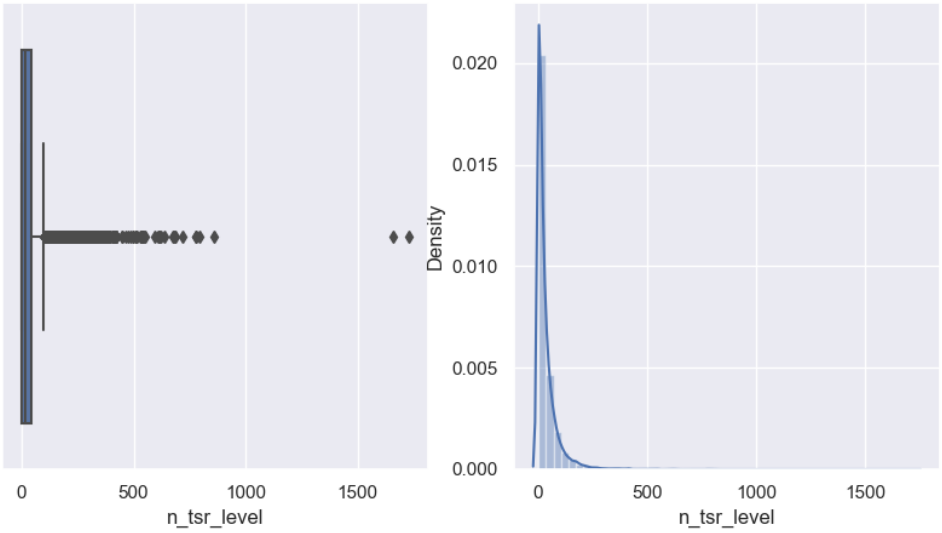


Figure B.4: Number of speeding events distribution

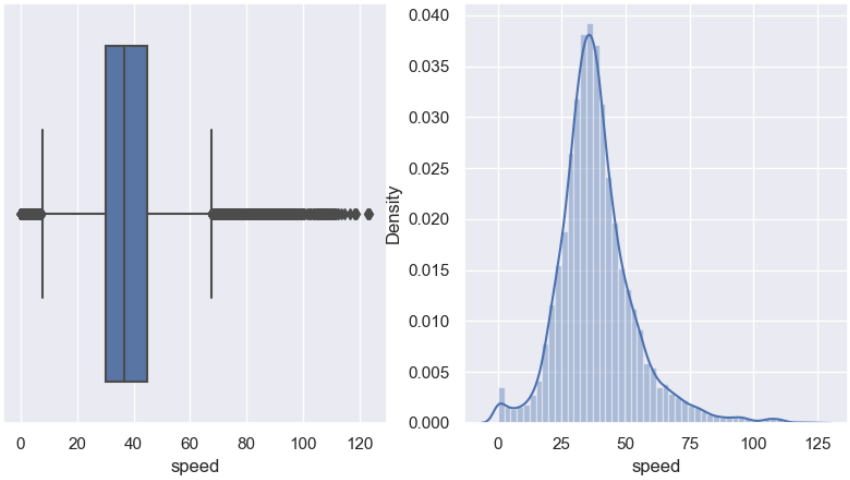


Figure B.5: Speed mean distribution

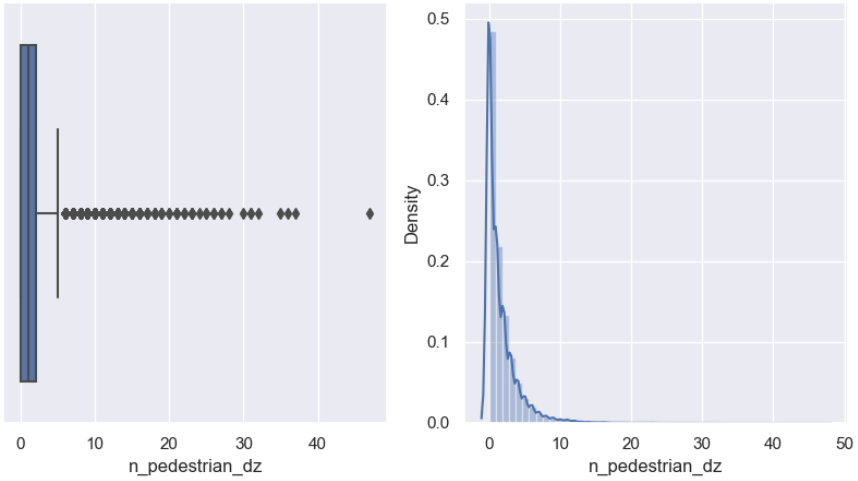


Figure B.6: Number of times a pedestrian was detected in danger zone distribution

Parameter Tunning Grids

This Appendix provides the grids used for parameter tuning over the pipeline. The "clf" prefix is used for classifier parameters whereas the "over" prefix is used for the ADASYN parameters. The XGBoost and SVM don't use ADASYN n_neighbors parameter because it takes too long to obtain results and, they have already good performance without it.

- Decision Tree
  - 'clf__criterion': ['gini', 'entropy']
  - 'clf__max_depth': [6, 8, 10, 12]
  - 'clf__min_samples_split': [0.05, 0.1, 0.15, 0.2]
  - 'clf__min_samples_leaf': [0.05, 0.1, 0.15, 0.2]
  - 'over__n_neighbors': [4, 5, 6, 8]
- Random Forest
  - 'clf__n_estimators': [100, 150, 200, 250]
  - 'clf__criterion': ['gini', 'entropy']
  - 'clf__min_samples_split': [0.02, 0.05, 0.1, 0.15, 0.2]
  - 'over__n_neighbors': [4, 5, 6, 8]
- XGBoost

  - 'clf__n_estimators': [150, 200, 250]
  - 'clf__max_depth': [6, 8]
  - 'clf__learning_rate': [0.01, 0.05, 0.1]
  - 'clf__colsample_bytree': [0.3, 0.5, 0.7]

- SVM with linear kernel
  - 'clf__C': [0.001, 0.01, 0.1, 1, 10, 100]