FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Fatigue State Estimator in Car Driving

Miguel Ângelo Coelho dos Santos

Mestrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Rui Esteves Araújo

October 17, 2023

Resumo

A fadiga na condução é uma preocupação significativa em termos de segurança rodoviária, representando um risco considerável para os utilizadores das vias. Estudos estimam que entre 10% a 20% dos acidentes rodoviários resultam da fadiga, a qual, gera redução da atenção, tempos de reação maiores e menor capacidade para tomada de decisões.

A dissertação inicia-se com uma revisão abrangente da literatura, explorando a modelação e o reconhecimento do comportamento do condutor com base em dados de condução e técnicas para a deteção e estimação do estado de sonolência do condutor. Diversos métodos, incluindo máquinas de vetores de suporte, abordagens bayesianas não paramétricas e redes neuronais artificiais são investigadas para a classificação do estilo de condução e a deteção de sonolência na condução.

A grande maioria das abordagens descritas na literatura da especialidade assentam no processamento de sinais provenientes de sensores adicionais ao funcionamento do automóvel. Neste trabalho procurou-se averiguar a possibilidade de inferir sobre o estado de sonolência utilizando apenas os sinais que caracterizam o movimento do veículo e que se encontram disponíveis via porto *On-Board Diagnostics*.

Foi desenvolvida uma solução técnica assente no processamento de dados relativos à velocidade, aceleração e posição do acelerador. O algoritmo de estimação do estado de fadiga é baseado na análise de agrupamento de dados através do algoritmo *K-means*. Face aos erros e inconsistência dos dados relativos à caracterização da posição do acelerador, o algoritmo foi adaptado para operar sem esses dados, recorrendo ao conceito de *features engineering* para maximizar o uso dos dados disponíveis para a estimação da fadiga,

Através da implementação do algoritmo desenvolvido, o estudo apresenta resultados iniciais que apontam para a possibilidade de caracterizar a fadiga com base nos padrões dos dados do movimento do automóvel. Dada a escassez de dados em bases de dados públicas, o método desenvolvido foi testado apenas de forma preliminar, tendo mesmo assim, obtido informações úteis para o projeto do estimador de fadiga.

Palavras-chave: Fadiga na condução, análise de agrupamento de dados, aprendizagem computacional, segurança no transporte, segurança rodoviária.

Abstract

Fatigue in driving is a significant concern in terms of road safety, representing a considerable risk to road users. Studies estimate that between 10% to 20% of road accidents result from fatigue, which leads to reduced attention, longer reaction times, and decreased decision-making ability.

The dissertation begins with a comprehensive literature review, exploring the modeling and recognition of driver behavior based on driving data, and techniques for detecting and estimating the driver's drowsiness state. Various methods, including support vector machines, non-parametric bayesian approaches, and artificial neural networks, are investigated for classifying driving style and detecting drowsiness while driving.

The vast majority of approaches described in the literature rely on processing signals from sensors in addition to the vehicle's operation. In this work, we sought to investigate the possibility of inferring the drowsiness state using only the signals that characterize the vehicle's motion and are available via On-Board Diagnostics.

A technical solution was developed based on the processing of data related to speed, acceleration, and accelerator position. The fatigue estimation algorithm is based on data clustering analysis using the K-means algorithm. Due to errors and inconsistencies in the accelerator position data, the algorithm was adapted to operate without this data, employing the concept of feature engineering to maximize the use of the available data for fatigue estimation.

Through the implementation of the developed algorithm, the study presents preliminary results that suggest the possibility of characterizing fatigue based on the patterns of vehicle motion data. Given the scarcity of data in public databases, the developed method was tested only in a preliminary manner, yet still yielded valuable insights for the fatigue estimator project.

Keywords: Driving fatigue, data clustering analysis, machine learning, transportation safety, road safety.

Acknowledgments

I would like to express my sincere gratitude to all the individuals and institutions who made this work possible and contributed to my academic growth. Without the support and involvement of each one of you, this achievement would not be possible.

First and foremost, I would like to thank my advisor, Dr. Rui Esteves Araújo, for his guidance throughout the entire research process. His dedication, patience, and insight were crucial factors in conducting this study.

I am grateful to the Faculdade de Engenharia da Universidade do Porto for the institutional support provided, as well as access to the resources and infrastructure necessary for carrying out this research. I am thankful to all the professors who contributed to my education throughout the course.

I would like to thank the participants of this study, whose active participation and collaboration were essential for obtaining the necessary data. I also appreciate those who granted permission for the use of research instruments and additional resources.

Finally, I express my gratitude to my friends and family for the love, support, and patience they have shown me during this period. Their words of encouragement and motivation were crucial in maintaining my drive during the difficulties encountered along the way.

Once again, to all those who have in any way contributed to the realization of this project, my sincere thanks. This achievement is also yours.

Miguel Ângelo Coelho dos Santos

"Success is not final, failure is not fatal:
It is the courage to continue that counts"

Winston Churchill

Contents

List of Figures

List of Tables

Abbreviations and Symbols

ANN	Artificial Neural Network
DCT	Discrete Cosine Transform
DP	Dirichlet Process
ECG	Electrocardiogram
EEG	Electroencephalogram
EER	Equal Error Rate
EOG	Electrooculogram
GBT	Gradient Boosting Tree
GMM-UBM	Gaussian Mixture Models - Universal Background Models
HDP	Hierarchical Dirichlet Process
HMM	Hidden Markov Model
HR	Heart Ratio
HRV	Heart Rate Variability
HSMM	Hidden Semi-Markov Model
IDR	Identification Rate
KSS	Karolinska Spleepiness Scale
kNN	k-Nearest Neighbors
MAF	Moving Average Filter
MC	Machine Learning
MLP	Multilayer Perceptrons
MSE	Mean Square Error
NCCC	Normalized Cross-Correlation Clustering
NDD	Naturalistic Driving Data
OBD	On-board Diagnostics
PPG	Pulse Plethysmography
PSD	Power Spectral Density
PVE	Peak-valley Extraction
RBF	Radial Basis Function
RF	Random Forest
ReLU	Rectified Linear Unit
SG	Savitzky - Golay
S3VM	Semi-supervised Vector Machine
STD	Standard Deviation
SVM	Support Vector Machine
WCSS	Within-Cluster Sum of Square

Chapter 1

Introduction

The fatigue is a critical factor that significantly affects the state of driving and poses substantial risks to road safety. It is a common issue that arises from prolonged driving, lack of adequate rest and monotonous driving conditions. Fatigued drivers experience reduced attention, impaired reaction times and diminished decision-making abilities, all of which can lead to an increased likelihood of accidents.

The consequences of a fatigued driver can be severe resulting in serious injuries and fatalities on the roads. As driver becomes drowsy or falls asleep behind the wheel, its ability to maintain the vehicle control diminishes leading to potential collisions with other vehicles or objects on their path. Moreover, fatigue-related accidents often occur without any prior warning signs making it a significant challenge for both drivers and road authorities to mitigate the risks effectively.

1.1 Context

Recognizing the importance of addressing driver fatigue in transportation safety, attending that 10% to 20% of road accidents [1] are a direct consequence of this condition, researchers and experts have been exploring various approaches to detect and prevent fatigue-related incidents. Analyzing driving behavior using advanced technologies and data-driven methodologies has emerged as a promising avenue to identify signs of fatigue and intervene timely.

This introductory chapter aims to provide an overview of the context in which the dissertation is situated. The study focuses on analyzing driver behaviour based on driving data, addressing issues related to fatigue events and methods capable of drowsiness detection. The transportation and safety domain are relevant areas of research, as driver fatigue and road safety have a direct impact on accident prevention and the enhancement of driving efficiency and comfort.

1.2 Objectives

The main objectives of this dissertation are:

- Investigate driver drowsiness detection and prediction;
- Develop and apply classification methodologies for driving fatigue based on machine learning techniques;
- Design and implement a code that integrates the studied techniques and methodologies, enabling reliable and relevant results;
- Present the results obtained demonstrating the effectiveness of the proposed methodologies in drowsiness detection.

1.3 Dissertation Structure

In addition to the introduction, this dissertation consists of 4 more chapters:

- **Chapter 2** - Literature Review

  This chapter provides a comprehensive literature review related to driver behavior and driver fatigue based on driving data. It covers relevant studies on modelling and recognizing driver behaviour, including research on driving style classification, driver drowsiness detection and biometric systems based on ECG;

- **Chapter 3** - Technical Solution Design

  This chapter presents the detailed design of the proposed technical solution. It outlines the initial requirements of the system including the selection of relevant variables. Additionally, it establishes the hypotheses to be tested and the methodology to be used, such as the implementation of Supervised Artificial Neural Networks, the use of the K-means algorithm for data clustering and features engineering;

- **Chapter 4** - Framework, Source Code and Results

  In this chapter, the developed code is presented, which integrates the different stages of the technical solution. Detailed information about the source code is provided, highlighting how each stage of the process was implemented. Furthermore, the results obtained are presented;

- **Chapter 5** - Conclusion and Future Work

  This chapter summarizes the main conclusions reached in the survey and highlights its contributions to the road safety. It also suggests topics for future work and improvements in the developed methodologies.

Chapter 2

Literature Review

2.1 Introduction

This chapter contains the analysis of the state of the art. It is a crucial part for the developed work because it contains theoretical analysis of driving styles, which is very important to know what type of driver we are studying and then evaluate the state of drowsiness that he can present (that can be different according to the drive style of the driver).

This chapter will explore algorithms designed for the analysis and assessment of somnolence, alongside a comprehensive examination of data analysis techniques and the development of an algorithm for data labeling. This will encompass critical considerations regarding the adequacy of data volume, as well as the elucidation of clustering methodologies for driving datasets.

From all of the analyzed papers, four of them were selected as the most representative of the state of art. This chapter will present the main techniques, methodologies, and results crucial for the development of the presented dissertation.

2.2 Modeling and Recognizing Driver Behavior Based on Driving Data: A Survey

Driver model research [2] has been made from the perspective of vehicle dynamics application and human factors. Considerable strides have been achieved through an extensive array of specialized investigations pertaining to various dimensions of human physiology and psychology, involving the acquisition of biological data. The output of this models are normally steering wheel angle/torque, acceleration or brake pedal position/pressure and the gear shift position. This paper aims to present methods of recognizing driver's characteristics and modeling driver's driving behavior/skill/state.

The main problem on this kind of work is that driving a car is a complex and dynamic task that requires not only accurate perceptions and cognitions from the driver, but also to process all the information (driving skill, driver state, vehicle performance and traffic) at a high rate of speed.

Modeling and recognizing the driver behavior or driving skill can be classified into four steps:

1. ***Modeling Driver Behavior:*** The model structure can be established, and parameters of the driver model can be identified based on human driving behavior, which might be classified roughly into three cases: parameter identification, nonparameter identification, and semiparameter identification;

2. ***Recognizing the Characteristics of Driver Behavior:*** After driving model has been determined, the driver behavior or driver's driving skill should be characterized. Here, many driving tasks or situations (such as car following, lane change, collision avoidance) are described with numerous mathematical methods adopted;

3. ***Evaluating and Verifying Based on the Driver Model:*** The objectives of identification followed by modeling of driver behavior are meant to improve the performance of vehicle dynamics and to design more intelligent driver systems. Therefore, the efficiency of the driver model needs to be evaluated and verified, especially in the field of handling quality and driver assistance systems;

4. ***Embedding Driver Characteristics into the Advanced Vehicle Systems:*** Producing more intelligent vehicle-driver systems is always the engineer's goal during the design process. Consequently, driver assistance systems that can timely accurately detect and predict the driver's attention and seamlessly integrate with the driver's characteristics are crucial.

Driver modeling is the simplification of the human driver and can represent the delay and physical characteristics of him, but the model has uncertainty and nonlinear characteristics. Considering that uncertainty both within individual driver and across different drivers, the uncertainty modeling of driver steering control behavior is addressed and the driver model is treated as a black box, where the input and output are lateral deviation from the center line of the road and the steering wheel angle. Advanced mathematical methods and advanced control theory could be done. For example, more nonlinear mathematical models can be used for characterizing the nonlinear driving behavior.

Human driving characteristics are presented from the control perspective in terms of human behavior activities, such as driver distraction, side-tasking, and driver impairments, and human limitation. They can be categorized into four groups:

1. ***Human Time Delay and Threshold Limitations:*** Humans can be treated as a nonlinear system with time delay and sense limitation, that can be different for individual drivers;

2. ***Visual Characteristics:*** Vision system could not capture the velocity and position information accurately;

3. ***Motion Influence:*** Due to the influence of vestibular, experience and/or skill level may also play a crucial role in a human-vehicle system;

4. ***Tactile and Haptic Information:*** Tactile and haptic information, such as steering wheel torque and the pedal position, conveyed through the steering wheel and throttle or acceleration pedals.

After obtaining the mathematical models, having all this into consideration, the Figure 2.1 is presented as an algorithm for identifying a driving model:

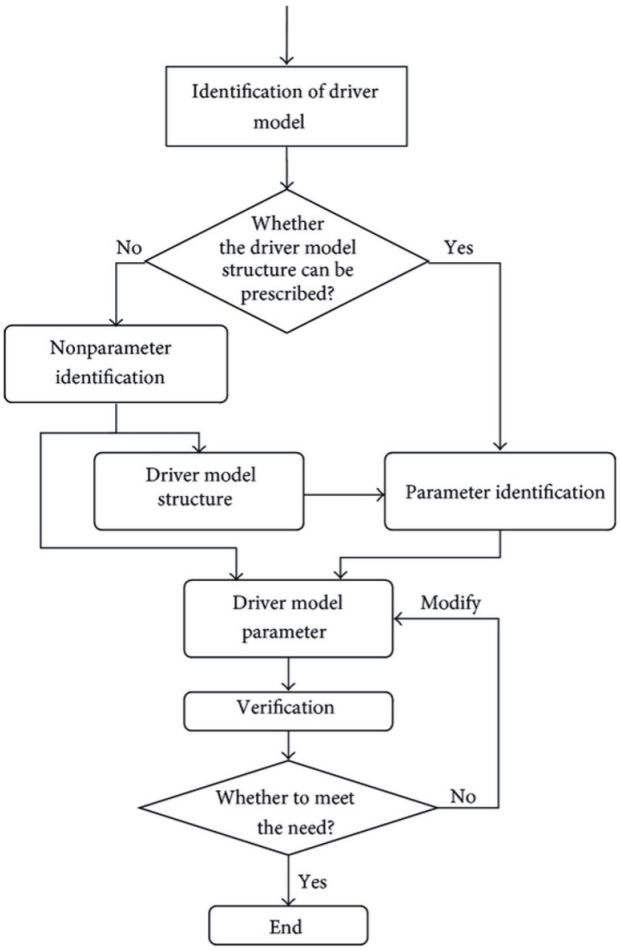


Figure 2.1: The flow diagram of driver model identification [2].

The [2] reveals a wide range of mathematical methods of modeling and recognition of driver characteristics which can be used to improve vehicle's dynamics performance, decrease driver's workload, and develop more intelligent driver assistance systems. The reviewed articles show that numerous driver models have been developed from different perspectives by using various identification methods. Therefore, some issues still exist, for example, which parameters and driving situations are more sensitive to driver's behavior/state/skill, that is, how to characterize the human driver more exactly, easily, and quickly? That is not the main goal of this dissertation but this analysis was useful to attend the prediction of driving behavior and understand driver's limitations and that can be useful to predict states of fatigue/drowsiness on driving.

2.3 Driving Style Classification Using a Semi-Supervised Support Vector

Driving style classification [3] has a vital role in a large variety of realms such as human-centric vehicle control systems, intelligent transportation systems, road safety and power management for electric vehicles. The National Highway Traffic Safety Administration research concludes that more than 23 percent of deaths in traffic accidents are related to driving styles in the United States.

An indirect way for understanding driving styles is based on an appropriate driver model. Another way is to directly analyze driving styles using pattern classification techniques based on experimental driving data. Techniques for classifying driving styles in the literature can be roughly divided into supervised methods and unsupervised methods as shown in Figure 2.2.

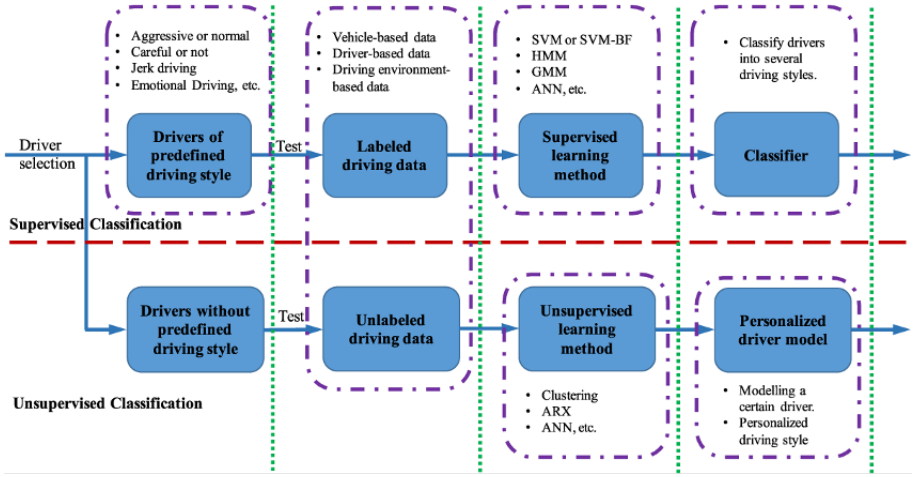


Figure 2.2: Schematic diagram of supervised and unsupervised methods for driving style classification [3].

The [3] shows a use of a support vector machine (SVM) to recognize driving styles based on the labeled information of the vehicle's inertial sensors.

Training data was collected from drivers, who were prelabeled with six driving patterns using rule-based approaches according to data analysts prior knowledge. However, this kind of method has drawbacks as:

- Manually labeling large amount of training data requires excessive effort;
- Manually labeling driving data can also cause subjective labels;
- Data analysts should be very familiar with driver behaviors.

In order to enhance the performance of driving style classification and reduce labeling efforts, a semi-supervised method, semi-supervised support vector machine (S3VM), was developed by combining the advantages of supervised and unsupervised methods. The S3VM is able to solve the pattern classification problem with a very small amount of labeled data. First, the driving data are

processed using a k-means clustering method, and then, a few distinct data clusters are selected and labeled using a rule-based approach. Second, the S3VM approach generates an optimal decision boundary by fully using the knowledge of the unlabeled data and labeled data. To solve the nonconvex optimization, it is introduced a differentiable surrogate of the loss function and a quasi-Newton (QN) algorithm, which makes it feasible to employ known optimization tools.

The S3VM method was able to generate a more objective decision boundary, compared to rule-based methods, especially for cases where the decision boundaries between two driving styles are not clear. S3VM also requires a very small amount of labeled data and hence can significantly reduce the labelling effort for training a classifier.

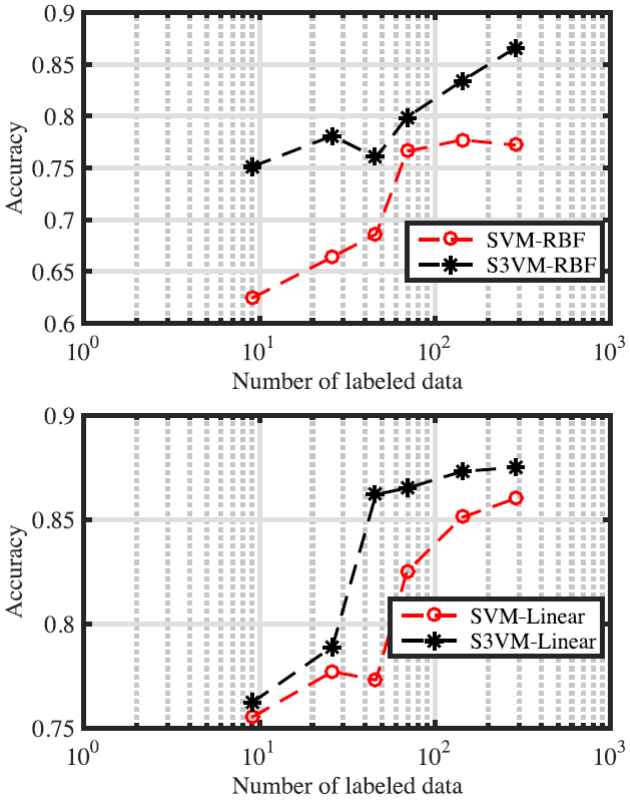


Figure 2.3: Classification accuracy of using S3VM and SVM based on an RBF kernel and a linear kernel [3].

As it can be seen in Figure 2.3, the S3VM generally performs better than SVM and as expected the performance of S3VM converges to a constant when more labeled data are utilized. S3VM can utilize unlabeled data information to obtain a well-trained classifier, which is not possible with the SVM method. In summary, S3VM shows better performance than SVM for driving style classifications where there are limited labeled data and a large amount of unlabeled data.

This review as an extremely important information about the driving style classification proving that S3VM method has more advantages comparing to the SVM method. It can be very useful to classify the level of fatigue/drowsiness on the drivers, which is one of the main goals of this dissertation.

2.4 How Much Data is Enough? A Statistical Approach with Case Study on Longitudinal Driving Behavior

Big data [4] has shown its uniquely powerful ability to reveal, model, and understand driver behaviors. The amount of data effects the experiment cost and conclusions in the analysis. Insufficient data may lead to inaccurate models while excessive data waste resources. This paper systematically investigates this issue to estimate how much naturalistic driving data (NDD) is needed for understanding driver behaviors from a statistical point of view.

The required amount of NDD depends on the problem to be solved, the way the problem is formulated, and the dataset to be analyzed, can be seen on Table 2.1

Project name	Conductor	Period	Mileage [mile]	Vehicle	Sensor	Drivers	Research topic
100 Car Naturalistic Driving Study [6]	Virginia Tech.	2001–2009	2×10^6	100 sedans	camera	109 primary drivers, 132 secondary drivers	Rear end collision
Automotive Collision Avoidance System [13]	University of Michigan	2004-2005	1.37×10^5	11 sedans	camera, radar	96 drivers	Forward collision warning (FCW)
Road Departure Crash Warning [14]	University of Michigan	2005–2006	8.3×10^4	11 sedans	camera, radar	11 drivers	Lane departure warning (LDW)
Sweden-Michigan Naturalistic Field Operational Test (SeMiFOT) [15]	University of Michigan	2008–2009	1.07×10^5	10 sedans, 4 trucks	camera, radar	39 drivers	FCW, LDW, blind spot information system, electronic stability control, and impairment warning
Integrated Vehicle-Based Safety Systems [16]	University of Michigan	2010–2011	sedans: 213&309; trucks: 601&944	16 sedans 10 heavy trucks	camera, radar	108 drivers for sedans; 18 professional truck drivers	Integrated warning
Safety Pilot Model Deployment [17]	University of Michigan	2012–2014	more than 3.4×10^7	2,800 various types of vehicles	camera, radar	2,700 volunteer drivers and several professional bus and truck drivers	Connected vehicle
Google driverless car [18]	Google	2012–present	more than 1.3×10^6	At least 50 sedans and SUVs	lidar, camera, radar	Google technicians and volunteers	Fully self-driven vehicle
Australian Naturalistic Driving Study or Australian 400-car Naturalistic Driving Study [19], [20]	Led by University of New South Wales	2015–present	4 months	400 vehicles	camera, CAN data, GPS	360 participants (180 in New South Wales and 180 in Victoria)	Safety at intersections; Speed choice; Interactions with vulnerable road users; Fatigue; Distraction and inattention; Crashes and near-crashes; Interactions with ITS
European naturalistic Driving and Riding for Infrastructure & Vehicle safety and Environment(UDRIVE) [21]	the 7th EU Framework Programme and 20 partners	2012–2017	On going	200 vehicles (cars, trucks, and scooters)	cameras, IMU sensors, GPS, Mobil Eye smart camera, CAN data, and Sound level	On going	Crash causation and risk; Everyday driving; Distraction and inattention; Vulnerable road users; Eco-driving
China Naturalistic Driving Study	Tongji University; VTTI; General Motors	2012–2015	more than 1.0×10^5	5 vehicles	–	90 drivers; each drove vehicle for 2 months	Exploring Chinese moped-vehicle conflict configurations; Examining car driver responses to moped-vehicle conflicts
Japan Naturalistic Driving Study [22]	Ministry of Land, Infrastructure, Transport and Tourism	2006–2008	–	60 vehicles (35 wagons & 25 sedans)	GPS, CAN data, acceleration sensor, camera	60 drivers (58 males & 2 females)	Accident causation research

Table 2.1: Major projects of naturalistic driving study in the world [4].

Modeling driver behaviors covers a wide range of topics, including, car-following, lane change, left/right turn, U-turn, distraction/inattention, secondary tasks, or brake behaviors. Data for modeling driver behaviors ranges widely from under 50 minutes to more than 5000 minutes.

In [4], it's presented and analyzed the reasons for these big differences. To facilitate the discussion and analysis, it's only used the car-following behaviors as an example, because car-following behavior is the most common event in driver behaviors.

Problem Formulation Methods:

The approach to formulating problems can result in diversity in the amount of NDD.

Modeling and analyzing driver's car-following behaviors, generally involves either a ***physically-based*** or a ***learning-based*** method.

- **Physically-based methods:** Usually describes driver behavior in the form of equations with physical meanings, in which parameters are used to fit the individual driver's characteristics via parameter estimation or calibration methods. The requisite amount of data depends on a number of unknown parameters. A physical model with many unknown parameters requires more driving data to fit driver behaviors. In addition, the amount of required data also depends on the method used to calibrate car-following models. For example, a calibration method using statistical techniques usually requires more data than that without considering the statistical features;

- **Learning-based methods:** Utilize machine learning techniques, without considering the physical meaning of the model parameters, to describe more complex and underlying non-linear relationships between different kinds of driver behaviors. Due to the complexity and diversity of drivers' car-following behaviors, it's difficult to capture stochastic features of drivers using physically-based model. A learning-based method is therefore introduced to solve these kinds of issues. For example, neural networks have been applied to modeling, analyzing, and characterizing driver behaviors.

Data Collection Approaches:

The approach to collecting driving data varies across research topics. Past data collection approaches included: **in-vehicle sensor data and video/camera data with a fixed field.**

- **In-vehicle sensor data:** In-vehicle sensor data, as the example of Figure 2.4, encompasses information collected by sensors like cameras and radar, which provide insights into adjacent vehicles within the same lane and the driver is personality traits. This data acquisition system includes details about driver actions (such as eye, hand, and foot movements), road characteristics (e.g., road curvature and lane width), information about nearby vehicles (like relative distance and speed), and ego vehicle data via the CAN-Bus (covering acceleration, vehicle speed, throttle position, and steering angle).
  For an individual driver, a vehicle equipped with this data acquisition system can be instrumental in the construction of driver behavior models, evaluating driver distraction and inattention, comprehending decision-making processes, and profiling personal attributes;

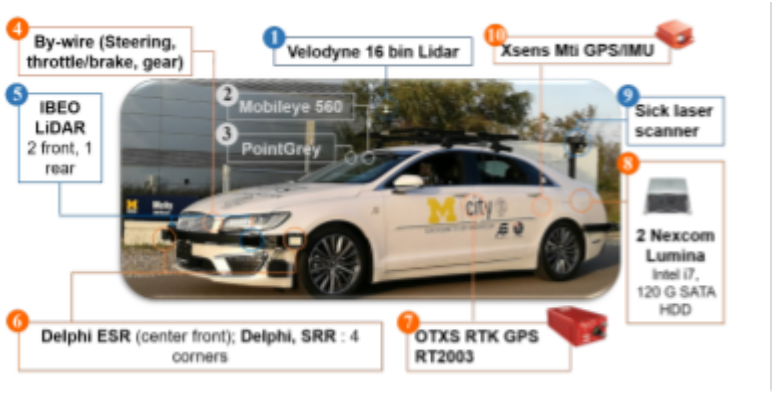


Figure 2.4: Example of in-vehicle data acquisition systems developed by University of Michigan [4].

- **Video/camera data with a fixed field test:** A lower cost alternative but efficient way is to install a video recorder at a fixed position, as the example of Figure 2.5, obtaining video-based data to analyze driver behaviors. This kind of data collection method allows researchers to obtain a huge amount of driving data for many vehicles with less time, more than 6 thousand vehicle trajectories take the researchers only about 45 minutes with this method. A video/camera in a fixed field can collect a great amount of driving data at a lower cost, but the diversity of data limits its application in deeply understanding and modeling driver behavior.

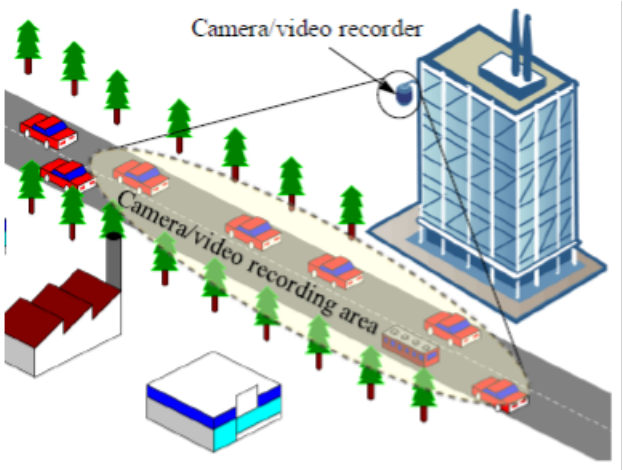


Figure 2.5: Illustration of data acquisition systems for car-following behaviors using a camera/video recorder with a fixed position [4].

The [4] proved to be essential to this present dissertation helping to define a method for analyzing the exact amount of data needed, and also provided important information of the way of data acquisition and processing algorithm. Due to this article, it was possible to estimate the amount of the data needed in terms of time, number of trips and the number of drivers. It also provided the amount of data acquisition sensors needed and the sampling rate of each.

2.5 Driving Style Analyses Using Primitive Driving Patterns with Bayesian Nonparametric Approaches

Driving style, in paper [5], refers to a set of dynamic activities/steps that a driver uses when he's driving, according to his/her personal judgement, experience and skills. Decomposing complex driver behavior into simple, smaller, and primitive patterns can facilitate identification and analysis of driving styles.

Driving pattern definition is related to how driving patterns are characterized, allowing a human or algorithmic observer to identify patterns from measured data. These characteristics tend to group into three categories:

1. **Physical Boundaries:** The definition to a pattern typically refers to a physical change that occurs when driver's operation/decision starts or ends. These natural physical boundaries can be specified by changes of vehicle steering angle, brake/accelerator pedal position or their combination. These characteristics may be specific to a particular operation, turn left or right, or generalize multiple operations, acceleration, and lane change;

2. **Template Boundaries:** Driving pattern can be defined by a user-provided template. Defining a primitive pattern by a template or a set of templates allows maximizing flexibility for users depending on special requirements. However, this approach is usually time-consuming for preparing the templates and can miss special cases;

3. **Derived Metric Boundaries:** A primitive pattern can also be defined by a change in metric, as variance, or derived signals based on supervised and unsupervised approaches. These aforementioned approaches, however, require prior knowledge about the number of patterns or clusters.

In [5], it's proposed a learning-based framework for driving styles analysis based on primitive driving patterns and for that, it's introduced a Bayesian nonparametric approach to directly learn primitive driving patterns from time series driving data without requiring prior knowledge about the number of primitive patterns.

Hidden Markov Model - Based Approaches:

- **Hidden Markov Model (HMM):** The core of HMM consists of two layers: a layer of hidden state and a layer of observation or emission, as represented in Figure 2.6.

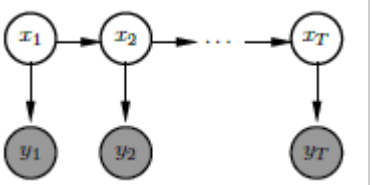


Figure 2.6: A graphical model of Hidden Markov model [5].

The shaded nodes are observations, and the white ones are latent states, in this case, they are primitive driving patterns.

- **Hidden Semi-Markov Model (HSMM):** The HSMM is an extension of HMM, as represented in Figure 2.7, and is traditionally defined by allowing the underlying process to be a semi-Markov chain, each state has a variable duration. In this paper, it's assumed that each hidden state's duration is given over an explicit distribution, called explicit duration HMM. Therefore, the generative process of a standard HMM with a random state duration time, drawn from some state-specific distribution when the state is entered.

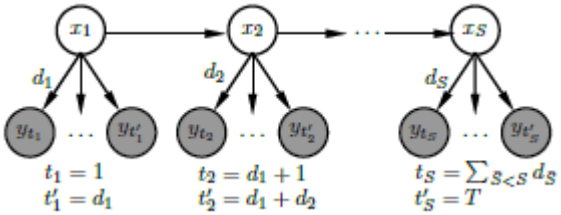


Figure 2.7: A graphical model of Hidden Semi-Markov model [5].

Where d denotes the duration of a state that enters at time t.

- **Hierarchical Dirichlet Process (HDP):** It's assumed that the number of latent dynamic nodes or patterns is unknown and these nodes of HMM and HSMM are subject to a specific distribution defined over a measure space, as represented in Figure 2.8. The Dirichlet process (DP) is a measure on measures and provides a distribution over discrete probability measures with an infinite collection of atoms on a parameter space (Theta) that is endowed with a base measure. The weights (Beta) are sampled by a stick-breaking construction and Tau is the data length.

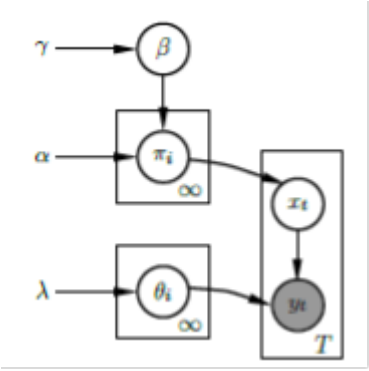


Figure 2.8: Graphical model of the Bayesian nonparametric HDP-HMM [5].

- **Sticky HDP-HMM and HDP-HSMM:** Applying the HDP prior to the HMM and HSMM, we can obtain the HDP-HMM, sticky HDP-HMM, and HDP-HSMM, as represented in Figure 2.9. For the sticky HDP-HMM by adding an extra parameter (Kappa > 0), it biases the process toward self-transition increasing the expected probability of self-transition by an amount proportional to Kappa.

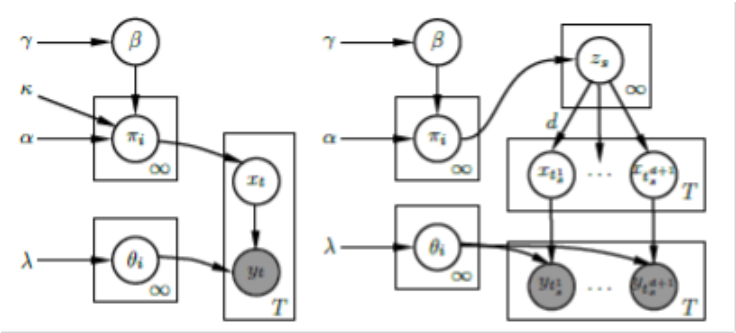


Figure 2.9: Graphical model of the Bayesian nonparametric Sticky HDP-HMM (on the left) and HDP-HSMM (on the right) [5].

Observation (or Emission) Model:

The observation model is determined by the type of function, which can be Gaussian emissions or switch linear dynamic models. One main challenge with non-parametric approaches is that one must derive all the necessary expressions to properly perform interference. It's assumed that observations are drawn from a Gaussian distribution.

Data Extraction and Preprocessing

In order to extract, in this case, car-following events, it was selected the following variables:

- Vehicle acceleration (a), which can directly reflect driver's intents and driving preference and, it's divided into five levels based on driver's vestibular and kinesthetic thresholds: aggressive acceleration, gentle acceleration, no acceleration, gentle deacceleration, and aggressive deacceleration;

- The relative range between subject and lead vehicles (Delta d). It can reflect driver's preference in headway, and it's divided into three levels: long distance, normal distance, and close distance;

- The relative range rate (Delta v). It can capture the dynamical relationship between two vehicles, the velocity of subject vehicle and the velocity of the lead vehicle, and it's divided into five levels: rapidly closing in, closing in, keeping, falling behind, and rapidly falling behind.

When training a model, in order to reduce the scale influence of different variables on the segmentation results, it was additionally normalized each variable of observation vector for each event so that the empirical variance of the set was equal to one.

After studying the HMM-based approaches, they conclude that the HDP-HSMM provides a better semantical way for analyzing driver's car following behaviors.

Normalized Frequency Distribution of Driving Patterns:

Instead of using the statistical metrics, they use the normalized frequency distribution of primitive driving patterns to characterize driving styles, which allows one to intuitively analyze driving habits.

Interdriver Differences in Driving Style:

The normalized distribution of primitive pattern is able to describe and analyze driving stiles for individuals. In order to compare the driving styles of two drivers, it's used the Kullback-Leibler divergence which can provide a flexible way to illustrate the similarity or divergence between drivers from a semantical perspective.

The [5] presents a new framework for driving style analysis using primitive driving patterns with Bayesian non-parametric methods which is very important for this dissertation because before starting to evaluate the state of fatigue of a driver, it's crucial to understand what kind of driver we are evaluating, and that will define the main criteria to characterize the level of drowsiness of that specific driver. This research includes important information about the approaches that can be used and the variables necessary to implement the system, such as the different groups categorization inside each variable.

2.6 Detection and Prediction Of Driver Drowsiness Using Artificial Neural Network Models

Driving a car [13] is a complex, multifaceted and potentially risky activity requiring full mobilization of physiological and cognitive resources to maintain performance over time. A driver's operational state while driving involves a complex set of psychological, physiological, and physical parameters. During driving activities, several factors can be critical: in particular, fatigue and monotony may cause a loss of attention, drowsiness and even sleepiness.

Unfortunately, drowsiness cannot be recorded directly but has to be estimated. The most effective recorder of signals related to drowsiness is the electroencephalogram (EEG) because it's the most direct indicator of central nervous system activity, however, it's quite intrusive and difficult to analyze such kind of information. So, the electrocardiogram (EKG), knowing that heart rate variability is linked to the automatic nervous system, this feature is often used as an indicator of drowsiness. Heart rate often decreases during driving and also when the driver is tired, but

the opposite may also occur, so, taken alone, the indicator cannot be considered as adequate and exclusive indicators of drowsiness or fatigue.

The standard deviation of car position relative to lane midline, and steering wheel movements, are the most common features used to detect drowsiness, but and again, they are not exclusive indicators of drowsiness because they can decrease with other factors such as distraction or with a decline in attention.

In order to solve this kind of problem various measures are often used jointly, such a hybrid approach minimizes the number of false alarms while maintaining a high rate of recognition (essential for good acceptance of the system by the human operator). Nonlinear modeling machine learning, such as Artificial Neural Networks (ANNs), is also often used. With these techniques, the model can extract information from noisy data, and can avoid over-fitting, making it generally more robust.

In [13], information was collected from different sources: physiological, behavioral, and psychological data from the driver, as well as performance information from the vehicle. The goal was to develop and evaluate a model with an ANN, so as to predict when a given impairment state will be reached in addition to detecting this impaired state. First, it was hypothesized that it is possible to predict when the impaired state will arise by using sensorimotor, physiological and performance indicators used to detect drowsiness. Second, it was hypothesized that adding information such as driving time and participant information will improve the accuracy of the model.

Materials and Methods:

- **Participants**

The participants were not allowed to drink alcohol, coffee, or tea, have valid driver's license for at least 6 months, no visual correction needed to drive, not susceptible to simulator sickness, and an Epworth scale (assessing susceptibility to drowsiness) below to 14. A score below 8 on this scale means the person has no sleep debt. A score of from 9 to 14 means the person shows signs of sleepiness, and above 15, the person shows signs of excessive sleepiness. Before the experiment, participants were questioned on their age, their quality of sleep, their caffeine consumption, driving frequency, and number of kilometers per year.

- **Protocol**

The participants drove during between 100 and 110 min in a static driving simulator in an air-conditioned room with temperature control set at 24º Celsius, after lunchtime. While driving, data on driving performance, eyelid and head movements, and physiological data were recorded using SCANER Studio for driving performance at 10 Hz, FaceLAB for sensorimotor signals at 60 Hz, and EKG pulse plethysmography (PPG), and respiration with the Biopac MP150 system and Acknowledge software ate 1000 Hz.

- **Data analyses and modeling**

The level of drowsiness, called ground truth, is evaluated using a method proposed by Wierwille and Ellsworth (1994), which used a scale between 0 and 10. For practical reasons in relation with the ANN, it was stipulated to use a smaller scale (from 0 to 4 with a step of 0.5). Each minute of the recorded video was analyzed and evaluated from 0 (alert) to 4 (extremely drowsy). The modeling process can be divided into two phases. First, one ANN detects the level of drowsiness from a predetermined set of features. Second, if drowsiness is under 1.5, a second ANN predicts (in min) when it will reach 1.5 and gives this time as its output. The neural network toolbox of Matlab R2013a was used to create the ANNs. Two feedforward neural networks were used with 2 hidden layers, and a back propagation training method was applied using the Levenberg-Marquart algorithm. The performance function used for learning was the mean squared error (the average squared error between the network outputs and the target output). To avoid overfitting, the total dataset was distributed in a training sub-dataset (70 percent of the total set, to learn the network's node weights), a validation sub-set (15 percent: to stop learning and avoid overtraining) and a testing sub-set (15 percent: to evaluate the model's ability to work on previously unseen data). Driving performance and driving behavior indicators (car dataset) used in the model were: lateral distance relative to the midline, time-to-line-crossing, steering wheel angle, accelerator pedal angle, shift relative to the lateral line, speed, and number of line crossings. Physiological features used in the model were the heart rate and its variability, and the respiration rate and its variability. Sensorimotor features extracted from FaceLab data were blink duration and its frequency, head movement in translation and rotation, and saccade frequency. In an attempt to rebase individual differences, we subtracted from each signal the mean of the first five minutes of this signal, so that the signal represents variation from an initial state. To optimize learning, each feature was normalized such that minimum and maximum values lie within [-1;1].

Results:

- **Detection**

Driving Time	Participant information	Dataset	Source	MSE	STD	|Error |95%	% Error < 0.5
0	0	Testing	All	0.43	0.04	1.16	0.63
0	0	Testing	Behavioral	0.42	0.02	1.16	0.64
0	0	Testing	Car	0.69	0.04	1.48	0.50
0	0	Testing	Physiological	**0.81***	0.05	1.51	0.43
0	1	Testing	All	0.41	0.04	1.10	0.62
0	1	Testing	Behavioral	0.39	0.04	1.14	0.69
0	1	Testing	Car	0.62	0.03	1.34	0.54
0	1	Testing	Physiological	0.76	0.03	1.52	0.44
1	0	Testing	All	0.27	0.02	0.91	0.80
1	0	Testing	Behavioral	0.23	0.02	0.80	0.83
1	0	Testing	Car	0.40	0.05	1.20	0.66
1	0	Testing	Physiological	0.38	0.05	1.06	0.70
1	1	Testing	All	0.24	0.02	0.84	0.81
1	1	Testing	Behavioral	**0.22#**	0.02	0.87	0.80
1	1	Testing	Car	0.23	0.06	0.73	0.86
1	1	Testing	Physiological	0.29	0.07	0.75	0.82

Table 2.2: Model performance in detecting drowsiness level for the testing dataset: MSE and STD. The worst performance (highest MSE) is highlighted in bold and with a * while the best performance (lowest MSE) is highlighted in bold and with an # [13].

The error is the difference between the real state (as given by the subjective evaluation, the ground truth) and the output, squared and averaged over epochs to provide the mean squared error of the trained model.

As we can see, concerning Table 2.2, the model performs better with all datasets used together or with the behavioral dataset used alone and using driving time and participant information. Performance is slightly worse with the physiological or car datasets used alone.

- **Prediction**

The error, for each epoch, is the difference between the time remaining from the current epoch before the target level is really reached (as per the subjective evaluation) and the time predicted by the trained model (square and averaged over epochs to provide the mean square error). The best performance is achieved with a combination of driving time, participant information and the behavioral dataset. Similar, but not higher, accuracy is achieved with the car and physiological datasets.

Discussion:

The objective of this study was to assess whether the time of occurrence of a given state of drowsiness could be predicted by using ANN models. Overall, using an ANN trained with the same information used to detect drowsiness, it is possible to predict when a driver's impairment will appear to an accuracy of approximately 5 min. In the worst case, for 95 percent of the test dataset, the model can predict when the impairment will appear to within 13.11 min. In the best case, the model can predict the impairment to within 1.97 min. An important point highlighted by these results is how temporal (driving time) and idiosyncratic (participant information) data impact model performance.

- **Dataset comparison: behavioral/physiological/car**

The dataset giving the best performance is the behavioral dataset, followed by the physiological dataset and finally the car dataset, both in detecting the degree of drowsiness and in prediction when a given drowsiness level will occur. A single ANN-based model may not be the best way to take advantage of the dependencies between the different sources of information. An alternative might be to linearly combinate the outputs of three ANNs, each trained with a different dataset: car, physiological and behavioral. The results also show that a model trained with the car dataset alone is less accurate than models trained with other datasets, and this may be due to the fact that driving activity and performance are non-linearly corelated with degree of drowsiness.

Generalization and inter-individual variability:

It is a major challenge to find a general model which can be trained with a limited number of drivers and then applied to other drivers, due to the inter-individual variability. It is now recognized that neurobehavioral and cognitive performances vary considerably from one individual to

another. Situational and personality factors, sleeping habits and driving history can contribute to the understanding of why some people fall asleep at the wheel while others do not.

The [13] is the most directly related with the current dissertation that I proposed, so it's obvious that it was given more attention and detail to this research. The software used to collect data, the software used to create the ANNs and the methods used, and the datasets that are better to use to evaluate the state of drowsiness, are valuable information given here and crucial to the progress of this these.

2.7 Towards a Continuous Biometric System Based on ECG Signals Acquired on the Steering Wheel

The article [6] proposes a biometric recognition system using electrocardiogram (ECG) signals acquired through a steering wheel in driving environments. The objective is to improve signal quality by combining Savitzky-Golay and moving average filters, as well as detecting and removing outliers using normalized cross-correlation and clustering. Features are extracted using Discrete Cosine Transform (DCT) and Haar Transform and are fed into various decision methods like Support Vector Machines (SVM), k-Nearest Neighbors (kNN), Multilayer Perceptrons (MLP), and Gaussian Mixture Models - Universal Background Models (GMM-UBM) for identification and authentication tasks. Personalized authentication and historical score weighting techniques are also studied. The proposed method achieves an identification rate (IDR) of 94.9% and an equal error rate (EER) of 2.66%, comparable to state-of-the-art methods. However, with limited training data, the IDR decreases to 70.9%, and the EER increases to 11.8%. The method shows promise for biometric recognition using ECG signals in driving environments and has the potential for continuous systems in noisy conditions. The [6] discusses the increasing use of biometric recognition systems in various applications, with ECG being explored as a potential biometric trait due to its universality, uniqueness, permanence, and liveliness. Previous research has investigated ECG-based biometric recognition in various configurations, including steering wheel-based acquisition in vehicles. The article aims to overcome challenges related to noise and signal loss in driving environments and explore the feasibility of continuous ECG-based biometric recognition.

Proposed Methodology:

The proposed method involves acquiring ECG signals on the steering wheel's surface during driving, as exemplified by Figure 2.10. Signal denoising is performed using a combination of Savitzky-Golay and moving average filters to address the dominant and unpredictable noise present in driving-acquired ECG signals. Signal preparation involves R-peak detection, heartbeat segmentation, amplitude normalization, and outlier detection using a method called Normalized Cross-Correlation Clustering (NCCC), as represented in Figure 2.11.

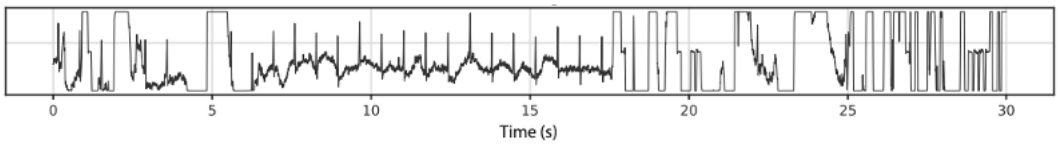


Figure 2.10: Example excerpt of the signal used, acquired on the steering wheel whilst driving [6].

It is relevant to remark the evident and unprecedented predominance of noise over the signal, especially the effect of varying impedance denoted by the frequent saturation periods, which pose significant threats to the reliability of the recognition process.

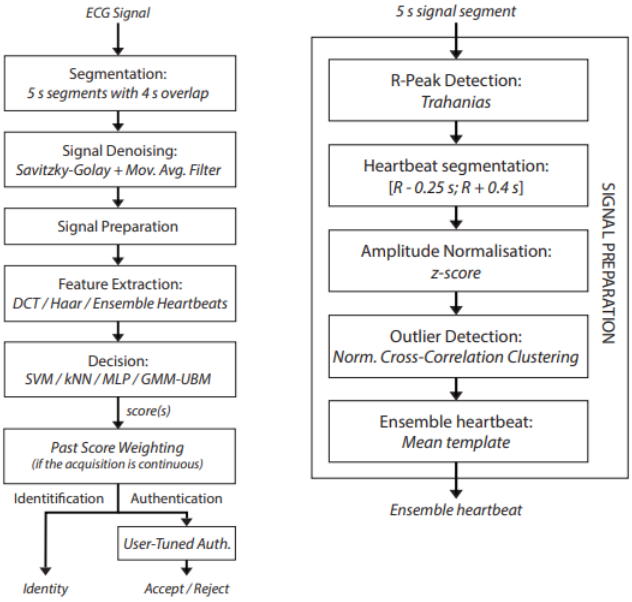


Figure 2.11: Overview of the proposed method, from acquisition to recognition (on the left), and the detailed process of the signal preparation block (on the right) [6].

Feature Extraction:

Features are extracted from the preprocessed ECG signals using Discrete Cosine Transform (DCT) and Haar Wavelet Transform. DCT coefficients are selected for the frequency range of [0, 40] Hz, while the second-level detail coefficients of Haar Wavelet Transform are chosen, resulting in a total of 52 and 163 features, respectively.

Recognition:

The recognition phase involves applying different decision methods such as SVM, kNN, MLP, and GMM-UBM to perform identification and authentication tasks. Additionally, personalized authentication and historical score weighting techniques are studied to improve the system's performance.

Results and Discussion:

The proposed method is evaluated using continuous ECG recordings acquired during driving. Signal denoising using the SG + MAF method outperforms other noise reduction methods. The NCCC outlier detection method is effective in rejecting noisy and false heartbeats. SVM with DCT features achieves the best overall results in identification and authentication tasks. The personalized authentication and historical score weighting techniques consistently improve the performance.

Conclusions:

Despite challenges posed by noise and signal loss in driving environments, the proposed method demonstrates the feasibility of ECG-based biometric recognition. Results indicate potential for continuous biometric systems in noisy driving environments, with future developments expected to enhance the method's performance.

2.8 Importance of subject-dependent classification and imbalanced distributions in driver sleepiness detection in realistic conditions

The [7] presents a comprehensive study on using electrocardiogram (ECG) and electrooculogram (EOG) signals for driver drowsiness detection in real driving conditions. Unlike previous studies in simulated environments, the authors argue that on-road studies are essential to avoid misleading results. The study employs supervised machine learning methods to analyze the influence of subject dependence on drowsiness classification. The results indicate that subject-dependent classification outperforms subject-independent classification, especially for detecting drowsy states. Additionally, the article highlights the challenges posed by imbalanced class distributions and suggests further investigation into transfer learning techniques and methods for addressing the imbalance.

The [7] addresses the crucial issue of detecting driver drowsiness using physiological signals like ECG and EOG to enhance road safety. While simulated environments have been used in previous studies, the authors emphasize the need for on-road data due to potential differences in driver behavior between simulated and real-world conditions. A field driving study was conducted as part of the SleepEye project to gather a database for analysis. The use of physiological signals offers promising insights into building more effective drowsiness detection systems than existing external-factor-based approaches.

Related Work:

The related work [7] section highlights various studies on driver drowsiness detection, considering different sources of information and experimental environments.

Physiological signals such as HRV extracted from ECG and blink events detected from EOG are essential features in these studies. However, the limitations of small participant sizes are noted.

Material and Methods:

The [7] describes the study's methodology, which involved recruiting 20 participants from the Swedish National Register of Vehicle Owners. Participants underwent sleep and health assessments, followed by real-world driving sessions. Physiological data from ECG and EOG were recorded using a Vitaport 3 device while participants were asked to evaluate their state using KSS scale, results are represented in Figure 2.12. The study employed a subject-dependent classification approach and aimed to explore subject-independent classification through a user-independent system that adapts to individual driver behavior.

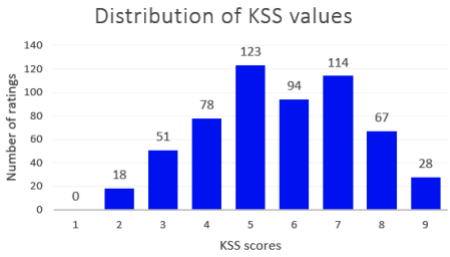


Figure 2.12: Distribution of KSS scores in the whole dataset [7].

No	Feature	Description
1	HR	heart rate
2	SDNN	standard deviation of NN intervals (HRV_i)
3	SDSD	standard deviation of differences between adjacent NN intervals ($HRV_i - HRV_{i-1}$)
4	RMSSD	square root of the mean of the sum squares of differences between adjacent NN intervals
5	NN50	number of pairs of successive NNs that differ by >50 ms
6	pNN50	NN50 divided by total number of NNs
7	NN20	number of pairs of successive NNs that differ by >20 ms
8	pNN20	NN20 divided by total number of NNs
9	HF	total energy in the high frequency band (0.15–0.4 Hz)
10	LF	total energy in the low frequency band (0.04–0.15 Hz)
11	VLF	total energy in the very low frequency band (0.003–0.04 Hz)
12	TP	total power in all bands (0.003–0.4 Hz)
13	HFnu	HF normalised (HF divided by the difference between TP and VLF)
14	LFnu	LF normalised (LF divided by the difference between TP and VLF)
15	VLFnu	VLF normalised (VLF divided by TP)
16	LF/HF	ratio between LF and HF

Table 2.3: Extracted ECG features. NN refers to the time between two normal R-peaks [7].

ECG and EOG-feature-based methodology for monitoring driver´s state:

The [7] outlines the preprocessing and feature extraction methods for ECG and EOG signals. For ECG, 16 features related to heart rate variability (HRV) were extracted, including time-domain and frequency-domain features, as represented in Table 2.3. EOG data allowed detection of blinks and saccades, and 17 blink-related features were computed. The methodology's accuracy in detecting R-peaks, blinks, and saccades was evaluated.

Classification:

The study [7] performed classifications using three datasets: ECG features, EOG features, and combined ECG + EOG features. Subject-dependent and subject-independent classifications were explored, with the latter involving data from n-1 individuals in the training set and data from the nth individual in the testing set. The imbalanced class distribution was addressed using cost-sensitive training and balancing techniques.

Results e Discussion:

The [7] evaluated the accuracy of drowsiness classifications using SVM, ANN, RF, and GBT classifiers. The combination of ECG and EOG features yielded improved results compared to using individual features. Subject-dependent classification outperformed subject-independent classification, indicating the significance of individual physiological differences in drowsiness detection. Balancing the data improved the classification of the "drowsy" class but introduced classification errors for the "awake" class, suggesting the need for a careful balance in alert systems.

Conclusions:

The [7] demonstrates the effectiveness of using ECG and EOG signals for driver drowsiness detection under realistic conditions. Subject-dependent classification proved superior, emphasizing the importance of considering individual physiological differences. Addressing imbalanced class distributions remains a challenge, and further research into advanced machine learning methods and combining objective and subjective measures is recommended to enhance drowsiness alert systems tailored to individual driver characteristics.

2.9 A literature review and research agenda on motivational technologies in transportation safety

The article [8] adelves into the underexplored realm of motivational technologies in the context of transportation safety, stressing the urgency for increased research and practical implementation in this field. Through a systematic review of 62 studies, the article examines the potential of motivational technologies to mitigate accidents and their consequences. While these technologies aim to induce behavioral changes, evidence for their long-term benefits and possible adverse effects remains limited. The study emphasizes the necessity of aligning motivational design with the cognitive demands of transportation tasks and explores the applicability of motivational technologies across various transportation modes. Additionally, the social aspects of these interventions in design and evaluation are highlighted. The article provides a comprehensive overview of motivational technologies' current state, furnishes design recommendations, and identifies crucial future research directions.

The [8] initiates by introducing the critical issue of transportation safety and the alarming number of fatalities caused by accidents. Acknowledging the role of human factors in transportation systems, the article underscores the lack of motivation for safety as a key hindrance, leading to unsafe behaviors and decisions. As traditional strategies like training and awareness fall short in inducing behavioral changes, the concept of motivational technologies emerges as a promising approach to enhance transportation safety. The article aims to provide a holistic synthesis and overview of motivational technologies' applicability in the context of transportation safety, encompassing different transportation domains, motivational design strategies, safety approaches, and resulting outcomes. The conclusion of the study and its limitations are also discussed.

Background:

This section lays the groundwork for the study, offering a comprehensive definition of safety in the transportation context and highlighting the inherent challenges. Motivation is introduced as a driving force behind human actions, classified into intrinsic and extrinsic forms. The article draws attention to the utilization of motivational technologies, such as gamification, persuasive technology, and serious games, which have been successfully employed to promote intrinsic motivation in mundane activities across various domains. Notably, the lack of an all-encompassing review on the application of motivational technologies in transportation safety is emphasized, prompting the need for this study.

Methods:

The paper [8] describes the systematic literature review conducted to explore motivational technologies in transportation safety. The methodology involved meticulously searching and analyzing data from pertinent studies, using three main categories of keywords: motivational technologies, transportation, and safety. From a pool of 873 studies, a total of 38 were deemed relevant and included in the analysis. The studies were extensively examined regarding transportation facets, safety improvement measures, motivational interventions, methodologies employed, and reported results. Various data collection methods and impact evaluation techniques were applied to assess the effectiveness of motivational technologies, as represented by Table 2.4.

Category	Attribute	Possible values
Publication details	Publication year	1989–2021
	Publication type	Conference publication, Journal article
	Number of citations	0–157
Transportation facets	Transportation modes	Airway; Pipeline; Railway; Roadway; Waterway
	Target audiences	Crew members; Cyclists; Drivers; Passengers; Pedestrians; Pilots
Safety improvement measures	Injury prevention phases	Pre-event; During-event; Post-event
	Human error management approaches	Person; System
Motivational interventions	Technology types	Gamification, Persuasive technologies, Serious games
	Affordances	Avatars or Characters; Badges or Rewards; Challenges or Goals; Cooperation or Teams; Feedback; Hints or Onboarding; Leaderboards or Rankings; Levels or Progression; Narratives or Storytelling; Points or Scores; Simulation or Virtual worlds; Time pressure
Methodology and outcomes	Methodological designs	Framework or guidelines; Design proposals or Pilot studies; Empirical evaluations; Qualitative, Quantitative, Mixed
	Data collection method	Diary studies; Eye movement tracking; Interaction logs; Interviews and spoken tests; Physiological measurements; Questionnaires or surveys; User observations; Vehicle data, Written and drawing tests
	Psychological outcomes	Engagement, flow or motivation; Enjoyment; Fatigue, monotony or strain; Locus of control or self-efficacy; Perceived alertness, distraction, arousal or boredom; Perceived learning, persuasiveness or efficacy; Perceived presence or reality; Perceived usability or usefulness; Perceptions of individual affordances; Risk awareness and perception or fear; Satisfaction or attitude; Willingness to use again; Negative; Neutral; Positive
	Behavioral outcomes	In-game performance; Knowledge acquisition or learning; Knowledge retention; Playing or usage time; Safer driving behavior

Table 2.4: Classification Scheme [8].

Results:

The results of the systematic review are presented, based on the analysis of the 62 selected studies. Primarily, a significant portion of the research focused on road safety, with a particular emphasis on the behavior of drivers, pedestrians, and cyclists. Among the motivational interventions used, serious games and gamification emerged as the predominant strategies, leveraging resources such as feedback, challenges, simulations, and virtual worlds, as represented by Figure 2.13. Psychological outcomes, such as enjoyment, usability, and engagement, were frequently evaluated, alongside behavioral outcomes concerning knowledge acquisition and improvements in driving behavior. It is noted that some results demonstrated positive impacts, while others remained neutral or inconclusive.

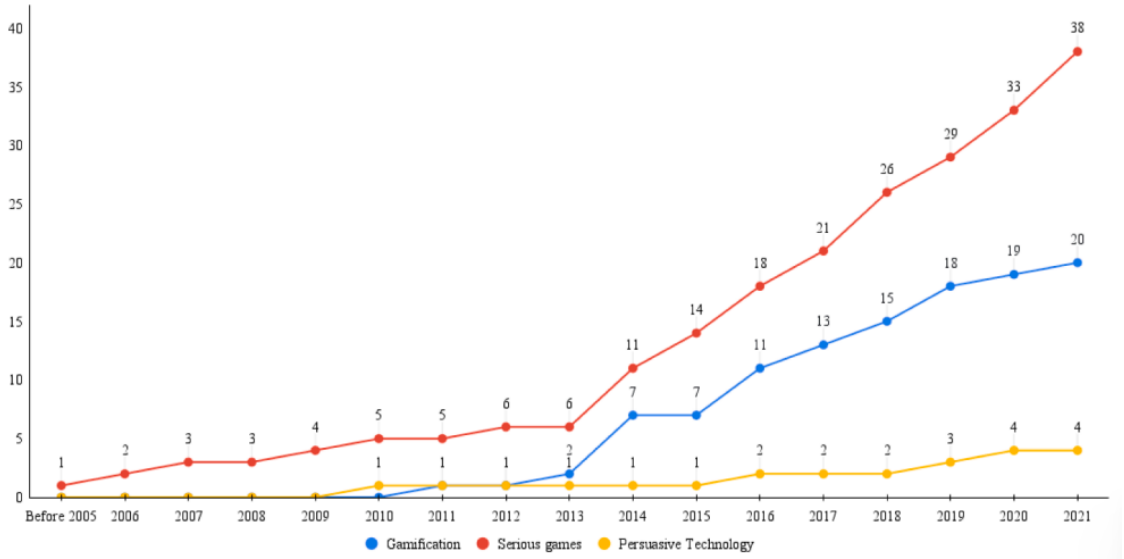


Figure 2.13: The development of different motivational designs [8].

Discussion:

The discussion section sheds light on the existing research gaps, especially the dearth of studies concerning motivational technologies in railway and pipeline safety, attributed to the prevalence of automation and controlled operations in these domains. The importance of considering the injury prevention phase and carefully selecting appropriate motivational interventions based on the specific safety improvement measures are underscored. The article offers three essential design recommendations to guide the successful implementation of motivational technologies in transportation safety. Additionally, the study outlines critical avenues for future research, advocating for the expansion of the research scope to encompass a broader range of transportation modes. It further suggests exploring social influences, such as citizen and worker engagement, shared goals, and cooperation in improving transportation safety. Furthermore, the article calls for in-depth research comparing motivational technologies with traditional safety training approaches

and delving into the transfer of knowledge from serious games to real-life safety behaviors. Long-term effects of motivational interventions and optimal methods of presenting feedback in diverse contexts are also identified as key areas requiring further investigation. To advance the field and attain concrete evidence of their real-world impact, the study emphasizes the significance of assessing motivational technologies' effects on safety indicators like accident rates.

Conclusions:

In conclusion, the article [8] offers a comprehensive synthesis of the state-of-the-art of motivational technologies in transportation safety through a meticulous systematic review. It elucidates those motivational technologies hold promise in enhancing individual safety behaviors and preventing accidents across various transportation modes, including roadways, airways, and waterways. The available evidence demonstrates their potential to induce psychological changes and facilitate the acquisition of safety-related knowledge. However, the limited evidence for behavioral changes beyond simulated settings necessitates considering real-world contexts and the transfer of acquired skills to authentic transportation environments. Consequently, the article imparts valuable design recommendations for motivational technology implementation, focusing on injury prevention phases, safety improvement approaches, and the cognitive demands of transportation tasks. Nevertheless, the study acknowledges its limitations, particularly in terms of keyword-based inclusion criteria and the generalization of individual study contents. To further advance the field, the study proposes several avenues for future research, encompassing a broader spectrum of transportation modes, exploration of system design beyond individual error management, and the inclusion of social aspects in the design and evaluation of motivational technologies. Methodologically, the study highlights the need for comparative investigations and qualitative research to assess the long-term impacts of motivational interventions on safety indicators, ultimately striving to unlock the full potential of motivational technologies in improving transportation safety.

2.10 Conclusion

All the articles and reviews presented on this chapter had an important rule to achieve the final system and functional architecture of the work. They served to make us know what already exists and what can be developed and optimized. Information about the amount of data, the kind of data and the best methods to achieve the goal of this dissertation also were very important for the work.

In the realm of scientific research, the community employs advanced methods and methodologies for the prediction and detection of driver fatigue, which is crucial for enhancing road safety. These approaches encompass the utilization of cutting-edge technologies, such as physiological monitoring sensors, analysis of steering patterns, and the identification of drowsiness signs, which include ECG and EEG data. Furthermore, interdisciplinary investigations and behavioral studies play a pivotal role in gaining a comprehensive understanding of the risk factors associated with fatigue-related incidents. This comprehensive and multifaceted approach aims to contribute to the development of safer solutions for preventing accidents linked to driver fatigue.

The systematic research carried out made it possible to gather a body of knowledge and methods that were crucial to the development of the dissertation. In addition, the analysis of the state of the art made it possible to sketch out an initial architecture for the system. Thus, the initial hypothesis was to investigate the possibility of developing an estimator without using the driver's image and adding additional sensors to the car.

Chapter 3

Technical Solution Design

3.1 Introduction

This chapter aims to explore and evaluate different solutions for detecting and preventing drowsiness and fatigue while driving to ensure road safety. The initial approaches considered were the use of a video camera, the measurement of heart rate and ECG data and the utilization of OBD data from the vehicle.

Two hypotheses were set: one involving a Feedforward Supervised Artificial Neural Network with backpropagation and other using K-means clustering for exploratory data analyses.

Finally, it is presented the flowchart representative of the systematic process of the proposed system.

3.2 Initial Approaches

3.2.1 Video Camera

The first solution that comes to mind as being the most effective and easy solution, when we talk about sleepiness/fatigue symptoms while driving, is the video camera because we can check the eye closure and the behavior of the driver, as represented by Figure 3.1. If the driver repeatedly closes their eyes and exhibits signs of discomfort, it is indicative of a momentary attention deficit. This unequivocally underscores the driver's compromised ability to continue operating the vehicle safely.

Although it initially appeared to be a highly promising solution to our issue, it was soon realized that potential challenges regarding privacy policies could emerge, because nowadays, cars are used as places to have an important, personal and private conversation, or can be seen as a more intimate place, and knowing that something or even someone can have access to a camera that can record the moment could be a big problem in the future, so, the use of a video camera is not the optimal path for this work.

Figure 3.1: Internal video camera (on the left) and monitored signals examples captured with an internal video camera (on the right) [9].

3.2.2 Heart Ratio and Electrocardiogram

It is known that in normal routine sleeping at night, the heart ratio, HR, decreases at the initial stage of sleeping. However, when driving, the HR can increase, decrease or stay neutral because of the parasympathetic and sympathetic activation due to both sleepiness and driving itself, and, that 2 states can be mixed (actually, it's not possible to distinguish them). So, [14] concluded that the HR decreases significantly under drowsy conditions while driving, compared to the overall driving HR by 9.3 percent and increases in driving by 7 percent as compared to normal routine HR.

So, the heart ratio itself do not allow to solve the problem because the HR values need to be hardly studied in order to understand the normal driving HR values that are not the same as a normal routine and be able of classify possible drowsiness events while driving, so, with the time limitation of this work and the aim of a low budget, this solution is not effective and can result in a less precise result and induce in wrong conclusions.

However, including the heart ratio in the algorithm, as represented by Figure 3.2 (assuming the use of ECG, because other heart ratio acquisition methods as smartwatches do not have the necessary accuracy for this kind of metrics) can be very useful to make some decisions in difficult situations of classification.

This dissertation does not include this type of acquisition once the necessary sensors are expensive and go off the low budget target.

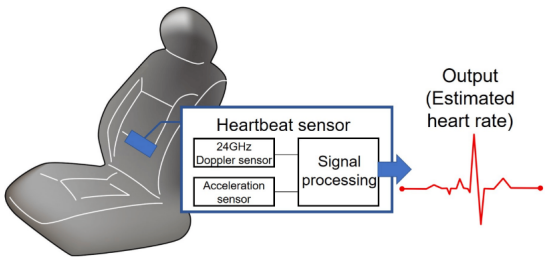


Figure 3.2: Outline example of an estimator heart rate system [10].

3.2.3 On-Board Diagnostics

On-board diagnostics (OBD) [15] is a term referring to a vehicle's self-diagnostic and reporting capability. The amount of diagnostic information available via OBD has varied widely since its introduction in the 1980s versions. Since 1996, all the cars were built with the OBDII interfaces capable of reading and writing on the unit control. With the OBD system we achieve the goal of having information about the vehicle velocity, acceleration, throttle position, fuel consumption, and a lot of other data.

With the aim of a "low budget" resource, the diagnostic interface (like ELM 327, represented in Figure 3.3) is a powerful component capable of reading the necessary data in real-time for a very low price putting this alternative as the more appropriate for this aim.

Using the OBD's software, it's possible to save the data from the trip and use it in real-time, turning possible a good approximation about the driver's behaviour and associate it with the fatigue.

Figure 3.3: ELM327 Mini [11].

3.2.4 Quiz

In order to have some indicators of the perception of the drivers themselves, a questionnaire was drawn up for professional drivers (see Appendix A), which, although the take-up was not significant, was an indicator of the symptoms to be considered when choosing the variables of the data set and developing the algorithm.

3.3 Data Set

3.3.1 Available Variables

At the beginning, we had the intention of doing real life scenarios with real values and use them in the algorithm responsible for classifying the data into the 3 categories defined.

However, after some trips using the ELM327 Mini, difficulties arose in accessing the trip history data and also the inability to export them as a a .csv or .txt file for use as a read file for the created code made as think about finding another kind of solution, and, more data means more efficiency and reliability, and to have the necessary amount of data, it would be necessary a lot of travels and time, so, me and my supervisor decided that we should work with tests that already happened and were published online.

Lucky or not, the OBDII website makes available some travels made with the ELM 147 and the values obtained during the trips.

Some of the variables that the ELM 147 is able to read are:

- Time;
- Average fuel consumption (MPG);
- Average speed (mph);
- Average speed (mph);
- Calculated instant fuel consumption (MPG);
- Distance travelled (miles);
- Engine RPM (rpm);
- Fuel used (gallon);
- Instant engine power (hp);
- MAF (g/sec);
- Throttle position (percent);
- Vehicle acceleration (g);
- Vehicle speed (mph);
- etc...

3.3.2 Used Variables

After the extensive theoretical study, from the extensive list of variables that could be used, we decided that only 3 were extremely necessary to the work, being them:

- The vehicle speed, the vehicle acceleration and the throttle position.

After an extensive analysis of the source file it was noticed that the throttle position values are almost the time null values. For that reason it was necessary to exclude that variable from the variable's list resulting only 2 variables:

- The vehicle speed and the vehicle acceleration.

Data as the steering wheel, the average speed, the distance travelled and the fuel consumed are also very important, but, if the system needs the data from these parameters, that just means that the program has already failed and the driver is in a very dangerous situation.

3.4 Hypotheses Set

3.4.1 Supervised Artificial Neural Network

The first attempt was the creation of a Feedforward Supervised Artificial Neural Network with backpropagation [16] because of its capability to learn non-linear models and to learn models in real-time. The ANN (see Appendix B) was created with an input layer, a hidden layer with 5 neurons and an output layer with 1 neuron, as represented in Figure 3.4. The network was trained to perform a regression task.

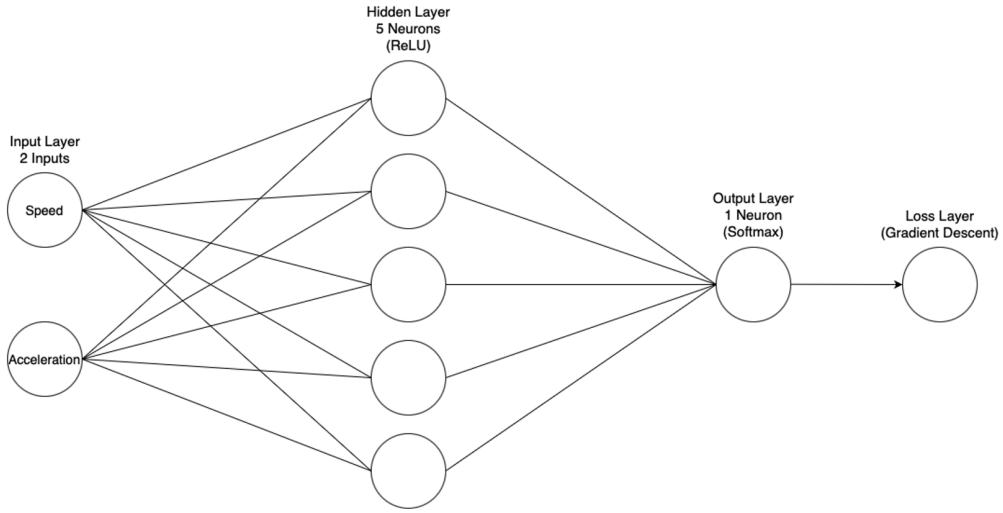


Figure 3.4: Neural Network Representation.

For the activation functions, for the feedforward, it was decided to use the Rectified Linear Unit activation function for the hidden layer due to its non-linearity, simplicity, computer efficiency and ability to address the vanishing gradient problem, which returns the input value if it is

greater than zero or zero otherwise and the Softmax activation function [17] for the output layer, which normalizes a vector of values to represent a probability distribution making possible the interpretation of that probabilities for multiclass classification tasks. For the Backpropagation [18], it was used the derivative of the ReLU function in the hidden layer that returns 1 if the input value is greater than zero or zero otherwise and the derivative of the Softmax function to update the weights of the output layer enabling it to learn a better representation of the data and improve its performance on the learning task. It was also considered the Gradient Descent algorithm [19] for the cost function. The cost function represents the discrepancy between the predicted output of the model and the actual output. The goal of gradient descent is to find the set of parameters that minimizes this discrepancy and improves the model's performance.The algorithm operates by calculating the gradient of the cost function which indicates the direction and magnitude of steepest ascent. However, since the objective is to minimize the cost function, gradient descent moves in the opposite direction of the gradient, known as the negative gradient direction. By iteratively updating the model's parameters in the negative gradient direction, gradient descent gradually converges towards the optimal set of parameters that yields the lowest cost.

This specific ANN takes in account some disadvantages as:

- The hidden layers have a non-convex loss function where there exists more than one minimum. Therefore different random weight initializations can lead to different validation accuracy;

- The necessity of tuning a number of hyperparameters such as the number of hidden layers, layers and iterations;

- The sensitivity to feature scaling.

The train method is used to train the network. It takes the training data, the corresponding true values and optional parameters such as the learning rate and the number of epochs for training. For each epoch, the method iterates over the training data (samples) and updates the network's weights using the backpropagation algorithm. Every 5 epochs, the method evaluates the performance of the network by calculating the Mean Squared Error (MSE) and prints it to the screen. If the current loss is greater than the loss from the previous iteration, the learning rate is reduced by 10% to avoid oscillations and converge to a local minimum.

3.4.2 K-means Clustering

3.4.2.1 Clustering

Clustering is one of the most common exploratory data analysis technique to get an intuition about the structure of the data. It can be defined as the task of identifying subgroups in the data such data points in the same group that are very similar while data points in different clusters that are very different. That similarity between the data points from the same cluster can be defined using the euclidean-based distance or correlation-based distance. Clustering is considered an unsupervised learning method since does not have the ground truth to compare the output of the clustering algorithm to the true labels to evaluate its performance.

3.4.2.2 K-means Algorithm

K-means [20] is one of the most used clustering algorithms. It is an iterative algorithm that tries to partition the dataset into pre-defined distinct non-overlapping clusters where each data points belongs to only one group. It tries to make the intra-cluster data points as similar as possible while also keeping the clusters as far as possible. It assigns data points to a cluster such that the sum of the squared distance between the data points and the cluster's centroid (arithmetic mean) is at the minimum. With less variation within clusters, the more homogeneous the data points are within the same cluster.

The K-means was created following the work structure of this method:

1. Specify the number of clusters;

2. Compute the sum of the squared distance between data points and centroids;

3. Assign each data point to the closest cluster (centroid);

4. Compute the centroids for the clusters by taking the average of all data points that belong to each cluster.

3.5 Flowchart

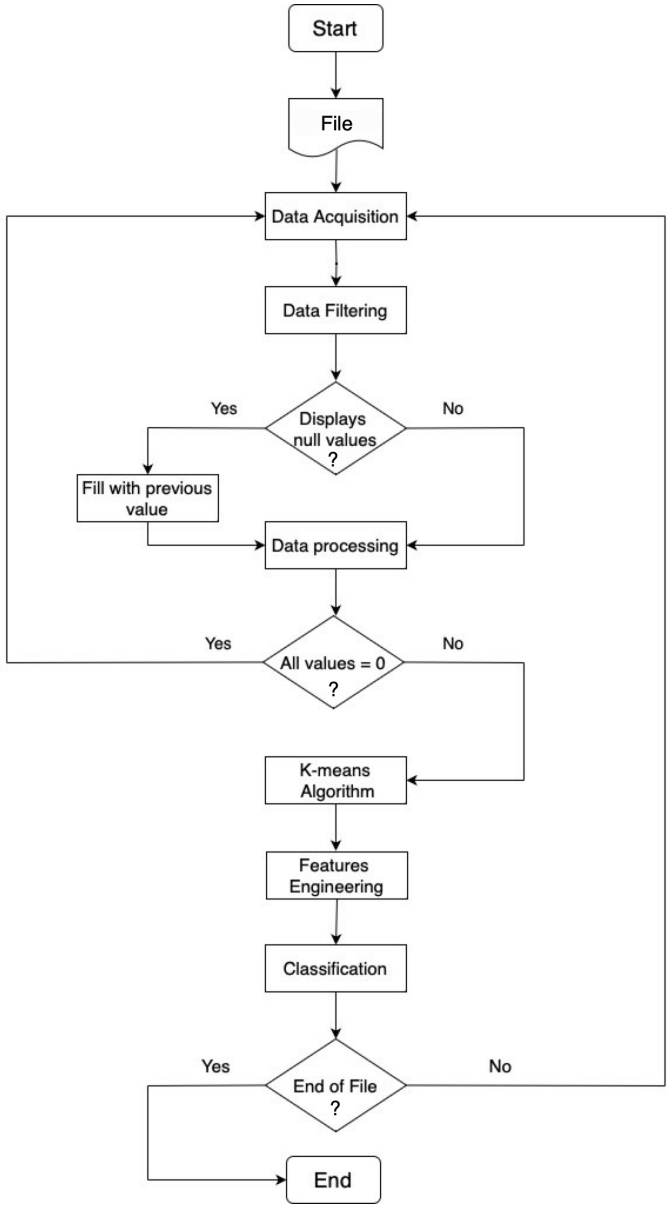


Figure 3.5: Flowchart.

The Figure 3.5 presents the representative flowchart of the system whose main steps are:

1. **Data Acquisition** -> It is read the desired file and stored all the important variables and their values;

2. **Data Filtering** -> After the Data Acquisition, the data needs to be filtered. Running a loop, if there are null values in the variables stored, they are filled with the previous value, not null, stored;

3. **Data Processing** -> The data values will be processed to be used in the classification method. If all the values on the variables are null the system restart Data Acquisition of the next minute of values. If not, the system proceeds;

4. **K-means Algorithm** -> The K-means algorithm will cluster the data points and calculate the cluster's centroids to be used on the classification method;

5. **Features Engineering** -> The code will discover the maximum value between the data points and theirs cluster's centroids to be used on the classification method;

6. **Classification** -> This method has the responsibility to forecast the state of fatigue of the driver every minute returning it in real-time. After that minute the code tests if the file ended. If not, proceeds to the next minute of Data Acquisition, otherwise, the program ends.

3.6 Conclusion

This chapter served as an initial exploration of potential solutions. The K-means algorithm was the chosen solution since it presented considerable better results than the Supervised Artificial Neural Network.

The subsequent chapter will focus on the implementation, evaluation, ad analyses of the chosen methods to create an effective system for preventing accidents caused by drowsy drivers and as result increase road safety.

Chapter 4

Framework, Source Code and Results

4.1 Introduction

This chapter presents the framework and implementation of the fatigue detection software (see Appendix C). The development of this software was motivated by the study and exploration of machine learning concepts, specially ANN and clustering algorithms.

Data Acquisition, Data Filtering and Data Processing are the 3 main stages presented. Within the Data Processing, it was employed the K-means algorithm for clustering and Features Engineering for data manipulation. The classification method, an essential part of the software, utilizes speed and acceleration to identify potential signs of driver's state of fatigue.

An example output shows how the software correctly identify the start and end of trip and detect possible signs of fatigue during the journey, issuing appropriate warning messages to the driver. Although tested on a database rather than real-time scenarios, the software's functionality was validated.

4.2 Framework

In one of the latest course units of the Master in Electrical and Computer Engineering, MC (Machine Learning), Artificial Neural Networks, clustering algorithms, including K-means algorithm, were approached, studied, and programmed. So, it was decided the same platform used on the class, Google Colab [21], and the programming language associated to it, Python, to create the program, as represented in Figure 4.1.

The Google Colaboratory [22], or "Colab" for short, is a product from Google Research and allows anybody to write and execute arbitrary python code through the browser and is especially well suited to machine learning, data analyses and education. It's "ace card" is the availability of the free graphics processing units (GPU) to allow fast training.

One of the excellent features of Google Colab is the use of text and code compartmentalisation. Basically, It is possible to split the code up into chucks of code and text in code cells and text cells

respectively. This will facilitate eligibility and give well defined checkpoints allowing the user the possibility of visualise the data's structure while coding .

Figure 4.1: Framework [12].

4.3 Source Code - "How It Works?"

4.3.1 Data Acquisition

```
def save_N_minute_values(url, filename, acquisition_time, position):
  end_of_file = 0

  # File Download
  urllib.request.urlretrieve(url, filename)

  # List to store the wanted values
  values_to_save = []

  # Opening the file to read
  with open(filename,'r') as file:
    lines = file.readlines()

  # Analysing the lines from the file
  for line in lines[position:]:

    # Getting the time values as a number
    time_string = line.split(',')[0]

    # Converting a time string to an object of datetime
    time_value = datetime.strptime(time_string, '%H:%M:%S.%f')
```

Figure 4.2: Data Acquisition Code: Part 1.

As can be seen in the Figure 4.2, for this part of the developed program it was build one function called *save_N_minute_values* whose inputs are:

- **The URL** -> With the file [23] composed by all the input values to be downloaded;
- **The filename assigned** -> The name of the file to save the downloaded data, in this case *automotive.txt* defined on the main script;
- **The *acquisition_time* defined** -> The desired time interval, in this case it was established 1 minute for the acquisition time in order to have short periods of data values and increase

the accuracy of the clustering model. This value is dynamic, that means that it is possible to change this value anytime in the main script;

- **The *position* variable** -> This variable defines the starting position in the file in order to allow the function start after the last value obtained in the last minute.

In this function it was used the *urllib.request* library to download the file from the specified URL and save it in the *filename*. The file is open in the read mode and its lines are stored in the variable *lines*.

The code iterates through the lines starting in the position indicated and ignores the first line, the headline. To that the variable position is initiated with 1 instead of 0.

The code extracts the time information from the current liner and convert it into a *datetime* object using a specified format '%H:%M:%S.%f'

```
  # Calculating time in seconds since the first line of that stretch
  elapsed_time = (time_value - datetime.strptime(lines[position].split(',')[0], '%H:%M:%S.%f')).total_seconds()

  # Getting N minutes of values (in seconds)
  if elapsed_time <= acquisition_time * 60:
    values_to_save.append(line)

# Checking if the file ended
if position == len(lines):
  end_of_file = 1

return values_to_save, end_of_file
```

Figure 4.3: Data Acquisition Code: Part 2.

The Figure 4.3 demonstrates the elapsed time, in seconds, since the first line of the stretch and is calculated by subtracting the current value *time_value* from the current time from the time of the first line of the stretch.

Values whose elapsed time is less than or equal to *acquisition_time * 60* seconds are added to the *values_to_save* list.

The code checks if the *position* is equal to the total length of the lines in the file. If so, it was reached the end of the file and the variable *end_of_file* is set to 1 (True).

The function returns the *values_to_save* list which contains the values from the file within the specified time interval, and the *end_of_file* variable, that indicates the end of the file.

4.3.2 Data Filtering

```
for line in values:

  line_values = line.strip().split(',')

  timer_time.append(line_values[0])

  if line_values[-3] == '' and line_values[-2] == '':
    speed.append(last_speed_value)
    acceleration.append(last_acceleration_value)

  if line_values[-3] != '' and line_values[-2] != '':
    speed.append(float(line_values[-2]))
    last_speed_value = float(line_values[-2])
    acceleration.append(float(line_values[-3]))
    last_acceleration_value = float(line_values[-3])

  if line_values[-3] != '' and line_values[-2] == '':
    speed.append(last_speed_value)
    acceleration.append(float(line_values[-3]))
    last_acceleration_value = float(line_values[-3])

  if line_values[-3] == '' and line_values[-2] != '':
    speed.append(float(line_values[-2]))
    last_speed_value = float(line_values[-2])
    acceleration.append(last_acceleration_value)
```

Figure 4.4: Data Filtering Code.

In this part of the code, possible to see in Figure 4.4, it is filtered the acquired data. First, it is used the *.split(',')* to split the values into a list of values using the comma as the separator. It also removes any leading or trailing white spaces using *strip()*.

Also, the first element, a time value, is appended to the list *timer_timer* to be used later on the code.

Then, conditional statements based on the presence of speed and acceleration were established:

- If both speed and acceleration are missing (empty strings), the code appends the previous (last) recorded values of speed and acceleration;
- If both speed and acceleration are present (non-empty), the code appends the current values of speed and acceleration. It also updates the variables *last_speed_value* and *last_acceleration_value* to the current speed and acceleration values to be used in case of missing data in subsequent lines;
- If acceleration is present and speed is missing, the code appends the previous (last) recorded value of speed and the current acceleration value. It also updates the *last_acceleration_value* to the current acceleration value;
- If acceleration is missing and speed is present, the code appends the current speed value and the previous (last) recorded value of acceleration. It also updates the *last_speed_value* to the current speed value.

4.3.3 Data Processing

4.3.3.1 K-Means Algorithm

```
# calculate_wcss function -> This function calculates intra-cluster sum of squares

def calculate_wcss(new_list):
  wcss = []
  for n in range (1, 5):
    kmeans = KMeans(n_clusters=n)
    kmeans.fit(new_list)
    wcss.append(kmeans.inertia_)

  return wcss
```

Figure 4.5: WCSS Code.

The Figure 4.5 represents the WCSS function (Within-Cluster Sum of Squares). The WCSS is a measure used to evaluate the clustering quality for different numbers of clusters.

It is done a loop between 1 and 5. This loop iterates from 1 to 4 (inclusive) to consider different numbers of clusters. This numbers were considered studying possibilities for good clusters. In the case of this file it was more than enough that interval. This values can be adapted if necessary.

For every stage of the loop the *fit* method is called in order to fit the K-means model to the data ad assigns each data point to its corresponding cluster. Then, it is called the *inertia_* attribute that gives the WCSS value for the current clustering. It represents the sum of squared distances between each data point and its assigned cluster's centroid. That WCSS value is appended to a list that will be returned from the function.

```
# The "Elbow Method" -> This function calculates the optimal number of clusters

def optimal_number_of_clusters(wcss):
  x1, y1 = 1, wcss[0]
  x2, y2 = 4, wcss[len(wcss)-1]

  distances = []

  for i in range(len(wcss)):
    x0 = i+1
    y0 = wcss[i]
    numerator = abs((y2-y1)*x0 - (x2-x1)*y0 + x2*y1 - y2*x1)
    denominator = sqrt((y2-y1)**2+(x2-x1)**2)
    distances.append(numerator/denominator)

  return distances.index(max(distances))+2
```

Figure 4.6: The "Elbow Method" Code.

The Figure 4.6 represents the "Elbow Method" whose functionality is to calculate the optimal number of clusters.

The function takes all the WCSS inputs from the previous function.

It is inicialized *x1* and *y1* with the values 1 and the first WCSS value that correspond to the first data point in the WCSS plot. The *x2* and *y2* are inicialized with the values 4 and the last WCSS value that correspond to the last data point in the WCSS plot. The values 1 and 4 are used as reference points to calculate the distances between data points and a line connecting these two reference points.

For each data point it is calculated the perpendicular distance from the data point to the line connecting the reference points previously described.

The numerator represents the absolute value of the cross-product between two vectors and the denominator represents the length of the line segment between the reference points. This is the formula to calculate the perpendicular distance from a point to a line.

After calculating all these distances, the function returns the index of the maximum distance plus 2. This value is added to account for the fact that the loops start at index 0 and clusters are numbered starting from 2 (since the optimal number of clusters is generally greater than or equal to 2 for this kind of data points.

```
def kmeans_algorithm(acceleration, speed, timer_time, fatigue_level, first_timer_time):
  global last_timer_time

  events_counter = 0

  # Variable used to the clustering result plots
  colors = ['r', 'g', 'b', 'y', 'c', 'm']

  # Creating a matrix that transposes and stores the arrays of acceleration and speed
  new_list = np.transpose(np.array([acceleration, speed]))

  # Checking if all the values are 0, if they are this fuction ends
  if np.all(np.isclose(new_list, 0)):
    return fatigue_level

  else:

    sum_of_squares = calculate_wcss(new_list)

    n = optimal_number_of_clusters(sum_of_squares)

    #Chamar o modelo KMeans
    kmeans = KMeans(n_clusters=n)

    label = kmeans.fit_predict(new_list)

    centroid = kmeans.cluster_centers_
```

Figure 4.7: K-means Algorithm Code.

The Figure 4.7 represents the k-means algorithm function with the classification method. Firstly, the function checks if the dataset contains only zeros. If all values are close to zero does not make any sense to perform the K-means algorithm (actually, the k-means can induce into wrong results) and the function returns the last value of fatigue calculated in the classification method. Then, it is called the function WCSS for different numbers of clusters and with the result finds the optimal number of clusters with the function previously described.

The function runs the K-means algorithm with the optimal number of clusters and uses the *fit_predict* method to perform the clustering. It assigns each data point to its corresponding cluster and returns a array of labels.

It is also acquired and stored the centroids using *kmeans.cluster_centers_* to be used in the classification method.

4.3.3.2 Features Engineering

```
# Features Enginnering

if max_speedfe < abs(speed[i]-centroid[label[i],1]):
  max_speedfe = abs(speed[i]-centroid[label[i],1])
  # Condition that guarantees that the maximum speed multplicative factor is stored with its inferior limit
  if speed[i] <= centroid[label[i],1]:
    max_speedfemf = speed[i]/centroid[label[i],1]
  if speed[i] > centroid[label[i],1]:
    max_speedfemf =centroid[label[i],1]/speed[i]

if max_accfe < abs(acceleration[i]-centroid[label[i],0]):
  max_accfe = abs(acceleration[i]-centroid[label[i],0])
```

Figure 4.8: Features Engineering Code.

The Figure 4.8 represents the code for the features engineering. Features Engineering is the process of creating new attributes/features from existing data to enhance the perform of a machine learning model.

The code is creating two features for the speed, the maximum speed difference between the data point and its centroid and the maximum speed multiplicative factor between the same points. Additionally, it creates another feature for the maximum acceleration difference between the data point and its centroid.

First, it's calculated the absolute difference between the current speed and the speed of the centroid of the associated cluster to that speed. If this difference is greater than the previously stored maximum speed difference, it becomes the new value for that variable.

Then, it's checked whether the current speed is less than or greater than the centroid speed to calculate the maximum speed multiplicative factor. If the current speed is less than or equal to the centroid speed, the maximum multiplicative factor will be the ratio between the current speed and the centroid speed. Otherwise, the maximum multiplicative factor will be the inverse ratio, in other words, the ratio between the centroid speed and the current value. The goal of this two last conditions is to guarantee that the stored value of maximum speed multiplicative factor is always the inferior limit of the interval.

After handling the speed-related features, the code than performs a similar process for the acceleration-related features. It calculates the absolute difference between the current acceleration and the acceleration of the centroid of the associated cluster. If this difference is greater than the previously stored maximum acceleration difference between the data point and its centroid, it becomes the new value for that variable.

By creating these features, the code aims to capture more relevant information from the data potentiating the improve of the machine learning model in predicting and classifying the driver's fatigue state. These features will be used in the subsequent steps of the algorithm to directly, in the case of the acceleration, and indirectly, in the case of the speed (used to validate the applied rule) contribute for the classification methodology.

The Figure 4.9 demonstrates an example of what can be obtained with this features:

```
Maximum speed difference between a data point and its centroid: 14.227055505283083 mph
Maximum multiplicative speed factor between a data point and its centroid: [0.5588074895473557, 1.7895250487963543]
Maximum acceleration difference between a data point and its centroid: 0.30031415971165953 g
```

Figure 4.9: Features Engineering Example.

4.3.4 Classification

```
# Classification Method
""" For the classification method it was considerer an value of +/- 0.3g for acceleration and +/- 7% for the velocity """

for i in range (len(speed)):
  if last_timer_time is None:
    last_timer_time = datetime.strptime(first_timer_time[i], '%H:%M:%S.%f')

  diff_time =  (datetime.strptime(timer_time[i], '%H:%M:%S.%f') - last_timer_time).total_seconds()

  if speed[i] > centroid[label[i], 1]*1.07 or speed[i] < centroid[label[i], 1]*0.93:
    if acceleration[i] > centroid[label[i], 0]+0.3 or acceleration[i] < centroid[label[i], 0]-0.3:
      if diff_time >= 3.0:
        fatigue_level = fatigue_level + 1

        if fatigue_level == 1:
          last_timer_time = datetime.strptime(timer_time[i], '%H:%M:%S.%f')
          print(timer_time[i])
          print("WARNING!\nThe driver shows possible signs of fatigue!\nTake a break!\n")

        if fatigue_level == 2:
          last_timer_time = datetime.strptime(timer_time[i], '%H:%M:%S.%f')
          print(timer_time[i])
          print("STOP!\nThe driver is fatigued!\nPark ASAP!\nRest for at least 15 to 20 minutes and ingest caffeine before restarting the trip!\n")
          fatigue_level = 0 #This equality takes on that the driver respects the advice"""

return fatigue_level
```

Figure 4.10: Classification Method Code.

The Figure 4.10 represents the code with the classification method that analyzes speed and acceleration data to identify potential signs of driver fatigue during a trip.

For the acceleration condition it was assumed that values between +/- 0.3 g (where g is the acceleration due to gravity) of the cluster centroid's acceleration is a sign that the driver is taking a trip without signs of fatigue.

For the speed it was considered values within +/- 7% of the cluster centroid's speed being normal. These condition was based on the "Regra dos 7" [24], a rule that determines the legal tolerance on the speed control. If the speed is between this percentage it's considered that the driver is having a normal trip without signs of fatigue.

For both this conditions it was used the features engineering as a base to the classification Method. For the acceleration, its value was defined using the analyses of the maximum acceleration value between the data point and its centroid and for the speed it was used as a confirmation for the condition created used per base the "Regra dos 7". It was resorted to other source files

to confirm and validate the features engineering and consequently the decision values used in the classification method.

First, the code checks if it's the first data point using the condition *if last_timer_time is None*. If this variable is not assigned yet takes the time value of the first data point stored in the variable **first_timer_time** defined on the main function.

Then, calculates the difference between the current data point and the previous data point where occurred a sign of fatigue storing it on the variable *diff_time*.

The code checks three conditions to identify potential driver fatigue:

- ***speed[i] > centroid[label[i], 1]*1.07 or *speed[i] < centroid[label[i], 1]*0.93** -> Checks if the speed deviates from the cluster centroid's speed by more than 7% in either direction;

- ***acceleration[i] > centroid[label[i], 0]*+0.3 or *acceleration[i] < centroid[label[i], 0]*-0.3** -> Checks if the acceleration deviates from the cluster centroid's acceleration by more than +/- 0.3 g;

- ***diff_time* >= 3.0** -> Checks if the variable *diff_time*, specified in the text above, is greater than or equal to 3 seconds (it is considered that if the variable is lower than the 3 seconds the trip belongs to the same fatigue event previously triggered).

If all these three conditions are met, the *fatigue_level* variable is incremented by 1, indicating a potential sign of driver fatigue.

Lastly, the code handle different fatigue levels:

- When the *fatigue_level* variable becomes 1, it means the driver is showing possible signs of fatigue and a warning message is printed;

- When the *fatigue_level* variable becomes 2, it means that the driver is fatigued and a more severe message is printed, advising the driver to stop for at least 15 to 20 minutes (value withdrawn from "Prevenção Rodoviária Portuguesa") and ingest caffeine before restarting the trip. The *fatigue_level* variable is the reset to 0, assuming that the driver follows the advice.

Every minute of data points, the function returns the fatigue level representing the overall of driver fatigue.

4.4 Results

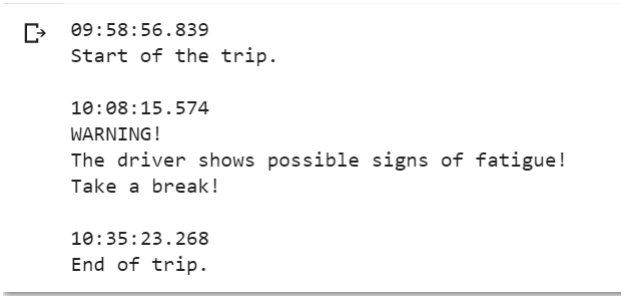


Figure 4.11: Result Example.

The Figure 4.11 is an example of how the information will be made available to the driver and corresponds to a stretch of the trip.

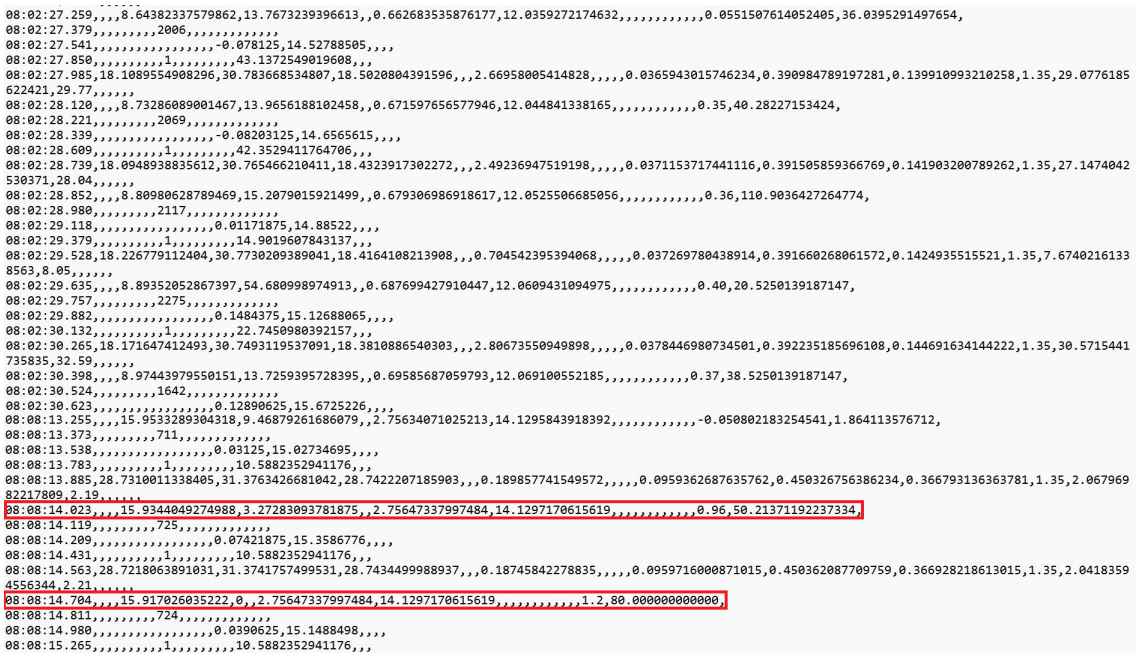
The trip started at 09:58:56.839 and the system correctly identified the start of the journey.

At 10:08:15.574, the software detected possible signs of driver fatigue and issued a warning message. It advised the driver to take a break, which demonstrates the software's capability to alert for potential safety concerns during the trip. As this is a database and not a real-life scenario, the driver did not stop the trip as recommended. However, the fatigue level does not evolve to another level of fatigue. This can mean that could just have been a distraction from the driver or a bad event classification. However the software was tested to other stretches and different files and all its functionality is working perfectly (it was possible to see the evolve of the state of fatigue).

The trip ended at 10:35:23.268 indicating that the software effectively tracked the journey's duration and provided valuable insights throughout the trip.

In order to demonstrate the proper functioning of all functionalities developed in the system were created different files with emulated data.

For this effect, and having the same file as base, the values of acceleration and speed were forced to fit the conditions of fatigue in some specific spots. All the values are completely out of reality and are exclusively used to verify and validate the presented system.



```
08:02:27.259,,,,8.64382337579862,13.7673239396613,,0.662683535876177,12.0359272174632,,,,,,,,,,,0.0551507614052405,36.0395291497654,
08:02:27.379,,,,,,,,,2006,,,,,,,,,,,,
08:02:27.541,,,,,,,,,,,,,,,,,-0.078125,14.52788505,,,,
08:02:27.850,,,,,,,,,1,,,,,,,,,43.1372549019608,,,
08:02:27.985,18.1089554908296,30.783668534807,18.5020804391596,,,2.66958005414828,,,,,0.0365943015746234,0.390984789197281,0.139910993210258,1.35,29.0776185
622421,29.77,,,,,,
08:02:28.120,,,,8.73286089001467,13.9656188102458,,0.671597656577946,12.044841338165,,,,,,,,,,,0.35,40.28227153424,
08:02:28.221,,,,,,,,,2069,,,,,,,,,,,,
08:02:28.339,,,,,,,,,,,,,,,,,-0.08203125,14.6565615,,,,
08:02:28.609,,,,,,,,,1,,,,,,,,,42.3529411764706,,,
08:02:28.739,18.0948938835612,30.765466210411,18.4323917302272,,,2.49236947519198,,,,,0.0371153717441116,0.391505859366769,0.141903200789262,1.35,27.1474042
530371,28.04,,,,,,
08:02:28.852,,,,8.80980628789469,15.2079015921499,,0.679306986918617,12.0525506685056,,,,,,,,,,,0.36,110.9036427264774,
08:02:28.980,,,,,,,,,2117,,,,,,,,,,,,
08:02:29.118,,,,,,,,,,,,,,,,,0.01171875,14.88522,,,,
08:02:29.379,,,,,,,,,1,,,,,,,,,14.9019607843137,,,
08:02:29.528,18.226779112404,30.7730209389041,18.4164108213908,,,0.704542395394068,,,,,0.037269780438914,0.391660268061572,0.1424935515521,1.35,7.6740216133
8563,8.05,,,,,,
08:02:29.635,,,,8.89352052867397,54.680998974913,,0.687699427910447,12.0609431094975,,,,,,,,,,,0.40,20.5250139187147,
08:02:29.757,,,,,,,,,2275,,,,,,,,,,,,
08:02:29.882,,,,,,,,,,,,,,,,,0.1484375,15.12688065,,,,
08:02:30.132,,,,,,,,,1,,,,,,,,,22.7450980392157,,,
08:02:30.265,18.171647412493,30.7493119537091,18.3810886540303,,,2.80673550949898,,,,,0.0378446980734501,0.392235185696108,0.144691634144222,1.35,30.5715441
735835,32.59,,,,,,
08:02:30.398,,,,8.97443979550151,13.7259395728395,,0.69585687059793,12.069100552185,,,,,,,,,,,0.37,38.5250139187147,
08:02:30.524,,,,,,,,,1642,,,,,,,,,,,,
08:02:30.623,,,,,,,,,,,,,,,,,0.12890625,15.6725226,,,,
08:08:13.255,,,,15.9533289304318,9.46879261686079,,2.75634071025213,14.1295843918392,,,,,,,,,,,-0.050802183254541,1.864113576712,
08:08:13.373,,,,,,,,,711,,,,,,,,,,,,
08:08:13.538,,,,,,,,,,,,,,,,,0.03125,15.02734695,,,,
08:08:13.783,,,,,,,,,1,,,,,,,,,10.5882352941176,,,
08:08:13.885,28.7310011338405,31.3763426681042,28.7422207185903,,,0.189857741549572,,,,,0.0959362687635762,0.450326756386234,0.366793136363781,1.35,2.067969
82217809,2.19,,,,,,
08:08:14.023,,,,15.9344049274988,3.27283093781875,,2.75647337997484,14.1297170615619,,,,,,,,,,,0.96,50.21371192237334,
08:08:14.119,,,,,,,,,725,,,,,,,,,,,,
08:08:14.209,,,,,,,,,,,,,,,,,0.07421875,15.3586776,,,,
08:08:14.431,,,,,,,,,1,,,,,,,,,10.5882352941176,,,
08:08:14.563,28.7218063891031,31.3741757499531,28.7434499988937,,,0.18745842278835,,,,,0.0959716000871015,0.450362087709759,0.366928218613015,1.35,2.0418359
4556344,2.21,,,,,,
08:08:14.704,,,,15.917026035222,0,,2.75647337997484,14.1297170615619,,,,,,,,,,,1.2,80.000000000000,
08:08:14.811,,,,,,,,,724,,,,,,,,,,,,
08:08:14.980,,,,,,,,,,,,,,,,,0.0390625,15.1488498,,,,
08:08:15.265,,,,,,,,,1,,,,,,,,,10.5882352941176,,,
```

Figure 4.12: Example of Emulated Data.

```
07:57:49.962
Start of the trip.

08:02:28.120
WARNING!
The driver shows possible signs of fatigue!
Take a break!

08:08:14.023
STOP!
The driver is fatigued!
Park ASAP!
Rest for at least 15 to 20 minutes and ingest caffeine before restarting the trip!

08:30:12.708
End of trip.
```

Figure 4.13: Result of the Emulated Data presented in Figure 4.12.

In Figure 4.13 it is possible to see the results of the input data presented in Figure 4.12. The results demonstrates the good functionality of the system evolving the fatigue level to a different stage where the driver is strongly advised to stop the vehicle and rest for at least 15 minutes.

Also, in Figure 4.12, it is possible to see 2 red rectangles. The rectangles represent two different events capable of trigger the conditions to increase the fatigue level. However, as can be verified in the Figure 4.13, the system does not recognize the event at 08:08:14.704 as a possible symptom of fatigue, proving that the system is working as it should due to the fact that was implemented in the source code that any event that occurs without 3 seconds having passed since the last one is considered the same event.

Chapter 5

Conclusions and Future Works

5.1 Conclusions

As referred in chapter 3, the throttle position variable had to be eliminated from the list of variables to be analyzed since that its value almost the time presents empty values. This occurrence can be result of an incorrect read on the sensor side (unlikely) or and indication that a speed control mechanism, such as cruise control, was being used.

The use of speed control mechanisms were not taken into consideration at the beginning of this study and without the variable in cause, the accuracy of the model decreased at the level of the clustering and also on the classification method. The project aimed to investigate the possibility of inferring the state of fatigue during driving using only the signals that characterize the vehicle's kinematics (low-budget approach).

It was decided to use the K-means algorithm instead of the Supervised Artificial Neural Network because this last one presented an accuracy value extremely low (in order of 5%) despite all the attempts of changing:

- The number of neurons in the hidden layer;
- The number of hidden layers;
- The loss function used (tried with Log-Cosh loss function [25]);
- The activation functions used (tried with the Sigmoid activation function).

The Artificial Neural Networks need an abundance of data values in order to be perfectly well trained, otherwise, it wouldn't work and despite the fact that the source file used was the one who had more data it was concluded that hadn't enough to elaborate the correct train of the Supervised Artificial Neural Network created.

Another aspect to consider is that the software assumes that the driver will follow the fatigue indications provided. In other words, the driver is expected to immobilize the vehicle and rest when instructed by the software. This detail is very important for the fatigue level since that the

code, when the driver hits the worrisome level of fatigue the variable *fatigue_level* restarts its value.

However, the software has shown promising functionality in analyzing speed and acceleration to detect fatigue-related patterns and enhance road safety. Its ability to identify potential signs of fatigue and provide timely warnings can contribute significantly to preventing accidents and encouraging responsible driving behavior. However, further testing and refinement may be necessary to ensure the software's accuracy and reliability in various driving scenarios.

5.2 Future Works

As future work, the inclusion of new variables and the creation of a more extensive data set are essential to refine the algorithm.

The features engineering were created but actually are not used, at least not directly, in the classification methodology. In the future should be tried to use it and develop it more. As example. in the features engineering, could be calculated how much times the speed and acceleration values are close to the maximum distance between the data point and the centroid and try to use this value directly in the classification methods as a component of the rules established.

On another view, should be made real-life tests using the OBD scanner such as the ELM 327 in order to improve the classification methods. First, should be tried to create an algorithm cable of reading the OBDII protocol. Secondly, for the tests could be done something like make a trip without noticeable signs of fatigue and another one considering the presence of some fatigue symptoms, in a safe environment and accompanied with someone else in order to guarantee the safety. Then, can be made a simple script capable of compare the values of speed and acceleration (and other variables if they exist) and with that have a more reliable criteria to use in the classification.

As a closing note, the creation of a mobile application capable of transmit the messages to the driver in real-time would be "the cherry on the top of the cake", unfortunately, there was no time for that, so, should be take into account for future works.

References

[1] Fatigue prevention. URL: https://prp.pt/portfolio-items/fadiga/#:~:text=O%20que%20fazer%20para%20evitar%20a%20Fadiga&text=Ter%20em%20aten%C3%A7%C3%A3o%20que%20determinados,ve%C3%ADculo%20e%20fazer%20alguns%20movimentos.

[2] W. Wang, J. Xi, and H. Chen. Modeling and recognizing driver behavior based on driving data: A survey. *Mathematical Problems in Engineering*, 2014.

[3] W. Wang, J. Xi, A. Chong, and L. Li. Driving style classification using a semi-supervised support vector machine. *IEEE Transactions on Human-Machine Systems*, 2017.

[4] W. Wang, C. Liu, and D. Zhao. How much data are enough? a statistical approach with case study on longitudinal driving behavior. *IEEE Transactions on Intelligent Vehicles*, 2(2):85–98, 2015.

[5] W. Wang, J. Xi, and D. Zhao. Driving style analysis using primitive driving patterns with bayesian nonparametric approaches. *IEEE Transactions on Intelligent Transportation Systems*, 2015.

[6] J. Pinto, J. Cardoso, A. Lourenço, and C. Carreiras. Towards a continuous biometric system based on ecg signals acquired on the steering wheel. *Sensors*, 17(10):2228, 2017. doi:10.3390/s17102228.

[7] C. S. Pinto, J. Cardoso, A. Lourenço, and C. Ahlstrom. The importance of subject-dependent classification and imbalanced distributions in driver sleepiness detection in realistic conditions. *IET Intelligent Transport Systems*, October 2018.

[8] F. Pilkington-Cheney and et al. The i-dreams intervention strategies to reduce driver fatigue and sleepiness for different transport modes. In *7th International IEEE Conference on Models and Technologies for Intelligent Transportation Systems*, June 2021.

[9] Interior camera uses ai to detect driver distraction and fatigue. URL: https://www.eenewseurope.com/en/interior-camera-uses-ai-to-detect-driver-distraction-and-fatigue.

[10] A review of heartbeat detection systems for automotive applications. URL: https://www.mdpi.com/1424-8220/21/18/6112.

[11] Elm327 mini. URL: https://www.olx.pt/d/anuncio/ficha-ligao-mini-obd-2-obd-ii-obd2-elm-327-elm327-v2-1-bluetooth-IDvhGIJ.html.

[12] Python and google colaboratory. URL: https://miteshparmar1.medium.com/structure-your-code-better-in-google-colab-with-text-and-code-cells-b6fa73feec20.

[13] C. Jacobé de Naurois, C. Bourdin, A. Stratulat, E. Diaz, and J. L. Vercher. Detection and prediction of driver drowsiness using artificial neural network models. *Accident Analysis and Prevention*, 2019.

[14] Sang-Ho Jo, Jin-Myung Kim, and Dong Kyoo Kim. Heart rate change while drowsy driving. *Journal of Physiology & Pathophysiology*, 12:29–38, 2021.

[15] What does obd stand for? URL: https://www.noregon.com/what-is-obd.

[16] Supervised neural networks - scikit-learn. URL: https://scikit-learn.org/stable/modules/neural_networks_supervised.html.

[17] The softmax function and its derivative. URL: https://eli.thegreenplace.net/2016/the-softmax-function-and-its-derivative/.

[18] Feedforward vs backpropagation - linkedin. URL: https://www.linkedin.com/pulse/feedforward-vs-backpropagation-ann-saffronedge1.

[19] How does the gradient descent algorithm work in machine learning? URL: https://www.analyticsvidhya.com/blog/2020/10/how-does-the-gradient-descent-algorithm-work-in-machine-learning/.

[20] K-means clustering algorithm: Applications, evaluation methods, and drawbacks. URL: https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a.

[21] Google colaboratory. URL: https://colab.research.google.com.

[22] Google colaboratory - faq. URL: https://research.google.com/colaboratory/faq.html.

[23] Automotive data set. URL: http://apmonitor.com/pds/uploads/Main/automotive.txt.

[24] Regra dos 7: A tolerância dos radares. URL: https://automais.autosport.pt/noticias/regra-dos-7-a-tolerancia-dos-radares/.

[25] Common loss functions used in machine learning. URL: https://builtin.com/machine-learning/common-loss-functions.

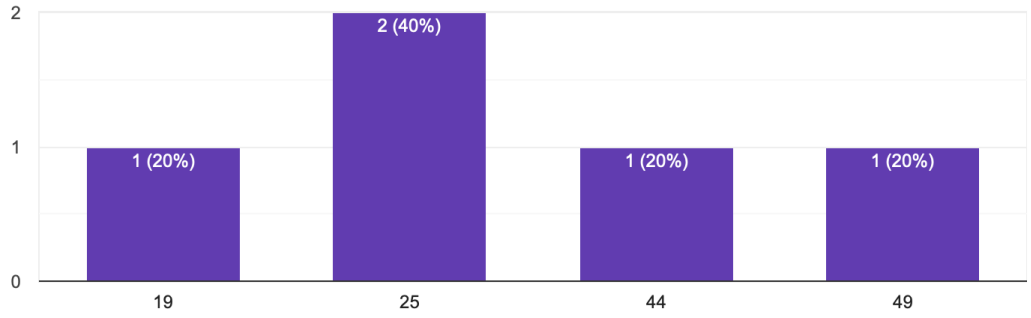
Appendix A

Quiz - Analysis and study of fatigue in driving

Idade:

5 respostas

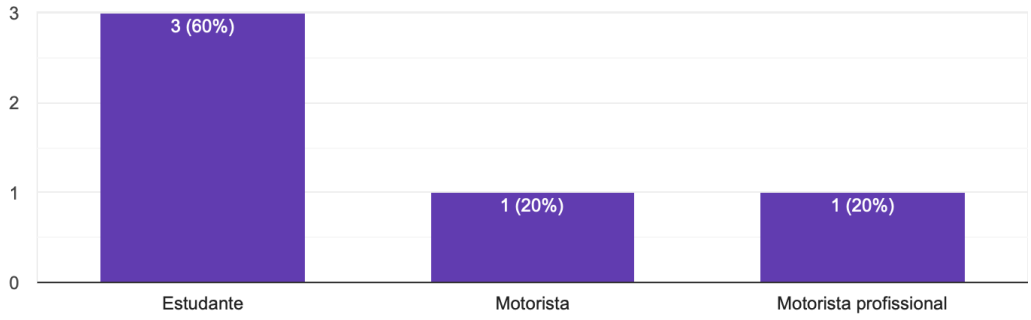
Copiar



Profissão:

Copiar

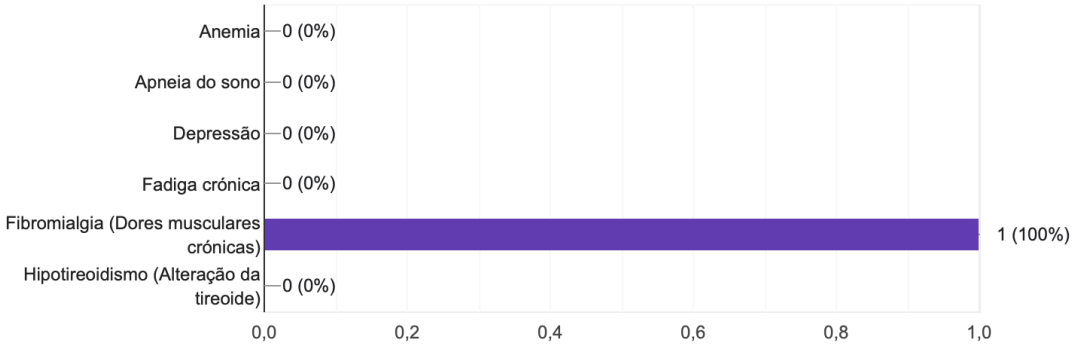
5 respostas



Já lhe diagnosticada alguma das seguintes condições de saúde?

Copiar

1 resposta



Alguma vez experienciou um cenário de fadiga/sonolência em primeira pessoa, ou presenciou uma situação semelhante com algum colega/amigo?

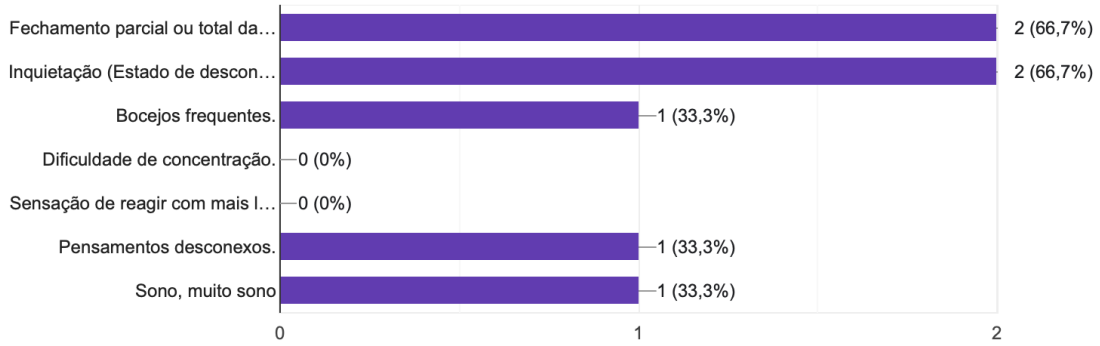
Copiar

5 respostas

Se respondeu sim à pergunta anterior, quais foram os sintomas apresentados pelo condutor? (No caso de ter respondido não, deve passar diretamente para a submissão do formulário, obrigado pelo tempo disponibilizado!)

Copiar

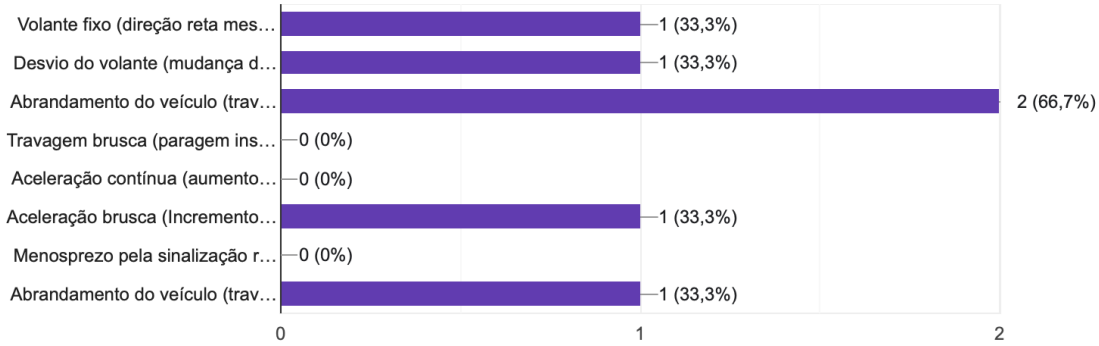
3 respostas



Quais as consequências, a nível da condução, visíveis perante a situação de fadiga/sonolência apresentadas pelo condutor?

Copiar

3 respostas



Appendix B

Supervised Artificial Neural Network Code

```
#ANN (Artificial Neural Network)
np.random.seed(42)
N_INPUTS = 30 #Mais ou menos 5s

#MSE loss function
def mse_loss(y_true,y_pred):
  return ((y_true-y_pred)**2).mean()

#ReLu activation function
def relu(x):
  if(x>0):
    return x
  else:
    return 0

#Derivative of ReLu
def deriv_relu(x):
  if(x>0):
    return 1
  else:
    return 0

#SoftMax activation function
def softmax(vector):
  return exp(vector)/exp(vector).sum()


#Derivative of Softmax
def deriv_softmax(vector):
  return (softmax*np.identity(softmax.size)-softmax.transpose()@softmax)


class NeuralNetwork:
  #Structure of the neural network:
    #- N_INPUTS inputs.
    #- a hidden layer with 5 neurons (h1, h2, h3, h4, h5).
    #- an output layer with 1 neurons (o1).
```

```
def __init__(self):
  #Biases
  self.b1=np.random.random()
  self.b2=np.random.random()
  self.b3=np.random.random()
  self.b4=np.random.random()
  self.b5=np.random.random()
  self.bo=np.random.random()

  #Weights
  self.w1o, self.w2o, self.w3o, self.w4o, self.w5o = np.random.random(5)
  self.wi1=np.random.random(N_INPUTS)
  self.wi2=np.random.random(N_INPUTS)
  self.wi3=np.random.random(N_INPUTS)
  self.wi4=np.random.random(N_INPUTS)
  self.wi5=np.random.random(N_INPUTS)

def feedforward(self,x):
  #-x is a numpy array with N_INPUTS elements.

  #Hidden layer
  self.sum_h1=np.dot(self.wi1,x)+self.b1
  self.sum_h2=np.dot(self.wi1,x)+self.b2
  self.sum_h3=np.dot(self.wi1,x)+self.b3
  self.sum_h4=np.dot(self.wi1,x)+self.b4
  self.sum_h5=np.dot(self.wi1,x)+self.b5

  self.h1=relu(self.sum_h1)
  self.h2=relu(self.sum_h2)
  self.h3=relu(self.sum_h3)
  self.h4=relu(self.sum_h4)
  self.h5=relu(self.sum_h5)

  #Output layer
  self.sum_o1=self.w1o*self.h1+self.w2o*self.h2+self.w3o*self.h3+self.w4o*self.h4+self.w5o*self.h5+self.bo
  self.o1=softmax(self.sum_o1)
  return self.o1
```

```
def train(self, data, y_trues, learn_rate=0.1, epochs=500):
  #-data is a (n x N_INPUTS) numpy array, n = # of samples in the dataset.
  #-y_trues is a numpy array with n elements.
  #Elements in y_true correspond to those in data.

  loss_prev=10000 #loss_prev is the loss of the previous iteration
  for epoch in range(epochs):
    for x,y_true in zip(data,y_trues):

      #1. Feedforward Step
      y_pred=self.feedforward(x)

      #2. Backpropagation Step
      #Partial derivates:
      d_L_d_ypred=-2*(y_true-y_pred)

      #Output layer: Neuron o1
      d_ypred_d_w1o=self.h1*deriv_softmax(self.sum_o1)
      d_ypred_d_w2o=self.h2*deriv_softmax(self.sum_o1)
      d_ypred_d_w3o=self.h3*deriv_softmax(self.sum_o1)
      d_ypred_d_w4o=self.h4*deriv_softmax(self.sum_o1)
      d_ypred_d_w5o=self.h5*deriv_softmax(self.sum_o1)

      d_ypred_d_bo=self.h1*deriv_softmax(self.sum_o1)

      d_ypred_d_h1=self.w1o*deriv_softmax(self.sum_o1)
      d_ypred_d_h2=self.w2o*deriv_softmax(self.sum_o1)
      d_ypred_d_h3=self.w3o*deriv_softmax(self.sum_o1)
      d_ypred_d_h4=self.w4o*deriv_softmax(self.sum_o1)
      d_ypred_d_h5=self.w5o*deriv_softmax(self.sum_o1)

      #Hidden layer: Neuron h1
      d_h1_d_wi1=x*deriv_relu(self.sum_h1)
      d_h1_d_b1=deriv_relu(self.sum_h1)

      #Hidden layer: Neuron h2
      d_h2_d_wi2=x*deriv_relu(self.sum_h2)
      d_h2_d_b2=deriv_relu(self.sum_h2)
```

```
            #Hidden layer: Neuron h3
            d_h3_d_wi3=x*deriv_relu(self.sum_h3)
            d_h3_d_b3=deriv_relu(self.sum_h3)

            #Hidden layer: Neuron h4
            d_h4_d_wi4=x*deriv_relu(self.sum_h4)
            d_h4_d_b4=deriv_relu(self.sum_h4)

            #Hidden layer: Neuron h5
            d_h5_d_wi5=x*deriv_relu(self.sum_h5)
            d_h5_d_b5=deriv_relu(self.sum_h5)

            #3. Gradient Descent
            #Output layer: Neuron o1
            self.w1o-=learn_rate*d_L_d_ypred*d_ypred_d_w1o
            self.w2o-=learn_rate*d_L_d_ypred*d_ypred_d_w2o
            self.w3o-=learn_rate*d_L_d_ypred*d_ypred_d_w3o
            self.w4o-=learn_rate*d_L_d_ypred*d_ypred_d_w4o
            self.w5o-=learn_rate*d_L_d_ypred*d_ypred_d_w5o

            self.bo-=learn_rate*d_L_d_ypred*d_ypred_d_bo

            #Hidden layer: Neuron h1
            self.wi1-=learn_rate*d_L_d_ypred*d_ypred_d_h1*d_h1_d_wi1
            self.b1-=learn_rate*d_L_d_ypred*d_ypred_d_h1*d_h1_d_b1

            #Hidden layer: Neuron h2
            self.wi2-=learn_rate*d_L_d_ypred*d_ypred_d_h2*d_h2_d_wi2
            self.b2-=learn_rate*d_L_d_ypred*d_ypred_d_h2*d_h2_d_b2

            #Hidden layer: Neuron h3
            self.wi3-=learn_rate*d_L_d_ypred*d_ypred_d_h3*d_h3_d_wi3
            self.b3-=learn_rate*d_L_d_ypred*d_ypred_d_h3*d_h3_d_b3

            #Hidden layer: Neuron h4
            self.wi4-=learn_rate*d_L_d_ypred*d_ypred_d_h4*d_h4_d_wi4
            self.b4-=learn_rate*d_L_d_ypred*d_ypred_d_h4*d_h4_d_b4
```

```
          #Hidden layer: Neuron h5
          self.wi5-=learn_rate*d_L_d_ypred*d_ypred_d_h5*d_h5_d_wi5
          self.b5-=learn_rate*d_L_d_ypred*d_ypred_d_h5*d_h5_d_b5

        #4. Performance assessment (per epoch)
        if epoch%5==0:
          y_preds=np.apply_along_axis(self.feeddorward,1,data)
          loss=mse_loss(y_trues,y_preds)
          print("Epoch %d: Loss: %.2f" %(epoch,loss))
        if loss>loss_prev:
          if learn_rate>0.002:
            learn_rate=learn_rate*.9 #decrease 90% of the previous value
            print("The epoch", epoch, "has new learn_rate: ", learn_rate)
            loss_prev=loss

#Create the ANN
model=NeuralNetwork()
```

Appendix C

Source Code

```
import urllib.request
import warnings
warnings.filterwarnings('ignore')
from datetime import datetime, timedelta
import matplotlib.pyplot as plt

def save_N_minute_values(url, filename, acquisition_time, position):
  end_of_file = 0

  # File Download
  urllib.request.urlretrieve(url, filename)

  # List to store the wanted values
  values_to_save = []

  # Opening the file to read
  with open(filename,'r') as file:
    lines = file.readlines()

  # Analysing the lines from the file
  for line in lines[position:]:

    # Getting the time values as a number
    time_string = line.split(',')[0]

    # Converting a time string to an object of datetime
    time_value = datetime.strptime(time_string, '%H:%M:%S.%f')

    # Calculating time in seconds since the first line of that stretch
    elapsed_time = (time_value - datetime.strptime(lines[position].split(',')[0], '%H:%M:%S.%f')).total_seconds()

    # Getting N minutes of values (in seconds)
    if elapsed_time <= acquisition_time * 60:
      values_to_save.append(line)
```

```
# Kmeans Clustering Method with Fatigue Classification Method

from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import numpy as np
from numpy import *

# calculate_wcss function -> This function calculates intra-cluster sum of squares

def calculate_wcss(new_list):
  wcss = []
  for n in range (1, 5):
    kmeans = KMeans(n_clusters=n)
    kmeans.fit(new_list)
    wcss.append(kmeans.inertia_)

  return wcss

# The "Elbow Method" -> This function calculates the optimal number of clusters

def optimal_number_of_clusters(wcss):
  x1, y1 = 1, wcss[0]
  x2, y2 = 4, wcss[len(wcss)-1]

  distances = []
```

```
# Checking if the file ended
if position == len(lines):
  end_of_file = 1

return values_to_save, end_of_file
```

```
[ ]     for i in range(len(wcss)):
          x0 = i+1
          y0 = wcss[i]
          numerator = abs((y2-y1)*x0 - (x2-x1)*y0 + x2*y1 - y2*x1)
          denominator = sqrt((y2-y1)**2+(x2-x1)**2)
          distances.append(numerator/denominator)

        return distances.index(max(distances))+2

      # kmeans_algorithm -> Includes the kmeans algorithm and the classifiction method
      def kmeans_algorithm(acceleration, speed, timer_time, fatigue_level, first_timer_time):
        global last_timer_time
        global max_speedfe
        global max_accfe
        global max_speedfemf

        events_counter = 0

        # Variable used to the clustering result plots
        colors = ['r', 'g', 'b', 'y', 'c', 'm']

        # Creating a matrix that transposes and stores the arrays of acceleration and speed
        new_list = np.transpose(np.array([acceleration, speed]))

        # Checking if all the values are 0, if they are this fuction ends
        if np.all(np.isclose(new_list, 0)):
          return fatigue_level

        else:

          sum_of_squares = calculate_wcss(new_list)

          n = optimal_number_of_clusters(sum_of_squares)

          #Chamar o modelo KMeans
          kmeans = KMeans(n_clusters=n)
```

```
label = kmeans.fit_predict(new_list)

centroid = kmeans.cluster_centers_

# This next part was used to determine the acceleration value for the classification rules together with data from diferent archieves
"""for i in range(n):
  cluster_points = new_list[label==i]  # pontos pertencentes ao cluster i
  plt.scatter(cluster_points[:, 0], cluster_points[:, 1], c=colors[i], label=f'Cluster {i}')
  plt.scatter(centroid[i, 0], centroid[i, 1], c='black', marker='x')  # centróide do cluster i

plt.xlabel('X')
plt.ylabel('Y')
plt.legend()
plt.show()"""

# Classification Method
""" For the classification method it was considerer an value of +/- 0.3g for acceleration and +/- 7% for the velocity """

for i in range (len(speed)):
  if last_timer_time is None:
    last_timer_time = datetime.strptime(first_timer_time[i], '%H:%M:%S.%f')

  diff_time =  (datetime.strptime(timer_time[i], '%H:%M:%S.%f') - last_timer_time).total_seconds()

  # Features Enginnering

  if max_speedfe < abs(speed[i]-centroid[label[i],1]):
    max_speedfe = abs(speed[i]-centroid[label[i],1])
    # Condition that guarantees that the maximum speed multplicative factor is stored with its inferior limit
    if speed[i] <= centroid[label[i],1]:
      max_speedfemf = speed[i]/centroid[label[i],1]
    if speed[i] > centroid[label[i],1]:
      max_speedfemf =centroid[label[i],1]/speed[i]

  if max_accfe < abs(acceleration[i]-centroid[label[i],0]):
    max_accfe = abs(acceleration[i]-centroid[label[i],0])
```

```
  if speed[i] > centroid[label[i], 1]*1.07 or speed[i] < centroid[label[i], 1]*0.93:
    if acceleration[i] > centroid[label[i], 0]+0.3 or acceleration[i] < centroid[label[i], 0]-0.3:
      if diff_time >= 3.0:
        fatigue_level = fatigue_level + 1

        if fatigue_level == 1:
          last_timer_time = datetime.strptime(timer_time[i], '%H:%M:%S.%f')
          print(timer_time[i])
          print("WARNING!\nThe driver shows possible signs of fatigue!\nTake a break!\n")

        if fatigue_level == 2:
          last_timer_time = datetime.strptime(timer_time[i], '%H:%M:%S.%f')
          print(timer_time[i])
          print("STOP!\nThe driver is fatigued!\nPark ASAP!\nRest for at least 15 to 20 minutes and ingest caffeine before restarting the trip!\n")
          fatigue_level = 0 #This equality takes on that the driver respects the advice"""

return fatigue_level
```

```
# Main Function

from datetime import datetime
import re
from google.colab import files

url = 'http://apmonitor.com/pds/uploads/Main/automotive.txt'
filename = 'automotive.txt'
acquisition_time = 1
position = 40000
last_acceleration_value = 0
last_speed_value = 0
fatigue_level = 0
last_timer_time = None
fatigue_level = 0
start_print = 0
max_speedfe = 0
max_accfe = 0
max_speedfemf = 0

while True:
  acceleration = []
  speed = []
  timer_time = []
  first_timer_time = []

  values, end_of_myprogramme = save_N_minute_values(url, filename, acquisition_time, position)

  for valor in values:
    if valor != '':
      match = re.search(r'\d{2}:\d{2}:\d{2}\.\d+', valor)
      if match:
        first_timer_times = match.group()
        first_timer_time.append(first_timer_times)
```

```
  if not start_print:
    print(first_timer_time[0])
    print("Start of the trip.\n")
    start_print = 1

  for line in values:

    line_values = line.strip().split(',')

    timer_time.append(line_values[0])

    if line_values[-3] == '' and line_values[-2] == '':
      speed.append(last_speed_value)
      acceleration.append(last_acceleration_value)

    if line_values[-3] != '' and line_values[-2] != '':
      speed.append(float(line_values[-2]))
      last_speed_value = float(line_values[-2])
      acceleration.append(float(line_values[-3]))
      last_acceleration_value = float(line_values[-3])

    if line_values[-3] != '' and line_values[-2] == '':
      speed.append(last_speed_value)
      acceleration.append(float(line_values[-3]))
      last_acceleration_value = float(line_values[-3])

    if line_values[-3] == '' and line_values[-2] != '':
      speed.append(float(line_values[-2]))
      last_speed_value = float(line_values[-2])
      acceleration.append(last_acceleration_value)

  fatigue_level = kmeans_algorithm(acceleration, speed, timer_time, fatigue_level, first_timer_time)

  if timer_time != []:
    end_time = timer_time
```

```
[ ]
  if end_of_myprogramme == 1:
    print(end_time[-1])
    print("End of trip.")
    print("Maximum speed difference between a data point and its centroid:", max_speedfe, "mph")
    print("Maximum speed multiplicative factor between a data point and its centroid: [{}, {}]".format(max_speedfemf, (1/max_speedfemf)))
    print("Maximum acceleration difference between a data point and its centroid:", max_accfe, "g")
    break


  else:
    position = position + len(speed)
```