FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Drive Companion

Luis Paulo Cirne Saldanha Durão

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Faculty Supervisor: Luís Filipe Pinto de Almeida Teixeira

Fraunhofer Supervisor: Tiago Rocha

September 18, 2018

Abstract

Driving is a very dangerous activity, both for the people inside and outside vehicles. This is because humans are fallible. For this reason, the analysis of the driving risk is an important matter, both for the driver and for public safety in general. The goal of this project was to develop a framework to analyse the risk of a person's driving style using Machine Learning techniques so that the application would improve its accuracy and use cases range over time. The analysis was done using the inertial sensors of a smartphone and the On-Board Diagnosis, OBD, to get parameters about the car's real-time state like speed, motor's RPM and engine load. Using this data, it is possible to detect features of the driving which can then be used to classify the user's driving style in one of 3 classes: low, medium and high risk. One of the challenges in this project is the fact that there are no datasets available with all these features and no pre-trained model for this kind of data. This means that all the data that was used had to be gathered and partially labelled by me and one volunteer. Such conditions resulted in a small and unbalanced dataset which crippled the results. It was, however possible to do a proof of concept of risky behaviours/features' detection.

Acknowledgements

I owe a big Thank You to my parents for paying my university fees and supporting me and another big Thank You to my sister who is always able to brighten my day with her positivity and for taking the time from one of her afternoons in the exams season to help me label a part of the dataset.

About my little brother... What can I really say? He is my biggest reason for getting up in the morning and face the day. I want to be successful so I can inspire him to be successful himself. He gives me the most love anyone gives me. He's the most important person in my life and his visits while I was writing this thesis and his company in between made it all more bearable.

I also am super grateful for my two best friends, Miguel and Marta. Also, they helped me review this document's grammar and structure before I submitted!

My squad also deserves my thanks, not because of any particular role in my dissertation but because of being such good friends throughout the whole course. We are now friends for life! Thanks Narciso, Landinha and Gonçalo!

Thank you, professor Luís Teixeira, for being one of the coolest teachers I've had and helping me navigate through the plethora of different algorithms and procedures of Machine Learning (that I knew nothing about and sill mostly don't because it's too big of a subject to learn by myself in just 4 months).

A huge thank you to professor André Restivo who helped me a lot even though he wasn't even my supervisor. I'm very grateful for you help and advices!

Eng. Tiago Rocha was especially helpful getting me to kickstart the project. Also he was the biggest contributor to the database. Thank you, Tiago.

Last but not least, I'd like to thank my arms for always being by my side, my legs for always supporting me, my fingers because I could always count on them and my chair for always having my back.

Luis Paulo Durão

"I see the beauty in the rules, the invisible code of chaos,
Hiding behind the menacing face of order"

Eliot Alderson

Contents

List of Figures

List of Tables

Symbols and Abbreviations

AI	Artificial Intelligence
UBI	Usage-Based Insurance
CNN	Convolutional Neural Network
CRISP-DM	Cross-Industry Standard Process for Data Mining
DC	Drive Companion
DS	Driving Style
FEUP	Faculdade de Engenharia da Unversidade do Porto
GPS	Global Positioning System
HLF	High-Level Features
LLF	Low-Level Features
MLF	Mid-Level Features
NN	Neural Network
OBD	On-Board Diagnostics
PID	Parameter Identifiers
REST	Representational State Transfer
UP	Universidade do Porto
SPP	Serial Port Profile

Chapter 1

Introduction

1.1 Problem Statement and Motivation

Driving has, for long, been the most dangerous transportation mean of all. In 2016, there was only one general aviation accident involving aircraft with a MTOM (maximum take-off mass) above 2250 kg and in 2013, a specially bad year for aviation accidents [3] (from 2006 to 2013, there had never been more than 5 fatalities per year), air transportation had a total of 11 fatalities [3] while road transportation recorded a mind-boggling number of 25401 fatalities [4], all in the EU-28 territory.

This is mainly due to human errors and the variety of choices the driver can take (when compared to rail transportation, for example, in which, practically, the only action that a driver can commit that can endanger the lives of the passengers is speeding and there are fewer train drivers, which leads to a finer selection and greater control over them). Hence, the analysis of a driver's DS, specifically its risk, is a very pertinent matter.

Over the last few years, there has been some research on detecting some mid-level features of a DS, like Rui Araujo, Angela Igreja and Ricardo de Castro's *Driving coach: A smartphone application to evaluate driving efficient patterns* [5] which is specified in analysing the fuel consumption of a driver and giving hints on how to improve the DS in order to increase the energetic efficiency, but none has yet been made, to the best of my knowledge, that uses an neural network approach (related work will be discussed in section 2.5).

In this context, this dissertation's theme was chosen because it is a groundbreaker in the usage of neural networks for DS risk analysis.

The possibilities of usages for this application are plenty and range from Usage-Based Insurance (UBI) to the analysis of a driver's DS after a revoked license due to past transgressions in order to ensure the driver's new DS is safe enough to acquire a permanent license again.

1.2 Objectives

The objectives of this project are to research the use of NNs (Neural Networks) in the analysis of driving data. This means analysing the applicability of NNs in:

1. detecting mid-level features (MLF) from low-level inputs/features (LLF)
2. detecting high-level features (HLF):
   (a) using Deep Neural Networks (inputting LLF directly into it and getting HLF in the output)
   (b) inputting LLF along with MLF into a NN to get the HLF

The deep neural network approach requires a big dataset which is why the alternative *(a)*, from here on out called the *mixed approach*, is the only feasible one in the context of this dissertation due to the the lack of data.

Low-Level Features are the data acquired directly from the smartphone application, like *accelerometer* and *engine's RPM*.

Mid-Level Features are features in the driving conditions that are derived from the LLF. These can be classes like *accelerating*, *slowing down*, *on a turn* or *on a straight road section*.

High-Level Features, in this context, are the classes that represent different risk levels.

1.3 Research questions

This dissertation aims to answer the following research questions:

- Is it possible to collect, analyse and classify vehicle driving data in real-time using a smartphone?
- Is it possible to tell dangerous situations apart from normal driving style using such a system?

In section 4.3 there is a discussion over these questions.

1.4 Project Overview

Detecting the DS risk from a set of raw sensorial and OBD data requires several steps of data pre-treatment, detecting features, labelling, clustering and training the model (not necessarily all or in that order).

The diagram that represents the overall process is represented in the image 1.1. It follows a simplified version of the CRISP-DM methodology, which will be explained in section 2.3.

First, the developed Android App acquires data and sends it raw into the remote DB after each session. Then, when there is enough data in the DB to work with, new data is inferred from the raw

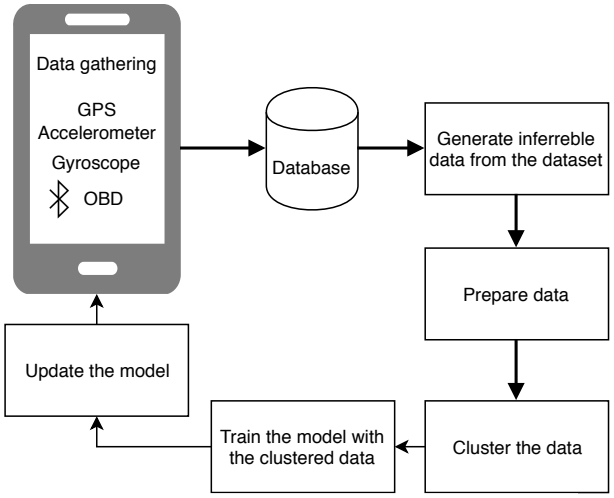


Figure 1.1: System overview

data like the standard deviation of the measurements in each time frame, as well as the average of the gyroscope and accelerometer data in a time frame to remove the noise. Next, clustering algorithms will be used to cluster the data and the one with the best performance will be used in the mixed approach [1].

After being clustered, the data will be used to train the model until it reaches a maximum in the fitness function. Then, the new model will be uploaded to the App so that it contains the most recent version.

The lack of data and the small variation in it, which creates a big challenge in clustering the data into meaningful clusters, was the biggest problem in this project. If the data is scarce, the clusters will be biased. If the data is not diversified enough, the clusters will aggregate noise instead of actual meaningful clusters.

Because there was only one functional OBD device [2] during most of the 5 months of this project, the gathered data is extremely biased and in small quantity, which means that the data analysis part of this thesis was negatively affected by factors over which I had little control.

Another fundamental problem is the subjectivity of the definition of risk without enormous collective datasets (on Google scale) so a correlation could be made between the DS and accidents, making it possible to actually predict the probability of an accident (quantitative risk) of a DS in real time.

To simplify this conundrum, an assumption was made: **the risk is to be evaluated as belonging to one of three qualitative (instead of quantitative) classes, high, medium and low risk, depending on the dangerous mid-level features the DS is classified as in each time-frame**.

[1] see section 1.2
[2] the other OBD devices only arrived 6 days from the delivery date of this thesis report

1.5 Contributions

The biggest contribution of this dissertation was the designing of the model, which, along with the App, can be later reused by researchers who intend to deepen the study on DS risk analysis.

Another contribution is assessing the effectiveness of clustering algorithms for non-visual data, more specifically, driving data.

1.6 Document Structure

First of all, I'll clarify that the language used in this dissertation is British English, not American English.

Chapter 2 describes some important concepts and processes that are relevant or fundamental in this project. It also mentions some of the related work that was previously done about this topic.

Chapter 3 contains the implementation details of the project.

Chapter 4 contains the results of the project.

Chapter 5 contains the conclusions and future work.

Chapter 2

Concepts and Related Work

2.1 Data gathering

Gathering the data was one of the biggest tasks in this thesis because there was no dataset available with the fields that were required to analyse the driving risk. For this reason, there was the need to build a way to gather the data and centralise it.

An Android App was firstly developed that only gathered data from the inertial sensors (accelerometer and gyroscope), the On-Board Diagnostics EML327 [6] device and the location. The App saves the data in a local database and then uploads it to a back-end server as soon as it's possible to do so.

2.1.1 On-Board Diagnostics

On-Board Diagnostics v2 is protocol defined in ISO9141 [7, 8], SAE J1962 [9], SAE J1978 [10], SAE J1979 [2] and SAE J2012 [11] and is the standardised way to collect data from the vehicle about errors, vehicle information and live data on the vehicle's operation like engine's rotations per minute, vehicle speed and throttle position. The only OBD mode used by the App is Mode 1 (live data mode).

An ELM371 [6] Bluetooth adapter was used to communicate with the vehicle's OBD port. ELM371 works as a middle-ware layer between the low-level commands of the OBD port and the RS232, a common serial communication standard. The Bluetooth adapter wraps the RS232 interface in the *Bluetooth Serial Port Profile* (Bluetooth SPP) which is based on *Bluetooth RFCOMM*, which, in turn, is based on the *ETSI 07.10* [12] protocol. The App then connects to this device via a Bluetooth SPP socket. There is a library that can be used to collect the OBD data: OpenXC. However, this library wasn't used because the command interface of the ELM327 is very easy to use and the library would be overkill. Simply using a Bluetooth Socket for sending and receiving data to and from the device was enough. Besides, as the library's website even says,

> Although Ford does implement the largest subset of the OBD-II standard, the typical vehicle only supports 20 - 40 sensors and is limited to emissions powertrain.

This situation is not unique to Ford - the majority of OBD-II PIDs are non-standard.

supporting the experimental conclusion[1] that there is a huge variance in the supported PIDs throughout the different cars. This means that the data that can be used in the project is only the data that is common in all cars. The most common PIDs are very easily to gather with ELM327 alone and don't require the time for learning how to use an API.

The PIDs used by the App are represented in table 2.1.

Table 2.1: Required Mode 1 OBD PIDs [2]

PID	Description	Unit	Mandatory
0x00	Get supported PIDs from 0x01 to 0x1F		Yes
0x04	% calculated engine load	%	Yes
0x05	Engine coolant temperature	$°C$	Yes
0x0C	Engine RPM	rpm	Yes
0x0D	Vehicle speed	km/h	Yes
0x11	Throttle position	%	Yes[2]
0x20	Get supported PIDs from 0x21 to 0x3F		Yes
0x40	Get supported PIDs from 0x41 to 0x5F		Yes
0x45	Relative throttle position	%	No
0x51	Fuel type	(see table 2.2)	No
0x5E	Engine fuel rate	L/h	No
0x60	Get supported PIDs from 0x61 to 0x7F		Yes
0x61	Driver's demand engine percent torque	%	No
0x62	Actual engine percent torque	%	No
0x63	Engine reference torque	$N \cdot m$	No

The fuel type PID is an enumerated type (see table 2.2).

The non-mandatory PIDs are not used in the model because they are less likely to be available in all vehicles, which makes their use unpractical. They are recorded and saved for the purpose of data the gathering system's expandability only, meaning they are kept so they can later be used in future work in case a work-around to this is found.

2.1.2 Inertial Sensors

Android's *SensorEvent API* [1] was used for the inertial sensors measurements. It defines the axis as represented in figure 2.1.

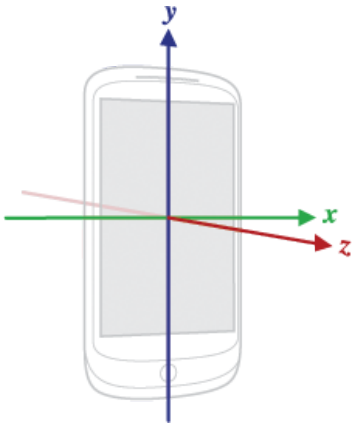
[1] In the 3 cars that I tested this in, the supported PIDs sets were all different

[2] Not used because one of the cars used to gather data doesn't report this PID correctly

Table 2.2: Fuel type codes [2]

0	Not available
1	Gasoline
2	Methanol
3	Ethanol
4	Diesel
5	LPG
6	CNG
7	Propane
8	Electric
9	Bifuel running Gasoline
10	Bifuel running Methanol
11	Bifuel running Ethanol
12	Bifuel running LPG
13	Bifuel running CNG
14	Bifuel running Propane
15	Bifuel running Electricity
16	Bifuel running electric and combustion engine
17	Hybrid gasoline
18	Hybrid Ethanol
19	Hybrid Diesel
20	Hybrid Electric
21	Hybrid running electric and combustion engine
22	Hybrid Regenerative
23	Bifuel running diesel

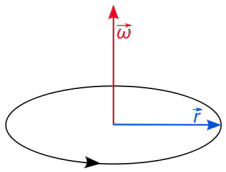
Figure 2.1: Axis as defined in SensorEvent API [1]



Two Sensors were used: the accelerometer and the gyroscope. The gyroscope is essential for noticing curves and rotation of the axes. The accelerometer was used instead of the linear acceleration because the gravity's constant influence in this sensor can be computed into the slope of the road and the low frequency part of it, the linear acceleration, can be used to detect when the driver is in turns, speeding up and slowing down.

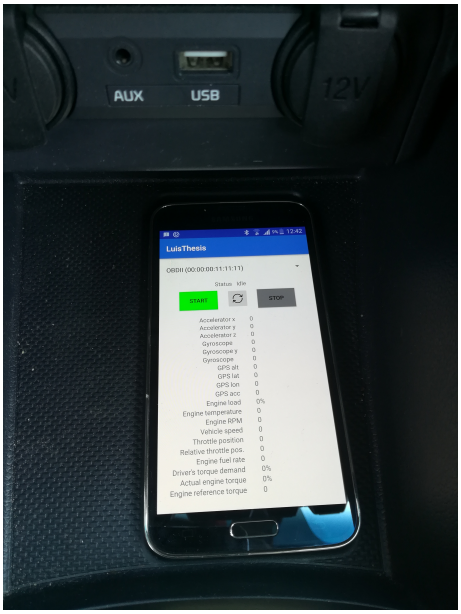
The x,y,z variables in the accelerometer are the accelerations in the directions of each axis in $m \cdot s^{-2}$. Those variables in the gyroscope represent the angular speed around each axis in $rad \cdot s^{-1}$ (angular speed vector has the same direction and sense as the axis'; see figure 2.2).

Figure 2.2: Angular Velocity



To reduce the noise and ease the data treatment, the phone should be placed static in an horizontal position (screen facing upwards) and the top of the phone facing the front of the car, like in figure 2.3.

Figure 2.3: Recommended phone placement



2.1.3 Location

Android's *FusedLocationProviderClient* [13] was used to poll the device's location because of its superior accuracy when compared to using only the GPS. This is due to the fact that *FusedLocationProviderClient* combines the GPS results with the GSM or Wi-Fi (whenever they are available) to get the best performance/accuracy and battery consumption balance, according to the programmer's demands. In the DC App, accuracy was prioritised.

The Location is used in this App so it is possible to manually validate some results by, for example, looking at a map and checking if the driver was indeed in a turn or somewhere where there may have been a dangerous situation (like driving too fast in a road with too many turns).

There was also an attempt to use a web-service to acquire the speed limit of a road by its coordinates but it wasn't possible because Google's API for speed limits is the only one that has reliable values and is not available for free[3]. Some e-mails were exchanged with Google and I was even contacted on the phone by an employee but my request for an exception on the grounds of educational use was rejected.

2.2 CRISP-DM

As mentioned in section 1.4, this project follows the CRISP-DM (Cross-industry standard process for data mining) methodology because it is a well described architecture for tackling data mining problems and fits perfectly with the system's needs. The overall work cycle is represented in figure 2.4.

Figure 2.4: CRISP-DM Process Diagram

(source: Wikipedia)

CRISP-DM was formally published in 2000 [14] by Shearer C.. It defines 6 major phases. Such phases and subtasks are represented in table 2.3 [15].

[3]The fees start at $50k

Table 2.3: CRISP-DM's phases and subtasks

Business Understanding	**Data Understanding**	**Data Preparation**	**Modelling**	**Evaluation**	**Deployment**
Determine Business Objectives	Collect Initial Data	Data cleaning	Select Modelling Technique	Evaluate Results	Plan Deployment
Assess Situation	Describe Data	Data Integration	Generate Test Design	Review Process	Plan Monitoring & Maintenance
Determine Data Mining Goals	Explore Data	Data Transformation	Build Model	Determine Next Steps	Produce Final Report
Produce Project Plan	Verify Data Quality	Data reduction	Assess Model		Review Project
		Data Reduction			

The implementation of the CRISP-DM in this project is detailed in section 3.4.

2.3 Data Preparation

Datasets are obtained by measuring real-life systems, which are extremely prone to noise. Besides that, the data may be in different scales and may also have offsets or be scattered in different amounts among the classes (asymmetrical datasets). Therefore, preparing the dataset is one of the most important parts of a Machine Learning project because the irregularities in the data cripple the models ability to learn and produce accurate results. Some algorithms are more and some are less sensitive to said irregularities in the data but all benefit from good, clean data. Data Preparation is also one of the main phases of the CRISP-DM methodology (see section 2.2).

The advantages of preparing the data are the following:

- information content will be better exposed
- errors in the predictions will be lower or, at most, the same
- acquaints the data miner with the data and, by doing so, increases the miners performance in producing better models, faster
- prevents *GIGO (Garbage In, Garbage Out)* which is a common concept in data science and also very self explanatory: if the data used to train the model isn't good, the predictions won't be either

- clean the real-world obtained data

According to professors João Mendes Moreira and José Luís Borges' subject *Knowledge Extraction and Machine Learning*'s lesson on *Data Preparation* at my faculty [15], the five major tasks/steps in data preparation are the ones represented on table 2.4.

Table 2.4: Data Preparation Steps

Task	Description
Data cleaning	Make the data more uniform
Data integration	Merge the data from the several sources
Data transformation	Normalise and aggregate the data
Data reduction	Reduce the volume of the data in such a way that it produces similar analytic results
Data discretisation[4]	Divide the range of a continuous attribute into intervals

2.4 Data Mining and Analysis

Data mining is the process of inferring new data from big entropy-prone data by analysing it from different angles and summarising it into data that can be useful in some context.

Data mining algorithms and methods can be separated into the following categories [16]:

- Anomaly detection
- Association rule learning
- Classification
- Regression
- Clustering
- Dimensionality reduction
- Evaluation
- Summarisation

In the context of this project, the relevant categories are *classification* and *clustering*.

2.4.1 Classification

Classification is a data mining function that assigns items in a collection to target categories or classes. The goal of classification is to accurately predict the target class for each case in the

[4] also known as *binning*

data. For example, a classification model could be used to identify a DS's risk as low, medium, or high [17].

A loss function is defined that represents how close the classifier is to the truth. In such functions, the closer the value is to zero, then better.

A classification task begins with a data set in which the class assignments are known. The model is then trained with that data in such a way that the loss function is minimised.

The classification method relevant in this project is Neural Network (NN), more specifically, Fully Connected Convolutional Neural Networks (FC-CNN).

FC-CNNs, from here on out simply called *CNN*s, are feed-forward neural networks which have a set of neurons connected to each other. Neurons are organised in layers and each neuron in a layer connects (sends its output) to all neurons in the next layer (hence the *Fully Connected* classifier). A layer only sends its output to the next layer and there can't be loops in the directional graph that is the Neural Network.

Each neuron i has a set of numbers (one for each input x_j it takes), W_i(array containing all these numbers for the neuron i), by which it multiplies the k inputs and then sums the products along with another number, b_i. This sum is then passed through an activation function, σ to return the output of that neuron, y_i (see equation (2.1)).

$$y_i = \sigma\left(\sum_{j=0}^{k-1}(x_j \cdot W_{i,j}) + b_i\right) = \sigma\left(\begin{bmatrix} W_{i,0} & W_{i,1} & \cdots & W_{i,k-1} \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{k-1} \end{bmatrix} + b_i\right) = \sigma\left(W_i \cdot x + b_i\right) \tag{2.1}$$

This process can be grouped by layers by using matrix multiplication. Let W be the n by k matrix in which each row is the W_i, having i as the neuron's index (see equation (2.2)), let b be the column array of size n (number of neurons in the layer) that contains all the biases from each neuron of the layer (see equation (2.3)) and let x be the column array of size k contain all layer's inputs (the previous layer's outputs - see equation (2.4)). Then let y be the column array of size n containing all the outputs of each neuron in the layer. y is then expressed in equation (2.5).

$$W_{|n\times k} = \begin{bmatrix} W_0^T & W_1^T & \cdots & W_{n-1}^T \end{bmatrix}^T \tag{2.2}$$

$$b_{|n\times 1} = \begin{bmatrix} b_0 & b_1 & \cdots & b_{n-1} \end{bmatrix}^T \tag{2.3}$$

$$x_{|k\times 1} = \begin{bmatrix} x_0 & x_1 & \cdots & x_{k-1} \end{bmatrix}^T \tag{2.4}$$

$$y_{|n\times 1} = \begin{bmatrix} y_0 & y_1 & \cdots & y_{n-1} \end{bmatrix}^T = \sigma\left(W_{|n\times k} \cdot x_{|k\times 1} + b_{|n\times 1}\right) \tag{2.5}$$

The layers' properties, the values that define a layer, are their W matrices, b vectors and activation functions.

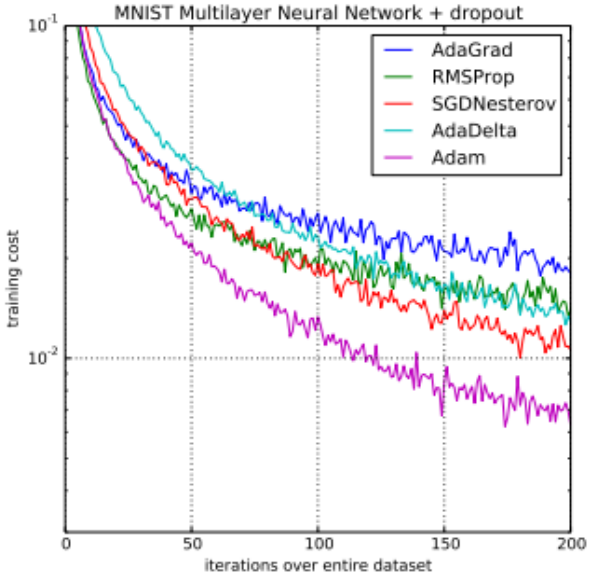
2.4.1.1 Training

Training is the process of tuning the weights and biases of each layer in such a way that the loss function is reduced. There are several methods for this. Some of the most common are the Genetic Algorithm, Gradient Descent and Adam. Adam [18] combines (and improves) the advantages of several other optimisation methods:

- Adaptive Gradient Algorithm (AdaGrad) - maintaining a per-parameter learning rate which improves performance on problems with sparse gradients
- Root Mean Square Propagation (RMSProp) - also maintains per-parameter learning rates but these are adapted based on the average of recent magnitudes of the gradients for the weight, meaning it has an improved performance with noisy training data.

The comparison of the loss function's evolution when training a model with different optimisers is represented in figure 2.5.

Figure 2.5: Optimisers performance comparison



The Adam optimiser was the chosen one for this project due its superior performance and speed.

2.4.1.2 Validation

Validation is the last step in the training process. It asserts the accuracy of the model by classifying data that wasn't used for training ("new data").

Over-fitting can be detected in the validation and happens when the model has a very high accuracy during training but a low accuracy in the validation data. This means that the model learned how to interpret the training data but didn't generalise it, and, therefore, can't later correctly classify new data.

2.4.2 Clustering

Clustering is used in this project to train the risk analysis model with unsupervised data by aggregating it into 3 clusters. Each risk class (high, medium and low risks) would be represented by the data in one of the 3 clusters.

Clustering is among the most basic human activities. Since we were children, we've always had the ability to classify objects and abstract ideas. It's what makes us be able to answer IQ-test questions like the one in figure 2.6 (taken from https://brilliant.org/problems/what-the-next-pattern-1/).

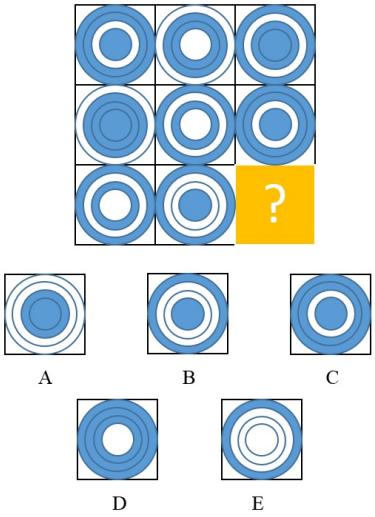


Figure 2.6: Data structure

Clustering is, then, the ability to group data without any previous knowledge over it.

There are several clustering algorithms. Two of them are the Bag-of-Features and the K-means algorithms.

Bag-of-features is an algorithm that is usually used for image classification but can be adapted to this problem by detecting mid-level features (risky behaviours) in the data and then using these to cluster the data over the risk level.

The K-means is another clustering algorithm but, unlike the bag-of-features algorithm, isn't guided in any way by the user. While the bag-of-features algorithm works with the provided features, which can be chosen according to their relevance in the meaning of the intended clusters (in this case, choosing risky events as features for the clustering of data over clusters that represent the driving risk) and assumes that all of them are relevant in the clustering process, K-means takes all the data and tries to find centroids in it. K-means could cluster the raw data into clusters that

represent, for example, whether the driver is going up or down a ramp or in a plane instead of the driving risk. K-means finds a pattern in the data which may or may not be the pattern the user is looking for. Because the driving risk is such a subjective and high-level feature, trying to cluster the raw data with K-means would be impractical.

2.5 Related work

This section is intended to be a brief summary of related work in the detection of driving risk and driving features.

1. Rui Araújo; Ângela Igreja; Ricardo de Castro; Rui Esteves Araújo, Driving coach: A smartphone application to evaluate driving efficient patterns", Intelligent Vehicles Symposium (IV), 2012 IEEE

   Link: http://ieeexplore.ieee.org/abstract/document/6232304/

   This paper was written by a former university professor of mine and some fellow students of my course and presents an application for smart phone created by them to help the users reduce the fuel consumption in their cars by helping the user adopt a better driving style.

   It uses fuzzy logic to implement the heuristics when analysing the sensors' data. This is a basic approach because the algorithm doesn't learn anything from the data. It just analyses it in order to give a qualitative evaluation of the driver's driving style based on heuristics.

2. Johannes Paefgen; Flavius Kehr; Yudan Zhai; Florian Michahelles, "Driving behavior analysis with smartphones: insights from a controlled field study", MUM '12 Proceedings of the 11th International Conference on Mobile and Ubiquitous Multimedia Article No. 36

   Link: http://dl.acm.org/citation.cfm?id=2406412

   This paper describes the creation of a smart phone application which is rather similar to mine except that this one disregards the plethora of measurements that can be taken from the car's sensors network via OBD. They had to tackle the problem of the noise in the sensors and they did so by combining several measurements before using their values. This approach will also be used in this project. The road condition is one of the things which is disregarded in most state-of-the-art solutions.

   This paper helped me think about a way to efficiently remove the sensors' noise.

3. T. Imkamon ; P. Saensom ; P. Tangamchit ; P. Pongpaibool, "Detection of hazardous driving behavior using fuzzy logic", Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, 2008. ECTI-CON 2008. 5th International Conference

   Link: http://ieeexplore.ieee.org/abstract/document/4600519/

   This paper's solution uses, like in the first one's, fuzzy logic to evaluate the sensors' stimuli and produce a qualitative evaluation of the driving style and one of its sensors is, like in the

first and second papers, a 3-axis accelerometer. Unlike the first and second papers, this uses the data from the engine and computer vision (a camera) as well. The data was labelled by questioning the passengers about each trip.

The difference from this project to mine is that the classification is done per trip instead of real-time.

4. Germaine L. Odenheimer; Marie Beaudet; Alan M. Jette; Marilyn S. Albert; Laura Grande; Kenneth L. Minaker, "Performance-Based Driving Evaluation of the Elderly Driver: Safety, Reliability, and Validity", Journal of Gerontology, Volume 49, Issue 4, 1 July 1994, Pages M153–M159

   Link: https://academic.oup.com/geronj/article-abstract/49/4/M153/565904

   This is the most interesting paper I read for the preparation of this dissertation albeit not as relevant to it as the other ones. There was a significantly bigger and more accurate dataset than in the others (30 different drivers, 2 independent evaluators seating in the back seat during each test drive and the drivers were subjected to cognitive tests). This study used human insight to correlate the age and cognitive functions with several driving behaviours, rather than a more technological approach like the one intended in this dissertation.

   This paper made me think about how I should do the risk classification. To present a quantitative value for the risk would require a lot of data, extensive statistical analysis over it and experts to validate the results. Besides, this data had to come from a big and reliable source like an insurance company (who holds a lot of data about driving accidents and the conditions that lead to said accidents) because the amount of data and accuracy needed to do a quantitative evaluation of the risk would be massive. This is why a qualitative classification of the risk was chosen: high, medium and low risk.

5. Tim Horberry; Janet Anderson; Michael A.Regan; Thomas J.Triggs; John Brown, "Driver distraction: The effects of concurrent in-vehicle tasks, road environment complexity and age on driving performance", Accident Analysis & Prevention, Volume 38, Issue 1, January 2006, Pages 185-191

   Link: http://www.sciencedirect.com/science/article/pii/S0001457505001521

   Unlike any one of the previous papers, there were no driving tests done but instead driving simulations to gather the data. It measures the performance of the driver in different driving conditions, operating or not the car's entertainment system and talking or not on the phone hands-free. It disregarded the discrepancies in drivers' cognitive functions, reaction times and other aspects that may affect a user's ability to drive under such circumstances. This is not necessarily a bad thing. It just means that this study focused solely on correlating the driving conditions and distractions with the performance of a specific driver, which in theory, should be correlatable with the performance of other drivers based on their characteristics.

6. Chalermpol Saiprasert; Thunyasit Pholprasit; Suttipong Thajchayapong, “Detection of Driving Events using Sensory Data on Smartphone”, International Journal of Intelligent Transportation Systems Research, January 2017, Volume 15, Issue 1, pages 17–28

   Link: http://doi.org/10.1007/s13177-015-0116-5

   This paper is about three algorithms for detecting risky behaviours in the driving style. Unlike in this project, this paper doesn’t try to get an abstract classification of the overall risk but rather the detection of risky situations. This is a work that I had to do to detect the mid-level features in the driving style in this project. The mid-level features are very important for the approach suggested in chapter 3 and for clustering the data using the bag-of-features clustering algorithm.

   The OBD data wasn’t used in this project, only the smartphone’s sensor data.

Chapter 3

Proposed System

3.1 Introduction

The whole system, its design and implementation will be detailed in this chapter.

As described in section 2.1, an Android Application was designed and implemented in order to obtain the data because there were no available datasets online due to this project's need for the combination of inertial sensors and OBD data. For this reason, the App was primarily designed to only gather data and send it to the back-end server. It can, however, be extended to include real-time analysis of the data after a proper model is defined, trained and uploaded into it (deployment).

The back-end server's purpose is to receive and keep all the data safe and centralised.

The designed model and its implementation of CRISP-DM is also discussed in this chapter.

3.2 The Data Gathering Application

As mentioned in 2.1, an Android application was developed to gather the data. This section will go into details about the App's functionality.

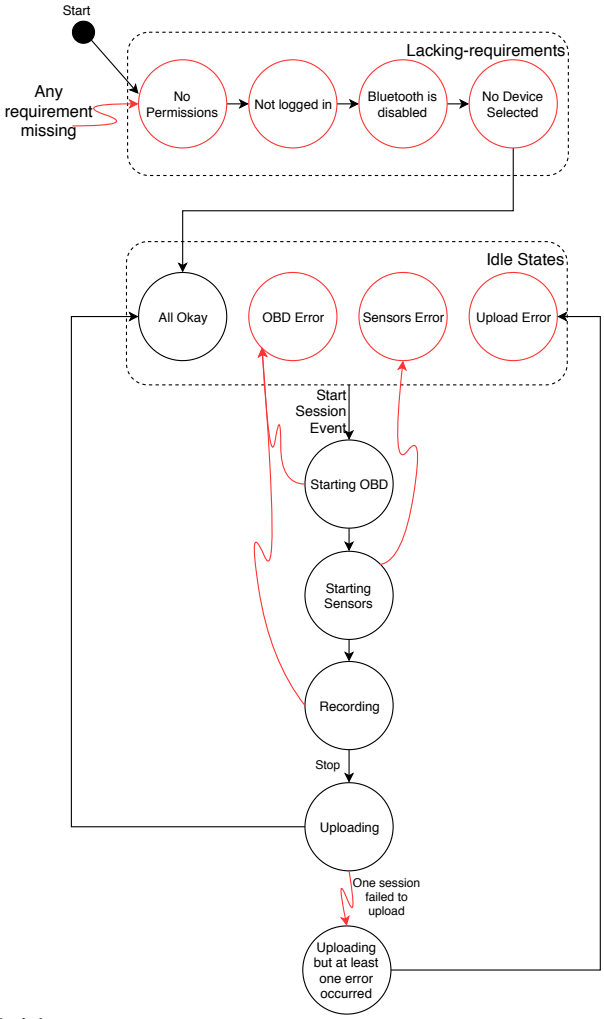
3.2.1 The flow - Service

This application follows a Finite State Machine (FSM) to control its state and states transitions. Such FSM is represented (simplified) in the figure 3.1. There are also events defined which trigger transitions if and only if they are legal in the current state.

Upon launching, the application checks if all of its requirements are met. Such requirements are:

1. the runtime permissions (coarse and fine location)
2. the login (explained in section 3.3)
3. the Bluetooth's state (which must be enabled)
4. whether or not a Bluetooth device was selected

Figure 3.1: Application flow



Subtitle:
-Black circles - normal states
-Red circles - error states
-Black straight lines - normal transitions
-Red swirly lines - transitions provoked by errors

Each requirement missing has its own error state. At any time of the execution, if an event that symbolises a lack of one of the requirements is fired, the FSM will jump into one of these states, according to whichever requirement is missing first (in the order in which they were listed above). These lacking-requirements states always cascade between themselves to guarantee that the Application will never go into an idle state with a requirement still missing (the code to check if a certain requirement is met is run upon entering the state and, if it is, the FSM jumps to the next lacking-requirement state or the Okay idle state). *Lacking permissions*, *user not logged in* and *Bluetooth disabled* result in pop-ups in the Activity to request the user to grant permissions, login and activate the Bluetooth, respectively.

If all requirements are met, the App will go into the Idle state, only then allowing the user to start recording a session.

Upon starting a session, the App attempts to connect to the selected Bluetooth device (referred to as *Bluetooth adapter* in 2.1). If it fails, the OBD error event will trigger resulting in the App jumping into an idle state that represents an said error. If it succeeds, it will poll the supported PIDs and check if all the mandatory PIDs (as defined in table 2.1) are supported (if they are not all supported, the same OBD error event as before is triggered). After this, the OBD is ready to start serving data.

Then, the accelerometer, gyroscope and location will be activated. If any of these fail to activate, the App's FSM will go into an idle state that represents a Sensor error and the OBD will be disconnected. If not, the application starts receiving periodic accelerometer, gyroscope, location and OBD data, storing them in a local SQLite database.

If, at any time during the recording period, there is an error in the sensors or in the OBD, the application will stop and go into the corresponding idle error state.

When the user requests the App to stop recording or when the App is idle and the user requests a manual upload, the application starts sending all the unsent and valid [1] sessions to the back-end server. Each successfully sent session is marked in the local database as such so it won't be sent again. After all sessions are sent or failed, the App will go to an idle state, either the Okay idle state if all sessions were sent successfully or an idle error state that represents an upload error if any of the sessions failed to do so.

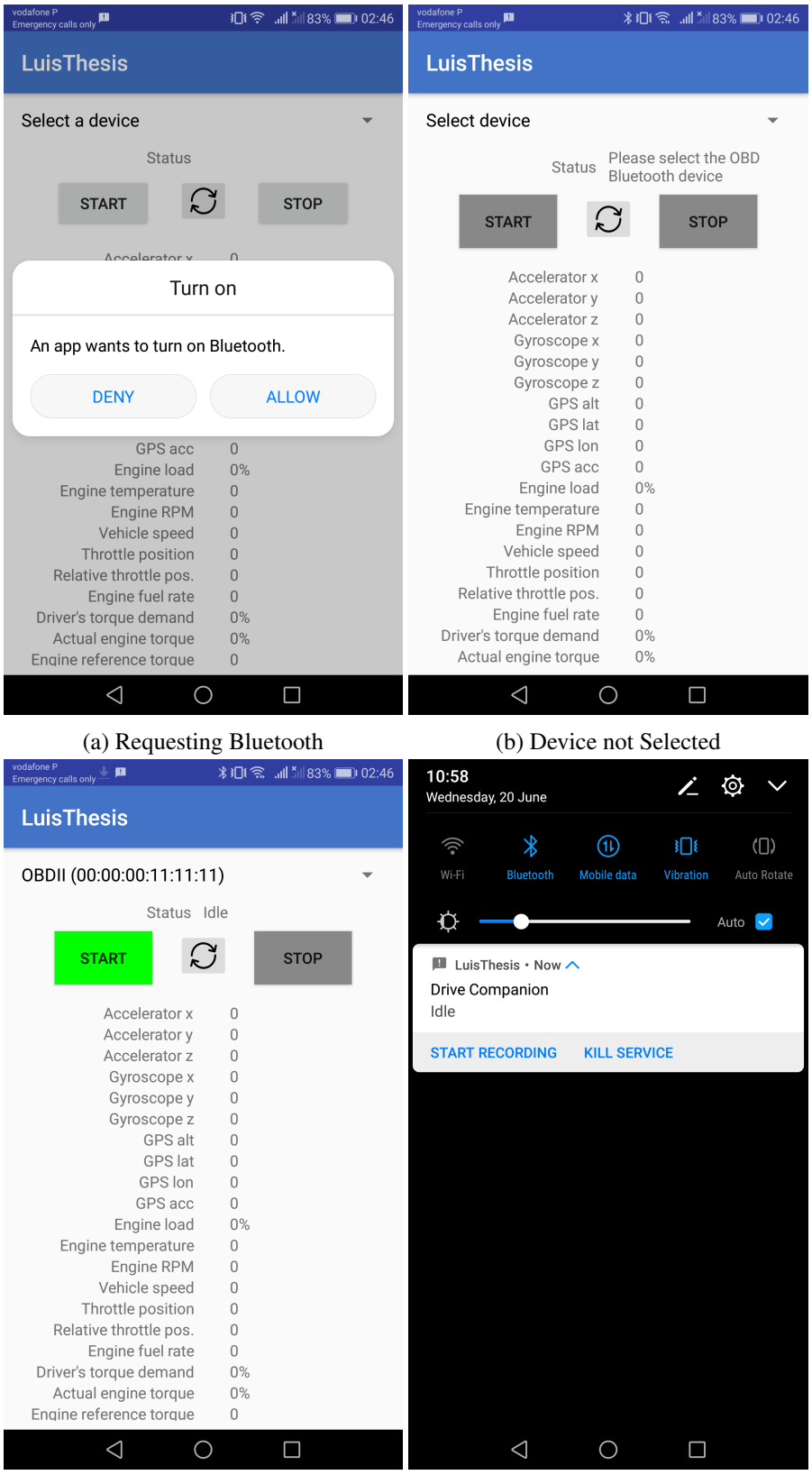
3.2.2 The User Interface

The user interface is simplistic and intuitive (figure 3.2). It is hollow, meaning that there is no business logic in it and it's just a puppet of the service.

The main components are the status bar, the start and stop buttons which activate, deactivate and change colours according to the service's FSM state, the selector for picking a paired Bluetooth Device and a *scrollview* with all the real-time data being shown and it's being recorded (see figure 3.2c).

There is also a notification that can be used to control the service and to check the service's state without opening the Activity (see figure 3.2d). That notification contains the same status as the status bar in the Activity and may be used to kill the service when the application is not running (recording or uploading).

[1] A valid session is one that wasn't forcefully interrupted and that contains data of all types (accelerometer, gyroscope, location and OBD)



(a) Requesting Bluetooth

(b) Device not Selected

(c) App ready

(d) The notification

Figure 3.2: User Interface

3.3 The Back-end

The back-end is also rather simple. It's a PHP web application that receives POST requests containing the data, checks for the authentication token and, if everything is okay, adds the payload to the database.

The communication is RESTful [19]. This architectural model for the communication is adequate because it's a client-server application, it's stateless (each package includes the authentication and no cookies or sessions are used) and uses a uniform JSON interface, as represented in figure 3.3.

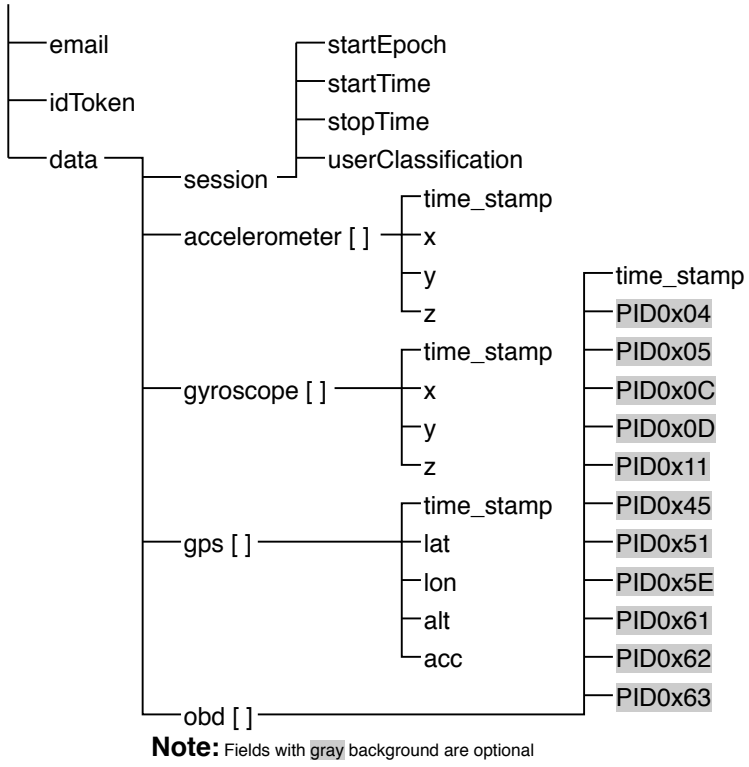


Figure 3.3: Data structure

The authentication is done using Google's OAuth2.0 API [20]. The user is prompted to login in the Activity, the API responds with a token-id that is sent along with the payload to the server. It is later used to authenticate him in the server by sending an API request to Google querying if the token-id is valid and the smartphone that is trying to send data is indeed the one who logged in.

The relational database engine used was PostgreSQL because it is extremely versatile, robust, fast and open-source. The DB script to create the schema is in appendix A.

3.4 CRISP-DM implementation

As mentioned in section 1.4, this project uses the CRISP-DM methodology which is briefly introduced in 2.2. It's important to note that only one cycle was done (there wasn't a restructure of the code following the evaluation phase) because the data was only available less than one and a half week before the delivery date due to the delay in receiving the OBD devices that were ordered in the internet and so there was not time to do more iterations nor to do a very deep analysis of the data.

The CRISP-DM implementation in this project is as follows (see table 2.3 for the enumeration of the phases and subtasks):

3.4.1 Business Understanding

In this phase, the problem should be analysed and the goals should be defined. The subtasks are:

1. **Determine Business Objectives**: The objective of the project is to mine driving data for information, specifically the driving risk;
2. **Assess the Situation**: There is neither a dataset available nor a pre-trained model nor any standard model for analysing driving data;
3. **Determine Data Mining Goals**: To be able to at least recognise some mid-level features (as defined in section 1.2) in the data;
4. **Produce Project Plan**: Find the best way to prepare the data, define time-frames for data to be analysed in, define a model for detecting MLF;

3.4.2 Data Understanding

In this phase, the miner should get acquainted with the data and know precisely what each field represents. The subtasks are:

1. **Collect Initial Data**: An initial sample was collected to get a sense of the values range, their progression over time and to test the data gathering App;
2. **Describe Data**: Already described in section 2.1;
3. **Explore Data**: The data was explored by looking at it in real-time. This was made by looking at the phone while performing actions to test each field (tilting the phone, pressing the car's throttle pedal, comparing the OBD values with the car's dashboard gauges information, ...);
4. **Verify Data Quality**[2]: The accelerometer proximately shows the standard value of gravity in the direction of the axis perpendicular to the floor while being held so I've considered it

[2]The verification was done in my own car and not in the candidates' cars; An assumption was made that the data in the candidates' cars is being correctly reported by the OBD, with exception to the throttle position in one of them.

to have a small, neglectable, error. The gyroscope was tested in the vinyl record player's spinner and it also reported accurate values (error of $\pm 0.45\%$, taking the spinner as a ground-truth by considering that it spins at the designated speed, which it should because the vinyl records play at the correct speed in the same spinner). Only mandatory PIDs of the OBD data were verified: % calculated engine load couldn't be verified because there was no way of actually measuring the engine's torque; Engine coolant temperature, Engine RPM and Vehicle speed were verified by comparing them with the values displayed on the dashboard, which also suggested a good accuracy with exception to the speed value which was slightly ($\sim 3 km/h$) lower on the OBD read than in the dashboard gauge.

3.4.3 Data Preparation

This is one of the most important parts in the CRISP-DM because it prevents the GIGO problem. The subtasks were implemented as follows.

3.4.3.1 Data Cleaning

Data rows with empty fields were discarded. This wasn't all that common because the data is already sanitised by the App before being sent to the back-end server.

3.4.3.2 Data Integration

The data is merged from the local databases into the server database either manually or at the end of each trip, which means that the data is already collected in one place, the server. This was described in section 3.3.

The integration of the data from each session (joining the database tables) was done with SQL.

3.4.3.3 Data Reduction/Selection

Only the most common PIDs were selected and made mandatory (see table 2.1). An exception was made and a mandatory PID, the Throttle Position, excluded from the data analysis because one of the candidate's car didn't report its value correctly. Because this candidate, the dissertation's supervisor from Fraunhofer, was one of the biggest contributors in the data gathering process, his data couldn't be excluded or else there would be even less data available.

The selected data is the following:

- Accelerometer's vector
- Gyroscope's vector
- % calculated engine load
- Engine coolant temperature
- Engine RPM

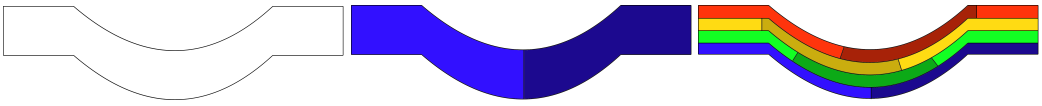
- Vehicle speed

The *% calculated engine load* field was selected because it can be associated with how long the user has been driving for. Besides, if this OBD PID turns out not to be relevant, the Neural Network will eventually learn it during the training and set its weights to zero in order to discard it.

3.4.3.4 Data Discretisation

Because the different data sources are independent from each other and have different yielding frequencies, the data needs to be aggregated into time intervals. This process is also called binning, more specifically equal-width binning because the data was divided in intervals of the same width, 5 seconds, regardless of the number of samples inside that same interval. This is preferable over the equal-depth binning because the whole objective of this subtask is to normalise the frequency of all data sources.

Let us now consider the practical case in which the driver is in a straight road segment, enters a turn and then another straight segment, like represented in 3.4a. Now let the time-frame be such that it ends in the middle of the turn as in figure 3.4b. Those intervals' classifications would be ambiguous because they cut the turn in half. This can be solved by training the model using overlapped time-frames like in figure 3.4c.



(a) The turn (b) Time-frames (c) Overlapped time-frames

Figure 3.4: Time-frames example

Considering one of the data-frames as the reference, the other three are offset by 1.25, 2.5 and 3.75 seconds but all are 5 seconds long.

In each interval, each field will produce two: the arithmetic average and the standard deviation. The reason for the average is so that the time-frame has the middle value of the measurements in it. The standard deviation is very important to detect variations in the data. If this wasn't done, information of event comprised completely inside a time-frame would be lost. For example, if a user did a turn to the right followed by a turn to the left and this whole episode was inside a single interval, this wouldn't be noticed because the average of the inertial sensors would be close to zero due to the opposite actions.

This whole process of grouping by offset time-frames is done by a big SQL query that's presented in appendix B.

3.4.4 Modelling

In this phase, the model is defined and trained. The implementation will be detailed later in section 3.5. The subtasks are:

1. **Select Modelling Technique**: The Modelling Technique selected for this project is a Convolutional Neural Network (CNN). It is assumed that the data is input while collected in time-frames (arithmetic average and standard deviation);

2. **Generate Test Design**: The Tests made are using holdout data, which is data that is removed from the dataset, and validating the model with it after training;

3. **Build the Model**: Parameters and Model's Description are in subsection 3.5.1;

4. **Assess Model**: Model's technical review is also in the subsection 3.5.1.

3.4.5 Evaluation

In this phase, the model is evaluated, reviewed and improvements defined. The subtasks are:

1. **Evaluate Results**: Results in chapter 4;

2. **Review Process**: Reviewing issues in the process will be in chapter 4.1;

3. **Determine Next Steps**: The next steps will be defined in the Future Work section (5).

3.4.6 Deployment

This is the phase with the least amount of practical work done because most of it never got out of the planning phase. The model was never actually deployed due to the fact that, by the end of this dissertation, there was not a final version that detected the DS risk. It was just impossible to do everything in one week. However, the deployment phase was still planned and is described in the following subtasks:

1. **Plan Deployment**: The project wasn't deployed but the plan for the deployment was made, nonetheless. The planned tool for deploying the model is *Google Firebase's Machine Learning* which features some interesting things like the developer being able to upload a model into the platform and have the model be deployed in every phone updated without the need to update the App, dynamic definition of a TensorFlow Lite Model both in structure and in the values (weights, biases and other parameters), among others. The App should be deployed when a working version of the model is created;

2. **Plan Monitoring and Maintenance**: After being deployed, the model need to be monitored so we can see if there is a drop in performance. This is actually a feature of the API that was going to be used to deploy the model, *Google Firebase's Machine Learning*.

3. **Produce Final Report**: CRISP-DM fits so well into this project that even the creation of this very document was proposed by it. The final report must answer all the questions and provide the results of all the subtasks in the cycle iteration, which is precisely the purpose of section 3.4.

4. **Review Project**: Project's revision and final considerations will be made in sections 4.1 and 5.

3.5 The Model

Following the logic proposed in the project's objectives (section 1.2), there should be an attempt to create a model to detect Mid-Level Features (MLF) from Low-Level Features (LLF)[3] and another one for detecting High-Level Features (HLF) from a combination of MLF and LLF[4].

3.5.1 Building the MLF detection model

The MLFs that were initially planned to be implemented are represented in the table 3.1. But because there was no time to design, analyse and train a Model for each MLF in that table, a proof of concept was build for the first one in the table: turn detection.

The features and the raw inputs from which they will be calculated are in table 3.1

Table 3.1: Data Features and respective inputs

Feature	Inputs
Turn	Accelerometer; Gyroscope
Acceleration	Accelerometer; Gyroscope; PID 0x04 0x0C 0x0D 0x11;
Brake	Accelerometer; Gyroscope; PID 0x04 0x0C 0x0D 0x11;
Overtaking	Accelerometer; Gyroscope; PID 0x04 0x0C 0x0D 0x011;

This model takes only the accelerometer and gyroscope data as inputs because they are the only relevant data fields (which are available in the dataset[5]) for telling a turn apart from a straight road segment.

Assumptions:

1. all models were trained with a learning rate of 0.01 and 200 epochs so the accuracies were comparable;

2. the activation functions in the hidden layers are all ReLU

3. the activation function in the output layer is the Sigmoid function

4. weights are randomly initialized[6]

5. biases are initialized to zero

[3] objective 1.
[4] objective 2.(b)
[5] Steering angle would have been nice to have but it isn't a standard OBD PID but rather a manufacturer-specific PID
[6] The random seed is set to 7 in the beginning of the Python script so it becomes deterministic, though

3.5.1.1 Creating the dataset files

The files were obtained with the PostgreSQL *COPY* instruction to extract the results of the queries into *.csv* files.

Files:

- **dataset.csv** is the file containing the inputs and their respective expected outputs. This file's data will be used to train the model. The header is: avg(acc_x), stddev(acc_x), avg(acc_y), stddev(acc_y), avg(acc_z), stddev(acc_z), avg(gyr_x), stddev(gyr_x), avg(gyr_y), stddev(gyr_y), avg(gyr_z), stddev(gyr_z), turn, straight. Columns *turn* and *straight* are the expected outputs for each input, which means they are always either 1 or 0 to symbolise whether the input belongs to the class or not respectively;

- **validation.csv** is the file containing the validation data. Its structure is the same as the previous one but this has less data rows;

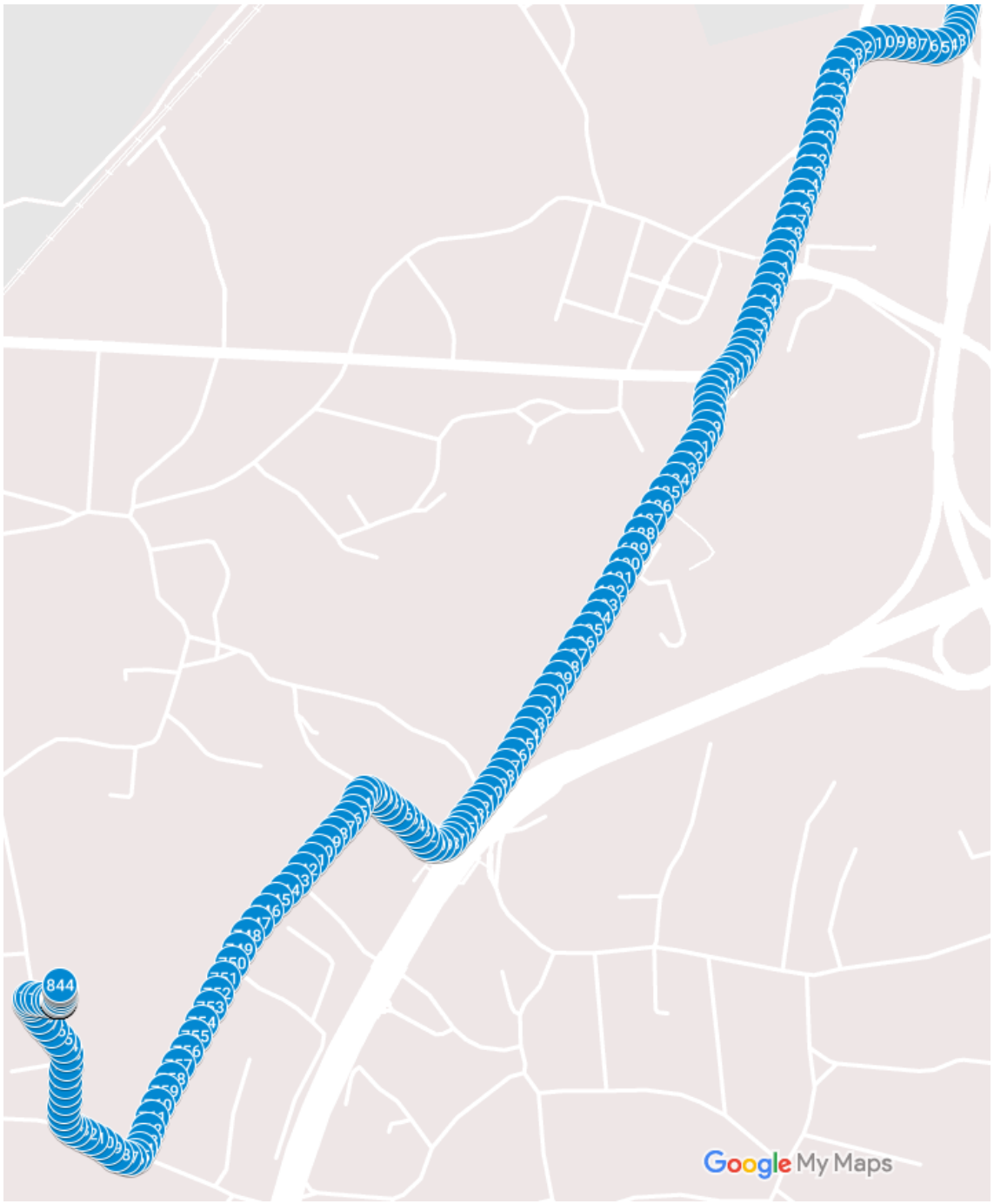
- **model.h5** is the file where the compiled and trained model will be saved so it can later be loaded again;

- **sess_xx_test** ("xx" is the decimal id number of the session) is the file where the data to be predicted is saved. The structure is avg(acc_x), stddev(acc_x), avg(acc_y), stddev(acc_y), avg(acc_z), stddev(acc_z), avg(gyr_x), stddev(gyr_x), avg(gyr_y), stddev(gyr_y), avg(gyr_z), stddev(gyr_z);

- **sess_xx_test_preds** ("xx" is the decimal id number of the session) is the file where the predictions of the model will be saved for every input in the previous file. The structure is just two columns with the prediction values for both *turn* and *straight* classes.

The dataset was created for training the turn-detection MLF CNN by manually looking at maps like the one on figure 3.5 and recording in a spreadsheet some location intervals in which each class happened. After having a list of turns and straight segments, the location pins were then correlated with the time-frames in which they occurred. Then the big query in the appendix B was run with the adequate WHERE clause to only show the data in the time-frames that were labelled. These values were then shuffled by rows and the last 20 cut and pasted into another file for validation. This resulted in 273 rows of data that were distributed like so: 169 rows of turns and 104 rows of straight road segments.

Figure 3.5: The map as it was used to analyse the sessions and label the data



3.5.1.2 Design and train the model

In order to define how many layers and neurons per layer should be used, a cycle of tests was performed for all of those configurations.

A very simple model of a single-layered (1 input layer, 1 hidden layer and 1 output layer) CNN was defined and trained. This was able to prove that if a model is too simple, it can't learn because after some epochs, it would forget the previous training. In this specific CNN the accuracy stayed at a low 65%, which is almost a random uniform distribution (50%). This allowed for a minimum complexity to be defined.

Now, in order to bind the problem between two complexity limits, finding an upper bound in the complexity would help the process of finding a more appropriate model for this data.

The more layers there are, the better the model can learn non-linear behaviours in the data. A model with only one layer would have a poor performance in learning non-linear behaviours. So, in order to get the upper bound mentioned in the previous paragraph, a model with three hidden layers started to be trained. After getting that upper bound, the model should have its complexity decreased for as long as there is overfitting in the training. To reduce the complexity, the number of neurons and layers can be decreased but always keeping the number of neurons descendent over the depth of the network to make the data be more and more dense ("dense" as in "meaning being closer to the output, albeit still abstract") the closer it is to the output layer.

The first attempt was a 10:8:7 CNN. Upon being trained, there was an improvement over the previous one-layered model with an accuracy of 70% using the validation data and a $1.2 \cdot 10^{-4}$ in the training loss function. This is a clear case of *overfitting* because the loss function's value was very low but the validation accuracy was only 70%. This means that the model learned how to recognise the training data itself instead of learning its meaning ("turns"/"straight road segments") so it performed poorly on new data.

There are many ways to reduce overfitting, one of them being to insert a dropout layer in between the convolutional layers. Dropout layers randomly force values to zero during the training (one of the layer's parameters is the probability of it forcing a zero). Figure 3.6 denotes the (exaggerated) effect of a dropout layer in a CNN. An important thing to keep in mind when using dropout layers in that the droupout-rate shouldn't be too high in the first layers because data lost in the beginning will be lost in the rest of the network as well, which means that greater dropout-rates in the beginning of the network have a greater impact than if they were closer to the end.

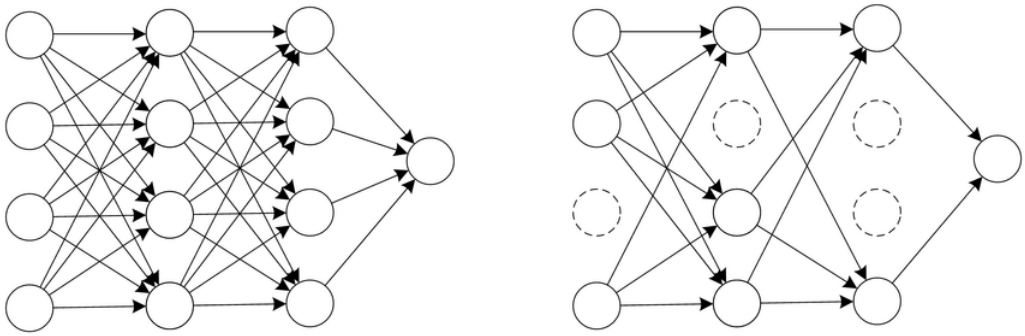
The number of neurons per layer was then decreased by 4, turning it into a 6:4:3 CNN. An accuracy of 75% in validation data and 96% in the training process was obtained. There was a reduction in the overfitting but it still happened: the validation accuracy was still low while the system was being really good at learning the test data. So now, to reduce the complexity, a layer was removed.

The new CNN was a 10:5. Upon training, an acceptable balance was reached: training accuracy was 98.72% and a validation accuracy of 82.5% (see figure 3.7). The validation data was still not resulting a very good accuracy but, upon plotting the results on a map, another reason became evident as to why the data wasn't generating more accurate results.

The analysis of this model's results is done in section 4.2.

Regarding the other MLFs in table 3.1, it would be much harder to get data to train them because they can't be recognised just by looking at a map like what was done to create the turns

Figure 3.6: Dropout effect on a CNN



(a) Standard Neural Network

(b) Network after Dropout

(source: https://goo.gl/p2RSyB)

and straight segments' dataset. They would require someone to actually go and drive a car on purpose just to recreate said MLFs. That wouldn't be feasible with one week to analyse the freshly collected data and write the rest of the thesis, which was on hold until the data was available. It wouldn't also be practical even if the OBD devices had arrived sooner. This is the problem of not having a dataset already prepared: one is completely bound to whatever data is available whenever it is available!

3.5.2 Building the HLF detection model

To train this model to detect risk, data should be clustered into three clusters, each representing a risk class: *low risk*, *medium risk* and *high risk*.

Mid-Level Features can be extracted from the data (Low-Level Features) and input into the HLF detection NN (instead of just inputting the raw data) to create a higher-level, less abstract input to the model. For example, a value in the range [0;1] can be added to each window of data representing the probability of it belonging to the "on a turn" or "sudden break" classes. This could theoretically improve the clustering algorithm's effectiveness by giving the dataset something that is more closely related to the driving risk than raw data. The problem with some completely unsupervised clustering algorithms like K-means is that they don't cluster the data with a specific target in mind. In the case of K-means, it clusters by finding patters, whatever they may be. As an extreme example, it could cluster by whether the used phone's manufacturer is *A* or *B* if such information could be statistically correlated with the measurements. This is where the mixed approach, where higher-level data is added to the dataset, comes into play. It improves the chance of the clustering algorithm finding a pattern that is relevant, in this case the driving risk classes.

With this said, the model that was planned to be used consists of a CNN per each MLF to pre-classify each time-frame before inputting it to the clustering algorithm and, later, to the HLF detection model.

Because the results were good in the proof of concept, I believe this would be possible to do as well if there was more time.

Figure 3.7: Jupyther Notebook screenshot of the 10:5 CNN training

```
inpu=np.genfromtxt(
    '/home/luisp/Desktop/teseData/features/dataset.csv',
    delimiter=',',
    skip_header=1
)
gd={'inputs': inpu[...,:-2], 'outputs_true': inpu[...,-2:]}
N_INPUTS=gd['inputs'].shape[1]
print('N_INPUTS='+str(N_INPUTS))
```

```
N_INPUTS=12
```

```
X=gd['inputs']
Y=gd['outputs_true']
model = Sequential()
model.add(Dense(10, input_dim=N_INPUTS, activation='relu'))
model.add(Dense(5, activation='relu'))
model.add(Dense(2, activation='sigmoid'))

model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

model.fit(X, Y, epochs=N_EPOCHS, batch_size=100)
```

```
273/273 [==============================] - 0s 21us/step - loss: 0.0835 - acc: 0.9853
Epoch 193/200
273/273 [==============================] - 0s 20us/step - loss: 0.0827 - acc: 0.9853
Epoch 194/200
273/273 [==============================] - 0s 26us/step - loss: 0.0819 - acc: 0.9872
Epoch 195/200
273/273 [==============================] - 0s 24us/step - loss: 0.0811 - acc: 0.9872
Epoch 196/200
273/273 [==============================] - 0s 24us/step - loss: 0.0804 - acc: 0.9872
Epoch 197/200
273/273 [==============================] - 0s 53us/step - loss: 0.0796 - acc: 0.9872
Epoch 198/200
273/273 [==============================] - 0s 26us/step - loss: 0.0789 - acc: 0.9872
Epoch 199/200
273/273 [==============================] - 0s 25us/step - loss: 0.0781 - acc: 0.9872
Epoch 200/200
273/273 [==============================] - 0s 22us/step - loss: 0.0774 - acc: 0.9872

<keras.callbacks.History at 0x7f75036319b0>
```

```
validate=np.genfromtxt(
    '/home/luisp/Desktop/teseData/features/validation.csv',
    delimiter=',',
    skip_header=1
)
scores = model.evaluate(validate[...,:-2], validate[...,-2:])
print("\n%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
```

```
20/20 [==============================] - 0s 47us/step

acc: 82.50%
```

Chapter 4

Results

4.1 The System's final review

The system's framework (App and back-end) is functional and works as expected. It can definitely still be improved and modularly scaled so it can be used to create new datasets for the later continuation of this project. The code is structured to improve the readability. There is some JavaDoc throughout the Android project and comments in the PHP back-end, easing the task of whoever is going to pick the project up and improve/scale it.

4.2 Results of the Mid-Level Features Detection proof of concept

After training the turns-detection CNN, the model is saved (see figure 4.1) and then used to predict the classes of a whole trip (see figure 4.2). The predictions are then saved in a csv file which, opened with *LibreOffice Calc*, can be easily mapped to the coordinates where each time-frame was recorded. This program is also used to normalise the data so that the sum of the predictions are all exactly 1 (this is the intended behaviour because the classes are mutually exclusive). The file can then be uploaded into *My Google Maps* (`https://www.google.com/mymaps`) to plot the trip and label it with the prediction value of being in a turn. Such map is represented (zoomed out) in figure 4.3. Most places are predicted correctly (example in figure 4.4) but some aren't, as one can see in figure 4.5 where a big turn wasn't classified as one and in figure 4.6 where two turns in the direction of north-east (car is going from south to north) are classified correctly but a turn in the direction of north-west isn't.

Figure 4.1: Jupyther Notebook saving the model

```
In [6]: model.save('/home/luisp/Desktop/teseData/features/model.h5')
```

It's noticeable that, even in figure 4.4, the model is biased to accepting turns to the right but not to the left. This is due to the turns dataset's unbalanced nature. One of the sessions with the

Figure 4.2: Jupyther Notebook creating a prediction

In [7]:
```
val2=np.genfromtxt(
    '/home/luisp/Desktop/teseData/sess_59_test.csv',
    delimiter=',',
    skip_header=1
)

#print(val2[...,:-1])

preds=model.predict(val2[...,:-1])

print(preds)
```

```
[[1.4246925e-08 1.0000000e+00]
 [1.4246925e-08 1.0000000e+00]
 [8.5232801e-05 9.9999523e-01]
 ...
 [9.9999392e-01 4.3917421e-06]
 [9.9999797e-01 1.1206184e-06]
 [9.9999797e-01 1.1206184e-06]]
```

In [11]:
```
np.savetxt(
    '/home/luisp/Desktop/teseData/sess_59_test_preds.csv',
    preds,
    delimiter=','
)
```

most turns used to gather the data was in Espinho, Portugal, which is a city with a block-like city planning (orthogonal streets). This is why there were a lot of turns in that session. The problem is that most of them were to the right so the model got biased to right turns.

Figure 4.3: Predictions map

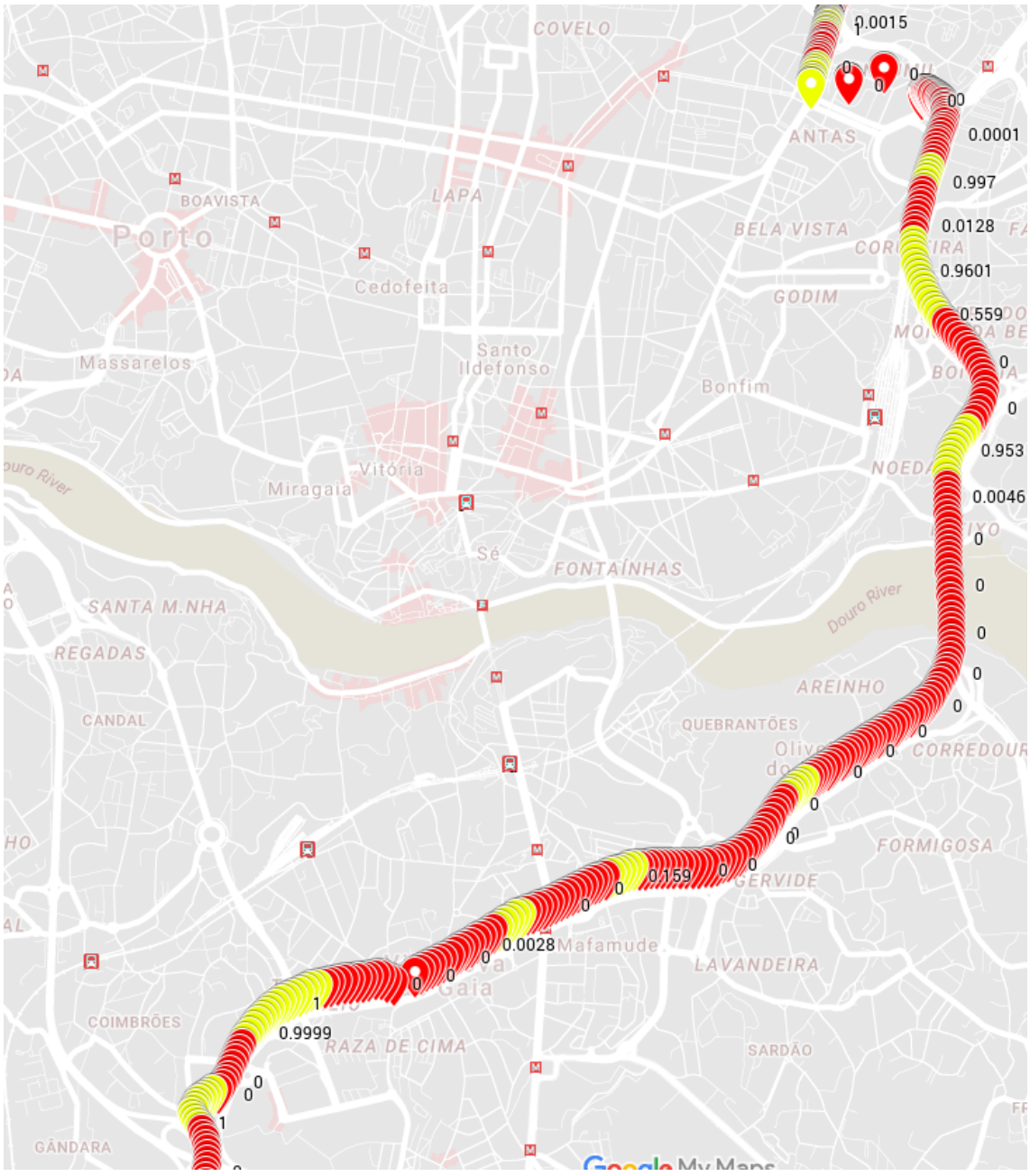


Figure 4.4: Correct predictions map excerpt example

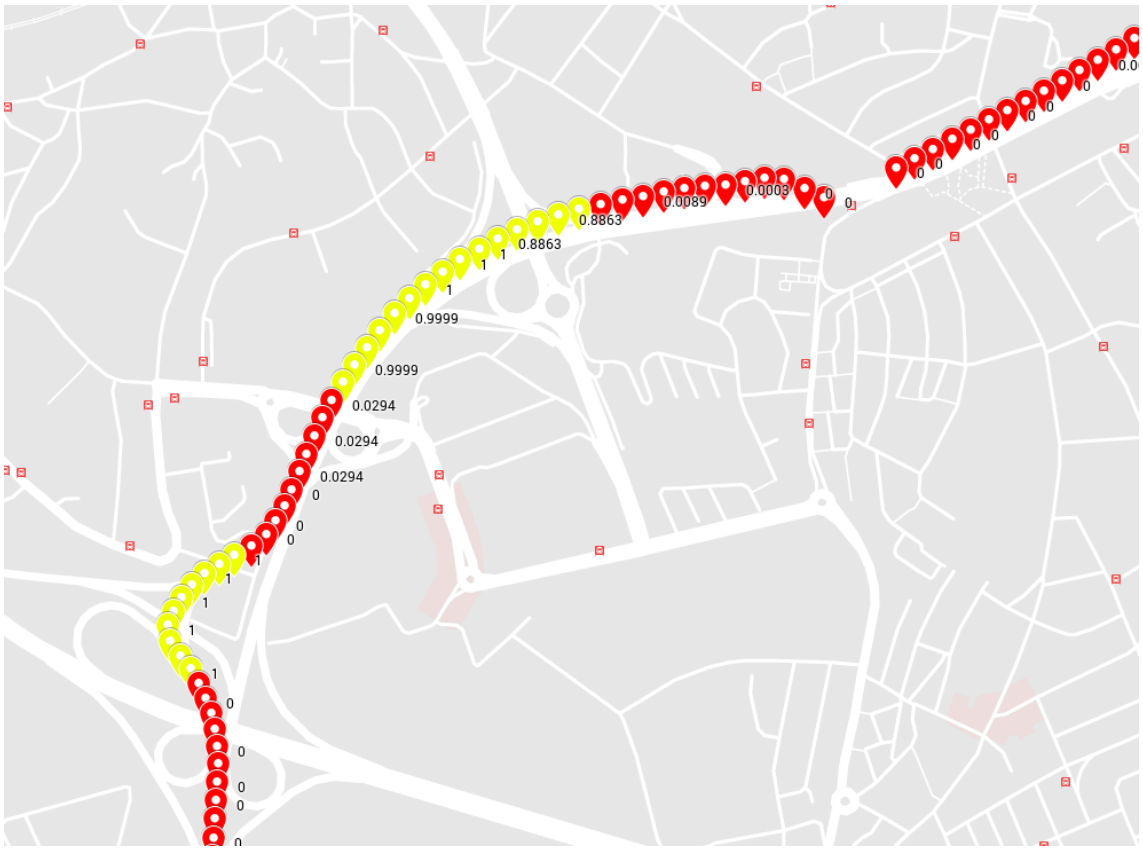


Figure 4.5: Wrong predictions map excerpt example no. 1

Figure 4.6: Wrong prediction map excerpt example no. 2

4.3 Answering the Research Questions

In this section, the questions asked in 1.3 will be answered.

4.3.1 Is it possible to collect, analyse and classify vehicle driving data in real-time using a smartphone?

The only problem while trying to classify data in real-time with a phone would be the deployment to it but, because there is an API for that already, as discussed in subsection 3.4.6 of the CRISP-DM implementation section, that problem is nullified. The only limitation is that the model must be compatible with the Lite version of Tensorflow. And because the models proposed in this dissertation are all CNNs, they are indeed compatible with the Lite version.

This way, because the detection of MLF is quite easily feasible, given a dataset with the other features labelled, it is possible to detect the MLF in real-time on the phone.

Nevertheless, the experience that was made proves that care must be taken when creating the dataset because the model behaves well even when trained with little data but suffers a lot with unbalanced datasets.

4.3.2 Is it possible to tell dangerous situations apart from normal driving style using such a system?

This question can't be answered objectively or backed by empirical evidence because the detection of HLF wasn't accomplished due to lack of time and data.

If I were to give an educated guess on the matter, though, I believe that it is, indeed, quite possible to tell dangerous situations apart from driving style with a system with this architecture because the models are very easily trained and are quite versatile (the model was trained with mostly tight turns to the right but still was able to detect wide turns to the right - see figure 4.4's wide turn and figure 4.6's north-most detected, yellow, turn).

If the MLF are so easily detectable, detecting HLF based on another CNN that receives raw data along with MLF data is very plausibly a possible endeavour.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

This work aimed to study and evaluate the viability of Machine Learning's use in the detection of driving risk. This couldn't be accomplished because there was no dataset available and the data that was recorded was very few and completely unlabelled. For this reason, the only solution that was trained was the turn detection because it is the only mid-level feature I thought of that could be labelled just by looking at the location plots and annotating the timestamp intervals. Other MLF would require recreating risky behaviours while driving a car and recording the data (several times), which is mostly neither legal nor practical.

The detected MLF (turns detection) which was used as proof of concept of the detection of MLFs using Neural Networks proves that it is possible to indeed detect MLFs from raw data, provided there is enough labelled data.

In regards of the project's quality, it could have been better if the OBD adapters had come sooner and there was a bigger dataset with more diversity in regards to driving risk conditions (most of the driving sessions are of safe driving styles).

One of the contributions of this thesis was the data gathering framework which can continue being used by some Fraunhofer's employees to gather more data so someone else can later pick up the project and train the models with more data. Another contribution of this thesis is the assertion of the possibility of detecting driving features from the chosen dataset using convolutional neural networks.

Overall, I feel that this work was too complex to begin with, that it had a very unstable and, frankly, unreachable goal (the driving risk) in this dissertation's time-frame because it is subjective and requires experts' opinions (for example insurance experts) for it to be validated. Also, I think the obtained results are what was expected with the kind of data and restrictions present in this project.

5.2 Future Work

There is a lot of future work that can be done in the scope of this project now that the data gathering part is much more advanced then it was in the beginning.

Some of it is:

- Gather labelled data on the other Mid-Level Features that were not yet explored in this project using the system that was developed during this dissertation;
- Adapt the model that was created for detecting turns to those other MLFs;
- Implement more sophisticated clustering algorithms that better take advantage of the MLFs;
- Train a the proposed HLF detection model with the clustered data;
- Try another non-CNN approach to the problem;
- Create a more objective, quantitative scale for the risk measurement;
- Gather data with more interference to make the model more robust to noise and dirtier data (for example, gather data with the phone in different positions);
- Implement the deployment module in the App. This is an easy task that can be easily accomplished but isn't useful until the others are completed. Especially the one to gather more data on and model the other MLFs

Appendix A

DB Schema Script

```
CREATE TABLE IF NOT EXISTS session (
  id BIGSERIAL PRIMARY KEY,
  userEmail VARCHAR(200) NOT NULL CHECK (userEmail~'^[a-z_\-.0-9]+@[a-z0-9]+(\.[a-z0-9]+)*\.[a-z]{2,}$'),
  startEpoch BIGINT NOT NULL CHECK (startEpoch>0),
  startTime BIGINT NOT NULL CHECK (startTime>0),
  stopTime BIGINT CHECK (stopTime>0),
  userclassification INTEGER CHECK(userclassification IS NULL OR (userclassification>=0 AND userclassification<=2)),
  UNIQUE (userEmail,startEpoch)
);

CREATE SEQUENCE IF NOT EXISTS data_id_seq;
SELECT setval('data_id_seq',1,false);

CREATE TABLE IF NOT EXISTS accelerometer_data (
  session_id BIGINT NOT NULL REFERENCES session(id) ON UPDATE CASCADE ON DELETE CASCADE,
  id BIGINT NOT NULL DEFAULT nextval('data_id_seq'),
  time_stamp BIGINT NOT NULL CHECK (time_stamp>0),
  x REAL NOT NULL,
  y REAL NOT NULL,
  z REAL NOT NULL,
  PRIMARY KEY (session_id,id)
);

CREATE TABLE IF NOT EXISTS gyroscope_data (
  session_id BIGINT NOT NULL REFERENCES session(id) ON UPDATE CASCADE ON DELETE CASCADE,
  id BIGINT NOT NULL DEFAULT nextval('data_id_seq'),
  time_stamp BIGINT NOT NULL CHECK (time_stamp>0),
  x REAL NOT NULL,
  y REAL NOT NULL,
  z REAL NOT NULL,
  PRIMARY KEY (session_id,id)
```

```
32 );
33
34 CREATE TABLE IF NOT EXISTS gps_data (
35   session_id BIGINT NOT NULL REFERENCES session(id) ON UPDATE CASCADE ON DELETE
       CASCADE,
36   id BIGINT NOT NULL DEFAULT nextval('data_id_seq'),
37   time_stamp BIGINT NOT NULL CHECK (time_stamp>0),
38   lat REAL NOT NULL,
39   lon REAL NOT NULL,
40   alt REAL NOT NULL,
41   acc REAL,
42   PRIMARY KEY (session_id,id)
43 );
44
45 CREATE TABLE IF NOT EXISTS obd_data (
46   session_id BIGINT NOT NULL REFERENCES session(id) ON UPDATE CASCADE ON DELETE
       CASCADE,
47   id BIGINT NOT NULL DEFAULT nextval('data_id_seq'),
48   time_stamp BIGINT NOT NULL CHECK (time_stamp>0),
49   PID0x04 REAL NOT NULL,
50   PID0x05 REAL NOT NULL,
51   PID0x0C REAL NOT NULL,
52   PID0x0D REAL NOT NULL,
53   PID0x11 REAL NOT NULL,
54   PID0x45 REAL DEFAULT NULL,
55   PID0x51 REAL DEFAULT NULL,
56   PID0x5E REAL DEFAULT NULL,
57   PID0x61 REAL DEFAULT NULL,
58   PID0x62 REAL DEFAULT NULL,
59   PID0x63 REAL DEFAULT NULL,
60   PRIMARY KEY (session_id,id)
61 );
```

Appendix B

Query to collect data with overlapped time-frames

```
SELECT
  *
FROM
  --region accelerometer
  (
    (
      SELECT
        session_id,
        trunc(time_stamp/5)*5 "wind",
        avg(x) "avg(acc_x)", COALESCE(stddev_samp(x),0) "stddev(acc_x)",
        avg(y) "avg(acc_y)", COALESCE(stddev_samp(y),0) "stddev(acc_y)",
        avg(z) "avg(acc_z)", COALESCE(stddev_samp(z),0) "stddev(acc_z)"
      FROM
        (
          SELECT
            session_id,
            (time_stamp-min_ts)/1e9 time_stamp,
            x,
            y,
            z
          FROM
            accelerometer_data JOIN
            (
              SELECT id session_id, starttime*1e6 min_ts
              FROM session
            ) min_calc USING (session_id)
        ) AS a
      GROUP BY session_id, wind
    )
    UNION
    (
      SELECT
```

```
33          session_id,
34          trunc((time_stamp+1.25)/5)*5-1.25 wind,
35          avg(x) "avg(acc_x)", COALESCE(stddev_samp(x),0) "stddev(acc_x)",
36          avg(y) "avg(acc_y)", COALESCE(stddev_samp(y),0) "stddev(acc_y)",
37          avg(z) "avg(acc_z)", COALESCE(stddev_samp(z),0) "stddev(acc_z)"
38        FROM
39          (
40            SELECT
41              session_id,
42              --trunc(((time_stamp-min_ts)/1e9+1.25)/5)*5-1.25,
43              (time_stamp-min_ts)/1e9 time_stamp,
44              x,
45              y,
46              z
47            FROM
48              accelerometer_data JOIN
49              (
50                SELECT id session_id, starttime*1e6 min_ts
51                FROM session
52              ) min_calc USING (session_id)
53          ) AS a
54        WHERE trunc((time_stamp+1.25)/5)*5-1.25>=0
55        GROUP BY session_id, wind
56      )
57      UNION
58      (
59        SELECT
60          session_id,
61          trunc((time_stamp+3.75)/5)*5-3.75 wind,
62          avg(x) "avg(acc_x)", COALESCE(stddev_samp(x),0) "stddev(acc_x)",
63          avg(y) "avg(acc_y)", COALESCE(stddev_samp(y),0) "stddev(acc_y)",
64          avg(z) "avg(acc_z)", COALESCE(stddev_samp(z),0) "stddev(acc_z)"
65        FROM
66          (
67            SELECT
68              session_id,
69              --trunc(((time_stamp-min_ts)/1e9+3.75)/5)*5-3.75,
70              (time_stamp-min_ts)/1e9 time_stamp,
71              x,
72              y,
73              z
74            FROM
75              accelerometer_data JOIN
76              (
77                SELECT id session_id, starttime*1e6 min_ts
78                FROM session
79              ) min_calc USING (session_id)
80          ) AS a
81        WHERE trunc((time_stamp+3.75)/5)*5-3.75>=0
```

```
82        GROUP BY session_id, wind
83      )
84      UNION
85      (
86        SELECT
87          session_id,
88          (time_stamp/5)::INTEGER*5-2.5 wind,
89          avg(x) "avg(acc_x)", COALESCE(stddev_samp(x),0) "stddev(acc_x)",
90          avg(y) "avg(acc_y)", COALESCE(stddev_samp(y),0) "stddev(acc_y)",
91          avg(z) "avg(acc_z)", COALESCE(stddev_samp(z),0) "stddev(acc_z)"
92        FROM
93          (
94            SELECT
95              session_id,
96              (time_stamp - min_ts) / 1e9 time_stamp,
97              x,
98              y,
99              z
100           FROM
101             accelerometer_data
102             JOIN
103             (
104               SELECT id session_id, starttime*1e6 min_ts
105               FROM session
106             ) min_calc USING (session_id)
107         ) AS a
108       WHERE (time_stamp/5)::INTEGER*5-2.5>=0
109       GROUP BY session_id, wind
110     )
111   ) acc JOIN
112   --endregion
113   --region gyroscope
114   (
115     (
116       SELECT
117         session_id,
118         trunc(time_stamp/5)*5 "wind",
119         avg(x) "avg(gyr_x)", COALESCE(stddev_samp(x),0) "stddev(gyr_x)",
120         avg(y) "avg(gyr_y)", COALESCE(stddev_samp(y),0) "stddev(gyr_y)",
121         avg(z) "avg(gyr_z)", COALESCE(stddev_samp(z),0) "stddev(gyr_z)"
122       FROM
123         (
124           SELECT
125             session_id,
126             (time_stamp-min_ts)/1e9 time_stamp,
127             x,
128             y,
129             z
130           FROM
```

```
131             gyroscope_data JOIN
132             (
133               SELECT id session_id, starttime*1e6 min_ts
134               FROM session
135             ) min_calc USING (session_id)
136         ) AS a
137       GROUP BY session_id, wind
138     )
139     UNION
140     (
141       SELECT
142         session_id,
143         trunc((time_stamp+1.25)/5)*5-1.25 wind,
144         avg(x) "avg(gyr_x)", COALESCE(stddev_samp(x),0) "stddev(gyr_x)",
145         avg(y) "avg(gyr_y)", COALESCE(stddev_samp(y),0) "stddev(gyr_y)",
146         avg(z) "avg(gyr_z)", COALESCE(stddev_samp(z),0) "stddev(gyr_z)"
147       FROM
148         (
149           SELECT
150             session_id,
151             --trunc(((time_stamp-min_ts)/1e9+1.25)/5)*5-1.25,
152             (time_stamp-min_ts)/1e9 time_stamp,
153             x,
154             y,
155             z
156           FROM
157             gyroscope_data JOIN
158             (
159               SELECT id session_id, starttime*1e6 min_ts
160               FROM session
161             ) min_calc USING (session_id)
162         ) AS a
163       WHERE trunc((time_stamp+1.25)/5)*5-1.25>=0
164       GROUP BY session_id, wind
165     )
166     UNION
167     (
168       SELECT
169         session_id,
170         trunc((time_stamp+3.75)/5)*5-3.75 wind,
171         avg(x) "avg(gyr_x)", COALESCE(stddev_samp(x),0) "stddev(gyr_x)",
172         avg(y) "avg(gyr_y)", COALESCE(stddev_samp(y),0) "stddev(gyr_y)",
173         avg(z) "avg(gyr_z)", COALESCE(stddev_samp(z),0) "stddev(gyr_z)"
174       FROM
175         (
176           SELECT
177             session_id,
178             --trunc(((time_stamp-min_ts)/1e9+3.75)/5)*5-3.75,
179             (time_stamp-min_ts)/1e9 time_stamp,
```

```
180             x,
181             y,
182             z
183           FROM
184             gyroscope_data JOIN
185             (
186               SELECT id session_id, starttime*1e6 min_ts
187               FROM session
188             ) min_calc USING (session_id)
189         ) AS a
190       WHERE trunc((time_stamp+3.75)/5)*5-3.75>=0
191       GROUP BY session_id, wind
192     )
193     UNION
194     (
195       SELECT
196         session_id,
197         (time_stamp/5)::INTEGER*5-2.5 wind,
198         avg(x) "avg(gyr_x)", COALESCE(stddev_samp(x),0) "stddev(gyr_x)",
199         avg(y) "avg(gyr_y)", COALESCE(stddev_samp(y),0) "stddev(gyr_y)",
200         avg(z) "avg(gyr_z)", COALESCE(stddev_samp(z),0) "stddev(gyr_z)"
201       FROM
202         (
203           SELECT
204             session_id,
205             (time_stamp - min_ts) / 1e9 time_stamp,
206             x,
207             y,
208             z
209           FROM
210             gyroscope_data
211             JOIN
212             (
213               SELECT id session_id, starttime*1e6 min_ts
214               FROM session
215             ) min_calc USING (session_id)
216         ) AS a
217       WHERE (time_stamp/5)::INTEGER*5-2.5>=0
218       GROUP BY session_id, wind
219     )
220   ) gyr USING (session_id, wind) JOIN
221   --endregion
222   --region obd
223   (
224     (
225       SELECT
226         session_id,
227         trunc(time_stamp / 5) * 5 "wind",
228         avg(pid0x04)              "avg(pid0x04)",
```

```
229         COALESCE(stddev_samp(pid0x04),0)       "stddev(pid0x04)",
230         avg(pid0x05)                "avg(pid0x05)",
231         COALESCE(stddev_samp(pid0x05),0)       "stddev(pid0x05)",
232         avg(pid0x0C)                "avg(pid0x0C)",
233         COALESCE(stddev_samp(pid0x0C),0)       "stddev(pid0x0C)",
234         avg(pid0x0D)                "avg(pid0x0D)",
235         COALESCE(stddev_samp(pid0x0D),0)       "stddev(pid0x0D)",
236         avg(pid0x11)                "avg(pid0x11)",
237         COALESCE(stddev_samp(pid0x11),0)       "stddev(pid0x11)"
238       FROM
239         (
240           SELECT
241             session_id,
242             (time_stamp - min_ts) / 1e9 time_stamp,
243             pid0x04,
244             pid0x05,
245             pid0x0C,
246             pid0x0D,
247             pid0x11
248           FROM
249             obd_data
250             JOIN
251             (
252               SELECT
253                 id                session_id,
254                 starttime * 1e6 min_ts
255               FROM session
256             ) min_calc USING (session_id)
257         ) AS a
258       GROUP BY session_id, wind
259     )
260     UNION
261     (
262       SELECT
263         session_id,
264         (time_stamp/5)::INTEGER*5-2.5 wind,
265         avg(pid0x04) "avg(pid0x04)",COALESCE(stddev_samp(pid0x04),0) "stddev(
    pid0x04)",
266         avg(pid0x05) "avg(pid0x05)",COALESCE(stddev_samp(pid0x05),0) "stddev(
    pid0x05)",
267         avg(pid0x0C) "avg(pid0x0C)",COALESCE(stddev_samp(pid0x0C),0) "stddev(
    pid0x0C)",
268         avg(pid0x0D) "avg(pid0x0D)",COALESCE(stddev_samp(pid0x0D),0) "stddev(
    pid0x0D)",
269         avg(pid0x11) "avg(pid0x11)",COALESCE(stddev_samp(pid0x11),0) "stddev(
    pid0x11)"
270       FROM
271         (
272           SELECT
```

```
273             session_id,
274             (time_stamp-min_ts)/1e9 time_stamp,
275             pid0x04,
276             pid0x05,
277             pid0x0C,
278             pid0x0D,
279             pid0x11
280           FROM
281             obd_data JOIN
282             (
283               SELECT id session_id, starttime*1e6 min_ts
284               FROM session
285             ) min_calc USING (session_id)
286         ) AS a
287       WHERE (time_stamp/5)::INTEGER*5-2.5>=0
288       GROUP BY session_id, wind
289     )
290     UNION
291     (
292       SELECT
293         session_id,
294         trunc((time_stamp+1.25)/5)*5-1.25 wind,
295         avg(pid0x04) "avg(pid0x04)",COALESCE(stddev_samp(pid0x04),0) "stddev(
    pid0x04)",
296         avg(pid0x05) "avg(pid0x05)",COALESCE(stddev_samp(pid0x05),0) "stddev(
    pid0x05)",
297         avg(pid0x0C) "avg(pid0x0C)",COALESCE(stddev_samp(pid0x0C),0) "stddev(
    pid0x0C)",
298         avg(pid0x0D) "avg(pid0x0D)",COALESCE(stddev_samp(pid0x0D),0) "stddev(
    pid0x0D)",
299         avg(pid0x11) "avg(pid0x11)",COALESCE(stddev_samp(pid0x11),0) "stddev(
    pid0x11)"
300       FROM
301         (
302           SELECT
303             session_id,
304             (time_stamp-min_ts)/1e9 time_stamp,
305             pid0x04,
306             pid0x05,
307             pid0x0C,
308             pid0x0D,
309             pid0x11
310           FROM
311             obd_data JOIN
312             (
313               SELECT id session_id, starttime*1e6 min_ts
314               FROM session
315             ) min_calc USING (session_id)
316         ) AS a
```

```
317       WHERE trunc((time_stamp+1.25)/5)*5-1.25>=0
318       GROUP BY session_id, wind
319     )
320     UNION
321     (
322       SELECT
323         session_id,
324         trunc((time_stamp+3.75)/5)*5-3.75 wind,
325         avg(pid0x04) "avg(pid0x04)",COALESCE(stddev_samp(pid0x04),0) "stddev(
    pid0x04)",
326         avg(pid0x05) "avg(pid0x05)",COALESCE(stddev_samp(pid0x05),0) "stddev(
    pid0x05)",
327         avg(pid0x0C) "avg(pid0x0C)",COALESCE(stddev_samp(pid0x0C),0) "stddev(
    pid0x0C)",
328         avg(pid0x0D) "avg(pid0x0D)",COALESCE(stddev_samp(pid0x0D),0) "stddev(
    pid0x0D)",
329         avg(pid0x11) "avg(pid0x11)",COALESCE(stddev_samp(pid0x11),0) "stddev(
    pid0x11)"
330       FROM
331         (
332           SELECT
333             session_id,
334             (time_stamp-min_ts)/1e9 time_stamp,
335             pid0x04,
336             pid0x05,
337             pid0x0C,
338             pid0x0D,
339             pid0x11
340           FROM
341             obd_data JOIN
342             (
343               SELECT id session_id, starttime*1e6 min_ts
344               FROM session
345             ) min_calc USING (session_id)
346         ) AS a
347       WHERE trunc((time_stamp+3.75)/5)*5-3.75>=0
348       GROUP BY session_id, wind
349     )
350   ) obd USING (session_id, wind) LEFT JOIN
351   --endregion
352   --region location
353   (
354     (
355       SELECT
356         session_id,
357         trunc(time_stamp / 5) * 5      "wind",
358         avg(lat)                       "avg(lat)", COALESCE(stddev_samp(lat), 0)
     "stddev(lat)",
```

```
359         avg(lon)                        "avg(lon)", COALESCE(stddev_samp(lon), 0)
     "stddev(lon)",
360         avg(alt)                        "avg(alt)", stddev_samp(alt)
     "stddev(alt)"
361       FROM
362         (
363           SELECT
364             session_id,
365             (time_stamp - min_ts) / 1e9 time_stamp, --to seconds since session
    start
366             lat,
367             lon,
368             alt
369           FROM
370             gps_data
371             JOIN
372             (
373               SELECT
374                 id                session_id,
375                 starttime * 1e6 min_ts
376               FROM session
377             ) min_calc USING (session_id)
378         ) AS a
379       GROUP BY session_id, wind
380       HAVING stddev_samp(alt) IS NOT NULL --discard data without height
381     )
382     UNION
383     (
384       SELECT
385         session_id,
386         (time_stamp/5)::INTEGER*5-2.5 wind,
387         avg(lat)                        "avg(lat)", COALESCE(stddev_samp(lat), 0)
     "stddev(lat)",
388         avg(lon)                        "avg(lon)", COALESCE(stddev_samp(lon), 0)
     "stddev(lon)",
389         avg(alt)                        "avg(alt)", stddev_samp(alt)
     "stddev(alt)"
390       FROM
391         (
392           SELECT
393             session_id,
394             (time_stamp - min_ts) / 1e9 time_stamp, --to seconds since session
    start
395             lat,
396             lon,
397             alt
398           FROM
399             gps_data
400             JOIN
```

```
401             (
402               SELECT
403                 id                session_id,
404                 starttime * 1e6 min_ts
405               FROM session
406             ) min_calc USING (session_id)
407         ) AS a
408       WHERE (time_stamp/5)::INTEGER*5-2.5>=0
409       GROUP BY session_id, wind
410       HAVING stddev_samp(alt) IS NOT NULL --discard data without height
411     )
412     UNION
413     (
414       SELECT
415         session_id,
416         trunc((time_stamp+1.25)/5)*5-1.25 wind,
417         avg(lat)                          "avg(lat)", COALESCE(stddev_samp(lat), 0)
     "stddev(lat)",
418         avg(lon)                          "avg(lon)", COALESCE(stddev_samp(lon), 0)
     "stddev(lon)",
419         avg(alt)                          "avg(alt)", stddev_samp(alt)
     "stddev(alt)"
420       FROM
421         (
422           SELECT
423             session_id,
424             (time_stamp - min_ts) / 1e9 time_stamp, --to seconds since session
    start
425             lat,
426             lon,
427             alt
428           FROM
429             gps_data
430             JOIN
431             (
432               SELECT
433                 id                session_id,
434                 starttime * 1e6 min_ts
435               FROM session
436             ) min_calc USING (session_id)
437         ) AS a
438       WHERE trunc((time_stamp+1.25)/5)*5-1.25>=0
439       GROUP BY session_id, wind
440       HAVING stddev_samp(alt) IS NOT NULL --discard data without height
441     )
442     UNION
443     (
444       SELECT
445         session_id,
```

```
446         trunc((time_stamp+3.75)/5)*5-3.75 wind,
447         avg(lat)                          "avg(lat)", COALESCE(stddev_samp(lat), 0)
      "stddev(lat)",
448         avg(lon)                          "avg(lon)", COALESCE(stddev_samp(lon), 0)
      "stddev(lon)",
449         avg(alt)                          "avg(alt)", stddev_samp(alt)
      "stddev(alt)"
450       FROM
451         (
452           SELECT
453             session_id,
454             (time_stamp - min_ts) / 1e9 time_stamp, --to seconds since session
     start
455             lat,
456             lon,
457             alt
458           FROM
459             gps_data
460             JOIN
461             (
462               SELECT
463                 id                session_id,
464                 starttime * 1e6 min_ts
465               FROM session
466             ) min_calc USING (session_id)
467         ) AS a
468       WHERE trunc((time_stamp+3.75)/5)*5-3.75>=0
469       GROUP BY session_id, wind
470       HAVING stddev_samp(alt) IS NOT NULL --discard data without height
471     )
472   ) gps USING (session_id, wind)
473 --endregion
474 WHERE session_id = 57 --change to the intended session or keep like this
475 ORDER BY session_id, wind;
```

References

[1] Google. *SensorEvent | Android Developers*. Google. https://developer.android.com/reference/android/hardware/SensorEvent.

[2] Society of Automotive Engineers. SAE J1979 - E/E Diagnostic Test Modes. Technical report, Society of Automotive Engineers, December 1991. https://www.sae.org/standards/content/j1979_199112/.

[3] Air safety statistics in the EU. Under "General Aviation" subtitle http://ec.europa.eu/eurostat/statistics-explained/index.php?title=Air_safety_statistics_in_the_EU.

[4] Road accident fatalities. http://ec.europa.eu/eurostat/statistics-explained/index.php?title=Road_accident_fatalities_-_statistics_by_type_of_vehicle.

[5] Rui Araujo, Angela Igreja, Ricardo de Castro, and Rui Esteves Araujo. Driving coach: A smartphone application to evaluate driving efficient patterns. In *2012 IEEE Intelligent Vehicles Symposium*. IEEE, jun 2012. https://ieeexplore.ieee.org/abstract/document/6232304/.

[6] ELM Electronics. *ELM327 - OBD to RS232 Interpreter*, July 2016. https://www.elmelectronics.com/wp-content/uploads/2016/07/ELM327DS.pdf.

[7] International Organization for Standardization. ISO 9141:1989 - Road vehicles — Diagnostic systems — Requirements for interchange of digital information. techreport, International Organization for Standardization, 1989. https://www.iso.org/obp/ui/#iso:std:iso:9141:ed-1:v1:en.

[8] International Organization for Standardization. ISO 9141-2:1994(en) - Road vehicles — Diagnostic systems — Part 2: CARB requirements for interchange of digital information. Technical report, International Organization for Standardization, 1994.

[9] Society of Automotive Engineers. SAE J1962 - OBD II Scan Tool. Technical report, Society of Automotive Engineers, June 1992. https://www.sae.org/standards/content/j1962_199206/.

[10] Society of Automotive Engineers. SAE J1978 - Diagnostic Connector Equivalent. Technical report, Society of Automotive Engineers, March 1992. https://www.sae.org/standards/content/j1978_199203/.

[11] Society of Automotive Engineers. SAE J2012 - Diagnostic Trouble Code Definitions. Technical report, Society of Automotive Engineers, March 1992. https://www.sae.org/standards/content/j2012_199203/.

[12] European Telecommunications Standards Institute. ETSI 07.10 - Digital cellular telecommunications system; Terminal Equipment to Mobile Station (TE-MS) multiplexer protocol. Technical report, European Telecommunications Standards Institute, 1998. http://www.etsi.org/deliver/etsi_ts/101300_101399/101369/07.01.00_60/ts_101369v070100p.pdf.

[13] Google. *FusedLocationProviderClient | Google APIs for Android*. Google. https://developers.google.com/android/reference/com/google/android/gms/location/FusedLocationProviderClient.

[14] Colin Shearer. The crisp-dm model: the new blueprint for data mining. *Journal of data warehousing*, 5(4):13–22, 2000.

[15] João Mendes Moreira and José Luís Borges. Knowledge Extraction and Machine Learning - Data Preparation, 2011. https://paginas.fe.up.pt/~ec/files_1112/week_03_Data_Preparation.pdf.

[16] Mariana Fernandez Afonso. Exploring Visual Content in Social Networks. mathesis, Faculty of Engineering of the University of Porto, July 2015.

[17] Oracle. *Data Mining Concepts - Classification*. Oracle. https://docs.oracle.com/cd/B28359_01/datamine.111/b28129/classify.htm#DMCON004.

[18] D.P. Kingma and L.J. Ba. Adam: A method for stochastic optimization. In *ICLR 2015*, 2015. http://hdl.handle.net/11245/1.505367.

[19] Roy T. Fielding and Richard N. Taylor. Principled design of the modern web architecture. In *Principled design of the modern web architecture*, pages 407–416, 2000. cited By 174.

[20] Google. *Using OAuth 2.0 to Access Google APIs | Google Identity Platform | Google Developers*. Google. https://developers.google.com/identity/protocols/OAuth2.

[21] Roy T. Fielding. Fielding Dissertation: CHAPTER 5: Representational State Transfer (REST). https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm.